Application Note

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

Xilinx Vitis AI *facedetect* Demo on Trenz Electronic TE0820-4EV SoM with TE0701-06 Carrier Board and Avnet HDMI In/Out FMC Card

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

Revision history

Rev.	Date	Author	Description
0	07.02.2023	L.Kohout	Document creation
1			
2			

Contents

Acknowledgement

Acknowledgement to the StorAIge project 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 program and France, Belgium, Czech Republic, Germany, Italy, Sweden, Switzerland, Turkey.

1 Introduction

This document provides a tutorial describing setup and run Vitis AI 2.0 *facedetect* demonstrator on Trenz TE0820-4EV SoM [1] with Trenz TE0701-06 carrier board [2] and Avnet HDMI In/Out FMC card [3]. The system uses a Xilinx DPU unit to accelerate calculations. An input video data are taken from the HDMI input (FMC), processed data are displayed via HDMI output (FMC). The system desktop is shown via HDMI output on TE0701 carrier board. As the TE0820-4EV provides H.264/H.265 Video Codec Unit (VCU), the demonstrator is configured to use the decoder of the VCU.

2 Requirements

2.1 Hardware

- Trenz Electronic TE0820-4EV SoM with Xilinx Zynq UltraScale+ ZU4EV-1Iand 2GB DDR4 [1].
- Trenz Electronic TE0701-06 carrier board [2].
- AVNET HDMI In/Out FMC card [3] (AES-FMC-HDMI-CAM-G).
- HDMI camera, tested device is LAMAX X10 TAURUS. This camera provides HDMI output in 1920x1080 resolution at 60 frames per second. The camera can be replaced by the output from the PC.
- HDMI monitor capable to display 1920x1080 resolution at 60 frames per second.

2.2 Software

- This tutorial is an extension of the *TE0820 test board Vitis AI Tutorial* [4] https://wiki.trenz-electronic.de/display/PD/TE0820+test+board+Vitis+AI+Tutorial. Before following this guide, it is required to go through the *TE0820 test board Vitis AI Tutorial* from the beginning up to step 3.1.1 *Create Extensible Vitis platform*. As many of the next steps are slightly different from the original tutorial, this document will describe all of them again. In Section select the *Fast Track* branch.

3 Create Extensible Vitis Platform

This section continues with the description of the extensible Vitis platform creation. It is assumed that the Vivado project has been already created and it is opened now. The working directory is *~/work/te0820_15_240/test_board*.

3.1 Build HW

To create a HW of the extensible Vitis platform follow the steps bellow.

1. Add HDMI IP cores to the Vivado project. The IPs are the content of the package attached with this document. Copy content of the folder

   ip_lib to **~/work/te0820_15_240/test_board/ip_lib**

   The *ip_lib* folder should contain these IP cores (folders):

```
├── axis_vid_det_1_0
├── hblank_det_1_0
├── im_hdmi_in
├── im_hdmi_out
├── labtools_fmeter
└── video_io_to_hdmi
```

NOTE: The project IP catalog will be updated automatically in the next step. Otherwise, it can be updated by the user with command "update_ip_catalog -rebuild" executed from the Vivado TCL console.

2. Block Design of the Vivado project must be opened for this step. The package attached with this application note contains a TCL file containing all commands to create a Vivado block design of the extensible Vitis platform. To create the platform copy the file

