



XAPP1280 (v1.1) June 02, 2016

UltraScale FPGA Post-Configuration Access of SPI Flash Memory using STARTUPE3

Authors: Steven Howell, Shashikant Jadhav, and Stephanie Tapp

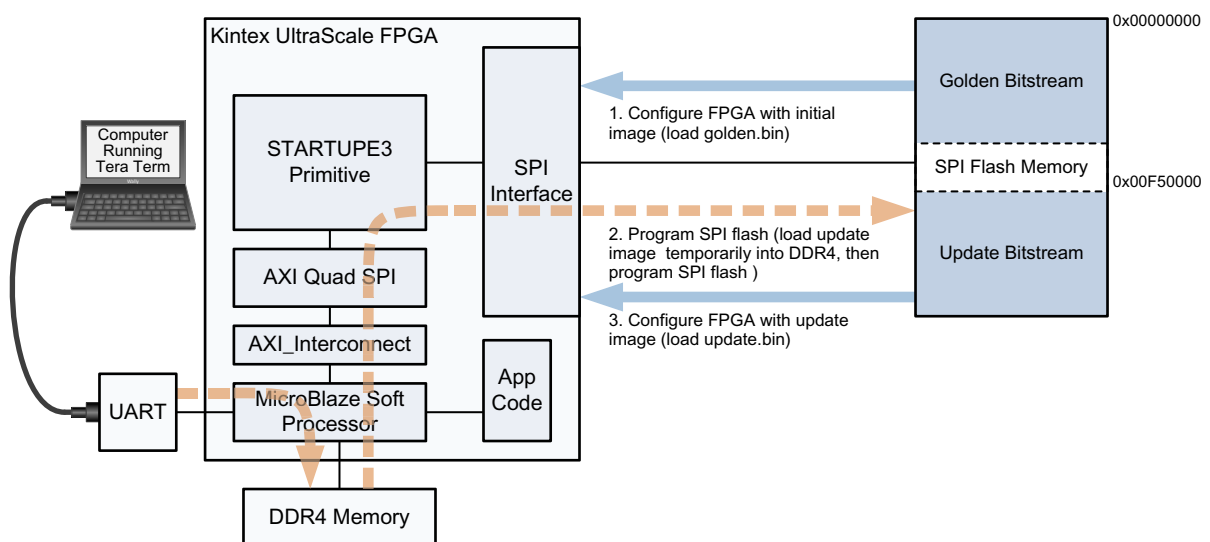
Summary

Serial NOR flash memory (referred to as SPI 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 and reference design demonstrates post-configuration access between a Kintex® UltraScale FPGA and SPI flash memory provided by a KCU105 evaluation board. Vivado® Design Suite 2016.1 or later is used as 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® soft processor core to interface to the AXI Quad SPI core and the STARTUPE3 primitive to implement post-configuration read and write access through a dedicated SPI interface to the on-board SPI flash memory. [Figure 1](#) shows operation of the post-configuration reference design.



X16222-030116

Figure 1: Reference Design Post-Configuration Access Flow



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 will disrupt SPI flash memory access.*

When the SPI configuration mode is used to configure the FPGA, the initial bitstream image loads from the SPI flash memory. After the FPGA is initially configured, the SPI configuration interface typically remains unused. However, the unused space in the SPI flash memory can be used to store additional configuration images or application data, eliminating the cost of adding extra memory and using more 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 SPI flash memory. The golden bitstream image includes the STARTUPE3 primitive, interface logic, IP cores, and constraints to enable reading and writing to the unused storage in the SPI flash memory.
- Step 2 runs application code on the MicroBlaze processor to download a new update bitstream from the computer by writing it into DDR4 memory temporarily and then moving it into the SPI flash memory. This step is controlled through the UART/Tera Term interface.
- Step 3 reconfigures the FPGA using the update bitstream image (`update.bin`) stored in the SPI flash memory and replaces 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 KCU105 Board](#)
 - [Download Reference Design Files](#)
 - [Generate Reference Design Project](#)
 - [Generate Programming Files](#)
 - [Configure FPGA with `golden.bin`](#)
 - [Program SPI Flash Memory with `update.bin`](#)
 - [Configure FPGA with `update.bin`](#)
- [Hardware System Details](#): Describes the hardware clock topology, AXI quad SPI 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 SPI flash memory post-configuration.

Recommended Design Experience

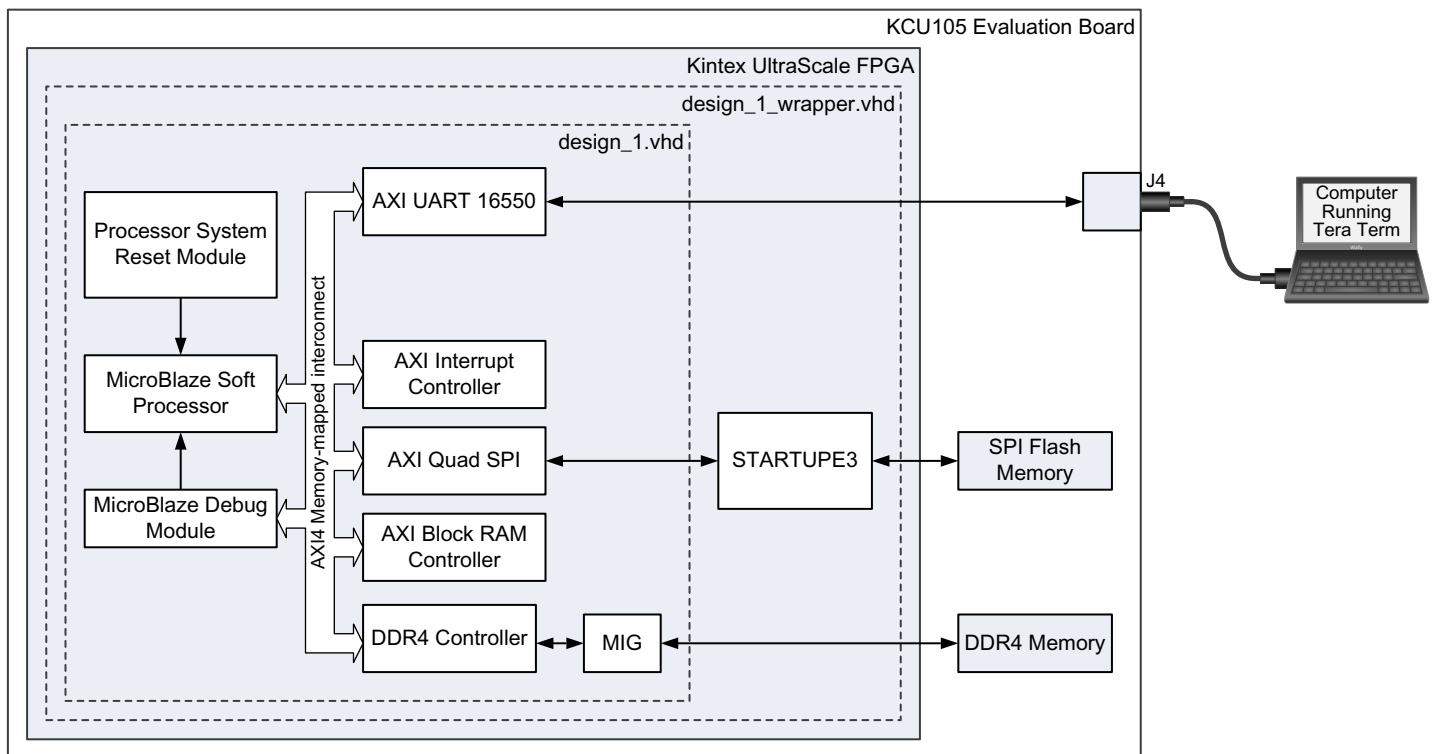
A general knowledge of:

- The UltraScale FPGA SPI 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 soft processor core drives the AXI Quad SPI core to read and write to SPI flash memory through the STARTUPE3 primitive. The software presents a command menu to the host computer running Tera Term through the AXI UART 16550 core. The menu provides commands to allow erasing, programming, and verifying SPI flash memory content. [Software System Details](#) describes the flash command set and software flow.

Vivado Design Suite IP Integrator (IPI) creates 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](#).



X16223-051616

Figure 2: Post-Configuration Reference Design System Block Diagram

Reference Design Files

Project script files:

- `run_kcu105(.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 SPI 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)
- `kcu105.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)
- `flash_qspi_rw.c` (SDK source code)
- `lscript.ld` (linker script)
- `kcu105_qspi_main.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 Soft Processor (microblaze_0)	9.5	100 MHz		N/A	N/A
DDR4 SDRAM MIG (ddr4_0)	1.1	300 MHz	0x8000_0000	1 GB	0xBFFF_FFFF
AXI Quad SPI (axi_quad_spi_0)	3.2	100 MHz	0x44A0_0000	64 KB	0x44A0_FFFF
AXI Block RAM Controller (axi_bram_ctrl_0)	8.3	100 MHz	0xC000_0000	1 MB	0xC00F_FFFF
AXI Interrupt Controller (microblaze_0_axi_intc)	4.1	100 MHz	0x4120_0000	64 KB	0x4120_FFFF
AXI UART 16550 (axi_uart16550_0)	2.0	100 MHz	0x44A1_0000	64 KB	0x44A1_FFFF
AXI Interconnect (axi_mem_intercon)	2.1	100 MHz		N/A	N/A
Processor System Reset Module (proc_sys_reset)	5.0	100 MHz		N/A	N/A

SPI Flash Memory Map

The reference design stores two images in the SPI flash memory (shown in [Figure 1](#)):

- Golden Bitstream starting at 0x00000000
- Update Bitstream starting at 0x00F50000

Running the Reference Design

This section describes how to:

- [Set Up Host Computer](#)
- [Set Up KCU105 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 SPI Flash Memory with update.bin](#) (Corresponds to step 2 in [Figure 1](#))
- [Configure FPGA with update.bin](#) (Corresponds to step 3 in [Figure 1](#).)

