AN 512: Using the Design Security Feature in Stratix III Devices

© March 2009 AN-512-1.1

Introduction

In today's highly competitive commercial and military environments, design security is becoming an important consideration for digital designers. As FPGAs start to play a role in larger and more critical system components, it is ever more important to protect the designs from unauthorized copying, reverse engineering, and tampering. Stratix® III devices address these concerns with the ability to decrypt a configuration bitstream using the 256-bit Advanced Encryption Standard (AES) algorithm, an industry standard encryption algorithm.

During device operation, Altera® FPGAs store configuration data in SRAM configuration cells. Because SRAM memory is volatile, SRAM cells must be loaded with configuration data each time the device powers up. Configuration data is typically transmitted from an external memory source, such as a flash memory or a configuration device, to the FPGA. It is possible to intercept the configuration data when it is being send from the memory source to the FPGA. The intercepted configuration data could then be used to configure another FPGA.

Stratix III FPGAs offer both volatile and non-volatile security key storage. When using the Stratix III design security feature, the security key is stored in the Stratix III device. Depending on the security mode, you can configure the Stratix III device using a configuration file that is encrypted with the same key, or for board testing, configure with a normal configuration file.

The design security feature is available when configuring Stratix III FPGAs using the fast passive parallel (FPP) configuration mode with an external host (such as a MAX® II device or microprocessor), or when using active serial (AS) or passive serial (PS) configuration schemes. The design security feature is not available when you are configuring your Stratix III FPGA using Joint Test Action Group (JTAG)-based configuration.

For more details, refer to "Supported Configuration Schemes" on page 25.

This document covers the following topics:

Overview of the Design Security Feature

The Stratix III design security feature is designed to protect against unauthorized copying, reverse engineering, and tampering. The following are some of the design approaches to make the solution secure:

- The non-volatile security key is stored in polyfuses under layers of metals among other polyfuses. It is very difficult to determine the functionality of a particular fuse by simple visual inspection. Moreover, additional physical security has been designed around the polyfuses to provide further security.
- Stratix III devices do not support configuration file readback. This prevents attempts to read back the configuration file after it is decrypted.
- Two 256-bit sequences are required to generate the 256-bit security key and therefore, are required to program the key into the Stratix III device. The FPGA design cannot be copied by programming a 256-bit security key into another FPGA and configuring it with an encrypted configuration file. It is virtually impossible to generate the two 256-bit sequences from the security key.
- For non-volatile key with tamper-protection bit set, the polyfuses used to store the security key are non-volatile and one-time programmable. No battery is needed. After the Stratix III device is programmed with the key, it can only be configured with configuration files encrypted with the same key. Attempts to configure the Stratix III device with an unencrypted configuration file or a configuration file encrypted with the wrong key results in configuration failure. Therefore, tampering of the design file can be detected.

Security Encryption Algorithm

Stratix III FPGAs have a dedicated decryption block that uses the AES algorithm to decrypt configuration data using a user-defined 256-bit security key. Prior to receiving the encrypted data, the user-defined 256-bit security key must be written into the device.

The AES algorithm is a symmetrical block cipher that encrypts and decrypts data in blocks of 256 bits. The encrypted data is subject to a series of transformations that includes byte substitutions, data mixing, data shifting, and key additions.

Stratix III FPGAs contain an AES decryptor block that uses the AES algorithm to decrypt the configuration data prior to configuring the FPGA device. If the security feature is not used, the AES decryptor is bypassed. The Stratix III AES implementation has been validated as conforming to the Federal Information Processing Standards FIPS-197.

For more information about the AES algorithm, refer to the *Federal Information Processing Standards Publication FIPS-197* at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf or the *AES Algorithm (Rijndael) Information* at http://csrc.nist.gov/CryptoToolkit/aes/rijndael/.

For more information about the Stratix III AES validation, refer to the *Advanced Encryption Standard Algorithm Validation List* published by the National Institute of Standards and Technology (NIST) at http://csrc.nist.gov/groups/STM/index.html.

Non-Volatile and Volatile Key Storage

Stratix III FPGAs offer both volatile and non-volatile security key storage. The volatile security key storage requires battery back-up but enables the security key to be updated, in which the non-volatile security key storage allows only one key to ever be programmed but does not require a battery.

Table 1 shows a comparison of volatile and non-volatile key storage.

Table 1. Volatile and Non-Volatile Key Comparison

Option	Volatile Key	Non-Volatile Key
Key Length	256 bits	256 bits
Key Programmability	Reprogrammable and erasable key	One-time programmable key
External Battery	Required	Not required
Key Programming Method *(1)*	On-board	Both on-board and off-board
Design Protection	Secure against copying and reverse engineering	Secure against copying, reverse engineering, and tampering *(2)*

Notes to Table 1:

(1) Key programming is carried out via the JTAG interface.

(2) Tampering is prevented only when the tamper-protection bit is set, thus preventing configuration with unencrypted Programmer Object Files (**.pof**) files.

Enabling the tamper-protection bit disables the test mode in Stratix III devices. This process is irreversible and prevents Altera from carrying out failure analysis if the test mode is disabled. Contact Altera Technical Support to enable the tamper-protection bit.

For more information about security modes that are available, refer to the *Design Security in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

Key Programming

Table 2 describes the four different methods for key programming.

Table 2. Key Programming Methods *(Note 1)* (Part 1 of 2)

Programming Procedure	Method	Programming Tool
On-Board Programming	Prototyping	EthernetBlaster, JTAG technologies, ByteBlaster™ II, USB-Blaster™ *(2)*
	Production	In-circuit tester®, JTAG technologies

Table 2. Key Programming Methods *(Note 1)* (Part 2 of 2)

Programming Procedure	Method	Programming Tool
Off-Board Programming	Prototyping	System General, third-party programming service
	Production	System General, third-party programming service

Notes to Table 2:

(1) Contact Altera Technical Support for information on programming support.

(2) ByteBlaster II and USB-Blaster support only volatile security key programming. For non-volatile key programming, use the EthernetBlaster or JTAG technologies.

Key programming uses the following definitions:

- **On-board:** procedure in which the device is programmed on the customer board.
- **Off-board:** procedure in which the device is programmed on a separate programming system.
- **Prototyping:** method initially used to verify proper operation of a particular method.
- **Production:** method used for large-volume production.

Hardware and Software Requirements

This section provides the hardware and software requirements for the Stratix III design security feature. When using this feature, a volatile or non-volatile security key is stored in the Stratix III FPGA. The design security key is programmed before the Stratix III is configured and enters user mode.

Hardware Requirements

Table 3 shows the voltage specifications that you need to follow for a successful security key programming.

Table 3. Voltage Specifications for Design Security Feature

Parameter	Key Programming Mode
Voltage (VCCPT)	2.5 V ± 0.125 V
TCK Period *(2)*	10 µs ± 1 µs
Ambient Temperature	25°C ± 5°C
Voltage (VCCBAT) *(1)*	1.0 V (minimum), 3.0 V (typical), 3.3 V (maximum)

Note to Table 3:

(1) V_{CCBAT} is a dedicated power supply for the volatile key storage and is not shared with other on-chip power supplies, such as V_{CCIO} or V_{CC}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition. If you do not use the volatile security key, you may connect the V_{CCBAT} to either ground or a 3.0 V power supply.

(2) The TCK period specification is applied to non-volatile key programming. The TCK period specification for volatile key programming is the same as device JTAG TCK specification.

Non-volatile key programming must be performed within the allowable TCK frequency, which approximates up to 100 kHz.

After power-up, you must wait 100 ms (PORSEL = 0) or 12 ms (PORSEL = 1) before beginning the key programming to ensure that VCCBAT is at its full rail.

Examples of lithium coin-cell type batteries that are used for volatile key storage purposes are BR1220 (–30° to +80°C) and BR2477A (–40°C to +125°C).

For more information about battery specifications, refer to the *Stratix III Device Datasheet: DC and Switching Characteristics* chapter in volume 2 of the *Stratix III Device Handbook*.

Additional operating conditions must be met in order to ensure proper operation. For proper operating conditions for Stratix III devices, refer to the *Stratix III Device Datasheet: DC and Switching Characteristics* chapter in volume 2 of the *Stratix III Device Handbook*.

Software Requirements

To enable the design security feature of Stratix III FPGAs, you must use the Quartus II software version 7.2 SP2 or later. You can obtain a license file to enable the Stratix III design security feature from Altera Technical Support.

Steps for Implementing a Secure Configuration Flow

To implement a secure configuration flow, perform the following steps as shown in Figure 1:

1. Generate the encryption key programming file and encrypt the configuration data.

   The Quartus II configuration software always uses the user-defined 256-bit security key to generate a key programming file and an encrypted configuration file. The encrypted configuration file is stored in an external memory, such as a flash memory or a configuration device. For more infomation, refer to "Step 1: Generate the Encryption Key Programming File and Encrypt Configuration File" on page 6.

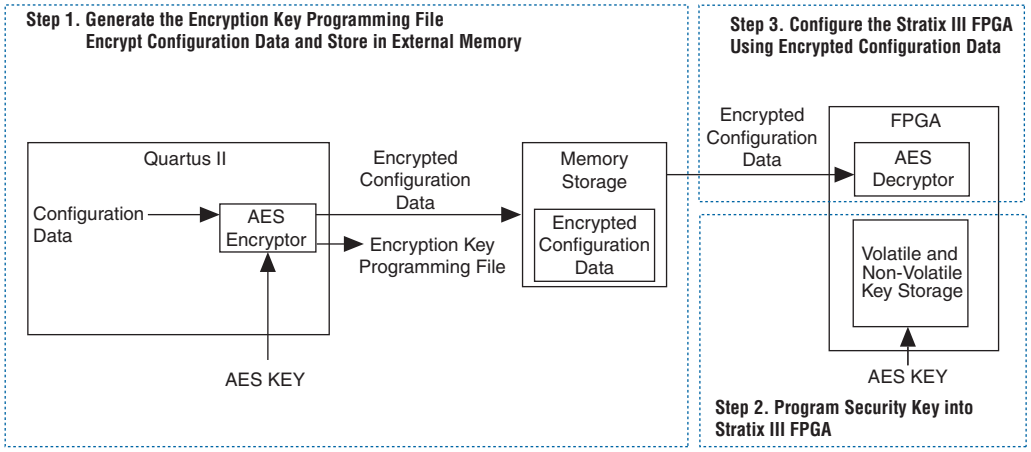
2. Program the user-defined 256-bit security key into the Stratix III device.

