Ultra96: Boot from microSD using FSBL

Overview

Thus far, we have relied on the tools to configure the ZU+ MPSoC PS properly. Although it wasn't explicitly pointed out previously, performing a *Run As* or *Debug As* operation in SDK first sources the `psu_init.tcl` file that was created during the export to SDK. In a true, embedded application, you will not have a JTAG cable connection that can transfer these settings. Your code must be able to do this before transferring control to an application. The code that sets up the ZU+ MPSoC PS is called the First Stage Boot Loader (FSBL).

Objectives

When this tutorial is complete, you will be able to:

- Create the FSBL
- Prepare the boot image
- Write and boot from microSD

Experiment Setup

Software

The software used to test this reference design is:

- Windows-7 64-bit
- Xilinx SDK 2018.2
- USB-JTAG and USB-UART drivers

Hardware

The hardware setup used to test this reference design includes:

- Win-7 PC with the following recommended memory[1]
  - 4 GB Typical and 5 GB Peak RAM available for the Xilinx tools to complete a XCZU3EG design
- Ultra96
- 96Boards Power Supply
- microSD Card Adapter
- USB-UART
  - Avnet USB-to-JTAG/UART Pod (available September 2018)
  - Any other USB-UART dongle

[1] Refer to https://www.xilinx.com/products/design-tools/vivado/memory.html

Copyright © 2018 Avnet, Inc. AVNET, "Reach Further," is a registered
trademark of Avnet, Inc. All other brands are the property of their respective owners.

Experiment 1: Create the FSBL

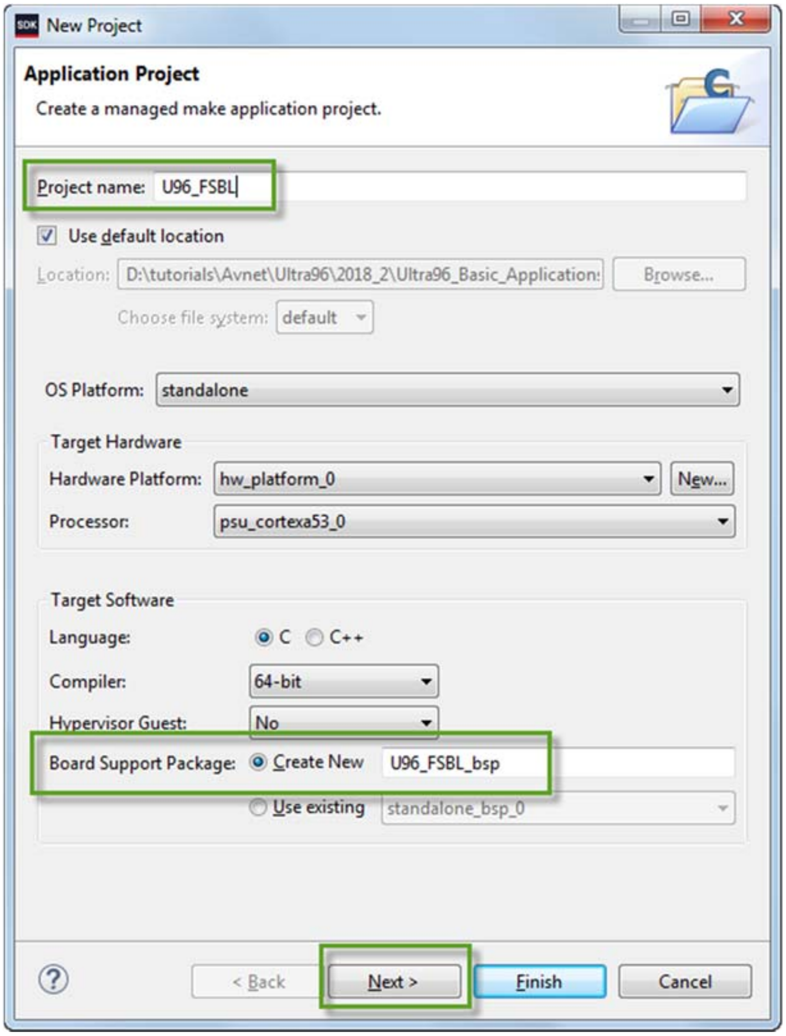
The first step is to create the FSBL application. This is a C program that embeds all the ZU+ MPSoC internal register settings that were established during the Vivado Block Design.

Similar to the flow for creating the Hello_World application, we will use SDK to generate a First Stage Bootloader application.

1. Launch SDK and open the workspace from the Hello World project.

2. **File → New → New Application Project**

3. Name it something like U96_FSBL and choose **Create New** BSP. The reason for creating a new BSP is that the FSBL BSP requires a library for the Flash, and the tools will automatically include this for us if we allow it to create the BSP. Click **Next**



New Project

Application Project

Create a managed make application project.

Project name: U96_FSBL

☑ Use default location

Location: D:\tutorials\Avnet\Ultra96\2018_2\Ultra96_Basic_Application

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: hw_platform_0

Processor: psu_cortexa53_0

Target Software

Language: ● C ○ C++

Compiler: 64-bit

Hypervisor Guest: No

Board Support Package: ● Create New U96_FSBL_bsp

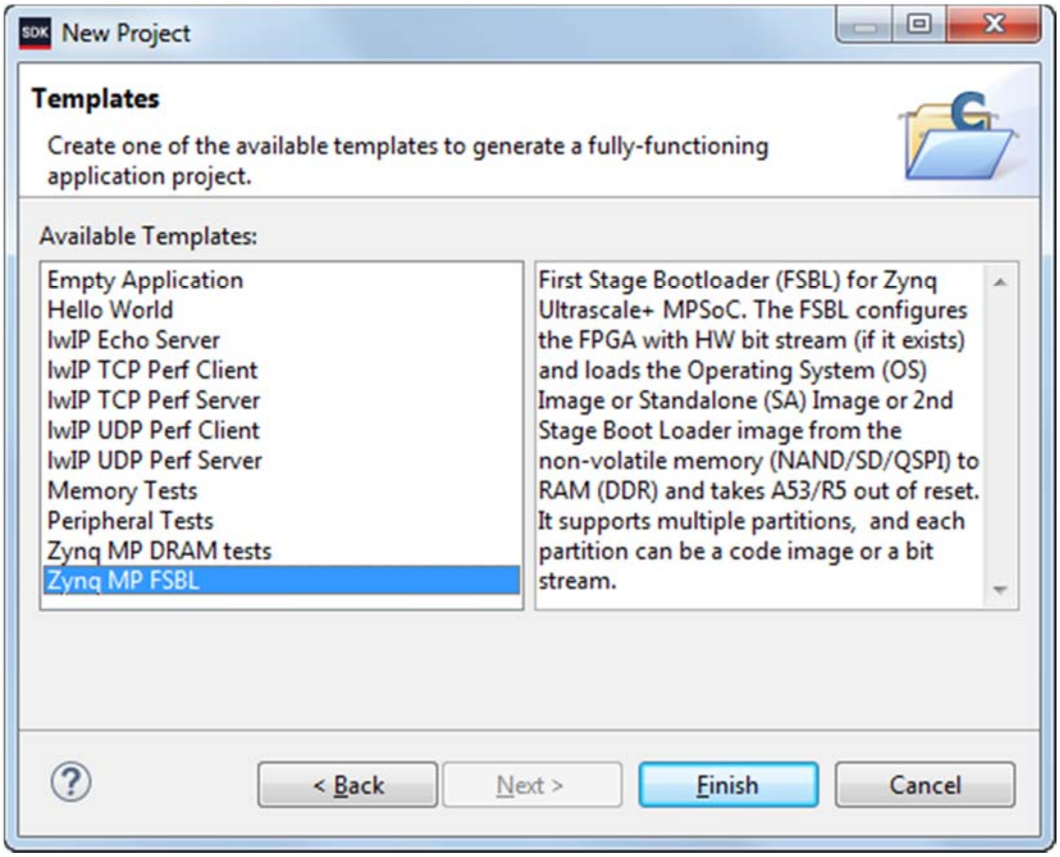
○ Use existing standalone_bsp_0

Figure 1 – FSBL Application

Copyright © 2018 Avnet, Inc. AVNET, “Reach Further,” is a registered
trademark of Avnet, Inc. All other brands are the property of their respective owners.

4. Select **Zynq MP FSBL**

5. Click **Finish**



Templates

Create one of the available templates to generate a fully-functioning application project.

Available Templates:

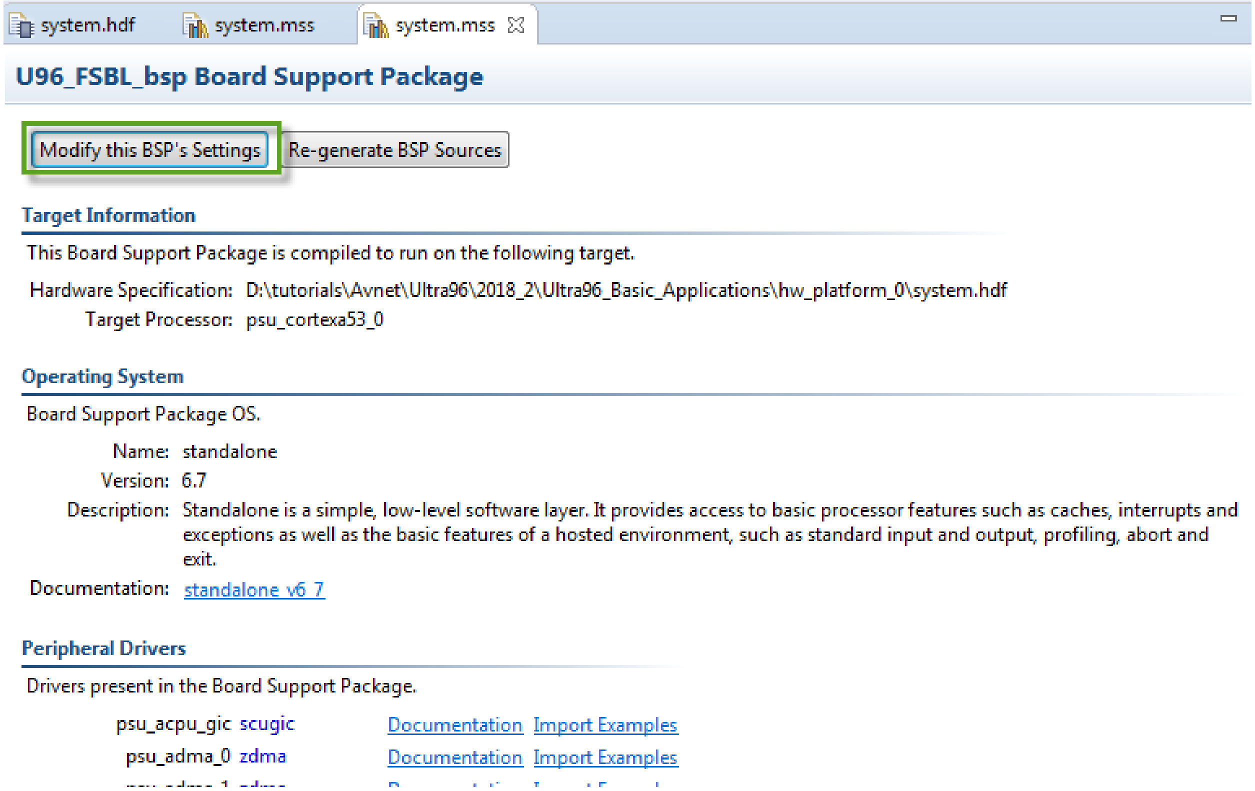
- Empty Application
- Hello World
- lwIP Echo Server
- lwIP TCP Perf Client
- lwIP TCP Perf Server
- lwIP UDP Perf Client
- lwIP UDP Perf Server
- Memory Tests
- Peripheral Tests
- Zynq MP DRAM tests
- Zynq MP FSBL

First Stage Bootloader (FSBL) for Zynq Ultrascale+ MPSoC. The FSBL configures the FPGA with HW bit stream (if it exists) and loads the Operating System (OS) Image or Standalone (SA) Image or 2nd Stage Boot Loader image from the non-volatile memory (NAND/SD/QSPI) to RAM (DDR) and takes A53/R5 out of reset. It supports multiple partitions, and each partition can be a code image or a bit stream.

Figure 2 – ZU+ MPSoC FSBL Template for Application

Copyright © 2018 Avnet, Inc. AVNET, “Reach Further,” is a registered
trademark of Avnet, Inc. All other brands are the property of their respective owners.

6. Similar to the standalone_bsp_0 we created in lab 2, we must modify the BSP's UART settings to properly output our results through the serial port. In the U96_FSBL_bsp system.mss file **click** on "Modify this BSP's Settings".



U96_FSBL_bsp Board Support Package

Target Information

This Board Support Package is compiled to run on the following target.

Hardware Specification: D:\tutorials\Avnet\Ultra96\2018_2\Ultra96_Basic_Applications\hw_platform_0\system.hdf

Target Processor: psu_cortexa53_0

Operating System

Board Support Package OS.

Name: standalone

Version: 6.7

Description: Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.

Documentation: standalone_v6_7

Peripheral Drivers

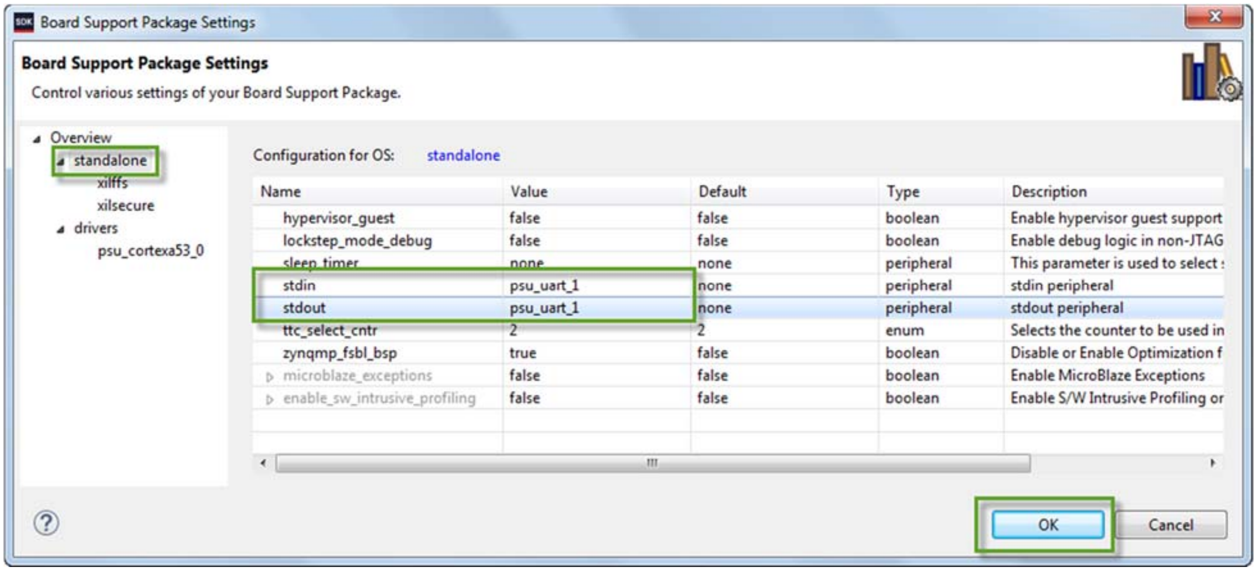
Drivers present in the Board Support Package.

psu_acpu_gic	scugic	Documentation	Import Examples
psu_adma_0	zdma	Documentation	Import Examples

Figure 3 -- Modify BSP

Copyright © 2018 Avnet, Inc. AVNET, "Reach Further," is a registered trademark of Avnet, Inc. All other brands are the property of their respective owners.

7. In the *Board Support Package Settings*, select standalone. Change the stdin and stdout Value to ps7_uart. This is done to align with the Ultra96's UART Serial connection. Select OK.



Board Support Package Settings

Board Support Package Settings

Control various settings of your Board Support Package.

- Overview
  - standalone
    - xilffs
    - xilsecure
  - drivers
    - psu_cortexa53_0

Configuration for OS: standalone

Name	Value	Default	Type	Description
hypervisor_guest	false	false	boolean	Enable hypervisor guest support
lockstep_mode_debug	false	false	boolean	Enable debug logic in non-JTAG
sleep_timer	none	none	peripheral	This parameter is used to select
stdin	psu_uart_1	none	peripheral	stdin peripheral
stdout	psu_uart_1	none	peripheral	stdout peripheral
ttc_select_cntr	2	2	enum	Selects the counter to be used in
zynqmp_fsbl_bsp	true	false	boolean	Disable or Enable Optimization f
microblaze_exceptions	false	false	boolean	Enable MicroBlaze Exceptions
enable_sw_intrusive_profiling	false	false	boolean	Enable S/W Intrusive Profiling or

Figure 4 – Modify stdin/stdout

Based on the modified settings in SDK, the BSP will automatically be built once it is added to the project. This may take a minute to compile the new BSP. The progress may be seen in the *Console* tab.

Copyright © 2018 Avnet, Inc. AVNET, "Reach Further," is a registered
trademark of Avnet, Inc. All other brands are the property of their respective owners.

Experiment 2: Prepare the Boot Image

The next step is to create a non-volatile boot image. Ultra96 has one non-volatile, primary bootable source, microSD.

1. In SDK, select Periph_Test.

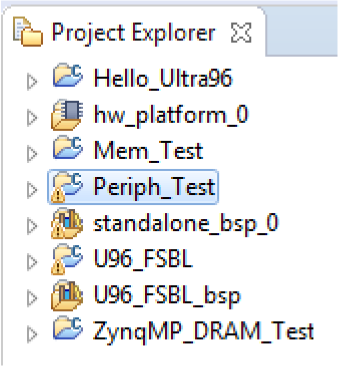


Figure 5 – Select Application to Boot

2. Select **Xilinx → Create Boot Image**.

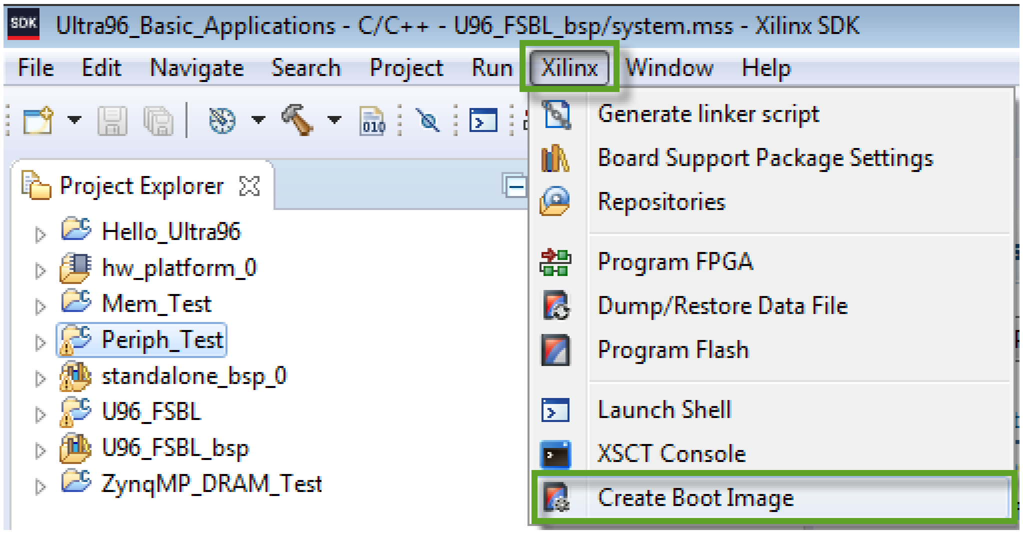
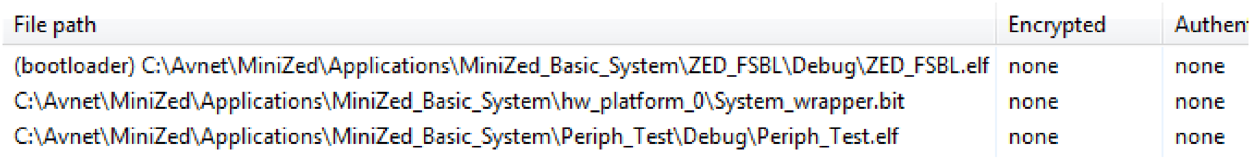


Figure 6 - Create ZU+ MPSoC Boot Image

Copyright © 2018 Avnet, Inc. AVNET, "Reach Further," is a registered trademark of Avnet, Inc. All other brands are the property of their respective owners.

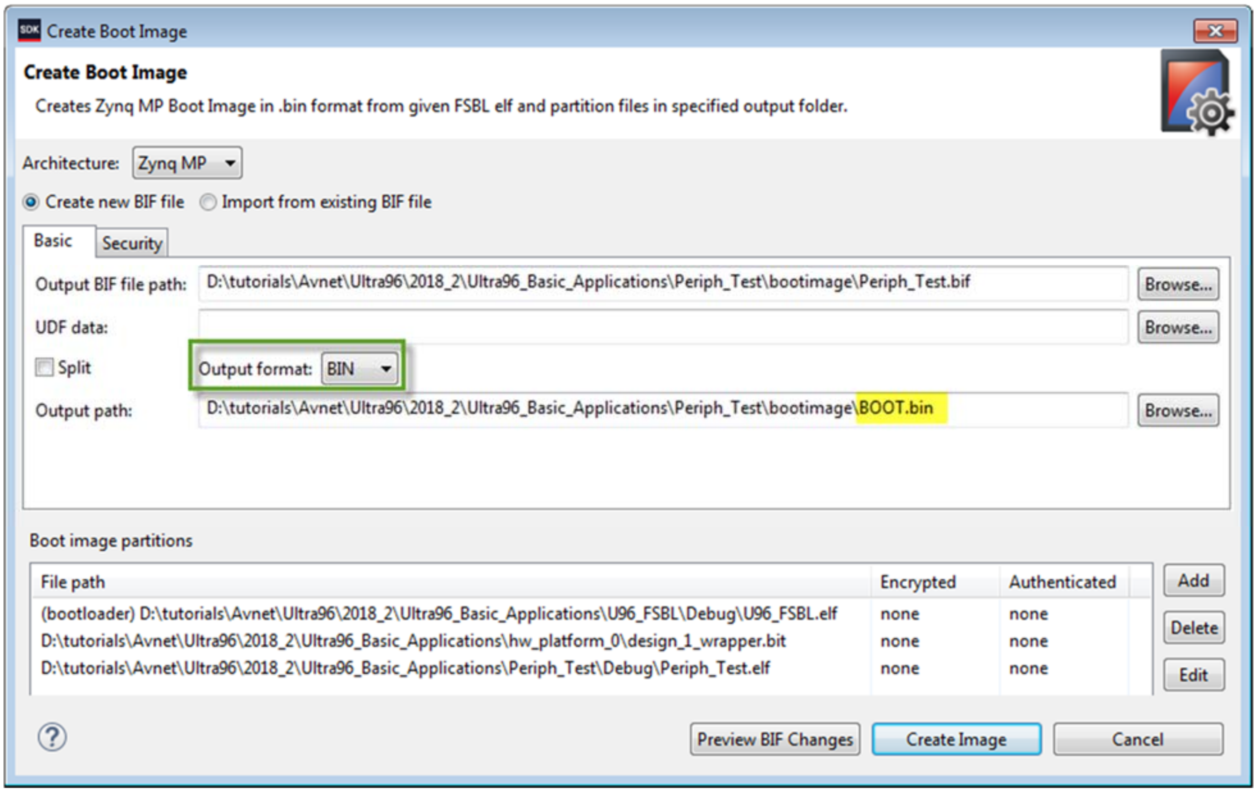
This will preload the FSBL ELF, bitstream, and Application ELF images. The order of the files is important. The FSBL is first, followed by the bitstream, followed by the Application. One function of the FSBL is to program the PL. After the PL is configured, the application is loaded.



File path	Encrypted	Authen
(bootloader) C:\Avnet\MiniZed\Applications\MiniZed_Basic_System\ZED_FSBL\Debug\ZED_FSBL.elf	none	none
C:\Avnet\MiniZed\Applications\MiniZed_Basic_System\hw_platform_0\System_wrapper.bit	none	none
C:\Avnet\MiniZed\Applications\MiniZed_Basic_System\Periph_Test\Debug\Periph_Test.elf	none	none

Figure 7 – Boot Image Partitions

Notice that by default, the output path points to BOOT.bin (highlighted), which is the bootimage required for SD boot, which is what we want. In other systems with QSPI, you may change the Output format to MCS.



Create Boot Image

Create Boot Image

Creates Zynq MP Boot Image in .bin format from given FSBL elf and partition files in specified output folder.

Architecture: Zynq MP

● Create new BIF file ○ Import from existing BIF file

Output BIF file path: D:\tutorials\Avnet\Ultra96\2018_2\Ultra96_Basic_Applications\Periph_Test\bootimage\Periph_Test.bif

UDF data:

□ Split

Output format: BIN

Output path: D:\tutorials\Avnet\Ultra96\2018_2\Ultra96_Basic_Applications\Periph_Test\bootimage\BOOT.bin

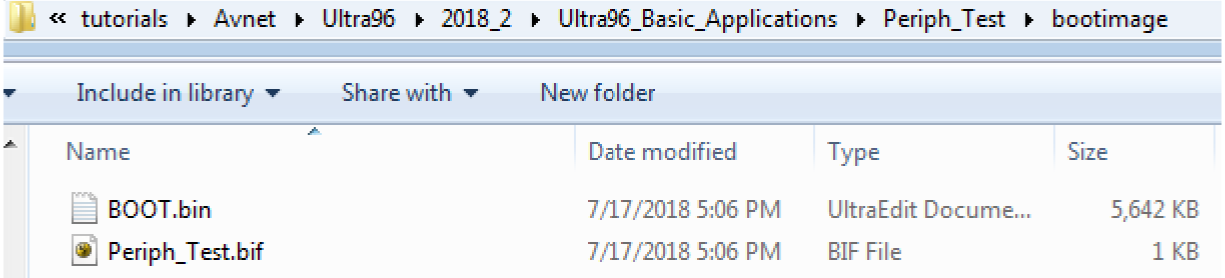
Boot image partitions

File path	Encrypted	Authenticated
(bootloader) D:\tutorials\Avnet\Ultra96\2018_2\Ultra96_Basic_Applications\U96_FSBL\Debug\U96_FSBL.elf	none	none
D:\tutorials\Avnet\Ultra96\2018_2\Ultra96_Basic_Applications\hw_platform_0\design_1_wrapper.bit	none	none
D:\tutorials\Avnet\Ultra96\2018_2\Ultra96_Basic_Applications\Periph_Test\Debug\Periph_Test.elf	none	none

Figure 8 – Create Image

3. Click **Create Image**.

4. Using Windows Explorer, navigate to the application directory, then into `Periph_Test`, then into the newly created **bootimage** directory. Notice that two files have been created: Periph_Test.bif and BOOT.bin. The BOOT.bin is all that is required to boot this bare metal application from microSD.


« tutorials ▸ Avnet ▸ Ultra96 ▸ 2018_2 ▸ Ultra96_Basic_Applications ▸ Periph_Test ▸ bootimage
Include in library ▾
Share with ▾
New folder

Name	Date modified	Type	Size
BOOT.bin	7/17/2018 5:06 PM	UltraEdit Docume...	5,642 KB
Periph_Test.bif	7/17/2018 5:06 PM	BIF File	1 KB

Figure 9 - Bootimage Directory

5. Copy BOOT.bin to a FAT-formatted microSD card.

Copyright © 2018 Avnet, Inc. AVNET, “Reach Further,” is a registered trademark of Avnet, Inc. All other brands are the property of their respective owners.

Experiment 3: Write and boot from microSD

First, we will program the microSD with the BIN file using our Host system.

1. Insert the microSD card into the Ultra96 microSD Card Cage (J4).
2. Set the Ultra96 boot mode switch SW2 to SD mode () as shown below.

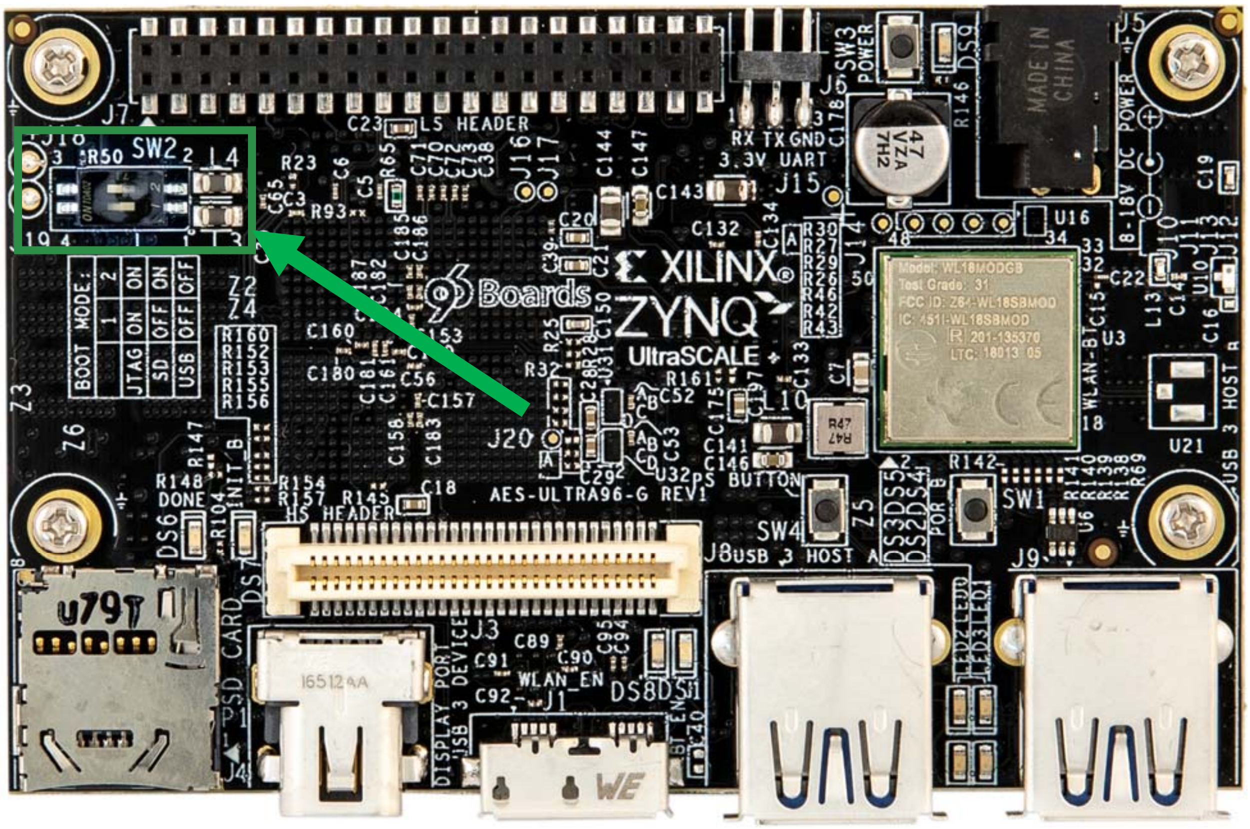


Figure 10 – Ultra96 Switch Location

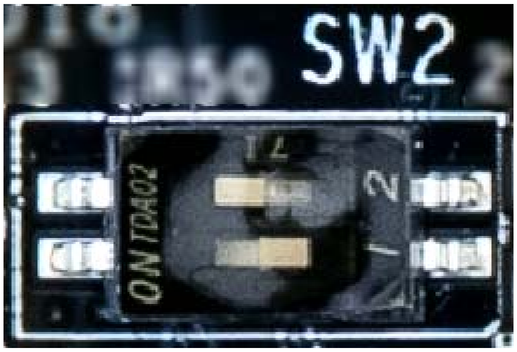


Figure 11 – microSD Boot Mode

Copyright © 2018 Avnet, Inc. AVNET, "Reach Further," is a registered
trademark of Avnet, Inc. All other brands are the property of their respective owners.

3. Close or disconnect the terminal that may have previously been open on your PC.

4. Connect a USB-UART dongle to the 3-pin J6 and a USB-JTAG dongle to the 7-pin J2 on the Ultra96.

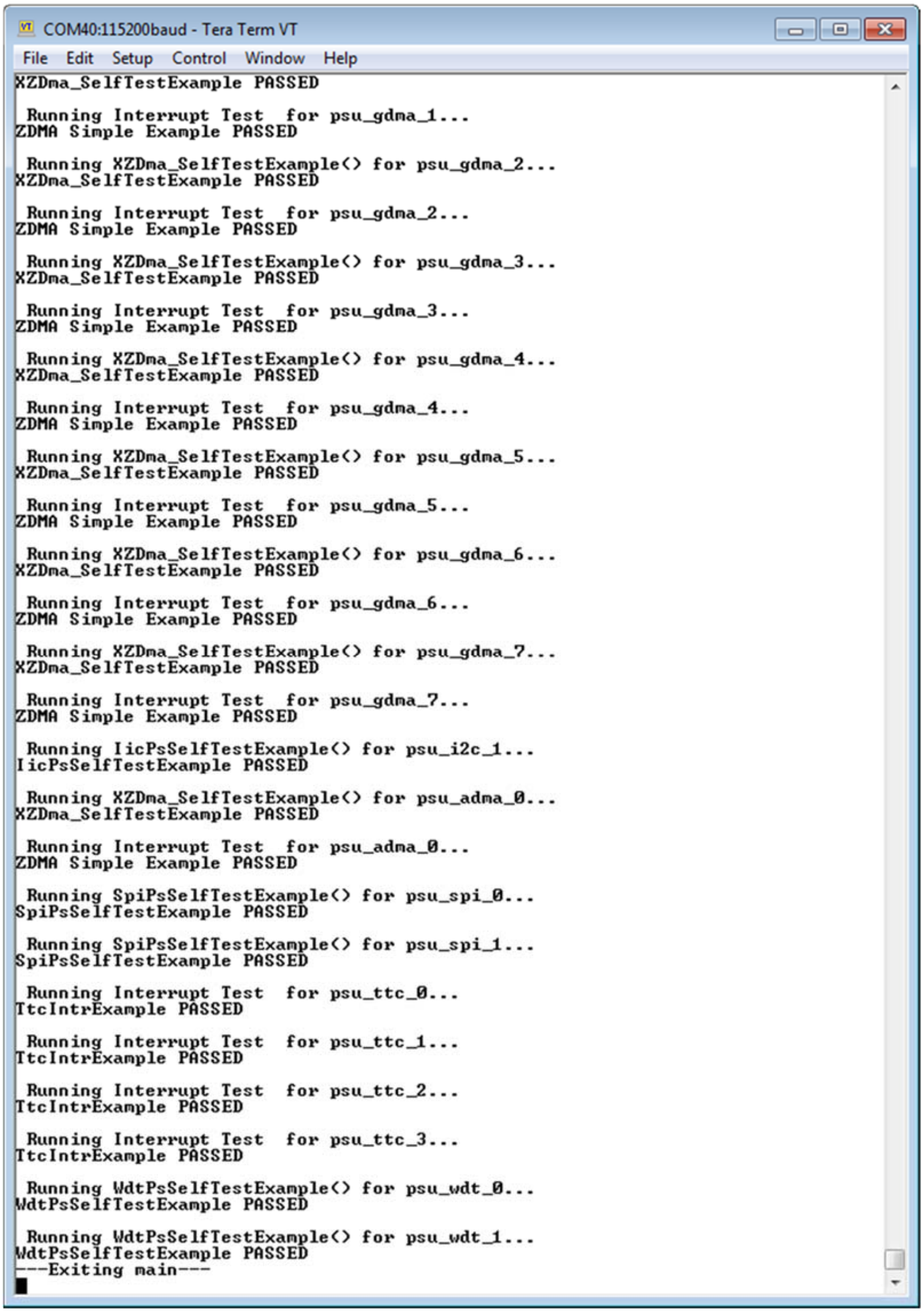
5. Plug in the 96Boards Power Supply to the barrel jack J5 on Ultra96.

6. Press and release the SW3 Power On button. The Green Power On LED (DS9) and the 4 Green User LEDs (between the two Type A USB ports) should illuminate. After a brief moment, the Green DS6 DONE LED should light.

7. Launch a terminal program (TeraTerm) with the 115200/8/n/1/n settings.

8. Push the RESET POR_B button (SW1). You should see the results in the terminal.

Copyright © 2018 Avnet, Inc. AVNET, “Reach Further,” is a registered
trademark of Avnet, Inc. All other brands are the property of their respective owners.


COM40:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_gdma_1...
ZDMA Simple Example PASSED

 Running XZDma_SelfTestExample() for psu_gdma_2...
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_gdma_2...
ZDMA Simple Example PASSED

 Running XZDma_SelfTestExample() for psu_gdma_3...
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_gdma_3...
ZDMA Simple Example PASSED

 Running XZDma_SelfTestExample() for psu_gdma_4...
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_gdma_4...
ZDMA Simple Example PASSED

 Running XZDma_SelfTestExample() for psu_gdma_5...
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_gdma_5...
ZDMA Simple Example PASSED

 Running XZDma_SelfTestExample() for psu_gdma_6...
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_gdma_6...
ZDMA Simple Example PASSED

 Running XZDma_SelfTestExample() for psu_gdma_7...
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_gdma_7...
ZDMA Simple Example PASSED

 Running IicPsSelfTestExample() for psu_i2c_1...
IicPsSelfTestExample PASSED

 Running XZDma_SelfTestExample() for psu_adma_0...
XZDma_SelfTestExample PASSED

 Running Interrupt Test  for psu_adma_0...
ZDMA Simple Example PASSED

 Running SpiPsSelfTestExample() for psu_spi_0...
SpiPsSelfTestExample PASSED

 Running SpiPsSelfTestExample() for psu_spi_1...
SpiPsSelfTestExample PASSED

 Running Interrupt Test  for psu_ttc_0...
TtcIntrExample PASSED

 Running Interrupt Test  for psu_ttc_1...
TtcIntrExample PASSED

 Running Interrupt Test  for psu_ttc_2...
TtcIntrExample PASSED

 Running Interrupt Test  for psu_ttc_3...
TtcIntrExample PASSED

 Running WdtPsSelfTestExample() for psu_wdt_0...
WdtPsSelfTestExample PASSED

 Running WdtPsSelfTestExample() for psu_wdt_1...
WdtPsSelfTestExample PASSED
---Exiting main---
```

Figure 12 – Results from microSD boot of Periph_Test

9. Close the terminal. Power off the board.

Copyright © 2018 Avnet, Inc. AVNET, “Reach Further,” is a registered
trademark of Avnet, Inc. All other brands are the property of their respective owners.

Revision History

Date	Version	Revision
19 Jul 2018	2018_2.01	Initial Avnet release for Vivado 2018.2