Application Note

Xilinx Vitis AI ‘facedetect’ and ‘resnet50’ Demo on Trenz Electronic TE0821-01-2cg-4GB SoM + TE0706-3 Carrier

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

Revision history

Rev.	Date	Author	Description
01	28.11.2022	Z.P.	Document creation
02	18.1.2023	Z.P.	Fixed path to Vitis AI Library facedetect demo, typo correction
03	20.03.2023	Z.P.	Fixed error in paths and commandline appearance inside docker in Appendix I
04	29.03.2023	Z.P.	Fixed step where docker installation is requested.

Contents

Acknowledgement

Acknowledgement to the project StorAIge and the corresponding Czech institutional support project No. 8A21009.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007321. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and France, Belgium, Czech Republic, Germany, Italy, Sweden, Switzerland, Turkey.

1 Description

This document provides tutorial how to setup and run Vitis AI 2.0 demo ‘facedetect’ and ‘resnet50’ on Trenz TE0821-01-2cg-4GB SoM assembled to TE0706 carrier board. The DPU unit used in zcu104 or zcu102 doesn’t fit into the used SoM. For that reason, the reduced DPU configuration is provided as well as recompiled models for demos.

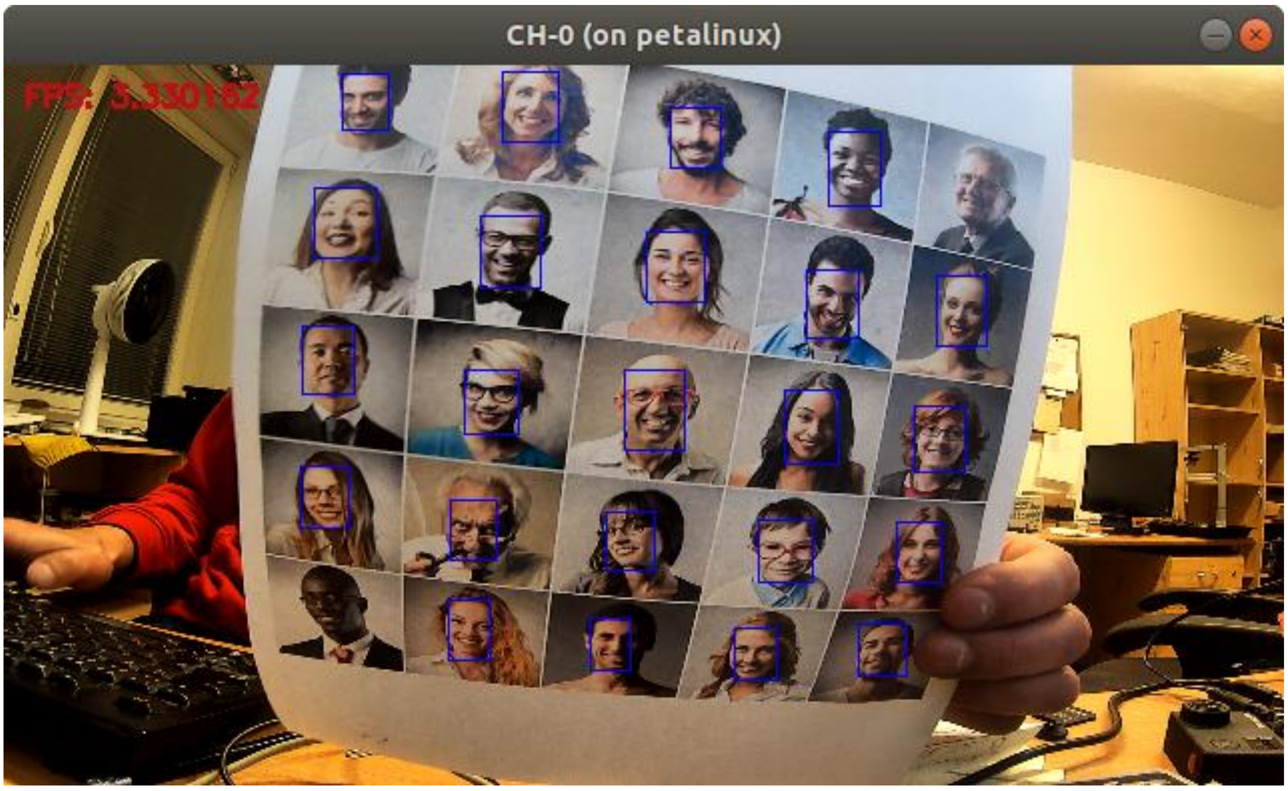


Figure 1: Example output from ‘facedetect’ demo

2 Requirements

1. Hardware:
    a. Trenz TE0821-01-2cg-4GB SoM installed on TE0706 and power source.
    b. USB webcam with USB cable, tested with See3CAM_CU30 - 3.4 Mpix Low Light USB Camera (Color).
    c. Ethernet cable.
2. Software:
    a. Before following this guide it is required to go through “TE0821 test board Vitis AI Tutorial” [1] from start with module “1” up to “Test 2: Run Vector Addition” section.