   For more infomation, refer to "Step 2a: Program the Volatile Security Key into Stratix III Device " on page 17 and "Step 2b: Program the Non-Volatile Security Key into Stratix III Device" on page 17.

3. Configure the Stratix III device.

   At power-up, the external memory source sends the encrypted configuration file to the Stratix III FPGA. The Stratix III device uses the stored security key to decrypt the file and to configure itself. For more infomation on how to configure Stratix III device with encrypted configuration data, refer to "Step 3: Configure the Stratix III Device with Encrypted Configuration Data" on page 24.

Figure 1. Stratix III Secure Configuration Flow



Step 1: Generate the Encryption Key Programming File and Encrypt Configuration File

To use the design security feature in Stratix III FPGAs, you must generate an encryption key programming file and encrypt your configuration files using the Quartus II software version 7.2 SP2 or later (make sure you use the same two 256-bit sequences for both). The security key is not saved into any Quartus II-generated configuration files and the actual 256-bit security key is generated from the two 256-bit sequences. This makes it impossible to copy the security key to other Stratix III FPGAs.

☞ To enable Stratix III design security feature, a license file must be obtained. Contact Altera Technical Support for assistance.

The encryption key programming file has different formats, depending on the hardware or system used for programming. There are three file formats supported by the Quartus II version 7.2 SP2 and later:

- JBC (**.ekp**)
- JEDEC STAPL (**.jam**)
- Serial Vector Format (**.svf**)

☞ Only the **.ekp** file type is generated automatically from the Quartus II software. You must create the **.jam** and **.svf** files using the Quartus II software if these files are required in the key programming. Quartus II software version 7.2 SP2 or later generates the JBC format of encryption key programming file in the same project directory.

☞ Altera recommends keeping the encryption key programming file confidential.

The JBC format is used with the EthernetBlaster communications cable or USB-Blaster download cable and the Quartus II software. The EthernetBlaster communications cable can support both volatile and non-volatile key programming whereas the USB-Blaster download cable is used only for volatile key programming. The **.jam** file format is generally used with third-party programming vendors and JTAG programmer vendors. The **.svf** file format is used with JTAG programmer vendors and in-circuit test vendors.

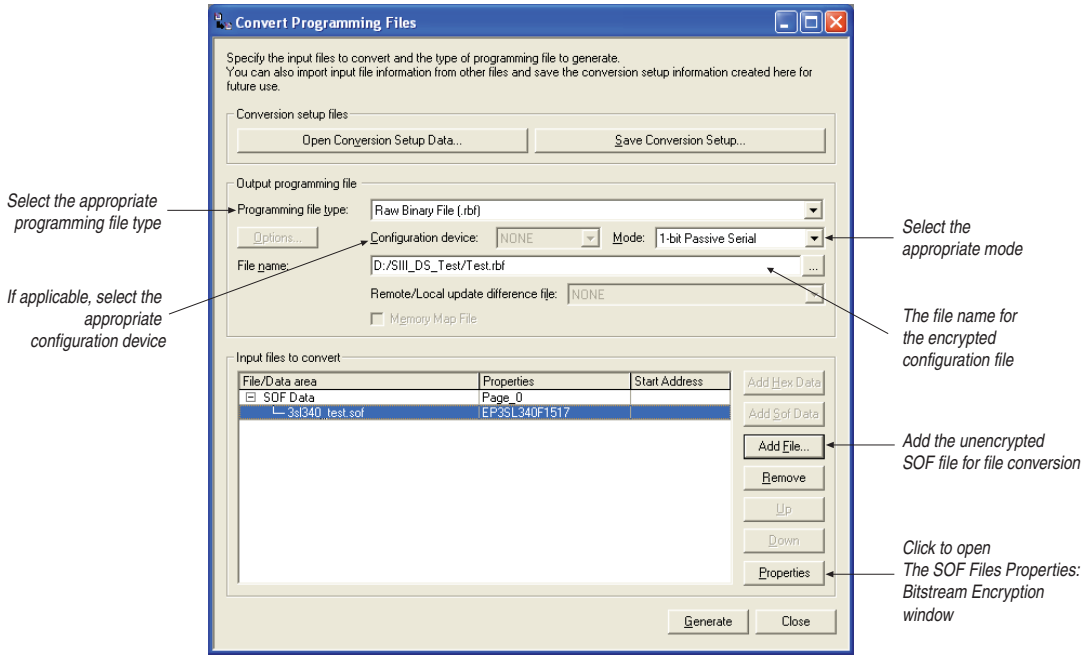
How to Generate the Single-Device Encryption Key Programming File and Encrypt the Configuration File Using the Quartus II 7.2 SP2 or Later

Use the following steps to generate a single-device encryption key programming file and encrypt your configuration file:

1. Obtain a license file to enable the Stratix III design security feature from Altera Technical Support.
2. Start the Quartus II software.
3. In the Tools menu, click **License Setup**. The Options window displays the **License Setup** options.
4. In the License file field, enter the location and name of the license file, or browse to and select the license file.
5. Click **OK**.
6. Compile your design by using one of the following options:
   a. In the Processing menu, click **Start Compilation**.
   b. In the Processing menu, select **Start**, and click **Start Assembler**.

      An unencrypted SRAM Object File (**.sof**) is generated.
7. In the File menu, click **Convert Programming Files**. The Convert Programming Files window appears (Figure 2).

Figure 2. Convert Programming Files Window



a. In the Convert Programming Files window, select the programming file type from the Programming file type list.

b. If applicable, select the appropriate configuration device from the Configuration device list.

c. Select the mode from the Mode list.

d. Type the file name in the File name field, or browse to and select the file.

e. Under the Input files to convert section, click **SOF Data**.

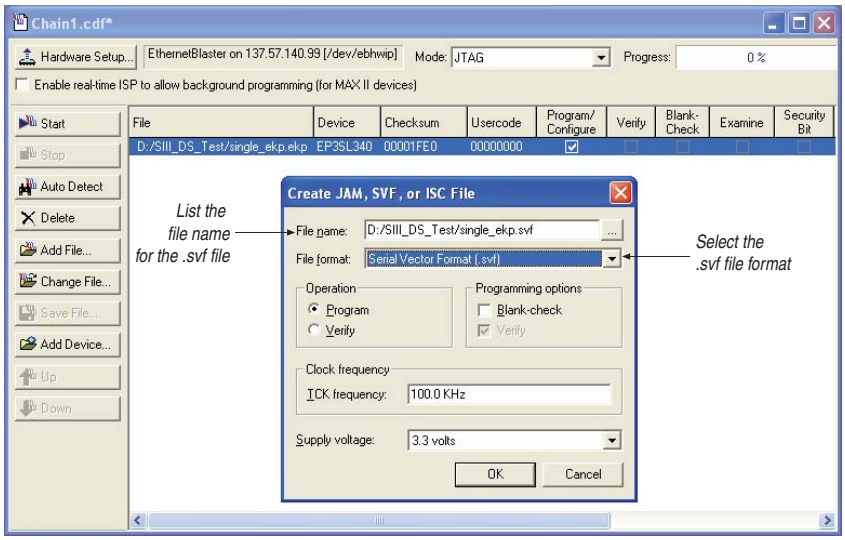
f. Click **Add File** to open the Select Input File window.

g. Browse to the unencrypted SOF file and click **Open.**

h. Under the Input files to convert section, click on the SOF file name. The field is highlighted.

i. Click **Properties.** The SOF Files Properties: Bitstream Encryption window appears (Figure 3).

Figure 3. SOF File Properties: Bitstream Encryption Window



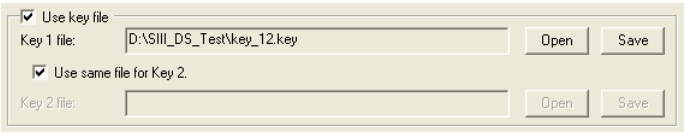
j. In the SOF Files Properties: Bitstream Encryption window, turn on **Generate encrypted bitstream**.

k. Turn on Generate key programming file and type the encryption key programming file path and file name in the text area, or browse to and select <*filename*>**.ekp**.

l. Add the keys to the pull-down list either by using a **.key** file or the Add button. The **Add** and **Edit** buttons bring up the Key Entry window. The **Delete** button deletes the currently selected key from the pull-down list(Figure 3).

Using the **.key** file option allows you to specify one or two key files in the corresponding pull-down list. You may use different files for the Key 1 and Key 2 fields, or use one **.key** file for both. (Figure 4).

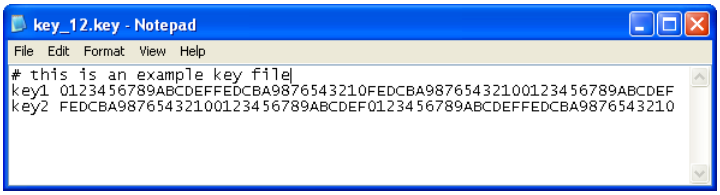
Figure 4. Use Key File Option



The **.key** file is a plain text file in which each line represents a key unless the line starts with "#". The "#" symbol is used to denote comments.

Each valid key line have the following format: *<key identity><white space><256-bit hexadecimal key>*, as shown in Figure 5.

Figure 5. Example of .key File



The key identity is an alphanumeric name that is used to identify the keys (similar as the key file entry). It is also the text displayed when the **Show entered keys** check box is turned off (Figure 6). It is displayed together with the full key when **Show entered keys** is turned on (Figure 7).

