7 Series FPGAs Configuration

User Guide

UG470 (v1.10) June 24, 2015

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/01/2011	1.0	Initial Xilinx release.
03/28/2011	1.1	Changed name of "New Features" section to 7 Series FPGA Features, added note to first bullet, and added last sentence to fourth bullet. Revised Design Considerations section for clarity: Added Configuration Bitstream Lengths section and Table 1-1. Added Configuration Pins section and Table 2-2, Table 2-3, and Table 2-4. Moved Configuration Banks Voltage Select section from Chapter 1 to Chapter 2 and added Table 2-9. Added signal CFGBVS to Figure 2-2, Figure 2-5, Figure 2-12, Figure 2-16 and Figure 2-19.

Date	Version	Revision
10/26/2011	1.2	Chapter 1, Configuration Overview: • Changed VCC_CONFIG to VCCO_0 • Added Virtex-7 family to Table 1-1 • Added Stacked Silicon Interconnect section Chapter 2, Configuration Interfaces: • Corrected pin names D[04-07] and D[08-15] in Table 2-2 • Added a Note in Table 2-4 describing the function of the DONE pin • Added cross-reference to the DONE pin in Table 2-4 to the following: Table 2-7, Figure 2-2 note 2, Table 2-8, Figure 2-5 note 3, Table 2-12, Figure 2-12 note 1, Figure 2-16 note 8, Figure 2-19 note 8 revised the title of Figure 2-17 • Updated the families that Master BPI synchronous read mode supports in the first sentence of the second paragraph in Synchronous Read Mode Support • Expanded the description of the BitGen -g BPI_sync_mode option in Synchronous Read Mode Support • Clarified the BitGen ConfigRate setting and revised the CCLK frequency in the first paragraph of, and revised the ADDR range bullet in Determining the Maximum Configuration Clock Frequency Chapter 5, Configuration Details: • Changed VCC_CONFIG to VCCO_0 • Clarified the BPI asynchronous and synchronous read modes • Revised the function description of the DONE pin in Startup (Step 8) and added cross-reference to the DONE pin in Table 2-4 • Updated the support for the BitGen DriveDone option in Table 5-12 note 2 • Clarified the description of the JTAG instruction register in JTAG Instructions • Clarified the description of WRAP_ERROR_1 and WRAP_ERROR_0 in Table 5-39 Chapter 8, Readback CRC: • Corrected names of clock source primitives ICAPE2 and STARTUPE2 in Table 8-1
02/03/2012	1.3	Revised Table 1-1. Added Init_B, DONE, and CCLK pin names to Master SPI x4 column in Table 2-2. Added URL link to iMPACT Help documentation in Master SPI Configuration Mode. Added Determining the Maximum Configuration Clock Frequency. Added Table 5-17.

Date	Version	Revision
07/19/2012	1.4	Changed “ICAP” to “ICAPE2” throughout document. CFGBVS descriptions updated throughout document. Changed “4.7Ω“ pull-up/pull-down resistor value to “1 kΩ or greater” under Overview. Changed “7 Series Features” heading to 7 Series FPGAs Configuration Differences from Previous FPGA Generations. Under this heading changed “D00” in note to “D0”, clarified the fourth bullet, added the sixth and seventh bullets, clarified the eighth bullet, and added the last paragraph. Replaced Table 1-1. Clarified the second paragraph under Protecting the FPGA Bitstream against Unauthorized Duplication by removing the word “unique”. Added the last sentence under Loading Multiple FPGAs with the Same Configuration Bitstream. Clarified first two paragraphs under Stacked Silicon Interconnect. Clarified the descriptions of CFGBVS, TDO, PROGRAM_B, CCLK, PUDC_B, CSO_B, and DOUT in Table 2-4. Clarified Configuration Banks Voltage Select section and Table 2-5 and Table 2-9. Clarified description of PUDC_B in Table 2-7, Table 2-8, Table 2-12, and Table 2-15. Added “RS[1:0]” to Figure 2-4. Changed references from XAPP974 to XAPP586 and XAPP502 to XAPP583. Added last sentence to second paragraph and changed “flash timing” to “x1 mode sequence” under Master SPI Configuration Mode. Added note relevant to Figure 2-13. Added last paragraph under SPI Densities over 128 Mb. Added fourth and fifth paragraphs under Synchronous Read Mode Support. Added last paragraph under Configuring through Boundary-Scan. Added V_{CCBRAM} to first and last paragraphs, deleted last paragraph under Device Power-Up (Step 1). Modified Figure 5-4. Clarified definition of GWE in Table 5-12 and added table note 3. Changed “PROG” to PROGRAM_B” under Loading Encrypted Bitstreams. Clarified first paragraph under Bitstream Encryption and Internal Configuration Access Port (ICAPE2). Clarified bit position descriptions in Table 5-17 and associated text under eFUSE Control Register (FUSE_CNTL). Changed “7 Series FPGA Unique Device Identifier (Device DNA)” heading to Device Identifier and Device DNA, clarified first paragraph, and added second paragraph. Added the last two sentences to first paragraph under JTAG Access to Device DNA and Identifier. Clarified first paragraph and added fifth paragraph in Chapter 6, Readback and Configuration Verification. Added SPI 32-bit addressing mode support exception under Fallback MultiBoot. Changed “PROG” to PROGRAM_B” in first paragraph under IPROG. Updated address bits in Figure 7-3.
11/02/2012	1.5	Deleted XC7A350T, XC7V1500T, and XC7VH290T devices from Table 1-1. Changed configuration bitstream length (bits) for XC7VH580T and XC7VH870T devices in Table 1-1. Corrected bit value for RBCRC_EN in Table 5-33. Deleted Reset On Error, which is automatically enabled with the fallback feature, in Chapter 7, Reconfiguration and MultiBoot. Updated description for DIN and D[00-31] pins in Table 2-4. Deleted following tables: 7 Series FPGA Serial Configuration Interface Pins, 7 Series FPGA SelectMAP Configuration Interface Pins, 7 Series FPGA SPI Configuration Interface Pins, and 7 Series FPGA Master BPI Configuration Interface Pins. Updated paragraph five in Synchronous Read Mode Support. Updated bullets in Golden Image and MultiBoot Image Design Requirements and Initial MultiBoot Design Considerations
01/02/2013	1.6	Added reference to Vivado Design Suite (added last paragraph and note under 7 Series FPGAs Configuration Differences from Previous FPGA Generations). Simplified part numbers in Table 1-1. Corrected XC7V2000T device JTAG IDCODE (added Note 2 to Table 1-1). Changed cell heading in Table 5-1 from “Xilinx Software Tool” to “Xilinx Tool”. Highlighted limitation imposed by a specific eFUSE security option (added caution to first row in Table 5-17 and replaced second-to-last paragraph of eFUSE Control Register (FUSE_CNTL) with a caution).
10/22/2013	1.7	Added 7A35T, 7A50T, and 7A75T devices. Updated CFGBVS descriptions throughout document (CFGBVS determines supported I/O voltages in Banks 14 and 15 in Artix-7 and Kintex-7 devices). Removed references to fallback not being supported in SPI 32-bit addressing mode.

Date	Version	Revision
08/22/2014	1.8	Added Production IDCODE revision and additional Artix-7 devices to Table 1-1. Added Table 1-2. Added or updated links to related documents. Added Configuration Debugging in Chapter 1. Added SelectMAP ABORT in Chapter 2. Updated Configuration Banks Voltage Select in Chapter 2 and CFGBVS descriptions throughout document. Split Table 2-6 into three tables to separate FPGA families. Modified Figure 2-9. Added Setting Configuration Options in the Vivado Tools and External Master Configuration Clock (EMCCLK) Option in Chapter 2. Added notes to Figure 2-11. Added supported devices to Parallel NOR Flash families, Table 2-14. Clarified Providing Power in Chapter 3. Added Vivado TCL commands to Table 5-1. Added STARTUPE2 Primitive and associated startup details to STARTUPE2 Primitive in Chapter 5. Added Bitstream Composition in Chapter 5. Added Persist Option and Accessing Configuration Registers through the SelectMAP Interface in Chapter 6. Added Configuration Monitor Mode and Design Examples in Chapter 7. Added Chapter 9, Multiple FPGA Configuration and Chapter 10, Advanced JTAG Usage. Updated configuration details throughout document.
11/14/2014	1.9	Capitalized Tcl constraint commands. Added Artix-7 7A15T device to Table 1-1. Entered "2 or later" in place of TBD for Virtex-7 7VH580T and 7VH870T devices in Table 1-1. Added 7A15T device to Master BPI Configuration Interface in Chapter 2. Added last two sentences to step 9 under notes relevant to Figure 2-17. Changed Table 5-40 title from "Control Register 1 (CTL1)" to "BPI/SPI Configuration Options Register". Modified first sentence under RS Pins in Chapter 7. Added second sentence to step 5 under notes relevant to Figure 9-4. Made minor correction to Figure 10-3. Deleted commands 7 and 8 and added "optional" to step 19 in Table 10-4. Clarified note 3 under Table 10-4.
06/24/2015	1.10	Added note 1 to Table 2-1. Deleted Master SPI data for RS0 and RS1 in last row of Table 2-2 and clarified description in Table 2-4. Added recommendation to paragraph following Table 2-5. Clarified Table 2-6 Note 1, Table 2-7 Note 2, and Table 2-8 Note 2. Clarified external clock source speed in first paragraph under External Master Configuration Clock (EMCCLK) Option in Chapter 2. Added PROG_B and INIT_B waveforms to Figure 2-8. Clarified first paragraph under External Master Configuration Clock (EMCCLK) Option in Chapter 2. Added I/O Transition at the End of Startup in Chapter 5. Added fourth sentence under Bitstream Encryption and Internal Configuration Access Port (ICAPE2) in Chapter 5. Clarified FUSE_DNA data in Table 5-16. Clarified DONE_CYCLE, GTS_CYCLE, and GWE_CYCLE descriptions in Table 5-31. Clarified FUSE_DNA data in Table 5-16. Added third sentence under Configuration Memory Frames in Chapter 5. Clarified RS_TS_B description in Table 5-35. Deleted Note 2 in Table 5-41. Clarified bit composition in first paragraph under Device Identifier and Device DNA in Chapter 5. Changed "bit 63" to bit 0" under JTAG Access to Device DNA and Identifier in Chapter 5. Added fifth bullet under Initial MultiBoot Design Considerations in Chapter 7. Added START_ADDR bits and clarified Note 1 in Table 7-2. Deleted external pull-up resistor requirement for DONE pin in Chapter 9, Multiple FPGA Configuration. Clarified first paragraph following Figure 10-2 and HIGH_IO description in Table 10-2.

Table of Contents

About This Guide

Xilinx® 7 series FPGAs include three FPGA families that are all designed for lowest power to enable a common design to scale across families for optimal power, performance, and cost. The Artix®-7 family is optimized for lowest cost and absolute power for the highest volume applications. The Virtex®-7 family is optimized for highest system performance and capacity. The Kintex®-7 family is an innovative class of FPGAs optimized for the best price-performance. This guide serves as a technical reference describing the 7 series FPGAs configuration.

This *7 Series FPGAs Configuration User Guide* is part of an overall set of documentation on the 7 series FPGAs, which is available on the Xilinx website at www.xilinx.com/7.

Guide Contents

This manual contains these chapters:

- Chapter 1, Configuration Overview
- Chapter 2, Configuration Interfaces
- Chapter 3, Boundary-Scan and JTAG Configuration
- Chapter 4, Dynamic Reconfiguration Port (DRP)
- Chapter 5, Configuration Details
- Chapter 6, Readback and Configuration Verification
- Chapter 7, Reconfiguration and MultiBoot
- Chapter 8, Readback CRC
- Chapter 9, Multiple FPGA Configuration
- Chapter 10, Advanced JTAG Usage

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

This chapter provides a brief overview of the 7 series FPGA configuration methods and features. Subsequent chapters provide more detailed descriptions of each configuration method and feature. The configuration methods and features described herein are available on all family members with few exceptions.

Overview

Xilinx® 7 series FPGAs are configured by loading application-specific configuration data—a bitstream—into internal memory. 7 series 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 datapath 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 (x1, x2, x4)
- Master Byte Peripheral Interface (BPI) flash configuration mode (x8 and x16) using parallel NOR flash

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 (≤ 1 kΩ), or tied directly to ground or V_{CCO_0}. The mode pins should not be toggled during and after configuration. 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 7 series device drives CCLK from an internal oscillator. To select the desired frequency, the bitstream **`-g ConfigRate`** option is used. The BitGen section of UG628, *ISE Command Line Tools User Guide* provides more information for the ISE Design Suite. The Device Configuration Bitstream Settings section of UG908, *Vivado Programming and Debugging User Guide* provides more information for the Vivado Design Suite. After configuration, the CCLK is turned OFF unless the persist option is selected or SEU detection is used. See Persist Option in Chapter 6. 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.

7 Series FPGAs Configuration Differences from Previous FPGA Generations

The 7 series devices support the same configuration interfaces supported on Virtex®-6 FPGAs except for the Master BPI-Down mode. Master BPI-Down mode is not supported in the 7 series FPGAs. In addition, a few of the configuration interfaces are enhanced with these features that enable faster configuration:

- The Master SPI configuration mode supports reading from an SPI flash using a data bus up to four bits wide, which is similar to the Spartan®-6 FPGA Master SPI configuration mode.

  Note: In the 7 series, the DIN pin function is assigned to a multi-function pin that shares the D01 configuration data bus pin in order to support the x2 or x4 SPI data widths. This is different from the Virtex-6 FPGA where DIN was a dedicated pin, and this is different from the Spartan-6 FPGA where DIN was assigned to the multi-purpose D0 configuration data bus pin.
- The Master SPI configuration mode supports clocking data on the negative edge, allowing for optimal use of the clock period and therefore faster configuration speed.
- The Master SPI configuration mode supports flash densities greater than 128 Mb.
- The Master BPI configuration mode supports reading from a BPI (parallel NOR) flash via the flash device's burst, synchronous read mode. The ADV_B pin is new relative to the Virtex-6 FPGA BPI interface to support the address latching required for the BPI synchronous read mode.
- The AES decryptor supports configuration data bus widths up to 16 bits wide.
- Relative to Virtex-6, the SelectMAP modes and ICAPE2 primitive do not have a BUSY pin/port. BUSY is not needed in the 7 series because the SelectMAP/ICAPE2 output data is deterministic (see Accessing Configuration Registers through the SelectMAP Interface, page 124.)
- See UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide* for the configuration and boundary scan components (primitives). The 7 series primitive names end with an "E2" suffix, whereas the Virtex-6 FPGA primitives ended with the "_VIRTEX6" suffix.

The 7 series devices support configuration interfaces with 3.3V, 2.5V, 1.8V, or 1.5V I/O. The configuration interfaces include the JTAG pins in bank 0, the dedicated configuration pins in bank 0, and the pins related to specific configuration modes in bank 14 and bank 15. To support the appropriate configuration interface voltage on bank 0, bank 14, and bank 15, the following is required:

- The configuration banks voltage select pin (CFGBVS) must be set to a High (V_{CCO_0}) or Low (GND) to set the configuration and JTAG I/O in banks 0, 14, and 15 for 3.3V/2.5V or 1.8V/1.5V operation, respectively. When CFGBVS is set to Low for 1.8V/1.5V I/O operation, the V_{CCO_0} supply and I/O signals to bank 0 must be 1.8V (or lower) to avoid device damage. If CFGBVS is Low, then any I/O pins used for configuration in banks 14 and 15 must also be powered and operated at 1.8V or 1.5V. See Configuration Banks Voltage Select, page 32 for further details.

The operating voltage of the I/O in bank 14 and bank 15 are determined by the V_{CCO_14} and V_{CCO_15} supplies, respectively. When bank 14 or bank 15 are used for configuration, the V_{CCO} supplies for the applicable banks should match the V_{CCO_0} voltage for voltage compatibility across the configuration interface. When CFGBVS is tied to GND for 1.8V/1.5V I/O operation, then if any configuration I/O are used in bank 14 or bank 15, VCCO_14 or VCCO_15 and the configuration I/O signals to bank 14 or bank 15 must be 1.8V or 1.5V to avoid device damage.

Most 7 series FPGAs are supported by both the ISE Design Suite, which also supports previous generations, and the newer Vivado Design Suite. The user options described in this user guide generally refer to the ISE Design Suite tool names, but the same options are found in the Vivado Design Suite. For example, the ISE Design Suite BitGen tool generates bitstreams. In Vivado, the WRITE_BITSTREAM Tcl command can be used. For more information, see:

- UG835, *Vivado Design Suite Tcl Command Reference Guide*
- UG908, *Vivado Design Suite User Guide: Programming and Debugging*

Note: The BitGen command options are Tcl properties in the Vivado Design Suite. See Appendix A, *Device Configuration Bitstream Settings*, in UG908 for details on the properties and values.

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 multi-function pins during configuration only. These multi-function pins are then released for general use when configuration is completed. 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.

Configuration Bitstream Lengths

FPGA designs are compiled into bitstreams. The bitstreams are loaded through a configuration interface to configure the FPGA with the design. A complete bitstream for each FPGA part type has a fixed length. Table 1-1 shows the bitstream lengths and other device-specific information for the 7 series FPGAs.

Table 1-1: **Bitstream Length**

Device	Configuration Bitstream Length (bits)	Minimum Configuration Flash Memory Size (Mb)	JTAG/Device IDCODE[31:0] (hex)(1)	Production IDCODE Revision	JTAG Instruction Length (bits)	Super Logic Regions
Artix-7 Family						
7A15T	17,536,096	32	X362D093	0 or later	6	N/A
7A35T	17,536,096	32	X362D093	0 or later	6	N/A
7A50T	17,536,096	32	X362C093	0 or later	6	N/A
7A75T	30,606,304	32	X3632093	1 or later	6	N/A
7A100T	30,606,304	32	X3631093	1 or later	6	N/A
7A200T	77,845,216	128	X3636093	1 or later	6	N/A
Kintex-7 Family						
7K70T	24,090,592	32	X3647093	0 or later	6	N/A
7K160T	53,540,576	64	X364C093	0 or later	6	N/A
7K325T	91,548,896	128	X3651093	4 or later	6	N/A
7K355T	112,414,688	128	X3747093	0 or later	6	N/A
7K410T	127,023,328	128	X3656093	1 or later	6	N/A
7K420T	149,880,032	256	X3752093	2 or later	6	N/A
7K480T	149,880,032	256	X3751093	2 or later	6	N/A
Virtex-7 Family						
7V585T	161,398,880	256	X3671093	0 or later	6	N/A
7V2000T	447,337,216	512	X36B3093(2)	2 or later	24	4
7VX330T	111,238,240	128	X3667093	0 or later	6	N/A
7VX415T	137,934,560	256	X3682093	3 or later	6	N/A
7VX485T	162,187,488	256	X3687093	3 or later	6	N/A
7VX550T	229,878,496	256	X3692093	3 or later	6	N/A
7VX690T	229,878,496	256	X3691093	3 or later	6	N/A
7VX980T	282,521,312	512	X3696093	0 or later	6	N/A
7VX1140T	385,127,680	512	X36D5093	2 or later	24	4
7VH580T	195,663,008	256	X36D9093	2 or later	22	2
7VH870T	294,006,336	512	X36DB093	2 or later	38	3

Notes:

1. The 'X' in the JTAG IDCODE value represents the revision field (IDCODE[31:28]) which can vary.
2. The 7V2000T IDCODE contains additional don't care ('X') bits beyond the revision field. The complete binary IDCODE[31:0] value with don't care bit positions is: `XXXX_0011_0110_1011_XX11_0000_1001_0011`.

FPGA Configuration Data Source

Xilinx 7 series FPGAs are designed for maximum flexibility. The FPGA either automatically loads itself with configuration data from a nonvolatile flash memory, or another external intelligent device such as a processor or microcontroller can download the configuration data to the FPGA. In addition, the configuration data can be downloaded from a host computer through a cable to the JTAG port of the FPGA.

Master Modes

The self-loading FPGA configuration modes, generically called *Master* modes, are available with either a serial or parallel datapath. 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 generates a configuration clock signal in an internal oscillator that drives the configuration logic and is visible on the CCLK output pin. The FPGA controls the configuration process.

Slave Modes

The externally controlled loading FPGA configuration modes, generically called *Slave* modes, are also available with either a serial or parallel 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 1-1. 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.

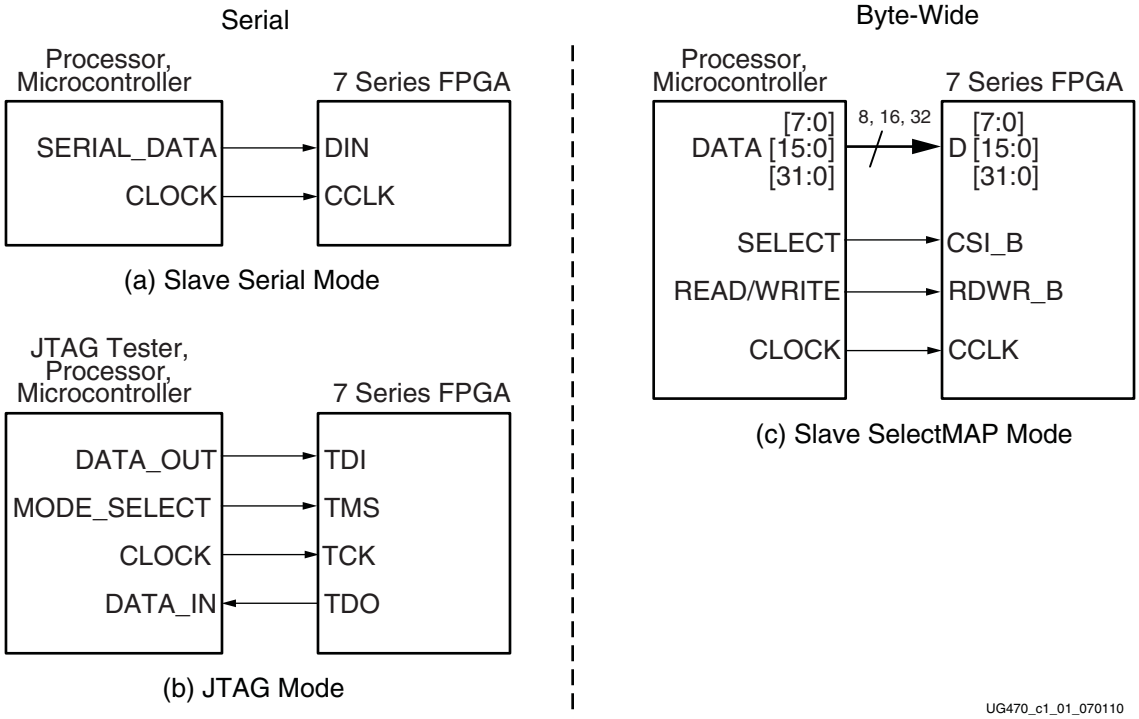


Figure 1-1: **Slave Configuration Modes**

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-pin JTAG interface is common on board testers and debugging hardware. In fact, the Xilinx programming cable for 7 series FPGAs, listed here, uses 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

The Basic Configuration Solution

In the basic configuration solution, the FPGA automatically retrieves its bitstream from a flash memory device at power-on. The FPGA has a serial peripheral interface (SPI) through which the FPGA can read a bitstream from a standard SPI flash device.

The iMPACT tool in the ISE® software can program select SPI flash memories. The iMPACT tool can communicate with the FPGA through its standard JTAG interface and can program the SPI flash, indirectly through the FPGA. See XAPP586, *Using SPI Flash with 7 Series FPGAs.* Similar capability is found in the Vivado device programmer.

The Low-Cost Configuration 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.

The High-Speed 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.

- 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 configuration clock signal and sends it out on the CCLK pin. By default, the CCLK frequency starts out low but bitstream options can either increase the internally generated CCLK frequency or switch the CCLK source to an external clock source from the EMCCLK pin. The maximum supported CCLK frequency setting depends on the read specifications for the attached nonvolatile memory. A faster memory enables faster configuration. When using the internal oscillator source for CCLK, the output frequency can vary with process, voltage, or temperature. The EMCCLK clock source option enables a precision external clock source for optimal configuration performance.
- Slave mode or Master mode using the EMCCLK option allows tighter tolerances and faster clocks.

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 5, Configuration Details. In addition, 7 series devices also have on-chip Advanced Encryption Standard (AES) decryption logic to provide a high degree of design security. See Bitstream Encryption in Chapter 5, 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 flash memory 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. Ganged or broad-side configuration is supported only in the slave serial or slave SelectMAP modes (see Chapter 9, Multiple FPGA Configuration).

Configuration Factors

Many factors determine which configuration solution is optimal for a system. There are also a great number 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 Chapter 5, Configuration Details.

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 configuration process involves many steps. Each step often involves a sequence of events. For example, the first step is the power-up sequence for the multiple power supplies. To understand the overall configuration time, a designer must understand the contribution of each step. The Vivado tools provide the Tcl command CALC_CONFIG_TIME which can be used to estimate configuration time. Use **`help calc_config_time`** for usage information.

More details can be found in Chapter 5, Configuration Details.

Stacked Silicon Interconnect

The devices in Table 1-1 with two or more super logic regions use stacked silicon interconnect (SSI) technology. The Virtex-7 FPGAs that are designed using stacked silicon interconnect technology support the same configuration modes as the monolithic 7 series devices.

Global configuration functions default to being controlled from the master super logic region (SLR) in an SSI device. SPI and JTAG configuration for SSI devices restricts ICAP read/write access as identified in Table 1-2.

Table 1-2: **ICAP Access to Configuration Memory**

Device Type	Configuration Mode	ICAP Access to Configuration Memory	
Monolithic Devices	All Modes	Entire Device	
SSI Devices		**Master SLR ICAP**	**Slave SLR ICAP**
	SPI x1, JTAG(1)	Master SLR Only(2)	Slave SLR
	SPI x2, SPI x4		N/A(3)
	All Other Modes	Entire Device	Slave SLR

Notes:

1. Master ICAP cannot access slave SLRs when mode pins are set to JTAG mode. As JTAG mode is always available independent of the mode pins, the JTAG mode pin setting is not recommended for SSI devices.
2. Master ICAP cannot access configuration memory in slave SLRs when configuring in SPI modes. Therefore partial reconfiguration must use the ICAP within the local SLR with SPI x1 configuration mode.
3. ICAP in slave SLRs cannot be used when configuring in SPI x2 or SPI x4 modes. Therefore partial reconfiguration is limited to the master SLR, and SEM IP (error correction) is not supported with these configuration modes.

JTAG configuration for SSI devices is supported only via iMPACT or the Vivado device programmer using either a JTAG cable connection or solutions based from the serial vector format (SVF) file.

Command sequence examples provided in Chapter 6, Readback and Configuration Verification (Table 6-1 through Table 6-5) and Chapter 7, Reconfiguration and MultiBoot (Table 7-1, Table 7-6, and Table 7-7) support monolithic 7 series devices and not SSI devices.

For more information on stacked silicon interconnect technology, see WP380, *Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency.*

Configuration Debugging

The best practices discussed in this section helps to enable debug and resolution if you encounter an issue when implementing a configuration solution. Before you embark on a full debug, create and test a simple design using the bitstream defaults (for example, a counter or LED output pattern). This simple design test helps eliminate any potential issues with advanced bitstream settings or board interfaces. Try configuration using a different method, such as configuring through JTAG instead of from a flash memory, to determine if the issue is specific to the configuration mode. The Xilinx Configuration Solution Center is an additional resource. For more information, see www.xilinx.com/support/answers/34904.htm.

The two most important configuration signals, INIT_B and DONE, should be connected to LED drivers. The pulsing of INIT_B from Low to High indicates the completion of initialization at power-up. A falling INIT_B signal during configuration can indicate a CRC error in the bitstream seen by the FPGA device. Recommendations for configuration and other pins are found in XMP277, *7 Series Schematic Design Checklist*. If configuration has not completed properly, the status register provides important information about what errors might have caused the failure. JTAG readback/verify can determine whether the intended configuration data was loaded correctly into the device.

The configuration simulation model (SIM_CONFIG) allows supported configuration interfaces to be simulated. This is a model of how the supported devices react to stimulus on the configuration interface. For more information, see UG626, *Synthesis and Simulation Design Guide*.

XILINX

Chapter 2

Configuration Interfaces

Xilinx® 7 series 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 the respective 7 series FPGAs data sheet. Configuration timing is relative to the CCLK at the pin, even in Master modes where the CCLK is generated internally.

Table 2-1: **7 Series FPGA Configuration Modes**

Configuration Mode	M[2:0]	Bus Width	CCLK Direction
Master Serial	000	x1	Output
Master SPI	001	x1, x2, x4	Output
Master BPI	010	x8, x16	Output
Master SelectMAP	100	x8, x16	Output
JTAG	101	x1	Not Applicable
Slave SelectMAP	110	x8, x16, x32(1)	Input
Slave Serial(2)	111	x1	Input

Notes:

1. The Slave SelectMAP x16 and x32 bus widths do not support AES-encrypted bitstreams.
2. This is the default setting due to internal pull-up resistors on the Mode pins.

Configuration Pins

Each configuration mode has a corresponding set of interface pins that span one or more I/O banks on the 7 series FPGA. Bank 0 contains the dedicated configuration pins and is always part of every configuration interface. Bank 14 and Bank 15 contain multi-function pins that are involved in specific configuration modes. The 7 series FPGAs data sheets specify the switching characteristics for configuration pins in banks operating at 3.3V, 2.5V, 1.8V, or 1.5V.

All JTAG and dedicated configuration pins are located in a separate, dedicated bank with a dedicated voltage supply (VCCO_0). The multi-function pins are located in Banks 14 and 15.

All dedicated input pins operate at the VCCO_0 LVCMOS level (LVCMOS18, LVCMOS25, or LVCMOS33). All active dedicated output pins operate at the VCCO_0 voltage level with the output standard set to LVCMOS, 12 mA drive, fast slew rate. For all modes that use multi-function I/O, the associated VCCO_14 or VCCO_15 must be connected to the appropriate voltage to match the I/O standard of the configuration device. The

multi-function pins are also LVCMOS, 12 mA drive, fast slew rate during configuration. If the Persist option is used (See Persist Option, page 124), the multi-function I/O for the selected configuration mode remain active after configuration, with the I/O standard set to the general-purpose default of LVCMOS, 12 mA drive, Slow slew rate.

Table 2-2 and Table 2-3 show the configuration mode pins and their location across the I/O banks.

Table 2-2: **Configuration Mode Pins (Table 1 of 2)**

Pin Name	Bank	JTAG (Only)	Slave Serial	Master Serial	Master SPI		
					x1	x2	x4
CFGBVS	0	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS
M[2:0]	0	M[2:0]=101	M[2:0]=111	M[2:0]=000	M[2:0]=001	M[2:0]=001	M[2:0]=001
TCK	0	TCK	TCK	TCK	TCK	TCK	TCK
TMS	0	TMS	TMS	TMS	TMS	TMS	TMS
TDI	0	TDI	TDI	TDI	TDI	TDI	TDI
TDO	0	TDO	TDO	TDO	TDO	TDO	TDO
PROGRAM_B	0	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B
INIT_B	0	INIT_B	INIT_B	INIT_B	INIT_B	INIT_B	INIT_B
DONE	0	DONE	DONE	DONE	DONE	DONE	DONE
CCLK	0	CCLK	CCLK	CCLK	CCLK	CCLK	CCLK
PUDC_B(1)	14	PUDC_B(1)	PUDC_B(1)	PUDC_B(1)	PUDC_B(1)	PUDC_B(1)	PUDC_B(1)
EMCCLK(2)	14	–	–	EMCCLK(2)	EMCCLK(2)	EMCCLK(2)	EMCCLK(2)
CSI_B	14	–	–	–	–	–	–
DOUT_CSO_B(3)(4)	14	–	[DOUT](3)	[DOUT](3)	[DOUT](3)	–	–
RDWR_B	14	–	–	–	–	–	–
FCS_B	14	–	–	–	FCS_B	FCS_B	FCS_B
D00_MOSI	14	–	–	–	MOSI	MOSI/D00	MOSI/D00
D01_DIN	14	–	DIN	DIN	DIN	DIN/D01	DIN/D01
D02	14	–	–	–	–	–	D02
D03	14	–	–	–	–	–	D03
D[04-07]	14	–	–	–	–	–	–
D[08-15]	14	–	–	–	–	–	–
A[00-15]_D[16-31]	14	–	–	–	–	–	–
A[16-28]	15	–	–	–	–	–	–
FOE_B	15	–	–	–	–	–	–
FWE_B	15	–	–	–	–	–	–
ADV_B	15	–	–	–	–	–	–

Table 2-2: **Configuration Mode Pins (Table 1 of 2) *(Cont'd)***

Pin Name	Bank	JTAG (Only)	Slave Serial	Master Serial	Master SPI		
					x1	x2	x4
RS0, RS1[5]	15	RS0, RS1[5]	RS0, RS1[5]	RS0, RS1[5]	–	–	–

Notes:
1. PUDC_B has special functionality during configuration but is independent of all configuration interfaces, i.e. PUDC_B does not need to be voltage compatible with other pins in a configuration interface.
2. EMCCLK is only used when the ExtMasterCclk_en option enables EMCCLK as an input for clocking the master configuration modes.
3. DOUT is only used in a serial configuration daisy-chain for outputting data to the downstream FPGA (or for the DebugBitstream option). Otherwise, DOUT is high-impedance.
4. CSO_B is only used in a parallel configuration daisy-chain for outputting a chip-enable signal to a downstream device. Otherwise, CSO_B is high-impedance.
5. RS0 and RS1 are only driven when a MultiBoot event is initiated or when the ConfigFallback option is enabled and a Fallback event occurs. Otherwise, RS0 and RS1 are high-impedance.
6. Empty cells indicate that the pin is not used in the configuration mode and is ignored and is high-impedance during configuration.

Table 2-3: **Configuration Mode Pins (Table 2 of 2)**

Pin Name	Bank	Master SelectMAP		Slave SelectMAP			Master BPI	
		x8	x16	x8	x16	x32	x8	x16
CFGBVS	0	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS
M[2:0]	0	M[2:0]=100	M[2:0]=100	M[2:0]=110	M[2:0]=110	M[2:0]=110	M[2:0]=010	M[2:0]=010
TCK	0	TCK	TCK	TCK	TCK	TCK	TCK	TCK
TMS	0	TMS	TMS	TMS	TMS	TMS	TMS	TMS
TDI	0	TDI	TDI	TDI	TDI	TDI	TDI	TDI
TDO	0	TDO	TDO	TDO	TDO	TDO	TDO	TDO
PROGRAM_B	0	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B
INIT_B	0	INIT_B	INIT_B	INIT_B	INIT_B	INIT_B	INIT_B	INIT_B
DONE	0	DONE	DONE	DONE	DONE	DONE	DONE	DONE
CCLK	0			CCLK	CCLK	CCLK	CCLK	CCLK
PUDC_B[1]	14	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]
EMCCLK[2]	14	EMCCLK[2]	EMCCLK[2]	–	–	–	EMCCLK[2]	EMCCLK[2]
CSI_B	14	CSI_B	CSI_B	CSI_B	CSI_B	CSI_B	–	–
DOUT_CSO_B[3][4]	14	[CSO_B][4]	[CSO_B][4]	[CSO_B][4]	[CSO_B][4]	[CSO_B][4]	[CSO_B][4]	[CSO_B][4]
RDWR_B	14	RDWR_B	RDWR_B	RDWR_B	RDWR_B	RDWR_B	–	–
FCS_B	14	–	–	–	–	–	FCS_B	FCS_B
D00_MOSI	14	D00	D00	D00	D00	D00	D00	D00
D01_DIN	14	D01	D01	D01	D01	D01	D01	D01
D02	14	D02	D02	D02	D02	D02	D02	D02
D03	14	D03	D03	D03	D03	D03	D03	D03
D[04-07]	14	D[04-07]	D[04-07]	D[04-07]	D[04-07]	D[04-07]	D[04-07]	D[04-07]
D[08-15]	14	–	D[08-15]	–	D[08-15]	D[08-15]	–	D[08-15]
A[00-15]_D[16-31]	14		–	–	–	D[16-31]	A[00-15]	A[00-15]
A[16-28]	15			–	–	–	A[16-28]	A[16-28]
FOE_B	15			–	–	–	FOE_B	FOE_B
FWE_B	15			–	–	–	FWE_B	FWE_B

Table 2-3: **Configuration Mode Pins (Table 2 of 2) *(Cont'd)***

Pin Name	Bank	Master SelectMAP		Slave SelectMAP			Master BPI	
		x8	x16	x8	x16	x32	x8	x16
ADV_B	15			–	–	–	ADV_B	ADV_B
RS0, RS1[5]	15	RS0, RS1[5]	RS0, RS1[5]	RS0, RS1[5]	RS0, RS1[5]	RS0, RS1[5]	RS0, RS1[5]	RS0, RS1[5]

Notes:
1. PUDC_B has special functionality during configuration but is independent of all configuration interfaces, i.e. PUDC_B does not need to be voltage compatible with other pins in a configuration interface.
2. EMCCLK is only used when the BitGen ExtMasterCclk_en option enables EMCCLK as an input for clocking the master configuration modes.
3. DOUT is only used in a serial configuration daisy-chain for outputting data to the downstream FPGA (or for the BitGen DebugBitstream option). Otherwise, DOUT is high-impedance.
4. CSO_B is only used in a parallel configuration daisy-chain for outputting a chip-enable signal to a downstream device. Otherwise, CSO_B is high-impedance.
5. RS0 and RS1 are only driven when a MultiBoot event is initiated or when the BitGen ConfigFallback option is enabled and a Fallback event occurs. Otherwise, RS0 and RS1 are high-impedance.
6. Empty cells indicate that the pin is not used in the configuration mode and is ignored and is high-impedance during configuration.

