UltraScale Architecture Configuration

User Guide

UG570 (v1.13) July 28, 2020

XILINX

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/28/2020	1.13	Added VU57P device to Table 1-3, added KU19P, VU23P, and VU57P devices to Table 1-4 and Table 1-5. Updated Table 8-4 descriptions for R_DIS_USER, R_DIS_SEC, and R_DIS_RSA. Updated Table 9-15 description for configuration word 30026001. Added information for encrypted bitstreams using an obfuscated key with the JTAG, SelectMAP, or ICAP interfaces.
03/31/2020	1.12	Added VU19P production IDCODE revision to Table 1-5. Updated value for GLUTMASK_B in Table 9-23.
09/30/2019	1.11	Added coverage of Virtex UltraScale+ VU19P, VU47P, and VU49P devices. Updated production IDCODE revisions for Virtex UltraScale+ FPGAs in Table 1-5. Updated descriptions for INIT_B and DONE in Table 1-9. Updated Figure 9-4 and Figure 9-13. Added information to SEU Detection and Correction in Chapter 10.
02/21/2019	1.10	Added coverage of Virtex UltraScale+ VU27P and VU29P devices. Clarified description of DONE pin in Table 1-9. Added section FRAME_ECCE4 in Chapter 7. Clarified Loading the Encryption Key in Chapter 8. Clarified description of R_DIS_KEY in Table 8-4. Changed “24” to “64” in Clocking to End of Start-up in Chapter 9. Clarified first paragraph under Serial Daisy Chain Configuration in Chapter 12.
04/09/2018	1.9	Added VU35P and VU37P devices to Table 1-3. Added VU31P, VU33P, VU35P, and VU37P devices to Table 1-4 and Table 1-5. Changed VREF to VCCO_0 in Figure 2-4 and Figure 2-5. Clarified Persist Option in Chapter 10.
12/21/2017	1.8	Clarified AES-CBC terminology throughout document. Updated Table 1-5. Added information to note 6 under Table 1-7 and Table 1-8. Clarified the minimum recommended operating voltage in second paragraph under Configuration Banks Voltage Select (Kintex UltraScale and Virtex UltraScale FPGAs) in Chapter 1. Updated External Master Configuration Clock (EMCCLK) Option in Chapter 1. Clarified bitstream generation under Master SPI Dual Quad (x8) in Chapter 2. Added note under Master BPI Synchronous Read in Chapter 4. Clarified ABORT sequence support under RDWR_B in Chapter 5. Added SelectMAP ABORT in Chapter 5. Added last paragraph under Applications in Chapter 7. Added UltraScale+ FPGAs column to Table 8-1. Updated Table 8-2 and Table 8-4. Added last paragraph under Device Power-Up (Step 1) and second paragraph under CRC Check (Step 7) in Chapter 9. Updated descriptions of RBCRC_ACTION in Table 9-30 and TIMER_CFG_MON in Table 9-34. Clarified first paragraph and step 10 under Configuration Memory Read Procedure (SelectMAP) in Chapter 10.

Date	Version	Revision
03/15/2017	1.7	Clarified differences between UltraScale and UltraScale+ device families throughout document. Added UltraScale+ devices to Table 1-3. Added UltraScale+ device bitstream lengths to Table 1-4 and JTAG and IDCODEs Table 1-5. Added important note to Recommended Design Flow and Configuration Factors in Chapter 1. Removed input arrow heads from SPI Flash DQ2 and DQ3 in Figure 2-4 and Figure 2-5. Added Additional Information in Chapter 2. Changed VCCO_0 to VCCO_65 in Figure 3-2. Deleted reference to specific Parallel NOR flash device in Figure 4-2, Figure 4-4, and text descriptions. Change "byte-swapped" to "bit-swapped" under File Generation in Chapter 4. Replaced fourth paragraph under Introduction in Chapter 6. Clarified description of MASTER_JTAG in Chapter 7. Added last paragraph under Bitstream Encryption and Authentication and last paragraph under Loading the Encryption Key in Chapter 8. Clarified second paragraph under eFUSE in Chapter 8. Corrected "0:255" to "255:0" in second row of Table 8-2. Updated bit positions 6 and 7 Table 8-5. Updated format of the FPGA JTAG IDCODE register under Device ID Check (Step 5) in Chapter 9. Changed "03651093" to "03822093" in Table 9-15. Updated Table 9-30. Added last sentence under first paragraph under Configuration Memory Read Procedure (SelectMAP) in Chapter 10. Updated values in step 8 and step 10 of Table 10-2. Changed "Readback CRC" to SEU Detection and Correction in Chapter 10 (section title). Added references to PG172, XAPP1191, XAPP1280, XAPP1267, XAPP1261, and XAPP1257 throughout document and in Appendix A, Additional Resources and Legal Notices.
12/16/2015	1.6	Updated Table 1-4 and Table 1-5. Added last sentence to first paragraph under MASTER_JTAG in Chapter 7. Added second paragraph and Table 8-1 under RSA Authentication to define restrictions as noted in XCN15038. Added last sentence to first paragraph under OTP eFUSE Registers in Chapter 8. Added second sentence to paragraph following Table 8-3.
11/24/2015	1.5	Added UltraScale+ device information. Added data to Table 1-5. Updated description of PROGRAM_B PUDC_B in Table 1-9. Clarified paragraph following Figure 9-4. Clarified first sentence under Delaying Configuration in Chapter 9. Added PROGRAM_B to Table 9-6. Clarified second paragraph under Fallback MultiBoot and first paragraph under IPROG in Chapter 11.
09/15/2015	1.4	Added configuration information for the KU025 device. Changed "SPI flash" to "Serial NOR flash". Added bitstream property names. Added KU025 Differences to Chapter 1. Updated device resources and bitstream lengths (Table 1-3, Table 1-4, and Table 1-5). Added Design Tools to Chapter 1. Added STARTUPE3 Connections to Dedicated Pins and Timing Considerations for Flash Connections to Chapter 7. Added BSPI_READ and FALL_Edge commands to Table 9-21. Added bitstream properties to Table 9-27 and Table 9-29. Added Readback Capture to Chapter 10. Added Ganged Asynchronous BPI Configuration to Chapter 12. Made minor clarifications throughout document.
03/31/2015	1.3	Added configuration information for Kintex UltraScale devices XCKU085 and XCKU095. Modified Table 1-4 and added Table 1-5. Clarified recommended external pull-up/down resistor values for pins M[2:0], PROGRAM_B, INIT_B, DONE, PUDC_B, BPI (FCS_B), and BPI (ADV_B) in Table 1-8. Clarified descriptions for CFGBVS in Table 1-9. Deleted Note 3 from Figure 2-2, Figure 2-4, Figure 3-2, Figure 4-2, Figure 4-4, Figure 5-2, and Figure 12-1 through Figure 12-5. Changed VCCO_0 resistor value from 330Ω to 4.7 kΩ in Figure 2-2, Figure 2-4, Figure 2-5, Figure 3-2, Figure 4-2, Figure 4-4, Figure 5-2, Figure 6-7, Figure 12-3, Figure 12-4, and Figure 12-5. Clarified USR_ACCESSE2 Advanced Uses and added Figure 7-9. Added Note to Files for Serial Daisy Chains. Updated word counts in Table 9-15. Changed bit 23 from "Reserved" to "CAPTURE" in Table 9-36. Added Table 9-37 and Table 9-39. Updated word counts in Table 10-2.

Date	Version	Revision
02/20/2015	1.2	Added reference to PG156 under Differences from Previous Generations. Added "Production IDCODE Revision" column to Table 1-4. Expanded Note 2 in Table 1-6. Added Persist Option and reference to Configuration Pins. Made minor clarifications to descriptions in Table 1-9. Added last sentence in fourth paragraph under Configuration Banks Voltage Select. Modified Note 2 and added Note 3 to Table 1-11. Deleted clock source specification from External Master Configuration Clock (EMCCLK) Option and referenced the specification. Changed VCCO_65 destination from VCCO_0 to VCCO_65 in Figure 2-2 and Figure 2-4. Added second sentence following Figure 4-3 and last sentence under Synchronous Read Sequence. Updated Figure 4-4. Made minor clarifications to notes relevant to Figure 4-2, Figure 4-3, Figure 6-7, and Figure 12-1 through Figure 12-5. Clarified second paragraph under Shift-Dr, Exit1-DR, Pause-DR, Exit2-DR, and Update-DR. Made minor clarifications to Table 6-3. Clarified first paragraph under Using Boundary-Scan Configuration in UltraScale FPGAs. Clarified ICAPE3 Resources. Added Rolling Keys section to Chapter 8. Added "Important" note to Loading Encrypted Bitstreams. Added last sentence under RSA Authentication. Clarified descriptions in Table 8-2. Added "RMA Impact" column to Table 8-5. Clarified Figure 8-2. Clarified JTAG Access to Device Identifier. Clarified Clear Configuration Memory (Step 2, Initialization). Added I/O Transition at the End of Startup section to Chapter 9. Expanded description for RS_TS_B in Table 9-30. Clarified BPI – Hardware RS Pin Design Considerations. Changed "PROGRAM_B" to "INIT_B" in Figure 12-2.
11/10/2014	1.1	Updated Table 1-4 and added Table 1-6. Added last paragraph under A High-Speed Configuration Option. Updated pin descriptions in Table 1-9. Clarified External Master Configuration Clock (EMCCLK) Option in Chapter 1. Updated Table 6-5. Updated pin descriptions in Table 7-10. Updated eFUSE in Chapter 8 and added Table 8-4 and Table 8-5. Added Chapter 9 through Chapter 13.
12/10/2013	1.0	Initial Xilinx release.

Table of Contents

Revision History 2

Chapter 1: Introduction

Introduction to the UltraScale Architecture 9
Overview 10
Differences Between UltraScale FPGA Families 11
Differences from Previous Generations 12
Design Considerations 16
Recommended Design Flow and Configuration Factors 24
Design Tools 25
Pinout Planning 26
Configuration Interfaces 28
Configuration Pins 28
Configuration Banks Voltage Select (Kintex UltraScale and Virtex UltraScale FPGAs) 39
Power-On Reset 42
External Master Configuration Clock (EMCCLK) Option 44
VBATT 45

Chapter 2: Master SPI Configuration Mode

Introduction 46
Master SPI Interface 46
Master SPI Quad (x4) 51
Master SPI Dual Quad (x8) 52
Serial NOR Flash Densities over 128 Mb 55
Multi-die Serial NOR Flash Devices 55
SPI Configuration Timing 55
Determining the Maximum Configuration Clock Frequency 56
Power-on Sequence Precautions 57
Additional Information 57

Chapter 3: Serial Configuration Mode

Introduction 58
Slave Serial Configuration 58

Master Serial Configuration 60
Clocking Serial Configuration Data 61

Chapter 4: Master BPI Configuration Mode

Introduction 62
Master BPI Interface 62
Master BPI Synchronous Read 64
Master BPI Asynchronous Read 68
Configuration Time 72
Power-on Sequence Precautions 74
File Generation 75
Parallel NOR Flash Programming Options 76

Chapter 5: SelectMAP Configuration Modes

Introduction 78
SelectMAP Configuration Interface 78
Single Device SelectMAP Configuration 79
SelectMAP Data Loading 81
SelectMAP ABORT 86

Chapter 6: Boundary-Scan and JTAG Configuration

Introduction 89
Boundary-Scan Using IEEE Standard 1149.1 90
Using Boundary Scan in Xilinx Devices 91
Boundary-Scan Design Considerations 92
TAP Controller and Architecture 94

Chapter 7: Design Entry

Introduction 107
BSCANE2 108
DNA_PORTE2 112
EFUSE_USR 113
FRAME_ECCE3 114
FRAME_ECCE4 115
ICAPE3 115
MASTER_JTAG 117
STARTUPE3 118
USR_ACCESSE2 123

Chapter 8: Bitstream Security, eFUSEs, and Device DNA

Introduction . . . 126
Readback Security . . . 126
Bitstream Encryption and Authentication . . . 127
eFUSE . . . 133
Device Identifier (Device DNA) . . . 137

Chapter 9: Configuration Details

Introduction . . . 141
Configuration Data File Formats . . . 141
Configuration Sequence . . . 145
Configuration Bitstream . . . 158
Configuration Registers . . . 164

Chapter 10: Readback Verification and CRC

Introduction . . . 180
Persist Option . . . 181
Readback Command Sequences . . . 181
Verifying Readback Data . . . 190
SEU Detection and Correction . . . 193
Readback Capture . . . 194

Chapter 11: MultiBoot and Reconfiguration

Introduction . . . 196
Fallback MultiBoot . . . 196
IPROG Reconfiguration . . . 203
Status Register for Fallback and IPROG Reconfiguration . . . 206
Dynamic Reconfiguration of Functional Blocks . . . 207

Chapter 12: Configuring Multiple FPGAs

Introduction . . . 208
Serial Configuration Modes . . . 208
Parallel Configuration Modes . . . 213

Chapter 13: Configuration Debugging

Introduction . . . 217
File Generation Review . . . 217
Status Pin Handling . . . 218
Status Register Use and JTAG Access . . . 219
Verification and Readback . . . 219
Configuration Sequence . . . 219
Initial Debug Steps . . . 220

Appendix A: Additional Resources and Legal Notices

Xilinx Resources 222
Solution Centers 222
Documentation Navigator and Design Hubs 222
References 223
Please Read: Important Legal Notices 224

Introduction

Introduction to the UltraScale Architecture

The Xilinx® UltraScale™ architecture is the first ASIC-class programmable architecture to enable multi-hundred gigabit-per-second levels of system performance with smart processing, while efficiently routing and processing data on-chip. UltraScale architecture-based devices address a vast spectrum of high-bandwidth, high-utilization system requirements by using industry-leading technical innovations, including next-generation routing, ASIC-like clocking, 3D-on-3D ICs, multiprocessor SoC (MPSoC) technologies, and new power reduction features. The devices share many building blocks, providing scalability across process nodes and product families to leverage system-level investment across platforms.

Virtex® UltraScale+™ devices provide the highest performance and integration capabilities in a FinFET node, including both the highest serial I/O and signal processing bandwidth, as well as the highest on-chip memory density. As the industry's most capable FPGA family, the Virtex UltraScale+ devices are ideal for applications including 1+Tb/s networking and data center and fully integrated radar/early-warning systems.

Virtex UltraScale devices provide the greatest performance and integration at 20 nm, including serial I/O bandwidth and logic capacity. As the industry's only high-end FPGA at the 20 nm process node, this family is ideal for applications including 400G networking, large scale ASIC prototyping, and emulation.

Kintex® UltraScale+ devices provide the best price/performance/watt balance in a FinFET node, delivering the most cost-effective solution for high-end capabilities, including transceiver and memory interface line rates as well as 100G connectivity cores. Our newest mid-range family is ideal for both packet processing and DSP-intensive functions and is well suited for applications including wireless MIMO technology, Nx100G networking, and data center.

Kintex UltraScale devices provide the best price/performance/watt at 20 nm and include the highest signal processing bandwidth in a mid-range device, next-generation transceivers, and low-cost packaging for an optimum blend of capability and cost-effectiveness. The family is ideal for packet processing in 100G networking and data centers applications as well as DSP-intensive processing needed in next-generation medical imaging, 8k4k video, and heterogeneous wireless infrastructure.

Zynq® UltraScale+ MPSoC devices provide 64-bit processor scalability while combining real-time control with soft and hard engines for graphics, video, waveform, and packet processing. Integrating an Arm®-based system for advanced analytics and on-chip programmable logic for task acceleration creates unlimited possibilities for applications including 5G Wireless, next generation ADAS, and Industrial Internet-of-Things.

This user guide describes the UltraScale architecture-based FPGAs configuration and is part of the UltraScale architecture documentation suite available at: www.xilinx.com/documentation.

Overview

This chapter provides a brief overview of the configuration methods and features for the UltraScale architecture-based FPGAs. Subsequent chapters provide more detailed descriptions of each configuration method and feature.

Xilinx FPGAs are highly flexible, reprogrammable logic devices. Like processors, Xilinx FPGAs are fully user programmable. For FPGAs, the program is called a bitstream, which defines the application-specific FPGA functionality. The bitstream loads into the FPGA internal memory at system power-up or on demand by the system.

Like processors and processor peripherals, Xilinx FPGAs can be reprogrammed, in system, on demand, an unlimited number of times. After programming, the FPGA bitstream is stored in highly robust CMOS configuration latches (CCLs). Although CCLs are reprogrammable like SRAM memory, CCLs are designed primarily for data integrity. Because the Xilinx FPGA bitstream is stored in CCLs, the device must be reconfigured after it is power cycled.

The process whereby the defining data is loaded or programmed into the FPGA is called configuration. Configuration is designed to be flexible to accommodate different application needs and, wherever possible, to leverage existing system resources to minimize system costs.

Similar to processors, Xilinx FPGAs optionally load or boot themselves automatically from an external nonvolatile memory device. Alternatively, similar to processor peripherals, Xilinx FPGAs can be downloaded or programmed by an external device, such as a microprocessor, DSP processor, microcontroller, PC, or board tester. The configuration datapath can be serial to minimize pin requirements, including configuration through the industry-standard IEEE 1149.1 JTAG boundary scan interface. A parallel configuration datapath provides maximum performance and access to industry-standard interfaces, ideal for external data sources like processors, or x8- or x16-parallel flash memory.

The configuration bitstream is loaded into the FPGA through special configuration pins. These configuration pins serve as the interface for a number of different configuration modes:

- Slave serial
- Slave SelectMAP (parallel) (x8, x16, and x32)
- JTAG boundary scan
- Master SPI (serial peripheral interface) (serial NOR flash x1, x2, x4, and dual x4, effectively x8)
- Master BPI (byte peripheral interface) (parallel NOR flash x8 and x16)
- Master serial
- Master SelectMAP (parallel) (x8 and x16)

The terms master and slave refer to the direction of the configuration clock (CCLK):

- In master configuration modes, the FPGA drives CCLK from an internal oscillator. Configuration options are used to select the desired frequency. After configuration, the CCLK is turned off by default, and the CCLK pin is 3-stated with a weak pull-up.
- In slave configuration modes, CCLK is an input.

The specific configuration mode is selected by setting the appropriate level on the 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 JTAG (boundary scan) configuration interface is always available, regardless of the mode pin settings.

The configuration modes are explained in detail in Chapter 2 through Chapter 5.

The FPGA can also control its own configuration through internal connections from the FPGA logic to the configuration logic. The device can be either fully reprogrammed with an alternative design it has selected, or partial reconfiguration allows specific regions of the FPGA to be reprogrammed with new functionality while applications continue to run in the remainder of the device.

Differences Between UltraScale FPGA Families

This document uses the Kintex UltraScale and Virtex UltraScale families as the basis for descriptions and examples. The following defines some of the differences in the Kintex UltraScale+ and Virtex UltraScale+ families:

- Master serial and master SelectMAP configuration modes are not supported in the UltraScale+ FPGAs. These modes are not recommended in the other UltraScale families.
- The configuration interface can operate only at 1.8V or 1.5V in the UltraScale+ FPGAs. There is no CFGBVS pin in UltraScale+ devices. When migrating from an UltraScale

FPGA to an UltraScale+ FPGA, the CFGBVS pin location becomes RSVDGND and must be connected to GND.

- The configuration timing and configuration rate options are different between UltraScale FPGAs and UltraScale+ FPGAs. The configuration frame size is 93 32-bit words in the UltraScale+ FPGAs and 123 32-bit words in the UltraScale FPGAs.

Differences from Previous Generations

UltraScale architecture-based FPGAs support similar configuration interfaces as the 7 series FPGAs, with most improvements targeted at improving configuration performance. Table 1-1 summarizes the key differences in available configuration modes.

Table 1-1: **Configuration Modes in UltraScale Architecture-based FPGAs Compared to 7 Series FPGAs**

Configuration Mode	UltraScale Architecture	7 Series
Slave serial mode	Yes	Yes
Slave SelectMAP mode	Yes	Yes
JTAG mode	Yes	Yes
Master SPI mode (x1, x2, x4)	Yes	Yes
Master SPI mode (dual quad, x8)	Yes	No
Master BPI mode	Yes	Yes
Master serial mode	Not recommended(1)	Yes
Master SelectMAP mode	Not recommended(1)	Yes

Notes:

1. The master SPI and BPI configuration modes are recommended over the legacy master serial and master SelectMAP modes because they provide a wider flash density selection and lower cost solution. See Differences Between UltraScale FPGA Families, page 11.

The master configuration modes are optimized to work with standard third-party flash memories. The SPI mode interfaces to standard x1, x2, or x4 serial NOR flash memories, while the BPI mode interfaces to x8 or x16 parallel NOR flash memories.

TIP: *The master serial and master SelectMAP configuration modes are supported but not needed for most applications. The 7 series FPGAs supported master serial mode for configuration from legacy serial PROMs or for custom, CPLD-based configuration state machines driven by the FPGA CCLK. The master SelectMAP mode has been superseded by the BPI configuration mode for direct configuration from parallel flash. See Differences Between UltraScale FPGA Families, page 11.*

The UltraScale architecture-based FPGAs add a new configuration mode for configuring from two quad SPI flash memories in parallel. The resulting x8 configuration reduces the

configuration time while still allowing for the use of standard, high-speed, low-cost serial NOR configuration memories.

This section describes other performance improvements.

- A higher performance internal configuration clock (CCLK) provides up to double the max frequency with less frequency variation.
- Because a significant amount of the power-up configuration time can be the power-on reset (POR) delay, the UltraScale architecture-based FPGAs offer a dedicated pin (POR_OVERRIDE) that can be set to reduce the delay when the user knows the power supplies will ramp quickly enough.
- The internal scanning for configuration bit errors caused by single event upsets (SEU) is also faster, reducing the time to mitigate an error.
- A new AES-GCM algorithm for decrypting encrypted bitstreams makes secure configuration performance on par with unencrypted configuration. The pin name for the RAM-based encryption key backup supply is V_{BATT} instead of V_{CCBATT}.
- The Device DNA unique identifier is increased from 57 bits to 96 bits.

The UltraScale architecture-based FPGAs combine RDWR_B and FCS_B on one pin and move it into the dedicated configuration bank 0. CSI_B and ADV_B are combined on another pin. The I/O flexibility is maximized by reducing the number of pins required for configuration.

Other configuration pins that were dual-purpose I/O in the 7 series and are dedicated in bank 0 for the UltraScale architecture-based FPGAs include the first four data pins D[03:00] and the control pin for pull-ups during configuration, PUDC_B.

The UltraScale architecture-based FPGAs use only one I/O bank (bank 65) for multi-function pins needed for some configuration modes. The pin-outs describe the banks for each pin. Configuration interfaces can be powered at 1.5V, 1.8V, 2.5V, or 3.3V. See Configuration Banks Voltage Select (Kintex UltraScale and Virtex UltraScale FPGAs), page 39 for voltage ranges supported by mode and by device.

The UltraScale architecture-based FPGAs continue to support the Internal Configuration Access Port (ICAP), providing direct access between FPGA logic and configuration functions. The ICAP interface is similar to the SelectMAP interface, and allows user logic to initiate active reconfiguration. A new Media Configuration Access Port (MCAP) provides a similar connection to the integrated block for PCI Express. A small initial bitstream can be loaded quickly at power-up to enable the PCIe interface, and then the rest of the configuration can be loaded through the PCIe interface - this known as tandem PCIe. Alternatively, the second part of the configuration can be loaded through the standard configuration interfaces - this is known as tandem PROM. See the *UltraScale Architecture Gen3 Integrated Block for PCI Express LogiCORE IP Product Guide* (PG156) [Ref 1] for information on configuration over the PCIe interface.

Chapter 7, Design Entry provides details on the configuration and boundary scan components. The following primitives are different than the 7 series primitives:

- DNA_PORTE2
  - Extended to 96 bits
- FRAME_ECCE3
  - Used by Soft Error Mitigation (SEM) IP
- FRAME_ECCE4
  - Used by Soft Error Mitigation (SEM) IP
- STARTUPE3
  - Adds access to more configuration pins (FCS_B and D[03:00])
- MASTER_JTAG
  - New feature to provide internal, secure access to the JTAG logic
- CAPTUREE2
  - No longer supported
  - Use the Vivado® Integrated Logic Analyzer to monitor the internal signals of a design. See *Integrated Logic Analyzer Product Guide* (PG172) [Ref 2].
- ICAPE3
  - Adds additional status signals
  - Supports x32 only

KU025 Differences

The smallest Kintex UltraScale device, the KU025, has a reduced set of configuration features. The KU025 does not support RSA authentication, SEU mitigation (SEM) IP, or post-configuration CRC.

Table 1-2 summarizes the differences between the 7 series and UltraScale families, including the features that are restricted in the Kintex UltraScale KU025 device.

Table 1-2: **Differences Between Families**

Feature	7 Series	Kintex UltraScale and Virtex UltraScale	Kintex UltraScale+ and Virtex UltraScale+
Internal CCLK	< 100 MHz	> 100 MHz	> 100 MHz
POR_OVERRIDE	No	Yes	Yes
SEM IP	Yes, uses FRAME_ECCE2	Yes (except KU025), faster, uses enhanced FRAME_ECCE3	Yes, faster, uses enhanced FRAME_ECCE4

Table 1-2: **Differences Between Families *(Cont'd)***

Feature	7 Series	Kintex UltraScale and Virtex UltraScale	Kintex UltraScale+ and Virtex UltraScale+
Readback CRC	Yes	No, use SEM IP instead (SEM IP is not supported in KU025)	No, use SEM IP instead
Encryption	AES-CBC	AES-GCM; similar performance to standard configuration	AES-GCM; similar performance to standard configuration
Authentication	HMAC	AES-GCM and RSA (except KU025)	AES-GCM and RSA
Device DNA	DNA_PORT, 57 bits	DNA_PORTE2, 96 bits	DNA_PORTE2, 96 bits
Bank 0 Voltage	1.5V-3.3V	1.5V-3.3V	1.5V-1.8V
Multi-function bank voltage	1.5V-3.3V	Kintex except KU095: 1.5V-3.3V Virtex and KU095: 1.5V-1.8V	1.5V-1.8V
Multi-function banks	14, 15	65	65
RDWR_B and FCS_B Pins	RDWR_B, FCS_B separate, multi-function	RDWR_FCS_B combined, dedicated	FDWR_FCS_B combined, dedicated
D[03:00], PUDC_B	Multi-function, not included in STARTUPE2	Dedicated, included in STARTUPE3	Dedicated, included in STARTUPE3
MCAP	N/A	Yes	Yes
MASTER_JTAG	N/A	Yes	Yes
CAPTUREE2	Yes	No	No
ICAP	ICAPE2; 16-bit, 32-bit	ICAPE3; 32-bit, additional status signals	ICAPE3; 32-bit, additional status signals

Differences in 3D ICs

3D ICs using SSI (Stacked Silicon Interconnect) technology support the same configuration modes as the monolithic devices. See the *UltraScale Architecture and Products Overview,* (DS890) [Ref 4] and the *3D ICs* website [Ref 5] for more information on devices using SSI technology.

3D ICs have multiple super logic regions (SLRs), each with its own configuration controller. One SLR is defined as the master, while the others are the slaves (see Table 1-3). The configuration banks and the PCIe blocks that support tandem configuration are always located in the master SLR. The SLRs are numbered from 0 at the bottom of the device floorplan.

Table 1-3: **UltraScale Devices Using SSI Technology**

Device	Master SLR Index	Slave SLR Index
KU085	SLR0	SLR1
KU115	SLR0	SLR1
VU125	SLR0	SLR1
VU160	SLR1	SLR0 and SLR2
VU190	SLR1	SLR0 and SLR2
VU440	SLR1	SLR0 and SLR2
VU5P	SLR0	SLR1
VU7P	SLR0	SLR1
VU9P	SLR1	SLR0 and SLR2
VU11P	SLR0	SLR1 and SLR2
VU13P	SLR1	SLR0, SLR2, and SLR3
VU19P	SLR1	SLR0, SLR2, and SLR3
VU27P	SLR1	SLR0, SLR2, and SLR3
VU29P	SLR1	SLR0, SLR2, and SLR3
VU35P	SLR0	SLR1
VU37P	SLR0	SLR1 and SLR2
VU45P	SLR0	SLR1
VU47P	SLR0	SLR1 and SLR2
VU57P	SLR0	SLR1 and SLR2

Notes:
1. All Master SLRs have a CONFIG_ORDER_INDEX value of 0.

Design Considerations

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

FPGA Configuration Data Source

UltraScale architecture-based FPGAs are designed for maximum flexibility. The FPGA either automatically loads itself with configuration data from a nonvolatile flash memory, or

another external device (a processor or microcontroller) can download the configuration data. 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. In master mode, the FPGA's configuration bitstream typically resides in nonvolatile memory on the same board. The FPGA internally generates a configuration clock signal called CCLK, and the FPGA controls the configuration process by sending a clock or addresses to the flash memory. See Figure 1-1.

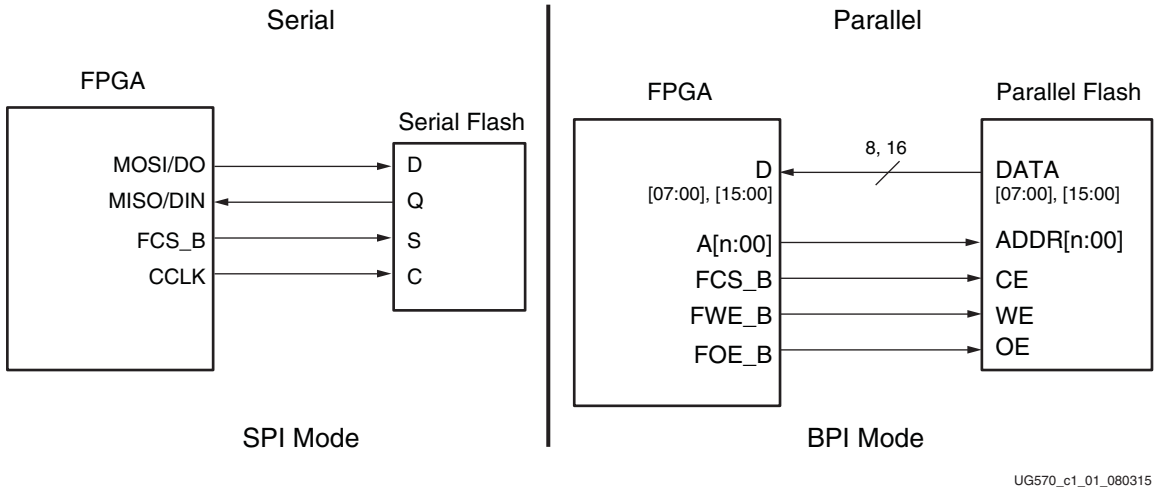


Figure 1-1: **Master Configuration Modes**

Slave Modes

The externally controlled loading FPGA configuration modes, generically called slave modes, are also available with either a serial or parallel datapath. In slave mode, an external processor, microcontroller, DSP processor, or tester downloads the configuration image into the FPGA, as shown in Figure 1-2. The advantage of the slave configuration modes is that the FPGA bitstream can reside almost anywhere in the overall system. The bitstream can reside in flash along with the host processor's code, on a hard disk, or somewhere over a network connection.

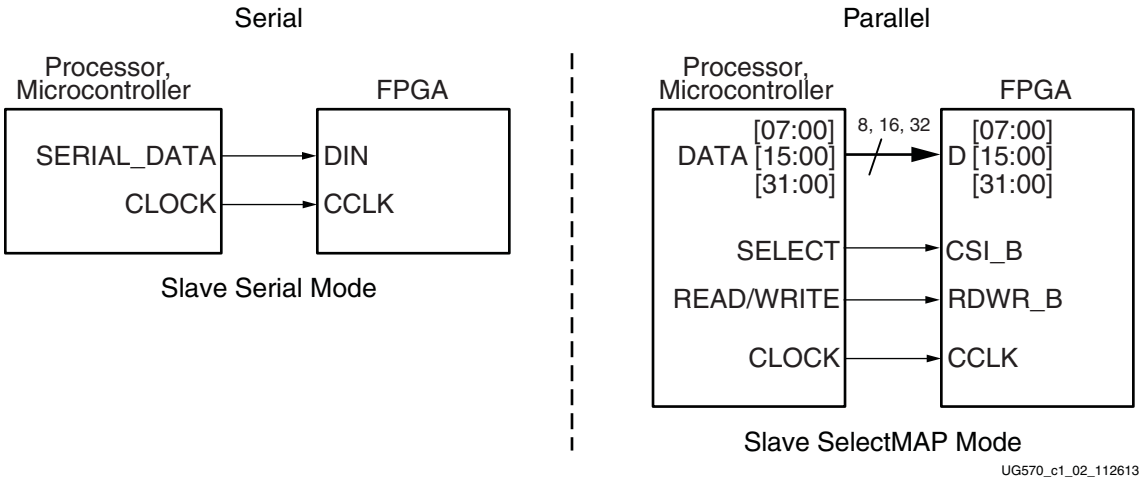


Figure 1-2: **Slave Configuration Modes**

The slave serial mode is an uncomplicated interface that consists of a clock and serial data input. The slave SelectMAP mode is a x8-, x16-, or x32-bit-wide processor peripheral interface, including a chip-select input and a read/write control input.

JTAG Connection

The JTAG mode is also a serial configuration mode, popular for prototyping and highly utilized for board test. The four-pin JTAG boundary scan interface is common on board testers and debugging hardware. The Xilinx programming cables for UltraScale architecture-based FPGAs use the JTAG interface for prototype download and debugging. Regardless of the configuration mode ultimately used in the application, it is best to also include a JTAG configuration path to ease design development. See Figure 1-3.

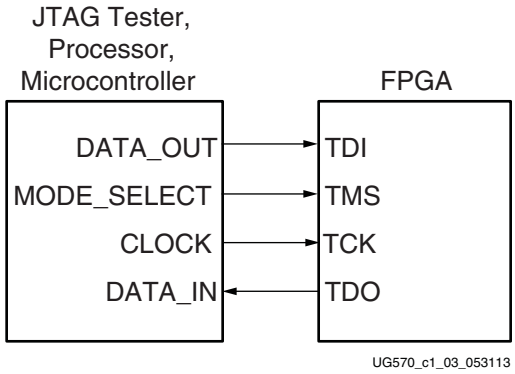


Figure 1-3: **JTAG Configuration Mode**

A Basic Configuration Solution

In a 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 serial NOR flash device. The Xilinx tools provide device programming support for select flash memories, by communicating with the FPGA through its standard JTAG interface and programming the flash indirectly through the FPGA.

A Low-Cost Configuration Solution

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

- If there is spare nonvolatile memory already available in the system, the bitstream 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 slave modes should be considered: slave serial or slave parallel mode, or JTAG mode.
- If nonvolatile memory is already required for an application, it is possible to leverage it to also store FPGA configuration bitstreams with small or no incremental cost. For example, the FPGA configuration bitstream 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.

A High-Speed Configuration 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.
- In master modes, the FPGA internally generates the CCLK configuration clock signal. 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.
- Using the external EMCCLK clock source option enables a precision external clock source for optimal configuration performance.

The general calculation used to estimate the configuration time is given in Equation 1-1.

$$ConfigurationTime = \frac{BitstreamSize}{ConfigurationRate \times DatabusWidth} \qquad \textit{Equation 1-1}$$

The UltraScale FPGA bitstream size can be found in Table 1-4, page 21. The maximum configuration clock frequency is dependent on the configuration mode and application implementation. Guidelines to calculate the maximum configuration clock frequency are provided in Chapter 2, Master SPI Configuration Mode, and Chapter 4, Master BPI Configuration Mode. If the configuration time from power-up is required then T_{POR} should be added to the configuration time.

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.

Protecting a Bitstream

Like processor code, a bitstream that defines the device's functionality loads into the device during power-on. Since this configuration data is stored off chip there exists a possibility of unauthorized duplication / modification.

Like processors, there are multiple techniques to protect the bitstream and any embedded intellectual property (IP) cores. The surest way to protect the confidentiality of your IP is to encrypt the configuration data using an AES-256 key. Keys for the on-chip decryption logic can be stored in either battery-backed RAM or one time programmable eFUSEs. This technique allows for off-chip storage of your IP protected with high grade encryption.

Loading Multiple FPGAs

Generally, each FPGA in a system has a unique bitstream. Multiple, different FPGA bitstreams 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, and programming can be done through ganged configuration. Ganged configuration is supported in the slave serial and slave SelectMAP modes, and in BPI asynchronous read mode.

Device Resources and Configuration Bitstream Lengths

A complete bitstream for each device has a fixed length for a given set of configuration options, independent of the logic used. However, bitstream options such as compression (BITSTREAM.GENERAL.COMPRESS) can change the required bitstream length. Because compressed bitstreams can change in size between design iterations, memory space should be reserved for the full uncompressed bitstream. Table 1-4 shows the default bitstream lengths and Table 1-5 shows other device-specific information.

Table 1-4: **Bitstream Length for UltraScale Architecture-based FPGAs**

Device	Configuration Bitstream Length (bits)	Minimum Configuration Flash Memory Size (Mb)	Configuration Frames	Frame Length in Words[1]	Configuration Array Size in Words[2]	Configuration Overhead in Words
Kintex UltraScale FPGAs						
KU025	128,055,264	128	32,530	123	4,001,190	537
KU035	128,055,264	128	32,530	123	4,001,190	537
KU040	128,055,264	128	32,530	123	4,001,190	537
KU060	192,999,264	256	49,030	123	6,030,690	537
KU085	386,012,288	512	98,060	123	12,061,380	1,504
KU095	286,746,912	512	72,848	123	8,960,304	537
KU115	386,012,288	512	98,060	123	12,061,380	1,504
Virtex UltraScale FPGAs						
VU065	200,713,824	256	50,990	123	6,271,770	537
VU080	286,746,912	512	72,848	123	8,960,304	537
VU095	286,746,912	512	72,848	123	8,960,304	537
VU125	401,441,408	512	101,980	123	12,543,540	1,504
VU160	602,156,064	1,024	152,970	123	18,815,310	2,067
VU190	602,156,064	1,024	152,970	123	18,815,310	2,067
VU440	1,031,731,104	1,024	262,110	123	32,239,530	2,067
Kintex UltraScale+ FPGAs						
KU3P	123,449,056	128	41,476	93	3,857,268	515
KU5P	123,449,056	128	41,476	93	3,857,268	515
KU9P	212,086,240	256	71,260	93	6,627,180	515
KU11P	188,647,264	256	63,384	93	5,894,712	515
KU13P	229,605,952	256	77,147	93	7,174,671	515
KU15P	290,744,896	512	97,691	93	9,085,263	515
KU19P	426,770,432	512	143,484	93	13,344,012	515
Virtex UltraScale+ FPGAs						
VU3P	213,752,800	256	71,820	93	6,679,260	515
VU5P	427,519,232	512	143,640	93	13,358,520	1,456
VU7P	427,519,232	512	143,640	93	13,358,520	1,456
VU9P	641,272,864	1,024	215,460	93	20,037,780	1,997
VU11P	679,913,248	1,024	228,444	93	21,245,292	1,997
VU13P	906,547,008	1,024	304,592	93	28,327,056	2,538
VU19P	1,592,895,936	2,048	535,220	93	49,775,460	2,538
VU23P	521,959,264	512	175,384	93	16,310,712	515

Table 1-4: **Bitstream Length for UltraScale Architecture-based FPGAs *(Cont'd)***

Device	Configuration Bitstream Length (bits)	Minimum Configuration Flash Memory Size (Mb)	Configuration Frames	Frame Length in Words(1)	Configuration Array Size in Words(2)	Configuration Overhead in Words
VU27P	906,547,008	1,024	304,592	93	28,327,056	2,538
VU29P	906,547,008	1,024	304,592	93	28,327,056	2,538
VU31P	226,632,928	256	76,148	93	7,081,764	515
VU33P	226,632,928	256	76,148	93	7,081,764	515
VU35P	453,279,488	512	152,296	93	14,163,528	1456
VU37P	679,913,248	1024	228,444	93	21,245,292	1997
VU45P	453,279,488	512	152,296	93	14,163,528	1456
VU47P	679,913,248	1,024	228,444	93	21,245,292	1997
VU57P	679,913,248	1,024	228,444	93	21,245,292	1997

Notes:

1. All UltraScale FPGA configuration frames consist of 123 32-bit words. All UltraScale+ FPGA configuration frames consist of 93 32-bit words.
2. Configuration array size equals the number of configuration frames times the number of words per frame.

Table 1-5: **JTAG and IDCODE for UltraScale Architecture-based FPGAs**

Device	JTAG/Device IDCODE[31:0] (hex)(1)	Production IDCODE Revision	JTAG Instruction Length (bits)
Kintex UltraScale FPGAs			
KU025	X3824093	1 or later	6
KU035	X3823093	1 or later	6
KU040	X3822093	1 or later	6
KU060	X3919093	1 or later	6
KU085	X380F093	1 or later	12
KU095	X3844093	2 or later	6
KU115	X390D093	1 or later	12
Virtex UltraScale FPGAs			
VU065	X3939093	0 or later	6
VU080	X3843093	2 or later	6
VU095	X3842093	2 or later	6
VU125	X392D093	1 or later	12
VU160	X3933093	1 or later	18
VU190	X3931093	1 or later	18
VU440	X396D093	1 or later	18
Kintex UltraScale+ FPGAs			
KU3P	X4A63093	0 or later	6
KU5P	X4A62093	0 or later	6
KU9P	X484A093	2 or later	6
KU11P	X4A4E093	0 or later	6
KU13P	X4A52093	1 or later	6
KU15P	X4A56093	1 or later	6
KU19P	X4ACF093		6
Virtex UltraScale+ FPGAs			
VU3P	X4B39093	1 or later	6
VU5P	X4B2B093	1 or later	12
VU7P	X4B29093	1 or later	12
VU9P	X4B31093	1 or later	18
VU11P	X4B49093	0 or later	18
VU13P	X4B51093	0 or later	24
VU19P	X4BA1093	0 or later	24
VU23P	X4ACE093		6
VU27P	X4B43093	0 or later	24

Table 1-5: **JTAG and IDCODE for UltraScale Architecture-based FPGAs *(Cont'd)***

Device	JTAG/Device IDCODE[31:0] (hex)(1)	Production IDCODE Revision	JTAG Instruction Length (bits)
VU29P	X4B41093	0 or later	24
VU31P	X4B6B093	1 or later	6
VU33P	X4B69093	1 or later	6
VU35P	X4B71093	1 or later	12
VU37P	X4B79093	1 or later	18
VU45P	X4B73093	1 or later	12
VU47P	X4B7B093	1 or later	18
VU57P	X4B61093		18

Notes:
1. The "X" in the JTAG IDCODE value represents the revision field (IDCODE[31:28]) which can vary.

Recommended Design Flow and Configuration Factors

Although configuration is typically a one-time event, independent of the FPGA operation, configuration choices can affect design options. Consider configuration decisions early in the design cycle to eliminate challenges late in your design cycle.

- Determine which configuration mode is optimal based upon the system's characteristics.
- Allow for JTAG configuration as an additional mode for debugging purposes.
- Provide easy access to the configuration control and status pins for debugging.
- Plan for the multi-function pins that are active during configuration and accordingly, check for conflicts with the other uses of these pins.
- Provide quality signal integrity for key signals during PCB layout including the configuration clock, even though configuration can operate at a low frequency.
- Consider all aspects of the configuration sequence to reduce configuration time, including the power-up time for the supplies.
- Consider the variety of options when generating the configuration bitstream and check for device operation conflicts.
- Inform the Xilinx tools via the CONFIG_VOLTAGE, CFGBVS (UltraScale FPGAs only), and CONFIG_MODE properties in your design constraint file.
- Generate the configuration bitstream using the latest version of the Xilinx tools targeting the version of the device that will be used (ES or production).

IMPORTANT: Bitstreams generated for an engineering sample should not be used in production devices. Production devices must always use bitstreams with the correct production device target.

- Correct DRC errors and review any warnings when the configuration bitstream is generated.
- Determine if remote upgrade will be needed and plan for it. See *SPI Flash Programming Including Bitstream Revision Selection* (XAPP1191) [Ref 3].

Design Tools

Before implementing the design and generating the bitstream data file, it is important to review the configuration settings to make sure they are correct for your design. Once the bitstream settings are correct, the bitstream data file can be generated using the `write_bitstream` Tcl command or by using the **Generate Bitstream** button in the Vivado flow navigator. There are three types of configuration settings in Vivado IDE:

1. Design Properties for Configuration
2. Configuration Bitstream Settings
3. Bitstream File Format Settings

Design Properties for Configuration

Design properties are used during implementation and therefore the configuration related options (CONFIG_MODE, CONFIG_VOLTAGE, and CFGBVS in UltraScale FPGAs) are important to set before bitstreams are generated. CONFIG_MODE specifies the planned configuration mode selection to be defined by the mode pins, CONFIG_VOLTAGE defines the planned configuration interface voltage level on V_{CCO_0} (and V_{CCO_65} when used), and CFGBVS defines whether the CFGBVS pin in UltraScale FPGAs is tied to V_{CCO} or GND. See Configuration Banks Voltage Select (Kintex UltraScale and Virtex UltraScale FPGAs), page 39 for more information on these features. Refer to the *Vivado Design Suite Properties Reference Guide* (UG912) [Ref 6] for more details on how to define these design properties.

Configuration Bitstream Settings

The most common configuration settings fall into the device configuration bitstream settings category. These settings are properties on the device model that begin with "BITSTREAM." and are used during bitstream generation. Configuration settings include options for the configuration clock, encryption, compression, startup, and mode-specific options.

Both design properties for configuration and configuration bitstream settings can be defined by using the **Edit Device Properties** dialog for the selected synthesized or implemented design netlist. The following steps describe how to set various properties using this method:

1. Open a synthesized or implemented design.
2. Select **Tools** > **Edit Device Properties**.
3. In the **Edit Device Properties** dialog, select one of the categories in the left-hand column.
4. Set the properties to the desired values, and click **OK**.
5. Select **File** > **Save Constraints** to save the updated properties to the target XDC file.

You can also set these properties using the `set_property` command in an XDC file or in the Tcl command window.

Bitstream File Format Settings

Bitstream file format settings include options for generating an ASCII version of a bitstream or files used for readback. The **Bitstream Settings** button in the Vivado flow navigator or the **Flow** > **Bitstream Settings** menu selection opens the **Bitstream** section in the **Project Settings** popup window. For more details on configuration options in the Vivado design tools, see the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7].

