Mercury XU5 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 XU5 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 XU5 SoC module.

Summary

This document first gives an overview of the Mercury XU5 SoC module reference design and then guides through the complete Xilinx MPSoC design flow for the Mercury XU5 SoC module in the getting started section. In addition, the internals and the boot options of the Mercury XU5 SoC module reference design are described.

Product Information	Code	Name
Product	ME-XU5	Mercury XU5 SoC Module

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-445-002	2020.1_v1.1.0	04.02.2021

Approval Information	Name	Position	Date
Written by	DDUE/ARUD	Design Engineer	25.01.2021
Verified by	GKOE	Design Expert	26.01.2021
Approved by	DIUN	Manager, BU SP	04.02.2021

License

Table of Contents

1 Overview

1.1 Introduction

The Mercury XU5 SoC module reference design demonstrates a system using the Mercury XU5 SoC module in combination with the Mercury+ PE1 base board. It presents the basic configuration of the device and contains a guided getting started tutorial.

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.

This reference design does not include any source code for software examples and instead Enclustra provides Application Notes [9] for some selected applications.

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 [3]

More information on the Mercury XU5 SoC module and the Mercury+ PE1 base board can be retrieved from their respective user manuals [4] [5].

The following directory structure applies to the XU5 Reference Design:

- `src` — Xilinx pinout and timing constraints and VHDL source code directory
- `scripts` — Scripts directory required for Vivado project creation
- `doc` — Reference Design documentation

Pre-generated binaries for any XU5 variant are released on the XU5 Reference Design Github page.

1.2 Prerequisites

- IT
  - A computer with a microSD card slot (optional[1]) running Windows 10 64-bit (or later)
- Software
  - Xilinx Vivado 2020.1 WebPack, Evaluation, Design or System Edition (check the Mercury XU5 SoC Module User Manual [4] for details on device support in Xilinx tools)
  - Xilinx Vitis IDE
  - Enclustra Module Configuration Tool (MCT) [6] (optional[2])
  - A terminal emulation program (e.g. Tera Term)
- Hardware
  - An Enclustra Mercury XU5 SoC module
  - An Enclustra Mercury+ PE1 base board
- Accessories
  - A 12 V DC power supply

[1] Only required for SD card boot mode
[2] May be used for flash programming, for MPSoC device configuration or for FTDI configuration.

- A standard micro USB cable
- A Xilinx JTAG programmer (e.g. Platform Cable USB II) (optional[3])

[3] Any FTDI device present on Enclustra hardware can be configured to Xilinx JTAG mode using the Enclustra MCT software [6].

2 Reference Design Description

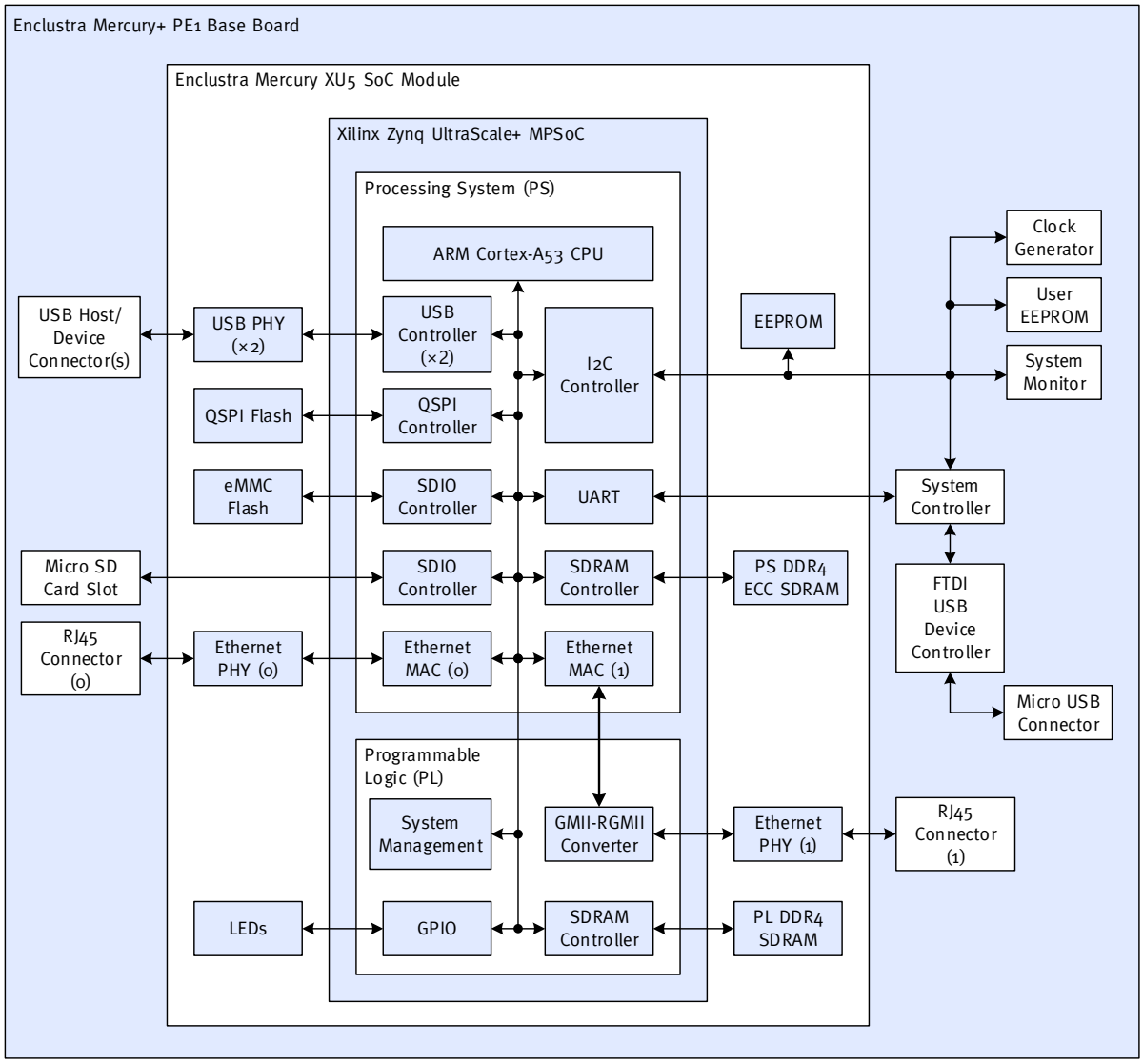

Figure 1: Hardware Block Diagram

2.1 Processing System (PS)

2.1.1 Clocks

The PS input clock frequency is configured to 33.33 MHz. The CPU clock frequency is configured to its corresponding maximum APU clock frequency, as specified in the Zynq UltraScale+ MPSoC Data Sheet (DS925) [2]. The maximum CPU (APU) clock performance depends on the device speedgrade and package. Beside that a 50 MHz and a 100 MHz clock are exported from PS to the PL.

