Field Update FPGAs While System Operates

A Lattice Semiconductor White Paper

May 2005

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

The Need for Seamless Field Logic Updates

An increasing number of engineers are designing equipment that allows the logic within Field Programmable Gate Arrays (FPGAs) to be updated while it is deployed in the field. The ability to update logic in the field provides unprecedented flexibility to fix bugs, respond to changing standards and add new features and services. See Figure 1.

Systems typically have been placed in a non-operational "maintenance mode" for the update of logic. However, there is increasing pressure on system manufacturers to increase system up-time. In many applications, demands for "5 nines" (99.999%) up-time are rapidly increasing. New solutions from programmable logic vendors that allow logic to be updated while the system operates are urgently needed.

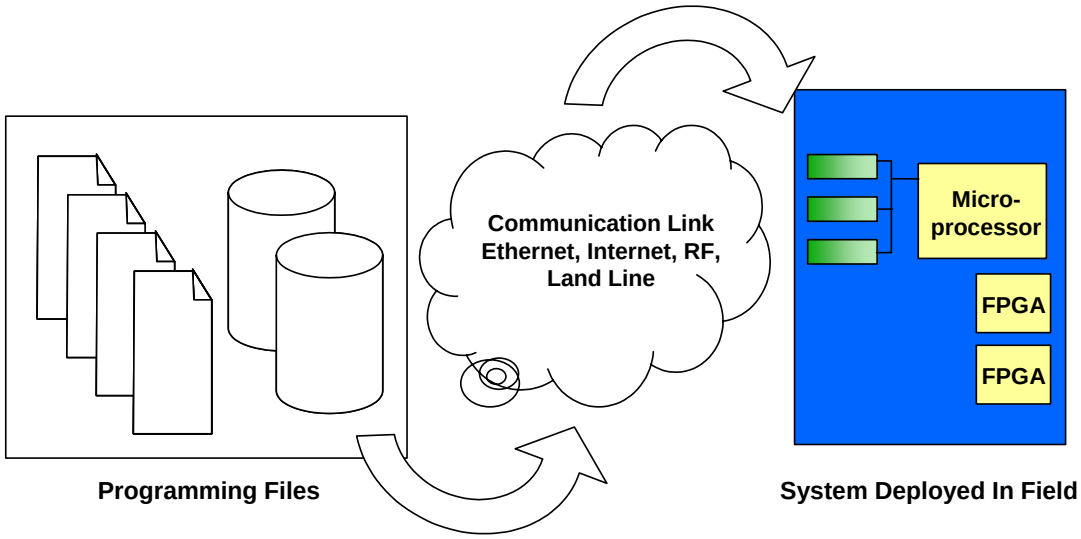

Figure 1 – Field Update of Logic

Programmable Logic Solution Requirements

There are four key capabilities required in order for programmable logic devices to be able to support the field update of logic while the system operates, as detailed below. Lattice delivers these capabilities with its Transparent Field Reconfiguration (TransFR™ or TFR) solution that allows the LatticeXP™ family of devices to be field updated while the system operates.

Embedded Programming Support

In order to do field programming it is necessary that the system microprocessor be able to program the FPGA. FPGA vendors typically provide C programs that can be compiled for and executed on the system microprocessor. These programs allow the microprocessor to take a file of programming data and provide it to the FPGA.

Fast Reconfiguration Capability

As the FPGA cannot process data (its inputs) while it is being reconfigured, it is essential that the entire update process be completed quickly to minimize the time the system has to wait for reconfiguration to take place.

Output States Preserved

Often, outputs from the FPGA drive key control signals within the system. For example, chip resets or power supply enables. During configuration it is critical that these signals continue to be driven in the correct state. A glitch on one of these signals could cause a system reset, which can result in a system re-boot process that takes several minutes.

Device States Controlled

It is essential to control the state of the logic within the FPGA prior to exiting the configuration process. This allows the device logic to drive the correct levels on its outputs immediately, when control of I/O is passed back to the FPGA logic at the end of the configuration cycle.

LatticeXP Architecture

At the core of the LatticeXP devices are 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 is 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. Rows of sysMEM Embedded Block RAM (EBR) provide 9kb blocks of memory for use in implementing larger memory blocks. At the end of the rows of sysMEM memories are sysCLOCK PLLs that allow clocks to be aligned for improved set-up and clock-to-out times, and new clocks to be synthesized.

Around the periphery of the device are sysIO 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 provide a pre-engineered high performance DDR memory interfaces up to 333Mbps. Generic DDR interfaces up to 700Mbps can also be implemented with the device. The various functional blocks of the architecture are interconnected with a routing fabric that provides an optimum balance between speed, flexibility and cost. Figure 2 illustrates the overall architecture.