Pinout Planning

The configuration mode(s) used in an application can affect the planning of the design pin-out. It is important to determine and plan for the configuration modes before beginning floorplanning or pin selection. The configuration mode not only determines the connectivity of selected pins, it also determines the V_{CCO} voltage required for the I/O bank that includes multi-function pins.

To determine the proper pin settings follow this procedure.

1. Determine the configuration mode(s) for the FPGA. Be sure to account not only for the primary configuration mode, but any additional configuration modes that are used for debugging or updates.
2. Determine the set of pins and the bank locations for both the primary and secondary configuration modes planned.
3. Determine how each of these pins are used and any restrictions placed on design usage as standard I/O. Consider internal and external pull-ups or pull-downs, connections to external devices, etc.

4. For each set of configuration pins, determine the common required I/O voltage support for the required configuration bank(s). Only compatible I/O standards can be used elsewhere in that bank.

Configuration Interfaces

Xilinx UltraScale FPGAs have seven configuration interfaces, and UltraScale+ FPGAs have five configuration interfaces. Each configuration interface corresponds to one or more configuration modes and bus width, shown in Table 1-6. For detailed interface timing information, see the respective data sheet [Ref 8] or [Ref 9].

Table 1-6: **Configuration Modes**

Configuration Mode	M[2:0]	Bus Width	CCLK Direction
Master serial(1)	000	x1	Output
Master SPI	001	x1, x2, x4, x8	Output
Master BPI	010	x8, x16	Output
Master SelectMAP(1)	100	x8, x16	Output
JTAG only(2)	101	x1	N/A
Slave SelectMAP	110	x8, x16, x32	Input
Slave Serial(3)	111	x1	Input

Notes:

1. Not recommended in UltraScale FPGAs, and not supported in UltraScale+ FPGAs. See Differences Between UltraScale FPGA Families, page 11.
2. JTAG mode is always available independent of the Mode pin settings. Setting the Mode pins to JTAG-only is not recommended for devices based on SSI technology due to restrictions on ICAP access.
3. Slave serial 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 two banks on the FPGA. Bank 0 contains the dedicated configuration pins and is always part of every configuration interface. Bank 65 contains multi-function pins that are involved in a few of the configuration modes. If the Persist option is used (see Persist Option, page 181), the multi-function I/O for the selected configuration mode remain active after configuration. Table 1-7 and Table 1-8 show the configuration pins and their locations across the I/O banks. See Differences Between UltraScale FPGA Families, page 11.

Table 1-7: **Configuration Pins - Serial Modes**

Pin Name	Bank	JTAG (Only)	Master SPI				Slave Serial	Master Serial
			x1	x2	x4	x8 (dual x4)		
POR_OVERRIDE	N/A	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE
VBATT	N/A	VBATT	VBATT	VBATT	VBATT	VBATT	VBATT	VBATT
CFGBVS(1)	0	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS
M[2:0]	0	M[2:0]=101	M[2:0]=001	M[2:0]=001	M[2:0]=001	M[2:0]=001	M[2:0]=111	M[2:0]=000
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	CCLK	CCLK
PUDC_B(2)	0	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)
RDWR_FCS_B	0	-	FCS_B	FCS_B	FCS_B	FCS_B	-	-
D00_MOSI	0	-	MOSI	D00_MOSI	D00_MOSI	D00_MOSI	-	-
D01_DIN	0	-	DIN	D01_DIN	D01_DIN	D01_DIN	DIN	DIN
D02	0	-	-	-	D02	D02	-	-
D03	0	-	-	-	D03	D03	-	-
D[07:04]	65	-	-	-	-	D[07:04]	-	-
D[15:08]	65	-	-	-	-	-	-	-
A[15:00]_D[31:16]	65	-	-	-	-	-	-	-
A[28:16]	65	-	-	-	-	-	-	-
EMCCLK(3)	65	-	EMCCLK(3)	EMCCLK(3)	EMCCLK(3)	EMCCLK(3)	-	EMCCLK(3)
CSI_ADV_B	65	-	-	-	-	-	-	-
DOUT_CSO_B(4)(5)	65	-	DOUT(4)	-	-	-	DOUT(4)	DOUT(4)
FOE_B	65	-	-	-	-	-	-	-
FWE_FCS2_B	65	-	-	-	-	FCS2_B	-	-
RS0, RS1(6)	65	-	-	-	-	-	-	-

Table 1-7: **Configuration Pins - Serial Modes *(Cont'd)***

Pin Name	Bank	JTAG (Only)	Master SPI				Slave Serial	Master Serial
			x1	x2	x4	x8 (dual x4)		

Notes:

1. CFGBVS is available in UltraScale FPGAs only.
2. 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.
3. EMCCLK is only used when the external master CCLK enable option enables EMCCLK as an input for clocking the master configuration modes.
4. DOUT is only used in a serial configuration daisy-chain for outputting data to the downstream FPGA (or for the Debug Bitstream option). Otherwise, DOUT is high-impedance.
5. 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.
6. RS0 and RS1 are only driven in BPI mode when a MultiBoot event is initiated or when the Configuration Fallback option is enabled and a Fallback event occurs. Otherwise, RS0 and RS1 are high-impedance. RS[1:0] pins are not recommended to be used in User mode when they are used for configuration.
7. Dashes indicate that the pin is not used in the configuration mode and is high-impedance and ignored during configuration.

Table 1-8: **Configuration Pins - Parallel Modes**

Pin Name	Bank	Master BPI		Master SelectMAP		Slave SelectMAP		
		x8	x16	x8	x16	x8	x16	x32
POR_OVERRIDE	N/A	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE
VBATT	N/A	VBATT	VBATT	VBATT	VBATT	VBATT	VBATT	VBATT
CFGBVS(1)	0	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS	CFGBVS
M[2:0]	0	M[2:0]=010	M[2:0]=010	M[2:0]=100	M[2:0]=100	M[2:0]=110	M[2:0]=110	M[2:0]=110
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	CCLK	CCLK
PUDC_B(2)	0	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)	PUDC_B(2)
RDWR_FCS_B	0	FCS_B	FCS_B	RDWR_B	RDWR_B	RDWR_B	RDWR_B	RDWR_B
D00_MOSI	0	D00	D00	D00	D00	D00	D00	D00
D01_DIN	0	D01	D01	D01	D01	D01	D01	D01
D02	0	D02	D02	D02	D02	D02	D02	D02
D03	0	D03	D03	D03	D03	D03	D03	D03
D[07:04]	65	D[07:04]	D[07:04]	D[07:04]	D[07:04]	D[07:04]	D[07:04]	D[07:04]
D[15:08]	65	-	D[15:08]	-	D[15:08]	-	D[15:08]	D[15:08]
A[15:00]_D[31:16]	65	A[15:00]	A[15:00]	-	-	-	-	D[31:16]
EMCCLK(3)	65	EMCCLK(3)	EMCCLK(3)	EMCCLK(3)	EMCCLK(3)	-	-	-
CSI_ADV_B	65	ADV_B	ADV_B	CSI_B	CSI_B	CSI_B	CSI_B	CSI_B

Table 1-8: **Configuration Pins - Parallel Modes *(Cont'd)***

Pin Name	Bank	Master BPI		Master SelectMAP		Slave SelectMAP		
		x8	x16	x8	x16	x8	x16	x32
DOUT_CSO_B[4][5]	65	CSO_B[5]	CSO_B[5]	CSO_B[5]	CSO_B[5]	CSO_B[5]	CSO_B[5]	CSO_B[5]
A[28:16]	65	A[28:16]	A[28:16]	-	-	-	-	-
FOE_B	65	FOE_B	FOE_B	-	-	-	-	-
FWE_FCS2_B	65	FWE_B	FWE_B	-	-	-	-	-
RS0, RS1[6]	65	RS0, RS1[6]	RS0, RS1[6]	-	-	-	-	-

Notes:

1. CFGBVS is available in UltraScale FPGAs only.
2. 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.
3. EMCCLK is only used when the external master CCLK enable option enables EMCCLK as an input for clocking the master configuration modes.
4. DOUT is only used in a serial configuration daisy-chain for outputting data to the downstream FPGA (or for the Debug Bitstream option). Otherwise, DOUT is high-impedance.
5. 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.
6. RS0 and RS1 are only driven in BPI mode when a MultiBoot event is initiated or when the Configuration Fallback option is enabled and a Fallback event occurs. Otherwise, RS0 and RS1 are high-impedance. RS[1:0] pins are not recommended to be used in User mode when they are used for configuration.
7. Dashes indicate that the pin is not used in the configuration mode and is high-impedance and ignored during configuration.

The definition of each configuration pin is summarized in Table 1-9.

Table 1-9: **Configuration Pin Definitions**

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
POR_ OVERRIDE	N/A	Dedicated	Input	Power On Reset Delay Override	All	N/A	Reduces T_{POR} time (from power up to INIT_B rise) as specified in data sheet. Connect directly to V_{CCINT} for a shorter T_{POR} time if required and if supported by the power-up timing of the configuration data source. Connect to GND for standard longer POR delay. **CAUTION!** *Do not allow this pin to float before and during configuration. Must be tied to V_{CCINT} or GND. Do not connect to V_{CCO_0}.*
VBATT	N/A	Supply Voltage	N/A	Battery Backup Supply	Serial, SPI, SelectMAP, BPI, JTAG (with BBRAM encryption)	N/A	Battery backup supply for the FPGA 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.
					Unused	N/A	If there is no requirement to use the decryptor key from the volatile key storage area, connect this pin to GND or V_{CCAUX}.

Table 1-9: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
M[2:0]	0	Dedicated	Input	Configuration Mode	All	≤1 kΩ	Determine the configuration mode. See Table 1-6 for the configuration mode settings. Connect each mode pin either directly, or via a ≤1 kΩ resistor, to V_{CCO_0} or GND.
TCK	0	Dedicated	Input	IEEE Std 1149.1 (JTAG) Test Clock	JTAG	10 kΩ	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.
					Unused	N/A	Ignored and can be left unconnected.
TMS	0	Dedicated	Input	JTAG Test Mode Select	JTAG	10 kΩ	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.
					Unused	N/A	Ignored and can be left unconnected.
TDI	0	Dedicated	Input	JTAG Test Data Input	JTAG	N/A	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.
					Unused	N/A	Ignored and can be left unconnected.
TDO	0	Dedicated	Output	JTAG Test Data Output	JTAG	N/A	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.
					Unused	N/A	Ignored and can be left unconnected.
PROGRAM_B	0	Dedicated	Input	Program (bar)	All	≤4.7 kΩ	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 is initiated upon the falling edge, and configuration (i.e. programming) sequence begins upon the following rising edge. PROGRAM_B can externally be held Low during power-up to stall the power-on configuration sequence at the end of the initialization process. Dedicated pins remain disabled while PROGRAM_B is held Low. Connect PROGRAM_B to an external ≤4.7 kΩ pull-up resistor to V_{CCO_0} to ensure a stable High input. Recommended push-button to GND to enable manual configuration reset.

Table 1-9: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
INIT_B	0	Dedicated	Bidirectional (open-drain)	Initialization (bar)	All	4.7 kΩ	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. Note that INIT_B does not drive Low when V_{CCINT} is powered down. In UltraScale+ devices the INIT_B pin might be seen as High (because of external resistors on board for INIT_B) for approximately 40 ms after power ON. The initial High time depends on the POR_OVERRIDE setting. With POR_OVERRIDE Low, the High time is approx. 40 ms. With POR_OVERRIDE High, the High time is approx. 9 ms.) 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. After configuration, INIT_B can optionally be leveraged to indicate when the FPGA has detected a configuration error. 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	All	4.7 kΩ	A High signal on the DONE pin indicates completion of the configuration sequence. By default, the DONE output is open-drain. ***Note:*** DONE has a default internal pull-up resistor of approximately 10 kΩ. External 4.7 kΩ resistor circuits are not required but are recommended. For multiple SLR devices the DONE pull-up resistor cannot be stronger than 4.7 kΩ. In UltraScale+ devices the DONE pin might be seen as High (because of external resistors on board for DONE) for approximately 40 ms after power ON. (The initial High time depends on the POR_OVERRIDE setting. With POR_OVERRIDE Low, the High time is approx. 40 ms. With POR_OVERRIDE High, the High time is approx. 9 ms.)
CCLK	0	Dedicated	Input or Output	Configuration Clock	Master Serial, Master SelectMAP, SPI, BPI (synchronous)	N/A	Runs the synchronous FPGA configuration sequence by default. The FPGA sources the configuration clock and drives CCLK as an output. Note: Treat CCLK as a critical clock signal to ensure good signal integrity.
					BPI (asynchronous)	N/A	High-impedance, and can be left unconnected, and flash CLK can be connected to GND.
					Slave Serial, Slave SelectMAP	N/A	An input and requires connection to an external clock source.
					JTAG	N/A	High-impedance, and can be left unconnected.

Table 1-9: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
PUDC_B	0	Dedicated	Input	Pull-Up During Configuration (bar)	All	≤1 kΩ	Active-Low input enables internal pull-up resistors on the SelectIO pins after power-up and during configuration, including multipurpose configuration pins when not used for the selected configuration mode. 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 an ≤1 kΩ resistor, to V_{CCO_0} or GND.
RDWR_FCS_B	0	Dedicated	Input or Output	Read/Write (bar) or Flash Chip Select (bar)	SelectMAP (RDWR_B)	N/A	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. 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.
					SelectMAP, Unused	N/A	Connect to GND.
					SPI (FCS_B)	2.4 kΩ	Active-Low chip select output that enables flash devices for configuration. Connect to the flash device chip-select input and connect to an external ≤4.7 kΩ pull-up resistor to V_{CCO_0} (2.4 kΩ recommended).
					BPI (FCS_B)	≤4.7 kΩ	Active-Low chip select output that enables flash devices for configuration. Connect to the flash device chip-select input and connect to an external ≤4.7 kΩ pull-up resistor to V_{CCO_0}.
					Serial, JTAG	N/A	High-impedance and ignored, and can be left unconnected.
D00_MOSI	0	Dedicated	Bidirectional	Data Bit 0 or Master-Output, Slave-Input	SPI x1 (MOSI)	N/A	Output for sending commands to the serial (slave) flash device. Connect to the flash serial data input (DQ0/D/SI/IO0) pin.
					SPI x2/x4/x8 (D00_MOSI)	N/A	Dual-purpose Data In/Out pin. Output for sending commands to the serial (slave) flash device. LSB data input from dual or quad flash device. Connect to the flash serial data input/output (DQ0/D/SI/IO0) pin.
					SelectMAP, BPI (D00)	N/A	Multi-purpose pin that functions as the D00 data input pin. See D[31:00] row in this table.
					Serial, JTAG	N/A	High-impedance and ignored, and can be left unconnected.

Table 1-9: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
D01_DIN	0	Dedicated	Input or Bidirectional	Data Bit 1 or Data Input	BPI, SelectMAP (D01)	N/A	Multi-purpose pin that functions as the D01 data input pin. See D[31:00] row in this table.
					Serial, SPI x1 (DIN)	N/A	Data input that receives serial data from the data source. Connect DIN to the serial data output pin of the serial data source (DQ1/Q/SO/IO1 pin). By default, data from DIN is captured on the rising edge of CCLK.
					SPI x2/x4/x8 (D01)	N/A	Data input from dual or quad flash device. Connect to the flash data output pin (DQ1/Q/SO/IO1 pin).
					JTAG	N/A	Ignored, and can be left unconnected.
D02	0	Dedicated	Input or Bidirectional	Data Bit 2	SPI x4/x8	4.7 kΩ	Connect to the flash quad data bit 2 output (DQ2/W#/WP#/IO2) pin and connect to an external 4.7 kΩ pull-up resistor to V_{CCO_0}.
					BPI, SelectMAP	N/A	Multi-purpose pin that functions as the D02 data input pin. See D[31:00] row in this table.
					Serial, SPI x1/x2, JTAG	N/A	Ignored, and can be left unconnected.
D03	0	Dedicated	Input or Bidirectional	Data Bit 3	SPI x4/x8	4.7 kΩ	Connect to the flash quad data bit 3 output (DQ3/HOLD#/IO3) pin and connect to an external 4.7 kΩ pull-up resistor to V_{CCO_0}.
					BPI, SelectMAP	N/A	Multi-purpose pin that functions as the D03 data input pin. See D[31:00] row in this table.
					Serial, SPI x1/x2, JTAG	N/A	Ignored, and can be left unconnected.
CFGBVS	0	Dedicated	Input or Bidirectional	Configuration Banks Voltage Select	All	N/A	Supported in Kintex UltraScale and Virtex UltraScale FPGAs only. Determines the I/O voltage operating range and voltage tolerance for the dedicated configuration bank 0, and during configuration for the configuration pins in bank 65, when those banks are HR I/O banks. 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 this pin must be tied High (connected to V_{CCO_0}). Tie CFGBVS to Low (connect to GND) only if the V_{CCO_0} for bank 0 is 1.5V or 1.8V. When bank 65 is used for configuration, it should have the same voltage as bank 0. See Configuration Banks Voltage Select (Kintex UltraScale and Virtex UltraScale FPGAs), page 39. **CAUTION!** *To avoid device damage, this pin must be connected correctly to either V_{CCO_0} or GND.*

Table 1-9: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
EMCCLK	65	Multi-function	Input	External Master Configuration Clock	SPI, BPI, Master Serial, Master SelectMAP	N/A	Optional external clock input for running the configuration logic in a master mode (versus the internal configuration oscillator). 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 and is forwarded for output as the master CCLK signal.
					Slave Serial, Slave SelectMAP, JTAG, or Unused	N/A	Ignored and can be left unconnected, or connected as an I/O pin after configuration.
CSI_ADV_B	65	Multi-function	Input or Output	Chip Select Input (bar) or Address Valid (bar)	SelectMAP (CSI_B)	N/A	Active-Low input that enables the FPGA SelectMAP configuration interface. 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.
					SelectMAP, Unused	N/A	Connect to GND.
					BPI (ADV_B) (with flash that support an address valid input)	≤4.7 kΩ	Active-Low address valid output signal for a parallel NOR flash that supports an address valid input. Connect to the parallel NOR flash address valid input pin and connect to an external ≤4.7 kΩ pull-up resistor to V_{CCO_65}. Used in synchronous mode (recommended). In asynchronous mode, ADV_B can be left unconnected and flash ADV can be tied to GND.
					BPI, Unused	N/A	Can be left unconnected, or connected as an I/O pin after configuration.
					Serial, SPI, JTAG	N/A	Ignored and high-impedance, and can be left unconnected, or connected as an I/O pin after configuration.
DOUT_CSO_B	65	Multi-function	Output	Data Output or Chip Select Output (bar)	Serial, SPI x1 (DOUT)	N/A	Data output for a serial configuration daisy-chain. If the device is in a serial configuration daisy-chain, then connect to the DIN of the downstream slave-serial FPGA.
					BPI, SelectMAP (CSO_B)	330Ω	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_65} and connect to the CSI_B input of the downstream device.
					SPI x2/x4/x8, JTAG, or Unused	N/A	High-impedance and can be left unconnected, or connected as I/O after configuration.
					All	N/A	***Note:*** DOUT can output data when the Debug Bitstream option is enabled.

Table 1-9: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
D[31:00]	65 D[03:00] are in Bank 0	Multi-function	Input or Bidirectional	Data Bus	Serial, SPI, SelectMAP, BPI	N/A	A subset or all of the D[31:00] pins are the data bus interface for the serial, SPI, SelectMAP or BPI modes. By default, data from the data bus is captured on the rising edge of CCLK. The remaining data pins are unused, ignored, and high impedance during configuration, and D[31:04] can be used as I/O after configuration. The data bus signals are inputs when reading the configuration data from the flash. Data bus signals can be outputs during a write to a parallel flash Read Configuration register or during SelectMAP readback. For serial and SPI modes, also see the D00-D03 rows in this table.
					SPI	4.7 kΩ	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 flash read mode to dual output (x2), quad output (x4), or dual quad (x8) modes. • For SPI x1/x2/x4/x8: Connect D00_MOSI to the flash serial data input (DQ0/D/SI/IO0) pin. • For SPI x1/x2/x4/x8: Connect D01_DIN to the flash serial data output (DQ1/Q/SO/IO1) pin. • For SPI x4/x8: Connect D02 to the flash quad data bit 2 output (DQ2/W#/WP#/IO2) pin and connect to an external 4.7 kΩ pull-up resistor to V_{CCO_65}. • For SPI x4/x8: Connect D03 to the flash quad data bit 3 output (DQ3/HOLD#/IO3) pin and connect to an external 4.7 kΩ pull-up resistor to V_{CCO_65}. • For SPI x8: Connect D[07:04] to the second flash in the same way as D[03:00] are connected to the first flash.
					SelectMAP	N/A	The FPGA monitors the D[07:00] for an auto-bus-width-detect pattern that determines whether only D[07:00] (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. The data bus signals are inputs when reading the configuration data from the flash, and outputs during readback.
					BPI	N/A	The FPGA monitors the D[07:00] for an auto-bus-width-detect pattern that determines whether only D[07:00] (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 flash. The D[31:16] pins are multi-purpose pins that function as the BPI address A[15:00] pins. See A[28:00] row in this table.
					JTAG	N/A	All data pins are unused, ignored, and high impedance during configuration.

Table 1-9: **Configuration Pin Definitions *(Cont'd)***

Pin Name	Bank	Type	Direction	Function	Mode	Recommended External Pull-Up/ Pull-Down	Description
A[15:00]_ D[31:16]	65	Multi-function	Input or Output	Address Bus LSBs or Data Bus MSBs	BPI (A[15:00])	N/A	Address Bus bits 15 to 0 - see A[28:00] row in this table.
					SelectMAP x32	N/A	Data Bus bits 31 to 16 - see D[31:00] row in this table.
					Serial, SPI, BPI, SelectMAPx8/x16, JTAG	N/A	Ignored, and can be left unconnected, or connected as I/O after configuration.
A[28:00]	65	Multi-function	Output	Address Bus	BPI	N/A	Output addresses to a parallel NOR flash. A00 is the least-significant address bit. Connect the FPGA A[28:00] 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 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.
					SelectMAP	N/A	The A[15:00] pins are multi-purpose pins that function as the D[31:16] data bus pins. See D[31:00] row in this table.
					Serial, SPI, JTAG	N/A	High-impedance, ignored during configuration, and can be left unconnected, or connected as I/O after configuration.
FOE_B	65	Multi-function	Output	Flash Output-Enable (bar)	BPI	≤4.7 kΩ	Active-Low output-enable control signal for a parallel NOR flash. Connect to the flash output-enable input and connect to an external ≤4.7 kΩ pull-up resistor to V_{CCO_65}.
					Serial, SPI, SelectMAP, JTAG	N/A	High-impedance, and can be left unconnected, or connected as I/O after configuration.
FWE_FCS2_B	65	Multi-function	Output	Flash Write-Enable (bar) or Flash Chip Select 2 (bar)	BPI (FWE_B)	≤4.7 kΩ	Active-Low write-enable control signal for a parallel NOR flash. Connect to the flash write-enable input and connect to an external ≤4.7 kΩ pull-up resistor to V_{CCO_65}.
					SPI x8 (FCS2_B)	2.4 kΩ	Active-Low chip select output that enables second SPI quad flash device for configuration in x8 mode.
					Serial, SPI x1/x2/x4, SelectMAP, JTAG	N/A	High-impedance, and can be left unconnected, or connected as I/O after configuration.
RS[1:0]	65	Multi-function	Output	Revision Select	BPI only, with fallback, or with MultiBoot	N/A	Revision selection output pins. Normally high-impedance during configuration. When the bitstream configuration fallback 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.

Configuration Banks Voltage Select (Kintex UltraScale and Virtex UltraScale FPGAs)

In the Kintex UltraScale and Virtex UltraScale FPGAs, the configuration banks voltage select (CFGBVS) pin must be set to High or Low to determine the I/O voltage support for the pins in bank 0, and for the multi-function pins in bank 65 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 connected to the V_{CCO_0} supply of 3.3V or 2.5V, the configuration I/O support operation at 3.3V or 2.5V. When the CFGBVS pin is connected to GND, the configuration I/O support operation at 1.8V or 1.5V. There is no CFGBVS pin in the Kintex UltraScale+ and Virtex UltraScale+ FPGAs because their configuration I/O only support operation at 1.8V or 1.5V. The pin location is labeled RSVDGND and it must be connected to GND.

Configuration is not supported below the minimum recommended operating voltage for 1.5V as specified in the data sheet. The CFGBVS pin setting determines the I/O voltage support for bank 0 at all times, before, during, and after configuration. CFGBVS similarly controls the voltage tolerance on bank 65, but only during configuration.

The UltraScale FPGAs have two I/O bank types for configuration: high-range (HR) I/O banks support 3.3V and lower I/O standards, and high-performance (HP) banks support I/O standards of 1.8V or lower. The dedicated configuration and JTAG I/O are located in bank 0, which is a high-range bank type on all Kintex UltraScale and Virtex UltraScale devices, and a high-performance bank in Kintex UltraScale+ and Virtex UltraScale+ devices. Several of the configuration modes also rely on pins in bank 65. Bank 65 is an HR bank in most Kintex UltraScale FPGAs, an HP bank in the KU095 and Virtex UltraScale FPGAs, and an HP bank in all Kintex UltraScale+ and Virtex UltraScale+ FPGAs.

Table 1-10 shows the CFGBVS pin connection options and the corresponding set of valid V_{CCO_0} supply and I/O voltages.

Table 1-10: **CFGBVS Pin Connection for Bank V_{CCO} Supplies and I/O Signal Voltages**

CFGBVS Pin Connection	Supported Banks V_{CCO} Supply and I/O Signal Voltages	
	Kintex UltraScale except KU095 Banks 0, 65	Virtex UltraScale and KU095 Bank 0
V_{CCO_0} (3.3V or 2.5V)	3.3V or 2.5V	3.3V or 2.5V
GND	1.8V or 1.5V	1.8V or 1.5V

CAUTION! *When CFGBVS is connected to GND for 1.8V or 1.5V I/O operation, the V_{CCO_0} and I/O signals to bank 0 must be 1.8V (or lower). Otherwise, the device can be damaged from the application of voltages to pins on Bank 0 that are greater than the 1.8V operation maximum.*

The interface pins associated with the configuration mode can span bank 0 and bank 65, primarily when using 8-bit or wider data interfaces. When both banks are used for a configuration interface, the V_{CCO} pins for both banks must receive the same voltage to ensure a consistent I/O voltage interface and timing for all of the configuration interface pins. Using the same voltage for banks 0 and 65 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 157.

Use these steps to determine the proper CFGBVS pin setting:

1. Determine the configuration mode(s) for the FPGA. Note that 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 1-7 and Table 1-8).
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 UltraScale and Kintex KU095 FPGAs only support 1.8V/1.5V configuration on bank 65.
5. Set the CFGBVS pin to support the required configuration I/O voltage. See Table 1-11 and Table 1-12 for the appropriate CFGBVS pin setting.

Table 1-11: **Kintex UltraScale (Except KU095) Compatible Voltages and CFGBVS Pin Connection**

Configuration Mode	Banks Used	Configuration Interface I/O Voltage	Compatible Bank Voltages		Required CFGBVS Pin Connection
			Bank 0 V_{CCO_0} Voltage	Bank 65 V_{CCO_65} Voltage[2]	
JTAG (Only)[1]	0	3.3V	3.3V	Any[3][4]	V_{CCO_0}
		2.5V	2.5V	Any[3][4]	V_{CCO_0}
		1.8V	1.8V	Any[4]	GND
		1.5V	1.5V	Any[4]	GND
Serial, or SPI x1/x2/x4 (without DOUT or EMCCLK)	0	3.3V	3.3V	Any[3][4]	V_{CCO_0}
		2.5V	2.5V	Any[3][4]	V_{CCO_0}
		1.8V	1.8V	Any[4]	GND
		1.5V	1.5V	Any[4]	GND

Table 1-11: **Kintex UltraScale (Except KU095) Compatible Voltages and CFGBVS Pin Connection**

Configuration Mode	Banks Used	Configuration Interface I/O Voltage	Compatible Bank Voltages		Required CFGBVS Pin Connection
			Bank 0 V_{CCO_0} Voltage	Bank 65 V_{CCO_65} Voltage(2)	
SPI x8, SelectMAP, or BPI, or Serial with DOUT, or other SPI with EMCCLK	0 and 65	3.3V	3.3V	3.3V	V_{CCO_0}
		2.5V	2.5V	2.5V	V_{CCO_0}
		1.8V	1.8V	1.8V	GND
		1.5V	1.5V	1.5V	GND

Notes:

1. JTAG interface is always supported in bank 0 at the V_{CCO_0} voltage level regardless of the configuration mode.
2. In most Kintex UltraScale FPGAs, bank 65 is an HR I/O bank, supporting voltages up to 3.3V. In the Virtex UltraScale and KU095 FPGAs, bank 65 is an HP I/O bank, limited to 1.8V or lower I/O standards.
3. Using 2.5V or 3.3V on bank 65 is recommended when bank 0 is at 2.5V or 3.3V. See I/O Transition at the End of Startup, page 157.
4. Bank 65 is not used for configuration in this mode. V_{CCO_65} can be set according to the needs of the I/O interface after configuration.

Table 1-12: **Virtex UltraScale and Kintex UltraScale KU095 Compatible Voltages and CFGBVS Pin Connection**

Configuration Mode	Banks Used	Configuration Interface I/O Voltage	Compatible Bank Voltages		Required CFGBVS Pin Connection
			Bank 0 V_{CCO_0} Voltage	Bank 65 V_{CCO_65} Voltage	
JTAG (Only)(1)	0	3.3V	3.3V	≤ 1.8V(3)	V_{CCO_0}
		2.5V	2.5V	≤ 1.8V(3)	V_{CCO_0}
		1.8V	1.8V	≤ 1.8V(3)	GND
		1.5V	1.5V	≤ 1.8V(3)	GND
Serial, or SPI x1/x2/x4 (without DOUT or EMCCLK)	0	3.3V	3.3V	≤ 1.8V(3)	V_{CCO_0}
		2.5V	2.5V	≤ 1.8V(3)	V_{CCO_0}
		1.8V	1.8V	≤ 1.8V(3)	GND
		1.5V	1.5V	≤ 1.8V(3)	GND
SPI x8, SelectMAP, or BPI, or Serial with DOUT, or other SPI with EMCCLK	0 and 65(2)	1.8V	1.8V	1.8V	GND
		1.5V	1.5V	1.5V	GND

Notes:

1. JTAG interface is always supported in bank 0 at the V_{CCO_0} voltage level regardless of the configuration mode.
2. In the Virtex UltraScale FPGAs and Kintex UltraScale KU095 FPGA, bank 65 is a high-performance bank, limited to 1.8V or lower I/O standards. CFGBVS does not affect that bank.
3. Bank 65 is not used for configuration in this mode. V_{CCO_65} can be set according to the needs of the I/O interface after configuration.

Setting Configuration Voltage Options in the Vivado Tools

The choice of configuration voltage must be communicated to the Vivado tools by setting the CONFIG_VOLTAGE and/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 errors 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 Properties**), or through Tcl commands. See *Vivado Properties Reference Guide* (UG912) [Ref 6] for details on the Tcl syntax. See *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 10] for examples of how Vivado tools use these options.

Power-On Reset

To ensure proper power-on behavior, the guidelines in the respective data sheet ([Ref 8] or [Ref 9]) must be followed. For configuration, UltraScale architecture-based FPGAs require power on the V_{CCO_0}, V_{CCAUX}, V_{CCBRAM}, and V_{CCINT} pins. Power sequencing requirements are described in the data sheet. The power supplies should ramp monotonically within the power supply ramp time range specified in the data sheet. All supply voltages should be within the recommended operating ranges; any dips in V_{CCINT} or V_{CCAUX} below their data retention voltages in the data sheet can result in loss of configuration data.

The FPGA automatically provides a delay between power-on and the beginning of configuration, called the power-on reset (POR) delay. The POR delay count is short or long depending on POR_OVERRIDE. The TPOR delay starts from the time the last required supply rail is supplied to the FPGA at 95% of its nominal value, and ends with the FPGA asserting the INIT_B pin, sampling the Mode pins, and starting to toggle the CCLK if master mode is selected.

POR_OVERRIDE

The Power On Reset Override select (POR_OVERRIDE) pin must be set High or Low to determine the power-on delay before configuration begins. The POR_OVERRIDE is a logic input pin referenced between V_{CCINT} and GND. When the POR_OVERRIDE pin is High at power-up (e.g., connected to the V_{CCINT} supply rail), the POR delay is shortened as specified in the data sheet. When the POR_OVERRIDE pin is Low (e.g., connected to GND), the POR delay is longer. POR_OVERRIDE should be connected to GND unless the flash will always be ready as soon as the FPGA is powered up (see Power-On Sequence Precautions for Flash).

IMPORTANT: *Do not connect POR_OVERRIDE to V_{CCO_0} as with bank 0 pins. POR_OVERRIDE must be connected to V_{CCINT} or GND. Do not leave POR_OVERRIDE floating.*

The device always waits for the V_{CCINT} power-on threshold to be met before determining the POR_OVERRIDE value, eliminating the possibility of false High readings. V_{CCINT} is recommended to ramp first.

Power-On Sequence Precautions for Flash

At power-on, the FPGA automatically starts its configuration procedure. When the FPGA is in a master configuration mode, the configuration flash must be awake and ready to receive commands and/or clocks before the FPGA begins sending them. Because different power rails can supply the FPGA and flash or because the FPGA and flash can respond at different times along the ramp of a shared power supply, special attention to the FPGA and 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 flash, or vice versa. In addition, some 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 default FPGA power-on reset time (T_{POR}) can sufficiently delay the start of the FPGA configuration procedure such that the flash becomes ready before the start of the FPGA configuration procedure.

IMPORTANT: *Configuration modes with a bus width of 8, 16, or 32 require V_{CCO_65}, in addition to the V_{CCO_0} that is built in to the power-on sequence requirement of the FPGA. Make sure V_{CCO_65} is supplied at or before V_{CCO_0} to ensure proper configuration.*

In general, the system design must consider the effect of the power sequence, the power ramps, FPGA power-on reset timing, and flash power-up timing on the timing relationship between the start of FPGA configuration and the readiness of the flash. Refer to the Xilinx data sheet for FPGA power supply requirements and timing, and check the flash data sheet for the flash power-up timing requirements.

One of these system design approaches can ensure that the flash is ready to receive commands before the FPGA starts its configuration procedure:

- Control the sequence of the power supplies such that the flash is certain to be powered and ready before the FPGA begins its configuration procedure.
- Use the longer T_{POR} delay by connecting POR_OVERRIDE to GND.
- 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 flash becomes ready.

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 150 MHz, a 35% tolerance means that the ConfigRate setting cannot be faster than 111 MHz. However, an external clock source can be applied as fast as the specification allows. UltraScale 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 (set the CONFIG_VOLTAGE property)
3. Connecting EMCCLK on the board to your board's oscillator or other clock source

Dedicated configuration logic can divide the EMCCLK input or use the full rate (divide by 1). The EXTMASTERCCLK_EN_en option is set in the Vivado tools with the BITSTREAM.CONFIG.EXTMASTERCCLK_EN property (see *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for details):

```
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN Disable|Div-1|Div-2
|Div-3|Div-4|Div-6|Div-8|Div-12|Div-16|Div-24|Div-48
```

The default is disable (use the internal CCLK).

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 signal integrity issues that can cause errors during configuration. EMCCLK is a single-ended clock input.

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.

VBATT

The dedicated V_{BATT} pin provides a backup power supply for the AES decryptor key memory, similar to V_{CCBATT} in the 7 series FPGAs. V_{BATT} is required only when bitstream encryption is used, the key is stored in battery-backed RAM, and a backup supply to the key space (powered by V_{CCAUX}) is desired. If encryption is not used or is used with the eFUSE key, tie V_{BATT} to V_{CCAUX} or GND. For more details on how V_{BATT} is used for encryption applications, see Chapter 8, Bitstream Security, eFUSEs, and Device DNA.

Master SPI Configuration Mode

Introduction

The master SPI configuration mode in UltraScale™ architecture-based FPGAs enables the use of low pin count, industry-standard serial NOR flash devices for bitstream storage. The FPGA supports a direct connection to the de facto standard, four-pin SPI interface of a serial NOR flash device for reading a stored bitstream.

Master SPI Interface

The master SPI configuration mode reads from standard 1-bit serial NOR flash, and can optionally read from flash devices that support x2 and x4 Fast Output Read operations. Xilinx UltraScale FPGAs also provide a x8 master SPI configuration mode by connecting to two identical flash memories that contain the bitstream split across both devices. These modes are proportionally faster than the standard 1-bit SPI interface and are selected with the bitstream option BITSTREAM.CONFIG.SPI_BUSWIDTH. In addition, a negative edge clocking mode (BITSTREAM.CONFIG.SPI_FALL_EDGE) is available to make better use of the entire clock period and allow higher configuration speed. The master SPI configuration interface is represented in Figure 2-1.

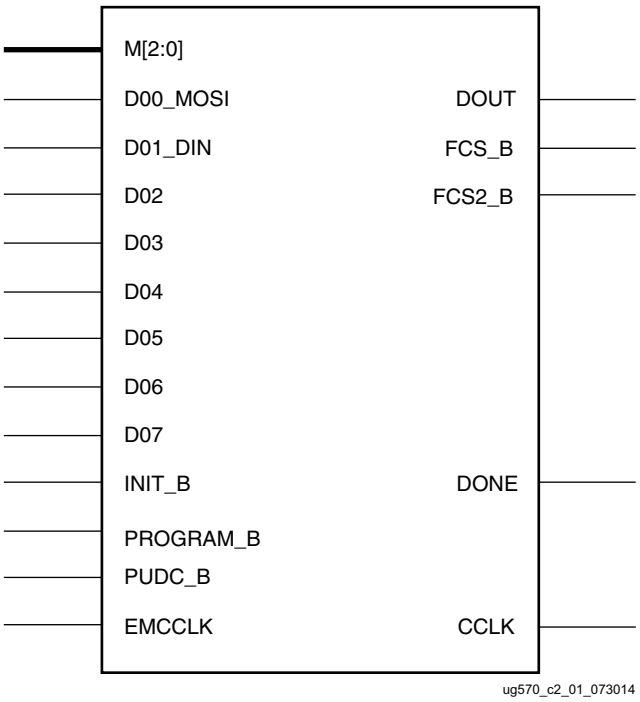
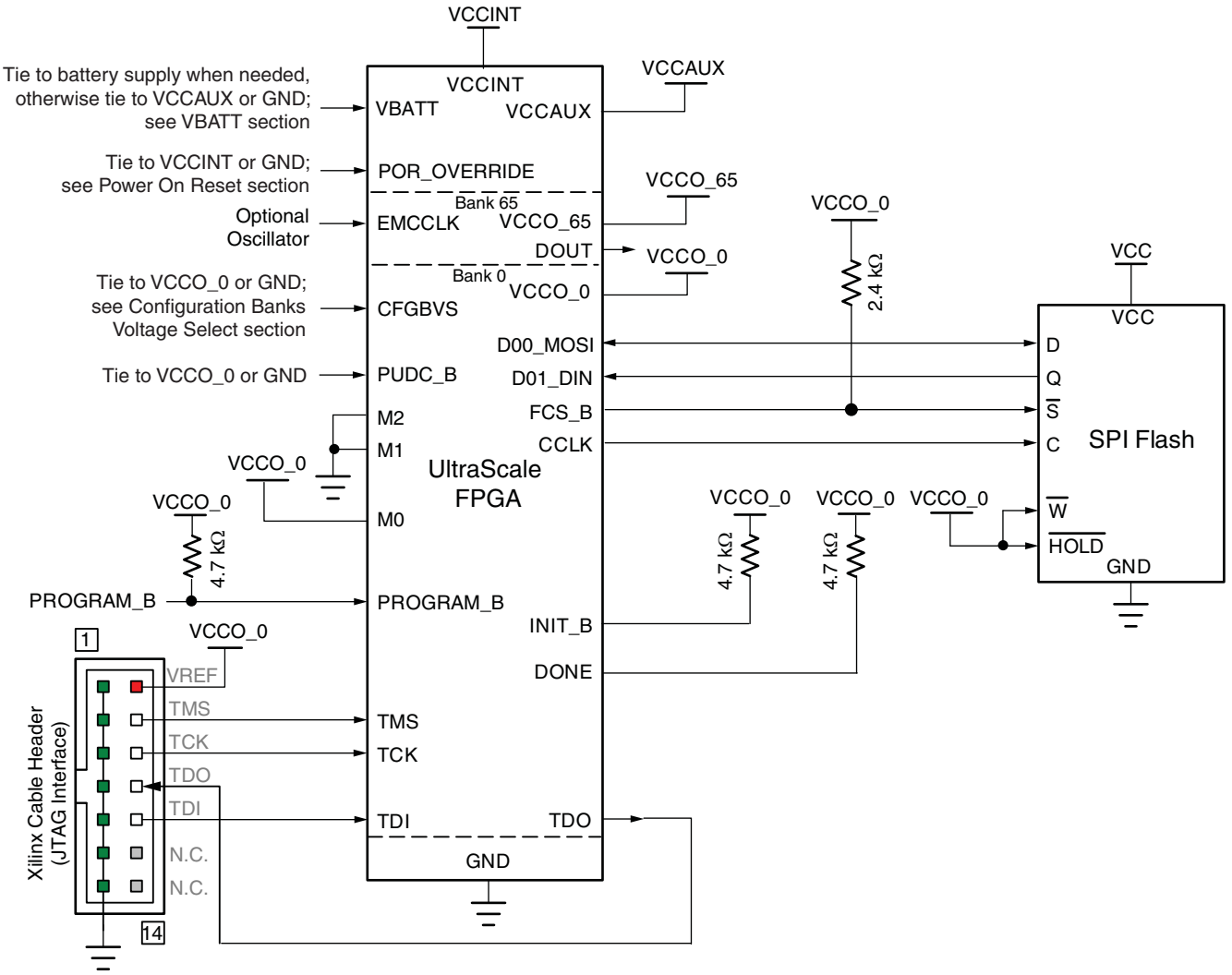


Figure 2-1: **SPI Configuration Interface**

Figure 2-2 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 D[00] pin as a dual-purpose Data In/Out pin. The data pins used only as FPGA inputs are shown as unidirectional in the figures, although in some cases they may be bidirectional before or after configuration. Daisy-chained configuration mode is only available in SPI x1 mode. The FPGA pin connections to the serial NOR flash involved in the master SPI mode are listed in Table 1-7, page 29.



Refer to the Notes following this figure for related information.

ug570_c2_02_031915

Figure 2-2: **SPI x1/x2 Configuration Interface Example**

Notes relevant to Figure 2-2:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required. See Table 1-9, page 31 for INIT_B signal details.
3. CCLK signal integrity is critical.
4. 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, x4, or x8 master SPI configuration modes.

5. 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.
6. The FPGA V_{CCO_0} supply must be compatible with the V_{CC} for the I/O of the flash device.
7. Data is clocked out of the flash on the CCLK falling edge and clocked in on the FPGA on the rising edge, unless negative edge clocking is enabled in the Vivado **Edit Device Properties** dialog.
8. The CCLK frequency is adjusted by the Vivado **Configuration Rate** bitstream setting (BITSTREAM.CONFIG.CONFIGRATE) if the source is the internal oscillator. Alternatively, the **Enable External Configuration Clock** option (BITSTREAM.CONFIG.EXTMASTERCCLK_EN) can switch the CCLK to source from the EMCCLK pin to use an external clock source. See EMCCLK Option, page 68 and File Generation, page 75 for details.
9. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. See Table 1-9, page 31 for PUDC_B signal details.

The Vivado tools provide control of configuration bitstream options through Tcl command line properties, and also provides support through a configuration dialog box. After loading a design, you can select **Tools** > **Edit Device Properties** to edit programming and configuration properties in the **Edit Device Properties** dialog box. For more details, see *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7].

The Vivado tools provide the ability to program a serial flash using an indirect programming method. This downloads a new FPGA design that provides a connection from the Vivado tools through the FPGA to the flash. Previous FPGA memory contents are lost during this operation. For the specific densities supported by the programming tools, consult UG908 [Ref 7].

For additional details on the SPI x1, x2, and x4 operation, including programming instructions, see *SPI Configuration and Flash Programming in UltraScale FPGAs* (XAPP1233) [Ref 11]. The SPI x1 mode sequence diagram is shown in Figure 2-3.

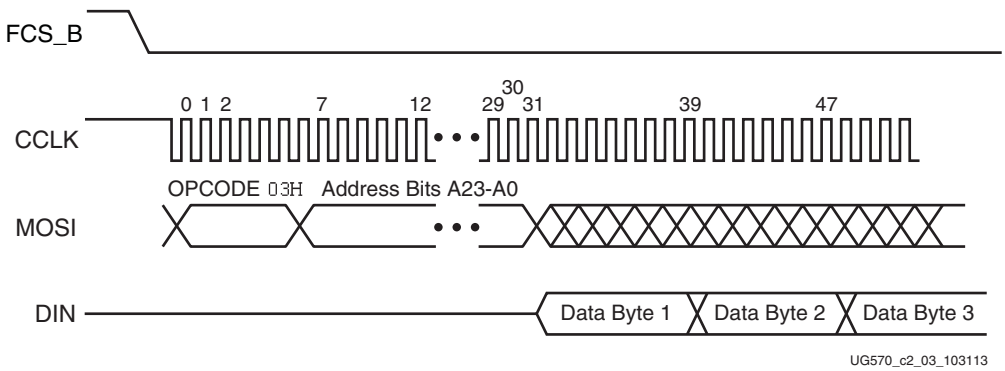


Figure 2-3: **UltraScale FPGA SPI x1 Mode Sequence**

Notes relevant to Figure 2-3:

1. Waveforms represent the relative sequence of events and are not to scale. See the flash memory data sheet for detailed SPI command and data timing.

