Mercury ZX1 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 ZX1 SoC module reference design and to provide the user with a step-by-step guide to the complete AMD® SoC design flow used for the Mercury ZX1 SoC module.

Summary

This document gives an overview of the Mercury ZX1 SoC module reference design and then guides through the complete AMD SoC design flow for the Mercury ZX1 SoC module in the getting started section. In addition, the internals and the boot options of the Mercury ZX1 SoC module reference design are described.

Product Information	Code	Name
Module	ME-ZX1	Mercury ZX1 SoC Module
Baseboard	ME-ST1	Mercury+ ST1 Base Board

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-489-023	2024.1_v1.1.3	15.11.2024

Approval Information	Name	Position	Date
Written by	ARUD	FPGA/SoC Senior Embedded Software Engineer	15.11.2024
Verified by	GKOE	Manager, FPGA/SoC Embedded Software Engineering	15.11.2024
Approved by	IJOS	Manager, BU SP	15.11.2024

Table of Contents

1 Overview

1.1 Introduction

The Mercury ZX1 SoC module reference design demonstrates a system using the Mercury ZX1 SoC module in combination with the Mercury+ ST1 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]
- Zynq 7000 SoC Embedded Design Turorial [14]

More information on the module and the base board can be found in the Mercury ZX1 SoC Module User Manual [2] and the Mercury+ ST1 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	Reference design root directory as described in Table 1
<vivado_dir>	<base_dir>/reference_design/Vivado/<module_name>	Vivado project directory

Table 2: Path Variables

1.3 Prerequisites

- IT infrastructure
  - Computer with a microSD card slot (optional[1])
  - Supported OS[2]
- Software
  - AMD Vitis 2024.1 Core Development Kit
  - Enclustra Module Configuration Tool (MCT) [4] (optional[3])
  - a terminal emulation program, for example, Tera Term
- Hardware
  - Enclustra Mercury ZX1 SoC module
  - Enclustra Mercury+ ST1 base board
- Accessories
  - 12 V DC power supply
  - microSD card (optional[1])
  - Micro-USB cable

[1] Only required for SD card boot mode.
[2] A comprehensive list of supported operating systems is given in the Vivado Design Suite Installation Guide [10].
[3] May be used for flash programming, for SoC device configuration or for FTDI configuration.

