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

XAPP1280 (v2.1.3) January 23, 2026

Summary

Serial Peripheral Interface NOR flash memory (referred to as SPI Flash memory) is a popular AMD 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 an AMD Kintex™ UltraScale™ FPGA and SPI flash memory provided by a KCU105 evaluation board. AMD Vivado™ Design Suite 2022.2 is used as the development environment.

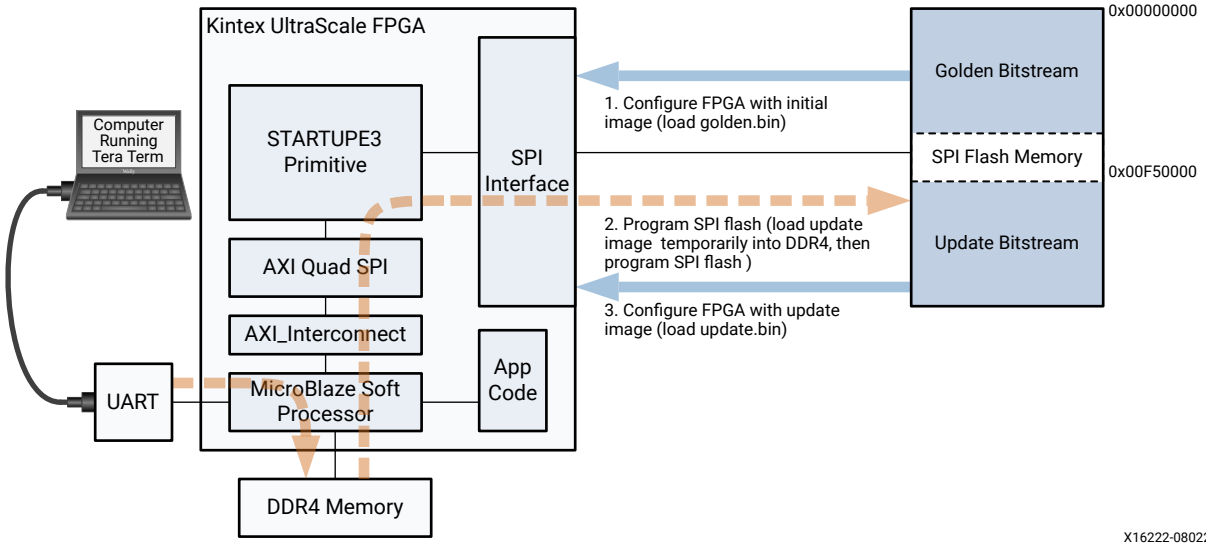
Download the reference design files for this application note from the AMD website. For detailed information about the design files, see Reference Design.

Note: This application note does not apply to AMD Spartan™ UltraScale+™ FPGAs.

Introduction

The reference design in this application note uses an AMD 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. The following figure shows operation of the post-configuration reference design.

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. The preceding figure 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:

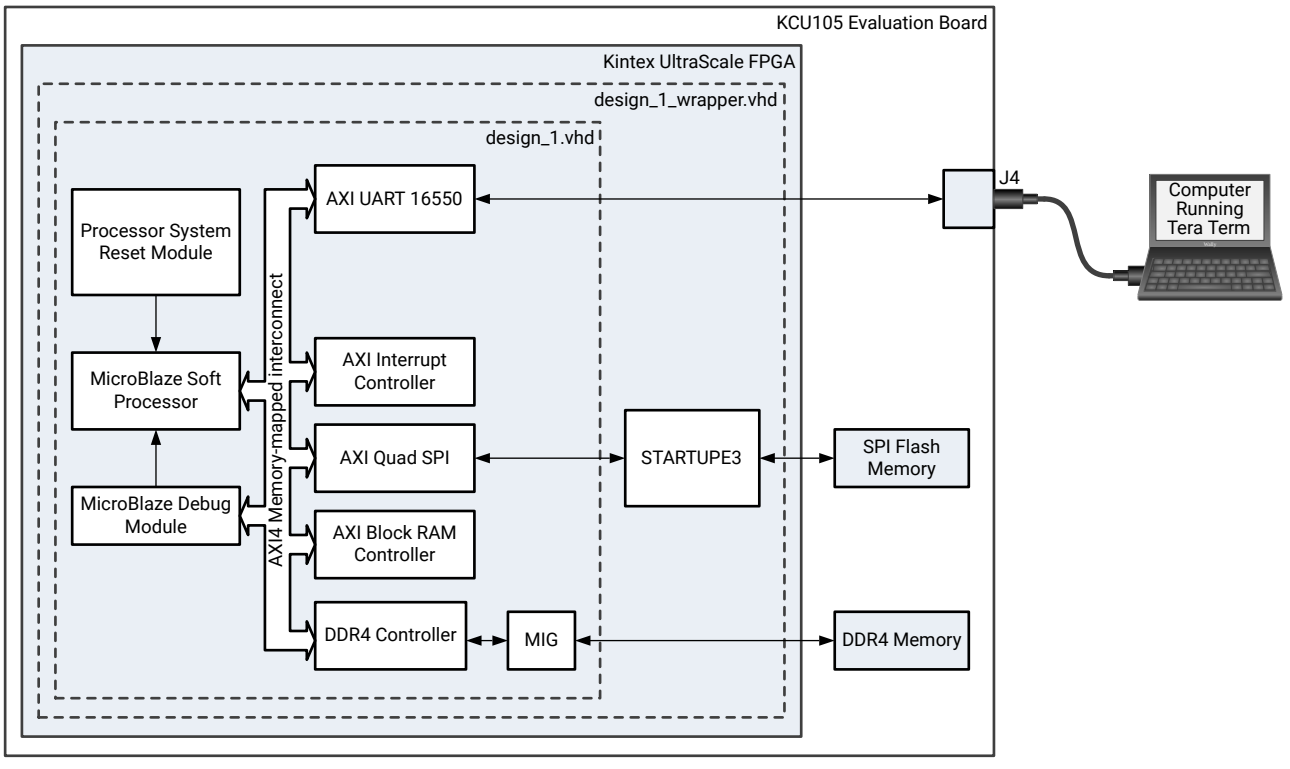
- The UltraScale FPGA SPI configuration mode. See *UltraScale Architecture Configuration User Guide* (UG570)
- Vivado Design Suite. See *Vivado Design Suite Quick Reference Guide* (UG975)

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.

AMD 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 the following figure.

Figure 2: **Post-Configuration Reference Design System Block Diagram**



Reference Design Files

Project script files:

- `run_kcu105 (.bat/.tcl)` (creates reference design project)

- `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` (AMD Vitis™ source code)
- `lscript.ld` (linker script)
- `quad_spi_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 Soft Processor (microblaze_0)	11.0 (Rev. 10)	100 MHz		N/A	N/A
DDR4 SDRAM MIG (ddr4_0)	2.2 (Rev. 17)	300 MHz	0x8000_0000	1 GB	0xBFFF_FFFF
AXI Quad SPI (axi_quad_spi_0)	3.2 (Rev. 26)	100 MHz	0x44A0_0000	64 KB	0x44A0_FFFF
AXI Block RAM Controller (axi_bram_ctrl_0)	4.1 (Rev. 7)	100 MHz	0xC000_0000	1 MB	0xC00F_FFFF
AXI Interrupt Controller (microblaze_0_axi_intc)	4.1 (Rev. 17)	100 MHz	0x4120_0000	64 KB	0x4120_FFFF
AXI UART 16550 (axi_uart16550_0)	2.0 (Rev. 29)	100 MHz	0x44A1_0000	64 KB	0x44A1_FFFF
AXI Interconnect (axi_mem_intercon)	2.1 (Rev. 28)	100 MHz		N/A	N/A
Processor System Reset Module (proc_sys_reset)	5.0 (Rev. 13)	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: Reference Design Post-Configuration Access Flow):

- 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: Reference Design Post-Configuration Access Flow)
- Program SPI Flash Memory with update.bin (Corresponds to step 2 in Figure 1: Reference Design Post-Configuration Access Flow)
- Configure FPGA with update.bin (Corresponds to step 3 in Figure 1: Reference Design Post-Configuration Access Flow)

Required Hardware and Software

- KCU105 evaluation board which includes:
  - AMD Kintex™ UltraScale™ XCKU040-2FFVA1156E FPGA (U1)
  - Micron MT25QU256ABA 256 Mb Quad SPI flash Memory (U35)

    ***Note*:** The Micron SPI flash memory MT25QU256ABA is compatible with the Micron SPI flash memory N25Q256A11.
  - Four Micron MT40A256M16LY-062E DDR4 512 MB Memories (U60-U63)
- Power supply: 100–240 VAC input, 12 VDC 5.0 A 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. (Example flow in application note is run on Windows. See `readme.txt` for Linux `run_kcu105.sh`).
- Vivado Design Suite 2022.2
- Vitis unified software platform 2022.2
- Tera Term terminal emulator program (version 4.105 was used for reference design testing)
- Silicon Labs Dual CP210x USB UART Drivers
- reference design files

Set Up Host Computer

If not already installed:

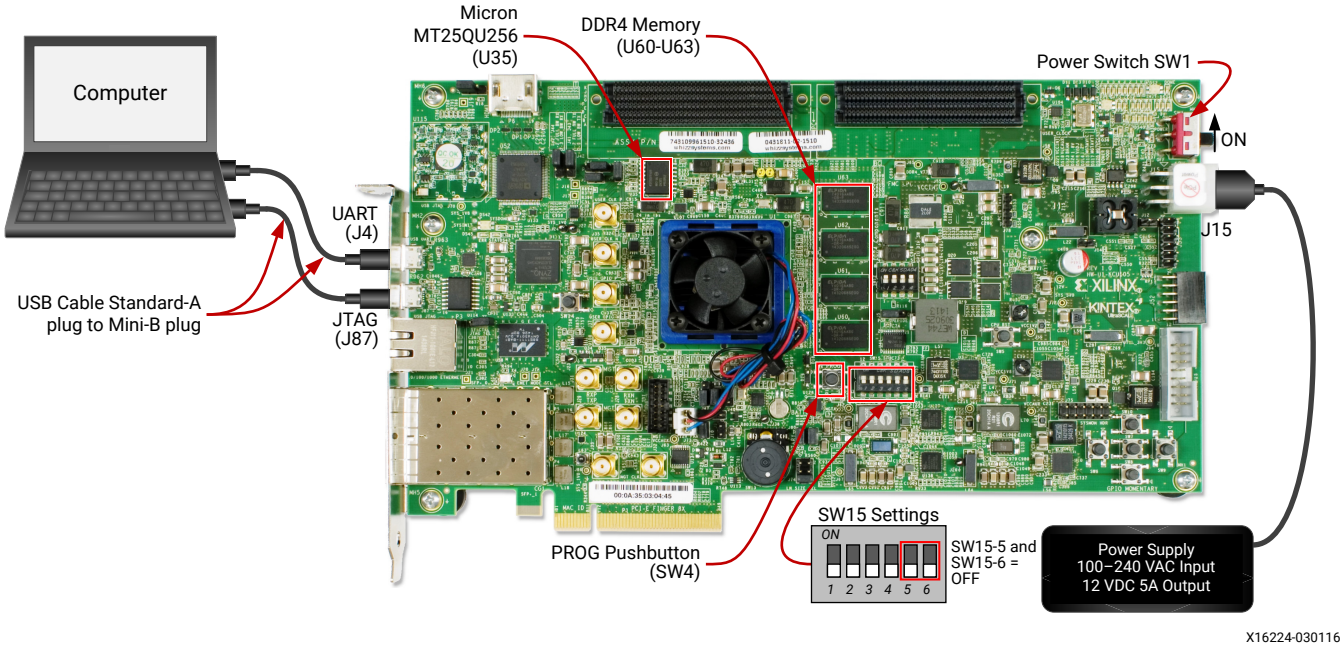
1. Install AMD Vivado™ Design Suite version 2022.2.
2. Download and install Tera Term (version 4.105 was used for reference design testing). Follow the procedure in *Tera Term Terminal Emulator Installation Guide* (UG1036).
3. Download and install the UART drivers. Follow the instructions in *Silicon Labs CP210x USB-to-UART Installation Guide* (UG1033).

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

Set Up KCU105 Board

Referring to the following figure:

Figure 3: **Connection Diagram for Preliminary Setup**

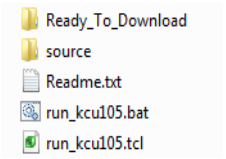


1. Place switch SW1 to the OFF position and connect the power supply to J15.
2. Connect the USB cables between the computer and the UART connector (J4) and JTAG connector (J87) on the KCU105 board and the computer.
3. Set the FPGA M2 pin to 0 by setting DIP switch SW15-6, to OFF as shown in the preceding figure. This selects the FPGA SPI configuration mode (M[2:0] = 001) because the FPGA M[1:0] pins are hard-wired to 01. Ensure DIP switch SW15-5 is in OFF position to disable the SYSCTLR_ENABLE.

Download Reference Design Files

Follow these steps to download the reference design files.

1. Download and unzip the reference design files to `c:\`. The following figure shows the resulting file structure that is created under the `c:\xapp1280_quad_spi` directory.



X16225-041216

2. Do one of the following:
   - Go to Generate Reference Design Project (do this to perform the steps to create a Vitis tools 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).

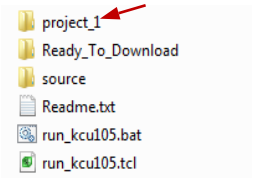
Generate Reference Design Project

Follow these steps to generate a reference design project.

1. Go to `c:\xapp1280_quad_spi`. Double-click `run_kcu105.bat` to run the batch file.

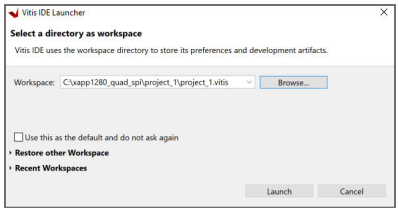
   This batch file generates the Vivado IPI hardware project (`project_1.xpr`) and exports the created hardware to Vitis (`project_1.vitis`) under the `project_1` directory. The following figure shows the project directory addition.

> **TIP:** *The project build takes approximately 90 minutes depending on host computer.*

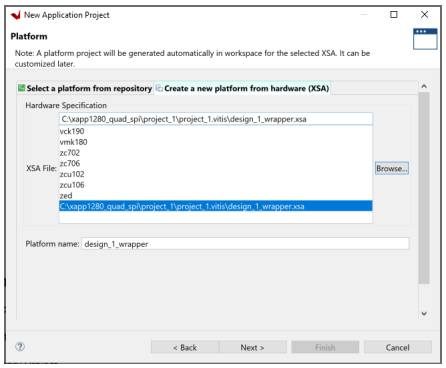


X16226-041216

2. Run the Vitis 2022.2 Integrated Design Environment (IDE) launcher from the AMD Design Tools. Ensure the workspace is selected as shown in the following figure and click **Launch**.

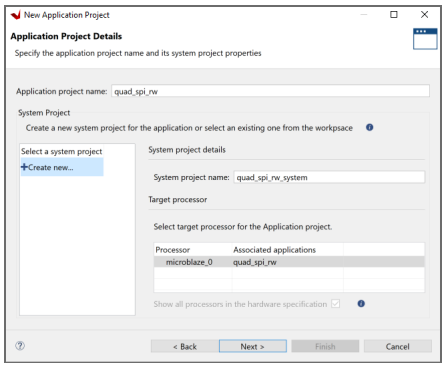


3. Create an application for the Quad SPI flash memory read and write access:
   a. Select **Create Application Project**.
   b. In the Create a New Application Project window, click **Next**.
   c. Select the Create a new platform from hardware (XSA) tab and browse to select the `design_1_wrapper.xsa` as shown in the following figure.



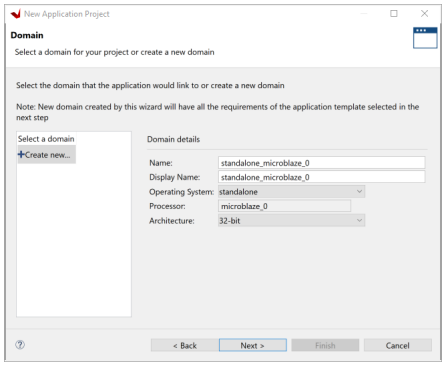
d. Click **Next** to open the applications project details window.

e. Under Applications project name enter quad_spi_rw as shown in the following figure.

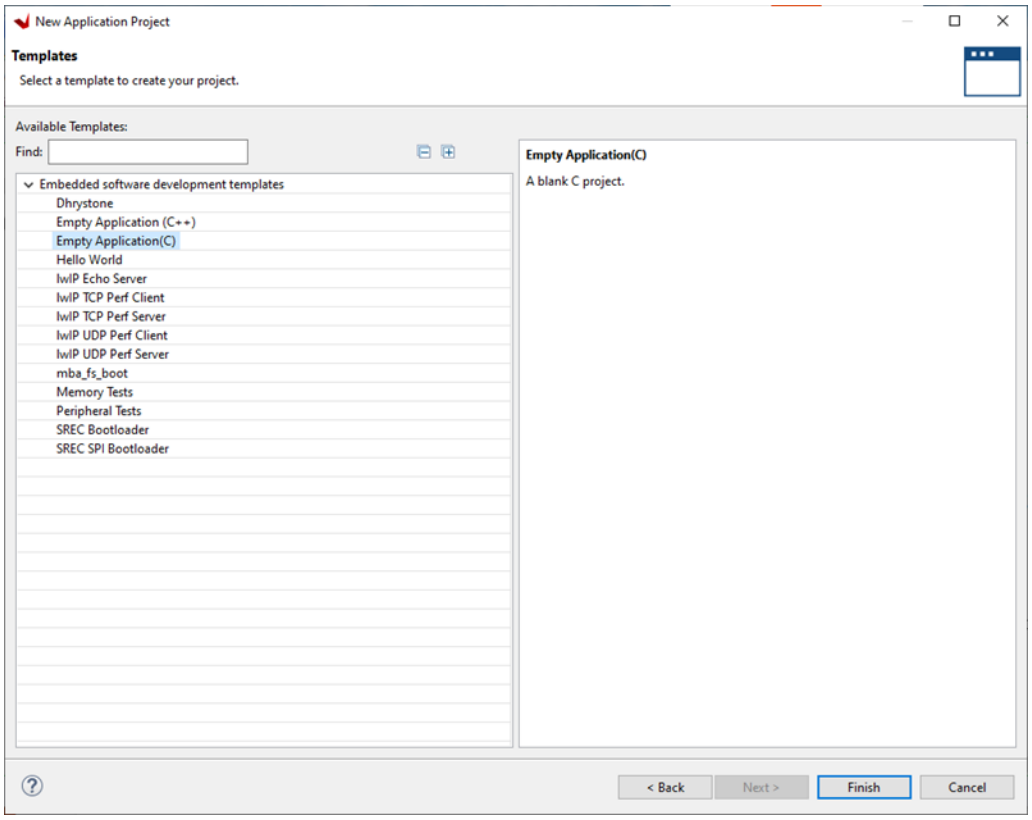


f. Click **Next** to select the domain for your project.

g. Click **Next** on the Domain window with settings as shown in the following figure.

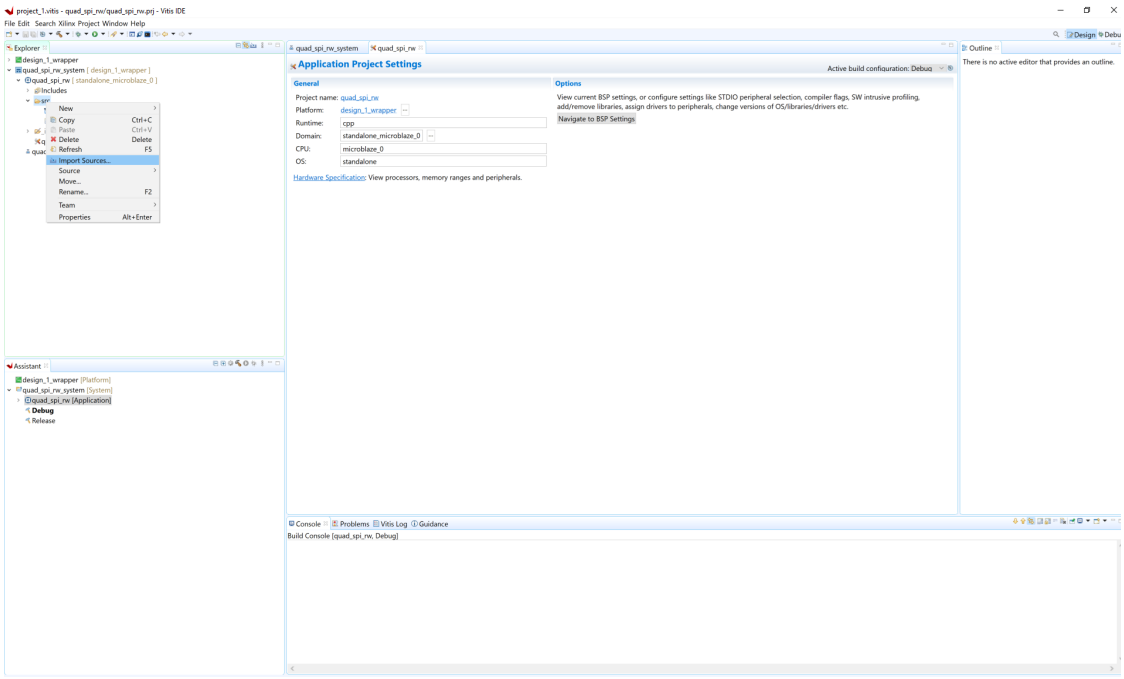


h. Select **Empty Application (C)** as shown in the following figure.

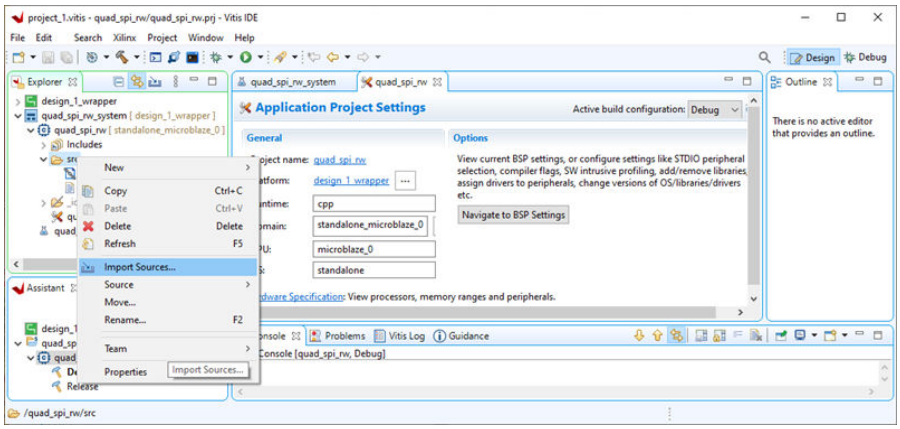


i. Click **Finish**.

j. Right-click **src** and select **Import Sources** as shown in the following figure.



k. Browse to the `C:\xapp1280_quad_spi\source` directory, select folder. Then select the `flash_qspi_rw.c` and `lscript.ld`.



l. Click **Finish** to add both files to the project and then click **Yes** when asked to Overwrite the `lscript.ld` in the `quad_spi_rw\src` folder.

m. Under the Vitis Explorer navigation pane right-click **quad_spi_fw** and select **Project** → **Build Project**.

4. (Optional Step) Review the MicroBlaze code: In the Vitis 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. 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 six functions: `SpiFlashWriteEnable()`, `SpiFlashWrite()`, `SpiFlashRead()`, `SpiFlashBulkErase()`, `SpiFlashGetStatus()`, and `SpiFlashQuadEnable()`.

Generate Programming Files

Follow these steps to generate programming files.

1. Close the Vitis IDE.
2. Go to `C:\xapp1280_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.*

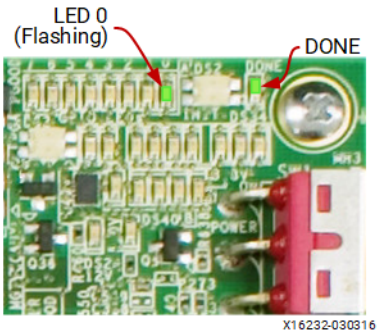
Configure FPGA with golden.bin

This part of the procedure corresponds to step 1 in Figure 1: Reference Design Post-Configuration Access Flow.

1. Turn on power to the KCU105 board by placing switch SW1 to the ON position as shown in Figure 3: Connection Diagram for Preliminary Setup. 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:\xapp1280_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 as shown in the following figure.



   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.*

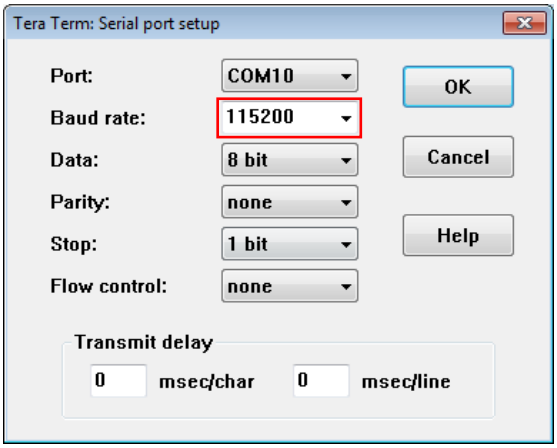
   Refer to *UltraScale Architecture Configuration User Guide* (UG570) and *MultiBoot and Fallback with SPI Flash in UltraScale FPGAs Application Note* (XAPP1257) for details on the MultiBoot and fallback features.

Program SPI Flash Memory with update.bin

This part of the procedure corresponds to step 2 in Figure 1: Reference Design Post-Configuration Access Flow.

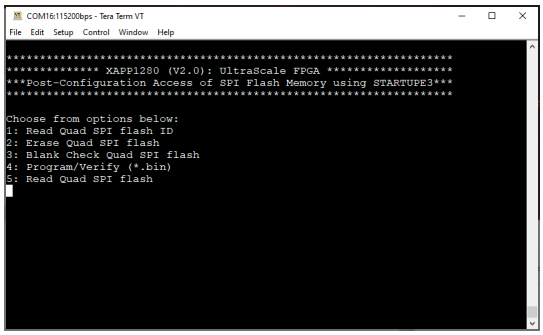
After the FPGA is running the `golden.bin` image, `update.bin` can be downloaded from the host computer and written into SPI flash at address `0x00F50000`, and then be used to reconfigure the FPGA.

1. Confirm the host computer and KCU105 evaluation board are connected as shown in Figure 3: Connection Diagram for Preliminary Setup.
2. Launch and set up Tera Term using the values shown in the following figure. Ensure that the Standard COM Port is selected (Refer to the Checklist and Debug Tips step 3 for details).



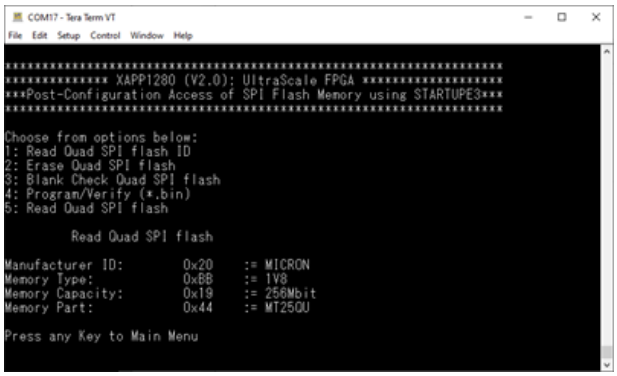
X16234-032316

3. 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 the following figure.



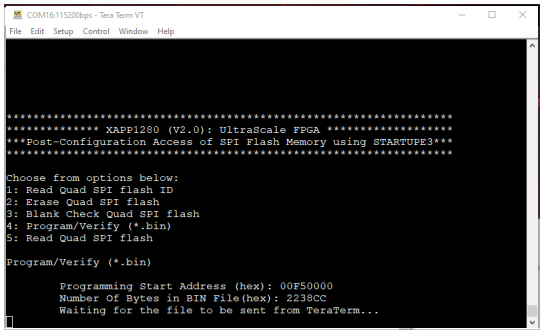
4. In the Tera Term window, enter 1 to display the SPI flash manufacturer information and ensure proper board setup as shown in the following figure.

   Note: The Micron SPI flash memory MT25QU256ABA is compatible with the Micron SPI flash memory N25Q256A11.

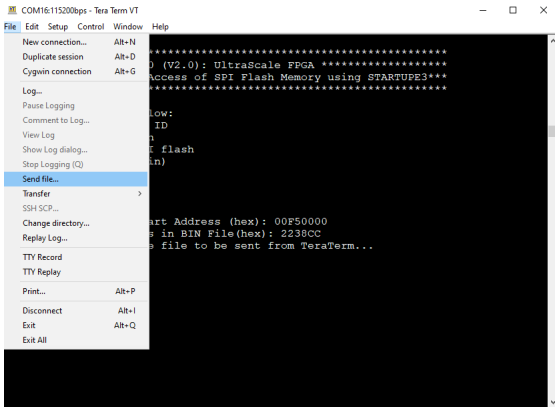


TIP: *The operations 3: Blank Check Quad SPI flash and 5: Read Quad SPI flash can also be performed. Both operations require a byte count to be input and will display results to the nearest page boundary (256 bytes). To perform a Bulk Erase, select* ***2: Erase Quad SPI flash*** *and at the next prompt,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

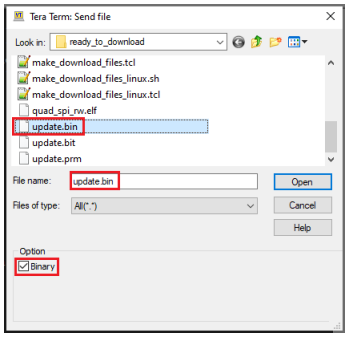
5. Enter 4 to program and verify the SPI flash memory with the `update.bin` image at offset `00F50000` and enter `2238CC` number of bytes when prompted.



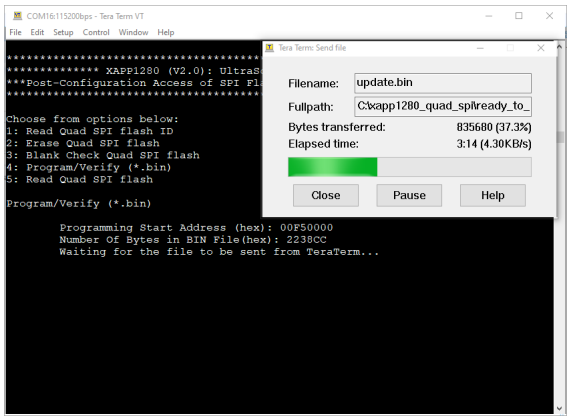
6. In the menu bar, click **File** → **Send file**.



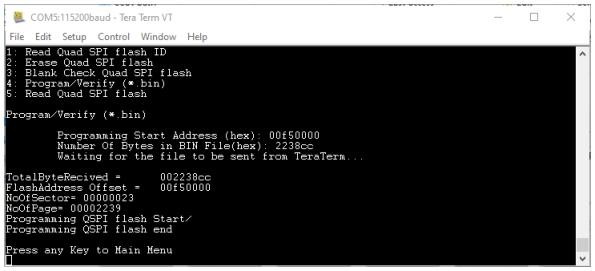
7. Choose `update.bin` under `C:\xapp1280_quad_spi\ready_to_download`. Select **Binary** and click **Open**.



Tera Term will display the progress as shown in the following figure.



When the file has been programmed a summary will be displayed as shown in the following figure.

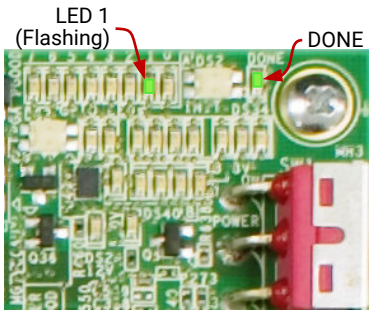


Configure FPGA with update.bin

This part of the procedure corresponds to step 3 in Figure 1: Reference Design Post-Configuration Access Flow.

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 the following figure (replacing the original `golden.bin` LED 0 pattern).



X16239-030116

***Note*:** 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.

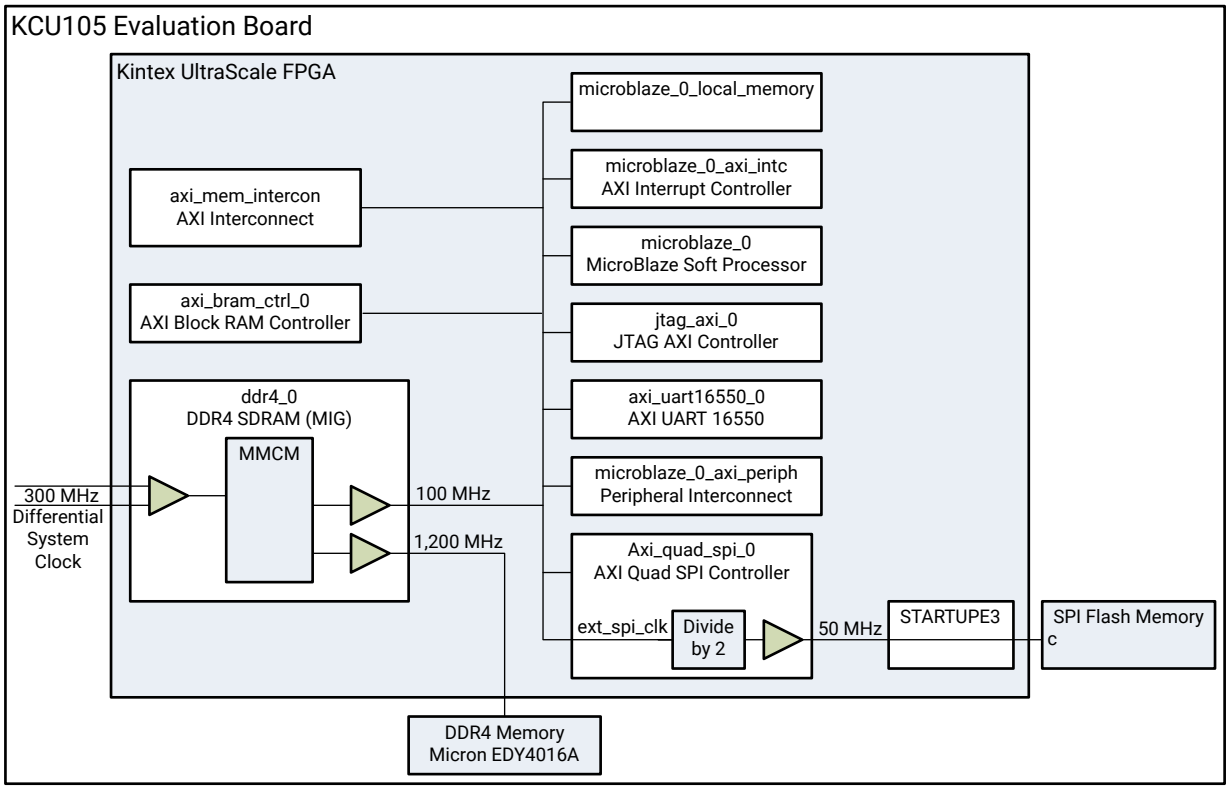
Hardware System Details

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 the following figure.

Figure 4: **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.

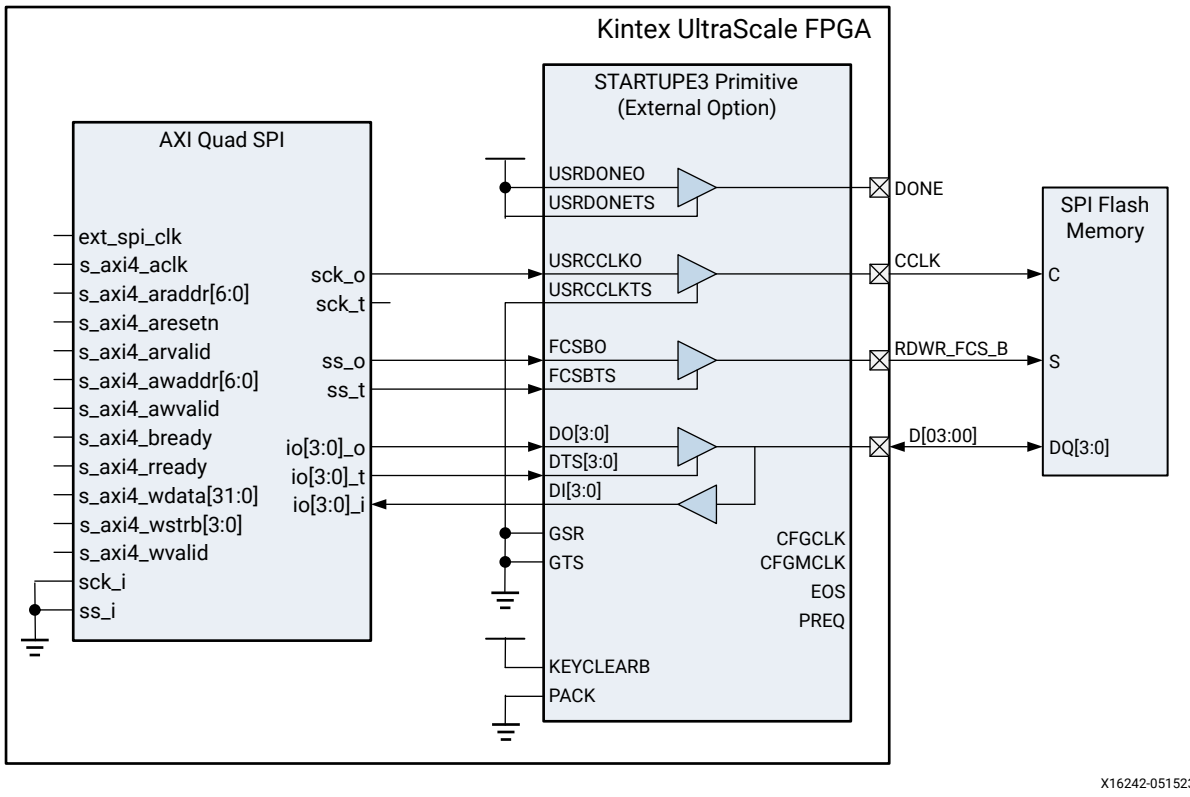
In the case of this reference design, the 100 MHz clock connected to the `ext_spi_clk` pin is divided by 2 to provide a 50 MHz clock to the SPI flash memory.

IMPORTANT! *This reference design is targeted for use with the Micron MT25QU256ABA 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. The Micron SPI flash memory MT25QU256ABA is compatible with the Micron SPI flash memory N25Q256A11.*

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 the following figure.

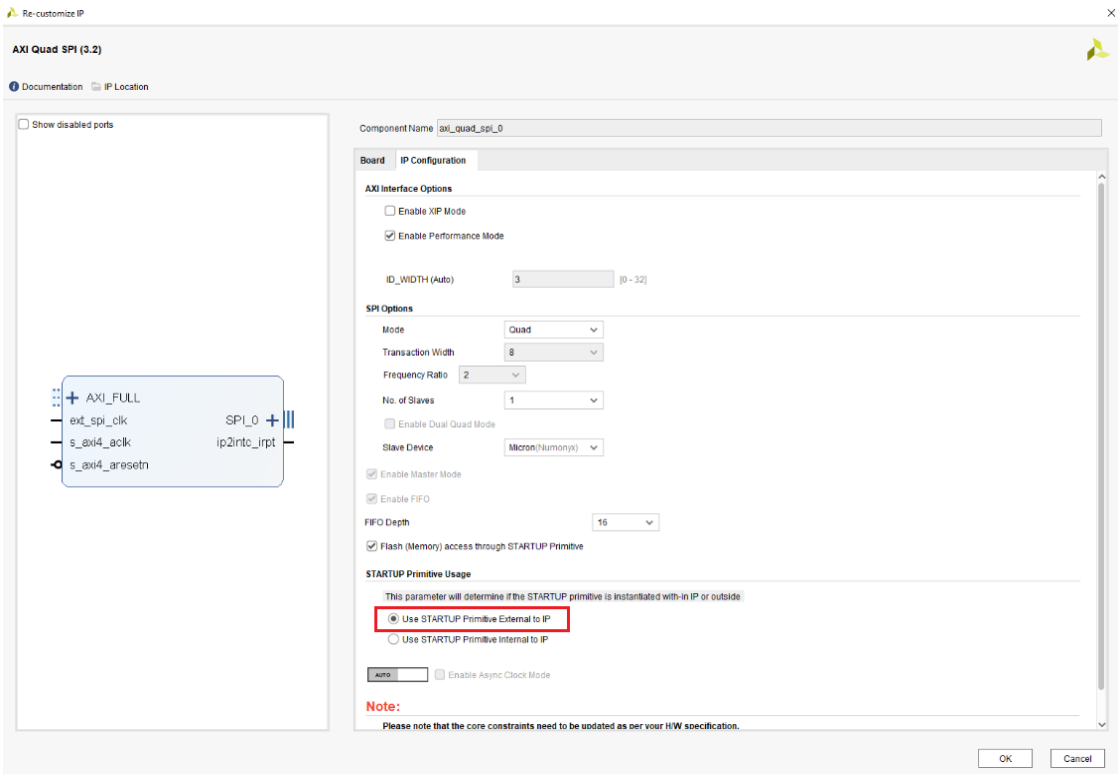
Figure 5: **AXI Quad SPI Core to STARTUPE3 Connectivity**



Customizing the AXI Quad SPI Core

The AXI Quad SPI core is setup specifically for the AMD Kintex™ UltraScale™ XCKU040-2FFVA1156E FPGA, and the Micron MT25QU256ABA SPI flash memory on the KCU105 evaluation board. The IP Integrator window shown in the following figure is used for customization.

Figure 6: **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 AXI4 interface instead of the default AXI4-Lite interface. See *AXI Quad SPI LogiCORE IP Product Guide* (PG153) 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 AMD 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. The following table 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: **STARTUPE3SPI Interface Dedicated Pins**

Logic Signal	STARTUPE3 Port	Dedicated Pin Name	FPGA (U1) Pin Location
spi_0_ss_o_0(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:

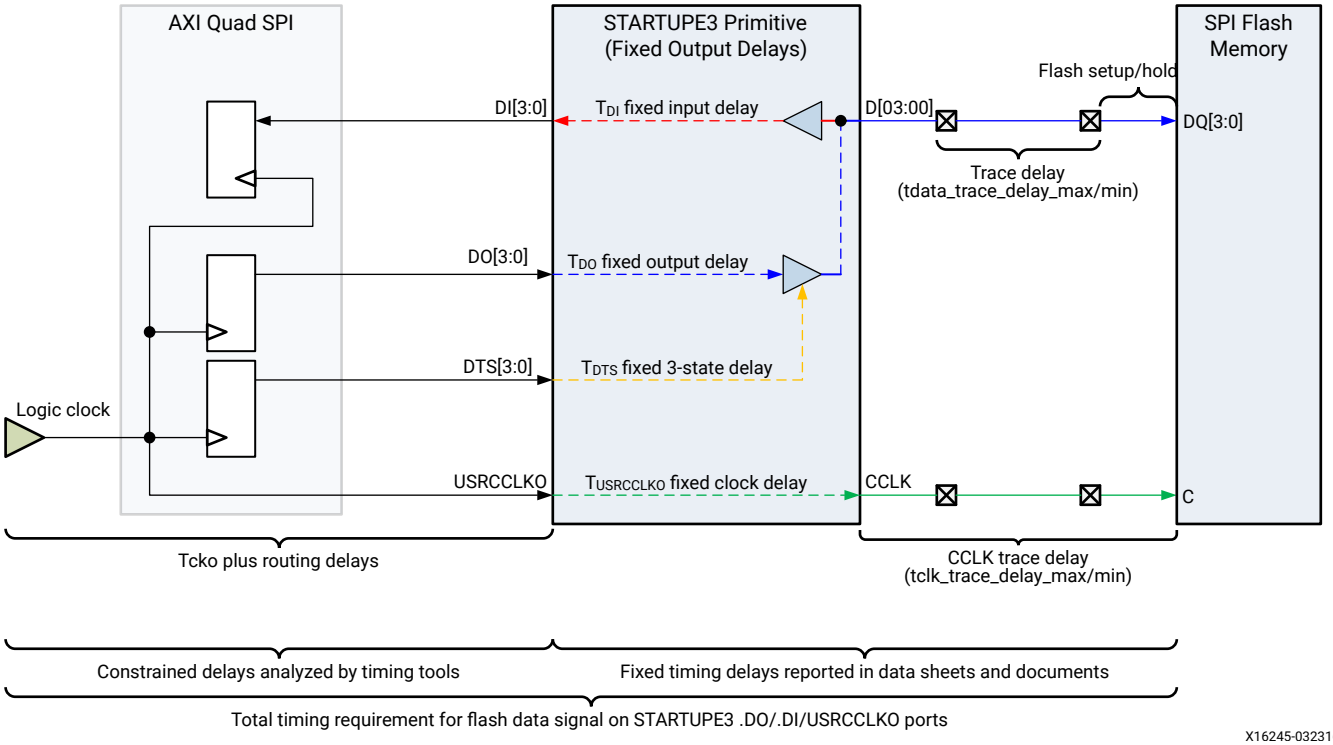
```
-- STARTUPE3: STARTUP Block
-- Kintex UltraScale+
-- Xilinx HDL Language Template, version 2022.2
STARTUPE3_inst : STARTUPE3
 -----------------------
 generic map
 (
 PROG_USR => "FALSE", -- Activate program event security feature. Requires
encrypted bitstreams.
 SIM_CCLK_FREQ => 0.0 -- Set the Configuration 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 => '0', -- 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. The following figure shows the timing paths that needs to be considered.

Figure 7: **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) and the *Micron Serial NOR Flash Memory MT25QU256ABA Data Sheet* (www.micron.com). Sample parameters are provided in the following tables 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			
TUSRCCLKO	STARTUPE3 USRCCLKO input port to CCLK pin output delay	1.00/6.70	ns, Min/Max
TDO	DO[3:0] ports to D03-D00 pins output delay	1.00/7.70	ns, Min/Max
TDTS	DTS[3:0] ports to D03-D00 pins 3-state delays	1.00/8.30	ns, Min/Max
TFCSBO	FCSBO port to FCS_B pin output delay	1.00/8.00	ns, Min/Max
TFCSBTS	FCSBTS port to FCS_B pin 3-state delay	1.00/8.00	ns, Min/Max
TDI	D03-D00 pins to DI[3:0] ports input delay	0.5/3.1	ns, Min/Max

Table 4: **Micron MT25QU256ABA 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)	6 ns
TDVCH (min)	Tsu	Data in setup time	1.75 ns
TCHDX (min)	Th	Data in hold time	2 ns

Example constraints for the KCU105 board are shown below. The timing delays for the board, and the Micron MT25QU256ABA flash memory are obtained from measurement and the flash data sheet. The Micron SPI flash memory MT25QU256ABA is compatible with the Micron SPI flash memory N25Q256A11.

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

##### MT25QU256 SPI device parameters set tco_min 1 # MIN Tco
set tco_max 6 # MAX Tco
set tsu 1.75 # SPI setup time requirement set th 2 # 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[*] {*STARTUP*_inst/DTS[*]}] 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) and *SPI Configuration and Flash Programming in UltraScale FPGAs* (XAPP1233).

```
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

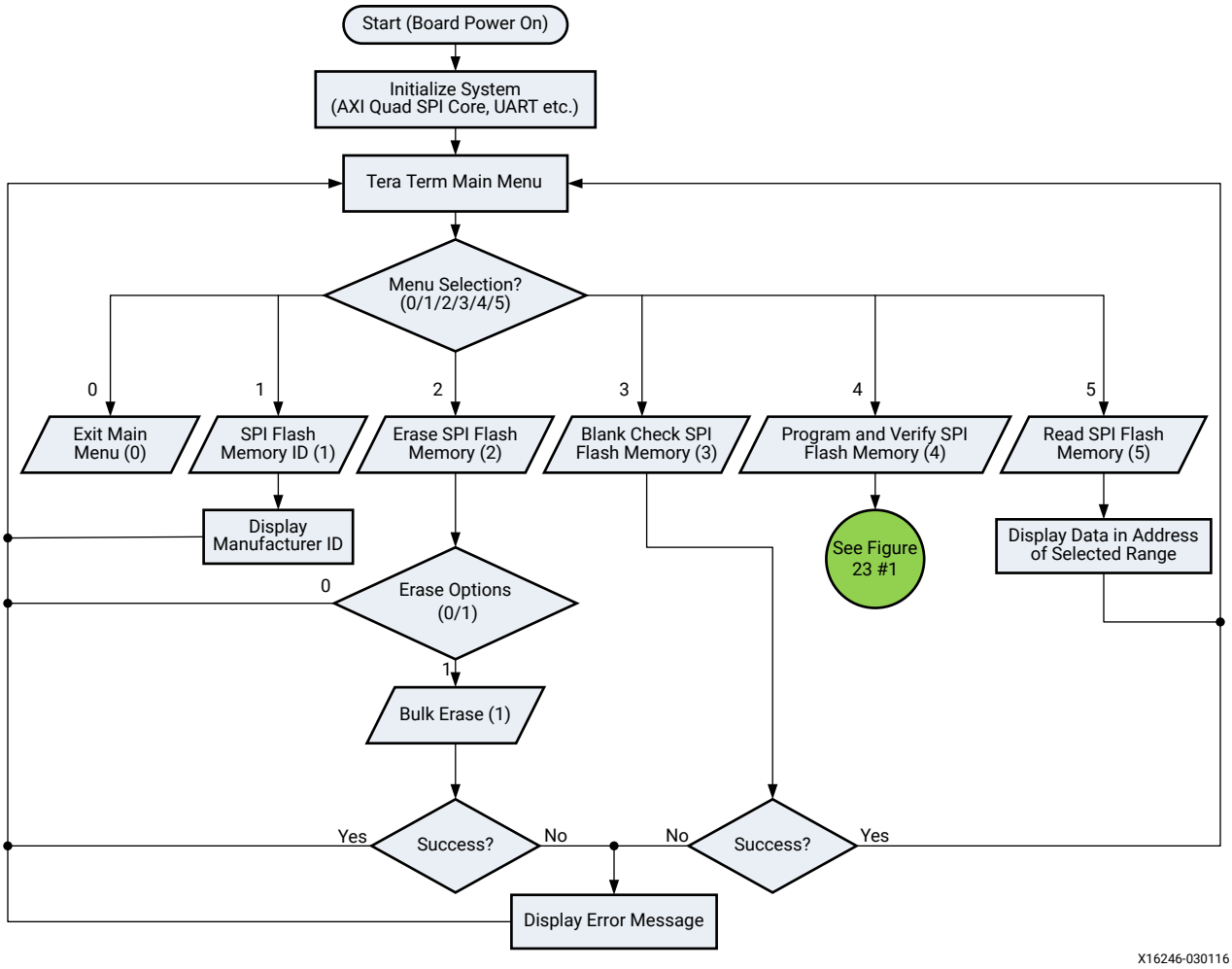
Software running on the MicroBlaze™ processor drives the AXI Quad SPI core to read and write to the SPI flash memory using the hardware connections to the STARTUPE3 primitive. The `flash_qspi_rw.c` file included with this reference design provides example low-level software read and write operations. 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 and MT25Q256U Data Sheets (www.micron.com). This memory is located on the KCU105 evaluation board at location U35 (see Figure 3: Connection Diagram for Preliminary Setup).

The functions in `flash_qspi_rw.care` 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 MT25QU256ABA SPI flash memory used on the KCU105 evaluation board and supports the set of instructions listed in Micron Serial NOR Flash Memory N25Q256A11E and MT25Q256U Data Sheets (www.micron.com).

Flash Command Set

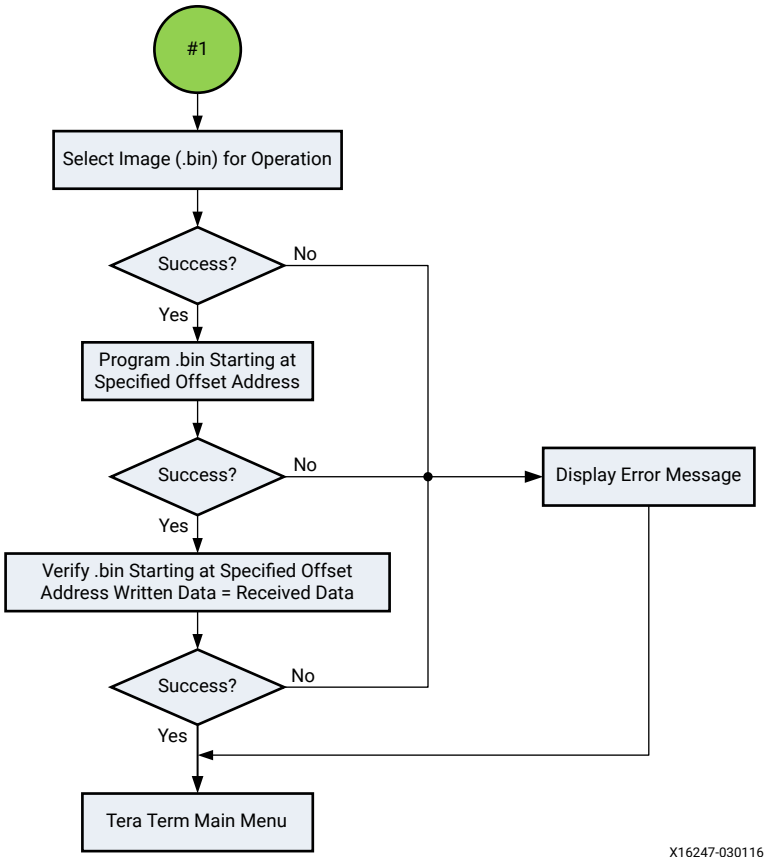
There are a number of commands that provide access to the flash array or perform other operations on the SPI flash memory. All SPI transactions in master mode depend upon commands supported by a slave device connected to the AXI QUAD SPI core. Refer to the respective SPI flash memory data sheet for the supported commands. The Micron MT25QU256ABASPI flash memory commands are described in Micron Serial NOR Flash Memory N25Q256A11E and MT25Q256U Data Sheets (www.micron.com). This reference design provides user access to a subset of commands by way of the menu shown in Program SPI Flash Memory with update.bin. The commands implemented for the SPI flash on the KCU105 are shown in the flow diagrams in the following figures.

Figure 8: **Flash Command Set Flow Diagram**



The flowchart for Program/Verify (Program SPI Flash Memory with update.bin Tera Term main menu option 4).

Figure 9: **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.

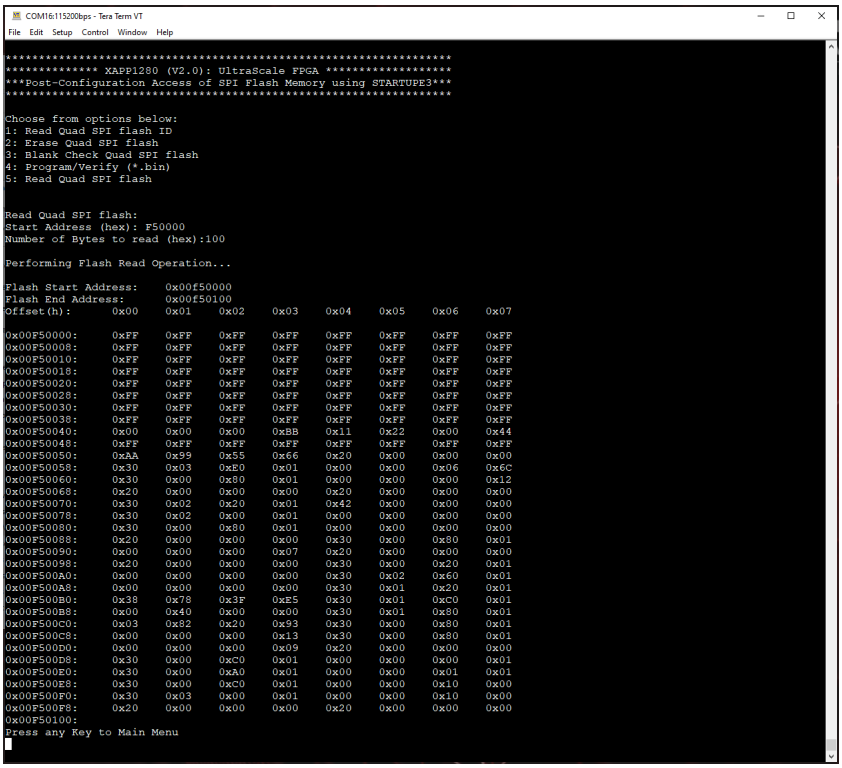
Read Data Command Sequence

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. Deassert 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 the preceding figure). 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.

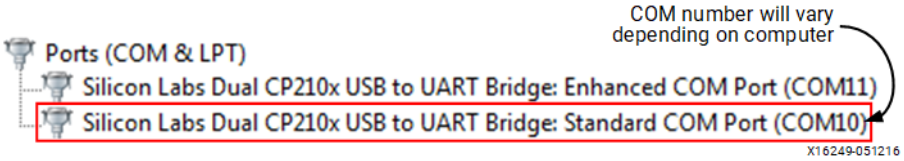
Program SPI Flash Memory with update.bin showcases option 4 in step 5, however, the Tera Term main menu also has individual operation options. When the individual operation option is selected (option 1, 2, 3, or 5) then the desired start address must be supplied along with either the stop address or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

Figure 10: **Option 5: Read Quad SPI Flash Output**



Checklist and Debug Tips

1. When Customizing the AXI Quad SPI Core, verify Use STARTUP Primitive External to the IP selected for this demonstration (see Figure 6: AXI Quad SPI Core Settings).
2. 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: Connection Diagram for Preliminary Setup.
   - Both UART and JTAG cables are connected from the KCU105 evaluation board to the host computer See Figure 3: Connection Diagram for Preliminary Setup.
3. If you do not see the Reference Design Main Menu displayed in Tera Term 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 (see the following figure). 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.



- Confirm Tera Term is installed correctly (version 4.105 was used for reference design testing).
- Confirm COM port settings in the Tera Term Port Settings tab match the values shown in step 2 of Program SPI Flash Memory with update.bin.

4. If the FPGA is not 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: Connection Diagram for Preliminary Setup for locations)
5. If the batch files do not run correctly:
   - Ensure AMD Vivado™ Design Suite 2022.2 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:
   a. Verify that the FPGA golden.bin can be programmed directly using the JTAG interface (J87).
   b. Perform a readback to check the contents of the SPI memory programmed at different locations.
   c. For user designs, ensure the write_bitstream constraints are implemented correctly for SPI mode as described in Hardware System Details.
   d. 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 MT25QU256ABA 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 Software Command-Line Tool (XSCT):
   a. Open XSCT command prompt by selecting **Xilinx Design Tools → Xilinx Software Command Line Tool 2022.2**.
   b. At the command line prompt XSCT% enter these commands:

```
XSCT% enter these commands:

xsct% connect
xsct% fpga -f c:/xapp1280_quad_spi/ready_to_download/golden.bit xsct%
ta 5
xsct% source xapp1280_quad_spi/source/xmd_idcode.tcl xsct% 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
```

9. An alternative to XSCT 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.
   b. 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
```

Related Information

Program SPI Flash Memory with update.bin

Conclusion

The ease of accessing the SPI flash memory from the AMD 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 AMD website. The following table shows the reference design matrix.

Table 5: **Reference Design Matrix**

Parameter	Description
General	
Developer names	Advanced Micro Devices, Inc.
Target device	AMD Kintex™ UltraScale™ FPGA (XCKU040-2FFVA1156E)
Source code provided	Yes
Source code format	VHDL
Design uses code and IP cores from existing application notes and reference designs or third party	AMD 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

Table 5: **Reference Design Matrix** *(cont'd)*

Parameter	Description
SPICE/IBIS simulations	N/A
Implementation	
Synthesis software tools/versions used	Vivado synthesis
Implementation software tools/versions used	Vivado Design Suite and AMD Vitis™ IDE 2022.2
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	KCU105 evaluation board

References

These documents provide supplemental material useful with this guide:

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 and MT25Q256U Data Sheets (www.micron.com)
7. *SPI Configuration and Flash Programming in UltraScale FPGAs* (XAPP1233)
8. KCU105 Evaluation Kit web page
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 LogiCORE IP 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.

Section	Revision Summary
01/23/2026 Version 2.1.3	
General updates	Editorial updates only. No technical content updates.

Section	Revision Summary
08/29/2024 Version 2.1.2	
General updates	Editorial updates only. No technical content updates.
08/27/2024 Version 2.1.1	
General updates	Editorial updates only. No technical content updates.
05/17/2023 Version 2.1	
Hardware System Details	Updated STARTUPE3 PACK port signal connection to GND in Figure 5: AXI Quad SPI Core to STARTUPE3 Connectivity.
05/10/2023 Version 2.0	
N/A	• Updated design version from AMD Vivado™ Design Suite and SDK IDE version 2016.1 to Vivado Design Suite and AMD Vitis™ IDE 2022.2. • Addressed path segmentation critical warning. • Updated design for MT25QU256 compatible with N25Q256A11E and updated DDR4 to MT40A256M16LY-062E.
06/02/2016 Version 1.1	
N/A	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.
System Overview	• Removed redundant AXI timer block in Figure 2: Post-Configuration Reference Design System Block Diagram. • Updated values in Table 1: IP Core Addresses.
Hardware System Details	Removed redundant AXI timer block in Figure 7: STARTUPE3 Timing Delays (same as AXI Interconnect block).
Generate Reference Design Project	Updated the IMPORTANT note on page 8. Updated Figure 6. Updated tip on page 16.
Hardware System Details	Updated STARTUPE3 parameters listing on page 28 and constraints on page 29.
Software System Details	Updated Figure 9: Menu Option 4 (Program/Verify) Flow Diagram.
Checklist and Debug Tips	Revised Checklist and Debug Tips number 5 and number 6.
04/15/2016 Version 1.0.1	
N/A	Corrected typographical errors and other minor edits.
04/05/2016 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices