

MiniZed: Boot from QSPI using FSBL

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 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 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

Experiment Setup

Software

The software used to test this reference design is:

- Windows-7 64-bit
- Xilinx SDK 2018.1
- FTDI FT2232H device driver

Hardware

The hardware setup used to test this reference design includes:

- Win-7 PC with the following recommended memory¹:
 - 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
 - 1.9 GB RAM available for the Xilinx tools to complete a XC7Z020 design
 - 2.7 GB RAM available for the Xilinx tools to complete a XC7Z030 design
- MiniZed
- USB cable (Type A to Micro-USB Type B)

¹ Refer to www.xilinx.com/design-tools/vivado/memory.htm

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**

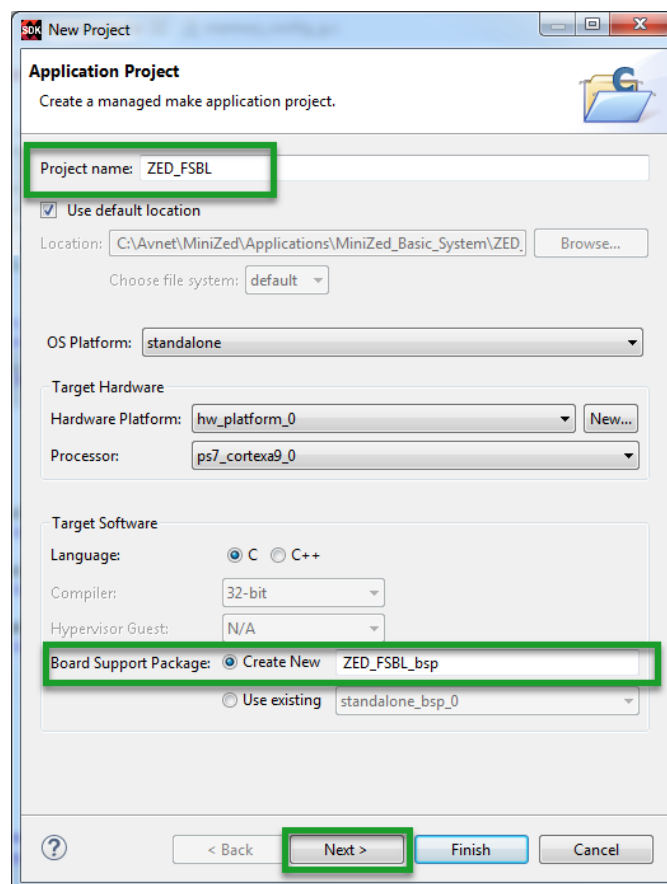


Figure 1 – FSBL Application

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

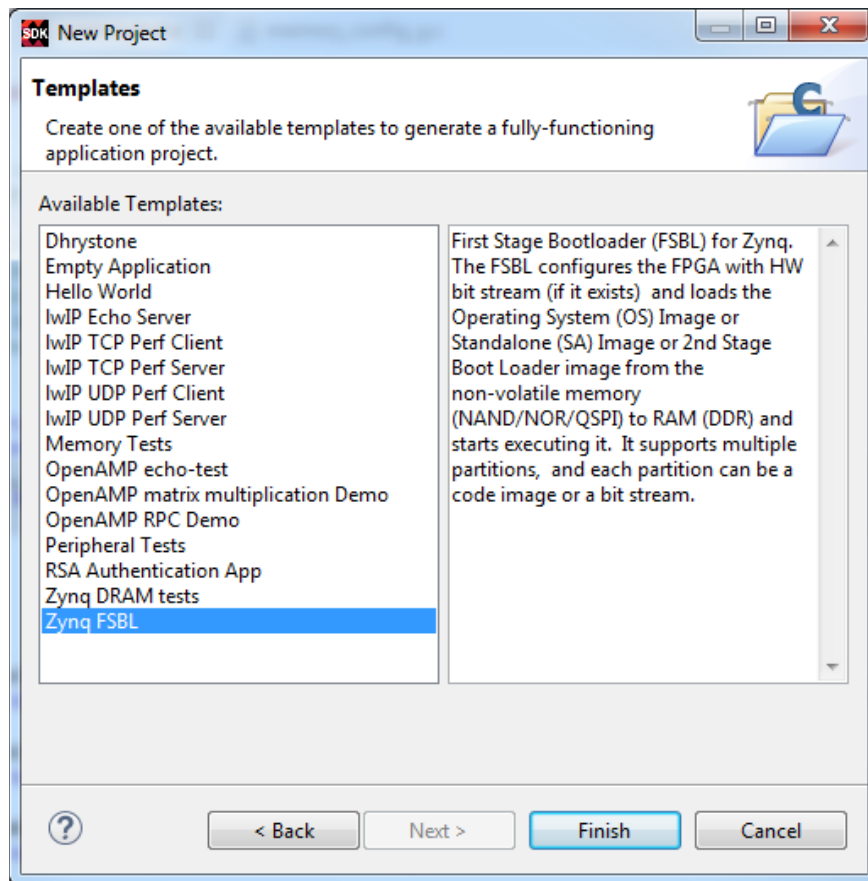


Figure 2 – Zynq FSBL Template for Application

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 ZED_FSBL_bsp system.mss file **click** on "Modify this BSP's Settings".

