Specify and Program Security Settings and Keys with SmartFusion2 and IGLOO2 FPGAs

Implementation Guide

November 2013

Introduction

In order to create a secure design it is critical to manage security keys, the fundamental cryptographic primitives that allow us to implement secure systems, as well as the various security settings used to protect the design from interference and potential theft. SmartFusion®2 and IGLOO®2 devices have a wealth of features and capabilities that provide automatic protection to secure information, making the most common Design Security requirements easy to implement. (For more information on how easy it is to secure your SmartFusion2 or IGLOO2 design from IP theft, refer to the "Other Resources" section under "Easy Design Security" in "To Learn More" section" at the end of this paper).

When it is important to create further security boundaries and capabilities a detailed understanding of the Detailed Security Model, used by SmartFusion2 and IGLOO2 devices, can be critical to getting the most out of these advanced devices. The security model describes how you can further protect sensitive data (as in Data Security applications where a database needs to be protected from overwriting—either accidental or by malicious intruders), or restrict access to sensitive features of the design (as in limiting the sources for configuration bitstream updates to a specific communication port so that unauthorized updates, perhaps from a network-based attack or virus are prevented) to go beyond the robust Design Security features supported automatically by the devices.

This white paper provides a detailed description of all the Security Keys and special security settings (like Passcodes and Lock-Bits) available to the user for creating more advanced security capabilities. The capabilities of the SmartFusion2 and IGLOO2 devices are extensive and can be somewhat daunting, initially. A short summary of the SmartFusion2 and IGLOO2 Simplified Security Model (described in the "Introduction to the SmartFusion2 and IGLOO2 Security Model"—which is a mild 'prerequisite' to reading the white paper and is included in the "To Learn More" section") serves as an introduction to the more complete Detailed Security Model. Once the Detailed Security Model is described, an example software flow showing the methods used to specify and program example Security Keys, Passcodes, and Lock-Bits is provided. This white paper is detailed enough so that the user can refer to it during a design to easily pick and choose the specific features required for their implementation.

Simplified Security Model Overview

Figure 1 on page 3 depicts the simplified security model used in both SmartFusion2 SoC FPGAs and IGLOO2 FPGAs. The System Controller is the heart of the security system and manages all programming, verification, design security key-management, and related operations. It interfaces with the Security Keys, the embedded Nonvolatile Memory array, and the flash FPGA fabric configuration memory. During normal operation it can also provide optional cryptographic user services.

For the purposes of this paper we will provide just a short description of the key elements of the Simplified Security Model as a lead-in to the more detailed description that follows.

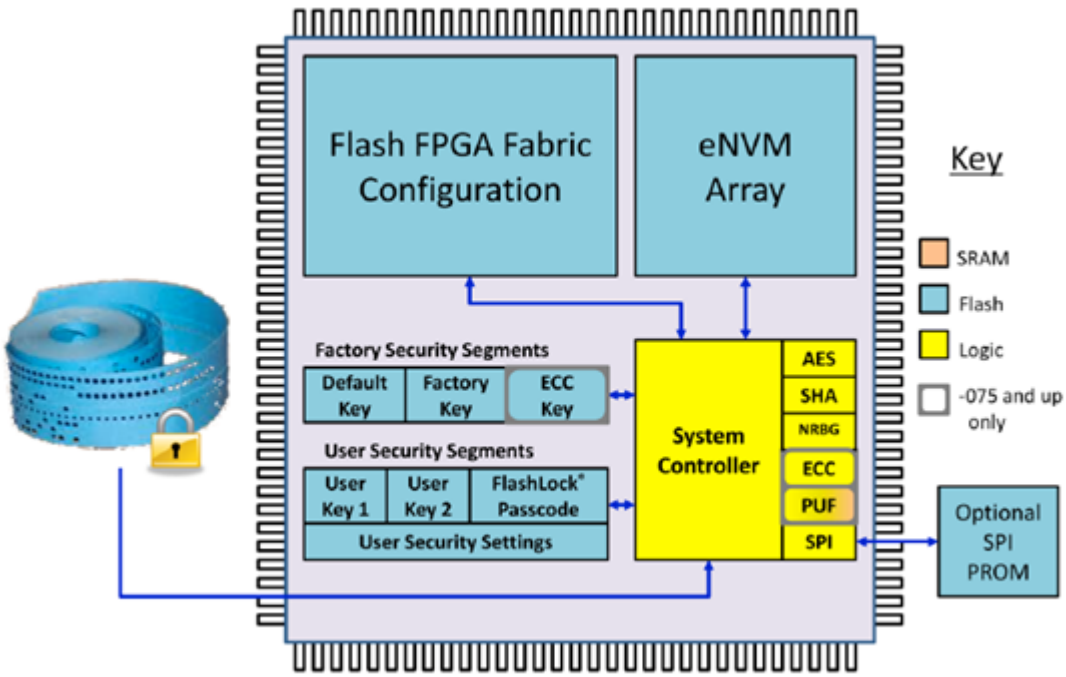

Figure 1: **Simplified Security Model Diagram**

System Controller

The System Controller manages all programming, verification, design security key-management, and related operations. The System Controller is a dedicated fixed-function hardened processor reserved for these functions, and is not reconfigurable in normal operation. Its programming and runtime operation are determined by a dedicated immutable metal-mask ROM. It also includes some dedicated scratch-pad SRAM, and hardware accelerators for specific cryptographic function support. Many high-level functions are supported by the System Controller that simplify the implementation of security features. For example, the Security Controller can program the device autonomously (without additional design effort) using one of the communications ports (SPI is shown as an example in Figure 1 above) and an external bitstream. The focus of this paper is on the security segments and locks. Refer to the resources listed in the "To Learn More" section" at the end of this paper for more information on the various security services implemented in the controller. In particular, the video "SmartFusion2 and IGLOO2 Cryptography Services" shows how services are invoked in an example design. Additionally, for more information on device programming refer to the Programming Users Guide listed in the "To Learn More" section of this paper.

eNVM Array

The embedded Nonvolatile Memory (eNVM) storage array within SmartFusion2 and IGLOO2 devices provides additional security capabilities to the designer. This memory array can be organized as separate pages each configured for specific functions. For example, pages can be designated as write-protected to make it easy to control sensitive data. Additionally, a novel NVM integrity check mechanism can be used to check the reliability and security of the user specified ROM segments in a device automatically upon power-up, or upon demand. These types of security features make it easy to provide higher levels of security without additional design effort or the use of User Logic.

Flash FPGA Fabric Configuration

Microsemi flash-based FPGA configuration memory cells are located within the FPGA fabric and directly control the routing switches and look-up tables used to implement the users design. This means the bitstream isn't exposed on every power up as is done by SRAM-based FPGAs. SmartFusion2 and IGLOO2 FPGAs only need to be configured once during manufacturing. If field updates are required, as an added layer of security they only accept encrypted bitstreams. Allowing only encrypted bitstreams makes it much more difficult for the bitstream format to be reverse engineered by researchers, malicious users, or adversaries. It also allows devices to be programmed by contract manufacturers without worry about possible theft from overbuilding, cloning, or reverse engineering and allows for secure remote updates.

Security Segments Store Keys, Passcodes and Locks

The heart of the security system is the robust nonvolatile storage of security keys. There are three user security segments ("User Keys 1", "User Keys 2", and "User Lock"). The first holds a user bitstream encryption key (UEK1). It can be the root key for encrypting and decrypting bitstreams and for authentication of bitstreams. This segment also holds the FlashLock® passcode. It is used to unlock access to the three user security segments, if re-entered and matched correctly. A correct match allows the user to update the segment contents. It also unlocks many of the user lock-bits, until such time as the device is reset; allowing operations such as programming or verification to continue that may have been disallowed by the stored lock-bit settings.