Required Hardware and Software

- KCU105 evaluation board which includes:
 - Kintex UltraScale XCKU040-2FFVA1156E FPGA (U1)
 - Micron N25Q256A11E 256 Mb Quad SPI flash Memory (U35)
 - Four Micron EDY4016AABG-DR-F-D DDR4 512 MB Memories (U60-U63)
- Power supply: 100–240 VAC input, 12 VDC 5.0A output (included with the KCU105 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 or later, Design Edition with SDK
- Tera Term terminal emulator program (version 4.87 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 or later.
2. Download and install Tera Term (version 4.87 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]

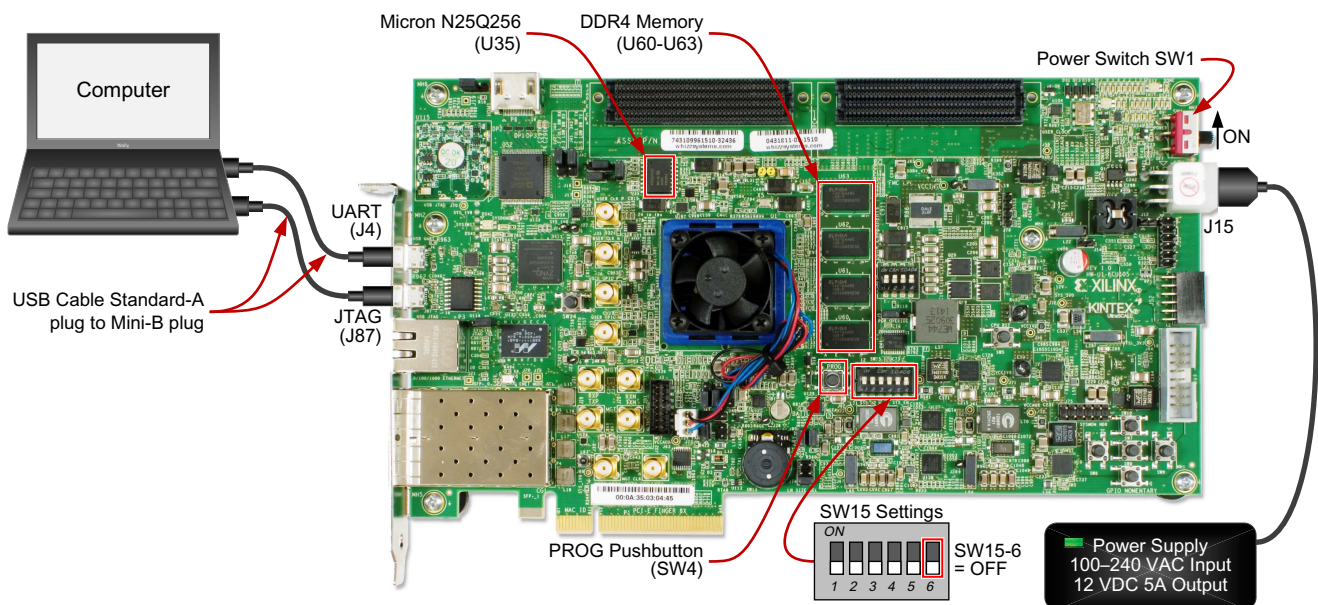


TIP: The UART communication settings are set up later in this procedure.

Set Up KCU105 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 (J87) on the KCU105 board and the computer.
3. Set the FPGA M2 pin to 0 by setting DIP switch SW15-6, to OFF as shown in [Figure 3](#). This selects the FPGA SPI configuration mode ($M[2:0] = 001$) because the FPGA $M[1:0]$ pins are hard-wired to 01.



X16224-030116

Figure 3: Connection Diagram for Preliminary Setup

Download Reference Design Files

1. Download and unzip the [Reference Design Files](#) to `c:\.` [Figure 4](#) shows the resulting file structure that will be created under the `c:\Xapp_QUAD_SPI` directory.

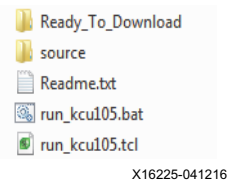


Figure 4: Reference Design File Structure

2. Do one of the following:
 - Go to [Generate Reference Design Project](#) (do this to perform the steps to create an SDK project and become more familiar with the reference design code).
 - Skip to [Configure FPGA with golden.bin](#) (do this to run the demonstration using the pre-generated bitstreams supplied with the reference design).

Generate Reference Design Project

1. Go to `c:\Xapp_QUAD_SPI`. Double-click `run_kcu105.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 90 minutes depending on host computer.

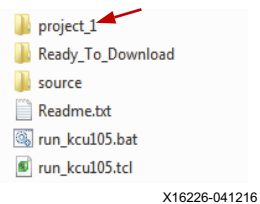
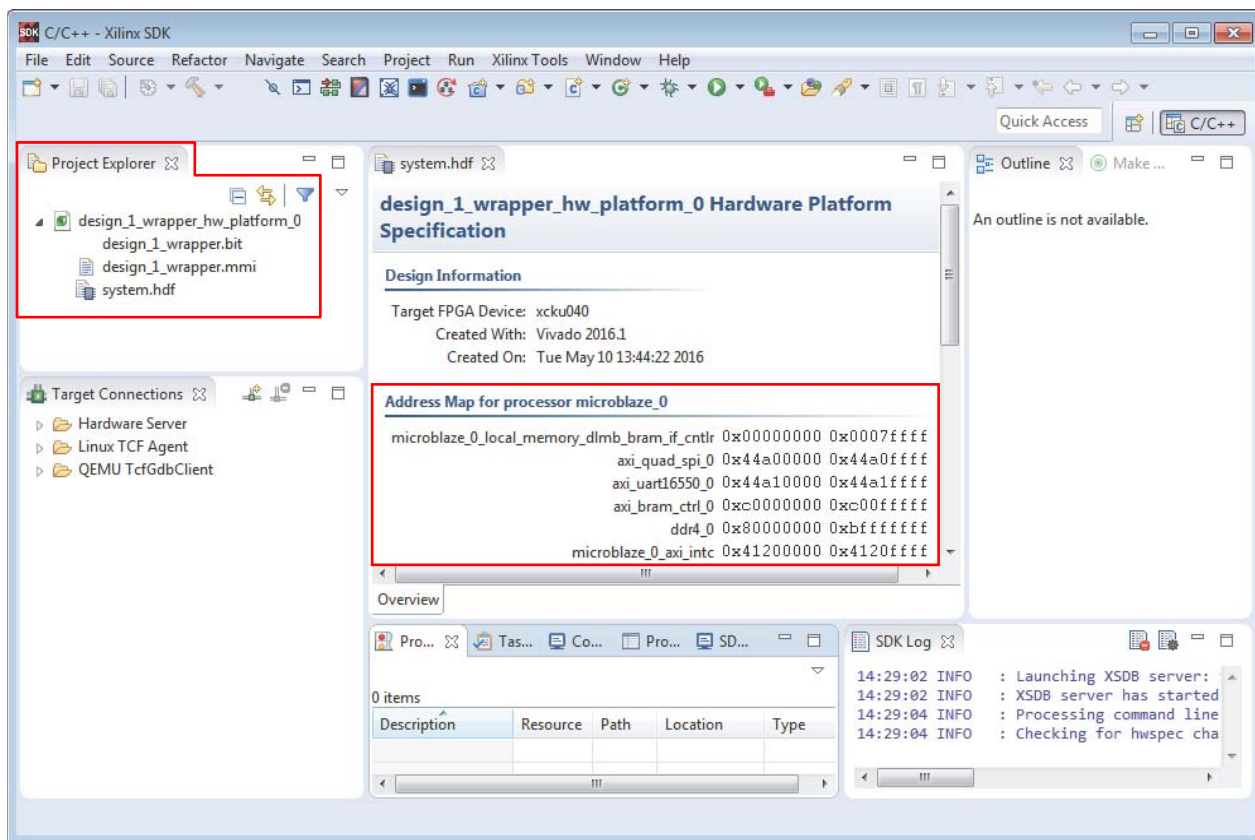


Figure 5: New Project Directory



IMPORTANT: Critical implementation warning [Place 30-73] is expected and can be ignored.

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.



X16227-051216

Figure 6: Project in the SDK IDE

2. Create an application for the Quad SPI 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 QUAD_SPI_RW.
 - c. Select, enter, or verify the values in the remaining fields shown in [Figure 7](#).

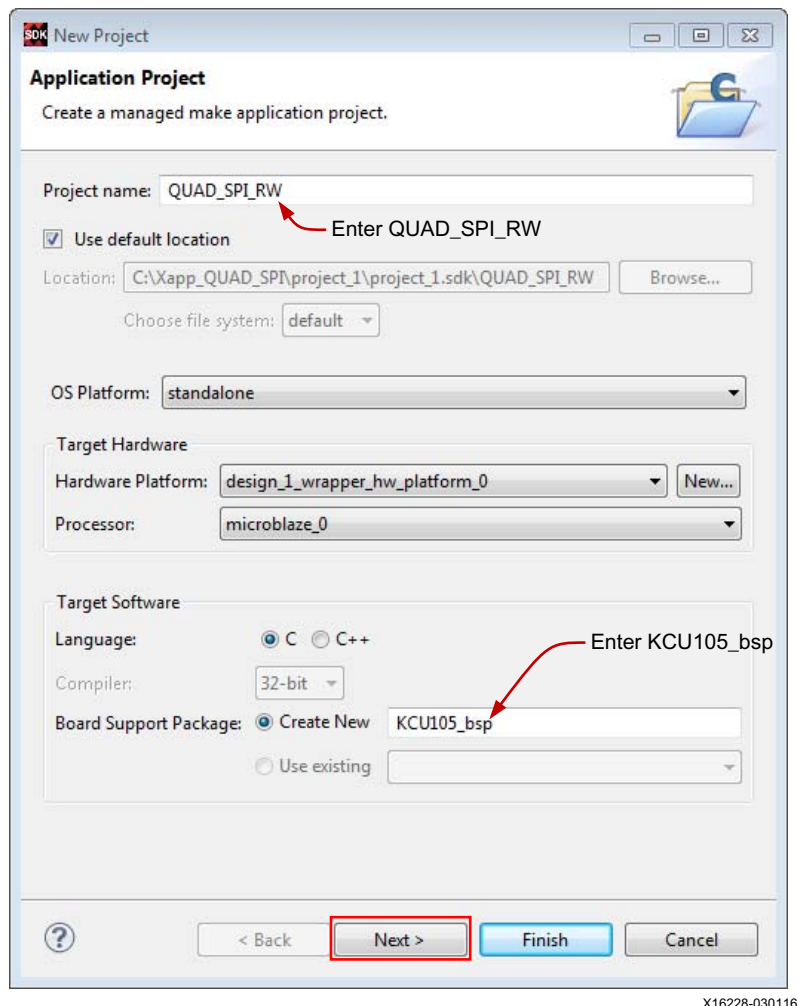
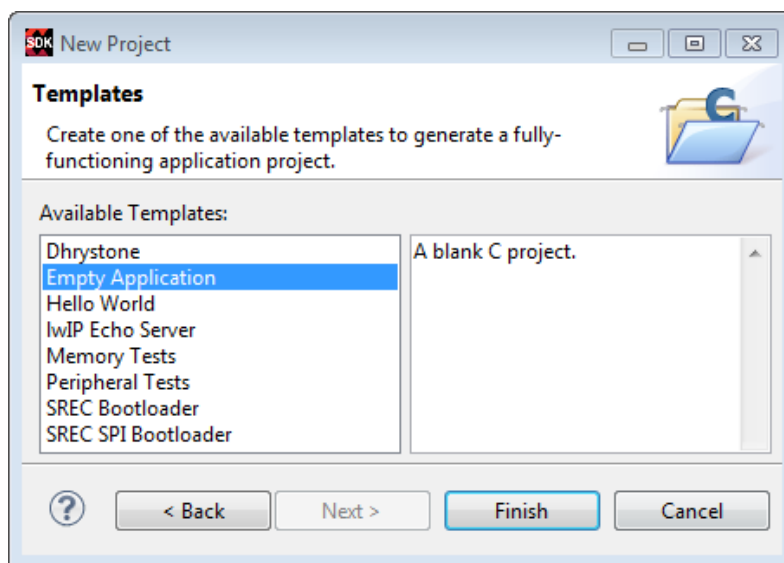