2 Reference Design Description

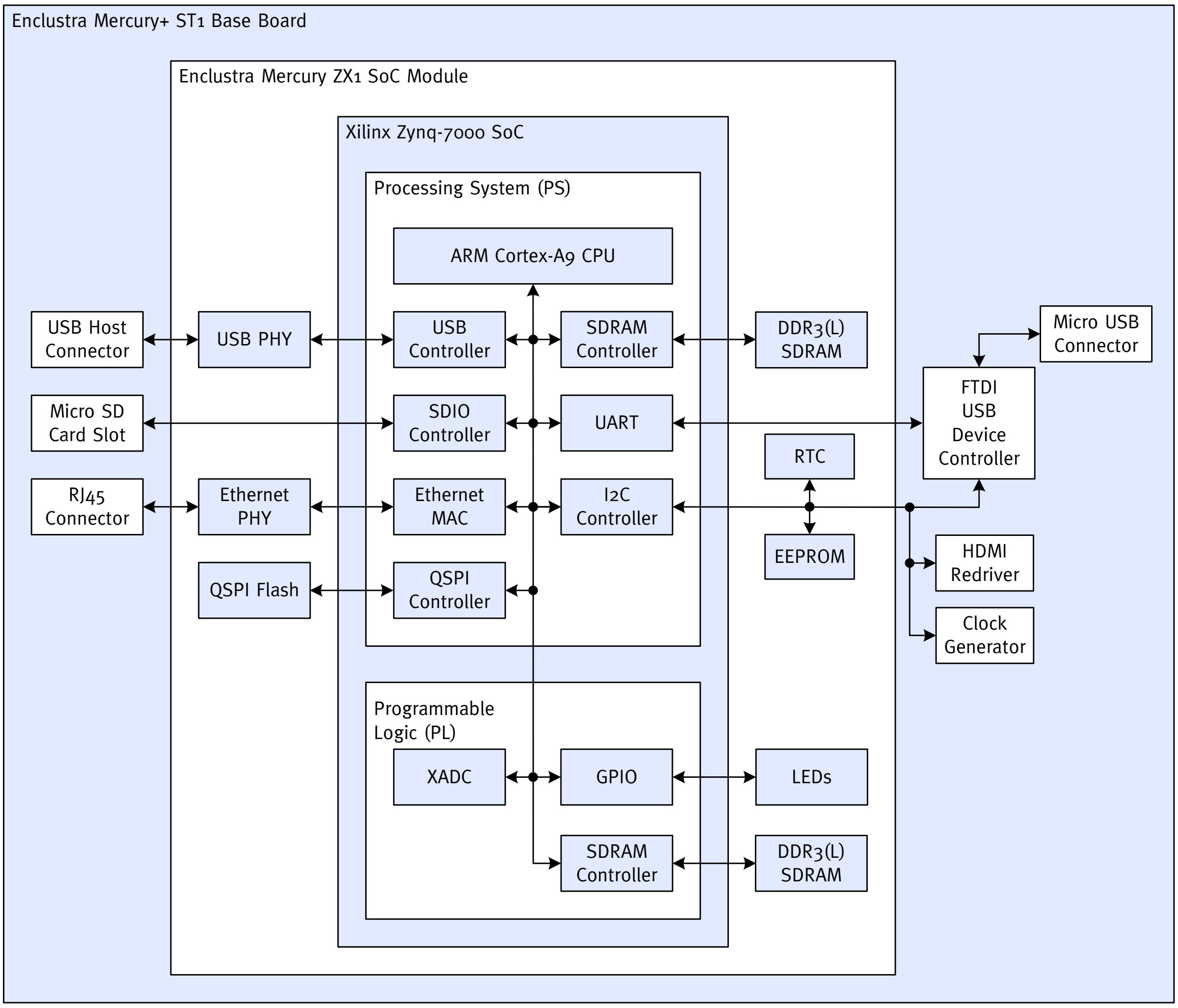

Figure 1: Hardware Block Diagram

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. The maximum CPU (APU) clock performance depends on the device speed grade and package.

A 50 MHz clock and a 100 MHz clock are exported from the PS to the PL. These clocks can be modified in the settings of the PS in the Vivado block design.

2.1.2 DDR3 SDRAM

The DDR3 SDRAM memory is configured as specified in the Mercury ZX1 SoC 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 PS DDR settings need to be adjusted accordingly. The DDR configuration can be modified in the settings of the processing system in Vivado.

2.1.3 microSD Card

The SD0 controller is mapped to MIO[40:45]. This enables microSD card access and booting from the microSD card. In order to boot from a microSD card the configuration DIP switches on the base board must be set according to Section 4.3.2.

> **Tip**
>
> MIO[40:45] are shared between the PS and the PL. To avoid conflicts, the PL pins must be configured to high impedance state before driving the PS MIO pins and vice versa. The PL pins are AA13, AA12, AB17, AB16, AC16 and AC17. In the reference design, this is done via the `BITSTREAM.CONFIG.UNUSEDPIN PULLNONE` property in the constraints file.

2.1.4 I2C

The I2C0 controller is mapped to EMIO pins. It is connected to the I2C bus on the base board. The I2C1 controller is mapped to EMIO pins. It is connected to the I2C FPGA bus on the base board. The available I2C devices on each bus are listed in the Mercury ZX1 SoC Module User Manual [2] and the Mercury+ ST1 Base Board User Manual [3].

2.1.5 Quad SPI Flash Controller

The quad SPI flash controller is mapped to MIO[1:6] in single slave select mode. In order to boot from the QSPI flash the configuration DIP switches on the base board must be set according to Section 4.2.2. Refer to the Mercury ZX1 SoC Module User Manual [2] for details about flash programming and usage. MIO[2:6] are shared between the NAND flash and the QSPI flash on the module. The reference design enables the QSPI flash.

