XILINX®

Chapter 3

Configuration

Summary

This chapter covers the following topics:

- Introduction
- Configuration Solutions
- Master Serial Programming Mode
- Slave Serial Programming Mode
- Master SelectMAP Programming Mode
- Slave SelectMAP Programming Mode
- JTAG/ Boundary Scan Programming Mode
  - Boundary-Scan for Virtex-II Devices Using IEEE Standard 1149.1
  - Boundary-Scan for Virtex-II Devices Using IEEE Standard 1532
- Configuration Details
- Readback

Introduction

Virtex-II devices are configured by loading application-specific configuration data into internal memory. Configuration is carried out using a subset of the device pins, some of which are dedicated, while others can be reused as general-purpose inputs and outputs after configuration is complete.

Depending on the system design, several configuration modes are selectable via mode pins. The mode pins M2, M1, and M0 are dedicated pins. An additional pin, HSWAP_EN, is used in conjunction with the mode pins to select whether user I/O pins have pull-up resistors during configuration. By default, HSWAP_EN is tied High (internal pull-up resistor), which shuts off pull-up resistors on the user I/O pins during configuration. When HSWAP_EN is tied Low, the pull-up resistors are on and therefore, the user I/Os have pull-up resistors during configuration.

Other dedicated pins are:

- CCLK - the configuration clock pin
- DONE - configuration status pin
- TDI, TDO, TMS, TCK - boundary-scan pins
- PROG_B - configuration reset pin

Depending on the configuration mode selected, CCLK can be an output generated by the Virtex-II FPGA or an input accepting externally generated clock data. For correct operation, these pins require a V_{CCAUX} of 3.3V to permit low-voltage transistor-to-transistor logic (LVTTL) operations.

All dual-function configuration pins are contained in banks 4 and 5. Bank 4 contains pins used in serial configuration modes, and banks 4 and 5 contain pins used for SelectMAP modes.

A persist option is available, which can be used to force pins to retain their configuration function even after device configuration is complete. If the persist option is not selected, then the configuration pins with the exception of CCLK, PROG_B, and DONE can be used for user I/O in normal operation. The persist option does not apply to boundary-scan related pins. The persist feature is valuable in applications that employ partial reconfiguration, dynamic reconfiguration, or readback.

Configuration Modes

Virtex-II supports the following configuration modes:

- Master-Serial
- Slave-Serial (default)
- Master SelectMAP
- Slave SelectMAP
- Boundary-Scan (IEEE 1532 and IEEE 1149)

Table 3-1 shows Virtex-II configuration mode pin settings.

Table 3-1: **Virtex-II Configuration Mode Pin Settings**

Configuration Mode[1]	M2	M1	M0	CCLK Direction	Data Width	Serial Dout[2]
Master Serial	0	0	0	Out	1	Yes
Slave Serial	1	1	1	In	1	Yes
Master SelectMAP	0	1	1	Out	8	No
Slave SelectMAP	1	1	0	In	8	No
Boundary Scan	1	0	1	N/A	1	No

Notes:

1. The HSWAP_EN pin controls the pullups. Setting M2, M1, and M0 selects the configuration mode, while the HSWAP_EN pin controls whether or not the pullups are used.
2. Daisy chaining is possible only in modes where Serial Dout is used. For example, in SelectMAP modes, the first device does NOT support daisy chaining of downstream devices.

Table 3-2 lists the total number of bits required to configure each device:

Table 3-2: **Virtex-II Bitstream Lengths**

Device	Total Number of Configuration Bits (including header)
XC2V40	360,096
XC2V80	635,296
XC2V250	1,697,184
XC2V500	2,761,888
XC2V1000	4,082,592
XC2V1500	5,659,296
XC2V2000	7,492,000
XC2V3000	10,494,368
XC2V4000	15,659,936
XC2V6000	21,849,504
XC2V8000	29,063,072

Configuration Process and Flow

The configuration process involves loading the configuration bitstream into the FPGA using the selected mode. There are four major phases in the configuration process:

- Clearing Configuration Memory
- Initialization
- Loading Configuration Data
- Device Startup

Figure 3-1 illustrates the configuration process flow.

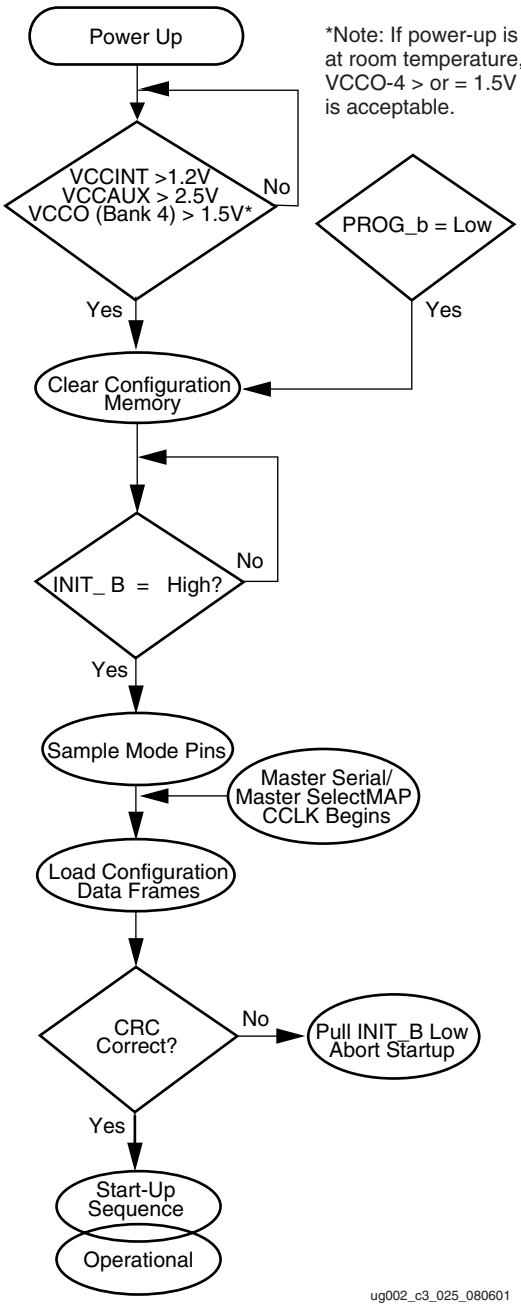

Figure 3-1: **Configuration Process**

Power Up

The V_{CCINT} power pins must be supplied with a 1.5V source. (Refer to the **Virtex-II Data Sheet (DS031)** for DC characteristics.) The IOB voltage input for Bank 4 (V_{CCO_4}) and the auxiliary voltage input (V_{CCAUX}) are also used as a logic input to the Power-On-Reset (POR) circuitry. Even if this bank is not being used, V_{CCO_4} must be connected to a 1.5V or greater source.

Clearing Configuration Memory

In the memory clear phase, non-configuration I/O pins are 3-stated with optional pull-up resistors. The INIT_B and DONE pins are driven Low by the FPGA, and the memory is cleared. After PROG_B transitions High, memory is cleared twice and initialization can begin.

The INIT_B pin transitions High when the clearing of configuration memory is complete. A logic Low on the PROG_B input resets the configuration logic and holds the FPGA in the clear configuration memory state. When PROG_B is released, the FPGA continues to hold INIT_B Low until it has completed clearing all of the configuration memory. The minimum Low pulse time for PROG_B is defined by the $T_{PROGRAM}$ timing parameter. There is no maximum value. The power-up timing of configuration signals is shown in Figure 3-2 and the corresponding timing characteristics are listed in Table 3-3.

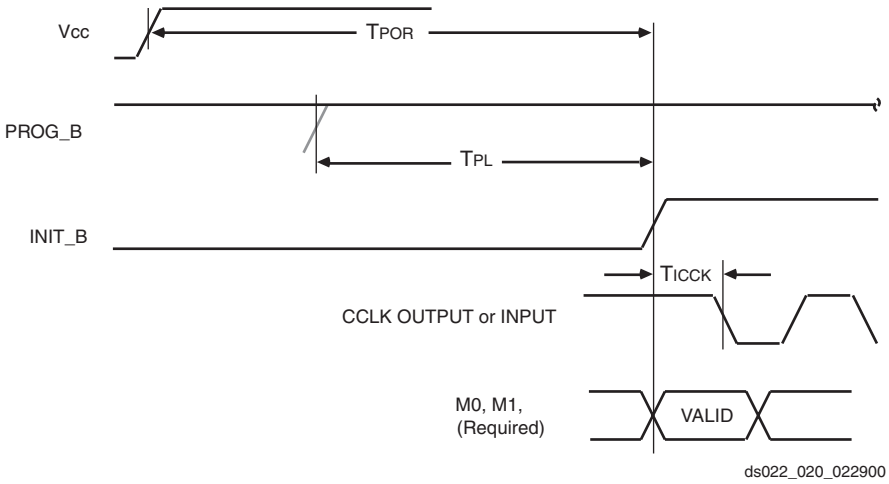

Figure 3-2: **Power-Up Timing Configuration Signals**

Table 3-3: **Power-Up Timing Characteristics**

Description	Symbol	Value	Units
Program Latency	T_{PL}	T_{PL} (2V8000)	4 µs per frame max
Power-on-Reset	T_{POR}	T_{PL} +2	ms, max
CCLK (output) Delay	T_{ICCK}	0.5	µs, min
		4.0	µs, max
Program Pulse Width	$T_{PROGRAM}$	300	ns, min

Initialization

For the initialization phase, the INIT_B pin is released, the mode pins are sampled, the appropriate pins become active, and the configuration process begins. It is possible to delay configuration by externally holding INIT_B Low.

Delaying Configuration

The INIT_B pin can also be held Low externally to delay configuration of the FPGA. The FPGA samples its mode pins on the rising edge of INIT_B. After INIT_B transitions to High, configuration can begin. No additional time-out or waiting periods are required, but configuration does not need to commence immediately after the transition of INIT_B. The configuration logic does not begin processing data until the synchronization word from the bitstream is loaded.

Loading Configuration Data

Once configuration begins, the target FPGA starts to receive data frames. Cyclic Redundancy Checking (CRC) is performed before and after the last data frame. CRC is also automatically checked after each block write to an internal data register (FDRI). If the CRC checks prove valid, the device start-up phase can begin.

If the CRC values do not match, INIT_B is asserted Low to indicate that a CRC error has occurred, startup is aborted, and the FPGA does not become active.

To reconfigure the device, the PROG_B pin should be asserted to reset the configuration logic. Recycling power also resets the FPGA for configuration. For more information on CRC calculation, see “Cyclic Redundancy Checking Algorithm” on page 296.

The details of loading configuration data in each of the five modes are discussed in the following sections:

- "Master Serial Programming Mode" on page 261
- "Master SelectMAP Programming Mode" on page 264
- "Slave Serial Programming Mode" on page 262
- "Slave SelectMAP Programming Mode" on page 266
- "JTAG/ Boundary Scan Programming Mode" on page 270

Device Startup

Device startup is a transition phase from the configuration mode to normal programmed device operation. Although the order of the start-up events are user programmable via software, the default sequence of events is as follows:

Upon completion of the start-up sequence, the target FPGA is operational.

The Start-Up Sequencer is an 8-phase sequential state machine that counts from phase 0 to phase 7. (See Figure 3-3.)

The Start-Up Sequencer performs the following tasks:

- Release the DONE pin.
- Negate GTS, activating all of the I/Os.
- Assert GWE, allowing all RAMs and flip-flops to change state.
- Assert EOS. The End-Of-Start-Up flag is always set in phase 7. This is an internal flag that is not user accessible.

BitGen options control the order of the Start-Up Sequence. The default Start-Up Sequence is the bold line in Figure 3-3. The Start-Up Sequence can also be stalled at any phase until either DONE has been externally forced High, or a specified DCM or DCI has established LOCK. For details, see Appendix B, "BitGen and PROMGen Switches and Options."

At the cycle selected for the DONE to be released, the sequencer always waits in that state until the DONE is externally released. However, this does not delay the GTS or GWE if they are selected to be released prior to DONE. Therefore, DONE is selected first in the sequence for default settings.

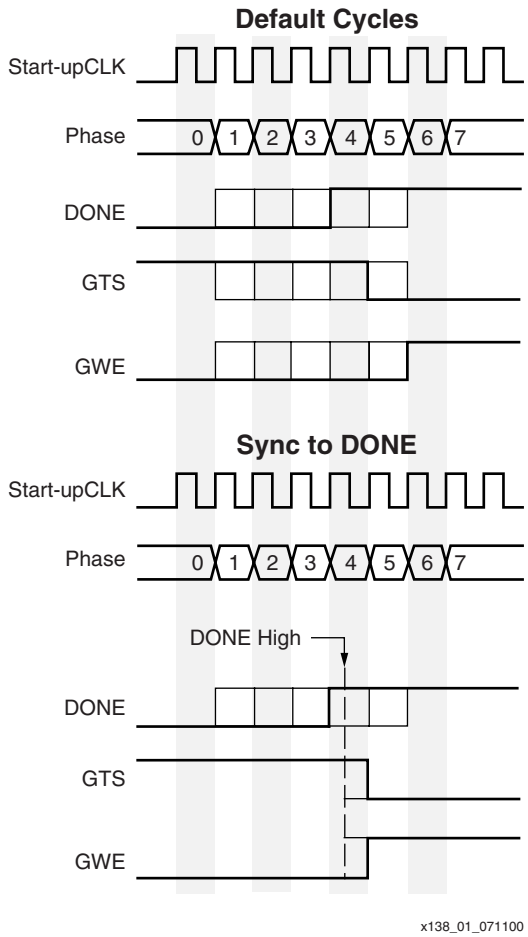

Figure 3-3: **Default Start-Up Sequence**

Configuration Pins

Certain pins in the FPGA are designated for configuration and are listed in Table 3-4. Some pins are dedicated to the configuration function and others are dual-function pins that can be user I/O after configuration.

Table 3-4: **Configuration Pins**

Name	Direction	Driver Type	Description
Dedicated Pins			
CCLK	Input/Output	Active	Configuration clock. Output in Master mode.
PROG_B	Input		Asynchronous reset to configuration logic.
DONE	Input/Output	Active/ Open-Drain	Configuration status and start-up control.
M2, M1, M0	Input		Configuration mode selection.
HSWAP_EN	Input		I/O pullups during configuration.
TMS	Input		Boundary Scan Mode Select.
TCK	Input		Boundary Scan Clock.
TDI	Input		Boundary Scan Data Input.
TDO	Output	Active	Boundary Scan Data Output.
Dual Function Pins			
DIN (D0)	Input/Output	Active Bidirectional	Serial configuration data input/SelectMAP readback data output.
D1:D7	Input/Output	Active Bidirectional	SelectMAP configuration data input, readback data output.
CS_B	Input		Chip Select (SelectMAP mode only).
RDWR_B	Input		Active Low write select, read select (SelectMAP mode only).
BUSY/DOUT	Output	Active	Serial configuration data output for serial daisy-chains (active).
INIT_B	Input/Output	Open-Drain	Delay configuration, indicate configuration error.

Mixed Voltage Environments

Virtex-II devices have separate voltage sources:

- V_{CCINT} = 1.5V powers the internal circuitry.
- V_{CCAUX} = 3.3V powers critical resources in the FPGA.
- V_{CCO} (1.5, 1.8, 2.5, or 3.3V) powers the IOB circuitry.

SelectI/O-Ultra is separated into eight banks of I/O groups. Each bank can be configured with one of several I/O standards. Refer to the Design Considerations section for I/O banking rules and available I/O standards. Before and during configuration, all I/O banks are set for the LVTTL standard, which requires an output voltage (V_{CCO}) of 3.3V for normal operation.

If V_{CCO} is less than 3.3V on banks 4 and 5, serial and SelectMAP configuration modes might have a lower frequency. (See Table 3-5).

Table 3-5: **Configuration Modes and V_{CCO} Voltages**

Configuration Mode	Pins Used	V_{CCO_4}	V_{CCO_5}
JTAG	Dedicated Pins	not a concern	not a concern
Serial	Dedicated Pins plus DOUT, DIN, and INIT	3.3V	not a concern
SelectMAP	Dedicated Pins plus dual-function pins	3.3V	3.3V

Notes:

1. If less than 3.3V ($V_{CCO\_4/5}$ = 2.5V), the configuration frequency might be as low as half of the typical frequency.

All dedicated configuration pins are powered by V_{CCAUX}. All dual-function configuration pins are located within banks 4 and 5. As described under Configuration Process and Flow, the V_{CCO_4} input voltage is used as a logic input to the power-on-reset (POR) circuitry.

For JTAG configuration mode, JTAG inputs are independent of V_{CCO} and work between 2.5V and 3.3V TTL levels (VIL max = .8V, VIH min = 2.0V). The JTAG input pins are 3.3V tolerant. The JTAG output (TDO) is an open-drain output and must be pulled up to the appropriate voltage level (typically 3.3V) through an external resistor. The value of the external pullup resistor depends on the capacitive loading on the TDO pin and the operating frequency, but it should not be less than 200 ohms. The optimal TDO pullup value can be determined through IBIS simulation.

For serial configuration mode, V_{CCO_4} pins require a 3.3V supply for output configuration pins to operate normally. In serial mode, all of the configuration pins are in bank 4.

For SelectMAP configuration mode, V_{CCO_4} and V_{CCO_5} pins require a 3.3V supply for output configuration pins to operate normally. In SelectMAP mode, all of the configuration pins are in banks 4 and 5.

If the Virtex-II device is being configured in serial or SelectMAP mode, and the desired I/O standard in banks 4 and 5 is for a voltage other than 3.3V, then V_{CCO_4} and V_{CCO_5} (SelectMAP only) must have 3.3V supplies at configuration, and they can be switched to the desired voltage after configuration is complete.

Configuration Solutions

Several configuration solutions are available to support Virtex-II, each targeted to specific application requirements. Guidance and support (application notes, reference designs, and so forth) is also available for designers looking to develop and implement their own configuration solution for Virtex FPGAs.