Master SPI Read Commands

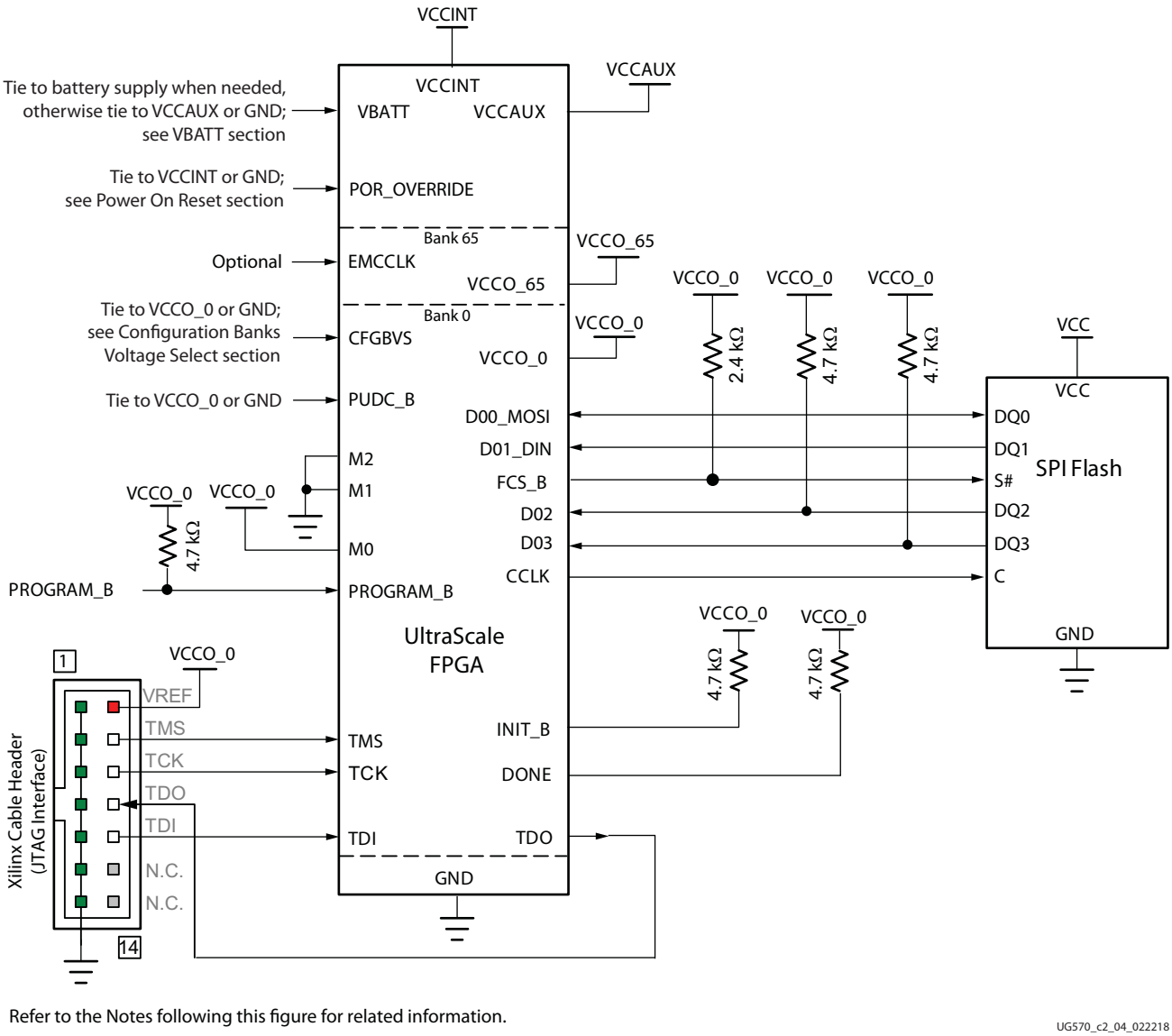
The master SPI configuration mode supports the read operations listed in Table 2-1. When starting the master SPI configuration mode, the FPGA transmits the Fast Read opcode (0Bh) with a 24-bit address of 0 to the flash. A bitstream loaded at address 0 in the primary flash can contain FPGA commands in the initial part of the bitstream that causes the configuration logic to issue one of the supported SPI instructions listed in Table 2-1. If the SPI command is for x1, x2, or x4 configuration, the FPGA then issues a new read command for the Fast Read (0Bh), Dual Output Fast Read (3Bh), or Quad Output Fast Read (6Bh) or the equivalent 32-bit address version (0Ch, 3Ch, or 6Ch respectively) to the primary flash. If the instruction starts the master SPI x8 configuration sequence, the Quad Output Fast Read (6Bh), or Quad Output Fast Read, 32-bit address (6Ch) read command is issued to both the primary and secondary flashes simultaneously. The Vivado Configuration Dialog programming tools are used to enable the new widths and 32-bit addressing commands in the bitstream.

Table 2-1: **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

Master SPI Quad (x4)

Xilinx UltraScale FPGAs support a x4 quad SPI master configuration width as shown in Figure 2-4.



Refer to the Notes following this figure for related information.

UG570_c2_04_022218

Figure 2-4: **Master SPI Quad (x4) Configuration Interface Example**

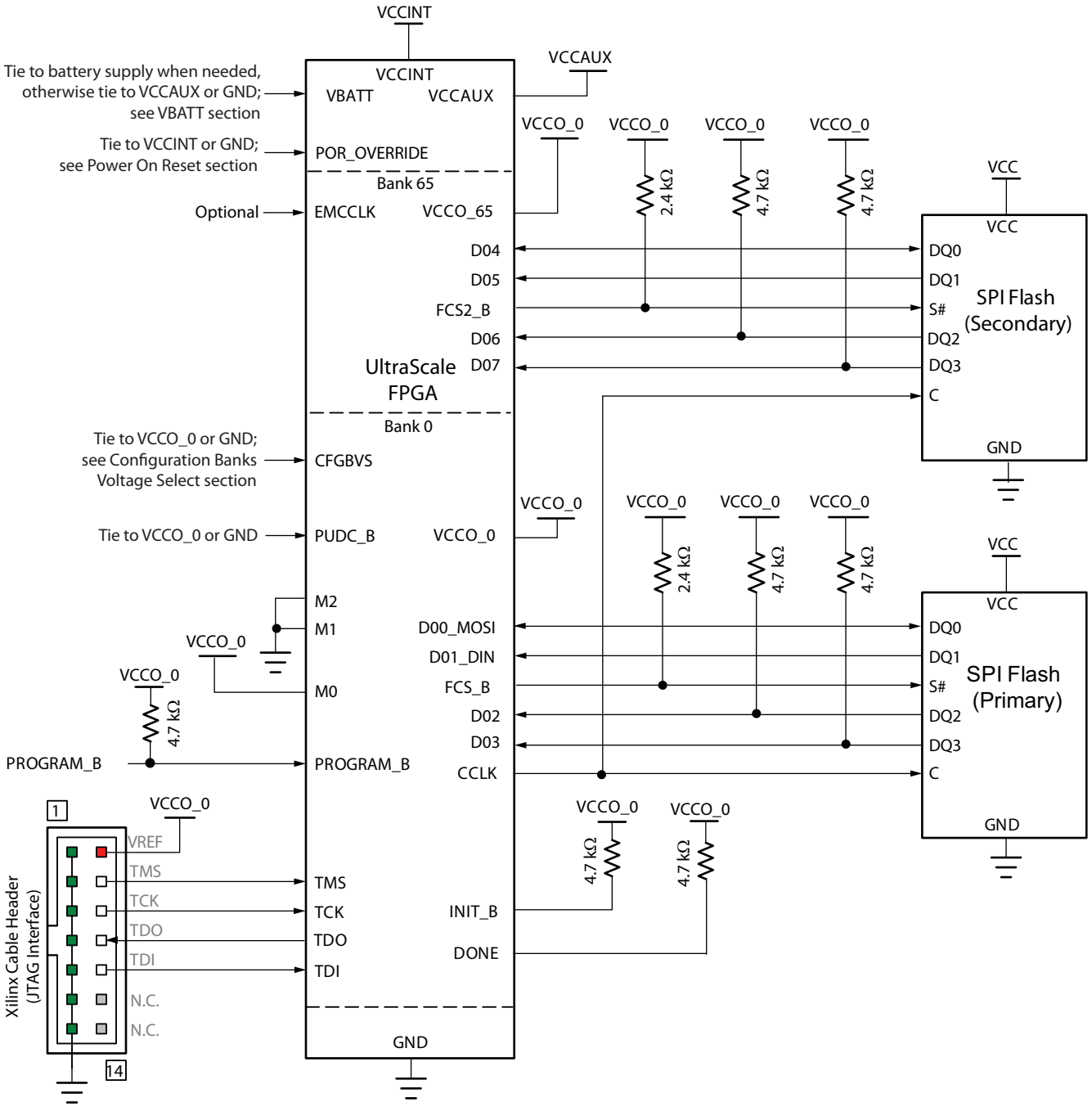
Notes relevant to Figure 2-4:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.

3. CCLK signal integrity is critical.
4. Daisy-chaining is not supported for x2, x4, or x8 master SPI configuration modes.
5. 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.
6. The FPGA V_{CCO_0} supply must be compatible with the V_{CC} for the I/O of the flash device.
7. Data is clocked out of the flash on the CCLK falling edge and clocked in on the FPGA on the rising edge, unless negative edge clocking is enabled in the Vivado **Configuration Dialog** box.
8. The CCLK frequency is adjusted by the Vivado **Configuration Rate** bitstream setting (BITSTREAM.CONFIG.CONFIGRATE) if the source is the internal oscillator. Alternatively, the **Enable External Configuration Clock** option (BITSTREAM.CONFIG.EXTMASTERCCLK_EN) can switch the CCLK to source from the EMCCLK pin to use an external clock source. See EMCCLK Option, page 68 and File Generation, page 75 for details.
9. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. See Table 1-9, page 31 for PUDC_B signal details.

Master SPI Dual Quad (x8)

Xilinx UltraScale architecture-based FPGAs introduce a master SPI x8 configuration width that uses two identical x4 SPI flash devices connected in a parallel fashion as shown in Figure 2-5. The clock, CCLK, is common for both flash devices with the select and data pins being separate, but identically driven when x8 mode is used.



Refer to the Notes following this figure for related information.

UG570_c2_05_022218

Figure 2-5: **Master SPI Dual Quad (x8) Configuration Interface Example**

Notes relevant to Figure 2-5:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.

3. CCLK signal integrity is critical.
4. Daisy-chaining is not supported for x2, x4, or x8 master SPI configuration modes.
5. 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.
6. The FPGA V_{CCO_0} supply must be compatible with the V_{CC} for the I/O of the flash.
7. Data is clocked out of the flash on the CCLK falling edge and clocked in on the FPGA on the rising edge, unless negative edge clocking is enabled in the Vivado Configuration Dialog.
8. The CCLK frequency is adjusted by the **Configuration Rate** option if the source is the internal oscillator. Alternatively, the **Enable External Configuration Clock** option can switch the CCLK to source from the EMCCLK pin to use an external clock source.
9. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. See Table 1-9, page 31 for PUDC_B signal details.

To generate a bitstream for x8 SPI mode, the bitstream should be generated with the property CONFIG_MODE to SPIx8. For x8 SPI configuration, the primary flash must contain the initial portion of a configuration bitstream that includes the x8 SPI configuration command. When the FPGA reads in this command, it will issue either Quad Output Fast Read (6Bh) or Quad Output Fast Read, 32-bit address (6Ch) simultaneously to both the primary and secondary flash memories. The secondary flash should contain dummy information that is equal in size to the initial portion of the bitstream in the primary flash. Beginning at the next address after the initial portion of the bitstream in the primary flash and after the dummy data in the secondary flash, the configuration bitstream is split evenly between the flash devices beginning with the first four bits in the primary flash and the next four bits in the secondary flash. The entire configuration bitstream will then be split between the two flash devices with the least significant nibble of each byte in the primary flash and the most significant nibble at the same address in the secondary flash.

The x8 SPI master configuration mode requires that the flash devices be identical and identically configured. For example, some flash devices have programmable latency or dummy cycles via nonvolatile configuration bits that may need to be set to allow high clock rates for the read commands. The latency cycles must be the same between the primary and secondary flash devices in order to maintain bit alignment.

Serial NOR Flash Densities over 128 Mb

Serial NOR flash densities over 128 Mb require more than the traditional 24-bit addressing that was standard before the introduction of 256 Mb and larger flashes. Flash vendors use various methods to support 32-bit addressing that may enable the 24-bit read commands to operate as a 32-bit read command. For example, a nonvolatile bit might be set in the flash that causes the flash to expect four address bytes after a 0Bh command. These methods should not be enabled for the flash devices used to configure UltraScale FPGAs.

The solution supported by the UltraScale FPGAs requires the flash to boot up in a 24-bit addressing mode for the 0Bh, 3Bh, and 6Bh commands and 32-bit addressing for the 0Ch, 3Ch, and 6Ch commands. The Vivado tool **Edit Device Properties** dialog box provides the option to enable 32-bit addressing. To generate a bitstream for flash densities over 128 Mb the property BITSTREAM.CONFIG.SPI_32BIT_ADDR should be set to Yes. See *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for details. Valid flash devices must support the instructions in Table 2-1 (SPI Instructions and Required Opcodes) to interface with the UltraScale FPGAs.

Multi-die Serial NOR Flash Devices

Some serial NOR flash vendors reach larger densities by stacking die in the same package. For use with UltraScale architecture-based FPGAs, these types of devices must offer a transparent interface and read behavior to the FPGA. UltraScale architecture-based FPGAs do not support single flash devices that use multiple select pins. UltraScale FPGAs also issue a single SPI read command to read in an entire bitstream. If a flash does not read seamlessly with a single command across any die boundaries, it cannot be used to store a configuration bitstream if the bitstream crosses the die boundary.

SPI Configuration Timing

Serial NOR flash devices clock data out on the falling edge and by default, the UltraScale FPGAs clock data in on the rising edge. This results in a lost half cycle that limits the maximum clock speed of the configuration solution (Figure 2-6). To gain maximum use of the clock period, the FPGA can be configured to clock data in on the falling edge.

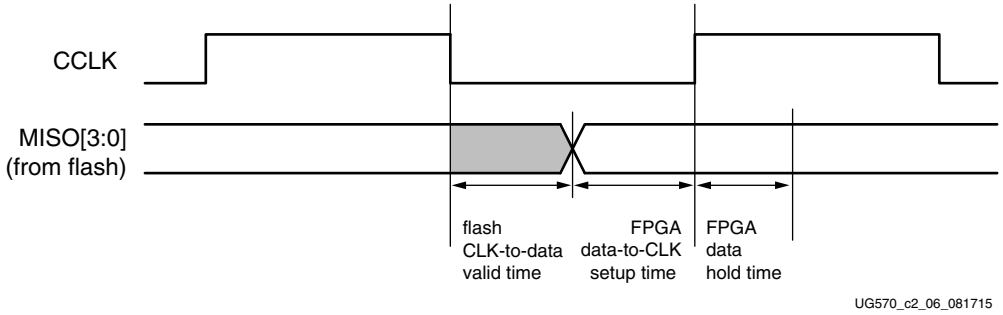


Figure 2-6: **Basic SPI Configuration Interface**

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 in the Vivado tool **Edit Device Properties** dialog box (BITSTREAM.CONFIG.SPI_FALL_EDGE Yes).

Determining the Maximum Configuration Clock Frequency

In master SPI mode, the FPGA delivers the configuration clock. The FPGA master configuration clock frequency is set through the Vivado tool **Edit Device Properties** dialog box. The configuration rate option sets the nominal configuration clock frequency.

The configuration rate setting can be increased for a faster configuration time, if the timing requirements discussed in this section are satisfied. When determining a valid configuration rate setting, these timing parameters must be considered:

- FPGA nominal master CCLK frequency (configuration rate setting)
- FPGA master CCLK frequency tolerance ($F_{MCCKTOL}$)
- SPI clock low to output valid (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 has been enabled in the Vivado tool **Edit Device Properties** dialog box.

The FPGA master configuration clock has a tolerance of $F_{MCCKTOL}$. Due to the master configuration clock tolerance ($F_{MCCKTOL}$), the Vivado tool **Edit Device Properties** dialog box configuration rate 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 low to output valid, and FPGA setup time, as shown in Equation 2-1.

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

The frequency tolerance of the FPGA master configuration clock can be a significant factor in this calculation at higher CCLK rates. 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 Vivado tool **Edit Device Properties** dialog box.

Power-on Sequence Precautions

At power-on, the FPGA automatically starts its configuration procedure. When the FPGA is in SPI configuration mode, the FPGA asserts FCS_B Low to select the flash and drives a read command to the flash. The 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 flash or because the FPGA and flash can respond at different times along the ramp of a shared power supply, special attention to the FPGA and flash power-on sequence or power-on ramps is essential. Refer to Power-On Sequence Precautions for Flash in Chapter 1.

Additional Information

For step-by-step instructions for using the SPI configuration mode with serial NOR flash, see *SPI Configuration and Flash Programming in UltraScale FPGAs* (XAPP1233) [Ref 11].

For examples of how to use the SPI flash after configuration, to store non-volatile user data or to remotely update configuration images, see *UltraScale FPGA Post-Configuration Access of SPI Flash Memory using STARTUPE3* (XAPP1280) [Ref 12].

Serial Configuration Mode

Introduction

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 3-1 shows the basic serial configuration interface.

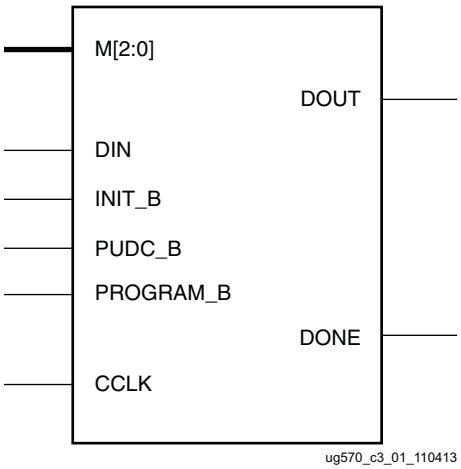
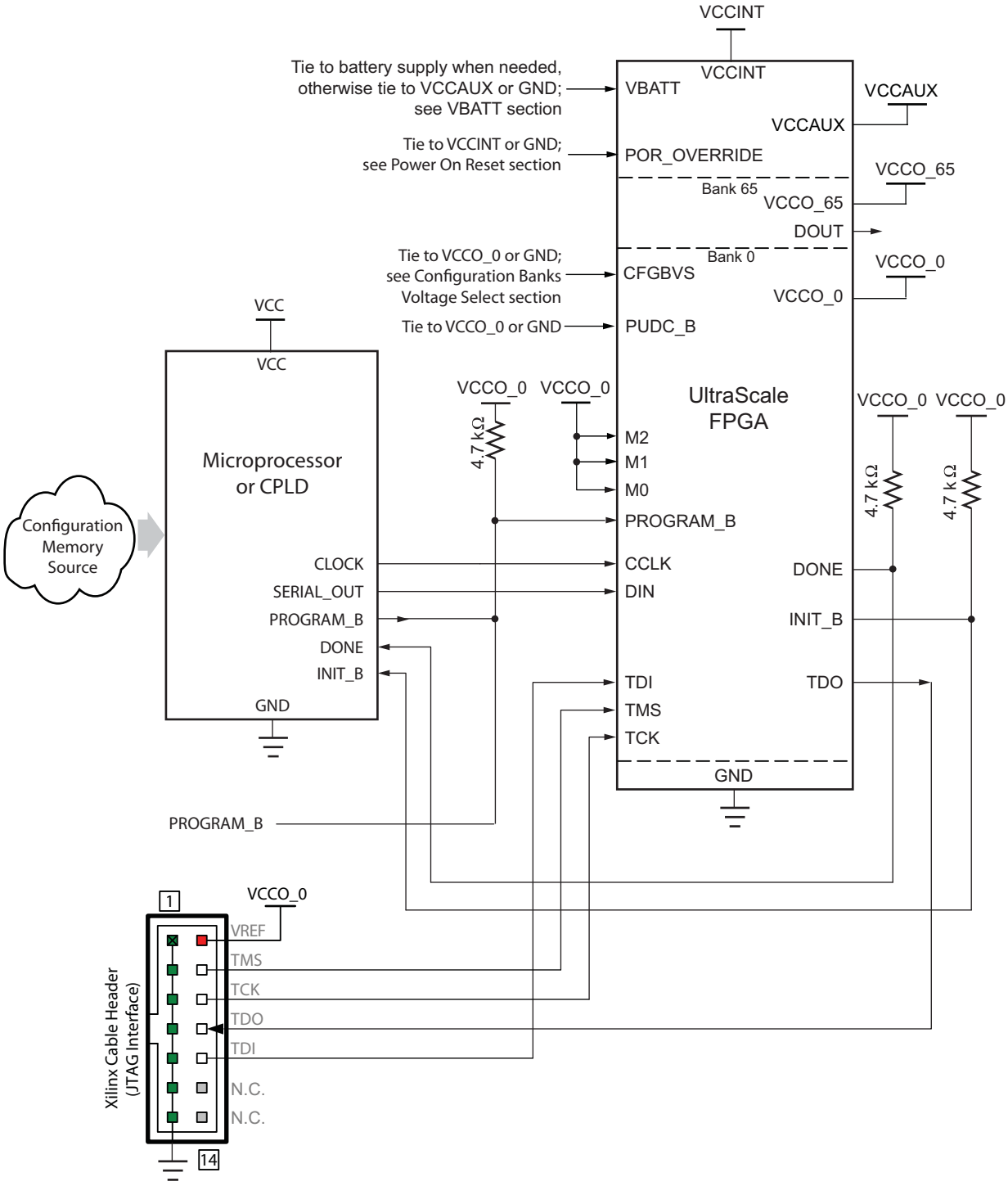


Figure 3-1: **Serial Configuration Interface**

The serial configuration interface pins shown in Figure 3-1 are defined in Table 1-9, page 31.

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 3-2). 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).



Refer to the Notes following this figure for related information.

ug570_c3_02_022117

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

Notes relevant to Figure 3-2:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required. See Table 1-9, page 31 for INIT_B signal details.
3. CCLK signal integrity is critical.
4. See the respective data sheet ([Ref 8] or [Ref 9]) for the V_{CCINT}, V_{CCAUX}, and V_{CCO_0} supply voltages.
5. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. See Table 1-9, page 31 for PUDC_B signal details.

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. Master serial mode is not recommended for new designs.

The Kintex UltraScale and Virtex UltraScale 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. The Kintex UltraScale+ and Virtex UltraScale+ FPGAs do not support master serial mode. Xilinx Platform Flash PROMs do not support UltraScale architecture-based FPGAs.

RECOMMENDED: *The alternative master SPI mode is the dominant configuration mode for a low-pin count configuration from a serial-type flash device. Master serial mode is not recommended for new designs. See Differences Between UltraScale FPGA Families, page 11.*

Clocking Serial Configuration Data

Figure 3-3 shows how configuration data is clocked into FPGAs in slave serial and master serial modes.

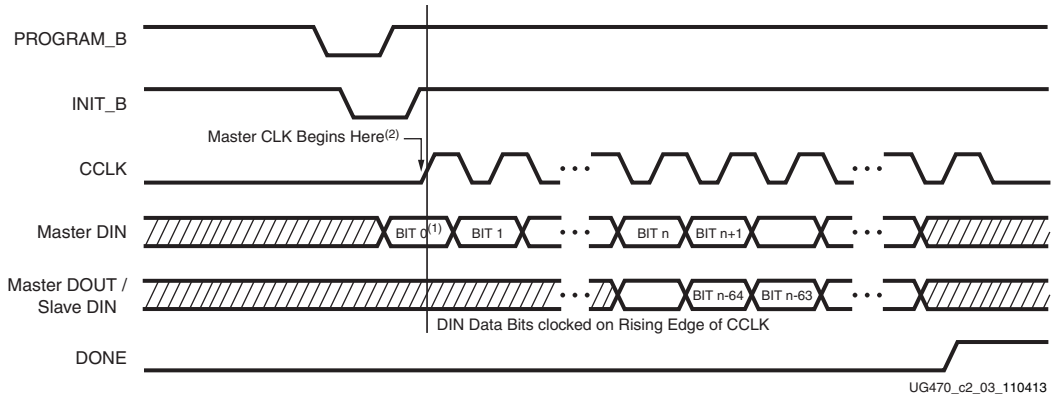


Figure 3-3: **Serial Configuration Clocking Sequence**

Notes relevant to Figure 3-3:

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 serial configuration mode, CCLK is driven only after INIT_B goes High to shortly after DONE goes High. Otherwise CCLK is in a high-impedance state.
3. CCLK can be free-running in slave serial mode.

XILINX

Chapter 4

Master BPI Configuration Mode

Introduction

The UltraScale™ architecture-based FPGAs master BPI configuration mode enables the use of high speed industry-standard parallel NOR flash devices for bitstream storage. The FPGA supports a direct connection to the address, data, and control signals of a parallel NOR flash for extracting a stored design image bitstream.

Master BPI Interface

The master BPI configuration mode supports parallel NOR flash read options: x16 synchronous and x8/x16 asynchronous. By default, the master BPI configuration mode uses the x8 asynchronous parallel NOR flash read option. For applications that require faster configuration times, the x16 data bus width with synchronous read option should be enabled as described in File Generation, page 75.

The master BPI configuration interface is represented in Figure 4-1. Detailed connections between the FPGA and the parallel NOR flash for master BPI configuration mode are shown in Figure 4-2 and Figure 4-4. The FPGA signals are defined in Table 1-9, page 31.

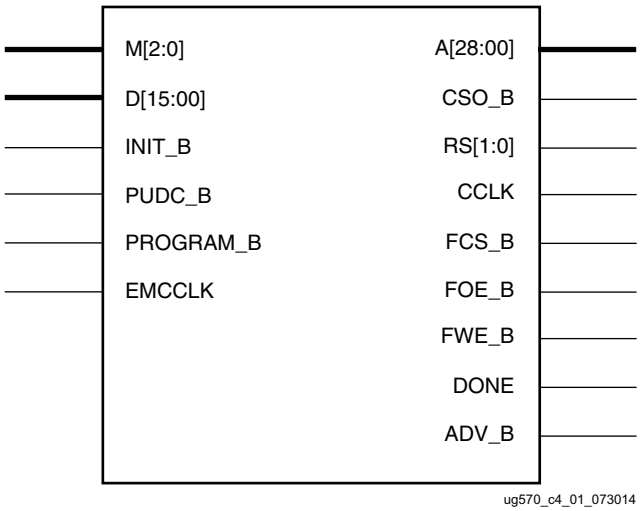


Figure 4-1: **Master BPI Configuration Mode Interface**

Parallel NOR flash is a popular option for storing and delivering the bitstream because the wide x16 data bus provides faster configuration over other flash alternatives. In addition to the faster configuration, systems that use parallel NOR flash memory for random-access, nonvolatile application data storage can also benefit from consolidating the configuration storage into a single memory device.

When choosing a parallel NOR flash for the configuration storage several factors should be considered:

- The storage capacity required by the application (current and migration options)
- The data bus width options for reduced configuration time
- The flash I/O voltage range

Refer to Table 1-4, page 21 for information on the bitstream size in order to determine the minimum flash density required for configuration. The configuration pins in bank 0 and the multi-purpose pins in bank 65 are used by the master BPI configuration mode interface and must receive the same V_{CCO} voltage and be compatible with the parallel NOR flash I/O specification. The flash data sheet should be reviewed carefully to ensure the feature and voltage requirements are supported.

Parallel NOR flash examples are provided in this section for Synchronous and Asynchronous read options. See *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for details for flash families that are supported and can be indirectly programmed using Vivado® tool device programmer.

Table 4-1 provides an overview of the UltraScale architecture-based FPGAs feature support with the flash read options. Refer to Master BPI Synchronous Read, page 64 and Master BPI Asynchronous Read, page 68 for details.

Table 4-1: **Asynchronous vs Synchronous Read Option Comparison**

UltraScale FPGA Feature Support	Synchronous Read	Asynchronous Read	Asynchronous Page Read
Data Bus Width	x16 only	x8 or x16	x8 or x16
Multiple FPGA Daisy Chain	Yes	Yes	No
Ganged FPGA Mode	No	Yes	No
Wraparound Error	No	Yes	Yes
Watchdog Timeout	Yes	Yes	Yes
Fallback	Yes	Yes	Yes
MultiBoot	Yes	Yes	Yes
Encryption	Yes	Yes	Yes
Compression	Yes	Yes	Yes

Master BPI Synchronous Read

Note: The master BPI synchronous read mode is not recommended for new designs. Xilinx recommends contacting your flash supplier for product availability.

The UltraScale™ architecture-based FPGAs master BPI configuration mode can read a bitstream from select parallel NOR devices that support burst, synchronous reads. The master BPI configuration mode with synchronous read is the fastest direct flash configuration option for UltraScale architecture-based FPGAs without the need for customized external control logic.

Figure 4-2 provides the connectivity diagram between the FPGA and parallel NOR flash for the master BPI configuration mode to support the synchronous read and the EMCCLK (external master configuration clock). Refer to the External Master Configuration Clock (EMCCLK) Option in Chapter 1 for configuration clock option details. Figure 4-2 supports both synchronous and asynchronous read modes. If only asynchronous mode is required for the application, refer to Figure 4-4 for connections that are optional.

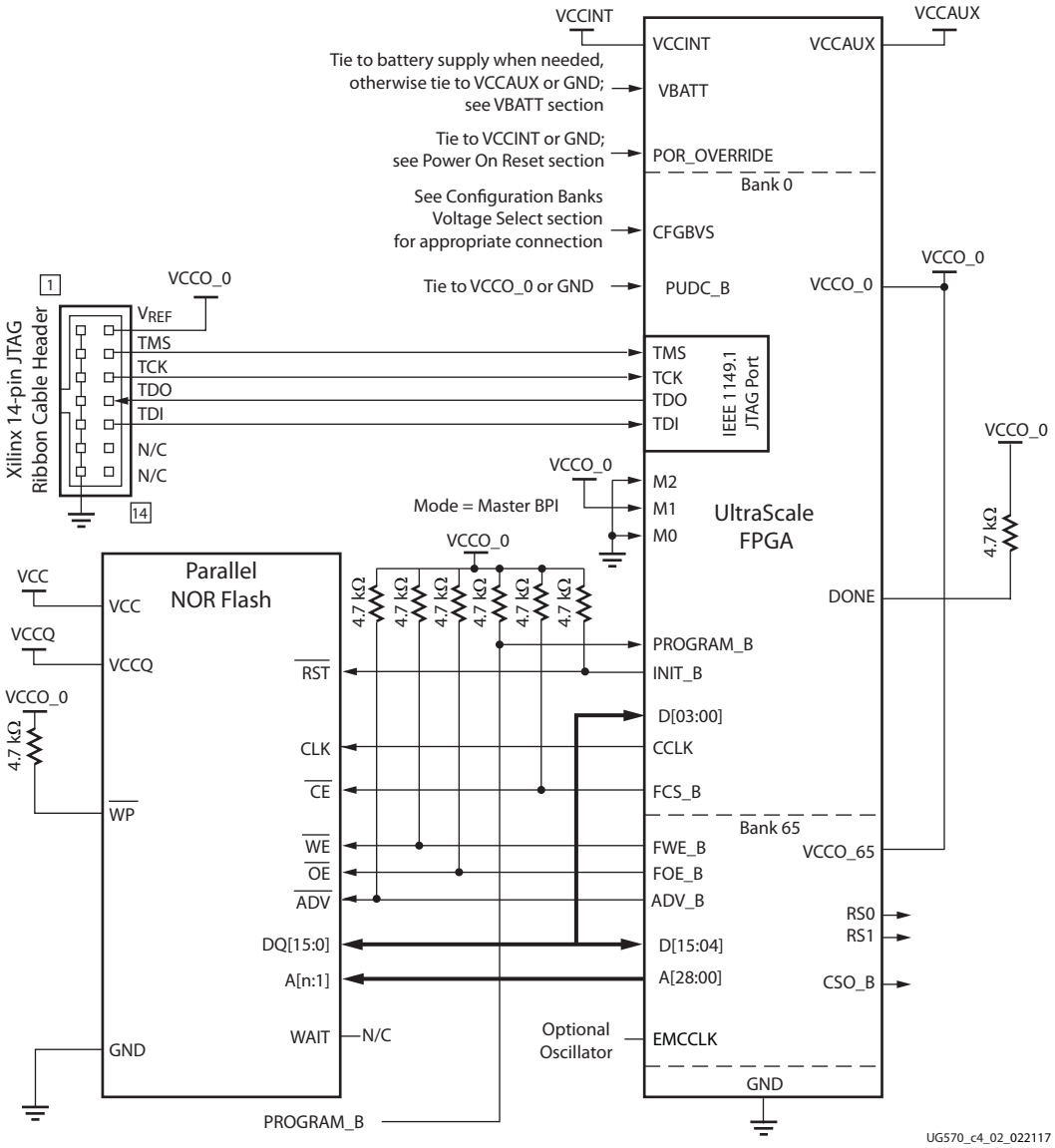


Figure 4-2: **Master BPI Configuration Interface Example for x16 Synchronous Read**

Notes relevant to Figure 4-2:

IMPORTANT: *Review the flash vendor's data sheet carefully to ensure that the flash LSB address signal is connected to the FPGA LSB address signal A[00].*

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required. See Table 1-9, page 31 for INIT signal details.
3. CCLK signal integrity is critical.

4. For the synchronous read example the x16 data bus interface is supported. x8 data bus interface is only supported in asynchronous read mode.
5. CSO_B should be connected to the CSI_B of the downstream FPGA for parallel daisy-chains.
6. The FPGA V_{CCO_0} supply must be compatible with the supply voltage for the I/O of the selected parallel NOR device.
7. The CCLK frequency is adjusted by the Vivado **Configuration Rate** bitstream setting (BITSTREAM.CONFIG.CONFIGRATE) if the source is the internal oscillator. Alternatively, the **Enable External Configuration Clock** option (BITSTREAM.CONFIG.EXTMASTERCCLK_EN) can switch the CCLK to source from the EMCCLK pin to use an external clock source. See EMCCLK Option, page 68 and File Generation, page 75 for details.
8. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. See Table 1-9, page 31 for PUDC_B signal details.
9. See the respective data sheet ([Ref 8] or [Ref 9]) for the V_{CCINT}, V_{CCAUX}, and V_{CCO_0} supply voltages.
10. The ADV_B and CCLK connections are required for synchronous read operation, but these connections to the flash are optional for asynchronous read mode. 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. On setups only targeting asynchronous read, the flash ADV_B and CLK lines can be tied to GND.
11. The RS[1:0] pins are not connected, as shown in Figure 4-2. This sample schematic supports single bitstream configuration. These output pins are optional and can be used for MultiBoot configuration.
12. The JTAG connections are shown for a simple, single-device JTAG scan chain. When multiple devices are on the JTAG scan chain, use the proper IEEE Std 1149.1 daisy-chain technique to connect the JTAG signals. The TCK signal integrity is critical for JTAG operation. Route, terminate, and if necessary, buffer the TCK signal appropriately to ensure signal integrity for the devices in the JTAG scan chain.

Synchronous Read Sequence

Figure 4-3 shows the sequence to initiate the BPI configuration synchronous read. The master BPI mode sequence occurs automatically after power-up when the bitstream has been generated with the synchronous read option and the mode pins are set to M[2:0] = `010`.

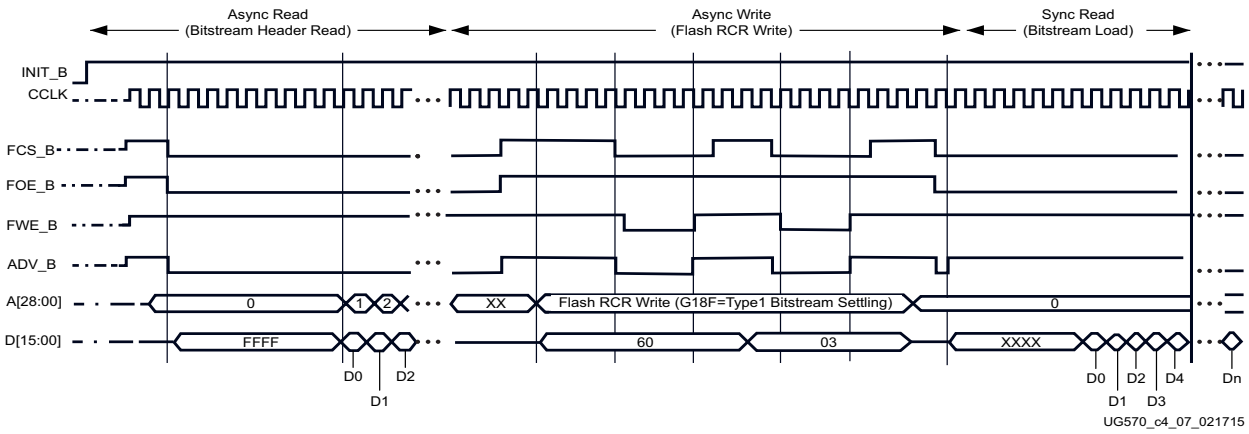


Figure 4-3: **Master BPI Configuration Mode Synchronous Read Waveform**

First, the FPGA reads the bitstream asynchronously to determine the targeted read option. The read always starts at the default internal CCLK rate. After the INIT_B signal is released and the control signals FCS_B, FOE, and ADV_B are asserted with a valid address A[28:00] then data is captured from the parallel NOR flash on the data bus D[15:0]. The FPGA reads the bitstream header to determine the flash read option selected for reading the configuration data. When a synchronous command is read in the bitstream header, the FPGA configuration controller initiates an asynchronous write to the Read Configuration Register (RCR) of the connected parallel NOR flash.

Next, the FPGA writes the flash RCR synchronous and latency bits to enable a flash synchronous read. To perform the asynchronous write operation, the FPGA asserts the FCS_B and FWE_B while the INIT_B and FOE_B are deasserted. The FPGA issues the Flash Configuration register write sequence of two write cycles. The first cycle has the Read Configuration Register (RCR) data on A[16:01] and command `0x60` on the data bus. The second cycle has the RCR data on A[16:01] and the command `0x03` on the data bus. The RCR values are different for the different flash families and are determined by the bitstream options, described in File Generation, page 75.

Lastly, the FPGA switches from the asynchronous read to synchronous read protocol and reinitiates the bitstream read. This sequence is implemented by the FPGA asserting the FCS_B and the FOE_B signals and having ADV_B asserted for one cycle with a valid address. The configuration data is then burst from the flash and read back by the FPGA. Once the header information is read, the configuration clock source can change to the user selection.

IMPORTANT: *It is important to understand that the flash is left in the same read mode that is used for configuration. For example, the flash is left in synchronous read mode after the FPGA is configured in the synchronous read mode.*

EMCCLK Option

By default, the master BPI configuration mode uses 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 (FMCCKTOL). UltraScale FPGAs support the ability to dynamically switch to an external clock source (EMCCLK) when in master BPI mode. For more details, see External Master Configuration Clock (EMCCLK) Option in Chapter 1.

Master BPI Asynchronous Read

By default, UltraScale FPGAs use the parallel NOR flash asynchronous read in the master BPI configuration mode. The FPGA drives the address bus from a given start address, and the flash sends back the bitstream data. The default start address is address 0, but the start address can be explicitly set in a MultiBoot reconfiguration procedure. In asynchronous read mode, supported bus widths of x8 and x16 are auto-detected. Figure 4-4 provides the connectivity diagram between the FPGA and parallel NOR flash for the x16 asynchronous read master BPI configuration mode.

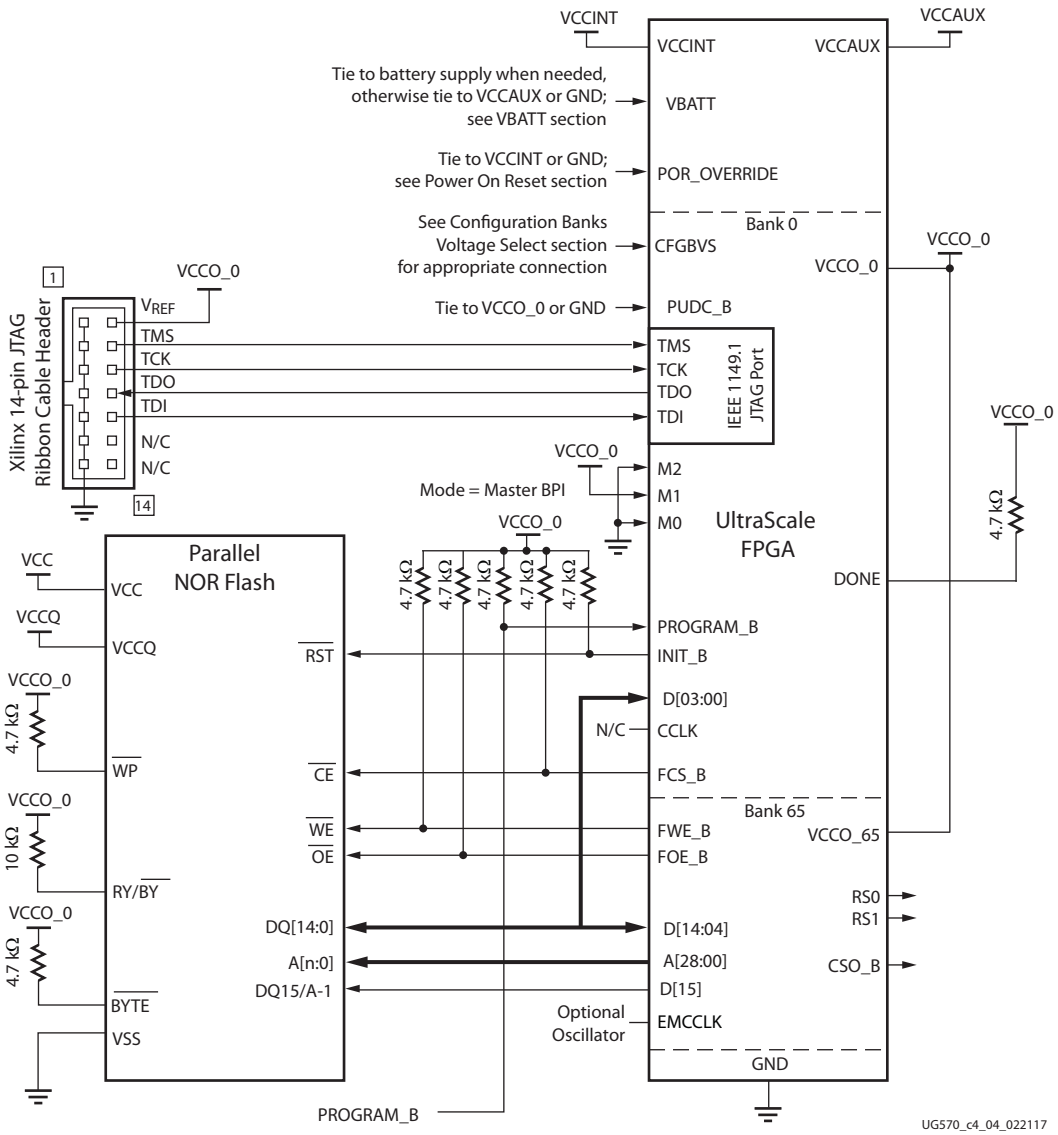


Figure 4-4: **Master BPI Configuration Interface Example for x16 Asynchronous Read**

Notes relevant to Figure 4-4:

1. Parallel NOR flash that have a BYTE# signal must set the BYTE# signal appropriately. For x16 data bus width the BYTE# signal must be set High. For x8 data bus width the BYTE# signal must be set Low. Refer to the flash data sheet for details.
2. Review the flash vendor's data sheet carefully to ensure that the flash LSB address signal is connected correctly depending on the vendor and data bus width used. Parallel NOR flash with the dual purpose DQ15/A-1 signal must ensure that it is connected properly. The DQ15/A-1 is a data pin in x16 mode. For the x8 mode the flash DQ15/A-1 is an LSB address line and needs to be connected to the FPGA A00..
3. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.

4. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required. See Table 1-9, page 31 for INIT signal details.
5. The x16 BPI interface is shown in Figure 4-4. For x8 BPI interfaces, only D[07:00] are used.
6. The flash vendor data sheet should be referred to for flash signal connectivity details. Ensure the FPGA LSB A00 is aligned to the flash LSB address (dependent on the flash family and data width selected).
7. CSO_B should be connected to the CSI_B of the downstream FPGA for parallel daisy-chains.
8. The FPGA V_{CCO_0} supply must be compatible with the V_{CC} for the I/O of the selected parallel NOR device.
9. The CCLK frequency is adjusted by the Vivado **Configuration Rate** bitstream setting (BITSTREAM.CONFIG.CONFIGRATE) if the source is the internal oscillator. Alternatively, the **Enable External Configuration Clock** option (BITSTREAM.CONFIG.EXTMASTERCCLK_EN) can switch the CCLK to source from the EMCCLK pin to use an external clock source. See EMCCLK Option, page 68 and File Generation, page 75 for details.
10. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. See Table 1-9, page 31 for PUDC_B signal details.
11. See the respective data sheet ([Ref 8] or [Ref 9]) for the V_{CCINT}, V_{CCAUX}, and V_{CCO_0} supply voltages.
12. ADV_B and CCLK connections are available on some supported flash families, but the connections are not required for asynchronous read operation. 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. On asynchronous read setups, if the flash has ADV_B and CLK lines, they can be tied to GND.
13. The RS[1:0] pins are not connected, as shown in Figure 4-4. This sample schematic supports single bitstream configuration. These output pins are optional and can be used for MultiBoot configuration.
14. 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.

Asynchronous Read Sequence

The sequence for a successful master BPI configuration with asynchronous read is shown in Figure 4-5. After power-up, Mode pins M[2:0] are sampled when the FPGA INIT_B output goes High. If the master BPI configuration mode (M[2:0] = `010`) 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 required to be connected to the parallel NOR flash device for asynchronous read, the FPGA outputs an address after the rising edge of CCLK, and the data is still sampled on the next rising edge of CCLK. In the master BPI mode with asynchronous read, 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), a wraparound error flag is raised in the Status register, and fallback reconfiguration starts.

