Mars AX3 FPGA Module

Reference Design for Mars PM3 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	Number	Name
Product	MA-AX3	Mars AX3 FPGA Module

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-426-001	03	16.01.2018

Approval Information	Name	Position	Date
Written by	DIUN	Design Engineer	02.12.2015
Verified by	GLAC	Technical Expert	15.12.2015
Approved by	RPAU	Quality Manager	16.01.2018

Copyright Reminder

Document History

Version	Date	Author	Comment
03	16.01.2018	DDUE	Updated to Vivado and SDK 2017.4, other style updates and clarifications
02	10.06.2016	DIUN	Minor style updates and clarifications
01	27.01.2016	DIUN	Version 01

Table of Contents

Overview

Introduction

The Mars AX3 FPGA module reference design demonstrates a system using the Mars AX3 FPGA module in combination with the Mars PM3 base board. It presents the basic configuration of the device and features some example applications.

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.

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 PM3 base board can be retrieved from their respective user manuals [4] [5].

Directory Structure

The Mars AX3 FPGA module reference design is delivered as a ZIP archive file with the following directory structure and contents:

- `binaries` — Pre-compiled binaries directory
- `scripts` — Scripts directory required for Vivado project creation
- `SdkExport` — Pre-generated hardware description files required for SDK applications
- `software` — Software projects directory
- `src` — Xilinx pinout and timing constraints and VHDL source code directory
- `Mars_AX3_Reference_Design_for_Mars_PM3_User_Manual.pdf` — User manual (this document)

Prerequisites

- IT
  - A computer running Windows 7 64-bit (or later)
- Software
  - Xilinx Vivado 2017.4 WebPack, Evaluation, Design or System Edition (check the Mars AX3 FPGA Module User Manual [4] for details on device support in Xilinx tools) with Tri-Mode Ethernet MAC and Tri-Mode Ethernet MAC Controller licenses (Permanent or Evaluation)
  - Xilinx Software Development Kit (SDK) 2017.4
  - Enclustra Module Configuration Tool (MCT) [6] (optional[1])
  - A terminal emulation program (e.g. Tera Term)
  - PuTTY (optional[2])
- Hardware
  - An Enclustra Mars AX3 FPGA module
  - An Enclustra Mars PM3 base board
- Accessories
  - A standard micro USB cable
  - A JTAG PM3 breakout board or a Xilinx JTAG breakout cable
  - A Xilinx JTAG programmer (e.g. Platform Cable USB) download cable

[1] May be used for flash programming, for FPGA device configuration or for Cypress FX3 configuration.
[2] Required for running the Ethernet lwIP example

Reference Design Description

Block Diagram

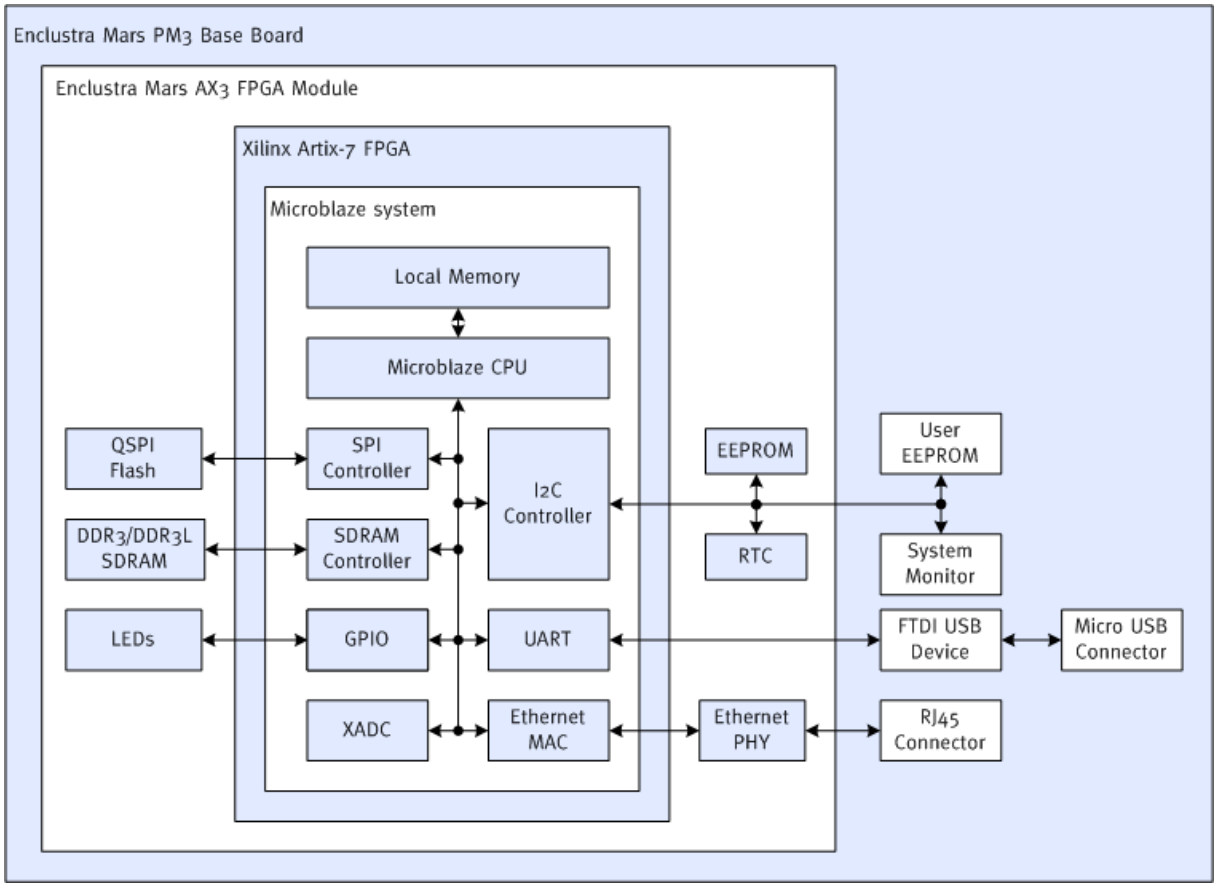

Figure 1: Hardware Block Diagram

Microblaze System

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.

Microblaze CPU

The Microblaze CPU has access to all peripheral devices via the AXI bus interconnect. The processor has a fast internal memory of 16 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.

DDR3/DDR3L SDRAM

The DDR3/DDR3L SDRAM memory runs at 400 MHz (800 Mbit/sec) at a voltage of 1.5 V by default. These parameters can be modified in the MIG IP core. For a voltage of 1.35 V, beside the changes in the block design, it is necessary to change the top level assignment to DDR3_VSEL signal from high impedance to logic low.

The memory size for the MIG dual controller is set to 256 MB. The size can be modified in the Address Editor tab in the block design.

The DDR settings in MIG wizard must be configured according to the Mars AX3 FPGA Module User Manual [4].

I2C

Table 1 lists the connected devices on the Mars AX3 FPGA module and Mars PM3 base board.

Device	Address (7-bit)	Vendor	Part Type
Real-time clock	0x6F	Intersil	ISL12020MIRZ
RTC user SRAM	0x57	Intersil	ISL12020MIRZ
Secure EEPROM	0x5C	Maxim	DS28CN01
I2C GPIO expander	0x21	Semtech	SX1505I087TRT
User EEPROM	0x54	Microchip	24AA128T-I/MNY
System monitor	0x2F	Texas Instruments	LM96080CIMT/NOPB

Table 1: I2C Devices

The device vendors or addresses of the I2C devices may change in future revisions of Mars AX3 FPGA module or Mars PM3 base board.

For detailed information on the I2C devices, please refer to the corresponding user manuals [4] [5].

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 [4] 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 can be done in Vivado by clicking “Enable STARTUPE2 Primitive” checkbox from the SPI controller settings within the block design.

UART

The UART on the Mars AX3 FPGA module is connected to the FTDI USB device controller on the Mars PM3 base board. The UART is configured as shown in Table 2.

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

Table 2: UART Configuration

Ethernet

The Ethernet MAC in the block design is connected to a 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.

Please note that the RGMII delays in the Ethernet PHY need to be configured before the Ethernet interface can be used. In the reference design this is done in the Ethernet lwIP example.

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.

GPIOs

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

The FPGA firmware contains a 24-bit counter freely running at 50 MHz. The MSB of this counter is used to blink LED0# on FPGA pin M16 with a frequency of approximately 3 Hz. LED1# is driven by the same counter and blinks twice as fast as LED0#.

FPGA Pin	Signal	Function
M16	LED0#	GPIO 0, Blinking LED counter MSB
M17	LED1#	GPIO 1, Blinking LED counter MSB-1
L18	LED2#	GPIO 2, controlled by the GPIO controller
M18	LED3#	GPIO 3, controlled by the GPIO controller

Table 3: FPGA GPIO Configuration

XADC

A Xilinx XADC IP core instance is connected to the Microblaze system via an AXI bus, in order to monitor the junction temperature of the device. The temperature threshold for this module is configured to 85° Celsius. This is the maximum recommended operating condition for commercial devices.

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 [7].

For Enclustra Mars modules an Enclustra heat sink is available for purchase along with the product. It represents an optimal solution to cool the Mars AX3 FPGA module - it is low profile (less than 7 mm tall) and covers the whole module surface. It comes with a gap pad for the FPGA device and four screws to attach it to the module PCB. For details, please refer to the Enclustra website.

