



XAPP1282 (v1.1) February 8, 2022

# UltraScale FPGA Post-Configuration Access of Parallel NOR Flash Memory using STARTUPE3

Authors: Steven Howell, Shashikant Jadhav, and Stephanie Tapp

## Summary

Parallel NOR flash memory is a popular UltraScale™ FPGA configuration solution. The value of this solution is increased when it is used post-configuration to store non-volatile user data or to remotely update configuration images. This application note demonstrates post-configuration access between a Virtex® UltraScale FPGA and parallel NOR flash memory provided by a VCU108 evaluation board. Vivado® Design Suite 2016.1 is the development environment.

You can download the [Reference Design Files](#) for this application note from the Xilinx® website. For detailed information about the design files, see [Reference Design](#).

## Introduction

The reference design in this application note uses a MicroBlaze® processor core to interface to the AXI external memory controller (AXI EMC) core and the STARTUPE3 primitive to implement post-configuration read and write access through a dedicated BPI configuration interface to the on-board parallel NOR flash memory. [Figure 1](#) shows operation of the post-configuration reference design.

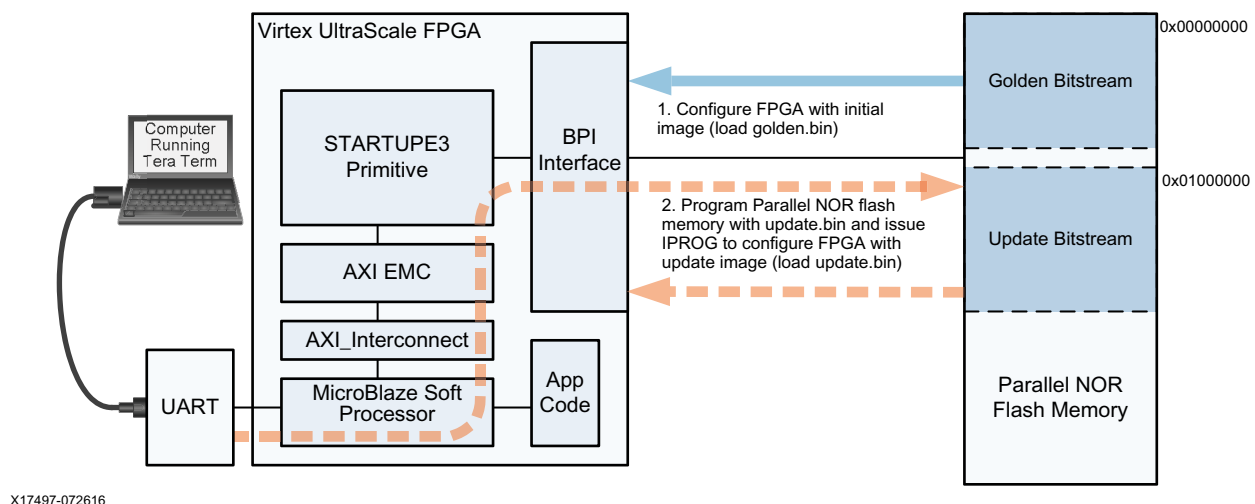


Figure 1: Reference Design Post-Configuration Access Flow

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing noninclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this [link](#) for more information.



**IMPORTANT:** *This application note is not suitable for use with an update image that is a partial bitstream. Programming a partial bitstream uses a shutdown command that disables the USRCCLK0/USRCLKTS connection from the STARTUPE3 block and disrupts parallel NOR flash memory access.*

When the BPI configuration mode is used to configure the FPGA, the initial bitstream image loads from the parallel NOR flash memory. After the FPGA is initially configured, the BPI configuration interface typically remains unused. However, the unused space in the parallel NOR flash memory can be used to store additional configuration images or application data, eliminating the cost of adding extra memory and reducing the required board space. [Figure 1](#) shows how the reference design executes this flow:

- Step 1 configures the FPGA using the golden bitstream image (`golden.bin`) stored in the parallel NOR flash memory. The golden bitstream image includes the STARTUPE3 primitive, interface logic, IP cores, and constraints to enable reading from and writing to the unused storage in the parallel NOR flash memory.
- Step 2 runs application code on the MicroBlaze processor to download a new update bitstream from the computer via Tera Term Xmodem protocol. Each 1024-byte data packet is checked using CRC, and then written into parallel NOR flash. Then an IPROG operation is performed and the FPGA is reconfigured from the update bitstream image (`update.bin`) replacing the original golden bitstream image (`golden.bin`).

## Document Organization

- [Introduction](#): Describes the overall reference design operation.
- [System Overview](#): Provides the reference design block diagram, file structure, and IP core addresses.
- [Running the Reference Design](#): Lists the hardware and software required to run the reference design and describes how to:
  - [Set Up Host Computer](#)
  - [Set Up VCU108 Evaluation Board](#)
  - [Download Reference Design Files](#)
  - [Generate Reference Design Project](#)
  - [Generate Programming Files](#)
  - [Configure FPGA with `golden.bin`](#)
  - [Program Parallel NOR Flash Memory with `update.bin`](#)
  - [Configure FPGA with `update.bin`](#)
- [Hardware System Details](#): Describes the hardware clock topology, AXI EMC core, and STARTUPE3 primitive details. Core customization, timing, and constraints are also detailed.
- [Software System Details](#): Describes reference design software and the flash command set.
- [Checklist and Debug Tips](#): Highlights settings which should be verified to ensure successful operation of the reference design. Provides tips to correct common oversights.

- [Conclusion](#): Summarizes the value of accessing the parallel NOR flash memory post-configuration.

## Recommended Design Experience

A general knowledge of:

- The UltraScale FPGA BPI configuration mode  
*See the [UltraScale Architecture Configuration User Guide](#) (UG570) [Ref 1]*
- Vivado Design Suite  
*See [Vivado Design Suite Quick Reference Guide](#) (UG975) [Ref 2]*

---

## System Overview

Software running on the MicroBlaze processor core drives the AXI EMC core to read and write to parallel NOR flash memory through the STARTUPE3 primitive in asynchronous mode. The software presents a command menu to the host computer running Tera Term through the AXI UART Lite interface. The menu provides commands to allow erasing, programming, and verifying parallel NOR flash memory content. [Software System Details](#) describes the flash command set and software flow.

Vivado Design Suite IP Integrator (IPI) is used to create a block diagram with the cores. The IPI block design and the STARTUPE3 primitive are instantiated within the top-level wrapper design file (`design_1_wrapper.vhd`) shown in [Figure 2](#).

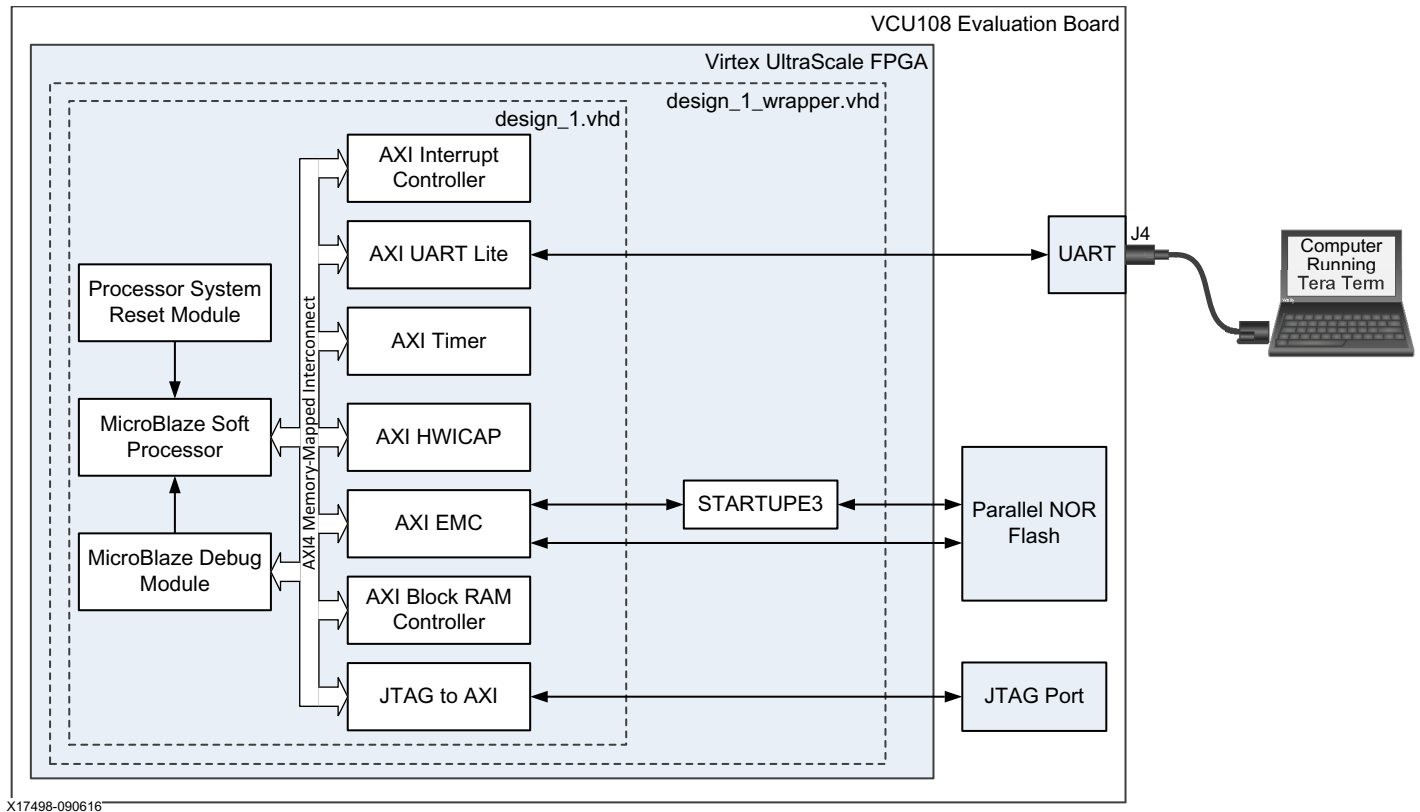


Figure 2: Post-Configuration Reference Design System Block Diagram

## Reference Design Files

Project script files:

- `run_vcu108(.bat/.tcl)` (creates reference design project and launches SDK)
- `make_download_files(.bat/.tcl)` (creates download `.bin` files (golden.bin, update.bin))
- `load_golden(.bat/.tcl)` (programs `golden.bin` into parallel NOR flash memory using the Vivado device programmer)

Reference design files:

- `design_1_wrapper.vhd` (top-level hardware file with the STARTUPE3 primitive) includes:
  - `design_1.vhd` (board design hardware wrapper)
  - `design_1.bd` (board description file)
- `vcu108.xdc` (constraints file that includes STARTUPE3 timing constraints and pin locations)
- binary configuration files:
  - `golden.bin` (pre-generated initial bitstream image)
  - `update.bin` (pre-generated update bitstream image)

- SDK source code files:
  - `vcu108_emc_rw.c` (main loop, Xmodem download, and flash update functions)
  - `vcu108_emc_rw.h`
- `lscript.ld` (linker script)
- `vcu108_emc_rw.elf` (generated software image)

## IP Core Address Map

Table 1: IP Core Addresses

IP Core	Version	Input Clock Frequency	Offset Address	Range	High Address
MicroBlaze Processor ( <code>microblaze_1</code> )	9.6	100 MHz		N/A	N/A
AXI Block RAM Controller ( <code>axi_bram_ctrl_1</code> )	4.0		0xC000_0000	1 MB	0xC003_FFFF
AXI Interrupt Controller ( <code>microblaze_1_axi_intc</code> )	4.1		0x4120_0000	64 KB	0x4120_FFFF
AXI UART Lite ( <code>axi_uartlite_0</code> )	2.0		0x4060_0000	64 KB	0x4060_FFFF
AXI Interconnect ( <code>axi_mem_intercon</code> )	2.1			N/A	N/A
AXI_TIMER ( <code>axi_timer_0</code> )	2.0		0x41C0_0000	64 KB	0x41C0_FFFF
JTAG_AXI ( <code>jtag_axi_0</code> )	1.1			N/A	N/A
AXI EMC ( <code>axi_emc_1</code> )	3.0	50 MHz	0x4800_0000	128 MB	0x4FFF_FFFF
AXI_HWCAP ( <code>axi_hwicap_0</code> )	3.0		0x4020_0000	N/A	0x4020_FFFF
Processor System Reset Module ( <code>proc_sys_reset</code> )	5.0			N/A	N/A

## Parallel NOR Flash Memory Map

The reference design stores two images in the parallel NOR flash memory (shown in [Figure 1](#)):

- Golden Bitstream starting at 0x00000000
- Update Bitstream starting at 0x01000000

## Running the Reference Design

This section describes the [Required Hardware and Software](#) and how to:

- [Set Up Host Computer](#)
- [Set Up VCU108 Evaluation Board](#)
- [Download Reference Design Files](#)
- [Generate Reference Design Project](#)
- [Generate Programming Files](#)
- [Configure FPGA with `golden.bin`](#) (Corresponds to step 1 in [Figure 1](#).)

- [Program Parallel NOR Flash Memory with update.bin](#) (Corresponds to step 2 in [Figure 1](#))
- [Configure FPGA with update.bin](#) (Corresponds to step 2 in [Figure 1](#).)

