Show *Core-generator-software system Xilinx* video

Show *how-to-configure-an-fpga Xilinx* video

So now the question becomes this:

1. **how do I program those FLASH memories that contain configuration bitstreams.**
2. **More importantly, how do I change the configuration when everything is soldered on a PCB?**

The answer is JTAG.

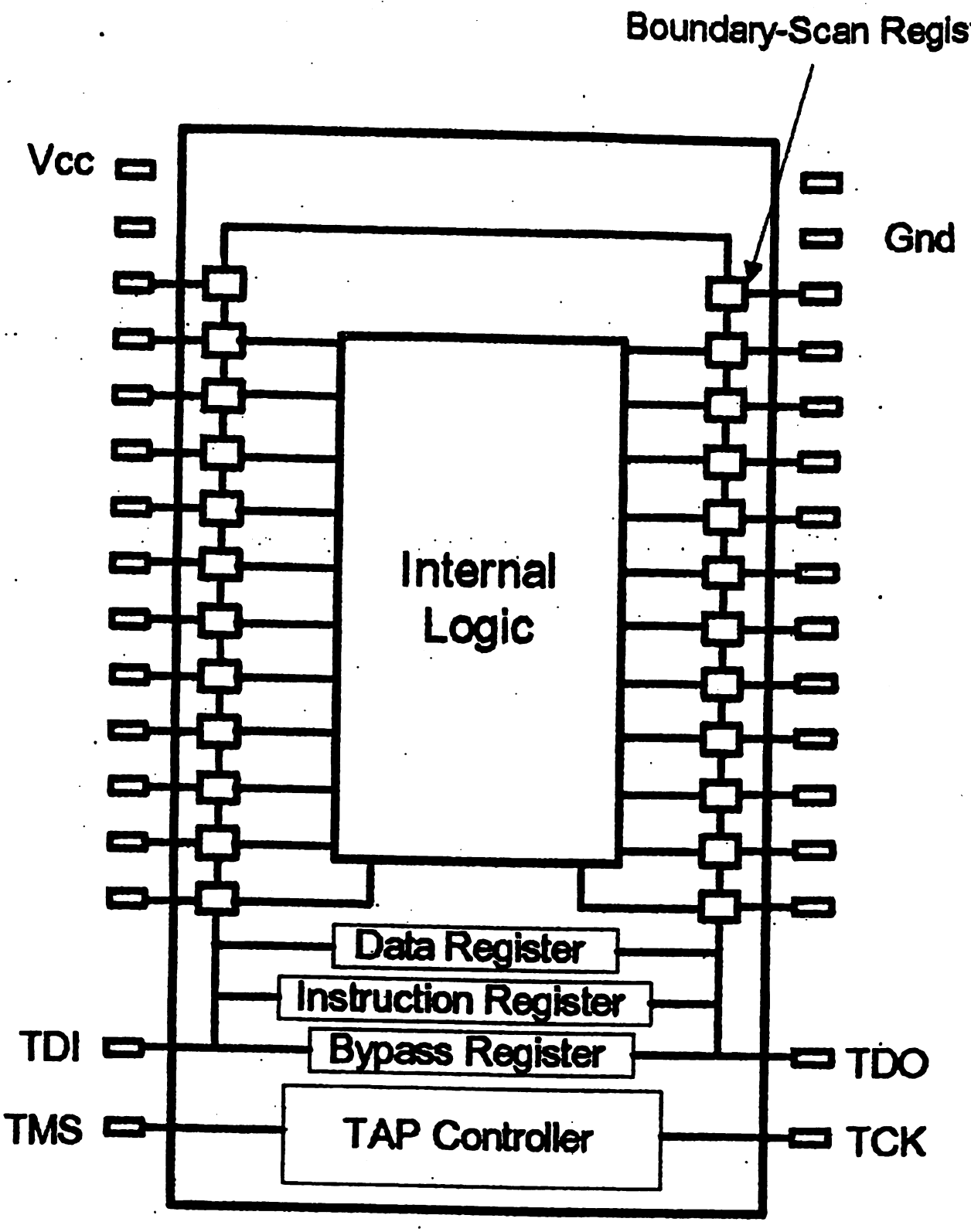
JTAG Basics

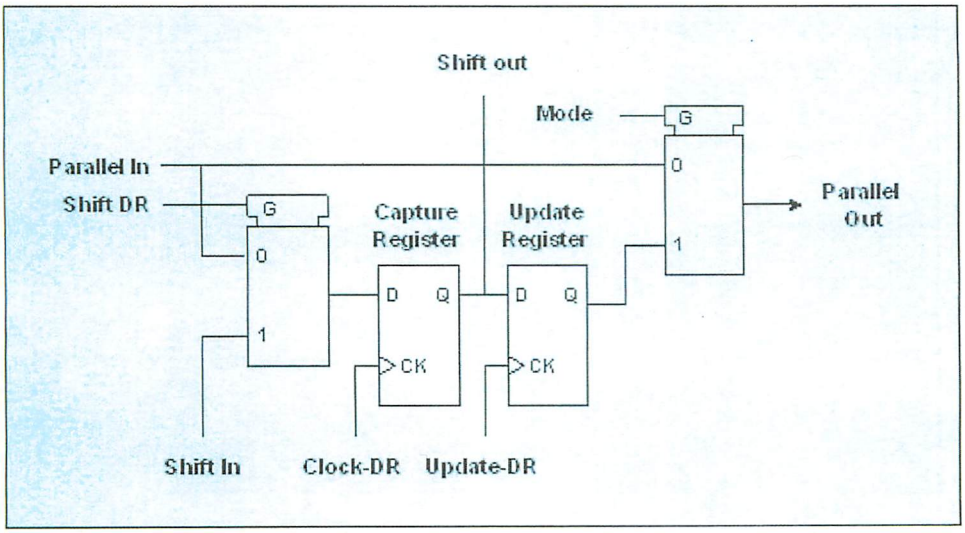
Joint test action group

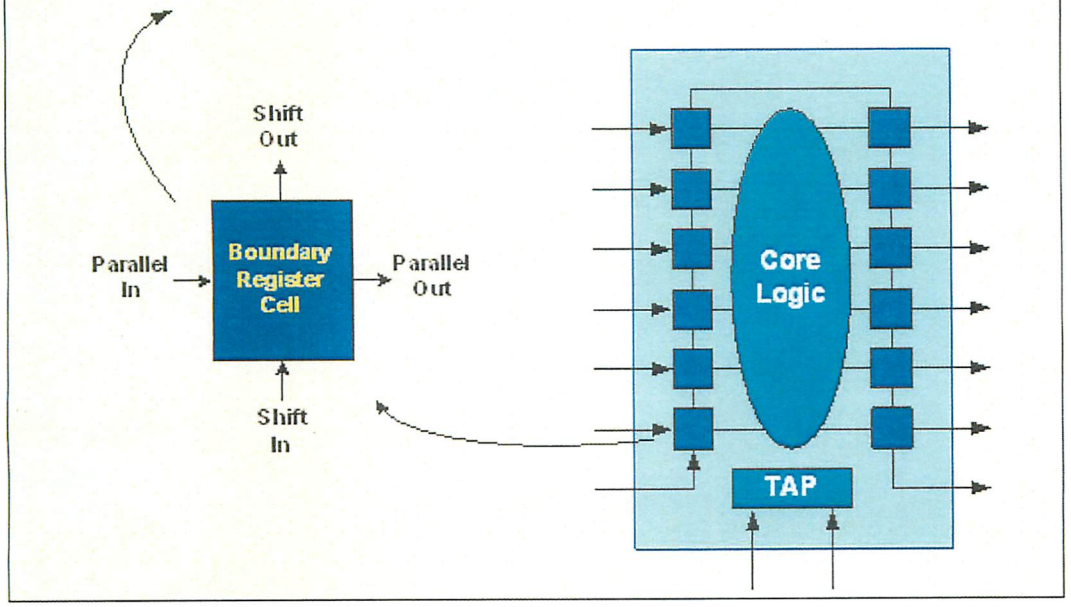
IEEE std 1149.1
"Test access port and boundary scan architecture"

defines Hwde/SFWE needed for boundary scan testing

Devices that meet JTAG Std are called
JTAG compliant


Boundary-Scan Register
Vcc
Gnd
Internal Logic
Data Register
Instruction Register
Bypass Register
TDI
TDO
TMS
TAP Controller
TCK


Shift out
Mode
G
Parallel In
Shift DR
G
0
1
Capture Register
Update Register
D
Q
CK
D
Q
CK
0
1
Parallel Out
Shift In
Clock-DR
Update-DR


Shift Out
Parallel In
Boundary Register Cell
Parallel Out
Shift In
Core Logic
TAP

XILINX®

XAPP139 (v1.7) February 14, 2007

Configuration and Readback of Virtex FPGAs Using JTAG Boundary-Scan

Summary

This application note demonstrates using a Boundary-Scan (JTAG) interface to configure and read back Virtex™ FPGA devices. Virtex devices have Boundary-Scan features that are compatible with IEEE Standard 1149.1. This application note is a complement to the configuration section in the Virtex data sheet and application note **XAPP138 "Virtex FPGA Series Configuration and Readback."** Xilinx recommends reviewing both the data sheet and XAPP138 prior to reading this document.

Note: The information in this application note also applies to the Virtex-E FPGA family.

Introduction

The IEEE 1149.1 Test Access Port (TAP) and Boundary-Scan architecture, commonly referred to as JTAG, is a popular testing method. JTAG is an acronym for the Joint Test Action Group, the technical subcommittee initially responsible for developing the standard. This standard provides a means to ensure the integrity of individual board-level components and their interconnections. With increasingly dense multi-layer PC boards and more sophisticated surface mounting techniques, Boundary-Scan testing is becoming widely used as an important debugging standard.

