Virtex-6 FPGA Configuration

User Guide

UG360 (v3.9) November 18, 2015

XILINX®

Notice of Disclaimer
The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials, or to advise you of any corrections or update. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at www.xilinx.com/legal.htm#tos.

© Copyright 2009–2015 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. PCI, PCI Express, PCIe, and PCI-X are trademarks of PCI-SIG. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/09	1.0	Initial Xilinx release.
09/16/09	1.1	Chapter 1: Revised the fourth paragraph in FPGA Density Migration, page 23. Chapter 2: Removed note about Master Serial mode under Figure 2-2. Added Platform Flash XL (XCF128X) under the bullet entitled Single-device Slave SelectMAP, page 32. Revised the first paragraph under High-Performance Platform Flash XL SelectMAP Configuration, page 34. Removed Figure 2-6 (Platform Flash XL High-Performance FPGA Configuration) and added a reference to DS617. Removed Figure 2-7 (Indirect Programming Solution for Platform Flash XL) and preceding paragraph. Revised Note 12 under Figure 2-6. Replaced V_{POR} with "minimum recommended operating voltage" in the fifth paragraph of BUSY, page 39. In Table 2-6, page 44, corrected typo for M[2:0] encoding. Chapter 4: Revised description of BSCAN_VIRTEX6 instance in first paragraph of BSCAN_VIRTEX6, page 69 and added attribute table (Table 4-2). Added EFUSE_USR primitive to EFUSE_USR, page 71. Added sentence about ONESHOT attribute to second paragraph of CAPTURE_VIRTEX6, page 70 and added attribute table (Table 4-4). In Table 4-10, revised Type and Settings values for FARSRC. In Table 4-11, revised descriptions of ECCERROR and SYNDROME[12:0] and changed CRC_ERROR to CRCERROR. Changed CS_B to CSB and RDWR_B to RDWRB in Table 4-11. Chapter 5: In Table 5-1, added sentence about DCLK pulsing to DEN and DWE descriptions. Chapter 6: Added HXT devices to Table 6-5, page 87 and Table 6-13, page 97. Revised Note 1 in Table 6-5. Inserted new paragraph about bitstream data before Table 6-6. Revised fourth paragraph in Device Power-Up (Step 1), page 92 with -1L sources and dual-mode pin supply voltages. Removed specifications from Table 6-10, revised V_{CCO_24} and V_{CCO_34} descriptions, and referred to the data sheet. Replaced V_{POR} with minimum recommended voltages in paragraph under Figure 6-4. Changed the format and definitions for the JTAG ID Code register in Check Device ID (Step 5), page 96. Changed ID_Err to ID_ERROR in Table 6-14. Revised second paragraph of Loading Encrypted Bitstreams, page 103. Added new section Bitstream Authentication, page 105. Removed the VFS Pin Bias Conditions table from VFS Pin, page 110 and referred to the data sheet for this information. Changed CS_B to CSI_B in SelectMAPAbort description of Table 6-30. Chapter 7: Changed CS_B to CSI_B throughout the chapter. In Configuration Register Read Procedure (SelectMAP), page 133, revised step 1 and step 3. In Configuration Memory Read Procedure (SelectMAP), page 134, corrected FDRO read length in step 8. Revised Note 1 in Table 7-3. Chapter 10: Updated flash pin names in Figure 10-4.

Date	Version	Revision
11/15/09	2.0	Globally changed ADDR to A and CFG_CLK to CFGCLK. Chapter 1: Changed Conforming to PCI Express Link Activation Requirements, page 22 to reflect PCI Express applications, not PCI applications. Included references to UG517. Revised Required I/O Voltages, page 22. Revised the last three paragraphs in FPGA Density Migration, page 23. Revised second paragraph of Protecting the FPGA Bitstream against Unauthorized Duplication, page 24 on the different techniques. Chapter 2: Added sentence to the end of Note 1 in Table 2-1. Revised descriptions of CCLK and INIT_B in Table 2-2. Deleted Note 6 following Figure 2-3 about CSO_B and MOSI. Clarified note about DIN data bits in Figure 2-4. Revised Note 2 following Figure 2-4. Clarified bus widths in description of D[31:0] in Table 2-3. Removed "at speeds to 800 Mb/s" from the first paragraph of High-Performance Platform Flash XL SelectMAP Configuration, page 34. Added CSI_B and RDWR_B and changed DOUT to BUSY in Figure 2-6. After Figure 2-6: Deleted Note 11 (data bus width in Master SelectMAP configuration) and Note 12 (Platform Flash PROMs supporting SelectMAP configuration modes). In Figure 2-7, added VCCO_34. After Figure 2-7: Revised Note 2 about supply inputs, and revised Note 9 about data bus widths with AES encrypted bitstreams. Changed D[0:n] to D[0:7] in Figure 2-8. Deleted Note 3 following Figure 2-9 about the CSI_B waveform. Added RCMD[7:0] as an input and moved CCLK as an output in Figure 2-11. Revised the description of FS[2:0] in Table 2-6. Moved Table 2-7 and changed its caption. Added third paragraph to Master SPI Configuration Interface, page 43. Tied FS1 to VCCO_0 in Figure 2-12. After Figure 2-12, deleted Note 11 about HSWAPEN and Note 14 about DCI match wait and MMCM lock wait. Revised the first bulleted item and removed reference to XAPP951 in Power-On Sequence Precautions, page 49. Revised description of D[15:0] in Table 2-8. Added RS[1:0] to Figure 2-15. After Figure 2-15, deleted Note 5 about FCS_B, FOE_B, and FWE_B; revised Note 10; and added Note 14. Revised JTAG Interface, page 58. Chapter 3: Removed "in full compliance with the standard" from Boundary-Scan for Virtex-6 Devices Using IEEE Standard 1149.1, page 63. Revised Note 1 in Table 3-1. Deleted "Data to be captured" and "Data to be driven out" waveforms in Figure 3-1. Revised second paragraph in Multiple Device Configuration, page 65 regarding the connections of PROGRAM_B, INIT_B, and DONE. Chapter 4: Removed "indirect SPI flash programming" from sentence following Table 4-13. Chapter 6: Revised Note 2 in Table 6-1. Revised description of D[7:0] in Table 6-2. Deleted file extension `TEK` and added Note 3 to Table 6-4. Revised first and last paragraphs of Bit Swapping, page 90. Revised last sentence of Implementation, page 106. Revised last paragraph of No On-Chip Key Storage for the HMAC Key is Required, page 106. Added "issue a JPROGRAM or IPROG command" as an option to the last paragraph of eFUSE Control Register (FUSE_CNTL), page 108. Revised first paragraph of JTAG Instructions, page 109. Revised Code column of Table 6-22. Added Table 6-23. In Table 6-40, changed "Fallback" to "Next" in description of START_ADDR. Added reference to DS152 above Table 6-45. Deleted section entitled "Identifier Memory Specifications" following Operation, page 127. Chapter 7: In Configuration Memory Read Procedure (SelectMAP), page 134, removed "+1" from the FDRO read length from step 8. Removed table note about assumptions from Table 7-3. Changed BOUNDARY_SCAN to BOUNDARY in first paragraph of Accessing Configuration Registers through the JTAG Interface, page 136. Revised last paragraph of Verifying Readback Data, page 142. Chapter 8: Revised first paragraph in chapter. Revised "Pins Controlled by WBSTAR" column for Master SPI Configuration Mode in Table 8-2. Revised sentence following Table 8-2. Chapter 10: In Table 10-1, row 2, added Spartan-6 architecture and changed 32 to 16 in the Maximum DOUT Bits column. Following Figure 10-4, added Note 10, indicating Fallback MultiBoot not supported in this configuration. Chapter 11: Revised last paragraph of Device Identification (IDCODE) Register, page 180. Deleted "Data to be captured" and "Data to be driven out" waveforms in Figure 11-4. Revised last paragraph of Configuring through Boundary-Scan, page 181. Revised flowchart in Figure 11-5. In Table 11-4: Deleted Step 12 about resetting the TAP; in the new Step 18, changed 12 to 2000 in the Description column and changed 12 clocks to 2000 in the # of Clocks column; and added Note 2. In Multiple Device Configuration, page 184: Deleted Step 5 about resetting the TAPs; and in the new Step 6, changed 12 to 2000 times.

Date	Version	Revision
01/18/10	3.0	Changed the CCLK direction in Table 2-2, Table 2-3, Table 2-6, and Table 2-8. Changed the direction and the description of INIT_B in Table 2-2, Table 2-3, Table 2-6, Table 2-8, Table 6-1, and Table 6-18. Changed all references to INIT to INIT_B. Changed register names: Boundary Scan to Boundary and Identification to Device Identification. Chapter 1: Revised the second paragraph of Required I/O Voltages on third-party flash. Chapter 2: Removed reference to the description of DONE in Table 2-2. Changed Note 7 on page 29, Note 5 on page 31, Note 10 on page 36, and Note 8 on page 38 about CCLK signal integrity. Changed Note 2 on page 31 about CCLK in Master configuration mode. Added sentence to the end of the CCLK description in Table 2-3. Clarified that CSO_B is three-statable in Table 2-3 and Table 2-8. Added BUSY to Figure 2-7. In RDWR_B, page 38, changed CSI_B to "asserted" in the paragraph under the bulleted list and revised the paragraph about changing the value of RDWR_B. Simplified the description of BUSY in BUSY. Added sentence to the end of the FCS_B description in Table 2-6. Changed Note 13 on page 48 about an external pull-up on the DONE pin. Replaced Note 1 and 2 on page 49 under Figure 2-13. Changed Dual to Dual-Purpose in Table 2-8. Removed statement about 3-stated after configuration from the FCS_B, FOE_B, FWE_B, RS[1:0], and CSO_B descriptions in Table 2-8. In the paragraph under Table 2-8 on page 52, changed the second sentence and removed the sentence about CCLK parallel termination from the paragraph. Replaced Figure 2-15. Under Figure 2-15, Added sentence about indirectly programming the flash to Note 6; deleted Note 7 through Note 9 about HSWAPEN; deleted Note 11 referring to Chapter 8; and added Note 7 and Note 12 through Note 18. Replaced Figure 2-16. Under Figure 2-16, Replaced all notes except Note 15, which was modified. Added Determining the Maximum Configuration Clock Frequency, page 56. Chapter 4: Revised the RESET pin description in Table 4-1. Removed "after INIT_B goes High" from the description of SIM_EFUSE_VALUE in Table 4-8. Removed references to cores in fourth paragraph of FRAME_ECC_VIRTEX6. Added FRAME_RBT_IN_FILENAME to Table 4-10. Added a third exception to the first paragraph in ICAP_VIRTEX6. Added DEVICE_ID and SIM_CFG_FILE_NAME to Table 4-12. Changed the PREQ and PACK descriptions in Table 4-13. Changed the DATAVALID description in Table 4-15. Chapter 6: Revised the description of DOUT_BUSY in Table 6-1. Appended a sentence to the end of Bus Width Auto Detection on page 89 about when the bus width is detected. In the second paragraph under Figure 6-4, removed the last sentence about POR retriggering if voltage plane dips below a specified level. In Table 6-12, added a sentence on bus width capture to the IWIDTH Access column. In the first paragraph of VBATT, page 105, removed last sentence on limitation of battery endurance. In the second paragraph of No On-Chip Key Storage for the HMAC Key is Required, added parallel chaining to the last sentence. Reorganized the second paragraph of VFS Pin, page 110. Removed numerical value of bitstreams in Note 3 on page 111 of Table 6-23. Added a sentence to the note about the Reserved bits in Table 6-24. Added note about the reserved bits to Table 6-26. Changed bit 9 to Reserved in Table 6-29 and removed it from Table 6-30. In Table 6-30, assigned the default values for bits 31, 7, and 0; added encodings for bits 12, 10, 8, and 6; and added to the SBITS description. In Table 6-38, assigned the default values for bits [3:2] and [1:0]; and added encodings for bits 17, 9, and 8. In Table 6-40, assigned the default values for bits [28:27], 26, and [25:0]. In Table 6-42, assigned the default values for bits 25, 24, and [23:0], and revised the CFG_MCLK range in the TIMER_VALUE description. Added a note to Table 6-44 on the default power state. Removed the "Default Initial Configuration Process" section. Changed the ISE tool version in iMPACT Access to Device Identifier. Chapter 7: In the first paragraph after Figure 7-5, removed the end of the last sentence on readback for Virtex devices. Chapter 8: In the second paragraph of Fallback Overview, page 147, removed the sentence about indirect BPI programming and the RS pin. In Note 4 under Figure 8-2, deleted bullet about address in SPI mode starting from 0. In Watchdog, page 152, removed the range from the 50 MHz nominal frequency. Renamed User Monitor Mode. Chapter 9: Changed CRC_ERROR bit to CRCERROR bit. Rephrased first paragraph of Post_CRC Constraints to indicate there are multiple constraints. In POST_CRC, corrected the name of the POST_CRC_INIT_FLAG constraint in the first paragraph and indicated the default value. In POST_CRC_INIT_FLAG, corrected the POST_CRC_INIT_FLAG constraint name in the first sentence of the third paragraph and indicated the default value. In Syntax Examples, corrected the POST_CRC_INIT_FLAG constraint name and merged the examples. Chapter 10: Changed Note 7 on page 162, Note 8 on page 164, Note 6 on page 166, Note 5 on page 167, and Note 10 on page 169 about CCLK signal integrity. Revised the DATA[0:7] and BUSY waveforms in Figure 10-6 and Figure 10-7. Changed the values in rows 2 and 4 of the typical sequence on page 171. In the second paragraph of SelectMAP Reconfiguration, removed the sentence about some pins not being available when the Persist option is on. Chapter 11: In Bit Sequence Boundary-Scan Register, clarified what the BSDL files represent and added a paragraph on the BSDLAnno utility. In the second paragraph of Instruction Register, removed sentences about the IR length being device specific and about the least-significant six bits of the instruction code.

Date	Version	Revision
07/30/10	3.1	Chapter 1: Changed the conditions for when the mode pins can be toggled in Overview under the bulleted list on page 17. Chapter 2: In Note 7 under Figure 2-2, Note 5 under Figure 2-3, Note 10 under Figure 2-6, and Note 8 under Figure 2-7, added sentences about termination topology. In Slave Serial Configuration, added sentences about the need to set the PERSIST bit in partial reconfiguration applications. Revised, reorganized, and added to content of Master SPI Configuration Interface. In Figure 2-12, changed series resistor between DIN and Q to a pull-down resistor. Under Figure 2-12, removed quick switches from Note 2 and added Note 5. In Figure 2-15, changed voltage values (VCCAUX tied to VCCAUX, VREF tied to VREF, and WP pulled up to VCCO_0). Under Figure 2-15 in Note 13 on page 54, clarified what JTAG mode ensures. In Board Layout for Configuration Clock (CCLK), changed mandatory language to recommended, moved the recommendation to use an IBIS simulator, removed guideline about termination at the end of the CCLK transmission line, and added reference to UG373. Chapter 3: In Configuring through Boundary-Scan, revised the first paragraph on mode pin settings. Chapter 4: In USR_ACCESS_VIRTEX6, corrected the spelling of USRCCLKO and USR_ACCESS_VIRTEX6. Chapter 6: Added table note to Table 6-13. In the paragraph under Table 6-16, removed references to MMCM lock. In Table 6-17, revised the MMCM_LOCK description. In Table 6-20, corrected register name to FUSE_CNTL. In eFUSE Control Register (FUSE_CNTL): corrected register names to FUSE_CNTL and FUSE_USER and clarified Comments for read_en_b_user. In Table 6-21, removed the LX40T row. In Table 6-25, corrected spelling of NOOP. In Table 6-29, changed state of bit 12 from x to 0. In Table 6-33 and Table 6-34, added EFUSE_BUSY, BAD_PACKET, HSWAPEN_B, and SYSMON_OVER_TEMP. In Table 6-35 and Table 6-36, changed LOCK_CYCLE, bits [8:6], to reserved. In Table 6-38, indicated default for RBCRC_ACTION. Chapter 7: In Configuration Register Read Procedure (SelectMAP), changed the number of NOOP commands to two from one in step 1 and added the addition NOOP command to Table 7-1. In Table 7-1, inserted additional NOOP after step 6, renumbered remaining steps, and removed Write from the explanation of step 8. Changed the step numbers in the sentence under Table 7-1. Chapter 8: In MultiBoot Bitstream Spacing and FPGA End of Startup, removed references to MMCM lock timing. Chapter 10: In the first paragraph of SelectMAP ABORT, added “as sampled by CCLK” to the end of the first sentence, and added a reference to Figure 2-10 to the end of the last sentence. In step 2 of Configuration Abort Sequence Description and step 2 of Readback Abort Sequence Description, clarified that RDWR_B is pulled High or Low synchronous to CCLK. Chapter 11: In TAP Controller and Architecture in the definition of Exit2-DR on page 175, changed the states to Shift-DR and Update-DR.

Date	Version	Revision
11/01/10	3.2	Chapter 2: In Power-On Sequence Precautions, page 49 and Power-On Sequence Precautions, page 56, added discussion about PROGRAM_B to the last paragraph and removed the bulleted item about holding the PROGRAM_B pin Low from power-up to delay the start of the FPGA configuration procedure. In Power-On Sequence Precautions, page 49, added "using an open-drain driver" to the last bulleted item. Added Table 2-9. Changed $T_{MCCKTOL}$ to $F_{MCCKTOL}$ in the paragraph above Equation 2-1. Chapter 6: In Delaying Configuration, deleted the bulleted item about holding PROGRAM_B Low and removed the PROGRAM_B row from Table 6-18. Removed the phrase about holding PROGRAM_B Low from the second sentence under Figure 6-4. Removed first row about waiting for MMCMs to lock from Table 6-15. Added sentence to end of first paragraph of Creating an Encrypted Bitstream about creating bitstreams with compression and encryption. In Loading Encrypted Bitstreams, changed the first sentence of the fourth paragraph on page 103 on partial reconfiguration. In Table 6-20, changed the Size and Contents columns for the FUSE_ID register row. Chapter 7: Replaced Figure 7-2. Chapter 8: In the second to the last sentence of the second paragraph in Fallback Overview, inserted text to clarify all configuration modes. Chapter 10: In note 9 under Figure 10-4, added phrase about the BitGen **SecAll** setting OFF (default). Added sentence about the drive strength for the PERSIST pins to the second paragraph of SelectMAP Reconfiguration.
11/03/11	3.3	Added this statement to indicate that the version 3.2 revision to this document incorporated the changes identified in XCN11009. Chapter 2: Updated description of PROGRAM_B in Table 2-2, Table 2-3, Table 2-6, and Table 2-8. Added VFS and VBATT to Figure 2-2 and added notes 9 and 10. Added VFS and VBATT to Figure 2-3 and added notes 6 and 7. Updated Figure 2-6 and added notes 12 and 13. Updated Figure 2-7 and added notes 10 and 11. Updated Figure 2-12 and added notes 15 and 16. Updated Figure 2-15, updated note 10 and added note 19. Chapter 4: Updated descriptions of all pins in Table 4-13. Chapter 6: Updated description of PROGRAM_B in Table 6-1. Added FPGA I/O Pin Settings During Configuration, including Table 6-3. Moved Delaying Configuration to be after Startup (Step 8). Added bulleted list of information security threats to Bitstream Security. Updated AES Overview. Added description of BBRAM to first paragraph of Loading the Encryption Key. Updated fourth paragraph of Loading Encrypted Bitstreams. Updated description of KEY_CLEAR and added URL to NIST key management site in AES_VIRTEX6. Updated Overview, page 105. Updated Creating an Encrypted Bitstream with Virtex-6 FPGA compression and encryption BitGen options. Added new first paragraph to eFUSE. Updated Contents column of FUSE_ID register in Table 6-20. Updated JTAG Instructions with description of programming eFUSEs. Updated description of connecting VFS pin to GND in VFS Pin. Updated description of GLUTMASK_B in Table 6-30. Chapter 8: Updated first paragraph of Fallback Overview, page 147. Chapter 9: Updated Table 9-1.
11/18/11	3.4	Chapter 2: Updated last sentence in Slave Serial Configuration. Chapter 6: Added defense-grade Virtex-6 XQ family to Table 6-5 and Table 6-13. Updated description of IWIDTH in Table 6-12. Updated Bitstream Security. Updated description of PERSIST in Table 6-30. Updated description of RS_TS_B in Table 6-40.

Date	Version	Revision
09/11/12	3.5	Chapter 2: Corrected signal directions of DIN and FCS_B in Figure 2-12. Chapter 4: Updated third paragraph of DNA_PORT. Updated Allowed Values column of Table 4-6. Chapter 6: Updated description of GWE and added note 3 to Table 6-17. Updated Device Identifier (Device DNA), including Figure 6-13. Removed "unique" from Figure 6-14. Chapter 11: Updated last paragraph of Introduction to say that preconfiguration weak pull-up resistors are disabled. In Table 11-4, added steps 5 to 10 and changed TMS column for step 18 from `1` to `0`.
04/18/13	3.6	Chapter 1: Removed link to iMPACT help in The Basic Configuration Solution. Chapter 2: Added link to iMPACT help in Master SPI Configuration Interface and Master BPI Configuration Interface. Updated second paragraph of Master BPI Configuration Interface. Added cautionary note before Table 2-9. Updated Micron J3 density and added new row for Spansion S29GLxxxS in Table 2-9. Chapter 6: Updated second row of Table 6-26 and removed table note. Chapter 7: Updated step 9 in Configuration Memory Read Procedure (SelectMAP). Updated configuration data for steps 2 and 3 in Table 7-2.
11/27/13	3.7	Chapter 6: Added Caution note to eFUSE Control Register (FUSE_CNTL). Updated description of FARSRC in Table 6-30. Chapter 9: Added POST_CRC_SOURCE to Post_CRC Constraints. Added POST_CRC_SOURCE to Table 9-3.
08/28/14	3.8	Chapter 6: Added Caution notes to bits 0 and 1 in Table 6-21. Updated description of CRCC in Table 6-28.
11/18/15	3.9	Chapter 9: Replaced list of error scenarios in SEU Correction with Table 9-2.

Table of Contents

Chapter 10: Advanced Configuration Interfaces

Chapter 11: Advanced JTAG Configurations

About This Guide

This document describes Virtex®-6 FPGA configuration. Complete and up-to-date documentation of the Virtex-6 family of FPGAs is available on the Xilinx website at http://www.xilinx.com/support/documentation/virtex-6.htm.

Guide Contents

This manual contains the following chapters:

Additional Documentation

The following documents are also available for download at: http://www.xilinx.com/support/documentation/virtex-6.htm.

- Virtex-6 Family Overview

  The features and product selection of the Virtex-6 family are outlined in this overview.

- Virtex-6 FPGA Data Sheet: DC and Switching Characteristics

  This data sheet contains the DC and Switching Characteristic specifications for the Virtex-6 family.

- Virtex-6 FPGA Packaging and Pinout Specifications

  This specification includes the tables for device/package combinations and maximum I/Os, pin definitions, pinout tables, pinout diagrams, mechanical drawings, and thermal specifications.

- Virtex-6 FPGA SelectIO Resources User Guide

  This guide describes the SelectIO™ resources available in all Virtex-6 devices.

- Virtex-6 FPGA Clocking Resources User Guide

  This guide describes the clocking resources available in all Virtex-6 devices, including the MMCM and PLLs.

- Virtex-6 FPGA Block RAM Resources User Guide

  The functionality of the block RAM and FIFO are described in this user guide.

- Virtex-6 FPGA Configurable Logic Block User Guide

  This guide describes the capabilities of the configurable logic blocks (CLBs) available in all Virtex-6 devices.

- Virtex-6 FPGA GTH Transceivers User Guide

  This guide describes the GTH transceivers available in all Virtex-6 HXT FPGAs except the XC6VHX250T and the XC6VHX380T in the FF1154 package.

- Virtex-6 FPGA GTX Transceivers User Guide

  This guide describes the GTX transceivers available in all Virtex-6 FPGAs except the XC6VLX760.

- Virtex-6 FPGA DSP48E1 Slice User Guide

  This guide describes the architecture of the DSP48E1 slice in Virtex-6 FPGAs and provides configuration examples.

- Virtex-6 FPGA Embedded Tri-Mode Ethernet MAC User Guide

  This guide describes the dedicated Tri-Mode Ethernet Media Access Controller available in all Virtex-6 FPGAs except the XC6VLX760.

- Virtex-6 FPGA System Monitor User Guide

  The System Monitor functionality available in all Virtex-6 devices is outlined in this guide.

- Virtex-6 FPGA PCB Design Guide

  This guide provides information on PCB design for Virtex-6 devices, with a focus on strategies for making design decisions at the PCB and interface level.

Additional Support Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/support/documentation/index.htm.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

XILINX®

Chapter 1

Configuration Overview

Overview

Virtex®-6 FPGAs are configured by loading application-specific configuration data—a bitstream—into internal memory. Virtex-6 FPGAs can load themselves from an external nonvolatile memory device or they can be configured by an external smart source, such as a microprocessor, DSP processor, microcontroller, PC, or board tester. In any case, there are two general configuration datapaths. The first is the serial datapath that is used to minimize the device pin requirements. The second datapath is the 8-bit, 16-bit, or 32-bit wide datapath that is used for higher performance or access (or link) to industry-standard interfaces, ideal for external data sources like processors, or x8- or x16-parallel flash memory.

Like processors and processor peripherals, Xilinx® FPGAs can be reprogrammed, in system, on demand, an unlimited number of times.

Because Xilinx FPGA configuration data is stored in CMOS configuration latches (CCLs), it must be reconfigured after it is powered down. The bitstream is loaded each time into the device through special configuration pins. These configuration pins serve as the interface for a number of different configuration modes:

- Master-Serial configuration mode
- Slave-Serial configuration mode
- Master SelectMAP (parallel) configuration mode (x8 and x16)
- Slave SelectMAP (parallel) configuration mode (x8, x16, and x32)
- JTAG/boundary-scan configuration mode
- Master Serial Peripheral Interface (SPI) flash configuration mode
- Master Byte Peripheral Interface Up or Down (BPI-Up or BPI-Down) flash configuration mode (x8 and x16 only)

The configuration modes are explained in detail in Chapter 2, Configuration Interfaces.

The specific configuration mode is selected by setting the appropriate level on the dedicated mode input pins M[2:0]. The M2, M1, and M0 mode pins should be set at a constant DC voltage level, either through pull-up or pull-down resistors (4.7 kΩ), or tied directly to ground or VCC_CONFIG. The mode pins should not be toggled before or during configuration, but they can be toggled after configuration. During configuration, the mode pins need to be at a valid logic level (either 0 or 1) when they are sampled on the rising edge of the INIT. See Chapter 2, Configuration Interfaces, for the mode pin setting options.

The terms Master and Slave refer to the direction of the configuration clock (CCLK):

- In Master configuration modes, the Virtex-6 device drives CCLK from an internal oscillator. To select the desired frequency, BitGen **-g ConfigRate** option is used. The BitGen section of UG628, *Command Line Tools User Guide* provides more information. After configuration, the CCLK is turned OFF unless the persist option is selected or SEU detection is used. The CCLK pin is 3-stated with a weak pull-up.
- In Slave configuration modes, CCLK is an input.

The JTAG/boundary-scan configuration interface is always available, regardless of the mode pin settings.

Design Considerations

To make an efficient system, it is important to consider which FPGA configuration mode best matches the system's requirements. Each configuration mode dedicates certain FPGA pins and can temporarily use other pins during configuration only. These non-dedicated pins are then released for general use when configuration is completed. See Table 6-1. Similarly, the configuration mode can place voltage restrictions on some FPGA I/O banks. Several different configuration options are available, and while the options are flexible, there is often an optimal solution for each system. Several topics must be considered when choosing the best configuration option: overall setup, speed, cost, and complexity.

FPGA Configuration Data Source

Virtex-6 FPGAs are designed for maximum flexibility. The FPGA either automatically loads itself with configuration data from a PROM, or another external intelligent device like a processor or microcontroller can download the configuration data to the FPGA.

Master Modes

The self-loading FPGA configuration modes, generically called *Master* modes, are available with either a serial or byte-wide datapath, as shown in Figure 1-1. The Master modes leverage various types of nonvolatile memories to store the FPGA's configuration information. In Master mode, the FPGA's configuration bitstream typically resides in nonvolatile memory on the same board, generally external to the FPGA. The FPGA internally generates a configuration clock signal called CCLK, and the FPGA controls the configuration process.

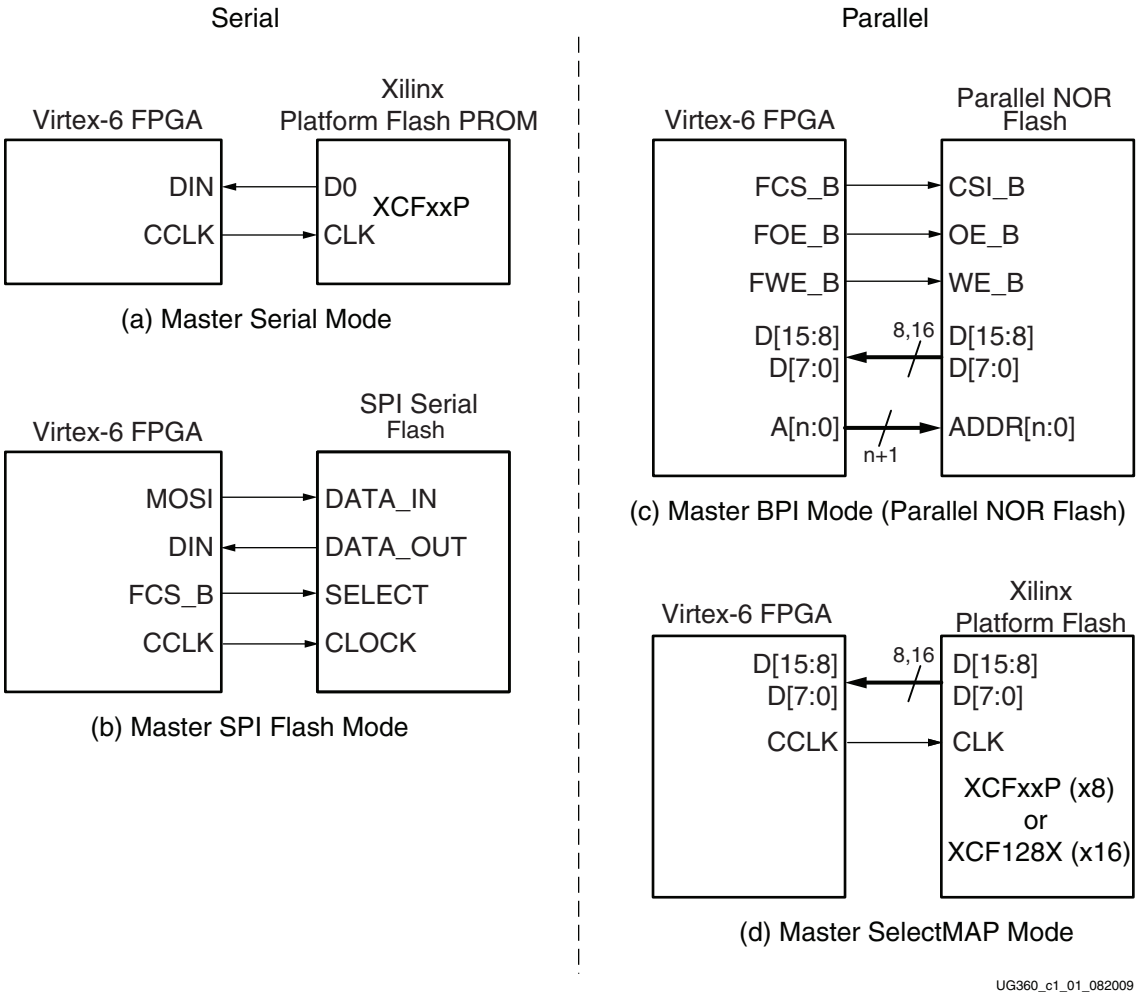


Figure 1-1: **Master Configuration Modes**

Slave Modes

The externally controlled loading FPGA configuration modes, generically called *Slave* modes, are also available with either a serial or byte-wide datapath. In Slave mode, an external "intelligent agent" such as a processor, microcontroller, DSP processor, or tester downloads the configuration image into the FPGA, as shown in Figure 1-2. The advantage of the Slave configuration modes is that the FPGA bitstream can reside almost anywhere in the overall system. The bitstream can reside in flash, onboard, along with the host processor's code. It can reside on a hard disk. It can originate somewhere over a network connection or another type of bridge connection.

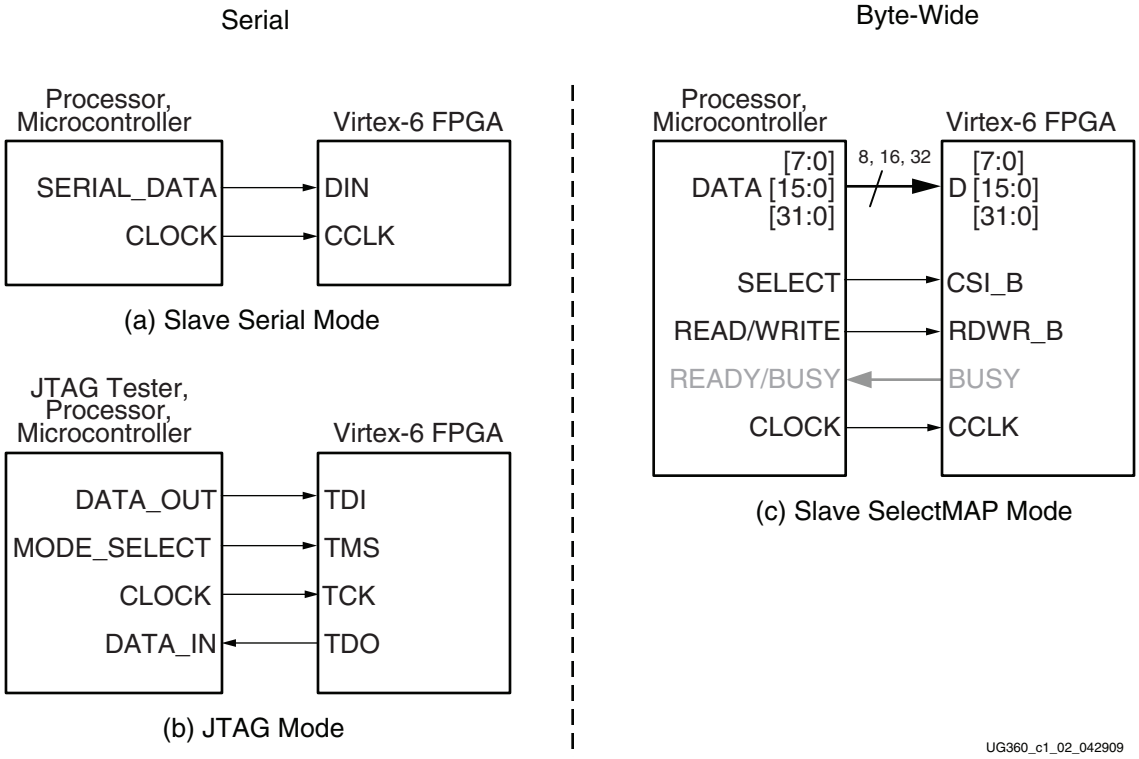


Figure 1-2: **Slave Configuration Modes**

The Slave Serial mode is extremely simple, consisting only of a clock and serial data input. The JTAG mode is also a simple serial configuration mode, popular for prototyping and highly utilized for board test. The Slave SelectMAP mode is a simple x8-, x16-, or x32-bit-wide processor peripheral interface, including a chip-select input and a read/write control input.

JTAG Connection

The four-wire JTAG interface is common on board testers and debugging hardware. In fact, the Xilinx programming cables for Virtex-6 FPGAs, listed here, use the JTAG interface for prototype download and debugging. Regardless of the configuration mode ultimately used in the application, it is best to also include a JTAG configuration path for easy design development. Also see Chapter 3, Boundary-Scan and JTAG Configuration.

- **Platform Cable USB II**
  http://www.xilinx.com/products/devkits/HW-USB-II-G.htm
- **Parallel Cable IV**
  http://www.xilinx.com/products/devkits/HW-PC4.htm

The Basic Configuration Solution

Basic options include either Master Serial mode using a Xilinx Platform Flash PROM or Master SPI mode using a third-party SPI PROM. These solutions use the fewest FPGA pins, have flexible I/O voltage support, and are fully supported by iMPACT, the Xilinx JTAG-based programming software. For more information, see the ISE iMPACT Help Software Manual.

The Low-Cost Priority Solution

The option with the lowest cost varies depending on the specific application.

- If there is spare nonvolatile memory already available in the system, the bitstream image can be stored in system memory. It can even be stored on a hard drive or downloaded remotely over a network connection. If so, one of the downloaded modes should be considered: Master BPI Mode and Slave Serial Mode, or JTAG Configuration Mode and Boundary-Scan.
- If nonvolatile memory is already required for an application, it is possible to consolidate the memory. For example, the FPGA configuration bitstream(s) can be stored with any processor code for the board. If the processor is a MicroBlaze™ embedded processor in the FPGA, the FPGA configuration data and the MicroBlaze processor code can share the same nonvolatile memory device.
- Virtex-6 FPGAs optionally configure directly from commodity SPI serial flash and parallel NOR flash memories. See Chapter 2, Configuration Interfaces. Also see XAPP973, *Indirect Programming of BPI PROMs with Virtex-5 FPGAs*, and XAPP974, *Indirect Programming of SPI Serial Flash PROMs with Spartan-3A FPGAs.*

The High-Speed Priority Option

Some applications require that the logic be operational within a short time. Certain FPGA configuration modes and methods are faster than others. The configuration time includes the initialization time plus the configuration time. Configuration time depends on the size of the device and speed of the configuration logic. For example, configuring at 30 MHz with a 16-bit data bus, a Virtex-6 LX130T device requires ~85 ms to receive its 43 Mb of configuration data. The Platform Flash XL is optimized for high-performance Virtex-6 FPGA configuration and has specific enhancements to ensure proper device synchronization. Additional information about the Platform Flash XL solution can be found in Chapter 2, Configuration Interfaces.

- At the same clock frequency, parallel configuration modes are inherently faster than the serial modes because they program 8, 16, or 32 bits at a time.
- Configuring a single FPGA is inherently faster than configuring multiple FPGAs in a daisy-chain. In a multi-FPGA design where configuration speed is a concern, each FPGA should be configured separately and in parallel.
- In Master modes, the FPGA internally generates the CCLK configuration clock signal. By default, the CCLK frequency starts out low but can be increased using the ConfigRate bitstream option. The maximum supported CCLK frequency setting depends on the read specifications for the attached nonvolatile memory. A faster memory enables faster configuration. The FPGA's CCLK output frequency varies with process, voltage, and temperature. The fastest guaranteed configuration rate depends on the slowest guaranteed CCLK frequency, as shown in the Virtex-6 FPGA data sheet. If an external clock is available on the board, it is also possible to configure the FPGA in a Slave mode while using Xilinx Platform Flash.
- Slave mode allows tighter tolerances and faster clocks. Combined with the Platform Flash XL, the solution can provide faster configuration times than a standard parallel NOR flash.

Conforming to PCI Express Link Activation Requirements

UG517, *LogiCORE IP Virtex-6 FPGA Integrated Block for PCI Express User Guide*, discusses a number of power and reset requirements. These requirements, when considered in an FPGA implementation, create several challenges that must be addressed for long-term reliability and broad interoperability. It is important to consider the link activation time in the PCI Express® application and ensure that the FPGA can complete configuration during the specified time. Many third-party flash vendors do not meet these specific time constraints. Refer to the *LogiCORE IP Virtex-6 FPGA Integrated Block for PCI Express User Guide* for a recommended Virtex-6 FPGA configuration solution.

Single and Multiple Configuration Images

In most FPGA applications, the FPGA is loaded only when the system is powered on.

However, some applications reload the FPGA multiple times while the system is operating, with different FPGA bitstreams for different functions. For example, the FPGA can be loaded with one bitstream to implement a power-on self-test, followed by a second bitstream with the final application. In many test equipment applications, the FPGA is loaded with different bitstreams to execute hardware-assisted tests. In this way, one smaller FPGA can implement the equivalent functionality of a larger ASIC or gate array device.

See Chapter 8, Reconfiguration and MultiBoot, for more information.

MultiBoot/Safe Update

In advanced applications, multiple bitstream images can be stored. One of the images can be upgraded by the user application, and real-time system upgrades can occur. The system can also recover from any failure booting from the initial image.

Required I/O Voltages

The chosen FPGA configuration mode places some constraints on the FPGA application, specifically the I/O voltage allowed on the FPGA's configuration banks. Virtex-6 FPGAs do not natively support 3.3V I/Os.

The SPI or BPI modes leverage third-party flash memory components. Use of third-party flash memory requires that the I/O voltage on the bank or banks attached to the third-party flash memory components complies with the input voltage. Use of these devices also requires appropriate handling.

The Xilinx Platform Flash PROM, on the other hand, supports a range of output voltages via a separate supply on the Platform Flash PROM.

Nonvolatile Data Storage

Some FPGA applications store data in external nonvolatile memory. Virtex-6 FPGAs provide useful enhancements for these applications.

- Virtex-6 FPGAs can configure directly from external commodity serial (SPI) or parallel flash PROMs (BPI).
- The flash PROM address, data, and control pins are only borrowed by the FPGA during configuration. After configuration, the FPGA has full read/write control over these pins.

- The FPGA configuration bitstreams and the application's nonvolatile data can share the same PROM, reducing overall system cost.

FPGA I/O Pin Settings during Configuration

Some of the FPGA pins used during configuration have dedicated pull-up resistors during configuration. However, the majority of user-I/O pins have optional pull-up resistors that can be enabled during the configuration process. During configuration, a single control line determines whether the pull-up resistors are enabled or disabled. The pin name is HSWAPEN.

Floating signal levels are problematic in CMOS logic systems. Other logic components in the system can require a valid input level from the FPGA. The internal pull-up resistors generate a logic High level on each pin. Generally, a device driving signals into the FPGA can overcome the pull-up resistor. Similarly, an individual pin can be pulled down using an appropriately sized external pull-down resistor.

In hot-swap or hot-insertion applications, the pull-up resistors provide a potential current path to the I/O power rail. Turning off the pull-up resistors disables this potential path. However, then external pull-up or pull-down resistors can be required on each individual I/O pin.

FPGA Density Migration

The package footprint and pinouts for Virtex-6 FPGAs are designed to allow migration between different densities within a specific family.

Likewise, an FPGA application can store other nonvolatile data in the flash memory, requiring a larger storage device.

To support design migration between device densities, sufficient configuration memory must be allowed to cover the largest device in the targeted package. For example, if using the Virtex-6 LX130T device, enough configuration memory to accommodate 44 Mb is required. To allow for migration to the Virtex-6 LX240T device, which is available in the same FF784 package, 74 Mb of configuration memory is required.

In downloaded applications, enough space in the memory must be reserved for the largest anticipated, uncompressed FPGA bitstream.

In self-loaded applications, a flash memory device footprint and the associated FPGA configuration mode should be used to facilitate easy migration.

Parallel flash memory devices support the widest density range in a common package footprint. For example, the Numonyx StrataFlash (P30) family accommodates 64 Mb through 1 Gb storage in the 64-Easy BGA package footprint. Alternatively, the 128 Mb Xilinx Platform Flash XL solution is large enough to cover FPGA density migration to all but four of the Virtex-6 family members and is ideal for applications requiring fast configuration times, such as PCI Express applications.

SPI serial flash vendors also offer a wide migration range in a common footprint, which is often multi-vendor compatible. The Numonyx M25P SPI serial flash family, for example, can accommodate Virtex-6 FPGA bitstreams from 32 Mb through 128 Mb in an SOIC16 common package footprint. This overview is provided as an example; flash vendors should be consulted for specific details.

Production Lifetime

An application's production lifetime should be considered. Commodity memories generally have a shorter production lifetime than the proprietary Xilinx Platform Flash PROMs. For example, if building an industrial application that will be manufactured for five years or more, then Xilinx Platform Flash PROMs provides better long-term availability.

Products with shorter production lifetimes can benefit from the multi-vendor pricing and multi-sourcing of commodity memories.

Protecting the FPGA Bitstream against Unauthorized Duplication

Like processor code, the bitstream that defines the FPGA's functionality loads into the FPGA during power-on. Consequently, this means that an unscrupulous company can capture the bitstream and create an unauthorized copy of the design.

Like processors, there are multiple techniques to protect the FPGA bitstream and any intellectual property (IP) cores embedded in the FPGA. The most powerful techniques are AES with the battery-backed SRAM key or AES with the eFUSE key. Device identification is a third technique, which uses a lower level of security and a device DNA. Device identification is described in detail in Chapter 6, Configuration Details. In addition, Virtex-6 devices also have on-chip Advanced Encryption Standard (AES) decryption logic to provide a high degree of design security. See Bitstream Encryption in Chapter 6, Configuration Details.

Loading Multiple FPGAs with the Same Configuration Bitstream

Generally, there is one configuration bitstream image per FPGA in a system. Multiple, different FPGA bitstream images can share a single configuration PROM by leveraging a configuration daisy-chain. However, if all the FPGAs in the application have the same part number and use the same bitstream, only a single bitstream image is required. An alternative solution, called a ganged or broad-side configuration, loads multiple, similar FPGAs with the same bitstream. See Chapter 10, Advanced Configuration Interfaces.

Configuration Factors

Many factors determine which configuration solution is optimal for a system. There are also a great deal of details that need to be accounted for. Configuration should be taken very seriously as to not cause problems later in the design cycle.

Designers need to understand the difference between dedicated configuration pins and reusable post configuration pins. Details can be found in the configuration details section.

Other issues that need to be considered are Data File formats and bitstream sizes. The size of the bitstream is directly affected by the device size and there are several formats in which the bitstream can be created.

The FPGA goes through certain sequences during the configuration process, from clearing internal memory to activating the I/Os. This process is called the configuration sequence. Designers should be aware of this sequence and its subsequences to understand the timing from power on to completed FPGA is configuration and start up.

Virtex-6 FPGAs also have enhanced security features such as AES encryption. This feature is very useful in protecting bitstream theft.

More details can be found in Chapter 6, Configuration Details.

Chapter 2

Configuration Interfaces

Virtex®-6 devices have five configuration interfaces. Each configuration interface corresponds to one or more configuration modes and bus width, shown in Table 2-1. For detailed interface timing information, see DS152, *Virtex-6 FPGA Data Sheet: DC and Switching Characteristics*.

Table 2-1: **Virtex-6 FPGA Configuration Modes**

Configuration Mode	M[2:0]	Bus Width(1)	CCLK Direction
Master Serial(2)	000	1	Output
Master SPI(2)	001	1	Output
Master BPI-Up(2)	010	8, 16	Output
Master BPI-Down(2)	011	8, 16	Output
Master SelectMAP(2)	100	8, 16	Output
JTAG	101	1	Input (TCK)
Slave SelectMAP	110	8, 16, 32	Input
Slave Serial(3)	111	1	Input

Notes:

1. The parallel configuration modes bus is auto-detected by the configuration logic. AES encrypted bitstreams are supported in modes using a bus width of 1 or 8.
2. In Master configuration mode, the CCLK pin is the clock source for the Virtex-6 FPGA internal configuration logic. The Virtex-6 FPGA CCLK output pin must be free from reflections to avoid double-clocking the internal configuration logic. Refer to the Board Layout for Configuration Clock (CCLK) section for more details.
3. This is the default setting due to internal pull-up termination on mode pins.

Serial Configuration Interface

In serial configuration modes, the FPGA is configured by loading one configuration bit per CCLK cycle. CCLK is an output in Master Serial mode and an input in Slave Serial mode.

Figure 2-1 shows the basic Virtex-6 FPGA serial configuration interface.

There are four methods of configuring an FPGA in serial mode:

- Single-Device Master Serial

  Typical setup includes a Xilinx® PROM, such as the Platform Flash XCFxxP.
- Single-Device Slave Serial

  Typical setup includes a processor providing data and clock.

- Multiple-Device Daisy-Chain Serial bus

  Multiple FPGAs are configured in series with different images from a PROM or processor.

- Multiple-Device Ganged Serial

  Multiple FPGAs are configured in parallel with the same image from a PROM or processor.

Master and Slave Serial basic configuration are described in this chapter. Advanced daisy-chain and ganged configuration methods are discussed in Chapter 10, Advanced Configuration Interfaces.

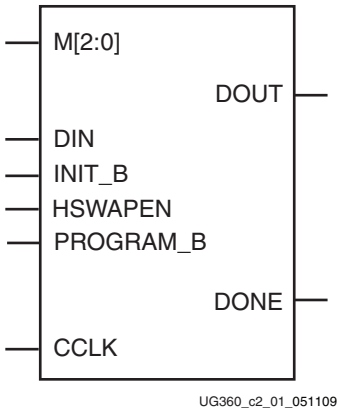


Figure 2-1: **Virtex-6 FPGA Serial Configuration Interface**

Table 2-2 describes the Serial Configuration Interface.

