Third Generation Non-Volatile FPGAs Enable System on Chip Functionality

A Lattice Semiconductor White Paper

June 2007

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Requirements for Non-volatile FPGAs

FPGA designers continue to exhibit a preference for non-volatile, reprogrammable FPGA solutions -- provided the associated cost premium is not too great. This preference typically is driven by a desire to access one or more of the capabilities that, to date, are exclusively provided by non-volatile solutions:

- Smallest board area and simplest system integration
- Fastest availability of logic after power-up
- Highest security

In addition to these traditional requirements, Lattice Semiconductor, based on its experience with two previous generations of non-volatile FPGA products, has identified the need for flexible on-chip non-volatile memory and a comprehensive solution for field update of logic as emerging requirements for non-volatile FPGAs. This white paper reviews these requirements and possible solutions.

Approaches to Delivering Non-volatile FPGAs

Today there are three classes of FPGAs in the marketplace. First is the traditional SRAM FPGAs that require an external non-volatile memory to configure the FPGA at system power up. Second is a hybrid approach that combines an SRAM FPGA and a non-volatile memory in a single package. The third class, often referred to as true, or monolithic, non-volatile FPGAs, embeds non-volatile elements onto the same die as the FPGA logic. As summarized in Figure 1, and discussed in this white paper, while the hybrid approach offers some benefits over the SRAM approach, the true non-volatile approach does the best job of delivering a small footprint, high security and instant-on operation at power-up.

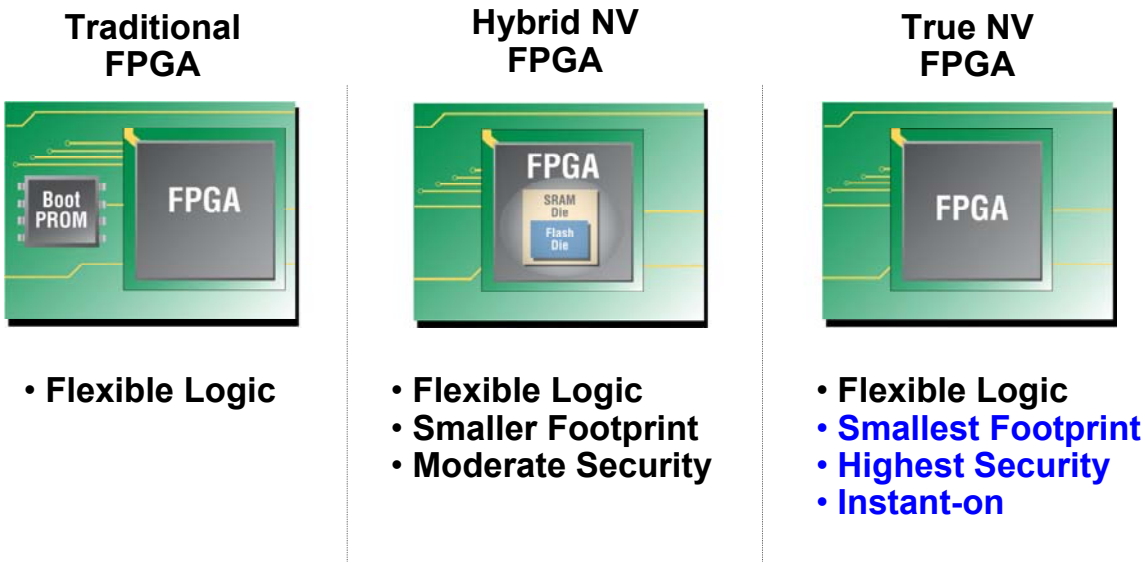


Figure 1 – Classes of FPGAs

Applications For Non-volatile, Reprogrammable FPGAs

Non-volatile, reprogrammable FPGAs are used for implementing system logic in a wide variety of end markets including communications, consumer, industrial, computing, military and automotive. Applications where non-volatile FPGAs are a particularly attractive option typically require reduced parts count and small footprint, rapid availability of logic or high-security.

