

Cosmos XZQ10 System Board

Reference Design User Manual

Purpose

The purpose of this document is to present to the user the overall view of the Cosmos XZQ10 system board reference design and to provide the user with a step-by-step guide to the complete Xilinx® SoC design flow used for the Cosmos XZQ10 system board.

Summary

This document first gives an overview of the Cosmos XZQ10 system board reference design and then guides through the complete Xilinx SoC design flow for the Cosmos XZQ10 system board in the getting started section. In addition, the internals and the boot options of the Cosmos XZQ10 system board reference design are described.

Product Information	Code	Name
Product	CO-XZQ10	Cosmos XZQ10 System Board

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-450-001	01	25.08.2020

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

License

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	Processing System (PS)	5
2.1.1	Clocks	5
2.1.2	PS DDR3L SDRAM	5
2.1.3	SD Card	6
2.1.4	I2C	6
2.1.5	Quad SPI Flash Controller	6
2.1.6	UART	6
2.1.7	Ethernet	7
2.1.8	USB	7
2.1.9	GPIOs	7
2.2	Programmable Logic (PL)	8
2.2.1	PL DDR3L SDRAM	8
2.2.2	GPIOs	8
2.2.3	XADC	8
3	Getting Started	9
3.1	Essential Information	9
3.2	FPGA Bitstream Generation	11
3.3	Vitis Workspace Preparation	12
3.4	Running Software Applications	14
4	Boot Configurations	16
4.0.1	Generating the Image File	16
4.1	QSPI Flash Boot	16
4.1.1	Preparing the Hardware	16
4.1.2	Programming the QSPI Flash	17
4.1.3	Booting from the QSPI Flash	22
4.2	SD Card Boot	22
4.2.1	Generating the Image Files	22
4.2.2	Preparing the Hardware	23
4.2.3	Programming the SD Card	23
4.2.4	Booting from the SD Card	23
5	Troubleshooting	24
5.1	Vivado Issues	24
5.2	Vitis Issues	24
5.3	JTAG Connection Issues	24
5.4	UART Connection Issues	24
5.5	QSPI Boot Issues	25

1 Overview

1.1 Introduction

The Cosmos XZQ10 system board reference design demonstrates a system using the Cosmos XZQ10 system board in combination with the Cosmos XZQ10 system board. It presents the basic configuration of the device and contains a guided getting started tutorial.

A troubleshooting section is included at the end of the document, to help the user solve potential issues related to board connectivity and/or system functionality.

Please note that the features presented in the reference design depend on the employed base board, therefore some features may not be available in certain hardware configurations.

An introduction to the Xilinx tools is provided by the documents below:

- Vivado Design Suite User Guide, Embedded Processor Hardware Design [1]
- Zynq® 7000 All Programmable SoC Embedded Design Tutorial [2]

More information on the Cosmos XZQ10 system board and the Cosmos XZQ10 system board can be retrieved from their respective user manuals [3] [4].

The following directory structure applies to the XZQ10 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 XZQ10 variant are released on the XZQ10 Reference Design Github page.

1.2 Prerequisites

- IT
 - A computer with a microSD card slot (optional¹) running Windows 10 64-bit (or later)
- Software
 - Xilinx Vivado 2020.1 WebPack, Evaluation, Design or System Edition (check the Cosmos XZQ10 System Board User Manual [3] for details on device support in Xilinx tools)
 - Xilinx Vitis IDE
 - Enclustra Module Configuration Tool (MCT) [5] (optional²)
 - A terminal emulation program (e.g. Tera Term)
- Hardware
 - An Enclustra Cosmos XZQ10 system board
 - An Enclustra Cosmos XZQ10 system board
- Accessories
 - A 12 V DC power supply
 - A standard micro USB cable
 - A Xilinx JTAG programmer (e.g. Platform Cable USB II) (optional³)

¹Only required for SD card boot mode

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

³Any FTDI device present on Enclustra hardware can be configured to Xilinx JTAG mode using the Enclustra MCT software [5].

