XAPP800 (v1.1.1) May 7, 2008

Configuring Xilinx FPGAs with SPI Flash Memories Using CoolRunner-II CPLDs

Summary

This application note describes a method to configure Xilinx FPGAs, such as Spartan®-IIE and Spartan-3 FPGAs, using inexpensive small Serial Peripheral Interface (SPI) flash memories. These devices are physically small with low-pin count, small outline packages. Using a CoolRunner™-II CPLD, the SPI bus and protocol are converted to those which are used by the FPGA for configuration, specifically Master Serial configuration.

Introduction

SPI Flash based memories are inexpensive and are available in footprint-compatible, low-pin count, small outline packages that could reduce system cost if there was a way to interface the SPI bus and protocol to the configuration pins of the FPGA. Using a CoolRunner-II CPLD, this is now possible. The CPLD implements a simple state machine to read the SPI Flash memory using the SPI protocol, and then routes the serial data stream to the DIN pin of the FPGA. The cost of the CPLD is offset by the cost savings realized by using the SPI memory. Furthermore, the SPI Flash memory is available after configuration, offering cost-effective, random accessible, non-volatile data storage to the FPGA.

This application note discusses the design and provides reference source code, found at the end of this document. The CPLD source code contains commented VHDL generics that allow the designer to easily configure the design to implement the specific instruction set for several manufacturers of this type of memory, such as Numonyx™, Atmel™, NexFlash™, PMC™ and SST™. By default, the CPLD code interfaces to a Numonyx M25P20 memory and assumes only one memory device is used. The source code will need modification to support multiple memory devices. The CPLD source code has been tested with the Numonyx M25P20 memory implemented in hardware.

The iMPACT software utility included with the ISE software allows the designer to perform Program, Erase, and Verify operations to the SPI Flash memory after it has been soldered to the PCB by using a Platform Cable USB and a PC. The iMPACT software supports only select SPI flash memory families from Numonyx and Atmel. See the iMPACT software for the list of support devices.

SPI Basics

The SPI bus is a full duplex, synchronous serial data link. In other words, data is sent and received on the same clock edge, in a serial fashion. As such, there are 4 signals associated with SPI.

- SCK - Serial Clock
- MOSI - Master Out Slave In
- MISO - Master In Slave Out
- SS_n - Slave Select, active Low

Figure 1 displays these signals and a typical arrangement. This system is set up in a Master/Slave configuration where the Master clocks data out of the Master and into the Slave on the MOSI pin using SCK as the clock. During the same clock cycle, data is clocked out of the Slave and into the Master on the MISO pin, although this is done on the opposite clock edge

than MOSI. Several Masters and Slaves can coexist on the same bus, and therefore a method of arbitration is necessary when more than one Master attempts to gain access to the bus. However, this reference design assumes that there is only one Master on the bus and therefore no arbitration is implemented. Since more than one Slave can exist on the bus, the Master needs a method to select which Slave is to be addressed during the transaction. The signal SS_n selects the specified Slave. This reference design assumes only one Slave is present and therefore must be modified to select additional Slave devices.

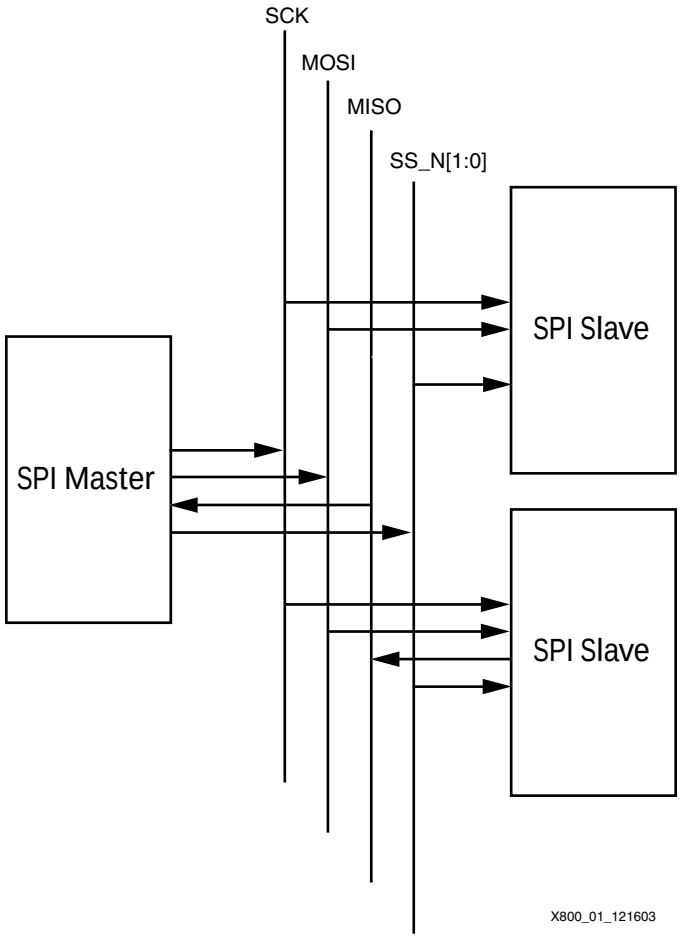

Figure 1: **Typical SPI System with a Single Master**

Clocking

SCK supports a variety of clock speeds, where this reference design assumes 20MHz in the simulation test bench. There are 4 types of clocking relationships the clock maintains with respect to the data. These are defined in the SPI Specification with two bits, CPOL and CPHA.

CPOL

CPOL dictates the idle state of the SCK. When CPOL is Low, the idle state of SCK is Low. Similarly, when CPOL is High, the idle state of SCK is High.

CPHA

CPHA indicates which clock edge the data is valid. When CPHA is Low, data is valid on the first edge of SCK, rising or falling. When CPHA is High, data is valid on the second edge of SCK. Whether the data is valid on a rising or falling edge is dependent upon the state of CPOL.

To illustrate the relationship between CPOL and CPHA, Figure 2 and Figure 3 are provided for reference, where Figure 2 displays the data transfer when CPHA is Low, and Figure 3 when CPHA is High.

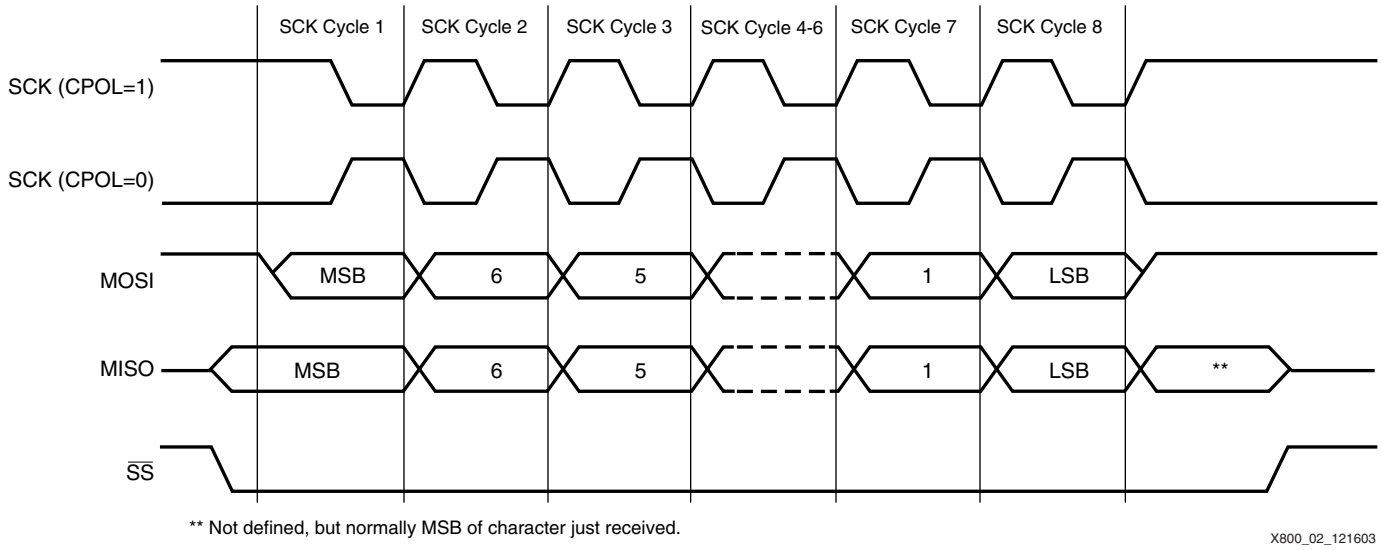

Figure 2: **Data Transfer on the SPI Bus with CPHA=0**

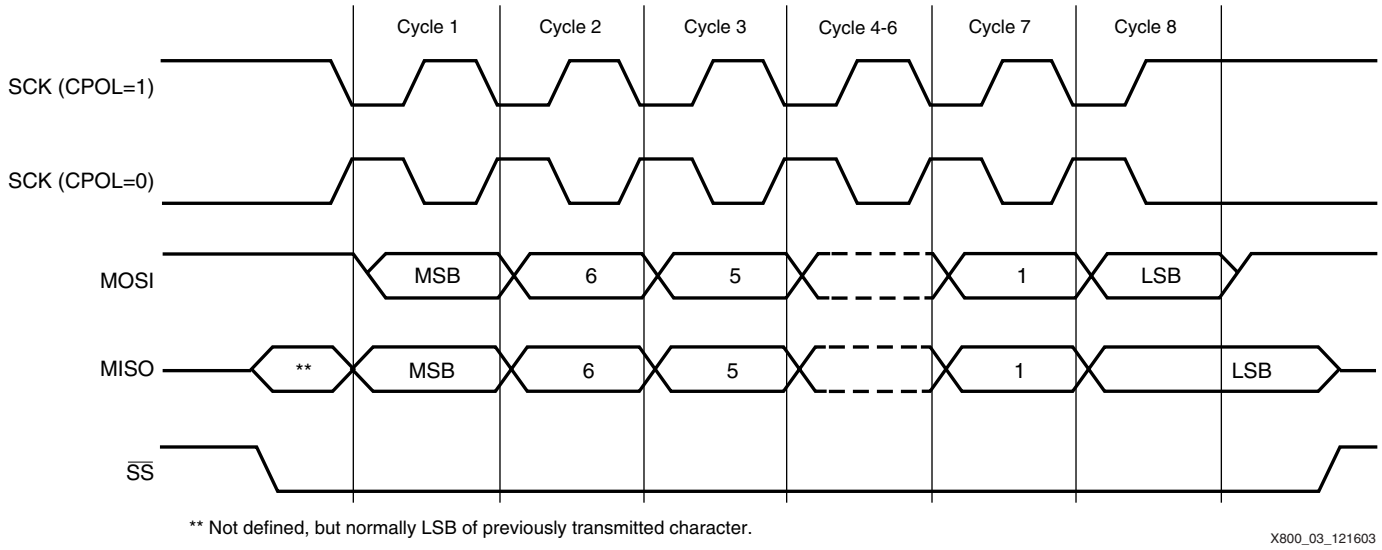

Figure 3: **Data Transfer on the SPI Bus with CPHA=1**

Data is clocked out of the device (Master or Slave) on one edge of the period, whereas data is clocked in on the next opposite edge of the period. For example, if data is clocked out of the Slave on the rising edge of SCK, then that data is clocked into the Master on the next falling edge of SCK. Simultaneously, the same sequence of data transfer occurs from the Master to the Slave. This full duplex data transfer can be viewed as a circular 16 bit shift register with 8 bits in the Master and 8 bits in the slave. The data simply trades places between the devices.

The SPI Specification does not describe a protocol for the data on the bus. Therefore, it is up to the designer to specify the protocol of the data transfer. SPI Flash memory devices define their particular protocol in the respective data sheets. Please consult the manufacturers data sheet for the appropriate protocol. Note that in particular, manufacturers implement different addressing schemes.

Timing relationships must follow those specified in both the SPI Specification and the SPI Flash memory data sheet.

System Design

This reference design implements an SPI interface between a Spartan-IIE FPGA and a Numonyx M25P20, 2 Mbit SPI Flash memory. A working prototype has been built to test the reference design and uses a CoolRunner-II CPLD (XC2C32-4-VQ44) to perform the data transfer between the SPI bus and the FPGA configuration signals. Other Xilinx FPGAs may be used with this reference design, and the subsections below describe the unique details found using the Spartan-IIE and the Spartan-3 FPGAs.

Note: This design was tested on the Numonyx M25P20, but will work with other M25Pxx parts, including M25P05, M25P10, M25P40, and M25P80.

Figure 4 shows how this reference design should be implemented in a system with an SPI Flash memory. This implementation provides a method of configuring the FPGA using an SPI based memory upon power up, or after the FPGA requests a reconfiguration. Once the FPGA has been configured, the FPGA may access the SPI memory as user space. This reference design provides for three methods of programming the SPI memory while mounted to the PCB. All of these scenarios are discussed in the subsections below.

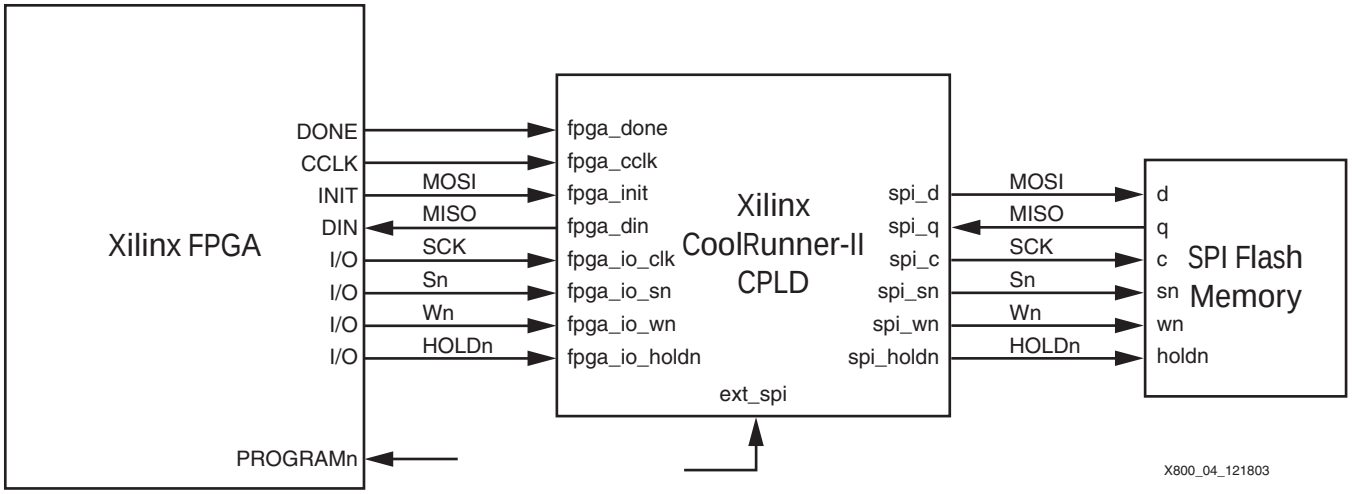

Figure 4: **SPI Memory Interface to a Xilinx FPGA**

FPGA Configuration

Upon power up, the FPGA requests configuration data using Master Serial mode in the usual manner. Both the DONE and INIT pins go Low, followed by the INIT pin going High indicating the start of configuration is requested. The state machine in the CPLD recognizes this event and begins to set up the SPI memory for a read operation.

The operation begins by bringing spi_sn Low, followed by passing CCLK through to spi_c and issuing the FAST_READ instruction. Once the instruction has been sent, the starting address is immediately transmitted followed by dummy bits. When the prescribed number of dummy bits has been sent to the memory, configuration data appears on the q pin of the memory. The state machine in the CPLD recognizes this event and passes the data through to the DIN pin of the FPGA. This data transfer continues until the DONE pin goes High. At this point, the CPLD does one of two things to its I/Os depending upon a VHDL generic setting in the source code. These two options are described in the next subsection.

After the FPGA has been successfully configured, the CPLD monitors the DONE pin to determine when the FPGA requests a reconfiguration cycle. When the DONE pin goes Low and the INIT pin cycles Low then High, the CPLD begins the SPI transaction again and presents the data to the FPGA in a Master Serial mode. FPGA reconfiguration is initiated by applying a Low value to the PROGRAMn pin of the FPGA.

SPI Memory as FPGA User Space

This reference design allows the designer to specify if the CPLD will be used as a simple buffer after FPGA configuration, or if the CPLD becomes electrically invisible to the FPGA I/Os after configuration has completed. The way the designer specifies either mode is by modifying the VHDL generic cpld_buffer found in the VHDL file spi_cpld.vhd. If this generic constant is set to a '1', the CPLD will become a buffer between the FPGA I/Os and the SPI memory I/Os. This allows the FPGA to use the SPI memory as a user space post-configuration. Alternately, if the generic cpld_buffer is set to a '0', the CPLD will become invisible to the FPGA I/Os so that it does not interfere with future transactions on these I/Os. The CPLD I/Os in this case go to a High-Z mode. It is up to the user do specify weak pull-ups on the CPLD I/Os as needed in the particular system.

Although the SPI Flash memory is mainly used to store FPGA configuration data, the memory can also be used as working memory space by the FPGA one it has been configured. As discussed earlier, the CPLD becomes a simple buffer and passes all SPI bus signals between the FPGA and the SPI memory post FPGA configuration. These signals can be then driven by the FPGA as needed to conduct memory transactions. The SPI bus includes the typical SCK, MOSI, MISO and SSn signals, but provided in this scenario are two other signals: Wn and HOLDn. Some memories contain these signals which provide additional control of the memory. This reference design is based on the Numonyx M25P20 where Wn is a write protect signal, active Low. HOLDn is an active Low hold signal used to pause any serial communications with the device without actually deselecting the device on the SSn pin.

The FPGA can be set up with a design that uses this memory as working space. This may be in the form of a state machine or a microcontroller, etc. This allows the FPGA to not only use the unused memory as user storage, but can also be used to overwrite the configuration data in the SPI memory. For example, if SPI Flash stored both the FPGA configuration data and the application code for an embedded controller, then it would be possible to update both hardware and software by writing a new image to the Flash. Alternatively, the configuration bits in the memory could be overwritten or erased by the FPGA in a security type application.

Programming the SPI Memory

There are 3 primary methods of programming the SPI memory as described below.

3rd Party Programmer

The most obvious method of programming the SPI memory is using a 3rd party programmer as would be done with PROMs. The advantage with this method is that mass production programming is possible. But the disadvantage is that it will be difficult to reprogram the device on the PCB unless the FPGA or some other controller on the SPI bus is set up to perform this operation.

JTAG Chain

The CoolRunner-II CPLD has JTAG test capabilities which include the standard PRELOAD and EXTEST commands. Using these commands, it is possible to drive and sample the pins of the CPLD with the JTAG chain and thereby stimulate the pins of the SPI memory via the traces routed on the PCB. This method is shown in Figure 5. Using a tool that understands the JTAG protocol as well as converts the applied data to SPI bus relationships, the SPI memory can be programmed via the JTAG chain of the CPLD

The advantage with this is by adding minimal hardware to the board, the memory can be quickly programmed. The added benefit is that the user now has access to the CPLD for configuration as well. The generation and execution of the EXTEST vectors is the responsibility

of the user.

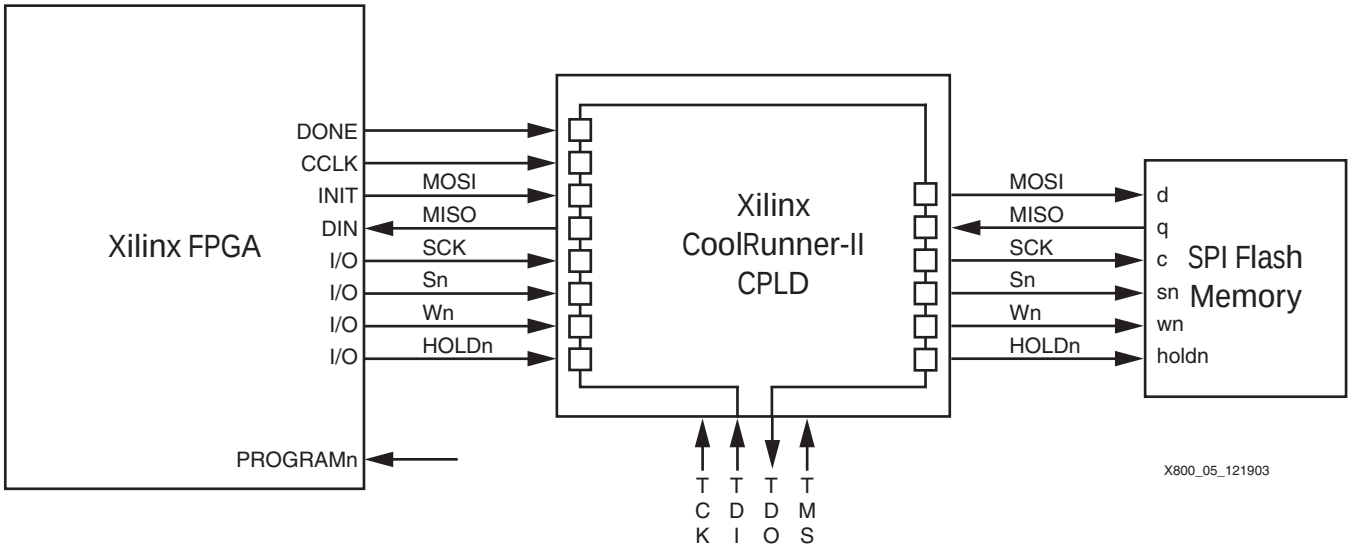

Figure 5: **Programming SPI Flash with the JTAG Chain**

Directly

It is also possible to program the SPI Flash Memory directly on the PCB by adding a Platform Cable USB connector to the PCB which taps into the SPI bus as shown in Figure 6. This method requires a cable and minimal hardware to connect to the bus. In addition, the ISE iMPACT software must be installed on a PC in order to control the SPI bus through the Platform Cable USB.

To facilitate this method, the CPLD must have its I/Os in a high impedance state with no weak pullup so that it does not interfere with SPI bus transactions during the programming operation. A signal has been added to one CPLD I/O which allows for the CPLD to go into a High-Z condition when needed. The pin that controls this function is called ext_spi and is shown in Figure 4. When ext_spi is brought High by an external signal, such as the cable used to program the memory, the CPLD I/Os that interface to the memory as well as those to the FPGA

will go into a High-Z condition. Placing a Low value on the ext_spi pin allows the CPLD state machine to drive the I/Os when required by the FPGA.

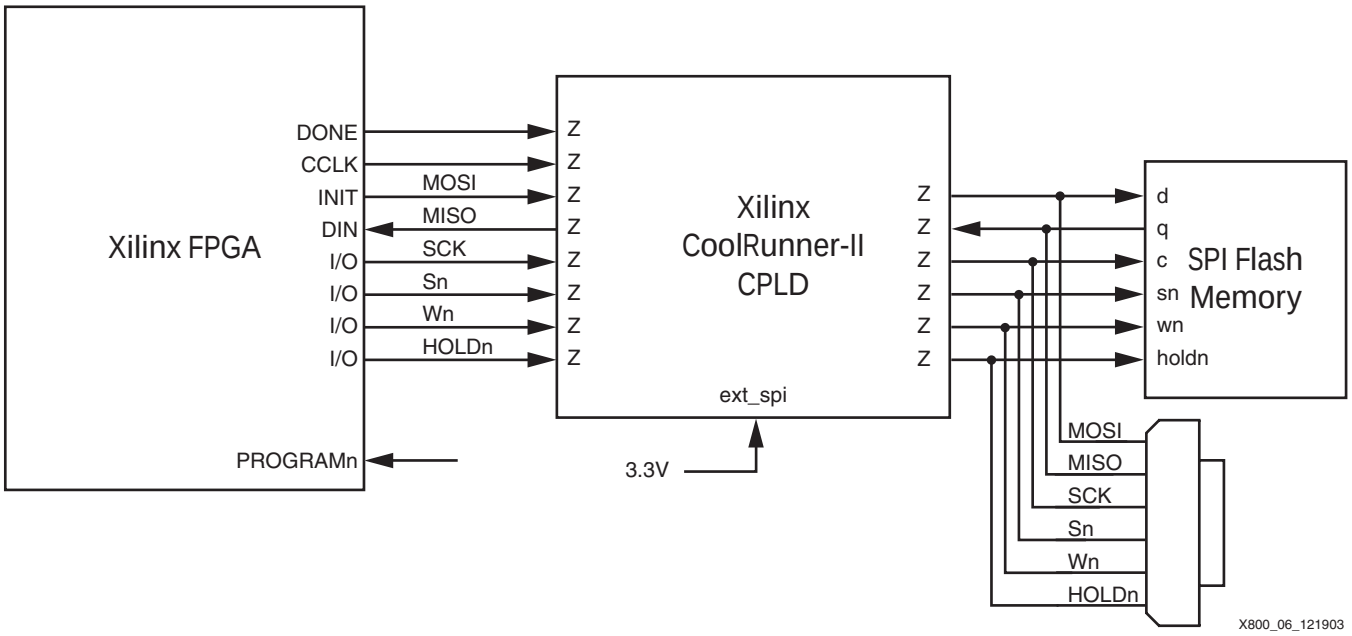

Figure 6: **Programming SPI Flash with Direct Cable**

Spartan-IIE

Although this reference design targets a CoolRunner-II CPLD (XC2C32-4-VQ44), the device selection can be substituted for any other Xilnx CPLD. In addition, this reference design targeted a Spartan-IIE FPGA (XC2S200E PQ208) for testing purposes. Other Xilinx FPGAs may be used with this reference design.

The most important thing to consider when using the Spartan-IIE is the I/O standard selected on the I/Os. SPI memories are typically 3.3V driven devices. Although the CoolRunner-II CPLDs have I/O banking, the 32 macrocell device (XC2C32) only has one bank. Therefore, the FPGA and the memory must match I/O voltage capabilities. The CPLD, in this case, must be configured with LVCMOS33 I/O functionality to properly interface to the SPI bus. Consequently, the Spartan-IIE must be configured with 3.3V LVTTL I/Os where the Vcco pins of Bank 2 and Bank 3 are supplied with 3.3V.

Spartan-3

As an example of another Xilinx FPGA that can be configured using this method, the Spartan-3 FPGA can be set up with the CoolRunner-II CPLD for SPI transactions. The Spartan-3 dual purpose configuration pins, in this case DIN and INITn, should be powered to 3.3V via V_{CCO_4} to interface to the 3.3V I/Os of the CPLD. Spartan-3 dedicated configuration pins, in this case DONE and CCLK, are set up for 2.5V signaling and therefore require 2.5V supplied to V_{CCAUX}. To successfully interface to the 3.3V I/Os of the CPLD, modifications are necessary to the interface between the FPGA and the CPLD as described in the Spartan-3 Functional Description Data Sheet, **DS099-2**. According to this data sheet, it is advisable to add a pullup resistors to both DONE and CCLK which are terminated to 3.3V. Both of these pins are set up as outputs in the Master Serial configuration mode. The pullup resistors must be selected to provide proper edge rates to the CPLD as described in the **CoolRunner-II CPLD Data Sheet**. In addition, the power supply for V_{CCAUX} must be able to tolerate reverse currents.

Again, a different Xilinx CPLD may be selected for the interface to acquire a different voltage match between these components.

Software Flows for SPI File Preparation and Programming

Preparing an SPI PROM File

This section provides guidelines and the software flow to create PROM files for SPI PROM. Before converting a FPGA bitstream into a SPI-formatted PROM file, the designer must verify the bitstream was generated with the **`bitgen -g StartupClk:Cclk`** option. This option ensures proper FPGA functionality by synchronizing the startup sequence to the internal FPGA clock.

The Xilinx ISE software tools, PROMGen or iMPACT, generate SPI-formatted PROM files from the FPGA bitstream. The SPI PROM serially outputs data bytes MSB first, while Xilinx PROMs output data LSB first. An SPI-formatted PROM file is bit-reversed within each byte from a standard Xilinx PROM file.

