Parallel Flash Programming and FPGA Configuration

September 2013 Application Note AN8077

Introduction

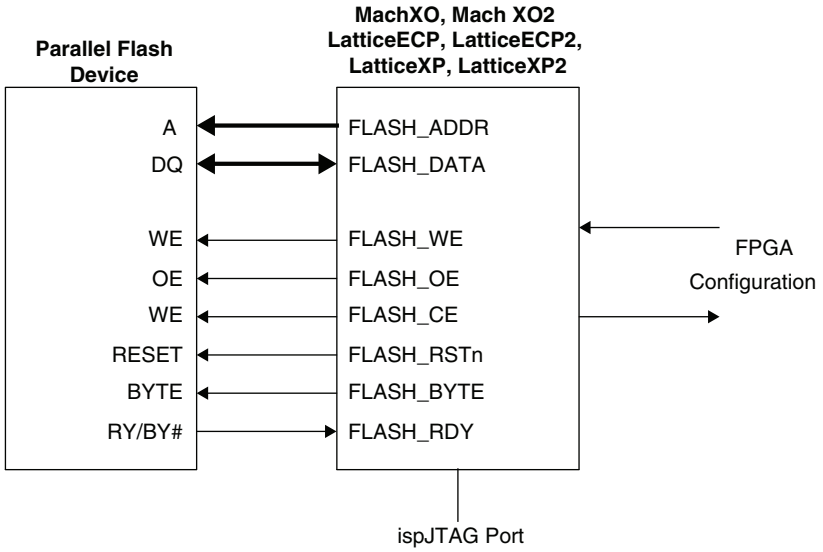
SRAM-based FPGA devices are volatile and require configuration at power up, with the configuration data held in an external device. Systems often task an embedded microprocessor with FPGA configuration, transferring the data from an on-board ROM or Flash memory. However, on systems that require fast configurations, or systems that do not have microprocessor resources readily available, a dedicated PROM is commonly used. Such PROM devices are typically expensive and are usually sourced from a single vendor.

An alternative solution is to use a Lattice non-volatile FPGA as an FPGA Loader. The FPGA Loader, coupled with a standard parallel Flash memory can perform the function of a PROM or microprocessor. This FPGA provides the JTAG programming interface to the Flash, as well as control of data to the other FPGAs for configuration. Figure 1 shows a diagram of the Flash and FPGA Loader device.

Note: Unless described otherwise, the remainder of this document will use the following terminology:

- *"FPGA Loader" – The non-volatile FPGA device that provides the Flash interface.*
- *"FPGA" – An FPGA to be configured by the FPGA Loader.*

Figure 1. Flash Programmer Interface



1. Pullup resistors may be required on some Flash pins. Refer to the Flash data sheet.
2. Flash signal names may vary slightly from those used in the diagram.

Supported Devices

The following devices are supported for implementation of the Flash Programmer:

- MachXO™
- MachXO2™

Supported devices for configuration through this design include any Lattice FPGA with serial or parallel support for conventional SRAM configuration. This includes:

- LatticeECP™
- LatticeECP2™
- LatticeECP2/M™
- LatticeECP3™
- LatticeXP™
- LatticeXP2™
- LatticeSC™
- LatticeSC/M™
- LatticeEC™

Note: Configuration from encrypted bitstreams for LatticeECP2/M 'S' series devices is currently not supported using the methods in this application note.

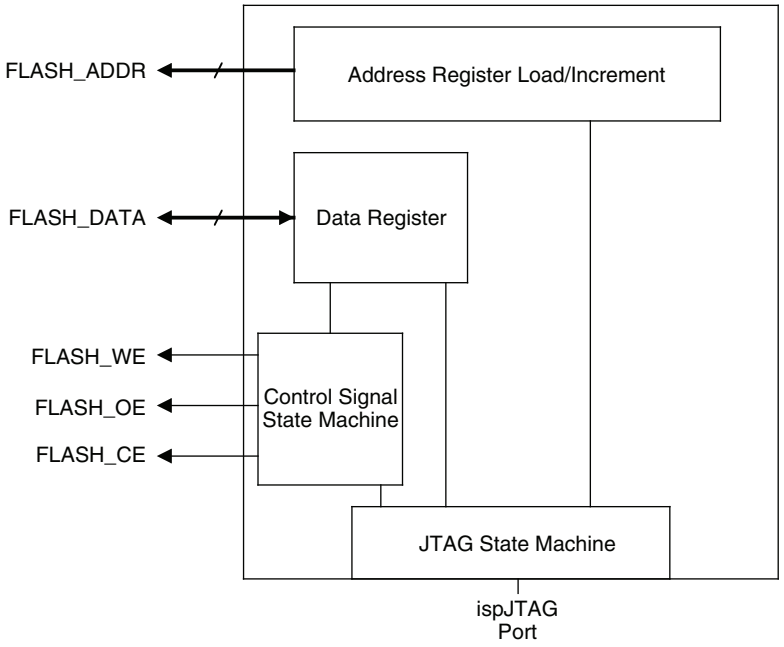
A list of supported parallel Flash devices is available in the Diamond Programmer. Diamond Programmer can be downloaded free of charge at www.latticesemi.com.

