Application Note

ÚTIA Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Support for TE0821 Modules in Vitis 2023.2, AI 3.5 SW, AI 3.0 DPUCZDX8G

Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout,
kadlec@utia.cas.cz, zdenek.pohl@utia.cas.cz, kohoutl@utia.cas.cz

Revision history

Rev.	Date	Author	Description
v01	23.12.2024	J.K	Initial release
v02	15.02.2025	J.K	Update to support no-prebuild bring-up package
v03	16.02.2025	J.K	Update of references

Contents

Acknowledgement

The EECONE project is supported by the Chips Joint Undertaking and its members, including the top-up funding by National Funding Authorities from involved countries under grant agreement no. 101112065.

https://zs.utia.cas.cz/index.php?ids=projects/eecone

https://eecone.com/eecone/home/

1 Introduction

EECONE project *https://eecone.com/eecone/home/* work package 4, task 4.3 is investigating measures to support second life of electronics due to modular design.

Work package 4 task 4.4 is investigating measures to support extension of life of electronics due to methodology of support used custom platform to adapt for the in-time-evolving design tools and embedded Linux PetaLinux operating system.

UTIA AV CR, v.v.i. (Institute of Information Theory and Automation of the Czech Academy of Sciences, in short UTIA) is not-for profit research institute located in Prague, Czech Republic. UTIA is involved as partner in both tasks, T4.3 and T4.4.

Both EECONE task require specification of comparable reference systems which are based on modular HW with potential for “second life” by reuse of modules or use cost optimized PCB HW without modularity.

Systems (with HW modularity or low cost single PCB) should be capable to perform similar challenging tasks. Systems have to be capable to accelerate in HW AI inference algorithms with video camera input for edge application like person detection, face detection, car-make or car-type detection and graphical output to local display or to the remote PC connected by wired Ethernet in a local network.

Systems should also support monitoring and control from remote PC connected by wired Ethernet in a local network.

The investigated measures and methodologies to support “second life” of HW modules are studied in T4.3. Measures to support extension of life of electronics by methodology of support for SW custom-platform for the in-time-evolving design tools and embedded Linux PetaLinux operating system are studied in T4.4.. We target support for developers designing embedded systems for commercial, AI inference-based edge-applications, mainly in the area of home automation.

Based on this framework and requirements, UTIA selected two types of systems:

- Low cost systems. See [2] and [3].
- Modul based systems. See [4], [8] and [5], [9].

[25] is this application note.

Both compared types of systems use STMicroelectronic STM32H573I-DK board for:

- Local system control on small graphical touch screen display
- Remote system control from www browser based on www-server or secure communication based on mqtt client. Board is supported by STMicroelectronic CubeMX SW framework and also by NetXDuo SW framework on top of ThreadX OS and FileX SW package.

See [1], [21] for the application note describing how to cpmpile www server SW project STMicroelectronic STM32H573I-DK kit.

The MCU used on STM32H573I-DK board [1], [7] is a 40nm chip with 32 bit ARM M33 MCU operating with 250 MHz clock, 2 MBytes of program flash memory and 640 KBytes of RAM.

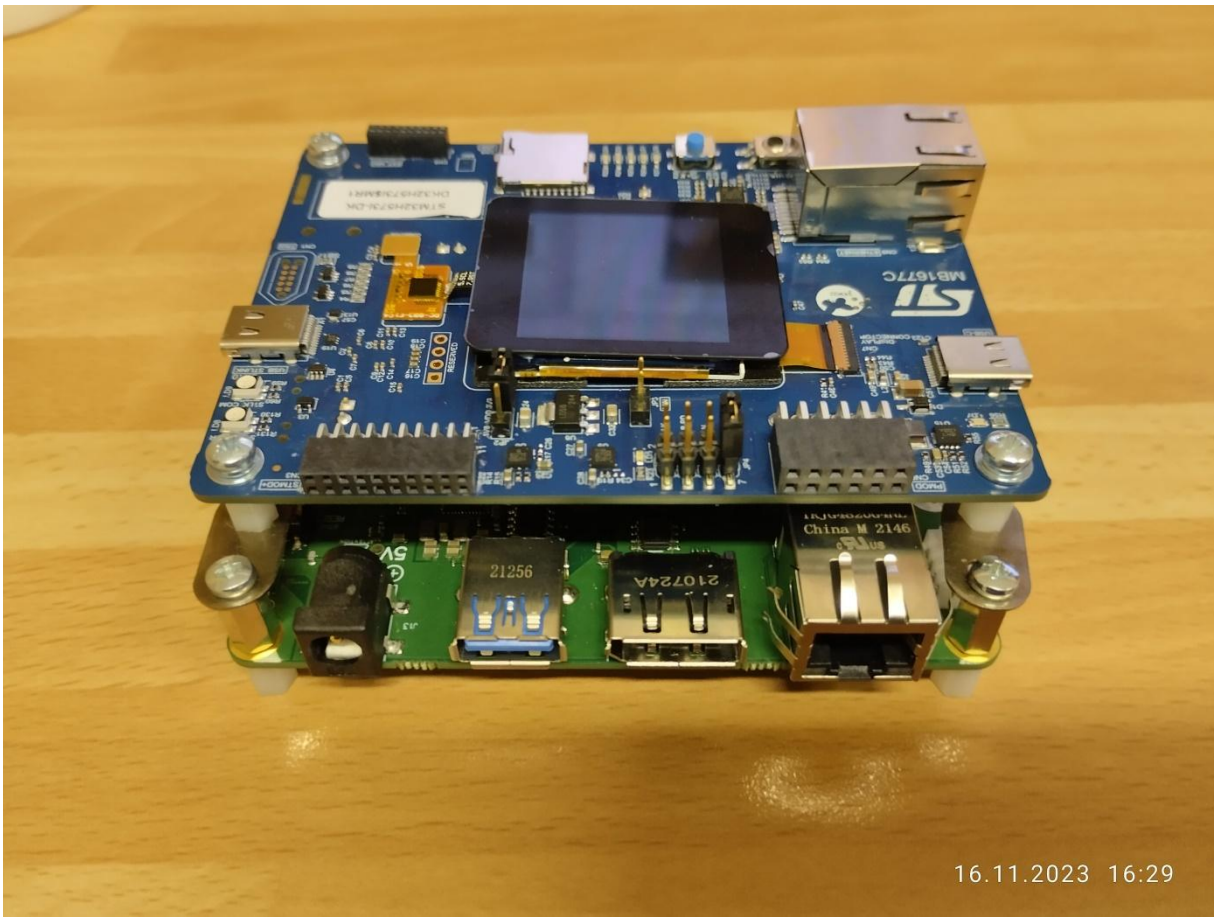
Compared systems [2], [3], [4], [5], [8], [9] use 16nm AMD ZynqUltrascale+ device with 64 bit ARM A53 microprocessor and programmable logic in the same device. Systems support Petalinux linux operatimg system.

- Low-cost systems have the fixed size AMD ZynqUltrascale+ device and DDR4 with all peripheral interfaces soldered on a single, low cost PCB.
- Module-based systems have an AMD ZynqUltrascale+ device and the DDR4 located on a 4x5cm module connected by connectors to a carrier board with all peripheral interfaces.

Different modules have different configurations of the AMD device and DDR4 memory.

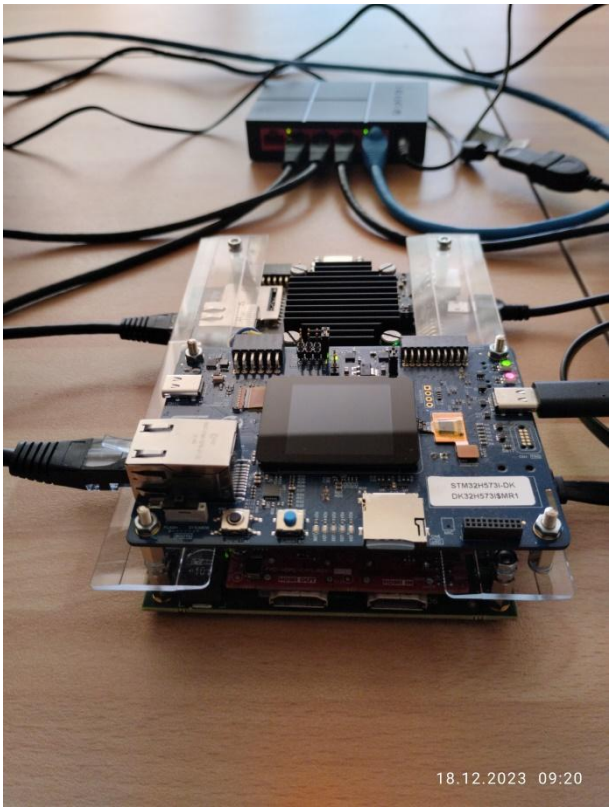
1.1 Low-cost systems used by UTIA in EECONE T4.3 and T4.4

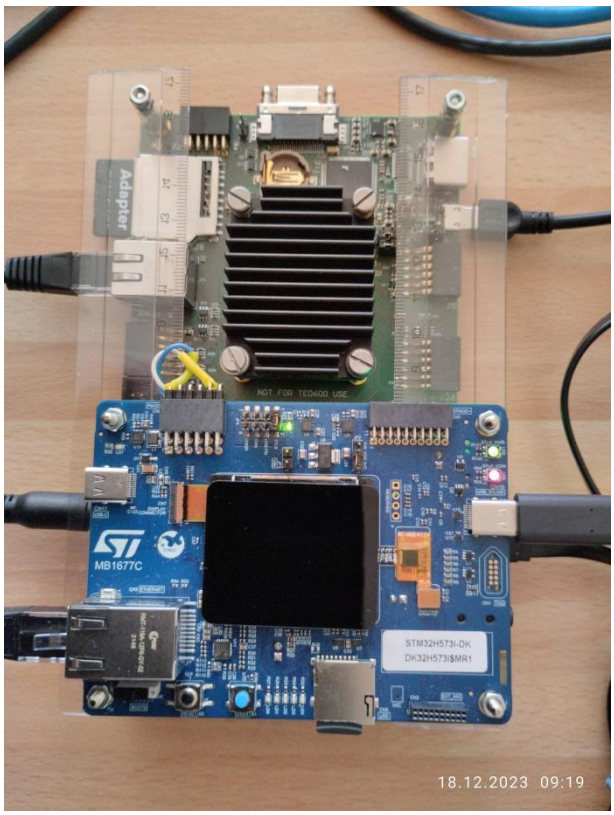
[1] **[7]**	STM32H573I-DK	https://www.st.com/en/evaluation-tools/stm32h573i-dk.html	Local or remote system control (www-server or secure mqtt client) for [2], [3]
[2]	TE0802-02-1BEV2-A	https://shop.trenz-electronic.de/en/TE0802-02-1BEV2-A-MPSoC-Development-Board-with-AMD-Zynq-UltraScale-ZU1EG-and-1-GB-LPDDR4?c=474	AMD Vitis AI 3.0 AMD DPU in PL USB camera HD VGA display or remote X11 desktop
[3]	TE0802-02-2AEV2-A	MPSoC Development Board mit AMD Zynq™ UltraScale+™ ZU2 und 1 GB LPDDR4 | Trenz Electronic GmbH Online Shop (EN) (trenz-electronic.de)	AMD Vitis AI 3.0 AMD DPU in PL USB camera HD VGA display or remote X11 desktop



1.2 Module-based systems used by UTIA in EECONE T4.3 and T4.4

[1] [7]	STM32H573I-DK TE0701-06 Carrier Board for Trenz Electronic 4 x 5 Modules TE0821 or TE0821	https://www.st.com/en/evaluation-tools/stm32h573i-dk.html https://shop.trenz-electronic.de/en/TE0701-06-Carrier-Board-for-Trenz-Electronic-4-x-5-Modules?c=261	Local or remote system control (www-server or secure mqtt client) for [4], [5] Carrier Board for range of 4x5 cm modules [4], [5].
[4] [8]	TE0821 Module: **17 module types** **24 module types**	https://shop.trenz-electronic.de/en/Products/Trenz-Electronic/TE08XX-Zynq-UltraScale/TE0821-Zynq-UltraScale/	AMD Vitis AI 3.0 AMD DPU in PL USB camera remote X11 desktop
[5] [9]	TE0820 Module: **115 module types** **119 module types**	https://shop.trenz-electronic.de/en/Products/Trenz-Electronic/TE08XX-Zynq-UltraScale/TE0820-Zynq-UltraScale/	AMD Vitis AI 3.0 AMD DPU in PL USB camera remote X11 desktop





This application note [4] and the accompanying evaluation package describe support for systems based on TE0821 modules. It is available for free public download from UTIA server dedicated to UTIA contributions to EECONE project:
https://zs.utia.cas.cz/index.php?ids=projects/eecone

It will be also available for free public download in format of an wiki tutorial on Trenz-Electronic wiki server:
https://wiki.trenz-electronic.de/display/PD/Vitis+AI+and+Vitis+Acceleration+Tutorials+with+Trenz+Electronic+Modules

1.3 Objective of This Application Note and Evaluation Package

This application note and the accompanying evaluation package describe system [**4**]**.**

This application note describes how to design custom HW platform with AMD DPU for Vitis 2023.2 AI 3.0 inference for family of Trenz Electronic modules TE0821 with AMD Zynq Ultrascale+ device.

This application note [24] is using AMD Vitis 2023.2 and PetaLinux 2023.2 tools installed on Ubuntu 20.04. The described configuration integrated AMD DPU IP, version v4.1.0, with architecture DPUCZDX8G present in Vitis AI 3.0 distribution with corresponding AI models.

Described board configuration can operate as small standalone computer with 1 Gb Ethernet connectivity, and remote X11 desktop. Support package for this application note will be available for public download from [24].

The installed AMD DPU configurations require recompilation of Vitis AI 3.0 examples and inference models in the Vitis AI framework. This compilation process will be described in separate application note [26].

This application supports family of TE0821 modules listed in next tables with ID 1 to 17.

Vitis AI 3.0 sample application results are presented for these modules:

- ID=1 module: TE0821-01-2AE31KA, device xczu2cg-sfvc784-1-e, 4GB DDR4
- ID=3 module: TE0821-01-3AE31KA, device xczu3cg-sfvc784-1-e, 4GB DDR4
- ID=5 module: TE0821-01-3BE21FA, device xczu3eg-sfvc784-1-e, 2GB DDR4

Module TE0821-01-2AE31KA has two A53 ARM cores and 4 GB DDR4. It is possible to implement configurations of the AMD DPU (from B512 up to B1024) in PL. Implementaton of B1024 is demonstrated.

Module TE0821-01-3AE31KA has two A53 ARM cores and 4 GB DDR4. It is possible to implement configurations of the AMD DPU (from B512 up to B1600) in PL. Implementaton of B1600 is demonstrated.

Module TE0821-01-3BE21FA has four A53 ARM cores and 2 GB DDR4. It is possible to implement configurations of the AMD DPU (from B512 up to B1600) in PL. Implementaton of B1600 is demonstrated.

Specification for each module ID defined in `TE0821_board_files.csv` file is input to the Vivado 2023.2 HW bring-up scripts. Is provided by company Trenz Electronic. It is part of the package provided for complete family of modules TE0821 for AMD Vivado 2023.2.

List of supported TE0821 modules is reprinted from the `TE0821_board_files.csv` file included in the evaluation package associated of this application note.

This application note and associated evaluation package enables support for “second-life” of 24 types of TE0821 modules.

CSV_VERSION=1.4

#Comment:-do not change matrix position or remove CSV_VERSION:

ID	PRODID	PARTNAME	BOARDNAME	SHORTNAME
1	TE0821-01-2AE31KA	xczu2cg-sfvc784-1-e	trenz.biz:te0821_2cg_1e:part0:3.0	2cg_1e_4gb_dd
2	TE0821-01-2AE31PA	xczu2cg-sfvc784-1-e	trenz.biz:te0821_2cg_1e:part0:3.0	2cg_1e_4gb_dd
3	TE0821-01-3AE31KA	xczu3cg-sfvc784-1-e	trenz.biz:te0821_3cg_1e:part0:3.0	3cg_1e_4gb_dd
4	TE0821-01-3AE31PA	xczu3cg-sfvc784-1-e	trenz.biz:te0821_3cg_1e:part0:3.0	3cg_1e_4gb_dd
5	TE0821-01-3BE21FA	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
6	TE0821-01-3BE21FC	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
7	TE0821-01-3BE21FL	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
8	TE0821-01-3BE21MA	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
9	TE0821-01-3BE21ML	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
10	TE0821-01-3BI21FA	xczu3eg-sfvc784-1-i	trenz.biz:te0821_3eg_1i:part0:2.0	3eg_1i_2gb
11	TE0821-01-3BI21FL	xczu3eg-sfvc784-1-i	trenz.biz:te0821_3eg_1i:part0:2.0	3eg_1i_2gb
12	TE0821-01-3BI21MA	xczu3eg-sfvc784-1-i	trenz.biz:te0821_3eg_1i:part0:2.0	3eg_1i_2gb
13	TE0821-01-3BI91ND	xczu3eg-sfvc784-1-i	trenz.biz:te0821_3eg_1i:part0:3.0	3eg_1e_4gb
14	TE0821-01-4DE31FL	xczu4ev-sfvc784-1-e	trenz.biz:te0821_4ev_1e:part0:3.0	4ev_1e_4gb_dd
15	TE0821-01-4DE31ML	xczu4ev-sfvc784-1-e	trenz.biz:te0821_4ev_1e:part0:3.0	4ev_1e_4gb_dd
16	TE0821-01-S003	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
17	TE0821-01-S004	xczu3cg-sfvc784-1-i	trenz.biz:te0821_3cg_1i:part0:2.0	3cg_1i_2gb
18	TE0821-02-3AI81MA	xczu3cg-sfvc784-1-i	trenz.biz:te0821_3cg_1i:part0:2.0	3cg_1i_2gb
19	TE0821-02-3BE81MA	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
20	TE0821-02-3BE81ML	xczu3eg-sfvc784-1-e	trenz.biz:te0821_3eg_1e:part0:2.0	3eg_1e_2gb
21	TE0821-02-3BI81MA	xczu3eg-sfvc784-1-i	trenz.biz:te0821_3eg_1i:part0:2.0	3eg_1i_2gb
22	TE0821-02-2AE91PA	xczu2cg-sfvc784-1-e	trenz.biz:te0821_2cg_1e:part0:4.0	2cg_1e_4gb
23	TE0821-02-3AE91PA	xczu3cg-sfvc784-1-e	trenz.biz:te0821_3cg_1e:part0:4.0	3cg_1e_4gb
24	TE0821-02-4DE91ML	xczu4ev-sfvc784-1-e	trenz.biz:te0821_4ev_1e:part0:4.0	4ev_1e_4gb

Supported modules with ID = 1 ... 24

Modules from this list mighed have been used originally in another context. That context might become obsolete or outdated, now. We provide support for reuse of such modules again in a large and challenging range of Vitis 2022 AI 3.0 [4], [6] and Vitis 2023.2 A 3.5 [24], [26] applications.

Time evolution of HW modules TE0821 [4], [24]:

Current HW Module	Predecessor HW Modules
TE0821-02-4DE91ML	TE0821-01-4DE31ML TE0821-01-4DE31FL
TE0821-02-3BI81MA	TE0821-01-3BI21MA TE0821-01-3BI21FA
TE0821-02-3BE81ML	TE0821-01-3BE21ML TE0821-01-3BE21FL TE0821-01-03EG-1EL
TE0821-02-3BE81MA	TE0821-01-3BE21MA TE0821-01-3BE21FA
TE0821-02-3AE91PA	TE0821-01-3AE31PA TE0821-01-3AE31KA
TE0821-02-2AE91PA	TE0821-01-2AE31PA TE0821-01-2AE31KA
TE0821-02-3BE81MC	TE0821-01-3BE21MC
TE0821-02-3BI91ND	TE0821-01-3BI91ND

HW changes are described in:

PCN-20190712 TE0821-01-_ SPI Flash Change and eMMC Change-v11-02_16_2021.pdf	16.02.2021
PCN-20210616 TE0821-01 eMMC Change and Product Update.pdf	27.09.2021
PCN-20240207 TE0821-01 Pull-up resistor removal for signal MR.pdf	07.02.2024
Revised PCN-20240621a TE0821-01 to TE0821-02 Hardware Revision Change.pdf	08.10.2024

2 Prepare Reference Design for Extensible Custom Platform

The design proces is described on module with ID=3: TE0821-01-3AE31KA has two A53 ARM cores and 4 GB DDR4. If your module has different ID, replace 3 with that ID.

In Ubuntu terminal, source paths to Vitis and Vivado tools by

```
$ source /tools/Xilinx/Vitis/2023.2/settings64.sh
```

Download `te0821_test_board.zip` from the acompanying evaluation package Vitis 2023.2:

It contains bring-up scripts for creation of Petalinux for range of modules in the directory named “`test_board`”.

Unzip te0821_test_board.zip to directory:

```
~/work/TE0821_03_240
```

All supported modules are identified in file:

```
~/work/TE0821_03_240/test_board/board_files/TE0821_board_files.csv
```

- We will select module ID=3 with name TE0821-01-3AE31KA, device xczu3cg-sfvc784-1-e, 4GB DDR4

It will work on TE0701-06 carrier board. We will use default clock 240 MHz. That is why we name the package TE0821_03_240 and proposed to unzip the TE0821 test_board Linux Design files into the directory:

```
~/work/TE0821_03_240
```

2.1 Reference HW for TE0821 module

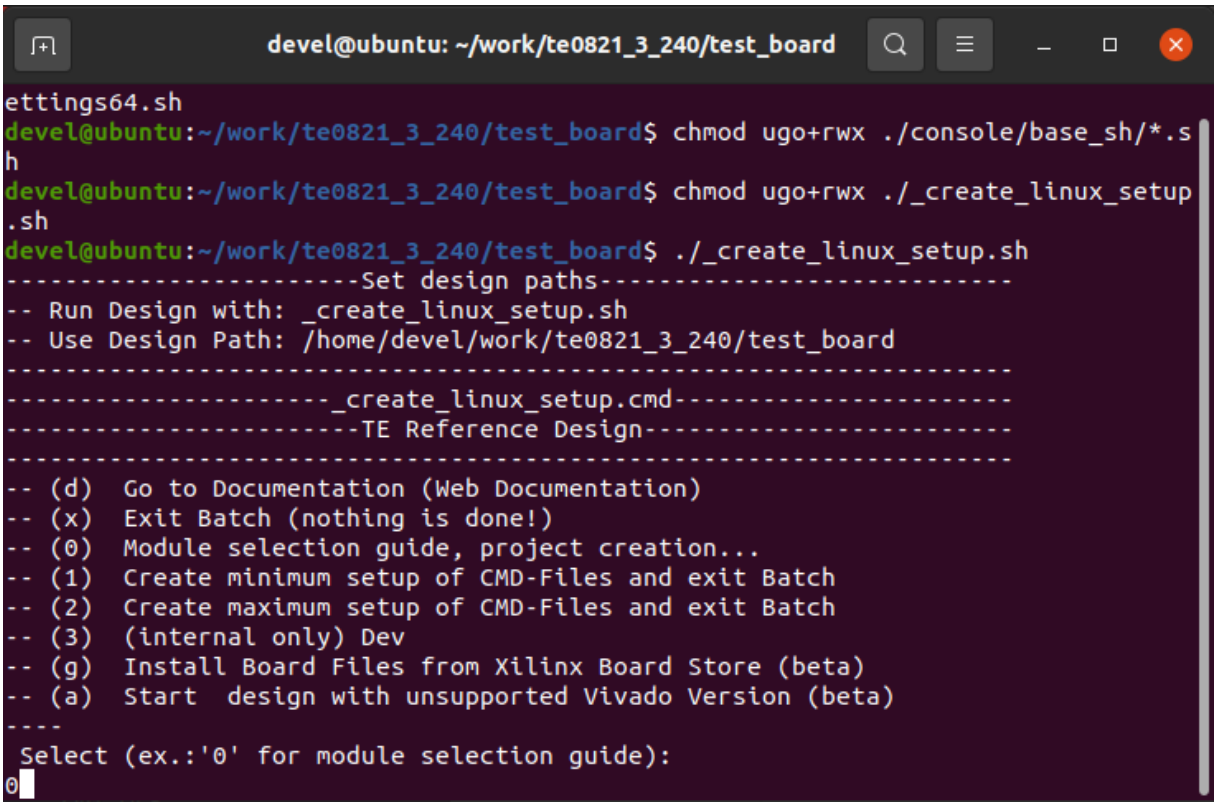
In Ubuntu terminal, change directory to the test_board directory:

```
$ cd ~/work/TE0821_03_240/test_board
```

Setup the test_board directory files for the Ubuntu 20.04 version of AMD Vitis 2023.2.1 tools. In Ubuntu terminal, execute:

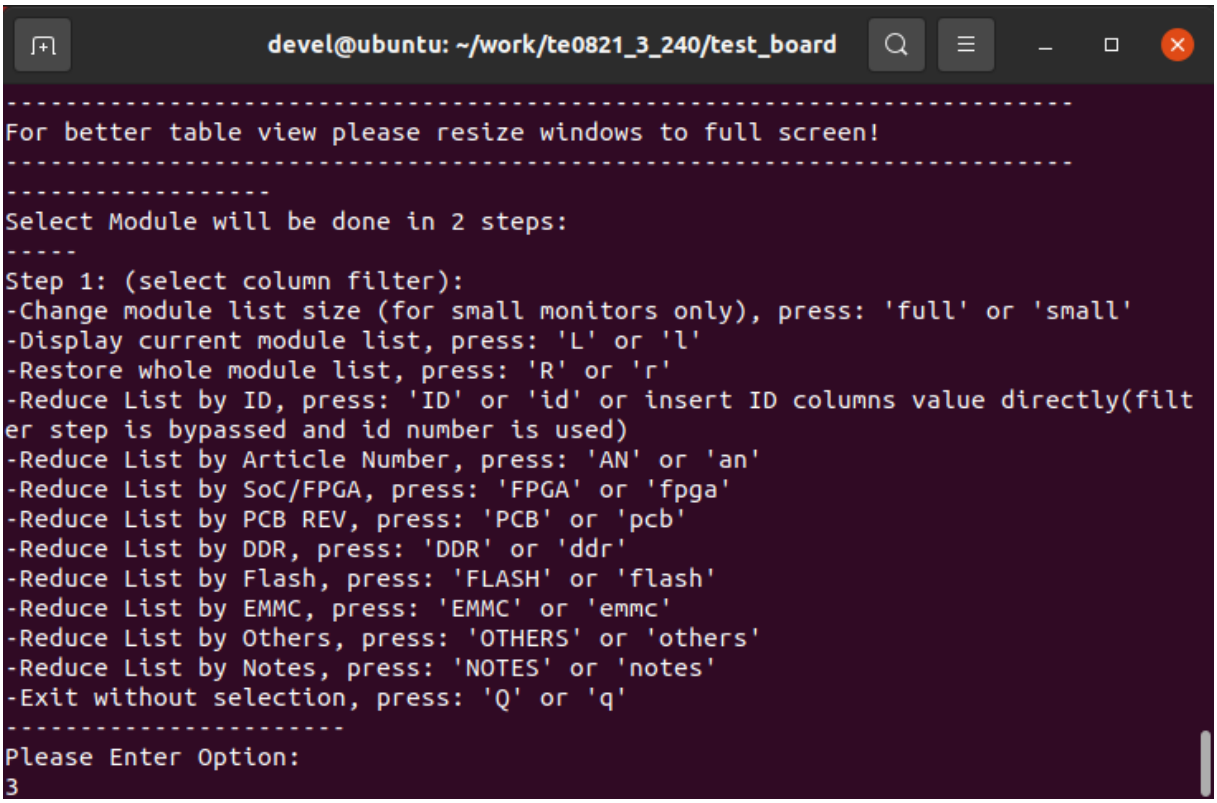
```
$ chmod ugo+rwx ./console/base_sh/*.sh

$ chmod ugo+rwx ./_create_linux_setup.sh

$ ./_create_linux_setup.sh
```

Select option (0) to open Selection Guide and press Enter



Select variant 3 from the selection guide, press enter and agree selection

Create Vivado Project with option 1



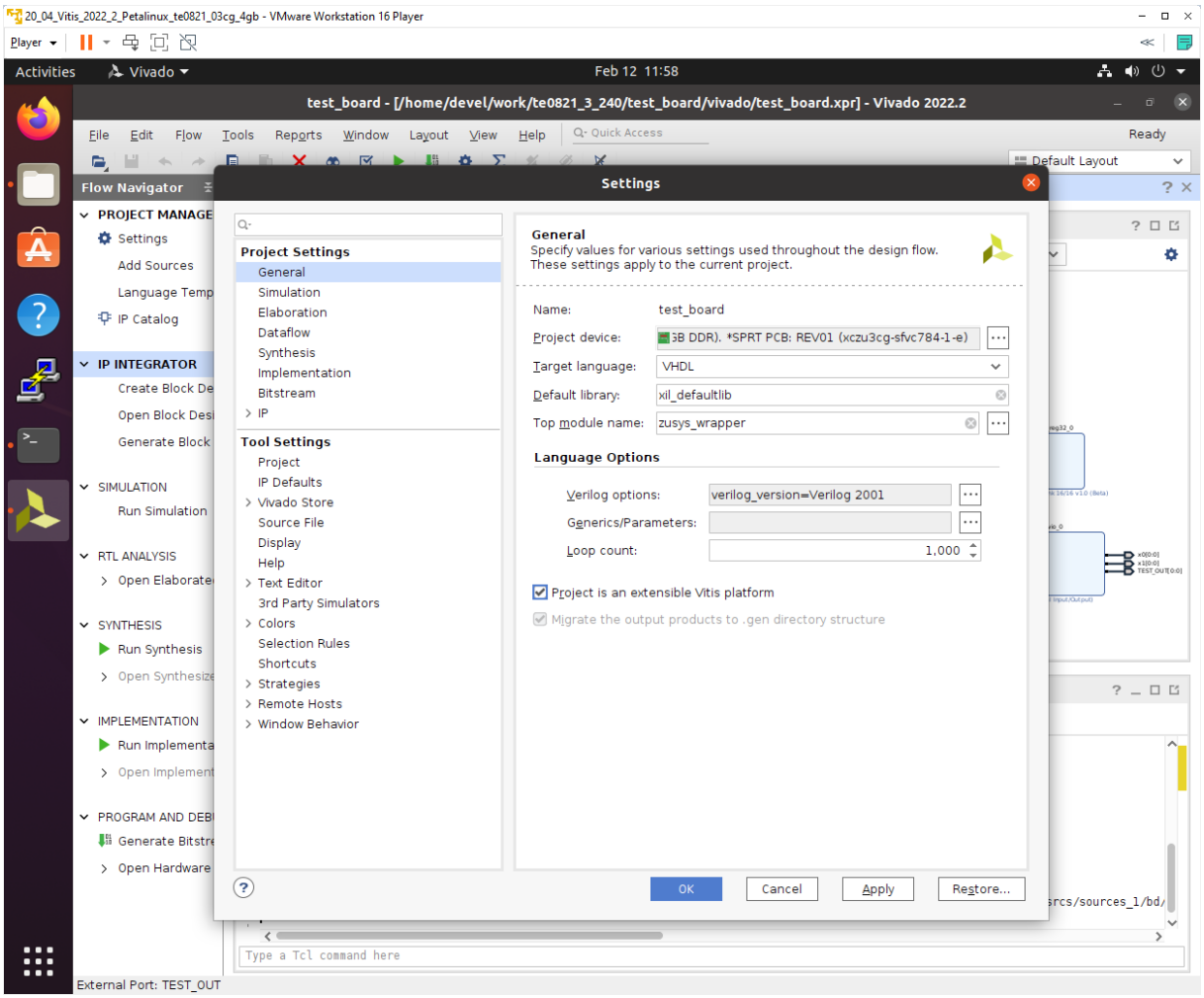
Vivado project will be generated for module with ID=3, name TE0821-01-3AE31KA, device xczu3cg-sfvc784-1-e, 4GB DDR4

3 HW support for Vitis Extensible Design Flow

3.1 Create Extensible Platform HW

This section describes manual creation of extensible platform HW in Vivado 20232.1. You can follow it or you can alternatively use the fast-track script described in Section 3.2 .

In Vivado project, click in **Flow Navigator** on **Settings**. In opened Settings window, select **General** in **Project Settings**, select **Project is an extensible Vitis platform**. Click on **OK**.

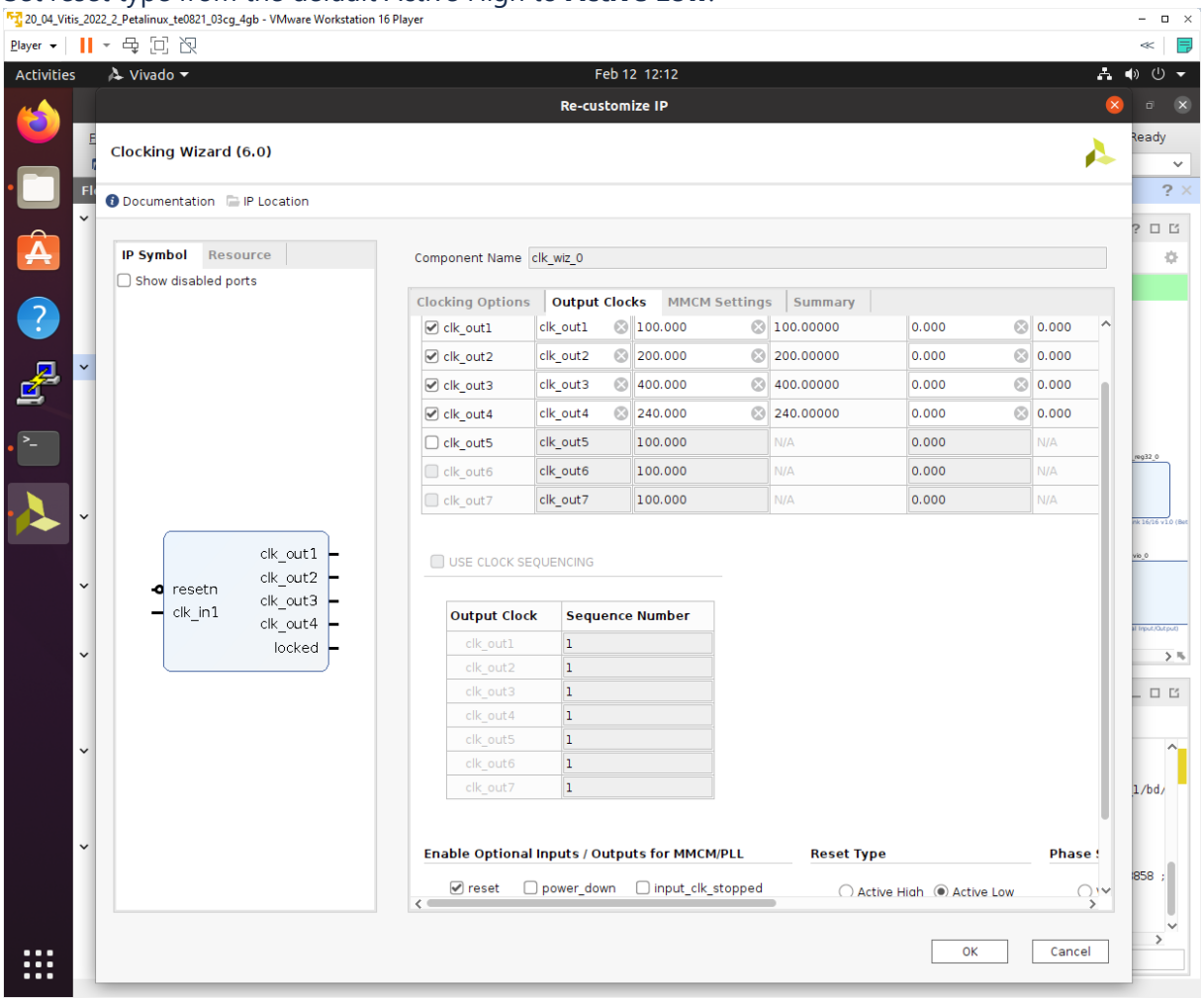


IP Integrator of project set up as an extensible Vitis platform has an additional Platform Setup window.

Add multiple clocks and processor system reset IPs
In IP Integrator Diagram Window, right click, select **Add IP** and add **Clocking Wizard** IP **clk_wiz_0**. Double-click on the IP to Re-customize IP window. Select Output Clocks panel. Select four clocks with frequency 100, 200, 400 and 240 MHz.
100 MHz clock will serve as low speed clock.
200 MHz and 400 MHz clock will serve as clock for possible AI engine.

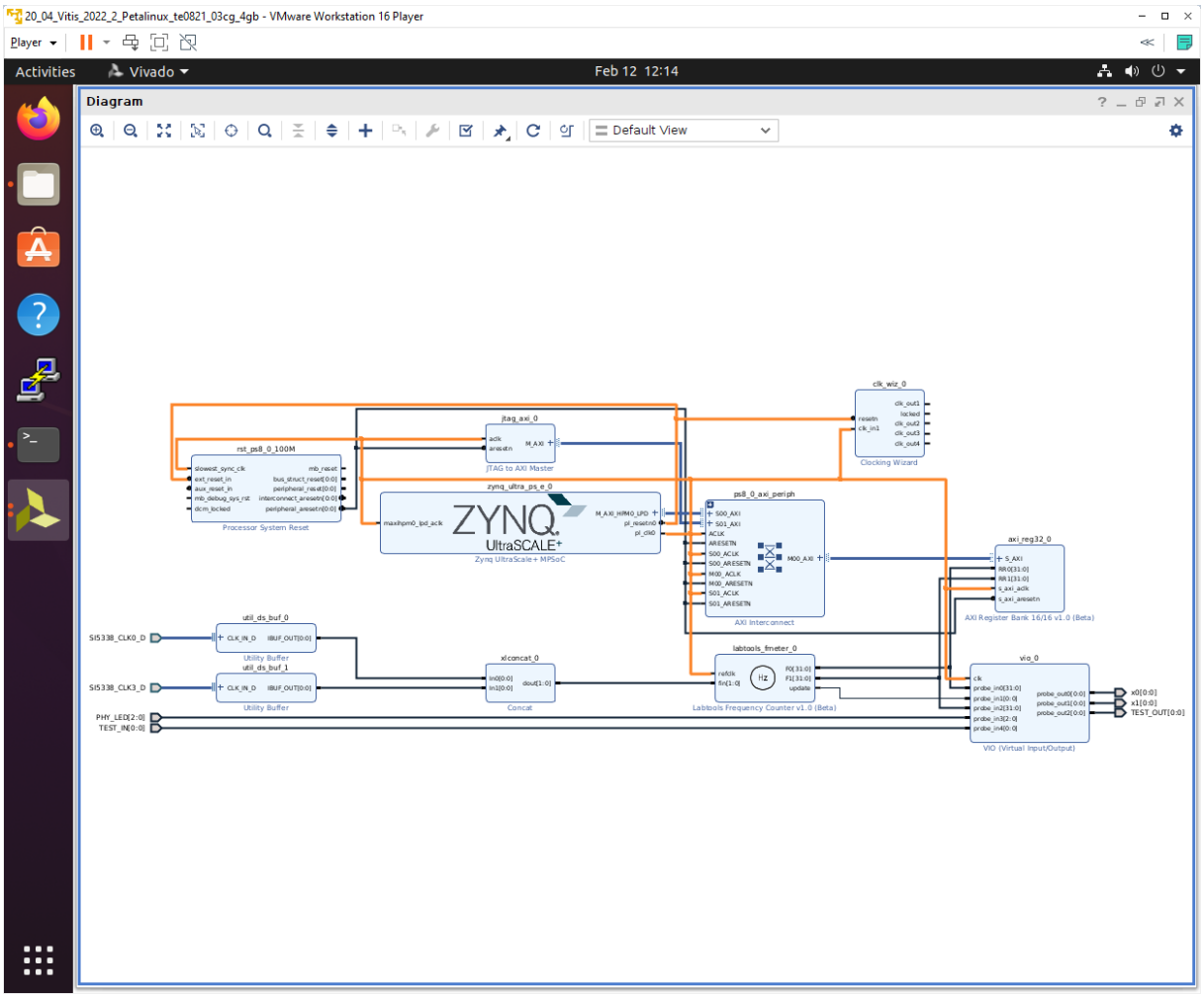
240 MHz clock will serve as the default extensible platform clock. By default, Vitis will compile HW IPs with this default clock.

Set reset type from the default Active High to **Active Low**.

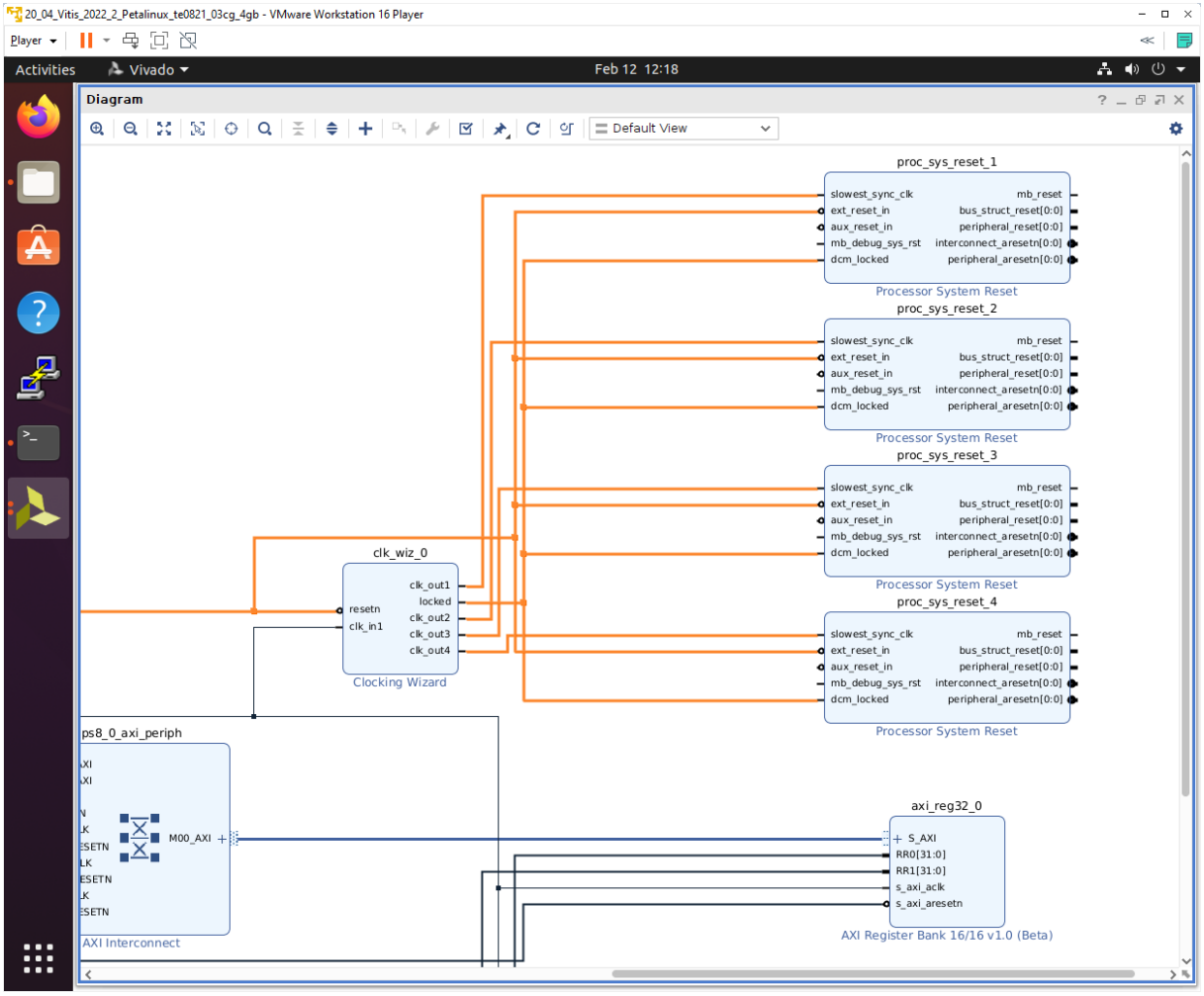


Clik on OK to close the Re-customize IP window.

Connect input **resetn** of **clk_wiz_0** with output **pl_resetn0** of **zynq_ultra_ps_e_0.**
Connect input **clk_in1** of **clk_wiz_0** with output **pl_clk0** of **zynq_ultra_ps_e_0**.



Add and connect four Processor System Reset blocks for each generated clock.

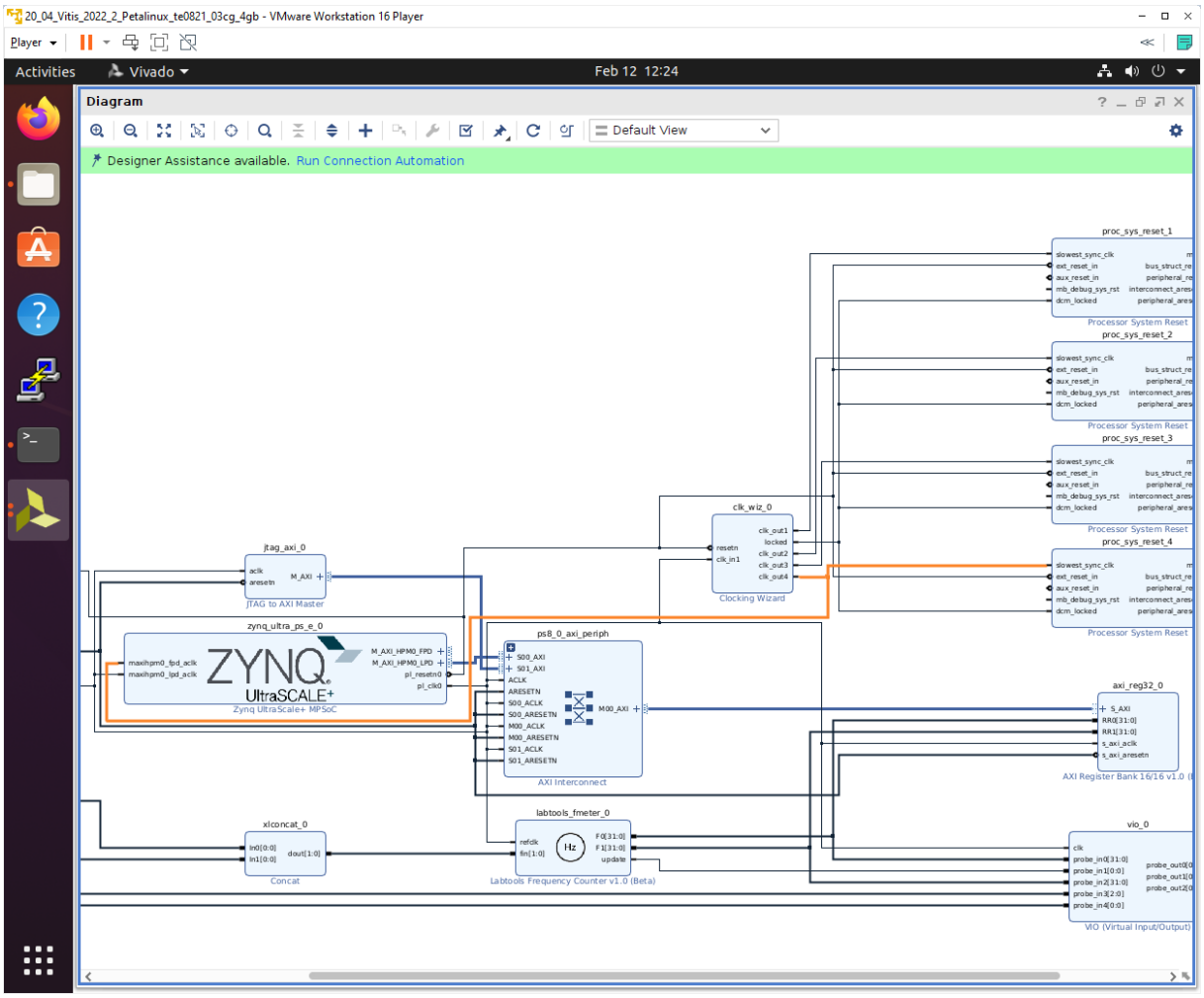


Open **Platform Setup** window of IP Integrator to define Clocks. In **Settings**, select **Clock**.

In “Enabled” column select all four defined clocks **clk_out1**, **clk_out2**, **clk_out3**, **clk_out4** of **clk_wiz_0** block.

In “ID” column keep the default Clock ID: **1, 2, 3, 4**

In “Is Default” column, select **clk_out4** (with ID=4) as the default clock. One and only one clock must be selected as default clock.



Double-click on **zynq_ultra_ps_e_0** block and enable **M_AXI_HPM0_FPD** port. Select data width 32bit. It will be used for integration of inerrupt controller on new dedicated AXI stream subsystem with 240 MHz clock. It will also enable new input pin **maxihpm0_fpd_aclk** of **zynq_ultra_ps_e_0.** Connect it to 240 MHz clock net.

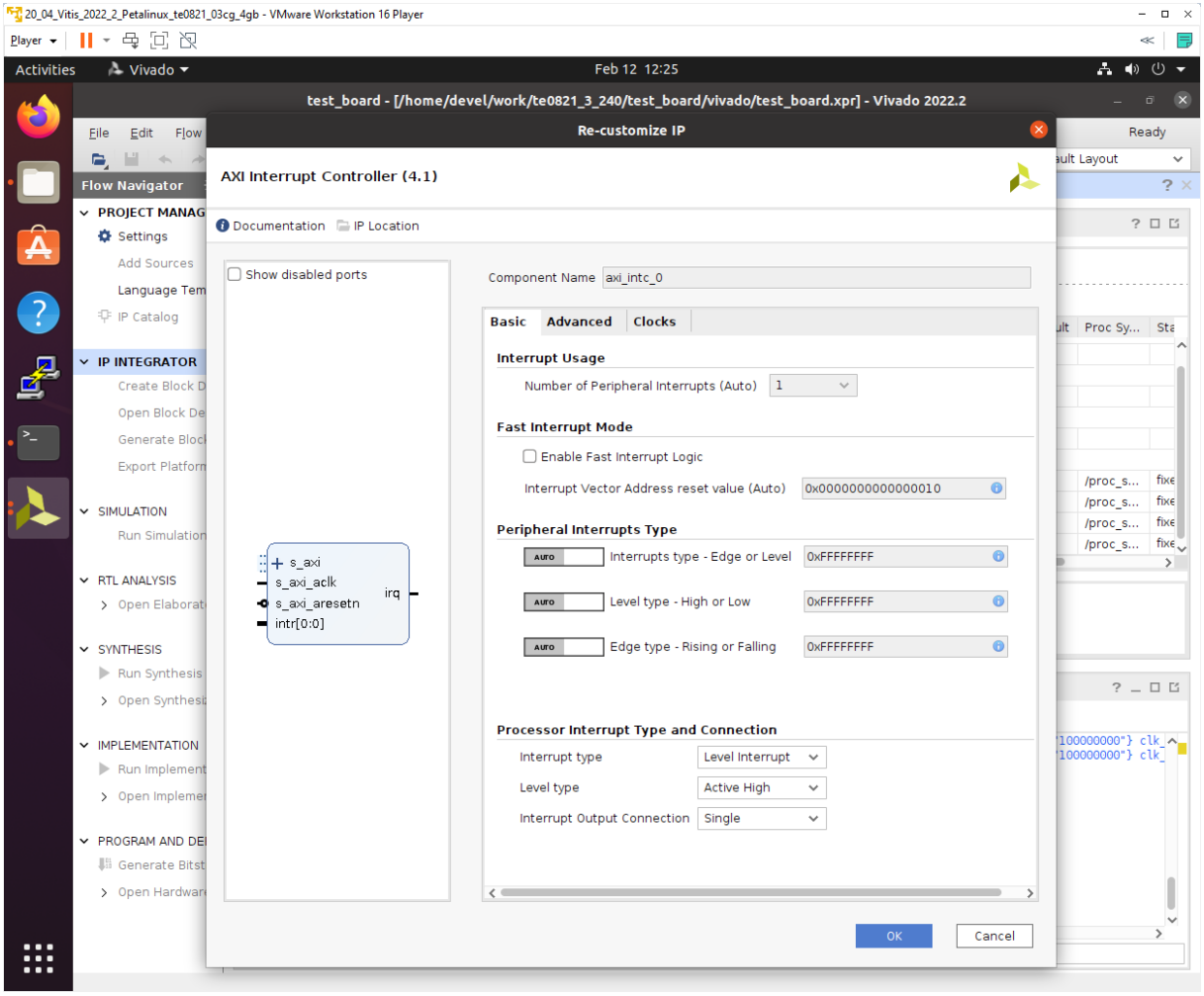
Connect input pin **maxihpm0_fpd_aclk** of **zynq_ultra_ps_e_0** to the 240 MHz **clk_out4** of **clk_wiz_0** IP block.

Add, customize and connect the AXI Interrupt Controller

Add AXI Interrupt Controller IP **axi_intc_0**.
Double-click on **axi_intc_0** to re-customize it.

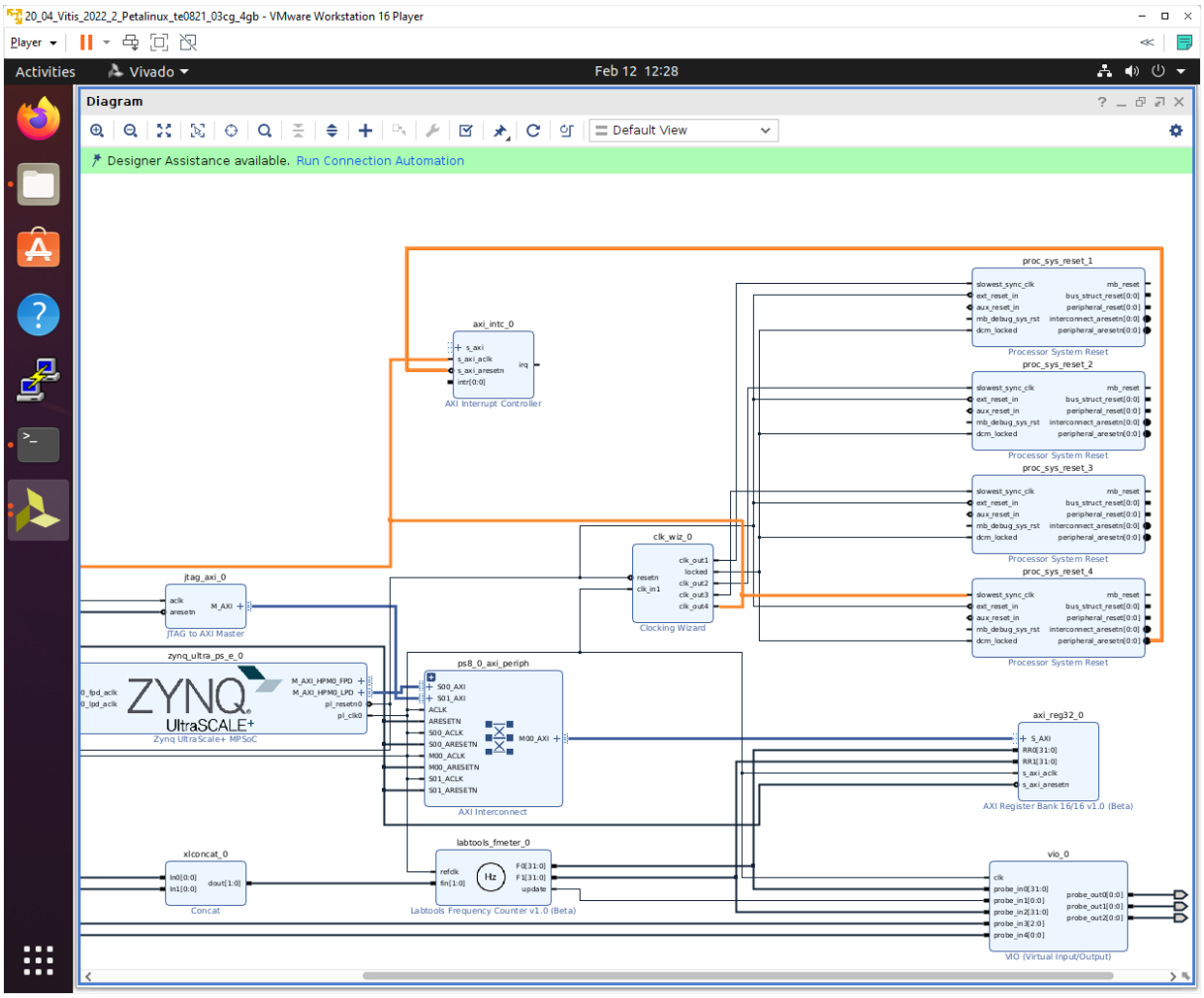
In “Processor Interrupt Type and Connection” section select the “Interrupt Output Connection” from “Bus” to “Single”.

Click on OK to accept these changes.

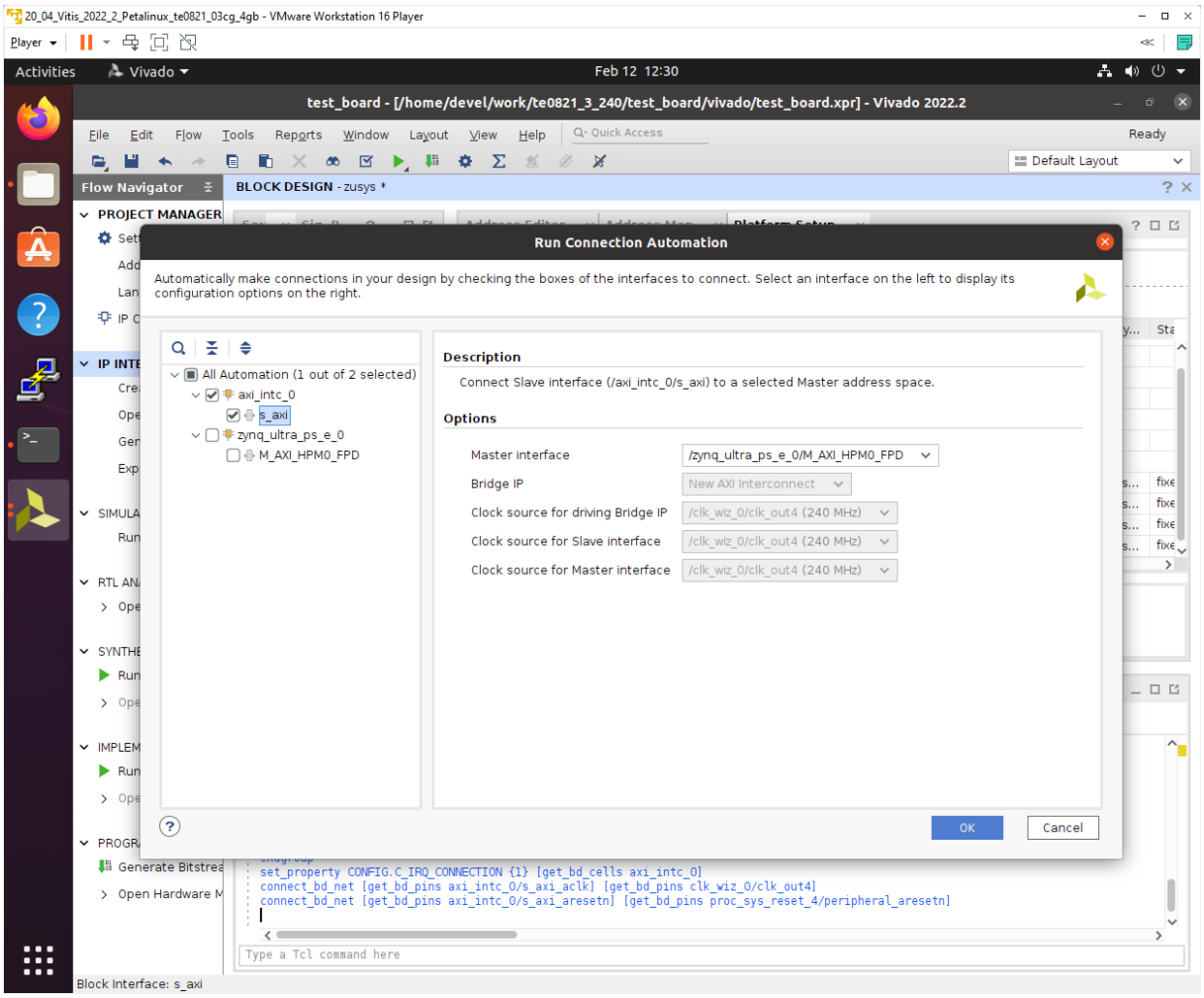


Connect interrupt controller clock input **s_axi_aclk** of **axi_intc_0** to output **dlk_out4** of **clk_wiz_0**. It is the default, 240 MHz clock of the extensible platform.

Connect interrupt controller input **s_axi_aresetn** of **axi_intc_0** to output **peripheral_aresetn[0:0]** of **proc_sys_reset_4** . It is the reset block for default, 240 MHz clock of the extensible platform.

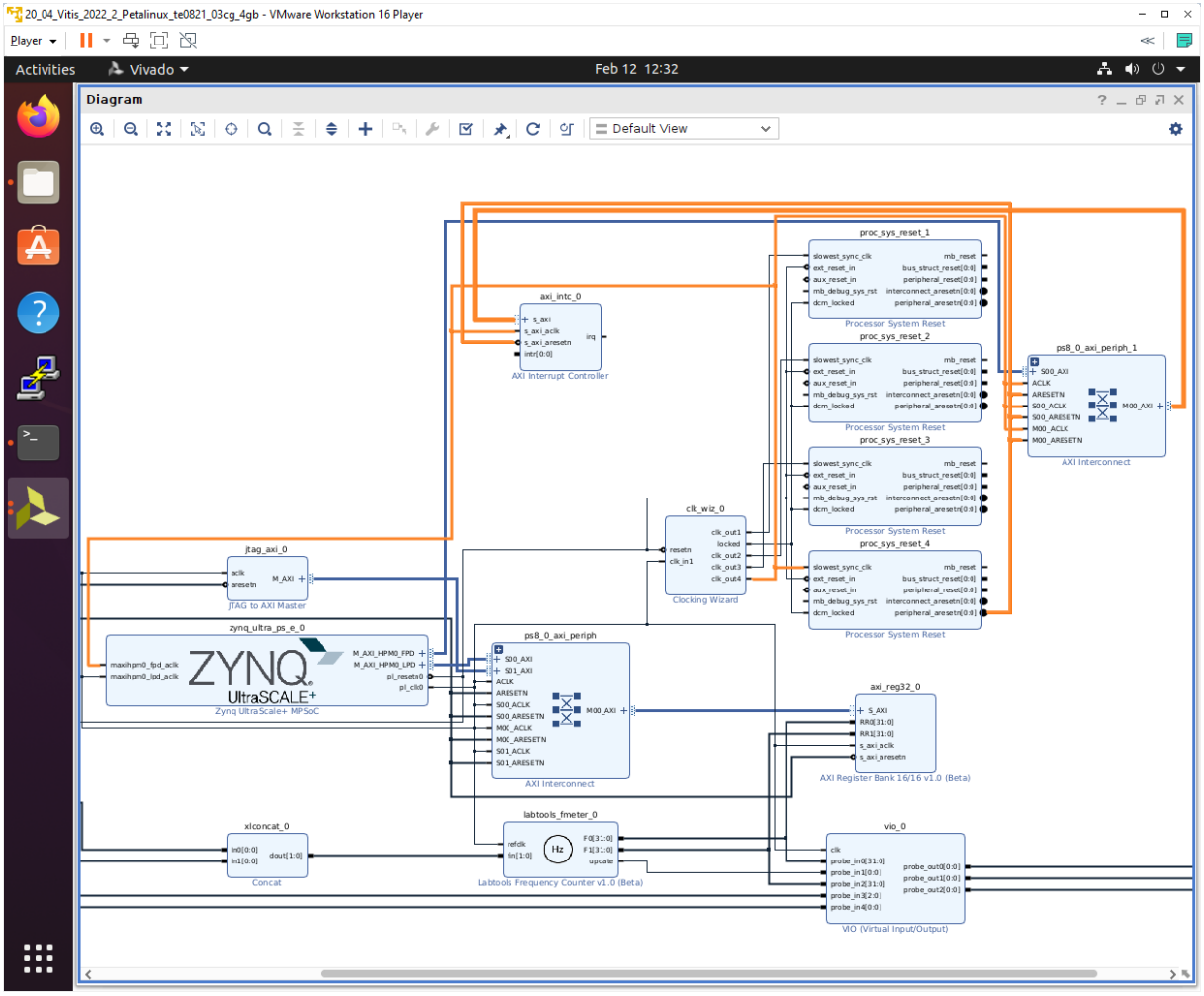


Use the **Run Connection Automation** wizard to connect the axi lite interface of interrupt controller **axi_intc_0** to master interface **M_AXI_HPM0_FPD** of **zynq_ultra_ps_e_0**.

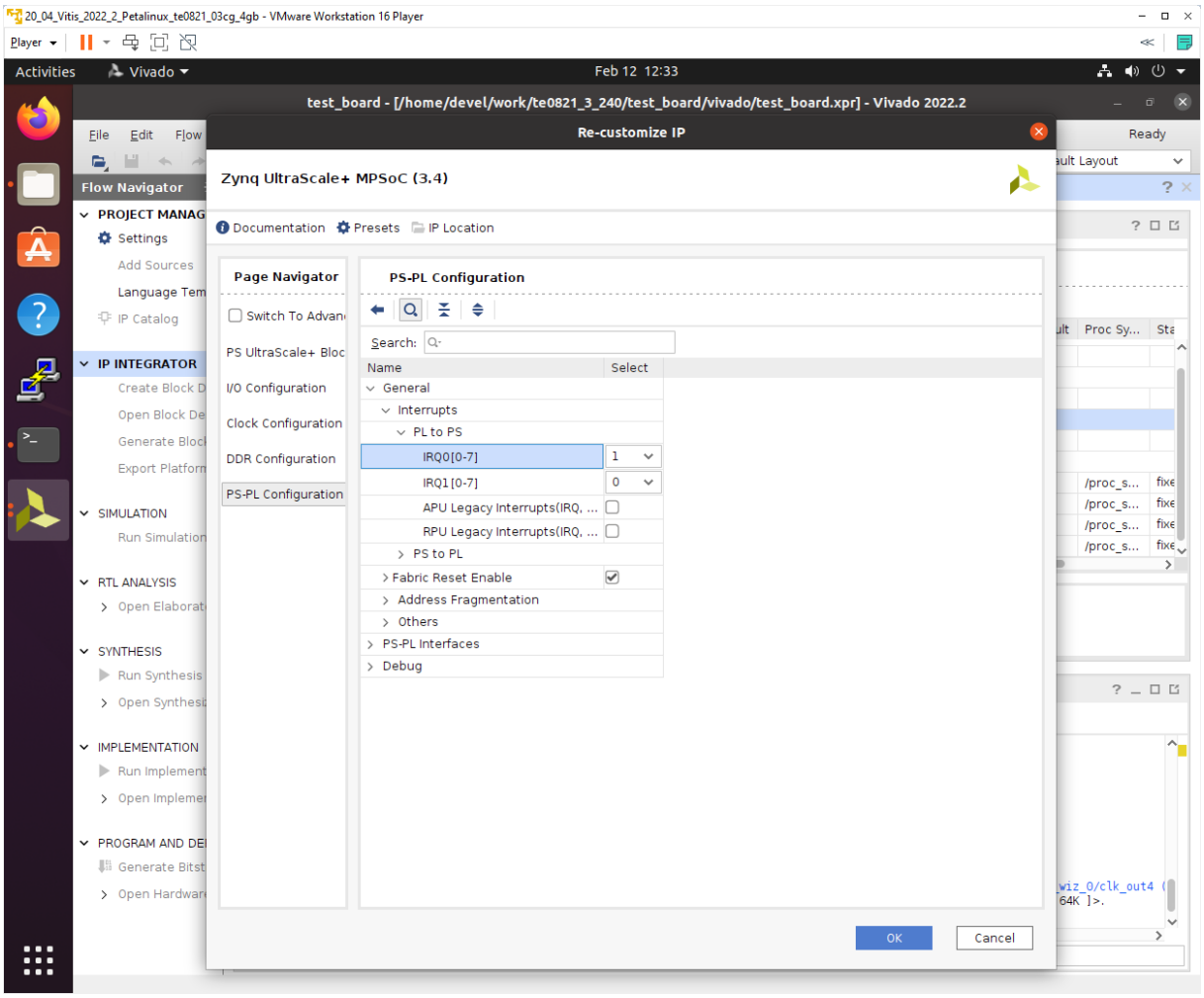


In Run Connection Automaton window, click **OK**.

New AXI interconnect **ps_8_axi_periph** is created. It connects master interface **M_AXI_HPM0_FPD** of **zynq_ultra_ps_e_0** with interrupt controller **axi_intc_0**.



Double-click on **zynq_ultra_ps_e_0** to re-customize it by enabling of an interrupt input **pl_ps_irq0[0:0]**. Click OK.



Modify the automatically generated reset network of AXI interconnect **ps_8_axi_periph** .

Disconnect input **S00_ARESETN** of **ps_8_axi_periph** from the network driven by output **peripherial_aresetn[0:0]** of **proc_sys_reset_4** block.

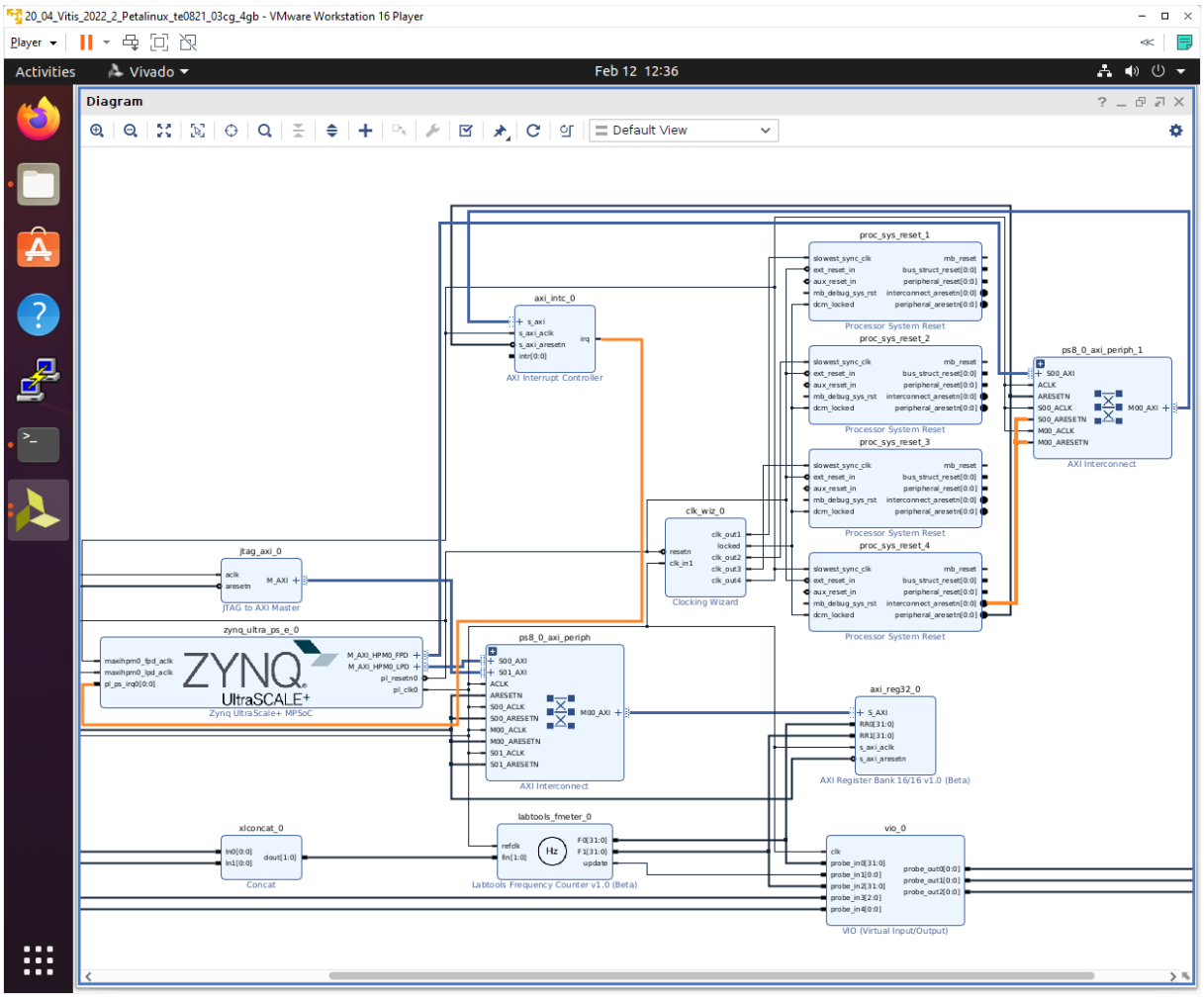
Connect input **S00_ARESETN** of **ps_8_axi_periph** block with output **interconnect_aresetn[0:0]** of **proc_sys_reset_4** block.

Disconnect input **M00_ARESETN** of **ps_8_axi_periph** block from the network driven by output **peripherial_aresetn[0:0]** of **proc_sys_reset_4** block.

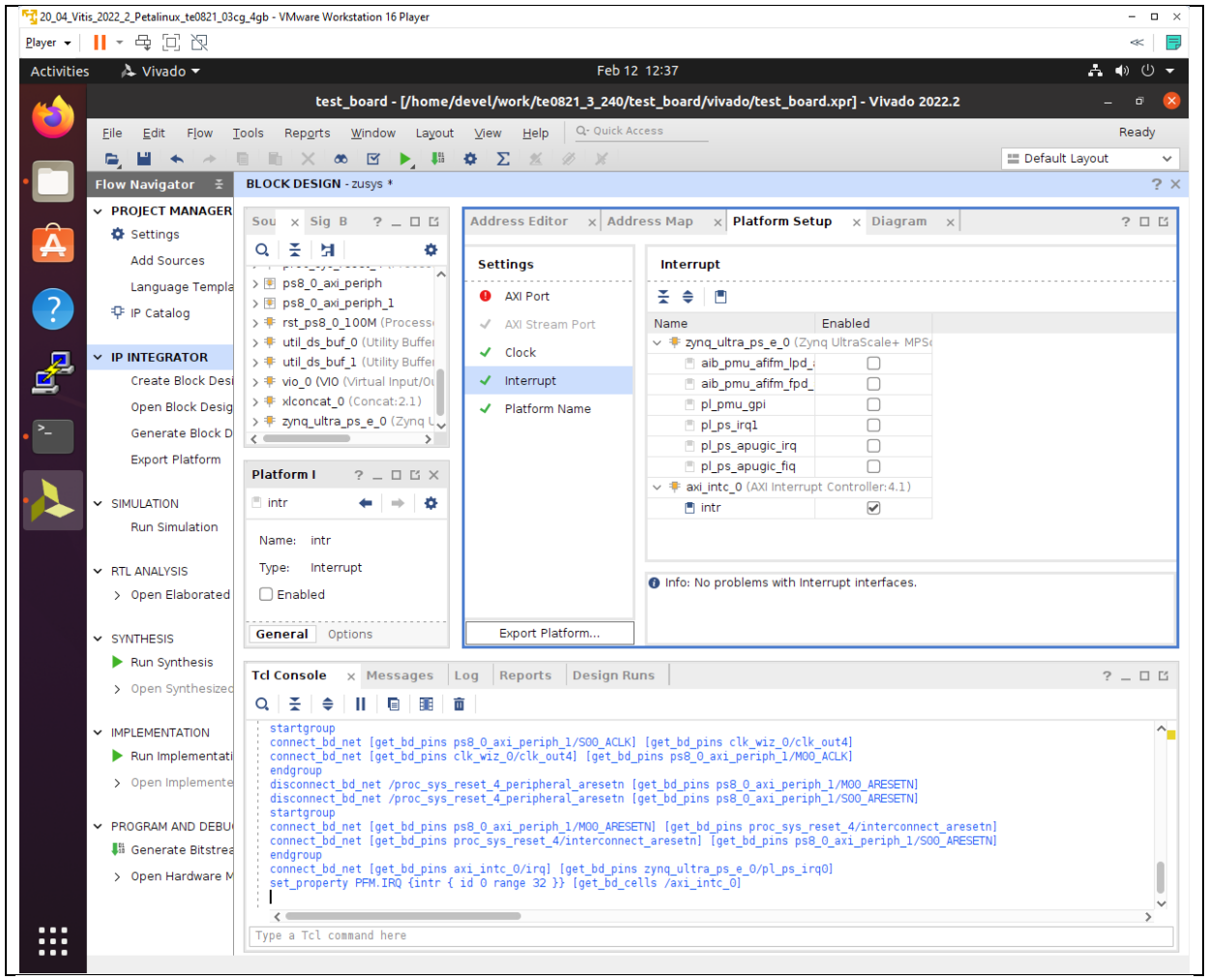
Connect input **M00_ARESETN** of **ps_8_axi_periph** to output **interconnect_aresetn[0:0]** of **proc_sys_reset_4** block.

This modification will make the reset structure of the AXI interconnect **ps_8_axi_periph** block identical to the future extensions of this interconnect generated by the Vitis extensible design flow.

Connect the interrupt **input pl_ps_irq0[0:0]** of **zynq_ultra_ps_e_0** block with output **irq** of **axi_intc_0** block.



In Platform Setup, select “Interrupt” and enable **intr** in the “Enabled” column.



Rename automatically generated name **ps8_0_axi_periph** of the interconnect to new name: **axi_interconnect_1** . This new name will be used in Platform Setup selection of AXI ports for the extensible platform.

In Platform Setup, select AXI Ports for **zynq_ultra_ps_e_0**:

Selec**M_AXI_HPM1_FPD** in column “Enabled”.

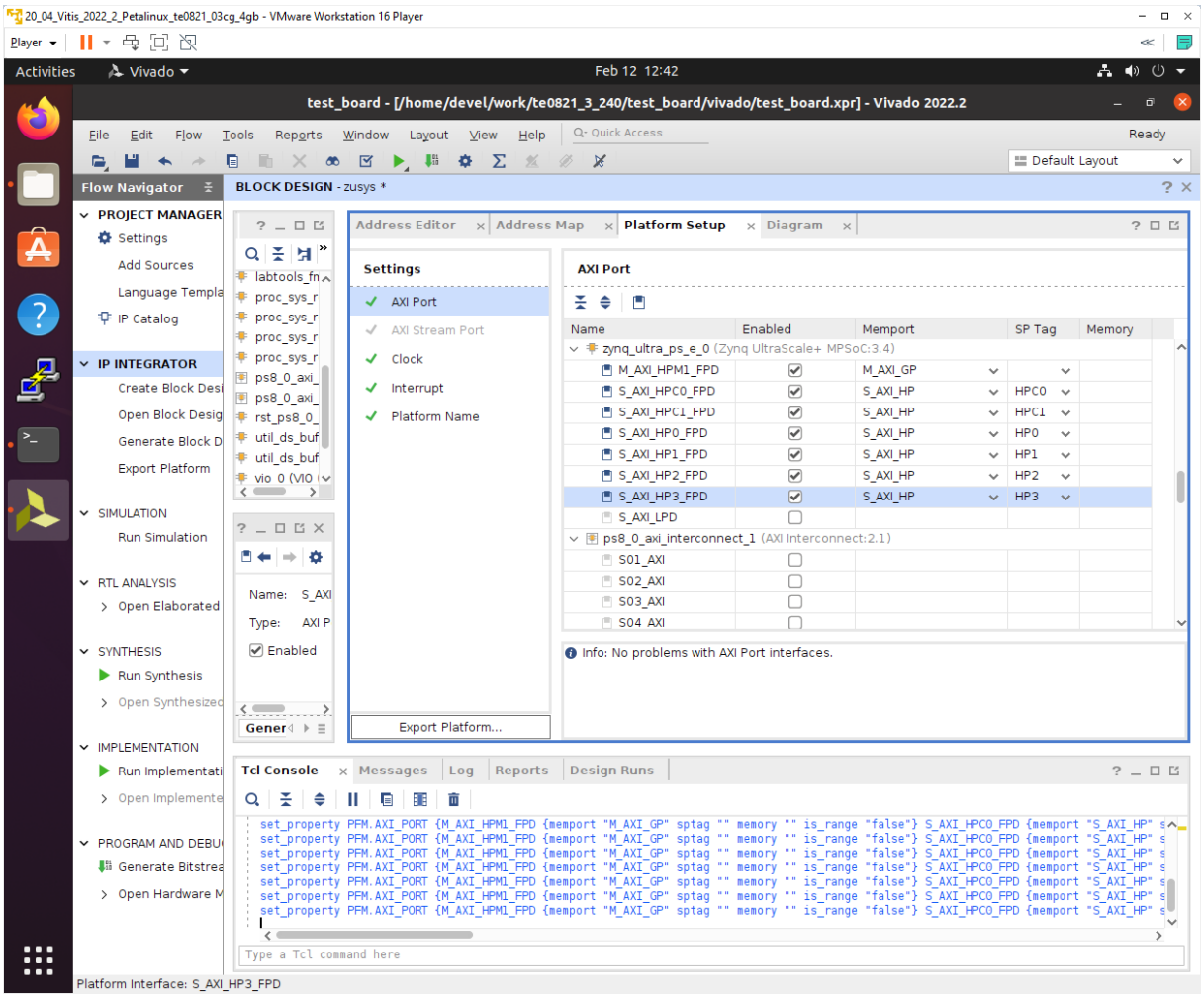
Select **S_AXI_HPC0_FPD** and **S_AXI_HPC1_FPD** in column “Enabled”.

For **S_AXI_HPC0_FPD**, change S_AXI_HPC to **S_AXI_HP** in column “Memport”.

For **S_AXI_HPC1_FPD**, change S_AXI_HPC to **S_AXI_HP** in column “Memport”.

Select **S_AXI_HP0_FPD, S_AXI_HP1_FPD, S_AXI_HP2_FPD, S_AXI_HP3_FPD** in column “Enabled”.

Type into the “sptag” column the names for these 6 interfaces so that they can be selected by v++ configuration during linking phase. **HPC0**, **HPC1**, **HP0**, **HP1**, **HP2**, **HP3**

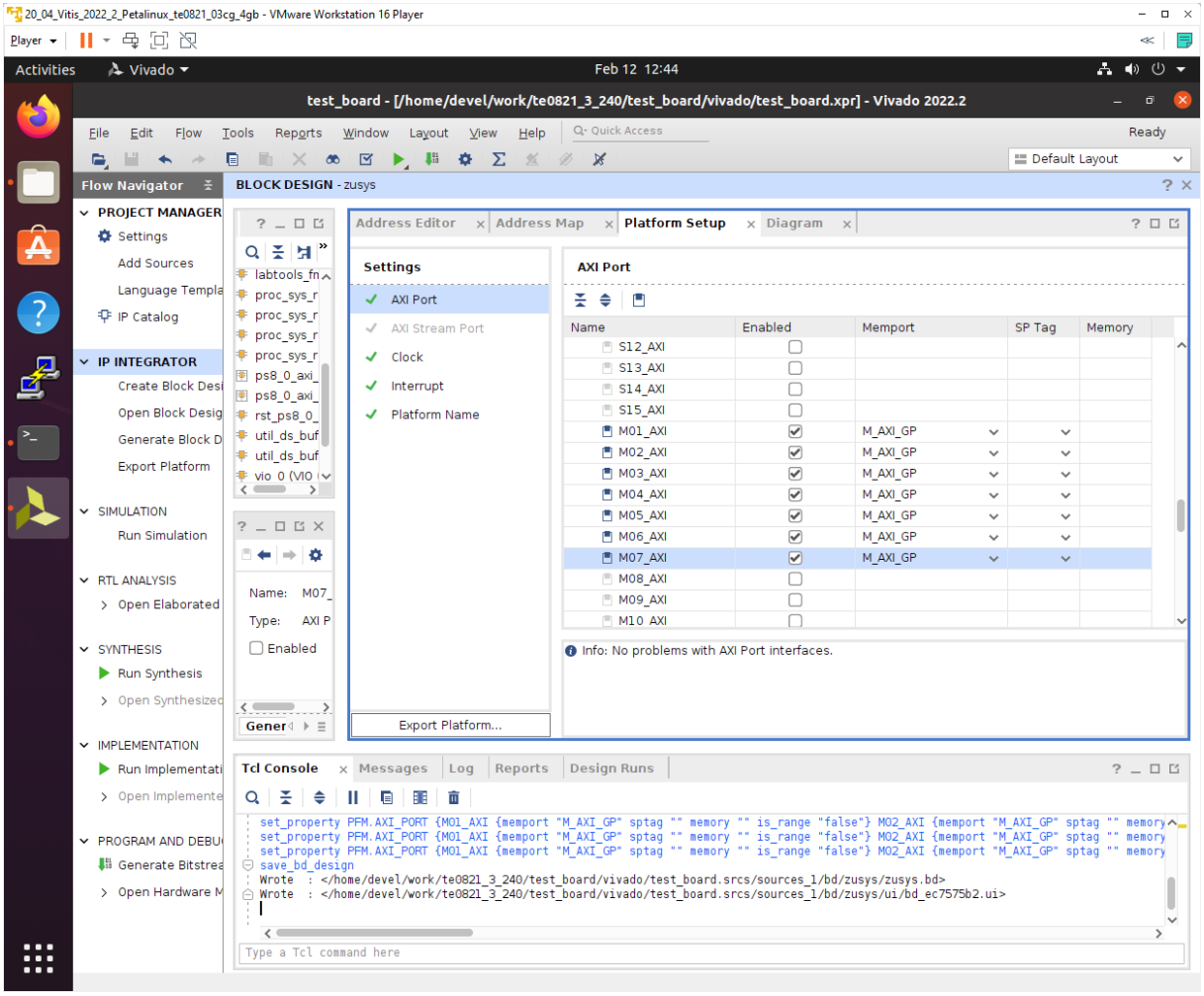


In “Platform Setup”, select AXI Ports for the recently renamed **axi_interconnect_1**:

Select **M01_AXI, M02_AXI, M03_AXI, M04_AXI, M05_AXI, M06_AXI** and **M07_AXI** in column “Enabled”.

Make sure, that you are selecting these AXI ports for the 240 MHz AXI interconnect **axi_interconnect_1**

Keep all AXI ports of the 100 MHz interconnect axi_interconnect_0 unselected. The AXI interconnect axi_interconnect_0 connects other logic and IPs which are part of the initial design.



The modifications of the default design for the extensible platform are completed, now.

In Vivado, save block design by clicking on icon "**Save Block Design**".

To continue the manual design path, go to Section 3.3 Validate design.

3.2 Fast Track for Creation of Extensible platform HW

HW modifications can be made by sourcing this script in Vivado with open diagram in IP Integrator.

Copy file from the accompanying support package

```
te0821_AI_3_0_eval_package\vivado\script_te0821.txt
```

to

```
~/work/TE0821_03_240/test_board/vivado/script_te0821.txt
```

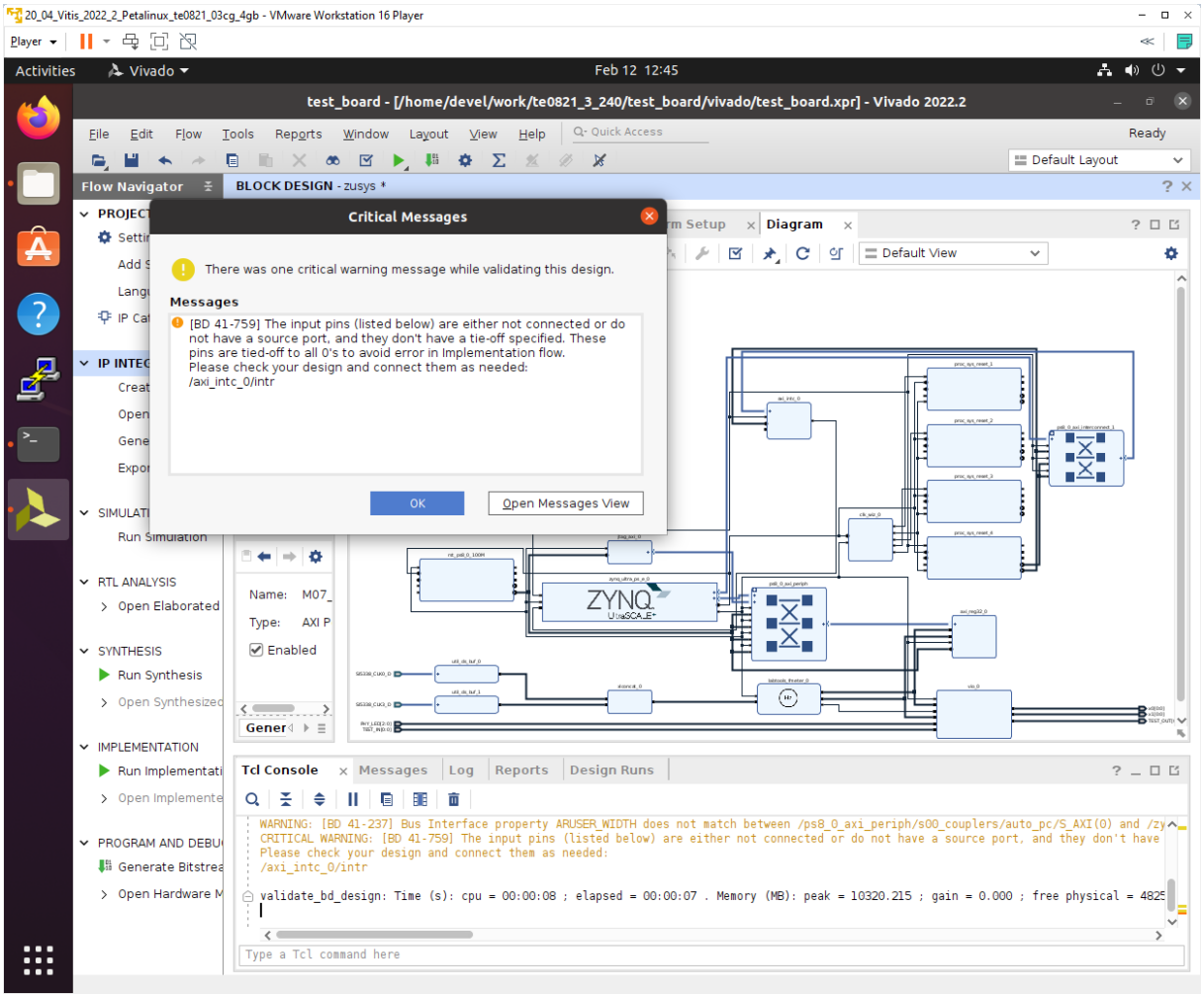
Execute in Vivado Tcl console this command:

```
source script_te0821.txt
```

3.3 Validate Design

Results of HW creation via Manual Track or Fast Track are identical.

Open diagram by clicking on zusys.bd if not already open.
In Diagram window, validate design by clicking on “Validate Design” icon.



Received Critical Messages window indicates that input intr[0:0] of axi_intc_0 is not connected. This is expected. The Vitis extensible design flow will autimatically connect this input to the common interrupt output from generated HW IPs.

Click OK.

Save design.

You can generate .pdf of the block diagram by clicking to any place in diagram window and selecting “Save as PDF File”. Use the offered default file name:

```
~/work/TE0821_03_240/test_board/vivado/zusys.pdf
```

3.4 Compile Created HW with Trenz Eletronic Script

In the running Vivado 2023.2.1 console, type following command and execute the corresponding .tcl scripts by <Enter>. It will take some time to compile this extensible HW design ti bitstream, and to export it to the corresponding .xsa package (with included bitstream).

```
TE::hw_build_design -export_prebuilt
```

The exported Vitis Extensible Hardware archive named `test_board_3cg_1e_4gb_dd.xsa` is created in the vivado folder.

```
~/work/TE0821_03_240/test_board/vivado/test_board_3cg_1e_4gb_dd.xsa
```

Keep the Vivado 2023.2.1 project open.

We will configure and compile Petalinux. This will create files nedded by Trenz Electronic script for generation of SW. This script will be called from the open Vivado. This script will also generate custom Trenz Electronic first-stage boot loader fsbl.elf . This file will be needed for creation of of the custom, extensible board support package for Vitis 2023.2.1. These steps are described in Chapter 4 of this application note.

4 Building Petalinux for Vitis AI 3.0 and AI3.5 SW Library

4.1 Vitis AI 3.0 models and Vitis AI 3.5 library

Petalinux 2023.2.1 distribution already contains recepies for the Vitis-AI 3.5 library.

Hovewer, the AMD DPUCZDX8G is works with the Vitis AI 3.0 models. There are no new models in the Vitis AI 3.5 for the AMD DPUCZDX8G used in AMD Zynq Ultrascale+ devices.

4.2 Building Petalinux for Extensible Design Flow

Keep the Vivado 2023.2.1 project open.

Open new terminal.

Change directory to the default Petalinux folder:

```
cd ~/work/TE0821_03_240/test_board/os/petalinux
```

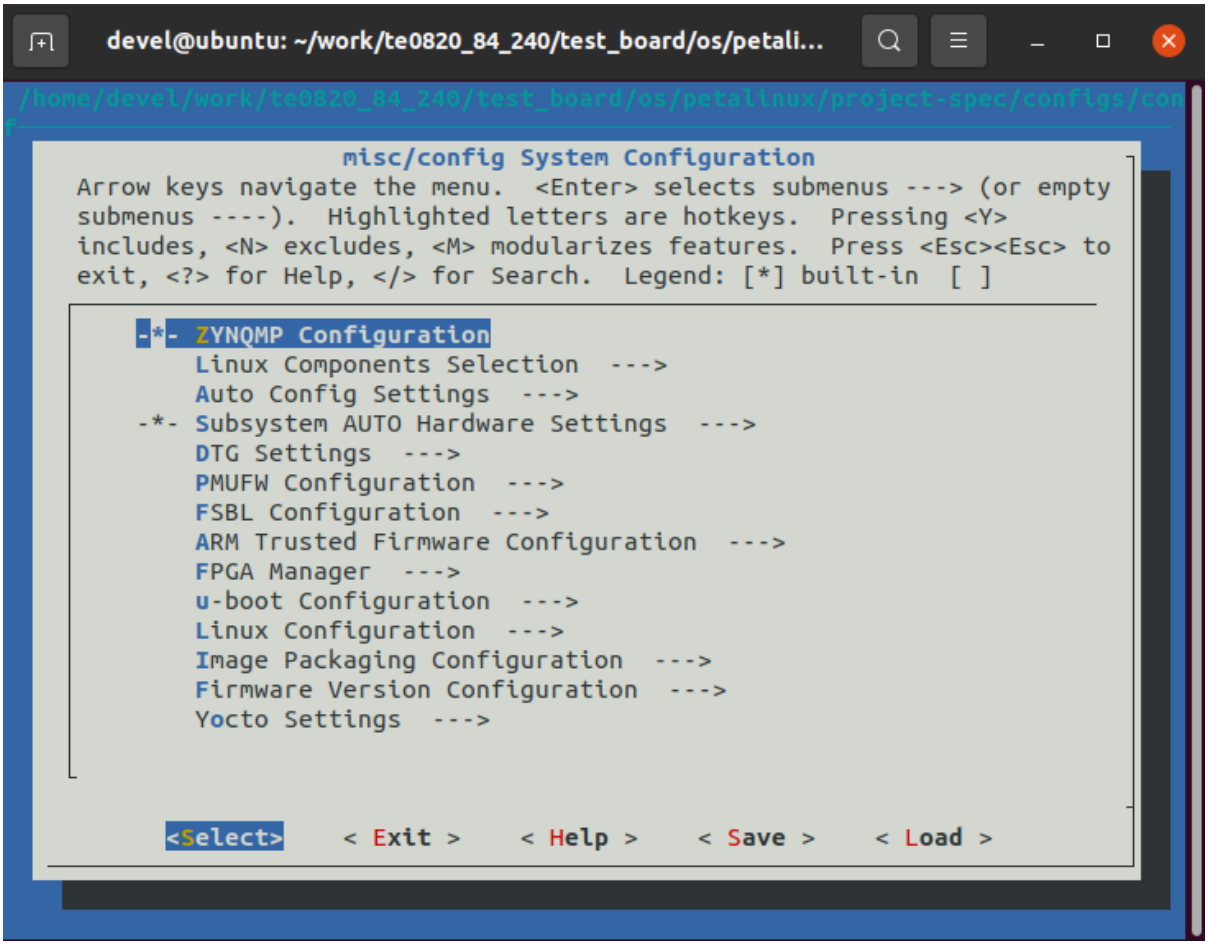
Source Vitis 2023.2.1 and Petalinux 2023.2 scripts to set environment for access to Vitis and PetaLinux tools.

```
$ source /tools/Xilinx/Vitis/2023.2/settings64.sh
$ source ~/petalinux/2023.2/settings.sh
```

Configure Petalinux with the `test_board_3cg_4gb.xsa` archive for the extensible design flow (created in section 3.4) by executing this PetaLinux command in the recently opened terminal:

```
$ petalinux-config --get-hw-description=
```

```
~/work/TE0821_03_240/test_board/vivado
```



Select **Exit->Yes** to close this window.

In text editor, modify the **user-rootfsconfig** file:

```
~/work/TE0821_86_240/test_board/os/petalinux/project-spec/meta-user/conf/user-rootfsconfig
```

In text editor, append these lines:

```
CONFIG_xrt
CONFIG_xrt-dev
CONFIG_zocl
CONFIG_opencl-clhpp-dev
CONFIG_opencl-headers-dev
CONFIG_packagegroup-petalinux-opencv
CONFIG_packagegroup-petalinux-opencv-dev
CONFIG_dnf
CONFIG_e2fsprogs-resize2fs
CONFIG_parted
CONFIG_resize-part
```

```
CONFIG_packagegroup-petalinux-vitisai
CONFIG_packagegroup-petalinux-self-hosted
CONFIG_cmake
CONFIG_packagegroup-petalinux-vitisai-dev
CONFIG_mesa-megadriver
CONFIG_packagegroup-petalinux-x11
CONFIG_packagegroup-petalinux-v4lutils
CONFIG_packagegroup-petalinux-matchbox
CONFIG_packagegroup-petalinux-vitis-acceleration
CONFIG_packagegroup-petalinux-vitis-acceleration-dev
CONFIG_packagegroup-petalinux-vitis-acceleration-essential
CONFIG_packagegroup-petalinux-vitis-acceleration-essential-dbg
CONFIG_packagegroup-petalinux-vitis-acceleration-essential-dev
CONFIG_packagegroup-core-ssh-dropbear
CONFIG_imagefeature-ssh-server-dropbear
CONFIG_imagefeature-ssh-server-openssh
CONFIG_openssh
CONFIG_openssh-sftp-server
CONFIG_openssh-sshd
CONFIG_openssh-scp
CONFIG_imagefeature-package-management
CONFIG_imagefeature-debug-tweaks
```

`xrt, xrt-dev` and `zocl` are required for Vitis acceleration flow.
`dnf` is for package management.
`parted, e2fsprogs-resize2fs` and `resize-part` can be used for ext4 partition resize.

Other included packages serve for natively building Vitis AI applications on target board and for running Vitis-AI demo applications with GUI.

The last three packages will enable use of the Vitis-AI 3.0 recepies for installation of the correspoding Vitis-AI 3.0 libraries into rootfs of PetaLinux.

Launch `rootfs` config:

```
$ petalinux-config -c rootfs
```

Enable user packages

Select `user packages`

```
user packages   --->

[*] cmake
[*] dnf
[*] e2fsprogs-resize2fs
[*] gpio-demo
[*] imagefeature-debug-tweaks
[*] imagefeature-package-management
[ ] imagefeature-ssh-server-dropbear
[*] imagefeature-ssh-server-openssh
[*] mesa-megadriver
```

```
[*] opencl-clhpp-dev
[*] opencl-headers-dev
[*] openssh
[*] openssh-scp
[*] openssh-sftp-server
[*] openssh-sshd
[ ] packagegroup-core-ssh-dropbear
[*] packagegroup-petalinux-matchbox
[*] packagegroup-petalinux-opencv
[*] packagegroup-petalinux-opencv-dev
[*] packagegroup-petalinux-self-hosted
[*] packagegroup-petalinux-v4lutils
[*] packagegroup-petalinux-vitis-acceleration
[*] packagegroup-petalinux-vitis-acceleration-dev
[*] packagegroup-petalinux-vitis-acceleration-essential
[ ] packagegroup-petalinux-vitis-acceleration-essential-dbg
[*] packagegroup-petalinux-vitis-acceleration-essential-dev
[*] packagegroup-petalinux-vitisai
[*] packagegroup-petalinux-vitisai-dev
[*] packagegroup-petalinux-x11
[*] parted
[*] peekpoke
[*] resize-part
[*] startup
[*] webfwu
[*] xrt
[*] xrt-dev
[*] zocl
```

Select all packages to have an asterisk `[*]`.

Still in the RootFS configuration window, go to root directory by select Exit once.

Enable OpenSSH and Disable Dropbear

Dropbear is the default SSH tool in Vitis Base Embedded Platform. If OpenSSH is used to replace Dropbear, the system could achieve faster data transmission speed over ssh. Created Vitis extensible platform applications may use remote display feature. Using of OpenSSH can improve the display experience.

Go to `Image Features`.
Disable `ssh-server-dropbear` and enable `ssh-server-openssh` and click Exit once.

Go to `Filesystem Packages->misc->packagegroup-core-ssh-dropbear` and disable `packagegroup-core-ssh-dropbear`.

Go to `Filesystem Packages` level by Exit twice.

Go to `console->network->openssh` and enable `openssh, openssh-sftp-server, openssh-sshd, openssh-scp.`

Go to root level by selection of Exit four times.

Enable Package Management

Package management feature can allow the board to install and upgrade software packages on the fly.

In rootfs config go to `Image Features` and enable

`package management` and `debug_tweaks`
options. Click OK**,** Exit twice and select Yes to save the changes.

4.3 Disable CPU IDLE in Kernel Config

CPU IDLE would cause processors get into IDLE state (WFI) when the processor is not in use. When JTAG is connected, the hardware server on host machine talks to the processor regularly. If it talks to a processor in IDLE status, the system will hang because of incomplete AXI transactions.

So, it is recommended to disable the CPU IDLE feature during project development phase.

It can be re-enabled after the design has completed to save power in final products.

Launch kernel config:

```
$ petalinux-config -c kernel
```

Ensure the following items are TURNED OFF by entering 'n' in the [ ] menu selection:

`CPU Power Management->CPU Idle->CPU idle PM support`

`CPU Power Management->CPU Frequency scaling->CPU Frequency scaling`

Exit and Yes **to** Save changes**.**

4.4 Add EXT4 rootfs Support

Let PetaLinux generate EXT4 rootfs. In terminal, execute:

```
$ petalinux-config
```

Go to `Image Packaging Configuration`.
Enter into `Root File System Type`

Select `Root File System Type  EXT4`

Change the `Device node` of SD device from the default value
`/dev/mmcblk0p2`

to new value required for the TE0821 module:
`/dev/mmcblk1p2`

Step up to

```
Image Packaging Configuration -->
```

modify `Root filesystem formats` from

```
cpio cpio.gz cpio.gz.u-boot ext4 tar.gz jffs2
```

to

```
ext4
```

Click on Exit and Yes to save changes.

4.5 Let Linux Use EXT4 rootfs During Boot

The setting of which rootfs to use during boot is controlled by bootargs. We would change bootargs settings to allow Linux to boot from EXT4 partition.

In terminal, execute:

```
$ petalinux-config
```

Change **DTG settings->Kernel Bootargs->generate boot args automatically** to NO.

Update **User Set Kernel Bootargs** to:

```
earlycon console=ttyPS0,115200 clk_ignore_unused root=/dev/mmcblk1p2 rw
rootwait cma=512M
```

Click **OK**, **Exit** three times and Save.

4.6 Build PetaLinux Image

In terminal, build the PetaLinux project by executing:

```
$ petalinux-build
```

The PetaLinux image files will be generated in the directory:

```
~/work/TE0821_03_240/test_board/os/petalinux/images/linux
```

Generation of PetaLinux takes some time and requires Ethernet connection and sufficient free disk space of the Ubuntu 20.04 PC**.**

4.7 Build Petalinux SDK

The SDK will be used by the Vitis 2023.2 tool to cross compile applications for newly created platfom.

In terminal, execute:

```
$ petalinux-build --sdk
```

The generated sysroot package **sdk.sh** will be located in directory

```
~/work/TE0821_03_240/test_board/os/petalinux/images/linux
```

Generation of SDK package takes some time and requires sufficient free disk space**.**
Time needed for these two steps depends also on number of allocated processor cores.

4.8 Created module-specific files

The PetaLinux compilation proces has created multiple module specific files. Some of them have to be copied into specific, module dependent directories:

Create directories:

```
~/work/TE0821_03_240/test_board/prebuilt/os/4GB

~/work/TE0821_03_240/test_board/prebuilt/os/2GB
```

Copy these five files:

Files	From	To
bl31.elf boot.scr image.ub system.dtb u-boot-dtb.elf	~/work/TE0821_03_240/test_board/os/petalinux/images/linux	~/work/TE0821_03_240/test_board_pfm/prebuilt/os/4GB

Rename the copied file `u-boot-dtb.elf` to `u-boot.elf` .

The directory

```
~/work/TE0821_03_240/test_board_pfm/prebuilt/os/4GB
```

contains these five files:

```
bl31.elf
boot.scr
image.ub
system.dtb
u-boot.elf
```

4.9 Compile Custom SW with Trenz Electronic Script

In Vivado 2023.2.1 Tcl Console, type the following command and start to execute the corresponding tcl scripts by < Enter>. It will take some time to compile.

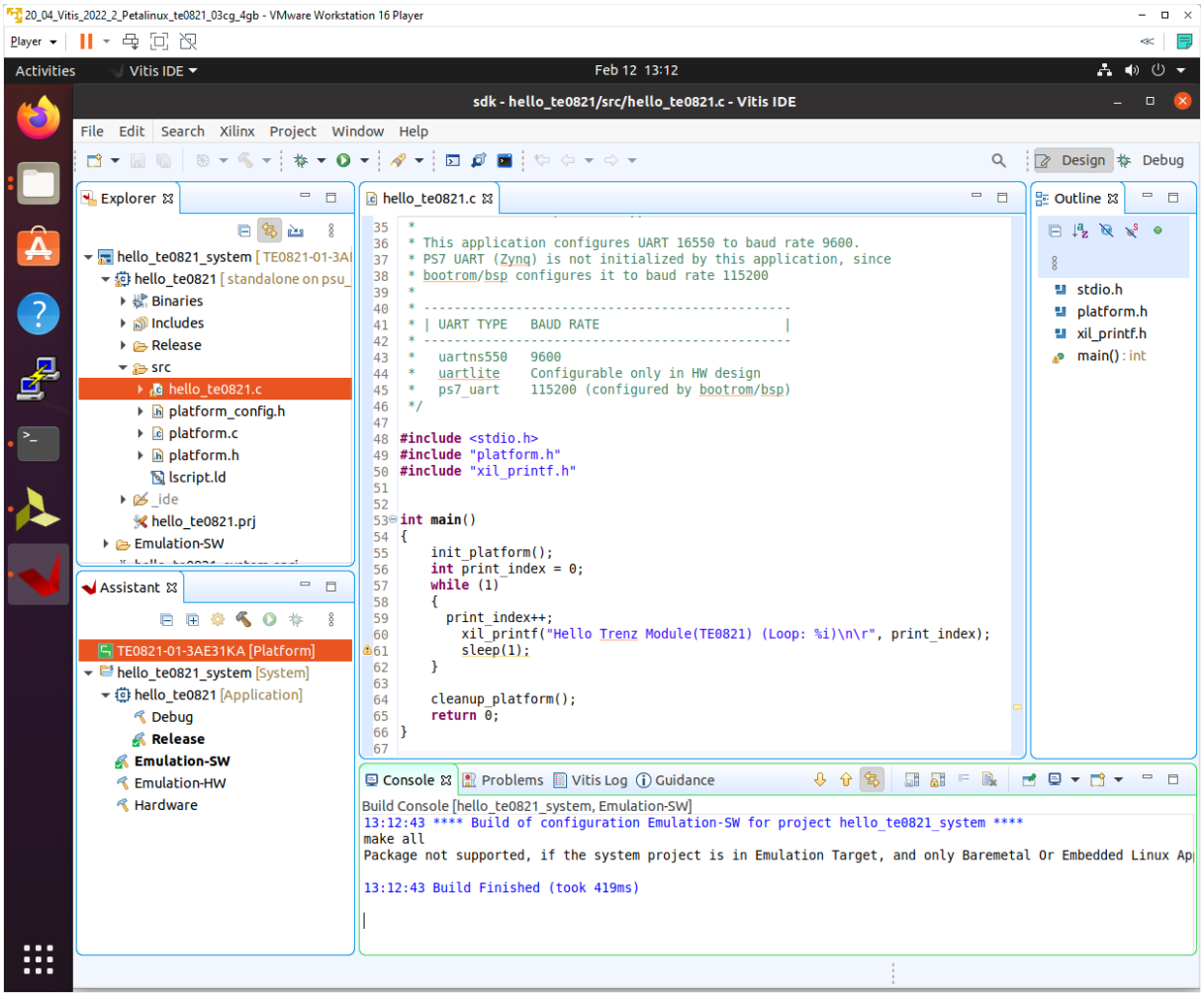
```
TE::sw_run_vitis -all
```

After the script controlling the SW compilation is finished, the Vitis 2023.2.1 --classic SDK GUI is opened.

Close the Vitis “Welcome” page.

Compile the two included SW projects.

A stand-alone custom Vitis platform `TE0821-01-3AE31KA` has been created and compiled.



The stand-alone `TE0821-01-3AE31KA` Vitis 2023.2 platform includes also the Trenz Electronic custom first-stage boot-loader project in folder `zynqmp_fsbl`. It includes SW extension specific for the Trenz Electronic initialization of the system. This includes Trenz Electronic specific inicialisations of external generators of clock signal for the module.

This custom `zynqmp_fsbl` project has been compiled and its executable file `fsbl.elf` was generated in:

```
~/work/TE0821_03_240/test_board/prebuilt/software/3cg_1e_4gb_dd/fsbl.elf
```

This customised first-stage boot-loader will be needed for the Trenz Electronic Vitis 2023.2.1 custom extensible SW platform.

Exit the opened Vitis 2023.2.1 SDK project.

In Vivado top menu select File->Close Project to close project. Click OK.

In Vivado top menu select File->Exit to close Vivado 2023.2.1. Click OK**.**

4.10 Copy Created Custom First-stage Boot-loader

Up to now, test_board directory has been used for all development.

```
~/work/TE0821_03_240/test_board
```

Create new folders:

```
~/work/TE0821_03_240/test_board_pfm/pfm/boot
~/work/TE0821_03_240/test_board_pfm/pfm/sd_dir
```

Copy the recently created custom first stage boot loader executable file from

```
~/work/TE0821_03_240/test_board/prebuilt/software/3cg_1e_4gb_dd/fsbl.el
f
```

to

```
~/work/TE0821_03_240/test_board_pfm/pfm/boot/fsbl.elf
```

4.11 Copy Files for Extensible Platform

Copy these four files:

Files	From	To
bl31.elf pmufw.elf system.dtb u-boot-dtb.elf	~/work/TE0821_03_240/test_board/os/petalinux/images/linux	~/work/TE0821_03_240/test_board_pfm/pfm/boot

Rename the copied file `u-boot-dtb.elf` to `u-boot.elf`

The directory

```
~/work/TE0821_03_240/test_board_pfm/pfm/boot
```

contains these five files:

```
bl31.elf
fsbl.elf
pmufw.elf
system.dtb
u-boot.elf
```

Copy files:

Files	From	To
boot.scr system.dtb	~/work/TE0821_03_240/test_board/os/petalinux/images/linux	~/work/TE0821_03_240/test_board_pfm/pfm/sd_dir

Copy file:

File	From	To

File	From	To
init.sh	~/work/TE0821_03_240/test_board/misc/sd	~/work/TE0821_03_240/test_board_pfm/pfm/sd_dir

init.sh is an place-holder for user defined bash code to be executed after the boot:

```
#!/bin/sh
normal="\e[39m"
lightred="\e[91m"
lightgreen="\e[92m"
green="\e[32m"
yellow="\e[33m"
cyan="\e[36m"
red="\e[31m"
magenta="\e[95m"

echo -ne $lightred
echo Load SD Init Script
echo -ne $cyan
echo User bash Code can be inserted here and put init.sh on SD
echo -ne $normal
```

4.12 Create Extensible Platform zip File

Create new directory tree:

```
~/work/TE0821_03_240_move/test_board/os/petalinux/images
~/work/TE0821_03_240_move/test_board/Vivado
~/work/TE0821_03_240_move/test_board_pfm/pfm/boot
~/work/TE0821_03_240_move/test_board_pfm/pfm/sd_dir
```

Copy all files from the directory:

Files	Source	Destination
all	~/work/TE0821_03_240/test_board/os/petalinux/images	~/work/TE0821_03_240_move/test_board/os/petalinux/images
all	~/work/TE0821_03_240/test_board_pfm/pfm/boot	~/work/TE0821_03_240_move/test_board_pfm/pfm/boot
all	~/work/TE0821_03_240/test_board_pfm/pfm/sd_dir	~/work/TE0821_03_240_move/test_board_pfm/pfm/sd_dir
test_board_3cg_1e_4gb.xsa	~/work/TE0821_03_240/test_board/Vivado/test_board_3cg_1e_4gb.xsa	~/work/TE0821_03_240_move/test_board/Vivado/test_board_3cg_1e_4gb.xsa

Zip the directory

```
~/work/TE0821_03_240_move
```

into ZIP archive:

```
~/work/TE0821_03_240_move.zip
```

The archive `TE0821_03_240_move.zip` can be used to create extensible platform on the same or on an another PC with installed Ubuntu 20.04 and Vitis tools, with or without installed Petalinux. The archive includes all needed components, including the Xilinx xrt library and the script sdk.sh serving for generation of the sysroot .

The archive has size approximately 3.6 GB and it is valid for the initially selected module number (84). This is the TE0821 HW module with xczu3cg-sfvc784-1-e device with 4 GB memory. The extensible Vitis platform will have the default clock 240 MHz.

Move the `TE0821_03_240_move.zip` file to an PC disk drive.

Delete:

```
~/work/TE0821_03_240_move
~/work/TE0821_03_240_move.zip
```

Clean the Ubuntu Trash.

4.13 Generation of SYSROOT

This part of development can be direct continuation of the previous Petalinux configuration and compilation steps.
Alternatively, it is also possible to implement all next steps on an Ubuntu 20.04 without installed PetaLinux Only the Ubuntu 20.04 and Vitis/Vivado installation is needed.
All required files created in the PetaLinux for the specific module (24) are present in the archive: `TE0821_03_240_move.zip`
In this case, unzip the archive to the directory:

```
~/work/TE0821_03_240_move
```

and copy all content of directories to

```
~/work/TE0821_03_240
```

Delete the **TE0821_03_240_move.zip** file and the **~/work/TE0821_03_240_move** directory to save filesystem space.

In Ubuntu terminal, change the working directory to:

```
~/work/TE0821_03_240/test_board/os/petalinux/images/linux
```

In Ubuntu terminal, execute script enabling access to Vitis 2023.2 tools.
Execution of script serving for setting up PetaLinux environment is not necessary:

In Ubuntu terminal, execute script

```
$ source /tools/Xilinx/Vitis/2023.2/settings64.sh
```

Execute script:

```
./sdk.sh
```

In Ubuntu terminal, execute script

```
$ ~/work/TE0821_03_240/test_board_pfm
```

```
PetaLinux SDK installer version 2023.2
```

```
=====================================

Enter target directory for SDK (default: /opt/petalinux/2023.2):
```

Enter:

```
/home/devel/work/TE0821_03_240/test_board_pfm
```

Reply: Y

SYSROOT directories and files for PC and for Zynq Ultrascale+ will be created in:

```
~/work/TE0821_03_240/test_board_pfm/sysroots/x86_64-petalinux-linux
~/work/TE0821_03_240/test_board_pfm/sysroots/cortexa72-cortexa53-
xilinx-linux
```

Once created, do not move these sysroot directories (due to some internally created paths).

4.14 Generation of Extensible Platform for Vitis

In Ubuntu terminal, change the working directory to:

```
~/work/TE0821_03_240/test_board_pfm
```

Start the Vitis tool by executing

```
$ vitis &
```

In Vitis “Launcher”, set the workspace for the extensible platform compilation:

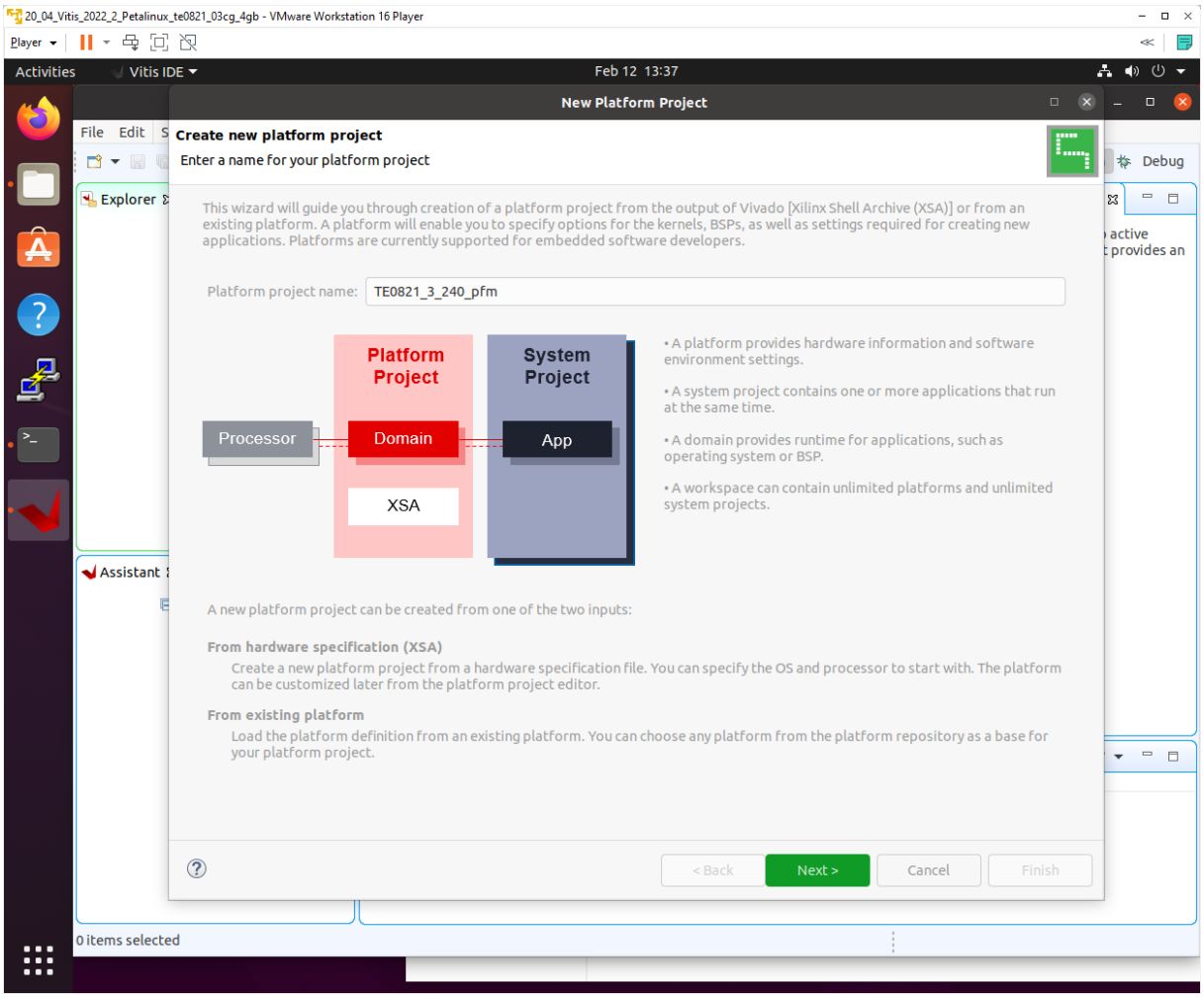
```
~/work/TE0821_03_240/test_board_pfm
```

Click on “Launch” to launch Vitis

Close Welcome page.

In Vitis, select in the main menu: `File -> New -> Platform Project`

Type name of the extensible platform: `TE0821_03_240_pfm`. Click Next.



Choose for hardware specification for the platform file:

```
~/work/TE0821_03_240/test_board/vivado/test_board_3cg_1e_4gb_dd.xsa
```

In “Software specification” select: `linux`

In “Boot Components” unselect: `Generate boot components`
(these components have been already generated by Vivado and PetaLinux design flow)

New window `TE0821_03_240_pfm` is opened.

Click on `linux on psu_cortex53` to open window Domain: `linux_domain`

In “Description” write: `xrt`

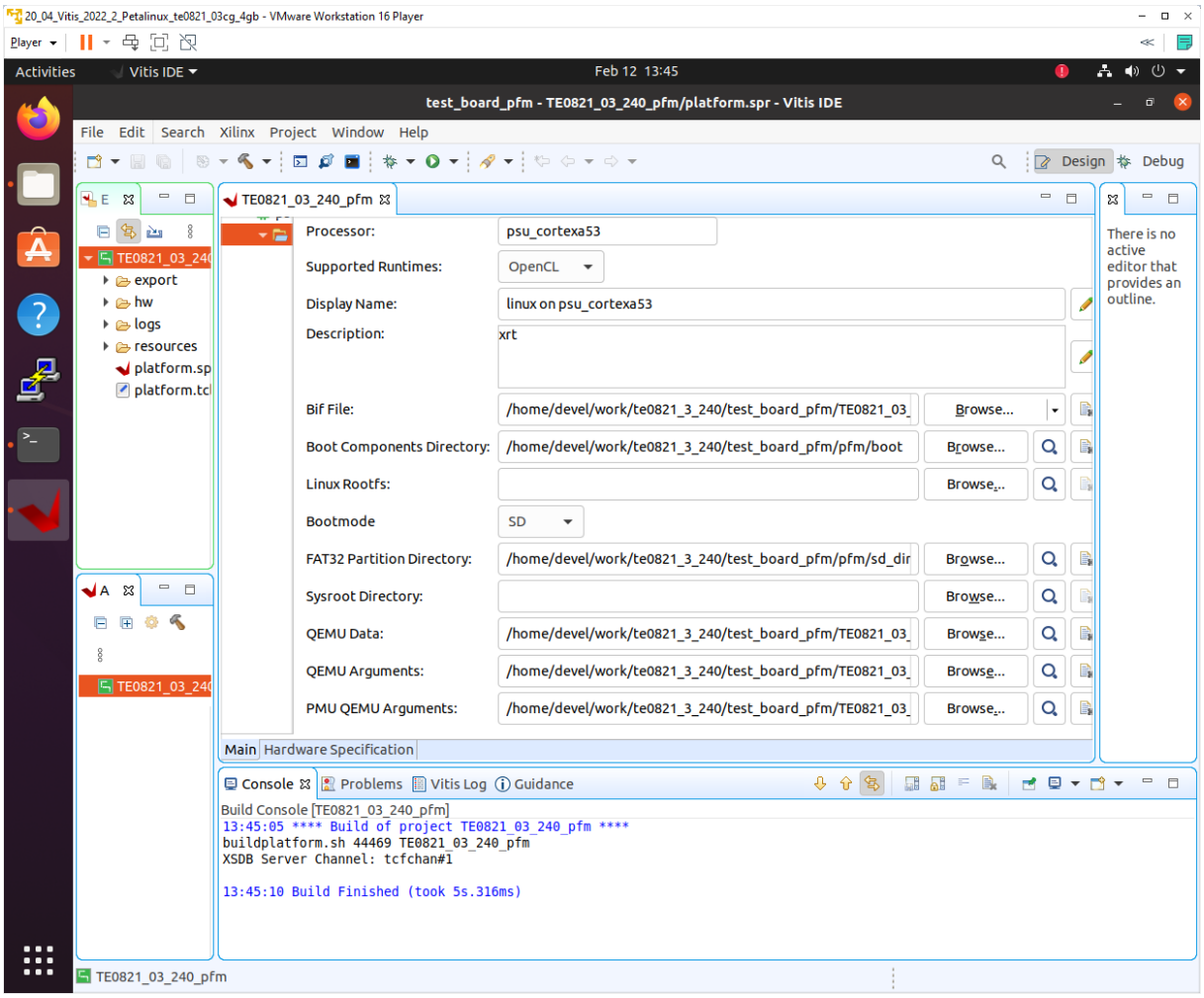
In “Bif File” find and select the pre-defied option: `Generate Bif`

In “Boot Components Directory” select:

```
~/work/TE0821_03_240/test_board_pfm/pfm/boot
```

In “FAT32 Partition Directory” select:

```
~/work/TE0821_03_240/test_board_pfm/pfm/sd_dir
```



In Vitis IDE “Explorer” section, click on `TE0821_03_240_pfm` to highlight it.

Right-click on the highlighted `TE0821_03_240_pfm` and select `build project` in the open submenu. Platform is compiled in few seconds.
Close the Vitis tool by selection: `File -> Exit`.

Vits extensible platform `TE0821_03_240_pfm` has been created in the directory:

```
~/work/TE0821_03_240/test_board_pfm/TE0821_03_240_pfm/export/TE0821_03_240_pfm
```

5 Platform Usage

5.1 Read Platform Info

Open terminal and set With Vitis 2023.2.1 environment. Use `platforminfo` tool to report information about the created custom extensible platform:

```
devel@ubuntu:~/work/te0821_03_240$ source
/tools/Xilinx/Vitis/2023.2/settings64.sh
```

```
devel@ubuntu:~/work/te0821_03_240$ platforminfo
/home/devel/work/te0821_03_240/test_board_pfm/te0821_03_240_pfm/export/
te0821_03_240_pfm/te0821_03_240_pfm.xpfm

==========================

Basic Platform Information

==========================

Platform:           te0821_03_240_pfm

File:
/home/devel/work/te0821_03_240/test_board_pfm/te0821_03_240_pfm/export/
te0821_03_240_pfm/te0821_03_240_pfm.xpfm

Description:

te0821_03_240_pfm

====================================

Hardware Platform (Shell) Information

====================================

Vendor:                             vendor

Board:                              zusys

Name:                               zusys

Version:                            1.0

Generated Version:                  2023.2.1

Hardware:                           1

Software Emulation:                 1

Hardware Emulation:                 0

Hardware Emulation Platform:        0

FPGA Family:                        zynquplus

FPGA Device:                        xczu3cg

Board Vendor:                       trenz.biz

Board Name:                         trenz.biz:te0821_3cg_1e:3.0

Board Part:                         xczu3cg-sfvc784-1-e

=================

Clock Information

=================

  Default Clock Index: 4

  Clock Index:         1
```

```
    Frequency:          100.000000
  Clock Index:          2
    Frequency:          200.000000
  Clock Index:          3
    Frequency:          400.000000
  Clock Index:          4
    Frequency:          240.000000
==================
Memory Information
==================
  Bus SP Tag: HP0
  Bus SP Tag: HP1
  Bus SP Tag: HP2
  Bus SP Tag: HP3
  Bus SP Tag: HPC0
  Bus SP Tag: HPC1
============================
Software Platform Information
============================
Number of Runtimes:             1
Default System Configuration:   te0821_03_240_pfm
System Configurations:
  System Config Name:                       te0821_03_240_pfm
  System Config Description:                te0821_03_240_pfm
  System Config Default Processor Group:    linux_domain
  System Config Default Boot Image:         standard
  System Config Is QEMU Supported:          1
  System Config Processor Groups:
    Processor Group Name:       linux on psu_cortexa53
    Processor Group CPU Type:   cortex-a53
    Processor Group OS Name:    linux
  System Config Boot Images:
    Boot Image Name:            standard
```

```
    Boot Image Type:
    Boot Image BIF:             te0821_03_240_pfm/boot/linux.bif
    Boot Image Data:            te0821_03_240_pfm/linux_domain/image
    Boot Image Boot Mode:       sd
    Boot Image RootFileSystem:
    Boot Image Mount Path:      /mnt
    Boot Image Read Me:         te0821_03_240_pfm/boot/generic.readme
    Boot Image QEMU Args:
te0821_03_240_pfm/qemu/pmu_args.txt:te0821_03_240_pfm/qemu/qemu_args.tx
t
    Boot Image QEMU Boot:
    Boot Image QEMU Dev Tree:
Supported Runtimes:
  Runtime: OpenCL
```

Clock information and memory information are set as expected.

5.2 Create and Compile Vitis 2023.2 Vector Addition Example

Create new directory `test_board_test_vadd` to test Vitis 2023.2 extendable flow example "vector addition"

```
~/work/TE0821_03_240/test_board_test_vadd
```

Current directory structure:

```
~/work/TE0821_03_240/test_board
~/work/TE0821_03_240/test_board_pfm
~/work/TE0821_03_240/test_board_test_vadd
```

Change working directory:

```
$cd ~/work/TE0821_03_240/test_board_test_vadd
```

In Ubuntu terminal, start Vitis2023.2 by:

```
$ vitis --classic --workspace . &
```

In Vitis IDE Launcher, select your working directory

```
~/work/TE0821_03_240/test_board_test_vadd
```

Click on `Launch` to launch Vitis.

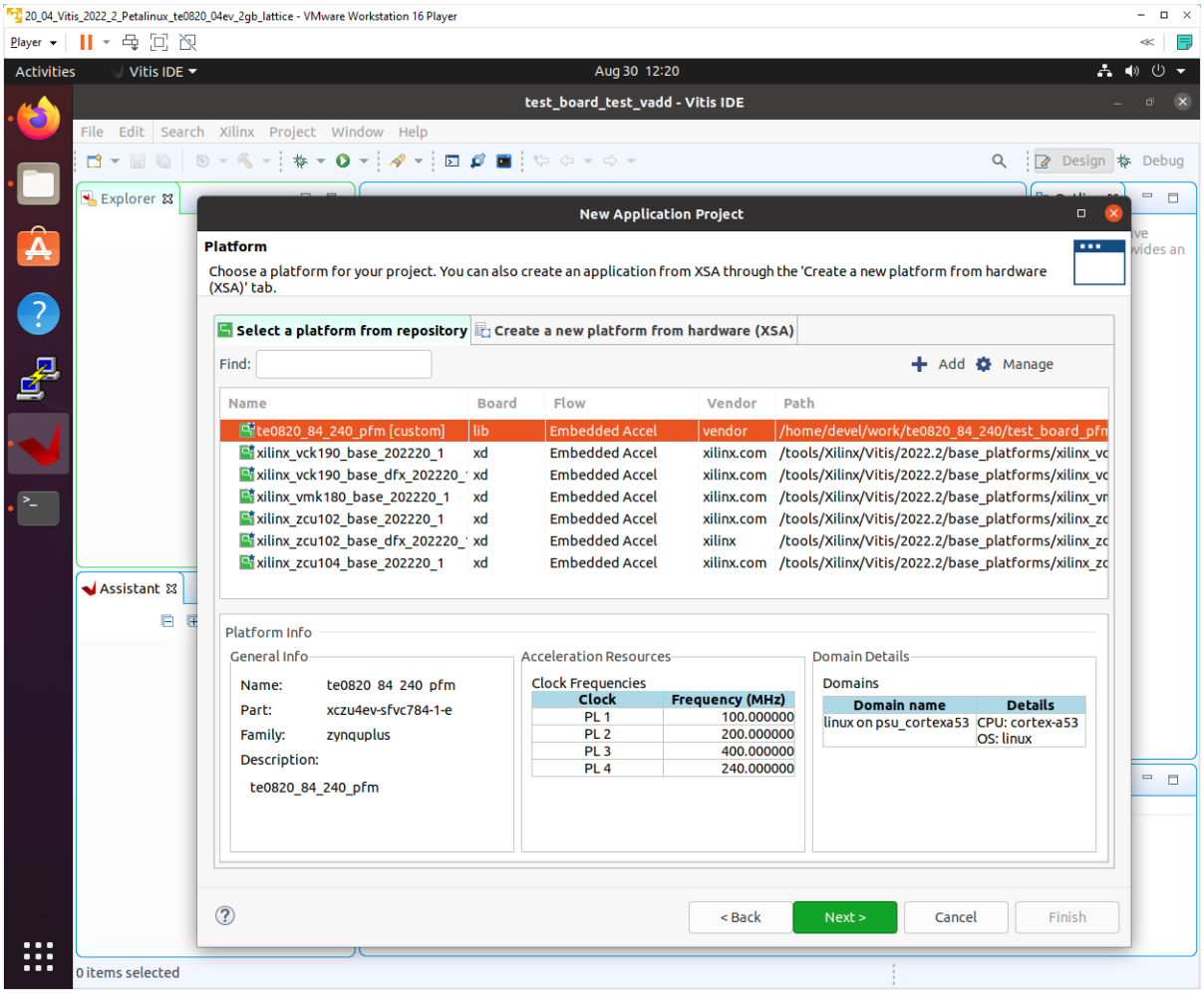
`Select File -> New -> Application project.` Click Next.

Skip welcome page if shown.

Click on [+ Add] icon and select the custom extensible platform TE0821_03_240_pfm[custom] in the directory:

```
~/work/TE0821_03_240/test_board_pfm/TE0821_03_240_pfm/export/TE0821_03_240_pfm
```

We can see available PL clocks and frequencies. PL4 with 240 MHz clock has been set as default in the platform creation process.



Click Next.
In Application Project Details window type into Application project name: test_vadd
Click Next.
In Domain window type (or select by browse):

Sysroot path:

```
~/work/TE0821_03_240/test_board_pfm/sysroots/cortexa72-cortexa53-xilinx-linux
```

Root FS:

```
~/work/TE0821_03_240/test_board/os/petalinux/images/linux/rootfs.ext4
```

Kernel Image:

```
~/work/TE0821_03_240/test_board/os/petalinux/images/linux/Image
```

Click Next.

In `Templates window`, if not done before, update `Vitis IDE Examples` and `Vitis IDE Libraries`.

Select Host Examples:
In `Find`, type: `vector add` to search for the `Vector Addition` example.

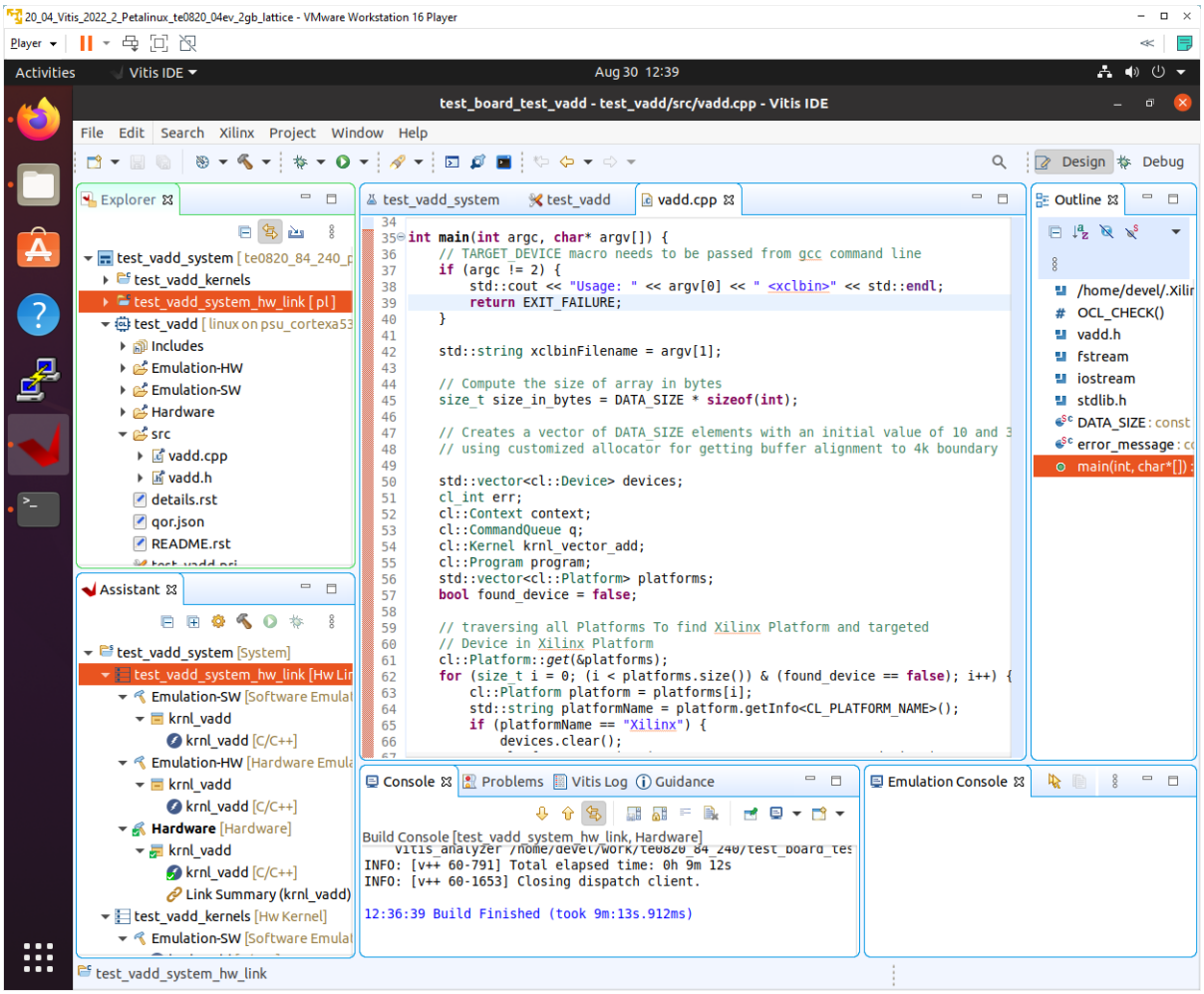
Select: `Vector Addition`
Click Finish
New project template is created.

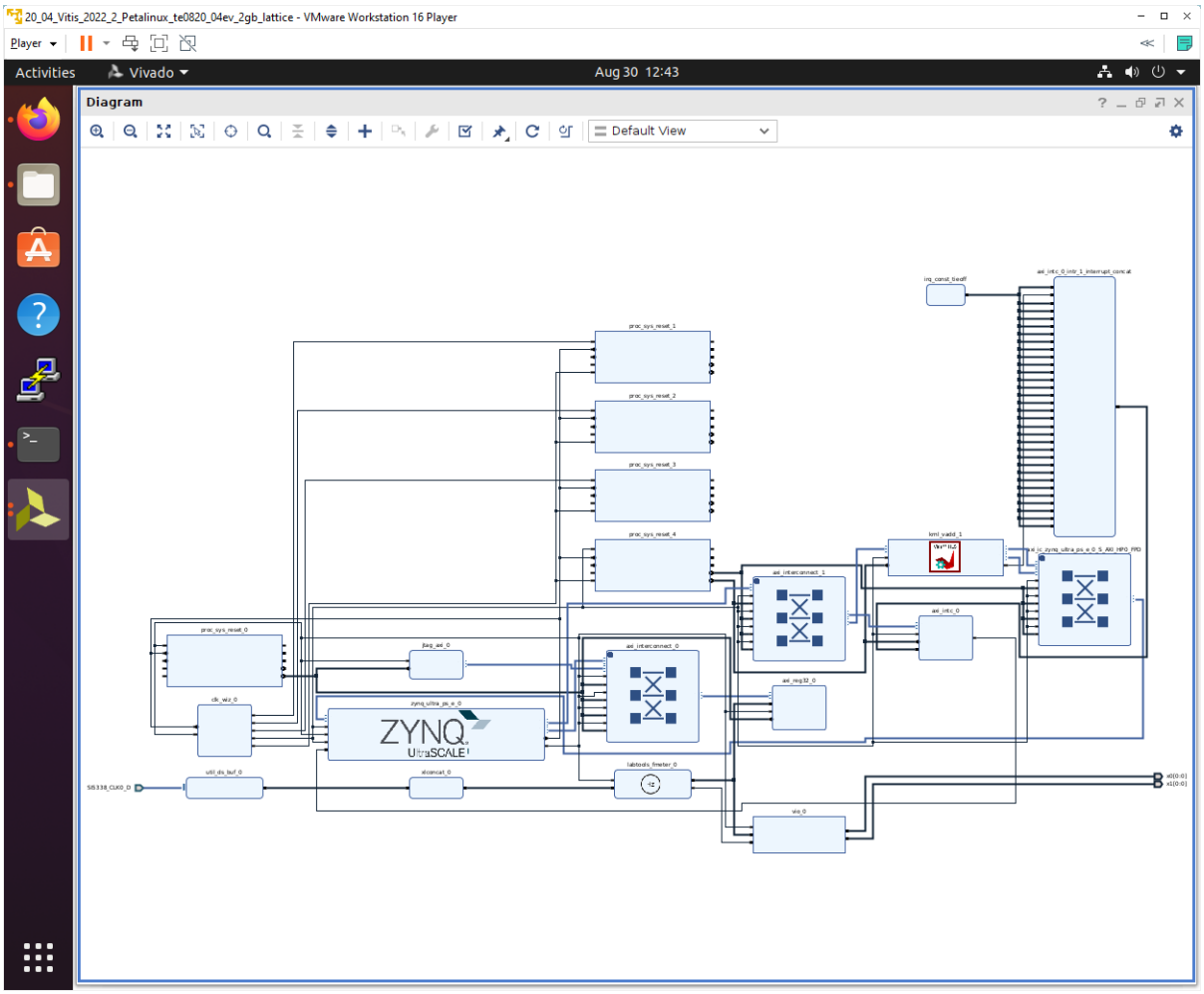
In test_vadd window menu “Active build configuration” switch from `SW Emulation` to `Hardware`.

In “Explorer” section of Vitis IDE, click on: `test_vadd_system[TE0821_03_240_pfm]` to select it.

Right Click on: `test_vadd_system[TE0821_03_240_pfm]` and select in the opened sub-menu: `Build project`

Vitis will compile. This step can take some time.





Created extended HW with integrated vadd IP block can be open and analysed in Vivado 2023.2.

5.3 Run Compiled test_vadd Example Application

The `sd_card.img` file is output of the compilation and packing by Vitis. It is located in directory:

```
~/work/TE0821_03_240/test_board_test_vadd/test_vadd_system/Hardware/package/sd_card.img
```

Write the sd card image `sd_card.img` to SD card.

In Windows Pro 10 (or Windows 11 Pro) PC, inst all program `Win32DiskImager` for this task. Win32 Disk Imager can write raw disk image to removable devices.
https://win32diskimager.org/

Insert the SD card to the TE0701-06 carrier board.

Connect PC USB terminal (115200 bps) card to the TE0701-06 carrier board.

Connect Ethernet cable to the TE0701-06 carrier board.

Power on the TE0701-06 carrier board.

In PC, find the assigned serial line COM port number for the USB terminal. In case of Win 10 or Win 11 use `device manager`.

In PC, open serial line terminal with the assigned COM port number. Speed 115200 bps.

On TE0701-06, reset button to start the system. USB terminal starts to display booting information.

In PC terminal, type:

```
sh-5.0# cd /media/sd-mmcblk1p1/
sh-5.0# ./test_vadd krnl_vadd.xclbin
```

The application test_vadd should run with this output:

```
INFO: Reading krnl_vadd.xclbin
Loading: 'krnl_vadd.xclbin'
Trying to program device[0]: edge
Device[0]: program successful!
TEST PASSED
sh-5.0#
```

The Vitis application has been compiled to HW and evaluated on custom system with extensible custom `TE0821_03_240_pfm` platform.

In PC terminal type:

```
# halt
```

System is halted. Messages relate to halt of the system can be seen on the USB terminal.

The SD card can be safely removed from the TE0701-06 carrier board, now.

The TE0701-06 carrier board can be disconnected from power.

System can be connected to the X11 terminal running on your PC Ubuntu with PuTTY application via Ethernet.

Find Ethernet IP address of your board by `ifconfig` command in PetaLinux terminal.
In PC Ubuntu OS, open `PuTTY` application.
In `PuTTY`, set Ethernet IP of your board.
In `PuTTY`, select `checkbox SSH->X11->Enable X11 forwarding`.

Use PC Ubuntu mouse and keyboard. In `PuTTY`, open PetaLinux terminal and login as: user: `root` pswd: `root`.

In opened PetaLinux terminal, start X11 desktop `x-session-manager` by typing:

```
root@Trenz:~# x-session-manager &
```

Click on X11 icon (A Unicode capable `rxvt`)

Terminal opens as an X11 graphic window. In X11 terminal `rxvt`, use Ubuntu PC keyboard and type:

```
sh-5.0# cd /media/sd-mmcblk1p1/

sh-5.0# ./test_vadd krnl_vadd.xclbin
```

The application test_vadd should run with this output:

```
INFO: Reading krnl_vadd.xclbin
Loading: 'krnl_vadd.xclbin'
Trying to program device[0]: edge
Device[0]: program successful!
TEST PASSED
sh-5.0#
```

The test_board has been running the PetaLinux OS and drives simple version of an X11 GUI on Ubuntu desktop.Application test_vadd has been started from X11 xrvt terminal emulator.

Close the rxvt terminal emulator by click ”x” icon (in the upper right corner) or by typing:

```
sh-5.0# exit
```

In X11, click `Shutdown` icon to safely close PetaLinux running on the test board.

System on the test board is halted. Messages related to halt of the system can be seen on the PC USB terminal.

The SD card can be safely removed from the test_board, now.
Close the PC USB terminal application.
The TE0701-06 carrier board can be disconnected from power, now.

6 Vitis AI 3.0 DPUCZDX8V_VAI_v3.0 Installation

This test implements simple AI 3.0 demo to verify DPU integration to our custom extensible platform. This tutorial follows Xilix Vitis Tutorial for zcu104 with necessary fixes and customizations required for our case.

We have to install correct Vitis project with the DPU instance from this repository:

https://github.com/Xilinx/Vitis-AI/tree/3.0/dpu

Page description contains table with supported targets. Use the line if theis table dedicated to DPUCZDX8G DPU for MPSoC and Kria K26 devices.

It is link for download of the programmable logic based DPU, targeting general purpose CNN inference with full support for the Vitis AI ModelZoo.
Supports either the Vitis or Vivado flows on 16nm Zynq® UltraScale+™ platforms.

Click on the Download link in the column: Reference Design

This will result in download of file:

```
~/Downloads/DPUCZDX8V_VAI_v3.0.tar.gz
```

It contains directory

```
~/Downloads/DPUCZDX8V_VAI_v3.0
```

Copy this directory to the directory:

```
~/work/DPUCZDX8V_VAI_v3.0
```

It contains HDL code for the DPU and also source files and project files to test the DPU with AI resnet50 inference example.

6.1 Create and Build Vitis Design

Create new directory `test_board_dpu_trd` to test Vitis extendable flow example `dpu trd`

~/work/TE0821_03_240/test_board_dpu_trd

Current directory structure:

```
~/work/TE0821_03_240/test_board
~/work/TE0821_03_240/test_board_pfm
~/work/TE0821_03_240/test_board_test_vadd
~/work/TE0821_03_240/test_board_dpu_trd
```

Change working directory:

```
$cd ~/work/TE0821_03_240/test_board_dpu_trd
```

In Ubuntu terminal, start Vitis by:

```
$ vitis --classic --workspace . &
```

In Vitis IDE Launcher, select your working directory

```
~/work/TE0821_03_240/test_board_dpu_trd
```

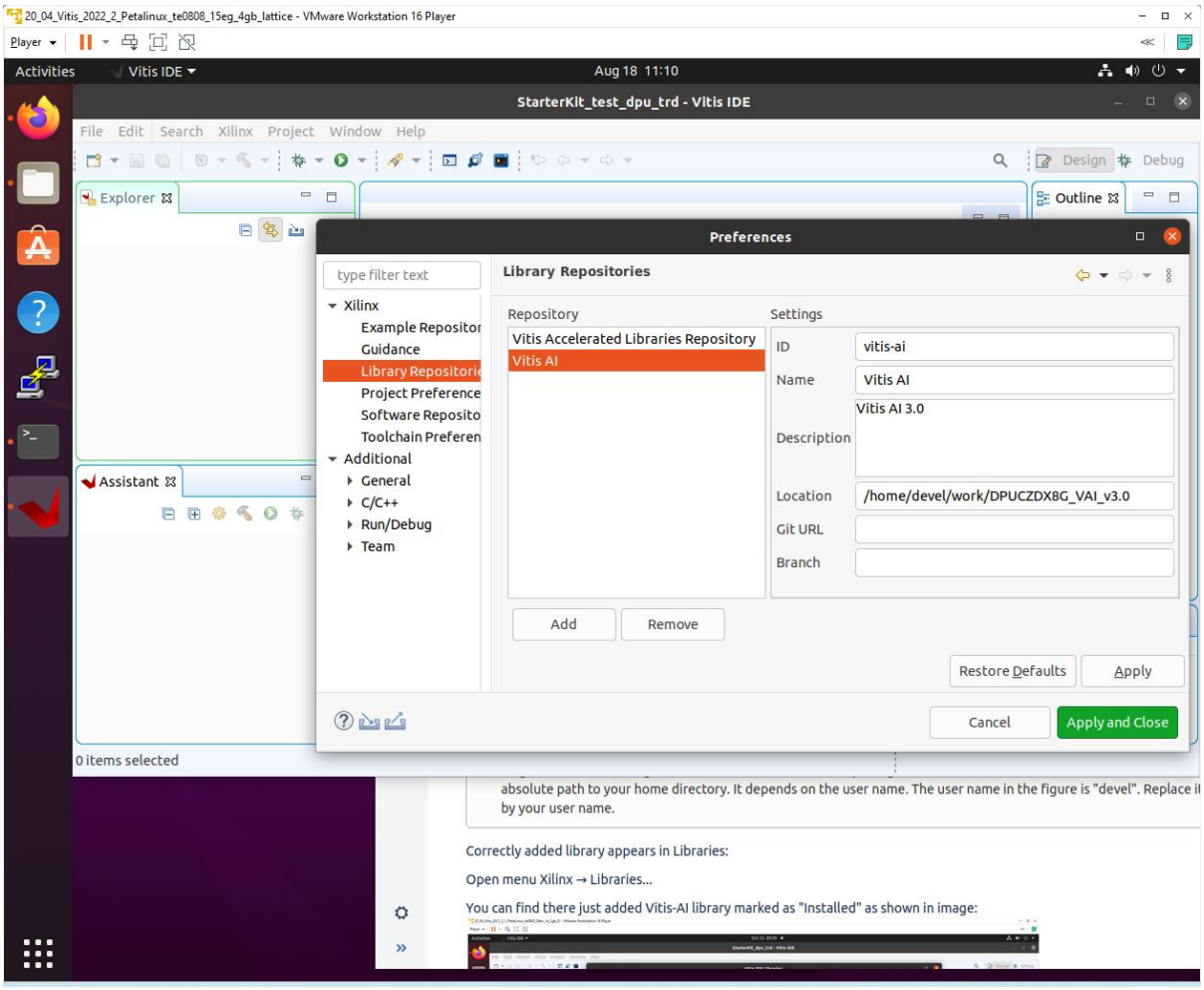
Click on `Launch` to start `Vitis`.

6.2 Add DPU Project template to the Vitis Extensible Flow

Open menu `Window → Preferences`

Go **`to`** `Library Repository` tab

Add `Vitis-AI` by clicking `Add` button and fill the form as shown below, use absolute path to your home folder in field `Location`



Click `Apply` and `Close`.

Field `Location` says that the Vitis-AI repository from github has been allready cloned into

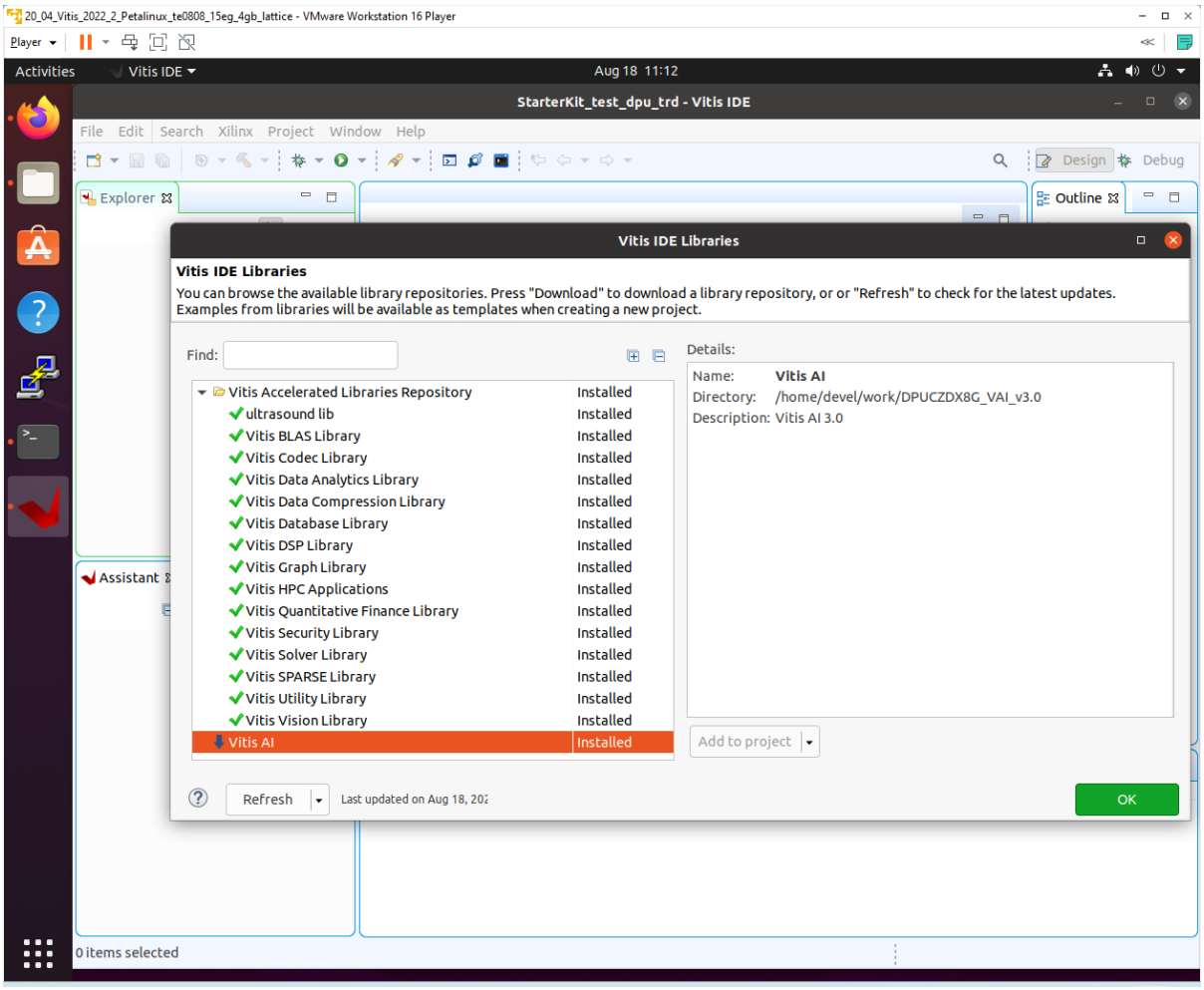
```
~/work/DPUCZDX8V_VAI_v3.0
```

folder, in the stage of Petalinux configuration. Use the `absolute path` to your home directory. It depends on the user name. The user name in the figure is "devel". Replace it by your user name.

Correctly added library appears in Libraries:

Open menu `Xilinx → Libraries...`

You can find there just added Vitis-AI library marked as `Installed`



6.3 Configure Project for the Vitis Extensible Flow with DPU

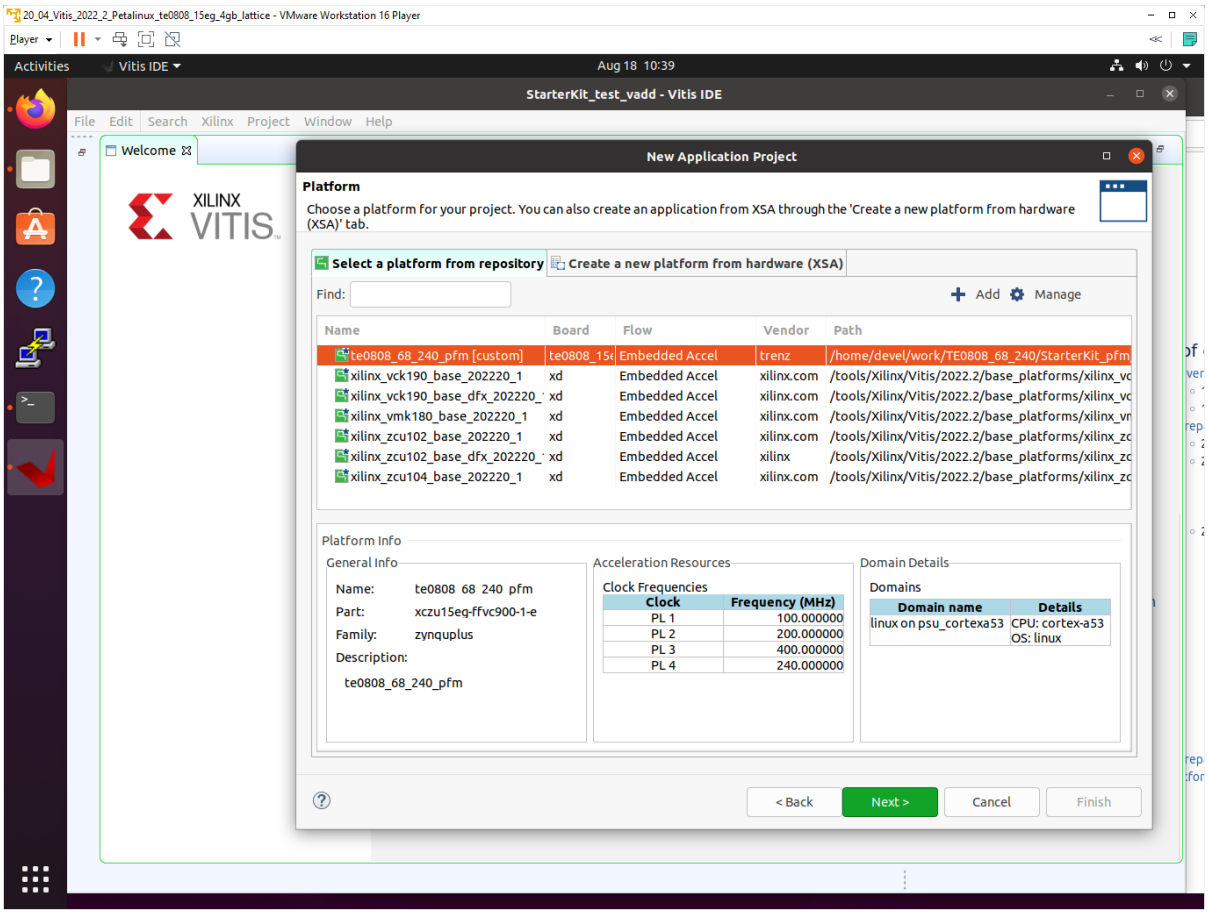
Select `File -> New -> Application project`. Click Next.

Skip welcome page, if it is shown.

Click on `[+ Add]` icon and select the custom extensible platform `TE0821_03_240_pfm[custom]` in the directory:

`~/work/TE0821_03_240/test_board_pfm/TE0821_03_240_pfm/export/TE0821_03_240_pfm`

We can see available PL clocks and frequencies. PL4 with 240 MHz clock was set as the default in the platform creation process.



Click Next.
In `Application Project Details` window type into Application project name: `dpu_trd`
Click Next.
In `Domain` window type (or select by browse):

“Sysroot path”:
```
~/work/TE0821_03_240/test_board_pfm/sysroots/cortexa72-cortexa53-xilinx-linux
```

“Root FS”:
```
~/work/TE0821_03_240/test_board/os/petalinux/images/linux/rootfs.ext4
```

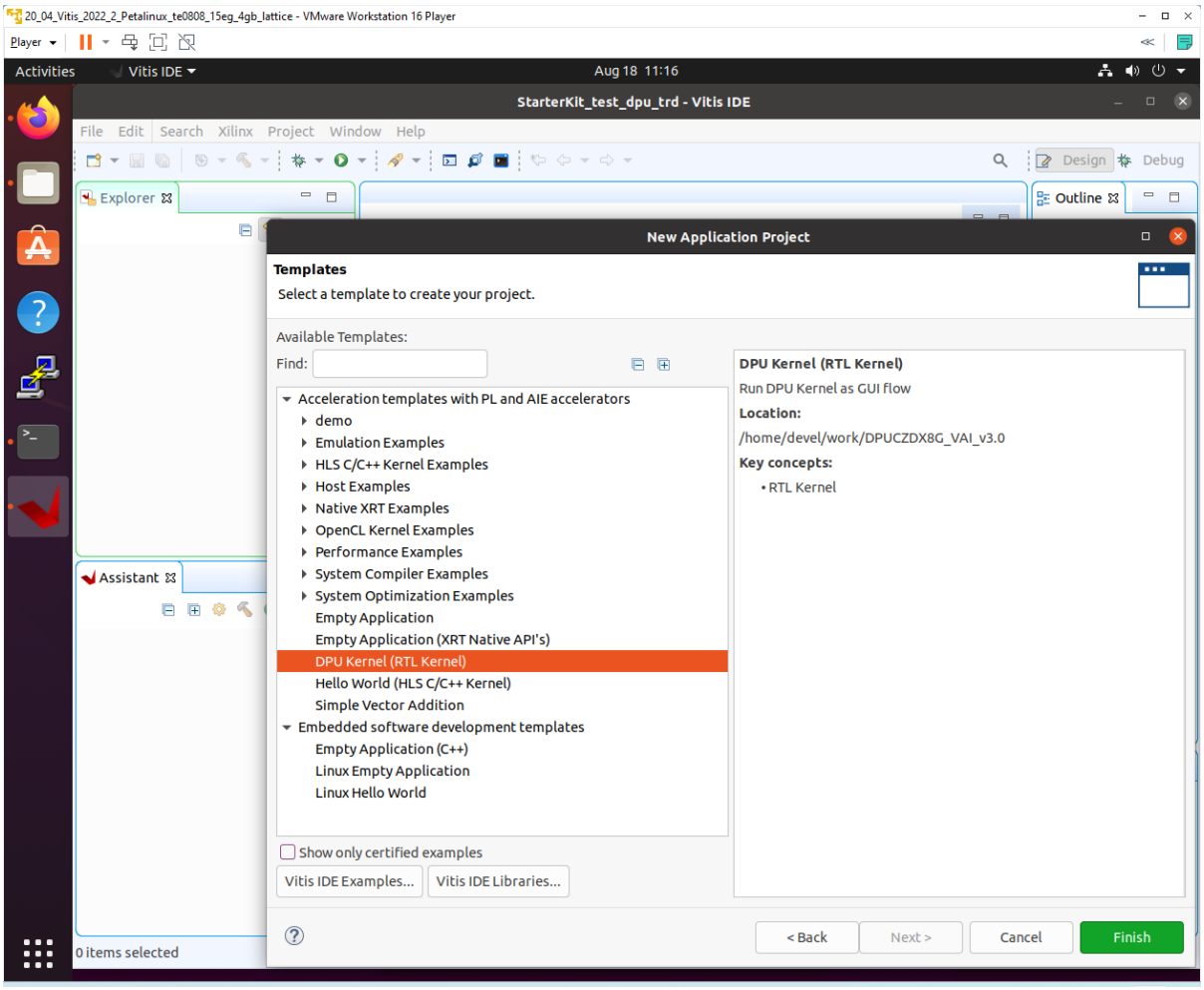
“Kernel Image”:
```
~/work/TE0821_03_240/test_board/os/petalinux/images/linux/Image
```

Click Next.

In `Templates window`, if not done before, update `Vitis IDE Examples` and `Vitis IDE Libraries`

In “Find”, type: `dpu` to search for the `DPU Kernel (RTL Kernel)` example.

Select: `DPU Kernel (RTL Kernel)`



Click Finish
New project template is created.

In dpu_trd window menu `Active build configuration` switch from `SW Emulation` to `Hardware`

File `dpu_conf.vh` located at `dpu_trd_kernels/src/prj/Vitis` directory contains DPU configuration.

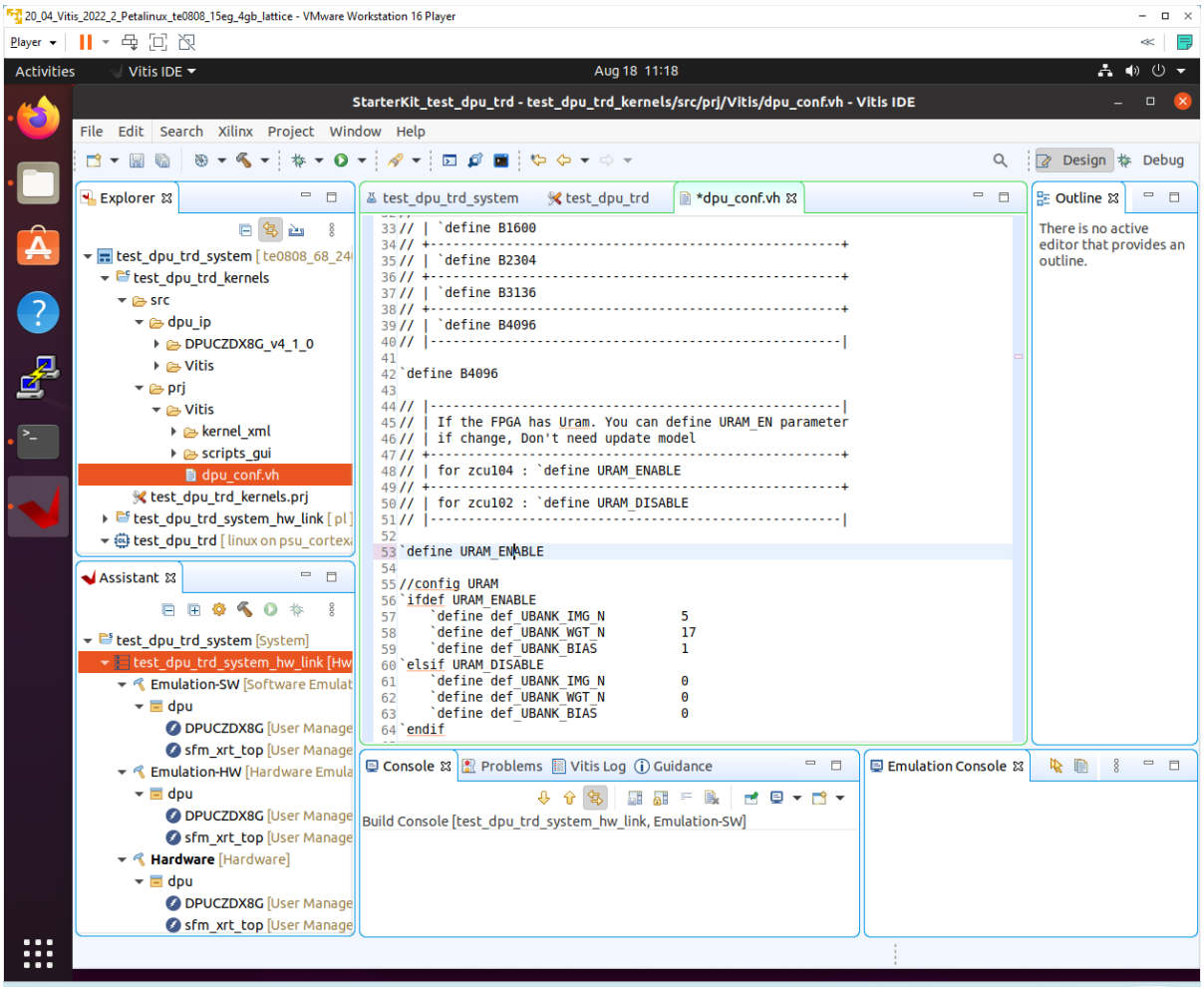
Open file `dpu_conf.vh` and change in line 37:

```
`define B4096
```

to

```
`define B1600
```

and save modified file.



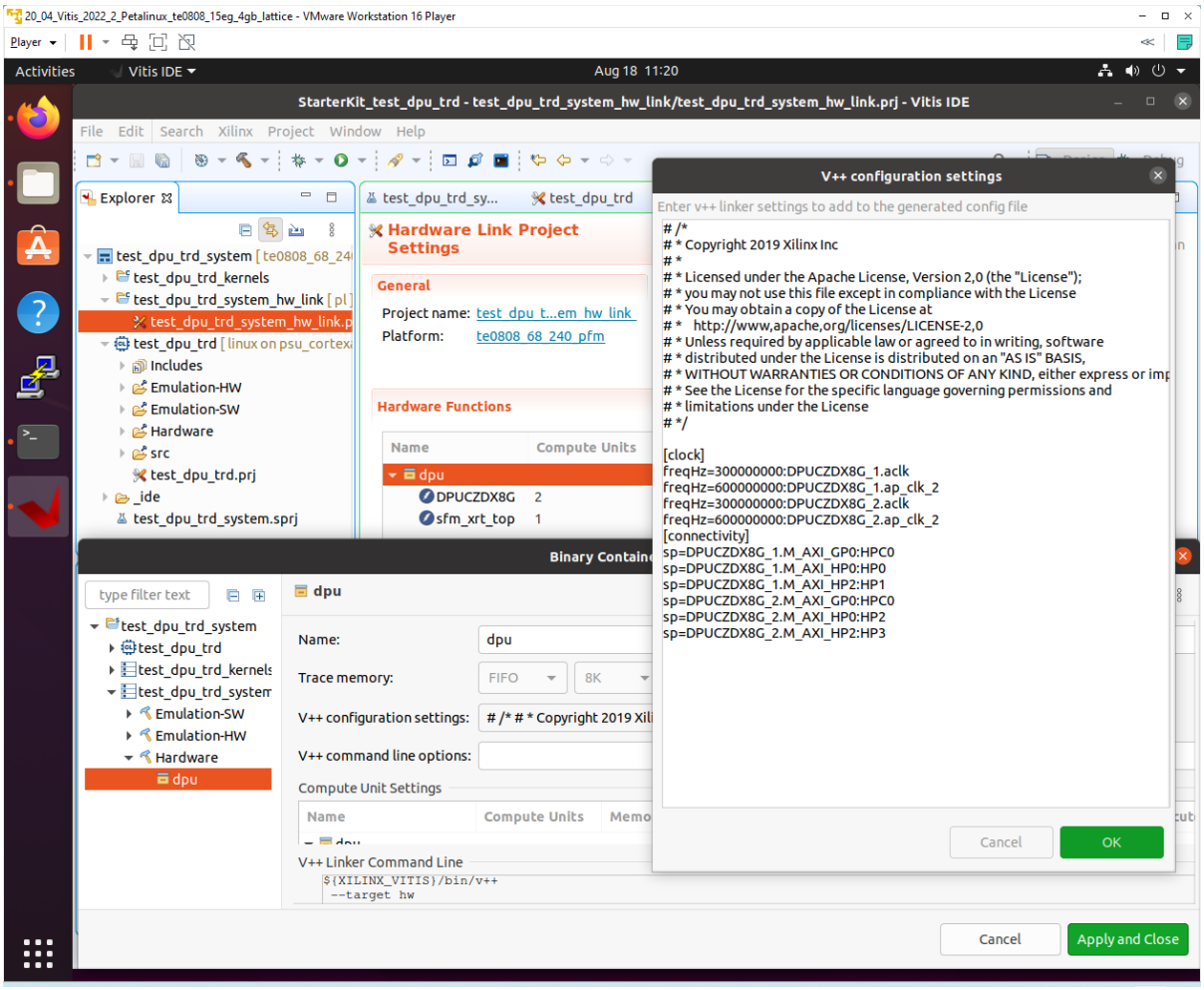
Go to `dpu_trd_system_hw_link` and double click on `dpu_trd_system_hw_link.prj`

Remove `sfm_xrt_top` kernel from binary container by right clicking on it and choosing remove.

Reduce number of DPU kernels to one.

6.4 Configure Connection of DPU kernel

On the same tab right click on `dpu` and choose `Edit V++ Options`



Click "`...`" button on the line of `V++ Configuration Settings` and modify configuration as follows:

```
[clock]
freqHz=200000000:DPUCZDX8G_1.aclk
freqHz=400000000:DPUCZDX8G_1.ap_clk_2

[connectivity]
sp=DPUCZDX8G_1.M_AXI_GP0:HPC0
sp=DPUCZDX8G_1.M_AXI_HP0:HP0
sp=DPUCZDX8G_1.M_AXI_HP2:HP1
```

6.5 Build the test_dpu_trd Project

In "`Explorer`" section of Vitis 2023.3.1. IDE, click on:

```
dpu_trd_system[te0821_03_240_pfm]
```

to select it.

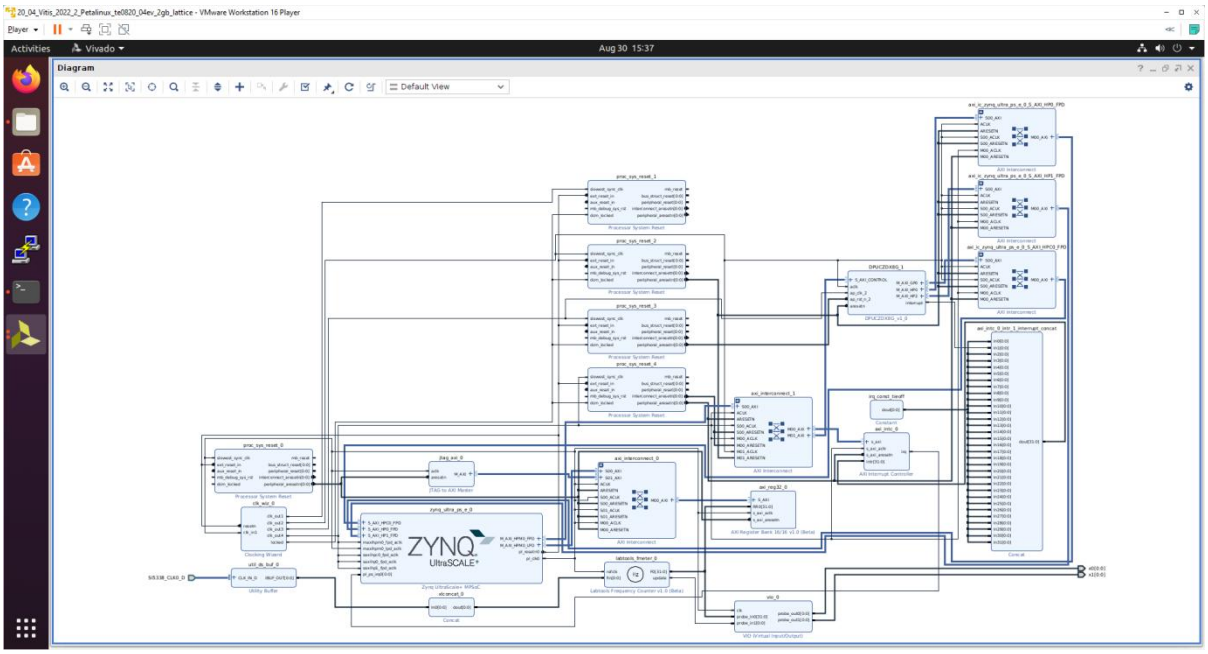
Right click on:

```
dpu_trd_system[te0821_03_240_pfm]
```

and select in the opened sub-menu: `Build project`

Compilation takes some time (approximately 30 minutes).

Created extended HW with integrated DPU with configuration `B4096` can be open and analysed in Vivado 2023.2



7 Prepare SD card with test_dpu_trd DPU

Write `sd_card.img` to SD card using SD card reader.

The **sd_card.img** file is output of the compilation and packing by Vitis. It is located in directory:

```
~/work/TE0821_03_240/test_board_dpu_trd/dpu_trd_system/Hardware/package
```

In Windows 10 (or Windows 11) PC, inst all program `Win32DiskImager` for this task. Win32 Disk Imager can write raw disk image to removable devices.
https://win32diskimager.org/

Boot the board and open terminal on the board either by connecting serial console connection, or by opening ethernet connection to ssh server on the board, or by opening terminal directly using window manager on board. Continue using the embedded board terminal.

Detailed guide how to run embedded board and connect to it can be found in Run Compiled Example Application for Vector Addition.

7.1 Resize EXT4 Partition

Check ext4 partition size by:

```
root@Trenz:~# cd /
root@Trenz:~# df .
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/root               564048    398340    122364  77% /
```

Resize partition

```
root@Trenz:~# resize-part /dev/mmcblk1p2
```

Check ext4 partition size again, you should see:

```
root@Trenz:~# df . -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                 6.1G    390.8M      5.4G   7% /
```

The available size would be different according to your SD card size.

Set path to `dpu.xclbin` :

```
root@petalinux:~# export
XLNX_VART_FIRMWARE=/run/media/mmcblk1p1/dpu.xclbin
```

7.2 Test the Integrated DPUCZDX8G

For both tested modules, the integrated DPU can be tested by command:

```
 xdputil query
```

Command and reply in case of module with ID=3 (DPU configuration B1600):

```
sh-5.1# xdputil query

WARNING: Logging before InitGoogleLogging() is written to STDERR

F20250216 10:19:12.589970   922 xrt_bin_stream.cpp:59] Check failed:
t.read(&buffer_[0], buffer_.size()).good()

/usr/bin/xdputil: line 20:   922 Aborted
/usr/bin/python3 -m xdputil $*

sh-5.1# export XLNX_VART_FIRMWARE=/run/media/mmcblk1p1/dpu.xclbin

sh-5.1# xdputil query

{

    "DPU IP Spec":{
```

```
        "DPU Core Count":1,
        "IP version":"v4.1.0",
        "generation timestamp":"2023-02-21 21-30-00",
        "git commit id":"7d32c41",
        "git commit time":2023022121,
        "regmap":"1to1 version"
    },
    "VAI Version":{
        "libvart-runner.so":"Xilinx vart-runner Version: 3.5.0-
b7953a2a9f60e23efdfced5c186328dd1449665c  2024-04-18-10:03:55 ",
        "libvitis_ai_library-dpu_task.so":"Advanced Micro Devices
vitis_ai_library dpu_task Version: 3.5.0-
b7953a2a9f60e23efdfced5c186328dd1449665c  2023-06-29 03:20:28 [UTC] ",
        "libxir.so":"Xilinx xir Version: xir-
b7953a2a9f60e23efdfced5c186328dd1449665c 2024-04-18-09:27:26",
        "target_factory":"target-factory.3.5.0
b7953a2a9f60e23efdfced5c186328dd1449665c"
    },
    "kernels":[
        {
            "AIE Frequency (Hz)":0,
            "DPU Arch":"DPUCZDX8G_ISA1_B1600",
            "DPU Frequency (MHz)":300,
            "IP Type":"DPU",
            "Load Parallel":2,
            "Load augmentation":"enable",
            "Load minus mean":"disable",
            "Save Parallel":2,
            "XRT Frequency (MHz)":300,
            "cu_addr":"0xa0010000",
            "cu_handle":"0xaaaae3d0f700",
            "cu_idx":0,
            "cu_mask":1,
            "cu_name":"DPUCZDX8G:DPUCZDX8G_1",
            "device_id":0,
            "fingerprint":"0x101000056010404",
            "name":"DPU Core 0"
```

```
        }
    ]
}
sh-5.1#
```

7.3 Test resnet50_pt model

The AMD DPUCZDX8G in configuration B1600 can be tested with AI 3.0 resnet50_pt model precompiled for the B1600 configuration.

Copy the precompiled AI 3.0 resnet50_pt model files accompanying this application note:

From

```
B1600\app\model\md5sum.txt
B1600\app\model\resnet50_pt.prototxt
B1600\app\model\resnet50_pt.xmodel
```

to target system directory

```
~/app/model/
```

Change the directory to `~/app/model/`

```
cd ~/app/model/
```

Note: If you integrate DPU with B0512, B1024 or B1600 configuration, use files from the evaluation package prepared for that configuration.

Test `resnet50_pt` model by command

```
sh-5.1# xdputil benchmark resnet50_pt.xmodel 1
```

Result of test

```
sh-5.1# export XLNX_VART_FIRMWARE=/run/media/mmcblk1p1/dpu.xclbin
sh-5.1# pwd
/home/root/app/model
sh-5.1# xdputil benchmark resnet50_pt.xmodel 1
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20250216 11:37:00.962606  1023 test_dpu_runner_mt.cpp:477] shuffle
results for batch...
I20250216 11:37:00.963408  1023 performance_test.hpp:73] 0% ...
I20250216 11:37:06.963547  1023 performance_test.hpp:76] 10% ...
I20250216 11:37:12.963698  1023 performance_test.hpp:76] 20% ...
I20250216 11:37:18.963845  1023 performance_test.hpp:76] 30% ...
I20250216 11:37:24.963991  1023 performance_test.hpp:76] 40% ...
```

```
I20250216 11:37:30.964146  1023 performance_test.hpp:76] 50% ...
I20250216 11:37:36.964300  1023 performance_test.hpp:76] 60% ...
I20250216 11:37:42.964447  1023 performance_test.hpp:76] 70% ...
I20250216 11:37:48.964599  1023 performance_test.hpp:76] 80% ...
I20250216 11:37:54.964742  1023 performance_test.hpp:76] 90% ...
I20250216 11:38:00.964879  1023 performance_test.hpp:76] 100% ...
I20250216 11:38:00.964937  1023 performance_test.hpp:79] stop and
waiting for all threads terminated....
I20250216 11:38:00.987602  1023 performance_test.hpp:85] thread-0
processes 1649 frames
I20250216 11:38:00.987635  1023 performance_test.hpp:93] it takes 22676
us for shutdown
I20250216 11:38:00.987655  1023 performance_test.hpp:94] FPS= 27.4722
number_of_frames= 1649 time= 60.0242 seconds.
I20250216 11:38:00.987706  1023 performance_test.hpp:96] BYEBYE
Test PASS.
sh-5.1#
```

Benchmark indicates **27.47 FPS** for inference of `resnet50_pt` model. Power consumption of the system during benchmark is increased to **8.6 W**.

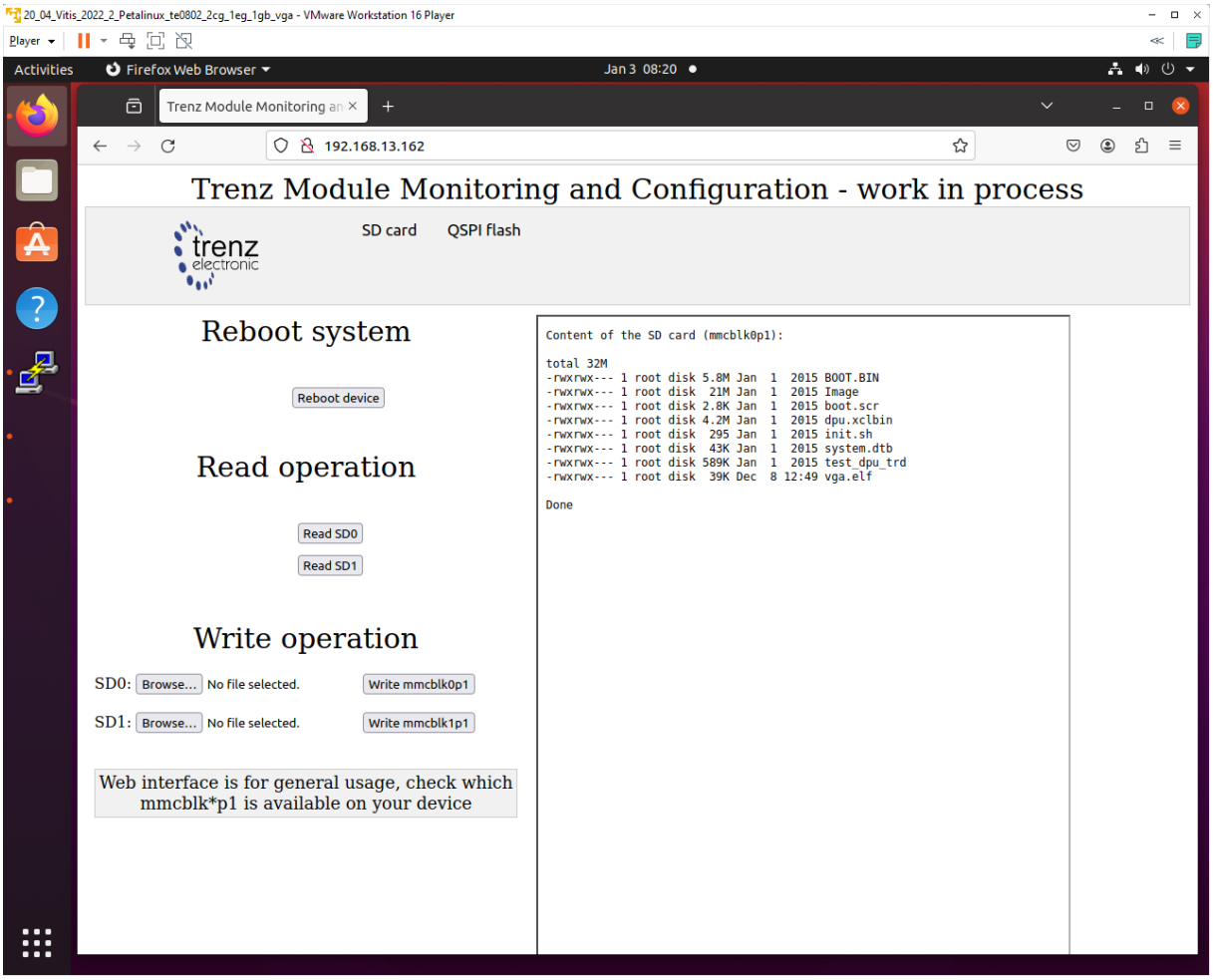
Compilation of AI 3.0 models for the AMD DPUCZDX8G with AI 3.5 SW is described in separate application note and evaluation package [10]:

Compilation of AI 3.0 models for Vitis 2023.2, AI 3.5 SW, AI 3.0 DPUCZDX8G.

7.4 Remote Monitoring and Configuration Support

The configured OS includes work in progress version of a remote monitoring and configuration support server. It can be used for remote reading of content of the SD card partition `mmcblk1p1` .

Button Reboot device can be used for system reboot. Ethernet connection is lost, but remote PC www browser remains open and waits for possible reconnection.



After reboot of the evaluation board, the network DHCP server assigns Ethernet address to the evaluation board.

If the network DHCP address assignment algorithm assigns the identical Ethernet address, the page can be refreshed and the connection is re-established again.

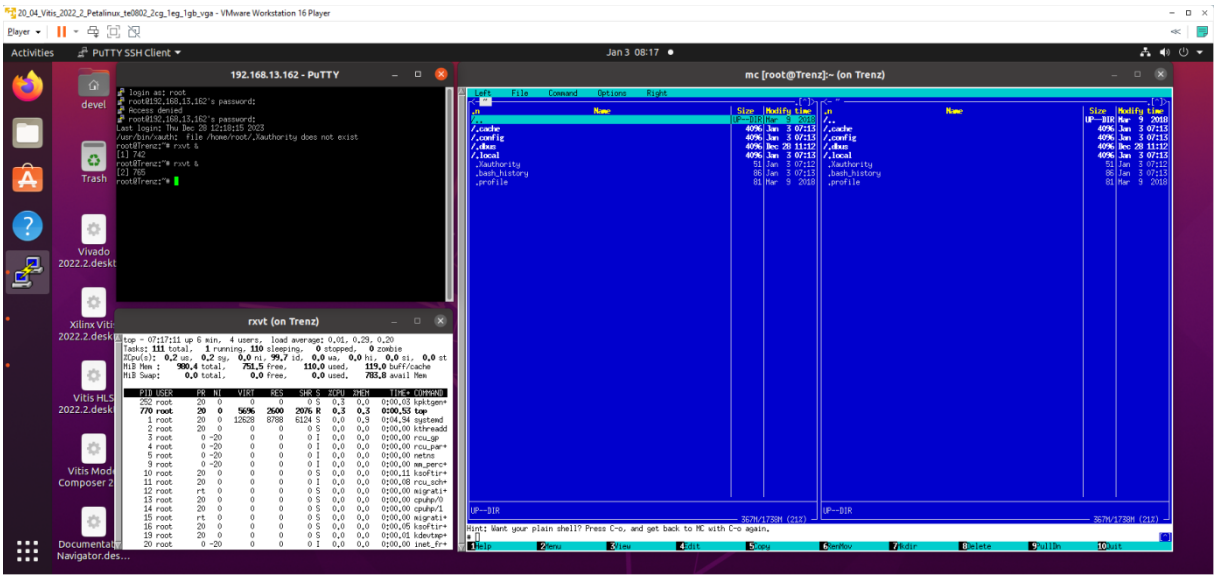
If the network DHCP address assignment algorithm assigns different Ethernet address, the connection has to be established on the new Ethernet address.

7.5 Remote Control from Ubuntu X11 Desktop.

The configured OS also supports X11 desktop on remote PC via Ethernet.
In remote PC in Ubuntu OS, in PuTTY terminal utility with ssh Ethernet connection to the board with enabled X11 forwarding.

Openning.
Log in to the evaluation board as user root with pswd root
Start two rxvt terminal emulators by typing in PuTTY terminal:
rxvt &
rxvt &
In first rxvt terminal emulator window start utility
top

In second rxvt terminal emulator start
mc
You can see two applications running on the evaluation board with output on the remote desktop. Remote PC kbd and mouse are used for control of these applications.



Closing.
On remote PC, close top utility by Ctrl-C. Stop `mc` utility by F10.
Close open terminal emulators by typing exit or by mouse click on x icon in the right top corner of terminal emulator window. Close `PuTTY` connection by typing exit or by mouse click on x icon in the right top corner of `PuTTY` window.

7.6 Remote Control in x-session-manager on Ubuntu X11 Desktop.

The configured OS also supports `x-session-manager` on X11 desktop on remote PC connected via Ethernet to the evaluation board.

Opening.
In remote PC in Ubuntu OS, start `PuTTY` terminal utility with ssh Ethernet connection to the board with enabled `X11` forwarding.
Log in to the evaluation board as user root with pswd root
In `PuTTY` terminal, start `x-session-manager` by typing:

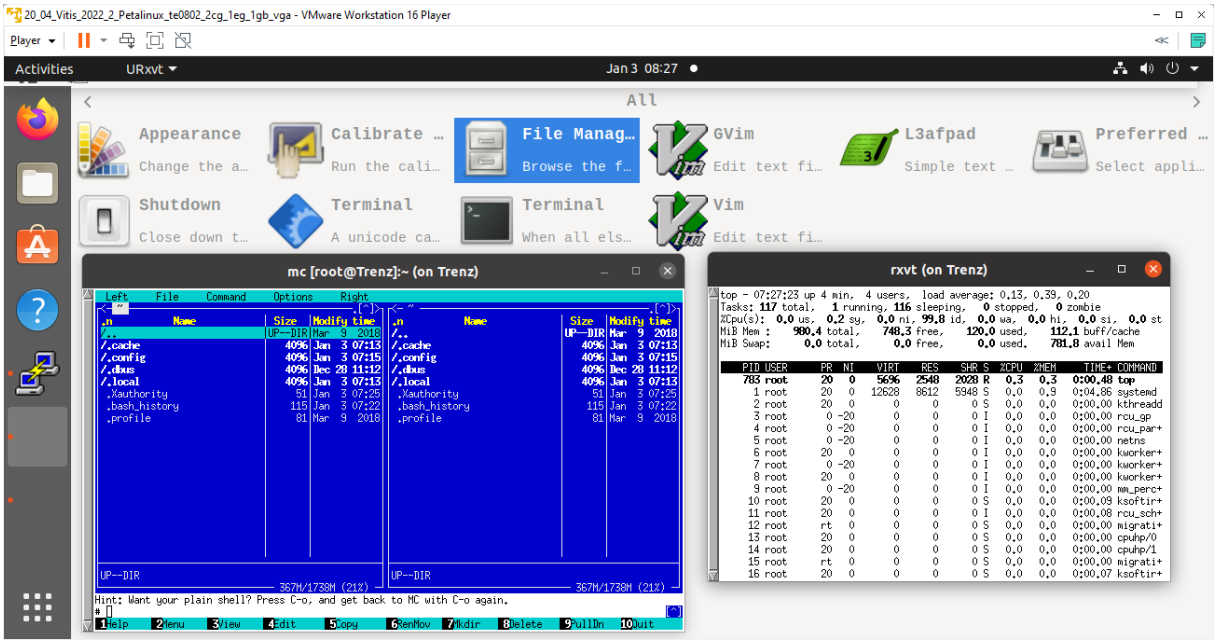
```
x-session-manager &
```

The desktop (displayed on the VGA display of the evaluation board) is also displayed in the remote PC X11 desktop. Start two rxvt terminal emulators by typing in PuTTY terminal:

```
rxvt &
rxvt &
```

In first `rxvt` terminal emulator window start utility `top`
In second `rxvt` terminal emulator start midnight commander `mc`

You can see two applications running on the evaluation board with output on the remote desktop. Remote PC kbd and mouse are used for control of these applications.



Closing.
On remote PC, close top utility by `Ctrl-C`. Stop `mc` utility by key `F10`.
Close open terminal emulators by typing exit or by mouse click on x icon in the right top corner of terminal emulator window. Close `PuTTY` connection by typing exit or by mouse click on x icon in the right top corner of `PuTTY` window.

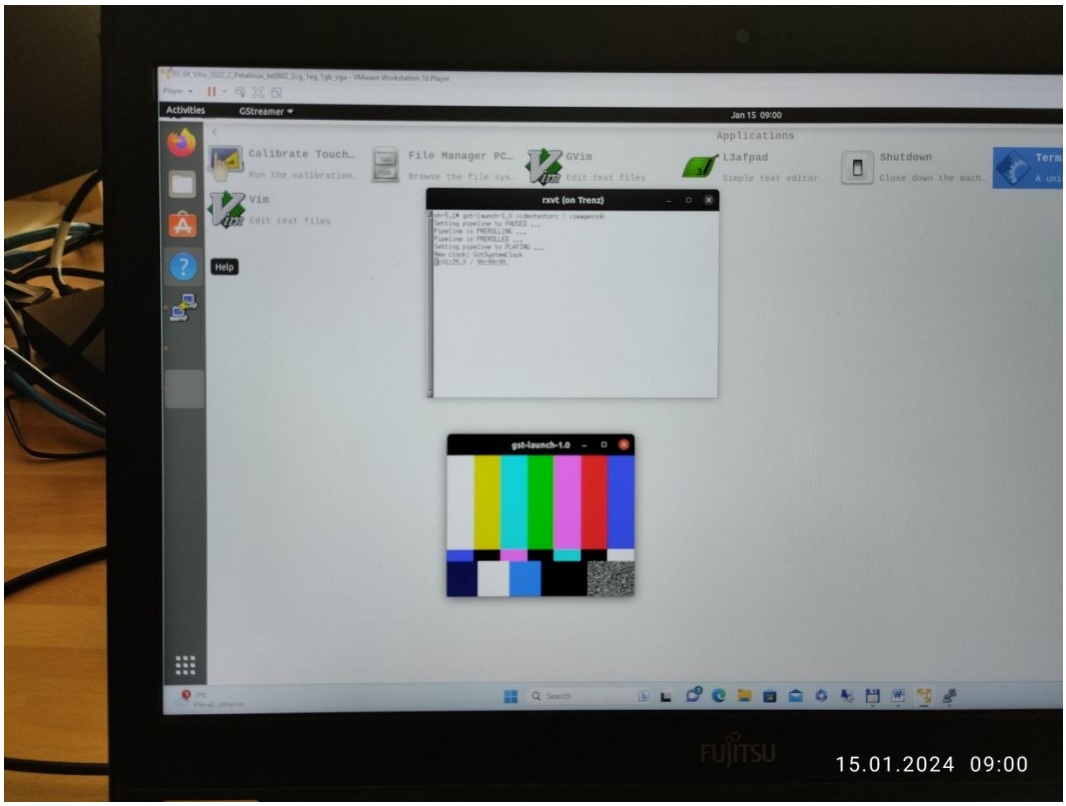
7.7 Display Test Pattern and Test USB Camera

Complete video chain can be tested with output to the X11 desktop.

To display the test pattern, use this gstreamer command:

```
gst-launch-1.0 videotestsrc ! ximagesink
```

Video output is directed to the local HD VGA display, if the command is started from local X11 console.



Test pattern is displayed on remote PC X11 desktop

7.8 Vitis AI 3.0 TE0821-01-2AE31KA, TE0701-06, DPU (B1024)

Vitis AI 3.0 exampes	Performance input from camera e2e (-t 1) [FPS]	Power with camera e2e (-t 1) [W]	Performance input from file e2e (-t 3) [FPS]	Power with input from file e2e (-t 3) [W]	GigaOps input from file e2e (-t 3) [Gops]
Yolov4 face mask detection Model: pt_face-mask-detection_512_512_0.67G_3.0	20.0	7.7	67.0	7.5	44.9
Vehicleclassification vehicle make Model: pt_vehicle-make-classification_VMMR_224_224_3.64G_3.0	20.0	8.1	44.3	7.9	161.2
Vehicleclassification vehicle type Model: pt_vehicle-type-classification_CarBodyStyle_224_224_3.64G_3.0	20.0	8.1	44.3	7.9	161.2
Classification vehicle color Model: pt_vehicle-color-classification_VCoR_224_224_3.64G_3.0	20.0	8.2	44.3	8.0	161.2
Classification Model: pt_resnet50_imagenet_224_224_8.2G_3.0	16.1	8.8	19.3	8.2	158.3
Classification Model: pt_resnet50_imagenet_224_224_0.3_5.8G_3.0	20.0	8.8	24.9	8.2	144.4
Classification Model: pt_resnet50_imagenet_224_224_0.4_4.9G_3.0	20.0	8.6	28.2	8.2	138.2
Classification Model: pt_resnet50_imagenet_224_224_0.5_4.1G_3.0	20.0	8.5	32.1	8.2	131.6
Classification Model: pt_resnet50_imagenet_224_224_0.6_3.3G_3.0	20.0	8.4	37.3	8.2	123.1
Classification Model: pt_resnet50_imagenet_224_224_0.7_2.5G_3.0	20.0	8.3	44.1	8.1	110.2

Measurement conditions:

- ID=1, TE0821-01-2AE31KA module 2cg-1e, 4GB DDR4, TE0701-06 carrier board
- DPU in B1024 configuration, 200/400 MHz clock.
- USB WWW camera ETERNICO ET201 Full HD, sensor JX_F23, 1920x1080, 20 FPS
- Keyboard RPi
- Mouse RPi
- Remote X11 desktop
- Power supply 12V/5A
- Power measured at the 230V power plug

7.9 Vitis AI 3.0 TE0821-01-3AE31KA, TE0701-06, DPU (B1600)

Vitis AI 3.0 exampes	Performance input from camera e2e (-t 1) [FPS]	Power with camera e2e (-t 1) [W]	Performance input from file e2e (-t 3) [FPS]	Power with input from file e2e (-t 3) [W]	GigaOps input from file e2e (-t 3) [Gops]
Yolov4 face mask detection Model: pt_face-mask-detection_512_512_0.67G_3.0	20.0	7.7	76.1	7.6	50.9
Vehicleclassification vehicle make Model: pt_vehicle-make-classification_VMMR_224_224_3.64G_3.0	20.0	7.9	61.2	8.4	222.7
Vehicleclassification vehicle type Model: pt_vehicle-type-classification_CarBodyStyle_224_224_3.64G_3.0	20.0	7.9	61.2	8.4	222.7
Classification vehicle color Model: pt_vehicle-color-classification_VCoR_224_224_3.64G_3.0	20.0	7.9	61.2	8.4	222.7
Classification Model: pt_resnet50_imagenet_224_224_8.2G_3.0	20.0	8.8	27.5	8.6	225.5
Classification Model: pt_resnet50_imagenet_224_224_0.3_5.8G_3.0	20.0	8.5	36.0	8.6	208.8
Classification Model: pt_resnet50_imagenet_224_224_0.4_4.9G_3.0	20.0	8.4	39.7	8.5	194.5
Classification Model: pt_resnet50_imagenet_224_224_0.5_4.1G_3.0	20.0	8.3	44.1	8.5	180.8
Classification Model: pt_resnet50_imagenet_224_224_0.6_3.3G_3.0	20.0	8.2	45.4	8.4	149.8
Classification Model: pt_resnet50_imagenet_224_224_0.7_2.5G_3.0	20.0	8.0	58.3	8.4	145.7

Measurement conditions:

- ID=3, TE0821-01-3AE31KA module 3cg-1e, 4GB DDR4, TE0701-06 carrier board
- DPU in B1600 configuration, 200/400 MHz clock.
- USB WWW camera ETERNICO ET201 Full HD, sensor JX_F23, 1920x1080, 20 FPS
- Keyboard RPi
- Mouse RPi
- Remote X11 desktop
- Power supply 12V/5A
- Power measured at the 230V power plug

7.10 Vitis AI 3.0 TE0821-01-3BE21FA, TE0701-06, DPU (B1600)

Vitis AI 3.0 exampes	Performance input from camera e2e (-t 1) [FPS]	Power with camera e2e (-t 1) [W]	Performance input from file e2e (-t 3) [FPS]	Power with input from file e2e (-t 3) [W]	GigaOps input from file e2e (-t 3) [Gops]
Yolov4 face mask detection Model: pt_face-mask-detection_512_512_0.67G_3.0	20.0	8.0	77.3	8.0	51.8
Vehicleclassification vehicle make Model: pt_vehicle-make-classification_VMMR_224_224_3.64G_3.0	20.0	8.6	61.7	9.2	224.6
Vehicleclassification vehicle type Model: pt_vehicle-type-classification_CarBodyStyle_224_224_3.64G_3.0	20.0	8.6	61.7	9.2	224.6
Classification vehicle color Model: pt_vehicle-color-classification_VCoR_224_224_3.64G_3.0	20.0	8.6	61.7	9.2	224.6
Classification Model: pt_resnet50_imagenet_224_224_8.2G_3.0	20.0	9.7	27.5	9.5	225.5
Classification Model: pt_resnet50_imagenet_224_224_0.3_5.8G_3.0	20.0	9.3	35.9	9.5	208.2
Classification Model: pt_resnet50_imagenet_224_224_0.4_4.9G_3.0	20.0	9.2	39.9	9.4	195.5
Classification Model: pt_resnet50_imagenet_224_224_0.5_4.1G_3.0	20.0	9.0	44.2	9.4	181.2
Classification Model: pt_resnet50_imagenet_224_224_0.6_3.3G_3.0	20.0	8.8	45.3	9.2	145.5
Classification Model: pt_resnet50_imagenet_224_224_0.7_2.5G_3.0	20.0	8.7	58.6	9.2	146.5

Measurement conditions:

- ID=5, TE0821-01-3BE21FA module 3eg-1e, 2GB DDR4, TE0701-06 carrier board
- DPU in B1600 configuration, 200/400 MHz clock.
- USB WWW camera ETERNICO ET201 Full HD, sensor JX_F23, 1920x1080, 20 FPS
- Keyboard RPi
- Mouse RPi
- Remote X11 desktop
- Power supply 12V/5A
- Power measured at the 230V power plug

8 References

[1] Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout: Support for STM32H573I-DK web server. (Application note, with evaluation package, UTIA). Published for public access from: https://zs.utia.cas.cz/index.php?ids=results&id=1_STM32H573_DK
This application and evaluation package will be based on the STM32CubeH5 Firmware Examples for STM32H5xx Series Application based on NetXDuo: **Nx_WebServer.**
This STM application provides an example of Azure RTOS NetX Duo stack usage on STM32H573G-DK board, it shows how to develop Web HTTP server based application.
https://htmlpreview.github.io/?https://raw.githubusercontent.com/STMicroelectronics/STM32CubeH5/master/Projects/STM32CubeProjectsList.html

[2] Lukáš Kohout, Jiří Kadlec, Zdeněk Pohl: Support for TE0802-02-1BEV2-A board with Vitis AI 3.0 DPU and VGA display (Application note with evaluation package, UTIA). Published for public free access from: https://zs.utia.cas.cz/index.php?ids=results&id=2_TE0802-02-1BEV2-A_AI_3_0_VGA

[3] Lukáš Kohout, Jiří Kadlec, Zdeněk Pohl: Support for TE0802-02-2AEV2-A board with Vitis AI 3.0 DPU and VGA display (Application note, with evaluation package, UTIA). Published for public access from: https://zs.utia.cas.cz/index.php?ids=results&id=3_TE0802-02-2AEV2-A_AI_3_0_VGA

[4] Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout: Support for module-based systems with TE0821 modules on TE0701 carrier board with Vitis AI 3.0 DPU (Application note, with evaluation package, UTIA). Published for free public access from: https://zs.utia.cas.cz/index.php?ids=results&id=4_TE0821_AI_3_0

[5] Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout: Support for module-based systems with TE0820 modules on TE0701 carrier board with Vitis AI 3.0 DPU (Application note, with evaluation package, UTIA). Published for free public access from: https://zs.utia.cas.cz/index.php?ids=results&id=5_TE0820_AI_3_0

[6] Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout, Raissa Likhonina: Description of compilation of Vitis AI 3.0 models for different configurations of AMD DPUs, (Application note, with evaluation package, UTIA). Published for free public access from: https://zs.utia.cas.cz/index.php?ids=results&id=6_TE_AI_3_0

[7] Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout: Support for STM32H573I-DK V1.4.0 web server. (Application note, with evaluation package, UTIA). Published for free public access from: https://zs.utia.cas.cz/index.php?ids=results&id=21_STM32H753_DK_V1_4_0
This application and evaluation package is based on the STM32CubeH5 ver 1.4 Firmware Examples for STM32H5xx Series Application based on NetXDuo: Nx_WebServer. https://www.st.com/en/development-tools/stm32cubeide.html

[8] Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout: Support for TE0821 Modules in Vitis 2023.2, AI 3.5 SW, AI 3.0 DPUCZDX8G (Application note, with evaluation package, UTIA). Published for free public access from: https://zs.utia.cas.cz/index.php?ids=results&id=24_TE0821_AI_3_5

[9]
Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout, Raissa Likhonina: Support for TE0820 Modules in Vitis 2023.2, AI 3.5 SW, AI 3.0 DPUCZDX8V (Application note, with evaluation package, UTIA). Published for free public access from: https://zs.utia.cas.cz/index.php?ids=results&id=25_TE0820_AI_3_5

[10]
Jiří Kadlec, Zdeněk Pohl, Lukáš Kohout, Raissa Likhonina: Compilation of AI 3.0 models for Vitis 2023.2, AI 3.5 SW, AI 3.0 DPUCZDX8V. (Application note, with evaluation package, UTIA). Published for free public access from: https://zs.utia.cas.cz/index.php?ids=results&id=26_TE_AI_3_5