The definition of each configuration pin is summarized in Table 2-4.

Table 2-4: **Configuration Pin Definitions**

Pin Name	Bank[1]	Type	Direction	Description
CFGBVS	0	Dedicated	Input	Configuration Banks Voltage Select CFGBVS determines the I/O voltage operating range and voltage tolerance for the dedicated configuration bank 0 and for the multi-function configuration pins in banks 14 and 15 in the Artix-7 and Kintex-7 families. CFGBVS selects the operating voltage for the dedicated bank 0 at all times in all 7 series devices. CFGBVS selects the operating voltage for the multi-function configuration banks 14 and 15 only during configuration. Connect CFGBVS High or Low per the bank voltage requirements. If the V_{CCO_0} supply for bank 0 is supplied with 2.5V or 3.3V, then the CFGBVS pin must be tied High (i.e. connected to V_{CCO_0}). Tie CFGBVS to Low (i.e. connected to GND), only if the V_{CCO_0} for bank 0 is less than or equal to 1.8V. If used during configuration, banks 14 and 15 should match the VCCO level applied to bank 0. ***Caution!*** To avoid device damage, CFGBVS must be connected correctly to either V_{CCO_0} or GND. See Configuration Banks Voltage Select, page 32 for more information. ***Note:*** The CFGBVS pin is not available on Virtex-7 HT devices. Virtex-7 HT devices support only 1.8V/1.5V operation for bank 0.
M[2:0]	0	Dedicated	Input	Configuration Mode M[2:0] determine the configuration mode. See Table 2-3, page 23 for the configuration mode settings. Connect each mode pin either directly, or via a ≤ 1 kΩ resistor, to V_{CCO_0} or GND.

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description
TCK	0	Dedicated	Input	IEEE Std 1149.1 (JTAG) Test Clock Clock for all devices on a JTAG chain. Connect to Xilinx cable header's TCK pin. Treat as a critical clock signal and buffer the cable header TCK signal as necessary for multiple device JTAG chains. If the TCK signal is buffered, connect the buffer input to an external weak (e.g. 10 kΩ) pull-up resistor to maintain a valid High when no cable is connected.
TMS	0	Dedicated	Input	JTAG Test Mode Select Mode select for all devices on a JTAG chain. Connect to Xilinx cable header's TMS pin. Buffer the cable header TMS signal as necessary for multiple device JTAG chains. If the TMS signal is buffered, connect the buffer input to an external weak (e.g. 10 kΩ) pull-up resistor to maintain a valid High when no cable is connected.
TDI	0	Dedicated	Input	JTAG Test Data Input JTAG chain serialized data input. For an isolated device or for the first device in a JTAG chain, connect to Xilinx cable header's TDI pin. Otherwise, when the FPGA is not the first device in a JTAG chain, connect to the TDO pin of the upstream JTAG device in the JTAG scan chain.
TDO	0	Dedicated	Output	JTAG Test Data Output JTAG chain serialized data output. For an isolated device or for the last device in a JTAG chain, connect to Xilinx cable header's TDO pin. Otherwise, when the FPGA is not the last device in a JTAG chain, connect to the TDI pin of the downstream JTAG device in the JTAG scan chain.
PROGRAM_B	0	Dedicated	Input	Program (bar) Active-Low reset to configuration logic. When PROGRAM_B is pulsed Low, the FPGA configuration is cleared and a new configuration sequence is initiated. Configuration reset initiated upon falling edge, and configuration (i.e. programming) sequence begins upon the following rising edge. Connect PROGRAM_B to an external ≤ 4.7 kΩ pull-up resistor to V_{CCO_0} to ensure a stable High input, and recommend push-button to GND to enable manual configuration reset. ***Note:*** Holding PROGRAM_B Low from power-on does not keep the FPGA configuration in reset. Instead, use INIT_B to delay the power-on configuration sequence.

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description
INIT_B	0	Dedicated	Bidirectional (open-drain)	Initialization (bar) Active-Low FPGA initialization pin or configuration error signal. The FPGA drives this pin Low when the FPGA is in a configuration reset state, when the FPGA is initializing (clearing) its configuration memory, or when the FPGA has detected a configuration error. Upon completing the FPGA initialization process, INIT_B is released to high-impedance at which time an external resistor is expected to pull INIT_B High. INIT_B can externally be held Low during power-up to stall the power-on configuration sequence at the end of the initialization process. When a High is detected at the INIT_B input after the initialization process, the FPGA proceeds with the remainder of the configuration sequence dictated by the M[2:0] pin settings. Connect INIT_B to a ≤ 4.7 kΩ pull-up resistor to V_{CCO_0} to ensure clean Low-to-High transitions.
DONE	0	Dedicated	Bidirectional	Done A High signal on the DONE pin indicates completion of the configuration sequence. The DONE output is an open-drain output by default. ***Note:*** DONE has an internal pull-up resistor of approximately 10 kΩ. There is no setup/hold requirement for the DONE register. These changes, along with the DonePipe register software default, eliminate the need for the DriveDONE driver-option. External 330Ω resistor circuits are not required but can be used as they have been in previous generations.
CCLK	0	Dedicated	Input or Output	Configuration Clock CCLK runs the synchronous FPGA configuration sequence in all modes except JTAG mode. • For slave modes: CCLK is an input and requires connection to an external clock source. • For master modes: The FPGA sources the configuration clock and drives CCLK as an output. • For JTAG mode: CCLK is high-impedance and can be left unconnected. ***Note:*** Treat CCLK as a critical clock signal to ensure good signal integrity (see the Signal Integrity page on xilinx.com for more information).

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description
PUDC_B	14	Multi-function	Input	Pull-Up During Configuration (bar) Active-Low PUDC_B input enables internal pull-up resistors on the SelectIO pins after power-up and during configuration. • When PUDC_B is Low, internal pull-up resistors are enabled on each SelectIO pin. • When PUDC_B is High, internal pull-up resistors are disabled on each SelectIO pin. PUDC_B must be tied either directly, or via a ≤ 1 kΩ to V_{CCO_14} or GND. ***Caution!*** Do not allow this pin to float before and during configuration.
EMCCLK	14	Multi-function	Input	External Master Configuration Clock Optional external clock input for running the configuration logic in a master mode (versus the internal configuration oscillator). See Setting Configuration Options in the Vivado Tools, page 36 for more information. • For master modes: The FPGA can optionally switch to EMCCLK as the clock source, instead of the internal oscillator, for driving the internal configuration engine. The EMCCLK frequency can optionally be divided via a bitstream setting (ExtMasterCclk_en) and is forwarded for output as the master CCLK signal. • For JTAG and slave modes: EMCCLK is ignored in the JTAG and slave modes and can be left unconnected.
CSI_B	14	Multi-function	Input	Chip Select Input (bar) Active-Low input that enables the FPGA SelectMAP configuration interface. • For master SelectMAP mode: Connect CSI_B directly, or via a ≤ 1 kΩ resistor, to GND. • For slave SelectMAP mode: An external configuration controller can control CSI_B for selecting the active FPGA on the SelectMAP bus, or in a parallel configuration daisy-chain, connect to the CSO_B pin of the upstream FPGA. • In all other modes: CSI_B is ignored and can be left unconnected.

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description
CSO_B	14	Multi-function	Output (open-drain)	Chip Select Output (bar) Active-Low open-drain output that can drive Low to enable the slave SelectMAP configuration interface of the downstream FPGA in a parallel configuration daisy-chain. • For BPI (asynchronous read only) and SelectMAP modes: If the device is in a parallel configuration daisy-chain and has a downstream device, then connect to an external 330Ω pull-up to V_{CCO_14} and connect to the CSI_B input of the downstream device. Otherwise, CSO_B is high-impedance. • For serial modes: CSO_B is a multi-purpose pin that functions as the DOUT pin. See DOUT row in this table. • For all other modes: CSO_B is high-impedance and can be left unconnected.
DOUT	14	Multi-function	Output	Data Output DOUT is the data output for a serial configuration daisy-chain. • For serial and SPI (x1 only) modes: If the device is in a serial configuration daisy-chain, then connect to the DIN of the downstream slave-serial FPGA. Otherwise, DOUT is high-impedance. • For BPI and SelectMAP modes: DOUT is a multi-purpose pin that functions as the CSO_B pin. See CSO_B row in this table. • For all other modes: DOUT is high-impedance and can be left unconnected. ***Note:*** DOUT can output data when the DebugBitstream option is enabled.
RDWR_B	14	Multi-function	Input	Read/Write (bar) RDWR_B determines the direction of the SelectMAP data bus. When RDWR_B is High, the FPGA outputs read data onto the SelectMAP data bus. When RDWR_B is Low, an external controller can write data to the FPGA through the SelectMAP data bus. • For master SelectMAP mode: Connect RDWR_B directly, or via a ≤ 1 kΩ resistor, to GND. • For slave SelectMAP mode: An external device controls the RDWR_B signal to control the direction of the SelectMAP data bus for read/write from/to the SelectMAP interface. • In all other modes: RDWR_B is ignored and can be left unconnected.

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description
D00_MOSI	14	Multi-function	Bidirectional	Master-Output, Slave-Input FPGA (master) SPI mode output for sending commands to the SPI (slave) flash device. • For SPI mode: Connect to the SPI flash data input pin. • For SelectMAP modes: The MOSI pin is a multi-purpose pin that functions as the D00 data input pin. See D[00-31] row in this table. • For all other modes: The MOSI pin function is not applicable, the pin is high-impedance during configuration, is ignored during configuration, and can be left unconnected.
D01_DIN	14	Multi-function	Bidirectional	Data Input DIN is the serial data input pin. By default, data from DIN is captured on the rising edge of CCLK. • For serial and SPI modes: DIN is the FPGA data input that receives serial data from the data source. Connect DIN to the serial data output pin of the serial data source. • For BPI and SelectMAP modes: The DIN pin is a multi-purpose pin that functions as the D01 data input pin. See D[00-31] row in this table. • For JTAG mode: DIN is ignored.

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description

D[00-31]	14	Multi-function	Bidirectional	Data Bus
				A subset or all of the D[00-31] pins are the data bus interface for the SPI x2, SPI x4, BPI, or SelectMAP modes. By default, data from the data bus is captured on the rising edge of CCLK.
				• For SPI mode: Configuration begins with the D00/MOSI and D01 pins of the data bus used for standard SPI (x1) serial data output and data input. Bitstream options can switch the SPI flash read mode to dual output (x2) or quad output (x4) modes.
				  • For SPI x1/x2/x4: Connect D00/MOSI to the SPI flash serial data input (DQ0/D/SI/IO0) pin.
				  • For SPI x1/x2/x4: Connect D01/DIN to the SPI flash serial data output (DQ1/Q/SO/IO1) pin.
				  • For SPI x4: Connect D02 to the SPI flash quad data bit 2 output (DQ2/W#/WP#/IO2) pin and connect to an external 4.7kohm pull-up resistor to VCCO_14.
				  • For SPI x4: Connect D03 to the SPI flash quad data bit 3 output (DQ3/HOLD#/IO3) pin and connect to an external 4.7kohm pull-up resistor to VCCO_14.
				  The remaining data pins are unused, ignored, and high impedance during configuration.
				• For SelectMAP modes: The FPGA monitors the D[00-07] for an auto-bus-width-detect pattern that determines whether only D[00-07] (x8 bus width) are used or a wider (x16 or x32) data bus width is used. Connect used data bus pins to the corresponding data pins on the data source.
				  Caution! The slave SelectMAP x16 and x32 data bus widths do not support configuration from AES-encrypted bitstreams.
				• For BPI mode: The FPGA monitors the D[00-07] for an auto-bus-width-detect pattern that determines whether only D[00-07] (x8 bus width) are used or a wider (x16) data bus width is used. Connect used data bus pins to the corresponding data pins on the BPI flash. The D[16-31] pins are multi-purpose pins that function as the BPI address A[00-15] pins. See A[00-28] row in this table.
				• For JTAG mode: None of the data pins are used.
				• For all modes: The unused data pins are high-impedance and ignored during configuration. The unused data pins can be left unconnected.

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description
A[00-28]	14	Multi-function	Output	Address Bus A[00-28] pins output addresses to a parallel NOR (BPI) flash. A00 is the least-significant address bit. • For BPI mode: Connect the FPGA A[00-28] pins to the parallel NOR flash address pins with the FPGA A00 pin connected to the least-significant flash address input pin that is valid for the used data bus width. Depending on the BPI flash type and used data bus width, the least-significant address bit of the flash can be A1, A0, or A-1. Note that any upper address pins that exceed the address bus width of the parallel NOR flash are driven during configuration, but can be used as I/O after configuration. • For SelectMAP mode: The A[00-15] pins are multi-purpose pins that function as the D[16-31] data bus pins. See D[00-31] row in this table. • In the other modes: A[00-28] are high-impedance, are ignored during configuration, and can be left unconnected.
FCS_B	14	Multi-function	Output	Flash Chip Select (bar) Active-Low chip select output that enables SPI or BPI flash devices for configuration. • For SPI and BPI modes: Connect the FPGA FCS_B to the flash device chip-select input and connect to an external ≤ 4.7 kΩ pull-up resistor to V_{CCO_14}. • For all other modes: FCS_B is high-impedance and can be left unconnected.
FOE_B	15	Multi-function	Output	Flash Output-Enable (bar) Active-Low output-enable control signal for a parallel NOR flash. • For BPI mode: Connect the FPGA FOE_B to the flash output-enable input and connect to an external ≤ 4.7 kΩ pull-up resistor to V_{CCO_15}. • For all other modes: FOE_B is high-impedance and can be left unconnected.
FWE_B	15	Multi-function	Output	Flash Write-Enable (bar) Active-Low write-enable control signal for a parallel NOR flash. • For BPI mode: Connect the FPGA FWE_B to the flash write-enable input and connect to an external ≤ 4.7 kΩ pull-up resistor to V_{CCO_15}. • For all other modes: FWE_B is high-impedance, and can be left unconnected.

Table 2-4: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank[1]	Type	Direction	Description
ADV_B	15	Multi-function	Output	Address Valid (bar) Active-Low address valid output signal for a parallel NOR flash. • For BPI mode with flash that support an address valid input: Connect the FPGA ADV_B to the parallel NOR flash address valid input pin and connect to an external ≤ 4.7 kΩ pull-up resistor to V_{CCO_15}. For BPI mode with flash that do not support an address valid input: Do not connect the ADV_B pin. • For all other modes: ADV_B is high-impedance, and can be left unconnected.
RS0, RS1	15	Multi-function	Output	Revision Select The RS0 and RS1 pins are revision selection output pins, intended to drive upper address lines on a parallel flash memory. Normally, RS0 and RS1 are high-impedance during configuration. However, the FPGA can drive the RS0 and RS1 pins under two possible conditions. When the ConfigFallback option is enabled, the FPGA drives RS0 and RS1 Low during the fallback configuration process that follows a detected configuration error. When a user-invoked MultiBoot configuration is initiated, the FPGA can drive the RS0 and RS1 pins to a user-defined state during the MultiBoot configuration process. If fallback is disabled (default) and if MultiBoot is not used, or if SPI mode is used, then RS0 and RS1 are high-impedance and can be left unconnected.
VCCBATT	N/A	Supply Voltage	N/A	Battery Backup Supply V_{CCBATT} is the battery backup supply for the FPGA's internal volatile memory that stores the key for the AES decryptor. For encrypted bitstreams that require the decryptor key from the volatile key memory area, connect this pin to a battery to preserve the key when the FPGA is unpowered. If there is no requirement to use the decryptor key from the volatile key storage area, connect this pin to GND or V_{CCAUX}. The pin name includes the "_0" bank designation but it is not an I/O and not affected by V_{CCO_0}.

Notes:

1. Each I/O is referenced to the V_{CCO} supply voltage for the bank in which the I/O is located. For example, "0" indicates the I/O is referenced to Bank 0's V_{CCO_0}.

Configuration Banks Voltage Select

The configuration banks voltage select (CFGBVS) pin must be set to High, or Low, in order to determine the I/O voltage support for the pins in bank 0, and for the multi-function pins in banks 14 and 15 when they are used during configuration. The CFGBVS is a logic input pin referenced between V_{CCO_0} and GND. When the CFGBVS pin is High (e.g., connected to the V_{CCO_0} supply of 3.3V or 2.5V), the configuration and JTAG I/O on bank 0 support operation at 3.3V or 2.5V during and after configuration. When the CFGBVS pin is Low

(e.g., connected to GND), the I/O in bank 0 support operation at 1.8V or 1.5V. Configuration is not supported at 1.2V.

CFGBVS similarly controls the voltage tolerance on banks 14 and 15, but only during configuration. When CFGBVS is High, the configuration I/O on banks 14 and 15 support operation at 3.3V or 2.5V during configuration. When the CFGBVS pin is Low, the configuration I/O in banks 14 and 15 support operation at 1.8V or 1.5V during configuration.

The 7 series FPGAs have two I/O bank types: high-range (HR I/O) banks support 3.3V, 2.5V, and a few lower voltage I/O standards, and high-performance (HP I/O) banks support I/O standards of 1.8V or lower voltage. The dedicated configuration and JTAG I/O are located in bank 0. Bank 0 is a high-range bank type on all devices except for the Virtex-7 HT devices. Several of the configuration modes also rely on pins in bank 14 and/or bank 15. Bank 14 and bank 15 are HR I/O banks in the Artix-7 and Kintex-7 families, but are always HP I/O banks in the Virtex-7 family. See UG475, 7 series *FPGAs Packaging and Pinout Guide* for bank information for each device.

Note: The CFGBVS pin is not available on Virtex-7 HT devices. Virtex-7 HT devices support only 1.8V operation for configuration banks.

The CFGBVS pin setting determines the I/O voltage support for bank 0 at all times, and for bank 14 and bank 15 during configuration. The V_{CCO} supply for each configuration bank must match the CFGBVS selection if used during configuration — 2.5V or 3.3V if CFGBVS is tied to VCCO_0, and 1.8V or 1.5V if CFGBVS is tied to GND.

Table 2-5 shows the CFGBVS pin connection options and the corresponding set of valid V_{CCO} supply and I/O voltages.

Table 2-5: **CVGBVS Pin Connection Options**

CFGBVS Pin Connection	Supported Configuration Banks 0/14/15 V_{CCO} Supply and I/O Signal Voltages		
	Artix-7, Kintex-7	Virtex-7 T, XT	Virtex-7 HT
Banks Affected	**0, 14, 15**	**0**	**none**
V_{CCO_0} (3.3V or 2.5V)	3.3V or 2.5V	3.3V or 2.5V	1.8V (no CFGBVS)
GND	1.8V or 1.5V	1.8V or 1.5V	

Caution! When CFGBVS is set to Low for 1.8V/1.5V I/O operation, the V_{CCO_0} and I/O signals to bank 0 must be 1.8V (or lower). VCCO_14 and VCCO_15 must also be 1.8V/1.5V if configuration I/O in those banks are used during configuration. Otherwise, the device can be damaged from the application of voltages to pins on these banks that are greater than the 1.8V operation maximum.

Depending on the configuration mode, the interface pins associated with the mode can span bank 0, bank 14, and bank 15. Typically, all three banks receive the same V_{CCO} voltage supply to ensure a consistent I/O voltage interface for all of the configuration interface pins. Using the same voltage for banks 0, 14, and 15 is recommended because it allows the option of using an 8-bit or wider configuration mode, and avoids the I/O transition described under I/O Transition at the End of Startup, page 90.

Use these steps to determine the proper CFGBVS pin setting:

1. Determine the configuration mode(s) for the FPGA.

   Note: The JTAG interface is always supported in bank 0 at the V_{CCO_0} voltage level regardless of the configuration mode.

2. For each configuration mode to be used for the FPGA, determine the set of pins used for the configuration mode and the bank locations (see Table 2-2 and Table 2-3).
3. For each set of configuration pins, determine the common required I/O voltage support for the required configuration bank(s).
4. Determine the target FPGA family. The Virtex-7 FPGAs only support 1.8V/1.5V configuration on banks 14 and 15. The Virtex-7 HT family only supports 1.8V configuration on bank 0 also, and therefore does not have a CFGBVS pin.
5. Set the CFGBVS pin to support the required configuration I/O voltage. See Table 2-9 through Table 2-11 for the appropriate CFGBVS pin setting.

Table 2-6: **Artix-7 and Kintex-7 FPGA Configuration Mode, Compatible Voltages, and CFGBVS Connection**

Configuration Mode	Banks Used	Configuration Interface I/O Voltage	HR Bank 0 V_{CCO_0}	HR Bank 14 V_{CCO_14}	HR Bank 15 V_{CCO_15}	CFGBVS
JTAG (only)	0	3.3V	3.3V	Any	Any	VCCO_0
		2.5V	2.5V	Any	Any	VCCO_0
		1.8V	1.8V	Any	Any	GND
		1.5V	1.5V	Any	Any	GND
Serial, SPI, or SelectMAP	0, 14(1)	3.3V	3.3V	3.3V	Any	VCCO_0
		2.5V	2.5V	2.5V	Any	VCCO_0
		1.8V	1.8V	1.8V	Any	GND
		1.5V	1.5V	1.5V	Any	GND
BPI	0, 14, 15	3.3V	3.3V	3.3V	3.3V	VCCO_0
		2.5V	2.5V	2.5V	2.5V	VCCO_0
		1.8V	1.8V	1.8V	1.8V	GND
		1.5V	1.5V	1.5V	1.5V	GND

Notes:

1. RS[1:0] for MultiBoot or Fallback are in bank 15 but are typically only used in BPI mode and not supported in SPI mode.

Table 2-7: **Virtex-7 T and XT FPGA Configuration Mode, Compatible Voltages, and CFGBVS Connection**

Configuration Mode	Banks Used	Configuration Interface I/O Voltage	HR Bank 0 V_{CCO_0}	HP Bank 14 V_{CCO_14}(1)	HP Bank 15 V_{CCO_15}(1)	CFGBVS
JTAG (only)	0	3.3V	3.3V	≤1.8V	≤1.8V	VCCO_0
		2.5V	2.5V	≤1.8V	≤1.8V	VCCO_0
		1.8V	1.8V	≤1.8V	≤1.8V	GND
		1.5V	1.5V	≤1.8V	≤1.8V	GND
Serial, SPI, or SelectMAP	0, 14(2)	1.8V	1.8V	1.8V	≤1.8V	GND
		1.5V	1.5V	1.5V	≤1.8V	GND
BPI	0, 14, 15	1.8V	1.8V	1.8V	1.8V	GND
		1.5V	1.5V	1.5V	1.5V	GND

Notes:

1. In the Virtex-7 FPGA, banks 14 and 15 are high-performance banks, limited to 1.8V or lower I/O standards. CFGBVS does not affect those banks.
2. RS[1:0] for MultiBoot or Fallback are in bank 15 but are typically only used in BPI mode and not supported in SPI mode.

Table 2-8: **Virtex-7 HT FPGA Configuration Mode and Compatible Voltages**

Configuration Mode	Banks Used	Configuration Interface I/O Voltage	HP Bank 0 V_{CCO_0}	HP Bank 14 VCCO_14	HP Bank 15 VCCO_15	CFGBVS(1)
JTAG (only)	0	1.8V	1.8V	≤1.8V	≤1.8V	N/A
Serial, SPI, or SelectMAP	0, 14(2)	1.8V	1.8V	1.8V	≤1.8V	N/A
BPI	0, 14, 15	1.8V	1.8V	1.8V	≤1.8V	N/A

Notes:

1. Virtex-7 HT devices only support 1.8V operation for configuration banks including bank 0. CFGBVS is not supported.
2. RS[1:0] for MultiBoot or Fallback are in bank 15 but are typically only used in BPI mode and not supported in SPI mode.

Table 2-9: **Configuration Mode, Compatible Voltages, and CFGBVS Pin Connection**

Configuration Mode	Configuration Interface I/O Voltage	Applicable Family			Compatible Bank Voltages			Required CFGBVS Pin Connection
		Artix-7 Family	Kintex-7 Family	Virtex-7 Family	Bank 0 V_{CCO_0} Voltage	Bank 14 V_{CCO_14} Voltage	Bank 15 V_{CCO_15} Voltage	
JTAG (Only)	3.3V	√	√	(3)	3.3V	Any(1)	Any(1)	V_{CCO_0}
	2.5V	√	√	(3)	2.5V	Any(1)	Any(1)	V_{CCO_0}
	1.8V	√	√	√	1.8V	Any(1)	Any(1)	GND
Serial, SPI, or SelectMAP	3.3V	√	√	N/A(1)	3.3V	3.3V	Any(1)	V_{CCO_0}
	2.5V	√	√	N/A(1)	2.5V	2.5V	Any(1)	V_{CCO_0}
	1.8V	√	√	√	1.8V	1.8V	Any(1)	GND
BPI	3.3V	√	√	N/A(1)	3.3V	3.3V	3.3V	V_{CCO_0}
	2.5V	√	√	N/A(1)	2.5V	2.5V	2.5V	V_{CCO_0}
	1.8V	√	√	√	1.8V	1.8V	1.8V	GND

Notes:

1. In the Virtex-7 FPGA, bank 14 and bank 15 are high-performance banks, limited to 1.8V or lower I/O standards.
2. JTAG interface is always supported in bank 0 at the V_{CCO_0} voltage level regardless of the configuration mode.
3. Virtex-7 HT devices support only 1.8V operation for bank 0.

Setting Configuration Options in the Vivado Tools

The choice of configuration voltage can be communicated to the Vivado tools by setting the CONFIG_VOLTAGE or CFGBVS properties. In addition, the CONFIG_MODE property can be defined so that the tools recognize which configuration pins are used. The Vivado tools provide warnings if there are any conflicts between configuration pin settings, such as an IOSTANDARD on a multi-function configuration pin that conflicts with the configuration voltage. These properties can be set in the Vivado configuration dialog (Edit Device Settings), or through Tcl commands. See UG912, *Vivado Properties Reference Guide* for details on the Tcl syntax. See UG899, *Vivado I/O and Clock Planning* for examples of how Vivado tools use these options.

External Master Configuration Clock (EMCCLK) Option

By default, the Master configuration modes use an internally generated configuration clock source CCLK. Using this clock option is convenient because an external clock generator source is not required. However, for applications where configuration time reduction is critical, the External Master Configuration Clock (EMCCLK) should be used. The EMCCLK clock allows the use of a more precise external clock source than the FPGA's internal clock with the master CCLK frequency tolerance ($F_{MCCKTOL}$). For example, when the master CCLK has a maximum frequency of 100 MHz, a 50% tolerance means that the ConfigRate setting cannot be faster than 66 MHz. However, an external clock source can be applied as fast as the specification allows. 7 series FPGAs support the ability to dynamically switch to an external clock source (EMCCLK) when in a Master mode.

Enable the external clock source option by:

1. Enabling the ExtMasterCclk_en bitstream generation option.
2. Defining the EMCCLK target voltage. The following methods can accomplish this:

- Bank 14 has another pin with an IOSTANDARD defined. The voltage defined on bank 14 is automatically applied to the EMCCLK.
- The EMCCLK signal is used in the design after configuration and has the IOSTANDARD defined.

3. Connecting EMCCLK on the board to your board's oscillator or other clock source.

Dedicated resources can divide the EMCCLK input by 2, 4, or 8 before the configuration logic, or use the full rate (divide by 1). The ExMasterCclk_en option is set in Vivado with the BITSTREAM.CONFIG.EXTMASTERCCLK_EN property. Refer to UG908, *Vivado Design Suite User Guide Programming and Debugging* for details.

```
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN Disable|div-8|div-4|
div-2|div-1
```

The default is Disable (use the internal CCLK).

The EMCCLK signal must be instantiated and used in the design providing the I/O standard definition as the EMCCLK is a multi-purpose pin, or the voltage level will be taken from another pin defined in bank 14. Lock the input to the EMCCLK pin location (see UG475, *7 Series FPGAs Packaging and Pinout Product Specification*). Connect the EMCCLK input to the oscillator or other clock source on the board. Use good signal integrity design practices, especially for very high-speed clocks, to avoid errors during configuration.

The configuration begins with the CCLK generated by the FPGA internal oscillator until the bitstream header is read. If the EMCCLK option is enabled then the FPGA switches from the internal oscillator to the clock found on the EMCCLK pin.

Serial Configuration Mode

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 7 series FPGA serial configuration interface.

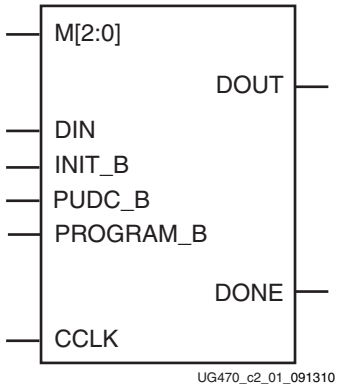
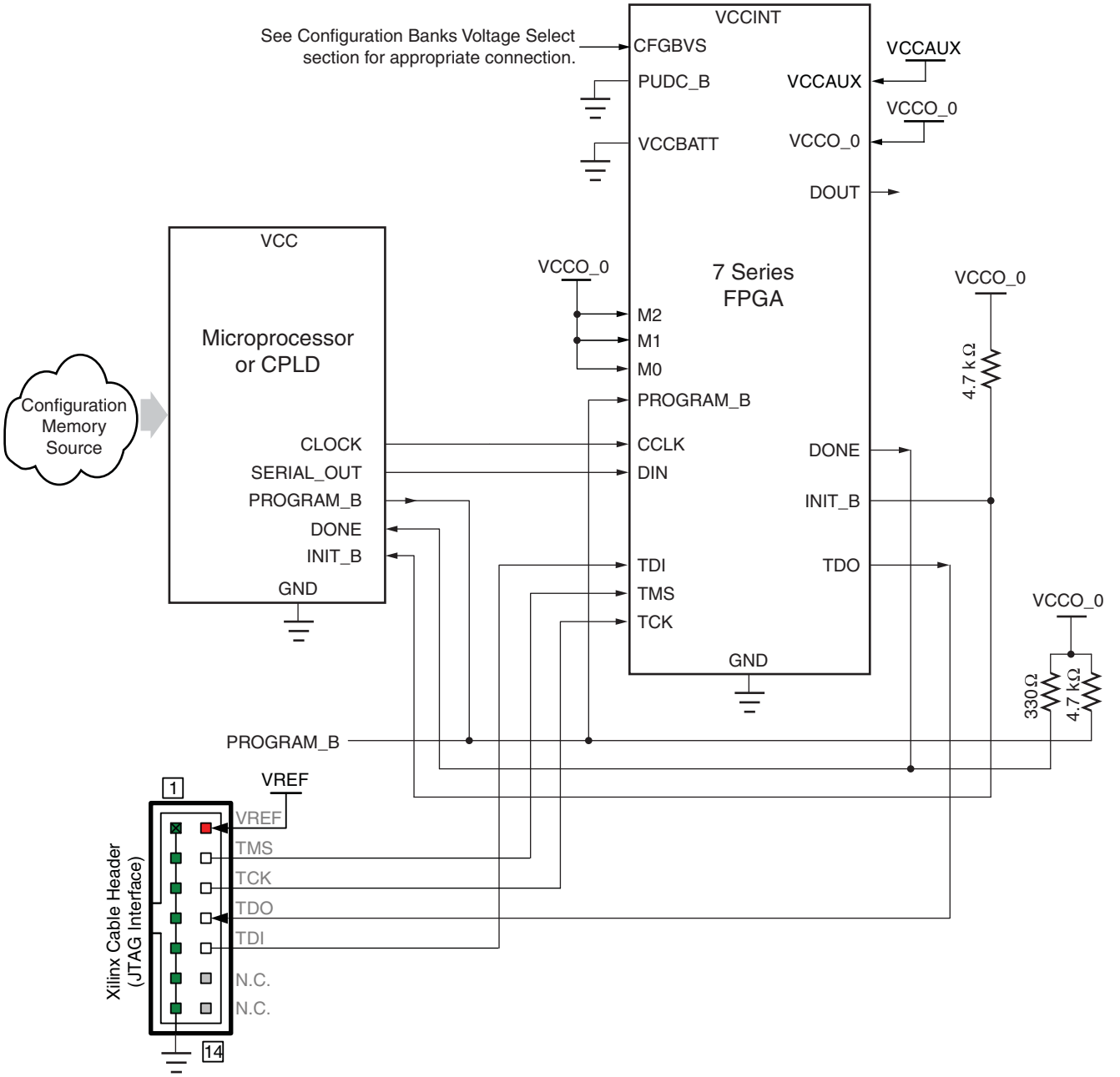


Figure 2-1: **7 Series FPGA Serial Configuration Interface**

The serial configuration interface pins shown in Figure 2-1 are defined in Table 2-4, page 24.

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 2-2). 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 39). For daisy chain information, see Chapter 9, Multiple FPGA Configuration.



Refer to the Notes following this figure for related information.

UG470_c2_02_021914

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

Notes relevant to Figure 2-2:

1. The 7 series FPGA V_{CCO_0} and the Xilinx Cable V_{REF} must have the same voltage.
2. The DONE pin is an open-drain output. See Table 2-4, page 24 for DONE signal details.
3. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
4. The bitstream startup clock setting must be set for CCLK for serial configuration.
5. CCLK signal integrity is critical.
6. V_{CCBATT} is the power source for the AES key stored in SRAM. It should be connected to a battery supply, when used.

Master Serial Configuration

The Master Serial configuration mode is the same as the Slave Serial configuration mode, except that the FPGA generates the CCLK. That is, the CCLK is an output in Master serial mode.

For the 7 series FPGAs, the Master SPI mode is the dominant configuration mode for a low-pin count configuration from a serial-type flash device.

The 7 series FPGAs support Master Serial mode for configuration from legacy serial PROMs (when applicable) or for custom, CPLD-based configuration state machines driven by the FPGA CCLK.

Clocking Serial Configuration Data

Figure 2-3 shows how configuration data is clocked into 7 series devices in Slave Serial and Master Serial modes.

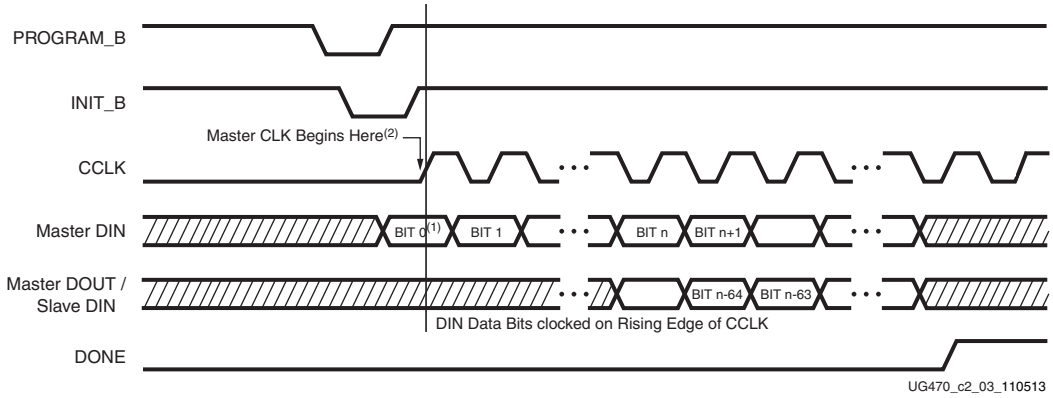


Figure 2-3: **Serial Configuration Clocking Sequence**

Notes relevant to Figure 2-3:

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. Data sheet timing is relative to the CCLK pin.
3. CCLK can be free-running in Slave Serial mode.

SelectMAP Configuration Mode

The SelectMAP configuration interface (Figure 2-4) provides an 8-bit, 16-bit, or 32-bit bidirectional data bus interface to the 7 series FPGA configuration logic that can be used for both configuration and readback. Readback and the read direction of the data bus are applicable only to Slave SelectMAP mode. For details, refer to Chapter 6, Readback and Configuration Verification. The bus width of SelectMAP is automatically detected (see Bus Width Auto Detection, page 76).

CCLK is an output in Master SelectMAP mode and an input in Slave SelectMAP mode. One or more 7 series devices can be configured through the SelectMAP bus.

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

- Single-device Slave SelectMAP

  Typical setup includes a processor providing data and clock. Alternatively, another programmable logic device, such as a CPLD, can be used as a configuration manager that configures the FPGA through the FPGA's Slave SelectMAP interface.

- Multiple-device daisy-chain SelectMAP bus

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

- Multiple-device ganged SelectMAP

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

The basic Master SelectMAP and Slave SelectMAP configuration methods are described in this chapter. The multiple-device configuration methods are described in Chapter 9. For information on the SelectMAP simulation model, see UG626 *Synthesis and Simulation Design Guide*, Chapter 6 *Simulating Your Design*.

BPI is the recommended configuration mode when using parallel flash (see Master BPI Configuration Interface, page 55).