Programming the Flash

Unlike some dedicated configuration devices, which can be reprogrammed in-circuit via IEEE 1149.1 compliant JTAG programming tools, standard Flash memories typically do not support JTAG programming. This design, in order to provide greater capability, provides a method for updating the Flash memory while the device is still in-circuit.

Flash programming using ispVM System software allows a unified programming environment. ispVM System converts the configuration data file into a set of instructions that are executed through the ispJTAG™ port. A small amount of additional logic in the FPGA Loader device increments the address and provides the necessary control signals, shown in Figure 2.

Figure 2. FPGA Loader Flash Programming Block Diagram



FPGA Configuration

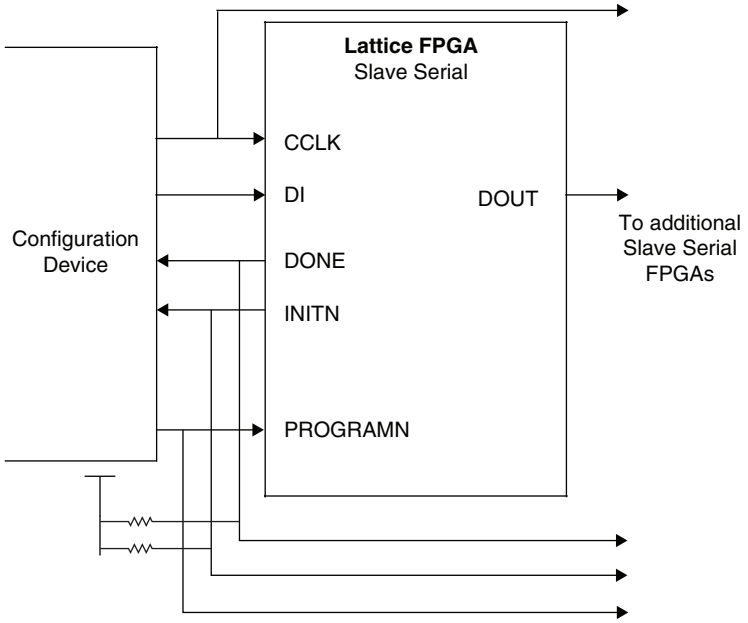
The FPGA device is configured automatically at power-up. While the power is ramping up to the device, a power-on-reset is triggered, external configuration mode pins are sampled, the DONE and INIT pins are driven low, and the internal SRAM cleared. Once the device reaches the proper voltage threshold, the INIT pin is released and must be pulled up externally to allow the start of configuration. This process can also be initiated at any time by toggling the PROGRAMn pin.

The FPGA supports several options for configuration, which can be broken down into two categories: parallel and serial. Parallel mode provides a byte-wide interface, while serial mode is based on a bit-wide data stream. For more details on FPGA configuration, refer to the corresponding Lattice technical note for the FPGA device of interest.

Serial Mode

In serial mode, the FPGA obtains its configuration data one bit at a time on DIN at every rising edge of CCLK. If additional FPGA devices are present, serial data will flow out of the DOUT pin to form a configuration daisy chain. Figure 3 shows a diagram for this mode.

