

Mars AX3 FPGA Module

Reference Design for Mars ST3 Base Board User Manual

Purpose

The purpose of this document is to present to the user the overall view of the Mars AX3 FPGA module reference design and to provide the user with a step-by-step guide to the complete Xilinx® FPGA design flow used for the Mars AX3 FPGA module.

Summary

This document first gives an overview of the Mars AX3 FPGA module reference design and then guides through the complete Xilinx FPGA design flow for the Mars AX3 FPGA module in the getting started section. In addition, the internals and the boot options of the Mars AX3 FPGA module reference design are described.

Product Information	Code	Name
Product	MA-AX3	Mars AX3 FPGA Module

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-426-005	1.0.0	15.12.2020

Approval Information	Name	Position	Date
Written by	DDUE/ARUD	Design Engineer	20.11.2020
Verified by	GKOE	Design Expert	21.11.2020
Approved by	DIUN	Manager, BU SP	20.11.2020

License

Copyright 2020 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

1	Overview	4
1.1	Introduction	4
1.2	Prerequisites	4
2	Reference Design Description	5
2.1	Microblaze System	5
2.1.1	Clocks	5
2.1.2	Microblaze CPU	5
2.1.3	DDR3/DDR3L SDRAM	5
2.1.4	GPIOs	6
2.1.5	I2C	6
2.1.6	Microblaze Debug Module (MDM)	6
2.1.7	SPI Controller	6
2.1.8	UART	6
2.1.9	Ethernet	7
2.1.10	XADC	7
3	Getting Started	8
3.1	Essential Information	8
3.2	Hardware Setup	9
3.3	FPGA Bitstream Generation	10
3.4	Vitis Workspace Preparation	11
3.5	Running Software Applications	13
4	Boot Configurations	15
4.0.1	Generating the Image File	15
4.1	QSPI Flash Boot	17
4.1.1	Preparing the Hardware	17
4.1.2	Programming the QSPI Flash	17
4.1.3	Booting from the QSPI Flash	21
5	Troubleshooting	22
5.1	Vivado Issues	22
5.2	Vitis Issues	22
5.3	JTAG Connection Issues	22
5.4	UART Connection Issues	22
5.5	QSPI Boot Issues	23

1 Overview

1.1 Introduction

The Mars AX3 FPGA module reference design demonstrates a system using the Mars AX3 FPGA module in combination with the Mars ST3 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]
- Vivado Design Suite Tutorial, Embedded Processor Hardware Design [2]

More information on the Mars AX3 FPGA module and the Mars ST3 base board can be retrieved from their respective user manuals [3] [4].

The following directory structure applies to the AX3 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 AX3 variant are released on the AX3 Reference Design Github page.

1.2 Prerequisites

- Software
 - Xilinx Vivado 2020.1 WebPack, Evaluation, Design or System Edition (check the Mars AX3 FPGA Module User Manual [3] for details on device support in Xilinx tools)
 - Xilinx Vitis IDE
 - Enclustra Module Configuration Tool (MCT) [5] (optional¹)
 - A terminal emulation program (e.g. Tera Term)
- Hardware
 - An Enclustra Mars AX3 FPGA module
 - An Enclustra Mars ST3 base board
- Accessories
 - A 12 V DC power supply
 - A standard micro USB cable
 - A Xilinx JTAG programmer (e.g. Platform Cable USB II) (optional²)

¹May be used for flash programming, for FPGA device configuration or for FTDI configuration.

²Any FTDI device present on Enclustra hardware can be configured to Xilinx JTAG mode using the Enclustra MCT software [5].