3 Test 3: Vitis-AI demo

1. Create new directory **test_board_dpu_trd**:

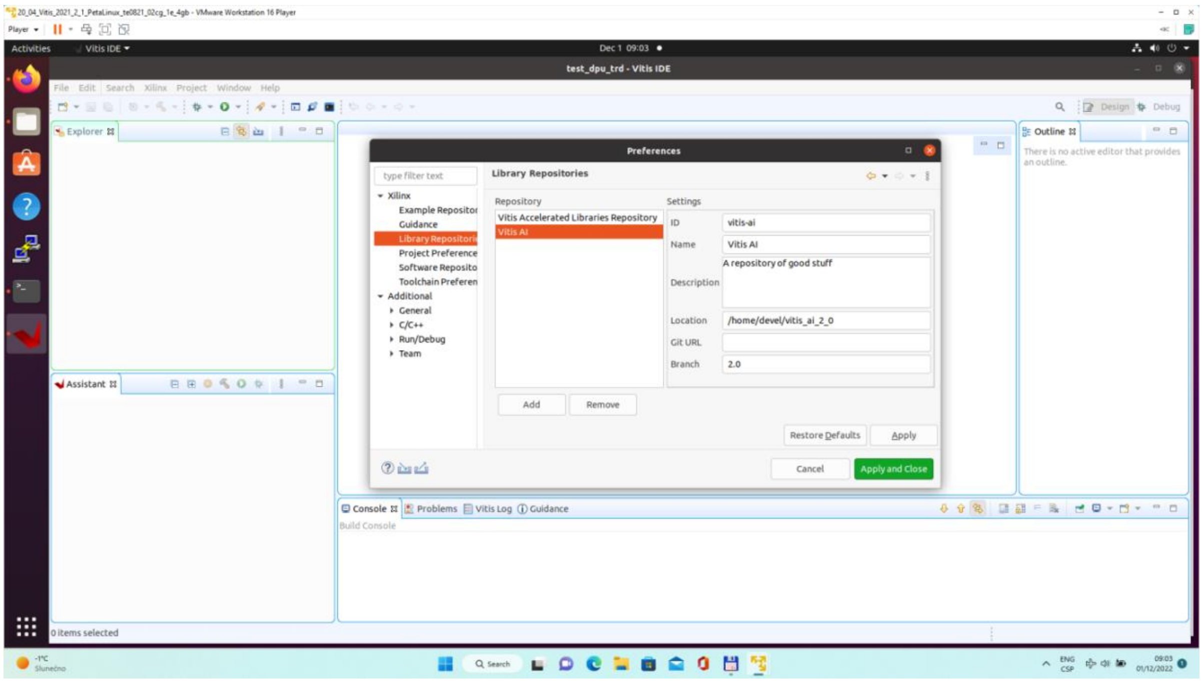
```
$ mkdir -p ~/work/TE0821_01_240/test_board_dpu_trd
$ cd ~/work/TE0821_01_240/test_board_dpu_trd
```

2. In Ubuntu terminal, start Vitis by:

```
$ vitis &
```

   In Vitis IDE Launcher, select your working directory
   ~/work/te0808_24_240/StarterKit_dpu_trd
   Click on Launch to launch Vitis.

3. Add Vitis-AI Repository to Vitis:
   a. Open menu **Window** → **Preferences**
      b. Go to **Library Repositories** tab
      c. Add Vitis-AI by clicking **Add** button and fill the form as shown below, use absolute path to your home folder in field **"Location"**:



Repository: Vitis Accelerated Libraries Repository; Vitis AI

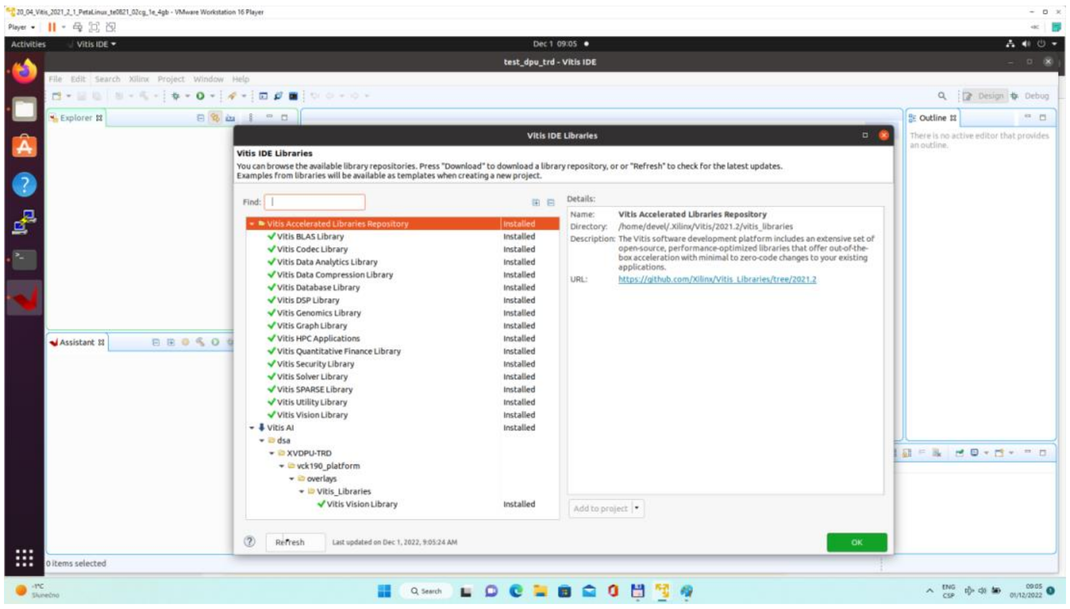
Settings:
ID: vitis-ai
Name: Vitis AI
Description: A repository of good stuff
Location: /home/devel/vitis_ai_2_0
Git URL:
Branch: 2.0

      d. Click **Apply and Close**.

   INFO: Field "Location" says that the Vitis-AI repository from github has been cloned into ~/vitis_ai_2_0 folder, already in the stage of Petalinux configuration in previous tutorial. 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.

      e. Check Correctly added library, it appears in Libraries:
         Open menu **Xilinx** → **Libraries...**

You can find there just added **Vitis-AI** library marked as **"Installed"**.



Vitis IDE Libraries

You can browse the available library repositories. Press "Download" to download a library repository, or or "Refresh" to check for the latest updates.
Examples from libraries will be available as templates when creating a new project.

Find:

Vitis Accelerated Libraries Repository	Installed
Vitis BLAS Library	Installed
Vitis Codec Library	Installed
Vitis Data Analytics Library	Installed
Vitis Data Compression Library	Installed
Vitis Database Library	Installed
Vitis DSP Library	Installed
Vitis Genomics Library	Installed
Vitis Graph Library	Installed
Vitis HPC Applications	Installed
Vitis Quantitative Finance Library	Installed
Vitis Security Library	Installed
Vitis Solver Library	Installed
Vitis SPARSE Library	Installed
Vitis Utility Library	Installed
Vitis Vision Library	Installed
Vitis AI	Installed
dsa	
XVDPU-TRD	
vck190_platform	
overlays	
Vitis_Libraries	
Vitis Vision Library	Installed

Details:

Name: Vitis Accelerated Libraries Repository
Directory: /home/devel/.Xilinx/Vitis/2021.2/vitis_libraries
Description: The Vitis software development platform includes an extensive set of open-source, performance-optimized libraries that offer out-of-the-box acceleration with minimal to zero-code changes to your existing applications.
URL: https://github.com/Xilinx/Vitis_Libraries/tree/2021.2

Add to project

Refresh Last updated on Dec 1, 2022, 9:05:24 AM OK

4. Create a Vitis-AI Design for our TE0821_01_240 custom platform
   f. Select **File -> New -> Application project**. Click Next.

   g. Skip welcome page if it is shown.

   h. Click on **"+ Add"** icon and select the custom extensible platform **TE0821_01_240_pfm[custom]** in the directory: ~/work/TE0821_01_240/test_board_pfm/TE0821_01_240_pfm/export/TE0821_01_240_pfm

