# Mercury ZX5 SoC Module

## Reference Design for Mercury+ ST1 Base Board User Manual

### Purpose

The purpose of this document is to present to the user the overall view of the Mercury ZX5 SoC module reference design and to provide the user with a step-by-step guide to the complete Xilinx® SoC design flow used for the Mercury ZX5 SoC module.

### Summary

This document first gives an overview of the Mercury ZX5 SoC module reference design and then guides through the complete Xilinx SoC design flow for the Mercury ZX5 SoC module in the getting started section. In addition, the internals and the boot options of the Mercury ZX5 SoC module reference design are described.

| Product Information | Code | Name |
|---|---|---|
| Product | ME-ZX5 | Mercury ZX5 SoC Module |

| Document Information | Reference | Version | Date |
|---|---|---|---|
| Reference / Version / Date | D-0000-445-002 | 1.0.0 | 25.08.2020 |

| Approval Information | Name | Position | Date |
|---|---|---|---|
| Written by | DDUE/ARUD | Design Engineer | 17.08.2020 |
| Verified by | GKOE | Design Expert | 18.08.2020 |
| Approved by | DIUN | Manager, BU SP | 25.08.2020 |

## License

## Table of Contents

# 1 Overview

## 1.1 Introduction

The Mercury ZX5 SoC module reference design demonstrates a system using the Mercury ZX5 SoC module in combination with the Mercury+ ST1 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.

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® 7000 All Programmable SoC Embedded Design Tutorial [2]

More information on the Mercury ZX5 SoC module and the Mercury+ ST1 base board can be retrieved from their respective user manuals [3] [4].

The following directory structure applies to the ZX5 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 ZX5 variant are released on the ZX5 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 ZX5 SoC Module User Manual [3] for details on device support in Xilinx tools)
  - Xilinx Vitis IDE
  - Enclustra Module Configuration Tool (MCT) [5] (optional[2])
  - A terminal emulation program (e.g. Tera Term)
- Hardware
  - An Enclustra Mercury ZX5 SoC module
  - An Enclustra Mercury+ ST1 base board
- Accessories
  - A 12 V DC power supply
  - A standard micro USB cable
  - A Xilinx JTAG programmer (e.g. Platform Cable USB II) (optional[3])

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

# 2 Reference Design Description

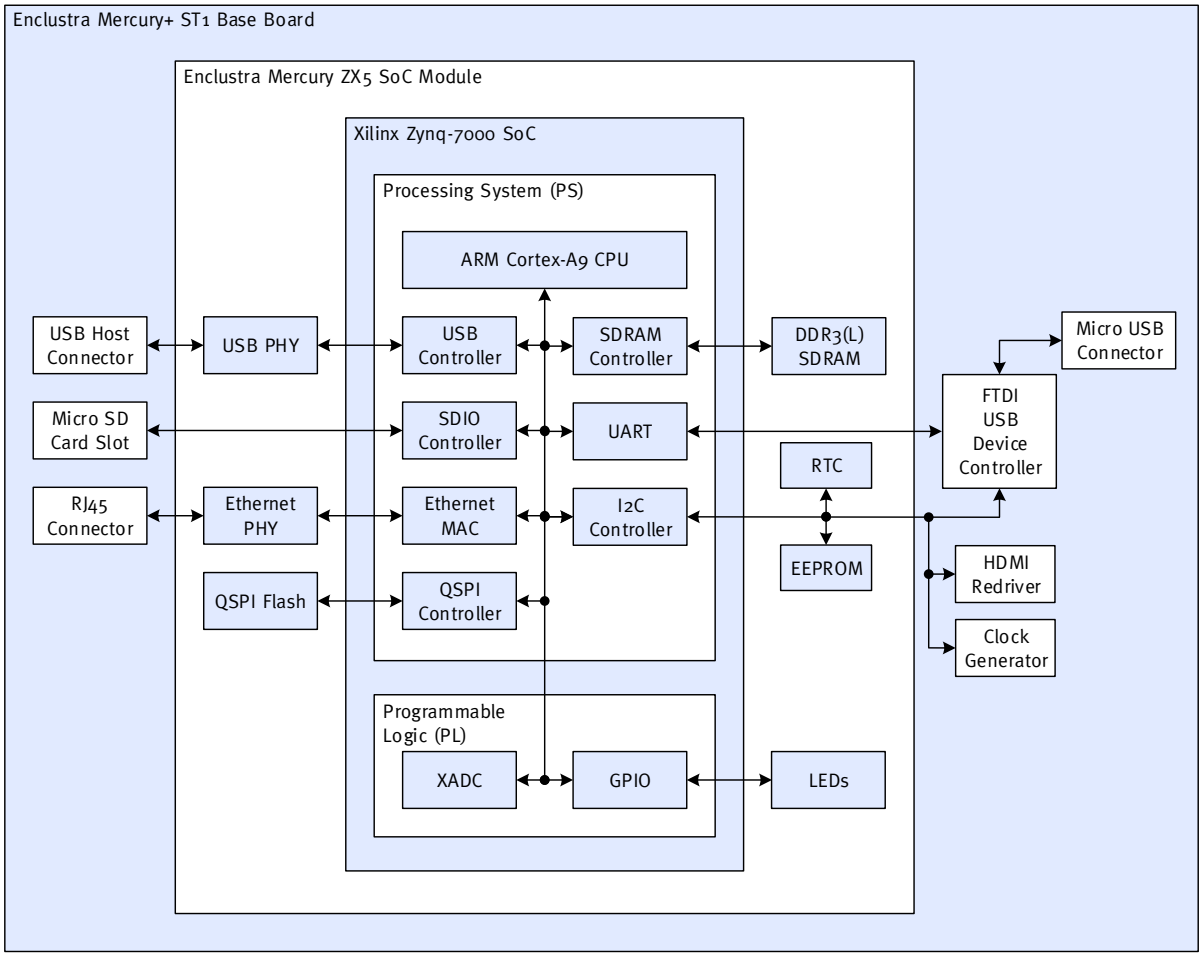


*Figure 1: Hardware Block Diagram*

## 2.1 Processing System (PS)

### 2.1.1 Clocks

The PS input clock frequency is configured to 33.33 MHz, while the CPU clock is configured to its corresponding maximum CPU frequency. The maximum CPU 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 DDR3L SDRAM

