Mercury+ XU1 SoC Module

Reference Design for Mercury+ PE1 Base Board User Manual

Purpose

The purpose of this document is to present to the user the overall view of the Mercury+ XU1 SoC module reference design and to provide the user with a step-by-step guide to the complete Xilinx® MPSoC design flow used for the Mercury+ XU1 SoC module.

Summary

This document first gives an overview of the Mercury+ XU1 SoC module reference design and then guides through the complete Xilinx MPSoC design flow for the Mercury+ XU1 SoC module in the getting started section. In addition, the internals and the boot options of the Mercury+ XU1 SoC module reference design are described.

Product Information	Number	Name
Product	ME-XU1	Mercury+ XU1 SoC Module

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-428-002	04	24.08.2018

Approval Information	Name	Position	Date
Written by	DIUN	Design Engineer	28.04.2017
Verified by	GLAC	Design Engineer	28.04.2017
Approved by	RPAU	Quality Manager	24.08.2018

Document History

Version	Date	Author	Comment
04	24.08.2018	DIUN	Updated signal names for revision 3 modules
03	15.08.2018	DIUN	Updated to Vivado and SDK 2018.2
02	18.01.2018	DIUN	Updated to Vivado and SDK 2017.4, updated to revision 2 modules, other style updates and clarifications
01	12.05.2017	DIUN	Version 01

Table of Contents

1 Overview **4**
1.1 Introduction 4
1.2 Directory Structure 4
1.3 Prerequisites 5

2 Reference Design Description **6**
2.1 Block Diagram 6
2.2 Processing System (PS) 6
2.2.1 Clocks 6
2.2.2 DDR4 SDRAM 6
2.2.3 SD Card 7
2.2.4 eMMC 7
2.2.5 I2C 7
2.2.6 Quad SPI Flash Controller 7
2.2.7 UART 8
2.2.8 Ethernet 8
2.2.9 USB 8
2.2.10 GPIOs 8
2.3 Programmable Logic (PL) 9
2.3.1 GPIOs 9
2.3.2 System Management 9

3 Getting Started **10**
3.1 Essential Information 10
3.2 Hardware Setup 11
3.3 FPGA Bitstream Generation 13
3.4 SDK Workspace Preparation 14
3.5 Running Software Applications 17
3.6 Embedded Software 18
3.6.1 General 18
3.6.2 Hello World Application 19
3.6.3 I2C Example Application 19
3.6.4 Memory Test Application 20
3.6.5 Flash Test Application 21
3.6.6 First Stage Boot Loader (FSBL) Application 22

4 Boot Configurations **23**
4.1 QSPI Flash Boot 23
4.1.1 Generating the Image Files 23
4.1.2 Preparing the Hardware 24
4.1.3 Programming the QSPI Flash 25
4.1.4 Booting from the QSPI Flash 28
4.2 SD Card Boot 28
4.2.1 Generating the Image Files 28
4.2.2 Preparing the Hardware 28
4.2.3 Programming the SD Card 28
4.2.4 Booting from the SD Card 29

5 Troubleshooting **30**
5.1 Vivado Issues 30
5.2 SDK Runtime Exceptions 30
5.3 JTAG Connection Issues 30
5.4 UART Connection Issues 30

1 Overview

1.1 Introduction

The Mercury+ XU1 SoC module reference design demonstrates a system using the Mercury+ XU1 SoC module in combination with the Mercury+ PE1 base board (any board variant) or with the regular Mercury PE1 base board (previous revisions). It presents the basic configuration of the device and features some example applications.

A troubleshooting section is included at the end of the document, to help the user solve potential issues related to board connectivity and/or system functionality.

Please note that the features presented in the reference design depend on the employed base board, therefore some features may not be available in certain hardware configurations.

An introduction to the Xilinx tools is provided by the documents below:

- Vivado Design Suite User Guide, Embedded Processor Hardware Design [1]
- Zynq UltraScale+ MPSoC: Embedded Design Tutorial, A Hands-On Guide to Effective Embedded System Design [2]

More information on the Mercury+ XU1 SoC module and the Mercury+ PE1 base board can be retrieved from their respective user manuals [3] [4].

1.2 Directory Structure

The Mercury+ XU1 SoC module reference design is delivered as a ZIP archive file with the following directory structure and contents:

- `binaries` — Pre-compiled binaries directory
- `scripts` — Scripts directory required for Vivado project creation
- `SdkExport` — Pre-generated hardware description files required for SDK applications
- `software` — Software projects directory
- `src` — Xilinx pinout and timing constraints and VHDL source code directory
- `Mercury_XU1_Reference_Design_for_Mercury_PE1_User_Manual.pdf` — User manual (this document)

1.3 Prerequisites

- IT
  - A computer with a microSD card slot (optional[1]) running Windows 7 64-bit (or later)
- Software
  - Xilinx Vivado 2018.2 WebPack, Evaluation, Design or System Edition (check the Mercury+ XU1 SoC Module User Manual [3] for details on device support in Xilinx tools)
  - Xilinx Software Development Kit (SDK) 2018.2
  - Enclustra Module Configuration Tool (MCT) [5] (optional[2])
  - A terminal emulation program (e.g. Tera Term)
- Hardware
  - An Enclustra Mercury+ XU1 SoC module
  - An Enclustra Mercury+ PE1 base board (any board variant) or a regular Mercury PE1 base board (previous revisions)
- Accessories
  - A standard micro USB cable
  - A Xilinx JTAG programmer (e.g. Platform Cable USB) download cable (optional[3])
  - A microSD card (optional[1])

[1]Only required for SD card boot mode
[2]May be used for flash programming, for MPSoC device configuration or for FTDI configuration.
[3]The FTDI device present on the base board can be configured to Xilinx JTAG mode using the Enclustra MCT software [5].

2 Reference Design Description

2.1 Block Diagram

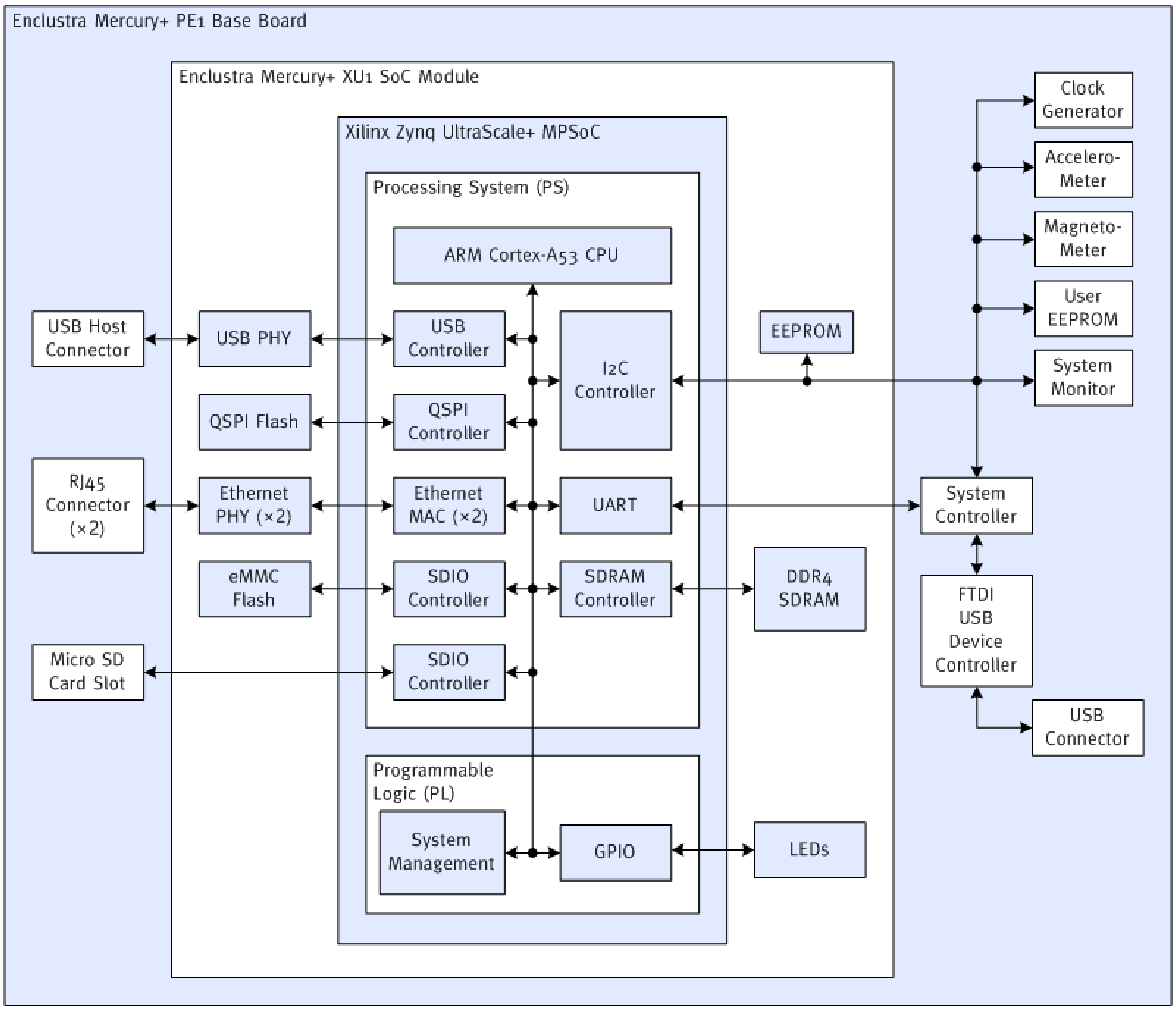


Figure 1: Hardware Block Diagram

2.2 Processing System (PS)

2.2.1 Clocks

The PS input clock frequency is configured to 33.33 MHz, while the CPU clock frequency is configured to 1.2 GHz. The CPU clock frequency multiplier can be modified in the processing system. A 50 MHz clock and a 100 MHz clock are exported from PS to the PL.

2.2.2 DDR4 SDRAM

The DDR4 SDRAM memory runs at 1200 MHz (2400 Mbit/sec) - this parameter can be modified in the processing system. ECC is enabled for the external memory. The DDR settings in the PS must be configured according to the Mercury+ XU1 SoC Module User Manual [3].

2.2.3 SD Card

The SD card is configured in the PS to the MIO 46..51 pins. This enables SD card access, as well as booting from the SD card.

To allow the Mercury+ XU1 SoC module to boot from the SD card, the hardware configuration on the Mercury+ PE1 base board must be done according to Section 4.2.2.

2.2.4 eMMC

The eMMC interface is configured in the PS to the MIO 13..22 pins. This enables eMMC flash access, as well as booting from the eMMC flash. To allow the Mercury+ XU1 SoC module to boot from the eMMC, the boot signals must be configured as described in the Mercury+ XU1 SoC Module User Manual [3].

2.2.5 I2C

The I2C controller I2C0 is configured to the MIO 10..11 pins. Table 1 lists the connected devices on the Mercury+ XU1 SoC module and Mercury+ PE1 base board.

Device	Address (7-bit)	Vendor	Part Type
Secure EEPROM	0x64	Atmel	ATSHA204A-MAHDA-T
System controller	0x0D	Lattice Semiconductor	LCMXO2-4000HC-6MG132I
System monitor	0x2F	Texas Instruments	LM96080CIMT/NOPB
Clock generator	0x70	Silicon Labs	SI5338B-B-GMR
User EEPROM	0x57	Microchip	24AA128T-I/MNY
Accelerometer	0x1D	ST Microelectronics	LSM303CTR
Magnetometer	0x1E	ST Microelectronics	LSM303CTR

Table 1: I2C Devices

Please note that, depending on the employed base board, some of the I2C devices may not be available in certain hardware configurations.

The device vendors or addresses of the I2C devices may change in future revisions of Mercury+ XU1 SoC module or Mercury+ PE1 base board.

For detailed information on the I2C devices, please refer to the corresponding user manuals [3] [4].

2.2.6 Quad SPI Flash Controller

The quad SPI flash controller is connected to MIO 0..6 pins in Single mode. The flash pins are shared between the trace interface and QSPI flash on the Mercury+ XU1 SoC module. Please refer to the Mercury+ XU1 SoC Module User Manual [3] for details about the MIO pin connections.

To allow the Mercury+ XU1 SoC module to boot from the QSPI flash, the hardware configuration on the Mercury+ PE1 base board must be done according to Section 4.1.2.

2.2.7 UART

The UART0 is mapped to MIO 38..39 pins and connected to the FTDI USB device controller on the Mercury+ PE1 base board. The UART is configured as shown in Table 2.

Parameter	Value
Baud rate	115'200
Data	8 bit
Parity	None
Stop	1 bit
Flow control	None

Table 2: UART Configuration

2.2.8 Ethernet

The Ethernet MACs GEM0 and GEM3 are mapped to MIO 26..37 pins and MIO 64..75 pins respectively in the PS and are connected to two Micrel KSZ9031 Ethernet PHYs on the Mercury+ XU1 SoC module using RGMII interfaces.

The Gigabit Ethernet PHYs share the MDIO interface, which is connected to MIO 76..77. The PHYs can be configured by using the corresponding addresses: PHY0 has address 3, PHY1 has address 7.

The second Ethernet interface is shared with the second USB interface, therefore only one of the two can be used at a given time. Please refer to the Mercury+ XU1 SoC Module User Manual [3] for details about the MIO pin connections.

2.2.9 USB

The USB controller USB0 on MIO 52..63 pins is connected to a USB3320C USB 2.0 PHY. This interface can be configured for USB host or device.

Depending on the required USB mode, the settings in the system controller and the DIP switches on the Mercury+ PE1 base board must be configured correctly. Please refer to the Mercury+ PE1 Base Board User Manual [4] for details.

The second USB interface is not used in the reference design, as the MIO pins are assigned to the Ethernet MAC. The second USB interface is shared with the second Ethernet interface, therefore only one of the two can be used at a given time. Please refer to the Mercury+ XU1 SoC Module User Manual [3] for details about the MIO pin connections.

2.2.10 GPIOs

The unused MIO pins from the PS are available as GPIOs. They are mapped to MIO pins 24..25 and 40..44 in the PS. The function of the general purpose pins on the Mercury+ XU1 SoC module is described in Table 3.

GPIO	Signal	Function
MIO 24	PS_LED0#	LED0#, see Mercury+ XU1 SoC Module User Manual [3] for details
MIO 25	PS_LED1#	LED1#, connected in parallel with FPGA pin AE8 on revision 1 modules, see Mercury+ XU1 SoC Module User Manual [3] for details
MIO 40..44	GPIO 40..44	GPIO, user function

Table 3: PS GPIO Configuration

On revision 1 modules, MIO pin 25 is connected in parallel to FPGA pin AE8. Make sure the FPGA pin is in high impedance state before driving the PS GPIO pin and vice versa.

2.3 Programmable Logic (PL)

2.3.1 GPIOs

A Xilinx GPIO controller in the PL is connected to the PS via an AXI bus. The PL GPIOs can be connected to the FPGA LED in the top level.

The FPGA firmware contains a 24-bit counter freely running at 50 MHz. The MSB of this counter is used to blink LED2# on FPGA pin AE8 with a frequency of approximately 3 Hz.

FPGA Pin	Signal	Function
AE8	LED2#	Blinking LED counter MSB

Table 4: FPGA Firmware I/O Configuration

On revision 1 modules, LED1# is mapped to AE8 package pin (instead on LED2#).

2.3.2 System Management

A System Management IP core instance is connected to the PS via an AXI bus, in order to monitor the temperature of the device. The temperature threshold for this module is configured to the industrial applications temperature, 85° Celsius.

The constraints provided in the reference design enable FPGA bitstream power-down, when the temperature increases above the threshold. In this case, the PL will be reset, while the ARM processor will still be running.

Depending on the user application, the Mercury+ XU1 SoC module may consume more power than can be dissipated without additional cooling measures; always make sure the MPSoC is adequately cooled by installing a heat sink and/or providing air flow. Temperature control and monitoring is very important in a complex design.

Information that may assist in selecting a suitable heat sink for the Mercury+ XU1 SoC module can be found in the Enclustra Modules Heat Sink Application Note [6].

3 Getting Started

This section describes the steps required to configure the Mercury+ XU1 SoC module and Mercury+ PE1 base board in order to run the example applications. The section includes information on how to:

- Mount the module and configure the base board
- Generate the FPGA bitstream
- Prepare the software workspace
- Run the software applications

The example applications, including the expected results of running the software, are described in detail in Section 3.6.

3.1 Essential Information

Warning!

Never mount or remove the Mercury+ XU1 SoC module to or from the Mercury+ PE1 base board while the Mercury+ PE1 base board is powered. Always remove or turn off the power supply before mounting or removing the Mercury+ XU1 SoC module.

Warning!

It is possible to mount the Mercury+ XU1 SoC module the wrong way round on the Mercury+ PE1 base board - always check that the mounting holes on the Mercury+ PE1 base board are aligned with the mounting holes of the Mercury+ XU1 SoC module.

The base board and module may be damaged if the module is mounted the wrong way round and powered up.

Warning!

Depending on the user application, the Mercury+ XU1 SoC module may consume more power than can be dissipated without additional cooling measures; always make sure the MPSoC is adequately cooled by installing a heat sink and/or providing air flow.

Warning!

Please read carefully the Mercury+ XU1 SoC module and Mercury+ PE1 base board user manuals before proceeding.

Note that when Enclustra MCT [5] is used for MPSoC configuration or flash programming, all other tools that may be connected to the FTDI device (e.g. Vivado Hardware Manager, SDK, UART terminal) must be closed.

3.2 Hardware Setup

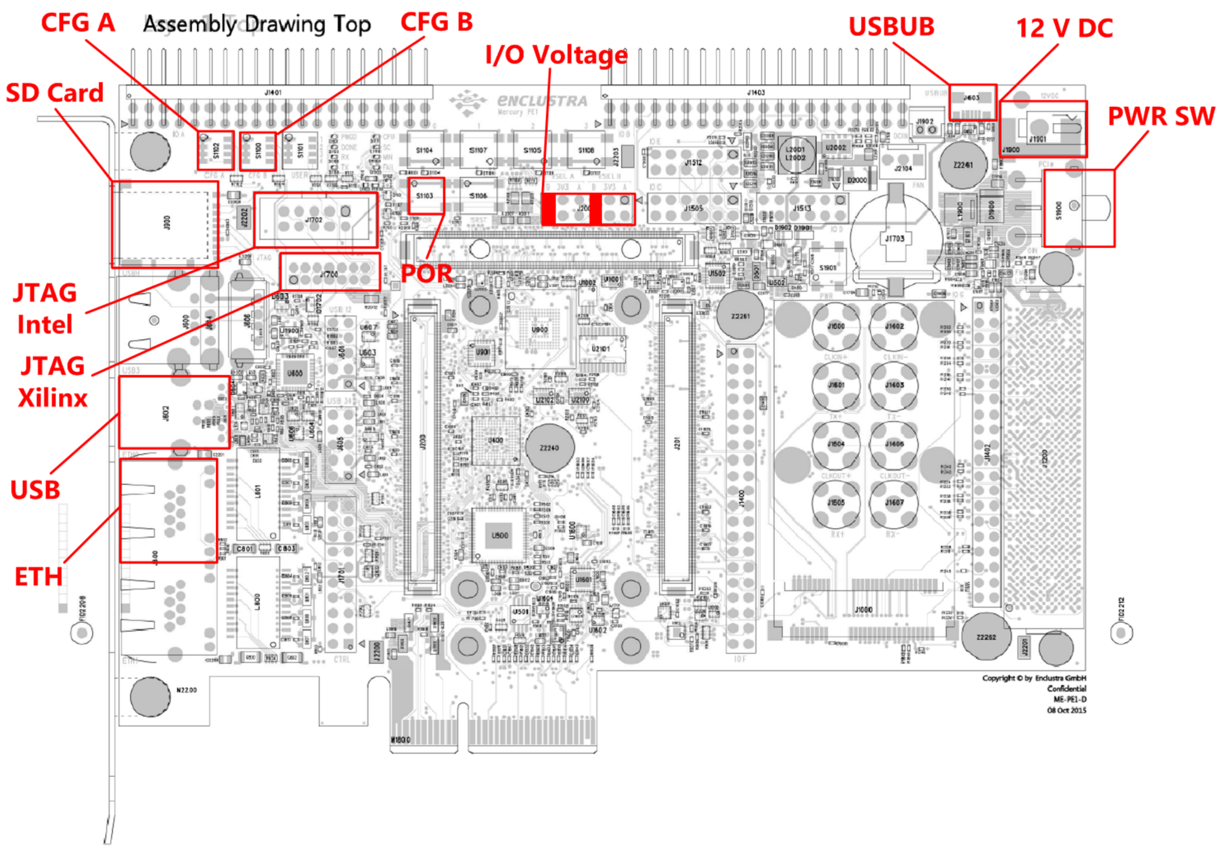


Figure 2: Mercury+ PE1 Base Board Assembly Drawing (Top View)