The second user keys security segment stores a second user encryption key (UEK2). This can be used interchangeably with UEK1. This segment has its own passcode, which allows overwriting of the key and passcode, after the passcode is successfully matched (and before the device is reset). This passcode does not unlock anything in the User Key 1 or User Lock segments. Use of the second user keys segment is strictly optional and can be used for additional features, like programming or updating a subset or class of products (with UEK2) versus all products (where UEK1 would be used) that may be in the field.

The User Lock segment holds the majority of lock-bits for setting user security options. These include options for configuring both design and data security. Many of the lock-bits are overridden if the FlashLock passcode is matched (assuming that other security options allow the FlashLock passcode to be active).

The FlashLock passcode is programmed as part of the initial user key injection procedure, and is stored in the User Key security segment in hashed form. If it is re-entered and matched later, it will temporarily unlock the three user segments, allowing changes to the keys, passcodes, and security settings (that is, lock-bits) stored there. The Permanent FlashLock mode can be used to turn a SmartFusion2 or IGLOO2 device into an OTP device that can't be inspected or modified, and is thus even more secure from possible intrusion.

Protected Storage in the eNVM Storage Array

The embedded Nonvolatile Memory (eNVM) storage array within SmartFusion2 and IGLOO2 devices provides additional security capabilities to the designer.

- The memory array can be organized as separate pages each configured for specific functions and optionally designated as write-protected.
- An automatic or user initiated
- NVM integrity check can verify the reliability and security of all the nonvolatile memory segments using a SHA-256 digest. If the digest is ever 'wrong' it can be flagged and responded to.

This short review of the key elements of the Simplified Security Model for SmartFusion2 and IGLOO2 FPGAs provides a good background on which we can now layer the much more involved Detailed Security Model appropriate for actually implementing (specifying and programming) features using the various Security Keys, Passcodes, and Lock-Bits available to the user.

Following the definitions for the Detailed Model, the software flow used to implement some example features, will be used to show how easy it is to create a more feature rich and robust Security Design.

Detailed Security Model for SmartFusion2 and IGLOO2 FPGAs

The three main classes of NVM that are the possible targets for programming in SmartFusion2 and IGLOO2 devices, are shown in the center column of Figure 2, below. From top to bottom they are: 1) the eNVM array(s), 2) the security segments, and 3) the FPGA Fabric (including configuration-related features not only of the logic look-up tables and routing fabric, but also I/O, DSP, SRAM, and other blocks). All the data in a segment is erased at the same time, however segments may be erased and written separately from other segments giving more flexibility for custom security use models.

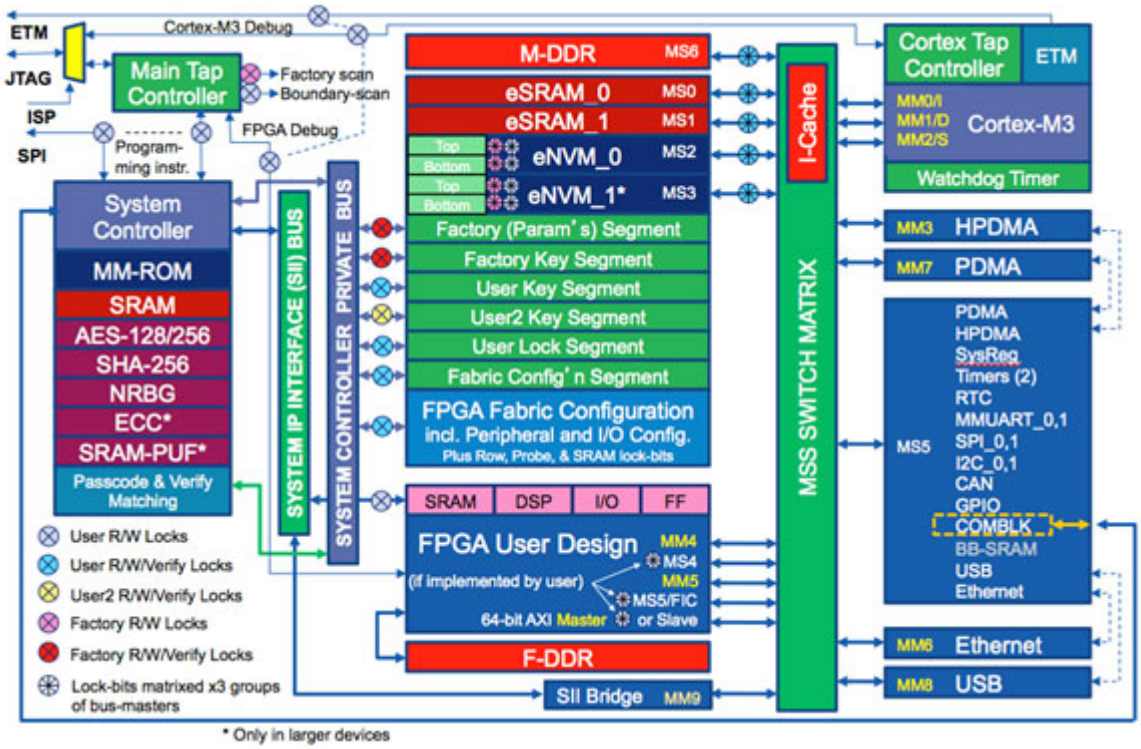

Figure 2: **Detailed Security Key and Settings Model Diagram**

In the left column is shown the System Controller and its major peripherals including hardware cryptographic accelerators. It operates, as previously described in the Simplified Security Model overview, by performing all the programming of the device nonvolatile memory. In the right column is shown the ARM®Cortex™-M3 Microcontroller Subsystem (MSS). The MSS is present in SmartFusion2 devices, but is largely removed in the IGLOO2 device parts. This has only a minor impact on the overall security model. The small circles with X's in them indicate some of the functions that can be disabled through the setting of various security options, often referred to as Lock-Bits. These lock-bits are nonvolatile.

As can be seen in the middle column of the figure, there are two eNVM blocks (two on larger devices and one on smaller devices (see the appropriate Product Briefs for specifics)) identified in Blue. Directly below these are the six security NVM segments identified in Green. The FPGA fabric configuration block, shown in light Blue, is directly below the security segments.

The actual FPGA user design is shown next and includes configuration-related features not only of the logic look-up tables and routing fabric, but also I/O, DSP, SRAM, and other blocks. Typically, there are lock-bits associated with each NVM segment that controls access and security features associated with the data in each segment. The above figure may look very complex at first glance, but if we break up the description to cover each key element of the diagram, and cover them one at a time the various capabilities will be easy to understand.

eNVM Blocks Security Features

The top pages of the eNVM array are reserved by Microsemi for use by the System Controller. In the four smallest SmartFusion2 devices, sixteen pages (the top one-half of the top sector) are reserved for the Device Certificate and for storing the digest for the static User portion of the eNVM. These pages are protected against overwriting by Factory page write-protect bits. In the larger devices the 64 page (the top two sectors) reserved portion of the eNVM is completely private to the System Controller for both reading and writing.