2.1.6 UART

The UART0 controller is mapped to MIO[46:47] and connected to the FTDI USB device controller on the base board. The UART is configured as shown in Table 3.

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

Table 3: UART Configuration

> **Tip**
>
> MIO[46:47] are shared between the PS and the PL. To avoid conflicts, the PL pins must be configured to high impedance state before driving the PS MIO pins and vice versa. The PL pins are AA15 and AA14. In the reference design, this is done via the `BITSTREAM.CONFIG.UNUSEDPIN PULLNONE` property in the constraints file.

2.1.7 Ethernet

The Ethernet MAC ENET 0 controller is mapped to MIO[16:27]. The MDIO interface is mapped to MIO[52:53]. It is connected to a Microchip KSZ9031 RGMII Ethernet PHY on the module. The PHY can be configured via the MDIO interface on address 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].

2.1.8 USB

The USB controller USB0 is mapped to MIO[28:39]. It is connected to a USB3320C USB2.0 PHY on the module. This interface can be configured in USB host, USB device and USB On-The-Go (OTG) mode. The DIP switches on the base board must be configured according to the desired USB mode. Refer to the Mercury+ ST1 Base Board User Manual [3] for details.

2.1.9 GPIOs

The unused MIO pins are available as GPIOs. For details on the I/O assignment, refer to the Mercury ZX1 SoC Module User Manual [2], Section "Multiplexed I/O (MIO) Pins". For details on the combination of the I/O assignment with the base board, refer to the Mercury+ ST1 Base Board User Manual [3].

2.2 Programmable Logic (PL)

2.2.1 DDR3 SDRAM

The DDR3 SDRAM memory in the PL is configured as specified in the Mercury ZX1 SoC Module User Manual [2]. The maximum DDR performance depends on the device speed grade, the package, the VCC_INT voltage and the memory chips equipped on the module. The DDR clock frequency can be modified in the settings of the Memory Interface Generator (MIG) IP core in Vivado.

2.2.2 GPIOs

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. The pin mapping is given in Table 4.

Pin	Signal	Function
H7	FPGA_LED0#	Blinking LED counter MSB

Table 4: PL Blinking LED Configuration

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

FPGA Pin	Signal	Function
H6	FPGA_LED1#	GPIO 0
H9	FPGA_LED2#_PL	GPIO 1

Table 5: PL AXI GPIO Configuration

2.2.3 System Management

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

3 Getting Started

This section describes the steps required to configure the Mercury ZX1 SoC module and the Mercury+ ST1 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 ZX1 SoC Module User Manual [2] and the Mercury+ ST1 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 ZX1 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 ZX1 SoC module may consume more power than can be dissipated without additional cooling measures.

- Ensure that the SoC 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+ ST1 base board is shown in Figure 2. The relevant interfaces are marked in red in the figure as well as in the instructions below.