Table 2-2: **Virtex-6 FPGA Serial Configuration Interface Pins**

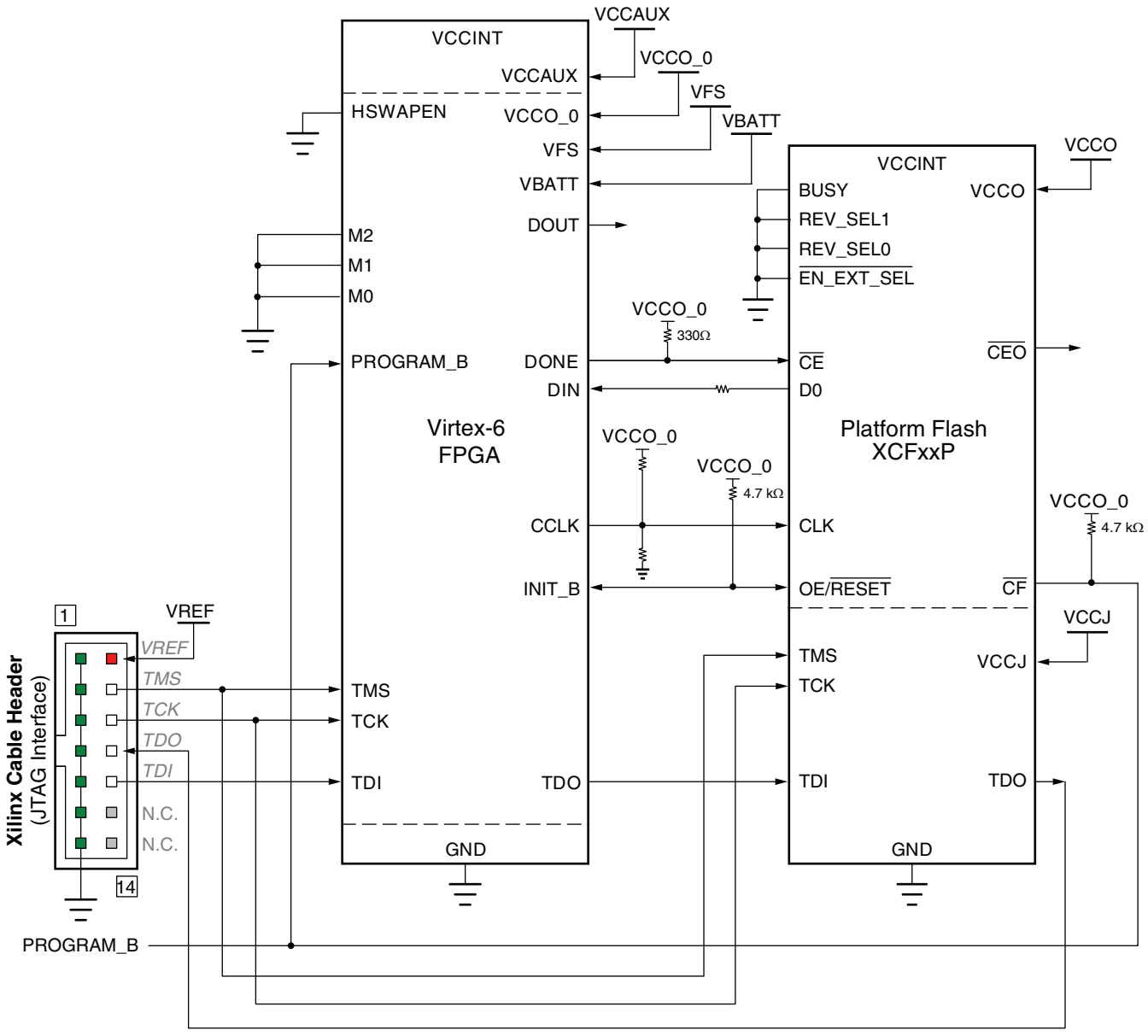
Pin Name	Type	Dedicated or Dual-Purpose	Description
M[2:0]	Input	Dedicated	Mode Pins – determine configuration mode. See Table 2-1, page 25.
CCLK	Bidirectional, Input, or Output	Dedicated	Configuration clock source for all configuration modes except JTAG. See Board Layout for Configuration Clock (CCLK), page 58 for board design considerations. CCLK is an output in Master Configuration modes and an input in Slave Configuration modes.
DIN	Input	Dedicated	Serial configuration data input, synchronous to rising CCLK edge.
DOUT	Output	Dedicated	Serial data output for downstream daisy-chained devices. Data provided on the falling edge of CCLK.
DONE	Bidirectional, Open-Drain, or Active	Dedicated	Active-High signal indicating configuration is complete: 0 = FPGA not configured 1 = FPGA configured

Table 2-2: **Virtex-6 FPGA Serial Configuration Interface Pins *(Cont'd)***

Pin Name	Type	Dedicated or Dual-Purpose	Description
INIT_B	Bidirectional, Input, or Output, Open-Drain	Dedicated	From power-on reset or PROGRAM_B reset, INIT_B is driven Low, indicating that the FPGA is initializing (clearing) its configuration memory. Before the Mode pins are sampled, INIT_B is an open-drain, active-Low input and can be held Low to delay configuration. After the Mode pins are sampled, the INIT_B output indicates if a CRC error occurred during configuration or a readback CRC error occurred after configuration (when enabled): 0 = CRC or IDCODE error (DONE is Low) or Readback CRC Error (DONE is High and Readback CRC is enabled). 1 = No CRC error, housecleaning is complete (needs an external pull-up resistor). When the SEU detection function is enabled, INIT_B is optionally driven Low when a readback CRC error is detected.
PROGRAM_B	Input	Dedicated	Active-Low full-chip reset that is edge sensitive before and during power-up. After power-up, this signal is level sensitive. PROGRAM_B cannot be used to delay configuration from power-up. See Delaying Configuration, page 100 for additional information.
HSWAPEN	Input	Dedicated	Controls I/O (except bank0 dedicated I/Os) pull-up during configuration. A weak pull-up resistor is built into this pin. 0 = Pull-up during configuration 1 = 3-stated during configuration

Master Serial Configuration

The Master Serial mode is designed so that the FPGA can be configured from a Xilinx configuration PROM, as shown in Figure 2-2.



Refer to the Notes following this figure for related information.

UG360_c2_02_091311

Figure 2-2: **Master Serial Mode Configuration Example**

Notes relevant to Figure 2-2:

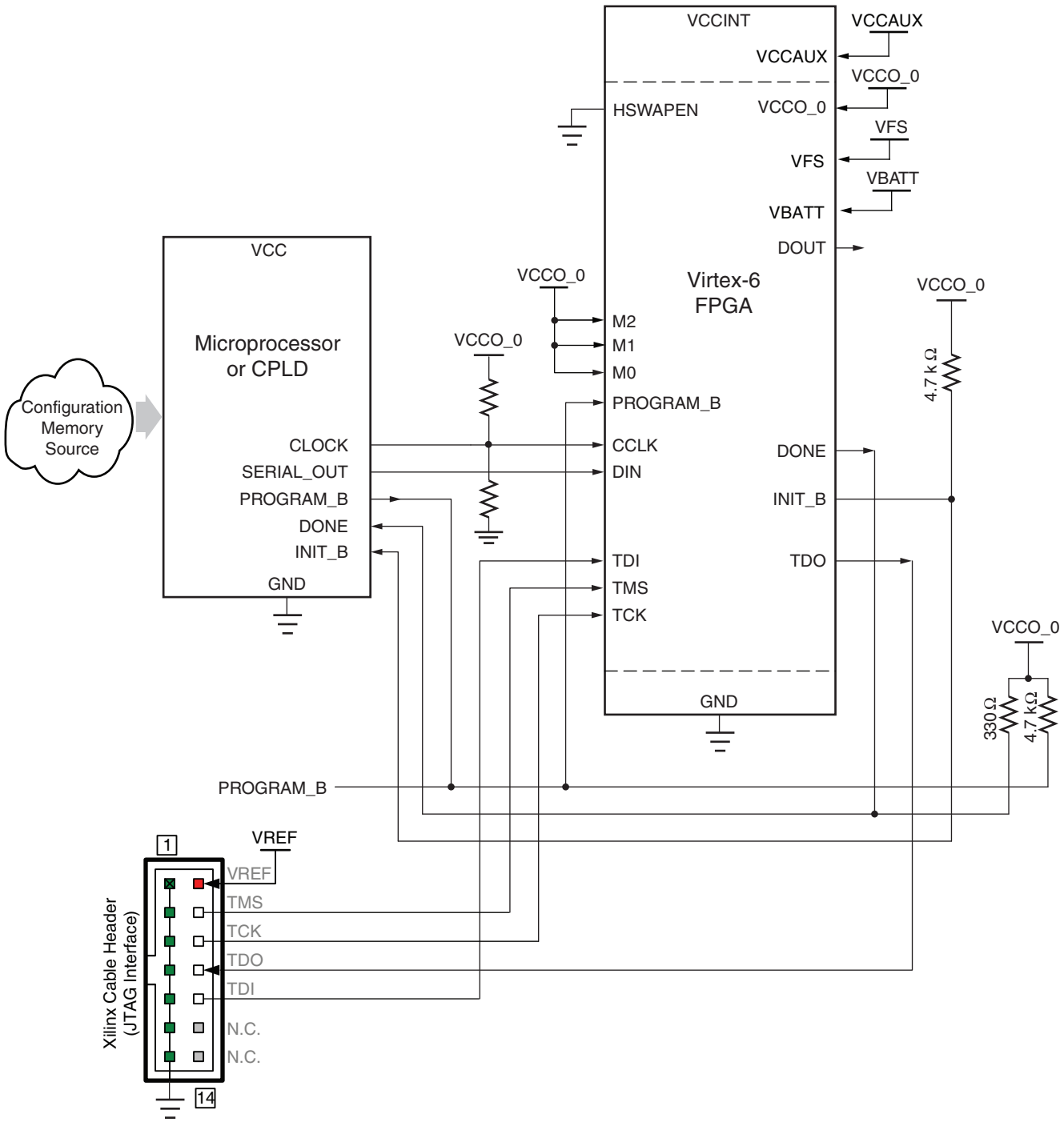
1. The Virtex-6 FPGA VCCO_0 supply input, the Platform Flash PROM V_{CCO} and VCCJ supply inputs, and the Xilinx Cable V_{REF} must be the same voltage.
2. The DONE pin is by default an open-drain output requiring an external pull-up resistor. However, the DONE pin has a programmable active driver. In this

arrangement, the active DONE driver can be enabled in BitGen (**DriveDone** option) to eliminate the need for an external pull-up resistor.

3. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
4. The BitGen startup clock setting must be set for CCLK for serial configuration, which is done by default in the software. See UG628, *Command Line Tools User Guide* for details.
5. The PROM in this diagram represents one or more Xilinx PROMs. Multiple Xilinx PROMs can be cascaded to increase the overall configuration storage capacity, further described in UG161, *Platform Flash PROM User Guide*.
6. The `BIT` file must be reformatted into a PROM file before it can be stored on the Xilinx PROM. Refer to Generating PROM Files, page 89, which outlines how to use iMPACT and PROMGen software to generate the required files.
7. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines. The example parallel Thevenin termination shown in Figure 2-2 is only one possible termination topology. The advantages of this option are simplicity of design and application; however, this topology can be less desirable for some applications because it can dissipate more power. This trade-off must be weighed against other trade-offs to determine the optimal termination topography for an interface.
8. Signal analysis (e.g., IBIS simulation) is recommended to ensure proper signal quality. Platform Flash PROM devices with low-impedance output drivers might require series termination (the series resistor is close to the Platform Flash PROM output) to manage the signal characteristics.
9. VFS is used for eFUSE programming. See eFUSE, page 107 for more details.
10. V_{BATT} is the power source for AES key storage. If AES encryption is not used, V_{BATT} can be tied to ground, V_{CCAUX}, or left unconnected. See Bitstream Security, page 101 for more details.

Slave Serial Configuration

Slave Serial configuration is typically used for devices in a serial daisy chain or when configuring a single device from an external microprocessor or CPLD (see Figure 2-3). Design considerations are similar to Master Serial configuration except for the direction of CCLK. CCLK must be driven from an external clock source, which also provides data (see Clocking Serial Configuration Data, page 31). Partial reconfiguration applications using Slave Serial configuration need to set the PERSIST bit (bit 3) in the CTL0 register to 1. This can be done using the BitGen **-g Persist:CTLReg** option to keep the serial port enabled after the first configuration.



Refer to the Notes following this figure for related information.

UG380_c2_03_091611

Figure 2-3: **Slave Serial Mode Configuration Example**

Notes relevant to Figure 2-3:

1. The Virtex-6 FPGA VCCO_0 and the Xilinx Cable V_{REF} must have the same voltage.
2. The DONE pin is by default an open-drain output requiring an external pull-up resistor. The DONE pin has a programmable active driver. To enable it, select the **`DriveDone`** option in BitGen.

3. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
4. The BitGen startup clock setting must be set for CCLK for serial configuration.
5. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines. The example parallel Thevenin termination shown in Figure 2-3 is only one possible termination topology. The advantages of this option are simplicity of design and application; however, this topology can be less desirable for some applications because it can dissipate more power. This trade-off must be weighed against other trade-offs to determine the optimal termination topography for an interface.
6. VFS is used for eFUSE programming. See eFUSE, page 107 for more details.
7. V_{BATT} is the power source for AES key storage. If AES encryption is not used, V_{BATT} can be tied to ground, V_{CCAUX}, or left unconnected. See Bitstream Security, page 101 for more details.

Clocking Serial Configuration Data

Figure 2-4 shows how configuration data is clocked into Virtex-6 devices in Slave Serial and Master Serial modes.

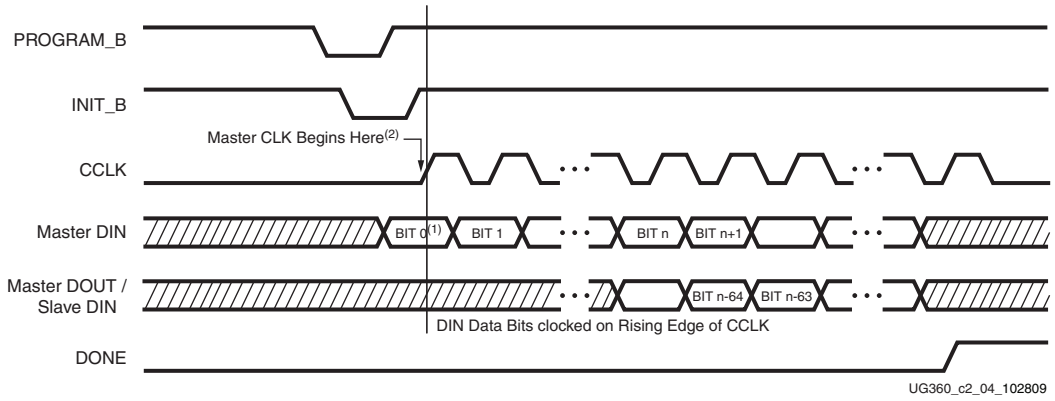


Figure 2-4: **Serial Configuration Clocking Sequence**

Notes relevant to Figure 2-4:

1. Bit 0 represents the MSB of the first byte. For example, if the first byte is `0xAA` (`1010_1010`), bit 0 = `1`, bit 1 = `0`, bit 2 = `1`, etc.
2. For Master configuration mode, CCLK is driven only after INIT_B goes High to shortly after DONE goes High. Otherwise CCLK is in a high-impedance state.
3. CCLK can be free-running in Slave Serial mode.

SelectMAP Configuration Interface

The SelectMAP configuration interface (Figure 2-5) provides an 8-bit, 16-bit, or 32-bit bidirectional data bus interface to the Virtex-6 FPGA configuration logic that can be used for both configuration and readback. For details, refer to Chapter 7, Readback and Configuration Verification. The bus width of SelectMAP is automatically detected (see Bus Width Auto Detection, page 88).

CCLK is an output in Master SelectMAP mode and an input in Slave SelectMAP mode. One or more Virtex-6 devices can be configured through the SelectMAP bus.

There are multiple methods of configuring an FPGA in SelectMAP mode:

- Single-device Master SelectMAP

  Typical setup includes a Xilinx PROM such as the Platform Flash (XCFxxP).

- Single-device Slave SelectMAP

  Typical setup includes a Xilinx PROM such as a Platform Flash XL (XCF128X) or a processor providing data and clock.

- Multiple-device daisy-chain SelectMAP bus

  Multiple FPGAs are configured in series with different images from a PROM or processor.

- Multiple-device ganged SelectMAP

  Multiple FPGAs are configured in parallel with the same image from a PROM or processor.

The basic Master SelectMAP and Slave SelectMAP configuration methods are described in this chapter. Advanced daisy-chain and ganged configuration methods are described in Chapter 10, Advanced Configuration Interfaces.

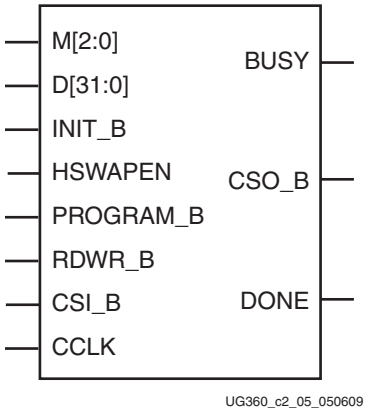


Figure 2-5: **Virtex-6 FPGA SelectMAP Configuration Interface**

Table 2-3 describes the SelectMAP configuration interface.

Table 2-3: **Virtex-6 FPGA SelectMAP Configuration Interface Pins**

Pin Name	Type	Dedicated or Dual-Purpose	Description
M[2:0]	Input	Dedicated	Mode pins - determine configuration mode (see Table 2-1, page 25.)
CCLK	Bidirectional, Input, and Output	Dedicated	Configuration clock source for all configuration modes except JTAG (see section Board Layout for Configuration Clock (CCLK), page 58). CCLK is an output in Master Configuration modes and an input in Slave Configuration modes.

Table 2-3: **Virtex-6 FPGA SelectMAP Configuration Interface Pins *(Cont'd)***

Pin Name	Type	Dedicated or Dual-Purpose	Description
D[31:0]	3-State Bidirectional	Dual-Purpose	Configuration and readback data bus, clocked on the rising edge of CCLK. See Parallel Bus Bit Order and Table 2-5, page 43. The data bus width can be x8 or x16 for Master SelectMAP configurations and x8, x16, or x32 for Slave SelectMAP configurations. Only x8 data bus widths are supported for configuring with AES encrypted bitstreams in SelectMAP mode.
BUSY	3-State Output	Dedicated	Indicates that the device is not ready to send readback data. For Virtex-6 devices, the BUSY signal is only needed for readback; it is not needed for configuration (see SelectMAP Data Loading, page 38).
DONE	Bidirectional, Open-Drain or Active	Dedicated	Active-High signal indicating configuration is complete: 0 = FPGA not configured 1 = FPGA configured
INIT_B	Bidirectional, Input, or Output, Open-Drain	Dedicated	From power-on reset or PROGRAM_B reset, INIT_B is driven Low, indicating that the FPGA is initializing (clearing) its configuration memory. Before the Mode pins are sampled, INIT_B is an open-drain, active-Low input and can be held Low to delay configuration. After the Mode pins are sampled, the INIT_B output indicates if a CRC error occurred during configuration or a readback CRC error occurred after configuration (when enabled): 0 = CRC or IDCODE error (DONE is Low) or Readback CRC Error (DONE is High and Readback CRC is enabled). 1 = No CRC error, housecleaning is complete (needs an external pull-up resistor). When the SEU detection function is enabled, INIT_B is optionally driven Low when a readback CRC error is detected.
PROGRAM_B	Input	Dedicated	Active-Low full-chip reset that is edge sensitive before and during power-up. After power-up, this signal is level sensitive. PROGRAM_B cannot be used to delay configuration from power-up. See Delaying Configuration, page 100 for additional information.
CSI_B	Input	Dedicated	Active-Low chip select to enable the SelectMAP data bus (see SelectMAP Data Loading): 0 = SelectMAP data bus enabled 1 = SelectMAP data bus disabled

Table 2-3: **Virtex-6 FPGA SelectMAP Configuration Interface Pins *(Cont'd)***

Pin Name	Type	Dedicated or Dual-Purpose	Description
RDWR_B	Input	Dedicated	Determines the direction of the D[x:0] data bus (see SelectMAP Data Loading): 0 = Inputs 1 = Outputs RDWR_B input can only be changed while CSI_B is deasserted, otherwise an ABORT occurs (see SelectMAP ABORT, page 169).
CSO_B	3-State Output	Dual-Purpose	Parallel daisy chain active-Low chip select output during configuration; otherwise this output is 3-stated. Not used in single FPGA applications.
HSWAPEN	Input	Dedicated	Controls I/O (except bank0 dedicated I/Os) pull-up during configuration. A weak pull-up resistor is built into this pin. 0 = Pull-up during configuration 1 = 3-stated during configuration

Single Device SelectMAP Configuration

High-Performance Platform Flash XL SelectMAP Configuration

The Platform Flash XL is specially optimized for high-performance Virtex-6 FPGA configuration and ease-of-use. Platform Flash XL integrates 128 Mb of in-system programmable flash storage and performance features for configurations. Power-on burst read mode and dedicated I/O power supply enable Platform Flash XL to mate seamlessly with the Virtex-6 FPGA SelectMAP configuration interface. A wide, 16-bit data bus delivers the FPGA configuration bitstream without wait states. For Platform Flash XL details, see DS617, *Platform Flash XL High-Density Configuration and Storage Device*.

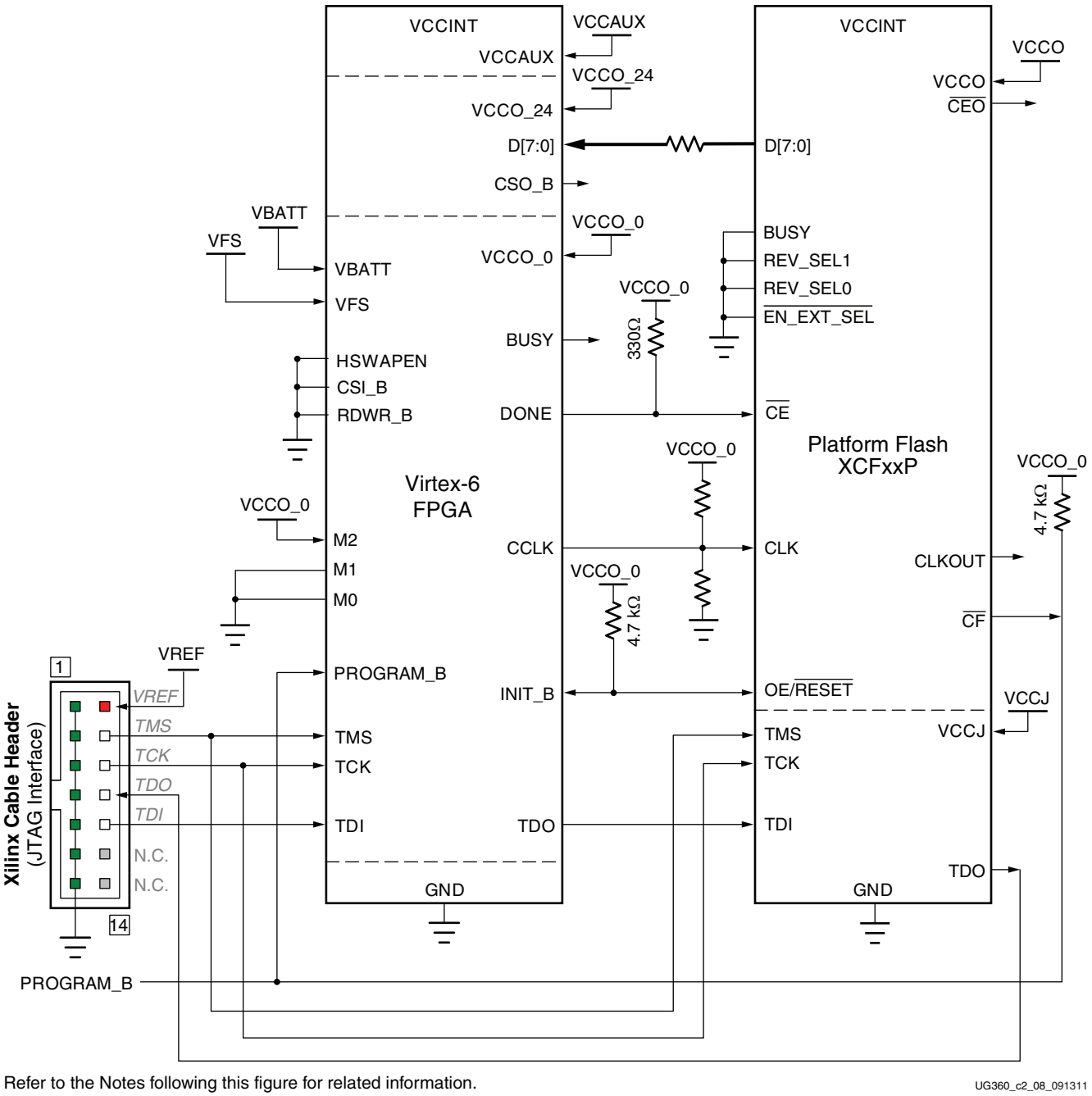
After configuration, the Virtex-6 FPGA can access any remaining memory space, beyond the bitstream, in the Platform Flash XL. The Platform Flash XL has a standard BPI NOR flash interface. In addition to the 16-bit data bus, which is dually used for SelectMAP configuration, the Platform Flash XL has a standard address bus and read/write control pins for random access reads and for sending CFI-compliant commands.

For prototype designs, the ISE® iMPACT software provides an indirect Platform Flash XL programming solution through the IEEE Std 1149.1 (JTAG) port of the Virtex-6 FPGA.

For details regarding the Virtex-6 FPGA SelectMAP configuration setup or the iMPACT indirect programming solution for the Platform Flash XL, see UG438, *Platform Flash XL User Guide*.

Platform Flash PROM SelectMAP Configuration

The simplest way to configure a single device in SelectMAP mode is to connect it directly to a configuration PROM, as shown in Figure 2-6. In this arrangement, the device is set for Master SelectMAP mode, and the RDWR_B and CSI_B pins are tied to ground for continuous data loading (see SelectMAP Data Loading, page 38).



Refer to the Notes following this figure for related information.

UG360_c2_08_091311

Figure 2-6: **Single Device Master SelectMAP (x8) Configuration Example**

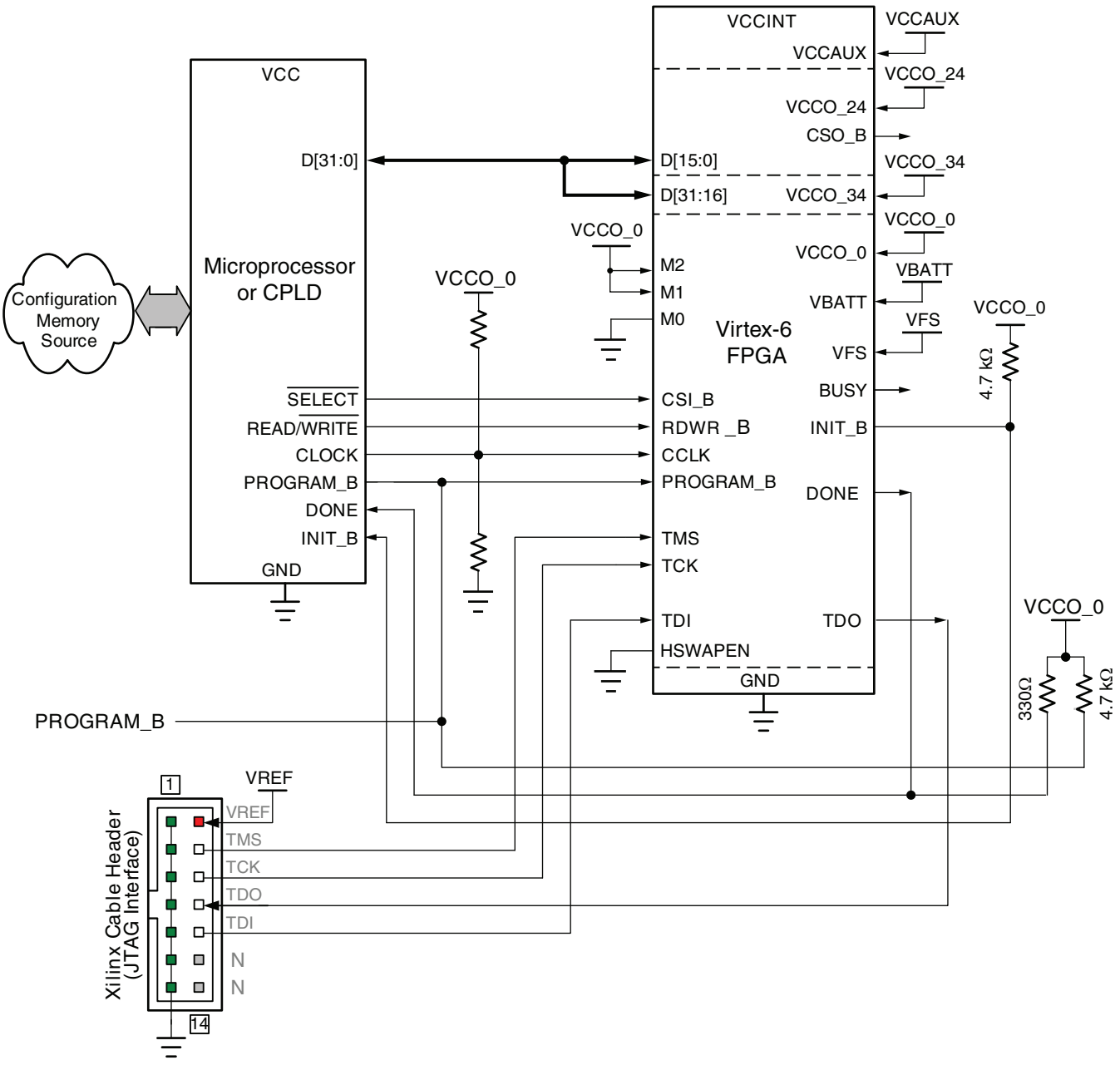
Notes relevant to Figure 2-6:

1. Virtex-6 FPGA supply input VCCO_0, the Platform Flash PROM V_{CCO} supply input and VCCJ, and the Xilinx Cable V_{REF} must have the same voltage. The Virtex-6 FPGA bank voltage VCCO_24 supplies the D[7:0] and CSO_B signals.
2. The DONE pin is by default an open-drain output requiring an external pull-up resistor. However, the DONE pin has a programmable active driver. In this arrangement, the active DONE driver can be enabled in BitGen (**`DriveDone`** option) to eliminate the need for an external pull-up resistor.

3. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
4. The BitGen startup clock setting must be set for CCLK for SelectMAP configuration.
5. The PROM in this diagram represents one or more Xilinx PROMs. Multiple PROMs can be cascaded to increase the overall configuration storage capacity.
6. The `BIT` file must be reformatted into a PROM file before it can be stored on the PROM. Refer to the Generating PROM Files, page 89.
7. The Xilinx Platform Flash PROM must be programmed for parallel mode when utilized in SelectMAP Configuration mode.
8. When configuring a Virtex-6 device in SelectMAP mode from a Xilinx configuration PROM, the RDWR_B and CSI_B signals can be tied Low (see SelectMAP Data Loading, page 38).
9. The BUSY signal does not need to be monitored for this setup and can be left unconnected (see SelectMAP Data Loading, page 38).
10. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines. The example parallel Thevenin termination shown in Figure 2-6 is only one possible termination topology. The advantages of this option are simplicity of design and application; however, this topology can be less desirable for some applications because it can dissipate more power. This trade-off must be weighed against other trade-offs to determine the optimal termination topography for an interface.
11. Signal analysis (e.g., IBIS simulation) is recommended to ensure proper signal quality. Platform Flash PROM devices with low-impedance output drivers might require series termination (the series resistor is close to the Platform Flash PROM output) to manage the signal characteristics.
12. VFS is used for eFUSE programming. See eFUSE, page 107 for more details.
13. V_{BATT} is the power source for AES key storage. If AES encryption is not used, V_{BATT} can be tied to ground, V_{CCAUX}, or left unconnected. See Bitstream Security, page 101 for more details.

Microprocessor-Driven SelectMAP Configuration

For custom applications where a microprocessor or CPLD is used to configure a single Virtex-6 device, either Master SelectMAP mode (use CCLK from the FPGA) or Slave SelectMAP mode can be used (Figure 2-7). Slave SelectMAP mode is preferred. See XAPP502, *Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode*, for information on configuring Virtex devices using a microprocessor.



Refer to the Notes following this figure for related information.

UG360_c2_09_091311

Figure 2-7: **Single Slave Device SelectMAP Configuration from Microprocessor or CPLD Example**

Notes relevant to Figure 2-7:

1. Refer to XAPP502, *Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode*, for a discussion of one possible implementation.
2. The processor or CPLD I/O needs to support a voltage that is compatible with the connected FPGA pins. The Virtex-6 FPGA VCCO_0 supply input and the Xilinx Cable V_{REF} must have the same voltage. Virtex-6 FPGA bank voltage VCCO_24 supplies the CSO_B and D[15:0] signals, and bank voltage VCCO_34 supplies the D[31:16] signals.

3. The DONE pin is by default an open-drain output requiring an external pull-up resistor. In this arrangement, the active DONE driver can be enabled, eliminating the need for an external pull-up resistor.
4. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
5. The BitGen startup clock setting must be set for CCLK for SelectMAP configuration.
6. The BUSY signal can be left unconnected if readback is not needed.
7. The CSI_B and RDWR_B signals can be tied to ground if only one FPGA is going to be configured and readback is not needed.
8. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines. The example parallel Thevenin termination shown in Figure 2-7 is only one possible termination topology. The advantages of this option are simplicity of design and application; however, this topology can be less desirable for some applications because it can dissipate more power. This trade-off must be weighed against other trade-offs to determine the optimal termination topography for an interface.
9. The Data bus width can be x8, x16, or x32 for Slave SelectMAP configuration. However, only x8 data bus widths are supported for configuring with AES encrypted bitstreams in SelectMAP mode.
10. VFS is used for eFUSE programming. See eFUSE, page 107 for more details.
11. V_{BATT} is the power source for AES key storage. If AES encryption is not used, V_{BATT} can be tied to ground, V_{CCAUX}, or left unconnected. See Bitstream Security, page 101 for more details.

SelectMAP Data Loading

The SelectMAP interface allows for either continuous or non-continuous data loading. Data loading is controlled by the CSI_B, RDWR_B, CCLK, and BUSY signals.

CSI_B

The Chip Select input (CSI_B) enables the SelectMAP bus. When CSI_B is High, the Virtex-6 device ignores the SelectMAP interface, neither registering any inputs nor driving any outputs. D and BUSY are placed in a High-Z state, and RDWR_B is ignored.

- If CSI_B = `0`, the device's SelectMAP interface is enabled.
- If CSI_B = `1`, the device's SelectMAP interface is disabled.

For a multiple device SelectMAP configuration, refer to Chapter 10, Advanced Configuration Interfaces.

If only one device is being configure through the SelectMAP and readback is not required, the CSI_B signal can be tied to ground, as illustrated in Figure 2-6.

RDWR_B

RDWR_B is an input to the Virtex-6 device that controls whether the data pins are inputs or outputs:

- If RDWR_B = `0`, the data pins are inputs (writing to the FPGA).
- If RDWR_B = `1`, the data pins are outputs (reading from the FPGA).

For configuration, RDWR_B must be set for write control (RDWR_B = 0). For readback, RDWR_B must be set for read control (RDWR_B = 1) while CSI_B is asserted. (For details, refer to Chapter 7, Readback and Configuration Verification.)

Changing the value of RDWR_B from Low to High while CSI_B is Low triggers an ABORT, and the configuration I/O changes from input to output asynchronously. The ABORT status appears on the data pins synchronously. Changing the value of RDWR_B from High to Low while CSI_B is Low also triggers an ABORT, and the configuration I/O changes from output to input asynchronously with no ABORT status readback (see SelectMAP ABORT, page 169). If readback is not needed, RDWR_B can be tied to ground or used for debugging with SelectMAP ABORT.

The RDWR_B signal is ignored while CSI_B is deasserted. Read/write control of the 3-stating of the data pins is asynchronous. The FPGA actively drives SelectMAP data without regard to CCLK if RDWR_B is set for read control (RDWR_B = 1, Readback) while CSI_B is asserted.

CCLK

All activity on the SelectMAP data bus is synchronous to CCLK. When RDWR_B is set for write control (RDWR_B = 0, Configuration), the FPGA samples the SelectMAP data pins on rising CCLK edges. When RDWR_B is set for read control (RDWR_B = 1, Readback), the FPGA updates the SelectMAP data pins on rising CCLK edges.

In Slave SelectMAP mode, configuration can be paused by stopping CCLK (see Non-Continuous SelectMAP Data Loading, page 41).

BUSY

BUSY is an output from the FPGA indicating when the device is ready to drive readback data. Virtex-6 FPGAs never drive the BUSY signal during configuration, even at the maximum configuration frequency with an encrypted bitstream. The Virtex-6 device only drives BUSY during readback. (For details, refer to Chapter 7, Readback and Configuration Verification.)

- If BUSY = 0 during readback, the SelectMAP data pins are driving valid readback data.
- If BUSY = 1 during readback, the SelectMAP data pins are not driving valid readback data.

When CSI_B is deasserted (CSI_B = 1), the BUSY pin is placed in a High-Z state. BUSY remains in a High-Z state until CSI_B is asserted.

Unless readback is used, the BUSY pin can be left unconnected.

Continuous SelectMAP Data Loading

Continuous data loading is used in applications where the configuration controller can provide an uninterrupted stream of configuration data. After power-up, the configuration controller sets the RDWR_B signal for write control (RDWR_B = 0) and asserts the CSI_B signal (CSI_B = 0), causing the device to drive BUSY Low (this transition is asynchronous). RDWR_B must be driven Low before CSI_B is asserted, otherwise an ABORT occurs (see SelectMAP ABORT, page 169).

On the next rising CCLK edge, the device begins sampling the data pins. Only D[0:7] are sampled by Configuration until the bus width is determined. See Bus Width Auto Detection, page 88 for details. After bus width is determined, the proper width of the data

bus is sampled for the Synchronization word search. Configuration begins after the synchronization word is clocked into the device.

After the configuration bitstream is loaded, the device enters the startup sequence. The device asserts its DONE signal High in the phase of the startup sequence that is specified by the bitstream (see Startup (Step 8) in Chapter 6). The configuration controller should continue sending CCLK pulses until after the startup sequence has finished. (This can require several CCLK pulses after DONE goes High. See Startup (Step 8) in Chapter 6 for details).

After configuration, the CSI_B and RDWR_B signals can be deasserted, or they can remain asserted. Because the SelectMAP port is inactive, toggling RDWR_B at this time does not cause an abort. Figure 2-8 summarizes the timing of SelectMAP configuration with continuous data loading.

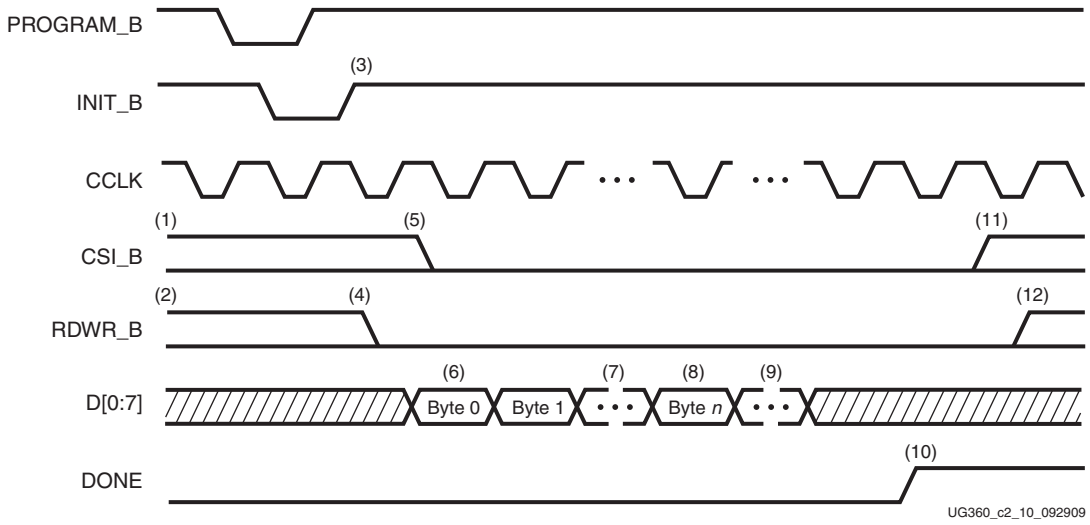


Figure 2-8: **Continuous x8 SelectMAP Data Loading**

Notes relevant to Figure 2-8:

1. CSI_B signal can be tied Low if there is only one device on the SelectMAP bus. If CSI_B is not tied Low, it can be asserted at any time.
2. RDWR_B can be tied Low if readback is not needed. RDWR_B should not be toggled after CSI_B has been asserted because this triggers an ABORT. (See SelectMAP ABORT, page 169).
3. The Mode pins are sampled when INIT_B goes High.
4. RDWR_B should be asserted before CSI_B to avoid causing an abort.
5. CSI_B is asserted, enabling the SelectMAP interface.
6. The first byte is loaded on the first rising CCLK edge after CSI_B is asserted.
7. The configuration bitstream is loaded one byte per rising CCLK edge.
8. After the startup command is loaded, the device enters the startup sequence.
9. The startup sequence lasts a minimum of eight CCLK cycles. (See Startup (Step 8) in Chapter 6.)
10. The DONE pin goes High during the startup sequence. Additional CCLKs can be required to complete the startup sequence. (See Startup (Step 8) in Chapter 6.)
11. After configuration has finished, the CSI_B signal can be deasserted.

12. After the CSI_B signal is deasserted, RDWR_B can be deasserted.
13. The data bus can be x8, x16, or x32 (for Slave SelectMAP).

Non-Continuous SelectMAP Data Loading

Non-continuous data loading is used in applications where the configuration controller cannot provide an uninterrupted stream of configuration data—for example, if the controller pauses configuration while it fetches additional data.

Configuration can be paused in two ways: by deasserting the CSI_B signal (Free-Running CCLK method, Figure 2-9) or by halting CCLK (Controlled CCLK method, Figure 2-10).

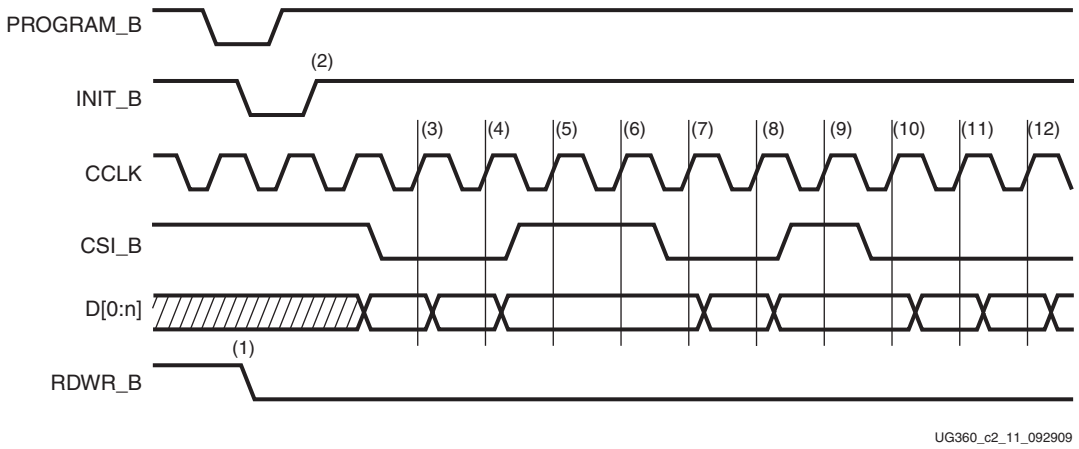


Figure 2-9: **Non-Continuous SelectMAP Data Loading with Free-Running CCLK**

Notes relevant to Figure 2-9:

1. RDWR_B is driven Low by the user, setting the D[0:n] pins as inputs for configuration. RDWR_B can be tied Low if readback is not needed. RDWR_B should not be toggled after CSI_B has been asserted because this triggers an ABORT. (See SelectMAP ABORT, page 169).
2. The device is ready for configuration after INIT_B goes High.
3. A byte is loaded on the rising CCLK edge. The data bus can be x8, x16, or x32 wide (for Slave SelectMAP).
4. A byte is loaded on the rising CCLK edge.
5. The user deasserts CSI_B, and the byte is ignored.
6. The user deasserts CSI_B, and the byte is ignored.
7. A byte is loaded on the rising CCLK edge.
8. A byte is loaded on the rising CCLK edge.
9. The user deasserts CSI_B, and the byte is ignored.
10. A byte is loaded on the rising CCLK edge.
11. A byte is loaded on the rising CCLK edge.
12. A byte is loaded on the rising CCLK edge.

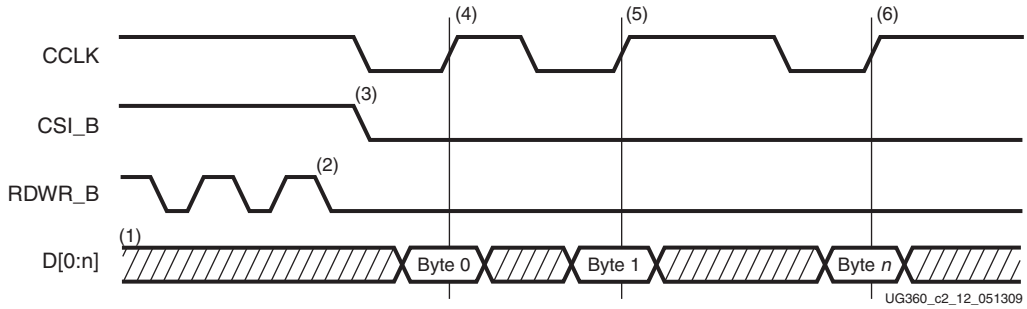


Figure 2-10: **Non-Continuous SelectMAP Data Loading with Controlled CCLK**

Notes relevant to Figure 2-10:

1. The Data pins are in the High-Z state while CSI_B is deasserted. The data bus can be x8, x16, or x32 (for Slave SelectMAP).
2. RDWR_B has no effect on the device while CSI_B is deasserted.
3. CSI_B is asserted by the user. The device begins loading configuration data on rising CCLK edges.
4. A byte is loaded on the rising CCLK edge.
5. A byte is loaded on the rising CCLK edge.
6. A byte is loaded on the rising CCLK edge.

SelectMAP Data Ordering

In many cases, SelectMAP configuration is driven by a user application residing on a microprocessor, CPLD, or in some cases another FPGA. In these applications, it is important to understand how the data ordering in the configuration data file corresponds to the data ordering expected by the FPGA.

In SelectMAP x8 mode, configuration data is loaded at one byte per CCLK, with the MSB of each byte presented to the D0 pin. This convention (D0 = MSB, D7 = LSB) *differs* from many other devices. For x16 and x32 modes, see Parallel Bus Bit Order, page 91. This convention can be a source of confusion when designing custom configuration solutions. Table 2-4 shows how to load the hexadecimal value `0xABCD` into the SelectMAP data bus.

Table 2-4: **Bit Ordering for SelectMAP 8-Bit Mode**

CCLK Cycle	Hex Equivalent	D0	D1	D2	D3	D4	D5	D6	D7
1	0xAB	1	0	1	0	1	0	1	1
2	0xCD	1	1	0	0	1	1	0	1

Notes:

1. D[0:7] represent the SelectMAP DATA pins.

Some applications can accommodate the non-conventional data ordering without difficulty. For other applications, it can be more convenient for the source configuration data file to be *bit swapped*, meaning that the bits in each byte of the data stream are reversed. For these applications, the Xilinx PROM file generation software can generate bit-swapped PROM files (see Configuration Data File Formats, page 87).

Figure 2-5 shows the bit ordering for the Virtex-6 FPGA x8, x16, and x32 modes. It also shows the bit ordering for the Virtex-4 FPGA x32 mode. Bit ordering for Virtex-5 FPGAs is the same as for Virtex-6 FPGAs.

Table 2-5: **Bit Ordering**

Virtex-6 and Virtex-5 FPGA Mode	Data Pins																													
	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**
x32	24	25	26	27	28	29	30	31	16	17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
x16																	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
x8																									0	1	2	3	4	5	6	7
Virtex-4 FPGA x32 Mode	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

Master SPI Configuration Interface

The FPGA pin connections used for Master SPI mode are shown in Figure 2-11 and described in Table 2-6. In Master SPI mode (M[2:0] = 001), the Virtex-6 FPGA configures itself from an attached industry-standard SPI serial flash PROM, as illustrated in Figure 2-12. The FPGA supplies the CCLK output clock from its internal oscillator and drives the clock input of the attached SPI flash PROM. For Virtex-6 FPGA Master SPI configurations, the default address always starts from 0. The SPI device must support I/Os of 2.5V or lower; otherwise level shifters are required because Virtex-6 devices do not directly support 3.3V I/O.

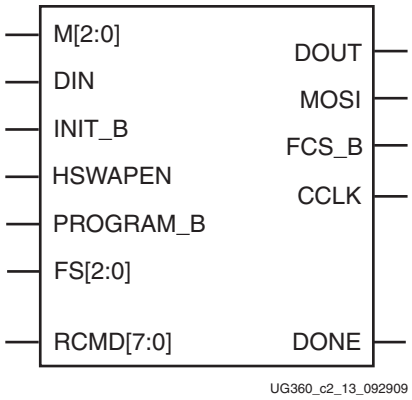


Figure 2-11: **Virtex-6 FPGA SPI Configuration Interface**

Table 2-6: **Virtex-6 FPGA SPI Configuration Interface Pins**

Pin Name	Type	Dedicated or Dual-Purpose	Description
M[2:0]	Input	Dedicated	Mode pins M[2:0] = `001` for SPI. These are sampled when INIT_B goes High.
HSWAPEN	Input	Dedicated	Controls I/O (except bank0 dedicated I/Os) Pull-up during configuration. A weak pull-up resistor is built into this pin. `0` = Pull-up during configuration `1` = 3-stated during configuration
DOUT	3-State Output	Dedicated	Used for serial daisy-chain configurations.
DONE	Bidirectional, Open-Drain, or Active	Dedicated	Active-High signal indicating configuration is complete: `0` = FPGA not configured `1` = FPGA configured
INIT_B	Bidirectional, Input, or Output, Open-Drain	Dedicated	From power-on reset or PROGRAM_B reset, INIT_B is driven Low, indicating that the FPGA is initializing (clearing) its configuration memory. Before the Mode pins are sampled, INIT_B is an open-drain, active-Low input and can be held Low to delay configuration. After the Mode pins are sampled, the INIT_B output indicates if a CRC error occurred during configuration or a readback CRC error occurred after configuration (when enabled): `0` = CRC or IDCODE error (DONE is Low) or Readback CRC Error (DONE is High and Readback CRC is enabled). `1` = No CRC error, housecleaning is complete (needs an external pull-up resistor). When the SEU detection function is enabled, INIT_B is optionally driven Low when a readback CRC error is detected.
PROGRAM_B	Input	Dedicated	Active-Low full-chip reset. This signal is edge sensitive before and during power-up and is level sensitive after power-up. PROGRAM_B cannot be used to delay configuration from power-up. See Delaying Configuration, page 100 for additional information.
FS[2:0]	Input	Dual-Purpose	SPI Flash Select Variant pins, sampled by the INIT_B rising edge. They are multiplexed with the D[2:0] pins. Refer to Table 2-7, page 45, for the Flash Select Variant options.
CCLK	Bidirectional, Output	Dedicated	Configuration clock output (to SPI).
FCS_B	Output	Dual-Purpose	Active-Low chip select output, clocked by the CCLK falling edge. If the SPI flash is not used after configuration, it is recommended that the FPGA design drive FCS_B High.

Table 2-6: **Virtex-6 FPGA SPI Configuration Interface Pins *(Cont'd)***

Pin Name	Type	Dedicated or Dual-Purpose	Description
MOSI	Output	Dual-Purpose	FPGA serial data output, clocked by the CCLK falling edge. This pin is multiplexed with the FOE_B pin.
DIN	Input	Dedicated	FPGA serial data input (from SPI), sampled by the CCLK rising edge.
RCMD[7:0]	Input	Dual-Purpose	Refer to the help section of the iMPACT software titled "Introduction to Indirect Programming - SPI or BPI Flash Memory" for a list of supported SPI flash devices that can be indirectly programmed. RCMD[7:0] are SPI read command strapping inputs (multiplexed on A[7:0]) when FS[2:0] = 001. They are sampled on the rising edge of INIT_B when used for SPI read command strapping.

Although SPI is a fairly standard four-wire interface, available SPI flash PROMs use different command protocols. The FPGA's flash select variant pins, FS[2:0], are sampled on the rising edge of the INIT_B signal, and they define how the FPGA communicates with the SPI flash. The FS[2:0] pins determine which SPI flash command the FPGA issues to start the read operation and the number of dummy bytes inserted before the FPGA expects to receive valid data from the SPI flash. Table 2-7 defines the SPI read command based on the FS[2:0] settings.