These clocks can be modified in the settings of the processing system in Vivado.

2.1.2 PS DDR4 SDRAM

The DDR4 SDRAM memory runs at its corresponding maximum PS DDR frequency. Depending on the module variant ECC RAM is enabled or disabled (see Mercury XU5 SoC Module User Manual [4] for details).

Note that the maximum DDR performance depends on the device speedgrade and package as specified in the Zynq UltraScale+ MPSoC Data Sheet (DS925) [2]. The DDR clock frequency can be modified in the settings of the processing system in Vivado and must be configured according to the Mercury XU5 SoC Module User Manual [4].

2.1.3 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 XU5 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.

2.1.4 eMMC

The eMMC interface is configured in the PS to the MIO 13..22 pins. This enables eMMC device access, as well as booting from the eMMC device. To allow the Mercury XU5 SoC module to boot from the eMMC, the boot signals must be configured as in 4.3.4.

For further details refer to the Mercury XU5 SoC Module and Mercury+ PE1 Base Board User Manual [4] [5].

2.1.5 I2C

The I2C controller I2C0 is configured to the MIO 10..11 pins. For available devices on the I2C bus refer to the Mercury XU5 SoC Module and Mercury+ PE1 Base Board User Manual [4] [5].

2.1.6 Quad SPI Flash Controller

The quad SPI flash controller is connected to MIO 0..6 pins in Single mode. To allow the Mercury XU5 SoC module to boot from the QSPI flash, the hardware configuration on the Mercury+ PE1 base board must be done according to Section 4.1.1.

2.1.7 UART

The UART0 is mapped to MIO 38..39 pins and connected to the FTDI 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

2.1.8 Ethernet

The Ethernet MAC GEM0 is mapped to MIO 26..37 pins and is connected to the Microchip (Micrel) KSZ9031 Ethernet PHY on the Mercury XU5 SoC module using an RGMII interface. The PHY can be configured via the MDIO management interface on PHY address 3.

A second Ethernet MAC GEM1 is mapped to the PL via EMIO. Refer to the section 2.2.2 for details.

2.1.9 USB

Two USB3320C USB 2.0 PHYs are available on the Mercury XU5 SoC module, both connected to the PS to I/O bank 502. USB PHY 0 can be configured as host or device, while USB PHY 1 can be used only as host.

The first USB controller USB0, is mapped on MIO 52..63 pins, while the second USB controller USB1, is mapped on MIO 64..75.

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 [5] for details.

2.1.10 GPIOs

The unused MIO pins from the PS are available as GPIOs. For details on the MIO assignement refer to the Multiplexed I/O (MIO) Pins section in the Mercury XU5 SoC module User Manual [4]. Check the connectivity of the MIOs that provide user functionality with the Mercury+ PE1 base board User Manual [5].

2.1.11 Video Codec Unit (VCU)

Xilinx Zynq Ultrascale+ modules with an EV device equipped, contain an embedded hard IP H.264/H.265 Video Codec Unit (VCU). For these module variants Enclustra will provide a Video Application Note describing the required implementation to support this feature. Please ask Enclustra for details (`info@enclustra.com`).

2.2 Programmable Logic (PL)

2.2.1 PL DDR4 SDRAM

The PL dedicated DDR4 SDRAM memory runs at its corresponding Maximum Physical Interface (PHY) Rate at a voltage of 1.2 V. The data width of this interface is 16 bits.

Note that the maximum DDR performance depends on the device speedgrade, the package and the VCC_INT voltage (0.85 V by default) as specified in the Zynq UltraScale+ MPSoC Data Sheet (DS925) [2]. The DDR clock frequency can be modified in the settings of Memory Interface Generator (MIG) IP core in Vivado and must be configured according to the Mercury XU5 SoC Module User Manual [4].

2.2.2 PL Ethernet

The Ethernet MAC GEM1 is mapped to EMIO pins, providing a GMII interface to the FPGA logic. This interface is routed to a GMII-to-RGMII converter IP core from Enclustra; the RGMII interface is available on FPGA bank E (25 for ZU2/ZU3 and 45 for ZU4/ZU5), for connection to one of the two Microchip (Micrel) KSZ9031 Ethernet PHYs. This PHY can be configured via the PL dedicated MDIO management interface on PHY address 3.

The GMII-to-RGMII converter IP core is provided as-is without further documentation or source files. Support for the GMII-to-RGMII converter can be supplied as a part of a support package or design services.

Please refer to the Mercury XU5 SoC Module User Manual [4] for the usage and configuration of the Ethernet PHYs.

2.2.3 GPIOs

A Xilinx GPIO controller in the PL is connected to the PS via an AXI bus. Some PL GPIOs are connected to LEDs in the top level, as described in Table 3.

The PL firmware contains a 24-bit counter freely running at 50 MHz. The MSB of this counter is used to blink LED1#_PL with a frequency of approximately 3 Hz.

PL Pin	Signal	Function
H2	LED1#_PL	Blinking LED counter MSB
P9	LED2#_PL	GPIO 0, controlled by the PL GPIO controller
K5	LED3#_PL	GPIO 1, controlled by the PL GPIO controller

Table 3: PL Firmware I/O Configuration

2.2.4 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 the FPGA is configured to its maximum allowed temperature.

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 XU5 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.

3 Getting Started

This section describes the steps required to configure the Mercury XU5 SoC module and Mercury+ PE1 base board in order to run a simple HelloWorld example application. The section includes information on how to:

- Mount the module and configure the Mercury+ PE1 base board
- Generate the PL bitstream
- Prepare the software workspace
- Run a software application

3.1 Essential Information

Warning!

Always check that the mounting holes on the Mercury+ PE1 base board are aligned with the mounting holes of the Mercury XU5 SoC module. The base board and module may be damaged if the module is mounted the wrong way round and powered up.

If the module cannot be mounted correctly due to the mechanical collision, please contact Enclustra support.

Warning!

Never mount or remove the Mercury XU5 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 XU5 SoC module.

Warning!

Please read carefully the Mercury XU5 SoC module and Mercury+ PE1 base board user manuals before proceeding.

Warning!

Depending on the user application, the Mercury XU5 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 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.

Note that when Enclustra MCT [6] is used for MPSoC configuration or flash programming, all other tools that may be connected to the FTDI device (e.g. Vivado Hardware Manager, Vitis, UART terminal) must be closed.

