



XAPP1296 (v1.0) June 23, 2017

# MultiBoot and Fallback Using ICAP in UltraScale+ FPGAs

Author: Guruprasad Kempahonnaiah

## Summary

This application note describes a key feature of UltraScale+™ FPGAs—MultiBoot. The MultiBoot feature in UltraScale+ FPGAs allows the FPGA application to load two or more FPGA bitstreams under the control of the FPGA application. The FPGA application triggers a MultiBoot operation, causing the FPGA to reconfigure from a different configuration bitstream. After a MultiBoot operation is triggered, the FPGA restarts its configuration process as usual. This document discusses step-by-step instructions to implement the MultiBoot feature using ICAP, different methods of triggering fallback, and details on how to use the boot status (BOOTSTS) register for debugging and verifying MultiBoot or fallback operation. The application note includes a reference design to demonstrate the MultiBoot capabilities of UltraScale+ FPGAs using ICAP in SPI mode.

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. 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. Bitstream images can be upgraded dynamically in the field, which is a huge advantage for designers. The FPGA MultiBoot feature enables switching between images in real time. 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 application note discusses the UltraScale+ FPGA MultiBoot and fallback feature with respect to the SPI (x1/x2/x4) configuration interface. For this application, a Micron MT25QU01 serial NOR flash memory device is used in SPI x4 configuration mode on the Xilinx KCU116 development board. For further details on the SPI x4 configuration interface, refer to the *UltraScale Architecture Configuration User Guide* (UG570) [\[Ref 1\]](#).

## Basics of MultiBoot and Fallback

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 that Trigger Fallback

These errors can trigger fallback during configuration:

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

Fallback can be enabled with the bitstream option `BITSTREAM.CONFIG.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 power-up, the golden image is loaded starting from address location `0x0` (Figure 1). At power-up, the golden image gets loaded initially. When a MultiBoot trigger event is recognized, the FPGA loads the MultiBoot image from the 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 `0x0`.

## MultiBoot Image

The MultiBoot image is loaded from an upper address space. If this image fails to configure, a fallback is automatically triggered to the golden image stored at address `0x0`. The fallback functionality allows for system recovery from any failure to load the MultiBoot image, and loads the golden image.

---

# MultiBoot Reference Design

This section describes the expected behavior of the MultiBoot reference design, as well as how to compile and verify the reference design using the KCU116 evaluation board. The reference design uses a golden image initial system setup to showcase the MultiBoot capability.

## Golden Image Initial System Setup

The golden image is loaded starting from address location 0 at FPGA power-up. Next, the golden image design triggers a MultiBoot image to be loaded. This step is beneficial when initial system checking is required prior to loading a run time image. The system checking or diagnostics can be contained in the golden image, and the run time operation can be contained in the MultiBoot image. The golden image loaded at power-up triggers booting from an upper address space. Multiple MultiBoot images can exist, and any design can trigger any other image

to be loaded. If an error occurs during loading of the MultiBoot image from the upper address space, the fallback circuitry triggers the golden image to be loaded from address 0x0. Figure 1 shows the flow for the golden image initial setup.

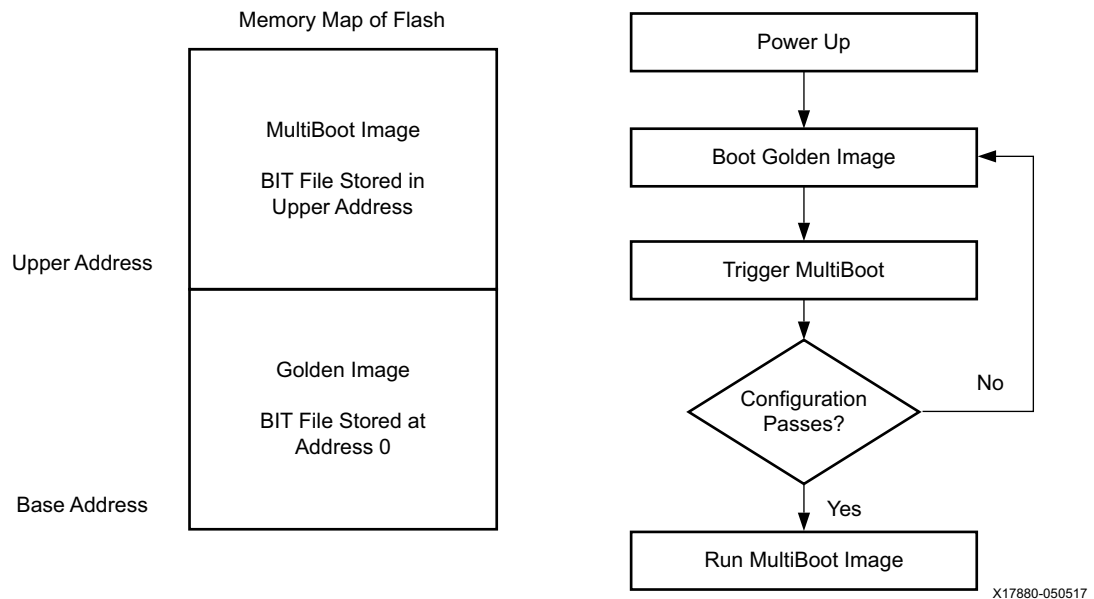


Figure 1: Golden Image Initial Flow Diagram

## Expected Behavior of Images

At power-on, the golden image is configured and the design runs a walking 1 pattern for the GPIO LEDs [0:7]. The UART can also be connected to check the image loaded.

The golden image waits for the DIP SW13[1] to be toggled 0 > 1 > 0 to issue an IPROG and jump to the MultiBoot image at 0x01000000. The reference design uses ICAP to jump to the MultiBoot image. When an IPROG is issued, a message can be seen on the UART.

After successful configuration of the MultiBoot image, the design runs a blinking pattern of all the GPIO LEDs [0:7]. The UART can also be connected to check the image that is loaded. Setting DIP SW13[1] to 1 when in the MultiBoot image causes the system to read the IDCODE of the FPGA and display it via the UART.