Figure 6. Key Identity

Figure 7. Key Identity and the Full Key Entry

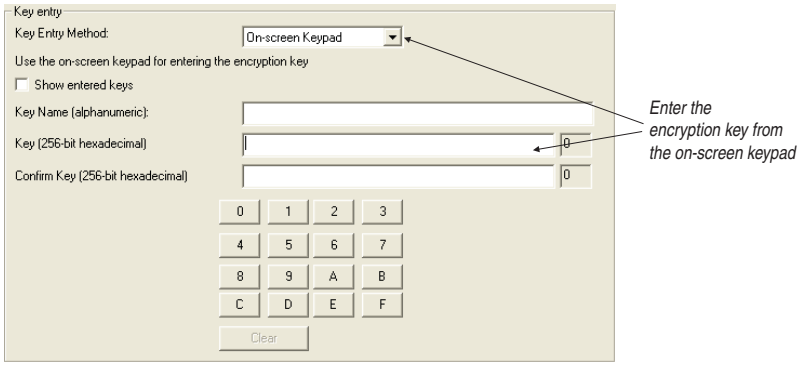
The keys in the pull-down list can be saved to a **.key** file. You need to click the corresponding **Save** button to save a key. This displays the standard File dialog box. All the keys in the pull-down list is saved to the selected or created **.key** file (Figure 4 on page 9).

Select the **Key Entry Method** to enter the encryption key either with the on-screen keypad or keyboard (Figure 8).

Figure 8. Key Entry Method

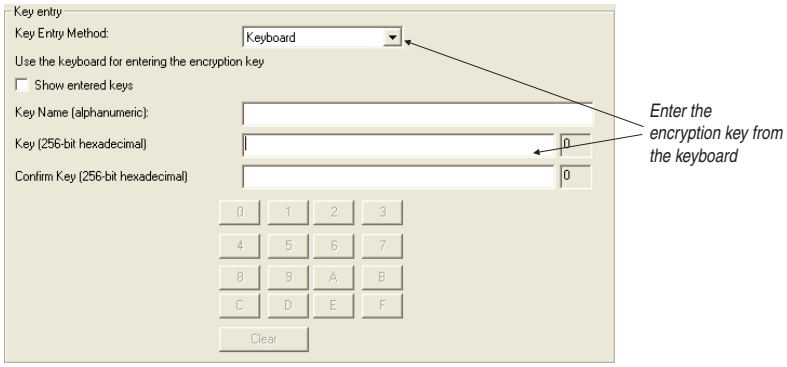
The on-screen keypad allows you to enter the keys using the keypad shown in Figure 9. Select a key and click on the on-screen keypad to enter values. You have the option of allowing the keys to be shown as they are entered; if this option is used, you do not need to confirm the key.

Figure 9. On-Screen Keypad



While the on-screen keypad is being used, any attempt to use the keyboard to enter the keys generates a pop-up notification and the key press is ignored. Alternatively, you can enter the encryption key from the keyboard (Figure 10).

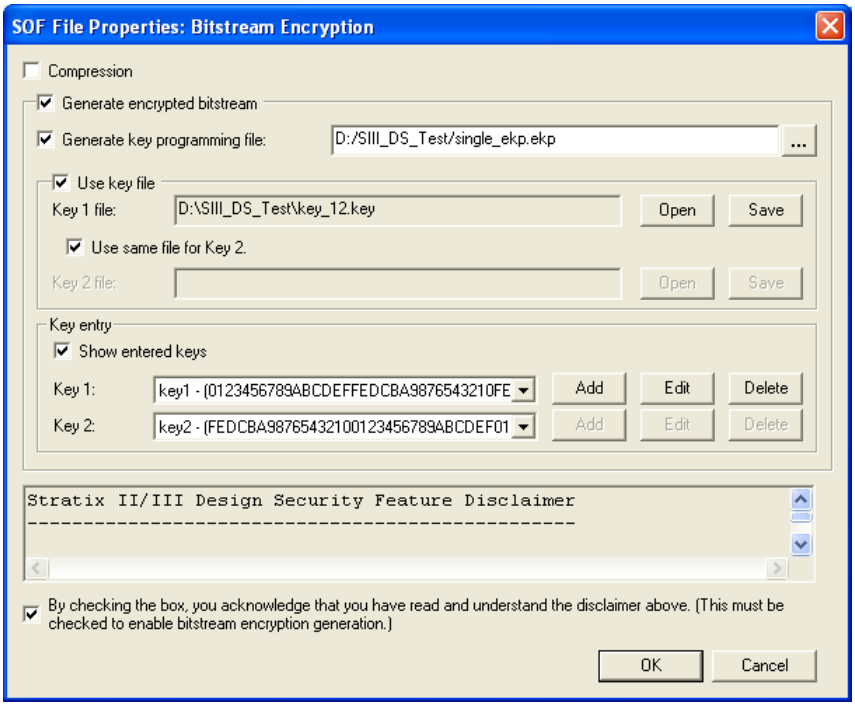
Figure 10. Keyboard



m. .Read the Stratix II or Stratix III Design Security Feature Disclaimer. If you agree to and acknowledge the Stratix II or Stratix III Design Security Feature Disclaimer, turn on the acknowledgement box (Figure 10).

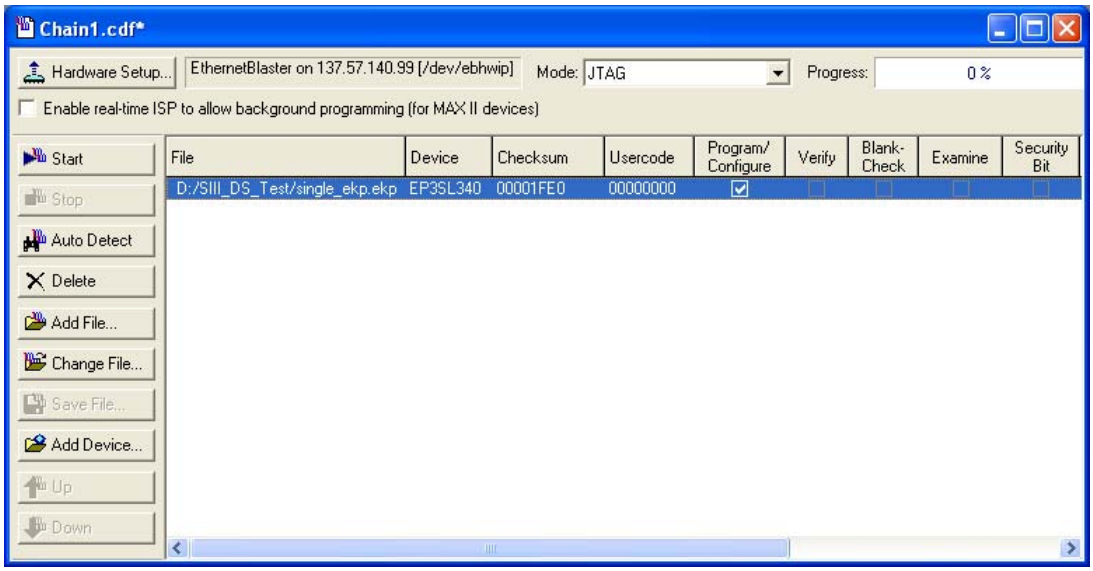
n. Click OK.

Figure 11. Example of User Interface



8. In the Convert Programming Files window, click **OK**. The <*filename*>**.ekp** and encrypted configuration file are generated in the same project directory.
9. In the Tools menu, click **Programmer**. The Programmer window appears (Figure 12).

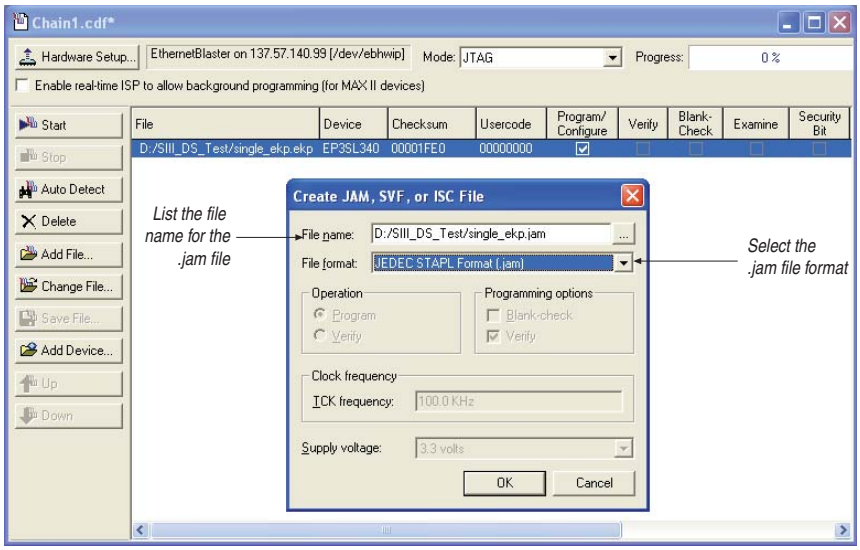
Figure 12. Programmer Window



10. In the Mode list, select JTAG as the programming mode.

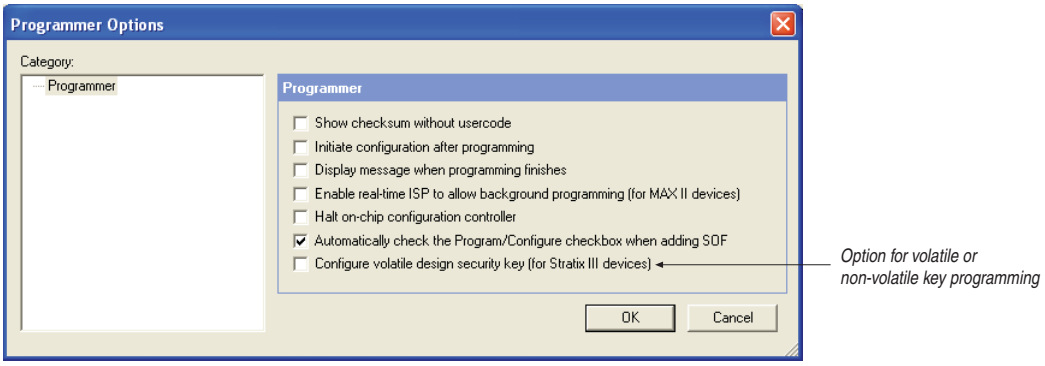
11. Click **Hardware Setup.** The Hardware Setup window appears.
    a. In the currently selected hardware list, select EthernetBlaster as the programming hardware.
    b. Click **Done**.
12. Click Add File. The Select Programmer File window appears.
    a. Type <*filename*>.**ekp** in the File name field.
    b. Click Open.
13. Highlight the encryption key programming (**.ekp**) file you added and click **Program/Configure**.
14. In the File menu, point to Create/Update and click **Create JAM, SVF, or ISC File.** The Create JAM, SVF or ISC File window appears (Figure 13).