Getting Started

This section describes the steps required to configure the Mars AX3 FPGA module and Mars PM3 base board in order to run the example applications. The section includes information on how to:

- Mount the module and configure the base board
- Generate the FPGA bitstream
- Prepare the software workspace
- Run the software applications

The example applications, including the expected results of running the software, are described in detail in Section 3.7.

Essential Information

Warning!

Never mount or remove the Mars AX3 FPGA module to or from the Mars PM3 base board while the Mars PM3 base board is powered. Always remove or turn off the power supply before mounting or removing the Mars AX3 FPGA module.

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 read carefully the Mars AX3 FPGA module and Mars PM3 base board user manuals before proceeding.

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

Hardware Setup

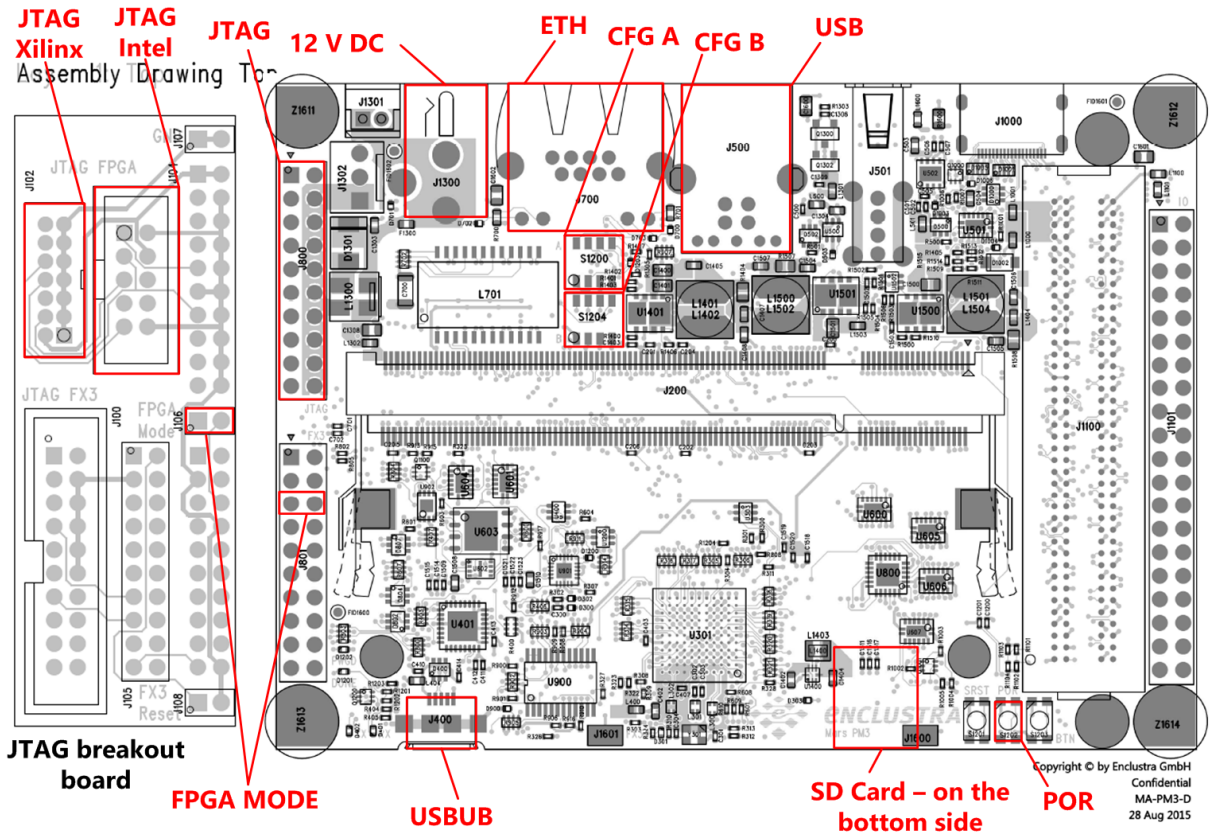

Figure 2: Mars PM3 Base Board Assembly Drawing (Top View)

Step	Description
1	Set the configuration DIP switches on the Mars PM3 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: ON, 2: ON, 3: OFF, 4: OFF] • CFG B = [1: ON, 2: OFF, 3: OFF, 4: OFF] For Mars PM3 base board revision 4 or older, set the configuration DIP switches as follows: • CFG = [1: OFF, 2: ON, 3: OFF, 4: OFF]
2	Mount the Mars AX3 FPGA module and the JTAG breakout board to the Mars PM3 base board.
3	Connect the micro USB cable between your computer and the Mars PM3 base board. Use the micro USB port labeled **USBUB** in Figure 2.
4	Connect the Xilinx Platform USB download cable to the JTAG connector of the JTAG PM3 breakout board (see label **JTAG Xilinx** in Figure 2). If a Xilinx JTAG breakout cable is used, use the pins labeled **JTAG** in Figure 2. Please refer to the Mars PM3 Base Board User Manual [5] for JTAG pins mapping.

Continued on next page...

Step	Description
5	Connect the 12 V DC power supply plug to the power connector of the Mars PM3 base board (see label **12 V DC** in Figure 2).
6	Open a terminal program on your computer (e.g. Tera Term) and open a serial port connection using the COM port labeled with the lower number from the two newly detected ports. For issues related to COM ports detection, refer to Section 5.4. Configure the UART parameters according to Section 2.2.6.

Table 4: Hardware Setup Step-By-Step Guide

FPGA Bitstream Generation

For a fast test of the provided software applications, the pre-generated bitstream included in the `binaries` directory may alternatively be used, therefore the steps described in this section may be skipped.

The `<base_dir>\binaries` directory includes bitstream files for any FPGA device that may be equipped on the module.

Step	Description
1	Edit the `fpga_part` variable in `scripts\settings.tcl` file, according to your FPGA device. This file includes module name and board information required for the project creation script. All settings, except for `fpga_part` should be left on default. The list of options for `fpga_part` is given in the comments within the Tcl file. Save the file after editing.
2	Start Xilinx Vivado 2017.4 and create the Mars AX3 FPGA module reference design project: 1. Click on the Tcl console at the bottom of the page and type: (a) `cd <base_dir>` (`<base_dir>` is the directory in which you extracted the archive contents). Note that you must use `/` for hierarchy separator, instead of `\`. (b) `source scripts/create_project.tcl` 2. Wait for completion
3	Run Synthesis, Implementation & Bitstream Generation in Vivado 2017.4: 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 SDK projects): 1. File → Export → Export Hardware 2. Enable Include bitstream checkbox If you want to save the .hdf file to another path than the default location: 3. Click on Choose Location → Select 4. Hit OK

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.

SDK Workspace Preparation

This section describes how to create and import the software applications. The steps are generic, and apply to all software examples provided in the release archive, along with this document.

The `<base_dir>\SdkExport` directory includes pre-generated hardware description files for any FPGA device that may be equipped on the module.

Step	Description
1	Start Xilinx SDK 2017.4 1. Select any workspace (e.g. `<base_dir>\workspace`)
2	Create a new board support package (BSP) 1. File → New → Board Support Package → Specify 2. In the New Hardware Project window: (a) For Project Name type hw_platform_0 (b) For Target Hardware Specification select the .hdf file you exported from Vivado, as described in Section 3.3. The default location used by Vivado is `<base_dir>\<vivado_proj_dir>\<project_name>.sdk\system_top.hdf` Alternatively, the pre-compiled hardware description file from `<base_dir>\SdkExport` may be used. (c) Hit Finish 3. In the New Board Support Package Project window: (a) For Project Name type standalone_bsp_0 (b) For Hardware Platform select hw_platform_0 (c) For Board Support Package OS select standalone (d) Hit Finish 4. In the Board Support Package Settings window (see Figure 3): (a) Enable the checkboxes corresponding to the drivers and libraries required in the BSP: lwip141 and xilmfs (b) Hit OK
3	Import the example projects into the workspace (see Figure 4): 1. File → Import... 2. Select General → Existing Projects into Workspace and hit Next 3. For Select root directory choose `<base_dir>\software` and hit OK 4. Enable checkbox Copy projects into workspace 5. Hit Finish
4	Build all projects 1. Hit Ctrl-B and wait for completion