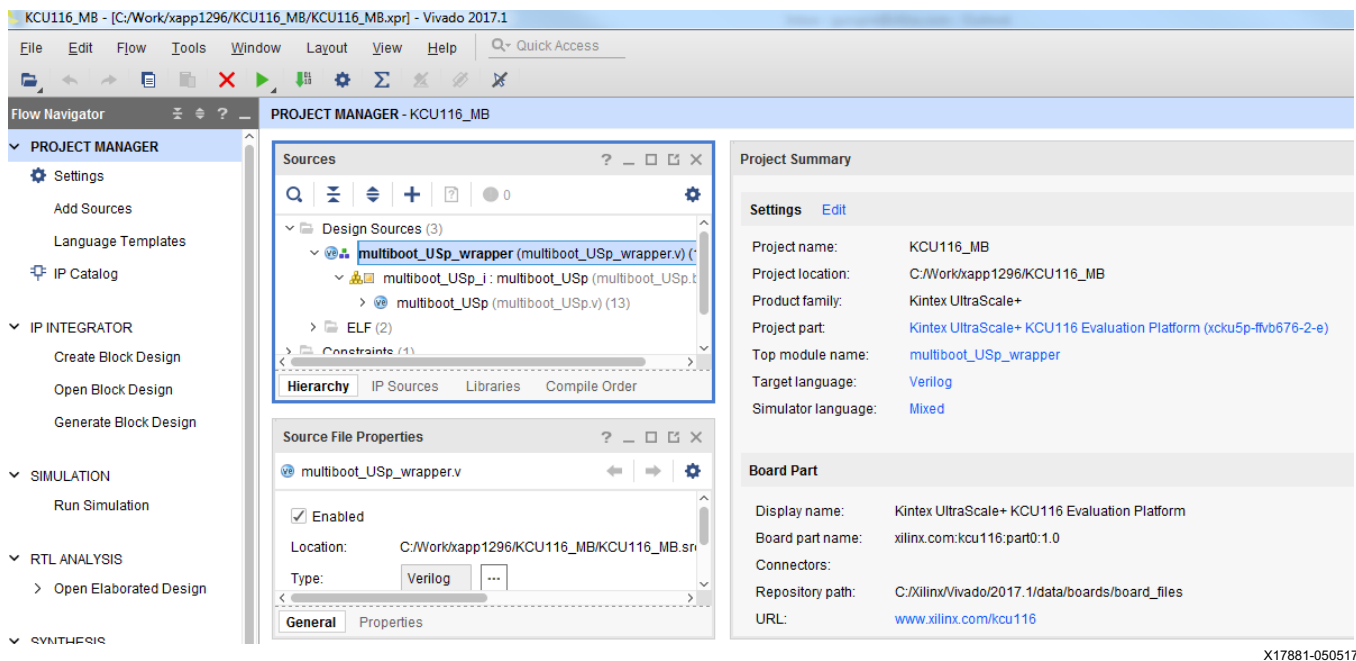
While in the golden image and DIP SW13[4] is toggled 0 > 1 > 0, an IPROG jump is issued to 0x02000000 via the ICAP. Because no valid bitstream (configuration image) is available at this address, the watchdog timer will timeout and trigger a fallback to the golden image.

The configuration time for the SPI x4 interface with 51.0 MHz CCLK setting is less than a second. Watchdog timeout takes approximately 15 seconds at the default CCLK frequency.

## Compiling MultiBoot Reference Design

The MultiBoot reference design has been implemented with IP Integrator (IPI). Follow the steps below to compile and generate golden and MultiBoot image files.

1. Open the Vivado® tools by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2017.1 > Vivado**.
  - a. Select **Open Project**.
  - b. Open the KCU116 MultiBoot reference design (<Directory>\KCU116\_MB.xpr), as shown in [Figure 1](#).

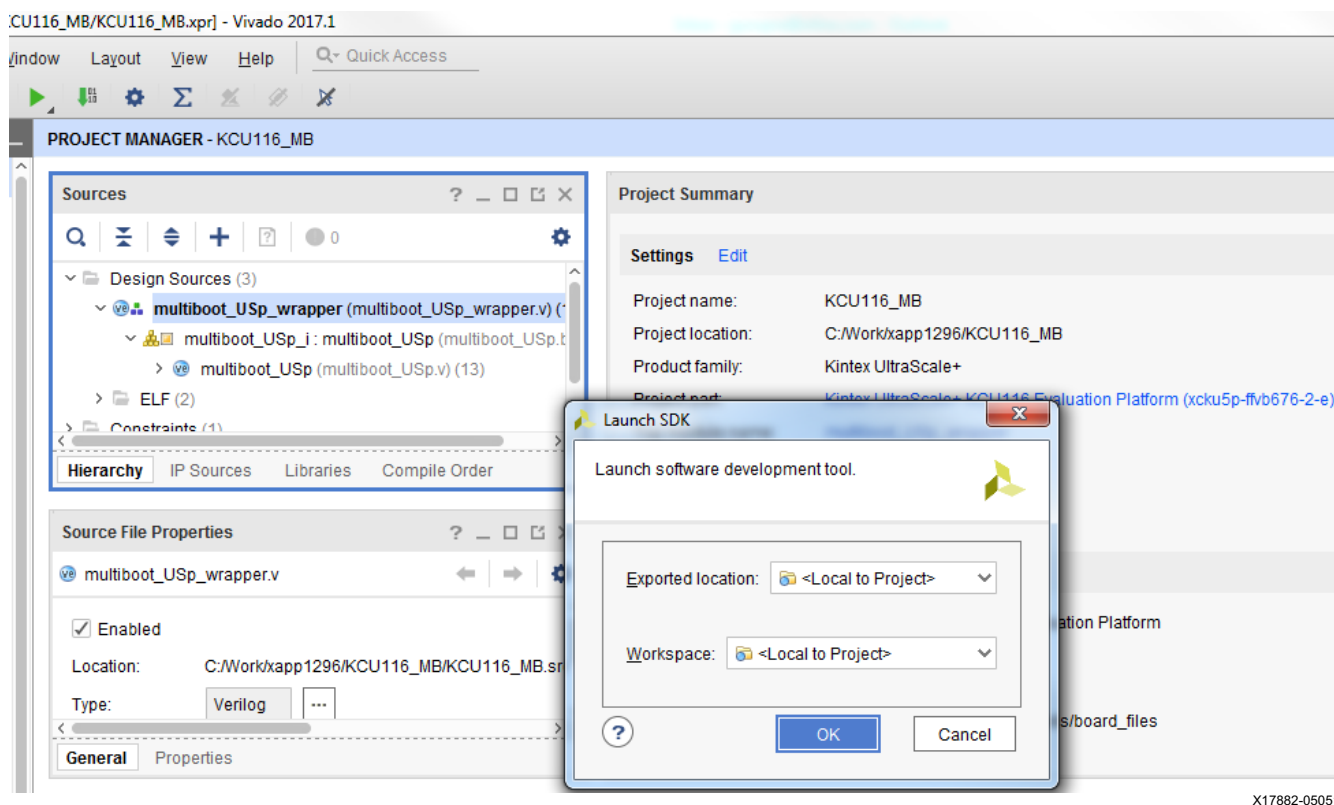


X17881-050517

*Figure 2: MultiBoot Reference Design Project*

- c. The reference design can be recompiled or exported to SDK.
  - To recompile, right-click **synth\_1**, select **Reset Runs**, then select **Generate Bitstream**.
  - To export to SDK, open the implemented design and select **File > Export > Export Hardware**.

