



## FPGA Design Security

**Florence Beaujean**  
**Applications Engineer, Lattice Semiconductor**

The reprogrammability of FPGAs makes them a very attractive option for system design when compared with ASICs, which have a high development cost. Furthermore, FPGAs offer increasingly robust performance and functionality, including 32-bit soft processors, SERDES, DSP blocks and high performance interfaces. Low cost FPGAs can now even meet the needs of high-volume applications. FPGAs facilitate an efficient response to the constraints of Time-to-Market by offering fast development times (market demands mean that development times are shortening) and remote update capabilities.

The outsourcing of production, field updates and the remote reconfiguration of firmware, however, can lead to FPGA copying, cloning or piracy. For some businesses, this can be a critical problem, because algorithms that are essential to the company's competitive advantage can be exploited. It is therefore very important to consider the security of FPGA designs.

There are two types of FPGAs: volatile FPGAs, which are based on SRAM and which are configured using an external boot device, and non-volatile FPGAs, which maintain the configuration in memory and so do not require an external device. SRAM FPGAs are volatile components: if their power supply is cut then the configuration is erased and reconfiguration is necessary before they can be operational again. They can be programmed by a processor or by the JTAG port using an external memory such as an SPI or parallel Flash. In terms of security, these components are vulnerable: it would be easy for a pirate to obtain the FPGA bitstream once the system has powered up. Flash and antifuse FPGAs are non-volatile components. These are programmed on power-up without using any external memory devices. These non-volatile components have the highest level of security. The two principal limitations of antifuse FPGAs are: 1) they are

not reprogrammable and 2) they have limited performance and memory capacity. On the other hand, Flash FPGAs, like SRAM FPGAs, are reprogrammable.

There are several types of piracy. *Cloning* replicates the system components' behavior even if the internal logic is not known. A low cost SRAM FPGA can be cloned easily by intercepting the FPGA bitstream and copying the configuration. *Reverse engineering* requires an understanding of the FPGA's logic so that it can be modified to meet the pirate's needs. Non-protected bitstreams can be analysed in order to replicate the original design, or the components can be de-capped to analyze the content. Yet another piracy technique that is widely used is to *produce more systems than the customer has ordered*. There is nothing to stop an unscrupulous subcontractor from doing this and then selling the surplus systems to the market without benefiting the patent owner. The final method of piracy is *theft of services*. This occurs when the pirate thwarts the system's security in order to gain access to a particular service, such as satellite TV programs.

There are various security measures available to combat piracy, the easiest of which is to use a security bit to prevent the FPGA's configuration data from being intercepted. With a security bit, if a pirate were to try to read the configuration data it would read zero. This type of protection can be found in all FPGAs. However, it is relatively easy to copy the configuration if the boot-up file is located outside the component, as is the case with standard SRAM FPGAs. The security bit provides a very good level of protection if the configuration data is held within the component itself, as with non-volatile FPGAs.

The fact that SRAM and Flash FPGAs, unlike antifuse FPGAs, are reprogrammable allows the user to adjust the application or to give the FPGA an entirely new functionality. Furthermore, it is now possible to modify the configuration file of the FPGA, and thus the system, remotely. However, reconfiguration, which is an advantage for some applications, can actually become a security weakness since it is possible for a pirate to intercept and obtain the new bitstream. This can be overcome by protecting the bitstream during its transfer.

Although considered to be more secure than SRAM FPGAs, Flash FPGAs nevertheless have their own shortcomings. They have limited density and memory capacity and are also limited in functionality and in I/O speed.

Lattice Semiconductor has developed a new and innovative response to these shortcomings by combining the advantages of the two technologies:

- Low cost SRAM for performance and memory capacity
- Flash for internal storage of the reconfiguration data

Lattice's unique flexiFLASH technology, which enables the combination of SRAM and Flash in the same FPGA, offers several possibilities which allow the combination of these different aspects:

- Bitstream configuration
- Remote re-configuration with minimal system disruption
- Data protection and component locking

## Design Security in Complex Systems

Engineers face the challenge of constructing increasingly complex systems.