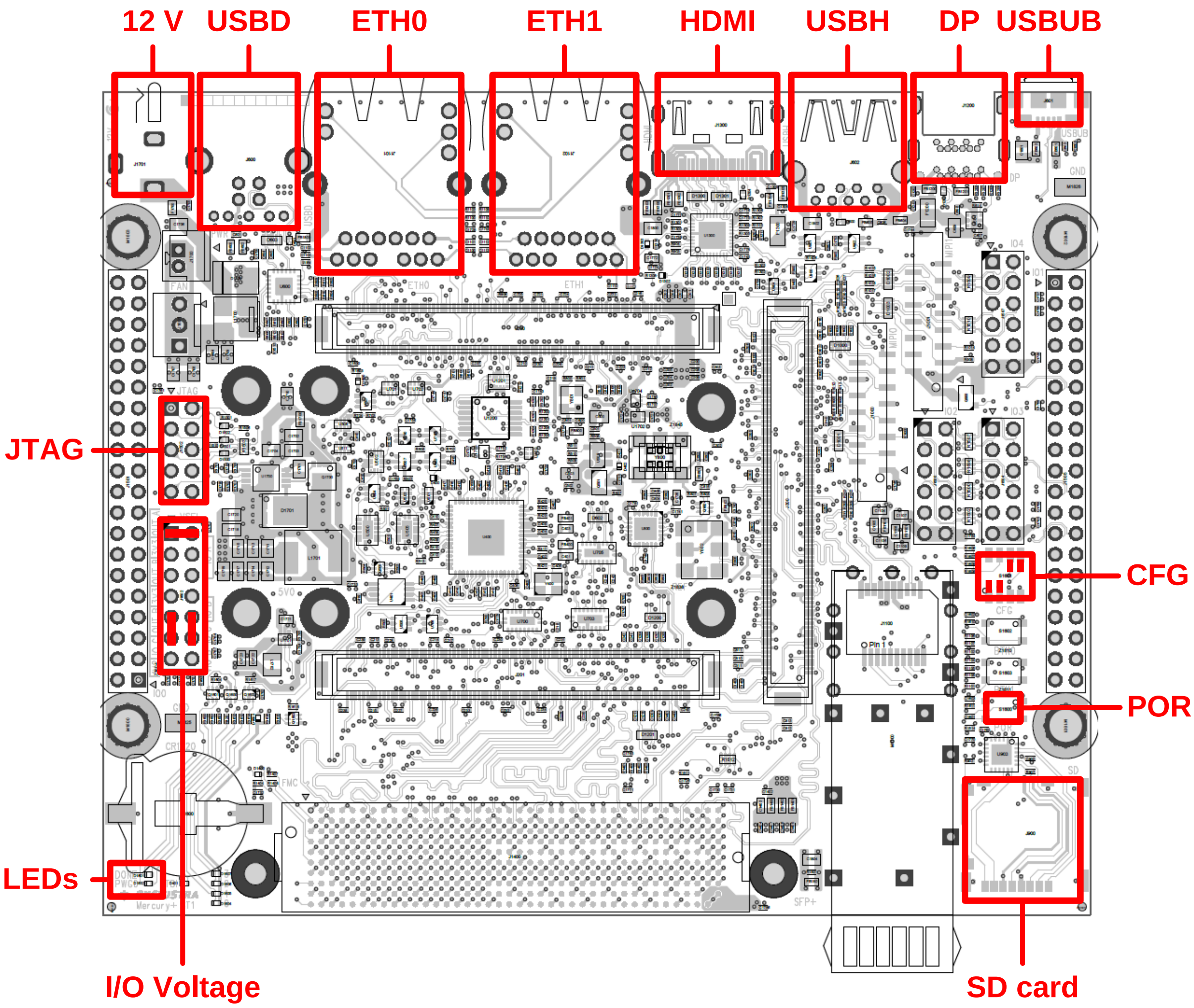

Figure 2: Mercury+ ST1 Base Board Assembly Drawing (Top View)

1. Ensure that no power is connected to the power connector of the baseboard (label **12 V**).
2. Set the I/O voltage jumpers on the base board according to label **I/O Voltage** (the jumpers are marked with red rectangles).
   - IO A = 2.5 V (position 1-2)
   - IO B = 1.8 V (position 10-12)
   - IO C = 1.8 V (position 9-11)
3. Set the configuration DIP switch on the base board as follows (label **CFG**):
   - CFG = [1: OFF, 2: OFF, 3: ON, 4: ON]
4. Install the Mercury ZX1 SoC 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**.
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 the 12 V DC power supply plug to the power connector of the base board (label **12 V**). The PWGD LED lights up (label **LEDs**).

7. Configure the FTDI in Xilinx JTAG mode using the Enclustra MCT [4].
    (a) Open MCT.
    (b) Select **Settings** > **Enable configuring any FTDI**.
    (c) Click **Enumerate**.
    (d) Select the detected device.
    (e) Select **FTDI Configuration**.
    (f) Select **Xilinx JTAG** as the device mode.
    (g) Click **Set device mode**.
    (h) Wait for completion.
    (i) Close MCT.
    (j) 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 2.1.6. 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 troubelshooting section 5.1 in case of problems.