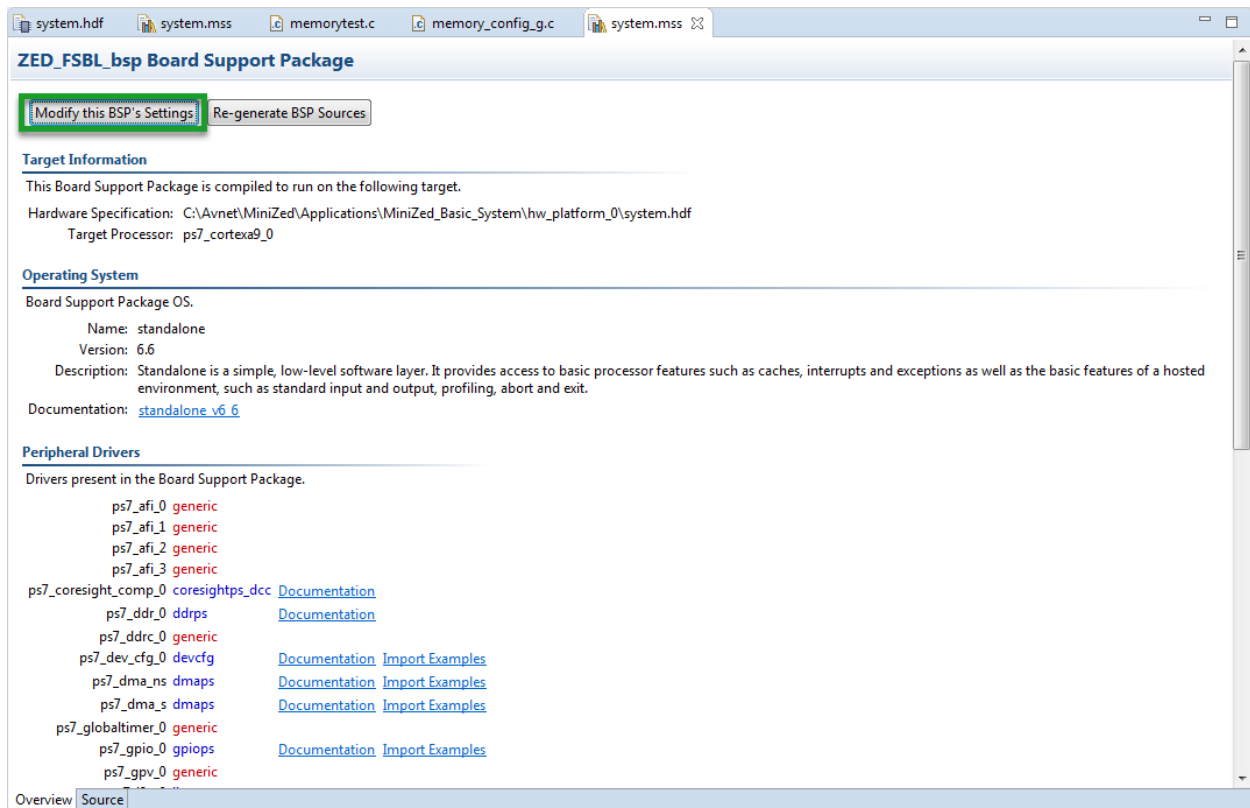


Figure 3 -- Modify BSP

7. In the *Board Support Package Settings*, select standalone

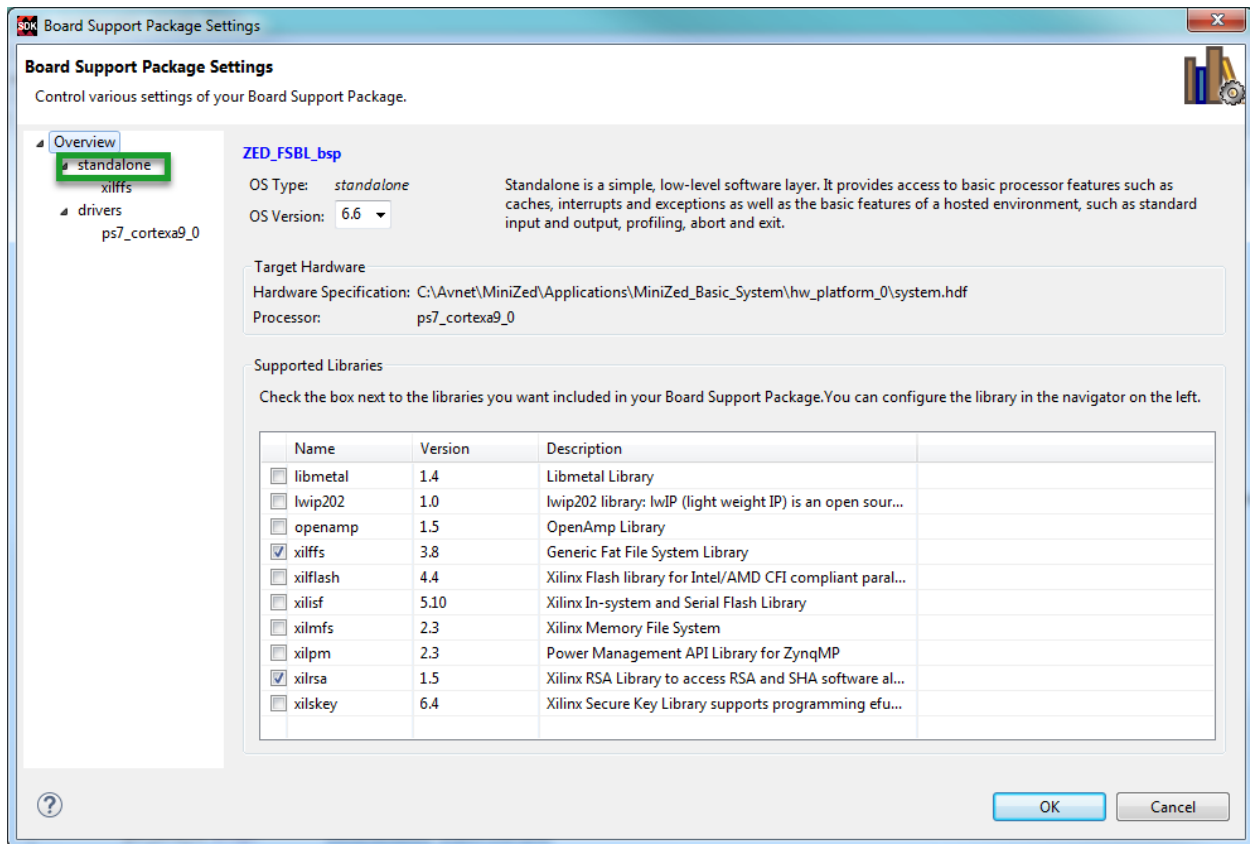


Figure 4 – Board Support Package Settings

- Change the stdin and stdout Value to ps7_uart. This is done to align with the MiniZed's UART Serial connection. Select OK.

| 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 | ps7_uart_1 | none | peripheral | stdin peripheral |
| stdout | ps7_uart_1 | none | peripheral | stdout peripheral |
| ttc_select_cntr | 2 | 2 | enum | Selects the counter to be used in |
| zynqmp_fsbl_bsp | false | 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 5 – 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.