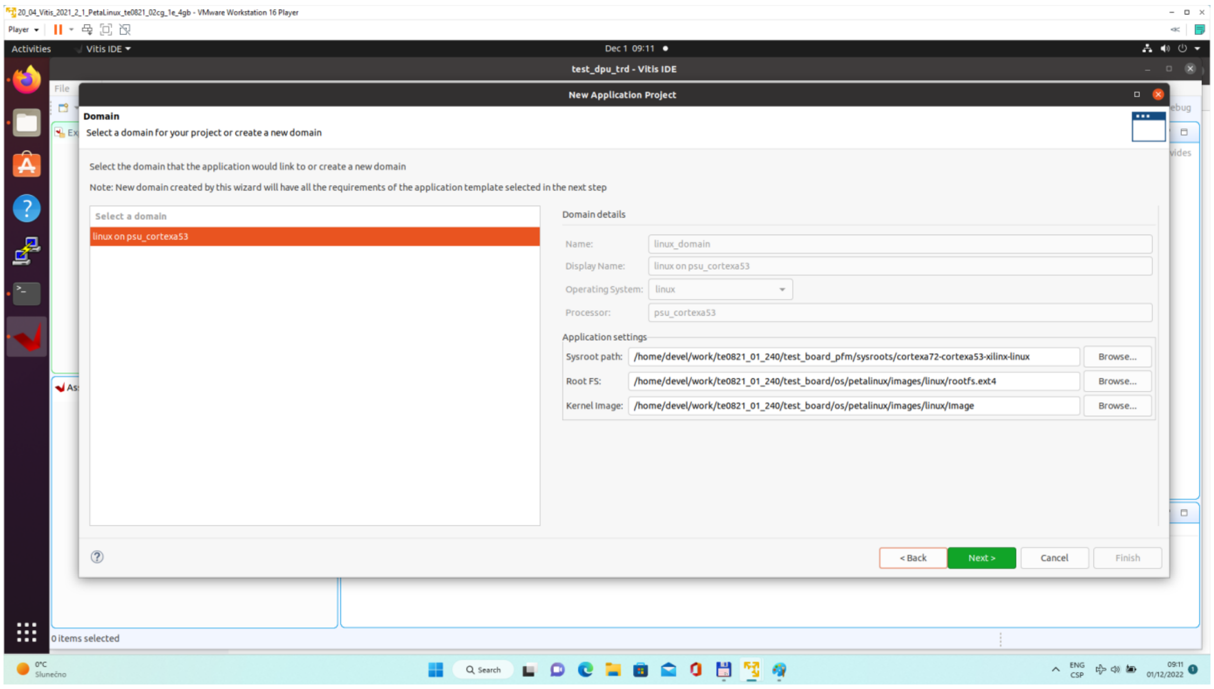
   i. Click Next.
   j. In "Application Project Details" window type into **Application project name**:

      dpu_trd

   k. Click Next.
   l. In "**Domain window**" type (or select by browse):

Field	Value	Description
"Sysroot path"	~/work/TE0821_01_240/test_board_pfm/sysroots/cortexa72-cortexa53-xilinx-linux	Petalinux compilation outputs from Trenz Vitis AI Tutorial
"Root FS"	~/work/TE0821_01_240/test_board/os/petalinux/images/linux/rootfs.ext4	
"Kernel Image"	~/work/TE0821_01_240/test_board/os/petalinux/images/linux/Image	

   m. Click Next.



