Advanced Configuration Security Usage Guide for Nexus Platform

Technical Note

FPGA-TN-02176-1.5

June 2021

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AES	Advanced Encryption Standard
ECC	Error Correction Code
ECDSA	Elliptic Curve Digital Signature Algorithm
HMAC	Hash-based Message Authentication Code
FPGA	Field Programmable Gate Array
OTP	One-Time Programmable
SHA	Secure Hash Algorithm
TMR	Triple Modular Redundancy

1. Introduction

Security is a major concern in FPGA applications for securing both the FPGA bitstream data (Customer Intelligent Property) and the FPGA device functionality (Customer System Integrity and Safety). The Lattice Nexus™ Platform, which includes the CrossLink™-NX, Certus™-NX, and CertusPro™-NX devices, provides a full suite of enhanced security algorithms and features:

- Password protection (128 bits) to protect the device configuration
- 256 bit Advanced Encryption Security (AES-256) for bitstream encryption
- Hash-based Message Authentication Code (HMAC with SHA-256) for bitstream authentication
- Elliptic Curve Digital Signature Algorithm (ECDSA) for bitstream authentication

With these enhanced security functions, devices built on the Nexus platform can meet a wide variety of customer security requirements in a complex system.

This technical note discusses how to access the enhanced security features for the Nexus platform using Lattice Radiant® Software.

Note: The enhanced security features for the Nexus device families are only available when using Lattice Radiant software Version 2.1 SP1 or later.

2. Overview

The Nexus platform provides a full suite of enhanced security features, which include:

- Password Protection
- Bitstream AES-CBC Encryption
- Bitstream HMAC Authentication
- Bitstream ECDSA Authentication

Some of these enhanced security features can be used individually. Most features, however, are combined to achieve optimal results. The typical uses of the enhanced security features are summarized in Table 2.1 below.

Table 2.1. Nexus Enhanced Security Settings

Security Protocol	Password 128 Bits	AES-256	HMAC-SHA256	ECDSA	Bitstream Format
Password Protection	Yes	—	—	—	Plain bitstream
AES-CBC Encryption	Optional	Yes	—	—	Encrypted bitstream
HMAC Authentication	Optional	Yes	Yes	—	Encrypted (bitstream + HMAC-SHA256 Signature)
ECDSA Authentication	Optional	—	—	Yes	Plain bitstream + ECDSA Signature
ECDSA Authentication with AES Encryption	Optional	Yes	—	Yes	Encrypted (bitstream + ECDSA Signature)

The on-chip non-volatile eFUSE memories are utilized to store the customer specified security setup, such as Password, AES encryption key, ECDSA (Elliptic Curve Digital Signature Algorithm) public key, and others. The user security setup is downloaded into shadow registers inside configuration logic during the device initialization process. To maximize the device security, all device security features are enabled (preset) at power up and selectively released based on user setup stored inside the on-chip eFUSE memory during the device initialization process. The initialization process immediately after power up is executed unconditionally before the external slave configuration port can be activated.

A set of non-volatile eFUSE memories needs to be set to select the authentication mode. There are three authentication modes:

- No Authentication
- ECDSA Authentication
- HMAC-SHA256 Authentication

2.1. Password Protection

Devices built on the Nexus platform offer 128-bit password protection for device configuration access, which you can optionally enable. Once enabled, the configuration access is locked and all configuration memory (volatile and non-volatile) commands are nullified. The 128-bit user specified password code is required to unlock the device configuration access using the LSC_ SHIFT_PASSWORD command. When the configuration task is completed, the ISC_DISABLE command execution brings the device back to locked state.

2.2. Bitstream AES-CBC Encryption

Devices built on the Nexus platform offer optimal 256-bit Advanced Encryption Standard (AES) to protect the bitstream. AES-CBC Encryption provides a mechanism to prevent reverse engineering of customer intellectual property. The 256-bit key is user specified, stored in the built-in non-volatile eFUSE memory for bitstream decryption. Hence, no special voltages are required to maintain the key within the FPGA. AES bitstream protection is available in all versions of the Nexus platform devices. Figure 2.1 shows the process of bitstream encryption and decryption supported by these devices. Note: The AES-CBC Initialization Vector is uniquely generated automatically by Lattice's software tools and not customer modifiable.

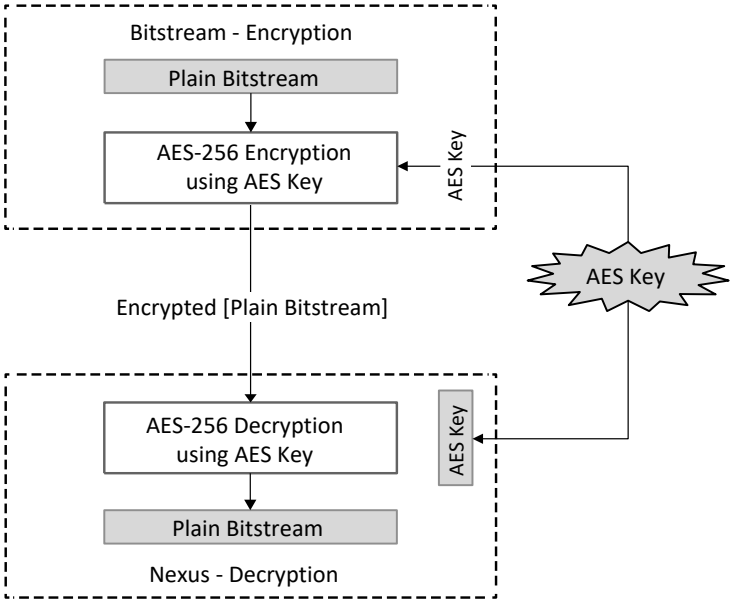

Figure 2.1. Nexus Bitstream Encryption and Decryption

2.3. Bitstream HMAC Authentication

The Hash-based Message Authentication Code is a specific type of message authentication code involving a cryptographic hash function and a secret cryptographic key (HMAC Key). The signature generated by the HMAC-SHA256, along with the HMAC key and the bitstream are encrypted with the device's AES key. The AES bitstream encryption is automatically enabled if the HMAC authentication is selected. Figure 2.2 shows the process of bitstream HMAC authentication.

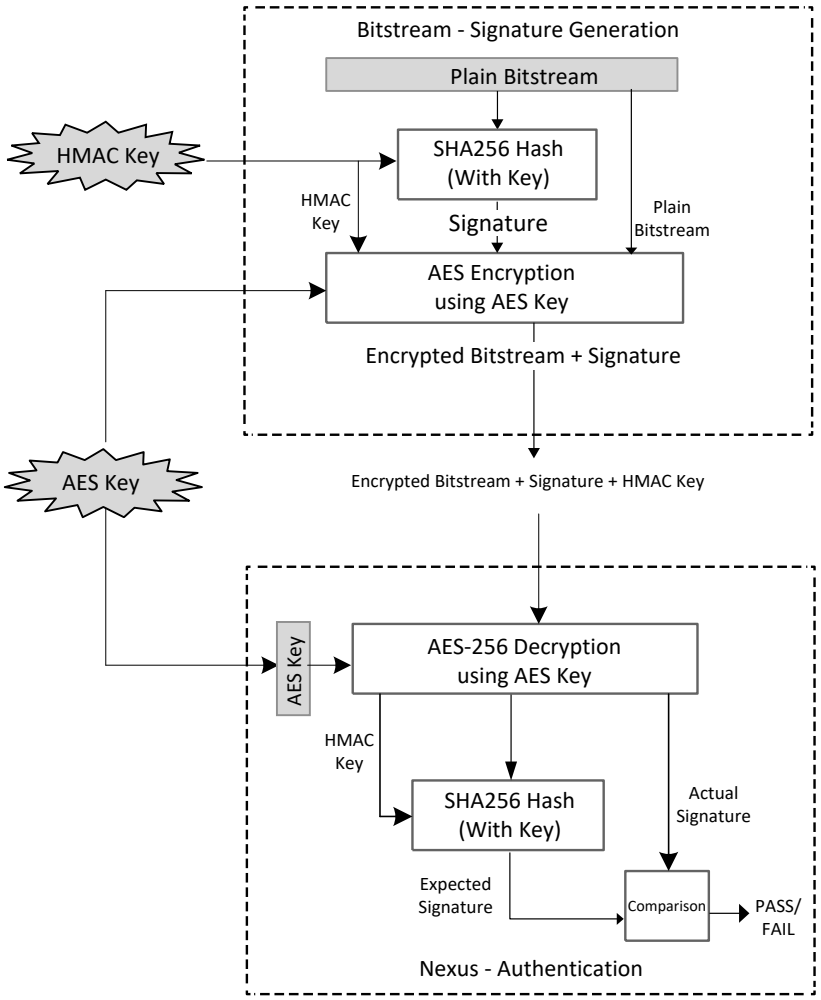

Figure 2.2. Nexus Bitstream HMAC Authentication

2.4. Bitstream ECDSA Authentication

The Elliptic Curve Digital Signature Algorithm (ECDSA) offers a variant of the Digital Signature Algorithm (DSA) that uses elliptic curve cryptography. The private or public key pair is used to generate a digital signature for the bitstream using ECDSA algorithm. Nexus platform devices optionally support the AES encryption combined with ECDSA authentication. Figure 2.3 shows the process of bitstream ECDSA authentication.