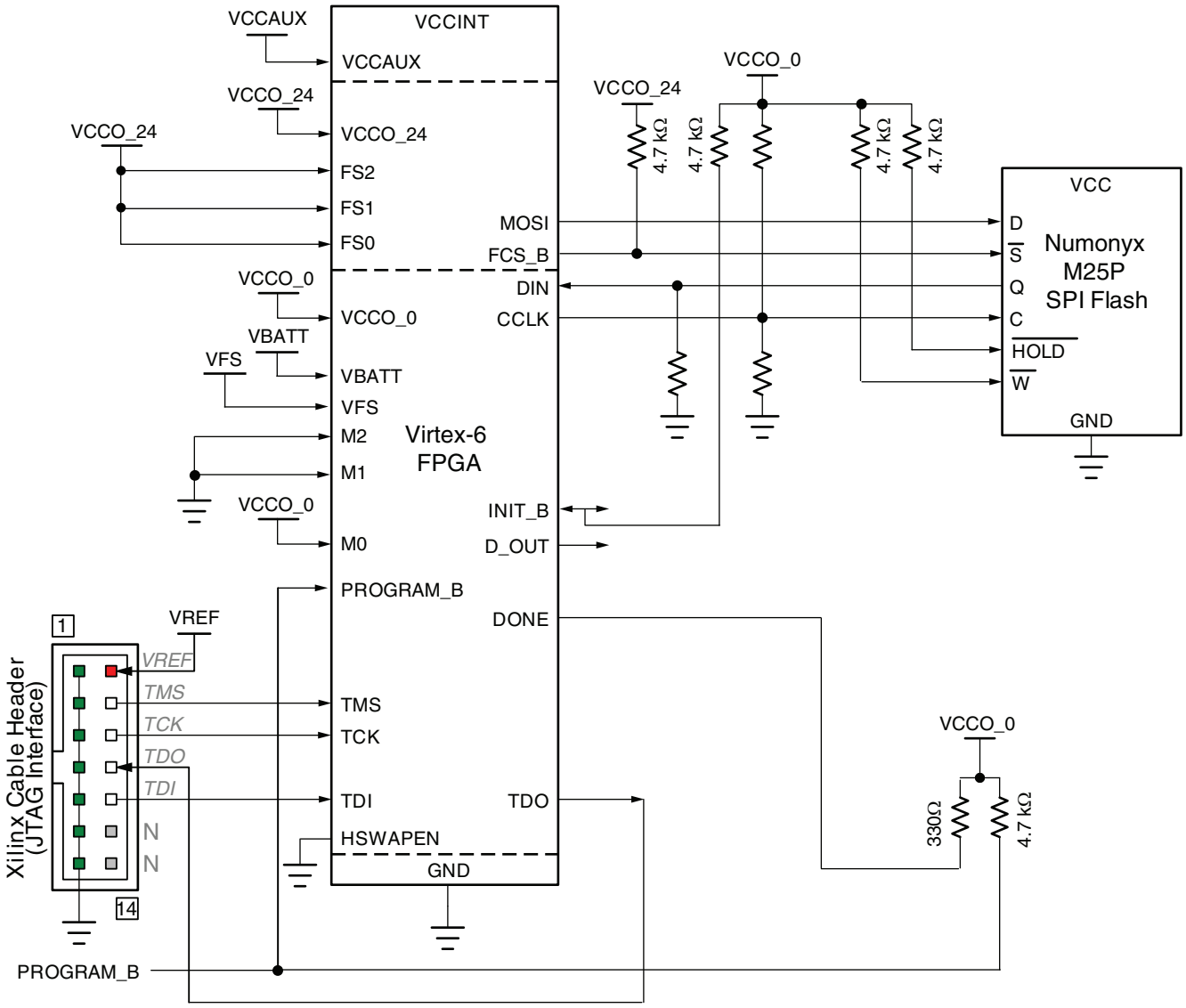
Table 2-7: **Virtex-6 FPGA SPI Read Command Flash Select Variant Table**

FS[2:0]	SPI Read Command	Comments
000	0xFF	
001	RCMD[7:0]	RCMD[7:0] on A[7:0] are sampled by the INIT_B rising edge, along with M[2:0] and FS[2:0]. RCMD[7:0] can be used to send read commands not provided by the FS selections. The timing requirements for FS[2:0] and RCMD[7:0] are the same as for M[2:0]. See the help section of the iMPACT software titled "Introduction to Indirect Programming – SPI or BPI Flash Memory" for a list of supported SPI flash devices that can be indirectly programmed.
010	0x52	
011	Reserved	
100	0x55	
101	0x03	x1 Read command, used by supported Numonyx and compatible devices.
110	0xE8	
111	0x0B	Recommended x1 Fast Read command, used by supported Numonyx and compatible devices.

The iMPACT programming software provides the ability to program an SPI serial flash indirectly using a Xilinx cable. The indirect programming method downloads a new FPGA design that provides a connection from the iMPACT software through the Virtex-6 FPGA to the SPI flash. Previous FPGA memory contents are overwritten by the indirect FPGA design and must be reloaded after an indirect operation is performed. For a list of supported SPI flash devices, refer to iMPACT help.

Some Virtex-6 FPGA SPI interface pins (DIN, CCLK) are dedicated for configuration and require the STARTUP_VIRTEX6 primitive to be used in the design for SPI flash access after configuration. Refer to STARTUP_VIRTEX6, page 75 for signal details.

Figure 2-12 shows the general connection diagram for SPI flash PROMs that support the `0x0B` FAST READ commands. If the SPI serial flash memory does not support I/Os of 2.5V or lower, an interfacing option is required for the Virtex-6 FPGAs, which do not natively support 3.3V I/O. Refer to XAPP899, *Interfacing Virtex-6 FPGAs with 3.3V I/O Standards*, for detailed options that meet the desired application goals. Figure 2-12 shows a Numonyx flash and one possible interface option on DIN to handle the SPI 3.3V input signal. An example simulation was performed using the M25P128_IBIS_SO16_V10 IBIS model. A value of 80Ω for the pull-down resistor was targeted using the M25P128 and Virtex-6 FPGA IBIS models. It is recommended to run the IBIS simulation for a given application to determine the appropriate pull-down value for the target board.



Refer to the Notes following this figure for related information.

UG360_c2_14_080712

Figure 2-12: **Virtex-6 FPGA SPI Configuration Interface**

Notes related to Figure 2-12:

1. Virtex-6 FPGA VCCO_0 supply input and Xilinx Cable V_{REF} must have the same voltage. Virtex-6 FPGA bank voltage VCCO_24 supplies the MOSI, FCS_B, and FS[2:0] signals.
2. The Virtex-6 FPGA does not support 3.3V I/O. An SPI device that supports 2.5V or lower I/O should be used, but a 3.3V flash can be interfaced via level shifters or a resistor option on the DIN signal, as shown in Figure 2-12. If the DIN pull-down resistor option is targeted, an IBIS simulation should be run for the given application to determine the appropriate pull-down value for the board.
3. FCS_B and MOSI are clocked by the CCLK falling edge.
4. DIN is clocked on the rising edge of the CCLK.

5. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines. The example parallel Thevenin termination shown in Figure 2-12 is only one possible termination topology. The advantages of this option are simplicity of design and application; however, this topology can be less desirable for some applications because it can dissipate more power. This trade-off must be weighed against other trade-offs to determine the optimum termination topography for an interface.
6. CCLK and DIN are dedicated Configuration I/Os.
7. FCS_B is a dual-mode I/O. MOSI is a dual-mode I/O, multiplexed with FOE_B. FS[2:0] are dual-mode I/Os sampled on the INIT_B rising edge, multiplexed with D[2:0].
8. The internal I/O pull-up resistors should be enabled for FCS_B, MOSI, and DIN.
9. There are additional pins on the SPI flash side, such as Write Protect and Hold. These pins are not used in FPGA configuration (read only). But they should be tied off appropriately according to the SPI vendor's specification.
10. If HSWAPEN is left unconnected or tied High, a pull-up resistor is required for FCS_B and MOSI.
11. If HSWAPEN is tied Low, the FCS_B and MOSI pins have internal weak pull-up resistors during configuration. After configuration, FCS_B and MOSI can be either controlled by I/O in user mode or by enabling a weak pull-up resistor through constraints.
12. CCLK always has a weak internal pull-up resistor. The CCLK frequency can be adjusted using the **`ConfigRate`** BitGen option.
13. An external pull-up resistor is required for the DONE pin.
14. Signal analysis (e.g., IBIS simulation) is recommended to ensure proper signal quality. SPI flash devices with low-impedance output drivers might require series termination (the series resistor is close to the SPI flash output) to manage the signal characteristics.
15. VFS is used for eFUSE programming. See eFUSE, page 107 for more details.
16. V_{BATT} is the power source for AES key storage. If AES encryption is not used, V_{BATT} can be tied to ground, V_{CCAUX}, or left unconnected. See Bitstream Security, page 101 for more details.

The Virtex-6 FPGA SPI flash timing diagram is shown in Figure 2-13.

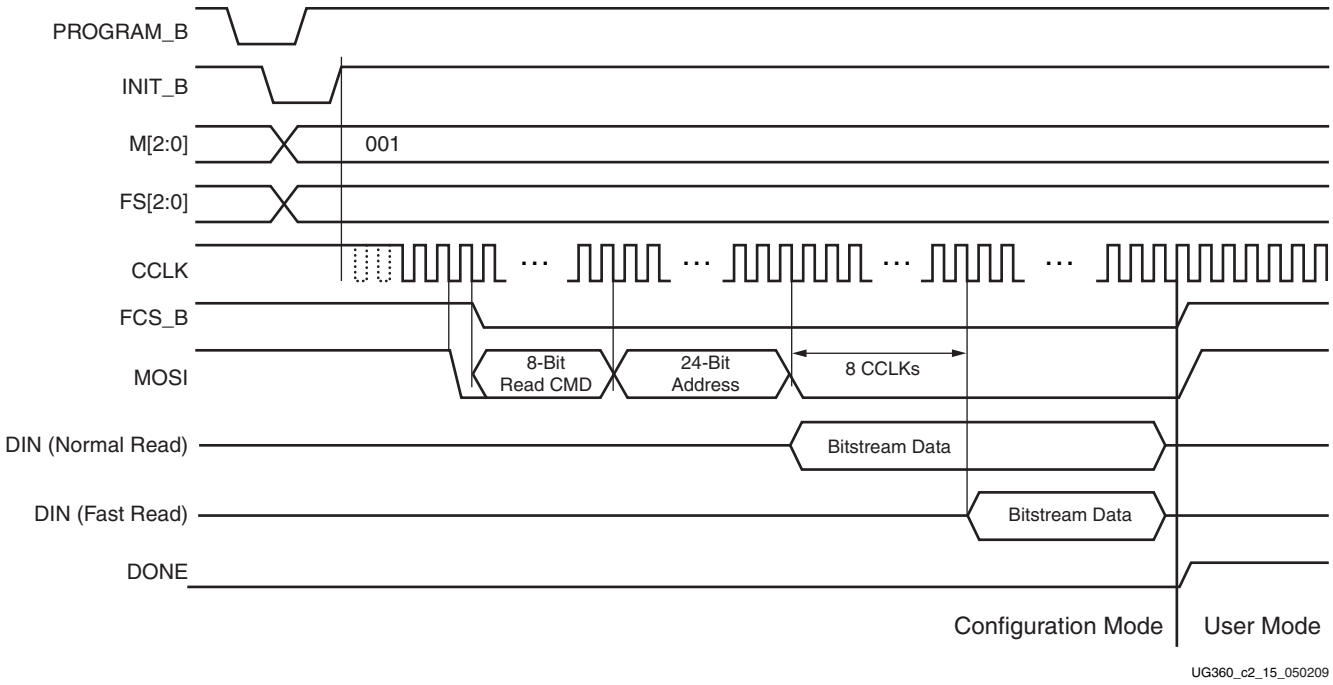


Figure 2-13: **Virtex-6 FPGA SPI Flash Timing Diagram**

Notes related to Figure 2-13:

1. The CCLK frequency starts off slowly (the default is approximately 2 MHz) providing sufficient margin for the FCS_B and MOSI activity. The CCLK frequency increases if the BitGen **`-g CONFIGRATE`** setting is utilized after the start of the read, where only the SPI clock-to-data output timing and FPGA DIN setup timing need to be considered.
2. When the SPI flash is not used after configuration, it is recommended to have the FPGA design drive the FCS_B High.

Power-On Sequence Precautions

At power-on, the FPGA automatically starts its configuration procedure. When the FPGA is in Master SPI configuration mode, the FPGA asserts FCS_B Low to select the SPI flash and drives a read command to the SPI flash. The SPI flash must be awake and ready to receive commands before the FPGA drives FCS_B Low and sends the read command.

Because different power rails can supply the FPGA and SPI flash or because the FPGA and SPI flash can respond at different times along the ramp of a shared power supply, special attention to the FPGA and SPI flash power-on sequence or power-on ramps is essential. The power-on sequence or power supply ramps can cause the FPGA to awake before the SPI flash or vice versa. In addition, some SPI flash devices specify a minimum time period, which can be several milliseconds from power-on, during which the device must not be selected. For many systems with near-simultaneous power supply ramps, the FPGA power-on reset time (TPOR) can sufficiently delay the start of the FPGA configuration procedure such that the SPI flash becomes ready before the start of the FPGA configuration procedure. In general, the system design must consider the affect of the power sequence, the power ramps, FPGA power-on reset timing, and SPI flash power-up timing on the timing relation between the start of FPGA configuration and the readiness of the SPI flash.

Check the *Virtex-6 FPGA Data Sheet: DC and Switching Characteristics* for Virtex-6 FPGA power supply requirements and timing. Check the SPI flash data sheet for the SPI flash power-up timing requirements.

Unlike previous FPGA families, holding the PROGRAM_B input Low from before or during the FPGA's power-on period does not hold the FPGA in configuration reset after the power-on-reset period. Virtex-6 FPGAs proceed with the standard power-on configuration sequence because the PROGRAM_B pin is an edge-sensitive input and only recognizes a falling edge after the FPGA's power-on-reset period. Instead, one of these system design approaches can ensure that the SPI flash is ready to receive commands before the FPGA starts its configuration procedure:

- Control the sequence of the power supplies such that the SPI flash is certain to be powered and ready to receive commands before the FPGA begins its configuration procedure.
- Hold the FPGA INIT_B pin Low using an open-drain driver from power-up to delay the start of the FPGA configuration procedure and release the INIT_B pin to High after the SPI flash becomes ready to receive commands.

SPI Serial Daisy Chain

In a serial daisy chain application, the leading device can be in SPI mode and all downstream devices in Slave Serial mode. In this case, all configuration bitstreams can be stored inside one SPI device. The bitstream format for Master and Slave Serial daisy chains is exactly the same. See Serial Daisy Chains, page 161 for details.

Master BPI Configuration Interface

From the Master Byte-wide Peripheral Interface (BPI), the Virtex-6 FPGA can configure itself in BPI-Up (M[2:0]=`010`) or BPI-Down (M[2:0] = `011`) mode. From this interface, the Virtex-6 FPGA configures itself from an industry-standard parallel NOR flash PROM, as illustrated in Figure 2-15.

The FPGA drives all 26 address lines to access the attached parallel flash during configuration. For configuration from industry-standard parallel NOR flash, only asynchronous read mode is used. In asynchronous read mode, the FPGA drives the address bus and the flash PROM drives back the bitstream data. Bus widths of x8 and x16 are supported. Bus widths are auto detected, as described in Bus Width Auto Detection, page 88. Refer to DS617, *Platform Flash XL High-Density Configuration and Storage Device* for the BPI-compatible flash device from Xilinx.

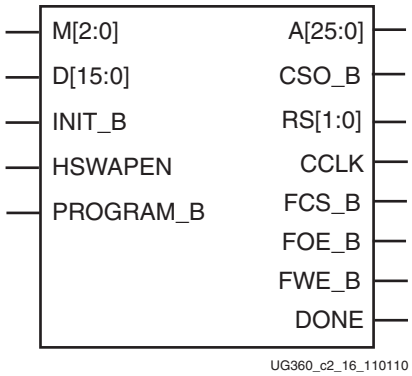


Figure 2-14: **Virtex-6 FPGA BPI Configuration Interface**

Table 2-8 defines the BPI configuration interface pins.

If the FPGA is subject to reprogramming or fallback during configuration from the BPI flash, then the INIT_B pin can be connected to the BPI reset to set the BPI into a known state.

Table 2-8: **Virtex-6 FPGA BPI Configuration Interface Pins**

Pin Name	Type	Dedicated or Dual-Purpose	Description
M[2:0]	Input	Dedicated	The Mode pins determine the BPI mode: `010` = BPI-Up mode `011` = BPI-Down mode
HSWAPEN	Input	Dedicated	Controls I/O (except Bank 0 dedicated I/Os) pull-up resistors during configuration. This pin has a built-in weak pull-up resistor. `0` = Pull-up during configuration `1` = 3-state during configuration
DONE	Bidirectional, Open-Drain, or Active	Dedicated	Active-High signal indicating configuration is complete: `0` = FPGA not configured `1` = FPGA configured
INIT_B	Bidirectional, Input, or Output, Open-Drain	Dedicated	From power-on reset or PROGRAM_B reset, INIT_B is driven Low, indicating that the FPGA is initializing (clearing) its configuration memory. Before the Mode pins are sampled, INIT_B is an open-drain, active-Low input and can be held Low to delay configuration. After the Mode pins are sampled, the INIT_B output indicates if a CRC error occurred during configuration or a readback CRC error occurred after configuration (when enabled): `0` = CRC or IDCODE error (DONE is Low) or Readback CRC Error (DONE is High and Readback CRC is enabled). `1` = No CRC error, housecleaning is complete (needs an external pull-up resistor). When the SEU detection function is enabled, INIT_B is optionally driven Low when a readback CRC error is detected.
PROGRAM_B	Input	Dedicated	Active-Low full-chip reset. This signal is edge sensitive before and during power-up and is level sensitive after power-up. PROGRAM_B cannot be used to delay configuration from power-up. See Delaying Configuration, page 100 for additional information.
CCLK	Bidirectional, Output	Dedicated	Configuration clock output. CCLK does not directly connect to BPI flash but is used internally to generate the address and sample read data.
FCS_B	Output	Dual-Purpose	Active-Low flash chip select output. This output is actively driven Low during configuration. It has a weak pull-up resistor during configuration. By default, this signal has a weak pull-down resistor after configuration.
FOE_B	Output	Dual-Purpose	Active-Low flash output enable. This output is actively driven Low during configuration. It has a weak pull-up resistor during configuration. By default, this signal has a weak pull-down resistor after configuration.
FWE_B	Output	Dual-Purpose	Active-Low flash write enable. This output is actively driven High during configuration. It has a weak pull-up resistor during configuration. By default, this signal has a weak pull-down resistor after configuration.

Table 2-8: **Virtex-6 FPGA BPI Configuration Interface Pins *(Cont'd)***

Pin Name	Type	Dedicated or Dual-Purpose	Description
A[25:0]	Output	Dual-Purpose	Address output.
D[15:0]	Input	Dual-Purpose	Data input, sampled by the rising edge of the FPGA CCLK. Only x8 data bus widths are supported for configuration from BPI flash with AES encrypted bitstreams.
RS[1:0]	Output	Dual-Purpose	Revision Select pins. Not used for typical single bitstream applications. RS[1:0] are 3-stated and pulled up with weak resistors during the initial configuration if the HSWAPEN pin enables the pull ups. If pull ups are disabled, then a weak external pull up is required (after power-up or assertion of PROGRAM_B). RS[1:0] are actively driven Low to load the fallback bitstream when a configuration error is detected. RS[1:0] can also be controlled by the user through the bitstream or ICAP.
CSO_B	3-State Output	Dual-Purpose	Parallel daisy chain. Active-Low chip select output during configuration; otherwise this output is 3-stated. Not used in single FPGA applications.

In BPI modes, the CCLK output is not connected to the BPI flash device. However, the FPGA outputs an address after the rising edge of CCLK, and the data is still sampled on the next rising edge of CCLK. The timing parameters related to BPI use CCLK as a reference. Virtex-6 FPGA BPI modes also support asynchronous page-mode reads to allow an increase in the CCLK frequency. See Page Mode Support, page 57 for details.

In the BPI-Up mode, the address starts at 0 and increments by 1 until the DONE pin is asserted. If the address reaches the maximum value (`26'h3FFFFFF`) and configuration is not done (DONE is not asserted), an error flag is raised in the status register, and fallback reconfiguration starts.

In the BPI-Down mode, the address start at `26'h3FFFFFF` and decrements by 1 until the DONE pin is asserted. If the address reaches the bottom (`26'h0`), and configuration is still not done (DONE is not asserted), an error flag is raised in the status register and fallback reconfiguration starts.

The iMPACT programming software provides the ability to program Top boot parallel NOR flash using an indirect programming method. This requires a small piece of IP to be added to the FPGA design, which provides a connection from the iMPACT software through the Virtex-6 device to the flash device. For a list of supported BPI devices, see Table 2-9 and for details on the latest software release, see the iMPACT help section titled "Introduction to Indirect Programming – SPI or BPI Flash Memory." For software flow details, see XAPP973, *Indirect Programming of BPI PROMs with Virtex-5 FPGAs*, which also applies to Virtex-6 devices.

> ***Caution!*** The Virtex-6 FPGA drives all BPI address lines during configuration, and the iMPACT software drives all of the address lines during indirect flash programming regardless of the targeted flash density. Because the address lines are multi-purpose pins, all functions for which the pins are used must be considered, including during configuration, indirect programming, and user design operations.

Table 2-9: **Supported BPI Flash Family**

Vendor	Family	Density	iMPACT Indirect Programming Support
Micron	P30	128 Mb - 1 Gb	Yes
Micron	J3	128 Mb - 256 Mb	Legacy (not recommended for new designs, J3D or J3F supports up to ISE software 12.2)
Spansion	S29GLxxxP	128 Mb - 1 Gb	Yes
Spansion	S29GLxxxS	128 Mb - 1 Gb	Yes

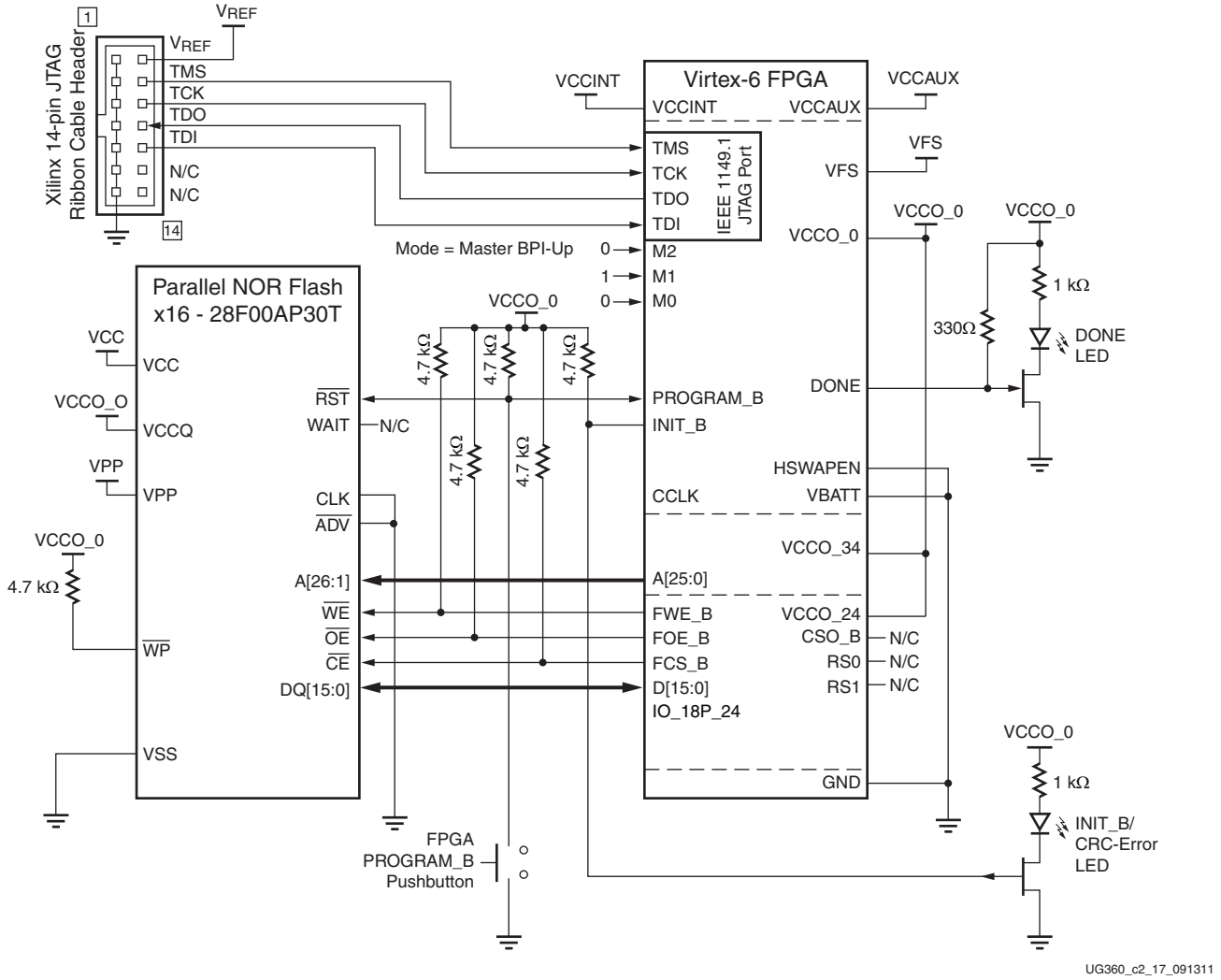


Figure 2-15: **Virtex-6 FPGA Master BPI Configuration Interface Example with Micron P30 Flash**

Additional notes related to Figure 2-15:

1. Virtex-6 FPGA VCCO_0 supply input and Xilinx Cable V_{REF} must be tied to the same voltage. Virtex-6 FPGA bank voltage VCCO_24 supplies FCS_B, FOE_B, FWE_B, D[15:0], and CSO_B signals and bank voltage VCCO_34 supplies the A[25:0] bus.
2. M[2:0] = `010` for BPI-Up mode and M[2:0] = `011` for BPI-Down mode.

3. Figure 2-15 shows the x16 BPI interface. For x8 BPI interfaces, only D[7:0] are used. See Bus Width Auto Detection, page 88.
4. Sending a bitstream to the data pin follows the same bit-swapping rule as in SelectMAP mode. See Parallel Bus Bit Order, page 91.
5. The CCLK outputs are not used to connect to flash but are used to sample flash read data during configuration. All timings are referenced to CCLK. The CCLK pin must *not* be driven or tied High or Low.
6. The RS[1:0] pins are not connected as shown in Figure 2-15. These output pins are only required for MultiBoot configuration. See Chapter 8, Reconfiguration and MultiBoot. Use iMPACT 11.3 (or later) to indirectly program the flash when the upper address bits are routed through the RS[1:0] pins to the flash.
7. When using the XCF128X Platform Flash XL in BPI-Up mode, the Virtex-6 FPGA User I/O pin IO_18P_24 is reserved for the indirect programming core. This signal is connected to the device L# signal. For detailed connection diagrams, refer to UG438, *Platform Flash XL Configuration and Storage Device User Guide*.
8. An external pull-up resistor is required on DONE.
9. For daisy chaining FPGAs in BPI mode, see Chapter 10, Advanced Configuration Interfaces.
10. The example shown is for a Micron P30 connection. For other supported BPI flash, the BPI flash vendor data sheet should be referred to for details on signal connectivity. To prevent address misalignment, close attention should be paid to the flash family address LSB for the byte/word mode used. Not all flash families use the A0 as the address LSB.
11. Only x8 data bus widths are supported for configuration from a BPI flash with AES encrypted bitstreams.
12. The JTAG connections are shown for a simple, single-device JTAG scan chain. When multiple devices are on the JTAG scan chain, use the proper IEEE Std 1149.1 daisy-chain technique to connect the JTAG signals. The TCK signal integrity is critical for JTAG operation. Route, terminate, and if necessary, buffer the TCK signal appropriately to ensure signal integrity for the devices in the JTAG scan chain.
13. The FPGA mode (M[2:0]) pins are shown set to Master BPI-Up mode (`010`). The implementation of a board-level option that enables the user to change the FPGA mode pins to JTAG mode (`101`) is recommended to enable JTAG-based debug capability for the FPGA during design. This is not required, but the JTAG mode setting ensures that there is no interference from the Master BPI-Up configuration during debug.
14. The FPGA HSWAPEN pin is tied to ground in this sample schematic. HSWAPEN can alternatively be tied High. Review the FPGA data sheet for the effect of the alternate HSWAPEN setting.
15. The Virtex-6 FPGA does not support AES decryption in the 16-bit wide configuration mode shown in this sample schematic. Thus, the V_{BATT} decryptor key battery power supply is unused and is tied to GND.
16. This sample schematic supports single bitstream configuration. Thus, FPGA RS[1:0] pins are not connected in this sample schematic. See the FPGA configuration user guide for use of the RS[1:0] pins. Use iMPACT 11.3 (or later) to indirectly program the supported flash when a few address bits are routed through the RS[1:0] pins to the flash.
17. DONE LED lights when DONE is High. INIT_B/CRC-Error LED lights when INIT_B is Low. Adjust the LED circuits and pull-up values for desired lighting results.

18. See DS152, *Virtex-6 FPGA Data Sheet: DC and Switching Characteristics*, for the V_{CCINT} supply voltage.
19. VFS is used for eFUSE programming. See eFUSE, page 107 for more details.

Figure 2-16 shows the BPI-Up configuration waveforms.

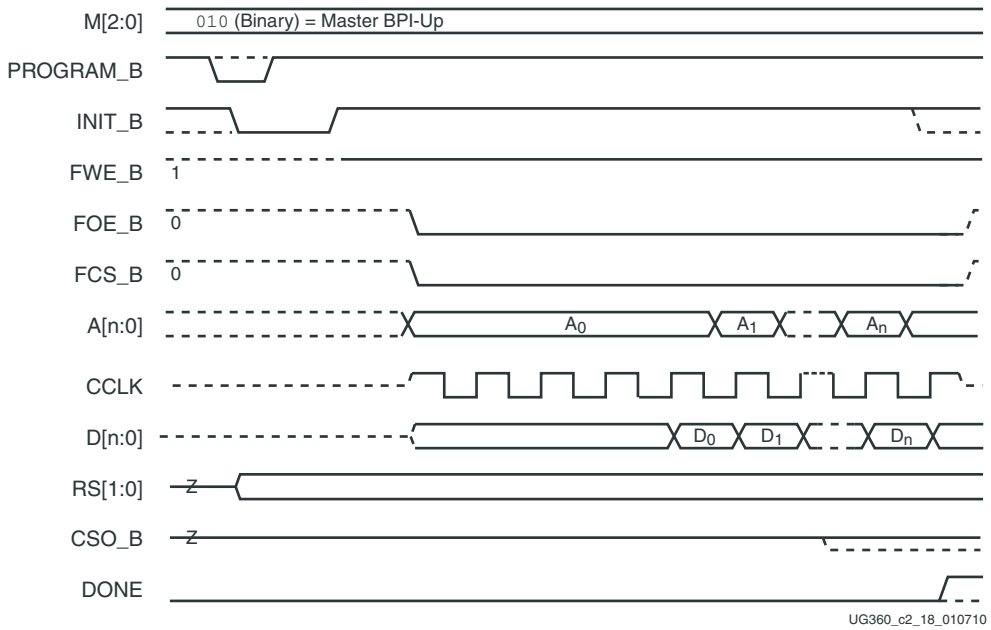


Figure 2-16: **Virtex-6 FPGA BPI-Up Configuration Waveforms**

Notes related to Figure 2-16:

1. Configuration starts with power-up (PROGRAM_B is pulled and kept High) or with a High-Low-High pulse to PROGRAM_B.
2. For power-up configuration, INIT_B starts Low. For PROGRAM_B initiated configuration, INIT_B drives Low when PROGRAM_B is pulsed Low.
3. RS[1:0] are typically high impedance. However, a MultiBoot (or Fallback event in BPI mode) can cause RS[1:0] to drive High or Low.
4. INIT_B releases at the end of the FPGA's internal initialization process. An external resistor pulls INIT_B High. On the rising edge of INIT_B, the FPGA samples its M[2:0] pins to determine the configuration mode.
5. Upon determining the Master BPI-Up mode from the M[2:0] pins, the FPGA drives FWE_B High, FOE_B Low, and FCS_B Low.
6. For Master mode, the FPGA drives CCLK for T_{ICCK} time after the rising edge of INIT_B.
7. The FPGA drives the initial address (A0) through its A[n:0] pins and holds the initial address at $T_{INITADDR}$ CCLK cycles. For a power-on configuration, the initial address is `0x0000000`. For a MultiBoot-triggered configuration, the address can be different.
8. The FPGA registers the 8-bit or 16-bit data word on the rising edge of CCLK.
9. For a multi-FPGA parallel configuration daisy chain, CSO_B can drive Low to select the next FPGA in the daisy chain for bitstream loading from the data bus.
10. Near the last 8-bit or 16-bit word (depending on the flash device) of the bitstream, the FPGA begins its startup sequence.

11. If the FPGA detects a CRC error during bitstream delivery, the FPGA drives its INIT_B pin Low. DONE stays Low.
12. If the FPGA successfully receives the bitstream, the FPGA releases its DONE pin during the startup sequence and an external resistor pulls DONE High.
13. During the startup sequence, the multi-purpose pins are activated with their configurations from the user's FPGA design. If not used in the FPGA design, the high-impedance FCS_B pin is pulled High by the external resistor to disable the flash.
14. At the end of configuration, the master CCLK is disabled into a high-impedance state by default.
15. Dual-Purpose configuration I/O switches to User mode after the GTS_cycle. By default, this is one cycle after DONE goes High.

Determining the Maximum Configuration Clock Frequency

In Master BPI mode, the FPGA delivers the configuration clock. The FPGA's master configuration clock frequency is set through the BitGen **`-g ConfigRate`** option. The BitGen **`-g ConfigRate`** option sets the nominal configuration clock frequency.

The default BitGen ConfigRate setting of 2 is recommended. This default value sets the nominal master CCLK frequency to 2 MHz, which satisfies timing requirements for the leading BPI flash families. The BitGen ConfigRate setting can be increased for a faster configuration time, if the timing requirements discussed in this section are satisfied. When determining a valid ConfigRate setting, these timing parameters must be considered:

- FPGA nominal master CCLK frequency (BitGen ConfigRate)
- FPGA Master CCLK frequency tolerance ($F_{MCCKTOL}$)
- ADDR[25:0] outputs valid after CCLK rising edge (T_{BPICCO})
- BPI flash address to output valid (access) time (T_{ACC})
- FPGA data setup time (T_{BPIDCC})

The FPGA's master configuration clock has a tolerance of $F_{MCCKTOL}$. Due to the master configuration clock tolerance ($F_{MCCKTOL}$), the BitGen **`-g ConfigRate`** option must be checked so that the period for the worst-case (fastest) master CCLK frequency is greater than the sum of the FPGA address valid time, BPI flash access time, and FPGA setup time, as shown in Equation 2-1.

$$\frac{1}{ConfigRate \times (1 + F_{MCCKTOL_{MAX}})} \geq T_{BPICCO} + T_{ACC} + T_{BPIDCC} \qquad \textit{Equation 2-1}$$

Power-On Sequence Precautions

At power-on, the FPGA automatically starts its configuration procedure. When the FPGA is in a Master-BPI configuration mode, the FPGA asserts FCS_B Low and drives a sequence of addresses to read the bitstream from a BPI flash. The BPI flash must be ready for asynchronous reads before the FPGA drives FCS_B Low and outputs the first address to ensure the BPI flash can output the stored bitstream.

Because different power rails can supply the FPGA and BPI flash or because the FPGA and BPI flash can respond at different times along the ramp of a shared power supply, special attention to the FPGA and BPI flash power-on sequence or power-on ramps is essential. The power-on sequence or power supply ramps can cause the FPGA to awake before the BPI flash or vice versa. For many systems with near-simultaneous power supply ramps, the FPGA power-on reset time (T_{POR}) can sufficiently delay the start of the FPGA configuration procedure such that the BPI flash becomes ready before the start of the FPGA

configuration procedure. In general, the system design must consider the effect of the power sequence, the power ramps, FPGA power-on reset time, and BPI flash power-on reset time on the timing relation between the start of FPGA configuration and the readiness of the BPI flash for asynchronous reads. Check the *Virtex-6 FPGA Data Sheet: DC and Switching Characteristics* data sheet for Virtex-6 FPGA power supply requirements and timing. Check DS617, *Platform Flash XL High-Density Configuration and Storage Device* for the BPI flash power supply requirements and timing.

Unlike previous FPGA families, holding the PROGRAM_B input Low from before or during the FPGA's power-on period does not hold the FPGA in configuration reset after the power-on-reset period. Virtex-6 FPGAs proceed with the standard power-on configuration sequence because the PROGRAM_B pin is an edge-sensitive input and only recognizes a falling edge after the FPGA's power-on-reset period. Instead, one of these system design approaches can ensure that the BPI flash is ready for asynchronous reads before the FPGA starts its configuration procedure:

- Control the sequence of the power supplies such that the BPI flash is certain to be powered and ready for asynchronous reads before the FPGA begins its configuration procedure.
- Hold the FPGA INIT_B pin Low from power-up to delay the start of the FPGA configuration procedure and release the INIT_B pin to High after the BPI flash becomes ready for asynchronous reads.

Page Mode Support

Many NOR flash devices support asynchronous page reads. The first access to a page usually takes the longest time (~100 ns), subsequent accesses to the same page take less time (~25 ns). These parameters are bitstream programmable in Virtex-6 devices to take advantage of page reads and maximize the CCLK frequency:

- Page sizes of 1 (default), 4, or 8.

  If the actual flash page size is larger then 8, the value of 8 should be used to maximize the efficiency.
- First access CCLK cycles of 1 (default), 2, 3, or 4. CCLK cycles must be 1 if the page size is 1.
- CCLK frequency

The sequence of page-mode operation is controlled by the Virtex-6 FPGA bitstream (see Table 6-15). After an FPGA reset, the default page size is 1, the first access CCLK is 1, and the master CCLK is running at slowest default frequency. The COR0 register contains master CCLK frequency control bits (see Configuration Options Register 0 (COR0), page 120). The COR1 register contains BPI flash page mode control bits (see Configuration Options Register 1 (COR1), page 123). After the COR1 register is programmed, the BPI address timing switches at the page boundary as shown in Figure 2-17. When the SWITCH command is received, the master CCLK switches to a user-desired frequency, using it to load the rest of the configuration.

Refer to the BitGen section of UG628, *Command Line Tools User Guide* for details on BitGen options.

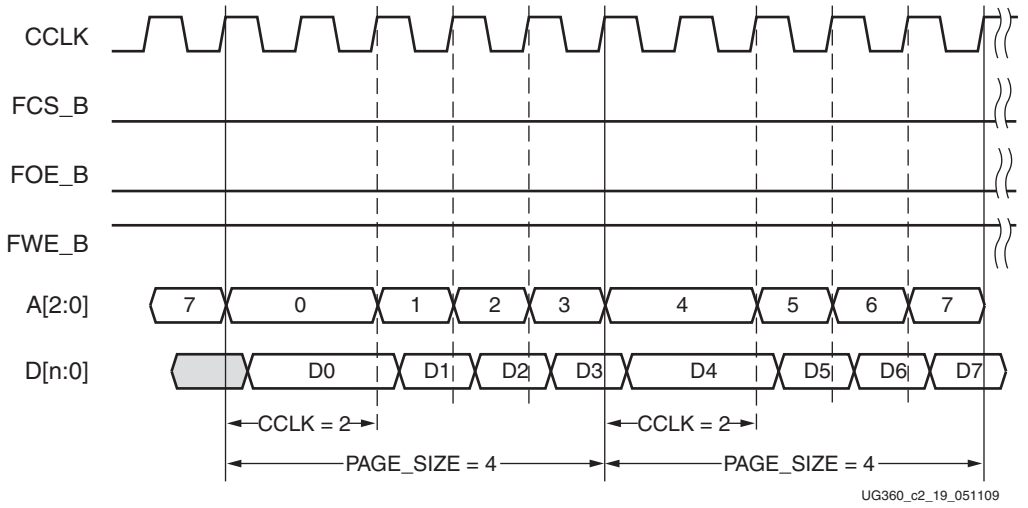


Figure 2-17: **BPI-Up Waveforms (Page Size = 4 and First Access CCLK = 2)**

Notes related to Figure 2-17:

1. Figure 2-17 shows BPI-Up mode, a page size of 4, and a first access CCLK of 2.
2. The data bus width can be x8 (n = 7) or x16 (n = 15).
3. For BPI-Down mode, the A[25:0] bus is extended for the desired CCLK cycles when A[1:0] = `2'b11` for a page size of 4.

JTAG Interface

From the four-pin JTAG interface, the Virtex-6 FPGA can be configured using Xilinx software (iMPACT or ChipScope™ software) and a Xilinx cable, directly from a processor or CPLD customer-specific design, or using third-party boundary-scan tools. The JTAG specific mode setting is (M[2:0] = `101`).

JTAG is the most important connection to place on the board. Although JTAG commands have priority over mode settings, it is recommended to have an M[2:0] option to enable the JTAG mode and operations without potential conflict from other configuration modes. For more information, refer to Chapter 3, Boundary-Scan and JTAG Configuration.

Board Layout for Configuration Clock (CCLK)

The Virtex-6 FPGA configuration I/Os use the LVCMOS fast slew rate 12 mA standard. This I/O standard has faster edge rates to support higher configuration frequencies. This requires more attention to PCB trace routing and termination for proper signal integrity.

The CCLK pin is the clock source for the Virtex-6 FPGA configuration logic during Master modes. The CCLK output must be free from reflections to avoid double-clocking. Xilinx recommends simulating the CCLK distribution with an IBIS simulator (such as HyperLynx) to check for glitches on each CLK input and output, including the CCLK of the master FPGA.

Figure 2-18 shows the master CCLK structure.

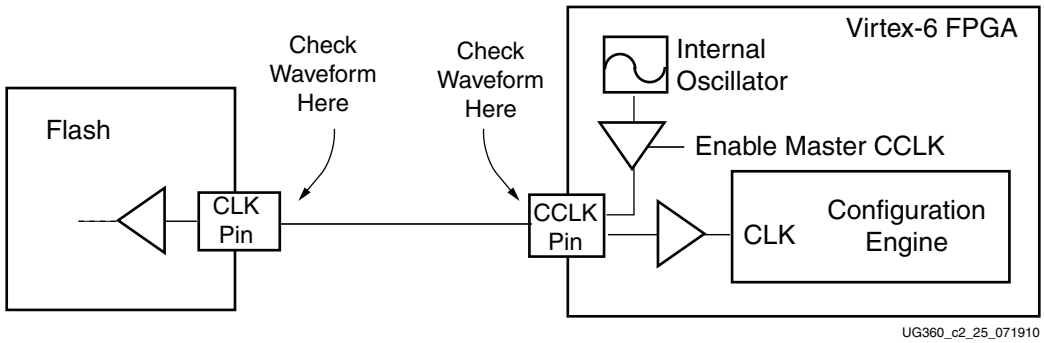


Figure 2-18: **Master CCLK Structure**

Basic Guidelines

- Route the CCLK net as a 50Ω controlled impedance transmission line.
- Always route the CCLK net without any branching; do *not* use a *star* topology (Figure 2-22).
- Stubs, if necessary, are recommended to be shorter than 8 mm (0.3 inches).

Familiarity with the advantages and disadvantages of available termination techniques helps the designer choose the best option for the target application. Refer to UG373, *Virtex-6 FPGA PCB Design Guide*, for detailed guidelines to determine the appropriate topology for the intended application and detailed trade-offs. Figure 2-19 through Figure 2-21 show a few possible topologies for CCLK distribution. Because the Master CCLK goes to high impedance at the end of the configuration sequence, the examples using parallel termination can be less desirable than other termination options because more power is dissipated. This trade-off must be weighed against other factors to determine the optimal termination topography for an interface.

Figure 2-19 shows the basic point-to-point topology for one CCLK driver (FPGA master) and one CCLK receiver (PROM or FPGA slave).

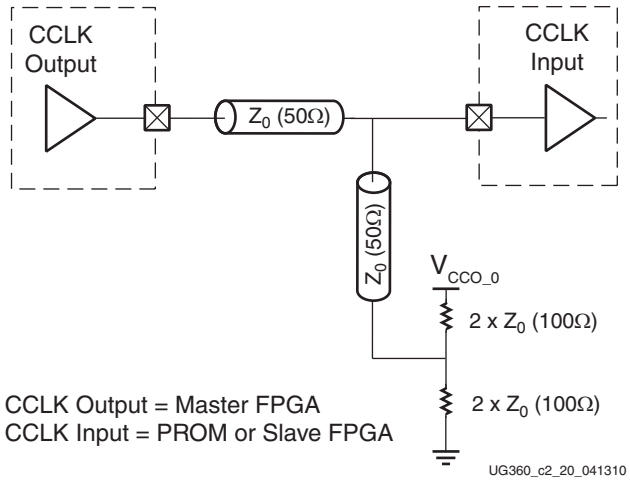


Figure 2-19: **Point-to-Point: One CCLK Output, One CCLK Input**

Figure 2-20 shows the basic multi-drop *flyby* topology for one CCLK driver and two CCLK receivers. The stub at CCLK input *1* has a length constraint.

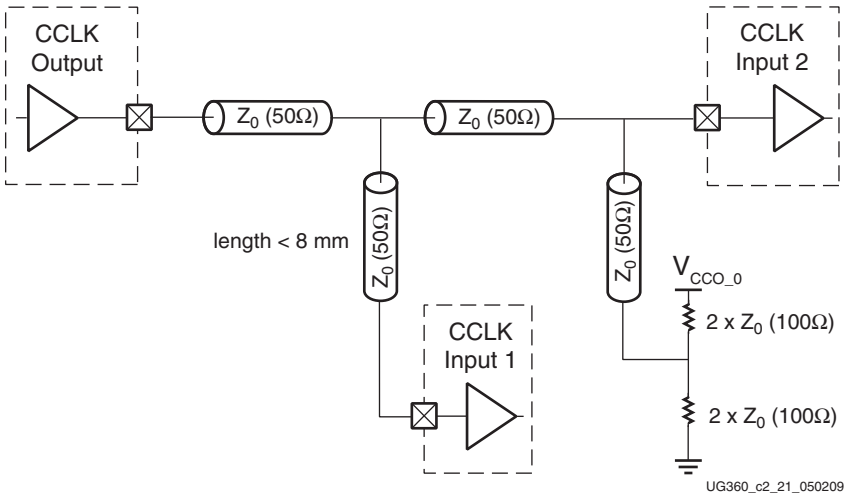


Figure 2-20: **Multi-Drop: One CCLK Output, Two CCLK Inputs**

Figure 2-21 shows the multi-drop *flyby* topology for one CCLK driver and more than two CCLK receivers (four in this example). All CCLK inputs except input *4* have length constraints.

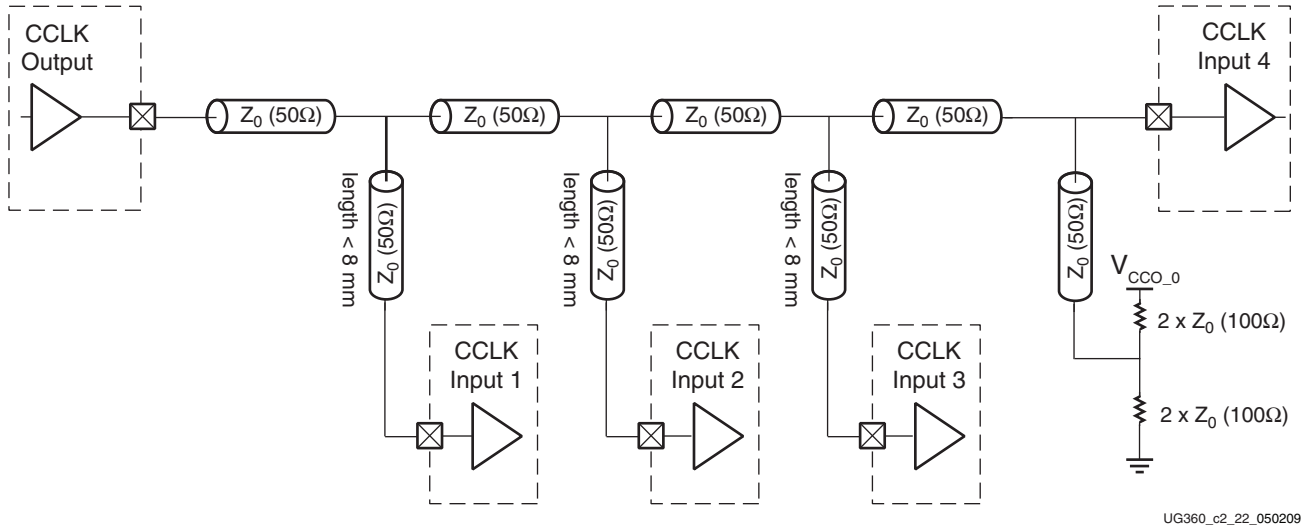


Figure 2-21: **Multi-Drop: One CCLK Output, More Than Two CCLK Inputs**

Figure 2-22 shows a *star* topology where the transmission line branches to the multiple CCLK inputs. The branch point creates a significant impedance discontinuity. This arrangement is **Not Recommended**.

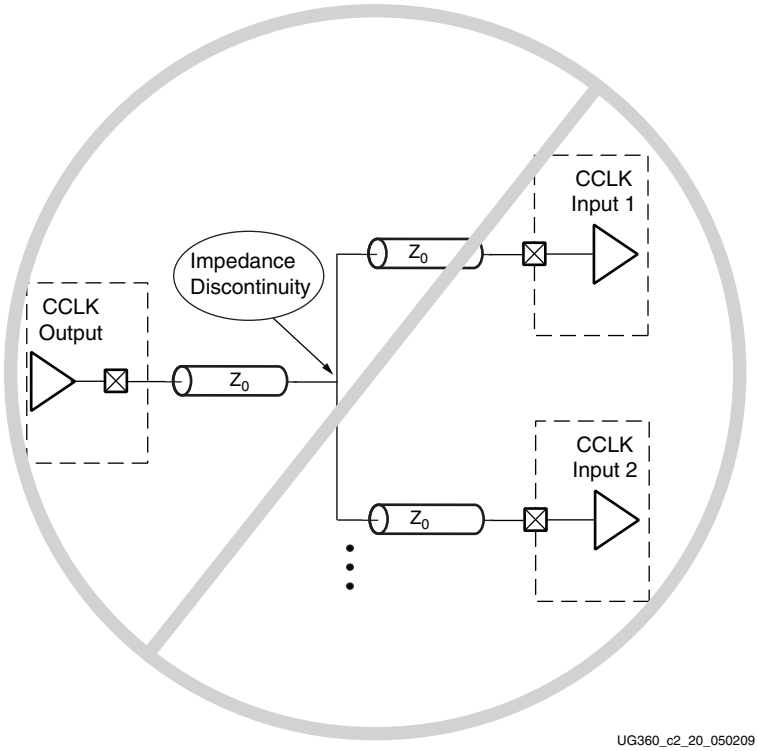


Figure 2-22: **Not Recommended**
Star Topology: One CCLK Output, Two CCLK Input

XILINX®

Chapter 3

Boundary-Scan and JTAG Configuration

Introduction

Virtex®-6 devices support IEEE standard 1149.1, defining Test Access Port (TAP) and boundary-scan architecture respectively. These standards ensure the board-level integrity of individual components and the interconnections between them. In addition to connectivity testing, boundary-scan architecture offers flexibility for vendor-specific instructions, such as configure and verify, which add the capability of loading configuration data directly to FPGAs and compliant PROMs. Test Access Port and boundary-scan architecture is commonly referred to collectively as JTAG.

Boundary-Scan for Virtex-6 Devices Using IEEE Standard 1149.1

The Virtex-6 family is fully compliant with the IEEE Standard 1149.1 Test Access Port and Boundary-Scan Architecture. The architecture includes all mandatory elements defined in the IEEE 1149.1 standard. These elements include the TAP, the TAP controller, the Instruction register, the instruction decoder, the Boundary register, and the Bypass register. The Virtex-6 family also supports a 32-bit Device Identification register and a Configuration register. Outlined in the following sections are the details of the JTAG architecture for Virtex-6 devices.

Test Access Port (TAP)

The Virtex-6 FPGA TAP contains four mandatory dedicated pins as specified by the protocol in Virtex-6 devices and in typical JTAG architecture (see Figure 11-1). Three input pins and one output pin control the IEEE Std 1149.1 boundary-scan TAP controller. Optional control pins, such as Test Reset (TRST), and enable pins 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 IEEE Std 1149.1 boundary-scan TAP controller is a state machine (16 states), shown in Chapter 11, Advanced JTAG Configurations.

The four mandatory TAP pins are outlined in Table 3-1.

Table 3-1: **Virtex-6 FPGA TAP Controller Pins**

Pin	Direction	Pre-Configuration Internal Pull Resistor	Description
TDI	In	Pull-up(1)	**Test Data In**. This pin is the serial input to all JTAG instruction and data registers. The state of the TAP controller and the current instruction determine the register that 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	Out	NA	**Test Data Out**. This pin is the serial output for all JTAG instruction and data registers. The state of the TAP controller and the current instruction determine the register (instruction or data) that 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. TDO is an active driver output.
TMS	In	Pull-up(1)	**Test Mode Select**. 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	In	NA	**Test Clock**. This pin is the JTAG Test Clock. TCK sequences the TAP controller and the JTAG registers in the Virtex-6 devices.