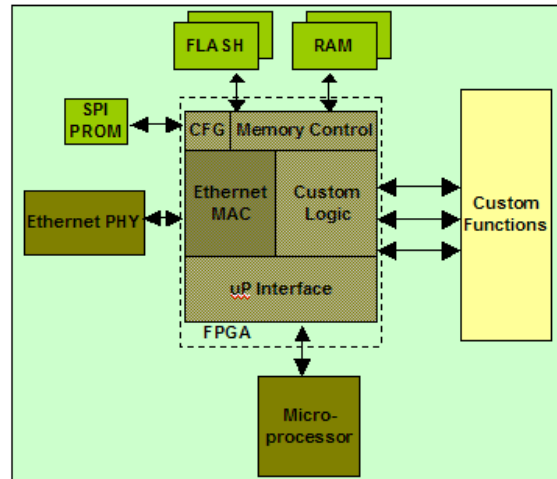


Figure 1

Figure 1 shows a system in which a microprocessor is linked to an FPGA. The microprocessor has an Ethernet connection via an external PHY and an integrated MAC in the FPGA. The microprocessor also contains the user logic, which differentiates this product from competitive products. The FPGA is configured from an SPI Flash. The Flash and RAM memories contain the microprocessor program and the data for the system. If the system needs to be updated, the data is sent via the Ethernet connection. In order to protect the application, the FPGA data and bitstream are encrypted; however, the data and instructions are transmitted unscrambled to the microprocessor, so a pirate would be able to view this data and copy it. A possible solution is to implement a soft microprocessor in the FPGA. It would then be possible to have access to the FPGA's internal memory banks, which are invisible from the outside. These memory banks can be utilised to store important algorithms and data.

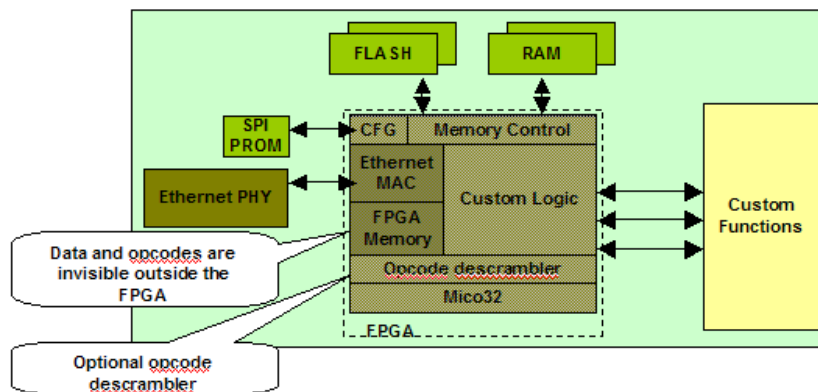
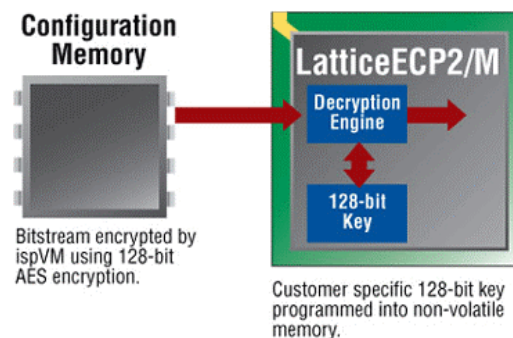


Figure 2

Figure 2 shows the same application, but with the processor integrated into the FPGA. The soft microprocessor used in Figure 2 is a LatticeMico32. The descrambler is created by the designer, and therefore the scrambling is confidential. The use of a descrambler allows instructions, which can then be encrypted, to be stored in an external memory. In order to encrypt the bitstream, Lattice offers, on several of its FPGAs, a 128-bit AES key.



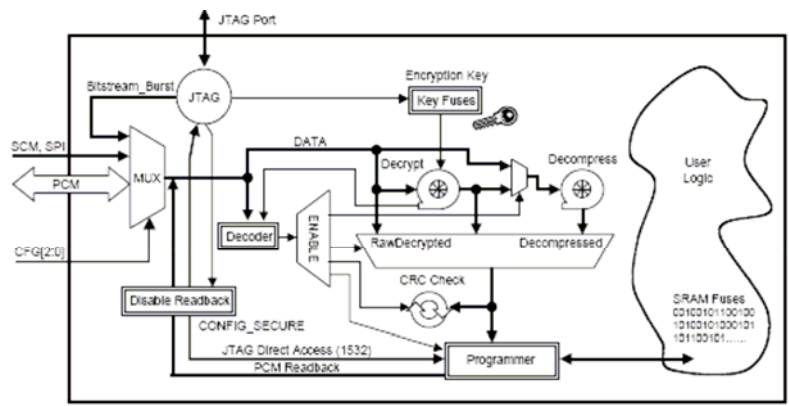
**Figure 3 - Design protection using a 128-bit AES key**

The engineer should first develop the FPGA code using ispLEVER (version 7.0 or later), Lattice’s software design tool suite. Once the synthesis, mapping, place and route and simulation check stages are complete, the engineer will generate a bitstream and program the board using ispVM System (a Lattice programming tool) for final validation. When the designer is satisfied with the system’s behaviour, it is time to protect the design. The bitstream is encrypted using either the Lattice ispLEVER tools or the Lattice programming tool ispVM System, with the user selecting the 128 bit key. The bitstream is encoded either with a hexadecimal (from 0 to F, not case sensitive) or an ASCII key (all alphanumeric characters supported as well as spaces, case sensitive). The bitstream then can be loaded into the configuration memory using any method of non-encrypted file coding.