- d. To launch SDK, select **File > Launch SDK** (Figure 2).



X17882-050517

Figure 3: Launch SDK

2. SDK software compile: The project automatically builds ELF files in SDK. When done, close SDK and return to the Vivado tools (Figure 4).

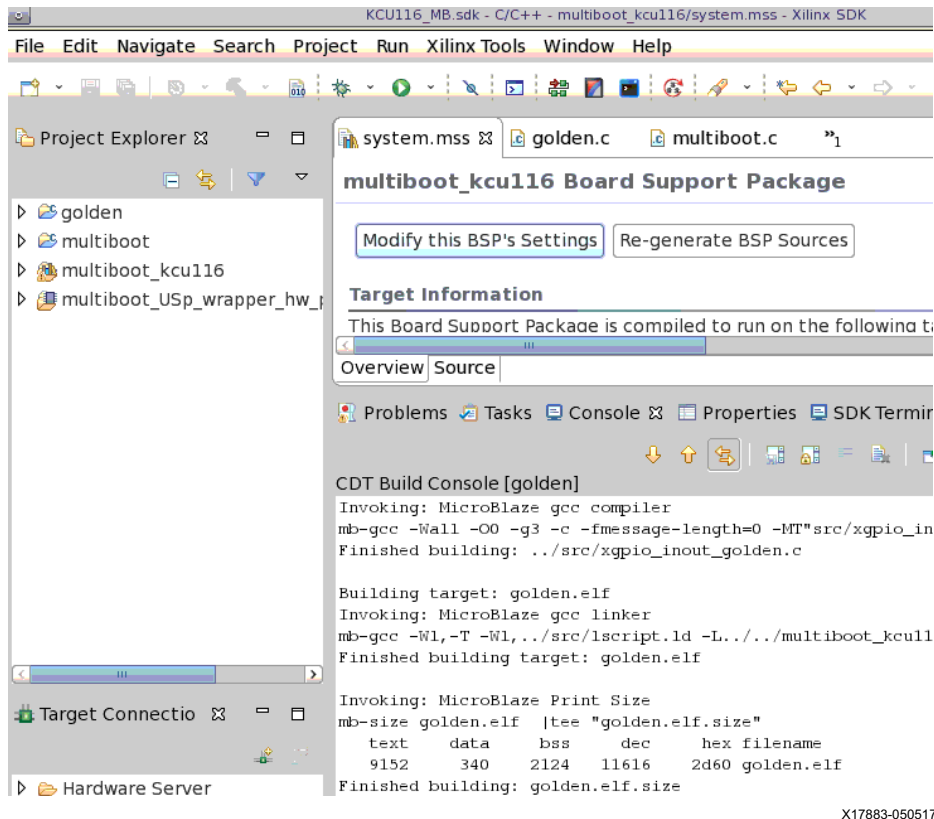


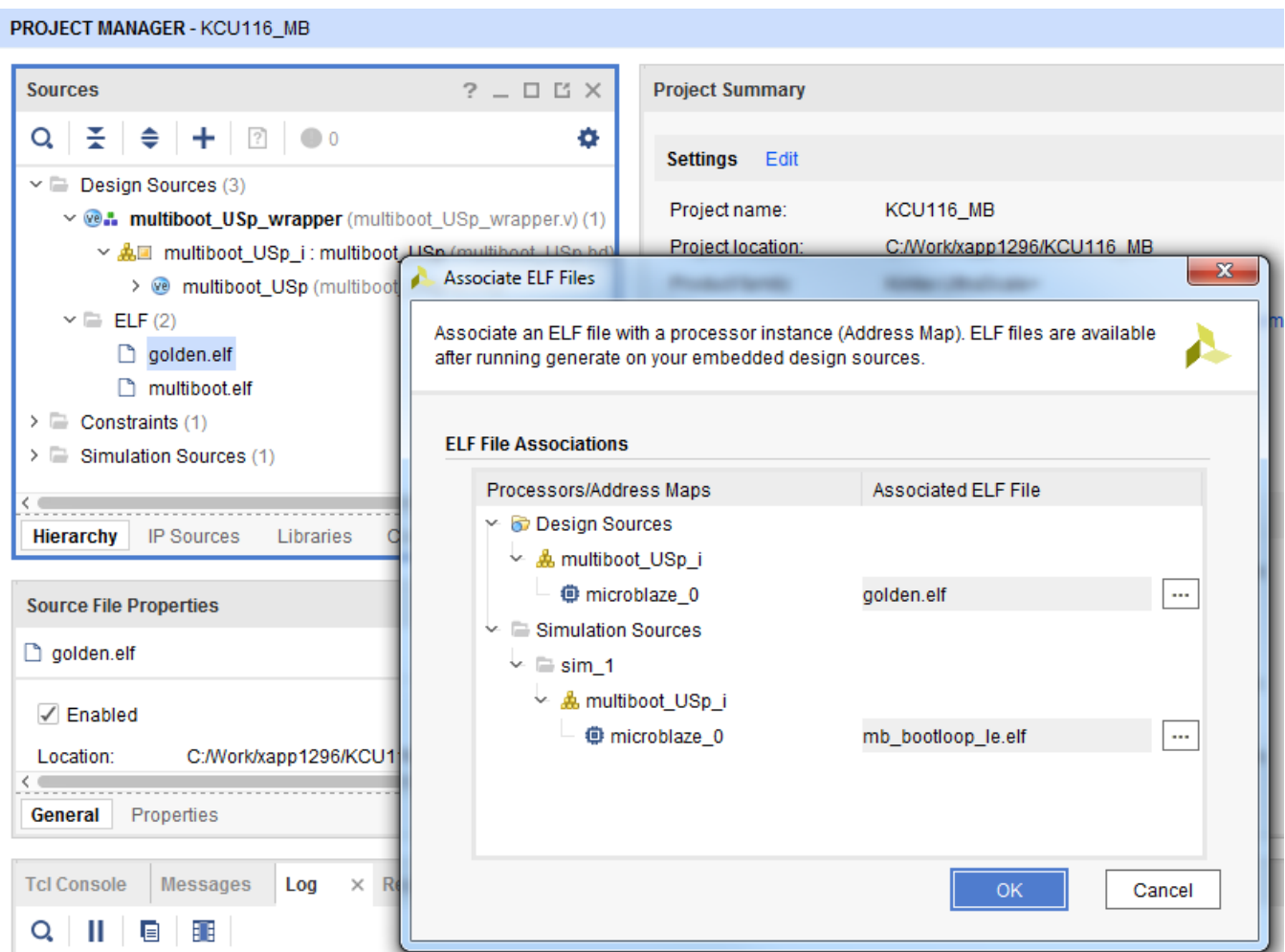
Figure 4: Compile ELF File in SDK

3. After SDK has compiled the ELF files, they must be associated with the design using the Associate ELF command. The ELF files in the SDK are located in the directories below and are associated in the Vivado Project:

```

<Directory>\KCU116_MB.sdk\golden\Debug\golden.elf