System Advanced Configuration Environment (System ACE™) Series

The System ACE series of configuration solutions offers a system-level configuration manager for designers using multiple FPGAs or FPGAs requiring multiple bitstreams. This solution combines standard industry Flash storage with Xilinx-designed configuration control. Features common to the entire System ACE family include:

- Support for multiple bitstreams
- Built-in support for embedded processors in FPGAs
- Support for reconfiguring, updating, or debugging systems over a network
- Built-in system interface
- Scalability (density) and re-useability (across many designs)
- Centralization of configuration control for reduced board space and simpler debugging
- Use of excess storage capacity for non-configuration, system storage

System ACE CF

System ACE CF (CompactFlash™) solution combines a standard CompactFlash Association (CFA) Type-I or Type-II memory module (CompactFlash or 1" disk drive) with a Xilinx-designed ACE Controller™ configuration control chip. See Figure 3-4.

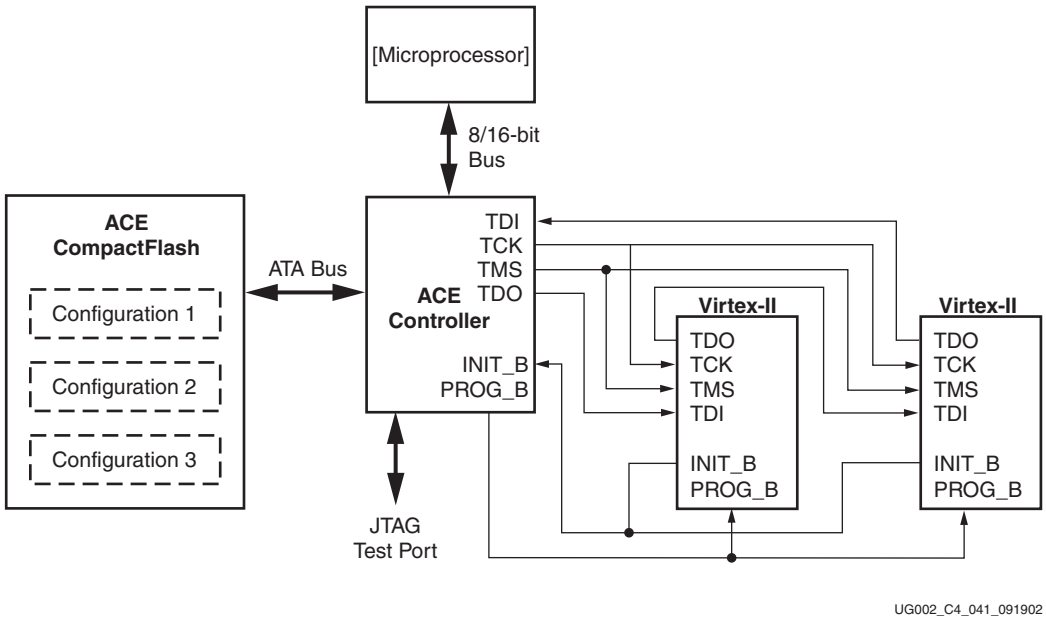

Figure 3-4: **System ACE CompactFlash and Controller**

The CompactFlash card stores an unlimited number of bitstreams and ranges in density from 128 Mb to 3 Gb. This card is capable of storing one large bitstream or several smaller bitstreams. If several bitstreams are used, the system can be set up so that individual bitstreams are callable as needed, allowing for dynamic reconfiguration of the Virtex-II device and other Xilinx FPGAs in the JTAG chain.

The ACE Controller drives bits through the FPGA JTAG chain and has three other ports:

- A port for interfacing with a microprocessor, a network, or a MultiLINX cable
- A port for interfacing with the CompactFlash card
- A port that provides access to the FPGA JTAG chain for FPGA testing or configuration via automatic test equipment or via desktop or third-party programmers

For further information on any System ACE product, visit www.xilinx.com.

System ACE Multi-Package Module (MPM)

System ACE MPM is a multi-package module consisting of a packaged standard Flash from AMD, a packaged FPGA, and a packaged configuration PROM, all in a 388-pin BGA package. The Flash stores configuration and other data, while the FPGA acts as an advanced configuration controller and is configured by the PROM. This solution provides high density and high-speed configuration capability in a single package, helping to simplify the design and manufacturing process. It is available in 16-Mbit, 32-Mbit, and 64-Mbit densities.

System ACE Soft Controller (SC)

System ACE SC is a downloadable version of the configuration controller found in System ACE MPM; versions are provided that support various standard Flash interfaces. System ACE SC provides all of the features of System MPM without the Single Package. It allows designers to use the Flash memory already in their system to store configuration data. The System ACE SC controller is available free of charge in the form of a PROM file that can be downloaded from the System ACE website. This pre-engineered solution is implemented by connecting up to four Flash chips on a board to an FPGA that will be used as a configuration controller and then downloading the controller file into a PROM. Figure 3-5 describes the controller for both System ACE MPM and System ACE SC.

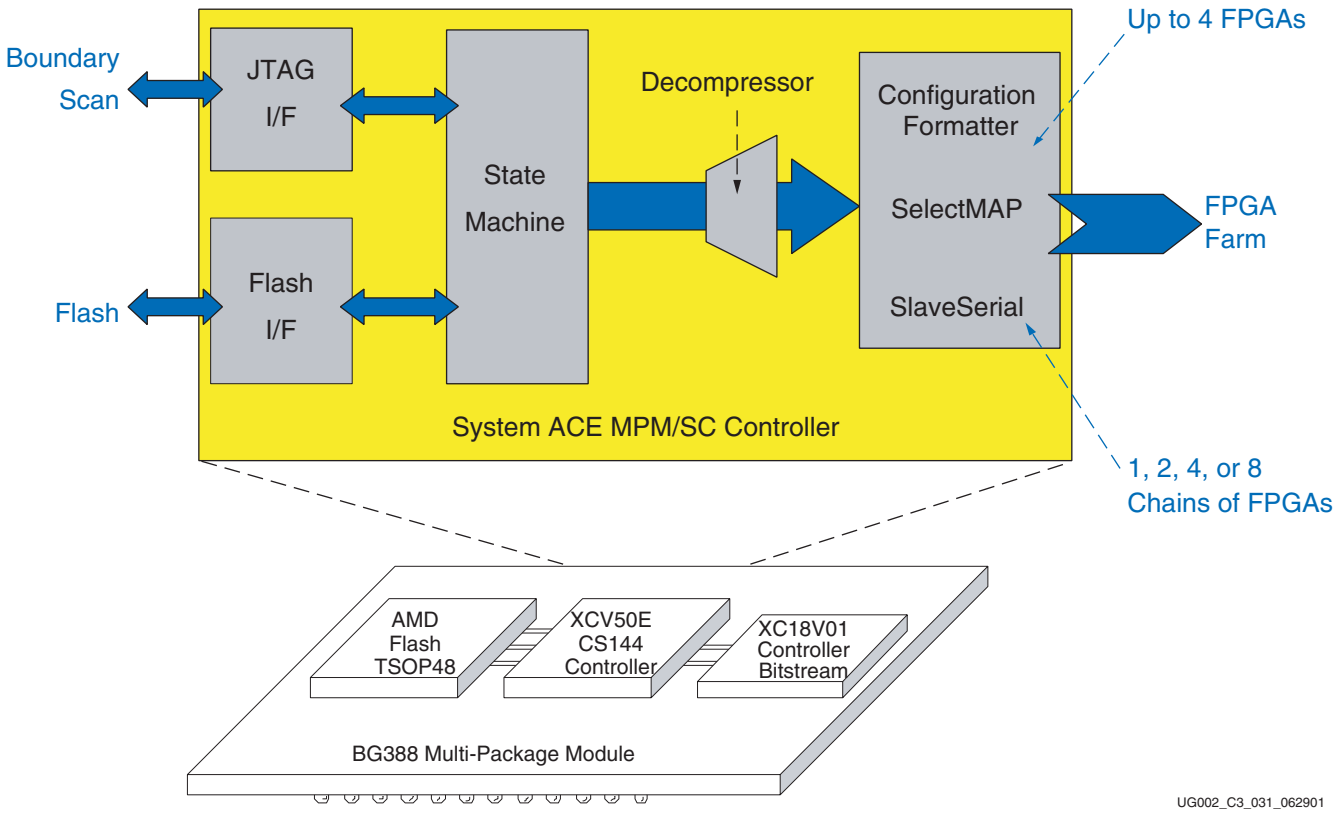

Figure 3-5: **System ACE MPM/SC Controller**

System ACE MPM and System ACE SC have these unique features:

- High speed configuration up to 154 Mb/sec
- Support for both SelectMAP (8-bit) (see Figure 3-6) and Slave Serial (1-bit) (see Figure 3-7) configuration
- Configuration of multiple FPGAs in parallel
- Bitstream compression for increased storage capability
- Storage of up to 8 different bitstreams

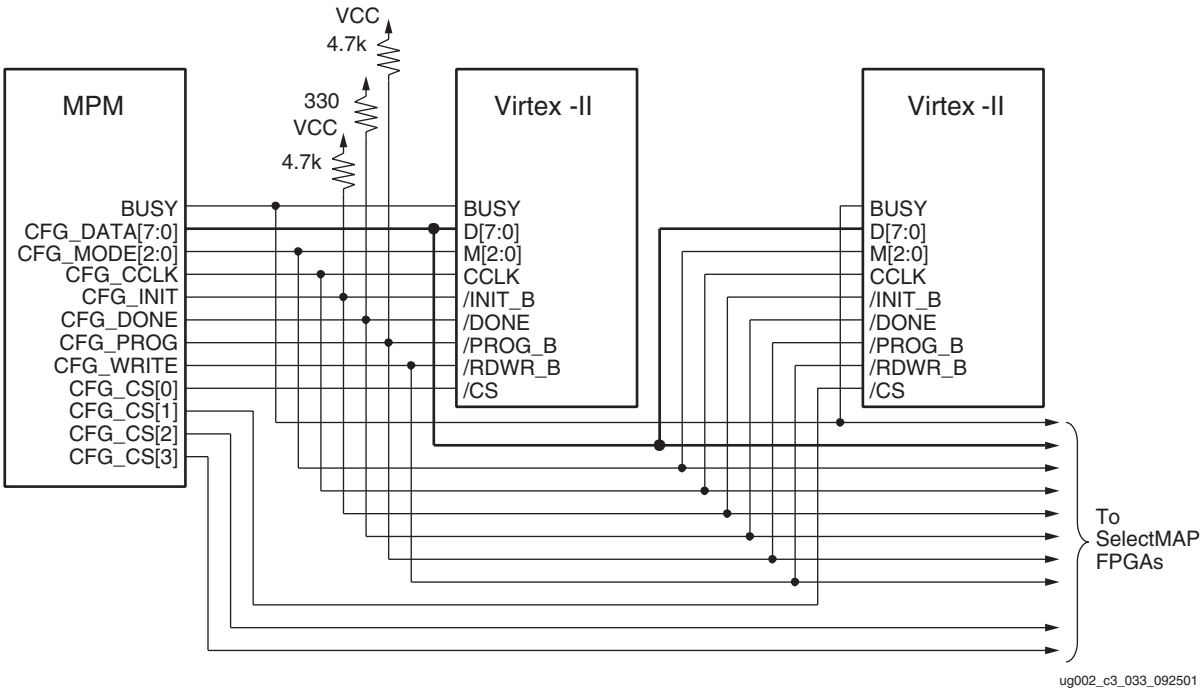

Figure 3-6: **SelectMAP (8-bit) Configuration**

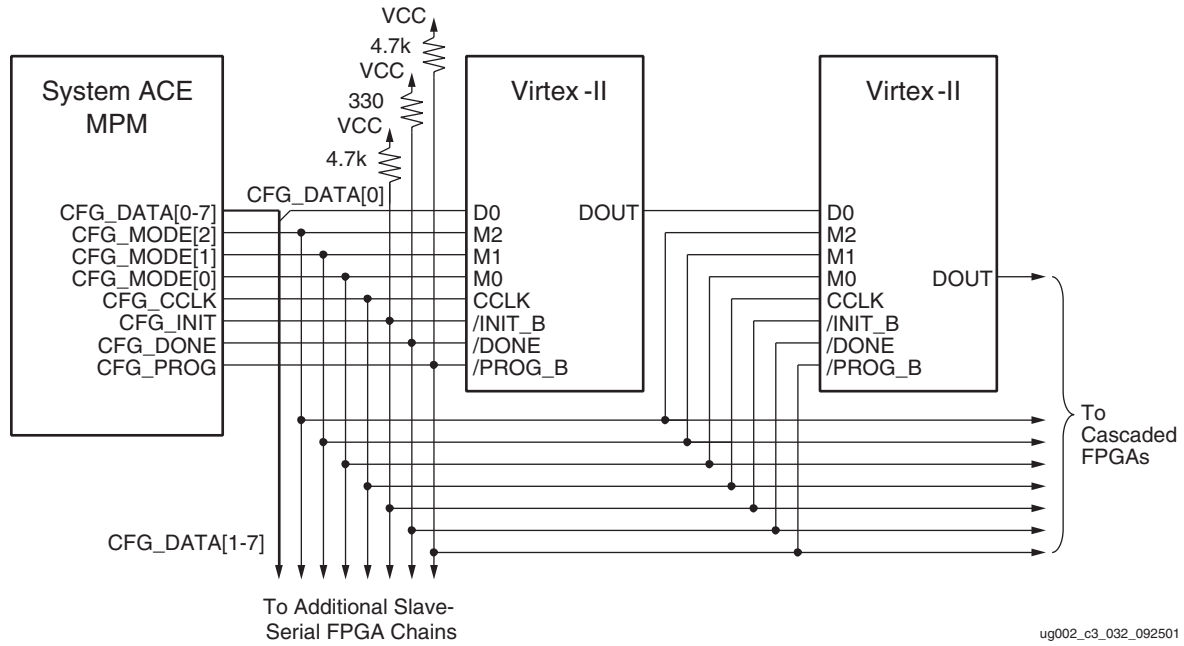

Figure 3-7: **Slave Serial (1-bit) Configuration**

Configuration PROMs

Using XC18V00 PROMs

The XC18V00 family of Flash in-system programmable (ISP) configuration PROMs offers the flexibility of re-programmability and multiple package offerings, combined with both serial and SelectMAP FPGA configurability. This family is programmable using Xilinx iMPACT software and ranges in density from 256 Kb to 4 Mb; these PROMs can also be cascaded to support larger bitstreams.

The 18V00 family offers data throughput rates of up to 264 Mb/s. It is also capable of triggering FPGA reconfiguration via a JTAG command. The parts can be programmed via cable, HW-130, or standard third party programmers. The XC18V00 PROMs are available in SO20, PC20, VQ44, and PC44 packages. Refer to Appendix C, "XC18V00 Series PROMs" for the latest version of the XC18V00 PROMs data sheet, as well as package diagrams for the entire PROM family. See Table 3-6 to determine which PROMs go with which Virtex-II FPGAs.

Using XC17V00 PROMs

The XC17V00 family of one-time programmable (OTP) PROMs provides a proven, low-cost, compact, and pre-engineered configuration solution. Ranging from 1 Mb to 16 Mb, this family is also the PROM density leader; it can also be daisy-chained to support larger bitstreams. This family supports serial configuration of Virtex-II FPGAs; in addition, the XC17V08 and XC17V16 support SelectMAP configuration modes.

The XC17V00 family can be used for stabilized designs that are in a high-volume production flow and/or for designs requiring a low-cost solution. XC17V00 PROMs can be programmed either by using the HW-130 or by using a variety of third-party programmers. The XC17V00 PROMs are available in VO8, SO20, PC20, VQ44, and PC44 packages. Data sheets for PROMs are available at www.xilinx.com. See Table 3-6 to determine which PROMs go with which Virtex-II FPGAs and see Appendix C, "XC18V00 Series PROMs" for package diagrams.

Flash PROMs With a CPLD Configuration Controller

Some designers prefer to leverage existing Flash memory in their system to store the configuration bitstreams. A small CPLD-based configuration controller can provide the mechanism to access the bitstreams in the FLASH and deliver them quickly to Virtex-II devices. The following application notes describe the details for a serial or SelectMAP configuration architecture using FLASH memories and CPLDs:

- XAPP079: *Configuring* Xilinx *FPGAs Using an XC9500 CPLD and Parallel PROM* (available on www.xilinx.com) describes an architecture that configures a chain of Virtex-II devices using Master-Serial mode. See Figure 3-8 for an example of FPGA configuration using a CPLD and a parallel PROM.

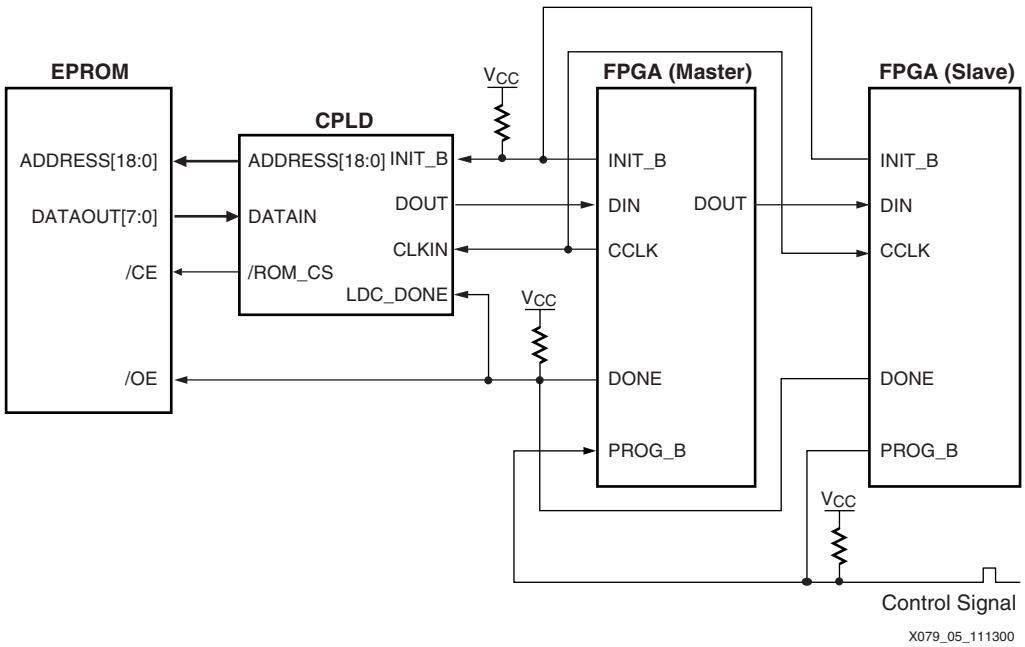

Figure 3-8: **Configuring Virtex-II Using a CPLD and Parallel PROM**

- XAPP137: *Configuring Virtex FPGAs From Parallel EPROMs With a CPLD* (available on www.xilinx.com) describes an architecture that configures one or more Virtex-II devices using the Slave SelectMAP mode. See Figure 3-9 for an example of FPGA configuration using a CPLD and a parallel EPROM.