Small Footprint Applications

Traditional SRAM FPGAs require a boot memory to load the SRAM configuration at power-up. Sometimes configuration loading is done via the on-board microprocessor, while in other applications a stand-alone boot memory is required. Neither solution is ideal. Booting from the system microprocessor introduces additional interdependencies between hardware and software development. It also requires the microprocessor to be up and running prior to the configuration of the FPGAs, precluding its use for system heartbeat functions. Using a stand-alone boot memory increases the board area footprint of the solution and the bill of materials (BOM), as well as associated costs. By

integrating the boot memory on-chip, non-volatile FPGAs provide an alternative and more elegant solution. Applications where this small footprint is beneficial include:

- Handheld barcode scanners
- Handheld barcode readers
- Smart phones
- Instrumentation and sensors
- Avionics

Applications Requiring Rapid Availability of Logic

On-chip, non-volatile memory typically allows devices to be ready for operation within approximately one millisecond of power-up, in contrast with SRAM and hybrid FPGA devices that typically require tens or hundreds of milliseconds for configuration. Figure 2 illustrates the difference in "wakeup times." The rapid logic availability of true non-volatile FPGAs is a desirable characteristic in many common applications:

- Plug & Play Bus Interfaces (PCI, PCI Express, CAN)
- Power-on-Reset Control
- Processor Bus Decode
- FPGA Loaders
- ASIC Initialization
- Low Power Designs Using Duty Cycling

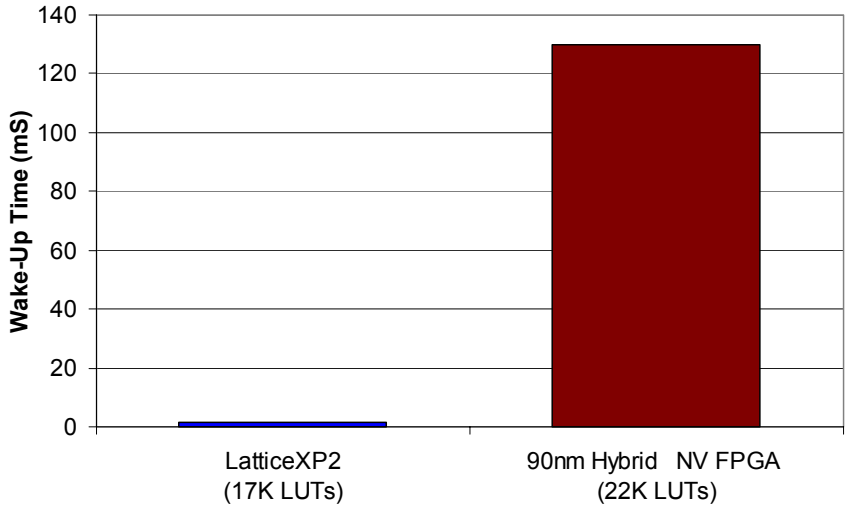


Figure 2 – Rapid Ability of Logic With Non-Volatile FPGAs

Applications Requiring Design Security

In today's complex systems, FPGAs are increasingly being used to replace functions traditionally performed by ASICs and even microprocessors. Ten years ago, the FPGA was at the periphery of the system--today it is at the heart. With current FPGA technology gate counts running into the millions, FPGAs are an attractive target for piracy. FPGA designers are increasingly concerned about issues that include:

- Cloning
- Reverse engineering
- Overbuilding
- Theft of service

The SRAM FPGAs most commonly used by system designers need to be configured from a boot device every time the system powers up. This link between the boot device and the FPGA represents a significant security risk because the configuration data is exposed and vulnerable to piracy during system power up. This situation is improved marginally with the hybrid approach, which embeds the non-volatile memory within the packages. However, relatively basic tools allow the removal of packaging materials and

access to the die interconnections. True, monolithic non-volatile devices have no such interconnections and so provide excellent design security.