Notes:

1. As specified by the IEEE Std 1149.1, the TMS and TDI pins both have internal pull-up resistors. These internal pull-up resistors are active, regardless of the mode selected. Refer to DS152, *Virtex-6 FPGA Data Sheet: DC and Switching Characteristics*, for internal pull-up values. BitGen can be used to enable the pull-up or pull-down resistor after configuration for all four mandatory pins. See UG628, *Command Line Tools User Guide* for more information.

Boundary-Scan Timing Parameters

Characterization data for some of the most commonly requested timing parameters, shown in Figure 3-1, are listed in the *Virtex-6 FPGA Data Sheet* in the Configuration Switching Characteristics table. For more information on the configuration flow details, refer to Chapter 11, Advanced JTAG Configurations.

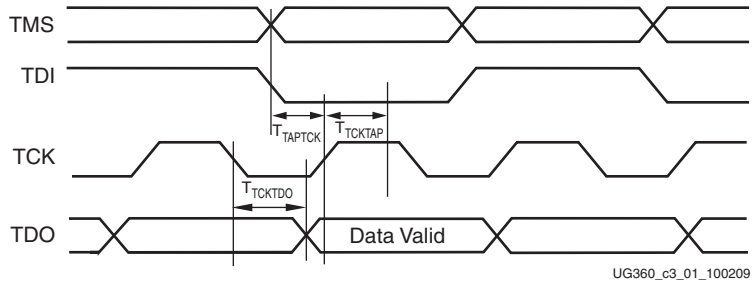


Figure 3-1: **Virtex-6 FPGA Boundary-Scan Port Timing Waveforms**

Using Boundary-Scan in Virtex-6 Devices

For single-device configuration, the TAP controller commands are issued automatically if the part is being configured with Xilinx® iMPACT software. The download cable must be attached to the appropriate four JTAG pins (TMS, TCK, TDI and TDO) to deliver the bitstream automatically from the computer port to the Virtex-6 FPGA. The iMPACT software automatically checks for proper connections and drives the commands to deliver and/or verify that the configuration bits are properly managed.

Figure 3-2 shows a typical JTAG setup with the simple connection required to attach a single device to a JTAG signal header, which can be driven from a processor, or a Xilinx programming cable under control of the iMPACT software. TCK is the clock used for boundary-scan operations. The TDO-TDI connections create a serial datapath for shifting data through the JTAG chain. TMS controls the transition between states in the TAP controller; see Chapter 11, Advanced JTAG Configurations. Proper physical connections of all of these signals are essential to JTAG functionality.

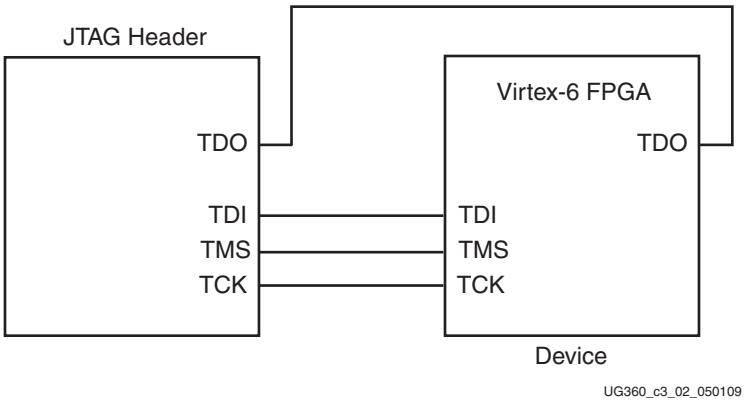


Figure 3-2: **Single Device JTAG Programming Connections**

Multiple Device Configuration

It is possible to configure multiple Virtex-6 devices in a chain. (See Figure 3-3.)

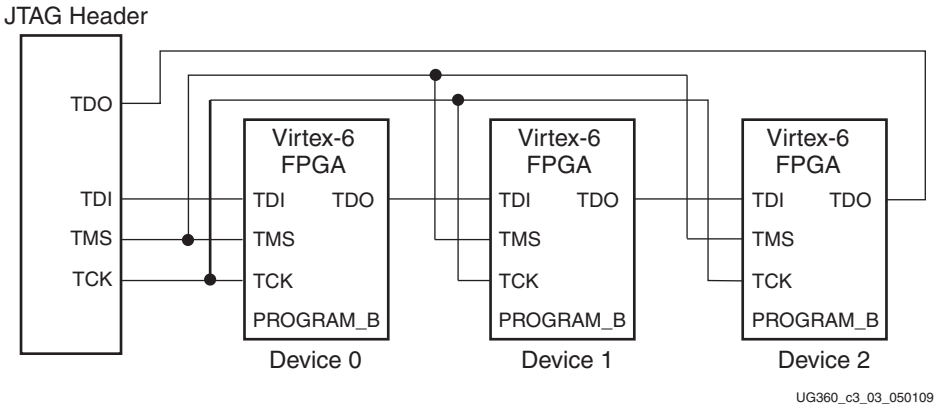


Figure 3-3: **Boundary-Scan Chain of Devices**

If JTAG is the only configuration mode, then PROGRAM_B, INIT_B, and DONE can each be connected to separate pull-up resistors.

The devices in the JTAG chain are configured one at a time. The multiple device configuration steps can be applied to any size chain as long as an excellent signal integrity is maintained. The iMPACT software automatically discovers the devices in the chain, starting from the one nearest to TDI coming from the JTAG header and the iMPACT software.

JTAG inputs use the V_{CCO_0} supply, also known as V_{CC_CONFIG}, for JTAG operations.

Chapter 11, Advanced JTAG Configurations provides a detailed description of the various TAP controller states, the JTAG instructions, and the architecture of the boundary-scan chain.

For details on the standard boundary-scan instructions EXTEST, INTEST, and BYPASS, refer to IEEE Std 1149.1 and Chapter 11, Advanced JTAG Configurations.

For further information on the startup sequence, bitstream, and internal configuration registers referenced here, refer to Chapter 6, Configuration Details and Chapter 11, Advanced JTAG Configurations.

Boundary-Scan Design Considerations

JTAG Signal Routing

The TCK and TMS signals go to all devices in the chain; consequently, their signal quality is important. For example, TCK should transition monotonically at all receivers to ensure proper JTAG functionality and must be properly terminated. The quality of TCK can limit the maximum frequency for reliable JTAG configuration.

Additionally, if the chain is large (three devices or more), TMS and TCK should be buffered to ensure that they have sufficient drive strength at all receivers, and the voltage at logic High must be compatible with all devices in the chain.

When interfacing to devices from other manufacturers, optional JTAG signals can be present (such as TRST and enables) and might need to be driven.

Providing Power

To ensure proper power-on behavior, the guidelines in the *Virtex-6 FPGA Data Sheet* must be followed. The power supplies should ramp monotonically within the power supply ramp time range specified in the *Virtex-6 FPGA Data Sheet*. All supply voltages should be within the recommended operating ranges; any dips in V_{CCINT} below V_{DRINT} or V_{CCAUX} below V_{DRI} (see the *Virtex-6 FPGA Data Sheet* for specific values) can result in loss of configuration data.

V_{CCO_0}, also known as V_{CC_CONFIG}, determines the I/O voltage for the configuration interface (JTAG and other modes). The voltage provided must be compatible with all configuration interfaces that will be used.

To ensure boundary-scan functionality, the guidelines in DS152, *Virtex-6 FPGA Data Sheet* for powering unused serial transceiver tiles must be followed.

Configuring through Boundary-Scan

If the Virtex-6 device is configured via boundary-scan on power-up, it is recommended to tie the mode pins to the JTAG configuration mode settings, i.e., `101` (M2 = `1`, M1 = `0`, M0 = `1`). This is not required because the JTAG configuration mode is available regardless

of the mode pin settings; setting the mode pins to JTAG only removes potential conflicts on the configuration pins.

The configuration flow for Virtex-6 device configuration with JTAG is discussed in Chapter 11, Advanced JTAG Configurations. This chapter includes details about the command sequences used for configuring a Virtex-6 device as a single device through boundary-scan or as part of a multiple-device boundary-scan chain. A configured device can be reconfigured by toggling the TAP and entering a CFG_IN instruction after pulsing the PROGRAM_B pin or issuing the shut-down sequence. See Chapter 11, Advanced JTAG Configurations.

Xilinx has proprietary programming cables (Parallel and USB) and boundary-scan programming software (iMPACT) for prototyping purposes. These are not intended for production environments but can be highly useful for verifying FPGA implementations and JTAG chain integrity.

When trying to access other devices in the JTAG chain, it is important to know the size of the instruction register length to ensure that the correct device receives the appropriate signals. This information can be found in the BSDL file for the device, provided in ISE software.

One of the most common boundary-scan vendor-specific instructions is the configure instruction. If the Virtex-6 device is configured via JTAG, the configure instructions occur independent from the mode pins. Chapter 11, Advanced JTAG Configurationsdetails device configuration through JTAG. The Virtex-6 FPGA JTAG configuration algorithm uses the SVF-based flow, provided in XAPP058, *Xilinx In-System Programming Using an Embedded Microcontroller*.

XILINX®

Chapter 4

User Primitives

The configuration primitives described in this chapter are provided for users to access FPGA configuration resources during or after FPGA configuration. The primitives include: BSCAN_VIRTEX6, CAPTURE_VIRTEX6, DNA_PORT, EFUSE_USR, FRAME_ECC_VIRTEX6, ICAP_VIRTEX6, STARTUP_VIRTEX6, and USER_ACCESS_VIRTEX6. For additional information and instantiation templates, refer to UG623, *Virtex-6 Libraries Guide for HDL Designs*.

BSCAN_VIRTEX6

JTAG is a standard four-pin interface: TCK, TMS, TDI, and TDO. Many applications are built around this interface. The JTAG TAP controller is a dedicated state machine inside the configuration logic. BSCAN_VIRTEX6 provides access between the JTAG TAP controller and user logic in fabric. There are up to four instances of BSCAN_VIRTEX6 for each device, and each instance of this design element handles one JTAG USER instruction (USER1 through USER4) as set with the JTAG_CHAIN attribute. To handle all four USER instructions, the user instantiates four of these elements and sets the JTAG_CHAIN attribute appropriately. Table 4-1 lists the BSCAN_VIRTEX6 fabric pins.

Table 4-1: **BSCAN_VIRTEX6 Pin Table**

Pin Name	Type	Description
SEL	Output	Active-High interface selection output. SEL = 1 when the JTAG instruction register holds the corresponding USER1-4 instruction. Change in Update_IR state. SEL changes on the falling edge of TCK in the UPDATE_IR state of the TAP controller.
RESET	Output	Active-High reset output. RESET = 1 during the TEST-LOGIC-RESET state.
TDI	Output	Fed through directly from the FPGA TDI pin.
DRCK	Output	DRCK is the same as TCK in the Capture_DR and Shift_DR states. If the interface is not selected by the instruction register, DRCK remains High.
CAPTURE	Output	Active-High pulse indicating the Capture_DR state. This signal is asserted on the falling edge of TCK.
UPDATE	Output	Active-High pulse indicating the Update_DR state. This signal is asserted on the falling edge of TCK.
SHIFT	Output	Active-High pulse indicating the Shift_DR state. This signal is asserted on the falling edge of TCK.

Table 4-1: **BSCAN_VIRTEX6 Pin Table *(Cont'd)***

Pin Name	Type	Description
TDO	Input	TDO input driven from the user fabric logic. This signal is internally sampled on the falling edge before being driven out to the FPGA TDO pin.
TMS	Output	Fabric connection to TAP input pin.
TCK	Output	Fabric connection to TAP Clock pin.
RUNTEST	Output	Asserted when TAP controller is in Run Test Idle state.

Table 4-2 defines the BSCAN_VIRTEX6 attribute.

Table 4-2: **BSCAN_VIRTEX6 Attribute**

Attribute	Type	Allowed Values (Default)	Description
JTAG_CHAIN	Integer	1, 2, 3, or 4 (1)	Sets the JTAG USER instruction number that this instance of the element will handle.

CAPTURE_VIRTEX6

The CAPTURE_VIRTEX6 primitive is used to capture I/O, CLB, and block RAM output flip-flop status, and then read back through the configuration interface. The CAP input is sampled by CLK to generate an internal gcap signal. The I/O and CLB flip-flop status are captured into an FPGA configuration memory cell when the gcap signal is High. There are operation modes, a one-shot mode, or a continuous mode.

In one-shot mode, after the first CAP falling edge, gcap is held to 0 to avoid further capturing. An explicit RCAP command is required to re-arm the capture circuit. To limit the readback operation to a single data capture, the ONESHOT = TRUE attribute is added to this element.

In continuous mode, the CAP input is simply sampled by CLK, and becomes the gcap signal, allowing the user to control when to capture.

CAPTURE_VIRTEX6 should not operate simultaneously with the FRAME_ECC_VIRTEX6 primitive or the Readback CRC function (see Chapter 9, Readback CRC) because capturing a value into configuration memory might cause a false error.

Table 4-3: **CAPTURE_VIRTEX6 Pin Table**

Pin Name	Type	Description
CLK	Input	Clock for sampling the CAP input.
CAP	Input	Active-High capture enable. The CAP input is sampled by the rising edge of CLK.

Table 4-4 defines the CAPTURE_VIRTEX6 attribute.

Table 4-4: **CAPTURE_VIRTEX6 Attribute**

Attribute	Type	Allowed Values (Default)	Description
ONESHOT	Boolean	TRUE, FALSE (TRUE)	Specifies the procedure for performing single readback per the CAP trigger.

DNA_PORT

The DNA_PORT provides access to a dedicated shift register, which can be loaded with the Device DNA data bits (ID) for a given Virtex®-6 device. In addition to shifting out the DNA data bits, this component allows for the inclusion of supplemental data bits for additional user data or allow for the DNA data to rollover (repeat DNA data after initial data has been shifted out). This component is primarily used in conjunction with other circuitry to build anti-cloning protection for the FPGA bitstream from possible theft.

Usage

The DNA_PORT component must be instantiated to be used in a design. The instantiation template is found within the ISE® software Project Navigator HDL templates. The instance declaration must be placed within the code. All inputs and outputs must be connected to the design to ensure proper operation.

To access the Device DNA data, the shift register must first be loaded by setting the active-High READ signal for one clock cycle. After the shift register is loaded, the data can be synchronously shifted out by enabling the active-High SHIFT input and capturing the data from the DOUT output port. If desired, additional data can be appended to the end of the 57-bit shift register by connecting the appropriate logic to the DIN port. If DNA data rollover is desired, the DOUT port can be connected directly to the DIN port to allow for the same data to be shifted out after completing the 57-bit shift operation. If no additional data is necessary, the DIN port can be tied to a logic zero. The attribute SIM_DNA_VALUE can optionally be set to allow for simulation of a possible DNA data sequence.

Table 4-5 lists the DNA_PORT connections. Refer to Device Identifier (Device DNA), page 127 for details.

Table 4-5: **DNA_PORT Primitive Connections**

Port Name	Direction	Function
DOUT	Output	Serial-shifted output data.
DIN	Input	User data input to the shift register.
READ	Input	Synchronous load of the shift register with the Device DNA data. A READ operation overrides a SHIFT operation.
SHIFT	Input	Active-High shift enable input.
CLK	Input	Input clock to the shift register.

Table 4-6 defines the DNA_PORT attribute.

Table 4-6: **DNA_PORT Attribute**

Attribute	Type	Allowed Values	Description
SIM_DNA_VALUE	57-bit vector	Any 57-bit value	Specifies a DNA value for simulation purposes (the actual value is specific to the particular device used).

EFUSE_USR

The EFUSE_USR primitive provides internal access to the 32 non-volatile fuses that can store bits specific to the design (e.g., a unique ID associated with each design). These eFUSEs must be externally programmed through JTAG.

Table 4-7 defines the EFUSE_USR pins.

Table 4-7: **EFUSE_USR Pin Table**

Pin Name	Type	Description
EFUSEUSR[31:0]	Output	Contains the 32-bit user eFUSE register value.

Table 4-8 defines the EFUSE_USR attribute.

Table 4-8: **EFUSE_USR Attribute**

Name	Type	Allowed Values (Default)	Description
SIM_EFUSE_VALUE	Hexadecimal	32'h00000000 to 32'hffffffff (32'h00000000)	Causes the simulation model to drive a static value onto these pins.

FRAME_ECC_VIRTEX6

The FRAME_ECC_VIRTEX6 primitive is similar to the FRAME_ECC_VIRTEX5 primitive with the addition of the CRCERROR flag for the Read Back CRC (RBCRC) error in Virtex-6 devices. When RBCRC is enabled, both Frame ECC and CRC are checked by the dedicated configuration logic.

The Virtex-6 FPGA Frame ECC logic detects single or double bit errors in configuration frame data. It uses a 13-bit Hamming code parity value that is calculated based on the frame data generated by BitGen. During readback, the Frame ECC logic calculates a *syndrome* value using all the bits in the frame including the ECC bits. If the bits have not changed from the original programmed values, SYNDROME[12:0] are all zeros. If a single bit has changed, including any of the ECC bits, the location of the bit is indicated by SYNDROME[11:0]. If two bits have changed, SYNDROME[12] is 0 and the remaining bits are non-zero. If more than two bits have changed, SYNDROME[12:0] are indeterminate. The *error* output of the block is asserted if one or two bits have changed.

To use the Frame ECC logic, the FRAME_ECC _VIRTEX6 primitive must be instantiated in the user's design, and readback must be performed using SelectMAP, JTAG, or ICAP interfaces. At the end of each frame of readback, the *syndrome_valid* signal is asserted for one cycle of the readback clock (CCLK, TCK, or ICAP_CLK). The number of cycles required to read back a frame varies with the interface used.

Repairing bits that have changed requires a user design. The FRAME_ECC_VIRTEX6 logic does *not* repair changed bits. The design must be able to store at least one frame of data or be able to fetch original frames of data for reload. A single frame is 2,592 bits. The simplest operation is:

1. A frame is read out through ICAP and stored in block RAM. The frame address is generated as each frame is read.
2. If an error is indicated by the error output of the FRAME_ECC block, the readback is halted and the SYNDROME value is saved. If SYNDROME[12] is 0 and SYNDROME[11:0] are non zero, the whole frame must be restored. If SYNDROME[12] is 1, SYNDROME[11:0] is used to locate the error bit in the saved frame.
3. The repaired frame is then written back into the frame address generated in step 1.
4. Readback then resumes on the next frame address.

Table 4-9 defines the FRAME_ECC_VIRTEX6 pins. See Chapter 9, Readback CRC, for more information.

Table 4-9: **FRAME_ECC_VIRTEX6 Pin Table**

Pin Name	Type	Description
SYNDROMEVALID	Output	Frame ECC syndrome valid pulse. Active one cycle for each frame. Used to sample ERROR and SYNDROME[12:0].
ECCERROR	Output	When SYNDROMEVALID is active, this output indicates if a frame error was detected or not: • ERROR = 1 when SYNDROME[12:0] is non-zero. • ERROR = 0 when SYNDROME[12:0] is all zeros.
SYNDROME[12:0]	Output	When SYNDROMEVALID is active, this output reflects the frame error condition: • No bit error: [12]==0, [11:0] ==0 • One bit error: [12]==1, [11:0] !=0 • Two bit errors: [12]==0, [11:0] != 0 • More than two bits: SYNDROME is unpredictable • Parity Bit Error: [12]==1, [11:0]==0
CRCERROR	Output	RBCRC error. See Chapter 9, Readback CRC.
FAR[23:0]	Output	Frame Address Register Value. • SEU Correction/Injection and ICAP applications can benefit from being able to see the FAR register. • This output can point to the EFAR or FAR configuration register depending on the FARSRC attribute.
SYNWORD[6:0]	Output	Out Word address of error. • The index (0 → 80) of the 32-bit word in the frame where an ECC error has been detected. Decoded from SYNDROME. • Valid when ECCERRORSINGLE is High.
SYNBIT[4:0]	Output	Bit address of error. • The index (0 → 31) of the bit w/ error in the word pointed to by SYNWORD in the frame detected. • Valid when ECCERRORSINGLE is High.
ECCERRORSINGLE	Output	Indicates a single-bit Frame ECC error detected.

Table 4-10 defines the FRAME_ECC_VIRTEX6 attributes.

Table 4-10: **FRAME_ECC_VIRTEX6 Attributes**

Name	Type	Settings (Default)	Description
FARSRC	String	“EFAR”, “FAR” (“EFAR”)	Determines if the output of FAR[23:0] configuration register points to the FAR (address of RBCRC) or EFAR (address of the error bit). Sets configuration option register bit CTL0[7].
FRAME_RBT_IN_FILENAME	String	“NONE”, <String> (“NONE”)	This input file contains frame data information for the raw bitstream (RBT) file. The FRAME_ECC model parses this file, calculates ECC, and outputs any error conditions.

ICAP_VIRTEX6

The ICAP_VIRTEX6 primitive works the same way as the SelectMAP configuration interface except it is on the fabric side, ICAP has a separate read/write bus, as opposed to the bidirectional bus in SelectMAP, and ICAP ignores the bus width auto-detect pattern and sets the data width via the ICAP_WIDTH parameter. ICAP has three data width selections through the ICAP_WIDTH parameter: x8, x16, and x32. The general SelectMAP timing diagrams and the SelectMAP bitstream ordering information, described in the SelectMAP Configuration Interface in Chapter 2, is applicable to ICAP. It allows the user to access configuration registers, readback configuration data, or partially reconfigure the FPGA after configuration is done.

Table 4-11: **ICAP_VIRTEX6 Pin Table**

Pin Name	Type	Description
CLK	Input	ICAP interface clock.
CSB	Input	Active-Low ICAP interface select. Equivalent to CSI_B in the SelectMAP interface.
RDWRB	Input	0 = WRITE, 1 = READ.
I[31:0]	Input	ICAP write data bus. The bus width depends on ICAP_WIDTH parameter. The bit ordering is identical to the SelectMAP interface. See SelectMAP Data Ordering in Chapter 2.
O[31:0]	Output	ICAP read data bus. The bus width depends on the ICAP_WIDTH parameter. The bit ordering is identical to the SelectMAP interface. See SelectMAP Data Ordering in Chapter 2.
BUSY	Output	Active-High busy status. Only used in read operations. BUSY remains Low during writes.

Table 4-12 defines the ICAP attribute.

Table 4-12: **ICAP Attribute**

Name	Type	Settings (Default)	Description
DEVICE_ID	HEX	<ID Code> (32'h04244093)	Specifies the pre-programmed Device ID code. See Table 6-13, page 97 for the list of Virtex-6 FPGA ID codes.
ICAP_WIDTH	ENUM	X8, X16, X32 (X8)	Selects the ICAP bus width.
SIM_CFG_FILE_NAME	String	"NONE", <String> ("NONE")	Specifies the raw bitstream (RBT) file to be parsed by the simulation model.

STARTUP_VIRTEX6

The STARTUP_VIRTEX6 primitive provides a fabric interface to allow users to control some of global signals after End Of Startup (EOS).

Table 4-13: **STARTUP_VIRTEX6 Pin Table**

Pin Name	Type	Description
EOS	Output	End of startup pin. This active-High pin is an output into the FPGA fabric. This pin echoes the EOS flag into the FPGA fabric. This output pin can be used as a reset signal.
CLK	Input	User startup clock pin. This pin is an input from the FPGA fabric and drives the startup clock for the device startup sequence.
GSR	Input	Global set/reset pin. This pin is an active-High input from the FPGA fabric. When this input is asserted, all flip-flops are restored to their initial value in the bitstream. For most applications, this pin should be tied Low.
GTS	Input	Global tristate pin. This active-High pin is an input from the FPGA fabric. When this input is asserted, all user I/Os are put into a high-Z state. For most applications, this pin should be tied Low.
USRCCLKO	Input	CCLK pin. This pin is an input from the FPGA fabric. This pin drives a custom, fabric-generated clock frequency onto the FPGA CCLK pin. This is useful for post-configuration access of external SPI flash devices.
USRCCLKTS	Input	User CCLK 3-state enable to CCLK pin. This pin is an input from the FPGA fabric. When this input is High, the FPGA CCLK pin is put into a high-Z state. Generally, this should be tied Low to prevent the CCLK pin going to a high-Z state.
USRDONEO	Input	DONE pin output value. This pin is an input from the FPGA fabric. This pin directly drives the FPGA DONE pin.

Table 4-13: **STARTUP_VIRTEX6 Pin Table *(Cont'd)***

Pin Name	Type	Description
USRDONETS	Input	User DONE 3-state enable to DONE pin. This pin is an input from the FPGA fabric. When this input is High, DONE is put into a high-Z state. Generally, this pin should be tied Low. Tying it High inhibits the assertion of DONE.
TCKSPI	Output	Direct from the TCK pin. This pin is an output into the FPGA fabric. It is a direct echo of the CCLK clock being driven onto the FPGA's configuration interface. This pin is useful for synchronizing an internal state machine to CCLK.
DINSPI	Output	Direct from the DIN pin. This pin is an output into the FPGA fabric. The data on this pin is the serial data being read from an SPI flash device connected to the FPGA. This pin is useful for reading back SPI flash contents.
CFGMCLK	Output	Configuration internal oscillator clock output. This pin is an output into the FPGA fabric. This pin outputs a clock signal with a typical 50 MHz frequency that is sourced from the FPGA's internal ring oscillator.
CFGCLK	Output	Configuration logic main clock output. This pin is an output into the FPGA fabric. This pin outputs a clock signal where the typical frequency is defined by the BitGen option **`-g configrate`**.
PREQ	Output	PROGRAM_B pulse or JPROGRAM request to FPGA logic. This pin is an output into the FPGA fabric. This pin is the "request" from the PROGRAM_B state machine to reset the device. This allows the assertion of the PROGRAM_B request to be gated until the design is in a state where the reset can be completed. This pin is only enabled if the PROG_USR attribute is set. It can be left open/floating for "safe" operation.
PACK	Input	PROGRAM_B or JPROGRAM acknowledge. This pin is an input from the FPGA fabric. This pin "acknowledges" the assertion of the PROGRAM_B signal and allows the remainder of the PROGRAM_B state machine to continue resetting the FPGA. This pin is only enabled if the PROG_USR attribute is set. It can be tied Low for "safe" operations.
KEYCLEARB	Input	Clear AES Decrypter Key from battery-backed RAM. This pin is an input from the FPGA fabric. This pin, when held Low for 250 ns ($T_{PROGRAM}$), erases the contents of the decryption keys from the battery-backed RAM (BBRAM). This pin is only enabled if the PROG_USR attribute is set.

TCKSPI and DINSPI are added to allow fabric access to dedicated TCK and DIN pins. These pins can be used for access to an SPI flash post configuration from the FPGA logic.

CFGMCLK is driven by the configuration internal oscillator. Its rate is approximately 50 MHz and can be used as a generic clock source instead of a ring oscillator in the FPGA logic.

CFGCLK is the main configuration clock.

Table 4-14 defines the STARTUP_VIRTEX6 attribute.

Table 4-14: **STARTUP_VIRTEX6 Attribute**

Attribute	Type	Allowed Values (Default)	Description
PROG_USR	Boolean	TRUE, FALSE (FALSE)	Activate program event security feature.

USR_ACCESS_VIRTEX6

The User Access Register (USR_ACCESS_VIRTEX6) is a 32-bit register that allows data from the bitstream to be directly accessible by the FPGA logic. The register has three outputs: CFGCLK, the 32-bit Data bus, and the DATAVALID signal that is asserted for one cycle of the configuration data source clock whenever a new value is available. The configuration data source clock can be CCLK or TCK.

The use model for this block allows data from a bitstream data storage source (e.g., PROM) to be accessed by the fabric after the FPGA has been configured. To accomplish this, the STARTUP_VIRTEX6 block should also be instantiated. The STARTUP_VIRTEX6 block has inputs that allow the user to take over the CCLK and DONE pins after the EOS signal is asserted. These pins are: USRCCLKO, USRCCLKTS, USRDONEO, and USRDONETS. The BitGen option **`-g DONE_cycle:Keep`** is used to prevent the DONE pin from going High and resetting the PROM. The USRCCLKO pin should be connected to a controlled clock in the fabric. In the bitstream, a write to the USR_ACCESS_VIRTEX6 register that follows the normal configuration bitstream provides the data. After EOS is asserted, the data packet is loaded by clocking the USRCCLKO pin while keeping USRCCLKTS Low (it can be tied Low in this usage).

Alternatively, the User Access register can provide a single 32-bit constant value to the fabric as an alternative to using a block RAM or LUT RAM to hold a constant.

Table 4-15: **USR_ACCESS_VIRTEX6 Pin Table**

Pin Name	Type	Description
CFGCLK	Output	Configuration clock.
DATA[31:0]	Output	User access data from bitstream.
DATAVALID	Output	DATA[31:0] is ready.

XILINX®

Chapter 5

Dynamic Reconfiguration Port (DRP)

Dynamic Reconfiguration of Functional Blocks

Background

In the Virtex®-6 family of FPGAs, the configuration memory is used primarily to implement user logic, connectivity, and I/Os, but it is also used for other purposes. For example, it is used to specify a variety of static conditions in functional blocks, such as clock management tiles (CMTs).

Sometimes an application requires a change in these conditions in the functional blocks while the block is operational. This can be accomplished by partial dynamic reconfiguration using the JTAG, ICAP, or SelectMAP ports. However, the dynamic reconfiguration port that is an integral part of each functional block simplifies this process greatly. Such configuration ports exist in CMTs, clock management, system monitor, serial transceivers, and PCIe® block (not I/Os).

Overview

This document describes the addressable, parallel write/read configuration memory that is implemented in each functional block that might require reconfiguration. This memory has the following attributes:

- It is directly accessible from the FPGA fabric. Configuration bits can be written to and/or read from depending on their function.
- Each bit of memory is initialized with the value of the corresponding configuration memory bit in the bitstream. Memory bits can also be changed later through the ICAP.
- The output of each memory bit drives the functional block logic, so the content of this memory determines the configuration of the functional block.

The address space can include status (read-only) and function enables (write-only). Read-only and write-only operations can share the same address space. Figure 5-1 shows how the configuration bits drive the logic in functional blocks directly in earlier FPGA families, and Figure 5-2 shows how the reconfiguration logic changes the flow to read or write the configuration bits.

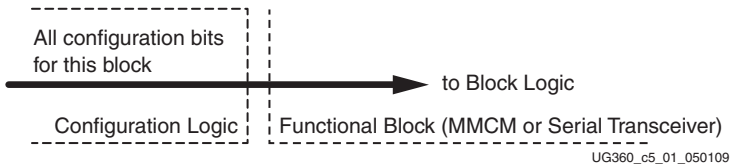


Figure 5-1: **Block Configuration Logic without Dynamic Interface**

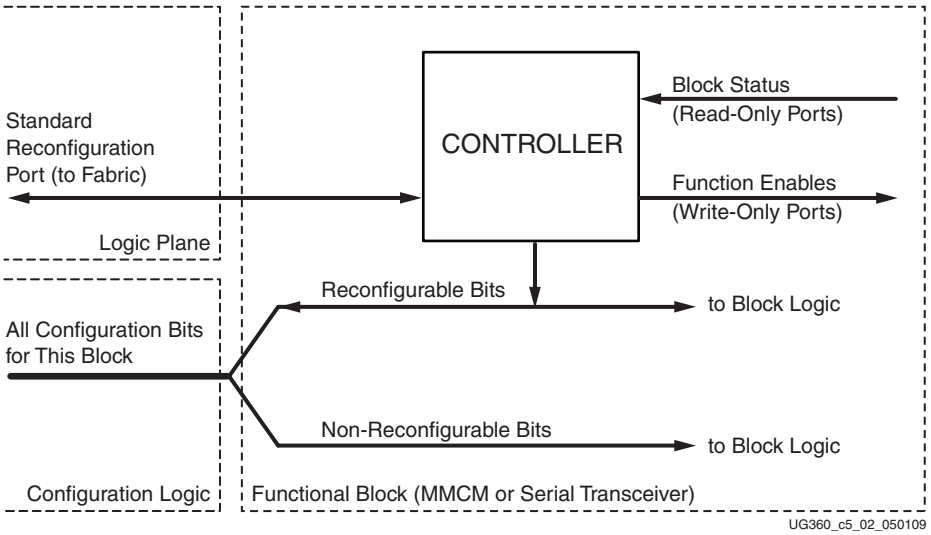


Figure 5-2: **Block Configuration Logic with Dynamic Interface**

Figure 5-3 is the same as Figure 5-2, except the port between the Logic Plane and Functional Block is expanded to show the actual signal names and directions.

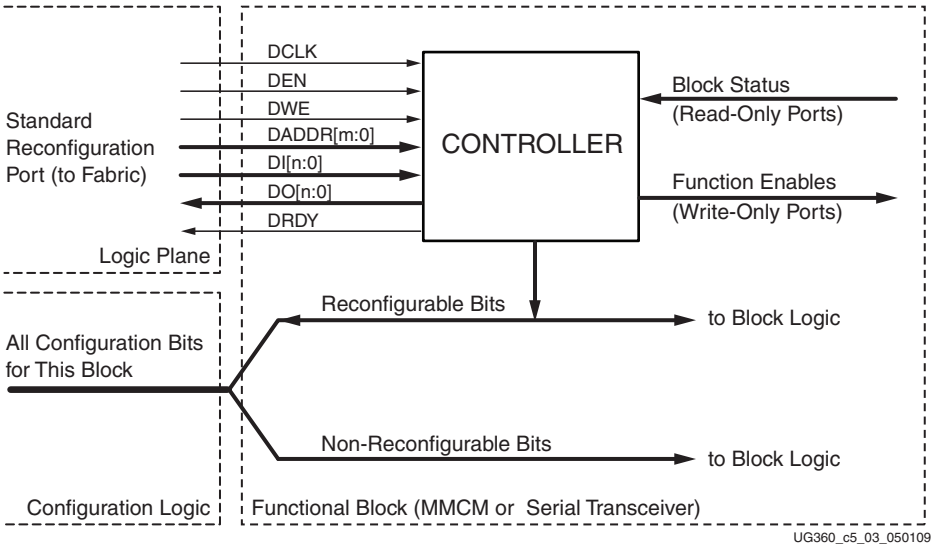


Figure 5-3: **Block Configuration Logic Expanded to Show Signal Names**

FPGA Fabric Port Definition

Table 5-1, page 82, lists each signal on the FPGA fabric port. The individual functional blocks can implement all or only a subset of these signals. The MCMM chapter in the *Virtex-6 FPGA Clocking Resources User Guide* shows the signals and functions implemented for the specific blocks. In general, the port is a synchronous parallel memory port, with separate read and write buses similar to the block RAM interface. Bus bits are numbered least-significant to most-significant, starting at 0. All signals are active High.

Synchronous timing for the port is provided by the DCLK input, and all the other input signals are registered in the functional block on the rising edge of DCLK. Input (write) data

is presented simultaneously with the write address and DWE and DEN signals prior to the next positive edge of DCLK. The port asserts DRDY for one clock cycle when it is ready to accept more data. The timing requirements relative to DCLK for all the other signals are the same. The output data is not registered in the functional blocks. Output (read) data is available after some cycles following the cycle that DEN and DADDR are asserted. The availability of output data is indicated by the assertion of DRDY.

Figure 5-4 and Figure 5-5 show the timing relationships between the port signals for write and read operations. Absolute timing parameters, such as maximum DCLK frequency, setup time, etc., are defined in the *Virtex-6 FPGA Data Sheet*.

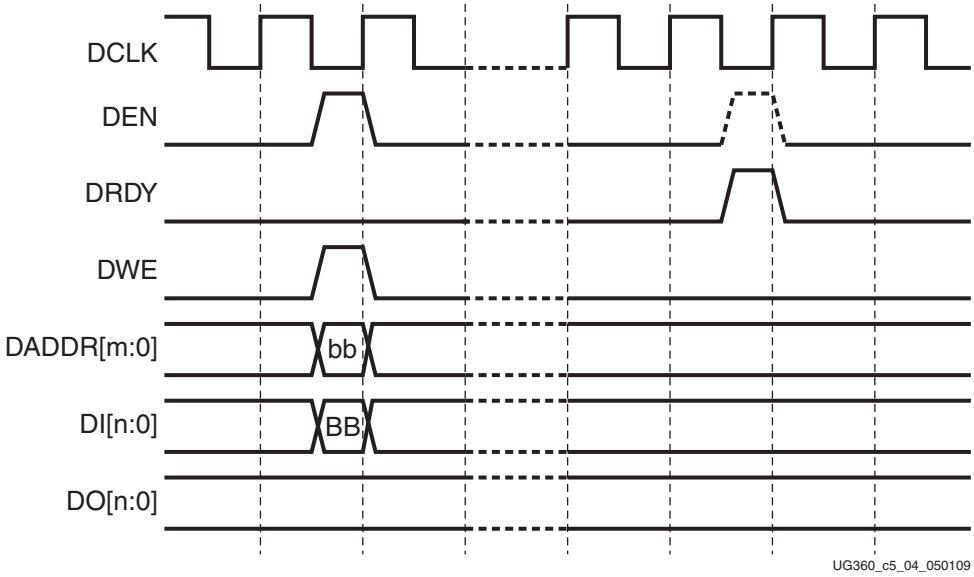


Figure 5-4: **Write Timing with Wait States**

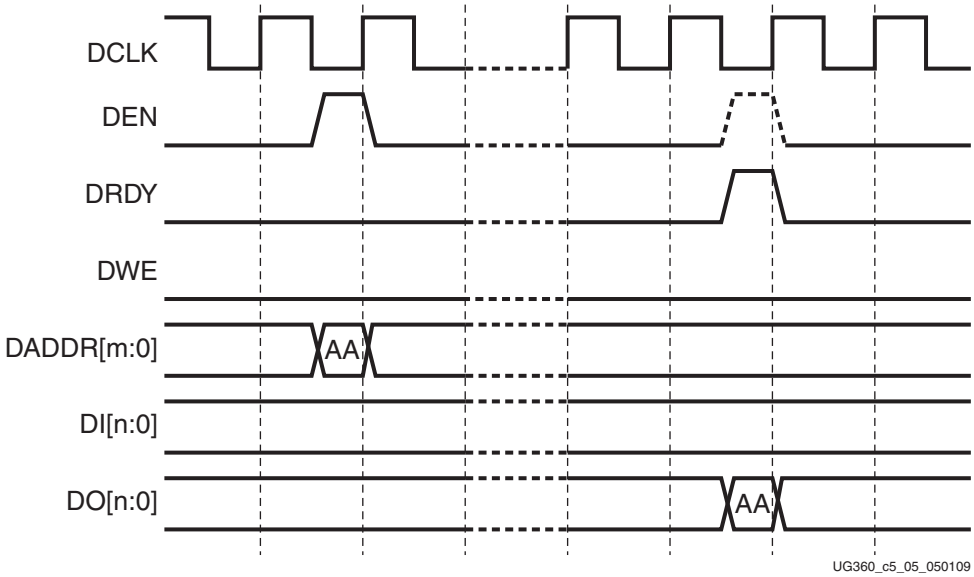


Figure 5-5: **Read Timing with Wait States**

Table 5-1: **Port Signal Definitions**

Signal Name	Direction(1)	Description
DCLK	Input	The rising edge of this signal is the timing reference for all the other port signals. The required hold time for the other input signals relative to the rising edge of DCLK is zero (maximum). Normally, DCLK is driven with a global clock buffer.
DEN	Input	This signal enables all port operations. If DWE is FALSE, it is a read operation, otherwise a write operation. For any given DCLK cycle, all other input signals are *don't care* if DEN is not active. DEN should only be pulsed for one DCLK cycle.
DWE	Input	When active, this signal enables a write operation to the port (see DEN, above). DWE should only be pulsed for one DCLK cycle.
DADDR[m:0]	Input	The value on this bus specifies the individual cell that is written or read on the next cycle of DCLK. The address is presented in the cycle that DEN is active.
DI[n:0]	Input	The value on this bus is the data that is written to the addressed cell. The data is presented in the cycle that DEN and DWE are active, and is captured in a register at the end of that cycle, but the actual write occurs at an unspecified time before DRDY is returned.
DO[n:0]	Output	If DWE was inactive when DEN was activated, the value on this bus when DRDY goes active is the data read from the addressed cell. At all other times, the value on DO[n:0] is undefined.
DRDY	Output	This signal is a response to DEN to indicate that the DRP cycle is complete and another DRP cycle can be initiated. In the case of a port read, the DO bus must be captured on the rising edge of DCLK in the cycle that DRDY is active. The earliest that DEN can go active to start the next port cycle is the same clock cycle that DRDY is active.

Notes:

1. Input denotes input (write) to the DRP.

Chapter 6

Configuration Details

Configuration Pins

Certain pins are dedicated to configuration (Table 6-1), while others are dual-purpose (Table 6-2). Dual-purpose pins serve both as configuration pins and as user I/O after configuration. Dedicated configuration pins retain their function after configuration.

Configuration constraints can be selected when generating the Virtex®-6 FPGA bitstream. Certain configuration operations can be affected by these constraints. For a description of the available constraints, see the software constraints guide.

Table 6-1: **Virtex-6 FPGA Dedicated Configuration Pins**

Pin Name	Type(1)	Description
M[2:0]	Input	Mode pins that determine configuration mode. Sampled on the rising edge of INIT_B.
CCLK	Input or Output	Configuration clock source for all configuration modes except JTAG. Refer to the Board Layout for Configuration Clock (CCLK), page 58 for details.
DIN	Input	Serial data input for serial configuration modes.
DOUT_BUSY	Output	In Serial configuration mode, the DOUT_BUSY pin acts as serial data output for daisy-chain configuration. DOUT_BUSY is also used in SelectMAP read operations; it remains Low during writes.
DONE	Bidirectional, Open-Drain, or Active	Active High signal indicating configuration is complete. 0 = FPGA not configured 1 = FPGA configured Refer to the BitGen section of UG628, *Command Line Tools User Guide* for software settings.
INIT_B	Bidirectional, Input or Output, Open-Drain	From power-on reset or PROGRAM_B reset, INIT_B is driven Low, indicating that the FPGA is initializing (clearing) its configuration memory. Before the Mode pins are sampled, INIT_B is an open-drain, active-Low input and can be held Low to delay configuration. After the Mode pins are sampled, the INIT_B output indicates if a CRC error occurred during configuration or a readback CRC error occurred after configuration (when enabled): 0 = CRC or IDCODE error (DONE is Low) or Readback CRC Error (DONE is High and Readback CRC is enabled). 1 = No CRC error, housecleaning is complete (needs an external pull-up resistor). When the SEU detection function is enabled, INIT_B is optionally driven Low when a readback CRC error is detected.

Table 6-1: **Virtex-6 FPGA Dedicated Configuration Pins *(Cont'd)***

Pin Name	Type[1]	Description
PROGRAM_B[2]	Input	Active-Low full-chip reset. This signal is edge sensitive before and during power-up and is level sensitive after power-up. PROGRAM_B cannot be used to delay configuration from power-up. See Delaying Configuration, page 100 for additional information.
HSWAPEN	Input	Active-High input used to disable weak preconfiguration I/O pull-up resistors: 0 = Weak preconfiguration I/O pull-up resistors enabled 1 = Weak preconfiguration I/O pull-up resistors disabled HSWAPEN has a weak pull-up prior to and during configuration. HSWAPEN must be connected to either disable or enable the pull-up resistors. The weak pull-up does not always provide a reliable 1.
TDI	Input	Test Data In. This pin is the serial input to all JTAG instruction and data registers. The state of the TAP controller and the current instruction determine the register that 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	Output	Test Data Out. This pin is the serial output for all JTAG instruction and data registers. The state of the TAP controller and the current instruction determine the register (instruction or data) that 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. TDO is an active driver output.
TMS	Input	Test Mode Select. 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	Input	Test Clock. This pin is the JTAG Test Clock. TCK sequences the TAP controller and the JTAG registers in Virtex-6 devices.
CSI_B	Input	Active-Low chip select to enable the SelectMAP data bus (see SelectMAP Data Loading, page 38): 0 = SelectMAP data bus enabled 1 = SelectMAP data bus disabled
RDWR_B	Input	Determines the direction of the SelectMAP data bus (see SelectMAP Data Loading, page 38): 0 = inputs (configuration data write) 1 = outputs (configuration data read) RDWR_B input can only be changed while CSI_B is deasserted, otherwise an ABORT occurs (see SelectMAP ABORT, page 169).

Notes:

1. The *Bidirectional* type describes a pin that is bidirectional under all conditions. If the pin is an input for some configuration modes or an output for others, it is listed as an *Input or Output* type.
2. Pulsing PROGRAM_B does not reset the JTAG TAP state machine.
3. All JTAG and serial dedicated configuration pins are located in a separate, dedicated bank with a dedicated V_{CC_CONFIG} supply.

Table 6-2: **Virtex-6 FPGA Dual-Purpose Configuration Pins**

Pin Name	BPI-Up/ BPI-Down	SelectMAP	SPI	I/O Bank	Description
A[25:16]	A[25:16]			34	BPI address bus output.
A[15:0]	A[15:0]	D[31:16]	RCMD [7:0]	34	BPI address bus output. SelectMAP data I/O for Slave SelectMAP mode only. SPI read command strapping inputs RCMD[7:0] (multiplexed on A[7:0]) when FS[2:0] = 001. Sampled on rising edge of INIT_B when used for SPI read command strapping.
RS[1:0](1)	RS[1:0]	RS[1:0]	RS[1:0]	24	Revision Select outputs.
FCS_B	FCS_B		FCS_B	24	BPI and SPI flash chip select output.
FOE_B	FOE_B		MOSI	24	BPI flash output enable. SPI data output from the FPGA.
FWE_B	FWE_B			24	BPI flash write enable.
CSO_B	CSO_B	CSO_B		24	Parallel daisy-chain chip select output.
D[7:0]	D[7:0]	D[7:0]	FS[2:0]	24	BPI data input, SelectMAP data in/out. SPI flash select variant. FS[2:0] are on D[2:0].
D[15:8]	D[15:8]	D[15:8]		24	BPI data input. SelectMAP data in/out.

Notes:

1. RS[1:0] are actively driven to 0 if a configuration error is detected or a watchdog timer event occurs. This can interfere with user logic when used as user I/O. When using the RS[1:0] pins, they should be restricted to be output-only in User mode. RS[1:0] are driven in SPI mode.
2. All dual-purpose pins are located in I/O banks with an associated bank supply.

FPGA I/O Pin Settings During Configuration

All of the dedicated FPGA configuration pins have pull-up resistors during configuration. All dual-mode I/O pins have optional pull-up resistors that can be enabled during the configuration process. During configuration, a single control line determines whether the pull-up resistors are enabled or disabled. The pin name is HSWAPEN (see Table 6-3). Post configuration depends on the user BitGen selection. By default, the unused I/Os are pull-down, and the dedicated configuration pins are pull-up.

Table 6-3: **Virtex-6 FPGA Configuration Pin Termination**

Pin	Pre-Configuration		Post-Configuration(2)
	HSWAPEN = 0 (enabled)	HSWAPEN = 1 (disabled)	
CCLK	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g CclkPin**
DOUT_BUSY	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g BusyPin**
HSWAPEN	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g HswapenPin**
PROGRAM_B	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g ProgPin**
DONE	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g DonePin -g DriveDone**
INIT_B	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g InitPin**
TDI	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g TdiPin**

Table 6-3: **Virtex-6 FPGA Configuration Pin Termination *(Cont'd)***

Pin	Pre-Configuration		Post-Configuration[2]
	HSWAPEN = 0 (enabled)	HSWAPEN = 1 (disabled)	
TMS	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g TmsPin**
TCK	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g TckPin**
TDO	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g TdoPin**
M0	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g M0Pin**
M1	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g M1Pin**
M2	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g M2Pin**
RDWR_B	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g RdWrPin**
DIN	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g DinPin**
CSI_B	Pull-up to VCCO_0	Pull-up to VCCO_0	BitGen **-g CsPin**
CSO_B	Pull-up to VCCO_24	No termination	User I/O
FCS_B	Pull-up to VCCO_24	No termination	User I/O
FOE_B/MOSI	Pull-up to VCCO_24	No termination	User I/O
RS[1:0]	Pull-up to VCCO_24	No termination	User I/O
FWE_B	Pull-up to VCCO_24	No termination	User I/O
CSO_B	Pull-up to VCCO_24	No termination	User I/O
FS[2:0]	Pull-up to VCCO_24	No termination	User I/O
MOSI	Pull-up to VCCO_24	No termination	User I/O
D[15:0]	Pull-up to VCCO_24	No termination	User I/O
D[31:16]	Pull-up to VCCO_34	No termination	User I/O
A[25:0][1]	Pull-up to VCCO_34	No termination	User I/O
RCMD[7:0]	Pull-up to VCCO_34	No termination	User I/O

Notes:

1. Setting the BitGen options configures the termination on the respective pin. The dedicated pins are set to pull-up by default. User I/O pins when unused in the design are set to pull-down by default.
2. Refer to the BitGen section of UG628, *Command Line Tools User Guide*, for software settings.

Configuration Data File Formats

Xilinx® design tools can generate configuration data files in a number of different formats, as described in Table 6-4. BitGen converts the post-PAR NCD file into a configuration file or a bitstream. PROMGen, the PROM file generator, converts one or more bitstream files into a PROM file. PROM files can be generated in a number of different file formats and does not need to be used with a PROM. They can be stored anywhere and delivered by any means.

Table 6-4: **Xilinx Configuration File Formats**

File Extension	Bit Swapping(1)	Xilinx Software Tool(2)	Description
BIT	Not Bit Swapped	BitGen (generated by default)	Binary configuration data file containing header information that does not need to be downloaded to the FPGA. Used to program devices from iMPACT with a programming cable.
RBT	Not Bit Swapped	BitGen (generated if **-b** option is set)	ASCII equivalent of the BIT file containing a text header and ASCII 1s and 0s. (Eight bits per configuration bit.)
BIN	Not Bit Swapped	BitGen (generated if **-g binary:yes** option is set) or PROMGen	Binary configuration data file with no header information. Similar to the BIT file. Can be used for custom configuration solutions (for example, microprocessors), or in some cases to program third-party PROMs.
MCS EXO	Bit Swapped(3)	PROMGen or iMPACT	ASCII PROM file formats containing address and checksum information in addition to configuration data. Used mainly for device programmers and iMPACT.
HEX	Determined by User	PROMGen or iMPACT	ASCII PROM file format containing only configuration data. Used mainly in custom configuration solutions.

Notes:

1. Bit swapping is discussed in the Bit Swapping section.
2. For complete BitGen and PROMGen syntax, refer to UG628, *Command Line Tools User Guide*.
3. PROM files are generally bit-swapped except in SPI Configuration mode. The PROMGen **-spi** option is used for SPI flash and creates a file that is not bit swapped.

Bitstream Overview

The Virtex-6 FPGA bitstream contains commands to the FPGA configuration logic as well as configuration data. Table 6-5 gives a typical bitstream length for each of the Virtex-6 devices.

Table 6-5: **Virtex-6 FPGA Bitstream Length**

Device(1)	Total Number of Configuration Bits(2)(3)
XC6VHX250T	79,862,624
XC6VHX255T	79,862,624
XC6VHX380T	119,784,608
XC6VHX565T	160,655,264
XC6VLX75T	26,239,328
XC6VLX130T	43,719,776
XC6VLX195T	61,552,736
XC6VLX240T	73,859,552

Table 6-5: **Virtex-6 FPGA Bitstream Length *(Cont'd)***

Device[1]	Total Number of Configuration Bits[2][3]
XC6VLX365T	96,067,808
XC6VLX550T	144,092,384
XC6VLX760	184,823,072
XC6VSX315T	104,465,888
XC6VSX475T	156,689,504
XQ6VLX130T	43,719,776
XQ6VLX240T	73,859,552
XQ6VLX550T	144,092,384
XQ6VSX315T	104,465,888
XQ6VSX475T	156,689,504

Notes:

1. Devices beginning with XQ are defense-grade FPGAs.
2. Most Virtex-6 devices can be configured by a single 128 Mb Platform Flash XL device. For densities greater than 128 Mb, it is recommended to use BPI (parallel NOR) flash devices.
3. The bitstream length represents the typical cases. Certain BitGen options can vary the bitstream length, such as **`Compress`**.