No other bus master can access these two sectors in the second eNVM block. Both the Primary and Secondary Device certificates are stored there, as well as the digests for the private region, and a digest for the static User portion of each of the two eNVM blocks. The Factory Activation Code and Key Code, and the User Activation Code and Key Codes (in “S” class devices), required for the working of the SRAM-PUF, are stored in the private section of the eNVM. Lock-Bits associated with the eNVM blocks are described in more detail in the Lock-Bits section of this paper.

Security Segments Detailed Description

Table 1 below gives a brief description of each security NVM segment. All encryption keys are encrypted (using AES-256 with a device- and key-specific key encryption key) while at rest, and all Passcodes are hashed (using SHA-256), and the hash stored rather than the plaintext passcode. Each of the main segments is described in the section below (Factory segments first followed by the User segments) along with descriptions of the specific keys and passcodes. Detailed description of the lock-bits follows this section.

Table 1: **Security NVM Segment Description**

Segment Name	Primary Contents	Size (bits)	Comments
Factory (Parameters)	Factory Passcode (FPK)	256	Hashed
	Factory Serial Number (FSN)	64	First (LSB) half of DSN
	Factory Parameters	~1500	Calibration Data
	Factory Lock-Bits	~20	–
Factory Keys	Serial Number Modifier (SNM)	64	Second (MSB) half of DSN
	Factory Key (FK)	256	Encrypted
	Factory Pseudo-PUF (FPP)	256	Used for PUF Emulation
	Factory ECC Private Key (SKFE)	384	Encrypted
	Default Key-Loading Key (KLK)	256	Encrypted
User Keys 1	User (FlashLock) Passcode (UPK1)	256	Hashed
	User Key 1 (UEK1)	256	Encrypted
	Debug Passcode (DPK)	256	Hashed
User Keys 2	User Passcode 2 (UPK2)	256	Hashed
	User Key 2 (UEK2)	256	Encrypted

Table 1: **Security NVM Segment Description (continued)**

User Lock	JTAG USERCODE	32	32 IEEE 1149.1 JTAG Instr.
	User Lock-Bits	~120	
Fabric Config.	Design Version	16	
	Version Back-Level	16	
	Design ID	256	

Factory Segments

The two Factory segments (“Factory”, and “Factory Keys” in the above table) are written by Microsemi before shipping devices. These Factory keys and passcodes are injected in a secure manner by making sure secure data is always encrypted when outside the device and that security keys are generated inside FIPS140-2 level 3 certified hardware security modules (or by the device itself) and encrypted for transport into the devices during wafer probe testing.

Factory Parameters Segment

Factory Passcodes (PPK and FPK)

The Factory Passcodes are required to put the device into Factory Test Mode. First, the 128-bit Factory Passcode Passkey (PPK) must be matched. If correct, it allows attempted matching of the second (main) 256-bit Factory Passcode (FPK). If that is also matched, Factory Test Mode is entered. This allows in-depth testing of the device by Microsemi, such as for failure analysis purposes.

It is possible to disable the Factory Test Mode, even permanently if desired. The lock-bit that disables Factory Passcode matching is set by default whenever a User first loads his user keys. This bit is protected by the FlashLock®passcode, which can if matched, roll back this bit. Another option is that the FlashLock passcode can be permanently disabled, and thus also indirectly permanently disabling Factory Passcode matching by preventing such roll-back.

The 128-bit passcode is common to a relatively large population of devices and is hardcoded (in encrypted form) in the device during fabrication. The main 256-bit Factory Passcode is unique in every device produced, and is stored (encrypted) in the Factory Key security NVM segment. Only Microsemi knows these passcodes or has the proprietary knowledge on how to enter and use Factory Test Mode. Various lock-bits are available to the user to disable and control Passcode matching and are described in the lock-bits section of this document.

There is another user lock-bit option that will permanently directly prohibit Factory Passcode matching. In devices where this lock-bit has been set the device can never again enter Factory Test Mode, and failure analysis is impossible. There is no built-in mechanism to clear a permanent lock once set, short of zeroization, which is also under the control of User lock-bits.

There is another lock-bit option that requires the use of the one-time-use passcode protocol for matching any passcodes. If this option is used, the Factory Key must also be known to enter Factory Test Mode.

Factory Serial Number (FSN)

The Device Serial Number (DSN) is comprised of two parts, the Factory Serial Number (FSN, the LSBs) and the Serial Number Modifier (SNM, the MSBs). The reason for this is that the two parts are stored in different Factory security segments to support various zeroization options. The 64-bit Factory Serial Number uniquely identifies a device. The Serial Number Modifier can be used to tell if a device has been ‘zeroized’ and a new set of Factory keys installed.

Factory Parameters

Factory parameters contain calibration data used for programming and other device operations and are not relevant to security operations of the device.

Factory Lock-Bits

Factory lock-bits are used by Microsemi to control access to device features and are typically only used by Microsemi. Any Factory lock-bits that impact user operations will be described in the appropriate section.

Factory Keys Segment

Serial Number Modifier (SNM)

The SNM is the MSB of the DSN and is described in the paragraph directly above.

Factory Key (FK)

The Factory key is the preferred key to use for loading the User's keys in the four smallest SmartFusion2 devices when security is important. This 256-bit key has a unique value in every device produced by Microsemi. If used for loading the User's keys, it is selected as the root key for encryption and authentication of the bitstream component used to load the User's keys. After the User's security settings are loaded the Factory key is automatically disabled for encryption purposes by a User lock-bit without any action required on the part of the User.

Since the Factory Key is a symmetric key, the programmer must know a derived/related key (for every device) in order to prepare bitstreams that can be decrypted by the devices, or to verify the device "knows" the Factory Key. This is done by installing encrypted key databases – distributed by Microsemi – into the Microsemi-supplied programmer tool's hardware security module (HSM). Each key database distributed by Microsemi is unique, thus preventing anyone else with one from decrypting another User's bitstream files.

Pseudo-PUF Secret (FPP)

The so-called Pseudo-PUF Secret (FPP) is a 256-bit secret generated by the device's own non-deterministic random bit generator (NRBG) during the manufacturing of the device at Microsemi, and stored permanently in the Factory Keys security NVM segment. In the four smallest SmartFusion2 "S" class devices, it is used for PUF emulation using a challenge-response protocol. It is never exported from the device or the System Controller and security subsystem, or during the manufacturing and key injection process steps at Microsemi, or internally to the user of the device.

Factory ECC Public Key Pair (SKFE, PKFE)

In the larger devices, those having the Elliptic Curve Cryptography (ECC) hardware accelerator, Microsemi uses an HSM on its production line to generate and inject a completely random unique-per-device 384-bit private ECC key (SKFE) that is stored in the Factory Keys security NVM segment in encrypted form. From this private key the associated 768-bit ECC public key (PKFE) is calculated, and this public key is certified in the secondary Device Certificate which is stored in the part of the eNVM array that is private to the System Controller. (This is only to prevent it from being overwritten, as the certificate is considered public data, and can be exported on demand). The Device Certificate is X.509 compliant and is signed by a Microsemi key used just for that purpose.

The ECC key pair can be used for establishing a shared symmetric key using the Elliptic Curve Diffie-Hellman (ECDH) protocol followed by a key derivation function. The 256-bit derived key becomes the root key which can then be used either for encrypting/authenticating a bitstream (such as for injecting User keys and security settings in a new device), or for authenticating the device using the key confirmation challenge-response protocol. Once the User's security settings are loaded, the Factory ECC key pair is automatically disabled for bitstream encryption purposes with a User lock-bit, without any action required on the part of the User.