3.2 Hardware Setup

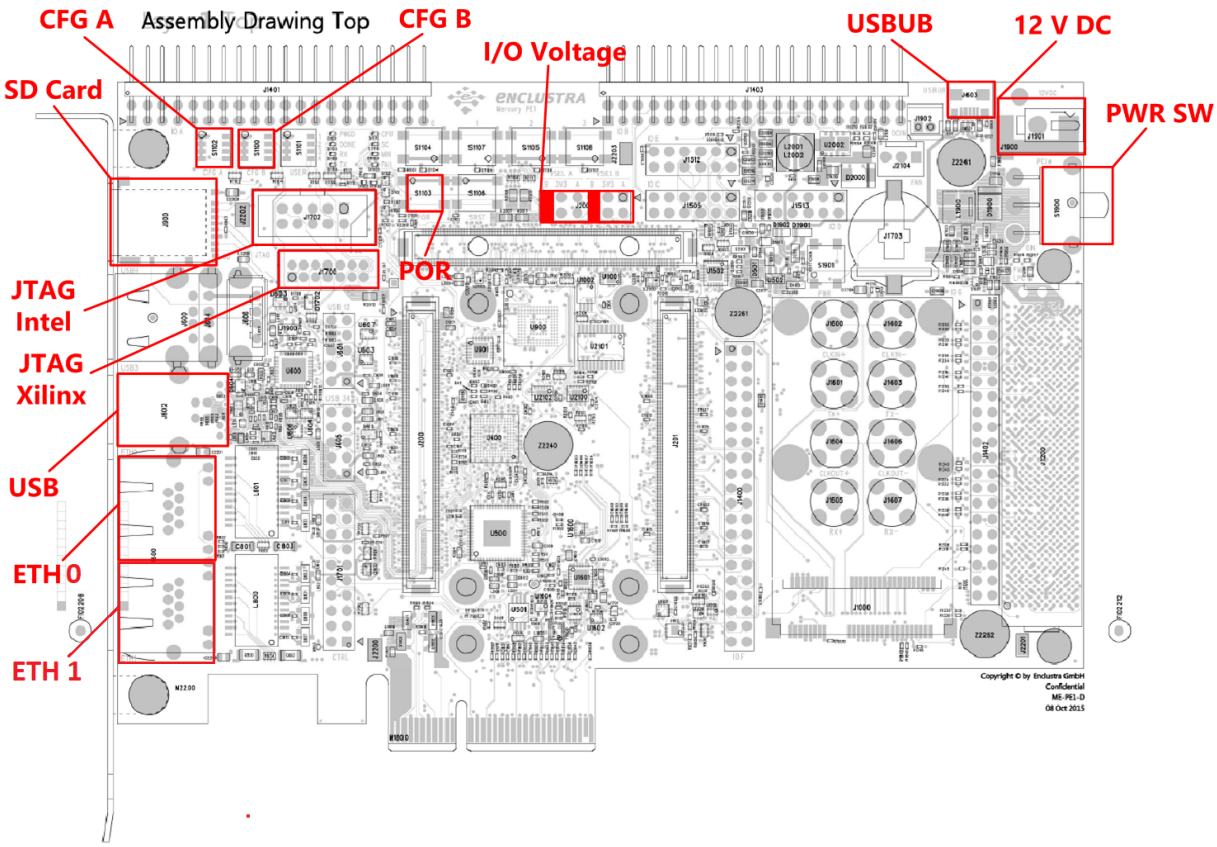

Figure 2: Mercury+ PE1 Base Board Assembly Drawing (Top View)

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 XU5 SoC module to the Mercury+ PE1 base board. Make sure that the mounting holes of the Mercury XU5 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	Make sure that the FTDI device on the Mercury+ PE1 base board is configured to Xilinx JTAG mode using Enclustra MCT [6] (or alternatively, connect the JTAG signals from the Xilinx Platform Cable USB to the JTAG connector of the Mercury+ PE1 base board (see label **JTAG Xilinx** in Figure 2). Details on the Xilinx JTAG mode configuration and on the JTAG connector are presented in the Mercury+ PE1 Base Board User Manual [5]
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 2.1.7.

Table 4: Hardware Setup Step-By-Step Guide

3.3 FPGA Bitstream Generation

For a fast test of the HelloWorld example application, the pre-generated bitstream may alternatively be used, therefore the steps described in this section may be skipped.

A pre-generated bitstream for any XU5 variant is released on the XU5 Reference Design Github page.

Step	Description
1	Configure the settings file: 1. Edit the `module_name` variable in `scripts/settings.tcl` file, according to your modules name. This file includes module name and board information required for the project creation script. All settings, except for `module_name` should be left on default. The list of options for `module_name` is given in the comments within the Tcl file. 2. Save the file after editing.

Continued on next page...

Step	Description
2	Start Xilinx Vivado 2020.1 and create the Mercury XU5 SoC module reference design project: 1. Click on the Tcl console at the bottom of the page and type: (a) `cd {<base_dir>}` where `<base_dir>` is the directory in which you extracted the archive contents. Note the `{}` around the path. (b) `source ./scripts/create_project.tcl` (c) Alternatively the script can be run by passing the module name as an argument instead of changing the `module_name` variable in the `settings.tcl` file: i. `vivado -mode batch -source ./scripts/create_project.tcl -tclargs <module_name>`. ii. `open_project ./Vivado/<module_name>/<project_name>.xpr` 2. Wait for completion
3	Run Synthesis, Implementation & Bitstream Generation in Vivado 2020.1: 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 Vitis IDE): 1. File → Export → Export Hardware 2. Choose Fixed for platform type and click Next 3. Select Include Bitstream and click Next 4. Leave the file name and export location as default and click Next 5. Click Finish

Table 5: FPGA Bitstream Generation Step-By-Step Guide

3.4 Vitis Workspace Preparation

This section describes how to create and run software example applications. The steps are generic, and apply to the software example templates in the Vitis IDE.

A pre-generated binary file of the HelloWorld example application and a hardware description file for any XU5 variant is released on the XU5 Reference Design Github page.

Step	Description
1	Start the Vitis IDE 2020.1 1. Select any workspace (e.g. `<base_dir>\workspace`)

Continued on next page...