## Required Hardware and Software

- VCU108 evaluation board which includes:
  - Virtex UltraScale XCVU095-2FFVA2104 FPGA (U1)
  - Micron MT28GU01GAAA1EGC 1 Gb (256 MB) Parallel NOR flash (U133)
- Power supply: 100–240 VAC input, 12 VDC 5.0A output (included with the VCU108 evaluation kit)
- Two USB cables, standard-A plug to micro-B plug
- Host computer with:
  - Two USB ports
  - Windows operating system supported by Vivado Design Suite
- Vivado Design Suite 2016.1, Design Edition with SDK
- Tera Term terminal emulator program (version 4.90 was used for reference design testing)
- Silicon Labs Dual CP210x USB UART Drivers
- [Reference Design Files](#)

## Set Up Host Computer

If not already installed:

1. Install Vivado Design Suite version 2016.1.
2. Download and install Tera Term (version 4.90 was used for reference design testing). Follow the procedure in *Tera Term Terminal Emulator Installation Guide* (UG1036) [\[Ref 3\]](#).
3. Download and install the UART drivers. Follow the instructions in *Silicon Labs CP210x USB-to-UART Installation Guide* (UG1033) [\[Ref 4\]](#)



---

**TIP:** The UART communication settings are set up later in this procedure.

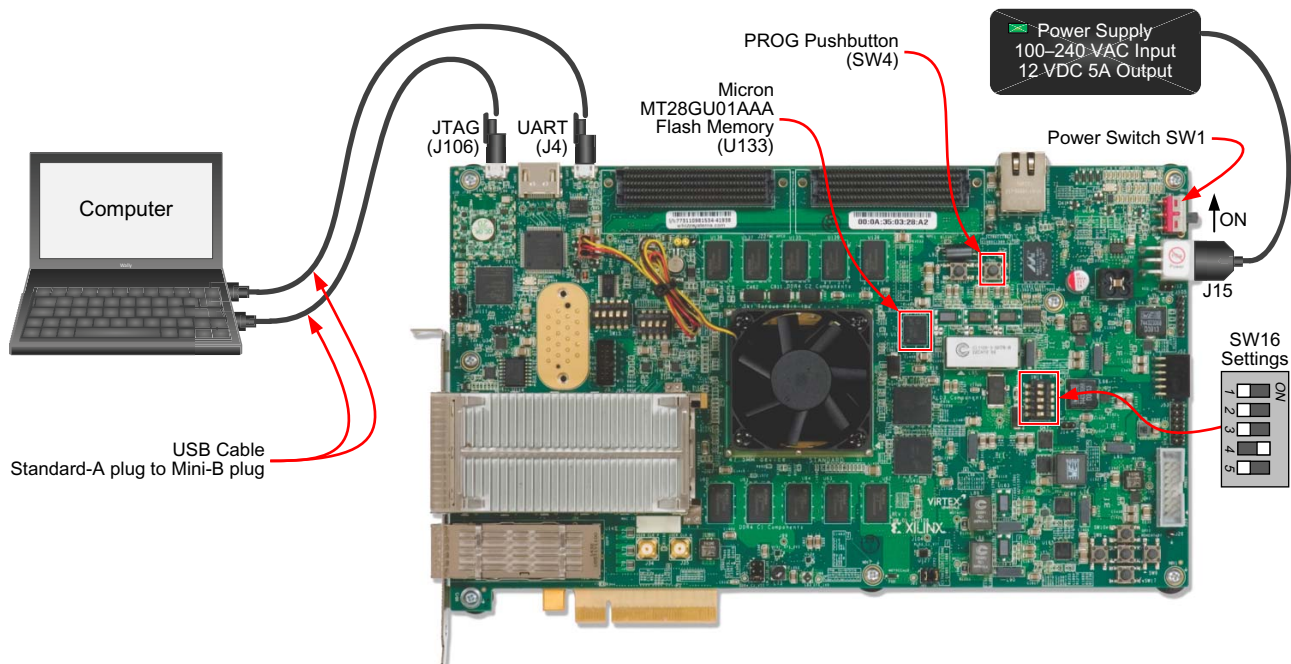
---

## Set Up VCU108 Evaluation Board

Referring to [Figure 3](#):

1. Place switch SW1 to the OFF position and connect the power supply to J15.
2. Connect the USB cables between the computer and the UART connector (J4) and JTAG connector (J106) on the VCU108 board.

- Set the FPGA DIP switch SW16 as shown in [Figure 3](#). SW16[3:5] select the BPI mode  $M[2:0] = 010$ . The parallel NOR flash upper address bits (A25 and A24) connected to the FPGA RS[1:0] are set by SW16[1:2].



X17499-072616

Figure 3: Connection Diagram for Preliminary Setup

## Download Reference Design Files

- Download and unzip the [Reference Design Files](#) to `c:\xapp1282`.

## Generate Reference Design Project

1. Go to `c:\xapp1282`. Double-click `run_vcu108.bat` to run the batch file.

This batch file generates the Vivado IPI hardware project (`project_1.xpr`) and exports the created hardware to SDK (`project_1.sdk`) under the `project_1` directory. [Figure 5](#) shows the project directory addition.



---

**TIP:** *The project build takes approximately 30 minutes depending on host computer.*

---

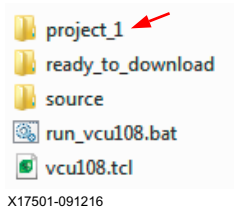
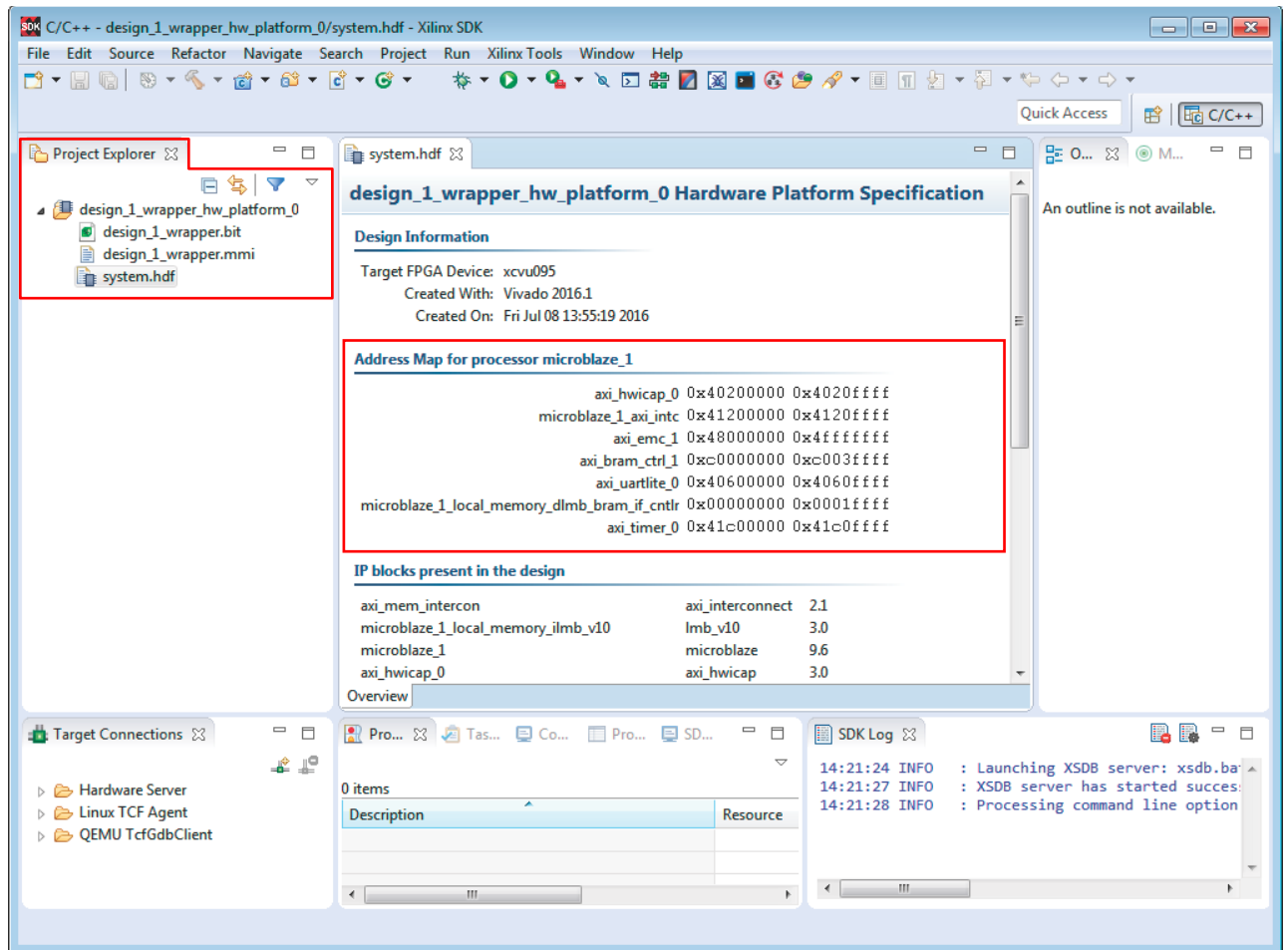


Figure 5: New Project Directory



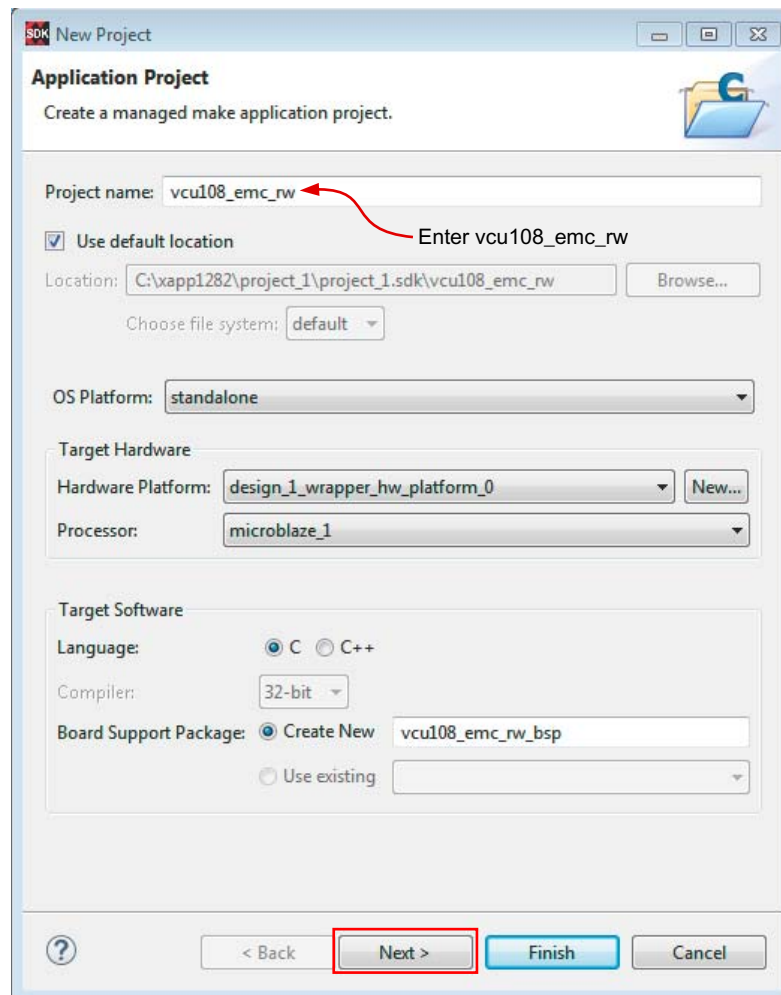
When the build is complete, the batch file opens the SDK Integrated Design Environment (IDE). [Figure 6](#) shows the project information in the IDE including the `system.hdf` file.



X17502-072616

**Figure 6: Project in the SDK IDE**

2. Create an application for the parallel NOR flash memory read and write access:
  - a. Select **File > New > Application Project** to open the new Application Project window.
  - b. In the **Project Name** field, Enter vcu108\_emc\_rw.
  - c. Select, enter, or verify the values in the remaining fields shown in [Figure 7](#).

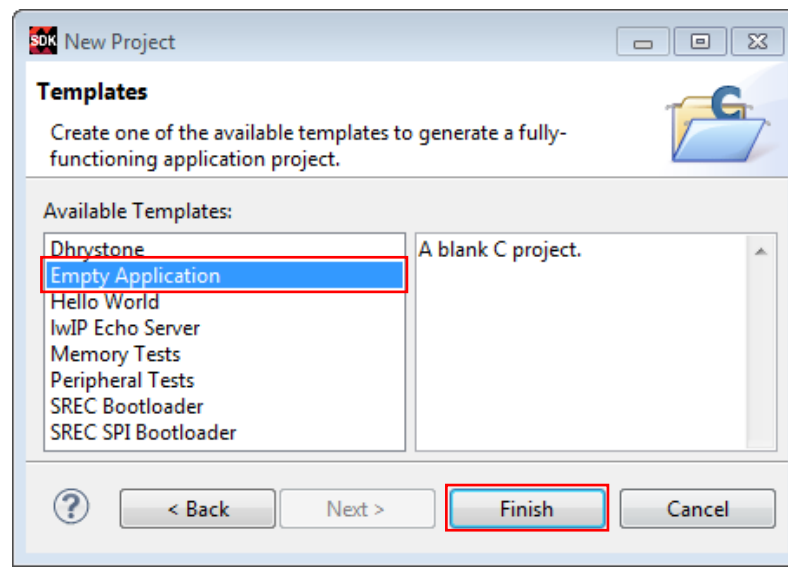


X17503-091516

*Figure 7: Create Application Project*

- d. Click **Next** to open the template window.

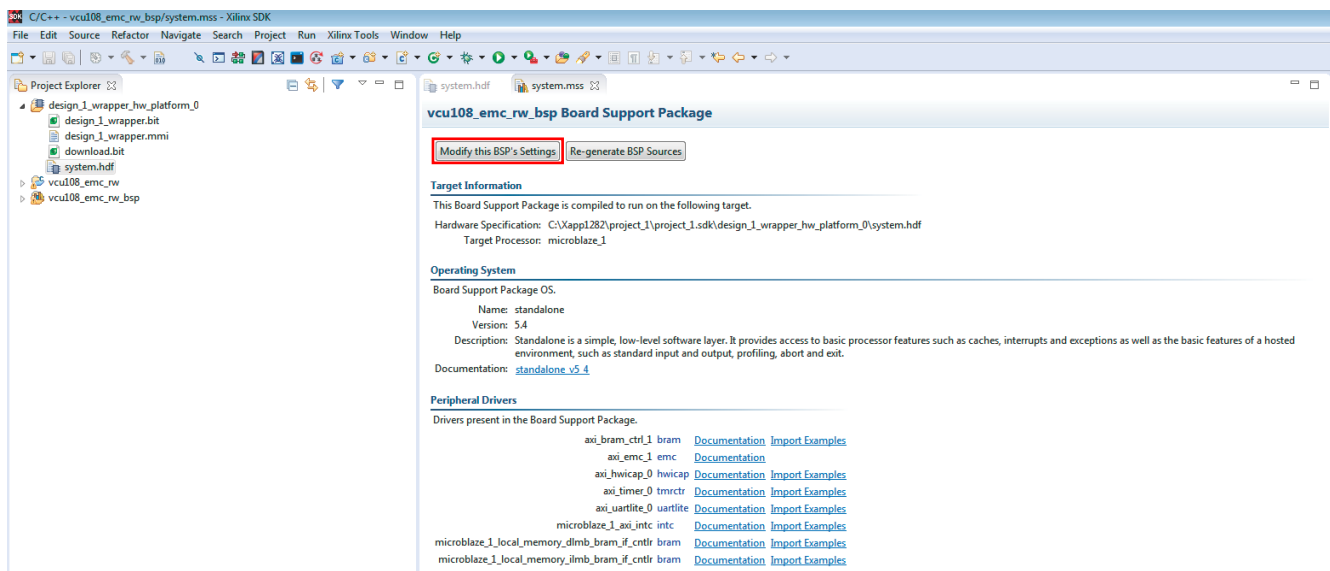
- e. Under Available Templates select **Empty Application** (Figure 8).