Domain

Select a domain for your project or create a new domain

Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

Select a domain: linux on psu_cortexa53

Domain details

Name: linux_domain

Display Name: linux on psu_cortexa53

Operating System: linux

Processor: psu_cortexa53

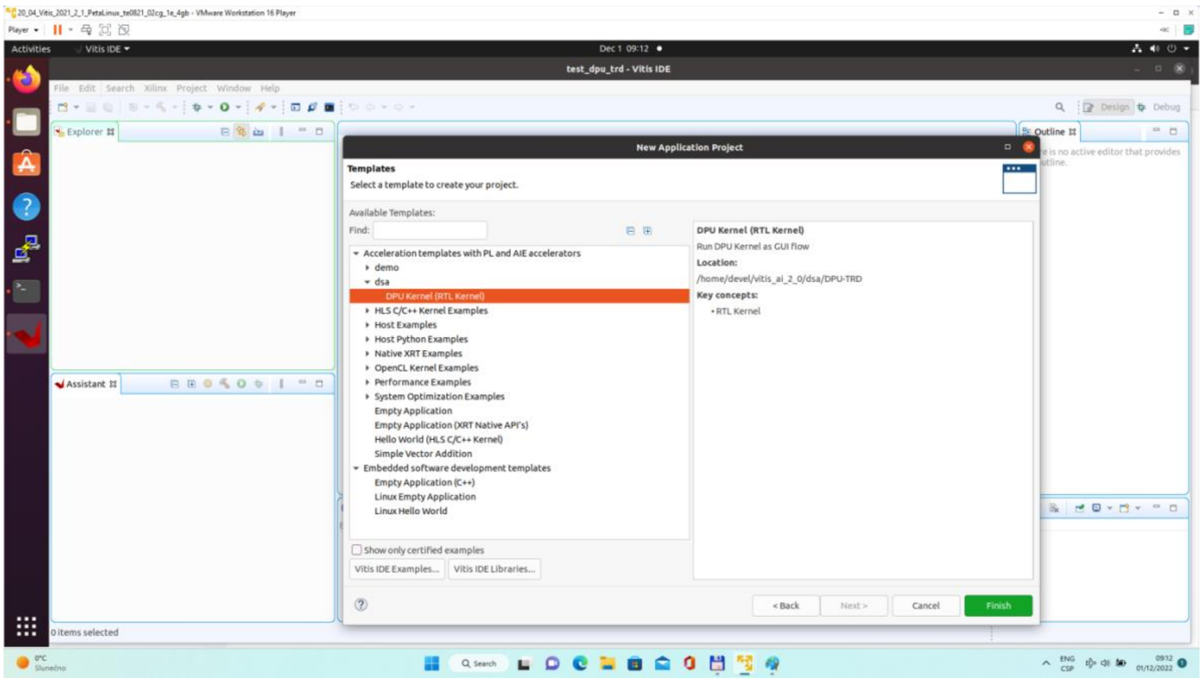
Application settings

Sysroot path: /home/devel/work/te0821_01_240/test_board_pfm/sysroots/cortexa72-cortexa53-xilinx-linux

Root FS: /home/devel/work/te0821_01_240/test_board/os/petalinux/images/linux/rootfs.ext4

Kernel Image: /home/devel/work/te0821_01_240/test_board/os/petalinux/images/linux/Image

n. In dsa folder, select: “**DPU Kernel (RTL Kernel)**”



Templates

Select a template to create your project.

Available Templates:

- Acceleration templates with PL and AIE accelerators
  - demo
  - dsa
    - DPU Kernel (RTL Kernel)
  - HLS C/C++ Kernel Examples
  - Host Examples
  - Host Python Examples
  - Native XRT Examples
  - OpenCL Kernel Examples
  - Performance Examples
  - System Optimization Examples
  - Empty Application
  - Empty Application (XRT Native API's)
  - Hello World (HLS C/C++ Kernel)
  - Simple Vector Addition
- Embedded software development templates
  - Empty Application (C++)
  - Linux Empty Application
  - Linux Hello World

Show only certified examples [ ]

DPU Kernel (RTL Kernel)

Run DPU Kernel as GUI flow

Location:

/home/devel/vitis_ai_2_0/dsa/DPU-TRD

Key concepts:

- RTL Kernel

o. Click **Finish**

New project template is created.

5. Configure project and DPU:
    p. In dpu_trd window menu **“Active build configuration”** switch from “SW Emulation” to **“Hardware”**.

