Spartan UltraScale+ FPGAs Configuration

User Guide

UG860 (v1.1) November 12, 2025

AMD

Table of Contents

Introduction

Introduction to the UltraScale Architecture

The AMD UltraScale™ architecture is the first ASIC-class 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 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.

AMD Spartan™ UltraScale+™ devices provide high I/O to logic ratio, integrated memory controllers, and advanced I/O that help Industrial, Vision and Healthcare (IVH), Audio, Video and Broadcast (AVB), Automotive, and other FPGA application developers looking to build cost-optimized solutions. Available in a wide array of packaging options, this family delivers a balance of cost, power, performance, and size.

AMD Artix™ UltraScale+™ devices provide high serial bandwidth and signal compute density in a cost-optimized device for critical networking applications, vision and video processing, and secured connectivity. Coupled with the innovative InFO packaging, which provides excellent thermal and power distribution, Artix UltraScale+ FPGAs are perfectly suited to applications requiring high compute density in a small footprint.

AMD Kintex™ UltraScale+™ devices provide the best price/performance/watt balance in a 16 nm 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.

AMD 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.

AMD Virtex™ UltraScale+™ devices provide the highest performance and integration capabilities in a 16 nm 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.

AMD Zynq™ UltraScale+™ 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.

The UltraScale architecture documentation suite is available at docs.amd.com.

Overview

***Note*:** This user guide covers Spartan UltraScale+ FPGAs boot and configuration. For other UltraScale architecture-based FPGA families, refer to *UltraScale Architecture Configuration User Guide* (UG570).

This chapter provides a brief overview of the boot and configuration methods, updated configuration modes, and additional configuration security features for the Spartan UltraScale+ FPGAs. Spartan UltraScale+ FPGAs enhanced configuration logic includes an optimized platform management controller (PMC) focused on device configuration and security. The dedicated PMC configuration controller runs the fixed BootROM and platform loader and manager (PLM) code to initiate the device configuration process. Subsequent chapters provide detailed descriptions of each boot and configuration feature.

The Spartan UltraScale+ FPGA configuration controller runs a fixed implementation of both the BootROM firmware and the platform loader and manager (PLM). The BootROM responsibilities include:

Like processors, AMD FPGAs are highly flexible user programmable logic devices. FPGAs can be reprogrammed in-system an unlimited number of times. The process of programming or loading the defining data into the FPGA is called configuration. After programming, the PL configuration data is stored in highly robust CMOS configuration latches (CCLs). Although CCLs are reprogrammable like SRAM, CCLs are designed primarily for data integrity. Because the FPGA PL configuration data is stored in CCLs, the device must be reconfigured after it is power cycled.

Configuration is designed to be flexible to accommodate different application needs and, wherever possible, to leverage existing system resources to minimize system costs. FPGAs can configure themselves automatically from an external, nonvolatile memory device. Alternatively, FPGAs can be downloaded or programmed by an external device, such as a microprocessor, DSP processor, microcontroller, PC, or board tester. The configuration data path can be serial to minimize pin requirements, including configuration through the industry-standard IEEE 1149.1 JTAG boundary-scan interface. A parallel configuration data path provides maximum performance and access to industry-standard interfaces, ideal for external data sources like processors.

For Spartan UltraScale+ FPGAs, the programmable device image (PDI) file defines the application-specific FPGA functionality programmed into the device. The PDI file contains the boot header, the platform loader manager fixed firmware for configuration and security, and the programmable logic (PL) data. The PDI is loaded into the FPGA through special configuration pins. These configuration pins serve as the interface for different configuration modes with various bus widths. See the configuration chapters for details on the specific Spartan UltraScale+ supported configuration modes listed:.

- Master Modes
  - Master SPI (x1, x2, x4)
  - Master OSPI (x1, x8)
- Slave Modes
  - Slave Serial (x1)
  - Slave SelectMAP (x8, x16, x32)
  - JTAG boundary-scan (x1)

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. For more information, see the Master Modes section. Optionally, external master configuration clock (`EMCCLK`) can be selected with a software option for advanced use cases. For more information, see the External Master Configuration Clock section.
- In slave configuration modes, `CCLK` is an input to the FPGA driven by an external host. For more information, see the Slave Modes section.

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 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 selected, or partial reconfiguration. This allows specific regions of the FPGA to be reprogrammed with new functionality while applications continue to run in the remainder of the device.

Spartan UltraScale+ Devices and JTAG IDCODEs

The following table shows the devices in the Spartan UltraScale+ family and their associated JTAG IDCODEs. The IDCODEs are used to identify the device. The Spartan UltraScale+ devices and JTAG IDCODEs table also provides the JTAG production IDCODE revision and JTAG instruction length.

Table 1: **Spartan UltraScale+ Devices and JTAG IDCODEs**

Device	JTAG Device IDCODE[31:0] (hex)(1)(2)	Production IDCODE Revision	JTAG Instruction Length (bits)
SU10P	X4E81093	0	6
SU25P	X4E82093	0	6
SU35P	X4E80093	0	6
SU45P	X4EB1093		6
SU55P	X4E90093		6
SU60P	X4EB2093		6
SU65P	X4E99093		6
SU100P	X4E98093		6
SU150P	X4EA1093		6
SU200P	X4EA0093		6

Notes:

1. The “X” in the JTAG IDCODE value represents the revision field (IDCODE[31:28]) which can vary.
2. If the IDCODE does not match the expected value, ensure that `PROGRAM_B` is not held Low.

Differences from Previous Generations

The Spartan UltraScale+ FPGAs include a PMC dedicated configuration controller, advancement on security features, updates on configuration modes, and improvements on key configuration features. The numerous enhancements from the UltraScale architecture-based FPGAs warrant a unique configuration user guide specific to the Spartan UltraScale+ FPGA device.

This section highlights the configuration and security differences from past FPGA devices. The UltraScale+ FPGAs provide a higher performance internal configuration clock (CCLK) with up to double the max frequency with less frequency variation. The UltraScale architecture-based FPGAs also offer a dedicated pin (POR_OVERRIDE) that can be set to reduce the delay when you know the power supplies will ramp quickly for configuration time.

Spartan UltraScale+ FPGAs support similar configuration interfaces as the other UltraScale architecture-based FPGAs and 7 series FPGAs. The following table summarizes the configuration mode differences.

Table 2: **Configuration Modes Comparison**

Configuration Mode	Spartan UltraScale+ FPGAs	Kintex and Virtex UltraScale FPGAs, Artix, Kintex, and Virtex UltraScale+ FPGAs	7 series FPGAs
Slave Serial	Yes(1)	Yes	Yes
Slave SelectMAP	Yes(1)	Yes	Yes
JTAG	Yes(2)	Yes	Yes
Master SPI (x1, x2, x4)	Yes(3)	Yes	Yes
Master OSPI (x1, x8)	Yes(4)	No	No
Master SPI (dual quad, x8)	No	Yes	No
Master BPI	No	Yes	Yes
Master Serial	No	Not recommended(4)	Yes
Master SelectMAP	No	Not recommended(4)	Yes

Notes:

1. Spartan UltraScale+ FPGAs require additional signals for the slave serial configuration mode (CS_B and READY) and slave SelectMAP configuration mode (BUSY). See the specific configuration mode chapter for additional details.
2. Spartan UltraScale+ FPGAs JTAG mode supports secure boot and configuration. For JTAG boundary-scan test, the configuration mode pins must be set to JTAG configuration mode (M[2:0]=101) and the PCB design must allow for the M[2:0] pins to be changed between the JTAG mode for JTAG boundary-scan test and the primary configuration mode for configuration. See Configuration and Mode Setting for more information.
3. Master SPI configuration mode has two M[2:0] selections in Spartan UltraScale+ FPGAs, SPI_24 for 24-bit addressing or SPI_32 for 32-bit addressing flash access.
4. Master SPI and master OSPI configuration modes are recommended over the legacy master serial and master SelectMAP configuration modes because they provide a wider flash density selection and lower cost solution.

The master configuration modes are optimized to work with third-party flash memories. The master SPI configuration mode interface connects to standard SPI NOR flash memory devices. The master OSPI configuration mode interface connects to select octal SPI flash memory devices, see *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for supported flash with each FPGA device.

TIP: *The Spartan UltraScale+ FPGAs add a new configuration mode to configure from select Octal SPI flash memory. The resulting 8-bit wide configuration data bus reduces the configuration time while still allowing for the use of standard, high-speed, low-cost SPI NOR configuration memories.*

In addition to the dedicated configuration bank 0, Spartan UltraScale+ FPGAs have one or two multi-function I/O banks that contain additional configuration mode pins. The multi-function I/O banks (Bank 65, Bank 66) availability depend on the family device member. See the *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575) for more details. Table 28 describes the configuration pin usage and the associated banks. Configuration interfaces can be powered at 1.5V or 1.8V. Spartan UltraScale+ also supports the multi-function I/O configuration banks at 1.2V to 3.3V when the bank is not used by the configuration mode interface.

The Spartan UltraScale architecture-based FPGAs support the internal configuration access port (ICAP) for soft error mitigation (SEM) IP. The media configuration access port (MCAP) provides a connection to the integrated block for PCI Express®. MCAP is provided in select Spartan UltraScale+ FPGAs as shown in Figure 17. Spartan UltraScale+ FPGAs can support authentication and encryption on MCAP. A small initial PDI can be loaded quickly at power-up to enable the PCIe interface. The rest of the configuration can be loaded through the PCIe interface. This is known as tandem PCIe and it is supported on all Spartan UltraScale+ FPGAs with the PCIE4CE. See the *UltraScale Architecture and Product Data Sheet: Overview* (DS890) for more information.

The following table summarizes the additional differences between Spartan UltraScale+ FPGAs, 7 series FPGAs, and UltraScale architecture-based FPGA devices.

Table 3: **Differences Between Families**

Feature Description	Spartan UltraScale+ FPGAs	Artix UltraScale+, Kintex UltraScale+, and Virtex UltraScale+ FPGAs	Kintex UltraScale and Virtex UltraScale FPGAs	7 Series FPGAs
Feature Type				
General				
Configuration Controller	PMC dedicated configuration controller (runs BootROM/PLM)	Configuration engine		
Programming image	Programmable device image (PDI)	Bitstream (.bit)		
Fallback	Yes, increments by 32 KB until reaches search llimit	Yes, jumps to address 0x00000000		
Readback CRC	No, use SEM IP instead		No, use SEM IP instead (SEM IP is not supported in KU025)	Yes
MCAP	Yes (3)	Yes		N/A
SelectMAP Readback (Capture and Verify)	No	Yes		
Configuration frame size (32-bit words)	93		123	101

Table 3: **Differences Between Families** *(cont'd)*

Feature Description	Spartan UltraScale+ FPGAs	Artix UltraScale+, Kintex UltraScale+, and Virtex UltraScale+ FPGAs	Kintex UltraScale and Virtex UltraScale FPGAs	7 Series FPGAs
Feature Type				
Design Entry				
AXI32 primitive	Yes, supports runtime access to crypto blocks AES-GCM, SHA3, PUF, point multiplier for ECC, TRNG, eFUSE programming, and post-configuration flash programming	No		
FUSE_CLK primitive	Yes, used for eFUSE programming	No		
SEMIP	Yes, faster, uses FRAME_ECCE4	Yes, faster, uses FRAME_ECCE4	Yes (except KU025), faster, uses FRAME_ECCE3	Yes, uses FRAME_ECCE2
Device DNA	DNA_PORTE2, 96 bits	DNA_PORTE2, 96 bits		DNA_PORT, 57 bits
MASTER_JTAG Primitive	Yes, internal secure access to the JTAG logic	Yes, internal secure access to the JTAG logic		N/A
CAPTUREE2 Primitive	No			Yes
FRAME_ECC Primitive	FRAME_ECCE4, reserved for SEM IP		FRAME_ECCE3, reserved for SEM IP	FRAME_ECCE2
ICAP Primitive	ICAPE3 reserved for SEM IP; 32-bit, additional status signals	ICAPE3; 32-bit, additional status signals		ICAPE2; 8-bit, 16-bit, 32-bit
STARTUP Primitive	STARTUPE3, adds access to dedicated configuration pins (`FCS_B` and `D[03:00]`), for post configuration access, KEYCLEARB used as secure use case tamper_fabric_b	STARTUPE3		STARTUPE2
Configuration Pin				
POR_OVERRIDE	Yes			No
Configuration Bank Voltage Selector (CFGBVS) pin	No		Yes	
RDWR_B and FCS_B Pins	RDWR_FCS_B combined, dedicated, reduced number of configuration pins			RDWR_B FCS_B; separate, multi-function
D[03:00], PUDC_B	Dedicated, included in STARTUPE3			Multi-function, not included in STARTUPE2

Table 3: **Differences Between Families** *(cont'd)*

Feature Description	Spartan UltraScale+ FPGAs	Artix UltraScale+, Kintex UltraScale+, and Virtex UltraScale+ FPGAs	Kintex UltraScale and Virtex UltraScale FPGAs	7 Series FPGAs
Feature Type				
Slave SelectMAP BUSY multi-function pin	Yes	No		
Slave Serial CS_B and READY dedicated pins	Yes	No		
Slave Serial DOUT pin	Dedicated, for slave serial daisy-chain	Multi-function, for slave serial, master serial, or SPI x1 mode daisy-chain.		
Master SPI configuration mode settings	M[2:0]=001 for SPI_24 (24 bit addressing); M[2:0]=010 for SPI_32 (32 bit addressing)	M[2:0]=001		
CCLK (configuration rate options)	21, 25, 31, 42, 51, 56, 63, 72, 85, 102, 127	2.7, 5.3, 8.0,10.6, 21.3, 31.9,36.4, 51.0, 56.7,63.8, 72.9, 85.0,102.0, 127.5,170.0	3, 6, 9, 12, 22,33, 40, 50, 57,69, 82, 87, 90,110, 115, 130, 148	3, 6, 9, 12, 16,22, 26, 33, 40,50, 66
Security				
Encryption	AES-GCM; similar performance to standard configuration	AES-GCM; similar performance to standard configuration		AES-CBC
Authentication	AES-GCM, HSS, ECDSA P-384, LMS(1)	AES-GCM and RSA	AES-GCM and RSA (except KU025)	HMAC
PUF	Yes	No		
TRNG	Yes	No		
A-HWRoT/S-HWRoT Secure Boot	Yes	No		
Access to crypto blocks (AES-GCM, SHA3, PUF, point multiplier for ECC, TRNG)	Yes	No		
Power Rail and Banks				
Bank 0 Voltage	1.5V-1.8V	1.5V-1.8V	1.5V-3.3V	
Multi-function banks voltage for configuration	1.5V-1.8V	1.5V-1.8V	Kintex except KU095: 1.5V-3.3V; Virtex and KU095: 1.5V-1.8V	1.5V-3.3V
Multi-function banks	65(3), 66(2)	65		14, 15
Battery-Backed RAM (BBRAM) and VBATT power rail	No	Yes		
Configuration Mode				
Master SPI or Slave SelectMAP Daisy-chain	No	Yes		

Table 3: **Differences Between Families** *(cont'd)*

Feature Description	Spartan UltraScale+ FPGAs	Artix UltraScale+, Kintex UltraScale+, and Virtex UltraScale+ FPGAs	Kintex UltraScale and Virtex UltraScale FPGAs	7 Series FPGAs
Feature Type				
Master SPI auto bus detection	Yes	No		
JTAG Access in Non-JTAG mode pin setting	JTAG PDI load available after issue of JPROG JTAG instruction	JTAG bitstream load available		

Notes:

1. See Table 39 for details on which devices are supported.
2. Spartan UltraScale+ FPGA multi-function I/O bank usage is dependent on the device and configuration mode. See Configuration Pins for more information. Multi-function configuration banks can run at 1.2V to 3.3V for user design logic if the bank is not used by the selected configuration mode.
3. MCAP is supported by select family members, see Figure 17.

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 including: overall setup, speed, cost, and complexity. These topics and others are discussed in this chapter.

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 PDI and check for device operation conflicts.
- Generate the configuration PDI using the latest version of the AMD tools targeting the version of the device that is used (ES or production).

> **IMPORTANT!** *A programmable device image generated for an engineering sample should not be used in production devices. Production devices must always use PDI with the correct production device target.*

- Correct DRC errors and review any warnings when the configuration PDI is generated.

See Chapter 2: Design Considerations for additional information on boot and configuration trade-offs for early planning phases of a design.

FPGA Configuration Data Source

Spartan UltraScale+ 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.

Configuration Image Size

Spartan UltraScale+ FPGAs use a proprietary programmable device image (PDI) file format to boot and configure. The AMD Vivado development platform generates the PDI. For more information on the PDI, see Chapter 3: Design Tools .

The programmable device image (PDI) includes a fixed implementation of the boot header and the platform loader and manager (plm.elf), and a user design programmable logic (PL) configuration data (.rcdo). The maximum size for a specific device PDI image can be affected by PDI generation options, such as compression. Compressed images can change in size between design iterations so memory space should be reserved for the full uncompressed PDI size. In addition, security features can affect the image size. For security features, a general guideline would be to triple the boot and PL contributors for an image size estimation.

The PDI file format estimated sizes should be considered during system architecture planning to ensure there is enough storage capacity. See Table 4 and Table 5 for an estimation on non-secure PDI sizes.

Table 4: **Estimated Maximum Size for Boot Header PLM Contributors to PDI Size**

PDI Contributor	Maximum Size (bits)
Boot Header	6,656
PLM (.elf) for SU10P, SU25P, SU35P	331,776
PLM (.elf) for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, SU200P	593,920

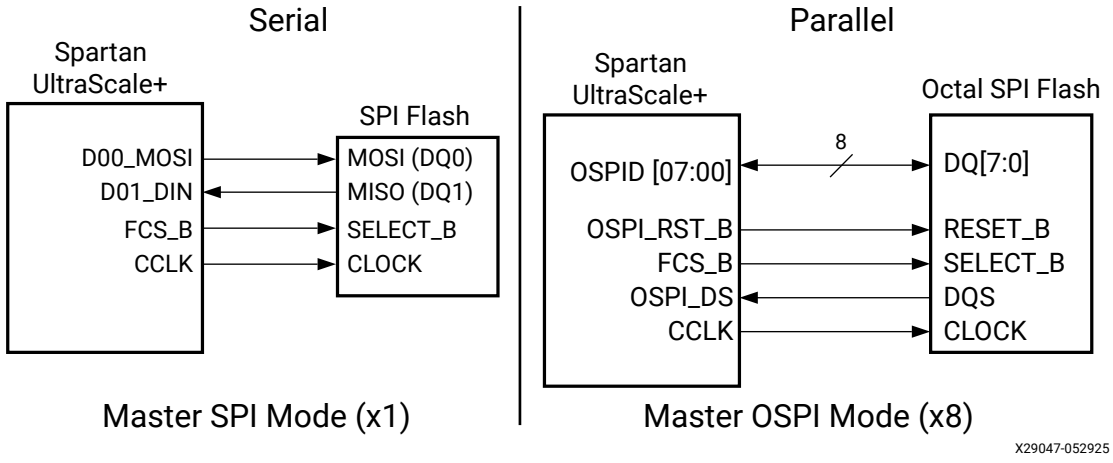
Table 5: **Estimated Maximum Uncompressed Size for PL Contributor to PDI Size**

Device	PL Configuration Maximum Size (bits)
SU10P	10,606,464
SU25P	10,606,464
SU35P	10,606,464
SU45P	19,922,944
SU55P	15,728,640
SU60P	29,360,128
SU65P	19,922,944
SU100P	29,360,128
SU150P	57,817,728
SU200P	57,817,728

Master Modes

FPGA master configuration modes are self-loading and available with a serial or parallel data path. In master mode, the FPGA PDI 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 and chip select to the flash memory device. See the following figure for details.

Figure 1: **Master Configuration Modes**

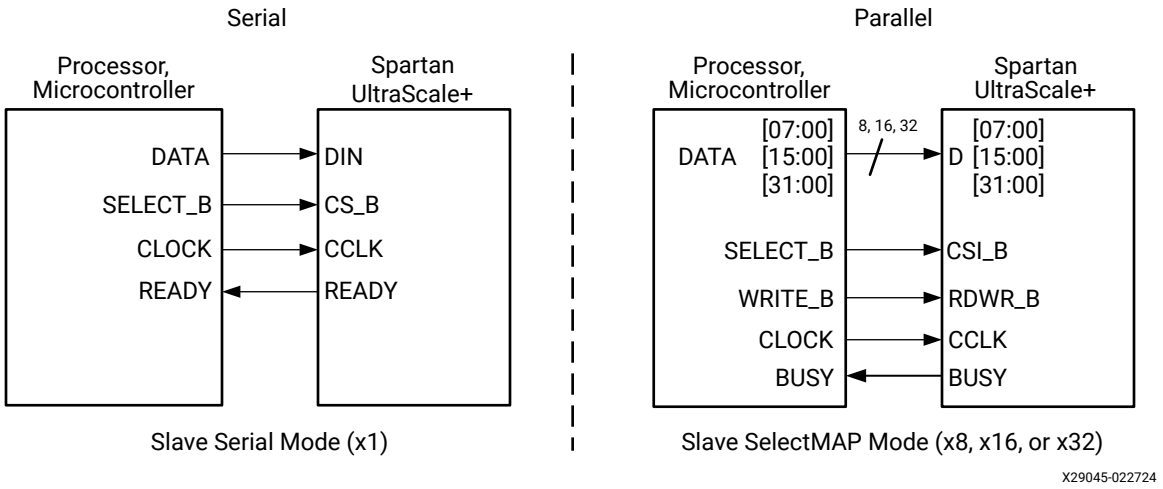


The CCLK internal frequency is selected with a PDI option. For the master configuration modes an alternate clock, the external master configuration clock (EMCCLK), can be enabled (see External Master Configuration Clock). Master SPI mode supports a serial x1 option as shown in Figure 1, but the mode also supports x2 and x4 data bus widths.

Slave Modes

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

Figure 2: **Slave Configuration Modes**

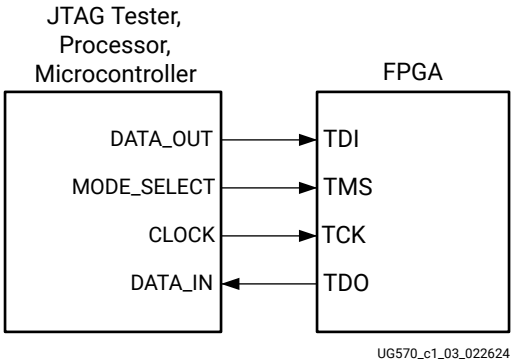


The Spartan UltraScale+ FPGA slave serial mode is a simple interface that consists of a clock, serial data input, chip-select input, and a READY monitor output. The Slave SelectMAP mode is an 8, 16, or 32-bit wide data interface that includes a chip-select input, a read/write control input, and a `BUSY` monitor signal.

JTAG Connection

The JTAG mode is also a serial configuration mode, popular for prototyping and highly used for board test. The four-pin JTAG boundary-scan interface is common on board testers and debugging hardware. The AMD programming cables for Spartan UltraScale+ FPGAs use the JTAG interface for prototype download and debugging. Regardless of the primary configuration mode used in a application, the physical JTAG configuration mode pin setting should be included to ease design development and debug. See the following figure for details.

Figure 3: **JTAG Configuration Mode**



A Basic Configuration Solution

In a basic configuration solution, the FPGA automatically retrieves its PDI from a flash memory device at power-on. The FPGA has a serial peripheral interface (SPI) through which the FPGA can read a PDI from a standard serial NOR flash device.

A Low-Cost Configuration Solution

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

- If there is spare nonvolatile memory available in the system, the PDI 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, slave SelectMAP, or JTAG configuration mode.
- If nonvolatile memory is already required for an application, it is possible to leverage it to also store FPGA configuration PDI with small or no incremental cost. For example, the FPGA configuration PDI 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.

- 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.
- 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.
- The maximum configuration clock frequency is dependent on the configuration mode and application implementation. If the configuration time from power-up is required, then T_{POR} should be added to the configuration time.

External Master Configuration Clock

By default, the master configuration modes use an internally generated configuration clock source (CCLK) specified by a PDI generation option (BITSTREAM.CONFIG.CONFIGRATE). 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 Spartan UltraScale+ 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 15% tolerance means that the ConfigRate setting cannot be faster than 127 MHz. However, an external clock source can have less tolerance to enable faster frequency up to the specification. 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 PDI generation option (BITSTREAM.CONFIG.EMCCLK) and selecting a frequency within the data sheet specification. The default is disable (use the internal CCLK).
2. Connecting the EMCCLK pin on the board to your board's oscillator or other clock source.

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 PDI header is read. If the BITSTREAM.CONFIG.EMCCLK option is enabled then the FPGA switches from the CCLK to the clock found on the EMCCLK pin.

MultiBoot Solution

Spartan UltraScale+ FPGAs supports fallback and MultiBoot. The behavior in this family compared to the other UltraScale+ FPGA families varies. In Spartan UltraScale+, conditions that trigger a fallback include: valid identification signature XLNX (0x584C4E58) not seen, the PLM offset/length word is corrupted, or when an IDCODE error or CRC error is seen.

If a valid boot header is not seen in the PDI at address 0x0000_0000 then the BootROM automatically jumps to the next 32 KB address and attempts to search for the signature XLNX. If no image is found then the FPGA continues to increment another 32 KB until it reaches the maximum search limit for the master configuration mode. See Table 6 for more information. If a Master SPI or Master OSPI configuration attempt fails, the JTAG Status register [14:12] internal configuration mode setting is changed to JTAG mode for debug.

IMPORTANT! *In other UltraScale+ families, it is recommended to have a golden image at location 0x0000_0000 that those devices fallback to if the update image is corrupted in a higher address location. Spartan UltraScale+ FPGA methodology differs because the update image at 0x0000_0000 allows a corrupted image to fallback to an upper address location.*

Table 6: **Fallback and MultiBoot Search Limits**

Configuration Mode	Fallback and MultiBoot	Maximum Search Limit
Master SPI(1) SPI_24	Yes	128 Mbits
Master SPI(1) SPI_32	Yes	4 Gbits
Master OSPI	Yes	8 Gbits
JTAG	No	N/A
Slave Serial	No	N/A
Slave SelectMAP	No	N/A

Notes:

1. For Master SPI configuration mode x4 dual-stacked, only the first flash device can be accessed during the BootROM phase.

Protecting a Programmable Device Image

A programmable device image (PDI) defines the device's functionality and loads into the device during power-on. Because this configuration data is stored off chip, a possibility of unauthorized duplication or modification exists.

There are multiple techniques to increase the protection of the PDI and any embedded intellectual property (IP) cores. One method to increase the protection of the confidentiality of your IP is to encrypt the configuration data using an AES-256 key. This technique allows for off-chip storage of your IP protected with high grade encryption. PDI authentication is another method that verifies the authenticity and integrity of the PDI.

Loading Multiple FPGAs

Generally, each FPGA in a system has a unique PDI. Multiple, different FPGA PDI's can share a single configuration flash memory by leveraging a configuration slave serial daisy-chain. However, if all the FPGAs in the application have the same part number and use the same PDI, only a single PDI image is required and programming can be done through ganged configuration. Ganged configuration is supported in the slave serial and slave SelectMAP modes.

Power-On Considerations

When using the Spartan UltraScale+ FPGA family, review the following power-on considerations:

- Ensure the proper power-on behavior and guidelines from the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) are followed. See Power-On Reset for more information.
- Ensure the flash is powered up and ready to receive commands and/or clocks before the FPGA begins to send them. See Power-On Sequence Precautions for Flash for more information.
- Ensure that the `POR_OVERRIDE` pin is tied Low unless the application can meet the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) requirement to shorten the POR delay for a faster configuration time. See POR_OVERRIDE for more information.

Power-On Reset

To ensure proper power-on behavior, the guidelines in the respective family data sheet must be followed. For configuration, *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) 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 T_{POR} 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 deasserting the `INIT_B` pin, sampling the Mode pins, and starting to toggle the `CCLK` if master mode is selected.

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. Special attention to the FPGA and flash power-on sequence or power-on ramps is essential. This is 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. The power-on sequence or power supply ramps can cause the FPGA to awaken or start before the flash, or vice versa. 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! *Master OSPI configuration mode with a bus width of x8 requires multi-function I/O configuration bank in addition to the V_{CCO_0} that is built into the power-on sequence requirement of the FPGA. Ensure the multi-function I/O configuration bank used 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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) 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.

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 V_{CCO_0}, V_{CCAUX}, V_{CCBRAM}, and V_{CCINT} power supplies are all ramped up to 95% of their normal value in a total time of <= 2 ms, the `POR_OVERRIDE` pin can be tied High at power-up (connected to the V_{CCINT} supply rail). The POR delay is shortened as specified in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) (fast POR counter) with the `POR_OVERRIDE` pin tied High. When the `POR_OVERRIDE` pin is Low (connected to GND), the POR delay is longer (slow POR counter). Connect `POR_OVERRIDE` 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.

Chapter 3

Design Tools

Before implementing the design and generating the PDI data file, review the configuration settings to make sure they are correct for your design. This section introduces the Spartan UltraScale+ design flow steps and the PDI programming file format.

Design Flow

Spartan UltraScale+ FPGAs high-level design flow in the Vivado Design Suite is similar to other UltraScale+ FPGAs. The key difference is the programming file generation step. Other UltraScale+ FPGAs use the bitstream programming file. Spartan UltraScale+ FPGAs require additional information for the PMC configuration controller to successfully boot and configure. The additional information is included in the file programming format called a programmable device image (PDI). See *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information.

IMPORTANT! *Spartan UltraScale+ FPGAs require a programmable device image (.pdi) and cannot be programmed with a bitstream (.bit) file.*

Figure 4: **Spartan UltraScale+ FPGA System Level Design Flow**

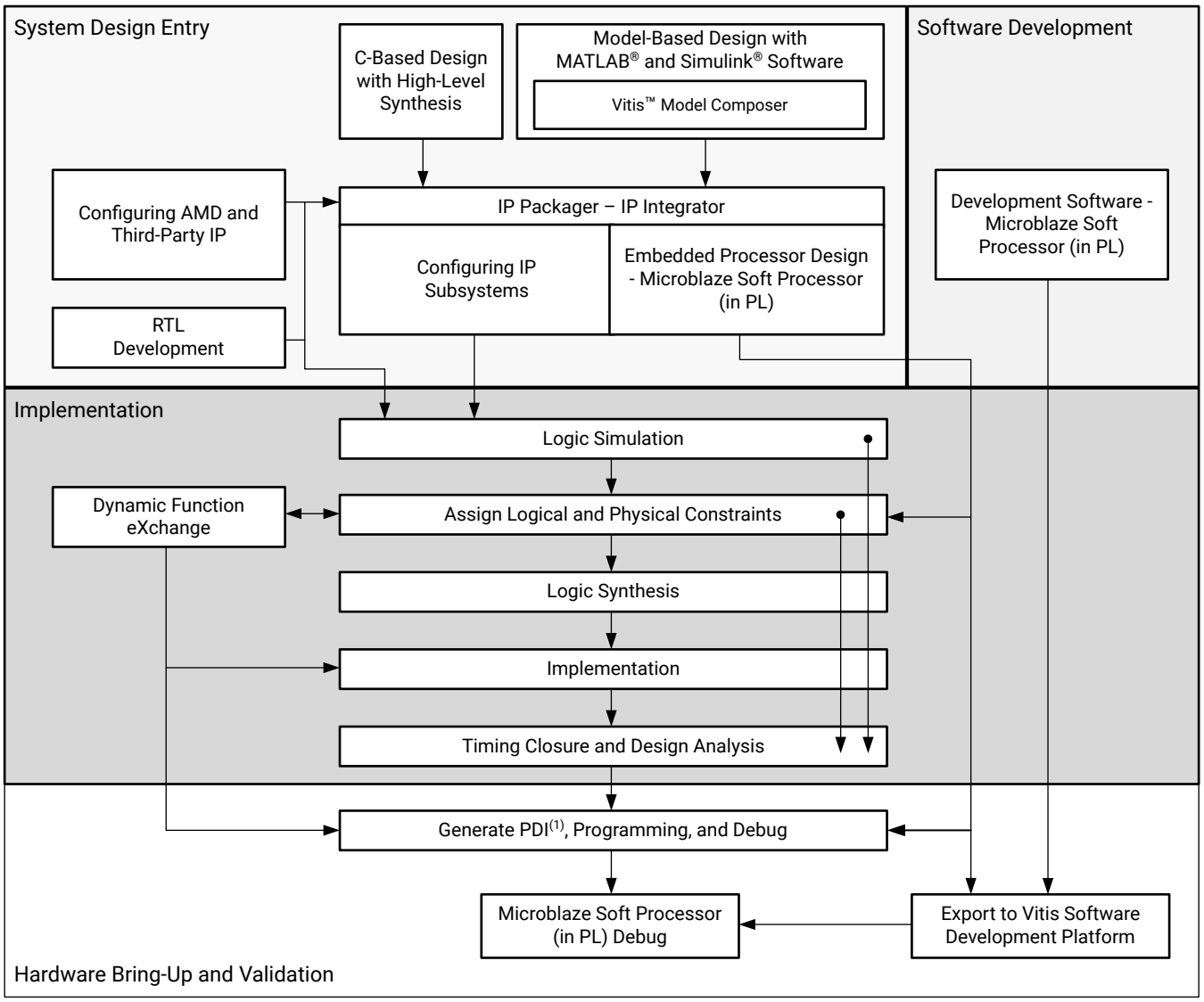


Figure Notes:

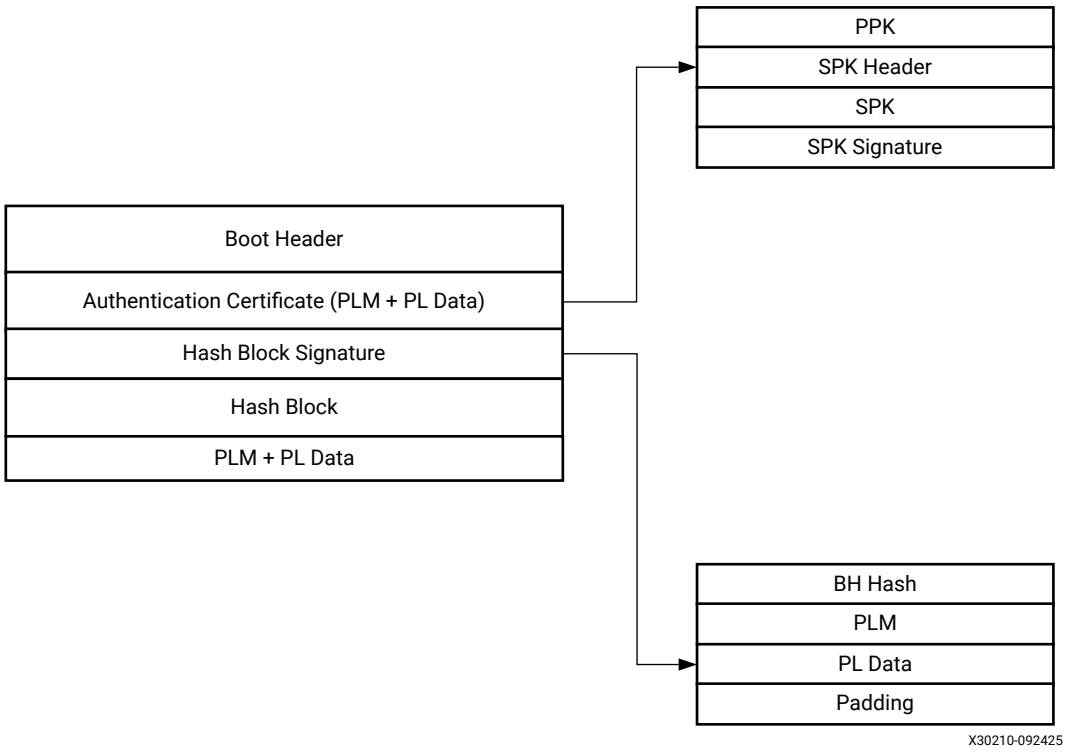
1. For Spartan UltraScale+ devices, the Vivado project mode (Flow Navigator - Program and Debug: Generate Device Image) will directly generate a PDI file for programming. For Spartan UltraScale+ devices, when using the Vivado non-project mode the write_bitstream command will generate the programmable logic .rdco (bitstream data) and the design.bif that are inputs into Bootgen. The Bootgen command must be run after generating the programmable logic .rdco. See example command:

```
bootgen -arch spartanup -image <design.bif> -o <design.pdi
```

PDI File Format

The PDI file format is loaded by the platform management controller into the Spartan UltraScale+ FPGA. A non-secure PDI contains a boot header, PLM (.elf), programmable logic (PL data bitstream equivalent is the .rcdo), and hash block for integrity checking. A secure PDI image can contain additional physically unclonable function (PUF) and public keys (SPK/PPK). The PUF helper data is stored in an additional PDI and is unique per device. See the following figure for an example of the secure PDI format without the PUF helper data.

Figure 5: **Secure PDI Format Example**



***Note*:** Artix, Kintex, and Virtex UltraScale+ FPGA families user designs were loaded with a PL bitstream (.bit). Spartan UltraScale+ FPGA new security features and configuration mode updates require boot header and PLM .elf firmware in addition to the programmable logic configuration data .rcdo (equivalent to bitstream). These components are included in the PDI file format that is required for loading the Spartan UltraScale+ FPGA. See the *Bootgen User Guide* (UG1283) for more information on the Spartan UltraScale+ PDI file format and the boot header.

Configuration PDI Properties

Spartan UltraScale+ FPGAs are programmed from a programmable device image (.pdi) to boot and configure. Configuration features are enabled through PDI properties (BITSTREAM.<category>.<property>) that are set in a Xilinx design constraints (XDC) file or through a Vivado Tcl command before execution of the write_bitstream command. Some examples of configuration and security features that are enabled with a PDI property include increasing the CCLK rate, using the external master configuration clock pin, and enabling OSPI DDR mode. For details on the available PDI properties see the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

Chapter 4

Design Entry

Introduction

Select AMD Spartan™ UltraScale+™ FPGA design elements 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.

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 *Spartan UltraScale+ Libraries Guide* (UG1704). These primitives all require instantiation in a design. Instantiation templates are found in the Libraries Guide and in the AMD Vivado™ language templates.

IMPORTANT! *FRAME_ECCE4 and ICAPE3 should only be instantiated through use of the soft error mitigation (SEM) IP.*

- AXI32
- BSCANE2
- DNA_PORTE2
- EFUSE_USR
- FRAME_ECCE4
- FUSE_CLK
- ICAPE3
- MASTER_JTAG
- STARTUPE3
- USR_ACCESSE2

AXI32

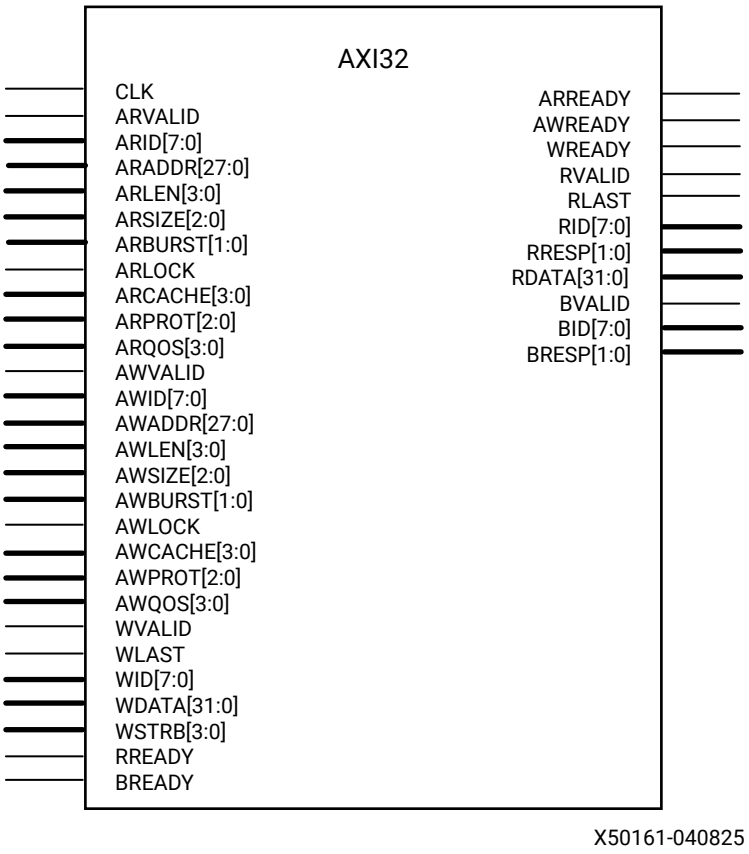
The AXI32 primitive is the hardened 32-bit AXI4 interface from the programmable logic (PL) to the platform management controller (PMC). The primitive supports run-time access to crypto blocks AES-GCM, SHA3, PUF, point multiplier for elliptic curve cryptography (ECC), true random number generator (TRNG), post-configuration flash programming, and eFUSE programming. The AXI32 primitive can be accessed in IP integrator (IPI) with the PMC Bridge IP or using the XPM_PMC_BRIDGE module to instantiate AXI32 in the user design.

For the AXI32, the AXI4 interface supports burst length 16, address signal limited to 28-bits, and AXI ID signals limited to 8-bits (256 possible IDs).

Primitive

The following figure shows the AXI32 primitive.

Figure 6: **AXI32 Primitive**



Pin Descriptions

The following table defines the AXI32 pin connections.

Table 7: **AXI32 Pin Descriptions**

Pin	Type	Width	Description
CLK	input	1	PL AXI clock
ARVALID	input	1	Read address valid
ARREADY	output	1	Read address ready
ARID[7:0]	input	8	Read address ID
ARADDR[27:0]	input	28	Read address
ARLEN[3:0]	input	4	Burst length
ARSIZE[2:0]	input	3	Transfer size
ARBURST[1:0]	input	2	Burst type
ARLOCK	input	1	Lock type
ARCACHE[3:0]	input	4	Memory type
ARPROT[2:0]	input	3	Protection type
ARQOS[3:0]	input	4	Quality of Service (QOS) identifier sent for each read transaction
AWVALID	input	1	Write address valid
AWREADY	output	1	Write address ready
AWID[7:0]	input	8	Write address ID
AWADDR[27:0]	input	28	Write address
AWLEN[3:0]	input	4	Burst length
AWSIZE[2:0]	input	3	Transfer size
AWBURST[1:0]	input	2	Burst type
AWLOCK	input	1	Lock type
AWCACHE[3:0]	input	4	Memory type
AWPROT[2:0]	input	3	Protection type
AWQOS[3:0]	input	4	Quality of Service (QOS) identifier sent for each write transaction
WVALID	input	1	Write valid
WREADY	output	1	Write ready
WLAST	input	1	Write last
WID[7:0]	input	8	Write ID tag
WDATA[31:0]	input	32	Write data
WSTRB[3:0]	input	4	Write strobes
RVALID	output	1	Read valid
RREADY	input	1	Read ready
RLAST	output	1	Read last
RID[7:0]	output	8	Read ID tag

Table 7: **AXI32 Pin Descriptions** *(cont'd)*

Pin	Type	Width	Description
RRESP[1:0]	output	2	Read response
RDATA[31:0]	output	32	Read data
BVALID	output	1	Write response valid
BREADY	input	1	Response ready
BID[7:0]	output	8	Response ID tag
BRESP[1:0]	output	2	Write response

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 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.

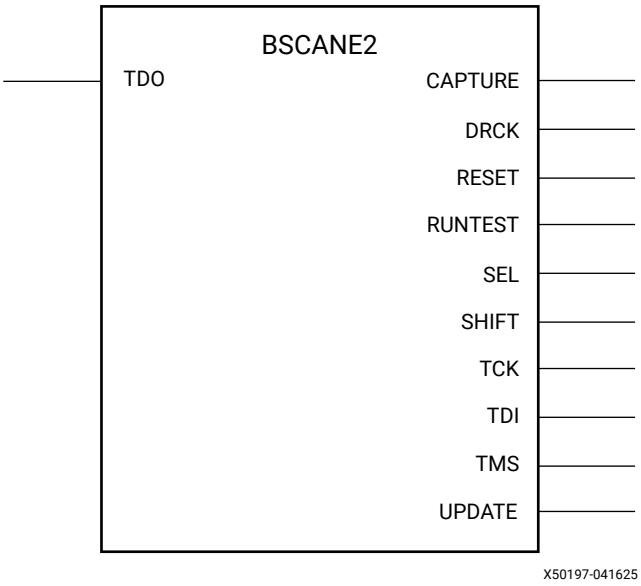
The BSCANE2 primitive is identical to that found in the 7 series and UltraScale architecture FPGAs.

For more details on boundary-scan logic, see Chapter 11: Boundary-Scan and JTAG Configuration Mode.

Primitive

The following figure shows the BSCANE2 primitive.

Figure 7: **BSCANE2 Primitive**



Pin Descriptions

The following table defines the BSCANE2 pin connections.

Table 8: **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-DR state. Internal logic should update from the internal scan register on the rising edge of the UPDATE signal.

Attributes

The following table describes the BSCANE2 primitive attributes.

Table 9: **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.

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 to shift data into an internal scan chain. The `TCK` and `TMS` pins are similarly monitored through the BSCANE2 primitive.

AMD 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 10: **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 AMD into the device via eFUSE bits, and is unchangeable, making it tamper resistant. The device DNA is primarily used to identify the specific device.

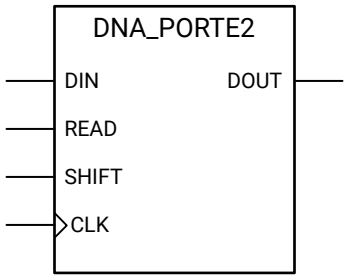
External applications can access the DNA value through the JTAG port. FPGA designs can access the DNA internally through a device DNA Access Port, which requires instantiation of the DNA_PORTE2 primitive. The DNA_PORTE2 primitive controls a dedicated 96-bit shift register for capturing and shifting the 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).

IMPORTANT! *The DNA_PORTE2 and the FUSE_CLK primitives cannot be used in the same design.*

Primitive

The following figure shows the DNA_PORTE2 primitive.

Figure 8: **DNA_PORTE2 Primitive**



Pin Descriptions

The following table 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 11: **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 2).
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. The following table describes the DNA_PORTE2 primitive attributes.

Table 12: **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, see Chapter 12: Security, eFUSEs, and Device DNA.

EFUSE_USR

Each Spartan UltraScale+ FPGA has 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.

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 read through the JTAG port, with bit 0 shifted out first.

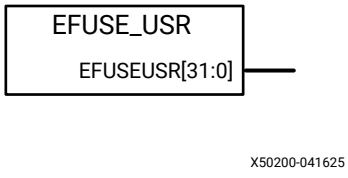
For internal access, the EFUSE_USR primitive must be instantiated. EFUSE_USR provides parallel access to all 32 bits.

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

Primitive

The following figure shows the EFUSE_USR primitive.

Figure 9: **EFUSE_USR Primitive**



Pin Descriptions

The following table describes the EFUSE_USR primitive pins.

Table 13: **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

The following table describes the EFUSE_USR primitive attributes.

Table 14: **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_ECCE4

The FRAME_ECCE4 primitive is reserved. This primitive is used when implementing the Soft Error Mitigation (SEM) IP. The SEM IP generated for Spartan UltraScale+ FPGAs are designed to use the FRAME_ECCE4 primitive automatically.

FUSE_CLK

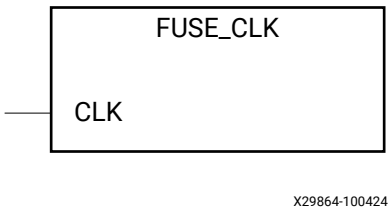
The Spartan UltraScale+ FPGA eFUSE programming operations require a clock source to be supplied through either the `EMCCLK` pin or FUSE_CLK primitive. See the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) for eFUSE programming clock requirements.

IMPORTANT! *The DNA_PORTE2 and the FUSE_CLK primitives cannot be used in the same design.*

Primitive

The following figure shows the FUSE_CLK primitive.

Figure 10: **FUSE_CLK Primitive**



Pin Descriptions

Table 15: **FUSE_CLK Pin Descriptions**

Attribute	Type	Width	Description
CLK	Input	1	Clock used for eFUSE programming.

ICAPE3

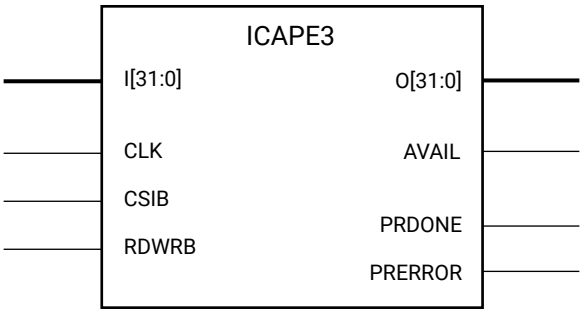
The ICAPE3 primitive is reserved. This primitive is used when implementing the soft error mitigation (SEM) IP. The SEM IP generated for Spartan UltraScale+ FPGAs are designed to use the ICAPE3 primitive automatically.

If persist is set, the ICAPE3 is disabled. In addition, JTAG and MCAP have priority over ICAPE3.

Primitive

The following figure shows the ICAPE3 primitive.

Figure 11: **ICAPE3 Primitive**



Pin Descriptions

The following table describes the ICAPE3 primitive pins.

Table 16: **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.

Table 16: **ICAPE3 Pin Descriptions** *(cont'd)*

Pin	Type	Width	Description
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

The following table describes the describes the ICAPE3 primitive attributes.

Table 17: **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).

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 and the most common use case. 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 the control register CTL0[bit 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.

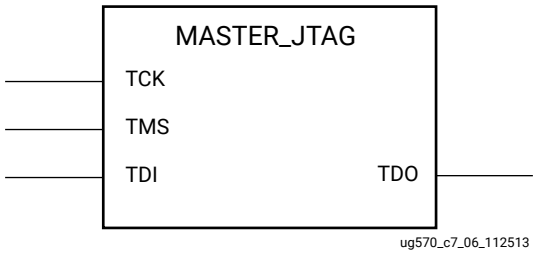
MASTER_JTAG

MASTER_JTAG provides control of the JTAG port from the FPGA logic, overriding the external pins. 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. The MASTER_JTAG primitive is not used for eFUSE programming in Spartan UltraScale+ FPGAs. Instead, Spartan UltraScale+ FPGAs rely on the AXI32 primitive for eFUSE programming. Because the external JTAG port is disabled, MASTER_JTAG prevents the use of the Vivado device programmer and the Vivado logic analyzer.

Primitive

The following figure shows the MASTER_JTAG primitive.

Figure 12: **MASTER_JTAG Primitive**



Pin Descriptions

The following table describes the MASTER_JTAG primitive pins.

Table 18: **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 primitive connects to the dedicated configuration pins (located in bank 0). The primitive allows post-configuration control of the SPI (x1, x2, or x4) flash device data pins `D[03:00]` and FCS_B. When the flash device is only used for configuration, the FPGA design does not require the STARTUPE3.

The SelectMAP configuration mode data pins `D[32:04]` and the OSPI configuration mode data pins OSPID[07:00] can be directly connected as part of the user design because they are multi-function bank configuration pins.

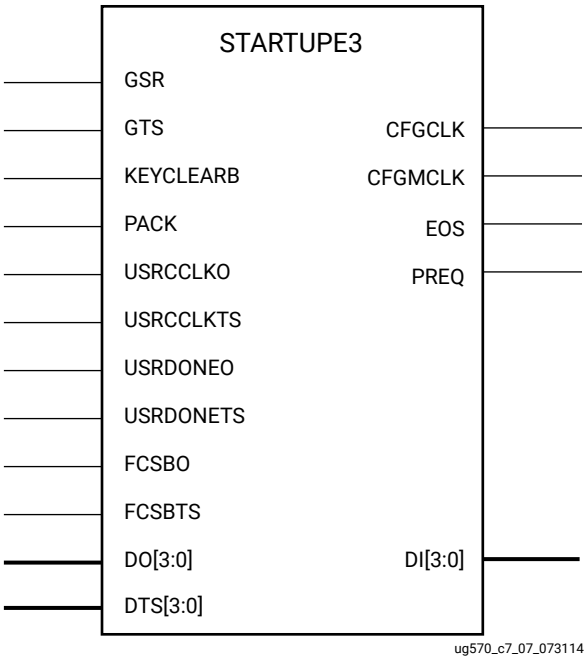
The STARTUPE3 is a superset of STARTUPE2, and designs are re-targeted automatically. The STARTUPE3 primitive for the UltraScale architecture-based FPGAs does not provide specification of the startup clock that was available in the STARTUPE2 for 7 series. Instead, it includes the FCS_B and separate input and output connections for the bidirectional `D[03:00]` pins.

The Spartan UltraScale+ FPGA STARTUPE3 primitive KEYCLEARB signal is used as the security use case tamper_fabric_b signal because this family does not have BBRAM.

Primitive

The following figure shows the STARTUPE3 primitive.

Figure 13: **STARTUPE3 Primitive**



Pin Descriptions

The following figure 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 19: **STARTUPE3 Pin Descriptions**

Pin	Type	Width	Description
CFGCLK	Output	1	Internal configuration control unit (CCU) clock output to the FPGA logic.
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. Refer to *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) for typical values and for the internal clock source core configuration tolerance.
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 start-up (EOS). 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	KEYCLEARB is used in Spartan UltraScale+ FPGAs as a security use case tamper_fabric_b. When the tamper_fabric_b is set Low it will cause the tamper response to be executed.
PACK	Input	1	PROGRAM_B pin or JPROG or IPROG request 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 or IPROG 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 might 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.

Table 19: **STARTUPE3 Pin Descriptions** *(cont'd)*

Pin	Type	Width	Description
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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930). The first three clock cycles on USRCCLKO after EOS 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	For Spartan UltraScale+ FPGAs the USR_CCLK_TS no longer tri-states the CCLK pin.
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

The following table describes the STARTUPE3 primitive attributes.

Table 20: **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.

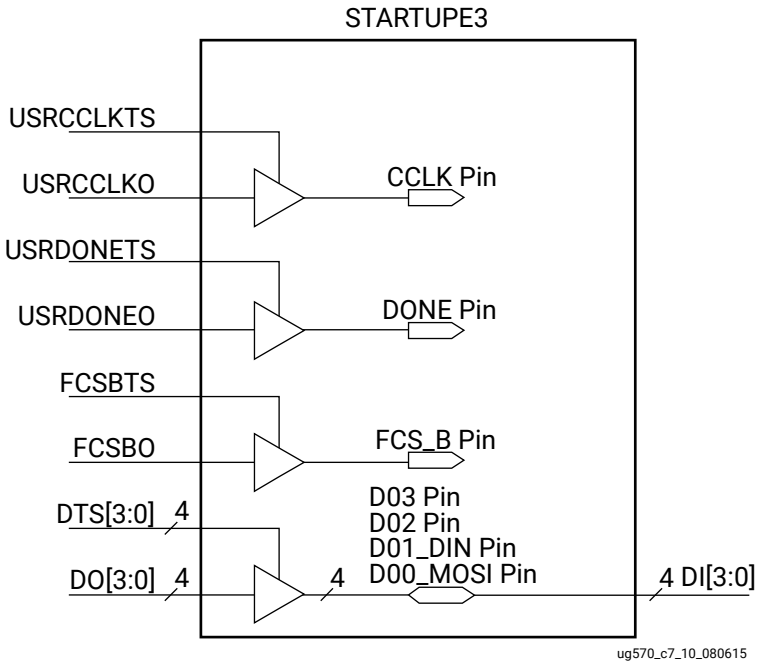
PROG_USR Attribute Description

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. An output from STARTUPE3 is the bidirectional pin input back into the user design (see the following figure).

Figure 14: **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.

The STARTUPE3 does not support input or output delay constraints. As a result, care should be taken to consider the performance requirements.

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, $T_{USRCCLKO}$.

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.

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 PDI. 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 PDI generation in Spartan UltraScale+ FPGAs. This avoids 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. 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 a PDI option, 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)
```

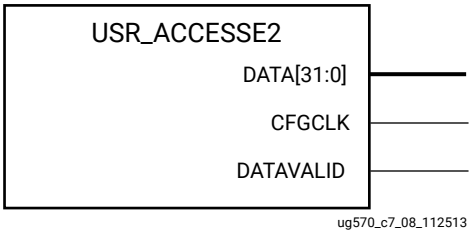
Where:

```
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
```

Primitive

The following figure shows the USR_ACCESSE2 primitive.

Figure 15: **USR_ACCESSE2 Primitive**



Pin Descriptions

The following table describes the USR_ACCESSE2 primitive pins.

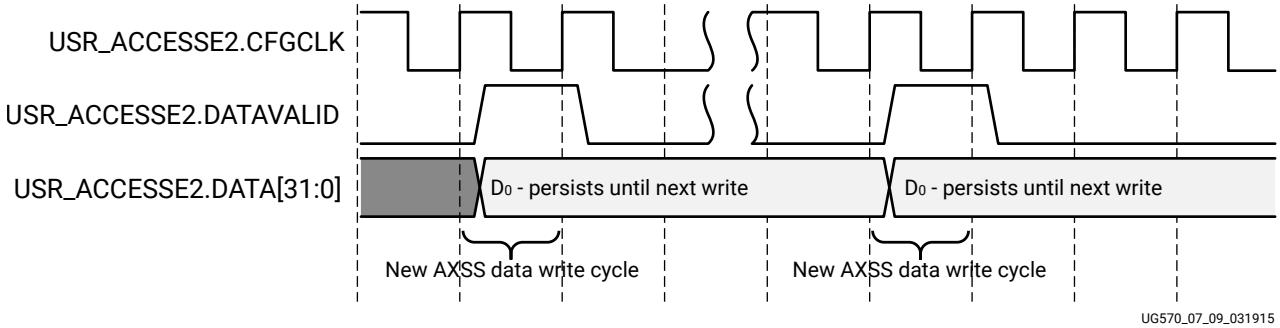
Table 21: **USR_ACCESSE2 Pin Descriptions**

Pin	Type	Width	Description
CFGCLK	Output	1	Internal configuration control unit (CCU) 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

Because USR_ACCESSE2 data is stored in the AXSS configuration register, the data can be dynamically updated through the JTAG or SelectMAP interface. For dynamic updates of the AXSS register, the waveforms from the USR_ACCESSE2 primitive that reflect the update events are shown in the following figure.

Figure 16: **USR_ACCESSE2 Update**



UltraScale+ devices add a special internal pin for the timing of the USR_ACCESSE2 output pins. The internal pin is not visible, but accessible through the Vivado get_pins command. To constrain USR_ACCESSE2 in UltraScale+, a clock needs to be defined on the internal clock pin. The clock definition triggers the proper timing arcs on the `DATA[31:0]` and `DATAVALID` interfaces. Pin descriptions are provided in the following table.

Table 22: **USR_ACCESSE2 Special UltraScale+ Device Internal Pin Descriptions**

USR_ACCESSE2 Pin	Internal Pin for Constraints	Description
CFGCLK	CCLK	Equivalent to the internal configuration control unit clock.
DATA[31:0]	CCLK	Rising-edge output register.
DATAVALID	CCLK	Rising-edge output register.

The following is an example of constraints for the special internal USR_ACCESSE2 timing pin, specifically an example for creating a primary clock on the internal pin. For a 200 MHz (5 ns), a 50% duty cycle `CCLK` clock constraint on the USR_ACCESSE2_inst.CCLK clock source pin covers the timing paths from `DATAVALID` or `DATA[31:0]` to the destination register clocked by USR_ACCESSE2.CFGCLK.

