****

How-to-configure-an-FPGA Xilinx video

Show summary slide nach

So now the question becomes this:

How do I program those flash memories—especially if they're soldered on a PCB?

Answer: JTAG

****

indirect programming of BPI PROMs with Virtex-5 application note

Defn (FPGA image)

a binary file that's the programming data for the FPGA

Note:

image = config. bitstream

= programming bitstream

= bitstream

Configuring FPGAs from SPI Serial Flash

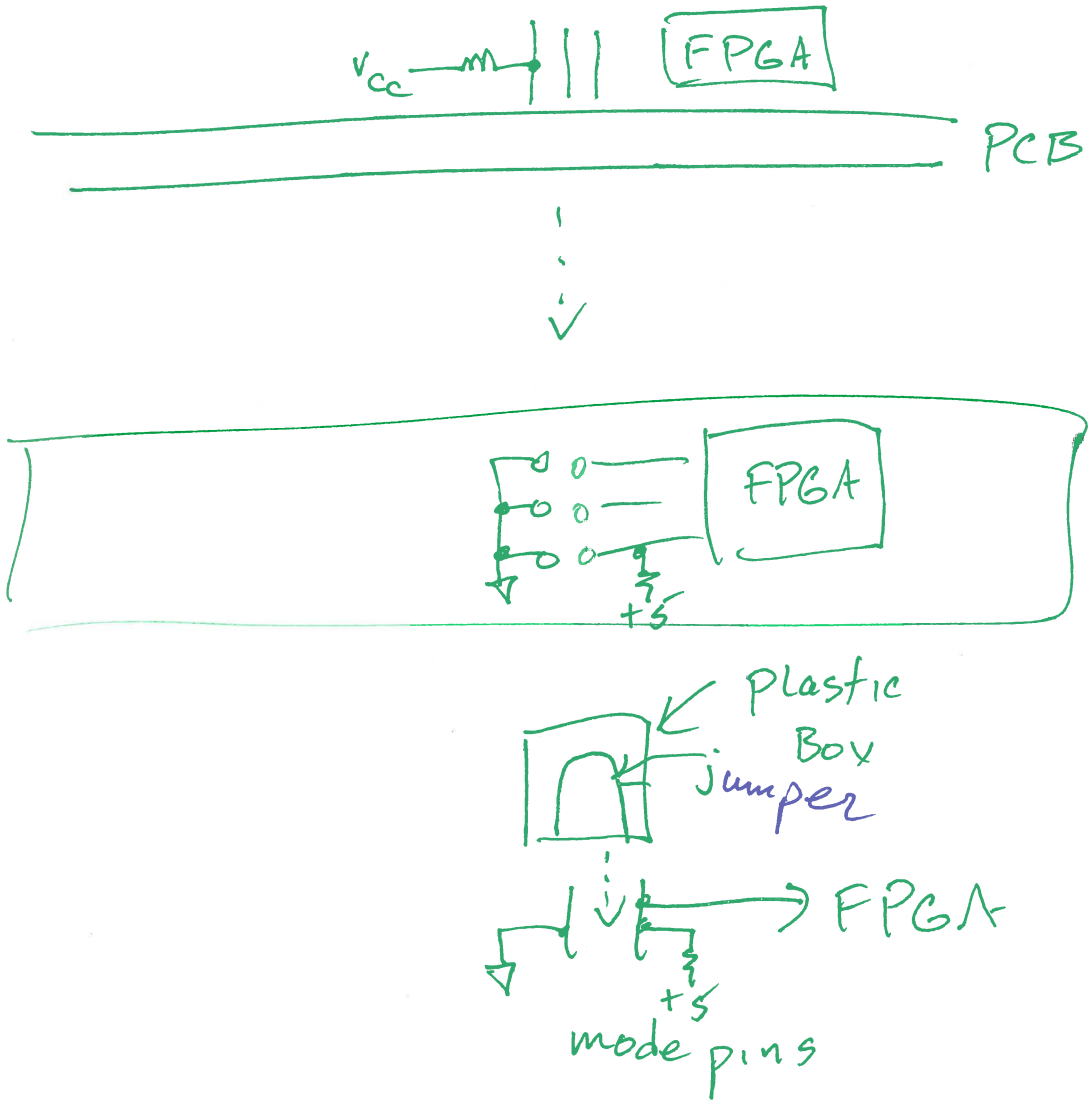
Spartan-3E and Virtex-5 FPGAs can be configured from a single SPI serial flash memory. The typical configuration density requirements for these FPGAs are provided in Table 2.

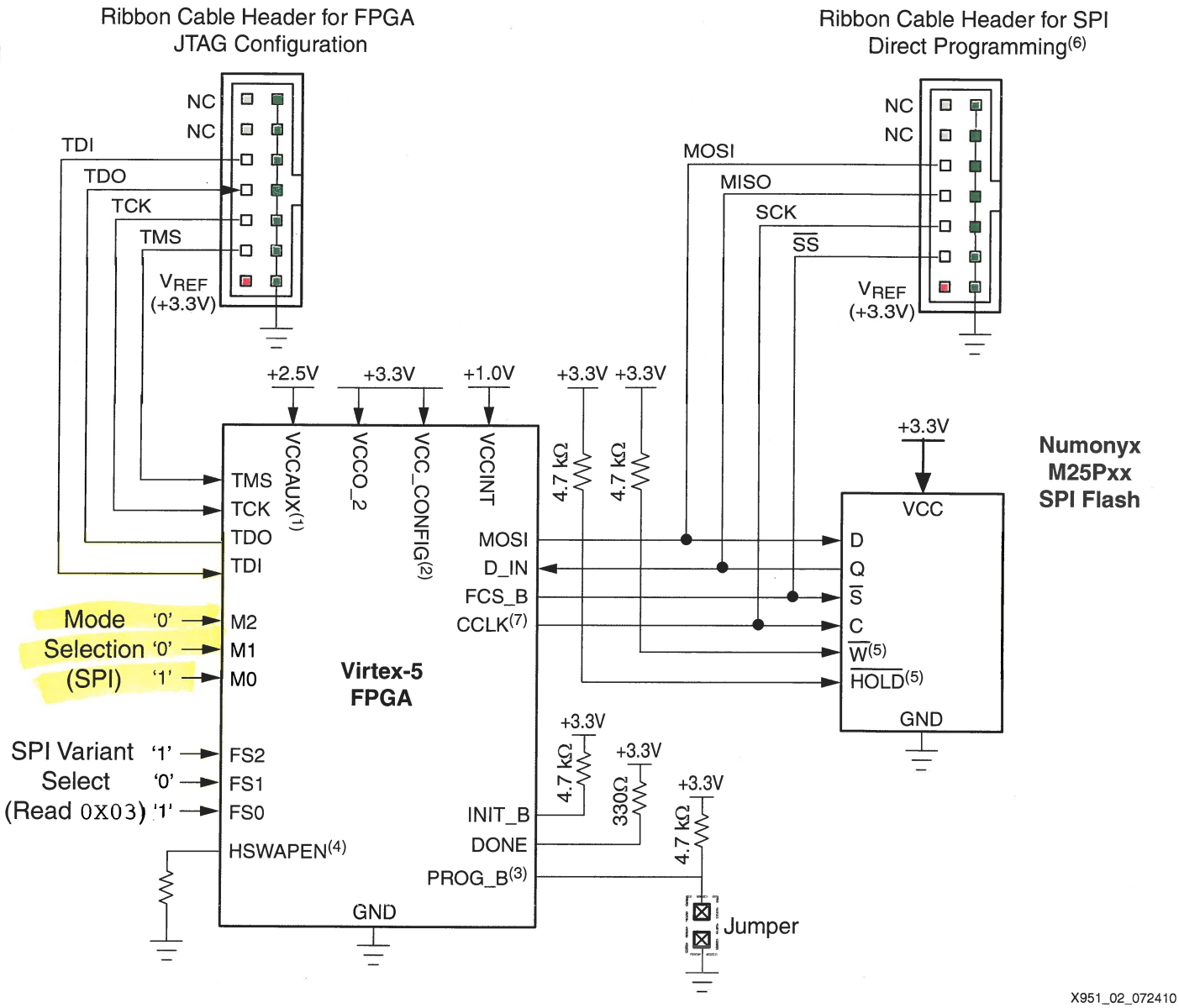
Table 2: **Typical Configuration Bit Requirements**