The DDR3L SDRAM memory runs at its corresponding maximum PS DDR frequency at a voltage of 1.35 V by default. The clock frequency for the controller can be modified in the Zynq system.

The DDR settings in the Zynq system must be configured according to the Mercury ZX5 SoC Module User Manual [3].

### 2.1.3 SD Card

The SD card is configured in the PS to the MIO 40..45 pins. This enables SD card access, as well as booting from the SD card.

To allow the Mercury ZX5 SoC module to boot from the SD card, the hardware configuration on the Mercury+ ST1 base board must be done according to Section 4.2.2.

### 2.1.4 I2C

The I2C controller I2C0 is configured to the EMIO pins. For available devices on the I2C bus refer to the Mercury ZX5 SoC Module and Mercury+ ST1 Base Board User Manual [3] [4]. An I2C Application Note is available as well providing sample code and more details about using I2C on Enclustra hardware [7].

### 2.1.5 Quad SPI Flash Controller

The quad SPI flash controller is connected to MIO 1..6 in Single Slave Select mode. MIO 2..6 pins are shared between NAND flash and QSPI flash on the Mercury ZX5 SoC module. Please refer to the Mercury ZX5 SoC Module User Manual [3] for details about flash programming and usage.

To allow the Mercury ZX5 SoC module to boot from the QSPI flash, the hardware configuration on the Mercury+ ST1 base board must be done according to Section 4.1.1.

### 2.1.6 UART

The UART0 is mapped to MIO 46..47 pins and connected to the FTDI controller on the Mercury+ ST1 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*

### 2.1.7 Ethernet

The Ethernet MAC ENET 0 is mapped to MIO 16..27 pins and is connected to the Microchip (Micrel) KSZ9031 Ethernet PHY on the Mercury ZX5 SoC module using an RGMII interface. The PHY can be configured via the MDIO management interface on PHY address 3. Please note that the RGMII delays in the Ethernet PHY need to be configured before the Ethernet interface can be used. Further details on PHY delay configuration are given in the Gigabit Ethernet Application Note by Enclustra [8].

### 2.1.8 USB

The USB controller USB0 on MIO 28..39 pins is connected to a USB3320C USB 2.0 PHY. This interface can be configured for USB host, USB device and USB On-The-Go (OTG) operations.

Depending on the required USB mode, the settings in the system controller and the DIP switches on the Mercury+ ST1 base board must be configured correctly. Please refer to the Mercury+ ST1 Base Board User Manual [4] for details.

### 2.1.9 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 ZX5 SoC module User Manual [3]. Check the connectivity of the MIOs that provide user functionality with the Mercury+ ST1 base board User Manual [4]

## 2.2 Programmable Logic (PL)

### 2.2.1 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 FPGA_LED0# with a frequency of approximately 3 Hz.

| PL Pin | Signal | Function |
|---|---|---|
| H5 | FPGA_LED0# | Blinking LED counter MSB |

*Table 3: PL Firmware I/O Configuration*

### 2.2.2 XADC

A Xilinx XADC 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 ZX5 SoC module may consume more power than can be dissipated without additional cooling measures; always make sure the SoC 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 ZX5 SoC module can be found in the Enclustra Modules Heat Sink Application Note [9].

# 3 Getting Started

This section describes the steps required to configure the Mercury ZX5 SoC module and Mercury+ ST1 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+ ST1 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+ ST1 base board are aligned with the mounting holes of the Mercury ZX5 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 ZX5 SoC module to or from the Mercury+ ST1 base board while the Mercury+ ST1 base board is powered. Always remove or turn off the power supply before mounting or removing the Mercury ZX5 SoC module.*

**Warning!**

*Please read carefully the Mercury ZX5 SoC module and Mercury+ ST1 base board user manuals before proceeding.*

**Warning!**

*Depending on the user application, the Mercury ZX5 SoC module may consume more power than can be dissipated without additional cooling measures; always make sure the SoC 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 [5] is used for SoC 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