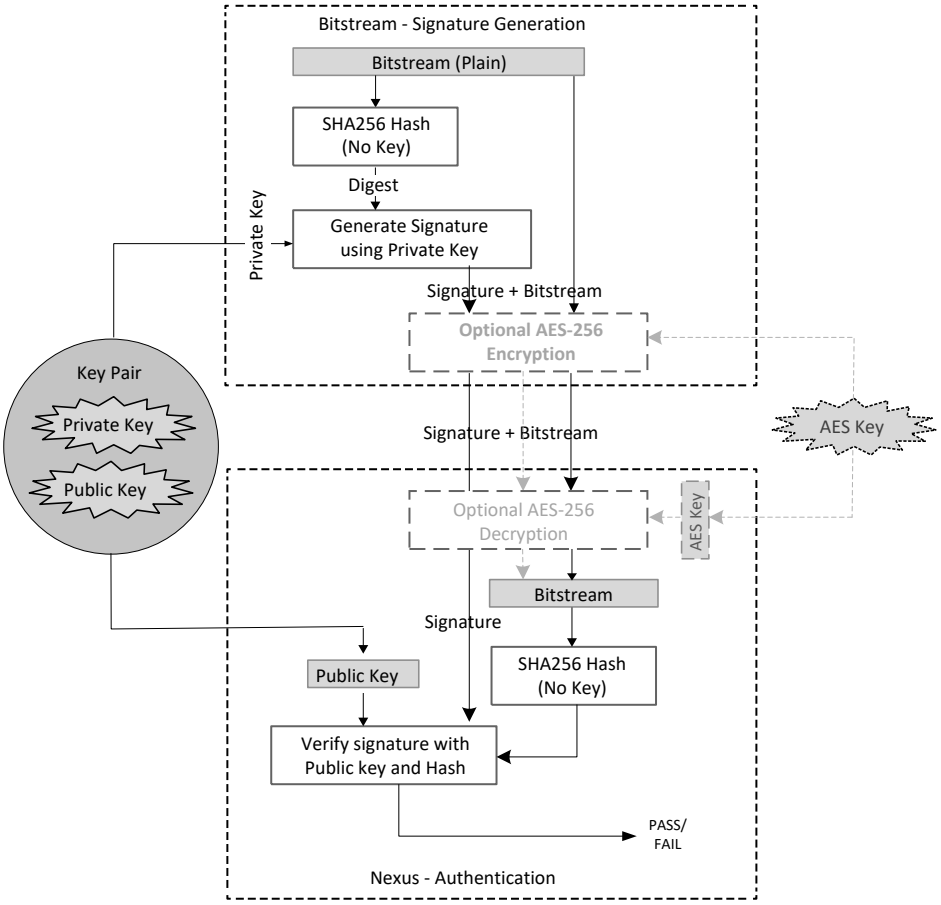

Figure 2.3. Nexus Bitstream ECDSA Authentication

2.5. Method for Creating an Encrypted Bitstream File and Key File

The enhanced security feature must be enabled to create a key file (.bek). The encrypted key file is generated using the Lattice Radiant design software. There are two ways to create an encrypted bitstream file:

- In the Lattice Radiant design software, configure security settings using the Security Settings Tool and then generate an encrypted bitstream from within the Process pane.
- In the Deployment Tool, specify security settings and then convert an unencrypted bitstream file (which was created in Lattice Radiant) into an encrypted bitstream file.

2.6. Setting Security and Encryption for FPGA Devices

Please note the following difference between using the Deployment Tool and the Security Settings Tool in creating an encrypted bitstream and key files:

When using the Security Settings Tool, Lattice Radiant will output an encrypted bitstream and key file.

When using the Deployment Tool, Lattice Radiant will output an unencrypted bitstream file. This file will then be encrypted using the Deployment Tool. Care should be taken to protect the unencrytped bitstream if using this process. Also, care must be taken to protect the key file when either method is used.

3. Enhanced Security Setup Using Lattice Radiant Software

Lattice Radiant Software provides the Security Settings Tool and Process Flow for setting up the enhanced security features.

To create the Encryption Key and Encrypted Bitstream:

1. In the **File List** pane, double-click **Strategy**.
2. The **Strategies** dialog box opens. Select **Bitstream** and set the required bitstream options.
3. Click **OK**.

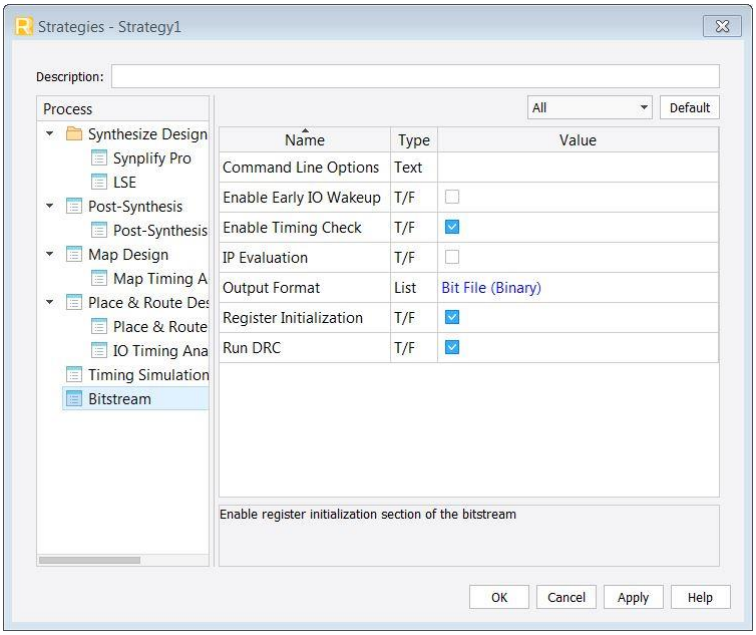

Figure 3.1. Bitstream Setup

4. In the main interface, select Tools > Bitstream Security Settings.

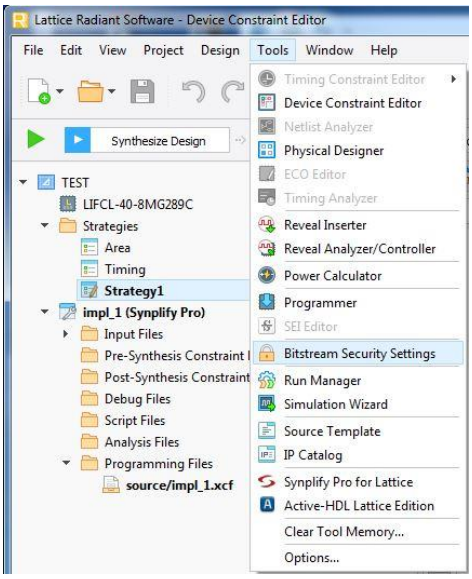

Figure 3.2. Bitstream Security Settings Tool in Radiant Software

5. The **Enter Password** dialog box opens with LATTICESEMI as the default password. Click **Change Password** to create a unique password for the encryption key file.

Figure 3.3. Key File Password Setup

6. In the **Change Password** dialog box, enter the password, which must be between eight and 16 characters.

Figure 3.4. Change Key File Password

7. Verify the password and click **OK**. The **Bitstream Security Settings** dialog box opens and provides the options for enhanced security feature setup.

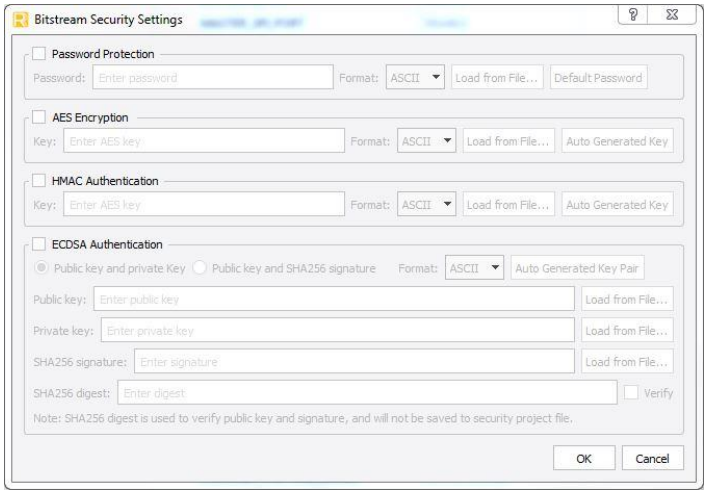

Figure 3.5. Bitstream Security Setup User Interface

3.1. Enabling Password Protection

To enable password protection:

1. Select the **Password Protection** check box as shown in Figure 3.6.
2. Click the **Format** down arrow and select the password format:
   a. **ASCII** – Alphanumeric values, using up to 16 characters
   b. **Hex** – Values of 0 through F, using up to 32 characters
   c. **Binary** – Values of 0 and 1, using up to 128 characters
3. Enter the desired password in the correct format in the **Password** field. You can also load a previously created password from an existing key file (.key). A password is required to open the key file.

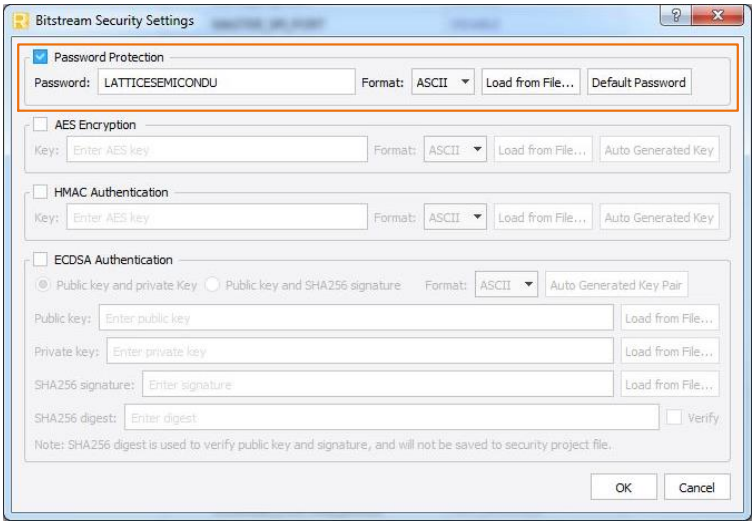

Figure 3.6. Setup Password Protection

4. Click **OK** after the setup is completed to create the key file (.bek). The default key file directory is *<Project Name>/security_setting/<Project Name>.key*.

3.2. Enabling Bitstream AES Encryption

To enable AES encryption:

1. Select the **AES Encryption** check box as shown in Figure 3.7.

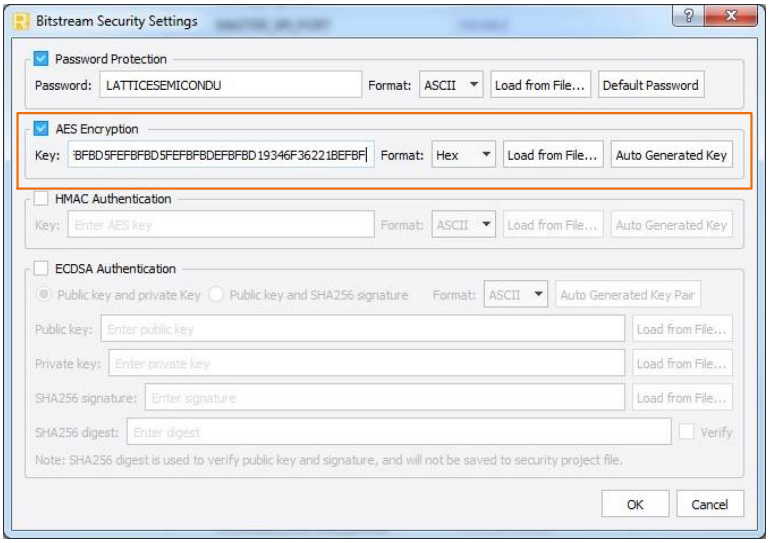

Figure 3.7. Setup AES Encryption

2. Click the **Format** down arrow and select the key format:
   a. **ASCII** – Alphanumeric values, using up to 32 characters
   b. **Hex** – Values of 0 through F, using up to 64 characters
   c. **Binary** – Values of 0 and 1, using up to 256 characters
3. Enter the key string in correct format in the **Key** field. The IV will be automatically generated.
4. Click the **Auto Generated Key** button to automatically fill out the Key string if you want to use the automatic key generation feature. You can also load a previously created AES key from an existing security project file (.bek). A password is required to open the file.
5. Click **OK** after the setup is completed to create the key file (.bek). The default key file directory is *<Project Name>/security_setting/<Project Name>.bek*.