In addition to protecting the design from theft or replication, in many systems it is important to ensure that the FPGA has not been tampered with. Applications with this requirement typically include:

- Credit card readers
- Automatic Teller Machines
- Weapons systems
- Gaming systems

Non-volatile Storage Applications

As non-volatile FPGAs continue to grow in density and capability, many designers want to integrate user accessible non-volatile memory elements within their designs. Typically these requirements fall into two categories.

The first category is the ability to integrate small stand alone EEPROM memories that are used within many systems to store data, such as electronic ID codes, version codes, date stamps, calibration settings and asset ID. Typically this class of memory is accessed relatively infrequently through some form of serial interface.

The second category is the ability to integrate large blocks of memory that are typically used to store data such as error codes, power on self test logs, data look-up tables and microprocessor code. While typically write operations to this data are relatively infrequent, the reading of this data can occur often and the speed of read operations impacts overall system performance.

In-Field Updates

It is increasingly important to provide the capability to update the FPGA configuration while equipment is deployed in the field, as illustrated in Figure 3. This capability provides equipment suppliers with a competitive advantage by allowing them to rapidly respond to changes in standards, add new services and, if necessary, fix bugs.

Figure 3 – In-Field Update of Logic

The LatticeXP2 FPGA Family

The LatticeXP2 family was developed using a 90nm embedded Flash process that was co-developed by Lattice Semiconductor and Fujitsu. The use of this technology enabled Lattice to reduce the cost of its LatticeXP2 devices by 50%, reducing the cost premium of these devices compared to the previous generation of LatticeXP devices. The devices combine the functional blocks used in the successful LatticeECP2 SRAM devices with Flash cells in an arrangement referred to as flexiFLASH.

LatticeXP2 Architecture – Functional Blocks

The core of the LatticeXP2 devices contains Programmable Function Units (PFUs) that allow the implementation of logic and, for 25% of the blocks, distributed memory. Logic is implemented using four input Look-Up Tables (LUT-4s) and register pairs, which is the *de facto* standard for the FPGA industry and well understood by system designers and logic synthesis tool suppliers. Distributed memory provides an efficient method for designers to implement small blocks of scratch pad memory. The family provides 5K to 40K LUTs.

Rows of sysMEM Embedded Block RAM (EBR) provide between 166K to 885Kbits of dual port memory in18kb blocks. Also provided as a row through the device are

sysDSP blocks that provide multiplication, addition, subtraction and accumulation capabilities suitable for implementing common DSP functions such as FIR filters, FFTs and complex arithmetic. The sysDSP blocks provide from 12 to 32 18x18 multipliers.

Around the periphery of the device are 86 to 540 sysI/O interfaces that allow the device to interconnect to a variety of I/O standards, including LVCMOS, PCI, LVTTL, LVDS, SSTL and HSTL. Additionally, LVPECL, BLVDS and RSDS interface standards can be emulated with the addition of external resistors. DLL calibrated DQS delay blocks, DDR registers and clock transfer circuitry allow the easy implementation of high performance DDR and DDR 2 memory interfaces up to 400Mbps. Generic DDR interfaces up to 750Mbps can also be implemented with the device.

The devices also provide up to four PLLs for clock synthesis and alignment. An on-chip oscillator provides a low-accuracy clock source suitable for many housekeeping functions such as watchdog timers and keyboard scan logic. Clocks can be distributed by eight global clocks, eight regional clocks or two high-speed edge clocks.

On the left and the right of the diagram can be seen the blocks of Flash memory used for device configuration.

The devices operate with a 1.2-volt core power supply and are provided in a number of space saving package options. Figure 4 illustrates the overall architecture and Figure 5 details the different family members. The remainder of this white paper examines the operation of the non-volatile features in more detail.

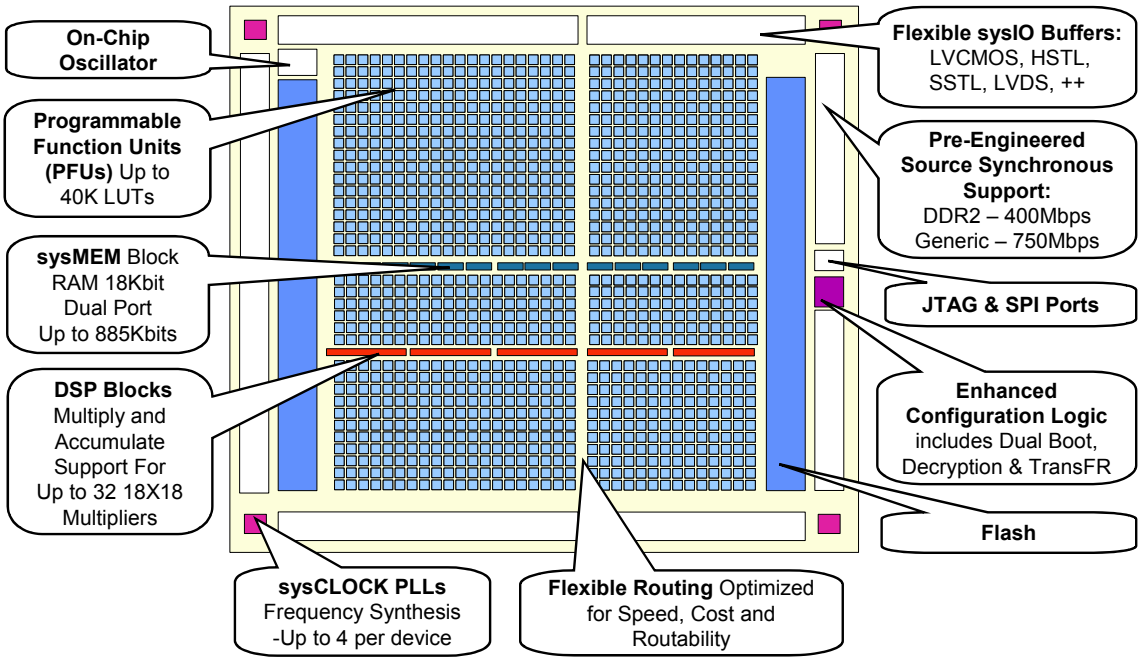


Figure 4 – LatticeXP2 Architecture

Device	XP2-5	XP2-8	XP2-17	XP2-30	XP2-40
LUTs (K)	5	8	17	29	40
EBR SRAM Blocks	9	12	15	21	48
EBR SRAM (Kbits)	166	221	276	387	885
Distributed RAM (Kbits)	10	18	35	56	83
# 18x18 Multipliers	12	16	20	28	32
PLLs	2	2	4	4	4
Package & IO Combinations					
132-ball csBGA (8x8mm)	86	86			
144-pin TQFP (20x20mm)	100	100			
208-pin PQFP (28x28mm)	146	146	146		
256-ball ftBGA (17x17mm)	172	201	201	201	
484-ball fpBGA (23x23mm)			358	363	363
672-ball fpBGA (27x27mm)				472	540

Figure 5 – LatticeXP2 FPGA Family

flexiFLASH Details

The LatticeXP2 devices combine Flash and SRAM cells in an architecture referred to as flexiFLASH. The device logic configuration and Embedded Block RAM data is stored in SRAM cells. At power-up or at user command these SRAM cells are loaded in a massively parallel fashion from blocks of on-chip Flash memory. The rapid transfer allows the instant-on characteristics of the devices while the on-die Flash memory results in a single chip solution. The on-chip Flash memory can be written via either a JTAG or SPI port as shown in Figure 5.