Figure 13. Create .jam File from Single-Device Encryption Key Programming File



15. Select the file format required, that is the JEDEC STAPL Format (**.jam**), for the encryption key programming file in the File format field.
16. Type the file name in the File name field, or browse to and select the file.
17. Click **OK** to generate the **.jam** file.
18. In the Tools menu, click **Programmer Options.** The Programmer Options window appears (Figure 14).

Figure 14. Programmer Options Window

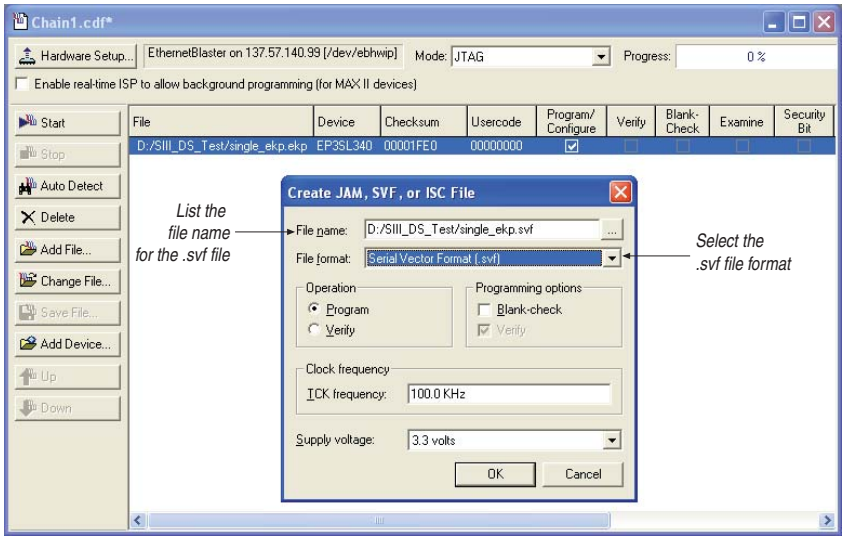


For Stratix III devices, you must turn off **Configure volatile design security key** to generate a non-volatile .svf file of the encryption key programming file (Figure 14).

19. Click **OK**.
20. Repeat Steps 15—17 to generate a **.svf** file of the encryption key programming file. Use the default setting in the Create JAM, SVF, or ISC File window when generating a **.svf** file of the encryption key programming file (Figure 15).

To generate a **.svf** file of the encryption key programming file using the Quartus II software version 7.2 SP2, a Quartus II software patch must be obtained. Contact Altera Technical Support for assistance.

Figure 15. Create .svf File from Single-Device Encryption Key Programming File



How to Generate the Single-Device Encryption Key Programming File Using the Quartus II Software Version 7.2 SP2 or Later With Command-Line Interface

There is a command-line interface that allows you to generate a single-device encryption key programming file. The command line uses the Quartus II Software Command-Line Executable, quartus_cpf, and requires the following syntax or options:

- `--gen_ekp/-e`
- --key/-k <*path to key file*>:<*key identity*>
- An SRAM Object File (user design)
- An encryption key programming file (encryption key programming file name that is required)

Example 1 shows two sets of keys that are stored in two different key files: key1 in **mykeys.key** and key2 in **otherkeys.key**.

Example 1.

```
quartus_cpf --key D:\SIII_DS\mykeys.key:key1 --key
D:\SIII_DS\otherkeys.key:key2 D:\SIII_DS\test.sof D:\SIII_DS\test.ekp
```

Example 2 shows two sets of keys that are stored in the same key file: key1 and key2 in **mykeys.key**.

Example 2.

```
quartus_cpf --key D:\SIII_DS\mykeys.key:key1:key2 D:\SIII_DS\test.sof
D:\SIII_DS\test.ekp
```

How to Generate the Multi-Device Encryption Key Programming File and Encrypt the Configuration File using Quartus II Software Version 7.2 SP2 or Later

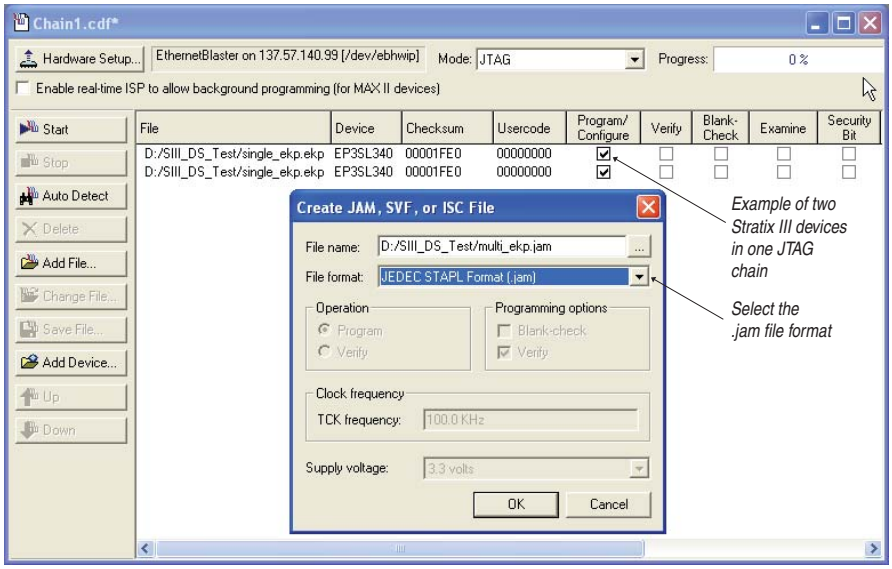
Use the following steps to generate a multi-device encryption key programming file and encrypt your configuration file:

1. Start the Quartus II software.
2. Repeat Steps 9 – 11 in "How to Generate the Single-Device Encryption Key Programming File and Encrypt the Configuration File Using the Quartus II 7.2 SP2 or Later " on page 7.
3. Click **Add File.** The Select Programmer File window displays.
   a. Select the single-device encryption key programming file, and type <*single_ekp*>**.ekp** in the File name field.
   b. Click **Open.**

For the correct sequence of devices in the same JTAG chain, you can use the Auto-Detect option in the Quartus II programmer. If one of the FPGA devices does not need to be key-programmed, you do not need to replace the device with the <*single_ekp*>**.ekp** file in the Quartus II programmer.

4. Repeat Step 3 for the number of the devices in the same chain. Ensure that the right device sequence is used when adding the encryption key programming files to the programmer window.
5. Highlight all the encryption key programming files you added and click **Program/Configure**.
6. In the File menu, point to **Create/Update** and click **Create JAM, SVF, or ISC File.** The Create JAM, SVF, or ISC File window appears. (Figure 16).

Figure 16. Multi-Device Key Programming: .jam File Generation



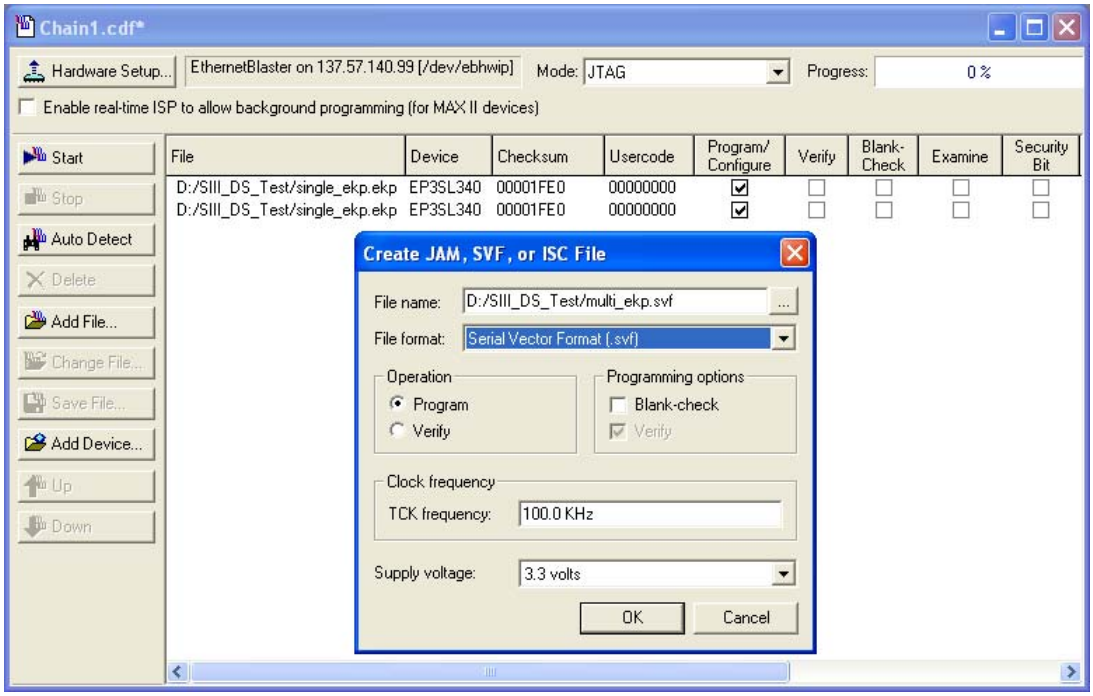
7. Select the file format required, that is the JEDEC STAPL Format (**.jam**), for all the encryption key programming files in the File format field.
8. Type the file name in the File name field, or browse to and select the file.
9. Click **OK** to generate the **.jam** file.
10. In the Tools menu, click **Programmer Options.** The Programmer Options window appears.

    For Stratix III devices, you must turn off **Configure volatile design security key** to generate a non-volatile **.svf** file of the encryption key programming file.

11. Click **OK**.
12. Repeat Steps 7–9 to generate **.svf** file for all the encryption key programming files. Use the default setting in Create JAM, SVF, or ISC File window when generating a **.svf** file of the encryption key programming file (Figure 17).

    To generate a **.svf** file of the encryption key programming file using the Quartus II software version 7.2 SP2, a Quartus II software patch must be obtained. Contact Altera Technical Support for assistance.