<Directory>\KCU116_MB.sdk\multiboot\Debug\multiboot.elf
  
```

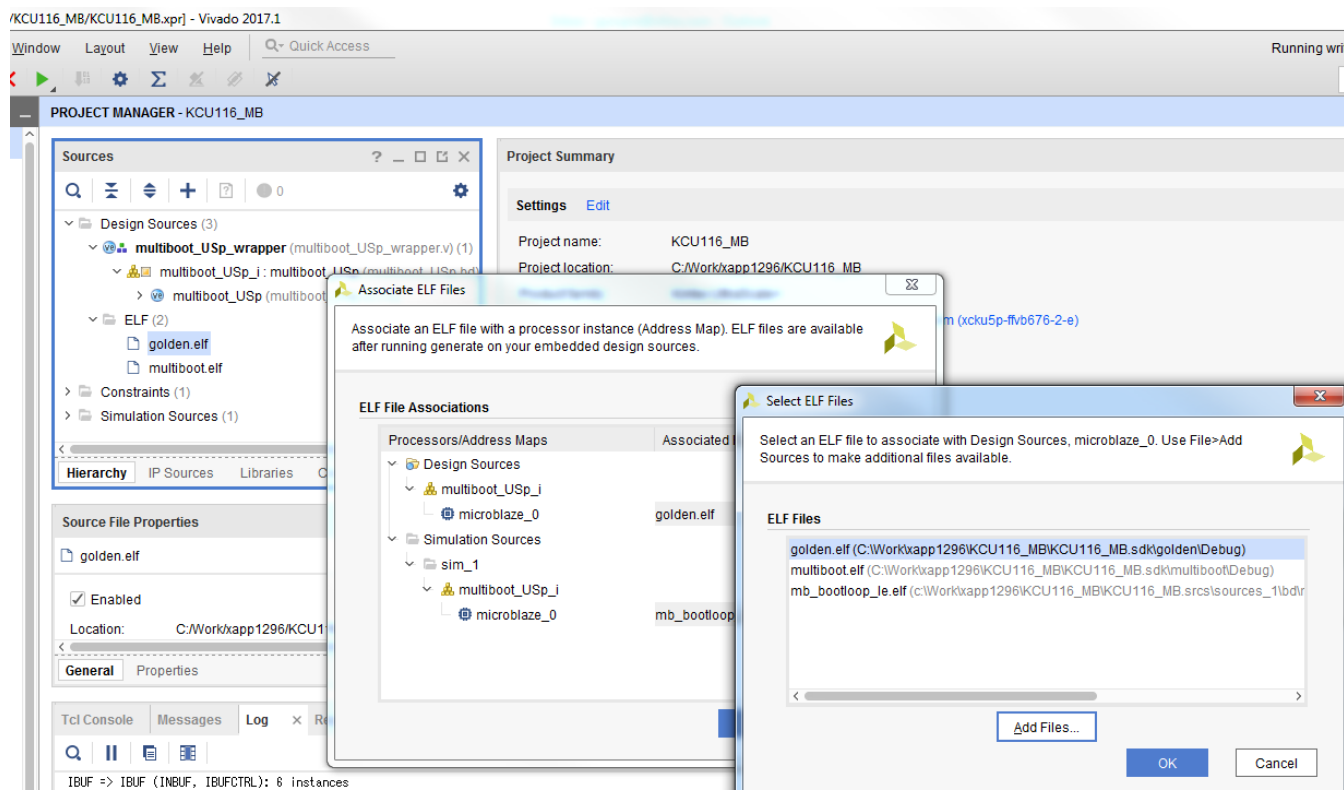
- Right-click one of the ELF files and select **Associate ELF Files** (Figure 5).



X17884-050517

Figure 5: Associate ELF File in the Vivado Tools

- Click the button to the right of `multiboot.elf`, select **Golden.elf**, then click **OK** twice (Figure 6).



X17885-050517

Figure 6: Associate Golden ELF File in the Vivado Tools

- Run `write_bitstream` in the Vivado tools. This generates the golden image bitstream.
- Rename the generated bit file so that it does not get overwritten when the next steps are executed.