> **Tip**
>
> Pre-generated bitstreams for all supported ZX1 product models are available on the ZX1 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>/reference_design/scripts/settings.tcl` file:
    (a) Set the `module_name` variable to the desired product model[4].
    (b) Save the file after editing[5].
2. Start AMD Vivado 2024.1.
3. Create the Mercury ZX1 SoC module reference design project:
    (a) Click on the **Tcl Console** at the bottom of the Vivado window.
        i. Type `cd {<base_dir>/reference_design}`
           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.

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

[5]The 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) 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**.
   (d) Leave the default settings and click **Next**.
   (e) Click **Finish**.

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

File	Description
`Mercury_ZX1_ST1.xsa`	Handoff file for the Vitis IDE
`Mercury_ZX1_ST1.runs/impl_1/Mercury_ZX1_ST1.bit`	Generated bitstream

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 ZX1 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_ZX1_ST1` 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. Ensure that **Generate Boot artifacts** is enabled.

10. Click **Next**.
11. In the summary screen, click **Finish**.
12. Wait until Vitis has completed all tasks and the created platform is available in the **Vitis Components** window.
13. In the **Flow** window, click **Build**.
14. Wait for completion.

The hardware platform for running bare-metal applications on the Mercury ZX1 SoC module is ready. The FSBL is located in the `<workspace>/Mercury_ZX1_ST1/export/Mercury_ZX1_ST1/sw/boot` directory (Table 7):

File	Description
`fsbl.elf`	Bootloader

Table 7: Vitis Platform Output

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 8:

File	Description
`HelloWorld.elf`	Executable bare-metal application file

Table 8: 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) Select **TCL** for **Board initialization**.
    (d) 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 bootloader, 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 ZX1 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 ZX1 SoC 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. In the **Flow** window, click **Create Boot Image**.
2. Click **Browse** next to **Output BIF File Path***.
3. Create a new folder called `output` .
4. Rename the `.bif` file to `HelloWorld.bif` and click **Save**.
5. The **Output Image*** path should point to the `output` folder automatically.
6. Click **Create Image**.

The build output is located in the previously created output folder and is described in Table 9. The `BOOT.BIN` file is the same for all boot modes.

File	Description
`BOOT.BIN`	Bootable image containing the HelloWorld application
`HelloWorld.bif`	Source file describing the individual components of the boot image

Table 9: 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 `BOOT.BIN` 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] application[6] 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**).
3. Disconnect the Micro-USB cable from the base board.
4. Set CFG to [1: ON, 2: ON, 3: ON, 4: ON].
5. Reconnect the Micro-USB cable.

[6] Only available on Windows OS.

6. Reconnect the power supply of the base board (label **12 V**).
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 `BOOT.BIN` file created in Section 4.1.
12. Disable **Boot after programming**.
13. Click **Program**.

> **Tip**
>
> Alternatively, Linux running directly on the target can be used to program flash memory devices such as QSPI or eMMC. Refer to the Enclustra Petalinux Documentation [15] for details.

JTAG Boot Mode

Programming flash memory devices such as QSPI or eMMC using Vivado or Vitis requires setting the JTAG boot mode first. Follow the steps described in the Enclustra MCT paragraph to set the JTAG boot mode and power the board. After setting the JTAG boot mode, follow the steps described in the Vitis section or the Vivado section below.

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 `BOOT.BIN` file created in Section 4.1.
5. Select **qspi-x4-single** for the **Flash Type**.
6. Click **Program**.

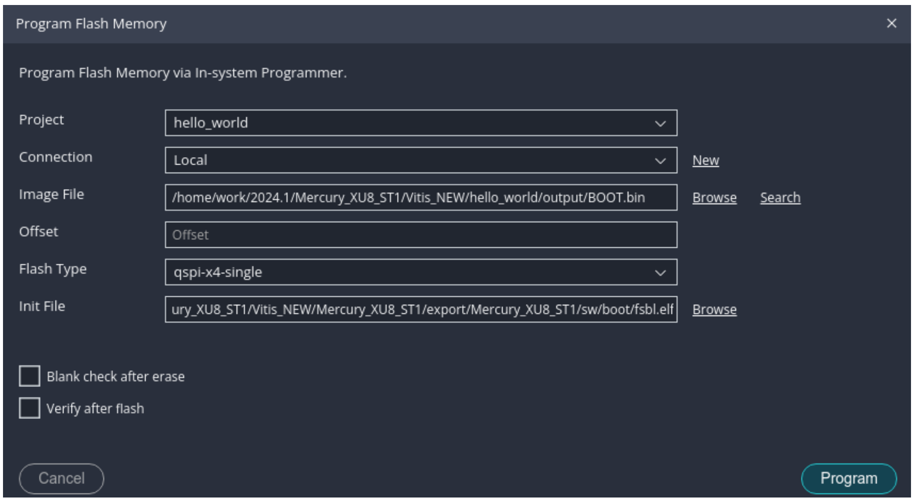

Figure 4: Example Dialog Vitis Program Flash Memory (QSPI)

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 SoC 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: x4-single
6. Select part **s25fl512s-1.8v** 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 `BOOT.BIN` file created in Section 4.1 as the **Configuration file**.
   (b) Select the `fsbl.elf` file created as part of the hardware platform in Section 3.4.2.
   (c) Leave the default settings for the rest of the options.
   (d) Click **OK**.
   (e) Wait for completion.

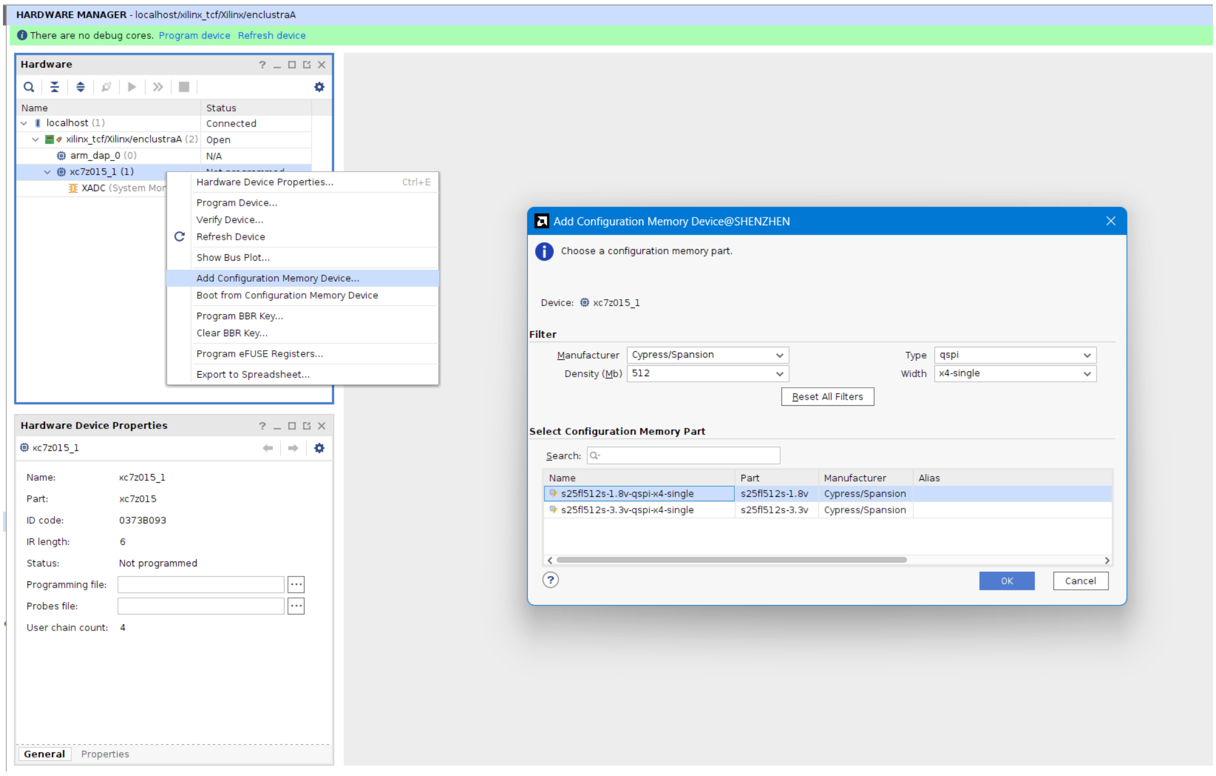

Figure 5: Example Dialog Vivado Configuration Memory (QSPI)

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 on the base board to [1: ON, 2: OFF, 3: ON, 4: ON] (label **CFG**).
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**).

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 2.2.2.

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

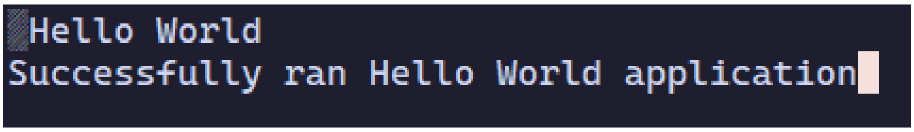

Figure 6: Example of the Terminal Output

4.3 SD Card Boot

4.3.1 Preparing the SD Card

1. Insert an SD card into the SD card slot of the computer.
2. Ensure that the SD card has a FAT32 formatted partition.
3. Copy the `BOOT.BIN` file created in Section 4.1 to the FAT32 partition of the SD card.

4.3.2 Preparing the Hardware

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

1. Detach the power supply of the base board (label **12 V**).
2. Set the configuration DIP switch CFG on the base board to [1: OFF, 2: OFF, 3: ON, 4: ON] (label **CFG**).
3. Insert the SD card into the SD card slot of the base board (label **SD Card**).

4.3.3 Booting from the SD Card

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

1. Ensure that the hardware configuration is done according to Section 4.3.2.
2. Connect the power supply to the base board (label **12 V**).

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

5 Troubleshooting

Ensure the issues are not listed in the Mercury ZX1 SoC Module Known Issues and Changes [7] or the Mercury+ ST1 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**.

Negative Skew Values Issue a Critical Warning

During project creation, 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`
This warning can be safely ignored for Zynq 7000 PS DDR interfaces. Refer to this AMD forum post for details.

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+ ST1 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+ ST1 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.
- Use an SD card image containing a Linux system to program the QSPI flash:
    1. Prepare the SD card image as described in the Enclustra Petalinux documentation [15].
    2. Create a folder named `QSPI` on the FAT32 partition of the SD card.
    3. Copy the `BOOT.BIN` created in Section 4.1 to the `QSPI` folder.
    4. Boot the system from the SD card as described in Section 2.1.3.
    5. Stop the boot process in u-boot by pressing any key during the 3s countdown.
    6. Program the QSPI flash using the following commands:

```
sf probe
sf erase 0x0 0x4000000
fatload mmc 1:1 0x2000000 QSPI/BOOT.BIN
sf write 0x2000000 0x0 $filesize
```

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

List of Tables

References

[1] Enclustra Support Channel
support@enclustra.com
[2] Mercury ZX1 SoC Module User Manual
https://www.enclustra.com/me-zx1-user-manual
[3] Mercury+ ST1 Base Board User Manual
https://www.enclustra.com/me-st1-user-manual
[4] Enclustra Module Configuration Tool (MCT)
https://www.enclustra.com/module-configuration-tool
[5] Mercury ZX1 ST1 Reference Design Releases
https://github.com/enclustra/Mercury_ZX1_ST1_Reference_Design/releases
[6] Enclustra Application Notes
https://github.com/enclustra/I2CAppNote
https://github.com/enclustra/GigabitEthernetAppNote
[7] Mercury ZX1 SoC Module Known Issues and Changes
https://www.enclustra.com/me-zx1-known-issues-and-changes
[8] Mercury+ ST1 Base Board Known Issues and Changes
https://www.enclustra.com/me-st1-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] Zynq 7000 SoC Technical Reference Manual, UG585, AMD Xilinx, 2023
https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM
[14] Zynq 7000 SoC Embedded Design Tutorial, UG1165, AMD Xilinx, 2024
https://docs.amd.com/r/en-US/ug1165-zynq-embedded-design-tutorial
[15] Enclustra Petalinux Documentation
https://github.com/enclustra/PetalinuxDocumentation