Step	Description
2	Create a new Platform Project 1. File → New → Platform Project 2. In the New Platform Project: (a) For Project Name type the `<project_name>` e.g. Mercury_XU5_PE1 (b) Hit Next (c) Select "Create a new platform from hardware (XSA)" (d) Hit the Browse button and select the Hardware Specification .xsa file you exported from Vivado, as described in Section 3.3. The default export location used by Vivado is `<base_dir>\<vivado_proj_dir>\<project_name>.xsa` Alternatively, the pre-compiled hardware description file may be used. (e) Wait for the file to be analyzed (f) For the Operating System select standalone (g) For the Processor select psu_cortexa53_0 (h) Check the "Generate boot components" checkbox, if it is not enabled yet. Vitis will then automatically generate the binaries for the FSBL & PMUFW. (i) Hit Finish and wait for completion 3. Build the platform by pressing Ctrl-B and wait for completion
3	Create a new application 1. File → New → Application Project 2. In the New Application Project window: (a) Click Next if Welcome Page is displayed (b) Select the previously generated platform and click Next (c) For Project Name type a description for the new application e.g. "HelloWorld" (d) For the System project select "Create New..." and use the default naming e.g. "HelloWorld_system" and click Next (e) For the Domain choose "standalone on psu_cortexa53_0" (f) Hit Next and wait for the tool to proceed (g) Select the HelloWorld (or any another) template [4] (h) Hit Finish and wait for completion 3. Build the application by pressing Ctrl-B and wait for completion 4. Add the pmufw.elf to BOOT.bin (a) Right-click on the Application Project and select "Create Boot Image" (b) Click on add in the bottom right and navigate to the already created pmufw.elf file (c) Change partition type to "pmu" (d) Click on "Ok" (e) Click on "Create Image" to generate the modified BOOT.bin file

Table 6: Vitis Workspace Preparation Step-By-Step Guide

[4] Depending on the selected sample project changes to the platform BSP might be necessary.

