

Mercury+ KX2 FPGA Module

Reference Design for Mercury+ PE3 Base Board User Manual

Purpose

The purpose of this document is to present to the user the overall view of the Mercury+ KX2 FPGA module reference design and to provide the user with a step-by-step guide to the complete AMD® FPGA design flow used for the Mercury+ KX2 FPGA module.

Summary

This document gives an overview of the Mercury+ KX2 FPGA module reference design and then guides through the complete AMD FPGA design flow for the Mercury+ KX2 FPGA module in the getting started section. In addition, the internals and the boot options of the Mercury+ KX2 FPGA module reference design are described.

Product Information	Code	Name
Module	ME-KX2	Mercury+ KX2 FPGA Module
Baseboard	ME-PE3	Mercury+ PE3 Base Board

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-489-006	2024.1_v1.0.1	08.11.2024

Approval Information	Name	Position	Date
Written by	ARUD	FPGA/SoC Senior Embedded Software Engineer	08.11.2024
Verified by	GKOE	Manager, FPGA/SoC Embedded Software Engineering	08.11.2024
Approved by	IJOS	Manager, BU SP	08.11.2024

License

Copyright 2024 by Enclustra GmbH, Switzerland.

Permission is hereby granted, free of charge, to any person obtaining a copy of this hardware, software, firmware, and associated documentation files (the "Product"), to deal in the Product without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Product, and to permit persons to whom the Product is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Product.

THE PRODUCT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE PRODUCT OR THE USE OR OTHER DEALINGS IN THE PRODUCT.

Table of Contents

Table of Contents	3
1 Overview	4
1.1 Introduction	4
1.2 Variables Paths	4
1.3 Prerequisites	5
2 Reference Design Description	6
2.1 Programmable Logic (PL)	7
2.2 Microblaze System	7
2.2.1 Clocks	7
2.2.2 Microblaze CPU	7
2.2.3 DDR3 SDRAM	7
2.2.4 I2C	7
2.2.5 Microblaze Debug Module (MDM)	7
2.2.6 Quad SPI Flash Controller	7
2.2.7 UART	7
2.2.8 Ethernet	8
2.2.9 GPIOs	8
2.2.10 System Management	9
3 Getting Started	10
3.1 Essential Information	10
3.2 Setting up the Hardware	10
3.3 Generating the FPGA Bitstream	12
3.4 Preparing the Boot Binaries	13
3.4.1 Preparing the Vitis Workspace	13
3.4.2 Creating the Hardware Platform	13
3.4.3 Building the HelloWorld Application	14
3.4.4 Running Software Applications	14
4 Boot Configurations	15
4.1 Generating the Image File	15
4.2 QSPI Flash Boot	15
4.2.1 Programming the QSPI Flash	15
4.2.2 Preparing the Hardware	17
4.2.3 Booting from the QSPI Flash	18
5 Troubleshooting	19
5.1 Vivado Issues	19
5.2 Vitis Issues	19
5.3 JTAG Issues	19
5.4 UART Connection Issues	19
5.5 QSPI Issues	20
5.6 MCT Issues	20
List of Figures	21
List of Tables	21
References	22

1 Overview

1.1 Introduction

The Mercury+ KX2 FPGA module reference design demonstrates a system using the Mercury+ KX2 FPGA module in combination with the Mercury+ PE3 base board. It presents the basic configuration of the device and contains a getting started tutorial.

A troubleshooting section is included at the end of the document to help users resolve potential issues related to board connectivity and system functionality.

This reference design does not include any source code for software examples. Instead, Enclustra provides application notes [6] for selected applications.

An introduction to the AMD tools is available online:

- Vivado Design Suite User Guide [9]
- Vitis Unified Software Platform Documentation [11]
- Embedded Design Tutorials [12]
- Microblaze Processor Embedded Design Tutorial [13]

More information on the module and the base board can be found in the Mercury+ KX2 FPGA Module User Manual [2] and the Mercury+ PE3 Base Board User Manual [3]. The following directory structure applies to the reference design:

Directory	Description
doc	Reference design documentation
scripts	Scripts directory required for project creation and settings
src	Design pinout, timing constraints and VHDL source code directory

Table 1: Directory Structure

1.2 Variables Paths

The following variables are used to substitute the full path to directories relevant for the reference design build process. Paths not already present in the reference design directory will be created during the build process.

Variable	Full Path	Description
<base_dir>	<base_dir>	Reference design root directory as described in Table 1
<vivado_dir>	<base_dir>/Vivado/<module_name>	Vivado project directory

Table 2: Path Variables

1.3 Prerequisites

- IT infrastructure
 - Computer
 - Supported OS¹
- Software
 - AMD Vitis 2024.1 Core Development Kit
 - Enclustra Module Configuration Tool (MCT) [\[4\]](#) (optional²)
 - a terminal emulation program, for example, Tera Term
- Hardware
 - Enclustra Mercury+ KX2 FPGA module
 - Enclustra Mercury+ PE3 base board
- Accessories
 - 12 V DC power supply
 - Micro-USB cable

¹A comprehensive list of supported operating systems is given in the Vivado Design Suite Installation Guide [\[10\]](#).