Devices containing Boundary-Scan logic can send data out on I/O pins in order to test connections between devices at the board level. The circuitry can also be used to send signals internally to test the device specific behavior. These tests are commonly used to detect opens and shorts at both the board and device level.

In addition to testing, Boundary-Scan offers the flexibility for a device to have its own set of user-defined instructions. The added common vendor-specific instructions, such as configure and verify, have increased the popularity of Boundary-Scan testing and functionality.

Boundary-Scan for Virtex Devices

The Virtex family is fully compliant with the IEEE Standard 1149.1 Test Access Port and Boundary-Scan architecture. The architecture includes all mandatory elements defined in the IEEE 1149.1 Standard. These elements include the TAP, the TAP controller, the instruction register, the instruction decoder, the Boundary-Scan register, and the bypass register. The Virtex family also supports some optional instructions – the 32-bit identification register and a configuration register in full compliance with the standard. Outlined in the following sections are the details of the JTAG architecture for Virtex devices.

Test Access Port

The Virtex TAP contains four mandatory dedicated pins as specified by the protocol (Table 1).

Table 1: **Virtex TAP Controller Pins**

Pin	Description
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
TCK	Test Clock

Three input pins and one output pin control the IEEE 1149.1 Boundary-Scan TAP controller. In addition to the required pins, there are optional control pins such as $\overline{\text{TRST}}$ (Test Reset) and enable pins, which can be found on devices from other manufacturers. Be aware of these optional signals when interfacing Xilinx devices with devices from different vendors because these signals can need to be driven. (To determine the set of signals that must be driven to enable IEEE 1149.1 compliance, see the vendor documentation for each device on the Boundary-Scan chain.)

The TAP controller is a 16-state state machine (Figure 1). Mandatory TAP pins are as follows:

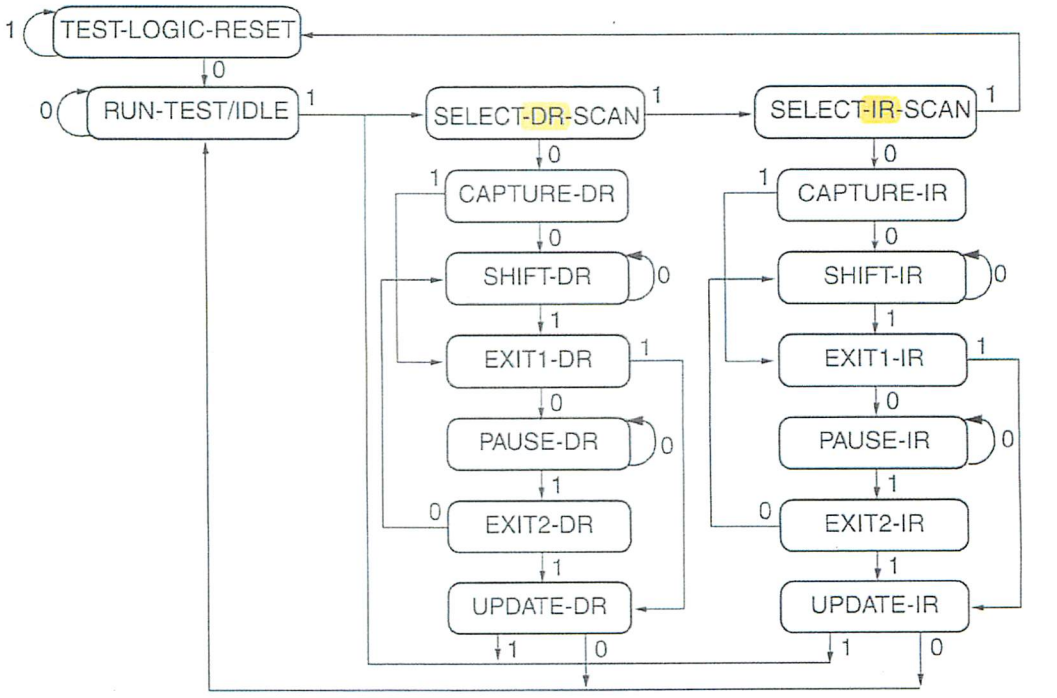
- TMS - The sequence of states through the TAP controller is determined by the state of the TMS pin on the rising edge of TCK. TMS has an internal resistive pull-up to provide a logic High if the pin is not driven.
- TCK - This pin is the JTAG test clock. It sequences the TAP controller and the JTAG registers in the Virtex devices.
- TDI - This pin is the serial input to all JTAG instruction and data registers. The state of the TAP controller and the current instruction held in the instruction register determine which register is fed by the TDI pin for a specific operation. TDI has an internal resistive pull-up to provide a logic High to the system if the pin is not driven. TDI is applied into the JTAG registers on the rising edge of TCK.
- TDO - This pin is the serial output for all JTAG instruction and data registers. The state of the TAP controller and the current instruction held in the instruction register determine which register (instruction or data) feeds TDO for a specific operation. TDO changes state on the falling edge of TCK and is active only during the shifting of instructions or data through the device. This pin is placed in a 3-state condition at all other times.

Note: As specified by the IEEE Standard, the TMS and TDI pins all have internal pull-ups. These internal pull-ups of 50-150 kΩ are active regardless of the mode selection.

When using the Boundary-Scan operations in Virtex devices, the V_{CCO} for Bank 2 must be at 3.3V for the TDO pin to operate at the required LVTTL level.

TAP Controller

Figure 1 diagrams a 16-state finite state machine. The four TAP pins control how the data is scanned into the various registers. The state of the TMS pin at the rising edge of the TCK determines the sequence of state transitions. There are two main sequences, one for shifting data into the data register and the other for shifting an instruction into the instruction register.



Note: The value shown adjacent to each state transition in this figure represents the signal present at TMS at the time of a rising edge at TCK.

x139_01_011207

Figure 1: **State Diagram for the TAP Controller**

Boundary-Scan Instruction Set

To determine the operation to be invoked, a 5-bit instruction is loaded into the instruction register. Table 2 lists the available Boundary-Scan instructions for Virtex devices.

Table 2: **Virtex Boundary-Scan Instructions**

Boundary-Scan Command	Binary Code (4:0)	Description
EXTEST	00000	Enables Boundary-Scan EXTEST operation
SAMPLE	00001	Enables Boundary-Scan SAMPLE operation
USER1	00010	Access user-defined register 1
USER2	00011	Access user-defined register 2
CFG_OUT	00100	Access the configuration bus for readback
CFG_IN	00101	Access the configuration bus for configuration
INTEST	00111	Enables Boundary-Scan INTEST operation

mandatory instr

Table 2: **Virtex Boundary-Scan Instructions *(Continued)***

Boundary-Scan Command	Binary Code (4:0)	Description
USERCODE	01000	Enables shifting out user code
IDCODE	01001	Enables shifting out of ID code
HIGHZ	01010	Places output pins in a 3-states condition while enabling the bypass register
JSTART	01100	Clocks the start-up sequence when StartClk is TCK
BYPASS	11111	Enables BYPASS
RESERVED	All other codes	Xilinx reserved instructions

mandatory instr

The mandatory IEEE 1149.1 commands are supported in Virtex devices along with several Xilinx vendor-specific commands. Virtex devices have a powerful command set. The EXTEST, INTEST, SAMPLE/PRELOAD, BYPASS, IDCODE, USERCODE, and HIGHZ instructions are all included. The TAP also supports two internal user-defined registers (USER1 and USER2) and configuration/readback of the device. The Virtex Boundary-Scan operations are independent of the mode selection. The Boundary-Scan mode in Virtex devices overrides the other mode selections. For this reason, Boundary-Scan instructions using the Boundary-Scan register (SAMPLE/PRELOAD, INTEST, EXTEST) must not be performed during configuration. All instructions except USER1 and USER2 are available before the Virtex device is configured. After configuration, all instructions are available.

JSTART is an instruction specific to the Virtex architecture and configuration flow. As described in Table 2, the JSTART instruction clocks the startup sequence when the appropriate bitgen option is selected. The instruction does not work correctly without the correct bitgen option selected.

```
bitgen -g startupclk:jtagclk designName.ncd
```

For details on the standard Boundary-Scan instructions, EXTEST, INTEST, and BYPASS, refer to the IEEE Standard. The user-defined registers (USER1/USER2) are described in a later section of this application note.

Boundary-Scan Architecture

Virtex devices have several registers including all registers required by the IEEE 1149.1. In addition to the standard registers, the family contains optional registers for simplified testing and verification (Table 3).

Table 3: **Virtex JTAG Registers**

Register Name	Register Length	Description
Instruction register	5 bits	Holds current instruction OPCODE and captures internal device status
Boundary-Scan register	3 bits per I/O	Controls and observes input, output, and output enable
Bypass register	1 bit	Device bypass
Identification register	32 bits	Captures device ID
JTAG configuration register	32 bits	Allows access to the configuration bus when using the CFG_IN or CFG_OUT instructions
USERCODE register	32 bits	Captures user-programmable code

Single Device Configuration

Note: Refer to XAPP058 for the recommended embedded solution.

