XAPP1257 (v1.0) September 30, 2015

MultiBoot and Fallback with SPI Flash in UltraScale FPGAs

Author: Wendy Curran

Summary

This application note describes the key concepts for building a successful MultiBoot design for the UltraScale™ FPGA serial peripheral interface (SPI) configuration mode. The MultiBoot feature in UltraScale FPGAs allows the FPGA application to load two or more FPGA bitstreams under the control of the FPGA application. This document discusses step-by-step instructions to implement the fallback feature using bitstream settings, different methods of triggering fallback, and details on how to use the boot status (BOOTSTS) register for debugging and verifying fallback operation. The application note includes a reference design to demonstrate the fallback capabilities of UltraScale FPGAs using SPI mode.

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 UltraScale FPGA's MultiBoot and fallback features support updating systems in the field. Bitstream images can be upgraded dynamically in the field, which is a huge advantage for designers. The FPGA MultiBoot feature enables switching between images on the fly. When an error is detected during the MultiBoot configuration process, the FPGA can trigger a fallback feature that ensures a known good design can be loaded into the device.

This document discusses the UltraScale FPGA MultiBoot and fallback feature with respect to the SPI (x1/x2/x4) configuration interface. For this application, a Micron N25Q256 serial NOR flash memory device is used in SPI x4 configuration mode on the Xilinx KCU105 development board. For further details on the SPI x4 configuration interface, refer to the *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1].

MultiBoot and Fallback Basics

The UltraScale architecture supports MultiBoot in SPI x1, x2, and x4, which allows the FPGA to load its bitstream from an attached SPI flash device containing two or more bitstreams. In this mode, the FPGA application triggers a MultiBoot operation, causing the FPGA to reconfigure from a different bitstream. After a MultiBoot operation is triggered, the FPGA restarts its configuration process as usual and clears its configuration memory except for the dedicated MultiBoot logic, the warm boot start address (WBSTAR) register, and the BOOTSTS register. The FPGA then reconfigures from the SPI flash device with the new bitstream.

Conditions for Fallback to be Triggered

These errors can trigger fallback during configuration:

- IDCODE error
- Cyclic redundancy check (CRC) error
- Watchdog timer timeout error

Fallback can also be enabled with the bitstream option ConfigFallback. The watchdog timer is disabled during fallback reconfiguration. If fallback reconfiguration fails, configuration stops and both INIT_B and DONE are held Low.

Golden Image

At FPGA device power-up, the golden image is loaded starting from address location 0 (Figure 1). The golden image contains an upper address space specified in the WBSTAR (next_config_addr) register. IPROG is automatically embedded in the bitstream when WBSTAR is set to an address value other than the default. At power-up, the golden image gets loaded and triggers booting of the MultiBoot image from this upper address space. It is possible to have multiple MultiBoot images, and any design can trigger any other image to be loaded. If an error occurs while the MultiBoot image is being booted that causes configuration to fail, the fallback circuitry triggers the golden image to be loaded from address 0.

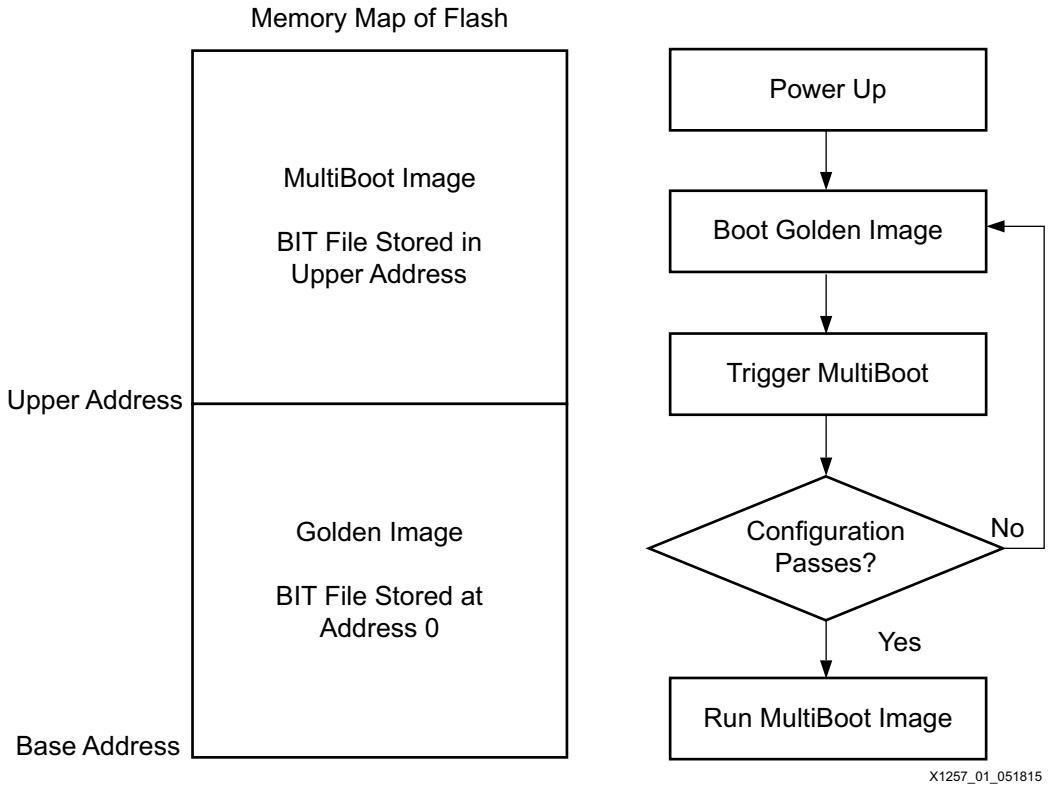

Figure 1: **Multiboot and Fallback Flow**

MultiBoot Image

At power-up, the MultiBoot image is first loaded from an upper address space. If this image fails to configure, a fallback is automatically triggered to the golden image stored at address 0. The fallback functionality allows for system recovery from any failure to load the MultiBoot image, and loads the golden image. Loading the golden image at this stage can fix any errors in the flash and trigger another configuration from the MultiBoot image.

MultiBoot Fallback Reference Design

This section describes how to generate and then verify a MultiBoot fallback reference design.

Generating a MultiBoot Design

Bitstream Settings for the MultiBoot Reference Design

Table 1 provides the required bitstream settings specific to the MultiBoot design. This includes all the default and available values, and the options used for the reference design.

Table 1: **Bitstream Settings**

Settings	Default Value	Possible Values	Golden Image	MultiBoot Image	Description
BITSTREAM.CONFIG.NEXT_CONFIG_ADDR	None	<string>	0x0400000	N/A	Sets the starting address for the next configuration in a MultiBoot setup.
BITSTREAM.CONFIG.NEXT_CONFIG_REBOOT	Enable	Enable, Disable	Enable	N/A	When set to Disable, the IPROG command is removed from the bitfile.
BITSTREAM.CONFIG.SPI_32BIT_ADDR	No	No, Yes	Yes	Yes	Enables SPI 32-bit address style, which is required for SPI devices with storage of 256 Mb and larger.
BITSTREAM.CONFIG.SPI_BUSWIDTH	None	NONE, 1, 2, 4	4	4	Sets the SPI bus to dual (x2) or quad (x4) mode for master SPI configuration from third-party SPI flash devices.
BITSTREAM.CONFIG.CONFIGFALLBACK	Enable	Enable, Disable	N/A	Enable	Enables or disables the loading of a default bitstream when a configuration attempt fails.
BITSTREAM.GENERAL.COMPRESS	False	True, False			Uses the multiple frame write feature in the bitstream to reduce the size of the bitstream, not just the Bitstream (.bit) file. Using Compress does not guarantee that the size of the bitstream shrinks.

Note: For all other available bitstream options, refer to *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 2].

The bitstream settings can be set via the Tcl console or directly in the XDC file. The settings specific to this reference design can be seen below.

Setting via the XDC File

This section lists the bitstream settings that are made via the XDC file.

Golden Image

```
set_property BITSTREAM.CONFIG.NEXT_CONFIG_ADDR 0x0400000 [current_design]
set_property BITSTREAM.CONFIG.NEXT_CONFIG_REBOOT ENABLE [current_design]
set_property BITSTREAM.CONFIG.SPI_32BIT_ADDR YES [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
```

MultiBoot Image

```
set_property BITSTREAM.CONFIG.CONFIGFALLBACK ENABLE [current_design]
set_property BITSTREAM.CONFIG.SPI_32BIT_ADDR YES [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
```

Notes:

- BITSTREAM.CONFIG.SPI_32BIT_ADDR is set because the targeted SPI device is the Micron N25Q256 serial NOR flash device.
- BITSTREAM.GENERAL.COMPRESS is optional but is used to reduce configuration time.

Generating SPI Flash Files Using write_cfgmem

An MCS file is required to program the configuration memory device. To generate an MCS file in the Vivado® Design Suite, the write_cfgmem Tcl command is used. See the examples below:

```
write_cfgmem -force -format MCS -size 32 -interface SPIx4 -loadbit "up 0x00000000
Golden.bit up 0x00400000 Update.bit" KCU105_multiboot_spix4.mcs
```

For more information on the write_cfgmem command and all available options, refer to *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 2] and *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 3] or the Tcl help for the write_cfgmem command.

Design Verification in Hardware

This section describes how to verify your MultiBoot fallback reference design in hardware.

Hardware Requirements

- KCU105 evaluation board.
- USB A-to-micro-B cable to plug into the KCU105 Digilent USB-to-JTAG module or Xilinx platform cable USB II.

Software Requirements

- Vivado Design Suite 2015.1.

Expected Behavior of Images

Included with the reference design are golden and update images (both BIT and MCS files are provided) that can be used to validate the behavior of each image.

- The golden image illuminates the GPIO LEDs [3:0] in rotation from left to right (on board from 3 to 0).
- The update image illuminates the GPIO LEDs [3:0] in rotation from right to left (on board from 0 to 3).

Board Setup

For successful SPI configuration, you must ensure that SW15 correctly reflects the SPI configuration mode. The FPGA mode pins M1 and M0 are hard-wired to logic 0 and 1, respectively. FPGA mode pin M2 is wired to SW15 position 6, allowing the M2 net to be pulled down to logic 0 to select Quad SPI (QSPI) mode (Figure 2). SW15.6 ensures a bitstream programmed into the dual QSPI flash is used to configure the UltraScale FPGA.

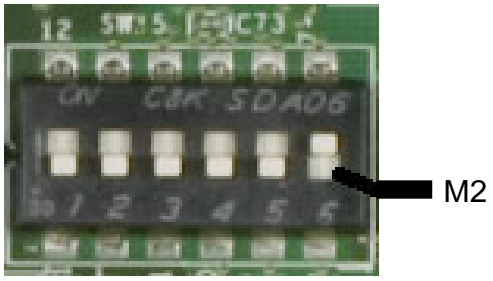

Figure 2: **KCU105 SW15**

Programming the Flash

To program the flash device, it is first necessary to connect to the hardware target in the Vivado Integrated Design Environment (IDE). Follow these steps to connect to the hardware target in the Vivado IDE:

1. From the Flow Navigator pane on the left hand side of the Vivado IDE in the Program and Debug section, click the **Open Hardware Manager** button.
2. When the Hardware Manager opens, click **Open Target**.
3. Select **Auto Connect** to try to automatically connect with the default settings to an attached board (Figure 3).

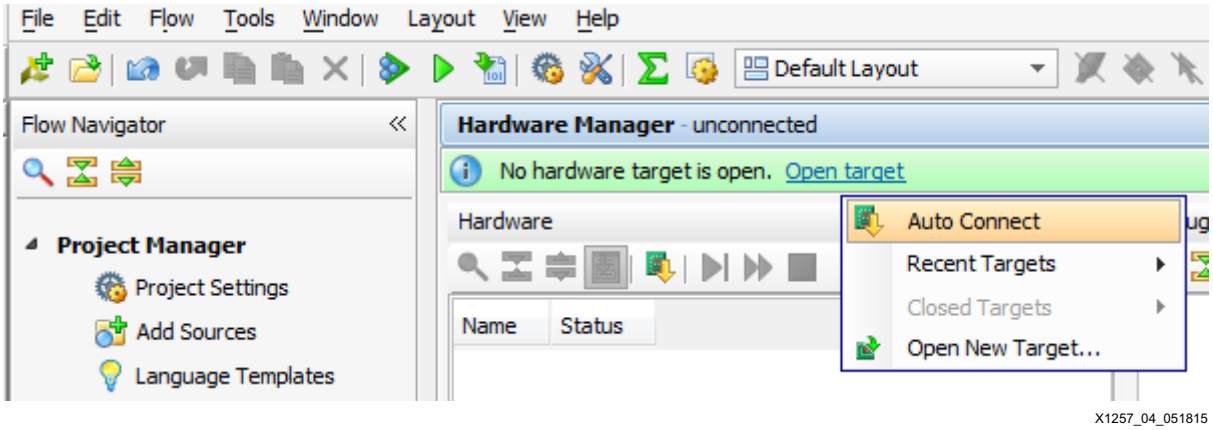

Figure 3: **Auto-Connecting to Board in Hardware Manager**

4. When the FPGA appears in the Hardware Manager console, right-click the FPGA and select **Add Configuration Memory Device...** (Figure 4).

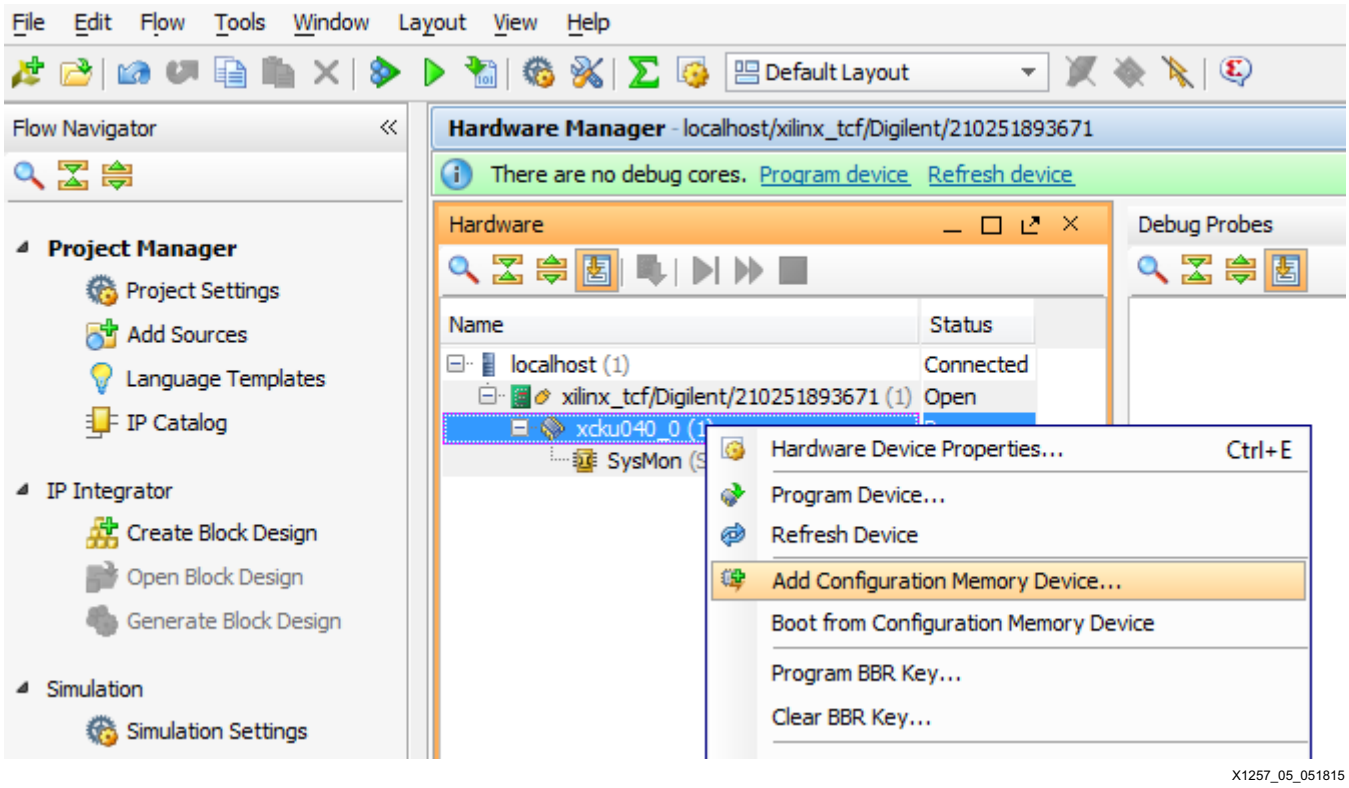

Figure 4: **Add Configuration Memory Device Option in the Vivado IDE**

5. Select the device for this reference design (i.e., N25Q256-1.8v-spi-x1_x2_x4) and click **OK** (Figure 5).

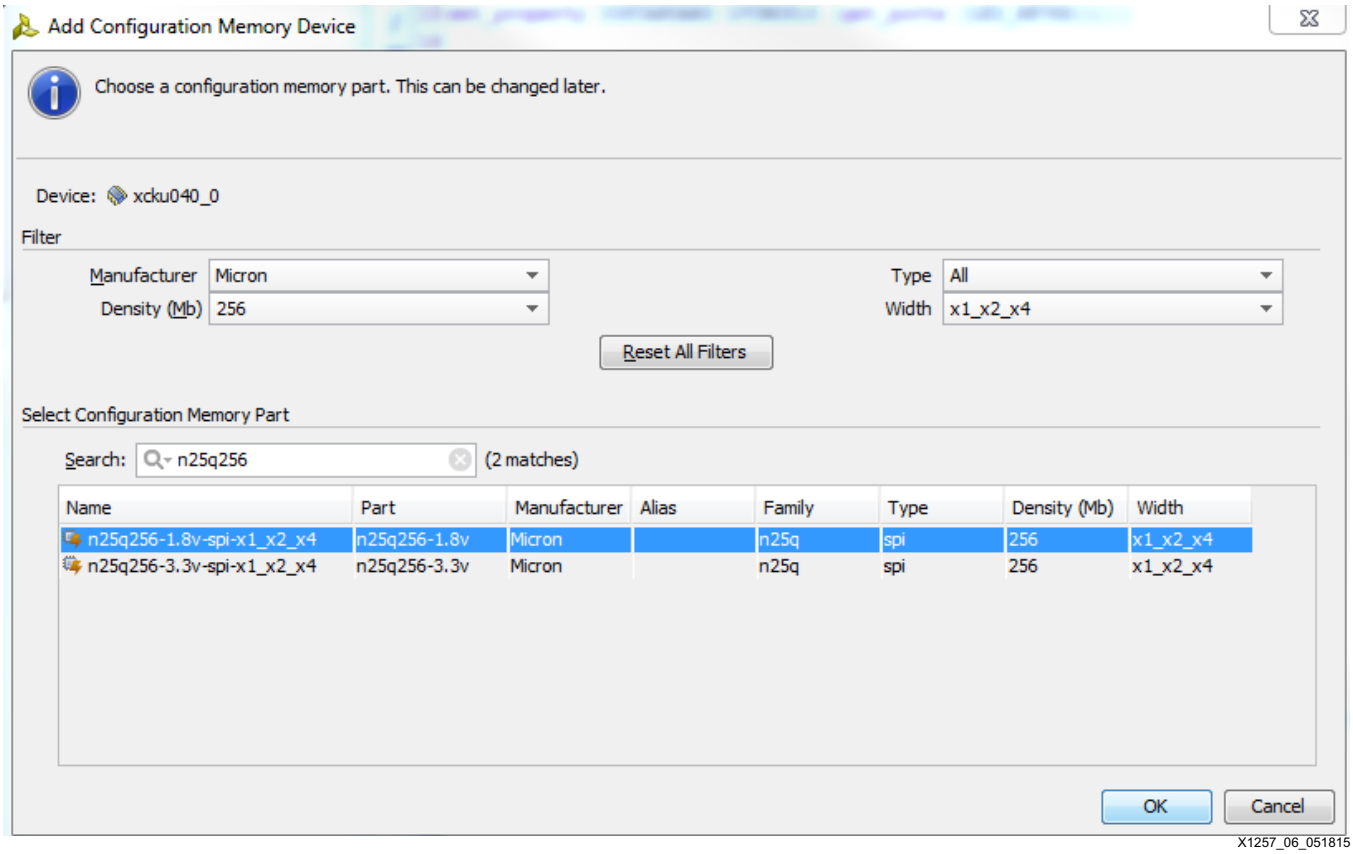

Figure 5: **Adding Configuration Memory Device in the Vivado IDE**

6. Click **OK** when prompted to program the configuration memory device.

7. Select the previously generated MCS file (`.../ready_to_download/KCU105_multiboot_spix4.mcs`) and click **OK** to begin programming (Figure 6).

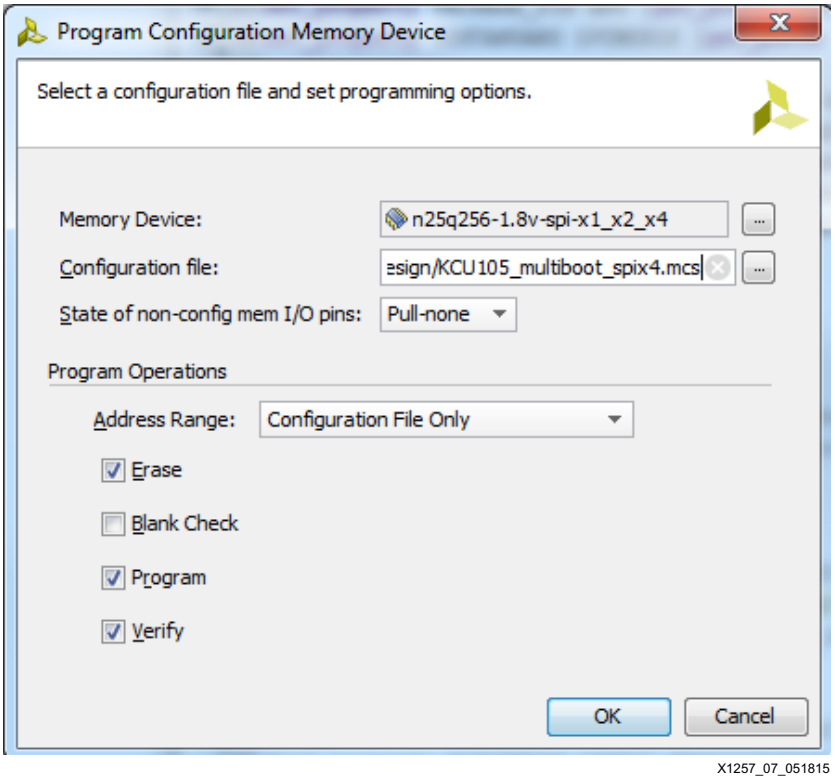

Figure 6: **Adding Configuration File in the Vivado IDE**

Verify MultiBoot Operation

The Vivado IDE displays the window shown in Figure 7 after the flash device has been successfully programmed.

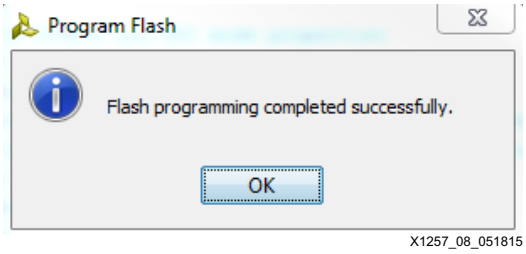

Figure 7: **Window Indicating Flash Programming is Successful**

To boot the FPGA with the image programmed into the flash device, use one of these methods:

1. Pulse PROGRAM_B by pulsing SW4 on the KCU105 board.
2. Use the boot_hw_device TCL command: `boot_hw_device [lindex [get_hw_devices] 0]`.