Figure 17. Multi-Device Key Programming: .svf File Generation



Step 2a: Program the Volatile Security Key into Stratix III Device

Before programming the volatile security key into the Stratix III FPGA, ensure that the FPGA can be configured successfully with an unencrypted configuration file. The volatile security key is a reprogrammable and erasable key. Before you program the Stratix III device with the volatile security key, you must provide an external battery to retain the volatile security key. Stratix III devices with successful volatile security key programmed can accept both encrypted and unencrypted configuration bitstreams. This enables the use of unencrypted configuration bitstreams for board-level testing.

Any attempt to configure a Stratix III device containing the volatile security key with a configuration file encrypted with the wrong key causes the configuration to fail. If this occurs, the nSTATUS signal from the FPGA pulses low and continues to reset itself.

You can program the security key into the Stratix III FPGA using on-board prototyping from Table 2 on page 3 .

Step 2b: Program the Non-Volatile Security Key into Stratix III Device

Before programming the non-volatile security key into the Stratix III FPGA, ensure that the FPGA can be configured successfully with an unencrypted configuration file. The security key is one-time programmable through the JTAG interface. You can program the non-volatile key into the Stratix III device without an external battery. Stratix III devices with successful non-volatile security key programmed can accept both encrypted and unencrypted configuration bitstreams. This enables the use of unencrypted configuration bitstreams for board-level testing.

Any attempt to configure a Stratix III device containing the non-volatile security key with a configuration file encrypted with the wrong key causes the configuration to fail. If this occurs, the nSTATUS signal from the FPGA pulses low and continues to reset itself.

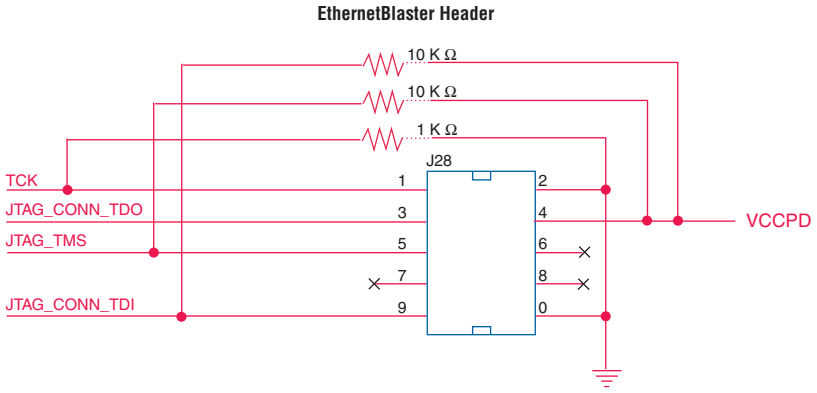
You can program the security key into the Stratix III FPGA using on-board prototyping, volume production, and off-board prototyping and production solutions from Table 2 on page 3 .

Volatile or Non-Volatile Security Key Programming Using EthernetBlaster and Quartus II Software

Connect the EthernetBlaster communications cable to the EthernetBlaster header as shown in Figure 18.

For additional information about connecting the EthernetBlaster communications cable, refer to the *EthernetBlaster Communications Cable User Guide*.

Figure 18. EthernetBlaster Header *(Note 1), (2)*



Notes to Figure 18:

(1) A 1-KΩ pull-down resistor is added to the TCK while 10-KΩ pull-up resistors are added to the TMS and TDI signals for security key programming.

(2) The EthernetBlaster header and USB-Blaster header are identical for security key programming.

How to Perform Single-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later

To perform single-device volatile or non-volatile security key programming using the Quartus II software through the EthernetBlaster, perform the following steps:

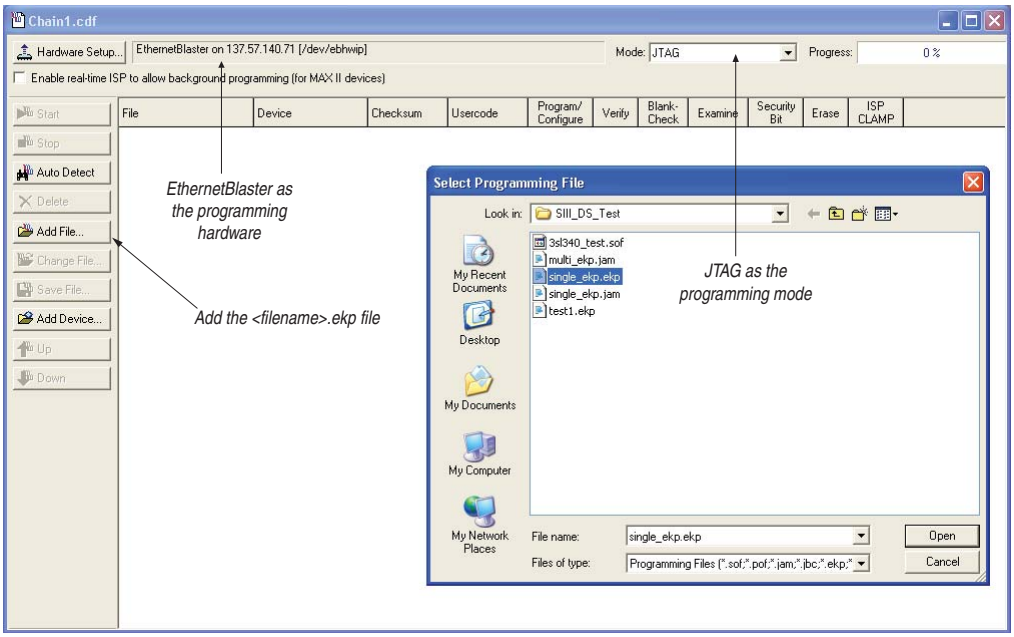
1. Check the firmware version of the EthernetBlaster. Verify that the JTAG firmware build number is 101 or greater. If the version precedes build number 101, apply the firmware upgrade.

   Apply the firmware upgrade (EBFW100101.tar.gz) to the EthernetBlaster unit. This updates the JTAG Firmware to Build 101. For firmware upgrade directions, refer to the *EthernetBlaster Communications Cable User Guide.*

2. Start the Quartus II software.

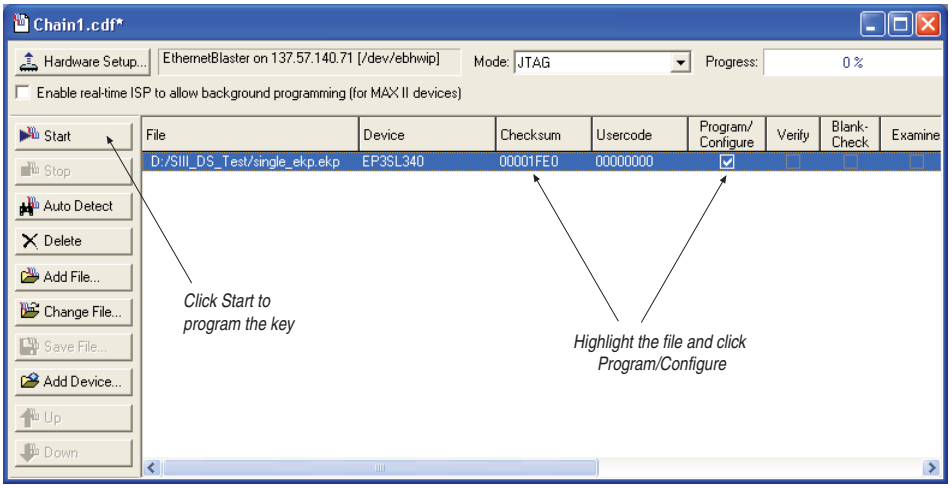
3. In the Tools menu, click **Programmer.** The Programmer window appears (Figure 19).

Figure 19. Key Programming Using EthernetBlaster and Quartus II Software



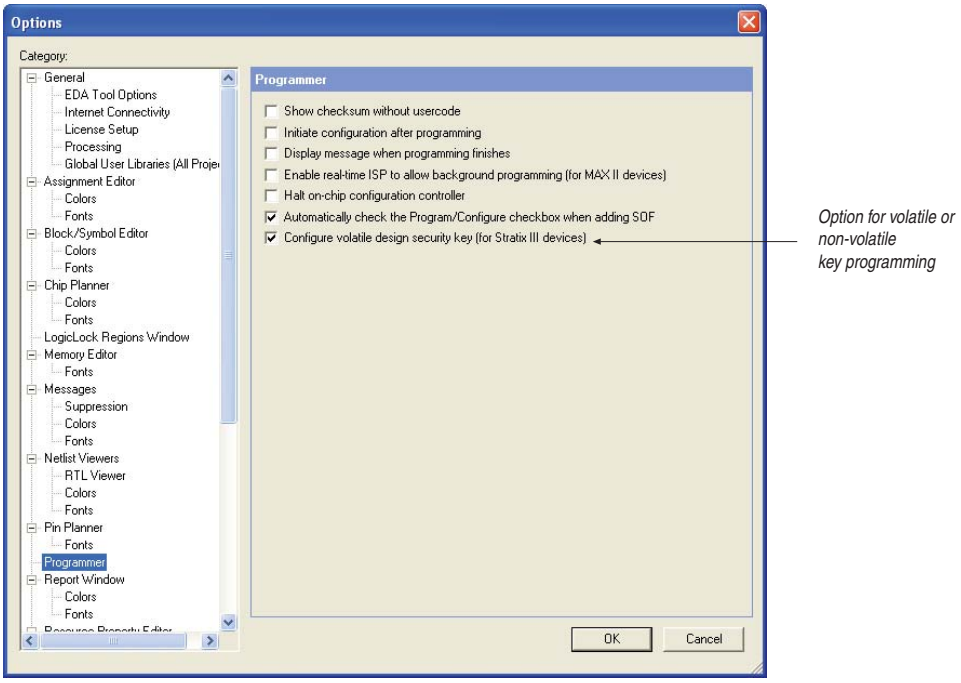
4. In the Mode list, select JTAG as the programming mode (Figure 19).
5. Click **Hardware Setup.** The Hardware Setup window appears.
   a. In the Currently selected hardware list, select EthernetBlaster as the programming hardware.
   b. Click **Done.**
6. Click **Add File.** The Select Programmer File window appears.
   a. Type <*filename*>**.ekp** in the File name field.
   b. Click **Open.**
7. Highlight the encryption key programming (**.ekp**) file you added and click **Program/Configure** (Figure 20).