Default Key-Loading Key (KLK)

The default Key-Loading Key (KLK) is used to load User keys and security settings in situations where high levels of security are not required. One such situation could be where programming is done in a completely trusted secure facility with cleared personnel and stringent data handling and protection processes in place.

The 256- bit default Key-Loading Key is common to a relatively large population of devices of the same type, and is “known” within the programming tool software. This makes it easiest to use, but not as secure as the other options, having a “software” rather than a “hardware” level of protection.

After the User’s security settings are loaded, the default Key-Loading Key (KLK) is automatically disabled by a User lock-bit reserved for this purpose, without any action required of the User. Typically the User security settings (User Lock security segment) and the first User Key and the FlashLock passcode (in the User Keys 1 security segment) are loaded with the same bitstream file.

User Key Segments

There are three User security segments (“User Keys 1”, “User Keys 2”, and “User Lock”). The User Keys 1 segment holds a User bitstream encryption key (UEK1). It can be the root key for encrypting and decrypting bitstreams, and for authentication of bitstreams, once it is loaded. This segment also holds the FlashLock® passcode (UPK1). It is used to unlock access to the three User security segments, if re-entered and matched correctly. A correct match allows the User to update the segment contents. It also unlocks many of the User lock-bits, until such time as the device is reset; allowing operations such as programming or verification to proceed that may have been disallowed by the stored lock-bit settings. Also in this segment is the Debug Passcode (DPK). It unlocks just certain lock-bits related to FPGA fabric and Cortex™-M3 debugging features (if present), but does not unlock nearly the number of lock-bits that the FlashLock passcode does. It does not allow the user to overwrite keys, passcodes, or security settings, for example. It too stays in effect only until the device is reset.

User Keys 1 Segment

User Passcode 1 (FlashLock® Passcode/UPK1)

The User Passcode, also known as the FlashLock® Passcode or UPK1, is a 256-bit passcode used for unlocking the three User security segments. (Passcodes are never used for encryption, only for escalating the privileges of whoever matches it successfully). It is loaded along with the other User keys and passcodes using an encrypted bitstream and stored (after being hashed) in the User Key security segment. When it is subsequently successfully matched it unlocks many of the User-set security lock-bits. This allows the contents of the three User NVM security segments to be erased and rewritten. A number of other options become temporarily available that may have been locked with User lock-bits. The device remains unlocked until reset or powered down. Changes to the security segment NVM settings do not take effect until reset or the next power-up cycle.

The Permanent FlashLock mode can be used to turn a SmartFusion2 or IGLOO2 device into an OTP device. This mode is considered quite secure because it disables most programming, verification, and debug operations. There are additional optional, layered, security settings (additional lock-bits) that when used in conjunction with Permanent FlashLock mode can provide even higher levels of security:

- The factory test mode can be permanently disabled
- The programming ports can be partially disabled
  - The JTAG boundary scan instructions can be disabled (EXTEST, SAMPLE/PRELOAD, CLAMP, HIGHZ, and EXTEST2).
  - Virtually all JTAG and SPI programming instructions can be disabled so they are ignored when parsed, even before other lock-bits (for example, for disallowing overwriting of the NVM) are checked.

User Key 1 (UEK1)

The User Key (UEK1) can be the root key used for bitstream encryption and authentication. Like many of the other possible root key options, it can also be used in the key verification protocol and as part of the keying material for creating encrypted one-time-use passcodes. In practice, it is mostly used for encryption of bitstreams used for field updates.

Debug Passcode (DPK)

The Debug Passcode is a 256-bit passcode that, when matched, unlocks certain lock-bits controlling the debug features of the device, namely the Cortex-M3 debugging features (for example, JTAG debugger, Single-Wire Viewer, and the Embedded Trace Macrocell), and the FPGA fabric debugging features (live probes). It is loaded along with other User keys and passcodes using an encrypted bitstream, and stored, after being hashed, in the User Keys NVM security segment. It can be changed if the main FlashLock passcode is matched, assuming other security settings allow it.

User Keys 2 Segment

User Passcode 2 (UPK2)

User Passcode 2 unlocks the User Keys 2 security NVM segment for erasing and writing when matched. It is a 256-bit passcode that is first hashed and then stored in the User Keys 2 security segment. It is loaded along with other User keys and passcodes using an encrypted bitstream. It is used primarily for changing the value of User Key 2 (UEK2), or itself, if desired.

User Key 2 (UEK2)

User Key 2 (UEK2) is completely interchangeable with User Key 1. Its use is completely optional. Having a second key is convenient for some use models. Because it is in a separate security segment from User Key 1, it can be written and erased at different times offering more flexibility. One important possible use model is that one of the User keys could be programmed with the same key across an entire project, and the other User key could be programmed with a unique value on a device-by-device basis.

Fabric Configuration Segment

The Fabric Configuration segment holds miscellaneous data required to support the FPGA fabric. Of interest from a User security viewpoint is that it holds the User Design Version, the User Version Back-level (used to avoid the programming of back-revision bitstreams—a potential attack threat) and the Design ID, all chosen by the User. The Design ID is public data that can be read from the device.

Keys Not Stored in Nonvolatile Memory

Several important keys are never actually stored in nonvolatile memory and are ephemeral in nature (they are created by an internal process and never actually loaded into NVM storage). These keys can be even more secure that other keys since they are even more difficult to attack since they vanish when power is removed. These keys are described below.

Bitstream Payload Key (KIP)

The Bitstream Payload Key (KIP) is actually never stored in nonvolatile memory in the device. It is an ephemeral key that is transported into the device using the Authorization Code key wrapper mechanism, used to decrypt all or part of the remainder of a bitstream, and then erased. This can be useful when portions of the configuration bitstream need to be different from device to device, but you don't want to create a different large bitstreams for each device.

Factory Key SRAM-PUF ECC Key Pair (SKFP, PKFP)

In the three largest devices, those having the Elliptic Curve Cryptography (ECC) hardware accelerator and the Quiddikey™ SRAM-PUF, Microsemi injects a completely random unique-per-device 384-bit private ECC key (SKFP) generated by a Microsemi HSM that is imported (in encrypted form) and enrolled as an extrinsic key by the SRAM-PUF. This entails creating a base activation code and a key code that are stored in the private section of the eNVM, with the option to be permanently deleted.

From the private key, the associated ECC public key (PKFP) is calculated. This 768-bit public ECC key is certified in the primary Device Certificate. Once the certificate is generated and saved in the device this key pair is permanently deleted from the HSM, leaving the only copy of the private key in existence protected by the SRAM-PUF in the device.

Since the security of the associated private key is rooted in the SRAM-PUF, which is analogous to a silicon biometric, the User can prove with a very high assurance level that the Device Certificate and the device itself go together.

As with the ECC key stored in the security NVM (see SKFE and PKFE described previously), this key pair can be used in an ECDH protocol followed by a key derivation function, to establish a 256-bit shared symmetric key; which can then be used either as the root key for encrypting or decrypting and authenticating a bitstream, or for assuring the authenticity of the device by performing a key confirmation protocol. Using this key pair is the preferred method of injecting encrypted User keys in new devices in larger devices, and provides a very high level of security.

SRAM-PUF Symmetric User Design-Security Key (KUS)

In the three largest “S” class devices, that is, those having both the SRAM-PUF feature and having user access to the premium security feature set, the user may enroll a symmetric key (KUS) reserved for design security purposes using the SRAM-PUF. This gives it the best available key protection. When the power to the PUF’s dedicated SRAM block is removed, which is nearly all the time, the PUF secret disappears.