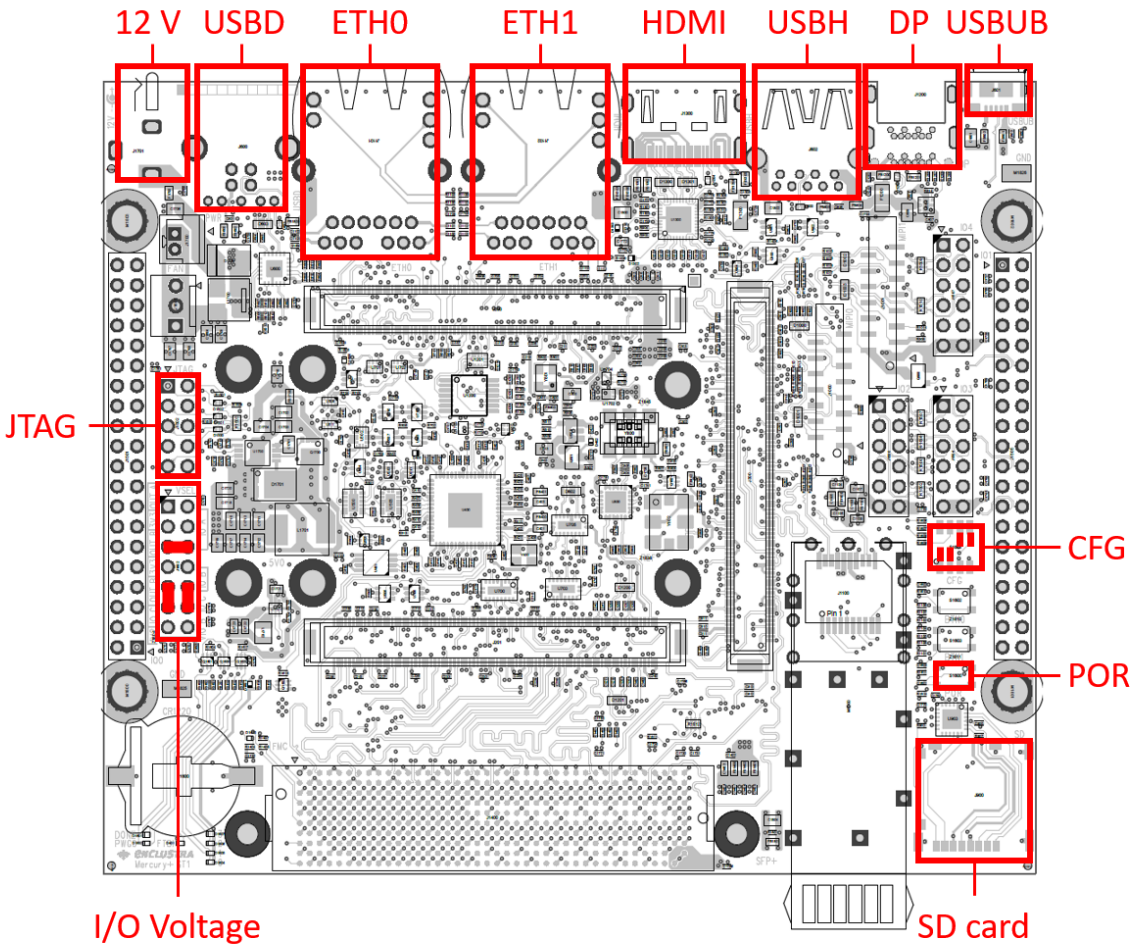


*Figure 2: Mercury+ ST1 Base Board Assembly Drawing (Top View)*

| Step | Description |
|---|---|
| 1 | Set the I/O voltage jumpers on the Mercury+ ST1 base board according to label **I/O Voltage** in Figure 2 (the jumpers are marked with red rectangles): |
| 2 | Set the configuration DIP switches on the Mercury+ ST1 base board as follows (see label **CFG** in Figure 2):<br>• CFG A = [1: OFF, 2: OFF, 3: ON, 4: ON] |
| 3 | Mount the Mercury ZX5 SoC module to the Mercury+ ST1 base board. Make sure that the mounting holes of the Mercury ZX5 SoC module are aligned with the mounting holes of the Mercury+ ST1 base board before proceeding. |
| 4 | Connect the micro USB cable between your computer and the Mercury+ ST1 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+ ST1 base board (see label **PWR** in Figure 2). |
| 6 | Make sure that the FTDI device on the Mercury+ ST1 base board is configured to Xilinx JTAG mode using Enclustra MCT [5] (or alternatively, connect the JTAG signals from the Xilinx Platform Cable USB to the JTAG connector of the Mercury+ ST1 base board (see label **JTAG Xilinx** in Figure 2).<br>Details on the Xilinx JTAG mode configuration and on the JTAG connector are presented in the Mercury+ ST1 Base Board User Manual [4] |
| 7 | 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.<br>For issues related to COM ports detection, refer to Section 5.4.<br>Configure the UART parameters according to Section 2.1.6. |

*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 ZX5 variant is released on the ZX5 Reference Design Github page.

| Step | Description |
|---|---|
| 1 | Configure the settings file:<br>1. Edit the `module_name` variable in `scripts/settings.tcl` file, according to your modules name.<br>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.<br>2. Save the file after editing. |

*Continued on next page...*

| Step | Description |
|---|---|
| 2 | Start Xilinx Vivado 2020.1 and create the Mercury ZX5 SoC module reference design project:<br>1. Click on the Tcl console at the bottom of the page and type:<br>(a) `cd {<base_dir>}`<br>where `<base_dir>` is the directory in which you extracted the archive contents. Note the `{}` around the path.<br>(b) `source ./scripts/create_project.tcl`<br>(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:<br>i. `vivado -mode batch -source ./scripts/create_project.tcl -tclargs <module_name>`.<br>ii. `open_project ./Vivado/<module_name>/<project_name>.xpr`<br>2. Wait for completion |
| 3 | Run Synthesis, Implementation & Bitstream Generation in Vivado 2020.1:<br>1. Click on Generate Bitstream from the Flow Navigator bar<br>2. In the Launch Runs window click OK - this will start automatically the entire implementation process<br>3. Wait for completion → select View Reports → OK |
| 4 | Export the hardware system information (required for the Vitis IDE):<br>1. File → Export → Export Hardware<br>2. Choose Fixed for platform type and click Next<br>3. Select Include Bitstream and click Next<br>4. Leave the file name and export location as default and click Next<br>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 ZX5 variant is released on the ZX5 Reference Design Github page.

