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:

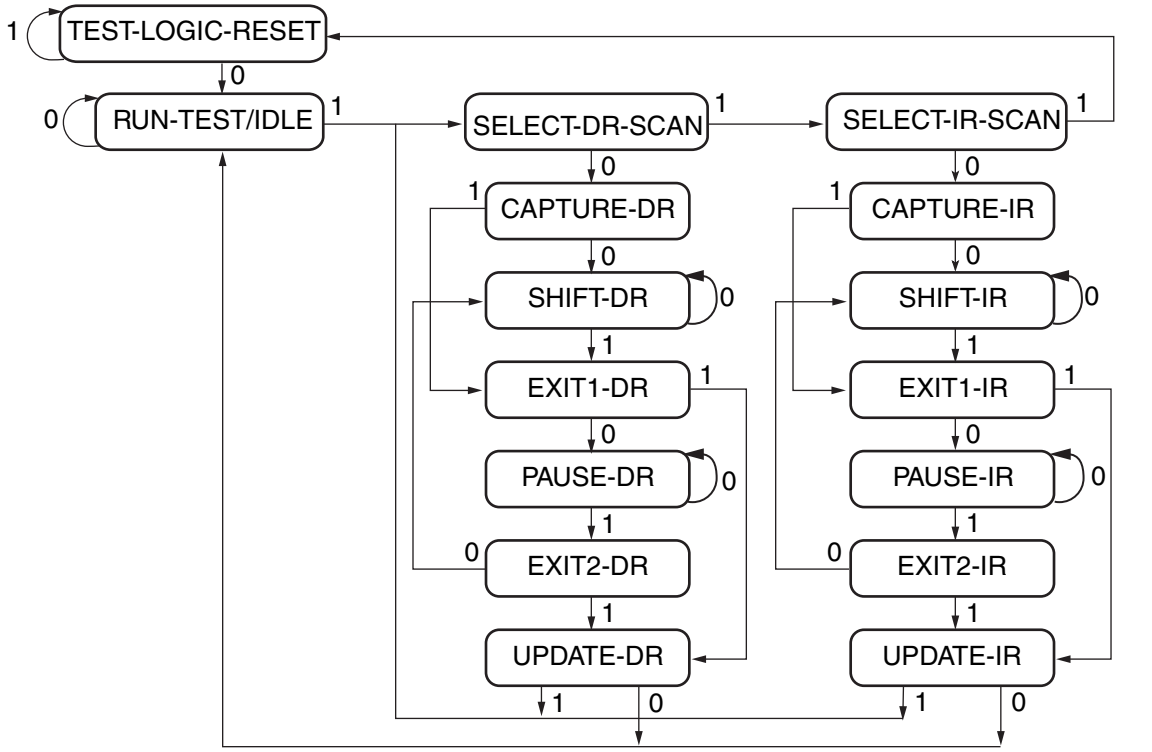
- TMS - The sequence of states through the TAP controller is determined by the state of the TMS pin on the rising edge of TCK. TMS has an internal resistive pull-up to provide a logic High if the pin is not driven.
- TCK - This pin is the JTAG test clock. It sequences the TAP controller and the JTAG 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

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

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

Boundary-Scan Register

The test primary data register is the Boundary-Scan register. The Boundary-Scan operation is independent of individual input/output block (IOB) configurations. Each IOB, bonded or unbonded, starts out as bidirectional with 3-state control. Later, it can be configured to be an input, output, or 3-state only. Therefore, three data register bits are provided per IOB (Figure 2).

When conducting a data register (DR) operation, the DR captures data in a parallel fashion during the CAPTURE-DR state. The data is then shifted out and replaced by new data during the SHIFT-DR state. For each bit of the DR, an update latch is used to hold the input data stable during the next SHIFT-DR state. The data is then latched during UPDATE-DR state when the TCK is Low.

The update latch is opened each time the TAP Controller enters the UPDATE-DR state. Care is necessary when exercising an INTEST or EXTEST to ensure the proper data has been latched before exercising the command. This is typically accomplished by using the SAMPLE/PRELOAD instruction.

Consider the presence of internal pull-ups and pull-down resistors when developing test vectors for testing opens and shorts. The IOB can be connected to an internal pull-up or pull-down resistor depending on the configuration state of the FPGA. (For more information on Virtex configuration modes and IOB connections to pull-up resistors, see the section on "Configuring through Boundary-Scan.")

- For an FPGA that is not yet configured, the Virtex configuration mode determines whether or not to connect the IOB to an internal pull-up resistor.
- For a configured FPGA, the connectivity of an IOB to an internal pull-up or pull-down resistor depends on the user configuration of the IOB or the BitGen setting for unused pins.

Figure 2 shows the Virtex Boundary-Scan architecture.