   vivado/te0820_EV_fast_track_vcu_vdma.tcl

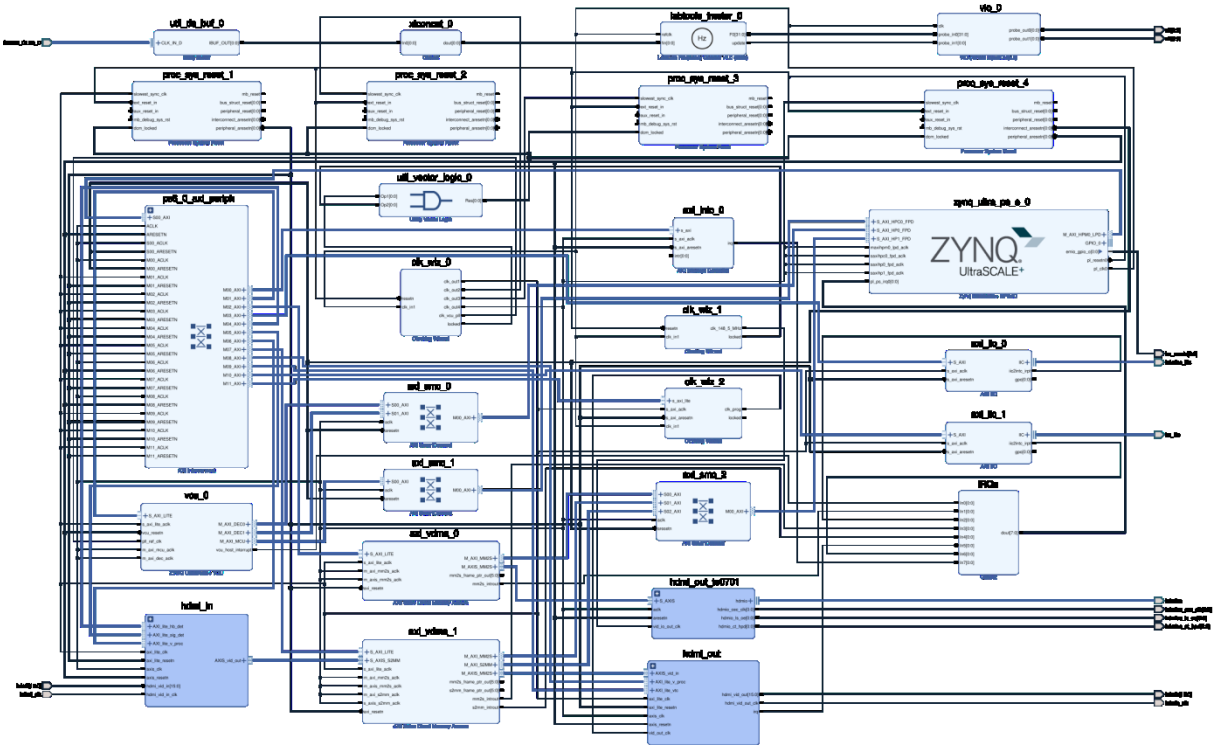
   into the folder

   ~/work/te0820_15_240/test_board/vivado

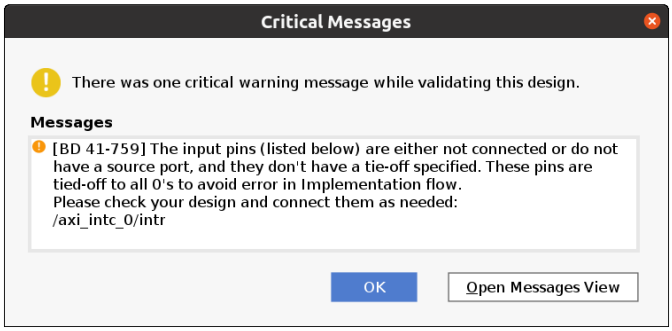
   and execute from the Vivado TCL console

   ```
   source te0820_EV_fast_track_vcu_vdma.tcl
   ```

   Expected block design is shown in figure bellow. To get an overview of the generated extensible platform, explore the *Platform Setup* tab.



3. In **Diagram** window, validate design by clicking on **Validate Design** icon. Received *Critical Messages* window indicates that input *intr[0:0]* of *axi_intc_0* is not connected. This is expected. The Vitis extensible design flow will connect this input to interrupt outputs from generated HW IPs. Kick *OK*.



4. Compile the created block design with Trenz scripts. In Vivado Tcl Console, type following script and execute it. It will take some time to compile HW.