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

*Continued on next page...*

| Step | Description |
|---|---|
| 2 | Create a new Platform Project<br><br>1. File → New → Platform Project<br>2. In the New Platform Project:<br>(a) For Project Name type the `<project_name>` e.g. Mercury_ZX5_ST1<br>(b) Check the "Use default location" checkbox, if it is not enabled yet.<br>(c) Hit Next<br>(d) Select "Create from hardware specification and hit Next<br>(e) Hit the Browse button and select the Hardware Specification .xsa file you exported from Vivado, as described in Section 3.3.<br>The default export location used by Vivado is `<base_dir>\<vivado_proj_dir>\<project_name>.xsa`.<br>Alternatively, the pre-compiled hardware description file may be used.<br>(f) Hit Next and wait for the file to be analyzed<br>(g) For the Operating System select standalone<br>(h) For the Processor select ps7_cortexa9_0<br>(i) Check the "Generate boot components" checkbox, if it is not enabled yet.<br>Vitis will then automatically generate the binaries for the FSBL.<br>(j) Hit Finish and wait for completion<br>(k) Build the platform by pressing Ctrl-B and wait for completion |
| 3 | Create a new application<br><br>1. File → New → Application Project<br>2. In the New Application Project window:<br>(a) For Project Name type a description for the new application e.g. "HelloWorld"<br>(b) Check the "Use default location" checkbox, if it is not enabled yet.<br>(c) For the System project select "Create New.." and use the default naming e.g. "HelloWorld_system"<br>(d) Hit Next<br>(e) In the "Select a platform from repository" tab choose the Platform that you have generated previously<br>(f) Hit Next<br>(g) For the Domain choose the standalone_domain and the C Language.<br>(h) Hit Next and wait for the tool to proceed<br>(i) Select the HelloWorld (or any another) template<br>(j) Hit Finish and wait for completion<br>(k) Build the application by pressing Ctrl-B and wait for completion |

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

## 3.5 Running Software Applications

This section describes how to run software applications on the Mercury ZX5 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:<br>1. Right click the previously generated application (e.g. HelloWorld) and select Run As → Run Configurations...<br>2. Right-click Single Application Debug and hit New or double-click on it<br>3. Enter a run configuration name in the Name field (e.g. HelloWorld)<br>4. Application tab:<br>(a) Enable ps7_cortexa9_0 checkbox<br>(b) In the Project Name field click browse and select an application (e.g. HelloWorld)<br>(c) In the Application field click search and select an .elf file (e.g. HelloWorld.elf)<br>(d) Enable Reset processor checkbox<br>(e) Hit Apply<br>5. Target Setup tab (see Figures 3 and 4):<br>(a) For Hardware Platform refer to the corresponding Platform: e.g. `${sdxTcfLaunchFile:project=HelloWorld;fileType=hw;}`<br>(b) For Bitstream file field, hit Search...<br>(c) Select Mercury_ZX5_ST1.bit and hit OK<br>(d) For FPGA Device and PS Device, use Auto Detect option<br>(e) Uncheck the "Use FSBL flow for initialisation"<br>(f) In the Initialization File field, hit Search...<br>(g) Select ps7_init.tcl and hit OK<br>(h) Enable checkboxes Reset entire system, Program FPGA, Run ps7_init and Run ps7_post_config<br>(i) Hit Apply |
| 2 | Make sure the Hardware is configured according to Section 3.2:<br>→ Connect the 12 V DC power supply plug to the power connector of the Mercury+ ST1 base board (see label **12 V DC** in Figure 2).<br>→ 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.<br>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.<br>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.<br>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.