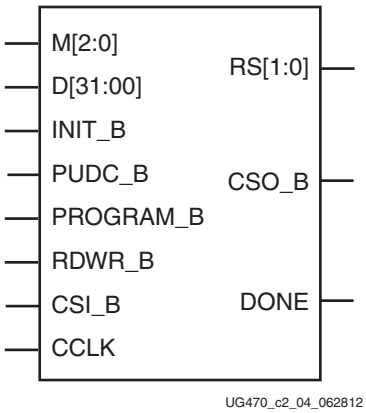


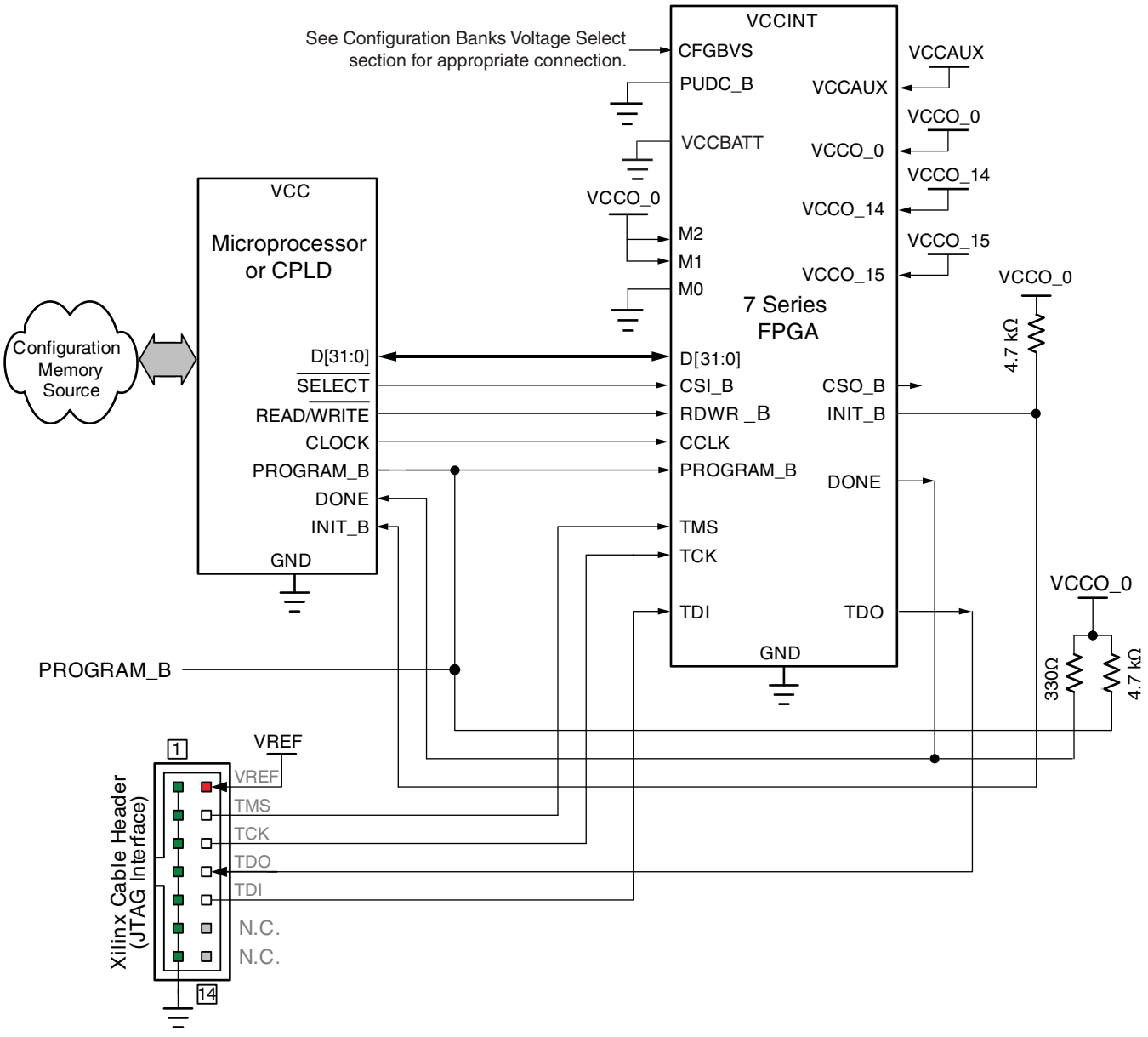
Figure 2-4: **7 Series FPGA SelectMAP Configuration Interface**

The SelectMAP configuration interface pins shown in Figure 2-4 are defined in Table 2-4, page 24.

Single Device SelectMAP Configuration

Microprocessor-Driven SelectMAP Configuration

For custom applications where a microprocessor or CPLD is used to configure a single 7 series device, either Master SelectMAP mode (use CCLK from the FPGA) or Slave SelectMAP mode can be used (Figure 2-5). Slave SelectMAP mode is preferred. See XAPP583, *Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave SelectMAP Mode*, for information on configuring Xilinx FPGAs using a microprocessor.



Refer to the Notes following this figure for related information.

UG470_c2_05_072114

Figure 2-5: **Single Slave Device SelectMAP Configuration from Microprocessor or CPLD Example**

Notes relevant to Figure 2-5:

1. Refer to XAPP583, *Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave 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 7 series FPGA V_{CCO_0} supply input and the Xilinx Cable V_{REF} must have the same voltage.
3. The DONE pin is an open-drain output. See Table 2-4, page 24 for DONE signal details.
4. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
5. The bitstream startup clock setting must be set for CCLK for SelectMAP configuration.
6. 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.
7. CCLK signal integrity is critical.
8. V_{CCBATT} is the power source for the AES key stored in SRAM. It should be connected to a battery supply, when used.
9. The Data bus width can be x8, x16, or x32 for Slave SelectMAP configuration. The Slave SelectMAP x16 and x32 bus widths do not support AES-encrypted bitstreams.

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, and CCLK signals.

CSI_B

The Chip Select input (CSI_B) enables the SelectMAP bus. When CSI_B is High, the 7 series device ignores the SelectMAP interface, neither registering any inputs nor driving any outputs. The D[31:0] pins 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.

If only one device is being configured through the SelectMAP and readback is not required, the CSI_B signal can be tied to ground.

RDWR_B

RDWR_B is an input to the 7 series 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 6, 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. If readback is not

needed, RDWR_B can be tied to ground or used for debugging with SelectMAP ABORT. See the SelectMAP ABORT section.

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 45).

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). RDWR_B must be driven Low before CSI_B is asserted, otherwise an ABORT occurs.

On the next rising CCLK edge, the device begins sampling the data pins. Only D[7:0] are sampled by Configuration until the bus width is determined. See Bus Width Auto Detection, page 76 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 5). 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 5 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 2-6 summarizes the timing of SelectMAP configuration with continuous data loading.

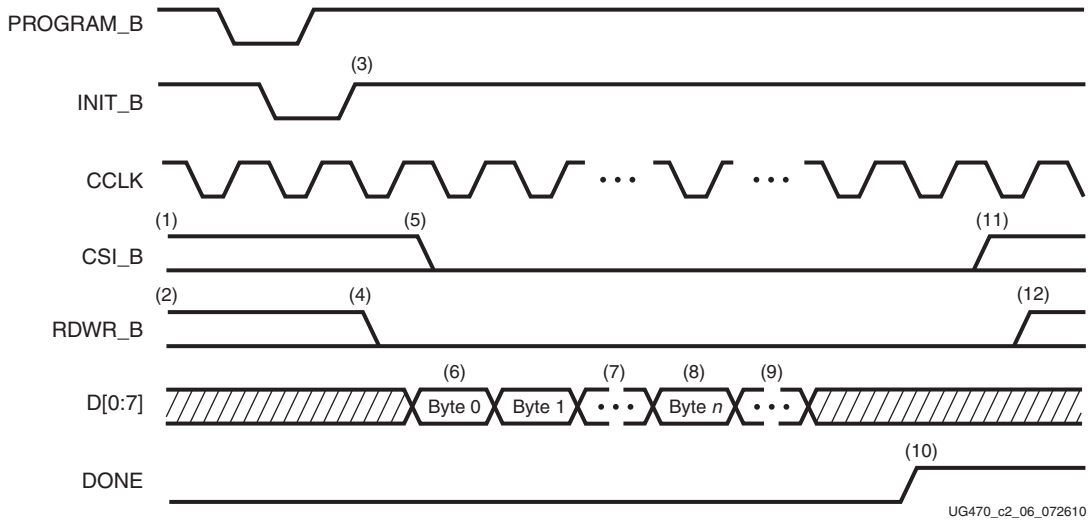


Figure 2-6: **Continuous x8 SelectMAP Data Loading**

Notes relevant to Figure 2-6:

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.
3. The Mode pins are sampled when INIT_B goes High. The Status register is updated to indicate x8 mode.
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 5.)
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 5.)
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). The Status register is updated after the bus width detection sequence.

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 2-7) or by halting CCLK (Controlled CCLK method, Figure 2-8).

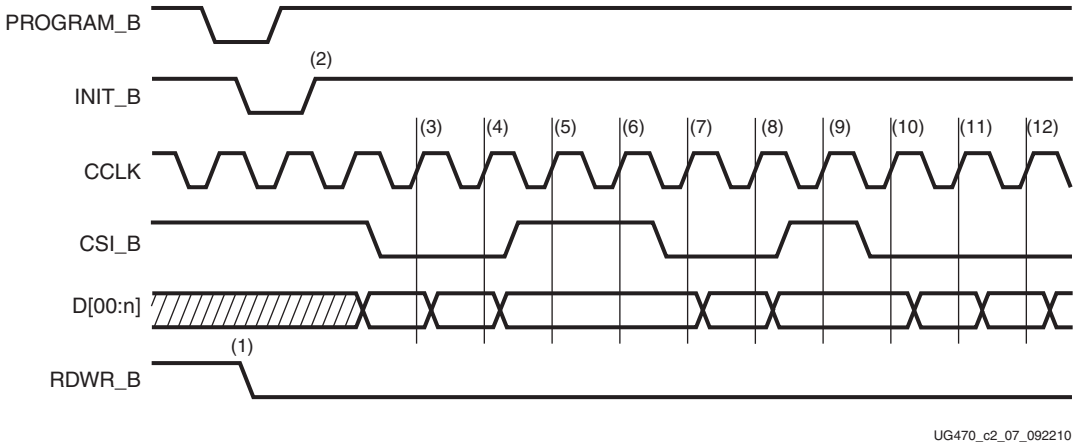


Figure 2-7: **Non-Continuous SelectMAP Data Loading with Free-Running CCLK**

Notes relevant to Figure 2-7:

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.
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.

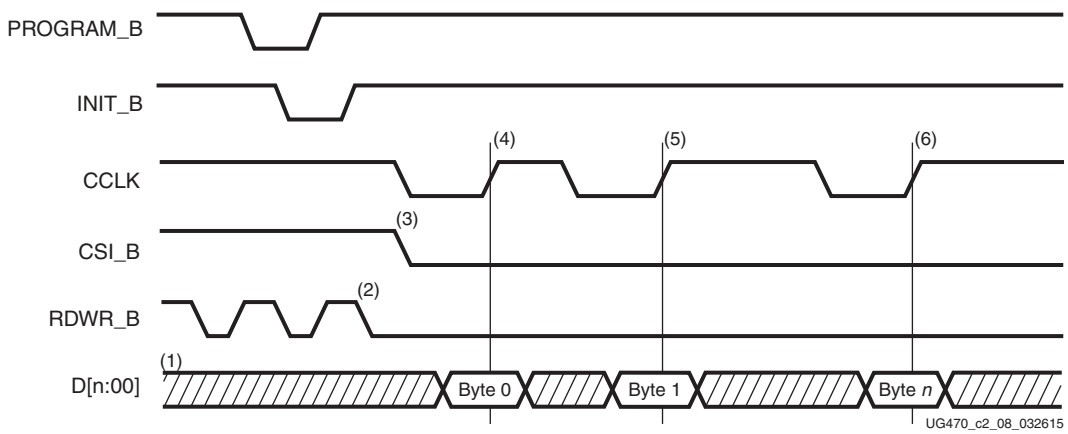


Figure 2-8: **Non-Continuous SelectMAP Data Loading with Controlled CCLK**

Notes relevant to Figure 2-8:

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 79. This convention can be a source of confusion when designing custom configuration solutions. Table 2-10 shows how to load the hexadecimal value `0xABCD` into the SelectMAP data bus.

Table 2-10: **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[00:07] 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 75).

Table 2-11 shows the bit ordering for the 7 series FPGA SelectMAP x8, x16, and x32 data bus widths. The 7 series FPGA SelectMAP data bus bit ordering is the same as the Virtex-6 FPGA SelectMAP data bus bit ordering.

Table 2-11: **Bit Ordering**

SelectMAP Data Bus Width	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

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[04:07] 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 Table 2-8.

Configuration Abort Sequence Description

An ABORT is signaled during configuration as follows:

1. The configuration sequence begins normally.
2. Pull the RDWR_B pin High synchronous to CCLK while the device is selected (CSI_B asserted Low).
3. 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 2-9.

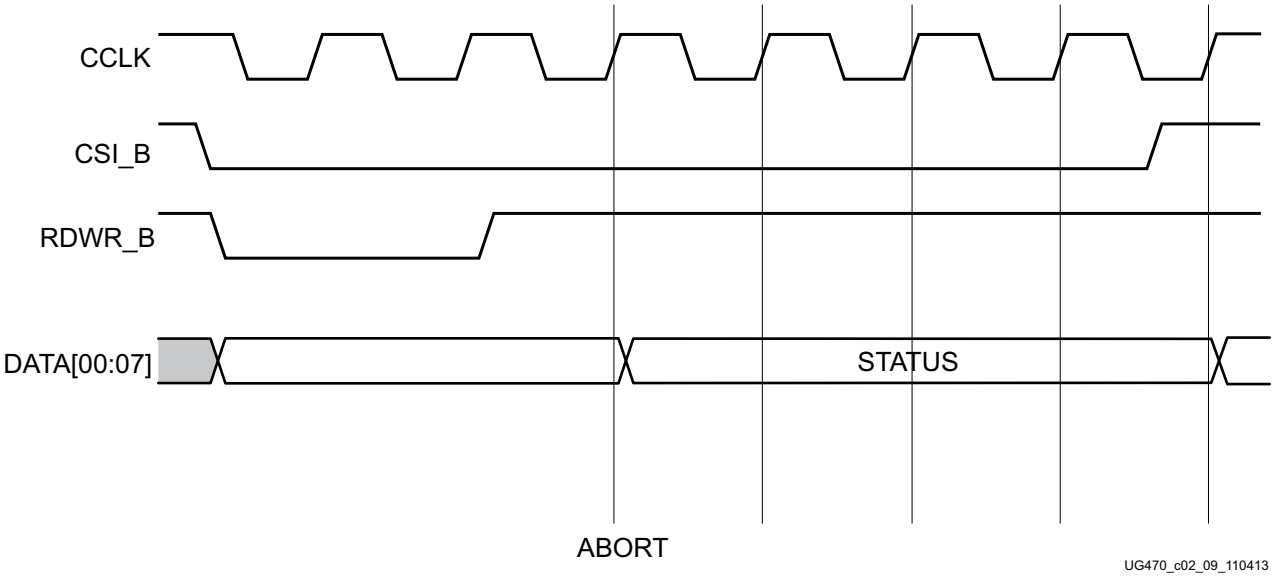


Figure 2-9: **Configuration Abort Sequence for SelectMAP Modes**

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. The ABORT ends when CSI_B is deasserted.

See Figure 2-10.

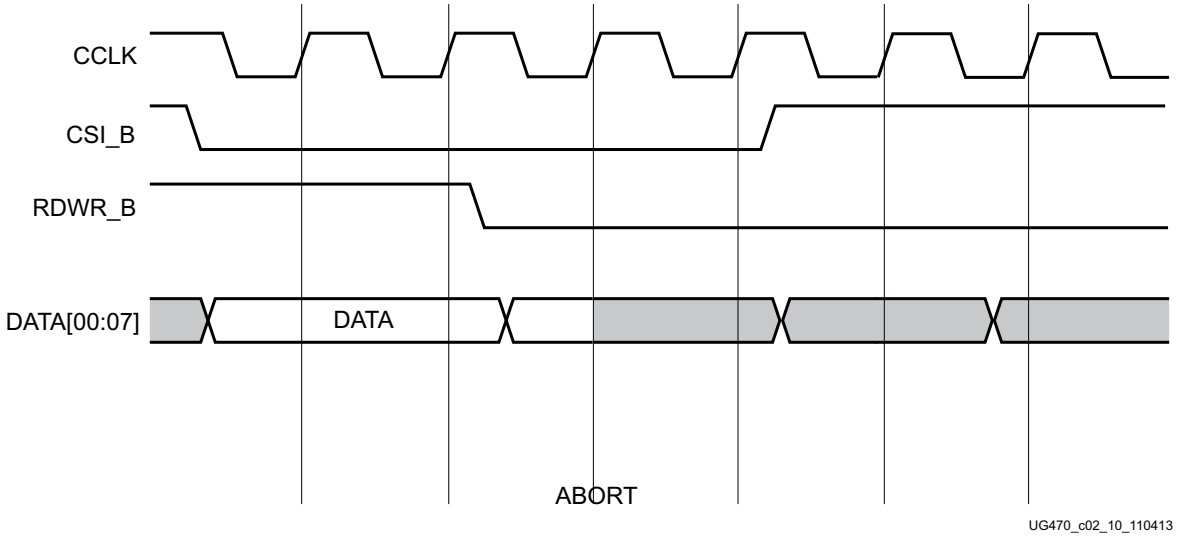


Figure 2-10: **Readback Abort Sequence**

ABORTs during readback are not followed by a status word because the RDWR_B signal is set for write control (FPGA D[x:00] pins are inputs).

ABORT Status Word

During the configuration ABORT sequence, the device drives a status word onto the D[00:07] pins. The status bits do not bit-swap. The other data pins are always High. The key for the status word is given in Table 2-12.

Table 2-12: **ABORT Status Word**

Bit Number	Status Bit Name	Meaning
D07	CFGERR_B	Configuration error (active Low) 0 = A configuration error has occurred. 1 = No configuration error.
D06	DALIGN	Sync word received (active High) 0 = No sync word received. 1 = Sync word received by interface logic.
D05	RIP	Readback in progress (active High) 0 = No readback in progress. 1 = A readback is in progress.
D04	IN_ABORT_B	ABORT in progress (active Low) 0 = Abort is in progress. 1 = No abort in progress.
D03-D00	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.

Master SPI Configuration Mode

The 7 series FPGA Master SPI configuration mode enables the use of low pin-count, industry-standard SPI flash devices for bitstream storage. The FPGA supports a direct connection to the de facto standard, four-pin interface of a SPI flash device for reading a stored bitstream.

The 7 series FPGA Master SPI configuration mode (Figure 2-11) can optionally read from SPI devices that support x2 and x4 Fast Output Read operations. These output modes are proportionally faster than the standard 1-bit SPI interface. In addition, a negative edge clocking mode is available to make better use of the entire clock period and allow higher configuration speed. SPI flash densities over 128 Mb requiring 32-bit addressing are also supported. (See SPI Densities over 128 Mb, page 53 for additional instructions and limitations.) For information on using SPI configuration with Stacked Silicon Interconnect (SSI) devices, see Stacked Silicon Interconnect, page 18.

Figure 2-12, page 51 shows the connections for a SPI configuration with a x1 or x2 data width. These connections are the same because the x2 mode uses the D00_MOSI pin as a dual-purpose Data In/Out pin. Daisy-chained configuration mode is only available in SPI x1 mode. The FPGA pin connections to the SPI flash involved in the Master SPI mode are listed in Table 2-4, page 24.

The iMPACT programming software provides the ability to program a SPI serial flash using an indirect programming method. This downloads a new FPGA design that provides a connection from the iMPACT software through the 7 series device to the SPI flash. Previous FPGA memory contents are lost during this operation. ISE iMPACT indirect programming supports the following SPI flash: Micron N25Q 3.3V and 1.8V, Winbond W25Q, and Spansion S25FL. For the specific densities supported by the programming tools, please consult the iMPACT Help documentation. Similar capability is offered by the Vivado Device Programmer.

For additional details on the SPI x1, x2, and x4 operation, including reference schematics and programming instructions, see XAPP586, *Using SPI Flash with 7 Series FPGAs*.

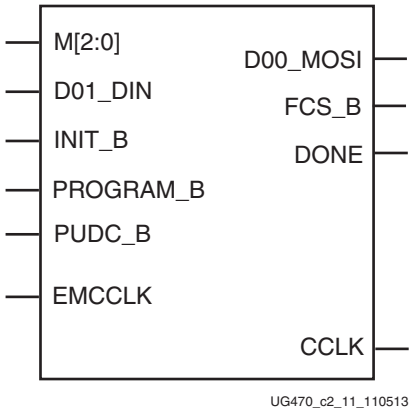
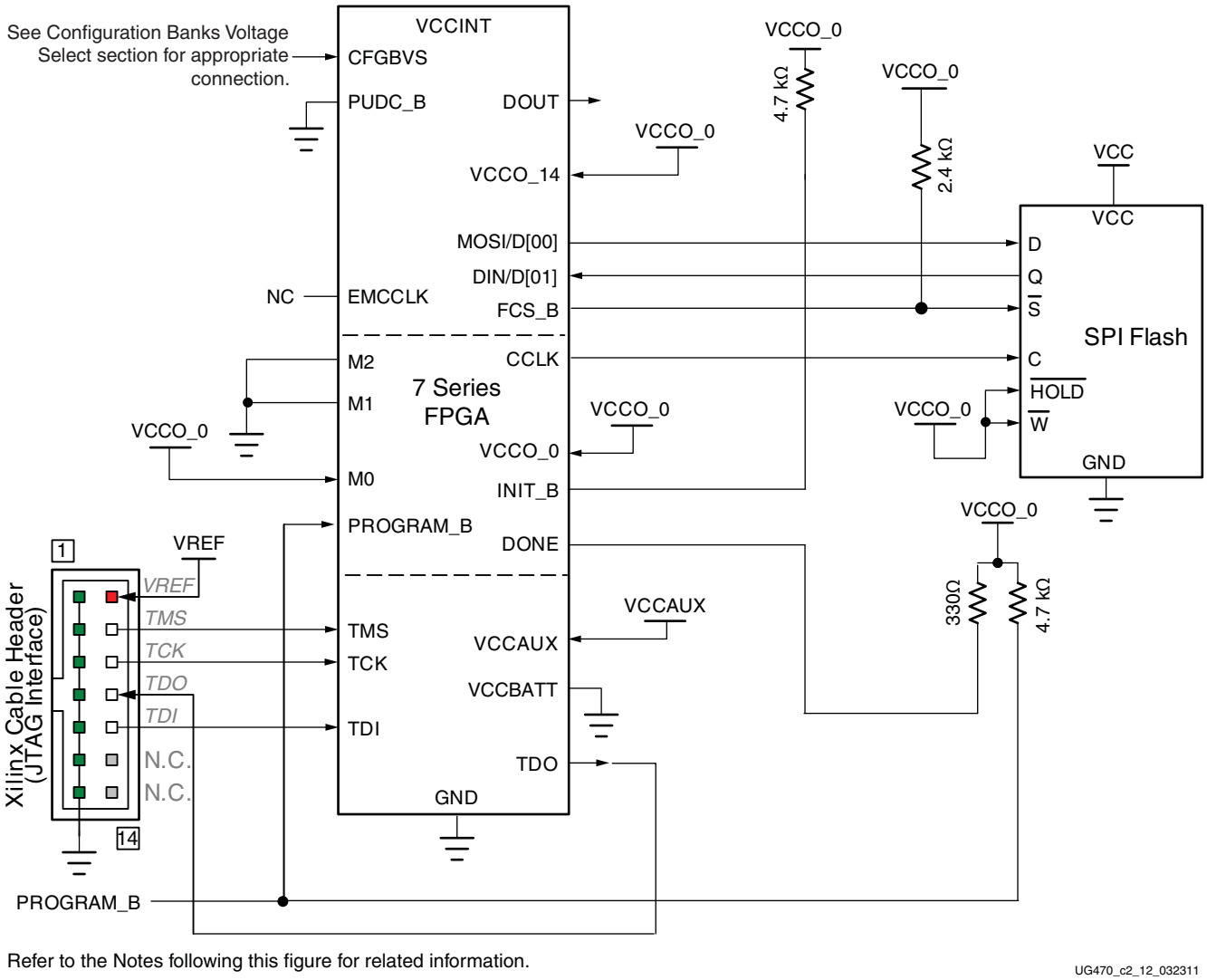


Figure 2-11: **7 Series FPGA SPI Configuration Interface**

The SPI configuration interface pins shown in Figure 2-11 are defined in Table 2-4, page 24.



Refer to the Notes following this figure for related information.

UG470_c2_12_032311

Figure 2-12: **7 Series FPGA SPI x1/x2 Configuration Interface**

Notes relevant to Figure 2-12:

1. The DONE pin is an open-drain output. See Table 2-4, page 24 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. The bitstream startup clock setting must be set for CCLK for SPI configuration.
4. CCLK signal integrity is critical.
5. DOUT should be connected to the DIN of the downstream FPGA for daisy-chained SPI x1 configuration mode. Daisy-chaining is not supported for x2 or x4 SPI modes.
6. A series resistor should be considered for the datapath from the flash to the FPGA to minimize overshoot. The proper resistor value can be determined from simulation.
7. The 7 series FPGA V_{CCO_0} supply must be compatible with the V_{CC} for the I/O of the SPI device.

8. Data is clocked out of the SPI on the CCLK falling edge and clocked in on the FPGA on the rising edge, unless negative edge clocking (spi_fall_edge:Yes) is enabled.
9. The CCLK frequency is adjusted by the **`ConfigRate`** option if the source is the internal oscillator. Alternatively, the ExtMasterCclk_en option can switch the CCLK to source from the EMCCLK pin to use an external clock source.
10. V_{CCBATT} is the power source for the AES key stored in SRAM. It should be connected to a battery supply, when used.

The 7 series FPGA SPI x1 mode sequence diagram is shown in Figure 2-13.

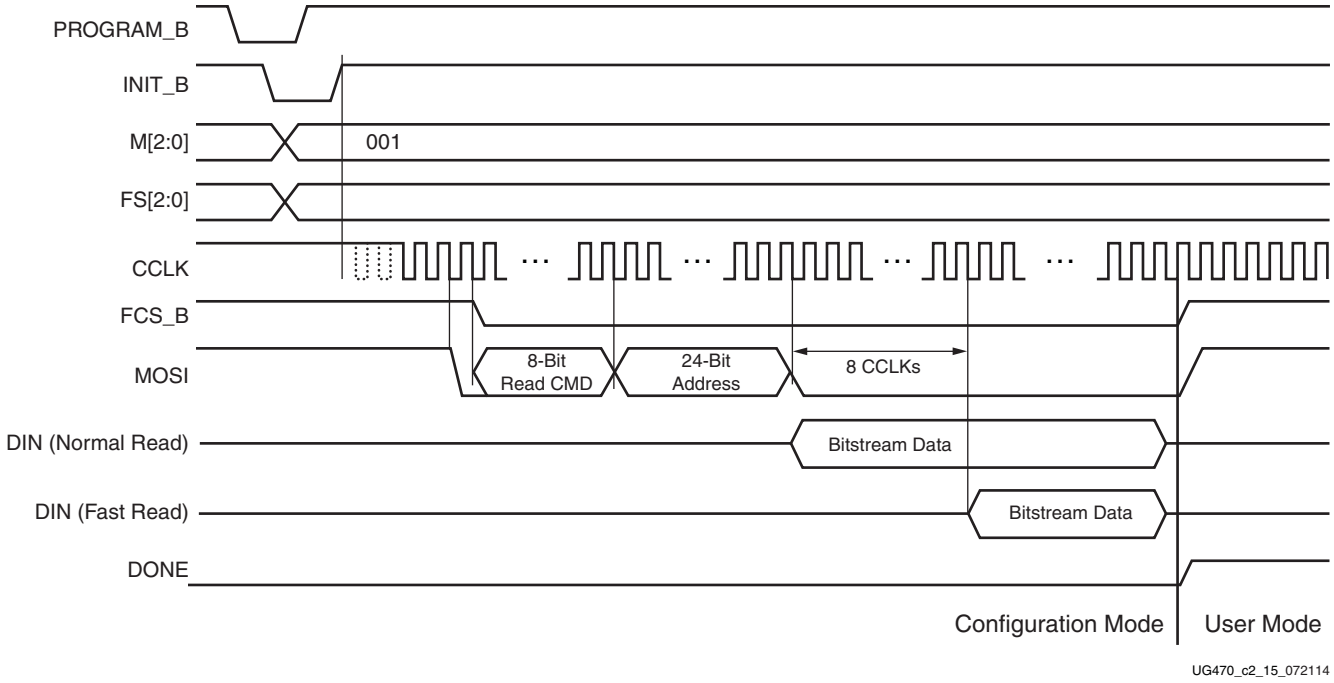


Figure 2-13: **7 Series FPGA SPI x1 Mode Sequence**

Notes relevant to Figure 2-13:

1. Waveforms represent the relative sequence of events and are not to scale. See SPI flash memory data sheet for detailed SPI command and data timing.
2. The CCLK frequency starts at 3 MHz typical providing sufficient margin for the FCS_B and MOSI activity. The CCLK frequency increases if the **`ConfigRate`** option 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.
3. When the SPI flash is not used after configuration, it is recommended to have the FPGA design drive the FCS_B High.

Master SPI Dual (x2) and Quad (x4) Read Commands

The Master SPI configuration mode in 7 series FPGAs supports Dual and Quad fast read operations. The FPGA first transmits the Fast Read opcode (`0Bh`) to the SPI then reads in the command to change the data width in the early part of the bitstream. The FPGA then issues a new read command for the Dual or Quad fast read operation to the SPI device and begins to read the appropriate data width.

To enable this configuration mode in software, the **spi_buswidth** option is used to insert the appropriate bitstream commands. The SPI read opcodes supported by the 7 series FPGAs are shown in Table 2-13.

Table 2-13: **SPI Instructions and Required Opcodes**

SPI Instruction	Opcode
Fast Read x1	0B
Dual Output Fast Read	3B
Quad Output Fast Read	6B
Fast Read, 32-bit address	0C
Dual Output Fast Read, 32-bit address	3C
Quad Output Fast Read, 32-bit address	6C

SPI Densities over 128 Mb

SPI flash densities over 128 Mb require more than the traditional 24-bit addressing that is standard for SPI devices. SPI vendors use various methods to support 32-bit addressing, and the solution supported by the 7 series FPGA requires the SPI to boot up in a 24-bit addressing mode and have dedicated opcodes for read instructions in 32-bit addressing mode. The software option **spi_32bit_addr** is used to generate a bitstream that can address flash densities over 128 Mb. This option must be used consistently for all bitstreams used for fallback MultiBoot such that all bitstreams have the option enabled if the SPI flash is over 128 Mb or all bitstreams have it disabled if the SPI flash is 128 Mb or lower.

Valid SPI devices must support the instructions in Table 2-13 to interface with the 7 series FPGA.

SPI Configuration Timing

SPI flash devices clock data out on the falling edge and by default, the 7 series FPGA clocks data in on the rising edge. This results in a lost half cycle that limits the maximum clock speed of the configuration solution (Figure 2-14). To gain maximum use of the clock period, the 7 series FPGA can be modified to clock data in on the falling edge.

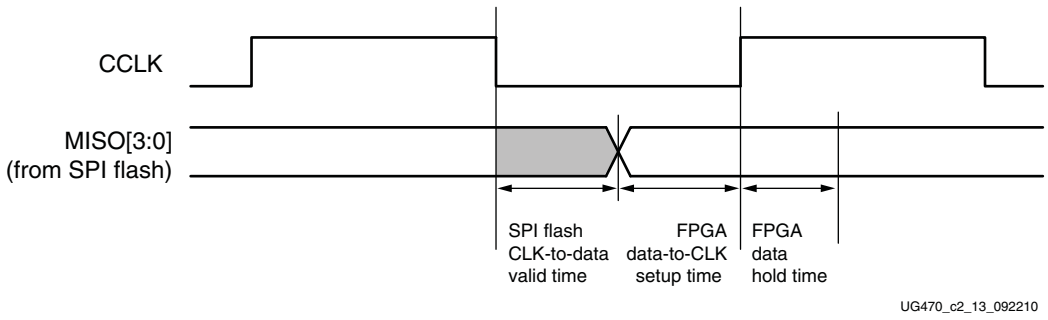


Figure 2-14: **Basic SPI Configuration Timing**

When configuration starts, the FPGA clocks data in on the rising edge. This continues until the FPGA reads the command in the early part of the bitstream that instructs it to change to the falling edge. This occurs before the command to change to external clocking or the

command to change the master clock frequency. The falling edge clocking option is enabled by setting the option **spi_fall_edge**.

Determining the Maximum Configuration Clock Frequency

In Master SPI mode, the FPGA delivers the configuration clock. The FPGA's master configuration clock frequency is set through the **ConfigRate** option. The **ConfigRate** option sets the nominal configuration clock frequency.

The 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 (ConfigRate)
- FPGA master CCLK frequency tolerance ($F_{MCCKTOL}$)
- SPI clock to out (T_{SPITCO})
- FPGA data setup time (T_{SPIDCC})

To maximize performance, the FPGA needs to use the falling edge clocking mode to take advantage of the entire clock period (see SPI Configuration Timing). The following details assume this option (spi_fall_edge) has been enabled.

The FPGA's master configuration clock has a tolerance of TMCCKTOL. Due to the master configuration clock tolerance (TMCCKTOL), the **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, SPI clock to out, and FPGA setup time, as shown in Equation 2-2.

$$\frac{1}{ConfigRate \times (1 + FMCCKTOL_{MAX})} \geq T_{SPITCO} + T_{SPIDCC} \qquad \textit{Equation 2-1}$$

The wide frequency tolerance of the FPGA master configuration clock is a significant factor in this calculation, and if maximum configuration speeds are needed, it is recommended to use an external clock to minimize the impact of that variable. This requires connection to the EMCCLK pin and enabling this option in the bitstream (ExtMasterClk_en).

Power-on Sequence Precautions

At power-on, the FPGA automatically starts its configuration procedure. When the FPGA is in Master Serial 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 awaken or start 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 (T_{POR}) 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 effect of the power sequence, the power ramps, FPGA power-on reset timing, and SPI flash power-up

timing on the timing relationship between the start of FPGA configuration and the readiness of the SPI flash. Refer to the respective 7 series FPGAs data sheet, for FPGA power supply requirements and timing, and check the SPI flash data sheet for the SPI flash power-up timing requirements.

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 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. Release the INIT_B pin to High after the SPI flash becomes ready to receive commands.

Master BPI Configuration Interface

The 7 series FPGA Master BPI configuration mode (Figure 2-15) enables the use of industry-standard parallel NOR (BPI) flash devices for bitstream storage. The FPGA supports a direct connection to the address, data, and control signals of a BPI flash for extracting a stored bitstream.

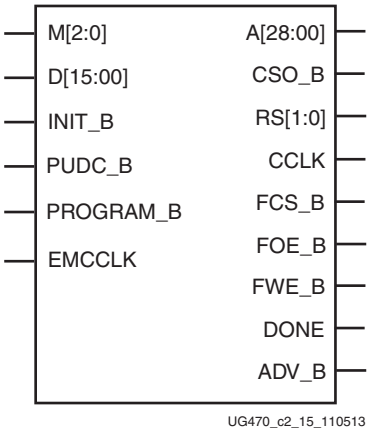


Figure 2-15: **7 Series FPGA Master BPI Configuration Interface**

The BPI configuration interface pins shown in Figure 2-15 are defined in Table 2-4, page 24. Note that some of the required pins are in bank 15. The CPG236 package for the Artix-7 7A15T, 7A35T and 7A50T devices does not bond out bank 15 and therefore does not support BPI configuration.

The 7 series FPGA Master BPI configuration mode has two BPI flash read modes available: asynchronous and synchronous. Faster configuration times can be achieved using the BPI flash synchronous read mode with the external master clock than in other direct configuration modes. In addition, a wider density range of parallel NOR flash can be accessed by up to 29 address lines.

By default, 7 series FPGAs use the asynchronous mode of the BPI flash to read bitstream data as shown in Figure 2-16, page 57. The FPGA drives the address bus from a given start address, and the BPI flash sends back the bitstream data. The default start address is address 0. The start address can be explicitly set in a MultiBoot reconfiguration procedure as outlined in Chapter 7, Reconfiguration and MultiBoot. In asynchronous read mode,

supported bus widths of x8 and x16 are auto-detected as described in Bus Width Auto Detection, page 76.

The 7 series FPGA Master BPI configuration mode can optionally read a bitstream from select BPI devices that support burst, synchronous read mode as illustrated in Figure 2-19 and described in Synchronous Read Mode Support, page 61. A BPI device that supports the synchronous read mode latches a given start address from the FPGA into its internal address counter. Then given a clock, the flash outputs data from the next sequential address location to its data bus during each clock period. In the synchronous read mode, a BPI device can deliver data many times faster than through its asynchronous read interface.

The iMPACT programming software provides the ability to program parallel NOR flash using an indirect programming method. This method downloads a new FPGA design that provides a connection from the iMPACT software through the 7 series FPGA to the BPI flash device. For more details, see XAPP587, *BPI Fast Configuration and iMPACT Flash Programming with 7 Series FPGAs*. Similar functionality will be offered by the Vivado device programmer. In embedded systems such as a PCIe interface, no JTAG connection may be available. In-system programming can be accomplished across the PCIe system instead. For more details, see XAPP518, *In-System Programming of BPI PROM for Virtex-6, Virtex-7, and Kintex-7 FPGAs Using PCI Express Technology*.

Table 2-14 lists supported parallel NOR flash families for the 7 series FPGAs. Refer to the specific FPGA for the number of address signals available that determine the maximum flash density supported for configuration.