²May be used for flash programming, for FPGA device configuration or for FTDI configuration.

2 Reference Design Description

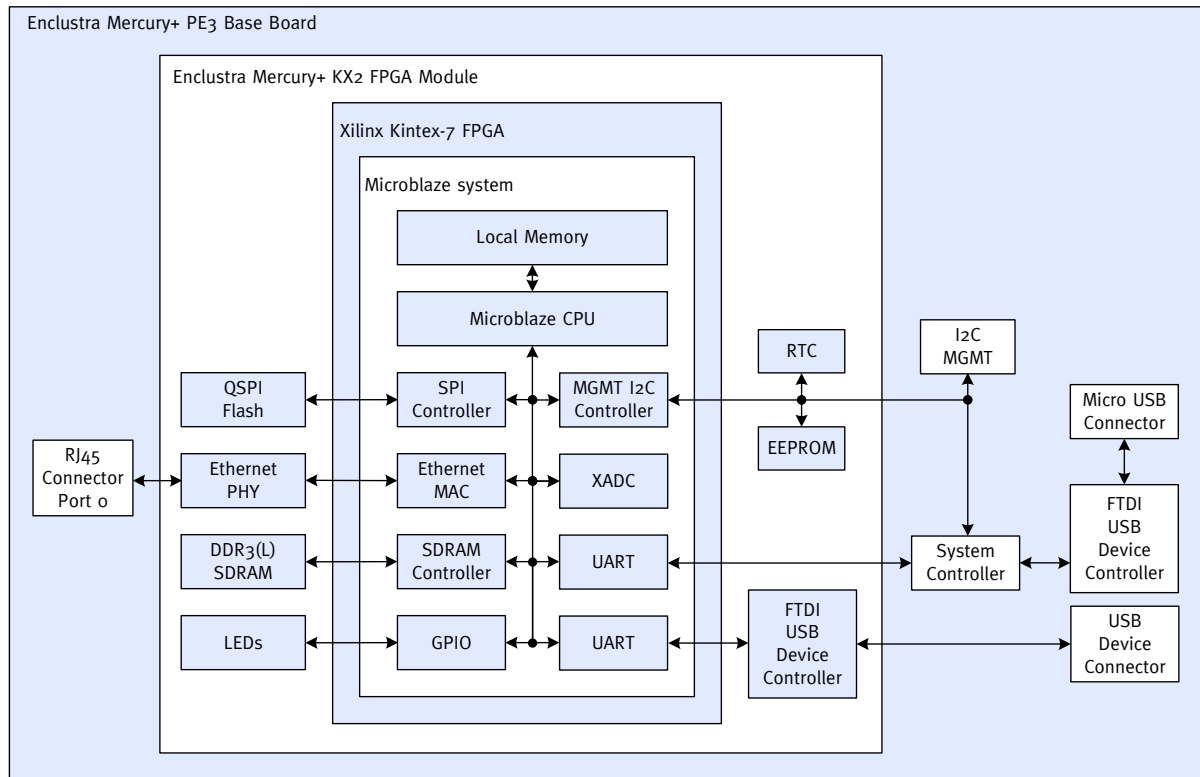


Figure 1: Hardware Block Diagram

2.1 Programmable Logic (PL)

2.2 Microblaze System

2.2.1 Clocks

The system clock is provided by a 50 MHz oscillator on the module. From this system clock the Memory Interface Generator (MIG) IP core generates the 100 MHz input clock for the Microblaze system. A 50 MHz clock, a 100 MHz clock, and a 200 MHz clock are exported from the block design to the top-level HDL file. These clocks can be modified in the settings of the MIG IP core in the Vivado block design.

2.2.2 Microblaze CPU

The Microblaze CPU has access to all peripheral devices via the AXI bus interconnect. The processor has an internal memory of 64 kB, 8 kB instruction cache and 8 kB data cache for external memory access. The size of the internal memory can be modified in the **Address Editor** tab in the Vivado block design.

2.2.3 DDR3 SDRAM

The DDR3 SDRAM memory is configured as specified in the Mercury+ KX2 FPGA Module User Manual [2]. The voltage is set to 1.5 V. The DDR3_VSEL signal can be used to control the DDR3 voltage. The design constraints need to be adjusted accordingly. The DDR configuration can be modified in the MIG IP core in the Vivado block design.

2.2.4 I2C

An AXI IIC controller is connected to the I2C MGMT bus on the base board. A second AXI IIC controller is connected to the I2C USER bus. The available I2C devices on each bus are listed in the Mercury+ KX2 FPGA Module User Manual [2] and the Mercury+ PE3 Base Board User Manual [3].

2.2.5 Microblaze Debug Module (MDM)

The MDM offers additional debugging features and provides a JTAG UART interface.

Tip

For JTAG UART usage, **stdin** and **stdout** must be set to the MDM instance in the board support package settings in Vitis.

2.2.6 Quad SPI Flash Controller

The AXI quad SPI flash controller is connected to the QSPI flash on the Mercury+ KX2 FPGA module. The configuration DIP switches on the Mercury+ PE3 base board must be set according to Section 4.2.2 in order to boot from the QSPI flash. The QSPI flash is connected to FPGA configuration pins and requires the **STARTUPE2** primitive to drive the clock. The SPI clock is set to high impedance in the top-level HDL file. The **Enable STARTUPE2 Primitive** checkbox is enabled in the IP core settings in Vivado.