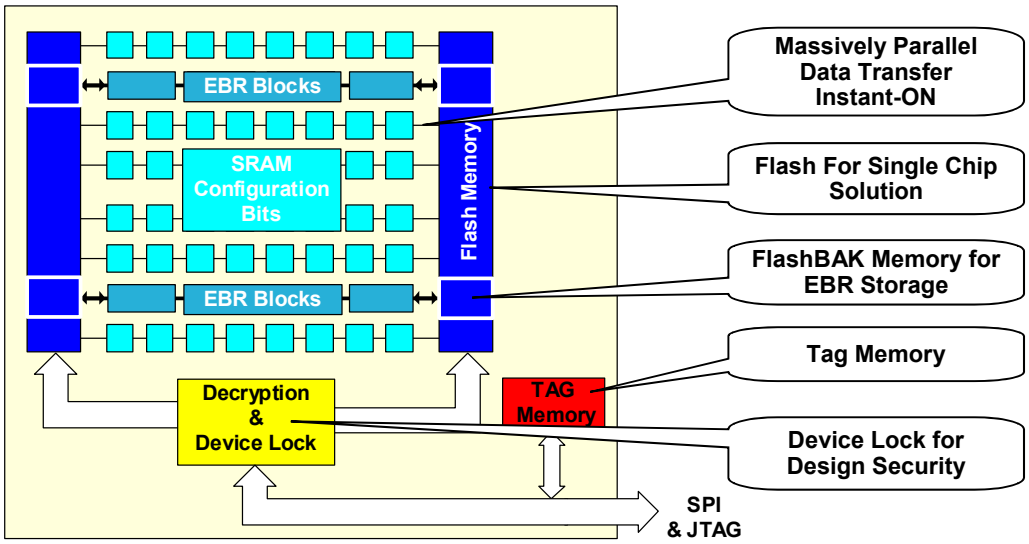


Figure 5 – flexiFLASH Architecture

FlashBAK Memory

To meet the need to store relatively large chunks of data, the LatticeXP2 devices provide an innovative capability referred to as FlashBAK memories. As explained previously, at power up the EBRs are loaded from on-chip Flash memory. The EBRs can then be read from or written to at speeds up to 350MHz. Then, as desired, by toggling a signal within the FPGA it is possible to rewrite the Flash bits corresponding to the EBR bits to be with the current EBR contents. This process takes approximately 1

second. In this way users get the high-performance and practically infinite read and write capability of SRAM combined with the non-volatile capability of Flash memory. The devices provide between 166K and 885Kbits of FlashBAK memory.

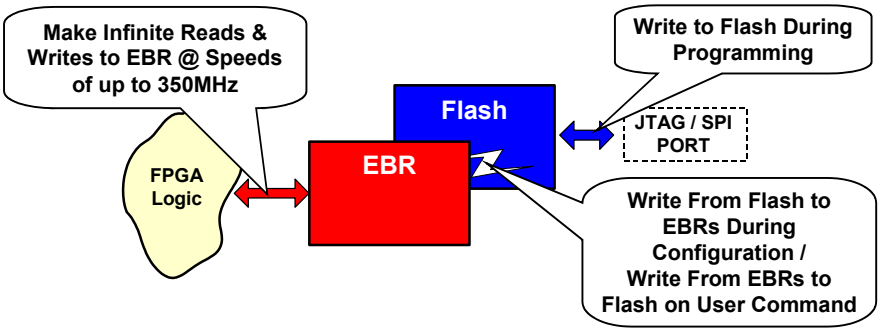


Figure 6 – Non-volatile FlashBAK Storage

Serial TAG Memories

To provide small amounts of serially accessed memory, each of the devices contains between 0.6K and 3.4Kbits of serial TAG memory. As shown in Figure 7, this memory can be accessed through either the device's JTAG interface or FPGA logic. This memory sits outside of the device's security mechanisms, allowing it to be accessed independently of the device's security settings.

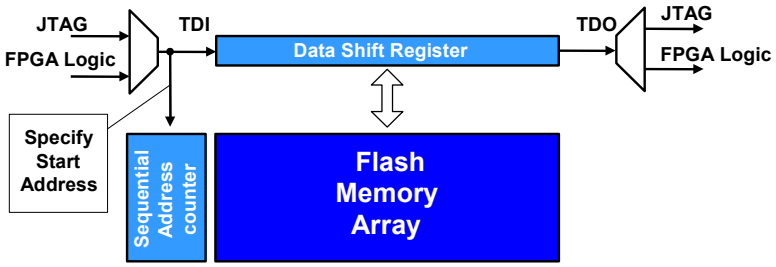


Figure 7 – Serial TAG Memory

Device Lock Provides Design Security

As noted previously, many designers want to ensure that their design remains securely locked within the FPGA. Physical examination of the FPGA and SRAM cells, which are buried under multiple levels of metal, is close to impossible. Security bits that prevent the device configuration from being interrogated via the JTAG or SPI ports complement this intrinsic security.