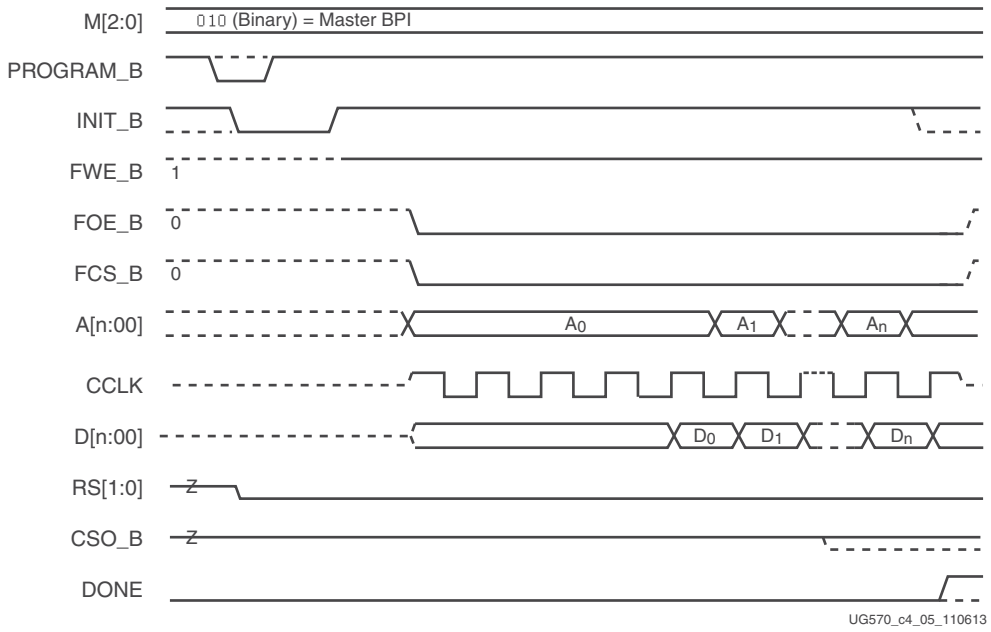


Figure 4-5: **Master BPI Configuration Mode Asynchronous Read Waveform**

Asynchronous Page Read Support

In addition to the basic asynchronous read support, the UltraScale FPGA supports parallel NOR asynchronous page reads. The asynchronous page reads enable faster configuration clock frequency than in the basic asynchronous mode. This read option is popular for parallel NOR flash devices that are not supported by the UltraScale FPGA synchronous read option.

In the asynchronous page read sequence, the first word read from a multiword page takes a standard asynchronous read time and a subsequent read of another word from the same page takes significantly less time. The sequence of the page read operation is controlled by the FPGA bitstream. The generation of a bitstream with page read support requires the setting of multiple bitstream properties to take advantage of page read and maximize the CCLK frequency. The page size bitstream property determines the number of words in each page. Words other than the first word of each page are read in one master CCLK cycle. The first read cycle bitstream property determines the number of CCLK cycles that are allotted

to reading the first word of each page. Refer to File Generation, page 75 for details on how to set the page size and first read cycle.

After an FPGA reset, the default page size is 1, the first access CCLK is 1, and the master CCLK is running at the slowest default frequency. The configuration register (COR1) contains parallel NOR flash page read control bits. After the COR1 register is programmed, the BPI address timing switches at the page boundary as shown in Figure 4-6. When the command is received, the master CCLK switches to a user-desired frequency, using it to load the rest of the configuration.

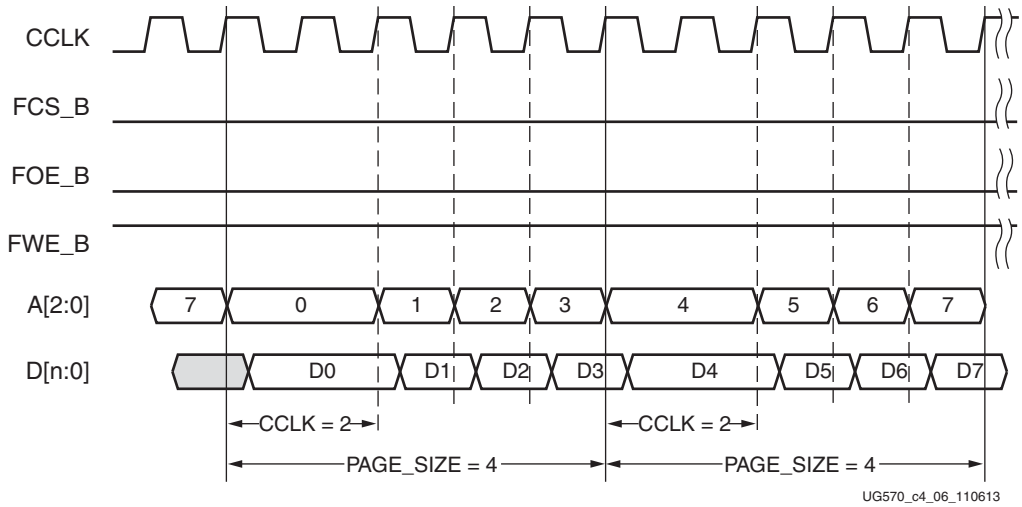


Figure 4-6: **Master BPI Configuration Mode Page Read Waveform (Page Size=4, First Access CCLK=2)**

Configuration Time

Synchronous Read

For the fastest parallel NOR flash configuration time, use the master BPI Configuration mode synchronous x16 read option with the EMCCLK. There are several system factors that must be considered when you determine the maximum configuration clock rate for synchronous reads in your application. These following parameters should be considered:

- Flash clock to out (T_{CHQV})
- FPGA data setup time (T_{BPIDCC})
- External master configuration clock frequency (EMCCLK Rate) or FPGA nominal master CCLK frequency (configuration Rate)
- External master configuration clock frequency tolerance (EMCCLK Tolerance) or FPGA master CCLK frequency tolerance (FMCCKTOL)

The following example utilizes the EMCCLK. The parallel NOR flash clock-to-out and the FPGA setup data sheet specifications are used to determine the maximum EMCCLK frequency. Board trace delay is also another factor that should be considered. An estimation for the maximum BPI Fast Configuration EMCCLK can be calculated with Equation 4-1 and must be less than the supported EMCCLK frequency (FEMCCK) specified in the FPGA data sheet.

$$MaxFreq = \frac{1}{FlashClockToOut(T_{CHQV}) + FPGADataSetup(T_{BPIDCC}) + BoardDelay} \quad \text{Equation 4-1}$$

For an application targeting a supported parallel NOR flash with a clock-to-out specification of T_{CHQV} = 5.5 ns and an FPGA data setup of T_{BPIDCC} = 3.5 ns, under the best case with EMCCLK clock tolerance and board delay negligible would be approximately 111 MHz which is less than the specified EMCCLK frequency (FEMCCK).

Asynchronous Page Read

The master BPI mode asynchronous page read gives you faster configuration times than the basic asynchronous read but not as fast as using the synchronous read. With page reads the first word read from a multiword page takes a standard asynchronous read time, but the subsequent read of another word from the same page takes significantly less time. To determine the fastest configuration clock rate for your page read implementation the following timing parameters should be considered:

External master configuration clock frequency (EMCCLK Rate) or FPGA nominal master CCLK frequency (configuration Rate)

- External master configuration clock frequency tolerance (EMCCLK Tolerance) or FPGA master CCLK frequency tolerance (FMCCKTOL)
- FPGA CCLK rising edge to address valid (T_{BPICCO})
- Parallel NOR flash address to output valid (access) time (T_{ACC})
- Parallel NOR flash page address to output valid (access) time (T_{APA})
- FPGA data setup time (T_{BPIDCC})
- Parallel NOR flash, number of words per page (BPI page size)
- FPGA number of CCLK cycles for the first word read (BPI first read cycle)

The BPI Page Size option is set to the number of words in a page, as defined by the Parallel NOR flash data sheet. Because the Parallel NOR flash has different timing for the read of the first word of a page, and for the subsequent read of a word from the same page, two timing checks are required in order to ensure the validity of the Configuration Rate and First Read Cycle attribute values.

First, the Configuration Rate setting must be checked. The period for the worst-case (fastest) master CCLK frequency must be greater than the sum of the FPGA address valid time, flash page access time, and FPGA setup time, as shown in Equation 4-2.

$$\frac{1}{ConfigRate \times (1 + FMCCKTOL_{MAX})} \geq T_{BPICCO} + T_{APA} + T_{BPIDCC} \quad \textit{Equation 4-2}$$

Second, the First Read Cycle must be checked. The First Read Cycle option specifies the number of FPGA CCLK cycles allocated for the reading of the first word of each page. The duration of the first read cycle is equivalent to the period of one CCLK cycle multiplied by the First Read Cycle option value. The worst-case allocated duration of the first read cycle must be greater than the sum of the FPGA address valid time, Parallel NOR flash read access time, and FPGA setup time, as shown in Equation 4-3.

$$\frac{BPIFirstReadCycle}{ConfigRate \times (1 + FMCCKTOL_{MAX})} \geq T_{BPICCO} + T_{ACC} + T_{BPIDCC} \quad \textit{Equation 4-3}$$

Asynchronous Read

The master BPI mode asynchronous read is the simplest parallel NOR flash setup, and significantly slower configuration times than the other read options. For the asynchronous read calculation, the following parameters must be considered:

- External master configuration clock frequency (EMCCLK Rate) or FPGA nominal master CCLK frequency (Configuration Rate)
- External master configuration clock frequency tolerance (EMCCLK Tolerance) or FPGA master CCLK frequency tolerance (FMCCKTOL)
- FPGA CCLK rising edge to address valid (T_{BPICCO})
- Parallel NOR flash address to output valid (access) time (T_{ACC})
- FPGA data setup time (T_{BPIDCC})

Equation 4-4 is the basic calculation to determine the configuration clock rate or EMCCLK rate:

$$\frac{1}{ConfigRate \times (1 + FMCCKTOL_{MAX})} \geq T_{BPICCO} + T_{ACC} + T_{BPIDCC} \quad \textit{Equation 4-4}$$

Power-on Sequence Precautions

At power on, a race condition between the FPGA and parallel NOR flash can exist because they can be supplied by different power rails or because they can respond at different times along the ramp of a shared power supply. Special attention to the FPGA and parallel NOR

flash power-on sequence or power-on ramps is essential. The parallel NOR flash interface signals are within FPGA dedicated bank 0 and I/O bank 65.

The FPGA sends the address to the parallel NOR flash to acquire the bitstream after the FPGA has completed its power-on reset sequence. The parallel NOR flash is not ready to receive an address until the parallel NOR flash power-on reset sequence has completed. Under specific conditions when the V_{CC} power supply to the parallel NOR flash powers up after the FPGA V_{CCINT} and V_{CCAUX} power supplies, the FPGA address counter can pass the critical start of the bitstream within the parallel NOR flash before the flash becomes responsive. The system must be designed such that the parallel NOR flash is ready to receive the address before the FPGA sends the address. For more details, see Power-On Sequence Precautions for Flash in Chapter 1.

File Generation

To implement a master BPI configuration mode solution you must create a design bitstream, then convert the bitstream into a flash programming file, and finally program the parallel NOR flash device. The following key properties should be reviewed when generating a bitstream for the master BPI configuration mode. These properties are also available through the Vivado tool **Edit Device Properties** dialog box. Refer to *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for more details.

The external master configuration clock (EMCCLK) property (BITSTREAM.CONFIG.EXTMASTERCCLK_EN) can be used in master modes to get a more precise configuration clock. The bitstream property must be set to enable the EMCCLK and to set the desired divider option. See External Master Configuration Clock (EMCCLK) Option in Chapter 1 for additional details.

```
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN
Disable|Div-1|Div-2|Div-3|Div-4|Div-6|Div-8|Div-12|Div-16|Div-24|Div-48
```

For faster performance, synchronous reads can be enabled for select parallel NOR flash. Specify the property BITSTREAM.CONFIG.BPI_SYNC_MODE with Type1 option or Type2 option according to what the selected family supports.

```
set_property BITSTREAM.CONFIG.BPI_SYNC_MODE Disable|Type1|Type2
```

If master BPI configuration with asynchronous read is required, but a faster performance is desired, the page mode and read cycle options can be used. To enable these features use the properties BITSTREAM.CONFIG.BPI_PAGE_SIZE and BITSTREAM.CONFIG.BPI_1ST_READ_CYCLE:

```
set_property BITSTREAM.CONFIG.BPI_PAGE_SIZE 1|4|8
```

- Page sizes are 1 (default), 4, or 8. If the actual flash page size is larger than 8, the value of 8 should be used to maximize the efficiency.

```
set_property BITSTREAM.CONFIG.BPI_1ST_READ_CYCLE 1|2|3|4
```

- First access CCLK cycles of 1 (default), 2, 3, or 4. CCLK cycles must be 1 if the page size is 1.

After bitstream generation it is also important during flash programming file generation to ensure the data ordering is setup correctly. On Xilinx FPGAs, data bit D00 is the most-significant bit (MSB) and bit D15 is the least significant bit (LSB). Consequently, it is crucial to understand how the data ordering in the configuration data file corresponds to the data ordering expected by the FPGA. UltraScale FPGA bitstream files (.bit, .rbt) are never bit-swapped. By default, for the BPI and SelectMAP modes the .mcs file formats are bit-swapped (see Bit Swapping, page 143). This convention is consistent across all Xilinx FPGAs. The master BPI configuration mode data ordering is the same as the SelectMAP data ordering. During the flash programming file generation the data bus width option must be set to x8 or x16 appropriately based on the target parallel NOR flash. Refer to *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for details.

Parallel NOR Flash Programming Options

Before configuring an UltraScale FPGA from a parallel NOR flash memory device, the flash must be programmed with the configuration data. Parallel NOR flash devices have a single interface for programming. Three primary methods to deliver the data to this interface follow:

- Off-board production programming by a third-party programmer, such as from BPM Microsystems or Data I/O. Refer to the selected flash vendor website for production programming support details.
  - This production programming method should be considered if the most critical factor is to decrease flash programming times for a high-volume production application. Off-board programming can often deliver faster programming times because they can limit overhead by interfacing directly to the flash. This solution can also make use of the enhanced programming higher voltage option.
- In-system production programming with a third-party vendor JTAG tool programming solution.
  - This method is popular if on-board production programming is required, but this method will be slower than off-board programming.
- Indirect in-system low-volume prototyping programming with Vivado Device Programmer
  - This method is popular if programming must be done on-board. This method can accommodate multiple design iterations and is extremely useful for debugging in a lab environment. The Vivado programming tool will provide the ability to program a parallel NOR flash indirectly. An FPGA design bitstream is downloaded first to provide a connection from the Vivado tools through the FPGA to the parallel NOR

flash. When using this method it is important to recognize that the previous FPGA memory design contents are lost during the flash operations. I/O signals that are not a part of the master BPI configuration mode interface are disabled. You must understand the behavior of the FPGA during this process and how it can affect other devices in the system. Refer to *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for the specific flash family members supported by the programming tools.

For step-by-step instructions for using the BPI configuration mode with parallel NOR flash, see *UltraScale FPGA BPI Configuration and Flash Programming* (XAPP1220) [Ref 13].

Chapter 5

SelectMAP Configuration Modes

Introduction

The SelectMAP configuration interface provides an 8-bit, 16-bit, or 32-bit bidirectional data bus interface to the FPGA configuration logic that can be used for both configuration and readback. Both master SelectMAP and slave SelectMAP interfaces are supported. See Differences Between UltraScale FPGA Families, page 11.

SelectMAP Configuration Interface

CCLK is an output in master SelectMAP mode and an input in slave SelectMAP mode. Slave SelectMAP mode is recommended; master SelectMAP mode is not supported in the Kintex UltraScale+ and Virtex UltraScale+ FPGAs. Master SelectMAP is supported in the Kintex UltraScale and Virtex UltraScale FPGAs for legacy applications. Xilinx Platform Flash PROMs do not support the UltraScale™ architecture-based FPGAs. For parallel modes, BPI configuration is recommended.

RECOMMENDED: *The alternative master BPI mode is the dominant configuration mode for configuration from a parallel-type flash device. Master SelectMAP mode is not recommended for new designs. See Differences Between UltraScale FPGA Families, page 11.*

Readback and the read direction of the data bus are applicable only to slave SelectMAP mode. The bus width of SelectMAP is automatically detected. One or more 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 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 SelectMAP configuration interface pins shown in Figure 5-1 are defined in Table 1-9, page 31.

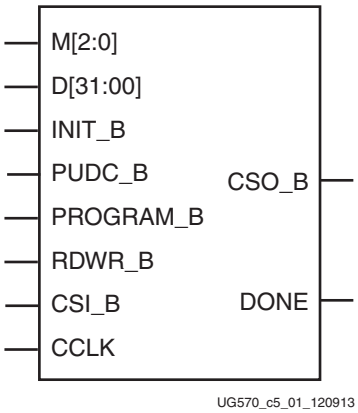


Figure 5-1: **SelectMAP Configuration Interface**

Single Device SelectMAP Configuration

For custom applications where a microprocessor or CPLD is used to configure a single FPGA, either master SelectMAP mode (use CCLK from the FPGA) or slave SelectMAP mode can be used (see Figure 5-2). Slave SelectMAP mode is preferred. See *Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave SelectMAP Mode* (XAPP583) [Ref 14] for information on configuring Xilinx FPGAs using a microprocessor.

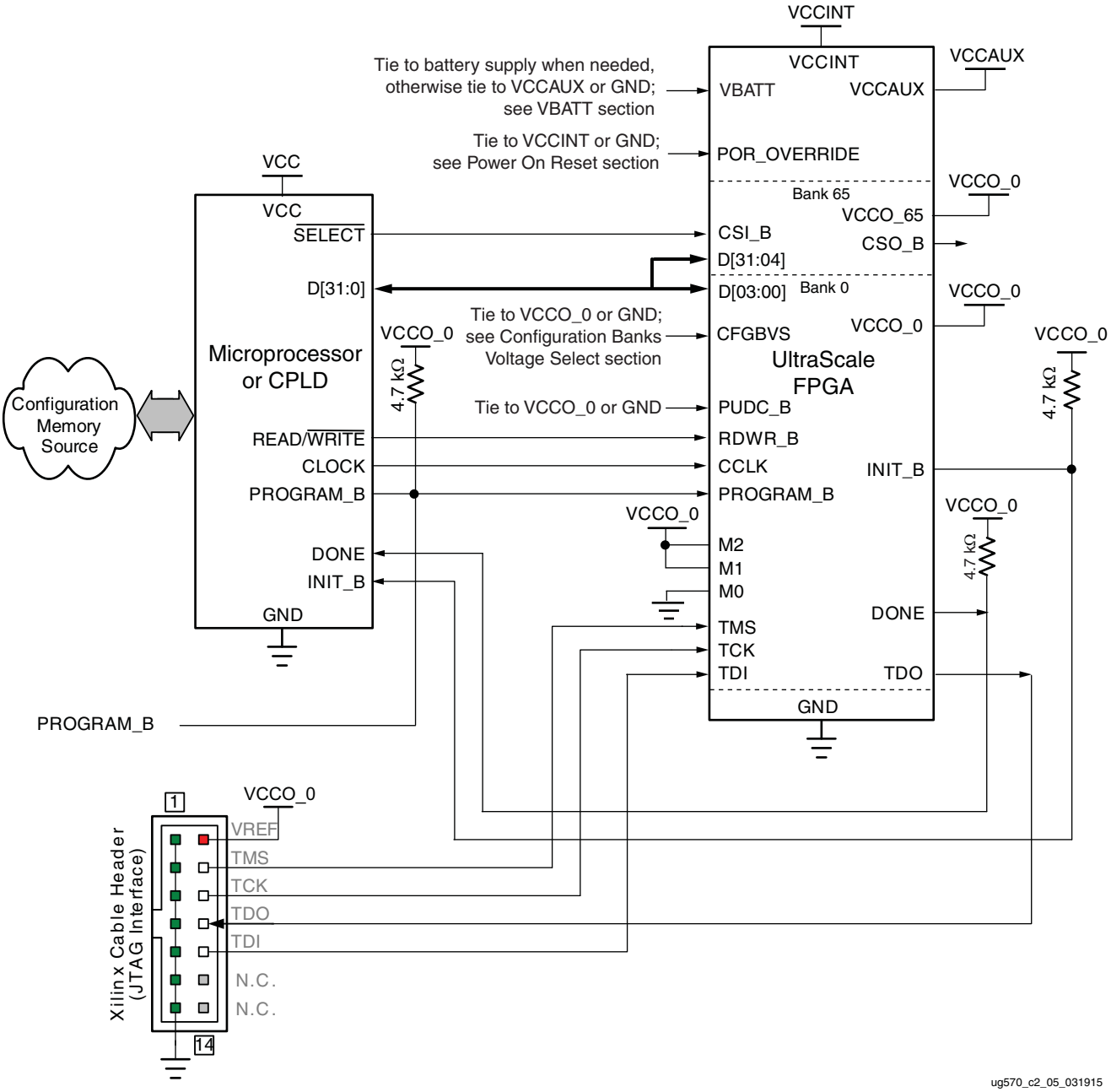


Figure 5-2: **Slave SelectMAP Configuration Interface Example**

Notes relevant to Figure 5-2:

1. Refer to *Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave SelectMAP Mode* (XAPP583) [Ref 14], 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.
3. The DONE pin is an open-drain output. See Table 1-9, page 31 for DONE signal details.

4. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required. See Table 1-9, page 31 for INIT_B signal details.
5. The CSI_B and RDWR_B signals can be tied to GND if only one FPGA is going to be configured and readback is not needed.
6. CCLK signal integrity is critical.
7. Data bus width can be x8, x16, or x32 for slave SelectMAP configuration.
8. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. See Table 1-9, page 31 for PUDC_B signal details.

SelectMAP Data Loading

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

CSI_B

The chip select input (CSI_B) enables the SelectMAP bus. When CSI_B is High, the FPGA ignores the SelectMAP interface, neither registering any inputs nor driving any outputs. The D[31:00] 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 interface and readback is not required, the CSI_B signal can be tied to ground.

RDWR_B

RDWR_B is an input to the FPGA 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.

3D ICs do not support the ABORT sequence. In monolithic devices, 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.

The RDWR_B signal is ignored while CSI_B is de-asserted. 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 84).

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 CCLK.

On the next rising CCLK edge, the device begins sampling the data pins. Only D[07:00] are sampled by configuration until the bus width is determined. 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. 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.

After configuration, the CSI_B and RDWR_B signals can be de-asserted, or they can remain asserted. Because the SelectMAP port is inactive, toggling RDWR_B at this time does not cause an ABORT. Figure 5-3 summarizes the timing of SelectMAP configuration with continuous data loading.

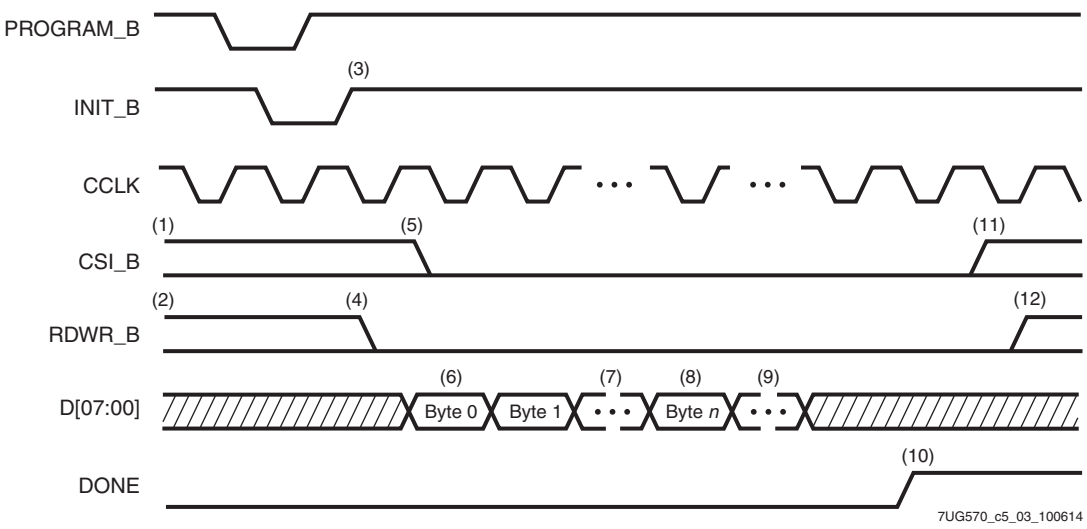


Figure 5-3: **Continuous x8 SelectMAP Data Loading**

Notes relevant to Figure 5-3:

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 on the next CCLK.
3. The Mode pins are sampled when INIT_B goes High.
4. RDWR_B should be asserted before CSI_B to avoid causing an ABORT on the next CCLK.
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.
10. The DONE pin goes High during the startup sequence. Additional CCLKs can be required to complete the startup sequence.
11. After configuration has finished, the CSI_B signal can be deasserted.
12. After the CSI_B signal is deasserted, RDWR_B can be deasserted.
13. The data bus can be x8, x16, or x32 (for slave SelectMAP).

Non-Continuous SelectMAP Data Loading

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

Configuration can be paused in two ways: by deasserting the CSI_B signal (Free-Running CCLK method, Figure 5-4) or by halting CCLK (Controlled CCLK method, Figure 5-5). For encrypted bitstreams using an obfuscated key with the SelectMAP or ICAP interface, do not pause bitstream loading by temporary de-assertion of the configuration interface chip-select (CSI_B). Instead, keep CSI_B asserted and stop the CCLK to pause bitstream loading. See Answer 73656 for details.

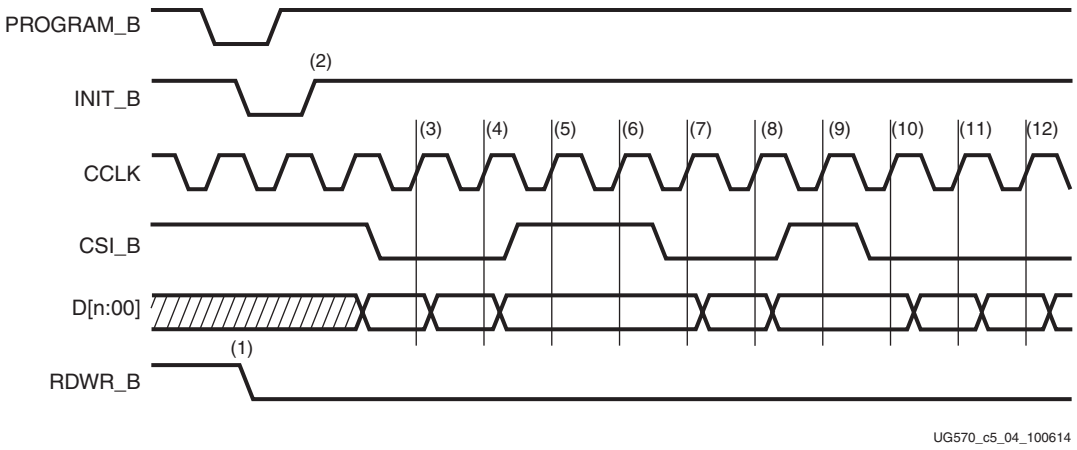


Figure 5-4: **Non-Continuous SelectMAP Data Loading with Free-Running CCLK**

Notes relevant to Figure 5-4:

1. RDWR_B is driven Low by the user, setting the D[n:00] 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 on the next CCLK.
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.

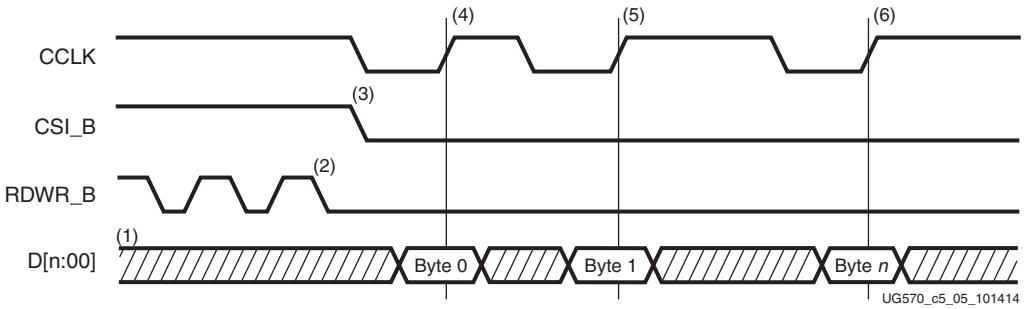


Figure 5-5: **Non-Continuous SelectMAP Data Loading with Controlled CCLK**

Notes relevant to Figure 5-5:

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 D00 pin. This convention (D00 = MSB, D07 = LSB) differs from many other devices. This convention can be a source of confusion when designing custom configuration solutions. Table 5-1 shows how to load the hexadecimal value 0xABCD into the SelectMAP data bus.

Table 5-1: **Bit Ordering for SelectMAP 8-Bit Mode**

CCLK Cycle	Hex Equivalent	D00	D01	D02	D03	D04	D05	D06	D07
1	0xAB	1	0	1	0	1	0	1	1
2	0xCD	1	1	0	0	1	1	0	1

Notes:
1. D[07:00] 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 tools can generate bit-swapped files.

Table 5-2 shows the bit ordering for the SelectMAP x8, x16, and x32 data bus widths.

Table 5-2: **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	09	08	07	06	05	04	03	02	01	00
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

3D ICs do not support the ABORT sequence. In monolithic devices 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 Figure 5-5.

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 5-6.

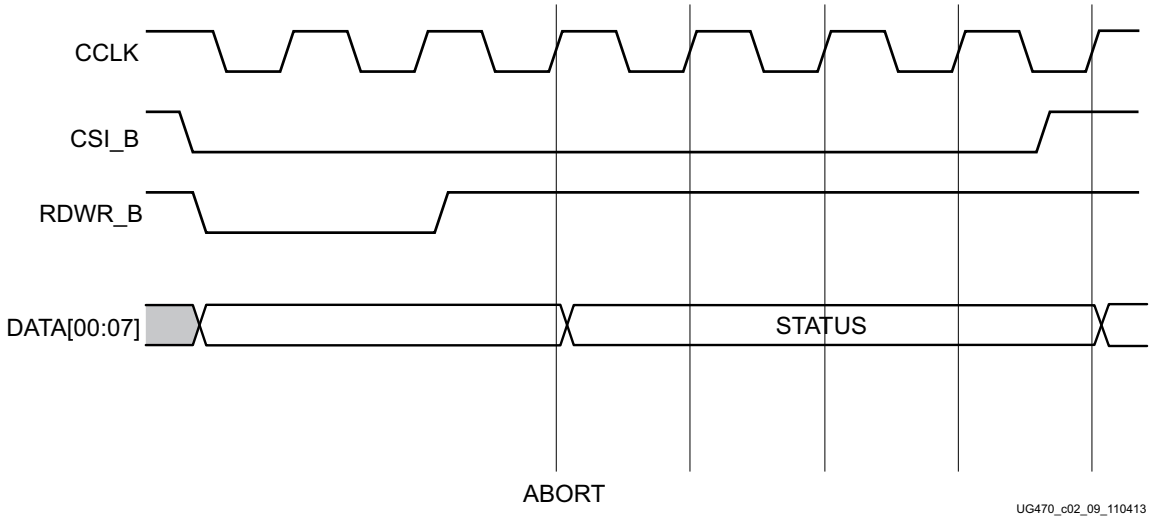


Figure 5-6: **Configuration Abort Sequence for SelectMAP Modes**

Readback Abort Sequence Description

An ABORT is signaled during readback as follows (see Figure 5-7):

5. The readback sequence begins normally.
6. The user pulls the RDWR_B pin Low synchronous to CCLK while the device is selected (CSI_B asserted Low).
7. The ABORT ends when CSI_B is deasserted.

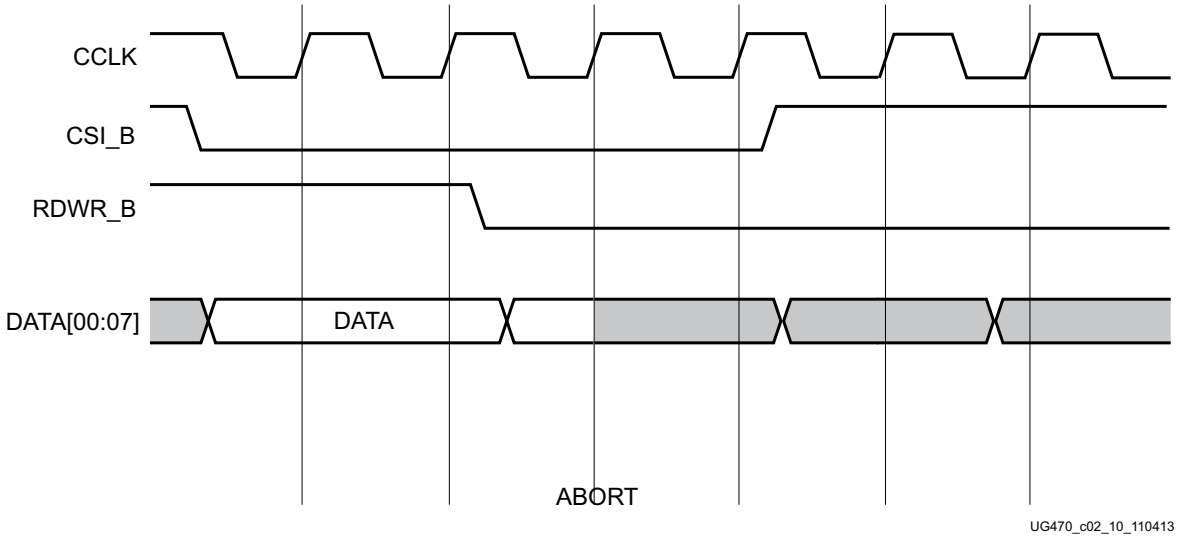


Figure 5-7: **Readback Abort Sequence**

ABORTs during readback are not followed by a status word because the RDWR_B signal is set for write control (FPGA D[x: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 5-3.

Table 5-3: **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-D02	RSVD	Reserved status bits
D01-D00	11	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.

Chapter 6

Boundary-Scan and JTAG Configuration

Introduction

Kintex® UltraScale™ and Virtex® UltraScale FPGAs support IEEE standards 1149.1 and 1149.6 (ACJTAG), defining a Test Access Port (TAP) and boundary-scan architecture. The Test Access Port and boundary-scan architecture is commonly referred to collectively as JTAG. JTAG is an acronym for the Joint Test Action Group, the technical subcommittee initially responsible for developing this standard. The boundary-scan architecture is used to ensure 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 connectivity testing, the 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.

For compliance with the pre-configuration BSDL file description, PUDC_B should be tied to V_{CCO_0} to disable pull-ups, which matches the pre-configuration BSDL file disable result description of 'Z' for when a boundary-scan controller disables the output to a pin. Otherwise, if PUDC_B is tied to GND, then pre-configuration weak pull-up resistors are enabled and the corresponding output disable result of 'PULL1' in the BSDL file is a more accurate match to the device pin behavior. However, to maximize boundary-scan tests for external pull-up or pull-down resistors with pre-configured devices, the default disable result value 'Z' in the pre-configuration BSDL file is recommended for both settings of PUDC_B, and all external pull-down resistors must be sufficiently strong to override potential internal pull-ups that are enabled when PUDC_B is tied to GND.

Boundary-Scan Using IEEE Standard 1149.1

UltraScale architecture is fully compliant with the IEEE Standard 1149.1 Test Access Port and Boundary-Scan Architecture. The UltraScale FPGAs include 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. UltraScale FPGAs also support a 32-bit Device Identification register and a Configuration register. This section outlines the details of the JTAG architecture.

Test Access Port (TAP)

The FPGA TAP contains four mandatory dedicated pins as specified by the protocol and as used in the typical JTAG architecture (see Table 6-1). Three input pins and one output pin control the IEEE Std 1149.1 boundary-scan TAP controller. Optional control pins, such as Test Reset (TRST) and enable pins, might be found on devices from other manufacturers. It is important to be aware of these optional signals when interfacing Xilinx devices with parts from different vendors because these optional pins might need to be driven.

Table 6-1: **TAP Controller Pins**

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 to 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, which change on the rising edge of TCK. TMS has an internal resistive pull-up to provide a logic High if the pin is not driven.

Table 6-1: **TAP Controller Pins**

Pin	Direction	Pre-Configuration Internal Pull Resistor	Description
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 default weak internal pull-up resistors, as specified by the IEEE Std 1149.1, as do TDO and TCK. These internal pull-up resistors are active, regardless of the mode selected. Refer to the data sheet for internal pull-up values.

Boundary Scan Timing Parameters

Characterization data for some of the most commonly requested timing parameters, shown in Figure 6-1, are listed in the respective data sheet ([Ref 8] or [Ref 9]) in the Configuration Switching Characteristics table.

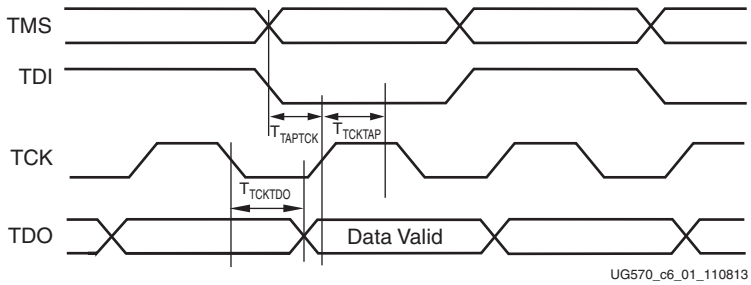


Figure 6-1: **Boundary Scan Port Timing Waveforms**

Using Boundary Scan in Xilinx Devices

For single-device configuration, the TAP controller commands are issued automatically if the part is being configured with Xilinx configuration 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 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 6-2 shows a typical JTAG setup with the simple connections 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 configuration 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.

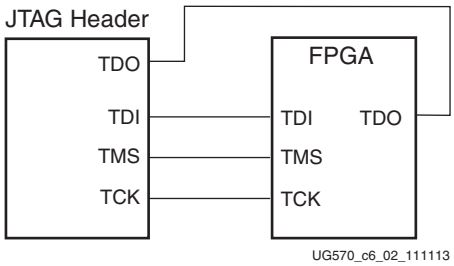


Figure 6-2: **Single Device JTAG Programming Connections**

It is also possible to configure multiple devices in a chain, as shown in Figure 6-3.

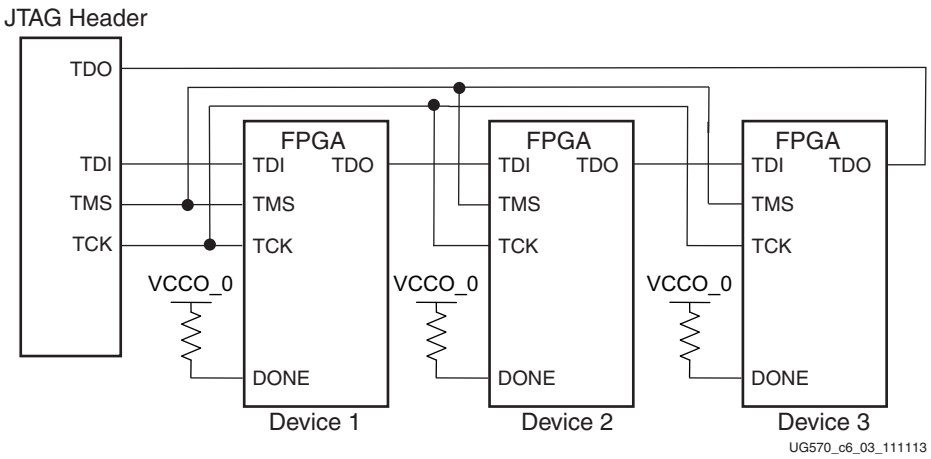


Figure 6-3: **Boundary-Scan Chain of Devices**

Boundary-Scan Design Considerations

JTAG Signal Routing

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

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

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

Providing Power

To ensure proper power-on behavior, the guidelines in the data sheet must be followed. The power supplies should ramp monotonically within the power supply ramp time range specified in the data sheet. All supply voltages should be within the recommended operating ranges; any dips below the data retention values in the data sheet can result in loss of configuration data.

To ensure boundary-scan functionality, any guidelines for powering unused serial transceiver tiles must be followed.

TAP Controller and Architecture

The FPGA TAP contains four mandatory dedicated pins as specified by the protocol given in Table 6-1 and illustrated in Figure 6-4, a typical JTAG architecture.

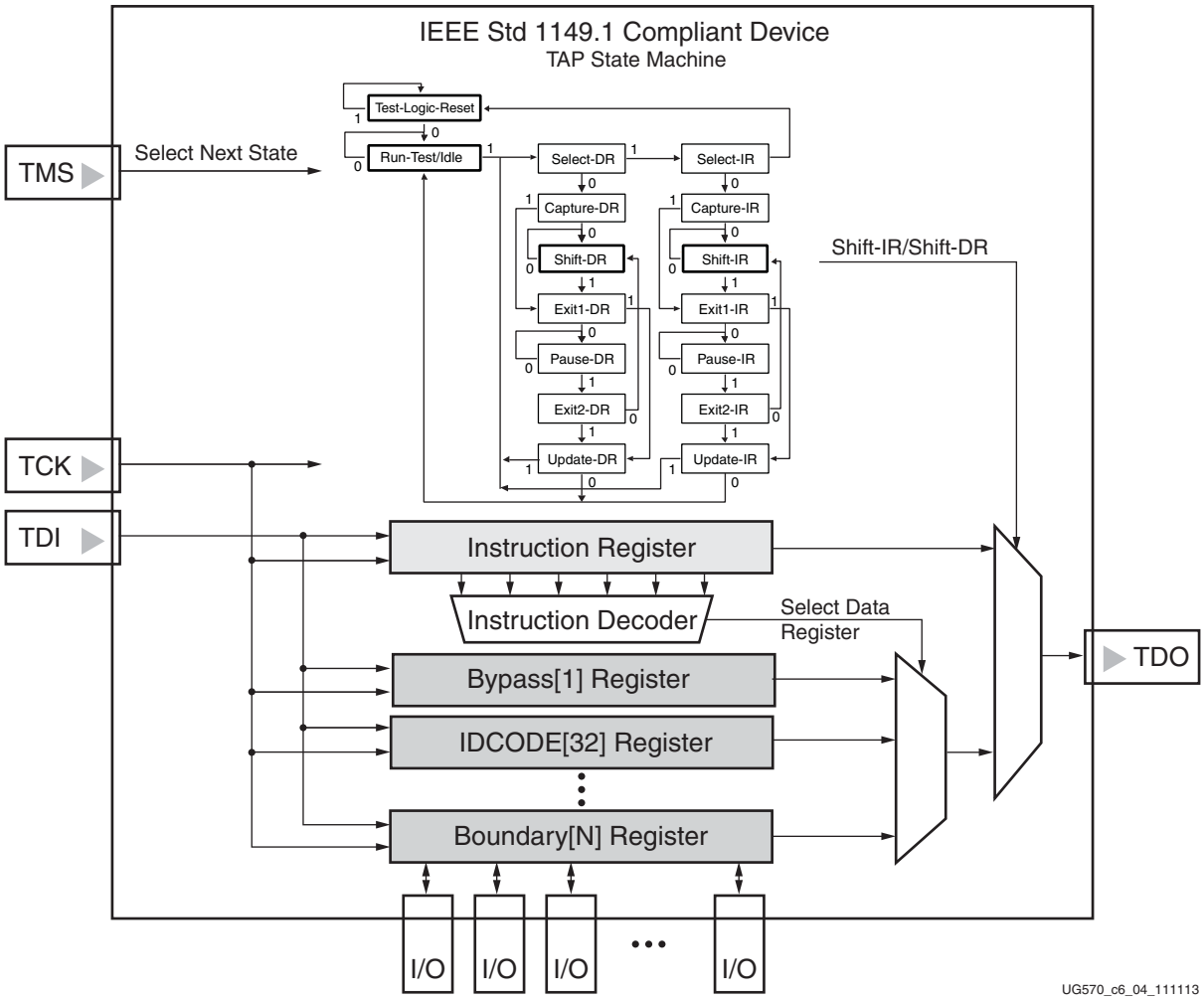


Figure 6-4: **Typical JTAG Architecture**

Figure 6-5 shows 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.

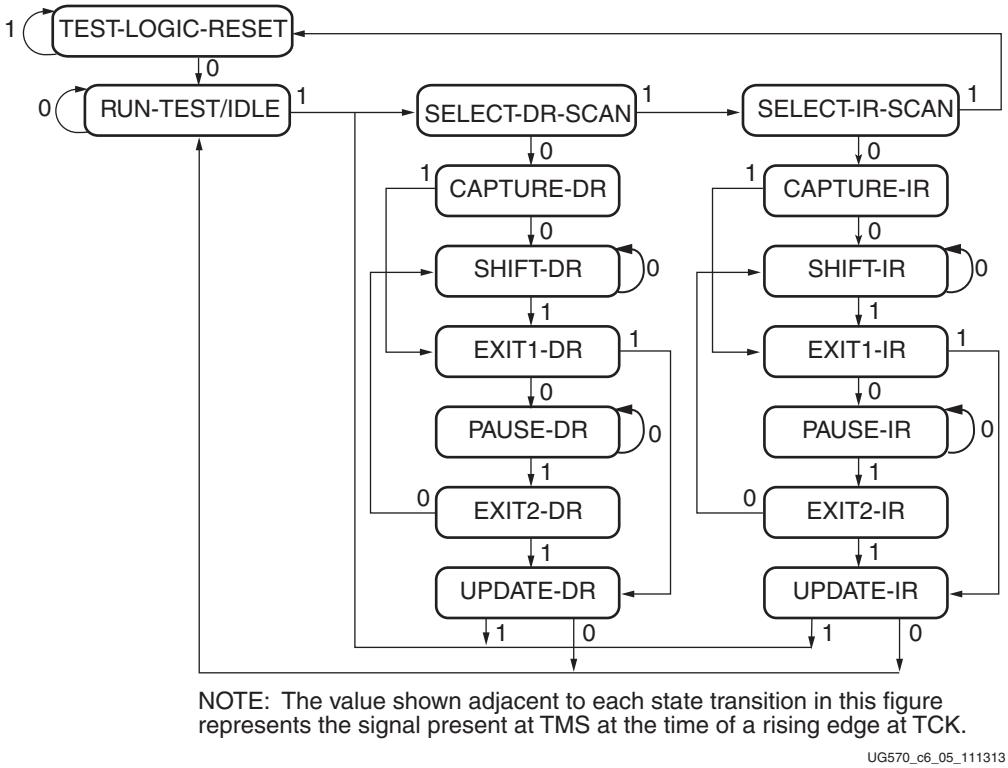


Figure 6-5: **Boundary-Scan TAP Controller State Machine**

The operation of each state is described as follows.