To configure a Virtex part as a single device through Boundary-Scan operations, use the steps listed in Table 8, which lists and describes the TAP controller commands required to configure a Virtex device. Ensure the bitstream is generated with the JTAG clock option:

```
bitgen -g startupclk:jtagclk designName.ncd
```

Also, when programming with iMPACT software, verify that the most current version of software is used. Refer to Figure 1 for the TAP controller states. These TAP controller commands are issued automatically if configuring the part with the iMPACT software.

Table 8: **Single Device Configuration Sequence**

	TAP Controller Step Description	Set and Hold		Number of Clocks
		TDI	TMS	TCK
1	On power-up, place a "1" on the TMS and clock the TCK five times. (This ensures starting in the TLR (Test-Logic-Reset) state.)	X	1	5
2	Move into the RTI state.	X	0	1
3	Move into the SELECT-IR state.	X	1	2
4	Enter the SHIFT-IR state.	X	0	2
5	Start loading the CFG_IN instruction .(1)	0101	0	4
6	Load the last bit of CFG_IN instruction when exiting SHIFT-IR (defined in the IEEE standard).	0	1	1
7	Enter the SELECT-DR state.	X	1	2
8	Enter the SHIFT-DR state.	X	0	2
9	Shift in the Virtex bitstream. (bit_N (MSB) is the first bit in the bitstream (1))	bit_1... bit_N	0	(Number of bits in bitstream) -1
10	Shift in the last bit of the bitstream. (bit_0 (LSB) is shifted on the transition to EXIT1-DR)	bit_0	1	1
11	Enter UPDATE-DR state.	X	1	1
12	Enter the SELECT-IR state.	X	1	2
13	Move to the SHIFT-IR state.	X	0	2
14	Start loading the JSTART instruction.(1) (The JSTART instruction initializes the startup sequence.)	1100	0	4
15	Load the last bit of the JSTART instruction.	0	1	1
16	Move to the SELECT-DR state.	X	1	2
17	Move to SHIFT-DR and clock the STARTUP sequence. (by applying a minimum of 12 clock cycles to the TCK).	X	0	≥14
18	Move to the UPDATE-DR state.	X	1	2
19	Return to the RTI state. (The device is now functional).	X	0	1

Note:

1. In the TDI column, the right-most bit is shifted in first.

ispDOWNLOAD® Cables

For Programming Lattice ISP™ Devices

November 2007

Features

- **Support for All Lattice Programmable Products**
  - 1.2V to 5V programming
  - Ideal for design prototyping and debugging
- **Connect to Multiple PC Interfaces**
  - USB (v.1.0, v.2.0)
  - PC Parallel Port
- **Easy-to-Use Programming Connectors**
  - Versatile flywire, 2 x 5 (.100") or 1 x 8 (.100") connectors
  - 6 feet (2 meters) or more of programming cable length (PC to DUT)
- **Lead-Free/RoHS Compliant Construction**

ispDOWNLOAD Cables

Lattice ispDOWNLOAD cables are designed to facilitate in-system programming for all Lattice Semiconductor ISP™ devices directly from a PC. With in-system programmability, hardware functions can be programmed and modified in real-time on the system board to give additional product features, shorten system design and debug cycle time, enhance product manufacturability and simplify field upgrades.

After you complete your logic design and create a programming file with the ispLEVER® development tools, you can use ispVM® System software to program devices on your board. The ispVM System software automatically generates the appropriate ISP commands, programming addresses and programming data based on information stored in the programming file and parameters you set in ispVM. Programming signals are then generated from the USB or parallel port of a PC and directed through the ispDOWNLOAD Cable to the device, no additional components are required for programming.

ispVM System software is included with all Lattice design tool products and available for download from the Lattice web site at www.latticesemi.com.

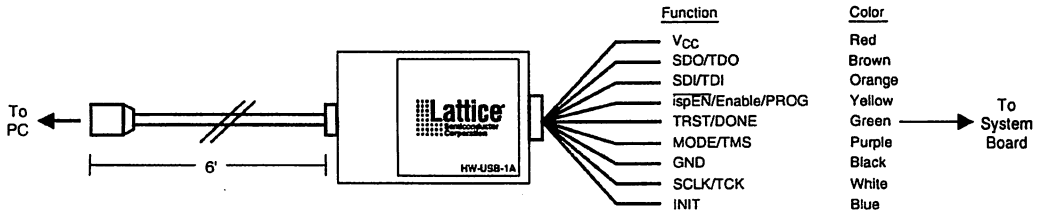
ispDOWNLOAD Cable Pin Definitions

The functions provided by of the ispDOWNLOAD cables correspond with available functions on Lattice programmable devices. Since some devices contain different programming features, the specific functions provided by the ispDOWNLOAD cable may depend on the selected target device. ispVM System software will automatically generate the appropriate functions based on the selected device. See Table 1 for an overview of the ispDOWNLOAD cable functions.

Table 1. ispDOWNLOAD Cable Pin Definitions

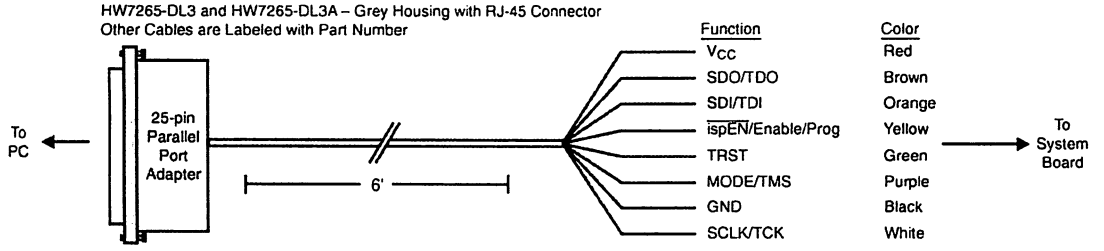
ispDOWNLOAD Cable Pin	Name	ispDOWNLOAD Cable Input/Output	Description
VCC	Programming Voltage	Input	Connect to V_{CC} or V_{CCJ} plane of the target device. Typical I_{CC} = 10mA. (Note: this may not be the same as a target device's V_{CCO} plane).
SDO/TDO	Test Data Output	Input	Used to shift data out via the IEEE1149.1 (JTAG) programming standard.
SDI/TDI	Test Data Input	Output	Used to shift data in via the IEEE1149.1 programming standard.
ispEN/Enable/ PROG	Enable	Output	Enable device to be programmed.
TRST	Test Reset	Output	Optional IEEE 1149.1 state machine reset.
DONE	DONE	Input	Done indicates status of configuration
MODE/TMS	Test Mode Select Input	Output	Used to control the IEEE1149.1 state machine.
GND	Ground	Input	Connect to ground plane of the target device
SCLK/TCK	Test Clock Input	Output	Used to clock the IEEE1149.1 state machine
INIT	Initialize	Input	Indicates that ORCA device is ready for configuration.