3. From the Vivado IDE, right-click the device and select **Boot from Configuration Memory Device** (Figure 8).

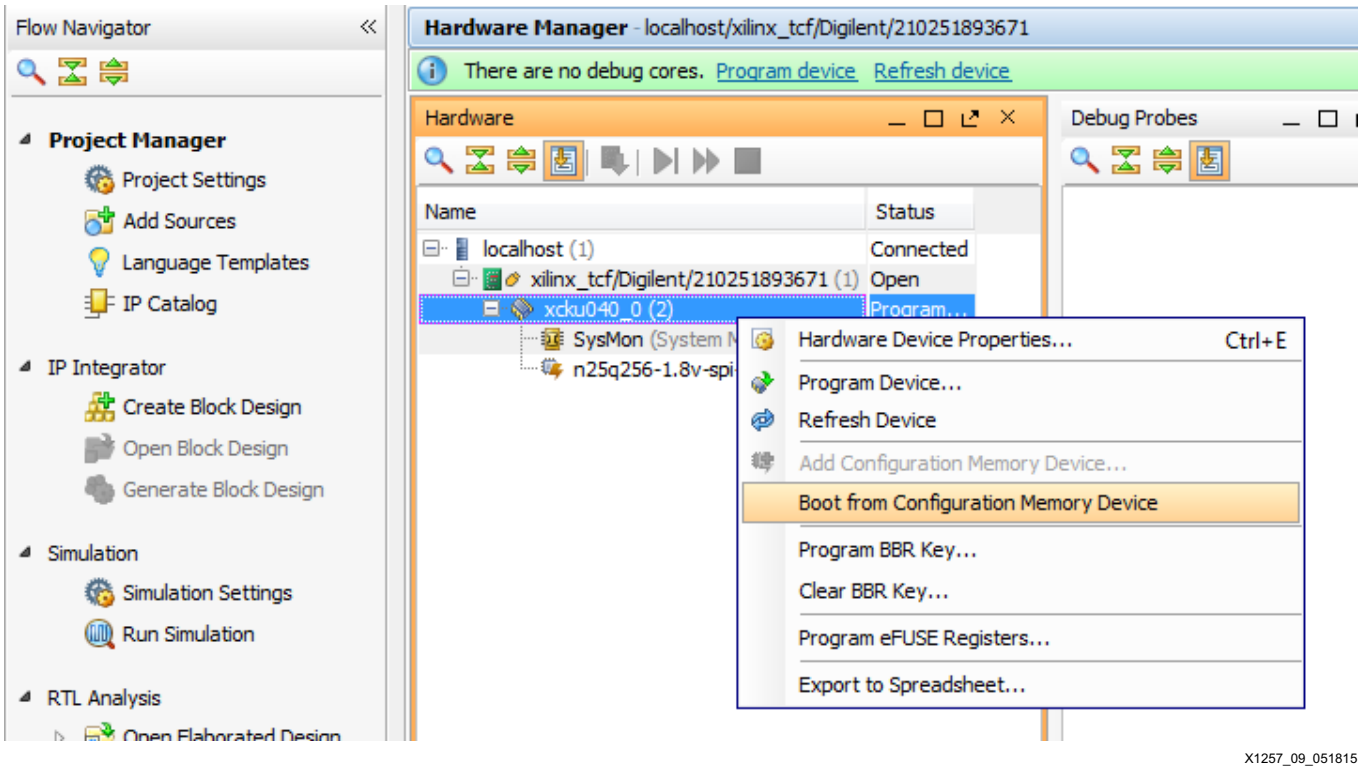

Figure 8: **Boot from Configuration Memory Device Option in Vivado Hardware Manager**

Verify that the FPGA was successfully configured with the update bitstream from the SPI flash using these methods:

- The DONE pin LED on the board should be illuminated.
- The GPIO LEDs [3:0] should illuminate in rotation from right to left, indicating the update bitstream was successfully loaded.

- Refresh the device by right-clicking the FPGA in the Vivado IDE and selecting **Hardware Device Properties** (Figure 9).