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

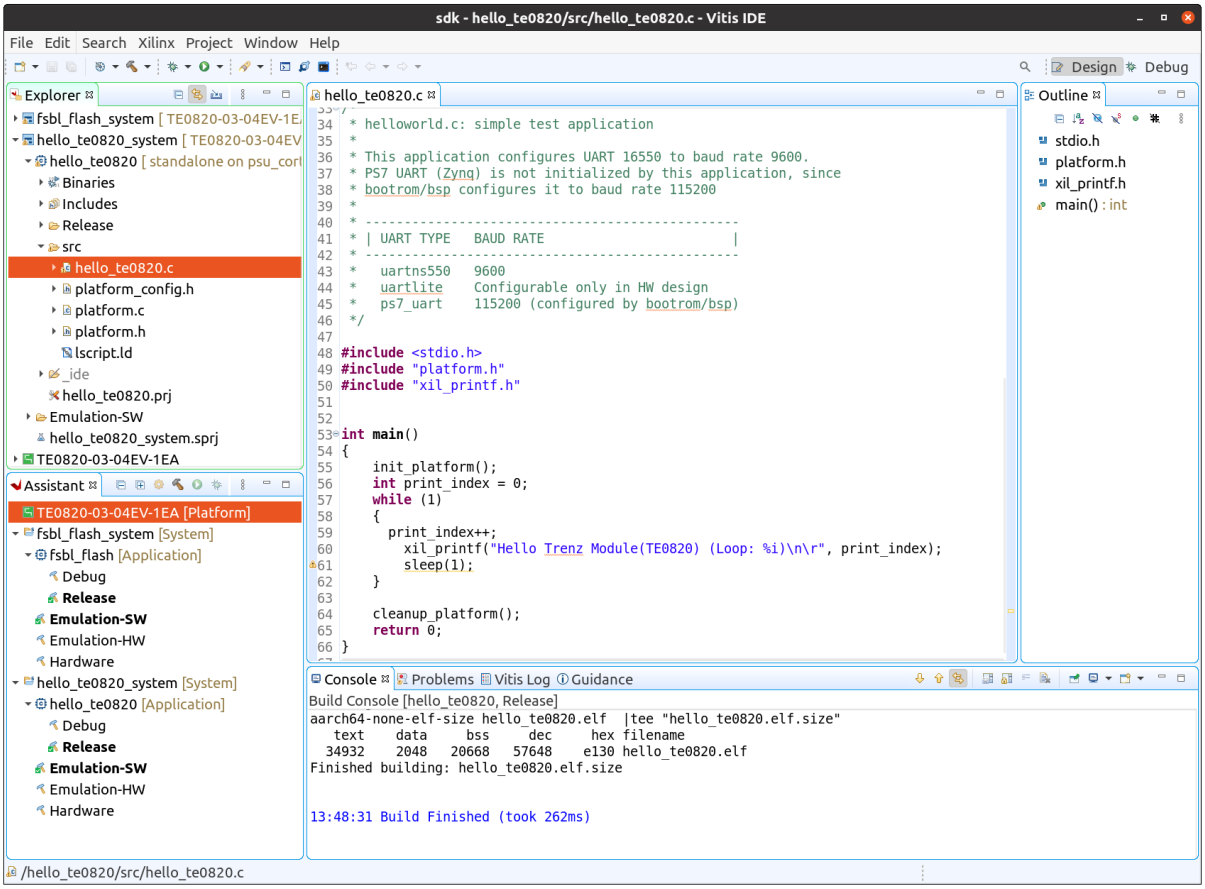
The resulting XSA file is created in the location:

~/work/te0820_15_240/test_board/vivado/test_board_4ev_1e_2gb.xsa

5. Compile custom SW with Trenz scripts (FSBL, ….). In Vivado Tcl Console, type the following script and execute it by Enter. It will take some time to compile.

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

After the script controlling SW compilation is finished, the Vitis SDK GUI is opened. Close the Vitis *Welcome* page. Compile the two included SW projects. Standalone custom Vitis platform **TE0820-03-04EV-1EA** has been created and compiled.



The **TE0820-03-04EV-1EA** Vitis platform includes Trenz Electronic custom first stage boot loader. It includes SW extension specific for the Trenz SoM initialization. This custom *zynqmp_fsbl* project has been compiled into executable file *fsbl.elf*. It is located in folder:

~/work/te0820_15_240/test_board/prebuilt/software/4ev_1e_2gb/fsbl.elf

6. Exit the opened Vitis SDK project.
7. Exit Vivado.

3.2 Get Vitis AI Libraries

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

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

   Click on green **Code** button and download **Vitis-AI-2.0.zip** file.

2. Unzip

   Vitis-AI-2.0.zip

   file to directory

   ~/vitis_ai_2_0.

3.3 Build OS and SDK

To configure the default Trenz Petalinux for the Vitis extensible platform follow the steps described in the list below.

1. Go to the Petalinux working directory:

```
cd ~/work/te0820_15_240/test_board/os/petalinux
```

2. Set the path to Petalinux tool:

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

3. Configure Petalinux with the current HW specification, which is part of the file **test_board_4ev_1e_2gb.xsa**.

```
petalinux-config --get-hw-description=../../vivado
```

   Select **Exit** and then **Yes** to close this window.

4. Customize main *device-tree* of the Petalinux project. Open file

   ~/work/te0820_15_240/test_board/os/petalinux/project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi

   and append these sections into the file

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

/ {
  reserved-memory {
    #address-cells = <2>;
    #size-cells = <2>;
    ranges;
    reserved1: buffer@0x28000000 {
      no-map;
      reg = <0x0 0x28000000 0x0 (1920 * 1080 * 4)>;
```

```
    };
  };

  framebuffer1: fbxserver {      // HDMI out
    #address-cells = <2>;
    #size-cells = <2>;
    compatible = "simple-framebuffer";
    reg = <0x0 0x28000000 0x0 (1920 * 1080 * 4)>;    // 1080p
    width = <1920>;
    height = <1080>;
    stride = <(1920 * 4)>;
    format = "a8b8g8r8";
  };
};

/delete-node/ &hdmi_in_v_proc_ss_0;
/delete-node/ &hdmi_out_v_proc_ss_0;
/delete-node/ &axi_vdma_0;
/delete-node/ &axi_vdma_1;