Figure 1. ispDOWNLOAD Cable In-System Programming Interface for the PC (HW-USB-1A or HW-USB-2A)[1]



1. Lattice PAC-Designer® software does not support programming with USB cables. To program ispPAC devices with these cables, use the ispVM System software.

Figure 2. ispDOWNLOAD Cable In-System Programming Interface for the PC[1]



1. HW7265-DL3, HW7265-DL3A, HW-DL-3B, HW-DL-3C and HW-DLN-3C are functionally equivalent products.

Figure 3. ispDOWNLOAD Cable In-System Programming Interface for the PC (pDS4102-DL2 or pDS4102-DL2A)

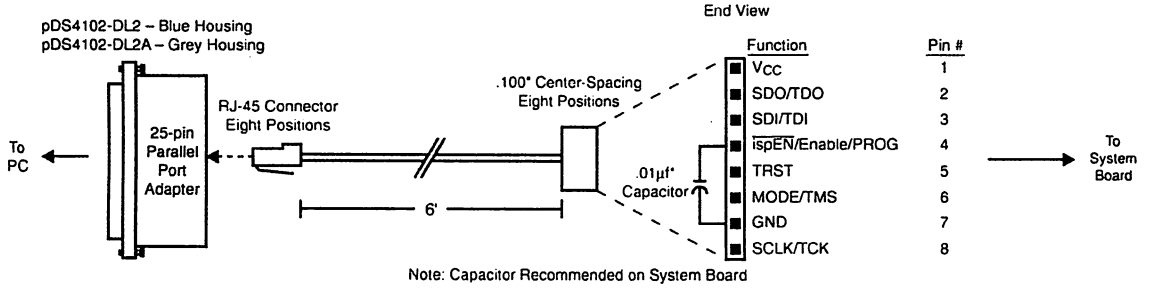
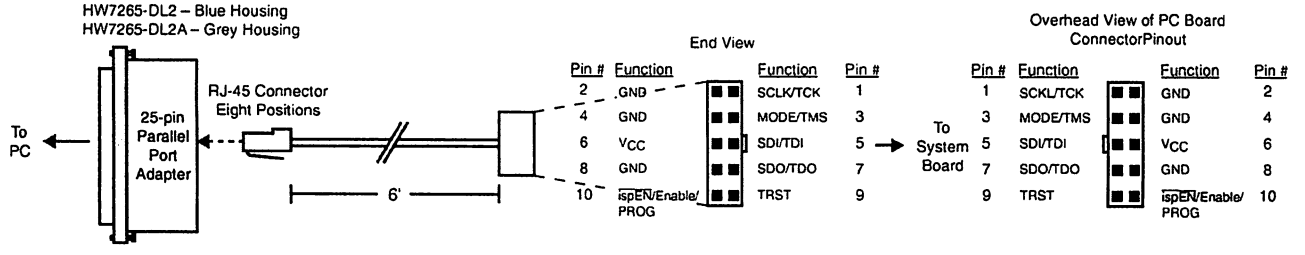


Figure 4. ispDOWNLOAD Cable In-System Programming Interface for the PC (HW7265-DL2 or HW7265-DL2A)



Programming Software

ispVM System is the preferred programming management software tool for all Lattice devices and download cables. The latest version of ispVM System is always available for download from the Lattice web site at www.latticesemi.com/software.

PAC-Designer is the design tool for Lattice ispPAC and ispCLOCK devices. PAC-Designer can also be used to program these devices. If using PAC-Designer for programming, pay special attention to the notes in this document, and the PAC-Designer system help.

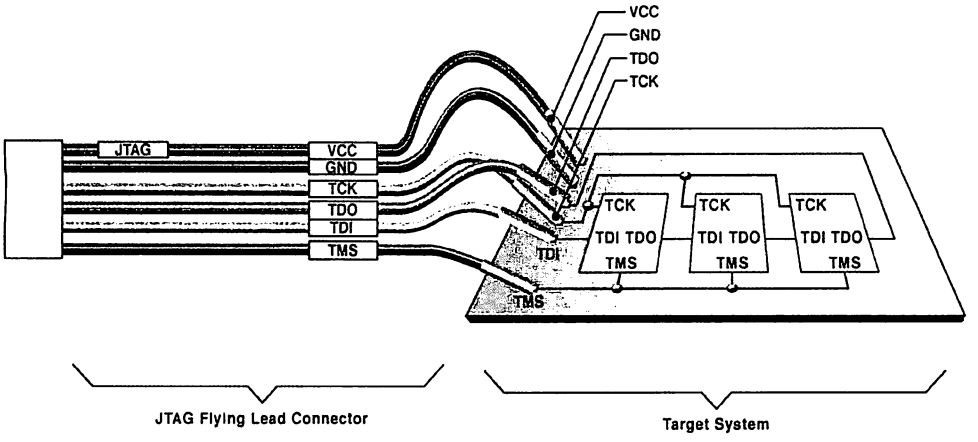


Figure 2-6 Parallel Download Cable Connection to JTAG Boundary-scan TAP

Appendix B contains schematic diagrams of the Parallel Download Cable.

Configuring the Parallel Download Cable

On PCs you can connect the parallel cable to your system's parallel printer port. The JTAG Programmer software will automatically identify the cable when correctly connected to your PC. If you choose to, you may also select this connection manually. To set up a parallel port manually:

`Output → Cable Setup`

Select the **`Parallel`** box and match to the port you are using, then click on **`OK`**.