```
create_clock -period 5.000 -name CLOCK [get_pins USR_ACCESSE2_inst/CCLK]
```

Example constraints are shown if USR_ACCESSE2.DATAVALID and USR_ACCESSE2.DATA max delays are required. In some cases, the skew between USR_ACCESSE2 and the first level of registers can result in a difficult timing closure on those paths. The skew is introduced by the clock buffer connected between the pin USR_ACCESSE2.CFGCLK and the clock pins of the logic registers. When such a scenario occurs, it is possible to over-constrain the paths driven by USR_ACCESSE2 by changing the path requirement using a max delay constraint. The over-constraining mechanism forces the logic connected to USR_ACCESSE2 to be placed close to it. You should use this mechanism carefully because over-constraining too much can result in difficulty with timing closure.

In the following example, the path requirement is changed from a 5 ns (clock period defined on `CCLK`) to 3.5 ns:

```
set_max_delay -from [get_pins <USR_ACCESS>/DATAVALID] -to [get_pins
datavalid_reg/D] 3.5
set_max_delay -from [get_pins <USR_ACCESS>/DATA[*]] -to [get_pins
data_reg[*]/D] 3.5
```

Chapter 5

Configuration Engine

Spartan UltraScale+ FPGAs feature a dedicated platform management controller for handling device boot and configuration. This controller consists of four main components:

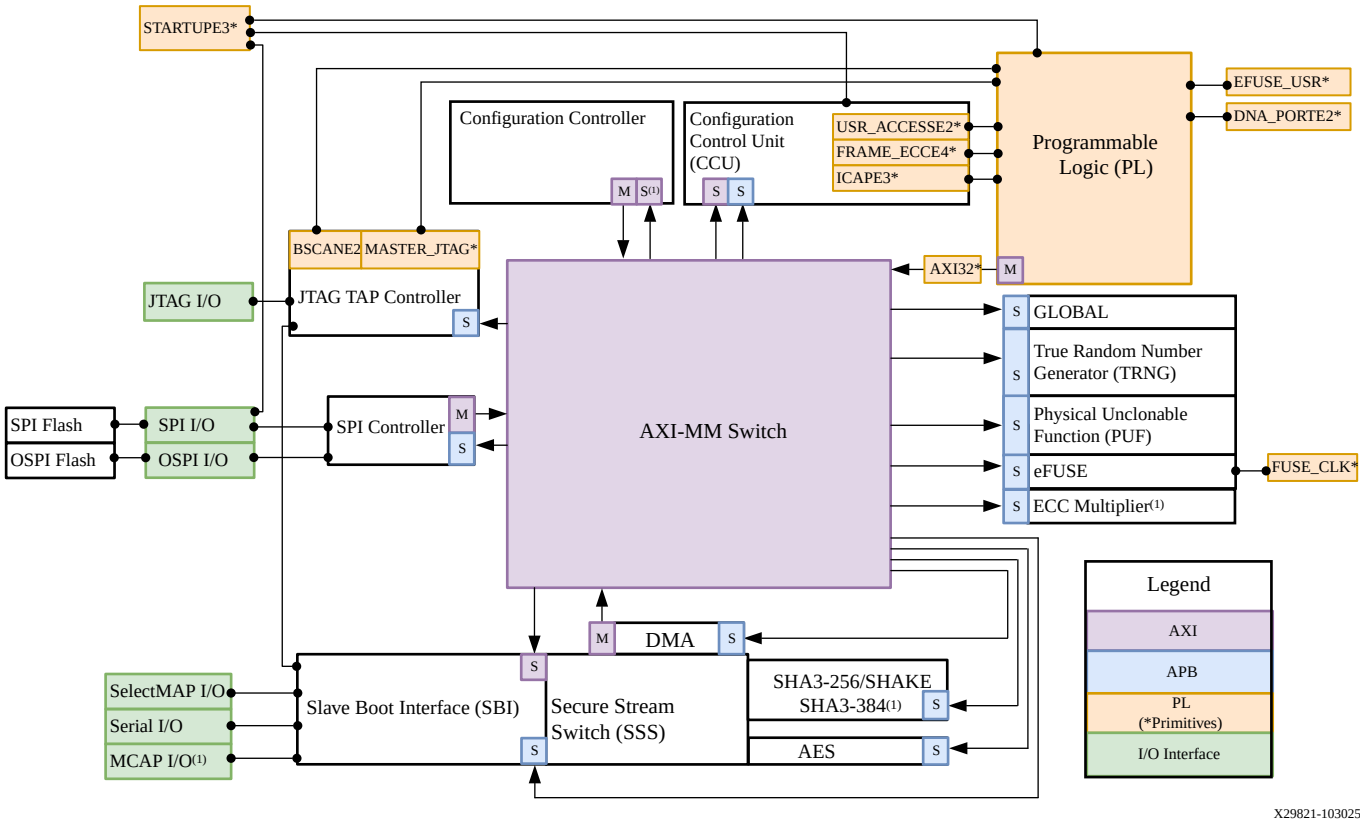
- Dedicated configuration controller.
- I/O interfaces for configuration mode protocols.
- Security component for cryptographic features.
- Configuration control unit (CCU) for managing programmable logic configuration data.

Spartan UltraScale+ FPGA Platform Management Controller Diagram

The platform management controller oversees security features and the configuration process of the Spartan UltraScale+ FPGA. It includes a configuration controller that runs BootROM firmware and the platform loader manager to configure the device after power-up. The following diagram illustrates connections from the controllers to the main AXI4 switch, pathways to the configuration control unit, configuration interfaces, and security features. Supported configuration protocols and bus widths include JTAG (x1), slave serial (x1), slave SelectMAP (x8, x16, x32), master SPI (x1, x2, x4), and master OSPI (x1, x8).

See Chapter 6: Configuration Interface and Pin Planning section for an overview of the interfaces and pins used in each configuration mode.

Figure 17: **Platform Management Controller Interconnect Diagram**



***Note*:**

1. Available only in the Spartan UltraScale+ SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P devices.

BootROM and Platform Loader Manager Overview

The Spartan UltraScale+ FPGA configuration controller runs a fixed implementation of both the BootROM firmware and the platform loader and manager (PLM). The BootROM responsibilities include:

- Read configuration mode and setup registers for selected mode.
- Search for signature in PDI boot header to confirm valid configuration mode connectivity.
- Load the PLM
- Initiate the PLM

The PLM responsibilities include:

- Set registers needed for the selected configuration mode and any user advanced settings.
- Load the PDI from the configuration source in both secure and non-secure modes.
- Process and complete the full PDI (PL .rcdo) configuration.
- Process and complete a partial PDI (PL .rcdo) configuration.

BootROM Initialization

At power-on the Spartan UltraScale+ FPGA internal $F_{OSC\_CORE\_CLK}$ runs, and for all configuration modes, the BootROM initializes the following control clocks:

- Internal PMC control clock frequency ($F_{PMC\_IRO\_CLK}$) is set to 255 MHz.
- Internal configuration control unit (CCU) clock frequency ($F_{CCU\_IRO\_CLK}$) is set to 127 MHz.

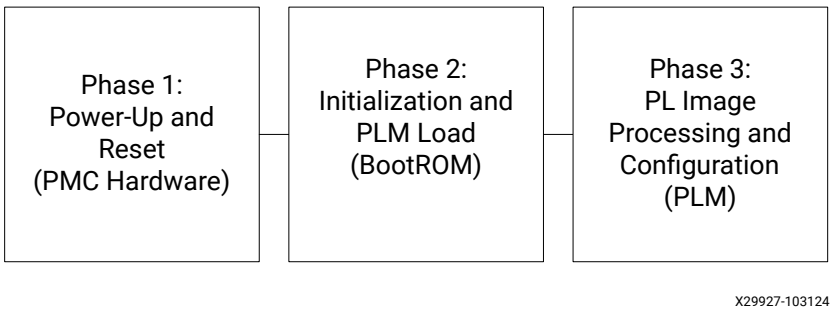
For Master SPI (SPI_24, SPI_32) or OSPI configuration modes, the configuration clock (CCLK) is set to 21 MHz.

***Note*:** The PLM does not modify the BootROM initialized PMC or CCU clock settings during configuration. The PLM only modifies the CCLK if the user design has selected a change with the Spartan UltraScale+ PDI properties. See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for the PDI properties.

Configuration Sequence

The platform management controller (PMC) supports security features and configuration. While each of the configuration mode interfaces differ, the basic steps for configuring a device is the same for all configuration modes. See the following figure for the configuration sequence startup phases.

Figure 18: **Configuration Sequence**



Phase 1: Platform Management Controller Hardware (Power-Up and Reset)

In phase 1, the PMC hardware detects the power is valid and the `PROGRAM_B` pin is released to initiate the configuration sequence. After power is applied to the device, the configuration hardware performs a series of tasks to capture the mode selection at the pins, initialize to a known secure state, and all registers in the PMC are zeroized (reset + verification of reset state) if enabled through eFUSEs. Before execution of the BootROM, the dedicated hardware hashes the immutable BootROM code using a dedicated SHA-3/384 engine and compares the calculated cryptographic hash against a golden copy stored in the device. If the hashes match, the integrity of the BootROM is validated, and the BootROM is enabled to execute.

Phase 2: BootROM (Initialization and Platform Loader and Manager)

In phase 2, the BootROM performs basic integrity checks, performs initialization of the programmable logic, and reads the mode register to start the configuration setup. The BootROM searches for the identification signature XLNX (`0x584C4E58`). After the identification signature is found, the boot header and hash block are read into the device. If authentication is enabled, the hash block is authenticated. The boot header checksum is validated against the checksum in the hash block. Next the PLM is loaded and it's checksum is validated against its checksum value in the hash block. If the PLM is encrypted, then it is decrypted before the bootROM hands off the process to PLM. The BootROM code enforces the secure boot modes (A-HWRoT and/or S-HWRoT) if enabled.

If a boot identification signature is not found at address `0x00000000`, then master modes conduct a fallback by default and increment to the next 32KB address in memory. At the new address, the BootROM will search again for the signature. This process continues until the search limit for the configuration mode is met. See Table 6 for more information.

Phase 3: Platform Loader and Manager (PL Image Processing and Configuration)

In phase 3, the PLM is responsible to complete the PDI image processing and load as well as ensure the PL starts up and asserts DONE. Phase 3 includes the PL startup and these steps:

- Register configuration: Sets the registers according to the selected configuration mode and applies any user-defined advanced settings (i.e., `CCLK` frequency increase or EMCCLK use).
- PDI load: Loads the PDI from the configuration source in both secure and non-secure modes.
- Full PDI configuration: Processes the full PDI (.rcdo) configuration, including necessary setup and initialization.
- Partial PDI configuration: Supports partial PDI (.rcdo) configurations, allowing for flexible device setup.

Start-up Sequence

At the end of Phase 3, after the PDI contents have been loaded to the programmable logic (PL) configuration frames the device enters a start-up sequence. The start-up sequence has steps (0-7), shown in Figure 19. The start-up sequence is on the CCU clock domain and performs the tasks outlined in the following table:

Table 23: **User-Selectable Cycle of Start-up Events**

Step	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 PDI options controlled by the BITSTREAM.STARTUP properties (refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908). For most applications the default start-up event sequence is recommended:

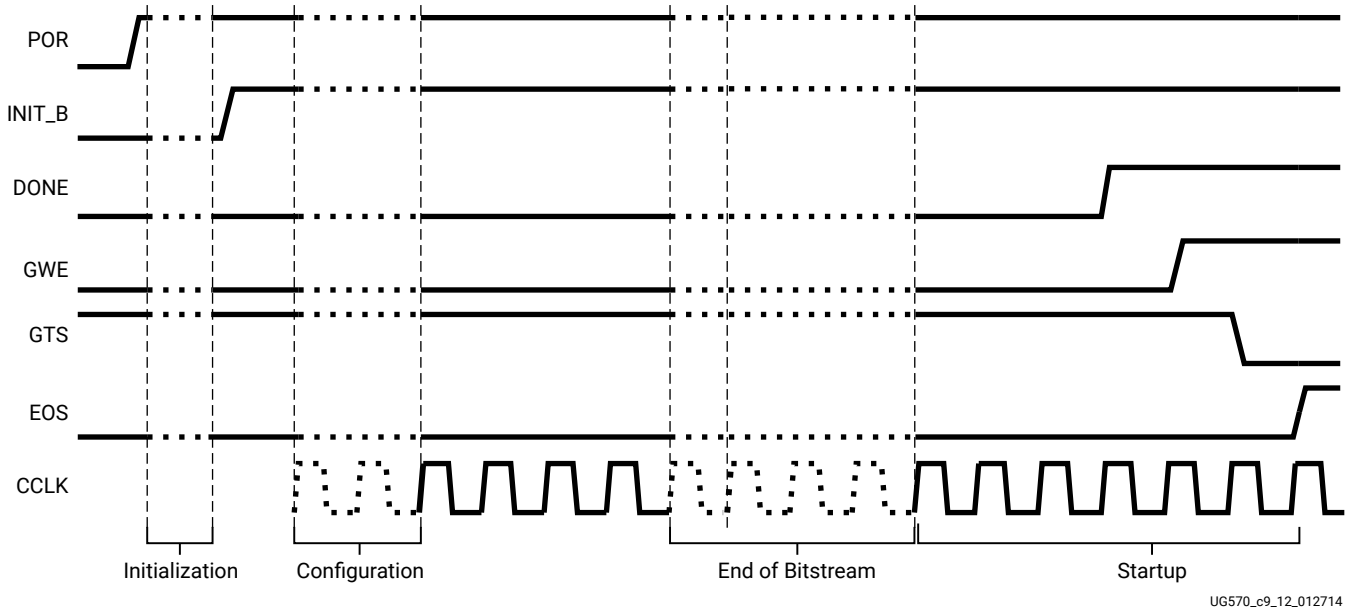
Table 24: **Default Sequence of Start-Up Events**

Step	Event
4	Release DONE pin
5	Negate Global 3-State (GTS), activating I/O
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 the DCI to match with the appropriate PDI options. These options are typically set to prevent the DONE, GTS, and GWE from being asserted before the MMCMs have locked and/or DCI has matched.

The start-up sequence steps are part of the Phase 3 which includes steps in the configuration through the start-up shown in Figure:

Figure 19: **Configuration Signal Sequencing (Default Start-Up Settings)**



Platform Management Controller Clocking Diagram

The Spartan UltraScale+ FPGAs configuration engine platform management controller (PMC) has different clock paths to support the new configuration modes and security features.

- The OSC_CORE_CLK is the internal oscillator clock that the device powers up from and that drives the Configuration Control Unit (CCU) clock CCU_IRO_CLK and the PMC dedicated configuration controller clock PMC_IRO_CLK.
- The eFUSE programming clock EFUSE_CLK source options are shown, programmable logic (PL) FUSE_CLK or the External Master Configuration Clock (`EMCCLK`) pin.
- The OSPI controller clock SPI_CLK source options to the OSPI controller clock are shown. The SPI_CLK drives the CCLK output to the Master SPI (SPI_24, SPI_32) and Master OSPI configuration modes.

The following figure provides the configuration and security feature clock sources for the Spartan UltraScale+ family.

Figure 20: **PMC Clocking Diagram**

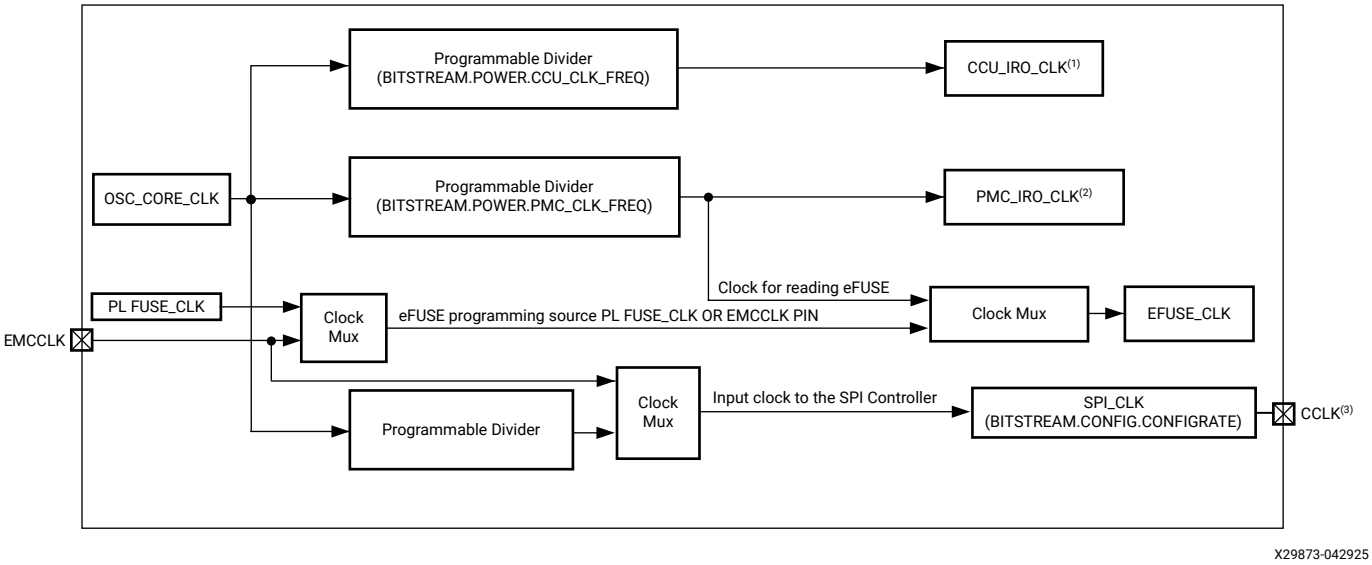


Figure Notes:

1. The CCU_IRO_CLK frequency cannot be changed by the user during configuration. The BITSTREAM.POWER.CCU_CLK_FREQ option is for post-configuration advanced usage only.
2. The PMC_IRO_CLK frequency cannot be changed by the user during configuration. The BITSTREAM.POWER.PMC_CLK_FREQ option is for post-configuration advanced usage only.
3. For SPI/OSPI configuration modes.

Chapter 6

Configuration Interface and Pin Planning

The configuration mode(s) used in an application can affect the planning of the design pinout. 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. See the Configuration Interfaces for more information.
2. Determine the set of pins and the bank locations for both the primary and secondary configuration modes planned. See Configuration Pins for more information.
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

AMD Spartan™ UltraScale+™ FPGAs have five configuration interfaces. The configuration interface is selected with the configuration mode pins `M[2:0]`.

The mode input pins M0, M1, and M2 pins must be set at a constant DC voltage level, either through a pull-up or pull-down resistor (<1 kΩ), or tied directly to ground or VCCO_0. For configuration initiated via assertion of the `PROGRAM_B` pin, the `M[2:0]` input values must be set before the deassertion of `PROGRAM_B`. The `M[2:0]` input value must be held at least throughout device configuration initialization, which is the period when the device is driving `INIT_B` Low.

For detailed configuration interface timing information, see the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930). Each configuration interface corresponds to one or more configuration modes and bus widths, shown in the following table.

Table 25: **Configuration Modes**

Configuration Mode	M[2:0]	Bus Width	Clock Direction
JTAG(1)	101	x1	Input (TCK)
Master SPI_24(2)	001	x1, x2, x4	Output (CCLK)
Master SPI_32(2)	010	x1, x2, x4	Output (CCLK)
Master OSPI	011	x1, x8	Output (CCLK)
Slave SelectMAP	110	x8, x16, x32	Input (CCLK)
Slave Serial	111	x1	Input (CCLK)

Notes:

1. It is recommended to implement a board-level switch or jumper that allows a board to switch from the target configuration mode to the JTAG configuration mode if JTAG is not the primary mode. Devices should be set to JTAG configuration mode during board-level JTAG boundary-scan testing and the mode can be used for loading prototype designs.
2. Spartan UltraScale+ FPGAs support a master SPI 24-bit addressing configuration mode (SPI_24) or a SPI 32-bit addressing configuration mode (SPI_32). The SPI_32 configuration mode option sends 32-bit addressing commands that addresses flash sizes greater than 128 Mb. See Configuration and Mode Setting for more information.

Configuration Pins

Each configuration mode has a corresponding set of interface pins that span up to three banks on the FPGA. Bank 0 contains the dedicated configuration pins and is always part of every configuration interface. Multi-function configuration banks (Bank 65, Bank66) include select configuration interface pins. If the persist option is used, the multi-function I/O for the selected configuration mode remain active after configuration. The following tables show the configuration pins and their locations across the I/O banks. See Differences from Previous Generations for more information.

Table 26: **Master Modes Configuration Pins**

Pin Name	Bank(6)	Master SPI_24 or SPI_32			Master OSPI	
		x1	x2	x4	x1	x8
POR_OVERRIDE	N/A	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE
M[2:0]	0	SPI_24: M[2:0]=001 SPI_32: M[2:0]=010	SPI_24: M[2:0]=001 SPI_32: M[2:0]=010	SPI_24: M[2:0]=001 SPI_32: M[2:0]=010	M[2:0]=011	M[2:0]=011
TCK	0	TCK	TCK	TCK	TCK	TCK
TMS	0	TMS	TMS	TMS	TMS	TMS
TDI	0	TDI	TDI	TDI	TDI	TDI
TDO	0	TDO	TDO	TDO	TDO	TDO
PROGRAM_B	0	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B

Table 26: **Master Modes Configuration Pins** *(cont'd)*

Pin Name	Bank[6]	Master SPI_24 or SPI_32			Master OSPI	
		x1	x2	x4	x1	x8
INIT_B	0	INIT_B	INIT_B	INIT_B	INIT_B	INIT_B
DONE	0	DONE	DONE	DONE	DONE	DONE
CCLK	0	CCLK	CCLK	CCLK	CCLK	CCLK
PUDC_B [1]	0	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]
RDWR_FCS_B	0	FCS_B	FCS_B	FCS_B	FCS_B	FCS_B
D00_MOSI_DOUT	0	MOSI	D00_MOSI	D00_MOSI	-	-
D01_DIN	0	DIN	D01_DIN	D01_DIN	-	-
D02	0	-	-	D02	-	-
D03	0	-	-	D03	-	-
D[07:04]_OSPID[03:00]	65 or 66	-	-	-	OSPID[01:00]	OSPID[03:00]
D[11:08]_OSPID[07:04]	65 or 66	-	-	-	-	OSPID[07:04]
OSPI_ECC_FAIL	65 or 66	-	-	-	OSPI_ECC_FAIL	OSPI_ECC_FAIL
OSPI_RST_B	65 or 66	-	-	-	OSPI_RST_B	OSPI_RST_B
OSPI_DS	65 or 66	-	-	-	-	OSPI_DS
EMCCLK[2]	65 or 66	EMCCLK[2]	EMCCLK[2]	EMCCLK[2]	EMCCLK[2]	EMCCLK[2]
FCS1_B[3()]	65 or 66	-	-	-	-	-

Notes:

1. `PUDC_B` has special functionality during configuration but is independent of all configuration interfaces.
2. `EMCCLK` is used when the external master `CCLK` enable option enables `EMCCLK` as an input for clocking the master configuration modes.
3. `FCS1_B` is only supported during post-configuration access of a dual-stacked SPI x4 flash.
4. JTAG pins (`TCK`, `TMS`, `TDI`, `TDO`) are listed with all configuration modes as the JTAG interface is recommended to be included for debug.
5. Dashes indicate that the pin is not used in the configuration mode and is high impedance and ignored during configuration.
6. Bank 65 is used by devices: SU55P, SU65P, SU100P, SU150P, and SU200P.
   Bank 66 is used by devices: SU10P, SU25P, SU35P, SU45P,and SU60P.

Table 27: **Slave Modes Configuration Pins**

Pin Name	Bank [5]	JTAG	Slave Serial	Slave SelectMAP		
		x1	x1	x8	x16	x32
POR_OVERRIDE	N/A	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE	POR_ OVERRIDE
M[2:0]	0	M[2:0]=101	M[2:0]=111	M[2:0]=110	M[2:0]=110	M[2:0]=110
TCK	0	TCK	TCK	TCK	TCK	TCK
TMS	0	TMS	TMS	TMS	TMS	TMS
TDI	0	TDI	TDI	TDI	TDI	TDI
TDO	0	TDO	TDO	TDO	TDO	TDO
PROGRAM_B	0	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B	PROGRAM_B
INIT_B	0	INIT_B	INIT_B	INIT_B	INIT_B	INIT_B
DONE	0	DONE	DONE	DONE	DONE	DONE
CCLK	0	-	CCLK	CCLK	CCLK	CCLK
PUDC_B[1]	0	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]	PUDC_B[1]
RDWR_FCS_B	0	-	-	RDWR_B	RDWR_B	RDWR_B
D00_MOSI_DOUT[2]	0	-	DOUT[2]	D00	D00	D00

Table 27: **Slave Modes Configuration Pins** *(cont'd)*

Pin Name	Bank (5)	JTAG	Slave Serial	Slave SelectMAP		
		x1	x1	x8	x16	x32
D01_DIN	0	-	DIN	D01	D01	D01
D02_CS_B	0	-	CS_B	D02	D02	D02
D03_READY	0	-	READY	D03	D03	D03
D[07:04]_OSPID[03:00]	65 or 66	-	-	D[07:04]	D[07:04]	D[07:04]
D[11:08]_OSPID[07:04]	65 or 66	-	-	-	D[11:08]	D[11:08]
D[31:12]	65	-	-	-	D[15:12]	D[31:12]
CSI_ B	65 or 66	-	-	CSI_B	CSI_B	CSI_B
BUSY	65 or 66	-	-	BUSY	BUSY	BUSY

Notes:

1. PUDC_B has special functionality during configuration but is independent of all configuration interfaces.
2. DOUT is only used in a slave serial configuration daisy-chain for outputting data to the downstream. Otherwise, DOUT is high impedance.
3. JTAG pins (TCK, TMS, TDI, TDO) are listed with all configuration modes as the JTAG interface is recommended to be included for debug.
4. Dashes indicate that the pin is not used in the configuration mode and is high-impedance and ignored during configuration.
5. Multi-function configuration pins BUSY, CSI_B and D[11:04]are in Bank 66 for devices: SU10P, SU25P, SU35P, SU45P, and SU60P. and in Bank 65 for devices: SU55P, SU65P, SU100P, SU150P, and SU200P

Configuration Pins Definitions

The definition of each configuration pin is summarized in the following table.

Table 28: **Configuration Pin Definitions**

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930). 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}.*
M[2:0]	0	Dedicated	Input	Configuration Mode	All	≤ 1 kΩ	Determine the configuration mode. See Configuration Interfaces 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 AMD 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 (for example, 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 AMD 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 (for example, 10 kΩ) pull-up resistor to maintain a valid High when no cable is connected.
					Unused	N/A	Ignored and can be left unconnected.

Table 28: **Configuration Pin Definitions** *(cont'd)*

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	Recommended External Pull-Up/Pull-Down	Description
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 AMD 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. If the TCK signal is buffered, connect the buffer input to an external weak (for example, 10 kΩ) pull-up resistor to maintain a valid High when no cable is connected.
					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 AMD 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 (that is, 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. If PROGRAM_B is held Low, JTAG operations can be restricted. 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 28: **Configuration Pin Definitions** *(cont'd)*

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	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. The DONE output is open-drain. ***Note***: DONE has a default internal pull-up resistor of approximately 10 kΩ. Connect DONE to a 4.7 kΩ pull-up resistor to VCCO_0 to ensure clean Low-to-High transitions. 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.)

Table 28: **Configuration Pin Definitions** *(cont'd)*

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	Recommended External Pull-Up/Pull-Down	Description
CCLK	0	Dedicated	Input or Output	Configuration Clock	SPI, OSPI	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.
					Serial, SelectMAP	N/A	An input and requires connection to an external clock source.
					JTAG	N/A	High-impedance, and can be left unconnected.
PUDC_B	0	Dedicated	Input	Pull-Up During Configuration (bar)	All	≤ 1 kΩ	Active-Low input enables internal pull-up resistors on the SelectIOpins after power-up and during configuration, including multi-function configuration pins when not used for the selected configuration mode. When PUDC_B is Low, internal pull-up resistors are enabled on each SelectIOpin. When PUDC_B is High, internal pull-up resistors are disabled on each SelectIOpin. 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	Read/Write input controls whether the data pins are inputs or outputs. When RDWR_B is Low, an external device drives the SelectMAP data bus. Write operation solution support is available configuration. When RDWR_B is High, the Versal device drives the SelectMAP data bus, regardless of the CSI_B level. Read operation (SelectMAP readback) solution support is not available.
					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).
					Serial, JTAG	N/A	High-impedance and ignored, and can be left unconnected.

Table 28: **Configuration Pin Definitions** *(cont'd)*

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	Recommended External Pull-Up/Pull-Down	Description
D00_MOSI_DOUT	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 (D00_MOSI)	N/A	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 (D00)	N/A	D00 data input pin. See D[31:00] row in this table.
					Serial (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.
					JTAG, OSPI	N/A	High-impedance and ignored, and can be left unconnected.
D01_DIN	0	Dedicated	Input or Bidirectional	Data Bit 1 or Data Input	SelectMAP (D01)	N/A	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).
					OSPI, JTAG	N/A	Ignored, and can be left unconnected.
D02_CS_B	0	Dedicated	Input or Bidirectional	Data Bit 2	SPI x4 (D02)	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}.
					Serial (CS_B)	4.7 kΩ	Active-Low chip select input that enables the slave serial interface
					SelectMAP (D02)	N/A	D02 data input pin. See D[31:00] row in this table.
					SPI x1/x2, OSPI, JTAG	N/A	Ignored, and can be left unconnected.

Table 28: **Configuration Pin Definitions** *(cont'd)*

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	Recommended External Pull-Up/Pull-Down	Description
D03_READY	0	Dedicated	Input or Bidirectional	Data Bit 3	SPI x4 (D03)	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}.
					Serial (READY)	N/A	READY is an open-drain output that must be connected to the host and monitored. READY requires an external 4.7 kΩ pull-up resistor. When the pin goes Low there are 24 CCLK cycles before the data buffer will overflow. When READY=0 the CS_B pin must be deasserted (CS_B=1) within 24 CCLK cycles to pause the data load until READY=1, ensuring a successful data load.
					SelectMAP (D03)	N/A	D03 data input pin. See D[31:00] row in this table.
					SPI x1/x2, OSPI JTAG	N/A	Ignored, and can be left unconnected.
EMCCLK	65 or 66	Multi-function	Input	External Master Configuration Clock	SPI, OSPI	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.
					Serial, SelectMAP, JTAG, or Unused	N/A	Ignored and can be left unconnected, or connected as an I/O pin after configuration.
					eFUSE Programming	N/A	For Spartan UltraScale+ FPGAs the EMCCLK can be a clock source for eFUSE programming.
CSI_ B	65 or 66	Multi-function	Input or Output	Chip Select Input (bar)	SelectMAP	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.