X17504-072616

Figure 8: Available Templates

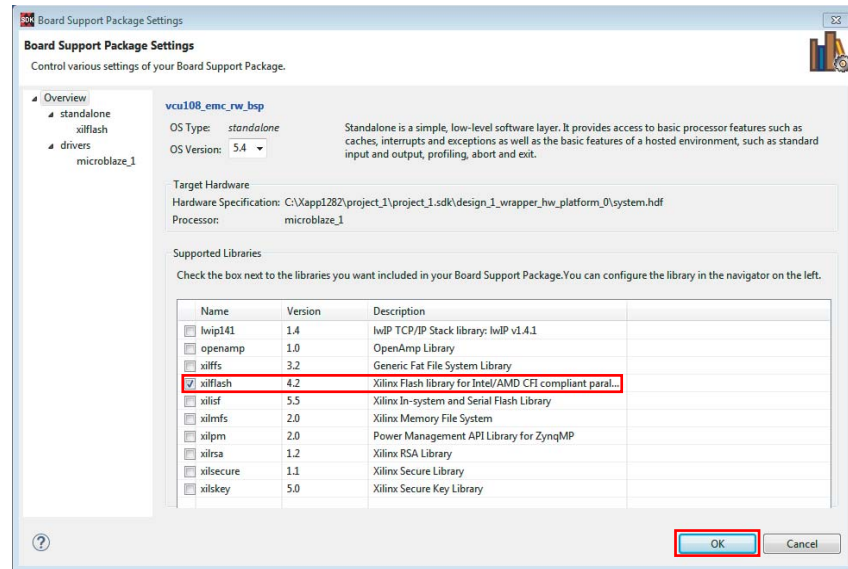
- f. Click **Finish** to build the application project.
- g. After the project build is complete, select **Modify this BSP's Settings** as shown in Figure 9.



X17505-072616

Figure 9: Modify BSP Settings

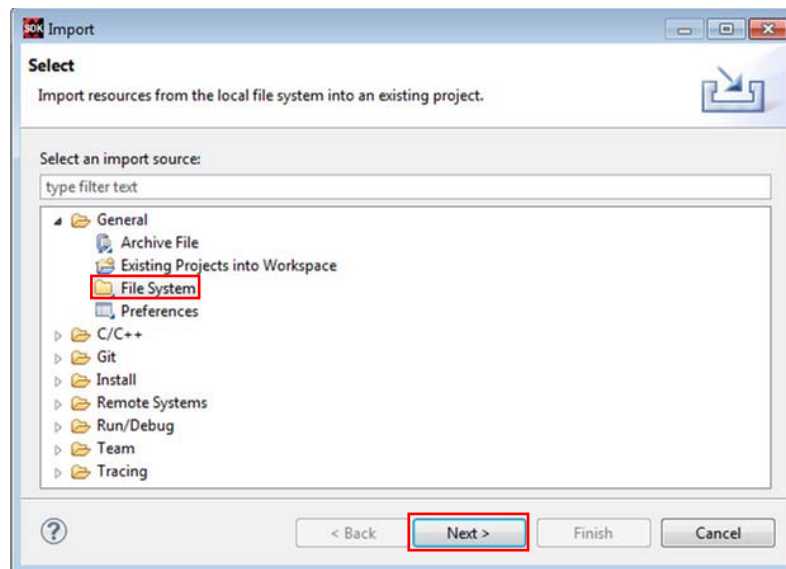
- h. In the Board Support Package Settings window shown in [Figure 10](#), select the Xilinx Flash Library **xilflash** and click **OK**. This step includes the software functions necessary to communicate with the parallel flash memory. The BSP is regenerated and recompiled.



X17506-072616

**Figure 10: Board Support Package Settings**

- i. In the SDK IDE Project Explorer tab, double-click the folder **vcu108\_emc\_rw** and select **src**. Right-click on **src** and select **Import**.
- j. In the Import window, expand the **General** folder and select **File System** ([Figure 11](#)).



X17507-072716

**Figure 11: Selecting the Local File System as an Import Resource**

- k. Click **Next**.

- l. Browse to the `C:\xapp1282\source` directory, click **OK**, and select the highlighted files shown in [Figure 12](#).

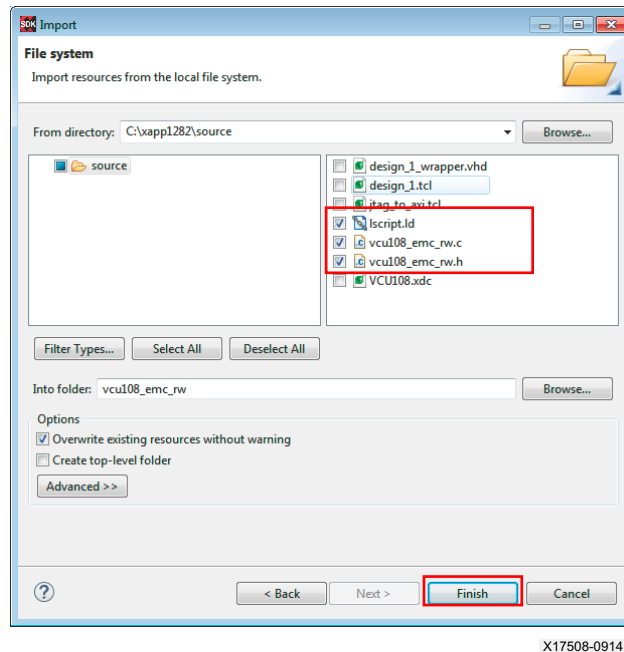


Figure 12: Import Files from the Local File System

- m. Click **Finish** to add all source files to the project.
3. (Optional Step) Review the MicroBlaze code: In the SDK project explorer, double-click **vcu108\_emc\_rw.c** under **src** to open the file and review some of the code. This file contains the main function as well as the code that presents the Tera Term command menu.
    - Near the top of the file are a set of definitions for the flash memory width, the number of bytes, blocks, pages, and the sizes of the blocks and pages in the flash memory. Flash memory test addresses for some of the test functions below are also defined. If the reference design is used with other parallel flash memory types, these definitions must be altered to match that flash.
    - Below the flash definitions are a set of definitions for the XMODEM protocol download function that brings the `update.bin` file from the computer to the MicroBlaze processor for writing into the flash memory. These define the appropriate characters for XMODEM-CRC (1K), as well as some of the states for the function, and some additional Tera Term commands to create the user interface.
    - A set of flash memory functions used in the reference design (and built from the lower level functions included in the `xilflash` library) are given at the top of the code in the *Function Prototypes* section. The code for these functions is shown below the main function. The main function is a `case` structure that presents a menu on the UART, and allows the user to choose flash operations (Read ID, Erase, Program, Read) or cause an IPROG to occur.

## IPROG

To see the IPROG code, go to the function `HwIcapLowLevelIPROG` in the `vcu108_emc_rw.c` file.

- IPROG is an internal programming of the FPGA commanded by a series of register writes through the internal configuration access port (ICAP). When this operation is chosen, the MicroBlaze processor performs a series of register writes to the `axi_hwicap` peripheral that causes the IPROG to occur. After IPROG commences, the FPGA erases all of the current configuration except for the Warm Boot Start Address Register (also called WBSTAR). This register can contain a non-zero address where the FPGA begins loading the configuration image.
- In this way, the user can command a reconfiguration of the FPGA and specify a location to find the updated configuration image in parallel flash. There are numerous ways that the address bits can be set to retrieve an image programmed in flash. In this reference design, a single example is shown that stores an `update.bin` image at a particular address and then retrieve it when IPROG occurs. For additional information on the WBSTAR register refer to the section [Configure FPGA with `update.bin`](#) or the *UltraScale Architecture*.

## Generate Programming Files

1. Close the SDK GUI.
2. Go to `C:\xapp1282\ready_to_download`. Double-click `make_download_files.bat` to run the batch file. The `golden.bin` and `update.bin` bitstream images are generated in the directory.



---

**TIP:** *Generation of these files takes approximately 5 minutes depending on host computer system.*

---

## Configure FPGA with `golden.bin`

This part of the procedure corresponds to step 1 in [Figure 1](#).

1. Turn on power to the VCU108 board by placing switch SW1 to the ON position as shown in [Figure 3](#). The power good LED, DS3 (located near SW1) illuminates green to indicate the board power is on and the power system is operating properly.
2. Go to `C:\xapp1282\ready_to_download`. Double-click `load_golden.bat` to run the batch file.

After `golden.bin` (the reference design initial bitstream) has been copied into the parallel NOR flash memory at address `0x00000000`, the batch file issues an FPGA reset to initiate FPGA configuration from the parallel NOR flash memory which now contains the `golden.bin` image.



---

**TIP:** *Programming the parallel NOR flash memory takes approximately 10 minutes depending on host computer system to erase, program, and verify the memory.*

---

3. Verify the FPGA is successfully configured with the reference design initial bitstream (`golden.bin`) by observing the DONE LED is lit and the LED 0 is flashing ([Figure 13](#)).

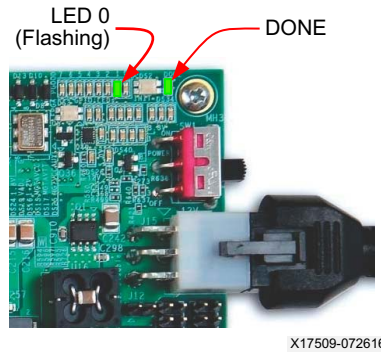


Figure 13: The Reference Design Initial Bitstream `golden.bin` is Running

## Program Parallel NOR Flash Memory with `update.bin`

This part of the procedure corresponds to step 2 in [Figure 1](#).

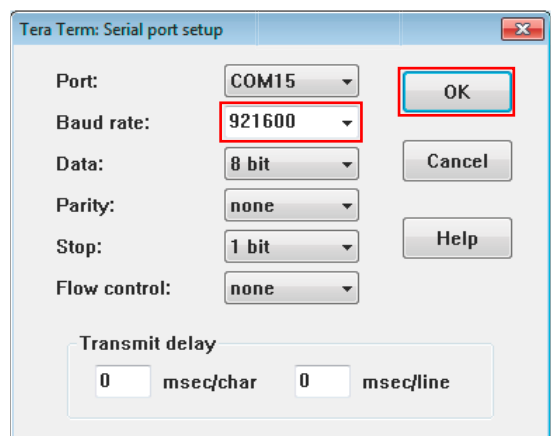
After the FPGA is running the `golden.bin` image, `update.bin` can be downloaded from the host computer and written into parallel NOR flash at address `0x01000000`, and then be used to reconfigure the FPGA.

1. Confirm the host computer and VCU108 evaluation board are connected as shown in [Figure 3](#).



**IMPORTANT:** For the operations shown in menu selection **3** (Program `<*.bin via XMODEM_1K>`) and selection **5** (Reconfigure from IPROG), enter the Programming Start Address `<hex>` value highlighted in [Figure 17](#) or [Figure 21](#). In this example, a flash memory start address of `0x01000000` is translated by the MicroBlaze processor code and written into the equivalent WBSTAR format as `0x40000000`. The automatic translation allows entering the flash memory address directly without having to calculate to the WBSTAR definition. If you intend to modify the design, or to use the design with a different board, the mapping of the flash address pins to the WBSTAR bit positions must be understood. Refer to the WBSTAR Register table in UltraScale Architecture Configuration User Guide (UG570) [\[Ref 1\]](#) for more information on WBSTAR.

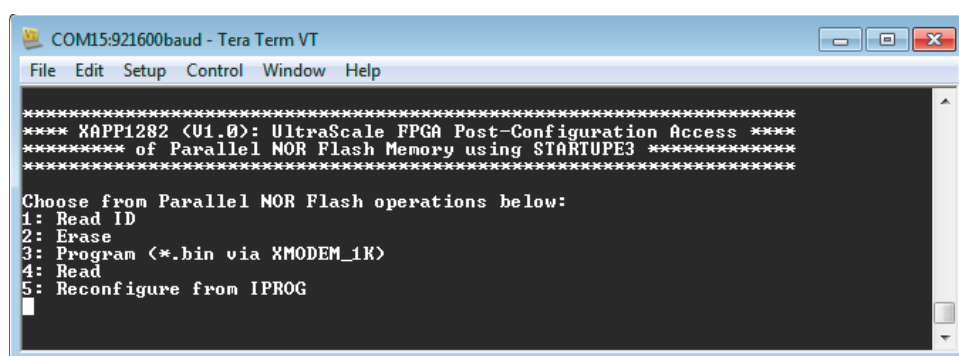
2. Launch Tera Term and under the **Setup > Serial port...** menu use the values shown in [Figure 14](#). Ensure that the Standard COM Port is selected (Refer to the Checklist and Debug Tips [step 4](#) for details).



X17510-072616

Figure 14: Tera Term Serial Port Settings

3. Pulse the PROG pushbutton (SW4 on the VCU108 board) to load the `golden.bin` code. If the serial connection set up properly, Tera Term displays the reference design menu shown in Figure 15.