Figure 20. Programming the Key



8. In the Tools menu, click **Options.** The Options window appears (Figure 21).
9. In the Category list, click P**rogrammer**. You can choose to turn on or turn off Configure volatile design security key (for Stratix III devices) to perform volatile or non-volatile security key programming.
10. Click **OK** to close the Options window.
11. Click **Start** to program the key.
12. The Quartus II software Message window provides information about the success or failure of the key programming operation.

Figure 21. Programming Options Window



How to Perform Single-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later With Command-Line Interface

To perform single-device volatile or non-volatile security key programming using the Quartus II 7.2 SP2 command-line interface through the EthernetBlaster, perform the following steps:

1. Perform Step 1 of "How to Perform Single-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later" on page 18.
2. To determine the EthernetBlaster cable port number that is connected to the JTAG server, type `quartus_jli -n` at the command-line prompt.
3. With the single_ekp.jam file generated in "Step 1: Generate the Encryption Key Programming File and Encrypt Configuration File" on page 6, execute volatile or non-volatile security key programming to a single FPGA by using the following command line:
   - Volatile security key programming:

     ```
     quartus_jli –c<n> single_ekp.jam -aKEY_CONFIGURE
     ```

   - Non-volatile security key programming:

     ```
     quartus_jli –c<n> single_ekp.jam -aKEY_PROGRAM
     ```

     <n> is the port number returned with the -n option.
4. The Quartus II software command-line executable provides information on the success or failure of the key programming operation.

For more information about command-line executable `quartus_jli`, refer to the *Using the Command-Line Executable in Quartus II Software* section in *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*.

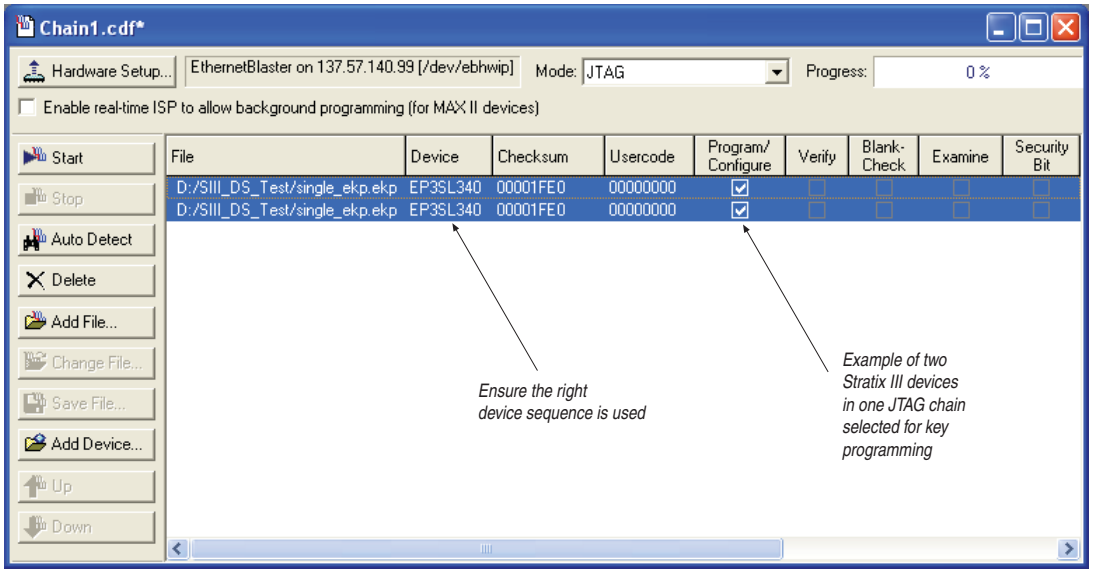
How to Perform Multi-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later

To perform multi-device volatile or non-volatile security key programming using the Quartus II software through the EthernetBlaster, perform the following steps:

1. Repeat Steps 1 – 5 in "How to Perform Single-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later" on page 18.
2. Click **Add File**. The Select Programmer File window displays (Figure 19).
   a. Programming using single-device encryption key programming files:
      i. Type <*single_device*>**.ekp** in the File name field.
      ii. Click **Open.**
      iii. Repeat Steps i–ii for the number of devices in the same chain.
      iv. Highlight the encryption key programming files you added and click **Program/Configure** (Figure 22).

For the correct sequence of the devices in the same JTAG chain, you can use the Auto-Detect option in the Quartus II Programmer.

Figure 22. Multi-Device Key Programming with Encryption Key Programming Files



b. Programming using multi-device **.jam** file:

i. Type <*multi_device*>.jam in the File name field.

ii. Click **Open.**

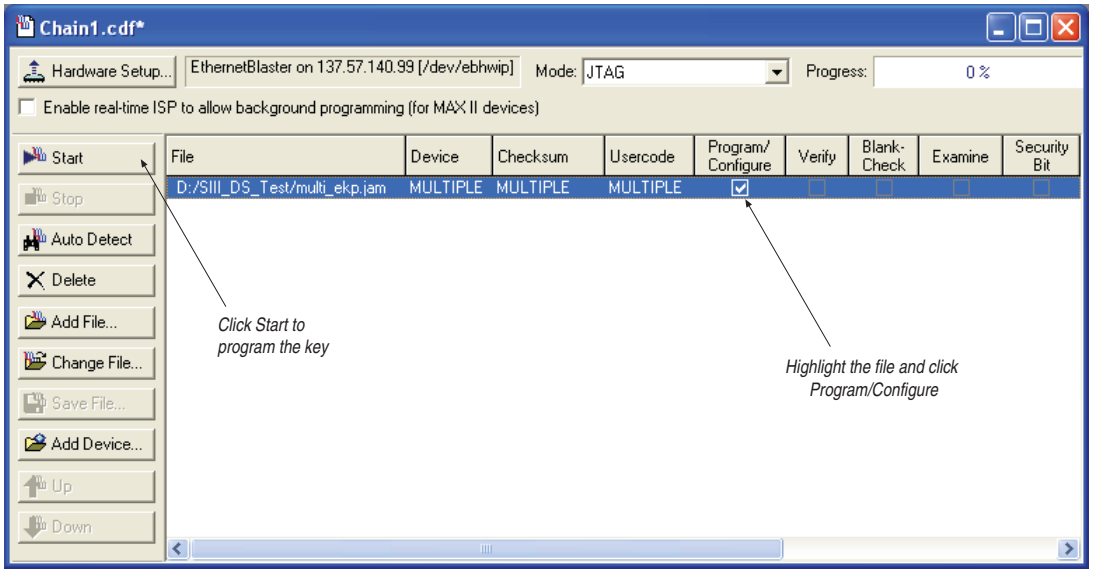
iii. Highlight the **.jam** file you added and click Program/Configure (Figure 23).

3. Repeat Steps 8—10 of "How to Perform Single-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later With Command-Line Interface " on page 21 to perform volatile or non-volatile security key programming.

4. Click **Start** to program the key (Figure 23).

   The Quartus II software Message window provides information on the success or failure of the key programming operation.

Figure 23. Multi-Device Key Programming with .jam Files



How to Perform Multi-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later With Command-Line Interface

To perform multi-device volatile or non-volatile security key programming using the Quartus II software version 7.2 SP2 or later command-line interface through the EthernetBlaster, perform the following steps:

1. Perform Step 1 of "How to Perform Single-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later With Command-Line Interface " on page 21.
2. To determine the EthernetBlaster cable port number that is connected to the JTAG server, type `quartus_jli -n` at the command-line prompt.
3. With the multi_ekp.jam file generated in "Step 1: Generate the Encryption Key Programming File and Encrypt Configuration File" on page 6, execute volatile or non-volatile security key programming for multiple FPGAs by using the following command line:
   - Volatile security key programming:

     ```
     quartus_jli -c<n> multi_ekp.jam -aKEY_CONFIGURE
     ```

   - Non-volatile security key programming:

     ```
     quartus_jli -c<n> multi_ekp.jam -aKEY_PROGRAM
     ```

     <n> is the port number returned with the -n option.

     The Quartus II software command-line interface provides information on the success or failure of the key programming operation.

Key Programming Using JTAG Technologies

The security programming for your design is performed using a **.svf** file (encryption key programming file in .svf format) and a JT 37xx boundary-scan controller in combination with a JT 2147 QuadPod system.

Procedures for the JTAG programming can be found on the JTAG technologies website at www.jtag.com.

Information about creating a .svf file to support multi-device programming is described in "How to Generate the Multi-Device Encryption Key Programming File and Encrypt the Configuration File using Quartus II Software Version 7.2 SP2 or Later " on page 15.

Key Programming Using System General and Other Third-Party Programming Vendors

The System General T9600 and H9600 programming equipment support encryption key programming for Stratix III devices. The System General programming models include the T9600 and H9600 equipment.

The following two files are required:

- JAM STAPL file (encryption key programming file in **.jam** format)
- Encrypted Raw Binary file (**.rbf**) that contains encrypted configuration data. This **.rbf** file is used to further verify the keys that have been programmed into the FPGA. It can be any design file encrypted with the same set of keys used to generate the **.jam** file.

The socket adapter availability for the Stratix III packages is shown in Table 4.

Table 4. Socket Availability for Stratix III Packages *(Note 1)*

Device	Package	Socket Adapter from System General
Stratix III	F484, F780, F1152, F1517, and F1760	Available in the middle or end of 2008

Note to Table 4:

(1) This information is preliminary.

The key programming service can be obtained from most Altera distributors. In general, check with Altera Technical Support for updated programming support status.

Step 3: Configure the Stratix III Device with Encrypted Configuration Data

The final step is to configure the protected Stratix III device with the encrypted configuration file.

During configuration, the encrypted configuration data is sent to the Stratix III FPGA. Using the previously stored security key, the FPGA decrypts the configuration data and uses the unencrypted data to configure itself. Only configuration files encrypted using the correct security key are accepted by the FPGA for successful configuration. Without a correct security key, a stolen encrypted file is useless.

Supported Configuration Schemes

The design security feature is available in all configuration methods, except in the JTAG-based configuration. Therefore, you can use the design security feature in FPP mode (when using an external controller, such as a MAX II device or a microprocessor and a flash memory), or in Active Serial (AS) and Passive Serial (PS) configuration schemes.