The key should now be stored in a programmable memory zone. The programming is performed by the component’s JTAG port. The FPGA now can be configured only by a bitstream encrypted by this key. Lattice FPGAs can be programmed using either the sysCONFIG interface or the JTAG interface. The sysCONFIG interface allows the user to re-enter the data using a mass configuration mode or a Flash SPI, or in parallel using a parallel configuration mode. The JTAG port, which conforms to IEEE 1149.1 and IEEE 1532 standards, allows the data to be programmed in Bitstream-Burst (or Fast Program) mode, or in 1532 mode. The JTAG port is used to program the AES 128-bit key in the component. No special mode is required to maintain this 128-bit key in the FPGA.

The use of an encoded bitstream in a Lattice FPGA prevents the component from being re-read. However, there are mechanisms to ensure that the configuration is operating

correctly. When the bitstream is unencrypted, the FPGA performs a CRC. If the programming is incorrect, the DONE signal stays at 0 and the INITN signal moves to 0. The JTAG Usercode register remains accessible to the user. When the FPGA decrypts the bitstream, it registers the bitstream usercode which can then be used to register an FPGA application version number.



**Figure 4**

Figure 4 shows the data path of an encrypted bitstream. When the data enters the FPGA, the decoder reads the preamble and all data preceding this is ignored. If the decoder detects an encrypted file, and if the FPGA key has not been programmed, the data is blocked and the DONE signal stays at '0' (meaning that the configuration failed). If the key is programmed, the FPGA checks the alignment preamble that indicates that all the following data should pass through the decoder. The FPGA then checks the standard preamble, which informs it whether the data has been compressed or not. If the data is not compressed, it is sent straight to the decoding unit. If it is compressed it is transmitted first to the decompression engine and then to the decoding unit. The SRAM is programmed once the CRC has been verified. The decompression and decryption engines are deactivated when the DONE bit is activated to allow other JTAG chain components to receive their configuration data.

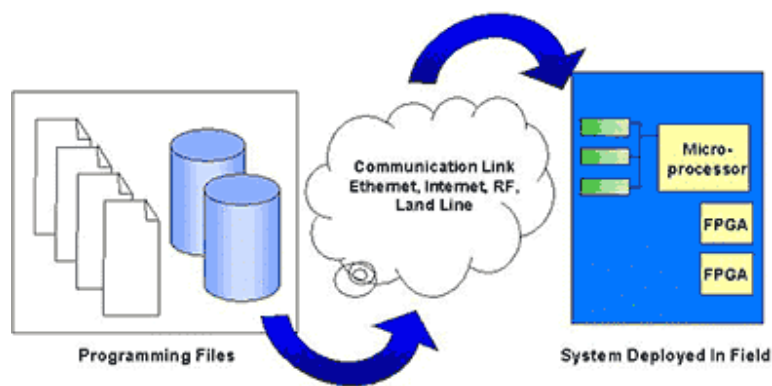
Lattice's economical ECP2/M FPGAs contain bitstream encryption functionality, which can also be found in the non-volatile LatticeXP2 FPGAs with built-in Flash. With the LatticeXP2 FPGAs, an SRAM map is incorporated into the same chip on which the application and Flash map containing the bitstream configuration work. These components are particularly useful when board space and fast start-up times are important.

### **FPGAs with Built-in Flash**

Non-volatile FPGAs offer the added security feature of Flash protection. It is possible for the user to prevent the Flash from being erased or reprogrammed in order to avoid an accidental or unauthorised operation. This functionality is protected by a 64-bit key.

When erasing or reprogramming the device, ispVM System checks if the Flash is protected; if it is, the user is asked to enter the 64-bit key. The ispVM then checks that this key corresponds with the key stored in the component. If it matches, the operation is performed, but if the key has been lost the component can no longer be erased.