Experiment 2: Prepare the Boot Image

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

1. In SDK, select `Periph_Test`.

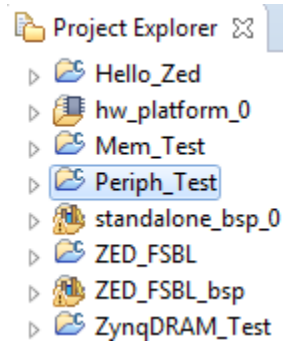


Figure 6 – Select Application to Boot

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

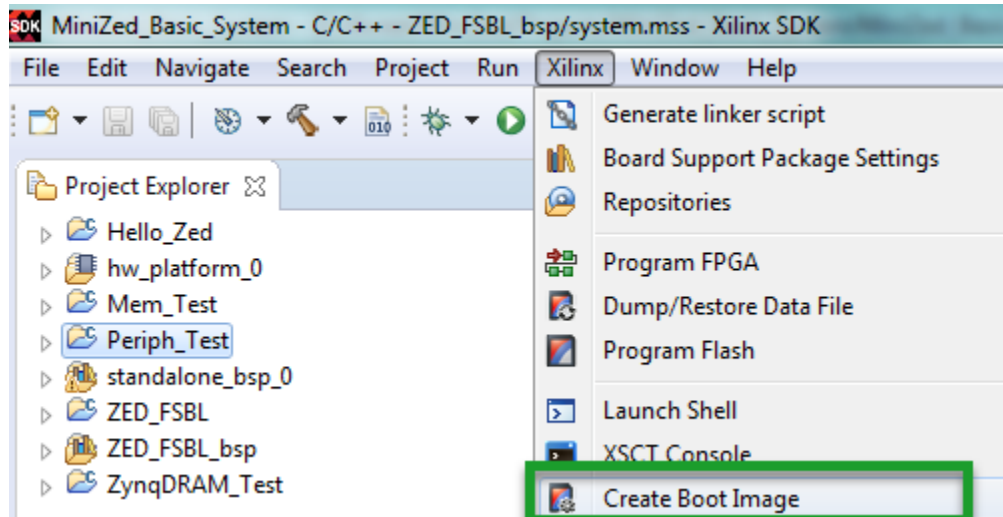


Figure 7 - Create Zynq Boot Image

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.

| 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 8 – Boot Image Partitions

Notice that by default, the output path points to BOOT.bin (highlighted), which is the bootimage required for SD boot. Unfortunately the MiniZed by default is unable to boot from SD Card, the output image must be changed to generate a QSPI bootimage (mcs).