To prevent against unauthorized tampering, the devices have a 64-bit code that, once set, is required for erasing and rewriting the Flash. For customers wanting an even more secure solution, a One Time Programmable (OTP) mode is available. Once the devices are placed in this mode it is no longer possible to erase or reprogram the devices.

Comprehensive In-Field Update

As reviewed earlier, designers implementing field updates have requirements for maximum equipment uptime, excellent reliability and high design security. To meet these requirements the LatticeXP2 devices provide TransFR I/O, Dual-boot and 128-bit AES decryption.

TransFR I/O For Maximum Uptime

Most FPGA tri-state their I/Os during system configuration. This lack of control normally necessitates the power to be cycled in order for the FPGA configuration to be updated. Like other Lattice FPGAs, the LatticeXP2 devices feature TransFR I/O that allows I/O states to be frozen during device configuration. This allows the devices to be field updated with a minimum of system disruption and downtime, allowing designers to meet the dual requirements of high system uptime, such as ‘5 nines’ (99.999%) availability, and field updating of logic. Figure 8 shows the four steps of a TransFR I/O update. Because the new configuration can be loaded into the LatticeXP2 on-chip Flash in the background, the loading of a new configuration to SRAM can occur very rapidly. It is

possible to lock the I/Os, apply the new configuration and release the I/Os in less than 2mS.

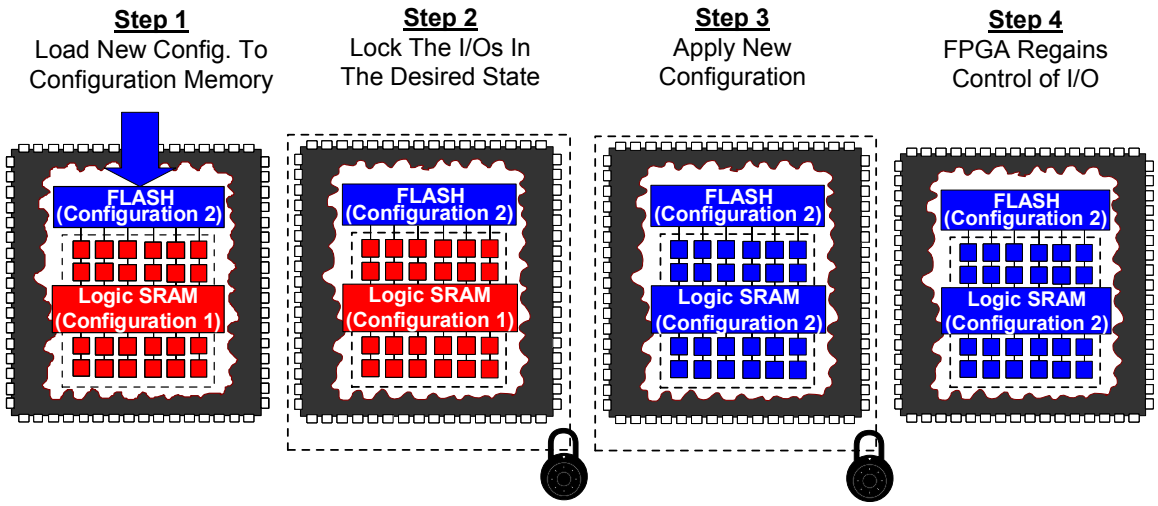


Figure 8 – Four Steps of a TransFR I/O Update

Dual-boot for Reliability

While the stored FPGA configuration is being updated there is typically a risk that a power or communication failure could result in a corrupted configuration and a non-operational system. If this occurs then an expensive visit is typically required from a field technician to correct the problem. To guard against this possibility, the LatticeXP2 devices can use an external SPI memory to implement the dual boot scheme.