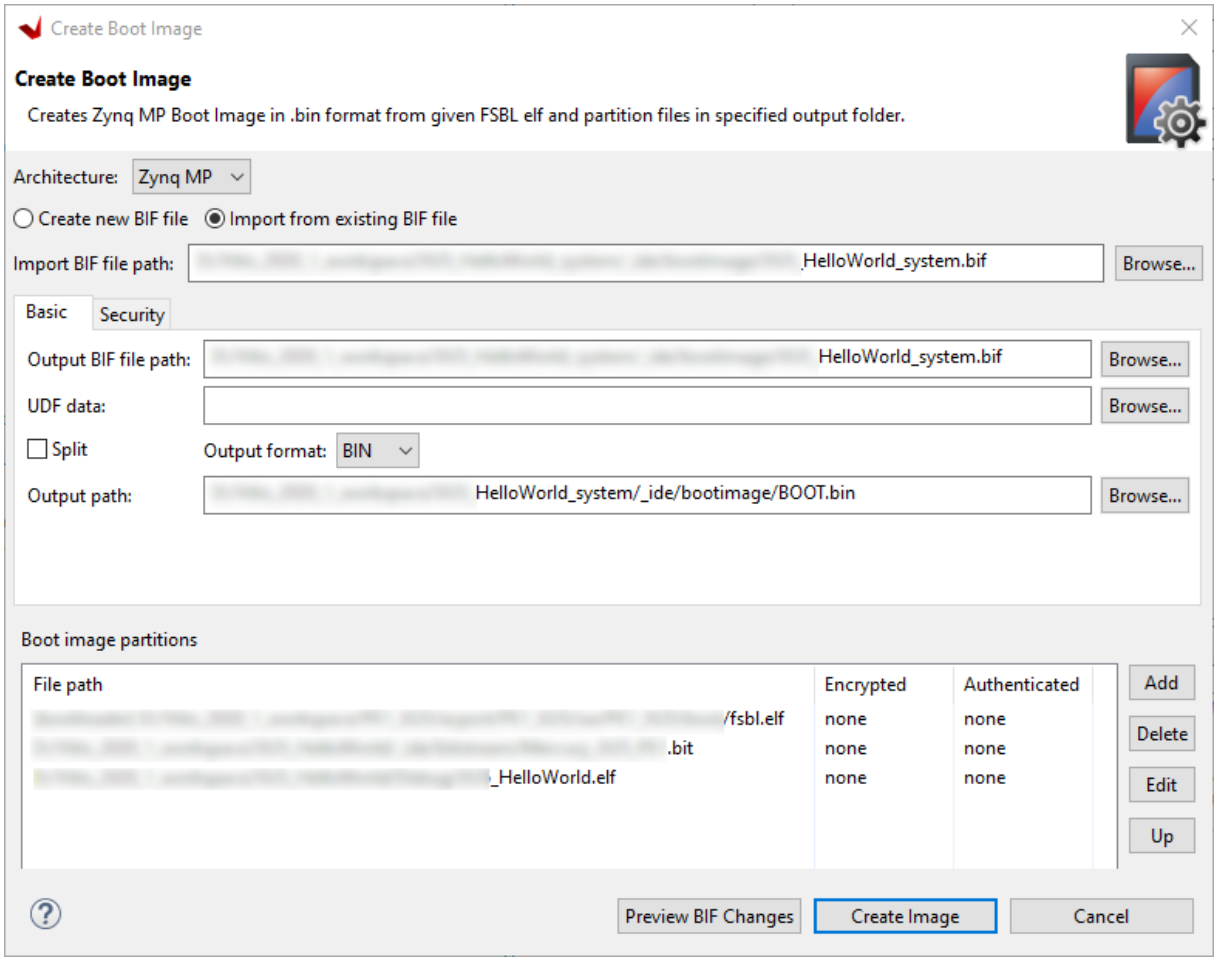

Figure 3: Create Boot Image

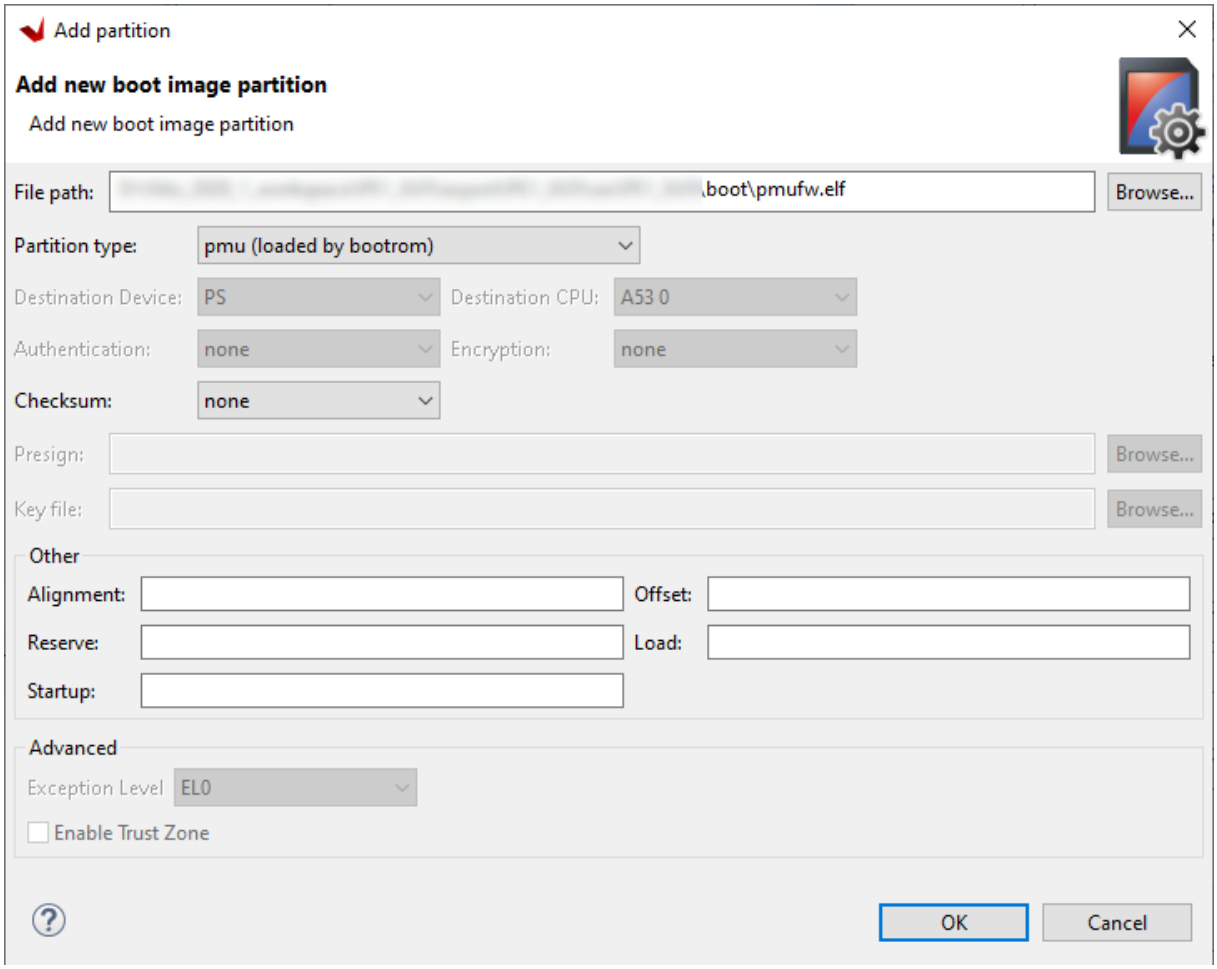

Figure 4: Add pmu to boot image

3.5 Running Software Applications

This section describes how to run software applications on the Mercury XU5 SoC module. The steps are generic, and apply to the software example templates in the Vitis IDE.

Step	Description
1	Create a run configuration for the application in Vitis IDE 2020.1: 1. Right click the previously generated application (e.g. HelloWorld) and select Run As → Run Configurations... 2. Right-click Single Application Debug and hit New or double-click on it 3. Enter a run configuration name in the Name field (e.g. HelloWorld) 4. 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 5. Target Setup tab (see Figures 5 and 6): (a) For Hardware Platform refer to the corresponding Platform: e.g. `${sdxTcfLaunchFile:project=HelloWorld;fileType=hw;}` (b) For Bitstream file field, hit Search... (c) Select Mercury_XU5_PE1.bit and hit OK (d) For FPGA Device and PS Device, use Auto Detect option (e) Uncheck the "Use FSBL flow for initialisation" (f) In the Initialization File field, hit Search... (g) Select psu_init.tcl and hit OK (h) Enable checkboxes Reset entire system, Reset APU, Program FPGA, Run psu_init and PL Powerup (i) Hit Apply
2	Make sure the Hardware is configured according to Section 3.2: → 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). → With a serial console program e.g. Tera Term connect to the COM port that corresponds to the Serial Converter B. For issues related to UART, refer to Section 5.4.
3	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 PLwith the specified bitstream and downloads the application program to the ARM processor. In some test setup cases it was observed that the Vitis 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 7: Running an Application Step-By-Step Guide

After the PL is successfully configured, the **DONE** LED should be lit. When the application is running successfully, the output of the HelloWorld application should appear on the UART console.