					Serial, SPI, OSPI, JTAG	N/A	Ignored and high-impedance, and can be left unconnected, or connected as an I/O pin after configuration.
BUSY	65 or 66	Multi-function	Output	Busy	SelectMAP	NA	BUSY is an active-High output that must be connected to the host and monitored. When BUSY is High when there are 24 CCLK cycles left before the data buffer will overflow. When BUSY=1 the CSI_B must be deasserted (CSI_B=1) within 24 CCLK cycles to pause the data load until BUSY=0, ensuring a successful data load.

Table 28: **Configuration Pin Definitions** *(cont'd)*

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	Recommended External Pull-Up/Pull-Down	Description
OSPI_DQS	65 or 66	Multi-function	Input	Data Strobe	OSPI	N/A	Data Strobe used in OSPI mode for DDR mode
					Serial, SPI, JTAG, SelectMAP	N/A	Ignored and high-impedance, and can be left unconnected, or connected as an I/O pin after configuration
OSPI_ECC_FAIL	65 or 66	Multi-function	Input	ECC Failure	OSPI	N/A	ECC Failure input status available from select OSPI flash devices. OSPI_ECC_FAIL input is ignored by default unless it is enabled in the PDI with a property.
					Serial, SPI, JTAG, SelectMAP	N/A	Ignored and high-impedance, and can be left unconnected, or connected as an I/O pin after configuration
OSPI_RST_B	65 or 66	Multi-function	Output	Reset	OSPI	N/A	Reset (active low) signal used in OSPI mode to reset the OSPI flash device
					Serial, SPI, JTAG, SelectMAP	N/A	Ignored and high-impedance, and can be left unconnected, or connected as an I/O pin after configuration
FCS1_B	65 or 66	Multi-function	Output	Chip Select for stacked dual flash (bar)	SPI	2.4 kΩ	Active-Low chip select output that enables dual stacked flash devices for post-configuration access. 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). FCS1_B input is ignored by default unless the feature is enabled in the PDI with a property.
					Serial, OSPI, JTAG, SelectMAP	N/A	Ignored and high-impedance, and can be left unconnected, or connected as an I/O pin after configuration
D[11:04]_OSPID[07:00]	65 or 66	Multi-function	Input or Bidirectional	Data Bus	SelectMAP x8/x16/x32 (D[11:04])	N/A	Data Bus bits 11 to 4. A subset or all of the D[31:00] pins are the databus interface for SelectMAP 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. See the D00-D03 rows in this table.
					OSPI (OSPID[07:00])	N/A	OSPID[07:00] pins are the data bus interface for the OSPI mode.
					SPI, JTAG	N/A	All data pins are unused, ignored, and high impedance during configuration

Table 28: **Configuration Pin Definitions** *(cont'd)*

Pin Name	Bank[1]	Type	Direction	Function	Mode/ Feature	Recommended External Pull-Up/Pull-Down	Description
D[31:12]	65	Multi-function	Input or Output	Data Bus	SelectMAP x16/x32	N/A	Data Bus bits 31 to 12 - are used in SelectMAP x32. Data Bus bits 15 to 12 are used in SelectMAP x16. See D[11:04] row and the D00-D03 rows in this table.
					Serial, SPI, OSPI, SelectMAP x8, JTAG	N/A	Ignored, and can be left unconnected, or connected as I/O after configuration.

Notes:

1. The multi-function I/O bank used is dependent on the Spartan UltraScale+ device. See the specific configuration mode chapter or *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575) for details.

Chapter 7

Master SPI Configuration Mode

Introduction

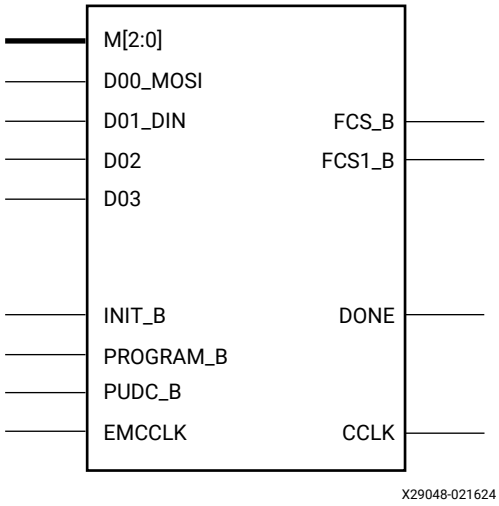
The master SPI configuration mode in AMD Spartan™ UltraScale+™ FPGAs enables the use of low pin count, industry-standard SPI NOR flash devices for programmable device image (PDI) storage. The FPGA supports a direct connection to the SPI interface of a NOR flash device for reading a stored PDI. See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for supported flash with each FPGA.

SPI Interface

The master SPI configuration mode supports multiple data bus widths and setups. The master SPI configuration mode can read from standard 1-bit (x1), 2-bit (x2), and 4-bit (x4) SPI flash devices. Spartan UltraScale+ FPGAs internal configuration logic automatically detects and selects the widest bus-width available from the user setup.

The master SPI 2-bit and 4-bit configuration modes are proportionally faster than the standard 1-bit SPI interface. The master SPI configuration interface is represented in the following figure.

Figure 21: **SPI Configuration Interface**



Spartan UltraScale+ FPGAs support a master SPI 24-bit addressing configuration mode (SPI_24) or a 32-bit addressing configuration mode (SPI_32). The SPI_32 configuration mode option addresses flash sizes greater than 128 Mb.

Spartan UltraScale+ FPGAs also provide a 8-bit (x8) master OSPI configuration mode. See Chapter 8: Master OSPI Configuration Mode for details.

I/O Configuration Detection

The internal configuration logic can detect the intended I/O width of the QSPI interface using the width detection parameter value (0x665599AA). During the SPI configuration mode process, the internal configuration logic firmware configures the controller with 4-bit I/O. Then the SPI device is read in 1-bit mode. It reads the width detection parameter in the PDI header. If the width detection parameter is equal to 0x665599AA, then it is assumed a valid header was found requesting a 4-bit I/O configuration. After reading the width detection parameter in 1-bit mode, the configuration firmware attempts to read the parameter in 4-bit mode. If 4-bit mode fails, it tries 2-bit mode bus width to attempt access to the QSPI device. The Spartan UltraScale+ FPGA internal configuration logic always attempts to use the widest width working on a setup. If a valid header is not detected then the BootROM increments the MultiBoot register address offset by 32 KB and attempts to locate the valid header. This is repeated until the search limit.

SPI Configuration Read Commands

The master SPI configuration mode supports the SPI flash device read operations listed in the following table.

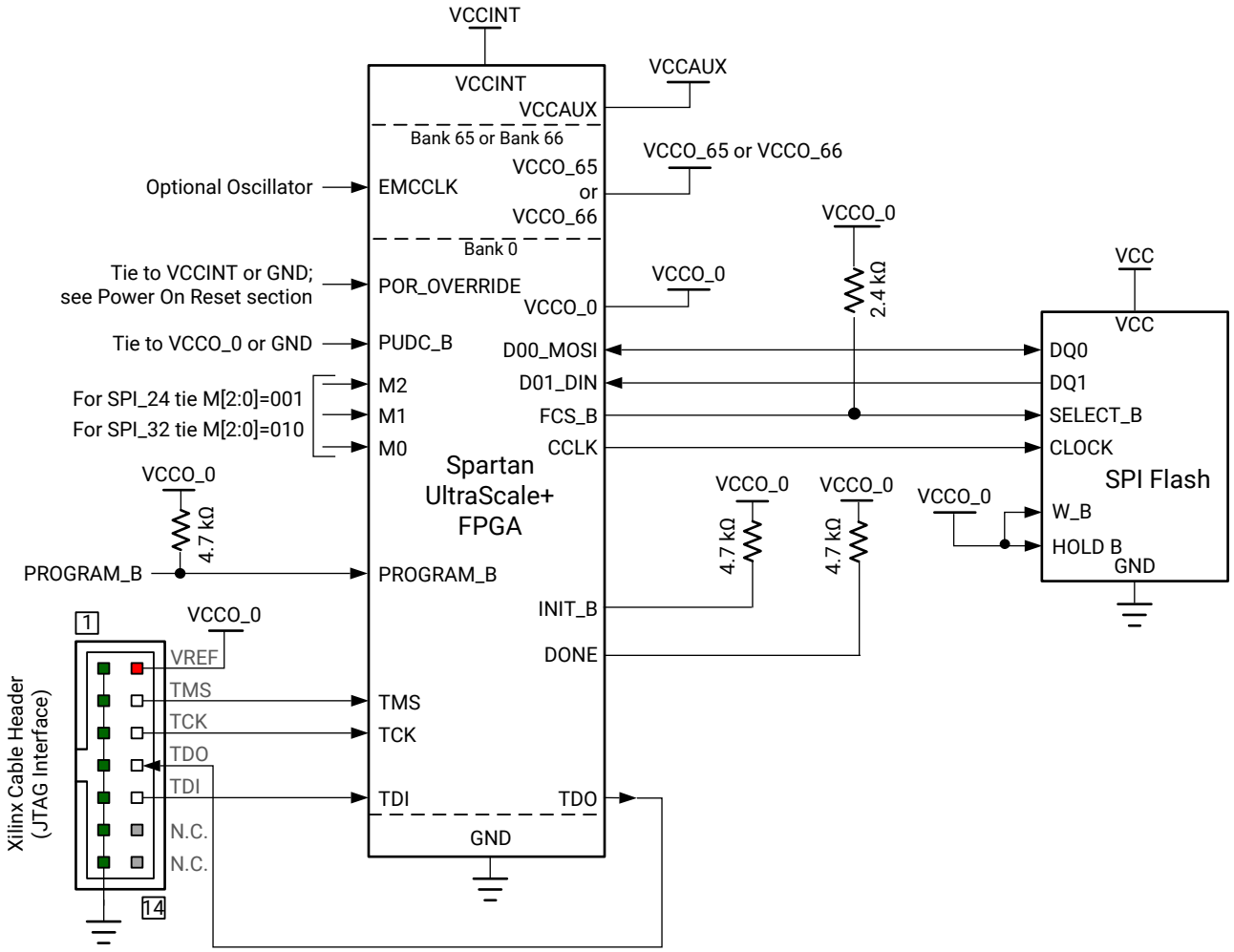
Table 29: **SPI Configuration Read Commands**

Configuration Mode	Data Width	Read Mode	Command Code (hex)	Dummy Cycles
SPI_24	1	Normal Read (3-Byte or 24-bit addressing)	03	-
SPI_24	2	Dual Output Fast Read (3-Byte or 24-bit addressing)	3B	8
SPI_24	4	Quad Output Fast Read (3-Byte or 24-bit addressing)	6B	8
SPI_32	1	Normal Read (4-Byte or 32-bit addressing)	13	-
SPI_32	2	Dual Output Fast Read (4-Byte or 32-bit addressing)	3C	8
SPI_32	4	Quad Output Fast Read (4-Byte or 32-bit addressing)	6C	8

SPI x1/x2 Configuration Interface Example

The following figure shows the connections for a SPI configuration with a 1-bit or 2-bit data width. These connections are the same because the 2-bit mode uses the `D[00]` pin as a 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. The FPGA pin connections to the SPI NOR flash involved in the master SPI mode are listed in Table 26 under Configuration Pins.

Figure 22: **SPI x1/x2 Configuration Interface Example**



Refer to the Notes following this figure for related information.

X29994-102924

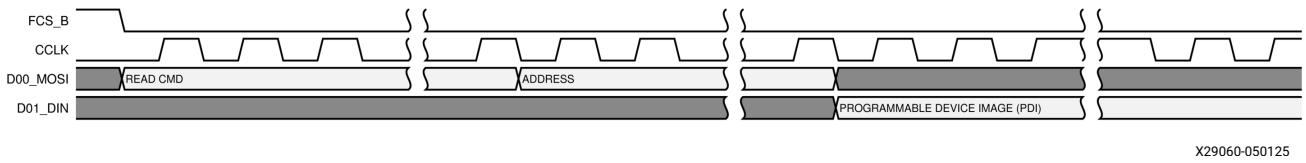
1. The DONE pin is by default an open-drain output. An external pull-up resistor is required. See Table 28 for DONE signal details.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required. See Table 28 for INIT_B signal details.
3. CCLK signal integrity is critical.
4. A series resistor should be considered for the data path from the flash to the FPGA to minimize overshoot. The proper resistor value can be determined from simulation.
5. The FPGA V_{CCO_0} supply must be compatible with the VCC for the I/O of the flash device.

6. 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 28 for `PUDC_B` signal details.
7. Dependent on the Spartan UltraScale+ FPGA, Bank 65 or Bank 66 is used for the multi-function configuration I/O. See the Configuration Pins section for details.

SPI x1 Mode Sequence

The SPI x1 mode allows the Spartan UltraScale+ FPGA to be loaded from an external SPI flash device. The sequence is shown in the following figure.

Figure 23: **SPI x1 Mode Sequence**

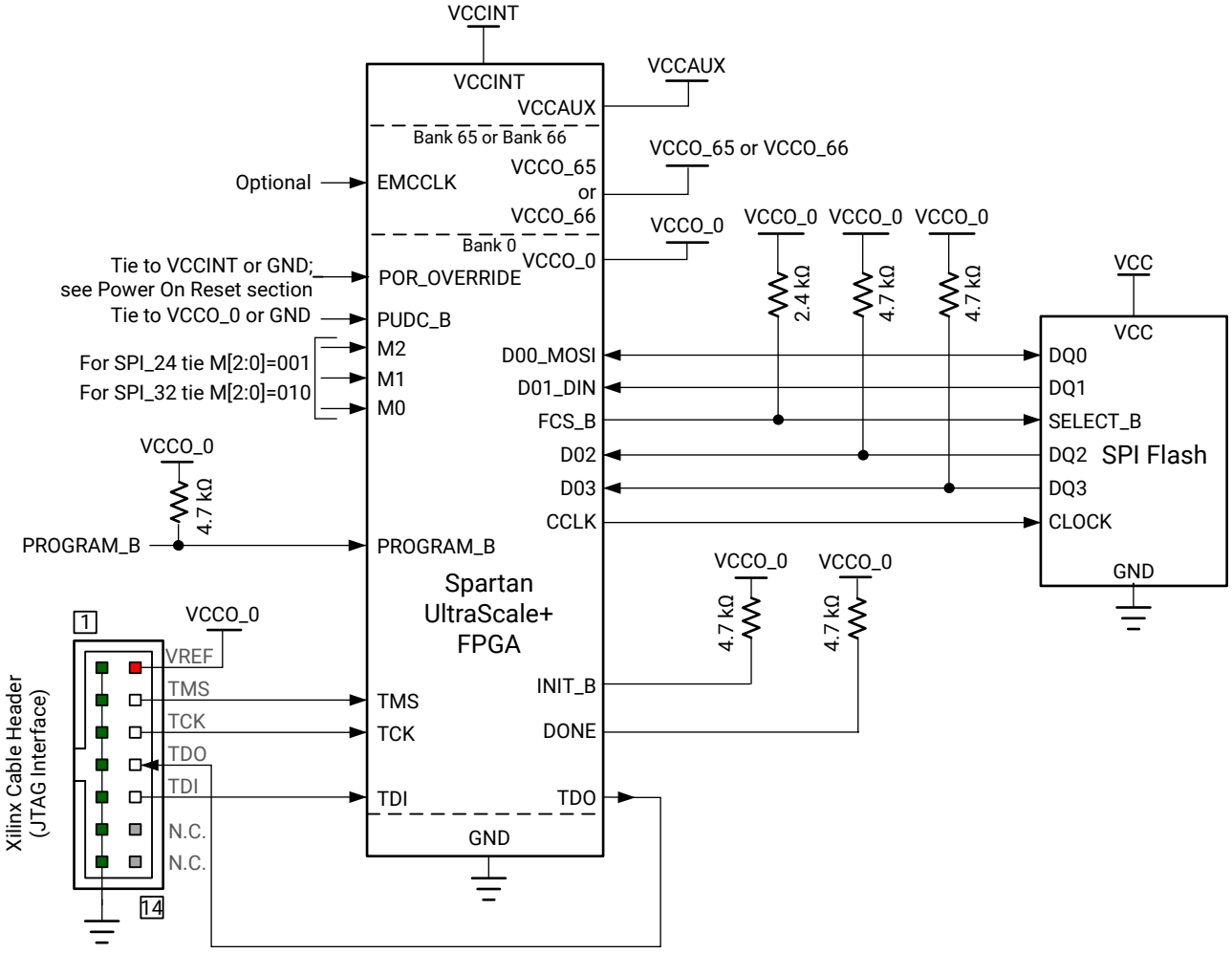


***Note*:** Waveforms represent the relative sequence of events and are not to scale. See the flash memory data sheet for detailed SPI command and data timing.

SPI x4 Configuration Interface Example

UltraScale FPGAs support a 4-bit (x4) quad SPI master configuration width as shown in the following figure.

Figure 24: **SPI x4 Configuration Interface Example**



Refer to the Notes following this figure for related information.

X29050-102924

1. The `DONE` pin is by default an open-drain output. An external pull-up resistor is required.
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. 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.
5. The FPGA V_{CCO_0} supply must be compatible with the V_{CC} for the I/O of the flash device.
6. The FPGA `PUDC_B` pin is tied to GND to enable internal pull-ups or it can be tied to VCCO_0 to 3-state the SelectIO pins after power-up and during configuration.

7. Dependent on the Spartan UltraScale+ FPGA, Bank 65 or Bank 66 is used for the multi-function configuration I/O.

Note: See Table 26 under Configuration Pins for signal details.

SPI NOR Flash Densities over 128 Mb

SPI 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 `03h` command. These methods should not be enabled for the flash devices used to configure Spartan UltraScale+ FPGAs.

The solution supported by the Spartan UltraScale+ FPGAs requires the flash to configure from a 24-bit address when the SPI_24 configuration mode is used or to configure from a 32-bit address when the SPI_32 configuration mode selection is used.

Multi-die SPI NOR Flash Devices

Spartan UltraScale+ FPGAs do not support configuration from a single flash device that uses multiple chip select pins. For use with Spartan UltraScale+ FPGAs, these types of devices must offer a transparent interface and read behavior to the FPGA. Spartan UltraScale+ FPGAs also issue a single SPI read command to read in an entire PDI. If a flash does not read seamlessly with a single command across any die boundaries, it cannot be used to store a configuration PDI if the PDI crosses the die boundary.

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. For more information, see Power-On Sequence Precautions for Flash.

Chapter 8

Master OSPI Configuration Mode

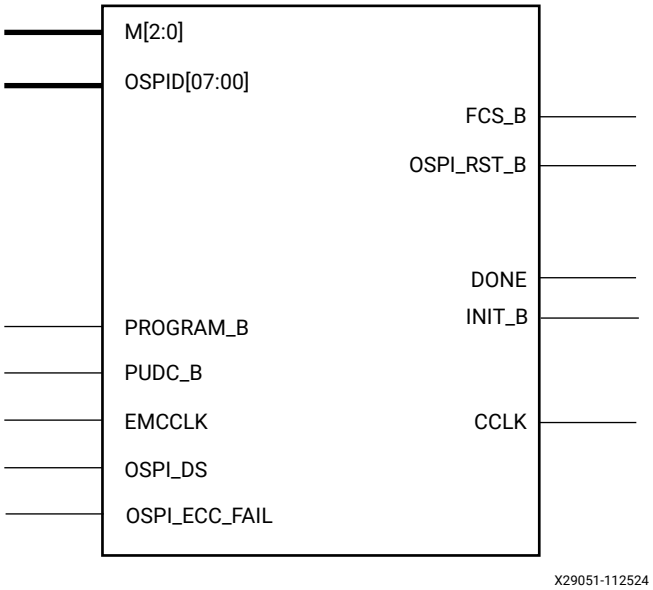
Introduction

The AMD Spartan™ UltraScale+™ FPGA OSPI configuration mode enables the use of select Octal SPI NOR flash devices for programmable device image (PDI) storage. The OSPI configuration mode supports a direct connection to the data, clock, data strobe, reset, and chip select signals of a octal SPI NOR flash for extracting a stored design image. See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for supported flash with each FPGA.

OSPI Interface

The master OSPI configuration mode supports select Octal SPI NOR flash with 8-bit data bus. For applications that require faster configuration times, the 8-bit data bus width provides an improvement over SPI configuration modes with 1-bit, 2-bit, or 4-bit data bus configurations. The master OSPI configuration interface is represented in the following figure.

Figure 25: **OSPI Configuration Interface**



OSPI Configuration Read Commands

The Master OSPI configuration mode supported Octal SPI flash read commands are listed in the following table.

Table 30: **Master OSPI Configuration Read Commands**

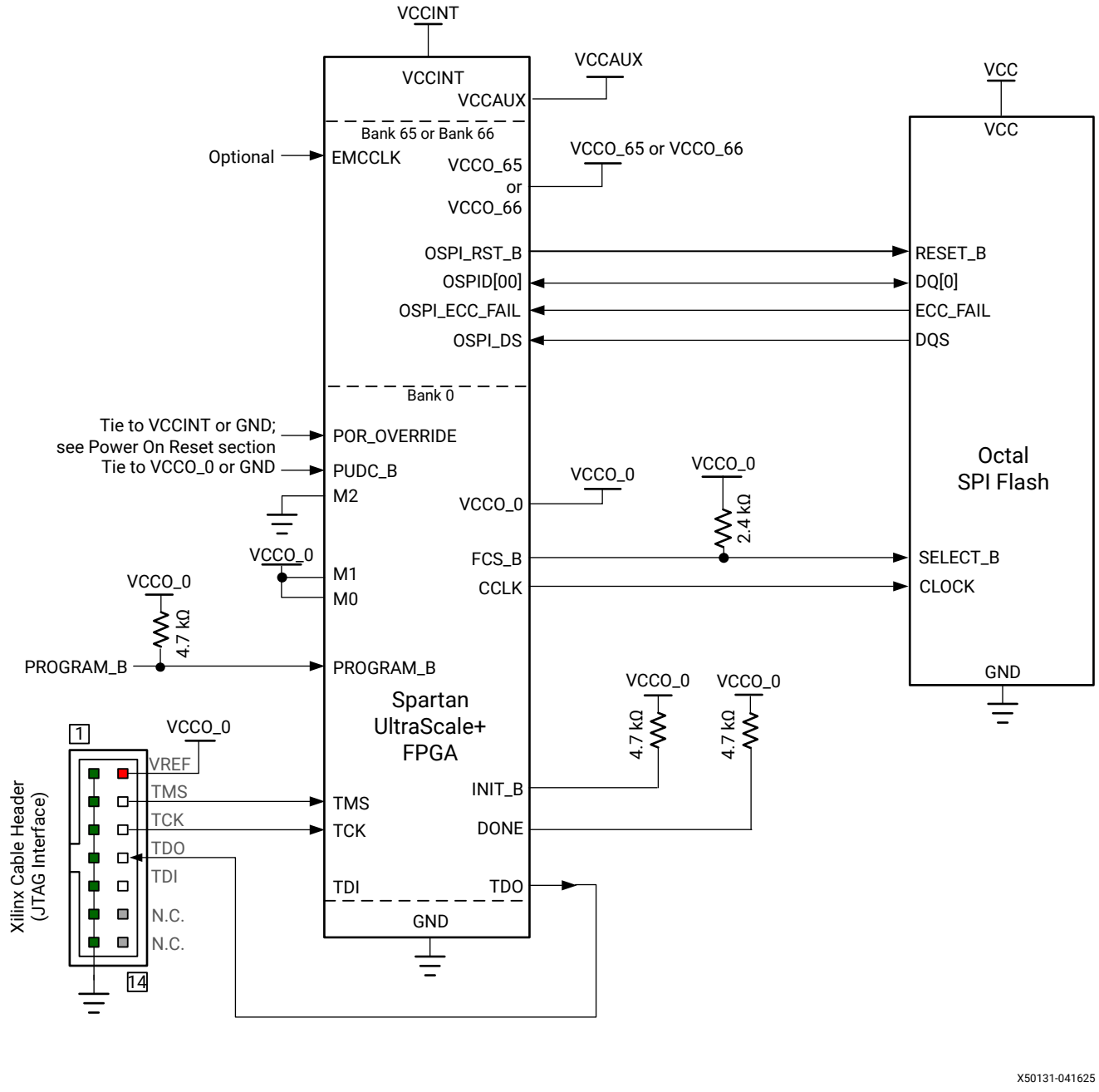
Configuration Mode	Data Width	Read Mode	Command Code (hex)	Dummy Cycles
OSPI	1	Read	03	-
OSPI	1	Read (4-byte)	13	-
OSPI	8	Octal Output fast read (4-byte)	7C	8

In the OSPI configuration mode, the Spartan UltraScale+ device initiates the configuration with the default 4-byte (32-bit) address octal output fast read command code `7Ch` and the BootROM searches for a valid header. If a valid header is not found, the device attempts to load the image using the 4-byte alternate addressing read command code `13h`. If a valid header is still not detected, the basic read command `03h` is tried. If the configuration attempt is unsuccessful after the third command, the BootROM increments the MultiBoot register read address offset by 32 KB and tries the OSPI command sequence again to locate a valid header in the PDI.

OSPI x1 Configuration Interface Example

See the following figure for the octal SPI interface connectivity.

Figure 26: **OSPI x1 Configuration Interface Example**



1. The `DONE` pin is by default an open-drain output. An external pull-up resistor is required.
2. The `INIT_B` pin is a bidirectional, open-drain pin. An external pull-up resistor is required.

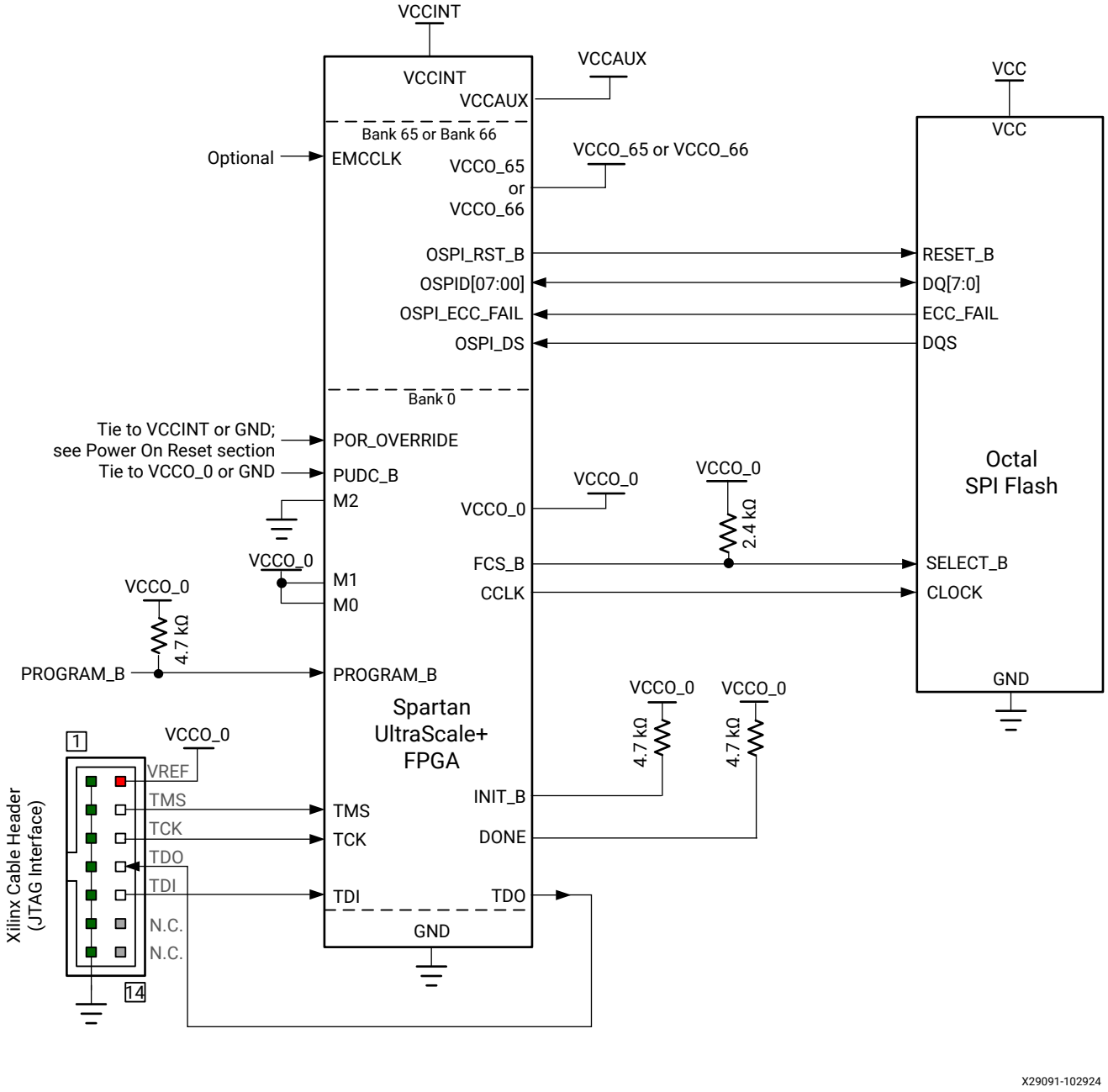
3. CCLK signal integrity is critical.
4. A series resistor should be considered for the data path from the flash to the FPGA to minimize overshoot. The proper resistor value can be determined from simulation.
5. The FPGA V_{CCO_0} supply must be compatible with the VCC for the I/O of the flash device.
6. 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.
7. Dependent on the Spartan UltraScale+ FPGA, Bank 65 or Bank 66 is used for the multi-function configuration I/O.

***Note*:** See Table 26 under Configuration Pins for signal details.

OSPI x8 Configuration Interface Example

See the following figure for the octal SPI interface connectivity.

Figure 27: **OSPI x8 Configuration Interface Example**



1. The `DONE` pin is by default an open-drain output. An external pull-up resistor is required.
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. A series resistor should be considered for the data path from the flash to the FPGA to minimize overshoot. The proper resistor value can be determined from simulation.
5. The FPGA V_{CCO_0} supply must be compatible with the VCC for the I/O of the flash device.

6. 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.
7. Dependent on the Spartan UltraScale+ FPGA, Bank 65 or Bank 66 is used for the multi-function configuration I/O.

***Note*:** See Table 26 under Configuration Pins for signal details.

Chapter 9

Slave Serial Configuration Mode

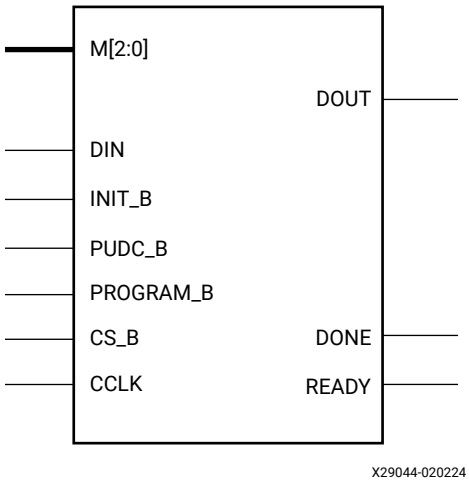
Introduction

In serial configuration mode, the FPGA is configured by loading one configuration bit per CCLK cycle. CCLK is an input in slave serial mode. AMD Spartan™ UltraScale+™ FPGAs include two additional signals on this interface from other AMD UltraScale™ architecture-based FPGA families, the READY and CS_B. The READY signal must be monitored by the host and when the pin is asserted it indicates that the FPGA is ready to accept data. The chip select (CS_B) signal is an active-Low signal that must be asserted (CS_B=0) before sending data to the FPGA.

Serial Interface

The serial configuration interface pins shown in the figure are defined in Table 28 under Configuration Pins Definitions. The following figure shows the basic serial configuration interface.

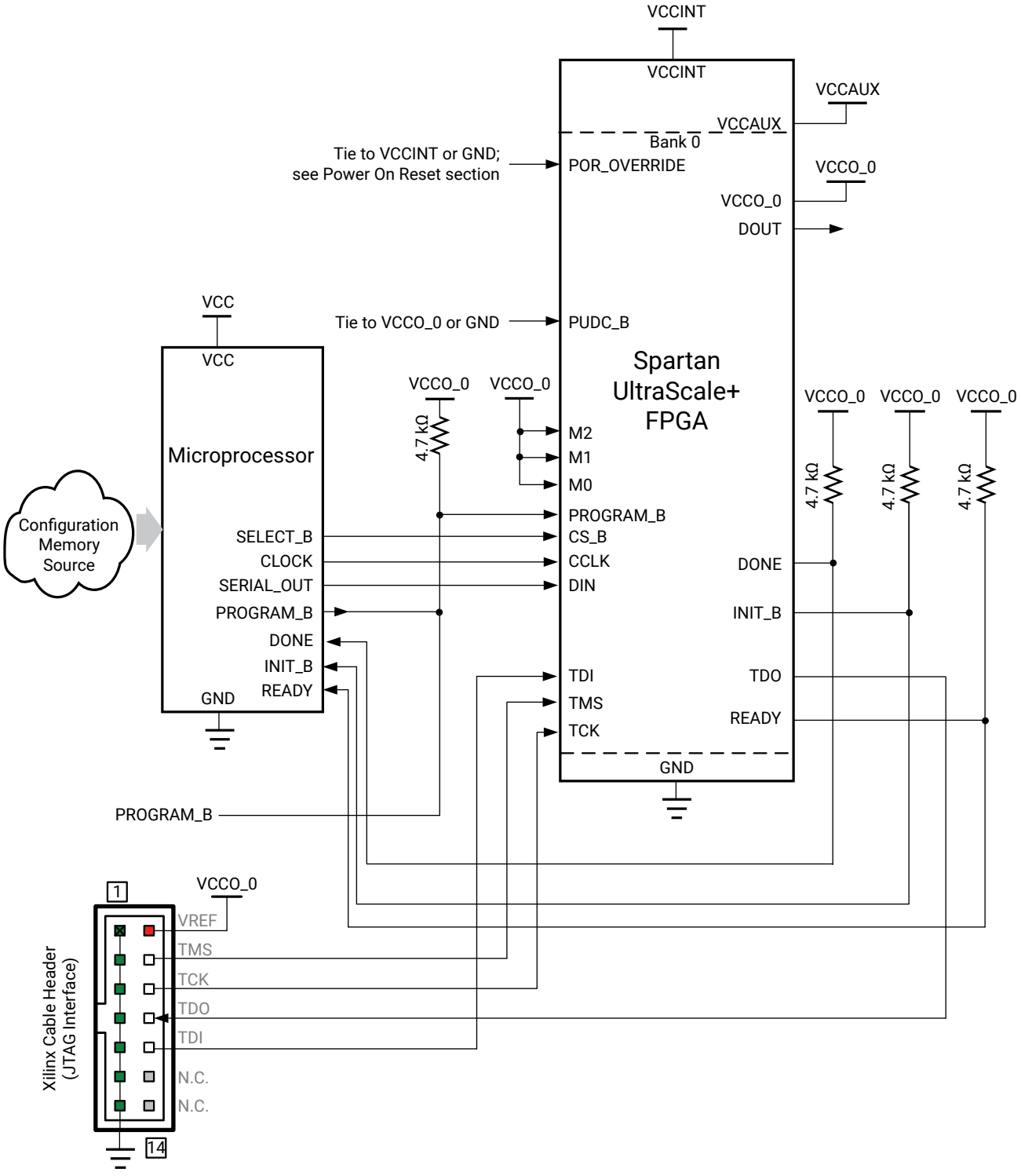
Figure 28: **Serial Configuration Interface**



Serial x1 Configuration Interface Example

Slave serial configuration is typically used for devices in a serial daisy chain or when configuring a single device from an external microprocessor (see the following figure). In Spartan UltraScale+ FPGA devices the `READY` signal can be deasserted at any stage during configuration, so this signal must be monitored to ensure that the interface is ready to accept data (`READY=1`). When the `READY` signal is deasserted it indicates that the FPGA data loading must be paused. The chip select input (`CS_B`) must be deasserted within 24 `CCLK` clock cycles to handle the data load stream without error.

Figure 29: **Serial x1 Configuration Interface Example**



Refer to the Notes following this figure for related information.

X29052-112624

1. The DONE pin is by default an open-drain output. An external pull-up resistor is required.
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. See the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) 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 SelectIOpins after power-up and during configuration.

***Note*:** See Table 26 under Configuration Pins for signal details.

Serial Mode Sequence

The Serial interface allows an external processor to load the configuration data. The functional waveform in this section shows an example of the Serial interface data, clock, and control signals sequence used to load data into the Spartan UltraScale+ FPGA. To configure the device in the Serial configuration mode, ensure that the dedicated boot mode pins are set to Serial (MODE[2:0])=111. The device samples the Serial DIN and clocks the READY on the rising CCLK edges. The waveform shows an example READY response. When the READY signal is deasserted, the CS_B signal must be deasserted within 24 CCLK cycles. Deasserting the CS_B will pause the PDI load until the READY signal is asserted again indicating the device is ready to receive data.

Figure 30: **Serial x1 Data Loading**

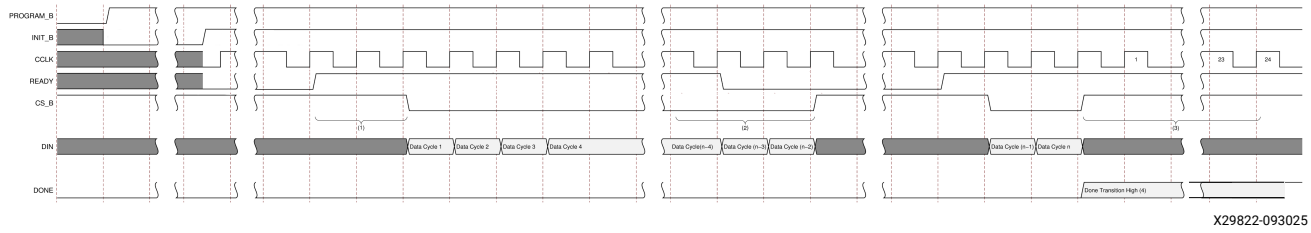


Figure Notes:

1. After READY is asserted and INIT_B is deasserted, the user can start data loading.
2. When READY is deasserted during Serial data loading, the CS_B must be deasserted within 24 CCLK cycles.
3. After the last data byte is sent, deassert CS_B. The time that it takes for DONE to transition high will be design dependent. If the design is configured to include ECC protected DDR memory, the DONE transition time can take up to 2 seconds.

Chapter 10

Slave SelectMAP Configuration Mode

Introduction

The AMD Spartan™ UltraScale+™ SelectMAP configuration interface provides an 8-bit, 16-bit, or 32-bit data bus interface to the FPGA configuration logic. In this mode, an external processor or controller drives the SelectMAP data, clock, and control signals (read/write and chip select) and needs to monitor the BUSY signal for configuration initiation and flow control (see Configuration Pins).

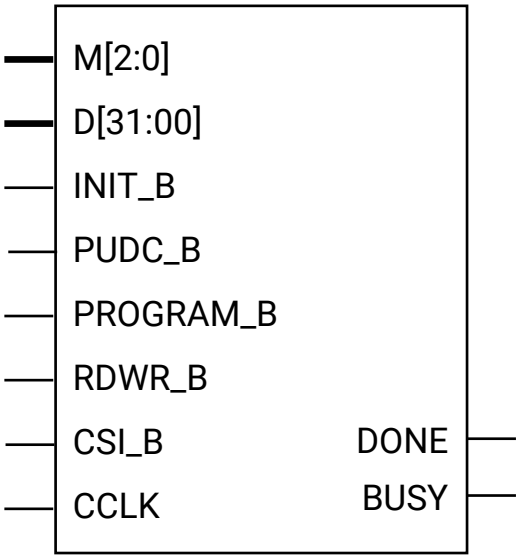
SelectMAP Interface

One or more devices can be configured through the SelectMAP bus. In the SelectMAP configuration mode there are multiple host options that can configure the Spartan UltraScale+ FPGA. For a typical setup, a processor provides data (D[31:00]) and clock (CCLK) inputs to one or multiple FPGAs on the data bus, the processor controls the chip select (CSI_B) input to select which FPGA to configure, and monitors the BUSY signal. The bus width in Slave SelectMAP configuration mode is automatically detected by the internal configuration logic and firmware.

Spartan UltraScale+ FPGA devices include a BUSY pin on the protocol interface that is not required on the other UltraScale+ FPGA devices. For Spartan UltraScale+ devices, the BUSY pin can be asserted at any stage during configuration and must be monitored to ensure that the interface is ready to accept data. When the BUSY signal is asserted it indicates that the FPGA data loading must be paused. The chip select input (CSI_B) must be deasserted to pause the data loading.

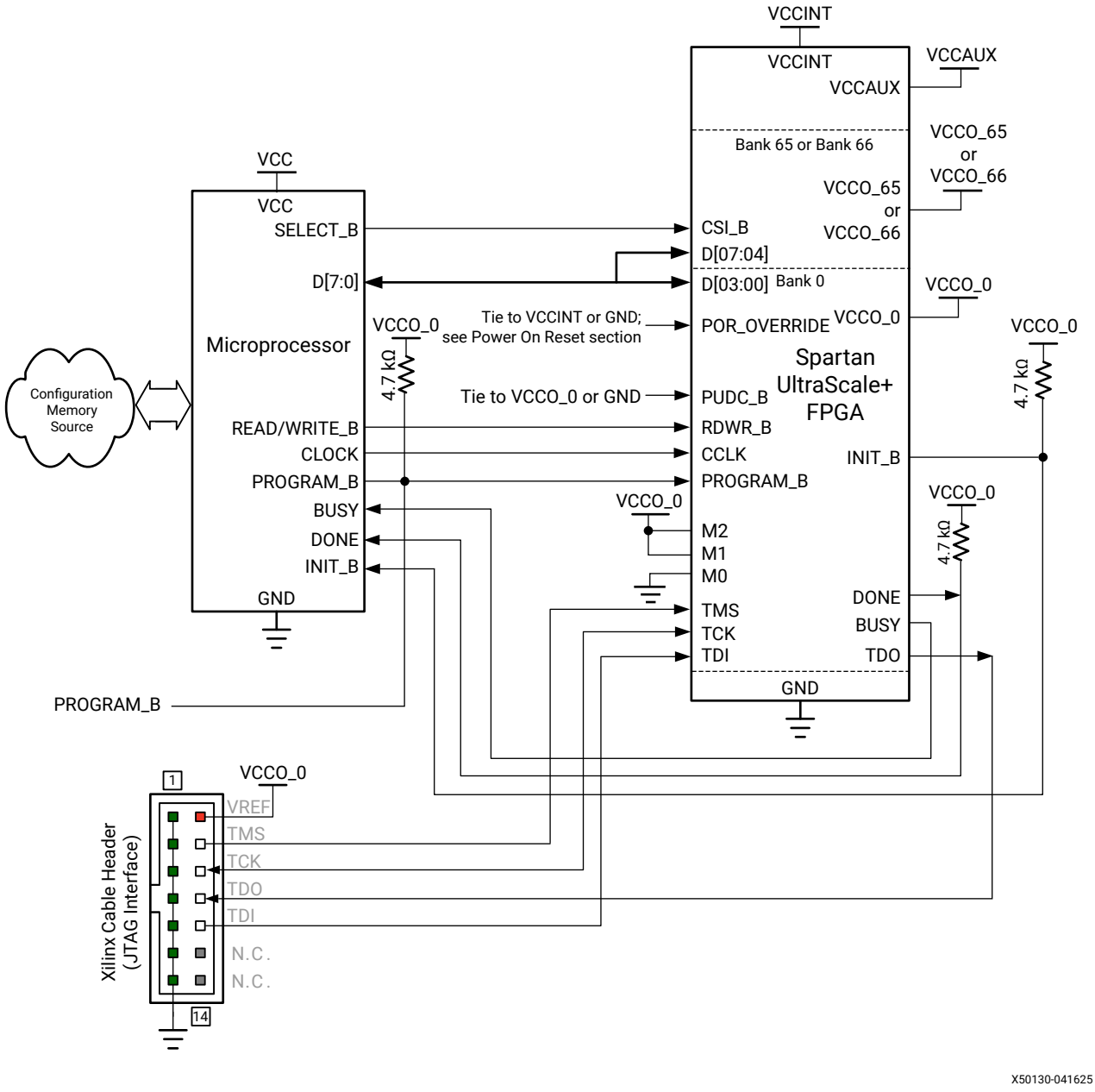
The SelectMAP configuration interface pins shown in the following figure are defined in Table 28.

Figure 31: **SelectMAP Configuration Interface**



SelectMAP x8 Configuration Interface Example

Figure 32: **SelectMAP x8 Configuration Interface Example**



1. The processor I/O needs to support a voltage that is compatible with the connected FPGA pins.
2. The `DONE` pin is an open-drain output. An external pull-up resistor is required.

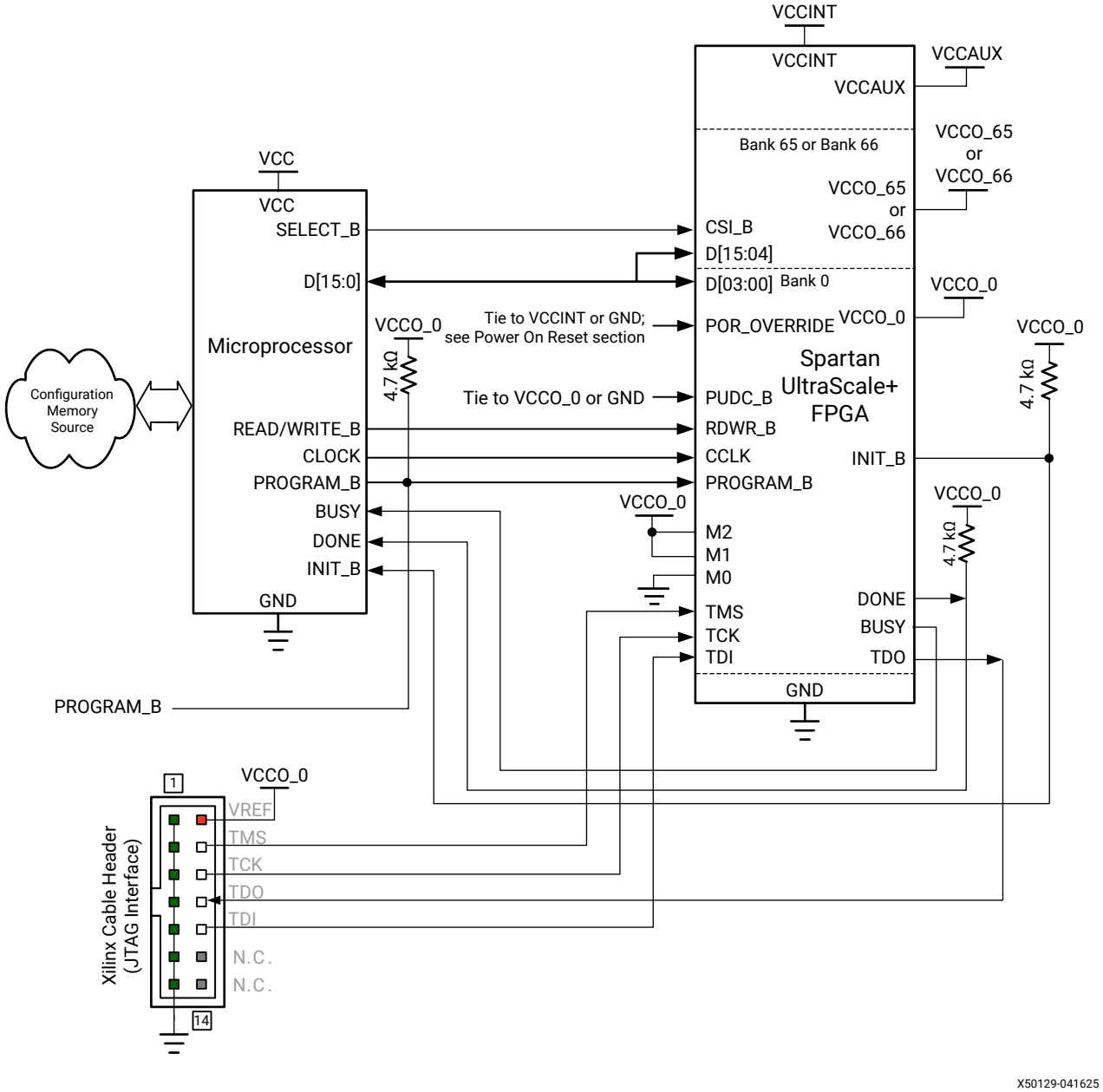
3. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
4. CCLK signal integrity is critical.
5. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to V_{CCO_0} to 3-state the SelectIOpins after power-up and during configuration.
6. Dependent on the Spartan UltraScale+ FPGA, Bank 65 and/or Bank 66 is used for the multi-function configuration I/O.

Note: See Table 26 under Configuration Pins for signal details.

SelectMAP x16 Configuration Interface Example

See the following figure for the SelectMAP interface connectivity.

Figure 33: **SelectMAP x16 Configuration Interface Example**



1. The processor I/O needs to support a voltage that is compatible with the connected FPGA pins.
2. The `DONE` pin is an open-drain output. An external pull-up resistor is required.
3. The `INIT_B` pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
4. `CCLK` signal integrity is critical.
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 SelectIOpins after power-up and during configuration.

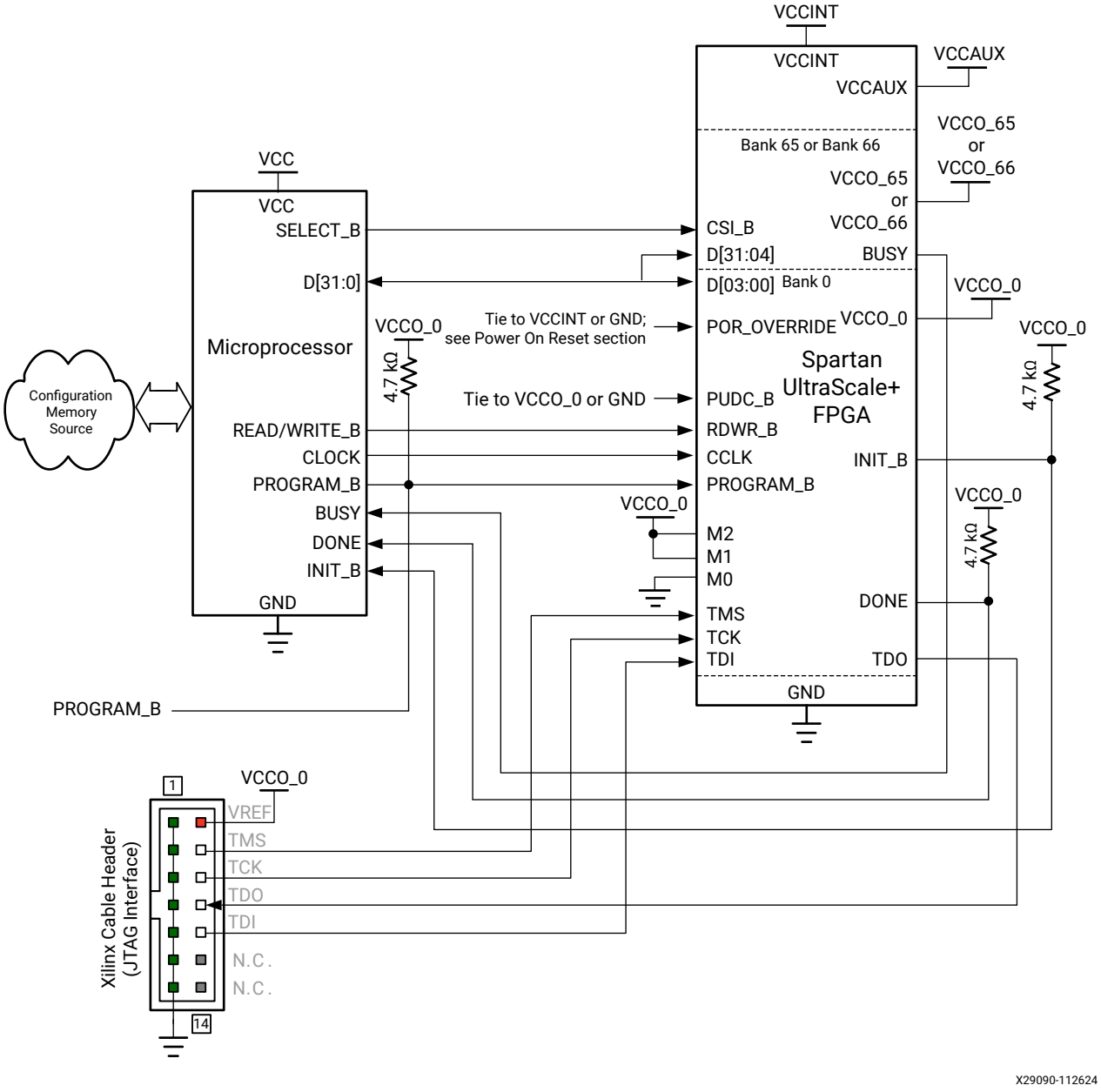
6. Dependent on the Spartan UltraScale+ FPGA, Bank 65 and/or Bank 66 is used for the multi-function configuration I/O.

Note: See Table 26 under Configuration Pins for signal details.

SelectMAP x32 Configuration Interface Example

See the following figure for the SelectMAP interface connectivity.

Figure 34: **SelectMAP x32 Configuration Interface Example**



1. The processor I/O needs to support a voltage that is compatible with the connected FPGA pins.
2. The `DONE` pin is an open-drain output. An external pull-up resistor is required.
3. The `INIT_B` pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
4. `CCLK` signal integrity is critical.
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 SelectIOpins after power-up and during configuration.

6. Dependent on the Spartan UltraScale+ FPGA, Bank 65 and/or Bank 66 is used for the multi-function configuration I/O.

Note: See Table 26 under Configuration Pins for signal details.

SelectMAP Data Loading

The Spartan UltraScale+ FPGA SelectMAP interface provides continuous data loading. Data loading is controlled by the BUSY (monitor signal), 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 3-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.

RDWR_B

RDWR_B is an input to the FPGA that controls the data path for write control. For configuration, RDWR_B must be set for write control (RDWR_B = 0). Slave SelectMAP solution readback support is not available in Spartan UltraScale+.

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.

BUSY

Busy active-High output asserts (BUSY=1) when the FPGA data loading must be paused. When BUSY asserts this indicates there are 24 CCLK clock cycles left before the internal configuration logic overflows and is not able to handle the data load stream.

Spartan UltraScale+ FPGAs use the SelectMAP interface pins described for data loading. A functional waveform example of the SelectMAP data loading process is provided along with bus detection details in the following sections.

SelectMAP Bit Order