8. Click the button to the right of `Multiboot.elf`, select **Multiboot.elf**, then click **OK** twice (Figure 7).

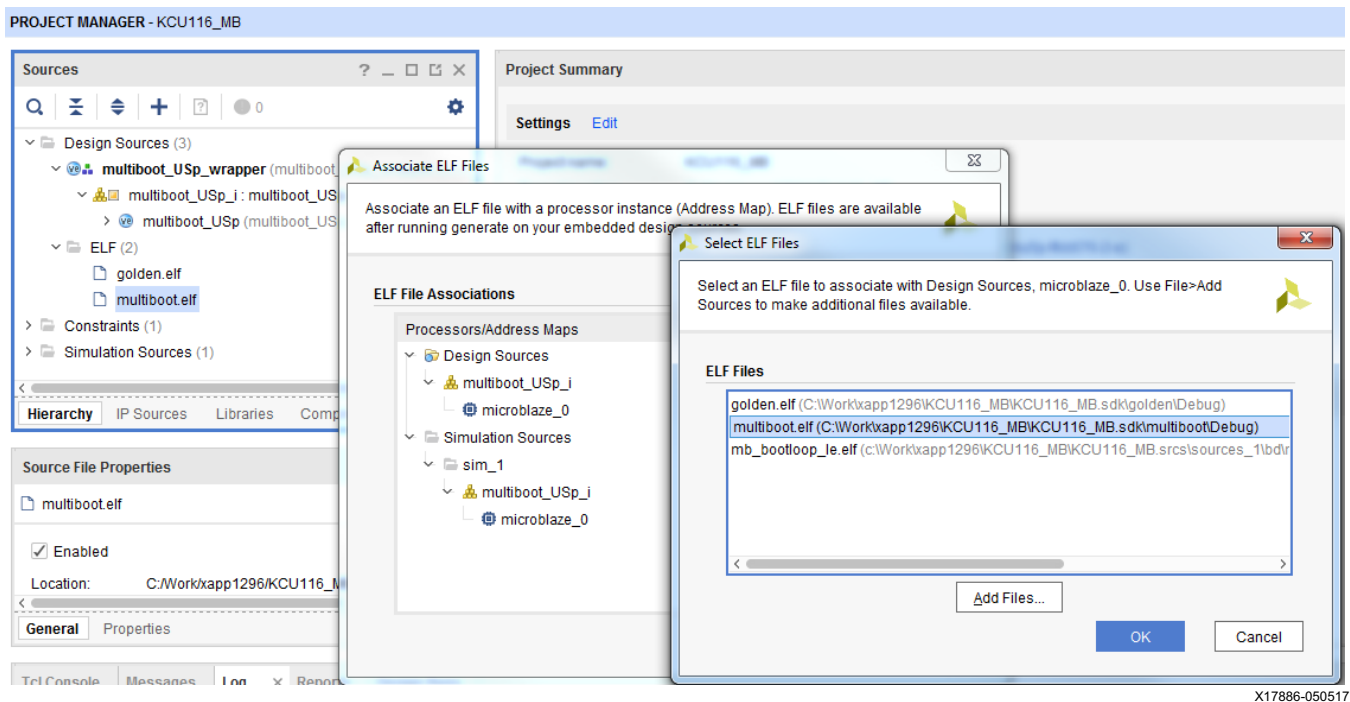


Figure 7: Associate MultiBoot ELF File in the Vivado Tools

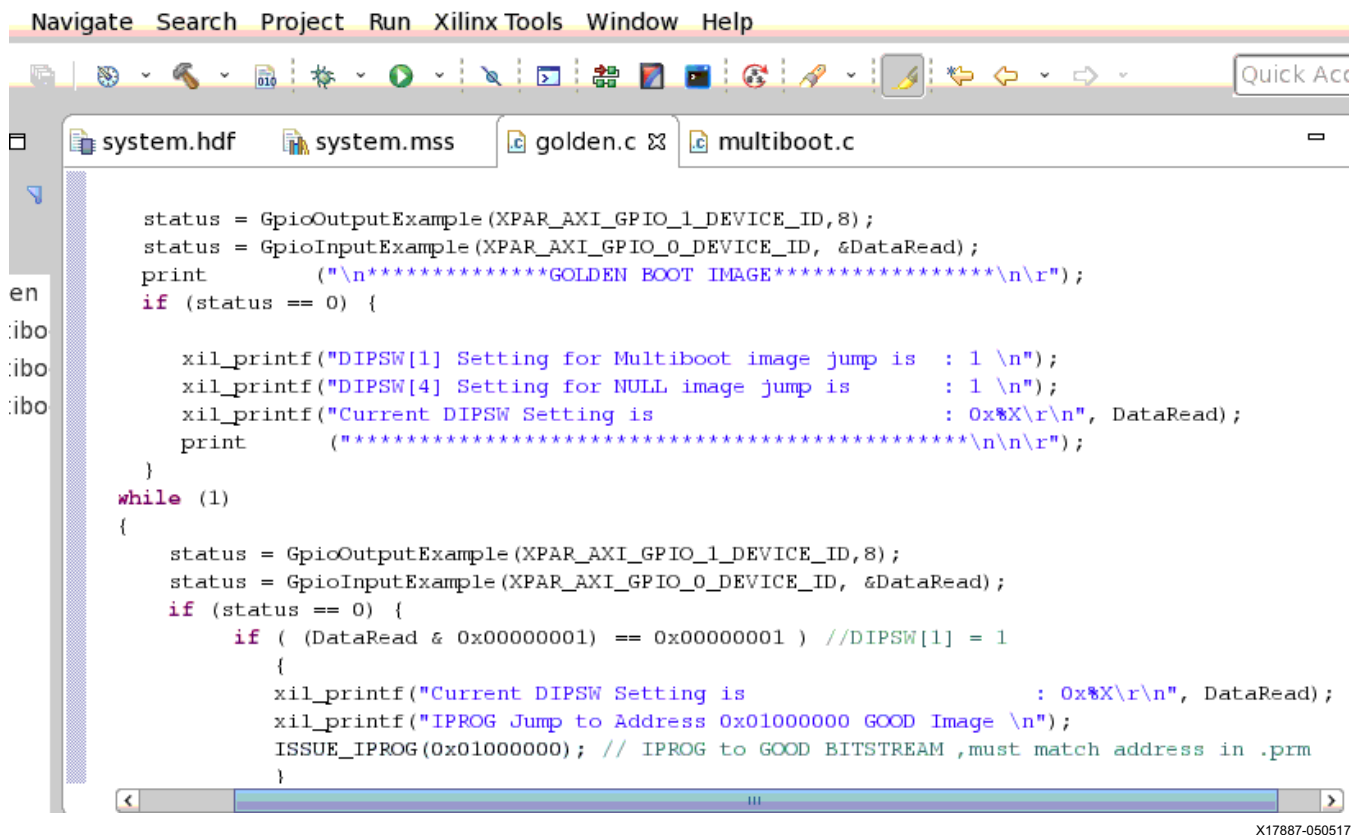
9. Run `write_bitstream` in the Vivado tools. This generates the MultiBoot image bitstream.
10. Generate the MCS file using `Create_MCS.tcl`.

The file `golden.c` in SDK continuously monitors DIPSW13 and controls the issue of IPROG based on the DIPSW13 status. Figure 8 shows the sequence of data programmed into the ICAP when IPROG is issued. The sequence of ICAP commands is from the "Example Bitstream for IPROG through ICAP" table in the *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1].

```
// Reusing IDCODE array and code to send IPROG
// From UG570 Table 11-3
static u32 ReadId[HWICAP_EXAMPLE_BITSTREAM_LENGTH] =
{
    0xFFFFFFFF, /* Dummy Word */
    0xAA995566, /* Sync Word*/
    0x20000000, /* Type 1 NO OP */
    0x30020001, /* Write WBSTAR cmd */
    0x01000000, /* Warm boot start address (Load the desired address) */
    0x30008001, /* Write CMD */
    0x0000000F, /* Write IPROG */
    0x20000000, /* Type 1 NO OP */
};
```

Figure 8: ICAP Commands for IPROG

Figure 9 shows the SDK `golden.c` MultiBoot address.



```

status = GpioOutputExample(XPAR_AXI_GPIO_1_DEVICE_ID, 8);
status = GpioInputExample(XPAR_AXI_GPIO_0_DEVICE_ID, &DataRead);
print      ("\n*****GOLDEN BOOT IMAGE*****\n\r");
if (status == 0) {

    xil_printf("DIPSW[1] Setting for Multiboot image jump is : 1 \n");
    xil_printf("DIPSW[4] Setting for NULL image jump is      : 1 \n");
    xil_printf("Current DIPSW Setting is                          : 0x%X\r\n", DataRead);
    print      ("*****\n\n\r");
}
while (1)
{
    status = GpioOutputExample(XPAR_AXI_GPIO_1_DEVICE_ID, 8);
    status = GpioInputExample(XPAR_AXI_GPIO_0_DEVICE_ID, &DataRead);
    if (status == 0) {
        if ( (DataRead & 0x00000001) == 0x00000001 ) //DIPSW[1] = 1
        {
            xil_printf("Current DIPSW Setting is                          : 0x%X\r\n", DataRead);
            xil_printf("IPROG Jump to Address 0x01000000 GOOD Image \n");
            ISSUE_IPROG(0x01000000); // IPROG to GOOD BITSTREAM ,must match address in .prm
        }
    }
}

```

X17887-050517

Figure 9: SDK `golden.c` MultiBoot Address

Toggling DIP SW13[1] 0 > 1 > 0 causes the design to issue an IPROG with the WBSTAR address set to 0x01000000.

Toggling DIP SW13[4] 0 > 1 > 0 causes the design to issue an IPROG with the WBSTAR address set to 0x02000000.

## Bitstream Settings for the MultiBoot Reference Design

Table 1 shows the bitstream settings used for the MultiBoot reference design. The settings are common to both golden and MultiBoot image bitstreams. The table captures all the default and available values, and the options used for the reference design.

Table 1: Bitstream Settings

Settings	Default Value	Possible Values	Design Settings	Description
BITSTREAM.CONFIG.SPI_BUSWIDTH	None	None, 1, 2, 4, 8	4	Sets the SPI bus to quad (x4) mode SPI configuration.
BITSTREAM.CONFIG.CONFIGFALLBACK	Enable	Enable, Disable	Enable	Enables or disables the loading of a default bitstream when a configuration attempt fails.

Table 1: Bitstream Settings (Cont'd)

Settings	Default Value	Possible Values	Design Settings	Description
BITSTREAM.CONFIG.SPI_32BIT_ADDR	No	No, Yes	Yes	Enables SPI 32-bit address style, which is required for SPI devices with storage of 256 Mb and larger.
BITSTREAM.CONFIG.CONFIGRATE	3	2.7, 5.3, 8.0, 10.6, 21.3, 31.9, 36.4, 51.0, 56.7, 63.8, 72.9, 85.0, 102.0, 127.5, 170.0	51.0	CCLK is set to 51.0 MHz.
BITSTREAM.CONFIG.TIMER_CFG	-	-	0x01FFFFFF	Sets the value of the watchdog timer in configuration mode.
BITSTREAM.GENERAL.COMPRESS	False	True, False	True	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.
BITSTREAM.CONFIG.SPI_FALL_EDGE	No	No, Yes	Yes	Sets the FPGA to use a falling edge clock for SPI data capture. This improves timing margins and might allow faster clock rates for configuration.

## Design Verification in Hardware

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

### Hardware Requirements

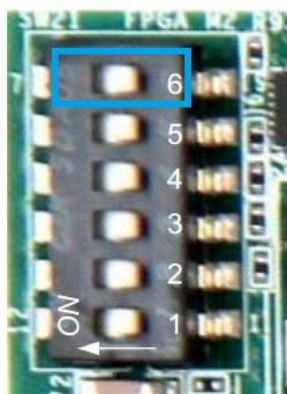
- KCU116 evaluation board.
- USB A to micro B cable to plug into the KCU116 Digilent USB-to-JTAG module or Xilinx platform cable USB II.
- USB A to micro B cable to plug into the KCU116 USB UART interface.

### Software Requirements

- Vivado Design Suite 2017.1 with SDK.
- TeraTerm or any other terminal software for UART connectivity:
  - Baud rate: 9600.
  - Parity, flow control: None.
  - Terminal Setup New Line Receive: Auto.

## Board Setup

The mode pin settings should be set to master SPI. M0 and M1 are hardwired on the KCU116 board. M2 should be set to 0 via the SW21 setting (Figure 10).



X17888-050517

Figure 10: **SW21.6 Set to 0**

The initial DIP SW13 settings should be set to 0000 (Figure 11).



X17889-050517

Figure 11: **DIP SW13[1:4] Set to 0000**

## Programming the Flash

The reference design has pre-generated MCS files that can be used to program the SPI flash MT25QU01 on the KCU116 board. Table 2 contains a description of these files.

Table 2: **MCS Files Description**

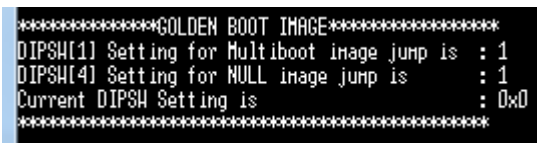
Filename	Description
Golden_n_Multiboot.mcs	Golden and MultiBoot image. Successful configuration of both golden and MultiBoot images can be demonstrated.
Golden_n_Multiboot_CRC_Err.mcs	MultiBoot image CRC is corrupted. Loading MultiBoot results in a CRC error and fallback is triggered.
Golden_n_Multiboot_ID_Err.mcs	MultiBoot image FPGA IDCODE is corrupted. Loading MultiBoot results in ID code error and fallback is triggered.