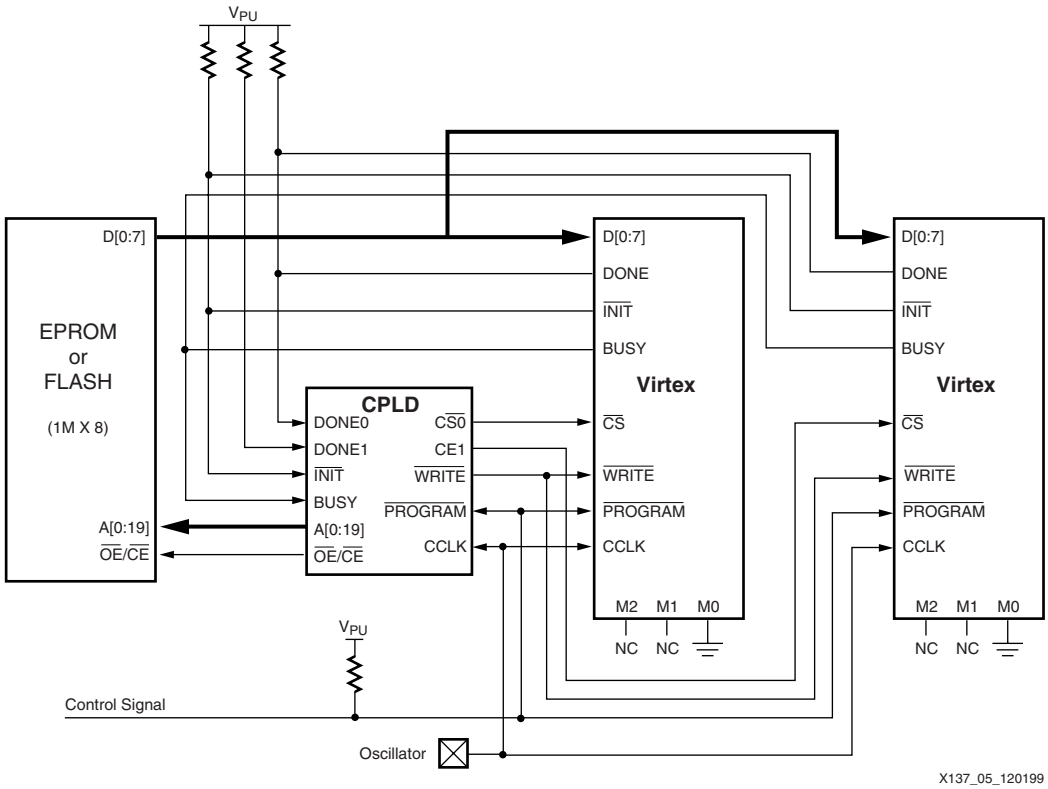

Figure 3-9: **Configuring Virtex-II from Parallel EPROMs**

Embedded Solutions

Using an Embedded Microcontroller

XAPP058: Xilinx *In-System Programming Using an Embedded Microcontroller* (available on www.xilinx.com) describes a compact and robust process that (re)configures Virtex-II devices directly from a microprocessor through the JTAG test port of the Virtex-II device. The process additionally supports (re)configuration of XC18V00 ISP PROMs and CPLDs that reside on the JTAG scan chain. Portable, reference C-code is provided with the application note for rapid implementation.

Using IEEE Standard 1532

Systems that implement an IEEE Standard 1532 player can configure Virtex-II devices. Users need a 1532 BSDL file and a 1532 configuration data file. 1532 BSDL files for Xilinx devices and information on the Xilinx J DRIVE 1532 configuration engine are available at http://www.support.xilinx.com.

PROM Selection Guide

Use Table 3-6 to determine which PROMs go with which Virtex-II FPGAs.

Table 3-6: **Using Virtex-II Devices With PROMs**

Virtex-II Device	Approximate Bitstream Length(3) (Megabits)	PROM Family		PROM Package				
		18Vxx	17Vxx	V08	SO20	PC20	PC44	VQ44
XCV2V40	0.36	18V01	17V01	x(1)	x	x		x(2)
XCV2V80	0.64	18V01	17V01	x(1)	x	x		x(2)
XCV2V250	1.7	18V02	17V02			x(1)	x	x
XCV2V500	2.8	18V04	17V04			x(1)	x	x
XCV2V1000	4.1	18V04	17V04			x(1)	x	x
XCV2V1500	5.7	18V04 18V02	17V08				x	x
XCV2V2000	7.5	2, 18V04	17V08				x	x
XCV2V3000	10.5	3, 18V04	17V16				x	x
XCV2V4000	15.7	4, 18V04	17V16				x	x
XCV2V6000	21.8	5, 18V04 18V02	17V16 17V08				x	x
XCV2V8000	29.0	7, 18V04	2, 17V16				x	x

Notes:

1. 17Vxx only
2. 18Vxx only
3. Different versions of software produce different bit counts and in some cases can reduce the PROM size requirement. To obtain the most accurate information, visit the Xilinx Answers Search Data Base and reference record 12326 at www.support.xilinx.com.

Software Support and Data Files

This section provides information on Xilinx device programming software and configuration-related data files.

iMPACT Software

For programming Virtex-II and other Xilinx devices with a personal computer, Xilinx provides iMPACT software as a part of the ISE software package. A free version of iMPACT software is also available through the WebPACK software suite. More information on WebPACK is available at **http://www.support.xilinx.com**.

Programming Cables

iMPACT software supports several Xilinx programming cables that are compatible with Virtex-II devices, including the Parallel Cable III, Parallel Cable IV, and MultiLINX cables. For more information on these cables, or to order programming cables online, visit **http://www.support.xilinx.com**.

Boundary Scan Interconnect Testing for Virtex-II Devices

Virtex-II devices support the EXTEST, INTEST, and SAMPLE/PRELOAD instructions required for Boundary Scan interconnect tests. Xilinx does not provide direct support for Boundary Scan software, although several third party suppliers offer Boundary Scan test equipment that is compatible with Virtex-II devices. A list of third-parties offering Boundary Scan test equipment is available online at **http://www.support.xilinx.com**.

Customers seeking to perform interconnect tests on a Virtex-II device using third-party boundary scan tester must have a Boundary Scan Description Language (BSDL) file for the Virtex-II device. BSDL files for all Xilinx devices are provided with the ISE software installation, and are also available online.

In-System Programming Data Files

Many third party JTAG configuration solutions and in-system configuration solutions require an SVF (Serial Vector Format) or STAPL (Standard Test and Programming Language) file. The SVF and STAPL file formats are used to convey Boundary Scan instructions in a generic format. Customers requiring an SVF or STAPL file can use iMPACT software to generate these files. For more information on SVF, STAPL, and In-System Programming (ISP), see the following resources online:

- The iMPACT *Software Manual* is included on the software manuals web page.
- In-System Programming details are contained in Xilinx Application Note 058.

Master Serial Programming Mode

In serial configuration mode, the FPGA is configured by loading one bit per CCLK cycle. In Master Serial mode, the FPGA drives the CCLK pin. In Slave Serial mode, the FPGAs CCLK pin is driven by an external source. In both serial configuration modes, the MSB of each data byte is always written to the DIN pin first.

The Master Serial mode is designed so the FPGA can be configured from a Serial PROM, Figure 3-10. The speed of the CCLK is selectable by BitGen options, see Appendix B, "BitGen and PROMGen Switches and Options." Be sure to select a CCLK speed supported by the PROM.

Figure 3-10 shows a Master Serial FPGA configuring from a PROM.

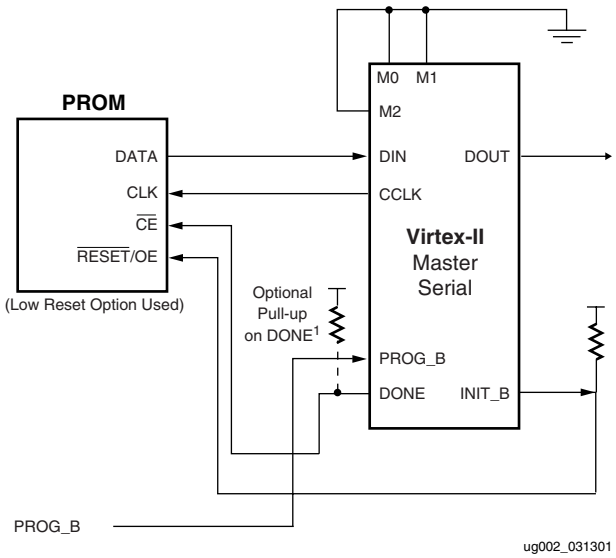

Figure 3-10: **Master Serial Mode Circuit Diagram**

Notes:

1. If the Virtex-II device has not selected the DriveDONE option, then an external pull-up resistor of 330Ω should be added to the DONE pin. This pull-up resistor is not needed if DriveDONE = Yes.

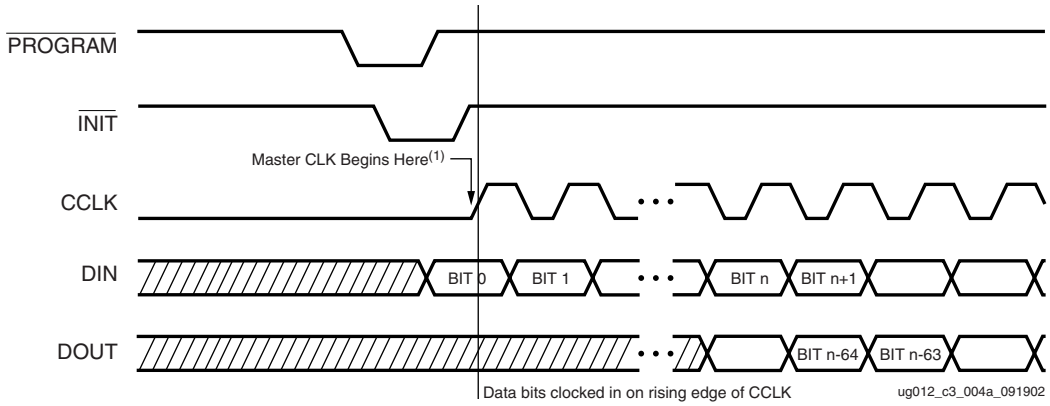

Figure 3-11: **Master Serial Configuration Clocking Sequence**

Notes:

1. For Master configurations, the CCLK does not transition until after initialization, as indicated by the arrow.

Slave Serial Programming Mode

In serial configuration mode, the FPGA is configured by loading one bit per CCLK cycle. In Slave Serial mode, the FPGAs CCLK pin is driven by an external source. In both serial configuration modes, the MSB of each data byte is always written to the DIN pin first.

The Slave Serial configuration mode allows for FPGAs to be configured from other logic devices, such as microprocessors, or in a daisy-chain fashion. Figure 3-12 shows a Master Serial FPGA configuring from a PROM with a Slave Serial FPGA in a daisy-chain with the Master.

Daisy-Chain Configuration

Virtex-II FPGAs can be used in a daisy-chain configuration only with XC4000X, SpartanXL, Spartan-II or other Virtex FPGAs. For serial daisy chains consisting of both 4k and Virtex devices, it is recommended that all Virtex-E, Virtex-II, and Virtex-II Pro devices be grouped at the beginning of the serial daisy chain, with the 4000X and Spartan devices following. For a serial daisy chain consisting only of Spartan-II or Spartan-E and Virtex, Virtex-E, Virtex-II, or Virtex-II Pro devices, there are no restrictions on the order of the devices in the chain. However, there are restrictions on the total number of bits that an FPGA can pass to the downstream FPGAs in the chain. If a Virtex-II FPGA is placed as the Master and a non-Virtex-II FPGA is placed as a slave, select a configuration CCLK speed supported by *all* devices in the chain.

The separate bitstreams for the FPGAs in a daisy-chain must be combined into a single PROM file, by using either iMPACT software or the PROMGen utility (see Appendix B, "BitGen and PROMGen Switches and Options"). Separate .bit files can *not* be simply concatenated together to form a daisy-chain bitstream.

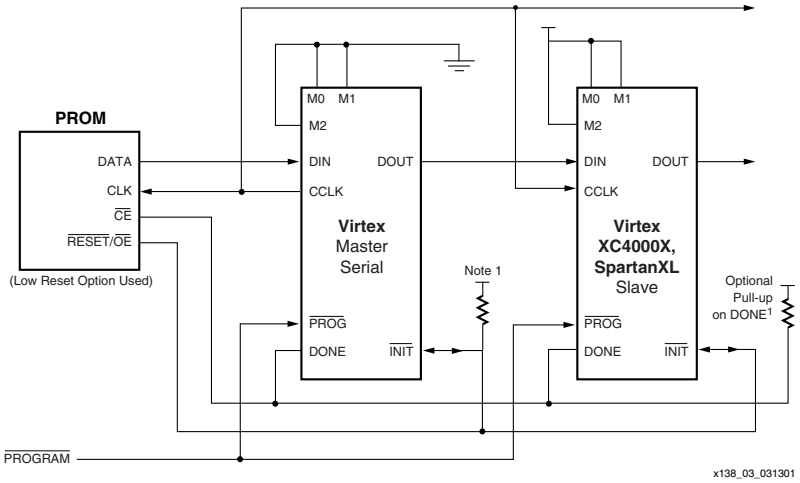

Figure 3-12: **Master/Slave Serial Mode Circuit Diagram**

Notes:

1. If none of the devices have been selected to DriveDONE, then an external pull-up resistor of 330 Ω should be added to the common DONE line. This pull-up resistor is not needed if DriveDONE = Yes. If used, DriveDONE should be selected only for the last device in the configuration chain.

The first device in the chain is the first to be configured. No data is passed onto the DOUT pin until all the data frames, start-up command, and CRC check have been loaded. CRC checks only include the data for the current device, not for any others in the chain. After finishing the first stream, data for the next device is loaded. The data for the downstream device appears on DOUT typically about 80 CCLK cycles after being loaded into DIN. This is due to internal packet processing. Each daisy-chained bitstream carries its own

synchronization word. Nothing of the first bitstream is passed to the next device in the chain other than the daisy-chained configuration data.

The DONE_cycle must be set before GTS, or during the same cycle to guarantee each Virtex-II device to move to the operation state when all the DONE pins have been released. When daisy-chaining multiple devices, either set the last device in the chain to DriveDONE, or add external pull-up resistors to counteract the combined capacitive loading on DONE. If non-Virtex devices are included in the daisy-chain, it is important to set their bitstreams to SyncToDONE with BitGen options. For more information on Virtex BitGen options, see Appendix B, "BitGen and PROMGen Switches and Options.".

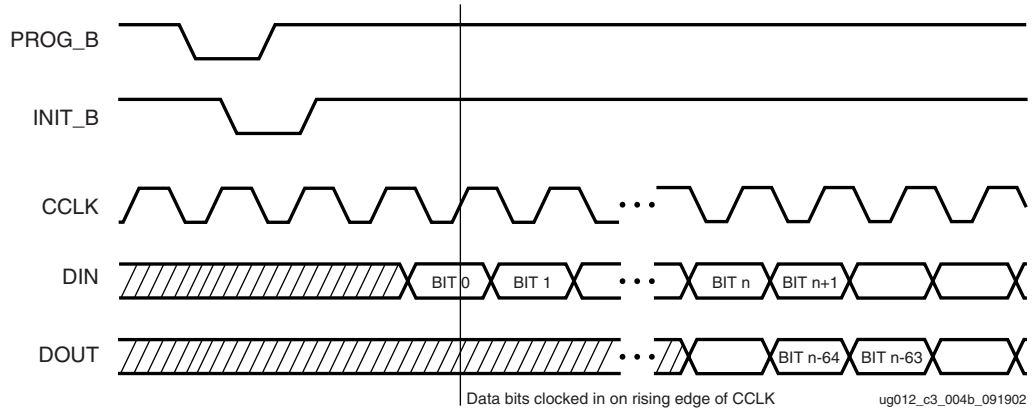

Figure 3-13: **Serial Configuration Clocking Sequence**

Notes:

1. For Slave configurations, a free running CCLK can be used, as shown in Figure 3-13.

Table 3-7: **Master/Slave Serial Mode Programming Switching**

	Description	Symbol	Values	Units
CCLK	DIN setup/hold, slave mode	T_{DCC}/T_{CCD}	5.0/0.0	ns, min
	DIN setup/hold, master mode	T_{DSCK}/T_{SCKD}	5.0/0.0	ns, min
	DOUT	T_{CCO}	12.0	ns, max
	High time	T_{CCH}	5.0	ns, min
	Low time	T_{CCL}	5.0	ns, min
	Maximum Frequency	F_{CC_SERIAL}	66	MHz, max
	Frequency Tolerance, master mode with respect to nominal		+45% -30%	

Master SelectMAP Programming Mode

The SelectMAP mode provides an 8-bit bidirectional data bus interface to the Virtex-II configuration logic that can be used for both configuration and readback. Virtex-II devices can not be serially daisy-chained when the SelectMAP interface is used. However, they can be connected in a parallel-chain as shown in Figure 3-16. The DATA pins (D0:D7), CCLK, RDWR_B, BUSY, PROG_B, DONE, and INIT_B can be connected in common between all of the devices. CS_B inputs should be kept separate so each device can be accessed individually. If all devices are to be configured with the same bitstream, readback is not being used, and CCLK is less than F_{CC}_SelectMAP, the CS_B pins can be connected to a common line so the devices are configured simultaneously.

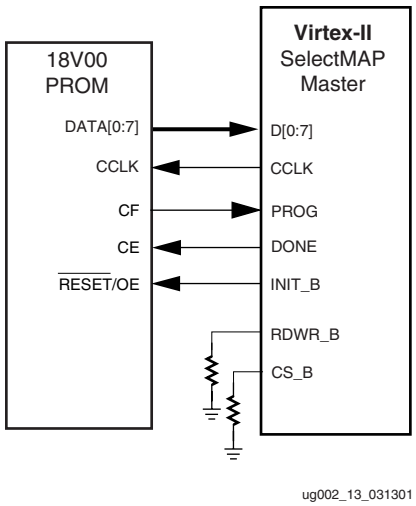

Figure 3-14: **Virtex-II Interfaced With an 18V00 PROM**

Notes:

1. If none of the Virtex-II devices have been selected to DriveDONE, add an external 330 Ω pull-up resistor to the common DONE line. This pull-up resistor is not needed if DriveDONE is selected. If used, DriveDONE should be selected only for the last device in the configuration chain.