The SelectMAP interface is typically driven by a user application residing on a microprocessor, microcontroller, or another FPGA or SoC. The Spartan UltraScale+ FPGA automatically detects the bus width during configuration. For these applications, it is valuable to understand how the data ordering in the programmable Spartan UltraScale+ device's image corresponds to the data ordering expected by the device's interface when debugging. The following table shows how to load the SelectMAP PDI data bits onto the SelectMAP data pins.

Table 31: **SelectMAP PDI Bus Detection Pattern**

SMAP Bus Width	SMAP_CLK Cycle															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
D[07:00]	0x00	0x00	0x00	0xDD	0x11	0x22	0x33	0x44	0x55	0x66	0x77	0x88	0x99	0xAA	0xBB	0xCC
D[15:00]	0x0000	0x00DD	0x1122	0x3344	0x5566	0x7788	0x99AA	0xBBCC								
D[31:00]	0x000000DD	0x11223344	0x55667788	0x99AABBCC												

Notes:

1. Spartan UltraScale+ SelectMAP mode does not require bit-swapping the data from the programming image.

SelectMAP Mode Sequence

The SelectMAP interface allows an external processor to load the configuration data. The functional waveform in this section shows an example of the SelectMAP interface data, clock, and control signals sequence used to load data into the Spartan UltraScale+ FPGA. To configure the device in SelectMAP, ensure that the dedicated boot mode pins are set to SelectMAP (`MODE[2:0])=110` and `RDWR_B=0`). The device samples the SelectMAP datapins on the rising `CCLK` edges and the `BUSY` signal is also clocked by the `CCLK`. The waveform shows the 16 byte bus detect pattern for the SelectMAP boot mode. The waveform shows an example `BUSY` response. When the `BUSY` signal is asserted, the `CSI_B` signal must be deasserted within 24 `CCLK` cycles.

Figure 35: **SelectMAP 32-bit Data Loading**

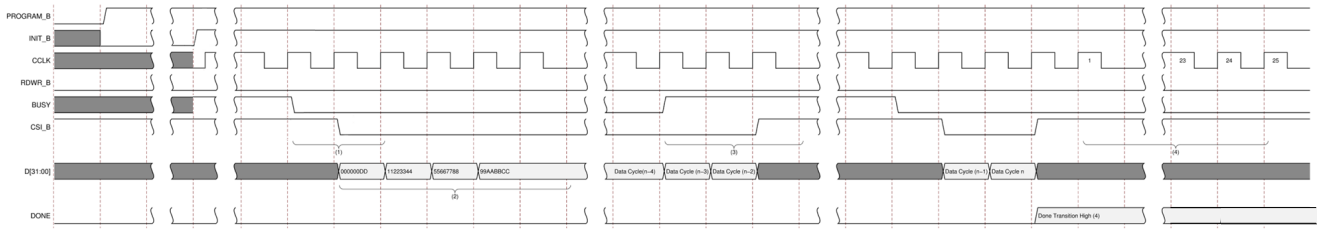


Figure Notes:

1. After `INIT_B` and `BUSY` are deasserted, the user can start data loading.
2. The SelectMAP bus detect pattern is the first 16 bytes of the PDI. The `JTAG_STATUS[5:4]` SelectMAP bus width field is updated when this pattern is read by the BootROM.

3. When BUSY is asserted during SelectMAP data loading, the CSI_B must be deasserted within 24 CCLK cycles ($C_{SMAPBUSYCS}$).
4. After the last data byte is sent, deassert CSI_B. The time that it takes for DONE to assert will be design dependent. If the design is configured to include ECC protected DDR memory, the DONE transition time can take up to 2 seconds.

Chapter 11

Boundary-Scan and JTAG Configuration Mode

Introduction

AMD Spartan™ UltraScale+™ FPGAs support IEEE standards 1149.1 and 1149.6 (ACJTAG), defining a test access port (TAP) and boundary-scan architecture. The TAP and boundary-scan architecture is commonly referred to collectively as JTAG. JTAG is an acronym for the Joint Test Action Group, the technical subcommittee responsible for developing the initial 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 the board level.

In addition to connectivity testing, the boundary-scan architecture offers flexibility for vendor-specific instructions, such as `JCONFIG`, 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, 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`. 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.

In addition, for compliance the PROGRAM_B pin must be High to have proper JTAG functionality. For example, if the PROGRAM_B pin is held Low, the IDCODE value might be incorrect.

Boundary-Scan Using IEEE Standard 1149.1

UltraScale architecture is fully compliant with the IEEE Standard 1149.1 TAP 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 the following table). 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 AMD devices with parts from different vendors because these optional pins might need to be driven.

Table 32: **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 32: **TAP Controller Pins** *(cont'd)*

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.

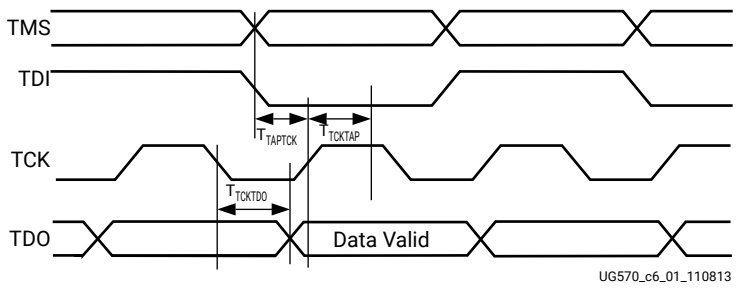
Notes:

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 *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) for internal pull-up values.

Boundary-Scan Timing Parameters

Characterization data for some of the most commonly requested timing parameters, shown in the following figure, are listed in the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) in the *Configuration Switching Characteristics* table.

Figure 36: **Boundary-Scan Port Timing Waveforms**

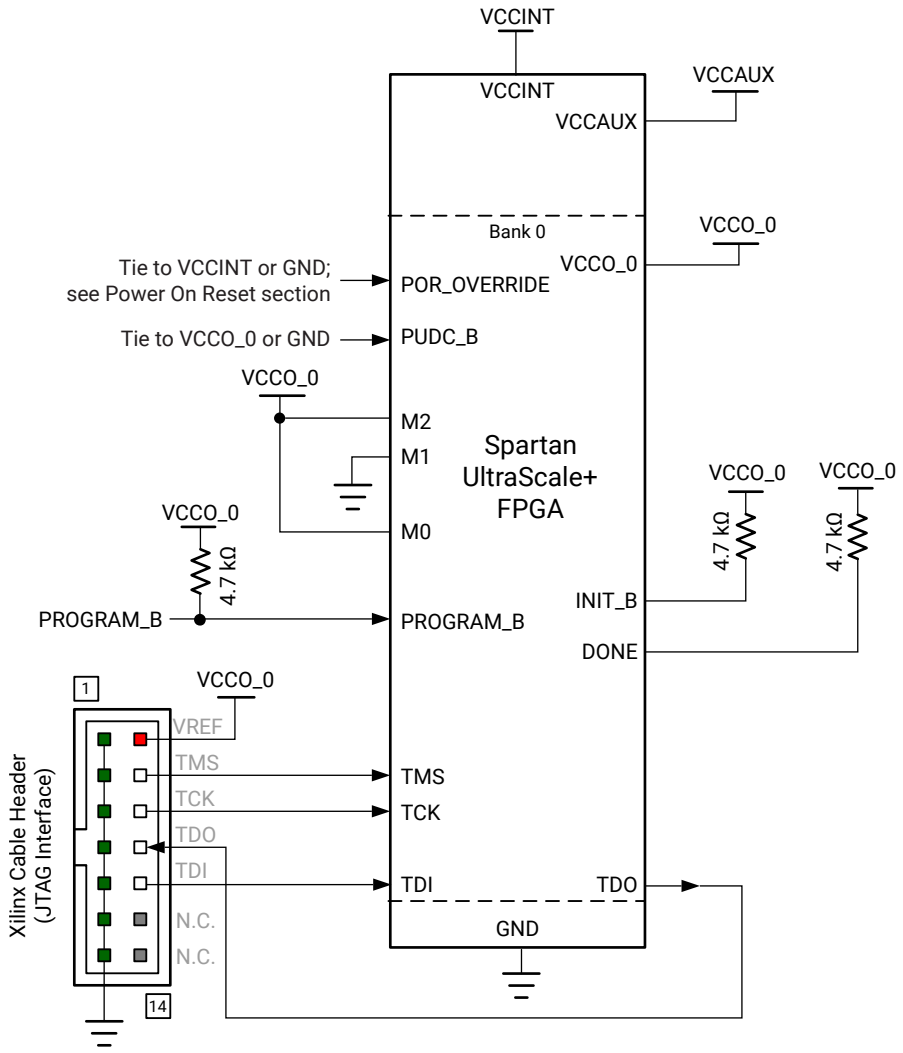


JTAG Configuration for Spartan UltraScale+ Devices

For single-device configuration, the TAP controller commands are issued automatically if the part is being configured with AMD Vivado Design Suite. The download cable must be attached to the appropriate four JTAG pins (TMS, TCK, TDI and TDO) to deliver the programmable device image (PDI) from the computer port to the FPGA. See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for supported cables and PDI programming detail. The Vivado Design Suite hardware manager tool automatically checks that a valid AMD device is connected and drive the commands to deliver and/or verify that the configuration bits are properly managed.

The following figure 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 an AMD programming cable under the control of the Vivado Design Suite or with a third-party JTAG test tool and STAPL file from Vivado HW Manager. The connections essential to configure the device in the JTAG configuration mode are shown.

Figure 37: **JTAG Configuration Interface Example**



Refer to the Notes following this figure for related information.

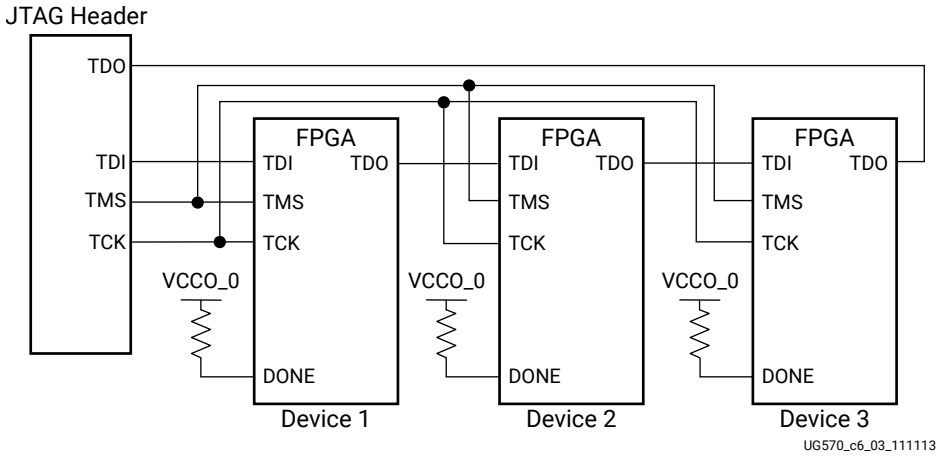
X50352-051325

1. The `DONE` pin is by default an open-drain output. An external pull-up resistor is required.
2. The `INIT_B` pin is a bidirectional, open-drain pin. An external pull-up resistor is required.
3. See the *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) for the VCCINT, VCCAUX, and VCCO_0 supply voltages.

4. The FPGA PUDC_B pin is tied to GND to enable internal pull-ups or it can be tied to VCCO_0 to 3-state the SelectIO pins after power-up and during configuration.

It is also possible to configure multiple devices in a chain, as shown in the following figure.

Figure 38: **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.

Securing a Device

For JTAG boundary-scan test, the device must not be secured because security features can block JTAG boundary-scan access. Security can be applied after the boundary-scan test.

Configuration and Mode Setting

The Boundary Scan Description Language (BSDL) file is a boundary-scan model of an unconfigured device. Boundary-scan tests based on the BSDL file must be applied to an unconfigured and unsecured device.

For JTAG boundary-scan test, the configuration mode pins must be set to the JTAG configuration mode (M[2:0]=101). To accommodate JTAG boundary-scan test, the PCB design must allow for the M[2:0] pins to be changed between the JTAG mode for JTAG boundary-scan test and the primary configuration mode for normal system configuration and operation. If the device is set to JTAG configuration mode, the device remains unconfigured after power-on initialization (unless a configuration PDI is delivered via JTAG) and then JTAG boundary-scan tests can be applied to the unconfigured device.

IMPORTANT! *In SU10P, SU25P, SU35P devices, JTAG boundary-scan is only accessible after power-up initialization when the device is in JTAG configuration mode and is unsecured. For other configuration modes, the JTAG security gate (default TAP_SECURITY.SEC_GATE register setting) blocks JTAG boundary-scan access in these devices. In unsecured devices, the security gate can be released by applying a JPROG instruction followed by a BYPASS instruction and waiting for 100 ms, which internally resets the device to JTAG configuration mode.*

TAP Controller and Architecture

The FPGA TAP contains four mandatory dedicated pins as specified by the protocol given in the Boundary-Scan Timing Parameters section and illustrated in the following figure, a typical JTAG architecture.

Figure 39: **Typical JTAG Architecture**

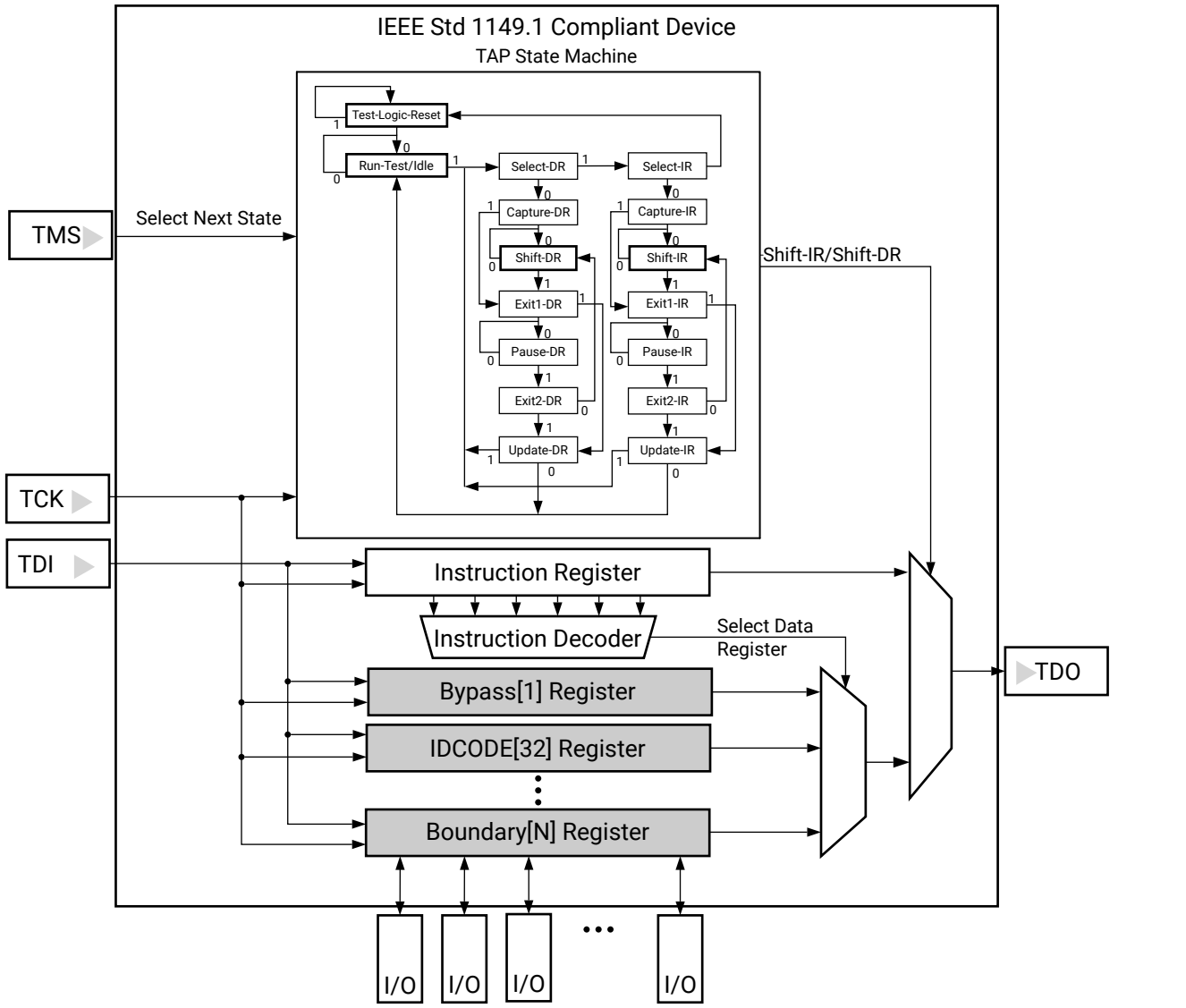
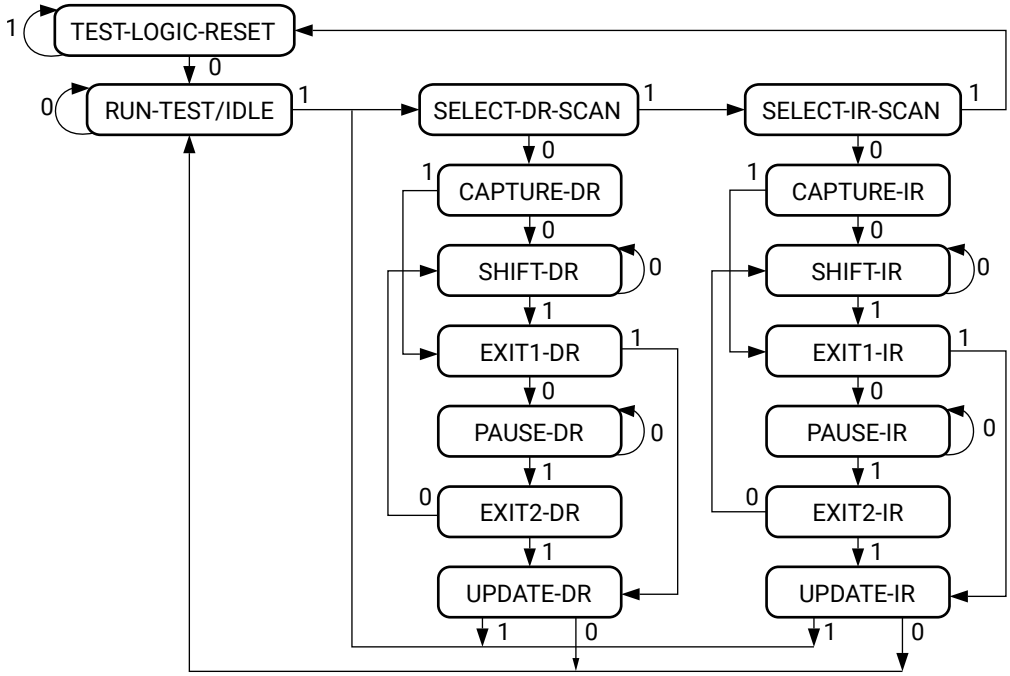


Figure 40 shows a 16-state finite state machine. The four TAP pins control how data is scanned into the various registers. The state of the `TMS` pin at the rising edge of `TCK` determines the sequence of state transitions. There are two main sequences, one for shifting data into the data register and 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 40: **Boundary-Scan TAP Controller State Machine**



NOTE: The value shown adjacent to each state transition in this figure represents the signal present at TMS at the time of a rising edge at TCK.

UG570_c6_05_111313

The following describes operation of each state.

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 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 AMD 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 of the device. INTEST is not supported. The `HIGHZ_IO` command is similar to the standard `HIGHZ` command but only disables the SelectIO™pins.

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

Boundary-Scan Architecture Registers

Spartan UltraScale+ 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 the following table).

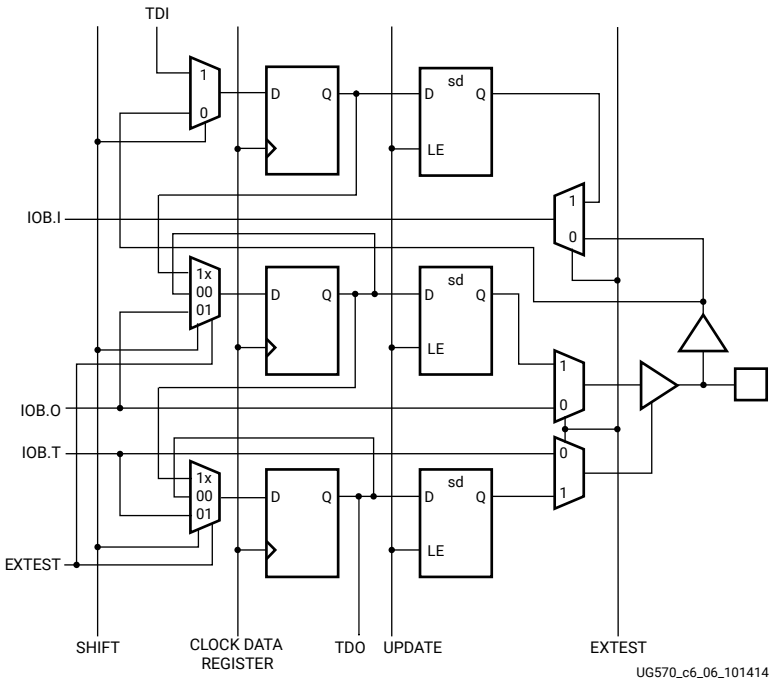
Table 33: **JTAG Registers**

Register Name	Register Length	Description
Boundary	Up to 3 bits per I/O	Controls and observes input, output, and output enable.
Instruction	6 bit	Holds the current instruction opcode.
Bypass	1 bit	Bypasses the device.
Device Identification (IDCODE)	32 bit	Captures the device IDCODE.
JTAG Configuration	N/A	Allows access to the configuration bus when using the JCONFIG instruction to load a programmable device image (PDI) and enables read-only connection from the internal configuration bus to the TDO pin.
USERCODE	32 bit	Captures the user-programmable code.
User-Defined (USER1, USER2, USER3, and USER4)	Design specific	Design specific registers.
JTAG Status	48 bit	Captures the overall device status.
Error Status	61 bit	Captures the error status of the device.
FW Status	96 bit	Captures the PMC PLM status for when debug option enable
DNA	96 bit	Captures the device DNA.

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. The following figure is a representation of the Spartan UltraScale+ FPGA boundary-scan architecture.

Figure 41: **Spartan 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 Spartan UltraScale+ FPGAs. The BSDL files can be obtained from the AMD 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.

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.

The instruction register is 6 bits wide for Spartan UltraScale+ devices. See Spartan UltraScale+ Devices and JTAG IDCODEs. To invoke an operation an opcode is loaded into the Spartan UltraScale+ FPGA JTAG boundary-scan Instruction register. The following table describes the boundary-scan instructions for Spartan UltraScale+ devices.

Table 34: **Spartan UltraScale+ FPGA JTAG Boundary-Scan Instructions**

Boundary-Scan Command	Binary Code [5:0]	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 1149.1 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 1149.1 BYPASS operation
JCONFIG	000101	Access the configuration bus for configuration
JRDBK	000100	Access the configuration bus for readback
JPROG	001011	Equivalent to PROGRAM_B pin and sets the internal mode pins to JTAG. This enables JTAG access when the M[2:0] are not physically set to the JTAG configuration mode
SYSMON_DRP	011100	System Monitor DRP access through JTAG. See the *UltraScale Architecture System Monitor User Guide* (UG580).

Table 34: **Spartan UltraScale+ FPGA JTAG Boundary-Scan Instructions** *(cont'd)*

Boundary-Scan Command	Binary Code [5:0]	Description
READ_DNA	110010	Read eFUSE DNA value
SYS_RST	110111	System reset instruction
STATUS	011111	Reads the JTAG_STATUS register
ERROR_STATUS	111110	Reads the PMC ERROR_STATUS register for error management
FW_STATUS	011101	Reads the PMC FW STATUS register for PLM debug, when BITSTREAM.CONFIG.MULTIBOOT_DEBUG property is enabled
RESERVED	All other codes	AMD reserved instructions

The following table shows the instruction capture values loaded into the IR as part of an instruction scan sequence.

Table 35: **Instruction Capture Value**

	IR[5]	IR[4]	IR[3]	IR[2]	IR[1:0]	
TDI →	0	0	0	0	01	→ TDO

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

Spartan 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 Spartan UltraScale+ Devices and JTAG IDCODEs for more information.

JTAG Configuration Register

The JCONFIG instruction enables a write-only connection from the TDI pin to the internal configuration bus with the JTAG configuration register.

USERCODE Register

The USERCODE instruction is supported in the Spartan UltraScale+ FPGAs. This USERCODE register allows you 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 PDI during file generation using the BITSTREAM.CONFIG.USERID property and is valid only after configuration.

USER1, USER2, USER3, and USER4 Registers

The USER1, USER2, USER3, and USER4 registers are only available after configuration. You must define these four registers 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).

JTAG Status Register

The JTAG Status register indicates the value of global signals and select dedicated configuration status pins and signals. A detailed explanation of each bit position is given in the following table.

Table 36: **JTAG Status Register Description**

Name	Bit Index	Description
RSVD READS 0	47	Reserved reads 0
INIT_COMPLETE	46	Device initialization complete
DCI_LOCK	45	DCI lock status
PLL_LOCK	44	PLL/DDRMC lock status
EOS	43	End of Startup
GWE	42	Global write enable
GTS CFG	41	Global tri-state
GHIGH	40	Global GHIGH
RESERVED	39:37	Reserved
PUDC_B	36	Value on the PUDC_B pin.
RESERVED	35	Reserved
DONE	34	Value on the DONE_PIN pin. A value of 1 on DONE indicates boot and configuration is complete.
JRDBK ERROR	33	JTAG readback status indicator. A value of 1 on JRDBK indicates an error reading data from the slave boot interface (SBI).
JCONFIG ERROR	32	JTAG data load error indicator. A value of 1 indicates the SBI is not ready to accept data.
PMC VERSION	31:28	PMC version
RESERVED	27:24	Reserved
JTAG SEC GATE	23	Security gate status.
RESERVED	22	Reserved

Table 36: **JTAG Status Register Description** *(cont'd)*

Name	Bit Index	Description
PMC SCAN CLEAR DONE	21	Scan clear done indicator. A value of 1 indicates the scan clear is complete.
PMC SCAN CLEAR PASS	20	Scan clear pass indication. A value of 1 means the scan clear passed.
CRP JTAG STATUS	19:17	Clock and reset status. A value of 3'b111 indicates the device is still held in POR reset.
RESERVED	16:15	Reserved
BOOT MODE	14:12	Value on the M[2:0] mode pins at release of PROGRAM_B.
RESERVED	11:8	Reserved
AES KEY ZEROIZED	7	AES key zeroized indicator. A value of 1 indicates all keys are zeroized.
RESERVED	6	Reserved
SELECTMAP BUS WIDTH	5:4	SelectMAP configuration mode bus width detected. • 00 = No bus width detected • 01 = SelectMAP 8-bit • 10 = SelectMAP 16-bit • 11 = SelectMAP 32-bit
SBI JTAG ENABLED	3	SBI JTAG indicator. A value of 1 indicates the SBI is configured to receive data from the JTAG interface.
SBI JTAG BUSY	2	SBI JTAG busy indicator. A value of 1 indicates the SBI is busy and cannot accept data when in JTAG mode.
RSVD READS 0	1	Reserved reads 0
RSVD READS 1	0	Reserved reads 1

Error Status Register

The error status register is read-only data register that contains error conditions associated with the PMC and configuration controller. See the following table for the error status register bit fields.

Table 37: **Error Status Register Description**

Name	Bit Index(1)
PMC_BOOT_ERR_DATA	60:31
DFX_UNEXPECTED_ACTIV_ERR	30
JTAG_TOGGLE_ERR	29
EFUSE_CACHE_ERR	28
ROM_VALID_ERR	27
PMC_FW_NCR_ERR	26
PMC_FW_CR_ERR	25
PMC_BOOT_NCR_ERR	24
PMC_BOOT_CR_ERR	23
SBI_SMAP_OVF_ERR	22

Table 37: **Error Status Register Description** *(cont'd)*

Name	Bit Index[1]
SSS_CFG_ERR	21
EFUSE_RD_ERR	20
EFUSE_PGM_ERR	19
OSPI_ERR	18
SBI_JTAG_OVF_ERR	17
SBI_SS_OVF_ERR	16
SBI_MCAP_OVF_ERR	15
SBI_SMAP_CONFIG_ERR	14
SBI_AXI_SLVERR	13
SBI_SMAP_ABORT	12
RESERVED	11:8
CCU_TAMPER_FABRIC	7
SYSMON_OT	6
CCU_BAD_PR_FORMAT	5
CCU_SEU_ECC_ERR	4
CCU_SEU_CRC_ERR	3
CCU_CRC_ERR	2
CCU_IDCODE_ERR	1
CCU_SLVERR	0

Notes:

1. After INIT_B pin deassertion after POR, bit fields Error Status[60:31] and [24:23] should be reviewed for any errors that occur during the BootROM phase. The remaining Error Status register bit fields are valid indicators after the INIT_B pin is released and PLM initiated.

FW Status Register

The FW Status register is read-only data register that contains PMC PLM status information for debug when the BITSTREAM.CONFIG.MULTIBOOT_DEBUG option is enabled. See the following table for the error status register bit fields.

Table 38: **FW Status Register Description**

Name	Bit Index
PMC_FW_NCR_FLAG	95
PMC_FW_CR_FLAG	94
PMC_FW_ERROR_DATA	93:64
PMC_FW_STATUS[1]	63:32

Table 38: **FW Status Register Description** *(cont'd)*

Name	Bit Index
PMC_FW_DATA	31:0

Notes:

1. PMC_FW_STATUS field provides PLM status information. Example: PMC_FW_STATUS [63:32]=0x00000010 indicates a full PDI load was completed successfully, and PMC_FW_STATUS[63:32]=0x000000AU can indicate the user configured clocks (related to improved performance OSPI DDR enable, >50MHz configrate, or EMCCLK) reinitialization with the flash failed.

DNA Register

The DNA register contains a unique device identifier (device DNA) that is 96-bits. The device DNA value can be accessed through JTAG. See Device Identifier (Device DNA) for more information.

Chapter 12

Security, eFUSEs, and Device DNA

Introduction

This chapter describes the AMD Spartan™ UltraScale+™ advanced security features including:

- Encryption and Authentication Secure Configuration
  - Asymmetric Hardware Root of Trust (A-HWRoT)
  - Symmetric Hardware Root of Trust (S-HWRoT)
- Physically Unclonable Function creates a unique signature or fingerprint for each device
- True Random Number Generator generates cryptographically secure random numbers.
- eFUSE
  - Programmable device image (PDI) key storage
  - Secure configuration mode management
  - 32-bit user value (EFUSE_USR)
  - Device Identifier (Device DNA)

Security Features Summary

The following table summarizes the different security feature capabilities for the devices in the Spartan UltraScale+ family.

Table 39: **Security Features Summary**

	SU10P	SU25P	SU35P	SU45P	SU55P	SU60P	SU65P	SU100P	SU150P	SU200P
Secure Configuration Algorithms	HSS			HSS, LMS, or ECDSA P-384						
Number of eFUSE Digests	3 PPKs or 2 PPKs + PUF HD Digest			2 PPKs or 1 PPKs + PUF HD Digest						
SHA3 Engine Support Algorithms	SHA3-256, SHAKE-256			SHA3-256, SHA3-384, SHAKE-256						
eFUSE Digest Algorithm	SHA3-256			SHA3-384						
AES	AES-GCM-256									
TRNG	Yes									

Encryption and Authentication Secure Configuration

The Spartan UltraScale+ FPGAs support encrypted and authenticated secure configuration to provide a high degree of design security. Two hardware root of trust (HWRoT) secure configuration modes are implemented to support the use of asymmetric authentication (I.e., public key) or symmetric authentication and encryption when configuring the device. The two root of trust modes are not mutually-exclusive. Each has its own set of rules. If both modes are used together, the rules enforced by the secure configuration are the combination rules from both root of trust modes.

Asymmetric Hardware Root of Trust (A-HWRoT) secure configuration forces the use of asymmetric authentication with optional encryption when configuring the device. Using A-HWRoT ensures authenticity and integrity of the programmable device image (PDI). Coupling the use of encryption with A-HWRoT provides confidentiality of the PDI. In Spartan UltraScale+ FPGAs, asymmetric authentication of the PDI is done using post-quantum cryptography (PQC) or in some devices ECDSA P-384. See Table 39 for details.

Symmetric Hardware Root of Trust (S-HWRoT) secure configuration forces the use of symmetric authentication and encryption when configuring the device to provide authenticity, integrity, and confidentiality. The S-HWRoT achieve authenticity of the image via the GCM mode of AES-256 by encrypting all portions of the PDI, excluding the boot header and the Hash Block. With S-HWRoT, the Hash Block is authenticated as additional authenticated data (AAD). The authenticated hash block data is used to validate the integrity of the PDI.

Spartan UltraScale+ FPGAs support dynamic power analysis (DPA) countermeasures of AES HW masking, key rolling, and authentication before decryption.

Advanced Encryption Standard Overview

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

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 (https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf). An advantage of the AES-GCM algorithm is that it also supports built-in authentication. Although the AES-GCM algorithm is a self authenticating algorithm, it does so with a symmetric key, meaning that the key used to encrypt is the same as the one to decrypt. This key must be protected as it is secret.

The Spartan UltraScale+ FPGA AES system consists of software-based PDI file encryption and on-chip PDI decryption using a dedicated hardened AES engine with dedicated memory for storing the encryption key. UltraScale architecture-based FPGAs store the encryption key internally in the nonvolatile, one-time-programmable eFUSE. The encryption key can only be programmed onto the device through the external JTAG port or through the internal AXI32 primitive and cannot be read back. The encryption key can either be stored unencrypted or encrypted with the physically unclonable function (PUF). Spartan UltraScale+ FPGAs provide the option for black (encrypted) key storage of the AES key using a key encryption key (KEK) created using the PUF. In this case the AES key is stored in a secure encrypted form to protect the AES key.

Programmable Device Image Authentication

In Spartan UltraScale+ FPGAs asymmetric authentication of the PDI is done using post-quantum cryptography (PQC) or in some devices ECDSA P-384. See Table 39 for details. The primary public key (PPK) is only used for verifying the signature of the secondary public key (SPK), while the SPK is used to authenticate hash block. The hash block contains hashes of the boot header, PLM, and the programmable logic bitstream. The authenticated hashes in the hash block are used to validate the boot header, PLM, and the bitstream. The following table lists the characteristics of each public key type.

Table 40: **Public Key Types**