3.3. Enabling Bitstream HMAC Authentication

To enable bitstream HMAC authentication:

1. Select the **HMAC Authentication** check box as shown in Figure 3.8.

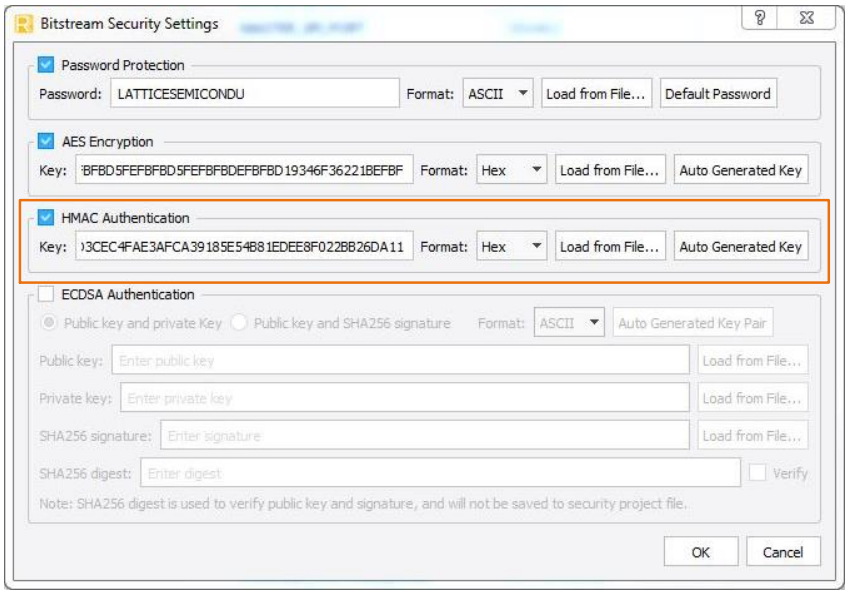

Figure 3.8. Setup Bitstream HMAC Authentication

2. Click the **Format** down arrow and select the HMAC key format:
   a. **ASCII** – Alphanumeric values, using up to 32 characters
   b. **Hex** – Values of 0 through F, using up to 64 characters
   c. **Binary** – Values of 0 and 1, using up to 256 characters
3. Input the key string in the correct format in the **Key** field.
4. Click the **Auto Generated Key** button to automatically fill out the Key string if you want to use the automatic key generation feature. You can also load a previously created HMAC key from an existing security project file (.prv). A password is required to open the existing security project.
5. If the AES Encryption is not yet enabled, set up the AES Encryption Key as described in the Enabling Bitstream AES Encryption section.
6. Click **OK** after the setup is completed to create the key file (.prv). The default key file directory is *<Project Name>/security_setting/<Project Nambe>.prv*.

3.4. Enabling Bitstream ECDSA Authentication

There are two ways to set up ECDSA authentication from the user interface:

- Public Key and Private Key

  Select the **Public key and private key** option. To automatically generate the public key and private key in Hex format, click the **Auto Generated Key Pair** button.

  You can also load a previously created public key from an existing security project file (.pub) and a private key from an existing security project file (.prv). A password is required to open the security project files (.pub and .prv). Password is required to open the existing security project.

- Public Key and SHA256 signature

  Select the **Public key and SHA256 signature** option. Input the public key string and SHA256 signature in the correct format inside the corresponding fields. You can load a previously created pubic key from an existing security project file (.pub) and the SHA256 signature from an existing security project file (.sig). A password is required to open the security project files (.pub and .sig) and the existing security project.

  For verification, you can select the **Verify** check box and input the SHA256 digest in the corresponding field. The SHA256 digest is used to verify public key and signature, which are not saved to the security project file.

To enable bitstream ECDSA authentication:

1. Select the **ECDSA Authentication** check box as shown in Figure 3.9.

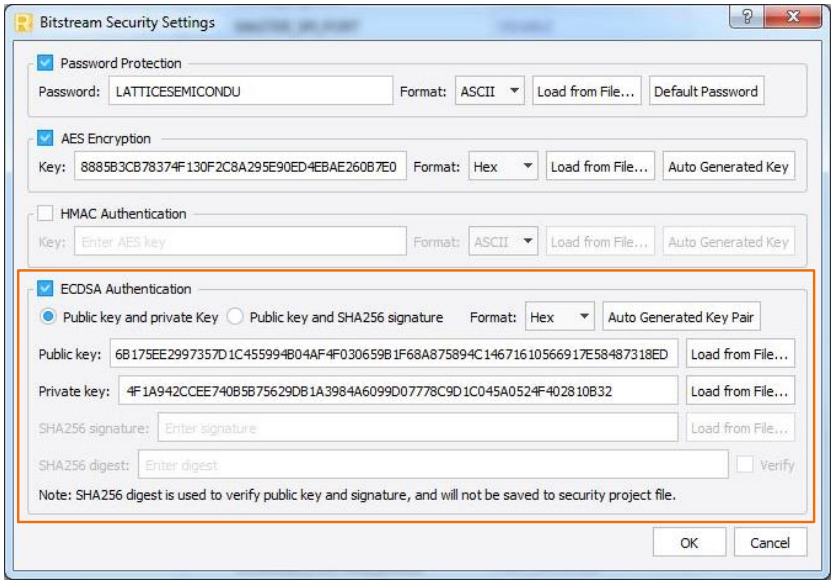

Figure 3.9. Setup Bitstream ECDSA Authentication

2. Click the **Format** down arrow and select the HMAC Key format:
   a. **ASCII** – Alphanumeric values, using up to 64 characters
   b. **Hex** – Values of 0 through F, using up to 128 characters
   c. **Binary** – Values of 0 and 1, using up to 512 characters
3. If the AES Encryption is required, set up the **AES Encryption Key** field as described in the Enabling Bitstream AES Encryption section.
4. Click **OK** after the setup is completed to create the key file (.bek). The default key file directory is *<Project Name>/security_setting/<Project Nambe>.pub (.prv or .sig)*.

4. Setting Security and Encryption Using the Deployment Tool

The Deployment Tool is a standalone tool that allows you to generate files for deployment for a single device or for multiple devices. It is also used to convert data files to other formats. The Deployment Tool is installed along with the Lattice Radiant Programmer, which can be included in the Lattice Radiant Software installation. The tool can also be installed separately. The Deployment Tool is launched from the Lattice Radiant Programmer.

The Lattice Radiant Programmer can be launched from:

- Microsoft Windows: Start > Programs > Lattice Radiant Software > Accessories > Lattice Radiant Programmer
- Lattice Radiant Software: Tools > Programmer

To set security and encryption using the Deployment Tool:

1. Open the Deployment Tool from Lattice Radiant Programmer by selecting **Tools > Deployment Tool**.
2. The Getting Started dialog box appears. Choose Create New Deployment.

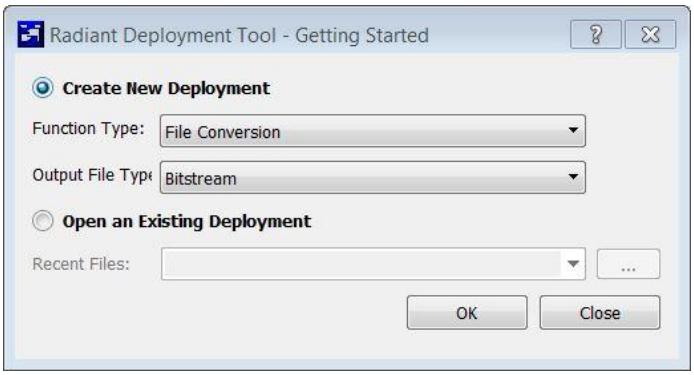

Figure 4.1. Deployment Tool Project Setup

3. In Function Type, select File Conversion.
4. In Output File Type, select Bitstream.
5. Click **OK**.
6. The **File Conversion** (Step 1) window is displayed. Click on the blank **File Name** column.
7. Click on the browse (…) button in the **File Name** column and select the unencrypted .bit file created by double-clicking on the Bitstream file in the **Lattice Radiant Process** window.

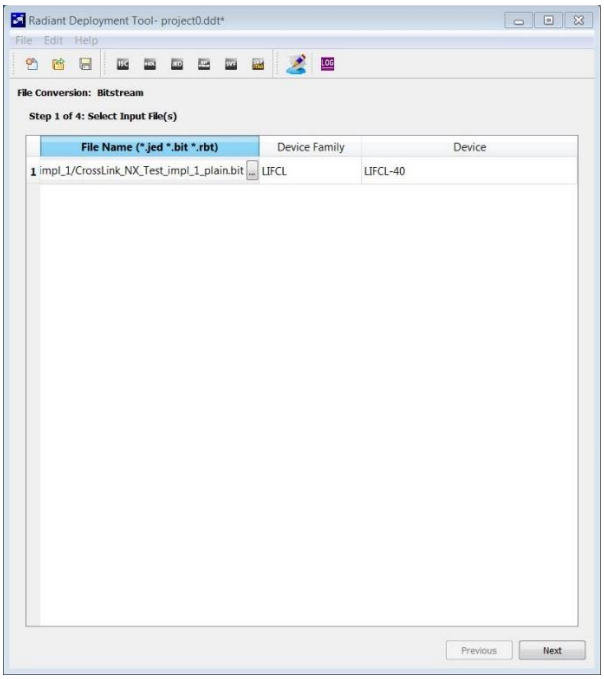

Figure 4.2. Deployment Tool Input File Setup

8. Select **Next** to go to the Step 2 window. Select the **Output Format**.

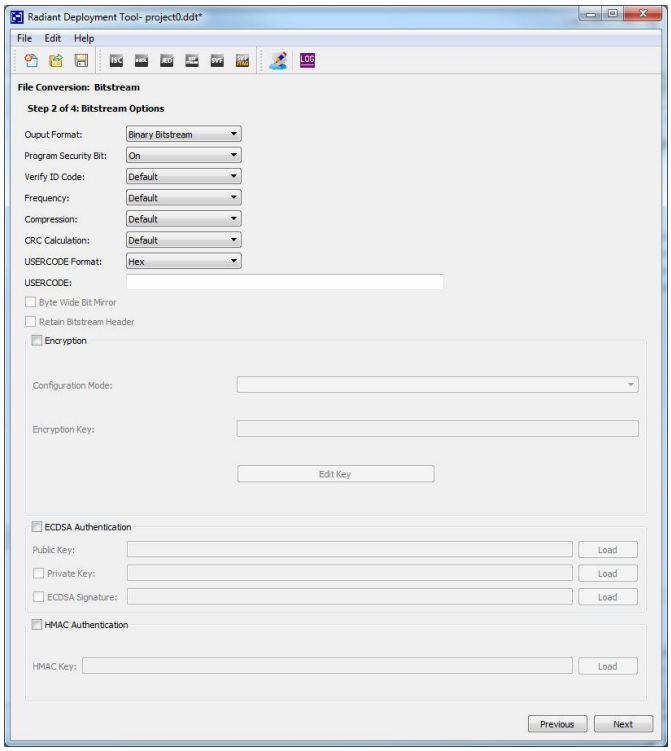

Figure 4.3. Deployment Tool Output Setup

9. To burn the security fuses to disable the ability to read back the bitstream from the SRAM in the FPGA, set the **Program Security Bit** to **On**.