Preparing an SPI PROM File Using the ISE PROMGen Command-Line Software

The Xilinx ISE PROMGen software takes an FPGA bitstream (`.bit`) file as input and, with the appropriate options, generates a memory image file for the data array of an SPI PROM. The output memory image file format is chosen through a PROMGen software command-line option. Typical file formats include Intel Hex (`.mcs`) and Motorola Hex (`.exo`).

The ISE PROMGen software utility is easily executed from a command-line (see Table 5, page 10 for ISE PROMGen software options used for SPI PROM file generation). An example PROMGen software command-line to generate an mcs-formatted file for a 64-megabit (8192 kilobytes) SPI PROM is:

```
promgen -spi -p mcs -o spi_prom.mcs -s 8192 -u 0 bitfile.bit
```

The **`-spi`** option is required to ensure proper bit ordering within the SPI PROM file. The **`-p mcs`** option specifies Intel Hex (`.mcs`) output file format. The **`-o spi_prom.mcs`** specifies output to the `spi_prom.mcs` file. The **`-s 8192`** specifies a PROM file image size of 8192 kilobytes. The **`-u 0`** option specifies the data to start at address zero and fill the data array in the up direction. The `bitfile.bit` is the input bitstream file.

Table 1 list the various PROMGen options and the functions.

Table 1: **Example PROMGen SPI PROM File Options**

PROMGen Option	Description
`-spi`	Used to maintain the correct bit ordering required to configure the FPGA from an SPI serial flash device.
`-p <format>`	PROM output file format. Commonly accepted PROM file formats include Intel Hex (`.mcs`) and Motorola Hex (`.exo`).
`-s <size>`	Specifies the PROM size in kilobytes. The PROM size must be a power of two for this option, and the default setting is 64 kilobytes.
`-u <address>`	Loads the `.bit` file from the specified starting address in an upward direction. This option must be specified immediately before the input bitstream file.

Preparing an SPI PROM File Using the ISE iMPACT Graphical Software

The ISE iMPACT 10.1 (or later) software integrates PROM file formatting and in-system programming features behind an intuitive graphical user interface (GUI). The PROMGen file formatting functionality is provided through a step-by-step wizard in the iMPACT software. The wizard steps through the output PROM file options and input bitstream selections. A final step is required for iMPACT to generate the PROM file.

In the iMPACT software, an SPI PROM file can be generated from a Xilinx FPGA bitstream through a simple eight step process. A prescribed sequence of dialog boxes (also known as a wizard) acts as a guide through most of the PROM file generation process.

The following section demonstrates the iMPACT software process for generating an SPI-formatted PROM file in the MCS file format for a 64 Mb SPI PROM. The demonstrated process takes the `bitfile.bit` FPGA bitstream file as input and generates a PROM file named `spi_prom.mcs`.

Step 1: Create a New Project for PROM File Generation

After launching the iMPACT software, the iMPACT project dialog box is displayed (Figure 7). Choose the "create a new project (.ipf)" option. Optionally, specify a project location via the **Browse...** button. Then, click **OK** to continue to step 2 in the process.

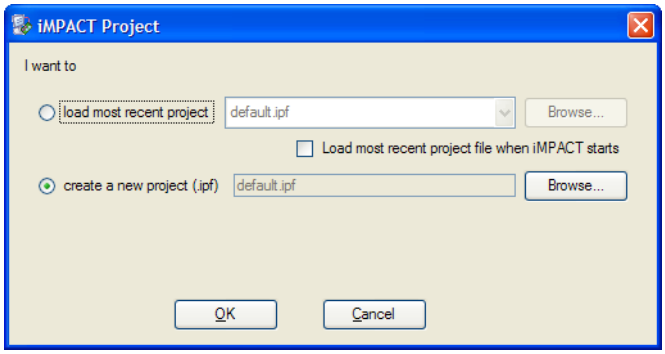

Figure 7: **Create a New Project for PROM File Generation**

Step 2: Choose to Prepare a PROM File

The first dialog box of the wizard displays the available kinds of projects that can be created (Figure 8). Check “Prepare a PROM File,” and select **Next** to proceed to step 3 of the process.

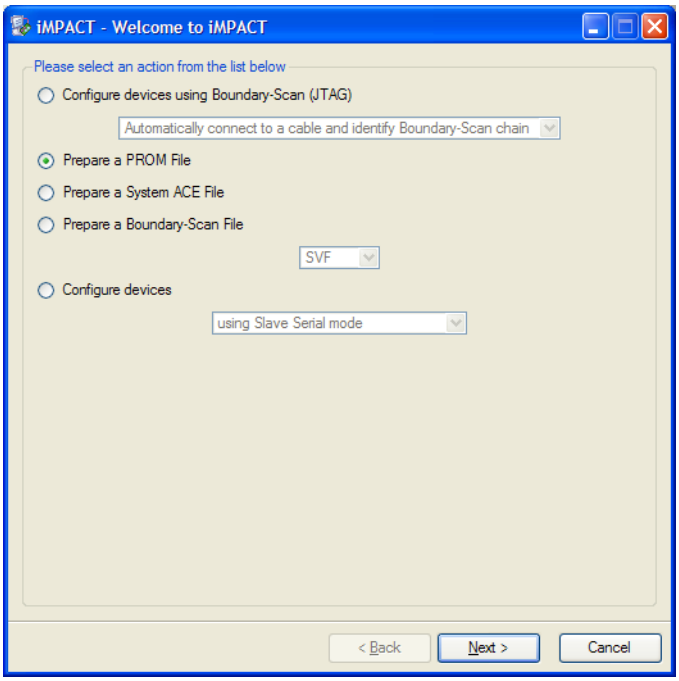

Figure 9: **Choose to Prepare a PROM File**

Step 3: Specify the Output SPI PROM File Options

The third step of the process is to specify the targeted PROM type, the PROM file format, and output file name and location (Figure 10). Choose to target the "3rd-Party SPI PROM" type. Select the "MCS" PROM file format. Maintain the default "Checksum Fill Value" which is a hexadecimal FF byte value. Specify the PROM file name to be `spi_prom` (to the spi_prom name, iMPACT automatically adds the `.mcs` file name extension corresponding to the chosen MCS PROM file format). Specify a desired directory location for the output `spi_prom.mcs` file. Click **Next** to continue to step 4.

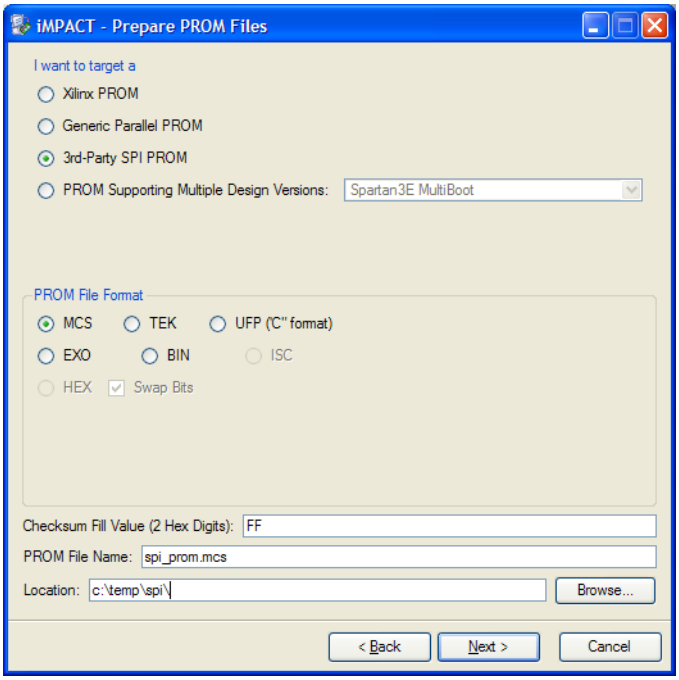

Figure 11: **Specify the Output SPI PROM File Options**

Step 4: Specify an SPI PROM Density

The fourth step of the process is to specify the size of the target SPI PROM (Figure 12). From the "Select SPI PROM Density" drop-down list, choose the 64M value that matches the 64 Mb size of the targeted SPI PROM in this demonstration. Click **Next** to proceed to step 5 of the process.

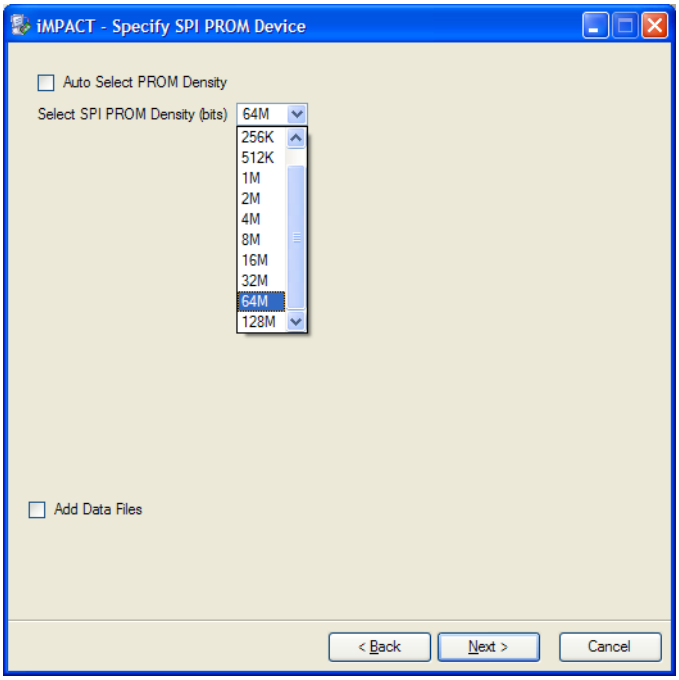

Figure 13: **Specify an SPI PROM Density**

Step 5: Summary of SPI PROM File Selections

The fifth step displays a summary of the options selected from the prior steps in the process (Figure 14). The summary shows that a PROM file in the `.mcs` file format with a fill value of hexadecimal FF is to be written to a file with a root name of `spi_prom` for a 64 Mb SPI PROM. Click **Finish** to complete the wizard and proceed to step 6 of the process.