A Virtex-6 FPGA bitstream consists of three sections:

- Bus Width Auto Detection
- Sync Word
- FPGA configuration

Bus Width Auto Detection

Bus width auto detection pattern is inserted at the beginning of every bitstream. It is used in parallel configuration modes to automatically detect configuration bus width. Because it appears before the Sync word, serial configuration modes ignore it.

For parallel configuration modes, the bus width is auto-detected by the configuration logic. A bus width detection pattern is put in the front of the bitstream. The configuration logic only checks the low eight bits of the parallel bus. Depending on the byte sequence received, the configuration logic can automatically switch to the appropriate external bus width. Table 6-6 shows an example bitstream with an inserted bus width detection pattern. When observing the pattern on the FPGA data pin, the bits are bit swapped, as described in Parallel Bus Bit Order.

The bitstream data in Table 6-6 shows the 32-bit configuration word for an unswapped bitstream. For a swapped bitstream format, the LSB and MSB for the individual bytes are swapped. For example, the Sync word at the FPGA pins in the D[31:0] bit order would be `0x5599AA66`. For swapped and unswapped formats, see Configuration Data File Formats.

Table 6-6: **Bus Width Detection Pattern**

D[24:31]	D[16:23]	D[8:15]	D[0:7]	Comments
`0xFF`	`0xFF`	`0xFF`	`0xFF`	
`0x00`	`0x00`	`0x00`	`0xBB`	Bus Width Pattern

Table 6-6: **Bus Width Detection Pattern *(Cont'd)***

D[24:31]	D[16:23]	D[8:15]	D[0:7]	Comments
0x11	0x22	0x00	0x44	Bus Width Pattern
0xFF	0xFF	0xFF	0xFF	
0xFF	0xFF	0xFF	0xFF	
0xAA	0x99	0x55	0x66	Sync Word
…	…	…	…	…

Bus width auto detection is transparent to most users, because all configuration bitstreams (BIT or RBT files) generated by the Xilinx ISE® Bitstream Generator (BitGen) software include the Bus Width Auto Detection pattern. These patterns are ignored by the configuration logic if the Mode pins are set to Master Serial, Slave Serial, JTAG, or SPI mode.

For the x8 bus, the configuration bus width detection logic first finds 0xBB on the D[0:7] pins, followed by 0x11. For the x16 bus, the configuration bus width detection logic first finds 0xBB on D[0:7] followed by 0x22. For the x32 bus, the configuration bus width detection logic first finds 0xBB, on D[0:7], followed by 0x44.

If the immediate byte after 0xBB is not 0x11, 0x22, or 0x44, the bus width state machine is reset to search for the next 0xBB until a valid sequence is found. Then it switches to the appropriate external bus width and starts looking for the Sync word. When the bus width is detected, the SelectMAP interface is locked to that bus width until a power cycle, PROGRAM_B pulse, JPROGRAM reset, or IPROG reset is issued.

Sync Word

A special Sync word is used to allow configuration logic to align at a 32-bit word boundary. No packet is processed by the FPGA until the Sync word is found. The bus width must be detected successfully for parallel configuration modes before the Sync word can be detected. Table 6-7 shows the Sync word in an unswapped bitstream format.

Table 6-7: **Sync Word**

31:24	23:16	15:8	7:0
0xAA	0x99	0x55	0x66

Generating PROM Files

PROM files are generated from bitstream files with the PROMGen utility. Users can access PROMGen directly from the command line or indirectly through the iMPACT File Generation Mode. For PROMGen syntax, refer to UG628, *Command Line Tools User Guide*. For information on iMPACT, refer to the ISE Software Documentation). PROM files serve to reformat bitstream files for PROM programming and combine bitstream files for serial daisy chains (see PROM Files for Serial Daisy Chains).

PROM Files for Serial Daisy Chains

Configuration data for serial daisy chains requires special formatting because separate BIT files cannot simply be concatenated together to program the daisy chain. The special formatting is performed by PROMGen (or iMPACT) when generating a PROM file from

multiple bitstreams. To generate the PROM file, specify multiple bitstreams using the PROMGen **-n**, **-u**, and **-d** options or the iMPACT File Generation Wizard. Refer to software documentation for details.

PROMGen reformats the configuration bitstreams by nesting downstream configuration data into configuration packets for upstream devices. Attempting to program the chain by sending multiple bitstreams to the first device causes the first device to configure and then ignore the subsequent data.

PROM Files for SelectMAP Configuration

The MCS file format is most commonly used to program Xilinx configuration PROMs that in turn program a single FPGA in SelectMAP mode. For custom configuration solutions, the BIN and HEX files are the easiest PROM file formats to use due to their *raw* data format. In some cases, additional formatting is required; refer to XAPP502, *Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode* for details.

If multiple configuration bitstreams for a SelectMAP configuration reside on a single memory device, the bitstreams must not be combined into a serial daisy chain PROM file. Instead, the target memory device should be programmed with multiple BIN or HEX files. If a single PROM file with multiple, separate data streams is needed, one can be generated in iMPACT by targeting a *Parallel PROM*, then selecting the appropriate number of data streams. This can also be accomplished through the PROMGen command line. Refer to PROMGen software documentation for details.

For Platform Flash XL-based SelectMAP configuration, use the iMPACT software to generate an MCS PROM file. Select the Xilinx XCF128X device as the target PROM device type for creating the file. See UG438, *Platform Flash XL User Guide*, for PROM file generation instructions.

PROM Files for SPI/BPI Configuration

The **-d** and **-u** options in PROMGen or the iMPACT File Generation Wizard are used to create PROM files for third-party flash devices. The output format supported by your third-party programmer should be chosen. Some BPI devices require endian-swapping to be enabled when programming the PROM file. Refer to the flash vendor's documentation.

Bit Swapping

Bit swapping is the swapping of the bits within a byte. The MCS and EXO PROM file formats are always bit swapped unless the PROMGen **-spi** option for the SPI Configuration mode is used. The HEX file format can be bit swapped or not bit swapped, depending on user options. The bitstream files (BIT, RBT, BIN) are never bit swapped.

The HEX file format contains only configuration data. The other PROM file formats include address and checksum information that should not be sent to the FPGA. The address and checksum information is used by some third-party device programmers, but is not programmed into the PROM.

Figure 6-1 shows how two bytes of data (0xABCD) are bit swapped.

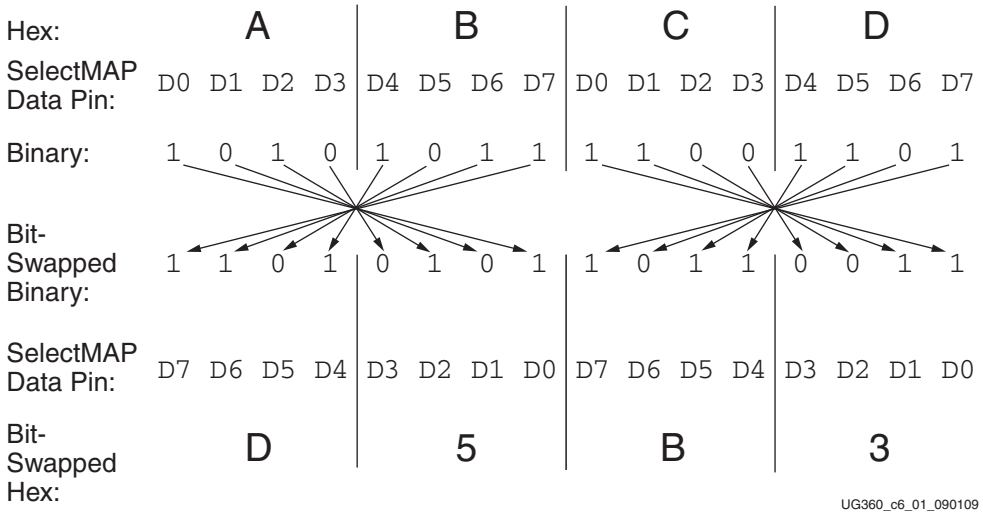


Figure 6-1: **Bit Swapping Example**

The MSB of each byte goes to the D0 pin regardless of the orientation of the data:

- In the bit-swapped version of the data, the bit that goes to D0 is the right-most bit
- In the non-bit-swapped data, the bit that goes to D0 is the left-most bit.

Whether or not data must be bit swapped is entirely application-dependent. Bit swapping is applicable for Serial, SelectMAP, or BPI PROM files.

Parallel Bus Bit Order

Traditionally, in SelectMAP x8 mode, configuration data is loaded one byte per CCLK, with the most-significant bit (MSB) of each byte presented to the D0 pin. Although this convention (D0 = MSB, D7 = LSB) differs from many other devices, it is consistent across all Xilinx FPGAs. The bit-swap rule also applies to Virtex-6 FPGA BPI-Up and BPI-Down x8 modes (see Bit Swapping, page 90).

In Virtex-6 devices, the bit-swap rule is extended to x16 and x32 bus widths, i.e., the data is bit swapped within each byte. The bit order in Virtex-6 FPGAs is the same as in Virtex-5 FPGAs. The SelectMAP x32 mode in Virtex-4 FPGAs does not bit swap.

Table 6-8 and Table 6-9 show examples of a sync word inside a bitstream. These examples illustrate what is expected at the FPGA data pins when using parallel configuration modes, such as Slave SelectMAP, Master SelectMAP, BPI-Up, and BPI-Down modes.

Table 6-8: **Sync Word Bit Swap Example**

Sync Word	[31:24](1)	[23:16]	[15:8]	[7:0]
Bitstream Format	0xAA	0x99	0x55	0x66
Bit Swapped	0x55	0x99	0xAA	0x66

Notes:

1. [31:24] changes from 0xAA to 0x55 after bit swapping.

Table 6-9: **Sync Word Data Sequence Example for x8, x16, and x32 Modes**

CCLK Cycle	1	2	3	4
D[7:0] pins for x8	0x55	0x99	0xAA	0x66
D[15:0] pins for x16	0x5599	0xAA66		
D[31:0] pins for x32	0x5599AA66			

Configuration Sequence

While each of the configuration interfaces is different, the basic steps for configuring a Virtex-6 device are the same for all modes. Figure 6-2 shows the Virtex-6 FPGA configuration process. The following subsections describe each step in detail, where the current step is highlighted in gray at the beginning of each subsection.

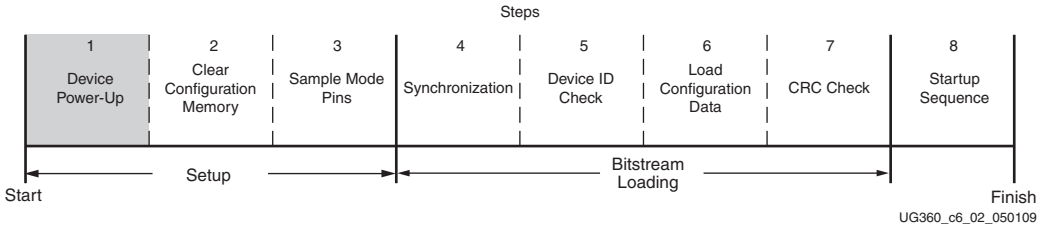


Figure 6-2: **Virtex-6 FPGA Configuration Process**

The Virtex-6 device is initialized and the configuration mode is determined by sampling the mode pins in three setup steps.

Setup (Steps 1-3)

The setup process is similar for all configuration modes (see Figure 6-3).

The setup steps are critical for proper device configuration. The steps include Device Power-Up, Clear Configuration Memory, and Sample Mode Pins.

Device Power-Up (Step 1)

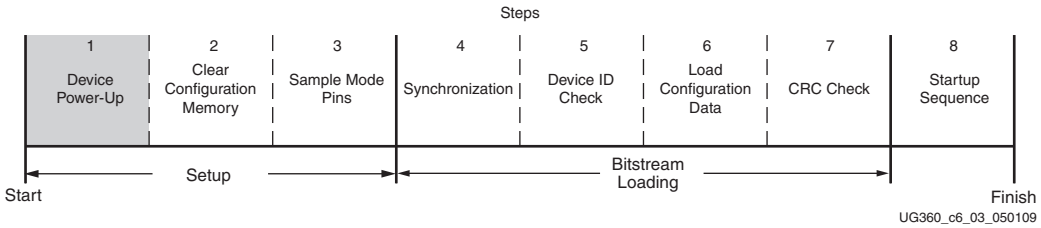


Figure 6-3: **Device Power-Up (Step 1)**

For configuration, Virtex-6 devices require power on the V_{CC_CONFIG} (V_{CCO_0}), V_{CCAUX}, and V_{CCINT} pins. Power sequencing requirements are described in the *Virtex-6 FPGA Data Sheet*.

All JTAG and serial configuration pins are located in a separate, dedicated bank with a dedicated V_{CC_CONFIG} supply (V_{CC_CONFIG} = V_{CCO_0}). The dual-mode pins are located in Banks 24 and 34. All dedicated input pins operate at V_{CC_CONFIG} LVCMOS level. All active

dedicated output pins operate at the V_{CC_CONFIG} voltage level with the output standard set to LVCMOS_12F.

For all modes that use dual-mode I/O, the associated V_{CCO_X} must be connected to the appropriate voltage to match the I/O standard of the configuration device. The pins are also LVCMOS_12F during configuration.

For power-up, the V_{CCINT} power pins must be supplied with 1.0V or 0.9V (for -1L) sources. None of the I/O voltage supplies except V_{CCO_0} needs to be powered for Virtex-6 FPGA configuration in JTAG or serial modes when RS[1:0] is not used. When configuration modes are selected that use the dual-mode pins (i.e., Master BPI, SelectMAP), V_{CCO_24}, V_{CCO_34}, or both must be also be supplied. Refer to Table 6-2, page 85 for the dual-mode pin supply requirements for the different configuration modes. Table 6-10 shows the power supplies required for configuration. Table 6-11 shows the timing for power-up. Refer to DS152, *Virtex-6 FPGA Data Sheet* for voltage ratings.

Table 6-10: **Power Supplies Required for Configuration**

Pin Name	Description
V_{CCINT}	Internal core voltage.
V_{BATT}(1)	Encryption Key battery supply.
V_{CCO_0}	Configuration bank supply voltage.
V_{CCAUX}	Auxiliary power input for configuration logic and other FPGA functions.
V_{CCO_24} V_{CCO_34}	Dual-mode configuration pin output supply voltage. Standard I/O voltage levels supported for configuration are 1.8V and 2.5V.

Notes:

1. V_{BATT} is required only when using bitstream encryption.

Table 6-11: **Power-Up Timing**

Description	Symbol
Program Latency	T_{PL}
Power-on Reset (POR)	T_{POR}
CCLK Output Delay	T_{ICCK}
Program Pulse Width	$T_{PROGRAM}$

Notes:

1. See DS152, *Virtex-6 FPGA Data Sheet* for power-up timing characteristics.

Figure 6-4 shows the power-up waveforms.

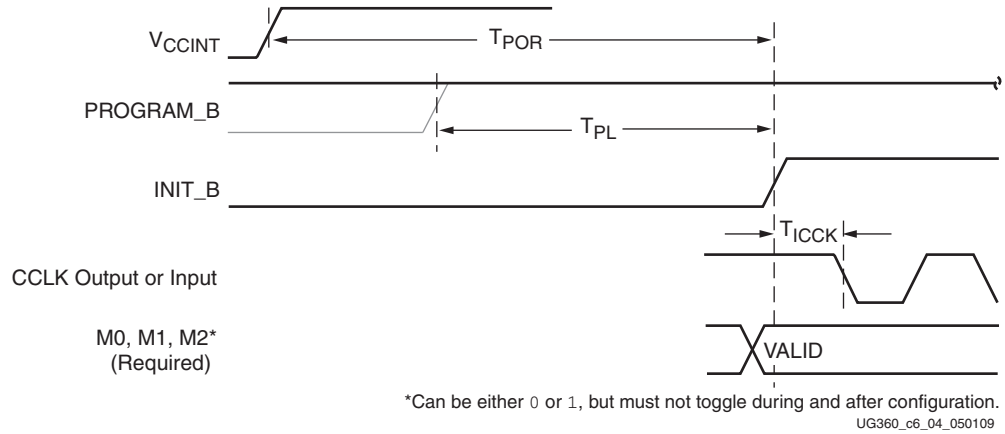


Figure 6-4: **Device Power-Up Timing**

V_{CCINT} should rise monotonically within the specified ramp rate. If this is not possible, delay configuration by holding the INIT_B pin (see Delaying Configuration) while the system power reaches the minimum recommended operating voltages for V_{CC_CONFIG}, V_{CCAUX}, V_{CCINT}. V_{CCO_24}, and V_{CCO_34} if they are used by the target configuration mode.

The configuration logic power input (V_{CCO_0}) and the auxiliary voltage input (V_{CCAUX}) are used as a logic input to the Power-On-Reset (POR) circuitry.

Clear Configuration Memory (Step 2, Initialization)

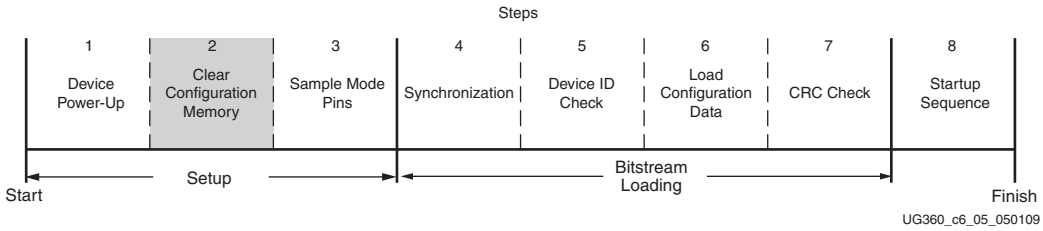


Figure 6-5: **Initialization (Step 2)**

Configuration memory is cleared sequentially any time the device is powered up, after the PROGRAM_B pin is pulsed Low, after the JTAG JPROGRAM instruction or the IPROG command is used, or during a fallback retry configuration sequence. During this time, I/Os are placed in a High-Z state except for the dedicated Configuration and JTAG pins. INIT_B is internally driven Low during initialization, then released after T_{POR} (Figure 6-4) for the power-up case, and T_{PL} for other cases. If the INIT_B pin is held Low externally, the device waits at this point in the initialization process until the pin is released.

The minimum Low pulse time for PROGRAM_B is defined by the $T_{PROGRAM}$ timing parameter. The PROGRAM_B pin can be held active (Low) for as long as necessary, and the device clears the configuration memory twice after PROGRAM_B is released.

Sample Mode Pins (Step 3)

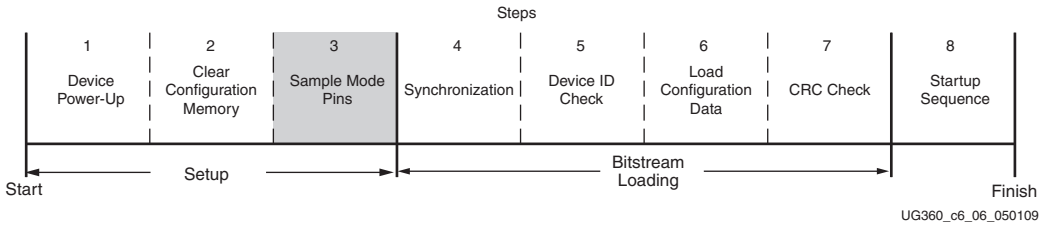


Figure 6-6: **Sample Mode Pins (Step 3)**

When the INIT_B pin transitions to High, the device samples the M[2:0], FS[2:0], and RCMD[7:0] pins and begins driving CCLK if in the Master modes. FS[2:0] and RCMD[7:0] are only used in SPI mode. At this point, the device begins sampling the configuration data input pins on the rising edge of the configuration clock.

Bitstream Loading (Steps 4-7)

The bitstream loading process is similar for all configuration modes; the primary difference between modes is the interface to the configuration logic. Details on the different configuration interfaces are provided in Chapter 2, Configuration Interfaces.

Synchronization (Step 4)

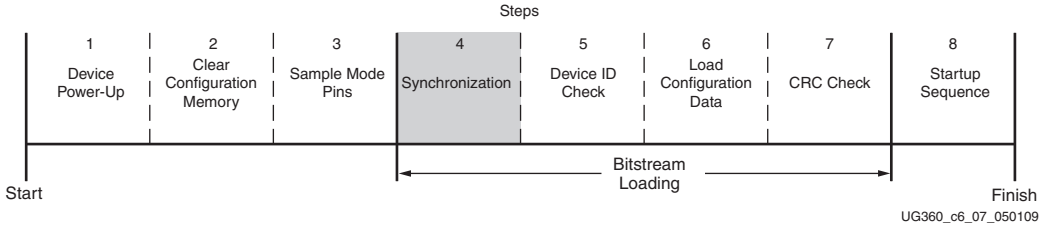


Figure 6-7: **Synchronization (Step 4)**

For BPI-Up, BPI-Down, Slave SelectMAP, and Master SelectMAP modes, the bus width must be first detected (refer to Bus Width Auto Detection). The bus width detection pattern is ignored by Slave Serial, Master Serial, SPI, and JTAG modes. Then a special 32-bit synchronization word (`0xAA995566`) must be sent to the configuration logic. The synchronization word alerts the device to upcoming configuration data and aligns the configuration data with the internal configuration logic. Any data on the configuration input pins prior to synchronization is ignored, except the "Bus Width Auto Detection" sequence.

Synchronization is transparent to most users because all configuration bitstreams (`BIT` files) generated by the BitGen software include both the bus width detection pattern and the synchronization word. Table 6-12 shows signals relating to synchronization.

Table 6-12: **Signals Relating to Synchronization**

Signal Name	Type	Access	Description
DALIGN	Status	Only available through the SelectMAP interface during an ABORT sequence. (See Configuration Abort Sequence Description, page 169.)	Indicates whether the device is synchronized.
IWIDTH	Status	Internal signal. Accessed only through the Virtex-6 FPGA Status register.(1) The Status register BUS_WIDTH bits indicate the detected bus width.	Indicates the detected bus width: 00 = x1 01 = x8 10 = x16 11 = x32 IWIDTH reflects the external port bus width when read back from the JTAG interface. IWIDTH reflects the SelectMAP bus width when read back from the SelectMAP interface. IWIDTH reflects the ICAP bus width when read back from the ICAP interface.

Notes:

1. Information on the Virtex-6 FPGA status register is available in Table 6-34. Information on accessing the device status register via JTAG or SelectMAP is available in Chapter 7, Readback and Configuration Verification.

Check Device ID (Step 5)

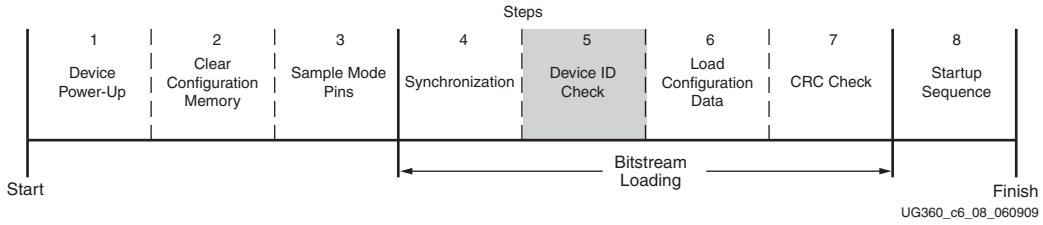


Figure 6-8: **Check Device ID (Step 5)**

After the device is synchronized, a device ID check must pass before the configuration data frames can be loaded. This prevents a configuration with a bitstream that is formatted for a different device. For example, the device ID check should prevent an XC6VLX130T from being configured with an XC6VLX75T bitstream.

If an ID error occurs during configuration, the device attempts to do a fallback reconfiguration.

The device ID check is built into the bitstream, making this step transparent to most designers. Table 6-13 shows the Virtex-6 device ID codes, and Table 6-14 shows the signals relating to the device ID check. The device ID check is performed through commands in the bitstream to the configuration logic, not through the JTAG IDCODE register in this case.

The Virtex-6 FPGA JTAG ID Code register has the following format:

```
vvvv:fffffff:aaaaaaaaa:ccccccccccc1
```

where:

v = version

f = 7-bit family code

a = 9-bit array code (includes 4-bit sub-family and 5-bit device code)

c = company code

Table 6-13: **Virtex-6 FPGA Device ID Codes**

Device[1]	IDCODE (Hex)[2]
XC6VHX250T	0x042A2093
XC6VHX255T	0x042A4093
XC6VHX380T	0x042A8093
XC6VHX565T	0x042AC093
XC6VLX75T	0x04244093
XC6VLX130T	0x0424A093
XC6VLX195T	0x0424C093
XC6VLX240T	0x04250093
XC6VLX365T	0x04252093
XC6VLX550T	0x04256093
XC6VLX760	0x0423A093
XC6VSX315T	0x04286093
XC6VSX475T	0x04288093
XQ6VLX130T	0x0424A093
XQ6VLX240T	0x04250093
XQ6VLX550T	0x04256093
XQ6VSX315T	0x04286093
XQ6VSX475T	0x04288093

Notes:

1. Devices beginning with XQ are defense-grade FPGAs.
2. The Virtex-6 FPGA ID Code version field reflects the silicon version and can vary.

Table 6-14: **Signals Relating to the Device ID Check**

Signal Name	Type	Access[1]	Description
ID_ERROR	Status	Internal signal. Accessed only through the Virtex-6 FPGA Status register.	Indicates a mismatch between the device ID specified in the bitstream and the actual device ID.

Notes:

1. Information on the Virtex-6 FPGA status register is available in Table 6-34. Information on accessing the device status register via JTAG or SelectMAP is available in Chapter 7, Readback and Configuration Verification.

Load Configuration Data Frames (Step 6)

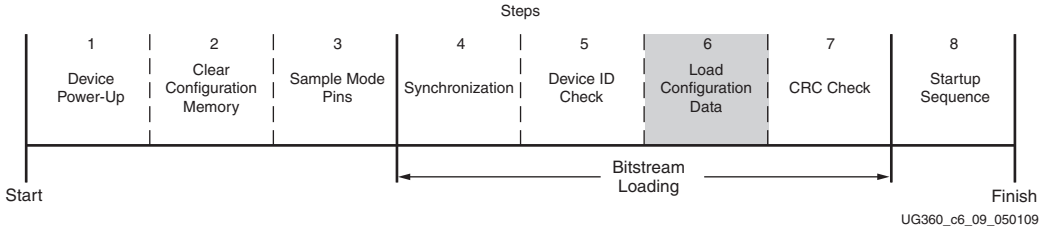


Figure 6-9: **Load Configuration Data Frames (Step 6)**

After the synchronization word is loaded and the device ID has been checked, the configuration data frames are loaded. This process is transparent to most users.

Cyclic Redundancy Check (Step 7)

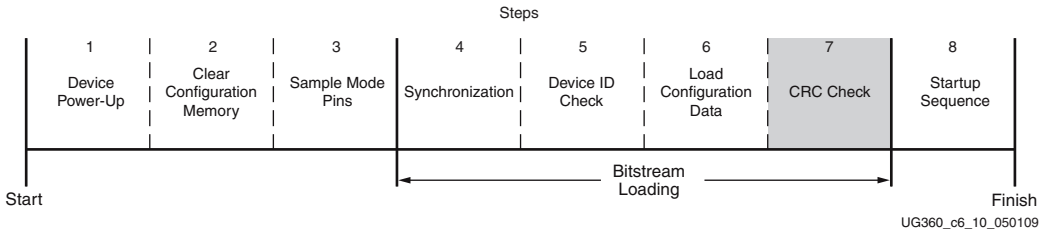


Figure 6-10: **Cyclic Redundancy Check (Step 7)**

As the configuration data frames are loaded, the device calculates a Cyclic Redundancy Check (CRC) value from the configuration data packets. After the configuration data frames are loaded, the configuration bitstream can issue a *Check CRC* instruction to the device, followed by an expected CRC value. If the CRC value calculated by the device does not match the expected CRC value in the bitstream, the device pulls INIT_B Low and aborts configuration. The CRC check is included in the configuration bitstream by default, although the designer can disable it if desired. (Refer to the "BitGen" section of UG628, *Command Line Tools User Guide*.) If the CRC check is disabled, there is a risk of loading incorrect configuration data frames, causing incorrect design behavior or damage to the device.

If a CRC error occurs during configuration from a mode where the FPGA is the configuration master, the device can attempt to do a fallback reconfiguration. In BPI-Up, BPI-Down, and SPI modes, if fallback reconfiguration fails again, the BPI/SPI interface can only be resynchronized by pulsing the PROGRAM_B pin and restarting the configuration process from the beginning. The JTAG interface is still responsive and the device is still alive, only the BPI/SPI interface is inoperable. In SelectMAP modes, either the PROGRAM_B pin can be pulsed Low or an ABORT sequence can be initiated (see SelectMAP Configuration Interface in Chapter 2).

Virtex-6 devices use a 32-bit CRC check. The CRC check is designed to catch errors in transmitting the configuration bitstream. There is a scenario where errors in transmitting the configuration bitstream can be missed by the CRC check: certain clocking errors, such as double-clocking, can cause loss of synchronization between the 32-bit bitstream packets and the configuration logic. Once synchronization is lost, any subsequent commands are not understood, including the command to check the CRC. In this situation, configuration fails with DONE Low and INIT_B High because the CRC was ignored. In BPI Modes, the

address counter eventually overflows or underflows to cause wraparound, which triggers fallback reconfiguration.

Startup (Step 8)

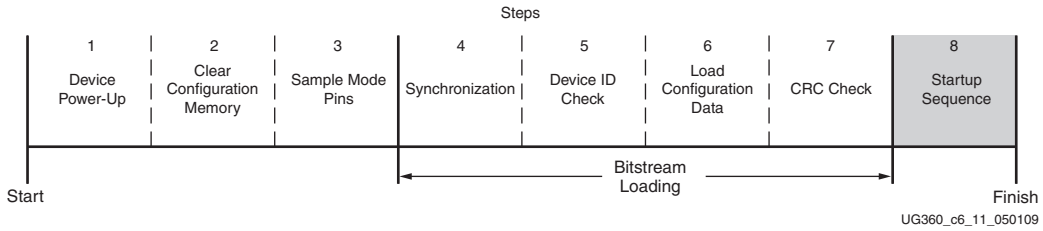


Figure 6-11: **Startup Sequence (Step 8)**

After the configuration frames are loaded, the bitstream instructs the device to enter the startup sequence. The startup sequence is controlled by an 8-phase (phases 0–7) sequential state machine. The startup sequencer performs the tasks outlined in Table 6-15.

Table 6-15: **User-Selectable Cycle of Startup Events**

Phase	Event
1–6	Wait for DCI to Match (optional)
1–6	Assert Global Write Enable (GWE), allowing RAMs and flip-flops to change state
1–6	Negate Global 3-State (GTS), activating I/O
1–6	Release DONE pin
7	Assert End Of Startup (EOS)

The specific order of startup events (except for EOS assertion) is user-programmable through BitGen options (refer to UG628, *Command Line Tools User Guide*). Table 6-15 shows the general sequence of events, although the specific phase for each of these startup events is user-programmable (EOS is always asserted in the last phase). Refer to Chapter 2, Configuration Interfaces for important startup option guidelines. By default, startup events occur as shown in Table 6-16.

Table 6-16: **Default BitGen Sequence of Startup Events**

Phase	Event
5	Negate GTS, activating I/O
4	Release DONE pin
6	Assert GWE, allowing RAMs and flip-flops to change state
7	Assert EOS

The startup sequence can be forced to wait for DCI to match with the appropriate BitGen options. These options are typically set to prevent DONE, GTS, and GWE from being asserted (preventing device operation) before the DCI has matched.

The DONE signal is released by the startup sequencer on the cycle indicated by the user, but the startup sequencer does not proceed until the DONE pin actually sees a logic High. The DONE pin is an open-drain bidirectional signal by default. By releasing the DONE pin, the device simply stops driving a logic Low and the pin goes into a High-Z state. An external pull-up resistor is needed for the DONE pin to reach a logic High in this case. Table 6-17 shows signals relating to the startup sequencer. Figure 6-12 shows the

waveforms relating to the startup sequencer.

Table 6-17: **Signals Relating to Startup Sequencer**

Signal Name	Type	Access[1]	Description
DONE	Bidirectional[2]	DONE pin or Virtex-6 FPGA Status Register	Indicates configuration is complete. Can be held Low externally to synchronize startup with other FPGAs.
Release_DONE	Status	Virtex-6 FPGA Status Register	Indicates whether the device has stopped driving the DONE pin Low. If the pin is held Low externally, Release_DONE can differ from the actual value on the DONE pin.
GWE[3]			Global Write Enable (GWE). When asserted, GWE enables the CLB and the IOB flip-flops as well as other synchronous elements on the FPGA.
GTS			Global 3-State (GTS). When asserted, GTS disables all the I/O drivers except for the configuration pins.
EOS			End of Startup (EOS). EOS indicates the absolute end of the configuration and startup process.
DCI_MATCH			DCI_MATCH indicates when all the Digitally Controlled Impedance (DCI) controllers have matched their internal resistor to the external reference resistor.
MMCM_LOCK			MMCM_LOCK indicates when all the clock outputs are valid.

Notes:

1. Information on the Virtex-6 FPGA status register is available in Table 6-34. Information on accessing the device status register via JTAG or SelectMAP is available in Chapter 7, Readback and Configuration Verification.
2. Open-drain output by default; optional driver enabled using the BitGen **`DriveDone`** option.
3. GWE is asserted synchronously to the configuration clock (CCLK) and has a significant skew across the part. Therefore, sequential elements might not be released synchronously to the user's system clock, and timing violations can occur during startup. It is recommended to reset the design after startup and/or apply some other synchronization technique.

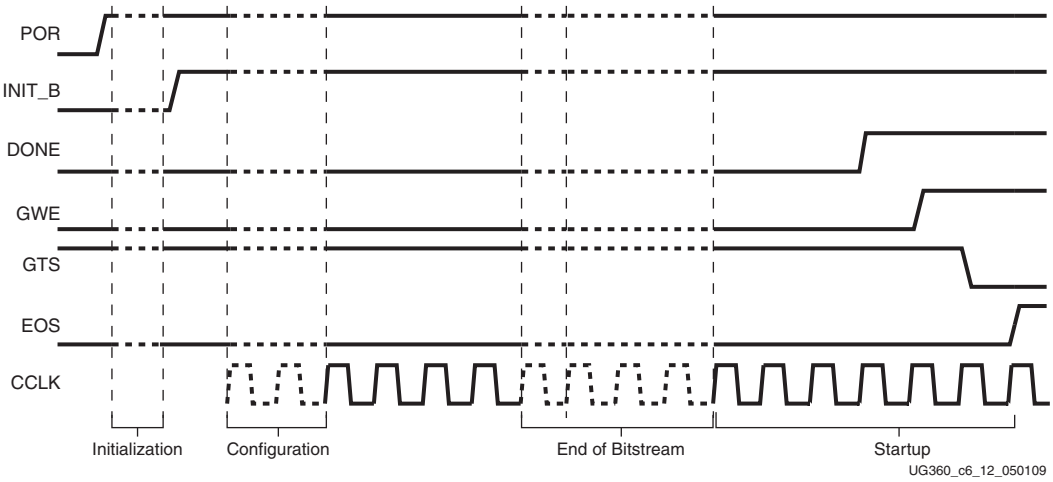


Figure 6-12: **Configuration Signal Sequencing (Default Startup Settings)**

Delaying Configuration

To delay configuration for Virtex-6 devices, the INIT_B pin is held Low during initialization (Figure 6-4). When INIT_B has gone High, configuration cannot be delayed

by subsequently pulling INIT_B Low. The signals related to initialization and delaying configuration are listed in Table 6-18.

Table 6-18: **Signals Relating to Initialization and Delaying Configuration**

Signal Name	Type	Access[1]	Description
INIT_B	Input, Output, or Open Drain	Externally accessible via the INIT_B pin	From power-on reset or PROGRAM_B reset, INIT_B is driven Low, indicating that the FPGA is initializing (clearing) its configuration memory. Before the Mode pins are sampled, INIT_B is an input that can be held Low to delay configuration. After the Mode pins are sampled, INIT_B is an open-drain, active-Low output that indicates if a CRC error occurred during configuration or a readback CRC error occurred after configuration (when enabled): 0 = CRC or IDCODE error (DONE is Low) or Readback CRC Error (DONE is High and Readback CRC is enabled). 1 = No CRC error, housecleaning is complete. When the SEU detection function is enabled, INIT_B is optionally driven Low when a readback CRC error is detected.
INIT_COMPLETE	Status[2]	Internal signal, accessible through the Virtex-6 FPGA status register	Indicates whether INIT_B signal is internally released.
MODE_STATUS[2:0]	Status	Internal signals, accessible through the Virtex-6 FPGA status register	Reflects the values sampled on the Mode pins when the status is read.
FS_STATUS[2:0]	Status	Internal signals, accessible through the Virtex-6 FPGA status register	Reflects the values sampled on the FS[2:0] pins when INIT_B is asserted High.

Notes:

1. Information on the Virtex-6 FPGA status register is available in Table 6-34, page 118. Information on accessing the device status register via JTAG is available in Chapter 11, Advanced JTAG Configurations. Information on accessing the device status register via SelectMAP is available in Chapter 7, Readback and Configuration Verification.
2. The Status type is an internal status signal without a corresponding pin.

Bitstream Security

This section discusses the available types of FPGA bitstream security including: bitstream encryption and bitstream authentication. See XAPP1084, *Developing Tamper Resistant Designs with Xilinx Virtex-6 and 7 Series FPGAs*, for additional details on bitstream encryption/authenication and advanced security features. Information security threats addressed by the Virtex-6 FPGA security features include:

- Reverse Engineering: A competitor analyzes an unencrypted bitstream to glean information about its function. The advanced encryption standard (AES) in Virtex-6 devices provides strong protection against this threat.
- Cloning/Overbuilding: A manufacturing facility builds more than is ordered and then sells to a competitor. A competitor copies the Xilinx FPGA bitstream and then uses it in an identical device, selling it as their own. AES provides strong protection against cloning. The bitstream cannot be successfully loaded into an FPGA that has not been previously loaded with the user's key.

- Spoofing: A competitor replaces the FPGA bitstream with their own bitsteam by replacing the source (ie., flash memory device). The eFUSE option in the Virtex-6 FPGA can prevent unencrypted bitstreams from being loaded.
- Tampering: A competitor modifies or extracts data from the user's design. Bitstream authentication provides cryptographically strong protection against ciphertext modification attacks.

Bitstream Encryption

Virtex-6 devices have on-chip AES decryption logic to provide a high degree of design security. Without knowledge of the encryption key, potential pirates cannot analyze an externally intercepted bitstream to understand or clone the design. Encrypted Virtex-6 FPGA designs cannot be copied or reverse-engineered.

The Virtex-6 FPGA AES system consists of software-based bitstream encryption and on-chip bitstream decryption with dedicated memory for storing the encryption key. Using the Xilinx ISE software, the user generates the encryption key and the encrypted bitstream. Virtex-6 devices store the encryption key internally in either dedicated RAM, backed up by a small externally connected battery, or in the eFUSE. The encryption key can only be programmed onto the device through the JTAG port.

During configuration, the Virtex-6 device performs the reverse operation, decrypting the incoming bitstream. The Virtex-6 FPGA AES encryption logic uses a 256-bit encryption key.

The on-chip AES decryption logic cannot be used for any purpose other than bitstream decryption; i.e., the AES decryption logic is not available to the user design and cannot be used to decrypt any data other than the configuration bitstream.

AES Overview

The Virtex-6 FPGA encryption system uses the Advanced Encryption Standard AES256 Cipher Block Chaining (CBC) Mode algorithm approved by the National Institute of Standards and Technology (NIST). (CBC is a "non-error propagating mode".) AES is an official standard supported by the NIST and the U.S. Department of Commerce (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf).

The Virtex-6 FPGA AES encryption system uses a 256-bit encryption key (the alternate key lengths of 128 and 192 bits described by NIST are not implemented) to encrypt or decrypt blocks of 128 bits of data at a time. According to NIST, there are 1.1×10^{77} possible key combinations for a 256-bit key.

Symmetric encryption algorithms such as AES use the same key for encryption and decryption. The security of the data is therefore dependent on the secrecy of the key.

Creating an Encrypted Bitstream

BitGen, provided with the Xilinx ISE software, can generate encrypted as well as non-encrypted bitstreams. For AES bitstream encryption, the user specifies a 256-bit key as an input to BitGen. BitGen in turn generates an encrypted bitstream file (`BIT`) and an encryption key file (`NKY`). Virtex-6 FPGAs allow for bitstreams to be created with both compression (**`-g compress`**) and encryption (**`-g encrypt`**).

For specific BitGen commands and syntax, refer to UG628, *Command Line Tools User Guide.*

Loading the Encryption Key

The encryption key can only be loaded onto a Virtex-6 device through the JTAG interface. The customer can program the key into either the eFUSE or battery-backed RAM (BBRAM):

- BBRAM is best suited for applications requiring reprogrammability, tamper resistance, and passive and active key loading. The BBRAM requires an external battery so that high-temperature operation can be limited, based on the battery vendor specifications.
- The eFUSE prevents spoofing and does not require an external battery, but is less secure than the BBRAM solution because the eFUSE key cannot be erased.

The iMPACT tool, provided with the Xilinx ISE software, can accept the NKY file as an input and program the key into BBRAM or eFUSE bits. Programming the key into the eFUSE bits can only be done with the iMPACT software and the Xilinx Platform Cable USB II. Programming the key into BBRAM can be done with iMPACT or third-party programmers. If a third-party tool is used to program the key into BBRAM, an SVF file can be created in the iMPACT tool containing the necessary JTAG commands needed to program the key.

To program the key, the device enters a special *key-access mode* using the ISC_PROGRAM_KEY instruction. In this mode, all FPGA memory, including the encryption key and configuration memory, is cleared. After the key is programmed and the key-access mode is exited, the key cannot be read out of the device by any means, and it cannot be reprogrammed without clearing the entire device. The key-access mode is transparent to most users.

When loading the key in the eFUSE bits, the user can read back the key for verification purposes and then the user must program the read_en_b_key and the write_en_b_key in the FUSE_CNTL register to disable reading and writing of the AES key. For more details, see eFUSE, page 107.

Loading Encrypted Bitstreams

Once the device has been programmed with the correct encryption key, the device can be configured with an encrypted bitstream. After configuration with an encrypted bitstream, it is not possible to read the configuration memory through JTAG or SelectMAP readback, regardless of the BitGen security setting.

While the device holds an encryption key, a non-encrypted bitstream can be used to configure the device only after POR or PROG is asserted, thus clearing out the configuration memory. In this case the key is ignored. After configuring with a non-encrypted bitstream, readback is possible (if allowed by the BitGen security setting). The encryption key still cannot be read out of the device, preventing the use of *Trojan Horse* bitstreams to defeat the Virtex-6 FPGA encryption scheme.

The method of configuration is not affected by encryption. The configuration bitstream can be delivered in any mode (Serial, JTAG, or any x8 parallel modes) from any configuration solution (PROM, System ACE™ controller, etc.). The x16 and x32 bus widths are not supported for encrypted bitstreams. Configuration timing and signaling are also unaffected by encryption.

Partial reconfiguration through external configuration interfaces is permitted with BitGen security settings of **`None`** or **`Level1`**. After an initial encrypted configuration, ICAP supports encrypted or unencrypted partial reconfiguration. After configuration, the device cannot be reconfigured without toggling the PROGRAM_B pin, cycling power, or issuing

the JPROGRAM instruction. Fallback reconfiguration and IPROG reconfiguration are enabled in Virtex-6 FPGAs after encryption is turned on. Readback is available through the ICAP primitive (see Bitstream Encryption and Internal Configuration Access Port (ICAP)). None of these events resets the key if V_{BATT} or V_{CCAUX} is maintained.

A mismatch between the key in the encrypted bitstream and the key stored in the device causes configuration to fail with the INIT_B pin pulsing Low and then back High if fallback is enabled, and the DONE pin remaining Low. In the event of an encrypted configuration failure, attempts to read or write to the configuration memory are blocked until after a successful re-configuration. This prevents successful System Monitor readback or eFUSE programming.

Bitstream Encryption and Internal Configuration Access Port (ICAP)

The Internal Configuration Access Port (ICAP) primitive provides the user logic with access to the Virtex-6 FPGA configuration interface. The ICAP interface is similar to the SelectMAP interface, although the restrictions on readback for the SelectMAP interface do not apply to the ICAP interface after configuration. Users can perform readback through the ICAP interface even if bitstream encryption is used. Unless the designer wires the ICAP interface to user I/O, this interface does not offer attackers a method for defeating the Virtex-6 FPGA AES encryption scheme.

Users concerned about the security of their design should *not*:

- Wire the ICAP interface to user I/O

-or-

- Instantiate the ICAP primitive.

Like the other configuration interfaces, the ICAP interface does not provide access to the key register.

AES_VIRTEX6

The AES supported in Virtex-6 FPGAs is identical to that supported in Virtex-5 and Virtex-4 devices. A 256-bit encryption key is loaded into eFUSE bits or battery-backed RAM by the user. BitGen, using AES, encrypts the bitstream.

This feature allows a user to encrypt their bitstream using 256-bit AES encryption in cipher block chaining (CBC) mode. The user can supply a 128-bit Initial Vector and 256-bit key, or let the software choose a pseudo-random key. Some security features such as KEY_CLEAR require that the part be configured with an encrypted bitstream in order to function. KEY_CLEAR clears both the AES 256-bit key in BBRAM and the expanded AES key in the decryptor.

Table 6-19: **BitGen Attributes**

Name	Type	Settings (Default)	Description
KeyFile	string	<design>.nky	Contains part AES key and part AES initial vector. BitGen creates a randomly generated key and initial vector if a file does not exist.
Encrypt	Boolean	Yes, No (*)	Enabled to encrypt the bitstream. Sets the CTL0[6] (dec) bit in the bitstream.

Table 6-19: **BitGen Attributes *(Cont'd)***

Name	Type	Settings (Default)	Description
Key0	string		Allows a key to be specified. Will be written into the Key file.
StartCBC	string		Allows the initial vector to be specified. Will be written into the Key file.
EncryptKeySelect	enum	bbram (*), efuse	Allows the user to choose between eFUSE and a battery-backed RAM key for encrypted bitstream. Sets the CTL0[31] (efuse_key) bit in the bitstream.

Xilinx does not provide specific key management guidance. It is up to the user to decide how to implement the key management for their system. Xilinx understands that key management is a very important element in a cryptographic system. For more information, refer to this key management site by NIST: http://csrc.nist.gov/groups/ST/toolkit/key_management.html.

V_{BATT}

The encryption key memory cells are volatile and must receive continuous power to retain their contents. During normal operation, these memory cells are powered by the auxiliary voltage input (V_{CCAUX}), although a separate V_{BATT} power input is provided for retaining the key when V_{CCAUX} is removed. Because V_{BATT} draws very little current (on the order of nanoamperes), a small watch battery is suitable for this supply. (To estimate the battery life, refer to V_{BATT} DC Characteristics in DS152, *Virtex-6 FPGA Data Sheet* and the battery specifications.)

V_{BATT} does not draw any current and can be removed while V_{CCAUX} is applied. V_{BATT} cannot be used for any purpose other than retaining the encryption keys when V_{CCAUX} is removed.

Bitstream Authentication

Overview

The Virtex-6 device is the first FPGA to offer cryptographically strong bitstream authentication. Virtex-6 devices have an on-chip bitstream keyed-Hash Message Authentication Code (HMAC) algorithm implemented in hardware to provide additional security beyond that provided by the AES decryption alone. Without knowledge of the AES and HMAC keys, the bitstream cannot be loaded, modified, intercepted, or cloned. AES provides the basic design security to protect the design from copying or reverse engineering, while HMAC provides assurance that the bitstream provided for the configuration of the FPGA was the unmodified bitstream allowed to load. Bitstream authentication makes bitstream tampering infeasible. Any bitstream tampering including single-bit flips are detected.

The HMAC algorithm uses a key that is provided to the ISE software. Alternately, the ISE software can automatically generate a random key. The HMAC key is separate and different from the AES key. The ISE software then utilizes the key and the SHA algorithm to generate a 256-bit result called the Message Authentication Code (MAC). The MAC, transmitted as part of the AES encrypted bitstream, verifies both data integrity and authenticity of the bitstream. Authentication covers the entire bitstream for all types of

control and data. When used, the Virtex-6 FPGA security solution always consists of both HMAC and AES.

Implementation

The Virtex-6 FPGA HMAC authentication system consists of an HMAC component in the ISE software and a hardware component integrated into every Virtex-6 FPGA. Both components generate a 256-bit MAC based on a key and the Secure Hash Algorithm (SHA256). During bitstream generation with the ISE software, the BitGen program generates a MAC that is embedded in the AES encrypted bitstream. During configuration, the HMAC/SHA256 engine in the FPGA calculates the MAC from the hardware AES decrypted data, and compares it with the MAC provided in the encrypted bitstream. If the two MACs match, the configuration goes to completion through the startup cycle. If the two MACs do not match and fallback is enabled, the fallback bitstream is loaded after the entire device configuration has been cleared. If fallback is not enabled, the configuration logic disables the configuration interface, blocking any access to the FPGA. Pulsing the PROGRAM_B signal or power-on reset is required to reset the configuration interface.

No On-Chip Key Storage for the HMAC Key is Required

The Virtex-6 FPGA authentication system uses the SHA256 FIPS PUB-182-2 (http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf) and HMAC FIPS PUB-198 (http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf) algorithms as published by the National Institute of Standards (NIST). Other bit variants of the of the SHA algorithm are not implemented.

The AES encrypted authenticated bitstream can be loaded through any of the external standard configuration interfaces except 16-bit and 32-bit SelectMAP (JTAG, serial, parallel, etc.) or ICAP. After the part has been configured with an encrypted bitstream, another unencrypted bitstream can only be loaded after the PROGRAM_B pin was asserted, there was a JTAG JPROGRAM command, or there was a power on reset invoked, thus clearing out all current configuration memory prior to loading the next configuration. The BitGen option **SecAll** can be specified to completely disable read/write access from an external configuration port after the first encrypted bitstream has been loaded. JTAG non-configuration functions are still available, but FPGA serial and parallel chaining are not possible in this mode.

Creating an Authenticated Bitstream

Because the HMAC must use a key different from the AES key, two keys are specified in the ISE software. The HMAC function is performed on the entire unencrypted bitstream utilizing the SHA256 function. The bitstream containing the HMAC key is then AES encrypted such that the only words prior to the encrypted bitstream are the sync word, a command telling the FPGA to use the encryptor, and a decrypt word count.

The Xilinx bitstream security function (always AES and HMAC together) is invoked by specifying the BitGen **-g Encrypt:yes** option. Virtex-6 FPGAs allow for bitstreams to be created with both compression (**-g compress**) and encryption (**-g encrypt**). (Refer to the BitGen section of UG628, *Command Line Tools User Guide* for more information.) The user can specify an HMAC key in the NKY file or let the ISE software generate a random key automatically.

The NKY file format is:

```
KEY HMAC <hex string>    (256 bit HMAC key)
```

For example:

```
Key HMAC 505daf31dea6930375003b9286bb183752457a90a79ace727b516f0009995a9e;
```

eFUSE

Virtex-6 FPGAs provide a non-volatile alternative for key storage using eFUSE. The eFUSE register bits are written using JTAG. Additional details on the trade-offs using eFUSE or BBRAM can be found at Loading the Encryption Key, page 103.

The fuse link is programmed (or burned or blown) by flowing a large current for a specific amount of time. Fuse programming current is provided by a fixed external voltage supply (VFS pin). The maximum level is controlled by an internally generated supply.

The resistance of a programmed fuse link is typically a few orders of magnitude higher than that of a pristine one. A programmed fuse is assigned a logic value of 1 and a pristine fuse 0.