Figure 7: Create Application Project

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

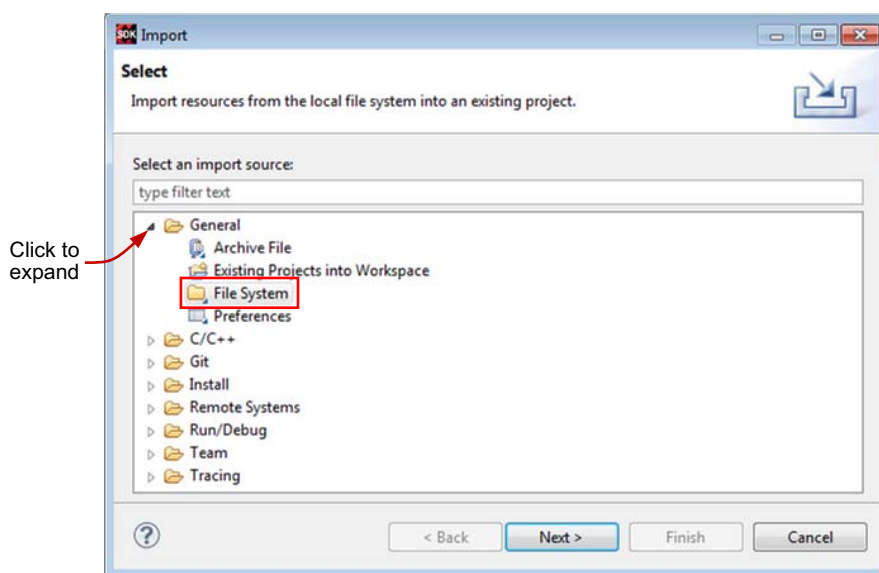
- e. Under Available Templates select **Empty Application** (Figure 8).



X16229-030116

Figure 8: Available Templates

- f. Click **Finish** to build the application project.
- g. In the SDK IDE Project Explorer tab, double-click the folder **QUAD_SPI_RW** and select **src**. Right-click on **src** and select **Import**.
- h. In the Import window, expand the **General** folder and select **File System** (Figure 9).

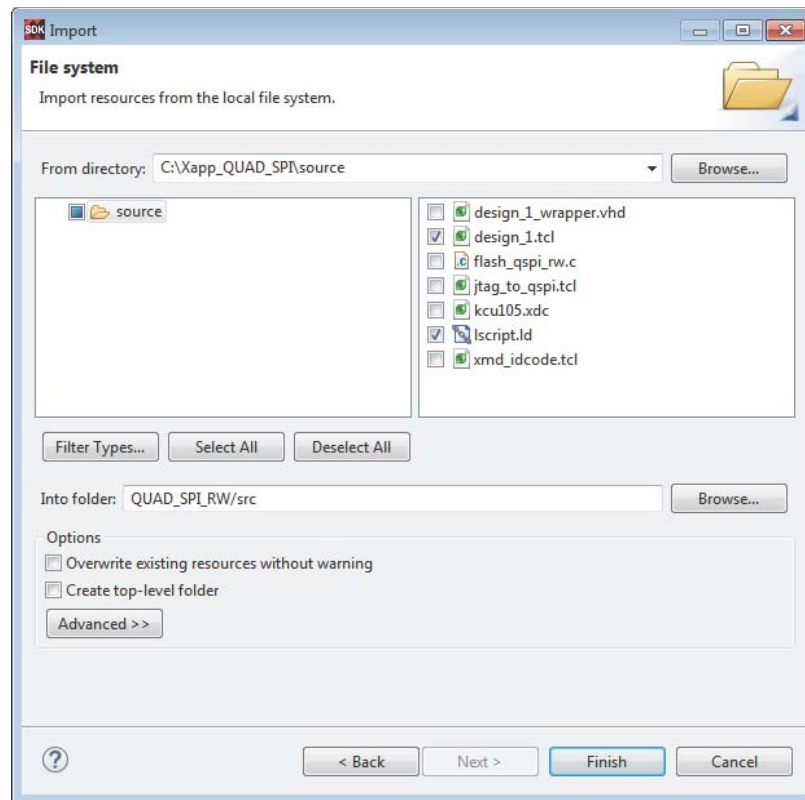


X16230-030116

Figure 9: Selecting the Local File System as an Import Resource

- i. Click **Next**.

- j. Browse to the C:\Xapp_QUAD_SPI\source directory, click **OK**, and select the **flash_qspi_rw.c** and **lscript.ld** (Figure 10).



X16231-051216

Figure 10: Import Files from the Local File System

- k. Click **Finish** to add both files to the project and then click **Yes** when asked to Overwrite the 'lscript.ld' in folder 'QUAD_SPI_RW\src'.
3. **(Optional Step) Review the MicroBlaze code:** In the SDK project explorer double-click **flash_qspi_rw.c** to open the file and review the MicroBlaze code. This code controls the UART/Tera Term command menu, defines the parameters of the SPI flash memory and controls reads and writes to memory:
- The SPI flash memory commands and their definitions are located near the top of the code listing under: `/** Definitions of the commands shown in this example. */`. The parameters and commands should match the targeted SPI flash memory device data sheet. The command sequences are described in [Software System Details](#).
 - Flash-sizing information is located under: `/** Number of bytes per page in the flash device */`. This information includes definitions for the flash memory page size, number of sectors, bytes per sector and other information. These parameters should match the data sheet values for the targeted SPI flash memory device.
 - Descriptions of the functions that are used to build each menu operation are located under: `/**Function Prototypes */`. The low level functions can be examined in detail by searching on the function name in the file. At the lowest level, all the operations that are needed for the flash memory can be created using the first six

functions: `SpiFlashWriteEnable()`, `SpiFlashWrite()`, `SpiFlashRead()`, `SpiFlashBulkErase()`, `SpiFlashGetStatus()`, and `SpiFlashQuadEnable()`.

Generate Programming Files

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



TIP: Generation of these files will take 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](#).

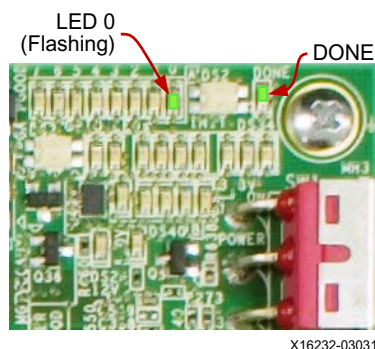
1. Turn on power to the KCU105 board by placing switch SW1 to the ON position as shown in [Figure 3](#). The power good LED, DS3 (located near SW1) illuminates green to indicate the board power is on and the power system is operating properly.
2. Go to `C:\Xapp_QUAD_SPI\Ready_To_Download`. Double-click `load_golden.bat` to run the batch file.

After `golden.bin` (the reference design initial bitstream) has been copied into the SPI flash memory at address `0x00000000`, the batch file will issue an FPGA reset to initiate FPGA configuration from the SPI flash memory which now contains the `golden.bin` image.



TIP: Programming the SPI flash memory will take 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 11](#)).



X16232-030316

Figure 11: The Reference Design Initial Bitstream `golden.bin` is Running



TIP: The `golden.bin` image header contains the next configuration address in the `BITSTREAM.CONFIG.NEXT_CONFIG_ADDR` property and fallback is enabled by the `BITSTREAM.CONFIG.CONFIGFALLBACK` property being set to `ENABLE`. When configuration begins during [step 2](#), the FPGA starts loading the `golden.bin` image starting at address `0x00000000` and when the header is read it will jump to the address contained in `BITSTREAM.CONFIG.NEXT_CONFIG_ADDR` (`0x00F50000`) and attempts to continue configuration. However, this location is initially blank because the `update.bin` image at `0x00F50000` has not yet been loaded, and the FPGA will fall back to load the `golden.bin` image. The FPGA will be configured with the `update.bin` image only after it has been programmed which occurs in the next section.

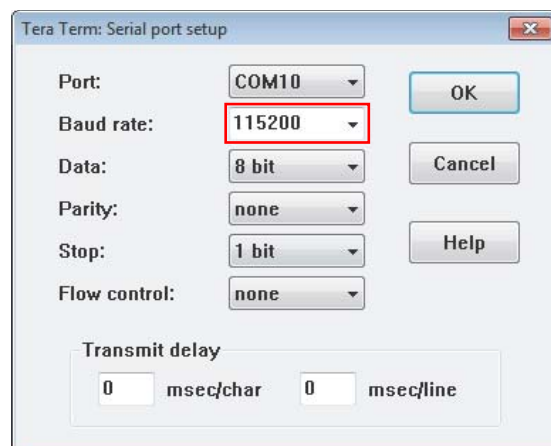
Refer to [\[Ref 1\]](#) and [\[Ref 15\]](#) for details on the MultiBoot and fallback features.

Program SPI 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 SPI flash at address `0x00F50000`, and then be used to reconfigure the FPGA.

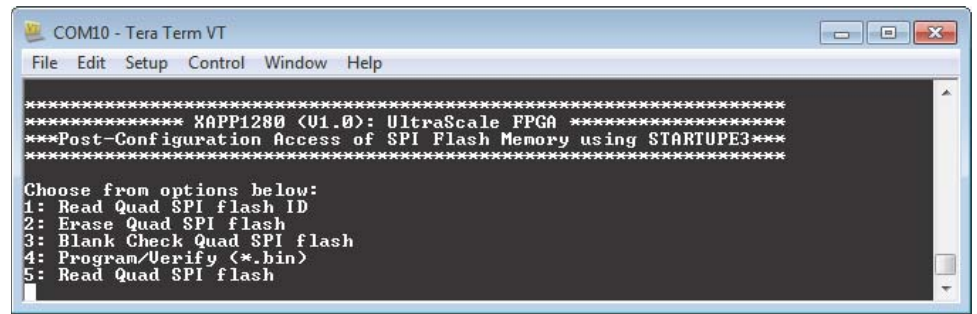
1. Confirm the host computer and KCU105 evaluation board are connected as shown in [Figure 3](#).
2. Launch and set up Tera Term using the values shown in [Figure 12](#). Ensure that the Standard COM Port is selected (Refer to the Checklist and Debug Tips [step 3](#) for details).



X16234-032316

Figure 12: Tera Term Serial Port Settings

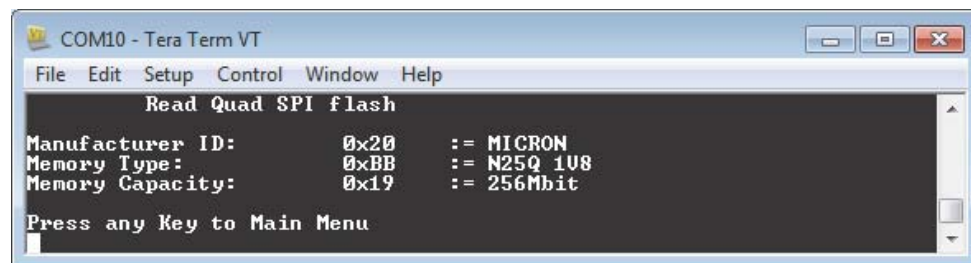
- Pulse the KCU105 PROG (SW4) pushbutton to load the `golden.bin` code. If the serial connection set up properly, Tera Term will display the reference design menu shown in Figure 13.