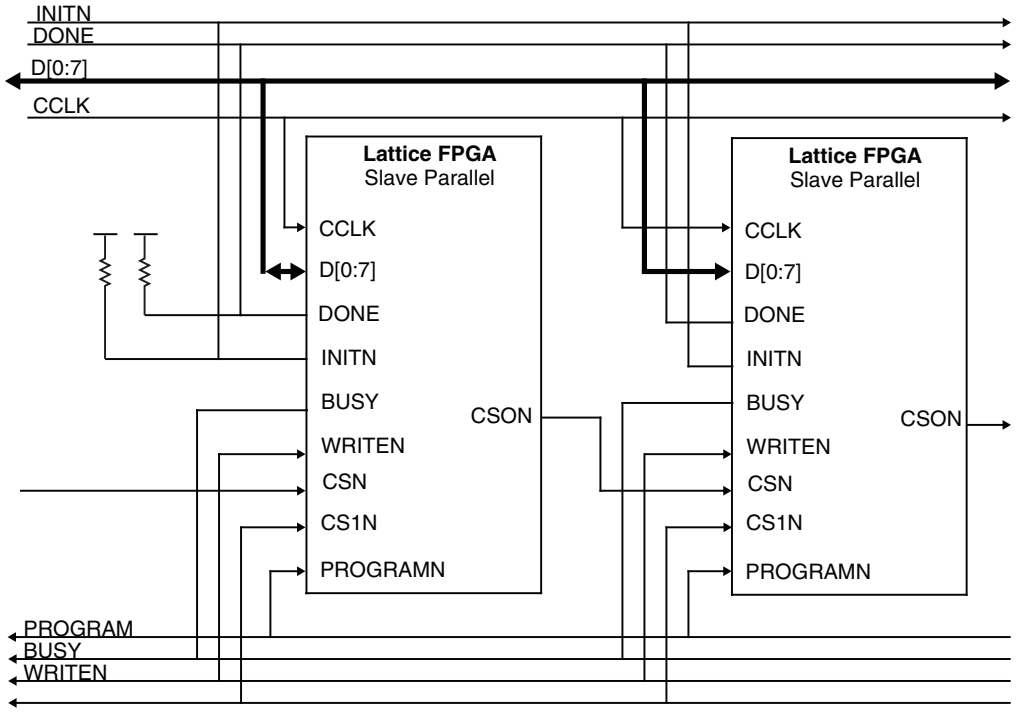
Figure 3. Serial Mode FPGA Signals



Parallel Mode

Parallel configuration mode operates with a byte-wide data bus and some additional control signals. To write to the parallel sysCONFIG™ port, the WRITEn, CSn, and CS1n pins must all be asserted low. To facilitate multiple FPGA configuration in parallel mode, the CSOn pin is used to assert the chip select input of the next device. The parallel configuration connections are shown in Figure 4.

Figure 4. Parallel Mode FPGA Signals



Implementation

The FPGA Loader design can be implemented using ispLEVER® or Lattice Diamond™ design software. The design uses approximately 220 registers and 230 LUT4s. The design source code can be compiled using the ispLEVER or Diamond design software to target the desired device with custom pin assignments.

Table 1. Device Resource Utilization

Design	Top-level File (*.vhd, *.v)	FFs[1]	LUTs[1]
Programmer Only	flash_prog	222	121
Programmer + Serial Configuration	flash_prog_load16_ss	204	110
Programmer + Parallel Configuration	flash_prog_load16_sp	230	210

1. Approximate utilization with default configuration.

Table 2. Netlist Files

Target Device[1]	Programmer IP Netlist File
MachXO	flash_programmer_xo.ngo
MachXO2	flash_programmer_xo2.ngo
LatticeECP2	flash_programmer_ecp2.ngo
LatticeXP	flash_programmer.ngo
LatticeXP2	flash_programmer_xp2.ngo

1. FPGA loader device

Design Fit Process Using Diamond Design Software

Requirements

- Lattice Diamond 2.1 or greater with a valid license
- ispLEVER with a valid license

Procedure

Note: This procedure assumes familiarity with the Lattice design software. For more information on using ispLEVER and Diamond, please see the tutorials included within the help system.

1. Extract the files from the distribution zip file for the intended implementation.
2. Launch the Diamond design tool and create a new project in the location of the files extracted in step 1. Select the project type according to the desired use of VHDL or Verilog.
3. Select a supported device as the target.
4. Import the top-level file, according to desired function and HDL preference. Uncomment the Macro parameter (XO, XO2, XP2, etc...) as per the selected device as an FPGA loader.
5. Launch the Design Planner to assign I/O types and pins for each of the signals. Timing preferences are also recommended. For example timing preferences, refer to the accompanying ‘preferences.txt’.
6. Execute the ‘Place and Route’ process to run the design flow.
7. To generate a programming file, execute the ‘Generate Data File’ process.

Implementation Notes

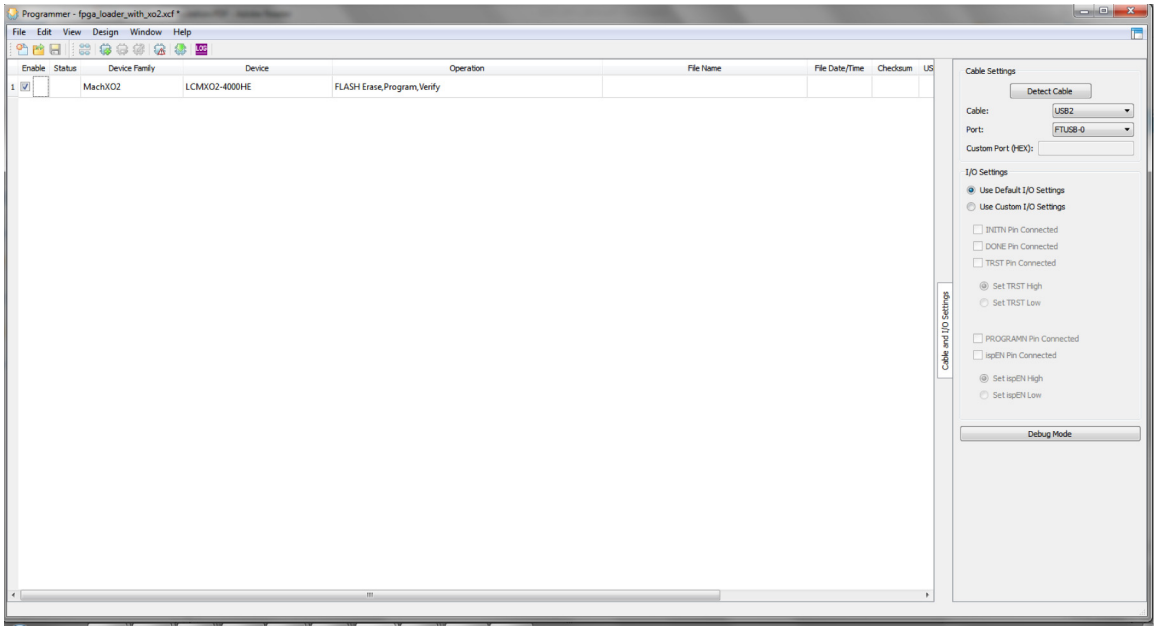
- When using 8/16 bit selectable Flash devices, 16-bit mode must be used during programming. Such Flash devices typically have a dedicated input pin (i.e. BYTE#) that should be asserted to the correct polarity during programming. After programming, either mode may be used for read access to the Flash device.
- When 8-bit only devices are supported, the lower 8 bits (FLASH_DQ[7:0]) should be connected to the Flash data bus.
- Only Flash devices that have uniform sector sizes are supported with sector erase functionality. Non-uniform (boot) sector devices are supported with an entire chip erase function. Flash devices with uniform sectors are strongly recommended.
- The fastest programming speeds are achieved using the USB version of the Lattice ispDOWNLOAD® cable with a PC supporting USB2.0.