In these devices where this feature is available, this key is interchangeable with User Key 1 and User Key 2. It is just like having a third User key. It is imported using the same methods as the other User keys, and potentially used in the same ways. The only difference is that it is protected by the SRAM-PUF with its activation code and key code stored in the Private eVNM sectors, rather than being stored in a security segment encrypted using the device’s key-encryption key.

SRAM-PUF User Design-Security ECC Key Pair (SKUP, PKUP)

Also in the three larger “S” class devices, the User may enroll an Elliptic Curve Cryptography (ECC) asymmetric private-public key pair. The private key is a 384-bit intrinsic key, generated using the entropy and the intrinsic properties of the SRAM-PUF. Since it is generated internally by the FPGA and never exported, it is not known to the User or the programmer tools. When the private key (SKUP) is generated and enrolled by the PUF, the public key (PKUP) is also computed from it. The public key is exported along with a keyed tag which can be used to authenticate it with a cryptographic chain that goes back to the Microsemi-signed Device Certificate. The public key is also stored in the System Controller’s private area of the eNVM. There is a system service which will return the public key internally to the design running in the FPGA.

One-Time Key (OTK, for Zeroization Recovery)

The one-time key is a short-lived key generated by the device using its own random bit generator and transmitted encrypted by a 4096-bit RSA public key first to the Programmer, and ultimately to Microsemi where it is used in the preparation of a new Factory key bitstream that can load a new set of Factory keys and Device Certificate(s) into a ‘zeroized device’. Once the new keys are loaded the One-Time Key is destroyed.

Lock-Bits Detailed Description

Now that the various security keys and passcodes have been described, it is appropriate to describe the various Factory and User lock-bit available to further control security functions within the device. Lock-bits are generally used to prohibit some function. For example, all NVM segments have lock-bits which prevent the erasing and overwriting of their own contents. These may be stored in the segment itself, or in another segment. The vast majority of User lock-bits are stored in the appropriately named User Lock segment.

Factory lock-bits are set and locked in the two Factory security segments by Microsemi before shipping the parts. Some Factory lock-bits prohibit the same function as a User lock-bit. In that case, if either one is set the function is disabled.

Many lock-bits can be temporarily unlocked using the appropriate passcode(s) assigned to that bit; some can only be changed by erasing and overwriting the NVM of the security segment to which they belong using an encrypted/authenticated bitstream.

If lock-bits are unlocked using a passcode, it is just temporary, until the next device reset, JTAG reset, or power-down. Any permanent changes to the NVM segments (that is, from a bitstream) take place after the reset, or at the next power-up cycle.

Although a lock-bit may be referred to in the singular in this document, all lock-bits are actually stored with physical redundancy.

Lock-Bits Detailed Description

The User Lock segment holds the JTAG USERCODE and a majority of lock-bits for setting User security options. These include options for configuring both Design and Data Security. Many of the lock-bits are overridden if the FlashLock passcode is matched (assuming that others of these are allowing the FlashLock passcode to be active). Although in most cases, the User Key 1 and User Lock segment are programmed at the same time by having these as separate segments, this does not necessarily have to be the case. It may be useful to refer back to Figure 2 on page 5 while reading this section, if it helps your understanding to visually identify the function blocks the described lock-bits control.

JTAG USERCODE

The JTAG USERCODE is a 32-bit value stored in the User Lock security NVM segment. It is read out of the device using the IEEE JTAG 1149.1—defined optional instruction by the same name. The USERCODE value is public information, and can be used by the User for any purpose. It is often used to identify the function of one FPGA design as differentiated with other designs, which may be loaded in the same or similar part number devices from Microsemi. It is loaded using an encrypted/authenticated bitstream along with the User's security settings. It can be changed if the FlashLock passcode is matched, and other security settings allow it. It can also be read internally using a system service.

Security Segment Erase/Write and Verify Lock-bits

This group of locks prevent, erasing and writing of the User security segments. The User Key, User Key 2, and User Lock security segments are automatically locked against erasing and overwriting after they are first programmed, without any action on the part of the User. These are amongst the locks that are temporarily overridden in the event of a FlashLock passcode match.

There is a lock-bit that prevents key verification either using the bitstream verification method or the key confirmation protocol. This will provide another layer of defense against the keys inadvertently leaking out, but has the disadvantage that once set, the keys cannot be verified to be programmed correctly, except using the digest-based verification method. The key confirmation protocol, in particular, is essential for checking that the Device Certificate belongs to the device it is purported to go with, so it would be necessary to do this check before locking key verification out.

Passcode Locks

The Factory Passcode, the User Passcode (also known as, the FlashLock® passcode, or UPK1), and the User 2 Passcode (UPK2) can be permanently locked out using User-set lock-bits. If the User permanently locks out the Factory Passcode, there will be no possibility of rolling back the security to enter Factory Test Mode, and therefore, no failure analysis is any longer possible. Permanently locking the FlashLock passcode is generally accompanied with setting a number of other security settings too. For example, field updates and verification are normally locked at the same time. The bundle of locks is referred to as Permanent Lock Mode, or sometimes as One-Time Programming (OTP) Mode.

Fabric NVM and eNVM Erase/Write Lock-bits

The FPGA fabric and the eNVM array have lock-bits that prevent them from being overwritten with a new encrypted/authenticated bitstream.

These locks are amongst those that are temporarily unlocked by a match of the FlashLock passcode. These are automatically set as part of the Permanent Lock Mode (also known as, OTP Mode) bundle of locks.

Fabric NVM and eNVM Verify Lock-bits

There are two methods of verification of the design configured in a SmartFusion2 or IGLOO2 FPGA. The first is the legacy method where the encrypted/authenticated bitstream is resubmitted to the device a second time. To verify the bitstream, the bitstream is read, authenticated, and decrypted, but instead of loading the NVM with it, the existing NVM is compared to the incoming bitstream and a pass-fail indication is returned after the whole bitstream has been processed. This method may be disabled within any specific bitstream file when the bitstream is created. Also, there are User lock-bits which can disable verification of the FPGA Fabric NVM or the eNVM using this method. In the three larger devices having two eNVM controllers, there is an additional lock-bit for the second eNVM.

The preferred method of verification in SmartFusion2 and IGLOO2 devices, uses message digests. When a design is loaded into NVM a digest of each of the segments loaded is computed. These digests can be used to tell if the programming was successful. Digests can be set to be computed automatically on power-up, or on-demand, over the FPGA fabric or the static portions of the eNVM through a system service call. There is a lock-bit that can disable verification using this method.

Key-Mode Lock-bits

Each key mode has a lock-bit which disables it. Some of these are set automatically. For example, in a new part one of the Factory keys may be used to load the User keys in encrypted form. This can be done by anyone with possession of the part. After the User keys are loaded, the Factory keys are automatically disabled, leaving only the User key modes in operation. Keys that are not loaded are also automatically disabled. The User may disable any additional key modes they wish. The key mode lock-bits are not unlocked by the FlashLock passcode. It is required to match the FlashLock Passcode to allow them to be erased.