&amba {
  hdmi_in_v_proc_ss_0: v_proc_ss_0@80000000 {
    clock-names = "aclk";
    clocks = <&misc_clk_1>;
    compatible = "xlnx,v-proc-ss-2.3", "xlnx,vpss-csc", "xlnx,v-vpss-csc";
    reg = <0x0 0x80000000 0x0 0x10000>;
    xlnx,colorspace-support = <1>;
    xlnx,csc-enable-window = "false";
    xlnx,max-height = <2160>;
    xlnx,max-width = <3840>;
    xlnx,num-video-components = <3>;
    xlnx,samples-per-clk = <1>;
    xlnx,topology = <3>;
    xlnx,use-uram = <0>;
    xlnx,video-width = <8>;
  };
  hdmi_out_v_proc_ss_0: v_proc_ss_1@80010000 {
    clock-names = "aclk";
    clocks = <&misc_clk_1>;
    compatible = "xlnx,v-proc-ss-2.3", "xlnx,vpss-csc", "xlnx,v-vpss-csc";
    reg = <0x0 0x80010000 0x0 0x10000>;
    xlnx,colorspace-support = <1>;
    xlnx,csc-enable-window = "false";
    xlnx,max-height = <2160>;
    xlnx,max-width = <3840>;
    xlnx,num-video-components = <3>;
    xlnx,samples-per-clk = <1>;
    xlnx,topology = <3>;
    xlnx,use-uram = <0>;
    xlnx,video-width = <8>;
  };

  /* VDMA TE0701 HDMIO*/
  axi_vdma_0: vdma_0@80050000 {
    #dma-cells = <1>;
    clock-names = "s_axi_lite_aclk", "m_axi_mm2s_aclk", "m_axis_mm2s_aclk";
    clocks = <&misc_clk_0>, <&misc_clk_1>, <&misc_clk_1>;
    compatible = "xlnx,axi-vdma-6.3", "xlnx,axi-vdma-1.00.a";
    interrupt-names = "mm2s_introut";
    interrupt-parent = <&gic>;
    interrupts = <0 90 4>;
    reg = <0x0 0x80050000 0x0 0x10000>;
    xlnx,addrwidth = <0x20>;
    xlnx,flush-fsync = <0x1>;
    xlnx,num-fstores = <0x1>;
    dma-channel@80050000 {
```

```
      compatible = "xlnx,axi-vdma-mm2s-channel";
      interrupts = <0 90 4>;
      xlnx,datawidth = <0x20>;
      xlnx,device-id = <0x0>;
    };
  };

  /* VDMA IMAGEON HDMI IN/OUT */
  axi_vdma_1: vdma_1@80060000 {
    #dma-cells = <1>;
    clock-names = "s_axi_lite_aclk", "m_axi_mm2s_aclk", "m_axis_mm2s_aclk",
                  "m_axi_s2mm_aclk", "s_axis_s2mm_aclk";
    clocks = <&misc_clk_0>, <&misc_clk_1>, <&misc_clk_1>,
             <&misc_clk_1>, <&misc_clk_1>;
    compatible = "xlnx,axi-vdma-6.3", "xlnx,axi-vdma-1.00.a";
    interrupt-names = "mm2s_introut", "s2mm_introut";
    interrupt-parent = <&gic>;
    interrupts = <0 92 4 0 93 4>;
    reg = <0x0 0x80060000 0x0 0x10000>;
    xlnx,addrwidth = <0x20>;
    xlnx,flush-fsync = <0x1>;
    xlnx,num-fstores = <0x7>;
    dma-channel@80060000 {
      compatible = "xlnx,axi-vdma-mm2s-channel";
      interrupts = <0 92 4>;
      xlnx,datawidth = <0x18>;
      xlnx,device-id = <0x1>;
      xlnx,genlock-mode ;
    };
    dma-channel@80060030 {
      compatible = "xlnx,axi-vdma-s2mm-channel";
      interrupts = <0 93 4>;
      xlnx,datawidth = <0x18>;
      xlnx,device-id = <0x1>;
      xlnx,genlock-mode ;
    };
  };
};

&clk_wiz_2 {
    compatible = "generic-uio";
    status = "okay";
};

&axi_vdma_0 {
    compatible = "generic-uio";
    status = "okay";
};

&axi_vdma_1 {
    compatible = "generic-uio";
    status = "okay";
};

&hdmi_out_v_tc_0 {
    compatible = "generic-uio";
    status = "okay";
};

&hdmi_in_v_proc_ss_0 {
    compatible = "generic-uio";
    status = "okay";
};
```

```
&hdmi_out_v_proc_ss_0 {
    compatible = "generic-uio";
    status = "okay";
};

&hdmi_in_axis_vid_det_0 {
    compatible = "generic-uio";
    status = "okay";
};

&hdmi_in_hblank_det_0 {
    compatible = "generic-uio";
    status = "okay";
};