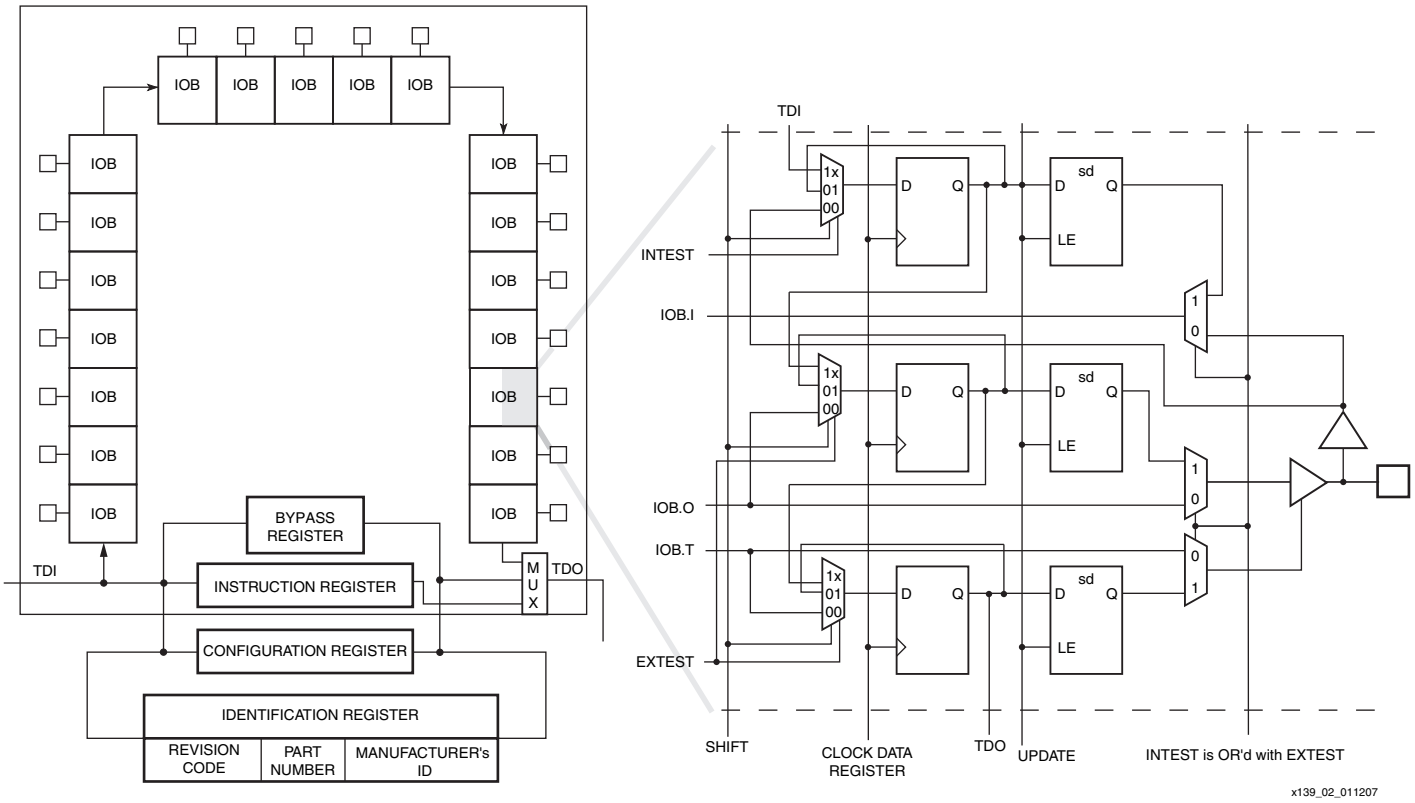


Figure 2: **Virtex Series Boundary-Scan Logic**

Bit Sequence

The order in each non-TAP IOB is described in this section. The input is first, followed by the output and finally the 3-state IOB control. The 3-state IOB control is closest to the TDO. The input-only pins contribute only the input bit to the Boundary-Scan I/O data register. The bit sequence of the device is obtainable from the “Boundary-Scan Description Language Files” (BSDL files) for the Virtex family. These files can be obtained from the Xilinx software download area. The bit sequence is invariant of the design. It always has the same bit order and the same number of bits.

Bypass Register

The other standard data register is the single flip-flop BYPASS register. It directly passes data serially from the TDI pin to the TDO pin during a bypass instruction. This register is initialized to zero when the TAP controller is in the UPDATE-DR state.

Instruction Register

The instruction register is a 5-bit register that loads the OPCODE necessary for the Virtex Boundary-Scan instruction set. This register loads the current OPCODE and captures internal device status.

Configuration Register (Boundary-Scan)

The configuration register is a 32-bit register. This register allows access to the configuration bus and readback operations.

Identification Register

The Virtex devices have an identification register, commonly referred to as the IDCODE register. This register is based upon the IEEE Standard 1149.1 and allows easy identification of the part being tested or programmed through Boundary-Scan. Table 4 lists the general format of the identification register.

Table 4: **Virtex Identification Register**

Revision Code	Part Number		Manufacturers ID	LSB
	Family Code	Part Size Code		
31 ... 28	27 ... 21	20 ... 12	11 ... 1	0
XXXX	0000011	YYYYYYYYY	0000 1001 001	1

Table 5 lists specific IDCODES assigned to Virtex series FPGAs.

Table 5: **IDCODEs Assigned to Virtex Series FPGAs**

FPGA	IDCODE
XCV50	v0610093h
XCV50E	v0A10093h
XCV100	v0614093h
XCV100E	v0A14093h
XCV150	v0618093h
XCV200	v061C093h
XCV200E	v0A1C093h
XCV300	v0620093h
XCV300E	v0A20093h
XCV400	v0628093h
XCV400E	v0A28093h
XCV405E	v0C28093h
XCV600	v0630093h
XCV600E	v0A30093h
XCV800	v0638093h
XCV812E	v0C38093h
XCV1000	v0640093h
XCV1000E	v0A40093h
XCV1600E	v0A48093h
XCV2000E	v0A50093h
XCV2600E	v0A5C093h
XCV3200E	v0A68093h

Notes:

1. The "v" in the IDCODE is the revision code field.

USERCODE Register

USERCODE is supported in the Virtex family as well. This register allows a user to specify a design-specific identification code. The USERCODE can be programmed into the device and read back for verification at a later time. The USERCODE is embedded into the bitstream during bitstream generation (`bitgen -g` UserID option) and is valid only after configuration.

USER1, USER2 Registers

The USER1 and USER2 Boundary-Scan instructions are valid only after configuration. The user can define data registers associated with the USER1 and USER2 instructions within the FPGA design. After the FPGA is configured, the user can access the USER1 and USER2 data registers through the TAP pins.

The BSCAN_VIRTEX library macro is required when creating the USER1 and USER2 data registers. This symbol is required only for driving internal scan chains (USER1 and USER2). The BSCAN_VIRTEX macro provides two user pins (SEL1 and SEL2) that determine the usage of USER1 or USER2 instructions respectively.

For these instructions, two corresponding pins (TDO1 and TDO2) allow user scan data to be shifted out of TDO. In addition, there are individual clock pins (DRCK1 and DRCK2) for each user register. There is a common input pin (TDI) and shared output pins that represent the state of the TAP controller (RESET, SHIFT, and UPDATE). Unlike the earlier FPGA families where the BSCAN macro was required to dedicate the TAP pins for Boundary-Scan, the Virtex TAP pins are dedicated and do not require the BSCAN_VIRTEX macro for normal Boundary-Scan instructions or operations.

The user implements the data register corresponding to the USER1 or USER2 Boundary-Scan instruction. Figure 3 shows a sample implementation of a USER1 data register that is connected to the BSCAN_VIRTEX macro. For HDL, the BSCAN_VIRTEX macro must be instantiated in the design.