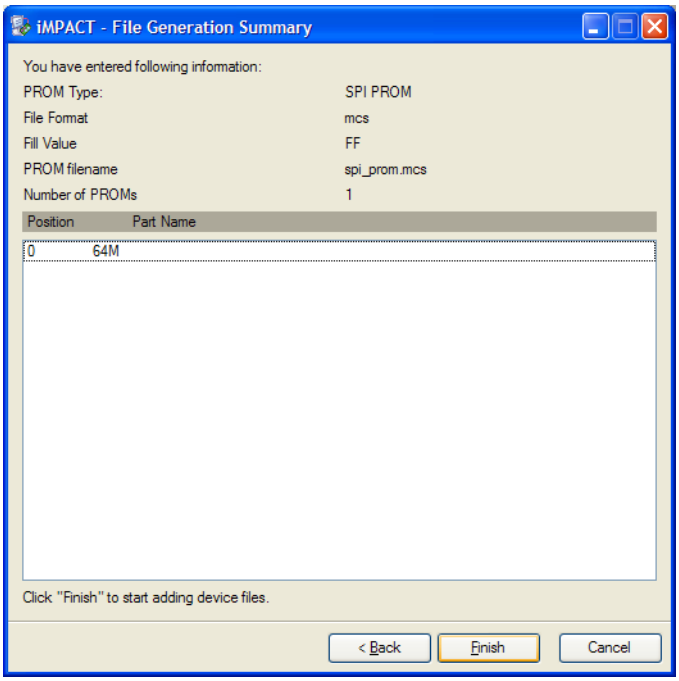

Figure 15: **Summary of SPI PROM File Selections**

Step 6: Automated Notification to Add a Device File to the SPI PROM File

After the iMPACT project wizard is finished, the iMPACT SPI PROM generation project is set to generate a specific PROM file with specific parameters. At this stage, the PROM file memory image is empty. The sixth step is to add an FPGA bitstream to the PROM file memory image. This step begins immediately after completion of the iMPACT project wizard with an automatic notification that the next step is to add a device file to the SPI PROM memory image. Click **OK** in the Add Device notification dialog box (Figure 16) to proceed to step 7.

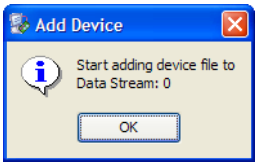

Figure 17: **Add Device Notification Dialog Box**

Step 7: Select the FPGA Bitstream File to Add to the SPI PROM Memory Image

After the Add Device notification, iMPACT automatically opens a file browser to select the FPGA bitstream (`.bit`) file to add to the SPI PROM memory image (Figure 18). Select the FPGA bitstream file to be written to the SPI PROM. Click **Open** in the browser to add the selected FPGA bitstream to the SPI PROM memory image. This action completes the automated iMPACT process for preparing a SPI PROM file to be generated. Proceed to step 8 to generate the SPI PROM file.

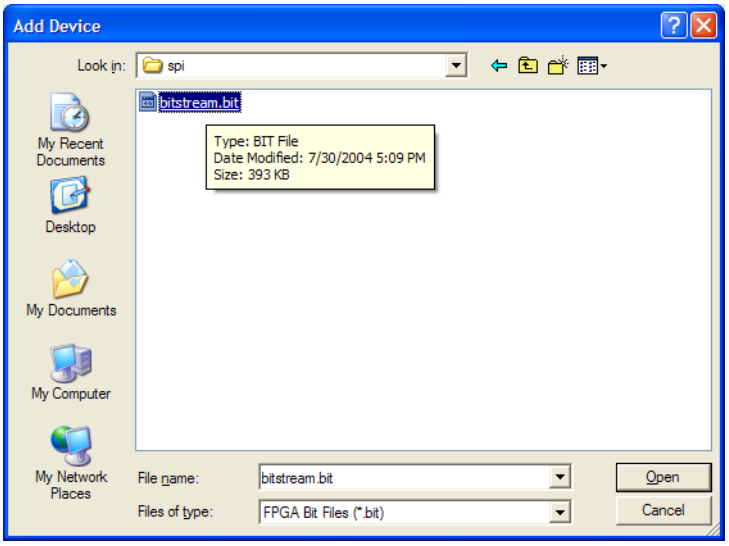

Figure 19: **Add Device File Browser**

Note: iMPACT 10.1 does not permit the addition of some FPGA devices as the first FPGA from the SPI PROM. The workaround is to first add a small FPGA bitstream of a device type that iMPACT permits, add the actual desired bitstream as the second FPGA, and then, delete the first bitstream after the add design process completes. An alternative is to use the PROMGen command-line tool to generate the PROM file from the FPGA bitstream. See the "Preparing an SPI PROM File Using the ISE PROMGen Command-Line Software," page 8 section.

Step 8: iMPACT Generate File Operation

The eighth and final step is to generate the PROM file. Under the iMPACT Operations menu, invoke the **Generate File** menu item (Figure 20, page 14). Once invoked, the Generate File menu item causes iMPACT to generate the specified SPI PROM file.

iMPACT reports a "PROM File Generation Succeeded" message after successful generation of the SPI PROM file.

After the Generate File operation has completed, the generated `spi_prom.mcs` file is available in the specified location. The `spi_prom.mcs` file can be used in any of the supported programming solutions to program the SPI PROM with the specified FPGA bitstream contained within the SPI PROM file.

Save the iMPACT SPI PROM generation project for quick regenerating of the SPI PROM file whenever the FPGA bitstream design is revised. To regenerate a SPI PROM file, reopen the saved iMPACT project, and invoke the Generate File operation. iMPACT generates a revised SPI PROM file from the new version of the FPGA bitstream file, assuming the revised bitstream file is located in the same location as the original bitstream file.

If a project is not loaded when using the iMPACT GUI interface, a user is guided through the wizard steps each time to create a new SPI-formatted PROM file. The designer is prompted to name the project and select the option "Prepare a PROM File," following "Step 1: Create a New Project for PROM File Generation," page 9, through "Step 8: iMPACT Generate File Operation," page 13, to generate a new SPI file.

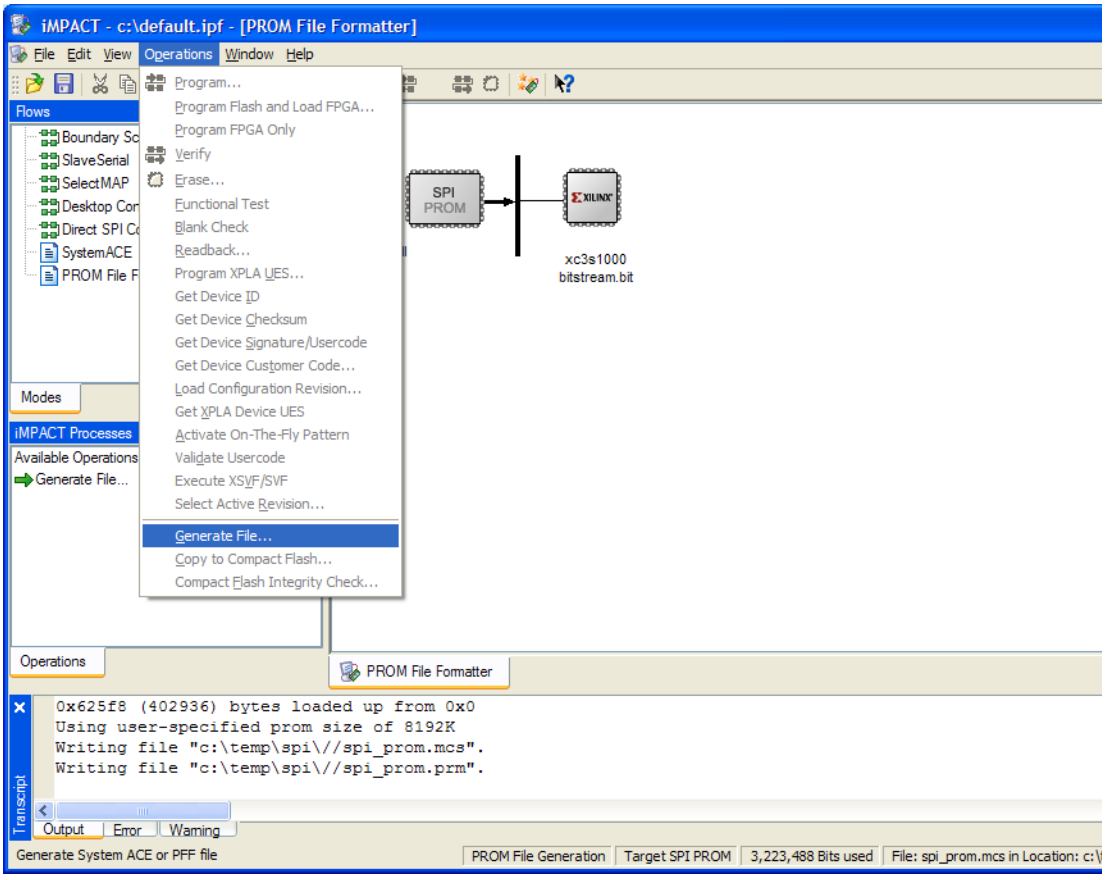

Figure 20: **Generate File Menu**

Using the ISE iMPACT Software to In-System Program SPI PROMs

In prototype applications, the ISE iMPACT 10.1 (or later) software can be used to in-system program select SPI serial flash devices with a memory image from a given SPI PROM file (see "Preparing an SPI PROM File," page 8 for instructions on the generation of a SPI PROM file).