Table 5 summarizes the configuration schemes that support the design security feature.

Table 5. Availability of Security Configuration Schemes

Configuration Scheme	Configuration Method	Design Security
FPP	MAX II device or microprocessor, and flash memory	✓ *(1)*
AS	Serial configuration device	✓
PS	MAX II device or microprocessor, and flash memory	✓
	Download cable	✓ *(2)*
JTAG	MAX II device or microprocessor, and flash memory	—
	Download cable	—

Notes to Table 5:

(1) In this mode, the host system must send a DCLK signal that is 4 times the data rate.

(2) The MicroBlaster™ tool is required to execute encrypted PS configuration using a **.rbf** file via ByteBlaster II or ByteBlasterMV™ download cable. For more information about configuration, refer to the Configuration Center. The Quartus II software version 7.2 SP2 supports encrypted **.rbf** file configuration via Altera download cables.

If you are using the MAX II device and flash memory configuration method, refer to the *MAX Series Configuration Controller Using Flash Memory* white paper for more information.

In addition, if your system contains a common flash interface (CFI) flash memory, you can use it for the FPGA configuration as well. The MAX II parallel flash loader (PFL) feature provides an efficient method to program CFI flash memory through the JTAG interface.

For more information about PFL, refer to *AN 386: Using the Parallel Flash Loader with the Quartus II Software*.

In JTAG mode, the configuration data does not use the same interface that is used in the FPP, AS, and PS configuration schemes. Therefore, design security is not available in JTAG-based configurations.

You can use the design security feature with other configuration features, such as the compression and the remote system upgrade features. When compression is used with the design security feature, the configuration file is first compressed and then encrypted in the Quartus II software. During configuration, the FPGA first decrypts and then uncompresses the configuration file.

You can perform Boundary Scan testing or use the SignalTap II logic analyzer to analyze functional data within the FPGA. However, JTAG configuration is not possible after the security key with tamper-protection bit set has been programmed into the Stratix III FPGA.

When using the SignalTap II logic analyzer, you must first configure the device with an encrypted configuration file using PS, FPP, or AS configuration modes. The design must contain at least one instance of the SignalTap II logic analyzer. After the FPGA is configured with a SignalTap II logic analyzer instance in the design and you open the SignalTap II logic analyzer window in the Quartus II software, you simply scan the chain and it is ready to acquire data over JTAG.

Serial FlashLoader Support with Encryption Enabled

Altera provides an in-system programming solution for serial configuration devices called Serial FlashLoader (SFL). The SFL megafunction is available with the Quartus II software version 6.0 SP1 or later. You may instantiate the SFL block to your design and have the flexibility to update the design stored in the serial configuration device without reprogramming the configuration device via the AS interface.

As long as the JTAG interface of the FPGA is accessible, you can use the SFL solution for your application. If you are using the Design Security feature with tamper protection bit is set, SFL solution will not work. Although the JTAG programming is not supported when the tamper protection bit is set, you may instantiate the SFL megafunction in your design and execute the SFL programming for the first time before non-volatile key programming with the tamper protection bit is set on the FPGA.

Perform the following procedures to use the SFL megafunction with the encryption feature enabled in a single-FPGA device chain:

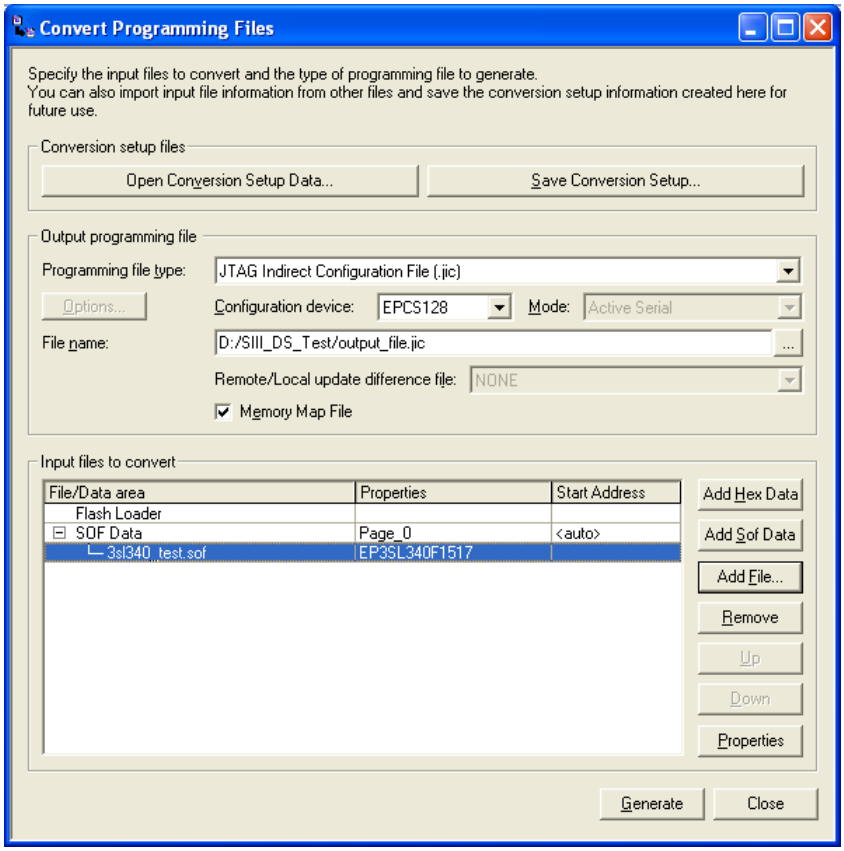
1. Start the Quartus II software.
2. Instantiate the SFL megafunction in your FPGA top-level design.

   For detailed explanation about instantiating SFL megafunction, refer to the *Instantiating SFL Megafunction in the Quartus II Software* section in *AN 370: Using the Serial FlashLoader With the Quartus II Software*.

3. Compile your design by using one of the following options. An unencrypted SRAM Object File (**.sof)** is generated.
   a. In the Processing menu, click **Start Compilation**.
   or
   b. In the Processing menu, select **Start**, and click **Start Assembler.**

4. Follow these steps to convert a **.sof** to a **.jic** file:
   a. In the File menu, choose Convert Programming Files.
   b. In the Convert Programming Files dialog box, scroll to the JTAG Indirect Configuration File (**.jic**) from the Programming file type field.
   c. In the Configuration device field, specify the serial configuration device.
   d. In the File name field, browse to the target directory and specify an output file name.
   e. Highlight the **.sof** data in the Input files to convert section (Figure 24).

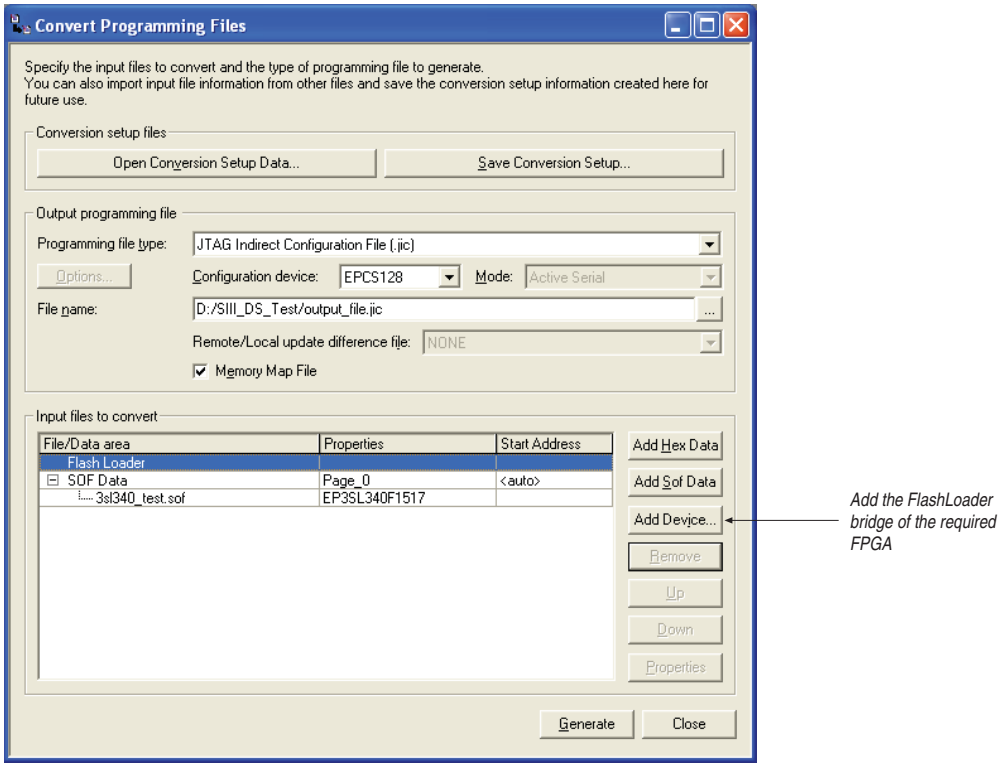
Figure 24. Generation of .jic File



   f. Click **Add File.**
   g. Select the **.sof** file that you want to convert to a **.jic** file.
   h. Click **OK**.
   i. Click on the **.sof** file name to encrypt the **.sof** file.

      To encrypt the .sof file, refer to the Step 7 in "How to Generate the Single-Device Encryption Key Programming File and Encrypt the Configuration File Using the Quartus II 7.2 SP2 or Later " on page 7.
   j. Highlight FlashLoader and click **Add Device** (Figure 25).

Figure 25. FlashLoader



k. Click **OK.** The Select Devices page appears.

l. Select the target FPGA that you are using to program the serial configuration device.

m. Click **OK.**

5. Program the serial configuration device with the encrypted **.jic** file.

For more information about programming the serial configuration device(s) with the **.jic** file that you just created, refer to the procedure in *Programming Serial Configuration Devices Using the Quartus II Programmer & JIC Files* section in *AN 370: Using the Serial FlashLoader With the Quartus II Software*.

6. Program the security keys into the FPGA device.

To program the security keys to a single FPGA, follow the steps in"How to Perform Single-Device Volatile or Non-Volatile Security Key Programming Using Quartus II Software Version 7.2 SP2 or Later" on page 18.