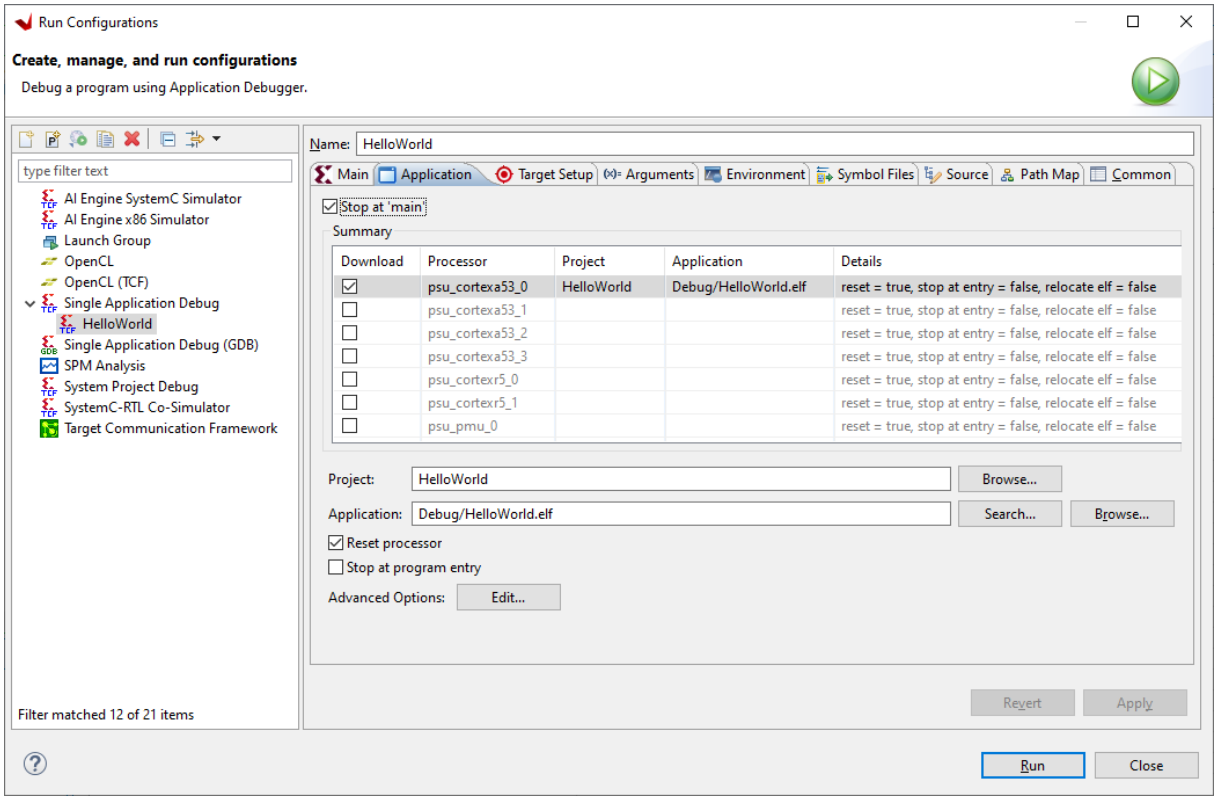

Figure 5: Run Configurations Settings - Application Tab

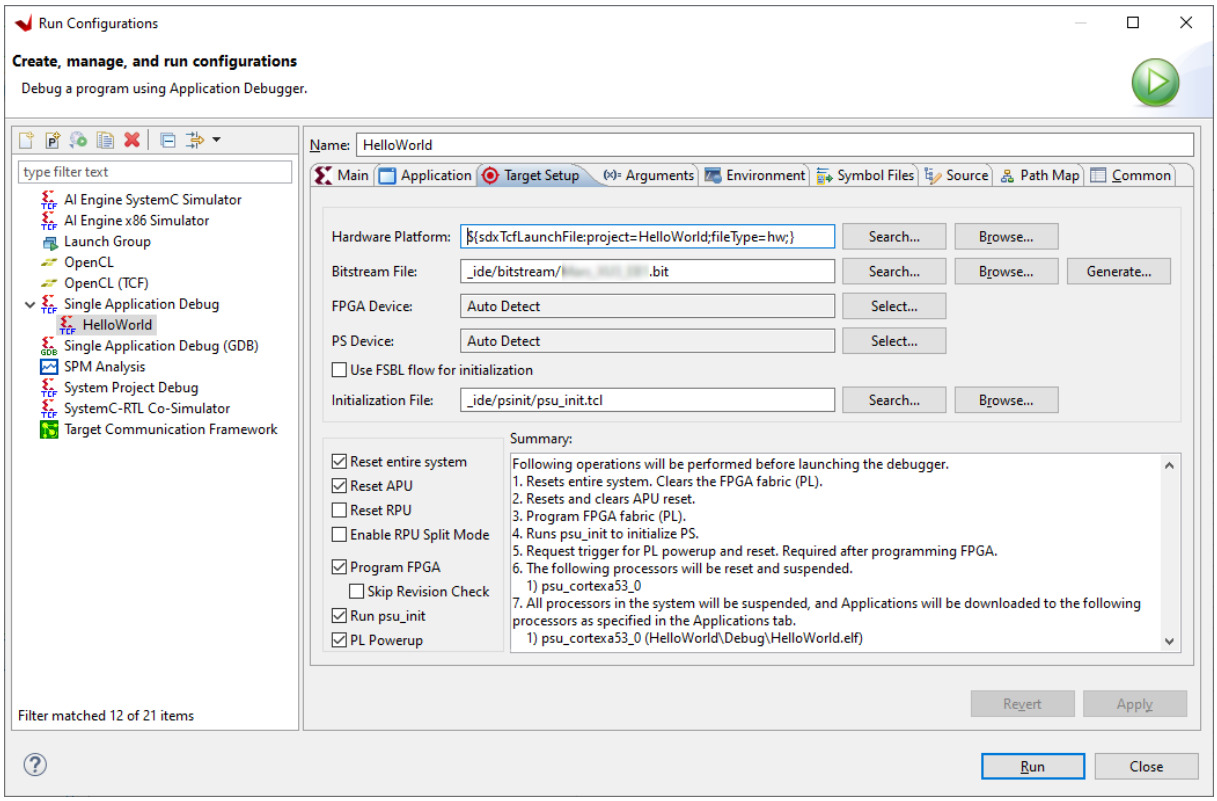

Figure 6: Run Configurations Settings - Target Setup Tab

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.

For a fast test of the boot configurations, the pre-generated .bin images may be used for boot, instead of rebuilding the image. You need to select the file corresponding to the Mercury XU5 SoC module variant. Pre-generated binaries for any XU5 variant are released on the XU5 Reference Design Github page.

4.0.1 Generating the Image File

Step	Description
1	Create the boot image from Xilinx Vitis 2020.1 (see Figure 7): 1. Right click on the application in the Project Explorer 2. Select Create Boot Image → Create Image An image will be created in `<workspace>\HelloWorld\_ide\bootimage\BOOT.bin`.

Table 8: Generating the Boot Image File Step-by-Step Guide

Beside the methods presented in this section, there are additional methods how a boot device can be programmed. Please refer to the Enclustra Build Environment's User Documentation for details [8].

4.1 QSPI Flash Boot

4.1.1 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: ON, 2: OFF, 3: OFF, 4: ON] • CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]

Continued on next page...

Step	Description
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 9: Preparing the Hardware for QSPI Flash Boot Mode Step-by-Step Guide

4.1.2 Programming the QSPI Flash

Step	Description
1	Program the boot image from Xilinx Vitis 2020.1 (see Figure 7): 1. Right click on the application in the Project Explorer 2. Select Program Flash Vitis will fill out the fields for the selected application automatically. 3. For Flash Type select qspi-x4-single 4. 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.
2*	**Optional** - if Vitis returns errors during flash programming or if the system does not boot properly, another option is to use Vivado to program the QSPI flash. 1. Flow → Open Hardware Manager 2. Click on Open target → Auto Connect 3. Right click on the corresponding MPSoC device in the left bar → Add Configuration Memory Device (see Figure 8) (a) For Select Configuration Memory Part choose the memory part according to the Mercury XU5 SoC Module User Manual [4], part type single. This is in most cases s25fl512s-1.8v-qspi-x4-single. (b) Hit OK 4. In Program Configuration Memory Device window (see Figure 9): (a) For Configuration file select the boot image generated as described in Section 4.0.1 (b) For Zynq FSBL select the FSBL binary generated with the Platform 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.

Continued on next page...