Table 2 lists the selected SPI serial flash memories that can be programmed with iMPACT.

Table 2: **SPI Serial Flash Programming Capability with iMPACT**

SPI Serial Flash Vendor	Family(1)
STMicroelectronics	M25P, M25PE, M45PE
Atmel	AT45DB

Notes:

1. Refer to the *iMPACT Software Manual* for SPI support.

The iMPACT software can program select SPI PROM using a simple seven-step process. A prescribed sequence of dialog boxes (or wizard) acts as a guide through most of the iMPACT programming process.

The following section demonstrates the iMPACT software process for in-system programming a M25P64 (64 Mb) STMicroelectronics SPI PROM. The demonstrated process takes the `spi_prom.mcs` SPI PROM file (generated in "Preparing an SPI PROM File," page 8 section) as input, erases the SPI PROM, programs the PROM file contents into the SPI serial flash device, and verifies the SPI PROM contents against the given SPI PROM file contents.

Step 1: Create a New Project for Direct In-System Programming

After launching the iMPACT software, the iMPACT Project dialog box is displayed (Figure 7, page 9). Choose the "create a new project (.ipf)" option. Optionally, specify a project location via the **Browse…** button. Then, click **OK** button to continue to step 2 in the process.

Step 2: Configure Devices Using the Direct SPI Configuration Mode

The second step begins with the iMPACT project wizard. The first dialog box of the wizard displays the available kinds of projects that can be created (Figure 21). Choose the "Configure devices" option. Then, select the **using Direct SPI Configuration mode** item from the associated drop-down list box. Click **Finish** to complete the new project setup process. At the completion of this process, iMPACT is set into a mode for in-system programming SPI serial flash memories using a direct cable connection to the SPI bus.

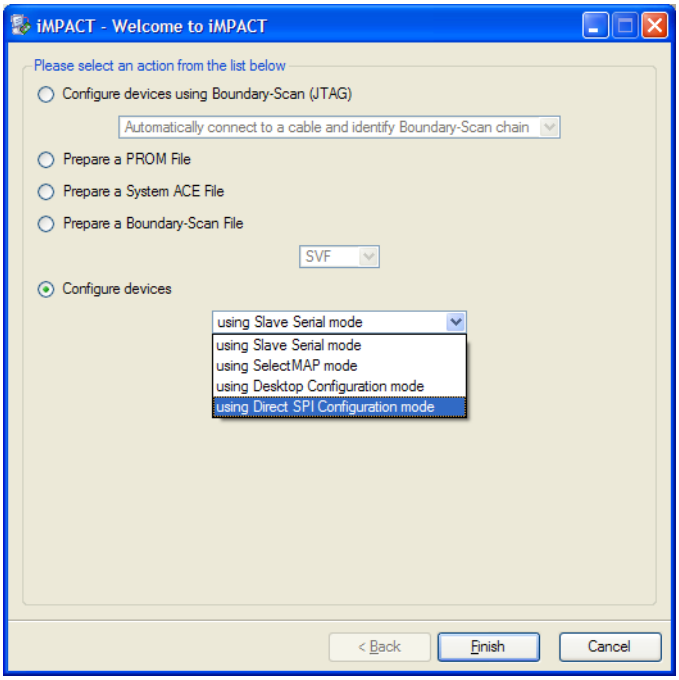

Figure 21: **Configure Devices Using the Direct SPI Configuration Mode**

Step 3: Add a SPI PROM File

After finishing the new project wizard, iMPACT automatically leads into the third step of the process. iMPACT automatically displays a file browser window to select a SPI PROM file for programming into the SPI serial flash device (Figure 22). Choose the `spi_prom.mcs` file, and click **Open**.

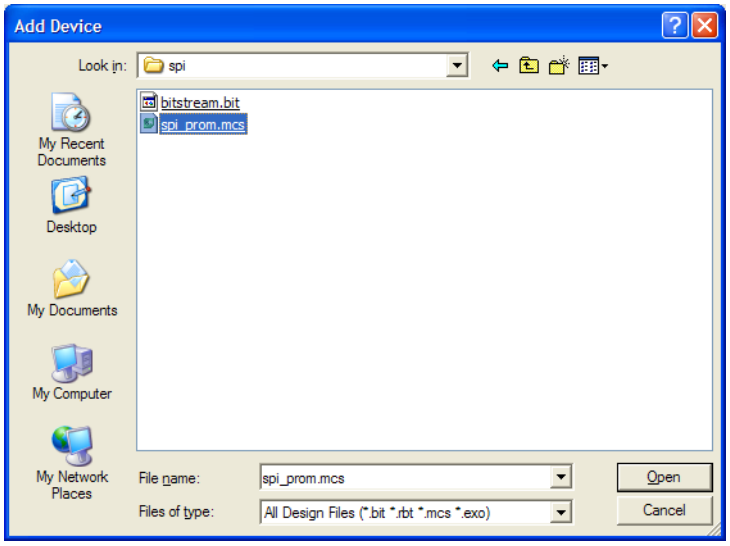

Figure 22: **Add a SPI PROM File**

Step 4: Select STMicroelectronics M25P64 Device Part Number

After selecting the SPI PROM file to load, iMPACT displays the Select Device Part Name dialog box (Figure 23). The fourth step of the process requires the target type of SPI PROM to be specified in this dialog box. Select the STMicroelectronics M25P64 part number for the target SPI PROM type used in this demonstration. Click **OK** to complete the SPI PROM programming setup.

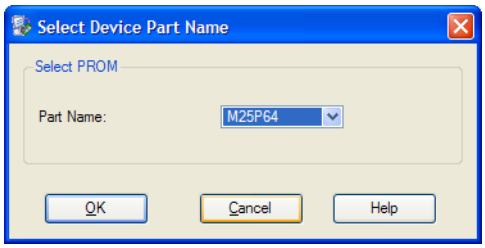

Figure 23: **Select Device Part Name Dialog Box**

Step 5: Specifies the Hardware Requirements for In-system Programming

The fifth step of the programming process specifies the hardware requirements for in-system programming of the SPI PROM:

- **Proper Xilinx cable connection:** The Xilinx cable must be properly connected to the computer and to the SPI bus of the target SPI PROM (see Figure 2, page 4 for hardware connections from the Xilinx cable to the SPI bus of the target SPI PROM).
- **Cable power:** If using the Xilinx Parallel Cable IV or Xilinx MultiPRO cable, then power must be applied to the cable.
- **Target system Power:** Power must also be supplied to the target system containing the SPI PROM.
- **Isolate target FPGA signals during the SPI programming process:** The target FPGA (and any other potential SPI PROM master device on the SPI bus) must be put into a mode to keep its SPI pins in an high-impedance state. Methods to put the FPGA SPI pins in a high-impedance state are described in "Hardware and Connections for SPI Programming," page 10. One common method requires the FPGA PROG_B pin to be held Low. The high-impedance condition of the FPGA SPI pins must be maintained throughout the remainder of the SPI PROM programming process in order to avoid contention with the in-system programming cable.

Step 6: Invoke the iMPACT Program Operation

The sixth step of the process programs the target SPI PROM with the selected SPI PROM file contents. Ensure the SPI PROM icon in the iMPACT window is selected by left-clicking on the SPI PROM icon (the SPI PROM icon is highlighted in green when selected). Select **Operations** → **Program** to begin programming (Figure 24).

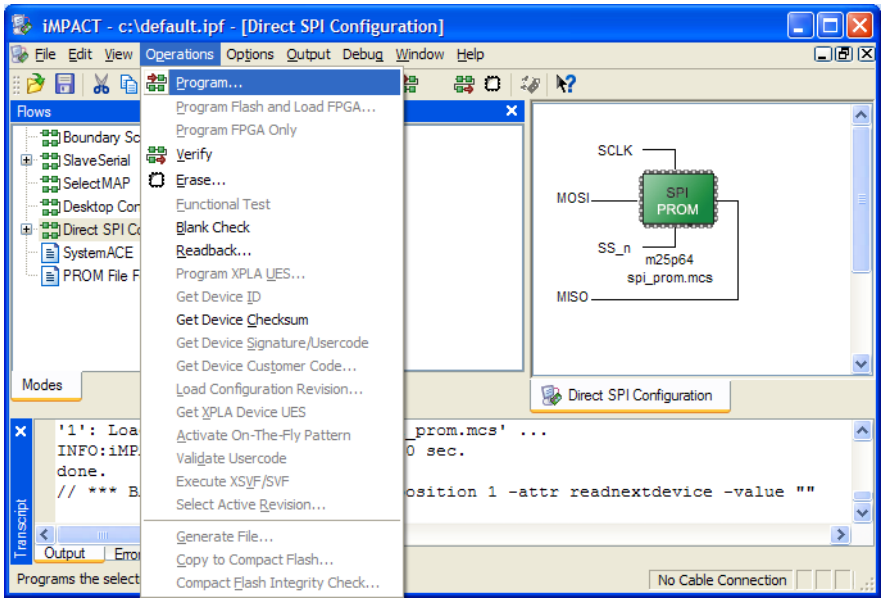

Figure 24: **Program Menu**

Note: Before you invoke the **Operations** → **Program** menu item, iMPACT 10.1 requires you to separately invoke the **Edit** → **Set Programming Properties** menu item to set the programming properties for the SPI PROM. The **Erase Before Programming** and **Verify** programming properties must be set.