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

    q. Remove **dpu_conf.vh** file and replace it by the one enclosed to this appnote:

```
//Setting the arch of DPU, For more details, Please read the PG338


/*====== Architecture Options ======*/
// |------------------------------------------------------|
// | Support 8 DPU size
// | It relates to model. if change, must update model
// +-------------------------------------------------------+
// | `define B512
// +-------------------------------------------------------+
// | `define B800
// +-------------------------------------------------------+
// | `define B1024
// +-------------------------------------------------------+
// | `define B1152
// +-------------------------------------------------------+
// | `define B1600
// +-------------------------------------------------------+
// | `define B2304
// +-------------------------------------------------------+
// | `define B3136
// +-------------------------------------------------------+
// | `define B4096
// |------------------------------------------------------|

`define B1600

// |------------------------------------------------------|
// | If the FPGA has Uram. You can define URAM_EN parameter
// | if change, Don't need update model
// +-------------------------------------------------------+
// | for zcu104 : `define URAM_ENABLE
// +-------------------------------------------------------+
// | for zcu102 : `define URAM_DISABLE
// |------------------------------------------------------|

`define URAM_DISABLE

//config URAM
`ifdef URAM_ENABLE
`define def_UBANK_IMG_N          5
`define def_UBANK_WGT_N          17
`define def_UBANK_BIAS           1
`elsif URAM_DISABLE
`define def_UBANK_IMG_N          0
`define def_UBANK_WGT_N          0
`define def_UBANK_BIAS           0
`endif

// |------------------------------------------------------|
// | You can use DRAM if FPGA has extra LUTs
// | if change, Don't need update model
// +-------------------------------------------------------+
// | Enable DRAM  : `define DRAM_ENABLE
// +-------------------------------------------------------+
// | Disable DRAM : `define DRAM_DISABLE
// |------------------------------------------------------|
`define DRAM_DISABLE

//config DRAM
`ifdef DRAM_ENABLE
`define def_DBANK_IMG_N          1
`define def_DBANK_WGT_N          1
`define def_DBANK_BIAS           1
`elsif DRAM_DISABLE
`define def_DBANK_IMG_N          0
`define def_DBANK_WGT_N          0
`define def_DBANK_BIAS           0
`endif

// |-----------------------------------------------------|
// | RAM Usage Configuration
// | It relates to model. if change, must update model
// +-----------------------------------------------------+
// | RAM Usage High : `define RAM_USAGE_HIGH
// +-----------------------------------------------------+
// | RAM Usage Low  : `define RAM_USAGE_LOW
// |-----------------------------------------------------|

`define RAM_USAGE_LOW

// |-----------------------------------------------------|
// | Channel Augmentation Configuration
// | It relates to model. if change, must update model
// +-----------------------------------------------------+
// | Enable  : `define CHANNEL_AUGMENTATION_ENABLE
// +-----------------------------------------------------+
// | Disable : `define CHANNEL_AUGMENTATION_DISABLE
// |-----------------------------------------------------|

`define CHANNEL_AUGMENTATION_DISABLE

// |-----------------------------------------------------|
// | DepthWiseConv Configuration
// | It relates to model. if change, must update model
// +-----------------------------------------------------+
// | Enable  : `define DWCV_ENABLE
// +-----------------------------------------------------+
// | Disable : `define DWCV_DISABLE
// |-----------------------------------------------------|

`define DWCV_DISABLE

// |-----------------------------------------------------|
// | Pool Average Configuration
// | It relates to model. if change, must update model
// +-----------------------------------------------------+
// | Enable  : `define POOL_AVG_ENABLE
// +-----------------------------------------------------+
// | Disable : `define POOL_AVG_DISABLE
// |-----------------------------------------------------|

`define POOL_AVG_ENABLE

// |-----------------------------------------------------|
// | support multiplication of two feature maps
// | It relates to model. if change, must update model
// +-----------------------------------------------------+
// | Enable  : `define ELEW_MULT_ENABLE
// +-----------------------------------------------------+
// | Disable : `define ELEW_MULT_DISABLE
// |-----------------------------------------------------|

`define ELEW_MULT_DISABLE

// +-----------------------------------------------------+
// | RELU Type Configuration
// | It relates to model. if change, must update model
// +-------------------------------------------------------+
// | `define RELU_RELU6
// +-------------------------------------------------------+
// | `define RELU_LEAKYRELU_RELU6
// |-------------------------------------------------------|

`define RELU_RELU6

// |-------------------------------------------------------|
// | DSP48 Usage Configuration
// | Use dsp replace of lut in conv operate
// | if change, Don't need update model
// +-------------------------------------------------------+
// | `define DSP48_USAGE_HIGH
// +-------------------------------------------------------+
// | `define DSP48_USAGE_LOW
// |-------------------------------------------------------|

`define DSP48_USAGE_LOW

// |-------------------------------------------------------|
// | Power Configuration
// | if change, Don't need update model
// +-------------------------------------------------------+
// | `define LOWPOWER_ENABLE
// +-------------------------------------------------------+
// | `define LOWPOWER_DISABLE
// |-------------------------------------------------------|

`define LOWPOWER_DISABLE

// |-------------------------------------------------------|
// | DEVICE Configuration
// | if change, Don't need update model
// +-------------------------------------------------------+
// | `define MPSOC
// +-------------------------------------------------------+
// | `define ZYNQ7000
// |-------------------------------------------------------|

`define MPSOC
```