Step	Description
1	Set the I/O voltage jumpers on the Mercury+ PE1 base board according to label **I/O Voltage** in Figure 2 (the jumpers are marked with red rectangles): • VSEL A = 1.8 V (position B) • VSEL B = 1.8 V (position B)
2	Set the configuration DIP switches on the Mercury+ PE1 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON] • CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]
3	Mount the Mercury+ XU1 SoC module to the Mercury+ PE1 base board. Make sure that the mounting holes of the Mercury+ XU1 SoC module are aligned with the mounting holes of the Mercury+ PE1 base board before proceeding.
4	Connect the micro USB cable between your computer and the Mercury+ PE1 base board. Use the micro USB port labeled **USBUB** in Figure 2.

Continued on next page...

Step	Description
5	Connect the 12 V DC power supply plug to the power connector of the Mercury+ PE1 base board (see label **12 V DC** in Figure 2).
6	Set the power switch of the Mercury+ PE1 base board to ON (see label **PWR SW** in Figure 2).
7	Connect the Xilinx Platform Cable USB to the JTAG connector of the Mercury+ PE1 base board (see label **JTAG Xilinx** in Figure 2). Alternatively, JTAG over USB may be used. For this, the FTDI device on the Mercury+ PE1 base board must be configured to Xilinx JTAG mode using Enclustra MCT [5]. Details on the Xilinx JTAG mode configuration are presented in the Mercury+ PE1 Base Board User Manual [4]
8	Open a terminal program on your computer (e.g. Tera Term) and open a serial port connection using the COM port labeled with the higher number from the two newly detected ports. For issues related to COM ports detection, refer to Section 5.4. Configure the UART parameters according to Section 2.2.7.
Note	On the Mercury+ XU1 SoC modules equipped with ES1 (engineering samples 1) MPSoC devices, FPGA configuration and program downloading to the ARM core via JTAG is possible only when PJTAG boot mode is used. On these modules, the examples can be tested without JTAG using the other boot modes available.

Table 5: Hardware Setup Step-By-Step Guide

Warning!

Please make sure that a single JTAG adapter is connected to the base board and enabled at a given moment, otherwise the development tools may report errors during JTAG connecting attempts.

3.3 FPGA Bitstream Generation

For a fast test of the provided software applications, the pre-generated bitstream included in the `binaries` directory may alternatively be used, therefore the steps described in this section may be skipped.

The `<base_dir>\binaries` directory includes bitstream files for any MPSoC device that may be equipped on the module.

Step	Description
1	Edit the `fpga_part` variable in `scripts\settings.tcl` file, according to your MPSoC device. This file includes module name and board information required for the project creation script. All settings, except for `fpga_part` should be left on default. The list of options for `fpga_part` is given in the comments within the Tcl file. Save the file after editing.
2	Start Xilinx Vivado 2018.2 and create the Mercury+ XU1 SoC module reference design project: 1. Click on the Tcl console at the bottom of the page and type: (a) `cd <base_dir>` (`<base_dir>` is the directory in which you extracted the archive contents). Note that you must use / for hierarchy separator, instead of \\. (b) `source scripts/create_project.tcl` 2. Wait for completion
3	Run Synthesis, Implementation & Bitstream Generation in Vivado 2018.2: 1. Click on Generate Bitstream from the Flow Navigator bar 2. In the Launch Runs window click OK - this will start automatically the entire implementation process 3. Wait for completion → select View Reports → OK
4	Export the hardware system information (required for the SDK projects): 1. File → Export → Export Hardware 2. Enable Include bitstream checkbox If you want to save the .hdf file to another path than the default location: 3. Click on Choose Location → Select 4. Hit OK

Table 6: FPGA Bitstream Generation Step-By-Step Guide

3.4 SDK Workspace Preparation

This section describes how to create and import the software applications. The steps are generic, and apply to all software examples provided in the release archive, along with this document.

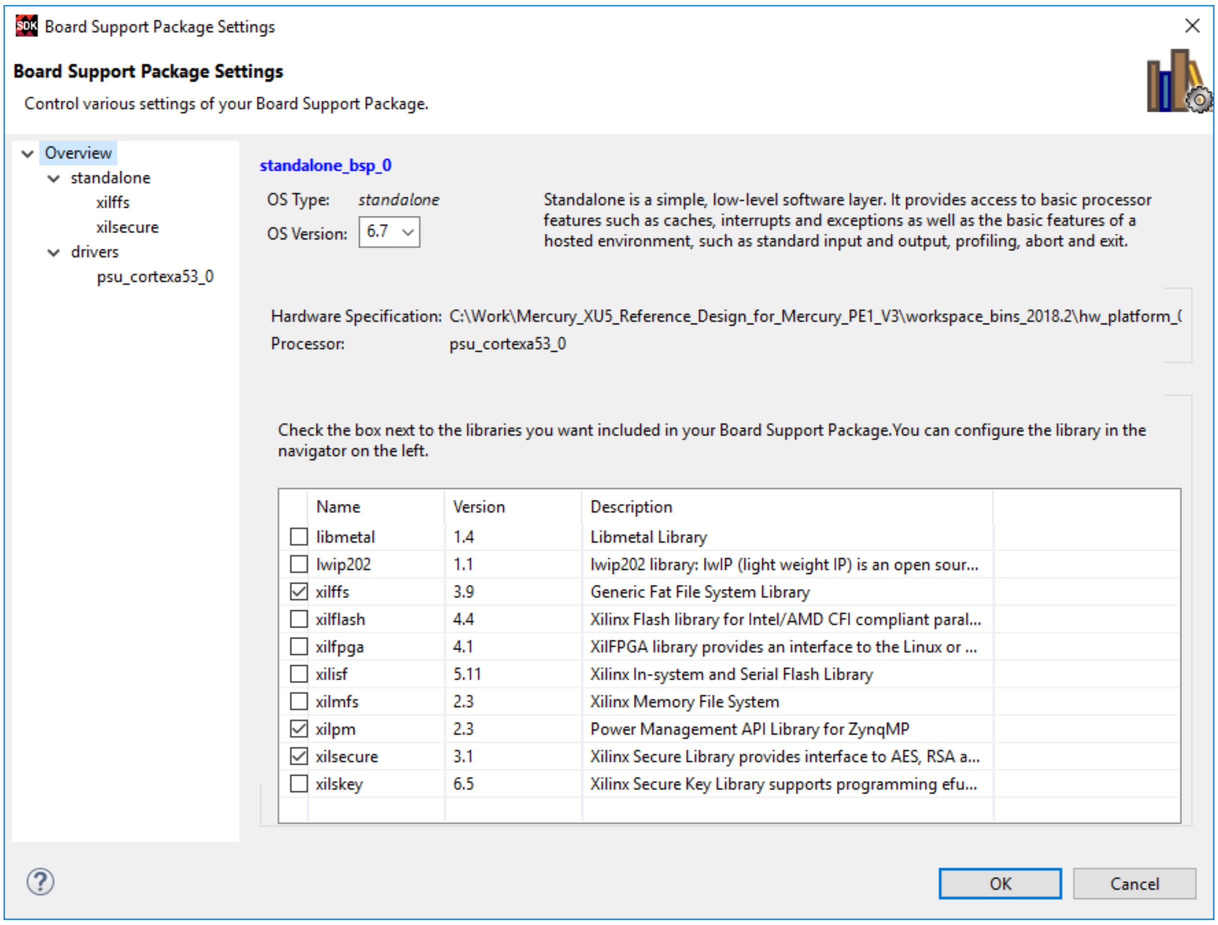
The `<base_dir>\SdkExport` directory includes pre-generated hardware description files for any MPSoC device that may be equipped on the module.

Step	Description
1	Start Xilinx SDK 2018.2 1. Select any workspace (e.g. `<base_dir>\workspace`)
2	Create a new board support package (BSP) 1. File → New → Board Support Package → Specify 2. In the New Hardware Project window: (a) For Project Name type hw_platform_0 (b) For Target Hardware Specification select the .hdf file you exported from Vivado, as described in Section 3.3. The default location used by Vivado is `<base_dir>\<vivado_proj_dir>\<project_name>.sdk\system_top.hdf` Alternatively, the pre-compiled hardware description file from `<base_dir>\SdkExport` may be used. (c) Hit Finish 3. In the New Board Support Package Project window: (a) For Project Name type standalone_bsp_0 (b) For Hardware Platform select hw_platform_0 (c) For CPU select psu_cortexa53_0 (d) For Compiler select 64-bit (e) For Board Support Package OS select standalone (f) Hit Finish 4. In the Board Support Package Settings window (see Figure 3): (a) Enable the checkboxes corresponding to the drivers and libraries required in the BSP: xilffs, xilpm, xilsecure (b) Hit OK
3	Create new First Stage Boot Loader application 1. File → New → Application Project 2. In the Application Project window: (a) For Project Name type FSBL (b) For Board Support Package select Use existing, and select standalone_bsp_0 (c) Leave the other settings on default values (d) Hit Next 3. In the Templates window: (a) Select Zynq MP FSBL (b) Hit Finish

Continued on next page...

Step	Description
4	Import the example projects into the workspace (see Figure 4): 1. File → Import... 2. Select General → Existing Projects into Workspace and hit Next 3. For Select root directory choose `<base_dir>\software` and hit OK 4. Enable checkbox Copy projects into workspace 5. Hit Finish
5	Build all projects 1. Hit Ctrl-B and wait for completion

Table 7: SDK Workspace Preparation Step-By-Step Guide


Board Support Package Settings

Board Support Package Settings

Control various settings of your Board Support Package.

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

standalone_bsp_0

OS Type: *standalone*

OS Version: 6.7

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.

Hardware Specification: C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace_bins_2018.2\hw_platform_(

Processor: psu_cortexa53_0

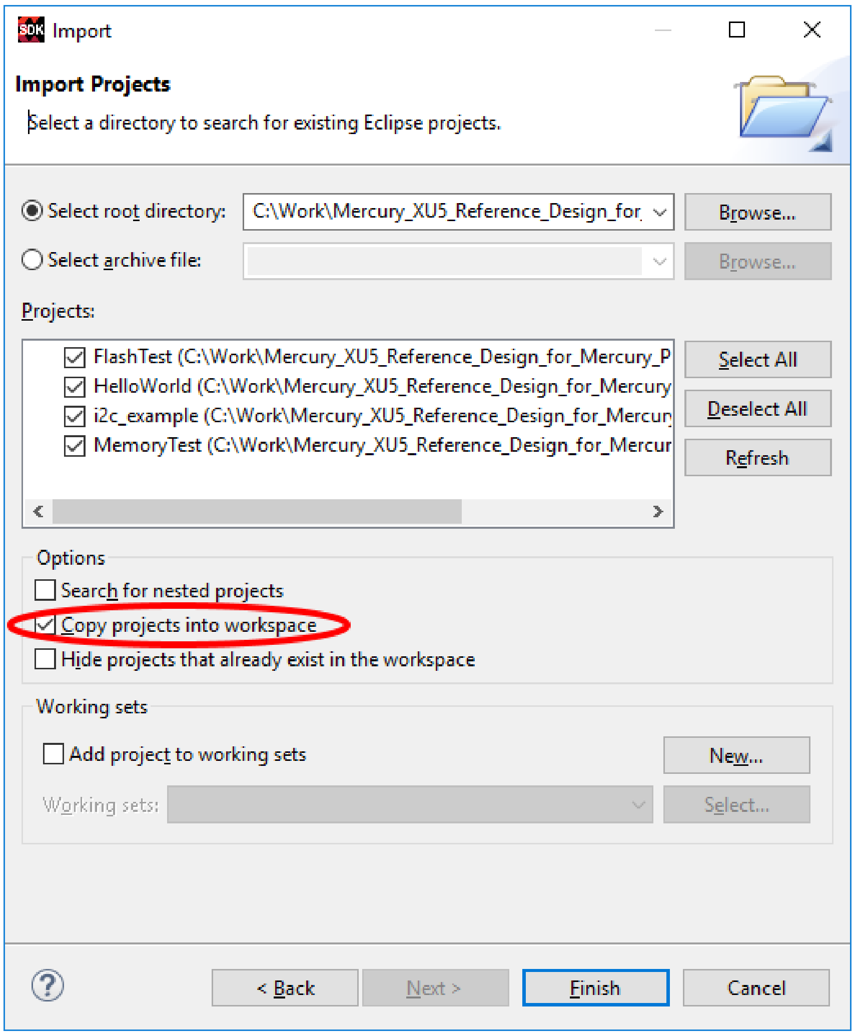
Check the box next to the libraries you want included in your Board Support Package.You can configure the library in the navigator on the left.

	Name	Version	Description
☐	libmetal	1.4	Libmetal Library
☐	lwip202	1.1	lwip202 library: lwIP (light weight IP) is an open sour...
☑	xilffs	3.9	Generic Fat File System Library
☐	xilflash	4.4	Xilinx Flash library for Intel/AMD CFI compliant paral...
☐	xilfpga	4.1	XilFPGA library provides an interface to the Linux or ...
☐	xilisf	5.11	Xilinx In-system and Serial Flash Library
☐	xilmfs	2.3	Xilinx Memory File System
☑	xilpm	2.3	Power Management API Library for ZynqMP
☑	xilsecure	3.1	Xilinx Secure Library provides interface to AES, RSA a...
☐	xilskey	6.5	Xilinx Secure Key Library supports programming efu...

Figure 3: Board Support PackageSettings

Warning!

For a successful build of the software examples the hardware project must be named "hw_platform_0" and that the BSP must be named "standalone_bsp_0".


Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\Work\Mercury_XU5_Reference_Design_for

( ) Select archive file:

Projects:

- [✓] FlashTest (C:\Work\Mercury_XU5_Reference_Design_for_Mercury_P
- [✓] HelloWorld (C:\Work\Mercury_XU5_Reference_Design_for_Mercury
- [✓] i2c_example (C:\Work\Mercury_XU5_Reference_Design_for_Mercury
- [✓] MemoryTest (C:\Work\Mercury_XU5_Reference_Design_for_Mercur

Options

- [ ] Search for nested projects
- [✓] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

Figure 4: Importing Software Projects

Warning!

Please note that the software applications may not work properly if they are imported in another software version than the one specified in this document. A solution for such cases is creating a new software project with the sources provided in the reference design.

Warning!

Please make sure that during import process the checkbox for copying the projects into workspace has been enabled, otherwise the build step will fail due to incorrect file paths.

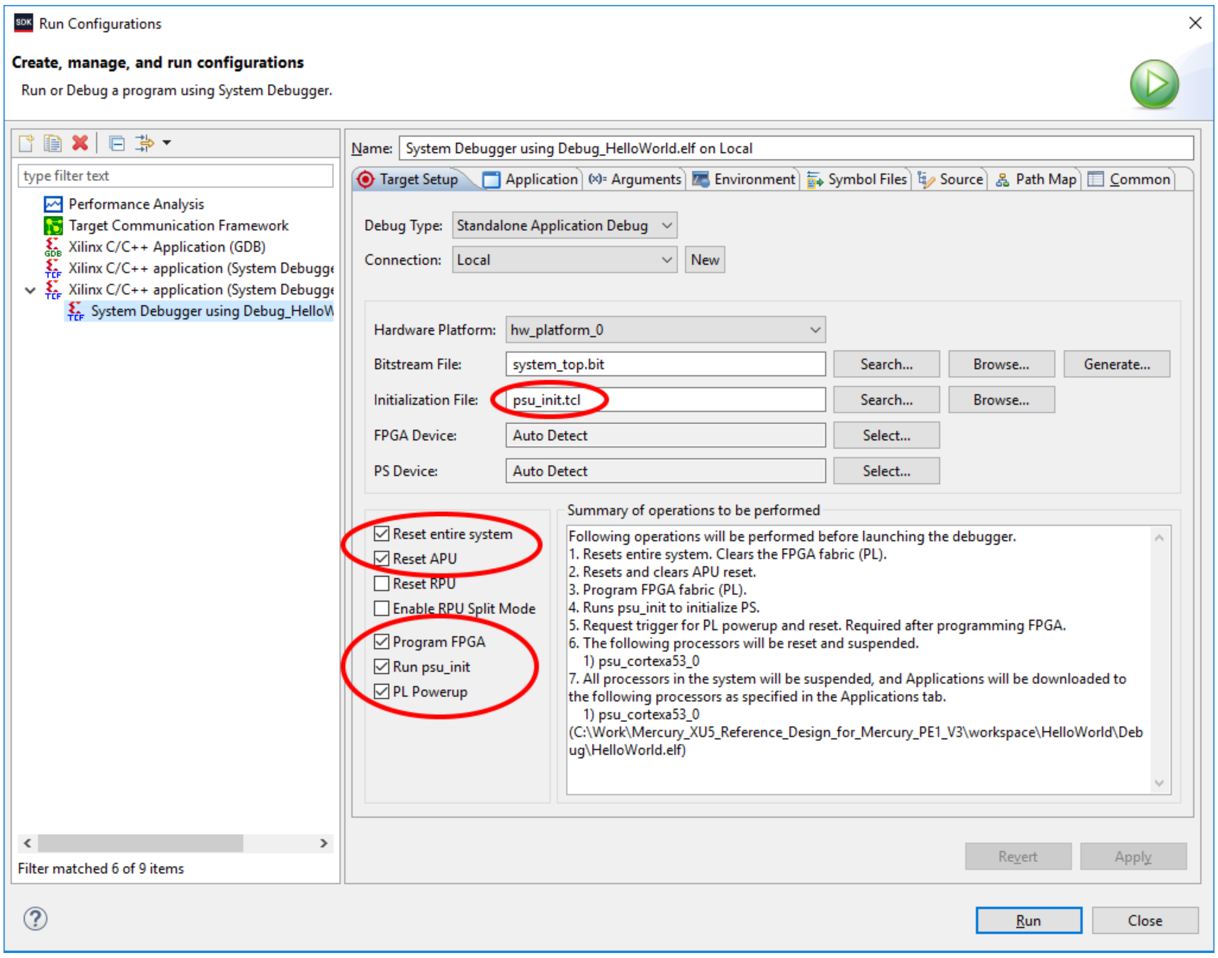
3.5 Running Software Applications

This section describes how to run software applications on the Mercury+ XU1 SoC module. The steps are generic, and apply to all software examples provided in the release archive, along with this document.

In order to execute the applications, the hardware needs to be configured as described in Section 3.2.

Step	Description
1	Create a run configuration for the application in SDK 2018.2: 1. Run → Run Configurations... 2. Right-click Xilinx C/C++ application (System Debugger) and hit New or double-click on Xilinx C/C++ application (System Debugger) 3. Enter a run configuration name in the Name field (e.g. HelloWorld) 4. Target Setup tab (see Figure 5): (a) For Hardware Platform select hw_platform_0 (b) For Bitstream file field, hit Search... (c) Select system_top.bit and hit OK (d) In the Initialization file field, hit Search... (e) Select psu_init.tcl and hit OK (f) For FPGA Device and PS Device, use Auto Detect option (g) Enable checkboxes Reset entire system, Reset APU, Program FPGA, Run psu_init and PL Powerup 5. Application tab: (a) Enable psu_cortexa53_0 checkbox (b) In the Project Name field click browse and select an application (e.g. HelloWorld) (c) In the Application field click search and select an .elf file (e.g. HelloWorld.elf) (d) Enable Reset processor checkbox (e) Hit Apply
2	Start the application by clicking the Run button. This method of starting the application resets the entire system, executes the required initialization for the PS, powers up the PL, configures the FPGA with the specified bitstream and downloads the application program to the ARM processor. In some test setup cases it was observed that the SDK tool was not able to start a second run session without a hardware reset. If required, power off and on the base board and restart the run configuration. For issues related to JTAG, refer to Section 5.3.

Table 8: Running an Application Step-By-Step Guide


Run Configurations

Create, manage, and run configurations

Run or Debug a program using System Debugger.

type filter text

- Performance Analysis
- Target Communication Framework
- Xilinx C/C++ Application (GDB)
- Xilinx C/C++ application (System Debugge
- Xilinx C/C++ application (System Debugge
  - System Debugger using Debug_HelloW

Filter matched 6 of 9 items

Name: System Debugger using Debug_HelloWorld.elf on Local

Target Setup | Application | Arguments | Environment | Symbol Files | Source | Path Map | Common

Debug Type: Standalone Application Debug

Connection: Local   New

Hardware Platform:	hw_platform_0			
Bitstream File:	system_top.bit	Search...	Browse...	Generate...
Initialization File:	psu_init.tcl	Search...	Browse...	
FPGA Device:	Auto Detect	Select...		
PS Device:	Auto Detect	Select...		

- ☑ Reset entire system
- ☑ Reset APU
- ☐ Reset RPU
- ☐ Enable RPU Split Mode
- ☑ Program FPGA
- ☑ Run psu_init
- ☑ PL Powerup

Summary of operations to be performed

```
Following operations will be performed before launching the debugger.
1. Resets entire system. Clears the FPGA fabric (PL).
2. Resets and clears APU reset.
3. Program FPGA fabric (PL).
4. Runs psu_init to initialize PS.
5. Request trigger for PL powerup and reset. Required after programming FPGA.
6. The following processors will be reset and suspended.
   1) psu_cortexa53_0
7. All processors in the system will be suspended, and Applications will be downloaded to
the following processors as specified in the Applications tab.
   1) psu_cortexa53_0
(C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\HelloWorld\Deb
ug\HelloWorld.elf)
```

Revert   Apply

Run   Close

Figure 5: Run Configurations Settings

After the FPGA is successfully configured, the **DONE** LED should be lit.

3.6 Embedded Software

This section describes the software examples and the expected UART output while running these applications.

3.6.1 General

The Mercury+ XU1 SoC module reference design comes with a number of example applications, which show how to initialize the peripheral controllers and how to access the external devices. All of them are bare-metal applications that are executed in the DDR SDRAM memory.

The applications included are a Hello World example, an I2C test that reads and prints out the module and board configuration, a flash test which checks the QSPI flash memory available on the Mercury+ XU1 SoC module, and a memory test that checks the DDR SDRAM memory available on the module.

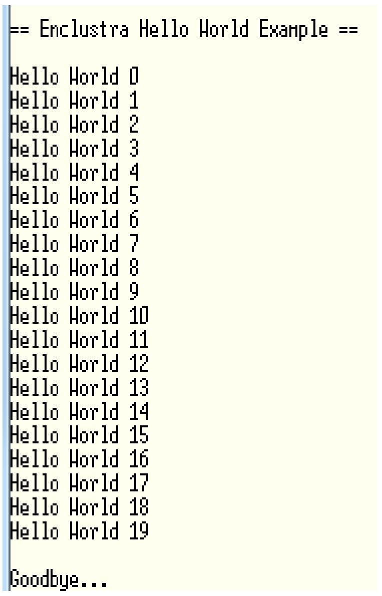
An additional section is included to describe how to create the First Stage Boot Loader (FSBL) application. This application is required during the process of creating a boot image for the module.

The procedure of importing, creating and building the example applications is explained in Sections 3.4 and 3.5.

3.6.2 Hello World Application

The Hello World application is a very simple application, which is used to demonstrate all the required steps for getting a bare-metal application running on the Mercury+ XU1 SoC module.

The Hello World application prints „Hello World x" for twenty times, while x is incremented by one at every iteration. Figure 6 shows the UART output of the Hello World application.



```
== Enclustra Hello World Example ==

Hello World 0
Hello World 1
Hello World 2
Hello World 3
Hello World 4
Hello World 5
Hello World 6
Hello World 7
Hello World 8
Hello World 9
Hello World 10
Hello World 11
Hello World 12
Hello World 13
Hello World 14
Hello World 15
Hello World 16
Hello World 17
Hello World 18
Hello World 19

Goodbye...
```

Figure 6: Hello World Application UART Output

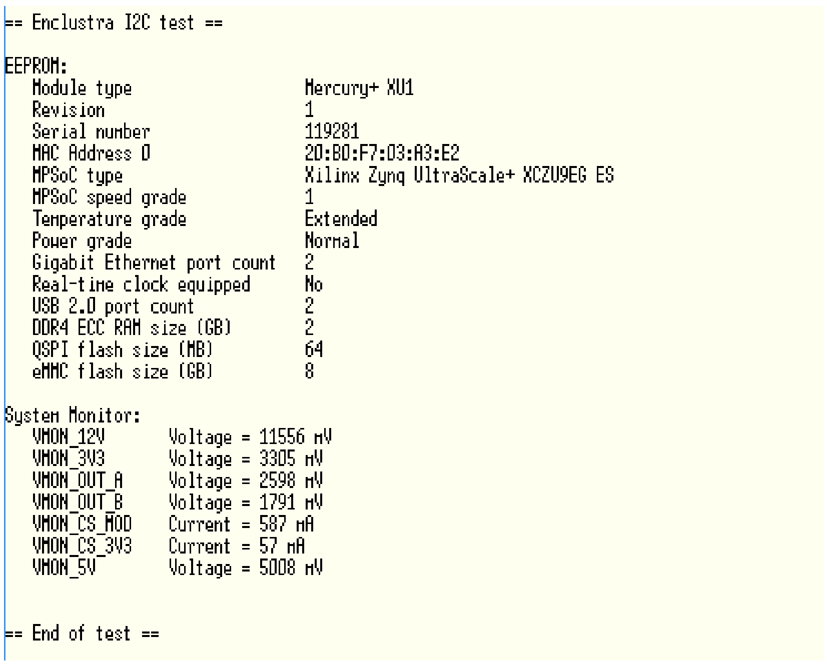
3.6.3 I2C Example Application

The I2C example application demonstrates the configuration and use of the I2C controller on the Mercury+ XU1 SoC module.

The I2C example application reads the module configuration data from the secure EEPROM on the Mercury+ XU1 SoC module, as well as voltage and current information from the system monitor present on the Mercury+ PE1 base board.

The module configuration includes: module type, serial number, DDR SDRAM and flash memory sizes, and first MAC address. The MAC addresses can be used by the user to configure the Ethernet MAC.

Figure 7 shows the UART output of the I2C application.



```
== Enclustra I2C test ==

EEPROM:
   Module type                  Mercury+ XU1
   Revision                     1
   Serial number                119281
   MAC Address 0                20:B0:F7:03:A3:E2
   MPSoC type                   Xilinx Zynq UltraScale+ XCZU9EG ES
   MPSoC speed grade            1
   Temperature grade            Extended
   Power grade                  Normal
   Gigabit Ethernet port count  2
   Real-time clock equipped     No
   USB 2.0 port count           2
   DDR4 ECC RAM size (GB)       2
   QSPI flash size (MB)         64
   eMMC flash size (GB)         8

System Monitor:
   VMON_12V      Voltage = 11556 mV
   VMON_3V3      Voltage = 3305 mV
   VMON_OUT_A    Voltage = 2598 mV
   VMON_OUT_B    Voltage = 1791 mV
   VMON_CS_MOD   Current = 587 mA
   VMON_CS_3V3   Current = 57 mA
   VMON_5V       Voltage = 5008 mV


== End of test ==
```

Figure 7: I2C Example Application UART Output

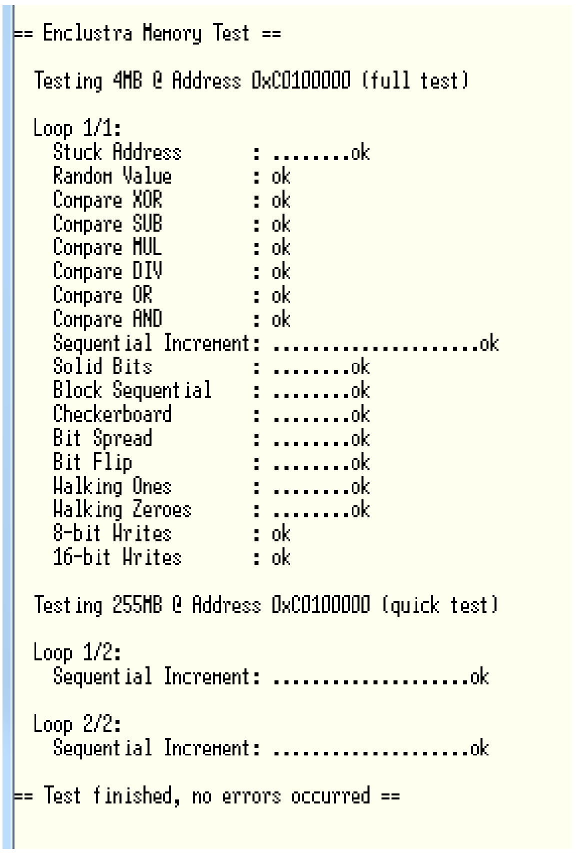
3.6.4 Memory Test Application

The memory test application performs several tests on the DDR memory present on the Mercury+ XU1 SoC module. A quick simple test and a detailed full test are executed.

The simple test is run on a bigger part of the memory and checks that the sequential incrementing of the address and writing values to the memory works as expected. The full test uses several writing patterns on two different parts of the memory, then reads the values and compares the results. This test is performed on a smaller part of the memory.

Figure 8 shows the UART output of the memory test application. Note that the starting address and the memory size configured for the test depend on the module and on the application settings.

Please note that depending on the DDR memory controller speed, memory cache enable and on the test size, the memory test may take several minutes to complete.



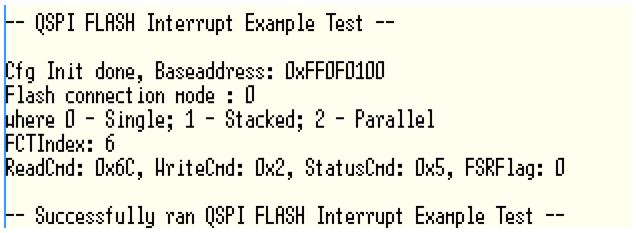
```
== Enclustra Memory Test ==

  Testing 4MB @ Address 0xC0100000 (full test)

  Loop 1/1:
    Stuck Address      : ........ok
    Random Value       : ok
    Compare XOR        : ok
    Compare SUB        : ok
    Compare MUL        : ok
    Compare DIV        : ok
    Compare OR         : ok
    Compare AND        : ok
    Sequential Increment: .....................ok
    Solid Bits         : ........ok
    Block Sequential   : ........ok
    Checkerboard       : ........ok
    Bit Spread         : ........ok
    Bit Flip           : ........ok
    Walking Ones       : ........ok
    Walking Zeroes     : ........ok
    8-bit Writes       : ok
    16-bit Writes      : ok

  Testing 255MB @ Address 0xC0100000 (quick test)

  Loop 1/2:
    Sequential Increment: ....................ok

  Loop 2/2:
    Sequential Increment: ....................ok

== Test finished, no errors occurred ==
```

Figure 8: Memory Test Application - UART Output Example (addresses and test sizes may vary)

3.6.5 Flash Test Application

The flash test application is based on the Xilinx SPI controller driver example and performs a write/read-back test of one sector starting at a specific address of the QSPI flash memory. The application initializes the SPI and the interrupt controller and includes read, write and erase functions that show how to access the flash memory on the Mercury+ XU1 SoC module using the Xilinx SPI controller.

Figure 9 shows the UART output of the flash test application.



```
-- QSPI FLASH Interrupt Example Test --

Cfg Init done, Baseaddress: 0xFF0F0100
Flash connection mode : 0
where 0 - Single; 1 - Stacked; 2 - Parallel
FCTIndex: 6
ReadCmd: 0x6C, WriteCmd: 0x2, StatusCmd: 0x5, FSRFlag: 0

-- Successfully ran QSPI FLASH Interrupt Example Test --
```

Figure 9: Flash Test Application UART Output

3.6.6 First Stage Boot Loader (FSBL) Application

The First Stage Boot Loader application is used in the boot image creation process described in Section 4.1.1. It is not used as an independent application. Section 3.4 describes the required steps for FSBL project generation.

4 Boot Configurations

Once a software application has been developed and tested, this can be used to build a boot image for the module.

The boot image contains the FSBL, the bitstream for programming the PL, and the software bare-metal application.

In order to use a software application for the boot image, the code must be mapped for execution from the external DDR memory. If the program is mapped to the on-chip memory, it will overwrite the boot loader during execution. The example applications in the reference design are mapped for execution from the DDR memory by default.

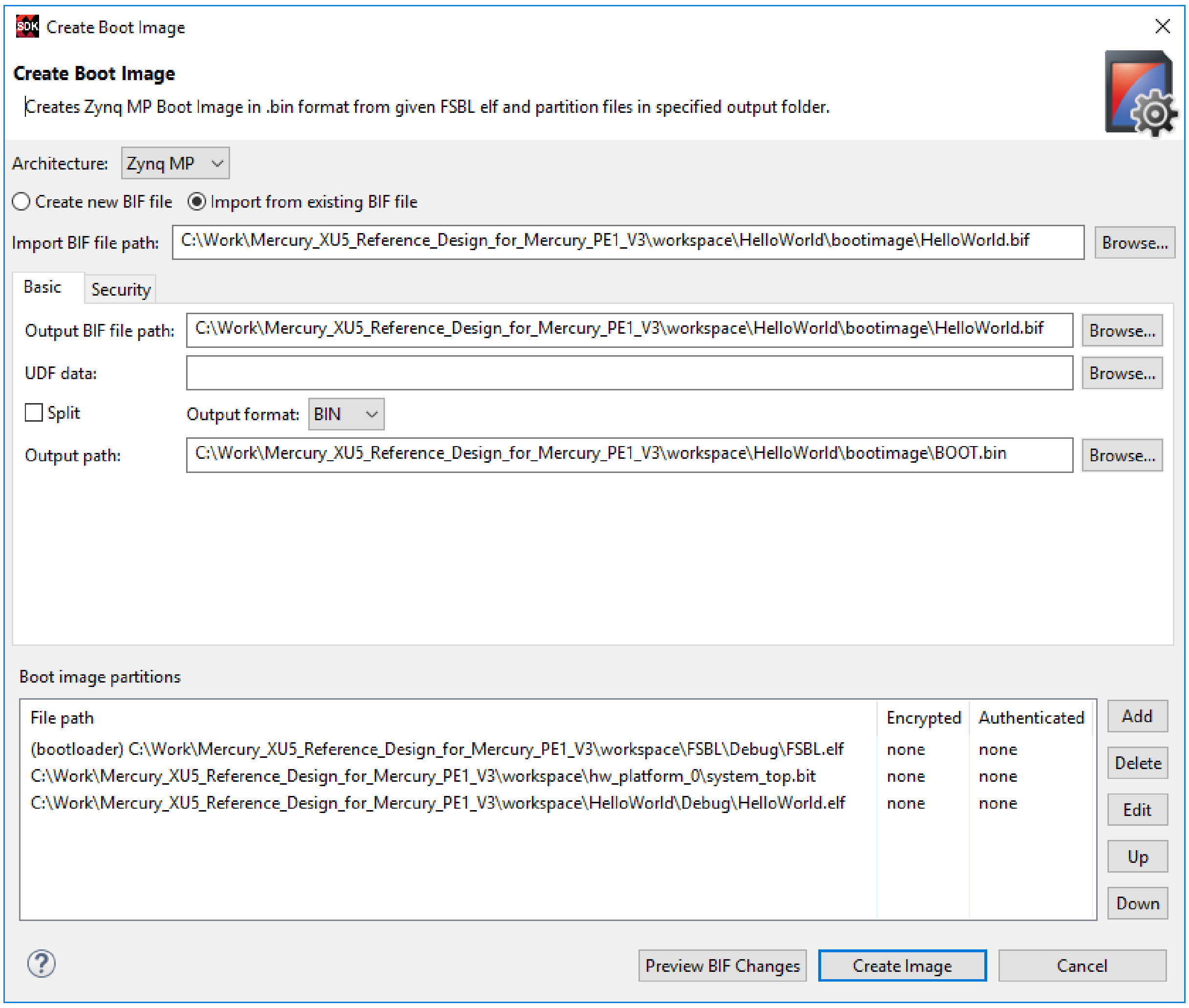
For a fast test of the boot configurations, the pre-generated .bin images included in the `<base_dir>\binaries` directory may be used for boot, instead of rebuilding the image. You need to select the file corresponding to the MPSoC device that is equipped on the module.

4.1 QSPI Flash Boot

4.1.1 Generating the Image Files

Step	Description
1	Create the boot image from Xilinx SDK 2018.2 (see Figure 10): 1. Right click on the application in the Project Explorer 2. Select Create Boot Image → Select Zynq MP for Architecture → Create Image An image will be created in `<workspace>\<app_name>\bootimage\BOOT.bin`.

Table 9: Generating the Image Files for QSPI Flash Boot Mode Step-by-Step Guide



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

Import BIF file path: C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\HelloWorld\bootimage\HelloWorld.bif

Basic | Security

Output BIF file path: C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\HelloWorld\bootimage\HelloWorld.bif

UDF data:

☐ Split Output format: BIN

Output path: C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\HelloWorld\bootimage\BOOT.bin

Boot image partitions

File path	Encrypted	Authenticated
(bootloader) C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\FSBL\Debug\FSBL.elf	none	none
C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\hw_platform_0\system_top.bit	none	none
C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\HelloWorld\Debug\HelloWorld.elf	none	none

Figure 10: Create Zynq Boot Image Settings

4.1.2 Preparing the Hardware

Step	Description
1	Set the power switch of the Mercury+ PE1 base board to OFF/PCIe (see label **PWR SW** in Figure 2).
2	Disconnect all USB cables from the Mercury+ PE1 base board.
3	Set the configuration DIP switches on the Mercury+ PE1 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON] • CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]
4	Connect the micro USB cable between your computer and the Mercury+ PE1 base board. Use the micro USB port labeled **USBUB** in Figure 2.
5	Set the power switch of the Mercury+ PE1 base board to ON (see label **PWR SW** in Figure 2).

Table 10: Preparing the Hardware for QSPI Flash Boot Mode Step-by-Step Guide

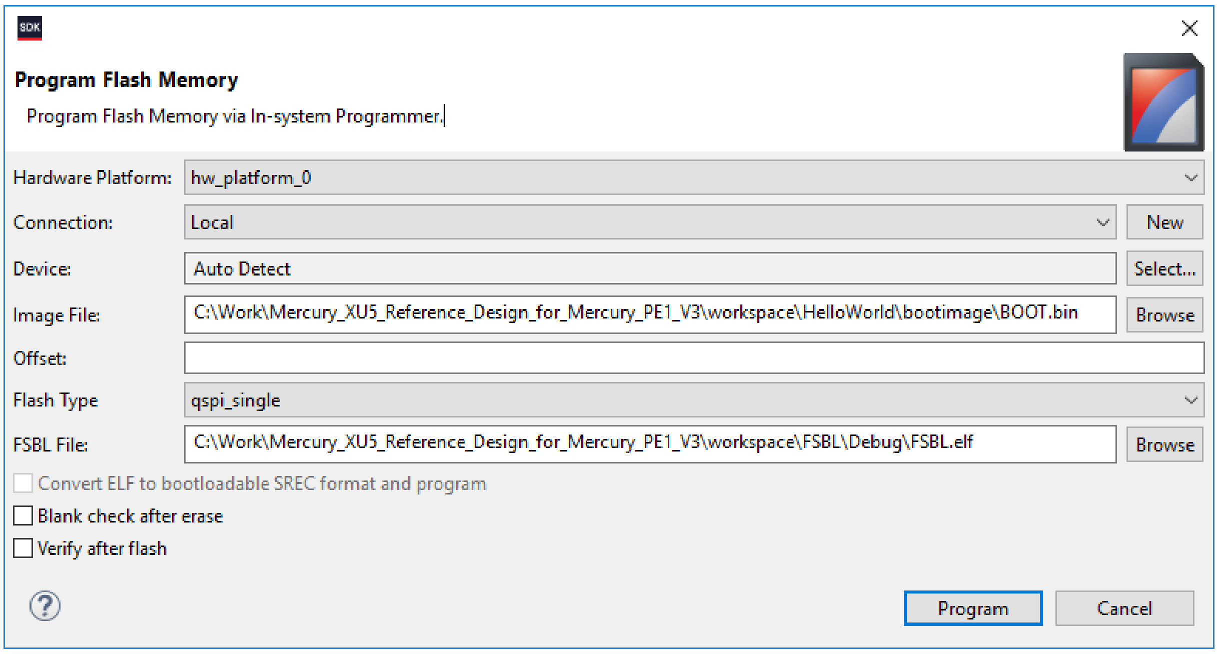
4.1.3 Programming the QSPI Flash

Step	Description
1	Open Xilinx SDK 2018.2: 1. Xilinx Tools → Program Flash 2. In Program Flash Memory window (see Figure 11): (a) For Image File select the boot image generated as described in Section 4.1.1 (b) For FSBL File select the FSBL binary generated as described in Section 3.4 (c) For Flash Type select qspi_single (d) Hit Program and wait for completion The settings in the pictures are for reference only. Note that the configuration file must be selected according to your application. Some Vivado tool versions support QSPI flash programming only in JTAG boot mode. Please check the Mercury+ XU1 SoC module and the Mercury+ PE1 base board user manuals for details on how to configure this boot mode. If this is not available, please use one of the alternative methods.
2*	**Optional** - if SDK returns errors during flash programming or if the system does not boot properly, another option is to use Vivado 2018.2 to program the QSPI flash. 1. Flow → Open Hardware Manager 2. Click on Open target → Auto Connect 3. Right click on the corresponding FPGA device in the left bar → Add Configuration Memory Device (see Figure 12) (a) For Select Configuration Memory Part choose the memory part according to the Mercury+ XU1 SoC Module User Manual [3], part type single. This is in most cases s25fl512s-qspi-x4-single. (b) Hit OK 4. In Program Configuration Memory Device window (see Figure 13): (a) For Configuration file select the boot image generated as described in Section 4.1.1 (b) For Zynq FSBL select the FSBL binary generated as described in Section 3.4 (c) In Program Operations section: • For Address Range select Entire Configuration Memory Device • Enable checkboxes Erase, Program and Verify • Hit OK and wait for completion The settings in the pictures are for reference only. Note that the memory part and the configuration file must be selected according to your application. Some Vivado tool versions support QSPI flash programming only in JTAG boot mode. Please check the Mercury+ XU1 SoC module and the Mercury+ PE1 base board user manuals for details on how to configure this boot mode. If this is not available, please use one of the alternative methods.
3*	**Optional** - alternatively, Enclustra Module Configuration Tool (MCT) [5] can be used to program the QSPI flash. The method presented below is available starting with modules revision 2.

Continued on next page...

Step	Description
	The procedure implies setting another boot mode than QSPI during flash programming, so that the MPSoC does not try to boot while the flash is being programmed. The other boot mode in this case is eMMC boot, therefore the method will be successful only if the eMMC flash is not programmed. 1. Close all other tools that may be connected to the FTDI device (Vivado Hardware Manager, SDK, UART terminal). 2. Set the power switch of the Mercury+ PE1 base board to OFF/PCIe (see label **PWR SW** in Figure 2). 3. Disconnect all USB cables from the Mercury+ PE1 base board. 4. Set CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON] 5. Set CFG B = [1: OFF, 2: ON, 3: ON, 4: OFF] 6. Connect a USB cable to the micro USB port on the Mercury+ PE1 base board (see label **USBUB** in Figure 2) 7. Set CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF] 8. Set the power switch of the Mercury+ PE1 base board to ON (see label **PWR SW** in Figure 2). 9. Perform QSPI flash programming in MCT and close MCT 10. After programming, remove the power supply from the Mercury+ PE1 base board (see label **12 V DC** in Figure 2). 11. Disconnect all USB cables from the Mercury+ PE1 base board and set the power switch of the Mercury+ PE1 base board to OFF/PCIe. 12. Reconnect the USB cable and disconnect and reconnect the UART terminal.

Table 11: Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide

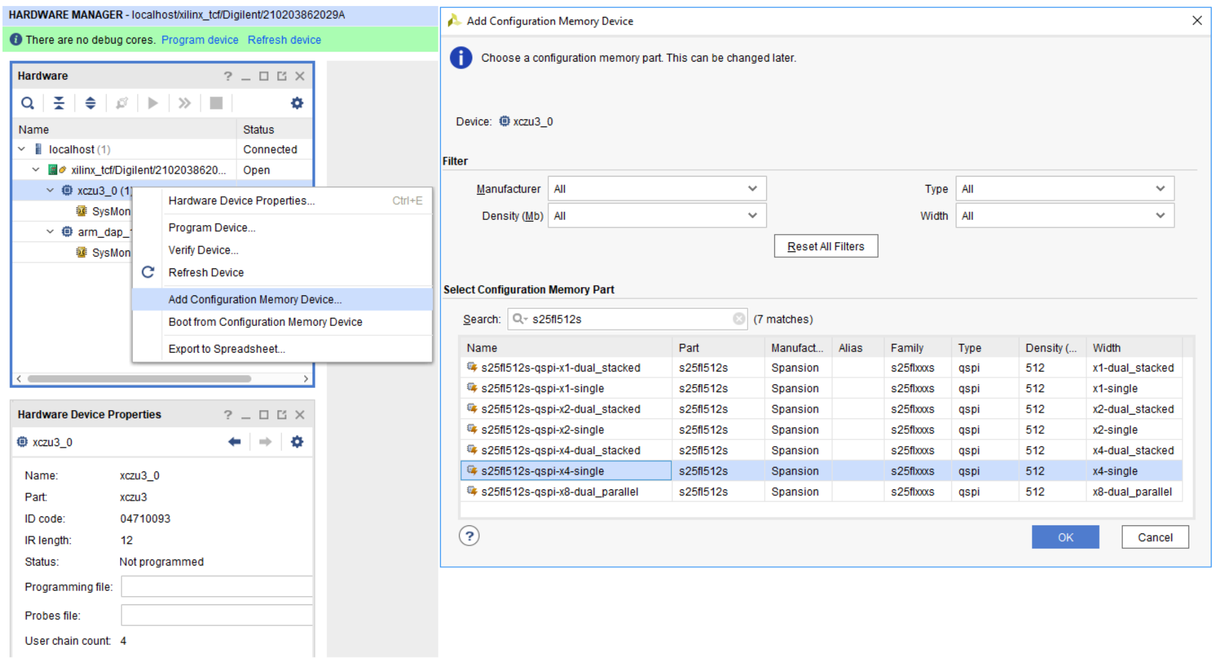

Program Flash Memory

Program Flash Memory via In-system Programmer.

Hardware Platform:	hw_platform_0
Connection:	Local
Device:	Auto Detect
Image File:	C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\HelloWorld\bootimage\BOOT.bin
Offset:	
Flash Type	qspi_single
FSBL File:	C:\Work\Mercury_XU5_Reference_Design_for_Mercury_PE1_V3\workspace\FSBL\Debug\FSBL.elf

☐ Convert ELF to bootloadable SREC format and program
☐ Blank check after erase
☐ Verify after flash

Figure 11: QSPI Flash Programming Settings in SDK


HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210203862029A

There are no debug cores. Program device Refresh device

Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/2102038620...	Open
xczu3_0 (1)	
SysMon	
arm_dap_	
SysMon	

Hardware Device Properties... Ctrl+E
Program Device...
Verify Device...
Refresh Device
Add Configuration Memory Device...
Boot from Configuration Memory Device
Export to Spreadsheet...

Hardware Device Properties

xczu3_0

Name:	xczu3_0
Part:	xczu3
ID code:	04710093
IR length:	12
Status:	Not programmed
Programming file:	
Probes file:	
User chain count:	4

Add Configuration Memory Device

Choose a configuration memory part. This can be changed later.

Device: xczu3_0

Filter

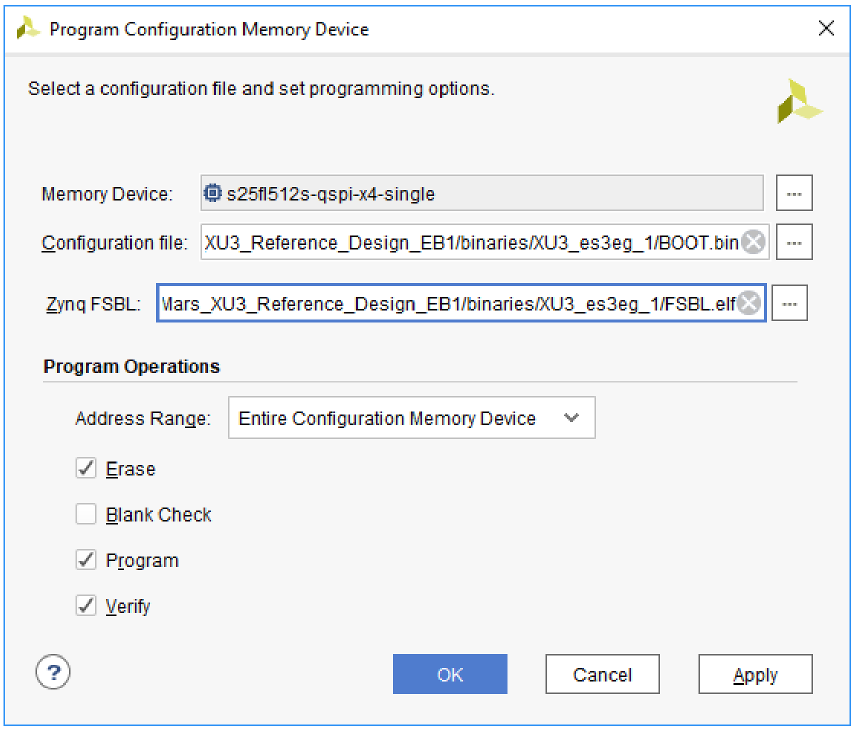
Manufacturer: All
Type: All
Density (Mb): All
Width: All

Select Configuration Memory Part

Search: s25fl512s (7 matches)

Name	Part	Manufact...	Alias	Family	Type	Density (...	Width
s25fl512s-qspi-x1-dual_stacked	s25fl512s	Spansion		s25flxxxs	qspi	512	x1-dual_stacked
s25fl512s-qspi-x1-single	s25fl512s	Spansion		s25flxxxs	qspi	512	x1-single
s25fl512s-qspi-x2-dual_stacked	s25fl512s	Spansion		s25flxxxs	qspi	512	x2-dual_stacked
s25fl512s-qspi-x2-single	s25fl512s	Spansion		s25flxxxs	qspi	512	x2-single
s25fl512s-qspi-x4-dual_stacked	s25fl512s	Spansion		s25flxxxs	qspi	512	x4-dual_stacked
s25fl512s-qspi-x4-single	s25fl512s	Spansion		s25flxxxs	qspi	512	x4-single
s25fl512s-qspi-x8-dual_parallel	s25fl512s	Spansion		s25flxxxs	qspi	512	x8-dual_parallel

Figure 12: QSPI Flash Programming Settings in Vivado - Adding the Memory Device


Program Configuration Memory Device

Select a configuration file and set programming options.

Memory Device: s25fl512s-qspi-x4-single

Configuration file: XU3_Reference_Design_EB1/binaries/XU3_es3eg_1/BOOT.bin

Zynq FSBL: Mars_XU3_Reference_Design_EB1/binaries/XU3_es3eg_1/FSBL.elf

Program Operations

Address Range: Entire Configuration Memory Device

- ✓ Erase
- ☐ Blank Check
- ✓ Program
- ✓ Verify

Figure 13: QSPI Flash Programming Settings in Vivado

Warning!

Some Vivado and SDK tool versions are reporting problems when configuring certain MPSoC devices or when using particular boot modes. Please try different tool versions and check the Xilinx documentation and forums for help on the reported issue.

4.1.4 Booting from the QSPI Flash

Step	Description
1	Set the power switch of the Mercury+ PE1 base board to OFF/PCIe (see label **PWR SW** in Figure 2).
2	Enable the QSPI flash boot mode by setting the configuration DIP switches on the Mercury+ PE1 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: ON, 2: OFF, 3: OFF, 4: ON] • CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]
3	Set the power switch of the Mercury+ PE1 base board to ON (see label **PWR SW** in Figure 2).
4	If you want to reload the configuration, press the power-on reset button (see label **POR** in Figure 2) and release it after a second.

Table 12: Booting from the QSPI Flash Step-by-Step Guide

4.2 SD Card Boot

4.2.1 Generating the Image Files

Please refer to Section 4.1.1 describing the steps required to generate a boot image.

4.2.2 Preparing the Hardware

Step	Description
1	Set the power switch of the Mercury+ PE1 base board to OFF/PCIe (see label **PWR SW** in Figure 2).
2	Enable the SD card boot mode (default) by setting the configuration DIP switches on the Mercury+ PE1 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON] • CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]

Table 13: Preparing the Hardware for SD Card Boot Mode Step-by-Step Guide

4.2.3 Programming the SD Card

Step	Description
1	Write the Xilinx SD card boot image to the SD card 1. Insert the SD card into the SD card slot of your computer 2. Copy the boot image generated for your application from `<workspace>\<app_name>\bootimage\BOOT.bin` to your SD card (directly in the root directory). Note that the name of the image must be preserved.

Table 14: Programming the SD Card for SD Card Boot Mode Step-by-Step Guide

4.2.4 Booting from the SD Card

Step	Description
1	Insert the SD card into the SD card slot of the Mercury+ PE1 base board (see label **SD Card** in Figure 2).
2	Set the power switch of the Mercury+ PE1 base board to ON (see label **PWR SW** in Figure 2).

Table 15: Booting from the SD Card Step-by-Step Guide

5 Troubleshooting

5.1 Vivado Issues

- If the changes in the block design (including licenses for special IPs) are not propagated into implementation, open the Hierarchy tab in Vivado and regenerate the block design files:
  1. Right click on the block design file (.bd)
  2. Click on Reset Output Products → Reset
  3. Click on Generate Output Products → Generate → OK

5.2 SDK Runtime Exceptions

- If the SDK reports runtime exceptions while downloading a program to memory, the following steps should be followed:
  1. Close SDK
  2. Shutdown any javaw, eclipse or xmd processes in Windows Task Manager
  3. Power off the Mercury+ XU1 SoC module
  4. Restart SDK and power on the Mercury+ XU1 SoC module

5.3 JTAG Connection Issues

- If the JTAG cable is not detected, the following steps should be followed:
  1. Make sure that the hardware configuration is made according to Section 3.2
  2. Check that only one JTAG adapter is active and connected to the hardware at a given moment. Make sure that you are not using both built-in JTAG and Xilinx Platform Cable USB. More information on the Xilinx JTAG mode configuration on the Mercury+ PE1 base board can be retrieved from the base board user manual [4].
  3. If built-in JTAG is used, check that the FTDI device is configured to Xilinx JTAG mode. This can be done using the Enclustra MCT software [5].
  4. Remove the USB connection and power supply from the Mercury+ PE1 base board and close SDK
  5. Reconnect the USB and power supply and start SDK again
  6. If built-in JTAG is used, check for UART connection issues (refer to Section 5.4)
  7. Reboot the computer if the problem persists

5.4 UART Connection Issues

- If the computer is not able to recognize the USB UART on the Mercury+ PE1 base board:
  1. Check that the USB cable is connected properly
  2. Check that the FTDI VCP drivers are installed
     (a) Open Device Manager
     (b) Universal Serial Bus controllers → USB Serial Converter A/B → Properties → Advanced tab → enable Load VCP checkbox
     (c) Reboot the computer if the COM port is still not detected
  3. Reinstall the FTDI drivers if the problem persists
- If the computer does not output any character in the terminal program:
  1. Check that the FTDI device is set to UART mode:
     (a) Download and open FT_Prog utility (this is a third party tool offered by the FTDI company to configure FTDI devices)
     (b) DEVICES → Scan and Parse
     (c) Check that for Port A and B the RS232 UART property is true

2. Check that the baud rate for the UART in the block design matches the baud rate set in the terminal program
3. Make sure that Enclustra MCT software is not open. After closing it, unplug and plug in again the USB cable corresponding to the UART communication.

List of Figures

1 Hardware Block Diagram 6
2 Mercury+ PE1 Base Board Assembly Drawing (Top View) 11
3 Board Support PackageSettings 15
4 Importing Software Projects 16
5 Run Configurations Settings 18
6 Hello World Application UART Output 19
7 I2C Example Application UART Output 20
8 Memory Test Application - UART Output Example (addresses and test sizes may vary) . . . 21
9 Flash Test Application UART Output 21
10 Create Zynq Boot Image Settings 24
11 QSPI Flash Programming Settings in SDK 26
12 QSPI Flash Programming Settings in Vivado - Adding the Memory Device 27
13 QSPI Flash Programming Settings in Vivado 27

List of Tables

1 I2C Devices 7
2 UART Configuration 8
3 PS GPIO Configuration 9
4 FPGA Firmware I/O Configuration 9
5 Hardware Setup Step-By-Step Guide 12
6 FPGA Bitstream Generation Step-By-Step Guide 13
7 SDK Workspace Preparation Step-By-Step Guide 15
8 Running an Application Step-By-Step Guide 17
9 Generating the Image Files for QSPI Flash Boot Mode Step-by-Step Guide 23
10 Preparing the Hardware for QSPI Flash Boot Mode Step-by-Step Guide 24
11 Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide 26
12 Booting from the QSPI Flash Step-by-Step Guide 28
13 Preparing the Hardware for SD Card Boot Mode Step-by-Step Guide 28
14 Programming the SD Card for SD Card Boot Mode Step-by-Step Guide 28
15 Booting from the SD Card Step-by-Step Guide 29

References

[1] Vivado Design Suite User Guide, Embedded Processor Hardware Design, UG898, Xilinx, 2016
[2] Zynq UltraScale+ MPSoC: Embedded Design Tutorial, A Hands-On Guide to Effective Embedded System Design, UG1209, Xilinx, 2017
[3] Mercury+ XU1 SoC Module User Manual
→ Ask Enclustra for details
[4] Mercury+ PE1 Base Board User Manual
→ Ask Enclustra for details
[5] Enclustra Module Configuration Tool (MCT)
→ Ask Enclustra for details
[6] Enclustra Modules Heat Sink Application Note
→ Ask Enclustra for details