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

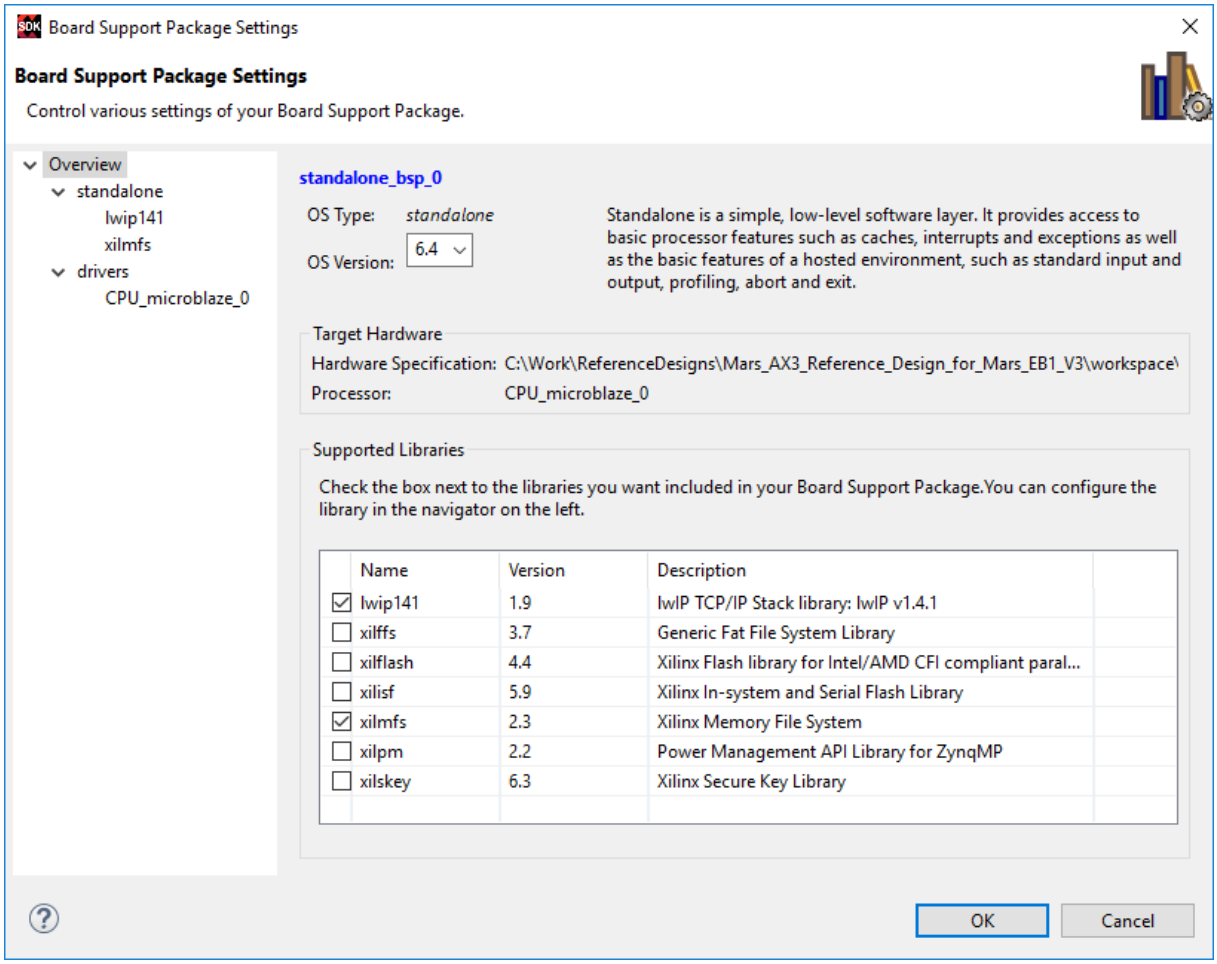

Figure 3: Board Support Package Settings

Warning!

For a successful build of the software examples the hardware project must be named "hw_platform_0" and that the BSP must be named "standalone_bsp_0".

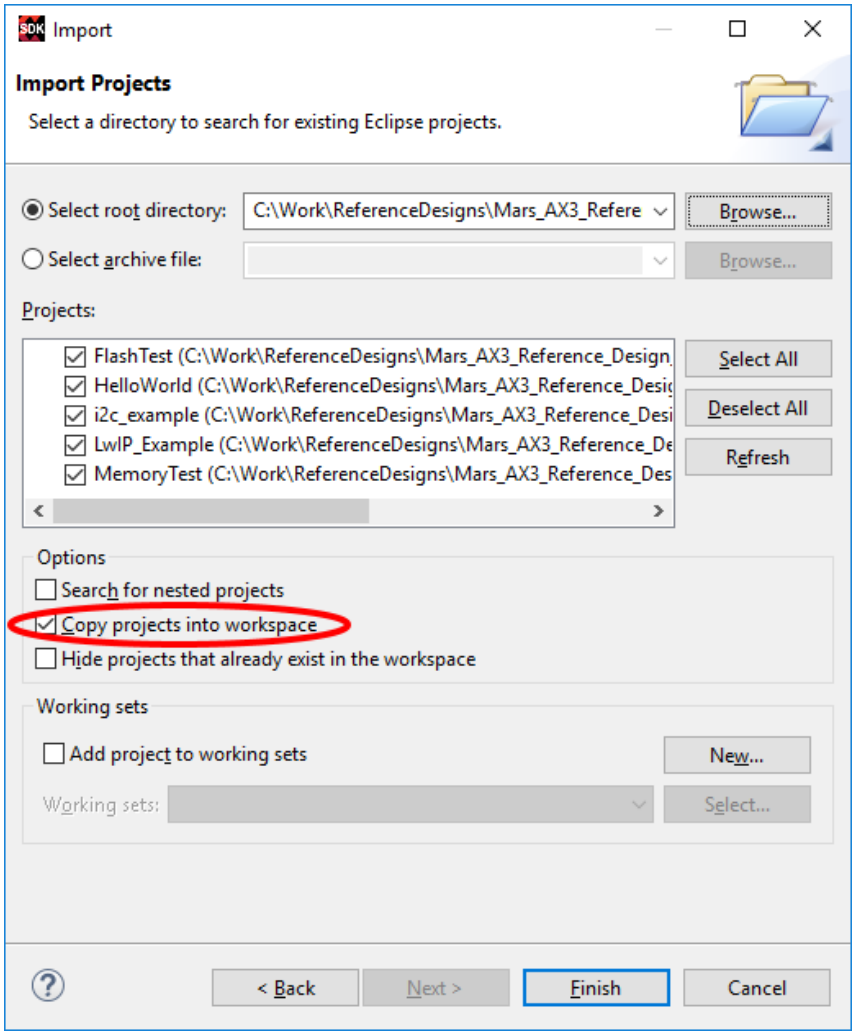

Figure 4: Importing Software Projects

Warning!

Please note that the software applications may not work properly if they are imported in another software version than the one specified in this document. A solution for such cases is creating a fresh software project with the sources provided in the reference design.

Warning!

Please make sure that during import process the checkbox for copying the projects into workspace has been enabled, otherwise the build step will fail due to incorrect file paths.

FPGA Programming

Step	Description
1	Open Xilinx SDK 2017.4: 1. Click on Xilinx Tools → Program FPGA 2. For Hardware Platform select hw_platform_0 3. For Bitstream field hit Search → select system_top.bit 4. For BMM/MMI field hit Search → select system_top.mmi 5. Hit Program The configuration is shown in Figure 5.
2	After the FPGA is successfully configured, the **DONE** LED should be lit. For Mars PM3 base board revision 4 or older, the **RDY** LED should be lit.

Table 7: FPGA Programming Step-By-Step Guide

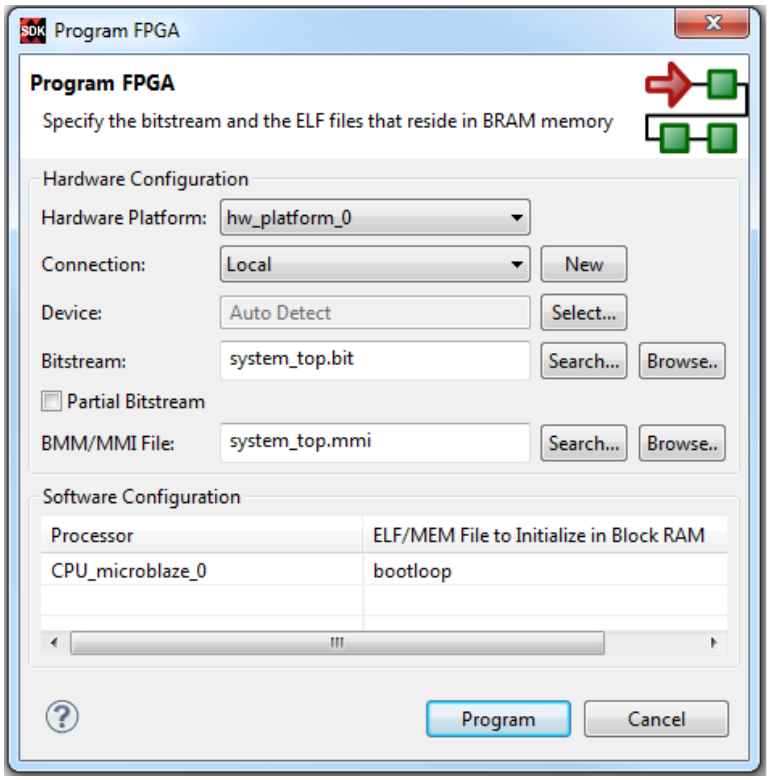

Figure 5: FPGA Programming Settings

Running Software Applications

This section describes how to run software applications on the Mars AX3 FPGA module. The steps are generic, and apply to all software examples provided in the release archive, along with this document.

In order to execute the applications, the hardware needs to be configured as described in Section 3.2.

Note that the FPGA must be programmed before running the software applications. Refer to Section 3.5 for details on FPGA programming.