10. To store device data such as firmware version number, manufacturer ID, programming date, programmer make, pattern code, etc., set a USERCODE. Select the **USERCODE Format** and then enter a **USERCODE**:
    a. **ASCII** – Alphanumeric values, using up to16 characters
    b. **Hex** – Values of 0 through F, using up to 32 characters
    c. **Binary** – Values of 0 and 1, using up to 128 characters
11. To create an encrypted bitstream from an unencrypted bitstream, select **Encryption**. Since the Nexus devices do not have different padding bits for different modes, The **Configuration Mode** option is no longer valid nor necessary.
12. Other file conversion options to choose from are:
    a. **Verify ID Code** – For bitstream debugging, inserts the verify device 32-bit JTAG IDCODE frame into the bitstream. The setting Default means do not override the bitstream. ON (insert) or OFF (omit) overrides the current setting in the bitstream. It is recommended to leave this as default.
    b. **Frequency** – Used to adjust the master clock configuration frequency in the bitstream for the two master modes, SPI and SPIm mode. The setting has no effect on the Slave modes. The setting of Default means keep the bitstream setting. Selections other than Default overrides the current setting in the bitstream.
    c. **Compression** – Used to compress the bitstream. Default means do not change the bitstream. ON (compress) or OFF (no compress) overrides the bitstream. It is recommended not to use compression with encryption. Compression not available on LatticeECP3 devices.
    d. **CRC Calculation** – Disables the frame-by-frame CRC for bitstream debugging. It is recommended to keep the default of the bitstream for maximum configuration reliability.
13. To enter the encryption key, click **Edit Key**. A Windows Explorer window opens and prompts you to load the Encryption file (.bek). Select the Encryption Key File (<Project Name>.bek) generated from the Lattice Radiant Bitstream Security Settings as described in Section 5. The dialog box will appears asking for Encryption File Password:

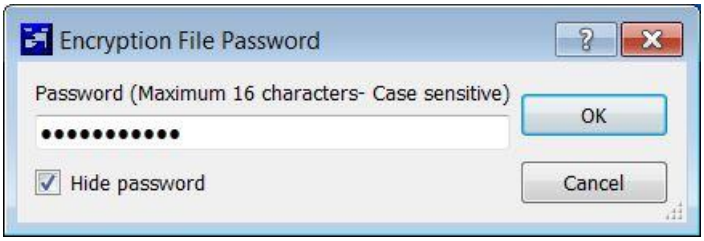

Figure 4.4. Password Input for Encrypt Key File

14. Enter the Encryption File Password into the prompt, to view what is being typed, un-check the Hide Password check box, then select **OK**. The Encryption Key field is auto filled with the key string from the Encryption Key file.
15. Setup for bitstream authentication
    a. To setup bitstream ECDSA authentication, check the **ECDSA Authentication** check box; Finish the setup from existing security project by pressing the **Load** buttons to load the **Public Key** from the .pub file, and load **Private Key** form the .prv file or load the **ECDSA Signature** from the .sig file. Password is required to open the existing security project.
    b. To setup bitstream HMAC authentication, check **HMAC Authentication** check box; Finish the setup from existing security project by pressing the **Load** button to load the **HMAC Key** from the .prv file. Password is required to open the existing security project.
16. Click **Next**.

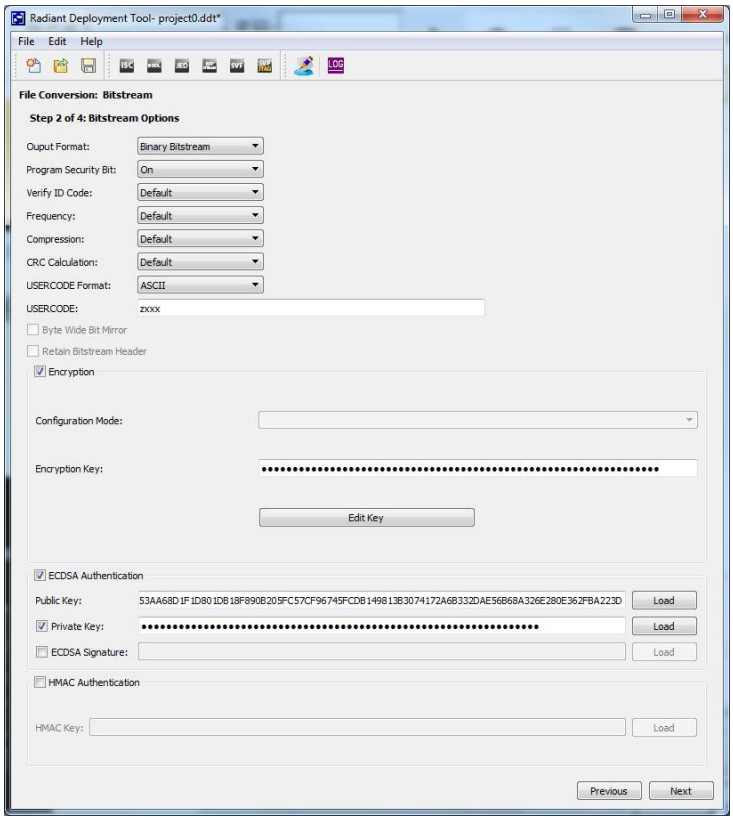

Figure 4.5. Deployment Tool Output Setup with AES Encryption and ECDSA Authentication

17. Select or enter an Output file name for the encrypted bitstream and click **Next** to proceed.

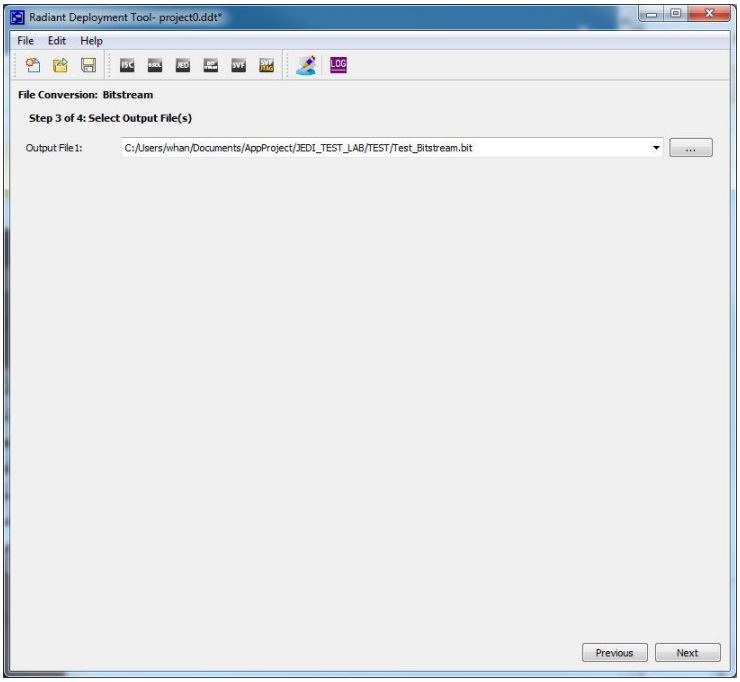

Figure 4.6. Deployment Tool Output File Setup

18. A Deployment Tool Summary appears. Click **Generate** to create the encrypted bitstream. An example of the summary and messages is shown below:

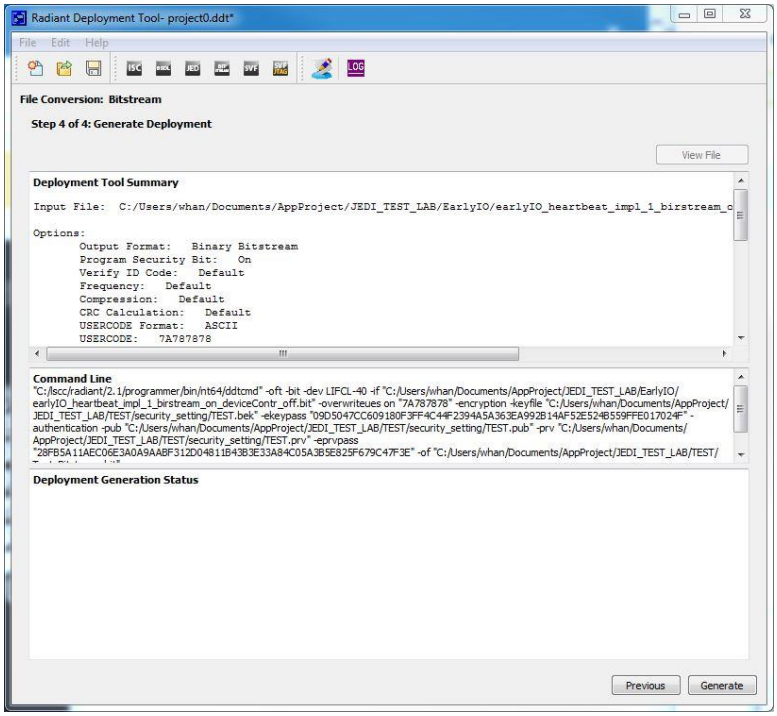

Figure 4.7. Lattice Radiant Deployment Tool Summary

You are now ready to download the encrypted bitstream using Lattice Radiant Programmer.