The following pins are involved in Master SelectMAP configuration mode:

DATA Pins (D[0:7])

The D0 through D7 pins function as a bidirectional data bus in the SelectMAP mode. Configuration data is written to the bus, and readback data is read from the bus. The bus direction is controlled by the RDWR_B signal. see “Configuration Details” on page 288. The D0 pin is considered the MSB of each byte.

RDWR_B

When asserted Low, the RDWR_B signal indicates that data is being written to the data bus. When High, the RDWR_B signal indicates that data is being read from the data bus.

CS_B

The Chip Select input (CS_B) enables the SelectMAP data bus. To write or read data onto or from the bus, the CS_B signal must be asserted Low. When CS_B is High, Virtex-II devices do not drive onto or read from the bus.

CCLK

The CCLK pin is a clock output in the Master SelectMAP interface. It synchronizes all loading and reading of the data bus for configuration and readback. The CCLK pin is driven by the FPGA.

Data Loading

To load data in the Master SelectMAP mode, a data byte is loaded on every rising CCLK edge as shown in Figure 3-15. If the CCLK frequency is less than F_{CC}_SelectMAP, this can be done without handshaking. For frequencies above F_{CC}_SelectMAP, the BUSY signal must be monitored. If BUSY is High, the current byte must be reloaded when BUSY is Low.

The first byte can be loaded on the first rising CCLK edge that INIT_B is High, and when both CS_B and RDWR_B are asserted Low. CS_B and RDWR_B can be asserted anytime before or after INIT_B has gone High. However, the SelectMAP interface is not active until after INIT_B has gone High. The order of CS_B and RDWR_B does not matter, but RDWR_B must be asserted throughout configuration. If RDWR_B is de-asserted before all data has been loaded, the FPGA aborts the operation. To complete configuration, the FPGA must be reset by PROG_B and reconfigured with the entire stream. For applications that need to de-assert RDWR_B between bytes, see "Controlled CCLK" on page 269.

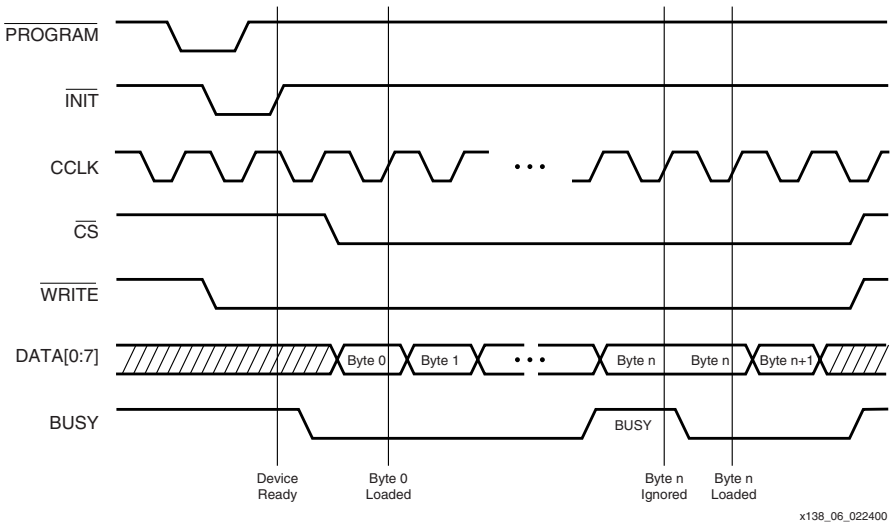

Figure 3-15: **Data Loading in SelectMAP**

Slave SelectMAP Programming Mode

The SelectMAP mode provides an 8-bit bidirectional data bus interface to the Virtex-II configuration logic that can be used for both configuration and readback. Virtex-II devices can not be serially daisy-chained when the SelectMAP interface is used. However, they can be connected in a parallel-chain as shown in Figure 3-16. The DATA pins (D0:D7), CCLK, RDWR_B, BUSY, PROG_B, DONE, and INIT_B can be connected in common between all of the devices. CS_B inputs should be kept separate so each device can be accessed individually. If all devices are to be configured with the same bitstream, readback is not being used, and CCLK is less than F_{CC}_SelectMAP, the CS_B pins can be connected to a common line so the devices are configured simultaneously.

Although Figure 3-16 does not show a control module for the SelectMAP interface, the SelectMAP interface is typically driven by a processor, micro controller, or some other logic device such as an FPGA or a CPLD.

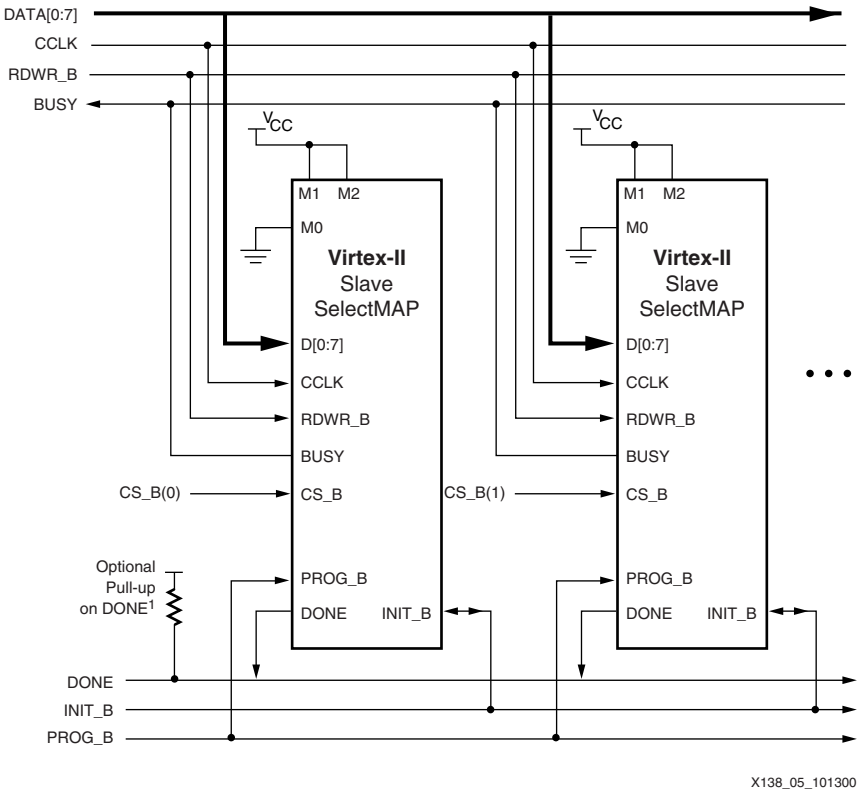

Figure 3-16: **Slave SelectMAP Mode Circuit Diagram**

Notes:

1. If none of the Virtex-II devices have been selected to DriveDONE, add an external 330 Ω pull-up resistor to the common DONE line. This pull-up resistor is not needed if DriveDONE = Yes. If used, DriveDONE should be selected only for the last device in the configuration chain.

The following pins are involved in Slave SelectMAP configuration mode:

DATA Pins (D[0:7])

The D0 through D7 pins function as a bidirectional data bus in the SelectMAP mode. Configuration data is written to the bus, and readback data is read from the bus. The bus direction is controlled by the RDWR_B signal. see “Configuration Details” on page 288.. The D0 pin is considered the MSB of each byte.

RDWR_B

When asserted Low, the RDWR_B signal indicates that data is being written to the data bus. When asserted High, the RDWR_B signal indicates that data is being read from the data bus.

CS_B

The Chip Select input (CS_B) enables the SelectMAP data bus. To write or read data onto or from the bus, the CS_B signal must be asserted Low. When CS_B is High, Virtex-II devices do not drive onto or read from the bus.

BUSY

When CS_B is asserted, the BUSY output indicates when the FPGA can accept another byte. If BUSY is Low, the FPGA reads the data bus on the next rising CCLK edge where both CS_B and RDWR_B are asserted Low. If BUSY is High, the current byte is ignored and must be reloaded on the next rising CCLK edge when BUSY is Low. When CS_B is *not* asserted, BUSY is 3-stated.

BUSY is only necessary for CCLK frequencies above F_{CC}_SelectMAP. For frequencies at or below F_{CC}_SelectMAP, BUSY is ignored, see “Data Loading” on page 265. For parallel chains, as shown in Figure 3-16, where the same bitstream is to be loaded into multiple devices simultaneously, BUSY should not be used. Thus, the maximum CCLK frequency for such an application must be less than F_{CC}_SelectMAP.

CCLK

Unlike the Master SelectMAP mode of configuration, the CCLK pin is an input in the Slave SelectMAP mode interface. The CCLK signal synchronizes all loading and reading of the data bus for configuration and readback. Additionally, the CCLK drives internal configuration circuitry. The CCLK can be driven either by a free running oscillator or an externally-generated signal.

Several scenarios exist when configuring the FPGA in SelectMAP mode, depending on the source of CCLK.

Free-Running CCLK

A free-running oscillator can be used to drive Virtex-II CCLK pins. For applications that can provide a continuous stream of configuration data, refer to the timing diagram discussed in "Data Loading" on page 265. For applications that cannot provide a continuous data stream, missing the clock edges, refer to the timing diagram discussed in "Non-Contiguous Data Strobe" on page 268. An alternative to a free-running CCLK is discussed in "Controlled CCLK" on page 269.

Express-Style Loading

In express-style loading, a data byte is loaded on every rising CCLK edge as shown in Figure 3-17. If the CCLK frequency is less than F_{CC}_SelectMAP, this can be done without handshaking. For frequencies above F_{CC}_SelectMAP, the BUSY signal must be monitored. If BUSY is High, the current byte must be reloaded when BUSY is Low.

The first byte can be loaded on the first rising CCLK edge that INIT_B is High, and when both CS_B and RDWR_B are asserted Low. CS_B and RDWR_B can be asserted anytime before or after INIT_B has gone High. However, the SelectMAP interface is not active until after INIT_B has gone High. The order of CS_B and RDWR_B does not matter, but RDWR_B must be asserted throughout configuration. If RDWR_B is de-asserted before all data has been loaded, the FPGA aborts the operation. To complete configuration, the FPGA must be reset by PROG_B and reconfigured with the entire stream.

For applications that need to de-assert RDWR_B between bytes, see "Controlled CCLK" on page 269.

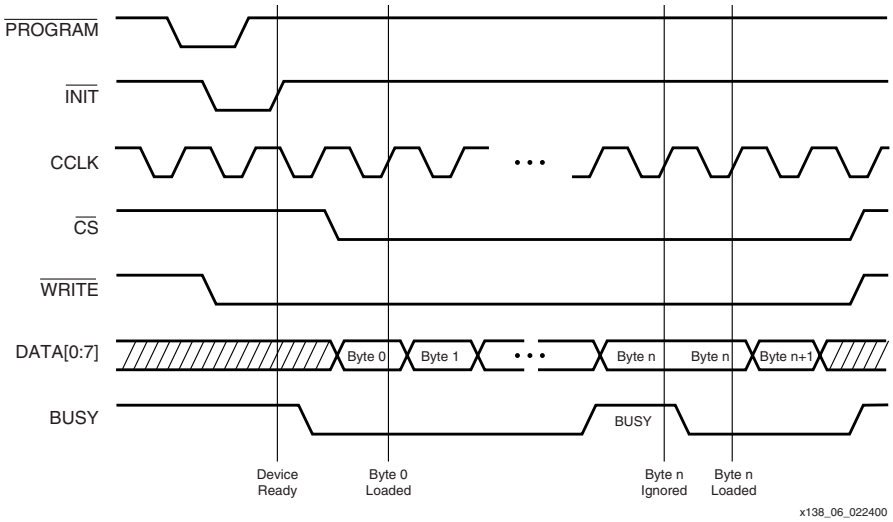

Figure 3-17: **"Express Style" Continuous Data Loading in SelectMAP**

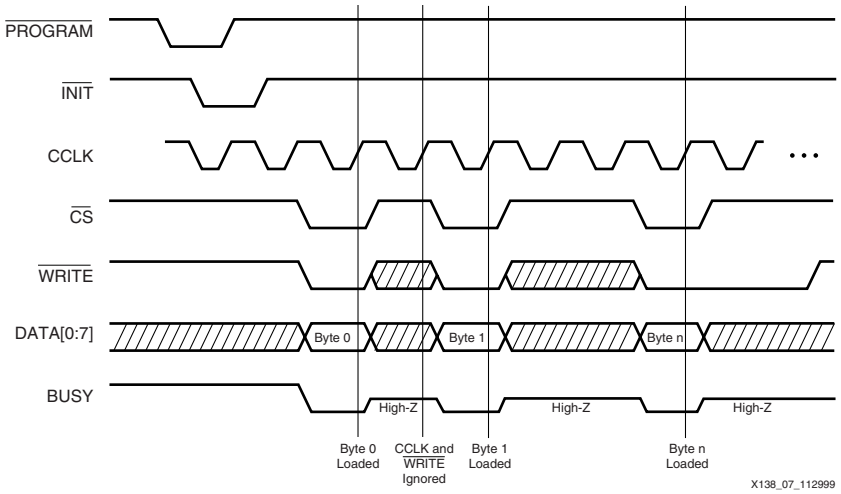

Figure 3-18: **Separating Data Loads by Multiple CCLK Cycles Using CS_B**

Non-Contiguous Data Strobe

In applications where multiple clock cycles might be required to access the configuration data before each byte can be loaded into the SelectMAP interface, data might not be ready for each consecutive CCLK edge. In such a case, the CS_B signal can be de-asserted until the next data byte is valid on the DATA[0:7] pins. This is demonstrated in Figure 3-18. While CS_B is High, the SelectMAP interface does not expect any data and ignores all CCLK transitions. However, RDWR_B must continue to be asserted while CS_B is asserted. If RDWR_B is High during a positive CCLK transition while CS_B is asserted, the FPGA aborts the operation. For applications that need to de-assert the RDWR_B signal without de-asserting CS_B, see "Controlled CCLK".

Controlled CCLK

Some applications require that RDWR_B be de-asserted between the loading of configuration data bytes asynchronously from the CS_B. Typically, this would be due to the RDWR_B signal being a common connection to other devices on the board, such as memory storage elements. In such a case, driving CCLK as a controlled signal instead of a free-running oscillator makes this type of operation possible. In Figure 3-19, the CCLK, $\overline{CS\_B}$, and RDWR_B are asserted Low while a data byte becomes active. Once the CCLK has gone High, the data is loaded. RDWR_B can be de-asserted and re-asserted as many times as necessary, just as long as it is Low before the next rising CCLK edge.

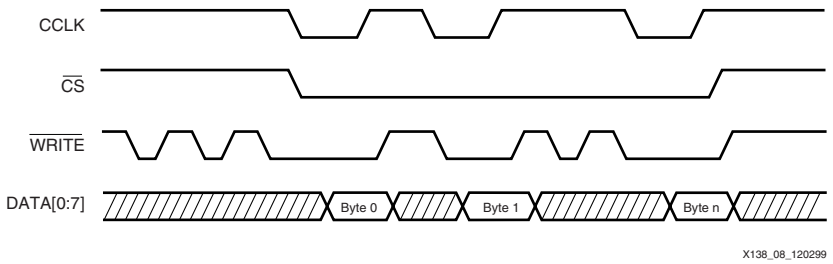

Figure 3-19: **Controlling CCLK for RDWR_B De-Assertion**

Table 3-8: **SelectMAP Write Timing Characteristics**

	Description	Symbol	Value	Units
CCLK	D_{0-7} Setup/Hold	T_{SMDCC}/T_{SMCCD}	5.0/0.0	ns, min
	CS_B Setup/Hold	T_{SMCSCC}/T_{SMCCCS}	7.0/0.0	ns, min
	RDWR_B Setup/Hold	T_{SMCCW}/T_{SMWCC}	7.0/0.0	ns, min
	BUSY Propagation Delay	T_{SMCKBY}	12.0	ns, max
	Maximum Frequency	$F_{CC_}$SelectMAP	66	MHz, max
	Maximum Frequency with no handshake	F_{CCNH}	66	MHz, max

JTAG/ Boundary Scan Programming Mode

Introduction

Virtex-II devices support the new IEEE 1532 standard for In-System Configuration (ISC), based on the IEEE 1149.1 standard. The IEEE 1149.1 Test Access Port and Boundary-Scan Architecture is commonly referred to as JTAG. 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 assure the integrity of individual components and the interconnections between them at the board level. 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-II Devices Using IEEE Standard 1149.1

The Virtex-II 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 Test Access Port (TAP), the TAP controller, the instruction register, the instruction decoder, the boundary-scan register, and the bypass register. The Virtex-II 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-II devices.

Test Access Port

The Virtex-II TAP contains four mandatory dedicated pins as specified by the protocol (Table 3-9).

Table 3-9: **Virtex-II TAP Controller Pins**

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

There are three input pins and one output pin to control the 1149.1 boundary-scan TAP controller. There are optional control pins, such as $\overline{\text{TRST}}$ (Test Reset) and enable pins, which might be found on devices from other manufacturers. It is important to be aware of these optional signals when interfacing Xilinx devices with parts from different vendors, because they might need to be driven.

The TAP controller is a 16-state state machine shown in Figure 3-20. The four mandatory TAP pins are outlined below.

- TMS - This pin determines the sequence of states through the TAP controller 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-II 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 only active during the shifting of instructions or data through the device. This pin is 3-stated at all other times.

Notes:

As specified by the IEEE Standard, the TMS and TDI pins all have internal pull-up resistors. These internal pull-up resistors of 50-150 kΩ are active, regardless of the mode selected.

For JTAG configuration mode, JTAG inputs are independent of V_{CCO} and work between 2.5V and 3.3V TTL levels (VIL max = .8V, VIH min = 2.0V). The JTAG input pins are 3.3V tolerant. The JTAG output (TDO) is an open-drain output and must be pulled up to the appropriate voltage level (typically 3.3V) through an external resistor. The value of the external pullup resistor depends on the capacitive loading on the TDO pin and the operating frequency, but it should not be less than 200 ohms. The optimal TDO pullup value can be determined through IBIS simulation.