Step	Description
1	Create a run configuration for the application in SDK 2017.4: 1. Run → Run Configurations... 2. Right-click Xilinx C/C++ application (GDB) and hit New or double-click on Xilinx C/C++ application (GDB) 3. Enter a run configuration name in the Name field (e.g. HelloWorld) 4. Target Setup tab (see Figure 6): (a) For Hardware Platform select hw_platform_0 (b) For CPU select CPU_microblaze_0 (c) In the Bitstream file field, hit Search... (d) Select download.bit and hit OK 5. Application tab: (a) In the Project Name field click browse and select an application (e.g. MemoryTest) (b) In the Application field click search and select an .elf file (c) Hit Apply
2*	**Optional** - for the lwIP example application, the following extra steps are required: 1. Select the Application tab (see Figure 7) 2. In the Data Files to download before launch section: (a) Click Add (b) Select the memory file system image from the lwIP example: `<base_dir>\software\lwip_example\image.mfs` (c) In Address field type 0x8f000000 and hit Open (d) Hit Apply
3	Start the application by clicking the Run button.

Table 8: Running an Application Step-By-Step Guide

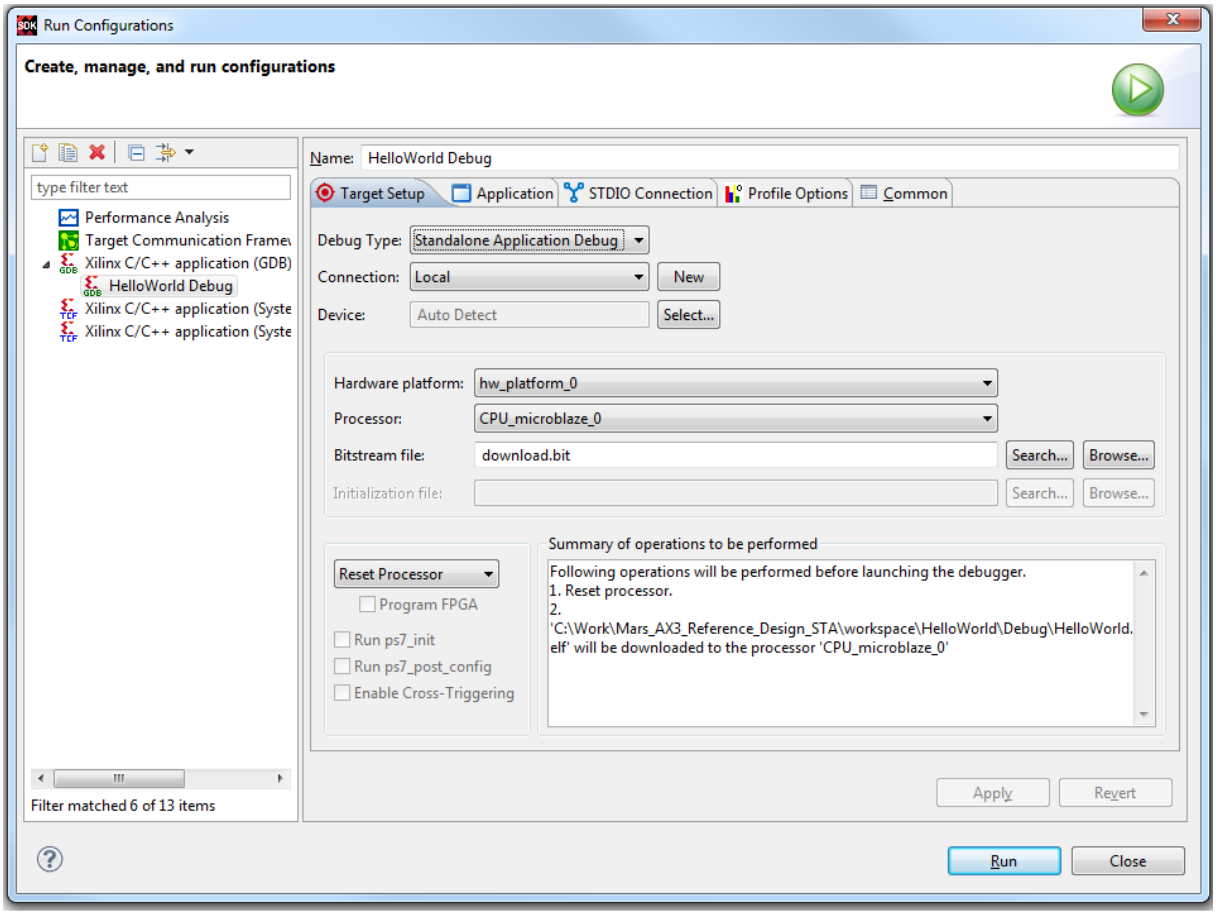

Figure 6: Run Configurations Settings

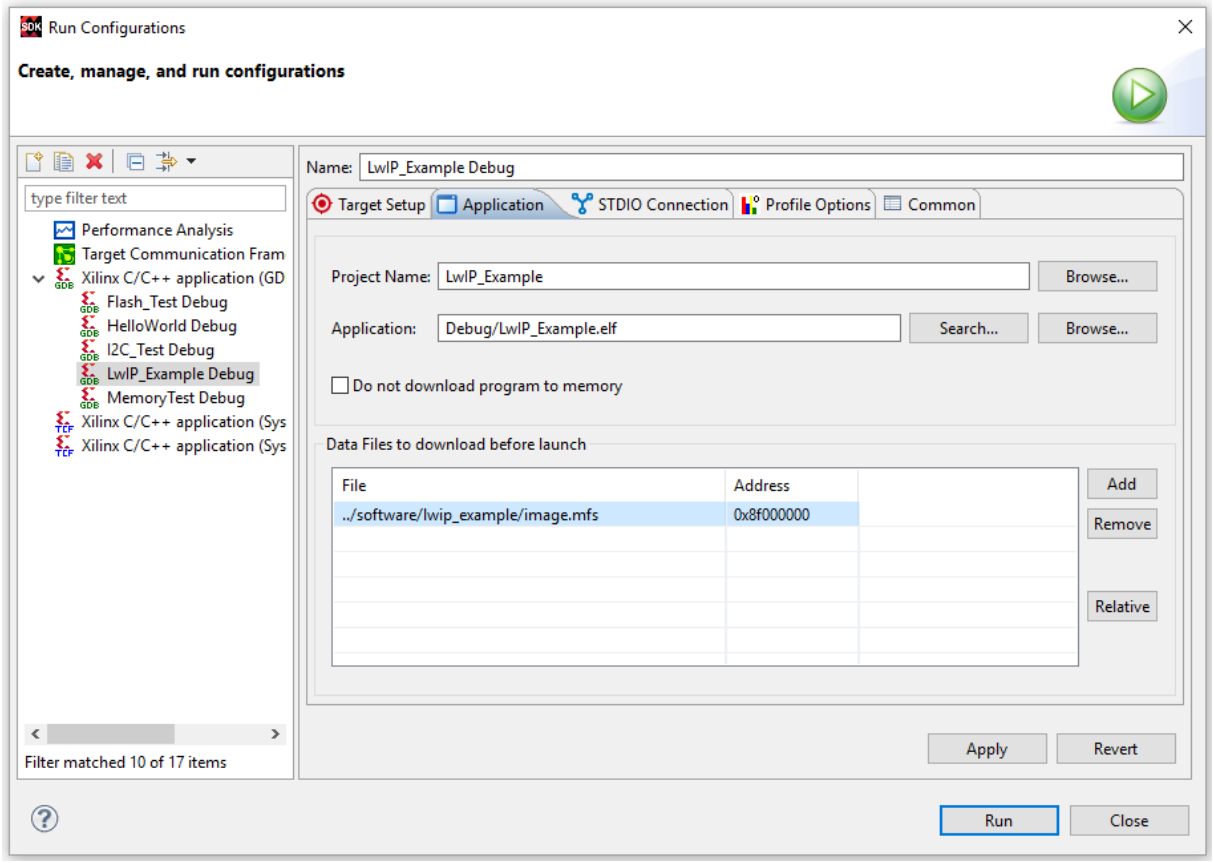

Figure 7: Run Configurations Settings for LwIP Application

Embedded Software

This section describes the software examples and the expected UART output while running these applications.

General

The Mars AX3 FPGA module reference design comes with a number of example applications, which show how to initialize the peripheral controllers and how to access the external devices. All of them are bare-metal applications that are executed in the DDR SDRAM memory.

The applications included are a Hello World example, an I2C test that reads and prints out the module and board configuration, a flash test which checks the QSPI flash memory available on the Mars AX3 FPGA module, a web server implementation example using lwIP networking stack, and a memory test that checks the DDR SDRAM memory available on the module.

The procedure of importing, creating and building the example applications is explained in Sections 3.4, 3.5 and 3.6.

Hello World Application

The Hello World application is a very simple application, which is used to demonstrate all the required steps for getting a bare-metal application running on the Mars AX3 FPGA module.

The Hello World application prints „Hello World x“ for twenty times, while x is incremented by one at every iteration. Figure 8 shows the UART output of the Hello World application.