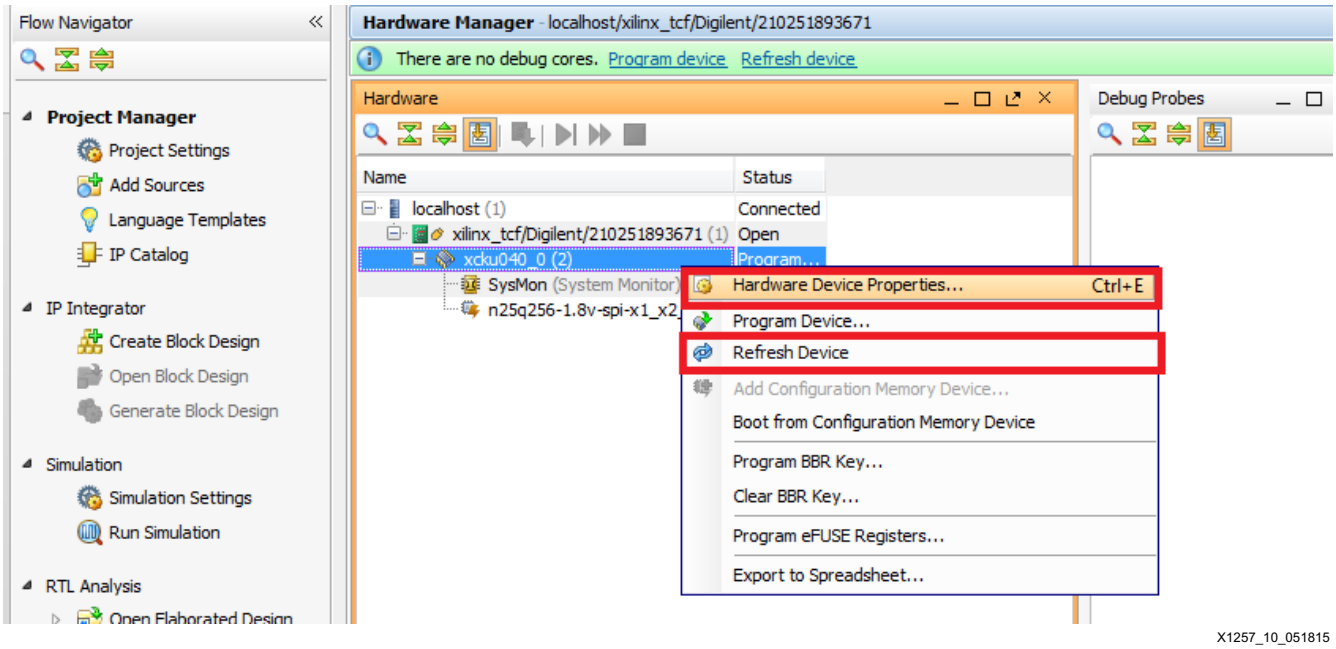

Figure 9: **Hardware Device Properties and Refresh Device Options in Vivado Hardware Manager**

- From the Properties box in the Vivado IDE, expand **BOOT_STATUS** and **CONFIG_STATUS** under **REGISTER**. The BOOT_STATUS register confirms that the IPROG (INTERNAL_PROG) flag that caused the jump to the Update bitstream is High. The CONFIG_STATUS register shows the DONE_PIN is High (Figure 10).

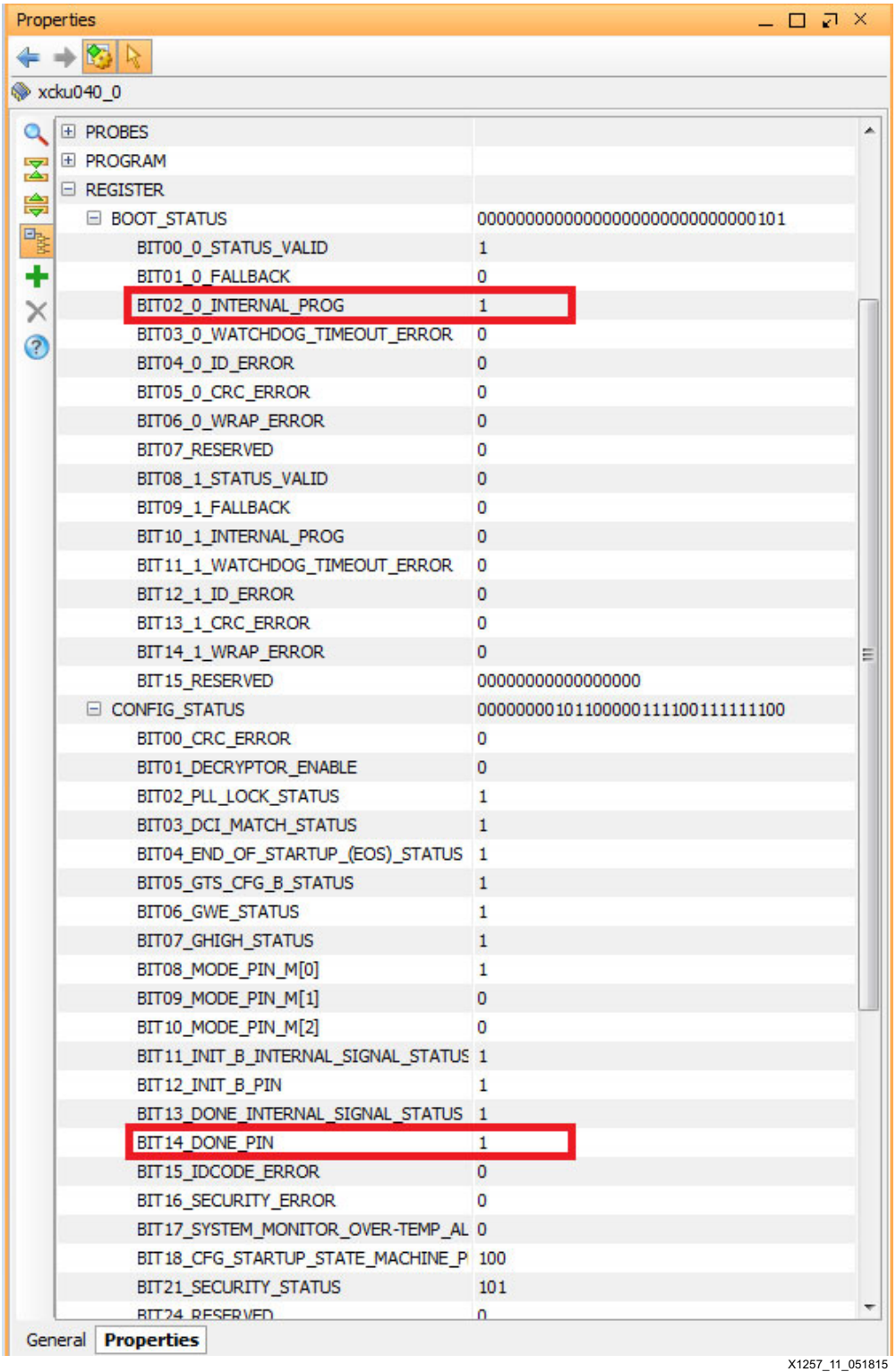

Figure 10: **BOOT_STATUS Indicating IPROG and DONE_PIN High**

Fallback Example – IDCODE Error

A possible test for a successful fallback is to deliberately edit the IDCODE of the update image. From the *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1], it can be seen that the IDCODE for the XCKU040 device is X3822093.

- In the reference design, the `KCU105_multiboot_spix4.mcs` file was opened in an editor (Figure 11).