Flying Lead Connectors

The flying lead connector has a 9-pin (6 signals, 3 keys) header connector that fits onto the cable's JTAG header. The pin order is listed in Table 2-3. These header connectors are keyed to assure

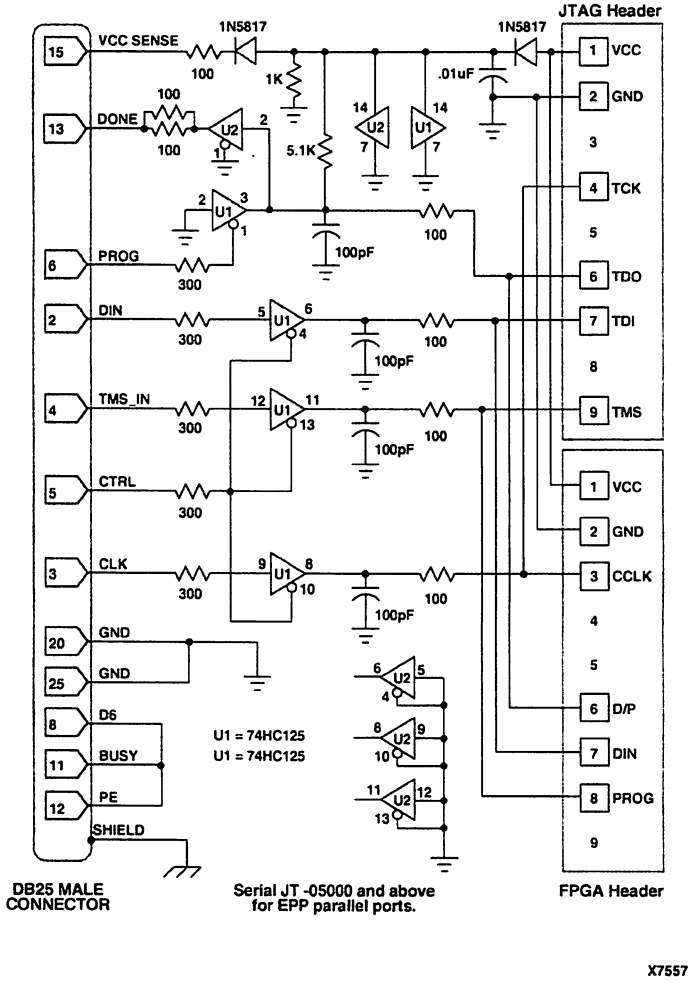


Figure B-1 Parallel Download Cable

Configuration Data File Formats

Xilinx design tools can generate configuration data files in a number of different formats, as described in Table 1-3. BitGen converts the post-PAR NCD file into a configuration file or a bitstream. PROMGen, the PROM file generator, converts one or more bitstream files into a PROM file. PROM files can be generated in a number of different file formats and does not need to be used with a PROM. They can be stored anywhere and delivered by any means.

Table 1-3: **Xilinx Configuration File Formats**

File Extension	Bit Swapping(1)	Xilinx Software Tool(2)	Description
BIT	Not Bit Swapped	BitGen (generated by default)	Binary configuration data file containing header information that does not need to be downloaded to the FPGA. Used to program devices from iMPACT with a programming cable.
RBT	Not Bit Swapped	BitGen (generated if `-b` option is set)	ASCII equivalent of the BIT file containing a text header and ASCII 1s and 0s. (Eight bits per configuration bit.)
BIN	• BitGen: Not Bit Swapped • PROMGen: Bit Swapped	BitGen (generated if `-g binary:yes` option is set) or PROMGen	Binary configuration data file with no header information. Similar to BIT file. Can be used for custom configuration solutions (for example, microprocessors), or in some cases to program third-party PROMs.
MCS EXO TEK	Bit Swapped	PROMGen or iMPACT	ASCII PROM file formats containing address and checksum information in addition to configuration data. Used mainly for device programmers and iMPACT.
HEX	Determined by User	PROMGen or iMPACT	ASCII PROM file format containing only configuration data. Used mainly in custom configuration solutions.

Notes:

1. Bit swapping is discussed in the "Bit Swapping" section.
2. For complete BitGen and PROMGen syntax, refer to the *Development System Reference Guide*.

Bitstream Overview

The Virtex-5 bitstream contains commands to the FPGA configuration logic as well as configuration data. Table 1-4 gives a typical bitstream length for each of the Virtex-5 devices.

Table 1-4: **Virtex-5 FPGA Bitstream Length**

Device	Total Number of Configuration Bits(1)
XC5VLX30	8,374,016
XC5VLX50	12,556,672
XC5VLX85	21,845,632
XC5VLX110	29,124,608
XC5VLX155	41,048,064
XC5VLX220	53,139,456
XC5VLX330	79,704,832

XAPP973 (v1.4) March 8, 2010

Indirect Programming of BPI PROMs with Virtex-5 FPGAs

Author: Stephanie Tapp

Summary

Virtex®-5 FPGAs and ISE® software support configuration from and programming of industry-standard, parallel NOR flash memory (BPI PROMs). Industry standard BPI PROMs are an alternate solution for Virtex-5 FPGA designs whose requirements are not met by the Platform Flash XL configuration and storage device (see [Ref 1] for more information on Platform Flash XL). The iMPACT software, included in the ISE® development software tools, provides indirect programming for select BPI PROMs during prototyping. This application note demonstrates how to program a Numonyx StrataFlash P30 BPI PROM indirectly using iMPACT 11.4 and a Xilinx cable. In this solution, the Virtex-5 FPGA serves as a bridge between the IEEE Std 1149.1 (JTAG) bus interface and the BPI bus interface. The required hardware setup, BPI-UP PROM file generation flow, and BPI indirect programming flow are shown. The Virtex-5 FPGA BPI-UP configuration sequence is also described.

Note: Parallel NOR flash memory is referred to by the term BPI PROM throughout this document.

Introduction

Xilinx FPGAs are CMOS configurable latch (CCL) based and must be configured at power-up from a non-volatile source. FPGA configuration is traditionally accomplished with a JTAG interface, a microprocessor, or the Xilinx PROMs (Platform Flash PROMs). In systems where the easiest solution is preferred, Master Serial mode with a Xilinx Platform Flash PROM is still the most popular configuration mode because it has:

- A direct JTAG interface for programming
- The smallest interface pin requirement for configuration
- Flexible I/O voltage support

Moreover, this solution is available for any Virtex-5 FPGA device (refer to [Ref 2] for more information).

