

Application Note



TE0820, Vitis 2024.2 Classic and Unified Extensible Platforms

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	26.04.2025	J.K.	Initial draft
v08	19.06.2026	J.K.	App. note and evaluation package EECONE

Contents

1	Introduction.....	1
1.1	Objective of this Application Note and Evaluation Package.....	2
2	Requirements	2
3	Extensible HW Design for TE0820 modules	3
3.1	Reference HW for TE0820 module	4
4	HW support for Vitis Extensible Design Flow	8
4.1	Create Extensible platform HW	8
4.2	Validate the Extensible Design.....	12
4.3	Compile and Export HW to the Extensible XSA Archive.....	13
5	Building Petalinux	13
5.1	Building Petalinux for the Extensible Design Flow.....	13
5.2	Disable CPU IDLE in Kernel Config	17
5.3	Add EXT4 rootfs Support	18
5.4	Use EXT4 rootfs During Boot	18
5.5	Build PetaLinux Image	18
5.6	Create Petalinux SDK	19
5.7	Copy Created Custom First Stage Boot Loader	19
5.8	Copy Files Required for the Extensible Platforms	19
5.9	Generation of SYSROOT	20
6	Vitis Unified Design Flow	21
6.1	Generation of Vitis Unified Extensible Platform	21
6.2	Read Vitis Unified Platform Info	24
7	Platform Usage	27
7.1	Create and Compile Vitis Unified Free Run Example	27
8	Migration from Vitis Classic to Vitis Unified Design Flow	38
8.1	Generation of Vitis Classic Extensible Platform.....	39
8.2	Read Vitis Classic Extensible Platform Info.....	40
8.3	Create and Compile Vitis Classic Extensible free run xrt example.....	43
8.4	Create and Compile Vitis Classic Extensible free_run_xo_xrt Example.....	44
8.5	Migrate Vitis Classic Extensible project to Vitis Unified	48
9	Export and Import of Vitis Unified 2024.2 Extensible Workspace	56
10	Final Summary.....	61
11	Remote Monitoring and Configuration Support	62
12	References	63

Application Note



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 should be capable to support AMD extensible design flow and accelerated computation in user defined HW IPs integrated in the programmable part of the device. Systems should also support remote monitoring and control from remote PC connected by wired Ethernet in a local network.

The investigated measures and methodologies to support “second life” of electronic modules (T4.3) and measures to support extension of life of electronics (T4.4) due to methodology of support used custom platform to adapt for the in-time-evolving design tools and embedded Linux PetaLinux operating system.

We target developers designing the final commercial, edge applications, mainly in the area of home automation.

Based on these requirements UTIA have selected two types of systems:

- Low cost systems. See [2], [3].
- Module based systems. See [4], [5] and [8], [9].
- [9] is this application note. It is update of [5].
- [5] was supporting system bring-up in AMD Vitis 2022.2 tool chain.
- Application note [9] is supporting system bring-up in AMD Vitis 2023.2 tool chain.

Both compared types of systems (low cost: [2], [3]) and (modular: [4], [5], [8], [9]) 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.

The MCU used on STM32H573I-DK board 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 use 16nm AMD ZynqUltrascale+ device with 64 bit ARM A53 Microprocessor and programmable logic in the same device and Petalinux OS.

- Low-cost systems have an AMD ZynqUltrascale+ device and DDR4 with all peripheral interfaces soldered on a single, low cost PCBs.
- Module-based systems have an AMD ZynqUltrascale+ device and DDR4 soldered on an 4x5 cm module connected by connectors to a carrier board with all peripheral interfaces

1.1 Objective of this Application Note and Evaluation Package

This application note describes Vitis Unified 2024.2 extensible design flow for TE0820 modules on TE0701-06 carrier board.

This application note covers:

- Application design flow performed in Vitis 2024.2 Unified.
- Application design flow in Vitis 2024.2 Classic.
- Migration from Vitis Classic workspace to Vitis Unified workspace.
- Export of Vitis Unified workspace and import to another Vitis Unified workspace.

AMD move to the Vitis Unified workspace is big challenge to the developers. Vitis 2024.2 is the last AMD tool chain flow supporting both, the Vitis Classic and Vitis Unified workspace.

The latest Vitis 2025.1 and Vitis 2025.2 toolchains will support only the Vitis Unified workspace.

That is why it is important for the long-term time-extended support to provide application note describing in detail to the developers how to move from the Vitis Classic and Vitis Unified workspace design flow for the family of TE0820 modules. This is the main objective of this application note and evaluation package.

2 Requirements

This evaluation package requires installation of AMD Vivado 2024.2, Vitis 2024.2 and PetaLinux 2024.2. The application note was tested on Ubuntu 22.04.6.

Change console from dash to bash

```
sudo dpkg-reconfigure dash
```

Press No to disable dash and activate as default the bash console.

Enable i386 architecture.

```
sudo dpkg --add-architecture i386
sudo apt-get update
```

Install these packages:

```
sudo apt-get install iproute2 gawk python3 build-essential gcc git make
net-tools libncurses5-dev tftpd zlib1g-dev libssl-dev flex bison
libselinux1 gnupg wget git 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 libstdl1.2-dev
pylint bc libtinfo5 subversion u-boot-tools -y
```

Installation of Vitis and Vivado 2024.2

Follow the Vitis Release Notes and Installation Guide (UG1742) 2024.2

<https://docs.amd.com/r/en-US/ug1742-vitis-release-notes/Vitis-Release-Notes>

Create Folder /tools/Xilinx

```
sudo mkdir /tools
sudo mkdir /tools/Xilinx
```

Change <owner> to user name

```
sudo chown <owner>:<owner> /tools/Xilinx
```

Example: for user devel

```
sudo chown devel:devel /tools/Xilinx
```

Install Vitis 2024.2 by:

```
$ ./xsetup
```

Important additional installation:

The AMD Vitis 2024.2 tools require installation of additional set of libraries. Without these libraries, the Vitis Unified 2024.2 GUI might experience problems. Install libraries by executing of this AMD script:

```
/tools/Xilinx/Vitis/2024.2/scripts/installLibs.sh
```

Installation of Petalinux 2024.2

Install Petalinux 2024.2 from the same archive downloaded from the AMD Vitis Core Development Kit - 2024.2 Full Project Installation:

```
$ ./xsetup
```

3 Extensible HW Design for TE0820 modules

The design proces is demonstrated for module with ID=23: TE0820-03-04EV-1EA, device xczu4ev-sfvc784-1-e, 2GB DDR4. If your module has different ID, replace 23 the ID.

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

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

Download TE0820 test_board Linux Design file for Vitis 2024.2 from:

https://shop.trenz-electronic.de/trenzdownloads/Trenz_Electronic/Modules_and_Module_Carriers/4x5/TE0820/

[Reference Design/2024.2/Reference Design/
TE0820-te0820-rd-vivado_2024.2-build_1_20250214171345.zip](#)

This Trez Electronic TE0820 archive contains bring-up scripts for creation of reference design HW in Vivado 2024.2 and initial configuration for PetaLinux 2024.2.

Unzip the archive to the directory:

```
~/work/te0820_023_240/
```

All supported modules are identified in file:

```
~/work/te0820_023_240/te0820-rd/board_files/TE0820_board_files.csv
```

We select module 23 with name TE0820-03-04EV-1EA, with device xczu04ev-sfvc784-1-e. We use default clock 240 MHz for HW accelerators.

3.1 Reference HW for TE0820 module

In Ubuntu terminal, change the directory to:

```
$ cd ~/work/te0820_023_240/te0820-rd
```

Setup the permissions for the Linux host machine these files:

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

Execute this bring up script:

```
$ ./_create_linux_setup.sh
```

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

```
devel@ubuntu: ~/work/te0820_023_240/te0820-rd
settings64.sh
devel@ubuntu:~/work/te0820_023_240/te0820-rd$ chmod ugo+rwx ./console/base_sh/*.sh
devel@ubuntu:~/work/te0820_023_240/te0820-rd$ chmod ugo+rwx ./_create_linux_setup.sh
devel@ubuntu:~/work/te0820_023_240/te0820-rd$ ./_create_linux_setup.sh
-----Set design paths-----
-- Run Design with: _create_linux_setup.sh
-- Use Design Path: /home/devel/work/te0820_023_240/te0820-rd
-----
-----_create_linux_setup.cmd-----
-----TE Reference Design-----
-----
-- (d) Go to Documentation (Web Documentation)
-- (x) Exit Batch (nothing is done!)
-- (0) Module selection guide, project creation...
-- (1) Create minimum setup of CMD-Files and exit Batch
-- (2) Create maximum setup of CMD-Files and exit Batch
-- (3) (internal only) Dev
-- (g) Install Board Files from Xilinx Board Store (beta)
-- (a) Start design with unsupported Vivado Version (beta)
-----
Select (ex.: '0' for module selection guide):
```

Select small display format, press enter

```
devel@ubuntu: ~/work/te0820_023_240/te0820-rd
-----
For better table view please resize windows to full screen!
-----
-----
Select Module will be done in 2 steps:
-----
Step 1: (select column filter):
-Change module list size (for small monitors only), press: 'full' or 'small'
-Display current module list, press: 'L' or 'l'
-Restore whole module list, press: 'R' or 'r'
-Reduce List by ID, press: 'ID' or 'id' or insert ID columns value directly(filter step is bypassed and id number is used)
-Reduce List by Article Number, press: 'AN' or 'an'
-Reduce List by SoC/FPGA, press: 'FPGA' or 'fpga'
-Reduce List by PCB REV, press: 'PCB' or 'pcb'
-Reduce List by DDR, press: 'DDR' or 'ddr'
-Reduce List by Flash, press: 'FLASH' or 'flash'
-Reduce List by EMMC, press: 'EMMC' or 'emmc'
-Reduce List by Others, press: 'OTHERS' or 'others'
-Reduce List by Notes, press: 'NOTES' or 'notes'
-Exit without selection, press: 'Q' or 'q'
-----
Please Enter Option:
small
```

Select variant 23

```

devel@ubuntu: ~/work/te0820_023_240/te0820-rd
|126|TE0820-05-S014C1 |4cg_1e_2gb |CAO |
-----|
|127|TE0820-05-S016 |3eg_1e_2gb |CAO |
-----|
|129|TE0820-05-S018 |3cg_1e_2gb |CAO |
-----|
|130|TE0820-05-S020 |3eg_1e_2gb |CAO |
-----|
|131|TE0820-05-S022 |3cg_1e_2gb |CAO |
-----|
|132|TE0820-05-S024C1 |2eg_1e_2gb |CAO |
-----|
|133|TE0820-05-S025 |2eg_1i_2gb |CAO |
-----|
-----|
Select Module will be done in 2 steps:
-----
Step 1: (select column filter):
-Change module list size (for small monitors only), press: 'full' or 'small'
-Display current module list, press: 'L' or 'l'
-Restore whole module list, press: 'R' or 'r'
-Reduce List by ID, press: 'ID' or 'id' or insert ID columns value directly(filter step is bypassed and id number is used)
-Reduce List by Article Number, press: 'AN' or 'an'
-Reduce List by SoC/FPGA, press: 'FPGA' or 'fpga'
-Reduce List by PCB REV, press: 'PCB' or 'pcb'
-Reduce List by DDR, press: 'DDR' or 'ddr'
-Reduce List by Flash, press: 'FLASH' or 'flash'
-Reduce List by EMMC, press: 'EMMC' or 'emmc'
-Reduce List by Others, press: 'OTHERS' or 'others'
-Reduce List by Notes, press: 'NOTES' or 'notes'
-Exit without selection, press: 'Q' or 'q'
-----
Please Enter Option:
23

```

Confirm selection by: y

```

devel@ubuntu: ~/work/te0820_023_240/te0820-rd
|133|TE0820-05-S025 |2eg_1i_2gb |CAO |
-----|
-----|
-----|
Select Module will be done in 2 steps:
-----
Step 1: (select column filter):
-Change module list size (for small monitors only), press: 'full' or 'small'
-Display current module list, press: 'L' or 'l'
-Restore whole module list, press: 'R' or 'r'
-Reduce List by ID, press: 'ID' or 'id' or insert ID columns value directly(filter step is bypassed and id number is used)
-Reduce List by Article Number, press: 'AN' or 'an'
-Reduce List by SoC/FPGA, press: 'FPGA' or 'fpga'
-Reduce List by PCB REV, press: 'PCB' or 'pcb'
-Reduce List by DDR, press: 'DDR' or 'ddr'
-Reduce List by Flash, press: 'FLASH' or 'flash'
-Reduce List by EMMC, press: 'EMMC' or 'emmc'
-Reduce List by Others, press: 'OTHERS' or 'others'
-Reduce List by Notes, press: 'NOTES' or 'notes'
-Exit without selection, press: 'Q' or 'q'
-----
Please Enter Option:
23
Last Input:<23>
Note: Input will be compared with list elements, wildcard * possible. Ex.*1*
Go back to top menu with 'q' or 'Q'
Step 2: Insert ID:
-----|
|ID |Product ID |SHORT DIR |Notes |
-----|
|23 |TE0820-03-04EV-1EA |4ev_1e_2gb |NA |
-----|
-----|
You like to start with this device? y/N

```

Create Vivado 2024.2 Project by selection of option 1.

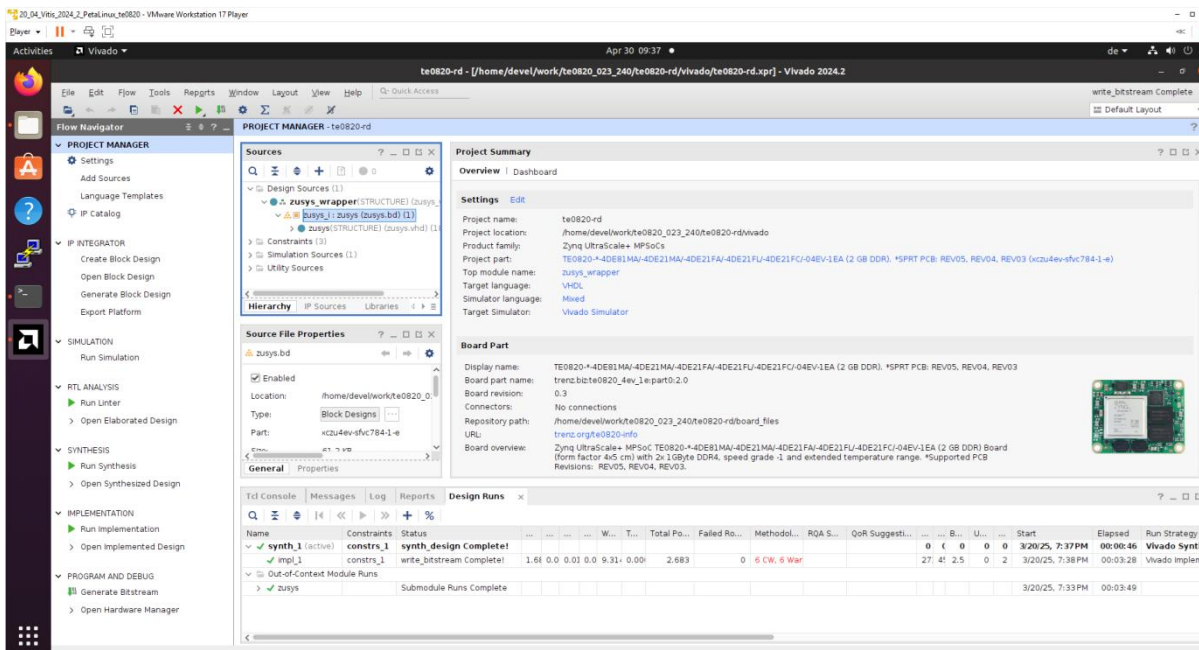
```

devel@ubuntu: ~/work/te0820_023_240/te0820-rd
-----
Step 1: (select column filter):
-Change module list size (for small monitors only), press: 'full' or 'small'
-Display current module list, press: 'L' or 'l'
-Restore whole module list, press: 'R' or 'r'
-Reduce List by ID, press: 'ID' or 'id' or insert ID columns value directly(filter step is bypassed and id number is used)
-Reduce List by Article Number, press: 'AN' or 'an'
-Reduce List by SoC/FPGA, press: 'FPGA' or 'fpga'
-Reduce List by PCB REV, press: 'PCB' or 'pcb'
-Reduce List by DDR, press: 'DDR' or 'ddr'
-Reduce List by Flash, press: 'FLASH' or 'flash'
-Reduce List by EMMC, press: 'EMMC' or 'emmc'
-Reduce List by Others, press: 'OTHERS' or 'others'
-Reduce List by Notes, press: 'NOTES' or 'notes'
-Exit without selection, press: 'Q' or 'q'
-----
Please Enter Option:
23
Last Input:<23>
Note: Input will be compared with list elements, wildcard * possible. Ex.*1*
Go back to top menu with 'q' or 'Q'
Step 2: Insert ID:
-----
|ID |Product ID          |SHORT DIR          |Notes          |
-----|-----|-----|-----|
|23 |TE0820-03-04EV-1EA  |4ev_1e_2gb        |JNA            |
-----|-----|-----|-----|

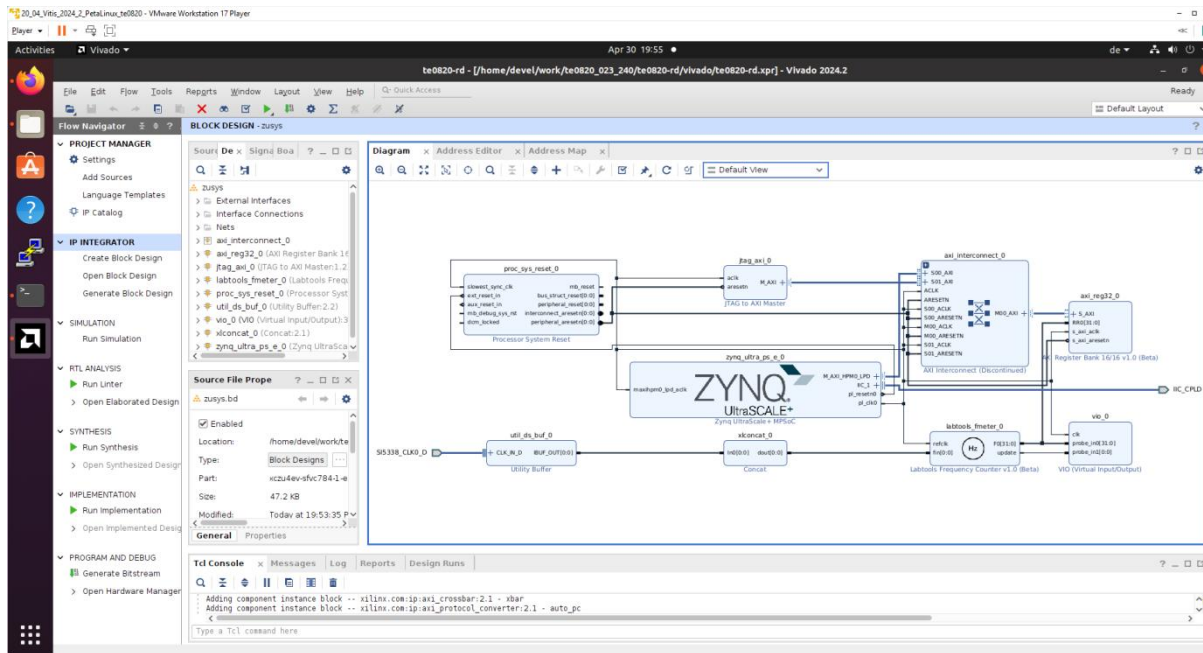
You like to start with this device? y/N
y
What would you like to do?
- Create and open delivery binary folder, press 0
- Create vivado project, press 1
- Both, press 2

```

Vivado 2024.2 Gui is open.



Open block design.



The block design describes the basic reference design for the TE0820 module.

Vivado 2024.2 GUI contains the initial HW design configured for TE0820 module with name TE0820-03-04EV-1EA, with device xczu04ev-sfvc784-1-e.