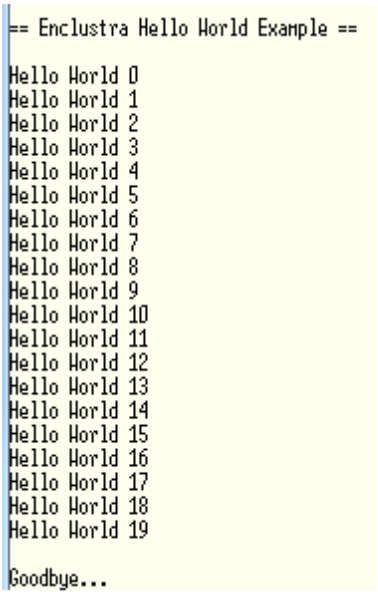

Figure 8: Hello World Application UART Output

I2C Example Application

The I2C example application demonstrates the configuration and use of the I2C controller on the Mars AX3 FPGA module.

The I2C example application reads the module configuration data from the secure EEPROM on the Mars AX3 FPGA module, as well as voltage information from the system monitor present on the Mars PM3 base board.

The module configuration includes: module type, serial number, DDR SDRAM and flash memory sizes, and first MAC address. The MAC addresses can be used by the user to configure the Ethernet MAC.

Figure 9 shows the UART output of the I2C application.

```
== Enclustra I2C test ==

EEPROM:
   Module type                  Mars AX3
   Revision                     2
   Serial number                106345
   MAC Address 0                20:B0:F7:03:3E:D2
   FPGA type                    Xilinx Artix-7 XC7A50T
   FPGA speed grade             1
   Temperature grade            Industrial
   Power grade                  Normal
   Ethernet port count          1
   Ethernet speed               Gigabit
   Real-time clock equipped     No
   USB device port count        0
   DDR3 RAM size (MB)           256
   SPI flash size (MB)          16

System Monitor:
   VCC_MAIN       Voltage = 12155 mV
   VCC_5V         Voltage = 5000 mV
   VCC_3V3        Voltage = 3312 mV
   VCC_IO         Voltage = 3290 mV
   VCC_1V2        Voltage = 1230 mV
   VMON_P41       Voltage = 1475 mV
   VCC_OUT        Voltage = 1795 mV


== End of test ==
```

Figure 9: I2C Example Application UART Output

Memory Test Application

The memory test application performs several tests on the DDR memory present on the Mars AX3 FPGA module. A quick simple test and a detailed full test are executed.

The simple test is run on a bigger part of the memory and checks that the sequential incrementing of the address and writing values to the memory works as expected. The full test uses several writing patterns on two different parts of the memory, then reads the values and compares the results. This test is performed on a smaller part of the memory.

Figure 10 shows the UART output of the memory test application. Note that the starting address and the memory size configured for the test depend on the module and on the application settings.

Please note that depending on the DDR memory controller speed, memory cache enable and on the test size, the memory test may take several minutes to complete.

```
== Enclustra Memory Test ==

  Testing 4MB @ Address 0xC0100000 (full test)

  Loop 1/1:
    Stuck Address       : ........ok
    Random Value        : ok
    Compare XOR         : ok
    Compare SUB         : ok
    Compare MUL         : ok
    Compare DIV         : ok
    Compare OR          : ok
    Compare AND         : ok
    Sequential Increment: ......................ok
    Solid Bits          : ........ok
    Block Sequential    : ........ok
    Checkerboard        : ........ok
    Bit Spread          : ........ok
    Bit Flip            : ........ok
    Walking Ones        : ........ok
    Walking Zeroes      : ........ok
    8-bit Writes        : ok
    16-bit Writes       : ok

  Testing 255MB @ Address 0xC0100000 (quick test)

  Loop 1/2:
    Sequential Increment: ....................ok

  Loop 2/2:
    Sequential Increment: ....................ok

== Test finished, no errors occurred ==
```

Figure 10: Memory Test Application - UART Output Example (addresses and test sizes may vary)

Flash Test Application

The flash test application is based on the Xilinx SPI controller driver example and performs a write/read-back test of one sector starting at a specific address of the QSPI flash memory. The application initializes the SPI and the interrupt controller and includes read, write and erase functions that show how to access the flash memory on the Mars AX3 FPGA module using the Xilinx SPI controller.

Figure 11 shows the UART output of the flash test application.

```
-- SPI Flash Test --

WARNING: This test will erase 1 Sector at address 0xF00000

Press Enter to continue or 'n' to abort
   writing to SPI Flash
   reading from SPI Flash

SPI Flash successfully tested!

-- SPI Flash Test Complete --
```

Figure 11: Flash Test Application UART Output

Ethernet LwIP Application

The Ethernet lwIP application represents a basic example of running a web server on the Mars AX3 FPGA module using the open-source TCP/IP networking stack lightweight IP (lwIP).
The example is based on the Xilinx lwIP example (see Xilinx Application Note [3]), modified for compatibility with the Ethernet PHY used on the Mars AX3 FPGA module.

In order to do this change, one of the files generated automatically in the board support package (xaxiemacif_physpeed.c) has been modified and copied to the project sources.

Before launching the Ethernet lwIP application, the network adapter of the computer needs to be configured as described below.
It is recommended to have a second network adapter for local connections and keep the current network setup on the default adapter, in order to avoid reconfiguring the connection parameters.

Computer Setup

Step	Description
1	Connect a network cable (RJ45) between your computer and the Mars PM3 base board. Use the connector marked with label **ETH** in Figure 2.
2	Open Control Panel → Network and Internet → Network and Sharing Center
3	Click on the Local Area Connection → Properties (see Figure 12) 1. Disable all connections except Internet Protocol Version 4 (TCP/IPv4) 2. Click on Internet Protocol Version 4 (TCP/IPv4) → Properties (a) Set the IP address to 192.168.1.1 (b) Set the subnet mask to 255.255.255.0 (c) Hit OK

Table 9: Computer Network Setup Step-By-Step Guide

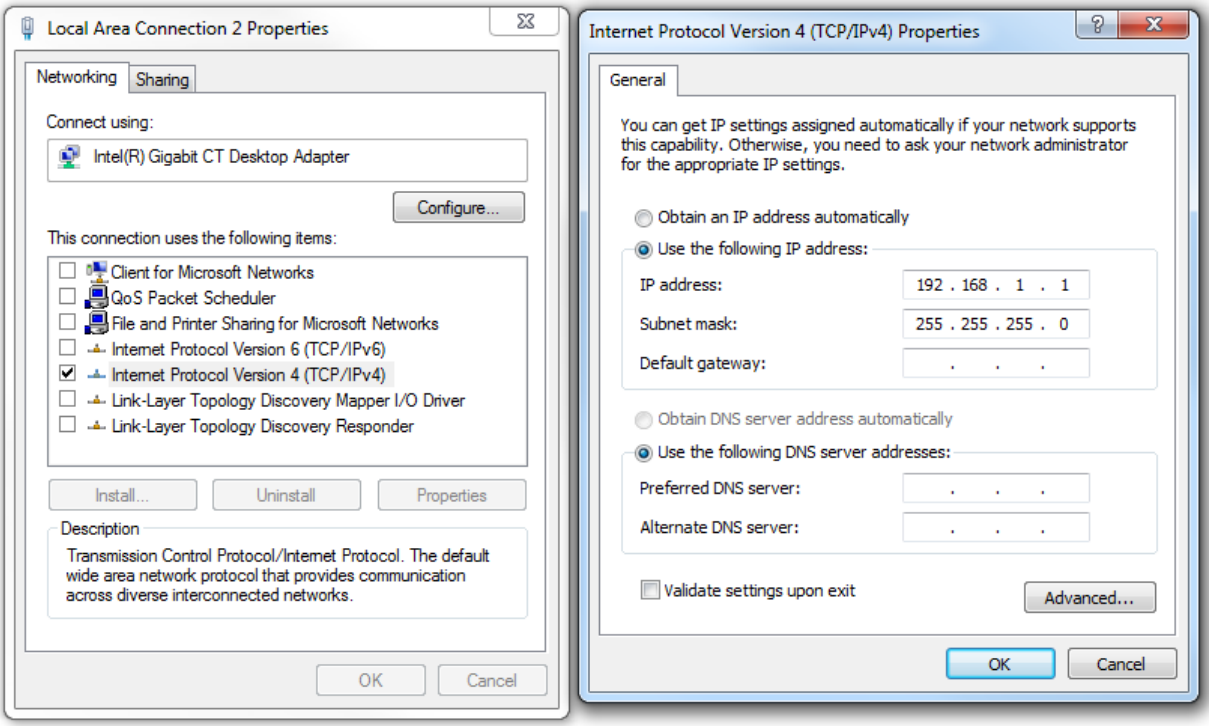

Figure 12: Network Adapter Configuration for LwIP Application

Please note that a memory file system (MFS) image must be downloaded to the RAM for the application to run. Section 3.6 describes the required steps to run the lwIP Ethernet application.

After having the computer setup, as well as the run configurations done, two tests can be performed using the lwIP application.

The testing procedure and the expected UART output are further presented.

Testing the LwIP Web Server

Step	Description
1	Open a web browser
2	Type http://192.168.1.10/ in the address bar A basic web page indicating the current state of the LEDs on the Mars AX3 FPGA module will be loaded in the browser. The user can toggle the LEDs using a button in the web page. Figure 13 illustrates the lwIP application web page.
3	Click the Toggle LEDs button, in order to change the state of the LEDs.