TAP Controller

Figure 3-20 diagrams a 16-state finite state machine. The four TAP pins control how data is scanned into the various registers. The state of the TMS pin at the rising edge of 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.

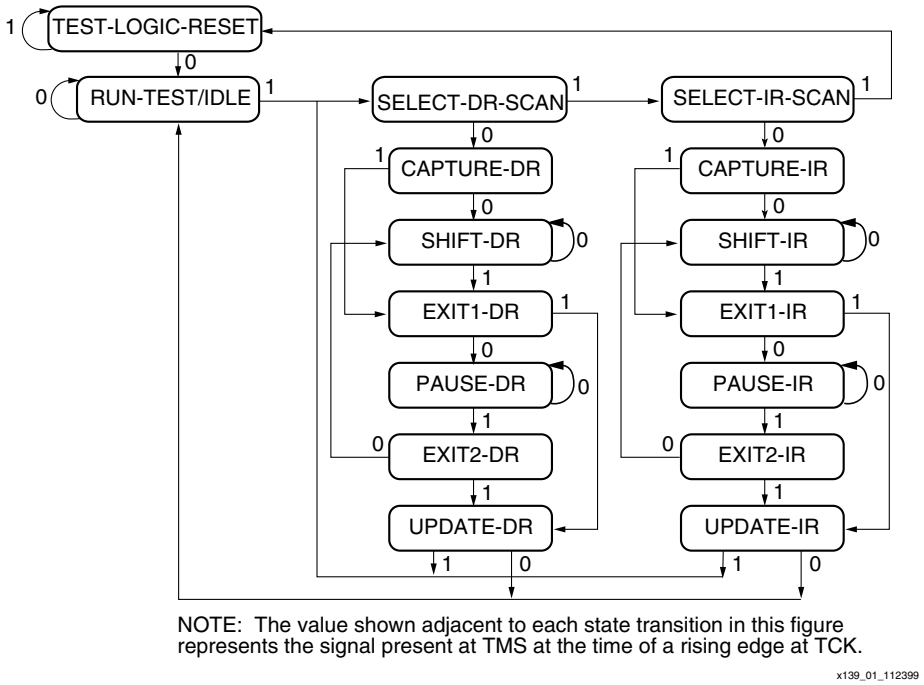

Figure 3-20: **State Diagram for the TAP Controller**

Boundary-Scan Instruction Set

To determine the operation to be invoked, an instruction is loaded into the Instruction Register (IR). The Instruction Register is 6 bits long in Virtex-II devices to support the new IEEE Standard 1532 for In-System Configurable (ISC) devices. Table 3-10 lists the available instructions for Virtex-II devices.

Table 3-10: **Virtex-II Boundary Scan Instructions**

Boundary Scan Command	Binary Code (5:0)	Description
EXTEST	000000	Enables boundary-scan EXTEST operation
SAMPLE	000001	Enables boundary-scan SAMPLE operation
USER1	000010	Access user-defined register 1
USER2	000011	Access user-defined register 2
CFG_OUT	000100	Access the configuration bus for readback
CFG_IN	000101	Access the configuration bus for configuration
INTEST	000111	Enables boundary-scan INTEST operation
USERCODE	001000	Enables shifting out user code
IDCODE	001001	Enables shifting out of ID code
HIGHZ	001010	3-states output pins while enabling the bypass register
JSTART	001100	Clocks the start-up sequence when StartClk is TCK
JSHUTDOWN	001101	Clocks the shutdown sequence
BYPASS	111111	Enables BYPASS
JPROG_B	001011	Equivalent to and has the same affect as PROG_B
RESERVED	All other codes	Xilinx reserved instructions

The mandatory IEEE 1149.1 commands are supported in Virtex-II devices, as well as several Xilinx vendor-specific commands. Virtex-II 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-II boundary-scan operations are independent of mode selection. The boundary-scan mode in Virtex-II devices overrides 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 a Virtex-II device is configured. After configuration, all instructions are available.

JSTART and JSHUTDOWN are instructions specific to the Virtex-II architecture and configuration flow. As described in Table 3-10, the JSTART and JSHUTDOWN instructions clock 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 "USER1, USER2 Registers" on page 276.

Boundary-Scan Architecture

Virtex-II device registers include all registers required by the IEEE 1149.1 Standard. In addition to the standard registers, the family contains optional registers for simplified testing and verification (Table 3-11).

Table 3-11: **Virtex-II JTAG Registers**

Register Name	Register Length	Description
Instruction register	6 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	64 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. Boundary-scan operation is independent of individual IOB configurations. Each IOB, bonded or un-bonded, starts 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 3-21).

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 the UPDATE-DR state when 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 that the proper data has been latched before exercising the command. This is typically accomplished by using the SAMPLE/PRELOAD instruction.

Consider internal pull-up and pull-down resistors when developing test vectors for testing opens and shorts. The boundary-scan mode determines if the IOB has a pull-up resistor. Figure 3-21 is a representation of Virtex-II Boundary-Scan Architecture.

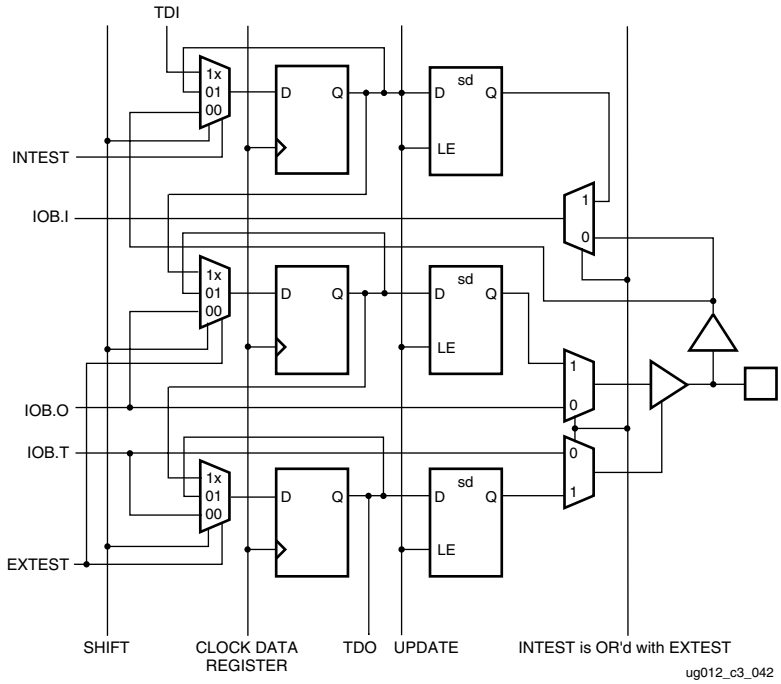

Figure 3-21: **Virtex Series Boundary Scan Logic**

Bit Sequence

The order in each non-TAP IOB is described in this section. The input is first, then 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 independent 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 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 CAPTURE-DR state.

Instruction Register

The instruction register is a 6-bit register that loads the OPCODE necessary for the Virtex-II boundary-scan instruction set. This register loads the current OPCODE and captures internal device status.

Configuration Register (Boundary-Scan)

The configuration register is a 64-bit register. This register allows access to the configuration bus and readback operations.

Identification Register

Virtex devices have a 32-bit identification register, commonly referred to as the IDCODE register. This register is based upon IEEE Standard 1149.1 and allows easy identification of the part being tested or programmed via boundary scan.

Virtex-II Identification Register

The Virtex-II JTAG ID Code register has the following format.

```
3322 2222222 211111111 110000000000
1098 7654321 098765432 109876543210   <-- bit positions(00 to 31)
vvvv:ffffffff:aaaaaaaaa:ccccccccccc1
```

where

v is the revision code and

f is the 7-bit family code = 0001000 0x08

a is the number of array rows in the part expressed in 9 bits.

XC2V40 =	8 =	0x08
XC2V80 =	16 =	0x010
XC2V250 =	24 =	0x018
XC2V500 =	32 =	0x020
XC2V1000 =	40 =	0x028
XC2V1500 =	48 =	0x030
XC2V2000 =	56 =	0x038
XC2V3000 =	64 =	0x040
XC2V4000 =	80 =	0x050
XC2V6000 =	96 =	0x060
XC2V8000 =	112 =	0x070

c is the company code = 00001001001 = 0x049*

*Since the last bit of the JTAG IDCODE is always one, the last three hex digits appear as 0x093.

	vvvv	ffff	fff	a	aaaa	aaaa	cccc	cccc	cccc
		0001	000	0	0001	1000	0000	1001	0011
XC2V250	v	1	0		1	8	0	9	3
XC2V500	v	1	0		2	0	0	9	3

ID Codes assigned to Virtex-II FPGAs are shown in Table 3-12.

Table 3-12: **Virtex-II Device ID Codes**

FPGA	IDCODE
XC2V40	v01008093
XC2V80	v01010093
XC2V250	v01018093
XC2V500	v01020093
XC2V000	v01028093
XC2V1500	v01030093
XC2V2000	v01038093
XC2V3000	v01040093
XC2V4000	v01050093
XC2V6000	v01060093
XC2V8000	v01070093

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 registers are only valid after configuration. These two registers must be defined by the user within the design. These registers can be accessed after they are defined by the TAP pins.

The BSCAN_VIRTEX2 library macro is required when creating these registers. This symbol is only required for driving internal scan chains (USER1 and USER2). The BSCAN_VIRTEX2 macro provides two user pins (SEL1 and SEL2) for determining 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 earlier FPGA families that required the BSCAN macro to dedicate TAP pins for boundary scan, Virtex-II TAP pins are dedicated and do not require the BSCAN_VIRTEX2 macro for normal boundary-scan instructions or operations.

Note that these are user-defined registers. The example (Figure 3-22) is one of many implementations. For HDL, the BSCAN_VIRTEX2 macro needs to be instantiated in the design.

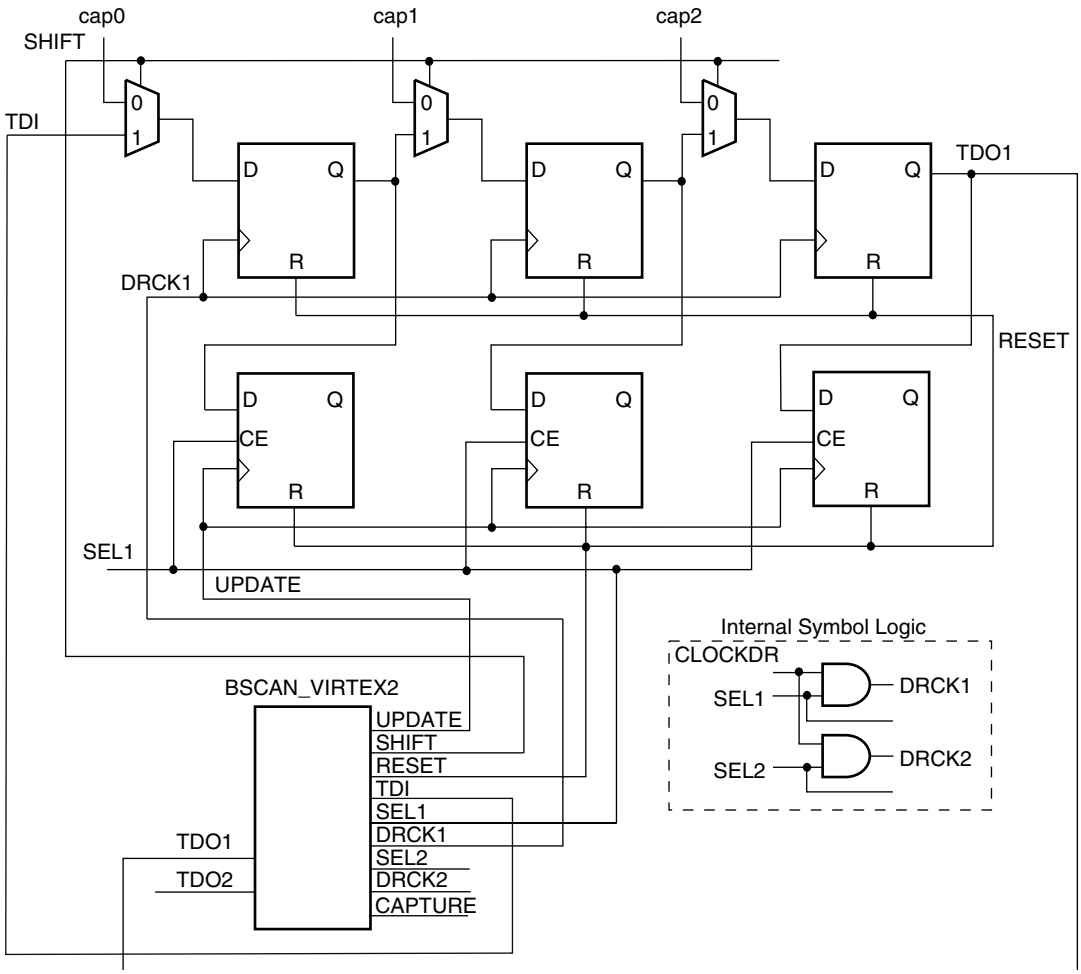

Figure 3-22: **BSCAN_VIRTEX2 (Example Usage)**

Characterization data for some of the most commonly requested timing parameters shown in Figure 3-23 is listed in Table 3-13.

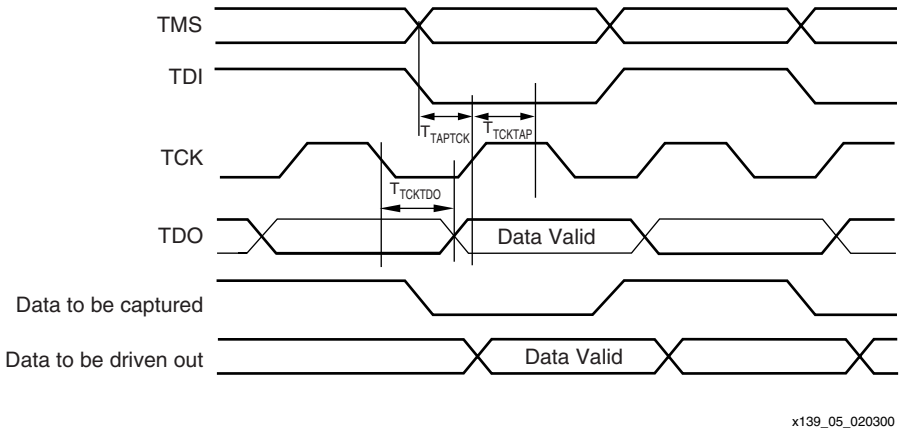

Figure 3-23: **Virtex-II Boundary Scan Port Timing Waveforms**

Table 3-13: **Boundary-Scan Port Timing Specifications**

Symbol	Parameter	Value	Units
T_{TAPTCK}	TMS and TDI setup time before TCK	4.0	ns, min
T_{TCKTAP}	TMS and TDI hold times after TCK	2.0	ns, min
T_{TCKTDO}	TCK falling edge to TDO output valid	11.0	ns, min
F_{TCK}	Maximum TCK clock frequency	33.0	MHz, max

For further information on the Startup sequence, bitstream, and internal configuration registers referenced here, refer to "Readback" on page 297.

Configuring Through Boundary-Scan

One of the most common boundary-scan vendor-specific instructions is the configure instruction. An individual Virtex-II device is configured via JTAG on power-up using TAP. If the Virtex-II device is configured on power-up, it is advisable to tie the mode pins to the boundary-scan configuration mode settings; 101 (M2 = 1, M1 = 0, M0 = 1).

Configuration flow for Virtex-II device configuration with JTAG is shown in Figure 3-24. The sections that follow describe how the Virtex-II device can be configured as a single device via boundary-scan or as part of a multiple-device scan chain.

A configured device can be reconfigured by toggling the TAP and entering a CFG_IN instruction after pulsing the PROG_B pin or issuing the shut-down sequence. (Refer to "Power Up" on page 249). For additional details on power-up or the start-up sequence in Virtex-II devices, see "Device Startup" on page 251.

Users seeking to implement a Virtex-II JTAG configuration algorithm are advised to use the SVF-based flow provided in Xilinx Application Note 058 (available on www.xilinx.com).

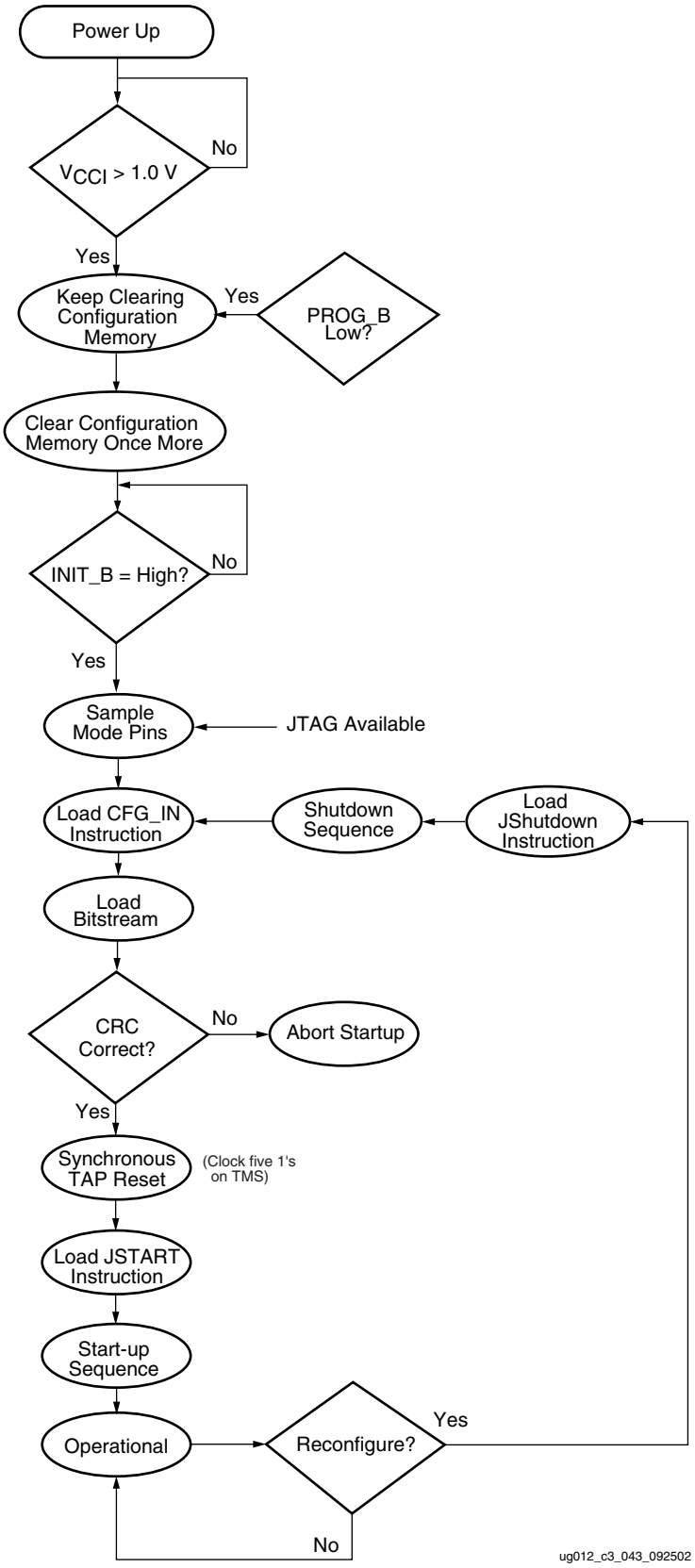

Figure 3-24: **Device Configuration Flow Diagram**

Single Device Configuration

Configure a Virtex-II part as a single device via boundary-scan operations as follows. Ensure that the bitstream is generated with the JTAG clock option.