Each logical bit consists of two eFUSE cells (primary and redundant), a flip-flop, and common logic elements for data multiplexing.

eFUSE Registers

A Virtex-6 FPGA has a total of four eFUSE registers. Table 6-20 lists the eFUSE registers in Virtex-6 devices with their sizes and usage. The eFUSE bits are addressed so that the LSB is shifted in/out first and MSB is last.

Table 6-20: **eFUSE Registers**

Register Name	Size (Bits)	Contents	Description
FUSE_KEY	256	Bitstream encryption key [0:255] (bit 255 shifted first)	Stores a key for use by AES bitstream decryptor. The eFUSE key can be used instead of the key stored in battery-backed SRAM. The AES key is used by the Virtex-6 FPGA decryption engine to load encrypted bitstreams. Depending on the read/write access bits in the FUSE_CNTL register, the AES key can be programmed and read through the JTAG port.
FUSE_USER	32	User defined [31:0] (bit 0 shifted first)	Stores a 32-bit user-defined code. This register is readable from the FPGA fabric using the EFUSE_USR primitive. Depending on the read/write access bits in the FUSE_CNTL register, the code can be programmed and read through the JTAG port.
FUSE_ID	64	Unique device identifier bits [63:0], where bits [6:0] are reserved and bits [63:7] correspond to the 57-bit ISC_DNA register and DNA_PORT value. Device DNA [63:7] (bit 63 shifted first)	Stores device DNA, a read-only register that is accessed through the JTAG port or the DNA_PORT primitive.

Table 6-20: **eFUSE Registers *(Cont'd)***

Register Name	Size (Bits)	Contents	Description
FUSE_CNTL	32	Control bits [31:0] (bit 0 shifted first)	Controls key use and read/write access to eFUSE registers. This register can be programmed and read through the JTAG port.

eFUSE bits are one-time programmable. FUSE_USER (EFUSE_USR) and a subset (57 bits) of FUSE_ID are used for the DNA_PORT. They are accessible to the FPGA logic by instantiation of corresponding user primitives.

eFUSE Control Register (FUSE_CNTL)

This register contains eight user programmable bits. These bits are used to select AES key usage and set the read/write protection for eFUSE registers, as detailed in Table 6-21. Bit 0 is shifted in or out first.

The eFUSE bits are one-time-programmable (OTP). Once programmed, they cannot be unprogrammed. For example, if access to a register is disabled, it cannot be re-enabled.

Table 6-21: **FUSE_CNTL Register Bits**

Bit #	Name	Description	Comments
0	cfg_aes_only	The FPGA can only be configured using the AES key stored in the eFUSE KEY register after this bit is programmed.	Unencrypted bitstreams or bitstreams encrypted using the battery-backed SRAM key can be used until this bit is programmed. ***Caution!*** If this bit is programmed, the device cannot be used unless the AES key is known. Return material authorizations (RMAs) cannot be accepted if this bit is programmed.
1	aes_exclusive	Fully disables partial reconfiguration.	Non-ICAP writes to FDRI are disabled. Partial reconfiguration is disabled. ***Caution!*** If this bit is programmed, RMAs are limited in device analysis and debug. An alternative to preventing readback and configuration is Security Level2.
2	read_en_b_key	Disable reading of the KEY register after this bit is programmed.	Users must program this bit after programming and verifying the AES key to prevent any reading of the AES key.

Table 6-21: **FUSE_CNTL Register Bits *(Cont'd)***

Bit #	Name	Description	Comments
3	read_en_b_user	Disable reading of FUSE_USER register after this bit is programmed.	Users are prevented from programming and reading the FUSE_USER register via JTAG when this bit is set, but this does not disable EFUSE_USR access.
4	write_en_b_key	Disable programming the KEY register after this bit is programmed.	Users must program this bit after programming and verifying the AES key to prevent any writing of the AES key.
5	-	-	Reserved
6	write_en_b_user	Disable programming the FUSE_USER register after this bit is programmed.	
7	write_en_b_cntl	Disable programming the FUSE_CNTL register after this bit is programmed.	
8:31	-	-	Reserved

When FUSE_CNTL[0] is NOT programmed:

- Encryption can be enabled or disabled via the BitGen options.
- The AES key stored in eFUSE or battery-backed SRAM can be selected via the BitGen options.

  Caution! When FUSE_CNTL[0] is programmed, only bitstreams encrypted with the eFUSE key can be used to configure the FPGA. This precludes device configuration from Xilinx test bitstreams and Xilinx pre-built bitstreams. Thus, Xilinx does not support RMA requests or iMPACT indirect SPI/BPI flash programming for devices that have the FUSE_CNTL[0] bit programmed.

Configuration memory is blocked after initial configuration if FUSE_CNTL[1] is programmed. The only way to reconfigure the device is to power cycle, issue a JPROGRAM or IPROG command, or pulse the PROGRAM_B pin.

JTAG Instructions

eFUSE registers can be read through JTAG ports. eFUSE programming can be done only via JTAG. Table 6-22 lists eFUSE-related JTAG instructions. The JTAG instruction register is 10 bits long. Refer to Chapter 11, Advanced JTAG Configurations, for general JTAG communication protocol. These instructions are not sufficient to program eFUSEs. A precise algorithm is required and not provided. The only supported method of programming eFUSEs is by using the iMPACT software and Xilinx Platform Cable USB II.

Table 6-22: **eFUSE-Related JTAG Instructions**

JTAG Instruction	Code	Action
FUSE_KEY	1111110001	Selects the FUSE_KEY register
FUSE_ID	1111110010	Reserved
ISC_DNA	1111010111	Selects a 57-bit subset of FUSE_ID (read-only)

Table 6-22: **eFUSE-Related JTAG Instructions *(Cont'd)***

JTAG Instruction	Code	Action
FUSE_USER	1111110011	Selects the FUSE_USER register
FUSE_CNTL	1111110100	Selects the FUSE_CNTL register

VFS Pin

In Virtex-6 devices, the VFS pin is the only extra pin dedicated to eFUSE operation. The VFS pin should be treated as a power supply pin for testing purposes such as power-up ramp and ESD stress.

Refer to DS152, *Virtex-6 FPGA Data Sheet*, for the VFS voltage specification during programming. For the read mode, the VFS pin needs to be lower than the V_{CCAUX} maximum operation condition. The VFS pin can be connected to GND if the application does not use the eFUSE or if the application does not program the eFUSE bits onboard.

Configuration Memory Frames

Virtex-6 FPGA configuration memory is arranged in frames that are tiled about the device. These frames are the smallest addressable segments of the Virtex-6 FPGA configuration memory space, and all operations must therefore act upon whole configuration frames. Frame counts and configuration sizes are shown in Table 6-23. Depending on BitGen options, additional overhead exists in the configuration bitstream. The exact bitstream length is available in the rawbits file (`.rbt`) created by using the **-b** option with BitGen or by selecting "Create ASCII Configuration File" in the Generate Programming File options popup in ISE software. Bitstream length (words) are roughly equal to the configuration array size (words) plus configuration overhead (words). Bitstream length (bits) are roughly equal to the bitstream length in words times 32.

Table 6-23: **Virtex-6 FPGA Frame Count, Frame Length, and Bitstream Size**

Device	Configuration Frames	Total Device Frames	Frame Lengths in Words(1)	Configuration Array Size in Words(2)	Bitstream Overhead in Words(3)
LX75T	7491	10116	81	606771	583
LX130T	16840	16860	81	1364040	583
LX195T	23720	23740	81	1921320	583
LX240T	28464	28488	81	2305584	583
LX365T	37032	37056	81	2999592	583
LX550T	55548	55584	81	4499388	583
LX760	71262	71298	81	5772222	583
SX315T	40272	40296	81	3262032	583
SX475T	60408	60444	81	4893048	583
HX250T	30780	30804	81	2493180	583

Table 6-23: **Virtex-6 FPGA Frame Count, Frame Length, and Bitstream Size *(Cont'd)***

Device	Configuration Frames	Total Device Frames	Frame Lengths in Words(1)	Configuration Array Size in Words(2)	Bitstream Overhead in Words(3)
HX255T	30780	30804	81	2493180	583

Notes:

1. All Virtex-6 FPGA configuration frames consist of 81 32-bit words.
2. Configuration array size equals the number of configuration frames times the number of words per frame.
3. Configuration overhead consists of commands in the bitstreams that are needed to perform configuration but do not themselves program any memory cells. Configuration overhead contributes to the overall bitstream size.

Configuration Packets

All Virtex-6 FPGA bitstream commands are executed by reading or writing to the configuration registers.

Packet Types

The FPGA bitstream consists of two packet types: Type 1 and Type 2. These packet types and their usage are described below.

Type 1 Packet

The Type 1 packet is used for register reads and writes. Only 5 out of 14 register address bits are used in Virtex-6 FPGAs. The header section is always a 32-bit word.

Following the Type 1 packet header is the Type 1 Data section, which contains the number of 32-bit words specified by the word count portion of the header.

Table 6-24: **Type 1 Packet Header Format**

Header Type	Opcode	Register Address	Reserved	Word Count
[31:29]	[28:27]	[26:13]	[12:11]	[10:0]
001	xx	RRRRRRRRRxxxxx	RR	xxxxxxxxxxx

Notes:

1. "R" means the bit is not used and reserved for future use. The reserved bits should be written as 0s.

Table 6-25: **OPCODE Format**

OPCODE	Function
00	NOOP
01	Read
10	Write
11	Reserved

Type 2 Packet

The Type 2 packet, which must follow a Type 1 packet, is used to write long blocks. No address is presented here because it uses the previous Type 1 packet address. The header section is always a 32-bit word.

Following the Type 2 packet header is the Type 2 Data section, which contains the number of 32-bit words specified by the word count portion of the header.

Table 6-26: **Type 2 Packet Header**

Header Type	Opcode	Word Count
[31:29]	[28:27]	[26:0]
010	xx	xxxxxxxxxxxxxxxxxxxxxxxxxx

Configuration Registers

Table 6-27 summarizes the Type 1 Packet registers. A detailed explanation of selected registers follows.

Table 6-27: **Type 1 Packet Registers**

Name	Read/Write	Address	Description
CRC	Read/Write	00000	CRC Register
FAR	Read/Write	00001	Frame Address Register
FDRI	Write	00010	Frame Data Register, Input Register (write configuration data)
FDRO	Read	00011	Frame Data Register, Output Register (read configuration data)
CMD	Read/Write	00100	Command Register
CTL0	Read/Write	00101	Control Register 0
MASK	Read/Write	00110	Masking Register for CTL0 and CTL1
STAT	Read	00111	Status Register
LOUT	Write	01000	Legacy Output Register (DOUT for daisy chain)
COR0	Read/Write	01001	Configuration Option Register 0
MFWR	Write	01010	Multiple Frame Write Register
CBC	Write	01011	Initial CBC Value Register
IDCODE	Read/Write	01100	Device ID Register
AXSS	Read/Write	01101	User Bitstream Access Register
COR1	Read/Write	01110	Configuration Option Register 1
CSOB	Write	01111	Used for daisy chain parallel interface, similar to LOUT
WBSTAR	Read/Write	10000	Warm Boot Start Address Register
TIMER	Read/Write	10001	Watchdog Timer Register
BOOTSTS	Read	10110	Boot History Status Register

Table 6-27: **Type 1 Packet Registers *(Cont'd)***

Name	Read/Write	Address	Description
CTL1	Read/Write	11000	Control Register 1
DWC	Read/Write	11010	Decrypt Word Count. Readable when bit 6 of CTL0 is 0.

CRC Register

Writes to this register perform a CRC check against the bitstream data. If the value written matches the current calculated CRC, the CRC_ERROR flag is cleared and startup is allowed.

FDRI Register

Writes to this register configure frame data at the frame address specified in the FAR register. See Bitstream Composition, page 127.

FDRO Register

This read-only register provides readback data for configuration frames starting at the address specified in the FAR register.

MASK Register

Writes to the CTL0 and CTL1 registers are bit-masked by the MASK register.

LOUT Register

Software uses this register to drive data to the DOUT pin during serial daisy-chain configuration.

MFWR Register

This register is used by the bitstream compression option.

CBC Register

This register is used by the bitstream encryption option to hold the Initial Vector for AES decryption.

IDCODE Register

Any writes to the FDRI register must be preceded by a write to this register. The provided IDCODE must match the device's IDCODE.

A read of this register returns the device IDCODE.

AXSS Register

This register supports the USR_ACCESS_VIRTEX6 primitive (see USR_ACCESS_VIRTEX6, page 77).

CSOB Register

Software uses this register to assert the CSO_B pin for parallel daisy-chain operation.

DWC Register

This register contains the decrypt word count.

Command Register (CMD)

The Command Register is used to instruct the configuration control logic to strobe global signals and perform other configuration functions. The command present in the CMD register is executed each time the FAR is loaded with a new value. Table 6-28 lists the Command Register commands and codes.

Table 6-28: **Command Register Codes**

Command	Code	Description
NULL	00000	Null Command
WCFG	00001	Writes Configuration Data: used prior to writing configuration data to the FDRI.
MFW	00010	Multiple Frame Write: used to perform a write of a single frame data to multiple frame addresses.
DGHIGH/ LFRM	00011	Last Frame: Deasserts the GHIGH_B signal, activating all interconnects. The GHIGH_B signal is asserted with the AGHIGH command.
RCFG	00100	Reads Configuration Data: used prior to reading configuration data from the FDRO.
START	00101	Begins the Startup Sequence: initiates the startup sequence. The startup sequence begins after a successful CRC check and a DESYNC command are performed.
RCAP	00110	Resets the CAPTURE signal after performing readback-capture in single-shot mode.
RCRC	00111	Resets CRC: Resets the CRC register.
AGHIGH	01000	Asserts the GHIGH_B signal: places all interconnect in a High-Z state to prevent contention when writing new configuration data. This command is only used in shutdown reconfiguration. Interconnect is reactivated with the LFRM command.
SWITCH	01001	Switches the CCLK frequency: updates the frequency of the Master CCLK to the value specified by the OFSEL bits in the COR0 register.
GRESTORE	01010	Pulses the GRESTORE signal: sets/resets (depending on user configuration) IOB and CLB flip-flops.
SHUTDOWN	01011	Begin Shutdown Sequence: Initiates the shutdown sequence, disabling the device when finished. Shutdown activates on the next successful CRC check or RCRC instruction (typically an RCRC instruction).
GCAPTURE	01100	Pulses GCAPTURE: Loads the capture cells with the current register states.
DESYNC	01101	Resets the DALIGN signal: Used at the end of configuration to desynchronize the device. After desynchronization, all values on the configuration data pins are ignored.

Table 6-28: **Command Register Codes *(Cont'd)***

Command	Code	Description
Reserved	01110	Reserved.
IPROG	01111	Internal PROG for triggering a warm boot.
CRCC	10000	When Readback CRC is selected, the CRC is calculated after full configuration and reconfiguration. Toggling GHIGH also calculates the CRC. This command can be used when GHIGH is not toggled during reconfiguration (active partial reconfiguration).
LTIMER	10001	Reload watchdog timer.

Control Register 0 (CTL0)

The CTL0 register is used to configure the Virtex-6 device. Writes to the CTL0 register are masked by the value in the MASK Register (this allows the GTS_USR_B signal to be toggled without respecifying the SBITS and PERSIST bits). The name of each bit position in the CTL0 register is given in Table 6-29 and described in Table 6-30.

Table 6-29: **Control Register 0 (CTL0)**

Description	EFUSE_KEY	ICAP_SELECT	Reserved	OverTempPowerDown	Reserved	ConfigFallback	Reserved	GLUTMASK_B	FARSRC	DEC	SBITS[1:0]	PERSIST	Reserved	GTS_USR_B
Bit Index	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
Value	0	0	x x x x x x x x x x x x x x x x x	0	x	0	x	1	0	0	0 0	0	x x	1

Table 6-30: **Control Register 0 Description**

Name	Bit Index	Description
EFUSE_KEY	31	Selects the AES key source: 0: Battery-back RAM (default) 1: eFUSE This bit is internally latched again when DEC is set. It cannot change after that to prevent switching of key sources although this bit can still be read/write.
ICAP_SELECT	30	ICAP Port Select. 0: Top ICAP Port Enabled (default) 1: Bottom ICAP Port Enabled
OverTempPowerDown	12	Enables the System Monitor Over-Temperature power down. 0: Disables Over-Temperature power down (default) 1: Enables Over-Temperature power down
ConfigFallback	10	Stops when CFG fails and disables fallback to the default bitstream. The BitGen option is **`ConfigFallback: Enable*/Disable`**. 0: Enables fallback (default) 1: Disables fallback
GLUTMASK_B	8	Global LUT mask signal. Masks any changeable memory cell readback value. 0: Masks changeable memory cell readback value. 1: Does not mask changeable memory cell readback values (default).
FARSRC	7	Determines the output of FAR[23:0] configuration register. 0: EFAR, the address of ECC error frame (default) 1: FAR, the address of RBCRC
DEC	6	AES Decryptor enable bit. 0: Decryptor disabled (default) 1: Decryptor enabled

Table 6-30: **Control Register 0 Description *(Cont'd)***

Name	Bit Index	Description
SBITS[1:0]	[5:4]	Security level. The Virtex-6 FPGA security level is extended to encrypted bitstreams. It is applicable to the Configuration port, not to ICAP. The security level takes affect at the end of the encrypted bitstream or after EOS for an unencrypted bitstream. 00: Read/Write OK (default) 01: Readback disabled 1x: Both Writes and Reads disabled Only FAR and FDRI allow encrypt write access for security levels 00 and 01.
PERSIST	3	The configuration interface defined by M2:M0 remains after configuration. Typically used only with the SelectMAP interface to allow reconfiguration and readback. See also SelectMAP Configuration Interface in Chapter 2. This option is also sometimes used for serial interface partial reconfiguration. See Slave Serial Configuration in Chapter 2. 0: No (default) 1: Yes
GTS_USR_B	0	Active-Low global 3-state I/Os. Turns off pull-ups if GTS_CFG_B is also asserted. 0: I/Os 3-stated 1: I/Os active (default)

Control Register 1 (CTL1)

The CTL1 register is used to configure the Virtex-6 device. This register is reserved. See Table 6-31.

Table 6-31: **Control Register 1 (CTL1)**

Description	Reserved																														
Bit Index	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
Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Frame Address Register (FAR)

The Virtex-6 devices are divided into two halves, the top and the bottom. All frames in Virtex-6 devices have a fixed, identical length of 2,592 bits (81 32-bit words).

The FAR is divided into five fields: block type, top/bottom bit, row address, column address, and minor address. See Table 6-32. The address can be written directly or can be auto-incremented at the end of each frame. The typical bitstream starts at address 0 and auto-increments to the final count.

Table 6-32: **Frame Address Register Description**

Address Type	Bit Index	Description
Block Type	[23:21]	Valid block types are CLB, I/O, CLK (000), block RAM content (001), and CFG_CLB (010). A normal bitstream does not include type 010.
Top_B Bit	20	Select between top-half rows (0) and bottom-half rows (1).
Row Address	[19:15]	Selects the current row. The row addresses increment from center to top and then reset and increment from center to bottom.
Column Address	[14:7]	Selects a major column, such as a column of CLBs. Column addresses start at 0 on the left and increase to the right.
Minor Address	[6:0]	Selects a frame within a major column.

Status Register (STAT)

The Status Register indicates the value of numerous global signals. The register can be read through the SelectMAP or JTAG interfaces. Table 6-33 gives the name of each bit position in the STAT register; a detailed explanation of each bit position is given in Table 6-34.

Table 6-33: **Status Register**

Description	EFUSE_BUSY	Reserved	BAD_PACKET	HSWAPEN_B	Reserved	BUS_WIDTH		FS				Reserved	STARTUP_STATE				SYSMON_OVER_TEMP	DEC_ERROR	ID_ERROR	DONE	RELEASE_DONE	INIT_B	INIT_COMPLETE	MODE			GHIGH_B	GWE	GTS_CFG_B	EOS	DCI_MATCH	MMCM_LOCK	PART_SECURED	CRC_ERROR
Bit Index	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
Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 6-34: **Status Register Description**

Name	Bit Index	Description
EFUSE_BUSY	31	Indicates that eFUSE logic is busy: 0: eFUSE logic is not busy 1: eFUSE logic is busy
BAD_PACKET	29	Illegal bitstream packet seen during first configuration: 0: No bad packet detected 1: Bad packet detected
HSWAPEN_B	28	HSWAPEN pin status.
BUS_WIDTH	[26:25]	CFG bus width auto detection result. If ICAP is enabled, this field reflects the ICAP bus width after configuration is done. 00 = x1 01 = x8 10 = x16 11 = x32

Table 6-34: **Status Register Description *(Cont'd)***

Name	Bit Index	Description
FS	[24:22]	SPI flash type select. Refer to Table 2-7, page 45 for the encoding.
STARTUP_STATE	[20:18]	CFG startup state machine (0 to 7). Phase 0 = 000 Phase 1 = 001 Phase 2 = 011 Phase 3 = 010 Phase 4 = 110 Phase 5 = 111 Phase 6 = 101 Phase 7 = 100
SYSMON_OVER_TEMP	17	System Monitor over-temperature alarm: 0: No over-temperature condition (default) 1: Over-temperature condition
DEC_ERROR	16	FDRI write attempted before or after decrypt operation: 0: No DEC_ERROR 1: DEC_ERROR
ID_ERROR	15	Attempt to write to FDRI without successful DEVICE_ID check. 0: No ID_ERROR 1: ID_ERROR
DONE	14	Value on DONE pin
RELEASE_DONE	13	Value of internal DONE signal: 0: DONE signal not released (pin is actively held Low) 1: DONE signal released (can be held Low externally)
INIT_B	12	Value on INIT_B pin
INIT_COMPLETE	11	Internal signal indicating initialization has completed: 0: Initialization has not finished 1: Initialization finished
MODE	[10:8]	Status of the Mode pins (M[2:0]).
GHIGH_B	7	Status of GHIGH_B: 0: GHIGH_B asserted 1: GHIGH_B deasserted
GWE	6	Status of GWE: 0: FFs and block RAM are write disabled 1: FFs and block RAM are write enabled
GTS_CFG_B	5	Status of GTS_CFG_B: 0: All I/Os are placed in High-Z state 1: All I/Os behave as configured

Table 6-34: **Status Register Description *(Cont'd)***

Name	Bit Index	Description
EOS	4	End of Startup signal from Startup Block: 0: Startup sequence has not finished 1: Startup sequence has finished
DCI_MATCH	3	0: DCI not matched 1: DCI is matched This bit is a logical AND function of all the MATCH signals (one per bank). If no DCI I/Os are in a particular bank, the bank's MATCH signal = 1.
MMCM_LOCK	2	0: MMCMs are not locked 1: MMCMs are locked This bit is a logical AND function of all MMCM LOCKED signals. Unused MMCM LOCKED signals = 1.
PART_SECURED	1	0: Decryptor security not set 1: Decryptor security set
CRC_ERROR	0	0: No CRC error 1: CRC error

Configuration Options Register 0 (COR0)

The Configuration Options Register 0 is used to set certain configuration options for the device. The name of each bit position in the COR0 is given in Table 6-35 and described in Table 6-36.

Table 6-35: **Configuration Options Register 0**

Description	Reserved	PWRDWN_STAT	Reserved	DONE_PIPE	DRIVE_DONE	SINGLE	OSCFSEL	SSCLKSRC	DONE_CYCLE	MATCH_CYCLE	Reserved	GTS_CYCLE	GWE_CYCLE
Bit Index	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
Value	0 0 0 0	0	X	0	0	0	0 0 0 0 0 0	0 x	0 1 1	1 1 1	1 1 1	1 0 1	1 0 0

Table 6-36: **Configuration Options Register 0 Description**

Name	Bit Index	Description
PWRDWN_STAT	27	Changes the DONE pin to a Powerdown status pin: 0: DONE pin 1: Powerdown pin
DONE_PIPE	25	0: No pipeline stage for DONEIN 1: Add pipeline stage for DONEIN The FPGA waits on DONE that is delayed by one StartupClk cycle. Use this option when StartupClk is running at high speeds.

Table 6-36: **Configuration Options Register 0 Description *(Cont'd)***

Name	Bit Index	Description
DRIVE_DONE	24	0: DONE pin is open drain 1: DONE is actively driven High
SINGLE	23	0: Readback is not single-shot New captured values are loaded on each successive CAP assertion on the CAPTURE_VIRTEX6 primitive. Capture can also be performed with the GCAPTURE instruction in the CMD register. 1: Readback is single-shot. The RCAP instruction must be loaded into the CMD register between successive readbacks.
OSCFSEL	[22:17]	Select CCLK frequency in Master modes (2 MHz – 60 MHz)
SSCLKSRC	[16:15]	Startup-sequence clock source. 00: CCLK 01: UserClk (per connection on the CAPTURE_VIRTEX6 block) 1x: JTAGClk
DONE_CYCLE	[14:12]	Startup cycle to release the DONE pin. 000: Startup phase 1 001: Startup phase 2 010: Startup phase 3 011: Startup phase 4 100: Startup phase 5 101: Startup phase 6 110: Startup phase 7 111: Keep
MATCH_CYCLE	[11:9]	Startup cycle to stall in until DCI matches. 000: Startup phase 0 001: Startup phase 1 010: Startup phase 2 011: Startup phase 3 100: Startup phase 4 101: Startup phase 5 110: Startup phase 6 111: No Wait

Table 6-36: **Configuration Options Register 0 Description *(Cont'd)***

Name	Bit Index	Description
GTS_CYCLE	[5:3]	Startup cycle to deassert the Global 3-State (GTS) signal. 000: Startup phase 1 001: Startup phase 2 010: Startup phase 3 011: Startup phase 4 100: Startup phase 5 101: Startup phase 6 110: GTS tracks DONE pin. BitGen option **-g GTS_cycle:Done** 111: Keep
GWE_CYCLE	[2:0]	Startup phase to deassert the Global Write Enable (GWE) signal. 000: Startup phase 1 001: Startup phase 2 010: Startup phase 3 011: Startup phase 4 100: Startup phase 5 101: Startup phase 6 110: GWE tracks DONE pin. BitGen option **-g GWE_cycle:Done** 111: Keep

Configuration Options Register 1 (COR1)

The Configuration Options Register 1 is used to set certain configuration options for the device. The name of each bit position in the COR1 is given in Table 6-37 and described in Table 6-38.

Table 6-37: **Configuration Options Register 1**

Description	Reserved	PERSIST_DEASSERT_AT_DESYNC	RBCRC_ACTION	Reserved	RBCRC_NO_PIN	RBCRC_EN	Reserved	BPI_1ST_READ_CYCLES	BPI_PAGE_SIZE

Bit Index	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
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-38: **Configuration Options Register 1 Description**

Name	Bit Index	Description
PERSIST_DEASSERT_AT_DESYNC	17	Controls deassertion of PERSIST with the DESYNC command. 0: Disables deassertion of PERSIST with the DESYNC command (default) 1: Enables deassertion of PERSIST with the DESYNC command
RBCRC_ACTION	[16:15]	00: Continue (default) 01: Halt 11: CorrectAndHalt 10: CorrectAndContinue
RBCRC_NO_PIN	9	Controls INIT_B as a readback CRC error status output pin. 0: Disables INIT_B as a readback CRC error status output pin (default) 1: Enables INIT_B as a readback CRC error status output pin
RBCRC_EN	8	Controls continuous readback CRC enable. 0: Enables continuous readback CRC (default) 1: Disables continuous readback CRC

Table 6-38: **Configuration Options Register 1 Description *(Cont'd)***

Name	Bit Index	Description
BPI_1ST_READ_CYCLES	[3:2]	First byte read timing: 00: 1 CCLK (default) 01: 2 CCLKs 10: 3 CCLKs 11: 4 CCLKs
BPI_PAGE_SIZE	[1:0]	Flash memory page size: 00: 1 byte/word (default) 01: 4 bytes/words 10: 8 bytes/words 11: Reserved

Warm Boot Start Address Register (WBSTAR)

The name of each bit position in the WBSTAR is given in Table 6-39 and described in Table 6-40.

Table 6-39: **WBSTAR Register**

Description	Reserved			RS[1:0]		RS_TS_B	START_ADDR																									
Bit Index	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
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-40: **WBSTAR Register Description**

Name	Bit Index	Description
RS[1:0]	[28:27]	RS[1:0] pin value on next warm boot. The default is 00.
RS_TS_B	26	RS[1:0] pins 3-state enable. 0: Enable RS 3-state (default) 1: Disable RS 3-state
START_ADDR	[25:0]	Next bitstream start address. The default start address is 0x0000000.

Watchdog Timer Register (TIMER)

The Watchdog timer is automatically disabled for fallback bitstreams. The name of each bit position in the TIMER register is given in Table 6-41 and described in Table 6-42.

Table 6-41: **TIMER Register**

Description	Reserved						TIMER_USR_MON	TIMER_CFG_MON	TIMER_VALUE																							
Bit Index	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
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-42: **TIMER Register Description**

Name	Bit Index	Description
TIMER_USR_MON	25	Watchdog is enabled during user mode: 0: Disabled (default) 1: Enabled
TIMER_CFG_MON	24	Watchdog is enabled during configuration: 0: Disabled (default) 1: Enabled
TIMER_VALUE	[23:0]	Watchdog time-out value, CFG_MCLK is used for this counter. CFG_MCLK is approximately 100 KHz to 300 KHz. The default is 0x000000.

Boot History Status Register (BOOTSTS)

This register can only be reset by POR, asserting PROGRAM_B, or issuing a JPROGRAM instruction. At EOS or an error condition, status (_0) is shifted to status (_1), and status (_0) is updated with the current status. The name of each bit position in the BOOTSTS register is given in Table 6-43 and described in Table 6-44.

Table 6-43: **BOOTSTS Register**

Description	Reserved																	WRAP_ERROR_1	CRC_ERROR_1	ID_ERROR_1	WTO_ERROR_1	IPROG_1	FALLBACK_1	VALID_1	Reserved	WRAP_ERROR_0	CRC_ERROR_0	ID_ERROR_0	WTO_ERROR_0	IPROG_0	FALLBACK_0	VALID_0
Bit Index	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
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-44: **BOOTSTS Register Description**

Name	Bit Index	Description
WRAP_ERROR_1	14	BPI address counter wraparound error
CRC_ERROR_1	13	CRC error

Table 6-44: **BOOTSTS Register Description *(Cont'd)***

Name	Bit Index	Description
ID_ERROR_1	12	ID error
WTO_ERROR_1	11	Watchdog time-out error
IPROG_1	10	Internal PROG triggered configuration
FALLBACK_1	9	1: Fallback to default reconfiguration, RS[1:0] actively drives 2'b00 0: Normal configuration
VALID_1	8	Status valid
WRAP_ERROR_0	6	BPI address counter wraparound error
CRC_ERROR_0	5	CRC error
ID_ERROR_0	4	ID error
WTO_ERROR_0	3	Watchdog time-out error
IPROG_0	2	Internal PROG triggered configuration
FALLBACK_0	1	1: Fallback to default reconfiguration, RS[1:0] actively drives 2'b00 0: Normal configuration
VALID_0	0	Status valid

Notes:

1. The default power-up state for all fields in this register is 0, indicating no error, fallback, or valid configuration detected. After configuration, a 1 in any bit indicates an error case, fallback, or completed configuration has been detected.

Bitstream Composition

Configuration can begin after the device is powered and initialization has finished, as indicated by the INIT_B pin being released. After initialization, the packet processor ignores all data presented on the configuration interface until it receives the synchronization word. After synchronization, the packet processor waits for a valid packet header to begin the configuration process.

Device Identifier (Device DNA)

The Virtex-6 FPGA contains an embedded, device identifier (device DNA). The identifier is nonvolatile, permanently programmed into the FPGA, and is unchangeable making it tamper resistant. Each device is programmed with a DNA value that is most often unique. However, up to 32 devices within the family can contain the same DNA value.

The FPGA application accesses the identifier value using the Device DNA Access Port (DNA_PORT) design primitive, shown in Figure 6-13.

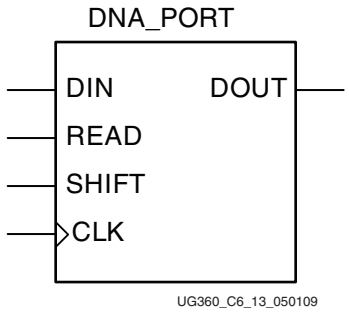


Figure 6-13: **Virtex-6 FPGA DNA_PORT Design Primitive**

Identifier Value

As shown in Figure 6-14, the device DNA value is 57 bits long.

Operation

Figure 6-14 shows the general functionality of the DNA_PORT design primitive. An FPGA application must first instantiate the DNA_PORT primitive, shown in Figure 6-13, within a design.

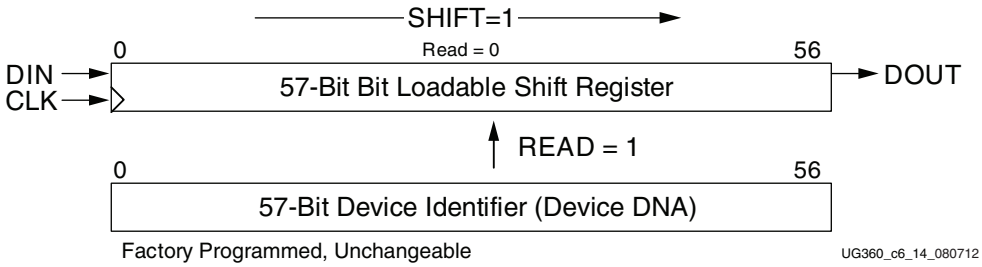


Figure 6-14: **DNA_PORT Operation**

To read the Device DNA, the FPGA application must first transfer the identifier value into the DNA_PORT output shift register. The READ input must be asserted during a rising

edge of CLK, as shown in Table 6-45. This action parallel loads the output shift register with all 57 bits of the identifier. The READ operation overrides a SHIFT operation.

To continue reading the identifier values, assert SHIFT followed by a rising edge of CLK, as shown in Table 6-45. This action causes the output shift register to shift its contents toward the DOUT output. The value on the DIN input is shifted into the shift register.

A Low-to-High transition on SHIFT should be avoided when CLK is High because this causes a spurious initial clock edge. Ideally, SHIFT should only be asserted when CLK is Low or on a falling edge of CLK.

If both READ and SHIFT are Low, the output shift register holds its value and DOUT remains unchanged. Refer to DS152, *Virtex-6 FPGA Data Sheet: DC and Switching Characteristics*, for identifer memory specifications.

Table 6-45: **DNA_PORT Operations**

Operation	DIN	READ	SHIFT	CLK	Shift Register	DOUT
HOLD	X	0	0	X	Hold previous value	Hold previous value
READ	X	1	X	↑	Parallel load with 57-bit ID	
SHIFT	DIN	0	1	↑	Shift DIN into bit 0, shift contents of Shift Register toward DOUT	Bit 56 of Shift Register

Notes:

X = Don't care
↑ = Rising clock edge

Extending Identifier Length

As shown in Figure 6-15, most applications that use the DNA_PORT primitive tie the DIN data input to a static value.

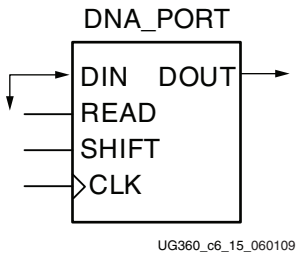


Figure 6-15: **Shift in Constant**

As shown in Figure 6-16, the length of the identifier can be extended by feeding the DOUT serial output port back into the DIN serial input port. This way, the identifier can be extended to any possible length.

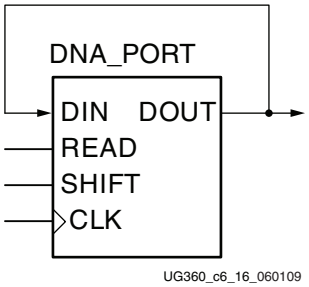


Figure 6-16: **Circular Shift**

It is also possible to add additional bits to the identifier using FPGA logic resources. As shown in Figure 6-17, the FPGA application can insert additional bits via the DNA_PORT DIN serial input. The additional bits provided by the logic resources could take the form of an additional fixed value or a variable computed from the device DNA.

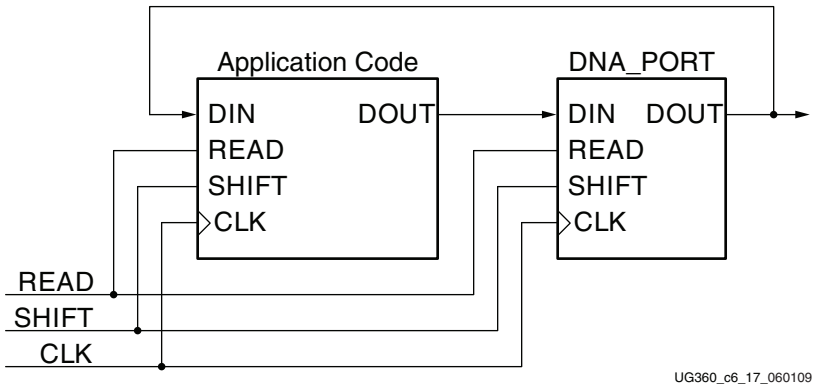


Figure 6-17: **Bitstream Specific Code**

JTAG Access to Device Identifier

The FPGA's internal device identifier, plus any values shifted in on the DIN input, can be read via the JTAG port using the private ISC_DNA command. This requires the ISC_ENABLE to be loaded before the ISC_DNA command is issued.

Bit 56 of the identifier, shown in Figure 6-14, appears on the TDO JTAG output following the ISC_DNA command when the device enters the Shift-DR state. The remaining Device DNA bits and any data on the input to the register are shifted out sequentially while the JTAG controller is left in the Shift-DR state.

iMPACT Access to Device Identifier

The iMPACT software in ISE 11.2 (and later) tools can also read the device DNA value. **`readDna -p <position>`** is the batch command that reads the device DNA from the FPGA.

XILINX®

Chapter 7

Readback and Configuration Verification

Virtex®-6 devices allow users to read configuration memory through the SelectMAP, ICAP, and JTAG interfaces. There are two styles of readback: Readback Verify and Readback Capture. During Readback Verify, the user reads all configuration memory cells, including the current values on all user memory elements (LUT RAM, SRL16, and block RAM). Readback Capture is a superset of Readback Verify—in addition to reading all configuration memory cells, the current state of all internal CLB and IOB registers is read, and is useful for design debugging.

To read configuration memory, users must send a sequence of commands to the device to initiate the readback procedure. Once initiated the device dumps the contents of its configuration memory to the SelectMAP or JTAG interface. The Accessing Configuration Registers through the SelectMAP Interface section and IEEE 1149.1 describe the steps for reading configuration memory.

Users can send the readback command sequence from a custom microprocessor, CPLD, or FPGA-based system, or use iMPACT to perform JTAG-based readback verify. iMPACT, the device programming software provided with the Xilinx® ISE® software, can perform all readback and comparison functions for Virtex-6 devices and report to the user whether there were any configuration errors. iMPACT cannot perform capture operations, although Readback Capture is seldom used for design debugging because the Integrated Logic Analyzer (ILA) in the ChipScope™ debugging tool, sold separately through the Xilinx website, provides superior design debugging functionality in a user-friendly interface.

Once configuration memory is read from the device, the next step is to determine if there are any errors by comparing the readback bitstream to the configuration bitstream. The Verifying Readback Data section explains how this is done.

Preparing a Design for Readback

There are two mandatory bitstream settings for readback: the BitGen security setting must not prohibit readback (**`-g security:none`**), and bitstream encryption must not be used. Additionally, if readback is to be performed through the SelectMAP interface, the port must be set to retain its function after configuration by setting the *persist* option in BitGen (**`-g Persist:Yes`**), otherwise the SelectMAP data pins revert to user I/O, precluding further configuration operations. Beyond these security and encryption requirements, no special considerations are necessary to enable readback through the Boundary-Scan port.

If capture functionality is needed, the CAPTURE_VIRTEX6 primitive can instantiated in the user design (Figure 7-6, page 145). Alternatively, writing the GCAPTURE command to the CMD register can be used (see Readback Capture). To capture the state of user

registers, the user design triggers the CAP input on this primitive, storing the current register values in configuration memory. The register values are later read out of the device along with all other configuration memory.

Readback Command Sequences

Virtex-6 FPGA configuration memory is read from the FDRO (Frame Data Register - Output) configuration register and can be accessed from the JTAG, SelectMAP, and ICAP interfaces. For the JTAG and SelectMAP interfaces, readback is possible while the FPGA design is active or in a shutdown state, although block RAMs cannot be accessed by the user design while they are being accessed by the configuration logic.

Accessing Configuration Registers through the SelectMAP Interface

To read configuration memory through the SelectMAP interface, users must set the interface for write control to send commands to the FPGA, and then switch the interface to read control to read data from the device. Write and read control for the SelectMAP interface is determined by the RDWR_B input: the SelectMAP data pins are inputs when the interface is set for Write control (RDWR_B = 0); they are outputs when the interface is set for Read control (RDWR_B = 1).

The CSI_B signal must be deasserted (CSI_B =1) before toggling the RDWR_B signal, otherwise the user causes an abort (refer to SelectMAP ABORT, page 169 for details).

The procedure for changing the SelectMAP interface from Write to Read Control, or vice versa, is:

1. Deassert CSI_B.
2. Toggle RDWR_B.

   RDWR_B = 0: Write control
   RDWR_B = 1: Read control
3. Assert CSI_B.
4. CSI_B and RDWR_B are synchronous to CCLK.
5. This procedure is illustrated in Figure 7-1.

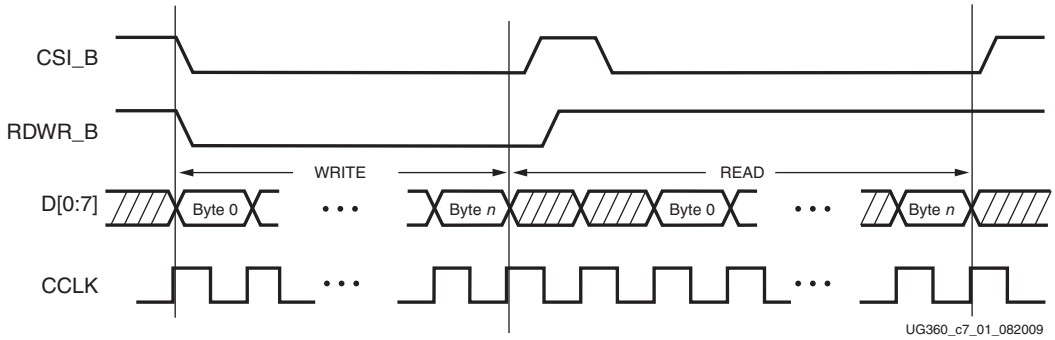


Figure 7-1: **Changing the SelectMAP Port from Write to Read Control**

Configuration Register Read Procedure (SelectMAP)

The simplest read operation targets a configuration register such as the COR0 or STAT register. Any configuration register with read access can be read through the SelectMAP interface, although not all registers offer read access. The procedure for reading the STAT register through the SelectMAP interface follows:

1. Write the Bus Width detection sequence and Synchronization word to the device followed by two no operation (NOOP) commands.
2. Write the *read STAT register* packet header to the device.
3. Write two NOOP commands to the device to flush the packet buffer.
4. Read one word from the SelectMAP interface; this is the Status register value.
5. Write the DESYNC command to the device.
6. Write two dummy words to the device to flush the packet buffer.

Table 7-1: **Status Register Readback Command Sequence (SelectMAP)**

Step	SelectMAP Port Direction	Configuration Data	Explanation
1	Write	FFFFFFFF	Dummy Word
2	Write	000000BB	Bus Width Sync Word
3	Write	11220044	Bus Width Detect
4	Write	FFFFFFFF	Dummy Word
5	Write	AA995566	Sync Word
6	Write	20000000	NOOP
7	Write	20000000	NOOP
8	Write	2800E001	Type 1 packet header to read STAT register
9	Write	20000000	NOOP
10	Write	20000000	NOOP
11	Read	SSSSSSSS	Device writes one word from the STAT register to the configuration interface
12	Write	30008001	Type 1 Write 1 Word to CMD
13	Write	0000000D	DESYNC Command
14	Write	20000000	NOOP
15	Write	20000000	NOOP

The user must change the SelectMAP interface from write to read control between steps 10 and 11, and back to write control after step 11, as illustrated in Figure 7-2.

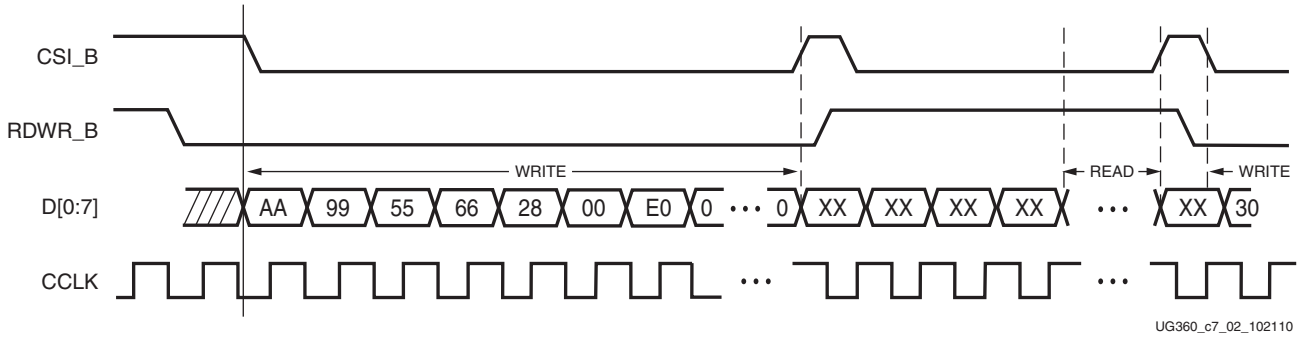


Figure 7-2: **SelectMAP Status Register Read**

To read registers other than STAT, the address specified in the Type-1 packet header in step 2 of Table 7-1 should be modified and the word count changed if necessary. Reading from the FDRO register is a special case that is described in Configuration Memory Read Procedure (SelectMAP).

Configuration Memory Read Procedure (SelectMAP)

The process for reading configuration memory from the FDRO register is similar to the process for reading from other registers. Additional steps are needed to accommodate the configuration logic. Configuration data coming from the FDRO register passes through the frame buffer. The first frame of readback data should be discarded.

1. Write the Bus Width detection sequence and Synchronization word to the device.
2. Write one NOOP command.
3. Write the Shutdown command, and write one NOOP command.
4. Write the RCRC command to the CMD register, and write one NOOP command.
5. Write five NOOP instructions to ensure the shutdown sequence has completed. DONE goes Low during the shutdown sequence.
6. Write the RCFG command to the CMD register, and write one NOOP command.
7. Write the Starting Frame Address to the FAR (typically `0x00000000`).
8. Write the *read FDRO register* packet header to the device. The FDRO read length is:

   FDRO Read Length = (words per frame) x (frames to read)

   One extra frame is read to account for the frame buffer. Users should strobe readback data while DOUT_BUSY is Low. The frame buffer produces one dummy frame at the beginning of the read. Also, one extra word is read in SelectMAP x8 mode.
9. Write 32 NOOP commands to the device to flush the packet buffer.
10. Read the FDRO register from the SelectMAP interface. The FDRO read length is the same as in step 9 above.
11. Write one NOOP instruction.
12. Write the START command, and write one NOOP command.
13. Write the RCRC command, and write one NOOP command.
14. Write the DESYNC command.
15. Write at least 64 bits of NOOP commands to flush the packet buffer. Continue sending CCLK pulses until DONE goes High.

Table 7-2 shows the readback command sequence.

Table 7-2: **Shutdown Readback Command Sequence (SelectMAP)**

Step	SelectMAP Port Direction	Configuration Data	Explanation
1	Write	FFFFFFFF	Dummy Word
		000000BB	Bus Width Sync Word
		11220044	Bus Width Detect
		FFFFFFFF	Dummy Word
		AA995566	Sync Word
2	Write	20000000	Type 1 NOOP Word 0
3	Write	30008001	Type 1 Write 1 Word to CMD
		0000000B	SHUTDOWN Command
		20000000	Type 1 NOOP Word 0
4	Write	30008001	Type 1 Write 1 Word to CMD
		00000007	RCRC Command
		20000000	Type 1 NOOP Word 0
5	Write	20000000	Type 1 NOOP Word 0
		20000000	Type 1 NOOP Word 0
		20000000	Type 1 NOOP Word 0
		20000000	Type 1 NOOP Word 0
		20000000	Type 1 NOOP Word 0
6	Write	30008001	Type 1 Write 1 Word to CMD
		00000004	RCFG Command
		20000000	Type 1 NOOP Word 0
7	Write	30002001	Type 1 Write 1 Word to FAR
		00000000	FAR Address = 00000000
8	Write	28006000	Type 1 Read 0 Words from FDRO
		48024090	Type 2 Read 147,600 Words from FDRO
9	Write	20000000	Type 1 NOOP Word 0
		. . .	Type 1 31 More NOOPs Word 0
10	Read	00000000	Packet Data Read FDRO Word 0
		. . .	
		00000000	Packet Data Read FDRO Word 147599
11	Write	20000000	Type 1 NOOP Word 0
12	Write	30008001	Type 1 Write 1 Word to CMD
		00000005	START Command
		20000000	Type 1 NOOP Word 0

Table 7-2: **Shutdown Readback Command Sequence (SelectMAP) *(Cont'd)***

Step	SelectMAP Port Direction	Configuration Data	Explanation
13	Write	30008001	Type 1 Write 1 Word to CMD
		00000007	RCRC Command
		20000000	Type 1 NOOP Word 0
14	Write	30008001	Type 1 Write 1 Word to CMD
		0000000D	DESYNC Command
15	Write	20000000	Type 1 NOOP Word 0
		20000000	Type 1 NOOP Word 0

User logic should strobe readback data while DOUT_BUSY is Low after switching from a write to a read (both CSI_B and RDWR_B are Low). DOUT_BUSY must be monitored to determine when the readback data is valid.

When readback is initiated, and after BUSY is deasserted, a number of dummy words depending on the SelectMAP bus width are read prior to valid data behind present. Table 7-3 lists the dummy readback cycles for the three SelectMAP widths.

Table 7-3: **Readback DOUT_BUSY Latency (SelectMAP)**

	x8	x16	x32
Read to DOUT_BUSY Latency	3 clocks	3 clocks	3 clocks

Accessing Configuration Registers through the JTAG Interface

JTAG access to the Virtex-6 FPGA configuration logic is provided through the JTAG CFG_IN and CFG_OUT registers. The CFG_IN and CFG_OUT registers are not configuration registers, rather they are JTAG registers like Bypass and Boundary. Data shifted into the CFG_IN register go to the configuration packet processor, where they are processed in the same way commands from the SelectMAP interface are processed.

Readback commands are written to the configuration logic by going through the CFG_IN register; configuration memory is read through the CFG_OUT register. The JTAG state transitions for accessing the CFG_IN and CFG_OUT registers are described in Table 7-4.

Table 7-4: **Shifting in the JTAG CFG_IN and CFG_OUT Instructions**

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
1	Clock five *1*s on TMS to bring the device to the TLR state	X	1	5
2	Move into the RTI state	X	0	1
3	Move into the Select-IR state	X	1	2
4	Move into the Shift-IR State	X	0	2
5	Shift the first nine bits of the CFG_IN or CFG_OUT instruction, LSB first	111000101 (CFG_IN)	0	9
		111000100 (CFG_OUT)		

Table 7-4: **Shifting in the JTAG CFG_IN and CFG_OUT Instructions *(Cont'd)***

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
6	Shift the MSB of the CFG_IN or CFG_OUT instruction while exiting SHIFT-IR	1	1	1
7	Move into the SELECT-DR state	X	1	2
8	Move into the SHIFT-DR state	X	0	2
9	Shift data into the CFG_IN register or out of the CFG_OUT register while in SHIFT_DR, MSB first	X	0	X
10	Shift the LSB while exiting SHIFT-DR	X	1	1
11	Reset the TAP by clocking five 1s on TMS	X	1	5

Configuration Register Read Procedure (JTAG)

The simplest read operation targets a configuration register such as the COR0 or STAT register. Any configuration register with read access can be read through the JTAG interface, although not all registers offer read access. The procedure for reading the STAT register through the JTAG interface follows:

1. Reset the TAP controller.
2. Shift the CFG_IN instruction into the JTAG Instruction Register through the Shift-IR state. The LSB of the CFG_IN instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
3. Shift packet write commands into the CFG_IN register through the Shift-DR state:
   a. Write the Synchronization word to the device.
   b. Write one NOOP instruction to the device.
   c. Write the *read STAT register* packet header to the device.
   d. Write two dummy words to the device to flush the packet buffer.