Step	Description
3*	**Optional** - alternatively, Enclustra Module Configuration Tool (MCT) [6] can be used to program the QSPI flash. 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, Vitis, 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: ON, 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 10: Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide

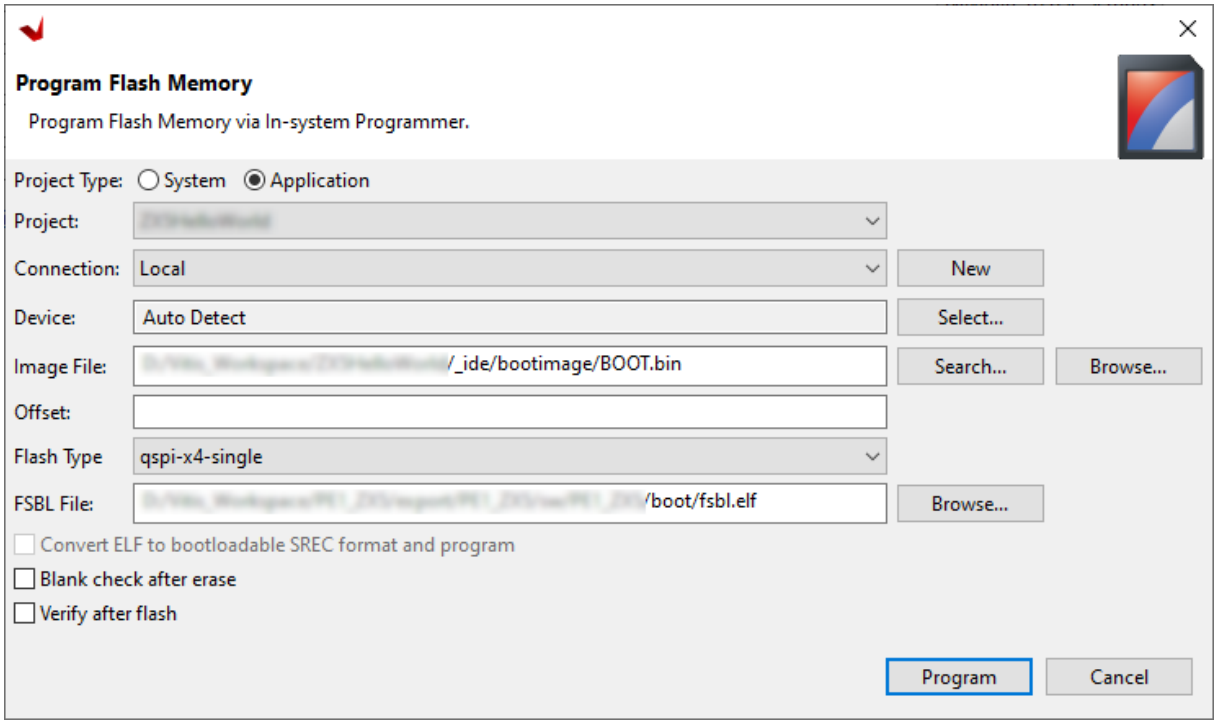

Figure 7: QSPI Flash Programming Settings in Vitis

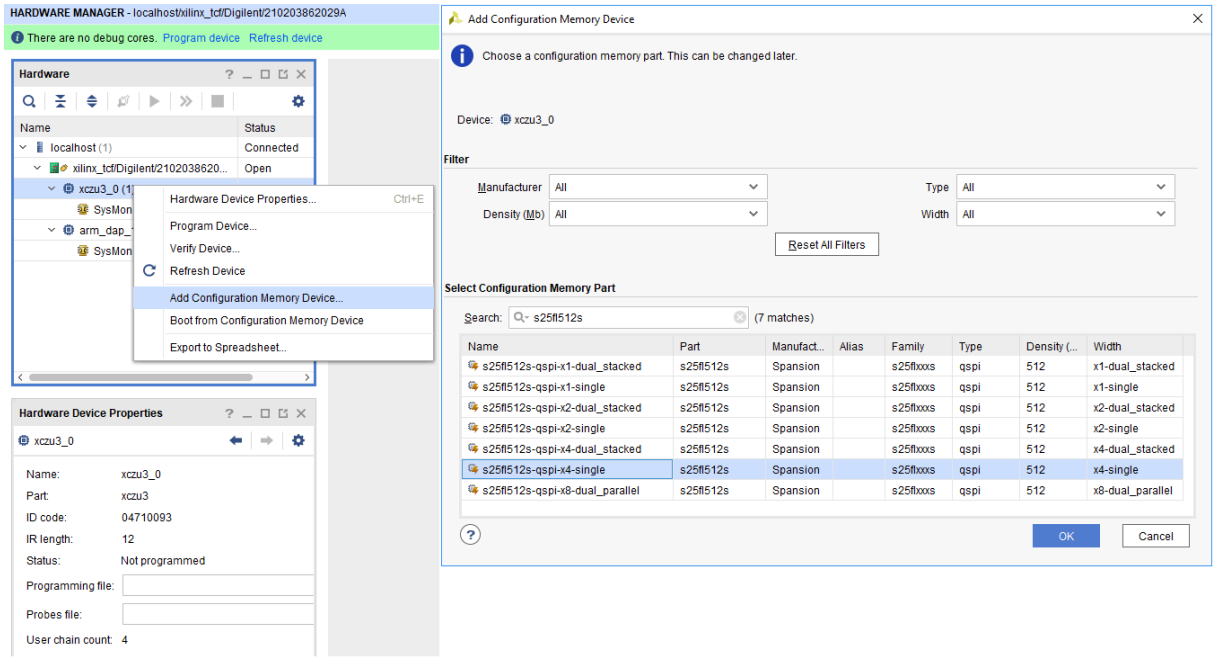

Figure 8: QSPI Flash Programming Settings in Vivado - Adding the Memory Device

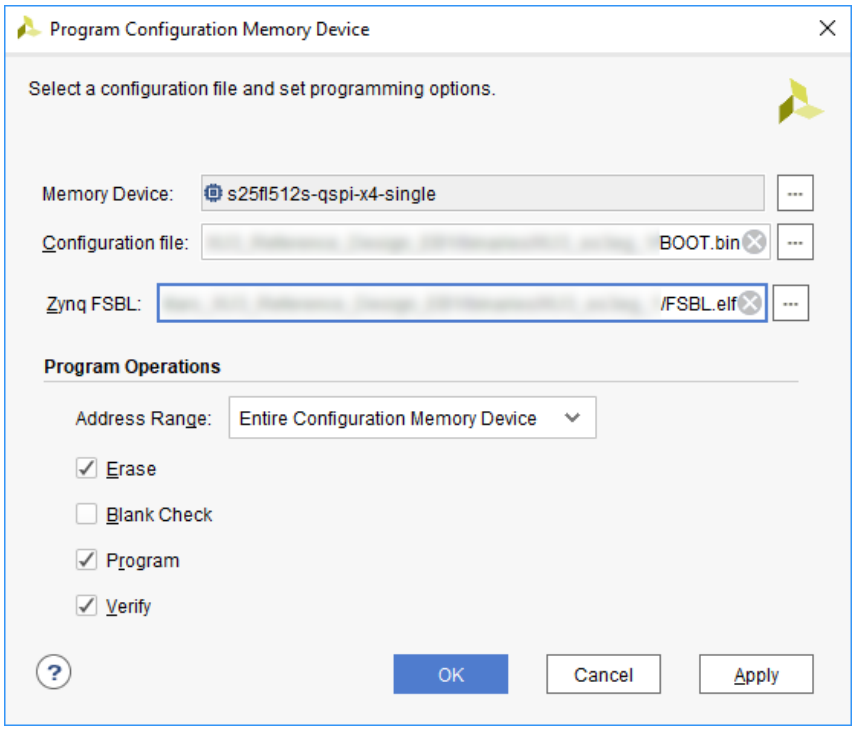

Figure 9: QSPI Flash Programming Settings in Vivado

Warning!