Using Diamond Programmer to Program the Flash Device

Diamond 2.1 and later provides support for the Parallel Flash Programmer.

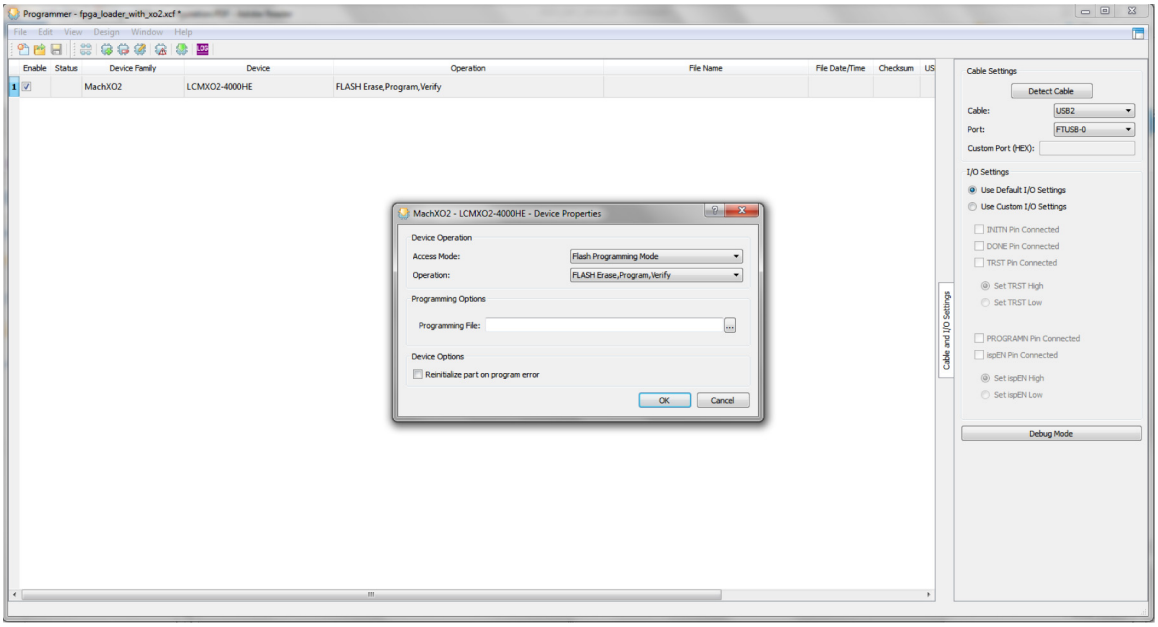
1. 1.Scan the chain or manually insert the devices representing the JTAG chain. Any non-Lattice devices should be appropriately handled by specifying instruction register lengths or importing a BSDL, SVF, or ISC file. An example chain is shown in Figure 5.

Figure 5. Diamond Programmer



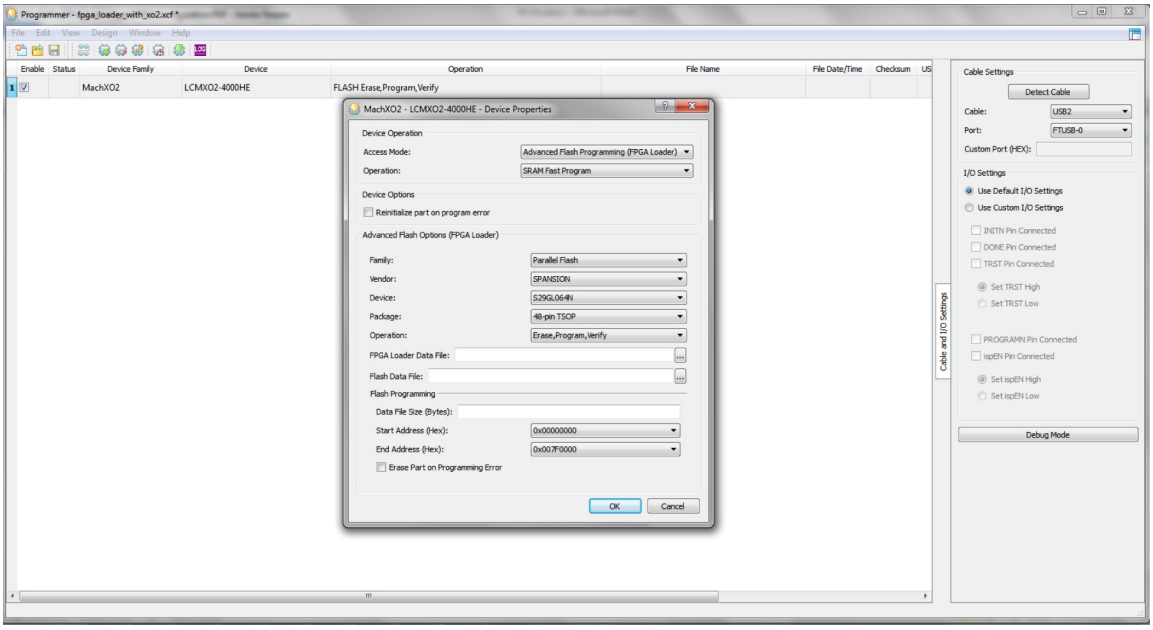
2. Double-click the device that will function as the FPGA Loader to edit the properties. The resulting window should appear as shown in Figure 6.

