TE0807 StarterKit Vitis AI Tutorial

Overview

Table of contents

Key Features

- Xilinx 2021.2 tools: Vivado 2021.2.1
- Vitis AI 2.0
- Vitis custom extensible platform
- Vector addition
- ResNet50 on DPU

Requirements

Type	Name	Version	Note
HW	te0807 Module	--	--
HW	TEB0808 Carrier	--	--
Diverse Cable	USB, Power...	--	--
Virtual Maschine	Oracle, VMware, MS WSL	--	optional
OS	Linux	Xilinx Supported OS	running on VM or native
Reference Design	TE0807-StarterKit-vivado_2021.2-build_18_20221017093227.zip	build 18 or higher to match Vivado 2021.2.1	Tutorial was created and tested with: TE0807-StarterKit-vivado_2021.2-build_18_20221017093227.zip module 52: TE0807-03-7DE21-A device: xczu7ev-fbvb900-1-e DDR4 :4GB
SW	Vitis	2021.2	--
SW	Vivado	2021.2.1	Vivado patch to 2021.2.1 is required by reference design package build 18 or higher
SW	Petalinux	2021.2	--
SW	Putty	--	--
Repo	Vitis-AI	2.0	https://github.com/Xilinx/Vitis-AI/tree/2.0

Prepare Development Environment

Virtual Machine

On Win10 Pro PC, you can use:

- Oracle VirtualBox 6.1
  https://www.virtualbox.org/
- VMware Workstation 16 Player
  https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
- Microsoft WSL. See Trenz installation guide for WSL
  https://wiki.trenz-electronic.de/display/PD/Xilinx+Tools+and+Win10+WSL

The presented extendible platform has been created on: Windows 10 Pro, ver. 21H2 OS build 19044.1889, VMware Workstation 16 Player (Version 16.2.4 build-20089737), Ubuntu 20.04 LTS Desktop 64-bit PC (AMD64)
https://linuxconfig.org/Ubuntu-20-04-download

Vitis/Vivado 2021.2 and creation of the extendible platform from ZIP archive has been also tested on:
Windows 11 Pro PC (upgrade from Windows 10 Pro, ver. 21H2 OS build 19044.1889)
VMware Workstation 16 Player (Version 16.2.4 build-20089737),
Ubuntu 20.04 LTS Desktop 64-bit PC (AMD64).
https://linuxconfig.org/Ubuntu-20-04-download

Linux OS

Only supported OS are selected Linux distributions. You will need either native or virtual PC with Linux distribution.

Create new VM with Linux OS supported by Vitis 2021.2 tools.

Use English as OS language for your Linux System. Keyboard language can be any language.
Other languages may cause errors in PetaLinux build process.

Set Language

In Ubuntu 20.04, open terminal and type command:

```
$ locale
```

Language is OK, if the command response starts with:

```
LANG=en_US.UTF-8
```

Set Bash as Terminal in Ubuntu

In Ubuntu, set bash as terminal.

```
$ sudo dpkg-reconfigure dash shell
```

select: no

Use of bash shell is required by Xilinx tools.

> ⚠ The Ubuntu 20.04 LTS terminal (selected as default installation) is dash.

Install OpenCL Client Drivers

On Ubuntu, install OpenCL Installable Client Driver Loader by executing:

```
$ sudo apt-get install ocl-icd-libopencl1
$ sudo apt-get install opencl-headers
$ sudo apt-get install ocl-icd-opencl-dev
```

Software Installation

Vitis 2021.2

Download Vitis

Download the Vitis Tools installer from the link below https://www.xilinx.com/support/download.html

Install Vitis

If Vitis 2021.2 is not installed, follow installation steps described in:

https://docs.xilinx.com/r/en-US/ug1393-vitis-application-acceleration/Vitis-Software-Platform-Installation

After a successful installation of the Vitis 2021.2 and Vivado 2021.2 in /tools directory, a confirmation message is displayed, with a prompt to run the installLibs.sh script.

Script location:
/tools/Vitis/2021.2/scripts/installLibs.sh

In Ubuntu terminal, change directory to /tools/Vitis/2021.2/script and run the script using sudo privileges:

```
$ sudo installLibs.sh
```

The command installs a number of necessary packages for the Vitis 2021.2 tools based on the actual OS version of your Ubuntu system.

Install y2k22_patch-1.2 to Vitis

If not applied before, apply the Xilinx y2k22_patch-1.2 to Vitis 2021.2 https://support.xilinx.com/s/article/76960?language=en_US

Install License Supporting Vivado

In Ubuntu terminal, source paths to Vivado tools by executing

```
$ source /tools/Xilinx/Vitis/2021.2/settings64.sh
```

Execute Vivado License Manager:

```
$ vlm
```

From vlm, login to your Xilinx account by an www browser.

In www browser, specify Vitis 2021.2 license. Select Linux target.

Download xilinx license file and copy it into the directory of your choice.
~/License/vitis_2021_2/Xilinx.lic

In vlm, select Load License -> Copy License

Putty

The putty terminal can be used for Ethernet connected terminal. Putty supports keyboard, mouse and forwarding of X11 for Zynq Ultrascale+ applications designed for X11 desktop GUI.

In Ubuntu terminal, execute:

```
$ sudo apt install putty
```

To test the installation, execute putty application from Ubuntu terminal by:

```
$ putty &
```

Exit from putty.

Petalinux 2021.2

Download Petalinux

Download the PetaLinux Tools installer from the link below https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools.html

Install Required Libraries

Install Petalinux 2021.2. Follow guideline described in:
https://wiki.trenz-electronic.de/display/PD/PetaLinux+KICKstart#PetaLinuxKICKstart-PetaLinux2021.2

Before PetaLinux installation, check UG1144 chapter "PetaLinux Tools Installation Requirements" and install missing tool/libraries with help of script plnx-env-setup.sh attached to the Xilinx Answer Record 73296 - PetaLinux: How to install the required packages for the PetaLinux Build Host?
https://www.xilinx.com/support/answers/73296.html

Use this page to download script: plnx-env-setup.sh

The script detects whether the Host OS is a Ubuntu, RHEL, or CentOS Linux distribution and then automatically installs all of the required packages for the PetaLinux Build Host.

The script requires root privileges. The script does not install the PetaLinux Tools. Command to run the script:

```
$ sudo ./plnx-env-setup.sh
```

Perform update of your PetaLinux and additional installation libraries.

```
$ sudo apt-get update
$ sudo apt-get install iproute2 gawk python3 python build-essential gcc
git make net-tools libncurses5-dev tftpd zlib1g-dev libssl-dev flex bison
libselinux1 gnupg wget git-core diffstat chrpath socat xterm autoconf
libtool tar unzip texinfo zlib1g-dev gcc-multilib automake zlib1g:i386
screen pax gzip cpio python3-pip python3-pexpect xz-utils debianutils
iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3
-y
```

Install Petalinux

and follow the directions in the "Installing the PetaLinux Tool" section of (UG1144).
https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf

To install petalinux do not start from shared folder, copy installer into your home directory.

```
$ mkdir -p ~/petalinux/2021.2
```

Copy petalinux-v2021.2-final-installer.run into ~/petalinux/2021.2

```
$ ./petalinux-v2020.2-final-installer.run
```

Source environment

```
$ source ~/petalinux/2021.2/settings.sh
```

Prepare Reference Design for Extensible Custom Platform

Update Vivado Project for Extensible Platform

Trenz Electronic Scripts allows posibility change some setup via enviroment variables, which depends on the used OS and PC performace.

To improve performance on multicore CPU add global envirment on line 64:
export TE_RUNNING_JOBS=10

to **/etc/bash.bashrc** or local to **design_basic_settings.sh**

For othervariables see also:

Project Delivery - Xilinx devices#EnvironmentVariables

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

```
$ source /tools/Xilinx/Vitis/2021.2/settings64.sh
```

Download TE0807 StarterKit Linux Design file(see Reference Design download link on chapter Requirements) with pre-build files to

~/Downloads/TE0807-StarterKit-vivado_2021.2-build_18_20221017093227.zip

This TE0807 StarterKit ZIP file contains bring-up scripts for creation of Petalinux for range of modules in zipped directory named “StarterKit”.

Unzip the file to directory:
~/work/te0807_52_240

All supported modules are identified in file: **~/work/te0807_52_240/StarterKit/board_files/TE0807_board_files.csv**

We will select module 52 with name TE0807-03-7DE21-A , with device xczu7ev-fbvb900-1-e on TEBF0808 carrier board. We will use default clock 240 MHz.
That is why we name the package te0807_52_240 and proposed to unzip the TE0807 StarterKit Linux Design files into the directory:
~/work/te0807_52_240

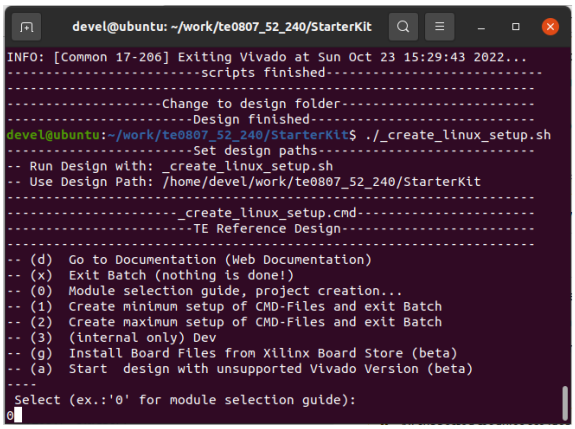
In Ubuntu terminal, change directory to the StarterKit directory:

```
$ cd ~/work/te0807_52_240/StarterKit
```

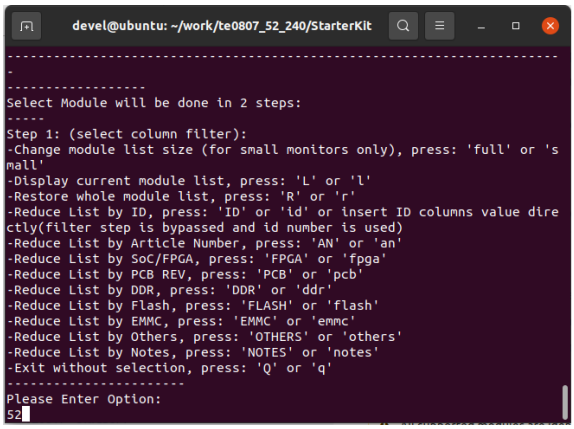
Setup the StarterKit directory files for a Linux host machine.
In Ubuntu terminal, execute:

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

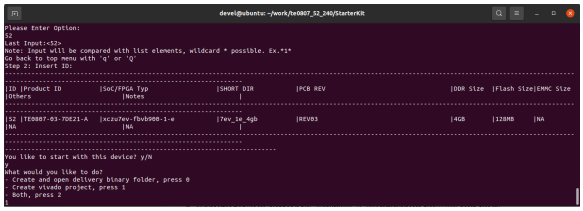
Select option (0) to open Selection Guide and press Enter



Select variant 52 from the selection guide, press enter and agree selection



Create Vivado Project with option 1



Vivado Project will be generated for the selected variant.

Selection Guide automatically modified ./design_basic_settings.sh with correct variant, so other provided bash files to recreate or open Vivado project again can be used later also.

In case of using selection guide, variant can be selected also manually:

Select option (2) to create maximum setup of CMD-Files and exit the script (by typing any key).

It moves main design bash scripts to the top of the StarterKit directory. Set these files as executable, from the Ubuntu terminal:

```
$ chmod ugo+rwx *.sh
```

In text editor, open file
~/work/te0807_52_240/StarterKit/design_basic_settings.sh

On line 63, change
export PARTNUMBER=LAST_ID
to
export PARTNUMBER=52

To improve performance on multicore CPU add on line 64:
export TE_RUNNING_JOBS=10

Vivado will be utilizing up to 10 parallel logical processor cores with this setup instead of the default of 2 parallel logical processor cores.

Save the modified file.

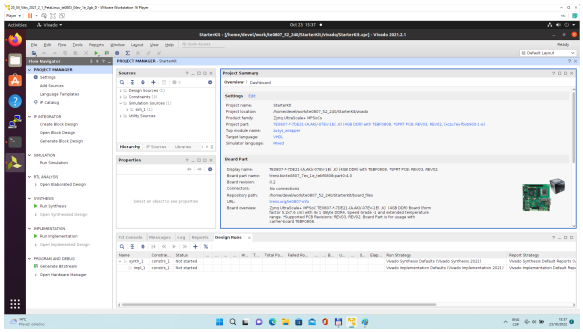
This modification will guide the Trenz TE0807 StarterKit Linux Design scripts to generate Vivado HW for the module 52 with name TE0807-03-7DE21-A , with device xczu7ev-fbvb900-1-e on TEBF0808 carrier board.

In Ubuntu terminal, change directory to
~/work/te0807_52_240/StarterKit

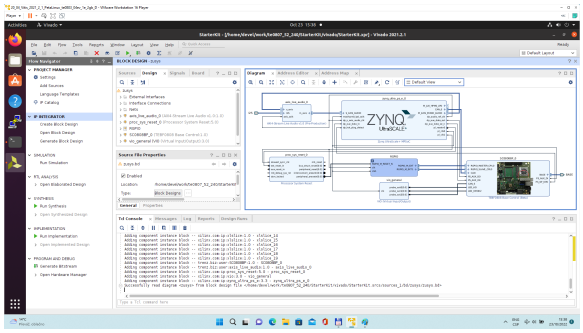
The Vivado will be opened and Trenz Electronic HW project for the TE0807 StarterKit Linux Design, part 52 will be generated by running this script:

```
$ ./vivado_create_project_guimode.sh
```

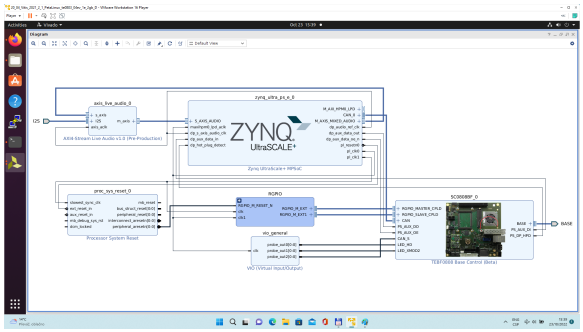
The Vivado will be opened and Trenz Electronic HW project for the TE0807 StarterKit Linux Design, part 52 will be generated.



In Vivado window Sources, click on zusys_wrapper and next on zusys.bd to open the HW diagram in IP integrator:



It is possible to display diagram in separate window by clicking on float icon in upper right corner of the diagram.



Zynq Ultrascale+ block is configured for the Trenz TE0807 StarterKit Linux Design on the TEBF0808 carrier board.

This is starting point for the standard PetaLinux system supported by Trenz with steps for generation of the PetaLinux system. Parameters of this system and compilation steps are described on Trenz Wiki pages:
https://wiki.trenz-electronic.de/display/PD/TE0807+StarterKit

Follow steps described in these wiki pages if you would like to create fixed, not extensible Vitis platform.

The Extensible Vitis platform generation steps are described in next paragraphs.

Create Extensible Vitis platform

To implement hardware this tutorial offers two alternatives: Fast Track or Manual Track:

- Choose Fast Track to use TCL script to do the same modifications as in manual track case automatically,
- Select Manual Track path if you want to see all required hardware modifications required for custom platform.

Fast Track

Block Design of the Vivado project must be opened for this step. Copy following TCL Code to the TCL comand console of Vivado:

TCL Script to prepare Extensible Vitits Platform

```
#activate extensible platform
set_property platform.extensible true [current_project]
save_bd_design

#set_property PFM_NAME "xilinx:te0808_15eg_1e_TEBF0808:zusys:0.0"
[get_files zusys.bd]
set_property PFM_NAME [string map {part0 zusys} [string map {trenz.biz
trenz} [current_board_part]]] [get_files zusys.bd]
set_property platform.design_intent.embedded {true} [current_project]
set_property platform.design_intent.datacenter {false} [current_project]
set_property platform.design_intent.server_managed {false}
[current_project]
set_property platform.design_intent.external_host {false} [current_project]
set_property platform.default_output_type {sd_card} [current_project]
set_property platform.uses_pr {false} [current_project]

save_bd_design
#set_property pfm_name {xilinx:te0808_15eg_1e_TEBF0808:zusys:0.0}
[get_files -all {zusys.bd}]
#set_property platform.name {zusys} [current_project]

#add clocking wizard
startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:clk_wiz:6.0 clk_wiz_0
endgroup

#clocking wizard config
set_property -dict [list CONFIG.CLKOUT2_USED {true} CONFIG.CLKOUT3_USED
{true} CONFIG.CLKOUT4_USED {true} CONFIG.CLKOUT2_REQUESTED_OUT_FREQ
{200.000} CONFIG.CLKOUT3_REQUESTED_OUT_FREQ {400.000} CONFIG.
CLKOUT4_REQUESTED_OUT_FREQ {240.000} CONFIG.RESET_TYPE {ACTIVE_LOW} CONFIG.
MMCM_CLKOUT1_DIVIDE {6} CONFIG.MMCM_CLKOUT2_DIVIDE {3} CONFIG.
MMCM_CLKOUT3_DIVIDE {5} CONFIG.NUM_OUT_CLKS {4} CONFIG.RESET_PORT {resetn}
CONFIG.CLKOUT2_JITTER {102.086} CONFIG.CLKOUT2_PHASE_ERROR {87.180} CONFIG.
CLKOUT3_JITTER {90.074} CONFIG.CLKOUT3_PHASE_ERROR {87.180} CONFIG.
CLKOUT4_JITTER {98.767} CONFIG.CLKOUT4_PHASE_ERROR {87.180}] [get_bd_cells
clk_wiz_0]

#connect clocking wizard inputs
connect_bd_net [get_bd_pins clk_wiz_0/resetn] [get_bd_pins
zynq_ultra_ps_e_0/pl_resetn0]
connect_bd_net [get_bd_pins clk_wiz_0/clk_in1] [get_bd_pins
zynq_ultra_ps_e_0/pl_clk0]

#add reset cores
startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:proc_sys_reset:5.0
proc_sys_reset_1
create_bd_cell -type ip -vlnv xilinx.com:ip:proc_sys_reset:5.0
proc_sys_reset_2
create_bd_cell -type ip -vlnv xilinx.com:ip:proc_sys_reset:5.0
proc_sys_reset_3
create_bd_cell -type ip -vlnv xilinx.com:ip:proc_sys_reset:5.0
proc_sys_reset_4
endgroup

#connect reset cores
connect_bd_net [get_bd_pins clk_wiz_0/clk_out1] [get_bd_pins
proc_sys_reset_1/slowest_sync_clk]
connect_bd_net [get_bd_pins clk_wiz_0/clk_out2] [get_bd_pins
proc_sys_reset_2/slowest_sync_clk]
```

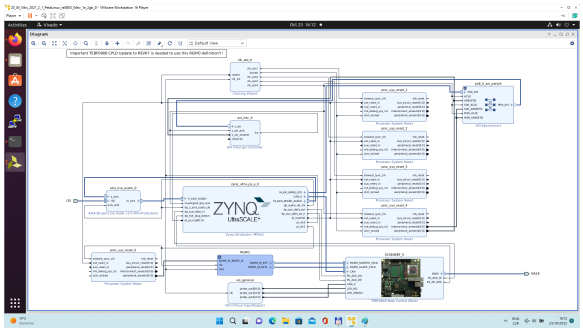
```
connect_bd_net [get_bd_pins clk_wiz_0/clk_out3] [get_bd_pins
proc_sys_reset_3/slowest_sync_clk]
connect_bd_net [get_bd_pins clk_wiz_0/clk_out4] [get_bd_pins
proc_sys_reset_4/slowest_sync_clk]
connect_bd_net [get_bd_pins clk_wiz_0/locked] [get_bd_pins proc_sys_reset_1
/dcm_locked]
connect_bd_net [get_bd_pins clk_wiz_0/locked] [get_bd_pins proc_sys_reset_2
/dcm_locked]
connect_bd_net [get_bd_pins proc_sys_reset_3/dcm_locked] [get_bd_pins
clk_wiz_0/locked]
connect_bd_net [get_bd_pins proc_sys_reset_4/dcm_locked] [get_bd_pins
clk_wiz_0/locked]
connect_bd_net [get_bd_pins proc_sys_reset_1/ext_reset_in] [get_bd_pins
zynq_ultra_ps_e_0/pl_resetn0]
connect_bd_net [get_bd_pins proc_sys_reset_2/ext_reset_in] [get_bd_pins
zynq_ultra_ps_e_0/pl_resetn0]
connect_bd_net [get_bd_pins proc_sys_reset_3/ext_reset_in] [get_bd_pins
zynq_ultra_ps_e_0/pl_resetn0]
connect_bd_net [get_bd_pins proc_sys_reset_4/ext_reset_in] [get_bd_pins
zynq_ultra_ps_e_0/pl_resetn0]

# add clocks to platform
set_property PFM.CLOCK {clk_out1 {id "1" is_default "false" proc_sys_reset
"/proc_sys_reset_1" status "fixed" freq_hz "100000000"} clk_out2 {id "2"
is_default "false" proc_sys_reset "/proc_sys_reset_2" status "fixed"
freq_hz "200000000"} clk_out3 {id "3" is_default "false" proc_sys_reset "
/proc_sys_reset_3" status "fixed" freq_hz "400000000"} clk_out4 {id "4"
is_default "true" proc_sys_reset "/proc_sys_reset_4" status "fixed"
freq_hz "240000000"}} [get_bd_cells /clk_wiz_0]

# prepare LPD interface for 240MHz for interrupt controller
disconnect_bd_net /zynq_ultra_ps_e_0_pl_clk1 [get_bd_pins zynq_ultra_ps_e_0
/maxihpm0_lpd_aclk]
connect_bd_net [get_bd_pins clk_wiz_0/clk_out4] [get_bd_pins
zynq_ultra_ps_e_0/maxihpm0_lpd_aclk]

#add interrupt core
startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:axi_intc:4.1 axi_intc_0
endgroup

#config interrupt core
set_property -dict [list CONFIG.C_KIND_OF_INTR.VALUE_SRC USER]
[get_bd_cells axi_intc_0]
set_property -dict [list CONFIG.C_KIND_OF_INTR {0x00000000} CONFIG.
C_IRQ_CONNECTION {1}] [get_bd_cells axi_intc_0]

#connect interrupt core
connect_bd_net [get_bd_pins axi_intc_0/s_axi_aclk] [get_bd_pins clk_wiz_0
/clk_out4]
connect_bd_net [get_bd_pins axi_intc_0/s_axi_aresetn] [get_bd_pins
proc_sys_reset_4/peripheral_aresetn]

startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:axi_interconnect:2.1
axi_interconnect_0
endgroup
set_property -dict [list CONFIG.NUM_MI {1}] [get_bd_cells
axi_interconnect_0]
connect_bd_net [get_bd_pins axi_interconnect_0/ACLK] [get_bd_pins clk_wiz_0
/clk_out4]
```

```
connect_bd_net [get_bd_pins axi_interconnect_0/ARESETN] [get_bd_pins
proc_sys_reset_4/peripheral_aresetn]
connect_bd_net [get_bd_pins axi_interconnect_0/S00_ARESETN] [get_bd_pins
proc_sys_reset_4/interconnect_aresetn]
connect_bd_net [get_bd_pins axi_interconnect_0/M00_ARESETN] [get_bd_pins
proc_sys_reset_4/interconnect_aresetn]
connect_bd_net [get_bd_pins axi_interconnect_0/S00_ACLK] [get_bd_pins
clk_wiz_0/clk_out4]
connect_bd_net [get_bd_pins axi_interconnect_0/M00_ACLK] [get_bd_pins
clk_wiz_0/clk_out4]

connect_bd_intf_net [get_bd_intf_pins zynq_ultra_ps_e_0/M_AXI_HPM0_LPD] -
boundary_type upper [get_bd_intf_pins axi_interconnect_0/S00_AXI]
connect_bd_intf_net -boundary_type upper [get_bd_intf_pins
axi_interconnect_0/M00_AXI] [get_bd_intf_pins axi_intc_0/s_axi]

#rename interconnect
set_property name ps8_0_axi_periph [get_bd_cells axi_interconnect_0]

#add zynqUS interrupt inputs and connect intr IP core
startgroup
set_property -dict [list CONFIG.PSU__USE__IRQ0 {1}] [get_bd_cells
zynq_ultra_ps_e_0]
endgroup
connect_bd_net [get_bd_pins axi_intc_0/irq] [get_bd_pins zynq_ultra_ps_e_0
/pl_ps_irq0]

# add interrputs to platform
set_property PFM.IRQ {intr { id 0 range 32 }} [get_bd_cells /axi_intc_0]

# add axi buses to platform
set_property PFM.AXI_PORT {M_AXI_HPM0_FPD {memport "M_AXI_GP" sptag "GP0"
memory "" is_range "false"} M_AXI_HPM1_FPD {memport "M_AXI_GP" sptag "GP1"
memory "" is_range "false"} S_AXI_HPC0_FPD {memport "S_AXI_HP" sptag
"HPC0" memory "" is_range "false"} S_AXI_HPC1_FPD {memport "S_AXI_HP"
sptag "HPC1" memory "" is_range "false"} S_AXI_HP0_FPD {memport "S_AXI_HP"
sptag "HP0" memory "" is_range "false"} S_AXI_HP1_FPD {memport "S_AXI_HP"
sptag "HP1" memory "" is_range "false"} S_AXI_HP2_FPD {memport "S_AXI_HP"
sptag "HP2" memory "" is_range "false"} S_AXI_HP3_FPD {memport "S_AXI_HP"
sptag "HP3" memory "" is_range "false"}} [get_bd_cells /zynq_ultra_ps_e_0]

#add interconnect ports to platform
set_property PFM.AXI_PORT {M01_AXI {memport "M_AXI_GP" sptag "" memory ""
is_range "false"} M02_AXI {memport "M_AXI_GP" sptag "" memory "" is_range
"false"} M03_AXI {memport "M_AXI_GP" sptag "" memory "" is_range "false"}
M04_AXI {memport "M_AXI_GP" sptag "" memory "" is_range "false"} M05_AXI
{memport "M_AXI_GP" sptag "" memory "" is_range "false"} M06_AXI {memport
"M_AXI_GP" sptag "" memory "" is_range "false"} M07_AXI {memport
"M_AXI_GP" sptag "" memory "" is_range "false"}} [get_bd_cells
/ps8_0_axi_periph]

# add addresses to unmapped peripherals
assign_bd_address

#save
save_bd_design

#save project XPR name
global proj_xpr
set proj_xpr [current_project]
append proj_xpr .xpr
```

```
#close project
close_project

# reopen project
open_project $proj_xpr

# open block design
open_bd_design [current_project].srcs/sources_1/bd/zusys/zusys.bd

#validate
#validate_bd_design
```

This script modifies the Initial platform Block design into the Extensible platform Block design and also defines define Platform Setup configuration.

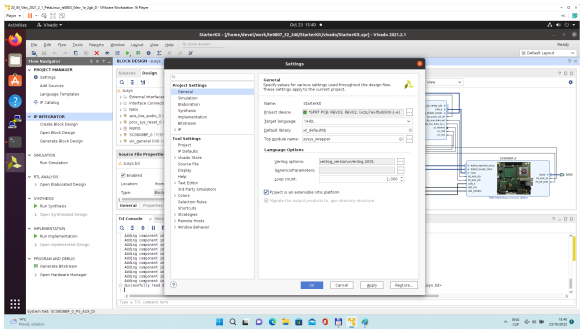
In Vivado, open the design explorer and Platform description.
The fast track result is identical to the manually performed modifications described in next sections. In Vivado, save block design by clicking on icon "**Save Block Design**".



Continue the design path with Validate Design.

Manual Track

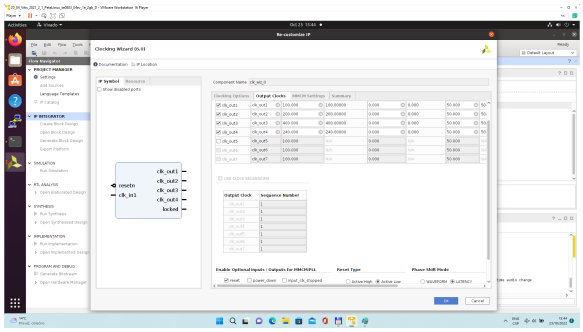
In Vivado project, click in **Flow Navigator** on **Settings**. In opened Settings window, select **General** in **Project Settings**, select **Project is an extensible Vitis platform**. Click on **OK**.



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

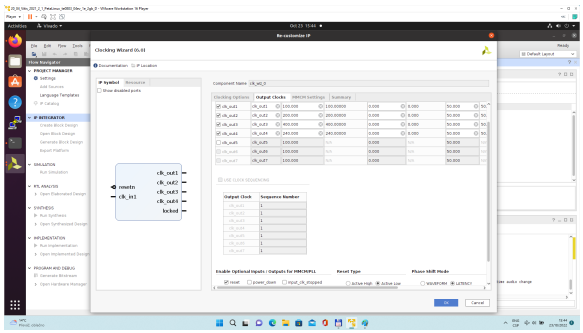
Add multiple clocks and processor system reset IPs
In IP Integrator Diagram Window, right click, select **Add IP** and add **Clocking Wizard** IP **clk_wiz_0**. Double-click on the IP to Re-customize IP window. Select Output Clocks panel. Select four clocks with frequency 100, 200, 400 and 240 MHz.
100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

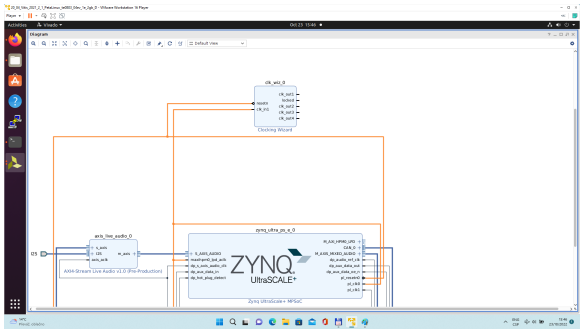


Clik on OK to close the Re-customize IP window.

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.

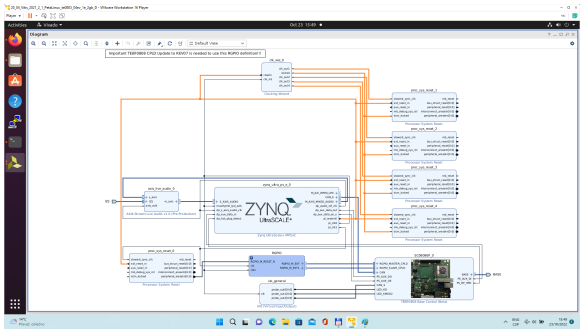


Open **Platform Setup** window of IP Integrator to define Clocks. In **Settings**, select **Clock**.

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**

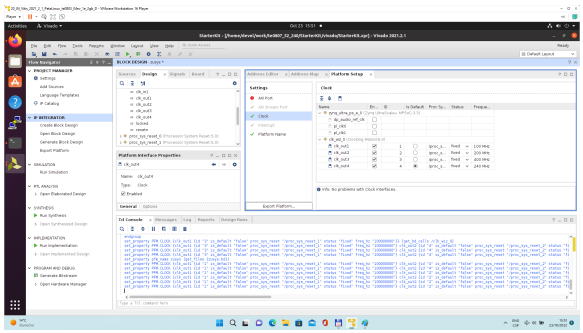
In “Is Default” column, select **clk_out4** (with ID=4) as the default clock. One and only one clock must be selected as default clock.



Disconnect input pin **maxihpm0_lpd_aclk** of **zynq_ultra_ps_e_0** from the 100 MHz clock net. This net is driven by **clock output pl_clk0** of **zynq_ultra_ps_e_0**.

Connect input pin **maxihpm0_lpd_aclk** of **zynq_ultra_ps_e_0** to the 240 MHz **clk_out4** of **clk_wiz_0** IP block.

These two modifications are made to support the axi-lite interface of an interrupt controller operating at 240 MHz clock, identical with the default extendable platform clock.



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.

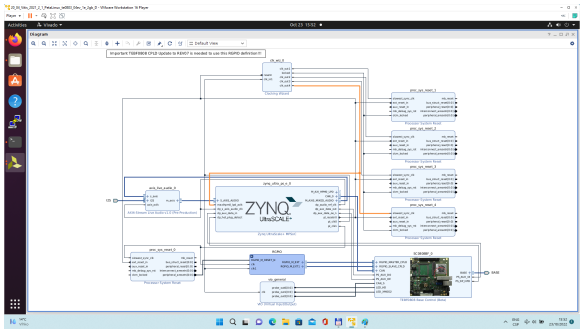
In “Processor Interrupt Type and Connection” section select the “Interrupt Output Connection” from “Bus” to “Single”.

In “Peripherial Interrupt Type” section, change the “Interrupts Types Edge or Level” from AUTO to MANUAL. Change the corresponding value from 0xFFFFFFFF to 0x00000000.

Click on OK to accept these changes.

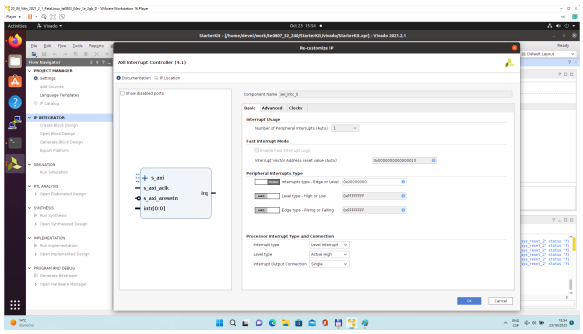
> ⚠ This re-configuration is manually setting all interrupts as level interrupts. With this setting, the PetaLinux automatically creates correct description of the interrupt controller in the device tree. The Vitis extensible flow generates HW IP blocks with level interrupts.

In case of user defined edge interrupts, the corresponding interrupt description will be added in an customised, interrupt controller description section of the user-defined device tree file ~/work/te0807_52_240/StarterKit/os/petalinux/project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi
For the default extensible te0807_52_240_pfm platform it is not needed.



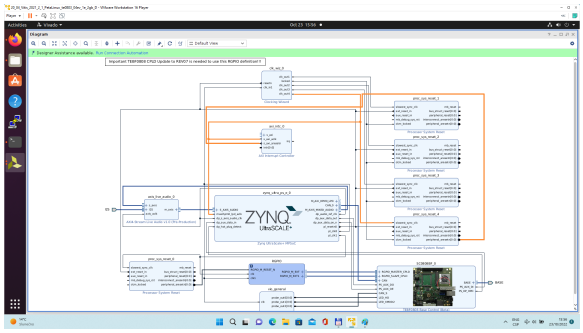
Connect interrupt controller clock input **s_axi_aclk** of **axi_intc_0** to clock output **dlk_out4** of **clk_wiz_0**. It is the default, 240 MHz clock of the extensible platform.

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.

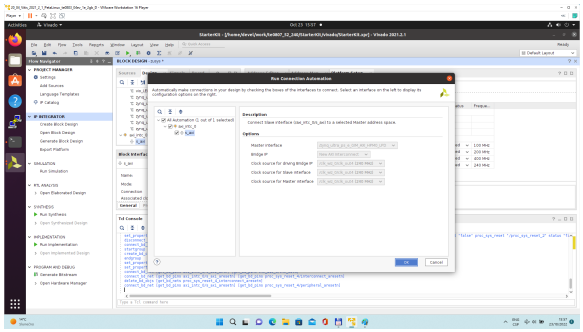


Use the Run Connection Automation wizard to connect the axi lite interface of interrupt controller **axi_intc_0** to **zynq_ultra_ps_e_0**. It is available in green line in top of the Diagram window.

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



New AXI interconnect **ps_8_axi_periph** is created and related connections are generated.



Vitis extensible design flow will be expanding the AXI interconnect **ps_8_axi_periph** for interfacing and configuration of registers of generated HW IP blocks with the default extensible platform clock 240 MHz.

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

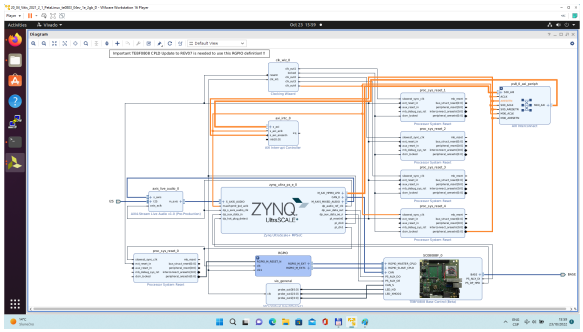
Disconnect input **S00_ARESETN** of **ps_8_axi_periph** from the network driven by output **peripherial_aresetn[0:0]** of **proc_sys_reset_4** block.

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

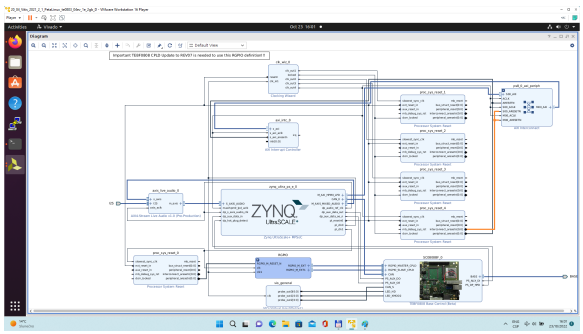
Disconnect input **M00_ARESETN** of **ps_8_axi_periph** block from the network driven by output **peripherial_aresetn[0:0]** of **proc_sys_reset_4** block.

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

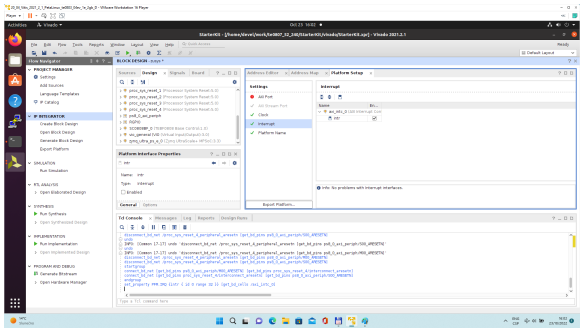
This modification will make the reset structure of the AXI interconnect **ps_8_axi_periph** block identical to the future extensions generated by the Vitis extensible design flow.



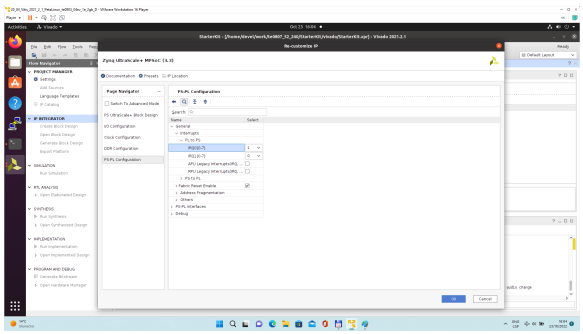
Double-click on **zynq_ultra_ps_e_0** to re-customize it by enabling of an interrupt input **pl_ps_irq0[0:0]**. Click OK.



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.



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

Select **M_AXI_HPM0_FPD** and **M_AXI_HPM1_FPD** in column “Enabled”.

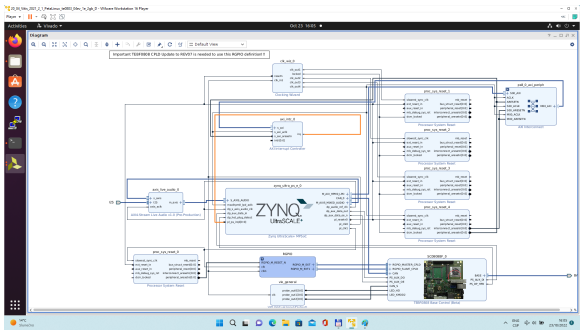
Select **S_AXI_HPC0_FPD** and **S_AXI_HPC1_FPD** in column “Enabled”.

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”.

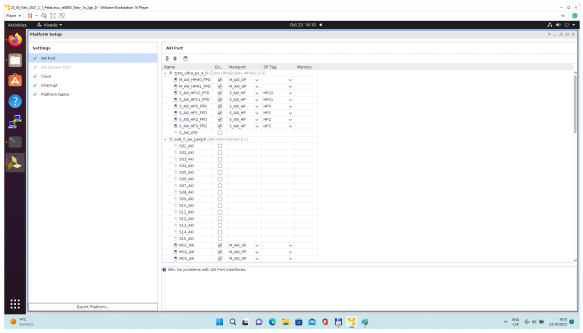
Select **S_AXI_HP0_FPD, S_AXI_HP1_FPD, S_AXI_HP2_FPD, S_AXI_HP3_FPD** in column “Enabled”.

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 **ps8_0_axi_periph**:

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



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

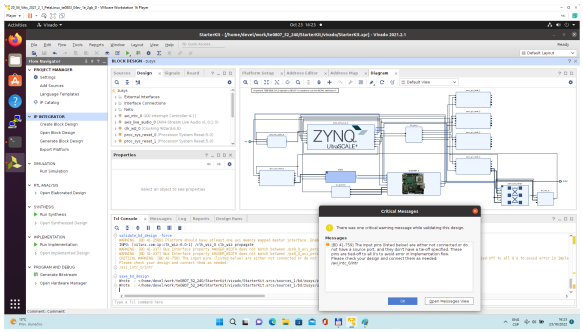
In Vivado, save block design by clicking on icon “**Save Block Design**”.

Continue the design path with Validate Design.

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 connect this input to interrupt outputs from generated HW IPs.

Click OK.

> Known Issue: Sometimes an error in validation process may occur reporting **create_pfm** function is not known. Workaroud is to close Vivado tool and reopen again to correclty load platform export API.

> 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/te0807_52_240/StarterKit/vivado/zusys.pdf**

Compile Created HW and Custom SW with Trenz Scripts

In Vivado Tcl Console, type following script and execute it by Enter. It will take some time to compile HW. HW design and to export the corresponding standard XSA package with included bitstream.

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

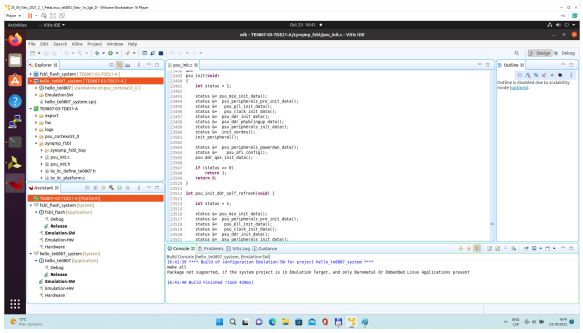
An archive for standard non-extensible system is created:
~/work/te0807_52_240/StarterKit/vivado/StarterKit_7ev_1e_4gb.xsa

In Vivado Tcl Console, type the following script and execute it by Enter. It will take some time to compile.

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

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

Close the Vitis “Welcome” page.
Compile the two included SW projects.
Standalone custom Vitis platform **TE0807-03-7DE21-A** has been created and compiled.



The **TE0807-03-7DE21-A** Vitis platform includes Trenz Electronic custom first stage boot loader in folder **zynqmp_fsbl**. It includes SW extension specific for the Trenz module initialisation.

This custom zynqmp_fsbl project has been compiled into executable file fsbl.elf. It is located in: **~/work /te0807_52_240/StarterKit/prebuilt/software/7ev_1e_4gb/fsbl.elf**

This customised first stage boot loader is needed for the Vitis extensible platform.
We have used the standard Trenz scripts to generate it for next use in the extensible platform.

Exit the opened Vitis 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. Click **OK**.

The exported Vitis Extensible Hardware platform named **StarterKit_7ev_1e_4gb.xsa** can be found in **viv ado** folder.

Copy Created Custom First Stage Boot Loader

Up to now, StarterKit directory has been used for all development.
~/work/te0807_52_240/StarterKit

Create new folders:
~/work/te0807_52_240/StarterKit_pfm/pfm/boot
~/work/te0807_52_240/StarterKit_pfm/pfm/sd_dir

Copy the recently created custom first stage boot loader executable file from
~/work/te0807_52_240/StarterKit/prebuilt/software/7ev_1e_4gb/fsbl.elf
to
~/work/te0807_52_240/StarterKit_pfm/pfm/boot/fsbl.elf

Building Platform OS and SDK

Configuration of the Default Trenz Petalinux for the Vitis Extensible Platform

Change directory to the default Trenz Petalinux folder
~/work/te0807_52_240/StarterKit/os/petalinux

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

```
$ source /tools/Xilinx/Vitis/2021.2/settings64.sh
$ source ~/petalinux/2021.2/settings.sh
```

Configure petalinux with the StarterKit_7ev_1e_4gb.xsa for the extensible design flow by executing:

```
$ petalinux-config --get-hw-description=~/work/te0807_52_240/StarterKit
/vivado
```

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

Customize Root File System, Kernel, Device Tree and U-boot

In text editor, append definition of 32 interrupts by this text:

```
&amba {         zyxclmm_drm {
                 compatible = "xlnx,zocl";
                 status = "okay";
                 reg = <0x0 0xA0000000 0x0 0x10000>;
                 interrupt-parent = <&axi_intc_0>;
                 interrupts = <0  4>, <1  4>, <2  4>, <3  4>,
                              <4  4>, <5  4>, <6  4>, <7  4>,
                              <8  4>, <9  4>, <10 4>, <11 4>,
                              <12 4>, <13 4>, <14 4>, <15 4>,
                              <16 4>, <17 4>, <18 4>, <19 4>,
                              <20 4>, <21 4>, <22 4>, <23 4>,
                              <24 4>, <25 4>, <26 4>, <27 4>,
                              <28 4>, <29 4>, <30 4>, <31 4>;
        };
};
```

to the **system-user.dtsi** file located in folder:
~/work/te0807_52_240/StarterKit/os/petalinux/project-spec/meta-user/recipes-bsp/device-tree/files/

Download the Vitis-AI 2.0 repository.
In browser, open page:

https://github.com/Xilinx/Vitis-AI/tree/2.0

Clik on green Code button and download Vitis-AI-2.0.zip file.
Unzip **Vitis-AI-2.0.zip** file to directory **~/Downloads/Vitis-AI**.

Copy **~/Downloads/Vitis-AI** to **~/vitis_ai_2_0**

Delete **Vitis-AI-2.0.zip**, delete **~/Downloads/Vitis-AI** , clean trash.

The directory **~/vitis_ai_2_0** contains the Vitis-AI 2.0 framework, now.

To install the Vitis-AI 2.0 version of shared libraries into rootfs (when generating system image by PetaLinux) we have to copy recepies recipes-vitis-ai to the Petalinux project :

Copy
~/vitis_ai_2_0/tools/Vitis-AI-Recipes/recipes-vitis-ai

to
~/work/te0807_52_240/StarterKit/os/petalinux/project-spec/meta-user/

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_vitis-ai-library
CONFIG_vitis-ai-library-dev
CONFIG_vitis-ai-library-dbg
```

to the **user-rootfsconfig** file:
~/work/te0807_52_240/StarterKit/os/petalinux/project-spec/meta-user/conf/user-rootfsconfig

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 2.0 recepies for installation of the correspoding Vitis-AI 2.0 libraries into rootfs of PetaLinux.

Enable all required packages in Petalinux configuration, from the Ubuntu terminal:

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

Select all **user packages** by typing “y”. All packages will have 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**.

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** and **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** option
Click **OK**, **Exit** twice and select **Yes** to save the changes.

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.

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 te0807 modules on TEBF0808 carrier:
/dev/mmcblk1p2

Exit and **Yes** to save changes.

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.

Build PetaLinux Image

In terminal, build the PetaLinux project by executing:

```
$ petalinux-build
```

The PetaLinux image files will be generated in the directory:
~/work/te0807_52_240/StarterKit/os/petalinux/images/linux

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

Create Petalinux SDK

The SDK is used by Vitis 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/te0807_52_240/StarterKit/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.

Copy Files for Extensible Platform

Copy these four files:

Files	From	To
bl31.elf pmufw.elf system.dtb u-boot-tb.elf	~/work/te0807_52_240/StarterKit/os/petalinux/images/linux	~/work/te0807_52_240/StarterKit_pfm/pfm/boot

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

The directory
~/work/te0807_52_240/StarterKit_pfm/pfm/boot
contains these five files:

1. bl31.elf
2. fsbl.elf
3. pmufw.elf
4. system.dtb
5. u-boot.elf

Copy files:

Files	From	To
boot.scr system.dtb	~/work/te0807_52_240/StarterKit/os/petalinux/images/linux	~/work/te0807_52_240/StarterKit_pfm/pfm/sd_dir

Copy file:

File	From	To
init.sh	~/work/te0807_52_240/StarterKit/misc/sd	~/work/te0807_52_240/StarterKit_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
```

Create Extensible Platform zip File

Create new directory tree:
~/work/te0807_52_240_move/StarterKit/os/petalinux/images
~/work/te0807_52_240_move/StarterKit/Vivado
~/work/te0807_52_240_move/StarterKit_pfm/pfm/boot
~/work/te0807_52_240_move/StarterKit_pfm/pfm/sd_dir

Copy all files from the directory:

Files	Source	Destination
all	~/work/te0807_52_240/StarterKit/os/petalinux/images	~/work/te0807_52_240_move/StarterKit/os/petalinux/images
all	~/work/te0807_52_240/StarterKit_pfm/pfm/boot	~/work/te0807_52_240_move/StarterKit_pfm/pfm/boot
all	~/work/te0807_52_240/StarterKit_pfm/pfm/sd_dir	~/work/te0807_52_240_move/StarterKit_pfm/pfm/sd_dir
StarterKit_7ev_1e_4gb.xsa	~/work/te0807_52_240/StarterKit/Vivado/StarterKit_7ev_1e_4gb.xsa	~/work/te0807_52_240_move/StarterKit/Vivado/StarterKit_7ev_1e_4gb.xsa

Zip the directory
~/work/te0807_52_240_move
into ZIP archive:
~/work/te0807_52_240_move.zip

The archive te0807_52_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 only for the initially selected module (52).
This is the te0807 HW module with xczu7ev-fbvb900-1-e device with 4 GB memory.
The extensible Vitis platform will have the default clock 240 MHz.

Move the **te0807_52_240_move.zip** file to an PC disk drive. Delete:
~/work/te0807_52_240_move
~/work/te0807_52_240_move.zip
Clean the Ubuntu Trash.

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 (52) are present in the archive: **te0807_52_240_move.zip**
In this case, unzip the archive to the directory:
~/work/te0807_52_240_move
and copy all content of directories to
~/work/te0807_52_240
Delete the te0807_52_240_move.zip ZIP file and the **~/work/te0807_52_240_move** directory to save filesystem space.

In Ubuntu terminal, change the working directory to:
~/work/te0807_52_240/StarterKit/os/petalinux/images/linux

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

```
$ source /tools/Xilinx/Vitis/2021.2/settings64.sh
```

In Ubuntu terminal, execute script

```
$ ./sdk.sh -d ~/work/te0807_52_240/StarterKit_pfm
```

SYSROOT directories and files for PC and for Zynq Ultrascale+ will be created in:
~/work/te0807_52_240/StarterKit_pfm/sysroots/x86_64-petalinux-linux
~/work/te0807_52_240/StarterKit_pfm/sysroots/cortexa72-cortexa53-xilinx-linux

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

Generation of Extensible Platform for Vitis

In Ubuntu terminal, change the working directory to:
~/work/te0807_52_240/StarterKit_pfm

Start Vitis tool by executing

```
$ vitis &
```

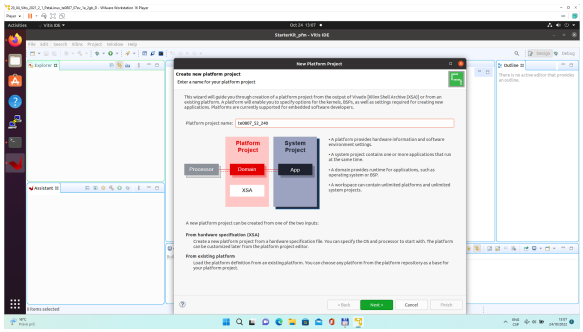
In Vitis “Launcher”, set the workspace for the extensible platform compilation:
~/work/te0807_52_240/StarterKit_pfm

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: **te0807_52_240_pfm**. Click Next.



Choose for hardware specification for the platform file:
~/work/te0807_52_240/StarterKit/vivado/StarterKit_7ev_1e_4gb.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 **te0807_52_240_pfm** is opened.

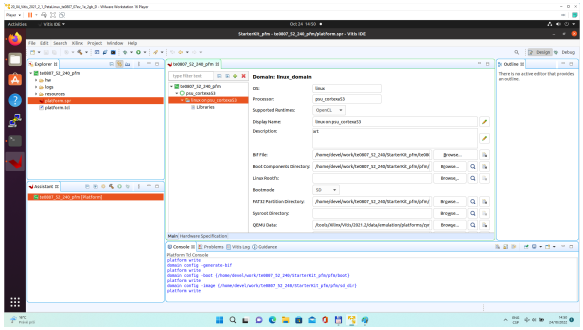
Click on **linux on psu_cortex53** to open window **Domain: linux_domain**

In “Description”: write **xrt**

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

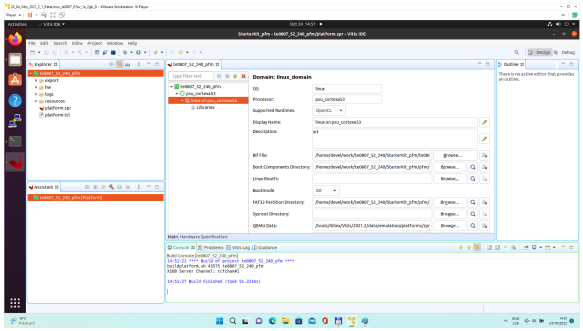
In “Boot Components Directory” select:
~/work/te0807_52_240/StarterKit_pfm/pfm/boot

In “FAT32 Partition Directory” select:
~/work/te0807_52_240/StarterKit_pfm/pfm/sd_dir



In Vitis IDE “Explorer” section, click on **te0807_52_240_pfm** to highlight it.

Right-click on the highlighted **te0807_52_240_pfm** and select **build project** in the open submenu.
Platform is compiled in few seconds.
Close Vitis tool by selection: **File -> Exit**.



Vits extensible platform **te0807_52_240_pfm** has been created in the directory:
~/work/te0807_52_240/StarterKit_pfm/te0807_52_240_pfm/export/te0807_52_240_pfm

Platform Usage

Test 1: Read Platform Info

With Vitis environment setup, platforminfo tool can report XPFM platform information.

```
platforminfo ~/work/te0807_52_240/StarterKit_pfm/te0807_52_240_pfm/export
/te0807_52_240_pfm/te0807_52_240_pfm.xpfm
```

Detailed listing from platforminfo utility

```
====================
Basic Platform Information
==========================
Platform:           te0807_52_240_pfm
File:               /home/devel/work/te0807_52_240/StarterKit_pfm
/te0807_52_240_pfm/export/te0807_52_240_pfm/te0807_52_240_pfm.xpfm
Description:
te0807_52_240_pfm


====================================
Hardware Platform (Shell) Information
====================================
Vendor:                           trenz
Board:                            zusys
Name:                             zusys
Version:                          4.0
Generated Version:                2021.2.1
Hardware:                         1
Software Emulation:               1
Hardware Emulation:               1
Hardware Emulation Platform:      0
FPGA Family:                      zynquplus
FPGA Device:                      xczu7ev
Board Vendor:                     trenz.biz
Board Name:                       trenz.biz:te0807_7ev_1e_TEBF0808:4.0
Board Part:                       xczu7ev-fbvb900-1-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:  te0807_52_240_pfm
System Configurations:
  System Config Name:                     te0807_52_240_pfm
```

```
  System Config Description:               te0807_52_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:            te0807_52_240_pfm/boot/linux.bif
    Boot Image Data:           te0807_52_240_pfm/linux_domain/image
    Boot Image Boot Mode:      sd
    Boot Image RootFileSystem:
    Boot Image Mount Path:     /mnt
    Boot Image Read Me:        te0807_52_240_pfm/boot/generic.readme
    Boot Image QEMU Args:      te0807_52_240_pfm/qemu/pmu_args.txt:
te0807_52_240_pfm/qemu/qemu_args.txt
    Boot Image QEMU Boot:
    Boot Image QEMU Dev Tree:
Supported Runtimes:
  Runtime: OpenCL
```

Test 2: Run Vector Addition Example

Create new directory **StarterKit_test_vadd** to test Vitis extendable flow example “vector addition”
~/work/te0807_52_240/StarterKit_test_vadd

Current directory structure:
~/work/te0807_52_240/StarterKit
~/work/te0807_52_240/StarterKit_pfm
~/work/te0807_52_240/StarterKit_test_vadd

Change working directory:

```
$cd ~/work/te0807_52_240/StarterKit_test_vadd
```

In Ubuntu terminal, start Vitis by:

```
$ vitis &
```

In Vitis IDE Launcher, select your working directory
~/work/te0807_52_240/StarterKit_test_vadd
Click on **Launch** to start Vitis.

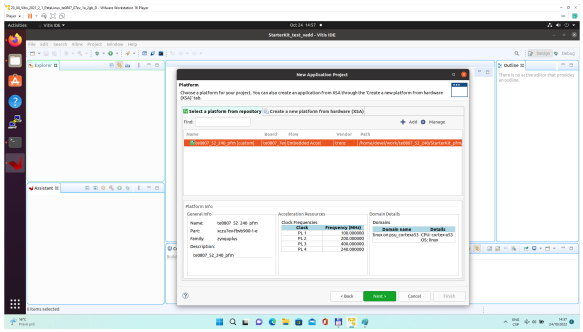
Select **File -> New -> Application project**. Click **Next**.

Skip welcome page if shown.

Click on “**+ Add**” icon and select the custom extensible platform **te0807_52_240_pfm[custom]** in the directory:
~/work/te0807_52_240/StarterKit_pfm/te0807_52_240_pfm/export/te0807_52_240_pfm

We can see available PL clocks and frequencies.

PL4 with 240 MHz clock is 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/te0807_52_240/StarterKit_pfm/sysroots/cortexa72-cortexa53-xilinx-linux
“Root FS”:
~/work/te0807_52_240/StarterKit/os/petalinux/images/linux/rootfs.ext4
“Kernel Image”:
~/work/te0807_52_240/StarterKit/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.

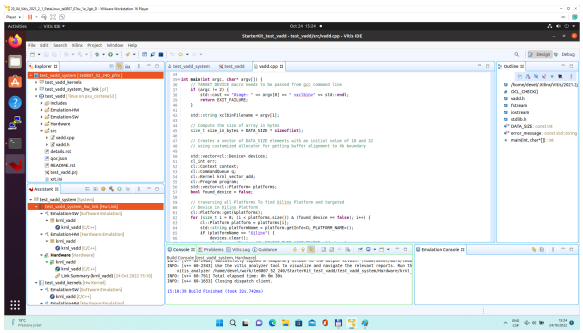
Select: “Vector Addition”
Click **Finish**
New project template is created.

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[te0807_52_240_pfm]** to select it.

Right Click on: **test_vadd_system[te0807_52_240_pfm]** and select in the opened sub-menu:
Build project

Vitis will compile:
In test_vadd_kernels subproject, compile the krnl_vadd from C++ SW to HDL HW IP source code
In test_vadd_system_hw_link subproject, compile the krnl_vadd HDL together with te0807_52_240_pfm into new, extended HW design with new accelerated (krnl_vadd) will run on the default 240 MHz clock. This step can take some time.
In test_vadd subproject, compile the vadd.cpp application example.



Run Compiled Example Application

The sd_card.img file is output of the compilation and packing by Vitis. It is located in directory:
~/work/te0807_52_240/StarterKit_test_vadd/test_vadd_system/Hardware/package/sd_card.img

Write the sd card image from the sd_card.img file 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 TEBF0808 carrier board.

Connect PC USB terminal (115200 bps) card to the TEBF0808 carrier board.

Connect USB Keyboard and USB Mouse to the TEBF0808 carrier board.

Connect Ethernet cable to the TEBF0808 carrier board.

Power on the TEBF0808 carrier board.

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

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

Connect Monitor to the Display Port connector of the TEBF0808 carrier board.

On TEBF0808, press button S1 to start the system (press the button for cca. 1 sec.).
(FMC fan starts to rotate, USB terminal starts to display booting information)

Display Port Monitor indicates text “Please wait: Booting…” (white text, black background).

X11 screen opens on Display port.

Mouse and keyboard connected to the TEBF0808 carrier board can be used.

Click on “Terminal” icon (A Unicode capable rxvt)

Terminal opens as an X11 graphic window.

In terminal, use keyboard connected to the TEBF0808 carrier board 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:

```
sh-5.0# cd /media/sd-mmcblk1p1/
sh-5.0# ./test_vadd krnl_vadd.xclbin
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 te0807_52_240_pfm platform.

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

```
# exit
```

In X11, click ”Shutdown” icon to close down safely.

System is halted. Messages relate to halt of the system can be seen on the USB terminal).
The Display Port output is switched off.
The TEBF0808 carrier board can be powered off by pressing on the S1 switch (cca. 1 sec long).
The FMC fan stops.

The SD card can be safely removed from the TEBF0808 carrier board, now.

The TEBF0808 carrier board can be disconnected from power.

Test 3: Vitis-AI Demo

This test implements simple AI 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.

Create and Build Vitis Design

Create new directory StarterKit_dpu_trd to test Vitis extendable flow example “dpu trd”
~/work/te0807_52_240/StarterKit_dpu_trd

Current directory structure:
~/work/te0807_52_240/StarterKit
~/work/te0807_52_240/StarterKit_pfm
~/work/te0807_52_240/StarterKit_test_vadd
~/work/te0807_52_240/StarterKit_dpu_trd

Change working directory:

```
$cd ~/work/te0807_52_240/StarterKit_dpu_trd
```

In Ubuntu terminal, start Vitis by:

```
$ vitis &
```

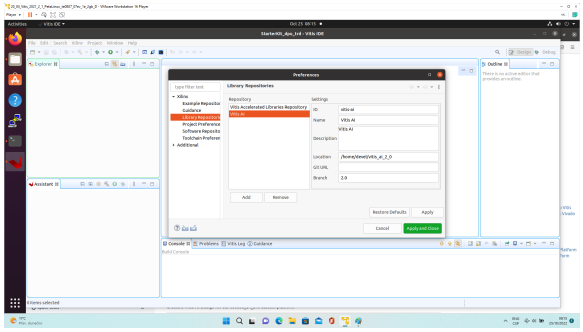
In Vitis IDE Launcher, select your working directory
~/work/te0807_52_240/StarterKit_dpu_trd
Click on Launch to launch Vitis.

Add Vitis-AI Repository to Vitis

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":



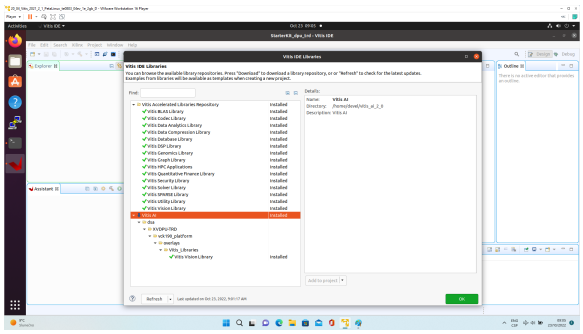
Click Apply and Close.

> Field "Location" says that the Vitis-AI repository from github has been cloned into ~ /vitis_ai_2_0 folder, allready in the stage of Petalinux configuration. It is the same Vitis-AI 2.0 package downloaded from the branch 2.0. 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" as shown in image:



Create a Vitis-AI Design for our te0807_52_240 custom platform

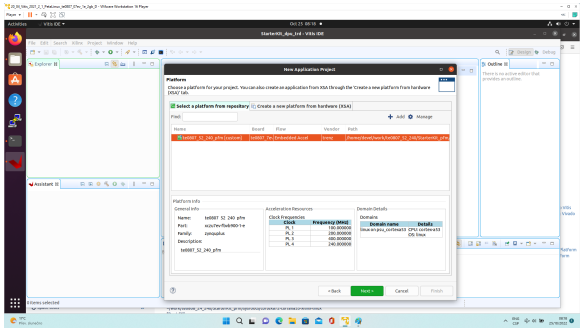
Select **File -> New -> Application project**. Click **Next**.

Skip welcome page if shown.

Click on "**+ Add**" icon and select the custom extensible platform **te0807_52_240_pfm[custom]** in the directory:
~/work/te0807_52_240/StarterKit_pfm/te0807_52_240_pfm/export/te0807_52_240_pfm

We can see available PL clocks and frequencies.

> ⚠ PL4 with 240 MHz clock is has been set as default in the platform creation process.

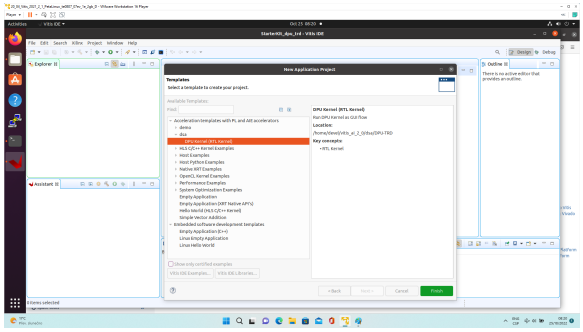


Click **Next**.
In "Application Project Details" window type into Application project name: **dpu_trd**
Click **Next**.
In "Domain window" type (or select by browse):
"Sysroot path":
~/work/te0807_52_240/StarterKit_pfm/sysroots/cortexa72-cortexa53-xilinx-linux
"Root FS":
~/work/te0807_52_240/StarterKit/os/petalinux/images/linux/rootfs.ext4
"Kernel Image":
~/work/te0807_52_240/StarterKit/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 URAM_DISABLE

to

`define URAM_ENABLE

and save modified file.

This modification is necessary for succsessfull implementation of the DPU on the zcu04-ev module with internal memories implemented in URAMs.

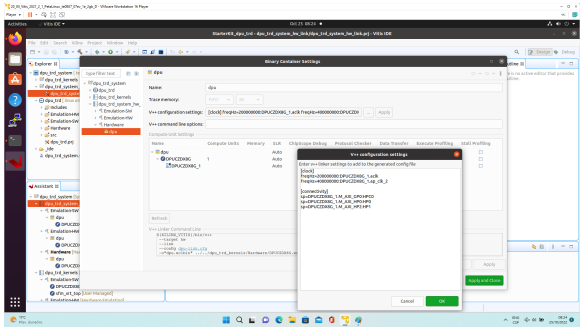
Go to **dpu_trd_system_hw_link** and double click on **dpu_trd_system_hw_link.prj**.

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

Reduce number of DPU kernels to one.

Configure connection of DPU kernels

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
```

Update packaging to add dependencies into SD Card

Create a new folder **img** in your project in **dpu_trd/src/app**

Download image from provided link and place it to newly created folder **dpu_trd/src/app/img.**

Double click **dpu_trd_system.sprj**

Click "**...**" button on **Packaging options**

Enter "**--package.sd_dir=../../dpu_trd/src/app**"

Click **OK**.

Build DPU_TRD application

In “Explorer” section of Vitis IDE, click on: **dpu_trd_system[te0807_52_240_pfm]** to select it.

Right Click on: **dpu_trd_system[te0807_52_240_pfm]** and select in the opened sub-menu:
Build project

Run DPU_TRD on Board

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/te0807_52_240/StarterKit_dpu_trd/dpu_trd_system/Hardware/package/

> 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/

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.

Check ext4 partition size by:

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

Resize partition

```
root@petalinux:~# resize-part /dev/mmcblk1p2
/dev/mmcblk1p2
Warning: Partition /dev/mmcblk1p2 is being used. Are you sure you want to
continue?
parted: invalid token: 100%
Yes/No? yes
End?  [2147MB]? 100%
Information: You may need to update /etc/fstab.

resize2fs 1.45.3 (14-Jul-2019)
Filesystem at /dev/mmcblk1p2 is mounted on /media/sd-mmcblk1p2; o
[   72.751329] EXT4-fs (mmcblk1p2): resizing filesystem from 154804 to
1695488 blocks
n-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
[   75.325525] EXT4-fs (mmcblk1p2): resized filesystem to 1695488
The filesystem on /dev/mmcblk1p2 is now 1695488 (4k) blocks long.
```

Check ext4 partition size again, you should see:

```
root@petalinux:~# 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.

Copy dependencies to home folder:

```
# Libraries
root@petalinux:~# cp -r /mnt/sd-mmcblk1p1/app/samples/ ~
# Model
root@petalinux:~# cp /mnt/sd-mmcblk1p1/app/model/resnet50.xmodel ~
# Host app
root@petalinux:~# cp /mnt/sd-mmcblk1p1/dpu_trd ~
# Images to test
root@petalinux:~# cp /mnt/sd-mmcblk1p1/app/img/*.JPEG ~
```

Run the application from **/home/root** folder and you can observe that "bell pepper" receives highest score.

```
root@petalinux:~# env XLNX_VART_FIRMWARE=/mnt/sd-mmcblk1p1/dpu.xclbin .
/dpu_trd bellpeppe-994958.JPEG
score[945]  =  0.992235     text: bell pepper,
score[941]  =  0.00315807   text: acorn squash,
score[943]  =  0.00191546   text: cucumber, cuke,
score[939]  =  0.000904801  text: zucchini, courgette,
score[949]  =  0.00054879   text: strawberry,
```

App. A: Change History and Legal Notices

Document Change History

To get content of older revision go to "Change History" of this page and select older document revision number.

Date	Document Revision	Authors	Description
Error rendering macro 'page-info' Ambiguous method overloading for method jdk.proxy279.$Proxy4022#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interface	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy279.$Proxy4022#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interface	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy279.$Proxy4022#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interface	• fixed link to Vitis AI Prepare Development Environment

com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]	com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]	com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]	
2022-10-27	v.21	UTIA	- initial release
--	all	**Error rendering macro 'page-info'** Ambiguous method	--

overloading for method jdk.proxy279.$Proxy4022#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interface com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]

		[interface com. atlassian. user.User, class java. lang.String, class com. atlassian. confluence. core. ContentEntit yObject]	

Document change history.