Some Vivado and Vitis 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.

4.1.3 Booting from the QSPI Flash

Step	Description
1	Check that the hardware configuration is done according to Section 4.1.1.
2	Press the power-on reset button (see label **POR** in Figure 2) and release it after a second.

Table 11: 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 4.0.1 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 12: 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 a FAT32 formatted SD card 1. Insert the SD card into the SD card slot of your computer 2. Copy the boot image generated for your application to your SD card (directly in the root directory). Note that the name of the image must be preserved.

Table 13: 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 14: Booting from the SD Card Step-by-Step Guide

4.3 eMMC Boot

4.3.1 Generating the Image Files

Please refer to Section 4.0.1 describing the steps required to generate a boot image.

4.3.2 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	Enable the eMMC 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): 1. Disconnect all USB cables from the Mercury+ PE1 base board. 2. Set CFG A = [1: ON, 2: OFF, 3: OFF, 4: ON] 3. Set CFG B = [1: OFF, 2: ON, 3: ON, 4: OFF] 4. Connect a USB cable to the micro USB port on the Mercury+ PE1 base board (see label **USBUB** in Figure 2). 5. Set CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF] 6. Connect the power supply to the Mercury+ PE1 base board (see label **12 V DC** in Figure 2). 7. Reconnect the UART terminal.

Table 15: Preparing the Hardware for eMMC Mode Step-by-Step Guide

4.3.3 Programming the eMMC

Step	Description
1	Program the boot image from Xilinx Vitis 2020.1 (see Figure 7): 1. Right click on the application in the Project Explorer 2. Select Program Flash Vitis will fill out the fields for the selected application automatically. 3. For Flash Type select emmc 4. 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 eMMC flash programming only in JTAG boot mode. Please check the Mercury XU5 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 an alternative method or contact Enclustra for support.

Continued on next page...

Step	Description
2*	**Optional** - if Vitis returns errors during flash programming or if the system does not boot properly, another option is to use Vivado to program the eMMC. 1. Flow → Open Hardware Manager 2. Click on Open target → Auto Connect 3. Right click on the corresponding MPSoC device in the left bar → Add Configuration Memory Device (see Figure 8) (a) For Select Configuration Memory Part choose emmc. This is in most cases jedec4.51-16gb-emmc. (b) Hit OK 4. In Program Configuration Memory Device window (see Figure 9): (a) For Configuration file select the boot image generated as described in Section 4.0.1 (b) For Zynq FSBL select the FSBL binary generated with the Platform 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 eMMC flash programming only in JTAG boot mode. Please check the Mercury XU5 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 an alternative method or contact Enclustra for support.

Table 16: Programming the eMMC for eMMC Boot Mode Step-by-Step Guide

Warning!

Some Vivado and Vitis 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.

4.3.4 Booting from the eMMC

Step	Description
1	Check that the hardware configuration is done according to Section 4.3.2.
2	Press the power-on reset button (see label **POR** in Figure 2) and release it after a second.

Table 17: Booting from the eMMC 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
- In Vivado 2019.2 and later versions it is not possible to program the PL if the device is not in JTAG boot mode. Please refer to AR75416 for a workaround.

5.2 Vitis Issues

- If the platform generation in Vitis is not successful or the generated platform is not selectable for applications:
  1. Close Vitis
  2. delete the workspace folder
  3. Restart Vitis and try creating the platform again.
- If Vitis shows the warning "There's no DDR_1 in the HW design. MMU translation table marks 32 GB DDR... " please check if more than 2GB PS DDR4 memory should be available. For detailed information please check the Xilinx Answer Records and Forum about this warning.

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. 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 [6]. More information on the Xilinx JTAG mode configuration on the Mercury+ PE1 base board can be retrieved from the Mercury+ PE1 base board user manual [5].
  3. 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.
  4. Remove the USB connection and the power supply from the Mercury+ PE1 base board and close Vitis
  5. Reconnect the USB and power supply and start Vitis again
  6. Check for UART Connection Issues (refer to Section 5.4)
  7. Reboot the computer if the problem persists
- If no device is detected, shutdown the hw_server process e.g. in the Windows Task Manager and try again.

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.

5.5 QSPI Boot Issues

- If the Mercury XU5 SoC module is not able to boot from the QSPI flash:
  1. Use Vivado to program the flash
     (a) Make sure that the Memory Device part type is correctly selected
     (b) Make sure Erase and Program options are enabled
     (c) Select Entire Configuration Memory Device for Address Range
  2. If the problem persists, a possible solution is to first erase the flash, and then program it either from Vivado or Vitis

  Please refer to Section 4.1.2 for details on QSPI flash programming.

5.6 Emmc Boot Issues

- If the Mercury XU5 SoC module is not able to boot from the eMMC flash:
  1. Use Vivado to program the eMMC
     (a) Make sure that the Memory Device part type is correctly selected
     (b) Make sure Erase and Program options are enabled
     (c) Select Entire Configuration Memory Device for Address Range
  2. If the problem persists, a possible solution is to first erase the eMMC, and then program it either from Vivado or Vitis

  Please refer to Section 4.3.3 for details on eMMC flash programming.
- With Xilinx 2020.1 tool version eMMC programming gets stuck and does not progress further. A possible workaround is to use tool version 2019.2 and select the binaries and FSBL generated with 2020.1.

List of Figures

List of Tables

References

[1] Vivado Design Suite User Guide, Embedded Processor Hardware Design, UG898, Xilinx, 2019
[2] Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics, DS925, Xilinx 2019
[3] Zynq UltraScale+ MPSoC: Embedded Design Tutorial, A Hands-On Guide to Effective Embedded System Design, UG1209, Xilinx, 2019
[4] Mercury XU5 SoC Module User Manual
→ Ask Enclustra for details
[5] Mercury+ PE1 Base Board User Manual
→ Ask Enclustra for details
[6] Enclustra Module Configuration Tool (MCT)
`https://www.enclustra.com/en/products/tools/module-configuration-tool/`
[7] Enclustra Modules Heat Sink Application Note
→ Ask Enclustra for details
[8] Enclustra Build Environment
`https://github.com/enclustra-bsp/bsp-Xilinx`
[9] Enclustra Application Notes
`https://github.com/enclustra/I2CAppNote`
`https://github.com/enclustra/GigabitEthernetAppNote`