Figure 6. Device Properties Dialog



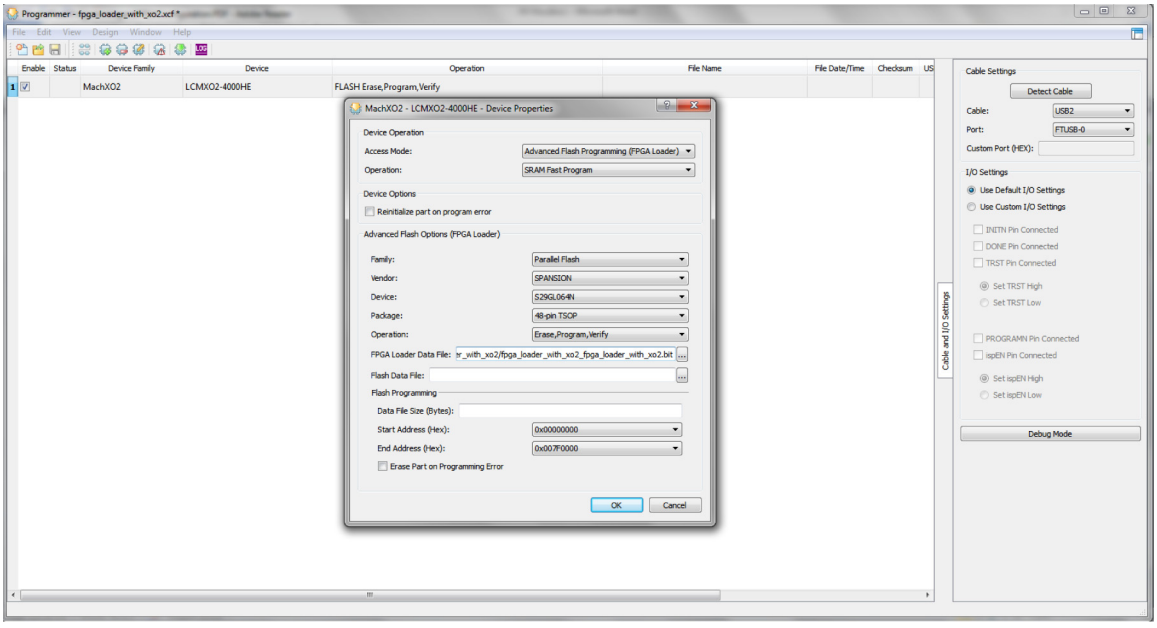
3. From the 'Access Mode' selection, choose “Advanced Flash Programming (FPGA LOADER)”. This opens the ‘Flash Programmer’ dialog box, as shown in Figure 7.