`Program_KCU116_SPI.tcl` can be used on the Vivado Tcl console to program the SPI flash on the KCU116 board. Edits to the Tcl file might be required depending on the MCS file to be programmed. Alternatively, the Vivado tools hardware manager can be used to program the SPIx4 MT25QU01 flash with any of the MCS files described in [Table 2](#). Refer to *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [\[Ref 2\]](#) for details on programming the flash.

## Verifying MultiBoot Operation

To boot the FPGA with the image (`Golden_n_Multiboot.mcs`) programmed into the flash device, pulse `PROGRAM_B` by pulsing SW5 on the KCU116 board. Verify that the FPGA was successfully configured with the golden bitstream from the SPI flash using these methods:

- The DONE pin LED on the board should be illuminated.
- The GPIO LEDs [0:7] should illuminate with a walking 1 pattern indicating the golden bitstream was successfully loaded.
- The UART terminal should show the message in [Figure 12](#).



```
*****GOLDEN BOOT IMAGE*****
DIPSW[1] Setting for Multiboot image jump is : 1
DIPSW[4] Setting for NULL image jump is      : 1
Current DIPSW Setting is                     : 0x0
*****
```

X17890-050517

Figure 12: Golden Image UART Display

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

- From the Properties box in the Vivado IDE, expand **BOOT\_STATUS** and **CONFIG\_STATUS** under REGISTER. The BOOT\_STATUS register confirms that the normal configuration is successful. The CONFIG\_STATUS register shows the DONE\_PIN is High (Figure 13).

▼ REGISTER	
▼ BOOT_STATUS	00000000000000000000000000000001
BIT00_0_STATUS_VALID	1
BIT01_0_FALLBACK	0
BIT02_0_INTERNAL_PROG	0
BIT03_0_WATCHDOG_TIMEOUT_ERROR	0
BIT04_0_ID_ERROR	0
BIT05_0_CRC_ERROR	0
BIT06_0_WRAP_ERROR	0
BIT07_0_SECURITY_ERROR	0
BIT08_1_STATUS_VALID	0
BIT09_1_FALLBACK	0
BIT10_1_INTERNAL_PROG	0
BIT11_1_WATCHDOG_TIMEOUT_ERROR	0
BIT12_1_ID_ERROR	0
BIT13_1_CRC_ERROR	0
BIT14_1_WRAP_ERROR	0
BIT15_1_SECURITY_ERROR	0
BIT16_RESERVED	0000000000000000
> CONFIG_STATUS	00010000100100000111100111111100

X17891-050517

Figure 13: Golden Image BOOT\_STATUS

After confirming that the golden image is successfully loaded, toggle the DIP SW13[1] 0 > 1 > 0. The DONE LED turns off for a moment, and the MultiBoot image is loaded. When IPROG is issued, the UART displays the message shown in Figure 14.



X17892-050517

Figure 14: IPROG Issue UART Display

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

- The DONE pin LED on the board should be illuminated.
- The GPIO LEDs [0:7] should illuminate with all LEDs in a blinking pattern (ON/OFF) indicating that the MultiBoot bitstream was successfully loaded.
- The UART terminal should display the message shown in Figure 15.



X17893-050517

Figure 15: MultiBoot Image UART Display

- Refresh the device by right-clicking the FPGA in the Vivado IDE and selecting **Hardware Device Properties**.
- 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 MultiBoot bitstream is High. The CONFIG\_STATUS register shows the DONE\_PIN is High ([Figure 16](#)).

REGISTER	
▼ BOOT_STATUS	000000000000000000000000100000101
BIT00_0_STATUS_VALID	1
BIT01_0_FALLBACK	0
BIT02_0_INTERNAL_PROG	1
BIT03_0_WATCHDOG_TIMEOUT_ERROR	0
BIT04_0_ID_ERROR	0
BIT05_0_CRC_ERROR	0
BIT06_0_WRAP_ERROR	0
BIT07_0_SECURITY_ERROR	0
BIT08_1_STATUS_VALID	1
BIT09_1_FALLBACK	0
BIT10_1_INTERNAL_PROG	0
BIT11_1_WATCHDOG_TIMEOUT_ERROR	0
BIT12_1_ID_ERROR	0
BIT13_1_CRC_ERROR	0
BIT14_1_WRAP_ERROR	0
BIT15_1_SECURITY_ERROR	0
BIT16_RESERVED	0000000000000000
> CONFIG_STATUS	00010000101100000111100111111100

**Figure 16: MultiBoot Image BOOT\_STATUS**

After confirming that the MultiBoot image is successfully loaded, setting the DIP SW13[1] to 1 causes the design to read the FPGA IDCODE and display it via the UART, as shown in [Figure 17](#).

FPGA IDCODE is : 4A62093

**Figure 17: MultiBoot Image FPGA IDCODE Read UART Display**

## Fallback Example – CRC Error

To boot the FPGA with the image (Golden\_n\_Multiboot\_CRC\_Err.mcs) programmed into the flash device, pulse PROGRAM\_B by pulsing SW5 on the KCU116 board. After confirming that the golden image is successfully loaded, toggle the DIP SW13[1] 0 > 1 > 0. The DONE LED turns off for a moment and the FPGA tries to load the MultiBoot image. Because the MultiBoot image is corrupted, the FPGA will fallback and load the golden bitstream.

- Refresh the device by right-clicking the FPGA in the Vivado IDE and selecting **Hardware Device Properties**.
- 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 caused the jump and CRC error, and the Fallback flag to go High. The CONFIG\_STATUS register shows the DONE\_PIN is High (Figure 18).