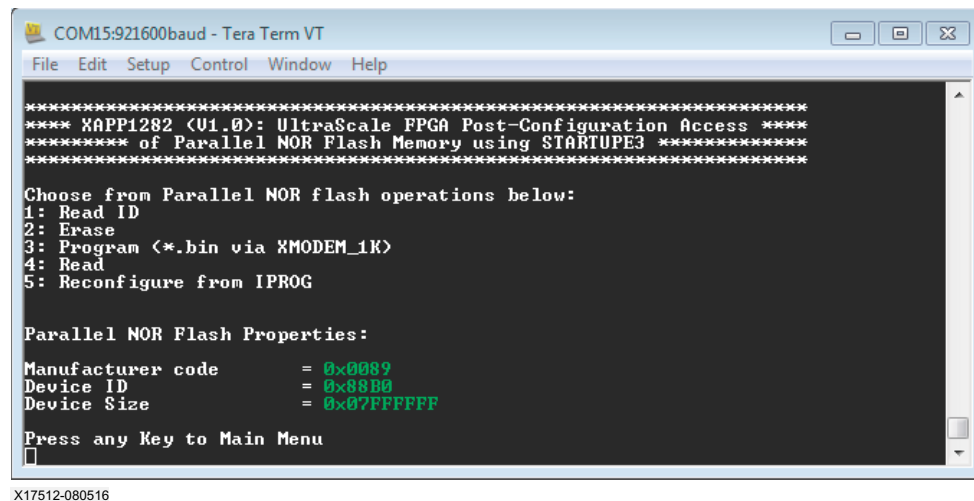


X17511-072616

Figure 15: Reference Design Flash Memory Command Menu

4. In the Tera Term window, enter **1** to display the parallel NOR flash manufacturer information and ensure proper board setup (Figure 16).





```

COM15:921600baud - Tera Term VT
File Edit Setup Control Window Help

*****
**** XAPP1282 (V1.0): UltraScale FPGA Post-Configuration Access ****
***** of Parallel NOR Flash Memory using STARTUPE3 *****

Choose from Parallel NOR flash operations below:
1: Read ID
2: Erase
3: Program (*.bin via XMODEM_1K)
4: Read
5: Reconfigure from IPROG

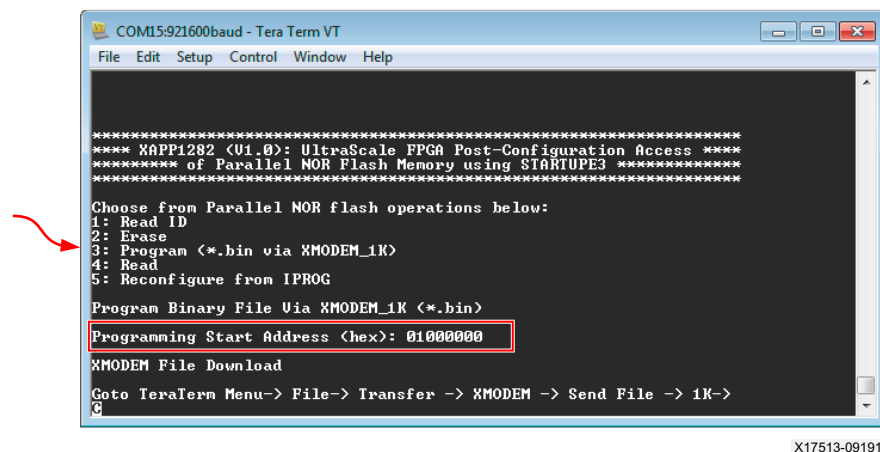
Parallel NOR Flash Properties:
Manufacturer code      = 0x0089
Device ID              = 0x88B0
Device Size            = 0x07FFFFFF

Press any Key to Main Menu

```

Figure 16: Option 1: Parallel NOR Flash Memory Information

- Enter **3** to program the parallel NOR flash memory with the update.bin image at offset (programming start address) 0x01000000 (Figure 17).



```

COM15:921600baud - Tera Term VT
File Edit Setup Control Window Help

*****
**** XAPP1282 (V1.0): UltraScale FPGA Post-Configuration Access ****
***** of Parallel NOR Flash Memory using STARTUPE3 *****

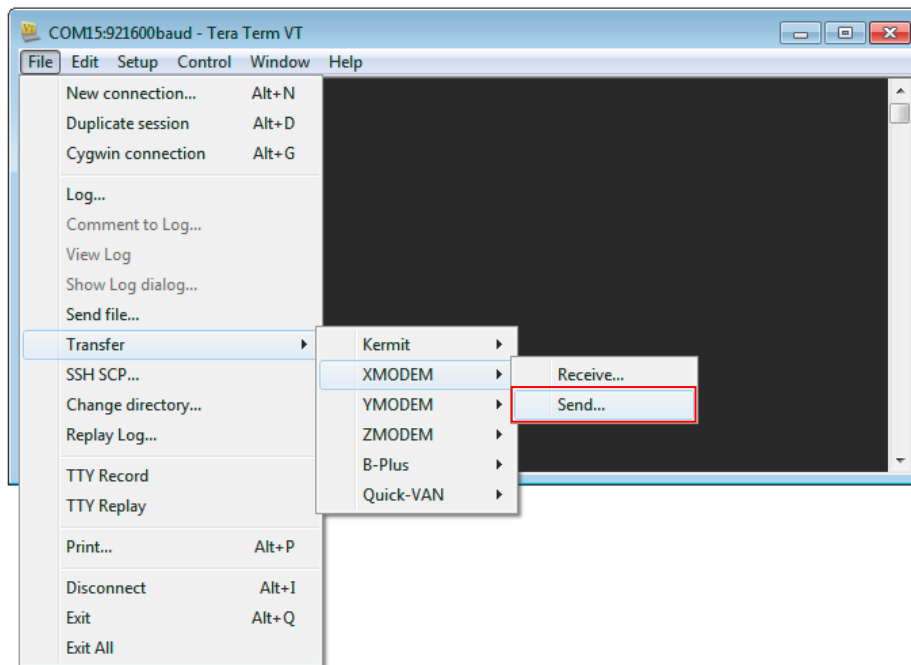
Choose from Parallel NOR flash operations below:
1: Read ID
2: Erase
3: Program (*.bin via XMODEM_1K)
4: Read
5: Reconfigure from IPROG

Program Binary File Via XMODEM_1K (*.bin)
Programming Start Address (hex): 01000000
XMODEM File Download
Goto TeraTerm Menu-> File-> Transfer -> XMODEM -> Send File -> 1K->

```

Figure 17: Option 3: Program

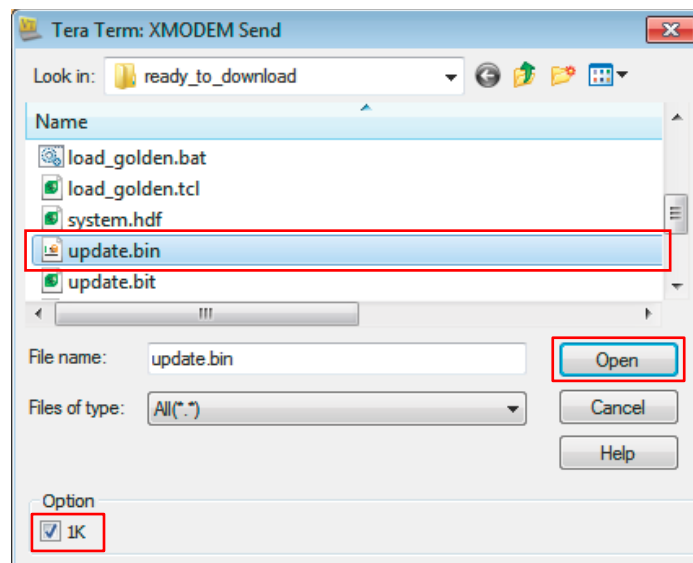
6. In the menu bar, select **File > Transfer > XMODEM > Send...** (Figure 18).



X17514-07261E

Figure 18: Send File Command

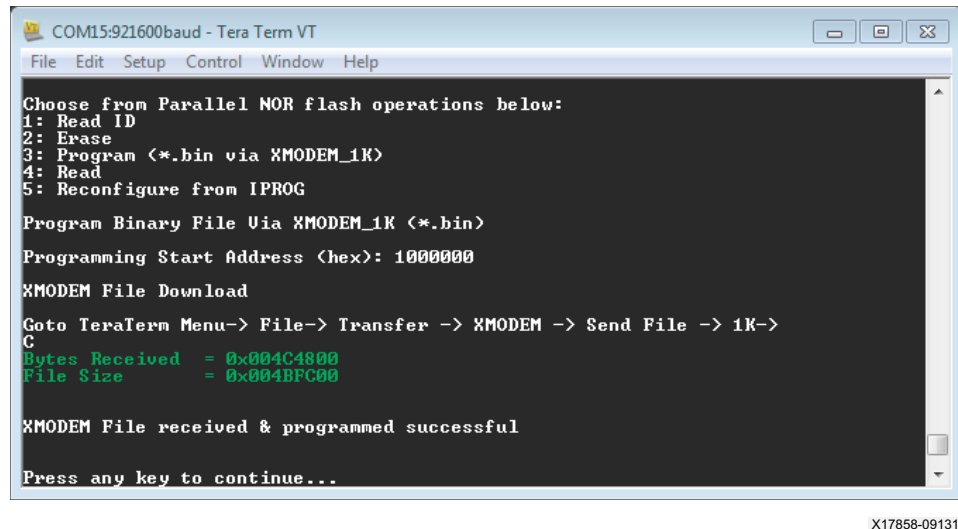
7. In the Look in field, enter **C:\xapp1282\ready\_to\_download**, select `update.bin`, select Option **1K** and click **Open** (Figure 19).



X17515-091316

Figure 19: Select `update.bin`

8. When the `update.bin` programming has successfully completed you will see the Tera Term window shown in [Figure 20](#).



```

COM15:921600baud - Tera Term VT
File Edit Setup Control Window Help

Choose from Parallel NOR flash operations below:
1: Read ID
2: Erase
3: Program (*.bin via XMODEM_1K)
4: Read
5: Reconfigure from IPROG

Program Binary File Via XMODEM_1K (*.bin)
Programming Start Address (hex): 1000000
XMODEM File Download

Goto TeraTerm Menu-> File-> Transfer -> XMODEM -> Send File -> 1K->
C
Bytes Received = 0x004C4800
File Size      = 0x004BFC00

XMODEM File received & programmed successful

Press any key to continue...

```

X17858-091316

Figure 20: Programming Status

## Configure FPGA with `update.bin`

This part of the procedure corresponds to the final part of step 2 in [Figure 1](#).

The `update.bin` (the update bitstream image) is placed in flash memory by the reference design at address `0x01000000`. After writing the image into flash memory, the reference design uses the FPGA internal configuration access port (ICAP) to reconfigure the FPGA with this new image.

The reconfiguration happens in two steps:

1. The MicroBlaze™ processor uses the AXI\_HWICAP IP core as an interface to the ICAP port to write a value into the warm boot start address register (WBSTAR). The contents of the WBSTAR specifies the flash memory starting address when the next internal programming command (IPROG) is issued to the ICAP interface.

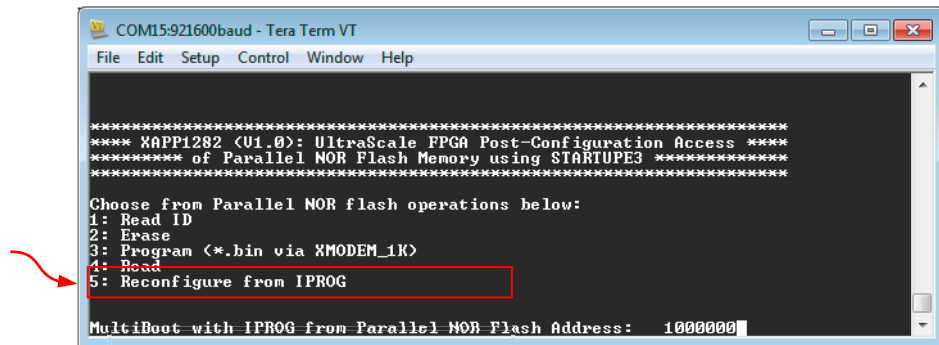
The WBSTAR register is described in *UltraScale Architecture Configuration User Guide* (UG570) [[Ref 1](#)], Table 9-31.

WBSTAR[31:30] (the upper two register bits) correspond to external FPGA pins RS[1:0]. These pins can be driven by the contents of WBSTAR or controlled using external pull-up or pull-down resistors. Bit 29 of the WBSTAR allows the RS[1:0] bits to either be 3-stated (controlled externally) or driven by the contents of WBSTAR [31:30].

The WBSTAR[28:0] bits connect to the address bits of the configuration flash, and allow the FPGA to control what address is used to load the configuration file.

When the user provides a flash memory start address in the Tera Term/UART menu, the MicroBlaze processor translates the address into a write command that sets the contents of the WBSTAR appropriately.

In the Tera Term window, select option **5**. When prompted for the parallel NOR flash address, enter **1000000** to load the update bitstream (Figure 21).



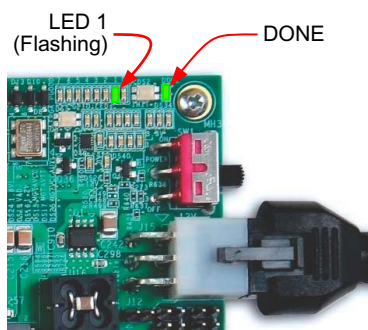
X17924-091416

Figure 21: Option 5: Reconfigure the IPROG

- The MicroBlaze processor executes step 2 (Figure 1) by writing a series of commands that cause a reconfiguration to occur. This set of writes to the ICAP interface (through the AXI\_HWICAP IP) is called IPROG, and results in a warm boot of the FPGA.

All of the previously configured contents of the FPGA are cleared except for the location specified in the WBSTAR address. Then, a reconfiguration occurs beginning at the flash address specified by the user and loaded into the WBSTAR by MicroBlaze. This results in the `update.bin` file being loaded into the FPGA.