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.

UltraScale FPGAs 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.

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

Boundary-Scan Architecture Registers

UltraScale architecture-based FPGAs 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 (see Table 6-2).

Table 6-2: **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 6-3, page 99.
Bypass Register	1 bit	Bypasses the device.
Device Identification Register	32 bits	Captures the device ID.
JTAG Configuration Register	Varies	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 in the devices based on SSI technology. 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 6-6 is a representation of the UltraScale FPGA boundary-scan architecture.

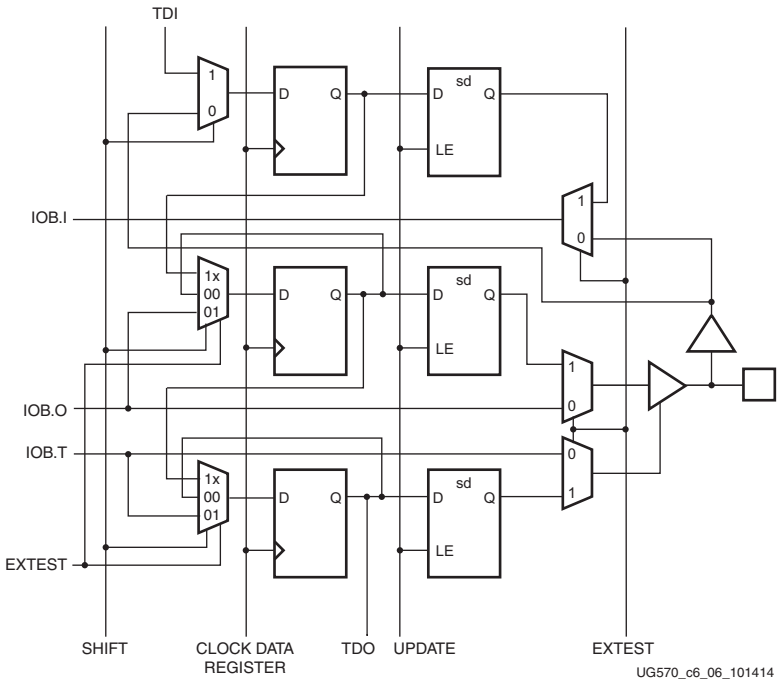


Figure 6-6: **UltraScale FPGA Boundary-Scan Logic**

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

The update latch is opened each time the TAP controller enters the UPDATE-DR state. Care is necessary when exercising an 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.

Bit Sequence of Boundary-Scan Register

This section describes the order of each non-TAP IOB. The input is first, the output second, and the 3-state IOB control third. The 3-state IOB control is closest to the TDO. The input-only pins contribute only the input bit to the boundary-scan I/O data register. The bit sequence of the device is obtainable from the Boundary-Scan Description Language Files (BSDL files) for the UltraScale FPGAs. (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 write_bsdl utility to automatically modify the BSDL file for post-configuration interconnect testing. The write_bsdl 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.

Instruction Register

The Instruction register (IR) 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 UltraScale FPGA boundary-scan instruction set is loaded into the Instruction register. The IR is 6 bits wide for monolithic UltraScale FPGAs. See Table 1-5, page 23 for IR length for other devices. Table 6-3 describes the boundary scan instructions for UltraScale FPGAs. See the BSDL files for commands and codes for UltraScale+ FPGAs. See Table 8-3, page 135 for eFUSE related instructions.

Table 6-3: **UltraScale FPGA Boundary-Scan Instructions**

Boundary-Scan Command	Binary Code [5:0](1)	Description
EXTEST	100110	Enables 1149.1 boundary-scan EXTEST operation
EXTEST_PULSE	111100	Enables 1149.6 EXTEST_PULSE operation for transceivers
EXTEST_TRAIN	111101	Enables 1149.6 EXTEST_TRAIN operation for transceivers
SAMPLE/PRELOAD	000001	Enables boundary-scan SAMPLE/PRELOAD 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
USERCODE	001000	Enables shifting out user code
IDCODE	001001	Enables shifting out of IDCODE
HIGHZ_IO	001010	Disable user I/O pins only while enabling the Bypass register
BYPASS	111111	Enables BYPASS
CFG_IN	000101	Access the configuration bus for configuration
CFG_OUT	000100	Access the configuration bus for readback
JPROGRAM	001011	Equivalent to PROGRAM_B pin
JSTART	001100	Clocks the startup sequence
JSHUTDOWN	001101	Clocks the shutdown sequence

Table 6-3: **UltraScale FPGA Boundary-Scan Instructions *(Cont'd)***

Boundary-Scan Command	Binary Code [5:0](1)	Description
SYSMON_DRP	110111	System Monitor DRP access through JTAG. See the DRP JTAG Interface section in the *UltraScale Architecture System Monitor User Guide* (UG580) [Ref 15].
ISC_NOOP	010100	No operation
RESERVED	All other codes	Xilinx reserved instructions

Notes:

1. Instruction register is larger for devices based on SSI technology (see Table 1-5, page 23). See the BSDL files for device-specific information.

Table 6-4 shows the instruction capture values loaded into the IR as part of an instruction scan sequence.

Table 6-4: **UltraScale 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	01	

Notes:

1. INIT is the INIT_B_INTERNAL_SIGNAL_STATUS bit.
2. Instruction register is larger for devices based on SSI technology (see Table 1-5, page 23). 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

UltraScale FPGAs 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 (see Table 1-5, page 23).

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 UltraScale FPGAs. 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 property) 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 primitive is required when creating these registers (see BSCANE2 in Chapter 7.)

Using Boundary-Scan Configuration in UltraScale FPGAs

One of the most common boundary-scan vendor-specific instructions is the Configure instruction. UltraScale architecture-based FPGAs support configuration through the standard boundary-scan (JTAG) port. If the device is configured via JTAG, the configure instructions occur independent from the selection on the mode pins. However, an explicit JTAG configuration mode setting is available when the devices are to be exclusively configured through the JTAG port. The JTAG mode pin setting restricts master ICAP access to slave SLRs, and therefore is not recommended for devices based on SSI technology. See Figure 6-7 for the pin connections for the JTAG configuration interface.

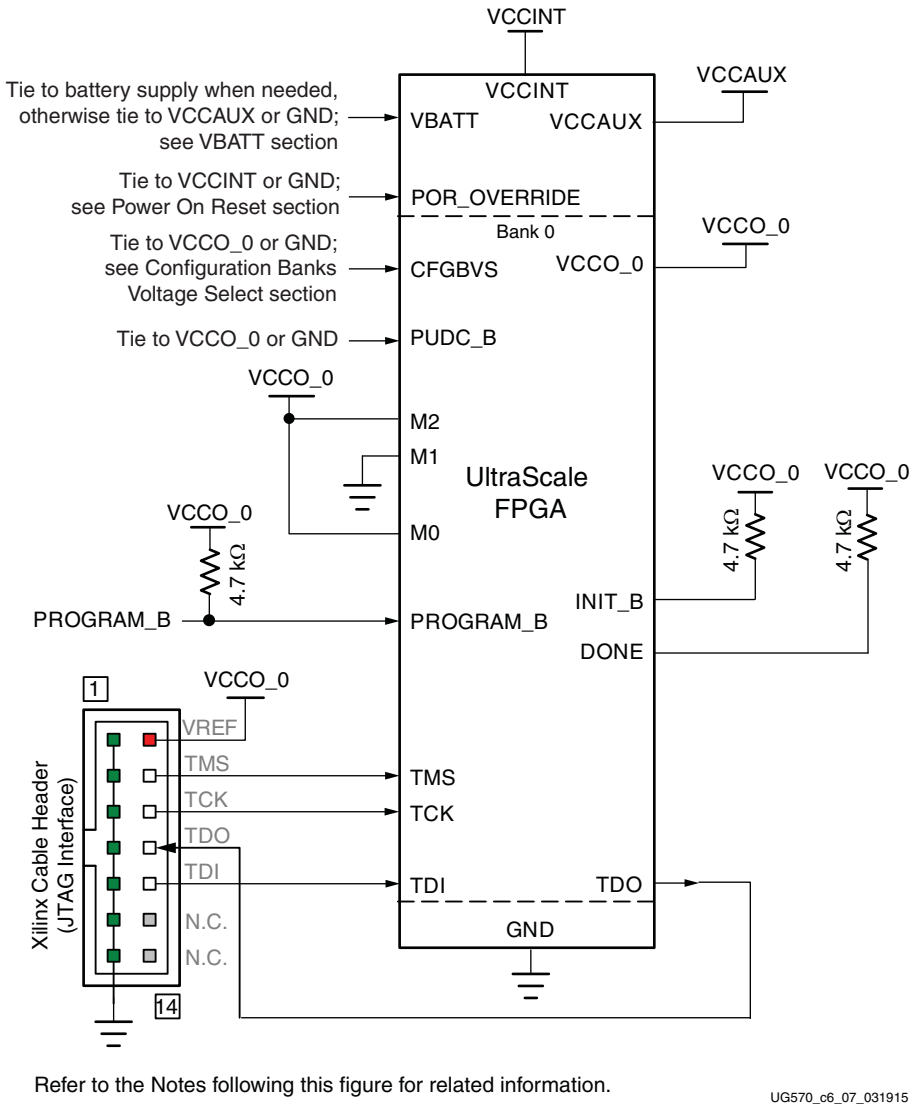


Figure 6-7: **JTAG Configuration Interface Example**

Notes relevant to Figure 6-7:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIO pins after power-up and during configuration. For BSDL compliance, PUDC_B should be tied to GND. See Table 1-9, page 31 for PUDC_B signal details.

Xilinx has USB proprietary programming cables 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.

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. The TAP controller is not reset by the PROGRAM_B pin and can only be reset by bringing the controller to the TLR state. The TAP controller is reset on power up.

The configuration flow for FPGA configuration with JTAG is shown in Figure 6-8. 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. The JTAG state machine must be in the Shift-DR state during configuration with the CFG_IN instruction.

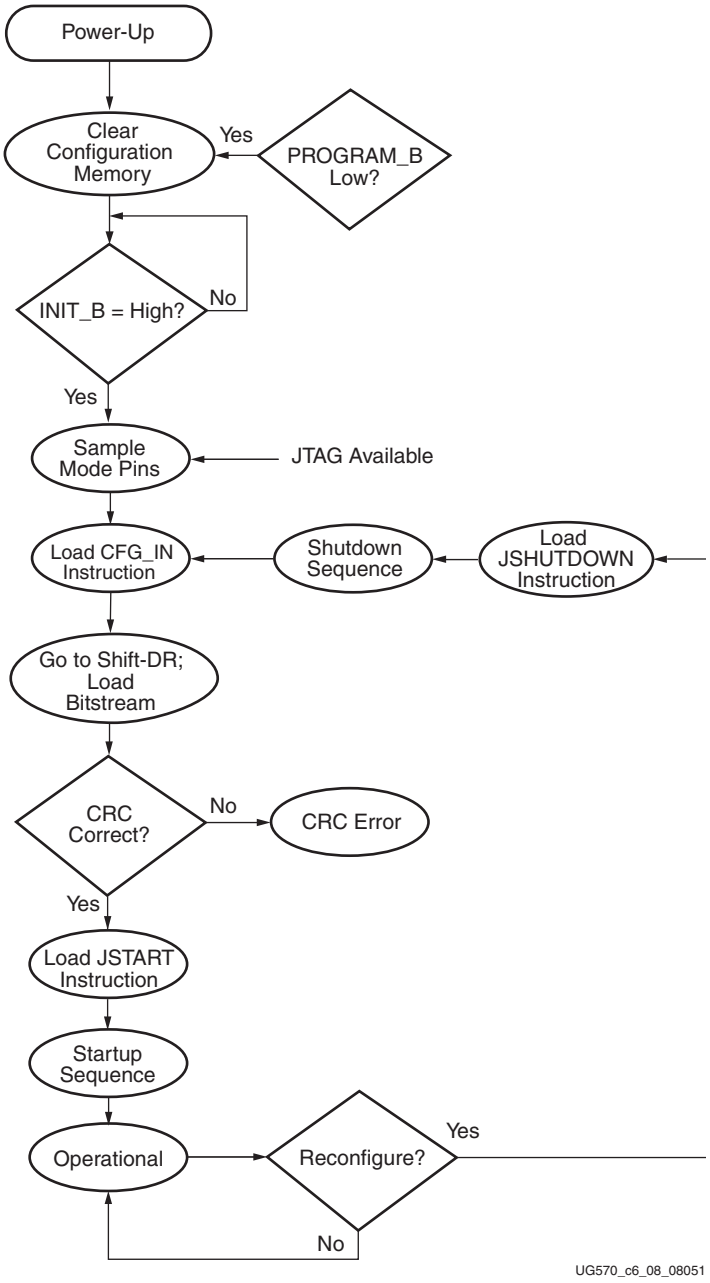


Figure 6-8: **Device Configuration Flow Diagram**

Single Device Configuration

Table 6-5 describes the TAP controller commands required to configure an UltraScale FPGA. Refer to Figure 6-5 for TAP controller states. These TAP controller commands are issued automatically if configuring the part with the Vivado® device programmer. For encrypted bitstreams using an obfuscated key with the JTAG interface, do not pause bitstream loading by temporary excursion from the JTAG Shift-DR state to the JTAG Pause-DR state. Instead, stay within the JTAG Shift-DR state and stop the JTAG TCK clock to pause bitstream loading. See Answer 73656 for details.

Table 6-5: **Single Device Configuration Sequence**

TAP Controller Step and Description		Set and Hold		No. 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 Test-Logic-Reset (TLR) state.	X	1	5
2	Move to the Run-Test/Idle (RTI) state.	X	0	1
3	Move to the SELECT-IR state.	X	1	2
4	Move to the SHIFT-IR state.	X	0	2
5	Start loading the JPROGRAM instruction, LSB first:	01011 (1)	0	5
6	Load the MSB of the JPROGRAM instruction when exiting SHIFT-IR, as defined in the IEEE standard.	0	1	1
7	Move to the UPDATE-IR state.	X	1	1
8	Move to the SELECT-IR state.	X	1	2
9	Move to the SHIFT-IR state.	X	0	2
10	Start loading the CFG_IN instruction, LSB first:	00101	0	5
11	Load the MSB of the CFG_IN instruction when exiting SHIFT-IR.	0	1	1
12	Move to the UPDATE-IR state.	X	1	1
13	Move to the RTI state and wait for t_{POR} delay.	X	0	20,000 (2)
14	Move to the SELECT-DR state.	X	1	1
15	Move to the SHIFT-DR state.	X	0	2
16	Shift in the FPGA bitstream. Bit n (MSB) is the first bit in the bitstream. (3) (4)	Bit 1 … bit n	0	(bits in bitstream) -1
17	Shift in the last bit of the bitstream. Bit 0 (LSB) shifts on the transition to EXIT1-DR.	Bit 0	1	1
18	Move to the UPDATE-DR state.	X	1	1
19	Move to the RTI state.	X	0	1
20	Move to the SELECT-IR state.	X	1	2
21	Move to the SHIFT-IR state.	X	0	2
22	Start loading the JSTART instruction (optional). The JSTART instruction initializes the startup sequence.	01100	0	5
23	Load the MSB of the JSTART instruction when exiting SHIFT_IR.	0	1	1
24	Move to the UPDATE-IR state.	X	1	1
25	Move to the RTI state and clock the startup sequence by applying a minimum of 2,000 clock cycles to the TCK.	X	0	2,000

Table 6-5: **Single Device Configuration Sequence *(Cont'd)***

TAP Controller Step and Description		Set and Hold		No. of Clocks
		TDI	TMS	TCK
26	Move to the TLR state. The device is now functional.	X	1	3

Notes:

1. Instruction register is larger for devices based on SSI technology (see Table 1-5, page 23). See the BSDL files for device-specific information.
2. Assumes a t_{POR} wait time of 20 ms is necessary until INIT_B is High and CFG_IN can be sent. In this example, the TCK cycle value is based on a TCK frequency of 1 MHz.
3. 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.
4. For 3D ICs based on Stacked Silicon Interconnect (SSI) technology, the JTAG TAP must stay in the SHIFT-DR state until the entire bitstream is shifted into the device in step 16, with the only exception being bit 0 which is shifted in step 17.

Multiple Device Configuration

It is possible to configure multiple UltraScale FPGAs in a chain (see Figure 6-3.) 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 excellent signal integrity is maintained. The configuration tools automatically discover the devices in the chain, starting from the one nearest to TDI coming from the JTAG header. If JTAG is the only configuration mode, then PROGRAM_B, INIT_B, and DONE can each be connected to separate pull-up resistors.

Refer to the state diagram in Figure 6-5 for the following TAP controller steps:

1. On power-up, place a logic 1 on the TMS and clock the TCK five times. This ensures starting in the TLR (Test-Logic-Reset) state.
2. Load the CFG_IN instruction into the target device (and BYPASS in all other devices).
3. Go through the RTI state (RUN-TEST/IDLE).
4. Load in the configuration bitstream per steps 13 through 17 in Table 6-5.
5. Repeat step 2 and step 3 for each device.
6. Load the JSTART command into all devices.
7. Go to the RTI state and clock TCK 2,000 times.

All devices are active at this point.

Design Entry

Introduction

Although most aspects of configuration happen before any design elements can have an effect on the process, there are some design elements that provide access to configuration-related features during device operation, after the initial configuration is complete. These design elements must be instantiated in the design. Most should only be used in specific situations that require them. User options that impact configuration are specified through properties (see Design Tools, page 25).

The following design primitives are related to configuration features and described in this chapter. Information on these and other primitives are also found in the *UltraScale Architecture Libraries Guide* (UG974) [Ref 16]. These primitives all require instantiation in a design. Instantiation templates for each are found in the Libraries Guide and in the Vivado Language Templates.

RECOMMENDED: *The FRAME_ECCE3 and FRAME_ECCE4 (for UltraScale+) should only be instantiated through use of the Soft Error Mitigation (SEM) IP.*

- BSCANE2
- DNA_PORTE2
- EFUSE_USR
- FRAME_ECCE3
- FRAME_ECCE4
- ICAPE3
- MASTER_JTAG
- STARTUPE3
- USR_ACCESSE2

BSCANE2

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 test access port (TAP) pins of the FPGA. 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, or when using indirect flash programming in the Vivado Device Programmer.

The BSCANE2 primitive is identical to that found in the 7 series FPGAs.

For more details on boundary scan and usage of the BSCANE2 primitive, see Chapter 6, Boundary-Scan and JTAG Configuration.

Primitive

Figure 7-1 shows the primitive.

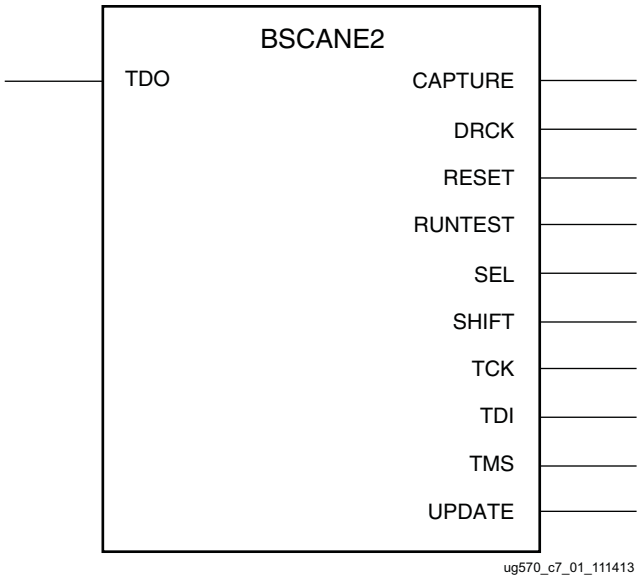


Figure 7-1: **BSCANE2 Primitive**

Pin Descriptions

Table 7-1 defines the BSCANE2 pin connections.

Table 7-1: **BSCANE2 Pin Descriptions**

Pin	Type	Width	Description
CAPTURE	Output	1	Asserted when TAP controller is in Capture-DR state.
DRCK	Output	1	Gated TCK output. When SEL is asserted, DRCK toggles when CAPTURE or SHIFT are asserted.
RESET	Output	1	Asserted when TAP controller is in Test-Logic-Reset state.
RUNTEST	Output	1	Asserted when TAP controller is in Run-Test/Idle state.
SEL	Output	1	Asserted when the USER instruction (USER1 – USER4) that corresponds to the BSCANE2 instance's JTAG_CHAIN attribute (1–4) is loaded as the active instruction in the JTAG Instruction register.
SHIFT	Output	1	Asserted when TAP controller is in Shift-DR state.
TCK	Output	1	Test Clock. Primitive output from external TAP pin to FPGA internal logic.
TDI	Output	1	Test Data Input. Primitive output from external TAP pin to FPGA internal logic.
TDO	Input	1	Test Data Output. Primitive input from User internal scan register to a flip-flop that registers the primitive input on the falling edge of TCK. The output of the flip-flop is forwarded to the external TAP TDO pin.
TMS	Output	1	Test Mode Select. Primitive output from external TAP pin to FPGA internal logic.
UPDATE	Output	1	Asserted when TAP controller is in Update state. Internal logic should update from the internal scan register on the rising edge of the UPDATE signal.

Attributes

Table 7-2 describes the BSCANE2 primitive attributes.

Table 7-2: **BSCANE2 Attributes**

Attribute	Type	Allowed Values	Default	Description
DISABLE_JTAG	BOOLEAN	False, True	False	Disables JTAG boundary scan.
JTAG_CHAIN	DECIMAL	1, 2, 3, 4	1	Chain designator number for USER command.

Applications

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 supports 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 3D ICs based on SSI technology, the BSCANE2 can only be instantiated in the master SLR. The tools automatically place the element in the correct SLR. Only the JTAG port on the master SLR can be accessed by the BSCANE2 primitive. For more details on SSI technology, see *UltraScale Architecture and Product Overview* (DS890) [Ref 4].

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.

The BSCANE2 primitive can be used to disable the external JTAG port by instantiating it and setting `DISABLE_JTAG=TRUE`. This prevents re-configuration through JTAG, including with the Vivado Device Programmer, by breaking the JTAG chain. The design property `set_property disable_jtag yes [current_design]` can also be used. These methods are preferred over the `write_bitstream` option `Bitstream.general.disable_jtag:Yes` (default is No). The primitive has priority; JTAG cannot be enabled by `write_bitstream` if it is disabled in the BSCANE2 attribute.

UltraScale+ devices add special internal pin names for the timing of some of the BSCANE2 pins. For constraints on BSCANE2 timing paths, use the following internal pin names for listed BSCANE2 pins.

Table 7-3: **BSCANE2 Special UltraScale+ Device Internal Pin Descriptions**

BSCANE2 Pin	Internal Pin for Constraints	Description
TCK	INTERNAL_TCK	Equivalent to device TCK pin
TMS	INTERNAL_TMS	Equivalent to device TMS pin
TDI	INTERNAL_TDI	Equivalent to device TDI pin
TDO	INTERNAL_TDO	Falling-edge output register

Example constraints for the special internal BSCANE2 timing pins:

- BSCANE2.TDO
  - A 20 MHz (50 ns period), 50% duty cycle TCK clock constraint on BSCANE2.INTERNAL_TCK clock source pin covers timing path from source register clocked by BSCANE2.TCK through BSCANE2.TDO to the falling-edge BSCANE2.INTERNAL_TDO register. Device TDO output timing is defined by data sheet T_{TCKTDO}.

```
create_clock -name TCK -period 50 -waveform {0 25} [get_pins */INTERNAL_TCK]
```

- BSCANE2.TDI
  - An input delay constraint for an external source with a falling-edge clock-to-output valid time of 15.0 ns, max through device TDI input pin through BSCANE2.INTERNAL_TDI pin through BSCANE2.TDI port to fabric register.

```
set_input_delay 15 -clock_fall -clock [get_clocks TCK] [get_pins BSCAN/INTERNAL_TDI]
```

- BSCANE2.TMS
  - An input delay constraint for an external source falling-edge clock-to-output valid time of 15.0 ns, maximum through device TMS input pin through BSCANE2.INTERNAL_TMS pin through BSCANE2.TMS port to fabric register.

```
set_input_delay 15 -clock_fall -clock [get_clocks TCK] [get_pins BSCAN/INTERNAL_TMS]
```

DNA_PORTE2

Each device contains a single unique, 96-bit, embedded, device identifier (device DNA). The identifier is nonvolatile, permanently programmed by Xilinx into the device via eFUSE bits, and is unchangeable, making it tamper resistant. The Device DNA is primarily used to identify the specific device.

External applications can access the DNA value through the JTAG port. FPGA designs can access the DNA internally through a Device DNA Access Port, which requires instantiation of the DNA_PORTE2 primitive. The DNA_PORTE2 primitive controls a dedicated 96-bit shift register for capturing and shifting the Device DNA value. The DNA_PORTE2 also allows for the inclusion of supplemental bits of user data, or allows for the DNA data to rollover (repeat DNA data after initial data has been shifted out).

Primitive

Figure 7-2 shows the DNA_PORTE2 primitive. The DNA_PORTE2 primitive is similar to the DNA_PORT primitive of earlier families except for the increase in the number of bits from 57 to 96. As a result, the DNA_PORTE2 cannot be directly migrated from the DNA_PORT primitive.

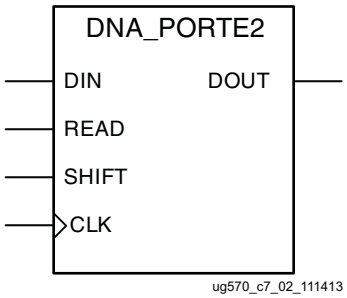


Figure 7-2: **DNA_PORTE2 Primitive**

Pin Descriptions

Table 7-4 describes the DNA_PORTE2 primitive pins. Connect all inputs and outputs to the design to ensure proper operation. All functions are synchronous to the CLK input.

Table 7-4: **DNA_PORTE2 Pin Descriptions**

Pin	Type	Width	Description
CLK	Input	1	User clock input.
DIN	Input	1	User data extension input pin.
DOUT	Output	1	DNA output pin. Transitions on the rising edge of CLK, LSB first.
READ	Input	1	Read DNA by pulsing High to parallel load 96 eFUSE bits into shift register (see Table 8-6).
SHIFT	Input	1	Active High shift enable. Shift requires Read=0.

Attributes

The attribute SIM_DNA_VALUE can be optionally set to allow for simulation of a possible DNA data sequence. By default, the Device DNA data bits are all zeros in the simulation model. Table 7-5 describes the DNA_PORTE2 primitive attributes.

Table 7-5: **DNA_PORTE2 Attributes**

Attribute	Type	Allowed Values	Default	Description
SIM_DNA_VALUE	Hex	Any 96-bit Hex value	All zeroes	Set DNA value for simulation

For more details on the Device DNA and using the DNA_PORTE2 primitive, Chapter 8, Bitstream Security, eFUSEs, and Device DNA.

EFUSE_USR

Each Kintex® UltraScale™ and Virtex® UltraScale device has a one set of 32 nonvolatile, user-defined, one-time-programmable eFUSE bits. These bits are commonly programmed by the user to define a custom user design ID. Programming is done through the JTAG port. Programming can be done using the Xilinx configuration tools and cables, or on a third-party programmer.

These 32 bits define the values in the FUSE_USER configuration register. Depending on the read/write access bits in the CNTL register, the 32 bits can be programmed and read through the JTAG port, with bit 0 shifted out first.

For internal access, the EFUSE_USR primitive must be instantiated. EFUSE_USR provides asynchronous parallel access to all 32 bits.

The EFUSE_USR primitive is identical to that in the 7 series and will directly migrate.

Primitive

Figure 7-3 shows the EFUSE_USR primitive.

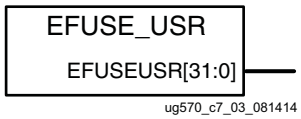


Figure 7-3: **EFUSE_USR Primitive**

Pin Descriptions

Table 7-7 describes the EFUSE_USR primitive pins.

Table 7-6: **EFUSE_USR Pin Descriptions**

Attribute	Type	Width	Description
EFUSEUSR[31:0]	Output	32	FUSE_USER register value output. Defined by customer-programmed eFUSE bits. For unprogrammed devices, all bits have a value of zero.

Attributes

Table 7-7 describes the EFUSE_USR primitive attributes.

Table 7-7: **EFUSE_USR Attributes**

Attribute	Type	Allowed Values	Default	Description
SIM_EFUSE_VALUE	Hex	Any 32-bit Hex value	All zeroes	Set EFUSE value for simulation

FRAME_ECCE3

The FRAME_ECCE3 primitive is reserved. This primitive is used when implementing the Soft Error Mitigation (SEM) IP.

The FRAME_ECCE3 in the Kintex UltraScale and Virtex UltraScale FPGAs is significantly different than the FRAME_ECCE2 of the 7 series devices. The FRAME_ECCE2 does not migrate directly to the FRAME_ECCE3, but the SEM IP automatically adapts to the new architecture. Note that the SEM IP is not supported in the KU025 device.

FRAME_ECCE4

The FRAME_ECCE4 primitive is reserved. This primitive is used when implementing the Soft Error Mitigation (SEM) IP.

The FRAME_ECCE4 primitive in the UltraScale+ devices is different than the FRAME_ECCE3 primitive in the UltraScale FPGA devices. The FRAME_ECCE3 primitive does not migrate directly to the FRAME_ECCE4 primitive, but the SEM IP generated for UltraScale+ devices is designed to use the FRAME_ECCE4 primitive automatically.

ICAPE3

The ICAPE3 provides post-configuration access to the configuration functions of the FPGA from the FPGA logic. Using this component, commands and data can be written to and read from the configuration logic. The ICAPE3 interface is similar to that for the external slave SelectMAP parallel 32-bit interface, including bit swapping (see Parallel Bus Bit Order, page 144). However, the ICAPE3 has independent input and output buses; the CSIB input ignores the input bus but the output bus can continue to toggle.

Because users can send an unencrypted partial bitstream and can perform readback through the ICAPE3 interface, users concerned about security should not connect the ICAPE3 component to external device pins. Because the improper use of this function can have a negative effect on the functionality and reliability of the FPGA, you should not use this element unless you are very familiar with its capabilities.

ICAPE3 can be useful in MultiBoot and active partial reconfiguration applications. ICAPE3 instantiation is required to issue an IPROG command that triggers the device to reload itself from the address specified in the WBSTAR (Warm Boot Starting Address) register. The WBSTAR register holds the address that the configuration controller uses after an IPROG command is issued.

Similar functionality is provided by the MCAP. Like ICAP, the MCAP can only be used after initial configuration, but it does not support readback. If Persist is set, the ICAPE3 is disabled. In addition, JTAG and MCAP have priority over ICAPE3.

The ICAPE3 for the UltraScale architecture-based FPGAs supports higher frequency and more output signals than were available in the ICAPE2 for the 7 series FPGAs. ICAPE3 only supports 32-bit interfaces, and does not have the ICAP_WIDTH attribute from the ICAPE2. ICAPE2 instantiations from 7 series designs are automatically migrated to ICAPE3 for the UltraScale FPGAs. The ICAPE3 is automatically used by some Xilinx IP, including the Soft Error Mitigation (SEM) IP.

Primitive

Figure 7-4 shows the ICAPE3 primitive.

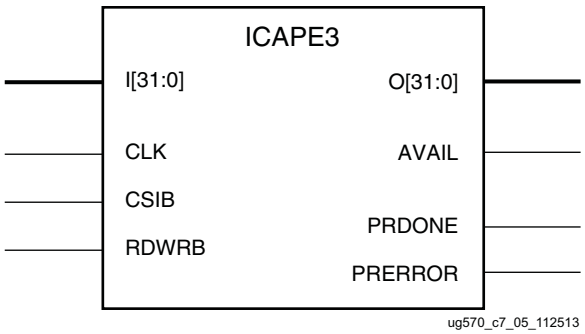


Figure 7-4: **ICAPE3 Primitive**

Pin Descriptions

Table 7-8 describes the ICAPE3 primitive pins.

Table 7-8: **ICAPE3 Pin Descriptions**

Pin	Type	Width	Description
AVAIL	Output	1	ICAP available (for JTAG or MCAP interrupt). Allows FPGA logic to react properly.
CLK	Input	1	Clock input. When the ICAPE3 primitive is instantiated, the ICAPE3 CLK input is always used as the Readback CRC clock.
CSIB	Input	1	Active-Low ICAP input enable.
I[31:0]	Input	32	Configuration data input bus.
O[31:0]	Output	32	Configuration data output bus. If no data is being read, contains current status.
PRDONE	Output	1	Partial Reconfiguration complete. Default is High. Goes Low when FDRI packet is seen, and goes back High when DESYNC is seen and EOS is High.
PRERROR	Output	1	Partial Reconfiguration error. Default is Low. Goes High when partial configuration bitstream error is detected. Can be reset by loading RCRC command.
RDWRB	Input	1	Read (Active High) or Write (Active Low) Select input.

Attributes

Table 7-9 describes the describes the ICAPE3 primitive attributes.

Table 7-9: **ICAPE3 Attributes**

Attribute	Type	Allowed Values	Default	Description
DEVICE_ID	Hex	All UltraScale FPGA IDCODE values	32'h03628093	Specifies the fixed Device IDCODE value to be used for simulation purposes.
ICAP_AUTO_SWITCH	String	Disable, Enable	Disable	Enable switching ICAP between top and bottom (not recommended).
SIM_CFG_FILE_NAME	String	<*Filename*>	None	Specifies the Raw Bitstream (RBT) file to be parsed by the simulation model.

ICAPE3 Resources

RECOMMENDED: *Only one ICAPE3 resource should be instantiated for an UltraScale FPGA, and it should be automatically placed by the tools.*

Each FPGA should be used as if it had one ICAPE3 resource, although there are actually two ICAPE3 resources per die for improved SEU protection. The tools automatically use the top ICAPE3 by default. Advanced users can use the write_bitstream option BITSTREAM.Readback.ICAP_Select to select the bottom resource. Control register 0 Bit 30 (ICAP_SELECT) enables the top ICAPE3 site when set to 0 (default), and enables the bottom ICAPE3 site when set to 1. User switching can be done by toggling CTL0<30> using the currently active ICAPE3. The device can automatically switch between the two ICAPE3 sites if enabled using the ICAP_AUTO_SWITCH attribute, using a sync word on 8 LSBs.

For 3D ICs based on Stacked Silicon Interconnect (SSI) technology, the ICAPE3 of one Super Logic Region (SLR) is defined as the master, with the ability to read from and write to all other SLRs. The tools automatically place an instantiated ICAPE3 in the correct master SLR. Note that when the mode pins are set to JTAG mode, the master SLR ICAP cannot access the slave SLRs. Because JTAG mode is always available, the mode pins do not need to be set to JTAG mode for configuration.

MASTER_JTAG

MASTER_JTAG provides control of the JTAG port from the FPGA logic, overriding the external pins. This is a new feature in the UltraScale architecture-based FPGAs. When MASTER_JTAG is instantiated, the external JTAG port is disabled at the end of configuration startup (EOS). Therefore MASTER_JTAG should not be instantiated except for a design requiring internal access to the JTAG port. This is intended only for eFUSE programming in advanced secure applications that cannot use the standard eFUSE programming methodologies. This component can be used for AES key programming (BBRAM or eFUSE),

USER eFUSE programming during runtime, or where external JTAG access is prohibited. Because the external JTAG port is disabled, MASTER_JTAG prevents the use of the Vivado® device programmer and the Vivado logic analyzer. For 3D ICs, MASTER_JTAG provides access only to the SLR in which it is instantiated, with an instruction register length of 6 bits.

Primitive

Figure 7-5 shows the MASTER_JTAG primitive.

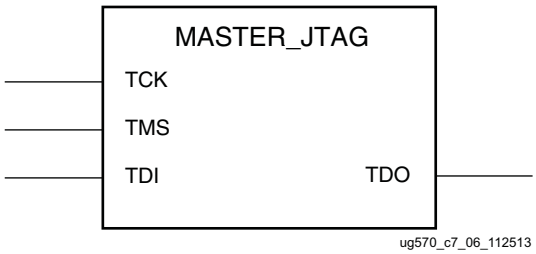


Figure 7-5: **MASTER_JTAG Primitive**

Pin Descriptions

Table 7-10 describes the MASTER_JTAG primitive pins.

Table 7-10: **MASTER_JTAG Pin Descriptions**

Pin	Type	Width	Description
TCK	Input	1	JTAG TCK clock pin.
TDI	Input	1	JTAG TDI input pin.
TDO	Output	1	JTAG TDO output pin.
TMS	Input	1	JTAG TMS input pin.

STARTUPE3

The STARTUPE3 design element is used to connect to selected dedicated configuration pins (located in bank 0). Control of dedicated configuration pins allows post-configuration access to the flash. When the flash is only used for configuration the FPGA design does not require the STARTUPE3.

For multi-purpose configuration pins located in bank 65, standard user logic can be implemented to connect to the pins required for access to the flash, with appropriate location constraints. For example, when using x8 or wider configuration modes, the STARTUPE3 is only used for the four LSBs of the configuration bus, D[03:00] located within bank 0. The higher order pins D[xx:04] can be directly connected as part of the user design.

The STARTUPE3 primitive for the UltraScale architecture-based FPGAs does not provide specification of the startup clock as was done in the STARTUPE2 for the 7 series. Otherwise, STARTUPE3 is a superset of STARTUPE2, and designs are retargeted automatically. The STARTUPE3 adds the ability to control the D00-D03 pins and the FCS_B pin as these pins are now in the dedicated configuration bank. The bidirectional D00-D03 pins have separate input and output connections to the STARTUPE3. Additional configuration pins can be controlled after configuration as standard I/O, including bidirectional I/O.

For devices based on Stacked Silicon Interconnect (SSI) technology, a single STARTUPE3 in the design is implemented in the master SLR and is automatically replicated to the other SLRs to provide global control of the device.

Primitive

Figure 7-6 shows the STARTUPE3 primitive.

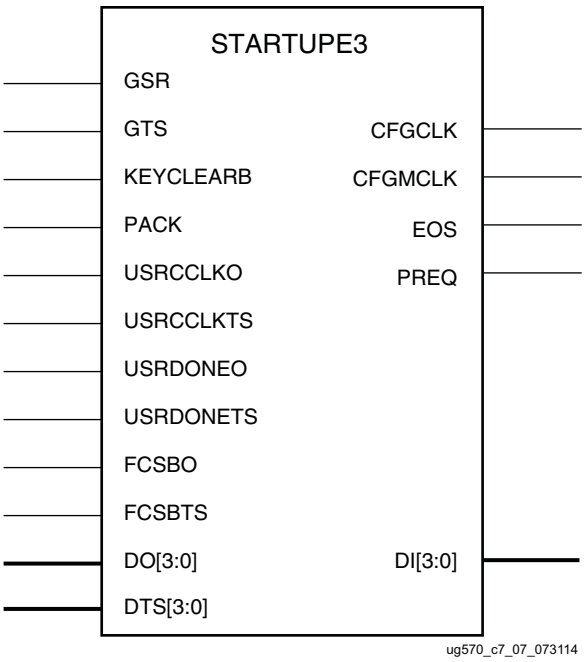


Figure 7-6: **STARTUPE3 Primitive**

Pin Descriptions

Table 7-11 describes the STARTUPE3 primitive pins. The three-state controls default to `1` to disable the outputs, KEYCLEARB defaults to a `1`, and the other inputs default to `0`.

Table 7-11: **STARTUPE3 Pin Descriptions**

Pin	Type	Width	Description
CFGCLK	Output	1	Configuration main clock output to the FPGA logic. CFGCLK reflects the signal on the CCLK pin, outputting a clock signal where the frequency is defined by the bitstream option for Configuration Rate. The output is active only during configuration, and in master modes with Persist enabled.
CFGMCLK	Output	1	Configuration internal oscillator clock output to the FPGA logic. CFGMCLK outputs a clock signal that is sourced from the FPGA internal oscillator. See the respective data sheet ([Ref 8] or [Ref 9]) for typical values.
DI(3:0)	Output	4	External D[03:00] configuration pin inputs that can be routed from STARTUPE3 to FPGA logic.
DO(3:0)	Input	4	FPGA logic signals that can be routed to STARTUPE3 to connect to the external D[03:00] configuration pins.
DTS(3:0)	Input	4	Three-state control of external D[03:00] output pins.
EOS	Output	1	Active High signal indicating the End Of Startup. EOS is an output into the FPGA logic. This output echoes the configuration logic EOS flag into the FPGA logic. EOS can be used as a reset signal.
FCSBO	Input	1	FPGA logic signal to external FCS_B configuration pin. FCSBO allows user control of FCS_B pin for Flash access.
FCSBTS	Input	1	Three-state control of external FCS_B output pin.
GSR	Input	1	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. ***Note:*** GSR cannot be used for the signal name.
GTS	Input	1	Global 3-state control. GTS is an active-High input from the FPGA logic. When this input is asserted High, all user I/Os except for configuration banks are put into a high-Z state. The GTS is automatically asserted during configuration and does not need to be asserted by the user. For most applications, this port should be tied Low. ***Note:*** GTS cannot be used for the signal name.
KEYCLEARB	Input	1	Clear AES Decrypter Key input from battery-backed RAM (BBRAM). KEYCLEARB is an input from the FPGA logic. This pin, when held Low, erases the contents of the decryption keys from the battery-backed RAM. KEYCLEARB can be triggered by a monitor for suspicious on-chip activity that may indicate an attack on the system.

Table 7-11: **STARTUPE3 Pin Descriptions *(Cont'd)***

Pin	Type	Width	Description
PACK	Input	1	PROGRAM_B pin or PROGRAM instruction ACKnowledge. PACK is an input from the FPGA logic. This pin *acknowledges* the assertion of the external PROGRAM_B signal or internal instruction 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. PACK can be tied Low for safe operations.
PREQ	Output	1	PROGRAM_B pulse, JPROGRAM, IPROG, or FALLBACK REQuest to FPGA logic. PREQ is an output into the FPGA logic. This pin is the *request* from the PROGRAM_B state machine to reset the device. This allows the assertion of the PROGRAM_B request to be intercepted and gated until the design is in a state where the reset can be completed. For example, you may want to hold off device re-configuration to allow non-resettable data elements to be cleared. This pin is only enabled if the PROG_USR attribute is set. PREQ can be left open/floating for safe operation.
USRCCLKO	Input	1	User CCLK input. USRCCLKO is an input from the FPGA logic. USERCCLKO drives a custom, FPGA-generated clock frequency onto the external FPGA CCLK pin. This is useful for post-configuration access of external flash devices. 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. 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.
USRCCLKTS	Input	1	User CCLK 3-state enable input to CCLK pin. USRCCLKTS is an input from the FPGA logic. When USRCCLKTS is High, the external FPGA CCLK pin is put into a high-Z state. Generally, USERCLKTS should be tied Low to prevent the CCLK pin going to a high-Z state.
USRDONEO	Input	1	DONE pin output signal. USRDONEO is an input from the FPGA logic. This pin directly drives the external FPGA DONE pin.
USRDONETS	Input	1	DONE 3-state enable output to DONE pin. USRDONETS is an input from the FPGA logic. When this input is High, DONE is put into a high-Z state. Generally, this pin should be tied Low. Tying USRDONETS High inhibits the assertion of DONE.

Attributes

Table 7-12 describes the describes the STARTUPE3 primitive attributes.

Table 7-12: **STARTUPE3 Attributes**

Attribute	Type	Allowed Values	Default	Description
PROG_USR	String	FALSE, TRUE	FALSE	Activate PROGRAM Request/ Acknowledge security feature.
SIM_CCLK_FREQ	Float (ns)	0.0 to 10.0	0.0	Set the Configuration Clock Frequency (ns) for simulation.

PROG_USR Attribute Description

The PROG_USR attribute allows the design to intercept re-programming requests. This can be helpful to allow the FPGA to complete an operation or to clear register states in dedicated resources such as the transceivers, before continuing with re-configuration by asserting PACK.

STARTUPE3 Connections to Dedicated Pins

A key feature of the STARTUPE3 component is to connect the user design to some of the dedicated configuration pins. These pins can be useful for accessing configuration flash as part of the user design. The STARTUPE3 represents connections to the external pins, so an input to the STARTUPE3 is a connection to an output pin or bidirectional pin, and an output from STARTUPE3 is the bidirectional pin input back into the user design (see Figure 7-7).

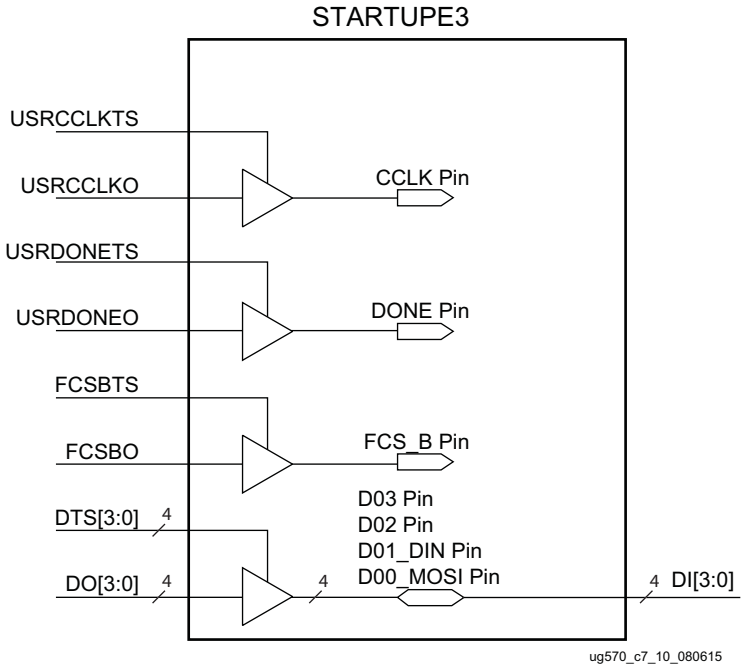


Figure 7-7: **STARTUPE3 Connections to Dedicated Pins**

Timing Considerations for Flash Connections

STARTUPE3 represents combinatorial connections, and does not contain any registers. Registers must be placed outside STARTUPE3. The USRCCLKO input to STARTUPE3 does not synchronize logic inside the block; it is a direct combinatorial connection to the CCLK pin after configuration. The USRCCLKO input would typically come from a global clock resource in the FPGA to synchronize the post-configuration interface to the flash. When using parallel NOR flash (BPI configuration), CCLK control is needed only if the flash supports synchronous transfers.