In addition to the traditional methods, a direct configuration interface to third-party BPI PROMs is included on Virtex-5 FPGAs to address changing system requirements. Systems with a BPI PROM already onboard for random-access, non-volatile application data storage can benefit from consolidating the configuration storage into the same memory device.

Similar to the traditional configuration memories, BPI PROMs must be loaded with the configuration data. BPI PROMs have a single interface for programming, and three primary methods to deliver the data to this interface:

- Third-party programmers (off-board programming)
- In-system programming (ISP) with an embedded processor
- Indirect ISP (using JTAG or custom solution)

Production programming is often accomplished off-board with a third-party programmer or in-system with a JTAG tool vendor. During the prototyping phase, indirect ISP is preferred to easily accommodate design iterations.

Because BPI PROMs do not have a JTAG interface, extra logic is required to serve as a bridge between the iMPACT programmer (using a cable to drive the JTAG bus interface), and the BPI

PROM (connected to the FPGA's BPI bus interface). This extra logic must be downloaded into the FPGA by iMPACT before indirect programming is possible.

This application note is divided into three main sections. The first section discusses the hardware connections required for the indirect in-system programming of BPI PROMs for prototype designs. The second section shows the Xilinx software tool flows for generating a PROM file formatted for 16-bit BPI-UP mode and then for programming the select BPI PROMs. The third section provides a basic configuration flow overview for the FPGA after the BPI PROM is programmed and describes expectations when using this indirect setup.

iMPACT Indirect In-System Programming with a Virtex-5 FPGA

The basic hardware setup required for the iMPACT indirect BPI PROM programming method is shown in Figure 1.

Xilinx provides a proprietary IP module that must be downloaded into the FPGA. This IP converts JTAG instructions into the command/control signals needed to program the flash memory.
After downloading the configuration bitstream, the FPGA reads to flash to configure itself.

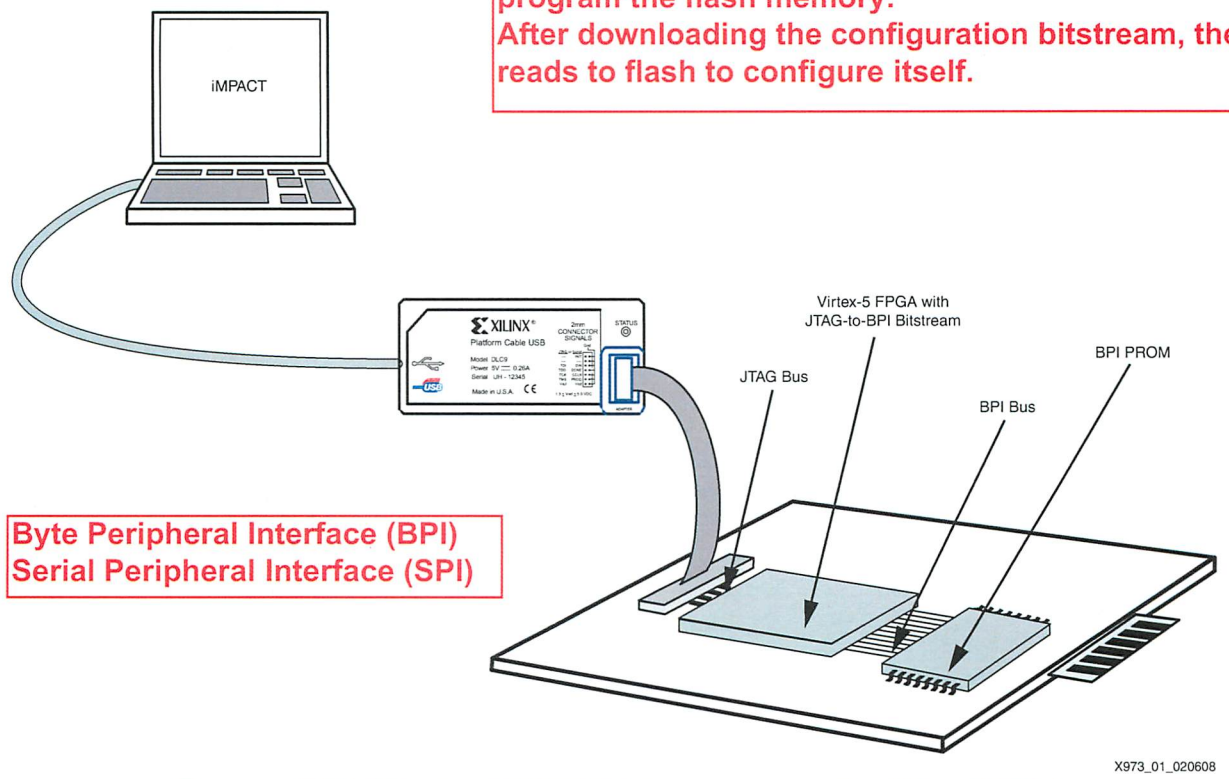


Figure 1: **iMPACT Indirect BPI PROM Programming with a Virtex-5 FPGA**

Minimum Requirements

- Virtex-5 FPGA
- BPI PROM (refer to Table 1)
- Xilinx Cable and Connector (refer to Table 4, page 7)
- ISE iMPACT Software 11.4

Note: Indirect BPI PROM programming was introduced with limited device support in iMPACT 9.2i. This application note demonstrates the software flow and lists the device support in iMPACT 11.4.