5. Programming Enhanced Security Setup

To program the key code onto the OTP fuses in the FPGA, use Lattice Radiant Programmer. Since Programmer is used primarily in the board design prototype development phase, you may choose to program the key lock fuse while programming the encryption key code. The purpose of the key lock fuse is to disable the reading of the fuse state of all the OTP fuses, which provides the key code as the first line of defense. Therefore, the only method to verify that the key code is correct is to configure the device with an encrypted bitstream.

5.1. Programming Password

To program the Password using Lattice Radiant Programmer:

1. Be sure the board with the Lattice FPGA is properly connected to your computer and turned on.
2. Open Lattice Radiant Programmer:
   - In the **Lattice Radiant** window, select **Tools > Programmer** or click the **Programmer** icon from the Lattice Radiant toolbar.

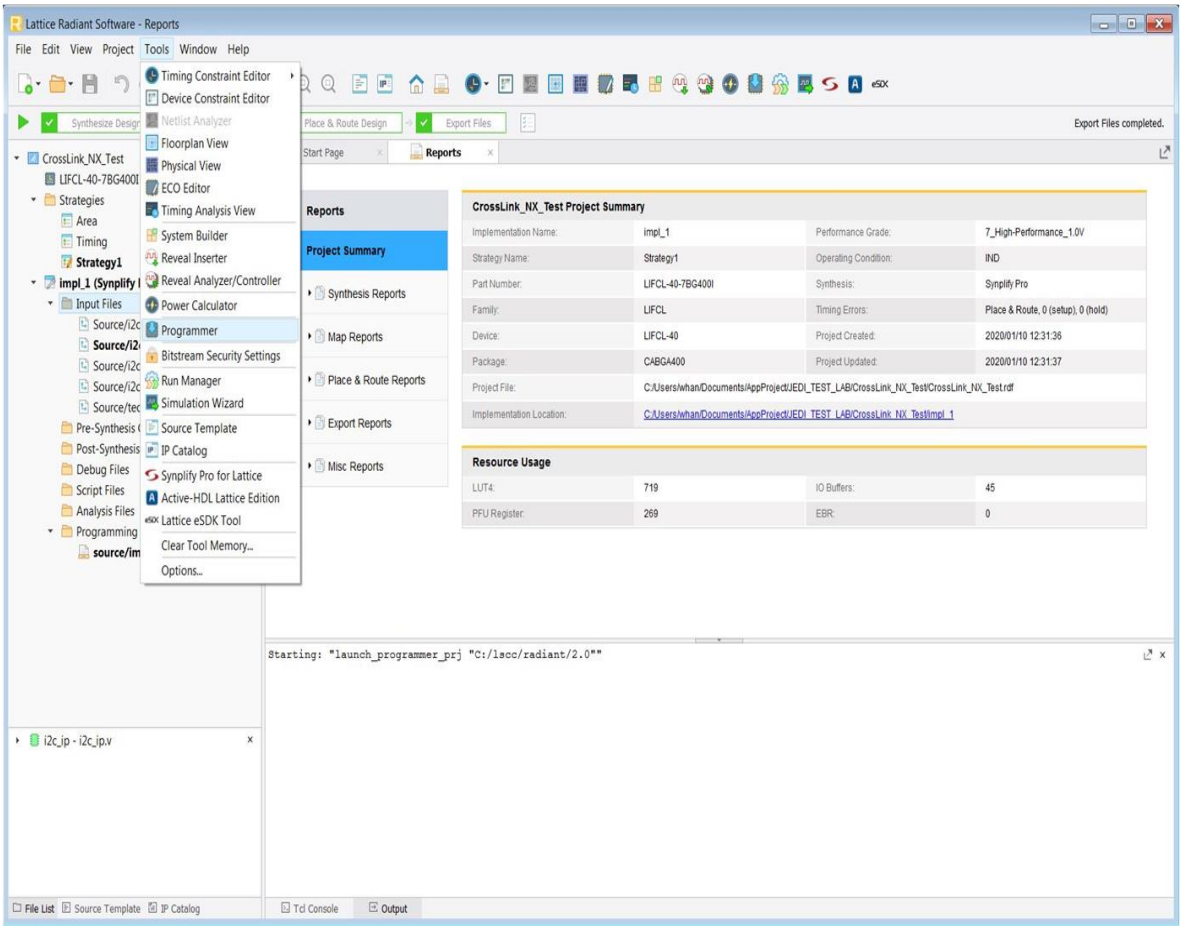

Figure 5.1. Invoke Programmer from Lattice Radiant Software

   - In Windows, go to the Windows Start menu and choose Programs > Lattice Radiant Software > Accessories > Lattice Radiant Programmer.
   - In Linux, from the *<install_path>/bin/lin* directory, enter the following on a command line:

     ```
     <Install_path>/.programmer
     ```

3. In the **Programmer** window, double-click in the **Operation** column for the FPGA device to program.

   The **Device Properties** window appears.

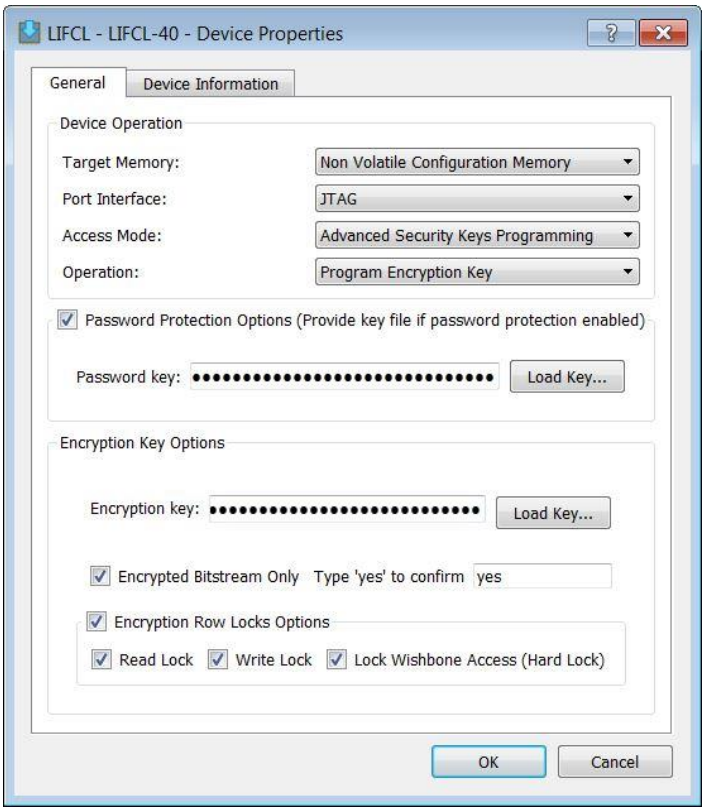

Figure 5.2. Lattice Radiant Programmer Device Property Setup

4. Select Non Volatile Configuration Memory as the Target Memory.
5. Select Port Interface using JTAG, Slave SPI or I2C.
6. Select Advanced Security Key Programming as Access Mode.
7. Select the **Program Password Key** from the **Operation** pull-down menu. The available options are:
   a. **Program Password Key** – Program 128-bit password protection key into the device. The key is from the .key file which is generated from the Lattice Radiant **Tools > Bitstream Security Settings**.
   b. **Read Password Key** – Read the contents of password protection key from the device.
   c. **Program Encryption Key** – Program the 256-bit encryption key into the device. The key is from the .bek file which is generated from the Lattice **Radiant Tools > Bitstream Security Settings**.
   d. **Read Encryption Key** – Read the contents of encryption key from the device.
   e. **Program TraceID** – Program 8-bit user's Unique ID header into the device.
   f. **Read TraceID** – Read the contents of Unique ID.
   g. **Program Locks Policies** – Program sector's OTP locking bits into the device. The locks policies included Lock the Read, Lock the Write, Lock Erase and Lock Wishbone Access (Hard Lock).
   h. **Read Locks Policies** - Reads the sector's OTP locking bits from the device.
   i. **Update Locks Polices** - Reads the sector's OTP locking bits from the device, modifies and programs sector's OTP locking bits back into the device.
   j. **Lock Ports Interface** – Program the port interface OTP locking bit into the device. The locks policies included Lock Port Access and Lock Wishbone Access (Hard Lock). The Port Interfaces included JTAG, Slave SPI and I2C.
8. Click **Load Key**. A Window Explorer window opens and prompts you to load Encryption file (.key).

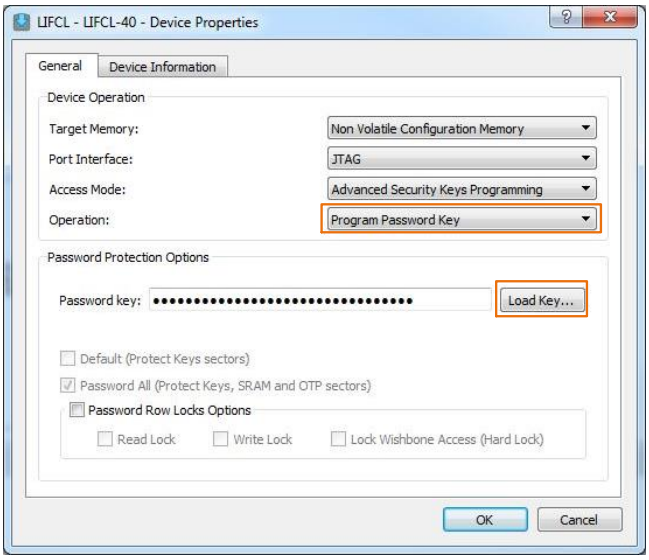

Figure 5.3. Load Password Key File

9. Select the Password Key File generated from the Lattice Radiant Bitstream Security Setting.
10. Enter the Encryption File Password into the prompt.
11. Click **OK**.

Figure 5.4. Input the Password for the Key File

The **Password Key** field is auto filled from the Key File (.key).

12. Check the **Password Row Lock Options**, and check the lock selections
    - Read Lock
    - Write Lock
    - Lock Wishbone Access (Hard Lock)
13. Click **OK** to complete the setup
14. Select the **Program** button in the **Programmer** toolbar to proceed with the programming operation.

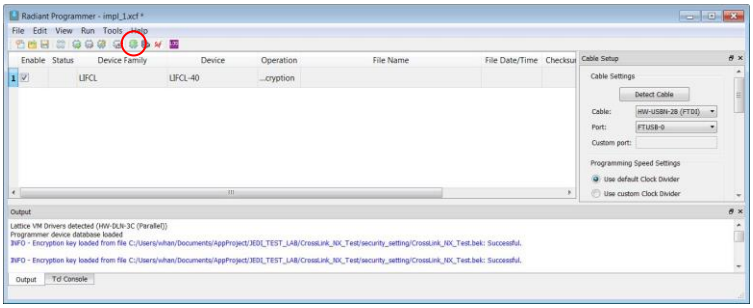

Figure 5.5. Non-Volatile Memory Programming

5.2. Programming Encryption Key

To program the AES encryption using Lattice Radiant Programmer, perform the step 1 to step 6 in the Programming Password section.