The STARTUPE3 does not support input or output delay constraints. As a result care should be taken to consider the performance requirements. The performance calculations are similar to those provided for calculating configuration frequency (see Equation 2-1 for SPI mode and Equation 4-1 through Equation 4-4 for BPI mode), but with the additional delays to and through the STARTUPE3 block. The delays between the STARTUPE3 ports and the device pins are noted in the data sheets [Ref 8] and [Ref 9]]. Because timing constraints are not supported for STARTUPE3, constrain the routing connected to the STARTUPE3 ports.

The flash clock Low to output valid time (T_{SPITCO} for SPI mode) must take into account the CCLK delay through the STARTUPE3, $T_{USRCCLKO}$. For parallel NOR flash where transfers are done synchronously, T_{CHQV} is needed to add $T_{USRCCLKO}$.

Similarly, the FPGA data setup time (T_{SPIDCC} for SPI mode) on D[03:00] is delayed by the setup time from the pins to the STARTUPE3 DI ports (T_{DI}) plus the routing delays from the STARTUPE3 DI port outputs to the slice flip-flops used. For the three types of asynchronous transfers for parallel NOR flash used in BPI configuration, the output delays for address (T_{BPICCO}) need to be added to the input delay constraints and any flash delays (e.g. T_{APA}, T_{ACC}). The output delay for address can be obtained by timing analysis.

Higher-order data pins D[xx:04] are routed directly to general-purpose I/O pins, so the delays can be constrained using standard input and output timing constraints. When setting input delays for serial NOR flash used in SPI mode, the clock polarity of the FPGA design must be taken into account. Data from the serial NOR flash device is launched off the falling edge of the clock.

USR_ACCESSE2

The USR_ACCESSE2 design element enables access to the 32-bit AXSS register within the configuration logic. This enables FPGA logic to access static data that can be set from the bitstream. The primitive and functionality for the UltraScale architecture-based FPGAs are identical to that for the 7 series.

The USR_ACCESSE2 register AXSS can be used to provide a single 32-bit constant value to the FPGA logic. The register contents can be defined during bitstream generation, avoiding the need to re-compile the design as would be required if distributed RAM was used to hold

the constant. A constant can be used to track the version of the design, or any other information you require. This is an alternative to the JTAG USERCODE instruction, which reads a 32-bit value defined by the write_bitstream option BITSTREAM.Config.UserID. USR_ACCESSE2 has the advantage of being directly accessible by the FPGA logic, and can store an automatically generated timestamp.

The contents of the USR_ACCESSE2 register AXSS can be defined with the write_bitstream option BITSTREAM.Config.USR_ACCESS, which can be set to NONE (default all zeroes), any 8-character hex value, or TIMESTAMP.

TIMESTAMP inserts the current timestamp into the AXSS register in this format:

```
ddddd_MMMM_yyyyyy_hhhhh_mmmmmm_ssssss
(bit 31) ..... (bit 0)
```

Where:

```
ddddd  = 5 bits to represent days 1-31 in a month
MMMM   = 4 bits to represent months 1-12 in a year
yyyyyy = 6 bits to represent years 0-63 (2000 to 2063)
hhhhh  = 5 bits to represent hours 0-23 in a day
mmmmmm = 6 bits to represent minutes 0-59 in an hour
ssssss = 6 bits to represent seconds 0-59 in a minute
```

For more details on USR_ACCESSE2, see *Bitstream Identification with USR_ACCESS using the Vivado Design Suite* (XAPP1232) [Ref 17].

Primitive

Figure 7-8 shows the USR_ACCESSE2 primitive.

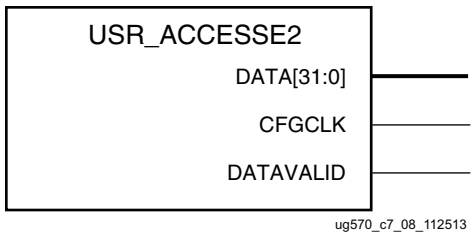


Figure 7-8: **USR_ACCESSE2 Primitive**

Pin Descriptions

Table 7-13 describes the USR_ACCESSE2 primitive pins.

Table 7-13: **USR_ACCESSE2 Pin Descriptions**

Pin	Type	Width	Description
CFGCLK	Output	1	Configuration Clock
DATA[31:0]	Output	32	Configuration Data reflecting the contents of the AXSS register
DATAVALID	Output	1	Active High Data Valid

USR_ACCESSE2 Advanced Uses

Since USR_ACCESSE2 data is stored in the AXSS Configuration register, the data can be dynamically updated through JTAG commands or through the use of ICAPE3. See Configuration Packets, page 163 for the form of packets that write to the AXSS register. For dynamic updates of the AXSS register, the waveforms from the USR_ACCESSE2 primitive that reflect the update events are shown in Figure 7-9.

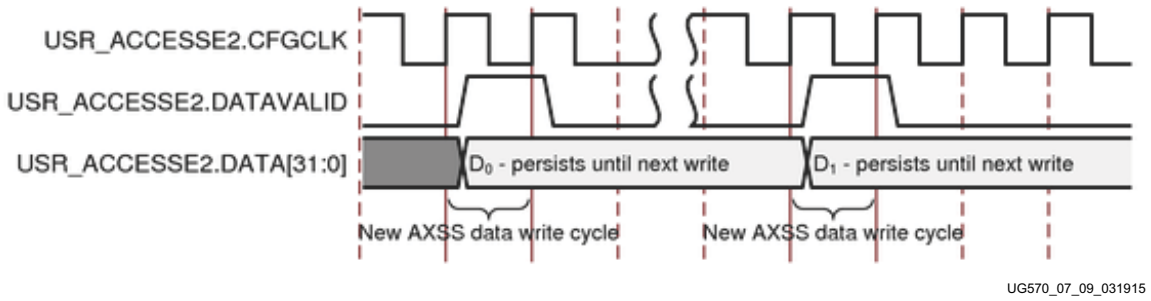


Figure 7-9: **USR_ACCESSE2 Update**

XILINX

Chapter 8

Bitstream Security, eFUSEs, and Device DNA

Introduction

This chapter discusses the available types of FPGA bitstream security including:

- Readback Security
- Bitstream Encryption and Authentication
- eFUSE
  - For bitstream key storage
  - For a user ID
  - For the pre-configured Device DNA

Readback Security

By default, an active FPGA configuration can be read back or reconfigured through the JTAG port, through the SelectMAP port if Persist is selected, or through the ICAPE3 primitive if it is instantiated in a design. A basic form of security is to prevent access to the configuration logic, such as by not allowing the configuration port to persist and not enabling ICAP connections to external pins. In addition, the bitstream readback security setting (BITSTREAM.READBACK.SECURITY) can be set to Level1 (disables readback), or Level2 (disables both readback and reconfiguration). The only way to remove a readback security setting in a configured FPGA is to clear the FPGA program by asserting PROGRAM_B or cycling power. If the user design is sensitive, bitstream encryption should be considered. Use of encryption automatically prevents readback via hardware gates and not just bitstream settings. It is the strongest method to prevent readback and protect your IP. The bitstream readback security setting does not affect readback for SEU detection. Refer to *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for details on the readback security options.

Bitstream Encryption and Authentication

The UltraScale architecture-based FPGAs have on-chip Advanced Encryption Standard (AES) decryption and authentication logic to provide a high degree of design security. Without knowledge of the encryption key, adversaries cannot analyze an externally intercepted bitstream to modify or clone the design. Encrypted FPGA designs cannot be copied or reverse-engineered.

The 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 Vivado® tools, the user generates the encryption key and the encrypted bitstream. UltraScale architecture-based FPGAs store the encryption key internally in either dedicated RAM, backed up by a small externally connected battery, or in the nonvolatile, one-time-programmable eFUSE. The selected option is defined with BITSTREAM.ENCRYPTION.ENCRYPTKEYSELECT set to BBRAM or EFUSE. The encryption key can only be programmed onto the device through the JTAG port, and cannot be read back.

During configuration, the FPGA device performs the reverse operation, decrypting the incoming bitstream. The 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. The AES decryption logic is not available to the user design and cannot be used to decrypt any data other than the configuration bitstream.

Although the AES-GCM algorithm is a self authenticating algorithm, it does so with a symmetric key, meaning that the key to encrypt is the same as the one to decrypt. This key must be protected as it is secret (hence storage to internal key space). However, if only authentication is desired, the UltraScale architecture provides for an alternative form of authentication in the form of RSA-2048. RSA is an asymmetric algorithm, meaning that the key to verify is not the same key used to sign. The verification is done with a public key. This key is public and does not need to be protected and does not need special secure storage. If desired, this form of authentication can be used in conjunction with encryption to provide both authenticity and confidentiality. See RSA Authentication, page 131.

For the step-by-step process to generate an encrypted bitstream and encryption keys using the Vivado Design Suite, see *Using Encryption and Authentication to Secure an UltraScale/UltraScale+ FPGA Bitstream* (XAPP1267) [Ref 19].

AES System Overview

The FPGA encryption system uses the AES-GCM (Advanced Encryption Standard - Galois/Counter Mode) authenticated encryption algorithm. The AES-GCM standard 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). An advantage of the AES-GCM algorithm is that it also supports built-in authentication.

The FPGA AES encryption system uses a 256-bit encryption key 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 the AES algorithm use the same key for encryption and decryption. The security of the data is therefore dependent on the secrecy of the key.

Rolling Keys

UltraScale FPGAs allow you to break up the bitstream into multiple AES encryption modules, each encrypted with its own unique key. The initial key is stored on-chip, while keys for each successive module are encrypted (wrapped) in the previous module. This feature, known as rolling keys, increases security against side-channel attacks such as differential power analysis (DPA). The bitstream option BITSTREAM.ENCRYPTION.KEYLIFE defines the number of encryption blocks per key. An encryption block is 128 bits (four 32-bit words). Fewer encryption blocks per key offers greater security but exponentially increases bitstream size and therefore configuration time. Selecting a value such as 1,024 or higher increases configuration size by about 15%, a value of 64 can increase bitstream size by 50%, and a value of 32 can double the bitstream size.

When using RSA authentication, certain block RAMs might be used to hold interim rolling keys, which impacts the ability to initialize those blocks. For a given block RAM column, each 36K block that resides in the bottom of a clock region is affected; essentially the first 36K block RAM starting at the bottom of a device and then every 12th 36K block RAM after that in a column (BRAM36_X*Y0, BRAM36_X*Y12, BRAM36_X*Y24, etc.). Those block RAMs can not be initialized to user-defined values when using RSA authentication. Those block RAMs are always initialized to 0 after configuration.

Creating an Encrypted Bitstream

Bitstream generator write_bitstream, provided with the Vivado tools, can generate encrypted as well as non-encrypted bitstreams. For AES bitstream encryption, select the option to enable bitstream encryption, and specify a 256-bit key as an input to the bitstream generator. The bitstream generator in turn generates an encrypted bitstream file (BIT) and an encryption key file (NKY).

To create an encrypted bitstream, the Tcl command "set_property BITSTREAM.ENCRYPTION.ENCRYPT Yes" is used. For specific bitstream generator commands and syntax, see *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7].

Loading the Encryption Key

The encryption key can only be loaded onto a device through the JTAG interface. The Vivado Device Programmer tool 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. 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 the RAM key cannot be reprogrammed without clearing the entire device. The key-access mode is transparent to most users.

The key can be programmed into the battery-backed RAM (BBRAM), which is powered by V_{CCAUX} or V_{BATT}, or into nonvolatile, one-time-programmable eFUSE bits. After programming, a CRC can be applied to verify proper programming of the key, but the key itself cannot be read back.

The encryption key itself can be encrypted using a fixed key that is never visible in the device. Encrypting the key is known as black key store (BKS) or key obfuscation. This option is disabled by default, and is set with the bitstream property BITSTREAM.ENCRYPTION.OBFUSCATEKEY ENABLE. When you set the BITSTREAM.ENCRYPTION.OBFUSCATEKEY property, the Vivado tool bitstream software creates a new key, ObfuscateKey, in the output NKY file. This obfuscated key is created by encrypting your AES-256 key with a metalized family key stored in the silicon. All FPGAs in the UltraScale family share the same family key. All FPGAs in the UltraScale+ family share the same family key, which is different than the UltraScale family key.

Xilinx does not provide the family key as part of the Vivado tools. Customers must send a request and must specify either the UltraScale family key or the UltraScale+ family key to secure.solutions@xilinx.com. The corresponding family key will then be distributed to qualified customers through the Product Licensing site on www.xilinx.com.

To specify the location of the family key you must set the following write_bitstream property: `set_property BITSTREAM.ENCRYPTION.FAMILY_KEY_FILEPATH C:/<anyDirectory>/familyKey_us.cfg [current_design]`.

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 PROGRAM_B or power-on reset (after a power cycle) 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

readback security setting). The encryption key still cannot be read out of the device, preventing the use of Trojan Horse bitstreams to defeat the FPGA encryption scheme.

An encrypted bitstream can be delivered through any configuration interface: JTAG, serial, SPI, BPI, SelectMAP, and ICAP. For encrypted bitstreams using an obfuscated key with the JTAG interface, do not pause bitstream loading by temporary excursion from the JTAG Shift-DR state to the JTAG Pause-DR state. Instead, stay within the JTAG Shift-DR state and stop the JTAG TCK clock to pause bitstream loading. For encrypted bitstreams using an obfuscated key with the SelectMAP or ICAP interfaces, do not pause bitstream loading by temporary de-assertion of the configuration interface chip-select (CSI_B). Instead, keep CSI_B asserted and stop the CCLK to pause bitstream loading. See Answer 73656 for details.

Bitstreams can be created with both compression and encryption. 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 even when encryption is turned on. Readback is available through the ICAPE3 primitive. None of these events resets the BBRAM key if V_{BATT} or V_{CCAUX} is maintained.

A mismatch between the key in the encrypted bitstream and the key stored in the device causes configuration to fail with the INIT_B pin pulsing Low and then back High if fallback is enabled, and the DONE pin remaining Low.

IMPORTANT: *Clear or Program the BBRAM to a known state before attempting to configure with an encrypted bitstream that uses the BBRAM as the key source. If you attempt to download an encrypted bitstream on power-up before the BBRAM key is programmed, the FPGA device might lock up. You must power-cycle the device and then load the BBRAM key before configuring with an encrypted bitstream.*

Bitstream Encryption and Internal Configuration Access Port (ICAP)

The Internal Configuration Access Port (ICAP), defined by the ICAPE3 primitive, provides the user logic with access to the FPGA configuration interface. The ICAP interface is similar to the SelectMAP interface, although the restrictions on readback for the SelectMAP interface do not apply to the ICAP interface after configuration. Users can send a partial bitstream, whether encrypted or unencrypted, or perform readback through the ICAP interface, even if bitstream encryption is used. Unless the designer wires the ICAPE3 primitive to user I/O, this interface does not offer attackers a method for defeating the FPGA AES encryption scheme.

Users concerned about the security of their design should not wire the ICAPE3 interface to user I/O. Connecting the ICAPE3 clock does not impact security.

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

VBATT

When an encryption key is stored in the FPGA battery-backed RAM (BBRAM), the encryption key memory cells are volatile and must receive continuous power to retain their contents. During normal operation, these memory cells are powered by the auxiliary voltage input (V_{CCAUX}), although a separate V_{BATT} power input is provided for retaining the key when V_{CCAUX} is removed. Because V_{BATT} draws very little current (on the order of nanoamperes), a small watch battery is suitable for this supply. To estimate the battery life, refer to V_{BATT} DC Characteristics in the respective data sheet ([Ref 8] or [Ref 9]) and the battery specifications.

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

Bitstream Authentication

The AES-GCM encryption standard also supports built-in authentication, enhancing security and eliminating the need to specify a separate HMAC key as in the 7 series FPGAs. Without knowledge of the AES-GCM key, the bitstream cannot be loaded, modified, intercepted, or cloned. Encryption provides the basic design security to protect the design from copying or reverse engineering, while authentication provides assurance that the bitstream provided for the configuration of the FPGA was the unmodified bitstream allowed to load. Authentication verifies both data integrity and authenticity of the bitstream. Authentication covers the entire bitstream for all types of control and data. Any bitstream tampering including single bit flips are detected.

If authentication passes, the configuration goes to completion through the startup cycle. If authentication fails 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.

RSA Authentication

The AES-GCM algorithm implements authentication and decryption at the same time. However, an alternative security method is to authenticate the bitstream data before it is sent to the decryptor. This method can be used to help prevent attacks on the decryption engine itself by making sure the data is authentic before performing any decryption. UltraScale architecture-based FPGAs support RSA-2048 authentication for this purpose.

RSA authentication is not supported in the Kintex UltraScale KU025 device, or when using serial or selected other configuration modes in the Kintex UltraScale and Virtex UltraScale FPGAs (see Table 8-1). For RSA authentication there are no configuration mode limitations in the Kintex UltraScale+ and Virtex UltraScale+ FPGAs.

Table 8-1: **UltraScale Devices and Configuration Modes Supporting RSA Authentication**

Interface	Width	Kintex UltraScale FPGAs				Virtex UltraScale FPGAs			Kintex UltraScale+ and Virtex UltraScale+ FPGAs
		KU025	KU035 KU040	KU060 KU085 KU115	KU095	VU080 VU095	VU065 VU125 VU160 VU190	VU440	
SelectMAP	32	N/A	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes
	16	N/A	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes
	8	N/A	No	No	Yes(1)	Yes(1)	No	Yes(1)	Yes
BPI	16	N/A	Yes	Yes(2)	Yes	Yes	Yes	Yes	Yes
	8	N/A	No	No	Yes(2)	Yes(2)	No	Yes	Yes
SPI	8	N/A	No	No	Yes	Yes	No	Yes	Yes
	4	N/A	No	No	No	No	No	Yes	Yes
	2	N/A	No	No	No	No	No	No	Yes
	1	N/A	No	No	No	No	No	No	Yes
JTAG	1	N/A	No	No	No	No	No	No	Yes
Serial	1	N/A	No	No	No	No	No	No	Yes

Notes:

1. Not supported if non-continuous SelectMAP data loading is implemented by deasserting the CSI_B signal.
2. Not supported if asynchronous page read is used.

RSA authentication is enabled with the bitstream properties `BITSTREAM.AUTHENTICATION.AUTHENTICATE` and `BITSTREAM.AUTHENTICATION.RSAPRIVATEKEYFILE`. RSA authentication can be used independent of bitstream encryption, meaning it can authenticate either an unencrypted or encrypted bitstream. The RSA configuration control logic reads the encrypted bitstream, including a public key and bitstream signature, into the device memory. The RSA configuration control logic then instructs the RSA engine to calculate the expected digest based on the public key and signature. After the bitstream is buffered and the RSA engine has calculated the expected digest, the actual digest is compared against that result. If RSA authentication passes and the configuration was not encrypted, the FPGA is released for operation. If RSA authentication passes and the configuration data was encrypted, then the FPGA is released for decryption of the bitstream. If RSA authentication fails, an error equivalent to an AES-GCM authentication error is generated. At this point the device either locks down or, if enabled, a fallback occurs.

A device configured with an RSA authenticated bitstream can take up to three times as long to configure as a standard uncompressed bitstream for that device. The actual time increase is dependent upon the mode of configuration. RSA authentication cannot be used in conjunction with bitstream compression, partial reconfiguration, or tandem configuration over the PCIe interface.

eFUSE

eFUSEs are nonvolatile one-time-programmable (OTP) cells used for some device settings, the factory-programmed Device DNA, and these user-programmable elements:

- AES-GCM encryption key
- RSA authentication key
- EFUSE_USR user value
- Control and security settings

The fuse link is programmed (or burned or blown) by flowing a large current for a specific amount of time. The resistance of a programmed fuse link is typically a few orders of magnitude higher than that of a pristine or unprogrammed fuse. A programmed fuse is assigned a logic value of `1`, and a pristine fuse has a logic value of `0`. User-programmable eFUSEs can be programmed with the Xilinx configuration tools (see UG908 [Ref 7]). eFUSE must not be programmed during device configuration activity. When in-system programming eFUSE, apply the following to minimize system activity and FPGA interface activity to reduce risks from system noise on JTAG signals or eFUSE circuits:

- Avoid device configuration, configuration readback, and readback CRC
- Temporarily change configuration mode pin settings from a master mode setting to the JTAG only mode setting
- Disable system clock sources

The Device DNA is a 96-bit eFUSE value that is factory-programmed and unique for each device. JTAG or the DNA_PORTE2 primitive is used to access the value. See Device Identifier (Device DNA), page 137 for more details.

The JTAG interface can be used to program the FUSE_USER 32-bit value. JTAG or the EFUSE_USR primitive is then used to access the data. See EFUSE_USR in Chapter 7, Design Entry.

The FPGA logic can access only the FUSE_USER register and the Device DNA. All other eFUSE bits are not accessible from the FPGA logic.

OTP eFUSE Registers

In addition to eFUSE registers used by the FPGA for security settings and selected other options, there are user-accessible eFUSE registers controlled by JTAG instructions. Table 8-2 lists the eFUSE registers with their sizes and usage. Note that because these registers store values using eFUSE bits, they can only be programmed once.

Table 8-2: **OTP eFUSE Registers**

Register Name	Size (Bits)	Contents	Description
FUSE_RSA	384	Bitstream authentication key [383:0] (bit 0 shifted first)	Stores an SHA-3 hash of the public key used for RSA bitstream authentication.
FUSE_KEY	256	Bitstream encryption key [255:0] (bit 0 shifted first)	Stores a key for use by AES-GCM bitstream decryption and authentication. The eFUSE key can be used instead of the key stored in battery-backed RAM.
FUSE_DNA	96	Device identifier programmed by Xilinx [95:0] (bit 0 shifted first)	Unique device identifier bits [95:0], corresponding to the 96-bit read-only DNA_PORTE2 primitive value known as Device DNA. See Device Identifier (Device DNA), page 137.
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 Chapter 7, Design Entry). Depending on the read/write access bits in the CNTL register, the code can be programmed and read through the JTAG port.
FUSE_USER_128	128	User defined [127:0] (bit 0 shifted first)	Stores a 128-bit user-defined code. This register is readable from the JTAG FUSE_USER_128 instruction. The JTAG FUSE_USER_128 data register length is 384 bits in UltraScale FPGAs or 176 bits in UltraScale+ FPGAs. Only bits [127:0] are supported for user code storage, and the remaining bits are reserved and can be any value.
FUSE_CNTL	21	Control Bits [20:0] (bit 0 shifted first)	In UltraScale FPGAs this controls key use and read/write access to eFUSE registers. This register can be programmed and read through the JTAG port.
	24	Control Bits [23:0] (bit 0 shifted first)	In UltraScale+ FPGAs this key use and read/write access to eFUSE registers. This register can be programmed and read through the JTAG port.
FUSE_SEC	32	Security Control Bits [31:0] (bit 0 shifted first)	Controls encryption and authentication options. Depending on the read/write access bits in the CNTL register, this register can be programmed and read through the JTAG port.

JTAG Instructions

eFUSE registers can be read through JTAG ports. eFUSE programming can be done only via JTAG. Table 8-3 lists eFUSE-related JTAG instructions.

Table 8-3: **eFUSE-Related JTAG Instructions**

JTAG Instruction	Code	Action
FUSE_RSA	011000	Selects the FUSE_RSA register
FUSE_KEY	110001	Selects the FUSE_KEY register
FUSE_DNA	110010	Selects the FUSE_DNA register
FUSE_USER	110011	Selects the FUSE_USER register
FUSE_USER_128	011001	Selects the FUSE_USER_128 register
FUSE_CNTL	110100	Selects the FUSE_CNTL register
FUSE_SEC	111011	Selects the FUSE_SEC register

The FUSE_CNTL and FUSE_SEC control registers are described in Table 8-4 and Table 8-5, respectively. All register bits are defined by an eFUSE and therefore each selection is permanent.

Table 8-4: **OTP eFUSE Control Register (FUSE_CNTL)**

Bit Position	Name	Description
0	R_DIS_KEY	Disables the CRC check that verifies the key and programming of the FUSE_KEY encryption key.
1	R_DIS_USER	Disables reading of the FUSE_USER 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.
2	R_DIS_SEC	Disables reading the FUSE_SEC security settings.
3–4	Reserved	Reserved
5	W_DIS_CNTL	Disables programming of the FUSE_CNTL control settings.
6	R_DIS_RSA	Disables reading the FUSE_RSA authentication key.
7	W_DIS_KEY	Disables programming of the FUSE_KEY encryption key and the CRC check that verifies the key.
8	W_DIS_USER	Disables programming of the FUSE_USER user code.
9	W_DIS_SEC	Disables programming of the FUSE_SEC security settings.
10–14	Reserved	Reserved
15	W_DIS_RSA	Disables programming of the FUSE_RSA authentication key.
16	W_DIS_USER_128	Disables programming of the FUSE_USER_128 user code.
17–23	Reserved	Reserved UltraScale bits extend through bit 20 and reserved UltraScale+ bits extend through bit 23.

Table 8-5: **OTP eFUSE Security Register (FUSE_SEC)**

Bit Position	Name	Description	RMA Impact
0	FUSE_SHAD_SEC[0]	Only allow encrypted bitstreams.	RMA not accepted(1)
1	FUSE_SHAD_SEC[1]	For encrypted bitstreams, force use of AES key stored in eFUSE. When this bit is NOT programmed, encryption and the key source can be selected via bitstream options - the FPGA can be configured using an unencrypted bitstream, or a bitstream encrypted with a key value stored in battery-backed RAM or eFUSE.	-
2	RSA_AUTH	Forces RSA authentication.	RMA not accepted(1)
3	FUSE_SHAD_SEC[3]	Disables external JTAG pins.	RMA analysis limited(2)
4	SCAN_DISABLE	Disables Xilinx test access.	RMA analysis limited(2)
5	CRYPT_DISABLE	Disables decryptor.	-
6	FUSE_BKS_ENABLE	Enable key obfuscation.	-
7–31	Reserved	Reserve.	-

Notes:

1. **IMPORTANT!** When FUSE_SHAD_SEC[0] or RSA_AUTH is programmed, only AES encrypted or RSA authenticated bitstreams, respectively, 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 accept return material authorization (RMA) requests or support indirect flash programming for devices that have the FUSE_SHAD_SEC[0] or RSA_AUTH bit programmed.
2. **IMPORTANT!** If this bit is programmed, return material authorization (RMA) returns are limited in device analysis and debug.

TIP: *Zynq UltraScale+ devices have distinct eFUSE registers unlike the UltraScale architecture FPGA eFUSE registers. The UltraScale architecture FPGA eFUSE registers are not supported in Zynq UltraScale+ devices. For Zynq UltraScale+ device eFUSE registers, see the PS eFUSE section in the Zynq UltraScale+ Device Technical Reference Manual (UG1085) [Ref 27].*

Device Identifier (Device DNA)

The FPGA contains an embedded, device identifier (Device DNA). The identifier is nonvolatile, permanently programmed by Xilinx into the FPGA, and is unchangeable making it tamper resistant. Each device is programmed with a unique DNA value.

External applications can access the DNA value through the JTAG port and FPGA designs can access the DNA through a Device DNA Access Port (DNA_PORTE2).

The FPGA application accesses the identifier value using the Device DNA Access Port (DNA_PORTE2) design primitive, shown in Figure 8-1.

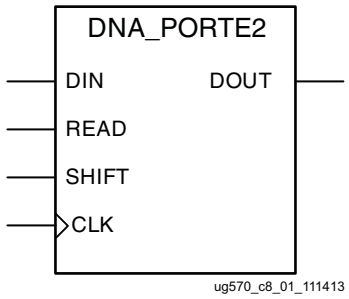


Figure 8-1: **FPGA DNA_PORTE2 Design Primitive**

Identifier Value and Operation

Figure 8-2 shows the general functionality of the DNA_PORTE2 design primitive. An FPGA application must first instantiate the DNA_PORTE2 primitive, shown in Figure 8-1, within a design. As shown in Figure 8-2, the Device DNA value is 96 bits long. The two LSBs and two MSBs have fixed values that can be used to detect the LSB and MSB of the 96-bit DNA.

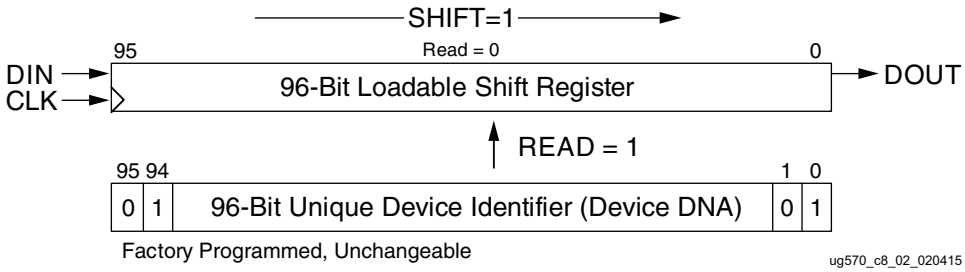


Figure 8-2: **DNA_PORTE2 Operation**

To read the Device DNA, the FPGA application must first transfer the identifier value into the DNA_PORTE2 output shift register. The READ input must be asserted during a rising edge of CLK, as shown in Table 8-6. This action parallel loads the output shift register with all 96 bits of the identifier. The LSB of the DNA value (DNA[0]=1) appears on DOUT

immediately after the load. The READ operation overrides a SHIFT operation, so READ should be asserted for at least one clock cycle and then removed.

Table 8-6: **DNA_PORTE2 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 96-bit ID	Bit 0 of Identifier
SHIFT	DIN	0	1	↑	Shift DIN into bit 95, shift contents of shift register toward DOUT	Bit 0 of shift register

Notes:
1. X = Don't care.
2. ↑ = Rising clock edge.

To continue reading the identifier values, assert SHIFT followed by a rising edge of CLK, as shown in Table 8-6. 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. All shift register functionality is synchronous to the CLK.

Extending Identifier Length

As shown in Figure 8-3, most applications that use the DNA_PORTE2 primitive tie the DIN data input to a static value.

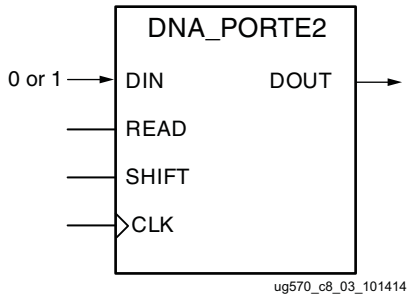


Figure 8-3: **Shift In Constant**

As shown in Figure 8-4, the length of the identifier can be extended by feeding the DOUT serial output port back into the DIN serial input port. This way, the identifier can be extended to any possible length.

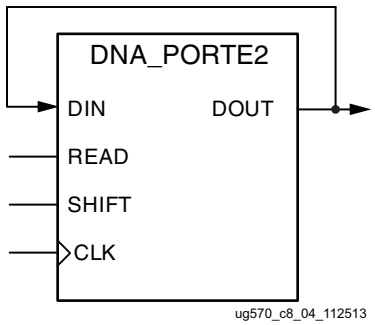


Figure 8-4: **Circular Shift**

It is also possible to add additional bits to the identifier using FPGA logic resources. As shown in Figure 8-5, the FPGA application can insert additional bits via the DNA_PORTE2 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.

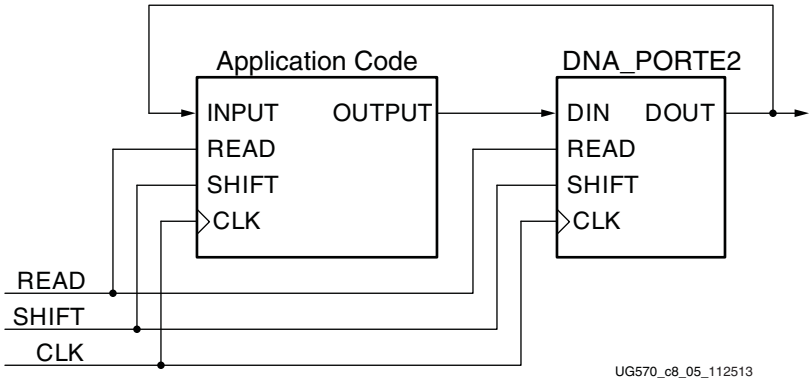


Figure 8-5: **Bitstream Specific Code**

JTAG Access to Device Identifier

The FPGA internal device identifier can be read via the JTAG port using the private FUSE_DNA command. Bit 0 of the identifier, shown in Figure 8-2, page 137, appears on the TDO JTAG output following the FUSE_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 Vivado Device Programmer also supports reading the Device DNA by viewing the eFUSE registers in the **Hardware Device Properties** window, or by using the following Tcl command:

```
report_property [lindex [get_hw_device] 0] REGISTER.EFUSE.FUSE_DNA
```

The user-defined eFUSE register FUSE_USER can be read similarly to FUSE_DNA, using the JTAG FUSE_USER command, the **Hardware Device Properties**, or reporting the `REGISTER.EFUSE.FUSE_USER` property. See *Vivado Design Suite User Guide Programming and Debugging* (UG908) [Ref 7] for more details.

XILINX

Chapter 9

Configuration Details

Introduction

You generally do not need to know the details of the configuration format and commands. However, this detail can be useful for debugging purposes. After initial configuration, you can send configuration commands to the device through the permanent JTAG interface, through the SelectMAP port if Persist is selected, or through the Internal Configuration Access Port if the ICAPE3 primitive is included in the design.

This chapter has the following sections:

- Configuration Data File Formats
- Configuration Sequence
- Clocking to End of Start-up
- Configuration Registers

Configuration Data File Formats

Xilinx design tools can generate configuration data files in several formats, as described in Table 9-1. The bitstream generator write_bitstream converts the post-implementation file into a configuration file or a bitstream. The write_cfgmem command converts one or more bitstream files into an MCS or other type of file. The MCS and other write_cfgmem files are known as PROM files, but the write_cfgmem 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 9-1: **Xilinx Configuration File Formats**

File Extension	Bit Swapping[1]	Xilinx Tool[2]	Description
BIT	Not bit swapped	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 Vivado® device programmer tool with a programming cable. Proprietary format for Vivado design tool use only.
RBT	Not bit swapped	write_bitstream -raw_bitfile	ASCII equivalent of the BIT file containing a text header and ASCII 1s and 0s. Eight bits per configuration bit. Proprietary format for Vivado design tool use only.
BIN	Not bit swapped	write_bitstream -bin_file	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 memories.
	Bit swapped	write_cfgmem -format BIN	
MCS	Bit swapped[3]	write_cfgmem -format MCS or Vivado device programmer	ASCII file format containing address and checksum information in addition to configuration data. Used mainly for device programmers and the Vivado device programmer tool.
HEX	Determined by user	write_cfgmem -format HEX or Vivado device programmer	ASCII 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 write_bitstream and write_cfgmem Tcl command syntax, refer to the Vivado Design Suite Tcl Command Reference Guide (UG835) [Ref 19].
3. MCS files are generally bit-swapped except in SPI or serial configuration mode. The write_cfgmem -interface SPIx1/2/4/8 option is used for serial NOR flash and creates a file that is not bit swapped.

Generating Configuration Memory Files

Configuration memory files are generated from bitstream files with the write_cfgmem utility. You can access write_cfgmem directly from the command line or indirectly through the Vivado device programmer. For write_cfgmem syntax, refer to the *Tcl Command Reference Guide* (UG835) [Ref 19]. The write_cfgmem command reformats bitstream files for flash memory programming and combines bitstream files for serial daisy chains.

The output from write_cfgmem is typically used to program the selected third-party flash memory device. The output format supported by your third-party programmer should be chosen. The write_cfgmem command -interface argument specifies the planned configuration interface. Valid values include SMAPx8 (default), SMAPx16, SMAPx32, SERIALx1, SPIx1, SPIx2, SPIx4, SPIx8, BPIx8, and BPIx16. This also determines if bit swapping is enabled or disabled (see Bit Swapping, page 143). Some parallel flash devices for BPI configuration require endian swapping to be enabled when creating the file. Refer to the flash vendor documentation.

Files for Serial Daisy Chains

Configuration data for serial daisy chains requires special formatting because separate BIT files cannot just be concatenated together to program the daisy chain. The special formatting is performed by write_cfgmem (or Vivado device programmer) when generating a file from multiple bitstreams. To generate the daisy chain file, specify multiple bitstreams following a single -loadbit argument in write_cfgmem; for example: `-loadbit "up|down <address1> <bitfile1.bit> <address2> <bitfile2.bit>"`. Refer to the tool documentation for details.

The write_cfgmem command 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.

Note: A daisy chain that includes UltraScale devices must be composed only of devices that are supported by the Vivado tools, from the 7 series and later.

Files for SelectMAP Configuration

For custom configuration solutions, the BIN and HEX files are the easiest file formats to use due to their raw data format. The MCS format is also supported for legacy applications. In some cases, additional formatting is required; refer to *Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave SelectMAP Mode* (XAPP583) [Ref 14], 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 file. Instead, the target memory device should be programmed with multiple BIN or HEX files. If a single file with multiple, separate data streams is needed, one can be generated in the Vivado device programmer by targeting a parallel memory, then selecting the appropriate number of data streams. This can also be accomplished through the write_cfgmem command line. Refer to the *Tcl Command Reference Guide* (UG835) [Ref 19] for details.

Bit Swapping

Bit swapping is the swapping of the bits within a byte. The MCS file format is always bit-swapped unless the write_cfgmem -interface SPIx1|SPIx2|SPIx4|SPIx8 option 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 memory 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 memory device.

Figure 9-1 shows how two bytes of data (0xABCD) are bit-swapped.

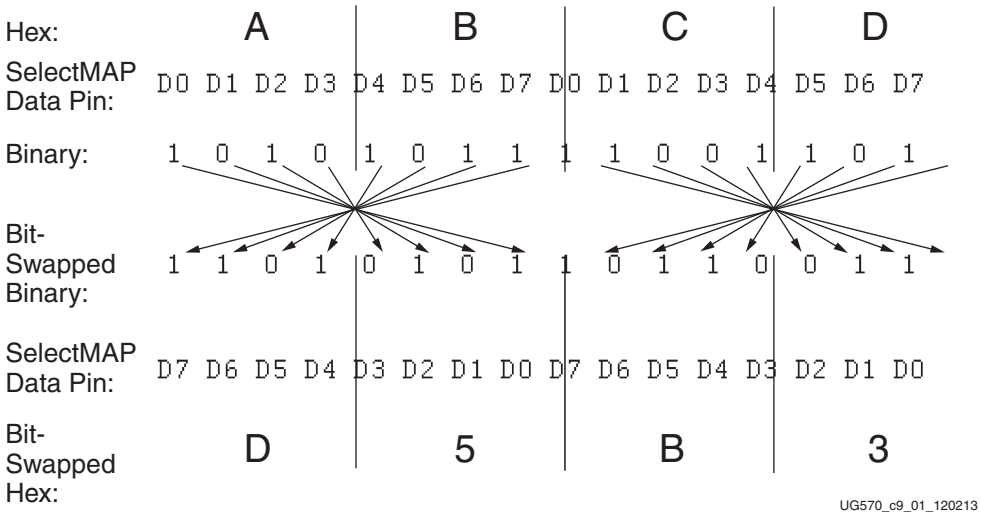


Figure 9-1: **Bit Swapping Example**

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 files, and for the ICAPE3 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 BPI x8 modes and to the ICAPE3 interface (see Bit Swapping). The bit swap rule is extended to x16 and x32 bus widths, that is, the data is bit swapped within each byte.

Table 9-2 and Table 9-3 show examples of a Sync word 0xAA995566 inside a bitstream (see Sync Word). 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 ICAPE3 interface.

Table 9-2: **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 9-3: **Sync Word Data Sequence Example for x8, x16, and x32 Modes**

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

Configuration Sequence

While each of the configuration interfaces is different, the basic steps for configuring a device are the same for all modes. Figure 9-2 shows the 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.

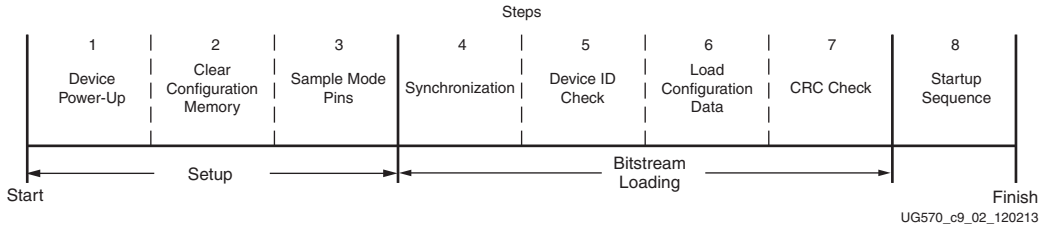


Figure 9-2: **FPGA Configuration Process**

The 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 9-3).

The setup steps are critical for proper device configuration. The steps include:

- Device Power-Up (Step 1)
- Clear Configuration Memory (Step 2, Initialization)
- Sample Mode Pins (Step 3)

Device Power-Up (Step 1)

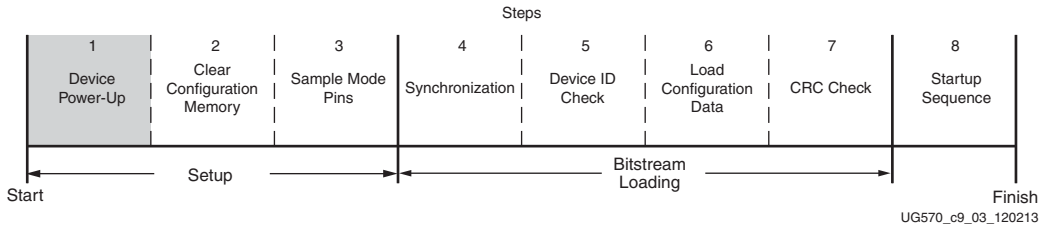


Figure 9-3: **Device Power-Up (Step 1)**

For configuration, devices require power on the V_{CCO_0}, V_{CCAUX}, V_{CCAUX_IO}, V_{CCBRAM}, V_{CCINT}, and V_{CCINT_IO} pins. Power sequencing requirements are described in the respective data sheet ([Ref 8] or [Ref 9]).

All JTAG and serial configuration pins are located in a separate, dedicated bank with a dedicated voltage supply (V_{CCO_0}). None of the I/O voltage supplies except V_{CCO_0} needs to be powered for FPGA configuration in JTAG or serial modes (up to SPI x4) when RS[1:0] is not used. 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.

The multi-function pins are located in bank 65. For all modes that use multi-function I/O (for example, master BPI, SPI x8, SelectMAP), the associated V_{CCO_65} 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. If the Persist option is used (see Persist Option, page 181), the multi-function I/O for the selected configuration mode remain active after configuration, with the I/O standard set to the default of LVCMOS, 12 mA drive, fast slew rate.

Table 9-4 shows the power supplies required for configuration. Table 9-5 shows the timing for power-up. Refer to the data sheet for voltage ratings. Standard I/O voltage levels supported for configuration are 1.5V, 1.8V, 2.5V, and 3.3V. None of the I/O voltage supplies except V_{CCO_0} needs to be powered for configuration in JTAG mode. When configuration modes are selected that use the multi-function pins (i.e., serial, master BPI, SPI, SelectMAP), V_{CCO_65} must also be supplied. In the Virtex UltraScale devices, and in the Kintex UltraScale KU095, bank 65 is an HP I/O bank, and therefore configuration interfaces requiring bank 65 must operate at 1.5V or 1.8V.

Table 9-4: **Power Supplies Required for Configuration**

Pin Name	Description
V_{CCINT}	Internal supply voltage.
V_{CCINT_IO}	Internal supply voltage for the I/O banks.
V_{BATT}[(1)]	AES decryptor key memory backup power supply; If the key memory is not used, you should tie this pin to V_{CCAUX} or GND.

Table 9-4: **Power Supplies Required for Configuration *(Cont'd)***

Pin Name	Description
V_{CCAUX}	Auxiliary supply voltage.
$V_{CCAUX\_IO\_\#}$	Auxiliary supply voltage for the I/O banks.
V_{CCBRAM}	Supply voltage for the block RAM.
V_{CCO_0}	Configuration bank supply voltage.
V_{CCO_65}	Multi-function configuration bank supply voltage.

Notes:

1. VBATT is required only when an AES key is stored in the FPGA battery-backed RAM for decryption of an encrypted bitstream.

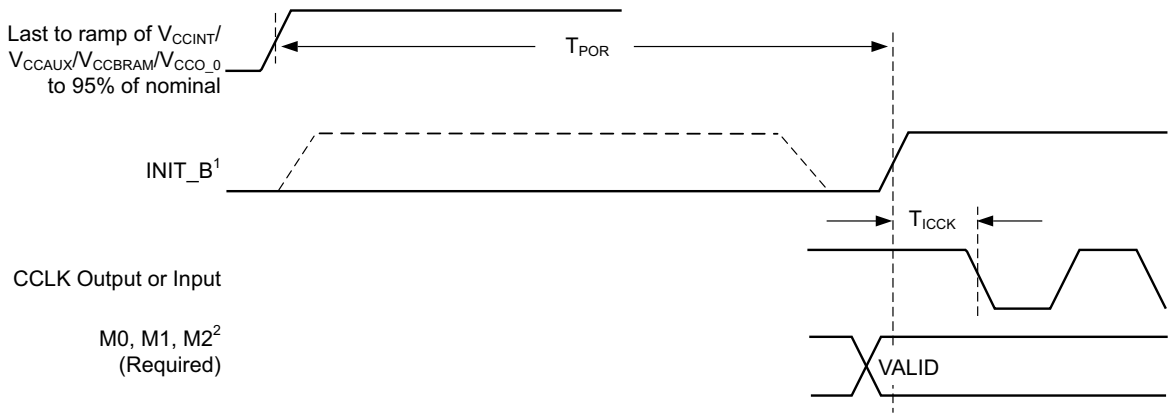
Table 9-5: **Power-Up Timing**

Symbol	Description
T_{PL}	Program latency.
T_{POR}	Power-on reset (POR).
T_{ICCK}	CCLK output delay.
$T_{PROGRAM}$	Program pulse width.

Notes:

1. See the data sheet for power-up timing characteristics.

Figure 9-4 shows the power-up waveforms.