Table 2-14: **Supported Parallel NOR Flash Families**

Vendor	Family	Density	Read Modes Supported for Configuration
Micron	P33 (Axcell™)	64 Mb - 1 Gb	Synchronous/Asynchronous
Micron	P30 (StrataFlash™, Axcell)	64 Mb - 1 Gb	Synchronous/Asynchronous
Micron	M29EW	64 Mb - 1 Gb	Asynchronous
Micron	G18F	128 Mb - 1 Gb	Synchronous/Asynchronous
Spansion	S29GLxxxP	128 Mb - 1 Gb	Asynchronous
Spansion	S29GLxxxS	128 Mb - 1 Gb	Asynchronous

Asynchronous Read Mode Support

In the Master BPI configuration mode, the 7 series FPGAs use the BPI flash asynchronous read mode by default to read bitstream data. After power-up, the Mode pins, M[2:0], are sampled when the FPGA's INIT_B output goes High. The Mode pins must be defined at the valid logic levels (Master BPI Configuration mode M[2:0] = `010`) at this time. The PUDC_B pin must remain at a constant logic level throughout the FPGA configuration. After the Master BPI configuration mode is determined, the FPGA drives the flash control signals (FWE_B High, FOE_B Low, and FCS_B Low). Although the CCLK output is not connected to the BPI flash device for BPI flash asynchronous read mode, 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 the CCLK pin as a reference. In the Master BPI mode, the address starts at 0 and increments by 1 until the DONE pin is asserted. If the address reaches the maximum value (`29'h1FFFFFFF`) and configuration is

not done (DONE is not asserted), an error flag is raised in the status register, and fallback reconfiguration starts. 7 series FPGA BPI mode also supports asynchronous page-mode reads to allow an increase in the CCLK frequency. See Page Mode Support, page 60 for details.

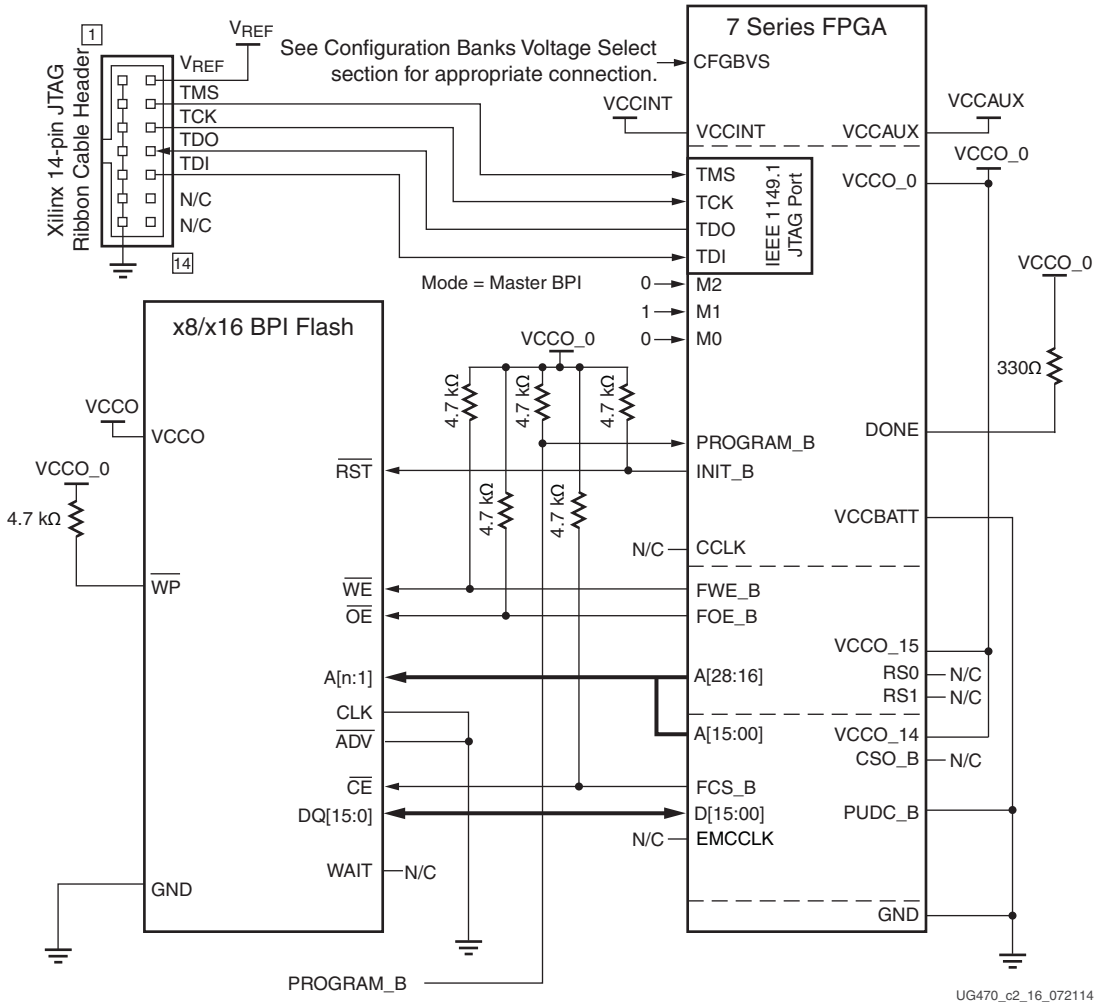


Figure 2-16: **7 Series FPGA Master BPI Configuration Interface - Asynchronous Read Example**

Notes relevant to Figure 2-16:

1. 7 series FPGA V_{CCO_0} supply input and Xilinx Cable V_{REF} must be tied to the same voltage.
2. 7 series FPGA bank voltage V_{CCO_14} supplies: A[15:00], FCS_B, D[15:00], EMCCLK, PUDC_B, and CSO_B signals. Bank voltage V_{CCO_15} supplies: A[28:16], FWE_B, FOE_B, ADV_B, RS0, and RS1 signals.
3. M[2:0] = `010` for BPI mode.
4. Figure 2-16 shows the x16 BPI interface. For x8 BPI interfaces, only D[07:00] are used. See Bus Width Auto Detection, page 76.
5. Sending a bitstream to the data pin follows the same bit-swapping rule as in SelectMAP mode. See Parallel Bus Bit Order, page 79.

6. The CCLK output is not used to connect to flash in the asynchronous read mode, but it is used to sample flash read data during configuration. All timing is referenced to CCLK. The CCLK pin must not be driven or tied High or Low.
7. The RS[1:0] pins are not connected as shown in Figure 2-16. These output pins are optional and can be used for MultiBoot configuration. See Chapter 7, Reconfiguration and MultiBoot.
8. The DONE pin is an open-drain output. See Table 2-4, page 24 for DONE signal details.
9. The BPI flash vendor data sheet should be referred to for details on the specific flash signal connectivity. To prevent address misalignment, the user should pay close attention to the flash family address LSB for the byte/word mode used. Not all flash families use the A01 as the address LSB.
10. 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.
11. The FPGA mode (M[2:0]) pins are shown set to Master BPI 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 configuration during debug.
12. The FPGA PUDC_B pin is tied to ground in this sample schematic enabling non-dedicated configuration I/O's internal pull-ups during configuration. PUDC_B can alternatively be tied High setting the non-dedicated configuration I/Os to 3-state during configuration.
13. V_{CCBATT} is the power source for the AES key stored in SRAM. It should be connected to a battery supply, when used.
14. This sample schematic supports single bitstream configuration. Thus, FPGA RS[1:0] pins are not connected in this sample schematic.
15. See the respective 7 series FPGAs data sheet for the V_{CCINT} supply voltage.

Figure 2-17 shows the Master BPI configuration waveform for an asynchronous read.

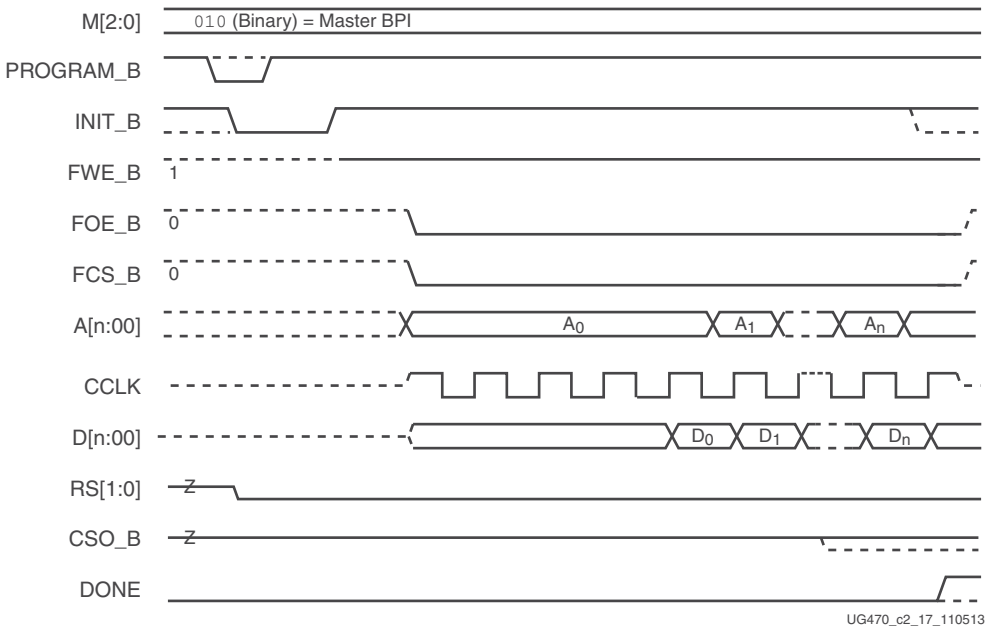


Figure 2-17: **7 Series FPGA Master BPI Configuration—Asynchronous Read Waveform**

Notes relevant to Figure 2-17:

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 configuration 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 after a T_{ICCK} delay after the rising edge of INIT_B.
7. The FPGA drives the initial address (A00) through its A[n:00] pins and holds the initial address for at least 10 CCLK cycles. For a power-on configuration, the initial address is `0x00000000`. 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. Only parallel daisy chains are supported from the BPI mode. See Chapter 9, Multiple FPGA Configuration.
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 internal 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. Multi-function configuration I/O switches to User mode after the GTS_cycle. By default, this is one cycle after DONE goes High.

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 7 series 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 7 series FPGA bitstream. 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 (01001), page 111). The COR1 register contains BPI flash page mode control bits (see Configuration Options Register 1 (01110), page 114). After the COR1 register is programmed, the BPI address timing switches at the page boundary as shown in Figure 2-18. When the SWITCH command is received, the master CCLK switches to a user-desired frequency, using it to load the rest of the configuration. See the BitGen section of UG628, *Command Line Tools User Guide* for details on BitGen options.

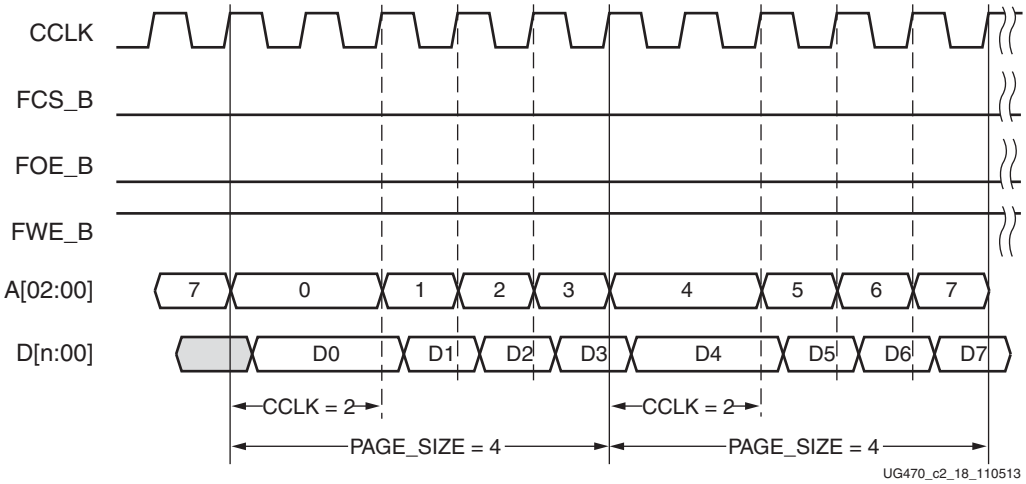


Figure 2-18: **BPI Waveforms (Page Size = 4 and First Access CCLK = 2)**

Notes related to Figure 2-18:

1. Figure 2-18 shows BPI 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).

Synchronous Read Mode Support

The 7 series FPGA Master BPI configuration mode with synchronous read is the fastest direct flash configuration option without the need for customized external control logic.

The Master BPI synchronous read mode (Figure 2-19) supports the flash families in Table 2-14 with synchronous support. The FPGA starts in asynchronous read mode, using the default CCLK frequency, and the bitstream header determines if the read mode continues asynchronously or if it switches to the faster synchronous read mode. Bitstream commands initiate the switch from asynchronous read to synchronous read if the **`BPI_sync_mode`** option is set. There are two available settings for the option: Type1 or Type2. Type1 is used to set the G18F flash family synchronous and latency bits and Type2 is used to set the P30. The switch to synchronous mode is done by the FPGA controller, which performs an asynchronous write to the BPI flash configuration register to set the device into synchronous mode and initiate a bitstream reread. To support the synchronous read mode, the FPGA CCLK output is connected to the BPI flash device, and the ADV_B FPGA signal must be connected to the Micron flash $\overline{\text{ADV}}$ signal.

The BPI flash configuration register synchronous bit setting is volatile and is cleared at power down or when a reset is issued to the BPI flash by FPGA INIT_B going Low.

After FPGA configuration is done, the BPI flash remains in synchronous read mode.

For use of synchronous read mode with the fallback feature, see the Initial MultiBoot Design Considerations section for golden image design considerations affecting synchronous read mode.

For additional details on the BPI synchronous read mode, including a reference schematic and programming instructions, see XAPP587, *BPI Fast Configuration and iMPACT Flash Programming with 7 Series FPGAs*.

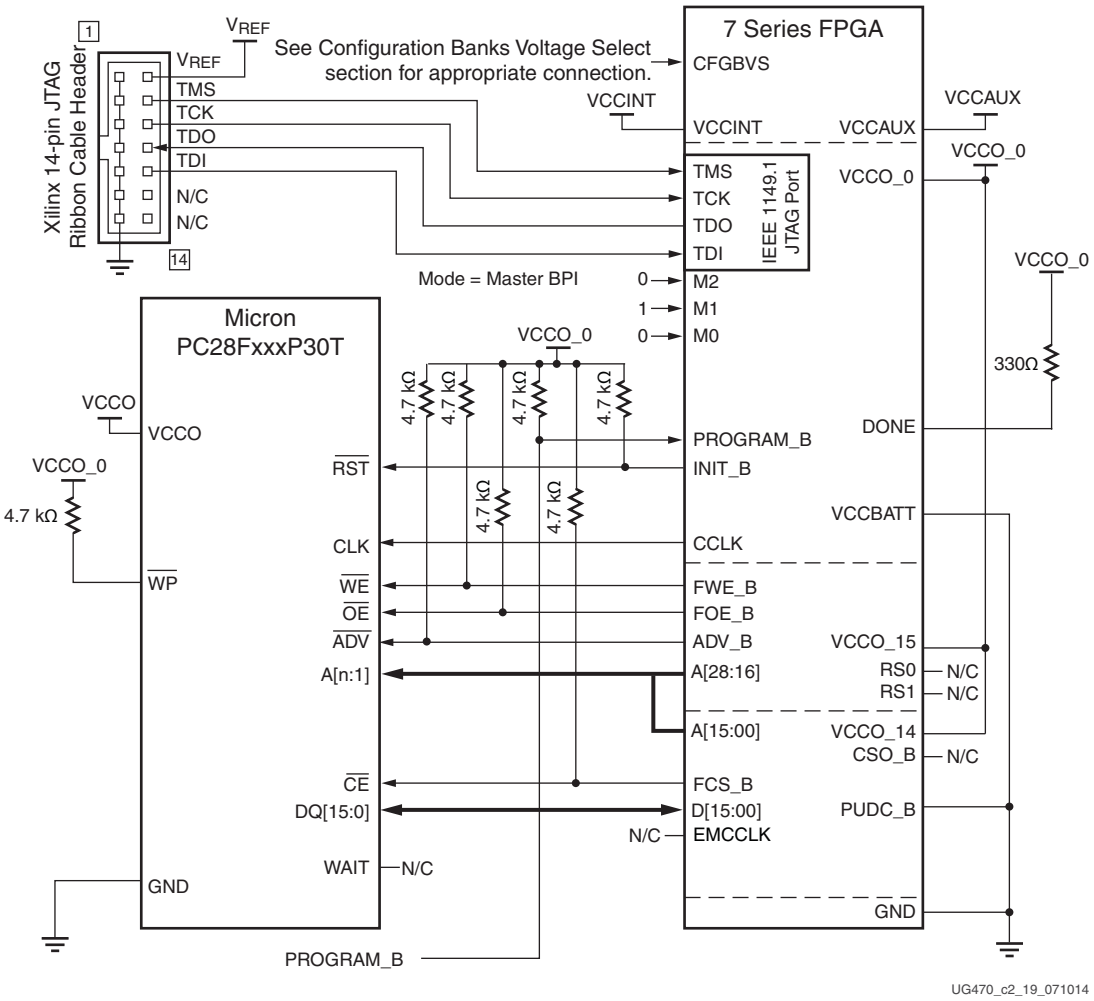


Figure 2-19: **7 Series FPGA Master BPI Configuration Interface Synchronous Read Example**

Notes relevant to Figure 2-19:

1. The 7 series FPGA synchronous read mode requires a few additional connections over the asynchronous mode. The FPGA CCLK must be connected to the flash CLK signal and the FPGA ADV_B must be tied to the flash $\overline{\text{ADV}}$ signal. This setup shown in Figure 2-19 supports the FPGA powering up initially in asynchronous mode and then switching over to synchronous mode.
2. 7 series FPGA $V_{CCO\_}0$ supply input and Xilinx Cable V_{REF} must be tied to the same voltage.
3. 7 series FPGA bank voltage V_{CCO_14} supplies: A[15:00], FCS_B, D[15:00], EMCCLK, PUDC_B, and CSO_B signals. Bank voltage V_{CCO_15} supplies: A[28:16], FWE_B, FOE_B, ADV_B, RS0, and RS1 signals.
4. M[2:0] = `010` for BPI mode.
5. Only the x16 data width shown in Figure 2-19 is supported by the synchronous read mode.
6. Sending a bitstream to the data pin follows the same bit-swapping rule as in SelectMAP mode. See Parallel Bus Bit Order, page 79.

7. The RS[1:0] pins are not connected as shown in Figure 2-19. These output pins are optional and can be used for MultiBoot configuration. See Chapter 7, Reconfiguration and MultiBoot.
8. The DONE pin is an open-drain output. See Table 2-4, page 24 for DONE signal details.
9. The JTAG connections are shown for a simple, single-device JTAG scan chain. When multiple devices are on the JTAG scan chain, the proper IEEE Std 1149.1 daisy-chain technique should be used 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.
10. The FPGA mode (M[2:0]) pins are shown set to Master BPI 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 configuration during debug.
11. The FPGA PUDC_B pin is tied to ground in this sample schematic enabling non-dedicated configuration I/O's internal pull-ups during configuration. PUDC_B can alternatively be tied High setting the non-dedicated configuration I/Os to 3-state during configuration.
12. The 7 series FPGA supports AES decryption in the 16-bit wide configuration mode but is not used in this setup. Thus the V_{CCBATT} decryptor key battery power supply is tied to GND.
13. See the respective 7 series FPGAs data sheet for the V_{CCINT} supply voltage.

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 **`ConfigRate`** option. The **`ConfigRate`** option sets the nominal configuration clock frequency. The default ConfigRate setting is recommended for the BPI asynchronous read mode. This default value sets the nominal master CCLK frequency to 3 MHz, which satisfies timing requirements for the leading BPI flash families. The 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 for asynchronous read mode, these timing parameters must be considered:

- FPGA nominal master CCLK frequency (ConfigRate)
- FPGA master CCLK frequency tolerance ($F_{MCCKTOL}$)
- ADDR[28:0] 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 $T_{MCCKTOL}$. Due to the master configuration clock tolerance ($T_{MCCKTOL}$), the **`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 2-2.

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

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 awaken 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 respective 7 series FPGAs data sheet for 7 series FPGA power supply requirements and timing. 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.

JTAG Interface

From the four-pin JTAG interface, the 7 series FPGA can be configured using Xilinx tools 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`). Xilinx tools use the JTAG interface, including ISE and the ChipScope™ Pro tool in the ISE Design Suite, and in the Vivado Design Suite lab tools.

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.

XILINX®

Chapter 3

Boundary-Scan and JTAG Configuration

Introduction

Xilinx® 7 series devices support IEEE standard 1149.1, defining a Test Access Port (TAP) and boundary-scan architecture. 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. Test Access Port and boundary-scan architecture is commonly referred to collectively as JTAG.

Boundary-Scan for 7 Series Devices Using IEEE Standard 1149.1

The 7 series 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 7 series family also supports a 32-bit Device Identification register and a Configuration register. This section outlines the details of the JTAG architecture for 7 series devices.

Test Access Port (TAP)

The 7 series FPGA TAP contains four mandatory dedicated pins as specified by the protocol in 7 series devices and in typical JTAG architecture. 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). For more details, see Chapter 10, Advanced JTAG Usage.

The four mandatory TAP pins are outlined in Table 3-1. These pins are located in configuration bank 0. For 2.5V or 3.3V operation, set V_{CCO_0} to 2.5V or 3.3V and connect CFGBVS to V_{CCO_0} (see Configuration Banks Voltage Select).

Table 3-1: **7 Series FPGA TAP Controller Pins**(1)

Pin	Direction	Pre-Configuration Internal Pull Resistor	Description
TDI	In	Pull-up	**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	Pull-up	**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. TDO has an internal resistive pull-up to provide a logic High if the pin is not active.
TMS	In	Pull-up	**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	Pull-up	**Test Clock**. This pin is the JTAG Test Clock. TCK sequences the TAP controller and the JTAG registers. TCK has an internal resistive pull-up to provide a logic High if the pin is not driven.

Notes:

1. TMS and TDI have internal pull-up resistors, as specified by IEEE Std 1149.1, as do TDO and TCK. These internal pull-up resistors are active, regardless of the mode selected. Refer to the respective 7 series FPGAs data sheet for internal pull-up values. Bitstream options 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 respective 7 series FPGAs data sheet in the Configuration Switching Characteristics table.

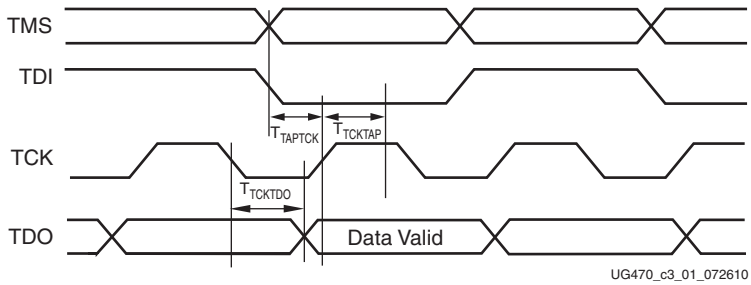


Figure 3-1: **7 Series FPGA Boundary-Scan Port Timing Waveforms**

Using Boundary-Scan in 7 Series Devices

For single-device configuration, the TAP controller commands are issued automatically if the part is being configured with Xilinx tools. 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 7 series FPGA. The tools automatically check for proper connections and drive 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 lab tools. 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. Proper physical connections of all of these signals are essential to JTAG functionality.

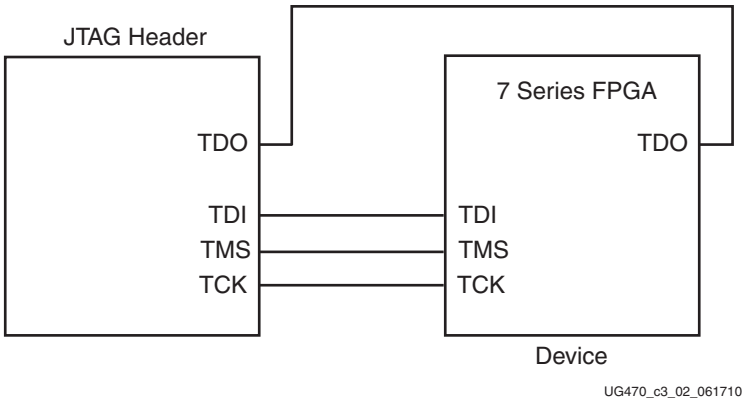


Figure 3-2: **Single Device JTAG Programming Connections**

Multiple Device Configuration

It is possible to configure multiple 7 series devices in a chain. (See Figure 3-3.)

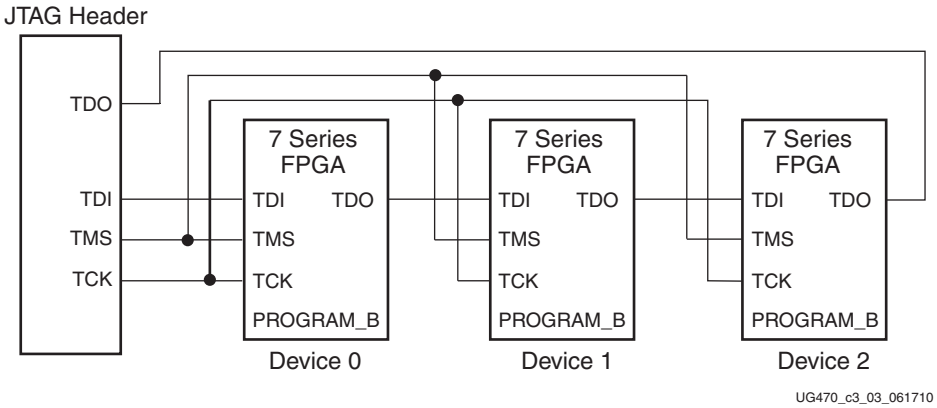


Figure 3-3: **Boundary-Scan Chain of Devices**

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 Xilinx tools automatically discover the devices in the chain, starting from the one nearest to TDI coming from the JTAG header.

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 carry out boundary-scan testing, SelectIO and transceiver pins need to be set up. For SelectIO pins, the SAMPLE instruction will force the input buffer to LVCMOS. If the input level is floating (without a pull-up), then additional current draw can be seen. For transceiver pins, the SAMPLE instruction will turn on the band-gap and pad driver to allow for AC-JTAG (IEEE 1149.6) operation. This will increase power consumption for an unconfigured device as the transceiver will be in power down mode. The current consumed will not be higher than normal operation.

Configuring through Boundary-Scan

The 7 series devices support configuration through the standard boundary-scan (JTAG) port. The devices support configuration through the JTAG port at any time, regardless of the configuration mode pin settings. However, an explicit JTAG configuration mode setting is available when the devices are to be exclusively configured through the JTAG port.

Xilinx has proprietary programming cables (parallel and USB) and boundary-scan programming tools 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. Standalone BSDL files are provided in the ISE tools and on the download center on xilinx.com. The Vivado tools have integrated BSDL information.

One of the most common boundary-scan vendor-specific instructions is the configure instruction. If the 7 series device is configured via JTAG, the configure instructions occur independent from the mode pins.

JTAG configuration for devices based on Stacked Silicon Interconnect (SSI) technology is supported only via iMPACT or the Vivado device programmer using either a JTAG cable connection or the serial vector format (SVF) file.

XILINX®

Chapter 4

Dynamic Reconfiguration Port (DRP)

Dynamic Reconfiguration of Functional Blocks

Background

In the 7 series 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 reconfiguration using the JTAG, ICAPE2, or SelectMAP ports. However, the dynamic reconfiguration port that is an integral part of many functional block simplifies this process greatly. Such configuration ports exist in CMTs, MMCMs/PLLs, XADC, serial transceivers, and the PCIe® block (not I/Os).

Overview

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

- It is directly accessible from the FPGA logic. 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 ICAPE2.
- 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 4-1 shows how the configuration bits drive the logic in functional blocks directly in earlier FPGA families, and Figure 4-2 shows how the reconfiguration logic changes the flow to read or write the configuration bits.

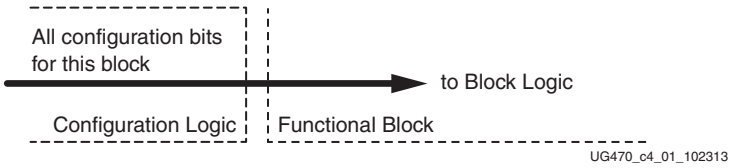


Figure 4-1: **Block Configuration Logic without Dynamic Interface**

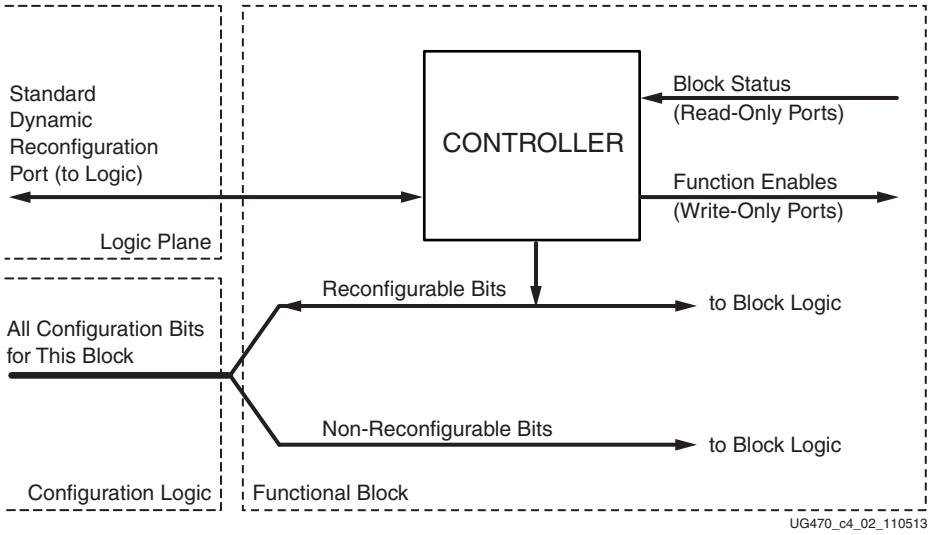


Figure 4-2: **Block Configuration Logic with Dynamic Interface**

Figure 4-3 is the same as Figure 4-2, except the port between the Logic Plane and Functional Block is expanded to show the actual signal names and directions.

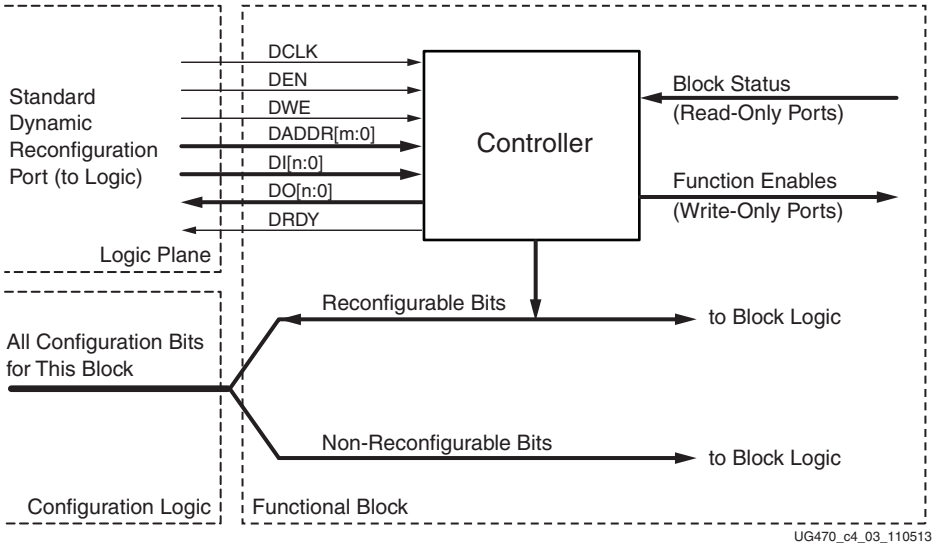


Figure 4-3: **Block Configuration Logic Expanded to Show Signal Names**

FPGA Logic Port Definition

Table 4-1, page 73, lists each signal on the FPGA logic port. The individual functional blocks can implement all or only a subset of these signals, and may use different names. 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 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.

Absolute timing parameters, such as maximum DCLK frequency, are defined in the respective 7 series FPGAs data sheet.

Table 4-1: **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. 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. DEN should only be pulsed for one DCLK cycle.
DWE	Input	When active, this signal enables a write operation to the port (see DEN).
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.

For more details on functional blocks with Dynamic Reconfiguration Ports, see the following documentation.

* XAPP888, *MMCM and PLL Dynamic Reconfiguration*
* UG480, *7 Series FPGAs XADC User Guide*
* UG476, *7 Series FPGAs GTX/GTH Transceiver User Guide*
* UG482, *7 Series FPGAs GTP Transceiver User Guide*
* PG054, *7 Series FPGAs Integrated Block for PCI Express Product Guide*

Configuration Details

Configuration Data File Formats

Xilinx design tools can generate configuration data files in a number of different formats, as described in Table 5-1. In the ISE tools, 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. The equivalent Vivado Tcl commands are WRITE_BITSTREAM and WRITE_CFGMEM. PROM files can be generated in a number of different file formats and do not need to be used with a PROM. They can be stored anywhere and delivered by any means.

Table 5-1: **Xilinx Configuration File Formats**

File Extension	Bit Swapping(1)	Xilinx Tool(2)	Description
BIT	Not Bit Swapped	ISE BitGen or Vivado write_bitstream (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 the iMPACT tool or Vivado device programmer with a programming cable.
RBT	Not Bit Swapped	ISE BitGen generated if **-b** option is set) or Vivado write_bitstream (generated with -raw_bitfile argument)	ASCII equivalent of the BIT file containing a text header and ASCII `1`s and `0`s. (Eight bits per configuration bit.)
BIN	Not Bit Swapped	ISE BitGen (generated if **-g binary:yes** option is set) or PROMGen, or Vivado write_bitstream (generated with -bin_file argument)	Binary configuration data file with no header information. Can be used for custom configuration solutions (for example, microprocessors), or in some cases to program third-party PROMs.
MCS	Bit Swapped(3)	ISE PROMGen or iMPACT, or Vivado write_cfgmem -format MCS	ASCII PROM file format containing address and checksum information in addition to configuration data. Used mainly for device programmers and the iMPACT tool.

Table 5-1: **Xilinx Configuration File Formats *(Cont'd)***

File Extension	Bit Swapping[1]	Xilinx Tool[2]	Description
HEX	Determined by User	ISE PROMGen or iMPACT, or Vivado write_cfgmem -format HEX	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, see UG628, *Command Line Tools User Guide*. For equivalent tools and syntax in the Vivado Design Suite, see UG908, *Vivado Design Suite Programming and Debugging 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.

The 7 series FPGA bitstream contains commands to the FPGA configuration logic as well as configuration data.

A 7 series 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 5-2 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 5-2 shows the 32-bit configuration word for an unswapped bitstream. For swapped and unswapped formats, see Configuration Data File Formats.

Table 5-2: **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
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 tools 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 5-3 shows the Sync word in an unswapped bitstream format.

Table 5-3: **Sync Word**

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

Generating Memory Files

PROM files are generated from bitstream files with the PROMGen utility in the ISE Design Suite, or with the Tcl command WRITE_CFGMEM in the Vivado Design Suite. Users can access PROMGen directly from the command line or indirectly through the iMPACT File Generation Mode. For PROMGen syntax, see UG628, *Command Line Tools User Guide*. For information on iMPACT, refer to the ISE software documentation. For information on WRITE_CFGMEM, see UG908, *Vivado Programming and Debugging User Guide*. 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. For the WRITE_CFGMEM Tcl command, use the argument **`-loadbit "up|down <address1> <bitfile1.bit> <address2> <bitfile2.bit>"`**. 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 XAPP583, *Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave 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.

PROM Files for SPI/BPI Configuration

The **-d** and **-u** options in PROMGen, the iMPACT File Generation Wizard, or the write_cfgmem **-loadbit** argument 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 PROM file format is always bit swapped unless the PROMGen **-spi option** or write_cfgmem **-interface spi1** | **spi2** | **spi4** 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 5-1 shows how two bytes of data (0xABCD) are bit swapped.

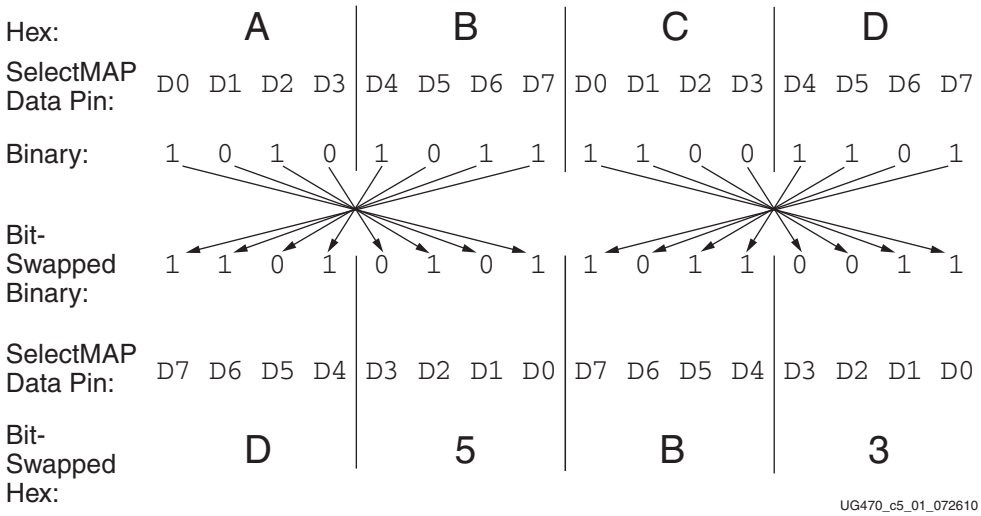


Figure 5-1: **Bit Swapping Example**

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, and for the ICAPE2 interface.