FPGA	Configuration Bits (Per Device)	Smallest SPI Serial Flash Required
Spartan-3E Family		
XC3S100E	581,344	1 Mb
XC3S250E	1,353,728	2 Mb
XC3S500E	2,270,208	4 Mb
XC3S1200E	3,837,184	4 Mb
XC3S1600E	5,969,696	8 Mb
Virtex-5 Family		
XC5VLX30	8,374,016	8 Mb
XC5VLX50	12,556,672	16 Mb
XC5VLX85	21,845,632	32 Mb
XC5VLX110	29,124,608	32 Mb
XC5VLX155	41,048,064	64 Mb
XC5VLX220	53,139,456	64 Mb
XC5VLX330	79,704,832	128 Mb
XC5VLX20T	6,251,200	8 Mb
XC5VLX30T	9,371,136	16 Mb
XC5VLX50T	14,052,352	16 Mb
XC5VLX85T	23,341,312	32 Mb
XC5VLX110T	31,118,848	32 Mb
XC5VLX155T	43,042,304	64 Mb
XC5VLX220T	55,133,696	64 Mb
XC5VLX330T	82,696,192	128 Mb
XC5VSX35T	13,349,120	16 Mb
XC5VSX50T	20,019,328	32 Mb
XC5VSX95T	35,716,096	64 Mb
XC5VSX240T	79,610,368	128 Mb
XC5VFX30T	13,517,056	16 Mb
XC5VFX70T	27,025,408	32 Mb
XC5VFX100T	39,389,696	64 Mb
XC5VFX130T	49,234,944	64 Mb
XC5VFX200T	70,856,704	128 Mb
XC5VTX150T	43,278,464	64 Mb
XC5VTX240T	65,755,648	128 Mb

Notes:

1. A larger SPI serial flash device can be used for daisy-chained applications, storing multiple FPGA configuration bitstreams, or for applications storing additional user data, such as code for the embedded MicroBlaze™ or PowerPC™ processors. Daisy-chaining multiple Spartan-3E FPGAs via a single flash is only supported on Stepping 1 and later.


V_{cc}
FPGA
PCB
FPGA
+5
plastic
Box
jumper
FPGA
+5
mode pins



Notes:

1. VCCO_2 supplies the SPI configuration dual-mode pins: MOSI, FCS_B, and FS[2:0]
2. VCC_CONFIG (Vcco_0) is the configuration output supply voltage and supplies the dedicated configuration pins: TMS, TCK, TDO, TDI, M[2:0], HSWAPEN, PROG_B, DONE, INIT_B, CCLK, D_IN.
3. PROG_B should be held Low during the direct programming of the SPI serial flash. PROG_B can be driven Low to High with external logic to reconfigure the FPGA.
4. HSWAPEN can be driven Low to enable pull-ups on I/O. Refer to Table 3, page 7 and [Ref 2] for details and options on this pin.
5. Control signals should be driven appropriately when programming the SPI serial flash. Signals such as the $\overline{W}$ and $\overline{HOLD}$ signals should be held High or inactive while programming the SPI serial flash. Refer to the vendor's data sheet for more details.
6. Refer to Table 5, page 11 for cable signal cross reference.
7. **Caution!** Care should be taken with the CCLK board layout. The Virtex-5 FPGA drives the internally generated CCLK signal to the FPGA CCLK output pin. The FPGA's internal configuration logic is clocked by the CCLK signal at the FPGA pin, therefore, any noise on the CCLK pin can affect the FPGA configuration. Guidelines and details for CCLK design see the "Board Layout for Configuration Clock (CCLK)" section in [Ref 3].

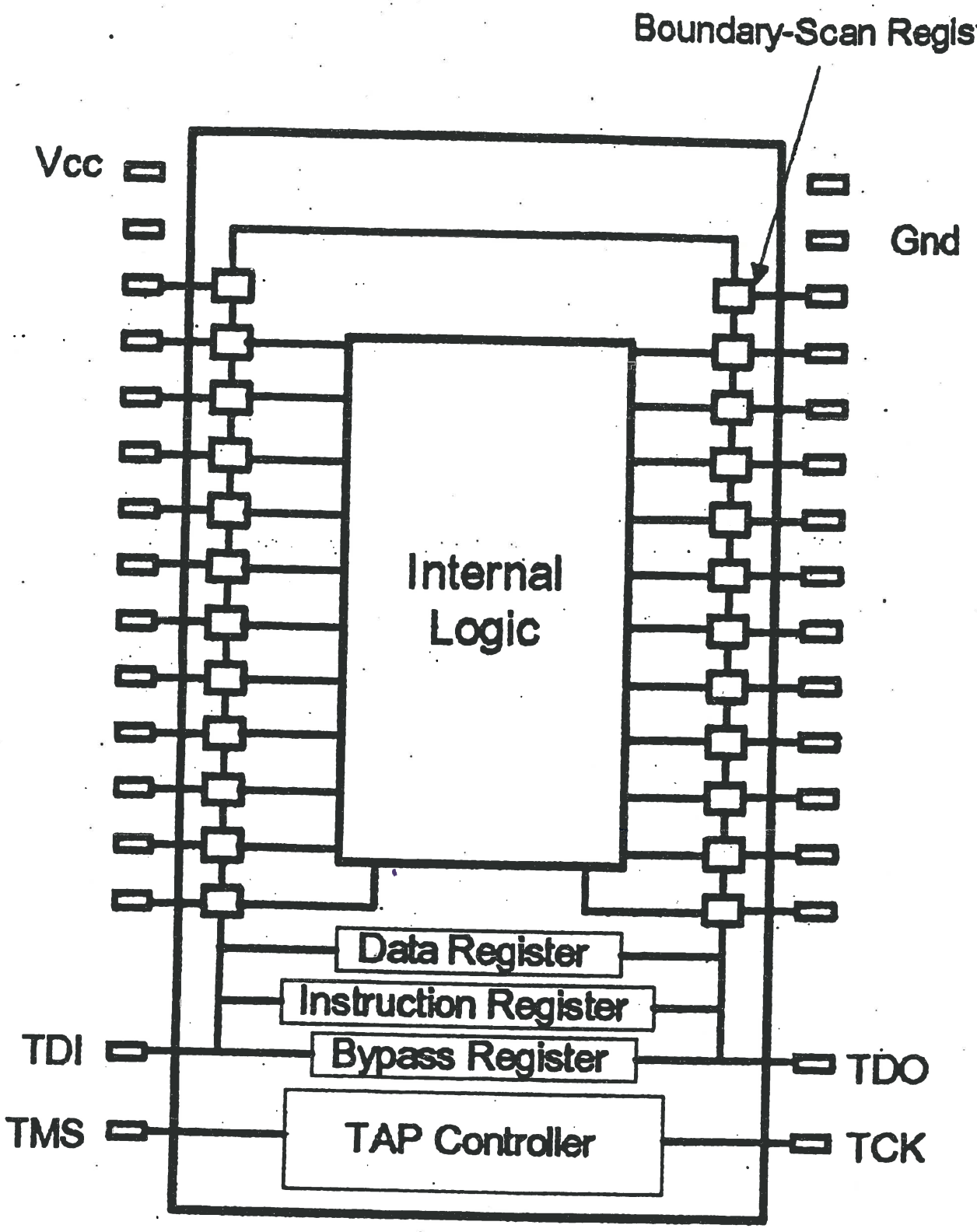
Figure 2: **Virtex-5 FPGA Configuration from Numonyx SPI Serial Flash Connection Diagram (Example for the Read Command 0x03)**