Notes:

1. In UltraScale+ devices the INIT_B pin might be seen as High (because of external resistors on board tied to INIT_B) for a period of time after power ON. (The initial High time depends on the POR_OVERRIDE setting. With POR_OVERRIDE Low, the High time is approx. 40 ms. With POR_OVERRIDE High, the High time is approx. 9 ms.)
2. Can be either 0 or 1, but must not toggle during and after configuration.

UG570_c9_04_082819

Figure 9-4: **Device Power-Up Timing**

To ensure proper power-on behavior, the guidelines in the respective UltraScale family data sheet must be followed. Power supplies must rise monotonically within the specified ramp rate. If this is not possible, delay configuration by holding the INIT_B or PROGRAM_B Low (see Delaying Configuration) while the system power reaches the minimum recommended

operating voltages. The T_{POR} specification begins when the last of the monitored supplies (V_{CCINT}, V_{CCAUX}, V_{CCBRAM}, V_{CCO_0}) reaches 95% of its recommended operating condition voltage. The actual t_{POR} delay begins earlier depending on the thresholds of the monitored voltages, resulting in a smaller minimum specification with a slower ramp. Note that the recommended power-on sequence in the data sheet, to achieve minimum current draw and ensure that the I/Os are 3-stated at power-on, has V_{CCO} applied last. The T_{POR} time includes a built-in delay to allow for voltages to stabilize before beginning configuration. For applications where power-on time is important, the POR_OVERRIDE pin can be tied to V_{CCINT}, which shortens the built-in delay. See the data sheet for the resulting T_{POR} time when the supplies are ramped quickly and POR_OVERRIDE is tied to V_{CCINT}. Note that V_{CCINT} is recommended to ramp first. For the standard T_{POR} delay, tie POR_OVERRIDE to ground. See Power-On Reset, page 42.

UltraScale devices with multiple SLRs (this does not apply to Ultrascale+ devices) can have the weak pull-up temporarily enabled on I/Os in the Slave SLR during the configuration sequence (between power on and assertion of the INIT_B configuration signal). In some boards, this can cause an undesired `0-1-0` transition on I/O in the slave SLR. It is recommended that any I/O pins in the slave SLR sensitive to a `0-1-0` transition during configuration be connected to I/Os in the Master SLR or include external pull-downs of 1 kΩ or stronger to the pin.

Delaying Configuration

To delay configuration, the INIT_B or PROGRAM_B pin should be held Low during initialization (see Figure 9-4). When INIT_B has gone High, configuration cannot be delayed subsequently by pulling INIT_B Low.

The signals relating to initialization and delaying configuration are defined in Table 9-6.

Table 9-6: **Signals Relating to Initialization and Delaying Configuration**

Signal Name	Type	Access[1]	Description
INIT_B	Input, output, or open drain	Externally accessible through 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_B_INTERNAL_SIGNAL_STATUS	Status[2]	Internal signal, accessible through the FPGA status register	Indicates whether INIT_B signal is internally released.

Table 9-6: **Signals Relating to Initialization and Delaying Configuration**

Signal Name	Type	Access[1]	Description
MODE_STATUS[2:0]	Status	Internal signals, accessible through the FPGA status register	Reflects the values sampled on the mode pins when the status is read.
PROGRAM_B	Input	Externally accessible through the PROGRAM_B pin.	Before the Mode pins are sampled, PROGRAM_B is an input that can be held Low to delay configuration.

Notes:

1. Information on the FPGA status register is available in Table 9-25. Information on accessing the device status register through SelectMAP is available in Chapter 10, Readback Verification and CRC.
2. The status type is an internal status signal without a corresponding pin.

After power-up, the device can be re-configured by toggling the PROGRAM_B pin Low (see Figure 9-5).

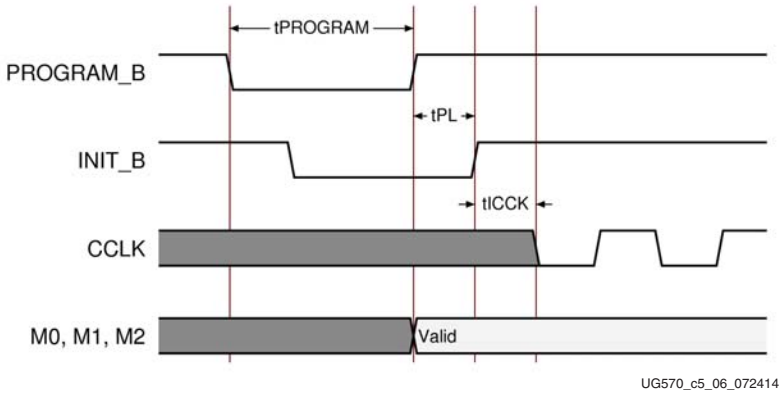


Figure 9-5: **Re-configuring the Device by Toggling the PROGRAM_B Pin Low**

Clear Configuration Memory (Step 2, Initialization)

Configuration memory (Figure 9-6) is cleared sequentially each time the device is powered up, after the PROGRAM_B pin is pulsed Low, after the JTAG JPROGRAM instruction or the IPROG command are used, or during a fallback retry configuration sequence. Block RAM and flip-flops can be initialized during configuration.

During this time, I/Os are drivers are disabled, except for the configuration and JTAG pins, through the use of the global three-state (GTS). User I/O pins are High-Z or pulled up depending on whether PUDC_B is High or Low, respectively. Both the dedicated configuration bank 0 and the multi-function bank 65 are enabled during configuration, independent of the mode pins. In devices based on SSI technology, banks 60 and 70 are also enabled during configuration, although they do not have configuration functions.

INIT_B is internally driven Low during initialization, then released after T_{POR} (Figure 9-4) for the power-up case, and T_{PL} for MultiBoot and fallback cases. If the INIT_B pin is held Low

externally, the device waits 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.

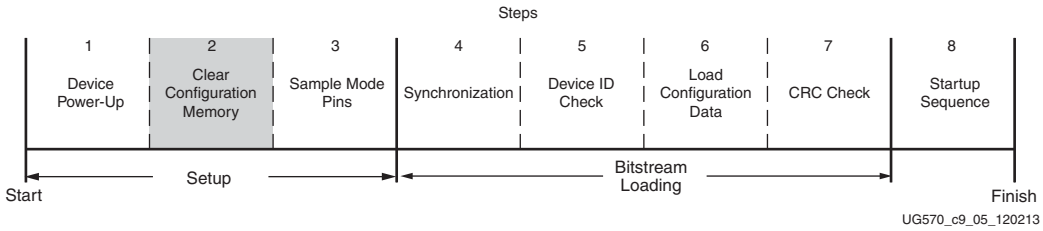


Figure 9-6: **Initialization (Step 2)**

Sample Mode Pins (Step 3)

When the INIT_B pin transitions to High, the device samples the M[2:0] mode pins (Figure 9-7) and begins driving CCLK if in the master modes. Then, the device begins sampling the configuration data input pins on the rising edge of the configuration clock.

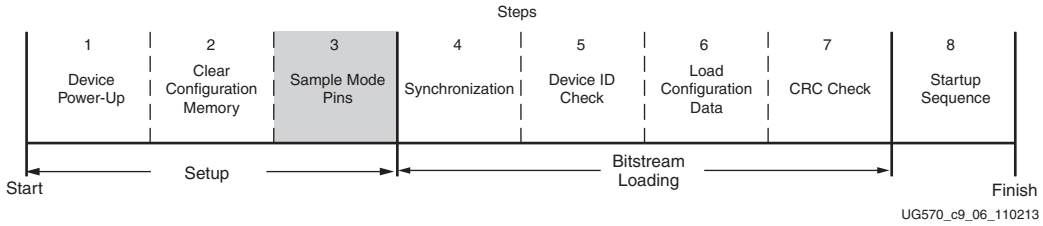


Figure 9-7: **Sample Mode Pins (Step 3)**

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 earlier chapters.

- Synchronization (Step 4)
- Device ID Check (Step 5)
- Load Configuration Data (Step 6)
- CRC Check (Step 7)

Synchronization (Step 4)

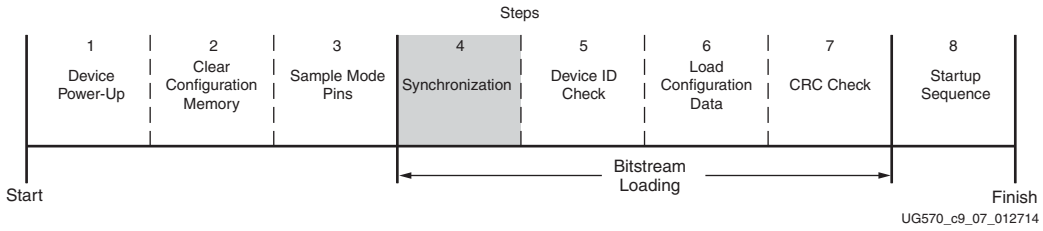


Figure 9-8: **Synchronization (Step 4)**

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 (Figure 9-8) 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 9-7 shows signals relating to synchronization.

Table 9-7: **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 FPGA status register.(1) The status register CFG_BUS_WIDTH_DETECTION bits indicate the detected bus width.	Indicates the detected bus width: 00 = x1 01 = x8 10 = x16 11 = x32 If ICAPE3 is enabled, this signal reflects the ICAPE3 width after configuration is done.

Notes:

1. Information on the FPGA status register is available in Table 9-25. Information on accessing the device status register through JTAG or SelectMAP is available in Chapter 10, Readback Verification and CRC.

Device ID Check (Step 5)

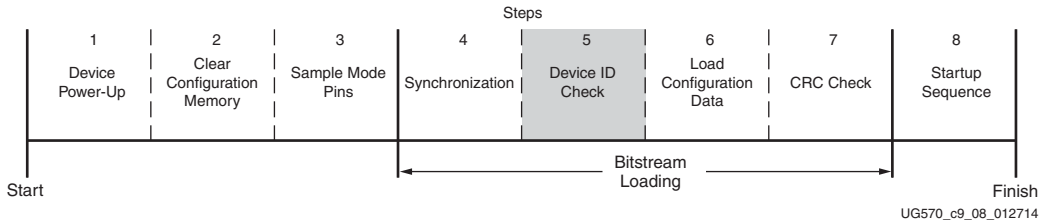


Figure 9-9: **Check Device ID (Step 5)**

After the device is synchronized, a device ID check must pass before the configuration data frames can be loaded (Figure 9-9). 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 FPGA JTAG IDCODE register has this format:

```
vvvv:dddddddddddddddd:ccccccccccc1
```

where:

v = version
d = 16-bit device code
c = company code

See Table 1-5, page 23 for IDCODE values.

Load Configuration Data (Step 6)

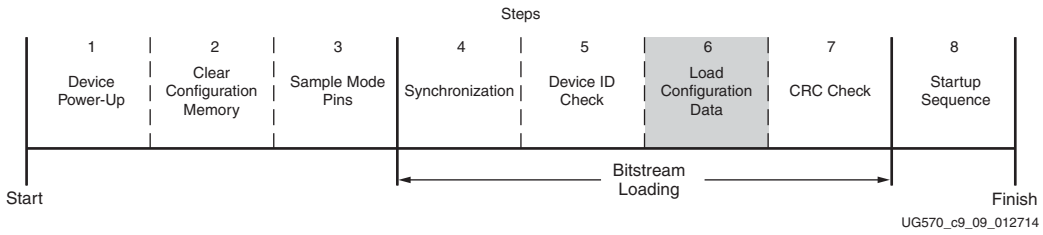


Figure 9-10: **Load Configuration Data Frames (Step 6)**

After the synchronization word is loaded and the device ID has been checked, the configuration data frames are loaded (Figure 9-10). This process is transparent to most users.

CRC Check (Step 7)

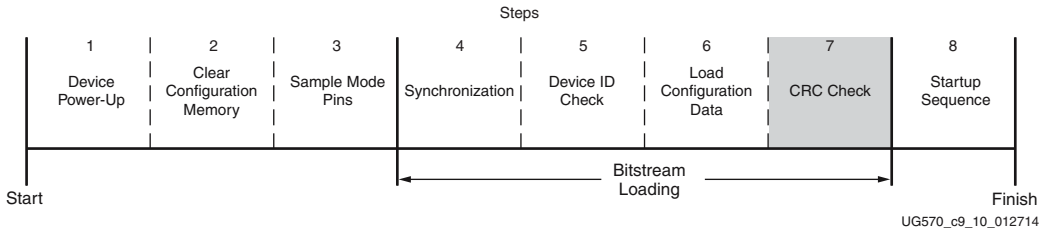


Figure 9-11: **Cyclic Redundancy Check (Step 7)**

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 you can disable it using BITSTREAM.GENERAL.CRC DISABLE , and you can disable the INIT_B error signal with BITSTREAM.CONFIG.INITSIGNALSERROR DISABLE. The CRC check and INIT_B error signal are recommended. 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.

For encrypted bitstreams (when the BITSTREAM.ENCRYPTION.ENCRYPT property is Yes), the CRC check is disabled and instead the AES-GCM authenticates the encrypted bitstream data. Errors in the bitstream data are reported in the status register as a security error.

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 active, 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 Chapter 5, SelectMAP Configuration Modes).

Xilinx 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, and the device does not complete configuration. 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.

Start-up Sequence (Step 8)

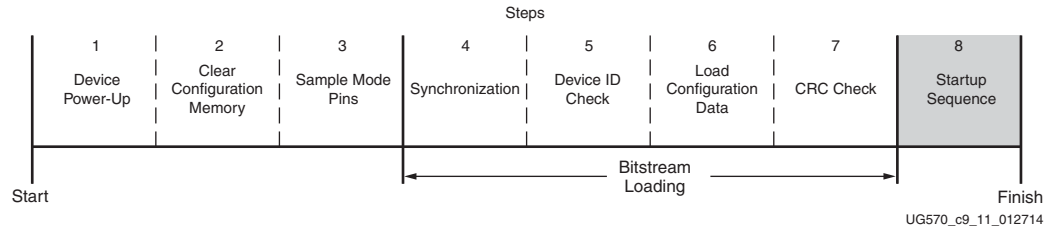


Figure 9-12: Start-up Sequence (Step 8)

After the configuration frames are loaded, the bitstream instructs the device to enter the start-up sequence (Figure 9-12). The start-up sequence is controlled by an 8-phase (phases 0-7) sequential state machine. The start-up sequencer performs the tasks outlined in Table 9-8.

Table 9-8: **User-Selectable Cycle of Start-up 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 start-up (EOS)

The specific order of start-up events (except for EOS assertion) is user-programmable through bitstream options controlled by the BITSTREAM.STARTUP properties (refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7]. Table 9-9 shows the general sequence of events, although the specific phase for each of these start-up events is user-programmable (EOS is always asserted in the last phase). By default, start-up events occur as shown in Table 9-9.

Table 9-9: **Default Sequence of Start-Up Events**

Phase	Event
4	Release DONE pin
5	Negate Global 3-State (GTS), activating I/O

Table 9-9: **Default Sequence of Start-Up Events *(Cont'd)***

Phase	Event
6	Assert GWE, allowing RAMs and flip-flops to change state
7	Assert End Of Start-up (EOS)

The start-up sequence can be forced to wait for the MMCMs to lock or for DCI to match with the appropriate bitstream 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.

Note: **Using DCI with the Multi-function Configuration Pins.** If any of the multi-function configuration pins in I/O bank 65 are assigned DCI I/O standards in the user design, the DCI calibration will not happen until after the pins are released from their configuration functions at the end of start-up. If the DCIUpdateMode is set to AsRequired, there will be an indeterministic delay after start-up until those pins are calibrated. If DCIUpdateMode is set to Quiet, the pins would never have their DCI values set. To avoid these issues, the DCIRESET primitive should be included, and 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 pins with DCI standards. For more details on DCI, see the *UltraScale Architecture SelectIO Resources User Guide* (UG571) [Ref 20].

The DONE signal is released by the start-up sequencer on the cycle indicated by the user options, but the start-up 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 a default internal pull-up resistor. There is no setup or hold requirement for the DONE register. Table 9-10 shows signals relating to the start-up sequencer. Figure 9-13 shows the waveforms relating to the start-up sequencer.

Table 9-10: **Signals Relating to Start-Up Sequencer**

Signal Name	Type	Access[1]	Description
DONE	Bidirectional[2]	DONE pin or FPGA status register	Indicates configuration is complete. Can be held Low externally to synchronize start-up with other FPGAs.
Release_DONE	Status	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 High, GWE enables the RAM, CLB flip-flops, and other synchronous elements on the FPGA.
GTS			Global 3-state (GTS). When asserted High, GTS disables all the I/O drivers except for the configuration pins.
EOS			End of start-up (EOS). EOS indicates the absolute end of the configuration and start-up 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_PLL_LOCKED			Asserted by default, or when MMCMs and PLLs have locked if the STARTUP_WAIT option is used.

Notes:

1. Information on the FPGA status register is available in Table 9-25. Information on accessing the device status register through JTAG or SelectMAP is available in Chapter 10, Readback Verification and CRC.
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 system clock, and timing violations can occur during start-up. It is recommended that you reset the design after start-up and/or apply some other synchronization technique.

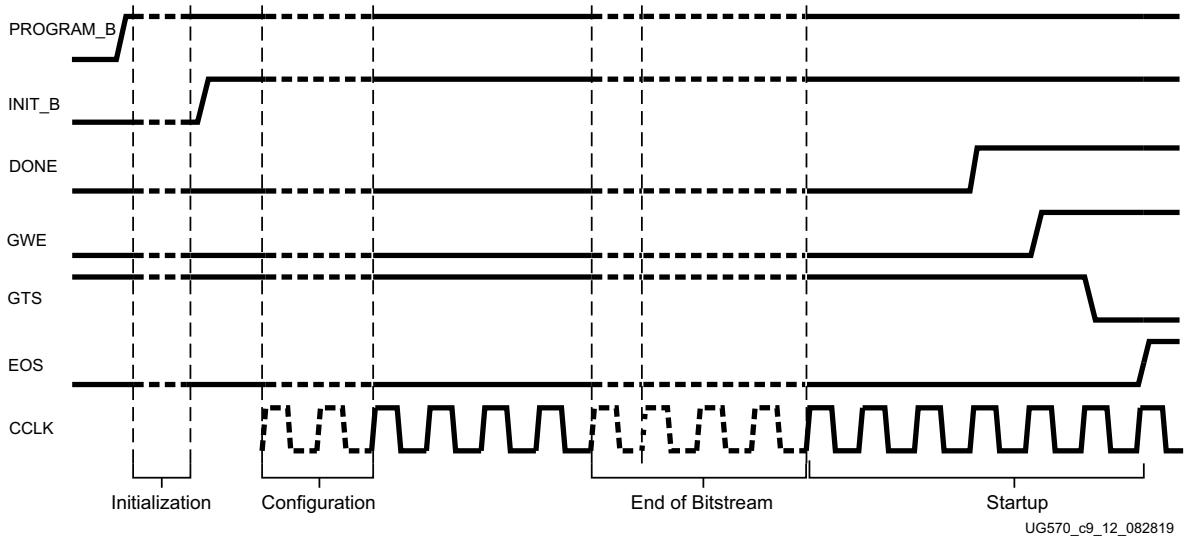


Figure 9-13: **Configuration Signal Sequencing (Default Start-Up Settings)**

Clocking to End of Start-up

By default, DONE is released in phase 4 of start-up. DONE indicates that configuration is complete and all data has been loaded, but some extra clock cycles need to be applied to ensure the start-up sequence completes correctly all the way to phase 7, End of Start-up. A conservative number for the clock cycles required after DONE is 64; this will account for the most common use cases. An undefined number of clock cycles are added If the bitstream options LCK_cycle and Match_cycle are set after phase 4 (DONE).

I/O Transition at the End of Startup

In all Kintex UltraScale FPGAs except for the KU095, 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. Note that this applies not only to the multi-function configuration bank 65, but also bank 70 in the KU085 and KU115. The transition occurs one CFGCLK after EOS (End Of Startup). To avoid this transition, set V_{CCO_65} (and V_{CCO_70}) to 2.5V or 3.3V, or drive the pin High externally (see Table 9-11). 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 STARTUPE3 primitive.

Table 9-11: **I/O Transition at End of Startup in Kintex UltraScale Family (Except KU095)**

V_{CCO_0}	V_{CCO_65} or V_{CCO_70}	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

Configuration Bitstream

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

A bitstream consists of three sections:

- Bus width auto detection
- Sync word
- FPGA configuration data

Bus Width Auto Detection

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 every bitstream (BIT or RBT). Because it appears before the Sync word, serial configuration modes ignore it (master serial, slave serial, JTAG, or SPI mode). 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 9-12 shows an example bitstream with an inserted bus width detection pattern. When observing the pattern on the FPGA data pins, the bits are bit swapped, as described in Parallel Bus Bit Order.

The bitstream data in Table 9-12 shows the 32-bit configuration word for an unswapped bitstream. For swapped and unswapped formats, see Configuration Data File Formats.

Table 9-12: **Bus Width Detection Pattern**

D[24:31]	D[16:23]	D[8:15]	D[0:7]	Comments
0xFF	0xFF	0xFF	0xFF	Ignored
0x00	0x00	0x00	0xBB	Bus Width Pattern
0x11	0x22	0x00	0x44	Bus Width Pattern
0xFF	0xFF	0xFF	0xFF	Ignored
0xFF	0xFF	0xFF	0xFF	Ignored

Table 9-12: **Bus Width Detection Pattern *(Cont'd)***

D[24:31]	D[16:23]	D[8:15]	D[0:7]	Comments
0xAA	0x99	0x55	0x66	Sync Word
. . .	. . .	. . .	. . .	. . .

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. See Table 9-13.

Table 9-13: **Bus Width Detection**

Bus Width	First Word		Second Word	
	Ignored Bits	D[0:7]	Ignored Bits	D[0:7]
x8	N/A	0xBB	N/A	0x11
x16	0x00	0xBB	0x11	0x22
x32	0x000000	0xBB	0x110022	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 9-14 shows the Sync word in an unswapped bitstream format.

Table 9-14: **Sync Word**

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

Configuration Memory Frames

FPGA configuration memory is arranged in frames that are tiled about the device. These frames are the smallest addressable segments of the FPGA configuration memory space, and all operations must therefore act upon whole configuration frames. All frames have a fixed, identical length. 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 raw_bitfile option when using the bitstream generator or by selecting *Create ASCII Configuration File* in the Generate Programming File options popup in the Vivado tools. 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. See Table 1-4, page 21.

Bitstream Composition

After synchronization, the configuration logic processes each 32-bit data word as a configuration packet or component of a multiple word configuration packet. Table Table 9-15 shows the composition of a sample KU040 bitstream, generated using default settings.

Table 9-15: **Sample KU040 Bitstream**

Configuration Data Word (hex)	Description
FFFFFFFF	Dummy pad word, word 1
FFFFFFFF	Dummy pad word, word 2
...	Dummy pad words 3-15
FFFFFFFF	Dummy pad word, word 16
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
20000000	NOOP
30022001	Packet Type 1: Write TIMER register, WORD_COUNT = 1
00000000	TIMER[31:0] = 00000000 (hex) (Placeholder for optional watchdog timer)
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
30002001	Packet Type 1: Write FAR register, WORD_COUNT=1

Table 9-15: **Sample KU040 Bitstream *(Cont'd)***

Configuration Data Word (hex)	Description
00000000	FAR (Frame address) = 00000000 (hex)
30026001	Packet Type 1: Write reserved/unused register, WORD_COUNT=1
00000000	Unused value = 00000000 (hex)
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. BPI page mode)
30018001	Packet Type 1: Write IDCODE register, WORD_COUNT = 1
03822093	IDCODE[31:0] = 03822093 (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
00000000	Bit mask for write to CTL0/CTL1 register. MASK[31:0] = 00000000 (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)
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 for future use)
20000000	NOOP
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)

Table 9-15: **Sample KU040 Bitstream *(Cont'd)***

Configuration Data Word (hex)	Description
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
503D0DA6	Packet Type 2: Write FDRI register, WORD_COUNT = 4,001,190
00000000	FDRI data word 1 (First bitstream configuration data word)
00000000	FDRI data word 2
...	FDRI data words 3 – 4,001,189
00000000	FDRI data word 4,001,190 (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)
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
00000100	Bit mask for write to CTL0/CTL1 register. MASK[31:0] = 00000100 (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

Table 9-15: **Sample KU040 Bitstream *(Cont'd)***

Configuration Data Word (hex)	Description
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) = DESYNC (Reset DALIGN) (Requires new Sync word)
20000000	NOOP (Pad remainder of bitstream with NOOPs)
...	...
20000000	NOOP (End of bitstream)

Configuration Packets

The configuration logic consists of a packet processor, a set of registers, and global signals that are controlled by the configuration registers. The packet processor controls the flow of data from the configuration interface to the appropriate register. The registers control all other aspects of configuration. All 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 use are described in this section.

Type 1 Packet

The Type 1 packet is used for register reads and writes. Only five out of 14 register address bits are used. 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. See Table 9-16 and Table 9-17.

Table 9-16: **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 9-17: **OPCODE Format**

OPCODE	Function
00	NOOP
01	Read
10	Write
11	Reserved

Type 2 Packet

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

Following the Type 2 packet header is the Type 2 Data section, which contains the number of 32-bit words specified by the word count portion of the header. See Table 9-18.

Table 9-18: **Type 2 Packet Header**

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

Configuration Registers

Table 9-19 summarizes the Type 1 packet registers. A detailed explanation of selected registers follows.

Table 9-19: **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.

Table 9-19: **Type 1 Packet Registers *(Cont'd)***

Name	Read/Write	Address	Description
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 start-up is allowed.

Frame Address Register (00001)

All frames have a fixed, identical length of 3,936 bits (123 32-bit words). See Differences Between UltraScale FPGA Families, page 11.

The frame address register (FAR) is divided into four fields: block type, row address, column address, and minor address (see Table 9-20 for UltraScale FPGAs and Table 9-21 for UltraScale+ FPGAs). 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 9-20: **UltraScale FPGA 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). A normal bitstream does not include types 010 or 011.
Row address	[22:17]	Selects the current row. The row addresses increment from bottom to top.
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.

Table 9-21: **UltraScale+ FPGA Frame Address Register Description**

Address Type	Bit Index	Description
Block type	[26:24]	Valid block types are CLB, I/O, CLK (000), block RAM content (001). A normal bitstream does not include types 010 or 011, or 100.
Row address	[23:18]	Selects the current row. The row addresses increment from bottom to top.
Column address	[17:8]	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	[7:0]	Selects a frame within a major column.

FDRI Register (00010)

Writing to this register configures 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 9-22 lists the command register commands and codes.

Table 9-22: **Command Register Codes**

Command	Code	Description
NULL	00000	Null command, no action.
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 start-up sequence: start-up sequence begins after a successful CRC check and a DESYNC command are performed.
RCRC	00111	Resets CRC: Resets the CRC register.

Table 9-22: **Command Register Codes *(Cont'd)***

Command	Code	Description
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 based on the ECLK_EN and OSCFSEL bits in the COR0 register.
GRESTORE	01010	Pulses the GRESTORE signal: sets/resets (depending on user configuration) 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).
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.
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.
BSPI_READ	10010	BPI/SPI re-initiate bitstream read.
FALL_EDGE	10011	Switch to negative edge clocking (configuration data capture on falling edge).

Control Register 0 (00101)

Control register 0 (CTL0) is used to configure the 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 9-23 and described in Table 9-24.

Table 9-23: **Control Register 0 (CTL0)**

Description	EFUSE_KEY	ICAP_SELECT	Reserved																	OverTempShutDown	Reserved	ConfigFallback	Reserved	GLUTMASK_B	Reserved	DEC	SBITS[1:0]		PERSIST	Reserved		GTS_USR_B
Bit Index	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	x	0	x	1	0	0	0	0	0	x	x	1

Table 9-24: **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	ICAPE3 port select: 0: Top ICAPE3 port enabled (default) 1: Bottom ICAPE3 port enabled
OverTempShutDown	12	Enables the System Monitor over-temperature shutdown: 0: Disables over-temperature shutdown (default) 1: Enables over-temperature shutdown
ConfigFallback	10	Stops when configuration fails and disables fallback to the default bitstream. The bitstream generator option is ConfigFallback: Enable/Disable. 0: Enables fallback (default) 1: Disables fallback
GLUTMASK_B	8	Global LUT mask signal. Masks any changeable memory cell readback value. 0: Masks changeable memory cell readback value, such as distributed RAM or SRL 1: Does not mask changeable memory cell readback values (default)
DEC	6	AES decryptor enable bit: 0: Decryptor disabled (default) 1: Decryptor enabled
SBITS[1:0]	[5:4]	Security level. The FPGA security level is extended to encrypted bitstreams. It is applicable to the configuration port, not to ICAPE3. 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 Chapter 5, SelectMAP Configuration Modes. 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. A 1 in the MASK register allows the corresponding bit in the CTL0 or CTL1 register to be written. The default is all 0s. The MASK register must be written before each write to CTL0 or CTL1.

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 9-25 gives the name of each bit position in the STAT register; a detailed explanation of each bit position is given in Table 9-26.

Table 9-25: **Status Register**

Description	Reserved					CFG_BUS_WIDTH_DETECTION		Reserved				CFG_STARTUP_STATE_MACHINE_PHASE			SYSTEM_MONITOR_OVER_TEMP	SECURITY_ERROR	IDCODE_ERROR	DONE_PIN	DONE_INTERNAL_SIGNAL_STATUS	INIT_B_PIN	INIT_B_INTERNAL_SIGNAL_STATUS	MODE_PIN_M[2:0]			GHIGH_B_STATUS	GWE_STATUS	GTS_CFG_B_STATUS	END_OF_STARTUP_(EOS)_STATUS	DCI_MATCH_STATUS	MMCM_PLL_LOCKED	DECRYPTOR_ENABLED	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 9-26: **Status Register Description**

Name	Bit Index	Description
CFG_BUS_WIDTH_DETECTION	[26:25]	Bus width auto detection result. This is the width of the internal configuration bus (not necessarily the same as the external configuration data width). For BPI and SelectMAP modes, value is set to 01 (x8) after mode pins are sampled and before bus width detection. If ICAPE3 is enabled, this field reflects the ICAPE3 bus width after configuration is done. RSA authentication sets the bus width to x32. 00 = x1 01 = x8 10 = x16 11 = x32
CFG_STARTUP_STATE_MACHINE_PHASE	[20:18]	Start-up 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
SYSTEM_MONITOR_OVER_TEMP	17	System Monitor over-temperature.
SECURITY_ERROR	16	Security violation: 0: No SECURITY_ERROR 1: SECURITY_ERROR
IDCODE_ERROR	15	Attempt to write to FDRI without successful DEVICE_ID check: 0: No IDCODE_ERROR 1: IDCODE_ERROR
DONE_PIN	14	Value on DONE_PIN pin.
DONE_INTERNAL_SIGNAL_STATUS	13	Value of internal DONE_INTERNAL_SIGNAL_STATUS signal: 0: Signal not released (pin is actively held Low) 1: Signal released (can be held Low externally)
INIT_B_PIN	12	Value on INIT_B_PIN pin
INIT_B_INTERNAL_SIGNAL_STATUS	11	Internal signal indicating initialization has completed: 0: Initialization has not finished 1: Initialization finished
MODE_PIN_M[2:0]	[10:8]	Status of the mode pins (M[2:0]).
GHIGH_B_STATUS	7	Status of GHIGH_B_STATUS: 0: GHIGH_B_STATUS asserted 1: GHIGH_B_STATUS deasserted
GWE_STATUS	6	Status of GWE: 0: Flip-flops and block RAM are write disabled 1: Flip-flops and block RAM are write enabled

Table 9-26: **Status Register Description *(Cont'd)***

Name	Bit Index	Description
GTS_CFG_B_STATUS	5	Status of GTS_CFG_B: 0: All I/Os are placed in High-Z state 1: All I/Os behave as configured
END_OF_STARTUP_(EOS)_STATUS	4	End of start-up signal from start-up block: 0: Start-up sequence has not finished 1: Start-up sequence has finished
DCI_MATCH_STATUS	3	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 MATCH signal = 1. 0: DCI not matched 1: DCI is matched
MMCM_PLL_LOCKED	2	This bit is a logical AND function of all MMCM/PLL LOCKED signals. Unused MMCM/PLL LOCKED signals = 1. 0: MMCMs/PLLs are not locked 1: MMCMs/PLLs are locked
DECRYPTOR_ENABLED	1	0: Decryptor security not set 1: Decryptor security set
CRC_ERROR	0	0: No CRC error 1: CRC error

LOUT Register (01000)

This register drives data to the DOUT pin during serial daisy-chain configuration.

Configuration Options Register 0 (01001)

The 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 9-27 and described in Table 9-28.

Table 9-27: **Configuration Options Register 0**

Description	Reserved					ECLK_EN	Reserved	DRIVE_DONE	Reserved	OSCFSEL						Reserved		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	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	0	0

Table 9-28: **Configuration Options Register 0 Description**

Name	Bit Index	Description
ECLK_EN	26	External master configuration clock (EMCCLK) enable. Bitstream property: BITSTREAM.CONFIG.EXTMASTERCCLK_EN.
DRIVE_DONE	24	0: DONE pin is open drain 1: DONE is actively driven High (not recommended)
OSCFSEL	[22:17]	Select CCLK frequency in master modes (3 MHz – 148 MHz for UltraScale FPGAs, and 2.7 MHz - 170 MHz for UltraScale+ FPGAs). Bitstream property: BITSTREAM.CONFIG.CONFIGRATE.
DONE_CYCLE	[14:12]	Start-up cycle to release the DONE pin: 000: Start-up phase 1 001: Start-up phase 2 010: Start-up phase 3 011: Start-up phase 4 100: Start-up phase 5 101: Start-up phase 6 Bitstream property: BITSTREAM.STARTUP.DONE_CYCLE.
MATCH_CYCLE	[11:9]	Start-up cycle to stall in until DCI matches: 000: Start-up phase 0 001: Start-up phase 1 010: Start-up phase 2 011: Start-up phase 3 100: Start-up phase 4 101: Start-up phase 5 110: Start-up phase 6 111: No wait Bitstream property: BITSTREAM.STARTUP.MATCH_CYCLE.

Table 9-28: **Configuration Options Register 0 Description *(Cont'd)***

Name	Bit Index	Description
LOCK_CYCLE	[8:6]	Start-up cycle to stall in until MMCMs lock: 000: Start-up phase 0 001: Start-up phase 1 010: Start-up phase 2 011: Start-up phase 3 100: Start-up phase 4 101: Start-up phase 5 110: Start-up phase 6 111: No wait Bitstream property: BITSTREAM.STARTUP.LCK_CYCLE.
GTS_CYCLE	[5:3]	Start-up cycle to deassert the Global 3-State (GTS) signal: 000: Start-up phase 1 001: Start-up phase 2 010: Start-up phase 3 011: Start-up phase 4 100: Start-up phase 5 101: Start-up phase 6 110: GTS tracks DONE pin. Bitstream property: GTS_CYCLE:DONE.
GWE_CYCLE	[2:0]	Start-up phase to deassert the Global Write Enable (GWE) signal: 000: Start-up phase 1 001: Start-up phase 2 010: Start-up phase 3 011: Start-up phase 4 100: Start-up phase 5 101: Start-up phase 6 110: GWE tracks DONE pin. Bitstream property: GWE_CYCLE:DONE

MFWR Register (01010)

This register is used by the bitstream compression option.

AES_IV 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 IDCODE. A read of this register returns the device IDCODE.

AXSS Register (01101)

This register supports the USR_ACCESSE2 primitive (see USR_ACCESSE2, page 123).

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 9-29 and described in Table 9-30.

Table 9-29: **Configuration Options Register 1**

Description	Reserved														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 9-30: **Configuration Options Register 1 Description**

Name	Bit Index	Description
Reserved	[31:18]	Reserved
RBCRC_ACTION	[17:15]	Controls readback action: 000: Continue Others: Reserved (use SEM IP for correction applications)
Reserved	[14:10]	Reserved
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
Reserved	[7:4]	Reserved

Table 9-30: **Configuration Options Register 1 Description *(Cont'd)***

Name	Bit Index	Description
BPI_1ST_READ_CYCLE	[3:2]	First byte read timing: 00: 1 CCLK (default) 01: 2 CCLKs 10: 3 CCLKs 11: 4 CCLKs Bitstream property: BITSTREAM.CONFIG.BPI_1ST_READ_CYCLE
BPI_PAGE_SIZE	[1:0]	Flash memory page size: 00: 1 byte/word (default) 01: 4 bytes/words 10: 8 bytes/words 11: Reserved Bitstream property: BITSTREAM.CONFIG.BPI_PAGE_SIZE

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 (WBSTAR) is given in Table 9-31 and described in Table 9-32.

Table 9-31: **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 9-32: **WBSTAR Register Description**

Name	Bit Index	Description
RS[1:0]	[31:30]	RS[1:0] pin value on next warm boot in BPI mode. 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 9-33 and described in Table 9-34.

Table 9-33: **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 9-34: **TIMER Register Description**

Name	Bit Index	Description
TIMER_USR_MON	31	Watchdog is enabled during user mode: 0: Disabled (default) 1: Enabled The watchdog clock in user mode is the CFGMCLK divided by 256. At a nominal CFGMCLK of 50 MHz and a tolerance of 15%, the watchdog clock would be 166–225 kHz.
TIMER_CFG_MON	30	Watchdog is enabled during configuration: 0: Disabled (default) 1: Enabled The configuration watchdog clock is CCLK, as it is defined for configuration. When using EMCCLK, the Watchdog Timer runs off of the CCLK initially and then switches over to EMCCLK after the bitstream image header is read.
TIMER_VALUE	[29:0]	Watchdog time-out value. 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 9-35 and described in Table 9-36.

Table 9-35: **BOOTSTS Register**

Description	Reserved																	WRAP_ERROR_1	CRC_ERROR_1	ID_ERROR_1	WATCHDOG_TIMEOUT_ERROR_1	INTERNAL_PROG_1	FALLBACK_1	STATUS_VALID_1	Reserved	WRAP_ERROR_0	CRC_ERROR_0	ID_ERROR_0	WATCHDOG_TIMEOUT_ERROR_0	INTERNAL_PROG_0	FALLBACK_0	STATUS_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 9-36: **BOOTSTS Register Description**

Name	Bit Index	Description
WRAP_ERROR_1	14	BPI address counter wraparound error, supported in asynchronous read mode.
CRC_ERROR_1	13	CRC error.
ID_ERROR_1	12	IDCODE error.
WATCHDOG_TIMEOUT_ERROR_1	11	Watchdog time-out error.
INTERNAL_PROG_1	10	Internal PROG triggered configuration.
FALLBACK_1	9	1: Fall back to default reconfiguration, RS[1:0] actively drives 2'b00 0: Normal configuration
STATUS_VALID_1	8	Status 1 is valid.
WRAP_ERROR_0	6	BPI address counter wraparound error, supported in asynchronous read mode
CRC_ERROR_0	5	CRC error.
ID_ERROR_0	4	IDCODE error.
WATCHDOG_TIMEOUT_ERROR_0	3	Watchdog time-out error.
INTERNAL_PROG_0	2	Internal PROG triggered configuration.
FALLBACK_0	1	1: Fall back to default reconfiguration, RS[1:0] actively drives 2'b00 0: Normal configuration

Table 9-36: **BOOTSTS Register Description *(Cont'd)***

Name	Bit Index	Description
STATUS_VALID_0	0	Status 0 is valid.

Notes:

1. The default power-up state for all fields in the BOOTSTS 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 device. This register is reserved except for the CAPTURE bit. Writes to the CTL1 register are masked by the value in the MASK register. The name of each bit position in the CTL1 register is given in Table 9-37 and described in Table 9-38.

Table 9-37: **Control Register 1 (CTL1)**

Description	Reserved								CAPTURE	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	0	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 9-38: **Control Register 1 Description**

Name	Bit Index	Description
CAPTURE	23	Controls readback capture enable: 0: Disables readback capture (default) 1: Enables readback capture

BPI/SPI Configuration Options Register (11111)

The BPI/SPI configuration options register (BSPI) is used to store certain configuration options for the device set by the tools. The name of each bit position in the BSPI register is given in Table 9-39 and described in Table 9-40.

Table 9-39: **BPI/SPI Configuration Options Register (BSPI)**

Description	Reserved				BPI_SYNC_MODE	BPI_SYNC_RCR															Reserved	SPI_32BIT_ADDR	SPI_BUSWIDTH		SPI_READ_OPCODE							
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

Table 9-40: **BPI/SPI Configuration Options Register Description**

Name	Bit Index	Description
BPI_SYNC_MODE	27	BPI configuration flash read mode: 0: Asynchronous Read (default) (See Master BPI Asynchronous Read, page 68) 1: Synchronous Read (See Master BPI Synchronous Read, page 64)
BPI_SYNC_RCR	[26:12]	BPI configuration flash Read Configuration register. Determined by property BITSTREAM.CONFIG.BPI_SYNC_MODE options Type1 or Type2. See XAPP1220 [Ref 13] for more information.
SPI_32BIT_ADDR	10	SPI address width: 0: 24-bit address (default) 1: 32-bit address (See Serial NOR Flash Densities over 128 Mb, page 55)
SPI_BUSWIDTH	[9:8]	SPI bus width: 00: x1 (default) 01: x2 10: x4 11: x8 (dual quad)
SPI_READ_OPCODE	[7:0]	SPI flash read instruction. See Table 2-1, page 50.

Readback Verification and CRC

Introduction

The UltraScale™ architecture-based FPGAs allow you to read configuration memory through the SelectMAP, ICAP, and JTAG interfaces. During readback, all configuration memory cells are read by default, including the current values on all user memory elements (LUT RAM, SRL, and block RAM).

To read configuration memory, you must send a sequence of commands to the device to initiate the readback procedure. You can send the readback command sequence from a microprocessor, CPLD, or FPGA-based system, or use the configuration tools to perform JTAG-based readback verify. 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. Xilinx device programming tools can automatically perform all readback and comparison functions and report whether there were any configuration errors.

There are two mandatory bitstream settings for readback through the SelectMAP or JTAG interfaces: the bitstream security setting must not prohibit readback, and bitstream encryption must not be used. Additionally, if readback is to be performed through the SelectMAP interface, the port must be set to retain its function after configuration by setting the persist option in the bitstream generator, 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.

Readback capture provides the ability to read the current user state of internal CLB registers, block RAM, distributed RAM, and SRL contents to check for proper design functionality. The feature provides easy access for observing the design state with little pre-planning and no added design logic resources. The readback capture flow is demonstrated in *Configuration Readback Capture in UltraScale FPGAs* (XAPP1230) [Ref 21].

Persist Option

The persist bitstream option (BITSTREAM.CONFIG.PERSIST YES) 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 as the JTAG port is dedicated and always available. 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. Persist and ICAP cannot be used at the same time. Persist is also not recommended for standard Master SPI/BPI configuration mode setups.

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 1-7, page 29 and Table 1-8, page 30, except that PUDC_B never persists. 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 standard default of LVCMOS, 12 mA drive, fast slew rate.

Readback Command Sequences

The following sections provide instructions for performing readback through the SelectMAP or JTAG interfaces. Readback through ICAP is similar to readback through the SelectMAP interface.

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.

See Table 10-1 for command sequence.

Table 10-1: **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 Type 1 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
12	Write	0000000D	DESYNC command
13	Write	20000000	NOOP
14	Write	20000000	NOOP

You must change the SelectMAP interface from write to read control between steps 9 and 10, and back to write control after step 10.

To read registers other than STAT, the address specified in the Type 1 packet header in step 7 of Table 10-1 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.

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:

   Kintex UltraScale and Virtex UltraScale:
   FDRO read length = (words per frame) x (frames to read +1) + 10

   Kintex UltraScale+ and Virtex UltraScale+:
   FDRO read length = (words per frame) x (frames to read + 1) + 25

   One extra frame is read to account for the frame buffer. The frame buffer produces one dummy frame at the beginning of the read. 10 or 25 extra words are read to account for pipelining.

9. Write 64 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 8. Readback data is valid deterministically three clock cycles after the CSI_B pin is asserted during readback.
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 10-2 shows the readback command sequence.

Table 10-2: **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

Table 10-2: **Shutdown Readback Command Sequence (SelectMAP) *(Cont'd)***

Step	SelectMAP Port Direction	Configuration Data	Explanation
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
		483D0E2B	Type 2 read 4,001,323 words from FDRO (for KU040)
9	Write	20000000	Type 1 NOOP word 0
		. . .	Type 1 63 more NOOPs word 0
10	Read	00000000	Packet data read FDRO word 0
		. . .	
		00000000	Packet data read FDRO word 4,001,322
11	Write	20000000	Type 1 NOOP word 0
12	Write	30008001	Type 1 write 1 word to CMD
		00000005	START command
		20000000	Type 1 NOOP word 0
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 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 10-3.

Table 10-3: **Shifting in the JTAG CFG_IN and CFG_OUT Instructions**

Step	Description	Set and Hold		Number of Clocks (TCK)
		TDI	TMS	
1	Clock five 1s 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
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 every configuration packet 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 10-4 describes the readback command sequence.

Table 10-4: **Status Register Readback Command Sequence (JTAG)**

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

Table 10-4: **Status Register Readback Command Sequence (JTAG) *(Cont'd)***

Step	Description	Set and Hold		Number of Clocks (TCK)
		TDI	TMS	
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
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 into 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 (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 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 10-5 shows the shutdown readback command sequence.

Table 10-5: **Shutdown Readback Command Sequence (JTAG)**

Step	Description	Set and Hold		Number 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	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
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 10-5: **Shutdown Readback Command Sequence (JTAG) *(Cont'd)***

Step	Description	Set and Hold		Number 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 10 words of pipeline data and 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 10-1.

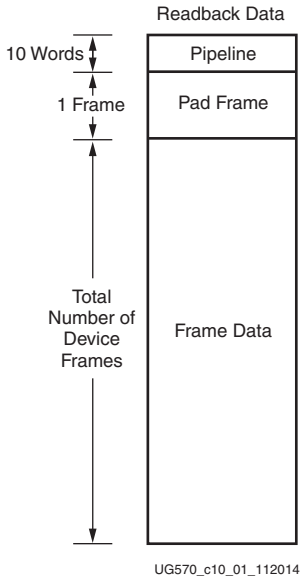


Figure 10-1: **Readback Data Stream**

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

Compare to Golden Readback File

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. See Figure 10-2.