Lock-bit to Require One-Time-Use Encrypted Passcodes

There is a lock-bit that disables all plaintext passcode matching. This effectively, by a process of elimination, requires the use of a one-time-use encrypted passcode protocol. This lock-bit affects all five passcodes in the device: the Factory Passcode (FPK) and the Factory Passcode Passkey (PPK), the User Passcode (also known as, the FlashLock® Passcode, UPK1), the User Passcode 2 (UPK2), and the Debug Passcode (DPK). Transmitting any crypto-variable (CV) in plaintext is risky and frowned upon. Use of this lock-bit is highly recommended whenever you are concerned about protecting your design Intellectual Property from being used for design security protocols. The cryptographic system services lock-bits are not unlocked by the FlashLock Passcode. It is required to match the FlashLock Passcode to allow them to be erased.

Hardware Firewall Lock-bits

In the premium “S” class devices there are lock-bits that control access to the eSRAM blocks, the eNVM block(s), and the MDDR memory controller. These are matrixed by groupings of bus masters: Cortex-M3 bus masters, DMA controllers, and FPGA fabric bus masters. They block read or write traffic from a specific group of bus masters to a specific memory slave (or portion thereof).

Specify and Program Security Keys, Security Settings, and Lock-Bits

Now that the multitude of security keys and lock-bits have been described in detail, it is useful to show how they are selected for programming into a SmartFusion2 or IGLOO2 device. Some of the security keys, settings, and lock-bits can be selected and defined within the Security Policy Manager—a section of the Libero SoC tool.

Let’s see, in detail, how the Security Policy Manager is used to select and specify security settings, security keys, and lock-bits within Libero SoC. After that, we will look at some example lock-bits that can be modified through the design to illustrate how this can be done and when it makes sense to do it.

It is important to note that you should check to see which device key and security features are supported in your software release, since new capabilities are typically being added with each new release.

Using the Policy Manager

Simply click on the Policy Manager icon within the Libero SoC tool, as seen in the middle of Figure 3 below. This will bring up the Security Policy Manager tool.

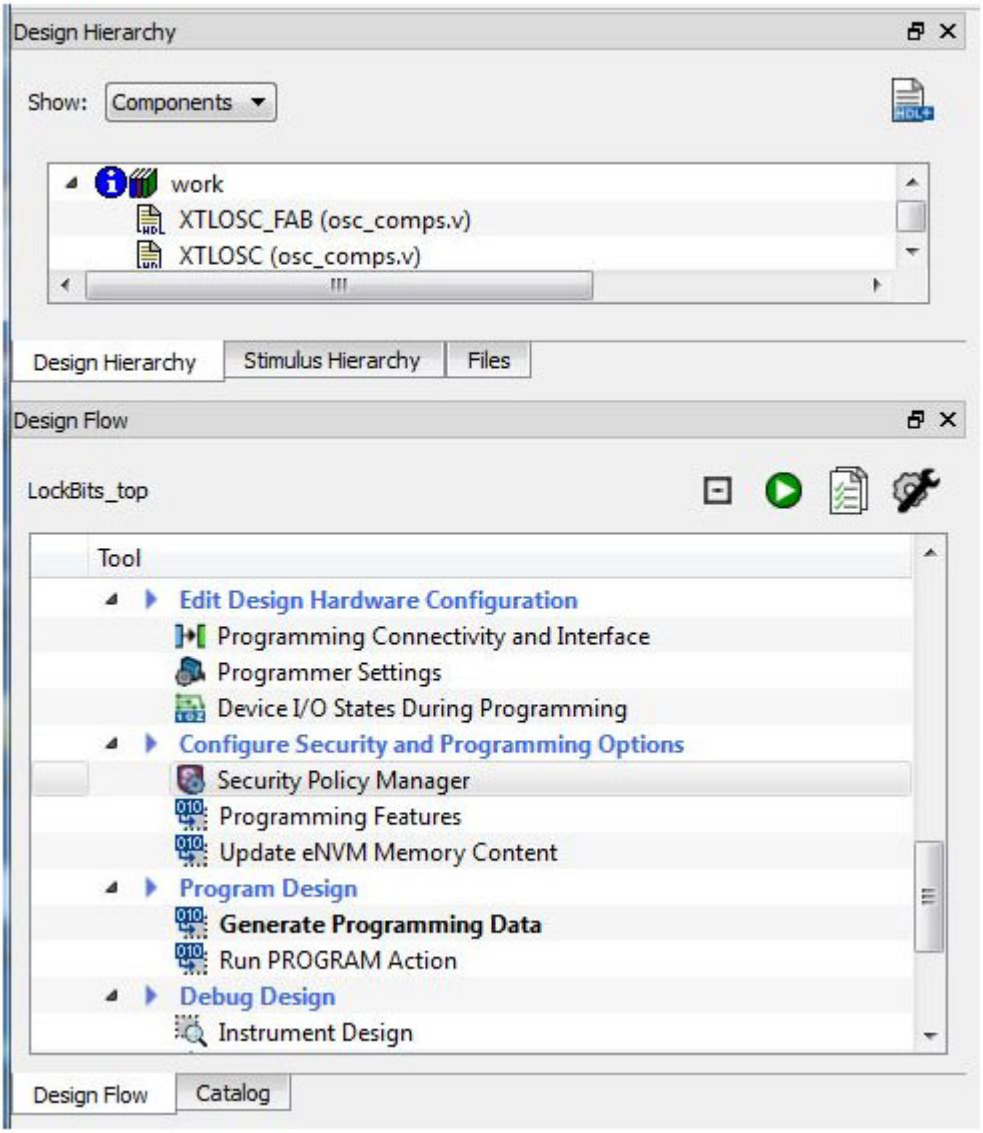

Figure 3: **Security Policy Manager Icon Within Libero**

The Security Policy Manager allows you to select various security settings and the security key values within your device. As seen in Figure 4 on page 15, the Security Policy Manager GUI allows the user to define the programming location and the field update more of operation.

Programming of the security keys and settings are done at a secure facility while the design can be programmed at either a secure or an unsecure facility.

As seen at the bottom of the diagram, the security keys are generated within the policy manager (by clicking on the lock icon) and use a secure key generation algorithm within Libero SoC.

In the secure facility, the Security Keys and Policies, and Programming Data File are all generated and used in the trusted manufacturing site.

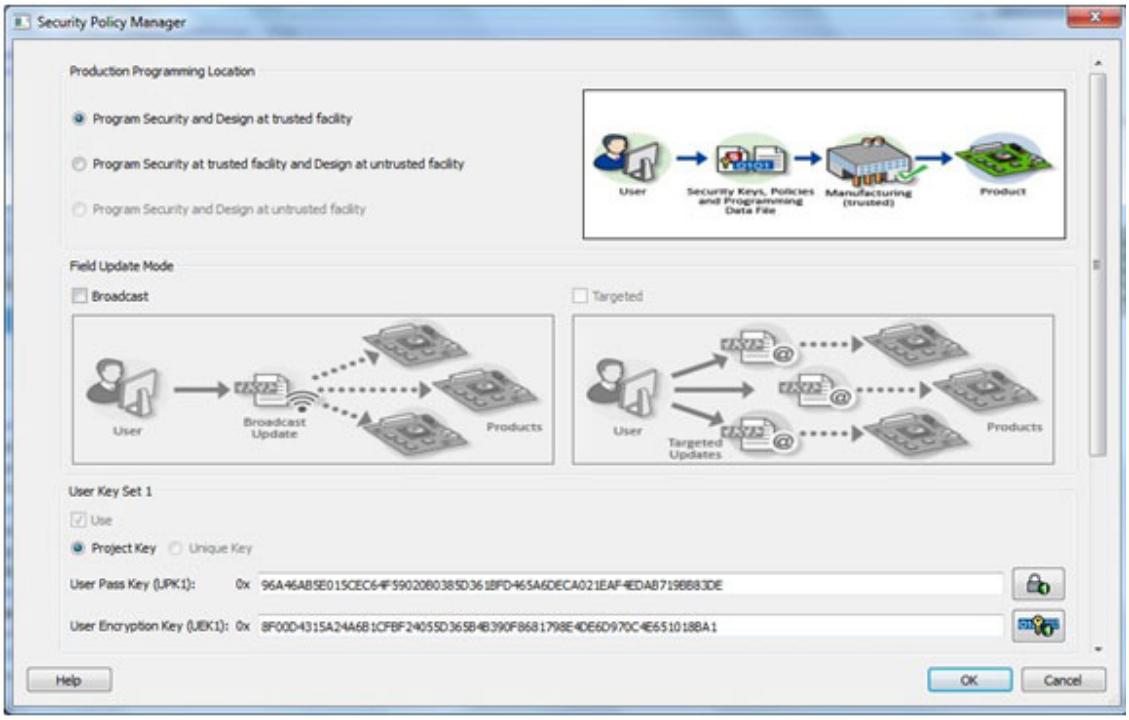

Figure 4: **Security Policy Manager Icon Within Libero**

If the preferred production flow is to use an untrusted facility, perhaps a contract manufacturer, the process shown in Figure 5 on page 16 is used. In this case, the Security keys and Policies file is generated separately from the encrypted Programming File. The Security Keys and Policies are programmed into devices at a trusted facility (perhaps a distributor) and these now secure devices are then sent to the contract manufacturer.

The Programming Files are applied to the secure devices and because these files are encrypted and can only be programmed into the devices with the correct security keys, the manufacturing process is secure.

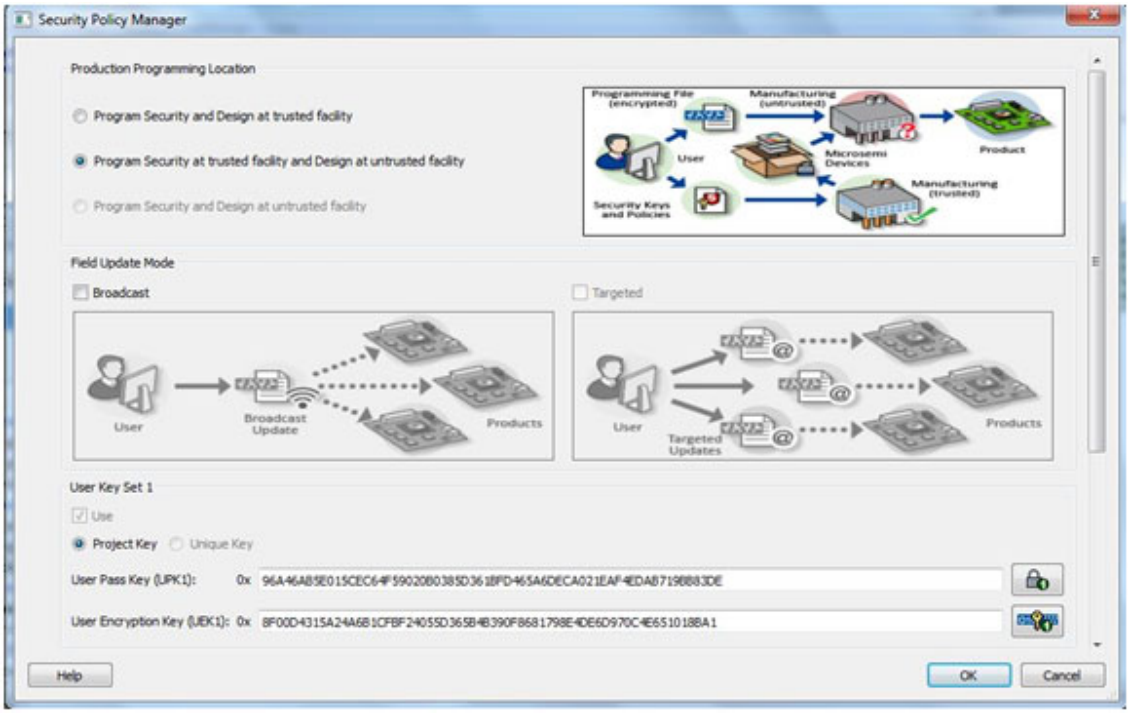

Figure 5: **Programming the Design at an Untrusted Facility**

Once the Production Programming location is determined, the Field Update mode can be selected. Currently only the broadcast mode is available and by selecting it, both User Key Set 1 and User Key Set 2 become available for use as Project Keys.

As illustrated in Figure 6, the project key will be applied to all units in the project.

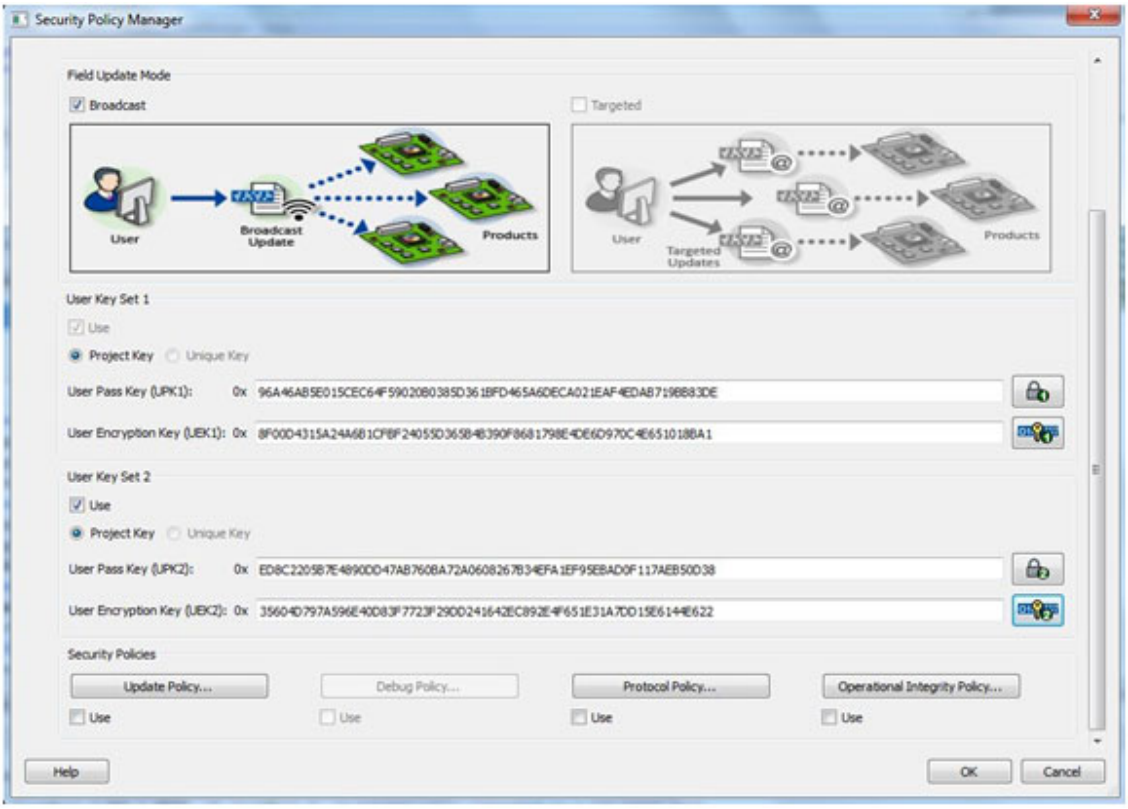

Figure 6: **Broadcast Field Update Mode**

The Update Policy is used to define how remote updates are to be protected and managed. The associated dialog box is opened by clicking on the Update Policy button in the lower left of the illustration. The Update Policy dialog box, given in Figure 8 on page 19, shows the various policy options available for selection. For example, fabric and eNVM updates can be protected by security keys through the appropriate selections in the top two drop-down boxes. Back-level protection can be enabled through the check box and the version entries to disallow updates to older revisions. This eliminates possible tampering through an update attack; attempting to install an old version of the firmware, perhaps one with known weaknesses the attacker wishes to exploit. Additionally, programming interfaces can be locked, requiring the correct security key to enable remote programming. Finally, all updates can be disabled by checking the Permanent Security check box.