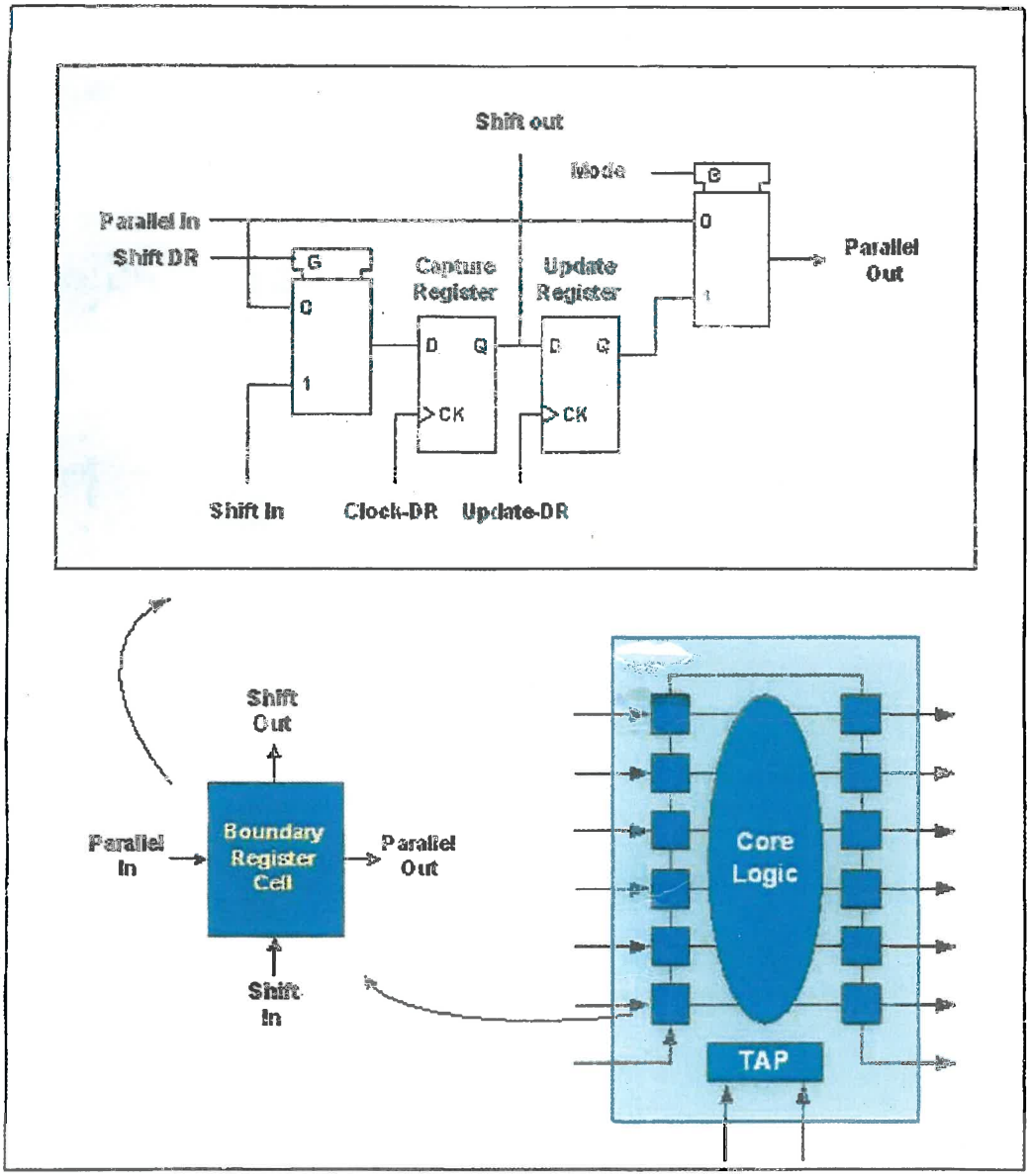
Joint test action Group

(JTAG)

programming of FPGAs

Always (or at least you should) provide a JTAG I/F on any FPGA-based design


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


Shift out
Mode
G
Parallel In
Shift DR
G
Capture Register
Update Register
0
Parallel Out
C
1
D
Q
CK
D
Q
CK
Shift In
Clock-DR
Update-DR
Shift Out
Parallel In
Boundary Register Cell
Parallel Out
Shift In
Core Logic
TAP

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

Table 1: **Virtex TAP Controller Pins**

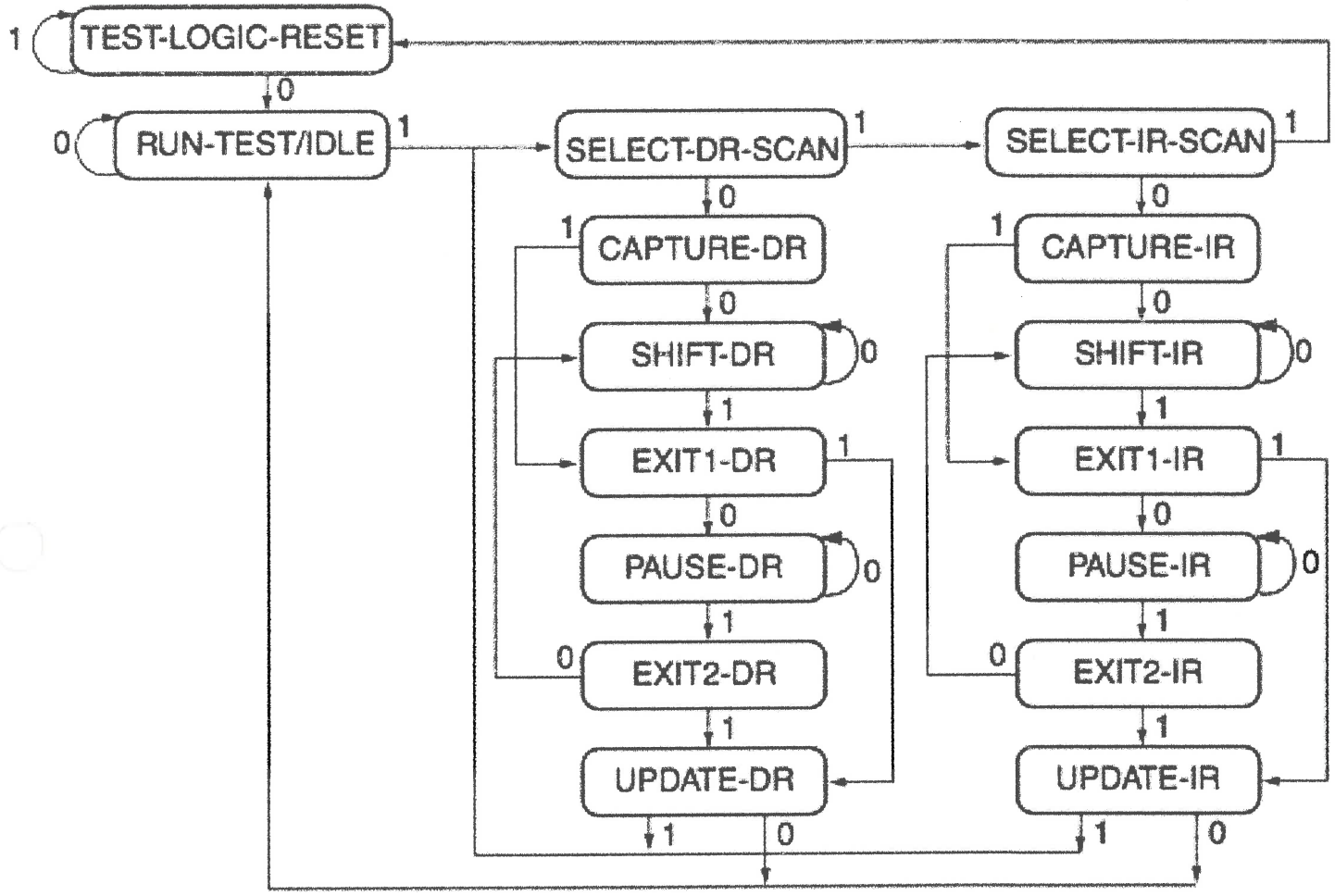
Pin	Description
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
TCK	Test Clock

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.