1. Select the **Program Encryption Key** from the **Operation** pull-down menu.
2. Check the **Password Protection Options** if the password protection is enabled. Click the **Load Key** button. Window Explorer opens and prompts you to load the Encryption file (.key).

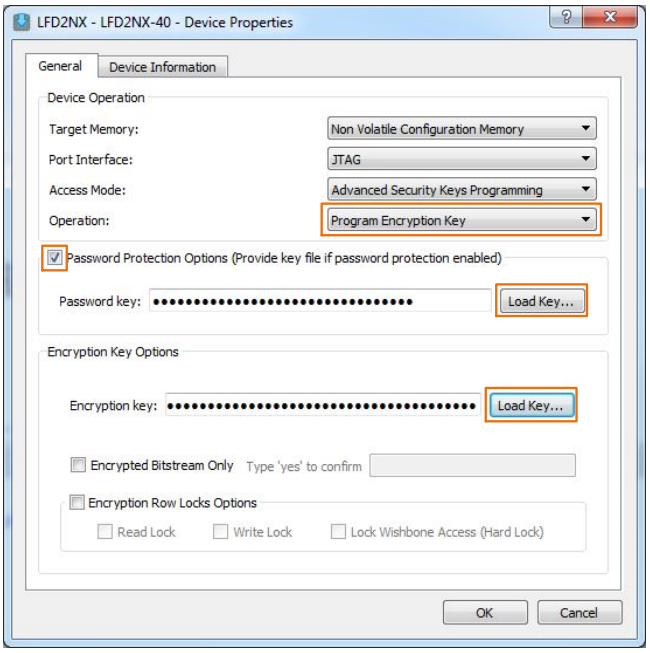

Figure 5.6. Input the Password and AES Key

3. Select the **Password key** file generated from the Lattice Radiant Bitstream Security Setting.
4. Enter the Encryption file password into the prompt.
5. Click **OK**.

Figure 5.7. Input the Password for the Key File

Note: The **Password Key** field is auto filled from the Key file (.key).

6. Under **Encryption Key Options**, click the **Load Key** button. Window Explorer opens and prompts you to load the Encryption file (.bek).
7. Select the **Encryption key file** generated from the Lattice Radiant Bitstream Security Setting.
8. Enter the Encryption file password into the prompt.
9. Click **OK**.

Note: The **Encryption Key** field is auto filled from the Encryption key file (.bek).

10. Select the **Encrypted Bitstream Only** option to allow the device to reject an unencrypted bitstream. Click **Yes** to confirm the selection.
11. Select **Encryption Row Lock Options** and choose the lock selections.
    - Read Lock
    - Write Lock
    - Lock Wishbone Access (Hard Lock)
12. Click **OK** to complete the setup.
13. Select the **Program** button in the **Programmer** toolbar to proceed with the programming operation.

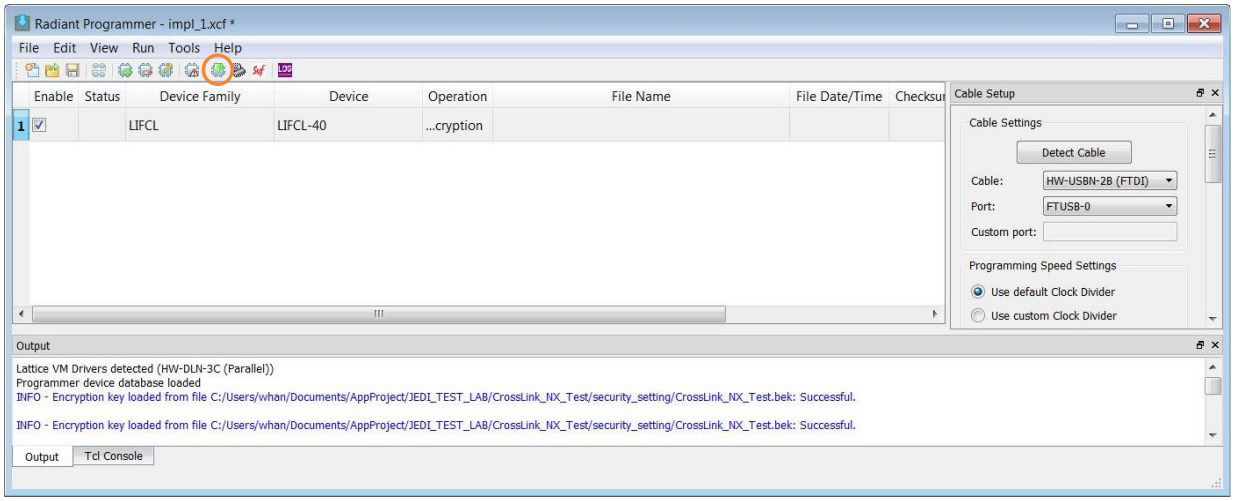

Figure 5.8. Non-Volatile Memory Programming

5.3. Programming ECDSA Public Key

To program the ECDSA Authentication Public Key using Lattice Radiant Programmer, perform the step 1 to step 6 in the Programming Password section.

1. In **Operation**, click the **Program Public Key** drop-down menu and select the options.
2. Select the **Password Protection Options** if the password protection is enabled.
3. Click the **Load Key** button. Window Explorer opens and prompts you to load the Encryption file (.key).

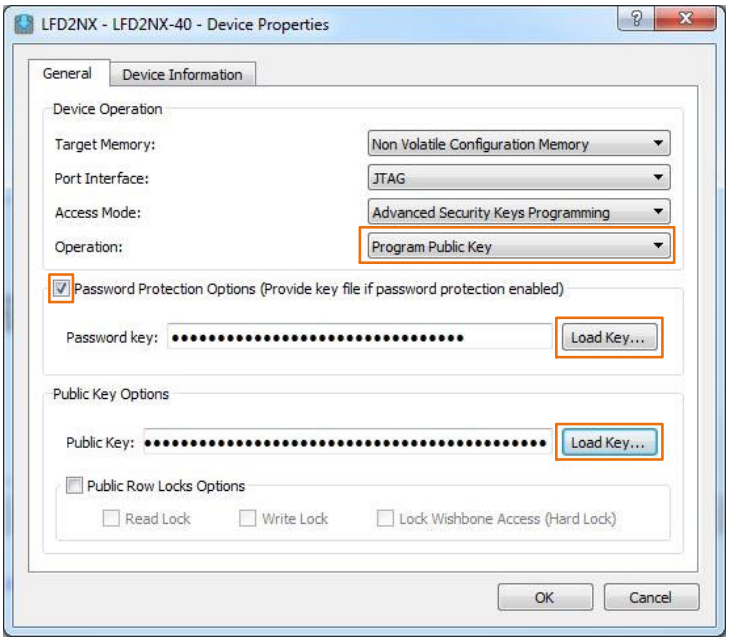

Figure 5.9. Input the Password and ECDSA Public Key

4. Select the Password key file generated from the Lattice Radiant Bitstream Security Setting.
5. Enter the Encryption file password into the prompt.
6. Click **OK**.

Figure 5.10. Input the Password for the Key File

Note: The **Password Key** field is auto filled from the Key file (.key).

7. In the **Public Key Options** field, click the **Load Key** button. Window Explorer opens and prompts you to load the ECDSA Public file (.pub).
8. Select the Public Key file generated from the Lattice Radiant Bitstream Security Setting.
9. Enter the Public file password into the prompt.
10. Click **OK**.

Note: The **Public Key** field is auto filled from the Public Key File (.pub).

11. Select the **Public Row Lock Options** and choose the lock options.
    - Read Lock
    - Write Lock
    - Lock Wishbone Access (Hard Lock)
12. Click **OK** to complete the setup
13. Select the **Program** button in the **Programmer** toolbar to proceed with the programming operation.

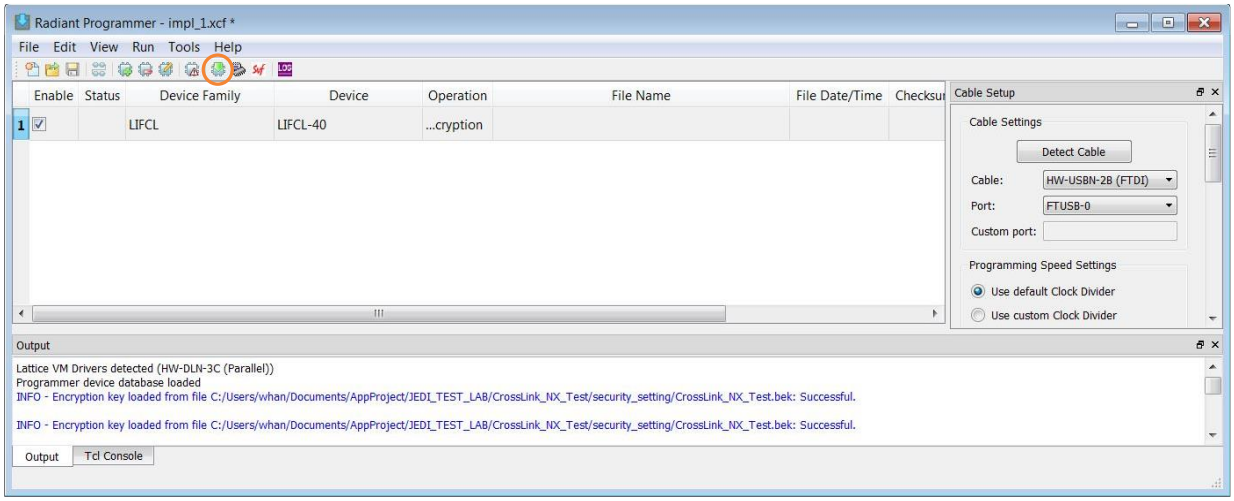

Figure 5.11. Non-Volatile Memory Programming

5.4. Programming Authentication Mode Selection

Nonvolatile EFUSE has to be programmed to select bitstream authentication mode. If the ECDSA public key is programmed, the authentication mode will be set to ECDSA automatically, hence this step can be skipped. To program the authentication mode selection, perform the step 1 to step 6 in Section 5.1.