X16235-030116

Figure 13: Reference Design Flash Memory Command Menu

- In the Tera Term window, enter **1** to display the SPI flash manufacturer information and ensure proper board setup (Figure 14).



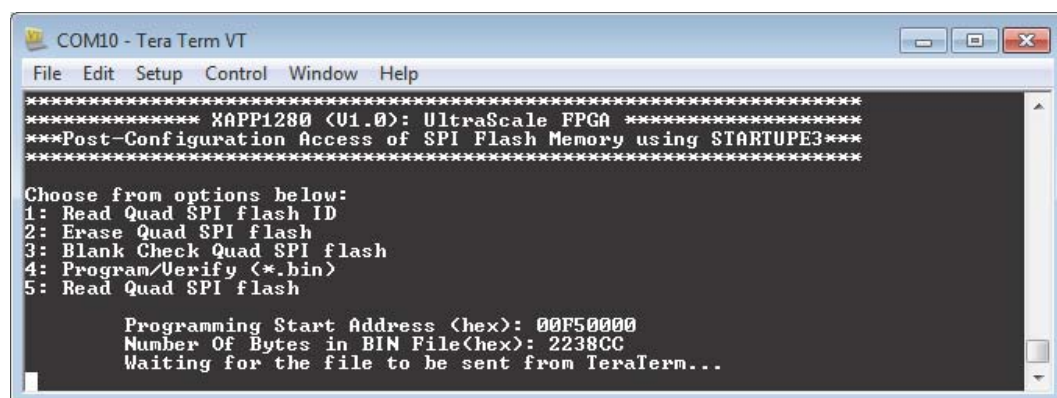
X16236-030116

Figure 14: Option 1: SPI Flash Memory Information



TIP: The operations **3: Blank Check Quad SPI flash** and **5: Read Quad SPI flash** can also be performed. Both operations require a byte count to be input and will display results to the nearest page boundary (256 bytes). To perform a Bulk Erase, select **2: Entire flash Erase (bulk erase)** and at the next prompt, select **1**.

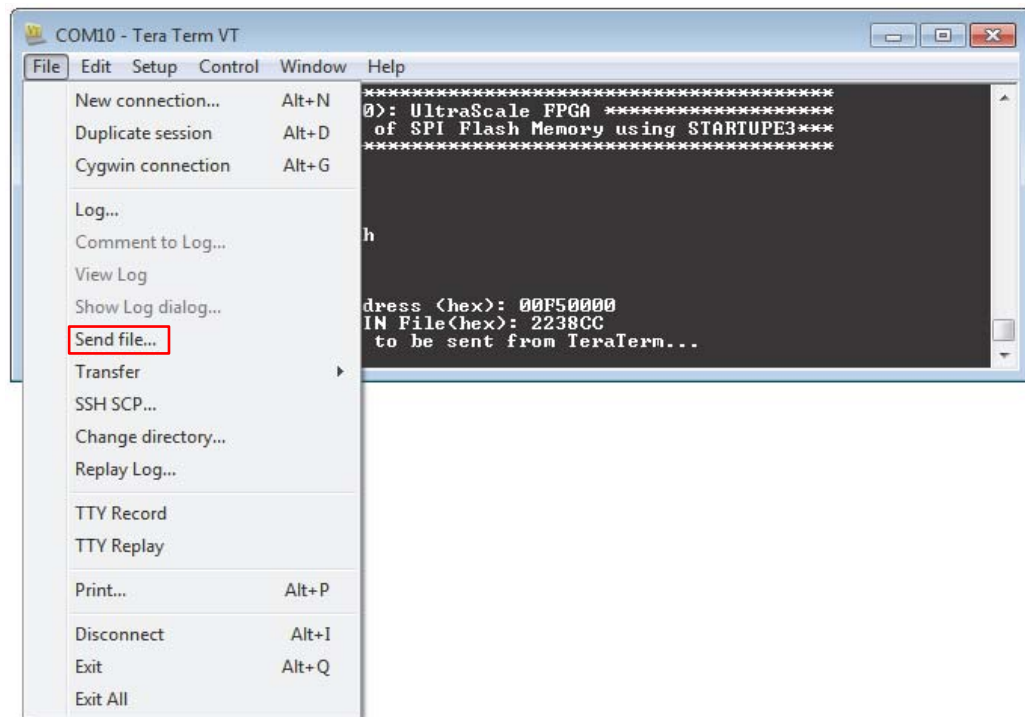
- Enter **4** to program and verify the SPI flash memory with the `update.bin` image at offset `0x00F50000` (Figure 15) and enter `0x2238CC` number of bytes when prompted.



X16237-051216

Figure 15: Option 4: Erase, Program, Verify

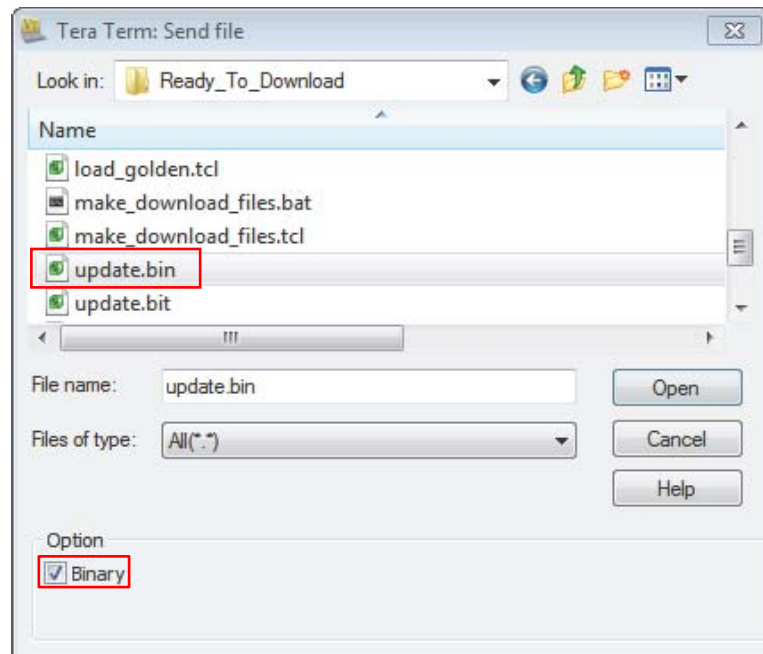
6. In the menu bar, click **File > Send file...** (Figure 16).



X16238-040116

Figure 16: Send File Command

7. Choose `update.bin` under `C:\Xapp_QUAD_SPI\Ready_To_Download`. Select **Binary** and click **Open** (Figure 17).



X16556-051216

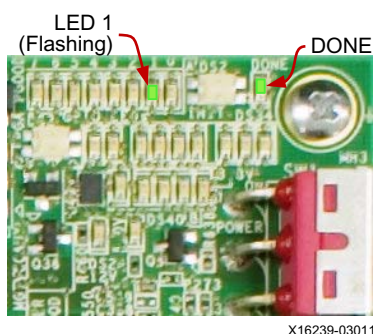
Figure 17: Select `update.bin`

Configure FPGA with `update.bin`

This part of the procedure corresponds to step 3 in [Figure 1](#).

The `update.bin` (the Update Bitstream image) has been moved into the SPI flash memory at the address `0x00F50000`. To reconfigure the FPGA:

1. Pulse the PROG pushbutton (SW4). When the FPGA is reset, it will start at address `0x00000000` and the `golden.bin` image header will be read which has a next config address command pointing to `0x00F50000`. The FPGA will jump to that address and configure the FPGA from the `update.bin` image.
2. 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 18](#) (replacing the original `golden.bin` LED 0 pattern).



X16239-030116

Figure 18: The Reference Design Update Bitstream `update.bin` is Running



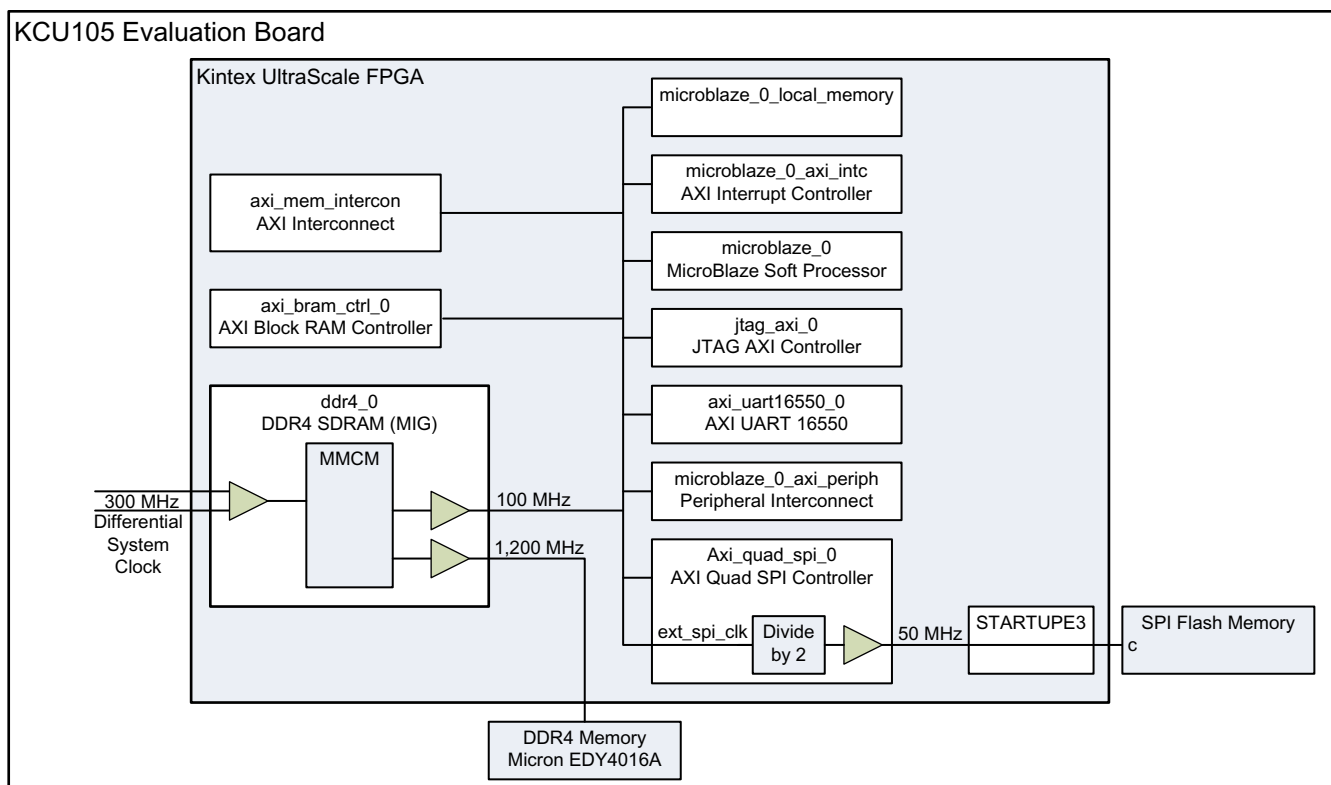
TIP: After `update.bin` is loaded into the FPGA, SPI flash operations can no longer be performed. To run the demonstration again, execute the procedure starting at [Configure FPGA with `golden.bin`](#).