&gpio {
    emio-gpio-width = <1>;
    status = "okay";
};
```

5. Customize *device-tree* of the Petalinux project for *u-boot*. Open file

   ~/work/te0820_15_240/test_board/os/petalinux/project-spec/meta-user/recipes-bsp/uboot-device-tree/files/system-user.dtsi

   and append these sections into the file

```
/delete-node/ &hdmi_in_v_proc_ss_0;
/delete-node/ &hdmi_out_v_proc_ss_0;
```

6. Add *Vitis-AI 2.0* library recipes to the Petalinux project. Copy

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

   to

   ~/work/te0820_15_240/test_board/os/petalinux/project-spec/meta-user/

7. Configure Petalinux rootfs.

   a. Append user defined rootfs packages to the petalinux *menuconfig*. To the file

      ~/work/te0820_15_240/test_board/os/petalinux/project-spec/meta-user/conf/user-rootfsconfig

      append these lines:

```
CONFIG_xrt
CONFIG_xrt-dev
CONFIG_zocl
CONFIG_opencl-clhpp-dev
CONFIG_opencl-headers-dev
CONFIG_packagegroup-petalinux-opencv
CONFIG_packagegroup-petalinux-opencv-dev
CONFIG_dnf
CONFIG_e2fsprogs-resize2fs
CONFIG_parted
CONFIG_resize-part
CONFIG_packagegroup-petalinux-vitisai
CONFIG_packagegroup-petalinux-self-hosted
CONFIG_cmake
CONFIG_packagegroup-petalinux-vitisai-dev
CONFIG_mesa-megadriver
CONFIG_packagegroup-petalinux-x11
CONFIG_packagegroup-petalinux-v4lutils
CONFIG_packagegroup-petalinux-matchbox
CONFIG_vitis-ai-library
CONFIG_vitis-ai-library-dev
```

```
CONFIG_vitis-ai-library-dbg
CONFIG_packagegroup-petalinux-gstreamer
CONFIG_libomxil
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
```

b. Open the petalinux rootfs configuration window, execute from the terminal:

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

c. Go to the **user packages** and deselect packages **imagefeature-ssh-server-dropbear** and **packagegroup-core-ssh-dropbear**. Enable all other packages.

```
[*] libomxil
[*] cmake (NEW)
[*] dnf
[*] e2fsprogs-resize2fs
[*] imagefeature-package-management
[ ] imagefeature-ssh-server-dropbear
[*] imagefeature-ssh-server-openssh
[*] mesa-megadriver
[*] opencl-clhpp-dev
[*] opencl-headers-dev (NEW)
[*] openssh
[*] openssh-scp
[*] openssh-sftp-server
[*] openssh-sshd
[ ] packagegroup-core-ssh-dropbear
[*] packagegroup-petalinux-gstreamer
[*] packagegroup-petalinux-matchbox
[*] packagegroup-petalinux-opencv
[*] packagegroup-petalinux-opencv-dev
[*] packagegroup-petalinux-self-hosted
[*] packagegroup-petalinux-v4lutils
[*] packagegroup-petalinux-vitisai (NEW)
[*] packagegroup-petalinux-vitisai-dev (NEW)
[*] packagegroup-petalinux-x11
[*] parted
[*] resize-part
[*] vitis-ai-library (NEW)
[*] vitis-ai-library-dbg (NEW)
[*] vitis-ai-library-dev (NEW)
[*] xrt
[*] xrt-dev
[*] zocl
```

d. **Exit** twice and select **Yes** to save changes

8. Configure Petalinux kernel.

a. Launch kernel config:

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

b. Disable CPU IDLE. Ensure the following items are **TURNED OFF** by entering 'n' in the [ ] menuconfig selection:

```
CPU Power Management  --->
  CPU Idle  --->
    [ ] CPU idle PM support
  CPU Frequency scaling  --->
    [ ] CPU Frequency scaling
```

NOTE: When the processor is not in use it is switched to CPU IDLE (WFI). When JTAG is connected, the hardware server on host machine talks to the processor regularly. If it talks to a processor in IDLE state, 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.

c. Enable simple framebuffer to support graphical output via HDMI on the TE0701.

```
Device Drivers  --->
  Graphics support  --->
    Frame buffer Devices  --->
      -*- Support for frame buffer devices  --->
        [*]   Simple framebuffer support
```

d. **Exit** kernel configuration window, select **Yes** to save changes.

9. Configure common Petalinux settings.

a. Launch Petalinux menuconfig window.

```
petalinux-config
```

b. Set Petalinux to use an extra EXT4 partition on the SD card for the file system. Switch the root file system from INITRD to **EXT4**:

```
Image Packaging Configuration  --->
  Root filesystem type (INITRD)  --->
    (X) EXT4 (SD/eMMC/SATA/USB)
```

Go back to

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

level and change the *Device node of SD* device from the default value

/dev/mmcblk0p2

to new value (use the second partition):

/dev/mmcblk1p2

c. Set Petalinux to build only the EXT4 file as the root file system image. The original setting may cause the Petelinux build to fail. Go to level

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

and modify the *Root filesystem formats* string from

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

to

ext4

d. Set the Petalinx boot arguments to use the EXT4 partition, pre-allocate 512 MB for CMA, and enable the generic UIO driver. Go to:

```
DTG Settings  --->
  Kernel Bootargs  --->
```

Disable

generate boot args automatically

option, and set

user set kernel bootargs

string to

earlycon console=ttyPS0,115200 clk_ignore_unused root=/dev/mmcblk1p2 rw rootwait cma=512M uio_pdrv_genirq.of_id=generic-uio

e. **Exit** this menuconfig window, select **Yes** to save changes.

10. Build the Petalinux image:

```
petalinux-build
```

The image files will be generated in the directory:

~/work/te0820_15_240/test_board/os/petalinux/images/linux

Compilation Petalinux takes some time, requires an internet connection and enough free disk space.

11. Create Petalinux SDK. It is a copy of the Petalinux root file system that is used by the Vitis tool to cross compile applications for newly created platfom. In terminal, execute:

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

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

~/work/te0820_15_240/test_board/os/petalinux/images/linux

Generation of the SDK package takes some time and requires enough free disk space.

3.4 Build Platform

To build the final extensible Vitis platform follow steps bellow:

1. Create the main platform folder **te0820_15_240_pfm** and the required subfolders:

```
mkdir -p ~/work/te0820_15_240/test_board_pfm/pfm/boot
mkdir -p ~/work/te0820_15_240/test_board_pfm/pfm/sd_dir
```

2. Go to the platform directory

```
cd ~/work/te0820_15_240/test_board_pfm
```

3. Copy all already compiled files to the platform directory.

   a. FSBL (see Section 3.1, Step 5)

```
cp ../test_board/prebuilt/software/4ev_1e_2gb/fsbl.elf pfm/boot/
```

   b. Petalinux image files (see Section 3.3, Step 10)

```
cp ../test board/os/petalinux/images/linux/bl31.elf       pfm/boot/

cp ../test board/os/petalinux/images/linux/pmufw.elf      pfm/boot/

cp ../test_board/os/petalinux/images/linux/system.dtb     pfm/boot/

cp ../test_board/os/petalinux/images/linux/u-boot-dtb.elf pfm/boot/u-boot.elf
```

```
cp ../test board/os/petalinux/images/linux/boot.scr       pfm/sd dir/

cp ../test_board/os/petalinux/images/linux/system.dtb     pfm/sd_dir/
```

   c. Trenz **init.sh** script that is an place-holder for user defined bash code to be executed during the boot sequence:

```
cp ../test_board/misc/sd/init.sh pfm/sd_dir/
```

The platform directory **te0820_15_240_pfm** should contain these files:

```
pfm
├── boot
│   ├── bl31.elf
│   ├── fsbl.elf
│   ├── pmufw.elf
│   ├── system.dtb
│   └── u-boot.elf
└── sd_dir
    ├── boot.scr
    ├── init.sh
    └── system.dtb
```

4. Generate the Petalinux root file system (SYSROOT) used for cross compilation within the Vitis tool. See result of Section 3.3, Step 11.