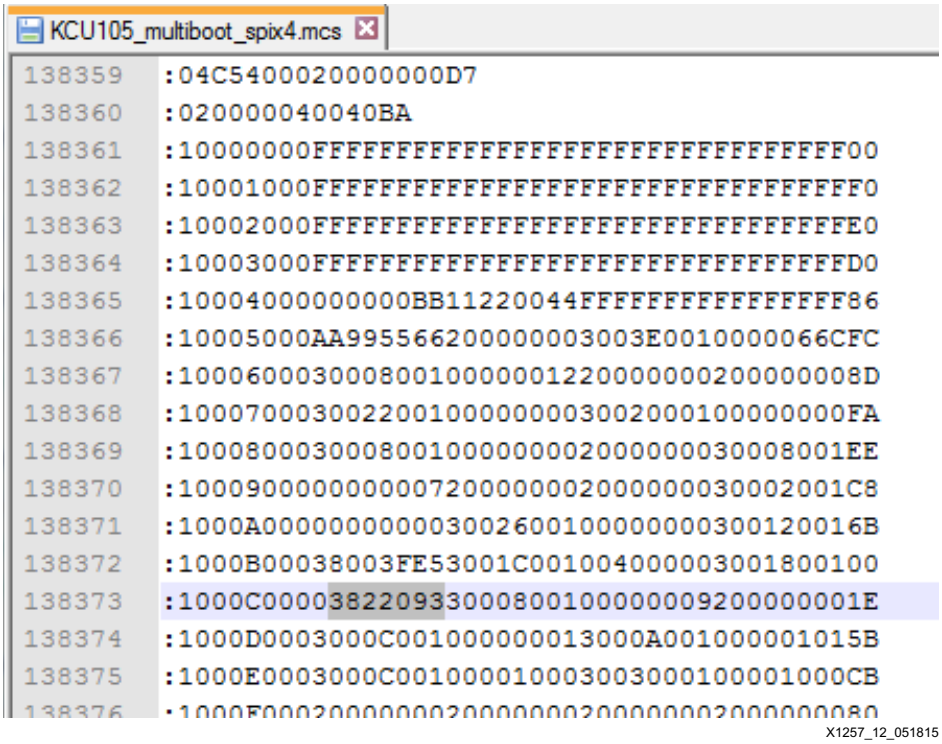

Figure 11: **Original MultiBoot Image**

- The IDCODE from the update image was located in the MCS file. This was then changed from X3822093 to X3823093 and the file was saved as `.../ready_to_download/KCU105_multiboot_corrupt_idcode_spix4.mcs` (Figure 12).

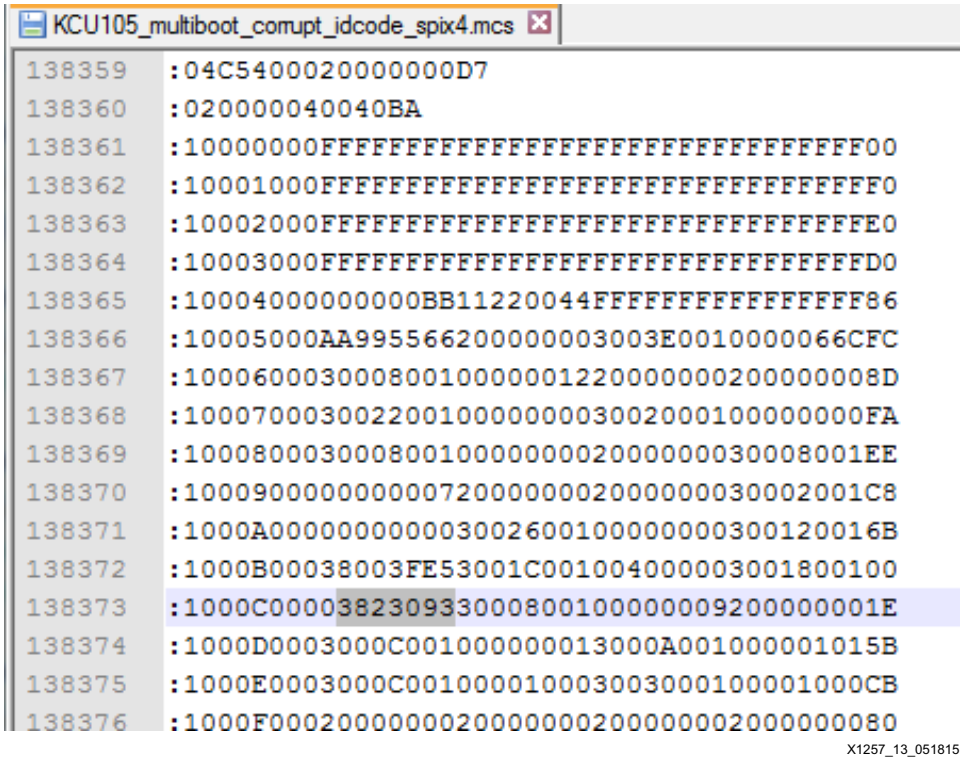

Figure 12: **Corrupt IDCODE MultiBoot Image**

- Program the SPI flash with the newly created MCS file using the steps outlined in Programming the Flash.
- Verify that the FPGA was successfully configured with the golden bitstream from the SPI flash using these methods:
  - Check that the DONE pin LED is illuminated on the board.
  - The GPIO LEDs [3:0] should illuminate in rotation from left to right, indicating the golden bitstream was successfully loaded.
  - Refresh the device by right-clicking on the FPGA in the GUI and selecting **Hardware Device Properties**.

- From the Properties box in the Vivado GUI, expand **BOOT_STATUS** and **CONFIG_STATUS** under **REGISTER**. The BOOT_STATUS register confirms that the IPROG (INTERNAL_PROG) flag that caused the jump to the update bitstream is High. The CONFIG_STATUS register shows the DONE_PIN is High (Figure 13).

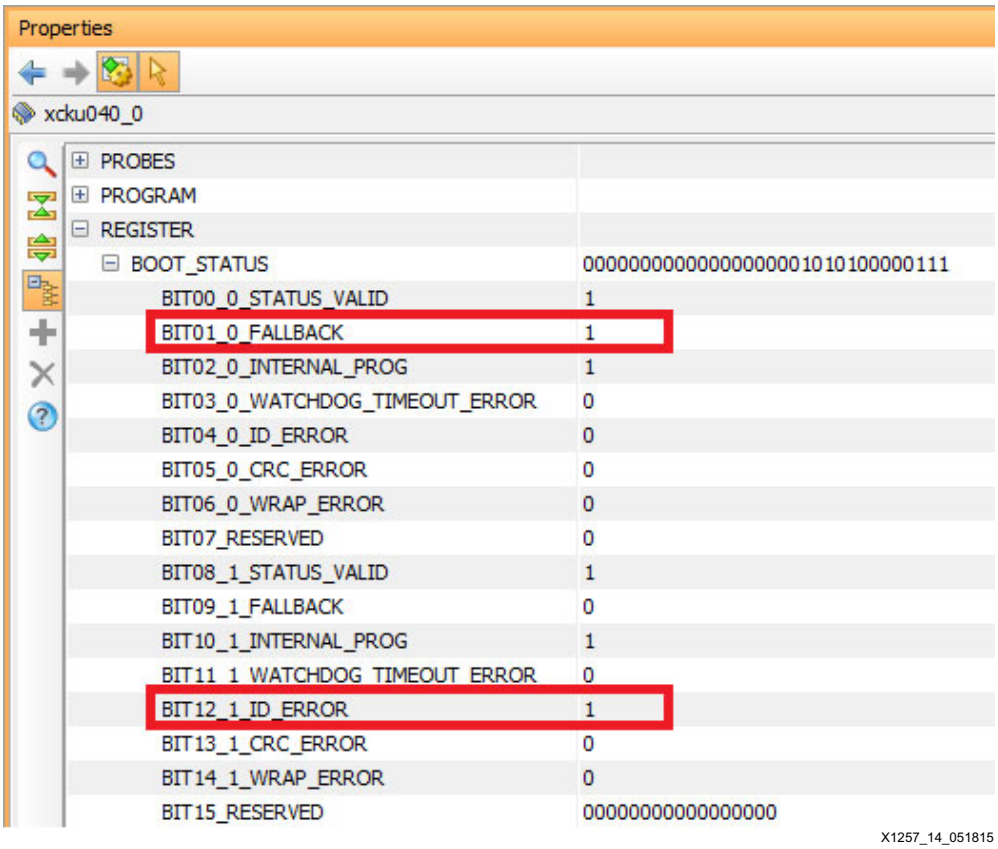

Figure 13: **BOOT_STATUS Indicating Fallback and ID_ERROR**

Fallback – Watchdog Timer

Sometimes, configuration might not start (maybe due to no image being present at the next_config_addr location) or configuration starts but does not complete (possibly due to a corrupt SYNC word in the update image or partially corrupted configuration source). In these scenarios, the watchdog timer can be used in the bitstream options to allow a retry of the configuration after a certain period of time. When the watchdog timer times out, the configuration logic loads the fallback bitstream. The watchdog timer can be set via the bitstream property BITSTREAM.CONFIG.TIMER_CFG, which must be set in both the golden and update images.

The TIMER register works by counting down from the start to the bitstream and is disabled by the end of the startup sequence. A fallback is triggered when the count reaches 0. Delays that need to be considered and included are any wait time in startup for MMCM wait, DCI match settings, or DONE. The TIMER register runs at approximately 50 MHz. A dedicated internal clock (CFG_MCLK) is used which has a nominal frequency of 50 MHz. The clock is pre-divided by 256 so that the watchdog timer clock period is about 5,120 ns. Given that the watchdog counter is 30 bits wide, the maximum possible watchdog value is about 5,500 seconds.

Design Debug

This section provides a list of steps that can be followed to debug common issues for MultiBoot with SPI flash.

File Generation

- Ensure all MultiBoot bitstream properties are correctly set for both the golden and update images (see Table 1).
- Ensure all SPI bitstream properties are correctly set (see Table 1).
- Ensure that the command to generate the flash programming file has all the correct options included as described in Generating SPI Flash Files Using write_cfgmem.
- Ensure that the update image start address is the same as specified in the bitstream property NEXT_CONFIG_ADDR.

Configuration

- Ensure that the design itself, including both the golden and update images, works as expected before introducing the MultiBoot properties. This helps to identify whether the root cause of an issue is related to the design or the MultiBoot properties. This can be achieved by configuring with the following MCS files included in the reference design:
  - `KCU105_golden_spix4.mcs`
  - `KCU105_update_spix4.mcs`

  These do not include the MultiBoot properties but can be used to show the behavior of the images:
  - The golden image illuminates the GPIO LEDs [3:0] in rotation from left to right.
  - The update image illuminates the GPIO LEDs [3:0] in rotation from right to left.
- Ensure that the flash device is completely erased before attempting to program the flash with the design. The erase can be verified using the blank check option.
- Check the BOOT_STATUS and CONFIG_STATUS registers for errors or behavioral issues to assist with the debug of the MultiBoot design. Ensure that the refresh device is completed before reading the registers.

Conclusion

This application note describes how to use the MultiBoot feature in UltraScale FPGAs that supports updating systems in the field. It provides guidance on how to implement this feature with respect to the SPI (x4) configuration interface. A reference design that demonstrates the operation of the MultiBoot feature is also provided.

Reference Design

You can download the reference design files for this application note from the Xilinx website.

Table 2 shows the reference design matrix.

Table 2: **Reference Design Matrix**

Parameter	Description
General	
Developer name	Wendy Curran
Target devices	Kintex UltraScale FPGAs
Source code provided	Yes
Source code format	VHDL
Design uses code and IP from existing Xilinx application note and reference designs or third party	N/A
Simulation	
Functional simulation performed	N/A
Timing simulation performed	N/A
Test bench used for functional and timing simulations	N/A
Test bench format	N/A
Simulator software/version used	N/A
SPICE/IBIS simulations	N/A
Implementation	
Synthesis software tools/versions used	Vivado Design Suite 2015.1
Implementation software tools/versions used	Vivado Design Suite 2015.1
Static timing analysis performed	N/A
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	KCU105 evaluation board and 256 Mb Micron serial NOR flash memory (N25Q256A)

References

1. *UltraScale Architecture Configuration User Guide* (UG570)
2. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
3. *Vivado Design Suite Tcl Command Reference Guide* (UG835)
4. *KCU105 Board User Guide* (UG917)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/30/2015	1.0	Initial Xilinx release.

Please Read: Important Legal Notices