Hardware System Details

The reference design includes a MicroBlaze soft processor core and peripheral IP cores to implement downloading the `update.bin` file. This section describes the system clocking topology, AXI Quad SPI core settings, STARTUPE3 primitive usage, and the constraints for post-configuration SPI 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 KCU105 board. The clock distribution is shown in Figure 19.



X16240-051716

Figure 19: Reference Design Clocking Topology

The differential clock is supplied to a mixed-mode clock manager (MMCM) inside the DDR4 SDRAM interface that outputs 1,200 MHz, 300 MHz, and 100 MHz. The 1,200 MHz clock is supplied to the external DDR4 memory, the 300 MHz clock is supplied to the AXI Block RAM controller and AXI memory interconnect module, and the 100 MHz clock is supplied to the MicroBlaze interrupt controller, JTAG-to-AXI controller, local memory controller, AXI 16550 UART, AXI peripheral interconnect module, and the AXI Quad SPI Controller.

The AXI Quad SPI Controller core receives the 100 MHz clock and uses it to clock transactions on the AXI interface. Depending on the customization selections of the this module, the clock

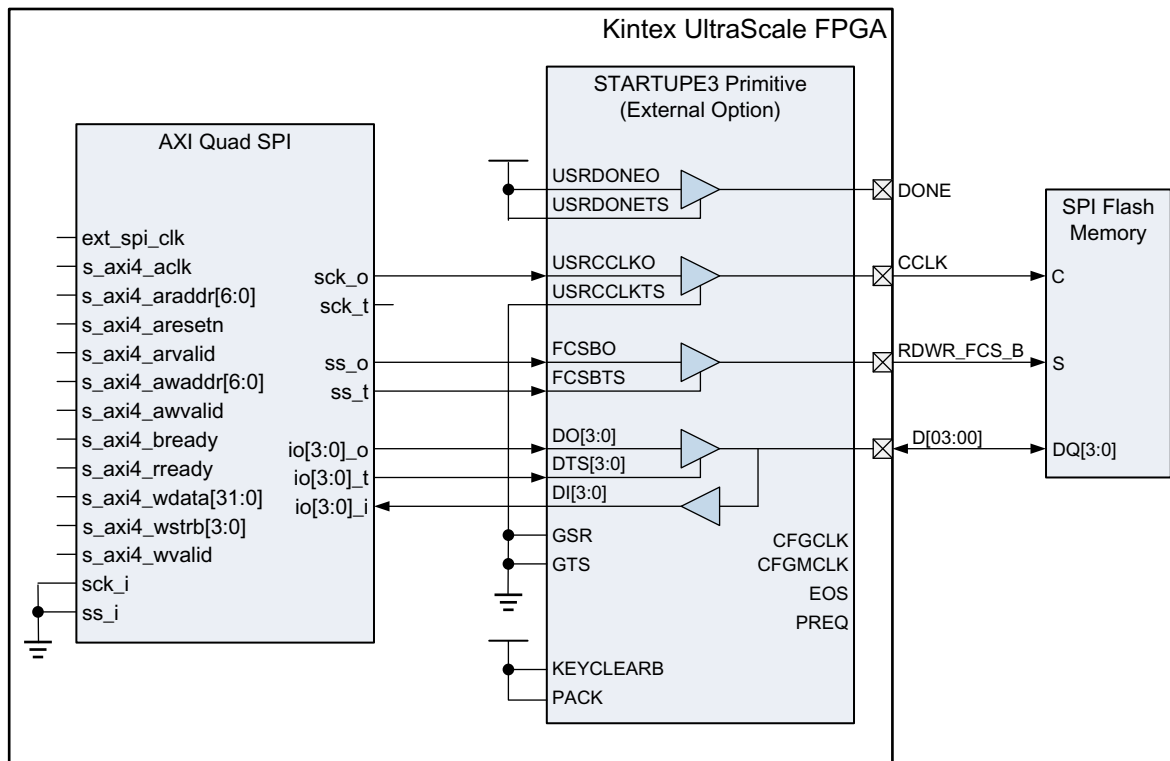
provided on the ext_spi_clk pin can be divided before being sent via the STARTUPE3 block to the clock input (C) of the SPI flash memory. In the case of this reference design, the 100 MHz clock connected to the ext_spi_clk pin is divided by 2 to provide a 50 MHz clock to the SPI flash memory.



IMPORTANT: This reference design is targeted for use with the Micron N25Q256A11E SPI flash memory provided on the KCU105 evaluation board. If the design is modified for use with other SPI flash memories, the clock frequency supplied to the MicroBlaze system, and to the SPI flash memory via the STARTUPE3 block might require adjustment.

AXI Quad SPI Core and STARTUPE3 Primitive

The connections required for post-configuration between the AXI Quad SPI core, the STARTUPE3 primitive, the FPGA SPI interface pins, and the external SPI flash are shown in Figure 20.

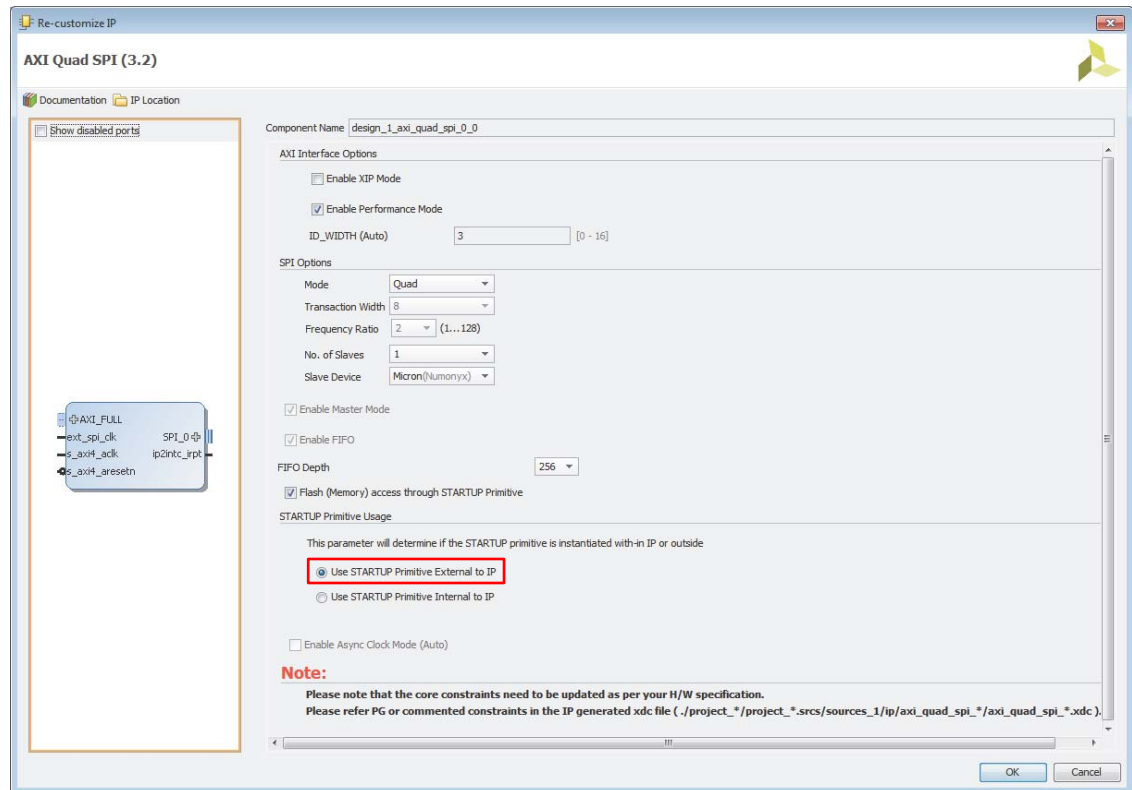


X16242-032316

Figure 20: AXI Quad SPI Core to STARTUPE3 Connectivity

Customizing the AXI Quad SPI Core

The AXI Quad SPI core is setup specifically for the Kintex UltraScale XCKU040-2FFVA1156E FPGA, and the Micron N25Q256A11E SPI flash memory on the KCU105 evaluation board. The IP Integrator window shown in [Figure 21](#) is used for customization.



X16243-051216

Figure 21: AXI Quad SPI Core Settings

The selections shown in the re-customization window are used by this application note reference design. The reason for the selections are described below:

- The **Enable Performance Mode** selects the full AXI4 interface instead of the default AXI4-Lite interface. See *AXI Quad SPI Product Guide* (PG153) [\[Ref 14\]](#) for more information.
- **ID_WIDTH (Auto) 3** is used because the SPI flash memory in Quad mode only responds with three valid bytes for the multiple I/O read ID command.
- The **Mode** is set to the **Quad** to support faster operation performance on the SPI flash with x4 data bus width instead of the default serial x1 option.

Selecting quad mode also defaults:

- The data bus **Transaction Width** to **8** for standard instructions.
- The **Frequency Ratio** is set to **2** to divide the 100 MHz clock on the ext_spi_clk pin by 2.
- **Master Mode** is selected because the AXI Quad SPI core is the SPI master.
- **Enable FIFO** is selected because the FIFO requires buffering in quad mode.

- **No. of Slaves** is set to **1** because the SPI flash memory is the only slave device.
- **Micron (Numonyx)** is selected in the **Slave Device** field because this is the type of SPI flash memory available on the KCU105 board.
- The **Use STARTUP Primitive External to the IP** is selected to make the connections to the STARTUPE3 primitive visible outside the AXI Quad SPI core.



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 Quad SPI core then the setting **Use STARTUP primitive internal to IP** could be used.

STARTUPE3 Primitive

The dedicated UltraScale FPGA SPI 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 SPI flash memory. To access these dedicated pins, the design instantiates the STARTUPE3 primitive. Table 2 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 2: STARTUPE3 SPI Interface Dedicated Pins

Logic Signal	STARTUPE3 Port	Dedicated Pin Name	FPGA (U1) Pin Location
spi_0_ss_o(0)	FCSBO	RDWR_FCS_B_0	U7
spi_0_sck_o	USRCCLKO	CCLK_0	AA9
flash_dq_o(3)	DO[3]	D03_0	Y7
flash_dq_o(2)	DO[2]	D02_0	AA7
flash_dq_o(1)	DO[1]	D01_DIN_0	AB7
flash_dq_o(0)	DO[0]	D00_MOSI_0	AC7
flash_dq_i(3)	DI[3]	D03_0	Y7
flash_dq_i(2)	DI[2]	D02_0	AA7
flash_dq_i(1)	DI[1]	D01_DIN_0	AB7
flash_dq_i(0)	DI[0]	D00_MOSI_0	AC7

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 SPI flash 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 spi_0_sck_o externally to the flash is accomplished by connecting it to the USRCCLKO port of the STARTUPE3 instantiation. There is no explicit port declaration for

sck 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 SPI 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.

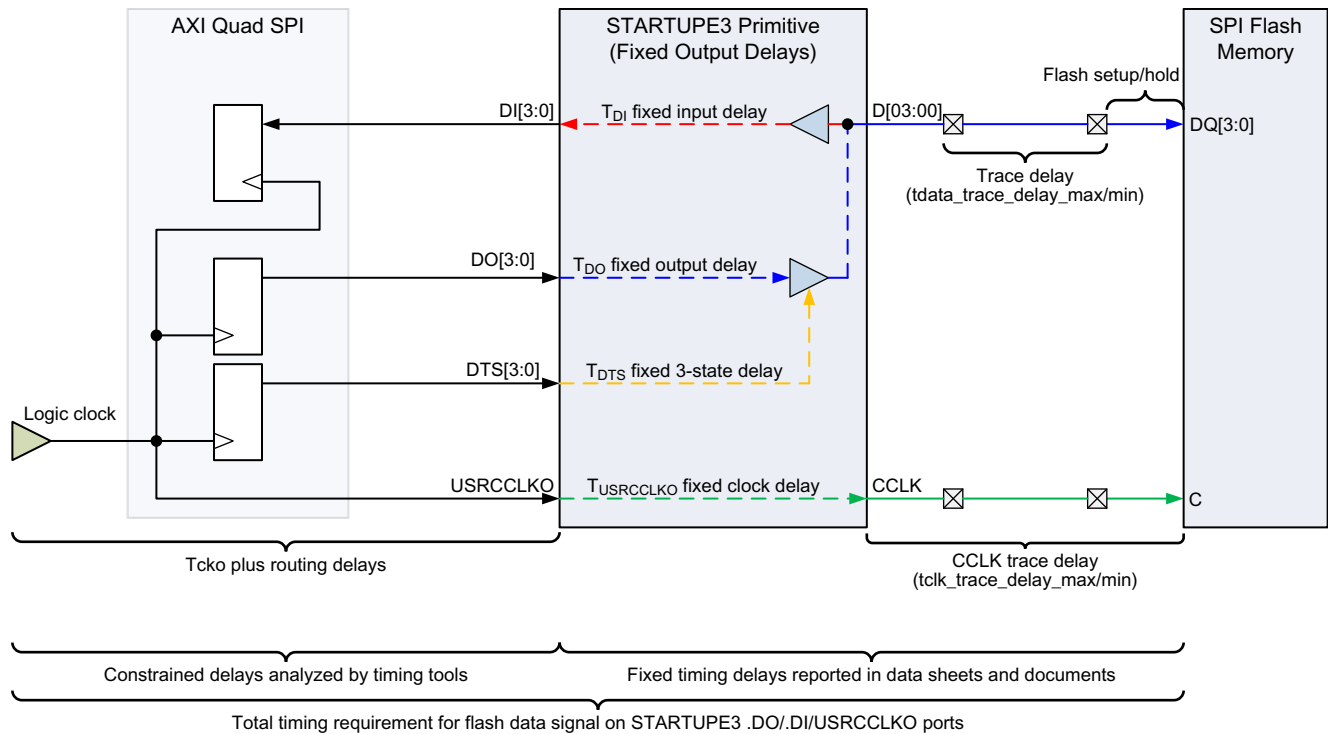
Since the connections between the AXI Quad SPI core and the STARTUPE3 primitive are made inside the top-level `design_1_wrapper.vhd` wrapper, the STARTUPE3 block does not appear in the IP Integrator block design. The instantiation of the STARTUPE3 is shown below:

```
-- STARTUPE3: STARTUP Block
-- Kintex UltraScale+
-- Xilinx HDL Language Template, version 2016.1
STARTUPE3_inst : STARTUPE3
-----
generic map
(
    PROG_USR      => "FALSE",      -- Activate program event security feature. Requires encrypted bitstreams.
    SIM_CCLK_FREQ => 0.0           -- Set the Configuration Clock Frequency (ns) for simulation
)
port map
(
    USRCCLKO => spi_0_sck_o,      -- 1-bit input: User CCLK input
    -----
    CFGCLK   => open,             -- 1-bit output: Configuration internal oscillator clock output
    CFGMCLK  => open,             -- 1-bit output: Configuration internal oscillator clock output
    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       => flash_dq_o,       -- 4-bit input: Allows control of the D pin output
    DI       => flash_dq_i,       -- 4-bit output: Allow receiving on the D input pin
    DTS      => flash_dq_t,       -- 4-bit input: Allows tristate of the D pin
    FCSBO    => spi_0_ss_o_0(0),  -- 1-bit input: Controls the FCS_B pin for flash access
    FCSBTS   => spi_0_ss_t,       -- 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 (BBRAM)
    PACK     => PACK_int,         -- 1-bit input: PROGRAM acknowledge input
    USRCCLKTS => spi_0_sck_t,     -- 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 of STARTUPE3_inst instantiation
```

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 ensure that the total path delays for the signals going into or coming out of the STARTUPE3 block meet timing requirements.

The fixed delay of the STARTUPE3 primitive is not automatically considered by timing tool analysis, and the user constraints must be written to take this into account. Because the timing tools assume user design timing ends at the USRCCLKO port, it is not possible to create a generated clock on the external CCLK pin. Instead, a generated clock is created on the USRCCLKO port that takes the STARTUPE3 delay, board delay, and setup time requirements for the SPI flash into account. [Figure 22](#) shows the timing paths that needs to be considered.



X16245-032316

Figure 22: 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 SPI flash memory are obtained from the *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892) [Ref 5] and the *Micron Serial NOR Flash Memory N25Q256A11E Data Sheet* [Ref 6]. Sample parameters are provided in Table 3 and Table 4 to be used for the constraints.

Table 3: Kintex UltraScale Timing Parameters

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

Table 4: Micron N25Q256A11E SPI Flash Memory Timing Parameters

Symbol	XDC Names	Descriptions	Units
TCLQX (min)	Tco_min	Output hold time (clock LOW)	1 ns
TCLQV (max)	Tco_max	Clock LOW to output valid under 30pF (DTR)	8 ns
TDVCH (min)	Tsu	Data in setup time	2 ns
TCHDX (min)	Th	Data in hold time	3 ns

Example constraints for the KCU105 board are shown below. The timing delays for the board, and the Micron N25Q256A11E flash memory are obtained from measurement and the flash data sheet.

```
##### STARTUPE3 parameters
set cclk_delay 6.7 # Tusrccclk maximum value

##### SPI device parameters
set tco_max 8 # MAX Tco
set tco_min 1 # MIN Tco
set tsu 2 # SPI setup time requirement
set th 3 # SPI hold time requirement

#### BOARD parameters-assumes data trace lengths are matched
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
##### Constraints
# Define a SCK Clock for the Quad SPI IP. Following command creates a divide by 2 clock.
#It also takes into account the delay added by STARTUP block to route the CCLK

create_generated_clock -name clk_sck -source [get_pins -hierarchical
*axi_quad_spi_0/ext_spi_clk] -edges {3 5 7} -edge_shift [list $cclk_delay $cclk_delay
$cclk_delay] [get_pins -hierarchical *USRCCLKO]
set_multicycle_path -setup -from clk_sck -to [get_clocks -of_objects [get_pins
-hierarchical */ext_spi_clk]] 2
set_multicycle_path -hold -end -from clk_sck -to [get_clocks -of_objects [get_pins
-hierarchical */ext_spi_clk]] 1
set_multicycle_path -setup -start -from [get_clocks -of_objects [get_pins -hierarchical
*/ext_spi_clk]] -to clk_sck 2
set_multicycle_path -hold -from [get_clocks -of_objects [get_pins -hierarchical
*/ext_spi_clk]] -to clk_sck 1

# Max delay constraints are used to instruct the tool to place IP near the STARTUPE3
#primitive. If needed adjust the delays appropriately. The data path is defined by the
#clock source of ext_spi_clk and is connected from the DDR MMCM UI Clock output
#design_1_ddr4_0_1_c0_ddr4_ui_clk

set_max_delay -from [get_pins -hier {*STARTUP*_inst/DI[*]}] 1.000 -datapath_only
set_max_delay -from [get_clocks design_1_ddr4_0_1_c0_ddr4_ui_clk] -to [get_pins -hier
{*STARTUP*_inst/USRCCLKO}] 1.000 -datapath_only
set_max_delay -from [get_clocks design_1_ddr4_0_1_c0_ddr4_ui_clk] -to [get_pins -hier
{*STARTUP*_inst/DO[*]}] 1.000 -datapath_only
```

These constraints account for the path delay to and from the AXI Quad SPI logic, the setup and hold requirements of the SPI 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 `kcu105.xdc` constraints file.

SPI Configuration Mode Constraints

The correct properties must be used during `golden.bin` file generation to ensure a successful master SPI x4 configuration. The properties listed below are included in the reference `.xdc` constraint file supporting the KCU105 evaluation board 1.8V SPI flash. For additional details on the options see *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1] and *SPI Configuration and Flash Programming in UltraScale FPGAs* (XAPP1233) [Ref 7].

```
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property BITSTREAM.CONFIG.SPI_32BIT_ADDR YES [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property BITSTREAM.CONFIG.SPI_FALL_EDGE YES [current_design]
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
set_property CONFIG_MODE SPIx4 [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.NEXT_CONFIG_REBOOT Enable [current_design]
set_property BITSTREAM.CONFIG.NEXT_CONFIG_ADDR 0x00F50000 [current_design]
```