2 Reference Design Description

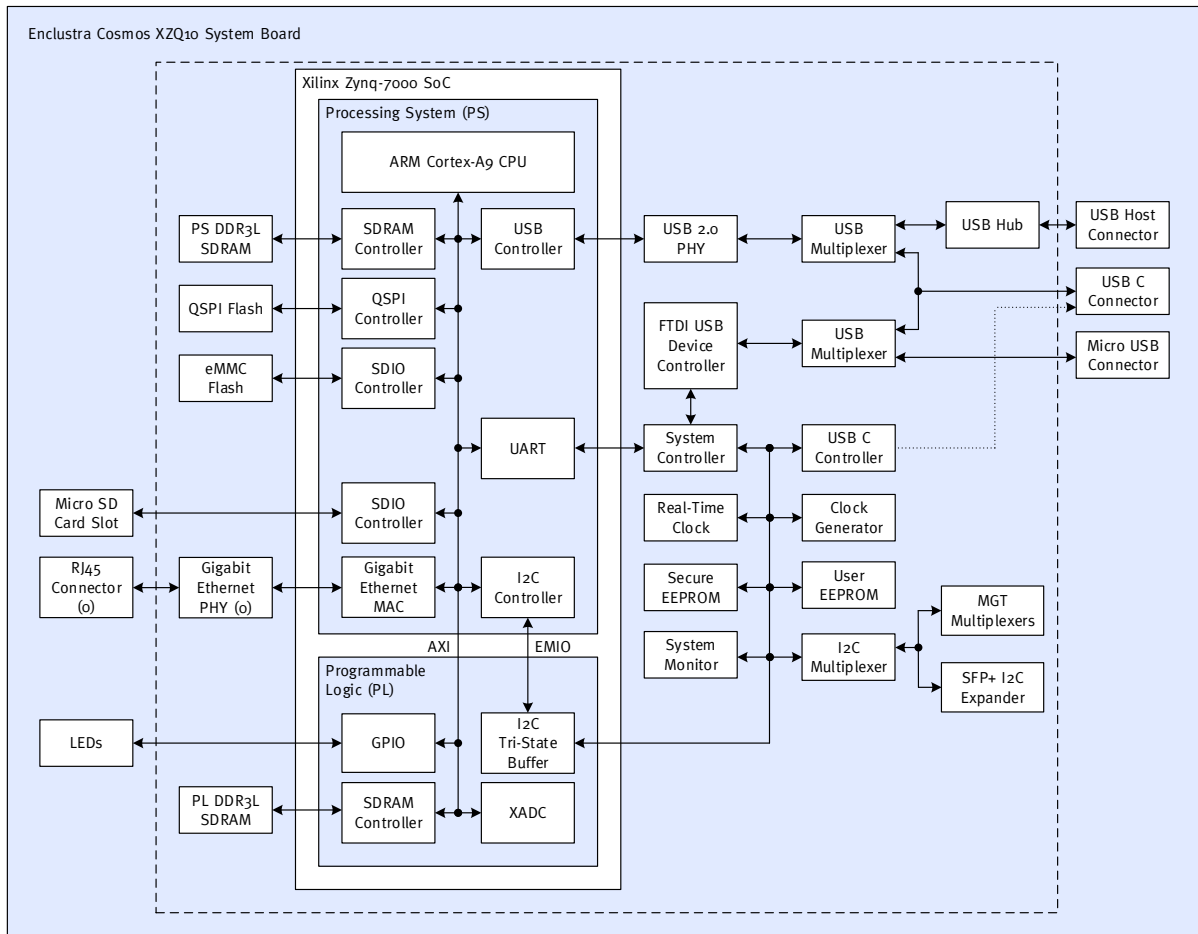


Figure 1: Hardware Block Diagram

2.1 Processing System (PS)

2.1.1 Clocks

The PS input clock frequency is configured to 33.33 MHz, while the CPU clock is configured to its corresponding maximum CPU frequency. The maximum CPU clock performance depends on the device speedgrade and package. Beside that a 50 MHz and a 100 MHz clock are exported from PS to the PL.

These clocks can be modified in the settings of the processing system in Vivado.

2.1.2 PS DDR3L SDRAM

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

The DDR settings in the Zynq system must be configured according to the Cosmos XZQ10 System Board User Manual [3].

2.1.3 SD Card

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

To allow the Cosmos XZQ10 system board to boot from the SD card, the hardware configuration on the Cosmos XZQ10 system board must be done according to Section 4.2.2.

Please note that the SD pins are shared between the PS and the PL.

Warning!

Because the MIOs 40..45 are connected to FPGA pins AA13, AA12, AB17, AB16, AC17, AC16 in parallel, make sure the FPGA pins are in high impedance state before driving the PS SD pins and vice versa.

2.1.4 I2C

The I2C controller I2C0 is configured to the EMIO pins. For available devices on the I2C bus refer to the Cosmos XZQ10 System Board and Cosmos XZQ10 System Board User Manual [3] [4]. An I2C Application Note is available as well providing sample code and more details about using I2C on Enclustra hardware [7].

2.1.5 Quad SPI Flash Controller

The quad SPI flash controller is connected to MIO 1..6 in Single Slave Select mode. MIO 2..6 pins are shared between NAND flash and QSPI flash on the Cosmos XZQ10 system board. Please refer to the Cosmos XZQ10 System Board User Manual [3] for details about flash programming and usage.

To allow the Cosmos XZQ10 system board to boot from the QSPI flash, the hardware configuration on the Cosmos XZQ10 system board must be done according to Section 4.1.1.

2.1.6 UART

The UART0 is mapped to MIO 46..47 pins and connected to the FTDI controller on the Cosmos XZQ10 system board. The UART is configured as shown in Table 2.

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

Table 2: UART Configuration

2.1.7 Ethernet

The Ethernet MAC ENET 0 is mapped to MIO 16..27 and MIO 52..53 pins and is connected to the Microchip (Micrel) KSZ9031 Ethernet PHY on the Cosmos XZQ10 system board using an RGMII interface. The PHY can be configured via the MDIO management interface on PHY address 3. Please note that the RGMII delays in the Ethernet PHY need to be configured before the Ethernet interface can be used. Further details on PHY delay configuration are given in the Gigabit Ethernet Application Note by Enclustra [8].

2.1.8 USB

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

Depending on the required USB mode, the settings in the system controller and the DIP switches on the Cosmos XZQ10 system board must be configured correctly. Please refer to the Cosmos XZQ10 System Board User Manual [4] for details.

2.1.9 GPIOs

The unused MIO pins from the PS are available as GPIOs. For details on the MIO assignment refer to the Multiplexed I/O (MIO) Pins section in the Cosmos XZQ10 system board User Manual [3]. Check the connectivity of the MIOs that provide user functionality with the Cosmos XZQ10 system board User Manual [4]

2.2 Programmable Logic (PL)

2.2.1 PL DDR3L SDRAM

The PL DDR3L SDRAM memory runs at its corresponding maximum frequency at a voltage of 1.35 V by default. The DDR settings can be modified in the Memory Interface Generator (MIG) IP core and must be configured according to the Cosmos XZQ10 system board User Manual [3].

2.2.2 GPIOs

A Xilinx GPIO controller in the PL is connected to the PS via an AXI bus. The least significant bit of the PL GPIO bus is connected to LED0# 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 DIO_LED# with a frequency of approximately 3 Hz.

FPGA Pin	Signal	Function
V19	DIO_LED#	Blinking LED counter MSB
V18	DII_LED#	GPIO 1, controlled by the PL GPIO controller

Table 3: FPGA Firmware I/O Configuration

2.2.3 XADC

A Xilinx XADC IP core instance is connected to the PS via an AXI bus, in order to monitor the temperature of the device. The temperature threshold for the FPGA is configured to its maximum allowed temperature.

The constraints provided in the reference design enable FPGA bitstream power-down, when the temperature increases above the threshold. In this case, the PL will be reset, while the ARM processor will still be running.

Depending on the user application, the Cosmos XZQ10 system board may consume more power than can be dissipated without additional cooling measures; always make sure the SoC is adequately cooled by installing a heat sink and/or providing air flow. Temperature control and monitoring is very important in a complex design.

Information that may assist in selecting a suitable heat sink for the Cosmos XZQ10 system board can be found in the Enclustra Modules Heat Sink Application Note [9].

3 Getting Started

This section describes the steps required to configure the Cosmos XZQ10 system board and Cosmos XZQ10 system board in order to run a simple HelloWorld example application. The section includes information on how to:

- Mount the module and configure the Cosmos XZQ10 system board
- Generate the PL bitstream
- Prepare the software workspace
- Run a software application

3.1 Essential Information

Warning!

The provided reference design is based on the Cosmos XZQ10 System Board revision 3 and should not be used with older revisions.

Warning!

Please read carefully the Cosmos XZQ10 system board user manual before proceeding.

Warning!

Depending on the user application, the Cosmos XZQ10 system board may consume more power than can be dissipated without additional cooling measures; always make sure the SoC is adequately cooled by installing a heat sink and/or providing air flow.

Warning!

Please make sure that a single JTAG adapter is connected to the base board and enabled at a given moment, otherwise the development tools may report errors during JTAG connecting attempts.

Note that when Enclustra MCT [5] is used for SoC configuration or flash programming, all other tools that may be connected to the FTDI device (e.g. Vivado Hardware Manager, Vitis, UART terminal) must be closed.

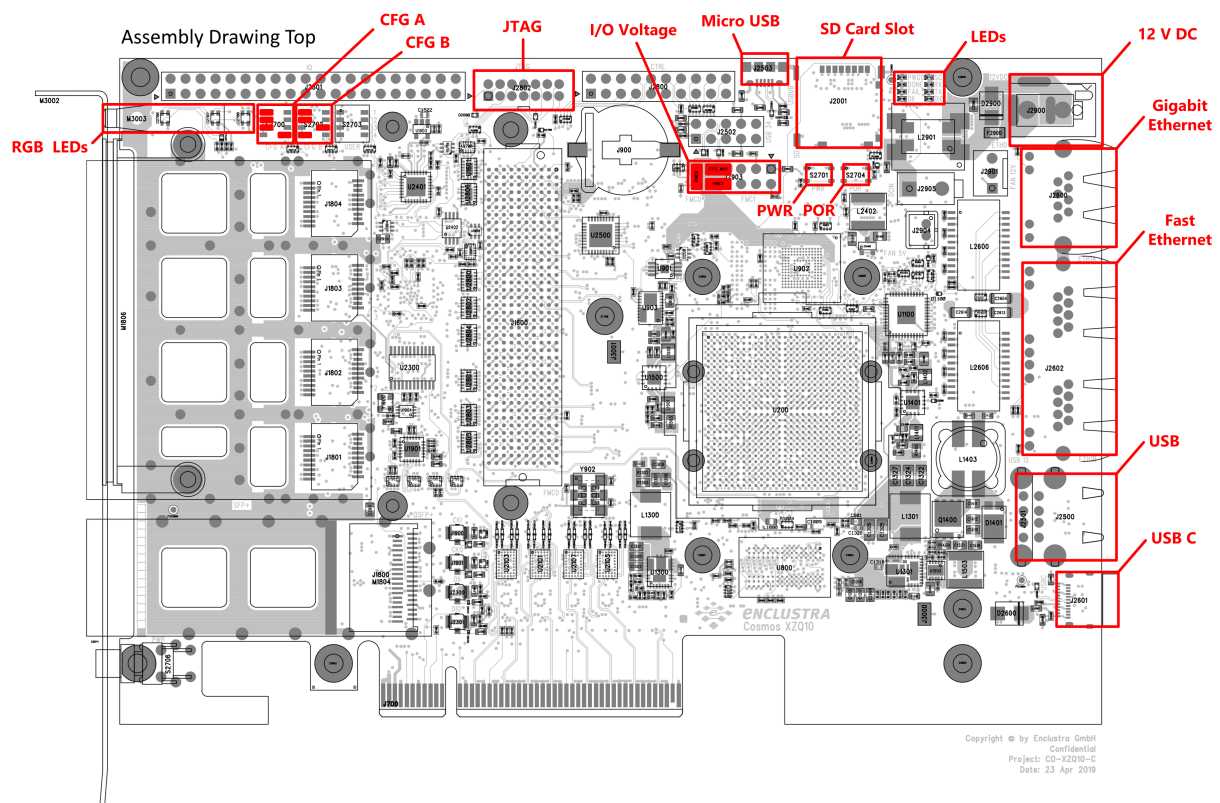


Figure 2: Cosmos XZQ10 System Board Assembly Drawing (Top View)

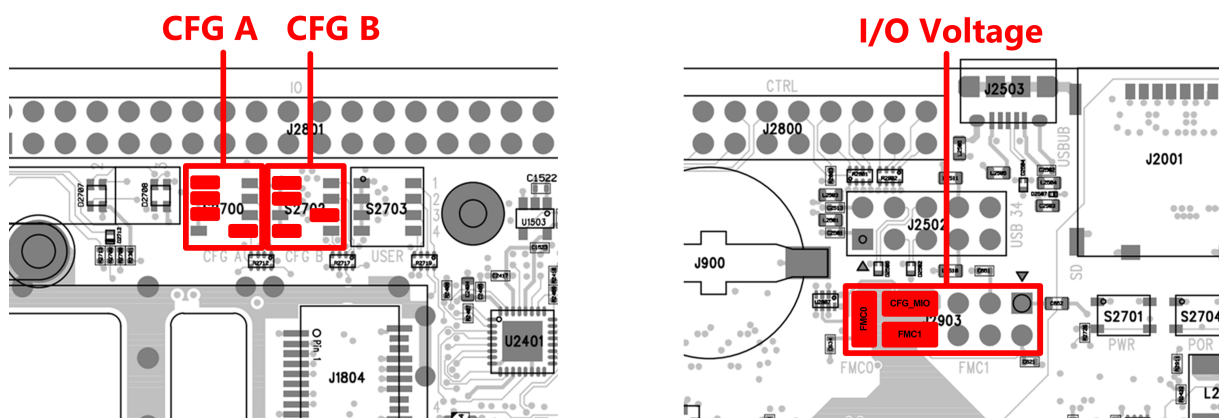


Figure 3: Cosmos XZQ10 System Board Assembly Drawing (CFG Switches and I/O Jumpers)

Step	Description
1	<p>Set the I/O voltage jumpers on the Cosmos XZQ10 system board according to label I/O Voltage in Figure 3 (the jumpers are marked with red rectangles):</p> <ul style="list-style-type: none"> • CFG_MIO (VCC_CFG_MIO) = 1.8 V (Jumper Pos. 7-9) • FMC0 (VCC_ADJ_FMC0) = 1.8 V (Jumper Pos. 11-12) • FMC1 (VCC_ADJ_FMC1) = 1.8 V (Jumper Pos. 8-10)
2	<p>Set the configuration DIP switches on the Cosmos XZQ10 system board as follows (see labels CFG A and CFG B in Figure 2):</p> <ul style="list-style-type: none"> • CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON] • Depending on the available USB cable, the CFG B DIP switches must be configured as follows: <ul style="list-style-type: none"> • For micro USB cable: CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF] • For type C USB cable: CFG B = [1: ON, 2: ON, 3: ON, 4: OFF]
3	<p>Connect the USB cable between your computer and the Cosmos XZQ10 system board. Use the micro USB port labeled Micro USB in Figure 2 or the type C USB port labeled USB C, according to your hardware configuration.</p>
4	<p>Connect the Xilinx Platform Cable USB to the JTAG connector of the Cosmos XZQ10 system board (see label JTAG in Figure 2).</p> <p>Alternatively, JTAG over USB may be used. For this, the FTDI device on the Cosmos XZQ10 system board must be configured to Xilinx JTAG mode using Enclustra MCT [5].</p> <p>Details on the Xilinx JTAG mode configuration are presented in the Cosmos XZQ10 System Board User Manual [4]</p>
5	<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 2.1.6.</p>

Table 4: Hardware Setup Step-By-Step Guide

3.2 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 XZQ10 variant is released on the XZQ10 Reference Design Github page.

Step	Description
------	-------------

Continued on next 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 Cosmos XZQ10 system board 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
3	<p>Run Synthesis, Implementation & Bitstream Generation in Vivado 2020.1:</p> <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	<p>Export the hardware system information (required for the Vitis IDE):</p> <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

3.3 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 XZQ10 variant is released on the XZQ10 Reference Design Github page.

Step	Description
1	<p>Start the Vitis IDE 2020.1</p> <ol style="list-style-type: none"> 1. Select any workspace (e.g. <base_dir>\workspace)
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. Cosmos_XZQ10_XZQ10 (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 3.2. 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 ps7_cortexa9_0 (i) Check the "Generate boot components" checkbox, if it is not enabled yet. Vitis will then automatically generate the binaries for the FSBL. (j) Hit Finish and wait for completion (k) 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) For Project Name type a description for the new application e.g. "HelloWorld" (b) Check the "Use default location" checkbox, if it is not enabled yet. (c) For the System project select "Create New.." and use the default naming e.g. "HelloWorld_system" (d) Hit Next (e) In the "Select a platform from repository" tab choose the Platform that you have generated previously (f) Hit Next (g) For the Domain choose the standalone_domain and the C Language. (h) Hit Next and wait for the tool to proceed (i) Select the HelloWorld (or any another) template (j) Hit Finish and wait for completion (k) Build the application by pressing Ctrl-B and wait for completion

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

3.4 Running Software Applications

This section describes how to run software applications on the Cosmos XZQ10 system board. 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) Enable ps7_cortexa9_0 checkbox(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 Figures 4 and 5):<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 Cosmos_XZQ10_XZQ10.bit and hit OK(d) For FPGA Device and PS Device, use Auto Detect option(e) Uncheck the "Use FSBL flow for initialisation"(f) In the Initialization File field, hit Search...(g) Select ps7_init.tcl and hit OK(h) Enable checkboxes Reset entire system, Program FPGA, Run ps7_init and Run ps7_post_config(i) Hit Apply
2	<p>Make sure the Hardware is configured according to Section ??:</p> <ul style="list-style-type: none">→ Connect the 12 V DC power supply plug to the power connector of the Cosmos XZQ10 system 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, executes the required initialization for the PS, powers up the PL, configures the PL with the specified bitstream and downloads the application program to the ARM processor.</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 PL is successfully configured, the **DONE** LED should be lit. When the application is running successfully, the output of the HelloWorld application should appear on the UART console.



Figure 4: Run Configurations Settings - Application Tab



Figure 5: Run Configurations Settings - Target Setup Tab

4 Boot Configurations

Once a software application has been developed and tested, this can be used to build a boot image for the module.

The boot image contains the FSBL, the bitstream for programming the PL and the software bare-metal application.

In order to use a software application for the boot image, the code must be mapped for execution from the external DDR memory. If the program is mapped to the on-chip memory, it will overwrite the boot loader during execution.

For a fast test of the boot configurations, the pre-generated .bin images may be used for boot, instead of rebuilding the image. You need to select the file corresponding to the Cosmos XZQ10 system board variant. Pre-generated binaries for any XZQ10 variant are released on the XZQ10 Reference Design Github page.

4.0.1 Generating the Image File

Step	Description
1	Create the boot image from Xilinx Vitis 2020.1 (see Figure 7): <ol style="list-style-type: none">1. Right click on the application in the Project Explorer2. Select Create Boot Image → Create Image An image will be created in <workspace>\HelloWorld_ide\bootimage\BOOT.bin.

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

Beside the methods presented in this section, there are additional methods how a boot device can be programmed. Please refer to the Enclustra Build Environment's User Documentation for details [6].

4.1 QSPI Flash Boot

4.1.1 Preparing the Hardware

Step	Description
1	Power off the Cosmos XZQ10 system board (see label 12 V DC in Figure 2).
2	Disconnect all USB cables from the Cosmos XZQ10 system board.

Continued on next page...

Step	Description
3	<p>Enable the QSPI flash boot mode by setting the configuration DIP switches on the Cosmos XZQ10 system board as follows (see labels CFG A and CFG B in Figure 2):</p> <ul style="list-style-type: none"> • CFG A = [1: ON, 2: OFF, 3: OFF, 4: ON] • Depending on the available USB cable, the CFG B DIP switches must be configured as follows: <ul style="list-style-type: none"> • For micro USB cable: <ol style="list-style-type: none"> 1. Set CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF] 2. Connect a USB cable to the micro USB port on the Cosmos XZQ10 system board (see label Micro USB in Figure 2) • For type C USB cable: <ol style="list-style-type: none"> 1. Set CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF] 2. Connect a USB cable to the USB type C port on the Cosmos XZQ10 system board (see label USB C in Figure 2) 3. Set CFG B = [1: ON, 2: ON, 3: ON, 4: OFF]
4	Power on the Cosmos XZQ10 system 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

In order to program Zynq-7000 devices a modification to the standard FSBL generated during platform creation is needed. The created FSBL is limited to only running the initialization (ps7_init()). For more details on this please refer to the answer record by Xilinx AR70548.

Step	Description
1	<p>Modify the created Platform project</p> <ol style="list-style-type: none"> 1. In the Platform project open platform.scr by double-clicking 2. Click on "Board Support Settings" under the standalone ps7_cortexa9_0 tab 3. Click on "Modify BSP Settings..." in the right hand window 4. Enable the xilffs library (See figure 6)

Continued on next page...

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

Table 10: Creating modified FSBL for QSPI programming Mode Step-by-Step Guide

Code snippet 1: Modification to FSBL

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

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

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

Step	Description
1	<p>Program the boot image from Xilinx Vitis 2020.1 (see Figure 7):</p> <ol style="list-style-type: none"> 1. Right click on the application in the Project Explorer 2. Select Program Flash Vitis will fill out the fields for the selected application automatically. 3. Replace the automatically found FSBL with the FSBL.elf from the generated FSBL project 4. For Flash Type select qspi-x4-single 5. 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 Manager 2. Click on Open target → Auto Connect 3. Right click on the corresponding SoC device in the left bar → Add Configuration Memory Device (see Figure 8) <ol style="list-style-type: none"> (a) For Select Configuration Memory Part choose the memory part according to the Cosmos XZQ10 System Board User Manual [3], part type single. This is in most cases s25fl512s-1.8v-qspi-x4-single. (b) Hit OK 4. In Program Configuration Memory Device window (see Figure 9): <ol style="list-style-type: none"> (a) For Configuration file select the boot image generated as described in Section 4.0.1 (b) For Zynq FSBL select the FSBL binary generated with the Platform as described in Section 3.3 (c) 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 <p>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.</p>

Continued on next page...

Step	Description
3*	<p>Optional - alternatively, Enclustra Module Configuration Tool (MCT) [5] can be used to program the QSPI flash.</p> <p>The procedure implies setting another boot mode than QSPI during flash programming, so that the SoC does not try to boot while the flash is being programmed. The other boot mode in this case is eMMC boot, therefore the method will be successful only if the eMMC flash is not programmed.</p> <ol style="list-style-type: none"> 1. Close all other tools that may be connected to the FTDI device (Vivado Hardware Manager, Vitis, UART terminal). 2. Set the power switch of the Cosmos XZQ10 system board to OFF/PCIe (see label PWR SW in Figure 2). 3. Disconnect all USB cables from the Cosmos XZQ10 system board. 4. Set CFG A = [1: ON, 2: OFF, 3: OFF, 4: ON] 5. Set CFG B = [1: OFF, 2: ON, 3: ON, 4: OFF] 6. Connect a USB cable to the micro USB port on the Cosmos XZQ10 system board (see label USBUB in Figure 2) 7. Set CFG B = [1: OFF, 2: OFF, 3: ON, 4: ON] 8. Set the power switch of the Cosmos XZQ10 system board to ON (see label PWR SW in Figure 2). 9. Perform QSPI flash programming in MCT and close MCT 10. After programming, remove the power supply from the Cosmos XZQ10 system board (see label 12 V DC in Figure 2). 11. Disconnect all USB cables from the Cosmos XZQ10 system board and set the power switch of the Cosmos XZQ10 system board to OFF/PCIe. 12. Reconnect the USB cable and disconnect and reconnect the UART terminal.

Table 11: Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide

	Name	Version	Description	
<input type="checkbox"/>	libmetal	2.0	Libmetal Library	
<input type="checkbox"/>	lwip211	1.1	Lwip211 library: lwIP (light weight IP) is an open sour...	
<input type="checkbox"/>	openamp	1.5	OpenAmp Library	
<input checked="" type="checkbox"/>	xilffs	4.2	Generic Fat File System Library	
<input type="checkbox"/>	xilflash	4.7	Xilinx Flash library for Intel/AMD CFI compliant paral...	
<input type="checkbox"/>	xilif	5.14	Xilinx In-system and Serial Flash LibraryWARNING: X...	
<input type="checkbox"/>	xilloader	1.0	Xilinx Versal Platform Loader Library	
<input type="checkbox"/>	xilplmi	1.0	Xilinx versal Platform Loader and Manager Interface ...	
<input type="checkbox"/>	xilpm	3.0	Platform Management API Library for ZynqMP and ...	
<input type="checkbox"/>	xilrsa	1.5	Xilinx RSA Library to access RSA and SHA software al...	
<input type="checkbox"/>	xilsem	1.0	Xilinx Versal Soft Error Mitigation Library	
<input type="checkbox"/>	xilsky	6.8	Xilinx Secure Key Library supports programming efu...	

Figure 6: Enable xilffs in Vitis

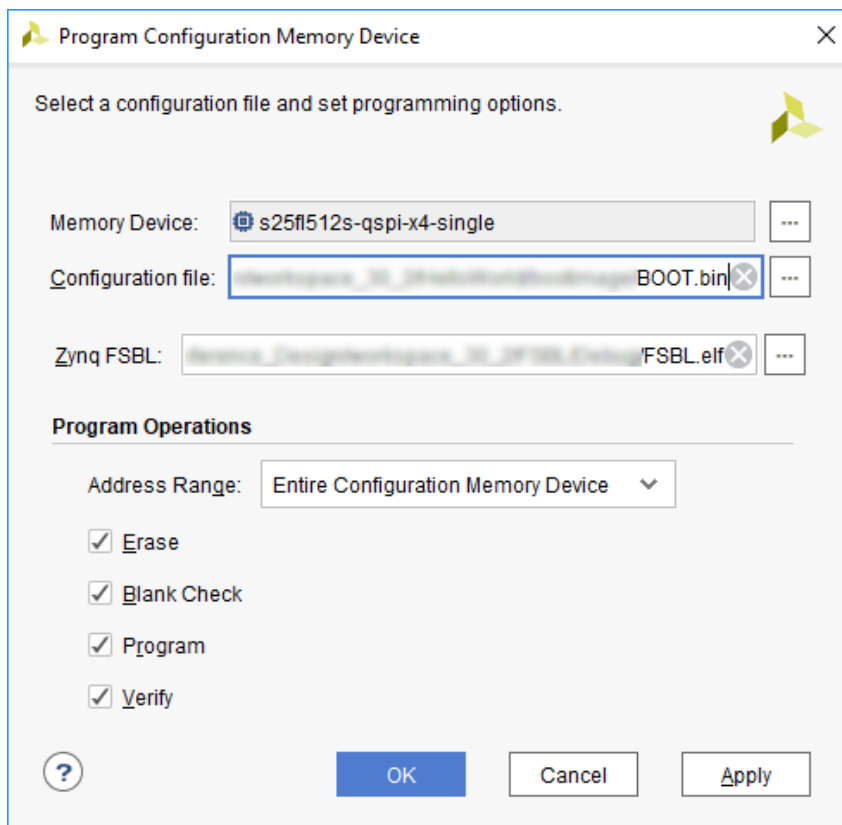


Figure 9: QSPI Flash Programming Settings in Vivado

Warning!

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

4.1.3 Booting from the QSPI Flash

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

Table 12: Booting from the QSPI Flash Step-by-Step Guide

4.2 SD Card Boot

4.2.1 Generating the Image Files

Please refer to Section 4.0.1 describing the steps required to generate a boot image.

4.2.2 Preparing the Hardware

Step	Description
1	Power off the Cosmos XZQ10 system board (see label 12 V DC in Figure 2).
2	Disconnect all USB cables from the Cosmos XZQ10 system board.
3	<p>Enable the SD card boot mode (default) by setting the configuration DIP switches on the Cosmos XZQ10 system board as follows (see labels CFG A and CFG B in Figure 2):</p> <ul style="list-style-type: none">• CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON]• Depending on the available USB cable, the CFG B DIP switches must be configured as follows:<ul style="list-style-type: none">• For micro USB cable:<ol style="list-style-type: none">1. Set CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]2. Connect a USB cable to the micro USB port on the Cosmos XZQ10 system board (see label Micro USB in Figure 2)• For type C USB cable:<ol style="list-style-type: none">1. Set CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]2. Connect a USB cable to the USB type C port on the Cosmos XZQ10 system board (see label USB C in Figure 2)3. Set CFG B = [1: ON, 2: ON, 3: ON, 4: OFF]

Table 13: Preparing the Hardware for SD Card Boot Mode Step-by-Step Guide

4.2.3 Programming the SD Card

Step	Description
1	<p>Write the Xilinx SD card boot image to a FAT32 formatted SD card</p> <ol style="list-style-type: none">1. Insert the SD card into the SD card slot of your computer2. Copy the boot image generated for your application to your SD card (directly in the root directory). <p>Note that the name of the image must be preserved.</p>

Table 14: Programming the SD Card for SD Card Boot Mode Step-by-Step Guide

4.2.4 Booting from the SD Card

Step	Description
1	Insert the SD card into the SD card slot of the Cosmos XZQ10 system board (see label SD Card in Figure 2).
2	Power on the Cosmos XZQ10 system board (see label 12 V DC in Figure 2).

Table 15: Booting from the SD Card Step-by-Step Guide

5 Troubleshooting

5.1 Vivado Issues

- If the changes in the block design (including licenses for special IPs) are not propagated into implementation, open the Hierarchy tab in Vivado and regenerate the block design files:
 1. Right click on the block design file (.bd)
 2. Click on Reset Output Products → Reset
 3. Click on Generate Output Products → Generate → OK
- During block design generation Vivado reports a critical warning (CRITICAL WARNING: [PSU-1] Parameter : PCW_UIPARAM_DDR_DQS_TO_CLK_DELAY_0 has negative value -0.026 . PS DDR interfaces might fail when entering negative DQS skew values.) which can be safely ignored for ZYNQ 7000 PS DDR interfaces. For more details please refer to this Xilinx forum post.

5.2 Vitis Issues

- If the platform generation in Vitis is not successful or the generated platform is not selectable for applications:
 1. Close Vitis
 2. Delete the workspace folder
 3. Restart Vitis and try creating the platform again.
- If Vitis shows the warning "There's no DDR_1 in the HW design. MMU translation table marks 32 GB DDR..." please check if more than 2GB PS DDR4 memory should be available. For detailed information please check the Xilinx Answer Records and Forum about this warning.

5.3 JTAG Connection Issues

- If the JTAG cable is not detected, the following steps should be followed:
 1. Make sure that the hardware configuration is made according to Section ??
 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 Cosmos XZQ10 system board can be retrieved from the Cosmos XZQ10 system 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 Cosmos XZQ10 system 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 Cosmos XZQ10 system board:
 1. Check that the USB cable is connected properly
 2. Check that the FTDI VCP drivers are installed
 - (a) Open Device Manager
 - (b) Universal Serial Bus controllers → USB Serial Converter A/B → Properties → Advanced tab → enable Load VCP checkbox

- (c) Reboot the computer if the COM port is still not detected
- 3. Reinstall the FTDI drivers if the problem persists
- If the computer does not output any character in the terminal program:
 - 1. Check that the FTDI device is set to UART mode:
 - (a) Download and open FT_Prog utility (this is a third party tool offered by the FTDI company to configure FTDI devices)
 - (b) DEVICES → Scan and Parse
 - (c) Check that for Port A and B the RS232 UART property is true
 - 2. Check that the baud rate for the UART in the block design matches the baud rate set in the terminal program
 - 3. If the UART used is mapped to the EMIO pins in the PS, resetting the ARM core will not suffice. Reprogramming the PL is necessary, as the UART lines go through PL.
 - 4. Make sure that Enclustra MCT software is not open. After closing it, unplug and plug in again the USB cable corresponding to the UART communication.

5.5 QSPI Boot Issues

- If the Cosmos XZQ10 system board 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	Cosmos XZQ10 System Board Assembly Drawing (Top View)	10
3	Cosmos XZQ10 System Board Assembly Drawing (CFG Switches and I/O Jumpers)	10
4	Run Configurations Settings - Application Tab	15
5	Run Configurations Settings - Target Setup Tab	15
6	Enable xilffs in Vitis	20
7	QSPI Flash Programming Settings in Vitis	21
8	QSPI Flash Programming Settings in Vivado - Adding the Memory Device	21
9	QSPI Flash Programming Settings in Vivado	22

List of Tables

2	UART Configuration	6
3	FPGA Firmware I/O Configuration	8
4	Hardware Setup Step-By-Step Guide	11
5	FPGA Bitstream Generation Step-By-Step Guide	12
6	Vitis Workspace Preparation Step-By-Step Guide	13
7	Running an Application Step-By-Step Guide	14
8	Generating the Boot Image File Step-by-Step Guide	16
9	Preparing the Hardware for QSPI Flash Boot Mode Step-by-Step Guide	17
10	Creating modified FSBL for QSPI programming Mode Step-by-Step Guide	18
11	Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide	20
12	Booting from the QSPI Flash Step-by-Step Guide	22
13	Preparing the Hardware for SD Card Boot Mode Step-by-Step Guide	23
14	Programming the SD Card for SD Card Boot Mode Step-by-Step Guide	23
15	Booting from the SD Card Step-by-Step Guide	23

References

- [1] Vivado Design Suite User Guide, Embedded Processor Hardware Design, UG898, Xilinx, 2019
- [2] Zynq-7000 All Programmable SoC Embedded Design Tutorial, UG1165, Xilinx, 2015
- [3] Cosmos XZQ10 System Board User Manual
→ Ask Enclustra for details
- [4] Cosmos XZQ10 System 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