XAPP1220 (v1.2) March 16, 2022

# UltraScale FPGA BPI Configuration and Flash Programming

Author: Stephanie Tapp and Ryan Rumsey

## Summary

The UltraScale™ architecture master Byte Peripheral Interface (BPI) configuration mode with synchronous read and the External Master Configuration Clock (EMCCLK) enable high-capacity nonvolatile parallel NOR flash storage and shorter configuration times when compared to master serial peripheral interface (SPI) configuration.

The UltraScale FPGA and parallel NOR flash (BPI flash memory) interface connectivity, flash programming steps with Vivado® Design Suite 2014.4, and the BPI configuration mode process are shown. Before using this application note, you should be familiar with the *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1] for FPGA configuration and the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 2] for device programming flows.

**IMPORTANT:** *The Master BPI configuration mode with synchronous read is not recommended for new designs due to flash availability. Xilinx recommends contacting your flash supplier for availability of supported flash devices. See Vivado Design Suite User Guide: Programming and Debugging (UG908) for supported flash devices.*

## Introduction

An UltraScale FPGA requires a configuration bitstream to be delivered after power-up. Parallel NOR flash is a popular option for storing and delivering the bitstream because the wide x16 data bus provides faster configuration times than SPI flash memory alternatives. Systems that use parallel NOR flash for random-access, nonvolatile application data storage can also benefit from consolidating the configuration storage into a single memory device.

Two general flows with parallel NOR flash are demonstrated in this application note, and are shown in Figure 1.

- Vivado Design Suite indirect flash programming with a JTAG cable via the FPGA
- FPGA BPI configuration from the bitstream stored in the parallel NOR flash

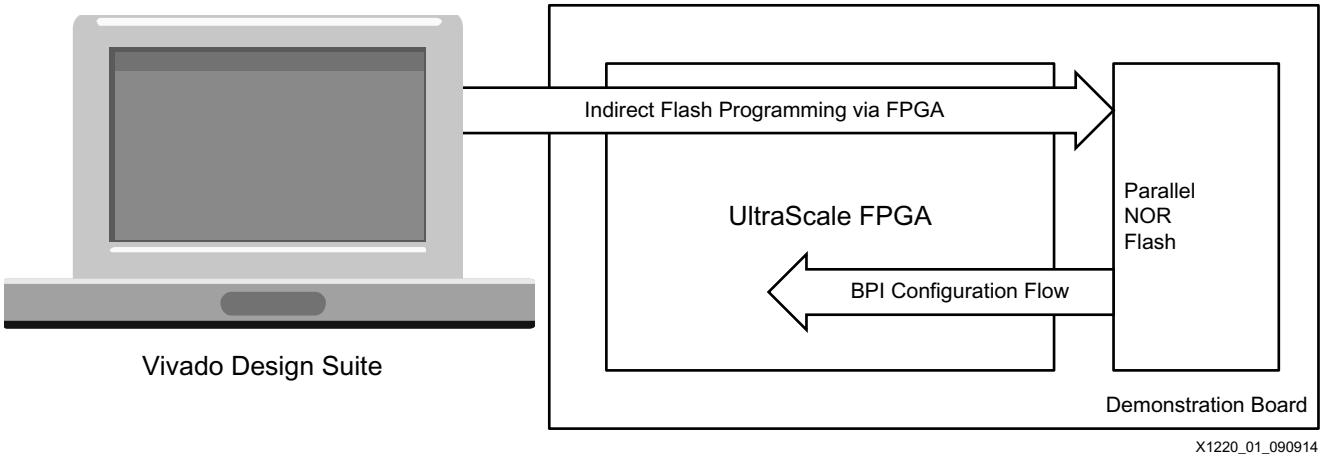


*Figure 1:* **BPI Configuration and Flash Programming Flows**

The following key steps are required to complete the two flows and perform a successful FPGA BPI configuration:

1. Set up appropriate connectivity between the FPGA and parallel NOR flash.
2. Create a bitstream (`.bit`) file using the Vivado Design Suite.
3. Create a flash programming file (`.mcs`) using the Vivado Design Suite.
4. Program the parallel NOR flash in-system using the Vivado Design Suite.
5. Configure the target FPGA from the parallel NOR flash (Power cycle or PROGRAM_B pulse).

The document sections are:

- UltraScale FPGA BPI Configuration and Flash Programming describes the basic two flows described in this application note.
- Hardware and Connectivity details the proper hardware setup and FPGA BPI configuration interface connections.
- File Generation for BPI Configuration describes the steps and options needed to generate FPGA bitstream and parallel NOR flash programming files.
- Indirect Parallel NOR Flash Programming, page 18 provides the instructions to program the parallel NOR flash.
- BPI Configuration Sequence describes the BPI configuration process with synchronous read and configuration time estimation.
- Configuration Times gives the details necessary to estimate the configuration time for a given FPGA and configuration frequency.
- Debug Guidance provides a summary of key tips to avoid common oversights when implementing the BPI configuration mode.

# UltraScale FPGA BPI Configuration and Flash Programming

Configuration is the process of downloading configuration data into an FPGA using an external source such as a flash device or microprocessor. In the BPI configuration mode, the FPGA supports a direct connection to the address, x16 or x8 data bus, and control signals of a parallel NOR flash for extracting a stored bitstream. The UltraScale FPGA BPI configuration interface supports two flash read options, asynchronous or synchronous, and two configuration clock options, internal configuration clock (CCLK) or External Master Configuration Clock (EMCCLK). The synchronous read option and EMCCLK are demonstrated in this application note because configuration data can be delivered significantly faster than with the asynchronous read mode and CCLK. Refer to Configuration Times, page 25 for an estimate on the configuration time.

The Vivado Design Suite provides the ability to indirectly program parallel NOR flash in-system with existing configuration connections between the parallel NOR flash and the FPGA. This programming feature configures the FPGA through JTAG with an indirect programming bitstream. This bitstream enables a path through the FPGA between the JTAG programming cable and the parallel NOR flash interface. The Vivado device in-system programming feature can enable the testing and debugging of multiple design iterations in the prototype phase, but the feature is not intended for high volume production programming. For production programming, consider programmer solutions from vendors such as BPM Microsystems or Data I/O.

## Selecting a Parallel NOR Flash

Several factors are considered when selecting the parallel NOR flash device for a configuration source, such as the storage capacity required by the application, the package type to meet board space requirements, the data bus width for configuration time, and the flash I/O voltage range. See *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1] for additional details. For a list of the supported flash devices, see *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 2].

This application note uses the Micron 28F00AG18F (MT28GU01GAAA1E) flash because it provides the fastest configuration time and has the required density.

# Hardware and Connectivity

This application note uses the following hardware to demonstrate the UltraScale FPGA BPI configuration (with synchronous read and EMCCLK) and flash programming:

- Virtex® UltraScale XCVU095
- Micron Parallel NOR Flash 28F00AG18F (MT28GU01GAAA1E)
- Digilent USB cable or Xilinx Platform Cable USB (see *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for a list of supported cables [Ref 2])

***Note:*** Micron Parallel NOR Flash 28F00AG18F has recently had a name change to MT28GU01GAAA1E. There was no functional change to the flash device, but the address signal names have changed. For this application note, the 28F00AG18F signal names are used.

## Parallel NOR Flash Connectivity Basics

The associated signals for the BPI configuration with EMCCLK interface between the UltraScale FPGA and the Micron parallel flash device are shown in Figure 2. Signal descriptions are listed in Table 1. The Micron Parallel NOR Flash referenced in this application note uses a 16-bit data bus, a 26-bit address bus and control signals.

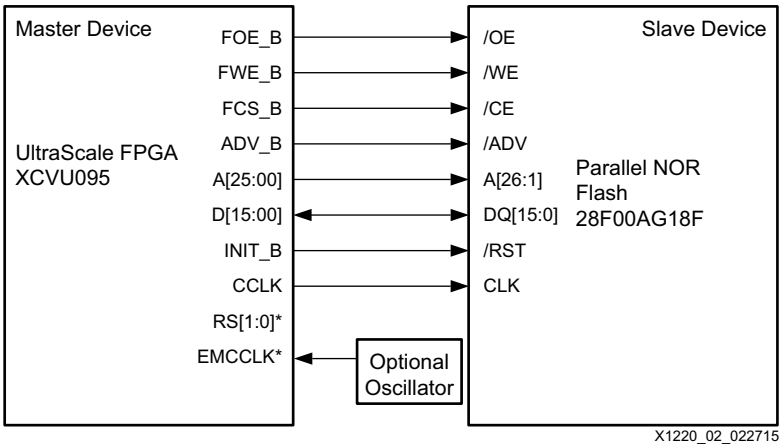


*Figure 2:* **Basic Parallel NOR Flash to FPGA Connections**

***Note:*** The RS[1:0] and EMCCLK signals are optional advanced features.

## BPI Configuration Interface

Figure 3 shows the UltraScale FPGA pins related to BPI configuration.

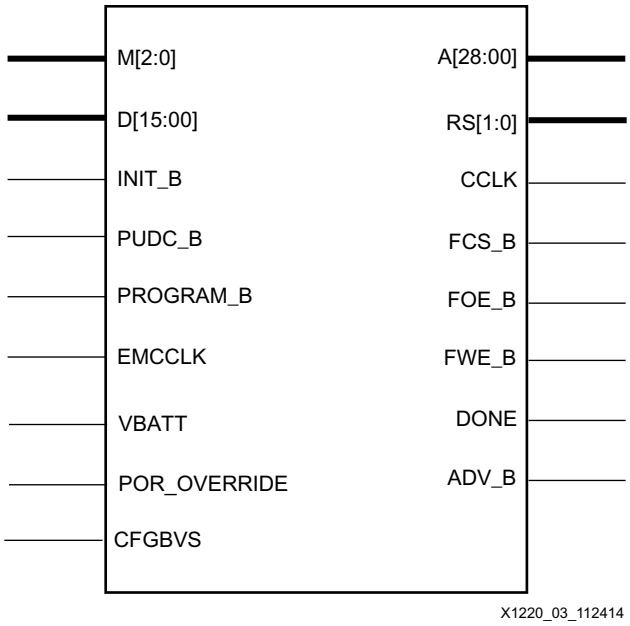


*Figure 3:* **Master BPI Configuration Mode Interface**

Table 1 describes the BPI configuration interface signal functions. See the *UltraScale Architecture Configuration User Guide*, Table 1-6: Configuration Pin Definitions (UG570) [Ref 1] and the Micron 28F00AG18F flash data sheet [Ref 3] for more detailed information on pin definitions.

*Table 1:* **UltraScale FPGA BPI Configuration Signal Descriptions**

| UltraScale FPGA Pin Name | Direction | Description |
|---|---|---|
| A[28:00] | Output | Address Bus – Output addresses to a parallel NOR (BPI) flash memory. A00 is the least-significant address bit. Connect the FPGA A[28:00] pins to the parallel NOR flash address pins with the FPGA A00 pin connected to the least-significant flash address input pin that is valid for the used data bus width. Depending on the parallel NOR flash and used data bus width, the least-significant address bit of the flash can be A1, A0, or A-1. Note that multi-purpose upper address pins that exceed the address bus width of the parallel NOR flash are still driven during configuration and Vivado Design Suite indirect flash programming. This must be considered if they are used as I/O after configuration. |
| RS[1:0] | Output | Revision Select – Used for multi-bitstream applications to select between revisions and provide fallback capability. RS[1:0] are actively driven Low to load the fallback bitstream when a configuration error is detected. See the *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1] for additional details on revision management. |

*Table 1:* **UltraScale FPGA BPI Configuration Signal Descriptions *(Cont'd)***

| UltraScale FPGA Pin Name | Direction | Description |
|---|---|---|
| CCLK | Output | Configuration Clock – This pin is the initial configuration clock source for all configuration modes except JTAG. CCLK is an output in master BPI configuration mode. During the BPI asynchronous read mode, CCLK does not directly clock the parallel NOR flash, but is used internally by the FPGA to generate the address and sample read data. During the BPI synchronous read mode, CCLK must be directly connected to the parallel NOR flash to clock the data out sequentially. |
| FCS_B | Output | Flash Chip Select (bar) – This active-Low flash chip select output. This output is actively toggled during configuration. |
| FOE_B | Output | Flash Output Enable (bar) – This active-Low flash output enable. This output is actively toggled during configuration. |
| FWE_B | Output | Flash Write Enable (bar) – Output is actively toggled during configuration. |
| DONE | Bidirectional | Done – A High signal on the DONE pin indicates completion of the configuration sequence. By default, the DONE output is open-drain.<br>**Note**: DONE has a default internal pull-up resistor of approximately 10 kΩ. External 4.7 kΩ resistor circuits are not required but are recommended. |
| ADV_B | Output | Address Valid (bar) – This active-Low address valid output. Required for the synchronous read option in Master BPI configuration mode. This signal is needed to tell when an address is valid in synchronous read mode. The signal also needs to be driven Low for asynchronous read mode. |
| M[2:0] | Input | Configuration Mode – The mode pins determine the configuration mode. M[2:0]=010 for Master BPI configuration mode. |
| D[15:00] | Bidirectional | Data Bus – This x16 data bus is sampled by the rising edge of the FPGA CCLK. Data is read from the flash on this bus and commands to write to the flash read configuration register are sent on this bus when a synchronous read command is seen by the configuration controller. The FPGA monitors the D[07:00] for an auto-bus-width-detect pattern that determines whether only D[07:00] (x8 bus width) is used or a wider (x16) data bus width is used. Connect used data bus pins to the corresponding data pins on the parallel NOR flash. |
| INIT_B | Bidirectional | Initialization (bar) – Active-Low FPGA initialization pin or configuration error signal. The FPGA drives this pin Low when the FPGA is in a configuration reset state, when the FPGA is initializing (clearing) its configuration memory, or when the FPGA has detected a configuration error. Upon completing the FPGA initialization process, INIT_B is released to high-impedance at which time an external resistor is expected to pull INIT_B High. 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. |

*Table 1:* **UltraScale FPGA BPI Configuration Signal Descriptions *(Cont'd)***

| UltraScale FPGA Pin Name | Direction | Description |
|---|---|---|
| PUDC_B | Input | Pull-Up During Configuration (bar) – Active-Low input enables internal pull-up resistors on the SelectIO™ pins after power-up and during configuration. When PUDC_B is Low, internal pull-up resistors are enabled on each SelectIO pin. When PUDC_B is High, internal pull-up resistors are disabled on each SelectIO pin. PUDC_B must be tied either directly or through a $\leq 1$ kΩ resistor to $V_{CCO\_0}$ or GND. |
| PROGRAM_B | Input | Program (bar) – This active-Low asynchronous full-chip reset. |
| EMCCLK | Input | External Master Configuration Clock – An external clock is supplied on this input, and the FPGA configuration controller switches over to use this clock instead of the CCLK (internal configuration clock) after the EMCCLK command is read from the bitstream header. The EMCCLK enables more predictable configuration times because the clock tolerance is determined by the external oscillator selected. |
| VBATT | N/A | Battery Backup Supply – This supply is for the FPGA internal volatile memory that stores the key for the AES decryptor. For encrypted bitstreams that require the decryptor key from the volatile key memory area, connect this pin to a battery to preserve the key when the FPGA is unpowered. If there is no requirement to use the decryptor key from the volatile key storage area, connect this pin to GND or $V_{CCAUX}$. |
| POR_OVERRIDE | Input | Power On Reset Override – Reduces time for power on reset ($T_{POR}$) from power on to INIT_B rise as specified in the UltraScale family data sheets *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS892) [Ref 4] and *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 5]. 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 $T_{POR}$ delay.<br>For applications where power on time is critical, the UltraScale FPGA POR_OVERRIDE feature should be considered. If the application's maximum power supply ramp rate can meet the UltraScale data sheet $T_{POR}$ specification, this signal can be tied to $V_{CCINT}$ and shorten the built-in delay.<br>**CAUTION**: Do not allow this pin to float before and during configuration. Must be tied to $V_{CCINT}$ or GND. Do not connect to $V_{CCO\_0}$. |
| CFGBVS | Input | Configuration Banks Voltage Select – Determines the I/O voltage operating range and voltage tolerance for the dedicated configuration bank 0, and for the configuration pins in bank 65 when those banks are HR I/O banks. Connect CFGBVS High or Low per the bank voltage requirements. If the $V_{CCO\_0}$ supply for bank 0 is supplied with 2.5V or 3.3V, this pin must be tied High (connected to $V_{CCO\_0}$). Tie CFGBVS to Low (connect to GND) only if the $V_{CCO\_0}$ for bank 0 is less than or equal to 1.8V. When bank 65 is used for configuration, it should have the same voltage as bank 0.<br>**CAUTION!** To avoid device damage, this pin must be connected correctly to either $V_{CCO\_0}$ or GND. |

Figure 4 shows the master BPI configuration interface connections needed for the x16 Synchronous Read mode. Signal connections for the JTAG port connections to support the Vivado tool's flash programming feature are included.

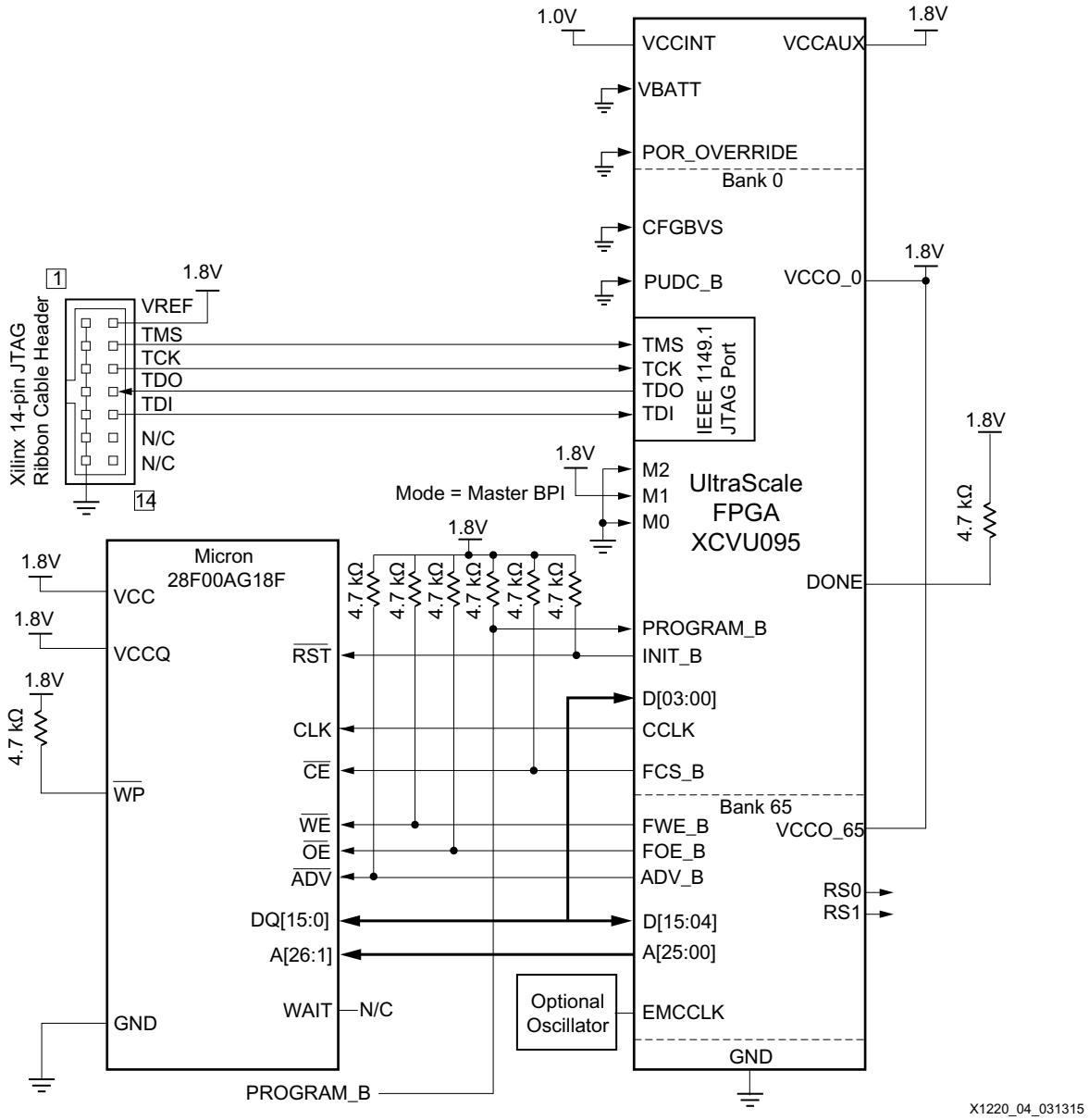


*Figure 4:* **Master BPI Configuration Mode x16 Example**

Notes relevant to Figure 4:

1. CFGBVS is tied to ground for this example because the Bank 0 and Bank 65 are set to 1.8V. If the design uses 2.5V or 3.3V, the CFGBVS pin must be set High. Refer to *UltraScale Architecture Configuration User Guide* (UG570) for details [Ref 1].
2. For applications where power on time is critical, the UltraScale FPGA POR_OVERRIDE feature should be considered. If the application's maximum power supply ramp rate can meet the UltraScale family data sheet $T_{POR}$ specification ([Ref 4] and [Ref 5]), this signal can be tied to $V_{CCINT}$ and shorten the built-in delay.
3. The UltraScale FPGA JTAG signals TCK, TMS, TDI, and TCK are necessary for the Vivado tool's flash programming feature. The JTAG interface is also a popular debug interface used on many application setups. See Indirect Parallel NOR Flash Programming, page 18 for steps on how to program the flash indirectly.

4. The EMCCLK maximum frequency is dependent on the target flash and FPGA. See Configuration Times, page 25 to determine the maximum frequency for a given setup.
5. RS[1:0] are optional revision management pins. In applications where revision control is required, the FPGA RS[1:0] pins are tied to the upper two flash address pins.
6. The UltraScale FPGA $V_{CCO\_0}$ supply must be compatible with the $V_{CCQ}$ on the parallel flash.
7. The Micron 28F00AG18F has a write protect (/WP) signal that should be tied appropriately to allow for indirect flash programming and configuration.
8. The Micron 28F00AG18F WAIT output signal is not required for BPI configuration with synchronous read. However, if the parallel NOR is accessed for user data after configuration in synchronous read, the WAIT signal can be connected to an FPGA I/O for application use. See Micron data sheet(s) [Ref 3] for WAIT operation.

# File Generation for BPI Configuration

The Vivado Design Suite creates UltraScale FPGA bitstreams, flash programming files, and indirectly programs parallel NOR flash devices. Special options must be set by the user to properly generate files for the BPI configuration mode.

Figure 5 outlines the Vivado tool flow required for indirect flash programming. The primary steps highlighted in this section are:

- Design constraints required for BPI configuration
- The write_bitstream command generates a bitstream after implementing your design.
- The write_cfgmem command creates a flash programming file containing your bitstream.
- Vivado Hardware Manager indirectly programs the parallel NOR flash with the flash programming file.

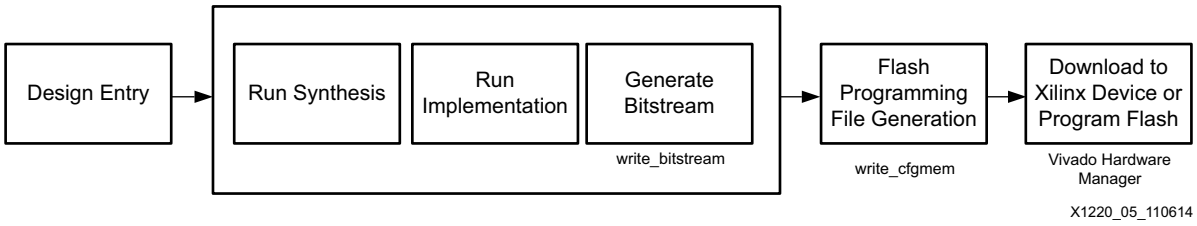


*Figure 5:* **Vivado Design Suite Tool Flow**

## Design Constraints for BPI Configuration

It is recommended to include key BPI configuration constraints in the XDC file prior to synthesizing and implementing a design in the Vivado Design Suite. The important constraints for the BPI Configuration mode are listed below, and descriptions of the options used in this application note are provided:

```
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN DIV-1 [current_design]
set_property BITSTREAM.CONFIG.BPI_SYNC_MODE TYPE1 [current_design]
set_property CONFIG_MODE BPI16 [current_design]
```

Applications that use the BPI configuration EMCCLK option must ensure that the voltage is defined for the EMCCLK multi-function pin. To enable the EMCCLK, the CONFIG_VOLTAGE property is required to define the multi-function pin voltage.

The Configuration Bank Voltage Selection (CFGBVS) pin must be connected to GND to support bank 0 operations at 1.8V demonstrated in this example. For more details on CFGBVS, see *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1]. To connect the CFGBVS pin to GND, use the CFGBVS property.

The bitstream size can be reduced by using the compression option. The rate of compression cannot be guaranteed, as it is dependent on the design. However, it reduces the size, and size reduction improves the parallel NOR flash programming time. Use the BITSTREAM.GENERAL.COMPRESS property set to TRUE.

When the EMCCLK is used, the configuration property must be set to enable it. Use the property BITSTREAM.CONFIG.EXTMASTERCCLK_EN. This property is set to DIV–1 in this example, because dividing the EMCCLK is not desired. The Vivado tcl command list_property_values BITSTREAM.CONFIG.EXTMASTERCCLK_EN [current_design] returns the available settings.

Finally, if the synchronous read is desired for faster configuration times, the property BITSTREAM.CONFIG.BPI_SYNC_MODE should be set. In this example it is set to TYPE1 for the 28F00AG18F flash.

## Preparing a Bitstream for BPI Configuration

This section provides examples on how to generate a bitstream for BPI configuration from an implemented design using Vivado tools. The two methods for generating bitstreams shown in the following examples are:

- Using the Vivado Integrated Design Environment (IDE) flow (Project Mode)
- Using the Vivado IDE Tcl console (Project mode) or Tcl shell (Non-Project mode)

Each method can be completed independently of the other. For detailed descriptions of these recommended bitstream options, see *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 2].

### *Vivado IDE Example to Generate Bitstream*

The bitstream can be generated from the Vivado IDE Generate Bitstream flow. Some default bitstream settings must be changed to support BPI configuration. Changing the bitstream properties in the XDC file before synthesizing a design as discussed in the Design Constraints for BPI Configuration section is recommended. If the edits are done prior to synthesis, click the **Generate Bitstream** selection under the Flow Navigator in Vivado Design Suite.

Alternatively, if a design already exists that needs to be modified to target BPI configuration, the properties can be set through the Edit Device Properties dialog box. To do this, open your synthesized or implemented design in the Vivado IDE and follow these steps:

1. To modify bitstream settings in the Flow Navigator, locate the **Program and Debug** option and click the **Bitstream Settings** button as shown in Figure 6.

   Alternatively, click **Tools > Edit Device Properties**.

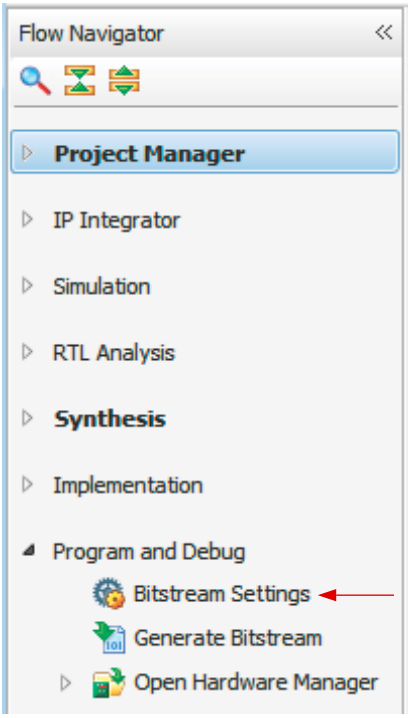


*Figure 6:* **Flow Navigator Window**

2. From the Project Settings window, edit device configuration bitstream settings using the **Configure additional bitstream settings** link, shown in Figure 7. This link only becomes available when the project has an opened synthesized or implemented design.

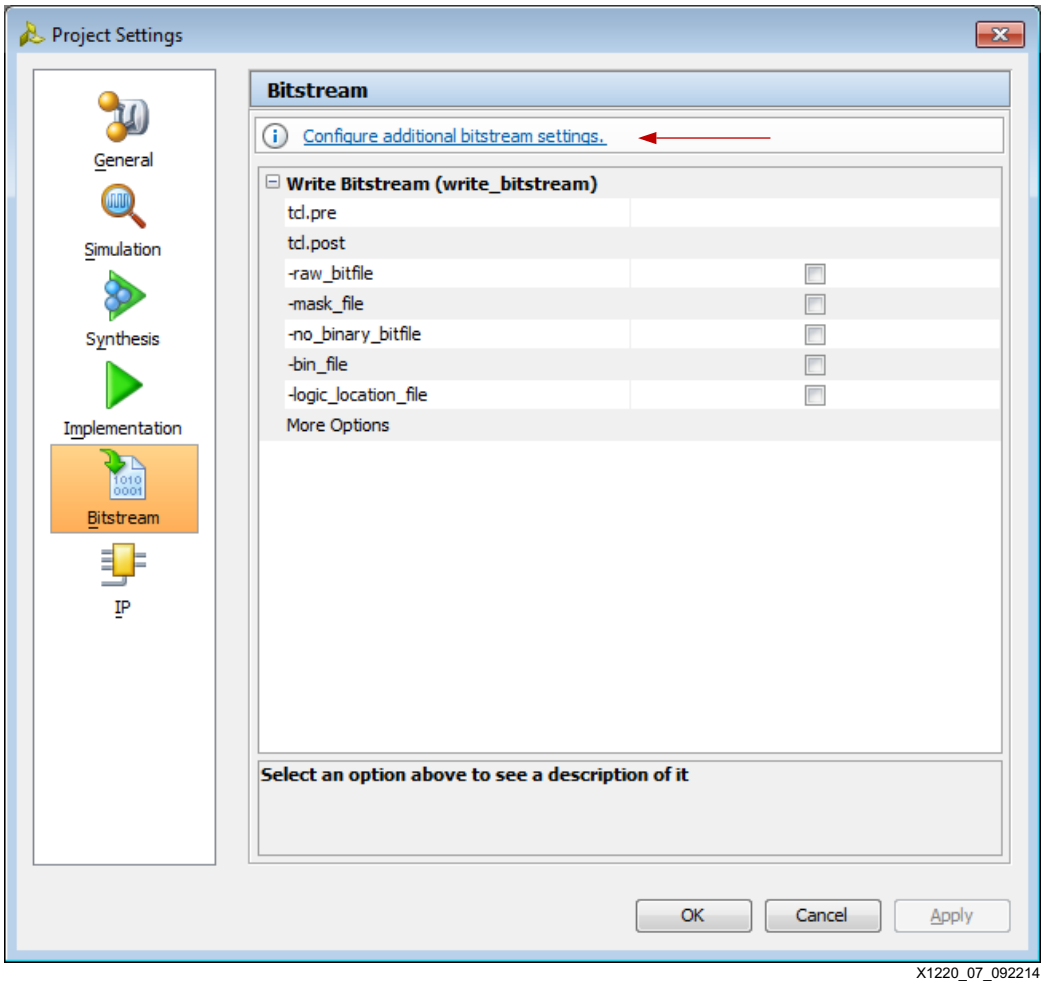


*Figure 7:* **Project Settings Bitstream Window**

3. In the General category shown in the left of Figure 8, set the **Enable Bitstream Compression** as **TRUE** to reduce programming and configuration time in general. This setting is optional and does not guarantee reduced configuration times.

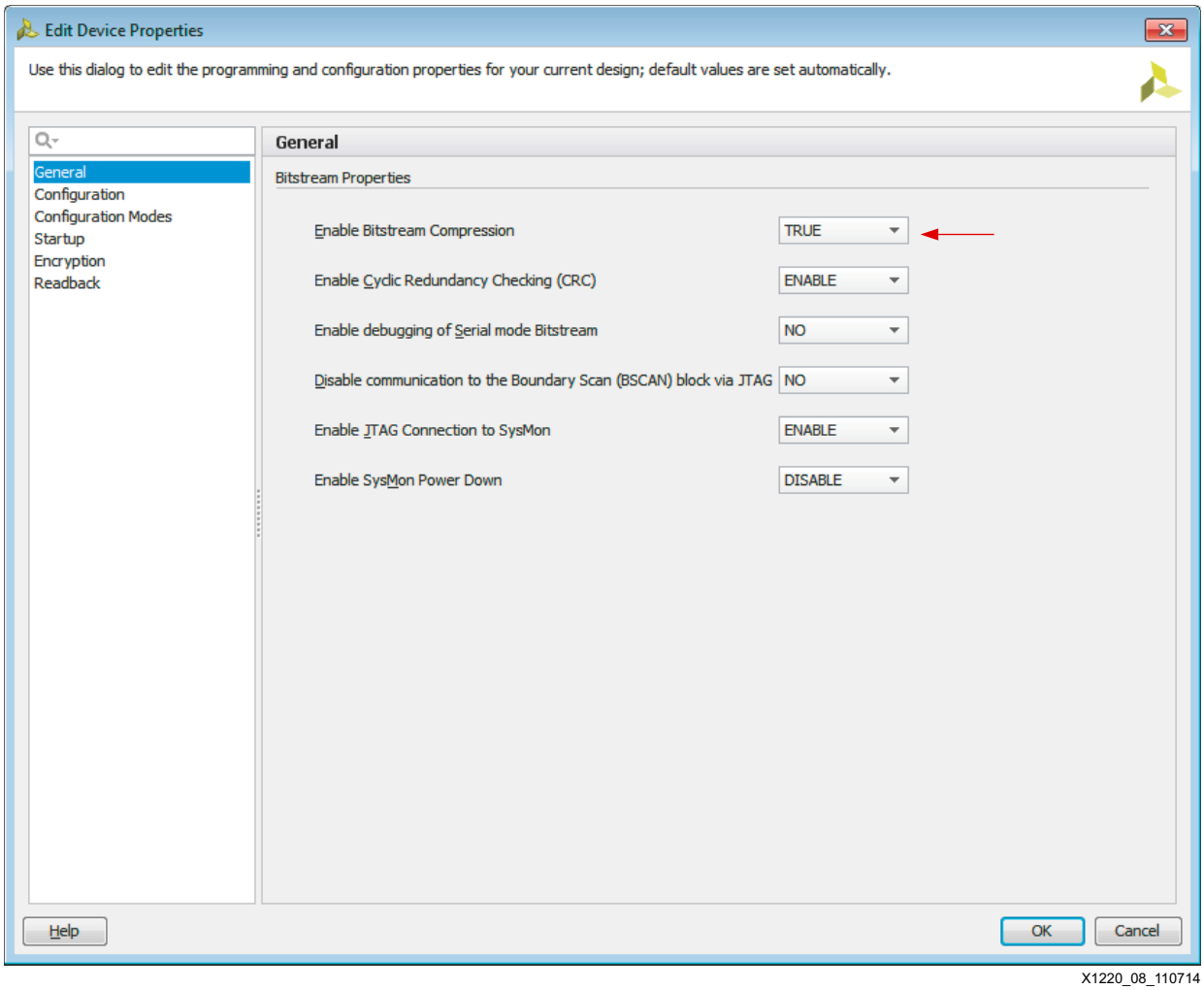


*Figure 8:* **General Options for Configuration Bitstream Settings**

4. Under the Configuration category, ensure that the options shown in Figure 9 are selected to generate a bitstream for BPI configuration with synchronous read capability using the external configuration clock (EMCCLK).

   Under **Configuration Setup**:

   a. Set **Enable external configuration clock and set divide value** to **DIV-1**.

   b. Set **Configuration Voltage** to **1.8**

   c. Set **Configuration Bank Voltage Selection** to **GND**.

      These settings ensure that 1.8V is used for the configuration bank as needed for this example.

Under **BPI Configuration:**

d. Set **Synchronous Mode** to **Type1** for the parallel NOR flash 28F00AG18F family.

   See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 2], for property details.

e. Click **OK** to accept modifying bitstream settings.

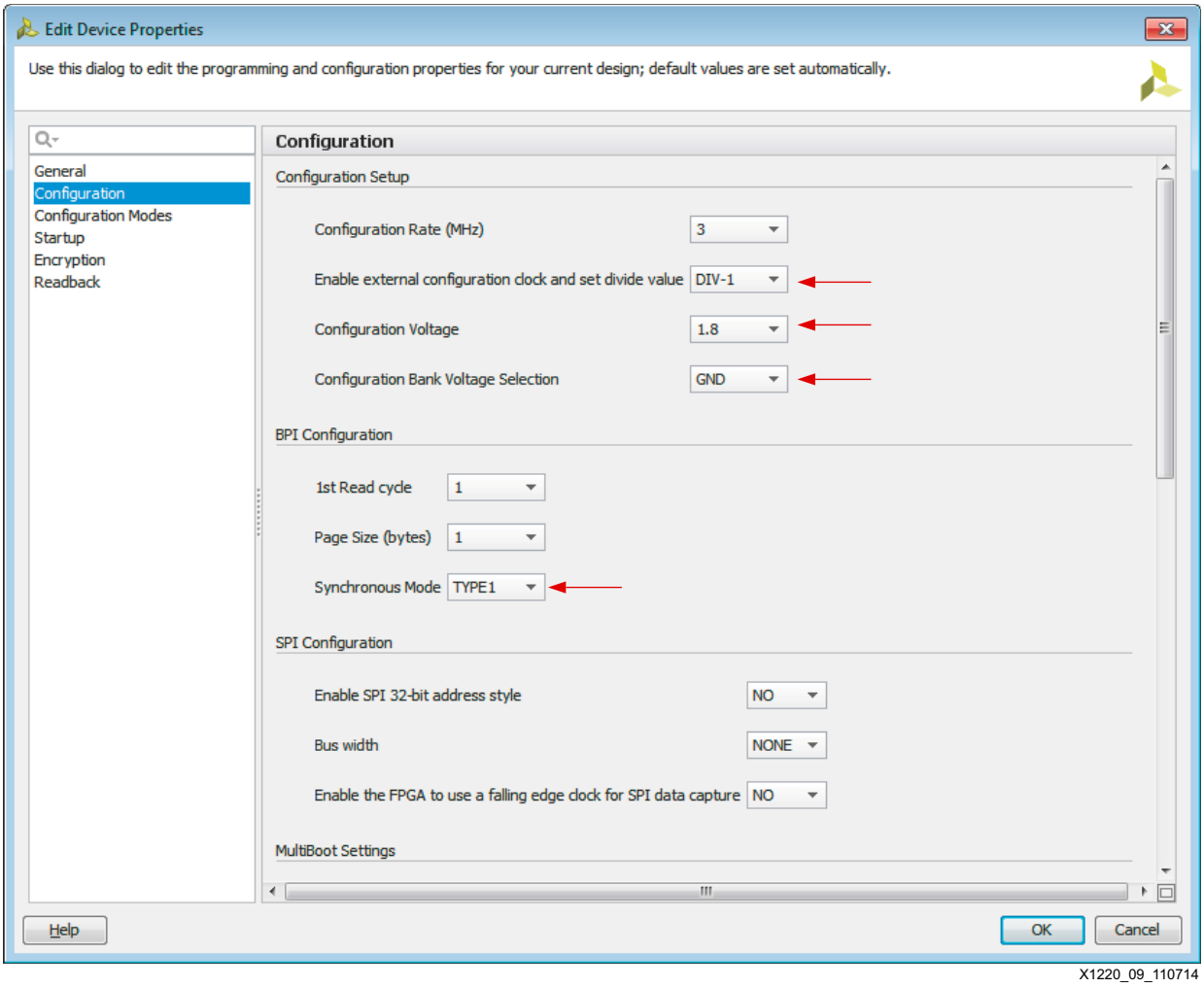


*Figure 9:* **Configuration Options for Configuration Bitstream Settings**

5. Under the Configuration Modes, ensure that the Master BPI-Up x16 option is selected, as shown in Figure 10.

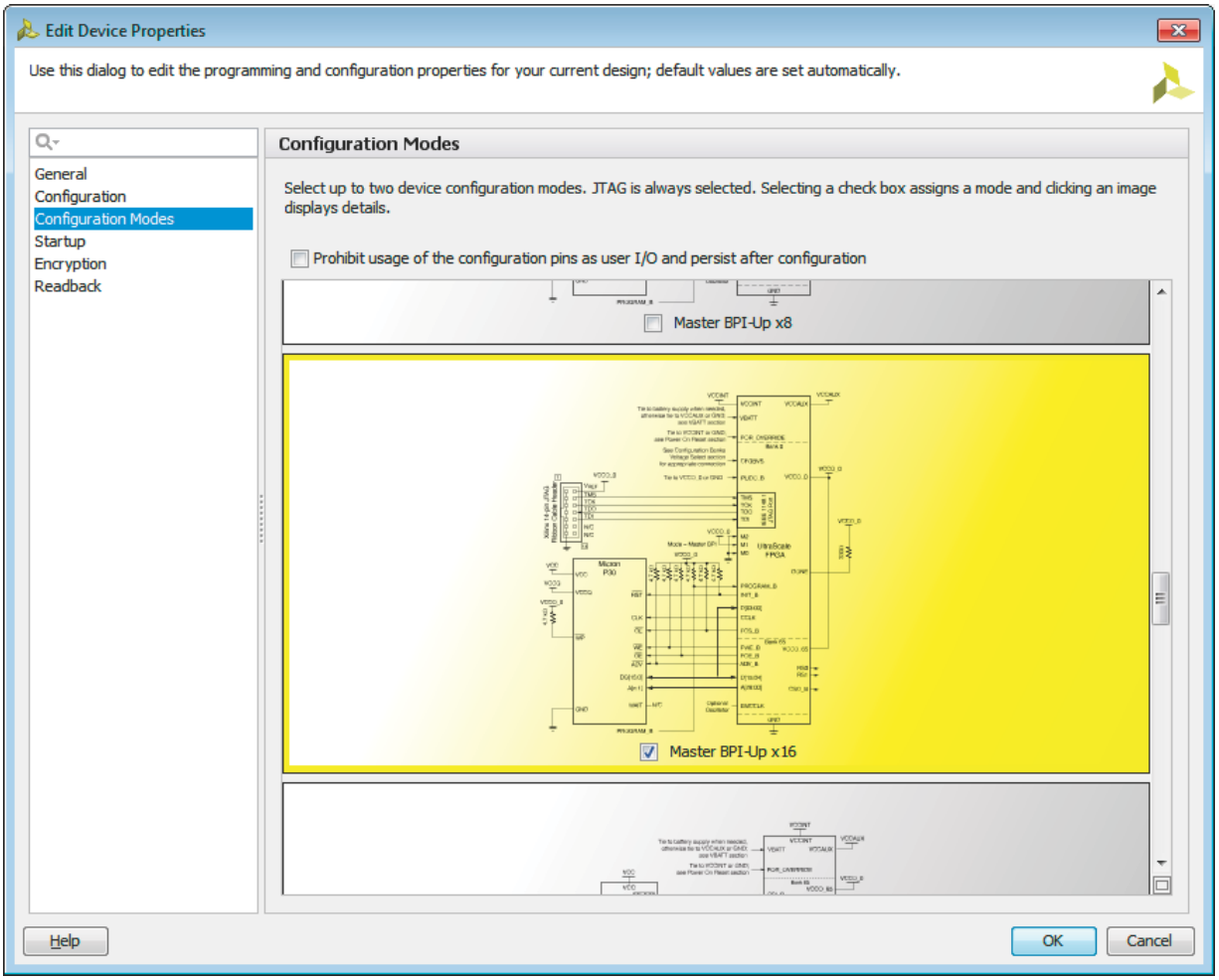


*Figure 10:* **Master BPI-Up x16 Option**

When you finish editing the properties, the constraints are not saved until you select **File > Save Constraints**. This writes the properties to the target constraints file. Xilinx recommends that you make changes to the device configuration properties in the synthesized design before running implementation.

Furthermore, after the constraint file is saved, the synthesized design might become out-of-date. Instead of re-synthesizing, open the **Design Runs** tab at the bottom of the Vivado tool window, right-click the current synthesis run (for example, synth_1), and select **Force Up-To-Date**.

6. After the constraints are saved, generate the bitstream for your design. In the Flow Navigator, under **Program and Debug**, press **Generate Bitstream**, or press **Flow > Generate Bitstream**. The location of the created bitstream is by default located in:

```
<Project_Dir>\Project_Name.runs\impl_1\
```

***Note:*** The location of the Vivado project subfolder directory is referred to as `<Project_Dir>` in this application note.

### *Vivado IDE Tcl Console Example to Generate Bitstream*

Bitstream properties and generation can be controlled through the Vivado IDE Tcl Console tab, shown in Figure 11, or through the Tcl command shell. Before generating the bitstream file, bitstream properties must be set as constraints for the design. Adding the properties as specified in the XDC file as described in the BPI Configuration Sequence section is recommended. If a design already exists and modifications are needed, here is an overview of the command line flow:

1. If the synthesized design is out of date, re-launch synthesis.
2. Open the synthesized design to modify constraint properties.
3. Set bitstream properties and save the constraints file.
4. Launch implementation, then open the implemented design when completed.
5. Generate the bitstream.

To generate a bitstream, the project must have an open implemented design. For the command to write a bitstream file for the current project, see Figure 11 (`write_bitstream -verbose <file name>`). The bitstream written is based on the open implemented design. The `-verbose` switch summarizes all of the `write_bitstream` options used. For more help using this command, type `write_bitstream -help` in the Tcl console.

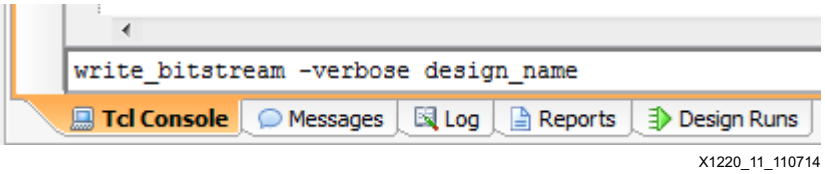


*Figure 11:* **Tcl Console Tab Example Command**

The Tcl commands that result from the steps in Vivado IDE Example to Generate Bitstream, page 11 are listed here. It is assumed that the target design is ready for synthesis:

```
launch_runs synth_1
open_run synth_1 -name netlist_1
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN DIV-1 [current_design]
set_property BITSTREAM.CONFIG.BPI_SYNC_MODE TYPE1 [current_design]
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
save_constraints
reset_run impl_1
launch_runs impl_1
close_design
open_run impl_1
write_bitstream Design_BIT
```

## Preparing a Parallel NOR Flash Programming File

The Vivado Design Suite can create a flash programming file (`.mcs`) from an FPGA bitstream (`.bit`) that can be used to program the parallel NOR flash. The basic command to write a flash programming file is:

```
write_cfgmem -format <arg> -size <arg> -interface <arg> -loadbit <arg> <file>
```

For a detailed description of the `write_cfgmem` command, use the `-help` command:

```
write_cfgmem -help
```

## Vivado IDE Tcl Console Example to Generate Flash Programming File

An example command to create a flash programming file for the 128 MB (1 Gb) 28F00AG18F is as follows:

```
write_cfgmem -format mcs -size 128 -interface BPIx16
-loadbit "up 0x0 <Project_Dir>/Project_Name.runs/impl_1/Design_Name.bit"
<Project_Dir>/Design_Name.mcs
```

This example sets the file format to the standard `.mcs` and uses a target flash size of 128 MB (1 Gb). The `-size` option is specified in megabytes. The `-interface` option determines any special data handling for the target mode, in this case the data formatting is x16. The `-loadbit` switch specifies how many bitstreams are targeted and their starting address location.

***Note:*** Be aware of white spaces in command line file paths. Either avoid using spaces or de-reference spaces in the command line with two backslashes: "\\".

The Tcl command line must use forward slashes (/) in the file path.

By default, Vivado Design Suite writes files into the folder it is launched from. To avoid this, you can either:

- Include the file path of the output file as shown in the previous example.
- Change the directory with the command: `cd <file path>`

Here is an alternate example for the target using `cd` and `write_cfgmem`:

```
cd <Project_Dir>
write_cfgmem -format mcs -size 128 -interface BPIx16
-loadbit "up 0x0 Project_Name.runs/impl_1/Design_Name.bit" Design_Name.mcs
```

# Indirect Parallel NOR Flash Programming

The basic setup of the Vivado indirect flash programming feature is shown in Figure 12 for an UltraScale FPGA evaluation board. The Vivado Hardware Manager user interface or Tcl commands can be used to program an on-board flash.

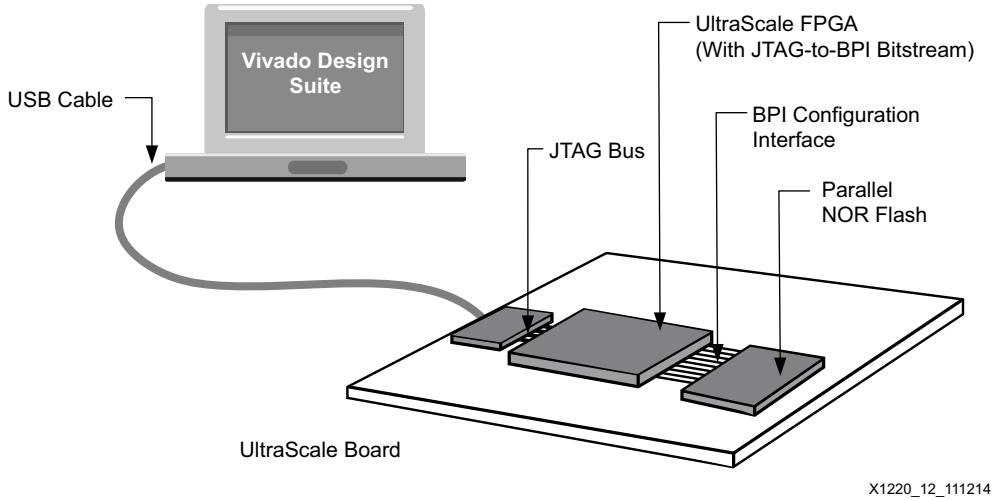


*Figure 12:* **Vivado Indirect Flash Programming Setup**

### *UltraScale FPGA Board Setup for Vivado Indirect Flash Programming*

The basic board setup to program the parallel NOR flash with the Vivado Design Suite and to properly configure the UltraScale FPGA is briefly highlighted as follows:

1. Ensure the settings for the mode pins M[2:0] and UltraScale BPI configuration interface pins are connected correctly. See Figure 4.
2. Power on the board.
3. Connect a supported JTAG Digilent USB cable or Xilinx Platform Cable USB II.

### *Vivado IDE Example to Program Parallel NOR Flash*

1. Open the Vivado Hardware Manager. In the Flow Navigator panel, under the **Program and Debug** tab, click **Open Hardware Manager** as highlighted in Figure 13. Alternatively, select **Flow > Open Hardware Manager**.

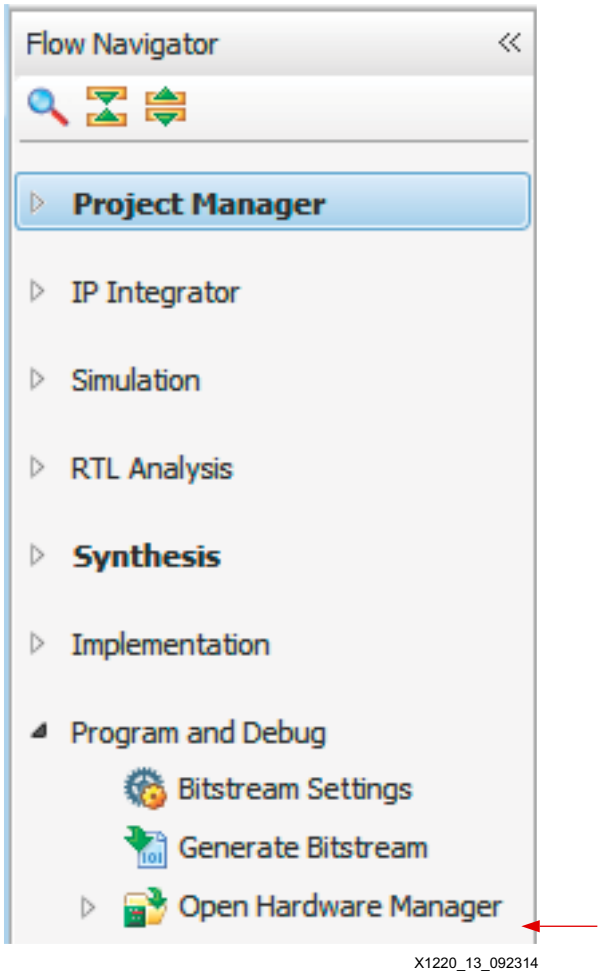


*Figure 13:* **Flow Navigator Window**

2. Open a hardware target by either selecting **Open Target > Open New Target** under the **Hardware Manager** tab in the Flow Navigator, or select **Tools > Open New Target**.

3. The Open Hardware Target Wizard window appears. Follow its guidance to select the XCVU095 FPGA. Select the frequency for the JTAG connection shown in Figure 14. For this example, the default frequency of 15 MHz is used. Click **Next** until you finish the wizard.

   If a hardware device is not seen, check the cable connections and power. Refer to Debug Guidance, page 26, for additional tips.

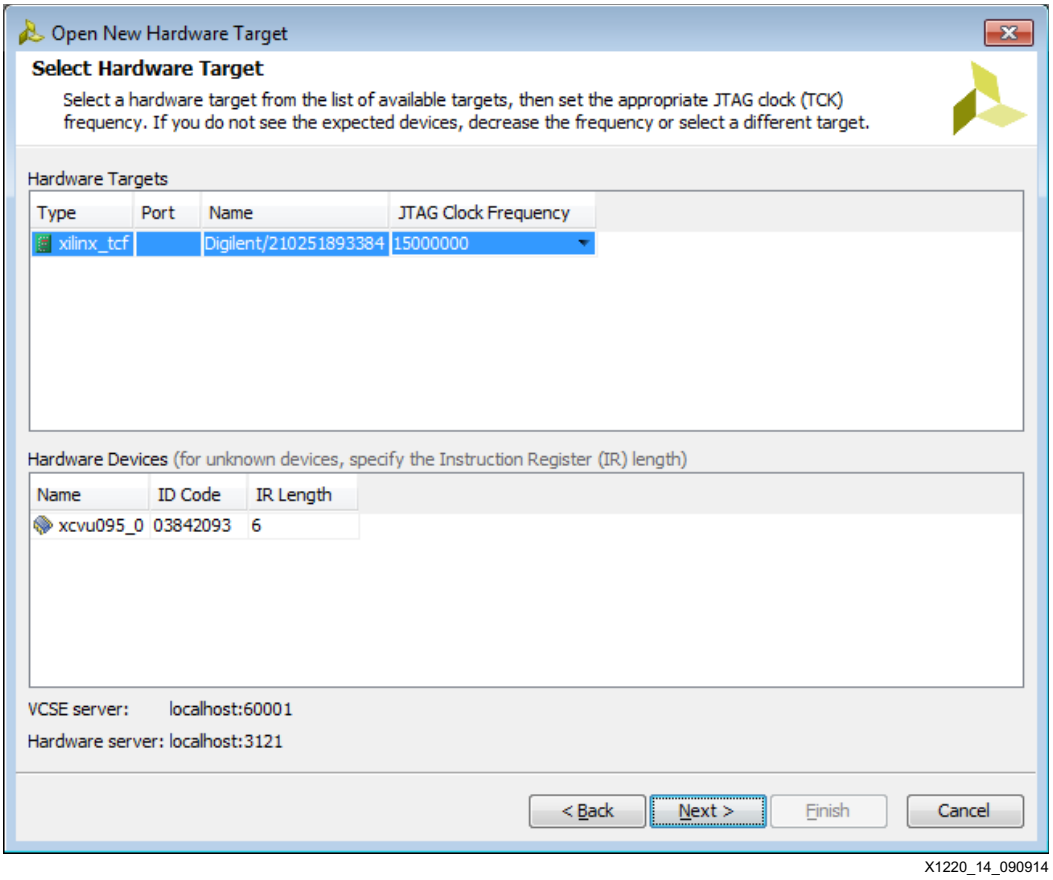


*Figure 14:* **Open Hardware Target Wizard JTAG Clock Frequency Selection Window**

4. Press **Add Configuration Memory > XCVU095_0** in the **Hardware Manager** tab under **Program and Debug** tab in the Flow Navigator. A window appears to select the configuration memory part as shown in Figure 15. Select the Micron (28F00AG18F) MT28GU01GAAX1E and click **OK**.

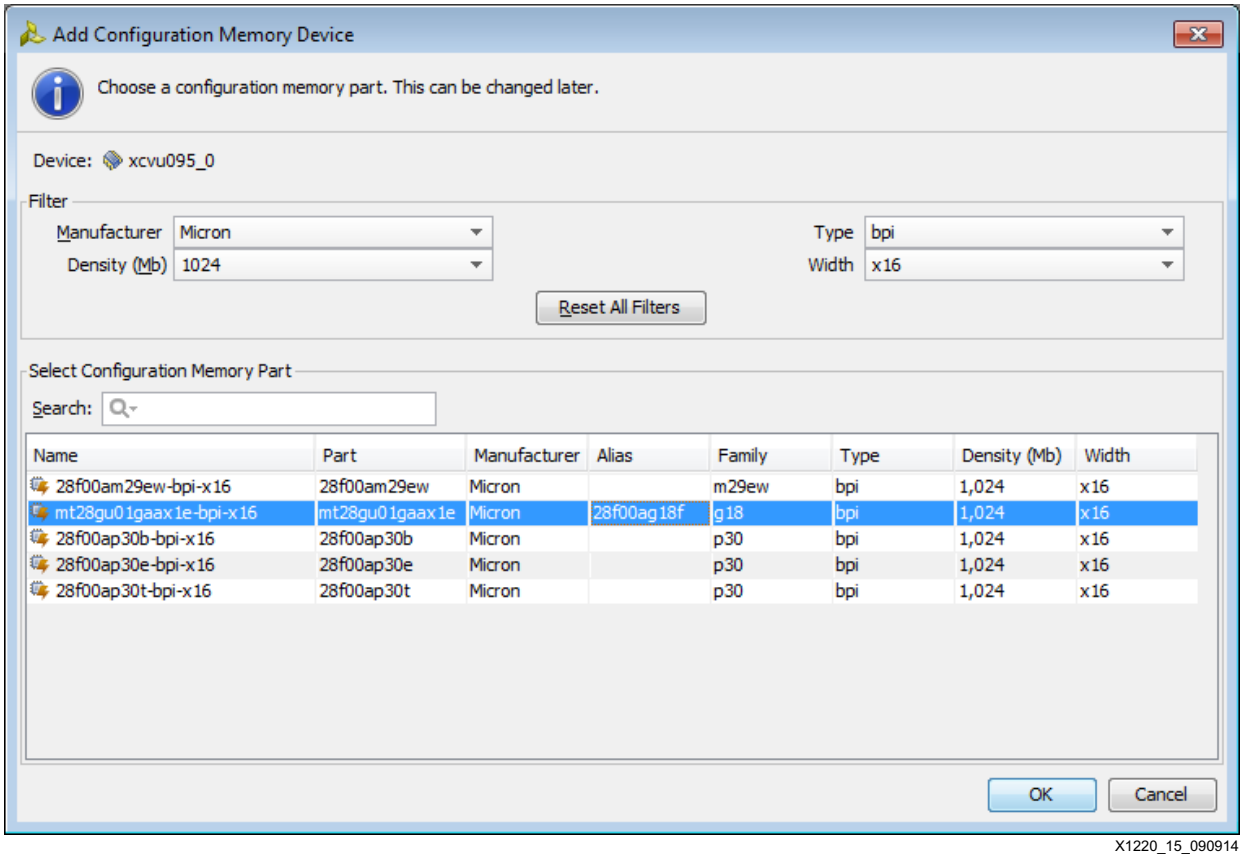


*Figure 15:* **Flash Part Selection**

5. By default, a dialog box asks if you want to program the configuration memory device now. Click **OK** and enter the configuration file for the memory device in the window that follows, as shown in Figure 16. Click **OK** to begin flash programming.

***Note:*** If the RS[1:0] are used for revisions, Vivado IDE must be told which FPGA address pins the RS[1:0] have replaced.

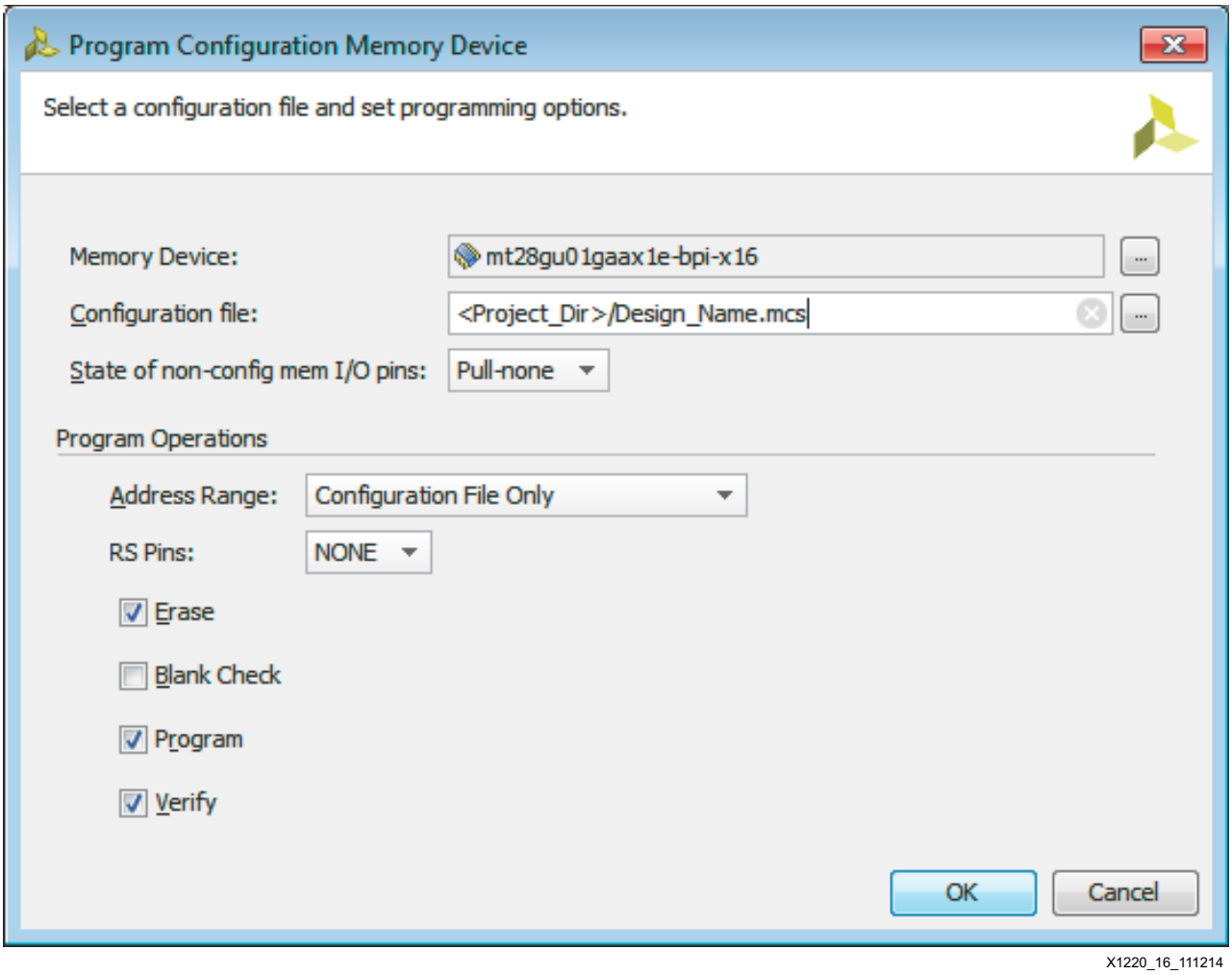


*Figure 16:* **Programming Parallel NOR Flash**

***Note:*** In most instances the default setting should be used for the **State of non-config mem I/O pins**. Designs that require the user I/O to all be pulled up or pulled down during indirect programming can use the alternate selections.

After programming has completed successfully, pulse the PROGRAM_B signal Low on the FPGA, or power cycle the board. The FPGA now configures the design from the parallel NOR flash over BPI configuration mode with synchronous read. See the BPI Configuration Sequence, page 24 for additional details. Refer to *UltraFast Design Methodology Guide for the Vivado Design Suite* [Ref 7] for more information.

## Programming Times

Sample operation times are provided for compressed and uncompressed XCVU095 bitstreams using the USB cable at 15 MHz (see Table 2). These values are for reference only and not guaranteed timing.

*Table 2:* **Reference Sample Flash Operation Times**

| Image | Times (in seconds) | | |
|---|---|---|---|
| | Flash Erase(1) | Flash Program | Flash Verify |
| XCVU095 sample compressed (40,623,464 bits) | 28 | 53 | 13 |
| XCVU095 uncompressed (286,746,912 bits) | 27 | 321 | 47 |
| Full 1 Gb image (1,073,741,824 bits) | 222 | 1,221 | 197 |

**Notes:**

1. Erase times vary based upon the amount of flash data stored at the time of erasing.

## Indirectly Program Parallel NOR Flash Tcl Console Example

This section of code lists the Tcl commands that result from following the steps in Vivado IDE Example to Program Parallel NOR Flash, page 19.

```
# Vivado script to program a parallel NOR flash
# The board should be connected to a programming cable and powered prior to running a script.
# The programming file is specified by the property PROGRAM_FILES in this example.
# Run this script from a Vivado command prompt: vivado -mode batch -source program_bpi.tcl

open_hw
connect_hw_server -url localhost:3121

# Set the current FPGA target. If multiple devices are in the JTAG chain, use the get_hw_devices
# command to help set the target FPGA.
# For setups with multiple cable connections, the user would have to select a specific target cable.
# Available cable frequencies are dependent on target cable. If a non-default frequency is desired
# this can be specified with the set_property PARAM.FREQUENCY. See UG908.

open_hw_target
current_hw_device [lindex [get_hw_devices] 0]

# Select target flash

create_hw_cfgmem -hw_device [lindex [get_hw_devices] 0] -mem_dev [lindex [get_cfgmem_parts
{mt28gu01gaax1e-bpi-x16}] 0]

# Set the address range for flash operations to the size of the programming file.
set_property PROGRAM.ADDRESS_RANGE {use_file} [ get_property PROGRAM.HW_CFGMEM [lindex
[get_hw_devices] 0 ]]

# Set the flash programming file
set_property PROGRAM.FILES {C:/Vivado_Workspace/Design/Design_Name.mcs} [ get_property
PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]

# Set the termination of unused pins when programming the flash
set_property PROGRAM.UNUSED_PIN_TERMINATION {pull-none} [ get_property PROGRAM.HW_CFGMEM [lindex
[get_hw_devices] 0 ]]
```

```
# Set the Revision Select pins, if unused the setting will be none.
set_property PROGRAM.BPI_RS_PINS {none} [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices]
0 ]]

# Set programming options for erase, program and verify. Blank check is not performed in this sample
# example.
set_property PROGRAM.BLANK_CHECK 0 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
set_property PROGRAM.ERASE 1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
set_property PROGRAM.CFG_PROGRAM 1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
set_property PROGRAM.VERIFY 1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]

# Check to ensure FPGA selected and supported flash memory are selected
startgroup
if {![string equal [get_property PROGRAM.HW_CFGMEM_TYPE [lindex [get_hw_devices] 0]] [get_property
MEM_TYPE [get_property CFGMEM_PART [get_property PROGRAM.HW_CFGMEM
[lindex [get_hw_devices] 0 ]]]]] } { create_hw_bitstream -hw_device [lindex [get_hw_devices] 0]
[get_property PROGRAM.HW_CFGMEM_BITFILE [ lindex [get_hw_devices]
0]]; program_hw_devices [lindex [get_hw_devices] 0]; };

# Program the flash
program_hw_cfgmem -hw_cfgmem [get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
refresh_hw_device [lindex [get_hw_devices] 0]
endgroup
```

# BPI Configuration Sequence

When the parallel NOR flash programming has completed successfully and the board has been power cycled or the PROGRAM_B reset signal pulsed, the BPI configuration sequence begins. This section discusses the basic sequence briefly, for details see the *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1].

In the BPI configuration mode (M[2:0] = 010), after the UltraScale FPGA samples the mode pins, it initializes and releases INIT_B. After the INIT_B signal is released and the control signals FCS_B, FOE_B, and ADV_B are asserted with a valid address A[28:00] being incremented, data is captured from the flash on the data bus D[15:0]. Using the default internal configuration clock CCLK, the bitstream header is read to determine which read mode is targeted by the design. When a synchronous command is read in the bitstream header, the FPGA configuration controller performs an asynchronous write to the Read Configuration register (RCR) of the connected flash to set the synchronous mode bit and latency bits. After the flash RCR has been successfully written, the FPGA controller begins to read the bitstream header. The bitstream header determines if the internal configuration clock is used or if the configuration clock switches to a user selection (i.e., EMCCLK). When EMCCLK is selected, a synchronous read at the EMCCLK rate is performed on the bitstream data. After configuration completes, the flash is left in synchronous read mode.

The timing waveform in Figure 17 shows the steps to initiate the BPI configuration synchronous read.

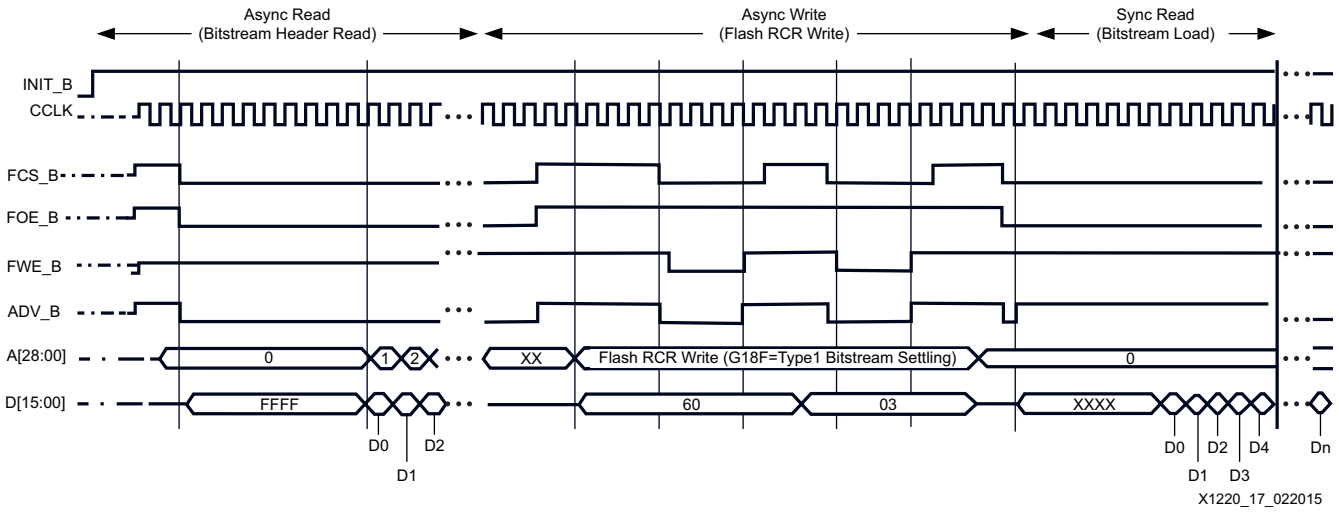


*Figure 17:* **Master BPI Configuration Mode Synchronous Read Waveform**

# Configuration Times

The BPI configuration setup in this application note uses EMCCLK. To calculate the maximum EMCCLK frequency, both the parallel NOR flash clock-to-out specification and the FPGA setup data sheet specifications are used. The maximum supported EMCCLK frequency ($F_{EMCCK}$) is specified in the UltraScale family data sheets [Ref 4] and [Ref 5], and must not be exceeded. An estimation for the maximum BPI configuration EMCCLK can be calculated with Equation 1.

*Equation 1*

$$Frequency_{MAX} = \frac{1}{a + b + c}$$

where:

- $a$ = Flash clock-to-out ($T_{CHQV}$)
- $b$ = FPGA data setup ($T_{BPIDCC}$)
- $c$ = Propagation delay between flash I/O pins and FPGA I/O pins

For example, if $T_{CHQV}$ is 5.5 ns and $T_{BPIDCC}$ is 3.5 ns and board delay is negligible, the maximum frequency is 111 MHz. See the flash data sheet and FPGA data sheet for actual $T_{CHQV}$ and $T_{BPIDCC}$ specifications.

The BPI configuration time is given with 90 MHz EMCCLK compared to the BPI configuration mode with asynchronous read and CCLK on the same board in Table 3. The calculation used to estimate the configuration time is given in Equation 2. If the configuration time from power on reset is required, then the $T_{POR}$ value from the FPGA data sheet should be added to this time from Equation 2. For applications sensitive to power-on reset time, refer to Table 1 for details on using POR_OVERRIDE to reduce the $T_{POR}$ time.

*Equation 2*

$$ConfigurationTime = \frac{a}{b \times c}$$

where:

$a$ = bitstream size
$b$ = configuration clock frequency
$c$ = data bus width

Table 3 shows sample BPI configuration times.

*Table 3:* **Sample BPI Configuration Times for Virtex UltraScale VU095 FPGA**

| Read Mode | Configuration Clock Source | Flash Data Width | Configuration Time for Bitstream (286,746,912 bits) |
|---|---|---|---|
| Synchronous read (reference example) | EMCCLK (at 90 MHz oscillator) | x16 | 199.13 ms |
| Asynchronous read | CCLK = 6 MHz | x16 | 3 s |
| Asynchronous read | CCLK= 3 MHz (default setting) | x16 | 6 s |

For the asynchronous read calculation, additional parameters must be considered and the CCLK tolerance ($F_{MCCKTOL}$) would limit the bitstream frequency setting to the 6 MHz typical.

Refer to Debug Guidance for tips on configuration.

# Debug Guidance

This section summarizes a checklist and common debug steps for the BPI configuration mode and indirect flash programming.

## Verify Schematic Connectivity

- UltraScale FPGA data pins D[03:00] are in bank 0, while D[15:04] are multi-function pins in bank 65. Ensure that both sets of pins are operating on the same voltage.
- Tie CFGBVS to GND and apply a 1.8V supply to $V_{CCO\_0}$ and $V_{CCO\_65}$ for compatibility with the 1.8V Micron 28F00AG18F flash.
- The JTAG pins are in bank 0 and they need follow the same requirements for this bank. For the Virtex UltraScale FPGA, the bank is set to 1.8V.
- Review master BPI configuration mode connectivity diagram.

  ***Note:*** The FPGA address bus is A[25:0] while the Micron 28F00AG18F flash address bus is A[26:1]. Most connections between the FPGA and flash appear offset by one because of the different bus labeling.

- Some parallel NOR flash families supported by x16 use the A[1] as the address LSB signal while others use A[0].
- It is very important that the FPGA signal integrity of the JTAG TCK and FPGA CCLK signals are maintained. Avoid using lengthy connections where possible. Using long connections can result in unwanted noise or voltage waveform reflections which degrade the integrity of FPGA signals. See the *Unidirectional Topographies and Termination* section of the *UltraScale Architecture PCB Design User Guide* (UG583) [Ref 6] for additional guidance.

## Ensure Correct File Generation

- Check the MCS file is correctly generated with the interface set to BPIx16 and the bitstream loaded upward from address `0x0`.
- When the EMCCLK option is enabled in bitstream settings, ensure that the I/O standard is defined. EMCCLK is a multi-function pin, so by default its I/O standard is undefined. If the design uses the EMCCLK pin and has properly constrained the EMCCLK I/O standard in the constraint file, no further action is required. If the EMCCLK pin is only used for configuration clocking purposes, include the following:

```
set_property CONFIG_VOLTAGE 1.8 [current_design]
```

- For BPI configuration, ensure that the synchronous mode type (Type1, Type2) is set appropriately according to the flash target device (Type1 for 28F00AG18F, Type2 for P30).
- Optionally, for faster configuration and programming times, use the bitstream compression setting.
- When using the internal oscillator source for the master CCLK, ensure the ConfigRate option does not exceed the maximum frequency supported by the target flash and FPGA for asynchronous read mode. Refer to the UltraScale data sheet specification for the FPGA CCLK tolerance $F_{MCCKTOL}$ [Ref 4] and [Ref 5].
- When applying bitstream settings in the Project Settings window (Figure 7, page 12), ensure that the design has successfully run synthesis and that the synthesized design is open. Vivado Design Suite does not allow applying advanced bitstream settings until a synthesized or implemented design is open.
- If the Vivado Design Suite cannot find the bitstream image, ensure that the file path for the flash programming file does not contain spaces, and when using Windows, does not exceed 260 characters. The Tcl command line containing the file path attempts to get the flash programming file after it finds a blank space.

## Steps to Perform if Indirect Parallel NOR Flash Programming is Unsuccessful

- Flash devices are nonvolatile. Ensure the flash was erased before programming.
- Use the Blank Check operation for verifying a prior erase operation.
- Ensure the correct flash memory part is selected in the Vivado Hardware Manager for your design. See the flash manufacturer data sheet for complete part naming information.

- Reduce the target cable frequency if a flash operation failure is seen. This reduction can help isolate a board signal integrity issue.
- Ensure the JTAG chain integrity is good:
  - Perform a basic FPGA IDCODE operation to verify the connections. If no target hardware is shown when connecting to the cable, verify that the cable is plugged in and the board is powered.
  - Program a simple bitstream into the FPGA.
  - If a flash indirect programming operation does not complete successfully, ensure the Common Flash memory Interface (CFI) Manufacturer ID and memory type read matches the expected family targeted. The Vivado Design Suite captures the CFI Manufacturer ID (Mfg ID) as the first step of any flash operation performed. For the 28F00AG18F, the Vivado Design Suite should report:

    Mfg ID : 89  Memory Type : 88b0  Device ID 1 : 0  Device ID 2: 0

    If the results from the Vivado Design Suite do not match the data sheet, ensure the right flash family and density were selected.
  - Capture the FPGA Status Register for additional insight.

## Steps to Perform if BPI Configuration is Unsuccessful

- If configuration was unsuccessful on a power cycle, try pulsing the PROGRAM_B signal to ensure there is not an issue with the order of the power-up sequence between the FPGA and flash devices.
- If configuration does not complete within the expected time, ensure that the write_bitstream settings are set for the read mode desired and the configuration clock. The default asynchronous read and internal default configuration clock run much slower than a bitstream with the synchronous read and EMCCLK options enabled.
- Try a basic asynchronous bitstream first with the internal CCLK and default bitstream settings. This step can be used to isolate signal integrity or bitstream property issues. The time it takes to configure the FPGA with the default configuration clock property (3 MHz) is significantly longer than a bitstream with synchronous read and EMCCLK enabled. See Table 3 for example configuration time.
- Check the MCS file using the Ensure Correct File Generation section.
- Use Vivado Device Programmer to check the configuration status registers.
- Use Vivado Device Readback the flash data to check that the flash contains the expected bitstream pattern.

# References

1. *UltraScale Architecture Configuration User Guide* (UG570)
2. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
3. Micron StrataFlash Embedded Memory MT28GU01GAAA1E (28F00AG18F) data sheets
4. *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS892)
5. *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893)
6. *UltraScale Architecture PCB Design User Guide* (UG583)
7. *UltraFast Design Methodology Guide for the Vivado Design Suite* (UG949)

# Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 03/16/2022 | 1.2 | In Summary, added a note about Master BPI configuration mode with synchronous read not being recommended for new designs. |
| 03/18/2015 | 1.1 | Revised Figure 17. Additional details are provided for POR_OVERRIDE and EMCCLK usage. |
| 12/08/2014 | 1.0 | Initial Xilinx release. |

# Please Read: Important Legal Notices