As illustrated in Figure 9, at power up the LatticeXP2 device attempts to load its SRAM configuration bits from on-chip Flash. If an error is detected during this process the FPGA then reads a backup, or golden, configuration from the external SPI memory. By operating the system in this manner, field updates can be made reliably regardless of power and communication outages.

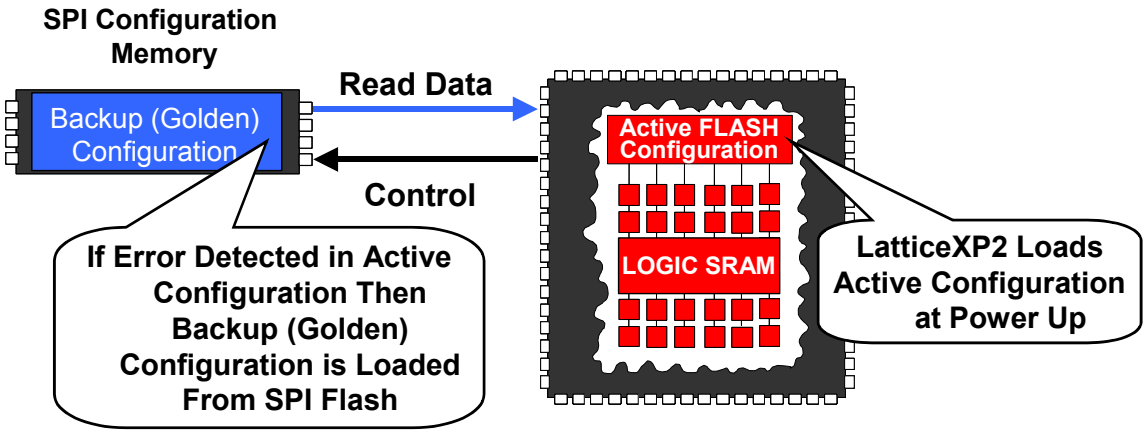
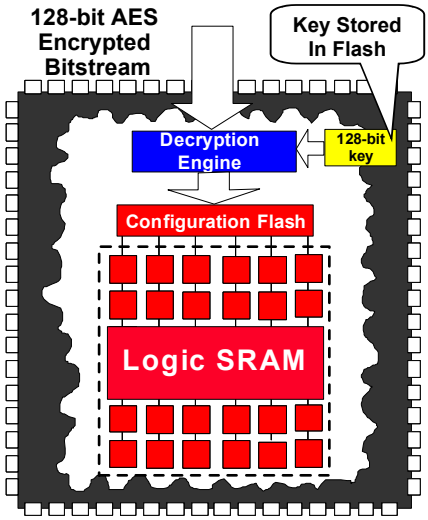


Figure 9 – Dual-boot For High Reliability Field Updates

128-bit AES Encryption For Design Security

The LatticeXP2 devices allow for the optional 128-bit AES encryption of programming data. As the devices receive encrypted data, it is decrypted using an on-chip decryption engine that refers to the user-defined key that is stored on-chip using Flash memory. The basic operation is shown in Figure 10. This mode of operation allows sensitive design data to be protected during in-field updates.



Summary

The primary reasons for designers to select non-volatile devices remain unchanged: small footprint, instant-on and high security. However, new requirements for non-volatile data storage and comprehensive field update solutions are emerging. All of these requirements must be satisfied while the price premium for a non-volatile solution is minimized relative to a traditional SRAM solution.

Today there are two approaches to delivering non-volatile FPGAs: hybrid and monolithic. The hybrid approach addresses primarily the requirement for a small footprint. The monolithic, also referred to as true, non-volatile approach satisfies the requirements for small footprint, instant –on and high security.

Through the use of 90nm embedded Flash technology, the LatticeXP2 FPGAs meet the primary and emerging reasons for selecting a non-volatile device while also reducing costs.

###