Verify the FPGA is successfully configured with the `update.bin` reference design. Ensure the DONE LED is lit and the LED 1 is blinking as shown in Figure 22 (replacing the original `golden.bin` LED 0 pattern).



X17516-090616

Figure 22: The Reference Design Update Bitstream `update.bin` is Running



**TIP:** After `update.bin` is loaded into the FPGA, parallel NOR flash operations can no longer be performed. To run the demonstration again, execute the procedure starting at [Configure FPGA with `golden.bin`](#).

# Hardware System Details

The reference design includes a MicroBlaze processor core and peripheral IP cores to implement downloading the `update.bin` file. This section describes the system clocking topology, AXI EMC core settings, STARTUPE3 primitive usage, and the constraints for post-configuration parallel NOR flash memory access.

## Clock Topology

The reference design uses a 300 MHz differential clock input from a Silicon Labs Si5335A quad clock generator/buffer located on the VCU108 board. The clock distribution is shown in Figure 23.

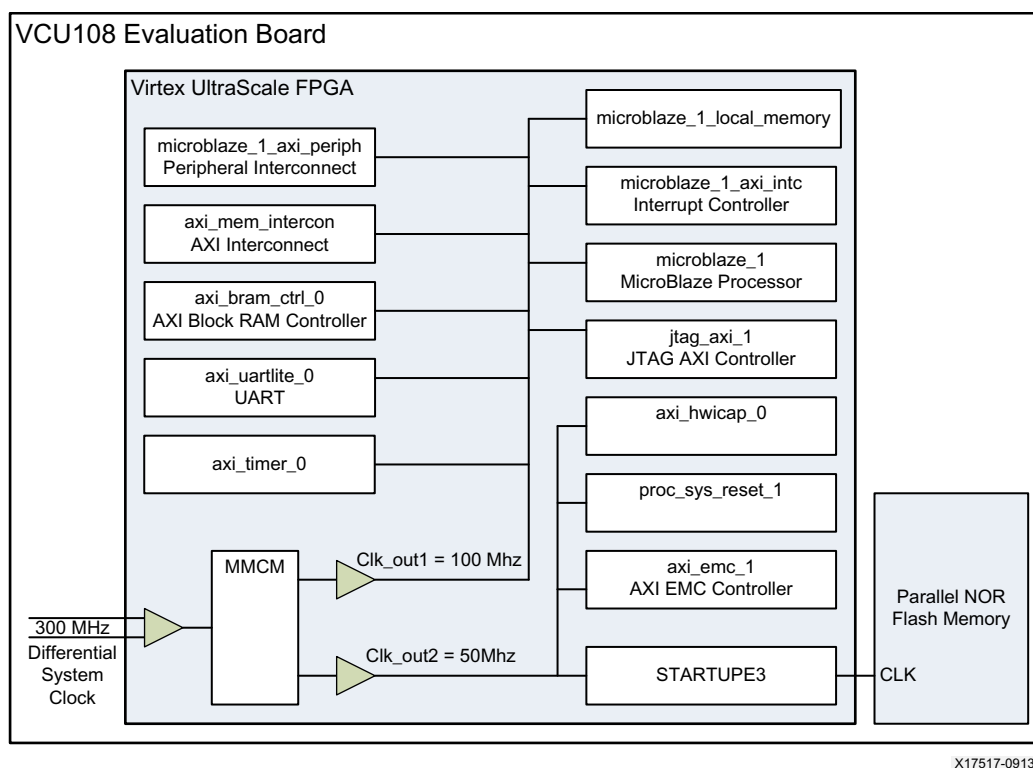


Figure 23: Reference Design Clocking Topology

The differential clock is supplied to a mixed-mode clock manager (MMCM) inside the clocking IP block (`clk_wiz_0`) that divides the input clock and provides a 100 MHz and 50 MHz clock to the rest of the system. The 50 MHz clock (`clk_out2`) is supplied to the AXI EMC, processor system reset, and `axi_hwicap` blocks, and the 100 MHz clock is supplied to the AXI peripheral interconnect and all the remaining peripherals in the system.

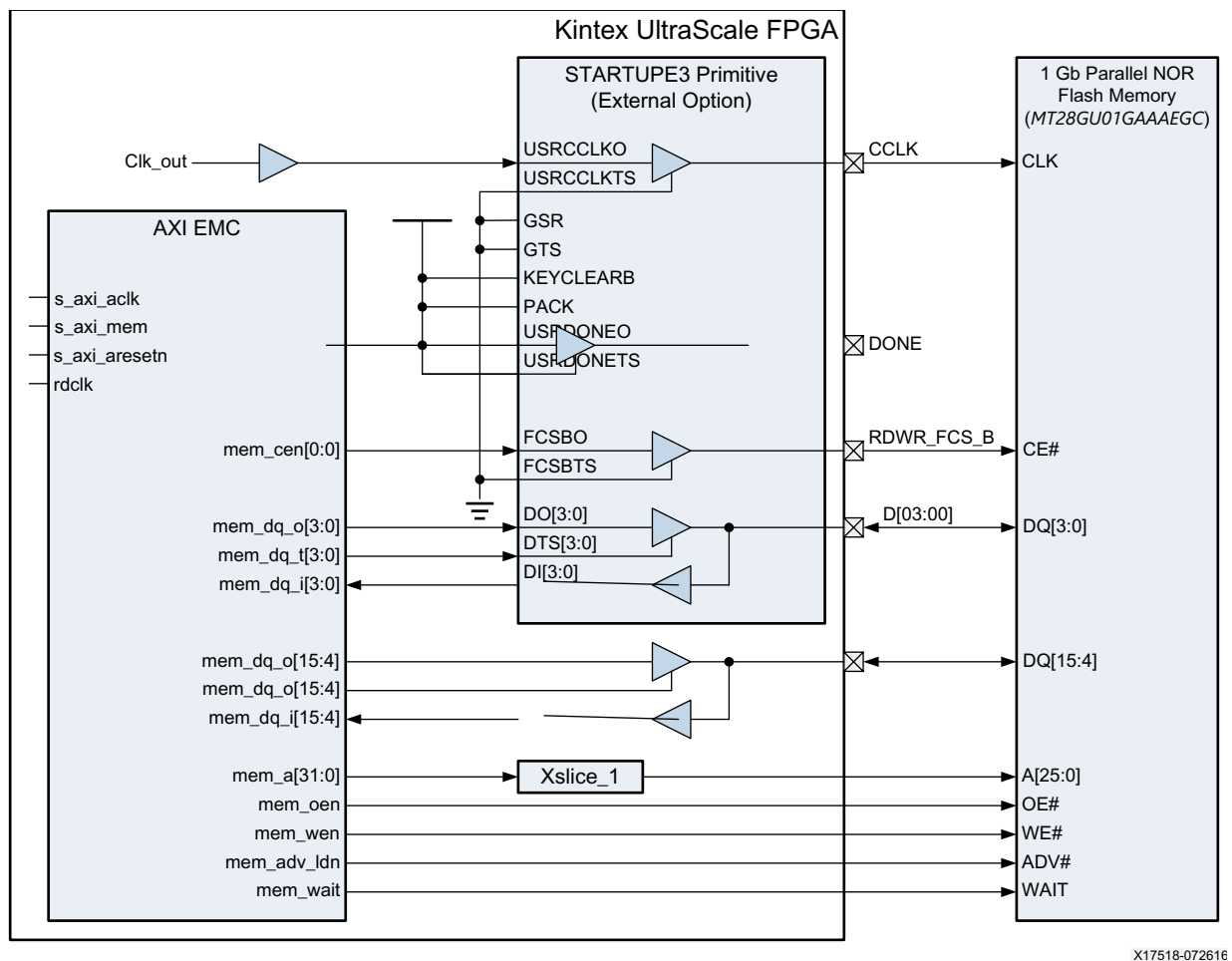
The 50 MHz clock used by the AXI EMC is also routed via the STARTUPE3 block to the parallel NOR flash memory. For the AXI EMC peripheral to function correctly, the same clock used by the flash must be provided to the AXI EMC. The AXI EMC (`axi_emc_1`) uses the 50 MHz clock to clock transactions on the AXI interface, as well as to create read and write transactions going to the parallel NOR flash memory.



**IMPORTANT:** This reference design is targeted for use with the Micron MT28GU01GAAA1EGC-0S1T flash memory provided on the VCU108 evaluation board. If the design is modified for use with other parallel NOR flash memories, the clock frequency supplied to the MicroBlaze system and to the parallel NOR flash memory via the STARTUPE3 block might require adjustment.

## AXI EMC Core and STARTUPE3 Primitive

The connections required for post-configuration access between the AXI EMC core, the STARTUPE3 primitive, the FPGA BPI configuration interface pins, and the external parallel NOR flash are shown in Figure 24.



X17518-072616

Figure 24: AXI EMC Core to STARTUPE3 Connectivity

The AXI EMC core address (mem\_a[31:0]) is truncated to 26 bits (Linear\_Flash\_address[25:0]) and routed to the dual-purpose I/O pins of the FPGA. The 16 data bits of the AXI EMC are divided into two vectors.

The first vector (mem\_dq[3:0]) is routed to the DI[3:0], DO[3:0], and DTS[3:0] ports of the STARTUPE3 primitive. These bits are routed through the STARTUPE3 block to dedicated FPGA pins interfacing with the parallel NOR flash. The second vector (upper 12 bits mem\_dq[15:4]) is

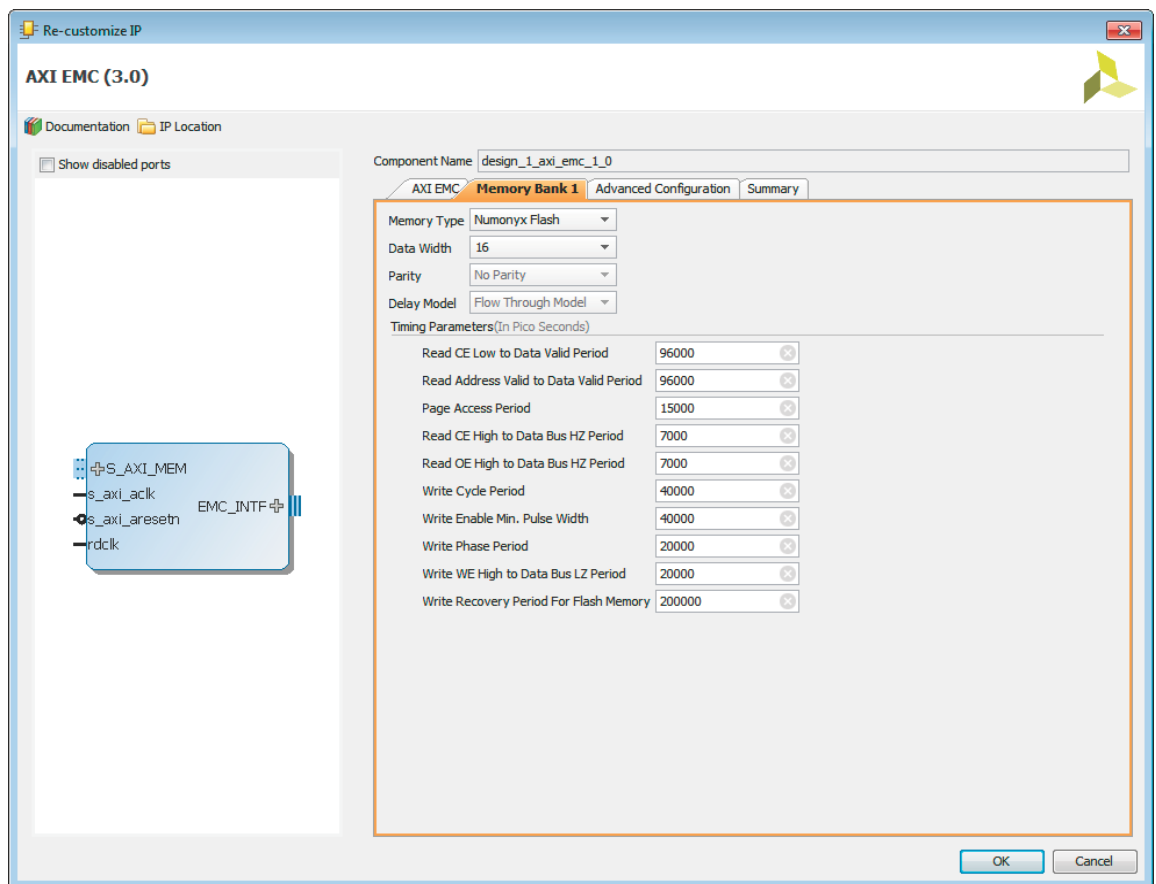
routed to 3-state IOBUFs in the top level wrapper design (`design_1_wrapper.vhd`) and then to dual purpose I/O pins of the FPGA.

The chip enable signal coming from the AXI EMC core is connected to the FCSBO port of the STARTUPE3 primitive, routed to the dedicated RDWR\_FCS\_B pin of the FPGA, and connected to the CE# pin of the parallel NOR flash. The `mem_adv_ldn`, `mem_oen`, `mem_wait`, and `mem_wen` signals coming from the AXI EMC do not travel through the STARTUPE3, and travel directly to dual-purpose FPGA pins and then to the parallel NOR flash.

The 50 MHz clock signal provided by the MMCM for the parallel NOR flash is routed to the USRCCLKO port of the STARTUPE3 primitive, travels to the dedicated CCLK pin of the FPGA, and then to the CLK pin of the parallel NOR Flash. The signals that travel through the STARTUPE3 primitive appear on the dedicated configuration pins of the FPGA, and do not require pin location constraints. The signals that travel from the AXI EMC to the dual use FPGA pins must be location constrained and are also handled slightly differently by the timing constraints.

## Customizing the AXI EMC IP Core (3.0)

The AXI EMC IP core (3.0) used in this example design is set up specifically for the XCVU095-2FFVA2104 FPGA, and the Micron (previously Numonyx) MT28GU01GAAA1EGC parallel NOR flash memory on the VCU108 board. The IP Integrator window shown in [Figure 25](#) is used for customization, and shows the timing parameter values used by the AXI EMC IP Core.



X17519-072616

Figure 25: Re-customize Window showing AXI EMC (3.0) Core Settings

Table 2 and [Timing Parameter Definitions](#) provide more timing parameter information.