Figure 7. Flash Programmer Dialog



4. Within the ‘CPLD or FPGA Device’ page, shown in Figure 8, set the following options:

Figure 8. CPLD or FPGA Device Page

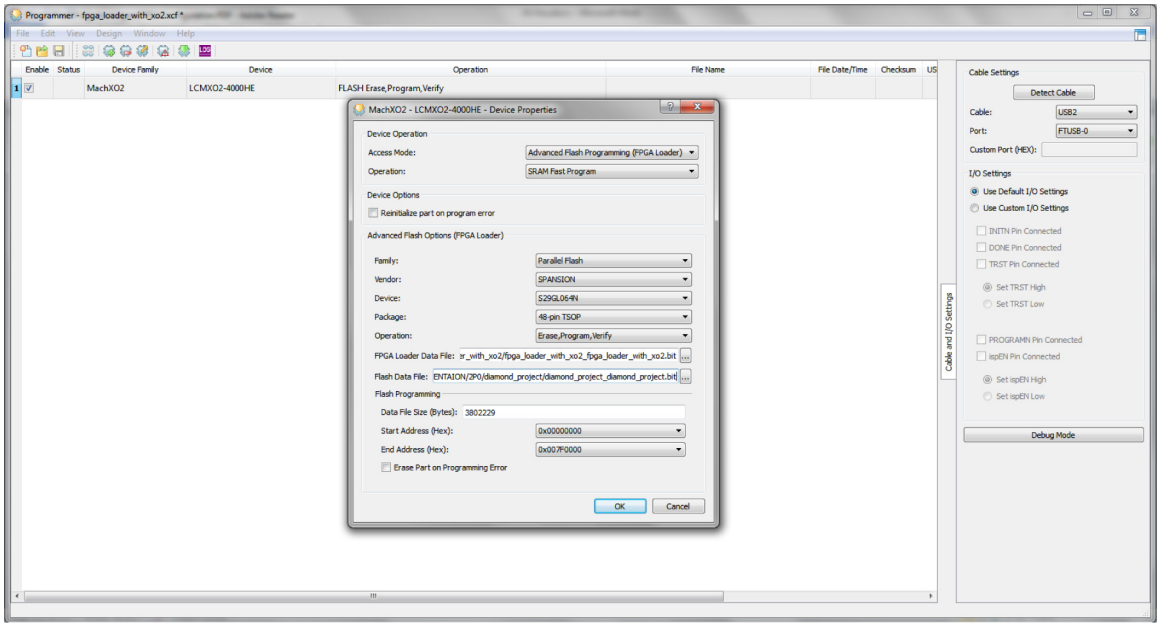


FPGA Loader Application Specific Data File – Browse to the JEDEC or bit stream file created by Diamond.

Operation – Select the appropriate programming option for the FPGA Loader device. If the device has already been configured with the correct file, the ‘Bypass’ operation can be used to skip this step.

5. Browse to the data file to program into the Flash device. If targeting a single Lattice FPGA, the bit stream (.bit) file may be used. Otherwise, choose the file as one of the supported PROM formats.

Figure 9. Configuration Data Setup Page



Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
February 2007	01.0	Initial release.
August 2007	01.1	Updated to support non-S version of LatticeECP2/M device family only under device support.
September 2013	01.2	Updated for Lattice Diamond design software support.
		Added support for MachXO2.
		Updated corporate logo.
		Updated Technical Support Assistance information.

Appendix A. 16-bit Flash to Slave Serial Configuration

VHDL toplevel file	flash_prog_load16_ss.vhd
Verilog toplevel file	flash_prog_load16_ss.v
Flash interface	Standard, 16-bit asynchronous
FPGA interface	Slave Serial

Figure 10. FPGA Loader Logic Diagram for Flash to Serial Configuration

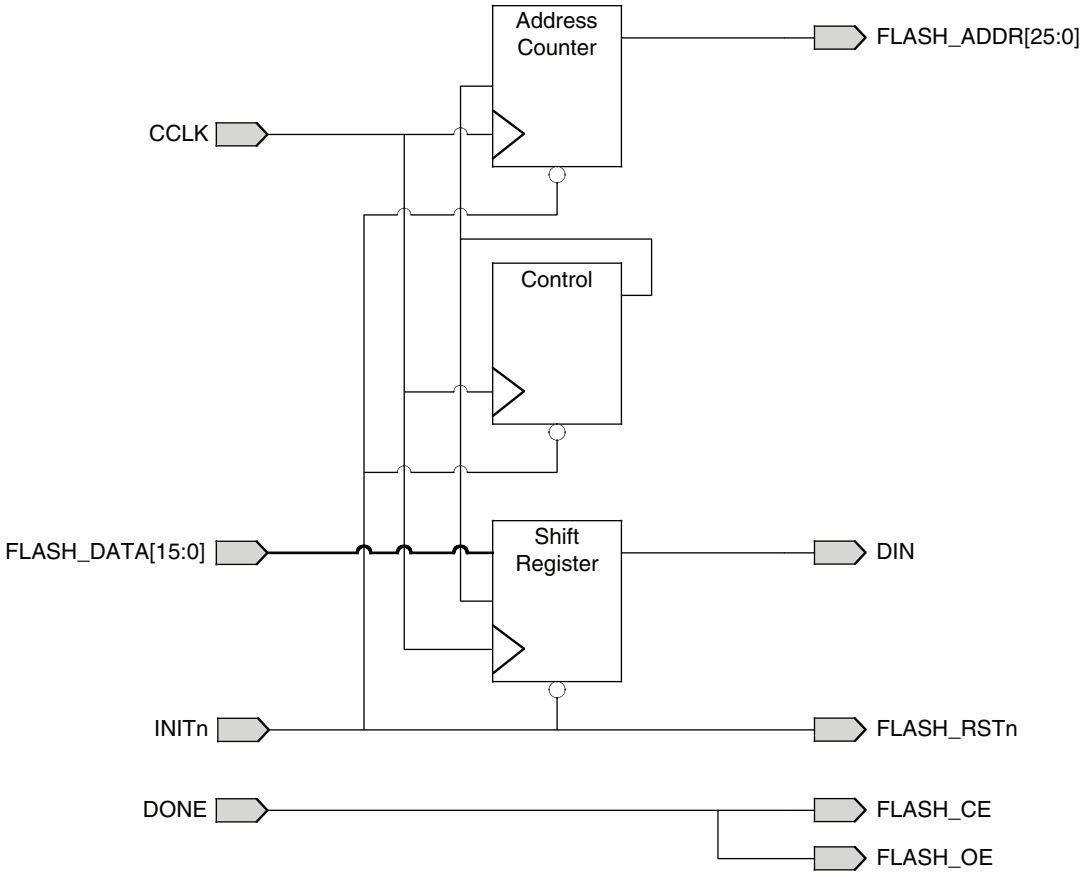
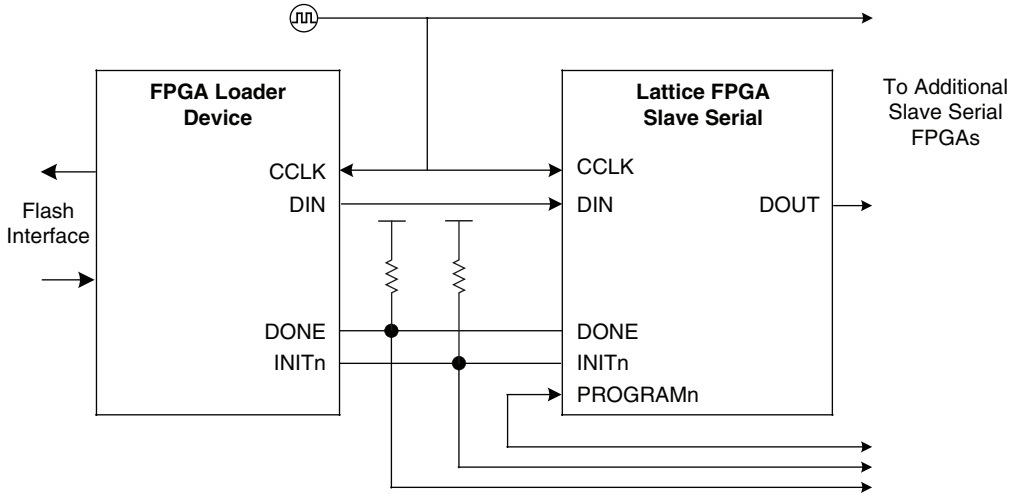