7. The encrypted FPGA is then configured by the programmed serial configuration device.

To program the key with a **.jam** file, you must convert the **.jic** file to a **.jam** file.

For more information about converting a **.jic** file to a **.jam** file, refer to *Converting JIC Files to JAM Files in the Quartus II Software* section in *AN 370: Using the Serial FlashLoader With the Quartus II Software*.

Specifying Configuration Schemes

To enable the design security feature, you must specify the configuration schemes used by setting the MSEL[] pin settings, as shown in Table 6.

Table 6. Stratix III Configuration Schemes When Using Design Security Feature

MSEL2	MSEL1	MSEL0	Configuration Scheme
0	1	0	PS
0	1	1	Fast AS (40 MHz) *(1)*
0	1	1	Remote system upgrade fast AS (40 MHz) *(1)*
0	0	1	FPP with design security feature and/or decompression enabled *(2)*

Notes to Table 6:

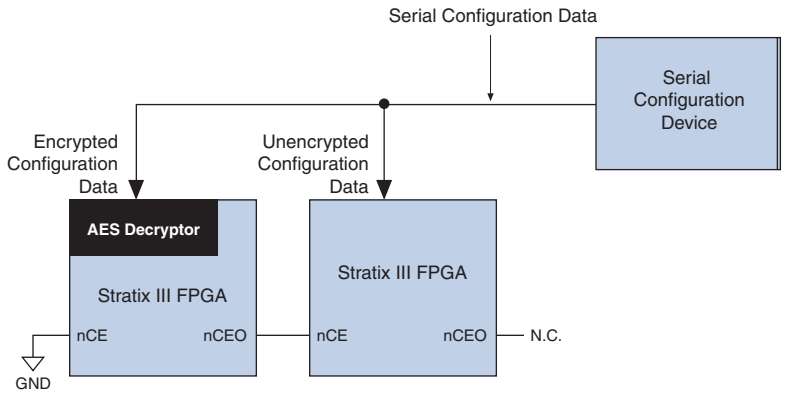
(1) Existing batches of the EPCS1 and EPCS4 devices manufactured on 0.15-µm process geometry support the AS configuration up to 40 MHz. However, batches of the EPCS1 and EPCS4 devices manufactured on 0.18-µm process geometry support only up to 20 MHz. The EPCS16, EPCS64, and EPCS128 serial configuration devices are not affected. For more information, refer to the Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet of the Configuration Handbook.

(2) These modes are only supported when using a MAX II device/microprocessor and flash memory for configuration. In these modes, the host system must generate an output DCLK signal that is 4 times the data rate.

Considerations When Choosing a Configuration Scheme

As shown in Figure 26, in a serial configuration scheme (AS or PS), you can cascade a series of Stratix III FPGAs that use the design security feature (accepts encrypted data) along with devices that do not use the design security feature, such as other Altera FPGAs that accept unencrypted data in the same configuration chain.

Figure 26. Stratix III Series Serial Configuration



When using FPP, the design security feature must be either enabled or disabled for all Stratix III FPGAs in the chain. The design security feature cannot be selectively enabled for individual devices within the chain because of the relationship between the DATA and DCLK signal, which is discussed in the following section. If the chain contains devices that do not support the design security feature, Altera recommends that you use a serial configuration scheme.

However, the DATA and DCLK signal relationship in FPP mode when using the decompression feature is the same as when using the design security feature. If all the devices in the chain use the decompression feature, the design security feature can be selectively enabled or disabled for each device.

DCLK Considerations When Using the FPP Configuration Scheme

The FPP configuration scheme with the design security feature enabled requires that an external host be used, such as a MAX II device or a microprocessor to control the flow of the DATA and DCLK signal to the Stratix III FPGA.

DCLK is the clock source that is used to clock the configuration process. Configuration data is received on the DATA[7..0] pins.

If you are using the Stratix III design security feature with the FPP configuration scheme, the external host must be able to send a DCLK frequency that is four times the data rate.

The 4× DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency supported in Stratix III devices is 100 MHz and produces a maximum data rate shown in Equation 1:

Equation 1.

$$100\ \text{MHz}/4 \times 8\ \text{bits} = 200\ \text{Mbps}$$

When using the design security feature, the first configuration data byte is latched on the FPGA on the first DCLK rising edge. Subsequent data bytes are latched four clock cycles after each previous data byte (that is, data is latched on the first, fifth, ninth DCLK rising edge, and so on). The configuration clock (DCLK) speed must be below the frequency specified in Equation 1 to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

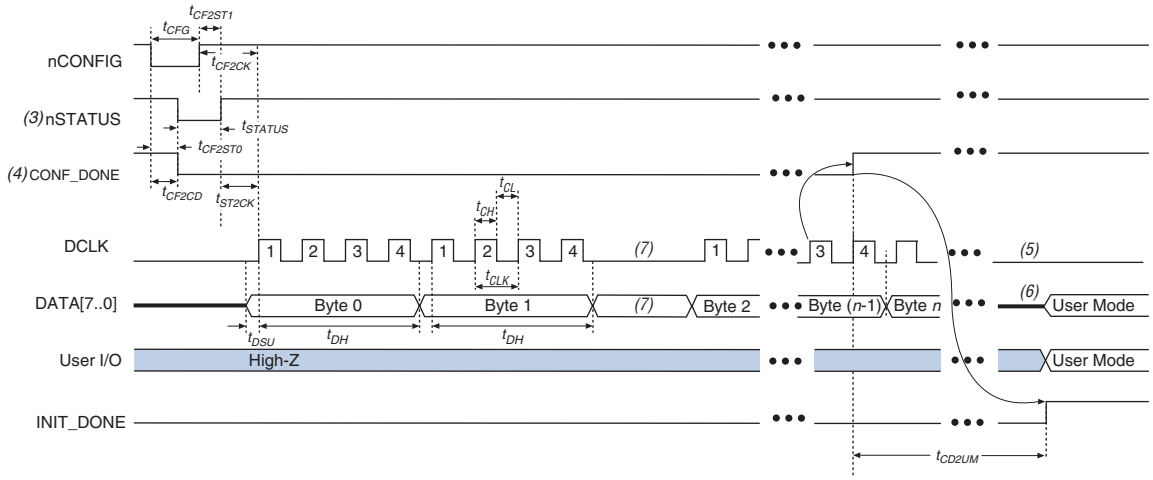
If you need to stop DCLK while using the Stratix III design security feature with the FPP configuration scheme, it can only be stopped three clock cycles after the last data byte was latched into the Stratix III FPGA.

Stopping DCLK three clock cycles after the last data byte was latched into the Stratix III FPGA gives the configuration circuit enough clock cycles to process the last byte of latched configuration data. When the clock restarts, data must be present on the DATA[7..0] pins prior to sending the first DCLK rising edge.

Timing Waveform for FPP Configuration

Figure 27 shows the timing waveform for the FPP configuration that uses a MAX II device or a microprocessor as an external host. This waveform shows the timing when the design security feature is enabled.

Figure 27. FPP Configuration Timing Waveform with Design Security Feature Enabled *(Note 1)*, *(2)*



Notes to Figure 27:

(1) This timing waveform should be used when the design security feature is used.

(2) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic-high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(3) Upon power-up, the Stratix III FPGA holds nSTATUS low during the power-on reset (POR) delay.

(4) Upon power-up, before and during configuration, CONF_DONE is low.

(5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.

(6) DATA[7..0] pins are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.

(7) If needed, DCLK can be paused. When DCLK restarts, the external host must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

Timing Parameters with Design Security Feature Enabled

Table 7 defines the timing parameters for Stratix III FPGAs that implement the FPP configuration scheme with the design security feature enabled.

Table 7. FPP Timing Parameters for Stratix III FPGAs with Design Security Feature Enabled *(Note 1)*, *(2)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Unit
t_{POR}	POR delay	12	100	ms
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	100 *(3)*	µs

Table 7. FPP Timing Parameters for Stratix III FPGAs with Design Security Feature Enabled *(Note 1)*, *(2)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Unit
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	100 *(3)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	100	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	5	—	ns
t_{DH}	Data hold time after rising edge on DCLK	30	—	ns
t_{CH}	DCLK high time	4	—	ns
t_{CL}	DCLK low time	4	—	ns
t_{CLK}	DCLK period	10	—	ns
f_{MAX}	DCLK frequency	—	100	MHz
t_{DATA}	Data rate	—	200	Mbps
t_R	Input rise time	—	40	ns
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	20	100	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (4,436 × CLKUSR period)	—	—

Notes to Table 7:

(1) This information is preliminary.

(2) Altera recommends that you use these timing parameters when the design security feature is used.

(3) This value can be obtained if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The minimum and maximum numbers apply only if the internal oscillator is selected as the clock source for initializing the device.

US Export Controls

The US export controls for the Stratix III family of devices are generally classified under the US Export Control Classification Numbers (ECCN) 3A001.a.7 or 3A991.d. Although Stratix III devices can perform decryption, the export control classification of the devices does not change as the decryption capability is only used to protect the configuration bitstream. Altera's Quartus II software development tools (version 7.2 SP2 or later), which encrypt the configuration bitstream, have been formally classified under US ECCN 5D002 c.1 and subject to export under license exception ENC as a "retail" commodity to most countries. You can contact opexp_imp@altera.com with any export-related questions.

Conclusion

As FPGAs move from being glue logic to implementing critical system functions, the need for design security is increasing. Stratix III FPGAs address this concern by providing built-in design security. Stratix III FPGAs not only offer high density, fast performance, and cutting-edge features to meet your design needs, but also protect your designs against IP theft and tampering of your configuration files.

Document Revision History

Table 8 shows the revision history for this application note.

Table 8. Document Revision History

Date and Revision	Changes Made	Summary of Changes
March 2009 v1.1	■ Updated Table 3 andTable 5. ■ Updated Notes to Table 3 and Table 5. ■ Updated handnote in "Hardware Requirements" section. ■ Updated Example 1 and Example 2 ■ Updated "Introduction" ,"Supported Configuration Schemes" ,"Serial FlashLoader Support with Encryption Enabled " and "Considerations When Choosing a Configuration Scheme " section.	—
April 2008 v1.0	Initial Release.	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support