Table 10: Testing the LwIP Web Server Step-By-Step Guide

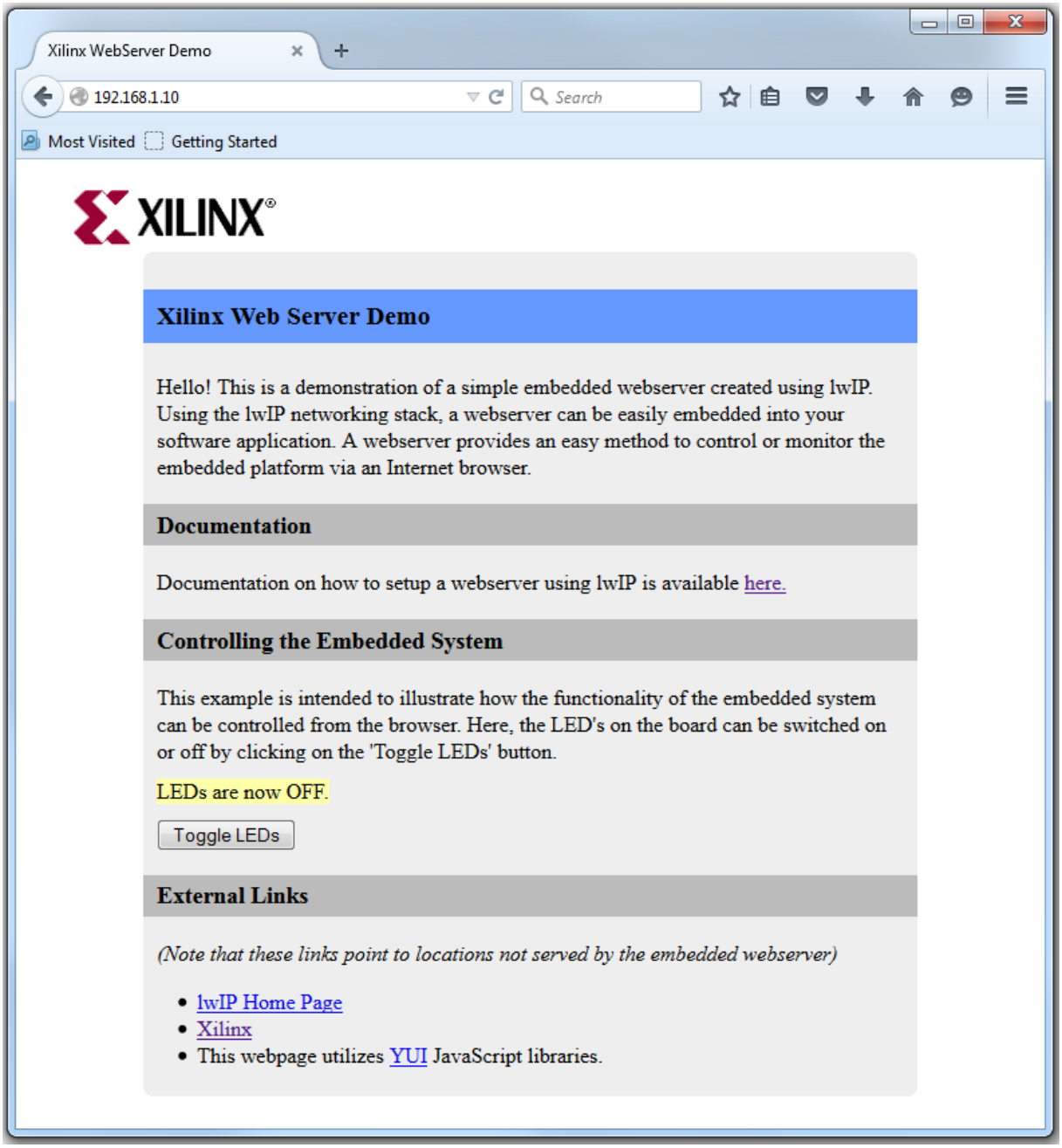

Figure 13: LwIP Application Web Page

Testing the LwIP Telnet Echo Server

Step	Description
1	Open a Telnet program (e.g. PuTTY) and configure the following parameters (see Figure 14): 1. For Host Name type 192.168.1.10 2. For Protocol select Telnet 3. For Port type 7 4. Click Open
2	Each character sent to the Mars AX3 FPGA module is immediately sent back (each character appears twice in the Telnet terminal).

Table 11: Testing the LwIP Telnet Echo Server Step-By-Step Guide

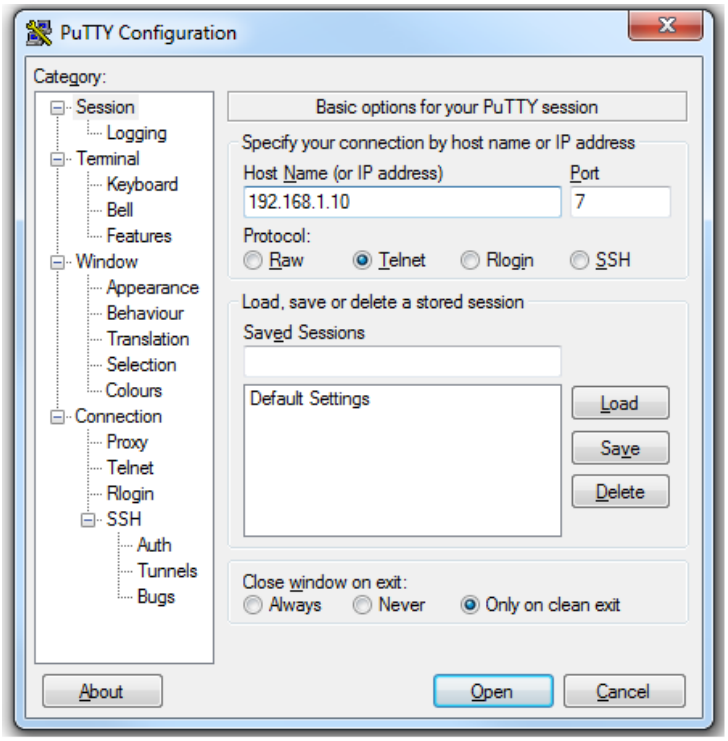

Figure 14: PuTTY Configuration for Telnet Communication

Figure 15 displays the UART output of the Ethernet lwIP application, corresponding to web page loading action and to Telnet echo calls.

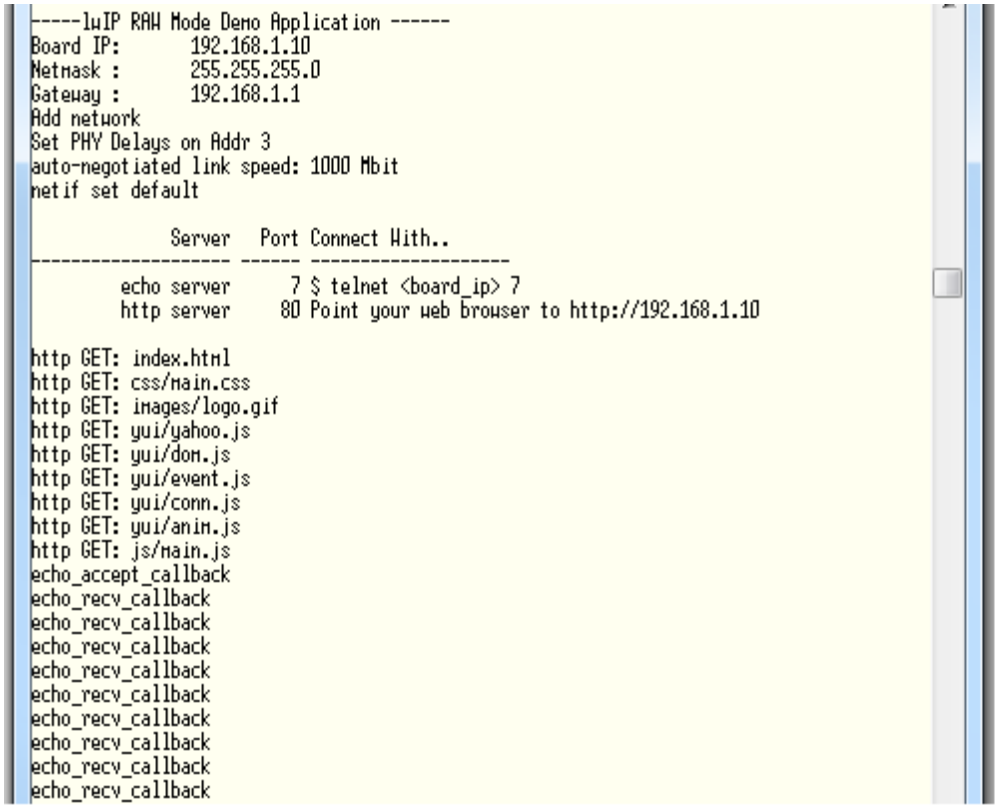

Figure 15: Ethernet LwIP Application UART Output

Boot Configurations

QSPI Flash Boot

This section describes how to program a bitstream or a boot image (.bit/.bin) to the QSPI flash, and how to boot from the QSPI flash memory on the Mars AX3 FPGA module.

Generating the Image Files

This section describes how to create a Microblaze boot image. Some of the examples in the reference design can be used to generate the boot image. The details on this procedure are further presented. If the FPGA bitstream does not contain a Microblaze system, steps 1 and 2 from this section can be skipped.