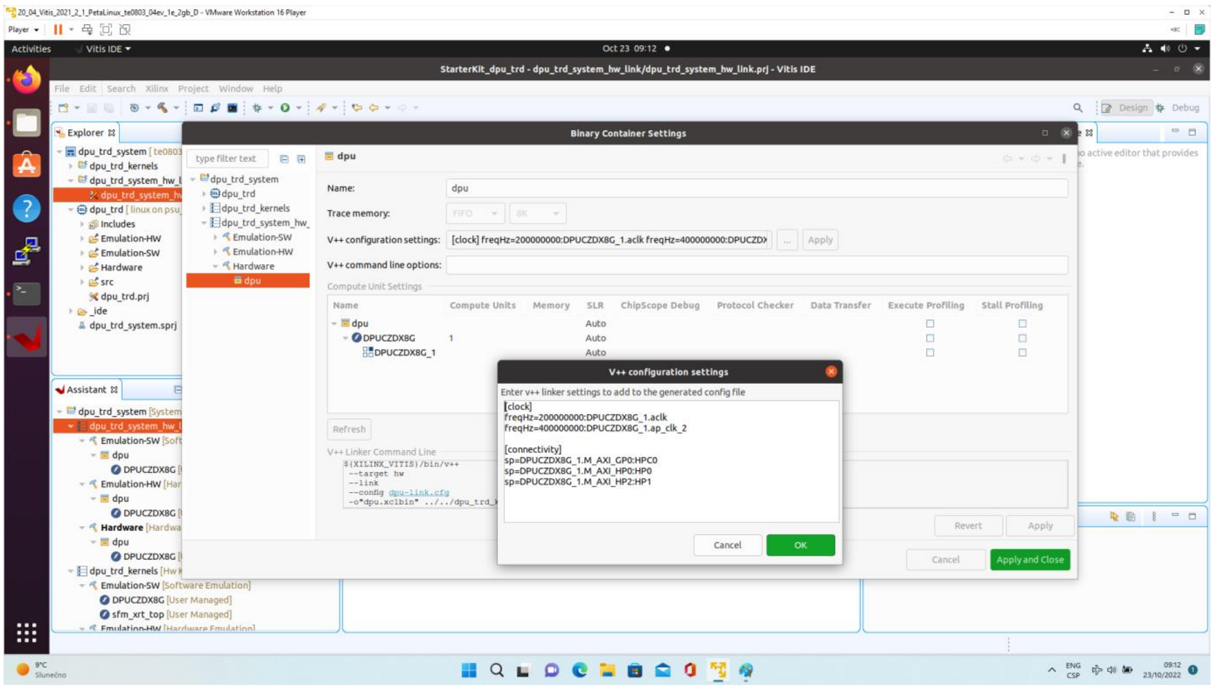
This modification is necessary for successful implementation of the DPU on the used module.

r. 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.

s. Configure connection of DPU kernels
   On the same tab right click on **dpu** and choose **Edit V++ Options**:


StarterKit_dpu_trd - dpu_trd_system_hw_link/dpu_trd_system_hw_link.prj - Vitis IDE

Binary Container Settings

Name: dpu

Trace memory: FIFO 8K

V++ configuration settings: [clock] freqHz=200000000:DPUCZDX8G_1.aclk freqHz=400000000:DPUCZDX

V++ command line options:

Compute Unit Settings

Name	Compute Units	Memory	SLR	ChipScope Debug	Protocol Checker	Data Transfer	Execute Profiling	Stall Profiling
dpu			Auto				☐	☐
DPUCZDX8G	1		Auto				☐	☐
DPUCZDX8G_1			Auto				☐	☐

V++ Linker Command Line

```
${XILINX_VITIS}/bin/v++
  --target hw
  --link
  --config dpu-link.cfg
  -o"dpu.xclbin" ../../dpu_trd_
```

V++ configuration settings

Enter v++ linker settings to add to the generated config file

```
[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
```

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

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

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

6. Build DPU_TRD application
   t. In “Explorer” section of Vitis IDE, click on: **dpu_trd_system[TE0821_01_240_pfm]** to select it.

   u. Right Click on: **dpu_trd_system[TE0821_01_240_pfm]** and select in the opened sub-menu: **Build project**

7. Run DPU_TRD on Board
   v. Write **sd_card.img** to SD card using SD card reader.

   w. The **sd_card.img** file is output of the compilation and packing by Vitis. It is located in directory: **~/work/TE0821_01_240/test_board_dpu_trd/dpu_trd_system/Hardware/package/**

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

x. Boot the board and open terminal on the board by opening ethernet connection to ssh server on the board with activated X11 forwarding. Continue using the embedded board terminal.
y. 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.
```

z. Switch from board terminal back to PC.
aa. Copy demo sources from PC to home folder on the board, connect to board using SFTP and copy to board **/home/root** two folders:

~/vitis_ai_2_0/demo/VART/resnet50 to **/home/root**
~/vitis_ai_2_0/demo/VART/common to **/home/root**

bb. Keep SFTP connection and copy to board **/home/root** folder:

~/vitis_ai_2_0/demo/Vitis-AI-Library/samples/facedetect to **/home/root**

cc. Open ZIP archive provided with this appnote and copy image and model files provided with this appnote from folder **B1600_2CG** to home folder in embedded board using SFTP. Copy following folders to **/home/root** (overwrite existing folders from previous step):

B1600_2CG/facedetect
B1600_2CG/images all to **/home/root**
B1600_2CG/resnet50

NOTE: Pre-compiled models for custom DPU are provided in ZIP archive enclosed to this appnote. Use **Appendix I** if you want to compile models for used DPU configuration again.

dd. Switch back to embedded terminal.
ee. Build executables on embedded host - visit both folders **resnet50** and **facedetect** and execute:

```
$ chmod +x build.sh
$ build.sh
```

ff. Run ‘resnet50’ demo

```
$ cd /home/root/resnet50
$ ./resnet50 resnet50.xmodel
```

The demo will use image located at folder ‘../images’ and result will be shown in terminal. The input image is shown on screen if X11 forwarding is used.

gg. Set environment variable:

```
export XLNX_VART_FIRMWARE=/mnt/sd-mmcblk1p1/dpu.xclbin
```

hh. Run ‘facedetect’ demo

```
$cd /home/root/facedetect
$./test_performance_facedetect densebox_640_360.xmodel test_performance_facedetect.list
```

The files listed in .list file will be used to evaluate DPU performance for “densebox_640_360” model. To run ‘facedetect’ demo using USB webcam first connect USB webcam and then start the demo:

```
$./test_video_facedetect densebox_640_360.xmodel 0 -t 1
```

Result can be seen in output window if X11 forwarding is used.

4 Appendix I – Compile Models for Custom DPU

1. Prerequisities
    a. Install Docker. See https://docs.docker.com/engine/install/.
    b. We do assume that “TE0821 test board Vitis AI Tutorial” [1] was implemented.
    c. We do also assume that this appnote tutorial was followed up to the step where dpu_trd project was built.

    NOTE: This part of guide is following instructions provided in Xilinx Vitis AI Github repository and Avnet tutorial at hackster.io (https://www.hackster.io/AlbertaBeef/vitis-ai-2-0-flow-for-avnet-vitis-platforms-06cfd6) where more details can be found.
2. Go to Vitis AI library and download models for ‘resnet50’ and ‘facedetect’
    a. Go to model folder and run:

```
$cd ~/vitis_ai_2_0/models/AI-Model-Zoo
$python3 downloader.py
```

   b. In downloader script choose:

      > Input: **cf densebox**
      > **2**
      > **1**

      ZIP archive with ‘densebox’ model will be downloaded from Xilinx

      Run downloader.py again to download also ‘resnet50’ model:

      >Input**: cf resnet50**
      >**1**

      ZIP archive with ‘resnet50’ model will be downloaded

   c. Unzip both archives downloaded in previous step.

3. Locate file **arch.json** in compiled dpu_trd project and copy it to **~/vitis_ai_2_0/models/AI-Model-Zoo**. The file contains fingerprint of current DPU hardware.
4. Find script file **compile_cf_model.sh** provided with this tutorial and copy it to ~/vitis_ai_2_0/models/AI-Model-Zoo.
5. (only first time) Pull docker container:

```
$docker pull xilinx/vitis-ai:2.0.0.1103
```

6. In the folder **~/vitis_ai_2_0** run docker and change directory to model Zoo:

```
$sh -x docker_run.sh xilinx/vitis-ai:2.0.0.1103

Vitis-AI /workspace > cd models/AI-Model-Zoo
```

7. Create directory for output (see **compile_cf_model.sh** script):

```
Vitis-AI /workspace/models/AI-Model-Zoo > mkdir compiled_output
```

8. Compile models:

```
$conda activate vitis-ai-caffe
(vitis-ai-caffe) Vitis-AI /workspace/models/AI-Model-Zoo > source ./compile_cf_model.sh
densebox_640_360 cf_densebox_wider_360_640_1.11G_2.0
(vitis-ai-caffe) Vitis-AI /workspace/models/AI-Model-Zoo > source ./compile_cf_model.sh
resnet50 cf_resnet50_imagenet_224_224_7.7G_2.0
```

9. Compiled models can be found in output folder as *.xmodel files. To successfully run ‘facedetect’ demo, it is also needed to have ‘*.prototxt’ file which can be found in package provided with this appnote.

5 References

[1] TE0821 test board Vitis AI Tutorial, Trenz Electronic Wiki: https://wiki.trenz-electronic.de/display/PD/TE0821+test+board+Vitis+AI+Tutorial