TMS is the FSM input



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**

	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

also called "download cables"

Platform Cable USB II

DS593 (v1.5) June 23, 2015

Features

- High-performance FPGA and PROM programming and configuration
  - Includes innovative FPGA-based acceleration firmware encapsulated in a small form factor pod attached to the cable
  - Leverages high-speed slave-serial mode programming interface

    Note: Slave-serial mode is supported in Xilinx® iMPACT software v10.1.
  - Recommended for prototyping use only
- Easy to use
  - Fully integrated and optimized for use with Xilinx iMPACT software
  - Intuitive multiple cable management from a single application
  - Supported on the following operating systems:

    Note: See the Xilinx design tool release notes for supported operating systems.
    - Microsoft Windows XP Professional
    - Microsoft Windows Vista
    - Red Hat Enterprise Linux
    - SUSE Linux Enterprise
  - Automatically senses and adapts to target I/O voltage
  - Interfaces to devices operating at 5V (TTL), 3.3V (LVCMOS), 2.5V, 1.8V and 1.5V
  - Intuitive flyleads-to-cable interface labeling
- Reliable
  - Backwards compatibility with Platform Cable USB, including Pb-Free (RoHS-compliant)
  - USB Integrators Forum (USB-IF) certified
  - CE and FCC compliant
  - Leverages industry standards, including JTAG boundary-scan IEEE 1149.1, SPI and USB 2.0
- Programs and configures all Xilinx devices
  - XC18V00 ISP PROMs
  - Platform Flash XCF00S/XCF00P/XL PROMs
  - All UltraScale™, 7 series, Virtex®, and Spartan® FPGA families, and Zynq-7000 AP SoCs
  - XC9500XL and CoolRunner™ XPLA3 / CoolRunner-II CPLDs

    Note: Xilinx iMPACT software or Vivado design tools are required for programming and configuration. See the design tool release notes for supported devices.
- Third-party PROM device programming support
  - Directly programs selected Serial Peripheral Interface (SPI) flash memory devices

    Note: Direct SPI flash memory programming supported in Xilinx iMPACT software v10.1.
  - Indirectly programs selected SPI or parallel flash memory devices via FPGA JTAG port
- Highly optimized for use with Xilinx design tools
  - Vivado® design tools or ISE® design tools
  - Embedded Development Kit
  - ChipScope™ Pro Analyzer
  - System Generator for DSP

Platform Cable USB II Description

Much more than just a simple USB cable, Platform Cable USB II (Figure 1) provides integrated firmware (hardware and software) to deliver high-performance, reliable and easy-to-perform configuration of Xilinx devices.

Platform Cable USB II attaches to user hardware for the purpose of configuring Xilinx FPGAs, programming Xilinx PROMs and CPLDs, and directly programming third-party SPI flash devices. In addition, the cable provides a means of indirectly programming Platform Flash XL, third-party SPI flash memory devices, and third-party parallel NOR flash memory devices via the FPGA JTAG port. Furthermore, Platform Cable USB II is a cost effective tool for debugging

embedded software and firmware when used with applications such as Xilinx's Embedded Development Kit and ChipScope Pro Analyzer.

Platform Cable USB II is an upgrade to and replaces Platform Cable USB. Similar to its popular predecessor, Platform Cable USB II is intended for prototyping environments only. Platform Cable USB II is backwards Compatible with Platform Cable USB and is supported by all Xilinx design tools that support Platform Cable USB.

Platform Cable USB II attaches to the USB port on a desktop or laptop PC using an off-the-shelf Hi-Speed USB A–B cable. The cable derives all operating power from the hub port controller — no external power supply is required.

Note: Sustained data transfer rates in a Hi-Speed USB environment vary according to the number of USB devices sharing the hub bandwidth. Native signaling rate (480 MHz) is not directly correlated to application throughput.

Device configuration and programming operations using Platform Cable USB II are supported by Xilinx iMPACT download software using boundary-scan (IEEE 1149.1/IEEE 1532), slave-serial mode, or serial peripheral interface (SPI). The Vivado design tools support device configuration with the Platform Cable USB II using boundary-scan (IEEE 1149.1).

Note: iMPACT software is bundled with the ISE design tools and WebPACK™ ISE software. The slave-serial mode and direct SPI are only supported in limited versions of the ISE iMPACT tool.

In addition, Platform Cable USB II is optimized for use with the Xilinx Embedded Development Kit, ChipScope Pro Analyzer, and System Generator for DSP. When used with these software tools, the cable provides a connection to embedded target systems for hardware configuration, software download, and real-time debug and verification. Target clock speeds are selectable from 750 kHz to 24 MHz.

Platform Cable USB II attaches to target systems using a 14-conductor ribbon cable designed for high-bandwidth data transfers. An optional adapter for attaching a flying lead set is included for backward compatibility with target systems not using a ribbon cable connector.

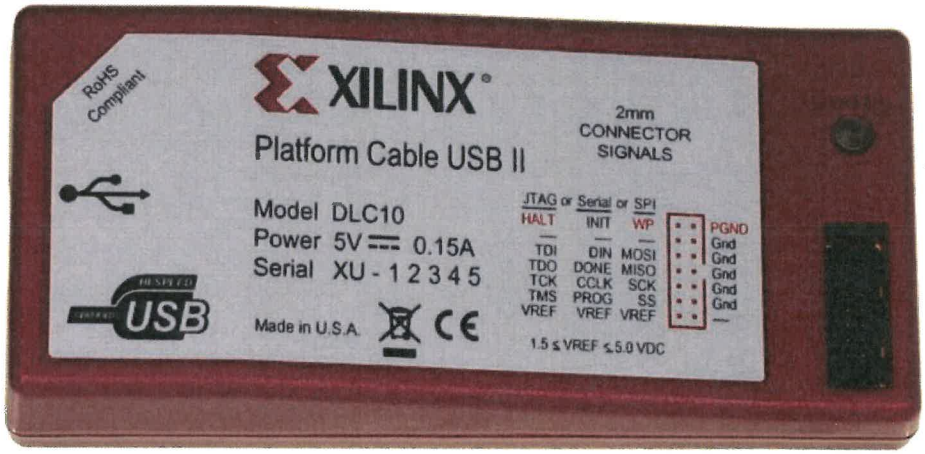


Figure 1: **Xilinx Platform Cable USB II**

Physical Description

The Platform Cable USB II electronics are housed in a recyclable, fire-retardant plastic case (Figure 2). An internal EMI shield attenuates internally generated emissions and protects against susceptibility to radiated emissions.

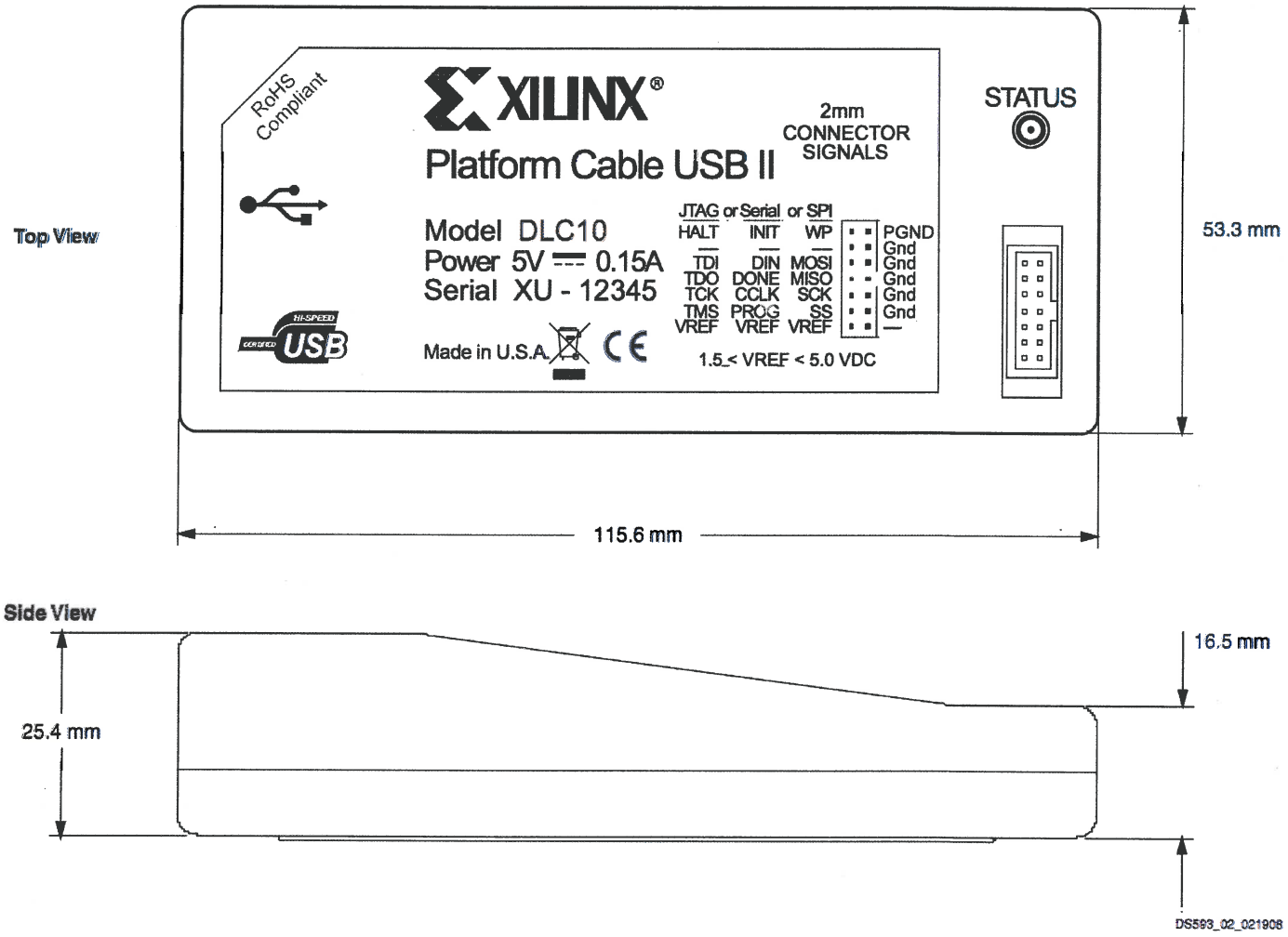


Figure 2: **Plastic Case Physical Description**

Operation

This section describes how to connect and use Platform Cable USB II.

