



AC701 MultiBoot Design

April 2015

XTP226

Revision History

Date	Version	Description
04/30/14	11.0	Recompiled for 2015.1.
11/24/14	10.0	Recompiled for 2014.4.
10/08/14	9.0	Recompiled for 2014.3.
06/09/14	8.0	Recompiled for 2014.2.
04/16/14	4.0	Recompiled for 2014.1.
12/18/13	3.0	Recompiled for 2013.4.
10/23/13	2.0	Recompiled for 2013.3. Added AR58291.
06/19/13	1.0	Initial version.

© Copyright 2015 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

- Artix-7 MultiBoot Capability
- Overview of 7 Series Fallback MultiBoot
- Xilinx AC701 Board
- Software Requirements
- AC701 Setup
- Compiling the MultiBoot Design
- Run MultiBoot Design
- Artix-7 MultiBoot Details
- References

Artix-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)

Artix-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

- AC701 MultiBoot Design Files (2015.1 C) ZIP file
- Available through <http://www.xilinx.com/ac701>

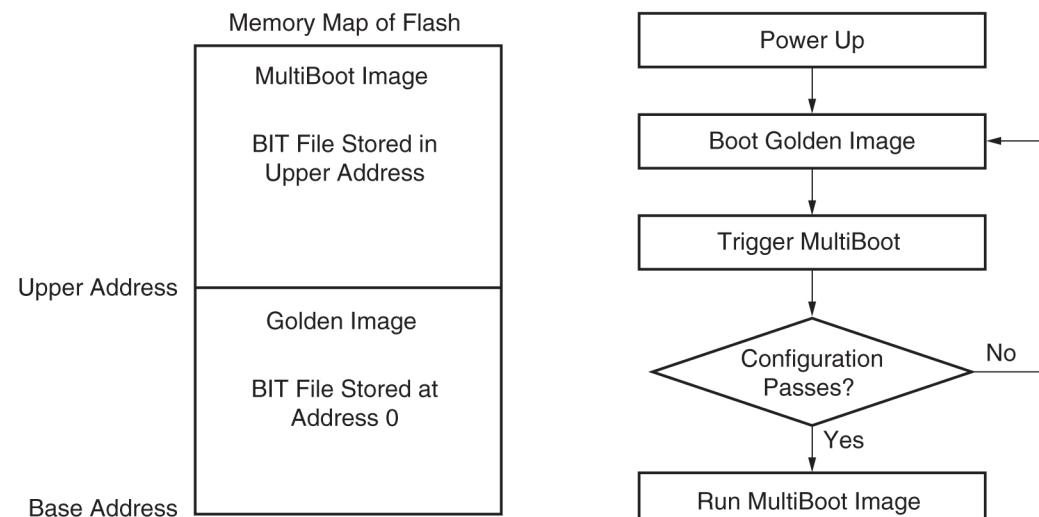
AC701 MultiBoot Design Description

► 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



Upgrade Example

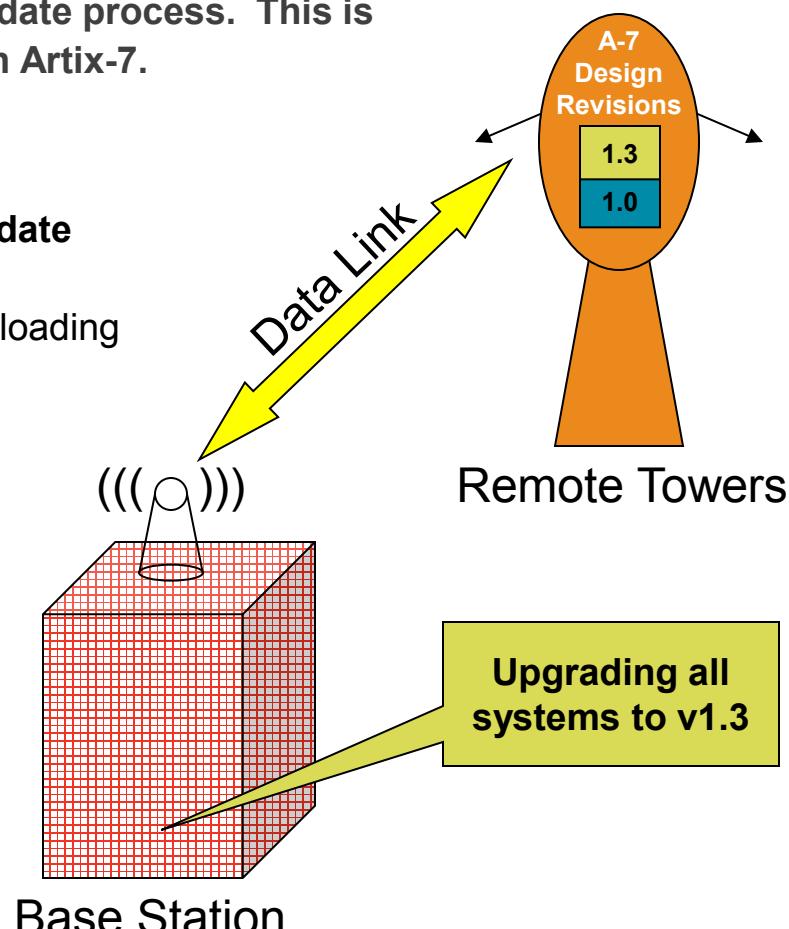
- **MultiBoot:** Process by which the FPGA selectively reprograms and reloads its bitstream from an attached external memory.
- **Safe update:** Field updating bitstream storage with a new bitstream in such a manner to prevent any failure due to a failure in the update process. This is accomplished with the enhanced MultiBoot available in Artix-7.

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

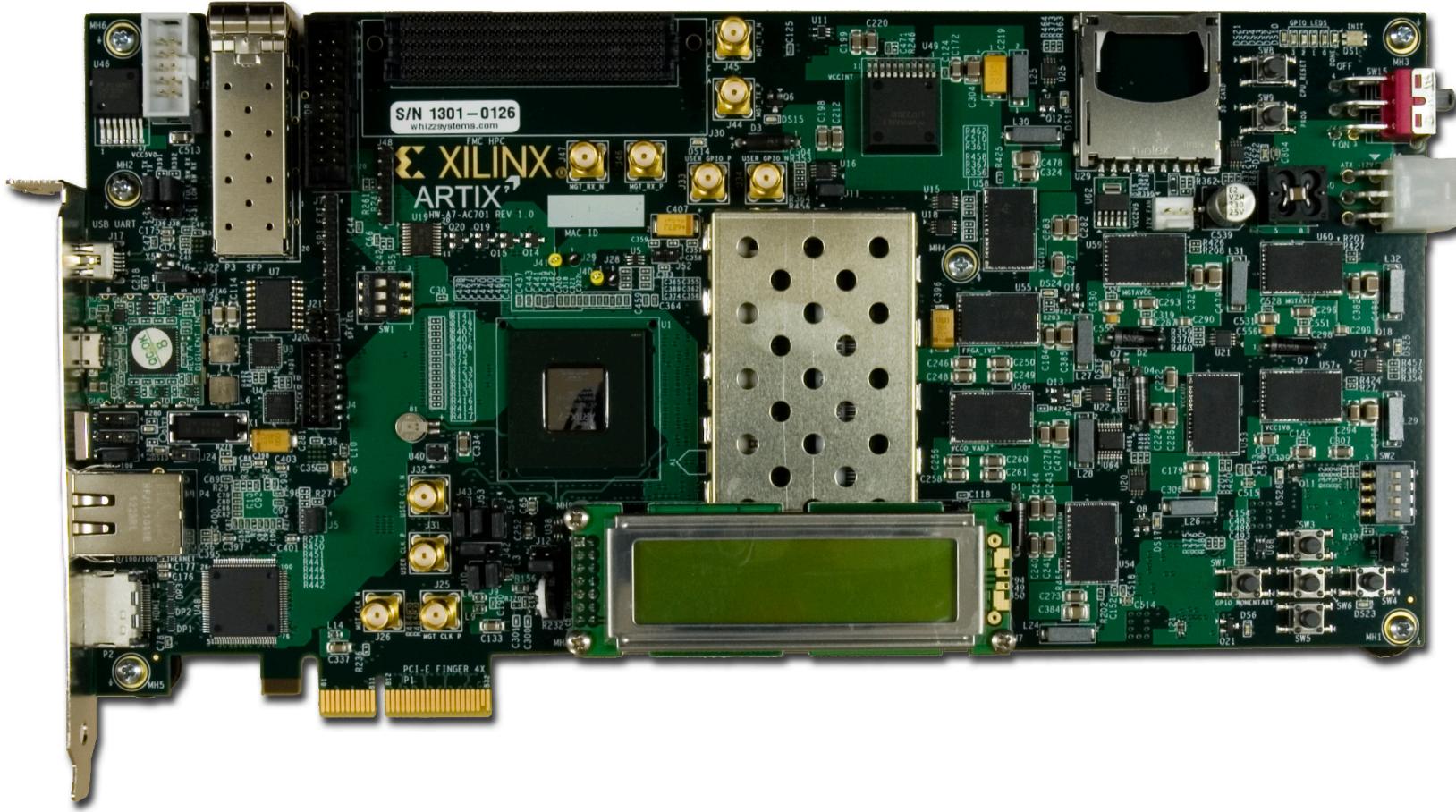
Yes – If there are no potential for disruptions during flash loading

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

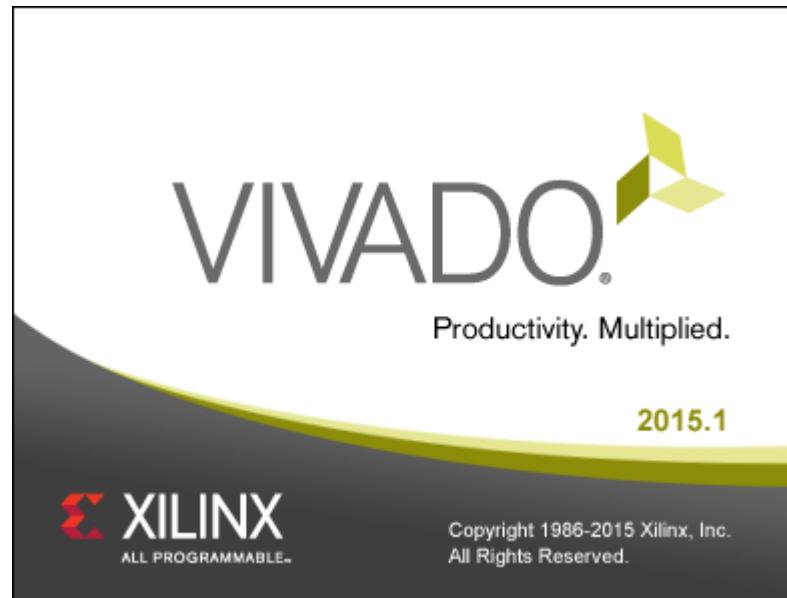


Xilinx AC701 Board



Vivado Software Requirements

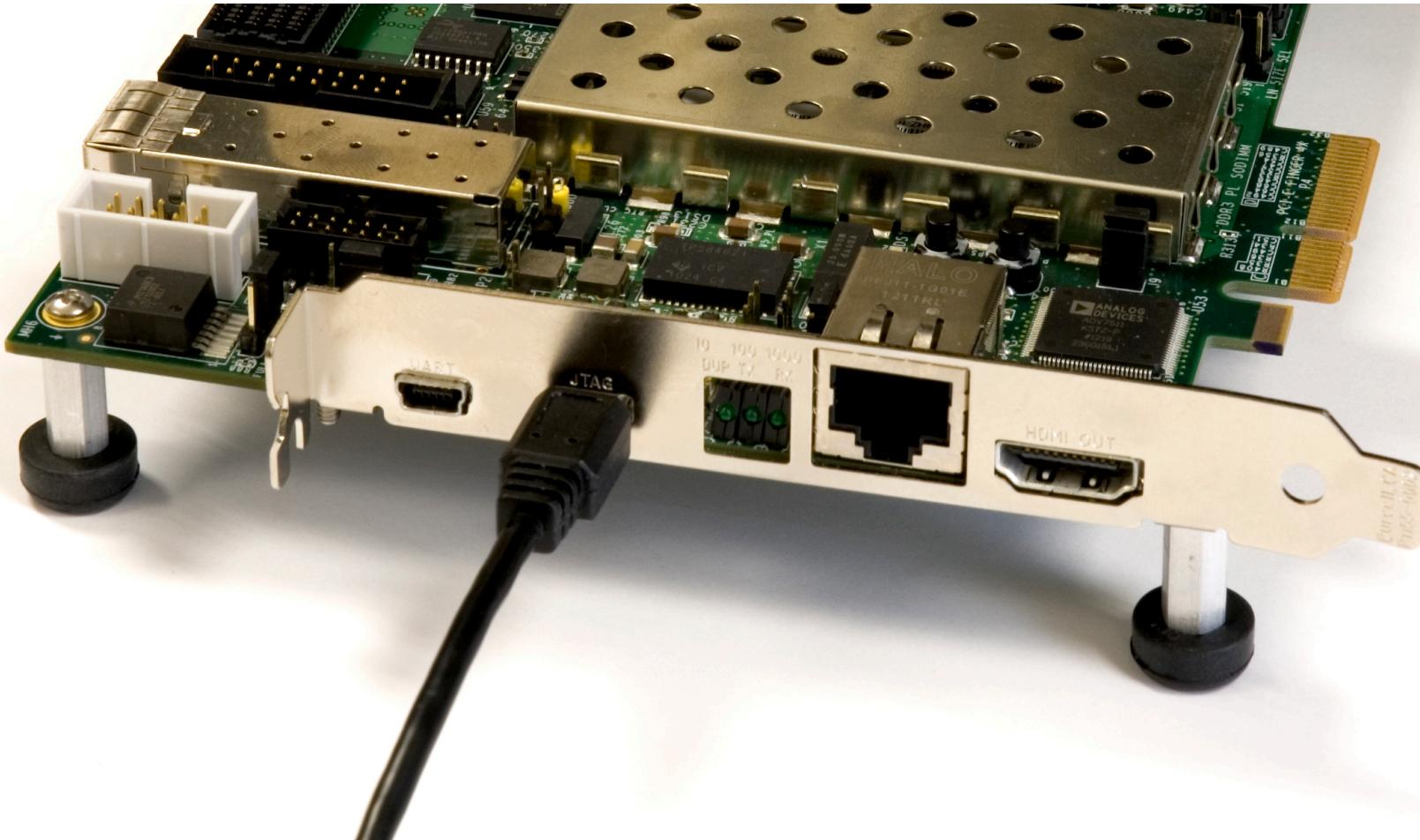
- Xilinx Vivado Design Suite 2015.1, Design Edition + SDK



Hardware Setup

► Connect a USB Type-A to Micro-B cable to the USB JTAG (Digilent) connector on the AC701 board

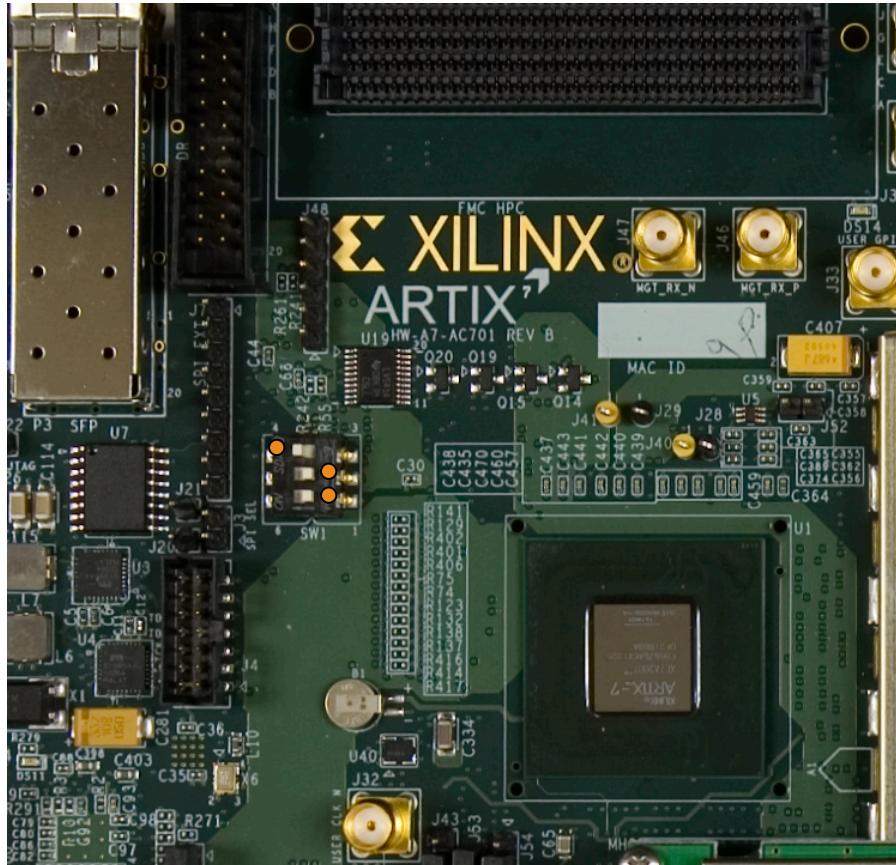
- Connect this cable to your PC
- Power on the AC701 board



Hardware Setup

► Set S1 to 001 (1 = on, Position 1 → Position 3)

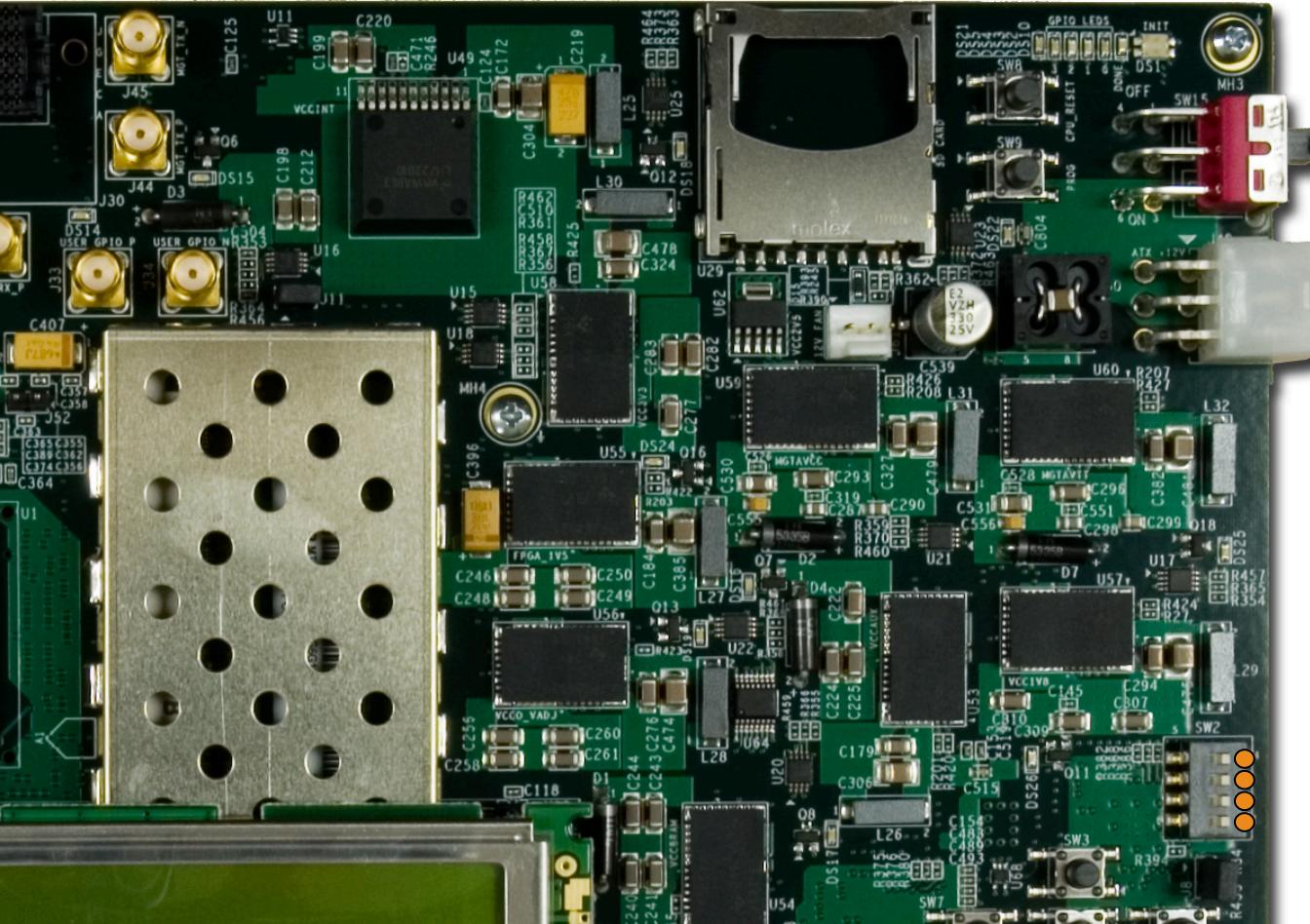
- This enables Master SPI configuration from the N25Q SPI Flash
- FPGA mode pins M[2:0] = 001