Step 7: Select iMPACT Programming Properties

In response to the invocation of the **Program** operation, iMPACT presents the Programming Properties dialog box (Figure 25). The seventh step of the process ensures the selection of proper programming properties. Ensure that both the **Verify** and the **Erase Before Programming** options are checked, ensuring proper programming of the SPI PROM. Click **OK** to begin the erase, program, and verify operations.

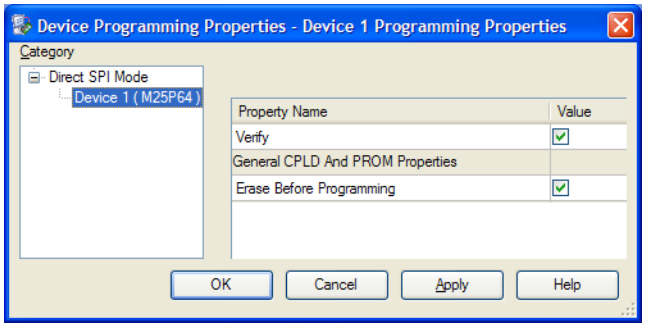

Figure 25: **Programming Properties Dialog Box**

At the start of the programming operation, iMPACT automatically connects to the cable attached to the computer. Then, iMPACT displays a Progress Dialog box as it progresses through the in-system erase, program, and verify operations (Figure 26). Depending on the size of the SPI PROM, size of the SPI PROM file image, and speed of the cable configuration, the programming operation can take anywhere from a few seconds to a few minutes to complete.

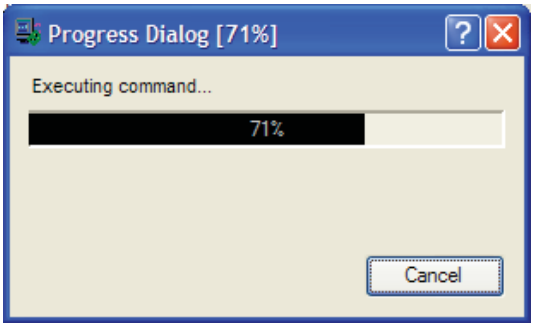

Figure 27: **Progress Dialog Box**

At the end of a successful Program operation, iMPACT reports a "Program Succeeded" message.

Save the iMPACT Direct SPI Configuration Mode project for quickly reprogramming the SPI PROM whenever the SPI PROM file is revised. To reprogram the SPI PROM, reopen the saved iMPACT project, and invoke the **Program** operation, ensure the selection of the **Erase** and **Verify Programming Properties**, and click **OK**. iMPACT reprograms the SPI PROM, assuming the revised SPI PROM file is located in the same location as the original SPI PROM file.

CPLD Design

This reference design, as implemented in the XC2C32-4-PC44, can obtain speeds of over 300 MHz. However, the limiting factor with speed is the SPI Flash memory which is typically limited to 25 MHz. The user must specify FPGA configuration speed by selecting the configuration speed using BitGen found in Xilinx ISE software. Note that the fastest FPGA configuration speed for Xilinx FPGAs is faster than the maximum speed of most SPI Flash memories.

State Machine

The main functionality of this reference design focuses around one state machine, found in processes named statem_comb and statem_reg. The following discussion references Figure 28.

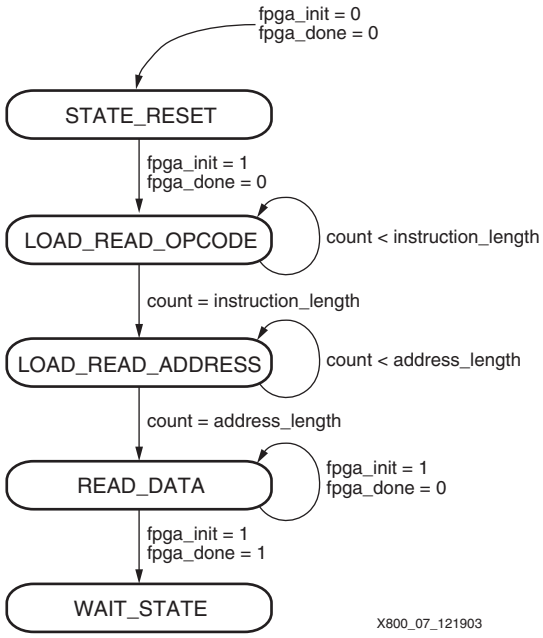

Figure 28: **State Machine**

STATE_RESET State

The state machine is clocked by fpga_cclk which arrives directly from the FPGA CCLK pin. When the fpga_init (INITn) and fpga_done (DONE) pins go Low, the state machine is reset to the STATE_RESET state.

While in this state, the variable bit width counter is reset. This counter tracks the number of instruction bits, address bits, and dummy bits sent to the SPI bus in subsequent states.

The STATE_RESET state also applies High values to fpga_din (DIN) to provide benign data to the FPGA DIN pin. This is necessary because there is a finite amount of time that the state machine must set up the SPI memory before data is read out of the memory. Meanwhile, the FPGA Master Serial configuration mode is expecting data to arrive on DIN once INITn goes High. The entire state machine waits to submit valid data to DIN until it reaches the READ_DATA state. To avoid erroneous data being applied to the DIN pin during this initial phase, harmless data in the form of 1's must be sent to DIN. During this time frame, the FPGA simply waits for the synchronization word to arrive, which indicates the beginning of valid configuration data.

Once in the STATE_RESET state, the state machine waits for the condition where fpga_init goes High, and fpga_done is still Low. When this occurs, the state machine progresses to the LOAD_READ_OPCODE state.

LOAD_READ_OPCODE State

This state sends the READ or FAST_READ instruction code to the SPI memory to set up the device for a read operation. The instruction is stored in the generic variable READ_INSTRUCTION and can be customized by the user for the specific SPI memory. Sample instruction codes are provided in the generic statement for some common memories.

In this state, spi_sn is brought Low to initiate communication with the SPI memory as shown in Figure 2 and Figure 3. At the same time, the counter is enabled and begins tracking the number of instruction bits sent to the SPI memory.

While in this state, instruction code data is placed on the spi_d pin and is shifted out on every clock edge, while the counter tracks the number of instruction code bits sent. The data and spi_sn signal occur on the opposite edge of fpga_cclk than the state machine to retain proper timing.

Simultaneously, 1's continue to be applied to the DIN pin of the FPGA via the fpga_din output signal.

When the counter reaches the bit length of the instruction as specified by the generic variable INSTRUCTION_LENGTH, the state machine transitions to the next state LOAD_READ_ADDRESS.

LOAD_READ_ADDRESS State

The purpose of this state is to shift out the starting address of the data located in the SPI memory followed by shifting out dummy bits as required by the particular memory. As such, spi_sn is held Low as it was in the previous state and the counter continues to increment without a reset.

The counter tracks the number of address bits shifted as well as the number of dummy bits. With this in mind, the terminal count is the sum of the number of address bits, dummy bits, and instruction bits (since the counter is not reset). Therefore, the user must set the size of variable width counter to a value which corresponds to the sum of these three values. This is done by adjusting the generic variable COUNTER_WIDTH.

Address bits are shifted out serially on the spi_d pin. This address is defined by the variable START_ADDRESS generic. The ADDRESS_LENGTH generic variable should be specified by the user to match the number of bits in the starting address. Once the starting address is shifted out, dummy bits are shifted out and are zeros in this reference design. The user needs to specify the number of dummy bits in the DUMMY_LENGTH generic variable.

During this process, the DIN pin of the FPGA continues to receive 1's as they are applied via the fpga_din output.

When the counter reaches terminal count, which again is defined as the sum of instruction bits, address bits and dummy bits, the state machine transitions to the READ_DATA state.

READ_DATA State

Once the SPI Memory is setup with the instruction, starting address, and any necessary dummy bits, data appears on the SPI Flash's q pin. At this point the state machine is in the READ_DATA state which simply applies a Low value on spi_sn and passes the data from the memory straight through the CPLD to the FPGA DIN pin via fpga_din. The pin spi_sn is held Low to continue receiving data from the memory.

When the FPGA's DONE pin goes High indicating the FPGA has completed receiving needed configuration bits, the state machine transitions to the WAIT_STATE state.

WAIT_STATE State

The controller waits in this state indefinitely until the FPGA's INIT and DONE pins go Low as sensed on the fpga_init and fpga_done pins respectively. This indicates a reconfiguration request by the FPGA, and the state machine moves to the RESET state to begin the cycle again.

CoolRunner-II Implementation

This reference design has targeted the CoolRunner-II XC2C32-4-VQ44 device. Using this device, 26 macrocells are used, which corresponds to 81% utilization. This utilization level allows for minor changes to the design such as adding support for multiple SPI Memory devices without exceeding the device resource limits.

Other Xilinx CPLDs may be used in lieu of the XC2C32-4-VQ44 device, allowing for a flexible selection of system voltage ranges and feature choices.

