

27 March 2017 Version 2016\_4.01

#### Overview

Thus far, we have relied on the tools to configure the Zynq PS properly. Although it wasn't explicitly pointed out previously, performing a *Run As* or *Debug As* operation in SDK first sources the ps7\_init.tcl file that was created during the export to SDK. In a true, embedded application, you will not have a JTAG cable connected 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 Zynq 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 QSPI
- Write and boot from the microSD Card

© 2016 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.



## Experiment Setup

#### Software

The software used to test this reference design is:

- Windows-7 64-bit
- Xilinx SDK 2016.2
- Silicon Labs CP201x USB-to-UART Bridge Driver
  - <u>www.microzed.org</u> → Support → Documentation → MicroZed
     Silicon Labs CP210x USB-to-UART Setup Guide
  - Note that MicroZed and both PicoZed FMC Carriers all use the same Silicon Labs CP2104 device, so the setup is the same.
- <u>Tera Term</u> or another terminal emulator
  - We will be booting outside of the SDK environment, so we will use an external terminal as well.
  - For additional information on setting up Tera Term, see the USB-to-UART Setup Guide listed above.

#### Hardware

The hardware setup used to test this reference design includes:

- Win-7 PC with the following recommended memory<sup>1</sup>:
  - 1.6 GB RAM available for the Xilinx tools to complete a XC7Z010 design
  - 2.3 GB RAM available for the Xilinx tools to complete a XC7Z015 design
  - $\circ$   $\,$  1.9 GB RAM available for the Xilinx tools to complete a XC7Z020 design
  - o 2.7 GB RAM available for the Xilinx tools to complete a XC7Z030 design
- One of the following:
  - o Avnet MicroZed 7010 or 7020
  - Avnet PicoZed 7010, 7015, 7020, or 7030 with either the PicoZed FMC Carrier V1 or PicoZed FMC Carrier V2
- USB cable (Type A to Micro-USB Type B)
- JTAG Programming Cable (Platform Cable, Digilent HS1, HS2, or HS3 cable)
  - If you don't already have a JTAG Cable, Avnet recommends the Digilent HS3 Cable
  - o <u>http://www.em.avnet.com/jtaghs3</u>
- microSD card
- microSD card adapter

<sup>&</sup>lt;sup>1</sup> Refer to <u>www.xilinx.com/design-tools/vivado/memory.htm</u>



## **Experiment 1: Create the FSBL**

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

Similar to the flow for creating the Hello\_World application, in SDK create 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 ZED\_FSBL and **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** 

sox New Project					
Application Project Create a managed make application project.					
Project name: ZED_FS					
Use default locatio		Danasa			
	Location: C:\Avnet\MicroZed\Applications\MZ_basic_system\ZED_FSE Browse Choose file system: default				
OS Platform: standa	lone	•			
Target Hardware					
Hardware Platform:	hw_platform_0	▼ New			
Processor:	ps7_cortexa9_0	•			
Target Software					
Language:					
Compiler: 2 Board Support Packa					
	O Use existing standalone_bsp_0				
	3				
?	< Back Next > Finish	Cancel			

Figure 1 – FSBL Application



- 4. Select **Zynq FSBL**
- 5. Click Finish

New Project Templates Create one of the available templates to gene application project.	erate a fully-functioning
Available Templates: Dhrystone Empty Application Hello World IwIP Echo Server Memory Tests OpenAMP echo-test OpenAMP matrix multiplication Demo OpenAMP RPC Demo Peripheral Tests RSA Authentication App Zynq DRAM tests Zynq FSBL	First Stage Bootloader (FSBL) for Zynq. 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/NOR/QSPI) to RAM (DDR) and starts executing it. It supports multiple partitions, and each partition can be a code image or a bit stream.
C Sack Ne	ext > Finish Cancel

Figure 2 – Zynq FSBL Template for Application



### **Experiment 2: Prepare the Boot Image**

The next step is to create a non-volatile boot image. MicroZed and PicoZed + FMC Carrier both have two non-volatile, primary bootable sources, QSPI and SD Card.

1. In SDK, select Periph\_Test.

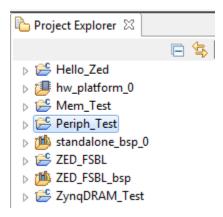


Figure 3 – Select Application to Boot

2. Select Xilinx Tools → Create Boot Image.

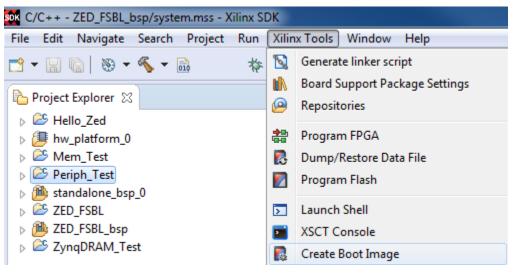


Figure 4 - Create Zynq Boot Image



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.

Boot image partitions		
File path	Encrypted	Auther
(bootloader) C:\Avnet\MicroZed\Applications\MZ_Basic_System\ZED_FSBL\Debug\ZED_FSBL.elf	none	none
C:\Avnet\MicroZed\Applications\MZ_Basic_System\hw_platform_0\System_wrapper.bit	none	none
$\label{eq:c:Avnet} C: Avnet \\ MicroZed \\ Applications \\ MZ_Basic_System \\ Periph_Test \\ Debug \\ Periph_Test.elf$	none	none

Figure 5 – Boot Image Partitions

Notice that by default, the output path points to BOOT.bin (highlighted), which is the bootimage required for SD boot. We will create this one first, and then we will need to repeat the operation for the QSPI bootimage (mcs).

Sok Create Boot Image				×
Create Boot Image Creates Zynq Boot Image in .bin format from given FSBL elf and p	artition files in s	pecified output fold	ler.	<b>1</b>
Architecture: Zynq				
Create new BIF file     Import from existing BIF file     Basic Security				
Output BIF file path: C:\Avnet\MicroZed\Applications\MZ_Basic	c_System\Periph	_Test\bootimage\F	eriph_Test.bif	Browse
UDF data:				Browse
Split Output format: BIN V	Split Output format: BIN			
Output path:         C:\Avnet\MicroZed\Applications\MZ_Basic_System\Periph_Test\bootimage\BOOT.bin         Browse				
Boot image partitions				
File path	Encrypted	Authenticated		Add
(bootloader) C:\Avnet\MicroZed\Applications\MZ_Basic_Syste	none	none		Delete
C:\Avnet\MicroZed\Applications\MZ_Basic_System\hw_platfo C:\Avnet\MicroZed\Applications\MZ_Basic_System\Periph_Te		none		Edit
				Up
Previe	ew BIF Changes	Create Image	e Ca	ncel

Figure 6 – Create Image



- 3. Click Create Image.
- 4. Select Xilinx Tools → Create Boot Image again.
- 5. Click **Browse** next to the *Output Path*. To keep things consistent, browse and select the previously used bootimage directory.
- $\texttt{C:Avnet}MicroZedApplicationsMZ\_Basic\_SystemPeriph\_Testbootimage$
- 6. Change the *Save as type* to **\*.mcs**. Type in **Periph\_Test.mcs** for the *File name* and then click **Open**.

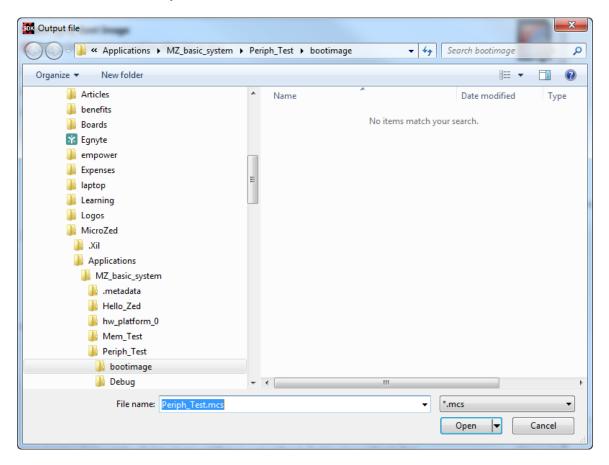


Figure 7 – Set Bootimage Output File to Periph\_Test.mcs



The resulting Bootimage dialog is shown below. Notice that we do not need to regenerate the BIF file since it was created when BOOT.bin was generated.

Sok Create Boot Image					×
	ts at the specified path and will be overwritte view the changes in bif contents before over		ied contents. Use	'Preview Bif	<b>K</b> ê
Architecture: Zynq	<ul> <li>Import from existing BIF file</li> </ul>				
Import BIF file path:	C:\Avnet\MicroZed\Applications\MZ_Basic_S	System\Periph_T	est\bootimage\Pe	riph_Test.bif	Browse
Basic Security					
Output BIF file path:	h: C:\Avnet\MicroZed\Applications\MZ_Basic_System\Periph_Test\bootimage\Periph_Test.bif Browse				
UDF data:	Browse			Browse	
Split	Output format: MCS 🔻				
Output path:	C:\Avnet\MicroZed\Applications\MZ_Basic_System\Periph_Test\bootimage\Periph_Test.mcs				
Boot image partitions					
File path		Encrypted	Authenticated		Add
(bootloader) C:\Avnet\MicroZed\Applications\MZ_Basic_Syste		none	none		Delete
C:\Avnet\MicroZed\Applications\MZ_Basic_System\hw_platfo		none	none		Delete
C:\Avnet\MicroZed\	Applications\MZ_Basic_System\Periph_Te	none	none		Edit
?	Previo	ew BIF Changes	Create Imag	je Car	cel

Figure 8 – Creating the QSPI Bootimage

7. Click Create Image.



8. Using Windows Explorer, navigate to the application directory, then into Periph\_Test, then into the newly created **bootimage** directory. Notice that three files have been created: .bif, .bin, and .mcs. These are all the required files to program either the QSPI or SD Card.

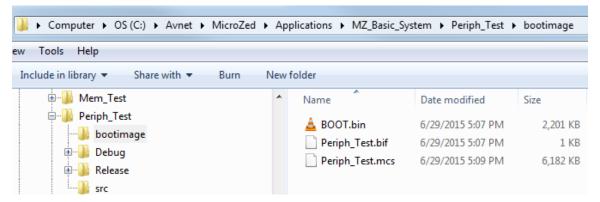


Figure 9 - Bootimage Directory



#### **Experiment 3: Write and boot from QSPI**

First, we will program the QSPI with the MCS file from within SDK.

- 1. Set up the MicroZed or PicoZed for Cascaded JTAG mode as before. Attach the JTAG cable. Power on.
- 2. In SDK, select Xilinx Tools → Program Flash.
- 3. Click the **Browse** button, and browse to the MCS image for the application you wish to program. Select it and click **Open**. You can selectively enable the *Blank check after erase* and/or the *Verify after flash* functions although each of these will add time to the procedure. Neither is required to program the Flash.

SDK	
Program Flash Me	emory
Program Flash Mer	mory via In-system Programmer.
Hardware Platform:	hw_platform_0
Connection:	Local
Device:	Auto Detect Select
Image File:	C:\Avnet\MicroZed\Applications\MZ_Basic_System\Periph_Test\bootimage\Periph_Test.mcs
Offset:	
Flash Type	qspi_single •
FSBL File:	Browse
Convert ELF to bo	otloadable SREC format and program
Blank check after	erase
🔲 Verify after flash	
?	Program Cancel

4. Click **Program**.

#### Figure 10 – Program QSPI Flash Memory

 The operation took approximately 50 seconds on a Win7 PC with a Digilent HS3 JTAG Cable for a 7010. You should see the following in the Program Flash Console window (if you see nothing in the console window, click the console pull-down ■ ■ and select Program Flash).



Flash Operation Successful

#### Figure 11 – QSPI Programmed Successfully

- 6. Power off the board. The JTAG cable may be unplugged.
- 7. Set the Boot Jumpers to QSPI
  - a. MicroZed: move JP3 to the 2-3 position.



Figure 12 – MicroZed QSPI Boot Mode

b. PicoZed: set the boot mode switch SW1 to QSPI mode as shown below.



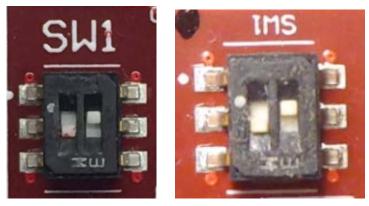


Figure 13 – PicoZed SOM SW1 Set to QSPI Boot 7010/20 on the Left; 7015/30 on the Right

- 8. Close or disconnect the terminal that may have previously been open on your PC.
- 9. Power on the board. The blue DONE LED should both be lit.
- 10. Plug in the USB-UART cable.
- 11. Launch a terminal program (TeraTerm) with the 115200/8/n/1/n settings.
- 12. Push the RST button. You should see the results in the terminal.

MicroZed/PicoZed: FSBL and Boot from QSPI and SD Card v2016\_4.01 27 March 2017



COM6:115200baud - Tera Term VT	
File Edit Setup Control Window Help	
Entering main	*
Running ScuGicSelfTestExample() for ps7_scugic_0 ScuGicSelfTestExample PASSED ScuGic Interrupt Setup PASSED	
Running XDmaPs_Example_W_Intr() for ps7_dma_s Test round Ø XDmaPs_Example_W_Intr PASSED	
Running Interrupt Test for ps7_ethernet_0 EmacPsDmaIntrExample PASSED	
Running QspiSelfTestExample() for ps7_qspi_0 QspiPsSelfTestExample PASSED	
Running DcfgSelfTestExample() for ps7_dev_cfg_0 DcfgSelfTestExample PASSED	
Running ScuTimerPolledExample<> for ps7_scutimer_0 ScuTimerPolledExample PASSED	
Running Interrupt Test for ps7_scutimer_0 ScuTimerIntrExample PASSED	
Running Interrupt Test for ps7_scuwdt_0 ScuWdtIntrExample PASSED	
Running Interrupt Test for ps7_ttc_0 TtcIntrExample PASSED Exiting main	
	-

Figure 14 – Results from QSPI boot of Periph\_Test

13. Close the terminal. Power off the board.



#### Experiment 4: Write and boot from the microSD Card

Next, we will prepare the microSD card with the bin file that SDK generated.

- 1. To program a microSD Card, insert the microSD card into an adapter and plug into your PC. Copy the BOOT.bin image to the microSD Card.
- 2. Be aware that the name "boot.bin" is essential. If you changed the name of the .bin file, rename it on the microSD card to **boot.bin** or **BOOT.bin**. THIS IS CRITICAL!
- 3. Eject the microSD card from the PC.
- 4. Insert the microSD card into the board slot.
- 5. Set the Boot Jumpers to SD Card:
  - a. MicroZed: both JP2 and JP3 in the 2-3 position

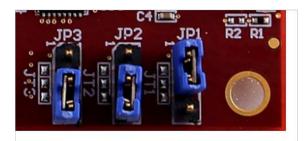


Figure 15 – MicroZed SD Card Boot Mode

b. PicoZed: set the boot mode switch SW1 to SD mode as shown below.

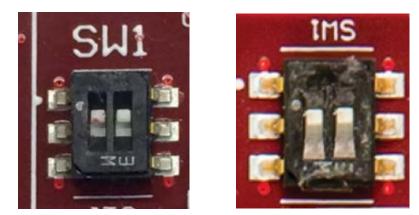


Figure 16 - PicoZed SW1 Set to SD Boot 7010/20 on the Left; 7015/30 on the Right



- 6. Close or disconnect the terminal that may have previously been open on your PC.
- 7. Plug in the USB-UART cable.
- 8. Launch a terminal program with the 115200/8/n/1/n settings.
- 9. Push the RST button. You should see the same results in the terminal that were shown during the QSPI test.



## **Revision History**

Date	Version	Revision
26 Aug 2013	2013_2.01	Initial Avnet release for Vivado 2013.2
06 Sep 2013	2013_2.02	Switching to Release build rather than Debug
10 Jun 2014	2014_1.01	Update to 2014.1. Added instructions that .bin and .mcs
		must each be generated individually now.
11 Jun 2014	2014_2.01	Update to 2014.2. Bitstream must manually be added to
		the Boot Image dialog now.
29 Jun 2015	2015_1.01	Update to 2015.1. Bitstream is no longer manual. Add
		support for PicoZed.
15 Jul 2015	2015_2.01	Update for 2015.2
06 Apr 2016	2015_4.01	Update to 2015.4. Add support for PZCC-FMC-V2.
01 Jun 2016	2015_4.02	Update to 2015.4. Clarified USB to Uart cable connection.
15 Sept 2016	2016_2.01	Updated to 2016.2
20 Jan 2017	2016_4.01	Updated to 2016.4