Figure 11. FPGA Loader to Slave Serial FPGA Configuration Signal Connections



Note: Master mode may also be used with applicable devices, where the clock source is from the FPGA. Only one FPGA should be designated as master, while any others connected in a daisy chain should be slaves.

Configuration Speed

The maximum allowed configuration frequency is dependent upon the Flash device used and the setup and clock-to-output times of the FPGA Loader device:

- Flash access time, or address-to-data time (t_{FLASH}). Standard parallel Flash devices are typically available in speeds from 60ns to 120ns.
- FPGA Loader data input setup time requirement (t_{SU_DATA})
- FPGA Loader delay from the clock input to the F_ADDRESS output ($t_{CO\_ADDRESS}$)

Note: t_{SU} and t_{CO} timing can be found using the Timing Analyzer tool within ispLEVER or Diamond after implementing the design.

Since the FPGA Loader uses a word look-ahead scheme, accesses to the Flash do not impose register-to-register timing restrictions on every clock. The Flash access is absorbed over 16 clock cycles, as reflected in the calculation below:

$$CCLK_{max} = \left(\frac{t_{CO\_ADDRESS} + t_{FLASH} + t_{SU\_DATA}}{16} \right)^{-1}$$

Appendix B. 16-bit Flash to Slave Parallel Configuration

VHDL toplevel file	flash_prog_load16_sp.vhd
Verilog toplevel file	flash_prog_load16_sp.v
Flash interface	Standard, 16-bit asynchronous read
FPGA interface	8-bit Parallel

Figure 12. FPGA Loader Logic Diagram for Flash to Parallel Configuration

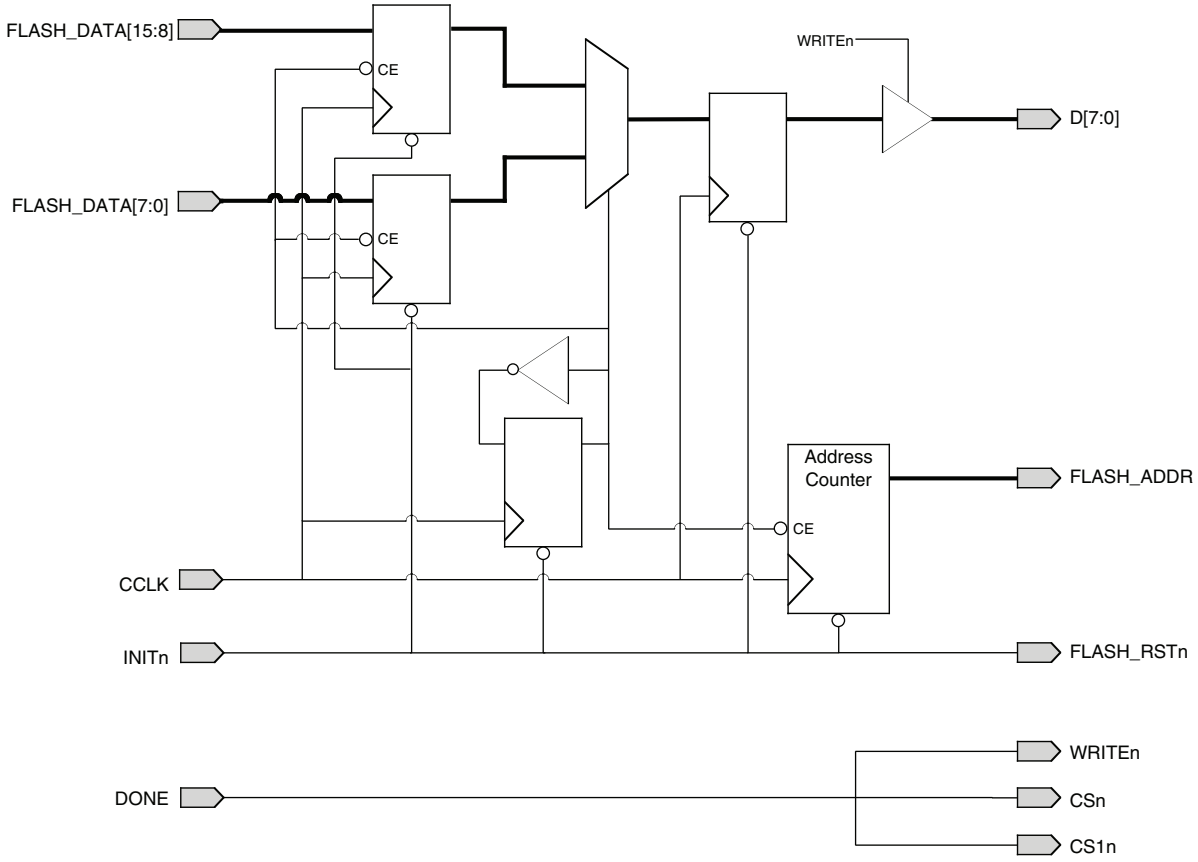
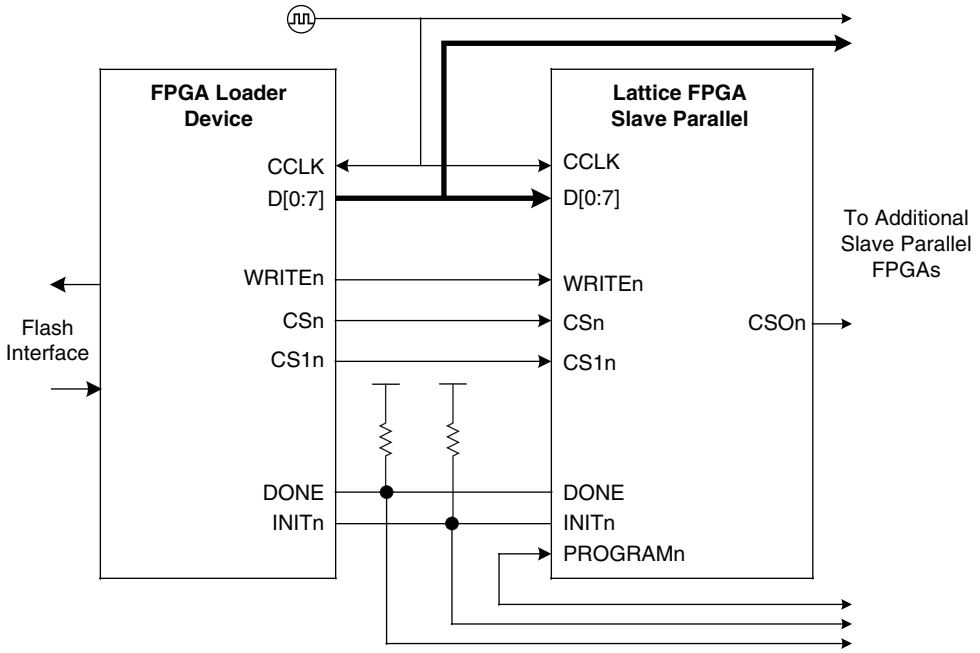


Figure 13. FPGA Loader to Slave Parallel FPGA Configuration Signal Connections



Configuration Speed

The maximum allowed configuration frequency is dependent upon the Flash device used and the setup and clock-to-output times of the FPGA Loader device:

- Flash access time, or address-to-data time (t_{FLASH}). Standard parallel Flash devices are typically available in speeds from 60ns to 120ns.
- FPGA Loader data input setup time requirement (t_{SU_DATA})
- FPGA Loader delay from the clock input to the F_ADDRESS output ($t_{CO\_ADDRESS}$)

Note: t_{SU} and t_{CO} timing can be found using the Timing Analyzer tool within ispLEVER or Diamond after implementing the design.

Since the FPGA Loader uses a word look-ahead scheme, accesses to the Flash do not impose register-to-register timing restrictions on every clock. The Flash access is absorbed over two clock cycles, as reflected in the calculation below:

$$CCLK_{max} = \left(\frac{t_{CO\_ADDRESS} + t_{FLASH} + t_{SU\_DATA}}{2} \right)^{-1}$$