Public Key	Number	Location	Revocable
Primary (PPK)	up to 3	External memory with hash in eFUSEs	Yes
Secondary (SPK)	96	Programmable device image (PDI)	Yes

To reduce the number of eFUSEs required in the device, the full public key is stored in the PDI while the hash (either SHA3-256 or SHA3-384) of each key is securely stored inside the device using eFUSEs. During the secure configuration process, the internal configuration logic and firmware first validates the integrity of the full public key stored externally by hashing it and then taking that hash and comparing it against the value stored in eFUSEs. There are also 96 SPKs available, each of which are also revocable. The SPK is delivered inside the authenticated PDI and is signed by the PPK, which is the primary purpose of the PPK. The SPK is intended to authenticate everything else.

If authentication passes, the configuration goes to completion through the startup cycle. Pulsing the PROGRAM_B signal or power-on reset is required to reset the configuration interface.

Differential Power Analysis Countermeasures

Spartan UltraScale+ FPGA differential power analysis (DPA) countermeasures include:

- AES HW masking
- Rolling keys
- Authentication before decryption

UltraScale+ FPGAs allow you to break up the PDI 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).

AES HW masking is employed in the AES engine. This additional masking countermeasure lowers the signal-to-noise ratio (SNR) making DPA attacks more difficult.

When using asymmetric authentication with encryption, Spartan UltraScale+ FPGAs authenticate all encrypted configuration data before feeding the data into the AES decryptor. This prevents an attacker from loading random unauthenticated data into the AES engine to try to determine the AES key.

Loading an Encrypted Programmable Device Image

The programmable device image (PDI) generation is provided with the Vivado tools. Both encrypted as well as non-encrypted PDI can be generated. For AES PDI encryption, select the option to enable encryption, and specify a 256-bit key as an input to the Vivado PDI generation tool.

After the PDI and key have been generated, the encryption key can be loaded onto a device through the JTAG interface. The Vivado device programmer can accept the key file as an input and program the device with the key through JTAG, using a supported AMD programming cable. The key can be programmed into 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 the PUF or a fixed key that is never visible in the device.

After the device has been programmed with the correct encryption key, the device can be configured with an encrypted PDI. After configuration with an encrypted PDI, it is not possible to read the configuration memory.

While the device holds an encryption key, a non-encrypted PDI 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. The encryption key cannot be read out of the device if a non-encrypted PDI is programmed, preventing the use of Trojan Horse PDI to defeat the FPGA encryption scheme. If the device is provisioned for S-HWRoT, it will only configure from an encrypted PDI.

An encrypted PDI can be delivered through any configuration interface: JTAG, serial, SPI, OSPI, and SelectMAP. PDI 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 SYS_RST instruction. After configuration, the device cannot be reconfigured without resetting the device by toggling the PROGRAM_B, issuing a JTAG instruction SYS_RST, or cycling the power. A mismatch between the key in the encrypted PDI and the key stored in the device causes configuration to fail with the INIT_B pin driving Low.

Physically Unclonable Function

The Spartan UltraScale+ FPGA contains a physically unclonable function (PUF). The PUF creates a signature or fingerprint of each device that is unique to that device. Its value is not known by AMD or the user enabling usage as a key encryption key (KEK). This KEK is a 256 bit key used to encrypt the users red key allowing its storage in black (encrypted) form. The black key can be stored in eFUSEs or the boot header of the PDI.

The PUF also outputs a user accessible unique ID that is cryptographically isolated from the PUF KEK itself despite using the same entropy source. While unique to each device, it is not considered a secret and does not have the same access protections as the KEK itself.

True Random Number Generator

The Spartan UltraScale+ FPGA is the first UltraScale based architecture family to include a true random number generator (TRNG). The TRNG is available for secure, randomized key generation and is available post configuration. The TRNG enables applications to be compliant with AIS-20/31 and NIST-800-90A/B/C standards. Data blocks are generated with a 32-bit wide interface and provides security strength of 256 bits with ring oscillator random sources.

To support these standards, the TRNG operates in three modes:

1. Entropy source output.
2. Internal seed + deterministic random number generator (DRNG) output.
3. External seed + DRNG output.

eFUSE

Spartan UltraScale+ FPGA eFUSEs are nonvolatile one-time-programmable (OTP) cells used for select device settings and user-programmable elements including:

- Programmable device image (PDI) key storage
  - AES-GCM encryption key
  - Authentication key hash
- Factory-programmed Device DNA (96-bits)
- User 32-bit value (`EFUSE_USR` primitive)
- Control and security settings

The fuse link is programmed (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`.

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. For more information, see Device Identifier (Device DNA).

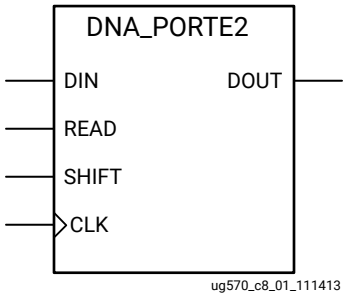
Device Identifier (Device DNA)

The FPGA contains an embedded, device identifier (Device DNA). The identifier is nonvolatile, permanently programmed by AMD 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`) or at registers through the AXI32 primitive.

The FPGA application accesses the identifier value using the device DNA access port (`DNA_PORTE2`) design primitive, shown in the following figure.

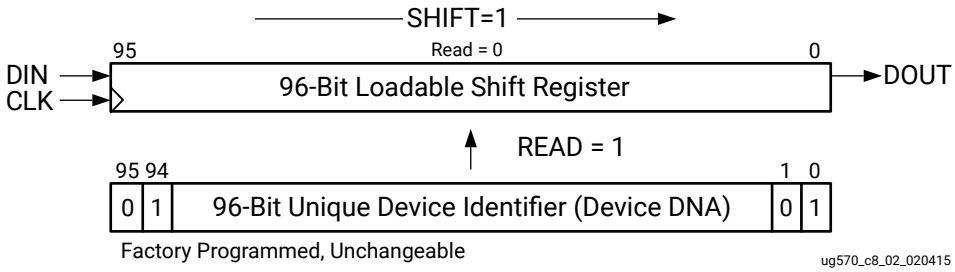
Figure 42: **FPGA DNA_PORTE2 Design Primitive**



Identifier Value and Operation

The following figure shows the general functionality of the DNA_PORTE2 design primitive. An FPGA application must first instantiate the DNA_PORTE2 primitive (shown in Figure 42) within a design. As shown in the following figure, 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 43: **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 the following table. 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 41: **DNA_PORTE2 Operations**

Operation	DIN	READ	SHIFT	CLK	Shift Register	DOUT
HOLD	X	0	0	X	Hold previous value	Hold previous value

Table 41: **DNA_PORTE2 Operations** *(cont'd)*

Operation	DIN	READ	SHIFT	CLK	Shift Register	DOUT
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:

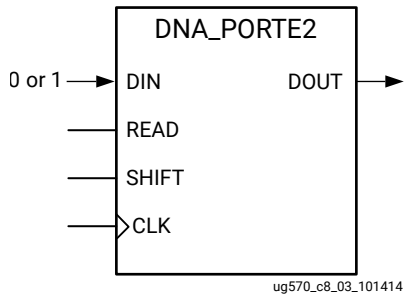
1. X = Do not care.
2. ↑ = Rising clock edge.

To continue reading the identifier values, assert `SHIFT` followed by a rising edge of `CLK`, as shown in Table 41. 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

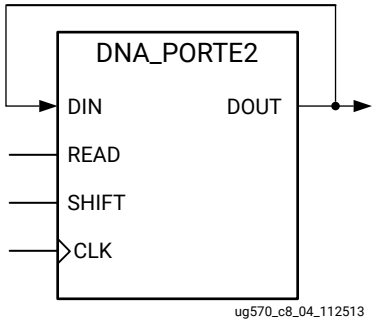
As shown in the following figure, most applications that use the `DNA_PORTE2` primitive tie the `DIN` data input to a static value.

Figure 44: **Shift In Constant**



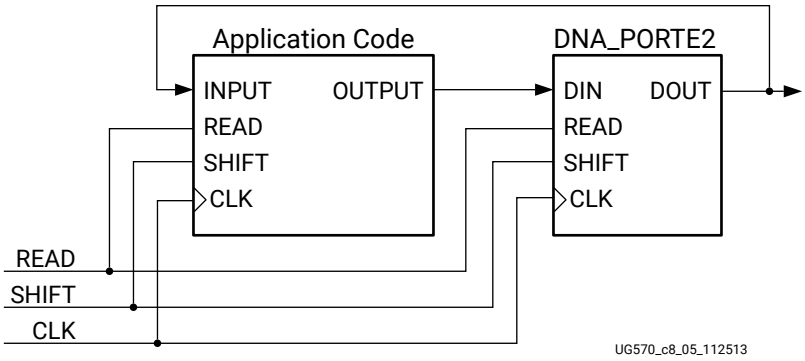
As shown in the following figure, the length of the identifier can be extended by feeding the `DOUT` serial output port back into the `DIN` serial input port. This way, the identifier can be extended to any arbitrary length.

Figure 45: **Circular Shift**



It is also possible to add additional bits to the identifier using FPGA logic resources. As shown in the following figure, 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 46: **Programmable Device Image (PDI) Specific Code**



Chapter 13

Error Management

The AMD Spartan™ UltraScale+™ PMC error aggregator module (EAM) allows the user to respond to specified error conditions. The EAM response options and the BootROM and PLM error codes are described in this section. The EAM responses can be set with the programmable device image (PDI) properties.

Error Aggregator Module

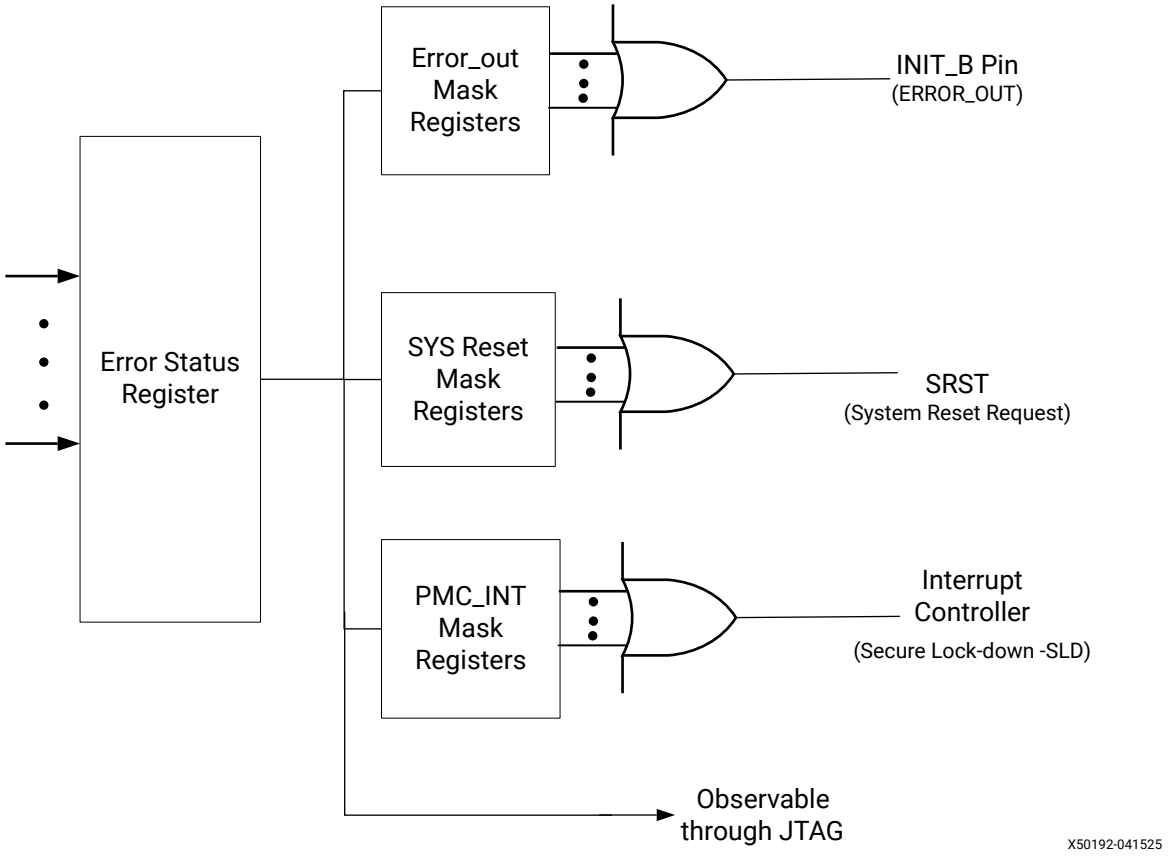
The Spartan UltraScale+ device has a error aggregation module (EAM) that aggregates the errors specified. The EAM then has three responses for handling the non-masked aggregated errors: assert INIT_B pin low, issue a system reset (SRST), or issue an interrupt that enters the secure lock-down (SLD) state. PLM allows for one option response on each error specified.

The following are example error conditions that can cause an EAM response.

- Timeout Error
- AXI Error
- SBI Overflow Error
- eFUSE Error
- Voltage Supply Error
- Temperature Error
- PMC ECC Error
- PL Error
- JTAG Toggle Error
- Tamper Error

The Error Aggregator Module (EAM) figure shows the available error management responses.

Figure 47: **Error Aggregator Module**



Error Codes

Spartan UltraScale+ FPGAs PMC dedicated configuration controller executes the BootROM and PLM. If the BootROM or the PLM detect an issue during the boot and configuration process an error code will be generated in the JTAG ERROR_STATUS register. See the Error Status Register section for details on the bit fields. The error is logged in the format detailed in the following table.

Table 42: **ERROR_STATUS Register BootROM and PLM Error Format**

Register Name	Bit Field	Error Description
ERROR_STATUS	60:55	Reserved
	54:43	First Error
	42:31	Last Error

If a BootROM or PLM error is encountered during programming it can be analyzed and decoded in Vivado using the utility called pdi_debug_util. This tool can perform the error code value look-up, PDI log analysis, and PDI programming and analysis. By using pdi_dbg_util, you can efficiently gather all the necessary information on a programming error to resolve Spartan UltraScale+ FPGA issues. For the latest error code analysis, see the *Debugging PDI Programming with pdi_dbg_util* section in *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

BootROM Error Codes

BootROM unique responsibilities to boot the Spartan UltraScale+ FPGA. If an issue is encountered during the boot process then an error condition can be flagged in the ERROR_STATUS register. The ERROR_STATUS register is accessible via JTAG. The BootROM error conditions are listed in the following table. See PLM Error Codes for error conditions flagged during the PDI boot and configuration.

Table 43: **BootROM Error Codes for SU10P, SU25P, and SU35P Devices**

Error Code	Description
0x0100	eFUSE cache reload timeout failure.
0x0101	eFUSE cache parity error observed even after reload.
0x0104	PL done bit initialization did not get set within configured time.
0x0107	Hardware exception in ROM
0x0120	Indicates hardware exception due to misaligned instruction address.
0x0121	Indicates hardware exception due to instruction access fault.
0x0122	Indicates hardware exception due to illegal instruction.
0x0124	Indicates hardware exception due to load address misaligned.
0x0125	Indicates hardware exception due to load access fault.
0x0126	Indicates hardware exception due to store address misaligned.
0x0127	Indicates hardware exception due to store access fault.
0x012B	Indicates hardware exception due to environment call.
0x012F	Indicates hardware exceptions from PMC dedicated configuration controller.
0x0201	Invalid configuration mode read from BOOT_MODE_USER register. Ensure configuration mode pins are set properly for the requested configuration mode.
0x0202	Image search cannot be done for slave configuration modes.
0x0203	BootROM unable to detect the image/width in SPI_24 configuration mode.
0x0204	BootROM unable to detect the image/width in SPI_32 configuration mode.
0x0205	BootROM unable to initialize Master OSPI configuration mode. Image not found.
0x0206	SelectMAP initialization failed
0x0207	SBI JTAG initialization failed
0x0208	DMA0/DMA1 test failed
0x020B	Master SPI (SPI_24 or SPI_32) MultiBoot value is beyond the search limit.
0x022C	Master OSPI configuration failed
0x022D	Master SPI (SPI_24 or SPI_32) timeout occurred while reading the PDI identification word.

Table 43: **BootROM Error Codes for SU10P, SU25P, and SU35P Devices** *(cont'd)*

Error Code	Description
0x022E	Master SPI (SPI_24 or SPI_32) PDI identification word not matched.
0x0230	The length of the opcode is not suitable for Master SPI (SPI_24 or SPI_32) configuration mode bus width.
0x0231	Slave serial initialization error
0x0233	Secure configuration not allowed in JTAG configuration mode.
0x0234	Flash Limit exceeded while searching for bus width.
0x0300	Boot header does not have a XLNX signature.
0x0301	Boot header checksum not matched.
0x0304	Width word read from device failed. Common for all configuration modes.
0x0305	Decryption with eFUSE only set in eFUSE, and key source is not from eFUSE.
0x0306	Boot header source offset for PLM is overlapping with boot header.
0x0307	Data Partition or total data partition length is crossing the permissible limit of 40.5 KB for PLM. Ensure these are not zero in your programmable device image (PDI).
0x0309	Golden image fallback search not supported for slave configuration modes.
0x030A	Image not found and reached end of SPI flash.
0x030B	Image not found and reached end of OSPI flash.
0x030D	Device read failed during the certificate read. This is common error for all configuration modes.
0x030E	All the PPK revoked through eFUSE. Boot and configuration not allowed.
0x030F	All the IDs revoked through eFUSE. Boot and configuration not allowed.
0x0310	Max 3 PPK choices can be made. Valid choice values as 0,1,2.
0x0311	Chosen PPK is revoked through eFUSE. Boot and configuration not allowed.
0x0312	Revoke ID should be 0-95
0x0313	Chosen Revoke ID is revoked through eFUSE. Boot and configuration not allowed.
0x0314	PPK Hash not matched with any of the three eFUSE locations.
0x0315	PLM length indicate as zero in PDI image
0x0318	PUF helper from boot header not allowed when decryption through eFUSE only set.
0x0319	PLM length is not aligned to 4 B
0x031A	Key source changed from previous to current image and authentication not enabled.
0x031B	Authentication status is changed between previous and current image. Boot and configuration not allowed.
0x031C	Source offset of PMC firmware in image is not aligned to 4B.
0x031D	Total PMC firmware length is not aligned to 4B.
0x031E	Verifying the PPK used has a non-zero hash in eFUSE
0x0324	Master SPI (SPI_24 or SPI_32) or Master OSPI: Device not showing idle status after completion of stig read.
0x0325	Master SPI (SPI_24 or SPI_32) or Master OSPI: Idle check failed after the DMA operation.
0x0326	Master SPI (SPI_24 or SPI_32) or Master OSPI: DMA read operation not completed within timeout.
0x0327	Data not received from slave serial or slave selectMAP host within timeout.
0x0328	SelectMAP indicated an abort operation.
0x0329	Total PLM length is less than PLM length in boot header, which is not allowed.

Table 43: **BootROM Error Codes for SU10P, SU25P, and SU35P Devices** *(cont'd)*

Error Code	Description
0x032A	Source offset of PLM in flash is beyond search limit.
0x032B	Data not received or not indicated that it is user JTAG configuration mode within time out.
0x032D	Source offset beyond 16 MB for SPI_24 configuration mode.
0x032F	Slave serial read error
0x0330	Slave serial overflow error
0x0331	Total PLM length is greater than expected over head when authentication/encryption/integrity is enabled.
0x0332	Boot header hash mismatch error
0x0333	Error indicates unable to read the hash block data from PDI.
0x0334	Error indicates unable to read the PPK key from PDI image.
0x0335	SPK header read error
0x0336	Error indicates unable to read the SPK key from PDI image.
0x0337	Error indicates unable to read the SPK signature from PDI image.
0x0338	Hash block signature read error
0x0339	Error indicates boot header hash block size incorrect.
0x033A	Actual PPK/SPK size error
0x033B	Actual PDI signature size error
0x033C	Total PPK/SPK size error
0x033D	Total PDI signature size error
0x033E	Total SPK sign size error
0x033F	Actual SPK sign size error
0x0346	SPK signature verification failed
0x0347	Hash block signature verification failed
0x0348	PUF image is detected
0x0349	Failed to read the hashblock GCM tag from PDI image.
0x034A	Invalid PUF helper data
0x034B	PUF HD not programmed
0x0388	Secure JTAG configuration is not allowed.
0x0389	DPA CM disabled through eFUSE and trying to enable through boot header.
0x03A0	In case of JTAG, secure configurationnot allowed error when authentication/encryption enabled.
0x03A1	DPA CM disabled through eFUSE and trying to enable through boot header.
0x03B2	SHA init error while performing operation over PPK.
0x03B3	SHA digest error while performing operation over PPK.
0x03C2	SHA init error while performing operation over BH.
0x03C3	SHA digest error while performing operation over BH.
0x03D0	AES AAD error
0x03D5	DMA wait for done error while transferring AES AAD data.
0x03D7	DMA wait for done error while transferring GCM tag.
0x03D8	AES wait for done error

Table 43: **BootROM Error Codes for SU10P, SU25P, and SU35P Devices** *(cont'd)*

Error Code	Description
0x03D9	GCM tag mismatch error
0x0400	Invalid address for register initialization.
0x0401	Device read error after register initialization.
0x0402	Boot header does not match original after register initialization.
0x0500	Key clear failed at fallback
0x050B	Family key lock error after using it to decrypt obfuscated key.
0x050C	Family key unlock error while using it to decrypt obfuscated key.
0x050D	DMA timeout occurred during the AAD operation.
0x050E	Key source is invalid
0x050F	PUF command not valid
0x0510	DMA done not asserted after pushing the KEK to AES engine within time out.
0x0511	Fetch Key Source value for eFUSE error.
0x0512	KEK key load failed into AES
0x0513	KEK IV load failed into AES
0x0514	KEK key source value fetch error
0x0515	PLM IV criteria not matched
0x052B	SSS configuration error for AES engine.
0x05BC	SHA init error
0x05BD	SHA start error
0x05BE	SHA init error while calculating hash on PLM.
0x05BF	SHA start error while calculating hash on PLM.
0x05C0	SHA error while updating PLM as data.
0x05C1	SHA error while updating PPK as data.
0x05C2	SHA error while updating SPK as data.
0x05C3	SHA error while updating SPK ID as data.
0x05C4	SHA finish error while calculation hash on PLM.
0x0702	Word read not asserted by PUF during regeneration.
0x0703	Key not ready within timeout by PUF.
0x0704	Key not converged during regeneration.
0x0705	Regeneration mode is not matching with either 4 Kb.
0x0706	PUF zeroization failed
0x0707	Invalid PUF command
0x0708	SHA invalid mode error
0x0709	SHA not initialized error
0x070A	SHA not started error
0x070B	SSS configuration error
0x070C	SHA invalid parameter error
0x070D	SHA not update error while checking state of SHA.
0x070E	Secure Lockdown triggered

Table 43: **BootROM Error Codes for SU10P, SU25P, and SU35P Devices** *(cont'd)*

Error Code	Description
0x070F	Secure Lockdown completed
0x0710	Mem Clear failed during Secure Lockdown.
0x0711	Mem Clear operation timed out during Secure Lockdown.
0x0712	Scan Clear Failed during Secure Lockdown.
0x0713	Scan Clear operation timed out during Secure Lockdown.
0x0714	Key Zeroize failed during Secure Lockdown.

Table 44: **BootROM Error Codes for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P Devices**

Error Code	Description
0x0100	eFUSE cache reload timeout failure.
0x0101	eFUSE cache parity error observed even after reload.
0x0103	INIT_B polling error
0x0104	This error comes when PL done bit initialization did not get set within configured time.
0x0107	Hardware exception in ROM
0x0120	Indicates hardware exception due to misaligned instruction address.
0x0121	Indicates hardware exception due to instruction access fault.
0x0122	Indicates hardware exception due to illegal instruction.
0x0124	Indicates hardware exception due to load address misaligned.
0x0125	Indicates hardware exception due to load access fault.
0x0126	Indicates hardware exception due to store address misaligned.
0x0127	Indicates hardware exception due to store access fault.
0x012B	Indicates hardware exception due to environment call.
0x012F	Indicates hardware exceptions from PMC dedicated configuration controller.
0x0201	Invalid configuration mode read from BOOT_MODE_USER register. Ensures configuration mode pins are set properly for requested configuration mode.
0x0202	Image search cannot be done for slave configuration modes.
0x0203	BootROM unable to detect the image/width in SPI_24 configuration mode.
0x0204	BootROM unable to detect the image/width in SPI_32 configuration mode.
0x0205	BootROM unable to initialize Master OSPI configuration mode. Image not found.
0x0206	SelectMAP initialization failed
0x0207	SBI JTAG initialization failed
0x0208	DMA0/DMA1 test failed
0x020B	Master SPI (SPI_24 or SPI_32) MultiBoot values are beyond the search limit.
0x0211	Master OSPI: MultiBoot value is beyond the search limit.
0x022C	Master OSPI configuration failed
0x022D	Master SPI (SPI_24 or SPI_32) timeout occurred while reading the PDI identification word.
0x022E	Master SPI (SPI_24 or SPI_32) PDI identification word not matched.

Table 44: **BootROM Error Codes for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P Devices** *(cont'd)*

Error Code	Description
0x0230	Error indicates the length of the opcode is not suitable for Master SPI (SPI_24 or SPI_32) bus width.
0x0231	Slave serial initialization error
0x0233	Secure configuration not allowed in JTAG
0x0234	Flash Limit exceeded while searching for Bus Width.
0x0235	Configuration attempted with a disabled configuration interface.
0x0300	XLNX signature not found in the PDI image.
0x0301	Boot header check sum not matched.
0x0304	Width word read from device failed. Common for all configuration modes.
0x0305	Decryption with eFUSE only set in eFUSE and key source is not from eFUSE.
0x0306	Boot header source offset is overlapping with boot header.
0x0307	Invalid PLM Length in the Boot Header. Max Allowed PLM Length is 72.5KB.
0x0309	Golden image search not supported for slave boot modes.
0x030A	Image not found and reached end of SPI flash.
0x030B	Image not found and reached end of OSPI flash.
0x030D	Device read failed during the certificate read. This is common error for all configuration modes.
0x030E	All the PPK revoked through eFUSE. Boot and configuration not allowed.
0x030F	All the IDs revoked through eFUSE. Boot and configuration not allowed.
0x0310	Max 2 PPK choices can be made. Valid choice values as 0;1.
0x0311	Chosen PPK is revoked through eFUSE. Boot and configuration not allowed.
0x0312	Revoke ID should be 0-95
0x0313	Chosen Revoke ID is revoked through eFUSE. Boot and configuration not allowed.
0x0314	PPK Hash not matched with any of the three eFUSE locations.
0x0315	PLM length indicate as zero in PDI image.
0x0318	PUF helper from boot header not allowed when decryption through eFUSE only set.
0x0319	PMC Firmware length is not aligned to 4B.
0x031A	Key source changed from previous to current image and authentication not enabled.
0x031B	Authentication status is changed between previous and current image. Boot and configuration not allowed.
0x031C	Source offset of PLM in image is not aligned to 4B.
0x031D	Total PMC Firmware length is not aligned to 4B.
0x031E	Ensure that the PPK used has a non-zero hash in eFUSE.
0x0323	PUFHD Length is invalid
0x0324	OSPI device not showing idle status after completion of stig read.
0x0325	OSPI idle check failed after the DMA operation.
0x0326	OSPI DMA read operation not completed within timeout.
0x0327	Data not received from slave serial or slave selectMAP host within timeout.
0x0328	SelectMAP configuration mode indicated an abort operation.
0x0329	Total PLM length is less than PLM length in boot header, which is not allowed.

Table 44: **BootROM Error Codes for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P Devices** *(cont'd)*

Error Code	Description
0x032A	Source offset of PLM in flash is beyond search limit.
0x032B	Data not received or not indicated that it is user JTAG configuration mode within time out.
0x032D	Source offset beyond 16 MB for SPI_24 configuration mode.
0x032F	Slave serial read error
0x0330	Slave serial overflow error
0x0331	Total PLM length is greater than expected over head when authentication/encryption/integrity is enabled.
0x0332	Boot header hash mismatch error
0x0333	Error indicates unable to read the hash block data from PDI.
0x0334	Error indicates unable to read the PPK key from PDI image.
0x0335	SPK header read error
0x0336	Error indicates unable to read the SPK key from PDI image.
0x0337	Error indicates unable to read the SPK signature from PDI image.
0x0338	Hash block signature read error.
0x0339	Error indicates boot header hash block size incorrect.
0x033A	Actual PPK/SPK size error
0x033B	Actual PDI signature size error
0x033C	Total PPK/SPK size error
0x033D	Total PDI signature size error
0x033E	Invalid Total SPK Signature Size in ECDSA Authentication.
0x033F	Invalid Actual SPK Signature Size in ECDSA Authentication.
0x0340	Invalid Total SPK Signature Size in HSS Authentication.
0x0341	Invalid Total Actual Signature Size in HSS Authentication.
0x0342	Invalid Total SPK Signature Size in LMS Authentication.
0x0343	Invalid Total Actual Signature Size in LMS Authentication.
0x0349	SPK signature verification failed
0x034A	Hash block signature verification failed.
0x034B	PUF image is detected
0x034C	Failed to read the hashblock GCM tag from PDI image
0x034D	Invalid PUF helper data
0x034E	PUF HD not programmed
0x034F	LMS OTS Checksum - buffer for digest is of invalid length (0).
0x0350	LMS OTS type not supported
0x0351	LMS OTS parameter look up; glitch detected.
0x0352	LMS OTS Invalid parameter
0x0353	LMS OTS Invalid digest per digit buffer.
0x0354	LMS OTS Invalid Signature buffer.
0x0355	LMS OTS Signature Verification - when signature to be used is at invalid address

Table 44: **BootROM Error Codes for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P Devices** *(cont'd)*

Error Code	Description
0x0356	LMS OTS Signature Verification - invalid address to buffer to pass back computed public key
0x0357	LMS OTS Signature Verification: OTS Sign len less than or equal to 4B.
0x0358	LMS OTS Signature Verification: OTS Sign type mismatch with public key.
0x0359	LMS OTS Signature Verification: parameter lookup failed (not supported type).
0x035A	LMS OTS Signature Verification: OTS Sign len less than required and more than 4B.
0x035B	LMS OTS Signature Verification: sha init failed during digest calculation.
0x035C	LMS OTS Signature Verification: Digest calculation operation failed.
0x035D	LMS Message Digest: checksum calculation failed during digest calculation.
0x035E	LMS OTS Signature Verification: hash chain per iteration SHA digest failed.
0x035F	LMS OTS Signature Verification: overall concatenation SHA digest failed.
0x0360	LMS Message Digest: Invalid address for instance pointer.
0x0361	LMS Message Digest: SHA start failed during digest calculation.
0x0362	LMS Message Digest: SHA update failed during digest calculation with I.
0x0363	LMS Message Digest: SHA update failed for initial blocks.
0x0364	LMS Message Digest - SHA update failed during for remaining data.
0x0365	LMS Message Digest - SHA finish failed during digest calculation.
0x0366	LMS Type not supported
0x0368	LMS Signature Verification: Invalid Parameters
0x0369	LMS Signature Verification: Invalid OTS pointer parameter.
0x036A	LMS Signature Verification: Invalid SHA pointer parameter.
0x036B	LMS Signature Verification: Invalid DMA pointer parameter.
0x036C	LMS Signature Verification: Invalid digest prefix parameter.
0x036D	LMS Signature Verification: Invalid digest checksum buffer.
0x036E	LMS Signature Verification: Invalid temp buffer
0x036F	LMS Signature Verification: Invalid Public key temp buffer.
0x0370	LMS Signature Verification: invalid address for expected public key at Merkle tree root.
0x0371	LMS Signature Verification: invalid len for expected public key at Merkle tree root less than or equal to 4.
0x0372	LMS Signature Verification: LMS public key parameter pick up failed (unsupported type).
0x0373	LMS Signature Verification: LMS signature buffer invalid address.
0x0374	LMS Signature Verification: invalid len for expected public key at Merkle tree root (total less than required by parameters).
0x0375	LMS Signature Verification: LMS signature length less than or equal 4B.
0x0376	LMS OTS type mismatch between LMS OTS signature and public key.
0x0377	LMS Signature Verification: LMS OTS parameters look up failed (unsupported type).
0x0378	LMS Signature Verification: LMS signature length less than required for LMS OTS signature.
0x0379	LMS Signature Verification: LMS type mismatch between LMS signature and public key.
0x037A	LMS Signature Verification: LMS signature parameters lookup failed (unsupported type).
0x037B	LMS Signature Verification: invalid node number 'q' in a Merkle tree.

Table 44: **BootROM Error Codes for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P Devices** *(cont'd)*

Error Code	Description
0x037C	LMS Signature Verification: LMS signature length error - (total len less than required by parameters).
0x037D	LMS Signature Verification: LMS OTS signature verification failed.
0x037E	LMS Signature Verification: LMS signature to public key (leaf node sha digest failed).
0x037F	LMS Signature Verification: LMS signature to public key - odd internal node SHA digest failed.
0x0380	LMS Signature Verification: LMS signature to public key - even internal node SHA digest failed.
0x0381	LMS Signature Verification: calculated LMS public key did not match with expected. Authentication failed.
0x0382	LMS Signature Verification: LMS public key comparison glitch detected. Authentication failed.
0x0383	LMS Signature Verification: CFI check failed.
0x0384	LMS HSS Init: HSS Invalid Parameters.
0x0385	LMS HSS Init: HSS Invalid key buffer.
0x0386	LMS HSS Init: HSS Invalid LMS instance.
0x0387	LMS HSS Init: HSS invalid OTS instance.
0x0388	LMS HSS Init: HSS public key at an invalid address.
0x0389	LMS HSS Init: HSS pub key len less than required.
0x038A	LMS HSS Init: HSS pub key's LMS OTS type parameter look up for level 1 tree - temporal glitch detected.
0x038B	LMS HSS Init: only two levels of Merkle trees are supported.
0x038C	LMS HSS Init: HSS pub key & Signature levels mismatch.
0x038D	LMS HSS Init: HSS pub key's LMS type parameter look up failed for level 0 tree.
0x038E	LMS HSS Init: HSS pub key's LMS OTS type parameter look up failed for level 0 tree.
0x038F	LMS HSS Signature verification: HSS signature len does not fit OTS signature for level 0.
0x0390	LMS HSS Signature verification: Level 0 LMS auth op failed.
0x0391	LMS HSS Signature verification: CFI Failed.
0x0392	LMS HSS Signature verification: authenticated public key for level 1 tree copy failed.
0x0393	LMS HSS Signature verification: authenticated public key for level 1 tree copy failed - iteration mismatch.
0x0394	LMS HSS Signature verification: HSS pub key's LMS type parameter look up failed for level 1 tree - in HSS init.
0x0395	LMS HSS Signature verification: HSS pub key's LMS type parameter look up failed for level 1 tree - in HSS Finish.
0x0396	LMS HSS Signature verification: SHA init failed.
0x0397	LMS HSS Signature verification: Invalid parameters.
0x0398	LMS HSS Signature verification: Invalid signature buffer.
0x0399	LMS HSS Signature verification: Invalid length mismatch.
0x039A	LMS HSS Signature verification: HSS pub key's LMS type 2 parameter look up failed for level 1 tree - in HSS Finish.
0x039B	LMS HSS Signature verification: HSS pub key's LMS OTS type parameter look up failed for level 1 tree.
0x039C	LMS HSS Signature verification: OTS signature at level 1 is less than 4B.

Table 44: **BootROM Error Codes for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P Devices** *(cont'd)*

Error Code	Description
0x039D	LMS HSS Signature verification: HSS signature len does not fit OTS signature for level 1.
0x039E	LMS HSS Signature verification: Level 1 LMS auth op failed.
0x039F	LMS HSS Finish CFI error
0x03A0	Secure JTAG boot is not allowed
0x03A1	DPA CM disabled through eFUSE and trying to enable through boot header.
0x03B2	SHA init error while performing operation over PPK.
0x03B3	SHA digest error while performing operation over PPK.
0x03C2	SHA init error while performing operation over BH.
0x03C3	SHA digest error while performing operation over BH.
0x03D0	AES AAD error
0x03D5	DMA wait for done error while transferring AES AAD data.
0x03D7	DMA wait for done error while transferring GCM tag.
0x03D8	AES wait for done error
0x03D9	GCM tag mismatch error
0x03DA	Boot Header copy glitch error
0x03DB	ECDSA signature R component zero
0x03DC	ECDSA signature S component zero
0x03DD	ECDSA signature R component ECC order error
0x03DE	ECDSA signature S component ECC order error
0x03DF	ECDSA public key validation failed
0x03E0	ECDSA signature verification failed
0x0400	Not an allowed address for register initialization.
0x0401	Device read failed after register initialization.
0x0402	Boot header is not matching with original after register initialization.
0x0500	Key Clear failed at Fallback
0x050B	Family key lock error after using it to decrypt obfuscated key.
0x050C	Family key unlock error while using it to decrypt obfuscated key.
0x050D	DMA timeout occurred during the AAD operation.
0x050E	Key source is invalid
0x050F	PUF command not valid
0x0510	DMA done not asserted after pushing the KEK to AES engine with in time out.
0x0511	Fetch Key Source value for eFUSE error.
0x0512	KEK key load failed into AES
0x0513	KEK IV load failed into AES
0x0514	KEK Key Source value fetch error
0x0515	PLM IV criteria not matched
0x052B	SSS configuration error for AES engine
0x05BC	SHA init error

Table 44: **BootROM Error Codes for SU45P, SU55P, SU60P, SU65P, SU100P, SU150P, and SU200P Devices** *(cont'd)*

Error Code	Description
0x05BD	SHA start error
0x05BE	SHA init error while calculating hash on PLM.
0x05BF	SHA start error while calculating hash on PLM.
0x05C0	SHA Update error while calculating hash on PLM.
0x05C1	SHA Update error while calculating hash on PPK.
0x05C2	SHA Update error while calculating hash on SPK.
0x05C3	SHA Update error while calculating hash on SPK ID.
0x05C4	SHA finish error
0x0702	Word read not asserted by PUF during regeneration.
0x0703	Key not ready within timeout by PUF.
0x0704	Key not converged during regeneration.
0x0705	Regeneration mode is not matching with either 12K/4K.
0x0706	PUF zeroization failed
0x0707	Invalid PUF command
0x0708	SHA invalid mode error
0x0709	SHA not initialized error
0x070A	SHA not started error
0x070B	SSS configuration error
0x070C	SHA invalid parameter error
0x070D	SHA not update error while checking state of SHA.
0x070E	Secure lockdown triggered
0x070F	Secure lockdown completed
0x0710	Mem clear failed during secure lockdown.
0x0711	Mem clear operation timed out during secure lockdown.
0x0712	Scan clear failed during secure lockdown.
0x0713	Scan clear operation timed out during secure lockdown.
0x0714	Key zeroize failed during secure lockdown.

PLM Error Codes

The following table lists PLM error codes that can be flagged during boot and configuration. The PLM error codes start above 0x0900. See BootROM Error Codes table for BootROM error conditions.

Table 45: **PLM Error Codes**

Error Code	Name	Description
0x0901	XPLM_ERR_DMA_LOOKUP	DMA driver lookup fails.

Table 45: **PLM Error Codes** *(cont'd)*

Error Code	Name	Description
0x0902	XPLM_ERR_DMA_CFG	DMA driver config fails.
0x0903	XPLM_ERR_DMA_SELFTEST	DMA Self-test fails. Occurs when DMA is in reset and PLM tries to initialize it.
0x0904	XPLM_ERR_DMA_XFER_WAIT	DMA transfer failed waiting for done.
0x0905	XPLM_ERR_CMD_APIID	Tried to process unregistered API ID.
0x0906	XPLM_ERR_CMD_HANDLER_NULL	Handler is not registered for the given CDO cmd.
0x0907	XPLM_ERR_MODULE_NOT_REGISTERED	The given CDO cmd is not registered.
0x0908	XPLM_ERR_INVLD_RESUME_HANDLER	The handler is not assigned.
0x0909	XPLM_ERR_INIT_CDO_INSTANCE	Failed to initialize the CDO instance.
0x090A	XPLM_ERR_CDO_HDR_ID	Valid CDO header ID is not present in CDO header. Can be triggered when a different partition type is processed as CDO.
0x090B	XPLM_ERR_CDO_CHECKSUM	CDO header checksum is wrong. Can be triggered when CDO header is corrupted.
0x090C	XPLM_ERR_MEMCPY_CMD_EXEC	Error copying CDO CMD to buffer.
0x090D	XPLM_ERR_INVALID_BREAK_LENGTH	The break length required to jump is less than the processed CDO length.
0x090E	XPLM_ERR_UNSUPPORTED_OSPI_FLASH_MAKE	SPI or OSPI flash make is not supported.
0x090F	XPLM_ERR_UNSUPPORTED_OSPI_SIZE	SPI or OSPI flash size is not supported.
0x0910	XPLM_ERR_OSPI_CFG	SPI/OSPI driver lookup fails.
0x0911	XPLM_ERR_OSPI_INIT	OSPI driver CFG fails.
0x0912	XPLM_ERR_OSPI_SEL_FLASH_CS0	OSPI driver is unable to select flash CS0. Check minor code for OSPI driver error code.
0x0913	XPLM_ERR_OSPI_READ_ID	Failed to read OSPI flash ID.
0x0914	XPLM_ERR_OSPI_READ_DATA	Failed to read the data from flash.
0x0915	XPLM_ERR_OSPI_SET_DDR_WRITE_ENABLE	Failed to enable write mode in flash.
0x0916	XPLM_ERR_OSPI_SET_DDR_WRITE_CFG_REG	Failed to configure DDR mode in flash.
0x0917	XPLM_ERR_OSPI_SET_DDR_DUAL_BYTE_OP_ENABLE	Failed to enable dual byte op code.
0x0918	XPLM_ERR_OSPI_SET_DDR_PHY	Failed to set the controller to DDR PHY mode
0x0919	XPLM_ERR_OSPI_SET_DDR_READ_CFG_REG	Failed to readback configuration from flash.
0x091A	XPLM_ERR_OSPI_SET_DDR_CFG_MISMATCH	Failed to validate configuration after enabling DDR PHY mode.
0x091B	XPLM_ERR_OSPI_DUAL_BYTE_OP_DISABLE	Failed to disable DUAL BYTE OP code.
0x091C	XPLM_ERR_OSPI_SINGLE_CONN_MODE	OSPI connection mode is other than single.
0x091D	XPLM_ERR_OSPI_SET_SDR_NON_PHY	Failed to set the controller to SDR NON PHY mode
0x091E	XPLM_ERR_OSPI_SET_SDR_PHY	Failed to set the controller to SDR PHY mode.
0x091F	XPLM_ERR_OSPI_COPY_LENGTH_OVERFLOW	Source address in OSPI copy exceeds flash size.
0x0920	XPLM_ERR_OSPI_4B_ADDR_MODE_WRITE_ENABLE	Failed to enable write mode in flash.
0x0921	XPLM_ERR_OSPI_4B_ADDR_MODE_ENTER_OR_EXIT_CMD	Failed to enter or exit 4 byte addressing mode.

Table 45: **PLM Error Codes** *(cont'd)*

Error Code	Name	Description
0x0922	XPLM_ERR_OSPI_4B_ADDR_MODE_ENTER_OR_EXIT_CMD_STATUS	Failed to verify 4 byte addressing mode.
0x0923	XPLM_ERR_OSPI_4B_ADDR_MODE_WRITE_DISABLE	Failed disable write mode in flash.
0x0924	XPLM_ERR_AES_KEY_CLEAR_TIMEOUT	Failed to clean AES key.
0x0925	XPLM_ERR_RTCA_OSPI_CLK_DIV_MAX	The ref clk divisor in RTCA sets more than 6-bits.
0x0926	XPLM_ERR_RTCA_OSPI_INVLD_DDR_PHY_CFG	OSPI configuration is set to DDR and NON_PHY mode.
0x0927	XPLM_ERR_QSPI_READ_ID	Failed to read SPI x4 flash ID.
0x0928	XPLM_ERR_QSPI_FLASH_MAKE_NOT_SUPPORTED	SPI x4 flash make is not supported.
0x0929	XPLM_ERR_QSPI_FLASH_SIZE_NOT_SUPPORTED	SPI x4 flash size is not supported.
0x092A	XPLM_ERR_QSPI_CFG_NOT_FOUND	SPI x4 driver lookup fails.
0x092B	XPLM_ERR_QSPI_CFG_INIT	SPI x4 driver configuration fails.
0x092C	XPLM_ERR_QSPI_FLASH_CS	SPI x4 driver failed to select flash.
0x092D	XPLM_ERR_QSPI_SINGLE_CONN_MODE	SPI x4 connection mode is other than single.
0x092E	XPLM_ERR_QSPI_COPY_LENGTH_OVERFLOW	Source address in OSPI copy exceeds flash size.
0x092F	XPLM_ERR_QSPI_READ	Failed to read the data from flash.
0x0930	XPLM_ERR_QSPI_LENGTH	SPI x4 read length is greater than flash size
0x0931	XPLM_ERR_QSPI_BUS_WIDTH_NOT_SUPPORTED	SPI x4 bus width is not supported.
0x0932	XPLM_ERR_QSPI_READ_BANK_SEL	Failed to select the flash bank for read.
0x0933	XPLM_ERR_QSPI_BANK_SEL	Failed to select the flash bank
0x0934	XPLM_ERR_QSPI_BANK_SEL_SEND_WREN	Failed to enable write mode during flash bank select.
0x0935	XPLM_ERR_QSPI_BANK_SEL_SEND_EXT_ADDR	Failed to extended address write command for Infineon flash make.
0x0936	XPLM_ERR_QSPI_BANK_SEL_VERIFY_EXT_ADDR	Failed to verify the extended address write cmd.
0x0937	XPLM_ERR_RTCA_QSPI_INVLD_DDR_CFG	SPI x4 configuration in RTCA is set to DDR mode.
0x0938	XPLM_ERR_IO_MOD_INIT	Error if failed to initialize IO module.
0x0939	XPLM_ERR_ASSERT	Assertion error
0x093A	XPLM_ERR_PPDI_ROM_INST_ZEROIZE	Failed to initialize ROM instance to zero.
0x093B	XPLM_ERR_PPDI_BH_ZEROIZE	Failed to initialize boot header table to zero.
0x093C	XPLM_ERR_PPDI_SBI_BUF_READ_WIDTH_WORD	Failed to read boot header from SBI buffer.
0x093D	XPLM_ERR_PPDI_SBI_BUF_READ_BH	Failed to read boot header from SBI buffer.
0x093E	XPLM_ERR_CALC_BH_CHECKSUM	Failed to validate Boot Header checksum.
0x093F	XPLM_ERR_HASH_BLOCK_ZEROIZE	Failed to initialize hash block to zero.
0x0940	XPLM_ERR_HASH_BLOCK_READ_SBI_BUF	Failed to read hash block from SBI buffer.
0x0941	XPLM_ERR_GCM_TAG_READ_SBI_BUF	Failed to read GCM TAG from SBI buffer.
0x0942	XPLM_ERR_HASH_BLOCK_AUTHENTICATION	Failed to authenticate Hash block.

Table 45: **PLM Error Codes** *(cont'd)*

Error Code	Name	Description
0x0943	XPLM_ERR_PPDI_INVALID_IMG_DET_WORD	Failed to validate "XLNX" image identification word.
0x0944	XPLM_ERR_INVALID_SECONDARY_BOOT_INTF	Secondary boot interface configured it not supported.
0x0945	XPLM_ERR_PPDI_DEC_EFUSE_ONLY	Error if BH encryption status is other than @ref XPLM_ENC_STATUS_eFUSE_PUF_KEK.
0x0946	XPLM_ERR_PPDI_INVALID_BH_ENC_STATUS	Encryption source or type is not supported.
0x0947	XPLM_ERR_PPDI_AHWROT_UNSIGNED_IMG	Failed to validate A-HWROT.
0x0948	XPLM_ERR_PPDI_SEC_TRANSITION_AUTH	Partial PDI authentication status is different from full PDI authentication status.
0x0949	XPLM_ERR_PPDI_SEC_TO_SEC_KEY_MISMATCH	Partial PDI encryption source or type does not match with full PDI.
0x094A	XPLM_ERR_PPDI_FAMILY_KEY_NOT_ALLOWED	Family key is used to encrypt partial PDI.
0x094B	XPLM_ERR_PPDI_TEMPORAL_ERR_AUTH_ENC_STATUS	Failed to validate encryption or authentication status.
0x094C	XPLM_ERR_PPDI_PUF_DISABLED	Error if PUF is disabled.
0x094D	XPLM_ERR_PPDI_PMCFWLEN_NON_ZERO	Partial PDI contains PLM firmware.
0x094E	XPLM_ERR_PPDI_CDO_LEN_ALIGN	CDO length is not byte aligned.
0x094F	XPLM_ERR_PPDI_PUF_IMG_NOT_SUPPORTED	PUF image ID is set for partial PDI.
0x0950	XPLM_ERR_PPDI_PUF_HD_BUFF_ZEROIZE	Failed to initialize PUF HD to zero.
0x0951	XPLM_ERR_PUF_HD_DIGEST_VALIDATION	Failed to validate PUF HD.
0x0952	XPLM_ERR_INIT_CHUNK_INST	Failed to initialize chunk instance.
0x0953	XPLM_ERR_SECURE_CLR	Failed to reset crypto engines.
0x0954	XPLM_ERR_PPDI_INVLD_PUF_DIGEST	Failed to validate PUF digest for partial PDI.
0x0955	XPLM_ERR_GLITCH_DETECTED	Glitch detected
0x0956	XPLM_ERR_URAM_CLR_BUSY_TIMEOUT	URAM clear not finished.
0x0957	XPLM_ERR_CDO_LENGTH_OVERFLOW	The length of the remaining CDO commands exceeds the expected limit (@ref XPLM_FINISH_CDO_READ_REMAINING_CMD_MAX_LENGTH) after detecting the 'finish_cdo_read' CDO command.
0x0958	XPLM_ERR_CDO_MOVE_FAILURE	Moving the remaining CDO commands to the top of the chunk buffer fails after detecting the 'finish_cdo_read' CDO command.
0x0959	XPLM_ERR_OSPI_SET_CLK_PRESCALE_4	Failed to set the prescalar for the OSPI clock.
0x095A	XPLM_ERR_OSPI_IDAC_EN_OPTION	Failed to enable IDAC controller in OSPI.
0x095B	XPLM_ERR_SEC_LOCKDOWN	Secure lockdown error for non-tamper events.
0x0C00	XPLM_ERR_FEATURE_NOT_SUPPORTED	Feature cmd: the CMD ID is not supported.
0x0C01	XPLM_ERR_MASK_POLL_INVLD_CMD_LEN	mask_poll cmd: The command length exceeds the maximum supported.
0x0C02	XPLM_ERR_MASK_POLL_TIMEOUT	mask_poll cmd: Timeout while expecting for a value.
0x0C11	XPLM_ERR_IDCODE	PLM IDCODE check error.
0x0C50	XPLM_ERR_RDBK_DISABLED	cframe_read cmd: Readback is disabled in secure boot.

Table 45: **PLM Error Codes** *(cont'd)*

Error Code	Name	Description
0x0C51	XPLM_ERR_RDBK_INVALID_INFR_SEL	cframe_read cmd: Invalid interface type is selected for readback.
0x0C52	XPLM_ERR_RDBK_PAYLOAD_TO_CCU	cframe_read cmd: Failed to transfer CFI read cmd paylod to CCU.
0x0C53	XPLM_ERR_RDBK_CCU_READ	cframe_read cmd: Failed to readback the bitstream from CCU.
0x0C54	XPLM_ERR_RDBK_READ_TIMEOUT	cframe_read cmd: Failed to read the readback data from the SBI interface.
0x0C55	XPLM_ERR_SET_MEM	set cmd: Failed to set the memory with the value.
0x0C56	XPLM_ERR_KEYHOLE_XFER	write_keyhole cmd: Failed to transfer the payload to CCU.
0x0C57	XPLM_ERR_MAX_NESTED_BEGIN	begin cmd: The number of nested begin exceeds the limit of 10.
0x0C58	XPLM_ERR_STORE_END_OFFSET	begin cmd: Failed to store the end offset of begin command.
0x0C59	XPLM_ERR_INVLD_BEGIN_END_PAIR	end cmd: Begin and end cmds are not paired.
0x0C5A	XPLM_ERR_INVLD_END_ADDR	end/break cmd: End address is not valid.
0x0C5B	XPLM_ERR_REG_READ_LEN_LIMIT	The register read length exceeds the supported limit of 128 registers.
0x0C5C	XPLM_ERR_REG_READ_DMA_XFER	The DMA transfer to SBI failed.
0x0C5D	XPLM_ERR_REG_READ_TIMEOUT	The data placed in SBI buffer is not read back.
0x0C5E	XPLM_ERR_REG_READ_ADDR_INVALID	The register address is not within the limits.

Chapter 14

Configuration Debugging

Introduction

Best practices are covered in this chapter to help to 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
- Initial Debug Steps

File Generation Review

Ensure the PDI properties and flash programming file options were implemented correctly for the targeted configuration mode. To verify the PDI options used by an image run the Tcl command:

```
report_property -all [current_design]
```

This command displays all the properties applied to the design. The default is applied when there are no values displayed. All DRC warnings received during PDI 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 pins are the `INIT_B` and the `DONE`. `INIT_B` rising from Low to High indicates the completion of initialization after power-up, and then the falling `INIT_B` signal later in the process can indicate a CRC error. `DONE` indicates a successful configuration of the device. 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 include:

Mode Pins

The Mode pins `M[2:0]` should be tied and static during configuration. The FPGA reads the modes pins at power-up to determine which configuration mode to use. JTAG mode is most commonly used for debugging. In Spartan UltraScale+ you must set the mode to JTAG if you want to configure in this configuration mode. If a non-JTAG mode is selected at power-up, the system must wait for a configuration attempt (either successful or failure) before the user can send attempt a JTAG configuration or the user must issue a JTAG instruction JPROG.

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.

INIT_B Pin

The `INIT_B` pin is active-Low and indicates when the device is in the reset state and initializing the device. Spartan UltraScale+ FPGAs will drive the `INIT_B` pin low at power-up or after reset on when properly powered (See *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930) for power rail values and power sequencing). 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 approximately 40 ms. With `POR_OVERRIDE` High, the High time is approximately 9 ms. After the device is initialized the `INIT_B` is released to a high impedance state. If the FPGA detects an error then the `INIT_B` is driven low. The `INIT_B` status signal provides key debug information during configuration.

PUDC_B Pin

The pull-up during configuration (`PUDC_B`) pin determines whether or not I/O pull-ups are enabled during configuration.

Status Register Use and JTAG Access

In addition to the status pins there are several key status registers that can be accessed through the JTAG interface. The Spartan UltraScale+ FPGA JTAG registers that provide valuable data for debug include:

- IDCODE
- JTAG_STATUS
- ERROR_STATUS
- FW_STATUS

AMD Vivado Design Suite device programmer captures the IDCODE and status registers information when connected to the Spartan UltraScale+ FPGA device. The status registers contain information about power rail levels, status pins (`DONE` pin, `INIT_B` pin) and mode pin settings. The status registers also provide SelectMAP bus width and correctable and non-correctable error codes.

Initial Debug Steps

During the configuration process there are basic checks that can be performed to debug and isolate any issues encountered. AMD Spartan UltraScale+ FPGAs PDI format includes a signature pattern (XLNX) that should be checked in the initial boot header. After the signature is seen, the BootROM will run software checks and initiate the configuration. The `CCLK` can be checked to ensure the default configuration rate is seen and then increased rate if an option is selected. Basic checks that can be performed include:

- Try configuration using a different mode. For example, if configuring from a flash device try changing the mode pins to JTAG configuration mode and downloading the PDI with the Vivado Design Suite.
- Try configuration with a different PDI image. If possible, have a known-good PDI image with a simple design that can be used as a test.
- If using a higher speed `CCLK` setting or external master configuration clock, try a slower clock frequency setting.
- Reduce the number of non-default configuration PDI options selected.
- Try applying additional clocks at the end of configuration if the device is in a Slave boot mode and not starting up.
- Verify that the data is being received properly at the destination devices.
- Verify that the latest version of the tools is used.

Appendix A

Additional Resources and Legal Notices

Finding Additional Documentation

Technical Information Portal

The AMD Technical Information Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Technical Information Portal, go to https://docs.amd.com.

Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the AMD Vivado™ IDE, select **Help → Documentation and Tutorials**.
- On Windows, click the **Start** button and select **Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

***Note*:** For more information on DocNav, refer to the *Documentation Navigator User Guide* (UG968).

Design Hubs

AMD 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 DocNav, click the **Design Hubs View** tab.
- Go to the Design Hubs web page.

Support Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Support.

References

These documents provide supplemental material useful with this guide:

1. *UltraScale Architecture and Product Data Sheet: Overview* (DS890)
2. *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930)
3. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
4. *UltraScale Architecture Configuration User Guide* (UG570)
5. *UltraScale Architecture SelectIO Resources User Guide* (UG571)
6. *Bootgen User Guide* (UG1283)
7. *Spartan UltraScale+ FPGAs SelectIO Resources User Guide* (UG861)
8. *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575)
9. *UltraScale Architecture System Monitor User Guide* (UG580)
10. *Vivado Design Suite Tcl Command Reference Guide* (UG835)
11. *Vivado Design Suite Properties Reference Guide* (UG912)
12. *Spartan UltraScale+ Libraries Guide* (UG1704)
13. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)
14. *Vivado Design Suite User Guide: Design Flows Overview* (UG892)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/12/2025 Version 1.1	
General Updates	Updated to include SU45P, SU60P, and removed SU50P.
Configuration Image Size	Updated to address how security features affect PDI size.
PDI File Format	Updated to clarify secure PDI format.
Configuration Sequence	Updated to detail the boot process.

Section	Revision Summary
Boundary-Scan Architecture Registers	Updated table 33: JTAG Registers to address JTAG configuration register length and the instruction description.
Instruction Register	Added Table 35: Instruction Capture Value.
JTAG Configuration Register	Enhanced section to explain the JCONFIG instruction register.
Security Features Summary	Updated to include SU45P, SU60P, and removed SU50P.
Encryption and Authentication Secure Configuration	Added information on hash block.
Programmable Device Image Authentication	Added information on hash block.
05/29/2025 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices

OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright