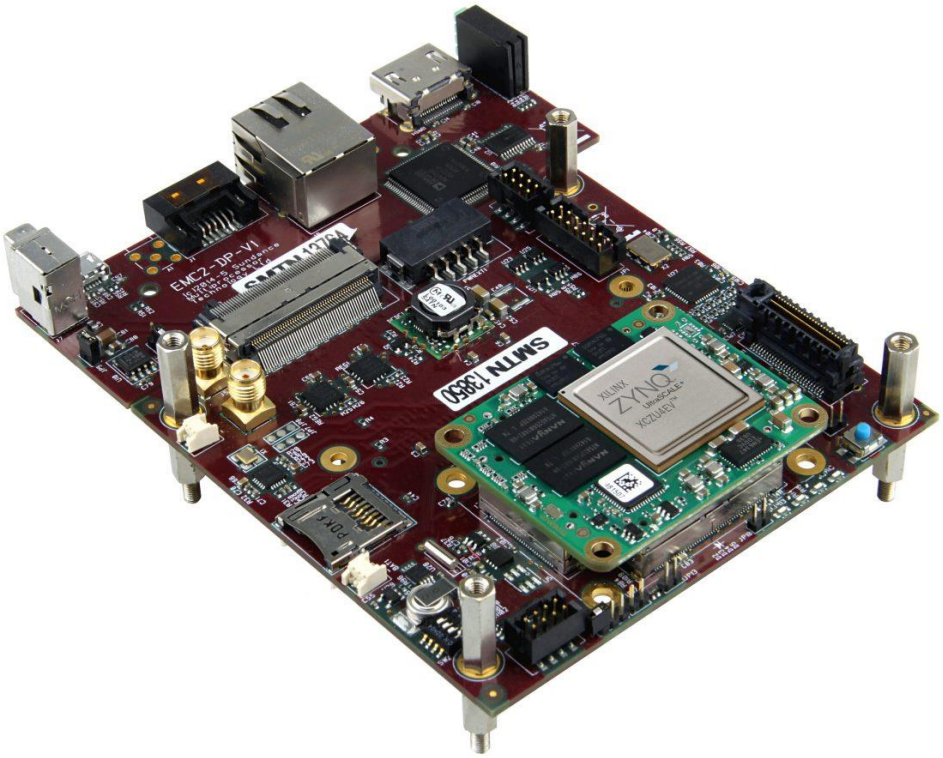
Unit / Module Description:	PCIe/104 OneBank + ARM + FPGA + FMC carrier
Unit / Module Number:	EMC2-DP
Document Issue Number:	3.1
Issue Date:	04/03/2019
Original Author:	Timoteo Garcia Bertoa

EMC2-DP V2
STARTER'S GUIDE



Sundance Multiprocessor Technology Ltd, Chiltern House,
Waterside, Chesham, Bucks. HP5 1PS.

This document is the property of Sundance and may not be copied nor communicated to a third party without prior written permission.
© Sundance Multiprocessor Technology Limited 2019

Certificate Number FM 55022

Revision History

Issue	Changes Made	Date	Initials
1.0	First draft.	2/6/16	TG
1.1	Added the board files for EMC². Updated information about boot images. Added HDMI test project tutorial. Added information about the SEIC connector.	23/6/16	TG
1.2	Added warning at 1.1.2	04/8/16	TG
1.3	Fixed errors + update for z7030	05/10/16	TG
1.4	Added HDMI SDSoC Platform tutorial	07/10/16	TG
2.0	Added chapter 4.Demos Changed the structure of 3.6 and 3.7 Added how to create an SDSoC platform in 2016.3 and above	15/10/17	EW
3.0	Re-worked the whole document, using Vivado 17.4, SDK 17.4, SDSoC 17.4	21/02/19	TG
3.1	Corrected paging and updated pictures	04/03/19	TG

Table of Contents

Table of Figures

1 Introduction

This document is a guide for those who own an EMC²-DP V2, and gives an overview of both hardware and software capabilities in order to set the board up to be used in any system.

This guide provides basic steps to create simple projects targeting the different Xilinx tools.

2 Hardware

2.1 How is the hardware distributed?

The EMC2-DP is an FMC carrier board, PC/104 form factor, with a SEIC extension board which provides connectivity for different interfaces such as HDMI, SATA, USB, etc.

The EMC2-DP has a 4 x 5 cm form factor socket, compatible with different SoCs, with Xilinx 7 Series, Zynq 7 series, Ultrascale or Zynq Ultrascale+ devices.

https://www.sundance.technology/som-cariers/pc104-boards/emc2-dp/

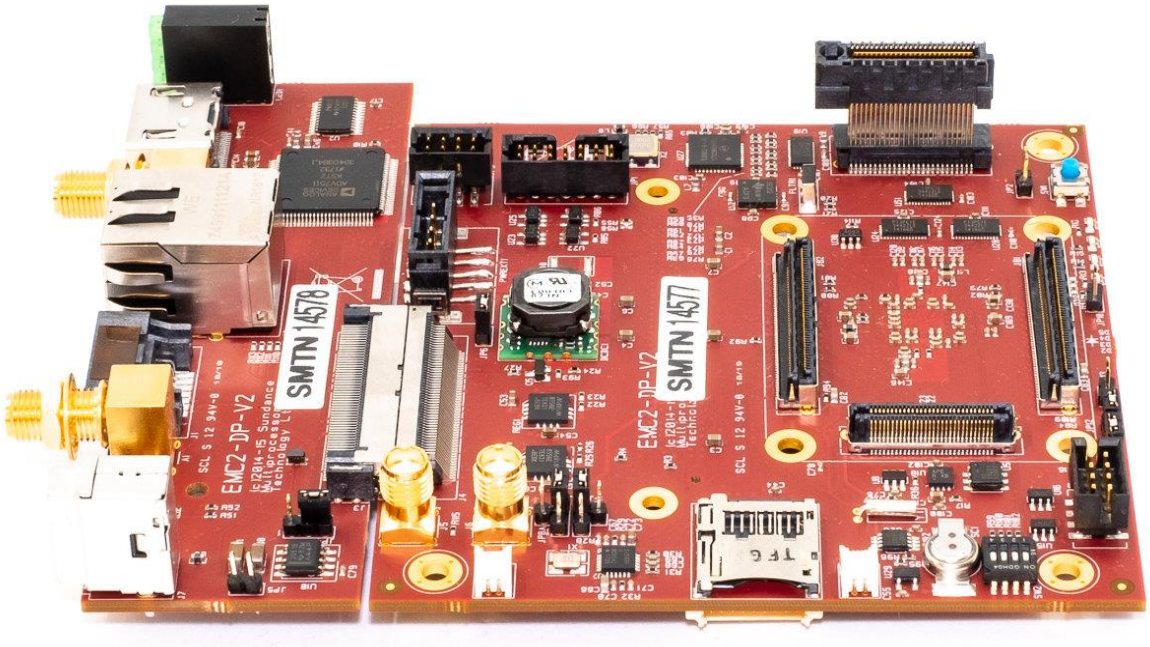


Figure 1 - EMC2-DP

The user might purchase a module with the EMC2-DP. Some examples are:

- 7 Series FPGA example: https://www.sundance.technology/system-on-modules-som/som-modules/te0712-xilinx-artix-7-fpga-som/

Figure 2 - TE0712, Xilinx Artix-7 FPGA SoM

- Zynq 7 series example: https://www.sundance.technology/system-on-modules-som/som-modules/te0715-7030-xilinx-zynq-arm-fpga-som/

Figure 3 - TE0715-7030, Xilinx Zynq ARM + FPGA SoM

- Ultrascale series example: https://www.sundance.technology/system-on-modules-som/som-modules/te0841/

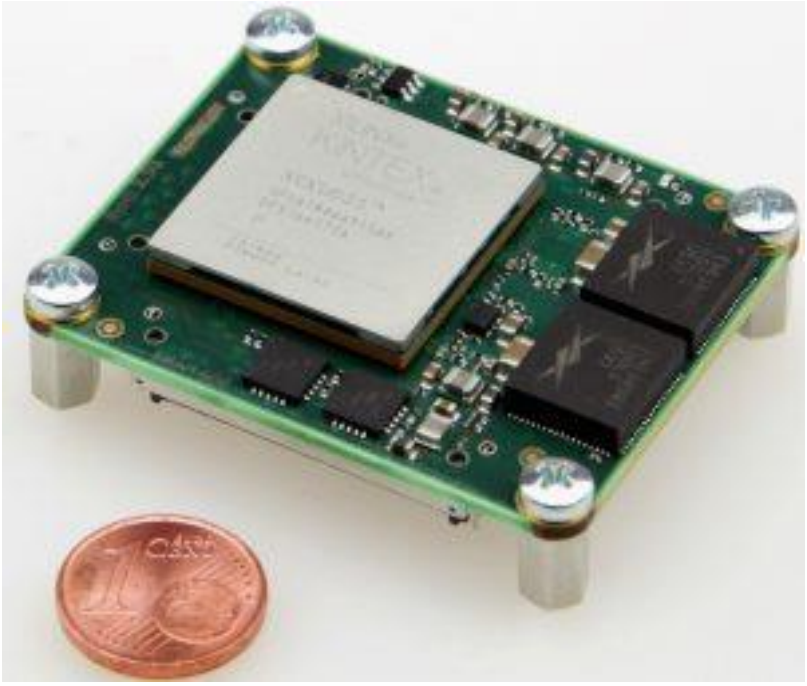


Figure 4 - TE0841, Xilinx Kintex UltraScale SoM

- Zynq Ultrascale+ series example: https://www.sundance.technology/system-on-modules-som/som-modules/teo820-zu4ev/

Figure 5 - TEO820-ZU4EV, Xilinx Zynq UltraScale+

2.2 How can I connect/disconnect the EMC2-DP and the SoM?

Connecting the module seems trivial, but it's recommended to place it on the carrier, and press down from two opposite corners, so that all the pins make contact at the same time, and none get bent.

In order to disconnect the module, follow a similar procedure, pushing up from the bottom of the carrier. Following the steps described in this video helps to avoid any damage to the hardware:

https://www.youtube.com/watch?v=QVOC8dk5n10&feature=youtu.be

2.3 How can I connect the EMC2-DP to the PSU?

The EMC² works well with any power supply which provides the 12V, 3.3V and 5V necessary for the board to work with all its capabilities. The one we recommend is the power supply FSP300-60GHS.

Figure 6 - Power supply

Connect the power supply to the board, using the following cable:

Figure 7 - Cable for the PSU

And connect the cable to the board:

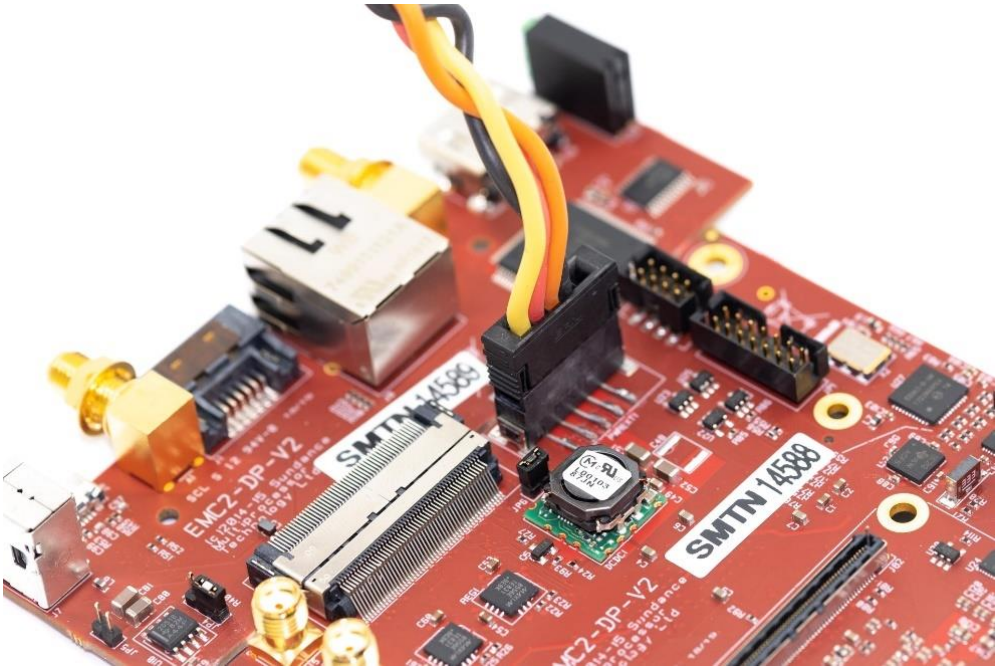


Figure 8 - Connecting the EMC² to the PSU

Turn on the PSU connecting it through the power connector and switching it on:

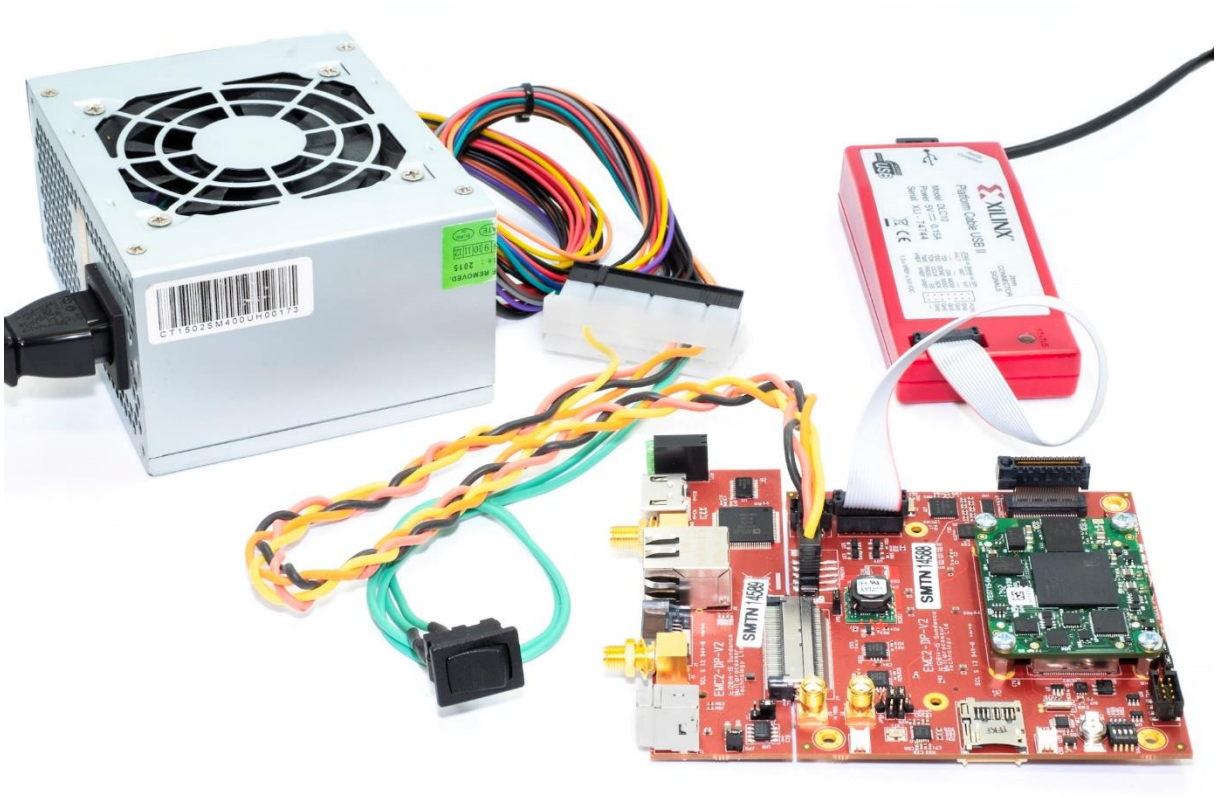


Figure 9 - EMC2-DP fully working

2.4 How can I configure the I/O Voltages?

WARNING: Never configure the IO Voltages with the board powered up!

This board needs an external supply of 3.3V for most of the components on board, as well as 5V and 12V for the PCIe and FMC ports. The IO voltages for the FPGA banks can be selected through the jumpers shown in the picture.

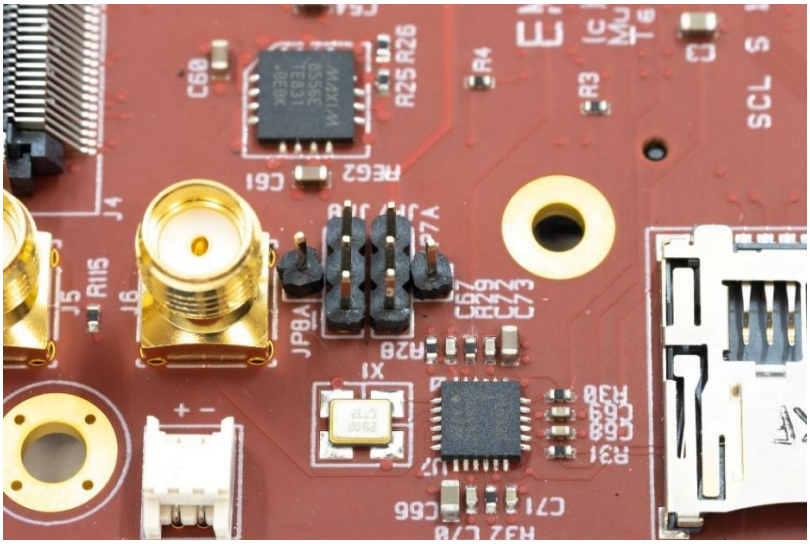


Figure 10 - JP7 and JP8

JP7 and JP8 select the different voltages available as follows:

- 3.3V: Position 1-2

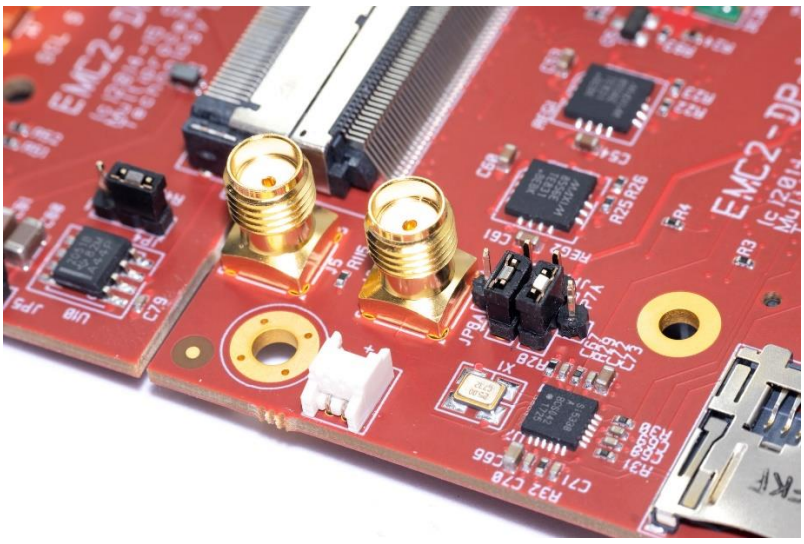


Figure 11 - 3.3V selection

- 2.5V: Position 2-JP7A and 2-JP8A

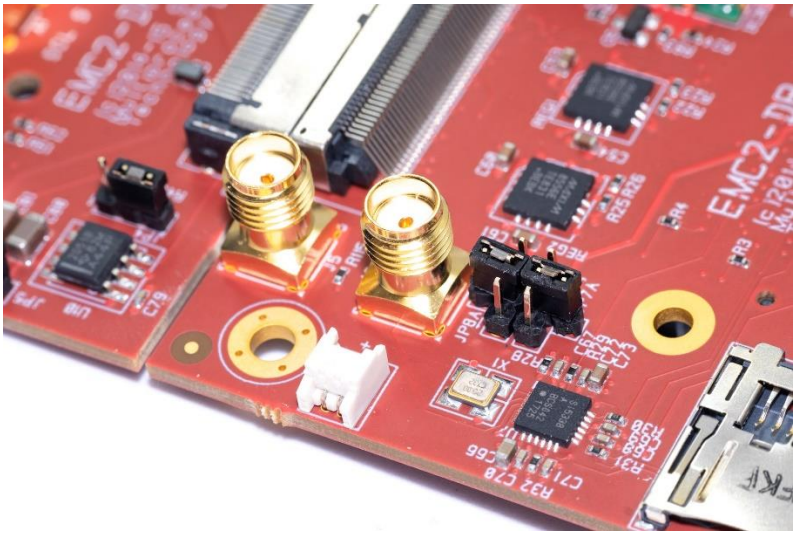


Figure 12 - 2.5V selection

- 1.8V: Position 2-3

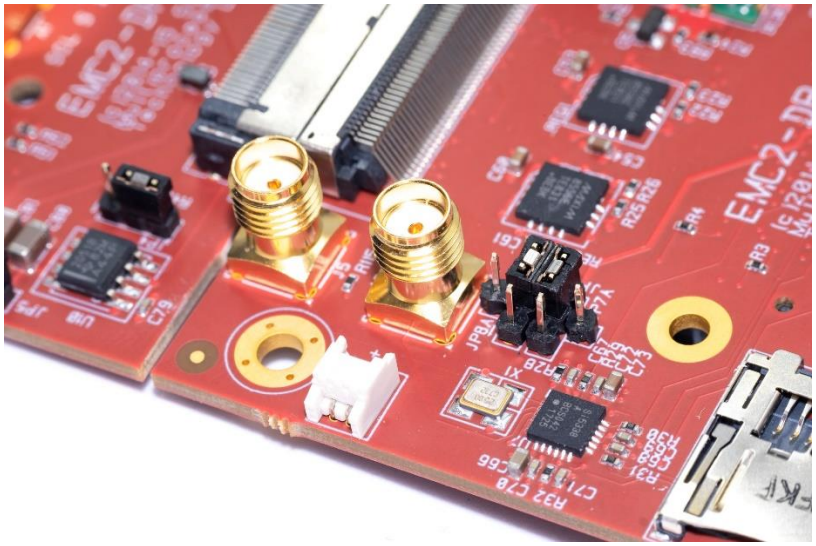


Figure 13 - 1.8V selection

JP7 selects the voltage which feeds mainly the HDMI and SEIC related signals (One of the FPGA banks, while JP8 selects the voltage which feeds mainly the FMC related signals (Two other different FPGA banks).

The FPGA banks, in Xilinx devices, are classified as High Range (HR), High Performance (HP), or High Density (HD). HP banks, for instance, only support up to 1.8V, which means that supplying 3.3V to the FPGA would damage the corresponding bank.

Depending on the SoM used with the carrier board, the specifications for the FPGA might differ, and therefore, there is a risk of damaging the device.

Sundance might provide the EMC2-DP only populating 1.8V selection. If it's not the case, **the user is responsible of ensuring that the voltage is correctly set.**

Contact Sundance if support is needed.

2.5 How can I configure the board to use PCIe?

The EMC2-DP can work in host or add/on mode, depending on the application and the needs of the user.

In case of using the EMC2-DP as host, there is one thing the user must do:

- JP12 must be set, as it is related to the “clock select” pin at the PCIe, and it will provide the 100MHz reference clock.

Figure 15 shows the connectivity.

If the board is used on a stack, it can be used in add/on mode, where JP12 should be unconnected.

To select the upstream port, SW2 should be configured as follows:

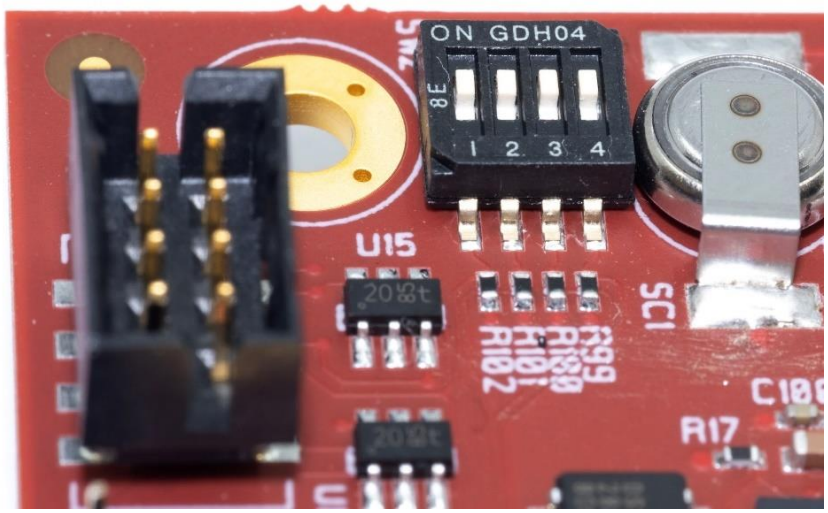


Figure 14 - Upstream port selection

(PEX) Port 0: (Pcie) Lane 0: 0000(LLLL): All On

(PEX) Port 4: (Pcie) Lane 1: 0100(LHLL): On-Off-On-On

(PEX) Port 1: (Pcie) Lane 4: 0001(LLLH): Off-On-On-On

(PEX) Port 5: (Pcie) Lane 5: 0101(LHLH): On-Off-On-Off

(PEX) Port 7: (Pcie) Lane 6: 0110(LHHL): Off-Off-Off-On

(PEX) Port 9: (Pcie) Lane 7: 0111(LHHH): On-Off-Off-Off

Where the PEX Port and Pcie Lane are the same thing, but called differently. L corresponds to “On” and H to “Off” from SW2.

Here there is an example of how JP12 and SW2 should be to set the board as host:

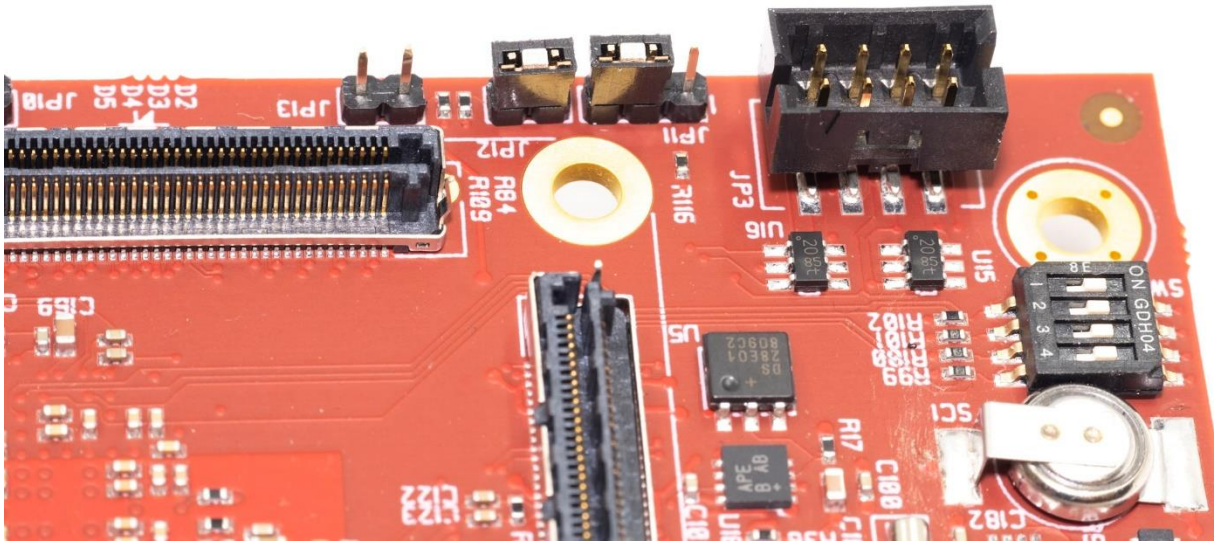


Figure 15 - JP12 sets the host mode, SW2 selects port 0 as upstream port

2.6 How can I boot from flash or SD card?

The jumper (JP11) is the boot mode for the "flash devices" to be set from the EMC2-DP.

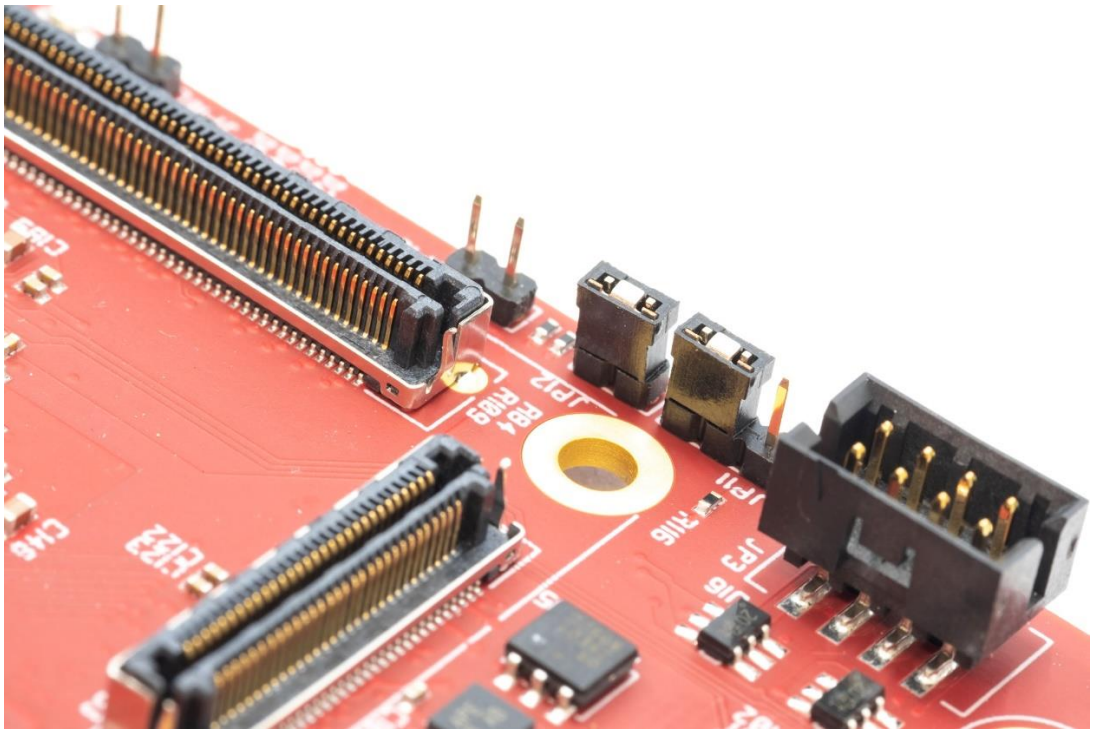


Figure 16 - Boot mode selection

The positions depending on where the user wants to boot from are:

- Position 1-2, QSPI flash mode
- Position 2-3, SD card mode (closer to JP12)

In Figure 17, Next to JP12, JP11 is set as SD booting mode.

2.7 Information about the SEIC Connector

The EMC²-DP has most of its capabilities available at the extension board, accessible through the SEIC connector.

Figure 17 - SEIC extension board

This connector is labelled as J3 and J4 on the extension board and main board respectively.

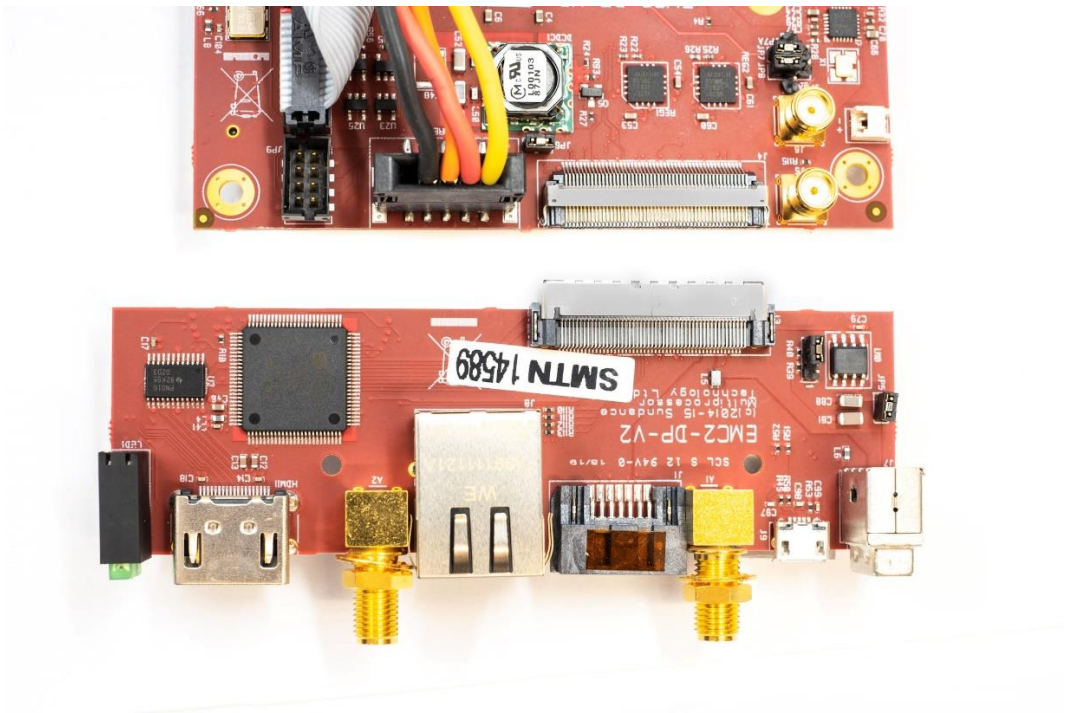


Figure 18 - SEIC connector

The pinout of the SEIC is represented in this schematic, where the top pins of both J3 and J4 are connected, as well as the bottom pins.

Figure 19 - SEIC Connector Pinout

3 Software

3.1 How can I use Vivado with the EMC²?

The EMC2-DP has been used in multiple versions of Vivado, from 2015, and it's recommended to use always the newer versions.

Board files from Trenz Electronic can be used depending on the module installed on the carrier.

Also, Sundance Multiprocessor Technology LTD provides documentation and resources in GitHub:

https://github.com/SundanceMultiprocessorTechnology/

Board files from Sundance are accessible at "VCS-1_Board_Files" repository:

https://github.com/SundanceMultiprocessorTechnology/VCS-1_Board_Files

Using board files from Trenz allows the user to have automatic configuration for the Zynq device and external DDR memory, whereas using the Sundance board files also provides some default configurations to use interfaces present in the carrier board, such as LEDs, UARTs, I2C devices, etc.

All the pin locations for the SEIC devices and FMC are described in the *EMC² Board IO* document, or a Master Pinout spreadsheet located in GitHub with the Board Files.

To include the board files in your system, download the corresponding files, and add them at the installation path of Vivado, normally:

<installationpath>/Xilinx/Vivado/20XX.X/data/boards/board_files/

NOTE: all the examples shown in this guide are using Vivado 17.4, SDK 17.4, SDx 17.4, and EMC2-DP + TE0820-4EV

Having Vivado open, create a new project, either on “Create Project”, or File → New Project:

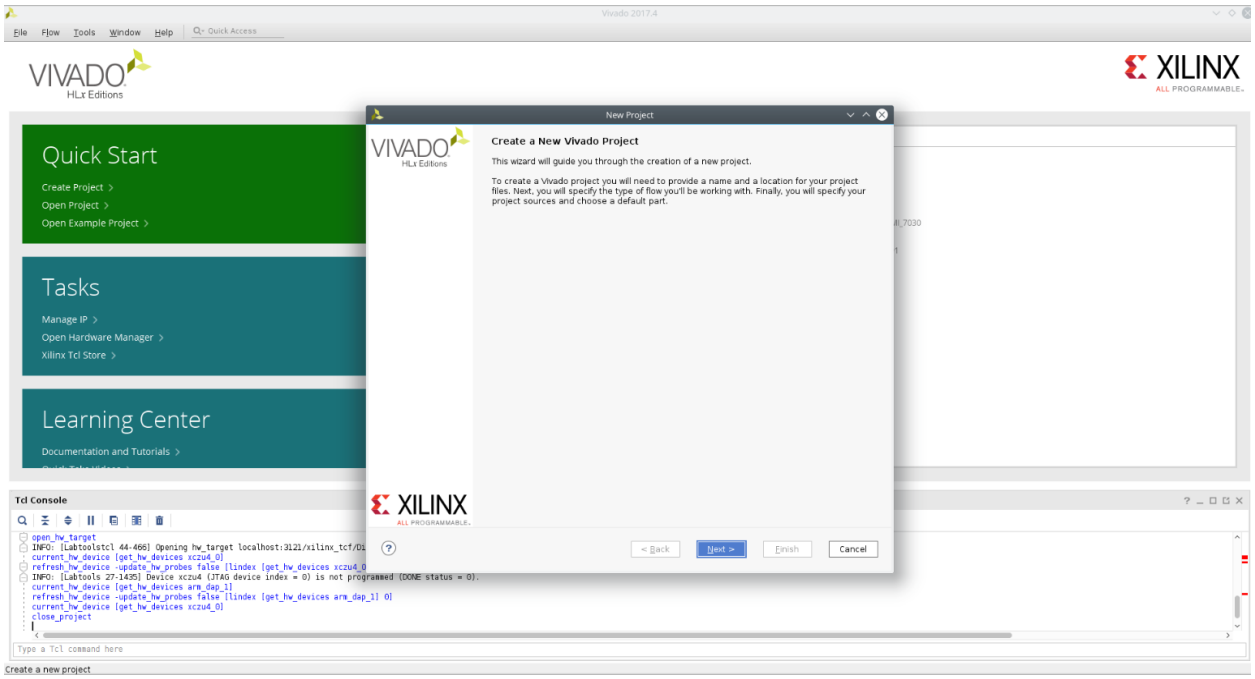


Figure 20 - Create a new project

Click “Next” and select the path of the project.

Choose RTL project and mark the square “Do not specify sources at this time” in case the user doesn’t require importing any source and it’s a blank project.

Then, when Vivado asks for a device part. Select “Boards”, and choose the corresponding board files.

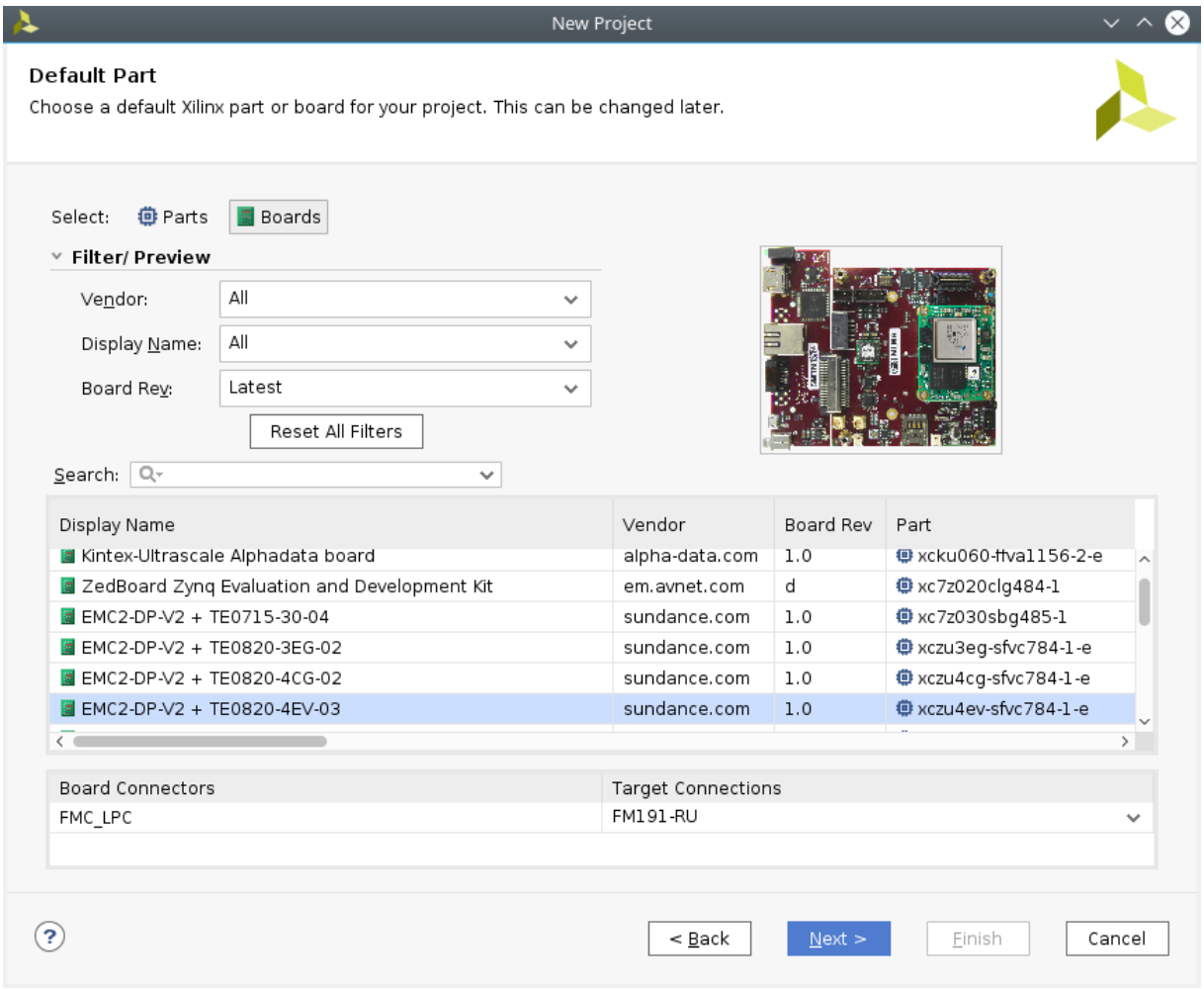


Figure 21 - Board file selection

Note that at “Board Connectors”, FMC_LPC is available, to connect compatible FMC boards, like FM191-RU.

Once the project is opened, the user can see the information about the board and the part used:

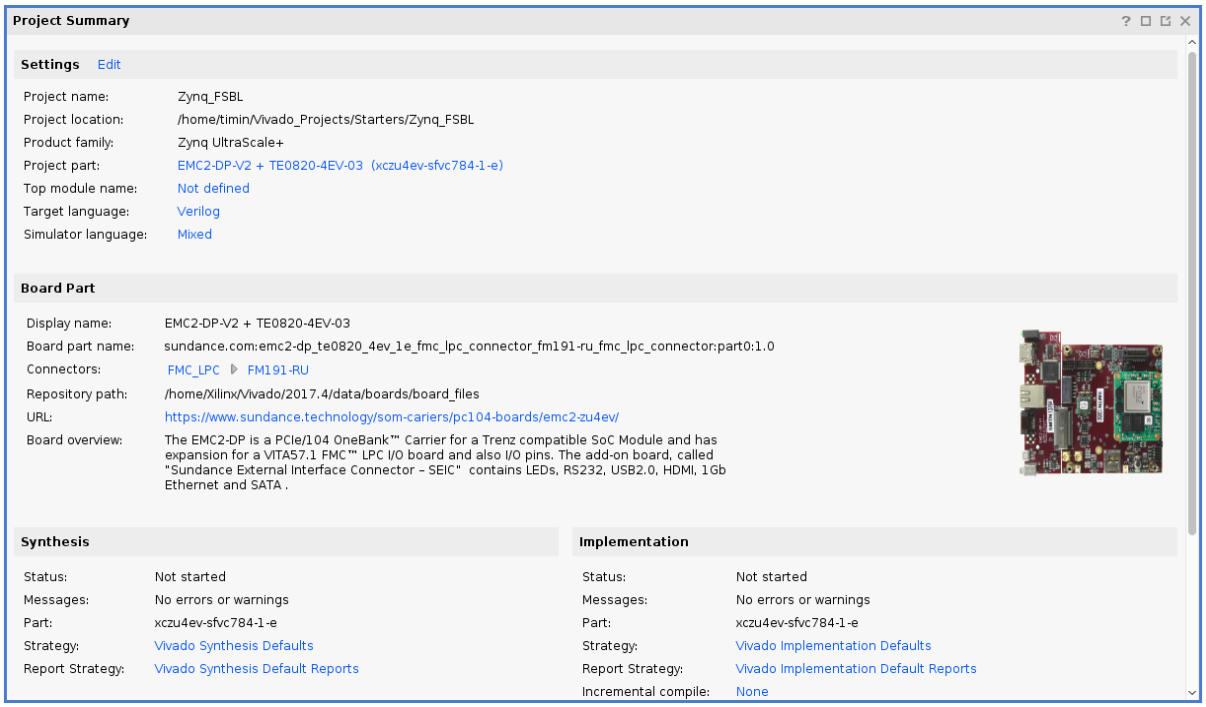


Figure 22 - Project information in Vivado

As soon as the user creates a new block design in IP Integrator, there will be some interfaces available, already set up and ready to use:

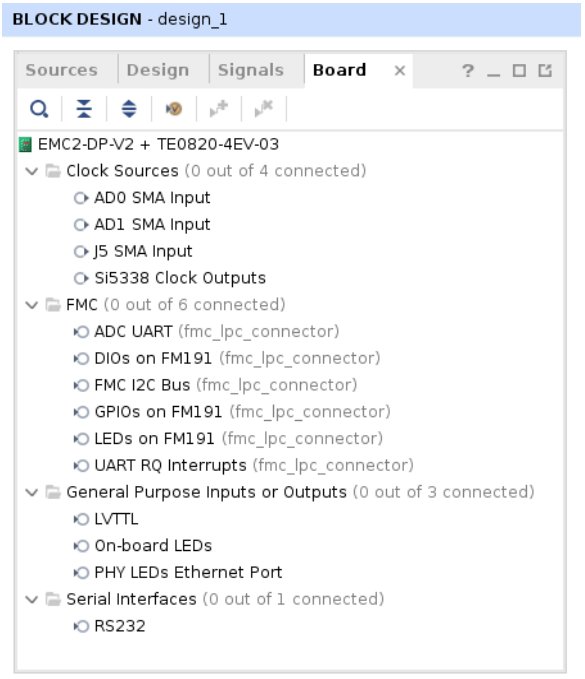


Figure 23 - Board interfaces in IP Integrator

The interfaces available are those ones connected to the FPGA. To use the rest of the capabilities within the PS, they can accessed through the MIO/EMIO pins.

As an example, to select one of the outputs of the clock synthesizer (Si5338), just double click on Si5338 Clock Outputs, at Clock Sources:

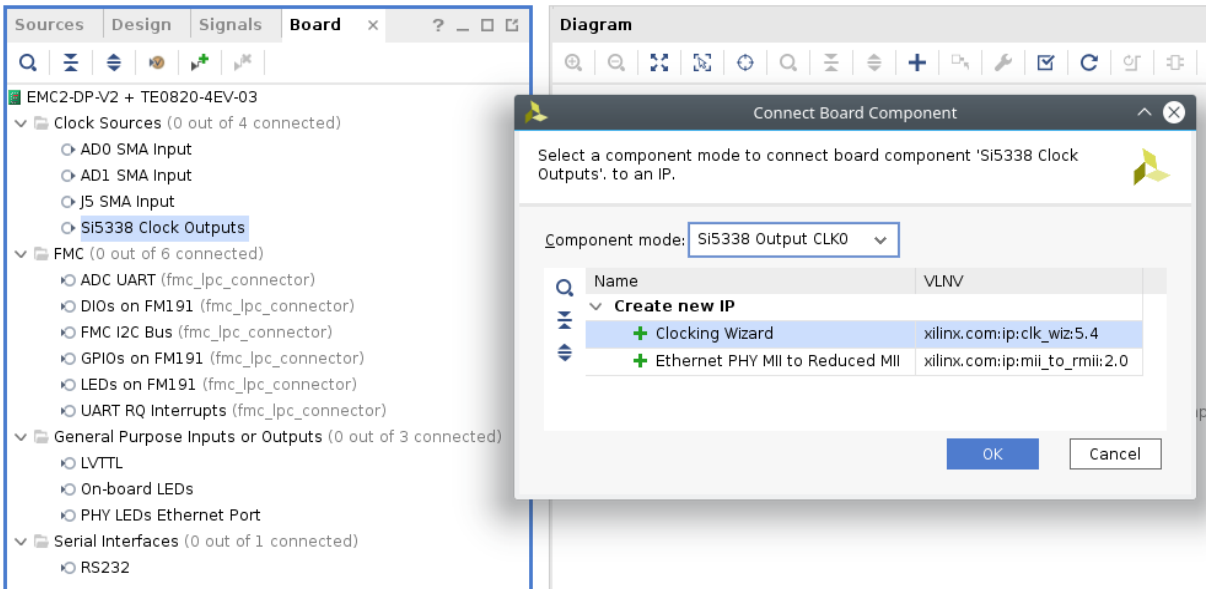


Figure 24 - How to use interfaces in IPI

Select “Si5338 Output CLK0” as component mode, and “Clocking Wizard” as the desired IP. Automatically, the input will be assigned to the IP selected.

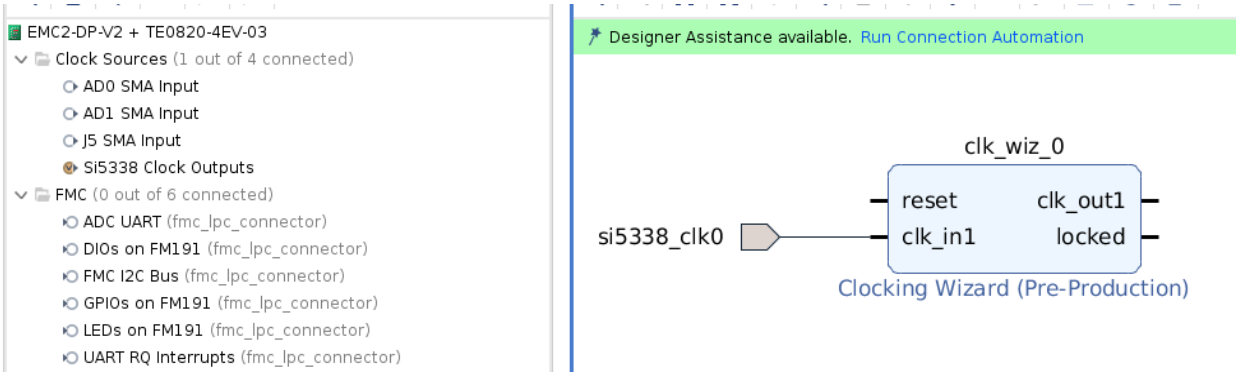


Figure 25 - Block generated through the interface

Remember that these capabilities are made so that the user can access easily to them without having to constrain them.

3.2 How can I create a Zynq FSBL?

To create a FSBL is necessary to determine the hardware of the system (PS+PL) with Vivado, and export that hardware design to the SDK environment, where it's possible to create an FSBL, which boots the PS, mapping the blocks in the PL into memory (if any).

Following the steps described in the previous section, create a project, having a blank block design in IP Integrator.

Add a Zynq Ultrascale+ MPSoC IP block, and connect *pl_clk0* to *maxihpm0_lpd_aclk*.

Alternatively, with tcl commands:

```
startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:zynq_ultra_ps_e:3.1
zynq_ultra_ps_e_0
endgroup
connect_bd_net [get_bd_pins zynq_ultra_ps_e_0/pl_clk0] [get_bd_pins
zynq_ultra_ps_e_0/maxihpm0_lpd_aclk]
```

Figure 26 - Zynq Ultrascale+Processing System

Then, click Run Block Automation, and press "OK".

```
apply_bd_automation -rule xilinx.com:bd_rule:zynq_ultra_ps_e -config
{apply_board_preset "1" }  [get_bd_cells zynq_ultra_ps_e_0]
```

This will automatically configure the Processing System, as per defined in the board files.

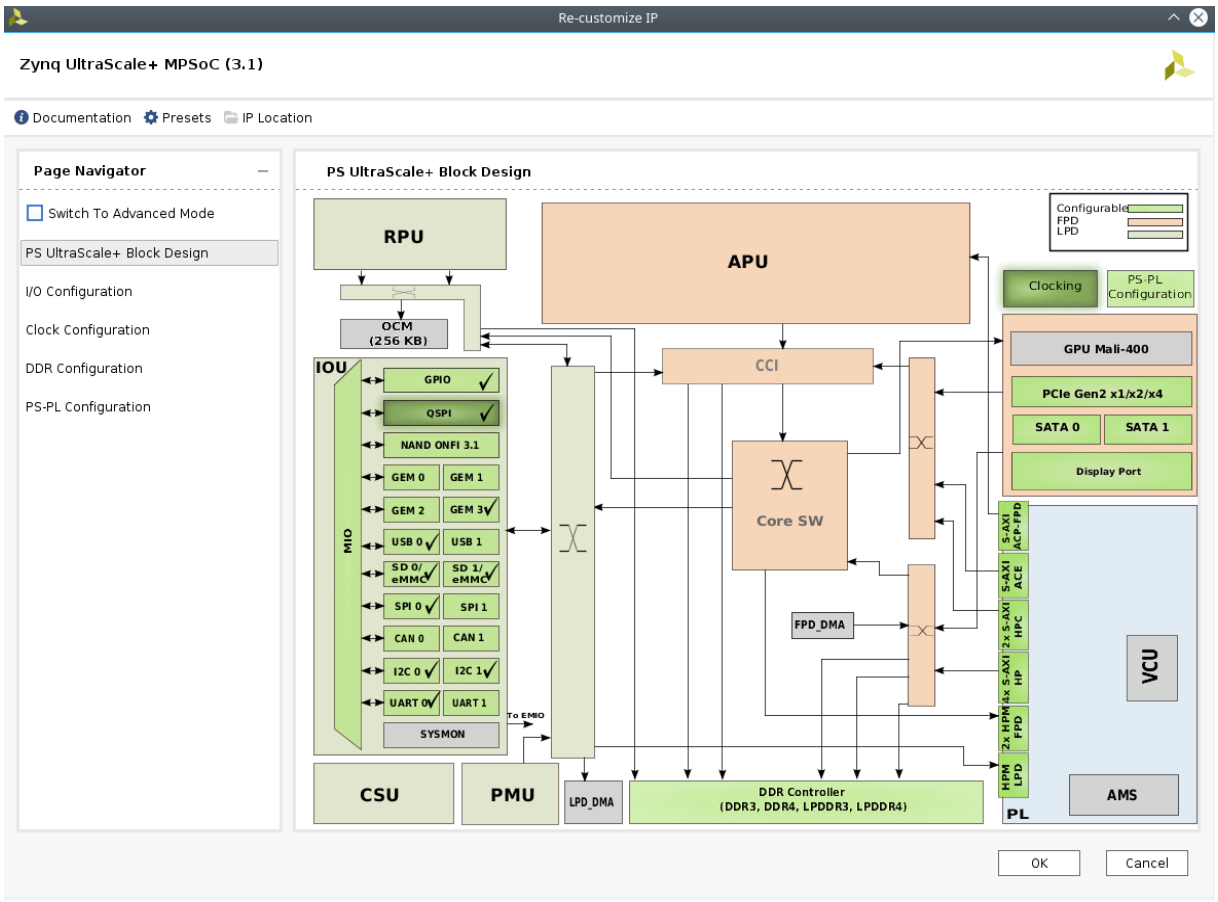


Figure 27 - Zynq Ultrascale+ architecture

It is highly recommended to read the Zynq Ultrascale+ documentation from Xilinx to understand the architecture and be able to configure the device properly.

Click “Validate design” and then “OK”

Figure 28 - Validate design

Create a Wrapper of the design. To do so, go to the “Sources” tab, and right click on the design → Create HDL Wrapper

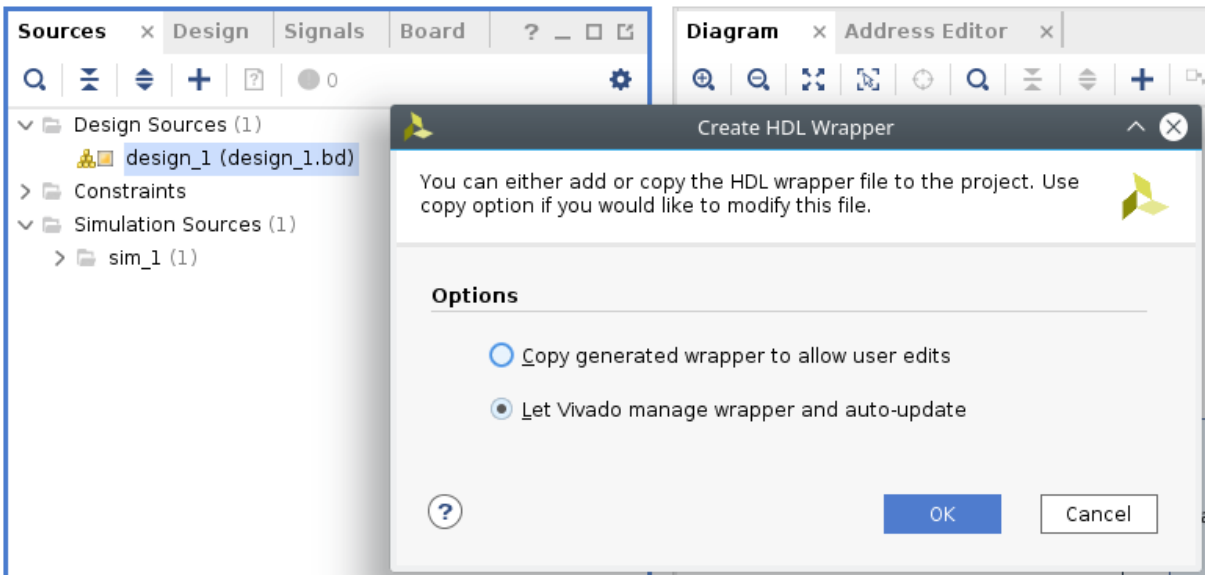


Figure 29 - HDL Wrapper

Generate the Bitstream, tool easy to find in the Flow Navigator.

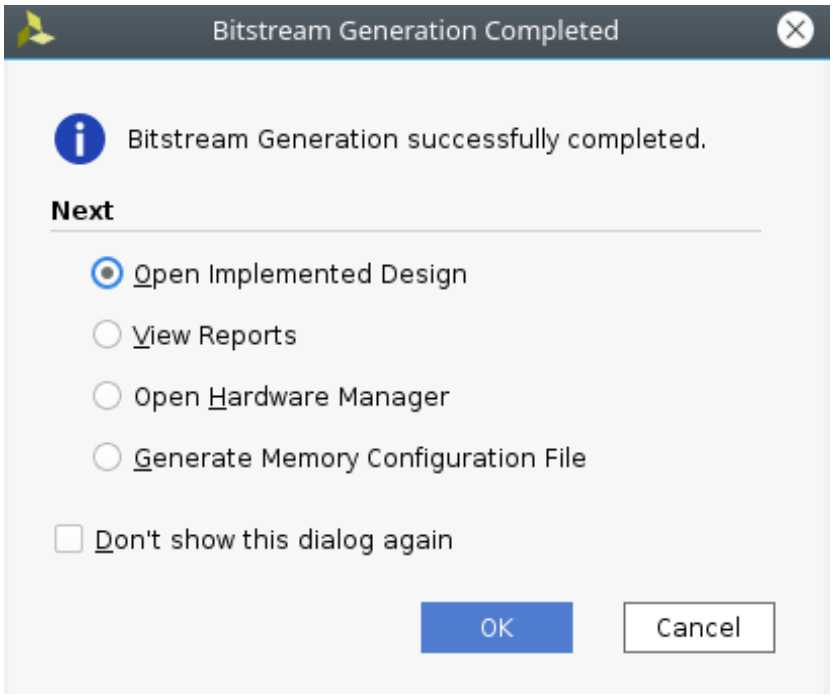


Figure 30 - Generating Bitstream

Export the hardware. Select “File -> Export -> Export Hardware”

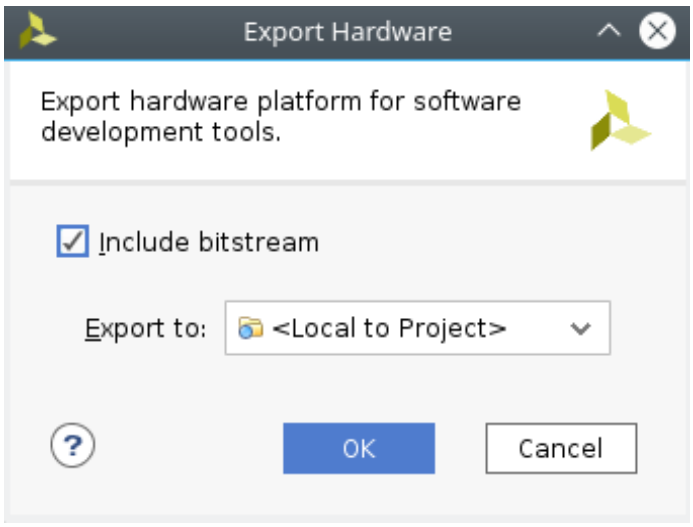


Figure 31 - Export hardware including the bitstream

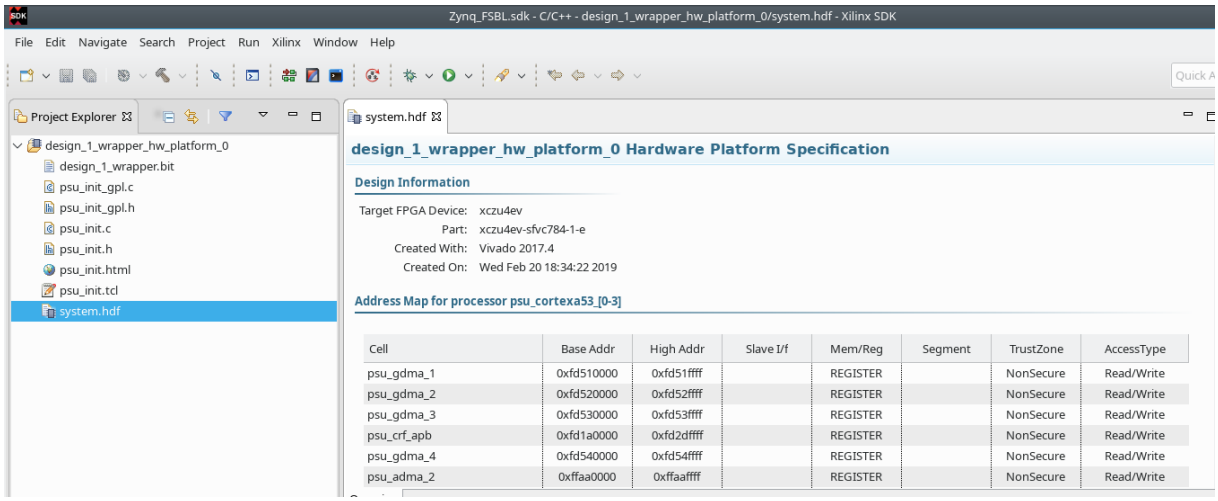


Figure 32 - Xilinx SDK

To create a FSBL, go to File -> New -> Application Project

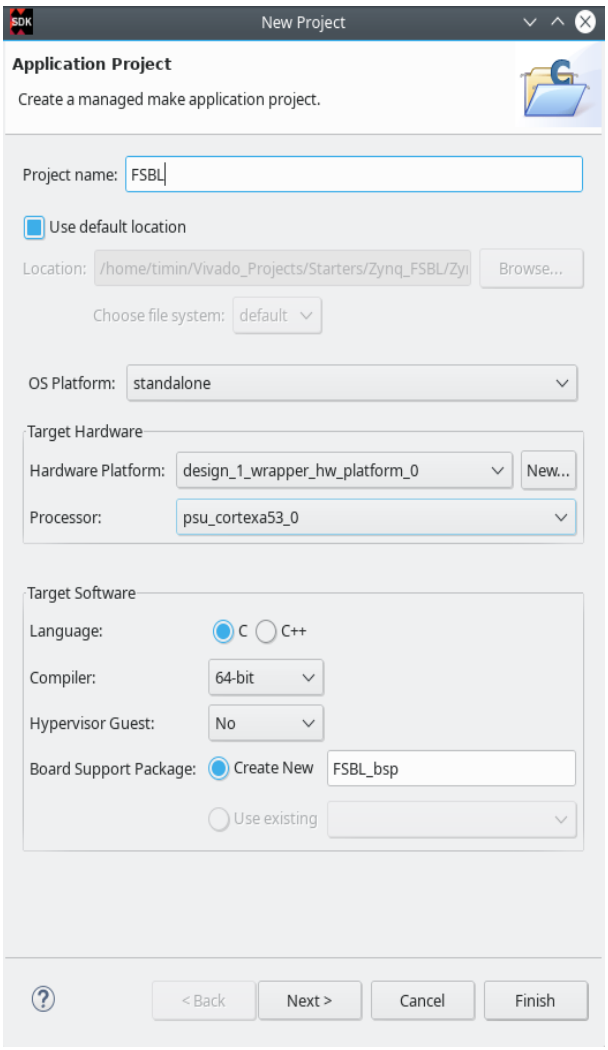


Figure 33 - Create FSBL Application Project

Give a name to the project, for example “FSBL”. It’s intuitive, as it shows the hardware platform (HDL Wrapper exported previously), the ARM core selected from the Zynq, and the OS platform, although in this case it’s a standalone application.

Press next and select “Zynq MP FSBL”

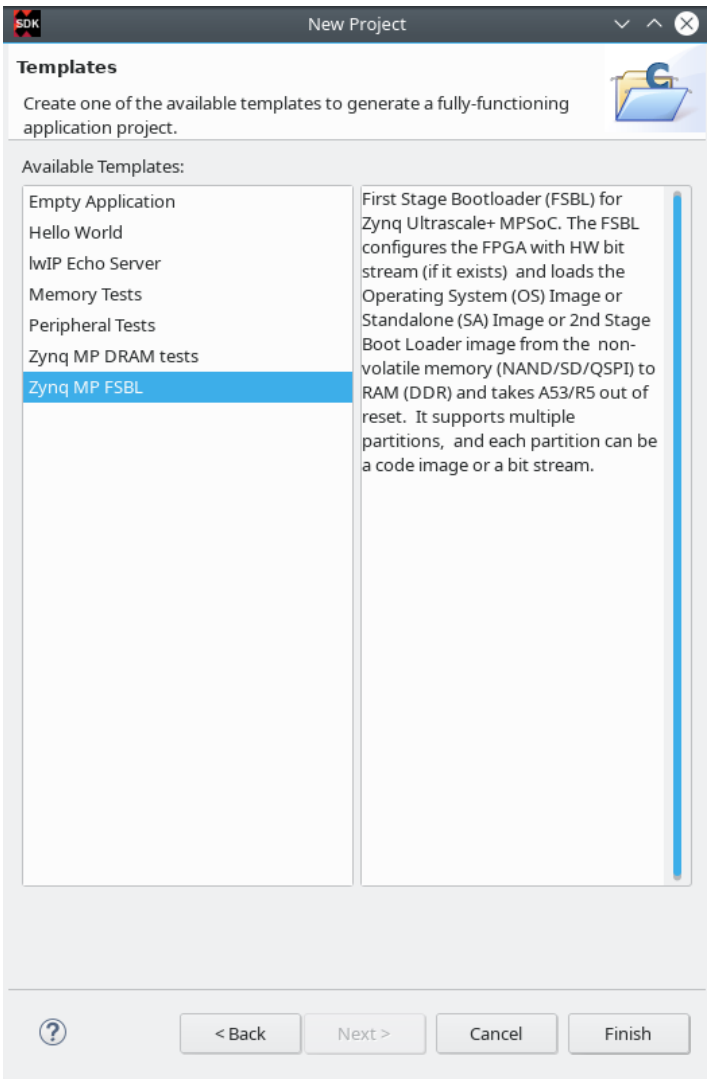


Figure 34 - Zynq FSBL project

Press “Finish” and the FSBL project will be added.

Right click on the project, and build it. That should generate an FSBL.elf file under “Debug”, which can be used as bootloader.

Now there is an FSBL that can be used for initialize the PS. It’s recommended to check out the file system.mss from the BSP project, because depending on the hardware exported, the user can find different code examples from Xilinx for the different peripherals included in the Zynq processing system (UART, DDR, I2C, USB, etc.).

3.3 How can I create SD boot file?

Assuming and FSBL project has been created for any hardware design:

Click FSBL -> Create Boot Image

The user can now see that SDK tries to export a .bif file, from the .elf file (FSBL project in SDK) and the .bit file (Vivado bitstream file).

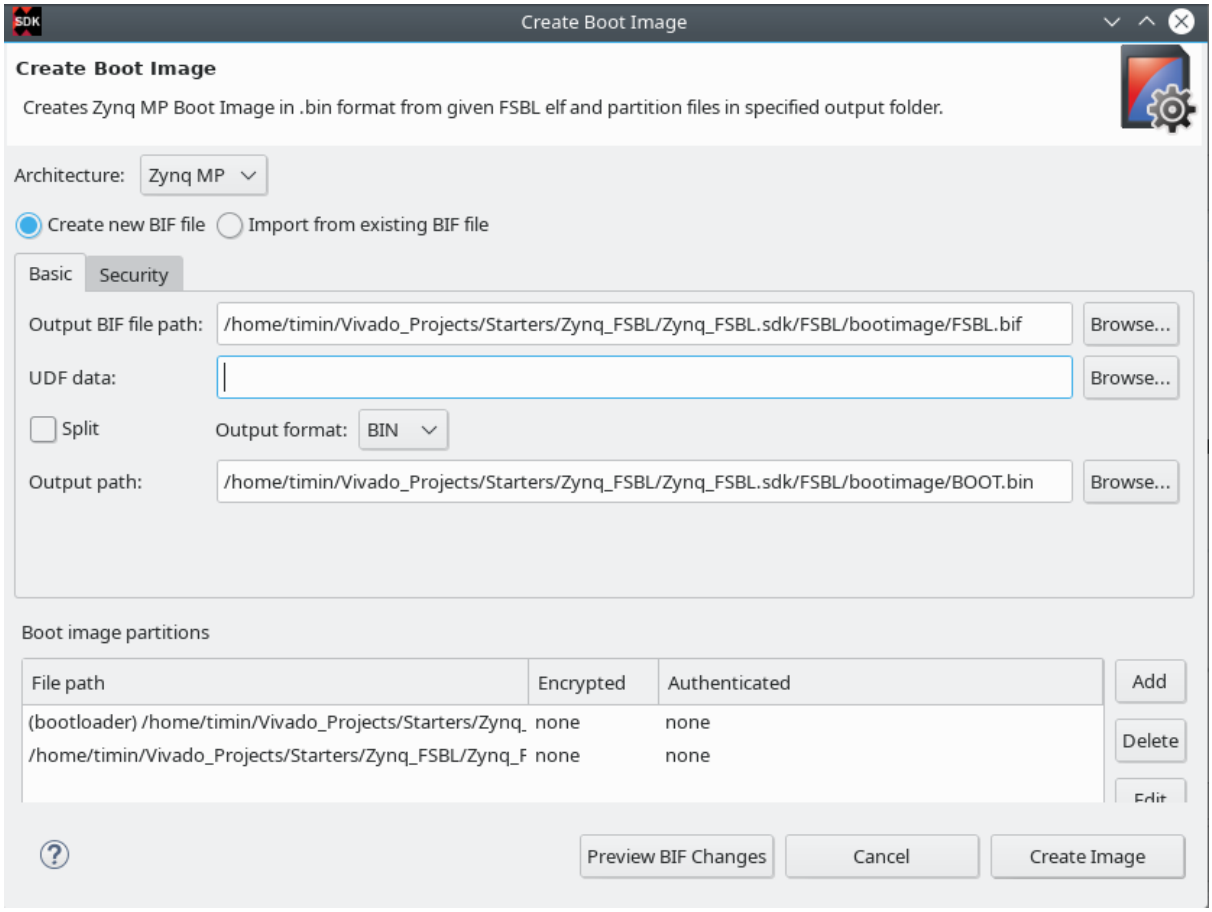


Figure 35 - Create image

Press “Create Image”.

Now at, *<projectpath>/<projectname>/projectname.sdk/FSBL/bootimage/* there should be a .bin file called “BOOT.bin” and a .bif file called “FSBL.bif”

This BOOT.bin file can be placed in a SD card, and selecting the jumpers on the EMC2-DP to boot from SD, described in this document, the Zynq device will be configured when powering up the board.

Remember that in this case it’s assumed that the project run in the Zynq is just the FSBL itself. The PS needs to be always booted in order to run any application. To do so, there should be 3 files merged into BOOT.bin: FSBL.elf (bootloader), .bit file (PL) and .elf file (application).

3.4 How can I program the FPGA from Flash in Vivado?

To program the FPGA from flash, the user needs to erase the flash and program the FPGA with a .bin or .mcs file.

Using the FSBL project in Vivado as example:

This project has a .bit file generated, but to program the flash a .bin file is needed. To let Vivado know that, it's possible to do it going to "Project Settings" in the Vivado Flow Navigator, and mark the .bin option in Bitstream options.

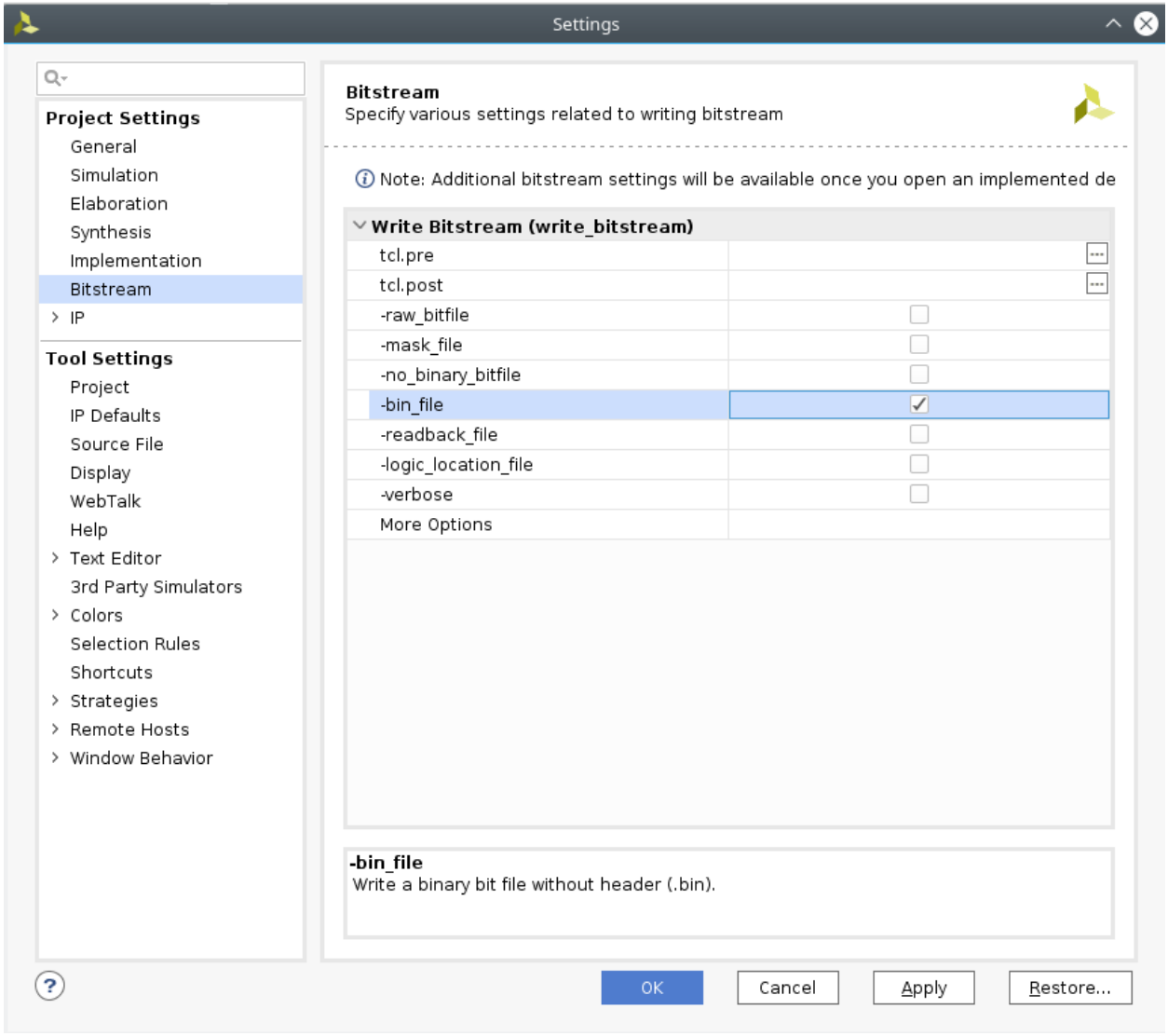


Figure 36 - Project settings

Now rewrite the bitstream, and open the hardware manager.

Alternatively, the flash memory **can be programmed with an .mcs file**. It can be generated in SDK in the same way the BOOT.bin file is generated for SD booting. Select MCS instead of BIN at the file format options.

Connect the EMC2-DP to the computer through JTAG, and power the board up.

Open a new target, and the EMC2-DP will be recognized by the software tool:

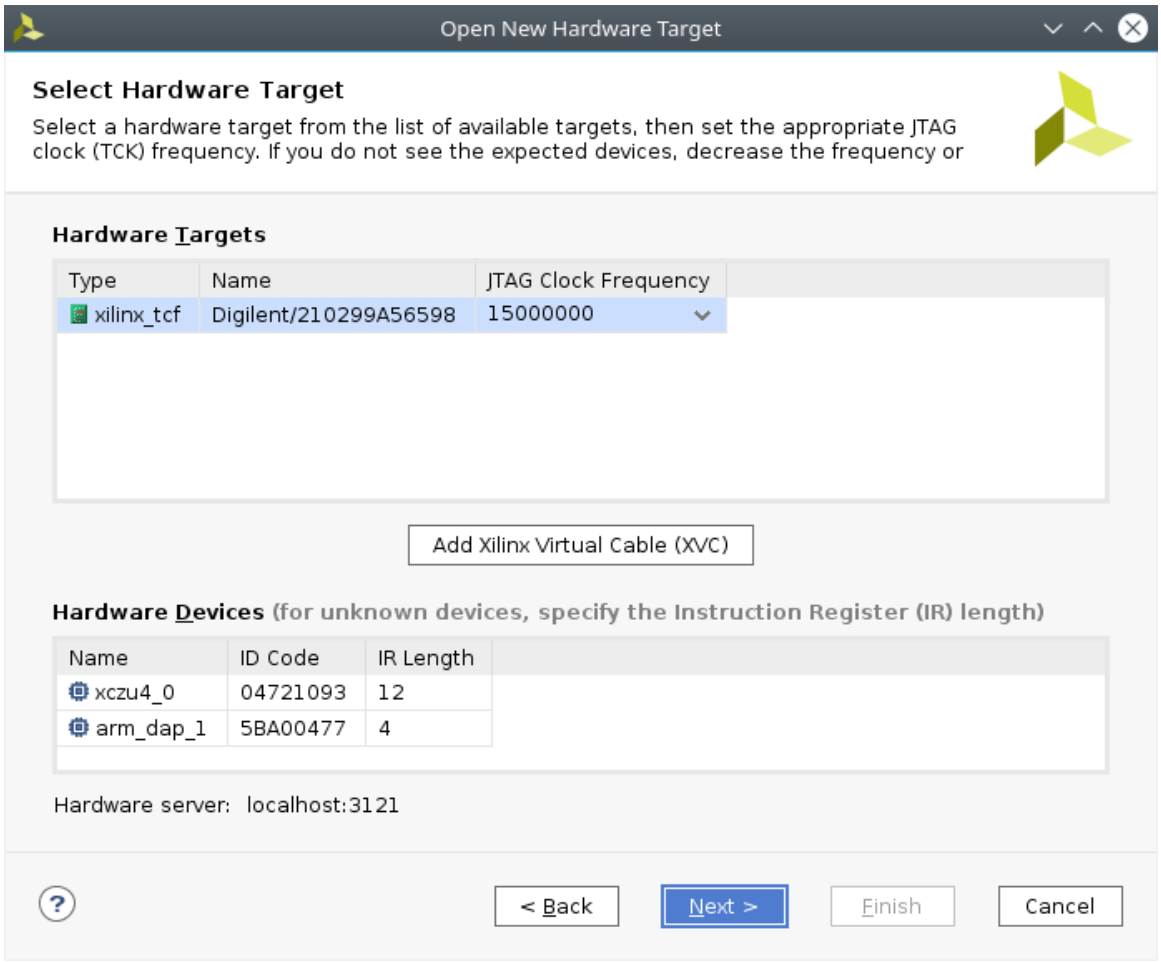


Figure 37 - Open target

The user can see at this point that the hardware manager sees the PL and the PS from the Zynq (FPGA + ARM). Press “Next” and “Finish”.

Now, to program the Flash, right click on the FPGA, and select “Add Configuration Memory Device”

Select the right flash memory part, and then, when it’s part of the JTAG chain, right click on the flash memory, and select “Program Configuration Memory Device”

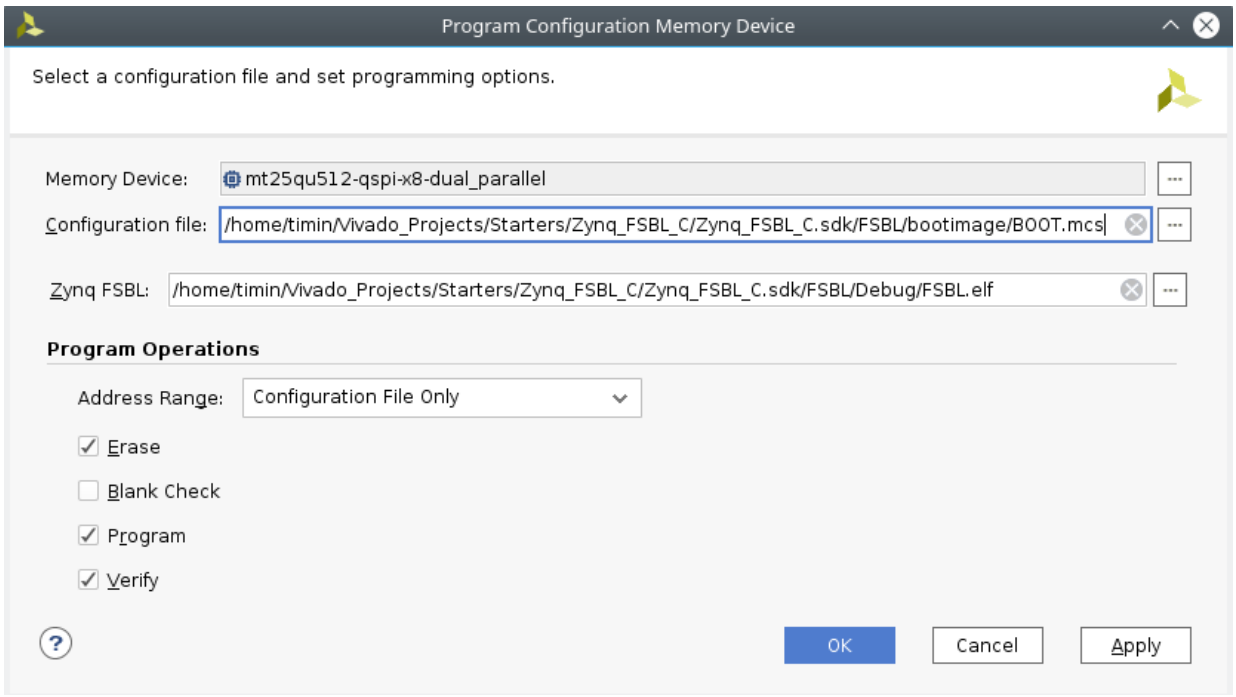


Figure 38 - Selecting memory flash part

Before pressing "OK", **JP11 should not be in SD boot mode**.

3.5 How can I create an SDSoC Platform?

The steps explained in order to create this platform can be used for any other platforms, changing the paths and names specified. For more detailed information about platforms and libraries, check these documents from Xilinx: UG1146 and UG1236

This platform has been made for a standalone application.

Create a project in Vivado, using IPI. The output clocks of the clock wizard are 100MHz, 200MHz and 300MHz respectively.

The project shown as example has been called “EMC2DP_TE0820_4CG_SDxPlatform”

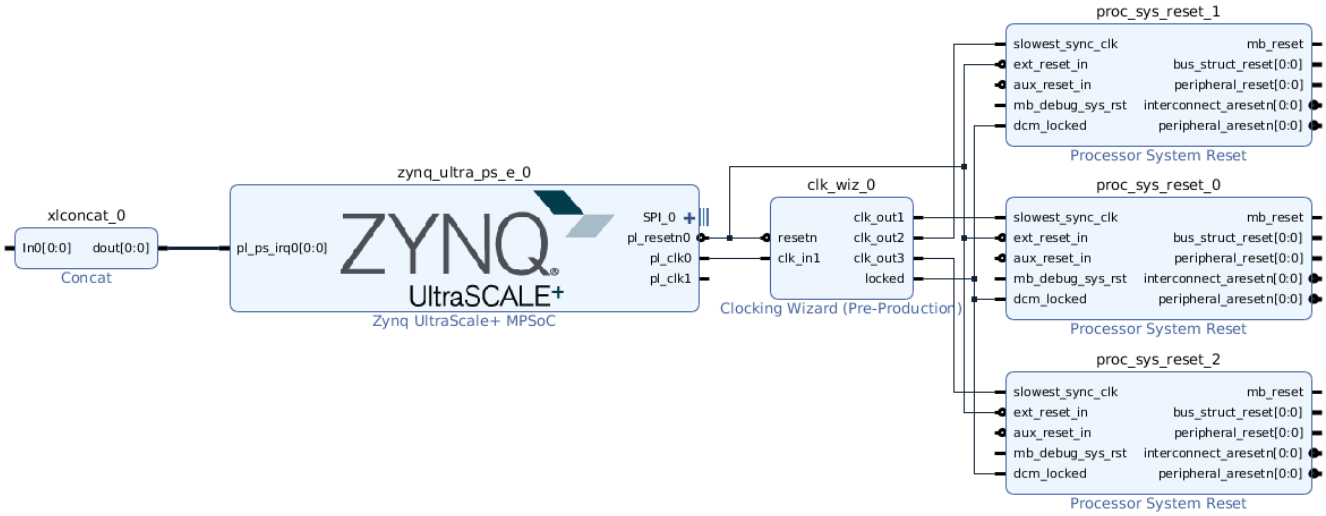


Figure 39 - Vivado design for the platform

Create a wrapper, generate the bitstream.

In the tcl console, introduce the following commands:

This command specifies the name of the platform, and where is the block design

```
set_property PFM_NAME "sundance.com:EMC2DP_TE0820_4CG:EMC2DP_TE0820_4CG:1.0" [get_files
./EMC2DP_TE0820_4CG_SDxPlatform.srcs/sources_1/bd/design_1/design_1.bd]
```

Declare the available clocks

```
set_property PFM.CLOCK { clk_out1 {id "0" is_default "true" proc_sys_reset "proc_sys_reset_0"
} clk_out2 {id "1" is_default "true" proc_sys_reset "proc_sys_reset_1" } clk_out3 {id "2"
is_default "true" proc_sys_reset "proc_sys_reset_2" } } [get_bd_cells /clk_wiz_0]
```

Declare the AXI available ports for SDSoC to use.

```
set_property PFM.AXI_PORT { M_AXI_HPM0_FPD {memport "M_AXI_GP"} M_AXI_HPM1_FPD {memport
"M_AXI_GP"} M_AXI_HPM0_LPD {memport "M_AXI_GP"} S_AXI_HPC0_FPD {memport "S_AXI_HPC" sptag
"HPC0"} S_AXI_HPC1_FPD {memport "S_AXI_HPC" sptag "HPC1"} S_AXI_HP0_FPD {memport "S_AXI_HP"
sptag "HP0"} S_AXI_HP1_FPD {memport "S_AXI_HP" sptag "HP1"} S_AXI_HP2_FPD {memport "S_AXI_HP"
sptag "HP2"} S_AXI_HP3_FPD {memport "S_AXI_HP" sptag "HP3"} } [get_bd_cells
/zynq_ultra_ps_e_0]
```

Declare the interrupts available

```
set intVar []
```

```
for {set i 0} {$i < 8} {incr i} {
lappend intVar In$i {}
}
set_property PFM.IRQ $intVar [get_bd_cells /xlconcat_0]
```

Specify local cache.

```
set_property dsa.ip_cache_dir [get_property ip_output_repo [current_project]]
[current_project]
```

Generate .dsa file

```
write_dsa EMC2DP_TE0820_4CG.dsa -include_bit -force
```

After this is done, export the hardware, launch SDK, and create an FSBL project, and a Hello World project (it is done in the same way as an FSBL, selecting the corresponding Hello World example).

Build both projects.

Having a .dsa file, create a new project in SDSoC:

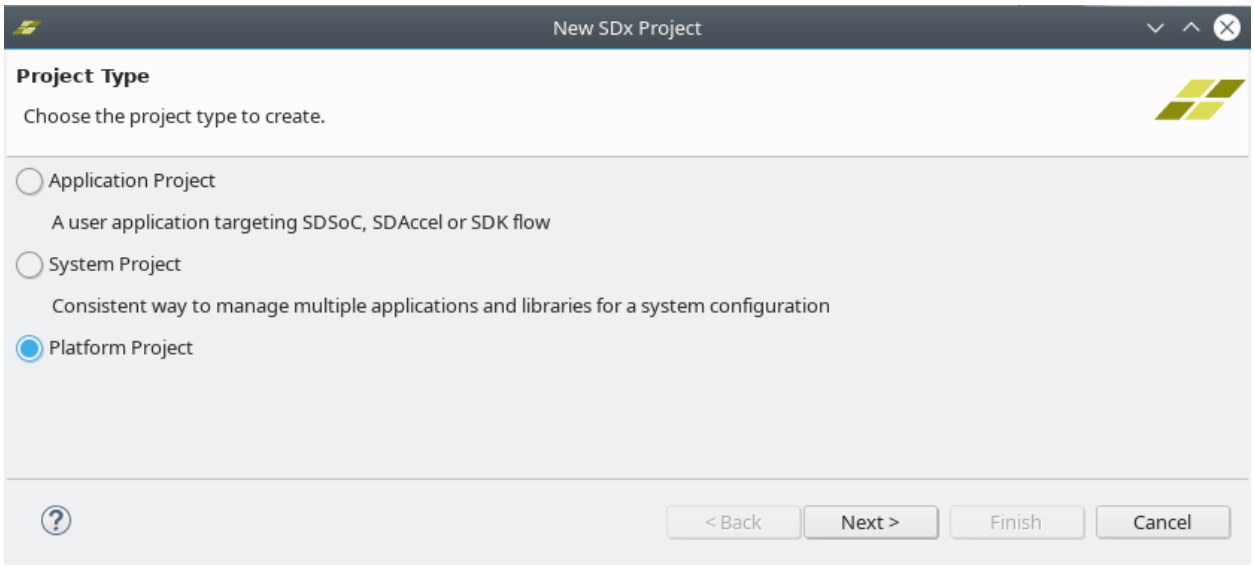


Figure 40 - SDSoC Platform Project

Specify the path where the .dsa file is stored:

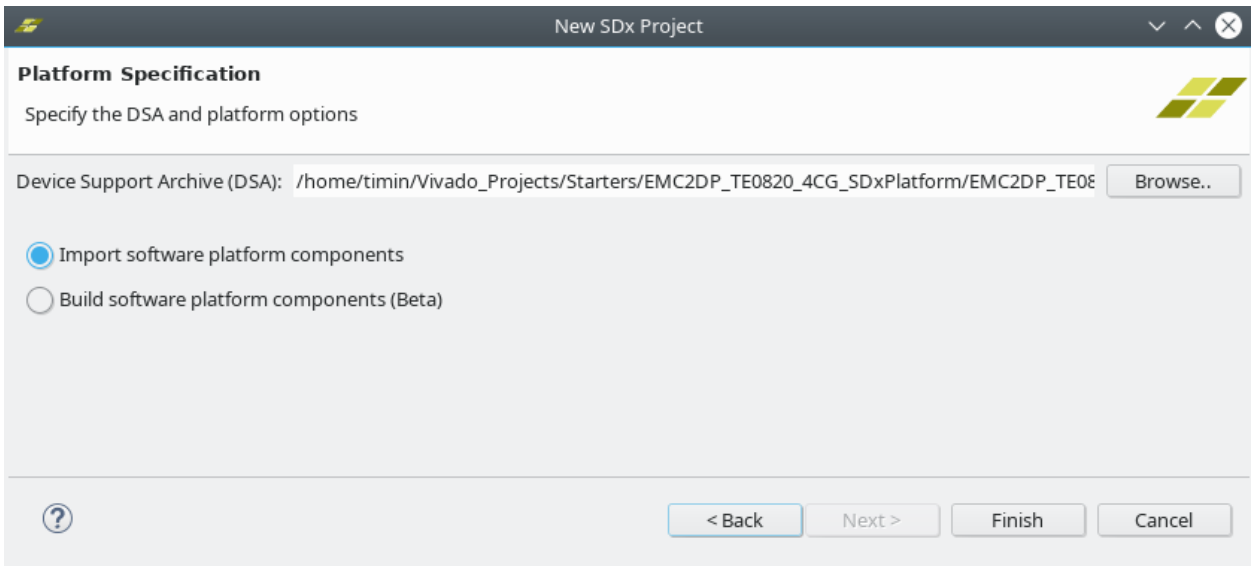


Figure 41 - Import .dsa file

Create the project.

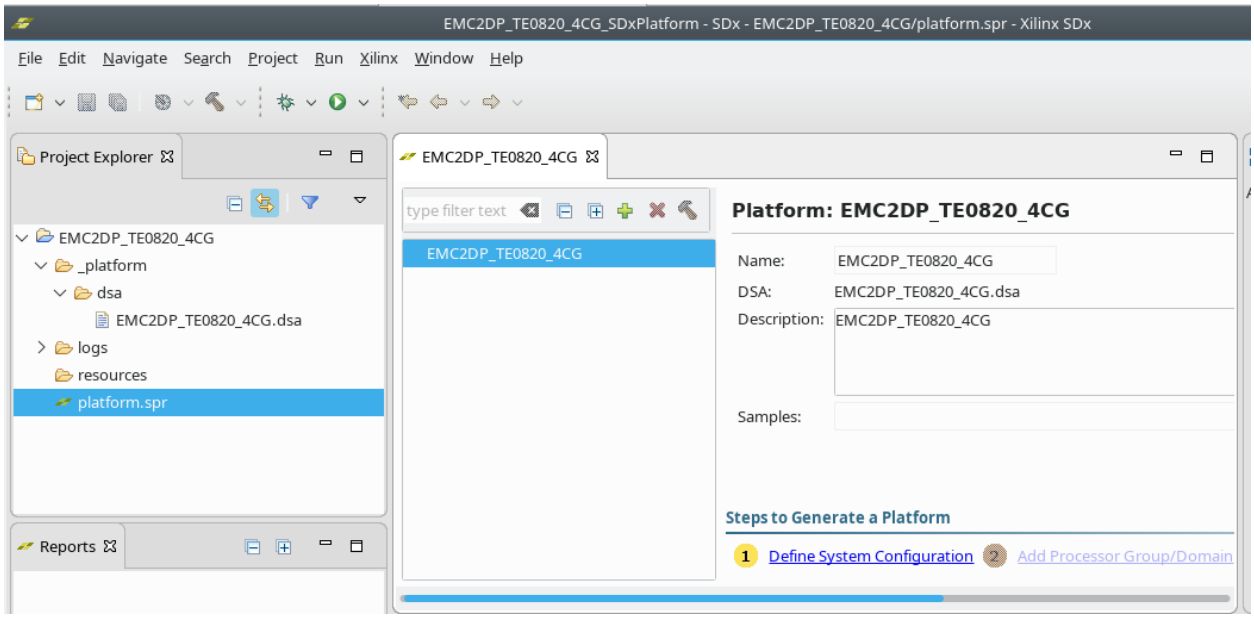


Figure 42 - Platform project opened

There are some icons at the platform window. Clicking on the "+" icon, add a new System Configuration. The paths for the boot and .bif files are the FSBL project created in SDK.

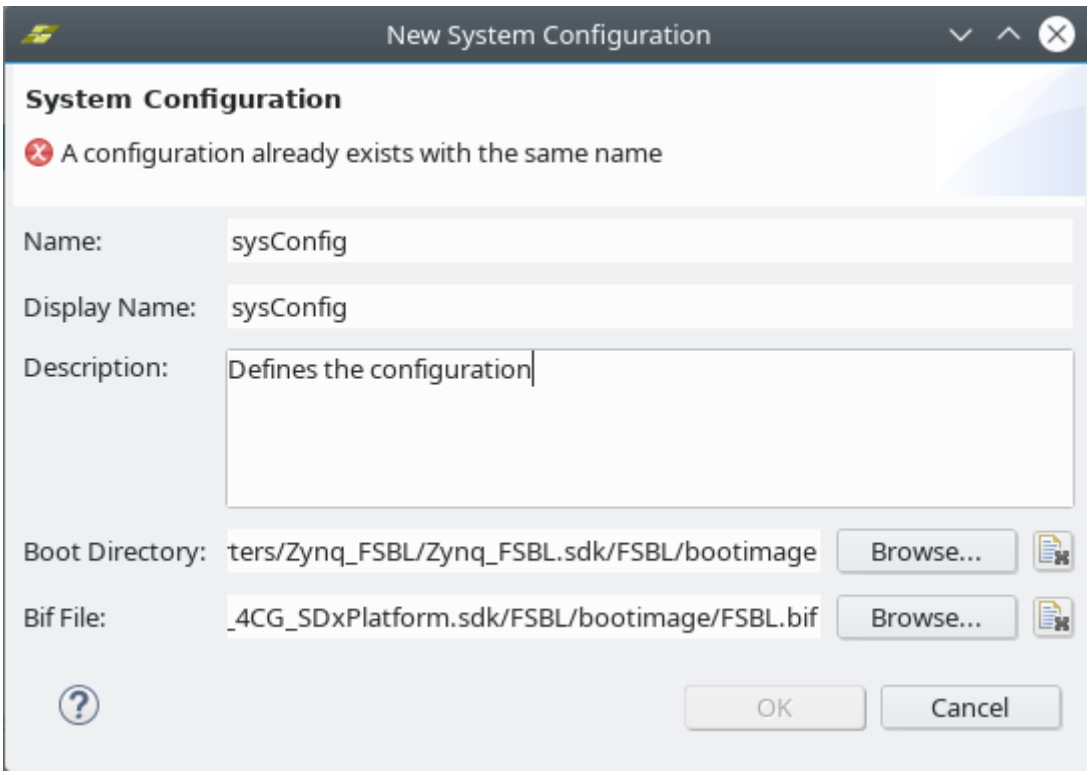


Figure 43 - System Configuration

Do the same with a new Domain.

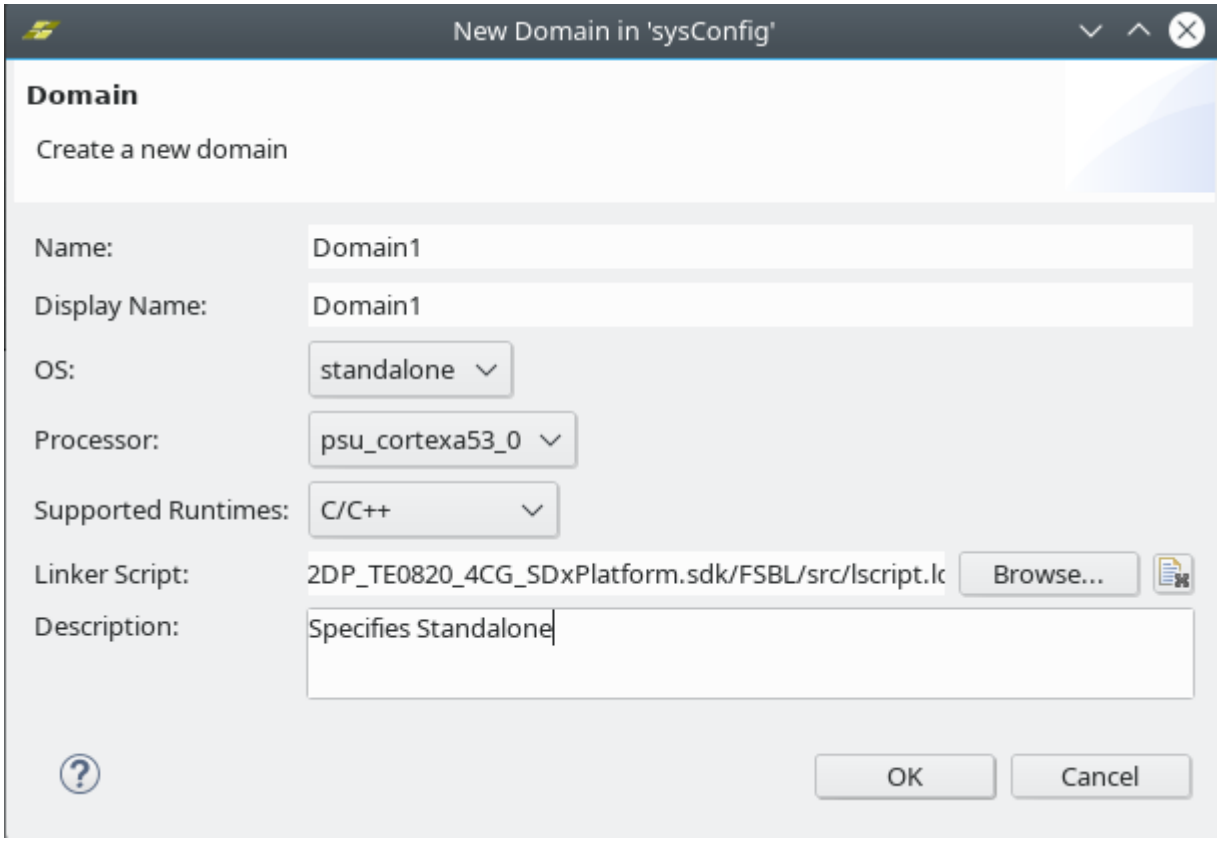


Figure 44 - Domain Configuration

On “Domain”, add at “Repositories”, a path where examples from Xilinx are stored. They can be downloaded from GitHub:

https://github.com/Xilinx/SDSoC_Examples

Under “Domain”, there are some paths to include.

- At “Application Settings”, tell the tool where the linker script is. (Hello World Application path)
- At “Board Support Package”, include the path of the Hello World .bsp folder
- At “Libraries”, include the lib folders of both FSBL and Hello World projects.

The icon that looks like a hammer, (Platform icon), click and select “Generate”. This will generate the platform files.

On the same icon, select “Add To Custom Repositories”, and the platform will be added to SDSoC.

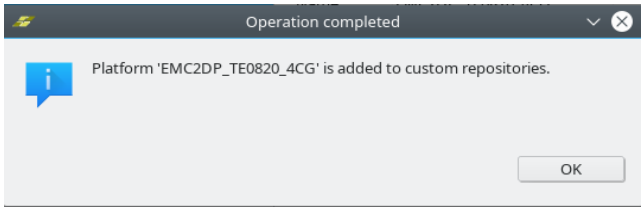


Figure 45 - Custom Platform added

Create a new project, and this time select Application Project.

Figure 46 - Create Application Project

Select the custom platform.

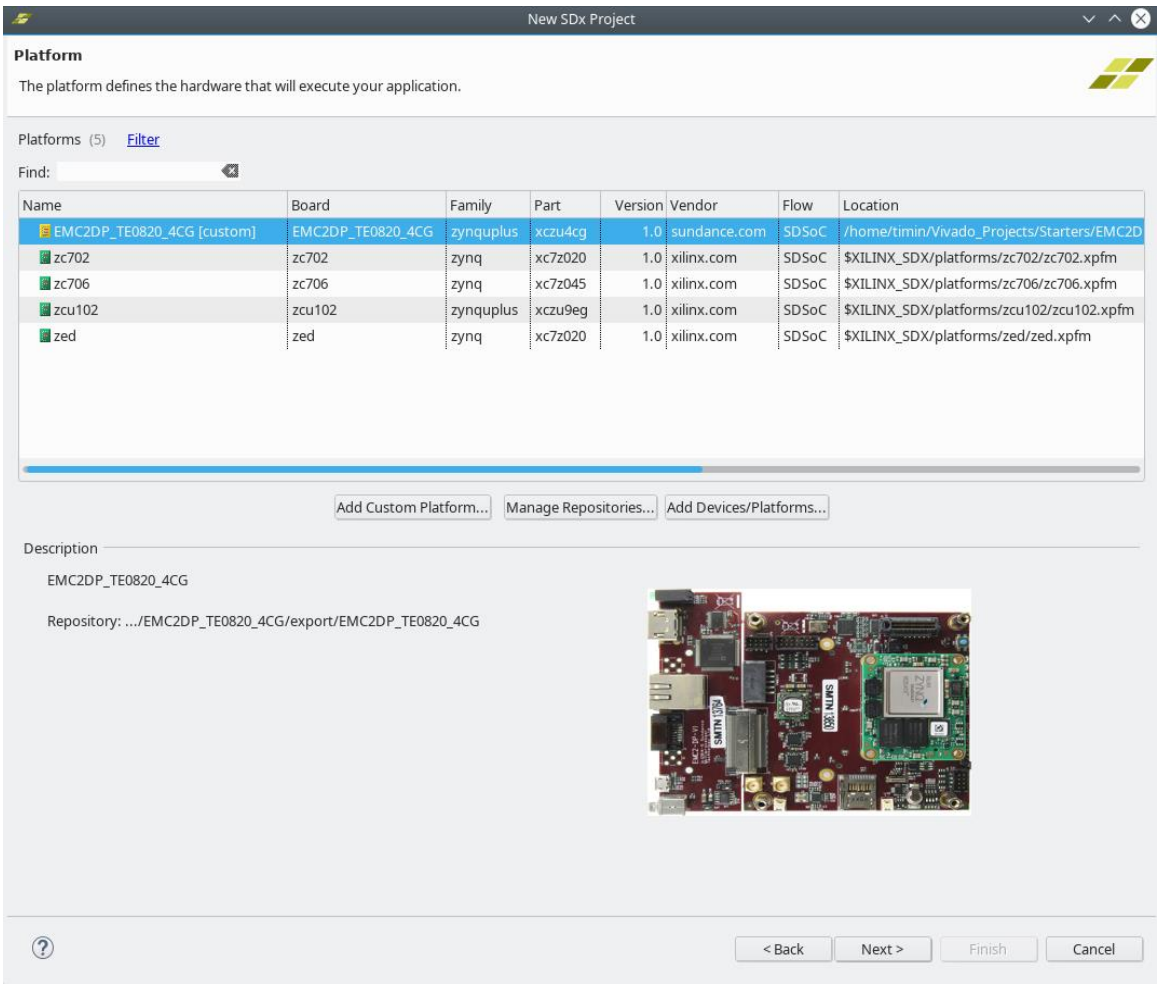


Figure 47 - Select Custom Platform

The repositories included now appear as possible application templates. Select the matrix multiplication example to accelerate in hardware.

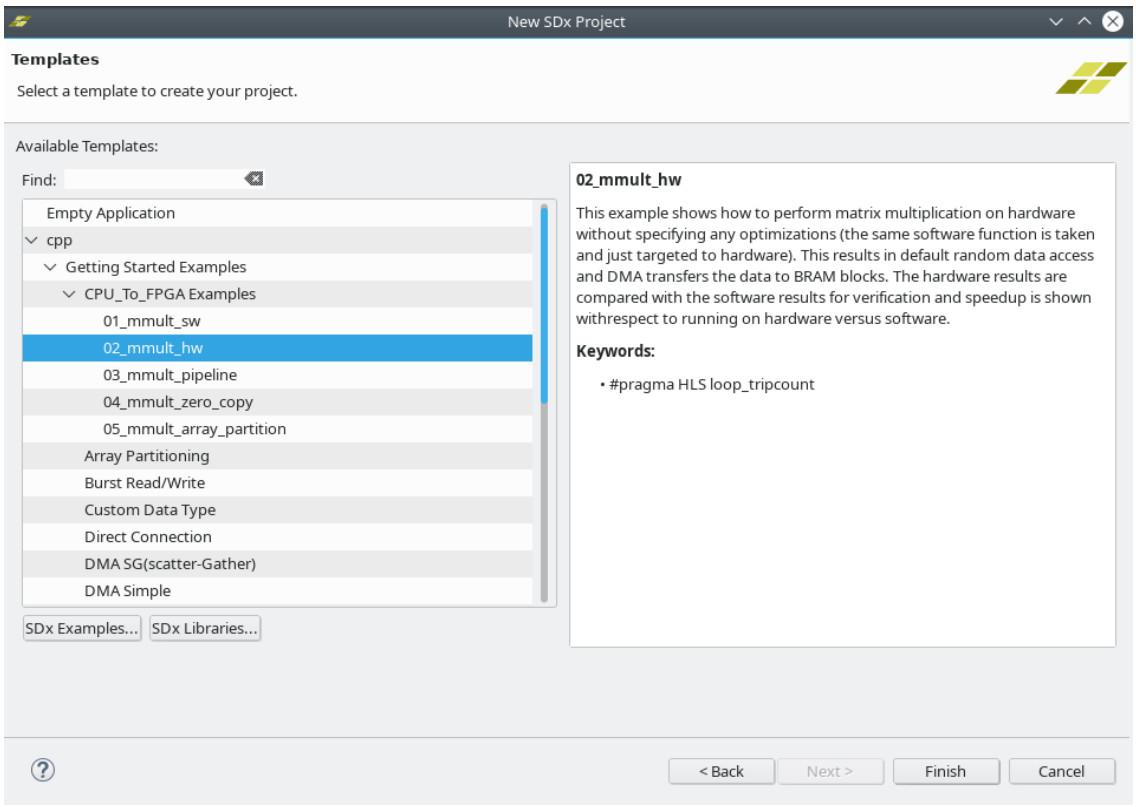


Figure 48 - Application templates

An application project is created, and ready to build. Notice that under “Hardware Functions”, the user can select which functions will be accelerated in hardware. (PL)

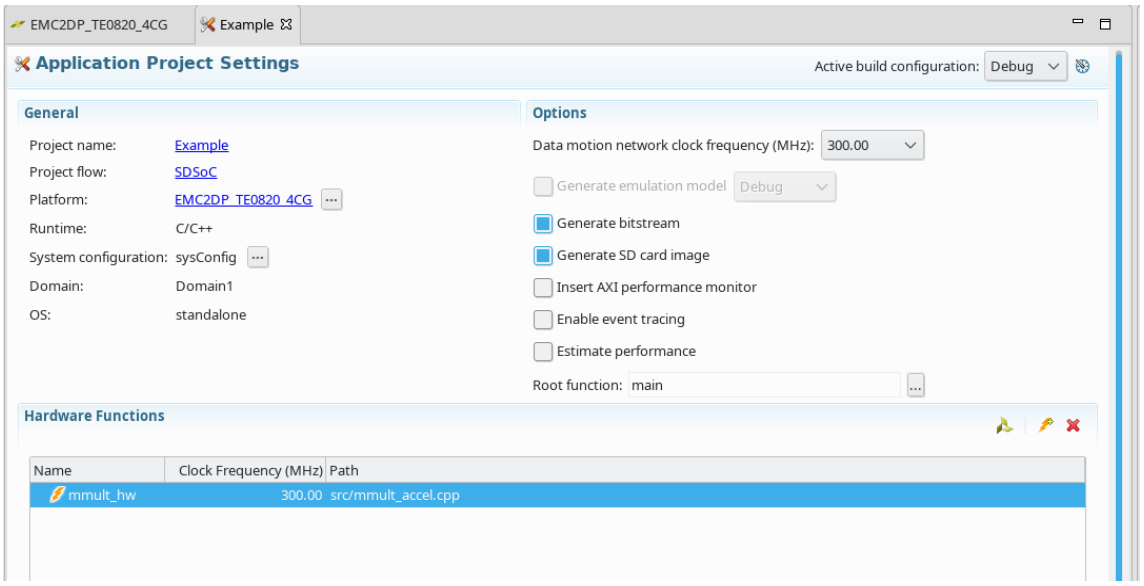


Figure 49 - Application window, accelerating function

At the Application Project window, select “Release” at “Active build configuration”, and, if desired, select “Estimate Performance”.

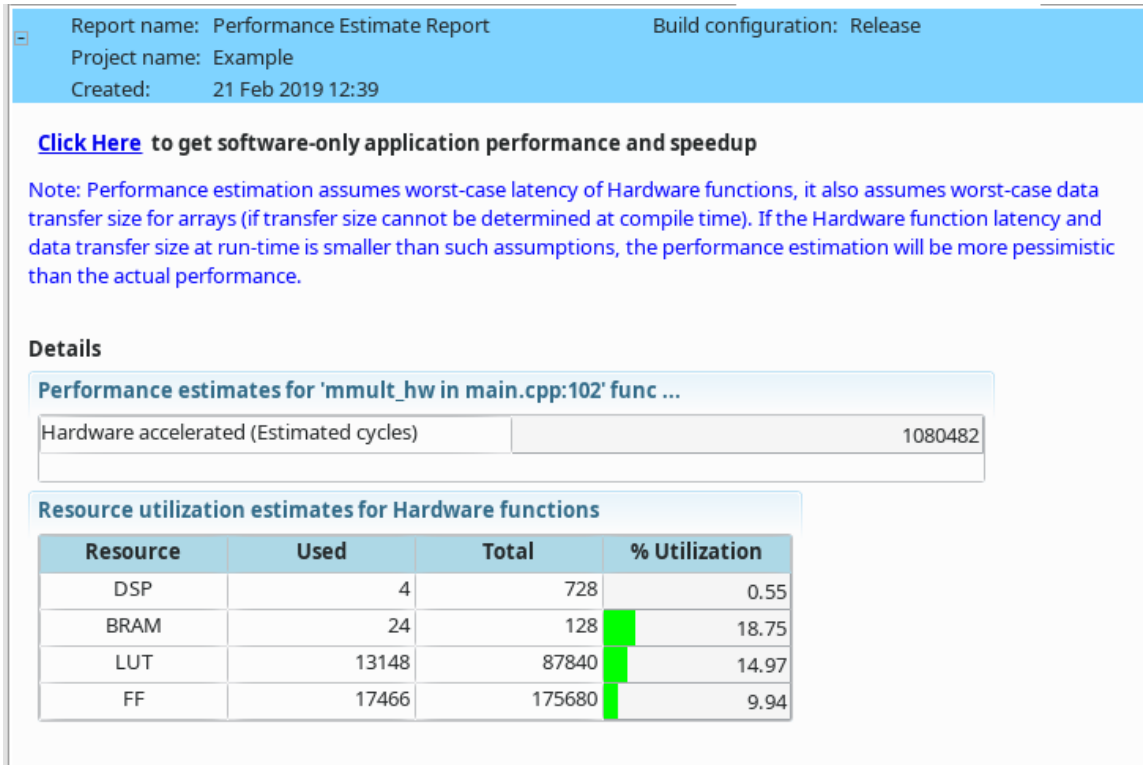
Building might take a long time, as synthesis and implementation is done. In order to avoid this waiting time, prebuilt files can be set up for the custom platform.

Build the project, and bootable files to place in SD card will be generated.

For Linux hosts, it might happen that “gmake” is not installed by default. In order to solve it, just link make to gmake:

sudo ln -s /usr/bin/make /usr/bin/gmake

As it can be seen:



Report name: Performance Estimate Report Build configuration: Release
Project name: Example
Created: 21 Feb 2019 12:39

Click Here to get software-only application performance and speedup

Note: Performance estimation assumes worst-case latency of Hardware functions, it also assumes worst-case data transfer size for arrays (if transfer size cannot be determined at compile time). If the Hardware function latency and data transfer size at run-time is smaller than such assumptions, the performance estimation will be more pessimistic than the actual performance.

Details

Performance estimates for 'mmult_hw in main.cpp:102' func ...

Hardware accelerated (Estimated cycles)	1080482

Resource utilization estimates for Hardware functions

Resource	Used	Total	% Utilization
DSP	4	728	0.55
BRAM	24	128	18.75
LUT	13148	87840	14.97
FF	17466	175680	9.94

Figure 50 - Estimation Performance

3.6 How can I create/run Sundance demos?

Sundance has developed some demos, using HDMI in/out FMC cards, HDMI output interface from the carrier, Camera Link, Applications for robotics applications using VCS-1, etc., for different architectures as Zynq 7 series or Zynq Ultrascale+, which can be found at:

Hackaday projects: https://hackaday.io/SundanceDotCom

GitHub: https://github.com/SundanceMultiprocessorTechnology

Contact Sundance for support