2 Reference Design Description

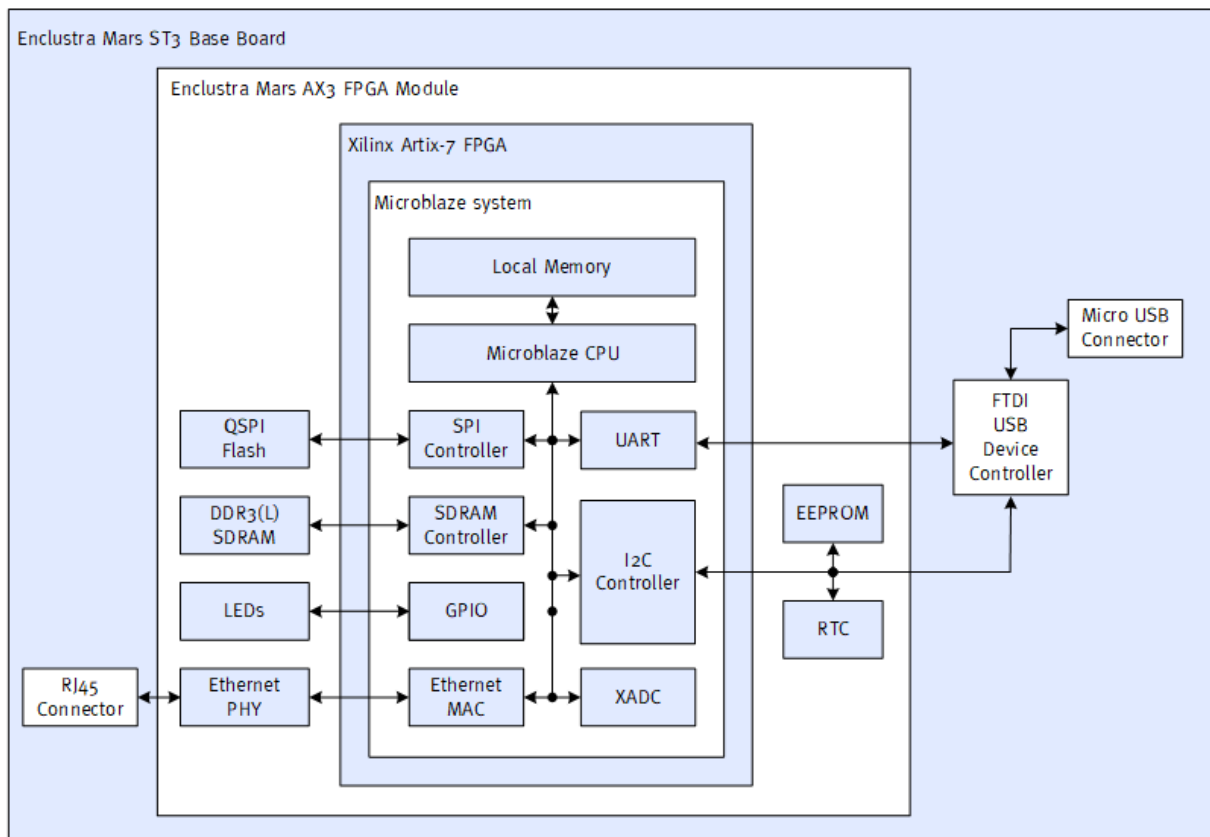


Figure 1: Hardware Block Diagram

2.1 Microblaze System

2.1.1 Clocks

The Microblaze system runs on a 100 MHz clock generated by a PLL inside the Memory Interface Generator (MIG) IP core from an on-board 50 MHz LVDS oscillator.

2.1.2 Microblaze CPU

The Microblaze CPU has access to all peripheral devices via the AXI bus interconnect. The processor has a fast internal memory of 64 kB, along with 8 kB instruction and 8 kB data cache for accesses to the external DDR3/DDR3L SDRAM memory.

The size of the internal memory can be modified from the Address Editor tab in the block design, while the cache dimensions can be modified from the Microblaze wizard.

2.1.3 DDR3/DDR3L SDRAM

The DDR3/DDR3L SDRAM memory runs at its corresponding maximum frequency at a voltage of 1.5 V by default. These parameters can be modified in the Memory Interface Generator (MIG) IP core. In addition to the IP core changes it is necessary to change the top level assignment of DDR3_VSEL signal from high impedance to logic low for a voltage of 1.35V.

The DDR settings in the MIG IP core must be configured according to the Mars AX3 FPGA module User Manual [3].

2.1.4 GPIOs

A Xilinx GPIO controller is connected to the Microblaze processor via an AXI bus. Some FPGA GPIOs are connected to LEDs in the top level, as described in Table 2.

The FPGA 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.

FPGA Pin	Signal	Function
M16	LED0#	Blinking LED counter MSB
M17	LED1#	GPIO 1, controlled by the FPGA GPIO controller
L18	LED2#	GPIO 2, controlled by the FPGA controller
M18	LED3#	GPIO 2, controlled by the FPGA controller

Table 2: FPGA Firmware I/O Configuration

2.1.5 I2C

For available devices on the I2C bus refer to the Mars AX3 FPGA Module and Mars ST3 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].

2.1.6 Microblaze Debug Module (MDM)

A Microblaze Debug Module (MDM) is connected to the AXI bus, providing JTAG UART for base boards that do not have a UART connection.

In order to use the MDM block, stdin and stdout must be configured to "JTAG UART" in the board support package (Vitis) and JTAG UART must be selected in the Run Configuration window.

2.1.7 SPI Controller

The SPI controller is connected to the QSPI flash on the Mars AX3 FPGA module. Please refer to the Mars AX3 FPGA Module User Manual [3] for details about flash programming and usage.

As the QSPI flash is connected to the configuration pins and the clock pin is not accessible as normal user I/O pin, the SPI controller requires the "STARTUPE2" primitive to drive the clock, and the SPI clock needs to be assigned to high impedance in the top-level HDL file.

The clock connection is done in Vivado by selecting the "Enable STARTUPE2 Primitive" checkbox from the SPI controller settings within the block design.

2.1.8 UART

The UART is configured as shown in Table 3.

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

Table 3: UART Configuration

2.1.9 Ethernet

The Ethernet MAC in the block design is connected to a Microchip (Micrel) KSZ9031 Ethernet PHY on the Mars AX3 FPGA module using RGMII interface. The PHY can be configured via the MDIO management interface on PHY address 3.

Note that without Tri-Mode Ethernet MAC and Tri-Mode Ethernet MAC Controller licenses (Permanent or Evaluation), the FPGA bitstream generation will not be permitted. Therefore, Enclustra also provides reference designs for all Mars AX3 FPGA module variants without Ethernet support.

An Ethernet Application Note is available as well providing sample code and more details about using the Ethernet on Enclustra hardware [8].

Please note that the RGMII delays in the Ethernet PHY need to be configured according to the Mars AX3 FPGA module User Manual [3] when the Ethernet interface can be used.

Note that the Mars AX3 FPGA module has two Gigabit Ethernet PHYs that share the MDIO interface. The PHYs can be configured by using the corresponding addresses: PHY0 has address 3, PHY1 has address 7. The second Ethernet interface of the Mars AX3 FPGA module is not used in the reference design.

2.1.10 XADC

A Xilinx XADC IP core instance is connected to the Microblaze CPU 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.

Depending on the user application, the Mars AX3 FPGA module may consume more power than can be dissipated without additional cooling measures; always make sure the FPGA 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 Mars AX3 FPGA 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 Mars AX3 FPGA module and Mars ST3 base board in order to run a simple HelloWorld example application. The section includes information on how to:

- Mount the module and configure the Mars ST3 base board
- Generate the FPGA bitstream
- Prepare the software workspace
- Run a software application

3.1 Essential Information

Warning!