1. Select the Enable HMAC Authentication Mode or Enable ECDSA Authentication Mode from the Operation drop-down menu.
2. Select the **Password Protection Options** if the password protection is enabled.
3. Click the **Load Key** button. Window Explorer opens and prompts you to load the Encryption file (.key).

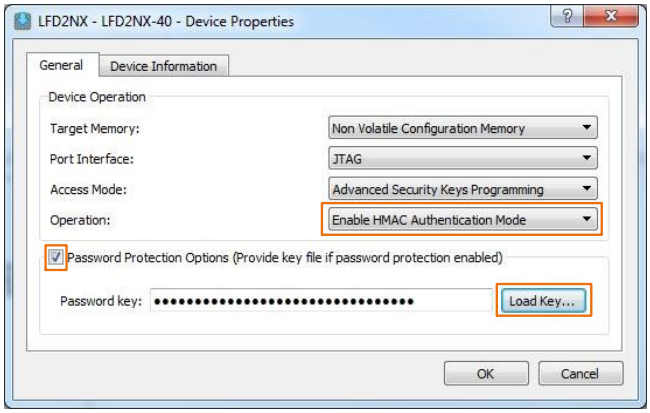

Figure 5.12. Input the Password and ECDSA Public Key

4. Select the Password Key file generated from the Lattice Radiant Bitstream Security Setting.
5. Enter the Encryption file password into the prompt.
6. Click **OK**.

Figure 5.13. Input the Password for the Key File

Note: The **Password Key** field is auto filled from the Key File (.key).

7. Click **OK** to complete the setup
8. Select the **Program** button in the **Programmer** toolbar to proceed with the programming operation.

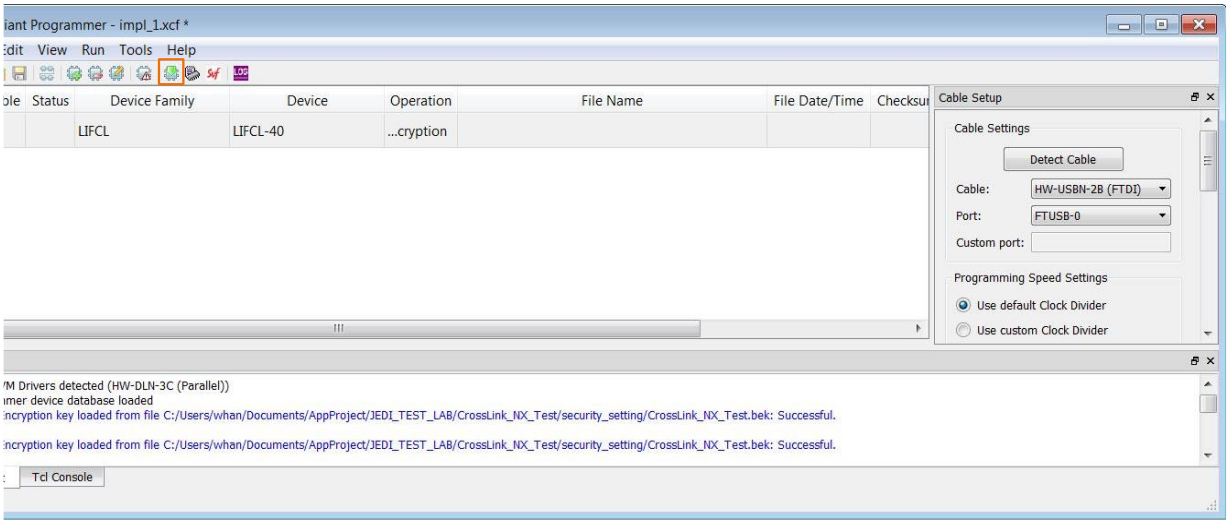

Figure 5.14. Non-Volatile Memory Programming

6. Programming Encrypted Bitstream

The encrypted bitstream programming procedure is identical to the programming procedure for a plain (unencrypted) bitstream. The following is an example of a SPI Flash Memory programming procedure.

To program the encryption key:

1. Ensure that the board with the Lattice FPGA is properly connected to your computer and turned on.
2. Open Lattice Radiant Programmer:
   - In the **Lattice Radiant** window, select **Tools > Programmer** or click the **Programmer** icon from the **Lattice Radiant** toolbar.
   - In Windows, go to the Windows Start menu and choose Programs > Lattice Radiant Software > Accessories > Lattice Radiant Programmer.
   - In Linux, from the *<install_path>/bin/lin directory*, enter the following on a command line:
     ```
     <Install_path>/.programmer
     ```