If a software application is configured to run from the FPGA on-chip memory, a simple boot image for the Microblaze system including the software application code can be created and written to the QSPI flash memory. 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.

In cases where the application is mapped to the DDR memory, a boot loader is necessary. Please refer to Xilinx Answer Records AR# 47909 and AR# 55026 and to Xilinx documentation for details on this procedure.

The examples included in the reference design are all mapped to the SDRAM memory and must be remapped for on-chip memory execution in order to be used to generate a Microblaze boot image including the software code for the processor. Please note that not all programs can fit into the on-chip memory - in these situations the code and data must be reduced to fit the size configured in Vivado project.

Step	Description
1	Change the memory for program data storage and code execution: 1. Right click on the application in SDK → Generate Linker Script 2. Select the CPU local memory for code, data, heap and stack sections (see Figure 16) 3. Hit Generate and re-build the software project
2	Create the FPGA bitstream (which includes the software code) from Xilinx SDK 2017.4 (see Figure 17): 1. Click on Xilinx Tools → Program FPGA 2. For Hardware Platform select hw_platform_0 3. For Bitstream field hit Search → select system_top.bit 4. For BMM/MMI field hit Search → select system_top.mmi 5. In the Software Configuration section select the application .elf file to be written into the FPGA Block RAM 6. Hit Program This procedure will start initializing the bitstream with the application ELF data and then program the FPGA with the resulting bitstream: `<base_dir>\workspace\hw_platform_0\download.bit` Please note that when selecting another .elf file, the `download.bit` file must be renamed or removed, so that SDK generates a new bitstream.

Continued on next page...

Step	Description
3	**Optional** - Create the boot image (.bin) to be written to the QSPI flash Xilinx SDK or Enclustra MCT support both .bit and .bin formats for the QSPI flash image, Vivado Hardware Manager supports only .mcs and .bin formats. There are two options for creating the .bin image to be written to the QSPI flash:
(a)	Program the QSPI flash from SDK 2017.4 (the .bin file will automatically be generated in `<base_dir>\workspace\hw_platform_0\cache` directory). Please refer to Section 4.1.3 for details on programming the flash.

(b)*	**Optional** - alternatively, the image can be created using bootgen command:
	
	1. Create a .bif file (`<base_dir>\workspace\hw_platform_0\bootimage.bif`) containing the following text:
	
	`the_ROM_image:`
	`{`
	`<base_dir>\workspace\hw_platform_0\download.bit`
	`}`
	
	2. Run bootgen command from Windows command line (if the command is split on multiple lines, use ^ as a separator):
	
	`> bootgen -arch fpga -image ^`
	`<base_dir>/workspace/hw_platform_0/bootimage.bif -w -o ^`
	`<base_dir>/workspace/hw_platform_0/BOOT.bin -interface spi`

Table 12: Generating the Image Files for QSPI Flash Boot Mode Step-by-Step Guide

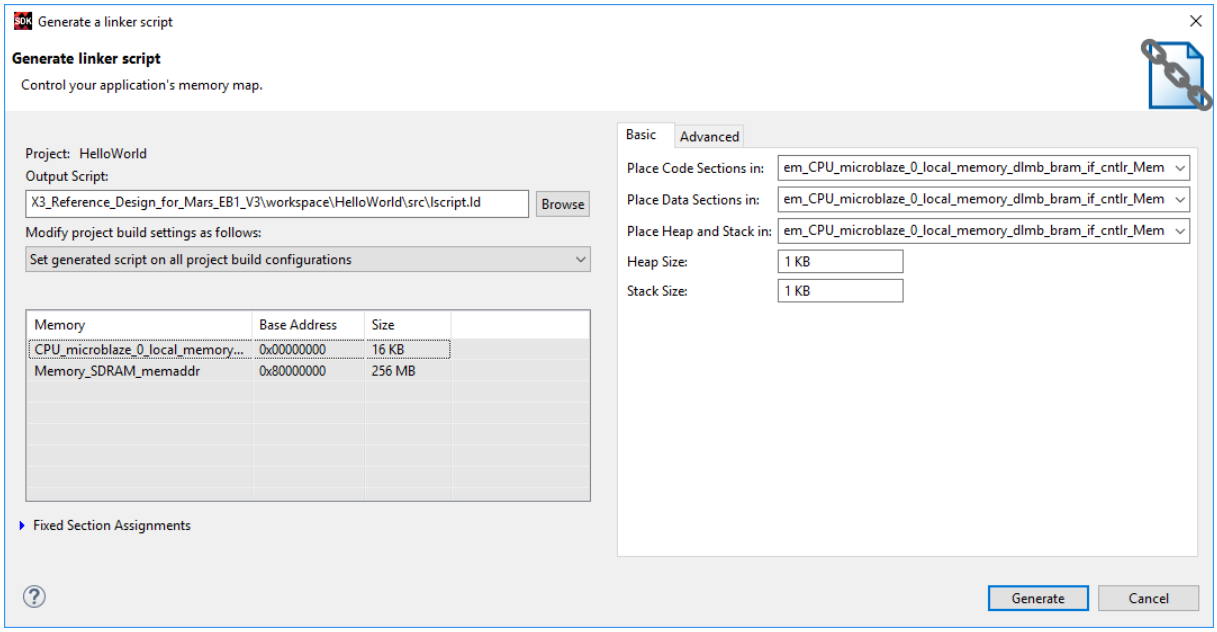

Figure 16: Generate the Linker Script for an SDK Application

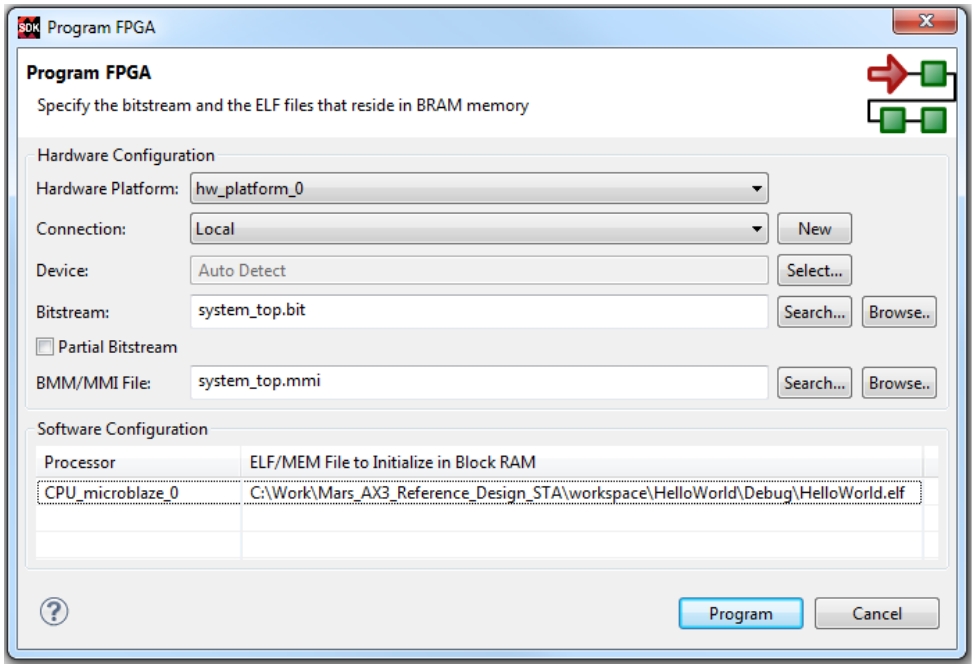

Figure 17: Create Microblaze Boot Image Settings

Preparing the Hardware

Step	Description
1	Remove the power supply from the Mars PM3 base board (see label **12 V DC** in Figure 2).
2	Set the configuration DIP switches on the Mars PM3 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: ON, 2: ON, 3: OFF, 4: OFF] • CFG B = [1: OFF, 2: OFF, 3: OFF, 4: OFF] For Mars PM3 base board revision 4 or older, enable the QSPI flash boot mode by removing the jumper between pins 5 and 6 of J801 of Mars PM3 base board or the FPGA MODE jumper located on the JTAG breakout board (see label **FPGA MODE** in Figure 2).
3	Connect the 12 V DC power supply plug to the power connector of the Mars PM3 base board (see label **12 V DC** in Figure 2).

Table 13: Preparing the Hardware for QSPI Flash Boot Mode Step-by-Step Guide

Programming the QSPI Flash

Step	Description
Note	Vivado and SDK 2017.4 tools do not have built-in support for Winbond QSPI flash present on the Mars AX3 FPGA module revisions 1 and 2. For these modules, the method described at step 3 must be used for QSPI flash programming. For modules starting with revision 3, Xilinx tools can be used, as described at steps 1 and 2.

Continued on next page...