*Always check that the Mars AX3 FPGA module is aligned to the socket of the Mars ST3 base board accordingly. The base board and module may be damaged if the module is not inserted properly 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 Mars AX3 FPGA module to or from the Mars ST3 base board while the Mars ST3 base board is powered. Always remove or turn off the power supply before mounting or removing the Mars AX3 FPGA module.

Warning!

Please read carefully the Mars AX3 FPGA module and Mars ST3 base board user manuals before proceeding.

Warning!

Depending on the user application, the Mars AX3 FPGA module may consume more power than can be dissipated without additional cooling measures; always make sure the FPGA 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 FPGA 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

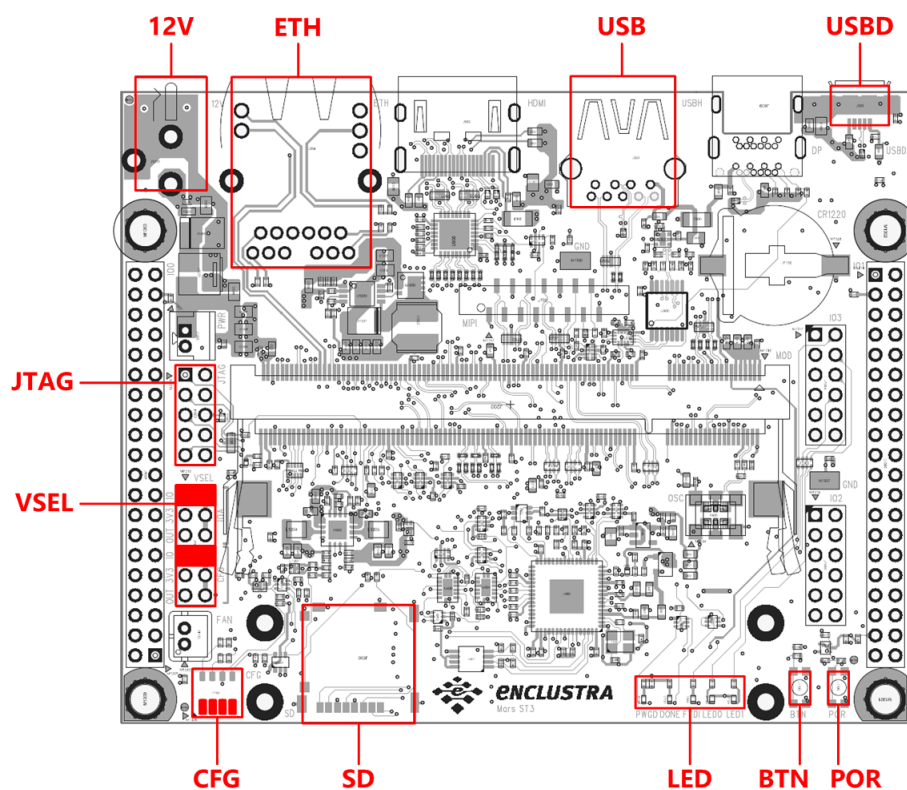


Figure 2: Mars ST3 Base Board Assembly Drawing (Top View)

Step	Description
1	Set the I/O voltage jumpers on the Mars ST3 base board according to label I/O Voltage in Figure 2 (the jumpers are marked with red rectangles): <ul style="list-style-type: none"> VCC_IOA to VCC_IO VCC_CFG to VCC_IO
2	Set VCC_IOB equal to VCC_CFG and set VCC_IO voltage to 1.8 V (i.e. all I/O voltages set to 1.8 V) by configuring the DIP switch on the Mars ST3 base board as follows (see CFG in Figure 2): <ul style="list-style-type: none"> CFG = [1: OFF, 2: OFF, 3: OFF, 4: OFF]
3	Mount the Mars AX3 FPGA module to the Mars ST3 base board.
4	Connect the micro USB cable between your computer and the Mars ST3 base board. Use the micro USB port labeled USB D in Figure 2.
5	Connect the 12 V DC power supply plug to the power connector of the Mars ST3 base board (see label 12V in Figure 2).

Continued on next page...

Step	Description
6	<p>Make sure that the FTDI device on the Mars ST3 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 Mars ST3 base board (see label JTAG in Figure 2).</p> <p>Details on the Xilinx JTAG mode configuration and on the JTAG connector are presented in the Mars ST3 Base Board User Manual [4]</p>
7	<p>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.</p> <p>For issues related to COM ports detection, refer to Section 5.4.</p> <p>Configure the UART parameters according to Section 2.1.8.</p>

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 AX3 variant is released on the AX3 Reference Design Github page.

Step	Description
1	<p>Configure the settings file:</p> <ol style="list-style-type: none"> 1. Edit the <code>module_name</code> variable in <code>scripts/settings.tcl</code> file, according to your modules name. This file includes module name and board information required for the project creation script. All settings, except for <code>module_name</code> should be left on default. The list of options for <code>module_name</code> is given in the comments within the Tcl file. 2. Save the file after editing.
2	<p>Start Xilinx Vivado 2020.1 and create the Mars AX3 FPGA module reference design project:</p> <ol style="list-style-type: none"> 1. Click on the Tcl console at the bottom of the page and type: <ol style="list-style-type: none"> (a) <code>cd {<base_dir>}</code> where <code><base_dir></code> is the directory in which you extracted the archive contents. Note the <code>{}</code> around the path. (b) <code>source ./scripts/create_project.tcl</code> (c) Alternatively the script can be run by passing the module name as an argument instead of changing the <code>module_name</code> variable in the <code>settings.tcl</code> file: <ol style="list-style-type: none"> i. <code>vivado -mode batch -source ./scripts/create_project.tcl -tclargs <module_name>.</code> ii. <code>open_project ./Vivado/<module_name>/<project_name>.xpr</code> 2. Wait for completion

Continued on next page...