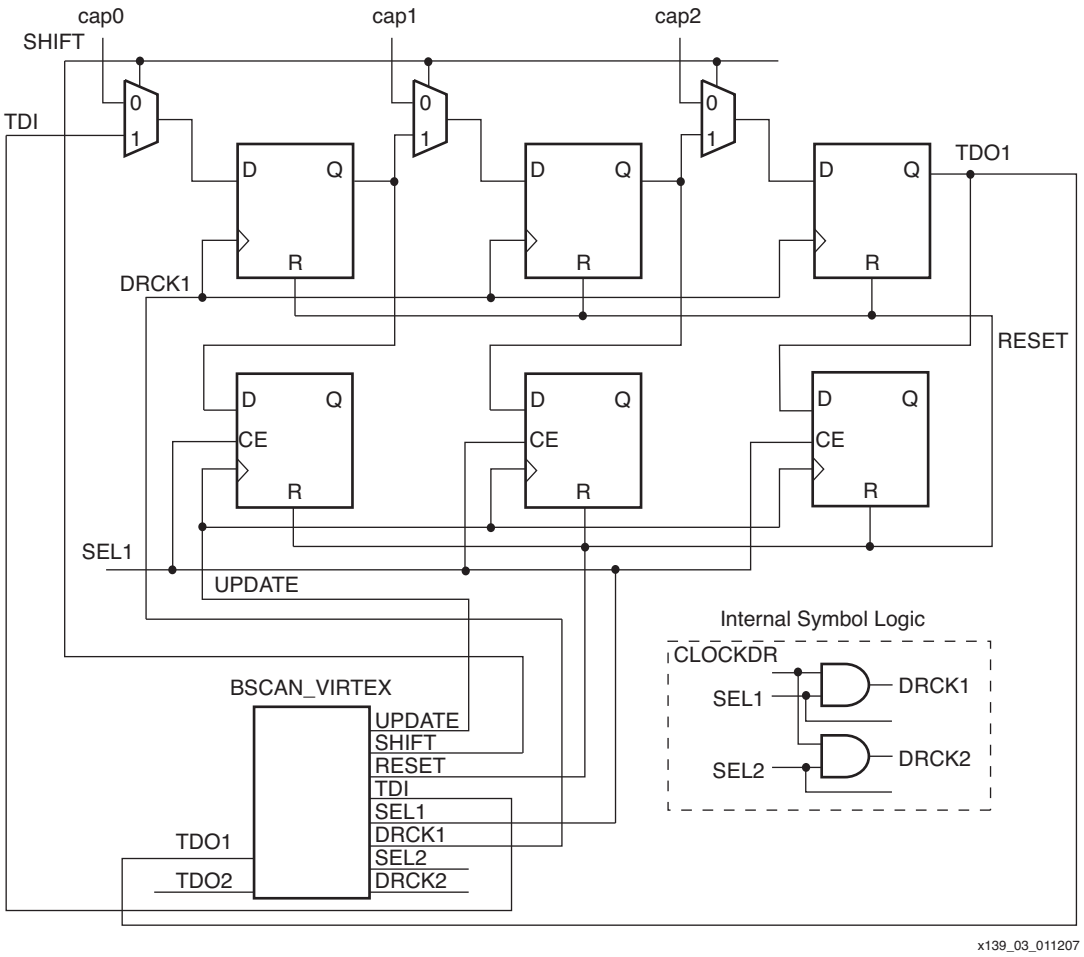


Figure 3: **BSCAN_VIRTEX (Sample Usage)**

Using Boundary-Scan in Virtex Devices

Figure 4 shows the Virtex Boundary-Scan Port Timing Waveforms.

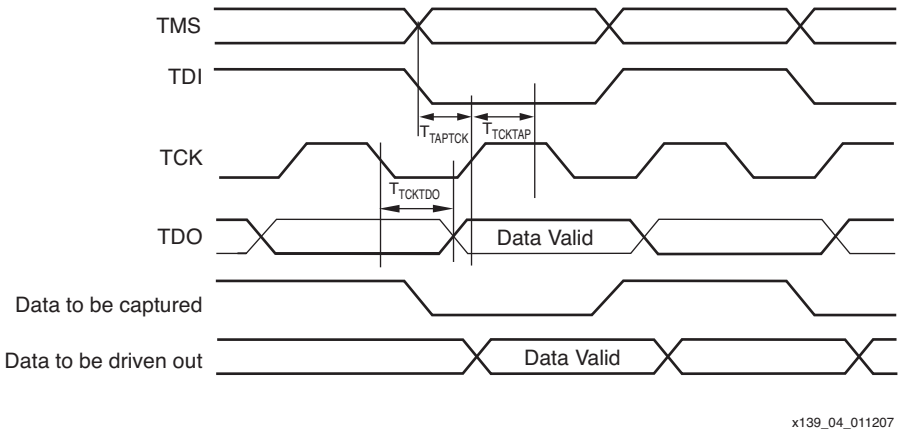


Figure 4: **Virtex Boundary-Scan Port Timing Waveforms**

Table 6 lists characterization data for some of the most commonly requested timing parameters.

For more information on the Startup sequence, bitstream, and internal configuration registers referred to in this application note, review application note **XAPP138 "Virtex FPGA Series Configuration and Readback."**

Table 6: **Boundary-Scan Port Timing Specifications**

Symbol	Parameter	-6	-5	-4	Units
T_{TAPTCK}	TMS and TDI setup time before TCK	4.0	4.0	4.0	ns minimum
T_{TCKTAP}	TMS and TDI hold times after TCK	2.0	2.0	2.0	ns minimum
T_{TCKTDO}	TCK falling edge to TDO output valid	11.0	11.0	11.0	ns maximum
F_{TCK}	Maximum TCK clock frequency	33.0	33.0	33.0	MHz maximum

Configuring through Boundary-Scan

One of the most common Boundary-Scan vendor-specific instructions is the configure instruction. An individual Virtex device is configured through JTAG on power-up using the TAP. If the Virtex device is configured on power-up, Xilinx recommends tying the mode pins to one of the following Boundary-Scan configuration mode settings.