Hardware Setup

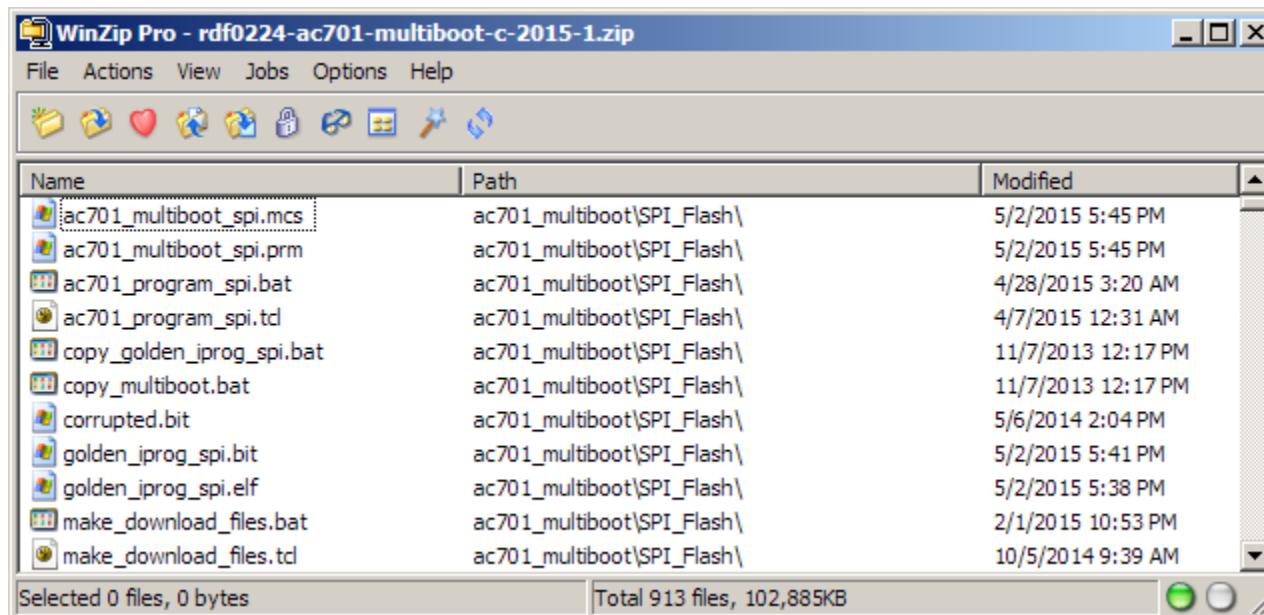
► Set S11 to 0000 (1 = on, Position 1 → Position 4)

- This sets the MultiBoot functions to off



Program SPI MultiBoot Design

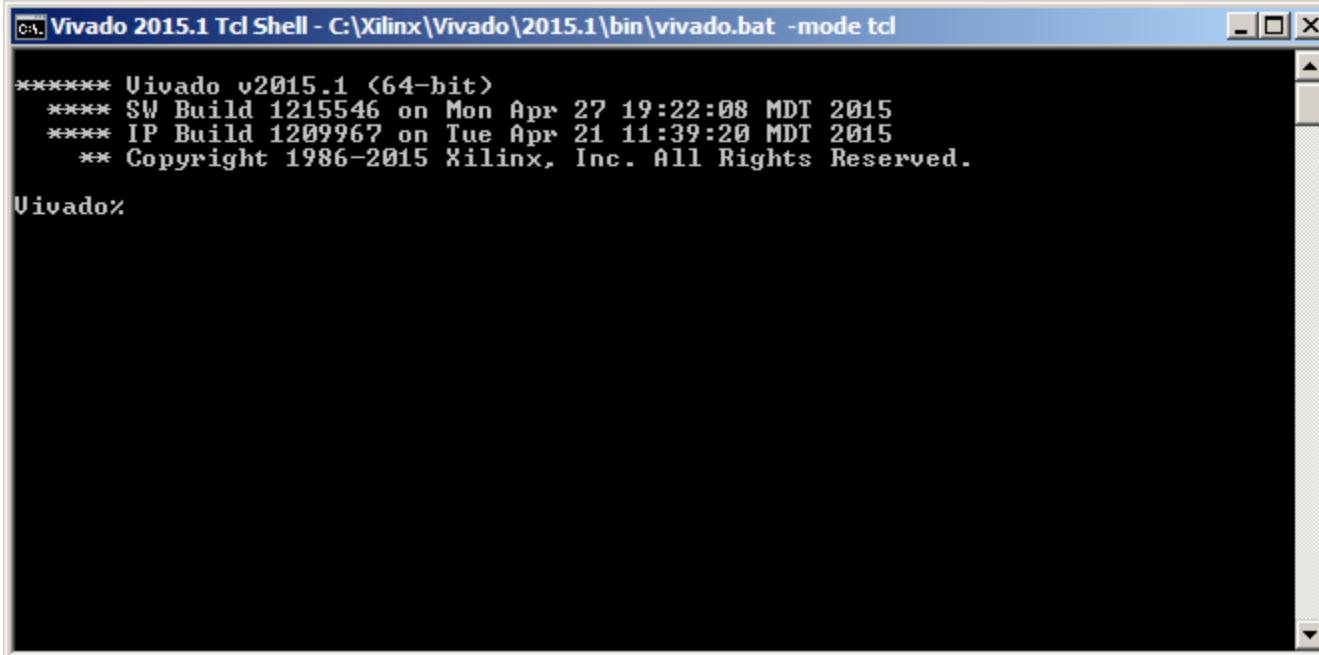
- Unzip the AC701 MultiBoot Design Files (2015.1 C) ZIP file
 - Available through <http://www.xilinx.com/ac701>



Program SPI MultiBoot Design

► Open a Vivado Tcl Shell:

**Start → All Programs → Xilinx Design Tools → Vivado 2015.1 →
Vivado 2015.1 Tcl Shell**

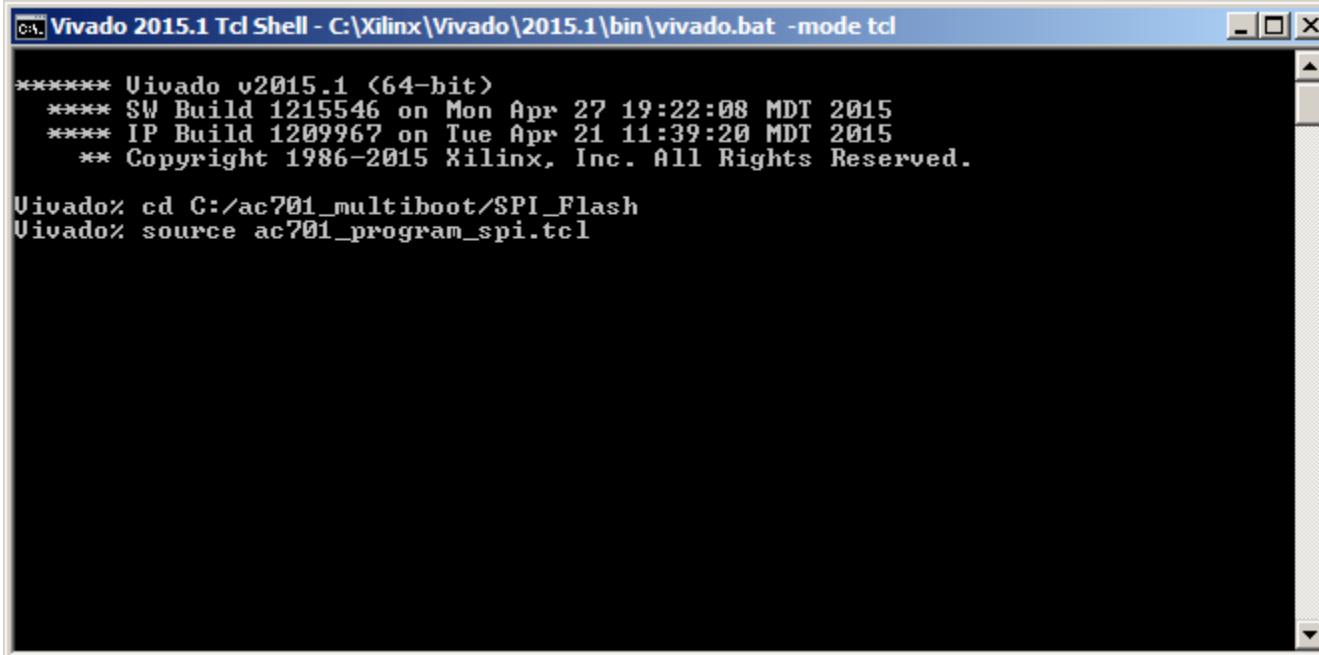


The screenshot shows a Windows command-line interface window titled "Vivado 2015.1 Tcl Shell - C:\Xilinx\Vivado\2015.1\bin\vivado.bat -mode tcl". The window displays the following text:
***** Vivado v2015.1 (64-bit)
**** SW Build 1215546 on Mon Apr 27 19:22:08 MDT 2015
**** IP Build 1209967 on Tue Apr 21 11:39:20 MDT 2015
** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.
Vivado%

Program SPI MultiBoot Design

- In the Vivado Tcl Shell type:

```
cd C:/ac701_multiboot/SPI_Flash  
source ac701_program_spi.tcl
```



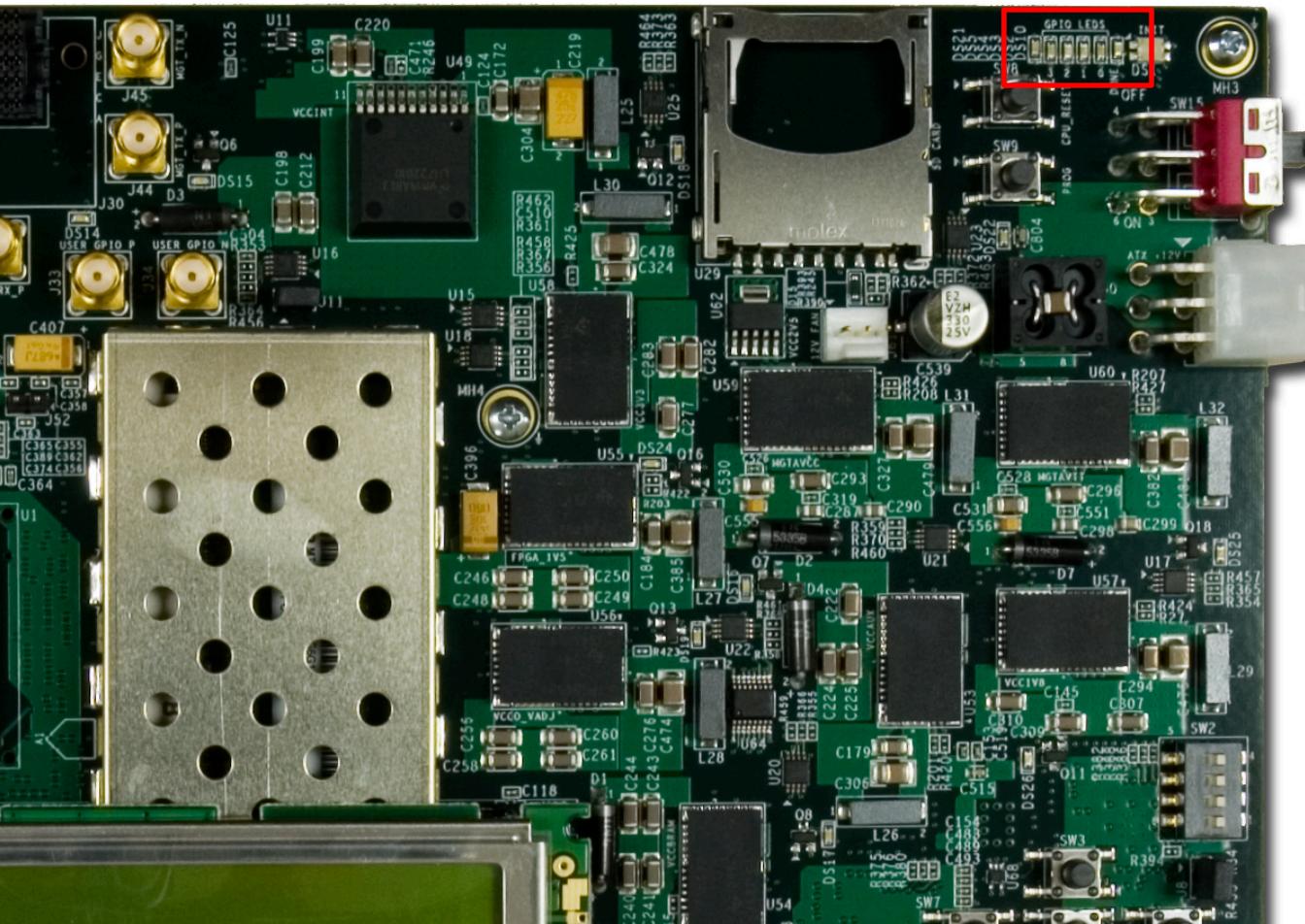
The screenshot shows a Windows command-line interface window titled "Vivado 2015.1 Tcl Shell - C:\Xilinx\Vivado\2015.1\bin\vivado.bat -mode tcl". The window displays the following text:

```
***** Vivado v2015.1 (64-bit)
***** SW Build 1215546 on Mon Apr 27 19:22:08 MDT 2015
***** IP Build 1209967 on Tue Apr 21 11:39:20 MDT 2015
** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.

Vivado> cd C:/ac701_multiboot/SPI_Flash
Vivado> source ac701_program_spi.tcl
```

Run MultiBoot Design

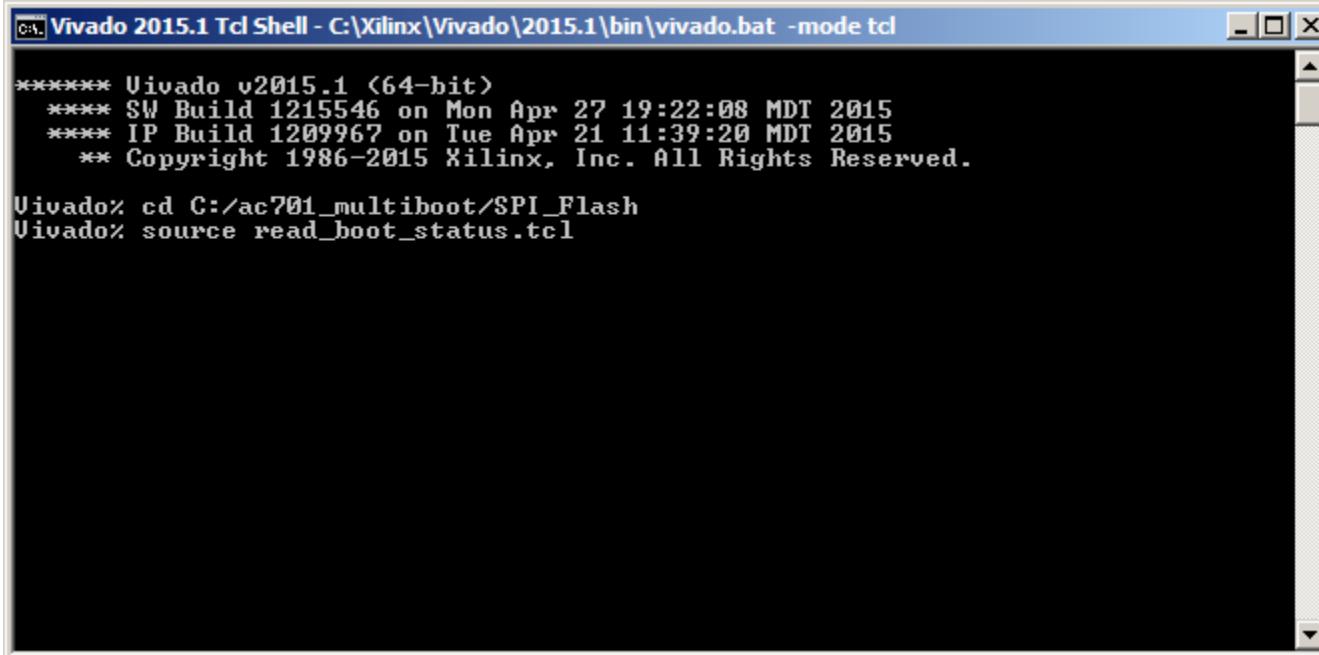
- Cycle board power
- The initial design, “golden_iprog_spi.bit”, shows a cycling LED pattern on the GPIO LEDs



Run MultiBoot Design

- In a Vivado Tcl Shell type:

```
cd C:/ac701_multiboot/SPI_Flash  
source read_boot_status.tcl
```



The screenshot shows a Windows command-line interface window titled "Vivado 2015.1 Tcl Shell - C:\Xilinx\Vivado\2015.1\bin\vivado.bat -mode tcl". The window displays the following text:

```
***** Vivado v2015.1 (64-bit)
***** SW Build 1215546 on Mon Apr 27 19:22:08 MDT 2015
***** IP Build 1209967 on Tue Apr 21 11:39:20 MDT 2015
** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.

Vivado> cd C:/ac701_multiboot/SPI_Flash
Vivado> source read_boot_status.tcl
```

Run MultiBoot Design

- View the “REGISTER.BOOT_STATUS.BIT00_0_STATUS_VALID”
- Since this was a regular boot, only “0_STATUS_VALID” is set

The screenshot shows the Vivado 2015.1 interface with the "Hardware Manager" window open. The "Tcl Console" tab is selected, displaying a table of boot status register properties. The first row, "REGISTER.BOOT_STATUS.BIT00_0_STATUS_VALID", is highlighted with a red box, indicating it is the value being checked.

Property	Type	Read-only	Value
REGISTER.BOOT_STATUS.BIT00_0_STATUS_VALID	string	true	1
REGISTER.BOOT_STATUS.BIT01_0_FALLBACK	string	true	0
REGISTER.BOOT_STATUS.BIT02_0_INTERNAL_PROG	string	true	0
REGISTER.BOOT_STATUS.BIT03_0_WATCHDOG_TIMEOUT_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT04_0_ID_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT05_0_CRC_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT06_0_WRAP_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT07_0_SECURITY_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT08_1_STATUS_VALID	string	true	0
REGISTER.BOOT_STATUS.BIT09_1_FALLBACK	string	true	0
REGISTER.BOOT_STATUS.BIT10_1_INTERNAL_PROG	string	true	0
REGISTER.BOOT_STATUS.BIT11_1_WATCHDOG_TIMEOUT_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT12_1_ID_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT13_1_CRC_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT14_1_WRAP_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT15_1_SECURITY_ERROR	string	true	0

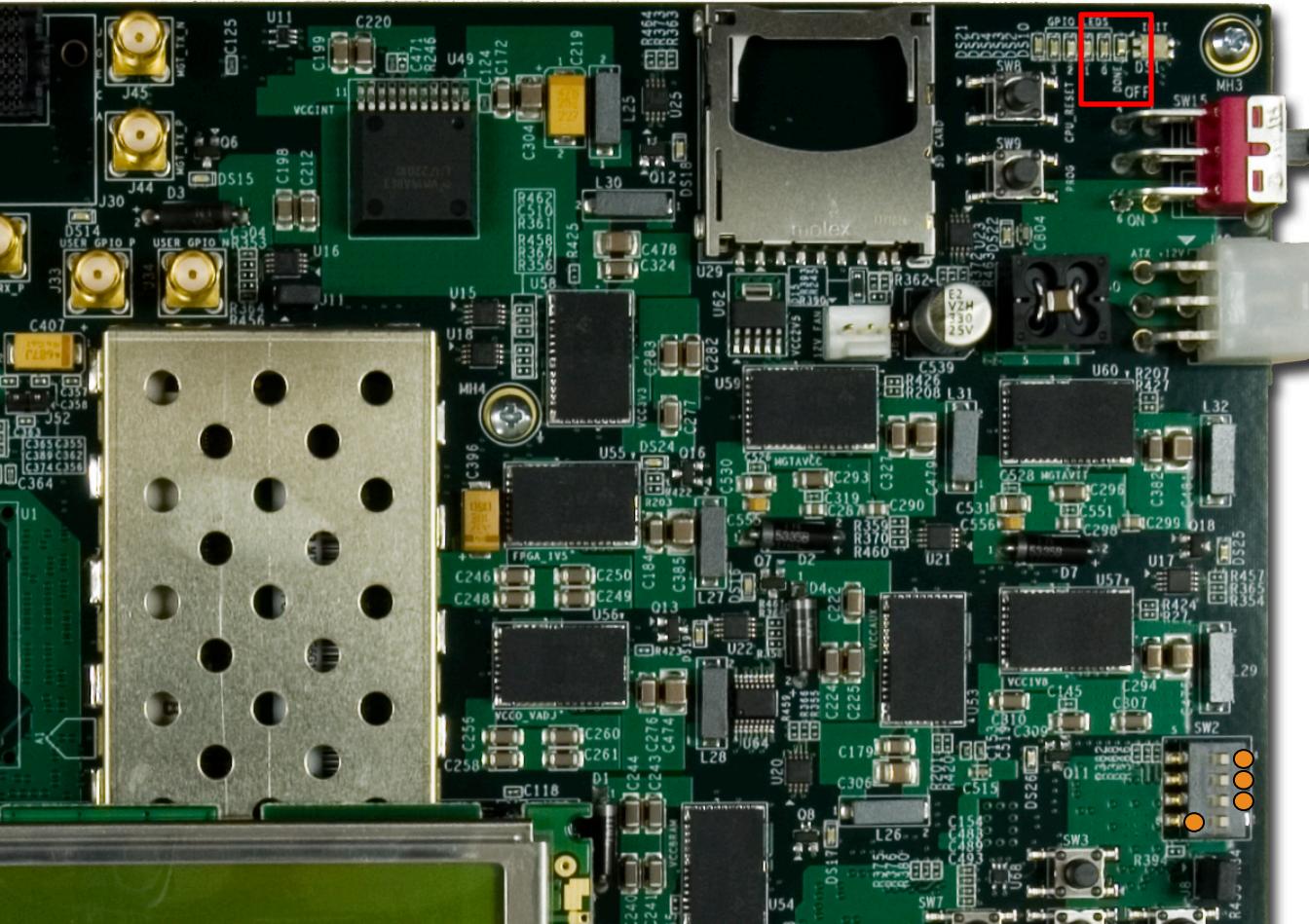
Type a Tcl command here

Tcl Console Messages

Run MultiBoot Design

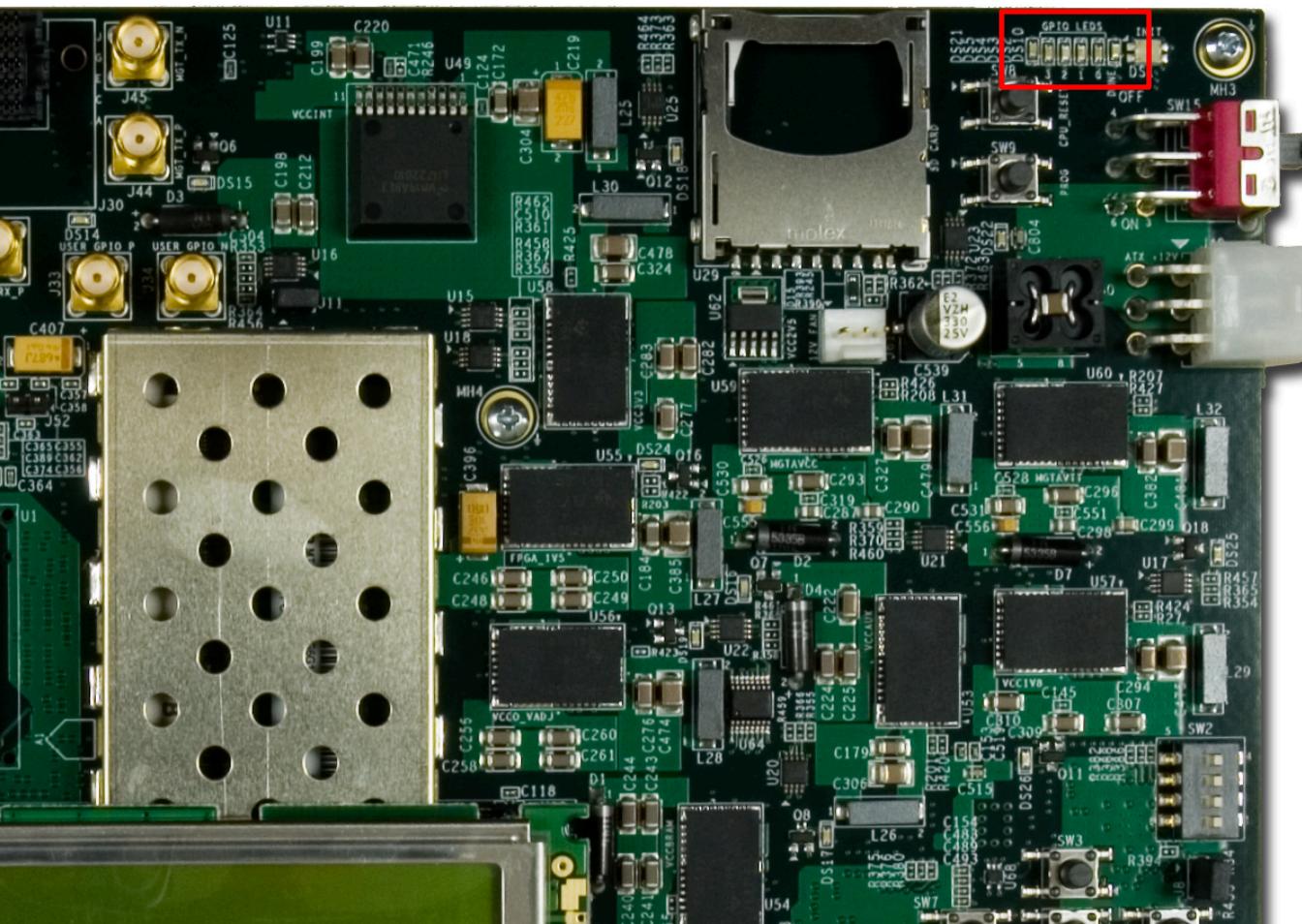
► Set S2 to 1000 (1 = on, Position 1 → Position 4)

- Issues an IPROG command to re-configure from a good MultiBoot bitstream
- Set it back to 0000 when the DONE LED goes out



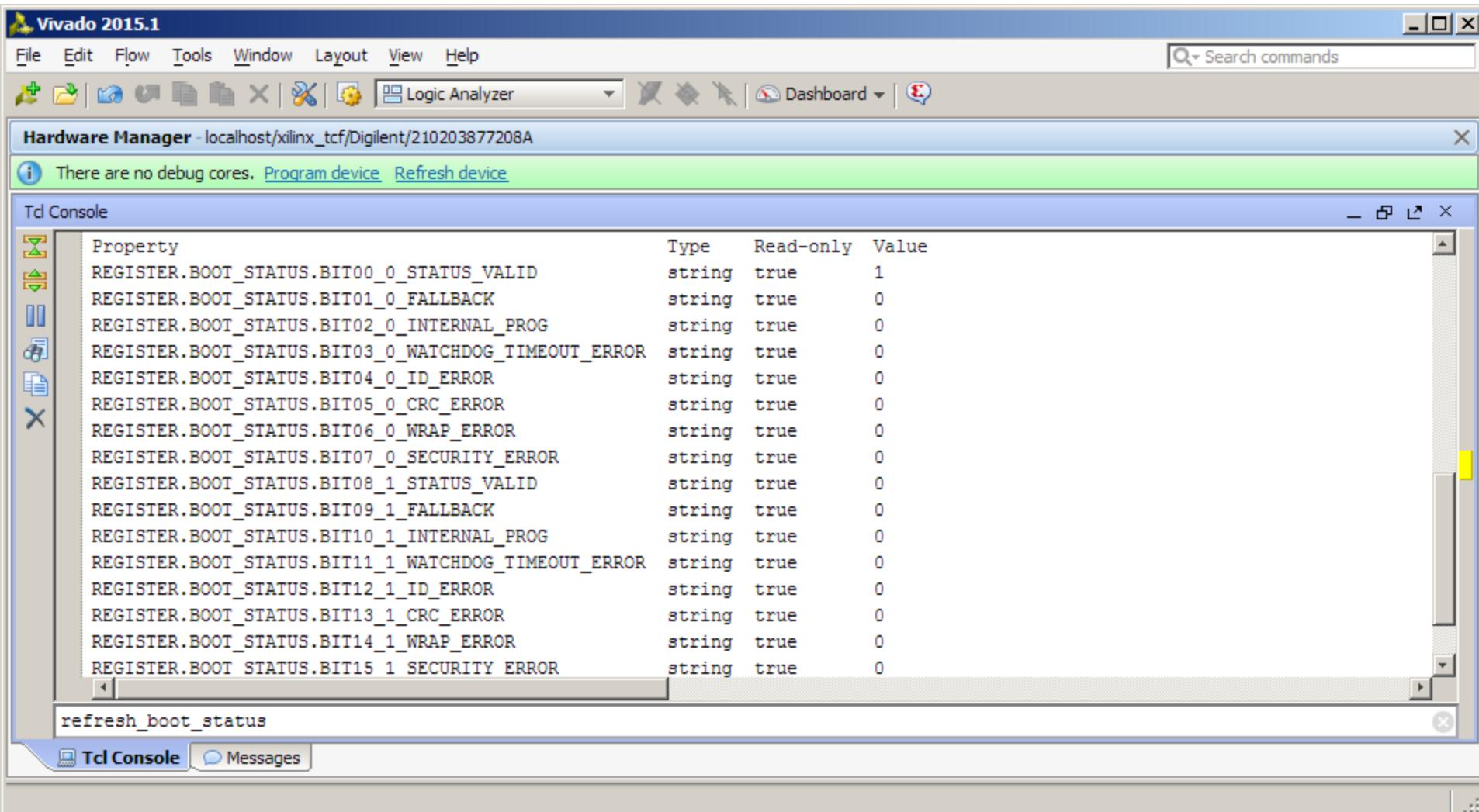
Run MultiBoot Design

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



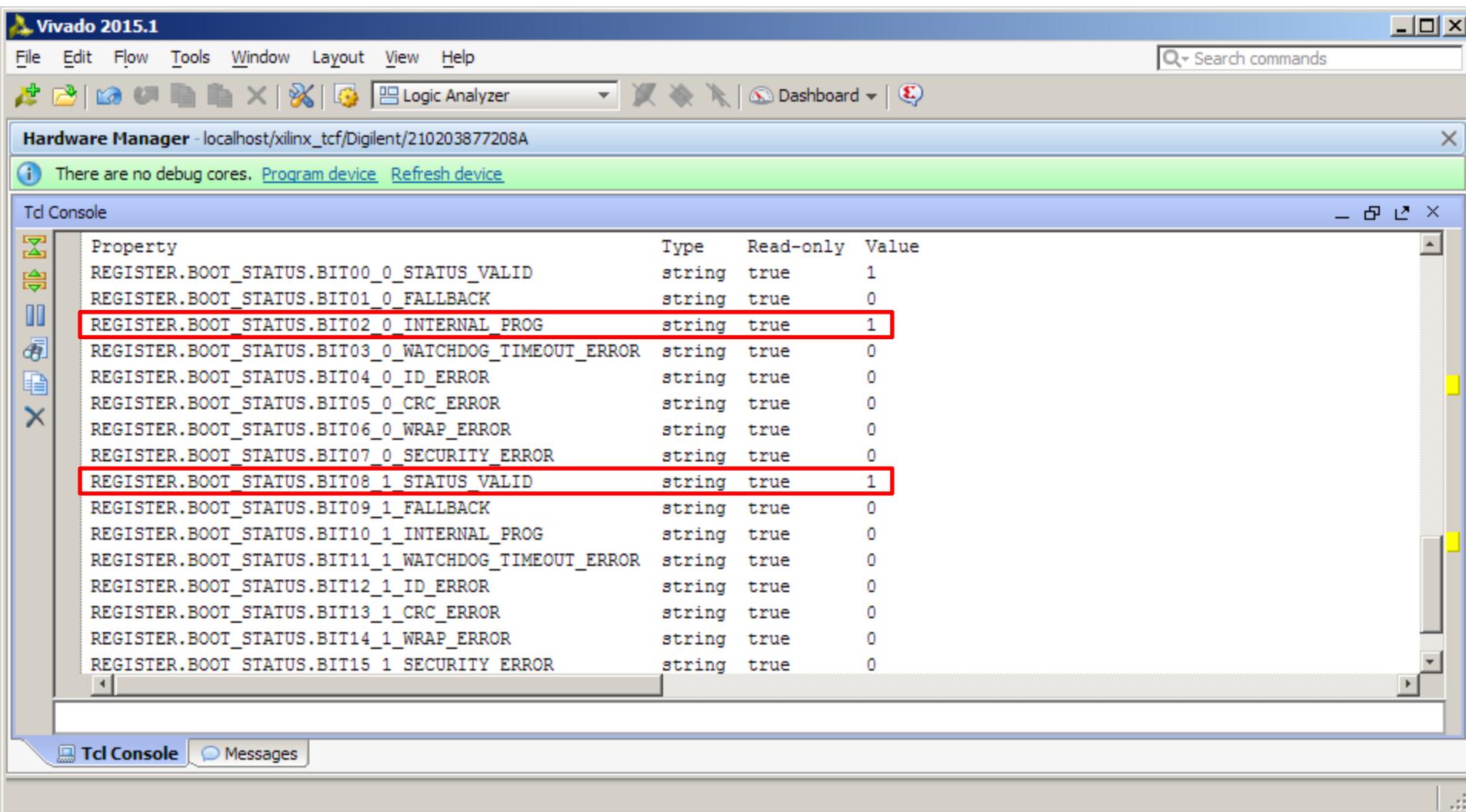
Run MultiBoot Design

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



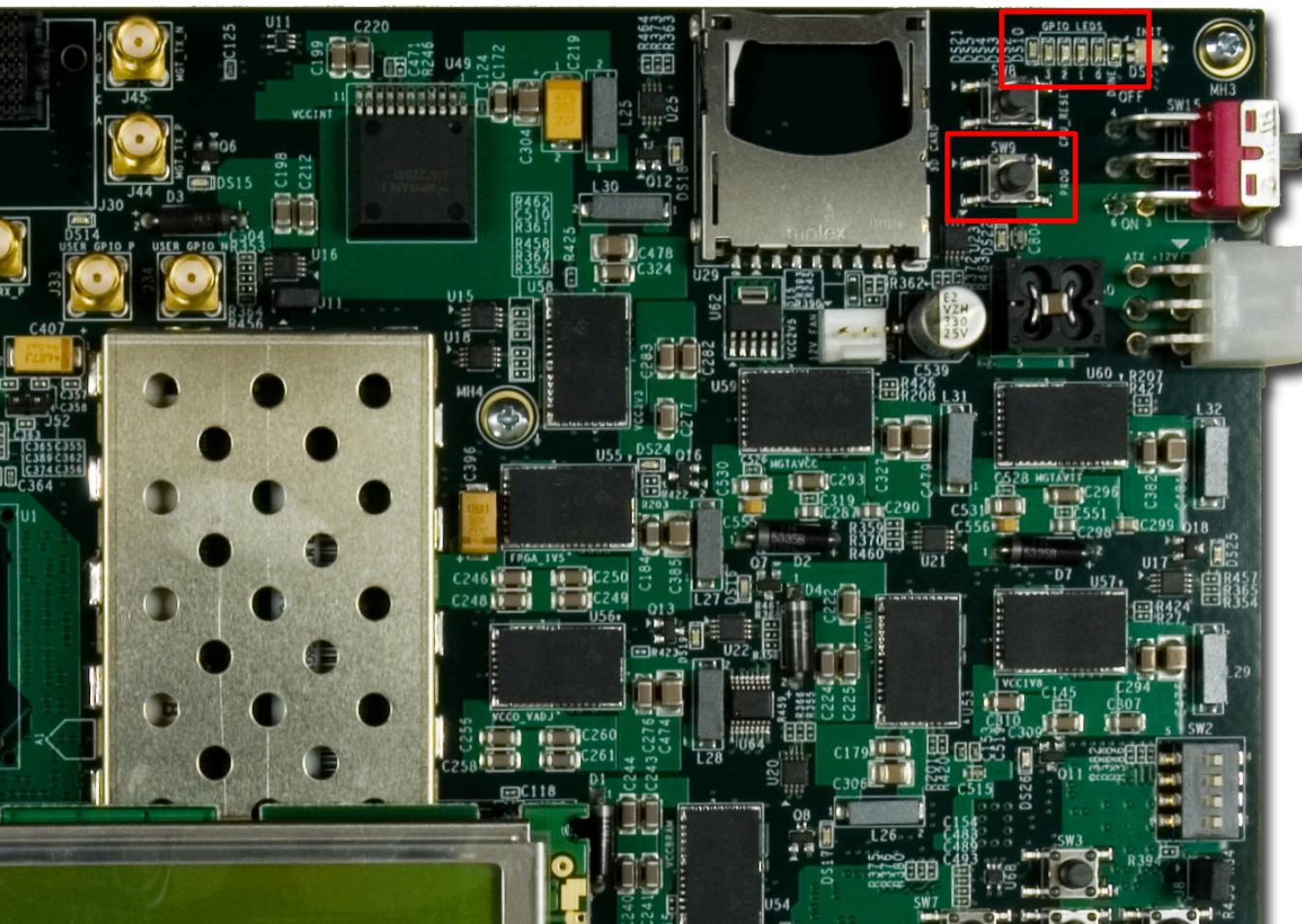
Run MultiBoot Design

- The following status register messages are now set:
 - INTERNAL_PROG and 1_STATUS_VALID



Run MultiBoot Design

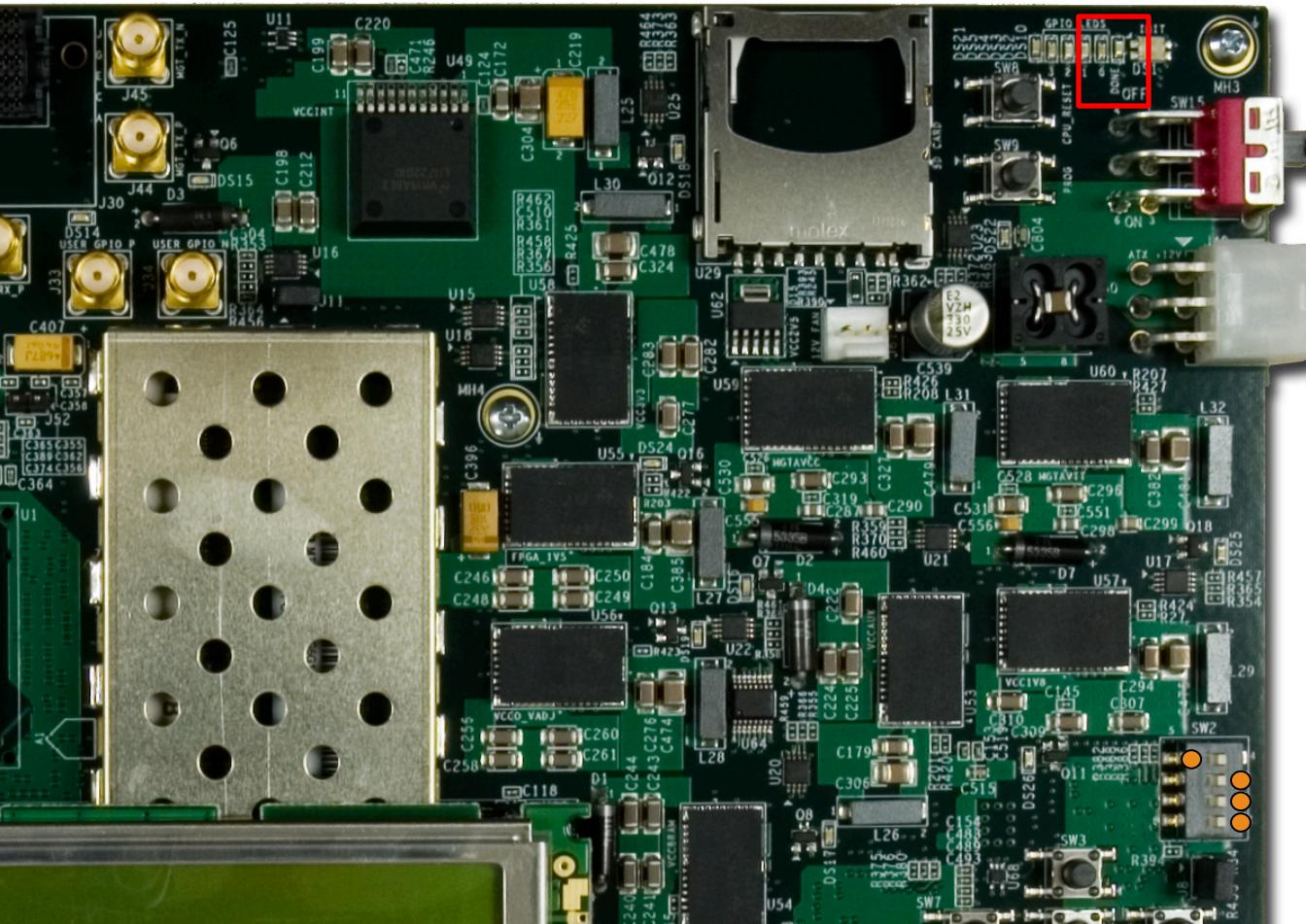
- Press PROG to return to the golden bitstream
- Wait until you see the cycling pattern on the LEDs



Run MultiBoot Design

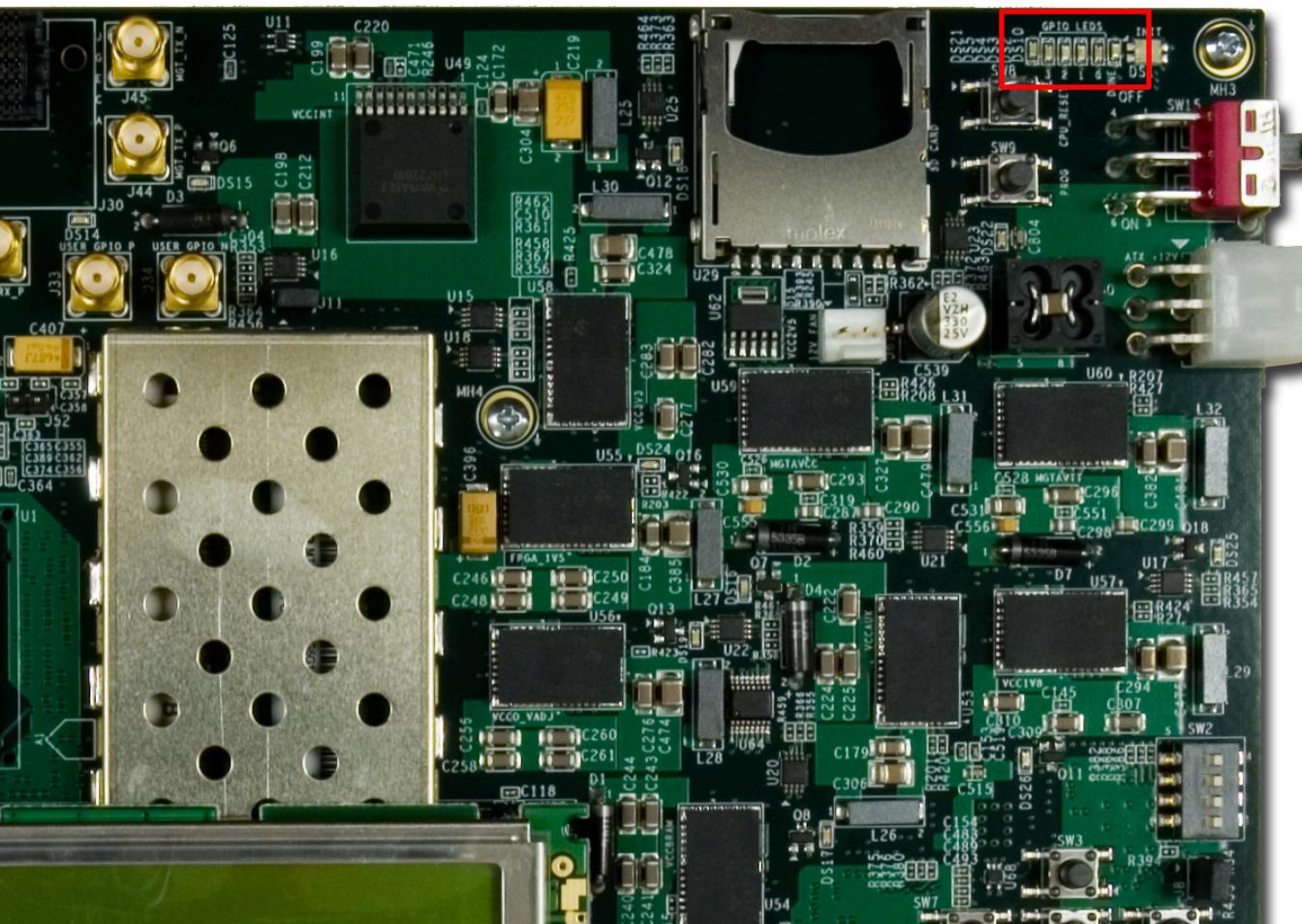
► Set S2 to 0001 (1 = on, Position 1 → Position 4)

- Issues an IPROG command to re-configure from a corrupted MultiBoot bitstream
- Set it back to 0000 when the DONE LED goes out



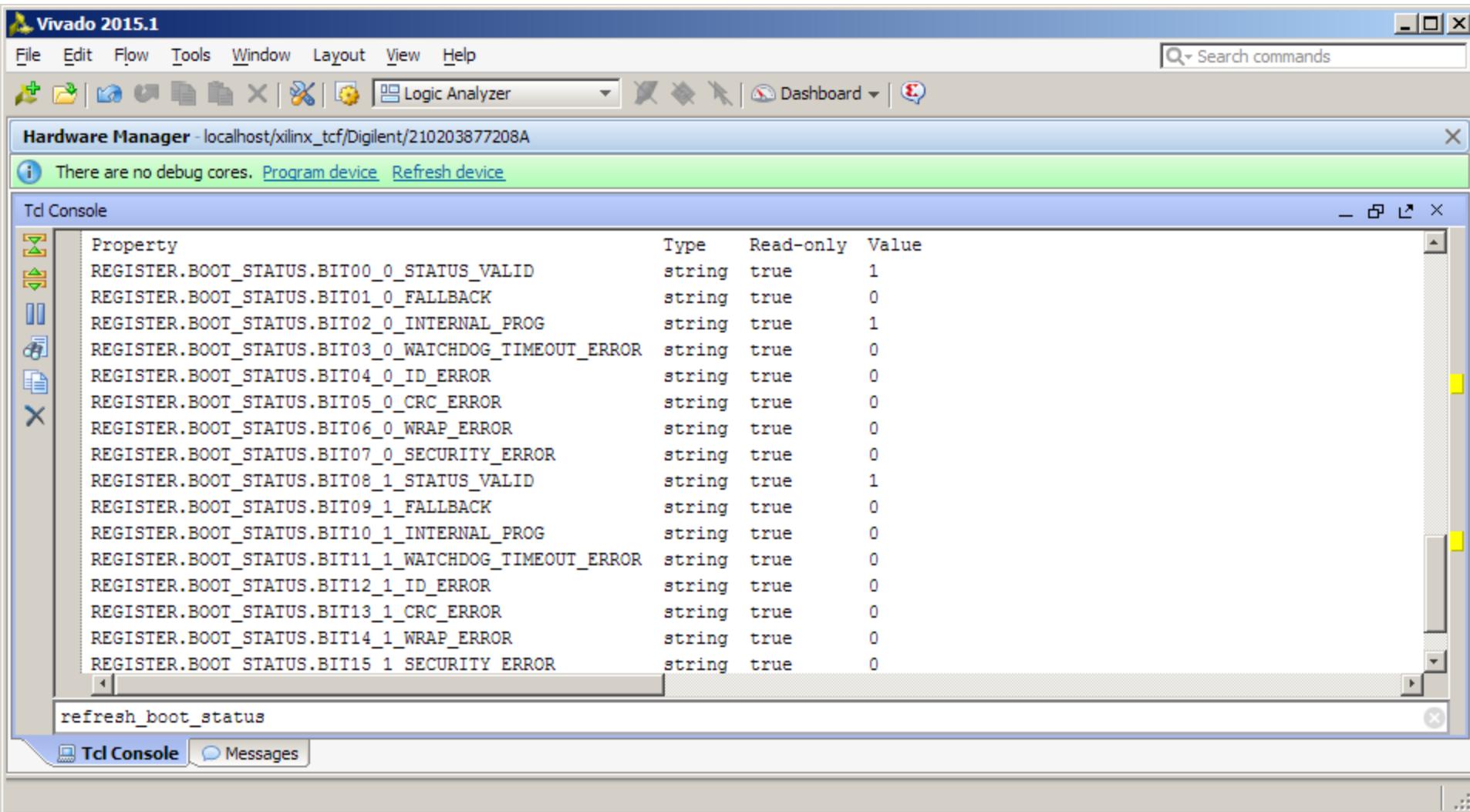
Run MultiBoot Design

- 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
- This happens very quickly; the DONE LED may appear to be blinking



Run MultiBoot Design

► Type `refresh_boot_status` and press Enter



Run MultiBoot Design

► The following status register messages are now seen:

- IPROG_0 is no longer active
- 0_FALLBACK, 1_INTERNAL_IPROG, and 1_CRC_ERROR are now active

The screenshot shows the Vivado 2015.1 interface with the "Hardware Manager" window open. The "Tcl Console" tab is selected, displaying a table of boot status register properties. Several rows in the table are highlighted with red boxes, indicating the active status register bits.

Property	Type	Read-only	Value
REGISTER.BOOT_STATUS.BIT00_0_STATUS_VALID	string	true	1
REGISTER.BOOT_STATUS.BIT01_0_FALLBACK	string	true	1
REGISTER.BOOT_STATUS.BIT02_0_INTERNAL_PROG	string	true	0
REGISTER.BOOT_STATUS.BIT03_0_WATCHDOG_TIMEOUT_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT04_0_ID_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT05_0_CRC_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT06_0_WRAP_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT07_0_SECURITY_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT08_1_STATUS_VALID	string	true	1
REGISTER.BOOT_STATUS.BIT09_1_FALLBACK	string	true	0
REGISTER.BOOT_STATUS.BIT10_1_INTERNAL_PROG	string	true	1
REGISTER.BOOT_STATUS.BIT11_1_WATCHDOG_TIMEOUT_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT12_1_ID_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT13_1_CRC_ERROR	string	true	1
REGISTER.BOOT_STATUS.BIT14_1_WRAP_ERROR	string	true	0
REGISTER.BOOT_STATUS.BIT15_1_SECURITY_ERROR	string	true	0

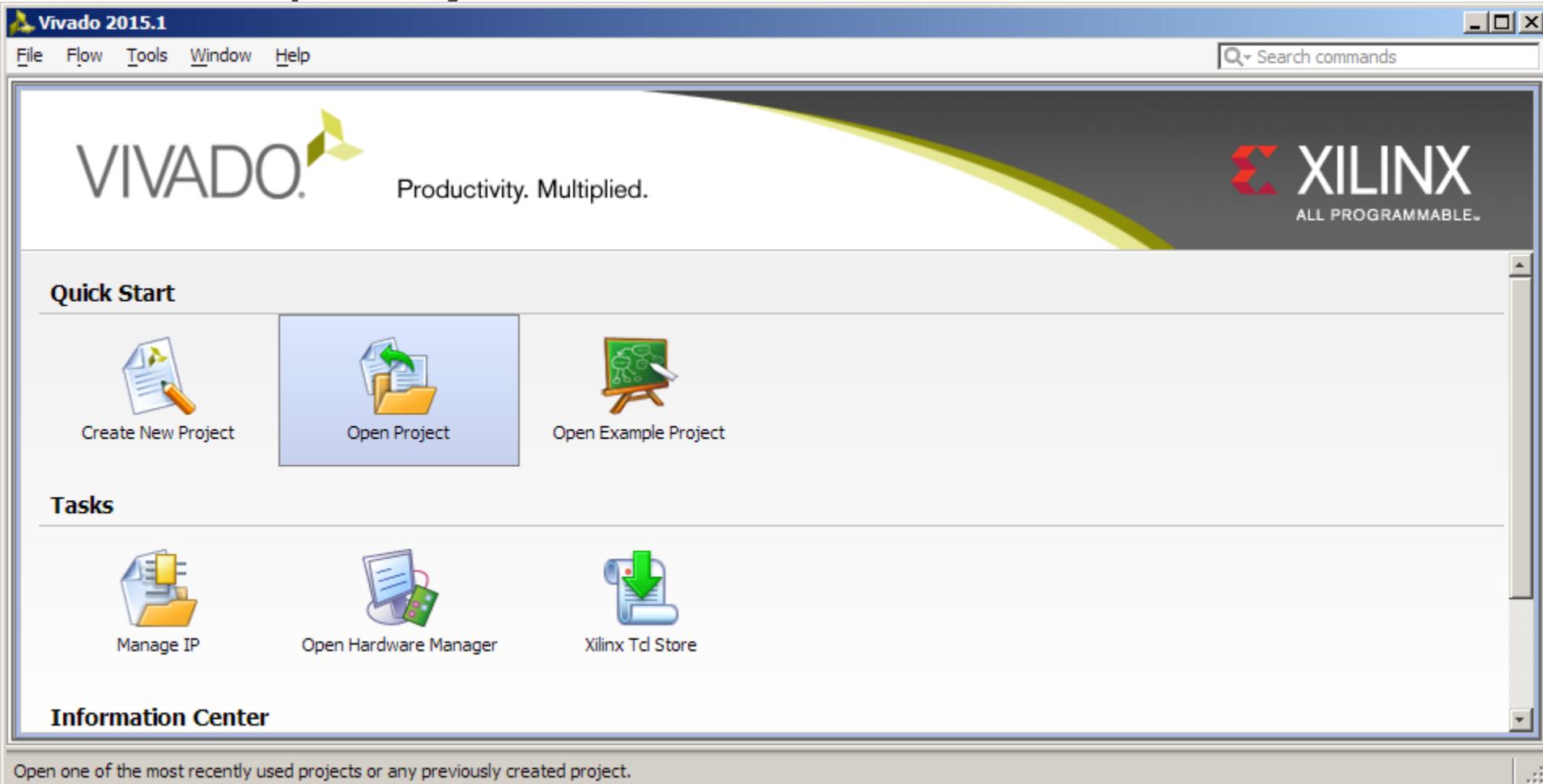
Compile AC701 MultiBoot Design

Compile AC701 MultiBoot Design

► Open Vivado

Start → All Programs → Xilinx Design Tools → Vivado 2015.1 → Vivado

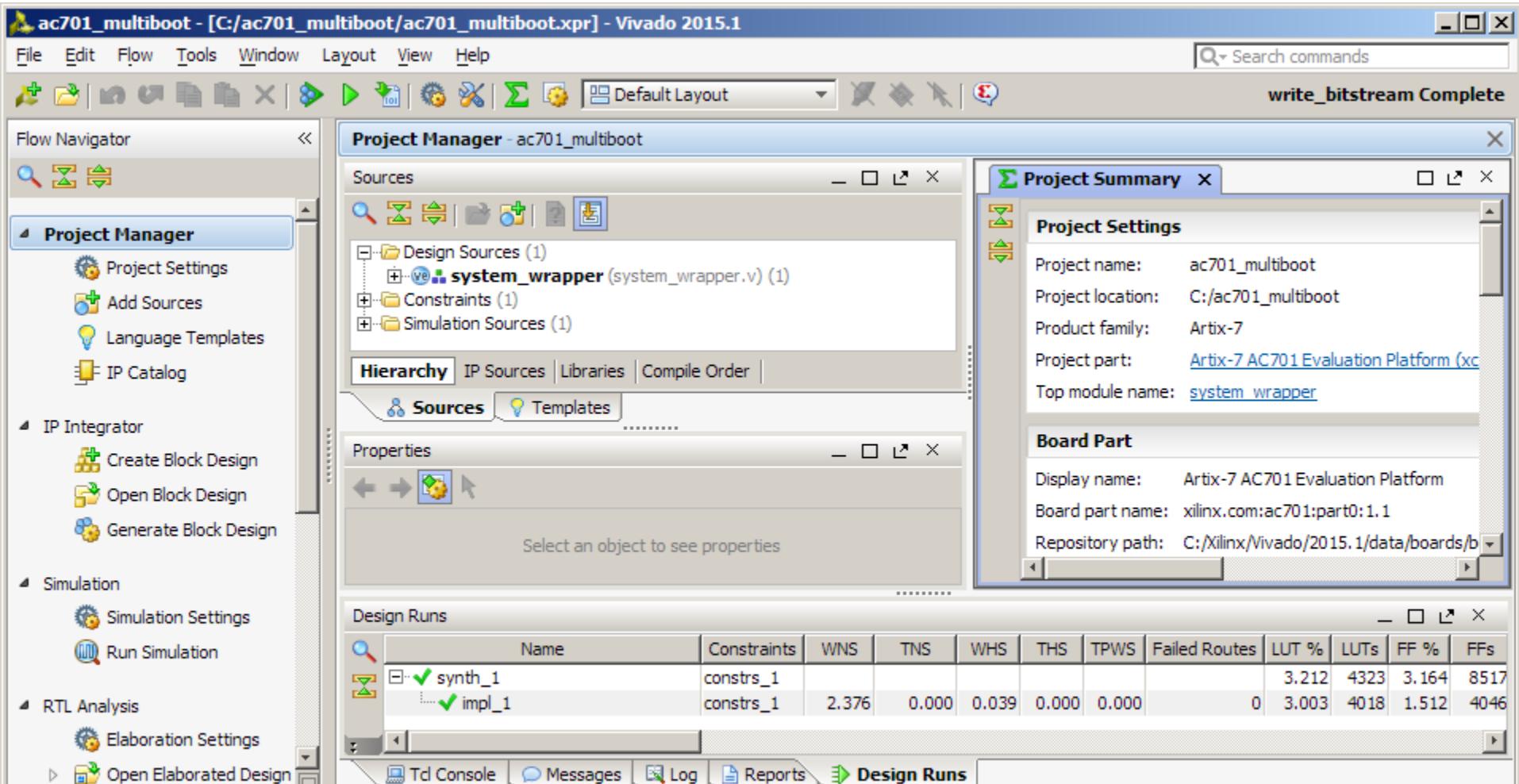
► Select Open Project



Compile AC701 MultiBoot Design

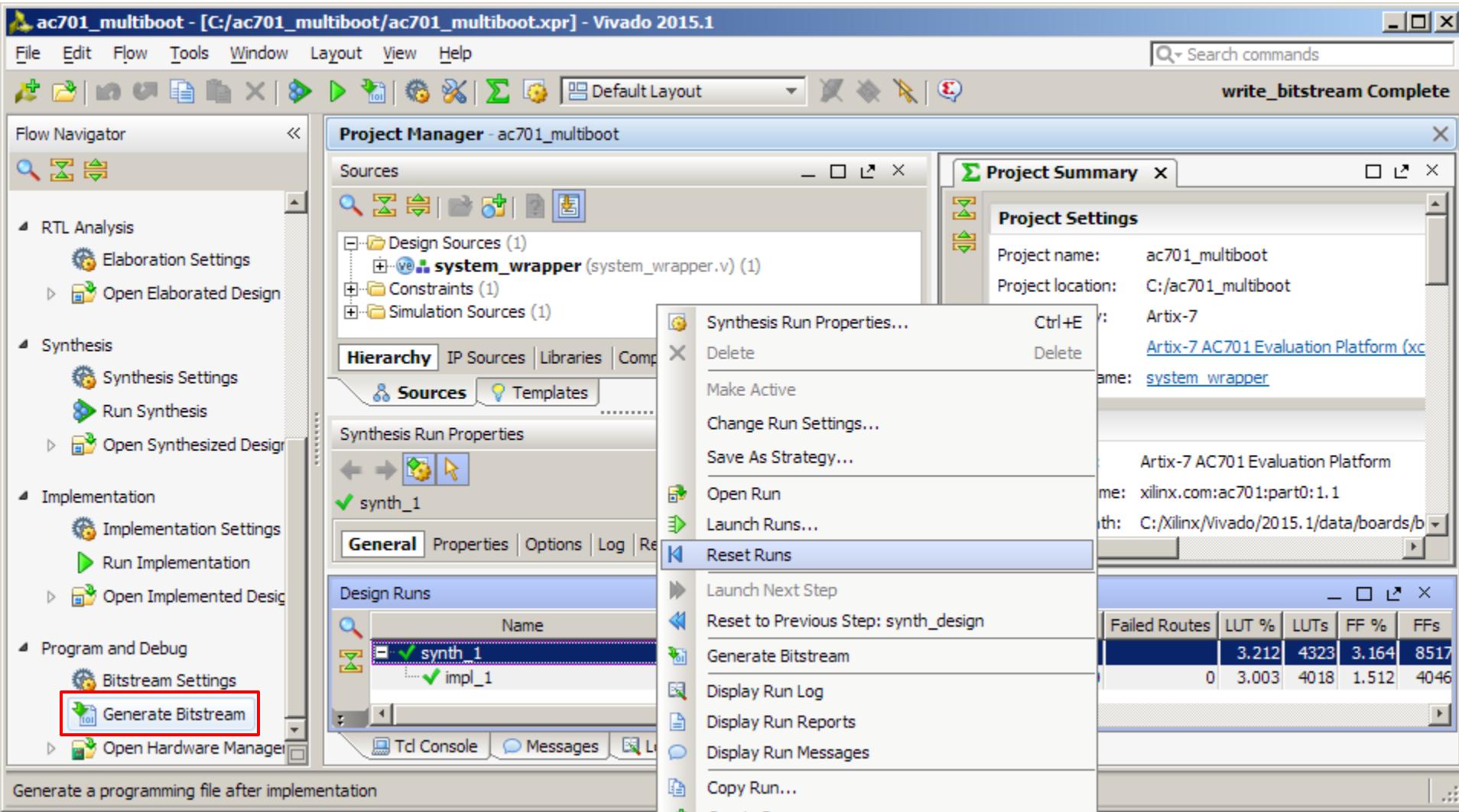
► Open the AC701 Design:

- <Design Name>\ac701_multiboot.xpr



Compile AC701 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**

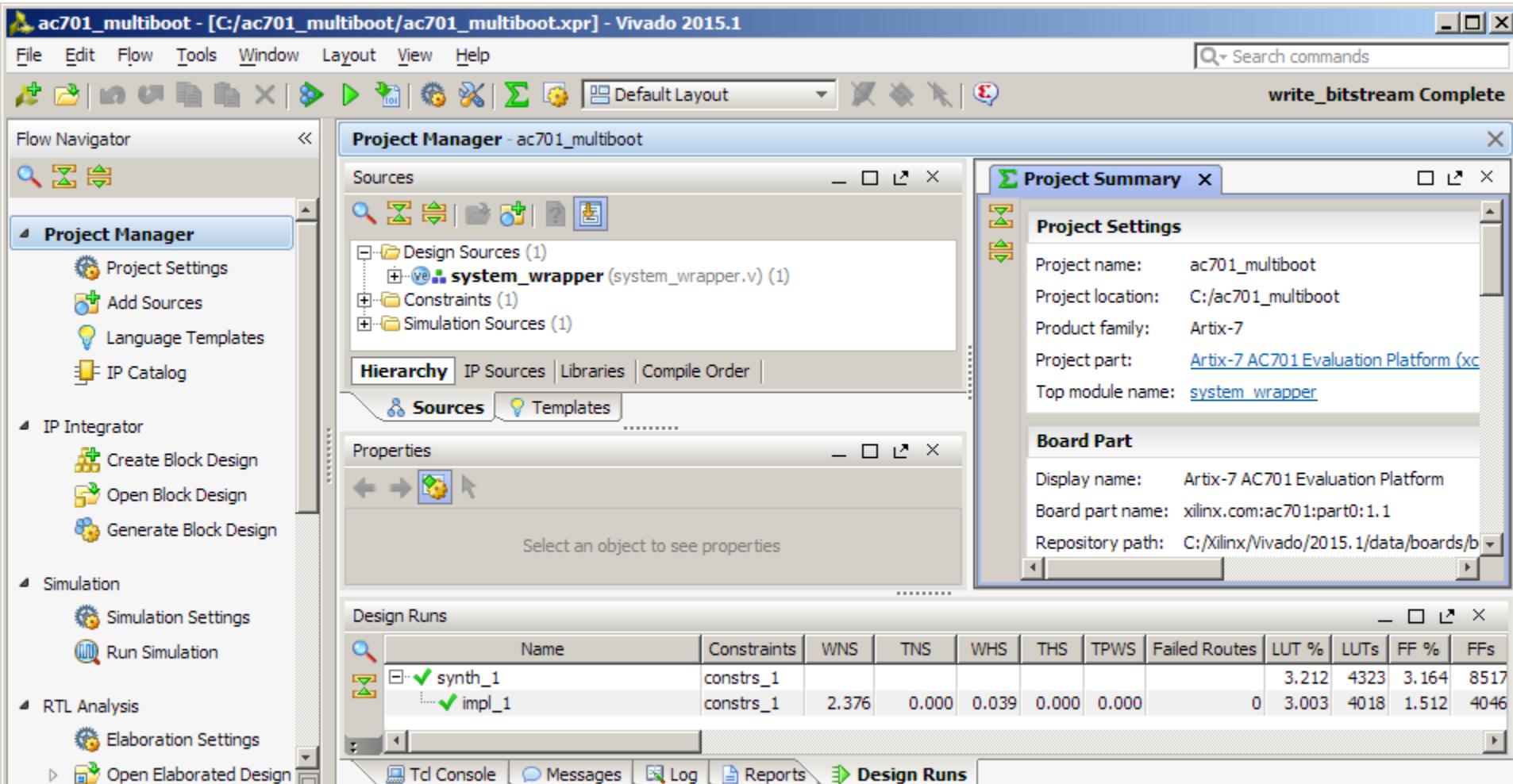


Note: Presentation applies to the AC701

XILINX ➤ ALL PROGRAMMABLE™

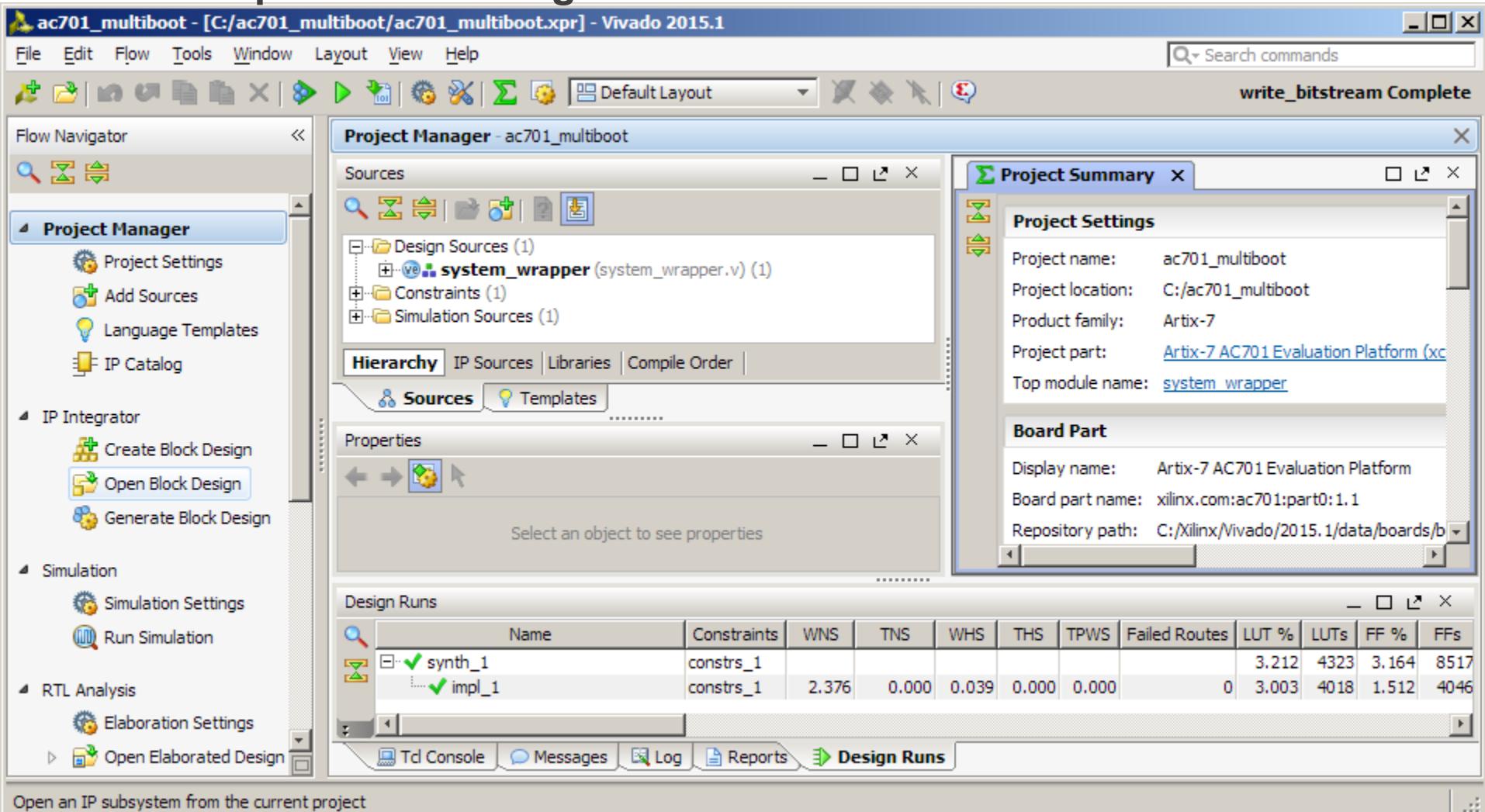
Compile AC701 MultiBoot Design

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



Compile AC701 MultiBoot Design

- The MultiBoot Design has been implemented with IP Integrator (IPI)
- Click Open Block Design



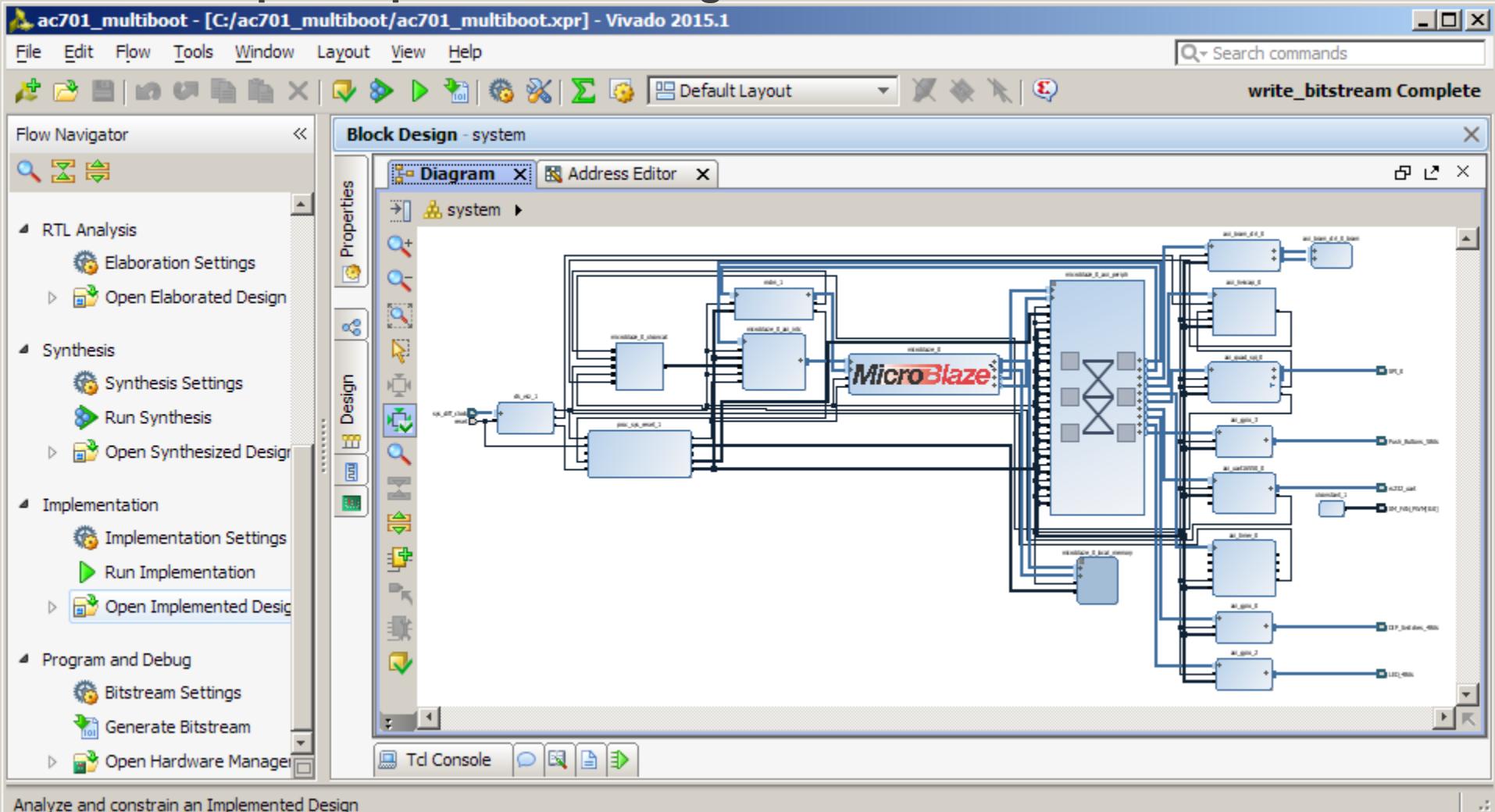
Open an IP subsystem from the current project

Note: Presentation applies to the AC701

XILINX ➤ ALL PROGRAMMABLE

Compile AC701 MultiBoot Design

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

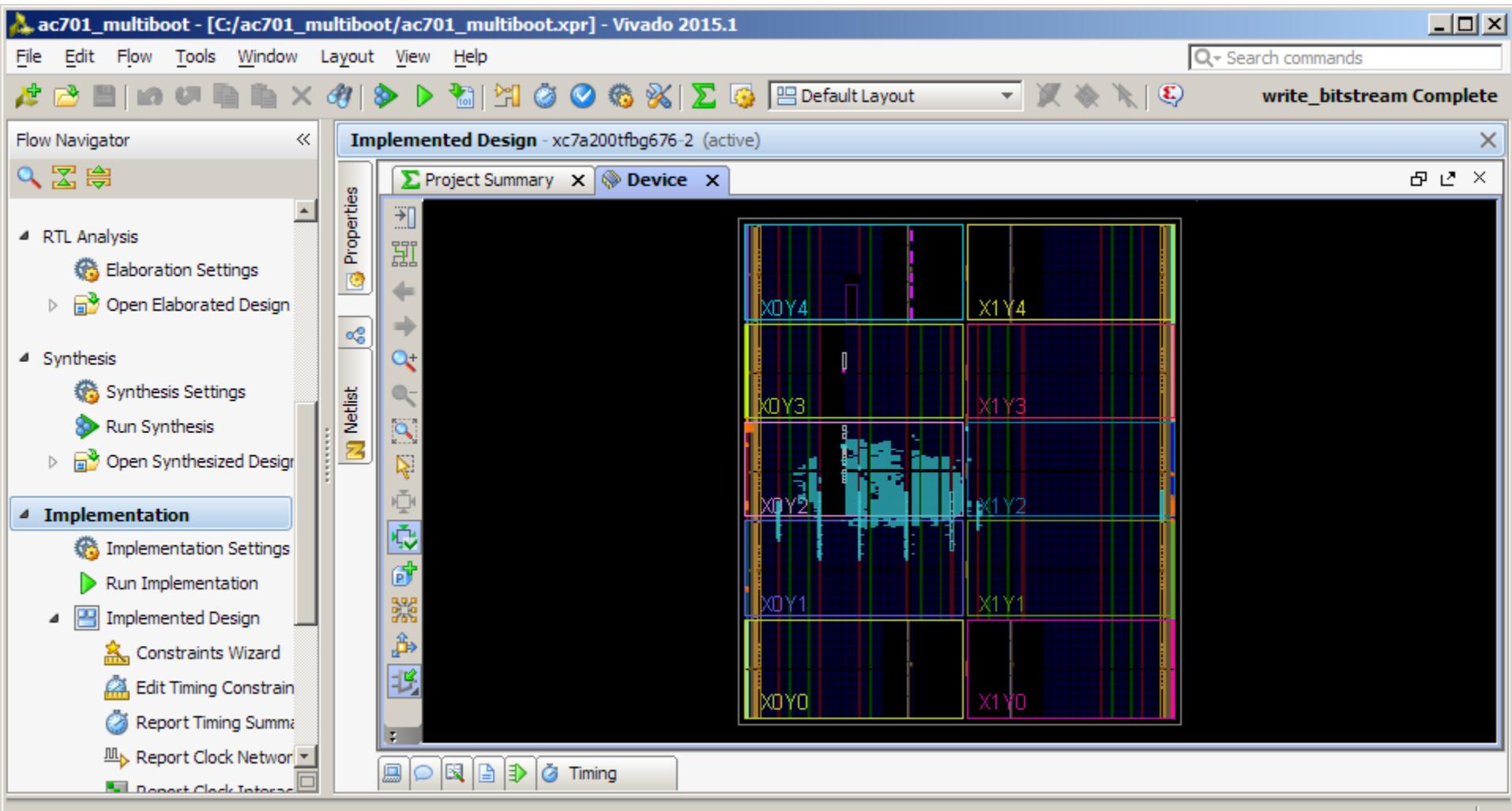


Note: Presentation applies to the AC701

XILINX ➤ ALL PROGRAMMABLE™

Compile AC701 MultiBoot Design

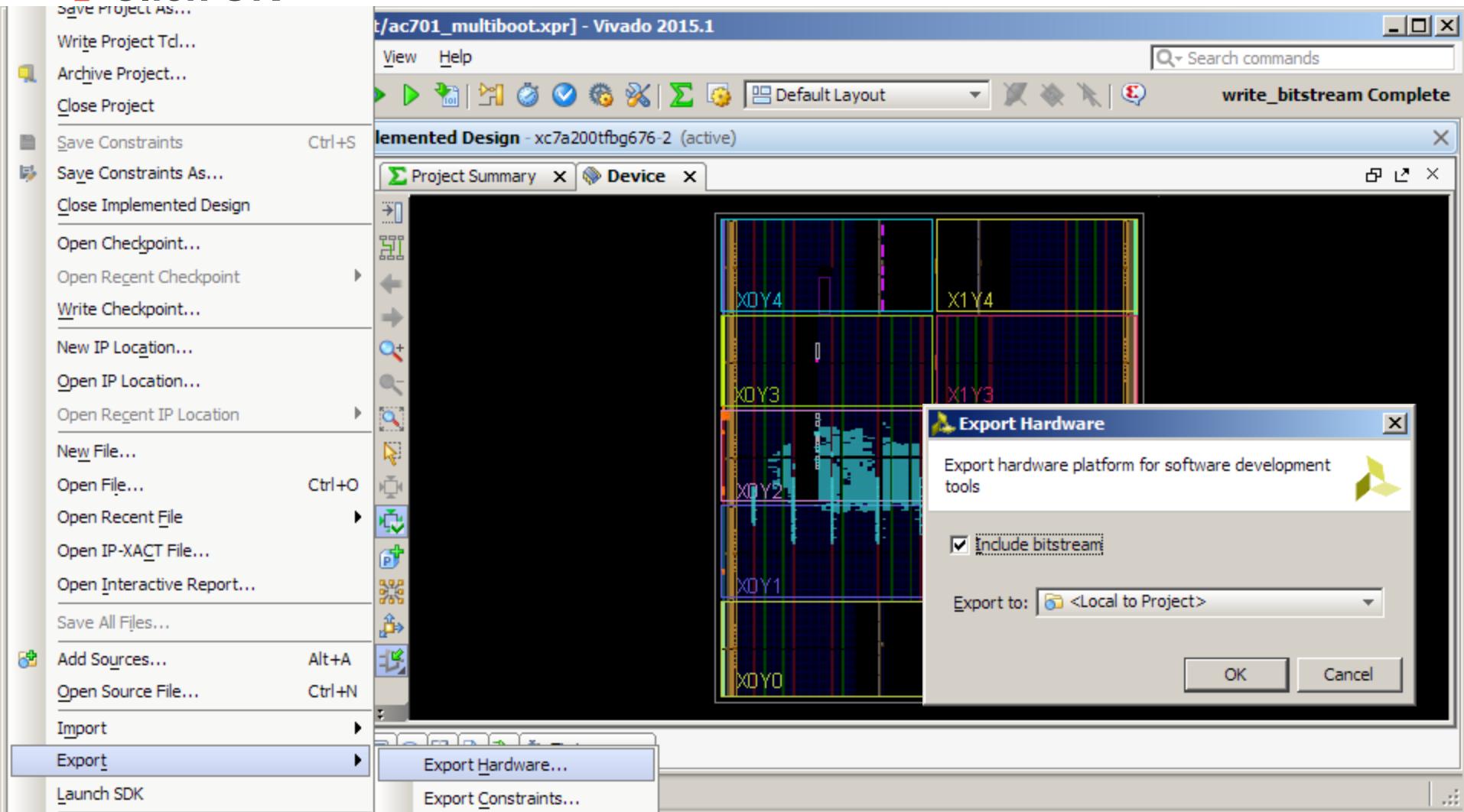
► View Implemented Design



Compile AC701 MultiBoot Design

► Select File → Export → Export Hardware

► Click OK



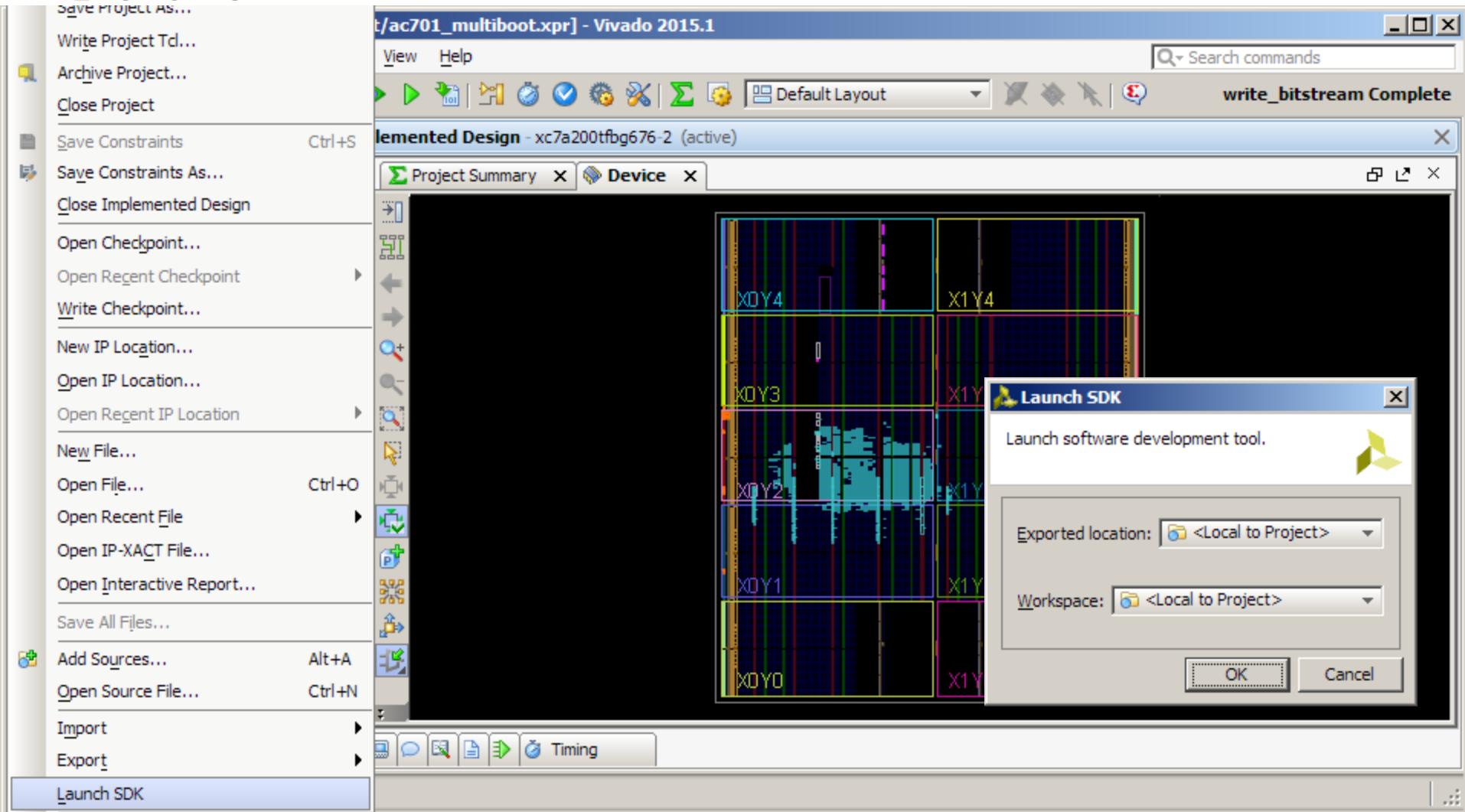
Note: Presentation applies to the AC701

XILINX ➤ ALL PROGRAMMABLE™

Compile AC701 MultiBoot Design

► Select File → Launch SDK

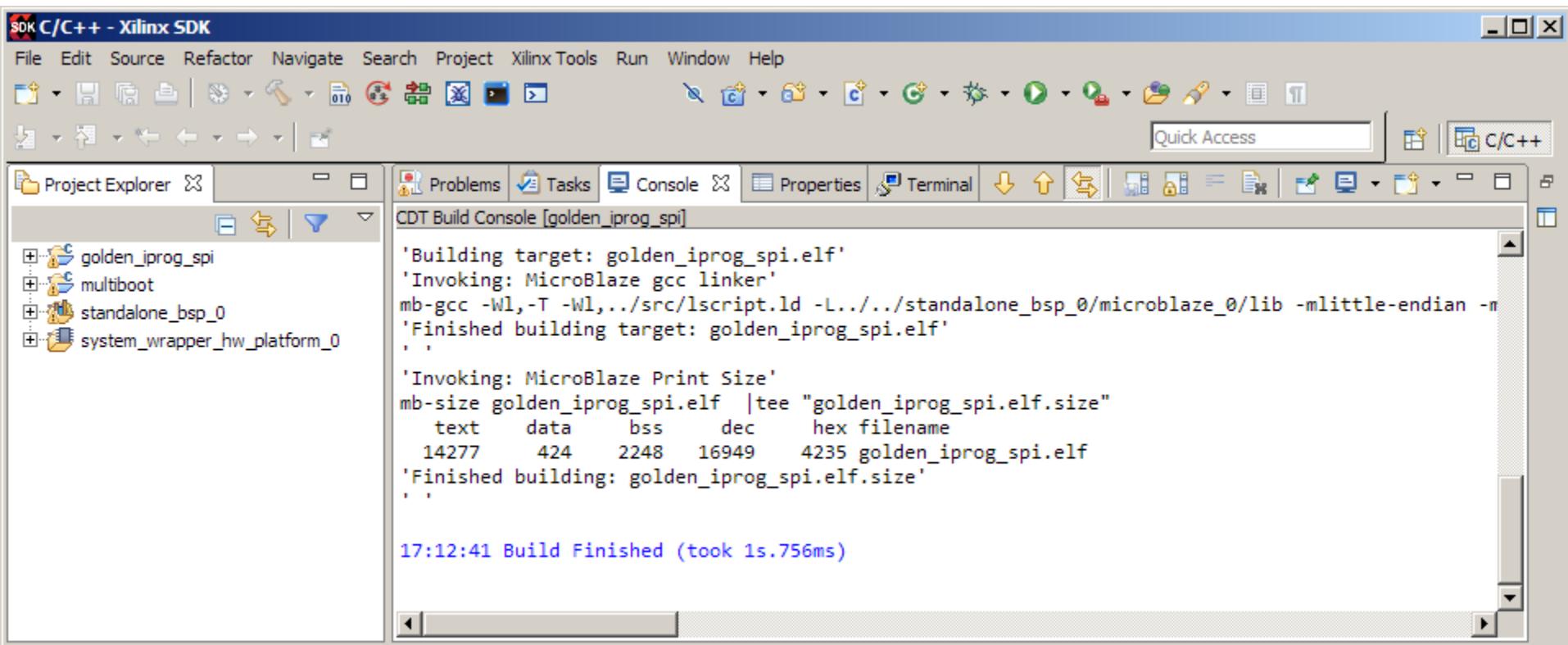
► Click OK



Compile AC701 Software in SDK

➤ SDK Software Compile - Build ELF files in SDK

- When done, close SDK and return to Vivado



The screenshot shows the Xilinx C/C++ - Xilinx SDK interface. The window title is "SDK C/C++ - Xilinx SDK". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Xilinx Tools, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Build. The left pane is the "Project Explorer" showing four projects: "golden_iprog_spi", "multiboot", "standalone_bsp_0", and "system_wrapper_hw_platform_0". The right pane is the "CDT Build Console" tab, which displays the following build log:

```
'Building target: golden_iprog_spi.elf'
'Invoking: MicroBlaze gcc linker'
mb-gcc -Wl,-T -Wl,../src/lscript.ld -L../../standalone_bsp_0/microblaze_0/lib -mlittle-endian -march=mb
'Finished building target: golden_iprog_spi.elf'
'

'Invoking: MicroBlaze Print Size'
mb-size golden_iprog_spi.elf | tee "golden_iprog_spi.elf.size"
    text      data      bss      dec      hex filename
  14277      424     2248   16949     4235 golden_iprog_spi.elf
'Finished building: golden_iprog_spi.elf.size'
'

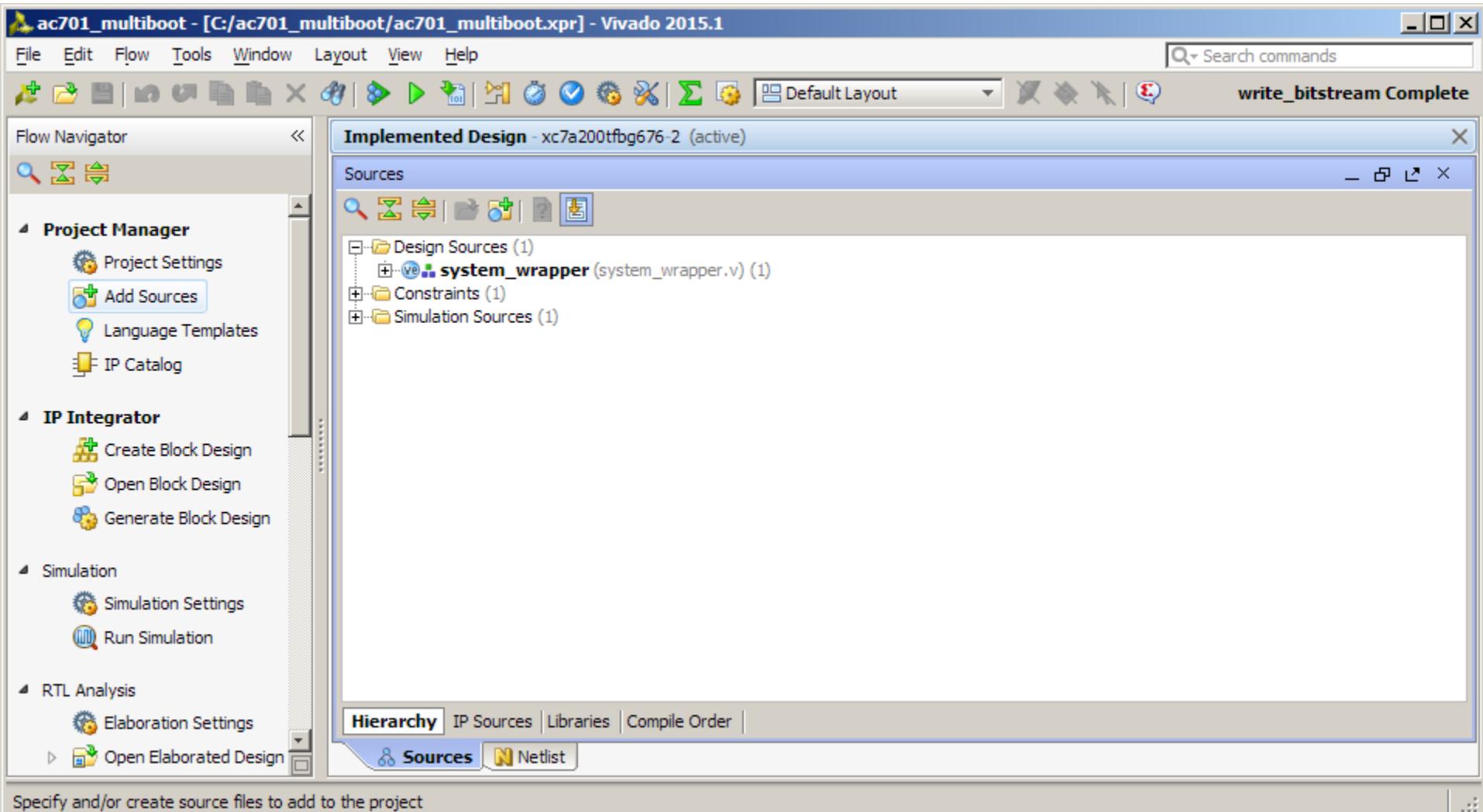
17:12:41 Build Finished (took 1s.756ms)
```

Compile AC701 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

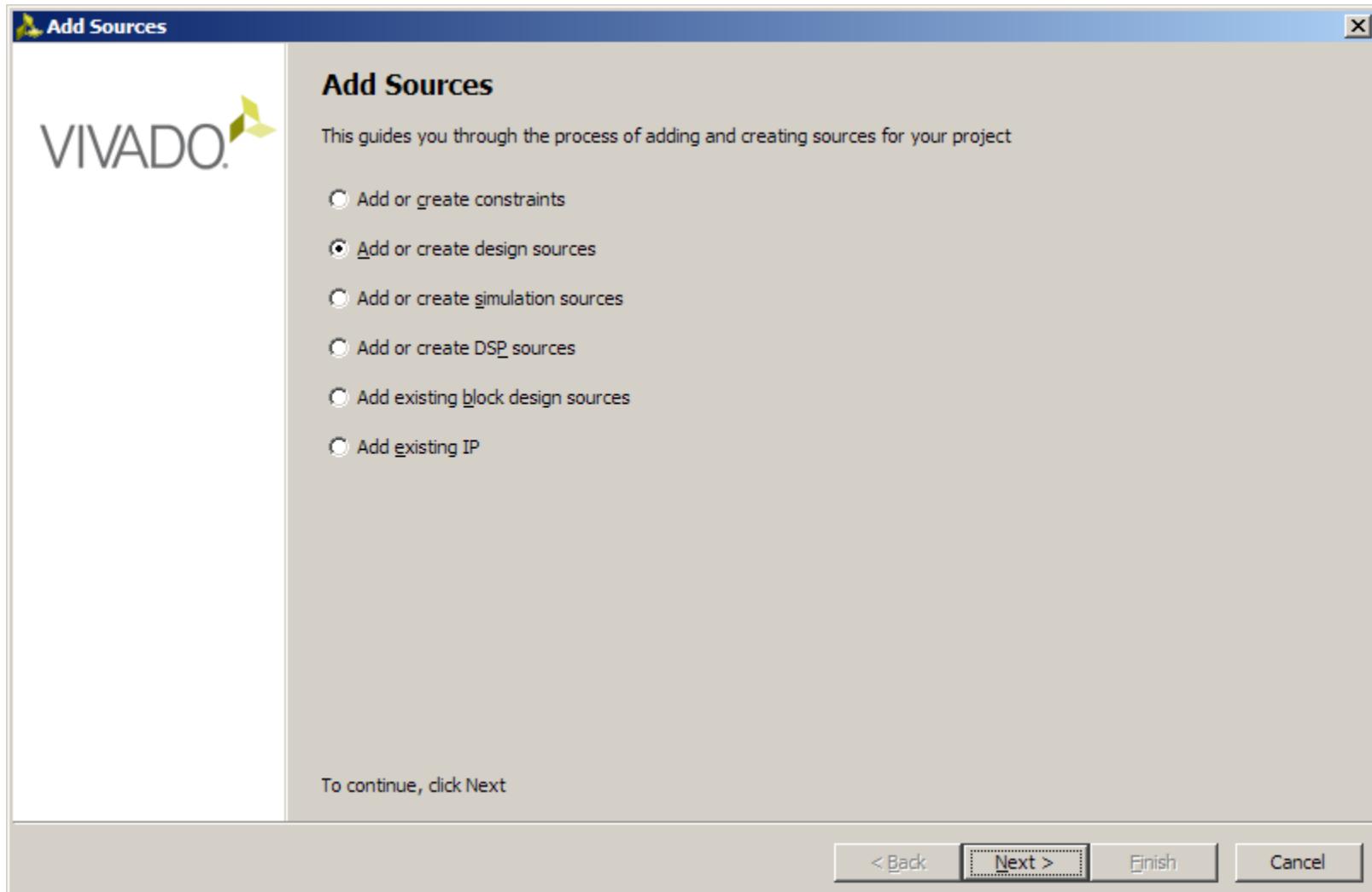
Compile AC701 MultiBoot Design

► Select Add Sources



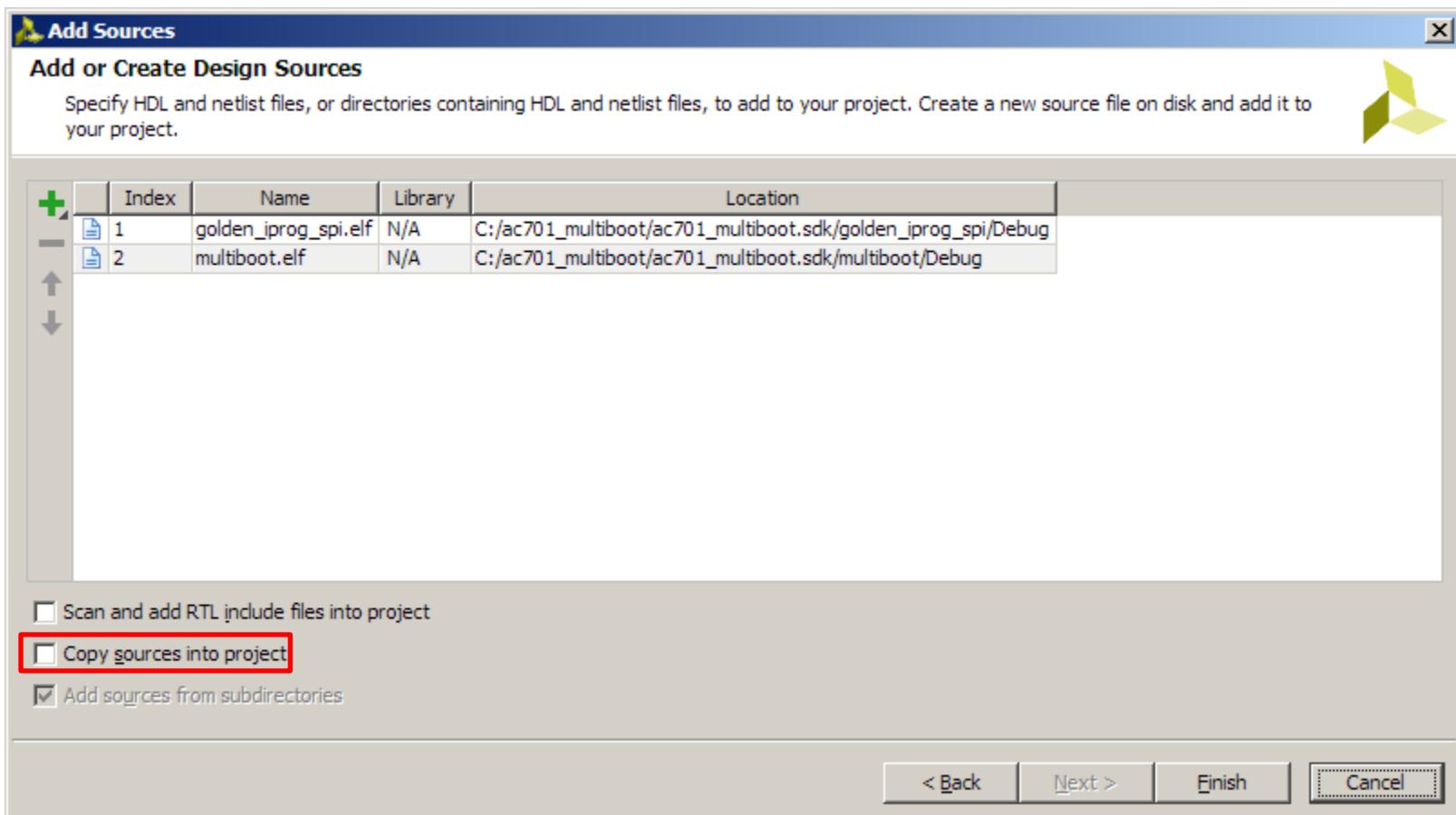
Compile AC701 MultiBoot Design

► Select Add or Create Design Sources



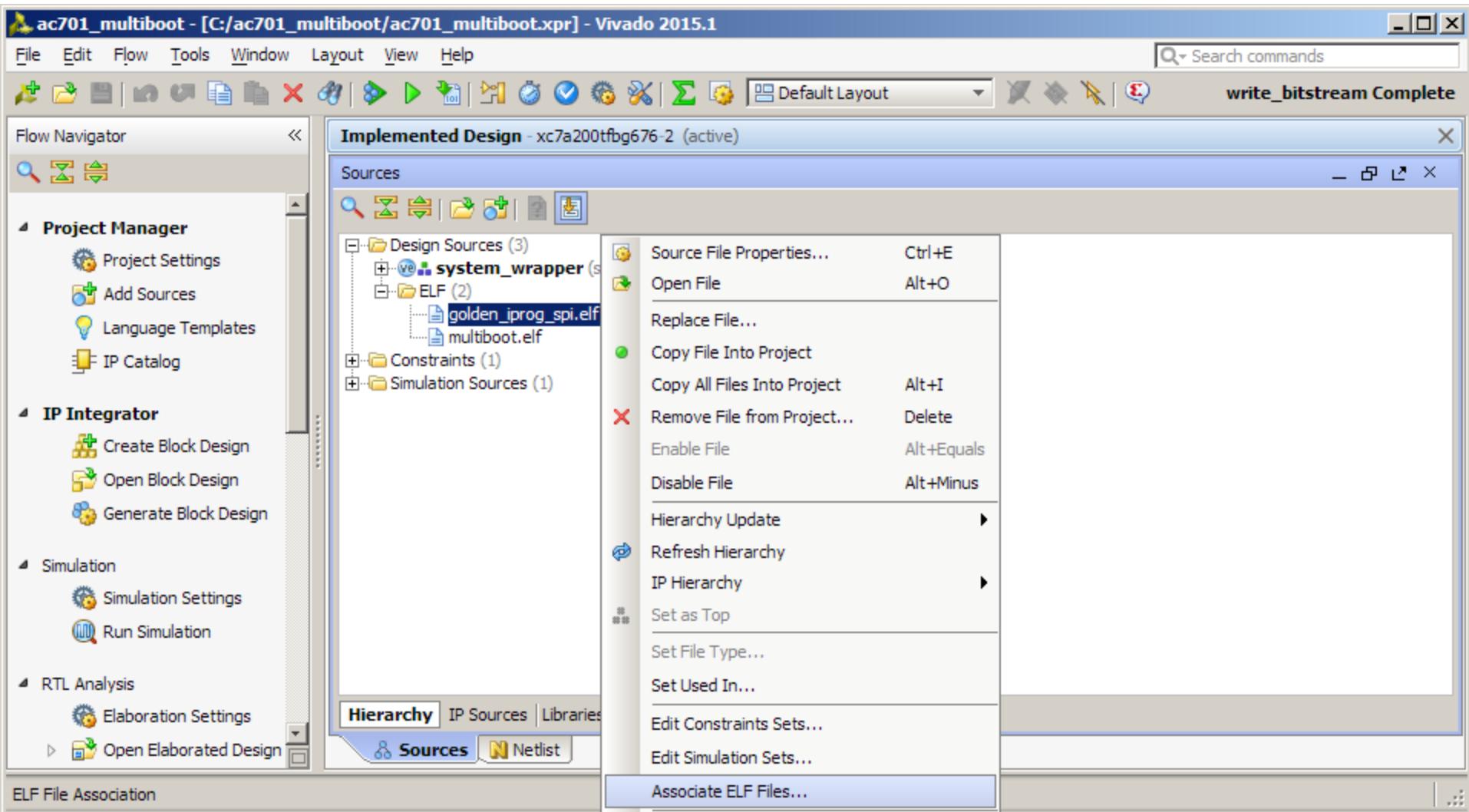
Compile AC701 MultiBoot Design

- 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



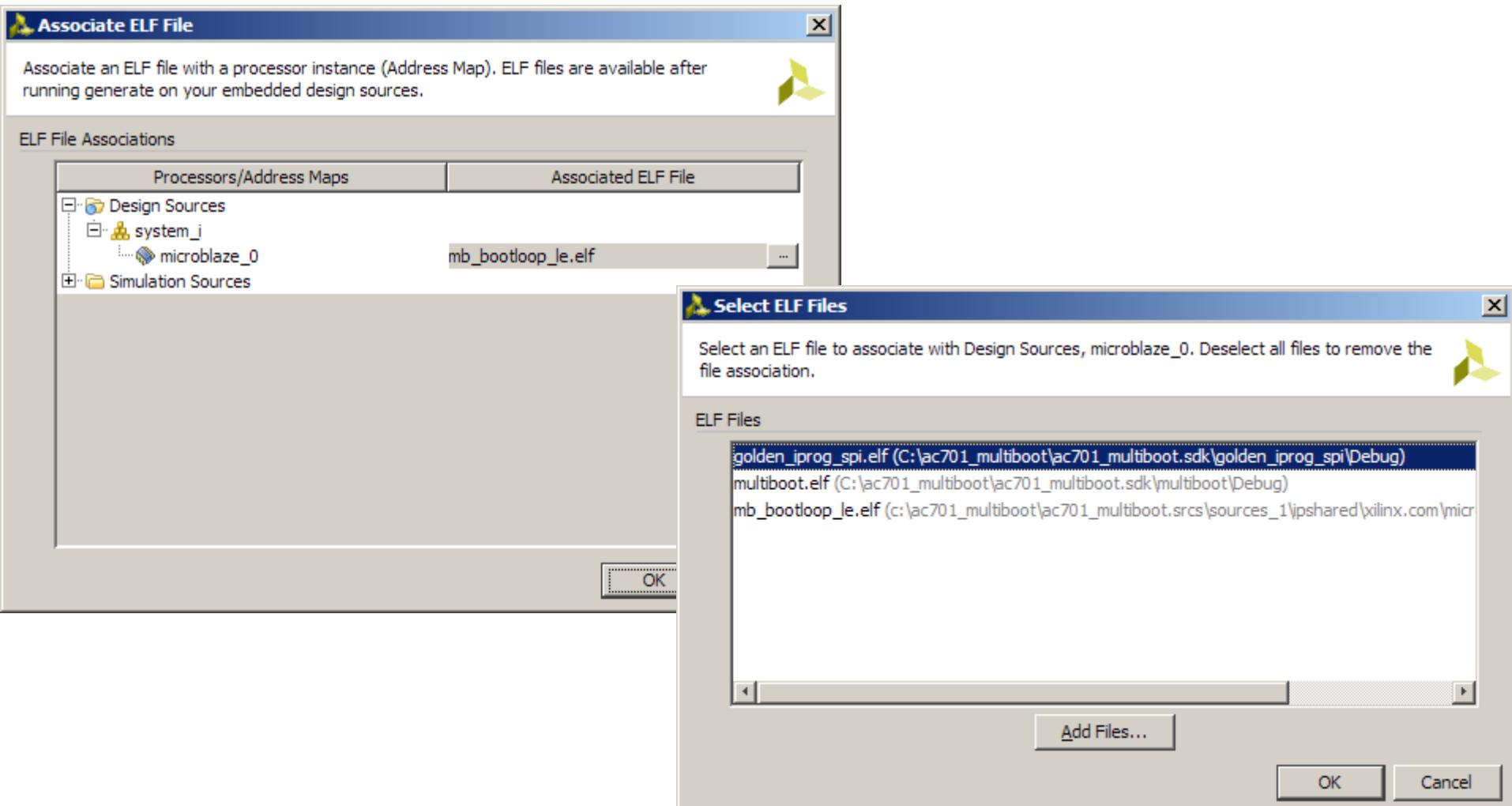
Compile AC701 MultiBoot Design

► Right-click on one of the ELF files and select Associate ELF files



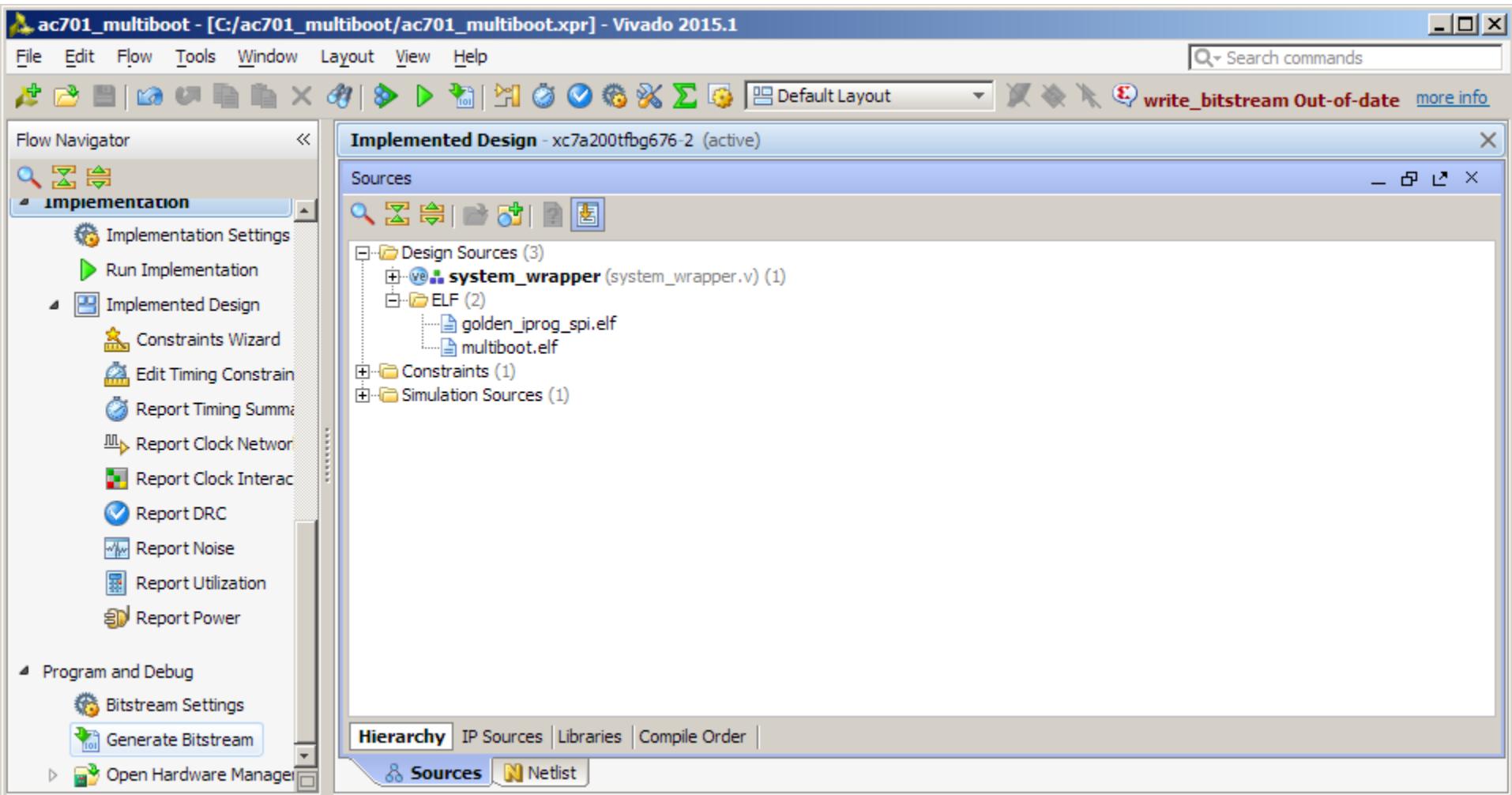
Compile AC701 MultiBoot Design

- Click the button to the right; select the `golden_iprog_spi.elf` then click OK twice



Compile AC701 MultiBoot Design

► Select Generate Bitstream

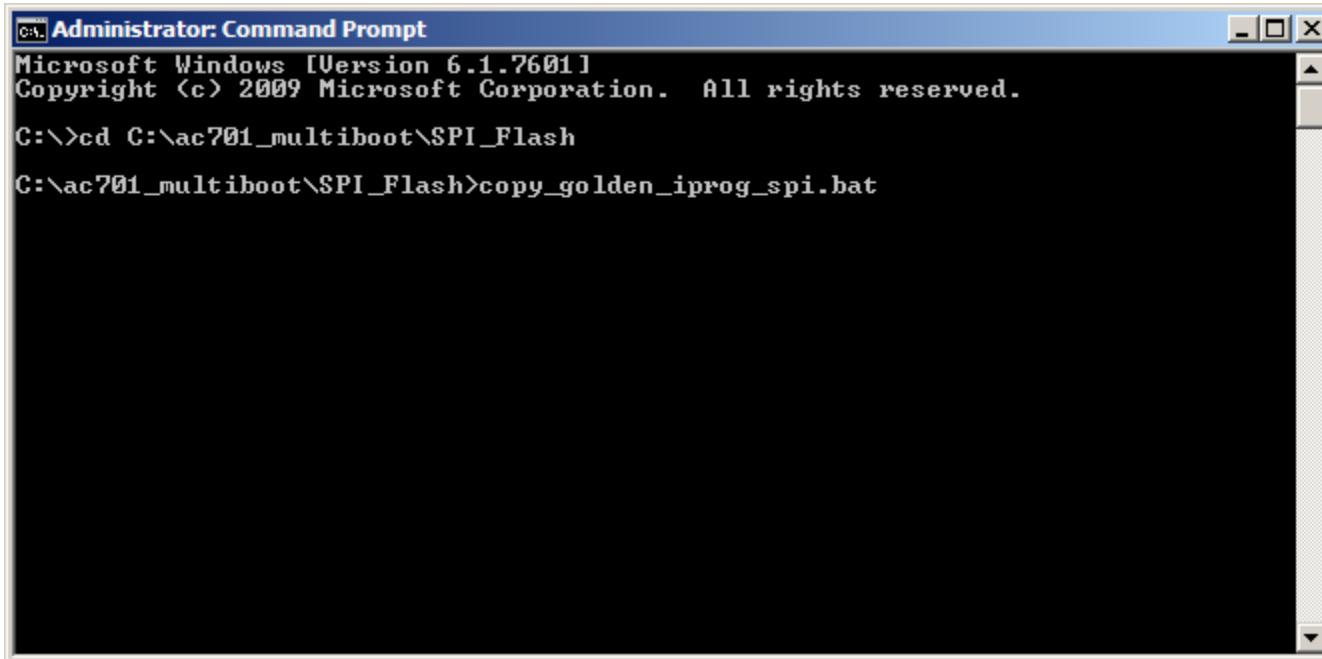


Compile AC701 MultiBoot Design

- Once the `write_bitstream` command finishes, open a Windows Prompt and enter these commands:

```
cd C:\ac701_multiboot\SPI_Flash
```

```
copy_golden_iprog_spi.bat
```



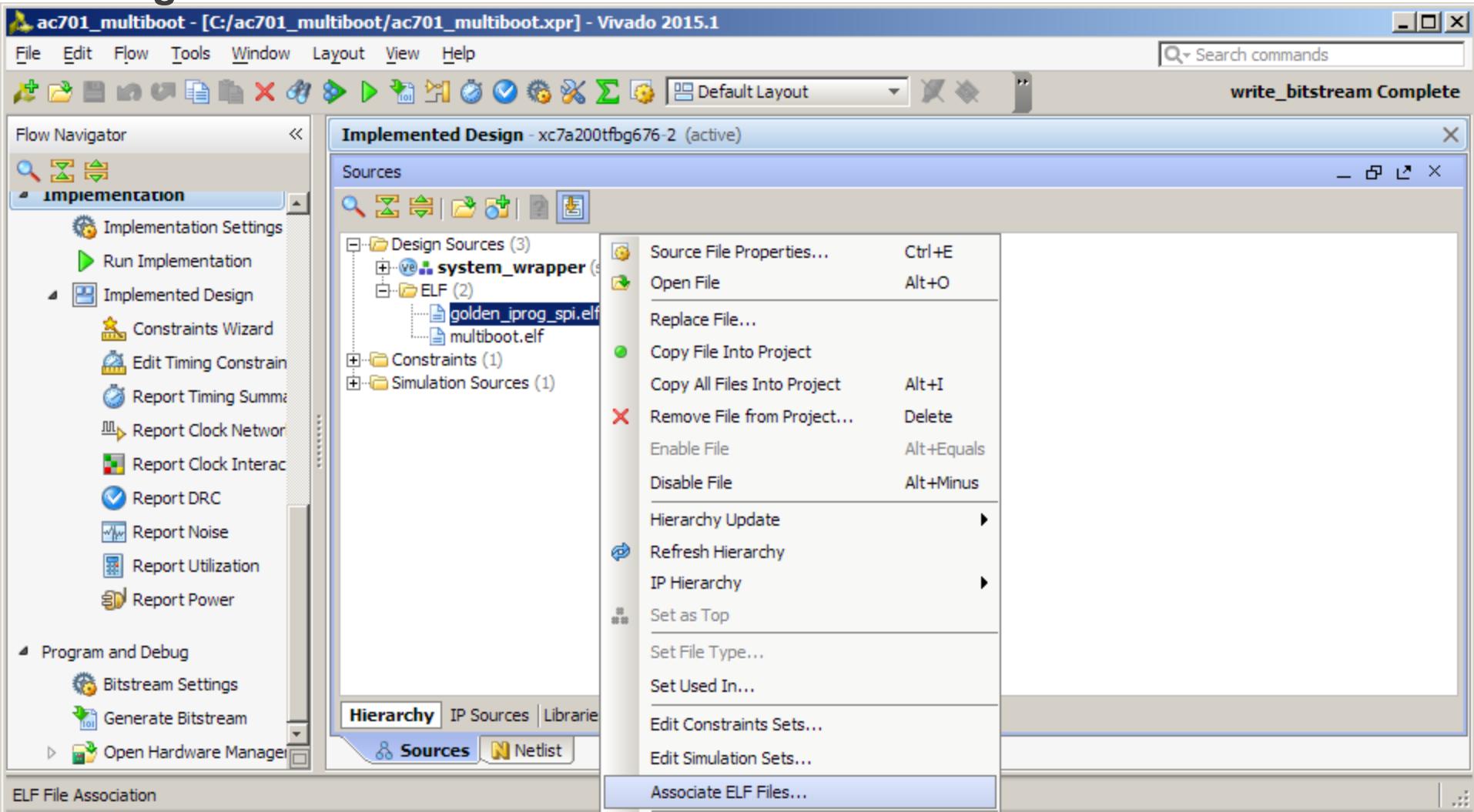
The screenshot shows a Windows Command Prompt window with the title bar "Administrator: Command Prompt". The window displays the following text:

```
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:>cd C:\ac701_multiboot\SPI_Flash
C:\ac701_multiboot\SPI_Flash>copy_golden_iprog_spi.bat
```

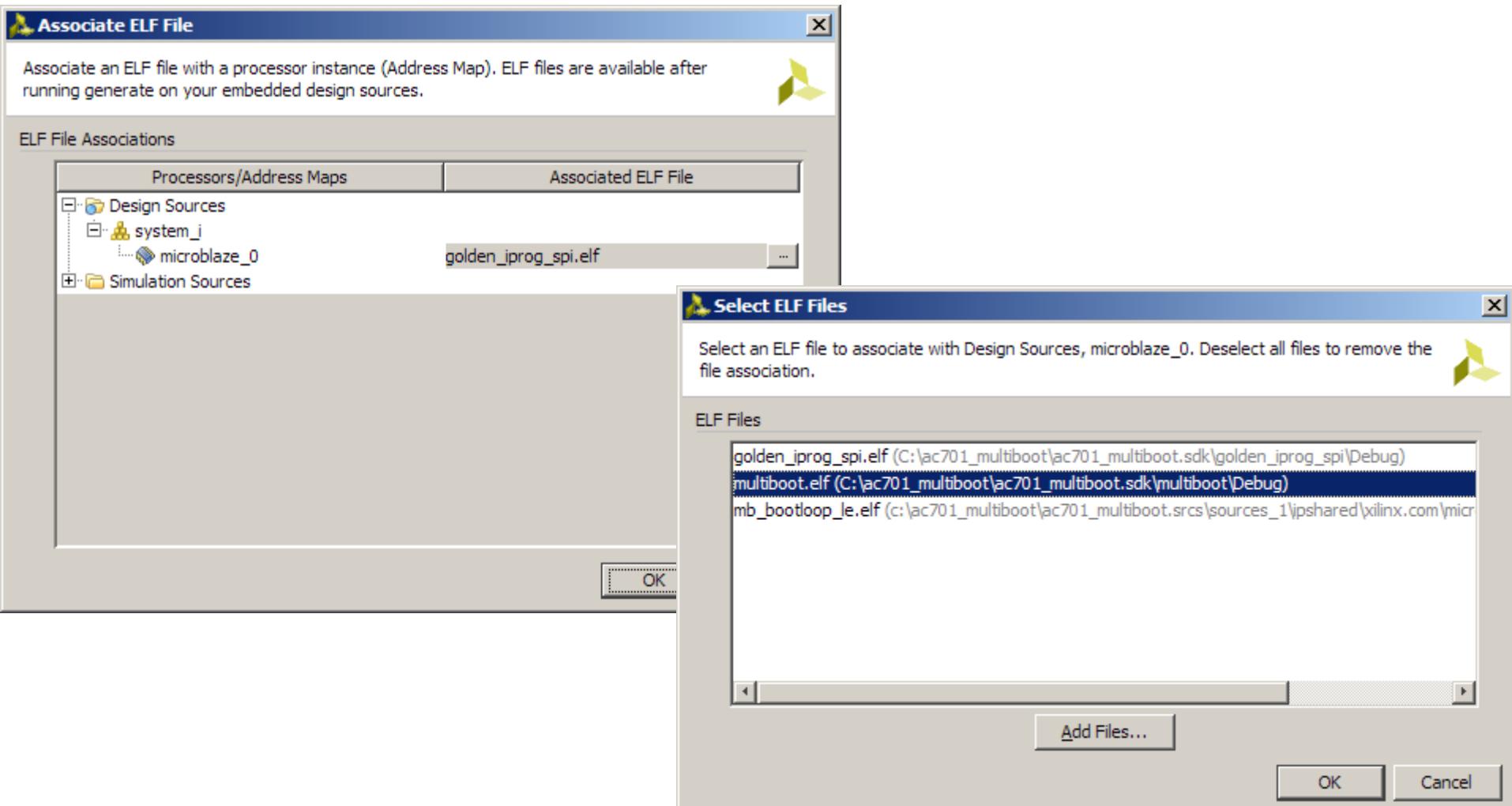
Compile AC701 MultiBoot Design

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



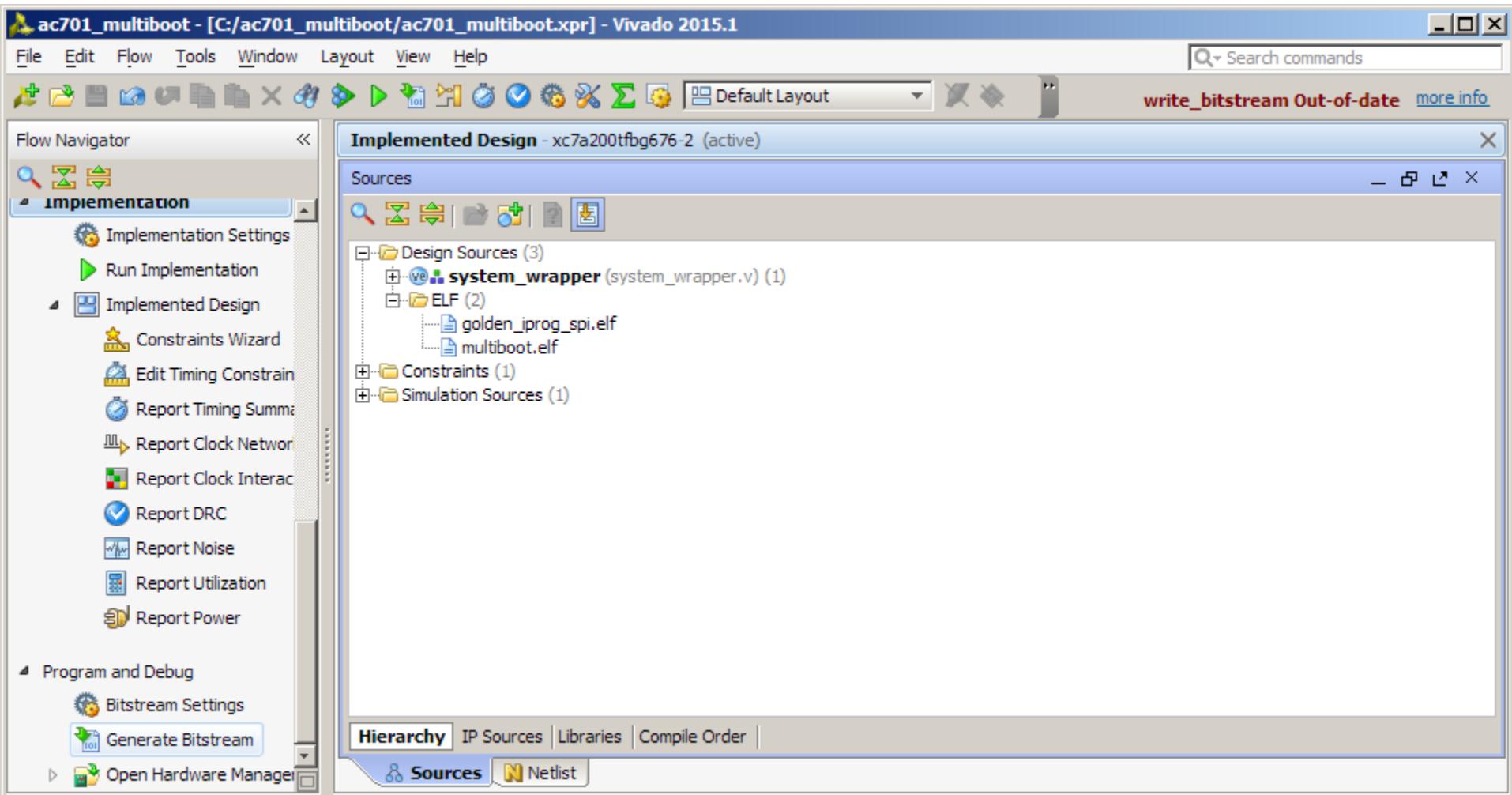
Compile AC701 MultiBoot Design

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



Compile AC701 MultiBoot Design

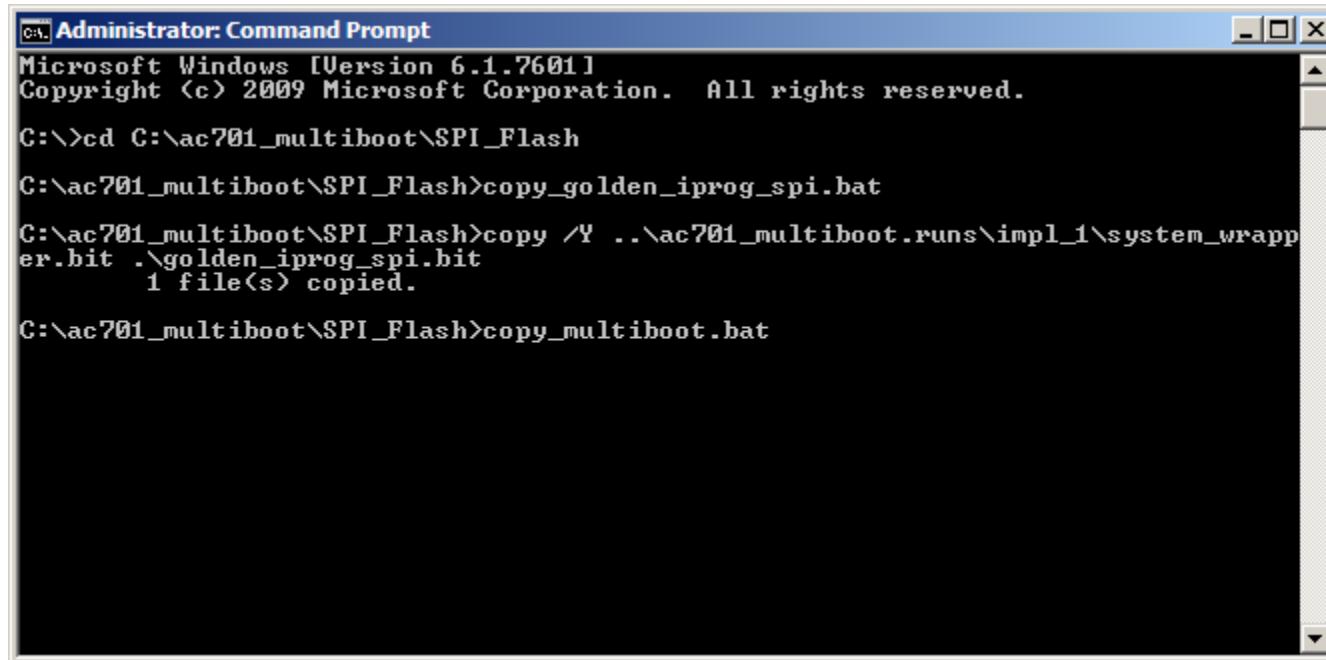
► Select Generate Bitstream



Compile AC701 MultiBoot Design

► From the Command Prompt type:

copy_multiboot.bat



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The window title bar is blue with white text. The main area of the window is black with white text, displaying the command history and output of the batch file. The text in the window is as follows:

```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

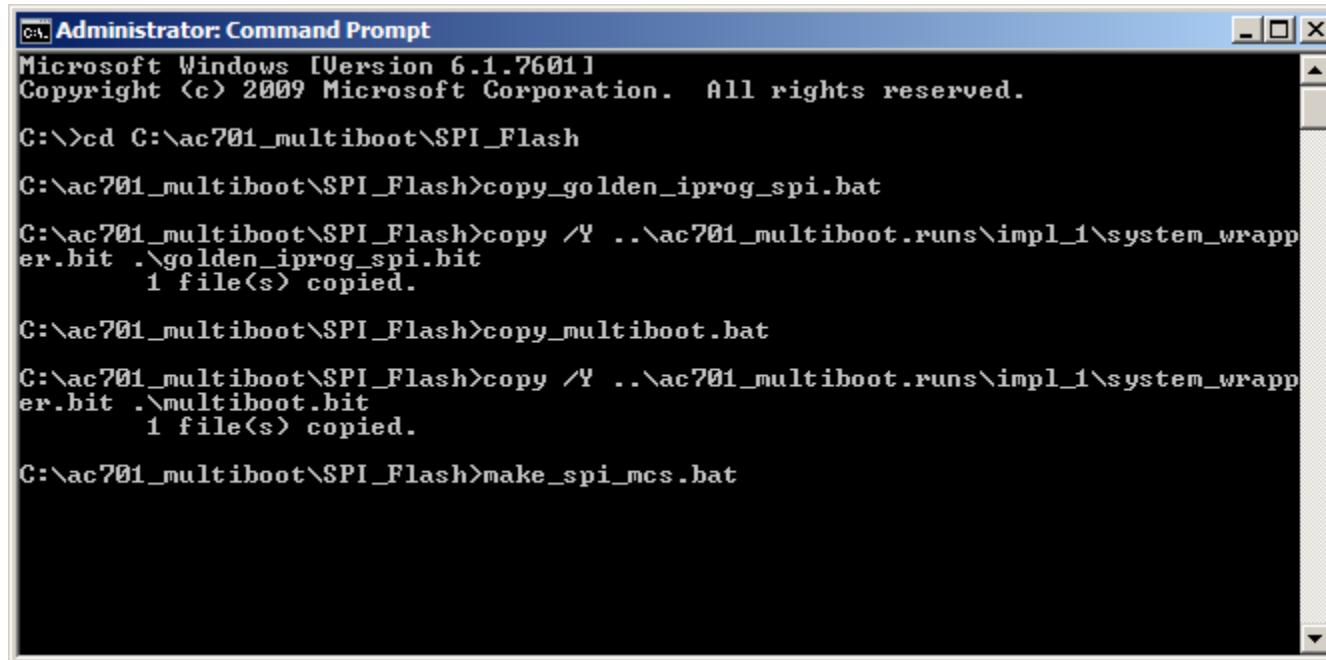
C:\>cd C:\ac701_multiboot\SPI_Flash
C:\ac701_multiboot\SPI_Flash>copy_golden_iprog_spi.bat
C:\ac701_multiboot\SPI_Flash>copy /Y ..\ac701_multiboot.runs\impl_1\system_wrapper.bit .\golden_iprog_spi.bit
               1 file(s) copied.

C:\ac701_multiboot\SPI_Flash>copy_multiboot.bat
```

Generate AC701 PROM File

► From the Command Prompt type:

make_spi_mcs.bat



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\>cd C:\ac701_multiboot\SPI_Flash
C:\ac701_multiboot\SPI_Flash>copy_golden_iprog_spi.bat
C:\ac701_multiboot\SPI_Flash>copy /Y ..\ac701_multiboot.runs\impl_1\system_wrapper.bit ..\golden_iprog_spi.bit
               1 file(s) copied.

C:\ac701_multiboot\SPI_Flash>copy_multiboot.bat
C:\ac701_multiboot\SPI_Flash>copy /Y ..\ac701_multiboot.runs\impl_1\system_wrapper.bit ..\multiboot.bit
               1 file(s) copied.

C:\ac701_multiboot\SPI_Flash>make_spi_mcs.bat
```

Artix-7 MultiBoot Details

MultiBoot Register Setup & Command

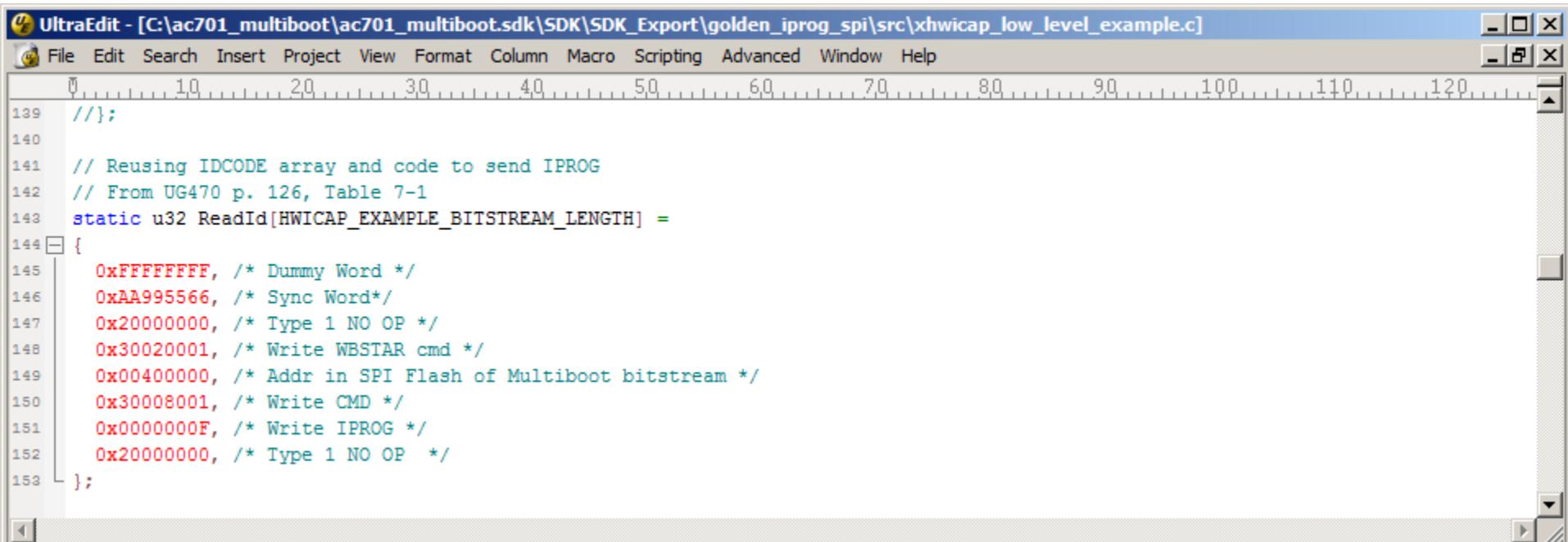
- Below are the values programmed into the ICAP via IPROG
 - This table is from UG470, 7 Series FPGAs Configuration User Guide

Table 7-1: Example Bitstream for IPROG through ICAP

Configuration Data (hex)	Explanation
FFFFFFFF	Dummy Word
AA995566	Sync Word
20000000	Type 1 NO OP
30020001	Type 1 Write 1 Words to WBSTAR
00000000	Warm Boot Start Address (Load the Desired Address)
30008001	Type 1 Write 1 Words to CMD
0000000F	IPROG Command
20000000	Type 1 NO OP

MultiBoot Register Setup & Command

- IPROG command issued via the software in `xhwicap_low_level_example.c`



The screenshot shows a window titled "UltraEdit - [C:\ac701_multiboot\ac701_multiboot.sdk\SDK\SDK_Export\golden_iprog_spi\src\xhwicap_low_level_example.c]". The code editor displays a C program snippet. The code is as follows:

```
139 //};  
140  
141 // Reusing IDCODE array and code to send IPROG  
142 // From UG470 p. 126, Table 7-1  
143 static u32 ReadId[HWICAP_EXAMPLE_BITSTREAM_LENGTH] =  
144 {  
145     0xFFFFFFFF, /* Dummy Word */  
146     0xAA995566, /* Sync Word*/  
147     0x20000000, /* Type 1 NO OP */  
148     0x30020001, /* Write WBSTAR cmd */  
149     0x00400000, /* Addr in SPI Flash of Multiboot bitstream */  
150     0x30008001, /* Write CMD */  
151     0x0000000F, /* Write IPROG */  
152     0x20000000, /* Type 1 NO OP */  
153 };
```

References

References

► IP Integrator Documentation

- Vivado Design Suite Tcl Command Reference Guide – UG835
 - http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_1/ug835-vivado-tcl-commands.pdf
- Designing IP Subsystems Using IP Integrator – UG994
 - http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_1/ug994-vivado-ip-subsystems.pdf

► 7 Series Configuration

- 7 Series FPGAs Configuration User Guide
 - http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

Documentation

Documentation

➤ Artix-7

- Artix-7 FPGA Family
 - <http://www.xilinx.com/products/silicon-devices/fpga/artix-7/index.htm>
- Design Advisory Master Answer Record for Artix-7 FPGAs
 - <http://www.xilinx.com/support/answers/51456.htm>

➤ AC701 Documentation

- Artix-7 FPGA AC701 Evaluation Kit
 - <http://www.xilinx.com/products/boards-and-kits/ek-a7-ac701-g.html>
- AC701 Getting Started Guide – UG967
 - http://www.xilinx.com/support/documentation/boards_and_kits/ac701/2014_3/ug967-ac701-eval-kit-getting-started.pdf
- AC701 User Guide – UG952
 - http://www.xilinx.com/support/documentation/boards_and_kits/ac701/ug952-ac701-a7-eval-bd.pdf