VHDL Code and Software Download

VHDL Code Disclaimer

This application note supports the Numonyx-series SPI Flash memories and directly compatible versions from NexFlash and PMC. With small modifications, this same CPLD design likely supports SPI Flash memories from Atmel Corporation and Silicon Storage Technology, Inc.

All VHDL source code, VHDL testbenches, and software files associated with this design are available. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design should be used only as an example design, not as a fully functional core. XILINX does not warrant the performance, functionality, or operation of this Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the Design in terms of correctness, accuracy, reliability or otherwise.

THIRD PARTIES INCLUDING MOTOROLA® MAY HAVE PATENTS ON THE SERIAL PERIPHERAL INTERFACE (SPI) BUS. BY PROVIDING THIS HDL CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THE SPI BUS IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD

PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGNS "AS IS" AS A COURTESY TO YOU.

Xilinx Software License

BY USING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS AND CONDITIONS OF THIS LICENSE. CLICKING THE APPLICATION NOTE LINK BELOW AND DOWNLOADING THE ZIP FILES INDICATES THAT YOU ACCEPT THE TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, DO NOT USE THE SOFTWARE. IF YOU HAVE ALREADY PURCHASED THE SOFTWARE, PROMPTLY RETURN THE SOFTWARE TO THE PLACE WHERE YOU OBTAINED IT AND YOUR MONEY WILL BE REFUNDED.

1. License. XILINX, INC. ("XILINX") hereby grants you a nonexclusive license to use the software included on this website, disk, diskette, tape or CD ROM, and related documentation (collectively the "Software"), solely for your use in developing designs for XILINX Programmable Logic devices, or to copy the Software for backup or archival purposes, or for distribution as part of your company's program product(s) which is sold to customers. No right is granted hereunder to use the Software, or its products, to develop designs for third party Programmable Logic devices. XILINX and its licensors own and retain title to the Software and to any patents, copyrights, trade secrets and other intellectual property rights therein and all copies you make of it. Except as expressly provided herein, no right, title or other interest in or to the Software is transferred to you.
2. Restrictions. The Software contains copyrighted material, trade secrets, and other proprietary information. In order to protect them you may not decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human-perceivable form. You agree not for any purpose to transmit the Software or display the Software's object code on any computer screen or to make any hard copy memory dumps of the Software's object code. You may not modify or prepare derivative works of the Software in whole or in part. You may not publish or disclose the results of any benchmarking of the Software, or use such results for your own competing software development activities, without the prior written permission of Xilinx. You may not make any copies of the Software, except to the extent necessary to be used on separate non-simultaneous computers as permitted herein, and one copy of the Software in machine-readable form for backup purposes only. You must reproduce on each copy of the Software the copyright and any other proprietary legends that were on the original copy of the Software.

   This Software may be time-based to permit you to evaluate certain products only for a predetermined time. After the predetermined time expires, the Software will be automatically disabled. Xilinx requests that you inform users that the Software will time-out after a predetermined time and that a full license can be obtained thereafter. XILINX MAKES NO REPRESENTATION OR WARRANTY THAT TIME-BASED ROUTINE CANNOT BE DISABLED; AND YOU ACKNOWLEDGE THAT YOU WILL BEAR ALL RISK AND LIABILITY IF THE TIME-BASED ROUTINE IS UNLAWFULLY DISABLED TO ALLOW THE UNLICENSED USE OF THE SOFTWARE.
3. LIMITED WARRANTY AND DISCLAIMER. SUBJECT TO APPLICABLE LAWS: (1) XILINX'S AND ITS LICENSORS' ENTIRE LIABILITY TO YOU AND YOUR EXCLUSIVE REMEDY UNDER THIS WARRANTY WILL BE FOR XILINX, AT ITS OPTION, AFTER RETURN OF THE DEFECTIVE SOFTWARE MEDIA, TO EITHER REPLACE SUCH MEDIA OR TO REFUND THE PURCHASE PRICE PAID THEREFOR AND TERMINATE THIS LICENSE; (2) EXCEPT FOR THE ABOVE EXPRESS LIMITED WARRANTY, THE SOFTWARE IS PROVIDED TO YOU AS IS; (3) XILINX AND ITS LICENSORS MAKE AND YOU RECEIVE NO OTHER WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX AND ITS LICENSORS SPECIFICALLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. XILINX DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR

REQUIREMENTS, OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT DEFECTS IN THE SOFTWARE WILL BE CORRECTED. FURTHERMORE, XILINX DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY OR OTHERWISE. XILINX CANNOT PROVIDE TECHNICAL SUPPORT FOR THE SOFTWARE AND WILL NOT BE RESPONSIBLE FOR ANY CONSEQUENCES OF THE USE OF THE SOFTWARE.

4. LIMITATION OF LIABILITY. SUBJECT TO APPLICABLE LAWS: (1) IN NO EVENT WILL XILINX OR ITS LICENSORS BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES ARISING FROM THE USE OR OPERATION OF THE SOFTWARE OR ACCOMPANYING DOCUMENTATION, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY; (2) THIS LIMITATION WILL APPLY EVEN IF XILINX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE; (3) THIS LIMITATION SHALL APPLY NOTWITHSTANDING THE FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDIES HEREIN; AND (4) XILINX'S AND ITS LICENSORS' LIMITATIONS OF LIABILITIES ARE NOT CUMULATIVE. THE LIMITATIONS OF REMEDIES AND DAMAGES IN THIS SOFTWARE LICENSE SHALL NOT APPLY TO PERSONAL INJURY (INCLUDING DEATH) TO ANY PERSON CAUSED BY XILINX'S NEGLIGENCE AND ARE SUBJECT TO THE PROVISION SET OUT BELOW UNDER THE HEADING GOVERNING LAW.
5. Term & Termination. You may terminate this License at any time by destroying the Software and all copies thereof. This License will terminate immediately without notice from Xilinx if you fail to comply with any provision of this License. Upon termination you must destroy the Software and all copies thereof.
6. Governmental Use. The Software is commercial computer software developed exclusively at Xilinx's expense. Accordingly, pursuant to the Federal Acquisition Regulations (FAR) Section 12.212 and Defense FAR Supplement Section 227.2702, use, duplication and disclosure of the Software by or for the Government is subject to the restrictions set forth in this License. Manufacturer is XILINX, INC., 2100 Logic Drive, San Jose, California 95124.
7. Export Restriction. You agree that you will not export or reexport the Software, reference images or accompanying documentation in any form without the appropriate government licenses. Your failure to comply with this provision is a material breach of this License.
8. Third Party Beneficiary. You understand that portions of the Software and related documentation have been licensed to XILINX from third parties and that such third parties are intended third party beneficiaries of the provisions of this License, including without limitation the limitation of liabilities set forth herein. Xilinx has assumed responsibility for the selection of all materials licensed from third parties, and the use of such materials as licensed hereunder.
9. Interoperability. If you acquired the Software in the European Union (EU), even if you believe you require information related to the interoperability of the Software with other programs, you shall not decompile or disassemble the Software to obtain such information, and you agree to request such information from Xilinx at the address listed above. Upon receiving such a request, Xilinx shall determine whether you require such information for a legitimate purpose and, if so, Xilinx will provide such information to you within a reasonable time and on reasonable conditions.
10. Governing Law. This License shall be governed by the laws of the State of California, without reference to conflict of laws principles, provided that if the Software is acquired in the EU, this License shall be governed by the laws of the Republic of Ireland. The local language version of this License shall apply to Software acquired in the EU. Irish law provides that certain conditions and warranties may be implied in contracts for the sale of

goods and in contracts for the supply of services. Such conditions and warranties are hereby excluded, to the extent such exclusion, in the context of this transaction, is lawful under Irish law. Conversely, such conditions and warranties, insofar as they may not be lawfully excluded, shall apply. Accordingly nothing in this License shall prejudice any rights that you may enjoy by virtue of Sections 12, 13, 14 or 15 of the Irish Sale of Goods Act 1893 (as amended). Nothing in this Agreement will be interpreted or construed so as to limit or exclude the rights or obligations of either party (if any) which it is unlawful to limit or exclude under the relevant national laws and, where applicable, the laws of any Member State of the EU which implement relevant European Communities Council Directives. Nothing in this Agreement will be interpreted or construed so as to limit or exclude the rights or obligations of either party (if any) which it is unlawful to limit or exclude under the relevant national laws and, where applicable, the laws of any Member State of the EU which implement relevant European Communities Council Directives.

11. General. If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License shall be enforced to the maximum extent permissible to effect the intent of the parties, and the remainder of this License shall continue in full force and effect. This License constitutes the entire agreement between the parties with respect to the use of this Software and related documentation, and supersedes all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter.

XAPP800 -**Design Files**

Numonyx Simulation Models

Included in the above reference design are the simulation models for the Numonyx SPI Flash Memories. To obtain the latest simulation models as well as any assistance regarding these memories, please visit the Numonyx website at:

http://www.numonyx.com

Conclusion

The CoolRunner-II CPLD is an excellent choice for configuring Xilinx FPGAs using SPI Flash memories since these CPLDs are low cost. Together with the low cost SPI Flash memory and the CoolRunner-II CPLD, overall configuration bit storage costs are reduced over conventional methods allowing for a more cost effective solution. Further, the low power characteristics of the CoolRunner-II CPLD provide a minimal impact to the power consumption of the total configuration bit storage solution.

Additional Information

CoolRunner-II Data Sheets, Application Notes, and White Papers

Device Packages

Online Store

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/20/04	1.0	Initial Xilinx release.
04/24/08	1.1	Updated for new software utilities.
05/07/08	1.1.1	Fixed link to design file.