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

Also, when using iMPACT software, verify that the most current version is being used.

Table 3-14 describes the TAP controller commands required to configure a Virtex-II device. Refer to Figure 3-20 for TAP controller states. These TAP controller commands are issued automatically if configuring the part with the iMPACT software.

Table 3-14: **Single Device Configuration Sequence**

TAP Controller Step Description		Set & Hold		# of Clocks
		TDI	TMS	TCK
1.	On power-up, place a logic “one” 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, LSB first:	111_000101	0	9
6.	Load the MSB of CFG_IN instruction when exiting SHIFT-IR, as defined in the IEEE standard.	1	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-II bitstream. Bit_n (MSB) is the first bit in the bitstream(1).	$bit_1 \ldots bit_n$	0	(bits in bitstream)-1
10.	Shift in the last bit of the bitstream. Bit_0 (LSB) shifts on the transition to EXIT1-DR.	bit_0	1	1
11.	Enter UPDATE-DR state.	X	1	1
12.	Reset TAP by clocking five 1’s on TMS	X	1	5
13.	Enter the SELECT-IR state.	X	1	2
14.	Move to the SHIFT-IR state.	X	0	2
15.	Start loading the JSTART instruction. The JSTART instruction initializes the startup sequence.	01100	0	5
16.	Load the last bit of the JSTART instruction.	0	1	1
17.	Move to the UPDATE-IR state.	X	1	1
18.	Move to RTI and clock the STARTUP sequence by applying a minimum of 12 clock cycles to the TCK.	X	0	Š12
19.	Move to the TLR state. The device is now functional.	X	1	3

Notes:

1. In the Configuration Register, data is shifted in from the right (TDI) to the left (TDO).

Multiple Device Configuration

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

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

Refer to the State Diagram in Figure 3-20 for the following TAP controller steps.

1. On power-up, place a logic "one" 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). Go through RTI (RUN-TEST/IDLE).
3. Load in the configuration bitstream per step 7 through step 11 in Table 3-14.
4. Repeat step 2 and step 3 for each device.
5. Reset all TAPs by clocking five 1's on TMS.
6. Load the JSTART command into all devices.
7. Go to RTI and clock TCK 12 times.
   All devices are active at this point.

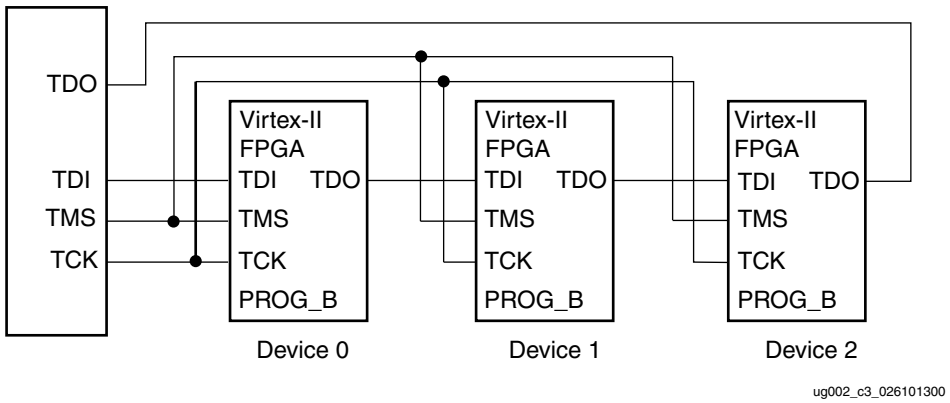

Figure 3-25: **Boundary Scan Chain of Devices**

Notes:

1. PROG_B pin should be deasserted during JTAG operation.

Reconfiguring Through Boundary Scan

The ability of Virtex-II devices to perform partial reconfiguration is the reason that the configuration memory is not cleared when reconfiguring the device. When reconfiguring a chain of devices, refer to step 3 in Table 3-14. There are two methods to reconfigure Virtex-II devices without possible internal contention. The first method is to pulse the PROG_B pin which resets the internal configuration memory. The alternate method is to perform a shutdown sequence, placing the device in a safe state. The following shutdown sequence includes using internal registers. (For details on internal registers, refer to "Readback" on page 297.)

1. Load the CFG_IN instruction.
2. In SHIFT-DR state, load the synchronization word followed by the Reset CRC Register (RCRC) command.

```
1111 1111 1111 1111 1111 1111 1111 1111-> Dummy word
1010 1010 1001 1001 0101 0101 0110 0110-> Synchronization word
0011 0000 0000 0000 1000 0000 0000 0001-> Header: Write to CMD register
0000 0000 0000 0000 0000 0000 0000 0111-> RCRC command
0000 0000 0000 0000 0000 0000 0000 0000-> flush pipe
0000 0000 0000 0000 0000 0000 0000 0000-> flush pipe
```

3. Load JSHUTDOWN.
4. Go to RTI and clock TCK at least 12 times to clock the shutdown sequence.
5. Proceed to SHIFT-IR state and load the CFG_IN instruction again.
6. Go to SHIFT-DR state and load the configuration bits. Make sure the configuration bits contain AGHIGH command, which asserts the global signal GHIGH_B. 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
```

7. When all configuration bits have been loaded, reset the TAP by clocking five 1's on TMS.
8. Go to SHIFT-IR state and load the JSTART instruction.
9. Go to RTI and clock TCK at least 12 times to clock the startup sequence.
10. Go to TLR state to complete the reconfiguration process.

Debugging Configuration

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

When using TAP controller pins, TDO is driven only in the SHIFT-DR and SHIFT-IR state. If the output of the TDO can be changed via an external pull-up resistor, 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. The INIT_B pin functions similar to a normal INIT_B but does not indicate a configuration error in boundary-scan configuration.

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

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

   1111 1111 1111 1111 1111 1111 1111 1111 -> Dummy word

   1010 1010 1001 1001 0101 0101 0110 0110 -> Synchronization word

   0010 1000 0000 0000 1110 0000 0000 0010 -> Read STATus Register [1])

   0000 0000 0000 0000 0000 0000 0000 0000 -> flush pipe

   0000 0000 0000 0000 0000 0000 0000 0000 -> flush pipe

   0000 0000 0000 0000 0000 0000 0000 0000 -> flush pipe

   Notes:

   1. Since the JTAG readback shift register is 64-bit long, two 32-bit words are needed to fill the shift register.

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

   0000 0000 0000 0000 0001 1MMM 1110 11101[1,2])

   Notes:

   1. MMM is the mode pins value.
   2. Assuming that the device is in normal operation mode.

   Since the read status activity causes the crc_error status to be asserted, it is important to clear the crc_error status to ensure normal device operation. This can be done by writing the precalculated CRC value to the CRC register or writing an RCRC command.

6. Go to SHIFT-IR state and load the CFG_IN instruction again.
7. Move to SHIFT-DR state and shift in the following bit pattern:

   0011 0000 0000 0000 1000 0000 0000 0001 -> Header: Write to CMD register

   0000 0000 0000 0000 0000 0000 0000 0111 -> RCRC command

   0000 0000 0000 0000 0000 0000 0000 0000 -> flush pipe

   0000 0000 0000 0000 0000 0000 0000 0000 -> flush pipe

8. Put the TAP in TLR state when finished.

The device status register also gives the status of the DONE and INIT_B signals. For information on the status register, refer to Figure 3-30.

ISC Modal States

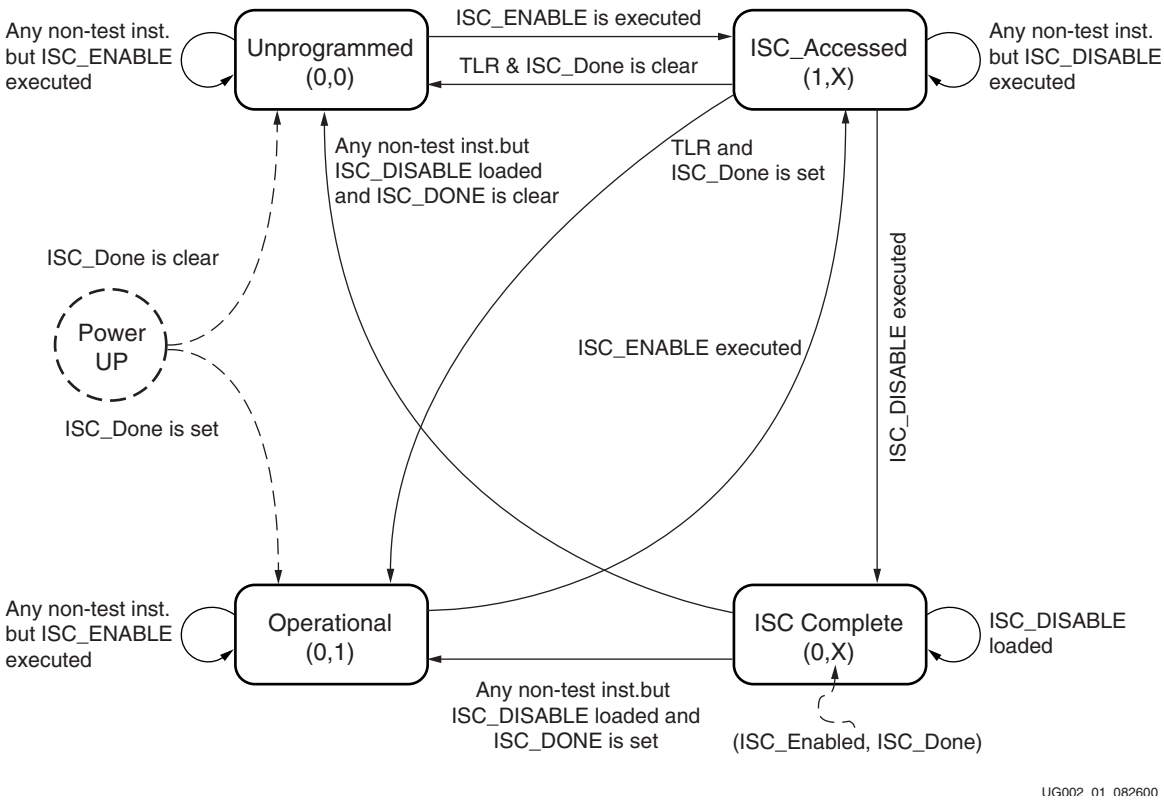

Figure 3-26: **ISC Modal States**

Once the device is powered up, it goes to an Unprogrammed state. The I/Os are all either 3-stated or pulled up. When ISC_ENABLE is successfully executed, the ISC_Enabled signal is asserted, and the device moves to ISC_Accessed state. When the device moves to ISC_Accessed state from Operational state, the shutdown sequence is executed. The I/Os are all either 3-stated or pulled up.

The StartUp sequence is executed when in the ISC_Accessed state. At the end of the StartUp Sequence, ISC_Enabled is cleared and the device moves to ISC_Complete. The minimum clock cycle requirement is the number of clock cycles required to complete the StartUp sequence. At the completion of the minimum required clock cycles, ISC_Enabled is deasserted.

Whether the StartUp sequence is successful or not is determined by CRC or configuration error status from the configuration processor. If the startup is completed, ISC_Done is asserted; otherwise, ISC_Done stays Low. The I/Os are either 3-stated or pulled up.

When ISC_Done is set in ISC_Complete state, the device moves to the Operational state. Otherwise, if ISC_Done is clear, the device moves to an Unprogrammed state. However, if the TAP controller goes to TLR state while the device is in ISC_Accessed state and if ISC_Done is set, then the device moves to the Operational state. However, the I/O is not active yet because the Startup sequence has not been performed. The Startup sequence has to be performed in the Operational state to bring the I/O active.

Clocking Startup and Shutdown Sequence (JTAG Version)

There are three clock sources for Startup and Shutdown sequence, CCLK, UserCLK, and JTAGCLK. Clock selection is set by BitGen. The Startup sequence is executed in ISC_Accessed state. When it is clocked by JTAGCLK, the Startup sequence receives the JTAGCLK in TAP Run/Test Idle state while ISC_DISABLE is the current JTAG instruction. The number of clock cycles in Run/Test Idle state for successful completion of ISC_DISABLE is determined by the number of clock cycles needed to complete the Startup sequence.

When UserCLK or CCLK is used to clock the Startup sequence, the user should know how many JTAGCLK cycles should be spent in Run/Test Idle to successfully complete the Startup sequence.

The Shutdown sequence is executed when the device transitions from an Operational to ISC_Accessed state. Shutdown is done while executing the ISC_ENABLE instruction. When the Shutdown sequence is clocked using JTAGCLK, the clock is supplied in the Run/Test Idle state of the ISC_ENABLE instruction. The number of clock cycles in Run/Test Idle is determined by the number of clock cycles needed to complete the Shutdown sequence.

When the Shutdown sequence is clocked by CCLK or UserCLK, the user is responsible for knowing how many JTAGCLK cycles in Run/Test Idle are needed to complete the Shutdown sequence.

Notes:

1. It has been decided that when configuring the device through JTAG, the startup and shutdown clock should come from TCK, regardless of the selection in BitGen.
2. In IEEE 1532 configuration mode, Startup and Shutdown clock source is always TCK.

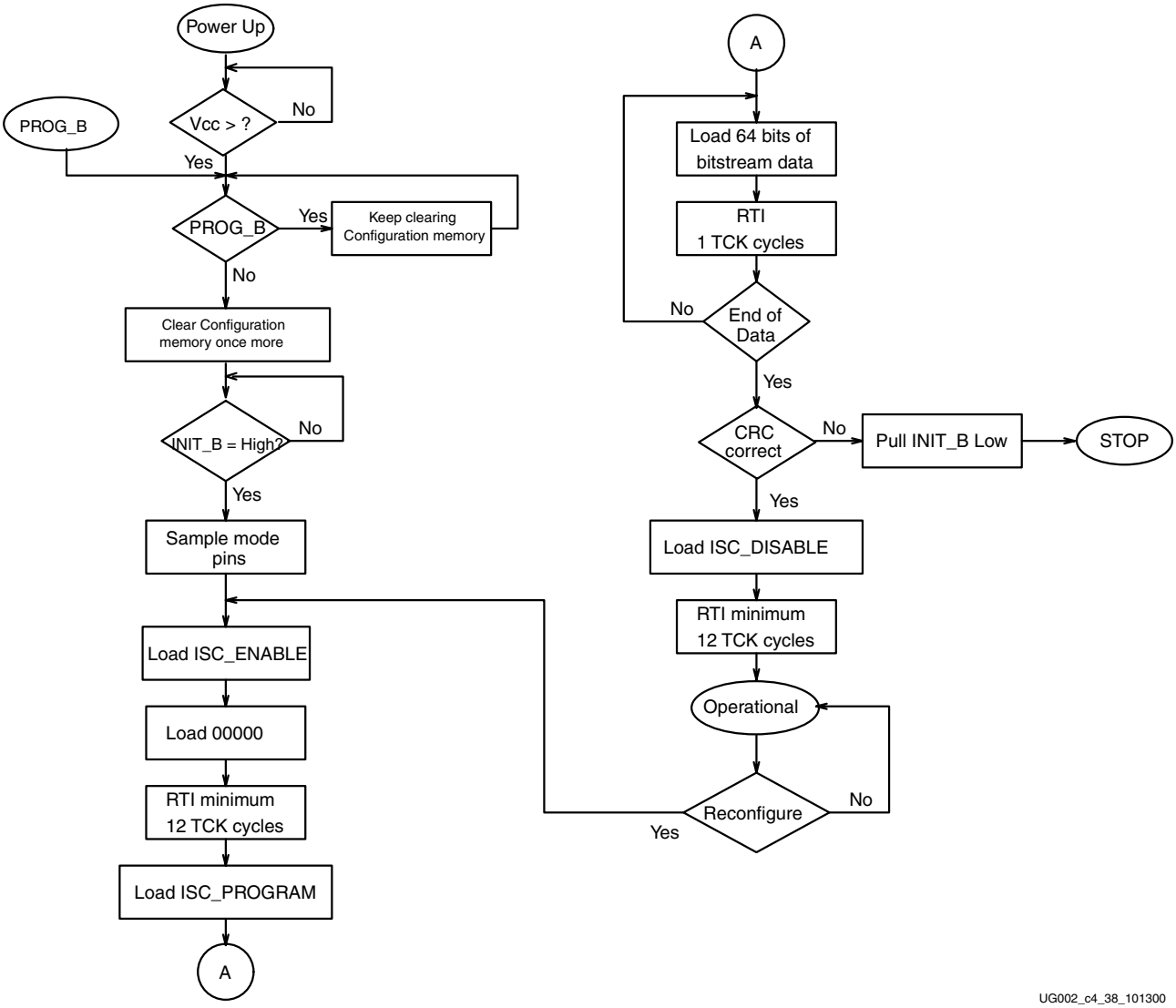

Figure 3-27: **IEEE 1532 Configuration Flow**

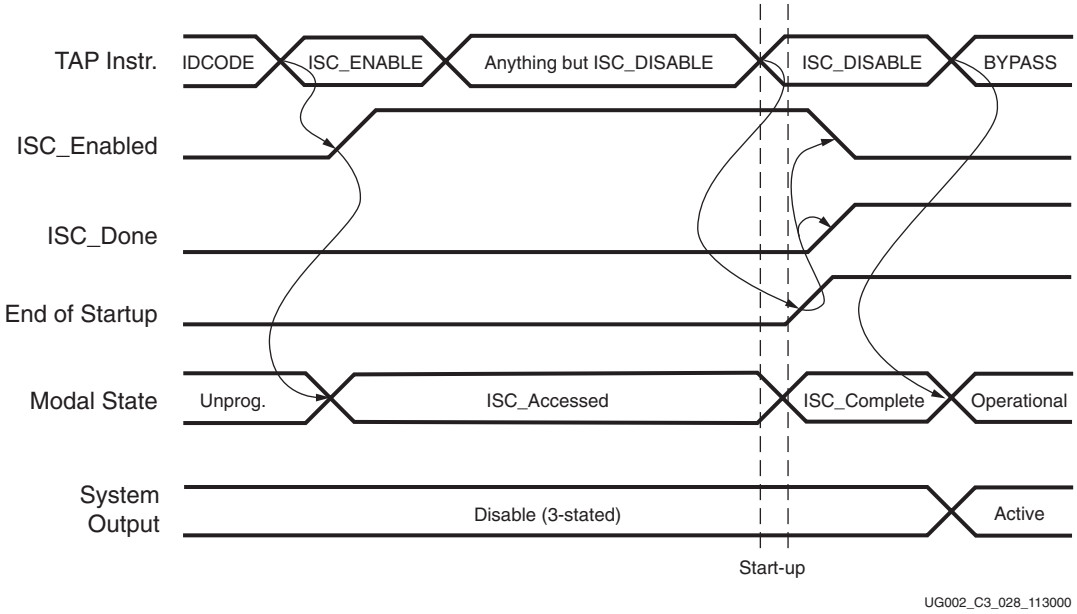

Figure 3-28: **Signal Diagram for Successful First Time ISC Configuration**

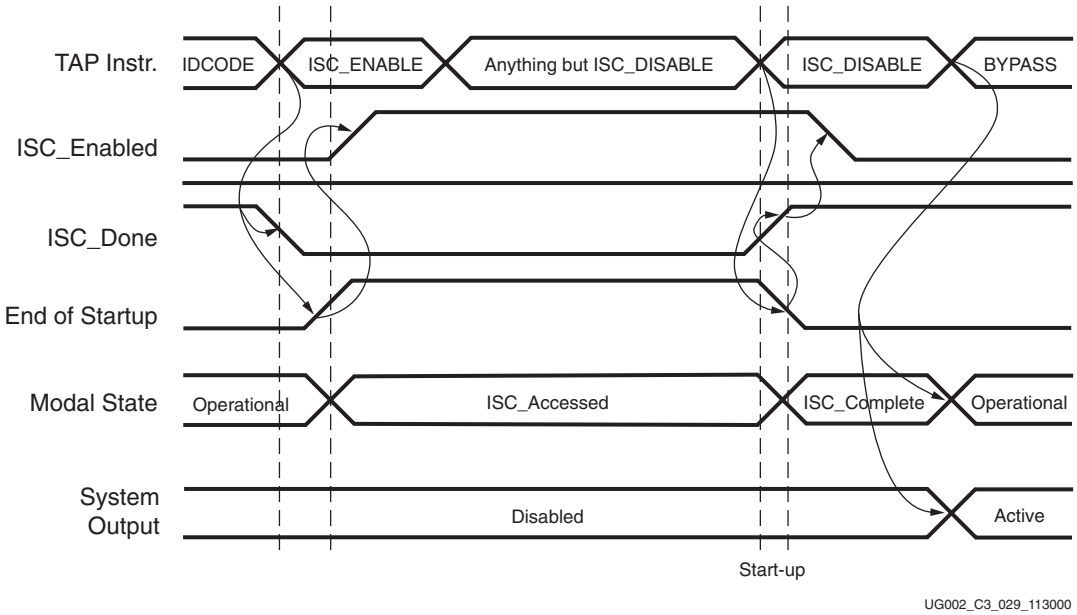

Figure 3-29: **Signal Diagram for Successful ISC Partial and Full Reconfiguration**

Configuration Details

This section provides a bit-level understanding of the configuration stream. For the purpose of debugging, designing embedded readback operations, or otherwise complex styles of configuring multiple FPGAs, the Virtex-II bitstream, internal configuration logic, and internal processing of configuration data are described here.

Data Frames

The internal configuration memory is partitioned into segments called "Frames." The portions of the bitstream that actually get written to the configuration memory are "Data Frames." The number and size of frames varies with device size as shown in Table 3-15. The total number of configuration bits for a particular device is calculated by multiplying the number of frames by the number of bits per frame, and then adding the total number of bits needed to perform the *Configuration Register Writes* shown in Table 3-15.

Table 3-15: **Virtex-II Configuration Data Frames and Programming Times**

Device	No. of Frames	Frame Length in Bits	Configuration Bits	Total No. of Bits (including header)	Approx. SelectMAP Download Time (50 MHz) ms	Approx. Serial Download Time (50 MHz) ms	Approx. JTAG Download Time (33 MHz) ms
XC2V40	404	832	360,096	339,040	0.84	6.72	10.19
XC2V80	404	1472	635,296	598,880	1.49	11.89	18.02
XC2V250	752	2112	1,697,184	1,593,696	3.97	31.76	48.13
XC2V500	928	2752	2,761,888	2,560,608	6.38	51.08	77.39
XC2V1000	1104	3392	4,082,592	3,752,800	9.36	74.90	113.48
XC2V1500	1280	4032	5,659,296	5,170,272	12.90	103.22	156.39
XC2V2000	1456	4672	7,492,000	6,813,024	17.01	136.05	206.13
XC2V3000	1804	5312	10,494,368	9,594,720	23.96	191.66	290.39
XC2V4000	2156	6592	15,659,936	14,226,784	35.53	284.25	430.68
XC2V6000	2508	7872	21,849,504	19,759,968	49.36	394.86	598.27
XC2V8000	2860	9152	29,063,072	26,194,272	65.44	523.49	793.17

Configuration Registers

The Virtex-II configuration logic was designed so that an external source can have complete control over all configuration functions by accessing and loading addressed internal configuration registers over a common configuration bus. The internal configuration registers that are used for configuration and readback are listed in Table 3-16. All configuration data, except the synchronization word and dummy words, is written to internal configuration registers.

Table 3-16: **Internal Configuration Registers**

Symbol	Register Name	Address
CRC	CRC Register	00000
FAR	Frame Address Register	00001
FDRI	Frame Data Input Register (Write Configuration Data)	00010
FDRO	Frame Data Output Register (Readback Configuration Data)	00011
CMD	Command Register	00100

Table 3-16: **Internal Configuration Registers**

Symbol	Register Name	Address
CTL	Control Register	00101
MASK	Masking Register for CTL	00110
STAT	Status Register	00111
LOUT	Legacy Output Register (DOUT for daisy chain)	01000
COR	Configuration Option Register	01001
MFWR	Multiple Frame Write	01010
FLR	Frame Length Register	01011
IDCODE	Product ID Code Register	01110

Command Register (CMD)

Commands shown in Table 3-17 are executed by loading the binary code into the CMD register.

Table 3-17: **CMD Register Commands**

Symbol	Command	Binary Code
WCFG	Write Configuration Data	0001
MFWR	Multi-Frame Write	0010
DGHIGH	De-asserts GHIGH	0011
RCFG	Read Configuration Data	0100
START	Begin STARTUP Sequence	0101
RCAP	Reset CAPTURE (after Single-Shot Capture)	0110
RCRC	Reset CRC Register	0111
AGHIGH	Assert GHIGH	1000
SWITCH	Switch CCLK Frequency	1001
GRESTORE	Pulse GRESTORE Signal	1010
SHUTDOWN	Begin SHUTDOWN Sequence	1011
GCAPTURE	Pulse GCAPTURE Signal (one shot)	1100
DESYNCH	Forces realignment to 32 bits	1101

Frame Length Register (FLR)

The FLR is used to indicate the frame size to the internal configuration logic. This allows the internal configuration logic to be identical for all Virtex-II devices. The value loaded into this register is the number of actual configuration words that get loaded into the configuration memory frames.

Configuration Option Register (COR)

The COR is loaded with the user selected options from bitstream generation. See Appendix B, "BitGen and PROMGen Switches and Options.".

Table 3-18: **Configuration Option Register**

Name	Description	Bits
CRC_BYPASS	Does not check against updated CRC value.	29
SHUT_RST_DCI	DCI resets if SHUTDOWN and AGHIGH are performed.	27
SHUT_RST_DCM	DCM resets if SHUTDOWN and AGHIGH are performed.	26
DONE_PIPE	Add pipeline stage to DONEIN.	25
DRIVE_DONE	DONE pin is an active driver, not open drain.	24
SINGLE	Readback capture is one shot.	23
OSCFSEL	Select CCLK frequency in Master Serial Mode.	22:17
SSCLKSRC	Select STARTUP block clock source.	16:15
DONE_CYCLE	Startup cycle when DONE is asserted/de-asserted.	14:12
MATCH_CYCLE	Stall in this Startup cycle until DCI match signals are asserted.	11:9
LOCK_CYCLE	Stall in this Startup cycle until DCM signals are asserted.	8:6
GTS_CYCLE	Startup cycle when GTS_CFG_B is de-asserted.	5:3
GWE_CYCLE	Startup cycle when GWE is asserted.	2:0

Control Register (CTL)

The CTL controls internal functions such as *Security* and *Port Persistence*.

Table 3-19: **Control Register**

Name	Description	Bits
SBITS	Security level.	4:5
PERSIST	Configuration ports remain after configuration.	3
Reserved	For internal use.	2:1
GTS_USR_B	Active Low global 3-state I/Os. Turns off pullups if GTS_CFG_B is also asserted.	0

Mask Register (MASK)

The MASK is a safety mechanism that controls which bits of the CTL register can be reloaded. Prior to loading new data into the CTL register, each bit must be independently enabled by its corresponding bit in the MASK register. Any CTL bit not selected by the MASK register is ignored when reloading the CTL register.

Frame Address Register (FAR)

The FAR sets the starting frame address for the next configuration data input write cycle.

Frame Data Register Input (FDRI)

The FDRI is the input stage for configuration data frames to be stored in the configuration memory. Starting with the frame address specified in the FAR, the FDRI writes its contents to the configuration memory frames. The FDRI automatically increments the frame

address after writing each frame for the number of frames specified in the FDRI write command. This is detailed in the next section.

CRC Register (CRC)

The CRC is loaded with a CRC value that is embedded in the bitstream and compared against an internally calculated CRC value. Resetting the CRC register and circuitry is controlled by the CMD register.

Frame Data Register Output (FDRO)

FDRO is an output stage for reading frame data from the configuration memory during readback. This works the same as the FDRI but with data flowing in the other direction.

Legacy Data Output Register (LOUT)

LOUT is pipeline data to be sent out the DOUT pin for serially daisy-chained configuration data output.

Status Register (STAT)

The STAT register contains bits that indicate the state of the device. Such bits include the status of error pins, global signals, the DCM, and DCI. This register is read-only and can be read using the JTAG or SelectMAP port for debugging purposes.

																RESERVED	RESERVED	ID_ERROR	DONE	INIT_B	MODE			GHIGH_B	GWE	GTS_CFG_B	IN_ERROR	DCI_MATCH	DCM_LOCK	RESERVED	CRC_ERROR
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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 3-30: **Status Register Fields**

Table 3-20: **Status Register**

Name	Description	Bit Location
ID_ERROR	IDCODE not validated while trying to write FDRI	13
DONE	DONEIN input form DONE pin	12
INIT_B	Value of CFG_RDY (INIT_B)	11
MODE	Value or MODE pins (M2, M1, M0)	10:8
GHIGH_B	Status of GHIGH	7
GWE	Status of GWE	6
GTS_CFG_B	Status of GTS_CFG_B	5
IN_ERROR	Legacy input error	4
DCI_MATCH	DCI matched	3
DCM_LOCK	DCM matched	2
Reserved	For internal use	1
CRC_ERROR	CRC error	0

Configuration Data Processing Flow

The complete (standard) reconfiguration of a Virtex-II device follows the internal flow shown in Figure 3-31. All associated configuration commands are listed in Table 3-21.

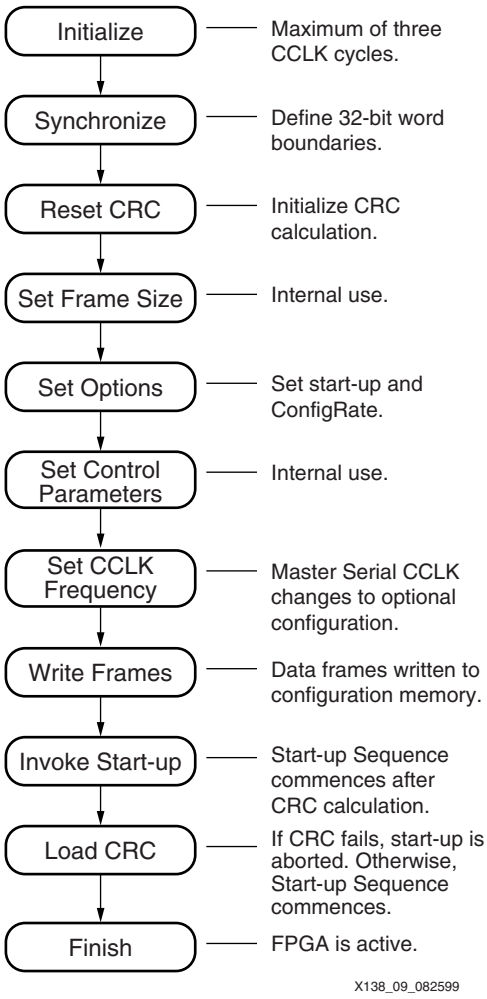

Figure 3-31: **Internal Configuration Processing Flow**

Table 3-21: **Configuration Register Writes**

Type	Number of 32-bit Words
Command Set 1	
Dummy words	1
Synchronization word	1
Write CMD (RCRC)	2
Write FLR	2
Write COR	2
Write ID	2
Write MASK	2
Write CMD (SWITCH)	2
Command Set 2	
Write FAR	2
Write CMD (WCFG)	2
Write FDRI	part size dependent
Write CMD (DGHIGH)	2
Command Set 3	
Write COR	2
Write CMD (START)	2
Write CTL	2
Write CRC	2
Write CMD (DESYNCH)	
Dummy words	4
TOTAL	40

The first command set prepares the internal configuration logic for the loading of the data frames. The internal configuration logic is first initialized with several CCLK cycles represented by dummy words, then it is synchronized to recognize the 32-bit word boundaries by the synchronization word. The CRC register and circuitry must then be reset by writing the RCRC command to the CMD register. The frame length size for the device being configured is then loaded into the FLR register. The configuration options are loaded into the COR. The CCLK frequency selected is specified in the COR; however, to switch to that frequency the SWITCH command must be loaded into the CMD register. The ID register is written to ensure that the correct bitstream is being used. Now the data frames can be loaded.

The second command set loads the configuration data frames. First, a WCFG (Write Configuration) command is loaded into the CMD register activating the circuitry that writes the data loaded into the FDRI into the configuration memory cells. To load a set of data frames, the starting address for the first frame is first loaded to the FAR, followed by a write command, and then by the data frames to the FDRI. The FDRI write command also specifies the amount of data that is to follow in terms of the number of 32-bit words that comprise the data frames being written. When all but the last frame has been loaded, an initial CRC checksum is loaded into the CRC register. The De-assert GHIGH (DGHIGH) is loaded into the CMD register.

The third command set initializes the Start-Up Sequence and finishes CRC checking. After all the data frames have been loaded, the START command is loaded into the CMD register, followed by any internal control data to CTL, the final CRC value into the CRC register, and the DESYNCH command to the CMD register. The four dummy words at the end are flushed through the system to provide the finishing CCLK cycles to activate the FPGA.

Standard Bitstream

Virtex-II devices have the ability to be only partially re-configured or read back. The standard bitstream, currently generated by BitGen, follows the format shown in Table 3-22, Table 3-23, and Table 3-24. *This format assumes D0 is considered the MSB*. It is divided into three tables to follow the three command sets described in the previous subsection.

Table 3-22 shows the first set of commands in the bitstream that prepare the configuration logic for rewriting the memory frames. All commands are described as 32-bit words, since configuration data is internally processed from a common 32-bit bus.

Table 3-22: **Bitstream Header and Configuration Options**

Data Type
Dummy word
Synchronization word
Packet Header: Write to CMD register
Packet Data: RCRC
Packet Header: Write to FLR register
Packet Data: Frame Length
Packet Header: Write to COR
Packet Data: Configuration options (user defined)
Packet Header: Write to ID register
Packet Data: IDCODE
Packet Header: Write to CMD register
Packet Data: SWITCH
Packet Header: Write to CMD register
Packet Data: WCFG

From Table 3-22, the first dummy word pads the front of the bitstream to provide the clock cycles necessary for initialization of the configuration logic. No actual processing takes place until the synchronization word is loaded. Since the Virtex-II configuration logic processes data as 32-bit words, but can be configured from a serial or 8-bit source, the synchronization word is used to define the 32-bit word boundaries. That is, the first bit after the synchronization word is the first bit of the next 32-bit word, and so on.

After synchronization, all data (register writes and frame data) are encapsulated in *packets*. There are two kinds of packets, Header and Data. A header packet has two types: Type 1 and Type 2. Type 1 Packet Headers are used for register writes. A combination of Type 1 and Type Packet Headers are used for frame data writes. A Type 1 Packet Header, shown in Figure 3-32, is always a single 32-bit word that describes the header type, whether it is a read/write function to a specific configuration register address (see Table 3-16) as the destination, and how many 32-bit words are in the following Packet Data portion. A Type 1 Packet Data portion can contain anywhere from 0 to 2,047 32-bit data words.

Packet Header	Type	Operation (Write/Read)	Register Address (Destination)	Byte Address	Word Count (32-bit Words)
Bits[31:0]	31:29	28:27	26:13	12:11	10:0
Type 1	001	10 / 01	XXXXXXXXXXXXXX	XX	XXXXXXXXXXX

X138_10_082599

Figure 3-32: **Type 1 Packet Header**

Packet Header	Type	Operation (Write/Read)	Word Count (32-bit Words)
Bits[31:0]	31:29	28:27	26:0
Type 2	010	10 / 01	XXXXXXXXXXXXXXXXXXXXXXXXXXX

X138_11_082599

Figure 3-33: **Type 2 Packet Header**

The first packet header in Table 3-22 is a Type 1 packet header that specifies writing one data word to the CMD register. The following packet data is a data word specifying a reset of the CRC register (compare the data field of Table 3-22 to the binary codes of Table 3-17).

The second packet header in Table 3-22 loads the frame size into the FLR.

The third packet header loads the configuration options into the COR register. The binary description of this register is not documented. Following this is a similar write of the SWITCH command to the CMD register which selects the CCLK frequency specified in the COR. Finally, the WCFG command is loaded into the CMD register so that the loading of frame data can commence.

The fourth packet header writes to the ID register. This ensures the correct bitstream for the correct Virtex-II family member.

Table 3-23 shows the packets that load all of the data frames, starting with a Type 1 packet header to load the starting frame address, which is always 0h.

Table 3-23: **Bitstream Data Frames and CRC Sequence**

Data Type
Packet Header: Write to FAR register
Packet Data: Starting frame address
Packet Header: Write to FDRI
Packet Header Type 2: Data words
Packet Data: Configuration data frames in 32-bit words. Total number of words specified in Type 2 Packet Header
Packet Data: CRC value
Packet Header: Write to CMD register
Packet Data: GRESTORE
Packet Header: Write to CMD register
Packet Data: DGHIGH
Packet Header: NO OP
Packet Data: one frame of NO OP

The loading of data frames requires a combination of Type 1 and Type 2 packet headers. Type 2 packet headers must always be preceded by a Type 1 packet header. The Type 2 packet data can be up to 67,108,863 data words in size.

The Type 2 packet header, shown in Figure 3-33, differs slightly from a Type 1 packet header in that there is no Register Address or Byte Address fields.

To write a set of data frames to the configuration memory, after the starting frame address has been loaded into the FAR, a Type 1 packet header issues a write command to the FDRI, followed by a Type 2 packet *header* specifying the number of data words to be loaded, and then followed by the actual frame data as Type 2 packet *data*. Writing data frames might require a Type 1/Type 2 packet header combination, or a Type 1 only. This depends on the amount of data being written.

Table 3-24 shows the packets needed to issue the start-up operations and load the final CRC check. The FPGA does not go active until after the final CRC is loaded. The number of clock cycles required to complete the start-up sequence depends on the BitGen options selected. Completion of the configuration process requires 8 to 16 clock cycles after the DESYNCH command. The DESYNCH command forces realignment to 32-bit boundaries and, therefore, a synchronization word is needed.

Table 3-24: **Bitstream Final CRC and Start-Up Sequence**

Data Type
Packet Header: Write to CMD register
Packet Data: START
Packet Header: Write to MASK
Packet Data: CTL mask
Packet Header: Write to CTL
Packet Data: Control commands
Packet Header: Write to CRC
Packet Data: CRC value
Packet Header: Write to CMD
Packet Data: DESYNCH command
Dummy word
Dummy word
Dummy word
Dummy word

Typically, DONE is released within the first seven CCLK cycles after the final CRC value is loaded, but the rest of the dummy data at the end of the stream should continue to be loaded. The FPGA needs the additional clock cycles to finish internal processing, but this is not a concern when a free-running oscillator is used for CCLK. In serial mode, this requires only 16 bits (two bytes), but in SelectMAP mode, this requires 16 bytes of dummy words at the end of the bitstream. Since the intended configuration mode to be used is unknown by Bitgen, four 32-bit dummy words (16 bytes) are always placed at the end of the bitstream.

Cyclic Redundancy Checking Algorithm

Virtex-II configuration uses a standard 16-bit CRC checksum algorithm to verify bitstream integrity during configuration. The 16-bit CRC polynomial is shown below.

$$CRC\text{-}16 = X^{16} + X^{15} + X^{2} + 1$$

The algorithm is implemented by shifting the data stream into a 16-bit shift register, shown in Figure 3-34. Register Bit(0) receives an XOR of the incoming data and the output of Bit(15). Bit(2) receives an XOR of the input to Bit(0) and the output of Bit(1). Bit(15) receives an XOR of the input to Bit(0) and the output of Bit(14).

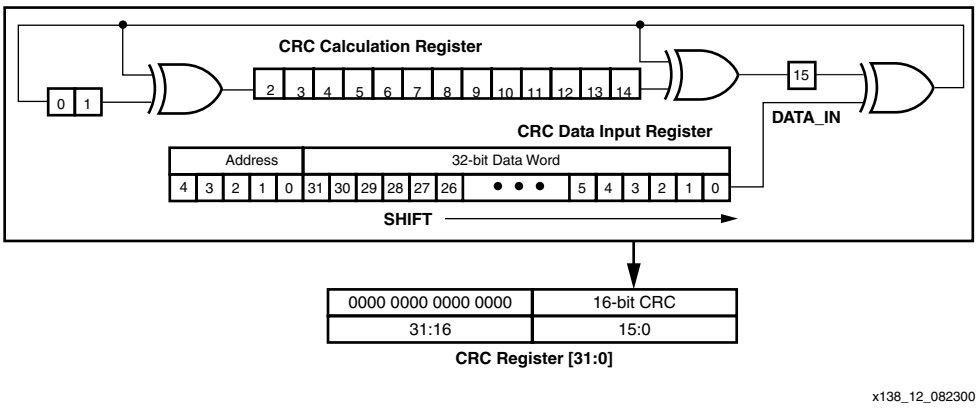

Figure 3-34: **Serial 16-bit CRC Circuitry**

A CRC Reset resets all the CRC registers to zero. As data is shifted into the CRC circuitry, a CRC calculation accumulates in the registers. When the CRC value is loaded into the CRC calculation register, the ending CRC checksum is loaded into the CRC Register. The value loaded into the CRC Register should be zero; otherwise, the configuration failed CRC check.

Not all of the configuration stream is loaded into the CRC circuitry. Only data that is written to one of the registers shown in Table 3-21 is included. For each 32-bit word that is written to one of the registers (Table 3-21), the address code for the register and the 32-bit data word is shifted LSB first into the CRC calculation circuitry, see Figure 3-34. When multiple 32-bit words are written to the same register, the same address is loaded after each word. All other data in the configuration stream is ignored and does not affect the CRC checksum.

This description is a model that can be used to generate an identical CRC value. The actual circuitry in the device is a slightly more complex Parallel CRC circuit that produces the same result.

Readback

Readback is the process of reading all the data in the internal configuration memory. This can be used to verify that the current configuration data is correct and to read the current state of all internal CLB and IOB registers as well as the current LUT RAM and block RAM values.

Readback is only available through the SelectMAP and Boundary Scan interfaces. This discussion covers the use of the SelectMAP interface for performing readback. For information on using the Boundary Scan interface for readback see "Readback When Using Boundary Scan" on page 298.

Readback Verification and Capture

Readback verification is used to verify the validity of the stored configuration data. This is most commonly used in space-based applications where exposure to radiation might alter the data stored in the configuration memory cells.

Readback capture is used to list the states of all the internal flip-flops. This can be used for hardware debugging and functional verification. When Capture is initiated, the internal register states are loaded into unused spaces in the configuration memory which can be extracted after a readback of the configuration memory.

While both *Verify* and *Capture* can be performed in one readback, each require slightly different preparation and post processing.

Preparing for Readback in Design Entry

If only a readback verification is to be performed, there are no additional steps at the time of design entry. However, if readback capture is to be used, the Virtex-II library primitive CAPTURE_VIRTEX2 must be instantiated in the user design as shown in Figure 3-35.

The CAPTURE_VIRTEX2 component is used in the FPGA design to control when the logic states of all the registers are captured into configuration memory. The CLK pin can be driven by any clock source that would synchronize Capture to the changing logic states of the registers. The CAP pin is an enable control. When CAP is asserted, the register states are captured in memory on the next CLK rising edge.

Capture can be performed in two ways: single-shot or continuous. In continuous capture, the CAP line is held High until the desired capture event occurs causing CAP to go Low. See Figure 3-35. Continuous capture does not require a readback operation to reset the CAPTURE block. In single-shot capture, the CAP line is pulsed once, and subsequent pulses are ignored until a readback operation has been performed. Captured data is read using the same process as a normal readback.

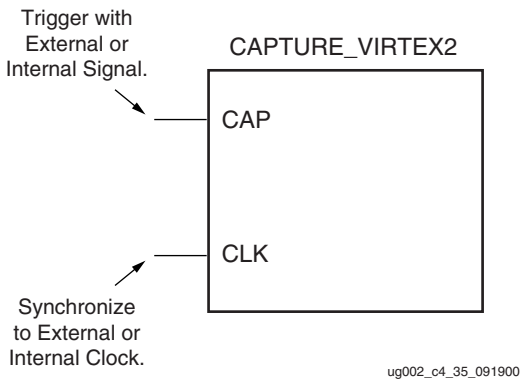

Figure 3-35: **Readback CAPTURE_VIRTEX2 Library Primitive**

Enabling Readback in the Software

Since readback is performed through the SelectMAP interface after configuration, the configuration ports must continue to be active by setting the persistence switch in BitGen. Additionally, a readback bit file, which contains the commands to execute a readback and a bitmap for data verification, can optionally be generated by setting the readback option in BitGen. An example of the BitGen command line is shown below.

```
bitgen -w -l -m -g readback -g persist:yes...
```

The **-w** option overwrites existing output. The **-l** option generates a *Logic Allocation* file. The **-m** option generates a *Mask* file. The **-g readback** option generates a *readback bit* file, and the -**g persist:yes** option keeps the SelectMAP interface active after configuration. For more information on BitGen options, see Appendix B, "BitGen and PROMGen Switches and Options."

Readback When Using Boundary Scan

Regular Readback Flow

It is highly recommended to perform shutdown before reading back bitstream to ensure normal operation. The Shutdown Sequence can be executed by loading the JSHUTDOWN instruction and spending at least 12 TCK cycles in RTI TAP controller state. CRC_ERROR status and configuration error (CFGERR) must be cleared after readback by issuing Reset CRC bitstream command or writing the correct CRC value to CRC register.

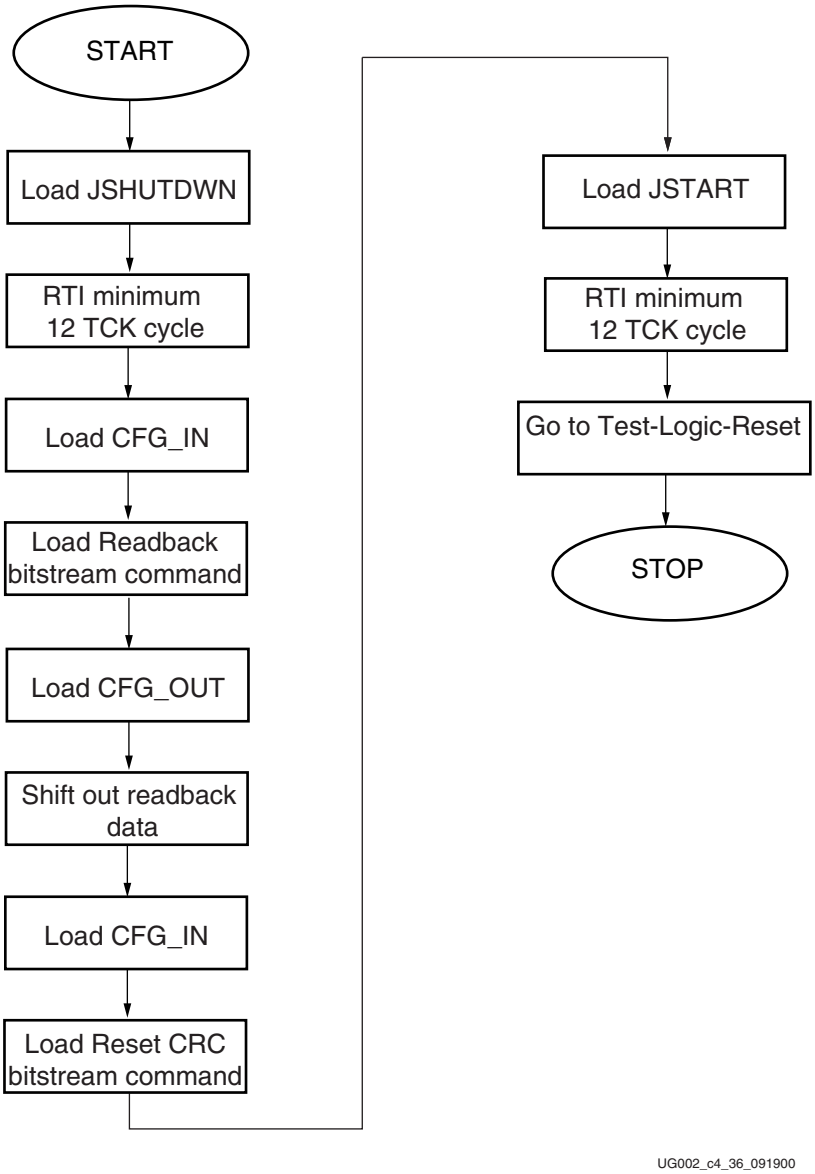

Figure 3-36: **Regular Readback Flow**

IEEE 1532 Readback Flow

In IEEE 1532 readback mode, full chip shutdown is performed when ISC_ENABLE is executed. At the end of readback, CRC Error status must be cleared by issuing Reset CRC command or writing the correct CRC value to CRC register. ISC_DISABLE cannot be executed correctly unless the CRC error status is cleared.

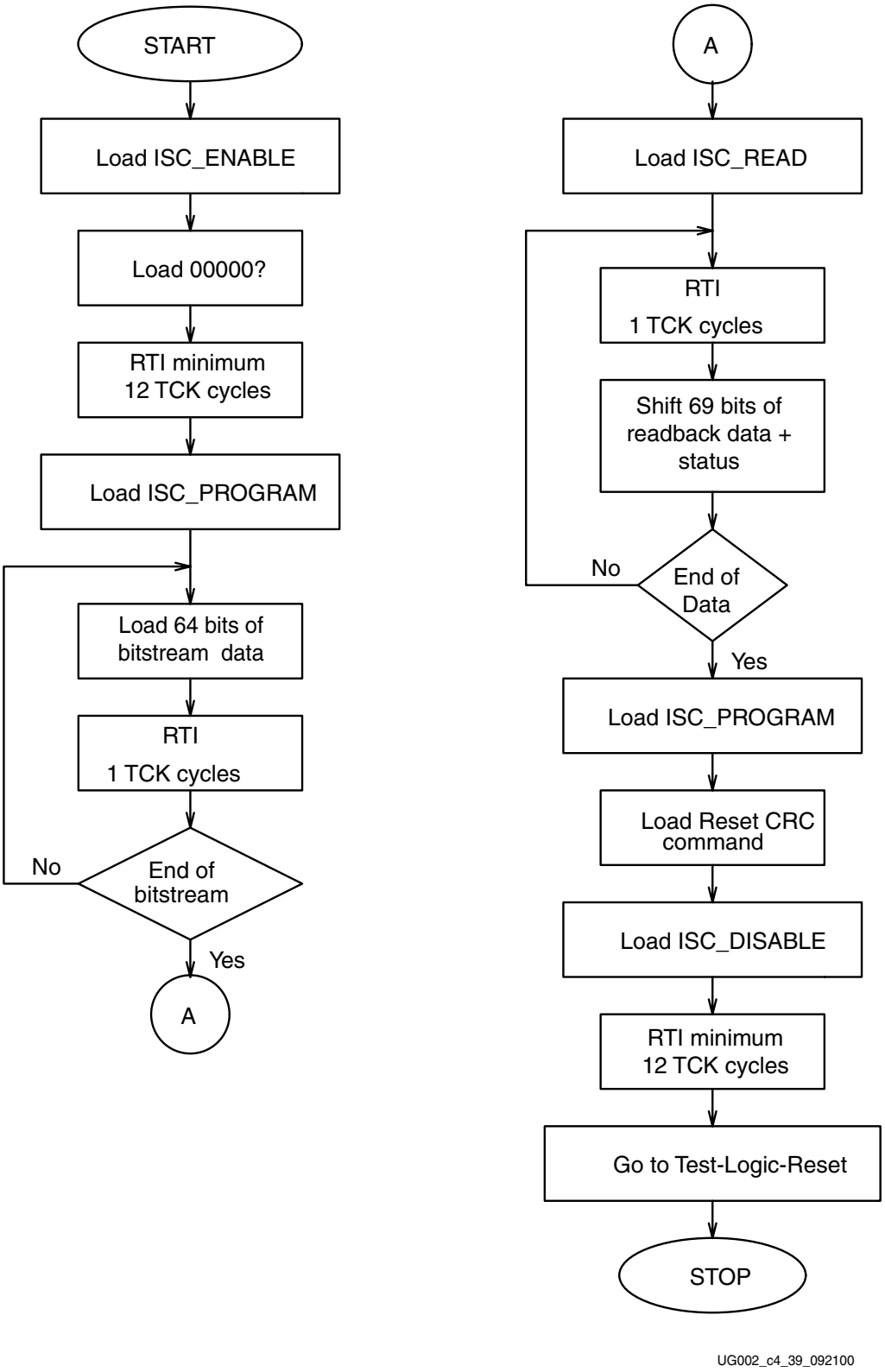

Figure 3-37: **IEEE 1532 Readback Flow**

Using ChipScope ILA

The ChipScope ILA functional verification tool is currently sold separately through the Xilinx web site. This program uses a combination of PC software and instantiated soft cores to capture states of internal signals. This data is read out of the JTAG USER1 scan chain using the MultiLINX cable or a parallel cable. ChipScope ILA supports only the Virtex architecture and allows for the functional verification and debugging of an FPGA configured design.

ChipScope ILA supports the high speed USB interface to the MultiLINX cable set on Windows 98/2000 platforms and the RS232 connection on Windows 95/98/2000/NT platforms. UNIX support is not available. More details are available under ChipScope ILA at: **www.xilinx.com**