**Table 2: AXI EMC IP Core Timing Parameters**

IP Re-Customization Settings			MT28GU01GAAA1EGC Parallel NOR Flash Memory Settings	
Parameter Description	User Parameter	Default Value (ps)	Parameter	User Value (ps) <sup>(1)</sup>
Read CE Low to Data Valid Period	C_TCEDV_PS_MEM_*	15,000	t <sub>ELQV</sub>	96,000
Read Address Valid to Data Valid Period	C_TAVDV_PS_MEM_*	15,000	t <sub>AVQV</sub>	96,000
Page Access Period	C_TPACC_PS_FLASH_*	25,000	t <sub>APA</sub>	15,000
Read CE High to Data Bus High-Z Period	C_THZCE_PS_MEM_*	7,000	t <sub>EHQZ</sub>	7,000
Read OE High to Data Bus High-Z Period	C_THZOE_PS_MEM_*	7,000	t <sub>GHQZ</sub>	7,000
Write Cycle Period	C_TWC_PS_MEM_*	15,000	t <sub>CW</sub> = (t <sub>WLWH</sub> + t <sub>ELWL</sub> + t <sub>WHEH</sub> )	40,000
Write Enable Minimum Pulse Width	C_TWP_PS_MEM_*	12,000	t <sub>WLWH</sub>	40,000
Write Phase Period	C_TWPH_PS_MEM_*	12,000	t <sub>WHWL</sub>	20,000
Write WE High to Data Bus Low Period	C_TLZWE_PS_MEM_*	0000	t <sub>WHGL</sub>	20,000
Write Recovery Period for Flash Memory	C_WR_REC_TIME_ME	27,000	t <sub>BHWH</sub>	200,000

**Notes:**

1. Parameter values in this column are from the Micron Parallel NOR Flash Memory MT28GU01GAAA1EGC Data Sheet [\[Ref 6\]](#).

### Timing Parameter Definitions

- **Read CE Low to Data Valid Period:** Read Chip Enable to Data Valid Period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to t<sub>ELQV</sub> for flash memory in the respective memory device data sheets.
- **Read Address Valid to Data Valid Period:** Read Address Valid to Data Valid Period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to t<sub>AA</sub> for asynchronous SRAM and t<sub>AVQV</sub> for flash memory in the respective memory device data sheets.
- **Page Access Period:** Page Access Period specification for the asynchronous flash memory in the selected bank. This configuration option is applicable only when memory type in the bank is Page Mode Flash Memory. This option is equivalent to t<sub>PACC</sub>/t<sub>APA</sub> in the Page Mode Flash device data sheet and must be assigned only when the memory type selected for the bank is Page Mode Flash.
- **Read CE High to Data Bus HZ Period:** Read CE High to data bus High impedance period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to t<sub>HZCE</sub> for asynchronous SRAM and t<sub>EHQZ</sub> for flash memory in the respective memory device data sheets.
- **Read OE High to Data Bus HZ Period:** Read OE High to data bus High-impedance period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to t<sub>HZOE</sub> for asynchronous SRAM and t<sub>GHQZ</sub> for flash memory in the respective memory device data sheets.



- **Write Cycle Period:** Write cycle time of the memory bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to  $t_{WC}$  for asynchronous SRAM and  $t_{CW}$  for flash memory in the respective memory device data sheets. The AXI EMC core uses this parameter to hold CEN logic-Low for each write at the memory interface.
- **Write Enable Minimum Pulse Width:** Write enable minimum pulse width duration of the memory bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to  $t_{WP}$  for Asynchronous SRAM and  $t_{PWE}$  for flash memory in the respective memory device data sheets. The AXI EMC core uses this parameter to hold WEN logic-Low for each write at the memory interface.
- **Write Phase Period:** Write cycle phase time period of the memory bank. This configuration option is applicable only when memory type in the bank is asynchronous. The AXI EMC core uses this parameter to determine the number of NOP cycles between two consecutive writes at the memory side. When this parameter is set to zero, CEN goes logic-High for one clock cycle between writes.
- **Write WE High to Data Bus LZ Period:** Write cycle Write Enable High to Data Bus Low impedance duration of memory bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to  $t_{LZWE}$  for asynchronous SRAM and  $t_{WHGL}$  for flash memory in the respective memory device data sheets. This option is also used to meet write recovery to read time requirements of the memory.
- **Write Recovery Period for Flash Memory:** Write recovery period for flash memories of the memory bank. This configuration option is applicable only when memory type in the bank is any flash memory or PSRAM. This option is equivalent to  $t_{WR}$  for flash memory in the respective memory device data sheets. The AXI EMC core uses this parameter to determine the number of NOP cycles at memory interface after each AXI WLAST. RECOMMENDED: You should provide the value for C\_AXI\_CLK\_PERIOD\_PS which is equivalent to AXI clock used. This value is used for internal calibration of counters when Asynchronous memories are used. By default this is set to 10,000 which is a value corresponding to 100 MHz AXI clock. If the AXI clock connected is not 100 MHz, the parameter has to be set using command **set\_property CONFIG.C\_AXI\_CLK\_PERIOD\_PS {time period in picoseconds of AXI clock} [get\_ips <component name of emc IP created>]** in the Tcl console.

## STARTUPE3 Primitive

The dedicated UltraScale FPGA BPI interface signals (RDWR\_FCS\_B\_0, CCLK\_0, D03\_0, D02\_0, D01\_DIN\_0, D00\_MOSI\_0) reside in Bank0. By default, these pins are not accessible for post-configuration access of the parallel NOR flash memory. To access these dedicated pins, the design instantiates the STARTUPE3 primitive. Table 3 shows the internal logic signal name associated with each dedicated configuration pin, the STARTUPE3 port it connects to, and the corresponding FPGA output pin location.

Table 3: STARTUPE3 BPI Configuration Interface Dedicated Pins

Logic Signal	STARTUPE3 Port	Dedicated Pin Name	FPGA (U1) Pin Location
Linear_Flash_ce_n(0)	FCSBO	RDWR_FCS_B_0	AJ11
Clk_out	USRCCLKO	CCLK_0	AF13
dq_o_e(3)	DO[3]	D03_0	AL11
dq_o_e(2)	DO[2]	D02_0	AM11
dq_o_e(1)	DO[1]	D01_DIN_0	AN11
dq_o_e(0)	DO[0]	D00_MOSI_0	AP11
dq_i(3)	DI[3]	D03_0	AL11
dq_i(2)	DI[2]	D02_0	AM11
dq_i(1)	DI[1]	D01_DIN_0	AN11
dq_i(0)	DI[0]	D00_MOSI_0	AP11

The STARTUPE3 primitive allows a user design to drive or 3-state the FPGA outputs on the dedicated pins by connecting the design to the appropriate STARTUPE3 ports. In addition, D[03:00] can be read as inputs. The dedicated FPGA pins are accessed by the user design through the .USRCCLKO, .USRCCLKTS, .DO, .DTS, .FCSBO, and .FCSBTS ports of the STARTUPE3 block. The data signals returning from the parallel NOR flash memory are provided to the user design through the .DI port. The UltraScale FPGA pins CCLK, D00\_MOSI\_0, D01\_DIN\_0, D02\_0, D03\_0, and RDWR\_FCS\_B\_0 are dedicated and not assigned in the constraints file because their physical locations cannot be altered.

For example, forwarding the Clk\_out externally to the flash is accomplished by connecting it to the USRCCLKO port of the STARTUPE3 instantiation. There is no explicit port declaration for Clk\_out at the top level of the design because the CCLK output of the FPGA can only be accessed via connections to the STARTUPE3 block. The STARTUPE3 port USRCCLKTS controls the enable of the USRCCLKO, and is tied to 0 so that the CCLK signal is always actively driven out to the parallel NOR flash memory. The signals going from the internal logic through the STARTUPE3 block to the dedicated external pins of the FPGA are not shown as top level ports of the design, and do not appear in the .xdc file.

Because the connections between the AXI EMC core and the STARTUPE3 primitive are made inside the top-level `design_1_wrapper.vhd` wrapper, the STARTUPE3 block does not appear in the IP Integrator block design. The instantiation of the STARTUPE3 is shown here:

```
STARTUPE3_inst : component STARTUPE3
    -----
    generic map
    (
        PROG_USR      => "FALSE", -- Activate program event security feature.
        SIM_CCLK_FREQ => 0.0      -- Set the Configuration Clock Frequency(ns) for simulation.
    )
    port map
    (
        CFGCLK => open,           -- 1-bit output: Configuration main clock output
        CFGMCLK => open,          -- 1-bit output: Configuration internal oscillator clock output
        DI => dq_i,               -- 4-bit output: Allow receiving on the D input pin
        EOS => startupe3_eos,      -- 1-bit output: Active-High output signal indicating the End Of Startup
        PREQ => open,             -- 1-bit output: PROGRAM request to fabric output
        DO => dq_o_e(3 downto 0), -- 4-bit input: Allows control of the D pin output
        DTS => dq_t_e(3 downto 0), -- 4-bit input: Allows tristate of the D pin
        FCSBO => Linear_Flash_ce_n(0), -- 1-bit input: Controls the FCS_B pin for flash access
        FCSBTS => '0',            -- 1-bit input: Tristate the FCS_B pin
        GSR => '0',               -- 1-bit input: Global Set/Reset input (GSR cannot be used for the port)
        GTS => '0',               -- 1-bit input: Global 3-state input (GTS cannot be used for the port name)
        KEYCLEARB => '1',         -- 1-bit input: Clear AES Decrypter Key input from Battery-Backed RAM
        PACK => PACK_int,         -- 1-bit input: PROGRAM acknowledge input
        USRCCLKO => Clk_out,       -- 1-bit input: User CCLK input
        USRCCLKTS => '0',         -- 1-bit input: User CCLK 3-state enable input
        USRDONEO => '1',          -- 1-bit input: User DONE pin output control
        USRDONETS => '1'          -- 1-bit input: User DONE 3-state enable output
    )
end STARTUPE3_inst;
```



**TIP:** Designs that require the use of the STARTUPE3 primitive for multiple cores should use the STARTUPE3 external selection as demonstrated in this reference design. The external option provides the most flexibility. If a user design only requires STARTUPE3 port access with the AXI EMC core then the setting **Use STARTUP primitive internal to IP** could be used.

## STARTUPE3 Timing Constraints

While the output pins of the STARTUPE3 block do not appear in the top-level `design_1_wrapper.vhd` wrapper, it is still necessary to apply timing constraints to signals going into, or coming, out of the STARTUPE3 block.

The data and control signals that go between the AXI EMC core to the parallel NOR flash memory in parallel to the signals traversing the STARTUPE3 block are aligned using the FIFO logic in the AXI EMC core. The required timing constraints for these signals are shown in [Figure 26](#). The timing parameters entered in [Figure 25](#) are extracted from the parallel NOR flash memory data sheet and set the alignment of the control signals to the data and address of the parallel NOR flash memory.

Figure 26 also shows the signals traveling through the STARTUPE3 block. The paths that are covered by timing constraints as well as the fixed delays that are not considered by the tools are shown. The paths that are constrained are covered by set\_max\_delay and set\_min\_delay constraints shown here.

```
#### Following are the timing parameters of MT28GU01GAAA1EGC-0SIT flash, delays for the STARTUPE3 element, and
#### delays related to the VCU108 board traces
#### Max Tco -- Clock to input delay for flash
set tco_max 5.5
set tco_min 5.5
#### Trace delays for VCU108 board
set tdata_trace_delay_max 0.25
set tdata_trace_delay_min 0.25
set tclk_trace_delay_max 0.2
set tclk_trace_delay_min 0.2
set startup_delay_max 6.700
set startup_delay_min 1.000
set board_del_max 1.000
set board_del_min 0.500

#### Input delay constraints (max/min) for Linear_flash_wait[0]
set_input_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -max [expr $tco_max + $tdata_trace_delay_max
+ $tclk_trace_delay_max + $board_del_max] [get_ports Linear_Flash_wait[0]]
set_input_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -min [expr $tco_min + $tdata_trace_delay_min
+ $tclk_trace_delay_min + $board_del_min] [get_ports Linear_Flash_wait[0]]
```

The logic to STARTUPE3 path constraints below are referenced in Figure 26.

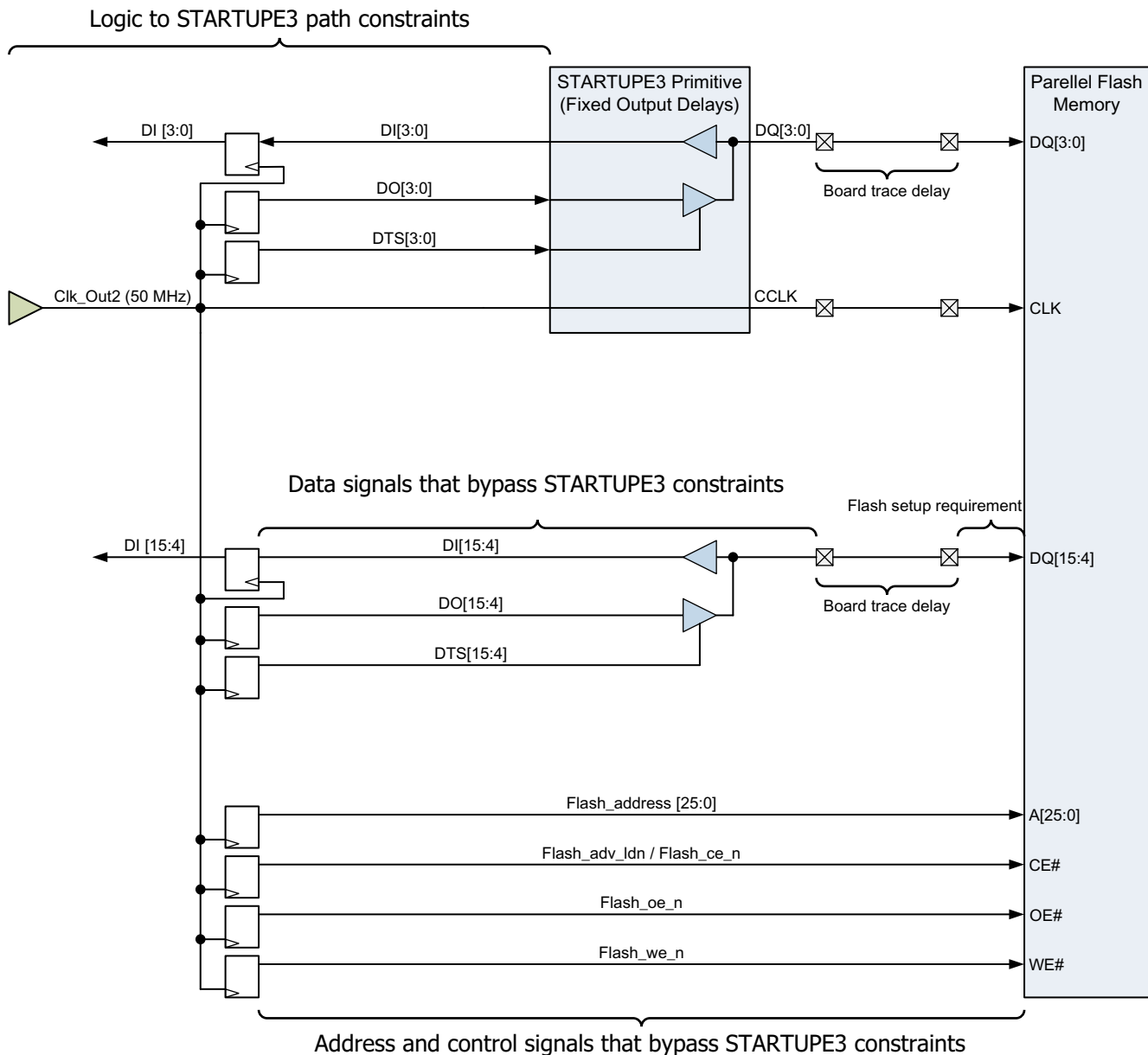
```
#### These constraints cover the paths going between logic and the STARTUPE3 block
set_max_delay -datapath_only -from [get_pins -hier {*STARTUP*_inst/DI[*]}] -to [get_cells -hier
{*Mem_DQ_I_v_reg[*]}] 1.500
set_max_delay -datapath_only -from [get_cells -hier {*mem_dq_o_reg_reg[*]}] -to [get_pins -hier
*STARTUP*_inst/DO[*]] 1.500
set_max_delay -datapath_only -from [get_cells -hier {*mem_dq_t_reg_reg[*]}] -to [get_pins -hier
*STARTUP*_inst/DTS[*]] 1.500
```

The data signals that bypass STARTUPE3 constraints below are referenced in Figure 26.

```
#### Input and output delay constraints max/min for mem_dq_io[*] signals that bypass STARTUPE3 block. Includes
#### startup_delay_max/min values to match the paths of the bits traveling through STARTUPE3
set_input_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -max [expr $tco_max + $tdata_trace_delay_max
+ $tclk_trace_delay_max + $startup_delay_max + $board_del_max] [get_ports mem_dq_io[*]]
set_input_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -min [expr $tco_min + $tdata_trace_delay_min
+ $tclk_trace_delay_min + $startup_delay_min + $board_del_min] [get_ports mem_dq_io[*]]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -max [expr $tdata_trace_delay_max -
$tclk_trace_delay_min + $startup_delay_max + $board_del_max] [get_ports mem_dq_io[*]]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -min [expr $tdata_trace_delay_min -
$tclk_trace_delay_max + $startup_delay_min + $board_del_min] [get_ports mem_dq_io[*]]
```

The address and control signals that bypass STARTUPE3 constraints below are referenced in [Figure 26](#).

```
#### Output delay constraints (max/min) for Linear_Flash_address[*], Linear_Flash_oe_n, Linear_Flash_we_n, and
#### Linear_Flash_adv_ldn pins
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -max [expr $tdata_trace_delay_max -
$tlck_trace_delay_min + $board_del_max] [get_ports Linear_Flash_address[*]]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -min [expr $tdata_trace_delay_min -
$tlck_trace_delay_max + $board_del_min] [get_ports Linear_Flash_address[*]]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -max [expr $tdata_trace_delay_max -
$tlck_trace_delay_min + $board_del_max] [get_ports Linear_Flash_oe_n[0]]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -min [expr $tdata_trace_delay_min -
$tlck_trace_delay_max + $board_del_min] [get_ports Linear_Flash_oe_n[0]]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -max [expr $tdata_trace_delay_max -
$tlck_trace_delay_min + $board_del_max] [get_ports Linear_Flash_we_n]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -min [expr $tdata_trace_delay_min -
$tlck_trace_delay_max + $board_del_min] [get_ports Linear_Flash_we_n]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -max [expr $tdata_trace_delay_max -
$tlck_trace_delay_min + $board_del_max] [get_ports Linear_Flash_adv_ldn]
set_output_delay -clock [get_clocks clk_out2_design_1_clk_wiz_0_0] -min [expr $tdata_trace_delay_min -
$tlck_trace_delay_max + $board_del_min] [get_ports Linear_Flash_adv_ldn]
```



X17520-091916

Figure 26: **STARTUPE3 Timing Delays**

To reduce the delay on the clock traveling from the design logic to the USRCCLKO pin, use the `set_max_delay` constraint. The data and 3-state signal paths to and from the STARTUPE3 primitive are constrained using `set_output_delay` and `set_input_delay` constraints with respect to the generated clock on the USRCCLKO pin. To show how the constraints are created, the delays of the STARTUPE3 block and the setup requirements of the parallel NOR flash memory are obtained from the *Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 5] and the *Micron Parallel NOR Flash Memory MT28GU01GAAA1EGC Data Sheet* [Ref 6]. Sample parameters are provided in Table 4 and Table 5 to be used for the constraints.

Some of the constraints above cover signals that travel from the AXI EMC core to the STARTUPE3 block. The remainder of these constraints cover the signals that travel directly from pins of the FPGA to the AXI EMC core without going through the STARTUPE3.

The fixed delays for the CCLK and the data signals traveling through the STARTUPE3 are shown in [Table 4](#).

**Table 4: Virtex UltraScale Timing Parameters**

Symbol	Description	V <sub>CCINT</sub> Operating Voltage = 0.95V	Units
		-2 Speed Grade	
STARTUPE3 Ports			
T <sub>USRCCLKO</sub>	STARTUPE3 USRCCLKO input port to CCLK pin output delay	1.00/6.70	ns, Min/Max
T <sub>DO</sub>	DO[3:0] ports to D03-D00 pins output delay	1.00/7.70	ns, Min/Max
T <sub>DTS</sub>	DTS[3:0] ports to D03-D00 pins 3-state delays	1.00/8.30	ns, Min/Max
T <sub>FCSBO</sub>	FCSBO port to FCS_B pin output delay	1.00/8.00	ns, Min/Max
T <sub>FCSBTS</sub>	FCSBTS port to FCS_B pin 3-state delay	1.00/8.00	ns, Min/Max
T <sub>DI</sub>	D03-D00 pins to DI[3:0] ports input delay	0.5/3.1	ns, Min/Max

**Table 5: Micron MT28GU01GAAA1EGC Parallel NOR Flash Memory Timing Parameters**

Symbol	Description	Units
T <sub>ELQV</sub> (max)	CE# LOW to output valid	96 ns
T <sub>AVQV</sub> (max)	Address to output valid	96 ns
T <sub>APA</sub> (max)	Page address access (non-MUX only)	15 ns
T <sub>ACC</sub> (min)	Read access time from address latching clock	96 ns
T <sub>GHQZ</sub> (max)	OE# HIGH to output in High-Z	7 ns
T <sub>WLWH</sub> (min)	WE# write pulse width LOW	40 ns
T <sub>ELWL</sub> (min)	CE#setup to WE# LOW	0 ns
T <sub>WHEH</sub> (min)	CE# hold from WE# HIGH	0 ns
T <sub>WHGL</sub> (min)	WE# HIGH to OE# LOW	0 ns
T <sub>BHWH</sub> (min)	WP# setup to WE# HIGH	200 ns
TEHQZ (max)	CE# HIGH to output in High-Z	7ns
TGHQZ (max)	OE# HIGH to output in High-Z	7 ns

Example constraints for the VCU108 board are shown in [Figure 26](#). The timing delays for the board, and the Micron MT28GU01GAAA1EGC flash memory are obtained from measurement and the flash memory data sheet.

These constraints account for the path delay to and from the AXI EMC IP core logic, the setup and hold requirements of the parallel NOR flash memory, the FPGA I/O port delays, the STARTUPE3 primitive delays, and the board trace delays. The timing constraints for the reference design are included in the `vcu108.xdc` constraints file. If this example design is adapted to a board other than the VCU108, values for the trace delay, clock delay, and output delay of the flash must be modified to ensure correct functionality.

## BPI Configuration Mode Constraints

The correct properties must be used during `golden.bin` file generation to ensure a successful master BPI x16 configuration. The properties listed in [Figure 25](#) are included in the

reference .xdc constraint file supporting the VCU108 evaluation board 1.8V parallel NOR flash. For additional details on the options see *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1] and *UltraScale FPGA BPI Configuration and Flash Programming Application Note* (XAPP1220) [Ref 7].

```
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
set_property CONFIG_MODE BPI16 [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGFALLBACK Enable [current_design]
set_property BITSTREAM.CONFIG.TIMER_CFG 0x00050000 [current_design]
set_property BITSTREAM.CONFIG.BPI_SYNC_MODE Type1 [current_design]
set_property BITSTREAM.CONFIG.UNUSEDPIN Pulldown [current_design]
set_property BITSTREAM.CONFIG.NEXT_CONFIG_REBOOT Enable [current_design]
set_property BITSTREAM.CONFIG.NEXT_CONFIG_ADDR 32'h01000000 [current_design]
```

---

## Software System Details

Software running on the MicroBlaze processor drives the AXI EMC core to read and write to the parallel NOR flash memory using the hardware connections to the STARTUPE3 primitive. The `vcu108_emc_rw.c` file included with this reference design provides example low-level software read and write operations. These read and write functions are used to implement the Micron flash memory operations described by the command descriptions in the *Micron Parallel NOR Flash Memory MT28GU01GAAA1EGC Data Sheet* [Ref 6]. This memory is located on the VCU108 evaluation board at location U133 (see Figure 3).

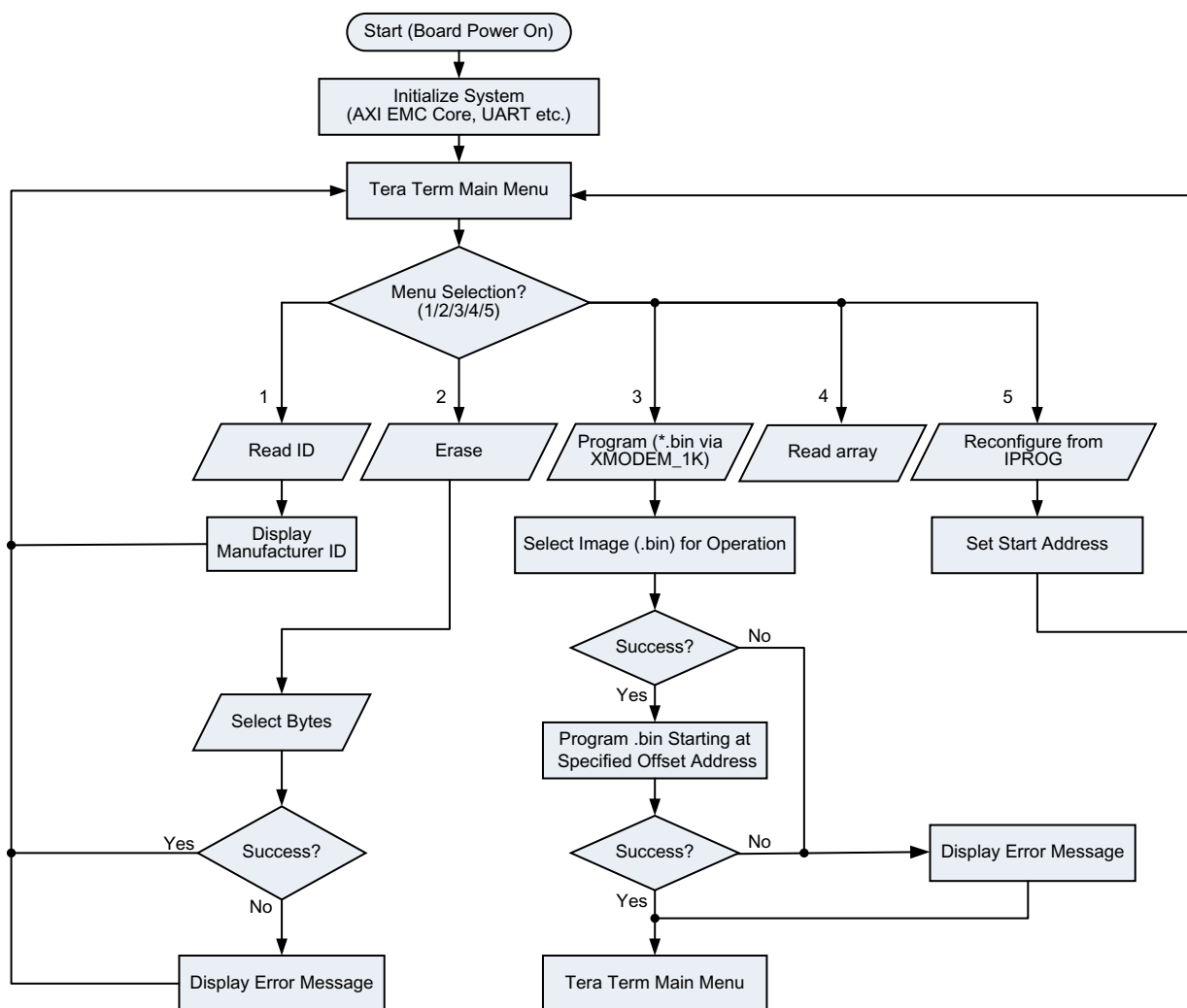
The functions in `vcu108_emc_rw.c` are provided as a starting point for creating memory operations that might be necessary with flash memories from other vendors. Most parallel NOR flash memories share a similar set of commands that allow control logic to read, write, or erase the flash array and access registers that control the flash memory behavior. This reference design uses the command set for the Micron MT28GU01GAAA1EGC parallel NOR flash memory used on the VCU108 evaluation board and supports the set of instructions listed in [Ref 6].