This freezes the security settings and disables updates to the fabric and the back-level protection settings.

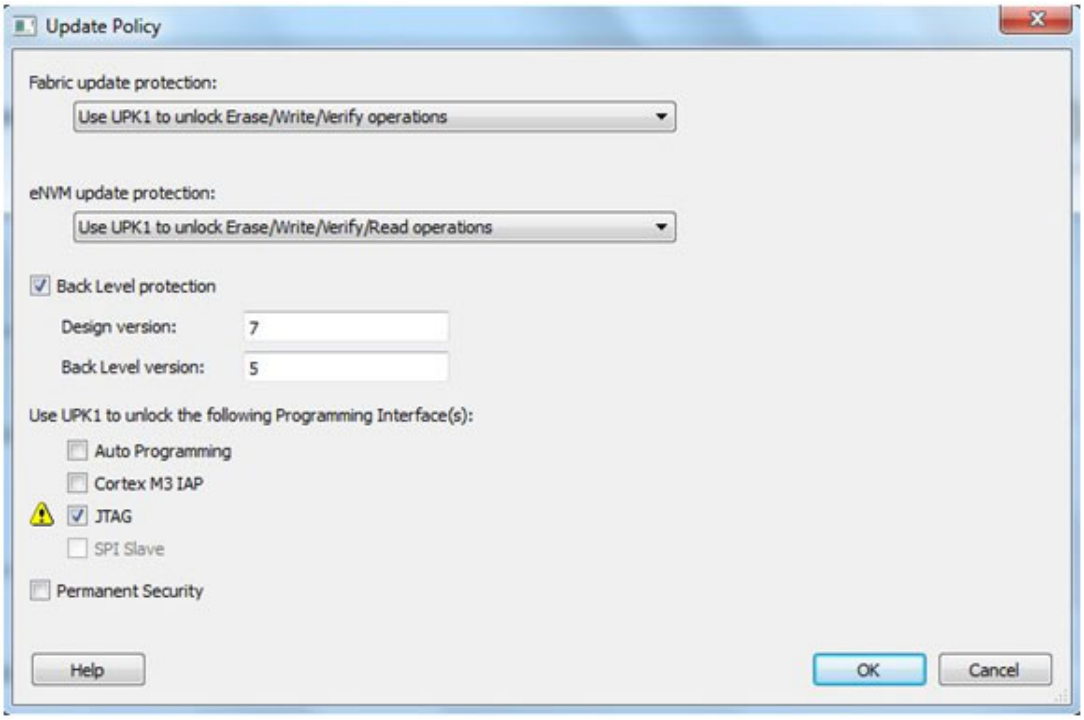

Figure 7: **Update Policy Dialog Box**

Some of these settings result in the programming of the appropriate lock-bits to enable or disable the featured required to implement the desired policy. The Policy Manager keeps all the settings consistent, saving the user from the time consuming and error prone selection of the needed lock-bits ‘by hand’. The Policy Manager generated lock-bit settings are ‘correct by construction’. Notice that the Policy Manager can also post ‘reminder’ notification icons when a specific policy setting is selected (as for the JTAG selection in Figure 7). This icon, when hovered over, reports that the JTAG programming interface is disabled for programming and that this lock is protected by UPK1. These reminders help clarify results of specific selections, reducing potential ‘cockpit errors’.

Additional policy settings are available, as shown at the bottom of Figure 6 on page 17. The dialog boxes opened by clicking on the Programming Protocol or Operational Integrity Policy buttons are shown in Figure 8 on page 19. The Protocol Policy allows Programming to be disabled from either or both of the User Keys.

If both are disabled, updates are not allowed.

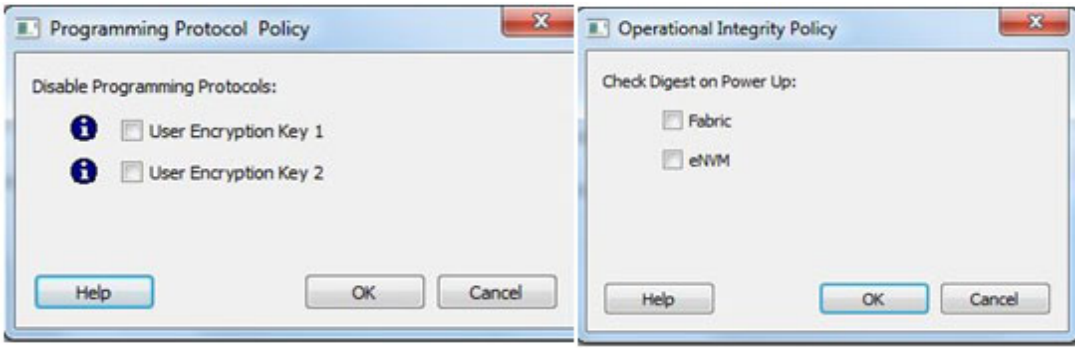

Figure 8: **Additional Policy Dialog Boxes**

The Operational Integrity dialog box controls the Check Digest. This checks the integrity of the Fabric and/or eNVM by computing a 'hash' function over all the storage associated with that element and comparing it to a previously generated check value. The lock-bits associated with this feature enable or disable the automatic execution of these checks during power up.

Once these selections have been made, the Policy Manager can be exited by clicking **OK**, and all the security settings and keys will be saved. The appropriate values will be added to the programming bit streams, as selected through the Policy Manager, when the programming files are generated by the user. These settings will all be active once the design powers-up and stays as defined, unless they are modified during the device operation.

Conclusion

SmartFusion2 and IGLOO2 devices have a wealth of advanced security features that make it possible to address many common security scenarios automatically. When even more advanced security capabilities are required, the user can specify and program a variety of additional security features using extra Security Keys, Passcodes, and Lock-Bits. This implementation guide described these features in detail and showed an example software flow appropriate for implementing these capabilities in your design. We intend this paper to be detailed enough so that you can use it as a guide during your design, and to better pick and choose the specific features required for their implementation. To learn more about security topics related to those explored in this paper, explore the items listed below or the many other items available on our Security website.

To Learn More

Easy Design Security

1. It's Easy to Protect Your Embedded System from Theft
2. Securing Your Embedded System Life Cycle
3. Securing Your Supply Chain Life Cycle

Other Resources

1. Introduction to the SmartFusion2 and IGLOO2 Security Model
2. SmartFusion2 and IGLOO2 Cryptography Services
3. SmartFusion2 SoC FPGA Programming Users Guide
4. Truth in Randomness: Practical Insights on Randomness, the Nature of the Universe, etc
5. Protect FPGAs from Power Analysis

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

55900172-0/11.13