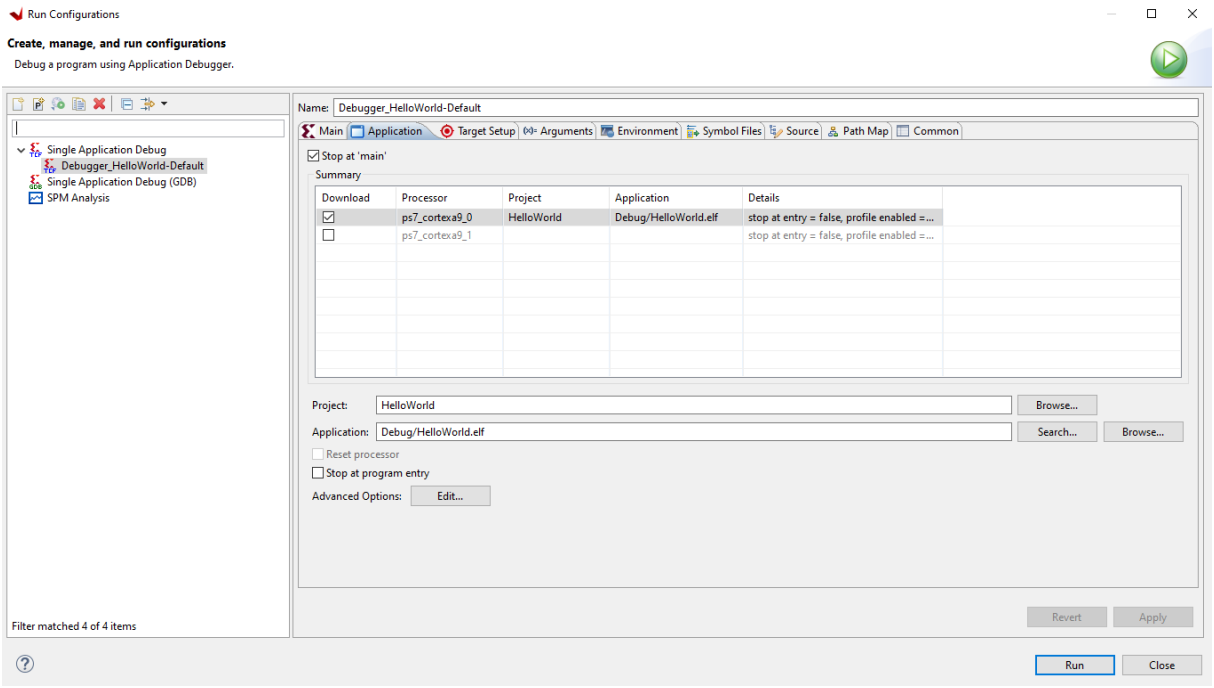


*Figure 3: Run Configurations Settings - Application Tab*

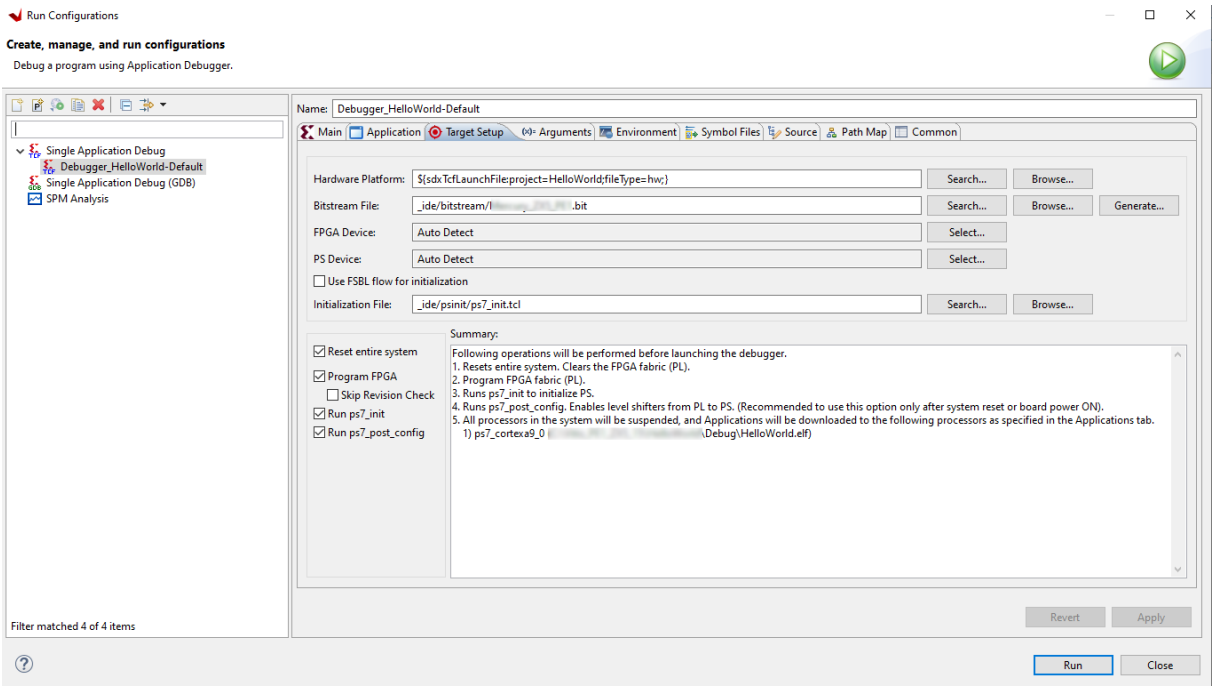


*Figure 4: Run Configurations Settings - Target Setup Tab*

# 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 ZX5 SoC module variant. Pre-generated binaries for any ZX5 variant are released on the ZX5 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 5):<br>1. Right click on the application in the Project Explorer<br>2. Select Create Boot Image → Create Image<br><br>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 [6].

## 4.1 QSPI Flash Boot

### 4.1.1 Preparing the Hardware

| Step | Description |
|---|---|
| 1 | Disconnect the power supply of the Mercury+ ST1 base board(see label **12 V** in Figure 2). |
| 2 | Disconnect all USB cables from the Mercury+ ST1 base board. |
| 3 | Set the configuration DIP switches on the Mercury+ ST1 base board as follows to enable JTAG boot mode (see label **CFG** in Figure 2):<br>• CFG A = [1: ON, 2: ON, 3: ON, 4: ON] |
| 4 | Connect the micro USB cable between your computer and the Mercury+ ST1 base board. Use the micro USB port labeled **USBUB** in Figure 2. |

*Continued on next page...*

| Step | Description |
|---|---|
| 5 | Reconnect the power supply of the Mercury+ ST1 base board(see label **12 V** 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 5):<br>1. Right click on the application in the Project Explorer<br>2. Select Program Flash<br>Vitis will fill out the fields for the selected application automatically.<br>3. For Flash Type select qspi-x4-single<br>4. Hit Program and wait for completion<br><br>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.<br>1. Flow → Open Hardware Manager<br>2. Click on Open target → Auto Connect<br>3. Right click on the corresponding SoC device in the left bar → Add Configuration Memory Device (see Figure 6)<br>(a) For Select Configuration Memory Part choose the memory part according to the Mercury ZX5 SoC Module User Manual [3], part type single.<br>This is in most cases s25fl512s-1.8v-qspi-x4-single.<br>(b) Hit OK<br>4. In Program Configuration Memory Device window (see Figure 7):<br>(a) For Configuration file select the boot image generated as described in Section 4.0.1<br>(b) For Zynq FSBL select the FSBL binary generated with the Platform as described in Section 3.4<br>(c) In Program Operations section:<br>• For Address Range select Entire Configuration Memory Device<br>• Enable checkboxes Erase, Program and Verify<br>• Hit OK and wait for completion<br><br>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) [5] can be used to program the QSPI flash.<br>The procedure implies setting another boot mode than QSPI during flash programming, so that the SoC 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.<br>1. Close all other tools that may be connected to the FTDI device (Vivado Hardware Manager, Vitis, UART terminal).<br>2. Disconnect the power supply of the Mercury+ ST1 base board(see label **12 V** in Figure 2).<br>3. Disconnect all USB cables from the Mercury+ ST1 base board.<br>4. Set CFG = [1: ON, 2: OFF, 3: ON, 4: ON]<br>5. Connect a USB cable to the micro USB port on the Mercury+ ST1 base board (see label **USBUB** in Figure 2)<br>6. Reconnect the power supply of the Mercury+ ST1 base board(see label **12 V** in Figure 2).<br>7. Perform QSPI flash programming in MCT and close MCT<br>8. After programming, remove the power supply from the Mercury+ ST1 base board (see label **12 V** in Figure 2).<br>9. Disconnect all USB and power supply cables from the Mercury+ ST1 base board.<br>10. 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*

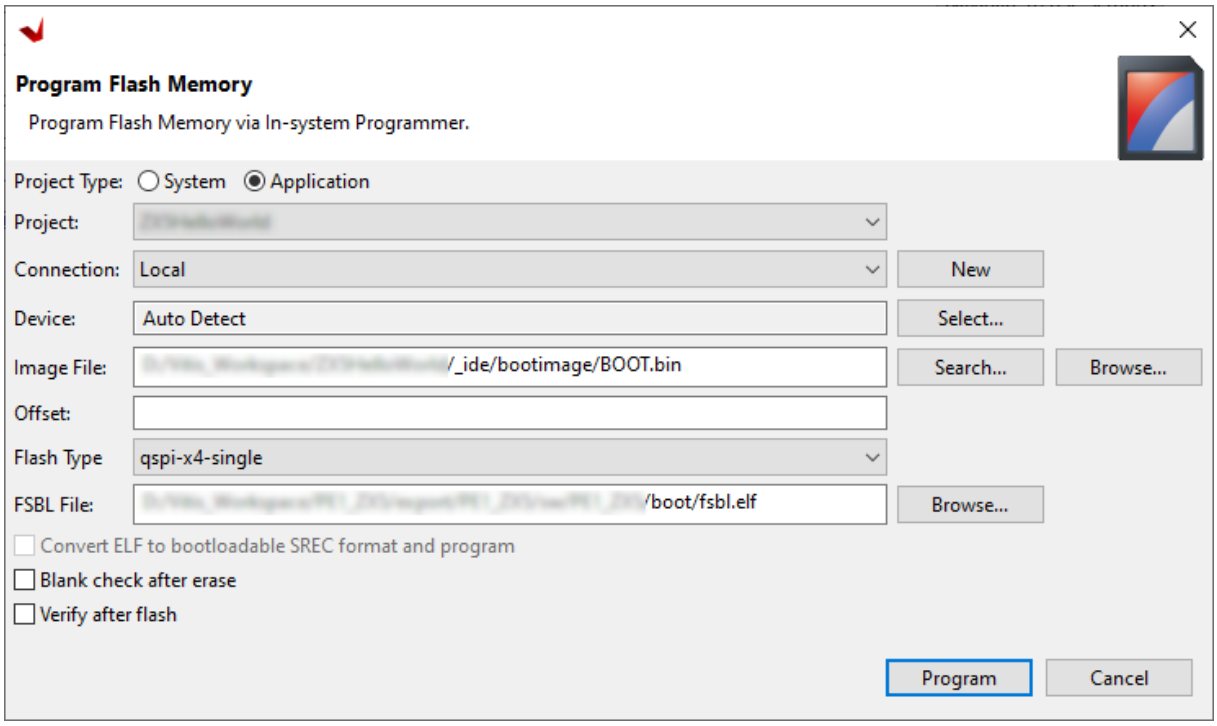


*Figure 5: QSPI Flash Programming Settings in Vitis*

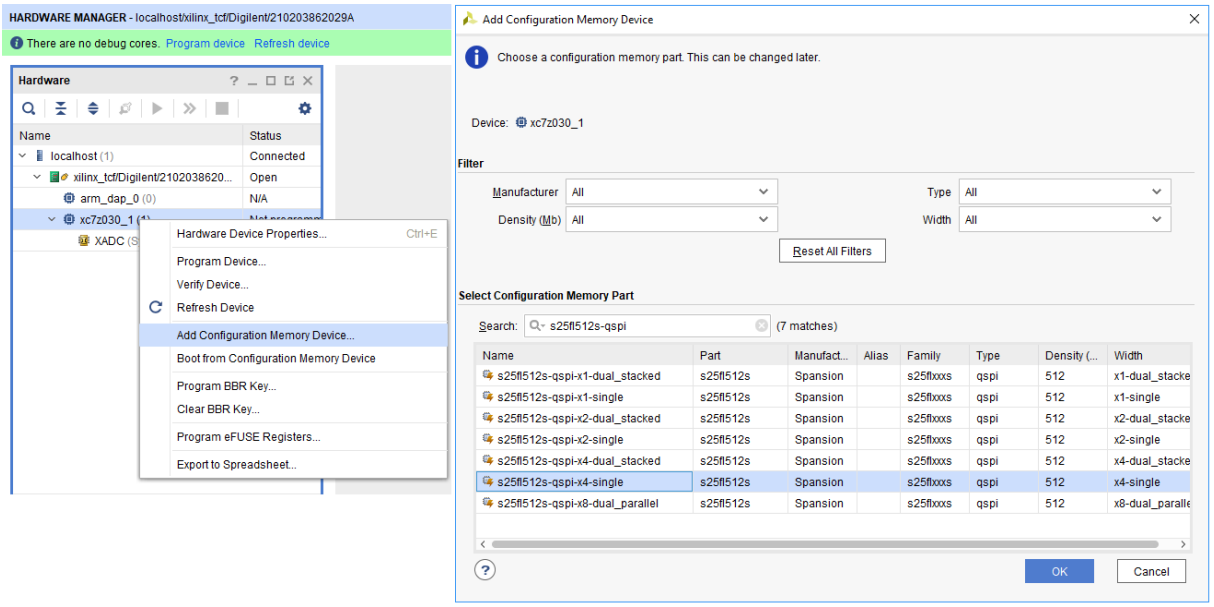


*Figure 6: QSPI Flash Programming Settings in Vivado - Adding the Memory Device*

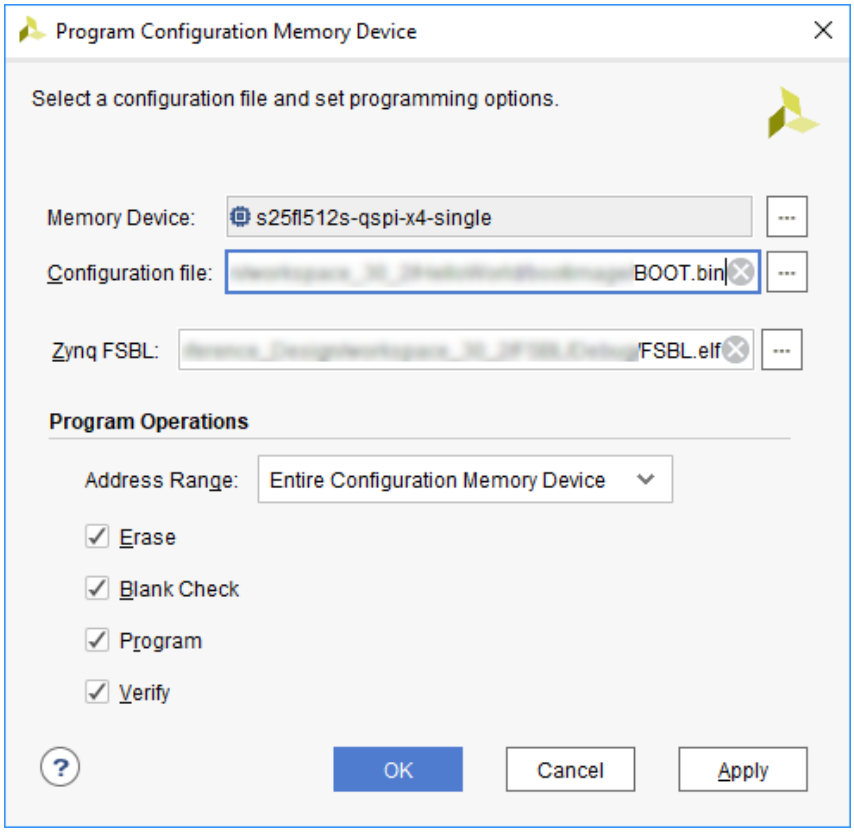


*Figure 7: QSPI Flash Programming Settings in Vivado*

**Warning!**

*Some Vivado and Vitis tool versions are reporting problems when configuring certain SoC 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 QSPI Flash hardware setup

| Step | Description |
|---|---|
| 1 | Disconnect the power supply of the Mercury+ ST1 base board(see label **12 V** in Figure 2). |
| 2 | Disconnect all USB cables from the Mercury+ ST1 base board. |
| 3 | Set the configuration DIP switches on the Mercury+ ST1 base board as follows to enable QSPI boot mode (see label **CFG** in Figure 2):<br>• CFG A = [1: ON, 2: OFF, 3: ON, 4: ON] |
| 4 | Connect the micro USB cable between your computer and the Mercury+ ST1 base board. Use the micro USB port labeled **USBUB** in Figure 2. |
| 5 | Reconnect the power supply of the Mercury+ ST1 base board(see label **12 V** in Figure 2). |

*Table 11: Booting from QSPI Flash Boot Mode hardware setup Step-by-Step Guide*

### 4.1.4 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 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 4.0.1 describing the steps required to generate a boot image.

### 4.2.2 Preparing the Hardware

| Step | Description |
|---|---|
| 1 | Disconnect the power supply of the Mercury+ ST1 base board(see label **PWR** in Figure 2). |

*Continued on next page...*

| Step | Description |
|---|---|
| 2 | Enable the SD card boot mode (default) by setting the configuration DIP switches on the Mercury+ ST1 base board as follows (see label **CFG** in Figure 2):<br>• CFG = [1: OFF, 2: OFF, 3: ON, 4: ON] |

*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 a FAT32 formatted SD card<br>1. Insert the SD card into the SD card slot of your computer<br>2. Copy the boot image generated for your application to your SD card (directly in the root directory).<br>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+ ST1 base board (see label **SD Card** in Figure 2). |
| 2 | Connect the power supply to the Mercury+ ST1 base board(see label **12 V** 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
- During block design generation Vivado reports a critical warning (CRITICAL WARNING: [PSU-1] Parameter : PCW_UIPARAM_DDR_DQS_TO_CLK_DELAY_0 has negative value -0.026 . PS DDR interfaces might fail when entering negative DQS skew values.) which can be safely ignored for ZYNQ 7000 PS DDR interfaces. For more details please refer to this Xilinx forum post.

## 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 [5]. More information on the Xilinx JTAG mode configuration on the Mercury+ ST1 base board can be retrieved from the Mercury+ ST1 base board user manual [4].
  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+ ST1 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+ ST1 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. If the UART used is mapped to the EMIO pins in the PS, resetting the ARM core will not suffice. Reprogramming the PL is necessary, as the UART lines go through PL.
  4. 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 ZX5 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.

- The program flash operation might fail. Another possible workaround is to add a modification to the standard FSBL generated during platform creation (besides the already mentioned workarounds in 4.1.2). The created FSBL is limited to only running the initialization (ps7_init()). For more details on this please refer to the answer record by Xilinx AR70548.

| Step | Description |
|---|---|
| 1 | Modify the created Platform project<br>1. In the Platform project open platform.scr by double-clicking<br>2. Click on "Board Support Settings" under the standalone ps7_cortexa9_0 tab<br>3. Click on "Modify BSP Settings..." in the right hand window<br>4. Enable the xilffs library (See figure 8) |

*Continued on next page...*

| Step | Description |
|---|---|
| 2 | Create a new application<br>1. File → New → Application Project<br>2. In the New Application Project window:<br>(a) For Project Name type FSBL<br>(b) Check the "Use default location" checkbox, if it is not enabled yet.<br>(c) For the System project select "Create New.." and use the default naming "FSBL_system"<br>(d) Hit Next<br>(e) In the "Select a platform from repository" tab choose the Platform that you have generated previously<br>(f) Hit Next<br>(g) For the Domain choose the standalone_domain and the C Language.<br>(h) Hit Next and wait for the tool to proceed<br>(i) Select the Zynq FSBL template<br>(j) Hit Finish and wait for completion<br>(k) In the created FSBL project folder modify main.c located in the src folder with the code snippet provided below this table (1).<br>(l) Build the application by pressing Ctrl-B and wait for completion |

*Table 16: Creating modified FSBL for QSPI programming Mode Step-by-Step Guide*

*Code snippet 1: Modification to FSBL*

```
/*
* Read bootmode register
*/
BootModeRegister = Xil_In32(BOOT_MODE_REG);
BootModeRegister &= BOOT_MODES_MASK;

//add this line to trick boot mode to JTAG
BootModeRegister = JTAG_MODE;
```

After the modified FSBL is ready, proceed with actually programming the QSPI Flash.

| Step | Description |
|---|---|
| 1 | Program the boot image from Xilinx Vitis 2020.1 (see Figure 5):<br>1. Right click on the application in the Project Explorer<br>2. Select Program Flash<br>Vitis will fill out the fields for the selected application automatically.<br>3. Replace the automatically found FSBL with the FSBL.elf from the generated FSBL project<br>4. For Flash Type select qspi-x4-single<br>5. Hit Program and wait for completion<br><br>The settings in the pictures are for reference only. Note that the configuration file must be selected according to your application. |

*Table 17: Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide*

| | Name | Version | Description |
|---|---|---|---|
| ☐ | libmetal | 2.0 | Libmetal Library |
| ☐ | lwip211 | 1.1 | lwip211 library: lwIP (light weight IP) is an open sour... |
| ☐ | openamp | 1.5 | OpenAmp Library |
| ☑ | xilffs | 4.2 | Generic Fat File System Library |
| ☐ | xilflash | 4.7 | Xilinx Flash library for Intel/AMD CFI compliant paral... |
| ☐ | xilisf | 5.14 | Xilinx In-system and Serial Flash LibraryWARNING: X... |
| ☐ | xilloader | 1.0 | Xilinx Versal Platform Loader Library |
| ☐ | xilplmi | 1.0 | Xilinx versal Platform Loader and Manager Interface ... |
| ☐ | xilpm | 3.0 | Platform Management API Library for ZynqMP and ... |
| ☐ | xilrsa | 1.5 | Xilinx RSA Library to access RSA and SHA software al... |
| ☐ | xilsem | 1.0 | Xilinx Versal Soft Error Mitigation Library |
| ☐ | xilskey | 6.8 | Xilinx Secure Key Library supports programming efu... |

*Figure 8: Enable xilffs in Vitis*

## List of Figures



## List of Tables



## References


[1] Vivado Design Suite User Guide, Embedded Processor Hardware Design, UG898, Xilinx, 2019
[2] Zynq-7000 All Programmable SoC Embedded Design Tutorial, UG1165, Xilinx, 2015
[3] Mercury ZX5 SoC Module User Manual
→ Ask Enclustra for details
[4] Mercury+ ST1 Base Board User Manual
→ Ask Enclustra for details
[5] Enclustra Module Configuration Tool (MCT)
→ Ask Enclustra for details
[6] Enclustra Build Environment
`https://github.com/enclustra-bsp/bsp-Xilinx`
[7] Enclustra I2C Application Note
→ Ask Enclustra for details
[8] Enclustra Gigabit Ethernet Application Note
→ Ask Enclustra for details
[9] Enclustra Modules Heat Sink Application Note
→ Ask Enclustra for details