   The MSB of all configuration packets sent through the CFG_IN register must be sent first. The LSB is shifted while moving the TAP controller out of the SHIFT-DR state.
4. Shift the CFG_OUT instruction into the JTAG Instruction Register through the Shift-IR state. The LSB of the CFG_OUT instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
5. Shift 32 bits out of the Status register through the Shift-DR state.
6. Reset the TAP controller.

Table 7-5: **Status Register Readback Command Sequence (JTAG)**

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
1	Clock five 1s on TMS to bring the device to the TLR state.	X	1	5
	Move into the RTI state.	X	0	1
	Move into the Select-IR state.	X	1	2
	Move into the Shift-IR state.	X	0	2

Table 7-5: **Status Register Readback Command Sequence (JTAG) *(Cont'd)***

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
2	Shift the first nine bits of the CFG_IN instruction, LSB first.	111000101 (CFG_IN)	0	9
	Shift the MSB of the CFG_IN instruction while exiting SHIFT-IR.	1	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2
3	Shift configuration packets into the CFG_IN data register, MSB first.	a: 0xAA995566 b: 0x20000000 c: 0x2800E001 d: 0x20000000 e: 0x20000000	0	159
	Shift the LSB of the last configuration packet while exiting SHIFT-DR.	0	1	1
	Move into the SELECT-IR state.	X	1	3
	Move into the SHIFT-IR state.	X	0	2
4	Shift the first nine bits of the CFG_OUT instruction, LSB first.	111000100 (CFG_OUT)	0	9
	Shift the MSB of the CFG_OUT instruction while exiting Shift-IR.	1	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2
5	Shift the contents of the STAT register out of the CFG_OUT data register.	0xSSSSSSSS	0	31
	Shift the last bit of the STAT register out of the CFG_OUT data register while exiting SHIFT-DR.	S	1	1
	Move into the Select-IR state.	X	1	3
	Move into the Shift-IR State.	X	0	2
6	Reset the TAP Controller.	X	1	5

The packets shifted in to the JTAG CFG_IN register are identical to the packets shifted in through the SelectMAP interface when reading the STAT register through SelectMAP.

Configuration Memory Read Procedure (1149.1 JTAG)

The process for reading configuration memory from the FDRO register through the JTAG interface is similar to the process for reading from other registers. However, additional steps are needed to accommodate frame logic. Configuration data coming from the FDRO register pass through the frame buffer, therefore the first frame of readback data is *dummy data* and should be discarded (refer to the FDRI and FDRO register description). The 1149.1 JTAG readback flow is recommended for most users.

1. Reset the TAP controller.

2. Shift the CFG_IN instruction into the JTAG Instruction Register. The LSB of the CFG_IN instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
3. Shift packet write commands into the CFG_IN register through the Shift-DR state:
   a. Write a dummy word to the device.
   b. Write the Synchronization word to the device.
   c. Write a NOOP instruction to the device.
   d. Write the RCRC command to the device.
   e. Write two dummy words to flush the packet buffer.
4. Shift the JSHUTDOWN instruction into the JTAG Instruction Register.
5. Move into the RTI state; remain there for 12 TCK cycles to complete the Shutdown sequence. The DONE pin goes Low during the Shutdown sequence.
6. Shift the CFG_IN instruction into the JTAG Instruction Register.
7. Move to the Shift-DR state and shift packet write commands into the CFG_IN register:
   a. Write a dummy word to the device.
   b. Write the Synchronization word to the device.
   c. Write a NOOP instruction to the device.
   d. Write the *write CMD register* header.
   e. Write the RCFG command to the device.
   f. Write the *write FAR register* header.
   g. Write the starting frame address to the FAR register (typically `0x0000000`).
   h. Write the *read FDRO register* Type-1 packet header to the device.
   i. Write a Type-2 packet header to indicate the number of words to read from the device.
   j. Write two dummy words to the device to flush the packet buffer.

   The MSB of all configuration packets sent through the CFG_IN register must be sent first. The LSB is shifted while moving the TAP controller out of the SHIFT-DR state.
8. Shift the CFG_OUT instruction into the JTAG Instruction Register through the Shift-DR state. The LSB of the CFG_OUT instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
9. Shift frame data from the FDRO register through the Shift-DR state.
10. Reset the TAP controller.

Table 7-6: **Shutdown Readback Command Sequence (JTAG)**

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
1	Clock five 1s on TMS to bring the device to the TLR state.	X	1	5
	Move into the RTI state.	X	0	1
	Move into the Select-IR state.	X	1	2
	Move into the Shift-IR State.	X	0	2

Table 7-6: **Shutdown Readback Command Sequence (JTAG) *(Cont'd)***

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
2	Shift the first nine bits of the CFG_IN instruction, LSB first.	111000101	0	9
	Shift the MSB of the CFG_IN instruction while exiting Shift-IR.	1	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2
3	Shift configuration packets into the CFG_IN data register, MSB first.	a: 0xFFFFFFFF b: 0xAA995566 c: 0x20000000 d: 0x30008001 e: 0x00000007 f: 0x20000000 g: 0x20000000	0	223
	Shift the LSB of the last configuration packet while exiting SHIFT-DR.	0	1	1
	Move into the SELECT-IR State.	X	1	3
	Move into the SHIFT-IR State.	X	0	2
4	Shift the first nine bits of the JSHUTDOWN instruction, LSB first.	111001101	0	9
	Shift the MSB of the JSHUTDOWN instruction while exiting SHIFT-IR.	1	1	1
5	Move into the RTI state; remain there for 12 TCK cycles.	X	0	12
	Move into the Select-IR state.	X	1	2
	Move into the Shift-IR State.	X	0	2
6	Shift the first nine bits of the CFG_IN instruction, LSB first.	111000101	0	9
	Shift the MSB of the CFG_IN instruction while exiting SHIFT-IR.	1	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2

Table 7-6: **Shutdown Readback Command Sequence (JTAG) *(Cont'd)***

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
7	Shift configuration packets into the CFG_IN data register, MSB first.	a: 0xFFFFFFFF b: 0xAA995566 c: 0x20000000 d: 0x30008001 e: 0x00000004 f: 0x30002001 g: 0x00000000 h: 0x28006000 i: 0x48024090 j: 0x20000000 k: 0x20000000	0	351
	Shift the LSB of the last configuration packet while exiting SHIFT-DR.	0	1	1
	Move into the SELECT-IR state.	X	1	3
	Move into the SHIFT-IR state.	X	0	2
8	Shift the first nine bits of the CFG_OUT instruction, LSB first.	111000100 (CFG_OUT)	0	9
	Shift the MSB of the CFG_OUT instruction while exiting Shift-IR.	1	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2
9	Shift the contents of the FDRO register out of the CFG_OUT data register.	…	0	number of readback bits – 1
	Shift the last bit of the FDRO register out of the CFG_OUT data register while exiting SHIFT-DR.	X	1	1
	Move into the Select-IR state.	X	1	3
	Move into the Shift-IR state.	X	0	2
10	End by placing the TAP controller in the TLR state.	X	1	3

Verifying Readback Data

The readback data stream contains configuration frame data that are preceded by one frame of pad data, as described in the Configuration Memory Read Procedure (SelectMAP). The readback stream does not contain any of the commands or packet information found in the configuration bitstream and no CRC calculation is performed during readback. The readback data stream is shown in Figure 7-3.

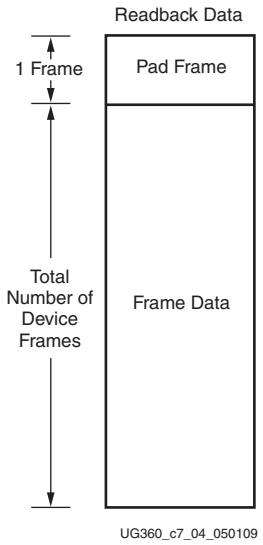


Figure 7-3: **Readback Data Stream**

The readback data stream is verified by comparing it to the original configuration frame data that were programmed into the device. Certain bits within the readback data stream must not be compared, because these can correspond to user memory or null memory locations. The location of *don't care* bits in the readback data stream is given by the mask files (`MSK` and `MSD`). These files have different formats although both convey essentially the same information. Once readback data have been obtained from the device, either of the following comparison procedures can be used:

1. Compare readback data to the `RBD` *golden* readback file. Mask by using the `MSD` file (see Figure 7-4).

   The simplest way to verify the readback data stream is to compare it to the `RBD` *golden* readback file, masking readback bits with the `MSD` file. This approach is simple because there is a 1:1 correspondence between the start of the readback data stream and the start of the `RBD` and `MSD` files, making the task of aligning readback, mask, and expected data easier.

   The `RBD` and `MSD` files contain an ASCII representation of the readback and mask data along with a file header that lists the file name, etc. This header information should be ignored or deleted. The ASCII 1s and 0s in the `RBD` and `MSD` files correspond to the binary readback data from the device. Take care to interpret these files as text, not binary sources. Users can convert the `RBD` and `MSD` files to a binary format using a script or text editor, to simplify the verify procedure for some systems and to reduce the size of the files by a factor of eight.

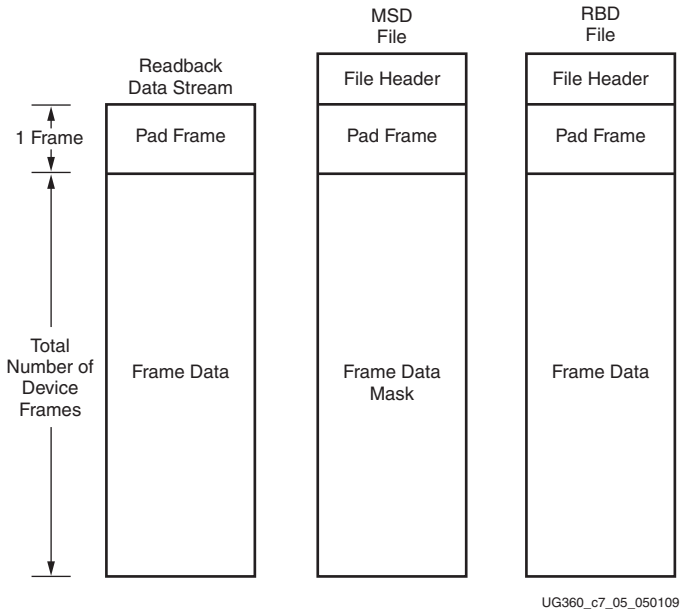


Figure 7-4: **Comparing Readback Data Using the MSD and RBD Files**

The drawback to this approach is that in addition to storing the initial configuration bitstream and the MSD file, the golden RBD file must be stored somewhere, increasing the overall storage requirement.

2. Compare readback data to the configuration BIT file, mask using the MSK file (see Figure 7-5).

   Another approach for verifying readback data is to compare the readback data stream to the frame data within the FDRI write in the original configuration bitstream, masking readback bits with the MSK file.

   After sending readback commands to the device, comparison begins by aligning the beginning of the readback frame data to the beginning of the FDRI write in the BIT and MSK files. The comparison ends when the end of the FDRI write is reached.

   This approach requires the least in-system storage space, because only the BIT, MSK, and readback commands must be stored.

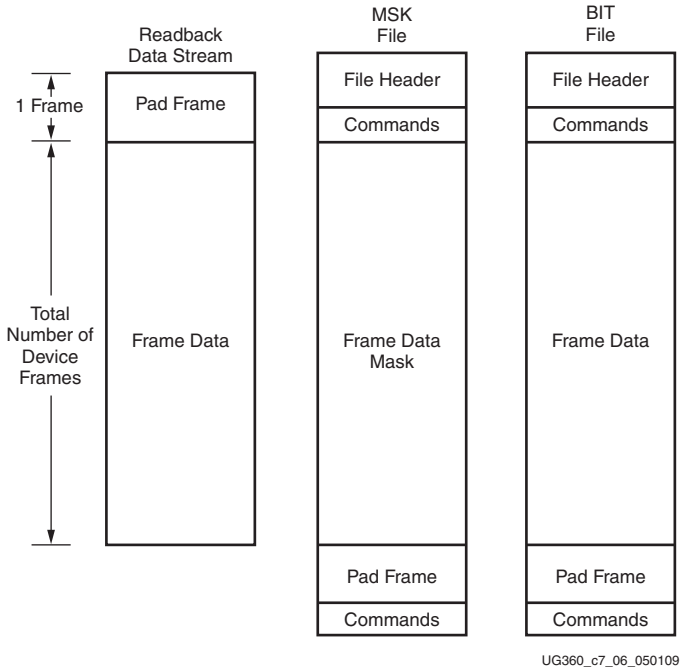


Figure 7-5: **Comparing Readback Data Using the MSK and BIT Files**

The RBA and RBB files contain expected readback data along with readback command sets. They are intended for use with the MSK file.

Readback Capture

The configuration memory readback command sequence is identical for both Readback Verify and Readback Capture. However, the Capture sequence requires an additional step to sample internal register values.

Users can sample block RAM outputs, and CLB and IOB registers by instantiating the CAPTURE_VIRTEX6 primitive in their design (Figure 7-6) and asserting the CAP input on that primitive while the design is operating. On the next rising clock edge on the CAPTURE_VIRTEX6 CLK input, the internal GRDBK signal is asserted, storing all CLB and IOB register values into configuration memory cells. These values can then be read out of the device along with the IOB and CLB configuration columns by reading configuration memory through the readback process. Register values are stored in the same memory cell that programs the register's initial state configuration, thus sending the GRESTORE command to the Virtex-6 FPGA configuration logic after the Capture sequence can cause registers to return to an unintended state.

Alternatively, the GRDBK signal can be asserted by writing the GCAPTURE command to the CMD register. This command asserts the GRDBK signal for two CCLK or TCK cycles, depending on the startup clock setting.

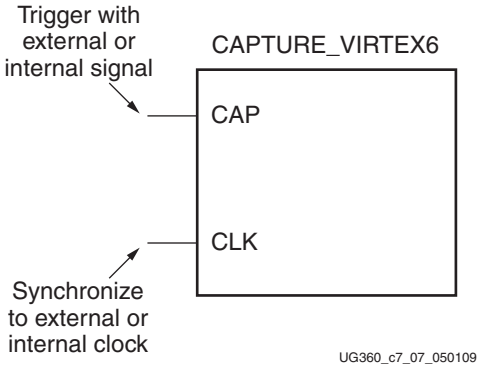


Figure 7-6: **Virtex-6 FPGA Library Primitive**

Table 7-7: **Capture Signals**

Signal	Description	Access
GCAPTURE	Captures the state of all slice and IOB registers. Complement of GRESTORE.	GCAPTURE command through the CMD register or *CAP* input on capture block, user controlled.
GRESTORE	Initializes all registers as configured.	CMD register and STARTUP_VIRTEX6 block.

If the CAP signal is left asserted over multiple clock cycles, the Capture cell is updated with the new register value on each rising clock edge. To limit the capture operation to the first rising clock edge, the user can add the ONESHOT attribute to the CAPTURE_VIRTEX6 primitive. More information on the ONESHOT attribute can be found in the Constraints Guide.

Once the configuration memory frames have been read out of the device, the user can pick the captured register values out of the readback data stream. The capture bit locations are given in the logic allocation file (`design.ll`) as described in Table 7-8.

Table 7-8 shows a snippet from a logic allocation file for the ISE software *jc2_top* example design. The line from the header comments explaining the line format has been moved to the start of the bit offset data for clarity. The Offset field gives the absolute bit offset from the beginning of the readback frame data. The Frame Address field gives the frame address that the capture bit is located in, and the Frame Offset field gives the bit offset from the start of the frame. The Information field gives the mapping between the bit and the user design—for example, the *DIR* register (Table 7-8) that is located in Slice X8Y15 is located at bit offset `100790`. Captured DFF values are stored in their inverted sense.

Table 7-8: **Logic Allocation File Format**

Offset	Frame Address	Frame Offset	Information
100714	0x000e0400	42	Block=B7, Latch=I, Net=RIGHT_IBUF
100734	0x000e0400	62	Block=A8, Latch=I, Net=RIGHT_IBUF
100754	0x000e0400	82	Block=B8, Latch=I, Net=RIGHT_IBUF
100790	0x000e0400	118	Block=SLICE_X8Y15, Latch=YQ, Net=DIR
119038	0x00100400	62	Block=C8, Latch=I, Net=STOP_IBUF

Table 7-8: **Logic Allocation File Format *(Cont'd)***

Offset	Frame Address	Frame Offset	Information
119132	0x00100400	156	Block=SLICE_X11Y14, Latch=YQ, Net=RUN
136566	0x00120200	118	Block=SLICE_X12Y15, Latch=XQ, Net=Q_3
136606	0x00120200	158	Block=SLICE_X12Y14, Latch=XQ, Net=Q_1
137300	0x00120400	20	Block=C9, Latch=O2, Net=Q_3
137320	0x00120400	40	Block=D9, Latch=O2, Net=Q_1
137398	0x00120400	118	Block=SLICE_X12Y15, Latch=YQ, Net=Q_2
137438	0x00120400	158	Block=SLICE_X12Y14, Latch=YQ, Net=Q_0
155662	0x00140400	78	Block=D10, Latch=O2, Net=Q_1
174046	0x00160400	158	Block=D13, Latch=O2, Net=Q_2

XILINX®

Chapter 8

Reconfiguration and MultiBoot

This chapter focuses on full bitstream reconfiguration methods introduced in the Virtex®-6 family.

Fallback MultiBoot

Fallback Overview

Virtex-6 FPGAs have dedicated MultiBoot logic, which is used for both fallback and warm boot (IPROG) reconfiguration. When fallback or IPROG happens, an internally generated pulse resets the entire configuration logic, except for the dedicated MultiBoot logic and the WBSTAR and BOOTSTS registers. This reset pulse pulls INIT_B and DONE Low, and restarts the configuration process by clearing configuration memory. See Setup (Steps 1-3), page 92. After the reset, the bitstream overwrites the WBSTAR starting address. See IPROG Reconfiguration, page 149 for details.

During fallback reconfiguration, the FPGA drives new values on the two dual-mode pins RS[1:0] (Revision Select). RS[1:0] are 3-stated and weakly pulled up during the first configuration, and weakly pulled down after configuration by default. The user can use external pull-up/pull-down resistors to override the FPGA weak pull-up resistor during first configuration to load the desired bitstream (see Figure 8-2, page 148). When a configuration error is detected, the configuration logic generates an internal reset pulse and actively drives RS[1:0] to `00` for loading the fallback bitstream in all configuration modes (Serial, SelectMAP, BPI-Up and SPI). One example of fallback reconfiguration is described in Fallback Example.

During configuration, the following errors can trigger fallback: an IDCODE error, a CRC error, a Watchdog Timer time-out error, or a BPI address wraparound error. After configuration, the Watchdog Timer, enabled in the user monitor mode (see User Monitor Mode), can also trigger fallback.

After successful fallback reconfiguration, the user design should readback the STATUS or BOOTSTS registers (see Status Register for Fallback and IPROG Reconfiguration) to verify the fallback was successful.

If fallback reconfiguration fails a second time, configuration stops and both INIT_B and DONE are held Low.

Fallback can also be disabled with BitGen option **`-g ConfigFallback:Disable`**.

Embedded IPROG (see IPROG Embedded in the Bitstream) is ignored during fallback reconfiguration. The Watchdog (see Watchdog) is disabled during and after fallback reconfiguration. A successful IPROG reconfiguration after fallback reconfiguration can re-enable the Watchdog.

Fallback Example

The simplest way to use the RS pins with a BPI flash is to connect them to the flash high address lines. In this example, the flash address space is divided into four sections, and each can be used to store one application bitstream (see Figure 8-1). The fallback bitstream (safe bitstream) must be stored at the RS[1:0] = 00 location. Refer to MultiBoot Bitstream Spacing for more information.

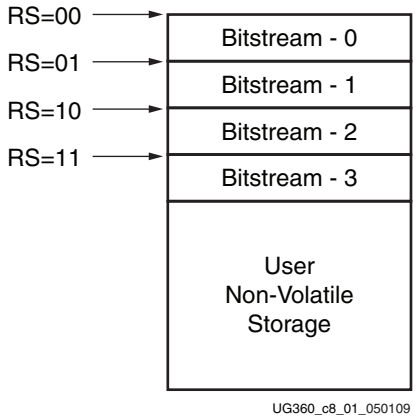


Figure 8-1: **BPI Flash Address Space for MultiBoot**

The initial bitstream can be selected using external resistors to override the internal weak pull-up resistor, as shown in Figure 8-2.

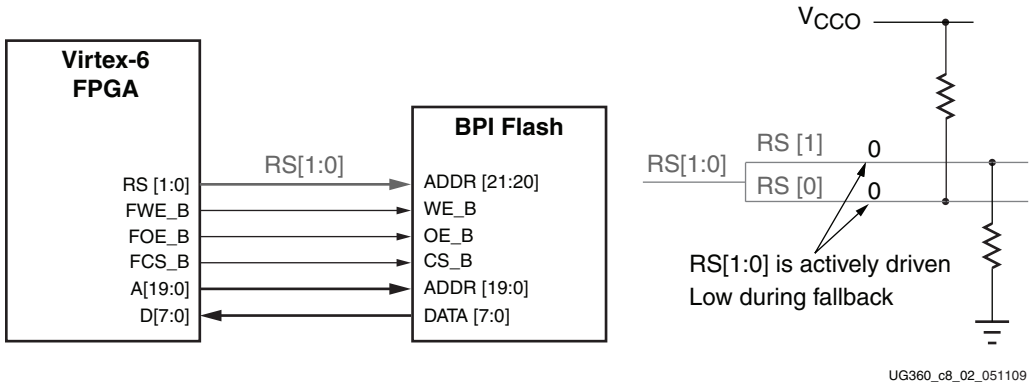


Figure 8-2: **Fallback Reconfiguration Usage for BPI**

Notes related to Figure 8-2:

1. Initial configuration is triggered by PROGRAM_B, JPROGRAM instruction, or POR.
2. RS[1:0] is 3-stated during initial configuration.
3. This example selects the initial bitstream using RS[1:0] = 01, as determined by the pull-up and pull-down resistors.
4. If initial configuration fails, RS[1:0] are actively driven Low:
   - For BPI-Up mode, A[25:0] starts from 0.
   - For BPI-Down mode, A[25:0] equals 0x3FFFFFF. A[21:20] are driven Low by the RS pins, and the resulting address for the fallback bitstream is 0x000FFFFF.
5. Configuration stops if the fallback bitstream fails a second time.

MultiBoot Bitstream Spacing

The Virtex-6 FPGA keeps loading the bitstream until the DONE pin goes High. When DCI lock is enabled before the DONE cycle, padding (all 0s or all 1s) is required between bitstreams to compensate for the total lock time. Otherwise, the following bitstream can potentially overwrite the previous bitstream. The DCI lock time is typically less than 1 ms. The formula for the padding size is cfg_bus_width x total_lock_time / CCLK_period.

IPROG Reconfiguration

The internal PROGRAM_B (IPROG) command has similar effect as a pulsing PROGRAM_B pin, except IPROG does not reset the dedicated reconfiguration logic. The start address set in WBSTAR (see Warm Boot Start Address Register (WBSTAR), page 124) is used during reconfiguration instead of the default address. The default is zero in BPI-Up and SPI modes, and the default is `0x3FFFFFF` in BPI-Down mode. The IPROG command can be sent through ICAP_VIRTEX6 or the bitstream. The IPROG Using ICAP_VIRTEX6 and IPROG Embedded in the Bitstream sections describe these two usages.

IPROG Using ICAP_VIRTEX6

The IPROG command can also be sent using the ICAP_VIRTEX6 primitive. After a successful configuration, the user design determines the start address of the next bitstream, and sets the WBSTAR register, and then issues an IPROG command using ICAP.

The sequence of commands are:

1. Send the Sync word.
2. Program the WBSTAR register for the next bitstream start address (see Warm Boot Start Address Register (WBSTAR), page 124).
3. Send the IPROG command.

Table 8-1 shows an example bitstream for the IPROG command using ICAP.

Table 8-1: **Example Bitstream for IPROG through ICAP**

Configuration Data (hex)	Explanation
FFFFFFFF	Dummy Word
AA995566	Sync Word
20000000	Type 1 NO OP
30020001	Type 1 Write 1 Words to WBSTAR
00000000	Warm Boot Start Address (Load the Desired Address)
30008001	Type 1 Write 1 Words to CMD
0000000F	IPROG Command
20000000	Type 1 NO OP

After the configuration logic receives the IPROG command, the FPGA resets everything except the dedicated reconfiguration logic, and the INIT_B and DONE pins go Low. After the FPGA clears all configuration memory, INIT_B goes High again. Then the value in WBSTAR is used for the bitstream starting address. The configuration mode determines which pins are controlled by WBSTAR.

Table 8-2: **WBSTAR Controlled Pins According to Configuration Mode**

Configuration Mode	Pins Controlled by WBSTAR
Master Serial	RS[1:0]
Master SPI	START_ADDR[23:0] are sent to the SPI device serially. The RS pins are set but not used.
Master BPI-Up	RS[1:0], A[25:0]
Master BPI-Down	RS[1:0], A[25:0]
Master SelectMAP	RS[1:0]
JTAG	RS[1:0]
Slave SelectMAP	RS[1:0]
Slave Serial	RS[1:0]

In all configuration modes, RS[1:0] is always controllable by WBSTAR. The START_ADDR field is only meaningful for the BPI-Up, BPI-Down, and SPI modes.

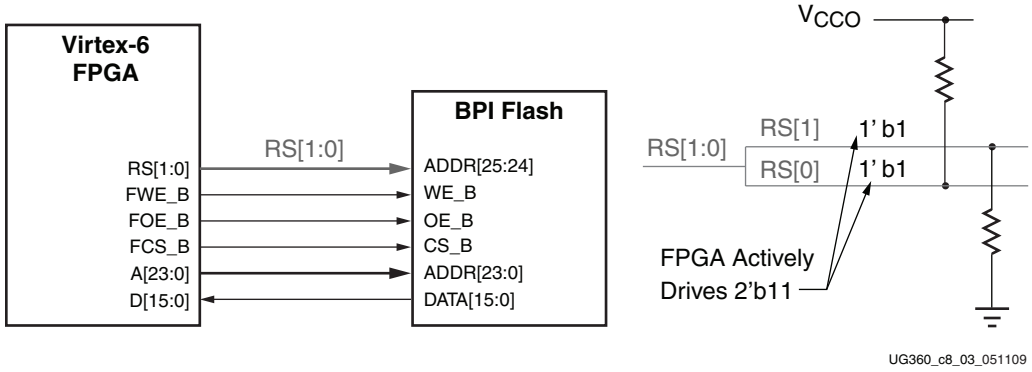


Figure 8-3: **IPROG in BPI Modes**

Notes relevant to Figure 8-3:

1. All BPI pins are dual mode I/Os. After configuration is DONE, these pins become user I/Os and can be controlled by user logic to access BPI flash for user data storage and programming.
2. In this example, RS[1:0] is set to `2'b11`. During IPROG reconfiguration, the RS[1:0] pins override the external pull-up and pull-down resistors. The user can specify any RS[1:0] value in the WBSTAR register.

IPROG Embedded in the Bitstream

WBSTAR and the IPROG command can be embedded inside a bitstream. A safe bitstream is stored at address 0 (in BPI-Up or SPI mode). Later a new application bitstream can be added to flash, simply by modifying the WBSTAR and the IPROG command in first bitstream. The FPGA directly loads the new bitstream. If the new bitstream fails, configuration falls back to the original bitstream (see Fallback MultiBoot). ISE software

inserts the blank write into WBSTAR and a place holder for the IPROG command in every Virtex-6 FPGA bitstream. For example, WBSTAR can be modified to a user-desired start address (see Warm Boot Start Address Register (WBSTAR), page 124). A NULL command after WBSTAR can be modified to IPROG by setting the four LSB bits to all ones (see Command Register (CMD), page 114).

Figure 8-4 illustrates this use model. Refer to MultiBoot Bitstream Spacing for more information.

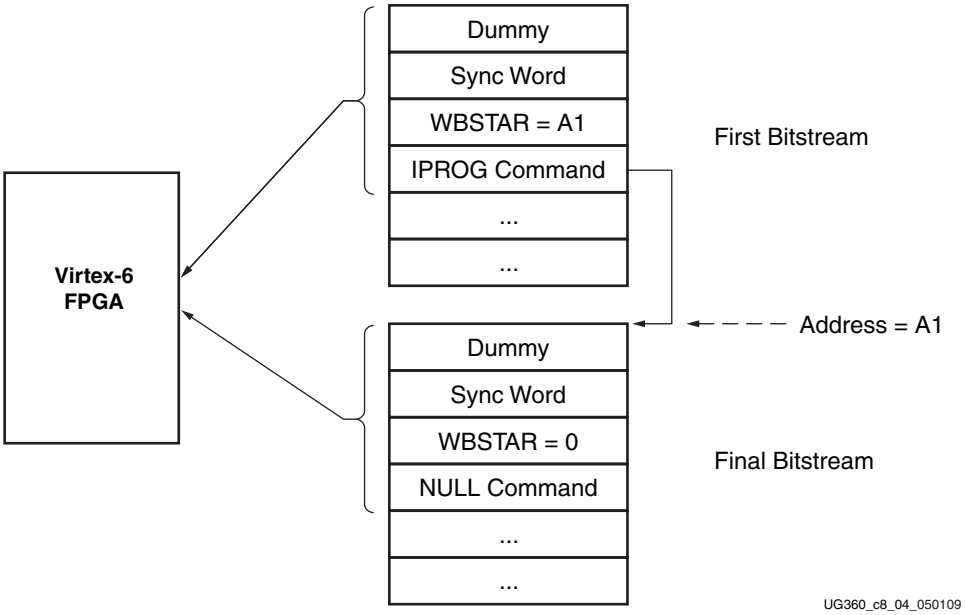


Figure 8-4: **IPROG Embedded in the Bitstream**

Status Register for Fallback and IPROG Reconfiguration

Virtex-6 devices contain a BOOTSTS that stores configuration history. BOOTSTS operates similar to a two-entry FIFO. The most recent configuration status is stored in Status_0, and the current value for Status_0 is shifted into Status_1. The Valid_0 bit indicates if the rest of Status_0 is valid or not. See Boot History Status Register (BOOTSTS), page 125.

Table 8-3 through Table 8-5 show the BOOTSTS values in some common situations.

Table 8-3: **Status after First Bitstream Configuration without Error**

	Reserved	WRAP_ERROR	CRC_ERROR	ID_ERROR	WTO_ERROR	IPROG	FALLBACK	VALID
Status_1	0	0	0	0	0	0	0	0
Status_0	0	0	0	0	0	0	0	1

Table 8-4: **First Configuration Followed by IPROG**

	Reserved	WRAP_ERROR	CRC_ERROR	ID_ERROR	WTO_ERROR	IPROG	FALLBACK	VALID
Status_1	0	0	0	0	0	0	0	1
Status_0	0	0	0	0	0	1	0	1

Table 8-5: **IPROG Embedded in First Bitstream, Second Bitstream CRC Error, Fallback Successfully**

	Reserved	WRAP_ERROR	CRC_ERROR	ID_ERROR	WTO_ERROR	IPROG	FALLBACK	VALID
Status_1	0	0	1	0	0	1	0	1
Status_0	0	0	0	0	0	1	1	1

Notes for Table 8-5:

1. Status_1 shows IPROG was attempted, and a CRC_ERROR was detected for that bitstream.
2. Status_0 shows a fallback bitstream was loaded successfully. The IPROG bit was also set in this case, because the fallback bitstream contains an IPROG command. Although the IPROG command is ignored during fallback, the status still records this occurrence.

Watchdog

The Virtex-6 FPGA Watchdog can be used to monitor configuration steps or user logic operation in the FPGA fabric. When the Watchdog times out, the configuration logic loads the fallback bitstream. The Fallback MultiBoot section provides more details.

The Watchdog uses a dedicated internal clock, CFG_MCLK, which has a nominal frequency of 50 MHz. The clock is predivided by 256, so that the Watchdog clock period is about 5120 ns. Given the watchdog counter is 24 bits wide, the maximum possible Watchdog value is about 86 seconds. The time value can be set via BitGen.

The Watchdog can be enabled in the bitstream or through any configuration port by writing to the TIMER register. The Watchdog is disabled during and after fallback reconfiguration. A successful IPROG reconfiguration initiated by a successful fallback reconfiguration is necessary to re-enable the Watchdog.

FPGA End of Startup

To use the Watchdog to monitor the bitstream configuration, set TIMER_CFG_MON to 1 and the desired TIMER_VALUE in a write to the TIMER register in the bitstream. The TIMER_VALUE should be adequate to cover the entire FPGA configuration time until startup is complete. Any wait time in startup for DCI match or DONE should also be included.

Once enabled, the watchdog timer starts to count down. If the timer reaches 0 and the FPGA has not reached the final state of startup, a watchdog time-out error occurs and triggers a fallback configuration.

User Monitor Mode

To use the Watchdog to monitor the user logic, set TIMER_USR_MON to 1 and the desired TIMER_VALUE in a write to the TIMER register in the bitstream. The user must constantly reset the watchdog counter before it times out, either by the LTIMER command or by directly accessing the TIMER register. The watchdog is automatically disabled when the device is shut down or on power down (including shutdown).

Table 8-6 shows an example bitstream for reloading the Watchdog using the LTIMER command.

Table 8-6: **Example Bitstream for Reloading the Watchdog with LTIMER**

Configuration Data (hex)	Explanation
FFFFFFFF	Dummy Word
AA995566	Sync Word
20000000	Type 1 NO OP
30008001	Type 1 Write 1 Words to CMD
00000000	NULL
20000000	Type 1 NO OP
30008001	Type 1 Write 1 Words to CMD
00000011	LTIMER Command
20000000	Type 1 NO OP
30008001	Type 1 Write 1 Words to CMD
0000000D	DESYNC
20000000	Type 1 NO OP

Table 8-7 shows an example bitstream for directly accessing the TIMER register:

Table 8-7: **Example Bitstream for Accessing the TIMER Register**

Configuration Data (hex)	Explanation
FFFFFFFF	Dummy Word
AA995566	Sync Word
20000000	Type 1 NO OP
30022001	Type 1 write 1 words to TIMER
00000000	TIMER value
20000000	Type 1 NO OP
30008001	Type 1 write 1 words to CMD
0000000D	DESYNC
20000000	Type 1 NO OP

XILINX®

Chapter 9

Readback CRC

Virtex®-6 devices include a feature to do continuous readback of configuration data in the background of a user design. This feature is aimed at simplifying detection of Single Event Upsets (SEUs) that cause a configuration memory bit to flip and can be used in conjunction with the FRAME ECC feature for advanced operations such as SEU corrections. To enable Readback CRC, the CONFIG user constraint POST_CRC is set to **Enable**. Once enabled, the configuration dedicated logic reads back continuously in the background to check the CRC of the configuration memory content. In the first round of readback, the ECC syndrome bits are calibrated. In the second round of readback, the CRC value is latched as the golden value for later comparison. The subsequent rounds of readback CRC value are compared against the golden value. When a single bit or double bit error is detected, ECCERROR is pulsed and the SYNDROME, SYNWORD, SYNBIT, ECCERRORSINGLE, and FAR information are presented. When a CRC mismatch is found, the CRCERROR pin of the FRAME_ECC_VIRTEX6 primitive is driven High. The INIT_B pin is then driven Low, and the DONE pin remains High. The CONFIG user constraint POST_CRC_INIT_FLAG can be optionally set to DISABLE to turn off INIT_B as the readback CRC flag. The error flag remains asserted until the next comparison if the error was not corrected. Readback CRC is halted and the error flag is cleared when the user logic accesses the configuration logic through an ICAP command, JTAG, or SelectMAP. When the user finishes accessing the configuration logic, readback CRC automatically resumes.

SEU Detection

Readback CRC logic runs under these conditions:

- Any configuration operation must finish with a DESYNC command to release the configuration logic. If a DESYNC command is not issued, the readback CRC logic cannot access the configuration logic and cannot run. The DESYNC command clears the CRC_ERROR flag.
- In addition, the JTAG instruction register (IR) must not contain any configuration instructions (CFG_IN, CFG_OUT, or ISC_ENABLE). When these instructions are present, at any time, the readback CRC logic can not access the configuration logic and cannot run. Any configuration operation performed via the JTAG interface should finish by loading the IR with a value other than these three configuration instructions.

The following dynamically changeable memory locations are masked during background readback:

- MLUT (RAM or SRL)
- Block RAM content is skipped during readback to avoid interfering with user functions. Block RAM is covered by its own ECC circuit during operation.
- Dynamic Reconfigure Port (DRP) memories are masked.

When enabled, the readback CRC logic automatically runs in the background after configuration is DONE, and when the following conditions hold:

- The FPGA is started up successfully, as indicated by the DONE pin going High.
- The configuration interface has been parked correctly. A normal bitstream has a DESYNC command at the end that signals to the configuration interface that it is no longer being used.
- If the JTAG interface is in use, the JTAG instruction register must not be set to CFG_IN, CFG_OUT, or ISC_ENABLE.

Readback CRC runs on different clock sources in different modes as indicated in Table 9-1.

Table 9-1: **Readback CRC Clock Sources**

ICAP Primitive	STARTUP Primitive	Master Modes	Slave Modes	JTAG Mode	Clock Source
Instantiated	x	x	x	x	CLK input of the ICAP primitive
Not Instantiated	Instantiated	x	x	x	USRCCLKO input of the STARTUP primitive
Not Instantiated	Not Instantiated	Yes	No	No	The Readback CRC clock source is the Master CCLK controlled by the BitGen option **`-g ConfigRate`** when Persist is enabled. Otherwise, the clock source is the internal oscillator with frequency constrained by the configuration constraint POST_CRC_FREQ.
Not Instantiated	Not Instantiated	No	Yes	No	The Readback CRC clock source is CCLK pin input when Persist is enabled. Otherwise, the source is the internal oscillator with frequency constrained by the configuration constraint POST_CRC_FREQ.
Not Instantiated	Not Instantiated	No	No	Yes	No clock (see paragraph below this table).

Because JTAG has the highest priority in the configuration mode, it takes over the configuration bus whenever it needs to. M[2:0] are recommended to be set to Master Serial mode when only JTAG configuration is intended, so that the internal oscillator provides a continuous clock. The JTAG Instruction Register must not be parked at the CFG_IN, CFG_OUT, or ISC_ENABLE instructions.

In a partial reconfiguration application, the configuration memory content changes, so the golden signature must be recalculated. The hardware golden CRC is automatically regenerated after any write to FDRI.

SEU Correction

If correction is enabled using the constraint POST_CRC_ACTION, then the readback CRC logic performs correction on single bit errors. During readback, the syndrome bits are calculated for every frame. If a single bit error is detected, the readback is stopped immediately. The frame in error is readback again, and using the syndrome information, the bit in error is fixed and written back to the frame. If the POST_CRC_ACTION is set to CorrectAndContinue, then the readback logic starts over from the first address. If the CorrectAndHalt option is set, the readback logic stops after correction.

Table 9-2 contains a list of different error scenarios and the corresponding behavior of the hardware correction logic when POST_CRC_ACTION is set.

Table 9-2: **POST_CRC_ACTION Summary**

POST_CRC_ACTION	Single Bit Error	Two or More Bit Errors in Different Frames (Single Bit Error per Frame)	Two or More Bit Errors in the Same Frame
HALT	• CRCERROR flag is asserted at the end of a full device scan. • INIT_B is asserted at the end of a full device scan. • ECCERROR flag is asserted while reading the frame with the error and is deasserted when the next frame is read. • RBCRC stops at end of a full device scan after the first error.		
CONTINUE	• CRCERROR flag is asserted at the end of a full device scan. • INIT_B is asserted at the end of a full device scan. • ECCERROR flag is asserted while reading the frame with the error and is deasserted when the next frame is read. • RBCRC continues from the next address.		
CorrectAndHalt	• CRCERROR flag is asserted while reading the frame with the error. • INIT_B is asserted while reading the frame with the error. • ECCERROR flag is asserted while reading the frame with the error and deasserted when the error is fixed. • RBCRC corrects single bit error and then stops.		
CorrectAndContinue	• CRCERROR flag is asserted while reading the frame with the error. • INIT_B is asserted while reading the frame with the error. • ECCERROR flag is asserted while reading the frame with the error and deasserted when the error is fixed. • CRCERROR/INIT_B returns High after a maximum of 447 clock cycles on the POST_CRC clock source (i.e., ICAP, CCLK). • RBCRC corrects the single bit error and restarts.	• CRCERROR flag and INIT_B are asserted at the end of a full device scan. • ECCERROR flag is asserted while reading the frame with errors. Built-in logic cannot correct more than one error in the frame so RBCRC continues onto the next frame and ECCERROR is deasserted if the next frame has no errors. • When RBCRC finishes the full scan and reaches the last address, it restarts from the starting address. The CRCERROR flag and INIT_B remain asserted and the ECCERROR flag is asserted again when reading the frame with errors.	

Post_CRC Constraints

There are several Virtex-6 FPGA constraints used for managing Single Event Upset (SEU) events. All constraints have the same propagation rule. They are placed as attributes on the CONFIG block, and then propagated to the physical design object.

POST_CRC

POST_CRC uses the FRAME_ECC_VIRTEX6 primitive's CRCERROR pin for signalling SEU events. In addition, INIT_B can be used as an SEU CRC error indicator by using the POST_CRC_INIT_FLAG constraint. During configuration, the INIT_B pin operates normally. After configuration, if SEU analysis is enabled, and INIT_B is enabled via the POST_CRC_INIT_FLAG, the INIT_B pin (default) serves as an SEU status pin. An SEU is detected when a comparison of the real-time computed CRC differs from the pre-computed CRC, the CRCERROR pin is driven High and the INIT_B pin is driven Low.

The POST_CRC constraint is the best way to convey this information. It attaches to the CONFIG constraint. POST_CRC can be used by PACE, PAR, and BitGen to reserve the INIT_B pin by not programming the IOB to drive the INIT_B pin.

POST_CRC can take two values:

- ENABLE
  SEU detection is enabled.
- DISABLE
  SEU detection is disabled (default).

POST_CRC_INIT_FLAG

POST_CRC_INIT_FLAG determines whether the Virtex-6 FPGA INIT_B pin is a source of the SEU error signal. Virtex-6 devices support the POST_CRC SEU detection mode.

The readback CRC feature compares a pre-computed CRC on the configuration bitstream or post-computed CRC against a CRC computed by internal logic based on periodic readback of the configuration memory cells. If a bit flips in the configuration memory cells, then an SEU is detected. Single bit flips are typically caused by background radiation.

POST_CRC_INIT_FLAG is used to disable the INIT_B pin as the readback CRC error status output pin. The error condition is still available from the FRAME_ECC_VIRTEX6 site. The best way to convey this information is to create the POST_CRC_INIT_FLAG constraint, which attaches to the CONFIG constraint. POST_CRC_INIT_FLAG can be used by BitGen to set a single bit in the COR1 register.

POST_CRC_INIT_FLAG can take two values:

- ENABLE
  Leave the INIT_B pin enabled as a source of the SEU error signal (default).
- DISABLE
  Disable the use of the INIT_B pin, with the FRAME_ECC site as the sole source of the SEU error signal.

POST_CRC_ACTION

POST_CRC_ACTION determines the behavior of the Readback CRC feature after a CRC error is detected.

POST_CRC_ACTION can take four values:

- CONTINUE
  Once a CRC error is detected, issue the error flag but continue to perform testing. (default)
- HALT
  Once a CRC error is detected, do not perform any further readback CRC testing.
- CorrectAndHalt
  If the CorrectAndHalt option is set, the readback logic stops after correction.
- CorrectAndContinue
  If the POST_CRC_ACTION is set to CorrectAndContinue, then the readback logic starts over from the first address.

POST_CRC_FREQ

POST_CRC_FREQ determines the frequency of the internally generated clock to the Readback CRC logic.

POST_CRC_FREQ can take these values:

NONE, 1, 2, 3, 6, 13, 25, 50

These values represent a nominal value for the internally generated clock frequency.

POST_CRC_SOURCE

POST_CRC_SOURCE determines the source of the CRC check. The CRC check can use a pre-computed value from software or a hardware calculated CRC by performing a readback. The default is the FIRST_READBACK option.

POST_CRC_SOURCE = [FIRST_READBACK|PRE_COMPUTED]

Syntax Examples

Table 9-3 lists the supported syntax examples for each constraint.

Table 9-3: **NCF and UCF Syntax Examples**

Constraint	Syntax Example
POST_CRC	`CONFIG POST_CRC = [ENABLE|DISABLE]`
POST_CRC_INIT_FLAG	`CONFIG POST_CRC_INIT_FLAG = [ENABLE|DISABLE]`
POST_CRC_ACTION	`CONFIG POST_CRC_ACTION = [CONTINUE|HALT|CorrectAndHalt|CorrectAndContinue]`
POST_CRC_FREQ	`CONFIG POST_CRC_FREQ = [NONE, 1, 2, 3, 6, 13, 25, 50]`
POST_CRC_SOURCE	`POST_CRC_SOURCE = [FIRST_READBACK| PRE_COMPUTED]`

Chapter 10

Advanced Configuration Interfaces

Serial Daisy Chains

Multiple Virtex®-6 devices can be configured from a single configuration source by arranging the devices in a serial daisy chain. In a serial daisy chain, devices receive their configuration data through their DIN pin, passing configuration data along to downstream devices through their DOUT pin. The device closest to the configuration data source is considered the most *upstream* device, while the device furthest from the configuration data source is considered the most *downstream* device.

In a serial daisy chain, the configuration clock is typically provided by the most upstream device in Master Serial mode. All other devices are set for Slave Serial mode. Figure 10-1 illustrates this configuration.

Another alternative is to use SPI mode for the first device. The daisy chain data is still sent out through DOUT in SPI mode.

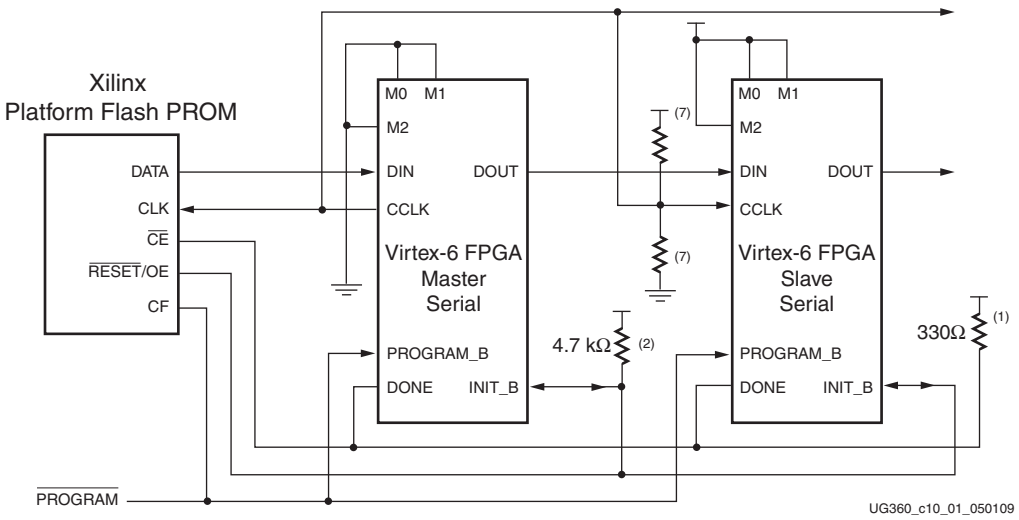


Figure 10-1: **Master/Slave Serial Mode Daisy Chain Configuration**

Notes relevant to Figure 10-1:

1. The DONE pin is by default an open-drain output requiring an external pull-up resistor. For all devices except the first, the active driver on DONE must be disabled. For the first device in the chain, the active driver on DONE can be enabled. See Guidelines and Design Considerations for Serial Daisy Chains.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.

3. The BitGen startup clock setting must be set for CCLK for serial configuration.
4. The PROM in this diagram represents one or more Xilinx® PROMs. Multiple Xilinx PROMs can be cascaded to increase the overall configuration storage capacity.
5. The `BIT` file must be reformatted into a PROM file before it can be stored on the Xilinx PROM. Refer to the Generating PROM Files, page 89 section.
6. On some Xilinx PROMs, the reset polarity is programmable. $\overline{\text{RESET}}$ should be configured as active Low when using this setup.
7. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines.
8. Serial daisy chains are specific to the Platform Flash XCFxxP PROM only.

The first device in a serial daisy chain is the last to be configured. CRC checks only include the data for the current device, not for any others in the chain. (See Cyclic Redundancy Check (Step 7) in Chapter 6.)

After the last device in the chain finishes configuration and passes its CRC check, it enters the Startup sequence. At the *Release DONE pin* phase in the Startup sequence, the device places its DONE pin in a High-Z state while the next to the last device in the chain is configured. After all devices release their DONE pins, the common DONE signal is either pulled High externally or driven High by the first device in the chain. On the next rising CCLK edge, all devices move out of the *Release DONE pin* phase and complete their startup sequences.

It is important that all DONE pins in a Slave Serial daisy chain be connected. Only the first device in the serial daisy chain should have the DONE active pull-up driver enabled. Enabling the DONE driver on downstream devices causes contention on the DONE signal.

Mixed Serial Daisy Chains

Virtex-6 devices can be daisy-chained with Virtex and Spartan® families. There are four important design considerations when designing a mixed serial daisy chain:

- Many older FPGA devices cannot accept as fast a CCLK frequency as a Virtex-6 device can generate. Select a configuration CCLK speed supported by all devices in the chain.
- Virtex-6 devices should always be at the beginning of the serial daisy chain, with older family devices located at the end of the chain.
- All Virtex device families have similar BitGen options. The guidelines provided for Virtex-6 FPGA BitGen options should be applied to all Virtex based devices in a serial daisy chain.
- The number of configuration bits that a device can pass through its DOUT pin is limited. This limit varies for different families (Table 10-1). The sum of the bitstream lengths for all downstream devices must not exceed the number in Table 10-1 for each family.

Table 10-1: **Maximum Number of Configuration Bits, Various Device Families**

Architecture	Maximum DOUT Bits
Virtex-6, Virtex-5, Virtex-4, Virtex-II Pro, and Virtex-II Devices	$32 \times (2^{27} - 1) = 4,294,967,264$
Spartan-3, Extended Spartan-3A, Spartan-3E, and Spartan-6 Devices	$16 \times (2^{27} - 1) = 2,147,483,632$
Virtex, Virtex-E, Spartan-II, and Spartan-IIE Devices	$32 \times (2^{20} - 1) = 33,554,216$

Guidelines and Design Considerations for Serial Daisy Chains

There are a number of important considerations for serial daisy chains:

Startup Sequencing (GTS)

GTS should be released before DONE or during the same cycle as DONE to ensure the Virtex-6 device is operational when all DONE pins have been released.

Active DONE Driver

All devices except the first should disable the driver on the DONE pin (refer to the BitGen section of UG628, *Command Line Tools User Guide* for software settings). The first device in a chain is programmed last:

- DriveDone is disabled (all devices except the first)
- DriveDone is enabled (first device)

Alternatively, the driver can be disabled for all DONE pins and an external pull-up resistor can be added to pull the signal High after all devices have released it.