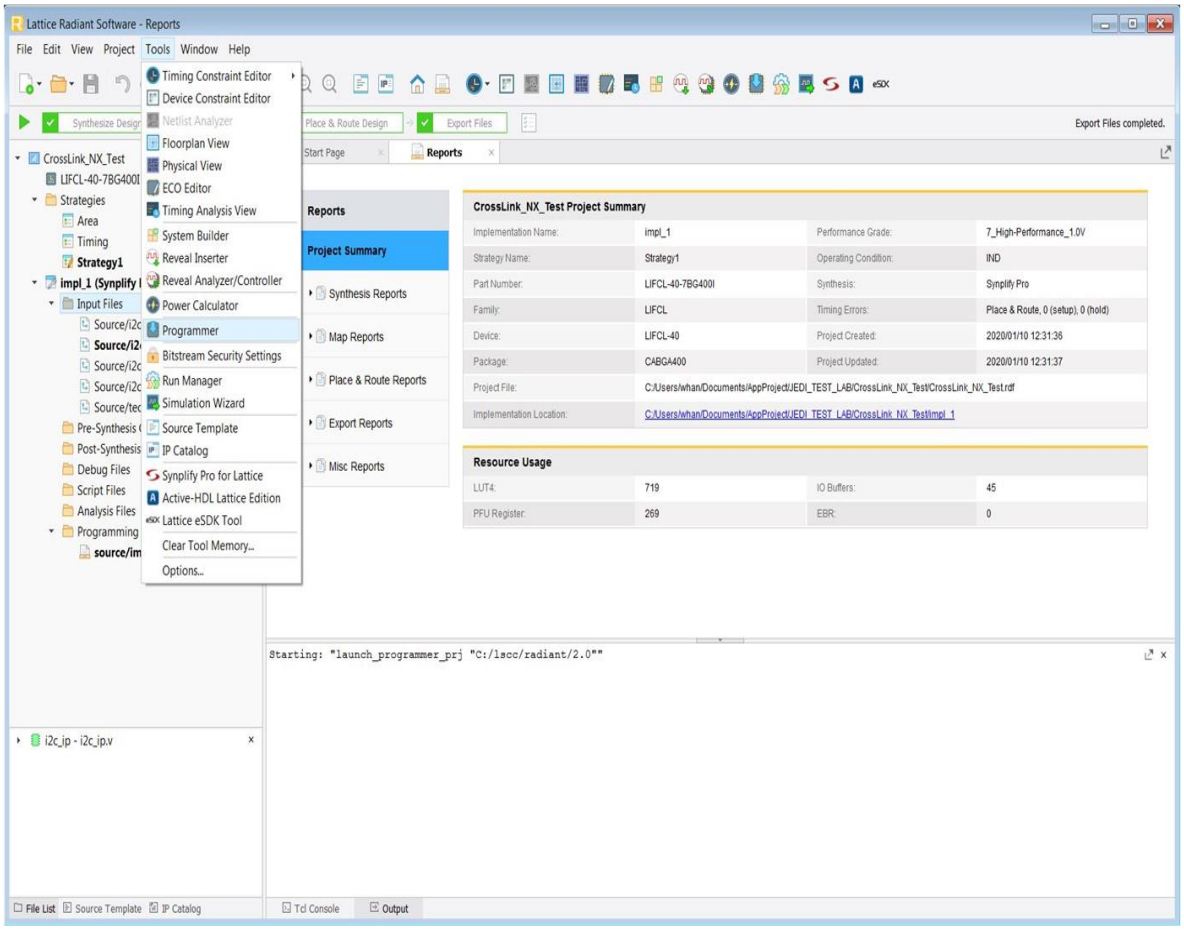

Figure 6.1. Invoke Programmer from Lattice Radiant Software

3. In the Lattice Radiant Programmer window, select Create a new project from a scan.
4. Click **OK**.

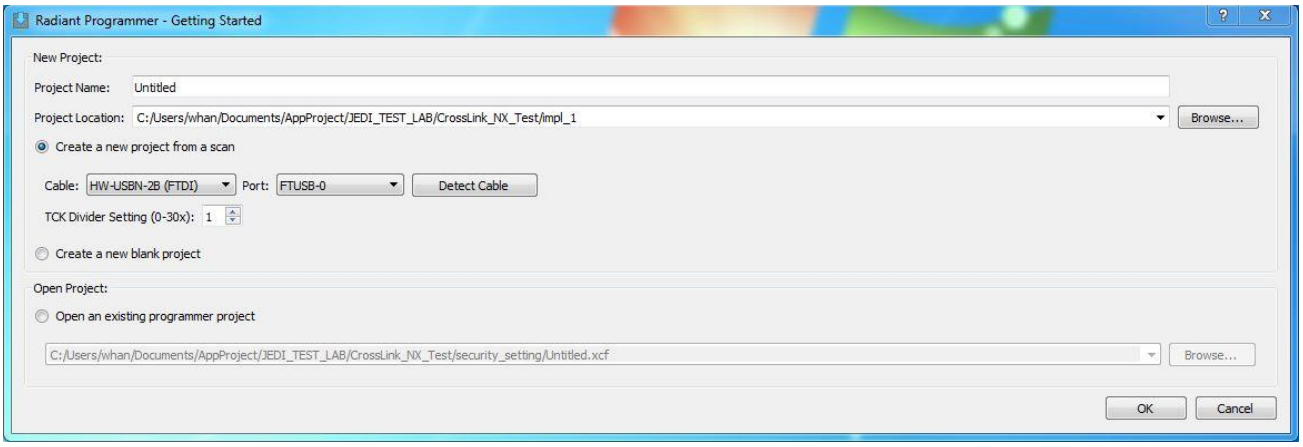

Figure 6.2. Lattice Radiant Programmer Start Up

5. In the Lattice Radiant Programmer window, select Edit > Device Properties to open the Device Properties Editor.

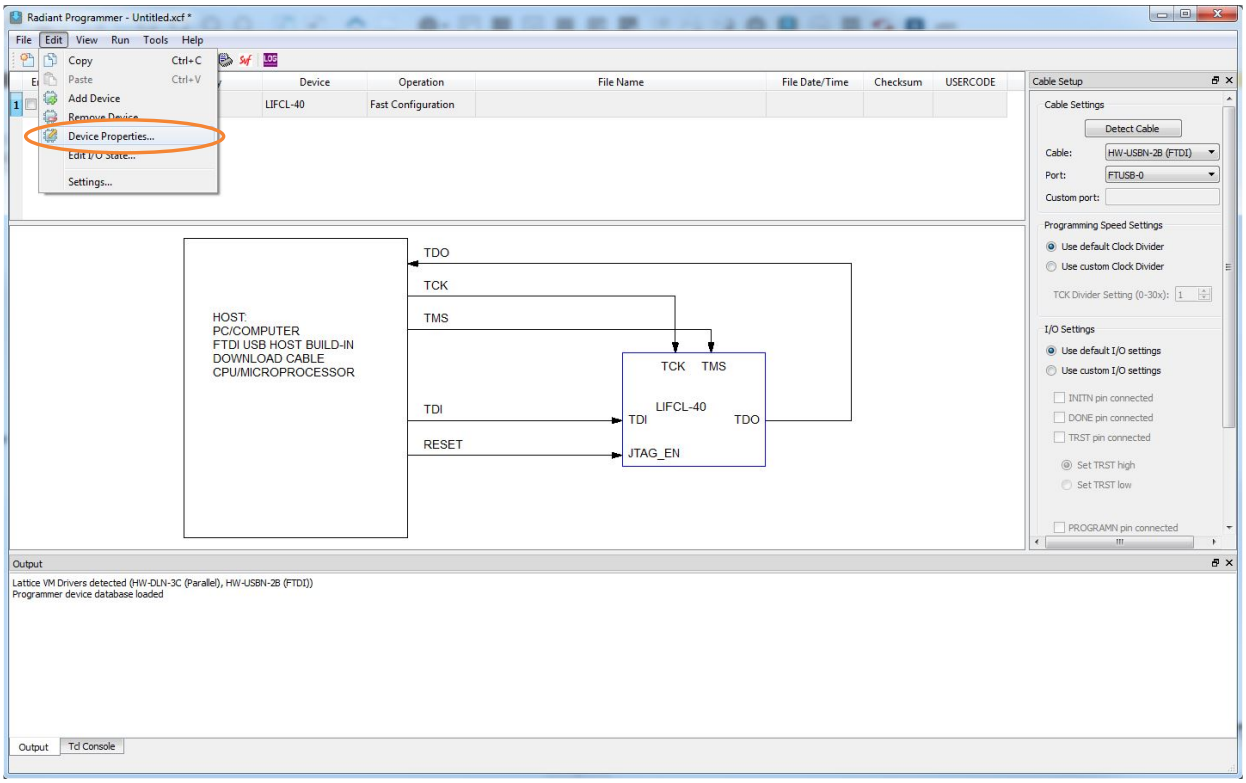

Figure 6.3. Lattice Radiant Programmer Operation Setup

6. In the Device Properties Editor window, select External SPI Flash Memory (SPI FLASH) as Target Memory.

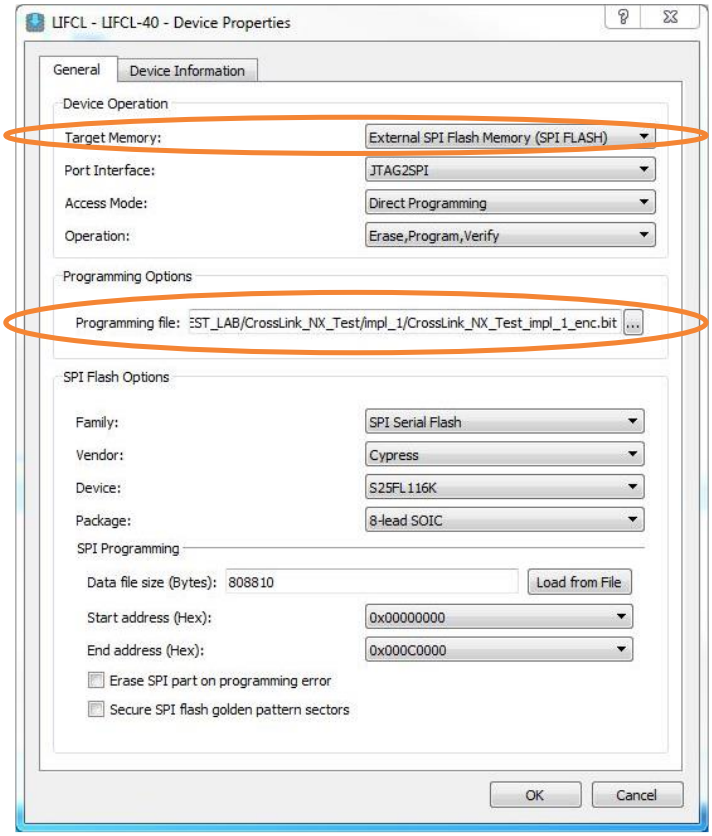

Figure 6.4. Lattice Radiant Programmer Device Properties Setup

7. Load the encrypted programming file. Make sure **Device Operation** and **SPI Flash Operation** selections are correct.
8. Click **OK**.
9. Click Program Device.

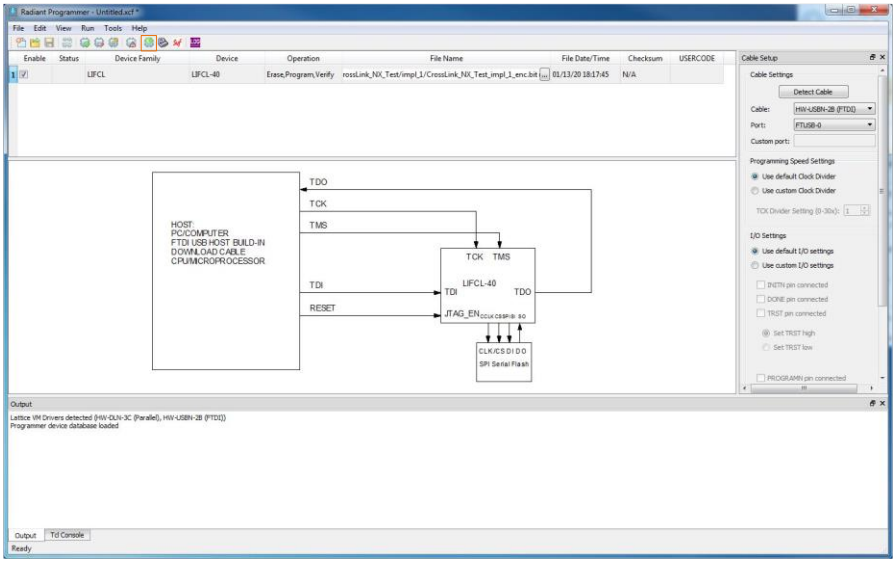

Figure 6.5. Lattice Radiant Programmer Activity Start

10. Successful programming is confirmed with the status **Operation: Successful**.

Glossary

A glossary of terms used in this document.

Term	Definition
Authentication	The algorithmic validation process to determine pass/fail results.
Bitstream	A binary file that contains commands and data that can configure FPGA devices.
Configuration	Program (write to the fuses of) a volatile device.
Decipher Key	The key code used for encryption or decryption.
Decrypt	Apply the reverse encryption process on an encrypted file.
Encrypt	The encryption action has taken place.
Encryption	Uses a password (key) and an algorithm to scramble a file.
Key Code	The binary key pattern for encryption or decryption.
Key Code Size	The fixed length of the key code in bits. For Nexus devices, it is 256 bits.
Key Lock Fuse	When programmed, the key lock fuse prevents the encryption key code from being read out.
Decrypt Only Fuse	When programmed, the device will only accept encrypted bitstream.
Non-Volatile Fuse	Fuses that keep the fuse state when power is turned off.
OTP Fuse	One-time programmable, non-volatile fuse that can only be programmed once, and cannot be erased.
Over Program	An error caused by blowing a fuse that is supposed to remain unprogrammed.
Private Key	The key code that is confined to the Trusted Area.
Program Fuse	An OTP fuse that is blown to produce a logical state 1.
Public Key	The key code that is not confined to the Trusted Area.
Un-program Fuse	An OTP fuse that is not blown (left intact) to produce a logical state 0.
Under Program	An error caused by leaving intact (un-blown) a fuse that was supposed to be programmed.
Unencrypt	No encryption action has taken place.
Trusted Area	The real or virtual space that houses all confidential and high-security material.

References

- sysCONFIG Usage Guide for Nexus Platform (FPGA-TN-02099)

Technical Support Assistance

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

Revision History

Revision 1.5, June 2021

Section	Change Summary
Introduction	Added CertusPro-NX support.

Revision 1.4, March 2021

Section	Change Summary
Overview	• Updated Figure 2.3. Nexus Bitstream ECDSA Authentication. • Changed security feature to Bitstream AES-CBC Encryption and updated description. • Updated the Method for Creating an Encrypted Bitstream File and Key File section. • Updated the Setting Security and Encryption for FPGA Devices section.
Enhanced Security Setup Using Lattice Radiant Software	Updated Step 3 in Enabling Bitstream AES Encryption. Updated Step 4 in Enabling Bitstream ECDSA Authentication.
Programming Enhanced Security Setup	Updated the caption of Figure 5.5, Figure 5.8, Figure 5.11, and Figure 5.14.
Programming Encrypted Bitstream	Updated step 7 to *Load the encrypted file*.
—	Minor editorial changes.

Revision 1.3, September 2020

Section	Change Summary
Introduction	• Changed the password protection to 128 bits. • Modified the description of 256-bit Advanced Encryption Security (AES-256) for bitstream encryption, removing the availability for both CrossLink-NX and Certus-NX devices.

Revision 1.2, June 2020

Section	Change Summary
All	Changed the document title to *Advanced Configuration Security Usage Guide for Nexus Platform*.
Introduction	• Added Hash-based Message Authentication Code (HMAC with SHA-256) for bitstream authentication support for both CrossLink-NX and Certus-NX device families. • Added Elliptic Curve Digital Signature Algorithm (ECDSA) for bitstream authentication support for both CrossLink-NX and Certus-NX device families. • Added a note to the last paragraph of this section specifying the supported Lattice Radiant Software is Version 2.1 SP1 or later.

Revision 1.1, June 2020

Section	Change Summary
All	Changed the document title to *Advanced Security Encryption Key for Nexus Platform*.
Introduction	Added support for the Nexus platform including Certus-NX and CrossLink-NX device families.
Overview	
NEXUS FPGA Bitstream Encryption Format	
Glossary	Changed *CrossLink-NX devices in Key Code Size* to *devices in Nexus product families*.

Revision 1.0, June 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com