Step	Description
3	Run Synthesis, Implementation & Bitstream Generation in Vivado 2020.1: <ol style="list-style-type: none"> 1. Click on Generate Bitstream from the Flow Navigator bar 2. In the Launch Runs window click OK - this will start automatically the entire implementation process 3. Wait for completion → select View Reports → OK
4	Export the hardware system information (required for the Vitis IDE): <ol style="list-style-type: none"> 1. File → Export → Export Hardware 2. Choose Fixed for platform type and click Next 3. Select Include Bitstream and click Next 4. Leave the file name and export location as default and click Next 5. Click Finish

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

Please note that without Tri-Mode Ethernet MAC and Tri-Mode Ethernet MAC Controller licenses (Permanent or Evaluation), the FPGA bitstream generation will not be permitted. Please contact Xilinx support for details.

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 AX3 variant is released on the AX3 Reference Design Github page.

Step	Description
1	Start the Vitis IDE 2020.1 <ol style="list-style-type: none"> 1. Select any workspace (e.g. <base_dir>\workspace)

Continued on next page...

Step	Description
2	<p>Create a new Platform Project</p> <ol style="list-style-type: none"> 1. File → New → Platform Project 2. In the New Platform Project: <ol style="list-style-type: none"> (a) For Project Name type the <project_name> e.g. Mars_AX3_ST3 (b) Check the "Use default location" checkbox, if it is not enabled yet. (c) Hit Next (d) Select "Create from hardware specification and hit Next (e) Hit the Browse button and select the Hardware Specification .xsa file you exported from Vivado, as described in Section 3.3. The default export location used by Vivado is <base_dir>\<vivado_proj_dir>\<project_name>.xsa. Alternatively, the pre-compiled hardware description file may be used. (f) Hit Next and wait for the file to be analyzed (g) For the Operating System select standalone (h) For the Processor select CPU_Microblaze_CPU (i) Hit Finish and wait for completion (j) Build the platform by pressing Ctrl-B and wait for completion
3	<p>Create a new application</p> <ol style="list-style-type: none"> 1. File → New → Application Project 2. In the New Application Project window: <ol style="list-style-type: none"> (a) Click "Next" on the Welcome page (if you have not disabled it) (b) Select the previously generated hardware platform and click "Next" (c) For Project Name type a description for the new application e.g. "HelloWorld" (d) The system project name should be filled automatically, e.g. "HelloWorld_system" (e) Hit Next (f) For the Domain choose the standalone_domain and the C Language. (g) Hit Next and wait for the tool to proceed (h) Select the HelloWorld (or any another) template (i) Hit Finish and wait for completion (j) 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 Mars AX3 FPGA module. The steps are generic, and apply to the software example templates in the Vitis IDE.

Step	Description
1	<p>Create a run configuration for the application in Vitis IDE 2020.1:</p> <ol style="list-style-type: none">1. Right click the previously generated application (e.g. HelloWorld) and select Run As → Run Configurations...2. Right-click Single Application Debug and hit New or double-click on it3. Enter a run configuration name in the Name field (e.g. HelloWorld)4. Application tab:<ol style="list-style-type: none">(a) Make sure "CPU_microblaze_CPU" is enabled(b) In the Project Name field click browse and select an application (e.g. HelloWorld)(c) In the Application field click search and select an .elf file (e.g. HelloWorld.elf)(d) Enable Reset processor checkbox(e) Hit Apply5. Target Setup tab (see Figure 3):<ol style="list-style-type: none">(a) For Hardware Platform refer to the corresponding Platform: e.g. <code>\${sdxTcfLaunchFile:project=HelloWorld;fileType=hw;}</code>(b) For Bitstream file field, hit Search...(c) Select Mars_AX3_ST3.bit and hit OK(d) Use the Auto Detect option for FPGA device(e) Enable checkboxes Reset entire system, Program FPGA(f) Hit Apply
2	<p>Make sure the Hardware is configured according to Section 3.2:</p> <ul style="list-style-type: none">→ Connect the 12 V DC power supply plug to the power connector of the Mars ST3 base board (see label 12 V DC in Figure 2).→ With a serial console program e.g. Tera Term connect to the COM port that corresponds to the Serial Converter B. For issues related to UART, refer to Section 5.4.
3	<p>Start the application by clicking the Run button.</p> <p>This method of starting the application resets the entire system, configures the FPGA with the specified bitstream and runs the application in the Microblaze processing system.</p> <p>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.</p> <p>For issues related to JTAG, refer to Section 5.3.</p>

Table 7: Running an Application Step-By-Step Guide

After the FPGA 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.