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 7 series FPGA BPI x8 modes and to the ICAPE2 interface (see Bit Swapping).

In 7 series 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 7 series FPGAs is the same as in Virtex®-6 FPGAs.

Table 5-4 and Table 5-5 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, and BPI modes, and when using the ICAPE2 interface.

Table 5-4: **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 5-5: **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			

Delaying Configuration

To delay configuration, the INIT_B pin should be held Low during initialization (Figure 5-4). When INIT_B has gone High, configuration cannot be delayed by subsequently pulling INIT_B Low.

The signals relating to initialization and delaying configuration are defined in Table 5-6.

Table 5-6: **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, initialization is complete.
INIT_COMPLETE	Status[2]	Internal signal, accessible through the 7 series FPGA status register	Indicates whether INIT_B signal is internally released.
MODE_STATUS[2:0]	Status	Internal signals, accessible through the 7 series FPGA status register	Reflects the values sampled on the Mode pins when the status is read.

Notes:

1. Information on the 7 series FPGA status register is available in Table 5-29, page 110. Information on accessing the device status register via SelectMAP is available in Chapter 6, Readback and Configuration Verification.
2. The Status type is an internal status signal without a corresponding pin.

Configuration Sequence

While each of the configuration interfaces is different, the basic steps for configuring a 7 series device are the same for all modes. Figure 5-2 shows the 7 series 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.

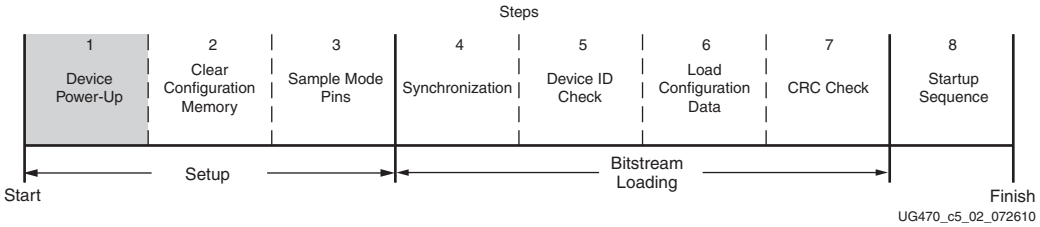


Figure 5-2: **7 Series FPGA Configuration Process**

The 7 series 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 5-3).

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)

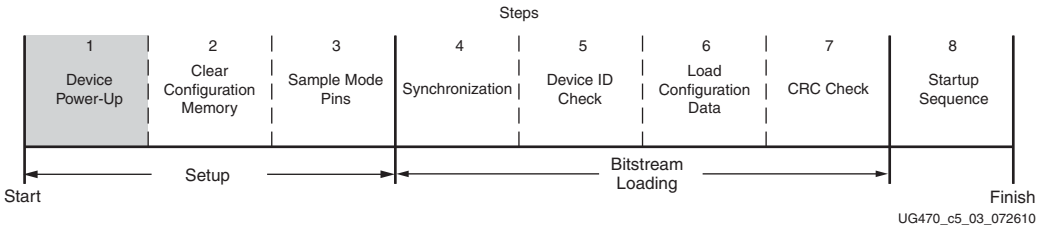


Figure 5-3: **Device Power-Up (Step 1)**

For configuration, 7 series devices require power on the V_{CCO_0}, V_{CCAUX}, V_{CCBRAM}, and V_{CCINT} pins. Power sequencing requirements are described in the respective 7 series FPGAs data sheet.

All JTAG configuration pins are located in a separate, dedicated bank with a dedicated voltage supply (V_{CCO_0}). The multi-function pins are located in Banks 14 and 15. All dedicated input pins operate at the V_{CCO_0} LVCMOS level. All active dedicated output pins operate at the V_{CCO_0} voltage level with the output standard set to LVCMOS, 12 mA drive, Fast slew rate. If the Persist option is used, the dual-mode I/O for the selected configuration mode remains active after configuration, with the I/O standard set to LVCMOS, 12 mA drive, Slow slew rate.

For all modes that use multi-function I/O, the associated V_{CCO_14} or V_{CCO_15} must be connected to the appropriate voltage to match the I/O standard of the configuration device. The pins are also LVCMOS, 12 mA drive, Fast slew rate during configuration.

For power-up, the V_{CCINT} power pins must be supplied with 1.0V or 0.9V (for -2L) sources. None of the I/O voltage supplies except V_{CCO_0} needs to be powered for 7 series FPGA configuration in JTAG mode. When configuration modes are selected that use the multi-function pins (i.e., Serial, Master BPI, SPI, SelectMAP), V_{CCO_14}, V_{CCO_15}, or both must be also be supplied. Table 5-7 shows the power supplies required for configuration. Table 5-8 shows the timing for power-up. Refer to the respective 7 series FPGAs data sheet for voltage ratings.

Table 5-7: **Power Supplies Required for Configuration**

Pin Name	Description
V_{CCINT}	Power supply for the internal core logic.
V_{CCBATT}(1)	AES decryptor key memory backup power supply; If the key memory is not used, the user should tie this pin to V_{CCAUX} or GND.
V_{CCAUX}	1.8V power supply for auxiliary circuits.
$V_{CCAUX\_IO\_\#}$	1.8V/2.0V power-supply pins for auxiliary I/O circuits.
V_{CCBRAM}	Power-supply pins for the FPGA's logic block RAM.
V_{CCO_0}	Configuration bank supply voltage.
V_{CCO_14} V_{CCO_15}	Multi-function configuration pin output supply voltage. Standard I/O voltage levels supported for configuration are 1.5V, 1.8V, 2.5V, and 3.3V.

Notes:

1. V_{CCBATT} is required only when an AES key is stored in the FPGA's battery-backed RAM for decryption of an encrypted bitstream.

Table 5-8: **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 the respective 7 series FPGAs data sheet for power-up timing characteristics.

Figure 5-4 shows the power-up waveforms.

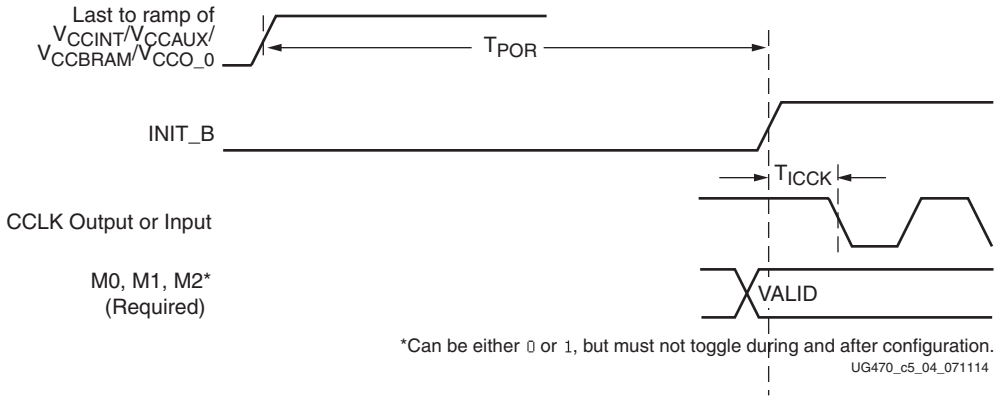


Figure 5-4: **Device Power-Up Timing**

To ensure proper power-on behavior, the guidelines in the respective 7 series FPGAs data sheet must be followed. The power supplies should ramp monotonically within the power supply ramp time range specified in the respective 7 series FPGAs 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 respective 7 series FPGAs data sheet for specific values) can result in loss of configuration data.

If a monotonic ramp is not possible, delay configuration by holding the INIT_B Low (see Delaying Configuration) while the system power reaches the minimum recommended operating voltages for V_{CCO_0}, V_{CCAUX}, V_{CCBRAM}, and V_{CCINT}. A few configuration modes involve bank 14 or bank 15, or both. When these banks are involved in configuration, their respective voltage supplies, V_{CCO_14} and/or V_{CCO_15}, must also reach their minimum recommended operating voltages prior to the release of INIT_B to High.

After power-up, the device can be re-configured by toggling the PROGRAM_B pin Low (see Figure 5-5).

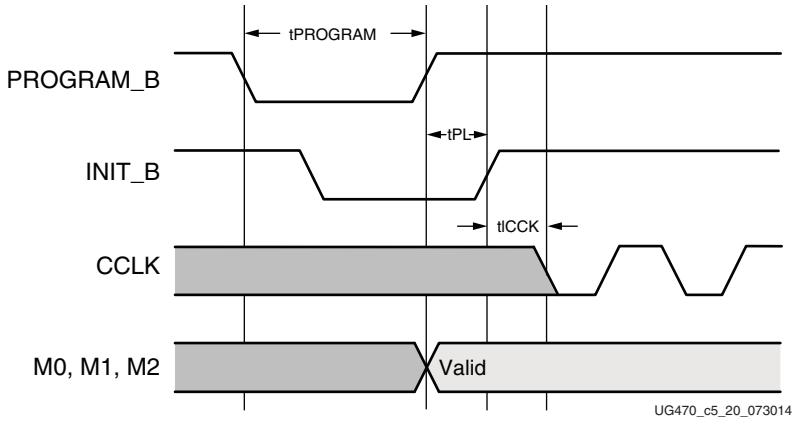


Figure 5-5: **Re-configuring the Device by Toggling the PROGRAM_B Pin Low**

Clear Configuration Memory (Step 2, Initialization)

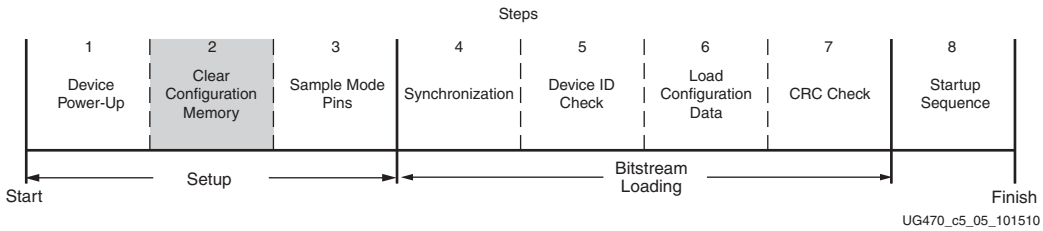


Figure 5-6: **Initialization (Step 2)**

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. Block RAM is reset to its initial state, and flip-flops are re-initialized through the assertion of the global set reset (GSR). During this time, I/Os are placed in a High-Z state except for the Configuration and JTAG pins, through the use of the global three-state (GTS). INIT_B is internally driven Low during initialization, then released after T_{POR} (Figure 5-4) 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, and the T_{POR} or T_{PL} delay is met.

The minimum Low pulse time for PROGRAM_B is defined by the $T_{PROGRAM}$ timing parameter.

Sample Mode Pins (Step 3)

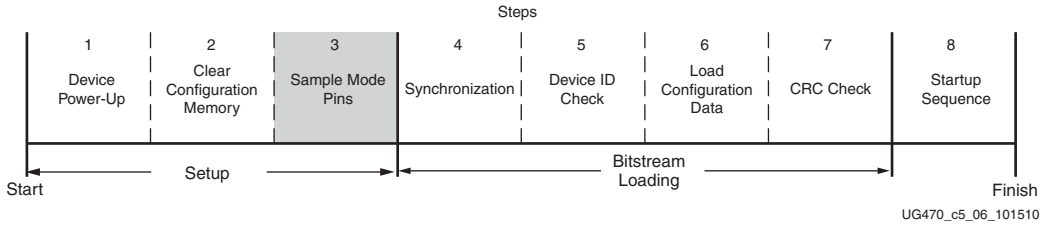


Figure 5-7: **Sample Mode Pins (Step 3)**

When the INIT_B pin transitions to High, the device samples the M[2:0] mode pins and begins driving CCLK if in the Master modes. At this point, the device begins sampling the configuration data input pins on the rising edge of the configuration clock. For BPI and SelectMAP modes, the bus width is initially x8, and the Status register reflects this. After the bus width detection sequence, the Status register is updated. The mode pins are sampled again only upon reconfiguration through a power cycle or assertion of PROGRAM_B.

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)

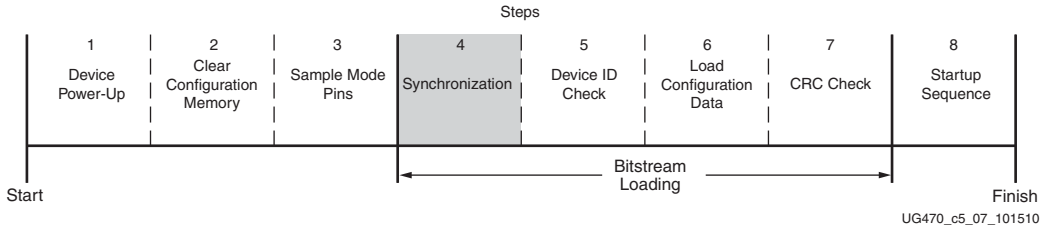


Figure 5-8: **Synchronization (Step 4)**

For BPI, 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 tools include both the bus width detection pattern and the synchronization word. Table 5-9 shows signals relating to synchronization.

Table 5-9: **Signals Relating to Synchronization**

Signal Name	Type	Access	Description
DALIGN	Status	Only available through the SelectMAP interface during an ABORT sequence.	Indicates whether the device is synchronized.
IWIDTH	Status	Internal signal. Accessed only through the 7 series 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 If ICAPE2 is enabled, this signal reflects the ICAPE2 width after configuration is done.

Notes:

1. Information on the 7 series FPGA status register is available in Table 5-29. Information on accessing the device status register via JTAG or SelectMAP is available in Chapter 6, Readback and Configuration Verification.

Check Device ID (Step 5)

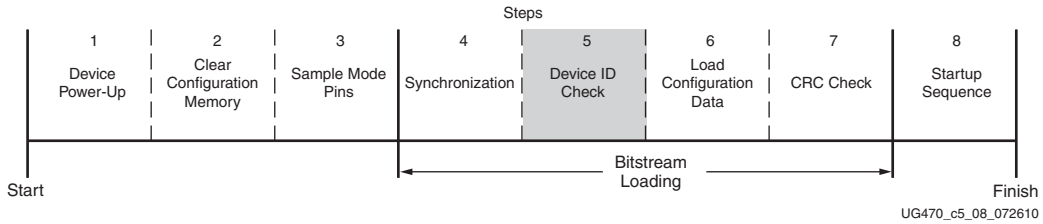


Figure 5-9: **Check Device ID (Step 5)**

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.

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. 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 7 series FPGA JTAG ID Code register has this 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

Load Configuration Data Frames (Step 6)

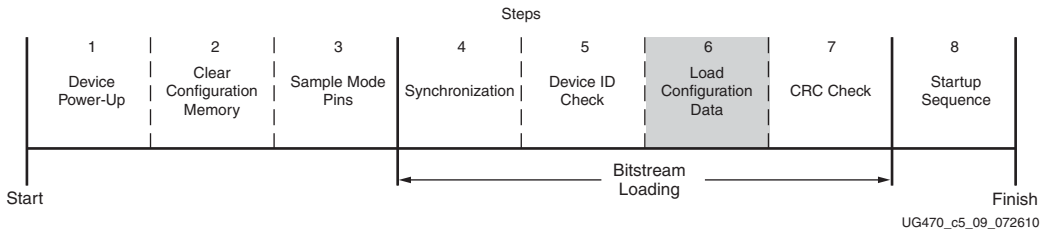


Figure 5-10: **Load Configuration Data Frames (Step 6)**

After the synchronization word is loaded and the device ID has been checked, the configuration data frames are loaded (see Configuration Memory Frames, page 103). This process is transparent to most users.

Cyclic Redundancy Check (Step 7)

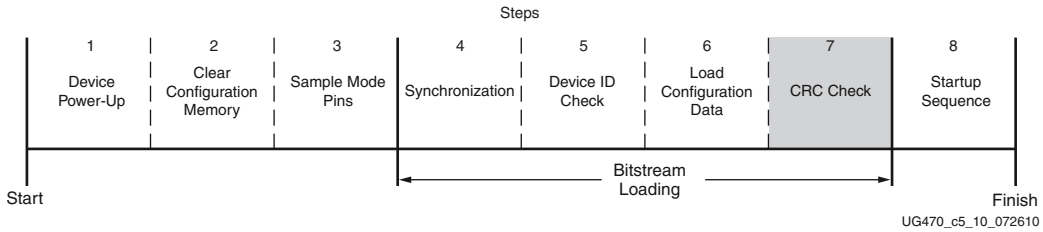


Figure 5-11: **Cyclic Redundancy Check (Step 7)**

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 (see 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 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 Mode in Chapter 2).

7 series 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. After 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 Mode asynchronous read, the address counter eventually overflows or underflows to cause

wraparound, which triggers fallback reconfiguration. BPI synchronous read mode does not support the wraparound error condition.

Startup (Step 8)

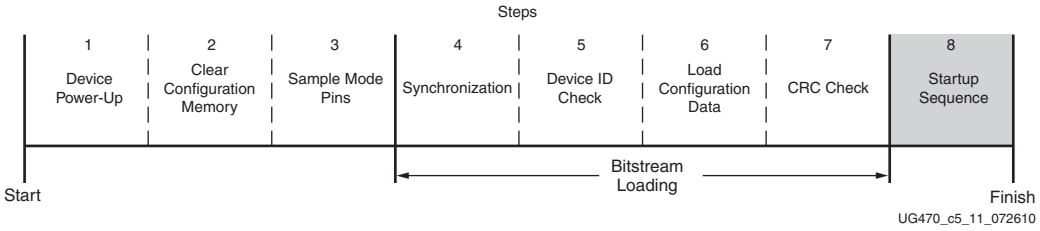


Figure 5-12: **Startup Sequence (Step 8)**

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 5-10.

Table 5-10: **User-Selectable Cycle of Startup Events**

Phase	Event
1–6	Wait for MMCMs to Lock (optional)
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 bitstream options (see UG628, *Command Line Tools User Guide*). Table 5-10 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 5-11.

Table 5-11: **Default Sequence of Startup Events**

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

The startup sequence can be forced to wait for the MMCMs to lock or for DCI to match with the appropriate options. These options are typically set to prevent DONE, GTS, and GWE from being asserted (preventing device operation) before the MMCMs have locked and/or 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 releasing the DONE pin, the

device stops driving a logic Low, and the pin is pulled up by an internal pull-up resistor. DONE_PIPE is enabled by default to add a register between the DONE pin and the configuration logic. See Table 2-4 for DONE signal changes in the 7 series FPGAs. Table 5-12 shows signals relating to the startup sequencer. Figure 5-13 shows the waveforms relating to the startup sequencer.

Table 5-12: **Signals Relating to Startup Sequencer**

Signal Name	Type	Access[1]	Description
DONE	Bidirectional[2]	DONE pin or 7 series FPGA Status Register	Indicates configuration is complete. Can be held Low externally to synchronize startup with other FPGAs.
Release_DONE	Status	7 series 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 management blocks are ready. This signal is asserted by default. It is active if the LOCK_WAIT option is used on an MMCM and the LockCycle option is used when the bitstream is generated.

Notes:

1. Information on the 7 series FPGA status register is available in Table 5-29. Information on accessing the device status register via JTAG or SelectMAP is available in Chapter 6, Readback and Configuration Verification.
2. Open-drain output.
3. GWE is asserted synchronously to the configuration clock (CCLK) and has a significant skew across the part. Therefore, sequential elements are not 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.

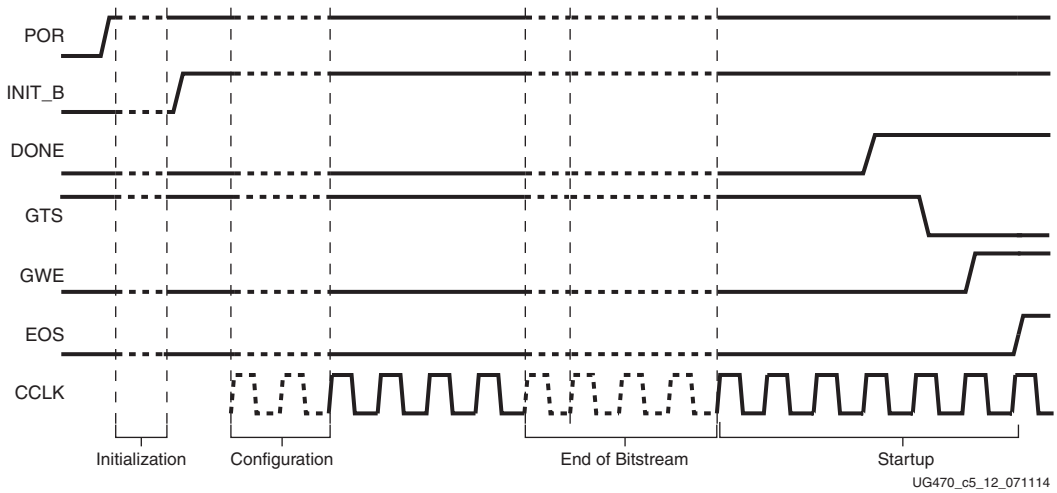


Figure 5-13: **Configuration Signal Sequencing (Default Startup Settings)**

Clocking to End of Startup

By default, DONE is released in phase 4 of startup, and DONE_PIPE is enabled to add one additional clock cycle of latency. DONE indicates that configuration is complete and all data has been loaded, but some extra clock cycles need to be applied to ensure the startup sequence completes correctly all the way to phase 7, End of Startup. A conservative number for the clock cycles required after DONE is 24; this will account for the most common use cases. The bitstream options LCK_cycle or Match_cycle will add an undefined additional number of clock cycles.

I/O Transition at the End of Startup

In the Artix-7 and Kintex-7 families, which have the multi-function configuration pins on HR I/O banks, if the V_{CCO} for the bank is 1.8V or lower, and if a pin on that bank is Low or floating, then the input might have a `0-1-0` transition to the interconnect logic during configuration startup. Because this transition occurs after GWE enables the internal logic, it might affect the internal state of the device after configuration. The transition occurs one CFGCLK after EOS (End Of Startup). To avoid this transition, set V_{CCO_14} and V_{CCO_15} to 2.5V or 3.3V, or drive the pin High externally (see Table 5-13). Otherwise, logic should be designed to ignore these affected input signals until at least 200 ns after one CFGCLK following the rising edge of EOS. CFGCLK and EOS can be monitored using the STARTUPE2 primitive.

Table 5-13: **I/O Transition at End of Startup in Artix-7 and Kintex-7 Devices**

V_{CCO_0}	V_{CCO_14} or V_{CCO_15}	Pin State	Input Transition
2.5V or 3.3V	1.8V or lower	`0` or floating	`0-1-0`
1.8V or lower	Any	Any	None
Any	2.5V or 3.3V	Any	None
Any	Any	`1`	None

Using DCI with the Multi-function Configuration Pins

If any of the configuration pins in I/O banks 14 or 15 are assigned DCI I/O standards in the user design, the DCIRESET primitive should also be included and used in the design. The design should pulse the RST input of DCIRESET and then wait for the LOCKED signal to be asserted prior to using any user input or outputs on the multi-function configuration pins with DCI standards. While the multi-function configuration I/O pins are performing as configuration pins, they ignore the initial DCI calibration that happens during device initialization. For more details, see UG471, *7 Series FPGAs SelectIO Resources User Guide*.

STARTUPE2 Primitive

The STARTUPE2 primitive (see Figure 5-14) provides an interface between the user logic and the configuration logic control and status signals. Many of the pins are related to the startup sequence, including the CLK signal to allow user specification of the startup clock. STARTUPE2 can be instantiated in a design to provide user control over selected configuration signals during device operation.

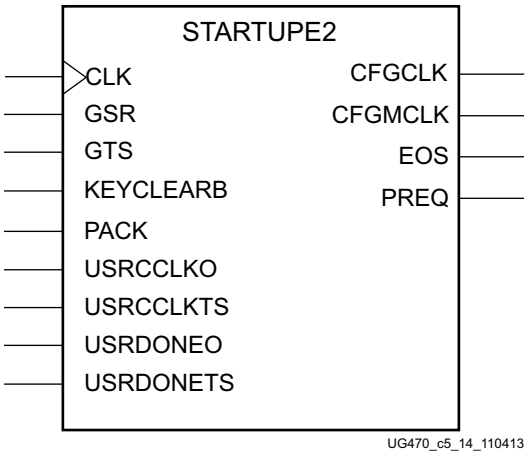


Figure 5-14: **STARTUPE2 Primitive**

Table 5-14 describes the pins on the STARTUPE2 primitive.

Table 5-14: **STARTUPE2 Pin Table**

Pin Name	Type	Description
CLK	Input	User startup clock pin. Input from the FPGA logic driving the device startup sequence clock. Provides a user-defined CCLK for the startup sequence.
GSR	Input	Not recommended – should be tied Low to disable. The GSR (Global Set/Reset) pin is an active High input from the FPGA logic that asserts an asynchronous set/reset that can be used to re-initialize CLB flip-flops. The GSR signal spans the entire device and is released asynchronous to the user clocks. Due to the asynchronous release and skew across the device it is likely that flip-flops are not released in the same clock cycle, and it is possible to have a metastable event. Applications that use GSR should either stop all clocks before GSR and/or reconfigure the device after GSR. The same flip-flop initialization (set/reset) is performed safely during device configuration prior to startup.
GTS	Input	Global three-state pin. Active-High input from the FPGA logic which puts all user I/Os except for configuration banks into a high-Z state. Same function is always asserted during configuration. For most applications, this pin should be tied Low.
KEYCLEARB	Input	Clear AES Decrypter Key from battery-backed RAM. Active-Low input from the FPGA fabric. This pin, when held Low for the $T_{PROGRAM}$ delay in the data sheet, erases the contents of the decryption keys from the battery-backed RAM (BBRAM).
PACK	Input	PROGRAM_B or JPROGRAM acknowledge. An input from the FPGA logic which "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.
PREQ	Output	PROGRAM_B pulse or JPROGRAM REQuest to FPGA logic. An output into the FPGA logic. This pin is the "request" from the PROGRAM_B state machine to reset the device, allowing 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.
USRCCLKO	Input	CCLK pin. An input from the FPGA logic driving a custom, logic-generated clock frequency onto the FPGA CCLK pin after configuration. Useful for post-configuration access of external SPI flash devices. See USRCCLKO for additional details.

Table 5-14: **STARTUPE2 Pin Table *(Cont'd)***

Pin Name	Type	Description
USRCCLKTS	Input	User CCLK three-state enable to CCLK pin. Active-High input from the FPGA logic which puts the FPGA CCLK pin into a high-Z state when used after configuration. For most applications, this pin should be tied Low.
USRDONEO	Input	DONE pin output value. An input from the FPGA logic which connects to the FPGA DONE pin.
USRDONETS	Input	User DONE three-state enable for DONE pin. Active-High input from the FPGA logic which puts DONE into a high-Z state. Typically tied Low to enable DONE.
CFGCLK	Output	Configuration logic main clock output. An output into the FPGA logic. Outputs a clock signal from the dedicated internal ring oscillator where the typical frequency is defined by the bitstream ConfigRate option. The output is active only during configuration, and in master modes with Persist enabled.
CFGMCLK	Output	Configuration internal oscillator clock output. An output into the FPGA logic. Outputs a clock signal from the dedicated internal ring oscillator with a typical 65 MHz frequency.
EOS	Output	End of Startup. Active-High output echoing the EOS flag into the FPGA logic. Can be used as a reset signal indicating the FPGA is ready for operation.

USRCCLKO

The USRCCLKO input on the STARTUPE2 primitive allows the user logic to drive the CCLK pin after configuration. USRCCLKO can also clock the POST_CRC readback logic when both are used (see Table 8-1). The delay from the internal USRCCLKO to the CCLK pin is defined as TUSRCCLKO in the data sheet. The first three clock cycles on USRCCLKO after End of Startup are used to switch the clock source and will not be output on the external CCLK pin. This helps prevent CCLK glitches in the transition from the internal oscillator to the user clock. However, if the External Master CCLK pin EMCCLK is used for configuration, it will continue to be seen on CCLK until the three clock cycles of USRCCLKO allow the transition to a new user clock.

Bitstream Security

This section discusses the available types of FPGA bitstream security including: bitstream encryption and bitstream authentication.

A basic form of security is to prevent readback. The bitstream Security setting can be set to Level1 (disables readback), or Level2 (disables both readback and reconfiguration).

Bitstream Encryption

7 series devices have on-chip Advanced Encryption Standard (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 7 series FPGA designs cannot be copied or reverse-engineered.

The 7 series 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 tools, the user generates the encryption key and the encrypted bitstream. 7 series 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 7 series device performs the reverse operation, decrypting the incoming bitstream. The 7 series 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 7 series FPGA encryption system uses the Advanced Encryption Standard (AES) encryption algorithm. AES is an official standard supported by the National Institute of Standards and Technology (NIST) and the U.S. Department of Commerce (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf).

The 7 series 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 selects the Encrypt option and specifies a 256-bit key as an input to BitGen using the Keyfile option. BitGen in turn generates an encrypted bitstream file (BIT) and an encryption key file (NKY).

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

The WRITE_BITSTREAM Tcl command option provides the same functionality in the Vivado Development System (see UG908, *Vivado Design Suite Programming and Debugging User Guide*). For a simple step-by-step process, see XAPP1239, *Using Encryption to Secure a 7 Series FPGA Bitstream Application Note*.

Loading the Encryption Key

The encryption key can only be loaded onto a 7 series device through the JTAG interface. The iMPACT tool or Vivado device programmer can accept the NKY file as an input and program the device with the key through JTAG, using a supported Xilinx programming cable.

To program the key, the device enters a special *key-access mode* using the XSC_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 in the FUSE_CNTL register to disable reading and writing of the AES key. For more details, see eFUSE, page 97.

Loading Encrypted Bitstreams

After 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 bitstream security setting.

While the device holds an encryption key, a non-encrypted bitstream can be used to configure the device only after POR or PROGRAM_B 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 7 series FPGA encryption scheme.

Most methods of configuration are not affected by encryption. 7 Series FPGAs allow for bitstreams to be created with both compression and encryption. An encrypted bitstream can be delivered through any configuration interface: JTAG, serial, SPI (including x1, x2, and x4 modes), BPI, SelectMAP, and ICAPE2. However, an encrypted bitstream has a few limitations or timing differences for some of the configuration methods. The Slave SelectMAP and ICAPE2 interfaces accept encrypted bitstreams only through the x8 bus. The Master SelectMAP and Master BPI interfaces accept encrypted bitstreams through either the x8 or x16 data bus, but for the x16 bus width, the master CCLK frequency is slowed to half of the **`ConfigRate`**, or half of the EMCCLK rate when ExtMasterCCLK_en is used. The slower CCLK begins early in the bitstream when the DEC (AES encryptor enable) bit is read, before the CCLK is updated based on the ConfigRate frequency or the external EMCCLK frequency.

The encrypted bitstream must configure the entire device because partial reconfiguration through the external configuration interfaces is not permitted for encrypted bitstreams. 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 7 series FPGAs even when encryption is turned on. Readback is available through the ICAPE2 primitive (see Bitstream Encryption and Internal Configuration Access Port (ICAPE2)). None of these events resets the key if V_{CCBATT} 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.

Bitstream Encryption and Internal Configuration Access Port (ICAPE2)

The Internal Configuration Access Port (ICAPE2) primitive provides the user logic with access to the 7 series FPGA configuration interface. For details on the ICAPE2 primitive, see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*. The ICAPE2 interface is similar to the SelectMAP interface, including bit-swapping (see Parallel Bus Bit Order, page 79). The ICAPE2 has independent input and output buses – the CSIB input ignores the input bus but the output bus can continue to toggle. Timing is similar to the SelectMAP interface, and the maximum frequency F_{ICAPCK} is included in the 7 series data sheets. The restrictions on readback for the SelectMAP interface do not apply to the ICAPE2 interface after configuration. Users can send a partial bitstream, whether encrypted or unencrypted, or can perform readback through the ICAPE2 interface, even if bitstream encryption is used. Unless the designer wires the ICAPE2 interface to user I/O, this interface does not offer attackers a method for defeating the 7 series FPGA AES encryption scheme.

Users concerned about the security of their design should *not* wire the ICAPE2 interface to user I/O. Connecting the ICAPE2 clock does not impact security. When the Readback CRC (POST_CRC) and bitstream encryption features are both enabled, the Readback CRC will not be operational unless the ICAPE2 is instantiated and a clock is provided.

Like the other configuration interfaces, the ICAPE2 interface does not provide access to the key register.

For more information on the ICAPE2 primitive, see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*.

Advanced Encryption Standard

The Advanced Encryption Standard (AES) supported in 7 series FPGAs is identical to that supported in Virtex-6 devices. A 256-bit encryption key is loaded into eFUSE bits or battery-backed RAM by the user. The Xilinx bitstream writer, 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 random key. Some security features, such as the ability for the FPGA logic to clear the AES key from battery-backed RAM, require that the part be configured with an encrypted bitstream to function.

Table 5-15: **Bitstream Options**

Name	Type	Settings (Default)	Description
KeyFile	string	<design>.nky	Contains part AES key and part AES initial vector. the bitstream writer creates a randomly generated key and initial vector if a file does not exist.
Encrypt	Boolean	Yes, No (default)	Enabled to encrypt the bitstream. Sets the CTL0[6] (dec) bit in the bitstream.
Key0	string		Allows a key to be specified. Written into the Key file.
StartCBC	string		Allows the initial vector to be specified. Written into the Key file.
EncryptKeySelect	enum	bbram (default), 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.

V_{CCBATT}

When an encryption key is stored in the FPGA's battery-backed RAM, 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_{CCBATT} power input is provided for retaining the key when V_{CCAUX} is removed. Because V_{CCBATT} 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_{CCBATT} DC Characteristics in the respective 7 series FPGAs data sheet and the battery specifications.)

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

Bitstream Authentication

Overview

7 series 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. Any bitstream tampering including single bit flips are detected.

The HMAC algorithm uses a key that is provided to the Xilinx software. Alternately, the software can automatically generate a random key. The HMAC key is separate and different from the AES key. The Xilinx 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 7 series FPGA security solution always consists of both HMAC and AES. Similar functionality is provided by the Vivado lab tools.

Implementation

The 7 series FPGA HMAC authentication system consists of an HMAC component in the Xilinx software and a hardware component integrated into every 7 series FPGA. Both components generate a 256-bit MAC based on a key and the Secure Hash Algorithm (SHA256). Bitstream generation creates 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 7 series 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 Slave SelectMAP in the x16 or x32 bus width. 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.

Creating an Authenticated Bitstream

Because the HMAC must use a key different from the AES key, two keys are specified in the Xilinx tools. 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 **Encrypt:yes** option (see the *BitGen* section of UG628, *ISE Command Line Tools User Guide*, or the *Device Configuration Bitstream Settings* section of UG908, *Vivado Design Suite Programming and Debugging User Guide* for more information). The user can specify an HMAC key in the NKY file or let the software generate a random key automatically.

The NKY file format is:

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

For example:

```
Key HMAC 505daf31dea6930375003b9286bb183752457a90a79ace727b516f0009995a9e;
```

For more information on using the various security protection features, see XAPP1084, *Developing Tamper Resistant Designs with Xilinx Virtex-6 and 7 Series FPGAs.*

eFUSE

eFUSE is a nonvolatile one-time-programmable technology used for selected configuration settings. The fuse link is programmed (or burned or blown) by flowing a large current for a specific amount of time. User-programmable eFUSEs can be programmed with the Xilinx configuration tools. The device should be unconfigured during eFUSE programming.

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

eFUSE Registers

A 7 series FPGA has a total of four eFUSE registers. Table 5-16 lists the eFUSE registers in 7 series devices with their sizes and usage.

Table 5-16: **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 7 series FPGA decryption engine to load encrypted bitstreams. Depending on the read/write access bits in the 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 logic using the EFUSE_USR primitive. See the 7 Series Libraries Guide for a description of the EFUSE_USR primitive. Depending on the read/write access bits in the CNTL register, the code can be programmed and read through the JTAG port.
FUSE_DNA	64	Device identifier programmed by Xilinx [63:0] (bit 0 shifted first)	Unique device identifier bits [63:0], where bits [63:7] correspond to the 57 bits [0:56] read-only XSC_DNA JTAG register and DNA_PORT primitive value known as Device DNA. See Device Identifier and Device DNA, page 118.
FUSE_CNTL	14	Control Bits CNTL [13:0] (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. The FPGA logic can access only the FUSE_USER register value and the 57-bit Device DNA subset of the FUSE_DNA register value. All other eFUSE bits are not accessible from the FPGA logic.

eFUSE Control Register (FUSE_CNTL)

This register contains user programmable bits (Table 5-17). These bits are used to select AES key usage and set the read/write protection for eFUSE registers. Bit 0 is shifted in or out first.