- `101` (M2=1, M1=0, M0=1: contains no pull-ups on I/Os)
- `001` (M2=0, M1=0, M0=1: contains pull-ups on I/Os)

Table 7 lists the mode pin settings for all configuration modes, including Boundary-Scan modes.

Table 7: **Virtex Configuration Modes**

Configuration Mode	M2	M1	M0	Pull-ups
Master Serial	0	0	0	No
Slave Serial	1	1	1	No
SelectMAP Mode	1	1	0	No
Boundary-Scan	1	0	1	No
Master Serial (with Pull-ups)	1	0	0	Yes
Slave Serial (with Pull-ups)	0	1	1	Yes
SelectMAP (with Pull-ups)	0	1	0	Yes
Boundary-Scan (with Pull-ups)	0	0	1	Yes

Figure 5 shows the configuration flow for Virtex device configuration with JTAG. The sections that follow describe how the Virtex device can be configured as a single device through Boundary-Scan or as part of a multiple-device scan chain.

A configured device can be reconfigured by toggling the TAP and entering the CFG_IN instruction after pulsing the PROG pin or issuing the shut-down sequence. (Refer to the "Reconfiguring through Boundary-Scan" section.) For additional details on power-up or the start-up sequence in Virtex devices, refer to application note **XAPP138**. In addition, application note **XAPP058, "Xilinx In-System Programming Using an Embedded Microcontroller"** has detailed information on using Virtex devices in an embedded solution.

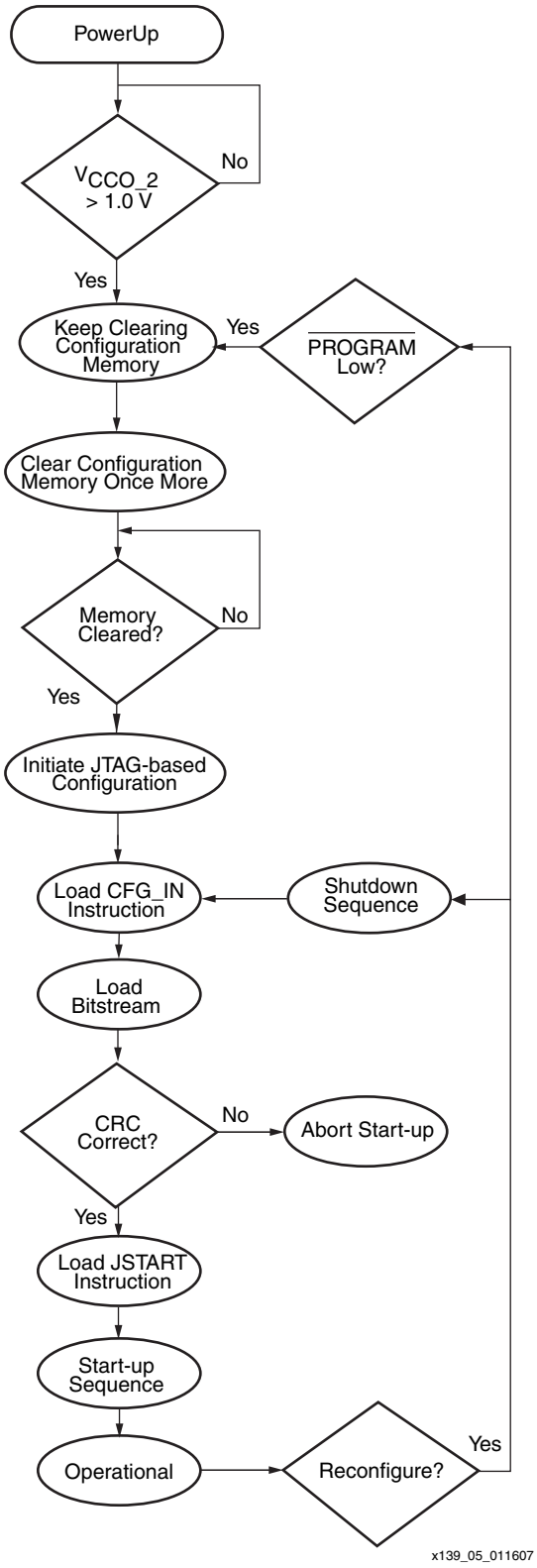


Figure 5: **Device JTAG Configuration Flow Diagram**

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.