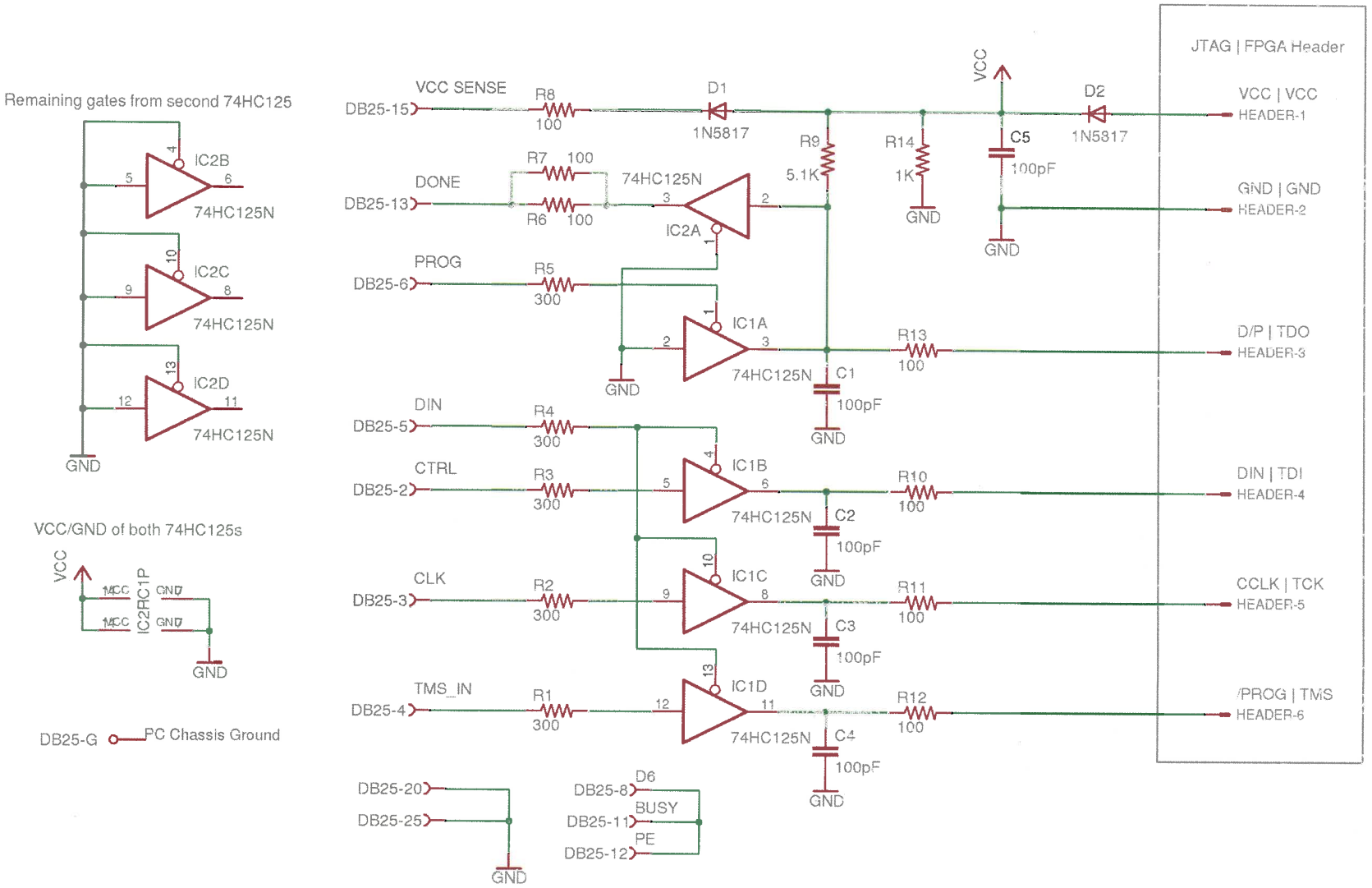
Minimum Host System Requirements

The host computer must contain a USB Host Controller with one or more USB ports. The controller can reside on the PC motherboard, or can be added using an expansion or PCMCIA card.

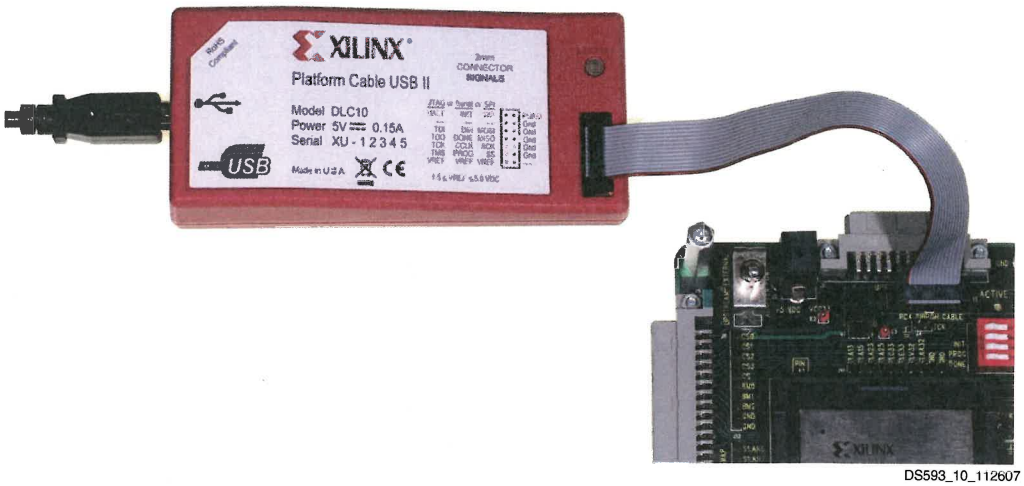
Platform Cable USB II is designed to take full advantage of the bandwidth of USB 2.0 ports, but it is also backward-compatible with USB 1.1 ports. Refer to USB Hub Types and Cable Performance, page 29 for additional information on connection environments and bandwidth.

For Platform Cable USB II compatibility with Vivado design tools, see the Vivado design tools release notes. The *Architecture Support and Requirements* chapter in the release notes lists the supported operating systems and cable installation requirements.

"JTAG/Parallel Download Cable" by Xilinx



The connector is a 2 mm shrouded keyed header. See Table 5, page 15 for vendor part numbers and pin assignments.



Notes:

1. Ribbon Cable: 14-pin conductor, 1.0 mm center, round-conductor flat cable, 28 AWG (7 x 36) stranded conductors, gray PVC with pin 1 edge marked.
2. 2 mm ribbon female polarized connector, IDC connection to ribbon. Contacts are beryllium copper plated 30 micro-inches gold plating over 50-micro-inches nickel. The connectors mate to 0.5 mm square posts on 2 mm centers.

Figure 10: **High Performance Ribbon Cable**

Flying Wire Adapter

An adapter with wires (Figure 11) is provided for attachment to legacy target systems that do not incorporate a shrouded 2 mm connector. The adapter makes it possible to use flying wires for connections to distributed terminals on a target system.

The adapter is a small circuit board with two connectors (Figure 12). The connector on the bottom side of the adapter mates with the 14-pin Platform Cable USB II male 2 mm connector. A 7-pin right-angle header on the top side of the adapter mates with the standard Xilinx flying wire set.

Note: This method of connection is not recommended because it can result in poor signal integrity. Additionally, damage can result if the leads are unintentionally connected to high voltages.

The Xilinx product number for the flying wire set is HW-USB-FLYLEADS-G.