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

Table 5-17: **eFUSE Control Register (FUSE_CNTL)**

Bit Position	Name	Description
0	CFG_AES_Only	Forces use of AES key stored in eFUSE. Prior to programming this bit, the FPGA may configure using an unencrypted bitstream, or a bitstream encrypted with a key value stored in battery-backed RAM. ***Caution!*** If this bit is programmed, the device cannot be used unless the AES key is known. Return material authorization (RMA) returns cannot be accepted if this bit is programmed.
1	AES_Exclusive	Disables partial reconfiguration from external configuration interfaces. However, partial reconfiguration is allowed via the ICAPE2. ***Caution!*** If this bit is programmed, return material authorization (RMA) returns are limited in device analysis and debug. An alternative to preventing readback and configuration is Security Level2.
2	W_EN_B_Key_User	Disables programming of AES key and FUSE_USER.
3	R_EN_B_Key	Disables reading of AES key as well as programming of AES key and user code. This does not disable reading the user code through the EFUSE_USR component, although it disables reading the user code through the JTAG port.
4	R_EN_B_User	Disables reading of user code as well as programming of AES key and user code.
5	W_EN_B_Cntl	Disables programming of control bits.
6 – 13	Reserved	Reserved

When FUSE_CNTL[0] is NOT programmed:

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

Caution! When FUSE_CNTL[0] is programmed, only bitstreams encrypted with the eFUSE key can be used to configure the FPGA through external configuration ports. 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.

External configuration ports are blocked from accessing the configuration memory 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 5-18 lists eFUSE-related JTAG instructions. The JTAG instruction register is 6 bits long, unless it is a device implemented with stacked silicon interconnect technology with super logic regions. See Table 1-1, page 14 and the Stacked Silicon Interconnect, page 18 section for details.

Table 5-18: **eFUSE-Related JTAG Instructions**

JTAG Instruction	Code[1]	Action
FUSE_KEY	110001	Selects the FUSE_KEY register
FUSE_USER	110011	Selects the FUSE_USER register
FUSE_CNTL	110100	Selects the FUSE_CNTL register

Notes:

1. Code is longer for devices based on SSI technology (7V2000T, 7VX1140T, 7VH580T, 7VH870T). See the BSDL files for device-specific information.

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 recognizes a specific data pattern, typically the sync word. For external parallel (BPI or SelectMAP mode) interfaces, the bus width auto detection pattern first sets the configuration interface bus width. See Bus Width Auto Detection, page 76 for details. After the bus width is set and for all other configuration interfaces, all data on the configuration interface is ignored until the synchronization word is recognized. See Sync Word, page 77 for details. After synchronization, the configuration logic processes each 32-bit data word as a configuration packet or component of a multiple word configuration packet. See Configuration Packets, page 104 for details.

Table 5-19 shows the composition of a sample XC7K325T bitstream, generated using default settings.

Table 5-19: **Sample XC7K325T Bitstream**

Configuration Data Word (hex)	Description
FFFFFFFF	Dummy pad word, word 1
FFFFFFFF	Dummy pad word, word 2
…	Dummy pad words 3-7
FFFFFFFF	Dummy pad word, word 8
000000BB	Bus width auto detect, word 1
11220044	Bus width auto detect, word 2
FFFFFFFF	Dummy pad word
FFFFFFFF	Dummy pad word
AA995566	Sync word
20000000	NOOP
30022001	Packet Type 1: Write TIMER register, WORD_COUNT=1
00000000	TIMER[31:0]=00000000 (hex) (Placeholder for optional watchdog timer)

Table 5-19: **Sample XC7K325T Bitstream *(Cont'd)***

Configuration Data Word (hex)	Description
30020001	Packet Type 1: Write WBSTAR register, WORD_COUNT=1
00000000	WBSTAR[31:0]=00000000 (hex) (Placeholder for optional MultiBoot next config address)
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
00000000	CMD[4:0]=00000 (binary) = NULL (Placeholder for optional MultiBoot next config reboot IPROG command)
20000000	NOOP
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
00000007	CMD[4:0]=00111 (binary) = RCRC (Reset CRC register) (Bitstream CRC coverage begins here)
20000000	NOOP
20000000	NOOP
30026001	Packet Type 1: Write CRC register, WORD_COUNT=1
00000000	CRC calculated value=00000000 (hex) (Placeholder for optional PRE_COMPUTED POST_CRC_SOURCE)
30012001	Packet Type 1: Write COR0 register, WORD_COUNT=1
02003FE5	COR0[31:0]=02003FE5 (hex) (Sets configuration options, e.g. EMCCLK, CCLK frequency, startup sequence)
3001C001	Packet Type 1: Write COR1 register, WORD_COUNT=1
00000000	COR1[31:0]=00000000 (hex) (Sets configuration options, e.g., POST_CRC enable, BPI page mode.)
30018001	Packet Type 1: Write IDCODE register, WORD_COUNT=1
03651093	IDCODE[31:0]=03651093 (hex) (If written IDCODE does not match device IDCODE, then error.)
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
00000009	CMD[4:0]=01001 (binary) = SWITCH (change CCLK frequency)
20000000	NOOP
3000C001	Packet Type 1: Write MASK register, WORD_COUNT=1
00000401	Bit mask for write to CTL0/CTL1 register. MASK[31:0]=00000401 (hex)
3000A001	Packet Type 1: Write CTL0 register, WORD_COUNT=1
00000501	CTL0[31:0]=00000501 (hex) (Note: Also check prior MASK.) (Sets configuration control options, e.g. fallback, readback, etc.)
3000C001	Packet Type 1: Write MASK register, WORD_COUNT=1
00000000	Bit mask for write to CTL0/CTL1 register. MASK[31:0]=00000000 (hex)

Table 5-19: **Sample XC7K325T Bitstream *(Cont'd)***

Configuration Data Word (hex)	Description
30030001	Packet Type 1: Write CTL1 register, WORD_COUNT=1
00000000	CTL1[31:0]=00000000 (hex) (Note: Also check prior MASK.) (Sets configuration control options; Reserved.)
20000000	NOOP
20000000	NOOP
20000000	NOOP
20000000	NOOP
20000000	NOOP
20000000	NOOP
20000000	NOOP
20000000	NOOP
30002001	Packet Type 1: Write FAR register, WORD_COUNT=1
00000000	FAR (Frame address) = 00000000 (hex)
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
00000001	CMD[4:0]=00001 (binary) = WCFG (Write configuration data)
20000000	NOOP
30004000	Packet Type 1: Write FDRI register, WORD_COUNT=0
502BA520	Packet Type 2: Write FDRI register, WORD_COUNT=2860320
00000000	FDRI data word 1 (First bitstream configuration data word)
00000000	FDRI data word 2
…	FDRI data words 3-2860319
00000000	FDRI data word 2860320 (Last bitstream configuration data word)
30000001	Packet Type 1: Write CRC register, WORD_COUNT=1
C81874CB	CRC[31:0]=C81874CB (hex) (If written CRC does not match computed CRC value, then error)
20000000	NOOP
20000000	NOOP
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
0000000A	CMD[4:0]=01010 (binary) = GRESTORE (Pulse GRESTORE signal)
20000000	NOOP
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
00000003	CMD=00011 (binary) = DGHIGH/LFRM (De-assert GHIGH_B)

Table 5-19: **Sample XC7K325T Bitstream *(Cont'd)***

Configuration Data Word (hex)	Description
20000000	NOOP
…	…
20000000	NOOP
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
00000005	CMD[4:0]=00101 (binary) = START (Begin STARTUP sequence)
20000000	NOOP
30002001	Packet Type 1: Write FAR register, WORD_COUNT=1
03BE0000	FAR (Frame address) = 03BE0000 (hex)
3000C001	Packet Type 1: Write MASK register, WORD_COUNT=1
00000501	Bit mask for write to CTL0/CTL1 register. MASK[31:0]=00000501 (hex)
3000A001	Packet Type 1: Write CTL0 register, WORD_COUNT=1
00000501	CTL0[31:0]=00000501 (hex) (Note: Also check prior MASK.) (Sets configuration control options, e.g. fallback, readback, etc.)
30000001	Packet Type 1: Write CRC register, WORD_COUNT=1
E3AD7EA5	CRC[31:0]=E3AD7EA5 (hex) (If written CRC does not match computed CRC value, then error)
20000000	NOOP
20000000	NOOP
30008001	Packet Type 1: Write CMD register, WORD_COUNT=1
0000000D	CMD[4:0]=01101 (binary) = DESYNCH (Reset DALIGN) (Requires new Sync word)
20000000	NOOP (Pad remainder of bitstream with NOOPs)
…	…
20000000	NOOP (End of bitstream)

Configuration Memory Frames

7 series FPGA configuration memory is arranged in frames that are tiled about the device. These frames are the smallest addressable segments of the 7 series FPGA configuration memory space, and all operations must therefore act upon whole configuration frames. Each frame consists of 101 32-bit words. Depending on bitstream 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) is roughly equal to the configuration array size (words) plus configuration overhead (words). Bitstream length (bits) is roughly equal to the bitstream length in words times 32.

Configuration Packets

All 7 series 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 in this section.

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 7 series 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 5-20: **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 5-21: **OPCODE Format**

OPCODE	Function
00	NOP
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 5-22: **Type 2 Packet Header**

Header Type	Opcode	Word Count
[31:29]	[28:27]	[26:0]
010	RR	xxxxxxxxxxxxxxxxxxxxxxxxxxx

Notes:

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

Configuration Registers

Table 5-23 summarizes the Type 1 Packet registers. A detailed explanation of selected registers follows.

Table 5-23: **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 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 Access Register
COR1	Read/Write	01110	Configuration Option Register 1
WBSTAR	Read/Write	10000	Warm Boot Start Address Register
TIMER	Read/Write	10001	Watchdog Timer Register
BOOTSTS	Read	10110	Boot History Status Register
CTL1	Read/Write	11000	Control Register 1
BSPI	Read/Write	11111	BPI/SPI Configuration Options Register

CRC Register (00000)

Writes to this register are used to 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.

Frame Address Register (00001)

The 7 series devices are divided into two halves, the top and the bottom. All frames in 7 series devices have a fixed, identical length of 3,232 bits (101 32-bit words).

The Frame Address Register (FAR) is divided into five fields: block type, top/bottom bit, row address, column address, and minor address (see Table 5-24). 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 5-24: **Frame Address Register Description**

Address Type	Bit Index	Description
Block Type	[25:23]	Valid block types are CLB, I/O, CLK (000), block RAM content (001), and CFG_CLB (010). A normal bitstream does not include type 011.
Top/Bottom Bit	22	Select between top-half rows (0) and bottom-half rows (1).
Row Address	[21:17]	Selects the current row. The row addresses increment from center to top and then reset and increment from center to bottom.
Column Address	[16:7]	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.

FDRI Register (00010)

Writes to this register configure frame data at the frame address specified in the FAR register.

FDRO Register (00011)

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

Command Register (00100)

The Command Register (CMD) 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 register is loaded with a new value. Table 5-25 lists the Command Register commands and codes.

Table 5-25: **Command Register Codes**

Command	Code	Description
NULL	00000	Null command, does nothing.
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: The startup sequence begins after a successful CRC check and a DESYNC command are performed.

Table 5-25: **Command Register Codes *(Cont'd)***

Command	Code	Description
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.
Reserved	01110	Reserved.
IPROG	01111	Internal PROG for triggering a warm boot.
CRCC	10000	When readback CRC is selected, the configuration logic recalculates the first readback CRC value after reconfiguration. Toggling GHIGH has the same effect. This command can be used when GHIGH is not toggled during the reconfiguration case.
LTIMER	10001	Reload Watchdog timer.

Control Register 0 (00101)

Control Register 0 (CTL0) is used to configure the 7 series 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 5-26 and described in Table 5-27.

Table 5-26: **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	x	x	0	x	0	0	0	0 0	0	x x	1

Table 5-27: **Control Register 0 Description**

Name	Bit Index	Description
EFUSE_KEY	31	Selects the AES key source: 0: Battery-backed 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	ICAPE2 Port Select. 0: Top ICAPE2 Port Enabled (default) 1: Bottom ICAPE2 Port Enabled
OverTempPowerDown	12	Enables the XADC Over-Temperature power down. 0: Disables Over-Temperature power down (default) 1: Enables Over-Temperature power down
ConfigFallback	10	Stops when configuration fails and disables fallback to the default bitstream. The bitstream 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. 1: Does not mask changeable memory cell readback values. 0: Masks changeable memory cell readback value, such as distributed RAM or SRL (default).
FARSRC	7	Determines the output of FAR[23:0] configuration register. 0: EFAR, the address of ECC error frame 1: FAR, the address of RBCRC (default)
DEC	6	AES Decryptor enable bit. 0: Decryptor disabled (default) 1: Decryptor enabled

Table 5-27: **Control Register 0 Description *(Cont'd)***

Name	Bit Index	Description
SBITS[1:0]	[5:4]	Security level. The 7 series FPGA security level is extended to encrypted bitstreams. It is applicable to the Configuration port, not to ICAPE2. 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 the **`Persist`** option in Preparing a Design for Readback in Chapter 6. `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)

MASK Register (`00110`)

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

Status Register (`00111`)

The Status Register (STAT) indicates the value of numerous global signals. The register can be read through the SelectMAP or JTAG interfaces. Table 5-28 gives the name of each bit position in the STAT register; a detailed explanation of each bit position is given in Table 5-29.

Table 5-28: **Status Register**

Description	Reserved					BUS_WIDTH		Reserved				STARTUP_STATE			XADC_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 5-29: **Status Register Description**

Name	Bit Index	Description
BUS_WIDTH	[26:25]	Configuration bus width auto detection result. For BPI and SelectMAP modes, value is set to 01 (x8) after mode pins are sampled and before bus width detection. If ICAPE2 is enabled, this field reflects the ICAPE2 bus width after configuration is done. 00 = x1 01 = x8 10 = x16 11 = x32
STARTUP_STATE	[20:18]	Configuration 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
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

Table 5-29: **Status Register Description *(Cont'd)***

Name	Bit Index	Description
GTS_CFG_B	5	Status of GTS_CFG_B: 0: All user I/Os are placed in High-Z state 1: All I/Os behave as configured
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

LOUT Register (01000)

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

Configuration Options Register 0 (01001)

Configuration Options Register 0 (COR0) is used to set certain configuration options for the device. The name of each bit position in the COR0 is given in Table 5-30 and described in Table 5-31.

Table 5-30: **Configuration Options Register 0**

Description	Reserved	Reserved	Reserved	Reserved	PWRDWN_STAT	Reserved	DONE_PIPE	DRIVE_DONE	SINGLE	OSCFSEL						SSCLKSRC		DONE_CYCLE			MATCH_CYCLE			LOCK_CYCLE			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	1	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 5-31: **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 (default) This option adds a register to the DONE signal prior to being read by any configuration logic. The FPGA waits on DONE that is delayed by one StartupClk cycle.
DRIVE_DONE	24	0: DONE pin is open drain 1: DONE is actively driven High (not recommended)
SINGLE	23	0: Readback is not single-shot New captured values are loaded on each successive CAP assertion on the CAPTUREE2 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 CAPTUREE2 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 6 111: Keep (not recommended)
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 5-31: **Configuration Options Register 0 Description *(Cont'd)***

Name	Bit Index	Description
LOCK_CYCLE	[8:6]	Startup cycle to stall in until MMCMs lock. 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
GTS_CYCLE	[5:3]	Startup cycle to deassert the 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. Bitstream option **-g GTS_cycle:Done** 111: Keep (not recommended)
GWE_CYCLE	[2:0]	Startup phase to deassert the 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. Bitstream option **-g GWE_cycle:Done** 111: Keep (not recommended)

MFWR Register (01010)

This register is used by the bitstream compression option.

CBC Register (01011)

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

IDCODE Register (01100)

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 (01101)

This register supports the USR_ACCESSE2 primitive. This primitive provides direct FPGA logic access to a 32-bit value stored by the FPGA bitstream, or re-written through the external or internal configuration ports as a configuration register. USR_ACCESS can be configured with a 32-bit user-specified value or automatically loaded by the bitstream generation program with a timestamp. The user-specified value can be used for revision, design tracking, or serial numbers. The timestamp feature is useful when several implementation runs have been performed but the source design itself is unchanged. For more details on this primitive, see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*. For more details on using the primitive and the AXSS register, see XAPP497, *Bitstream Identification with USR_ACCESS*.

Configuration Options Register 1 (01110)

Configuration Options Register 1 (COR1) is used to set certain configuration options for the device. The name of each bit position in the COR1 is given in Table 5-32 and described in Table 5-33.

Table 5-32: **Configuration Options Register 1**

Description	Reserved	PERSIST_DEASSERT_AT_DESYNC	RBCRC_ACTION	Reserved	RBCRC_NO_PIN	RBCRC_EN	Reserved	BPI_1ST_READ_CYCLE	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 5-33: **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 01: Halt 11: Correct_And_Halt 10: Correct_And_Continue

Table 5-33: **Configuration Options Register 1 Description *(Cont'd)***

Name	Bit Index	Description
RBCRC_NO_PIN	9	Controls INIT_B as a readback CRC error status output pin. 0: Enables INIT_B as a readback CRC error status output pin (default) 1: Disables INIT_B as a readback CRC error status output pin
RBCRC_EN	8	Controls continuous readback CRC enable. 0: Disables continuous readback CRC (default) 1: Enables continuous readback CRC
BPI_1ST_READ_CYCLE	[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 (10000)

The warm boot start address register (WBSTAR) specifies the MultiBoot address location to be used when the IPROG command is applied. The name of each bit position in the warm boot start address register is given in Table 5-34 and described in Table 5-35.

Table 5-34: **WBSTAR Register**

Description	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 5-35: **WBSTAR Register Description**

Name	Bit Index	Description
RS[1:0]	[31:30]	RS[1:0] pin value on next warm boot. The default is 00.
RS_TS_B	29	RS[1:0] pins 3-state enable. 0: 3-state enabled (RS[1:0] disabled) (default) 1: 3-state disabled (RS[1:0] enabled)
START_ADDR	[28:0]	Next bitstream start address. The default start address is address zero.

Watchdog Timer Register (10001)

The Watchdog timer is automatically disabled for fallback bitstreams. The name of each bit position in the Watchdog Timer Register (TIMER) is given in Table 5-36 and described in Table 5-37.

Table 5-36: **TIMER Register**

Description	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 5-37: **TIMER Register Description**

Name	Bit Index	Description
TIMER_USR_MON	31	Watchdog is enabled during user mode: 0: Disabled (default) 1: Enabled
TIMER_CFG_MON	30	Watchdog is enabled during configuration: 0: Disabled (default) 1: Enabled
TIMER_VALUE	[29:0]	Watchdog time-out value. The watchdog clock is approximately 125 KHz to 380 KHz. The default value is zero.

Boot History Status Register (10110)

The Boot History Status Register (BOOTSTS) 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 5-38 and described in Table 5-39.

Table 5-38: **BOOTSTS Register**

Description	Reserved	HMAC_ERROR_1	WRAP_ERROR_1	CRC_ERROR_1	ID_ERROR_1	WTO_ERROR_1	IPROG_1	FALLBACK_1	VALID_1	HMAC_ERROR_0	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 5-39: **BOOTSTS Register Description**

Name	Bit Index	Description
HMAC_ERROR_1	15	HMAC error
WRAP_ERROR_1	14	BPI address counter wraparound error, supported in asynchronous read mode
CRC_ERROR_1	13	CRC error
ID_ERROR_1	12	ID_CODE 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 1 is valid
HMAC_ERROR_0	7	HMAC error
WRAP_ERROR_0	6	BPI address counter wraparound error, supported in asynchronous read mode
CRC_ERROR_0	5	CRC error
ID_ERROR_0	4	ID_CODE 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

Table 5-39: **BOOTSTS Register Description *(Cont'd)***

Name	Bit Index	Description
VALID_0	0	Status 0 is 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.

Control Register 1 (`11000`)

Control Register 1 (CTL1) is used to configure the 7 series device. This register is reserved. See Table 5-40.

Table 5-40: **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

BPI/SPI Configuration Options Register (`11111`)

The BPI/SPI configuration options register (BSPI) is reserved and is used to store certain configuration options for the device set by the tools. The name of each bit position in the COR1 is given in Table 5-41.

Table 5-41: **BPI/SPI Configuration Options Register**

Description	Reserved				BPI_sync_mode(1)	Read Configuration Register (determined by BPI_sync_mode; see XAPP587 for more information)															Reserved		SPI_buswidth(2)		SPI_read_opcode (see Table 2-13, page 53)							
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	1	0	1	1

Notes:

1. See Synchronous Read Mode Support, page 61.
2. See Master SPI Dual (x2) and Quad (x4) Read Commands, page 52.

Device Identifier and Device DNA

The 7 series FPGA contains an embedded, 64-bit device identifier which is used to provide a 57-bit Device DNA value. The identifier is nonvolatile, permanently programmed by Xilinx into the FPGA, and is unchangeable making it tamper resistant. Each device is programmed with a 57-bit DNA value that is most often unique. However, up to 32 devices within the family can contain the same DNA value. The JTAG FUSE_DNA

command can be used to read the entire 64-bit value that is always unique. Device DNA is composed of bits 63 to 7 of the 64-bit FUSE_DNA value.

External applications can access the Device DNA or FUSE_DNA values through the JTAG port, and FPGA designs can access the DNA only through a Device DNA Access Port (DNA_PORT).

The FPGA application accesses the DNA value using the Device DNA Access Port (DNA_PORT) design primitive, shown in Figure 5-15.

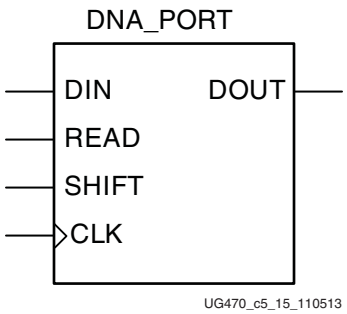


Figure 5-15: **7 Series FPGA DNA_PORT Design Primitive**

DNA Value

As shown in Figure 5-16, the Device DNA value is 57 bits long.

Operation

Figure 5-16 shows the general functionality of the DNA_PORT design primitive. An FPGA application must first instantiate the DNA_PORT primitive, shown in Figure 5-15, within a design.

To read the Device DNA, the FPGA application must first transfer the DNA value into the DNA_PORT output shift register. The READ input must be asserted during a rising edge of CLK, as shown in Table 5-42. This action parallel loads the output shift register with all 57 bits of the DNA. The READ operation overrides a SHIFT operation.

To continue reading the DNA bits, assert SHIFT followed by a rising edge of CLK, as shown in Table 5-42. 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.

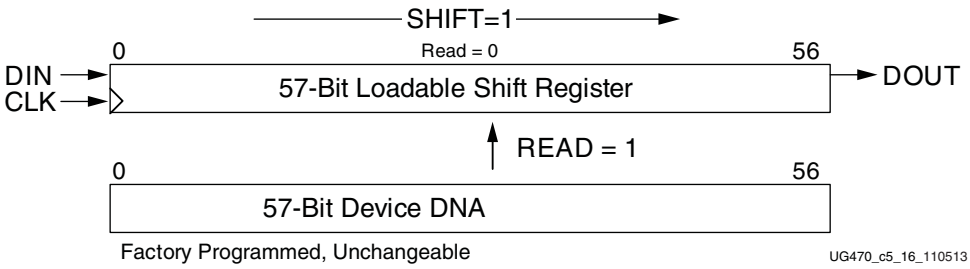


Figure 5-16: **DNA_PORT Operation**

If both READ and SHIFT are Low, the output shift register holds its value and DOUT remains unchanged. Refer to the respective 7 series FPGAs data sheet for DNA memory specifications.

Table 5-42: **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 DNA	Bit 56 of Identifier
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 DNA Length

As shown in Figure 5-17, most applications that use the DNA_PORT primitive tie the DIN data input to a static value.

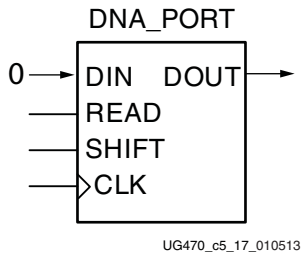


Figure 5-17: **Shift in Constant**

As shown in Figure 5-18, the length of the DNA can be extended by feeding the DOUT serial output port back into the DIN serial input port. This way, the DNA can be extended to any possible length.

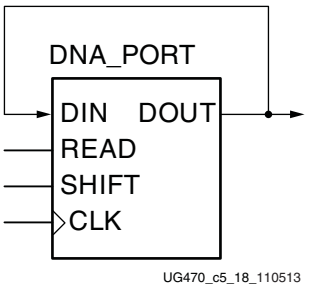


Figure 5-18: **Circular Shift**

It is also possible to add additional bits to the DNA using FPGA logic resources. As shown in Figure 5-19, 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.

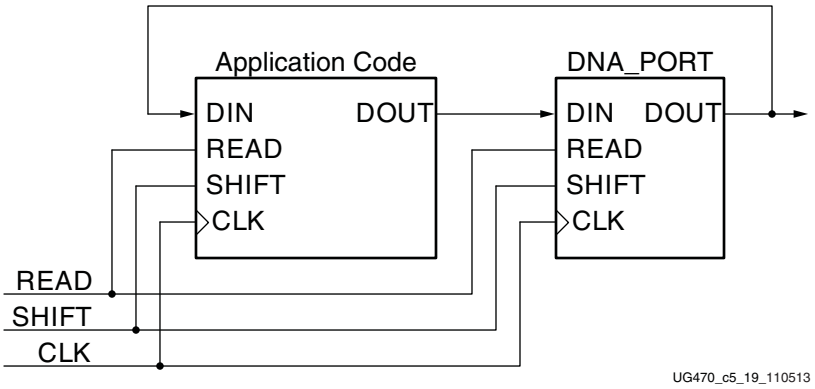


Figure 5-19: **Bitstream Specific Code**

JTAG Access to Device DNA and Identifier

The FPGA's internal Device DNA can be read via the JTAG port using the private XSC_DNA command. This requires the ISC_ENABLE to be loaded before the XSC_DNA command is issued. ISC_ENABLE stops an FPGA design. An ISC_DISABLE is required to restart an FPGA design after the XSC_DNA command is issued.

Bit 56 of the identifier, shown in Figure 5-16, appears on the TDO JTAG output following the XSC_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.

The entire unique 64-bit identifier can read via the JTAG port using the private FUSE_DNA command. The functionality is similar to XSC_DNA, except that bit 0 of the identifier appears on the TDO JTAG output following the FUSE_DNA command.

iMPACT Access to Device Identifier

The iMPACT software can also read the 57-bit Device DNA value.
`readDna -p <position>` is the batch command that reads the Device DNA from the FPGA. The Vivado device programmer will also have the ability to read the Device DNA value.

XILINX®

Chapter 6

Readback and Configuration Verification

Xilinx® 7 series devices allow users to read configuration memory through the SelectMAP, ICAPE2, 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, SRL, 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. After initiation the device dumps the contents of its configuration memory to the SelectMAP or JTAG interface.

Users can send the readback command sequence from a custom microprocessor, CPLD, or FPGA-based system, or use the tools 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 7 series 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 ChipScope™ Pro tool provides superior design debugging functionality in a user-friendly interface. Similar functionality is provided in the Vivado lab tools.

After 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.

The following sections provide instructions for performing readback of monolithic devices through the SelectMAP or JTAG interfaces. Readback through ICAPE2 is similar to readback through the SelectMAP interface. For SSI devices, only readback verify is supported via the iMPACT tool.

Preparing a Design for Readback

There are two mandatory bitstream settings for readback: the security setting must not prohibit readback (**`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 (**`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 CAPTUREE2 primitive can instantiated in the user design. 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.

Persist Option

The persist bitstream option maintains the configuration logic access to the multi-function configuration pins after configuration. The persist option is primarily used to maintain the SelectMAP port after configuration for readback access, but persist can be used with any configuration mode. Persist is not needed for JTAG configuration since the JTAG port is dedicated and always available. Persist and ICAP cannot be used at the same time.

The persist option can also be used to reconfigure the device from an external controller without pulsing the PROGRAM_B pin or using the JTAG port. The multi-function pins that persist depend on the configuration mode pin settings, and are the same as those shown for each configuration mode in Table 2-2, page 22 and Table 2-3, page 23, except that PUDC_B and DOUT_CSO_B never persist. Any I/O pins that persist cannot be used as I/O in the user design. Use the CONFIG_MODE constraint to reserve the correct pins during implementation of the design. Persisted I/O use the general-purpose I/O standard default of LVCMOS, 12 mA drive, Slow slew rate.

Readback Command Sequences

Accessing Configuration Registers through the SelectMAP Interface

To read configuration memory through the SelectMAP interface, 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 to avoid an abort (refer to SelectMAP ABORT for details).

The procedure for changing the SelectMAP interface between Write and Read Control 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. Readback data is valid deterministically three clock cycles after the CSI_B pin is asserted during readback

This procedure is illustrated in Figure 6-1.

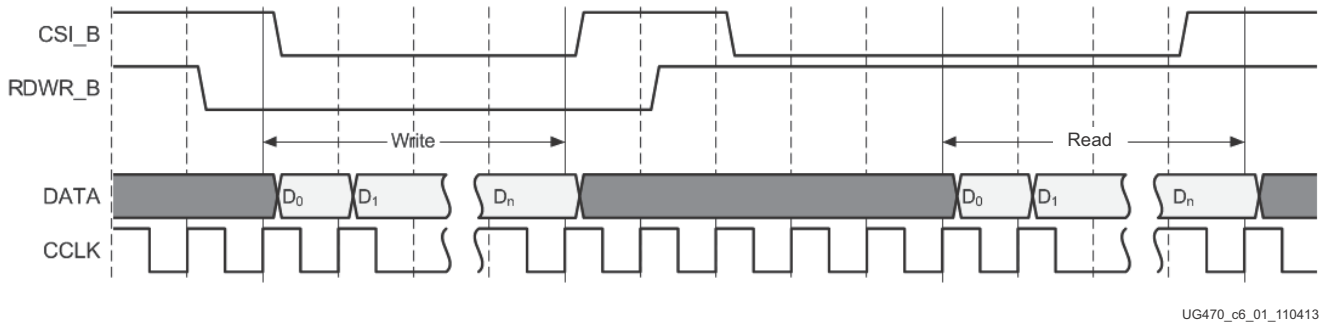


Figure 6-1: **Changing the SelectMAP Port from Write to Read Control**

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 at least one NOOP.
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 6-1: **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	2800E001	Write Type1 packet header to read STAT register
8	Write	20000000	NOOP
9	Write	20000000	NOOP
10	Read	SSSSSSSS	Device writes one word from the STAT register to the configuration interface
11	Write	30008001	Type 1 Write 1 Word to CMD

Table 6-1: **Status Register Readback Command Sequence (SelectMAP) *(Cont'd)***

Step	SelectMAP Port Direction	Configuration Data	Explanation
12	Write	0000000D	DESYNC Command
13	Write	20000000	NOOP
14	Write	20000000	NOOP

The user must change the SelectMAP interface from write to read control between steps 8 and 9, and back to write control after step 9.

To read registers other than STAT, the address specified in the Type-1 packet header in step 2 of Table 6-1 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 at least 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. 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 two dummy words 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.
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 6-2 shows the readback command sequence.

Table 6-2: **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	02000000	Type 1 NOOP Word 0
3	Write	30008001	Type 1 Write 1 Word to CMD
		0000000B	SHUTDOWN Command
		02000000	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 147,662 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 (first 101 words are a dummy frame)
		. . .	
		00000000	Packet Data Read FDRO Word 147661
11	Write	20000000	Type 1 NOOP Word 0

Table 6-2: **Shutdown Readback Command Sequence (SelectMAP) *(Cont'd)***

Step	SelectMAP Port Direction	Configuration Data	Explanation
12	Write	30008001	Type 1 Write 1 Word to CMD
		00000005	START Command
		20000000	Type 1 NOOP Word 0
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

Accessing Configuration Registers through the JTAG Interface

JTAG access to the 7 series 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 6-3.

Table 6-3: **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 five bits of the CFG_IN or CFG_OUT instruction, LSB first	00101 (CFG_IN) 00100 (CFG_OUT)	0	5
6	Shift the MSB of the CFG_IN or CFG_OUT instruction while exiting SHIFT-IR	0	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

Table 6-3: **Shifting in the JTAG CFG_IN and CFG_OUT Instructions *(Cont'd)***

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
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 at least 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 6-4: **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
2	Shift the first five bits of the CFG_IN instruction, LSB first.	00101 (CFG_IN)	0	5
	Shift the MSB of the CFG_IN instruction while exiting SHIFT-IR.	0	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2

Table 6-4: **Status Register Readback Command Sequence (JTAG) *(Cont'd)***

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
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 five bits of the CFG_OUT instruction, LSB first.	00100 (CFG_OUT)	0	5
	Shift the MSB of the CFG_OUT instruction while exiting Shift-IR.	0	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 at least one 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 at least one 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 6-5: **Shutdown Readback Command Sequence (JTAG)**

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
	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
2	Shift the first five bits of the CFG_IN instruction, LSB first.	`00101`	0	5
	Shift the MSB of the CFG_IN instruction while exiting Shift-IR.	0	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2

Table 6-5: **Shutdown Readback Command Sequence (JTAG) *(Cont'd)***

Step	Description	Set and Hold		# of Clocks (TCK)
		TDI	TMS	
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 five bits of the JSHUTDOWN instruction, LSB first.	01101	0	5
	Shift the MSB of the JSHUTDOWN instruction while exiting SHIFT-IR.	0	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 five bits of the CFG_IN instruction, LSB first.	00101	0	5
	Shift the MSB of the CFG_IN instruction while exiting SHIFT-IR.	0	1	1
	Move into the SELECT-DR state.	X	1	2
	Move into the SHIFT-DR state.	X	0	2

Table 6-5: **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 five bits of the CFG_OUT instruction, LSB first.	00100 (CFG_OUT)	0	5
	Shift the MSB of the CFG_OUT instruction while exiting Shift-IR.	0	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 6-2.

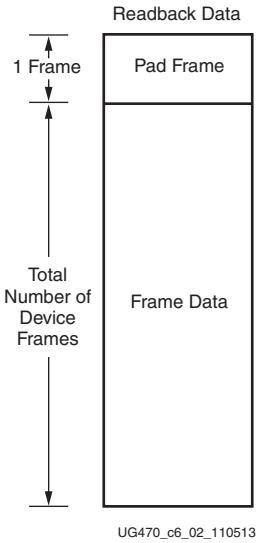


Figure 6-2: **Readback Data Stream**

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, such as block or distributed RAM, SRLs, or DRP memories, 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. A `1` in the mask file indicates a bitstream location that should be a *don't care*. After readback data has been obtained from the device, either of these comparison procedures can be used:

1. Compare readback data to the RBD *golden* readback file. Mask by using the MSD file (see Figure 6-3).

   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.

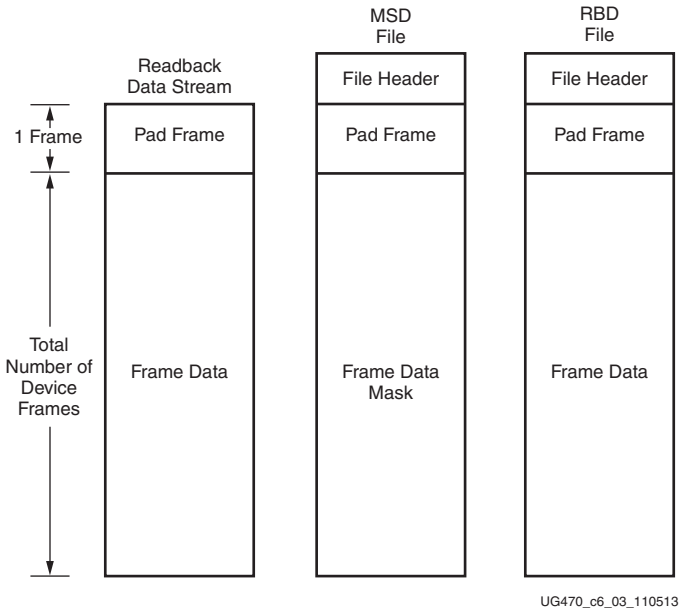


Figure 6-3: **Comparing Readback Data Using the MSD and RBD Files**

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 6-4).

   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.

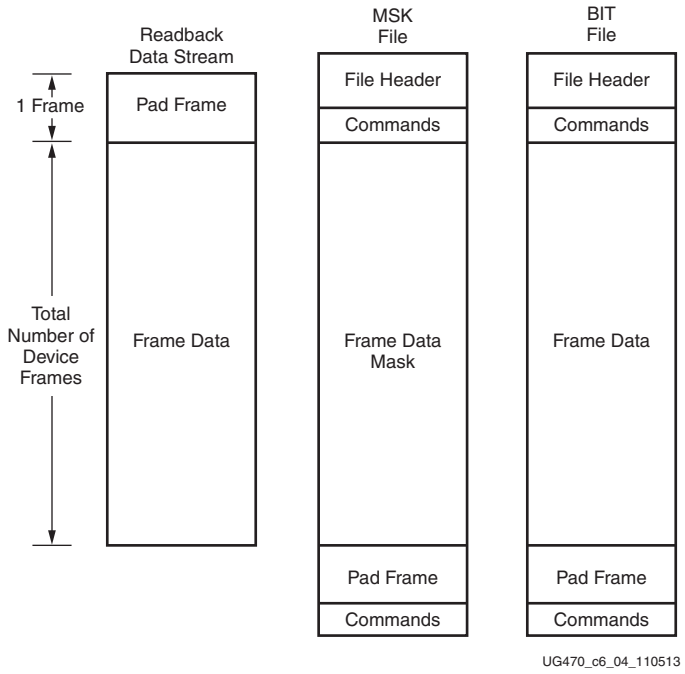


Figure 6-4: **Comparing Readback Data Using the MSK and BIT Files**

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 CAPTUREE2 primitive in their design and asserting the CAP input on that primitive while the design is operating. On the next rising clock edge on the CAPTUREE2 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 7 series 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 start-up clock setting.

XILINX

Chapter 7

Reconfiguration and MultiBoot

This chapter focuses on full bitstream reconfiguration methods in 7 series FPGAs.