Vivado 2024.2 project can be generated for the selected variant of module 23.

This design is used for evaluation of fixed embedded design with PetaLinux.

This basic reference design for fixed embedded design flow will be modified to the extensible design. It will be used by the Vitis Unified 2024.2 extensible design flow (and also by Vitis Classi 2024.2 extensible design flow) in next section of this application note.

4 HW support for Vitis Extensible Design Flow

This section describes automated creation of extensible platform HW.

4.1 Create Extensible platform HW

Open IP Integrator Diagram Window.

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

Copy file:

```
te0820_AI_3_0_eval_package\vivado\script_te0820.txt
```

to

```
~/work/te0820_023_240/te0820-rd/vivado/script_te0820.txt
```

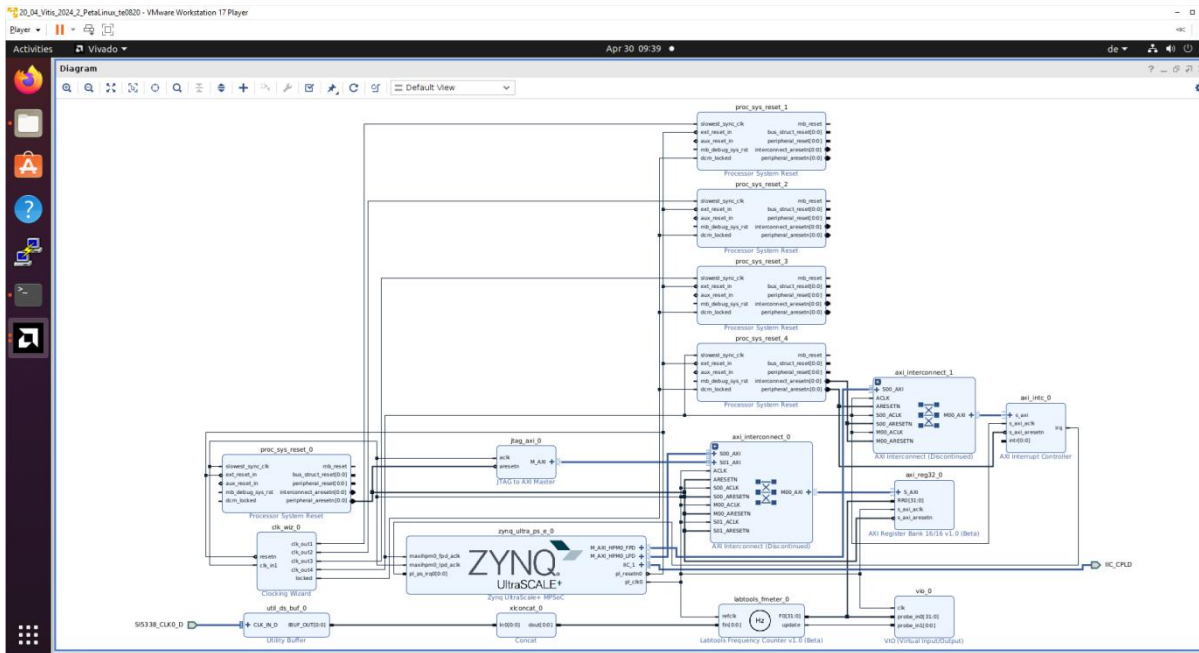
In Vivado Tcl console, Execute this command:

```
source script_te0820.txt
```

It will add to the HW:

- 100 MHz clock will serve as low speed clock.
- 200 MHz and 400 MHz clock will serve as clock for the AMD DPU AI 3.0 HW IP.
- 400 MHz clock will serve as high speed clock.
- 240 MHz clock will serve as the default extensible platform clock. By default, Vitis will compile HW IPs with this default clock.
- Four Processor System Reset blocks for each generated clock.
- AXI Interrupt Controller IP.
- HPC0, HPC1, HP0, HP1, HP2, HP3 are reserved for the Vitis extensible flow.
- M01_AXI, M02_AXI, M03_AXI, M04_AXI, M05_AXI, M06_AXI and M07_AXI are reserved for the Vitis extensible flow.

Created HW system is displayed in IP Integrator Diagram Window.



Script defined this standard Extensible platform setup:

Extensible platform setup: AXI ports

Platform Setup

Settings

- AXI Port
- AXI Stream Port
- Clock
- Interrupt
- Platform Name

AXI Port

Name	Enabled	Memport	SP Tag	Memory
> axi_interconnect_0 (AXI Interconnect:2.1)				
> axi_interconnect_1 (AXI Interconnect:2.1)				
zynq_ultra_ps_e_0 (Zynq UltraScale+ MPSoC:3.5)				
M_AXI_HPM1_0	<input checked="" type="checkbox"/>	M_AXI_GP		
S_AXI_HP0_F	<input checked="" type="checkbox"/>	S_AXI_HP	HP0	
S_AXI_HP1_F	<input checked="" type="checkbox"/>	S_AXI_HP	HP1	
S_AXI_HP2_F	<input checked="" type="checkbox"/>	S_AXI_HP	HP2	
S_AXI_HP3_F	<input checked="" type="checkbox"/>	S_AXI_HP	HP3	
S_AXI_HPC0_0	<input checked="" type="checkbox"/>	S_AXI_HP	HPC0	
S_AXI_HPC1_0	<input checked="" type="checkbox"/>	S_AXI_HP	HPC1	
S_AXI_LPD	<input type="checkbox"/>			

Info: No problems with AXI Port interfaces.

Export Platform...

Extensible platform setup: AXI lite ports

The screenshot shows the 'Platform Setup' window with the 'AXI Port' tab selected. The left sidebar shows 'Settings' with 'AXI Port' checked. The main area displays a table of AXI interconnects and their ports.

Name	Enabled	Memport	SP Tag	Memory
> axi_interconnect_0 (AXI Interconnect:2.1)				
▼ axi_interconnect_1 (AXI Interconnect:2.1)				
M01_AXI	<input checked="" type="checkbox"/>	M_AXI_GP	▼	▼
M02_AXI	<input checked="" type="checkbox"/>	M_AXI_GP	▼	▼
M03_AXI	<input checked="" type="checkbox"/>	M_AXI_GP	▼	▼
M04_AXI	<input checked="" type="checkbox"/>	M_AXI_GP	▼	▼
M05_AXI	<input checked="" type="checkbox"/>	M_AXI_GP	▼	▼
M06_AXI	<input checked="" type="checkbox"/>	M_AXI_GP	▼	▼
M07_AXI	<input checked="" type="checkbox"/>	M_AXI_GP	▼	▼
M08_AXI	<input type="checkbox"/>			
M09_AXI	<input type="checkbox"/>			

Info: No problems with AXI Port interfaces.

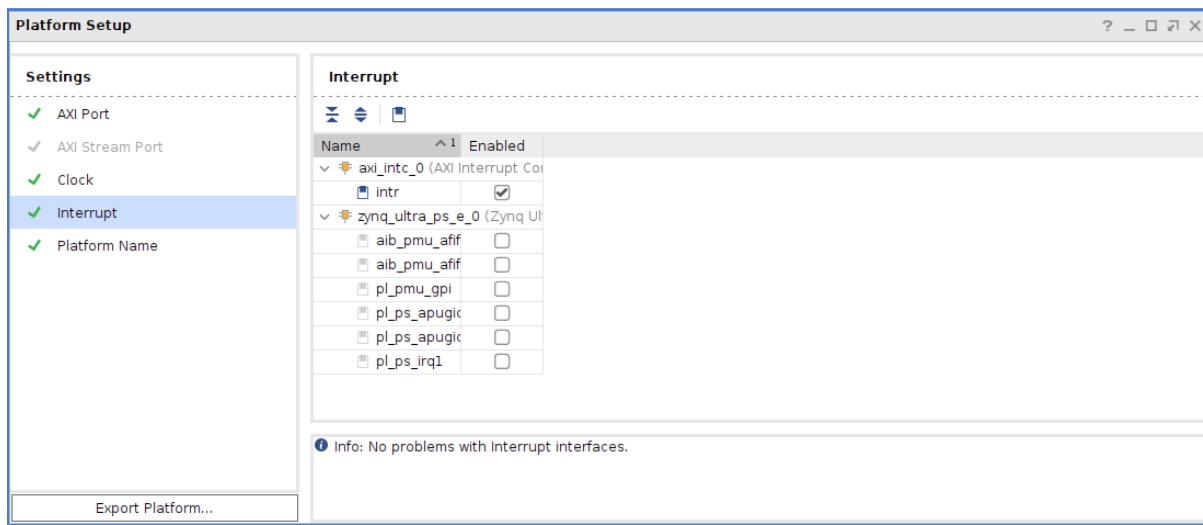
Extensible platform setup: AXI lite clocks

The screenshot shows the 'Platform Setup' window with the 'Clock' tab selected. The left sidebar shows 'Settings' with 'Clock' checked. The main area displays a table of clock outputs and buffers.

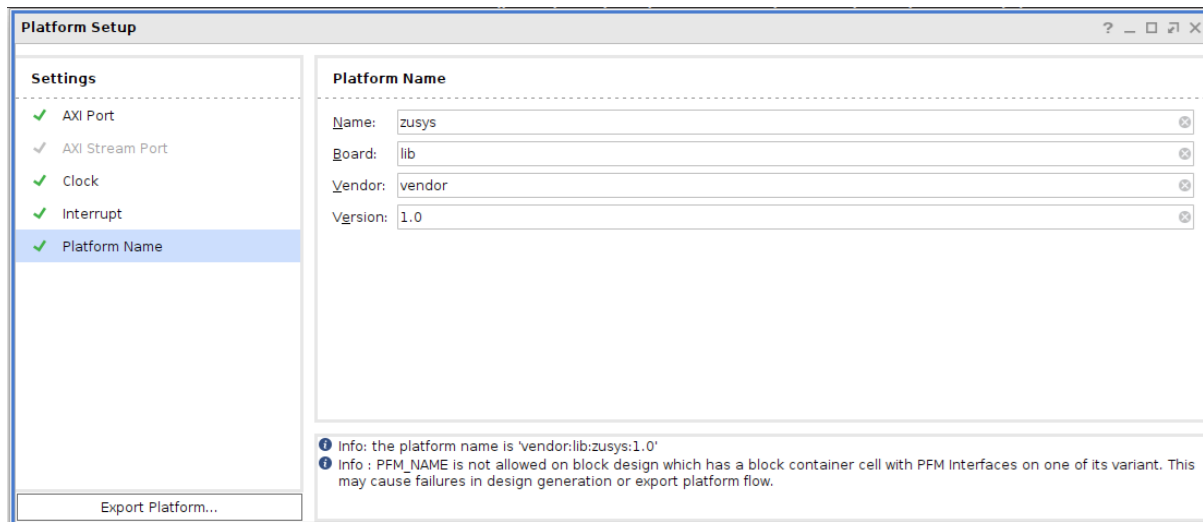
Name	Enabled	ID	Is Default	Proc Sy...	Status	Freque...
▼ clk_wiz_0 (Clocking Wizard:6.0)						
clk_out1	<input checked="" type="checkbox"/>	1	<input type="radio"/>	/proc_s...	fixed	100 MHz
clk_out2	<input checked="" type="checkbox"/>	2	<input type="radio"/>	/proc_s...	fixed	200 MHz
clk_out3	<input checked="" type="checkbox"/>	3	<input type="radio"/>	/proc_s...	fixed	400 MHz
clk_out4	<input checked="" type="checkbox"/>	4	<input checked="" type="radio"/>	/proc_s...	fixed	240 MHz
▼ util_ds_buf_0 (Utility Buffer:2.2)						
IBUF_OUT	<input type="checkbox"/>					
▼ zynq_ultra_ps_e_0 (Zynq UltraScale+ MPSoC:3.5)						
pl_clk0	<input type="checkbox"/>					

Info: No problems with Clock interfaces.

Extensible platform setup: Interrupts



Extensible platform setup: Platform name



4.2 Validate the Extensible Design

Open IP Integrator by clicking on zusys.bd if not already open.
Validate design by clicking on the 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.

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

```
~/work/te0820_023_240/te0820-rd/vivado/zusys.pdf
```

4.3 Compile and Export HW to the Extensible XSA Archive

In Vivado Tcl Console, type the following script and execute it by Enter.

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

HW design is compiled and exported to the XSA package with included bitstream.

Archive te0820-rd_4ev_1e_2gb.xsa for extensible system is created:

```
~/work/te0820_023_240/te0820-rd/vivado/te0820-rd_4ev_1e_2gb.xsa
```

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.

5 Building Petalinux

5.1 Building Petalinux for the Extensible Design Flow

Change directory to the default Petalinux folder

```
~/work/te0820_023_240/te0820-rd/os/petalinux
```

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

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh  
$ source ~/petalinux/2024.2/settings.sh
```

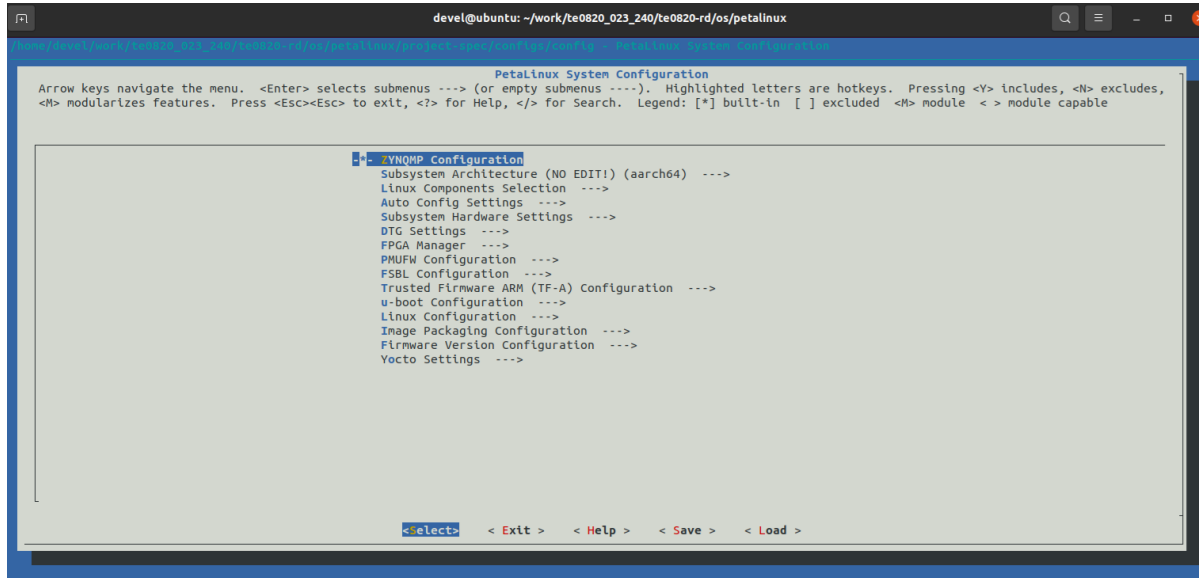
Configure petalinux with the HW defined in te0820-rd_4ev_1e_2gb.xsa archive for the extensible design flow by executing:

```
$ petalinux-config --get-hw-description=  
/home/<user>/work/te0820_023_240/te0820-rd/vivado
```

Replace <user> by your user name.

In our case, <user>=devel use:

```
$ petalinux-config --get-hw-description=  
/home/devel/work/te0820_023_240/te0820-rd/vivado
```



Select Exit->Yes to close this window.

In text editor, modify the user-rootfsconfig file:

```
~/work/te0820_023_240/te0820-rd/os/petalinux/project-spec/meta-user/conf/user-rootfsconfig
```

In text editor, append these lines:

```
#Note: Mention Each package in individual line
#These packages will get added into rootfs menu entry

CONFIG_gpio-demo
CONFIG_peekpoke
CONFIG_startup
CONFIG_webfwu

CONFIG_packagegroup-xilinx-matchbox
CONFIG_packagegroup-xilinx-matchbox-dev

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

```
CONFIG_vai-benchmark

CONFIG_xrt
CONFIG_xrt-dev
CONFIG_zocl
CONFIG_openc1-clhpp-dev
CONFIG_openc1-headers-dev
CONFIG_packagegroup-petalinux-opencv
CONFIG_packagegroup-petalinux-opencv-dev
CONFIG_dnf
CONFIG_e2fsprogs-resize2fs
CONFIG_parted
CONFIG_resize-part
CONFIG_cmake
CONFIG_mesa-megadriver
CONFIG_packagegroup-petalinux-v4lutils
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
```

Launch rootfs config:

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

Enable user packages

Select user packages

```
[*] cmake
[*] dnf
[*] e2fsprogs-resize2fs
[*] gpio-demo
[*] imagefeature-debug-tweaks
[*] imagefeature-package-management
[ ] imagefeature-ssh-server-dropbear
[*] imagefeature-ssh-server-openssh
[ ] mesa-megadriver
[*] opencv-clhpp-dev
[*] opencv-headers-dev
[*] openssh
[*] openssh-scp
[*] openssh-sftp-server
[*] openssh-sshd
[ ] packagegroup-core-ssh-dropbear
[ ] packagegroup-petalinux-opencv
[ ] packagegroup-petalinux-opencv-dev
[ ] packagegroup-petalinux-v4lutils
[*] packagegroup-petalinux-vitis-acceleration-essential
[ ] packagegroup-petalinux-vitis-acceleration-essential-dbg
[*] packagegroup-petalinux-vitis-acceleration-essential-dev
[*] packagegroup-xilinx-matchbox
[*] packagegroup-xilinx-matchbox-dev
[*] parted
[*] peekpoke
[*] resize-part
[*] startup
[ ] vai-benchmark
[ ] vitis-ai-library
```

```
[ ] vitis-ai-library-dbg
[ ] vitis-ai-library-dev
[*] webfwu
[ ] xrt
[ ] xrt-dev
[ ] zocl
```

Select all packages to have an asterisk [*].

xrt, xrt-dev and zocl required for Vitis extensible design flow are already enabled by selected package groups:

packagegroup-petalinux-vitis-acceleration-essential

packagegroup-petalinux-vitis-acceleration-essential-dev

dnf serves for package management in the embedded system.

parted, e2fsprogs-resize2fs and resize-part serves for ext4 partition resize in the embedded system.

webfwu is Trenz Electronic utility for www management of the embedded system.

packagegroup-xilinx-matchbox and packagegroup-xilinx-matchbox is X11 desktop.

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

Select Exit and Yes to save the changes.

5.2 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

Select Exit and Yes to save changes.

5.3 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 TE0820 module:
/dev/mmcblk1p2

Go to

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

Modify Root filesystem formats from

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

to

```
ext4
```

Select Exit and Yes to save changes.

5.4 Use EXT4 rootfs During Boot

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.

5.5 Build PetaLinux Image

In terminal, build the PetaLinux project by executing:

```
$ petalinux-build
```

The PetaLinux image files will be generated in the directory:

```
~/work/te0820_023_240/te0820-rd/os/petalinux/images/linux
```

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

5.6 Create Petalinux SDK

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

In terminal, execute:

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

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

```
~/work/te0820_023_240/te0820-rd/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 the number of PC processor cores.

5.7 Copy Created Custom First Stage Boot Loader

Create new folders:

```
~/work/te0820_023_240/te0820-rd_pfm/pfm/boot  
~/work/te0820_023_240/te0820-rd_pfm/pfm/sd_dir
```

5.8 Copy Files Required for the Extensible Platforms

Copy five files:

Files	From	To
bl31.elf pmufw.elf system.dtb u-boot-dtb.elf zynqmp_fsbl.elf	~/work/te0820_023_240/ te0820-rd/os/petalinux/ images/linux	~/work/te0820_023_240/ te0820-rd_pfm/pfm/boot

Rename copied file from u-boot-dtb.elf to u-boot.elf

Rename copied file from zynqmp_fsbl.elf to fsbl.elf

The zynqmp_fsbl.elf file contains also the Trenz Electronic patches to the standard PetaLinux zynqmp_fsbl . These patches are defined by Trenz Electronic in the directory:

```
~/work/te0820_023_240/te0820-rd/os/petalinux/project-spec/meta-  
user/recipes-bsp/embeddedsw
```

The directory:

```
~/work/te0820_023_240/te0820-rd_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/te0820_023_240/ te0820-rd/os/petalinux/ images/linux	~/work/te0820_023_240/ te0820-rd_pfm/ pfm/sd_dir

Copy file:

File	From	To
init.sh	~/work/te0820_023_240/ te0820-rd/misc/sd	~/work/te0820_023_240/ te0820-rd_pfm/pfm/sd_dir

The directory:

```
~/work/te0820_023_240/te0820-rd_pfm/pfm/sd_dir
```

contains these three files:

```
boot.scr
system.dtb
init.sh
```

5.9 Generation of SYSROOT

In Ubuntu terminal, change the working directory to:

```
~/work/te0820_023_240/te0820-rd/os/petalinux/images/linux
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

In Ubuntu terminal, execute (for <user>= devel) script:

```
$ ./sdk.sh -d /home/devel/work/te0820_023_240/te0820-rd_pfm
```

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

```
~/work/te0820_023_240/te0820-rd_pfm/sysroots/x86_64-petalinux-linux
~/work/te0820_023_240/te0820-rd_pfm/sysroots/
cortexa72-cortexa53-xilinx-linux
```

Once created, do not move or rename these two directories (due to some internally created absolute paths).

6 Vitis Unified Design Flow

6.1 Generation of Vitis Unified Extensible Platform

Create new directory:

```
~/work/te0820_023_240/te0820-rd_pfm_un
```

Current directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
```

In Ubuntu terminal, change the working directory to:

```
~/work/te0820_023_240/te0820-rd_pfm_un/
```

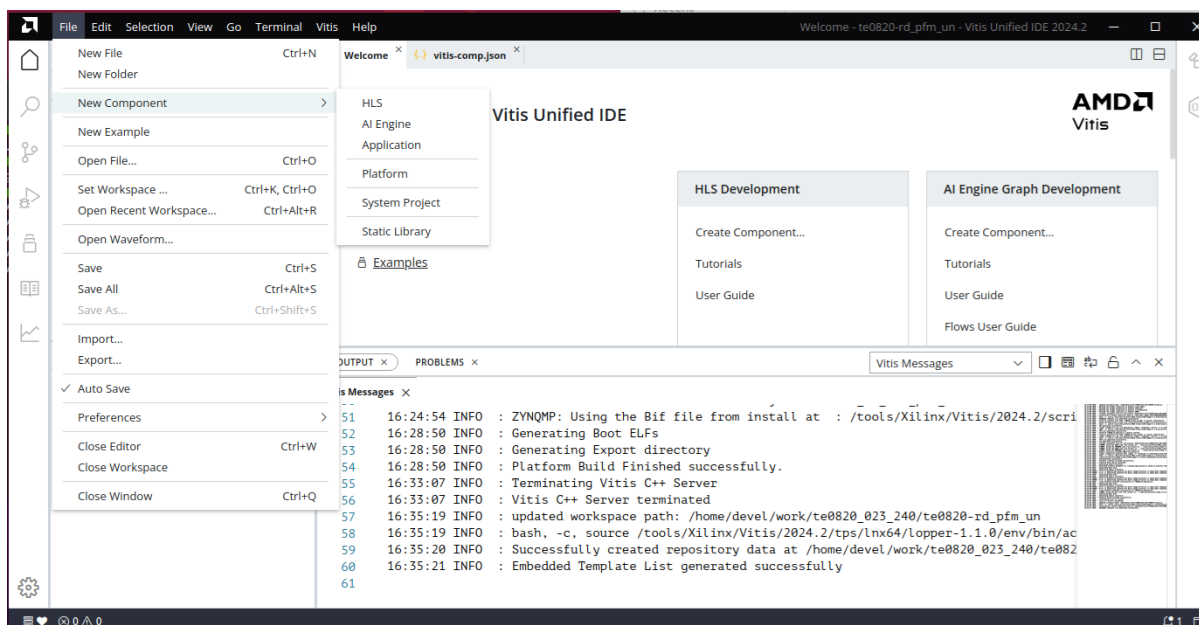
In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

Start the Vitis Unified version of the Vitis tool by executing:

```
$ vitis -w . &
```

In Vitis Unified GUI, select in the main menu: File -> New Component -> Platform



Type name of the extensible platform: te0820_23_240_pfm_un. Click Next.

For hardware specification, select extensible platform archive:

```
~/work/te0820_023_240/te0820-rd/vivado/te0820-rd_4ev_1e_2gb.xsa
```

Select: linux

Select: psu_cortex53

Unselect box: Generate Boot artefacts

(these components have been already generated by PetaLinux compilation)

Create Platform Component

Name and Location > Flow > OS and Processor > Summary

Select Operating System and Processor

Specify the details for the initial domain to be added to the platform component. The platform can be modified later to add new domains or change settings by opening the platform editor.

Operating system: linux

Processor: psu_cortex53

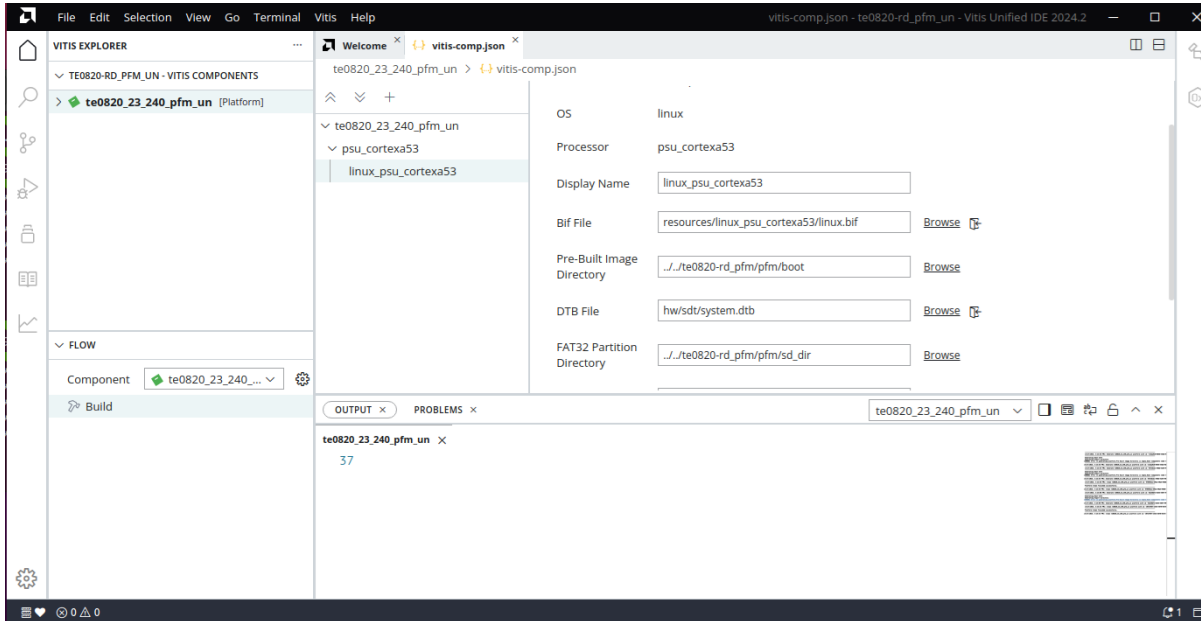
Generate Boot artifacts

> Generate Device Tree Blob (DTB)

Cancel Back Next

New window te0820_23_240_pfm is opened.

Click on linux on psu_cortex53 to open window Domain: linux_domain



In Bif File find and select: Generate Bif

In Boot Components Directory select:

```
~/work/te0820_023_240/te0820-rd_pfm/pfm/boot
```

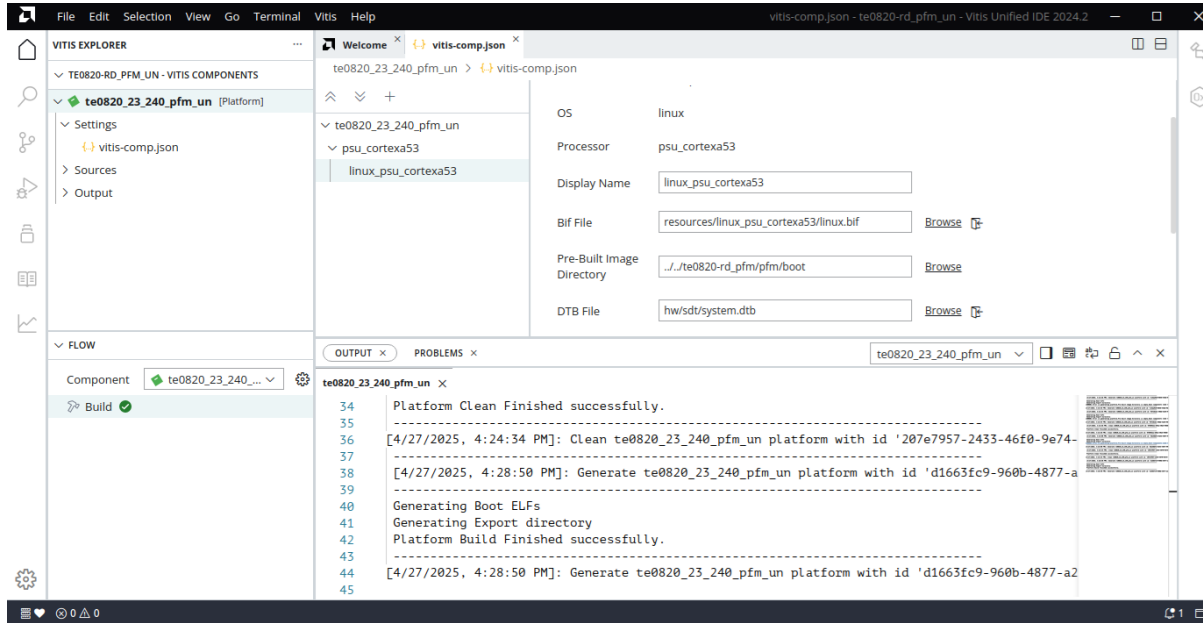
In FAT32 Partition Directory select:

```
~/work/te0820_023_240/te0820-rd_pfm/pfm/sd_dir
```

The Boot Components Directory and the FAT32 Partition Directory are identical to Vitis Classic and to Vitis Unified. These directories contain files generated by Petalinux compilation.

In Vitis IDE Explorer section, click on `te0820_23_240_pfm_un` to highlight it.

Select Build for the component `te0820_23_240_pfm_un` in the flow section of the Vitis Explorer. Vitis Unified platform `te0820_23_240_pfm_un` is compiled in few seconds.



Close the Vitis Unified GUI.

Vits Unified extensible platform `te0820_23_240_pfm_un` has been created.

6.2 Read Vitis Unified Platform Info

With Vitis Unified environment setup, `platforminfo` tool can report Vitis Unified platform information for for Vitis Unified platform `te0820_23_240_pfm_un` :

```
devel@ubuntu:~/work/te0820_023_240/te0820-22rd_pfm_un/
te0820_23_240_pfm_un/export/te0820_023_240_pfm_un$
source/tools/Xilinx/Vitis/2024.2/settings64.sh

devel@ubuntu:~/work/te0820_023_240/te0820-rd_pfm_un/
te0820_23_240_pfm_un/export/te0820_023_240_pfm_un$
platforminfo te0820_23_240_pfm_un.xpfm

=====

Basic Platform Information

=====

Platform:                te0820_23_240_pfm_un

File:                    /home/devel/work/te0820_023_240/
te0820rd_pfm_un/te0820_023_240_pfm_un/export/te0820_023_240_pfm_un/
te0820_23_240_pfm_un.xpfm

Description:

Hardware:                1
Has Software Platform(s): 1
Has Hardware Emulation:  0
```

=====
Hardware Platform (Shell) Information
=====

Vendor: vendor
Board: zusys
Name: zusys
Version: 1.0
Generated Version: 2024.2
Is Extensible: 1
Supports Hardware Target: 1
Is a Hardware Emulation Platform: 0
FPGA Family: zynqplus
FPGA Device: xczu4ev
Board Vendor: trenz.biz
Board Name: trenz.biz:te0820_4ev_1e:2.0
Board Part: xczu4ev-sfvc784-1-e
Resources:
BRAM Count: 128
DSP Count: 728
LUT Count: 87840
FF Count: 175680
URAM Count: 0

=====
AIE Partitions
=====

Start Col: 0
Columns: 0

=====
Clock Information
=====

Default Clock Index: 4
Clock Index: 1
Frequency: 100.000000

```
Status:          fixed
Clock Index:     2
Frequency:       200.000000
Status:          fixed
Clock Index:     3
Frequency:       400.000000
Status:          fixed
Clock Index:     4
Frequency:       240.000000
Status:          fixed
```

```
=====
AIE Hardware Information
=====
```

```
Arch: NO_AIE
NPI Base Address: 0x0
AXI Base Address:
Shim Row Start: -1 # Rows: 0
Core Row Start: -1 # Rows: 0
```

```
=====
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:          2
Default System Configuration: te0820_23_240_pfm_un
System Configurations:
```

```
System Config Name:          te0820_23_240_pfm_un
System Config Description:
System Config Default Processor Group:  linux_psu_cortexa53
System Config Default Boot Image:      standard
System Config Is QEMU Supported:       1
System Config Processor Groups:
  Processor Group Name:         linux_psu_cortexa53
  Processor Group CPU Type:     cortex-a53
  Processor Group OS Name:     linux_psu_cortexa53
System Config Boot Images:
  Boot Image Name:             standard
  Boot Image Type:
  Boot Image BIF:              boot/linux.bif
  Boot Image Data:             linux_psu_cortexa53/image
  Boot Image Boot Mode:
  Boot Image RootFileSystem:
  Boot Image Mount Path:
  Boot Image Read Me:
  Boot Image QEMU Args:        qemu/pmu_args.txt:qemu/qemu_args.txt
  Boot Image QEMU Boot:
  Boot Image QEMU Dev Tree:
Supported Runtimes:
  Runtime: C/C++
```

7 Platform Usage

7.1 Create and Compile Vitis Unified Free Run Example

This Vitis Unified example demonstrates integration of free run kernel increment, performing simple integer addone operation. Input stream data for the free running kernel are defined in mem_read kernel and output stream data are collected by kernel mem_write. Kernels are defined in CPP and compiled by HLS into RTL .xo objects. Kernels are linked to the Vitis unified HW platform. Host CPP program running on Arm A53 processor is communicating with kernels with XRT API and performs SW verification of results computed in HW.

Create new directory:

```
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
```

Current directory structure:

```
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
```

In Ubuntu terminal, change the working directory to:

```
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

Start the `vitis` unified version of Vitis tool by executing

```
$ vitis -w . &
```

Local Vitis Unified Platform Platform is created, as first step.

Local platform is created from the existing `te0820_23_240_pfm_un` platform.

In Vitis Unified GUI, select in the main menu: `File -> New Component -> Platform`

Create Platform Component

Name and Location > Flow > Summary

Select Platform Creation Flow

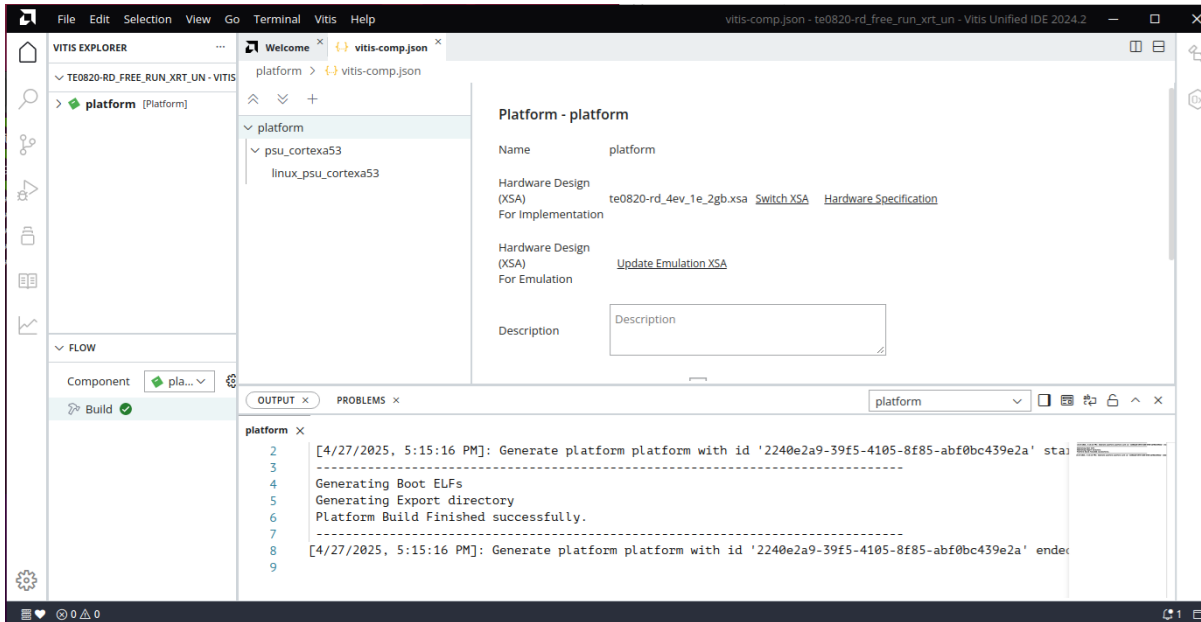
By selecting an existing platform in the repository or workspace, a local copy will be created so that you can do further modification on its software domains.

Hardware Design Existing Platform

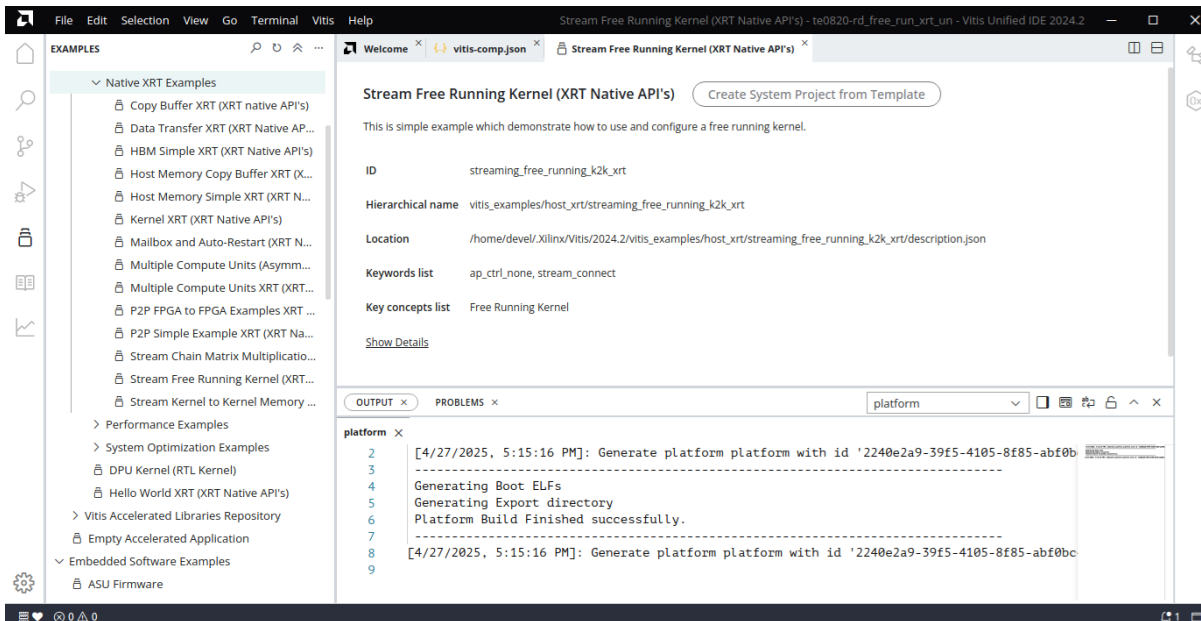
NAME	BOARD	FLOW	VENDOR	PATH
te0820_23_240_pfm_un (1)				...m_un/te0820_23_240_pfm_un/export/te0820_23_240_pfm_un
<input checked="" type="checkbox"/> te0820_23_240_pfm_un	lib	Embedded Accel	xilinx.com	...export/te0820_23_240_pfm_un/te0820_23_240_pfm_un.xpfm
base_platforms (5)				...se_platforms
<input checked="" type="checkbox"/> xilinx_vck190_base_202420_1	vck190	Embedded Accel	xilinx.com	...vck190_base_202420_1/xilinx_vck190_base_202420_1.xpfm
<input checked="" type="checkbox"/> xilinx_zcu104_base_202420_1	xd	Embedded Accel	xilinx.com	...zcu104_base_202420_1/xilinx_zcu104_base_202420_1.xpfm
<input checked="" type="checkbox"/> xilinx_vmk180_base_202420_1	xd	Embedded Accel	xilinx.com	...vmk180_base_202420_1/xilinx_vmk180_base_202420_1.xpfm
<input checked="" type="checkbox"/> xilinx_vek280_base_202420_1	xd	Embedded Accel	xilinx.com	...vek280_base_202420_1/xilinx_vek280_base_202420_1.xpfm
<input checked="" type="checkbox"/> xilinx_zcu102_base_202420_1	xd	Embedded Accel	xilinx.com	...zcu102_base_202420_1/xilinx_zcu102_base_202420_1.xpfm

Cancel Back Next

Build the local Platform platform.



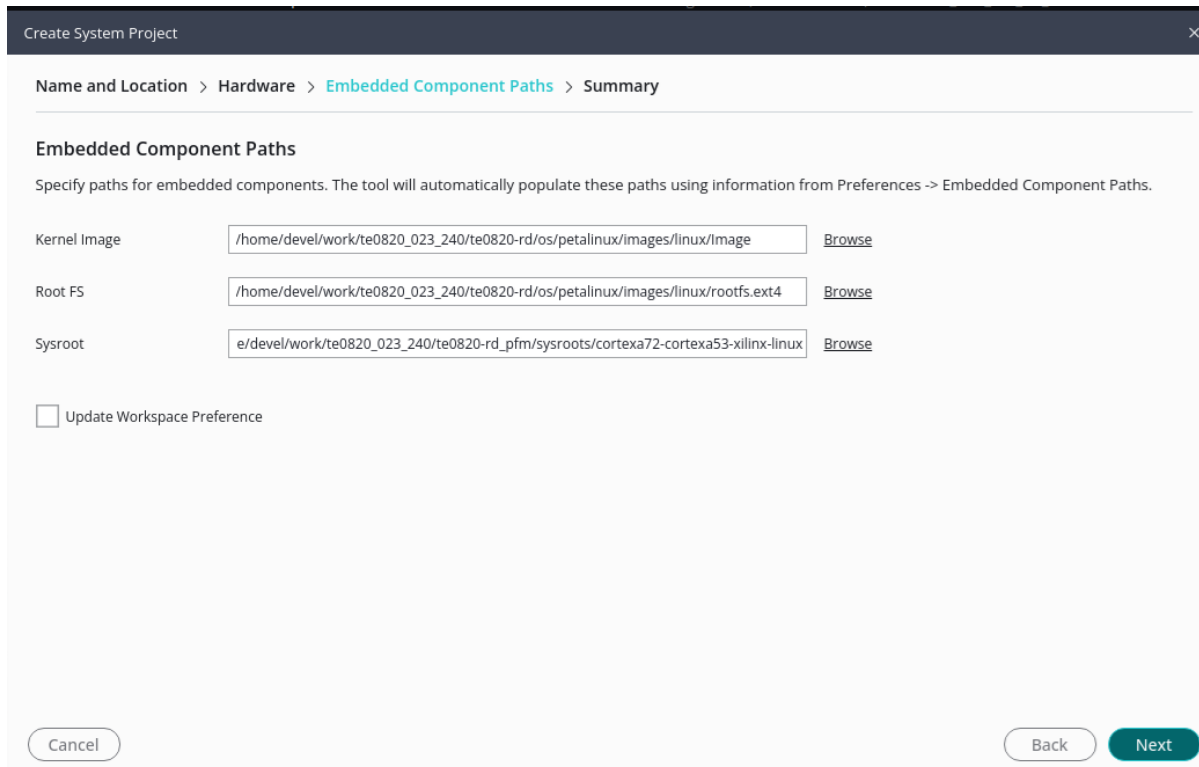
Open example template section Native XRT examples.
 Select Stream Free Running Kernel (XRT Native API).
 Click on Create System Program from Template.



Define Embedded Component Paths to Kernel Image, Root FS and Sysroot:

```
~/work/te0820_023_240/te0820-rd/os/petalinux/images/linux/Image
~/work/te0820_023_240/te0820-rd/os/petalinux/images/linux/rootfs.ext4
~/work/te0820_023_240/te0820-rd_pfm/sysroots/
cortexa72-cortexa53-xilinx-linux
```

These Embedded Components are identical for the Vitis Classic flow and for the Vitis Unified flow.

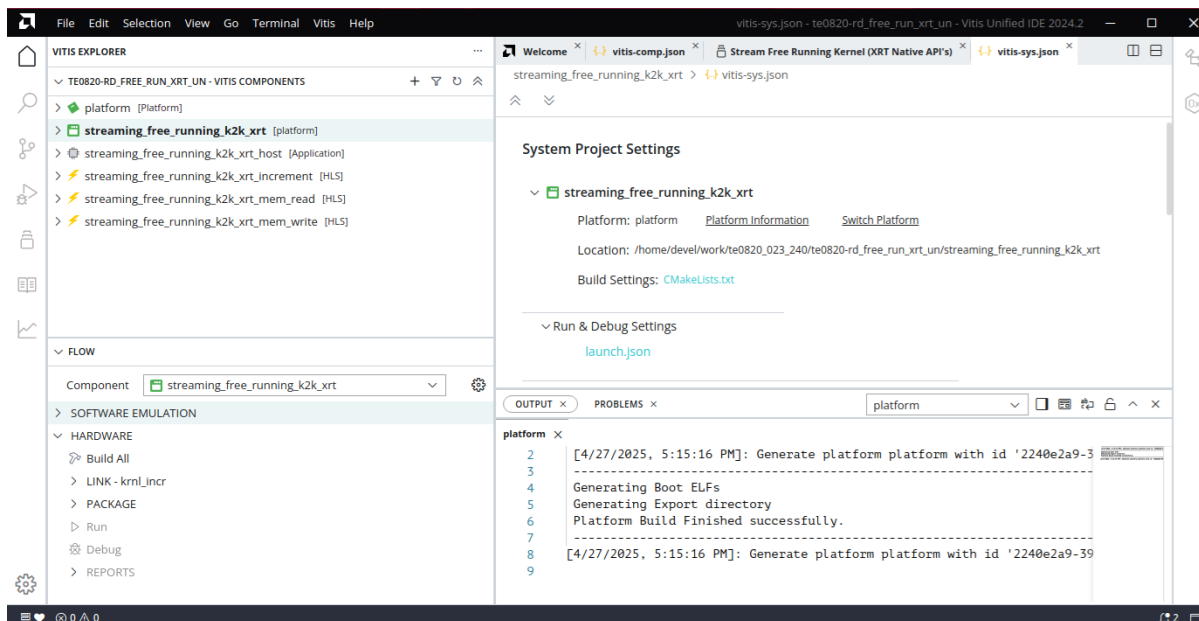


Vitis Unified project is created.

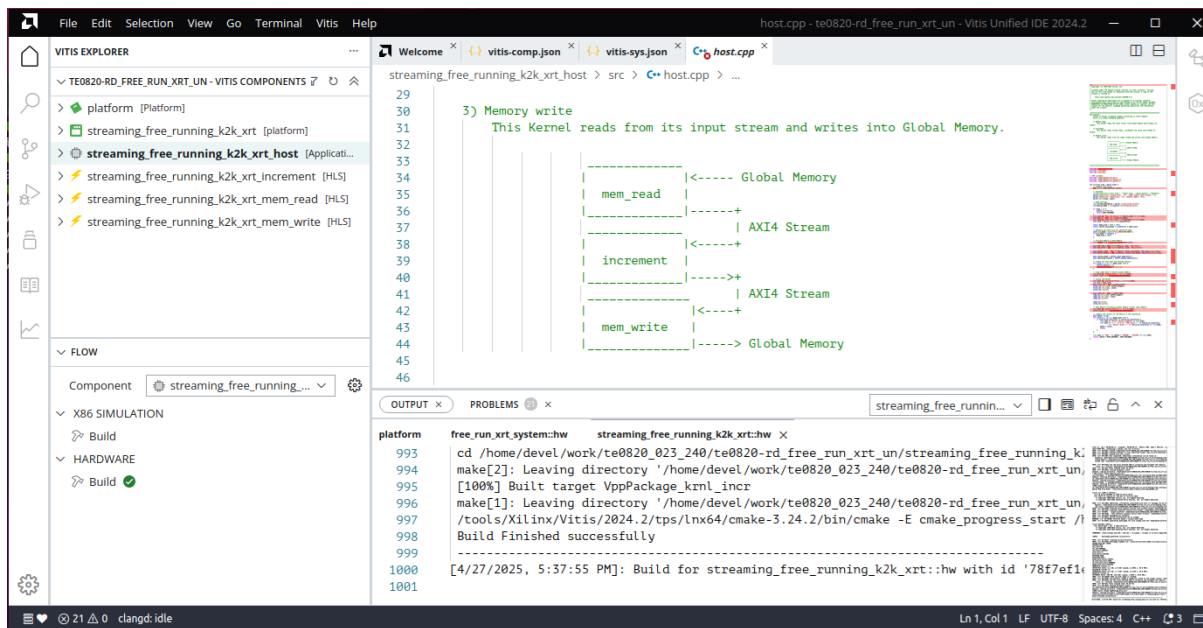
Select system component `streaming_free_running_k2k_xrt`.

In Flow section select **HARDWARE** target and click on **Build all**.

The three kernels will be compiled and linked with the local copy of the Vitis Unified platform. Host CPP project will be compiled and packed. SD card will be created.



Build finished successfully.



Close Vitis Unified GUI.

Host component `streaming_free_running_k2k_xrt_host` contains CPP source code for Arm A53 host processor with Linux.

The SD card image `sd_card.img` is created in the directory:

```
~/work/te0820_023_240/te0820-rd_free_run_xrt_un/  
streaming_free_running_k2k_xrt/build/hw/package/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/>

Create SD card and start evaluation board.

Change directory to

```
/run/media/mmcblk1p1/
```

Run Stream Free Running Kernel (XRT Native API) by command

```
./streaming_free_running_k2k_xrt_host -x krnl_incr.xclbin
```

The application returns:

```
TEST PASSED
```

Test of Vitis Unified application have passed.

```
COM5 - PuTTY
[ OK ] Started Getty on tty1.
[ OK ] Started Serial Getty on ttyPS0.
[ OK ] Reached target Login Prompts.
[ OK ] Started Target Communication Framework agent.
[ OK ] Reached target Multi-User System.
Starting Record Runlevel Change in UTMP...
[ OK ] Finished Record Runlevel Change in UTMP.

*****
*****
The PetaLinux source code and images provided/generated are for demonstration pu
rposes only.
Please refer to https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/2741928025
/Moving+from+PetaLinux+to+Production+Deployment
for more details.
*****
*****
PetaLinux 2024.2+release-S11061705 Trenz ttyPS0

Trenz login: root (automatic login)

Init Start
Run init.sh from SD card
Load SD Init Script
User bash Code can be insered here and put init.sh on SD
Init End
root@Trenz:~# cd /run/media/mmcblkpl1/
root@Trenz:/run/media/mmcblkpl1# ./streaming_free_running_k2k_xrt_host -x krnl_i
ncr.xclbin
Open the device0
Load the xclbin krnl_incr.xclbin
Copying data...
Launching Kernel...
Getting Results...
TEST PASSED
root@Trenz:/run/media/mmcblkpl1#
```

Reboot Linux by typing:

```
root@Trenz:reboot
```

Linux is rebooted.

The Vitis Unified application can be tested in X11 terminal connected to the board by local Ethernet.

Type ifconfig to get the Ethernet address assigned to the board by the DHCP server:

```
root@Trenz:ifconfig
```

```
COM5 - PuTTY
[ OK ] Started Getty on tty1.
[ OK ] Started Serial Getty on ttyPS0.
[ OK ] Reached target Login Prompts.
[ OK ] Started Target Communication Framework agent.
[ OK ] Reached target Multi-User System.
      Starting Record Runlevel Change in UTMP...
[ OK ] Finished Record Runlevel Change in UTMP.

*****
The PetaLinux source code and images provided/generated are for demonstration purposes only.
Please refer to https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/2741928025/Moving+from+PetaLinux+to+Production+Deployment
for more details.
*****
PetaLinux 2024.2+release-S11061705 Trenz ttyPS0

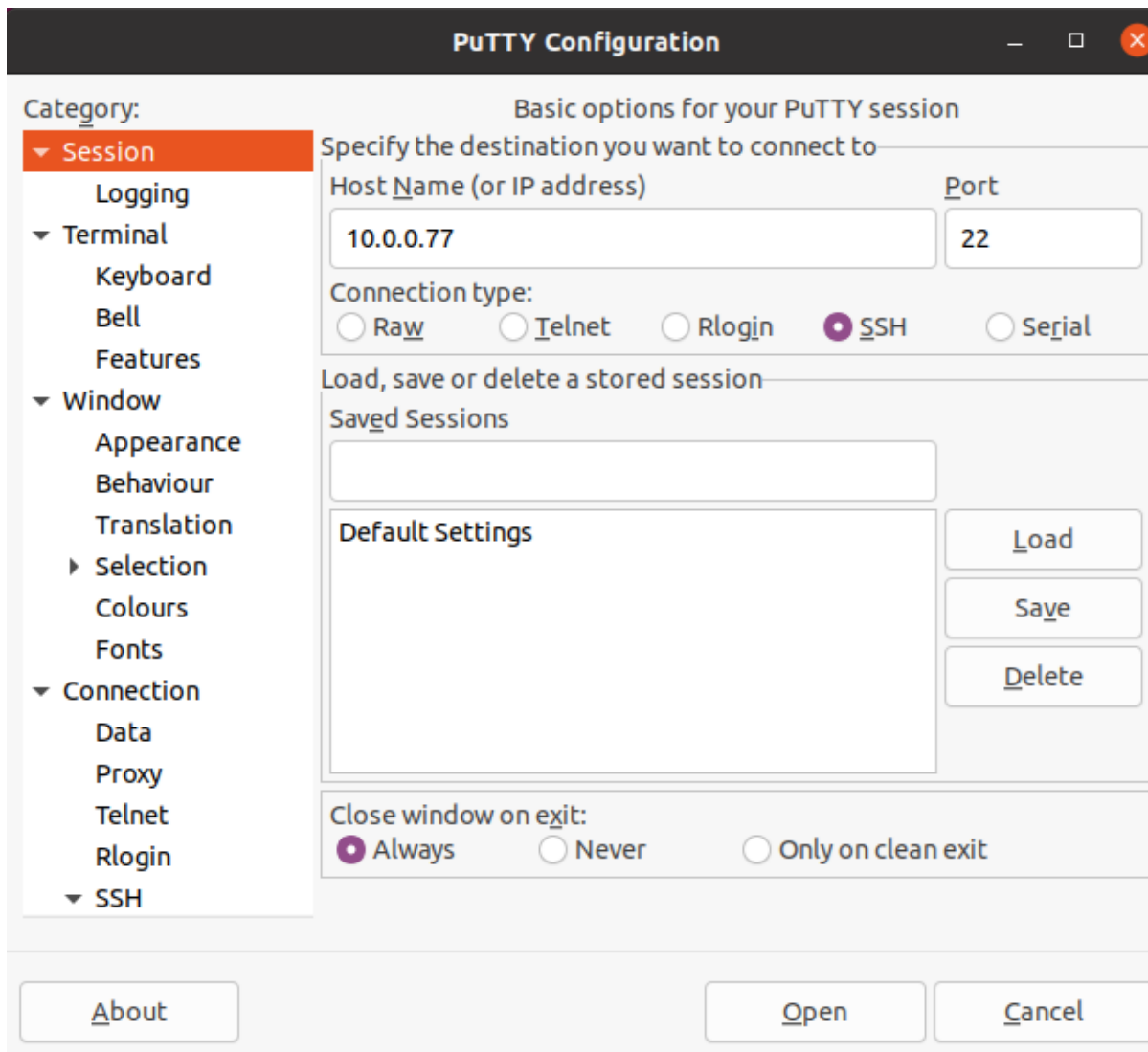
Trenz login: root (automatic login)

Init Start
Run init.sh from SD card
Load SD Init Script
User bash Code can be inserted here and put init.sh on SD
Init End
root@Trenz:~# ifconfig
end0      Link encap:Ethernet  HWaddr 54:10:EC:6E:3D:13
          inet addr:10.0.0.77  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::5610:ecff:fe6e:3d13/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:279  errors:0  dropped:0  overruns:0  frame:0
          TX packets:167  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19028 (18.5 KiB)  TX bytes:15288 (14.9 KiB)
          Interrupt:49

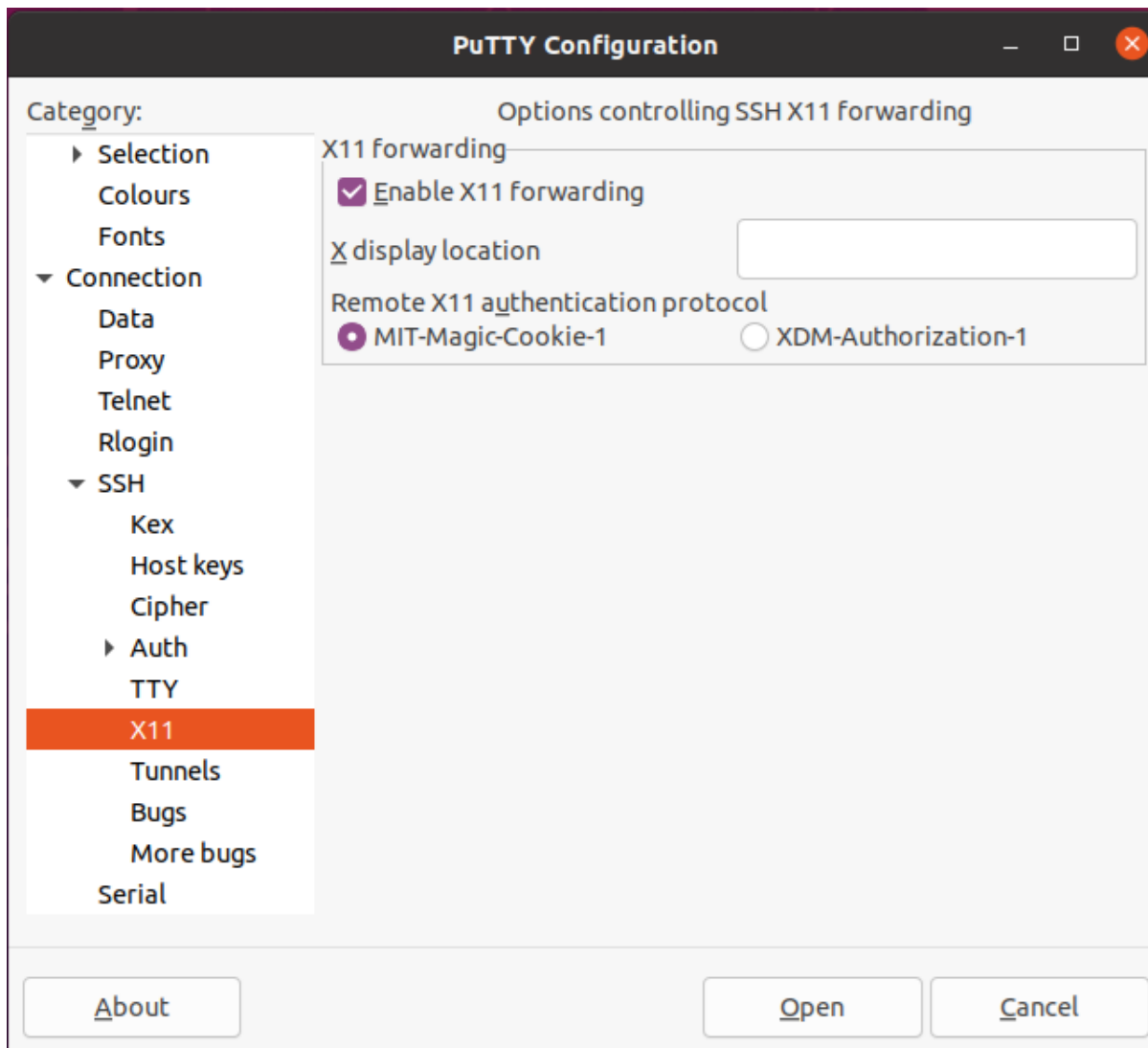
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:80  errors:0  dropped:0  overruns:0  frame:0
          TX packets:80  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6080 (5.9 KiB)  TX bytes:6080 (5.9 KiB)

root@Trenz:~# █
```

In PC Ubuntu, open PuTTY terminal.



Enable X11 forwarding in SSH → X11 menu of PuTTY terminal.

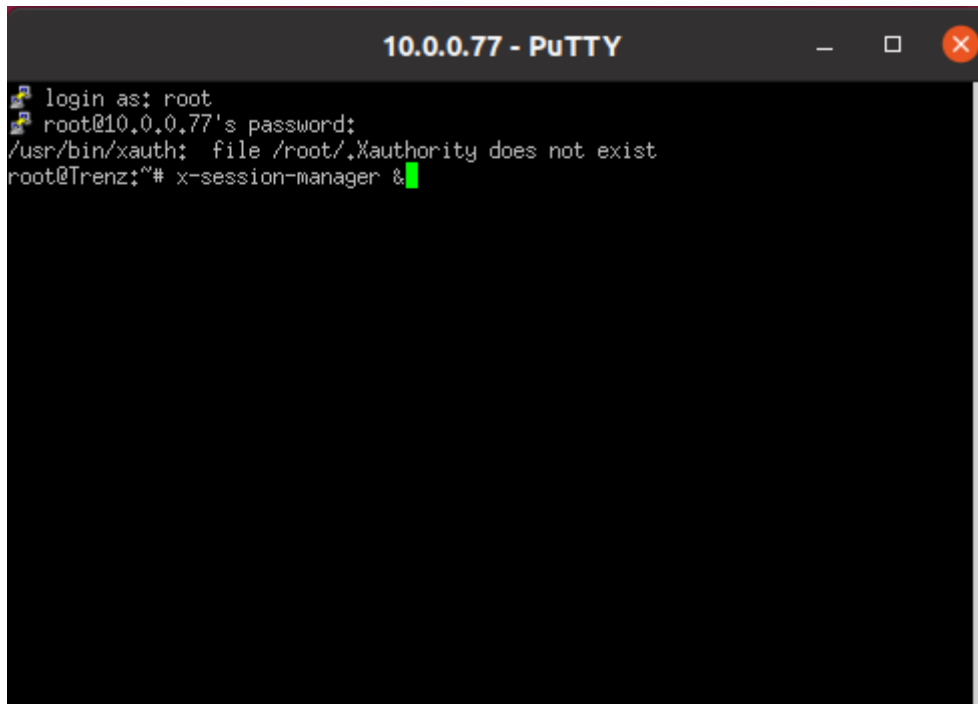


Start PuTTY terminal.

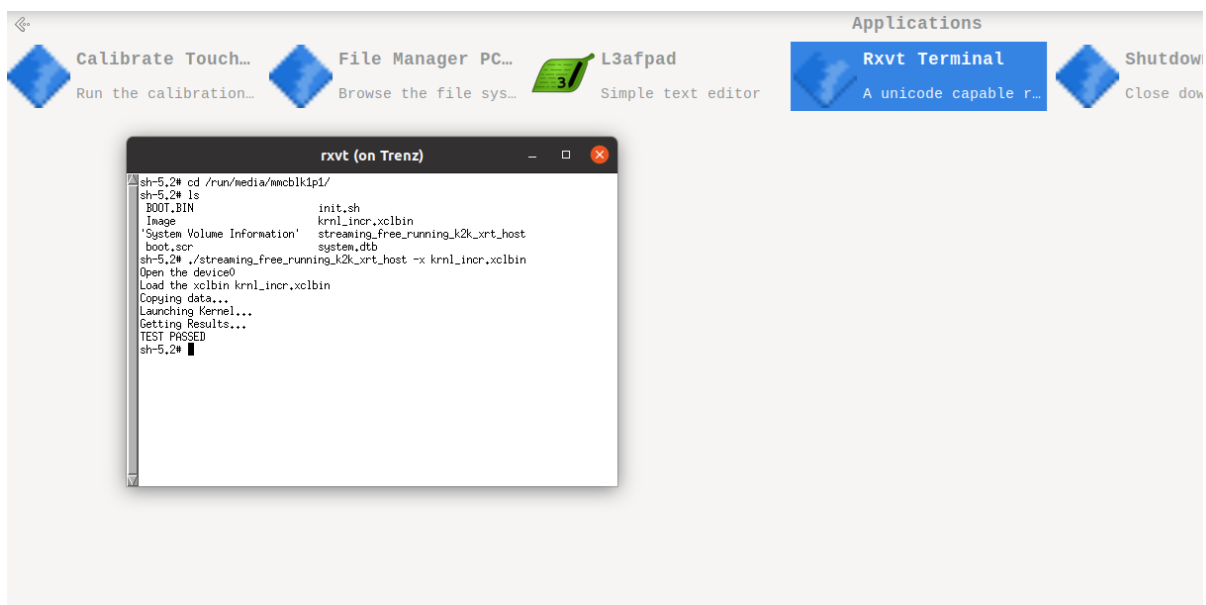
Login as root with pswd root .

Start x-session manager by typing

```
x-session manager &
```



Open Rxt Terminal and test the Vitis Unified application.



Click on Shutdown icon to close the X11 connection and to halt the Linux running on board.

```
COM5 - PuTTY
[ OK ] Stopped IPv6 Packet Filtering Framework.
[ OK ] Stopped IPv4 Packet Filtering Framework.
[ OK ] Stopped Generate network units from Kernel command line.
[ OK ] Stopped D-Bus System Message Bus.
[ OK ] Stopped target Basic System.
[ OK ] Stopped target Path Units.
[ OK ] Stopped Dispatch Password Requests to Console Directory Watch.
[ OK ] Stopped Forward Password Requests to Wall Directory Watch.
[ OK ] Stopped target Slice Units.
[ OK ] Removed slice User and Session Slice.
[ OK ] Stopped target Socket Units.
[ OK ] Closed D-Bus System Message Bus Socket.
[ OK ] Closed sshd.socket.
[ OK ] Stopped target System Initialization.
[ OK ] Closed Syslog Socket.
[ OK ] Closed Network Service Netlink Socket.
Stopping Network Name Resolution...
Stopping Network Time Synchronization...
[ OK ] Stopped Network Name Resolution.
[ OK ] Stopped Network Time Synchronization.
[ OK ] Stopped Apply Kernel Variables.
[ OK ] Stopped Load Kernel Modules.
[ OK ] Stopped Create Volatile Files and Directories.
[ OK ] Stopped target Local File Systems.
Unmounting /run/media/mmcblkpl...
Unmounting Temporary Directory /tmp...
Unmounting /var/volatile...
[ OK ] Unmounted Temporary Directory /tmp.
[ OK ] Unmounted /var/volatile.
[ OK ] Stopped target Swaps.
[ OK ] Unmounted /run/media/mmcblkpl.
[ OK ] Reached target Unmount All Filesystems.
[ OK ] Stopped File System Check on /dev/mmcblkpl.
[ OK ] Removed slice /system/systemd-fsck.
[ OK ] Stopped target Preparation for Local File Systems.
[ OK ] Stopped Remount Root and Kernel File Systems.
[ OK ] Stopped Create Static Device Nodes in /dev.
[ OK ] Stopped Create Static Device Nodes in /dev gracefully.
[ OK ] Reached target System Shutdown.
[ OK ] Reached target Late Shutdown Services.
[ OK ] Finished System Power Off.
[ OK ] Reached target System Power Off.
[ 1234.913682] reboot: Power down
```

Linux can be also halted by typing halt in the terminal

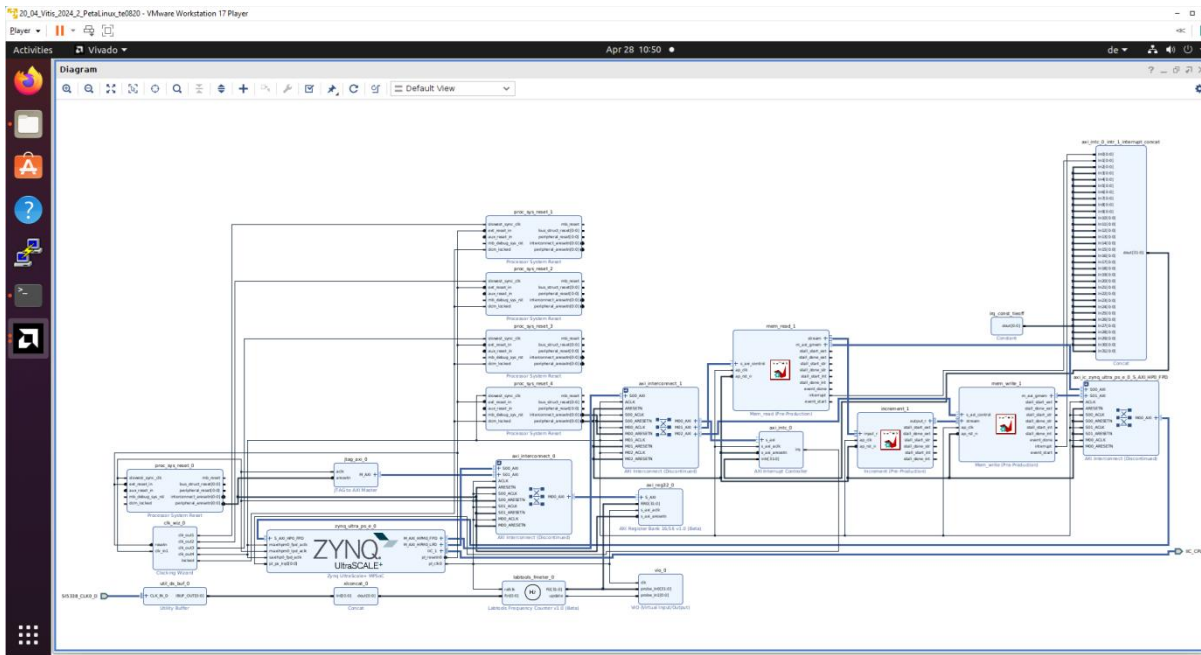
```
halt
```

Remove SD card, close terminal and switch off the board.

The created block design can be open in Vivado project located in the directory:

```
~/work/te0820_023_240/te0820-rd_free_run_xrt_un/
streaming_free_running_k2k_xrt/build/hw/hw_link/
krnl_incr/krnl_incr/vivado/vpl/prj/prj.xpr
```

The HW IP kernels are integrated with the PS host.



Kernels work with the default 240 MHz clock.

The Vitis Unified extensible design flow is completed.

8 Migration from Vitis Classic to Vitis Unified Design Flow

This chapter explains migration from system developed in Vitis Classic design flow to Vitis Unified design flow.

To explain and demonstrate the migration process, we have to create Vitis Classic system, based on the Vitis Classic extensible platform.

We will also explain how to migrate from Vitis Classic project to Vitis Unified project, if project contains some CPP kernels and some precompiled RTL .xo kernels. These RTL .xo kernels might have been created from the CPP source code by the Vitis HLS flow or imported from another toolchain like the AMD Model Composer 2024.2 tool.

This is brief content the described design flow:

- Vitis Classic Extensible platform will be created first.
- First Vitis Classic Free-run (native XRT) example project will be created from a Vitis Classic template. It will serve for compilation of the increment.xo RTL kernel.
- Second Vitis Classic project Free-run_xo (native XRT) example project will be created from the same Vitis Classic template. However, the CPP definition of the increment kernel will be replaced by an imported increment.xo RTL kernel from the first project.
- Finally, the second Vitis Classic Free-run_xo (native XRT) project will be migrated to the Vitis Unified project and compiled in the Vitis Unified design flow.

8.1 Generation of Vitis Classic Extensible Platform

Current directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
```

In Ubuntu terminal, change the working directory to:

```
~/work/te0820_023_240/te0820-rd_pfm
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

Start the `vitis classic` version of Vitis tool by executing:

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

In Vitis Classic “Launcher”, launch the `vitis classic` version of Vitis.

Close Welcome page.

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

Type name of the extensible platform: `te0820_23_240_pfm` . Click Next.

For hardware specification, select extensible platform archive:

```
~/work/te0820_023_240/te0820-rd/vivado/te0820-rd_4ev_1e_2gb.xsa
```

This archive is identical for the Vitis Unified extensible platform as well as for the Vitis Classic extensible platform.

In Software specification select: `linux`

In Boot Components unselect Generate boot components
(these components have been already generated by PetaLinux flow)

New window `te0820_23_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/te0820_023_240/te0820-rd_pfm/pfm/boot
```

In FAT32 Partition Directory select:

```
~/work/te0820_023_240/te0820-rd_pfm/pfm/sd_dir
```

Click on `te0820_23_240_pfm_un` to highlight it.

Right-click on the highlighted `te0820_23_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 Classic extensible platform `te0820_23_240_pfm` has been created.

8.2 Read Vitis Classic Extensible Platform Info

With Vitis environment setup, `platforminfo` tool can report the Vitis Classic platform information for Vitis Classic platform `te0820_23_240_pfm`:

```
devel@ubuntu:~/work/te0820_023_240/te0820rd_pfm/
te0820_23_240_pfm/export/te0820_023_240_pfm$
source/tools/Xilinx/Vitis/2024.2/settings64.sh

devel@ubuntu:~/work/te0820_023_240/te0820-rd_pfm/te0820_023_240_pfm/
export/te0820_023_240_pfm$ platforminfo te0820_23_240_pfm.xpfm

=====

Basic Platform Information

=====

Platform:          te0820_23_240_pfm
File:              /home/devel/work/te0820_023_240/
te0820-rd_pfm/te0820_023_240_pfm/export/te0820_023_240_pfm/
te0820_23_240_pfm.xpfm
Description:
te0820_23_240_pfm

Hardware:          1
Has Software Platform(s): 1
Has Hardware Emulation: 0

=====

Hardware Platform (Shell) Information

=====

Vendor:            vendor
Board:             zusys
Name:              zusys
Version:           1.0
Generated Version: 2024.2
Is Extensible:     1
```

```
Supports Hardware Target:      1
Is a Hardware Emulation Platform: 0
FPGA Family:                   zynqplus
FPGA Device:                   xczu4ev
Board Vendor:                  trenz.biz
Board Name:                    trenz.biz:te0820_4ev_1e:2.0
Board Part:                    xczu4ev-sfvc784-1-e
```

Resources:

```
BRAM Count:                   128
DSP Count:                    728
LUT Count:                    87840
FF Count:                    175680
URAM Count:                   0
```

=====

AIE Partitions

=====

```
Start Col: 0
# Columns: 0
```

=====

Clock Information

=====

```
Default Clock Index: 4
Clock Index:         1
  Frequency:         100.000000
  Status:            fixed
Clock Index:         2
  Frequency:         200.000000
  Status:            fixed
Clock Index:         3
  Frequency:         400.000000
  Status:            fixed
Clock Index:         4
  Frequency:         240.000000
  Status:            fixed
```

```
=====
AIE Hardware Information
=====
Arch: NO_AIE
NPI Base Address: 0x0
AXI Base Address:
Shim Row Start: -1 # Rows: 0
Core Row Start: -1 # Rows: 0
```

```
=====
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: te0820_23_240_pfm
System Configurations:
  System Config Name:          te0820_23_240_pfm
  System Config Description:   te0820_23_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:           te0820_23_240_pfm/boot/linux.bif
Boot Image Data:         te0820_23_240_pfm/linux_domain/image
Boot Image Boot Mode:    sd
Boot Image RootFileSystem:
Boot Image Mount Path:   /mnt
Boot Image Read Me:      te0820_23_240_pfm/boot/generic.readme
Boot Image QEMU Args:
te0820_23_240_pfm/qemu/pmu_args.txt:te0820_23_240_pfm/qemu/qemu_args.tx
t
Boot Image QEMU Boot:
Boot Image QEMU Dev Tree:
Supported Runtimes:
```

8.3 Create and Compile Vitis Classic Extensible free run xrt example

Create new directory `te0820-rd_free_run_xrt` to test Vitis extendable flow example

```
~/work/te0820_023_240/te0820-rd/te0820-rd_free_run_xrt
```

Current directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xrt
```

Change working directory:

```
$cd ~/work/te0820_023_240/te0820-rd_free_run_xrt
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

In Ubuntu terminal, start `vitis classic` version of Vitis by:

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

In Vitis IDE Launcher, launch Vitis classic.

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

Skip welcome page, if shown.

Click on [+ Add] icon and select the custom extensible platform `te0820_23_240_pfm` [custom] in the directory:

```
~/work/te0820_023_240/te0820-rd_pfm/te0820_23_240_pfm/  
export/te0820_23_240_pfm
```

We can see all available PL clocks and frequencies. PL4 with 240 MHz clock was set as the default in the extensible HW generation proces in the Vivado 2024.2.

Click Next.

In Domain window select by browse:

Sysroot path:

```
~/work/te0820_023_240/te0820-rd_pfm/  
/sysroots/cortexa72-cortexa53-xilinx-linux
```

Root FS file:

```
~/work/te0820_023_240/te0820-rd/  
/os/petalinux/images/linux/rootfs.ext4
```

Kernel Image file:

```
~/work/te0820_023_240/te0820-rd/os/petalinux/images/linux/Image
```

Click Next.

Select Native XRT Examples:

Select: Stream Free Running Kernel (XRT Native API)

Click Finish

New project template is created.

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

In “Explorer” section of Vitis Classic IDE, click on: `free_run_xrt_system` [te0820_23_240_pfm] to select it.

Right Click on: `free_run_xrt_system` [te0820_23_240_pfm] and select the sub-menu: Build project

Vitis Classic will compile the project.

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

```
~/work/te0820_023_240/te0820-rd_free_run_xrt/free_run_xrt_system/  
Hardware/package/sd_card.img
```

8.4 Create and Compile Vitis Classic Extensible `free_run_xo_xrt` Example

Create new directory `te0820-rd_free_run_xo_xrt` to test Vitis extendable flow example

```
~/work/te0820_023_240/te0820-rd/te0820-rd_free_run_xo_xrt
```

Current directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xrt
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt
```

Change working directory:

```
$ cd ~/work/te0820_023_240/te0820-rd_free_run_xo_xrt
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

In Ubuntu terminal, start Vitis Classic GUI by:

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

In Vitis IDE Launcher, launch Vitis Classic.

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

Skip welcome page if shown.

Click on [+ Add] icon and select the custom extensible platform te0820_23_240_pfm[custom] in the directory:

```
~/work/te0820_023_240/te0820-rd_pfm/te0820_23_240_pfm/
export/te0820_23_240_pfm
```

Click Next.

In Domain window select by browse:

Sysroot path:

```
~/work/te0820_023_240/te0820-rd_pfm/
/sysroots/cortexa72-cortexa53-xilinx-linux
```

Root FS:

```
~/work/te0820_023_240/te0820-rd/
/os/petalinux/images/linux/rootfs.ext4
```

Kernel Image:

```
~/work/te0820_023_240/te0820-rd/os/petalinux/images/linux/Image
```

Click Next.

Select Native XRT Examples:

Select: Stream Free Running Kernel (XRT Native API)

Click Finish

New project template is created.

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

In “Explorer” section of Vitis Classic IDE, click on: free_run_xo_xrt_system [te0820_23_240_pfm] to select it.

Click on free_run_xo_xrt_kernels and delete inkrement kernel icon.

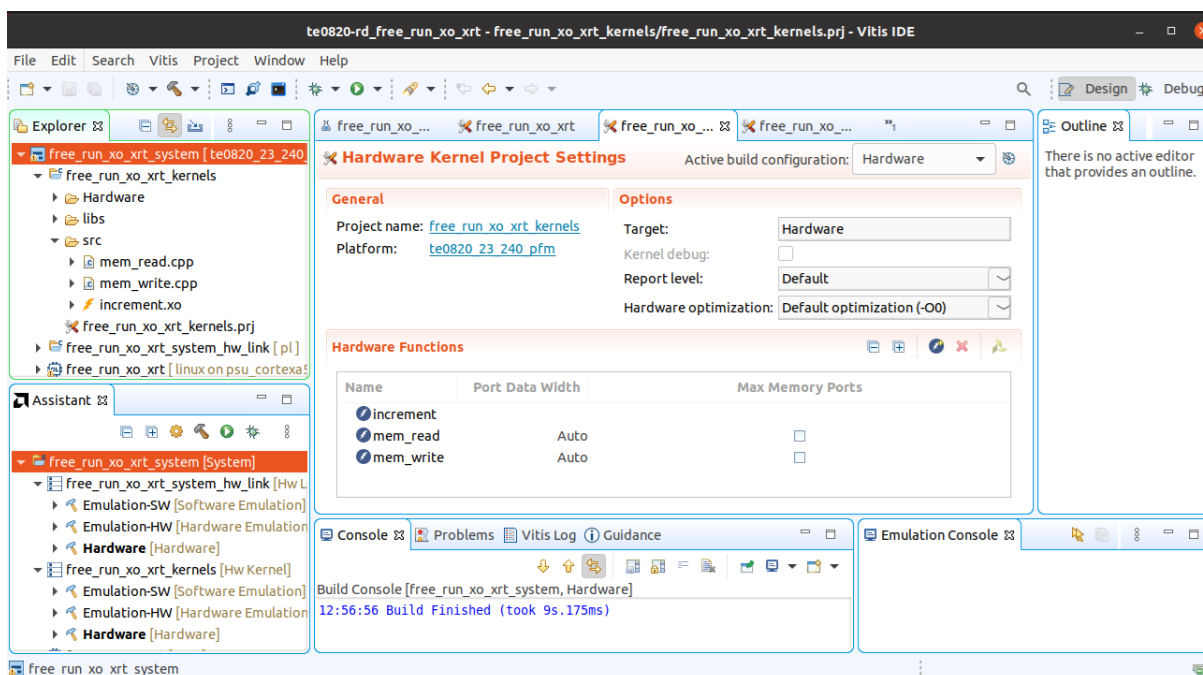
Delete file:

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt/  
free_run_xo_xrt_kernels/src/increment.cpp
```

Click on free_run_xo_xrt_kernels and import increment.xo kernel from the directory:

```
/home/devel/work/te0820_023_240/te0820-rd_free_run_xrt/  
free_run_xrt_kernels/Hardware/build
```

Click on free_run_xo_xrt_kernels and add the increment icon representing the imported increment.xo prebuild RTL kernel.



The RTL kernel increment.xo is imported to the Vitis Classic extensible project, now.

Right Click on: free_run_xrt_xo_system [te0820_23_240_pfm] and select: Build project

Vitis Classic will compile the project.

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

```
~/work/te0820_023_240/te0820-rd_free_run_xrt/free_run_xrt_system/  
Hardware/package/sd_card.img
```

Prepare for migration from Vitis Classic to Vitis Unified

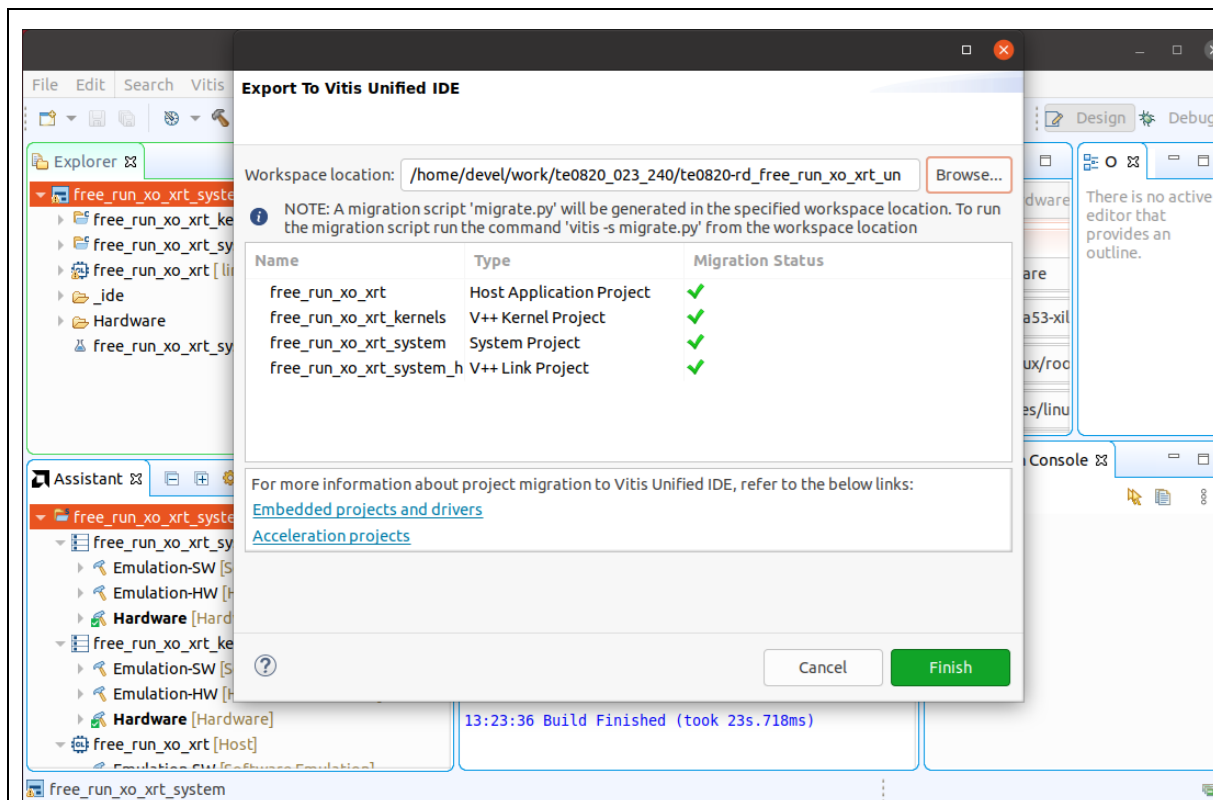
Create new empty directory `te0820-rd_free_run_xo_xrt_un`

```
~/work/te0820_023_240/te0820-rd/te0820-rd_free_run_xo_xrt_un
```

It will serve for test of the migration from the Vitis 2024.2 Classic Extensible workspace (with imported RTL object `increment.xo`) to Vitis 2024.2 Unified Extensible workspace.

In Vitis Classic GUI, select

Vitis → Export to Vitis Unified IDE



Select the intended Vitis Unified GUI workspace location.

```
~/work/te0820_023_240/te0820-rd/te0820-rd_free_run_xo_xrt_un
```

Migration python script `migrate.py` is created by Vitis Classic GUI to this workspace location:

```
~/work/te0820_023_240/te0820-rd/te0820-rd_free_run_xo_xrt_un/  
migrate.py
```

Close the Vitis Classic GUI, now.

8.5 Migrate Vitis Classic Extensible project to Vitis Unified

Current directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xrt
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt
```

Modify created migration python script migrate.py created in Vitis Classic in text editor.

```
~/work/te0820_023_240/ te0820-rd/te0820-rd_free_run_xo_xrt_un/
migrate.py
```

The path to the imported RTL increment.xo kernel needs to be changed.
Find line:

```
system_project.add_precompiled_kernel(xo_file_path = 'src/increment.xo',
containers = [container_name])
```

and replace src by the absolute path:

```
/home/devel/work/te0820_023_240/te0820-rd_free_run_xrt/
free_run_xrt_kernels/Hardware/build
```

The resulting line is:

```
system_project.add_precompiled_kernel(xo_file_path = '
/home/devel/work/te0820_023_240/te0820-rd_free_run_xrt/
free_run_xrt_kernels/Hardware/build/increment.xo', containers =
[container_name])
```

Change working directory:

```
$cd ~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

Start Vitis Unified GUI by:

```
vitis -w . &
```

Update the initial empty workspace and close the Vitis Unified GUI.

This step sets initial default environment required by the Vitis Unified 2024.2 GUI:

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/_ide
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/.Xil.py
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/.gitignore
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/migrate.py
```

At this stage, the Vitis Unified can be used for sourcing of the migration.py script:

```
vitis -s migrate.py
```

The migrated Vitis Unified project is created with this report to the console:

```
devel@ubuntu:~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un$ source
/tools/Xilinx/Vitis/2024.2/settings64.sh
```

```
devel@ubuntu:~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un$ vitis
-s migrate.py
```

```
***** Vitis Development Environment
```

```
***** Vitis v2024.2 (64-bit)
```

```
**** SW Build 5238363 on 2024-11-08-17:54:51
```

```
** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
```

```
** Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights
Reserved.
```

```
-----
Migrating classic Vitis projects to Vitis Unified IDE
-----
```

```
Vitis Server started on port '33925'.
```

```
Successfully created Vitis client on workspace
/home/devel/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
```

```
Setting user platform repositories...
```

```
Migrating projects from classic Vitis IDE
```

```
Creating application component 'free_run_xo_xrt'
```

```
Migrating build settings for the application project
'free_run_xo_xrt'...
```

```
NOTE: This script only provides migration of symbol definitions (-D),
include paths (-I), libraries (-l) and library paths (-L). Any other
compiler or linker settings can be set manually in the migrated
application's UserConfig.cmake file.
```

```
Migrating HW Kernel project : 'free_run_xo_xrt_kernels'
```

```
Creating HLS component 'mem_read_free_run_xo_xrt_kernels'
```

```
Migrating kernel 'mem_read' from project 'free_run_xo_xrt_kernels' as
HLS component
```

```

Migrating HW Kernel project : 'free_run_xo_xrt_kernels'
Creating HLS component 'mem_write_free_run_xo_xrt_kernels'
Migrating kernel 'mem_write' from project 'free_run_xo_xrt_kernels' as
HLS component
Skipping project 'free_run_xo_xrt_system_hw_link' ...
Creating system project 'free_run_xo_xrt_system'
Adding binary container 'krnl_incr' in project 'free_run_xo_xrt_system'
Shutting down Vitis server running on port '33925'
devel@ubuntu:~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un$

```

Listing of the sourcing of the migration.py script by the Vitis Unified.

```

devel@ubuntu: ~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
devel@ubuntu:~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
devel@ubuntu:~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un$ vitis -s migrate.py

***** Vitis Development Environment
***** Vitis v2024.2 (64-bit)
**** SW Build 5238363 on 2024-11-08-17:54:51
** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
** Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.

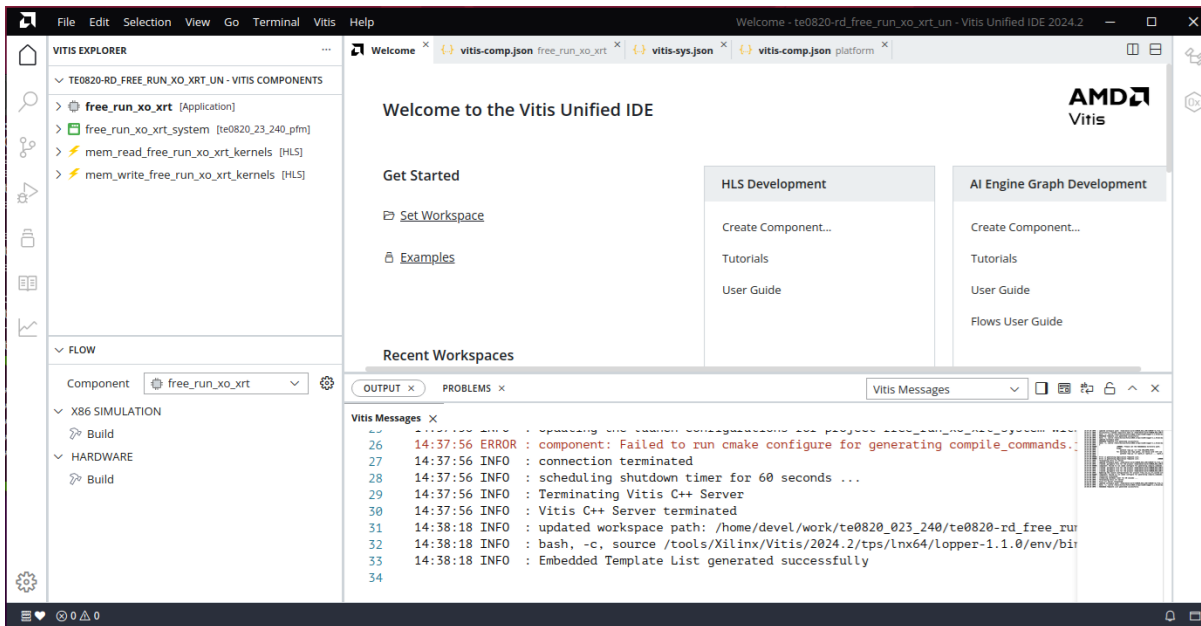
-----
Migrating classic Vitis projects to Vitis Unified IDE
-----
Vitis Server started on port '33925'.
Successfully created Vitis client on workspace /home/devel/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
Setting user platform repositories...
Migrating projects from classic Vitis IDE
Creating application component 'free_run_xo_xrt'
Migrating build settings for the application project 'free_run_xo_xrt'...
NOTE: This script only provides migration of symbol definitions (-D), include paths (-I), libraries (-l) and library paths (-L). Any other compiler or
linker settings can be set manually in the migrated application's UserConfig.cmake file.
Migrating HW Kernel project : 'free_run_xo_xrt_kernels'
Creating HLS component 'mem_read_free_run_xo_xrt_kernels'
Migrating kernel 'mem_read' from project 'free_run_xo_xrt_kernels' as HLS component
Migrating HW Kernel project : 'free_run_xo_xrt_kernels'
Creating HLS component 'mem_write_free_run_xo_xrt_kernels'
Migrating kernel 'mem_write' from project 'free_run_xo_xrt_kernels' as HLS component
Skipping project 'free_run_xo_xrt_system_hw_link' ...
Creating system project 'free_run_xo_xrt_system'
Adding binary container 'krnl_incr' in project 'free_run_xo_xrt_system'
Shutting down Vitis server running on port '33925'
devel@ubuntu:~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un$

```

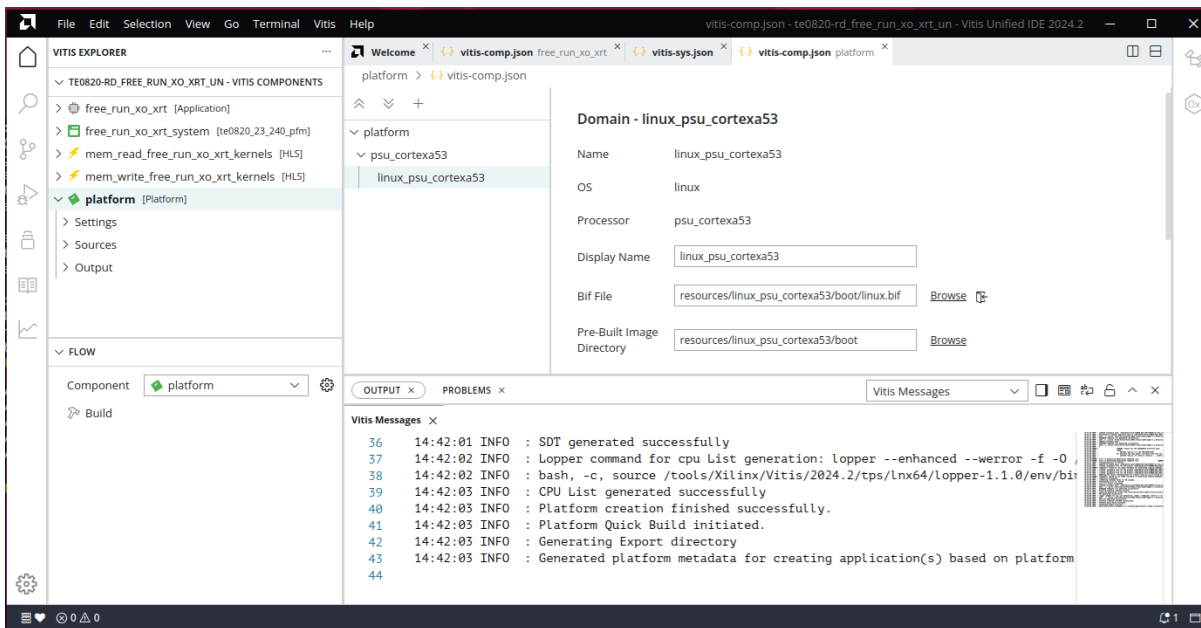
Start the Vitis Unified GUI, now:

```
$ vitis -w . &
```

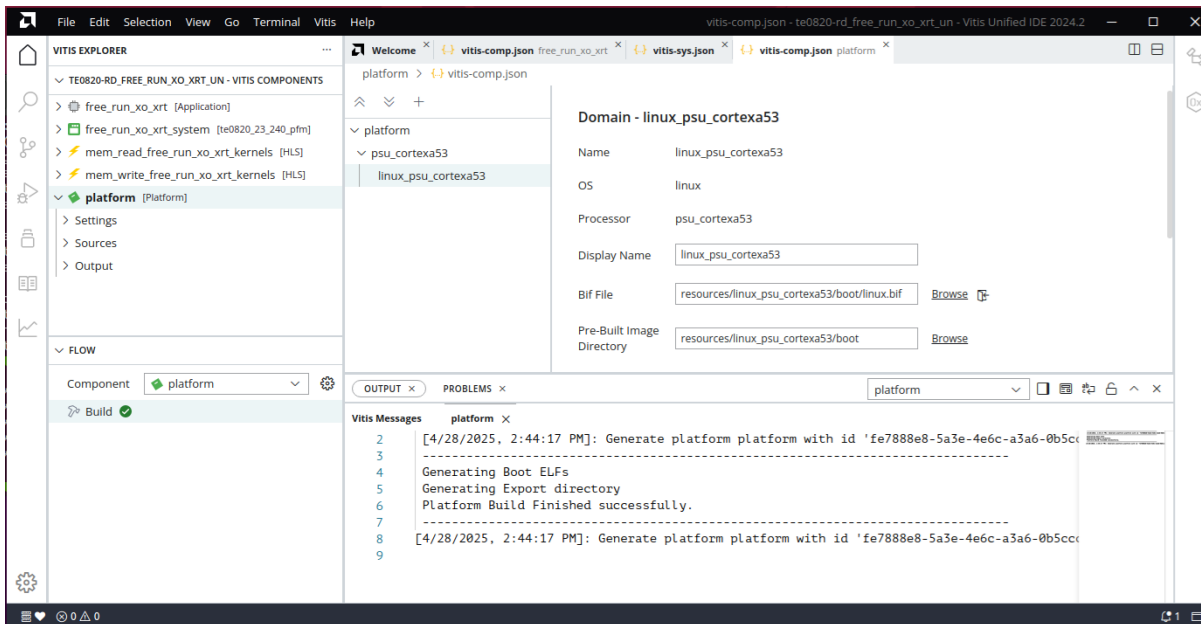
The migrated Vitis Unified workspace is opened:



Create local Vitis Unified Platform named platform from the existing Vitis Unified platform te0820_23_240_pfm_un.

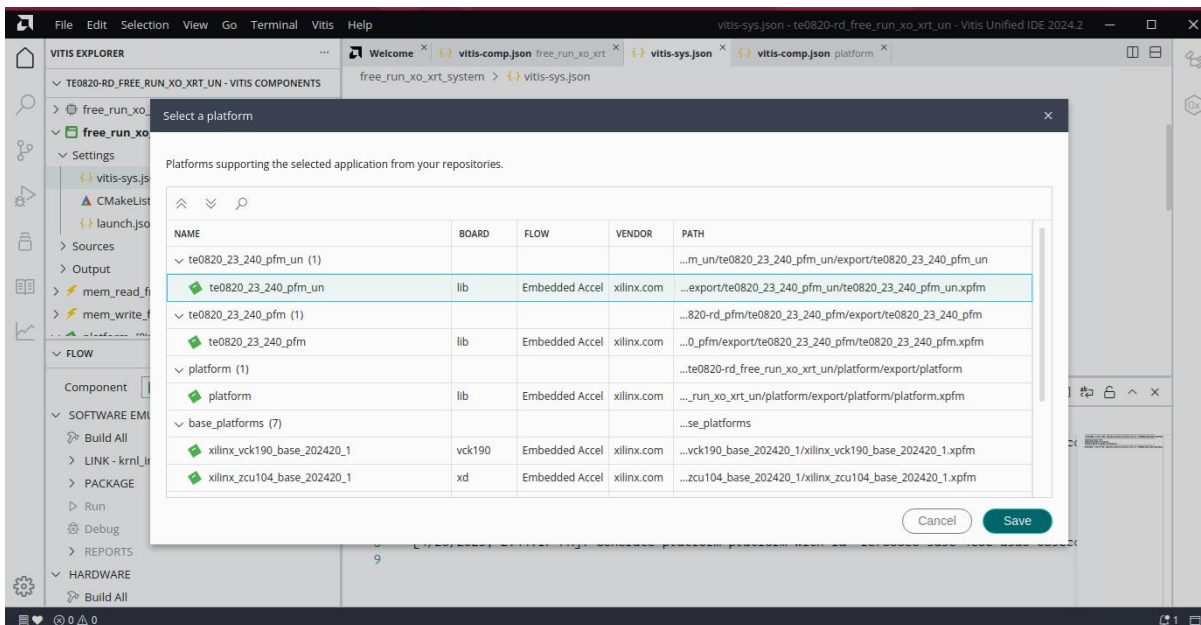


Build the created local Vitis Unified Platform named platform.



Select the free_run_xo_xrt_system component.

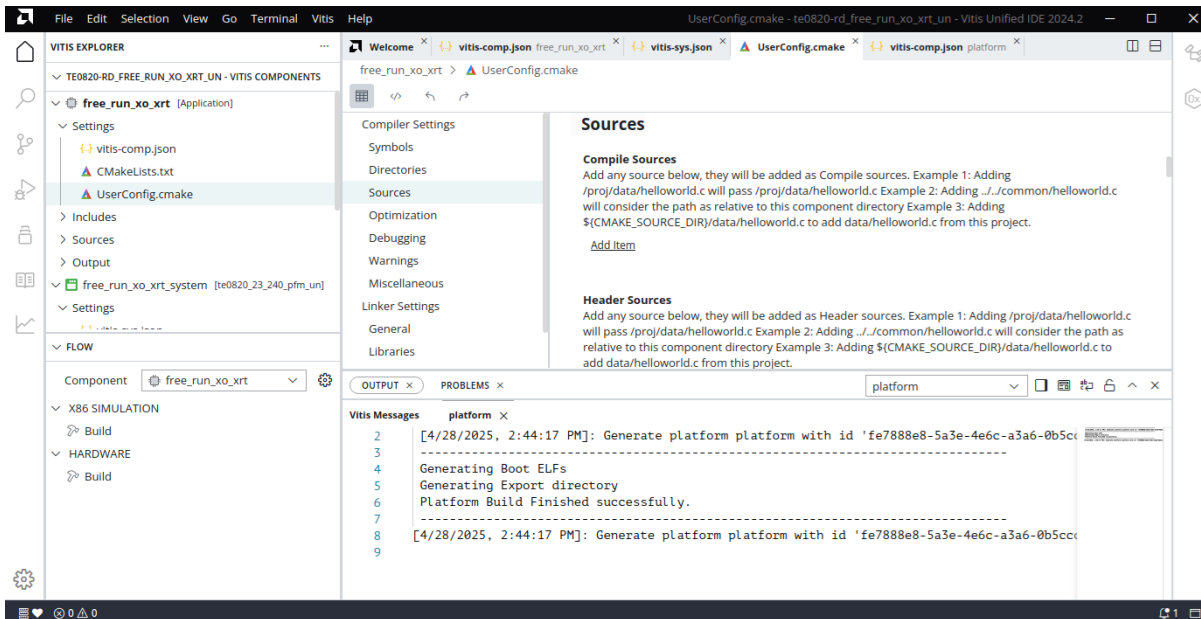
In submenu Settings click on the vitis-sys.json configuration and select the platform.
Select the Vitis Unified platform te0820_23_240_pfm_un.



Select the free_run_xo_xrt component.

In submenu Settings click on the UserConfig.cmake .

In Compiler Settings scroll to Sources.



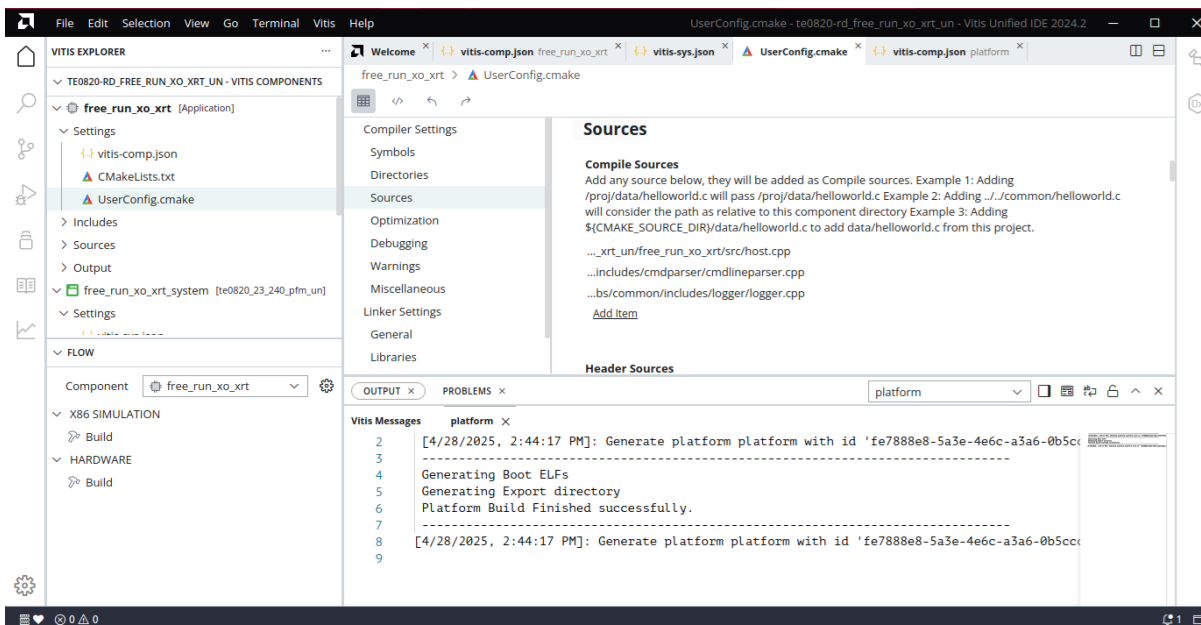
Add path to all three CPP files present in the project:

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/free_run_xo_xrt/src/host.cpp
```

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/free_run_xo_xrt/libs/common/includes/cmdparser/cmdlineparser.cpp
```

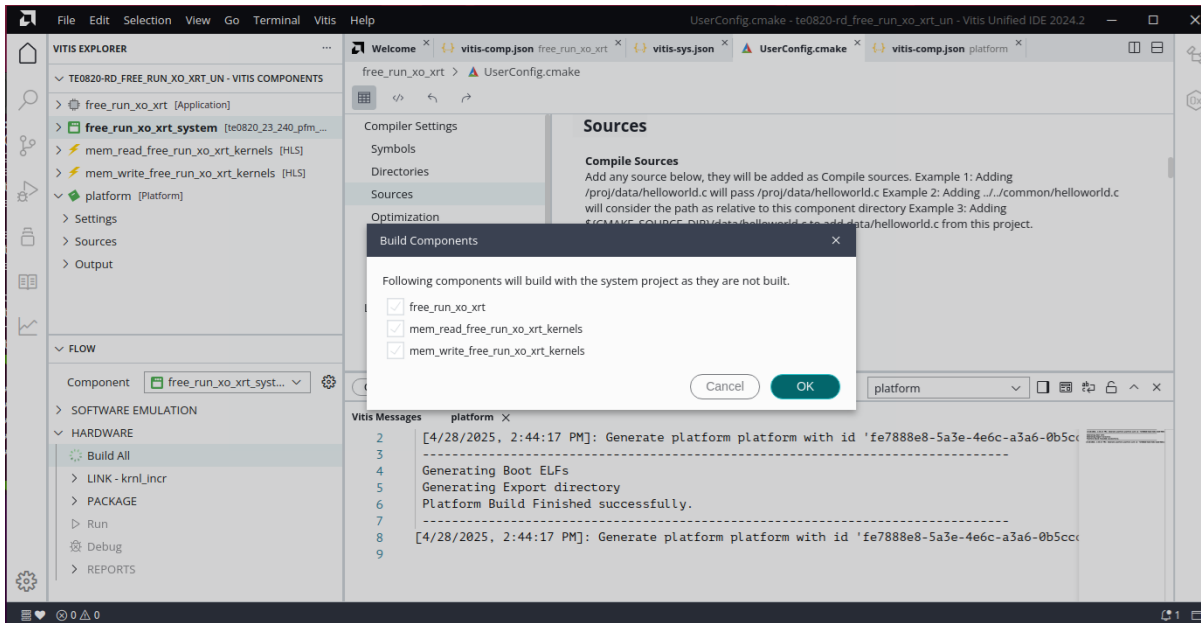
```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/free_run_xo_xrt/libs/common/includes/logger/logger.cpp
```

Paths to all three CPP files are defined, now:

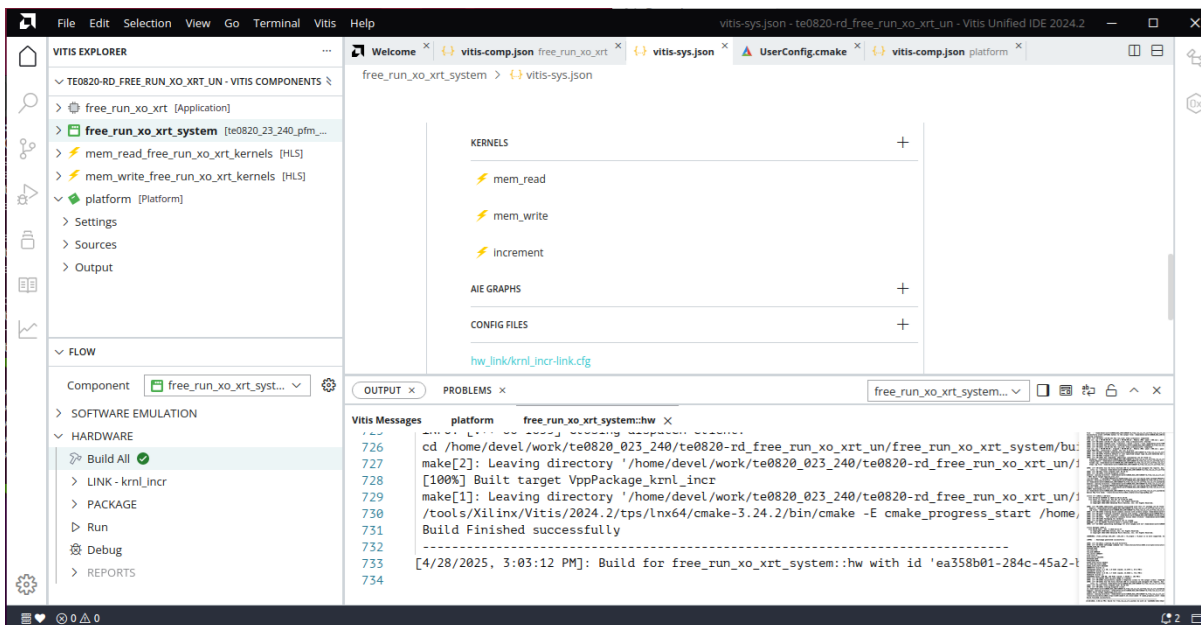


Select `free_run_xo_xrt_system`.

Click on `Build All` under `HARDWARE` to build the Vitis Unified project.



The Vitis Unified extensible project is build.



The SD card image is located in:

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/
free_run_xo_xrt_system/build/hw/package/package/sd_card.img
```

Write the SD card image `sd_card.img` to the SD card. In Windows Pro 10 (or Windows 11 Pro) PC, use program Win32DiskImager for this task.

Create SD card from the `sd_card.img` and start the evaluation board.

Change directory to

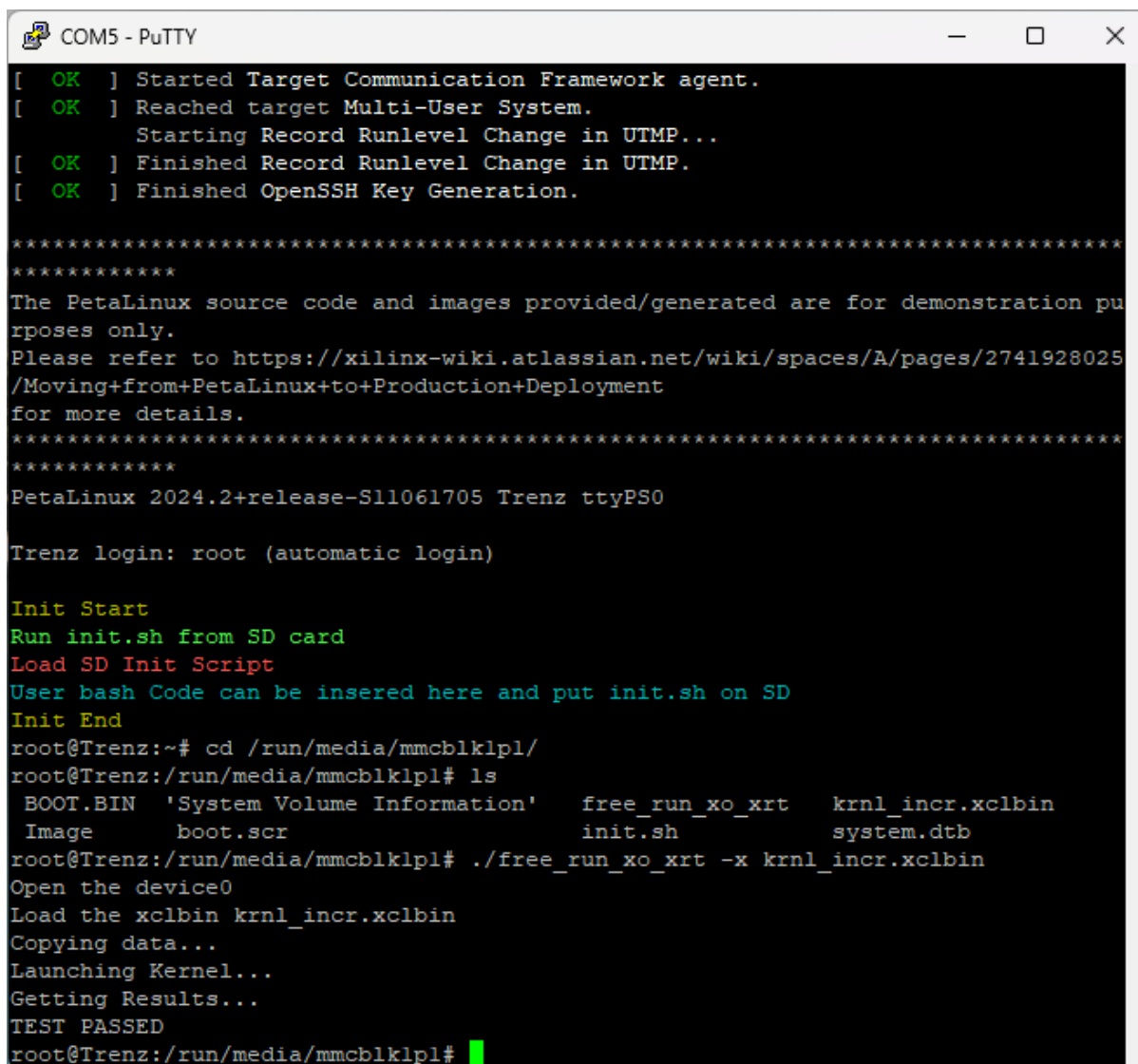
```
/run/media/mmcblk1p1/
```

Run the Migrated Free Run Application by the command:

```
./free_run_xo_xrt -x krnl_incr.xclbin
```

The application returns:

```
TEST PASSED
```



```
COM5 - PuTTY
[ OK ] Started Target Communication Framework agent.
[ OK ] Reached target Multi-User System.
       Starting Record Runlevel Change in UTMP...
[ OK ] Finished Record Runlevel Change in UTMP.
[ OK ] Finished OpenSSH Key Generation.

*****
*****
The PetaLinux source code and images provided/generated are for demonstration pu
rposes only.
Please refer to https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/2741928025
/Moving+from+PetaLinux+to+Production+Deployment
for more details.
*****
*****
PetaLinux 2024.2+release-S11061705 Trezn ttyPS0

Trezn login: root (automatic login)

Init Start
Run init.sh from SD card
Load SD Init Script
User bash Code can be insered here and put init.sh on SD
Init End
root@Trezn:~# cd /run/media/mmcblk1p1/
root@Trezn:/run/media/mmcblk1p1# ls
BOOT.BIN  'System Volume Information'  free_run_xo_xrt  krnl_incr.xclbin
Image     boot.scr                     init.sh          system.dtb
root@Trezn:/run/media/mmcblk1p1# ./free_run_xo_xrt -x krnl_incr.xclbin
Open the device0
Load the xclbin krnl_incr.xclbin
Copying data...
Launching Kernel...
Getting Results...
TEST PASSED
root@Trezn:/run/media/mmcblk1p1#
```

PetaLinux can be halted by typing halt in the terminal.

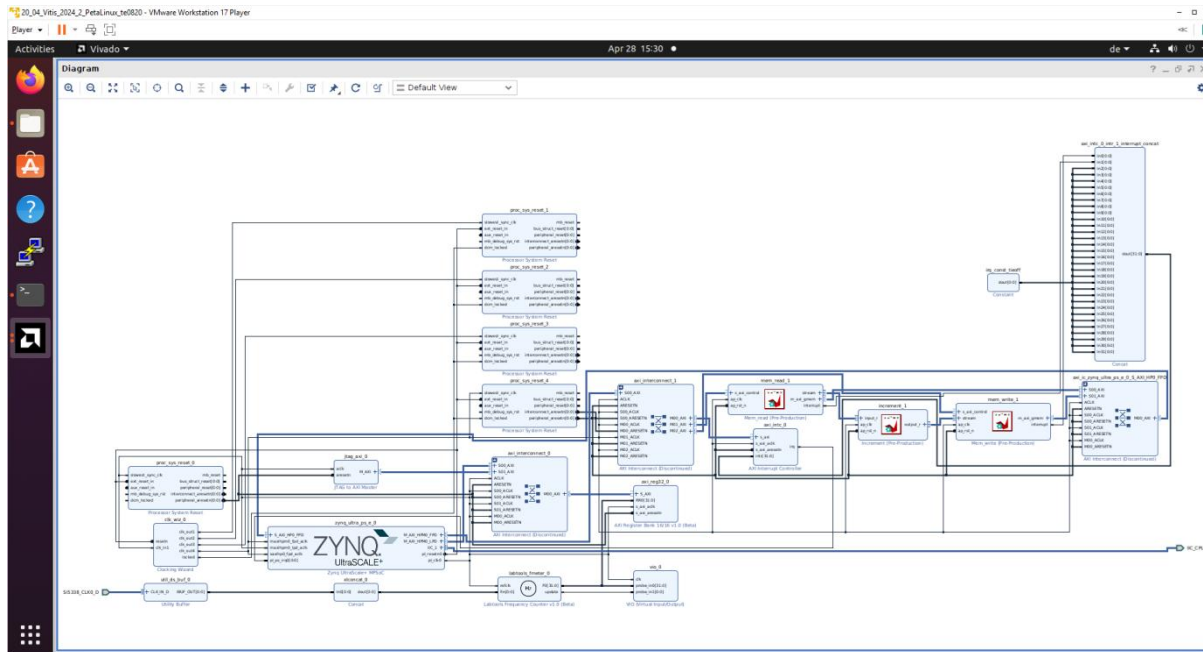
```
halt
```

Remove SD card, close the terminal and switch off the board.