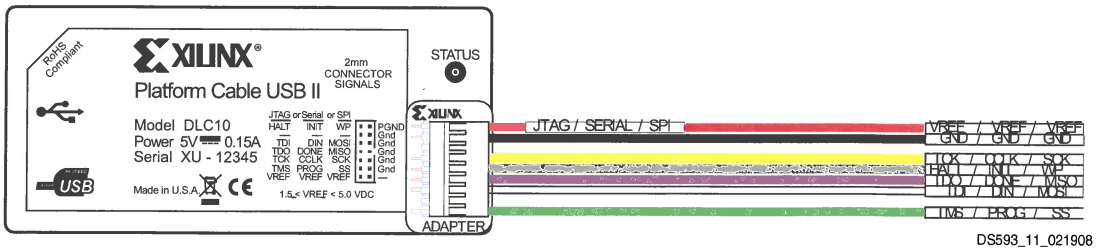
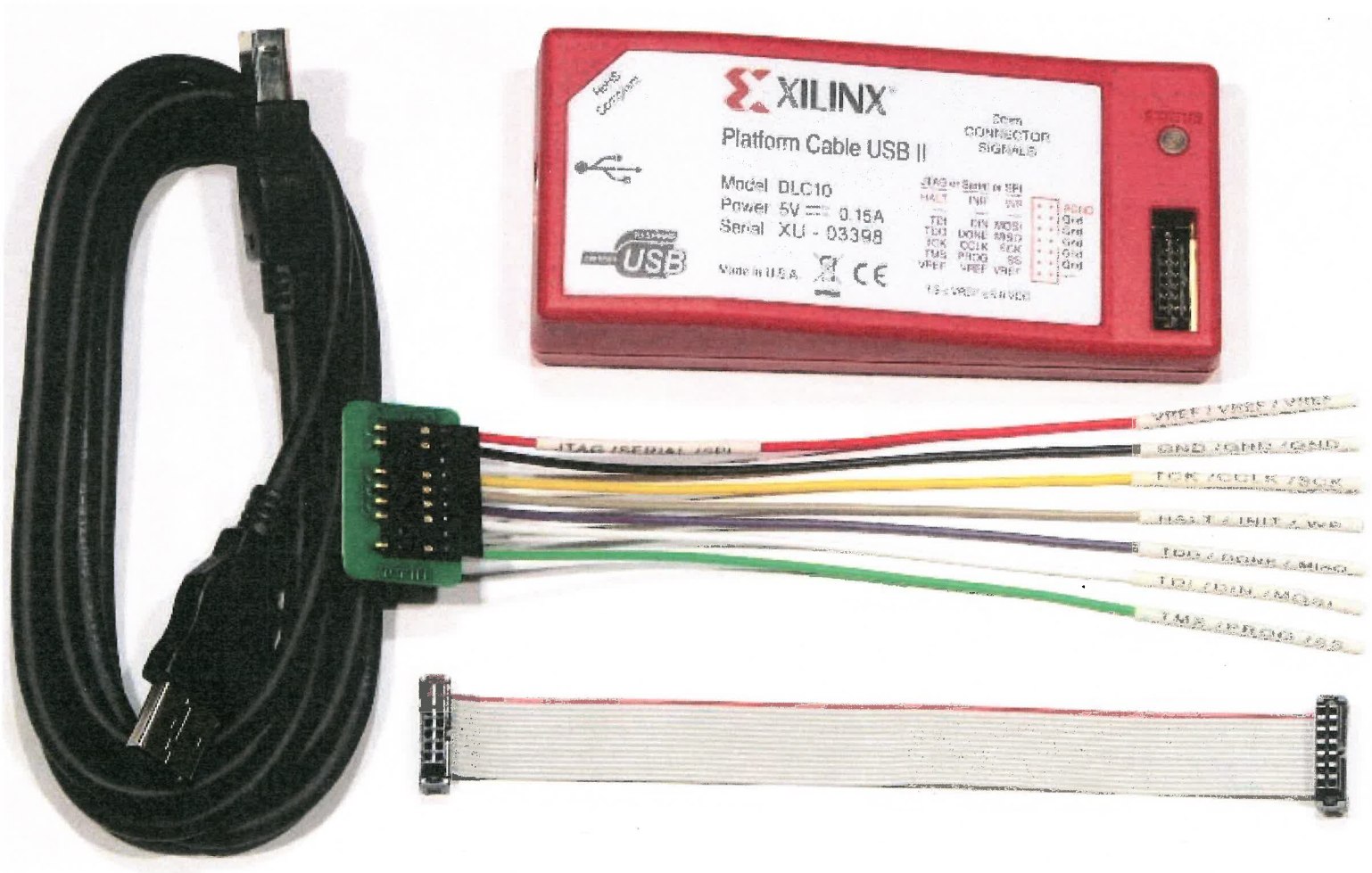


Figure 11: **Flying Wire Adaptor (Top) with Wires**


XILINX
Platform Cable USB II
Model DLC10
Power 5V 0.15A
Serial XU - 03398
USB
Made in U.S.A.
CE
20-pin CONNECTOR SIGNALS

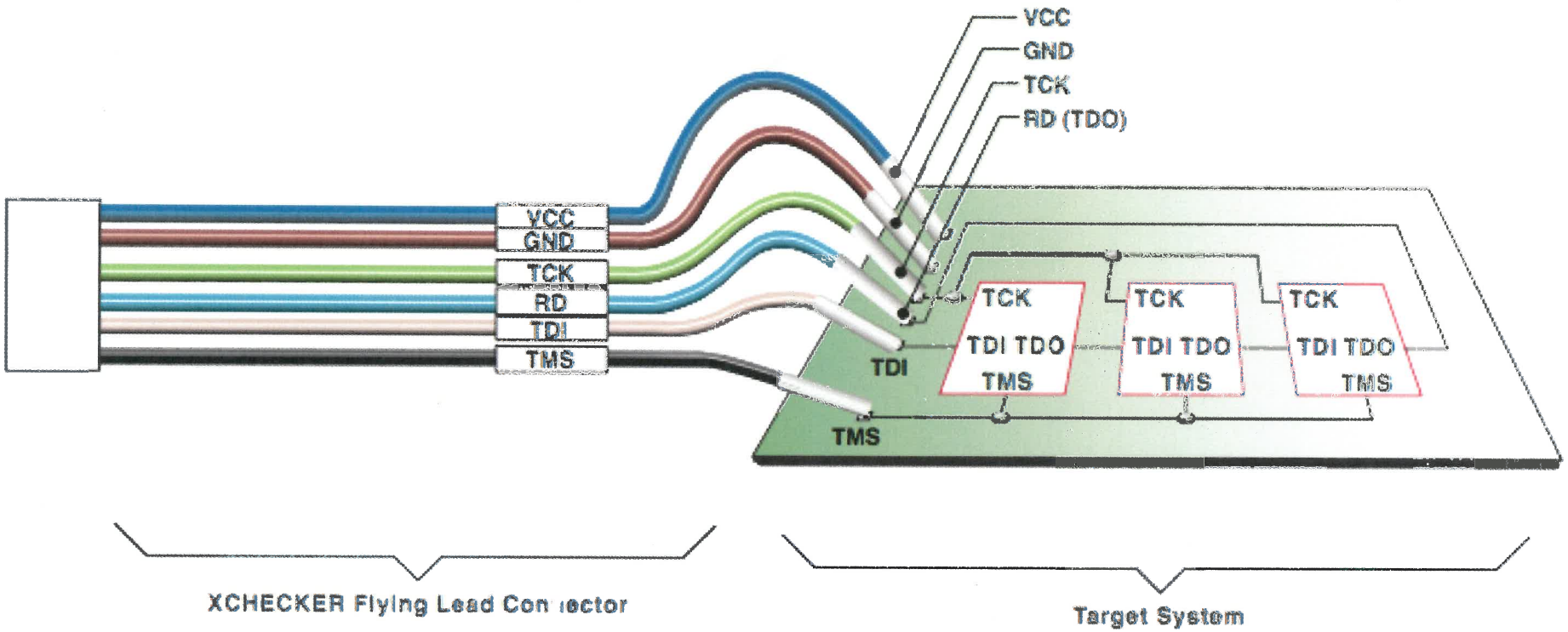


Figure 2-3 XChecker Connections to JTAG Boundary-scan TAP

Cable Connections

Connections between the cable assembly and the target system use only 6 of the sixteen leads. For connection to JTAG boundary-scan systems you need only ensure that the VCC, GND, TDI, TCK, TMS and RD (TDO) pins are connected.

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 bitstream, the FPGA then reads the flash memory to configure itself.

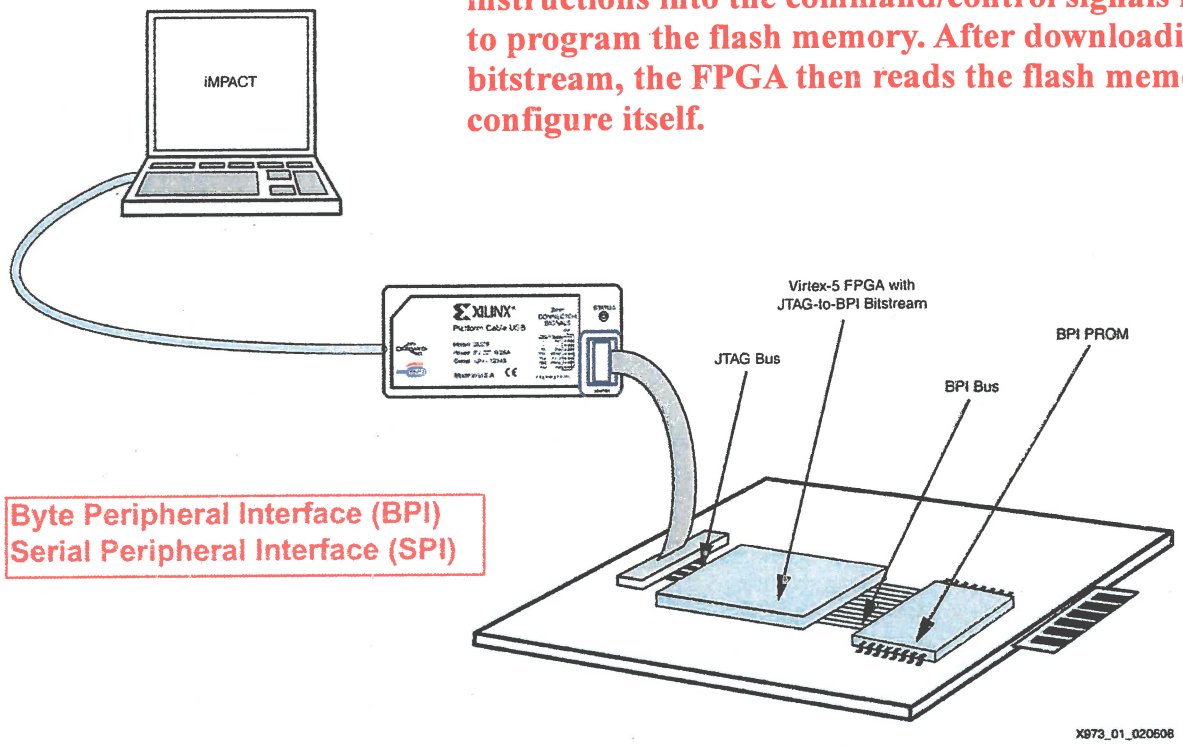


Figure 1: **iMPACT Indirect BPI PROM Programming with a Virtex-5 FPGA**

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.

SPI Serial Flash Programming Using ispJTAG on LatticeECP/EC FPGAs

September 2012 Technical Note TN1078

Introduction

Like all SRAM FPGAs the LatticeECP™ and LatticeEC™ devices need to be configured at power-up. This configuration can be done via:

1. Serial Peripheral Interface (SPI) boot memory
2. Traditional FPGA boot memory
3. JTAG
4. Microprocessor interface

If a boot memory is desired the SPI approach provides a number of advantages over traditional FPGA boot memory:

1. SPI devices are available from multiple vendors ensuring stable supply
2. The cost of SPI memory is up to 75% less than traditional FPGA boot memory
3. SPI memory is available in space saving 8-pin packages that are considerably smaller than packages used for traditional FPGA boot memory

Like all boot memories, SPI Serial Flash needs to be loaded with the FPGA configuration data. There are three options for programming an SPI memory used in conjunction with a LatticeECP/EC device. The SPI memory can be configured off-board using a stand-alone programmer, the memory can be programmed on-board using its SPI interface, or the memory can be programmed on-board via JTAG through the LatticeECP/EC device.

This technical note details the on-board configuration of SPI memory via the JTAG interface.

Related Documents

The following documents are available for download from the Lattice web site at www.latticesemi.com.

- *LatticeECP & EC – Low-Cost FPGA Configuration via Industry-Standard SPI Serial Flash*
- *LatticeECP/EC Family Handbook*
- Lattice technical note TN1053, *LatticeECP/EC sysCONFIG™ Usage Guide*
- *ispDOWNLOAD® Cable Data Sheet*

Hardware and Software Requirements

- An ispDOWNLOAD Cable, either USB or Parallel. Refer to the *ispDOWNLOAD Cable Data Sheet* for part numbers
- Properly installed ispLEVER® 4.2 or later
- Properly installed ispVM® System 14.3 or later

SPI/SPIX Differences

The majority of SPI Serial Flash on the market support a common read Operation Code (Op Code). LatticeECP/EC devices offer direct connection to these devices by hardwiring the read Op Code (03H) into the FPGA silicon. These devices are sometimes referred to as SPI3 devices because they support this common Read Op Code.

SPIX mode allows the LatticeECP/EC to easily interface to SPI Serial Flash devices that support a different read Op Code. This can be done with pull-up and pull-down resistors on the PCB wired to the SPID[7:0] pins, telling the

FPGA which Read Op Code that particular Flash device supports. If the configuration pins CFG2, CFG1, and CFG0 are 0, 0, 1 (respectively) at power-up the FPGA will use the Read Op Code hardwired on the PCB to access the Flash.

Table 21-1. Device Configuration Codes

CFG2	CFG1	CFG0	Configuration Mode
0	0	0	SPI Serial Flash
0	0	1	SPIX Serial Flash
1	0	0	Master Serial
1	0	1	Slave Serial
1	1	0	Master Parallel
1	1	1	Slave Parallel
X	X	X	ispJTAG™ (always available)

Note: The configuration mode pins indicate the type of device the FPGA will configure from at power-up. For SPI Serial Flash the mode pins should be set to '000' or '001'.

Table 21-2. Manufacturers of "SPI3" Compatible Flash

Manufacturer	Device Family
ST Microelectronics	M25P
NexFLASH	NX25P
Silicon Storage Technology	SST25VF
Saifun	SA25F
Spansion	S25FL
PMC	Pm25LV
Atmel	AT25F

Note: This is not meant to be an exhaustive list of manufacturers or device families.

SPI Serial Flash Sizing

As depicted in Table 21-3, the density of the FPGA determines the size requirement for the SPI Serial Flash. Smaller Flash sizes can be realized by using the compression option in ispLEVER.

Table 21-3. Selecting Flash Density

Family	Device	Max Config Bits (Mb)	Required Boot Memory (Mb)	
			No Comp	Comp (typ)
LatticeECP/EC	EC1	0.6	1	25%
	EC3	1.1	2	25%
	ECP6/EC6	1.8	2	25%
	ECP10/EC10	3.1	4	25%
	ECP15/EC15	4.3	8	25%
	ECP20/EC20	5.3	8	25%
	ECP33/EC33	7.9	8	25%

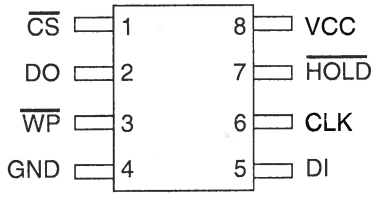
Hardware

This section describes how to physically wire the SPI Serial Flash into a design.

SPI Serial Flash Interface

The standard pin-out for 8-pin SPI Serial Flash memories is shown below (top view):

Figure 21-1. 8-Pin SPI Flash Memory, Standard Pinout



Note: 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 LatticeECP/EC packages these signals are located in bank 3.

The SPI interface is a 4-wire serial interface comprised of the following signals:

1. **CS – Chip Select, Input:** Enables and disables device operation. When high the device is at standby power levels and the output is tri-stated. When low the device powers up and instructions can be written to and data read from the device.
2. **CLK – Serial Clock, Input**: Provides timing for the interface. The Serial Data Input (DI) is latched on the rising edge of CLK. Serial Data Output (DO) changes after the falling edge of CLK.
3. **DI – Serial Data In, Input**: When the device is enabled (CS is low) this pin allows instructions, addresses, and data to be serially written to the device. Data is latched on the rising edge of the CLK.
4. **DO – Serial Data Out, Output**: When the device is enabled (CS is low) this pin allows data and status to be serially read from the device. Data is shifted out on the falling edge of the CLK.