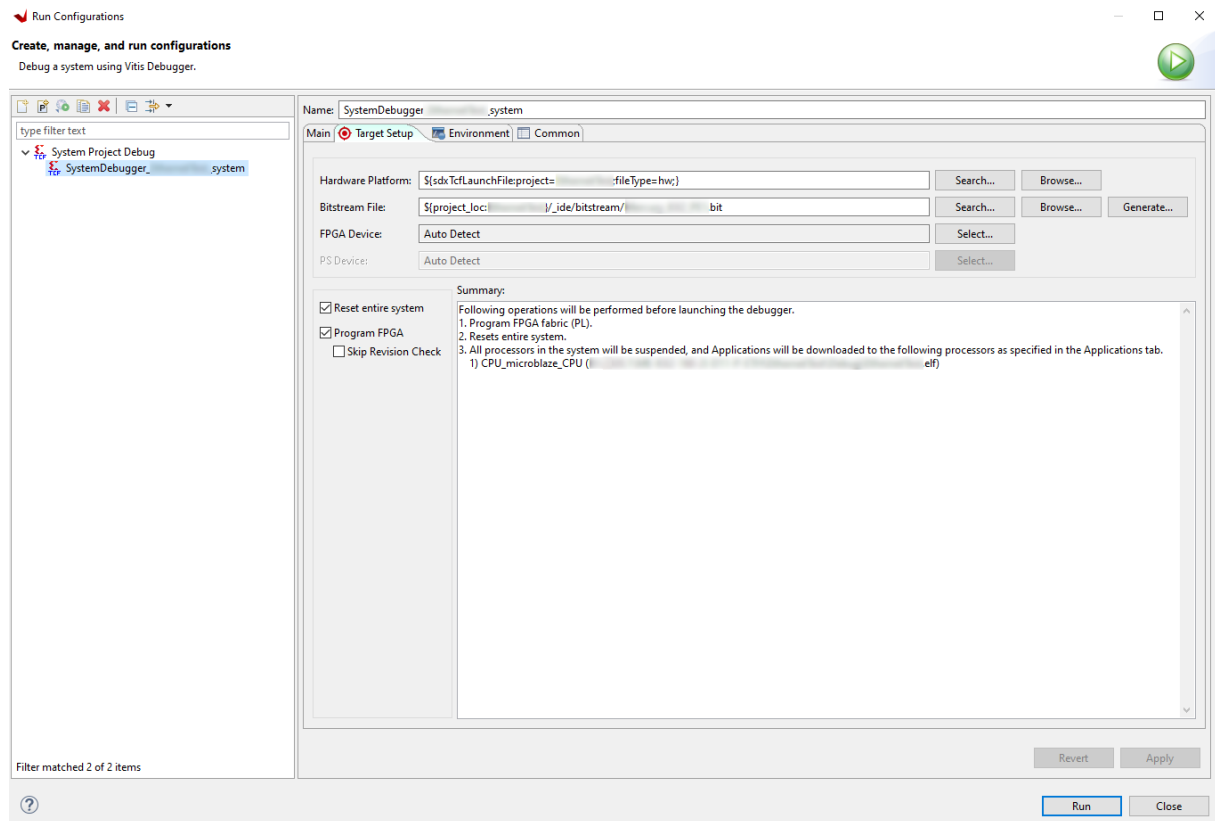


Figure 3: Run Configurations Settings - Target Setup Tab

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 bitstream for programming the FPGA and the software bare-metal application.

The entire software program is written into the on-chip memory. Therefore, when the FPGA is configured with the bitstream, the microprocessor is started and the program is automatically loaded.

For a fast test of the boot configurations, the pre-generated download.bit file may be used to program the flash memory with the bitstream and the example project, instead of rebuilding the image. You need to select the file corresponding to the Mars AX3 FPGA module variant. Pre-generated binaries for any AX3 variant are released on the AX3 Reference Design Github page.

4.0.1 Generating the Image File

Step	Description
1	<p>Create the boot image from Xilinx Vitis 2020.1:</p> <ol style="list-style-type: none">1. Right click on the application in the Project Explorer2. Select Generate Linker Script3. Leave all settings as default (see Figure 4) and click Generate4. Click Yes when asked if you want to overwrite the already existing file5. Rebuild the project6. Right click the project and select Program FPGA7. Under the ELF/MEM File to Initialize in Block RAM select the application .elf file to be written into the FPGA block RAM (see Figure 5)8. Click Generate <p>A bitstream file with the application ELF data will be created in <workspace>\HelloWorld_ide\bootimage\download.bit.</p>
2	<p>Optional - Create the boot image to be written to the QSPI flash</p> <p>Xilinx Vitis or Enclustra MCT support both .bit and .bin formats for the QSPI flash image, but the Vivado Hardware Manager supports only .bin and .mcs formats.</p> <p>It has been found that in Vivado the download.bit image file can be written to the QSPI flash by renaming it to download.bin. If the system should not boot or work as expected afterwards, a .bin or .mcs file can be generated as in the Vivado GUI:</p> <ol style="list-style-type: none">1. In the Vivado menu bar in the Tools menu, open the "Generate Memory Configuration File" window.2. Choose the boot file format first (BIN or MCS) and for the Memory Part select the s25fl512s-spi-x1_x2_x43. In the Options section, set the Interface to SPIx14. Enable the Load bitstream file checkbox5. At Start address "00000000" with direction "up", add the download.bit file.6. From the checkboxes at the bottom, enable the Overwrite option7. Click Ok to generate the boot image.

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

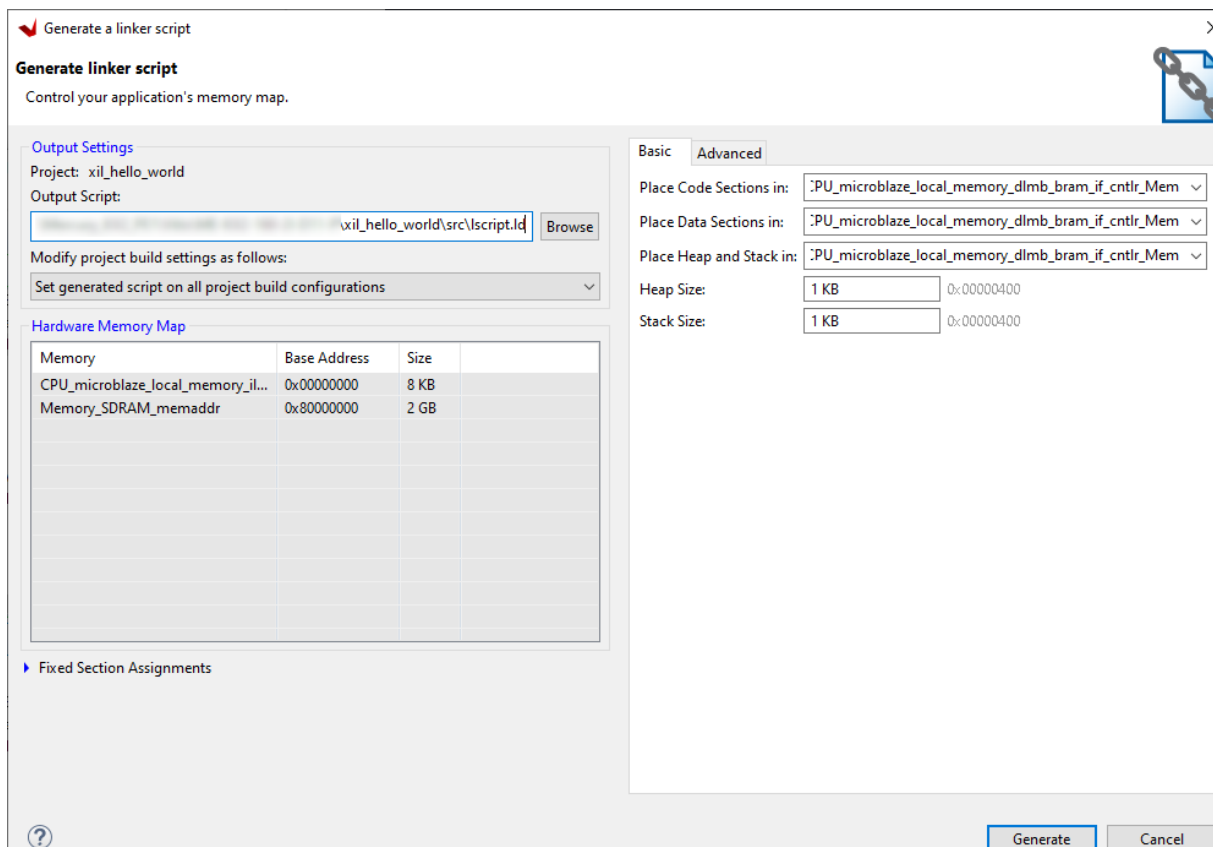


Figure 4: Generate linker scripts settings

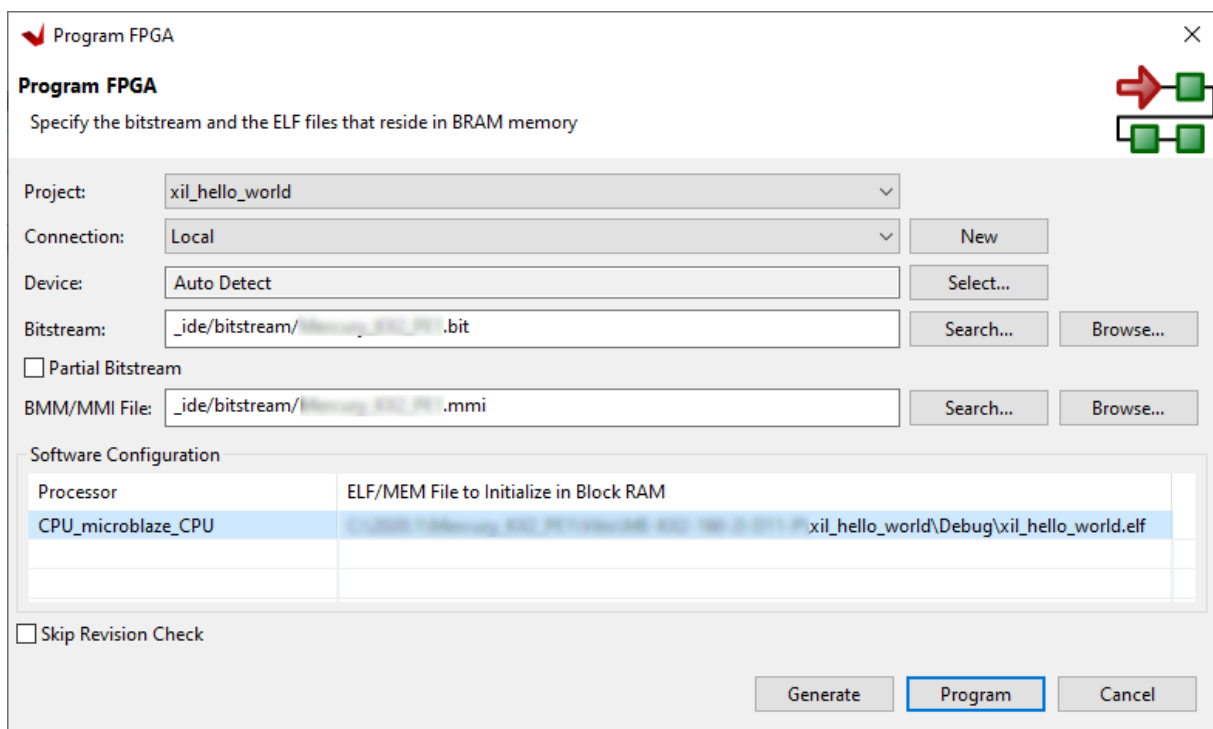


Figure 5: Initialize Block RAM settings

4.1 QSPI Flash Boot

4.1.1 Preparing the Hardware

Step	Description
1	Remove the power supply from the Mars ST3 base board (see label 12 V DC in Figure 2).
2	Enable the QSPI flash boot mode by setting the configuration DIP switches on the Mars ST3 base board as follows (see labels CFG in Figure 2): <ul style="list-style-type: none">• CFG = [1: ON, 2: OFF, 3: OFF, 4: ON]
3	Connect the 12 V DC power supply plug to the power connector of the Mars ST3 base board (see label 12 V DC 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	<p>Program the boot image from Xilinx Vitis 2020.1 (see Figure 6):</p> <ol style="list-style-type: none">1. Right click on the application in the Project Explorer2. Select Program Flash3. For Image File select the generated .bit, .bin or .mcs file4. For Flash Type select s25fl512s-spi-x1_x2_x45. Hit Program and wait for completion <p>The settings in the pictures are for reference only. Note that the configuration file must be selected according to your application.</p>
2*	<p>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.</p> <ol style="list-style-type: none">1. Flow → Open Hardware Manager2. Click on Open target → Auto Connect3. Right click on the corresponding FPGA device in the left bar → Add Configuration Memory Device (see Figure 7)<ol style="list-style-type: none">(a) For Select Configuration Memory Part choose the memory part according to the Mars AX3 FPGA Module User Manual [3], part type single. This is in most cases s25fl512s-spi-x1_x2_x4.(b) Hit OK4. In Program Configuration Memory Device window (see Figure 8):<ol style="list-style-type: none">(a) For Configuration file select the boot image generated as described in Section 4.0.1. Note that Vivado only accepts .bin and .mcs file formats.(b) In Program Operations section:<ul style="list-style-type: none">• For Address Range select Entire Configuration Memory Device• Enable checkboxes Erase, Program and Verify• Hit OK and wait for completion

Continued on next page...

Step	Description
	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*	<p>Optional - alternatively, Enclustra Module Configuration Tool (MCT) [5] can be used to program the QSPI flash. Note that the .mcs format is not supported.</p> <p>Close all other tools that may be connected to the FTDI device (Vivado Hardware Manager, Vitis, UART terminal).</p> <p>Before programming the QSPI flash from MCT, make sure the hardware configuration on the Mars ST3 base board is done according to 4.1.1</p> <p>In order to see the output in the UART terminal, close the MCT tool after QSPI flash programming and disconnect and reconnect the USB cable from the board and the UART terminal.</p>

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

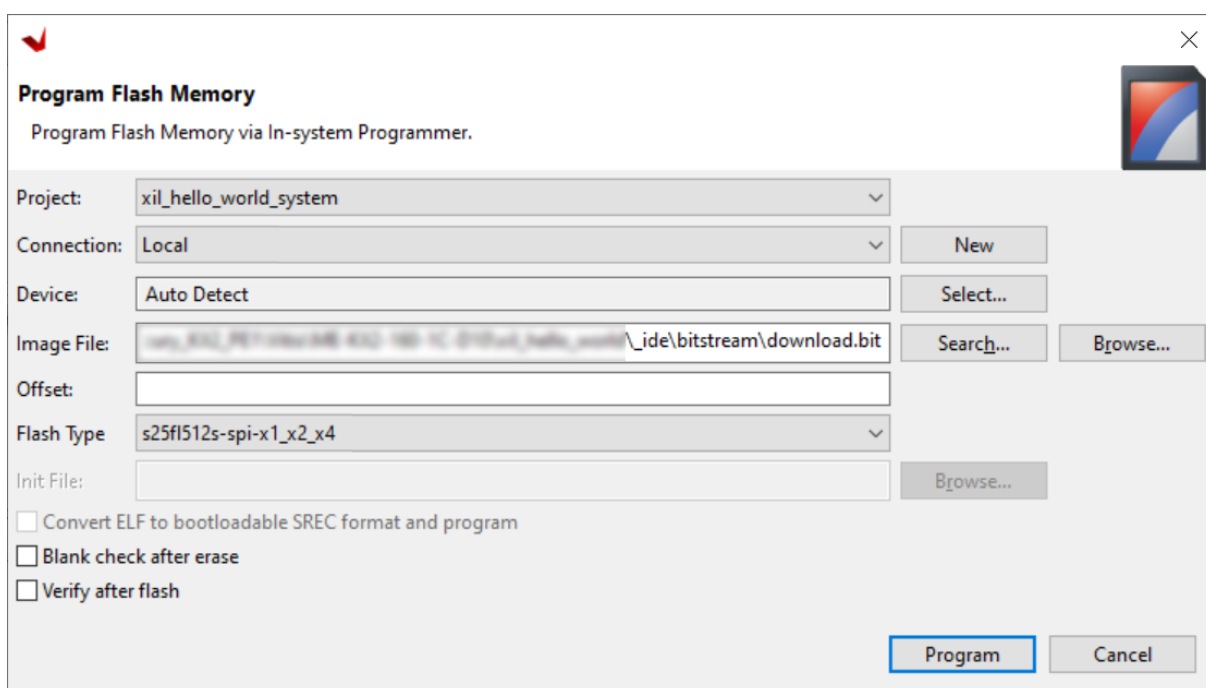


Figure 6: QSPI Flash Programming Settings in Vitis

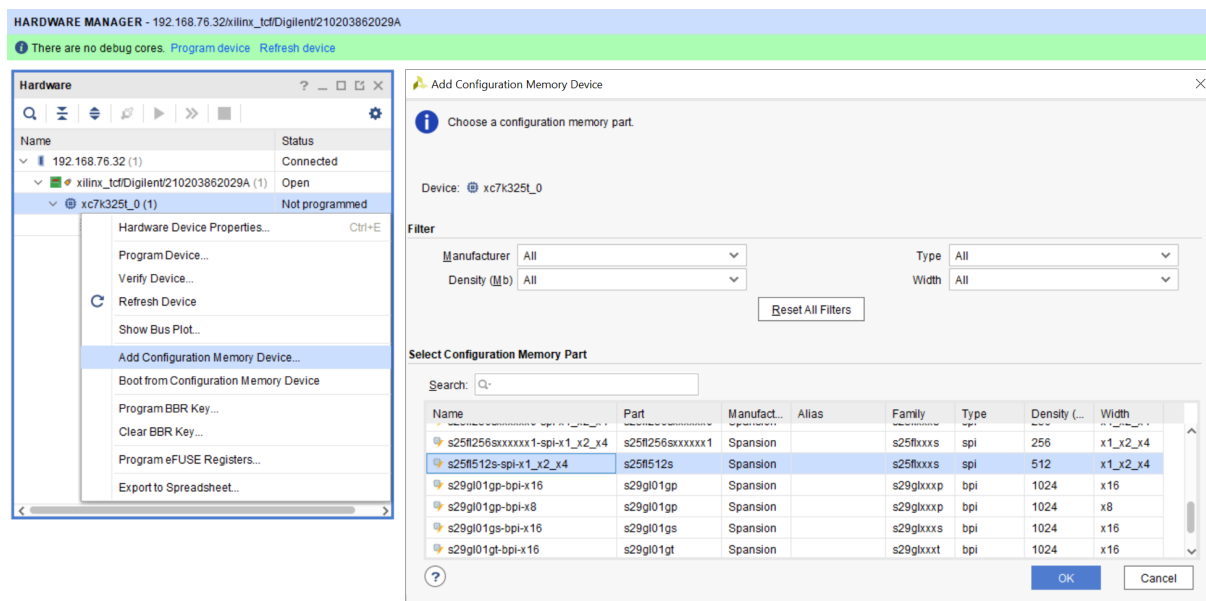


Figure 7: QSPI Flash Programming Settings in Vivado - Adding the Memory Device

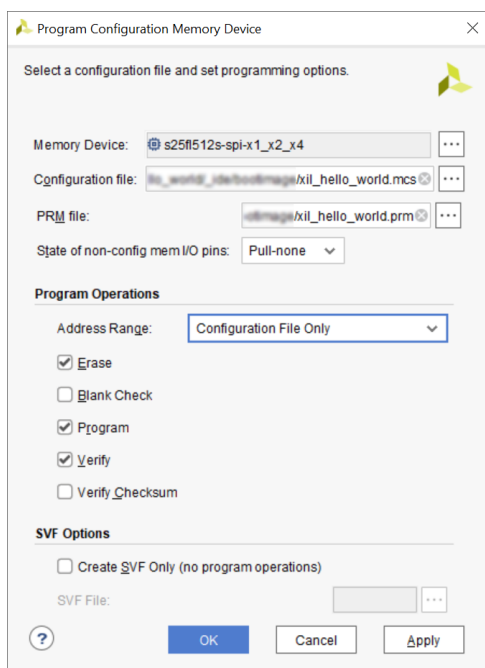


Figure 8: QSPI Flash Programming Settings in Vivado

Warning!

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

4.1.3 Booting from the QSPI Flash

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

Table 11: Booting from the QSPI Flash Step-by-Step Guide

5 Troubleshooting

5.1 Vivado Issues

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

5.2 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.

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 Mars ST3 base board can be retrieved from the Mars ST3 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 Mars ST3 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 Mars ST3 base board:
 1. Check that the USB cable is connected properly
 2. Check that the FTDI VCP drivers are installed
 - (a) Open Device Manager
 - (b) Universal Serial Bus controllers → USB Serial Converter A/B → Properties → Advanced tab → enable Load VCP checkbox
 - (c) Reboot the computer if the COM port is still not detected
 3. Reinstall the FTDI drivers if the problem persists
- If the computer does not output any character in the terminal program:
 1. Check that the FTDI device is set to UART mode:
 - (a) Download and open FT_Prog utility (this is a third party tool offered by the FTDI company to configure FTDI devices)
 - (b) DEVICES → Scan and Parse
 - (c) Check that for Port A and B the RS232 UART property is true

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

5.5 QSPI Boot Issues

- If the Mars AX3 FPGA 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.

List of Figures

1	Hardware Block Diagram	5
2	Mars ST3 Base Board Assembly Drawing (Top View)	9
3	Run Configurations Settings - Target Setup Tab	14
4	Generate linker scripts settings	16
5	Initialize Block RAM settings	16
6	QSPI Flash Programming Settings in Vitis	19
7	QSPI Flash Programming Settings in Vivado - Adding the Memory Device	20
8	QSPI Flash Programming Settings in Vivado	20

List of Tables

2	FPGA Firmware I/O Configuration	6
3	UART Configuration	7
4	Hardware Setup Step-By-Step Guide	10
5	FPGA Bitstream Generation Step-By-Step Guide	11
6	Vitis Workspace Preparation Step-By-Step Guide	12
7	Running an Application Step-By-Step Guide	13
8	Generating the Boot Image File Step-by-Step Guide	15
9	Preparing the Hardware for QSPI Flash Boot Mode Step-by-Step Guide	17
10	Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide	19
11	Booting from the QSPI Flash Step-by-Step Guide	21

References

- [1] Vivado Design Suite User Guide, Embedded Processor Hardware Design, UG898, Xilinx, 2020
- [2] Vivado Design Suite Tutorial Embedded Processor Hardware Design, UG940, Xilinx, 2020
- [3] Mars AX3 FPGA Module User Manual
→ Ask Enclustra for details
- [4] Mars ST3 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