Fallback MultiBoot

Overview

The 7 series FPGAs MultiBoot and fallback features support updating systems in the field. Bitstream images can be upgraded dynamically in the field. The FPGA MultiBoot feature enables switching between images on the fly. When an error is detected during the MultiBoot configuration process, the FPGA can trigger a fallback feature that ensures a known good design can be loaded into the device.

When fallback happens, an internally generated pulse resets the entire configuration logic, except for the dedicated MultiBoot logic, the warm boot start address (WBSTAR), and the boot status (BOOTSTS) registers. This reset pulse pulls INIT_B and DONE Low, clears the configuration memory, and restarts the configuration process from address 0 with the revision select (RS) pins driven to 00. After the reset, the bitstream overwrites the WBSTAR starting address.

During configuration, the following errors can trigger fallback:

- An IDCODE error
- A CRC error
- A Watchdog timer time-out error
- A BPI address wraparound error

Fallback can also be enabled with the bitstream option **`ConfigFallback`**. Embedded IPROG is ignored during fallback reconfiguration. The Watchdog timer is disabled during fallback reconfiguration. If fallback reconfiguration fails, configuration stops and both INIT_B and DONE are held Low.

Implementation of a robust in-system update solution involves a set of decisions. First, a method for system setup needs to be determined. Next, design considerations can be added for a specific configuration mode. Finally, HDL design considerations need to be taken into account and files need to be generated properly. This chapter walks through each stage of this process.

The MultiBoot and fallback feature can be used with all master configuration modes. See SPI Densities over 128 Mb, page 53 for requirements in SPI mode. Fallback MultiBoot is not supported for the GTZ transceivers in the Virtex-7 HT FPGAs. The GTZ transceivers can be reconfigured after MultiBoot using the DRP.

Golden Image Initial System Setup

The golden image is loaded from address space 0 at power up. Next, the golden image design triggers a MultiBoot image to be loaded. This step is beneficial when initial system checking is required prior to loading a run-time image. The system checking or diagnostics can be contained in the golden image, and the run-time operation can be contained in the MultiBoot image. At power up, the golden image is always loaded. This design triggers booting from an upper address space. Multiple MultiBoot images can also exist, and any design can trigger any other image to be loaded. If an error occurs during loading of the MultiBoot image from the upper address space, the fallback circuitry triggers the golden image to be loaded from address 0.

Figure 7-1 shows the flow for the initial setup of the golden image.

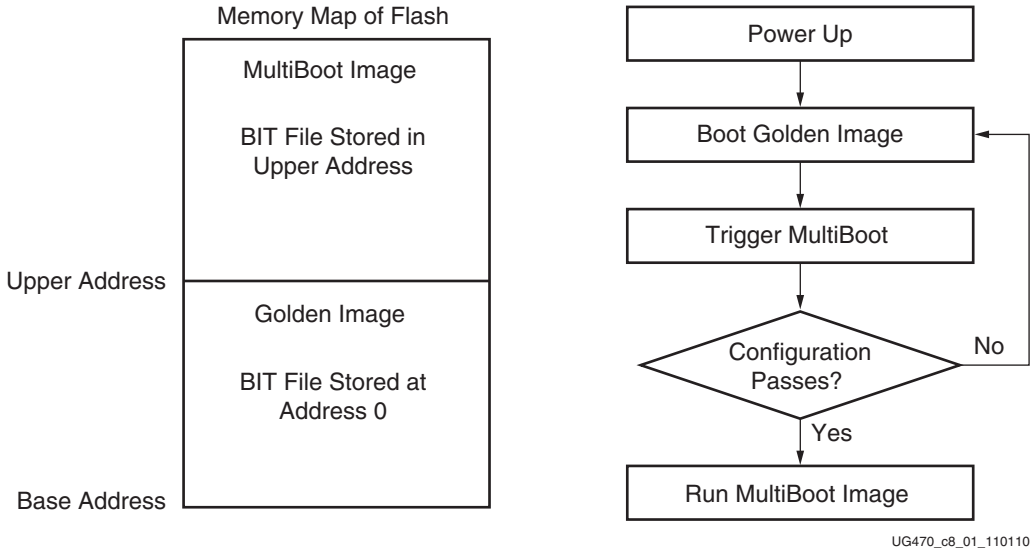


Figure 7-1: **Initial Golden Image Flow Diagram**

Golden Image and MultiBoot Image Design Requirements

The design requirements for both golden and MultiBoot images are:

- There are no hardware specific requirements, except when using RS[1:0] pins for address control in the BPI mode; see the BPI - Hardware RS Pin Design Considerations section.
- The IPROG command is embedded in the golden image for the next configuration address, or is issued via code through ICAPE2 within the golden image design.
- The WBSTAR register is set to jump to an address in the bitstream or via ICAPE2.
- The MultiBoot image must be stored in flash at the address in the WBSTAR register.
- The Watchdog timer is enabled in order to recover from incompletely programmed bitstreams.

Initial MultiBoot Image System Setup

The MultiBoot image is first loaded at power up from an upper address space. If this image fails configuration, the device automatically triggers a fallback to the golden image stored at address 0. This enables systems to upgrade their own bit files and then boot from power

up to the latest image. The upgrade process can occur, and then the design can trigger a reload of the most recent version of the design. Fallback logic ensures the system recovers from any failure to load the MultiBoot image and loads the golden image. The golden image can then fix any errors in the flash and trigger a configuration from the MultiBoot image again.

Figure 7-2 shows the flow for the initial setup of the MultiBoot image.

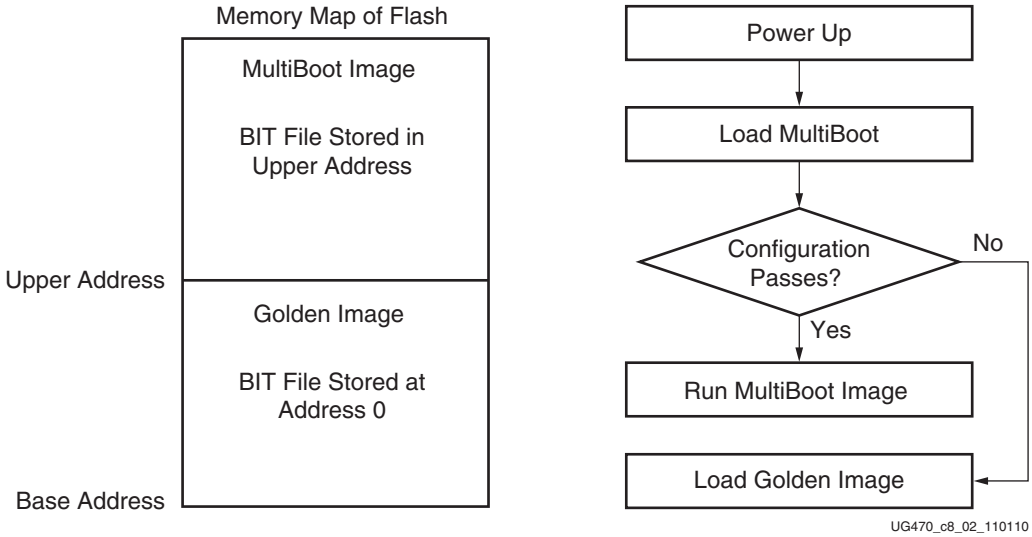


Figure 7-2: **Initial MultiBoot Image Flow Diagram**

Initial MultiBoot Design Considerations

Design considerations for the golden image are:

- The WBSTAR setting in the bitstream points to the MultiBoot location.
- An IPROG command is inserted to trigger loading of MultiBoot at power up.
- The Configuration timer is enabled.
- ICAPE2 instantiated with code to issue an IPROG command can also be included if the golden image can repair the flash and trigger another loading from the MultiBoot image.
- For SPI configuration mode, fallback loading of the golden images always uses the x1 configuration mode.
- For BPI configuration mode, fallback loading of the golden image always uses the asynchronous read mode. Consequently, bitstream settings for synchronous read mode (BPI_sync_mode) are ignored during fallback loading of the golden image. As a result, the golden image settings for CCLK frequency (ConfigRate or ExtMasterCclk_en) must be set within the system capabilities for asynchronous read operation.

Design considerations for the MultiBoot image are:

- The WBSTAR setting in the bitstream points to the MultiBoot location.
- The Configuration timer is enabled.
- ICAPE2 instantiated with code to issue an IPROG command can also be included if the MultiBoot image can upgrade the flash and trigger another loading from the upgraded MultiBoot image.

BPI - Hardware RS Pin Design Considerations

In BPI mode, the RS pins need to be wired to upper address bits with a pull-up resistor on one of the RS pins tied to an upper address line. With this hardware implementation, the system is exclusive of the WBSTAR address, and the bitstream options are the same for each image. Refer to RS Pins, page 141 for further details.

Process Details for MultiBoot and Fallback

The FPGA MultiBoot and fallback events can involve several configuration components in the FPGA. This section provides details about each FPGA configuration command, register, bitstream setting, and pin that can be involved in a MultiBoot or Fallback event.

IPROG

The Internal PROGRAM (IPROG) command is a subset of the functionality of pulsing the PROGRAM_B pin. The fundamental difference is that the IPROG command does not erase the WBSTAR, TIMER, and other internal registers used to initiate MultiBoot and fallback. The IPROG command triggers an initialization, and both INIT and DONE go Low when the IPROG command is issued followed by an attempt to configure.

This command can be issued one of two ways. In the first way, the IPROG command can be issued via the ICAPE2, which is controlled by user logic. This allows user logic to initiate device reconfiguration. In the second way, the IPROG command can be embedded in the bitstream during bitstream generation. In this scenario, the WBSTAR and IPROG commands are set at the beginning of the golden bit file. At power up, the device starts reading the BIT file from the flash and reads in the WBSTAR register and IPROG command. The IPROG command triggers the device to reload from the address specified. If there is an issue with the upper image, the base address is loaded again. At this point, the IPROG command is skipped by the configuration controller because the device saw an error. A fallback condition blocks the IPROG command from being processed, and the device continues to load the golden image. After a successful configuration, the IPROG command can be issued to the device, which enables the golden image to trigger configuration from a MultiBoot image.

WBSTAR Register

The WBSTAR (Warm Boot Start Address) register holds the address that the configuration controller uses after an IPROG command is issued. This can be either in the form of an address or values for the RS pins. This register can be loaded from the bitstream or from the ICAPE2. If the register is not set in the bitstream, it is loaded with a default value of 0s. Therefore, after the golden image sets the WBSTAR value and initiates a multiboot configuration, the multiboot pattern will reset the WBSTAR to 0 by default.

At power up, the device issues the read command to the flash followed by a start address of 0. After the WBSTAR command has been loaded and the IPROG command is issued, the configuration controller issues the read command from the address specified by the WBSTAR address.

Watchdog Timer

The Watchdog timer has two modes (configuration monitor and user logic monitor), which are mutually exclusive of each other.

In configuration monitor mode, the timer register is set in the BIT file. This timer value is then used for both the configuration of the bitstream, which sets the value, as well as any

subsequent loads triggered by an IPROG command. The timer register needs to be set in all BIT files.

The timer register counts down from the start to the bitstream and is disabled by the end of the start-up sequence. If the count reaches 0, a fallback is triggered. The start-up sequence can be delayed by the PLL Wait or DCI Match settings; these delays need to be taken into account. The timer register runs at approximately 65 MHz. For situations where configuration does not begin, or begins properly but does not complete, such as for an invalid or a partially corrupted configuration source, the Watchdog timer allows the device to automatically re-attempt configuration after a reasonable delay.

RS Pins

The dual-purpose RS pins are disabled by default. The RS pins drive Low during a fallback for BPI or Master SelectMAP mode, but do not drive Low during SPI mode. For initial MultiBoot systems, the RS pins are wired to upper address bits of the flash and strapped High or Low with a pull-up or pull-down resistor, respectively. At power up, the system boots to the upper address space defined by the pull-up resistors on the RS and address line connections. During a fallback, the RS pins drive Low and the device boots from address space 0. The RS pins should be tied to upper addresses defined by the system to allow for full bit files to be stored in each memory segment.

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 (10000), page 115) is used during reconfiguration instead of the default address. The default is zero in BPI and SPI modes. The IPROG command can be sent through ICAPE2 or the bitstream. The IPROG Using ICAPE2 and IPROG Embedded in the Bitstream sections describe these two usages. Note that the ICAPE2 interface is similar to the SelectMAP interface, and therefore the input configuration bus needs to be bit-swapped. See Parallel Bus Bit Order, page 79. For more information on ICAPE2, see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*.

IPROG Using ICAPE2

The IPROG command can also be sent using the ICAPE2 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 ICAPE2.

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 (10000), page 115).
3. Send the IPROG command.

Table 7-1 shows an example bitstream for the IPROG command using ICAPE2.

Table 7-1: **Example Bitstream for IPROG through ICAPE2**

Configuration Data (hex)[1]	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

Notes:

1. See Parallel Bus Bit Order, page 79.

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 7-2: **WBSTAR Controlled Pins According to Configuration Mode**

Configuration Mode	Pins Controlled by WBSTAR
Master Serial	RS[1:0]
Master SPI	START_ADDR[23:0] is sent to the SPI device serially. (1)
Master BPI	RS[1:0], A[28:00]
Master SelectMAP	RS[1:0]
JTAG	RS[1:0]
Slave SelectMAP	RS[1:0]
Slave Serial	RS[1:0]

Notes:

1. When using ICAPE2 to set the WBSTAR address, the 24 most significant address bits should be written to WBSTAR[23:0]. For SPI 32-bit addressing mode, WBSTAR[23:0] are sent as address bits [31:8]. The lower 8 bits of the address are undefined and the value could be as high as 0xFF. Any bitstream at the WBSTAR address should contain 256 dummy bytes before the start of the bitstream..

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

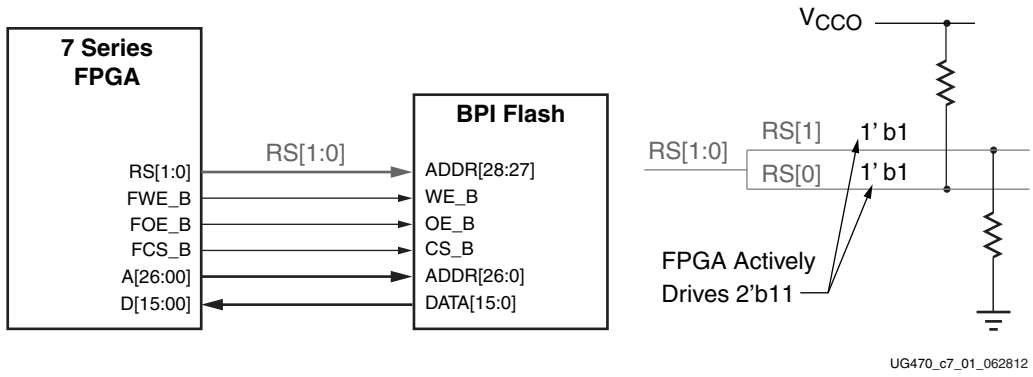


Figure 7-3: **IPROG in BPI Modes**

Notes relevant to Figure 7-3:

1. All BPI pins, except the CCLK pin, are multi-function I/Os. After configuration is finished (the DONE pin goes High), 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 or SPI mode). Later a new application bitstream can be added to flash, by modifying the WBSTAR and the IPROG command in the first bitstream. The FPGA directly loads the new bitstream. If the new bitstream fails, configuration falls back to the original bitstream (see Fallback MultiBoot). The Xilinx tools insert the blank write into WBSTAR and a place holder for the IPROG command in every 7 series FPGA bitstream. For example, WBSTAR can be modified to a user-desired start address (see Warm Boot Start Address Register (`10000`), page 115). A NULL command after WBSTAR can be modified to IPROG by setting the four LSB bits to all ones (see Command Register (`00100`), page 106).

Figure 7-4 illustrates this use model.

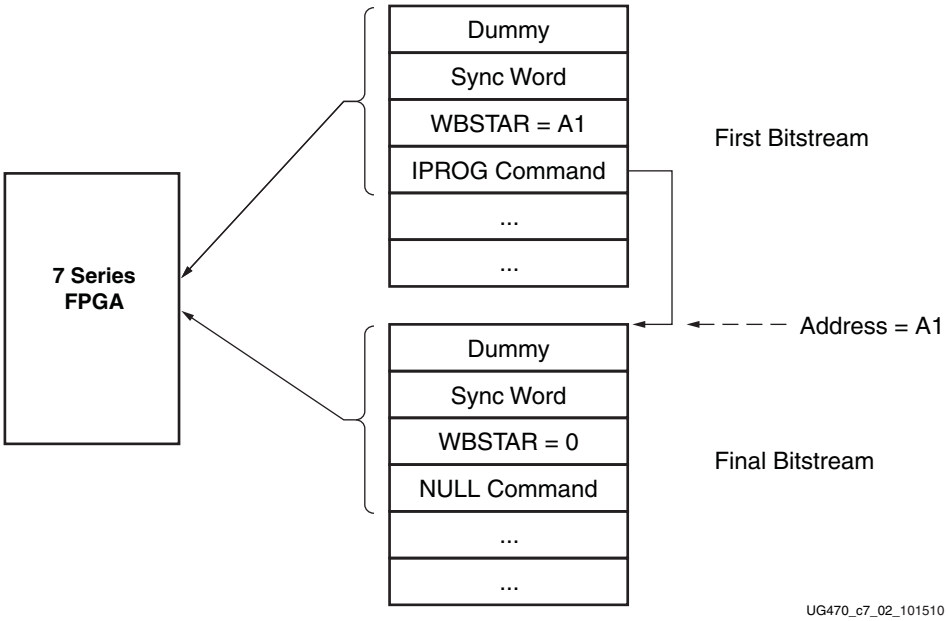


Figure 7-4: **IPROG Embedded in the Bitstream**

Status Register for Fallback and IPROG Reconfiguration

7 series 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 (10110), page 117.

Table 7-3 through Table 7-5 show the BOOTSTS values in some common situations.

Table 7-3: **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 7-4: **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 7-5: **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

Table 7-5: **IPROG Embedded in First Bitstream, Second Bitstream CRC Error, Fallback Successfully**

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

Notes:

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

The Watchdog uses a divided version of a dedicated internal clock, CFGMCLK, which has a nominal frequency of 65 MHz. The clock is predivided by 256, so that the Watchdog clock period is about 4,000 ns. Given the Watchdog counter is 30 bits wide, the maximum possible Watchdog value is about 4,230 seconds. The time value can be set via bitstream options.

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.

Configuration Monitor Mode

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, MMCM lock, or DONE should also be included.

After it is 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.

The Watchdog provides protection against errant configuration operations such as the following.

- Configuration or MultiBoot operations to an invalid start location
- Configuration or MultiBoot operations to a valid start location, but loaded with an incomplete or partially valid configuration bitstream.

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 7-6 shows an example bitstream for reloading the Watchdog using the LTIMER command.

Table 7-6: **Example Bitstream for Reloading the Watchdog with LTIMER**

Configuration Data (hex)[1]	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

Notes:

1. See Parallel Bus Bit Order, page 79.

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

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

Configuration Data (hex)[1]	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

Notes:

1. See Parallel Bus Bit Order, page 79.

Design Examples

The following documentation provides additional examples of using MultiBoot and reconfiguration:

1. XAPP1081, *QuickBoot Method for FPGA Design Remote Update*
2. XAPP733, *Applying MultiBoot and the LogiCORE IP Soft Error Mitigation Controller*

XILINX

Chapter 8

Readback CRC

Xilinx® 7 series 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. SEU mitigation is best implemented by using the Xilinx Soft Error Mitigation IP (SEM IP), available at www.xilinx.com/products/intellectual-property/SEM.htm.

To enable Readback CRC, the CONFIG user constraint POST_CRC is set to **Enable**. After it is enabled, the configuration dedicated logic reads back continuously in the background to check the CRC of the configuration memory content. The frequency is set by the CONFIG constraint POST_CRC_FREQ.

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, or a known value can be supplied setting the CONFIG constraint POST_CRC_SOURCE to PRE_COMPUTED. 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_ECCE2 primitive is driven High (see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*. for more information on the FRAME_ECCE2 primitive). 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 ICAPE2 command, JTAG, or SelectMAP. When the user finishes accessing the configuration logic, readback CRC automatically resumes.

The CONFIG user constraint POST_CRC_ACTION can be used to specify what action to take when an error is found. The options are Halt, Continue, Correct_And_Halt, or Correct_And_Continue. See UG625, *Constraints Guide* for POST_CRC constraint details in the ISE tools, and UG912, *Vivado Design Suite Properties Guide* for details on using the Vivado tools.

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 cannot 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.

These dynamically changeable memory locations are masked during background readback:

- SLICEM LUTs (RAM or SRL)
- Block RAM content is skipped during readback to avoid interfering with user functions. Block RAM is optionally covered by its own ECC circuit during operation.
- Dynamic Reconfiguration Port (DRP) memories are masked.

When enabled, the readback CRC logic automatically runs in the background after configuration is DONE, and when these conditions hold:

- The FPGA is configured 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 8-1.

Table 8-1: **Readback CRC Clock Source**

ICAPE2 Primitive	STARTUPE2 Primitive	POST_CRC _FREQ	Configuration Mode	CCLK	POST_CRC Clock Source
Instantiated	Don't Care	Don't Care	Don't Care	Don't Care	CLK Input of the ICAPE2
Not Instantiated	Instantiated and USRCCLKO Connected	Don't Care	Don't Care	Don't Care	USRCCLKO Input of the STARTUPE2 Primitive
Not Instantiated	Not Instantiated, or Instantiated but not using USRCCLKO	Enabled	Don't Care	Don't Care	Internal Oscillator: Defined by POST_CRC_FREQ
Not Instantiated	Not Instantiated, or Instantiated but not using USRCCLKO	Not Used	Master	EMCCLK Enabled (ExtMasterCclk_en)	EMCCLK Pin Input
Not Instantiated	Not Instantiated, or Instantiated but not using USRCCLKO	Not Used	Master	Default (Internal Oscillator)	Internal Oscillator: Master CCLK Defined by ConfigRate Option

Table 8-1: **Readback CRC Clock Source *(Cont'd)***

ICAPE2 Primitive	STARTUPE2 Primitive	POST_CRC _FREQ	Configuration Mode	CCLK	POST_CRC Clock Source
Not Instantiated	Not Instantiated, or Instantiated but not using USRCCLKO	Not Used	Slave	External Input	CCLK Pin Input
Not Instantiated	Not Instantiated, or Instantiated but not using USRCCLKO	Not Used	JTAG	Not Used	No Clock (see following paragraph)

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 read back 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 Correct_And_Continue, then the readback logic starts over from the first address. If the Correct_And_Halt option is set, the readback logic stops after correction.

Here is a list of different error scenarios and the corresponding behavior of the hardware correction logic when POST_CRC_ACTION is set to Correct_And_Continue. Readback CRC starts scanning from the starting address:

1. Single Bit Error:
   a. After a frame with an erroneous bit is read, ECCERROR is asserted, correction is started, and ECCERROR is deasserted after the bit in error is fixed.
   b. The RBCRC cycle resumes from the starting address without asserting CRCERROR.
2. Two or more errors in different frames:
   a. After a frame with an erroneous bit is read, ECCERROR is asserted, correction is started, and ECCERROR is deasserted after the bit in error is fixed.
   b. The RBCRC cycle restarts from the starting address and continues until it reads the frame with the second bit in error.
   c. Again, the ECCERROR is asserted, a correction is started, and ECCERROR is deasserted after the bit in error is fixed.

d. If this is the last error, RBCRC resumes from the starting address and continues normally without asserting CRCERROR.

e. If there is one more error in a different frame, steps b and c are repeated.

3. Two or more errors in the same frame:

   a. After a frame with erroneous bits is read, ECCERROR is asserted.

   b. The built-in logic cannot correct more than one error in a frame so RBCRC continues on to the next frame, and ECCERROR is updated based on the current frame.

   c. When RBCRC reaches the last address, CRCERROR is set.

   d. RBCRC cycle restarts from the starting address and keeps the CRCERROR flag asserted.

   e. The RBCRC behavior at this point depends on the POST_CRC_ACTION attribute.

Notes on POST_CRC_ACTION:

- If set to Halt, RBCRC stops on the first error and sets the CRCERROR flag.
- If set to Continue, RBCRC asserts CRCERROR if an error is present and continues from the starting address with CRCERROR asserted.
- If set to Correct_And_Halt, RBCRC corrects a single bit error, halts RBCRC, and asserts CRCERROR.
- If set to Correct_And_Continue, RBCRC corrects a single bit error and restarts from the starting address without asserting CRCERROR.

XILINX

Chapter 9

Multiple FPGA Configuration

In applications requiring multiple FPGAs, all of the devices can be configured from a single configuration source. FPGAs that use the same configuration file can be "gang" loaded at the same time. FPGAs that use different configuration files can be loaded sequentially, either through built-in FPGA logic in a "daisy chain," or using external logic. This chapter covers the following topics:

- Serial Configuration Modes
  - Serial Daisy Chain Configuration
  - Ganged Serial Configuration
- Parallel Configuration Modes
  - Multiple Device SelectMAP Configuration
  - Parallel Daisy Chain Configuration
  - Ganged SelectMAP Configuration

Serial Daisy Chain Configuration

Multiple 7 series 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 the DIN pin, passing configuration data along to downstream devices through the 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, all devices are set for Slave Serial mode, and the configuration clock is provided externally. Figure 9-1 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. An SPI daisy chain does not support fallback reconfiguration.

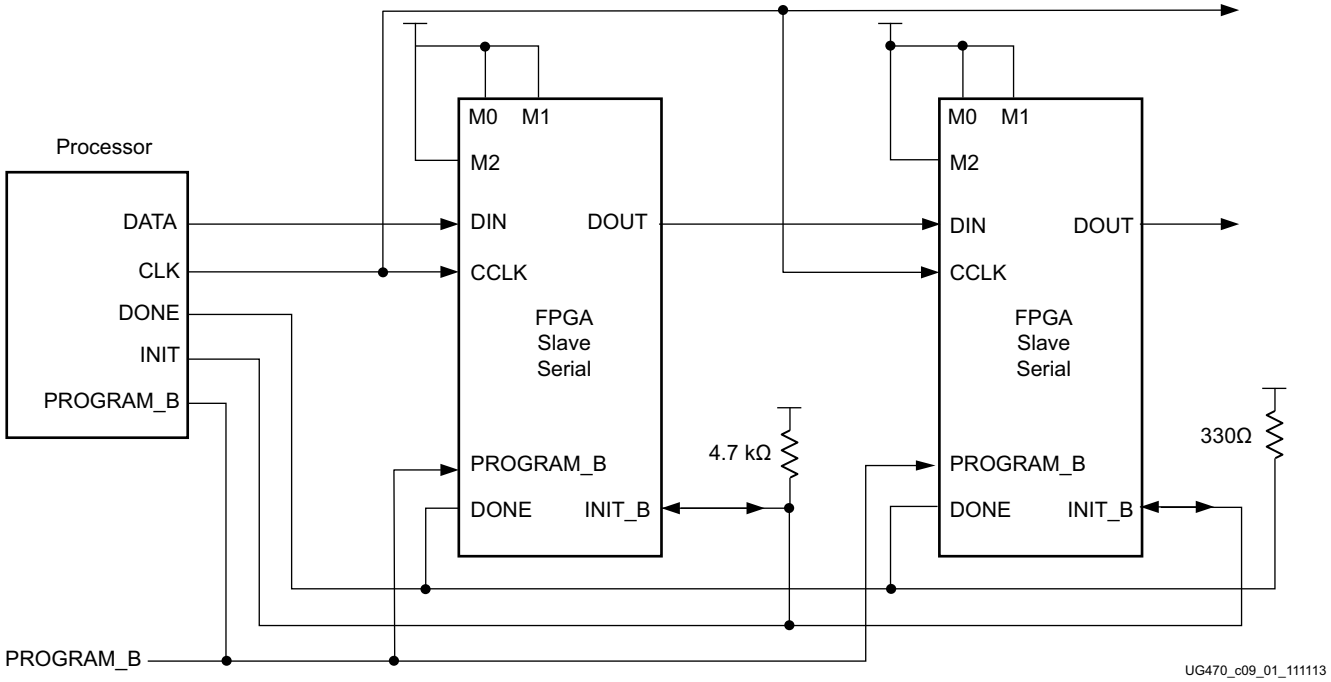


Figure 9-1: **Slave Serial Mode Daisy Chain Configuration**

Notes relevant to Figure 9-1.

1. The DONE pin is an open-drain output. It is important that all DONE pins in a Slave Serial daisy chain be connected. 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 bitstream startup clock setting must be set for CCLK for serial configuration.

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 5.)

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 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.

Mixed Serial Daisy Chains

7 series devices can be daisy-chained with earlier Virtex and Spartan® families. Important design considerations when designing a mixed serial daisy chain are:

- Many older FPGA devices cannot accept as fast a CCLK frequency as a 7 series device can generate. Select a configuration CCLK speed supported by all devices in the chain.

- 7 series devices should always be at the beginning of the serial daisy chain, with older family devices located at the end of the chain.
- All 7 series device families have similar bitstream options. The guidelines provided for 7 series FPGA bitstream options should be applied to all devices in a serial daisy chain, when possible.
- The number of configuration bits that a device can pass through its DOUT pin is limited. For 7 series and Virtex devices starting with the Virtex-II family, the limit is 4,294,967,264 bits. For the Spartan-6 and Spartan-3 generation devices, the limit is 2,147,483,632 bits. The sum of the bitstream lengths for all downstream devices must not exceed this number.

Guidelines and Design Considerations for Serial Daisy Chains

This section describes 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 device is operational when all DONE pins have been released.

Connect All DONE Pins

The DONE pins for all devices must be connected in a serial daisy chain to prevent configuration failure. 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. If additional time is required for the DONE signal to rise, the **`DonePipe`** option can be set for all devices in the serial daisy chain. (See UG628, *Command Line Tools User Guide* for settings.)

Ganged Serial Configuration

More than one device can be configured simultaneously from the same bitstream using a ganged serial configuration setup (Figure 9-2). In this arrangement, the serial configuration pins are tied together such that each device sees the same signal transitions. For ganged serial configuration, all devices must be identical.

Configuration can be driven by an external configuration controller, reading the bitstream from flash or other memory.

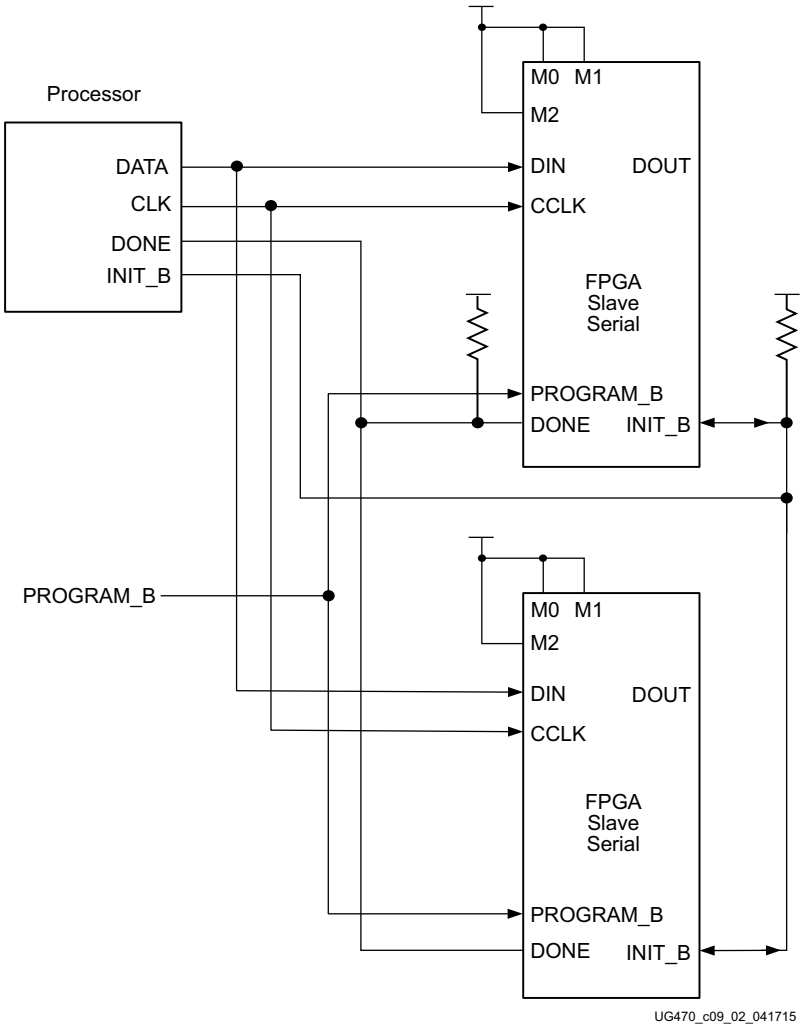


Figure 9-2: **Ganged Serial Configuration**

Notes relevant to Figure 9-2.

1. The DONE pin is an open-drain output.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. The startup clock setting must be set for CCLK for serial configuration.
4. For ganged serial configuration, all devices must be identical (same IDCODE) and must be configured with the same bitstream.

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.

- DONE Pins Can be Disconnected

  For debugging purposes, it is often helpful to have a way of disconnecting individual DONE pins from the common DONE signal. 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.

- CCLK as Clock Signal for Board Layout

  The CCLK signal is relatively slow, but the edge rates on the 7 series FPGA input buffers are very fast. Even minor signal integrity problems on the CCLK signal can cause the configuration to fail. The typical failure symptom is that DONE is still Low and INIT_B is High. Therefore, design practices that focus on signal integrity, including signal integrity simulation with IBIS, are 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 7 series devices in Slave SelectMAP mode can be connected on a common SelectMAP bus (Figure 9-3). In a SelectMAP bus, the DATA, CCLK, RDWR_B, 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 devices after configuration, the RDWR_B signal must be handled appropriately. (For details, refer to Chapter 6, Readback and Configuration Verification.) Otherwise, RDWR_B can be tied Low. Refer to Bitstream Loading (Steps 4-7) in Chapter 5 and to Chapter 6, Readback and Configuration Verification.

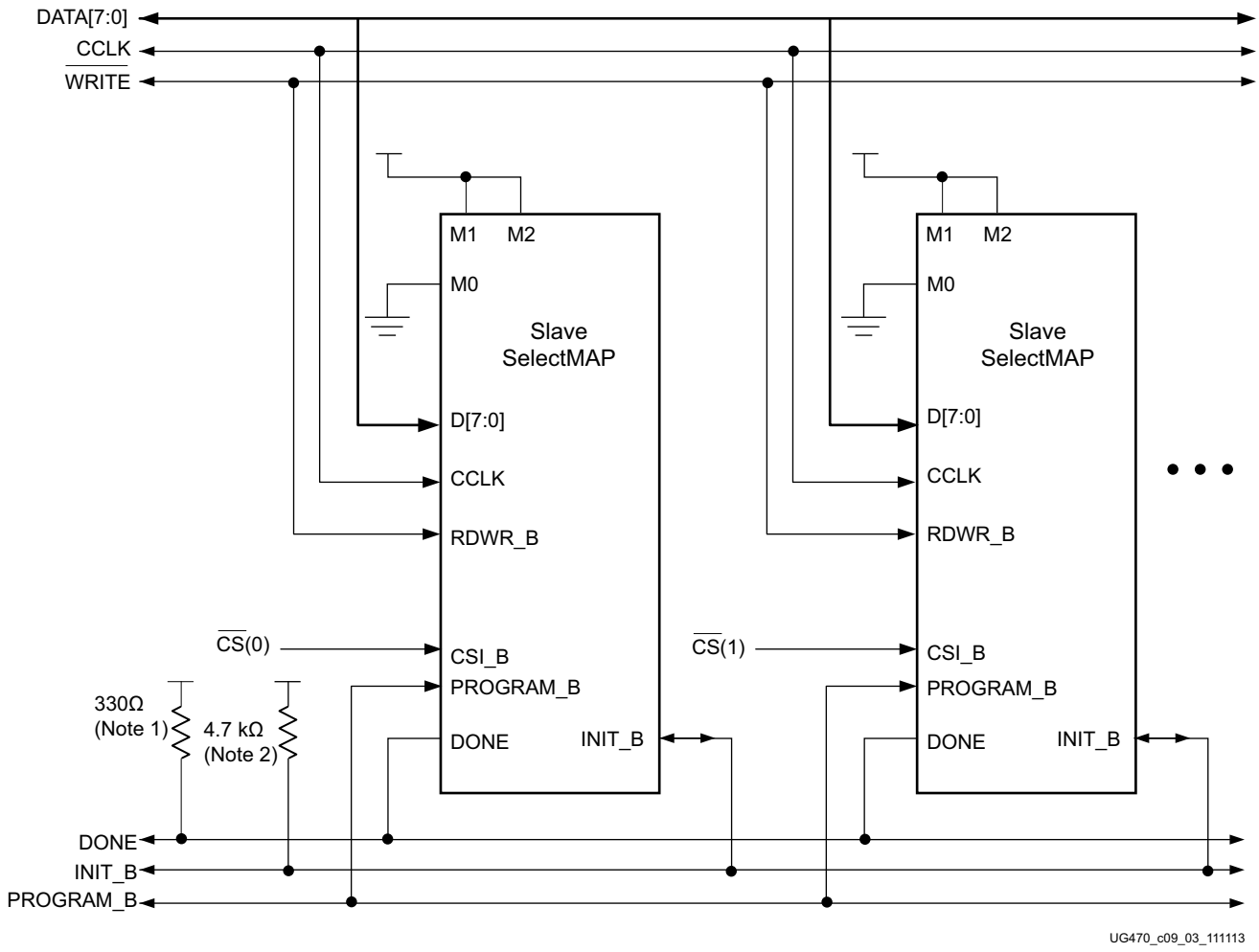


Figure 9-3: **Multiple Slave Device Configuration on an 8-Bit SelectMAP Bus**

Notes relevant to Figure 9-3.