The SPI interface also supports the following two functions:

1. **HOLD – Input**: Allows device to be paused without de-selecting it. When HOLD is low DO will be tri-stated while DI and CLK are ignored. This function is useful when multiple devices are sharing the same SPI signals.
2. **WP – Write Protection, Input**: Used to prevent inadvertent writing of the Status Register Block Protect bits.

Note: The LatticeECP/EC SPI interface supports the basic 4-wire interface, but the user is free to implement these additional functions if desired.

ispJTAG Interface

The ispJTAG interface supports both IEEE 1149.1 Boundary Scan and IEEE 1532 In-System Configuration. Standard pinouts for 1x10, 1x8, and 2x5 download headers are shown in Table 21-4. The 1x10 header is preferred but ultimately the header chosen will depend on the available download cable. All new download cables have uncommitted "flywire" connections, so they can be attached to any of the header styles. Direction, in Table 21-4, refers to the cable, for example "output" indicates an output from the cable to the FPGA.

Table 21-4. Download Header Pinout

Pin Name	1x10	1x8	2x5	Direction	Description
V_{CCJ}	1	1	6	—	3.3V or 2.5V
TDO	2	2	7	Input	Test Data Out
TDI	3	3	5	Output	Test Data In
PROGRAMN	4	4	10	Output	Forces FPGA config, N/C
TRST	5	5	9	Output	Test Reset, N/C
TMS	6	6	3	Output	Test Mode Select
GND	7	7	2, 4, 8	—	Ground
TCK	8	8	1	Output	Test Clock
DONE	9			Input	FPGA configuration complete, optional
INITN	10			Input	FPGA ready for configuration, optional

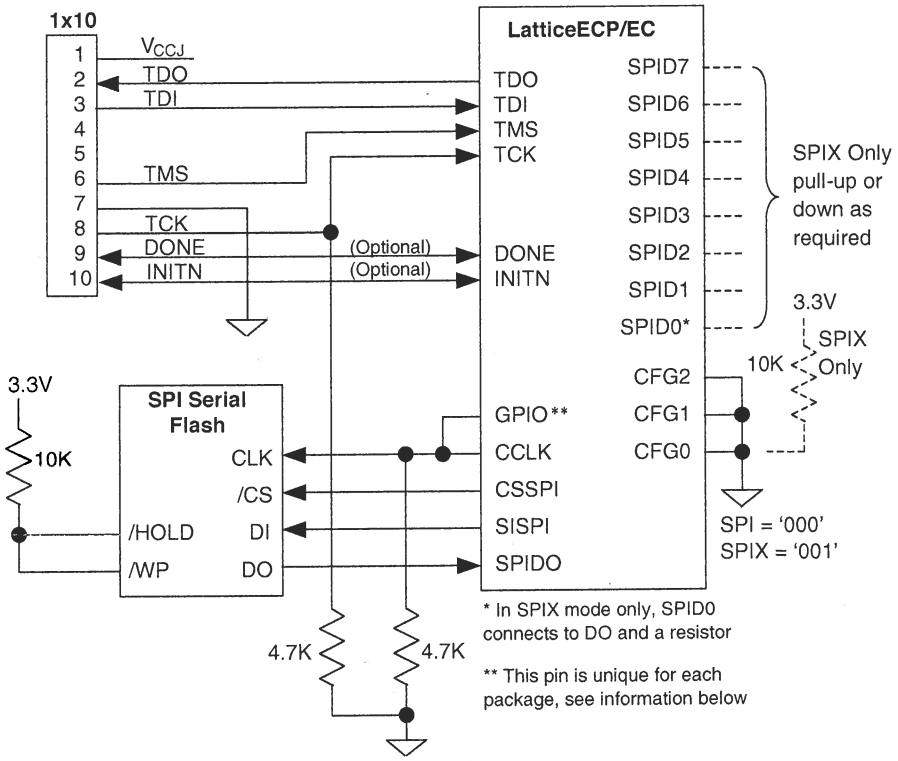
1. **VCCJ** – V_{CC} JTAG: Powers the ispDOWNLOAD Cable.
2. **TDO** – Test Data Out: Serial data read from the test device to the cable.
3. **TDI** – Test Data In: Serial data written from the cable to the device.
4. **PROGRAMN**: Initiates a configuration sequence when asserted low. Not used and should not be connected on the board.
5. **TRST** – Test Reset: Not used and should not be connected on the board.
6. **TMS** – Test Mode Select: Controls the IEEE 1149.1 state machine.
7. **TCK** – Test Clock: Clocks the IEEE 1149.1 state machine.
8. **GND**: Digital ground.
9. **DONE** – Optional, USB Only: Open drain, internal pull-up. A high indicates that the FPGA configuration sequence 9 was completed successfully.
10. **INITN** – Optional, USB Only: Open drain, internal pull-up. A high indicates that the FPGA is ready to be configured. A low indicates the FPGA is not ready to be configured, or an error occurred during configuration.

Note: Use of the DONE and INITN pins, while optional, does allow ispVM System to check that configuration completed successfully. If DONE or INITN are wired to the connector then the proper dialog box(es) must be checked in the Cable and I/O Port Setup section of ispVM System. See the Software section of this document for more details.

Schematic

The schematic in Figure 21-2 illustrates how to wire the ispJTAG connector, FPGA, and SPI Serial Flash.

Figure 21-2. 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 21-5.
- 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.

8. Click on **OK**.
9. From the main project window click the green **GO** button on the toolbar; this will begin the download process.
10. Upon successful download, in order to configure the FPGA with the new configuration, the user must cycle power to the FPGA or pulse the FPGA's Program pin low then high.

Including the SPI Interface in the FPGA Design

If the user wishes JTAG to have access to the SPI Serial Flash while the FPGA is operating, for instance to allow background configuration updates, then the Soft SPI Interface must be instantiated into the user code. Once a configuration bitstream containing the user code and the Soft SPI Interface has been created the programming sequences will be identical to those detailed above (see step 5).

Sample Code

The following code samples are simple VHDL and Verilog files that show how to instantiate the netlist file, i.e. the Soft SPI Interface. The netlist file should be placed in the same directory as the top design file. In the following examples the netlist file is called SPITOP.ngo. The netlist file, along with these sample source files, is freely available from the Lattice Semiconductor web site at www.latticesemi.com.

Verilog

Here is the same design in Verilog.

```
module design_top(rst, sclk, cnt_out, spi_c, spi_d, spi_sn, spi_q) ;

input         sclk, rst;
output [7:0]  cnt_out;

// SPI Serial Flash pins

input         spi_q;
output        spi_d, spi_sn, spi_c;

reg [7:0]     cnt_out;

// User code

always @ (posedge sclk or posedge rst)
     begin
          if (rst)
               cnt_out = 2'h00;
          else
               cnt_out = cnt_out + 1;
      end

// Instantiate Soft SPI Interface

SPITOP spi_ip
            (
             .SPI_PIN_C(spi_c),
             .SPI_PIN_D(spi_d),
             .SPI_PIN_SN(spi_sn),
             .SPI_PIN_Q(spi_q)
             );

endmodule

module SPITOP (SPI_PIN_C, SPI_PIN_D, SPI_PIN_SN, SPI_PIN_Q);

output      SPI_PIN_D, SPI_PIN_SN, SPI_PIN_C ;
input       SPI_PIN_Q;

endmodule
```

The Soft SPI Interface is only needed to connect JTAG to the SPI Serial Flash interface. If the SPI Serial Flash will be used by the user design, for instance as scratch memory, and background access via JTAG will not be required, then the Soft SPI Interface is not needed, and the user code can access the SPI pins directly. Remember that the configuration memory must start at address zero; any user defined memory space must be located above the configuration data. It is recommended that an address above the maximum possible configuration size be chosen. For instance if an ECP/EC20 device is being used, select a scratch pad starting address above 5.3Mb (see Table 21-3 in this document).

Locking the Pins

The last thing the user needs to do is tell ispLEVER which pins are connected to the SPI Serial Flash device (see Figure 21-2 and Table 21-5). From the ispLEVER Project Navigator window click on the package name in the left window, then double click on the Pre-Map Preference Editor in the right window (see Figure 21-15).