### Flash Command Set

There are a number of commands that provide access to the flash array or perform other operations on the parallel NOR flash memory. All parallel NOR flash transactions depend upon commands supported by the flash memory connected to the AXI EMC core. Refer to the respective parallel NOR flash memory data sheet for the supported commands. The Micron MT28GU01GAAA1EGC parallel NOR flash memory commands are described in [Ref 6]. This



reference design provides user access to a subset of commands by way of the menu shown in Figure 15. See Figure 27 for the flash command set flow diagram.



X17521-072616

Figure 27: Flash Command Set Flow Diagram

Each of the supported operations shown in Figure 27 has a specific command sequence. The command sequences are provided below for each of the major operations.

The XILFLASH library implements the functionality for flash memory devices that conform to the common flash interface (CFI) standard. CFI allows a single flash library to be used for an entire family of parts. This is not a library for a specific device, but for a set of command read, write, and erase algorithms. CFI permits the determination of which algorithm to utilize at runtime.

The XILFLASH Library API description for basic parallel NOR flash access operations is provided here:

1. **Flash Initialize:** The function call `XFlash_Initialize()` should be called by the application before any other function is used by the application from the library. The initialization

function checks for the device family and initializes the XFlash instance with the family-specific data (for example, Micron or Spansion, etc.). After the target flash CFI query is read successfully, the vector table maintained for the further operations like erase, read, and write operations. The vector table that contains the function pointers to family-specific APIs is setup and the family specific initialization routine is called that performs these tasks:

- a. *Issue the CFI query command:* Write base address with 0x0098 (CFI query command).
  - b. *Identify the Flash family and layout:* From CFI data at 0x10, 0x11, 0x12 (QRY character hex value).
  - c. *Populate the vector table with the geometry readback:* Provides specific options from the flash architecture. See *Micron Parallel NOR Flash Memory MT28GU01GAAA1EGC Data Sheet*.
  - d. *Set up the Vector Table with the CFI data:* Readback the base address with data from the vector table).
2. **Flash Erase:** The erase operations are provided to erase a block of specified address range in the flash memory. The erase call is blocking in that control is returned back to user only after the erase operation is completed successfully or an error is reported. The number of bytes to erase can be any number as long as it is within the bounds of the devices. Flash erase performs these tasks:
    - a. Lock the block.
    - b. Write 0x20, block address.
    - c. Unlock the block address.
    - d. For confirmation erase with 0xD0, block address.
    - e. Read Status register = 0x7 then the block is erased.
  3. **Flash Write:** The flash write operation can be used to write a minimum of zero bytes and a maximum including the entire flash memory. If the offset address specified to write is outside of the flash memory, or if the number of bytes specified from the offset address exceed flash memory boundaries, an error is reported back to the user. The write is blocking in that the control is returned back to user only after the write operation is completed successfully or an error is reported. This operation programs the flash memory with the data specified in the user buffer. The source and destination addresses must be aligned to the width of the flash data bus. If the processor supports unaligned access, then the source address does not need to be aligned to the flash width. However, this library is generic, and because some processors (such as MicroBlaze processors) do not support unaligned access, this API requires that the source address be aligned. Flash write performs these tasks:
    - a. Lock the block = 0x20.
    - b. If the device is initialized the state will be IsReady = 1.
    - c. *Flash Reset:* This function resets the flash device and places it in read mode.
    - d. Unlock the block = 0x60h.
    - e. Send buffer data to the flash device.
    - f. If returned, get the success then the write operation gets fixed.

4. **Flash Read:** The flash read operation is used to read a minimum of zero bytes and maximum of entire flash memory. If the offset address specified to read is out of flash boundary an error is reported back to the user. The read function reads memory locations beyond the flash memory boundary. Care should be taken to make sure that the number of bytes + offset address is within the flash memory address boundaries. The write is blocking in that the control is returned back to user only after the read operation is completed successfully or an error is reported.
  - a. Lock the block = 0x20.
  - b. If the device is initialized the state will be IsReady = 1.
  - c. *Flash Reset:* this function resets the flash device and places it in read mode.
  - d. Unlock the block = 0x60h.
  - e. Read the block of data from the flash memory from offset address.

---

## Checklist and Debug Tips

1. When [Customizing the AXI EMC IP Core \(3.0\)](#), verify **Use STARTUP Primitive External to the IP** selected for this demonstration (see [Figure 25](#)).
2. Ensure [Set Up VCU108 Evaluation Board](#) is completed before running the reference design demonstration. This ensures that:
  - The board power is connected properly and power switch SW1 is in the ON position.
  - Master BPI configuration mode is properly set (Mode pins are set to M[2:0] = 010). See DIP switch SW16 settings shown in [Figure 3](#).
  - Both UART and JTAG cables are connected from the VCU108 evaluation board to the host computer See [Figure 3](#).
3. If a Xilinx Platform Cable USB II is used instead of the on-board JTAG cable supplied with the VCU108 evaluation board, the reference design scripts need to be modified for the different cable target. Refer to *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [\[Ref 9\]](#).
4. If you do not see the Reference Design Main Menu displayed in Tera Term ([Figure 15](#)) or if UART communication is not working between the host computer and the VCU108 board (J4), then verify the Tera Term and Computer COM port setup is correct:
  - Confirm proper COM port driver is selected in **Windows Device Manager**. Select the *standard* version, not the enhanced version ([Figure 28](#)). The COM number might be different on different computers. If the VCU108 System Controller Main Menu is displayed in Tera Term this indicates that the enhanced COM port was incorrectly selected instead of the standard COM port target.

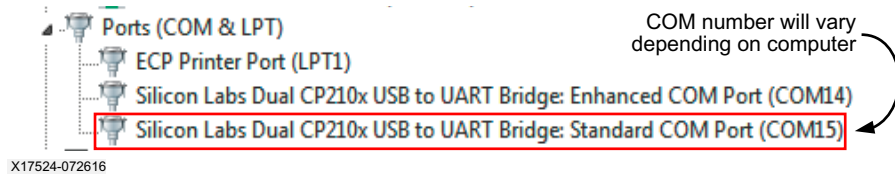
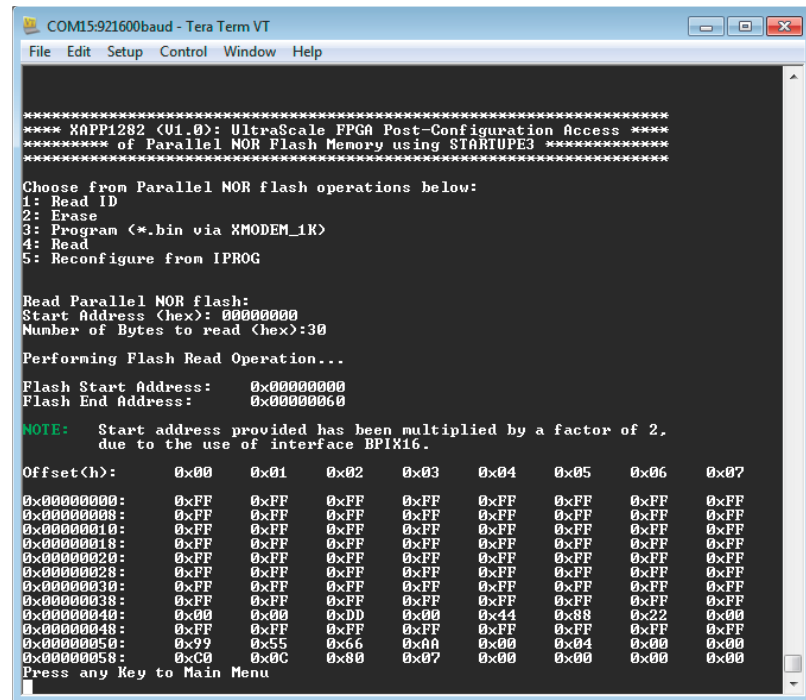


Figure 28: Standard COM Port Driver

- Confirm Tera Term is installed correctly (version 4.90 was used for reference design testing).
  - Confirm COM port settings in the Tera Term **Port Settings** tab match the values shown in [Figure 14](#).
5. If the FPGA does not seem to be configuring:
- Confirm a valid bitstream image is loaded into parallel NOR flash memory
  - Start FPGA programming by momentarily pressing the FPGA PROG pushbutton SW4 or cycle power using SW1 (See [Figure 3](#) for locations)
6. If the batch files do not run correctly:
- Ensure Vivado Design Suite 2016.1 is installed and that the path is set up correctly.
7. The DONE LED does not light and the Master BPI configuration mode is properly set on DIP switch SW16:
- Verify that the FPGA `golden.bin` can be programmed directly using the JTAG interface (J106)
  - Perform a readback to check the contents of the parallel NOR flash memory programmed at different locations ([Figure 29](#)).



```

***** XAPP1282 (U1_0): UltraScale FPGA Post-Configuration Access *****
***** of Parallel NOR Flash Memory using STARTUPE3 *****
*****

Choose from Parallel NOR flash operations below:
1: Read ID
2: Erase
3: Program (*.bin via XMODEM_1K)
4: Read
5: Reconfigure from IPROG

Read Parallel NOR flash:
Start Address (hex): 00000000
Number of Bytes to read (hex):30

Performing Flash Read Operation...

Flash Start Address: 0x00000000
Flash End Address: 0x00000060

NOTE: Start address provided has been multiplied by a factor of 2,
      due to the use of interface BPIx16.

Offset(h): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
0x00000000: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000008: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000010: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000018: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000020: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000028: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000030: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000038: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000040: 0x00 0x00 0x00 0x00 0x00 0x00 0x22 0x00
0x00000048: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00000050: 0x99 0x55 0x66 0xAA 0x00 0x04 0x00 0x00
0x00000058: 0xC0 0x0C 0x80 0x07 0x00 0x00 0x00 0x00

Press any Key to Main Menu

```

X17523-090616

Figure 29: Option 4: Read Parallel NOR Flash Output

- For user designs, ensure the write\_bitstream constraints are implemented correctly for BPI configuration mode as described in [Hardware System Details](#).
  - In the rare instance that a download error occurs during the update.bin download and LED0 is still blinking, erase and try to program update.bin again. If XMODEM hangs, make sure the device has been erased.
  - The DONE LED is lit, but the wrong pattern is loaded. Ensure load\_golden.bat was used. The pattern for the initial image (golden.bin) is LED0 blinking, and the pattern for the update image (update.bin) is LED1 blinking.
8. This reference design uses operations that are targeted for the Micron MT28GU01GAAA1EGC parallel NOR flash memory. If your design is based on this reference design but targets a different flash memory, the design must be modified to support the different flash memory. Refer to the target flash vendor data sheet.
  9. The JTAG-to-AXI IP enables the Vivado hardware manager to send low-level JTAG commands via the JTAG cable. To perform an IDCODE check on the parallel NOR flash memory use the following Tcl command sequence:
    - a. In the Tcl console change the path to the c:/xapp1282/source directory and type the commands below to display the parallel NOR flash ID:
 

```
source jtag_to_axi.tcl
```
  10. As an option, the Platform Cable USB II can be used as a JTAG interface between the computer and the VCU108 evaluation board, but this will require changes to the Vivado scripts provided to indicate the cable targeted.

## Conclusion

The ease of accessing the parallel NOR flash memory from the UltraScale FPGA post-configuration has been demonstrated. Accessing parallel NOR flash memory for multiple bitstreams or remote updates has the potential to not only reduce board space but also decrease the memory storage solution cost.

## Reference Design

You can download the [Reference Design Files](#) for this application note from the Xilinx website. [Table 6](#) shows the reference design matrix.

**Table 6: Reference Design Matrix**

Parameter	Description
<b>General</b>	
Developer names	Steven Howell, Shashikant Jadhav, and Stephanie Tapp
Target device	Virtex UltraScale FPGA (XCVU095-2FFVA2104)
Source code provided	Yes
Source code format	Verilog
Design uses code and IP cores from existing Xilinx application notes and reference designs or third party	Vivado Design Suite
<b>Simulation</b>	
Functional simulation performed	No
Timing simulation performed	No
Test bench used for functional and timing simulations	No
Test bench format	N/A
Simulator software/version used	N/A
SPICE/IBIS simulations	N/A
<b>Implementation</b>	
Synthesis software tools/versions used	Vivado synthesis
Implementation software tools/versions used	Vivado Design Suite and SDK 2016.1
Static timing analysis performed	Yes
<b>Hardware Verification</b>	
Hardware verified	Yes
Hardware platform used for verification	VCU108 evaluation board

---

## References

1. *UltraScale Architecture Configuration User Guide* ([UG570](#))
2. *Vivado Design Suite Quick Reference Guide* ([UG975](#))
3. *Tera Term Terminal Emulator Installation Guide* ([UG1036](#))
4. *Silicon Labs CP210x USB-to-UART Installation Guide* ([UG1033](#))
5. *Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* ([DS893](#))
6. *Micron Parallel NOR Flash Memory MT28GU01GAAA1EGC Data Sheet* ([www.micron.com](http://www.micron.com))
7. *UltraScale FPGA BPI Configuration and Flash Programming Application Note* ([XAPP1220](#))
8. *VCU108 Evaluation Kit web page* ([www.xilinx.com/vcu108](http://www.xilinx.com/vcu108))
9. *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#))
10. *Vivado Design Suite Tcl Command Reference Guide* ([UG835](#))
11. *VCU108 Evaluation Kit Quick Start Guide* ([XTP400](#))
12. *VCU108 Board User Guide* ([UG1066](#))
13. *VCU108 Software Install and Board Setup Tutorial* ([XTP368](#))
14. *AXI EMC Product Guide* ([PG100](#))

---

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/27/2016	1.0	Initial Xilinx release.
02/08/2022	1.1	In <a href="#">STARTUPE3 Timing Constraints</a> , updated startup_delay_max and startup_delay_min values.

## Please Read: Important Legal Notices

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

### **AUTOMOTIVE APPLICATIONS DISCLAIMER**

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING 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 2016–2022 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.