Connect All DONE Pins

It is important to connect the DONE pins for all devices in a serial daisy chain. Failing to connect the DONE pins can cause configuration to fail. For debugging purposes, it is often helpful to have a way of disconnecting individual DONE pins from the common DONE signal, so that devices can be individually configured through the serial or JTAG interface.

DONE Pin Rise Time

After all DONE pins are released, the DONE pin should rise from logic 0 to logic 1 in one CCLK cycle. External pull-up resistors are required. If additional time is required for the DONE signal to rise, the BitGen **`DonePipe`** option can be set for all devices in the serial daisy chain. (Refer to the BitGen section of UG628, *Command Line Tools User Guide* for software settings.)

Ganged Serial Configuration

More than one device can be configured simultaneously from the same bitstream using a *ganged* serial configuration setup (Figure 10-2). In this arrangement, the serial configuration pins are tied together such that each device sees the same signal transitions. One device is typically set for Master Serial mode (to drive CCLK) while the others are set for Slave Serial mode. For ganged serial configuration, all devices must be identical.

Configuration can be driven from a configuration PROM or from an external configuration controller.

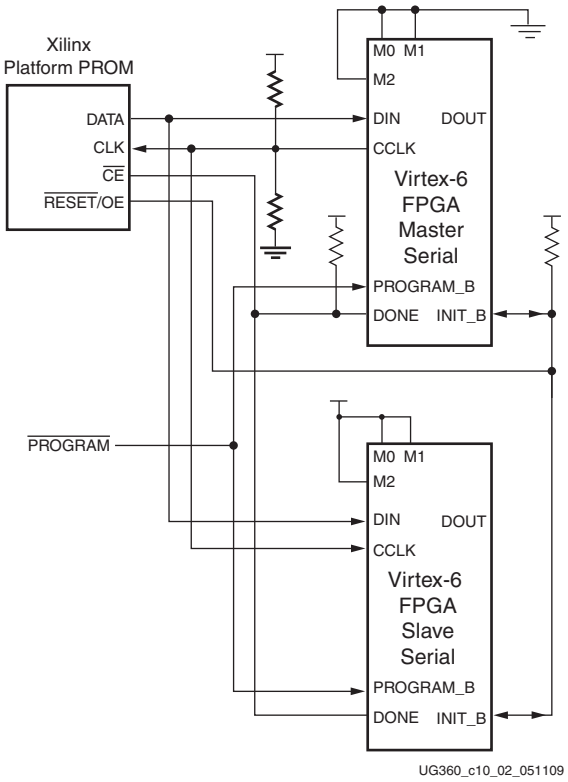


Figure 10-2: **Ganged Serial Configuration**

Notes relevant to Figure 10-2:

1. For ganged serial configuration, the optional DONE driver must be disabled for all devices if one device is set for Master mode because each device might not start up on exactly the same CCLK cycle. An external pull-up resistor is required in this case.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. The BitGen startup clock setting must be set for CCLK for serial configuration.
4. The PROM in this diagram represents one or more Xilinx PROMs. Multiple PROMs can be cascaded to increase the overall configuration storage capacity.
5. The `BIT` file must be reformatted into a PROM file before it can be stored on the PROM. Refer to Generating PROM Files, page 89.
6. On some Xilinx PROMs, the reset polarity is programmable. $\overline{\text{RESET}}$ should be configured as active Low when using this setup.
7. For ganged serial configuration, all devices must be identical (same IDCODE) and must be configured with the same bitstream.
8. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines.
9. Ganged serial configuration is specific to the Platform Flash XCFxxP PROM only.

There are a number of important considerations for ganged serial configuration:

- Startup Sequencing (GTS)

  GTS should be released before DONE or during the same cycle as DONE to ensure all devices are operational when all DONE pins have been released.

- Disable the Active DONE Driver for All Devices

  For ganged serial configuration, the active DONE driver must be disabled for all devices if the DONE pins are tied together, because there can be variations in the startup sequencing of each device. A pull-up resistor is therefore required on the common DONE signal.

  `-g DriveDone:no` (BitGen option, all devices)

- Connect all DONE pins if using a Master Device

  It is important to connect the DONE pins for all devices in ganged serial configuration if one FPGA is used as the Master device. Failing to connect the DONE pins can cause configuration to fail for individual devices in this case. If all devices are set for Slave Serial mode, the DONE pins can be disconnected (if the external CCLK source continues toggling until all DONE pins go High).

  For debugging purposes, it is often helpful to have a way of disconnecting individual DONE pins from the common DONE signal.

- DONE Pin Rise Time

  After all DONE pins are released, the DONE pin should rise from logic 0 to logic 1 in one CCLK cycle. If additional time is required for the DONE signal to rise, the BitGen **`DonePipe`** option can be set for all devices in the serial daisy chain.

- Configuration Clock (CCLK) as Clock Signal for Board Layout

  The CCLK signal is relatively slow, but the edge rates on the Virtex-6 FPGA input buffers are very fast. Even minor signal integrity problems on the CCLK signal can cause the configuration to fail. (Typical failure mode: DONE Low and INIT_B High.) Therefore, design practices that focus on signal integrity, including signal integrity simulation with IBIS, are recommended.

- Signal Fanout

  Designers must focus on good signal integrity when using ganged serial configuration. Signal integrity simulation is recommended.

- PROM Files for Ganged Serial Configuration

  PROM files for ganged serial configuration are identical to the PROM files used to configure single devices. There are no special PROM file considerations.

Multiple Device SelectMAP Configuration

Multiple Virtex-6 devices in Slave SelectMAP mode can be connected on a common SelectMAP bus (Figure 10-3). In a SelectMAP bus, the DATA, CCLK, RDWR_B, BUSY, PROGRAM_B, DONE, and INIT_B pins share a common connection between all of the devices. To allow each device to be accessed individually, the CSI_B (Chip Select) inputs must not be tied together. External control of the CSI_B signal is required and is usually provided by a microprocessor or CPLD.

If Readback is going to be performed on the device after configuration, the RDWR_B and BUSY signals must be handled appropriately. (For details, refer to Chapter 7, Readback and Configuration Verification.)

Otherwise, RDWR_B can be tied Low and BUSY can be ignored. Unlike earlier Virtex devices, the BUSY signal never needs to be monitored when configuring Virtex-6 devices. Refer to Bitstream Loading (Steps 4-7) in Chapter 6 and to Chapter 7, Readback and Configuration Verification.

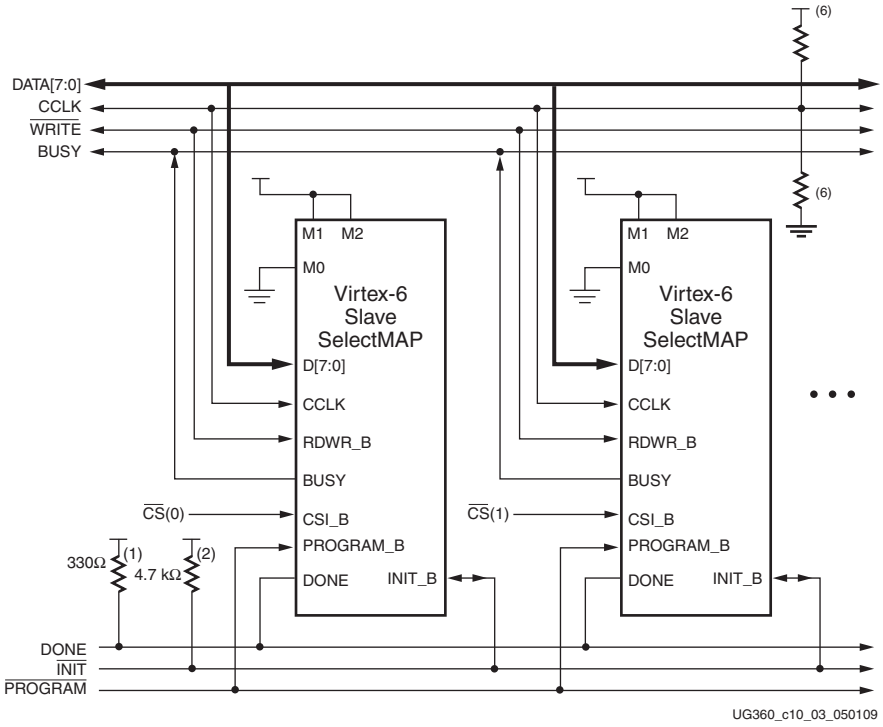


Figure 10-3: **Multiple Slave Device Configuration on an 8-Bit SelectMAP Bus**

Notes relevant to Figure 10-3:

1. The DONE pin is by default an open-drain output requiring an external pull-up resistor. In this arrangement, the active DONE driver must be disabled.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. The BitGen startup clock setting must be set for CCLK for SelectMAP configuration.
4. The BUSY signals can be left unconnected if readback is not needed.
5. An external controller such as a microprocessor or CPLD is needed to control configuration.
6. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines.
7. The data bus can be x8, x16, or x32 (for Slave SelectMAP).

Parallel Daisy Chain

Virtex-6 FPGA configuration supports parallel daisy-chain. Figure 10-4 shows an example schematic of the leading device in BPI mode. The leading device can also be in Master or Slave SelectMAP modes. The D[15:0], CCLK, RDWR_B, PROGRAM_B, DONE, and INIT_B pins share a common connection between all of the devices. The CSI_B pins are daisy chained.

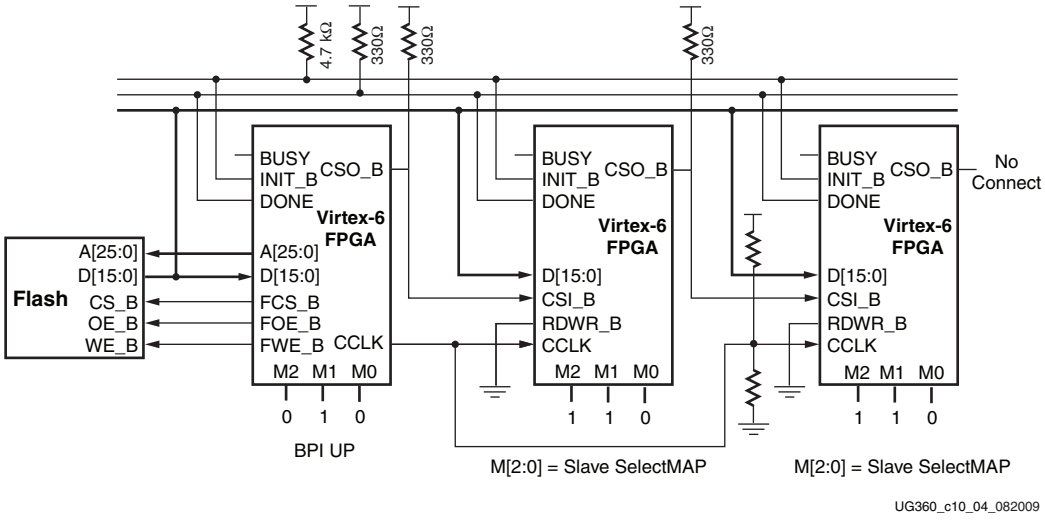


Figure 10-4: **Parallel Daisy Chain**

Notes relevant to Figure 10-4:

1. The DONE pin is by default an open-drain output requiring an external pull-up resistor. In this arrangement, the active DONE driver must be disabled.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up is required.
3. The BitGen startup clock setting must be set for CCLK for SelectMAP configuration.
4. The BUSY signals can be left unconnected if readback is not needed.
5. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines.
6. The FCS_B, FWE_B, FOE_B, CSO_B weak pull-up resistors should be enabled, otherwise external pull-up resistors are required for each pin. By default, all dual-mode I/Os have weak pull-downs after configuration.
7. The first device in the chain can be Master SelectMAP, Slave SelectMAP, BPI-Up, or BPI-Down.
8. Readback in the parallel daisy chain scheme is currently not supported.
9. AES decryption is not available in x16 or x32 mode, only in x8 mode when the BitGen **SecAll** setting is OFF (default).
10. Fallback MultiBoot is not supported in this configuration.

Ganged SelectMAP

It is also possible to configure simultaneously multiple devices with the same configuration bitstream by using a ganged SelectMAP configuration. In a ganged SelectMAP arrangement, the CSI_B pins of two or more devices are connected together (or tied to ground), causing all devices to recognize data presented on the D pins.

All devices can be set for Slave SelectMAP mode if an external oscillator is available, or one device can be designated as the Master device, as illustrated in Figure 10-5.

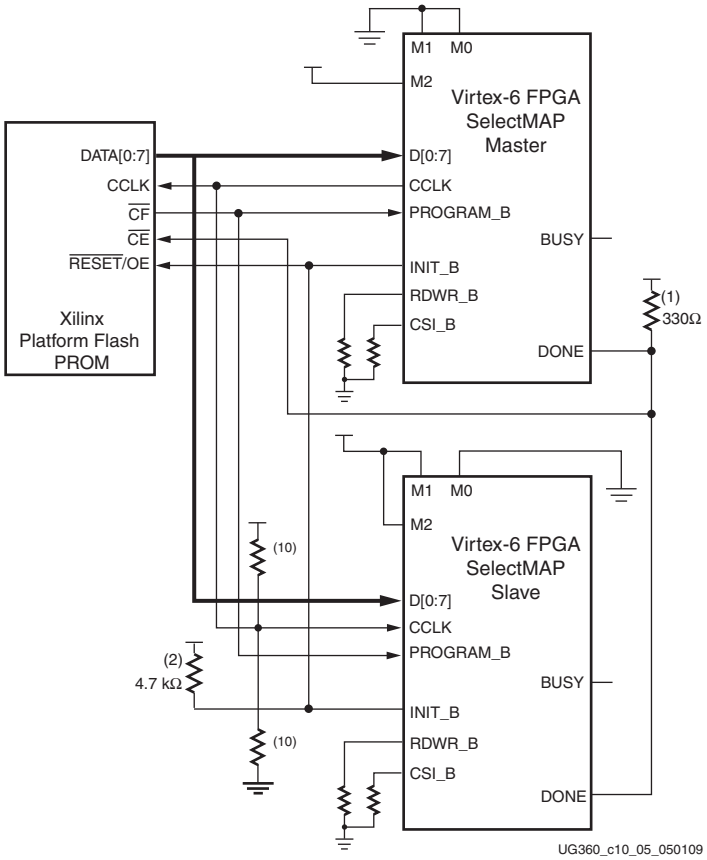


Figure 10-5: **Ganged x8 SelectMAP Configuration**

Notes relevant to Figure 10-5:

1. The DONE pin is by default an open-drain output requiring an external pull-up resistor. In this arrangement, the active DONE driver must be disabled for both devices.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. The BitGen startup clock setting must be set for CCLK for SelectMAP configuration.
4. The BUSY signal is not used for ganged SelectMAP configuration.
5. The PROM in this diagram represents one or more Xilinx PROMs. Multiple Xilinx PROMs can be cascaded to increase the overall configurations storage capacity.
6. The `BIT` file must be reformatted into a PROM file before it can be stored on the Xilinx PROM. Refer to the Generating PROM Files, page 89 section.

7. On some Xilinx PROMs, the reset polarity is programmable. Reset should be configured as active Low when using this setup.
8. The Xilinx PROM must be set for parallel mode. This mode is ***not*** available for all devices.
9. When configuring a Virtex-6 device in SelectMAP mode from a Xilinx configuration PROM, the RDWR_B and CSI_B signals can be tied Low (see SelectMAP Data Loading, page 38).
10. CCLK signal integrity is critical. See Board Layout for Configuration Clock (CCLK), page 58 for termination guidelines.
11. Ganged SelectMAP configuration is specific to the Platform Flash XCFxxP and Platform Flash XL PROMs.

If one device is designated as the Master, the DONE pins of all devices must be connected with the active DONE drivers disabled. An external pull-up resistor is required on the common DONE signal. Designers must carefully focus on signal integrity due to the increased fanout of the outputs from the PROM. Signal integrity simulation is recommended.

Readback is not possible if the CSI_B signals are tied together, because all devices simultaneously attempt to drive the Data signals.

SelectMAP ABORT

An ABORT is an interruption in the SelectMAP configuration or readback sequence occurring when the state of RDWR_B changes while CSI_B is asserted as sampled by CCLK. During a configuration ABORT, internal status is driven onto the D[7:4] pins over the next four CCLK cycles. The other D pins are always High. After the ABORT sequence finishes, the user can resynchronize the configuration logic and resume configuration. For applications that must deassert RDWR_B between bytes, see the Controlled CCLK method shown in Figure 2-10, page 42.

Configuration Abort Sequence Description

An ABORT is signaled during configuration as follows:

1. The configuration sequence begins normally.
2. The user pulls the RDWR_B pin High synchronous to CCLK while the device is selected (CSI_B asserted Low).
3. BUSY goes High if CSI_B remains asserted (Low). The FPGA drives the status word onto the data pins if RDWR_B remains set for read control (logic High).
4. The ABORT lasts for four clock cycles, and Status is updated.

See Figure 10-6.

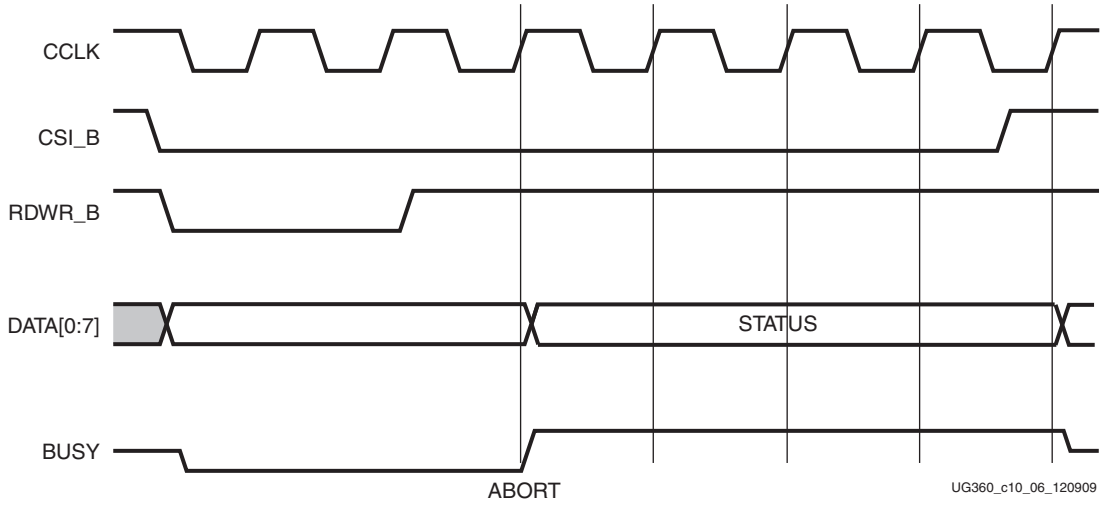


Figure 10-6: **Configuration Abort Sequence for SelectMAP Modes**

Readback Abort Sequence Description

An ABORT is signaled during readback as follows:

1. The readback sequence begins normally.
2. The user pulls the RDWR_B pin Low synchronous to CCLK while the device is selected (CSI_B asserted Low).
3. BUSY goes High if CSI_B remains asserted (Low).
4. The ABORT ends when CSI_B is deasserted.

See Figure 10-7.

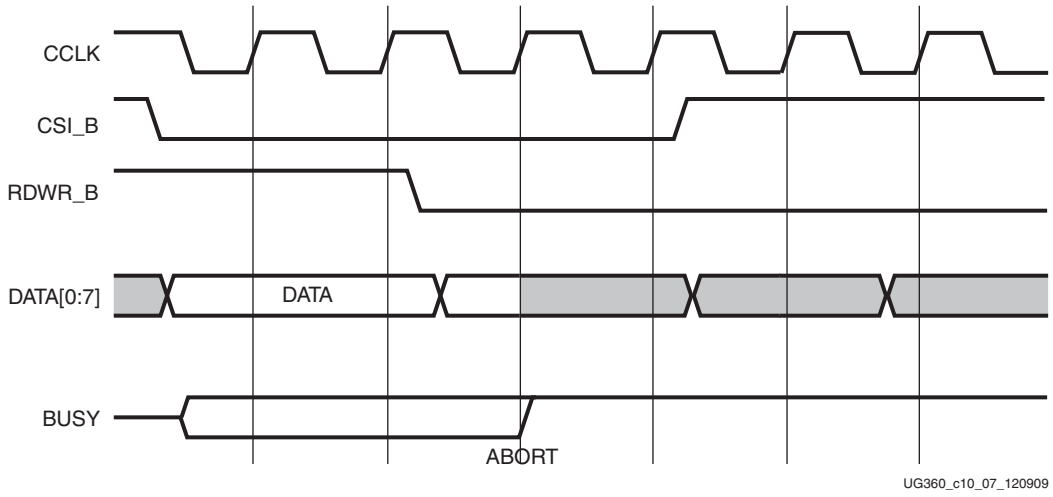


Figure 10-7: **Readback Abort Sequence**

ABORTs during readback are *not* followed by a status word because the RDWR_B signal is set for write control (FPGA D[x:0] pins are inputs).

ABORT Status Word

During the configuration ABORT sequence, the device drives a status word onto the D[7:0] pins. The status bits do not bit-swap. The other data pins are always High. The key for that status word is given in Table 10-2.

Table 10-2: **ABORT Status Word**

Bit Number	Status Bit Name	Meaning
D7	CFGERR_B	Configuration error (active Low) 0 = A configuration error has occurred. 1 = No configuration error.
D6	DALIGN	Sync word received (active High) 0 = No sync word received. 1 = Sync word received by interface logic.
D5	RIP	Readback in progress (active High) 0 = No readback in progress. 1 = A readback is in progress.
D4	IN_ABORT_B	ABORT in progress (active Low) 0 = Abort is in progress. 1 = No abort in progress.
D3-D0	1111	Fixed to ones.

The ABORT sequence lasts four CCLK cycles. During those cycles, the status word changes to reflect data alignment and ABORT status. A typical sequence might be:

```
11011111  => DALIGN = 1,  IN_ABORT_B = 1
10001111  => DALIGN = 0,  IN_ABORT_B = 0
10001111  => DALIGN = 0,  IN_ABORT_B = 0
10001111  => DALIGN = 0,  IN_ABORT_B = 0
```

After the last cycle, the synchronization word can be reloaded to establish data alignment.

Resuming Configuration or Readback After an Abort

There are two ways to resume configuration or readback after an ABORT:

- The device can be resynchronized after the ABORT completes.
- The device can be reset by pulsing PROGRAM_B Low at any time.

To resynchronize the device, CSI_B must be deasserted then reasserted. Configuration or readback can be resumed by sending the last configuration or readback packet that was in progress when the ABORT occurred. Alternatively, configuration or readback can be restarted from the beginning.

SelectMAP Reconfiguration

The term *reconfiguration* refers to reprogramming an FPGA after its DONE pin has gone High. Reconfiguration can be initiated by pulsing the PROGRAM_B pin (this method is identical to configuration) or by resynchronizing the device and sending configuration data.

To reconfigure a device in SelectMAP mode without pulsing PROGRAM_B, the BitGen **`persist`** option must be set—otherwise, the DATA pins become user I/O after configuration. The drive strength for the PERSIST pins is 12_SLOW. Reconfiguration must be enabled in BitGen. By default, the SelectMAP 8 interface (D0–D7) is preserved unless another SelectMAP width has been selected with the CONFIG_MODE constraint.

XILINX®

Chapter 11

Advanced JTAG Configurations

Introduction

Virtex®-6 devices support IEEE standards 1149.1 and 1149.6. JTAG is an acronym for the Joint Test Action Group, the technical subcommittee initially responsible for developing this standard. This standard ensures the board-level integrity of individual components and the interconnections between them. With multi-layer PC boards becoming increasingly dense and with more sophisticated surface mounting techniques in use, boundary-scan testing is becoming widely used as an important debugging tool.

Devices containing boundary-scan logic can send data out on I/O pins 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.

For BSDL compliance, HSWAPEN should be set to `0`. If HSWAPEN is set to `1`, then preconfiguration weak pull-up resistors are disabled.

JTAG Configuration/Readback

TAP Controller and Architecture

The Virtex-6 FPGA TAP contains four mandatory dedicated pins as specified by the protocol given in Table 3-1 and illustrated in Figure 11-1, a typical JTAG architecture.

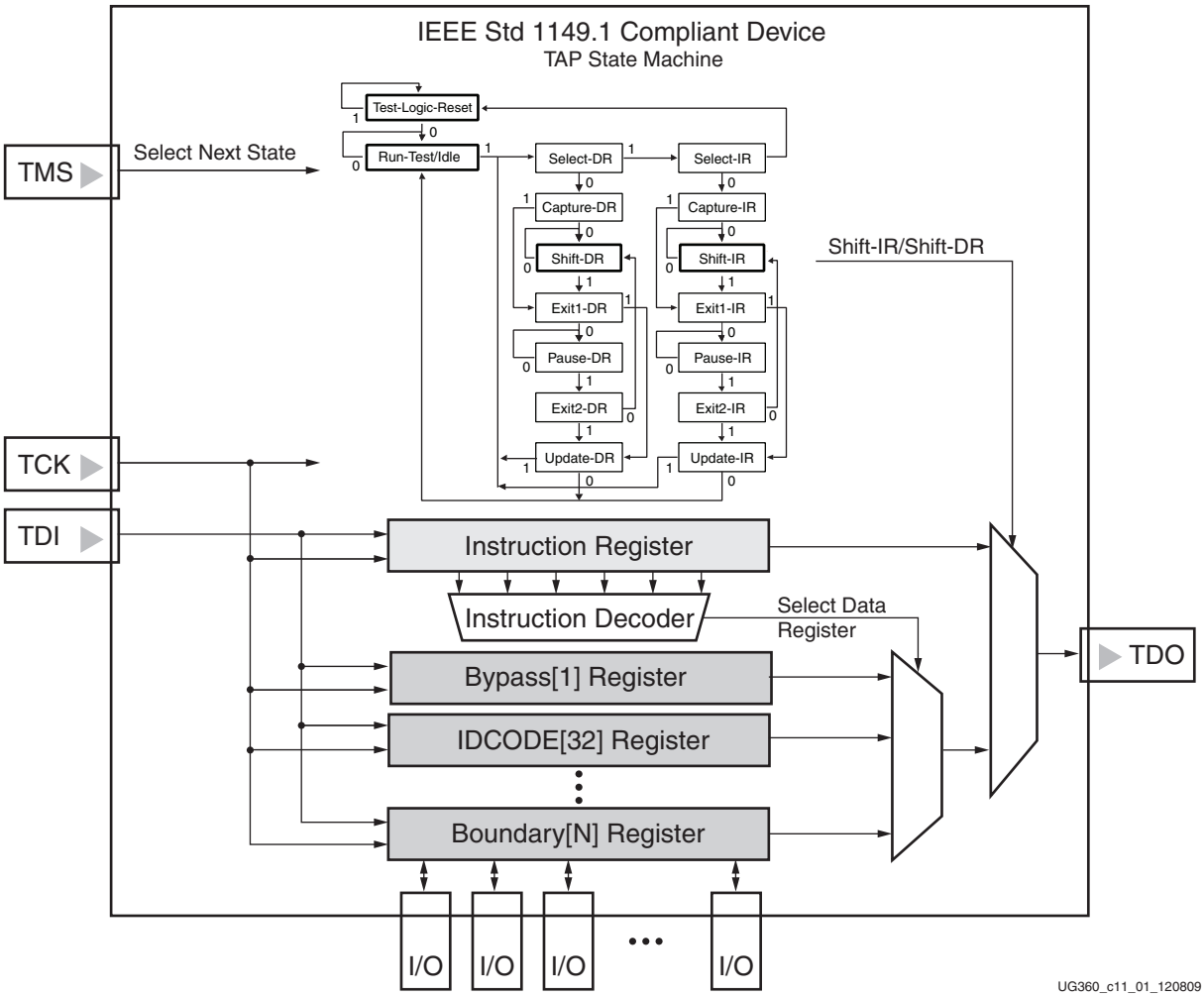


Figure 11-1: **Typical JTAG Architecture**

Figure 11-1 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.

A transition between the states only occurs on the rising edge of TCK, and each state has a different name. The two vertical columns with seven states each represent the Instruction Path and the Datapath. The data registers operate in the states whose names end with "DR," and the instruction register operates in the states whose names end in "IR." The states are otherwise identical.

The operation of each state is described below.

Test-Logic-Reset:
All test logic is disabled in this controller state, enabling the normal operation of the IC. The TAP controller state machine is designed so that regardless of the initial state of the controller, the Test-Logic-Reset state can be entered by holding TMS High and pulsing TCK five times. Consequently, the Test Reset (TRST) pin is optional.

Run-Test-Idle:
In this controller state, the test logic in the IC is active only if certain instructions are present. For example, if an instruction activates the self test, then it is executed when the controller enters this state. The test logic in the IC is idle otherwise.

Select-DR-Scan:
This controller state controls whether to enter the Datapath or the Select-IR-Scan state.

Select-IR-Scan:
This controller state controls whether or not to enter the Instruction Path. The controller can return to the Test-Logic-Reset state otherwise.

Capture-IR:
In this controller state, the shift register bank in the Instruction Register parallel loads a pattern of fixed values on the rising edge of TCK. The last two significant bits must always be `01`.

Shift-IR:
In this controller state, the instruction register gets connected between TDI and TDO, and the captured pattern gets shifted on each rising edge of TCK. The instruction available on the TDI pin is also shifted in to the instruction register.

Exit1-IR:
This controller state controls whether to enter the Pause-IR state or Update-IR state.

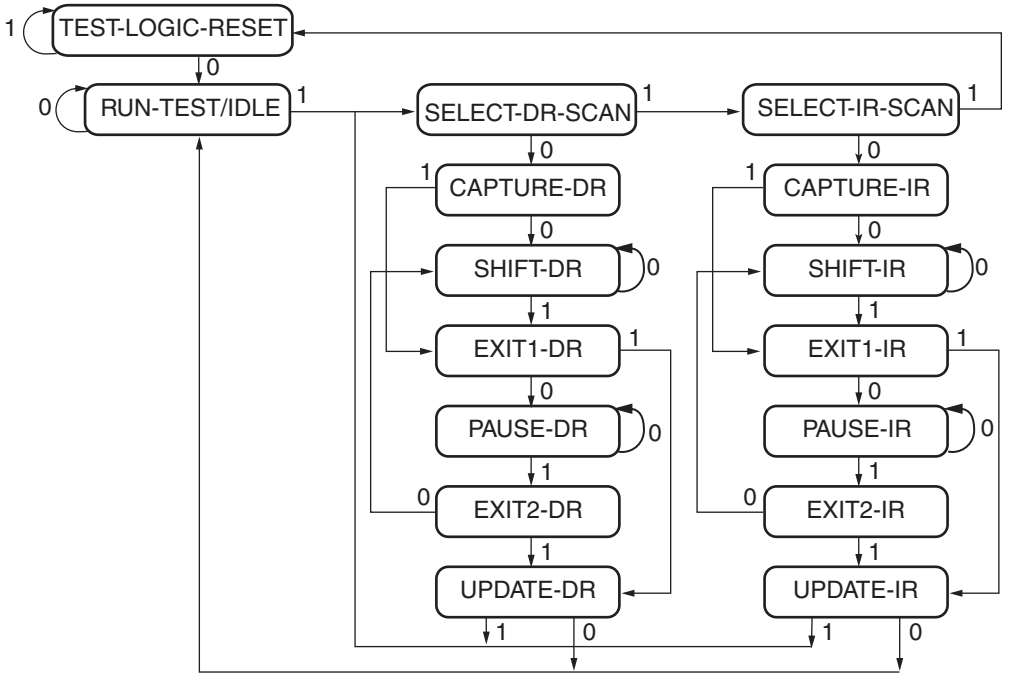
Pause-IR:
This state allows the shifting of the instruction register to be temporarily halted.

Exit2-DR:
This controller state controls whether to enter either the Shift-DR state or Update-DR state.

Update-IR:
In this controller state, the instruction in the instruction register is latched to the latch bank of the Instruction Register on every falling edge of TCK. This instruction becomes the current instruction after it is latched.

Capture-DR:
In this controller state, the data is parallel-loaded into the data registers selected by the current instruction on the rising edge of TCK.

Shift-Dr, Exit1-DR, Pause-DR, Exit2-DR, and Update-DR:
These controller states are similar to the Shift-IR, Exit1-IR, Pause-IR, Exit2-IR, and Update-IR states in the Instruction path.



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

UG360_c11_02_050109

Figure 11-2: **Boundary-Scan TAP Controller**

Virtex-6 devices support the mandatory IEEE Std 1149.1 commands as well as several Xilinx vendor-specific commands. The EXTEST, INTEST, SAMPLE/PRELOAD, BYPASS, IDCODE, USERCODE, and HIGHZ instructions are all included. The TAP also supports internal user-defined registers (USER1, USER2, USER3, and USER4) and configuration/readback of the device.

The Virtex-6 FPGA boundary-scan operations are independent of mode selection. The boundary-scan mode in Virtex-6 devices overrides other mode selections. For this reason, boundary-scan instructions using the Boundary register (SAMPLE/PRELOAD, INTEST, and EXTEST) must not be performed during configuration. All instructions except the user-defined instructions are available before a Virtex-6 device is configured. After configuration, all instructions are available.

JSTART and JSHUTDOWN are instructions specific to the Virtex-6 device architecture and configuration flow. In Virtex-6 devices, the TAP controller is not reset by the PROGRAM_B pin and can only be reset by bringing the controller to the TLR state. The TAP controller is reset on power up.

For details on the standard boundary-scan instructions EXTEST, INTEST, and BYPASS, refer to IEEE Std 1149.1.

Boundary-Scan Architecture

Virtex-6 device registers include all registers required by IEEE Std 1149.1. In addition to the standard registers, the family contains optional registers for simplified testing and verification (Table 11-1).

Table 11-1: **Virtex-6 FPGA JTAG Registers**

Register Name	Register Length	Description
Boundary Register	3 bits per I/O	Controls and observes input, output, and output enable.
Instruction Register	10 bits	Holds the current instruction opcode and captures internal device status. Refer to Table 11-3, page 179.
Bypass Register	1 bit	Bypasses the device.
Device Identification Register	32 bits	Captures the Device ID.
JTAG Configuration Register	32 bits	Allows access to the configuration bus when using the CFG_IN or CFG_OUT instructions.
USERCODE Register	32 bits	Captures the user-programmable code.
User-Defined Registers (USER1, USER2, USER3, and USER4)	Design specific	Design specific.

Boundary Register

The test primary data register is the Boundary register. Boundary-scan operation is independent of individual IOB configurations. Each IOB, bonded or unbonded, 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 11-1).

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.

Internal pull-up and pull-down resistors should be considered when test vectors are being developed for testing opens and shorts. The HSWAPEN pin determines whether the IOB has a pull-up resistor. Figure 11-3 is a representation of Virtex-6 FPGA boundary-scan architecture.

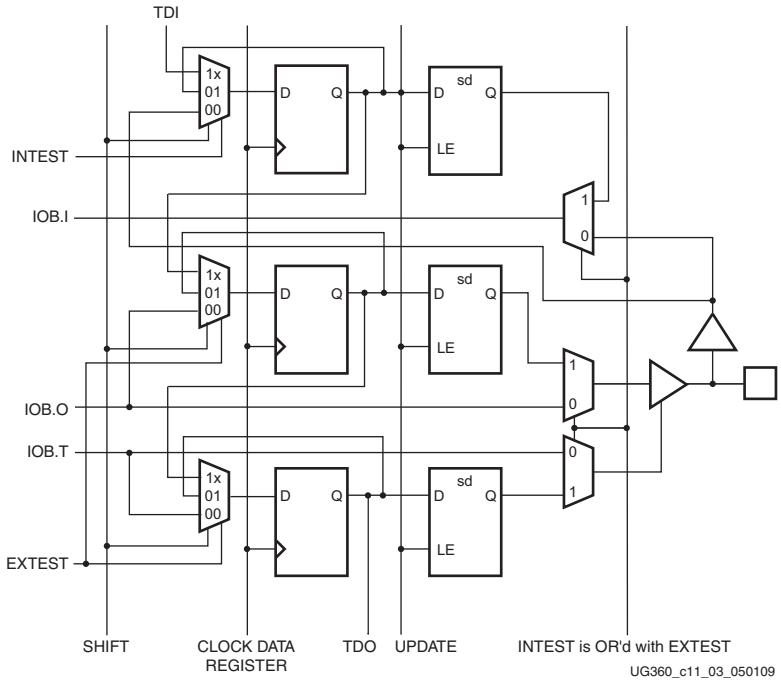


Figure 11-3: **Virtex-6 FPGA Boundary-Scan Logic**

Bit Sequence Boundary-Scan Register

This section describes the order of each non-TAP IOB. 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-6 family. (The BSDL files can be obtained from the Xilinx software download area and represent an unconfigured FPGA.) The bit sequence always has the same bit order and the same number of bits and is independent of the design.

For boundary-scan testing with a configured FPGA, Xilinx offers the BSDLAnno utility to automatically modify the BSDL file for post-configuration interconnect testing. The BSDLAnno utility obtains the necessary FPGA design information from the routed NCD file, and generates a BSDL file that reflects the post-configuration boundary-scan architecture of the device. Refer to the BSDLAnno chapter in UG628, *Command Line Tools User Guide*.

Instruction Register

The Instruction Register (IR) for the Virtex-6 device is connected between TDI and TDO during an instruction scan sequence. In preparation for an instruction scan sequence, the instruction register is parallel-loaded with a fixed instruction capture pattern. This pattern is shifted out onto TDO (LSB first), while an instruction is shifted into the instruction register from TDI.

To determine the operation to be invoked, an OPCODE necessary for the Virtex-6 FPGA boundary-scan instruction set is loaded into the Instruction Register. The IR is 10 bits wide for Virtex-6 devices. Table 11-2 lists the available instructions for Virtex-6 devices. See Table 6-22, page 109 for eFUSE-related JTAG instructions.

Table 11-2: **Virtex-6 FPGA Boundary-Scan Instructions**

Boundary-Scan Command	Binary Code [9:0]	Description
EXTEST	1111000000	Enables boundary-scan EXTEST operation.
SAMPLE	1111000001	Enables boundary-scan SAMPLE operation.
USER1	1111000010	Access user-defined register 1.
USER2	1111000011	Access user-defined register 2.
USER3	1111100010	Access user-defined register 3.
USER4	1111100011	Access user-defined register 4.
CFG_OUT	1111000100	Access the configuration bus for readback.
CFG_IN	1111000101	Access the configuration bus for configuration.
INTEST	1111000111	Enables boundary-scan INTEST operation.
USERCODE	1111001000	Enables shifting out user code.
IDCODE	1111001001	Enables shifting out of ID code.
HIGHZ	1111001010	3-state output pins while enabling the Bypass register.
JPROGRAM	1111001011	Equivalent to and has the same effect as PROGRAM.
JSTART	1111001100	Clocks the startup sequence when StartClk is TCK.
JSHUTDOWN	1111001101	Clocks the shutdown sequence.
SYSMON	1111110111	System Monitor DRP access through JTAG. See the DRP interface section in the *Virtex-6 FPGA System Monitor User Guide*.
ISC_ENABLE	1111010000	Marks the beginning of ISC configuration. Full shutdown is executed.
ISC_PROGRAM	1111010001	Enables in-system programming.
ISC_PROGRAM_KEY	1111010010	Change security status from secure to non-secure mode and vice versa.
ISC_NOOP	1111010100	No operation.
ISC_READ	1111010101	Used to read back the device configuration data and the battery-backed RAM (BBR). The BBR holds the encryption key for the AES bitstream encryption.
ISC_DISABLE	1111010111	Completes ISC configuration. Startup sequence is executed.
BYPASS	1111111111	Enables BYPASS.
RESERVED	All other codes	Xilinx reserved instructions.

Table 11-3 shows the instruction capture values loaded into the IR as part of an instruction scan sequence.

Table 11-3: **Virtex-6 FPGA Instruction Capture Values Loaded into IR as Part of an Instruction Scan Sequence**

TDI	IR[9:6]	IR[5]	IR[4]	IR[3]	IR[2]	IR[1:0]	→ TDO
	Reserved	DONE	INIT(1)	ISC_ENABLED	ISC_DONE	0 1	

Notes:

1. INIT is the status bit of the INIT_COMPLETE signal.

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.

Device Identification (IDCODE) Register

Virtex devices have a 32-bit identification register called the IDCODE register. The IDCODE is based on IEEE Std 1149.1 and is a fixed, vendor-assigned value that is used to identify electrically the manufacturer and the type of device that is being addressed. This register allows easy identification of the part being tested or programmed by boundary scan, and it can be shifted out for examination by using the IDCODE instruction.

The least significant bit of the IDCODE register is always `1` (based on JTAG IEEE 1149.1). The last three hex digits appear as `0x093`.

JTAG Configuration Register

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

USERCODE Register

The USERCODE instruction is supported in the Virtex-6 family. This register allows a user to specify a design-specific identification code. The USERCODE can be programmed into the device and can be read back for verification later. The USERCODE is embedded into the bitstream during bitstream generation (BitGen **`-g UserID`** option) and is valid only after configuration. If the device is blank or the USERCODE was not programmed, the USERCODE register contains `0xFFFFFFFF`.

USER1, USER2, USER3, and USER4 Registers

The USER1, USER2, USER3, and USER4 registers are only available after configuration. These four 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_VIRTEX6 library macro is required when creating these registers. This symbol is only required for driving internal scan chains (USER1, USER2, USER3, and USER4).

A common input pin (TDI) and shared output pins represent the state of the TAP controller (RESET, SHIFT, and UPDATE). Virtex-6 FPGA TAP pins are dedicated and do not require the BSCAN_VIRTEX6 macro for normal boundary-scan instructions or operations. For HDL, the BSCAN_VIRTEX6 macro must be instantiated in the design.

Using Boundary-Scan in Virtex-6 Devices

Characterization data for some of the most commonly requested timing parameters shown in Figure 11-5 are listed in the *Virtex-6 FPGA Data Sheet* in the Configuration Switching Characteristics table.

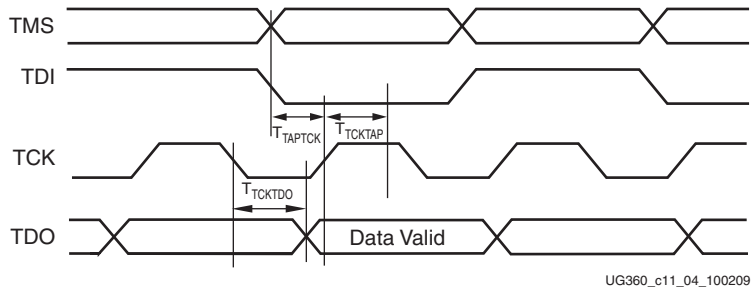


Figure 11-4: **Virtex-6 FPGA Boundary-Scan Port Timing Waveforms**

For further information on the startup sequence, bitstream, and internal configuration registers referenced here, refer to Configuration Sequence, page 92.

Configuring through Boundary-Scan

One of the most common boundary-scan vendor-specific instructions is the configure instruction. If the Virtex-6 device is configured via JTAG 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`).

The configuration flow for Virtex-6 device configuration with JTAG is shown in Figure 11-5. The sections that follow describe how the Virtex-6 device can be configured as a single device through the 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 PROGRAM_B pin or issuing the shut-down sequence. (See Figure 11-5.)

Designers who wish to implement the Virtex-6 FPGA JTAG configuration algorithm are encouraged to use the SVF-based flow provided in XAPP058, *Xilinx In-System Programming Using an Embedded Microcontroller* and XAPP424, *Embedded JTAG ACE Player*.

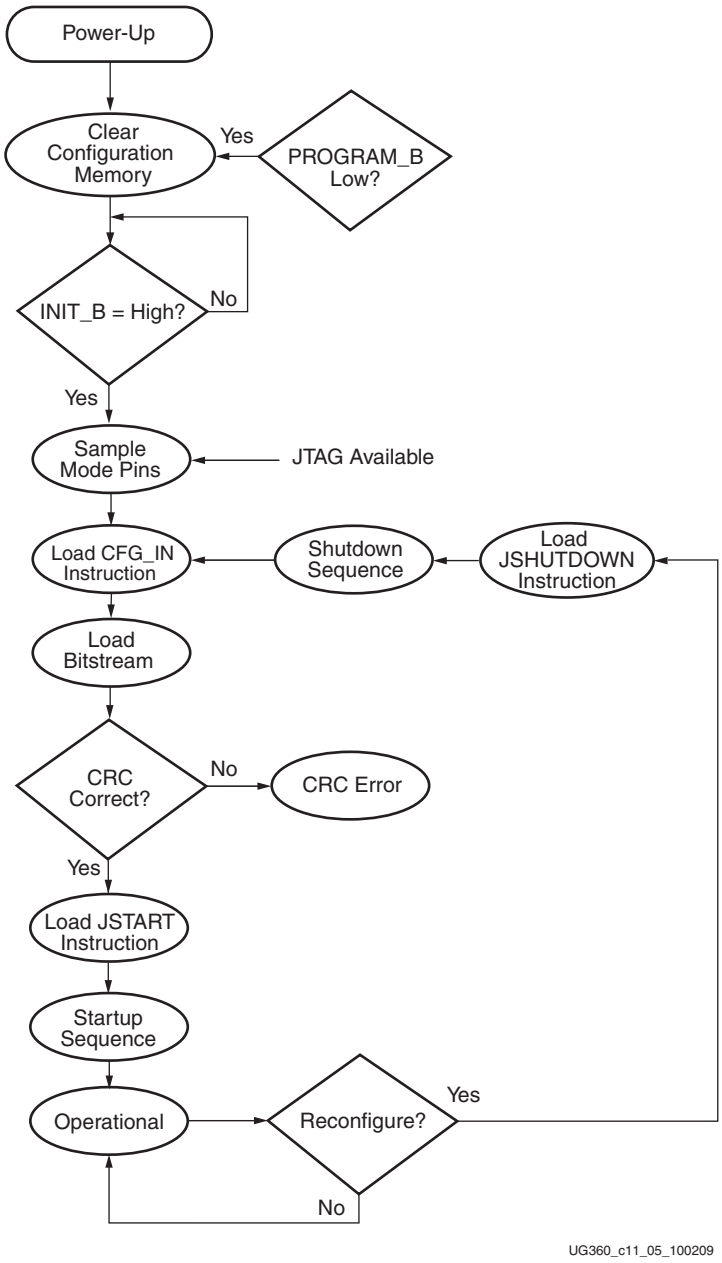


Figure 11-5: **Device Configuration Flow Diagram**

Single Device Configuration

Table 11-4 describes the TAP controller commands required to configure a Virtex-6 device. Refer to Figure 11-2, page 176 for TAP controller states. These TAP controller commands are issued automatically if configuring the part with the iMPACT software.

Table 11-4: **Single Device Configuration Sequence**

TAP Controller Step and Description		Set and Hold		# of Clocks
		TDI	TMS	TCK
1.	On power-up, place a logic 1 on the TMS, and clock the TCK five times. This ensures starting in the TLR (Test-Logic-Reset) state.	X	1	5
2.	Move into the RTI state.	X	0	1
3.	Move into the SELECT-IR state.	X	1	2
4.	Enter the SHIFT-IR state.	X	0	2
5.	Start loading the JPROGRAM instruction, LSB first:	111001011	0	9
6.	Load the MSB of the JPROGRAM instruction when exiting SHIFT-IR, as defined in the IEEE standard.	1	1	1
7.	Start loading the ISC_NOOP instruction, LSB first:	111010100	0	9
8.	Load the MSB of the ISC_NOOP instruction when exiting SHIFT-IR, as defined in the IEEE standard.	1	1	1
9.	Place a logic 1 on the TMS and clock the TCK five times. This ensures starting in the TLR (Test-Logic-Reset) state.	X	1	5
10.	Move into the RTI state.	X	0	10,000(1)
11.	Start loading the CFG_IN instruction, LSB first:	111000101	0	9
12.	Load the MSB of CFG_IN instruction when exiting SHIFT-IR, as defined in the IEEE standard.	1	1	1
13.	Enter the SELECT-DR state.	X	1	2
14.	Enter the SHIFT-DR state.	X	0	2
15.	Shift in the Virtex-6 FPGA bitstream. Bit_n (MSB) is the first bit in the bitstream(1).	$bit_1 ... bit_n$	0	(bits in bitstream)-1
16.	Shift in the last bit of the bitstream. Bit_0 (LSB) shifts on the transition to EXIT1-DR.	bit_0	1	1
17.	Enter UPDATE-DR state.	X	1	1
18.	Move into RTI state.	X	0	1
19.	Enter the SELECT-IR state.	X	1	2
20.	Move to the SHIFT-IR state.	X	0	2
21.	Start loading the JSTART instruction. The JSTART instruction initializes the startup sequence.	111001100	0	9
22.	Load the last bit of the JSTART instruction.	1	1	1
23.	Move to the UPDATE-IR state.	X	1	1

Table 11-4: **Single Device Configuration Sequence *(Cont'd)***

TAP Controller Step and Description		Set and Hold		# of Clocks
		TDI	TMS	TCK
24.	Move to the RTI state and clock the startup sequence by applying a minimum of 2000 clock cycles to the TCK.	X	0	2000
25.	Move to the TLR state. The device is now functional.	X	1	3

Notes:

1. At this RTI state, a minimum wait time of 10 ms is necessary. In this example, the TCK cycle value is based on a TCK frequency of 1 MHz.
2. In the Configuration Register, data is shifted in from the right (TDI) to the left (TDO), MSB first. (Shifts into the Configuration Register are different from shifts into the other registers in that they are MSB first.)
3. FPGAs that begin in an unconfigured state need to be reconfigured and require a preceding JPROGRAM sequence to clear the prior configuration or a JSHUTDOWN sequence to shutdown the FPGA. If JPROGRAM is used to reconfigure, the JPROGRAM needs to be loaded, followed by a BYPASS instruction. Then loop on loading the BYPASS instruction until the INIT bit becomes 1 before the CFG_IN instruction is sent.

Multiple Device Configuration

It is possible to configure multiple Virtex-6 devices in a chain. (See Figure 11-6.) The devices in the JTAG chain are configured one at a time. The multiple device configuration steps can be applied to any size chain.

Refer to the state diagram in Figure 11-1 for the following TAP controller steps:

1. On power-up, place a logic 1 on the TMS and clock the TCK five times. This ensures starting in the TLR (Test-Logic-Reset) state.
2. Load the CFG_IN instruction into the target device (and BYPASS in all other devices). Go through the RTI state (RUN-TEST/IDLE).
3. Load in the configuration bitstream per step 13 through step 17 in Table 11-4.
4. Repeat step 2 and step 3 for each device.
5. Load the JSTART command into all devices.
6. Go to the RTI state and clock TCK 2000 times.

All devices are active at this point.

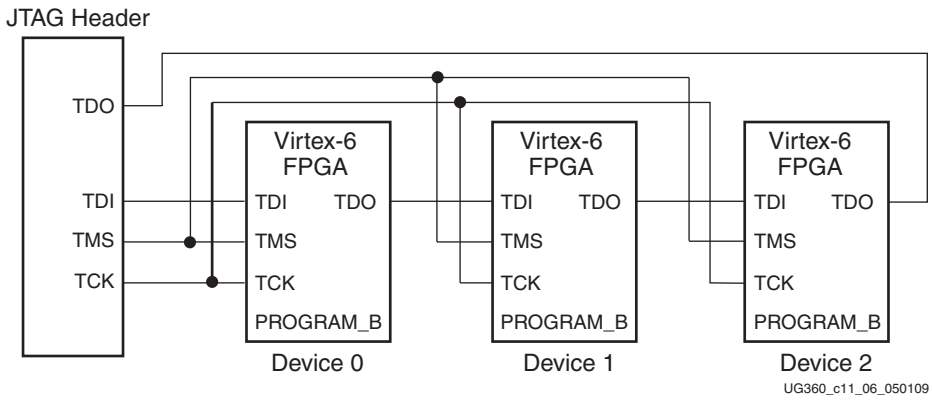


Figure 11-6: **Boundary-Scan Chain of Devices**