Multiple Device Configuration

It is possible to configure multiple Virtex devices in a chain. The devices in the JTAG chain are configured one at a time. The multiple device configuration steps are described generally to be applied to any size chain. Ensure the bitstream is generated with the JTAG clock option:

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

Refer to Figure 1 for the following TAP controller steps:

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.
2. Load the CFG_IN instruction into the target device (and BYPASS in all other devices.)
3. For a chain of Virtex devices, shift in leading zeros before the bitstream if the value of N is not equal to zero. Use the following equation to determine the number of leading zeros for each bitstream. M is the targeted device position in the chain. As shown in Figure 6, the first device position in the chain is zero. N is the number of zeros required. Mod is the modulus operation.

$$N = 32 - mod\left(\frac{M}{32}\right)$$

$$N = 32 - M \quad for \quad (M \leq 32)$$

Example:

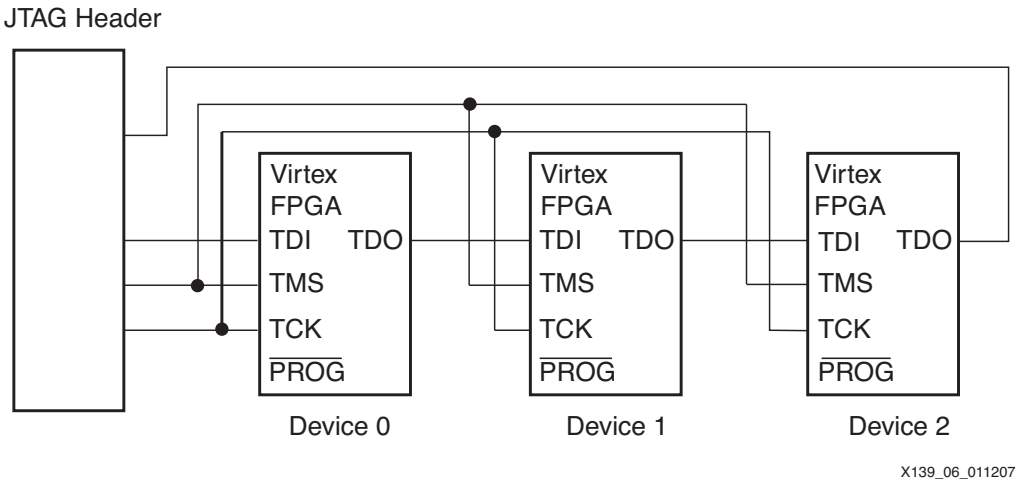
The third device, position 2, in Figure 6 requires 30 leading zeros.

$N = 32 - 2 \quad N = 30 \quad number\,of\,leading\,0s$

The following example is for position 47:

$$N = 32 - mod\left(\frac{47}{32}\right) \quad N = 17 \; number\,of\,leading\,0s$$

4. Go through RTI (RUN-TEST/IDLE).
5. Repeat steps 2 through 4 for each successive device, repeat step 2 and step 4.
6. Load the JSTART command into all devices.
7. Go to SHIFT-DR and clock TCK 12 times.
8. All devices are active at this point.



Notes:

1. The $\overline{\text{PROG}}$ pin should be deasserted during JTAG operations.

Figure 6: **Boundary-Scan Chain of Devices**

Reconfiguring through Boundary-Scan

Note: Refer to **XAPP058** for the recommended embedded solution.

Virtex FPGAs support reconfiguration through Boundary-Scan. The ability to reconfigure a Virtex FPGA with a different design enables an FPGA to perform multiple functions during different periods in time.

FPGA reconfiguration comprises an initial reconfiguration setup procedure that prevents internal contention followed by the appropriate configuration sequence from the "Configuring through Boundary-Scan" section. (The appropriate configuration sequence from the Configuring Through Boundary-Scan section depends on whether the Boundary-Scan chain comprises a single device or multiple devices.) If the initial reconfiguration setup procedure is not performed, then internal contention can occur as a new configuration overwrites an existing configuration within the FPGA.

Either of the two following initial configuration setup methods prevent internal contention.

- One method is a pulse to the $\overline{\text{PROG}}$ pin. A pulse to the $\overline{\text{PROG}}$ pin causes the FPGA to clear the entire internal configuration memory of the FPGA.
- The alternate method is to perform a shutdown sequence. The shutdown sequence places the FPGA in a safe state for reconfiguration. When the FPGA is to be only partially reconfigured, the shutdown sequence is preferred because it allows the unchanged portion of the pre-existing configuration to remain within the configuration memory.

The shutdown sequence is as follows. (For details on internal registers, see application note **XAPP138**.)