2.2.7 UART

The AXI UART controller settings are given in Table 3. One AXI UART controller is connected to the PE3 FTDI chip while the other AXI UART controller is connected to the KX2 FTDI chip.

Parameter	Value
Baud rate	115200
Data	8 bit
Parity	None
Stop	1 bit
Flow control	None

Table 3: UART Configuration

2.2.8 Ethernet

The AXI Ethernet MAC is connected to a Microchip KSZ9031 RGMII Ethernet PHY on the Mercury+ KX2 FPGA module.

Tip

Bitstream generation will fail without valid licenses (permanent or evaluation) for the AXI Ethernet Sub-system IP cores. To exclude the Ethernet MAC from the block design, remove the **-ETH** flag from the `settings.tcl` file.

The PHYs can be configured via the MDIO interface on address 3 (Ethernet 0) and address 7 (Ethernet 1)³.

Tip

The RGMII delays need to be configured before the Ethernet interface can be used. Further details on PHY delay configuration are given in the Enclustra Gigabit Ethernet Application Note [6].

2.2.9 GPIOs

The FPGA firmware contains a 24-bit counter freely running at 50 MHz. The MSB of this counter is used to blink LED0# with a frequency of approximately 3 Hz. The pin mapping is given in Table 4.

Pin	Signal	Function
U9	LED0#	Blinking LED counter MSB

Table 4: FPGA Blinking LED Configuration

In addition, the block design instantiates an AXI GPIO IP block to control the LEDs described in Table 5.

Pin	Signal	Function
V12	LED1#	GPIO 0
V13	LED2#	GPIO 1
W13	LED3#	GPIO 2

Table 5: FPGA AXI GPIO Configuration

³The reference design only uses Ethernet 0.

2.2.10 System Management

An XADC IP core instance is connected via AXI bus to the Microblaze CPU 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.

3 Getting Started

This section describes the steps required to configure the Mercury+ KX2 FPGA module and the Mercury+ PE3 base board in order to run a simple HelloWorld application example:

1. Install the module and configure the base board.
2. Generate the bitstream.
3. Prepare the software workspace.
4. Run a software application.

Read the Mercury+ KX2 FPGA Module User Manual [2] and the Mercury+ PE3 Base Board User Manual [3] carefully before proceeding.

3.1 Essential Information

Pre-generated binaries may be used instead of building them manually as described in the following sections. The binaries for any supported KX2 product model and boot mode are released on the reference design Github release page [5].

Tip

Workarounds and fixes for potential issues can be found in Section 5.

NOTICE



Damage to the device due to overheating

Depending on the user application, the Mercury+ KX2 FPGA module may consume more power than can be dissipated without additional cooling measures.

- Ensure that the FPGA is always adequately cooled by installing a heat sink and/or providing air flow.

3.2 Setting up the Hardware

NOTICE



Damage to the device when mounting or removing the module

Mounting or removing the module while the base board is powered can lead to damage to the module or the base board.

- Ensure that the base board is not powered before mounting or removing the module.

The assembly drawing of the Mercury+ PE3 base board is shown in Figure 2. The relevant interfaces are marked in red in the figure as well as in the instructions below.

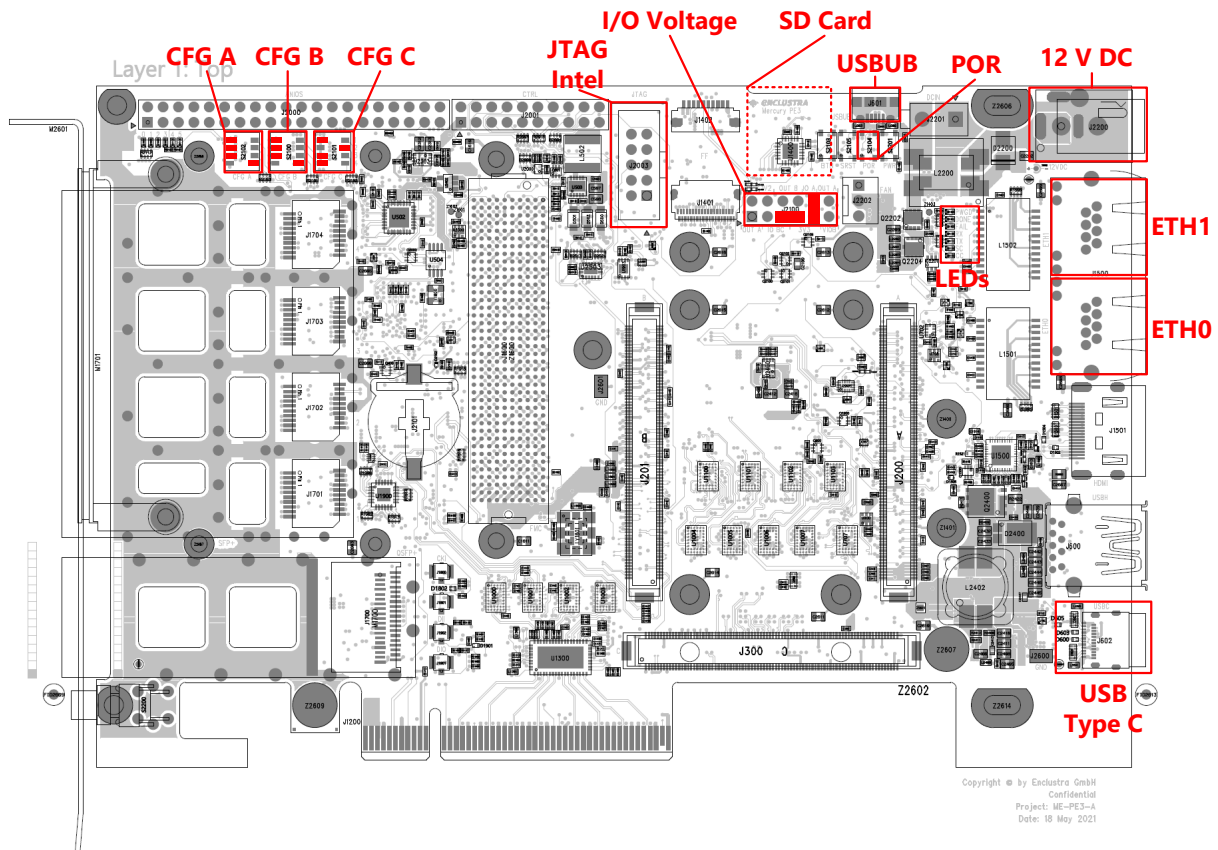


Figure 2: Mercury+ PE3 Base Board Assembly Drawing (Top View)

1. Ensure that no power is connected to the power connector of the base board (label **12 V DC**).
2. Set the I/O voltage jumpers on the base board according to label **I/O Voltage** (the jumpers are marked with red rectangles):
 - VSEL A = 3.3 V (position 9-10)
 - VSEL BC = 3.3 V (position 5-7)
3. Set the configuration DIP switches on the base board as follows (labels **CFG A**, **CFG B** and **CFG C**):
 - CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON]
 - CFG B = [1: OFF, 2: OFF, 3: OFF, 4: ON]
 - CFG C = [1: OFF, 2: ON, 3: OFF, 4: OFF]
4. Install the Mercury+ KX2 FPGA module on the base board.

Tip

A small golden square on the bottom left corner of Enclustra modules is provided as landmark. The same landmark is provided on the Enclustra base boards to help orient the module in the right direction when connecting it.

5. Connect the Micro-USB cable between the computer and the base board using the Micro-USB port labeled **USBUB**. The SC LED starts blinking (label **LEDs**).
6. **NOTICE: Damage to the device when applying power. Ensure that the mounting holes on the base board are aligned with the mounting holes of the module before applying power.** Connect a 12 V DC power supply plug to the power connector of the base board (label **12 V DC**). The PWGD LED lights up (label **LEDs**).
7. Configure the FTDI in Xilinx JTAG mode using the Enclustra MCT [4].
 - (a) Open MCT.
 - (b) Click **Enumerate**.

- (c) Select the detected device.
- (d) Select **FTDI Configuration**.
- (e) Select **Xilinx JTAG** as the device mode.
- (f) Click **Set device mode**.
- (g) Wait for completion.
- (h) Close MCT.
- (i) Disconnect and reconnect the Micro-USB cable.

Tip

Alternatively, an AMD JTAG USB programmer can be used. Connect the JTAG signals to the JTAG connector on the base board.

- 8. Two new serial ports should be detected by the OS.
- 9. Open the serial port with the highest number of the two detected ports and configure it using the parameters specified in Section 2.2.7. Any suitable utility for establishing a serial connection can be used. The output of the HelloWorld application will be printed in this terminal after running a software application, as presented in Section 3.4.4.

3.3 Generating the FPGA Bitstream

Follow the steps described in this section to generate the reference design bitstream manually. Refer to the troubleshooting section 5.1 in case of problems.

Tip

Pre-generated bitstreams for all supported KX2 product models are available on the KX2 reference design Github release page [5]. These can be used directly instead of generating the bitstream manually. If using those files directly, the bitstream does not need to be generated manually and this section can be skipped.

1. Configure the `<base_dir>/scripts/settings.tcl` file:
 - (a) Set the `module_name` variable to the desired product model⁴.
 - (b) Save the file after editing⁵.
2. Start AMD Vivado 2024.1.
3. Create the Mercury+ KX2 FPGA module reference design project:
 - (a) Click on the **Tcl Console** at the bottom of the Vivado window.
 - i. Type `cd {<base_dir>}`
Note the curly brackets around the path.
 - ii. Type `source ./scripts/create_project.tcl`
 - iii. Wait for completion (the created project opens automatically).
4. Compile the design and generate the programming files:
 - (a) Click on **Generate Bitstream**.
 - (b) Click **Yes** in the next window.
 - (c) Click **Launch runs on local host** with the default number of jobs. This will run all design steps from synthesis to implementation and bitstream generation.
 - (d) Wait for completion.
 - (e) Select **View Reports** and click **OK**.
5. Export the hardware platform:
 - (a) Select **File > Export > Export Hardware...**
 - (b) Click **Next**.
 - (c) Select **Include bitstream** and click **Next**.

⁴Valid values for the variables are listed at the beginning of the file.

⁵The Vivado project directory can be set to a custom value by adjusting the `<vivado_dir>` variable. However, the documentation assumes that the default path is used.

- (d) Leave the default settings and click **Next**.
- (e) Click **Finish**.

The build output is located in `<base_dir>/Vivado/<module_name>` and is described in Table 6:

File	Description
Mercury_KX2_PE3.xsa	Handoff file for the Vitis IDE
Mercury_KX2_PE3.runs/impl_1/Mercury_KX2_PE3.bit	Generated bitstream
Mercury_KX2_PE3.runs/impl_1/Mercury_KX2_PE3.mmi	Memory map file

Table 6: Vivado Programming Files

3.4 Preparing the Boot Binaries

This section describes the whole build flow for generating a bootable binary image file. The AMD Vitis IDE 2024.1 is used in the following sections.

Tip

Pre-generated binaries for all supported KX2 product models are available on the reference design Github release page [5]. These can be used directly instead of generating the binaries manually. In this case, the following section can be skipped.

3.4.1 Preparing the Vitis Workspace

Follow these steps to create a new workspace:

1. Create an empty folder for the workspace.
2. Open the AMD Vitis IDE 2024.1:
 - (a) Click **Open Workspace**.
 - (b) Select the previously created empty folder for the Vitis workspace and click **OK**.

The workspace setup for the HelloWorld application is complete.

3.4.2 Creating the Hardware Platform

Add a new hardware platform to the workspace created in Section 3.4.1:

1. In the **Welcome** splash screen, select **Create Platform Component**.
2. Use `Mercury_KX2_PE3` as the **Component name**.
3. Select the workspace folder as the **Component location**.
4. Click **Next**.
5. Choose **Hardware Design**.
6. Click **Browse** and select the XSA file generated in Section 3.3.
7. Click **Next**.
8. Wait for completion on the next screen. **Operating system** and **Processor** should be filled automatically.
9. Click **Next**.
10. In the summary screen, click **Finish**.
11. Wait until Vitis has completed all tasks and the created platform is available in the **Vitis Components** window.
12. In the **Flow** window, click **Build**.
13. Wait for completion.

The hardware platform for running bare-metal applications on the Mercury+ KX2 FPGA module is ready.

3.4.3 Building the HelloWorld Application

Create a new application from a template and build it by following these steps:

1. Select **View > Examples > Hello World**.
2. Click **Create Application Component from Template**.
3. Select `HelloWorld` for the **Component name** and click **Next**.
4. Select the platform created in Section 3.4.2 and click **Next**.
5. Leave the default settings and click **Next**.
6. Click **Finish**.
7. In the **Flow** window, click **Build**.
8. Wait for completion.

The build output is located in `<workspace>/HelloWorld/build/` and is described in Table 7:

File	Description
<code>HelloWorld.elf</code>	Executable bare-metal application file

Table 7: Vitis Application Build Output

3.4.4 Running Software Applications

This section describes how to run software applications.

1. Start AMD Vitis 2024.1.
2. Open the workspace created in Section 3.4.1.
3. In the **Vitis Components** window, select the HelloWorld application project.
4. Prepare the run configuration:
 - (a) In the **Flow** window, click the gear icon on the right side while hovering over the **Run** button.
 - (b) Use `HelloWorld` as the **Launch Config Name**.
 - (c) Leave the default settings.
5. Ensure that the hardware setup is done according to Section 3.2.
6. Click the **Run** button.

The expected output in the serial terminal console is given in Figure 6. For troubleshooting refer to Section 5.2.

```
Hello World
Successfully ran Hello World application
```

Figure 3: Example of the Terminal Output for the HelloWorld Application

4 Boot Configurations

A boot image can be created containing the previously generated binaries to enable booting from various boot modes. The boot image contains the bitstream for programming the FPGA logic and the software bare-metal application.

Tip

Pre-generated images may be used for booting instead of rebuilding the image. The binaries for any supported KX2 product model and boot mode are released on the reference design Github release page [5].

4.1 Generating the Image File

This section describes how to generate a boot image for the Mercury+ KX2 FPGA module using the AMD Vitis IDE. The workspace, hardware platform and application from Sections 3.4.1, 3.4.2 and 3.4.3 are used. To create a bootable image file in the AMD Vitis IDE, follow these steps:

1. Select **Vitis > Program Device**.
2. Select the bitstream created in Section 3.3 as the **Bitstream/PDI**.
3. Select the memory map file created in Section 3.3 as the **BMM/MMI File**.
4. Click **bootloop** under **ELF/MEM File to Initialize in Block RAM** and select **Browse**.
5. Navigate to the build folder of the HelloWorld application and select the `HelloWorld.elf` file.
6. Click **Generate**.
7. Vitis will report an error if JTAG is not connected when clicking **Generate**. However, the bitstream containing the HelloWorld application is generated successfully.
8. Close the **Program Device** window.

The build output is located in the `<workspace>/HelloWorld/_ide` folder and is described in Table 8. The `Mercury_KX2_PE3.bit` file is the same for all boot modes.

File	Description
<code>Mercury_KX2_PE3.bit</code>	Bitstream containing the HelloWorld application

Table 8: Boot Image

4.2 QSPI Flash Boot

4.2.1 Programming the QSPI Flash

There are several ways of programming the QSPI flash memory with the `Mercury_KX2_PE3.bit` created in Section 4.1. Only **one** method needs to be followed in order to program the QSPI flash.

Enclustra MCT

Enclustra provides the MCT [4] application⁶ to program the QSPI flash. The JTAG boot mode first needs to be set in order to program the QSPI flash via the FTDI using the Enclustra MCT application. Refer to Figure 2 to locate the labels on the base board.

1. Ensure that all tools that might be connected to the FTDI (Vivado Hardware Manager, Vitis, any serial terminal) are closed.
2. Disconnect the power supply of the base board (label **12 V DC**).
3. Disconnect the Micro-USB cable from the base board.

⁶Only available on Windows OS.

4. Set CFG A to [1: ON, 2: ON, 3: OFF, 4: ON].
5. Reconnect the Micro-USB cable.
6. Reconnect the power supply of the base board (label **12 V DC**).
7. Open MCT and click **Enumerate**.
8. After enumeration completed successfully, select the detected module.
9. Select **SPI Flash Configuration**.

Tip

While not necessary, it is recommended to erase the flash memory before programming the FPGA. This ensures that the flash memory is in a known state before programming. In order to do that, select **Erase > Full chip** and click **Erase**.

10. Choose **Program** as the operation.
11. Click **Browse** and select the `Mercury_KX2_PE3.bit` file created in Section 4.1.
12. Disable **Boot after programming**.
13. Click **Program**.

Vitis

In order to program the QSPI flash using Vitis, follow these steps:

1. In Vitis, open the workspace created in Section 3.4.1.
2. Select **Vitis > Program Flash**. A dialog as the one shown in Figure 4 appears.
3. In the **Program Flash Memory** window, click **Browse**.
4. Select the `Mercury_KX2_PE3.bit` file created in Section 4.1.
5. Select **s25fl512s-spi-x1_x2_x4** for the **Flash Type**.
6. Click **Program**.

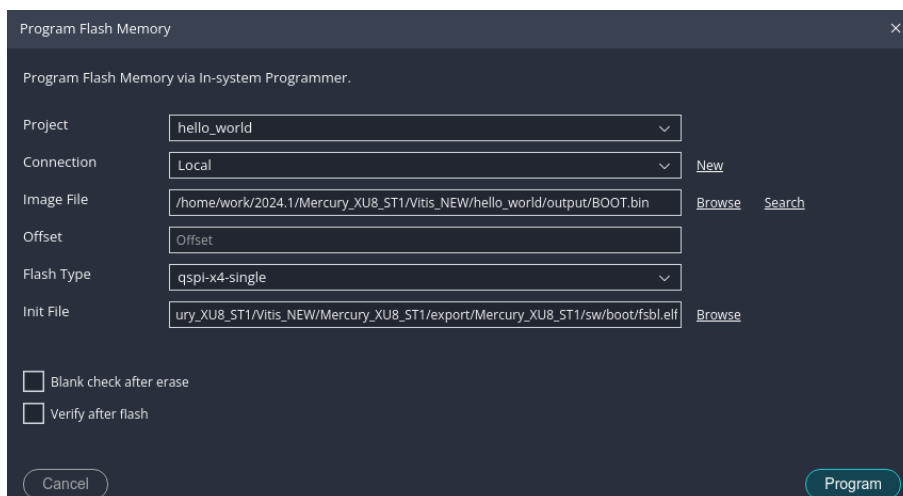


Figure 4: Example Dialog Vitis Program Flash Memory (QSPI)

Vivado

In order to program the QSPI flash using Vivado, follow these steps:

1. Open Vivado.
2. Click **Open Hardware Manager**.
3. Select **Open target > Auto Connect**. The hardware manager should be able to connect to the device as shown in Figure 5.
4. Right-click the FPGA and select **Add Configuration Memory Device....** A dialog as the one shown in Figure 5 appears.
5. In the new window, select the following:

- (a) Manufacturer: Cypress/Spansion
 - (b) Type: qspi
 - (c) Density: 512
 - (d) Width: x1_x2_x4
6. Select part **s25fl512s** from the listed options.
 7. Click **OK**.
 8. In the new window, click **OK** again to program the configuration memory.
 9. In the **Program Configuration Memory Device** dialog, do the following:
 - (a) Select the `Mercury_KX2_PE3.bit` file created in Section 4.1 as the **Configuration file**.
 - (b) Leave the default settings for the rest of the options.
 - (c) Click **OK**.
 - (d) Wait for completion.

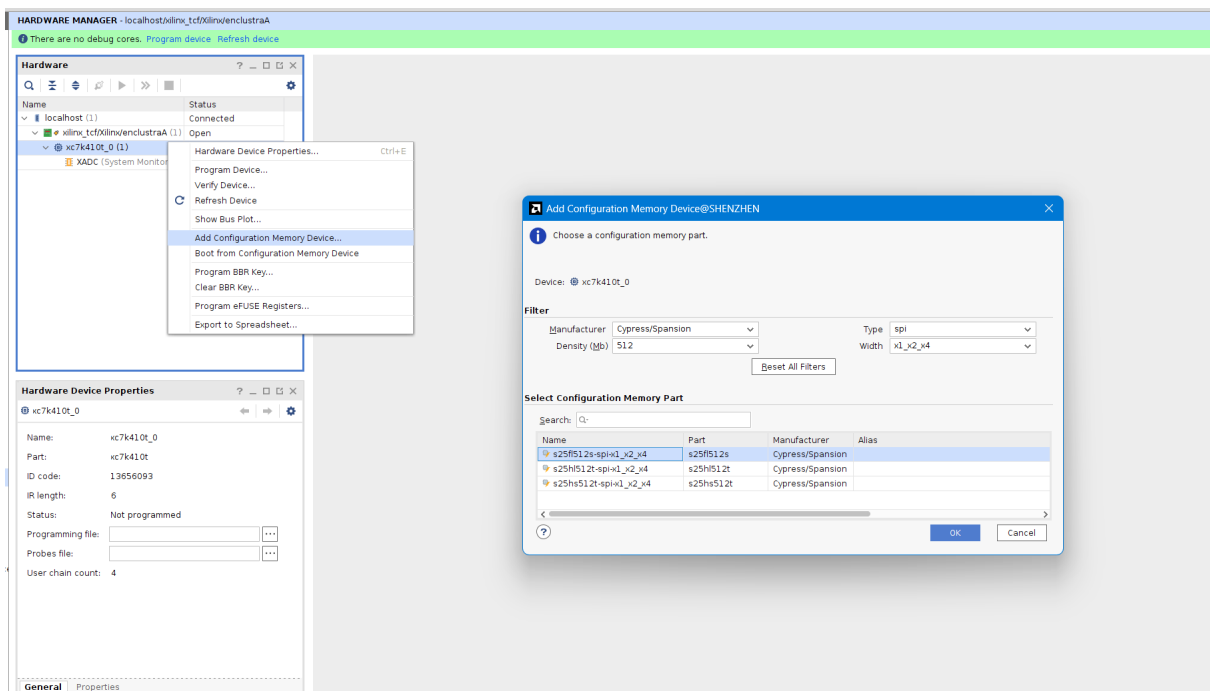


Figure 5: Example Dialog Vivado Configuration Memory (QSPI)

4.2.2 Preparing the Hardware

Refer to Figure 2 to locate the labels on the base board.

1. Disconnect the power supply of the base board (label **12 V**).
2. Disconnect all USB cables from the base board.
3. Set the configuration DIP switch **CFG A** on the base board as follows (label **CFG A**):
 - CFG A = [1: ON, 2: OFF, 3: OFF, 4: ON]
4. Connect the Micro-USB cable between the computer and the base board using the Micro-USB port labeled **USBUB**.
5. Reconnect the power supply of the base board (label **12 V DC**).

4.2.3 Booting from the QSPI Flash

Refer to Figure 2 to locate the labels on the base board.

1. Ensure that the hardware configuration is done according to Section 4.2.2.
2. Reconnect the serial terminal.
3. Press the power-on reset button (label **POR**) and release it after 1 s.
4. The DONE LED (label **LEDs**) lights up after the bitstream has been loaded.
5. The LED on the module starts blinking as described in Section 2.2.9.

The expected output in the serial terminal console is given in Figure 6.

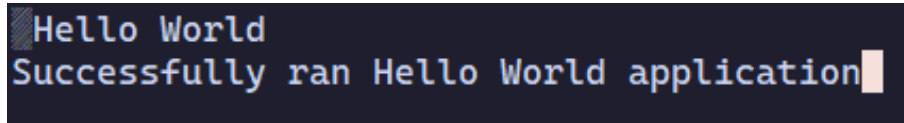
A screenshot of a terminal window with a dark background. The text 'Hello World' is displayed on the first line, and 'Successfully ran Hello World application' is displayed on the second line. A small white cursor is visible at the end of the second line.

Figure 6: Example of the Terminal Output

5 Troubleshooting

Ensure the issues are not listed in the Mercury+ KX2 FPGA Module Known Issues and Changes [7] or the Mercury+ PE3 Base Board Known Issues and Changes [8].

5.1 Vivado Issues

The Windows OS Path is too Long.

The path length limitation of Windows OS can lead to seemingly random errors during the build process.

- Reduce the path length to the Vivado project upon project creation. Refer to [AR52787](#) for further details.

The Block Design Changes Are Not Propagated into the Implementation Step.

Regenerate the block design files:

1. Right-click the block design (.bd) file.
2. Click **Reset Output Products** > **Reset**.
3. Right-click the block design (.bd) file.
4. Click **Generate Output Products** > **Generate**.

5.2 Vitis Issues

The Platform Component Generation Fails.

Use a clean workspace:

1. Close Vitis.
2. Delete the workspace folder.
3. Restart Vitis and retry the steps described in Sections [3.4.1](#) and [3.4.2](#).

The Software Application Launch Fails.

After launching a software application as described in Section [3.4.4](#), launching a second run might fail with the following error:

```
ERROR : Failed to initialize the hardware Invalid target. Use "connect" command to connect to hw_server/TCF agent
```

The problem is usually fixed by clicking the **Run** button again.

5.3 JTAG Issues

The JTAG Is Not Recognized.

- Ensure that the hardware configuration complies with Section [3.2](#).
- Ensure that only **one** JTAG adapter is active and connected to the hardware. Using an AMD JTAG Programmer and the built-in JTAG via FTDI simultaneously will not work. Refer to the Mercury+ PE3 Base Board User Manual [3] for details regarding the built-in JTAG functionality.
- Ensure that the device is powered when trying to connect via JTAG.

5.4 UART Connection Issues

The COM Ports are not Recognized.

- Ensure that the USB cable is properly connected to the Micro-USB connector on the Mercury+ PE3 base board
- Ensure that the FTDI VCP drivers are installed properly.
- (Windows OS only) Ensure that VCP is enabled:
 1. Open the Windows **Device Manager**.
 2. Under **Universal Serial Bus Controllers**, right-click **USB Serial Converter A** and select **Properties**.
 3. In the **Advanced** tab, activate the **Load VCP** checkbox.
 4. Disconnect and reconnect the Micro-USB cable.
 5. After a refresh of the **Device Manager**, two new COM ports should appear in the **Ports (COM & LPT)** section.

There is No Output in the Serial Terminal.

- (Windows OS only) Ensure that no instance of the Enclustra MCT [4] is running. If MCT was open:
 1. Close MCT.
 2. Disconnect and reconnect the Micro-USB cable.
 3. Reopen the serial terminal.

There are Unexpected Characters in the Serial Terminal Output.

Ensure that the serial terminal settings comply with Table 3.

5.5 QSPI Issues

The QSPI Programming Fails.

Try any of the other methods described in Section 4.2.1.

The QSPI Boot Fails.

Ensure that the hardware setup complies with Section 4.2.2.

5.6 MCT Issues

The Module Enumeration Fails.

1. Disconnect all USB cables.
2. Ensure that all tools are closed that may be connected to the FTDI (Vivado Hardware Manager, Vitis, any serial terminal).
3. Ensure that no leftover instances of the hw_server process are running using the Windows task manager.
4. Configure the DIP switches according to Section 3.2.
5. Reconnect the USB cable.
6. Ensure that VCP is enabled as described in Section 5.4.
7. Click **Enumerate** again.

If enumeration is still unsuccessful, select **Settings > Enable Diagnostic Logging**. Click **Enumerate** and send the log file to the Enclustra Support Channel [1] for further analysis.

List of Figures

1	Hardware Block Diagram	6
2	Mercury+ PE3 Base Board Assembly Drawing (Top View)	11
3	Example of the Terminal Output for the HelloWorld Application	14
4	Example Dialog Vitis Program Flash Memory (QSPI)	16
5	Example Dialog Vivado Configuration Memory (QSPI)	17
6	Example of the Terminal Output	18

List of Tables

1	Directory Structure	4
2	Path Variables	4
3	UART Configuration	8
4	FPGA Blinking LED Configuration	8
5	FPGA AXI GPIO Configuration	8
6	Vivado Programming Files	13
7	Vitis Application Build Output	14
8	Boot Image	15

References

- [1] Enclustra Support Channel
support@enclustra.com
- [2] Mercury+ KX2 FPGA Module User Manual
<https://www.enclustra.com/me-kx2-user-manual>
- [3] Mercury+ PE3 Base Board User Manual
<https://www.enclustra.com/me-pe3-user-manual>
- [4] Enclustra Module Configuration Tool (MCT)
<https://www.enclustra.com/module-configuration-tool>
- [5] Mercury+ KX2 PE3 Reference Design Releases
https://github.com/enclustra/Mercury_KX2_PE3_Reference_Design/releases
- [6] Enclustra Application Notes
<https://github.com/enclustra/I2CAppNote>
<https://github.com/enclustra/GigabitEthernetAppNote>
- [7] Mercury+ KX2 FPGA Module Known Issues and Changes
<https://www.enclustra.com/me-kx2-known-issues-and-changes>
- [8] Mercury+ PE3 Base Board Known Issues and Changes
<https://www.enclustra.com/me-pe3-known-issues-and-changes>
- [9] Vivado Design Suite User Guide, UG910, AMD Xilinx, 2024
<https://docs.amd.com/r/en-US/ug910-vivado-getting-started>
- [10] Vivado Design Suite User Guide: Release Notes, Installation, and Licensing, UG973, AMD Xilinx, 2024
<https://docs.amd.com/r/en-US/ug973-vivado-release-notes-install-license/Release-Notes>
- [11] Vitis Unified Software Platform, UG1416, AMD Xilinx, 2024
<https://docs.amd.com/v/u/en-US/ug1416-vitis-documentation>
- [12] Embedded Design Tutorials
<https://github.com/Xilinx/Embedded-Design-Tutorials/tree/master>
- [13] MicroBlaze Processor Embedded Design User Guide, UG1579, AMD Xilinx, 2024
<https://docs.amd.com/r/en-US/ug1579-microblaze-embedded-design/Introduction>