▼ REGISTER	
▼ BOOT_STATUS	0000000000000000000010010100000011
BIT00_0_STATUS_VALID	1
BIT01_0_FALLBACK	1
BIT02_0_INTERNAL_PROG	0
BIT03_0_WATCHDOG_TIMEOUT_ERROR	0
BIT04_0_ID_ERROR	0
BIT05_0_CRC_ERROR	0
BIT06_0_WRAP_ERROR	0
BIT07_0_SECURITY_ERROR	0
BIT08_1_STATUS_VALID	1
BIT09_1_FALLBACK	0
BIT10_1_INTERNAL_PROG	1
BIT11_1_WATCHDOG_TIMEOUT_ERROR	0
BIT12_1_ID_ERROR	0
BIT13_1_CRC_ERROR	1
BIT14_1_WRAP_ERROR	0
BIT15_1_SECURITY_ERROR	0
BIT16_RESERVED	0000000000000000
> CONFIG_STATUS	00010000101100000111100111111100

X17896-050517

Figure 18: CRC Error BOOT\_STATUS



## Fallback Example – IDCODE Error

To boot the FPGA with the image (Golden\_n\_Multiboot\_ID\_Err.mcs) programmed into the flash device pulse PROGRAM\_B by pulsing SW5 on the KCU116 board. After confirming that the golden image is successfully loaded, toggle the DIP SW13[1] 0 > 1 > 0. The DONE LED turns off and the FPGA tries to load the MultiBoot image. Because the MultiBoot image IDCODE is corrupted intentionally, the FPGA will fallback and load the golden bitstream.

- Refresh the device by right-clicking the FPGA in the Vivado IDE and selecting **Hardware Device Properties**.
- 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 caused the jump and ID error, and the Fallback flag to go High. The CONFIG\_STATUS register shows the DONE\_PIN is High (Figure 19).

▼ REGISTER	
▼ BOOT_STATUS	000000000000000000001010100000011
BIT00_0_STATUS_VALID	1
BIT01_0_FALLBACK	1
BIT02_0_INTERNAL_PROG	0
BIT03_0_WATCHDOG_TIMEOUT_ERROR	0
BIT04_0_ID_ERROR	0
BIT05_0_CRC_ERROR	0
BIT06_0_WRAP_ERROR	0
BIT07_0_SECURITY_ERROR	0
BIT08_1_STATUS_VALID	1
BIT09_1_FALLBACK	0
BIT10_1_INTERNAL_PROG	1
BIT11_1_WATCHDOG_TIMEOUT_ERROR	0
BIT12_1_ID_ERROR	1
BIT13_1_CRC_ERROR	0
BIT14_1_WRAP_ERROR	0
BIT15_1_SECURITY_ERROR	0
BIT16_RESERVED	0000000000000000
> CONFIG_STATUS	00010000100100000111100111111100

X17897-050517

Figure 19: IDCODE Error BOOT\_STATUS

## Fallback Example – Watchdog Timer

To boot the FPGA with the image<sup>(1)</sup> programmed into the flash device, pulse PROGRAM\_B by pulsing SW5 on the KCU116 board.

After confirming that the golden image is successfully loaded, toggle the DIP SW13[4] 0 > 1 > 0. The DONE LED turns off and the FPGA tries to load the MultiBoot image. Because the jump is to address 0x02000000 and no valid configuration image is available, the FPGA watchdog timer will timeout (timeout is approximately 15s) and trigger a fallback. The FPGA will fallback and load the golden bitstream.

- Refresh the device by right-clicking the FPGA in the Vivado IDE and selecting **Hardware Device Properties**.
- 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 caused the jump and watchdog timeout error, and the Fallback flag to go High. The CONFIG\_STATUS register shows the DONE\_PIN is High (Figure 20).

▼ REGISTER	
▼ BOOT_STATUS	000000000000000000000000110100000011
BIT00_0_STATUS_VALID	1
BIT01_0_FALLBACK	1
BIT02_0_INTERNAL_PROG	0
BIT03_0_WATCHDOG_TIMEOUT_ERROR	0
BIT04_0_ID_ERROR	0
BIT05_0_CRC_ERROR	0
BIT06_0_WRAP_ERROR	0
BIT07_0_SECURITY_ERROR	0
BIT08_1_STATUS_VALID	1
BIT09_1_FALLBACK	0
BIT10_1_INTERNAL_PROG	1
BIT11_1_WATCHDOG_TIMEOUT_ERROR	1
BIT12_1_ID_ERROR	0
BIT13_1_CRC_ERROR	0
BIT14_1_WRAP_ERROR	0
BIT15_1_SECURITY_ERROR	0
BIT16_RESERVED	0000000000000000
> CONFIG_STATUS	00010000101100000111100111111100

X17898-050517

Figure 20: Watchdog Timeout Error BOOT\_STATUS

1. Any of the provided MCS files can be used for the image.

# Debug

This section provides a checklist to debug common issues for MultiBoot with SPI flash.

## Design

- All bitstream properties are correctly set for both the golden and MultiBoot images (see [Table 2](#)).
- All SPI bitstream properties are correctly set (see [Table 2](#)).
- The command to generate the flash programming file has all the correct options and address settings. Refer to `Create_MCS.tcl`.
- The IPROG jump address specified in `Golden.c` matches the address specified while re-generating the MCS file.

## Hardware

- The mode pin settings are for master SPI configuration.
- DIPSW13 is set to all 0s initially.
- UART baud rate is set to 9600.
- 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 reference design. Ensure that the refresh device is completed before reading the registers.
- SW16 is mapped to reset of the reference design. Pushing the switch puts the design in reset.
- `PROG_B` SW5 switch is released.

---

# Conclusion

This application note describes how the MultiBoot feature in UltraScale+ FPGAs can be used either for updating systems in the field or for loading different configuration images in real time. 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 provided for the KCU116 board.

## Reference Design

Download the [reference design files](#) for this application note from the Xilinx website. [Table 3](#) shows the reference design matrix.

**Table 3: Reference Design Matrix**

Parameter	Description
<b>General</b>	
Developer name	Guruprasad Kempahonnaiah
Target devices	UltraScale+ FPGAs
Source code provided	Yes
Source code format	Verilog, C
Design uses code and IP from existing Xilinx application note and reference designs or third party	N/A
<b>Simulation</b>	
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
<b>Implementation</b>	
Synthesis software tools/versions used	Vivado Design Suite 2017.1
Implementation software tools/versions used	Vivado Design Suite 2017.1 Xilinx SDK 2017.1
Static timing analysis performed	N/A
<b>Hardware Verification</b>	
Hardware verified	Yes
Hardware platform used for verification	KCU116 evaluation board

## 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. *KCU116 Evaluation Board User Guide* ([UG1239](#))
5. *Embedded System Tools Reference Manual* ([UG1043](#))

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/23/2017	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>.

### **AUTOMOTIVE APPLICATIONS DISCLAIMER**

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2017 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.