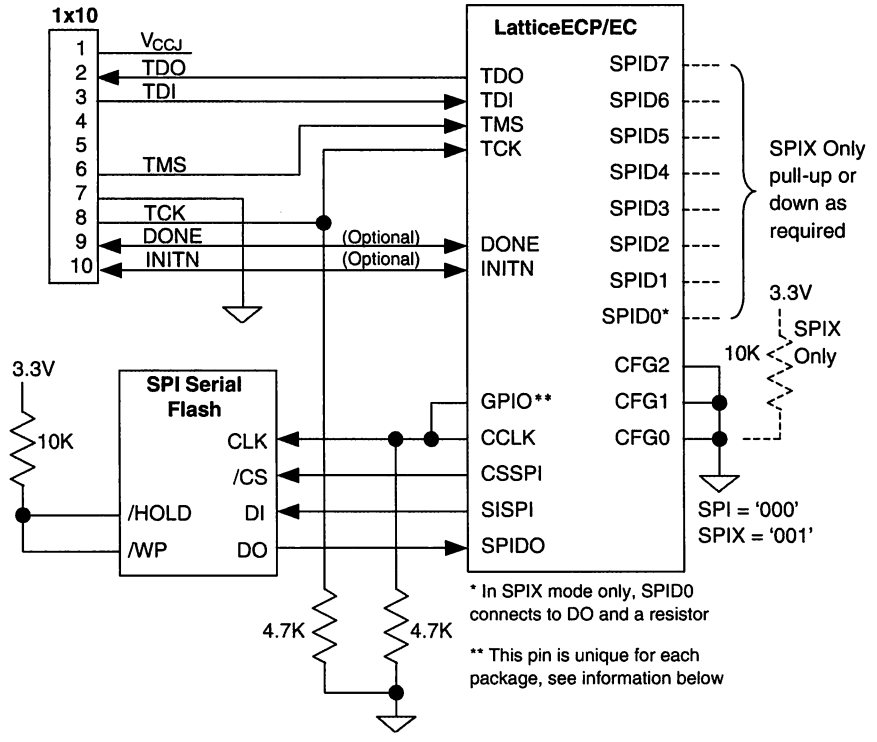
JTAG programming software, such as iMPACT by Xilinx, uses sequences of JTAG instructions to perform specific programming and verification operations selected by the user.

However, these software tools do *not* support execution of individual JTAG instructions. In other words, the user can't write custom sequences of JTAG instructions.

Schematic

The schematic in Figure 20-2 illustrates how to wire the ispJTAG connector, FPGA, and SPI Serial Flash.

Figure 20-2. Hardware Schematic



- The download header has standard 0.1 inch pin-to-pin spacing.
- The 4.7K pull-down resistors prevent spurious clock pulses during V_{CC} ramp-up. Place the resistors close to their clock line to keep the stub length as short as possible.
- The CCLK frequency can be as high as 50MHz, so keep this trace fairly short.
- V_{CCIO} for the bank that drives the signals to the SPI Serial Flash must match the SPI Serial Flash V_{CC} (today that voltage is 3.3V but this will change over time as Flash chip geometries decrease). For all packages these signals are located in bank 3.
- During configuration CCLK drives the SPI Serial Flash CLK pin, but once the FPGA completes configuration CCLK goes into tri-state. The CCLK pin is not accessible by user code so CCLK needs to be wired to a nearby General Purpose I/O pin (GPIO) to allow the FPGA fabric to supply a clock to the SPI Serial Flash. This pin is part of the Soft SPI Interface and is unique to each package. If the user embeds the Soft SPI Interface into their code then this pin, along with the other pins wired to the SPI Serial Flash, must be locked using the Pre-Map Preference Editor in ispLEVER. A complete list of these pins is found in Table 20-5.
- In addition to standard decoupling practices, place a decoupling capacitor close to the connector's V_{CCJ} pin. Any standard ceramic capacitor value may be used, for example 0.1 µF, 0.01 µF, etc.

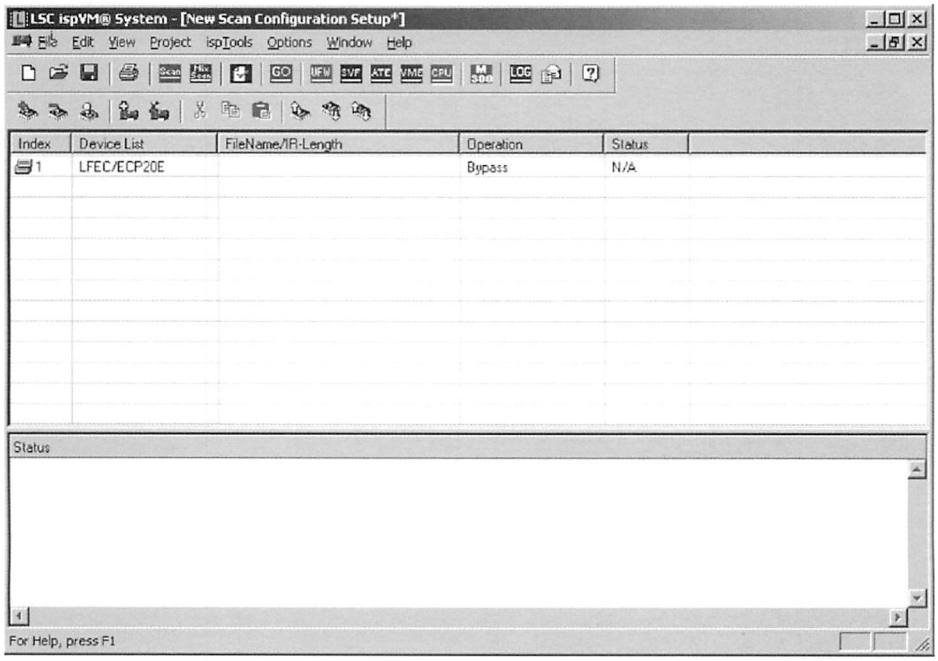
Programming Procedure

In order to program SPI Serial Flash via ispJTAG the FPGA must contain the Soft SPI Interface. Programming the SPI Serial Flash with ispVM System and an ispDOWNLOAD cable makes this transparent to the user. The software simply loads a default Soft SPI Interface bitstream into the FPGA and then loads the user bitstream into the Flash. Once programing of the SPI Serial Flash is complete the FPGA configures itself by reading the Flash. Again, software makes all of this transparent to the user so that it is no different than programming any other serial boot device.

The following instructions describe the process of selecting the FPGA, selecting the SPI Serial Flash, and programming the SPI Serial Flash. The following screen shots are from ispVM System 15.0.

1. Connect the ispDOWNLOAD Cable to the appropriate header and apply power to the board.
2. Start the ispVM System software.
3. From the main window click on the **Scan** button located on the toolbar. The LatticeECP/EC device should be detected automatically (if it's not detected, check the ispJTAG connections and make sure the board is powered up). The resulting screen should be similar to Figure 20-3.

Figure 20-3. Main Window, Scan Complete



4. Double-click the number in the **Index** column and select the appropriate device to open the Device Information window, shown in Figure 20-4.