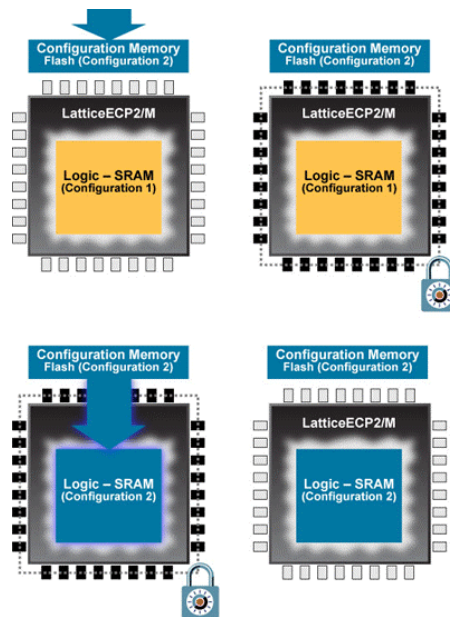
These safety measures not only protect the user's board applications but also protect against piracy that could modify the system's behaviour. These features also help with remote system updates. Lattice FPGAs offer this functionality so that users can update their systems by reprogramming the FPGA without disrupting the components around it. This functionality is called TransFR.



**Figure 5 – Remote Updates with Minimal System Disruption**

One of the most noteworthy features of FPGAs is that they are so easy to reconfigure. However, reconfiguration traditionally has meant significant system disruption. However, Lattice has developed TransFR to minimize the inconveniences caused by reconfiguration. Several Lattice FPGA families support TransFR technology, including those with built in Flash such as MachXO, LatticeXP and LatticeXP2, as well as SRAM FPGAs like the LatticeECP2/M.

TransFR is a technique which uses basic task programming and boundary scan cells simultaneously. Basic task programming modifies the content of the non-volatile memory (internal or external) so that the SRAM part of the FPGA continues to function. Lattice FPGAs have boundary scan cells that can be sampled and pre-loaded, allowing the FPGA I/Os to be controlled during the programming.



**Figure 6 – TransFR’s Four Steps**

TransFR is performed in 4 steps:

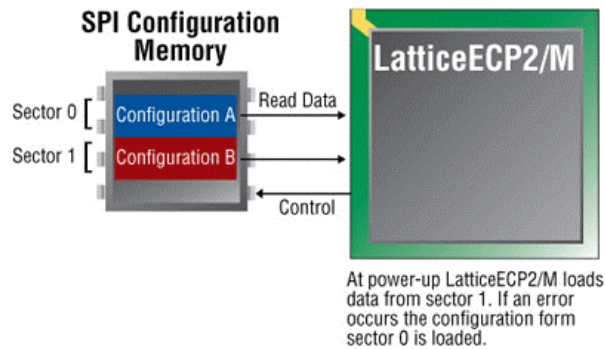
- 1- Basic task programming. The system reprograms the FPGA’s non-volatile storage memory without disrupting the SRAM functionality.
- 2- The I/O states are captured and the state is either maintained or pre-determined by the user. The I/O values remain the same throughout the entire re-configuration process in order to prevent a system shutdown.
- 3- The JTAG commands are then used to transfer the new configuration in the non-volatile memory. Once the SRAM programming is complete, a global reset is performed in order to place the component in a known location. The I/O states are released and the PLLs are relocked.
- 4- The outputs are released and the internal logic stops monitoring the I/Os.

TransFR works with the assistance of the ispLEVER development tool and the ispVM System programming tool.

The update has now been completed using a 128-bit AES key encrypted file and Lattice TransFR technology. However, a system failure could still be caused by a system weakness or a criminal act. Although the FPGA bitstream is encrypted, it could have been corrupted by either an illegal operation or by an error in transmission. Corruption will cause the FPGA to be blocked and the system cannot restart correctly. In order to overcome this malfunction, Lattice has implemented “dual boot.”

### **Dual Boot Configuration**

Two program files are stored in the configuration memory.



**Figure 7 – Dual Boot Configuration**

Having activated the PROGRAM pin or after a JTAG refresh instruction on start-up, the FPGA downloads the bitstream stored in Sector 1. If a CRC error occurs, the FPGA automatically reverts to Sector 0 configuration. This functionality is also available on Lattice XP2 non-volatile FPGAs. The Flash component contains the active configuration and an SPI Flash is used to store the safe (“golden”) configuration.

Lattice has a large range of products that enable designers to secure their designs. Lattice ECP2/M FPGAs offer a high level of security with 128-bit AES keys. These economical components offer a large memory capacity (up to 5.3 Mbits) high speed I/Os and SERDES. Non-volatile LatticeXP2 FPGAs are unique in the market and provide an extremely high level of security. Thanks to FlexiFlash technology, the bitstream is stored in the component Flash and can be protected by the 128-bit AES key. Furthermore, TransFR facilitates a totally secure remote system update.

Design security should be part of the design process, not an afterthought. Whichever techniques are chosen, it is essential the design be protected against tampering, whether it is inadvertent or illegal.