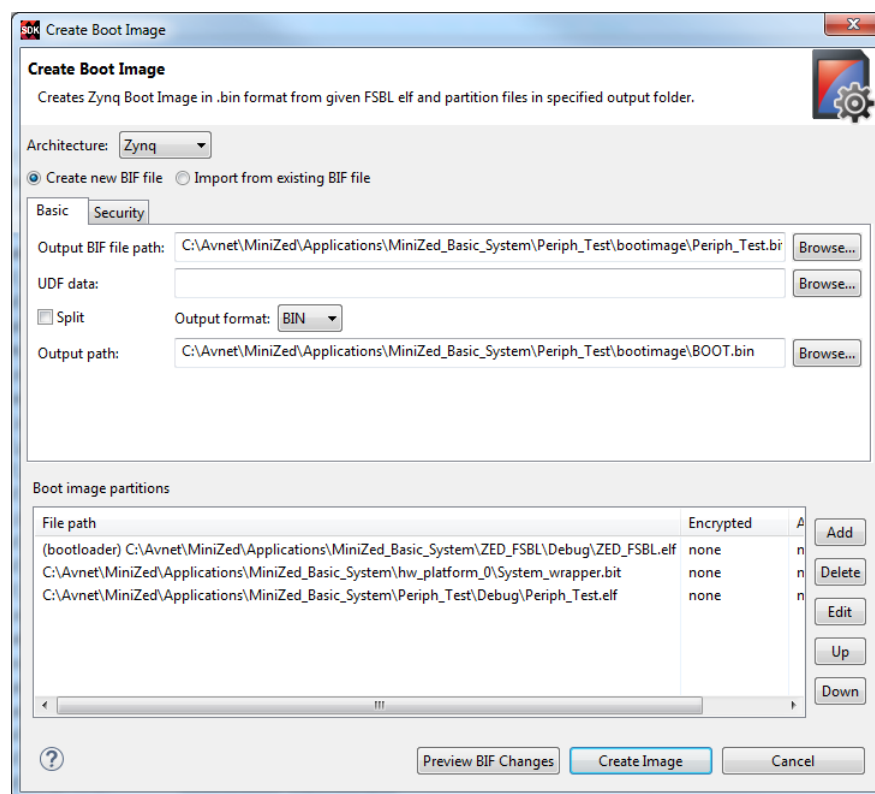


Figure 9 – Create Image

3. Change the *Output format* to ***.mcs**. Change **Boot.mcs** to **Periph_Test.mcs** for the *File name*.

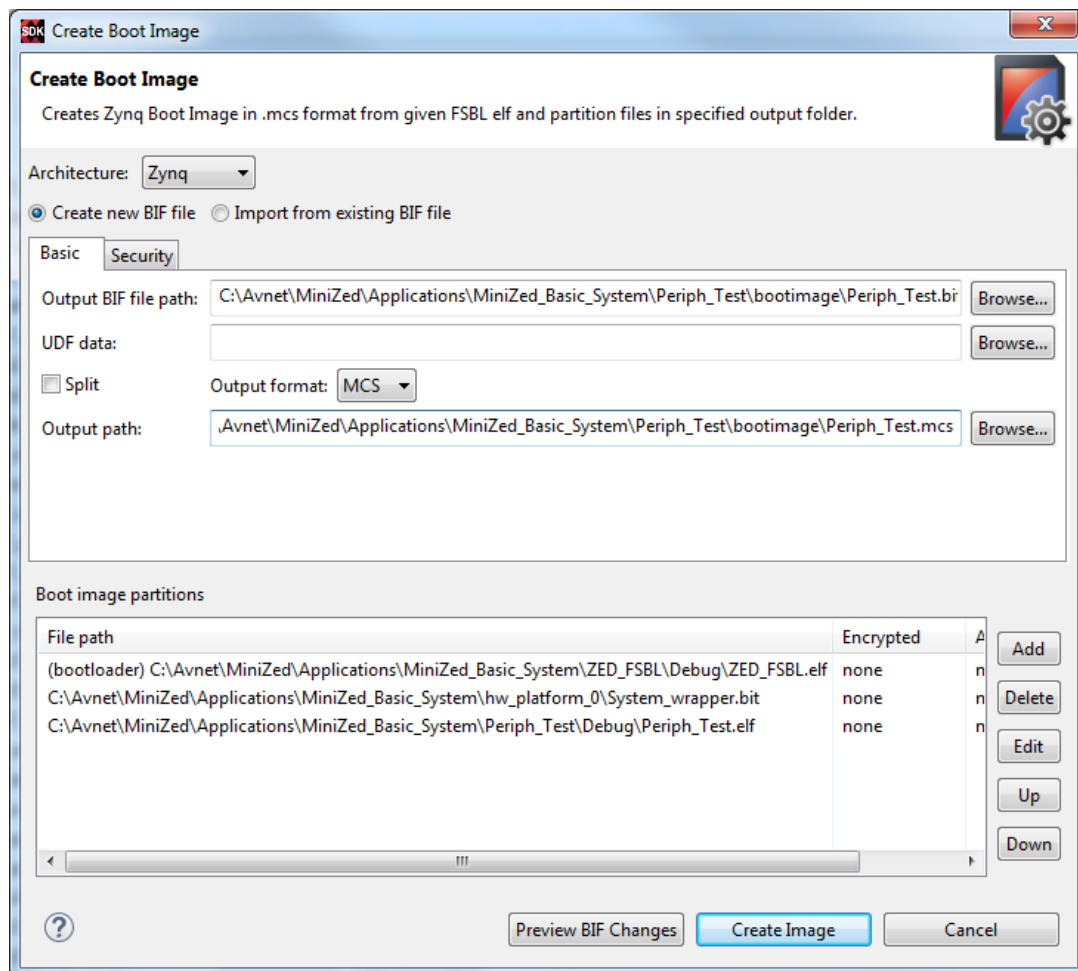


Figure 10 – Creating the QSPI Bootimage

4. Click **Create Image**.
5. 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: `.bif`, `.mcs`. These are all the required files to program the QSPI.

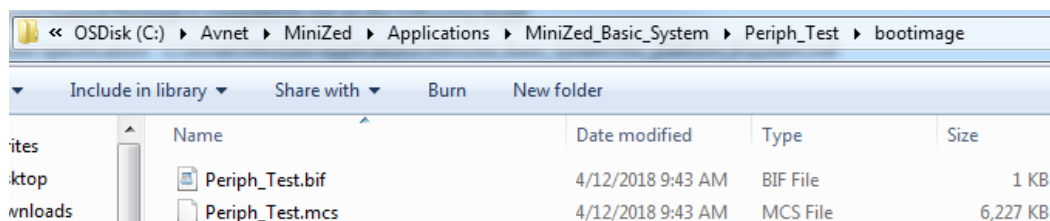


Figure 11 - Bootimage Directory

Experiment 3: Write and boot from QSPI

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

1. Set up the MiniZed for Cascaded JTAG mode as before. Attach the micro-USB to J2 and your computer.
2. In SDK, select **Xilinx Tools** → **Program Flash**.
3. Click the image file **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.
4. Click the FSBL File **Browse** button, and browse to the ZED_FSBL.elf file you generated. This should be located somewhere similar to

C:\Avnet\MiniZed\Applications\MiniZed_Basic_System\ZED_FSBL\Debug\ZED_FSBL.elf

5. Click **Program**.

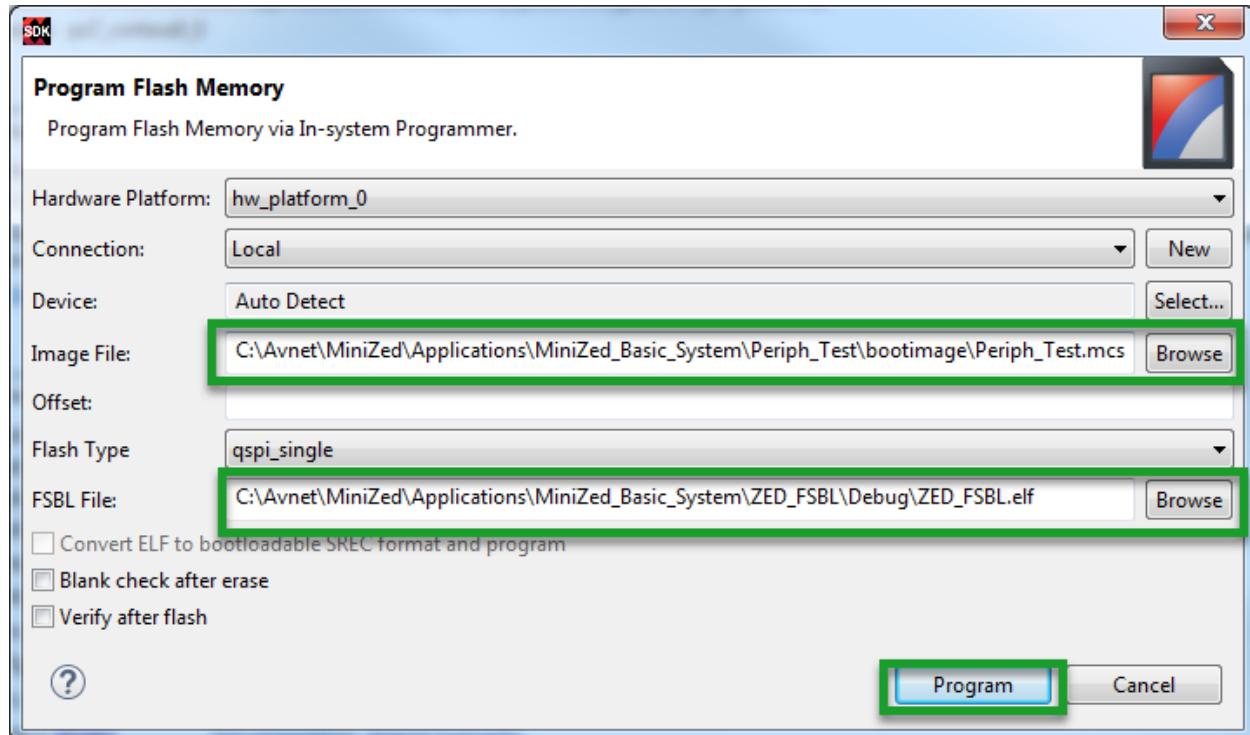



Figure 12 – Program QSPI Flash Memory

6. The operation took approximately 40 seconds on a Win7 PC. 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).

```
Zynq> 100%  
sf write FFFC0000 220000 A308  
  
device 0 offset 0x220000, size 0xa308  
  
SF: 41736 bytes @ 0x220000 Written: OK  
  
Zynq> Program Operation successful.  
INFO: [Xicom 50-44] Elapsed time = 13 sec.  
  
Flash Operation Successful
```

Figure 13 – QSPI Programmed Successfully

7. Power off the board by unplugging J2 micro-USB connector.
8. Set the MiniZed boot mode switch SW1 to QSPI mode ('F' for Flash) as shown below.

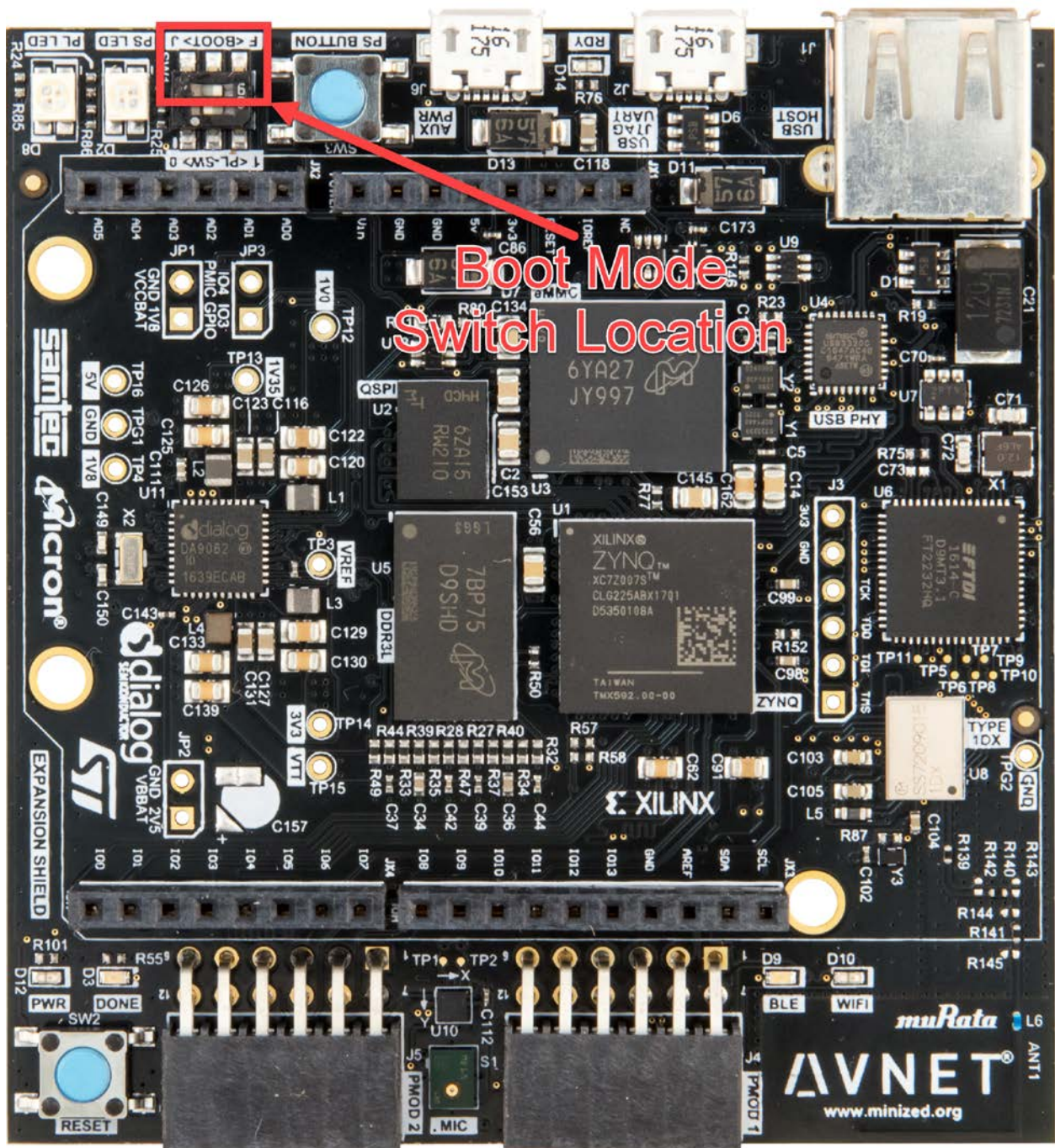


Figure 14 – MiniZed Switch Location

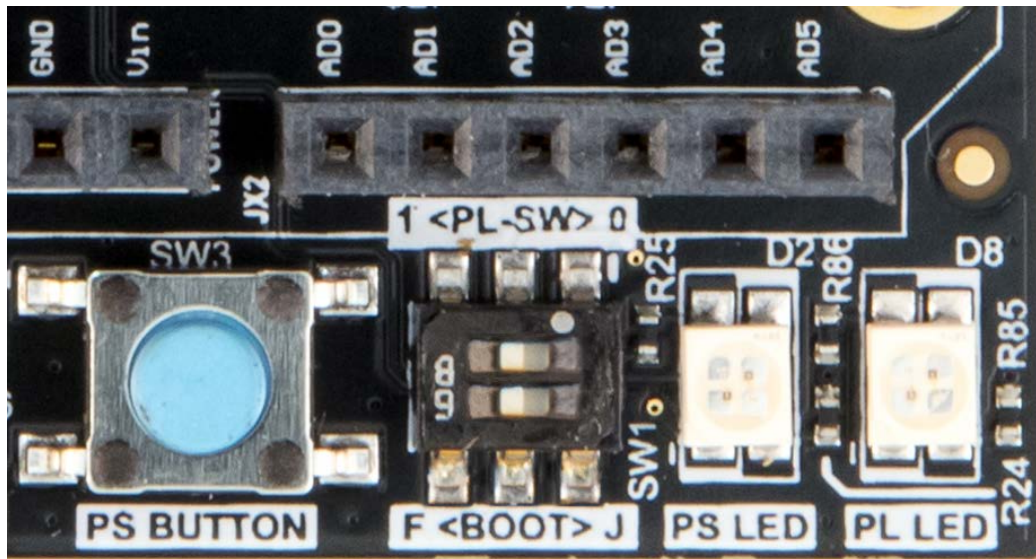
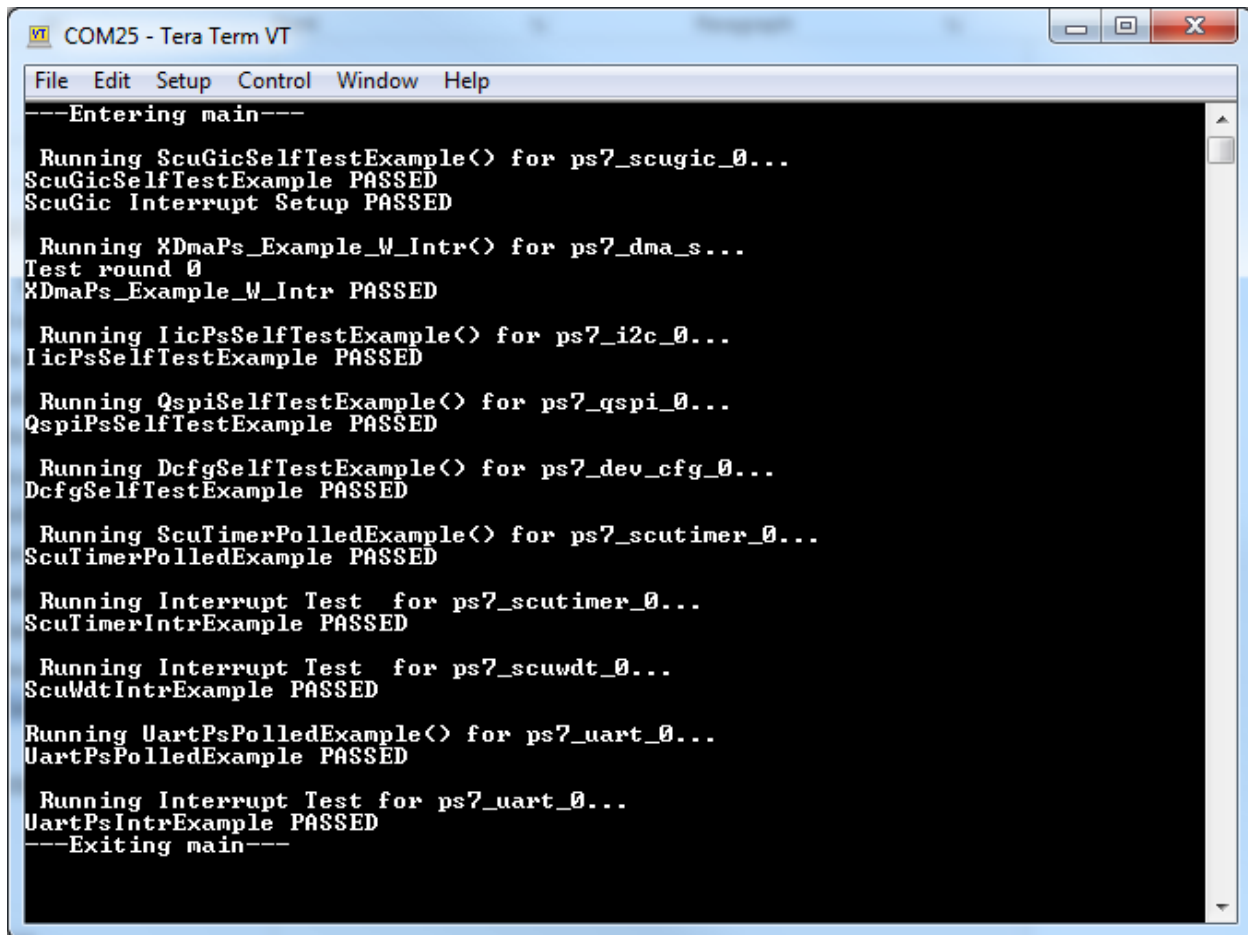


Figure 15 – QSPI/Flash Boot Mode

9. Close or disconnect the terminal that may have previously been open on your PC.
10. Power on the board by reconnecting the USB-JTAG-UART J2 connection. A blue LED should illuminate meaning the bitstream was loaded.
11. Launch a terminal program (TeraTerm) with the 115200/8/n/1/n settings.
12. Push the RESET button (SW2). You should see the results in the terminal.



```
COM25 - 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 0
XDmaPs_Example_W_Intr PASSED

Running IicPsSelfTestExample() for ps7_i2c_0...
IicPsSelfTestExample 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 UartPsPolledExample() for ps7_uart_0...
UartPsPolledExample PASSED

Running Interrupt Test for ps7_uart_0...
UartPsIntrExample PASSED
---Exiting main---
```

Figure 16 – Results from QSPI boot of Periph_Test

13. Close the terminal. Power off the board.

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 |
| 07 Jun 2017 | 2017_1.01 | Updated to 2017.1 for MiniZed |
| 10 Apr 2018 | 2018_1.01 | Updated to 2018.1 for MiniZed |