Software System Details

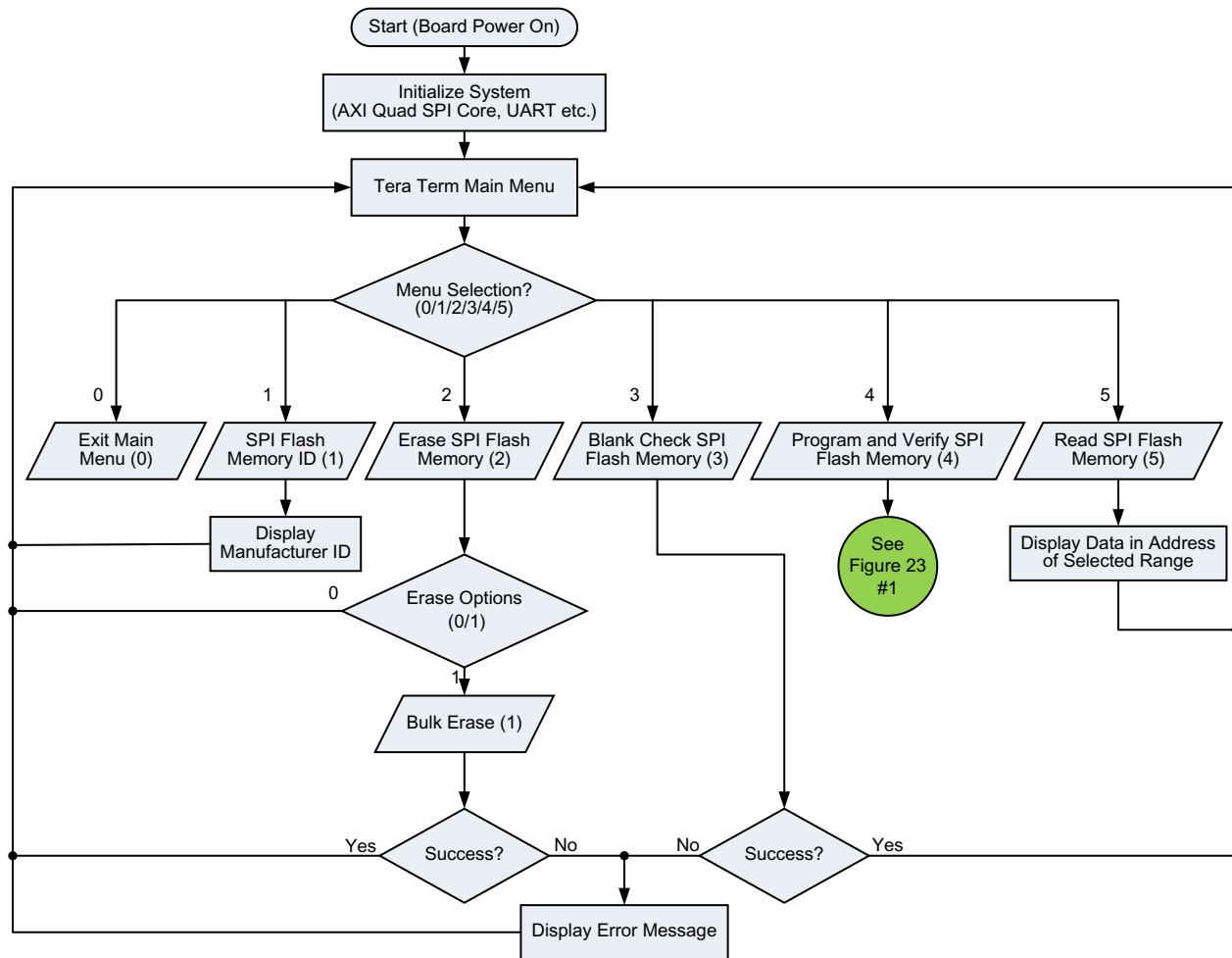
Software running on the MicroBlaze processor drives the AXI Quad SPI core to read and write to the SPI flash memory using the hardware connections to the STARTUPE3 primitive. The `flash_qspi_rw.c` file included with this reference design provides example low-level software read and write operations. These read and write functions are used to implement the Micron flash memory operations described by the command descriptions in the *Micron Serial NOR Flash Memory N25Q256A11E Data Sheet* [Ref 6]. This memory is located on the KCU105 evaluation board at location U35 (see Figure 3).

The functions in `flash_qspi_rw.c` are provided as a starting point for creating memory operations that might be necessary with flash memories from other vendors. Most SPI 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 N25Q256A11E SPI flash memory used on the KCU105 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 SPI flash memory. All SPI transactions in master mode depend upon commands supported by a slave device connected to the AXI QUAD SPI core. Refer to the respective SPI flash memory data sheet for the supported commands. The Micron N25Q256A11E SPI 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 13. The

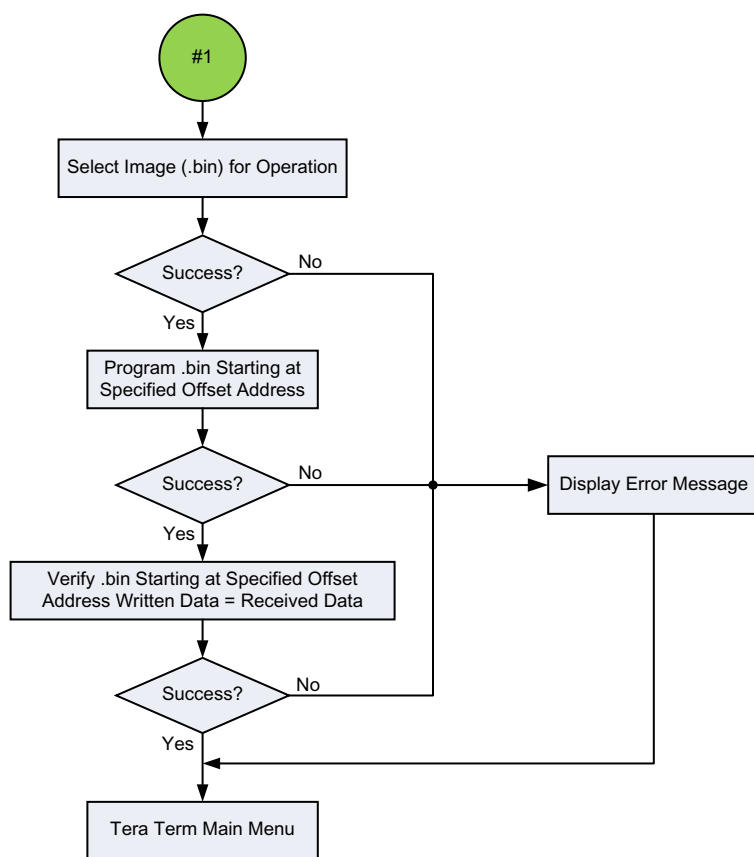
commands implemented for the SPI flash on the KCU105 are shown in the flow diagrams in [Figure 23](#) and [Figure 24](#).



X16246-030116

Figure 23: Flash Command Set Flow Diagram

The flowchart for Program/Verify (Figure 15 Tera Term main menu option 4).



X16247-030116

Figure 24: Menu Option 4 (Program/Verify) Flow Diagram

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

Write enable command sequence: The write enable command must be issued before every write transaction (erase/program data to the flash/register write of flash) command to the flash. Disable the master transaction by asserting the master inhibit bit of SPI control register (SPICR) and reset the RX and TX FIFOs through SPICR. Example: Write 0x1E6 to SPICR.

1. Issue the write enable command by writing 0x06 into SPI Data Transmit Register (SPIDTR).
2. Issue chip select by writing 0x00 to SPI Slave Select Register (SPISSR).
3. Enable master transaction by deasserting the SPICR, master inhibit bit.
4. Deassert chip select by writing 0x01 to SPISSR.
5. Disable master transaction by asserting the SPICR master inhibit bit.

Erase Command Sequence:

1. Reset RX and TX FIFOs through SPICR.
2. Issue the bulk erase command to erase the entire flash followed by the flash base address.
Example: Write 0xC7 to SPIDTR

3. Issue chip select by writing 0x00 to SPISSR.
4. Enable master transaction by deasserting the SPICR master inhibit bit.
5. Deassert chip select by writing 0x01 to SPISSR.
6. Disable master transaction by asserting the SPICR master inhibit bit.

Program Data Command Sequence:

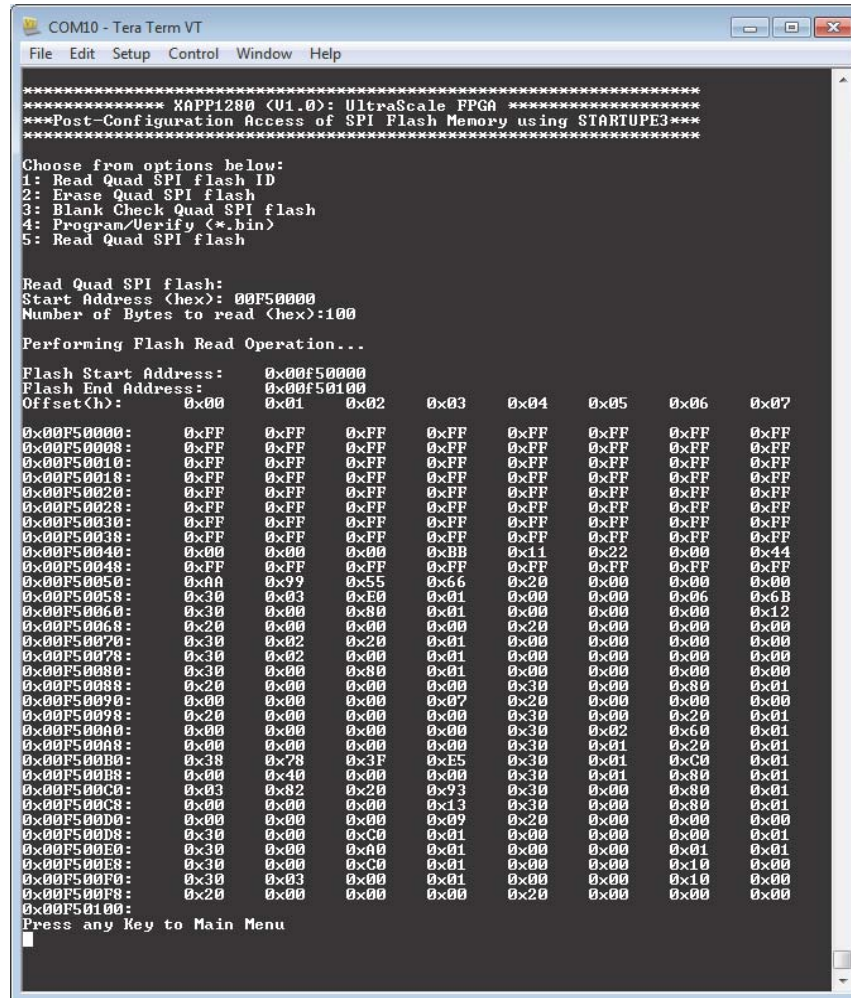
1. Reset RX and TX FIFOs through SPICR.
2. Issue the write data command into SPIDTR to write data into any specific sector followed by the flash sector address. Fill SPIDTR with the data to be written to flash. The maximum data size depends upon the configured QSPI FIFO size.
3. Issue chip select by writing 0x00 to SPISSR. Enable master transaction by deasserting the SPICR master inhibit bit.
4. Deassert chip select by writing 0x01 to SPISSR.
5. Disable master transaction by asserting the SPICR master inhibit bit.

Read Data Command Sequence:

1. Reset RX and TX FIFOs through SPICR.
2. Issue the read data command into SPIDTR to read data from any specific sector followed by the flash sector address. Fill SPIDTR with the dummy data to read required data from the flash.
3. Issue chip select by writing 0x00 to SPISSR.
4. Enable master transaction by deasserting the SPICR master inhibit bit.
5. De-assert chip select by writing 0x01 to SPISSR.
6. Disable master transaction by asserting SPICR master inhibit bit.
7. Read SPIDRR, to get the read data that is received from the SPI bus.