1. Load the CFG_IN instruction into the JTAG instruction register. Next, go to the SHIFT-DR state.
2. In the SHIFT-DR state, shift in the following sequences in steps 2 through 4. (This writes the COR (Configuration Option Register) with the SHUTDOWN bit = 1. It also indicates that the startup sequencer should perform a shutdown sequence.) The most significant bit (MSB) is the left bit and it is shifted in first.

```
0011 0000 0000 0001 0010 0000 0000 0001
-> Header: Write to COR
0000 0000 1010 0000 1011 1111 0011 1101
-> COR data sets SHUTDOWN = 1
```

3. Write the START command to the CMD (Command) register by shifting in the following data:

```
0011 0000 0000 0000 1000 0000 0000 0001
-> Header: Write to CMD
0000 0000 0000 0000 0000 0000 0000 0101
-> START command
```

4. Write the precalculated Cyclic Redundancy Check (CRC) value to the CRC register, or write the Reset CRC Register (RCRC) command to the CMD register as shown:

```
0011 0000 0000 0000 1000 0000 0000 0001
-> Header: Write to CMD
0000 0000 0000 0000 0000 0000 0000 0111
-> RCRC command
0000 0000 0000 0000 0000 0000 0000 0000
-> flush pipe
```

5. Now proceed to the SHIFT-IR and load the JTAG JSTART command into the instruction register.
6. Go to the SHIFT-DR and clock TCK 13 times to clock the shutdown sequence.
7. Proceed to the SHIFT-IR state and load the CFG_IN instruction again.
8. In the SHIFT-DR state, shift in the sequences in steps 8 and 9. This writes the AGHIGH command to the CMD register to assert the GHIGH_B signal. This prevents contention while writing configuration data.

```
0011 0000 0000 0000 1000 0000 0000 0001
-> Header: Write to CMD
0000 0000 0000 0000 0000 0000 0000 1000
-> AGHIGH command asserts GHIGH_B
```

9. Write the COR with SHUTDOWN = 0 and go to RTI (RUN-TEST/IDLE) by shifting in the following sequence:

```
0011 0000 0000 0001 0010 0000 0000 0001
-> Header: Write to COR
0000 0000 1010 0000 0011 1111 1111 1111
-> COR data sets SHUTDOWN = 0

0011 0000 0000 0000 1000 0000 0000 0001
-> Header: Write to CMD
0000 0000 0000 0000 0000 0000 0000 0101
-> Header: Start command
0011 0000 0000 0000 1000 0000 0000 0001
-> Write to CMD
0000 0000 0000 0000 0000 0000 0000 0111
-> RCRC command
0000 0000 0000 0000 0000 0000 0000 0000
-> flush pipe
```

10. Proceed to the SHIFT-IR state and load the JTAG JSTART instruction.
11. Go to the SHIFT-DR and clock TCK 13 times to clock the simulation sequence.
12. Go to the SHIFT-IR state and load the CFG_IN instruction again.
13. In the SHIFT-DR state, shift in the following sequence steps 13 through 15. (This writes the Configuration Option Register (COR) with the shutdown bit = 1. It also indicates that the startup sequencer should perform a shutdown sequence.)

```
0011 0000 0000 0001 0010 0000 0000 0001
-> Header: Write to COR
0000 0000 1010 0000 1011 1111 0010 1101
-> COR data sets SHUTDOWN = 1
```

14. Write Start command to command (CMD) register by shifting in the following data:

```
0011 0000 0000 0000 1000 0000 0000 0001
-> Header: Write to CMD
0000 0000 0000 0000 0000 0000 0000 0101
-> Start command
```

15. Write the RCRC command to the CMD register as shown:

```
0011 0000 0000 0000 1000 0000 0000 0001
-> Header: Write to CMD
0000 0000 0000 0000 0000 0000 0000 0111
-> RCRC command
0000 0000 0000 0000 0000 0000 0000 0000
-> flush pipe
```

16. Proceed to the SHIFT-IR and write the JSTART command into the Instruction register.
17. Go to SHIFT-DR and clock TCK 13 times to clock the shutdown sequence.
18. Go to the SHIFT-IR state and load the CFG_IN instruction.
19. Enter the SHIFT-DR state and shift in the bitstream data.
20. Proceed to SHIFT-IR and load the JSTART command into the instruction register.
21. Proceed to SHIFT-IR and load the BYPASS instruction.

Debugging Configuration

To verify successful configuration, there are several options. Some of the most helpful verification steps include using the TAP pins and the readback command. Using the Virtex TAP controller and status pins is discussed first.

When using the TAP controller pins, TDO is driven only in the SHIFT-DR and SHIFT-IR state. If the output of the TDO can be changed by using an external pull-up, the TAP is not in SHIFT-IR or SHIFT-DR. If the TAP can be controlled precisely, use this to test the application.

In JTAG configuration, the status pin (DONE) functions the same as in the other configuration modes. The DONE pin can be monitored to determine if a bitstream has been completely loaded into the device. If DONE is Low, the entire bitstream has not been sent or the start-up sequence is not finished. If DONE is High, the entire bitstream has been received correctly. When the FPGA detects a bitstream CRC error, DONE remains Low and INIT is driven Low.

If the DONE pin is not asserted High, there are several possible reasons.

1. The bitstream option `bitgen -g startupclk:jtagclk` described in the section "Software Support and Data Files" was not used.
2. The JSTART instruction was not issued.
3. There was an error in the bitstream.