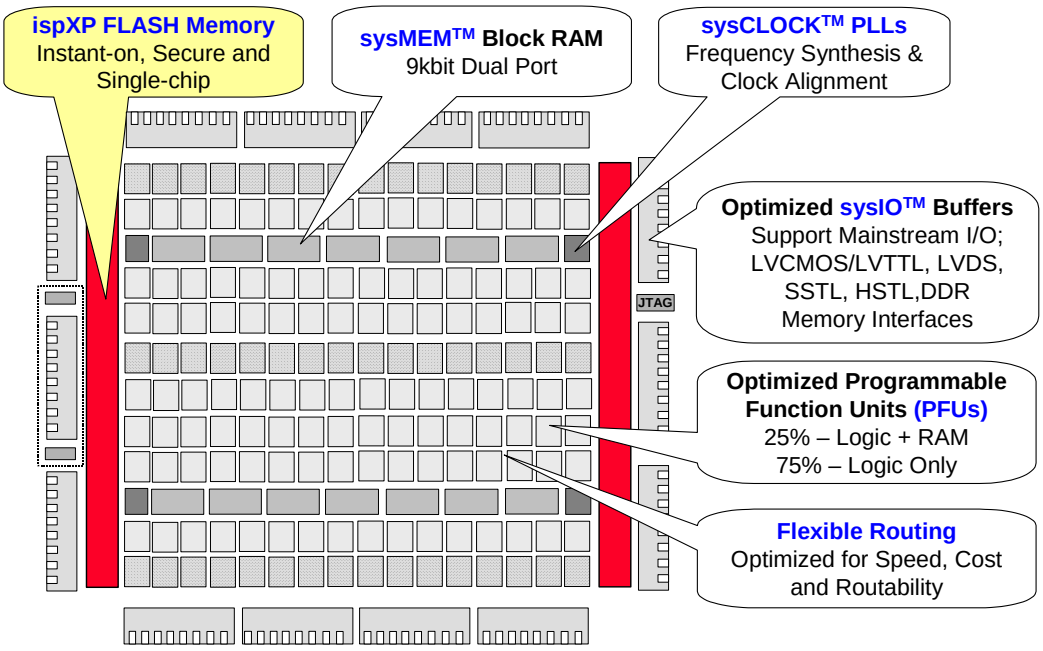

Figure 2 – LatticeXP Architecture

Non-Volatile and SRAM Memory

SRAM configuration bits control the operation of the LatticeXP devices. At power-up, these bits are loaded via the on-chip, non-volatile memory, resulting in logic availability in less than 1ms after power good. During device operation the SRAM may be reconfigured from the Flash by toggling a pin or issuing the correct commands through the device configuration ports. Both the Flash memory and the SRAM memory can be reprogrammed/reconfigured via either a JTAG port or a sysCONFIG port (8-bit microprocessor interface). Configuration of the SRAM via these ports takes between tens and hundreds of milliseconds, depending on the chosen interface and device size. The Flash memory can be programmed in as little as 2 seconds. Figure 3 shows the operation of the different memories within the LatticeXP devices.

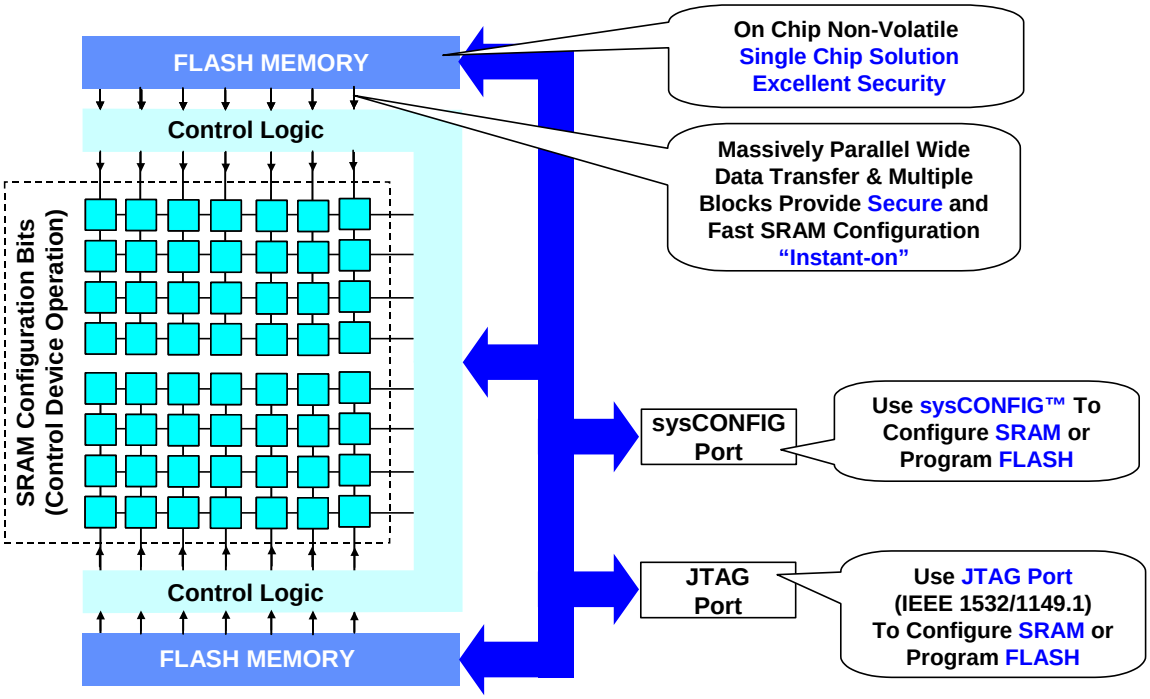

Figure 3 – Configuration Memories in the LatticeXP

Field Logic Updates Using TransFR Solution

The LatticeXP devices can be updated while a system continues to operate using the TransFR solution. This capability consists of four steps which are easily orchestrated

with Lattice's ispVM® toolset, which can be downloaded free of charge from the Lattice web site at www.latticesemi.com. The ispVM programming tool runs on a Windows machine and can communicate with the LatticeXP devices via a programming cable. Alternatively, the ispVM tool will generate the source code required for ispVM Embedded. This source code can be complied and then executed on the embedded processor within the system. The ispVM software will generate a series of programming files that are interpreted by the ispVM Embedded software or a user's own code in order to complete the desired programming operation.

Step 1—Background Programming

The first step of the TransFR process is to background program the Flash portion of the FPGA configuration. During the programming of the Flash, the logic continues to operate based on the configuration stored in the SRAM. For example, Figure 4 shows the device operating based on configuration 1, which is in the SRAM, while configuration 2 is loaded into the Flash memory.

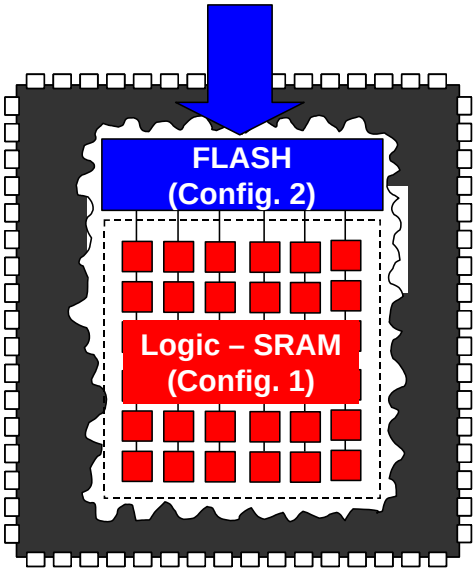

Figure 4 -- Background Programming

Step 2—I/O States Locked

The second step of the process is to lock the I/Os. Through the boundary SCAN registers, all the I/Os can be individually locked high/low, high-impedance or sampled and then driven to the current value. This I/O lock state can be specified individually on a pin-by-pin basis within the ispVM I/O Vector Editor graphical interface (Figure 5).

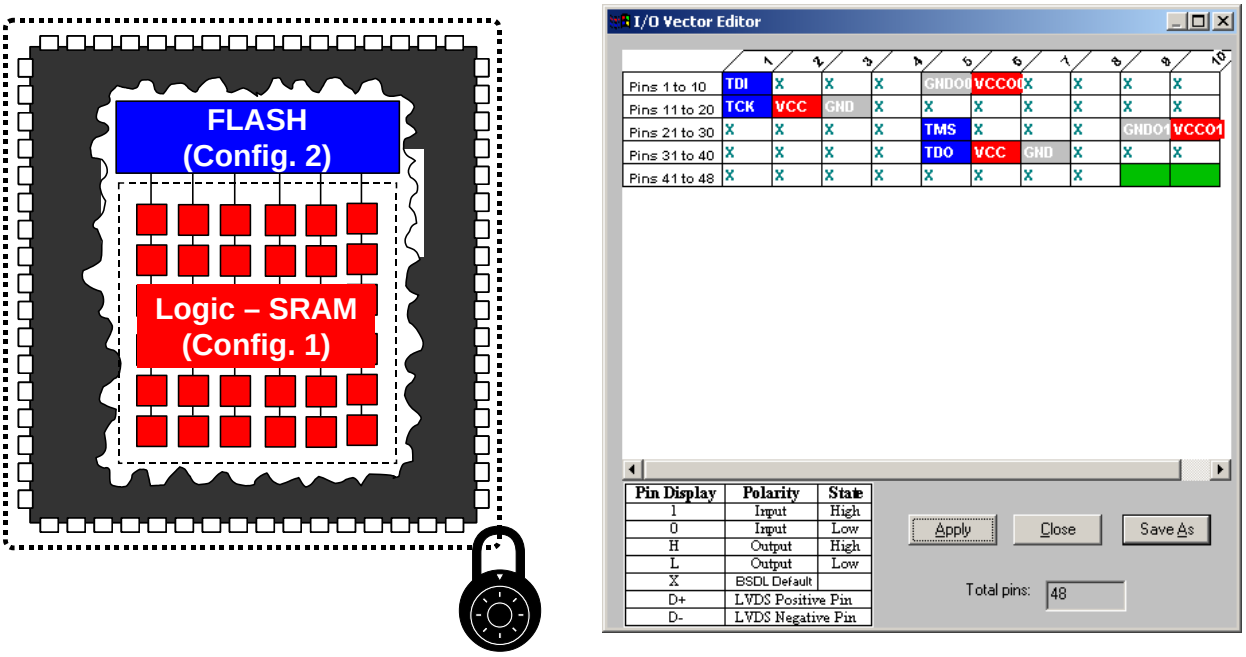

Figure 5 -- Locking I/O States Using ispVM

Step 3—Update SRAM Configuration

The next step of the configuration process is to update the SRAM memory from the Flash memory, as show in Figure 6. In the LatticeXP devices, this transfer is done in a massively parallel fashion and takes less than 1mS.

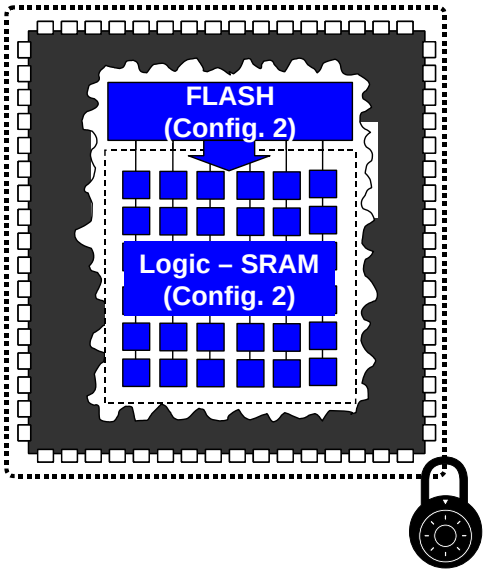

Figure 6 -- Updating the SRAM Configuration From FLASH

Step 4—Transfer I/O Control To User Logic

The last step in the TransFR process is to return control of the I/O to the user logic. After configuration, the FPGA logic has been reset and is responsive to inputs. PLLs will relock if necessary, taking less than 150uS. If desired, the FPGA can be placed into a known state by manipulating pin(s) via an external device or through JTAG. Finally, control of the I/Os is returned to the user logic and the FPGA can resume operation in the system.

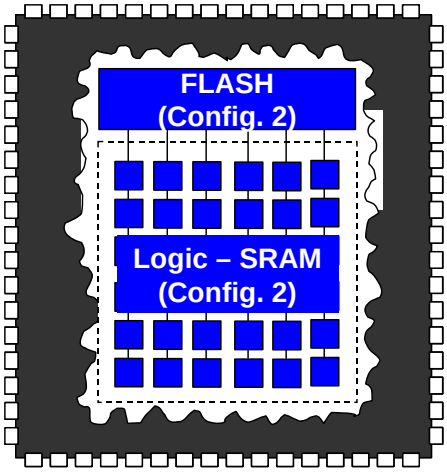

Figure 7 – FPGA Resumes Normal Operation

Summary

There is increasing interest in the ability to field update logic within FPGAs without interrupting system operation. This is driven by a desire to utilize the flexibility associated with in field updates combined with a need to maintain high system availability.

Lattice's TransFR solution uniquely allows designers to transparently update logic without disturbing system operation. The solution combines the inherent flexibility of the LatticeXP devices' dual SRAM + FLASH configuration spaces with easy to use ispVM software, which is available at no cost. This provides designers with a method to field update logic while meeting demands for increased system up-time.

For more information on TransFR please contact your Lattice sales representative or visit the Lattice web site at www.latticesemi.com.

###