Step	Description
1	**For revision 3 and newer**: Open Xilinx SDK 2017.4: 1. Xilinx Tools → Program Flash 2. In Program Flash Memory window (see Figure 18): (a) For Image File select the download.bit file generated as described in Section 4.1.1 (b) For Flash Type select the type according to the Mars AX3 FPGA Module User Manual [4]. This is in most cases s25fl512s-qspi-x1_x2_x4. (c) Enable Blank Check after erase checkbox (d) Hit Program and wait for completion
2*	**For revision 3 and newer**: **Optional** - if SDK returns errors during flash programming or if the system does not boot properly, another option is to use Vivado 2017.4 to program the QSPI flash. 1. Flow → Open Hardware Manager 2. Click on Open target → Auto Connect 3. Right click on the corresponding FPGA device in the left bar → Add Configuration Memory Device (see Figure 19) (a) For Select Configuration Memory Part choose the memory part according to the Mars AX3 FPGA Module User Manual [4] This is in most cases s25fl512s-qspi-x1_x2_x4. (b) Hit OK 4. In Program Configuration Memory Device window (see Figure 20): (a) For Configuration file select the boot image (.bin) generated as described in Section 4.1.1 (b) In Program Operations section: • For Address Range select Entire Configuration Memory Device • Enable checkboxes Erase, Blank Check, Program and Verify • Hit OK and wait for completion The settings in the pictures are for reference only. Note that the memory part and the configuration file must be selected according to your application.
3	**For any hardware revision**: Use Enclustra Module Configuration Tool (MCT) [6] to program the QSPI flash. Before programming the QSPI flash from MCT, make sure the hardware configuration on the Mars PM3 base board is done according to Section 4.1.2. Use USB 3.0 type B port (see label **USB** in Figure 2) to program the QSPI flash using MCT.

Table 14: Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide

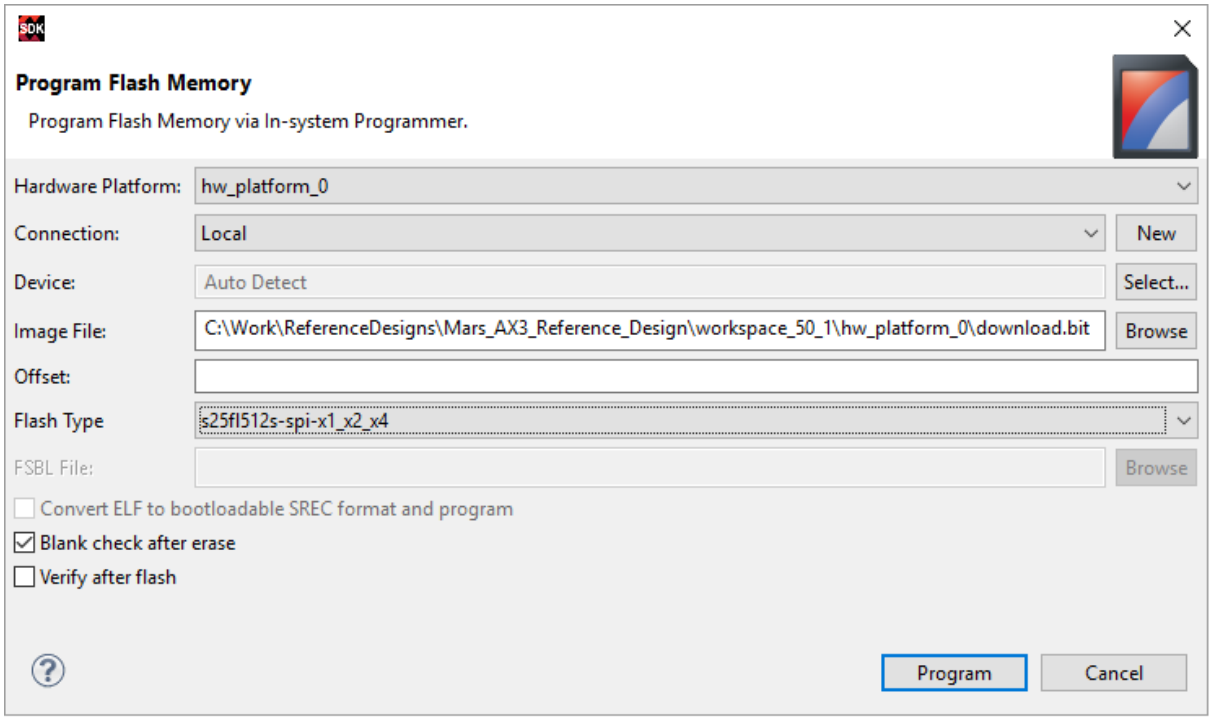

Figure 18: QSPI Flash Programming Settings in SDK

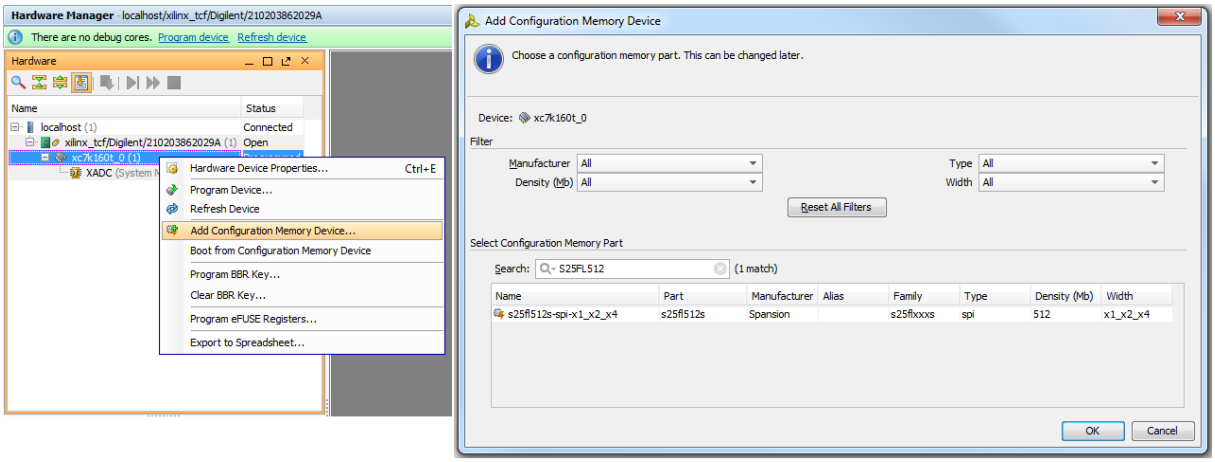

Figure 19: QSPI Flash Programming Settings in Vivado - Adding the Memory Device

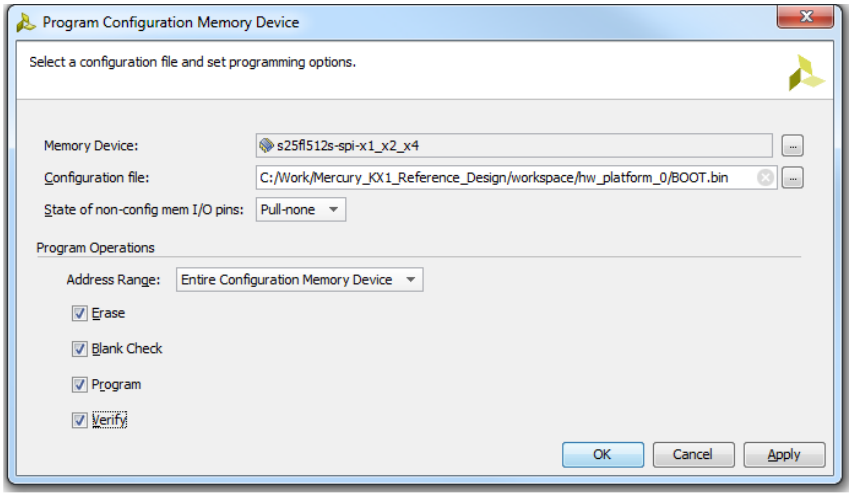

Figure 20: QSPI Flash Programming Settings in Vivado

Booting from the QSPI Flash

Step	Description
1	Check that the hardware configuration is done according to Section 4.1.2.
2	Press the power-on reset button (see label **POR** in Figure 2) and release it after a second. For Mars PM3 base board revision 4 or older, press the power-on reset button labeled **LD**.

Table 15: Booting from the QSPI Flash Step-by-Step Guide

Troubleshooting

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

SDK Runtime Exceptions

- In order to avoid runtime exceptions issued by SDK, always stop running processes with the red Terminate button. In debug mode, use this button before resetting or disconnecting the Mars AX3 FPGA module.
- If the SDK reports runtime exceptions while downloading a program to memory, the following steps should be followed:
  1. Close SDK
  2. Shutdown any javaw, eclipse, xmd or hw_server processes in Windows Task Manager
  3. Power off the Mars AX3 FPGA module
  4. Restart SDK and power on the Mars AX3 FPGA module

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. Remove the USB connection and power supply from the Mars PM3 base board and close SDK
  3. Reconnect the USB and power supply and start SDK again
  4. Reboot the computer if the problem persists

UART Connection Issues

- If the computer is not able to recognize the USB UART on the Mars PM3 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.

List of Figures

List of Tables

References

[1] Vivado Design Suite User Guide, Embedded Processor Hardware Design, UG898, Xilinx, 2016
[2] Vivado Design Suite Tutorial, Embedded Processor Hardware Design, UG940, Xilinx, 2014
[3] Xilinx XAPP1026, LightWeight IP Application Examples, Xilinx, 2014
[4] Mars AX3 FPGA Module User Manual
→ Ask Enclustra for details
[5] Mars PM3 Base Board User Manual
→ Ask Enclustra for details
[6] Enclustra Module Configuration Tool (MCT)
→ Ask Enclustra for details
[7] Enclustra Modules Heat Sink Application Note
→ Ask Enclustra for details