1. The DONE pin is an open-drain output.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. Fallback reconfiguration is not supported.
4. The startup clock setting must be set for CCLK for SelectMAP configuration.
5. An external controller such as a microprocessor or CPLD is needed to control configuration.
6. The data bus can be x8, x16, or x32 (for Slave SelectMAP).

Parallel Daisy Chain Configuration

7 series FPGA configuration supports a parallel daisy-chain. Figure 9-4 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[15:00], 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.

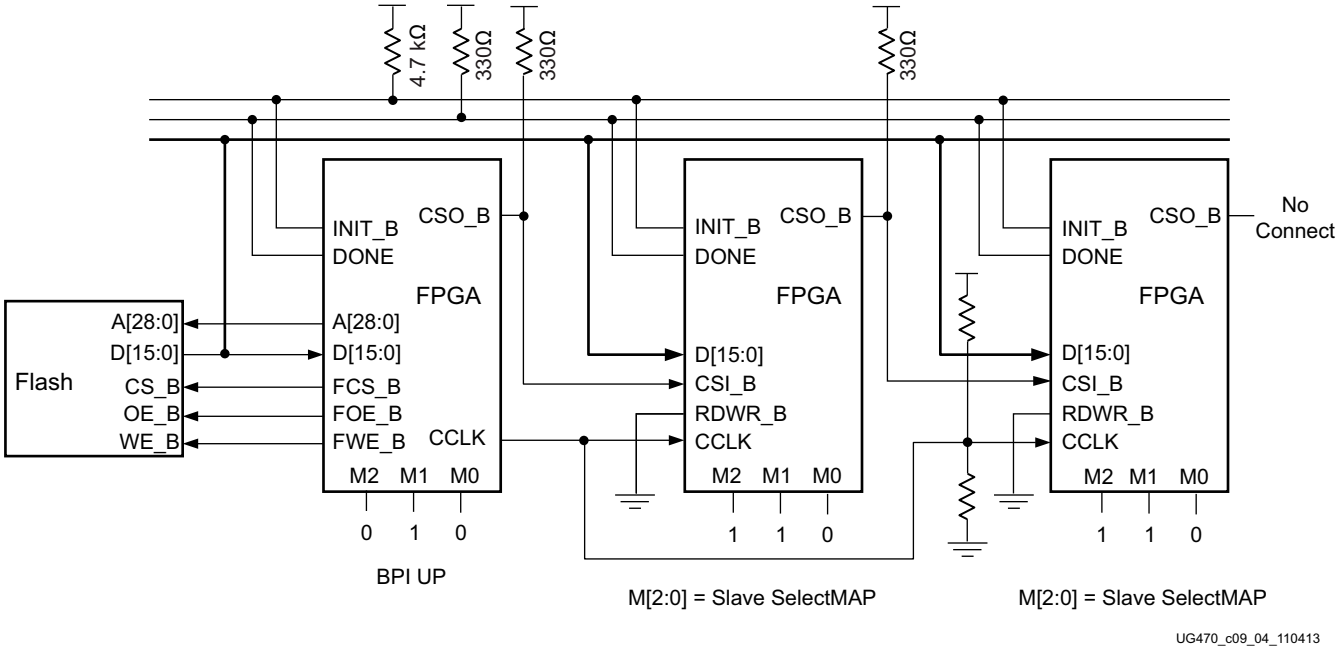


Figure 9-4: **Parallel Daisy Chain**

Notes relevant to Figure 9-4.

1. The DONE pin is an open-drain output.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up is required.
3. The startup clock setting must be set for CCLK for SelectMAP configuration.
4. 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 multi-function I/Os have weak pull-downs after configuration.
5. The first device in the chain can be Master SelectMAP, Slave SelectMAP, or BPI. The following devices must be in Slave SelectMAP mode.
6. Readback in the parallel daisy chain scheme is not supported.
7. Fallback reconfiguration is not supported.

Ganged SelectMAP Configuration

It is also possible to configure multiple devices simultaneously 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 GND), 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, as illustrated in Figure 9-5.

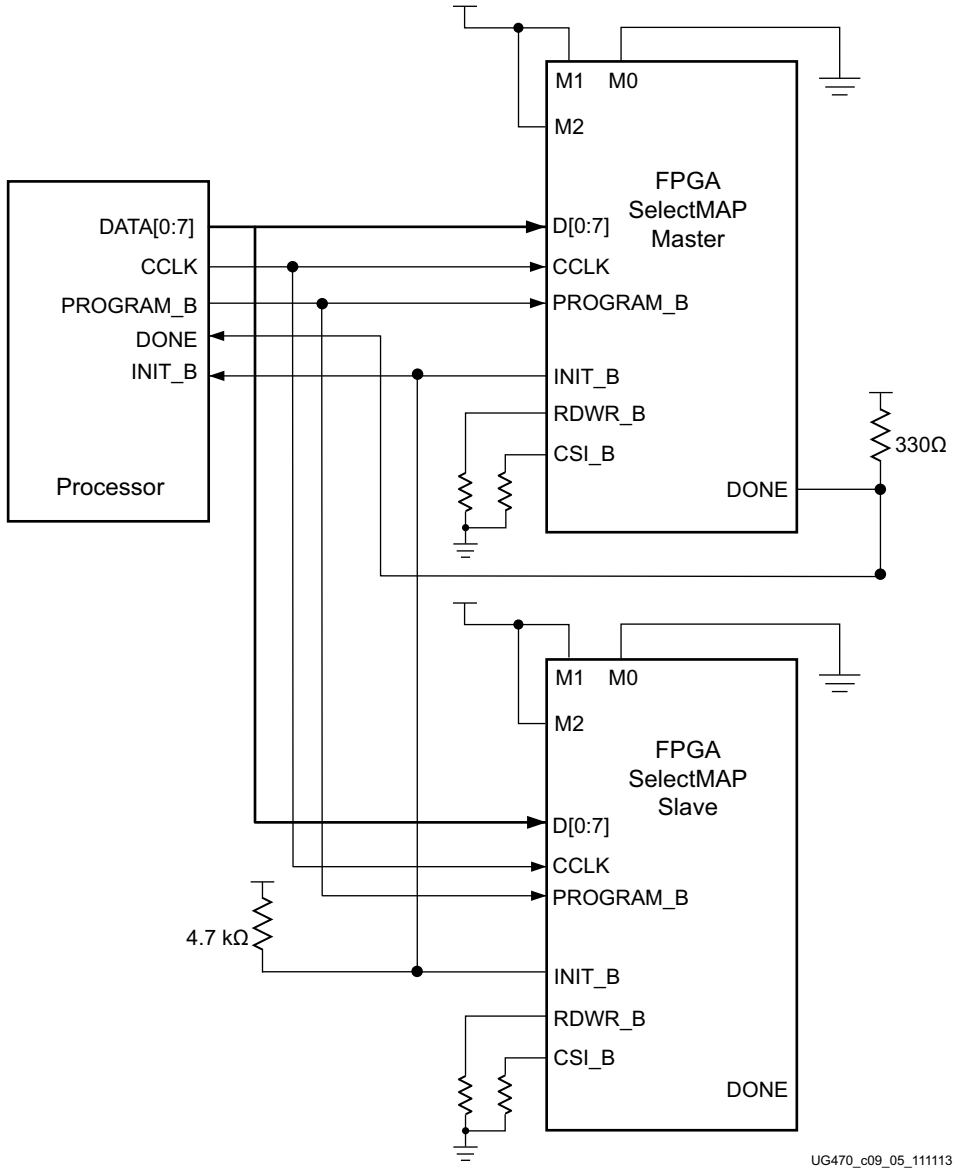


Figure 9-5: **Ganged x8 SelectMAP Configuration**

Notes relevant to Figure 9-5.

1. Synchronous SelectMAP mode is not supported.
2. Fallback reconfiguration is not supported.
3. The DONE pin is an open-drain output.
4. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
5. The startup clock setting must be set for CCLK for SelectMAP configuration.

An external pull-up resistor is required on the common DONE signal. Designers must carefully focus on signal integrity due to the increased fanout. 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.

XILINX

Chapter 10

Advanced JTAG Usage

Introduction

7 series devices support *IEEE Standard Test Access Port and Boundary-Scan Architecture* (IEEE Std 1149.1) and *IEEE Boundary-Scan Standard for Advanced Digital Networks* (IEEE 1149.6). The Joint Test Action Group (JTAG) is 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 boundary-scan description language (BSDL) compliance, PUDC_B should be set to 0. If PUDC_B is set to 1, then pre-configuration weak pull-up resistors are disabled.

JTAG Configuration/Readback

TAP Controller and Architecture

The FPGA TAP contains four mandatory dedicated pins as specified by the protocol given in Table 3-1 and illustrated in Figure 10-1, a typical JTAG architecture.

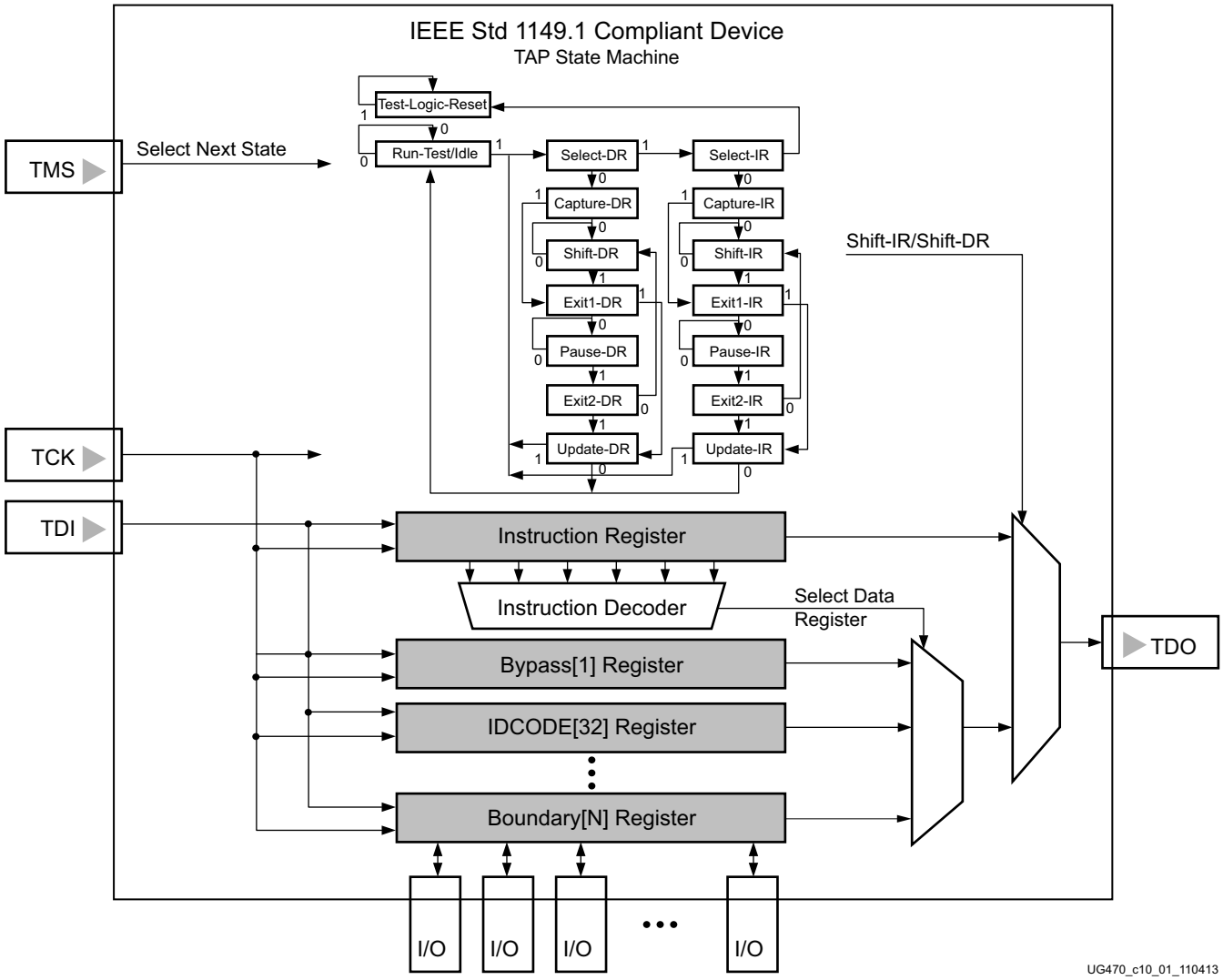


Figure 10-1: **Typical JTAG Architecture**

Figure 10-2 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 Data Path. 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 here.

- 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 Data Path 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.

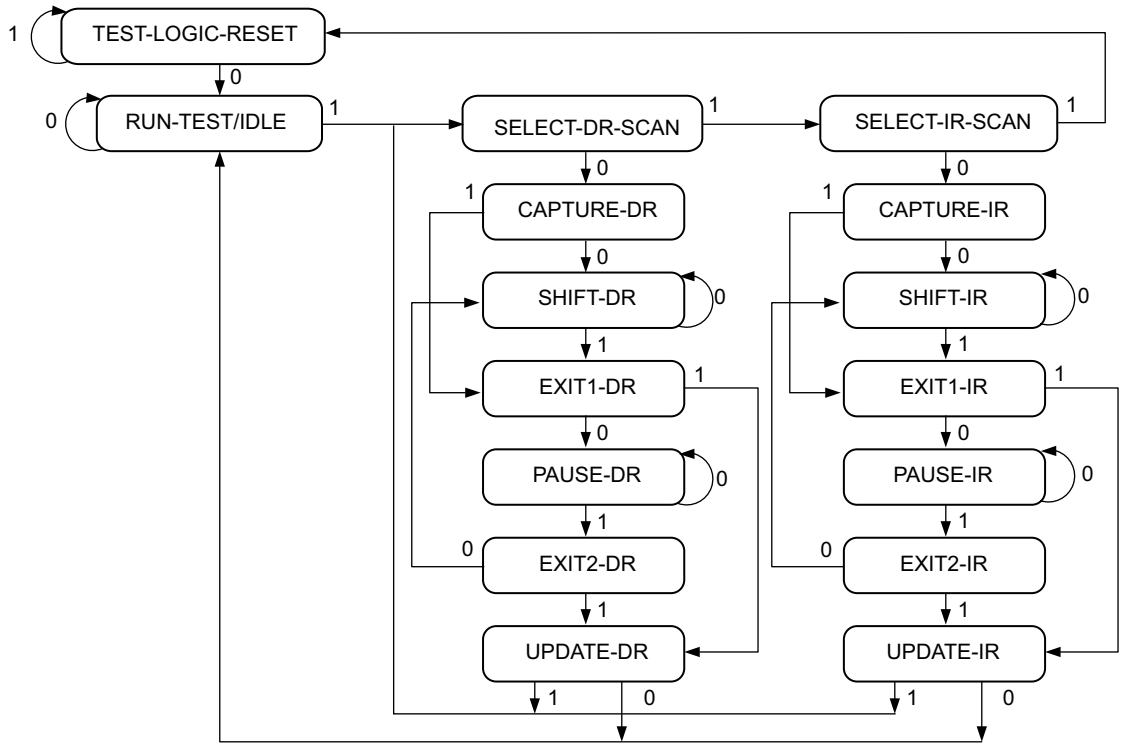


Figure 10-2: **Boundary-Scan TAP Controller**

7 series devices support the mandatory IEEE Std 1149.1 commands, as well as several Xilinx vendor-specific commands. The EXTEST, SAMPLE/PRELOAD, BYPASS, IDCODE, and USERCODE instructions are all included. The TAP also supports internal user-defined registers (USER1, USER2, USER3, and USER4) and configuration/readback of the device. INTEST is not supported. The HIGHZ_IO command is similar to the standard HIGHZ command but only disables the user I/O pins.

The FPGA boundary-scan operations are independent of mode selection. The boundary-scan mode overrides other mode selections. For this reason, boundary-scan instructions using the Boundary register (SAMPLE/PRELOAD and EXTEST) must not be performed during configuration. All instructions except the user-defined instructions are available before a device is configured. After configuration, all instructions are available.

JSTART and JSHUTDOWN are instructions specific to the FPGA device architecture and configuration flow. In 7 series 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. Clock 5 1's into TMS to guarantee that the state machine is in the TLR state. The TAP controller is reset on power up.

For details on the standard boundary-scan instructions EXTEST and BYPASS, refer to IEEE Std 1149.1.

Boundary-Scan Architecture

7 series 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 10-1).

Table 10-1: **7 Series 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	6 bits(1)	Holds the current instruction opcode and captures internal device status. Refer to Table 10-3.
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.

Notes:

1. The Instruction Register size increases from 22 to 38 bits in the devices based on SSI technology (7V2000T, 7VX1140T, 7VH580T, 7VH870T). See the BSDL files for device-specific information.

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 10-1).

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 for 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 PUDC_B pin determines whether the IOB has a pull-up resistor. Figure 10-3 represents of the 7 series FPGA boundary-scan architecture.

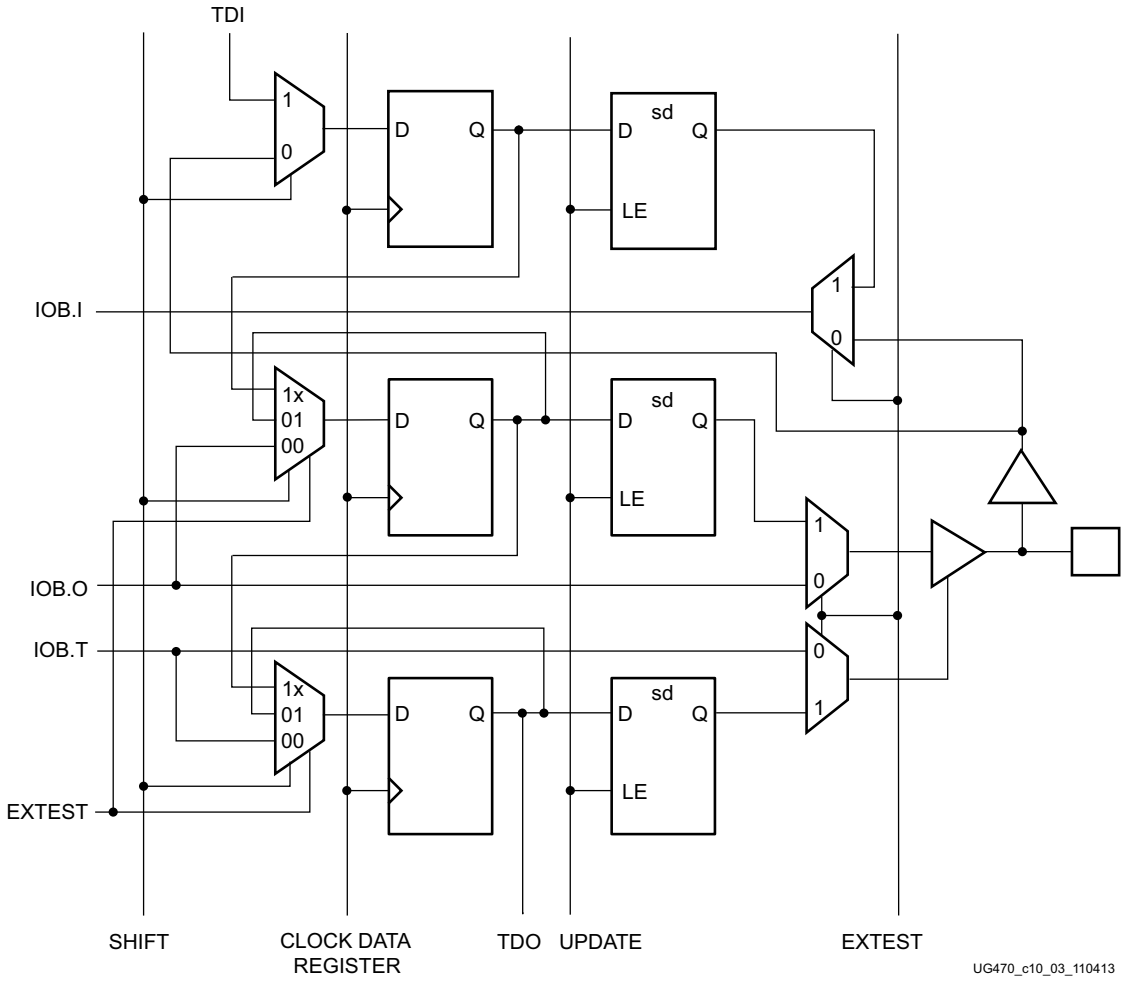


Figure 10-3: **7 Series FPGA Boundary-Scan Logic**

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 BSDL files for the 7 series family. (The BSDL files can be obtained from the Xilinx 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 ISE BSDLAnno utility to automatically modify the BSDL file for post-configuration interconnect testing. The BSDLAnno utility obtains the necessary FPGA design information from the implemented design, 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*. Similar functionality is available in the Vivado lab tools.

Instruction Register

The Instruction Register (IR) for the 7 series 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 7 series FPGA boundary-scan instruction set is loaded into the Instruction Register. The IR is 6 bits wide for monolithic 7 series devices. Table 10-2 lists the available instructions for 7 series devices. See JTAG Instructions in Chapter 5 for eFUSE-related JTAG instructions.

Table 10-2: **7 Series FPGA Boundary-Scan Instructions**

Boundary-Scan Command	Binary Code [5:0](1)	Description
EXTEST	100110	Enables boundary-scan EXTEST operation.
EXTEST_PULSE	111100	Enables boundary-scan EXTEST_PULSE operation for transceivers
EXTEST_TRAIN	111101	Enables boundary-scan EXTEST_TRAIN operation for transceivers
SAMPLE	000001	Enables boundary-scan SAMPLE operation.
USER1	000010	Access user-defined register 1.
USER2	000011	Access user-defined register 2.
USER3	100010	Access user-defined register 3.
USER4	100011	Access user-defined register 4.
CFG_OUT	000100	Access the configuration bus for readback.
CFG_IN	000101	Access the configuration bus for configuration.
USERCODE	001000	Enables shifting out user code.
IDCODE	001001	Enables shifting out of ID code.
HIGHZ_IO	001010	3-state I/O pins only, while enabling the Bypass register.
JPROGRAM	001011	Equivalent to and has the same effect as PROGRAM.
JSTART	001100	Clocks the startup sequence when StartClk is TCK.
JSHUTDOWN	001101	Clocks the shutdown sequence.
XADC_DRP	110111	XADC DRP access through JTAG. See the DRP interface section in UG480, *XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide* for the 7 series.
ISC_ENABLE	010000	Marks the beginning of ISC configuration. Full shutdown is executed.
ISC_PROGRAM	010001	Enables in-system programming.

Table 10-2: **7 Series FPGA Boundary-Scan Instructions *(Cont'd)***

Boundary-Scan Command	Binary Code [5:0](1)	Description
XSC_PROGRAM_KEY	010010	Change security status from secure to non-secure mode and vice versa.
XSC_DNA	010111	Read 57-bit Device DNA value.
FUSE_DNA	110010	Read 64-bit Device DNA value.
ISC_NOOP	010100	No operation.
ISC_DISABLE	010110	Completes ISC configuration. Startup sequence is executed.
BYPASS	111111	Enables BYPASS.
RESERVED	All other codes	Xilinx reserved instructions.

Notes:

1. Instruction register is larger for devices based on SSI technology (7V2000T, 7VX1140T, 7VH580T, 7VH870T). See the BSDL files for device-specific information.

Table 10-3 shows the instruction capture values loaded into the IR as part of an instruction scan sequence.

Table 10-3: **7 Series FPGA Instruction Capture Values Loaded into IR as Part of an Instruction Scan Sequence**

TDI	IR[5]	IR[4]	IR[3]	IR[2]	IR[1:0]	→ TDO
	DONE	INIT(1)	ISC_ENABLED	ISC_DONE	0 1	

Notes:

1. INIT is the status bit of the INIT_COMPLETE signal.
2. Instruction register is larger for devices based on SSI technology (7V2000T, 7VX1140T, 7VH580T, 7VH870T). See the BSDL files for device-specific information.

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

7 series 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 7 series 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 (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 BSCANE2 library primitive is required when creating these registers.

BSCANE2 Primitive

The BSCANE2 primitive allows access between the internal FPGA logic and the JTAG Boundary Scan logic controller. This allows for communication between the internal running design and the dedicated JTAG pins of the FPGA, the Test Access Port (TAP). The BSCANE2 primitive must be instantiated to gain internal access to the JTAG pins. The BSCANE2 primitive is not needed for normal JTAG operations that use direct access from the JTAG pins to the TAP controller. The BSCANE2 is automatically added to a design when using the Vivado Logic Analyzer. Figure 10-4 shows the BSCANE2 Primitive.

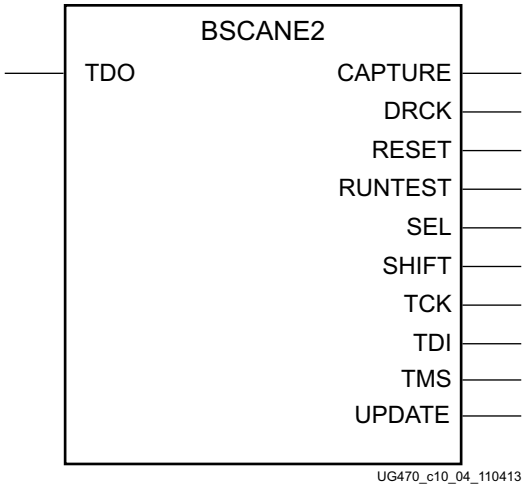


Figure 10-4: BSCANE2 Primitive

A typical user application requiring instantiation of the BSCANE2 is to create internal, private scan registers in the FPGA logic. These scan registers propagate through the FPGA logic, not through the boundary I/O as is true with standard JTAG boundary scan. Each instance of this primitive will support one JTAG USER instruction, with multiple instantiations differentiated with the JTAG_CHAIN attribute. To handle all four USER instructions (USER1 through USER4), instantiate four BSCANE2 primitives and set the JTAG_CHAIN attribute uniquely on each.

For devices based on SSI technology, the BSCANE2 can only be instantiated in the Master SLR. The tools will automatically place the element in the correct SLR. Only the JTAG port on the master SLR can be accessed by the BSCANE2 primitive.

The BSCANE2 primitive can also be used to control or monitor activity on the JTAG TAP port. A signal on the TDO input of the primitive passes through an output timing register, where the TDO input to the primitive is registered on the falling edge of TCK as it is passed to the external TDO output pin when a USER instruction is active. The associated primitive's SEL output goes High to indicate which USER1-USER4 instruction is active. The DRCK output provides access to the data register clock generated by the TAP controller. The RESET, UPDATE, SHIFT, and CAPTURE pins represent the decoding of the corresponding state of the boundary scan internal state machine. The TDI port provides access from the external TDI pin of the JTAG TAP in order to shift data into an internal scan chain. The TCK and TMS pins are similarly monitored through the BSCANE2 primitive.

For information on the BSCANE2 simulation model, see UG626, *Synthesis and Simulation Design Guide*, Chapter 6, *Simulating Your Design*.

Using Boundary-Scan in 7 Series Devices

Characterization data for some of the most commonly requested timing parameters shown in Figure 10-5 are listed in the Configuration Switching Characteristics table in the respective 7 series data sheet on xilinx.com.

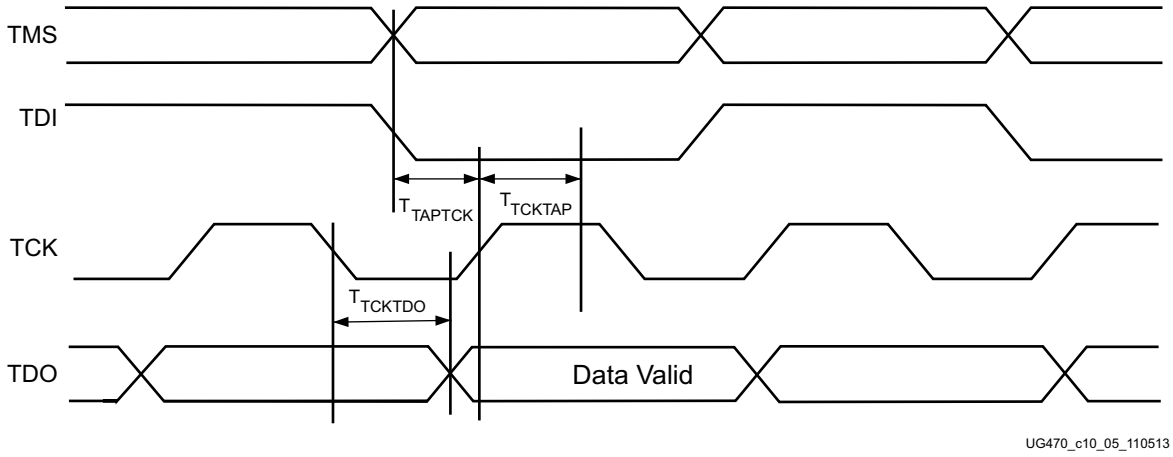


Figure 10-5: **Boundary-Scan Port Timing Waveforms**

For further information on the startup sequence, bitstream, and internal configuration registers referenced here, refer to appropriate configuration sequence in Chapter 9.

Configuring through Boundary-Scan

One of the most common boundary-scan vendor-specific instructions is the Configure instruction. If the 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 FPGA configuration with JTAG is shown in Figure 10-6. The sections that follow describe how the 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 shutdown sequence (see Figure 10-6)

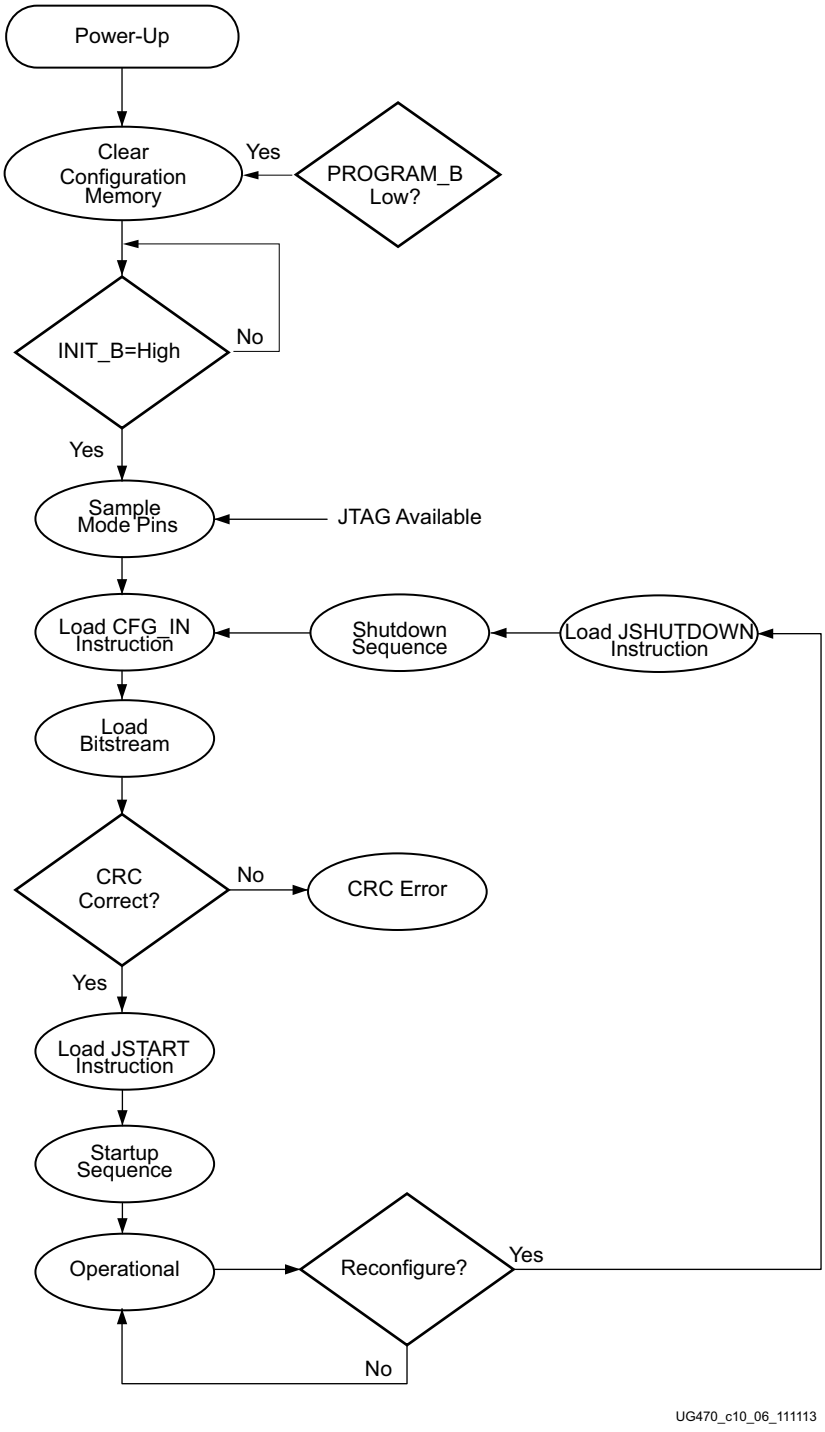


Figure 10-6: **Device Configuration Flow Diagram**

Single Device Configuration

Table 10-4 describes the TAP controller commands required to configure a 7 series device. Refer to Figure 10-2 for TAP controller states. These TAP controller commands are issued automatically if configuring the part with the ISE iMPACT tools or the Vivado device programmer.

Table 10-4: **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:	01011(4)	0	5
6.	Load the MSB of the JPROGRAM instruction when exiting SHIFT-IR, as defined in the IEEE standard.	0	1	1
7.	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
8.	Move into the RTI state.	X	0	10,000(1)
9.	Start loading the CFG_IN instruction, LSB first:	00101	0	5
10.	Load the MSB of CFG_IN instruction when exiting SHIFT-IR, as defined in the IEEE standard.	0	1	1
11.	Enter the SELECT-DR state.	X	1	2
12.	Enter the SHIFT-DR state.	X	0	2
13.	Shift in the FPGA bitstream. Bit_n (MSB) is the first bit in the bitstream(2).	bit_1 ... bit_n	0	(bits in bitstream)-1
14.	Shift in the last bit of the bitstream. Bit0 (LSB) shifts on the transition to EXIT1-DR.	bit_0	1	1
15.	Enter UPDATE-DR state.	X	1	1
16.	Move into RTI state.	X	0	1
17.	Enter the SELECT-IR state.	X	1	2
18.	Move to the SHIFT-IR state.	X	0	2
19.	Start loading the JSTART instruction (optional). The JSTART instruction initializes the startup sequence.	01100	0	5
20.	Load the last bit of the JSTART instruction.	0	1	1
21.	Move to the UPDATE-IR state.	X	1	1

Table 10-4: **Single Device Configuration Sequence *(Cont'd)***

TAP Controller Step and Description		Set and Hold		# of Clocks
		TDI	TMS	TCK
22.	Move to the RTI state and clock the startup sequence by applying a minimum of 2000 clock cycles to the TCK.	X	0	2000
23.	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 need to be reconfigured 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.
4. Instruction register is larger for devices based on SSI technology (7V2000T, 7VX1140T, 7VH580T, 7VH870T). See the BSDL files for device-specific information.

Multiple Device Configuration

It is possible to configure multiple 7 series devices in a chain. (See Figure 10-7) 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 10-2 for these 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 10-4.
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.

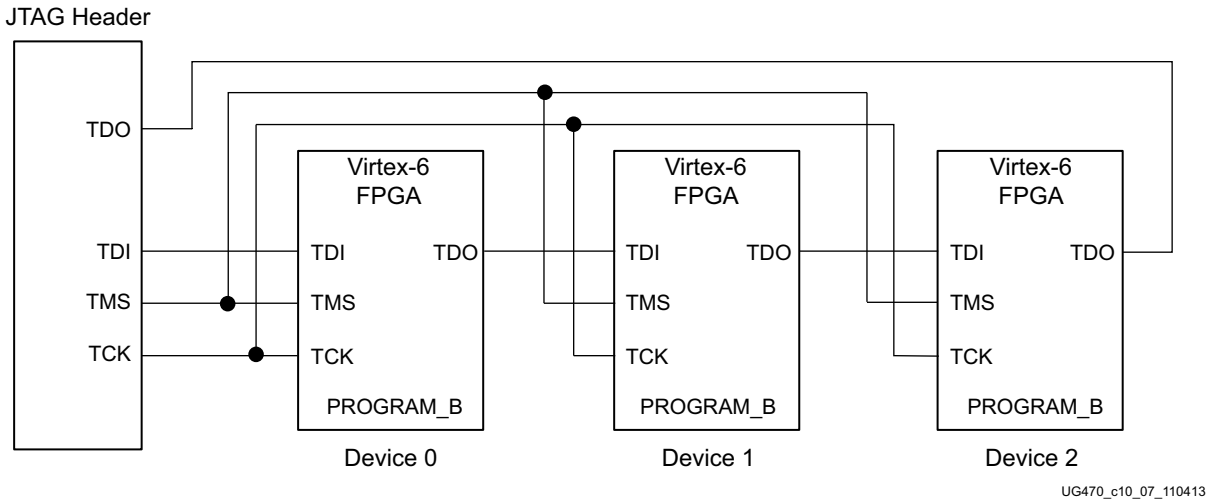


Figure 10-7: **Boundary-Scan Chain of Devices**