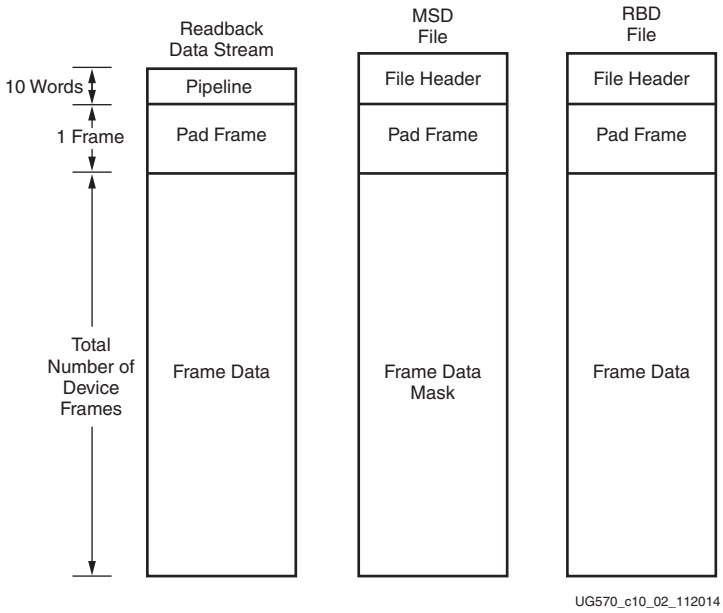


Figure 10-2: **Comparing Readback Data Using the MSD and RBD Files**

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.

Compare to Frame Data in Bitstream

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 (see Figure 10-3).

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.

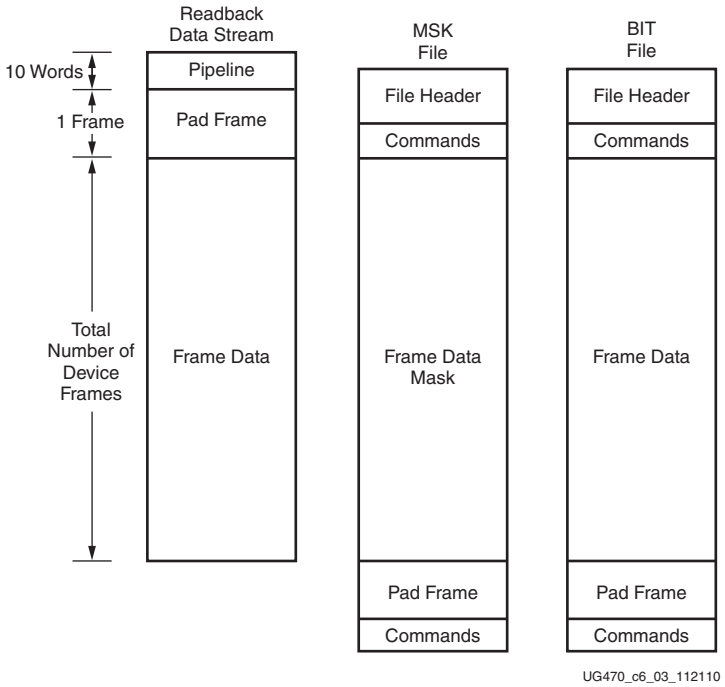


Figure 10-3: **Comparing Readback Data Using the MSK and BIT Files**

SEU Detection and Correction

UltraScale architecture-based FPGAs include the ability to do continuous readback of configuration data in the background of a user design. A cyclic redundancy code (CRC) check value from the readback data can be compared to a golden value to validate the integrity of the bitstream. This feature is aimed at simplifying detection of single event upsets (SEU) that cause a configuration memory bit to flip, and is used by the Soft Error Mitigation (SEM) IP.

RECOMMENDED: *Use the Xilinx Soft Error Mitigation (SEM) IP, which automates the implementation of SEU detection and correction. Note that readback CRC is not supported outside of the SEM IP, and that the SEM IP is not supported in the KU025 device.*

After readback CRC is enabled, the dedicated configuration logic reads back continuously in the background to check the CRC of the configuration memory content. As some configuration bits can change during operation, such as distributed RAM, the readback CRC function masks these bits so that variable locations are ignored. These dynamically changeable memory locations are masked during background readback:

- SLICEM LUT (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. The DESYNC command clears the CRC_ERROR flag.
- The JTAG interface is not controlling the internal configuration bus via the JTAG CFG_IN instruction, CFG_OUT instruction, or ISC_ENABLE function.

The SEM IP uses the ICAPE3 and runs the readback CRC on the ICAPE3 CLK source.

In UltraScale devices, any use of configuration readback (Readback CRC, SEM-IP, internal or external SEU scrubbing, and other configuration readback activities) or partial reconfiguration exercises data lines that have a small amount of coupling into the VCO of the MMCM and PLLs. As a result, varying amounts of increased time interval error jitter can be observed. See AR#71314 for guidance and mitigation techniques. See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7] for write_bitstream properties information.

In a partial reconfiguration application, the configuration memory content changes, so the golden signature must be recalculated. The golden CRC must be re-calculated by the SEM IP after any partial reconfiguration event. See *Demonstration of Soft Error Mitigation IP and Partial Reconfiguration Capability on Monolithic Devices* (XAPP1261) [Ref 22].

The readback CRC will be interrupted by a reconfiguration command. Except when reconfiguring through JTAG, the readback CRC logic might need up to 130 clock cycles to complete before new configuration packets can be applied. Therefore, bitstreams that might be used to reconfigure a device running readback CRC should pad 130 NOOPs after the SYNC word.

For more information, see

www.xilinx.com/products/intellectual-property/SEM.htm

Readback Capture

Readback capture uses the same readback process but requires additional commands to be issued during the readback sequence to read the user state of the internal CLB registers. The UltraScale FPGAs do not have dedicated capture memory cells, so the CAPTUREE2 primitive and GCAPTURE commands available in 7 series FPGAs are no longer required or supported in UltraScale FPGAs. In the UltraScale FPGAs, a write to the mask (MSK) register and control 1 (CTL1) register (CAPTURE bit[23]) are required to enable the readback capture of the CLB

registers. In the UltraScale FPGAs, you must stop or disable the clock associated with the user state elements being targeted throughout the duration of the readback capture sequence. The write_bitstream option `-logic_location` creates an ASCII format `.ll` file for bit mapping the locations of the block RAM, distributed RAM, SRLs, and CLB registers within the readback data.

For more information on readback capture, see *Configuration Readback Capture in UltraScale FPGAs* (XAPP1230) [Ref 21].

MultiBoot and Reconfiguration

Introduction

This chapter focuses on full bitstream reconfiguration methods available in UltraScale™ architecture-based FPGAs.

Fallback MultiBoot

The FPGA MultiBoot and fallback features support updating bitstream images 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. The MultiBoot and fallback feature can be used with all master configuration modes.

When fallback occurs, an internally generated pulse resets the entire configuration logic, except for the dedicated MultiBoot logic, the WBSTAR (warm boot start address), the BSPI, and the BOOTSTS (boot status) 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 (in BPI mode). 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 timeout error
- A BPI address wraparound error

Fallback can also be enabled with the bitstream option ConfigFallback (BITSTREAM.CONFIG.CONFIGFALLBACK ENABLE). 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.

Golden Image Initial System Setup

The golden image is loaded starting from address location 0 at FPGA 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. The golden image loaded at power-up triggers booting from an upper address space. Multiple MultiBoot images can 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 11-1 shows the flow for the initial setup of the golden image.

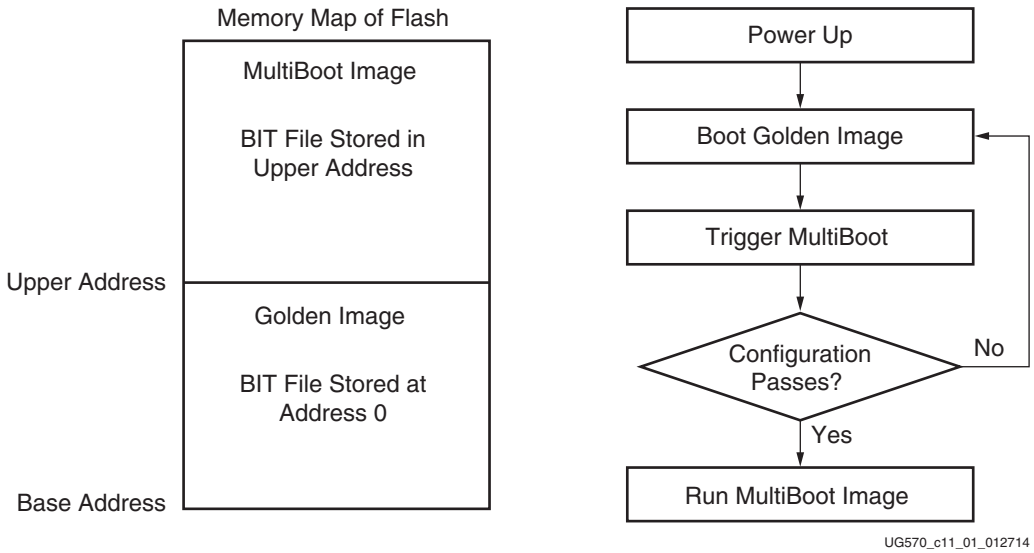


Figure 11-1: **Initial Golden Image Flow Diagram**

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 by the bitstream setting for the next configuration address (BITSTREAM.CONFIG.NEXT_CONFIG_ADDR), or is issued by code through the ICAP primitive ICAPE3 instantiated within the golden image design. The IPROG command can be turned off in a bitstream with BITSTREAM.CONFIG.NEXT_CONFIG_REBOOT DISABLE, so that the golden image is loaded on power up.
- The WBSTAR (warm boot start address) register is set to jump to an address in the bitstream settings or through ICAP.
- The MultiBoot image must be stored in flash at the address in the WBSTAR register.
- The Watchdog Timer is enabled in the bitstream options to recover from incompletely programmed flash 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 11-2 shows the flow for the initial setup of the MultiBoot image.

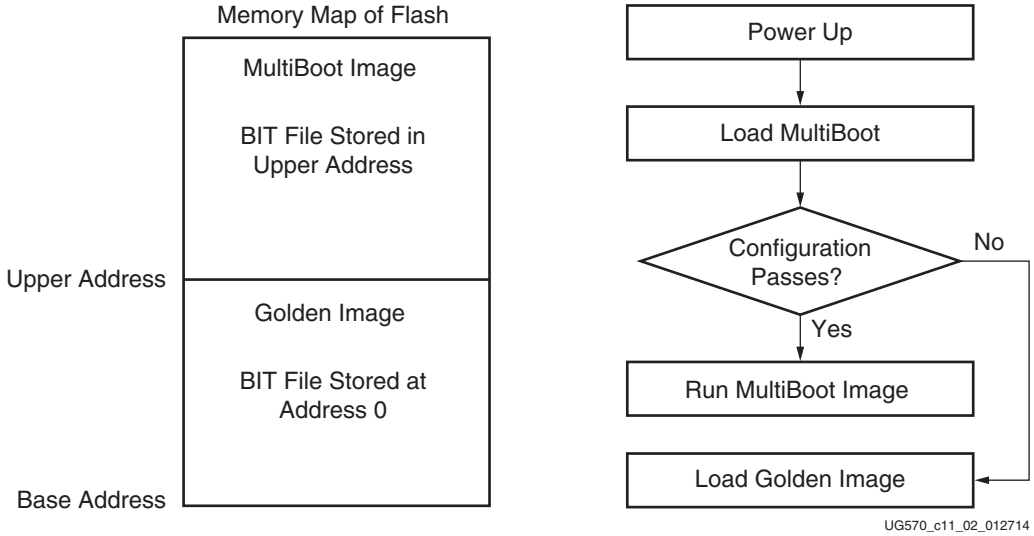


Figure 11-2: **Initial MultiBoot Image Flow Diagram**

Initial MultiBoot Design Considerations

Design considerations for the golden image are:

- The WBSTAR setting in the bitstream options points to the MultiBoot location.
- An IPROG command is inserted through the bitstream options to trigger loading of MultiBoot at power-up.
- The Watchdog Timer is enabled in the bitstream options.
- ICAPE3 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.

Design considerations for the MultiBoot image are:

- The WBSTAR setting in the bitstream options points to the MultiBoot location.
- The Watchdog Timer is enabled in the bitstream options.
- ICAPE3 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 at least one of the RS pins tied to an upper address line. The other RS pin could be pulled up or down depending on the bitstream size, flash size, and where you would like the

power-up bistream to be located in the flash. 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 for further details.

MultiBoot Bitstream Spacing

The bitstream keeps loading until the end of startup (EOS). When MMCM wait or DCI match is enabled before the DONE startup cycle, padding (all 0s or all 1s) is required between bitstreams to compensate for the total wait time. Otherwise, the following bitstream can potentially overwrite the previous bitstream. The formula for the padding size is cfg_bus_width x total_wait_time / CCLK_period.

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 pins 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, BSPI, and BOOTSTS 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 through the ICAP, which is controlled by user logic. This allows user logic to initiate device reconfiguration. In the second way, the IPROG command can be embedded 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. Now, 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 in BPI mode. This register can be loaded from bitstream options or from the ICAP. If the register is not set in the bitstream options, 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 resets 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 the more common configuration monitor mode, when the Watchdog Timer times out, the configuration logic loads the fallback bitstream. 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. The Fallback MultiBoot section provides more details.

In configuration monitor mode, the TIMER register is set in the BIT file by the bitstream generator. 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 MMCM wait or DCI match settings; these delays need to be taken into account. The TIMER register runs at the CCLK frequency.

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

In user logic monitor mode, the Watchdog Timer uses a dedicated internal clock, CFGMCLK, which has a nominal frequency of 50 MHz. The clock is pre-divided by 256, so that the Watchdog Timer clock period is about 5,120 ns. Given the watchdog counter is 30 bits wide, the maximum possible watchdog value is about 5,500 seconds. The time value can be set using the bitstream options.

FPGA End of Start-Up

To use the Watchdog Timer to monitor the bitstream configuration, set the bitstream property BITSTREAM.CONFIG.TIMER_CFG, or 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 start-up is complete. Any wait time in start-up 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 start-up, a watchdog timeout error occurs and triggers a fallback configuration.

User Monitor Mode

To use the Watchdog Timer to monitor user logic, set the bitstream property BITSTREAM.CONFIG.TIMER_USR, or set TIMER_USR_MON to 1 and the desired TIMER_VALUE in a write to the TIMER register in the bitstream. The user design 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 11-1 shows an example bitstream for reloading the Watchdog Timer using the LTIMER command.

Table 11-1: **Example Bitstream for Reloading the Watchdog Timer with LTIMER**

Configuration Data (hex)	Explanation
FFFFFFFF	Dummy word
AA995566	Sync word
20000000	Type 1 NOOP
30008001	Type 1 Write 1 words to CMD
00000000	NULL
20000000	Type 1 NOOP
30008001	Type 1 Write 1 words to CMD
00000011	LTIMER command
20000000	Type 1 NOOP
30008001	Type 1 Write 1 words to CMD
0000000D	DESYNC
20000000	Type 1 NOOP

Table 11-2 shows an example bitstream for directly accessing the TIMER register.

Table 11-2: **Example Bitstream for Accessing the TIMER Register**

Configuration Data (hex)	Explanation
FFFFFFFF	Dummy word
AA995566	Sync word
20000000	Type 1 NOOP
30022001	Type 1 Write 1 words to TIMER
xxxxxxxx	TIMER value

Table 11-2: **Example Bitstream for Accessing the TIMER Register**

Configuration Data (hex)	Explanation
20000000	Type 1 NOOP
30008001	Type 1 Write 1 words to CMD
0000000D	DESYNC
20000000	Type 1 NOOP

RS Pins

The dual-purpose RS pins are disabled by default. The RS pins drive Low during a fallback for Master BPI configuration 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.

Tying the RS pins to the flash upper address pins allows for easy selection between up to four images. When using this feature with the basic BPI asynchronous read, the user should be aware that the bitstream size must be equal or less than one fourth the size of the flash as the RS pins are held at a static value allowing access to one fourth of the flash with one selection.

IPROG Reconfiguration

The internal PROGRAM_B (INTERNAL_PROG or IPROG) command has similar effect as pulsing the 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) in Chapter 9) is used during SPI or BPI reconfiguration instead of the default address. The default is zero. The IPROG command can be sent through ICAP or the bitstream. The IPROG Using ICAP and IPROG Embedded in the Bitstream sections describe these two procedures. Note that the ICAP interface is similar to the SelectMAP interface, and therefore the input configuration bus needs to be bit-swapped (see Parallel Bus Bit Order, page 144). For more information on ICAPE3, see the *UltraScale Architecture Libraries Guide* (UG974) [Ref 16].

IPROG Using ICAP

The IPROG command can also be sent through ICAP using the ICAPE3 primitive. After a successful configuration, the user design determines the start address of the next bitstream, sets the WBSTAR register, and then issues an IPROG command using ICAP.

The command sequence is:

1. Send the Sync word.
2. Program the WBSTAR register for the next bitstream start address (see Warm Boot Start Address Register (10000) in Chapter 9).
3. Send the IPROG command.

Table 11-3 shows an example bitstream for the IPROG command using ICAP.

Table 11-3: **Example Bitstream for IPROG through ICAP**

Configuration Data (hex)[(1)]	Explanation
FFFFFFFF	Dummy word
AA995566	Sync word
20000000	Type 1 NOOP
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 NOOP

Notes:
1. See Parallel Bus Bit Order, page 144.

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. See Table 11-4.

Table 11-4: **WBSTAR Controlled Pins According to Configuration Mode**

Configuration Mode	Pins Controlled by WBSTAR
Master SPI	START_ADDR is sent to the flash device serially.
Master BPI	RS[1:0], A[28:00]

RS[1:0] is controllable by WBSTAR in BPI mode only. The START_ADDR field is only meaningful for the BPI and SPI modes.

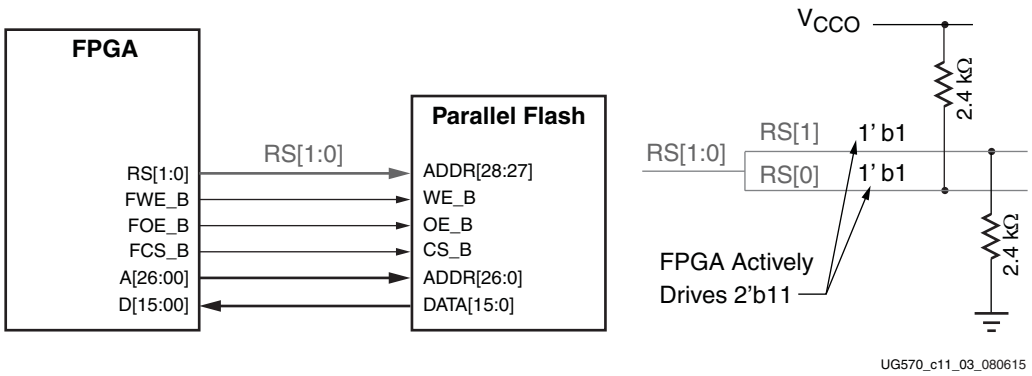


Figure 11-3: **IPROG in BPI Modes**

Notes relevant to Figure 11-3:

1. All BPI pins, except the CCLK, FCS_B, and D[03:00] pins, are multi-function I/Os. After configuration is finished (the DONE pins goes High), these pins become user I/Os and can be controlled by user logic to access 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. You can specify any RS[1:0] value in the WBSTAR register using BITSTREAM.CONFIG.REVISIONSELECT.

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 FPGA bitstream. For example, WBSTAR can be modified to a user-desired start address (see Warm Boot Start Address Register (10000) in Chapter 9). A NULL command after WBSTAR can be modified to IPROG by setting the four LSB bits to all ones (see Command Register (00100) in Chapter 9).

Figure 11-4 illustrates this use model.

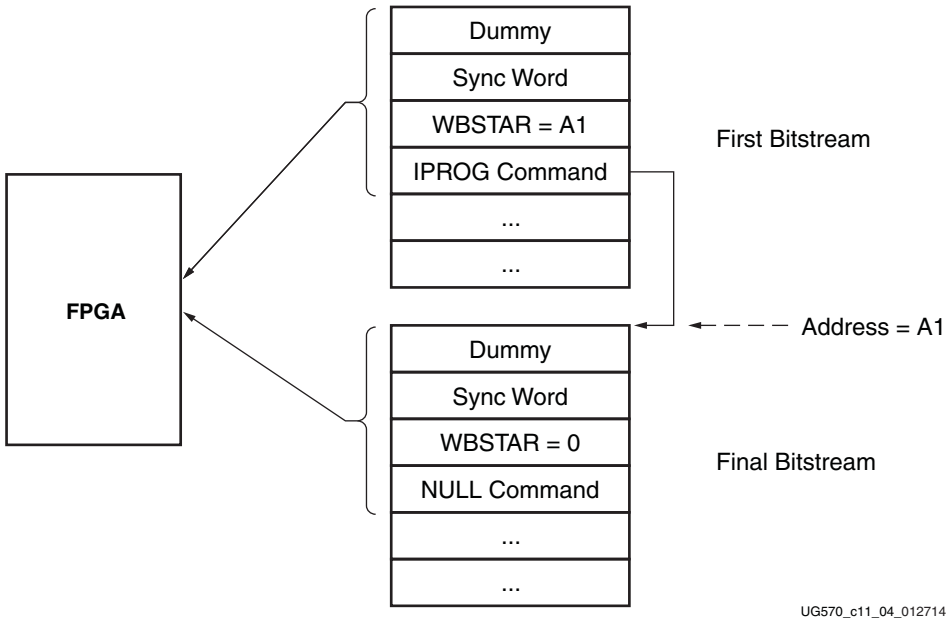


Figure 11-4: **PROG Embedded in the Bitstream**

Status Register for Fallback and IPROG Reconfiguration

UltraScale architecture-based FPGAs contain a BOOTSTS register 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) in Chapter 9.

Table 11-5, Table 11-6, and Table 11-7 show the BOOTSTS values in some common situations.

Table 11-5: **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 11-6: **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 11-7: **IPROG Embedded in First Bitstream, Second Bitstream CRC Error, Fallback Successfully**

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

Notes for Table 11-7:

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.

For an example design, see *MultiBoot and Fallback with SPI Flash in UltraScale FPGAs* (XAPP1257) [Ref 23].

Dynamic Reconfiguration of Functional Blocks

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, ICAP, or SelectMAP ports. However, the dynamic reconfiguration port (DRP) that is an integral part of many functional blocks simplifies this process greatly. Such configuration ports exist in CMTs, System Monitor, serial transceivers, and the integrated block for PCI Express. Refer to the user guides for each block for more information on their dynamic reconfiguration ports.

XILINX

Chapter 12

Configuring Multiple FPGAs

Introduction

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

Note: A daisy chain that includes UltraScale devices must be composed only of devices that are supported by the Vivado tools, from the 7 series and later.

Serial Configuration Modes

Serial configuration modes include serial daisy chain, mixed serial daisy chain, and ganged serial.

Serial Daisy Chain Configuration

Multiple FPGAs can be configured from a single configuration source by arranging the devices in a serial daisy chain. In a serial daisy chain, devices receive their configuration data through their DIN pin, passing configuration data along to downstream devices through their DOUT pin. Data on DOUT is clocked out on the falling edge of CCLK. Data is captured on DIN of the downstream device on the rising edge of CCLK. The device closest

to the configuration data source is considered the most upstream device, while the device furthest from the configuration data source is considered the most downstream device.

In a serial daisy chain, the configuration clock is typically provided by the most upstream device in SPI mode. All other devices are set for slave serial mode. Figure 12-1 illustrates this configuration.

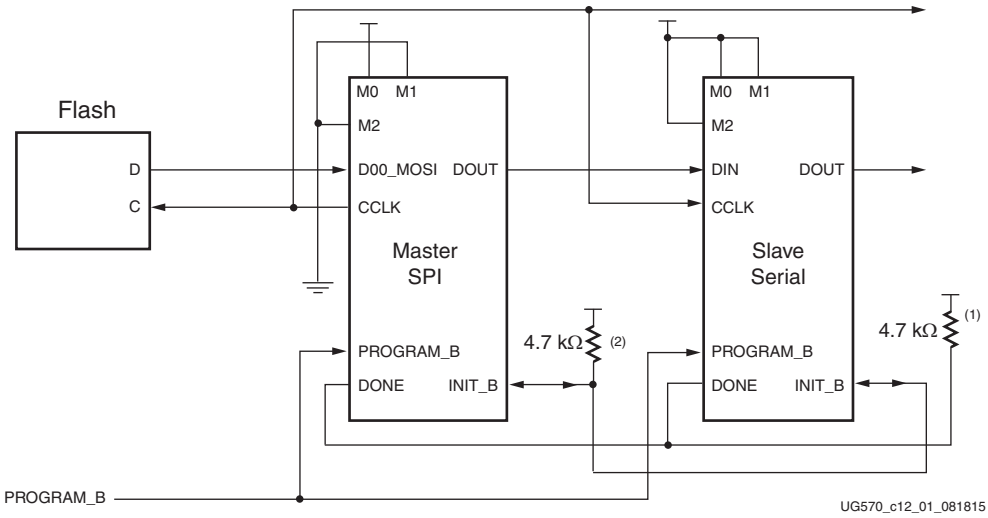


Figure 12-1: **Master/Slave Serial Mode Daisy Chain Configuration Interface Example**

Notes relevant to Figure 12-1:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. See Figure 2-2, page 48 for a more detailed view of the master SPI connections.
4. Fallback MultiBoot is not supported in this 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 CRC Check (Step 7) in Chapter 9.)

After the last device in the chain finishes configuration and passes its CRC check, it enters the start-up sequence. At the *release DONE pin* phase in the start-up 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 pulled High externally. On the next rising CCLK edge, all devices move out of the release DONE pin phase and complete their start-up sequences.

It is important that all DONE pins in a slave serial daisy chain be connected.

Mixed Serial Daisy Chains

UltraScale™ FPGAs can be daisy-chained with earlier 7 series families. There are four important design considerations when designing a mixed serial daisy chain:

- Select a CCLK frequency that is compatible with all devices in the daisy chain.
- UltraScale architecture-based FPGAs should always be at the beginning of the serial daisy chain, with 7 series devices located at the end of the chain.
- All UltraScale architecture-based FPGAs have similar bitstream options. The guidelines provided for UltraScale architecture-based FPGAs 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 UltraScale architecture-based FPGAs and 7 series FPGAs, the limit is 4,294,967,264 bits. The sum of the bitstream lengths for all downstream devices must not exceed this number.

Guidelines and Design Considerations for Serial Daisy Chains

There are a number of important considerations for serial daisy chains:

- Start-up 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

  It is important to connect the DONE pins for all devices in a serial daisy chain. Failing to connect the DONE pins can cause configuration to fail. For debugging purposes, it is often helpful to have a way of disconnecting individual DONE pins from the common DONE signal, so that devices can be individually configured through the serial or JTAG interface.

- DONE pin rise time

1. After all DONE pins are released, the DONE pin should rise from logic 0 to logic 1 in one CCLK cycle. See Table 1-9, page 31 for DONE signal details. 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.

Ganged Serial Configuration

More than one device can be configured simultaneously from the same bitstream using a ganged serial configuration setup (Figure 12-2). In this arrangement, the serial configuration pins are tied together such that each device sees the same signal transitions.

One device is typically set for master SPI mode (to drive CCLK) while the others are set for slave serial mode. For ganged serial configuration, all devices must be identical.

Configuration can also be driven by an external configuration controller as shown in Figure 12-2, reading the bitstream from flash or other memory.

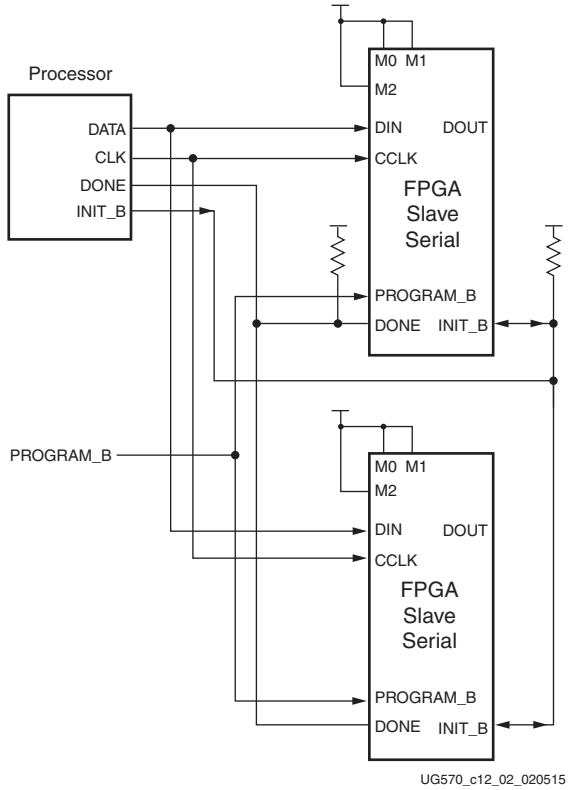


Figure 12-2: **Ganged Serial Configuration Interface Example**

Notes relevant to Figure 12-2:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. All devices must be identical (same IDCODE) and must be configured with the same bitstream.
4. See Figure 3-2, page 59 for a more detailed view of the slave serial connections.

There are several important considerations for ganged serial configuration:

- Start-up 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.

- Connect all DONE pins if using a master device

  It is important to connect the DONE pins for all devices in ganged serial configuration if one FPGA is used as the master device. Failing to connect the DONE pins can cause configuration to fail for individual devices in this case. If all devices are set for slave serial mode, the DONE pins can be disconnected (if the external CCLK source continues toggling until all DONE pins go High).

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

- DONE pin rise time

  After all DONE pins are released, the DONE pin should rise from logic 0 to logic 1 in one CCLK cycle. If additional time is required for the DONE signal to rise, the DonePipe option can be set for all devices in the serial daisy chain.

- Configuration clock (CCLK) as clock signal for board layout

  The CCLK signal is relatively slow, but the edge rates on the UltraScale FPGA's input buffers are very fast. Even minor signal integrity problems on the CCLK signal can cause the configuration to fail. (Typical failure mode: DONE Low and INIT_B High.) Therefore, design practices that focus on signal integrity, including signal integrity simulation with IBIS, are recommended.

- Signal fanout

  Designers must focus on good signal integrity when using ganged serial configuration. Signal integrity simulation is recommended.

- Files for ganged serial configuration

  Files for ganged serial configuration are identical to the files used to configure single devices. There are no special file considerations.

Parallel Configuration Modes

Parallel configuration modes include multiple device SelectMAP, parallel daisy chain, ganged SelectMAP, and ganged asynchronous BPI.

Multiple Device SelectMAP Configuration

Multiple UltraScale FPGAs in slave SelectMAP mode can be connected on a common SelectMAP bus (Figure 12-3). 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 device after configuration, the RDWR_B signal must be handled appropriately. (For details, refer to Chapter 10, Readback Verification and CRC.)

Otherwise, RDWR_B can be tied Low. Refer to Bitstream Loading (Steps 4-7) in Chapter 9.

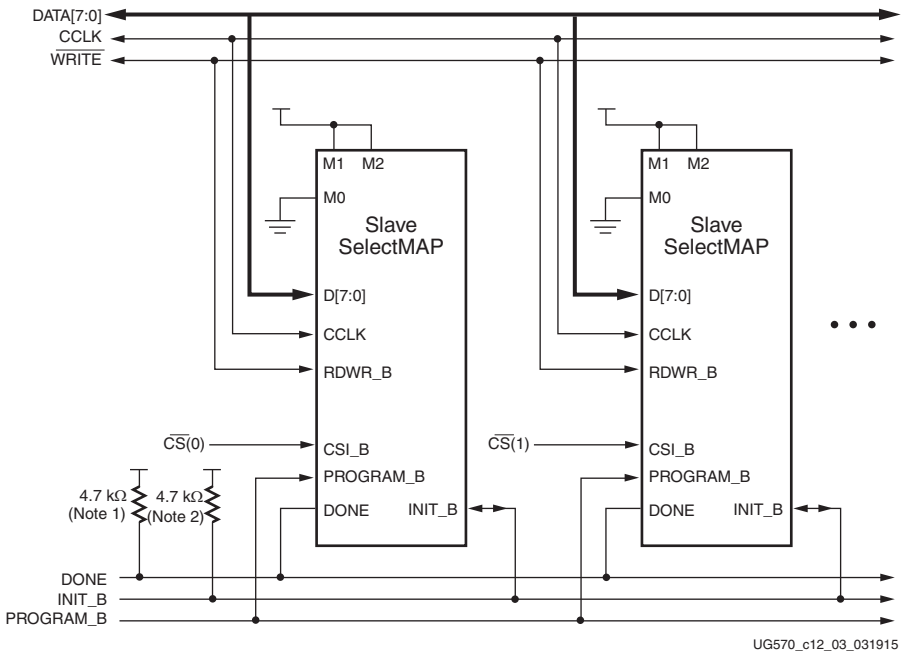


Figure 12-3: **Multiple Slave Device Configuration Interface Example on an 8-Bit SelectMAP Bus**

Notes relevant to Figure 12-3:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.

2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. An external controller such as a microprocessor or CPLD is needed to control configuration.
4. The data bus can be x8, x16, or x32 (for slave SelectMAP).
5. See Figure 5-2, page 80 for a more detailed view of the slave SelectMAP connections.

Parallel Daisy Chain Configuration

UltraScale FPGA configuration supports a parallel daisy-chain. Figure 12-4 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.

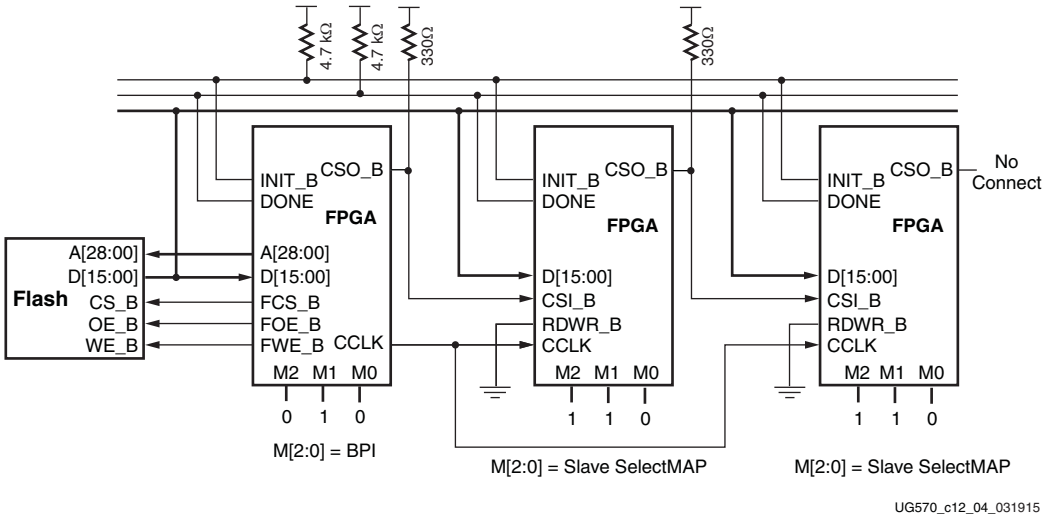


Figure 12-4: **Parallel Daisy Chain Configuration Interface Example**

Notes relevant to Figure 12-4:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up is required.
3. The FCS_B, FWE_B, FOE_B, CSO_B weak pull-up resistors should be enabled, otherwise external pull-up resistors are required for each pin. By default, all dual-mode I/Os have weak pull-downs after configuration.
4. The first device in the chain can be master SelectMAP, slave SelectMAP, or BPI. See Figure 4-2, page 65 for a more detailed view of the BPI connections.
5. Readback in the parallel daisy chain scheme is not supported.

6. Fallback MultiBoot is not supported in this configuration.

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 ground), causing all devices to recognize data presented on the D pins.

All devices can be set for slave SelectMAP mode if an external oscillator is available as illustrated in Figure 12-5, or one device can be designated as the master device.

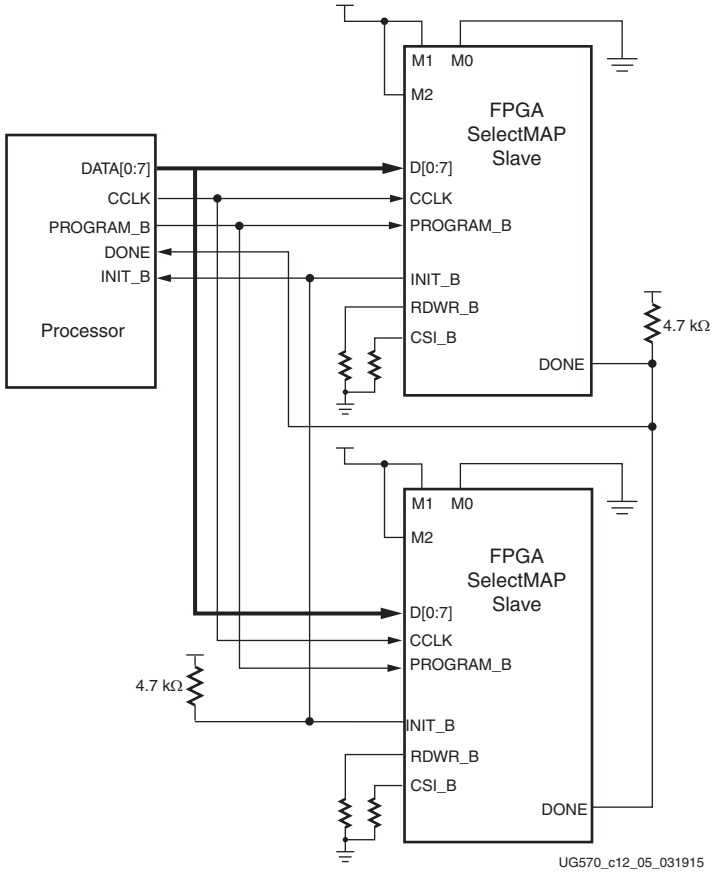


Figure 12-5: **Ganged x8 SelectMAP Configuration Interface Example**

Notes relevant to Figure 12-5:

1. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. See Figure 5-2, page 80 for a more detailed view of the slave SelectMAP connections

If one device is designated as the master, the DONE pins of all devices must be connected. The DONE pin is by default an open-drain output. See Table 1-9, page 31 for DONE signal details. 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.

Ganged Asynchronous BPI Configuration

It is also possible to configure multiple devices simultaneously with the same configuration bitstream by using a ganged master BPI configuration, using asynchronous read mode. The ganged BPI configuration is similar to that shown in Figure 4-4, but with the data bus connected to multiple FPGAs. The DONE and INIT_B pins on all devices are connected together.

Configuration Debugging

Introduction

Some best practices are discussed in this chapter that will help resolve issues that might be encountered when implementing a configuration solution. Topics discussed include:

- File Generation Review
- Status Pin Handling
- Status Register Use and JTAG Access
- Verification and Readback
- Configuration Sequence
  - Configuration Start-Up Considerations
  - Remote Update Considerations
- Initial Debug Steps

File Generation Review

Ensure the bitstream properties and flash programming file options were implemented correctly for the targeted configuration mode. To verify the bitstream generation options used by an image, run the Tcl command:

```
report_property -all [current_design]
```

This command will display all properties applied to a design. Where there are no values displayed, the default is applied. Also, review the flash programming file generation options. Verify that the proper data widths and data ordering options are used for the flash programming file generation. All DRC warnings received during configuration file generation should be reviewed and corrected.

Status Pin Handling

There are physical status pins that are recommended to be accessible on the board for debug. The two most important signals are the INIT_B and DONE signals. The pulsing of INIT_B from Low to High indicates the completion of initialization at power-up, and then a falling INIT_B signal later in the process can indicate a CRC error. Having access to the INIT_B and DONE signals is critical for FPGA configuration debug.

In addition to the status signals, there are key configuration pins that provide helpful information and should be handled carefully to prevent problems during configuration. These pins are listed below:

- Mode pins M[2:0]
- PROGRAM_B pin
- CFGBVS pin
- PUDC_B pin

Mode Pins M[2:0]

The mode pins should be tied and static during configuration. The FPGA reads mode pins at power-up to determine which configuration mode to use. JTAG mode is most commonly used for debugging, and although it is always available, setting the mode pins to select JTAG mode will prevent interference from other configuration modes.

PROGRAM_B Pin

The PROGRAM_B pin re-configures the FPGA and is often tied to a push button for easy access. The pin must be held High during the configuration process.

CFGBVS Pin

The configuration bank voltage select pin (UltraScale FPGAs only) must be tied appropriately to GND or V_{CCO} to support the 1.8V or 3.3V maximum range required by your design.

PUDC_B

The PUDC_B pin determines whether or not I/O pull-ups are enabled during configuration.

More details on the configuration pins are found in Chapter 1, Introduction.

Status Register Use and JTAG Access

The starting point for internal debugging should always be the status register, which requires access to the JTAG port. FPGA status register data can be read in the Vivado® device programmer through JTAG. For example, if the DONE and INIT_B signals are Low, this register captures the specific error conditions that can help identify the type of failure. In addition, the status register allows you to verify the mode pin settings M[2:0] and the bus width detect. Details on status register use are provided in Status Register (00111) in Chapter 9.

Verification and Readback

If FPGA configuration is not successful, a quick JTAG verify or readback operation on the FPGA contents can eliminate issues during programming or verify if a rare SEU event has occurred. To perform a JTAG verify operation with the Vivado device programmer, a mask (`.msk`) file is required and is created during the bitstream generation phase. For more details, see Chapter 10, Readback Verification and CRC.

Configuration Sequence

During the configuration process, there are some basic checks that can be performed to help isolate an issue. Xilinx FPGA bitstreams have a unique header. The header includes a synchronization word and can include an auto detect, a configuration clock type, and a rate setting. For UltraScale™ architecture-based FPGAs, this sync word is shown:

```
AA995566
```

The sync word is a valuable debug parameter. You can scope the data pins and when you see the synchronization word, you know that the bitstream header is seen. Shortly after this, there should be transitions of the increased configuration clock rate if the configuration rate speed-up or external master CCLK (EMCCLK) options are used.

Configuration Start-Up Considerations

There is a common sequence to be followed for the FPGA power-up, described in detail in Chapter 9, Configuration Details. Special options can require modifications to the default sequence. For example, when an MMCM is used, the "MMCM lock" option might need to be used to wait for the MMCM to lock before beginning configuration. There are also the "Wait for PLL" or "DCI match" options. If any of these options are used, then ensure the images in the configuration source are properly spaced for MultiBoot images. Also, when using slave

modes or the master mode EMCCLK option, ensure enough clock cycles are supplied to complete the start-up sequence.

If you do not clock the start-up completely, some of the following symptoms can be observed:

- I/O remains disabled.

  Multi-function configuration and I/O pins operate in LVCMOS rather than the specified I/O standard.

- ICAP interface cannot be accessed from the FPGA logic because the configuration logic is locked.

  This will occur if the device has not reached the end of start-up state. The device can be fully operational before the device reaches this end of start-up state. This can lead to ICAP read and write failures or multi-function pins not operating in the correct I/O standard. This event is indicated by the EOS signal being driven High. This can be observed in the STATUS register or detected in the FPGA using the STARTUPE3 primitive.

  For designs accessing the ICAP, it is good design practice to instantiate the STARTUPE3 primitive. This primitive has an EOS pin, which will indicate when the configuration process has completed and the ICAP is available for read and write access.

Remote Update Considerations

Xilinx FPGAs support MultiBoot and fallback features that make updating systems in the field more robust. Bitstream images can be upgraded dynamically in the field. The MultiBoot and fallback features can be used with all master configuration modes. For more details on these options, see Fallback MultiBoot in Chapter 11.

Initial Debug Steps

- Try configuration using a different mode. For example, if downloading through a cable, try programming the bitstream into a flash device and configuring.
- Try configuration with a different bitstream. If possible, have a known-good configuration file that can be used as a test.
- If using a higher speed CCLK setting or external clock, trying slowing down the clock.
- Reduce the number of non-default configuration options selected.
- Try applying additional clocks at the end of configuration.
- Verify that data is being received properly at the destination devices.

- Verify that you have the latest version of the tools. Even if you must use an earlier version of the implementation tools, the latest configuration tools can be downloaded for free and used independently by going to the download center and selecting "Lab Tools."

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

1. *UltraScale Architecture Gen3 Integrated Block for PCI Express LogiCORE IP Product Guide* (PG156)
2. *Integrated Logic Analyzer Product Guide* (PG172)
3. *SPI Flash Programming Including Bitstream Revision Selection* (XAPP1191)
4. *UltraScale Architecture and Product Overview* (DS890)
5. 3D ICs website
6. *Vivado Design Suite Properties Reference Guide* (UG912)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
8. *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892)
9. *Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS893)
10. *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899)
11. *SPI Configuration and Flash Programming in UltraScale FPGAs* (XAPP1233)
12. *UltraScale FPGA Post-Configuration Access of SPI Flash Memory using STARTUPE3* (XAPP1280)
13. *UltraScale FPGA BPI Configuration and Flash Programming* (XAPP1220)
14. *Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave SelectMAP Mode* (XAPP583)
15. *UltraScale Architecture System Monitor User Guide* (UG580)
16. *UltraScale Architecture Libraries Guide* (UG974)
17. *Bitstream Identification with USR_ACCESS using the Vivado Design Suite* (XAPP1232)
18. *Using Encryption and Authentication to Secure an UltraScale/UltraScale+ FPGA Bitstream* (XAPP1267)
19. *Vivado Design Suite Tcl Command Reference Guide* (UG835)
20. *UltraScale Architecture SelectIO Resources User Guide* (UG571)
21. *Configuration Readback Capture in UltraScale FPGAs* (XAPP1230)
22. *Demonstration of Soft Error Mitigation IP and Partial Reconfiguration Capability on Monolithic Devices* (XAPP1261)
23. *MultiBoot and Fallback with SPI Flash in UltraScale FPGAs* (XAPP1257)
24. *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575)
25. *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909)

26. *Vivado Design Suite Tutorial: programming and Debug* (UG936)
27. *Zynq UltraScale+ Device Technical Reference Manual* (UG1085)
28. Configuration Solution Center
29. Configuration Forum

Please Read: Important Legal Notices