In addition to the external pin monitoring, an internal test can be conducted. The second method includes the following steps to capture the contents of the internal device status register:

1. Move the TAP to the Test-Logic-Reset (TLR) state.
2. Go to the SHIFT-IR state and load in the CFG_IN instruction.
3. Go to the SHIFT-DR state and shift in the following 64-bit pattern with the MSB (left most bit), shifted in first.

```
0010 1000 0000 0000 1110 0000 0000 0001
-> Header: Read Status Register
0000 0000 0000 0000 0000 0000 0000 0000
-> flush pipe
```

4. After shifting in this pattern, load the CFG_OUT instruction in the SHIFT-IR state.
5. Move to the SHIFT-DR and clock TCK 32 times while reading TDO. The data seen on TDO is the contents of the status register. The last bit out is a one if a CRC error occurred. If successful, it should be:

```
0000 0000 0000 0000 0111 1111 0101 1110
                    CRC Bit
```

The device status register also gives the status of the DONE and INIT signals. For information on the status register, refer to Figure 7 and the application note **XAPP151, "Virtex Series Configuration Architecture User Guide."**

Refer to the “ISP checklist” **XAPP104. "A Quick JTAG ISP Checklist,"** for general techniques on how to reduce noise and signal degradation in JTAG chains.

LOCK				IN_ERROR	GTS_CFG	GWE_B	GSR_B	GHIGH_B	MODE			INIT	DONE																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Notes:
1. An *X* in a bit field indicates that the value is variable and must be set.
2. Heavy vertical lines are used to separate fields. Light vertical lines separate nibbles in the word.

Figure 7: **Status Register Fields**

Readback Instructions

Readback is available through both the Boundary-Scan and SelectMAP™ interfaces. It is the process of verifying that the current configuration data in the device is correct by reading out the data in the internal configuration memory.

Verify Readback

Readback verification is not unique to JTAG. This section discusses Boundary-Scan specific steps to perform a readback operation. For detailed information on verify/readback not specific to JTAG, refer to **XAPP138**.

1. Load in the CFG_IN instruction into the JTAG IR and then go to the SHIFT-DR.
2. Shift in a packet to write the starting frame address into the Frame Address Register (FAR). For a full-chip readback, this is frame zero of the CLB column zero.
3. Shift in a packet to write the Read Configuration Data (RCFG) command into the CMD register.
4. Shift in a packet header requesting a read of the Frame Data Output Register (FDRO). The word count should reflect the number of frames desired to be read.
5. Shift in an extra 32 bits. These are to flush the pipeline through the packet processor. Then go back to RTI.
6. Load the CFG_OUT instruction into the JTAG IR and then go to the Shift-DR.
7. Clock TCK and read TDO. There is one word plus one frame of garbage before the readback data appears.
8. To read the block RAM data, repeat steps one through seven by substituting the appropriate block RAM address for the starting CLB address. It is recommended that the system be halted prior to block RAM readback.

Software Support and Data Files

The current version of iMPACT software that supports the Virtex devices is Xilinx software version 2.1i. The Xilinx tool set includes the iMPACT software to program and get the Virtex IDCODE. For test vectors EXTEST or INTEST, or to utilize other JTAG features present in the device, see **www.xilinx.com** for third party Boundary-Scan software tools.

Note: To perform any configuration operations through JTAG, the bitgen option should be set for the JTAG clock option:

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

For Readback operations, this option must be used:

```
bitgen -w -l -m -g readback
```

Readback is supported in the current version of iMPACT software, but only in the form of the configuration verify operation.

Revision History

The following table shows the revision history for this document.

Date	Revision #	Activity
10/28/99	1.0	Initial release
12/8/99	1.1	Fixed error on AGHIGH command on page 11.
2/18/00	1.2	Reformatted to one column.

Date	Revision #	Activity
2/20/02	1.3	Added Virtex-E Extended Memory devices to Table 4. Revised Manufacturers ID in Figure 3.
04/03/02	1.4	Updated JTAG Programmer to iMPACT software.
05/29/03	1.5	Updated "Reconfiguring through Boundary-Scan," and "Debugging Configuration."
09/12/03	1.6	Made edits to step 9, page 15, step 10, page 16, step 13, page 16, and Figure 7.
02/14/07	1.7	Changed text in "Boundary-Scan Register." Old Figure 3 changed to Table 4, and the numbers of all subsequent figures and tables changed. Changed text in "USER1, USER2 Registers." Changed figure and figure title for Figure 4. Changed Figure 5. Changed numbering for steps in "Multiple Device Configuration" and clarified text. Changed text in "Reconfiguring through Boundary-Scan." Changed text in "Debugging Configuration."