The reference design menu option 4 performs program > verify with one selection (see [Figure 24](#)). The verify is a read of the full .bin file compared to the original .bin contents. If an error is detected the address of the failure will be provided.

[Program SPI Flash Memory with update.bin](#) showcases option 4 in [step 5](#), however, the Tera Term main menu also has individual operation options. When the individual operation option is selected ([Figure 15](#), option 1, 2, 3, or 5) then the desired start address must be supplied along with either the stop address or number of bytes. An example read instruction is shown in [Figure 25](#).



```

COM10 - Tera Term VT
File Edit Setup Control Window Help

***** XAPP1280 (v1.0): UltraScale FPGA *****
***Post-Configuration Access of SPI Flash Memory using STARTUPE3***
*****

Choose from options below:
1: Read Quad SPI flash ID
2: Erase Quad SPI flash
3: Blank Check Quad SPI flash
4: Program/Verify (*.bin)
5: Read Quad SPI flash

Read Quad SPI flash:
Start Address (hex): 00F50000
Number of Bytes to read (hex):100

Performing Flash Read Operation...

Flash Start Address: 0x00F50000
Flash End Address: 0x00F50100
Offset(h): 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07

0x00F50000: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50008: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50010: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50018: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50020: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50028: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50030: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50038: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50040: 0x00 0x00 0x00 0xBB 0x11 0x22 0x00 0x44
0x00F50048: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x00F50050: 0xA0 0x99 0x55 0x66 0x20 0x00 0x00 0x00
0x00F50058: 0x30 0x03 0xE0 0x01 0x00 0x00 0x00 0x00
0x00F50060: 0x30 0x00 0x00 0x01 0x00 0x00 0x00 0x12
0x00F50068: 0x20 0x00 0x00 0x00 0x20 0x00 0x00 0x00
0x00F50070: 0x30 0x02 0x20 0x01 0x00 0x00 0x00 0x00
0x00F50078: 0x30 0x02 0x00 0x01 0x00 0x00 0x00 0x00
0x00F50080: 0x30 0x00 0x00 0x01 0x00 0x00 0x00 0x00
0x00F50088: 0x20 0x00 0x00 0x00 0x00 0x00 0x00 0x01
0x00F50090: 0x00 0x00 0x00 0x07 0x20 0x00 0x00 0x00
0x00F50098: 0x20 0x00 0x00 0x00 0x30 0x00 0x20 0x01
0x00F500A0: 0x00 0x00 0x00 0x00 0x00 0x02 0x00 0x01
0x00F500A8: 0x00 0x00 0x00 0x00 0x30 0x01 0x20 0x01
0x00F500B0: 0x38 0x78 0x3F 0xE5 0x30 0x01 0xC0 0x01
0x00F500B8: 0x00 0x40 0x00 0x00 0x30 0x01 0x00 0x01
0x00F500C0: 0x03 0x82 0x20 0x73 0x30 0x00 0x00 0x01
0x00F500C8: 0x00 0x00 0x00 0x13 0x30 0x00 0x00 0x01
0x00F500D0: 0x00 0x00 0x00 0x09 0x20 0x00 0x00 0x00
0x00F500D8: 0x30 0x00 0xC0 0x01 0x00 0x00 0x00 0x01
0x00F500E0: 0x30 0x00 0xA0 0x01 0x00 0x00 0x01 0x01
0x00F500E8: 0x30 0x00 0xC0 0x01 0x00 0x00 0x10 0x00
0x00F500F0: 0x30 0x03 0x00 0x01 0x00 0x00 0x10 0x00
0x00F500F8: 0x20 0x00 0x00 0x00 0x20 0x00 0x00 0x00
0x00F50100:
Press any Key to Main Menu
  
```

X16248-030116

Figure 25: Option 5: Read Quad SPI Flash Output

Checklist and Debug Tips

- When [Customizing the AXI Quad SPI Core](#), verify **Use STARTUP Primitive External to the IP** selected for this demonstration (see [Figure 21](#)).
- Ensure [Set Up KCU105 Board](#) is completed before running the reference design demonstration. This will:
 - Ensure the board power is connected properly and power switch SW1 is in the ON position.
 - Master SPI configuration mode is properly set (Mode pins are set to M[2:0] = 001). See DIP switch SW15 settings shown in [Figure 3](#).
 - Both UART and JTAG cables are connected from the KCU105 evaluation board to the host computer See [Figure 3](#).

3. If you do not see the Reference Design Main Menu displayed in Tera Term ([Figure 13](#)) or if UART communication is not working between the host computer and the KCU105 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 26](#)). The COM number might be different on different computers. If the KCU105 System Controller Main Menu is displayed in Tera Term this is an indicator that the enhanced COM port was incorrectly selected instead of the standard COM port target.

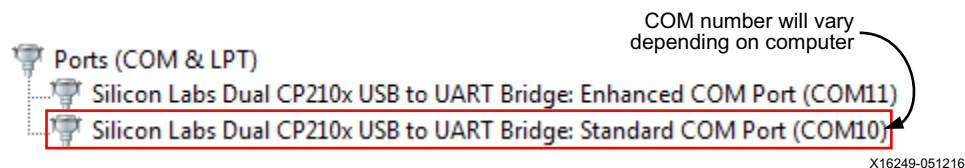


Figure 26: **Standard COM Port Driver**

- Confirm Tera Term is installed correctly (version 4.87 was used for reference design testing).
 - Confirm COM port settings in the Tera Term **Port Settings** tab match the values shown in [Figure 12](#).
4. If the FPGA does not seem to be configuring:
 - Confirm a valid bitstream image is loaded into SPI memory
 - Start FPGA programming by momentarily pressing the FPGA PROG pushbutton SW4 or cycle power using SW1 (See [Figure 3](#) for locations)
 5. If the batch files do not run correctly:
 - Ensure Vivado Design Suite 2016.1 or later is installed and that the path is setup correctly.
 6. The DONE LED does not light and the Master SPI configuration mode is properly set on DIP switch SW15:
 - Verify that the FPGA `golden.bin` can be programmed directly using the JTAG interface (J87)
 - Perform a readback to check the contents of the SPI memory programmed at different locations.
 - For user designs, ensure the `write_bitstream` constraints are implemented correctly for SPI 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, download `update.bin` again.
 - The DONE LED is lit, but the wrong pattern is loaded. Ensure that the right image was used when running the reference design. The pattern for the initial image (`golden.bin`) is LED0 blinking, and the pattern for the update image (`update.bin`) is LED1 blinking.

7. This reference design uses operations that are targeted for the Micron *N25Q256A11E* SPI 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.
8. If the Manufacturer ID (option 1) is not working, check the IDCODE via JTAG using the Xilinx Microprocessor Debugger (XMD):
 - a. Open XMD command prompt in the SDK IDE by selecting Xilinx Tools > XMD Console.
 - b. At the command line prompt XMD% enter these commands:

```
XMD% fpga -f ./source/Golden.bit
XMD% stop
XMD% rst
XMD% connect mb mdm
```

The resulting output will look like this:

```
JTAG chain configuration
-----
Device      ID Code      IR Length    Part Name
  1         13822093          6      xcku040

MicroBlaze Processor Configuration :
-----
Version.....9.5
Optimization.....Performance
Interconnect.....AXI-LE
MMU Type.....No_MMU
No of PC Breakpoints.....1
No of Read Addr/Data Watchpoints...0
No of Write Addr/Data Watchpoints..0
Instruction Cache Support.....on
Instruction Cache Base Address.....0x80000000
Instruction Cache High Address.....0xffffffff
Data Cache Support.....on
Data Cache Base Address.....0x80000000
Data Cache High Address.....0xffffffff
Exceptions Support.....off
FPU Support.....off
Hard Divider Support.....off
Hard Multiplier Support.....off
Barrel Shifter Support.....off
MSR clr/set Instruction Support....off
Compare Instruction Support.....off
Data Cache Write-back Support.....off
Fault Tolerance Support.....off
Stack Protection Support.....off

Connected to "mb" target. id = 0
Starting GDB server for "mb" target (id = 0) at TCP port no 1234

c. At the command line prompt XMD% enter these commands:

XMD% source ./source/xmd_idcode.tcl
XMD% idcode
```

The resulting output will look like this:

```
Mfg ID:      44A0006C:    00000020
Memory Type:44A0006C:    000000BB
Memory Size:44A0006C:    00000019
Device ID 1:44A0006C:    00000010
Device ID 2:44A0006C:    00000000
```

9. An alternative to XMD is to use the JTAG-to-AXI IP in the reference design. 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 SPI flash memory use the following Tcl command sequence:
 - a. Start Vivado hardware manager. In the Tcl console type:

```
Connect_hw_server
Open_hw_target -jtag_mode 1
Set current_hw_jtag [get_property hw_jtag [lindex [get_hw_targets] 0]]
source jtag_to_qspi.tcl
```

Conclusion

The ease of accessing the SPI flash memory from the UltraScale FPGA post-configuration has been demonstrated. Accessing SPI 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 5](#) shows the reference design matrix.

Table 5: Reference Design Matrix

Parameter	Description
General	
Developer names	Steven Howell, Shashikant Jadhav, and Stephanie Tapp
Target device	Kintex UltraScale FPGA (XCKU040-2FFVA1156E)
Source code provided	Yes
Source code format	VHDL (Verilog next phase)
Design uses code and IP cores from existing Xilinx application notes and reference designs or third party	Vivado Design Suite
Simulation	
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
Implementation	
Synthesis software tools/versions used	Vivado synthesis
Implementation software tools/versions used	Vivado Design Suite and SDK 2016.1
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	KCU105 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. *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* ([DS892](#))
6. *Micron Serial NOR Flash Memory N25Q256A11E Data Sheet* (www.micron.com)
7. *SPI Configuration and Flash Programming in UltraScale FPGAs* ([XAPP1233](#))
8. *KCU105 Evaluation Kit web page* (www.xilinx.com/kcu105)
9. *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#))
10. *Vivado Design Suite Tcl Command Reference Guide* ([UG835](#))
11. *KCU105 Evaluation Kit Quick Start Guide* ([XTP391](#))
12. *KCU105 Board User Guide* ([UG917](#))
13. *KCU105 Software Install and Board Setup Tutorial* ([XTP352](#))
14. *AXI Quad SPI Product Guide* ([PG153](#))
15. *MultiBoot and Fallback with SPI Flash in UltraScale FPGAs Application Note* ([XAPP1257](#))

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/02/2016	1.1	Updated document content and reference design files to support Vivado Design Suite 2016.1. Changed Vivado Design Suite version references from 2015.4 to 2016.1. Removed redundant AXI timer block in Figure 2 and Figure 19 (same as AXI Interconnect block). Updated values in Table 1 . Updated the IMPORTANT note on page 7 . Updated Figure 6 . Updated tip on page 13 . Updated Figure 21 . Updated STARTUPE3 parameters listing on page 23 and constraints on page 24 . Revised Checklist and Debug Tips number 5 and number 6.
04/15/2016	1.0.1	Corrected typographical errors and other minor edits.
04/05/2016	1.0	Initial Xilinx release.

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>.

© Copyright 2016 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.