KC705 MultiBoot Design

October 2014
## Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/08/14</td>
<td>10.0</td>
<td>Recompiled for 2014.3.</td>
</tr>
<tr>
<td>06/09/14</td>
<td>9.0</td>
<td>Recompiled for 2014.2. AR44635 and AR60411 fixed.</td>
</tr>
<tr>
<td>04/25/14</td>
<td>6.1</td>
<td>Added AR44635 and AR60411.</td>
</tr>
<tr>
<td>04/16/14</td>
<td>6.0</td>
<td>Recompiled for 2014.1.</td>
</tr>
<tr>
<td>12/18/13</td>
<td>5.0</td>
<td>Recompiled for 2013.4.</td>
</tr>
<tr>
<td>10/23/13</td>
<td>4.0</td>
<td>Recompiled for 2013.3. AR58291 added.</td>
</tr>
<tr>
<td>07/01/13</td>
<td>3.1</td>
<td>Minor updates.</td>
</tr>
<tr>
<td>06/19/13</td>
<td>3.0</td>
<td>Recompiled for 2013.2. AR55431 fixed.</td>
</tr>
<tr>
<td>04/03/13</td>
<td>2.0</td>
<td>Recompiled for 2013.1. AR55431 added.</td>
</tr>
<tr>
<td>12/18/12</td>
<td>1.0</td>
<td>Initial version. AR53392 added.</td>
</tr>
</tbody>
</table>

© Copyright 2014 Xilinx, Inc. All Rights Reserved.
XILINX, the Xilinx logo, the Brand Window and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

NOTICE OF DISCLAIMER: The information disclosed to you hereunder (the “Information”) is provided “AS-IS” with no warranty of any kind, express or implied. Xilinx does not assume any liability arising from your use of the Information. You are responsible for obtaining any rights you may require for your use of this Information. Xilinx reserves the right to make changes, at any time, to the Information without notice and at its sole discretion. Xilinx assumes no obligation to correct any errors contained in the Information or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE INFORMATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS.
Overview

- Kintex-7 MultiBoot Capability
- Overview of 7 Series Fallback MultiBoot
- Xilinx KC705 Board
- Software Requirements
- KC705 Setup
- Compiling the MultiBoot Design
- Run MultiBoot Design
- Kintex-7 MultiBoot Details
- References

**Note:** This presentation applies to the KC705
Kintex-7 MultiBoot Capability

What is MultiBoot?

- The MultiBoot feature 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.

- More details can be found in UG470

MultiBoot Capability

- FPGA application controlled configuration
- Bitstream selection of multiple applications

Safe Update

- Golden bitstream
- Upgradeable bitstream
- Failure recovery
  - Possible Triggers (CRC error, IDCODE error, WDT timeout)

Note: Presentation applies to the KC705
Kintex-7 MultiBoot Capability

» Overview of 7 Series Fallback MultiBoot
  - The 7 series FPGAs MultiBoot and fallback features support updating systems in the field. Bitstream images can be upgraded dynamically in the field. 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.
  - The MultiBoot feature 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.

» Reference Design Source and Application
  - KC705 MultiBoot Design Files (2014.3 C) ZIP file
  - Available through http://www.xilinx.com/kc705

Note: Presentation applies to the KC705
KC705 MultiBoot Design Description

Description

- Design consists of three bitstreams, golden_iprog_spi.bit, multiboot.bit, and corrupted.bit loaded in the PROM at 0x0, 0x400000, and 0x800000
- The golden_iprog_spi.bit has software to allow a reboot to the second or third bitstreams with certain switch settings
- The multiboot.bit is compiled with this property set:
  ```
  set_property BITSTREAM.CONFIG.CONFIGFALLBACK Enable [current_design]
  ```
- The corrupted.bit is a multiboot.bit with one byte changed to force a CRC error when loaded

This diagram shows the general process

- Since the corrupted.bit doesn’t pass configuration, the Golden Image is booted

Note: Presentation applies to the KC705
Upgrade Example

- **MultiBoot**: Process by which the FPGA selectively reprograms and reloads its bitstream from an attached external memory.

- **Safe update**: Field updating bitstream storage with a new bitstream in such a manner to prevent any failure due to a failure in the update process. This is accomplished with the enhanced MultiBoot available in Kintex-7.

Can a multi-boot system be implemented without safe update design considerations?

Yes – If there are no potential for disruptions during flash loading.

How is a system upgraded?

1. New multi-boot image is created
2. System setup to receive the new image
3. User application erases section of Flash
4. The new image is delivered into the system’s Flash
5. User application resets system

Note: Presentation applies to the KC705
Xilinx KC705 Board
Vivado Software Requirements

  - Combined installer

Note: Presentation applies to the KC705
Hardware Setup

- Connect a USB Type-A to Micro-B cable to the USB JTAG (Digilent) connector on the KC705 board
  - Connect this cable to your PC
  - Power on the KC705 board
Hardware Setup

➤ Set SW13 to 00001 (1 = on, Position 1 → Position 5)
  - This enables Master SPI configuration from the QSPI Flash
    • Flash A25, A24 = 00
    • FPGA mode pins M[2:0] = 001

Note: Presentation applies to the KC705
Hardware Setup

Set SW11 to 0000 (1 = on, Position 1 → Position 4)
– This sets the MultiBoot functions to off
Program SPI MultiBoot Design

- Unzip the KC705 MultiBoot Design Files (2014.3 C) ZIP file
  - Available through http://www.xilinx.com/kc705

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy_golden_iprog_spi.bat</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>11/7/2013 2:31 PM</td>
</tr>
<tr>
<td>copy_multiboot.bat</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>11/7/2013 2:31 PM</td>
</tr>
<tr>
<td>corrupted.bit</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>6/2/2014 1:32 AM</td>
</tr>
<tr>
<td>golden_iprog_spi.bit</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/10/2014 1:40 AM</td>
</tr>
<tr>
<td>golden_iprog_spi.elf</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/10/2014 1:29 AM</td>
</tr>
<tr>
<td>kc705_multiboot_spi.mcs</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/10/2014 3:43 AM</td>
</tr>
<tr>
<td>kc705_multiboot_spi.bat</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/6/2014 3:20 AM</td>
</tr>
<tr>
<td>kc705_program_spi.tcl</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/6/2014 1:10 PM</td>
</tr>
<tr>
<td>make_download_files.bat</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/6/2014 3:20 AM</td>
</tr>
<tr>
<td>make_download_files.tcl</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/5/2014 9:39 AM</td>
</tr>
<tr>
<td>make_spi.mcs.bat</td>
<td>kc705_multiboot\SPI_Flash\</td>
<td>10/6/2014 3:20 AM</td>
</tr>
</tbody>
</table>

Selected 0 files, 0 bytes Total 1104 files, 96,819KB

Note: Presentation applies to the KC705
Program SPI MultiBoot Design

Open a Vivado Tcl Shell:

Start → All Programs → Xilinx Design Tools → Vivado 2014.3 → Vivado 2014.3 Tcl Shell

Note: Presentation applies to the KC705
Program SPI MultiBoot Design

In the Vivado Tcl Shell type:

```
cd C:/kc705_multiboot/SPI_Flash
source kc705_program_spi.tcl
```

Note: Takes about 10 minutes
Run MultiBoot Design

- Cycle board power
- The initial design, “golden_iprog_spi.bit”, shows a cycling LED pattern on the GPIO LEDs
Run MultiBoot Design

In a Vivado Tcl Shell type:

```
source C:/kc705_multiboot/SPI_Flash/read_boot_status.tcl
```
Run MultiBoot Design

- View the "REGISTER.BOOTT_STATUS.BIT00_0_STATUS_VALID"
- Since this was a regular boot, only "0_STATUS_VALID" is set

```plaintext
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Read-only</th>
<th>Visible</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT00_0_STATUS_VALID</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT01_0_FALLBACK</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT02_0_INTERNAL_PROG</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT03_0_WATCHDOG_TIMEOUT_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT04_0_ID_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT05_0_CRC_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT06_0_WRAP_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT07_RESERVED</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT08_1_STATUS_VALID</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT09_1_FALLBACK</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT10_1_INTERNAL_PROG</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT11_1_WATCHDOG_TIMEOUT_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT12_1_ID_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT13_1_CRC_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT14_1_WRAP_ERROR</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER.BOOTT_STATUS.BIT15_RESERVED</td>
<td>string</td>
<td>true</td>
<td>true</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Run MultiBoot Design

- Set SW11 to 0001 (1 = on, Position 1 → Position 4)
  - Issues an IPROG command to re-configure from a good MultiBoot bitstream
  - Set it back to 0000 when the DONE LED goes out
Run MultiBoot Design

The MultiBoot design, “multiboot.bit”, shows a rapid blinking LED pattern on the GPIO LEDs.
Run MultiBoot Design

- The script adds a procedure “refresh_boot_status”
- Type refresh_boot_status and press Enter
Run MultiBoot Design

The following status register messages are now set:

- 0_INTERNAL_PROG and 1_STATUS_VALID
Run MultiBoot Design

» Press PROG to return to the golden bitstream
» Wait until you see the cycling pattern on the LEDs
Run MultiBoot Design

- Set SW11 to 1000 (1 = on, Position 1 → Position 4)
  - Issues an IPROG command to re-configure from a corrupted MultiBoot bitstream
  - Set it back to 0000 when the DONE LED goes out
Run MultiBoot Design

- The corrupted.bit generates a CRC error and cannot load the FPGA
- The FPGA falls back to the golden bitstream with a cycling LED pattern
Run MultiBoot Design

➢ Type `refresh_boot_status` and press Enter
Run MultiBoot Design

The following status register messages are now seen:

- 0_INTERNAL_PROG is no longer active
- 0_FALLBACK, 1_INTERNAL_IPROG, and 1_CRC_ERROR are now active
Compile KC705 MultiBoot Design
Compile KC705 MultiBoot Design

» Open Vivado
Start → All Programs → Xilinx Design Tools → Vivado 2014.3 → Vivado

» Select Open Project

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

Open the KC705 Design:

- `<Design Name>/kc705_multiboot.xpr`

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

The design is fully implemented; you can recompile, or export to SDK

- To recompile, right-click `synth_1`, select **Reset Runs** then **Generate Bitstream**

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

Once done, both the Synthesis and Implementation will have green check marks

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

The MultiBoot Design has been implemented with IP Integrator (IPI)

Click Open Block Design

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

- All the IP Blocks used in the design can be seen in this view
- Click Open Implemented Design

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

View Implemented Design

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

» Select File → Export → Export Hardware

» Click OK

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

- Select File → Launch SDK
- Click OK
Compile KC705 Software in SDK

SDK Software Compile - Build ELF files in SDK

- Project builds automatically
- When done, close SDK and return to Vivado

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

- With Vivado, a special flow is used for compressed embedded designs
  - SDK can only program an ELF file into an uncompressed bitstream
  - Compression is used here to fit the three MultiBoot bitstreams into a smaller area for programming the SPI flash

- After SDK has compiled the ELF files, they must be associated with the design
  - To do this, we will use the **Add Sources** command, followed by the **Associate ELF** command

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

Select Add Sources
Compile KC705 MultiBoot Design

Select Add or Create Design Sources

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

➢ Add golden_iprog_spi.elf and multiboot.elf from the SDK tree
➢ Make sure Copy sources into project is deselected
   – This eliminates the need to re-Associate the ELF each time it is recompiled
➢ Click Finish

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

Right-click on one of the ELF files and select Associate ELF files
Compile KC705 MultiBoot Design

➢ Click the button to the right; select the multiboot.elf then click OK twice
Compile KC705 MultiBoot Design

Select Generate Bitstream

Note: Presentation applies to the KC705
Once the write_bitstream command finishes, open a Windows Prompt and enter these commands:

```
   cd C:\kc705_multiboot\SPI_Flash
   copy_multiboot.bat
```

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

- Now associate the second ELF file
- Right-click one of the ELF files and select Associate ELF files

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

› Click the button to the right of the multiboot.elf; select the golden_iprog_spi.elf then click OK twice

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

Select Generate Bitstream

Note: Presentation applies to the KC705
Compile KC705 MultiBoot Design

From the Command Prompt type:

copy_golden_iprog_spi.bat

Note: Presentation applies to the KC705
Generate KC705 MultiBoot PROM File

From the Command Prompt type:

`make_spi_mcs.bat`

*Note:* Presentation applies to the KC705
Kintex-7 MultiBoot Details
**MultiBoot Register Setup & Command**

» Below are the values programmed into the ICAP via IPROG
  – This table is from UG470, 7 Series FPGAs Configuration User Guide

---

<table>
<thead>
<tr>
<th>Configuration Data (hex)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFFFFFF</td>
<td>Dummy Word</td>
</tr>
<tr>
<td>AA995566</td>
<td>Sync Word</td>
</tr>
<tr>
<td>20000000</td>
<td>Type 1 NO OP</td>
</tr>
<tr>
<td>30020001</td>
<td>Type 1 Write 1 Words to WBSTAR</td>
</tr>
<tr>
<td>00000000</td>
<td>Warm Boot Start Address (Load the Desired Address)</td>
</tr>
<tr>
<td>30008001</td>
<td>Type 1 Write 1 Words to CMD</td>
</tr>
<tr>
<td>0000000F</td>
<td>IPROG Command</td>
</tr>
<tr>
<td>20000000</td>
<td>Type 1 NO OP</td>
</tr>
</tbody>
</table>

---

**Note:** Presentation applies to the KC705
MultiBoot Register Setup & Command

- IPROG command issued via the software in xhwicap_low_level_example.c

```
// Reusing IDCODE array and code to send IPROG
// From UG470 p. 126, Table 7-1
static u32 ReadId[HWICAP_EXAMPLE_BITSTREAM_LENGTH] = {
    0xFFFFFFFF, /* Dummy Word */
    0xAA995566, /* Sync Word*/
    0x20000000, /* Type 1 NO OP */
    0x30020001, /* Write WBSTAR cmd */
    0x00400000, /* Addr in SPI Flash of Multiboot bitstream */
    0x30008001, /* Write CMD */
    0x0000000F, /* Write IPROG */
    0x20000000, /* Type 1 NO OP */
};
```
References
References

IP Integrator Documentation

– Vivado Design Suite Tcl Command Reference Guide

– Designing IP Subsystems Using IP Integrator

7 Series Configuration

– 7 Series FPGAs Configuration User Guide
Documentation
Documentation

Kintex-7

- Kintex-7 FPGA Family
- Design Advisory Master Answer Record for Kintex-7 FPGAs

KC705 Documentation

- Kintex-7 FPGA KC705 Evaluation Kit
- KC705 Getting Started Guide
- KC705 User Guide