The created block design can be open in Vivado project located in the directory:

```
/home/devel/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/  
free_run_xo_xrt_system/build/hw/hw_link/krnl_incr.build/link/  
vivado/vpl/prj/prj.xpr
```

The HW IP kernels are integrated with the PS host.



Kernels work with the default 240 MHz clock.

The Vitis Classic 2024.2 extensible workspace `te0820-rd_free_run_xo_xrt` with an external RTL kernel `increment.xo` has been migrated to the Vitis Unified 2024.2 extensible workspace. It has been compiled and tested on the evaluation board.

The external RTL kernel `increment.xo` was compiled from the CPP source code in another Vitis Classic 2024.2 extensible workspace `te0820-rd_free_run_xrt`.

9 Export and Import of Vitis Unified 2024.2 Extensible Workspace

Vitis Unified 2024.2 projects can be exported into a .zip file and imported in another directory. This will be demonstrated for the Vitis Unified project located in the directory:

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
```

Change working directory:

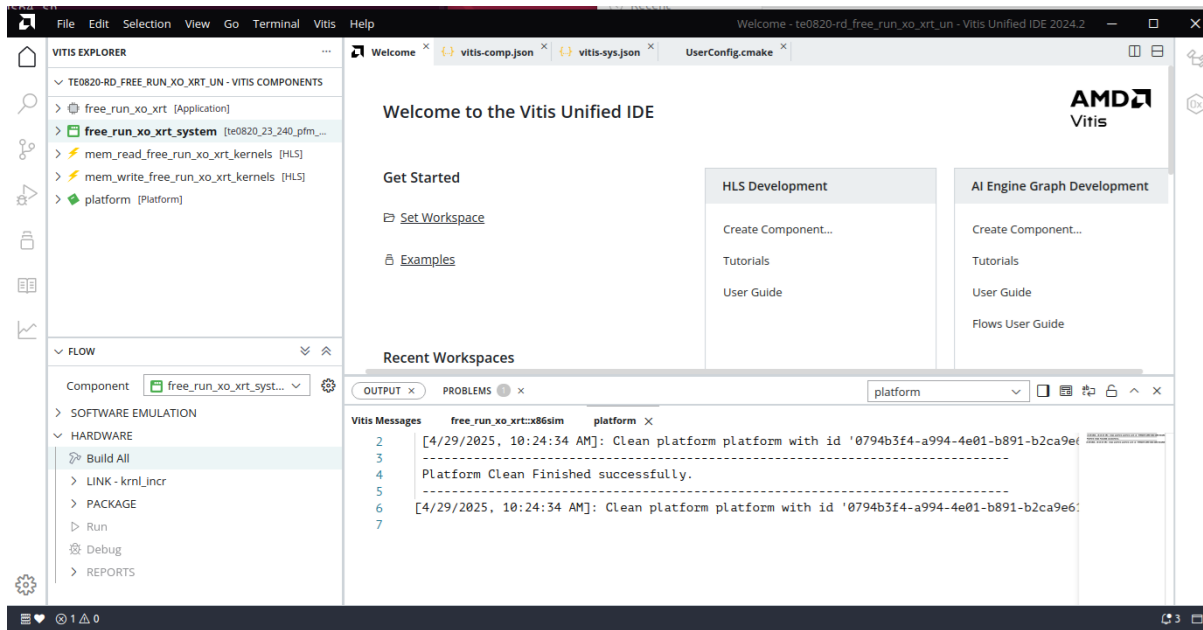
```
$cd ~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

Start Vitis Unified GUI by:

```
vitis -w . &
```



Select and click on Clean build icon to reduce size of the exported zip file:

free_run_xo_xrt_system → Clean build

free_run_xo_xrt → Clean build

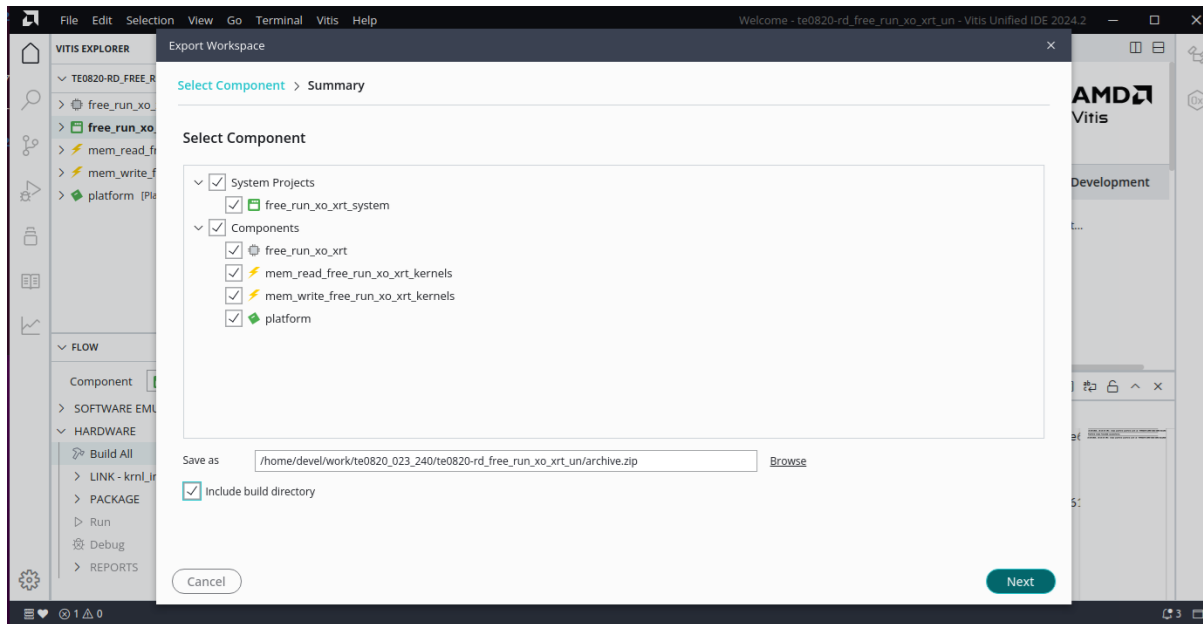
platform → Clean build

Export

Select

File → Export Workspace

Select the Include build directory box (it is unselected by default). It is needed to export precompiled RTL objects:



Click on:
Next → Finish

The Exported Workspace archive archive.zip is generated in:

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/archive.zip
```

Import

Create new directory `te0820-rd_free_run_xo_xrt_un_2` to test the Import of the Vitis Unified 2024.2 extensible workspace from the exported archive:

```
~/work/te0820_023_240/ te0820-rd/te0820-rd_free_run_xo_xrt_un_2
```

Current directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un_2

~/work/te0820_023_240/te0820-rd_free_run_xrt
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt
```

Change working directory:

```
$ cd ~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un_2
```

In Ubuntu terminal, execute script enabling access to Vitis 2024.2 tools.

```
$ source /tools/Xilinx/Vitis/2024.2/settings64.sh
```

Start Vitis Unified GUI by:

```
vitis -w . &
```

Select Import Workspace

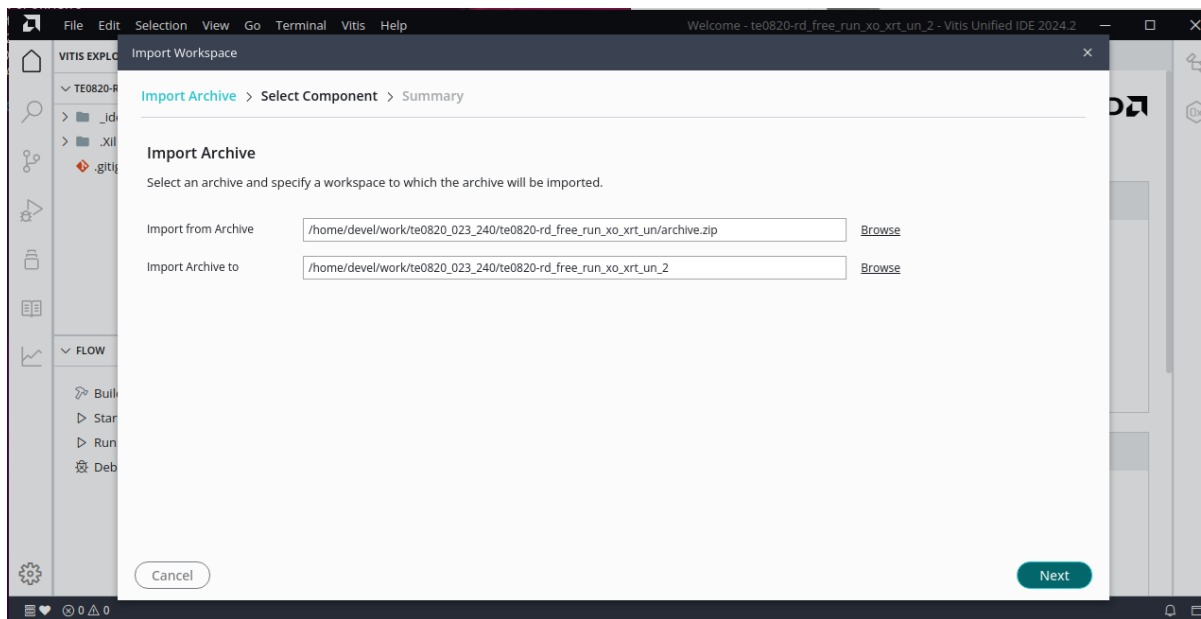
File → Import

Select the exported archive.zip file:

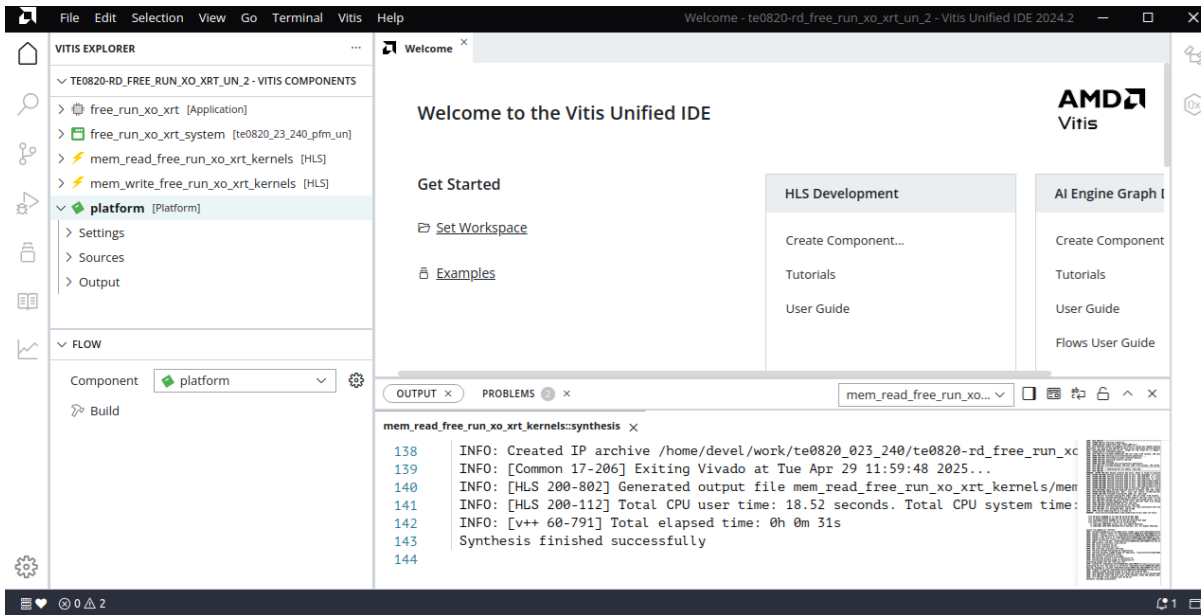
```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un/archive.zip
```

To start the Import of the Workspace, click on:

Next → Next → Finish



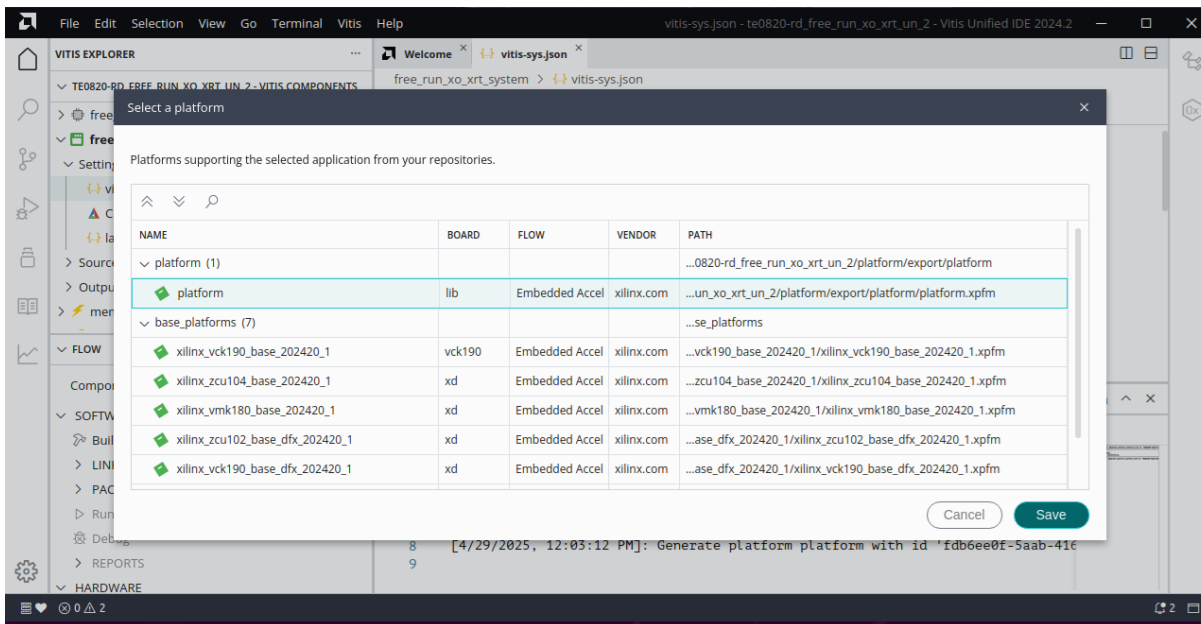
Vitis Unified 2024,2 extensible workspace is imported:



Select platform component.
Click on the Build icon.

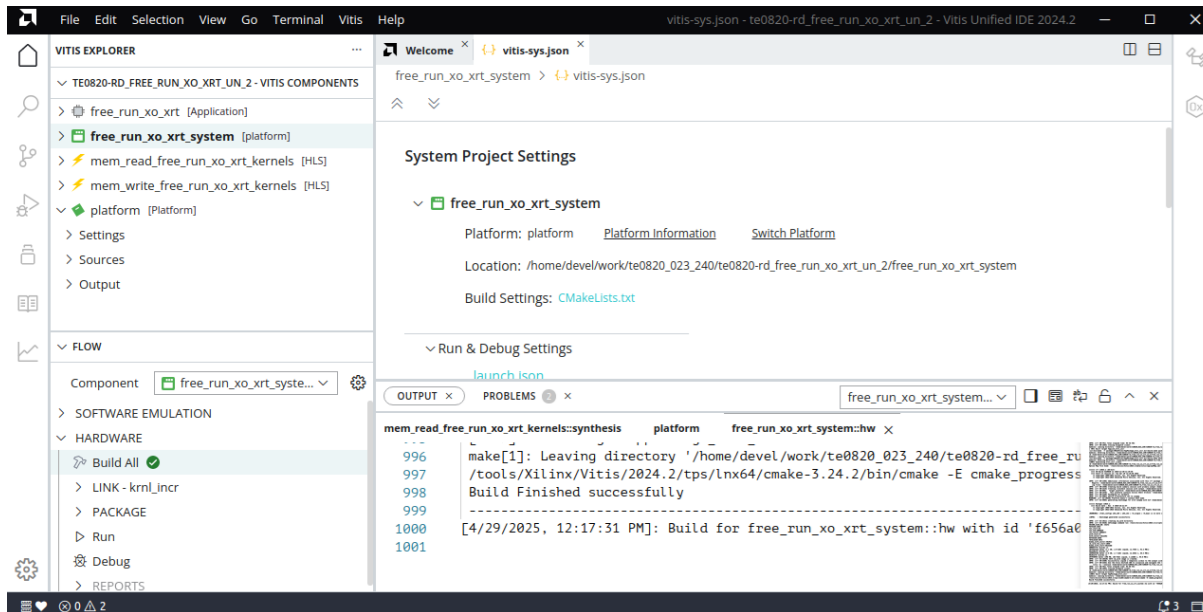
Select free_run_xo_xrt_system component.
Click on Settings → vitis-sys.json to display it.

Select local platform and click on Save.



Select free_run_xo_xrt_system component.
Click on Build icon.

The Imported Vitis Unified extensible project is compiled.



Resulting SD card image can be found in the directory:

```
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un_2/
free_run_xo_xrt_system/build/hw/package/package/sd_card.img
```

It can be tested on the board.

10 Final Summary

Final Vitis Unified extensible workspace directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_pfm_un
~/work/te0820_023_240/te0820-rd_free_run_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt_un_2
```

- te0820-rd Extensible .XSA archive, Compiled Petalinux files, PetaLinux image and filesystem.
- te0820-rd_pfm Vitis Classic extensible te0820-rd_pfm platform, sysroot, boot files, sd_card files
- te0820-rd_pfm_un Vitis Unified extensible te0820-rd_pfm_un platform, boot files, sd_card files
- te0820-rd_free_run_xrt_un free_run_xrt_un app. created from Vitis Unified extensible template application
- te0820-rd_free_run_xo_xrt_un free_run_xo_xrt_un app. migrated from Vitis Classic workspace to Vitis Unified workspace

- te0820-rd_free_run_xo_xrt_un_2 free_run_xrt_un app exported from Vitis Unified workspace to archive.zip and imported to new Vitis Unified workspace in a diferent directory.

Final Vitis Classic extensible workspace directory structure:

```
~/work/te0820_023_240/te0820-rd
~/work/te0820_023_240/te0820-rd_pfm
~/work/te0820_023_240/te0820-rd_free_run_xrt
~/work/te0820_023_240/te0820-rd_free_run_xo_xrt
```

- te0820-rd Extensible HW .XSA archive, Compiled Petalinux files, PetaLinux image and filesystem.
- te0820-rd_pfm Vitis Classic Extensible te0820-rd_pfm platform, sysroot, boot files, sd_card files.
- te0820-rd_free_run_xrt free_run_xrt_un app. created from Vitis Classic template
- te0820-rd_free_run_xo_xrt free_run_xo_xrt_un app created from Vitis Classic template, kernel increment is inported as a prebuild RTL increment.xo object from te0820-rd_free_run_xrt Vitis Classic workspace. Project demonstrate migration from Vitis Classic workspace to Vitis Unified workspace te0820-rd_free_run_xo_xrt_un .

11 Remote Monitoring and Configuration Support

The configured OS includes 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.

12 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