```
../test_board/os/petalinux/images/linux/sdk.sh -d .
```

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

~/work/te0820_15_240/test_board_pfm/sysroots/x86_64-petalinux-linux
~/work/te0820_15_240/test_board_pfm/sysroots/cortexa72-cortexa53-xilinx-linux

NOTE: Once it is created, do not move these SYSROOT directories (due to some internally created paths).

5. Set path to the Vitis tool

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

6. Start Vitis

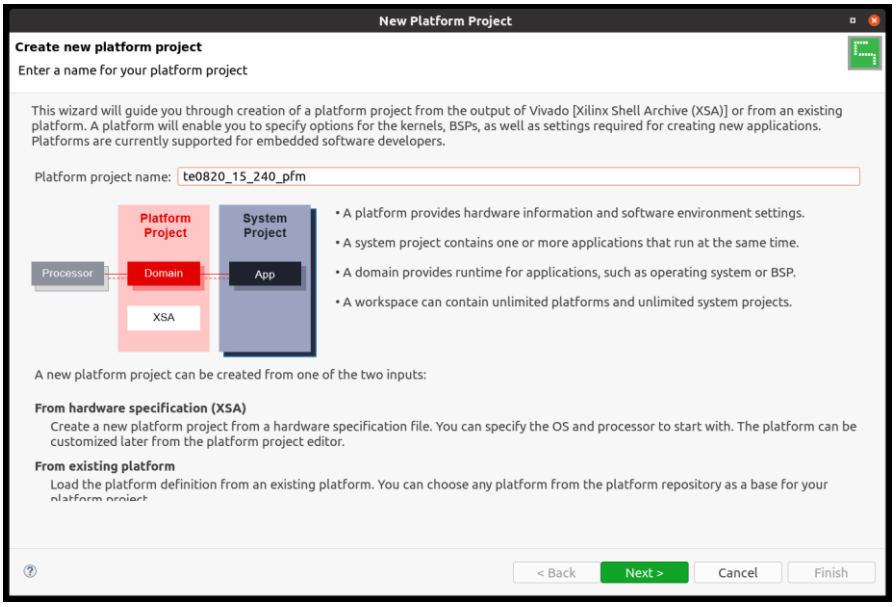
```
vitis
```

In *Vitis IDE Launcher*, set the workspace for the extensible platform compilation:

~/work/te0820_15_240/test_board_pfm

Close Welcome page.

7. Create a new platform project,
   a. Open menu **File→New→Platform Project…**
   b. Type name of the extensible platform **te0820_15_240_pfm**. Click **Next**.

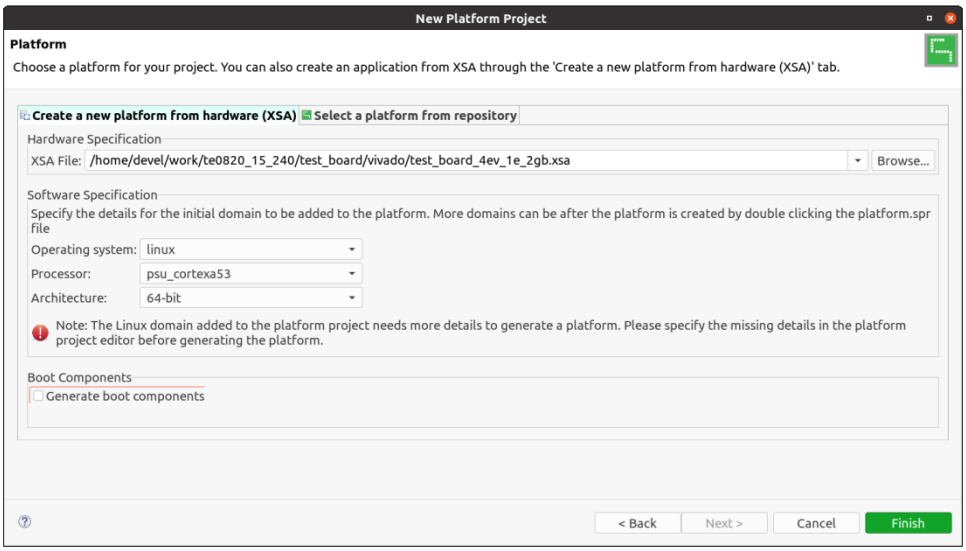


c. In **Hardware Specification** browse for the XSA platform file:

~/work/te0820_15_240/test_board/vivado/test_board_4ev_1e_2gb.xsa

d. In **Software specification** select: **linux**

In **Boot Components** unselect **Generate boot components**



Click **Finish**. New window **te0820_15_240_pfm** is opened.

8. Modify *Linux Domain* of the platform.

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

b. In **Description**: write **xrt**

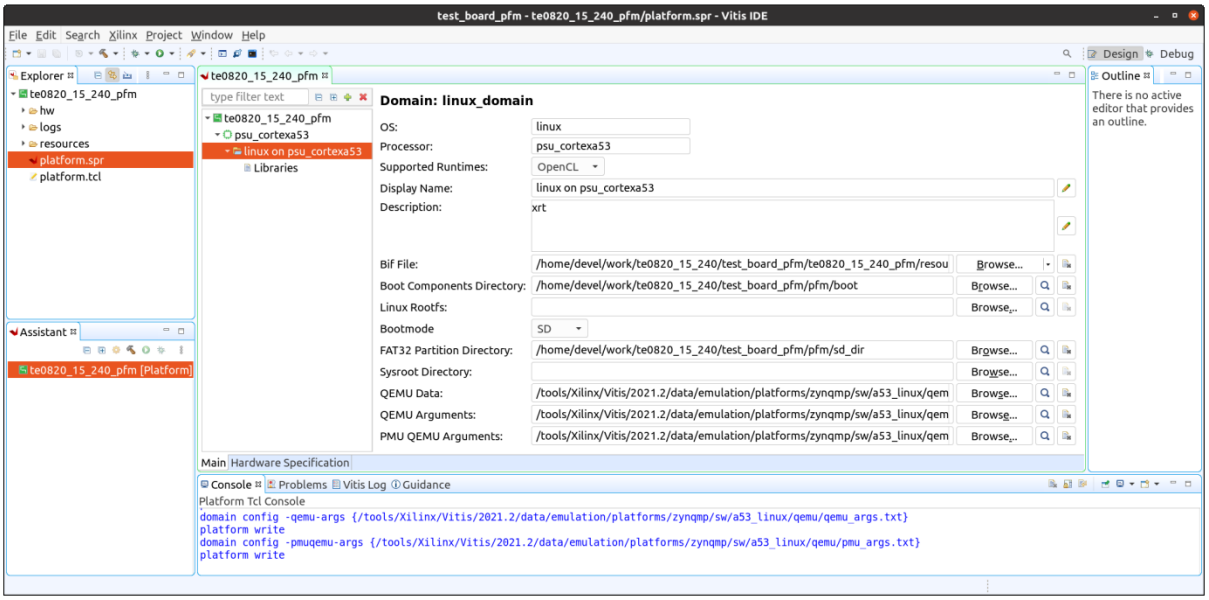
c. In **Bif File** find and select the pre-defied option **Generate Bif**

d. In **Boot Components Directory** select:

~/work/te0820_15_240/test_board_pfm/pfm/boot

e. In **FAT32 Partition Directory** select:

~/work/te0820_15_240/test_board_pfm/pfm/sd_dir



9. Build the platform. In Vitis IDE **Explorer** section, click on **te0820_15_240_pfm** to highlight it. Right-click on the highlighted **te0820_15_240_pfm** and select **Build Project** in the opened submenu. Platform is compiled in few seconds.
10. Close the Vitis tool, menu: **File→Exit**.

The Vits extensible platform *te0820_15_240_pfm* has been created in the directory:

~/work/te0820_15_240/test_board_pfm/te0820_15_240_pfm/export/te0820_15_240_pfm

3.5 Platform Inspection

When the path to Vitis tools is set, the **platforminfo** tool can be used to report the extensible Vitis platform (XPFM) information. From the **test_board_pfm** folder execute:

```
platforminfo te0820_15_240_pfm/export/te0820_15_240_pfm/te0820_15_240_pfm.xpfm
```

Report:

```
==========================
Basic Platform Information
==========================
Platform:       te0820_15_240_pfm
File:           /home/devel/work/te0820_15_240/test_board_pfm/te0820_15_240_pfm/
                export/te0820_15_240_pfm/te0820_15_240_pfm.xpfm
Description:
te0820_15_240_pfm

====================================
Hardware Platform (Shell) Information
====================================
Vendor:                           trenz
Board:                            zusys
Name:                             zusys
Version:                          2.0
Generated Version:                2021.2.1
Hardware:                         1
Software Emulation:               1
Hardware Emulation:               1
Hardware Emulation Platform:      0
FPGA Family:                      zynquplus
FPGA Device:                      xczu4ev
Board Vendor:                     trenz.biz
Board Name:                       trenz.biz:te0820_4ev_1e:2.0
Board Part:                       xczu4ev-sfvc784-1-e

=================
Clock Information
=================
  Default Clock Index: 4
  Clock Index:         1
    Frequency:         100.000000
  Clock Index:         2
    Frequency:         200.000000
  Clock Index:         3
    Frequency:         400.000000
  Clock Index:         4
    Frequency:         240.000000

==================
Memory Information
==================
  Bus SP Tag: HP2
  Bus SP Tag: HP3
  Bus SP Tag: HPC1
```

```
============================
Software Platform Information
============================
Number of Runtimes:            1
Default System Configuration:  te0820_15_240_pfm
System Configurations:
  System Config Name:                       te0820_15_240_pfm
  System Config Description:                te0820_15_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_15_240_pfm/boot/linux.bif
    Boot Image Data:            te0820_15_240_pfm/linux_domain/image
    Boot Image Boot Mode:       sd
    Boot Image RootFileSystem:
    Boot Image Mount Path:      /mnt
    Boot Image Read Me:         te0820_15_240_pfm/boot/generic.readme
    Boot Image QEMU Args:
te0820_15_240_pfm/qemu/pmu_args.txt:te0820_15_240_pfm/qemu/qemu_args.txt
    Boot Image QEMU Boot:
    Boot Image QEMU Dev Tree:
Supported Runtimes:
  Runtime: OpenCL
```

4 Facedetect Vitis-AI Demo

At this point, we use the extensible Vitis platform compiled in previous steps. The platform will be extended with the Xilinx Deep Learning Processor Unit (DPU). To add the DPU to the platform follow steps bellow:

1. Create new directory *test_board_dpu_trd*:

   ```
   mkdir -p ~/work/te0820_15_240/test_board_dpu_trd
   cd ~/work/te0820_15_240/test_board_dpu_trd
   ```

2. Start Vitis, in Ubuntu terminal execute

   ```
   vitis
   ```

   In Vitis IDE Launcher, select your working directory

   ~/work/te0820_15_240/test_board_dpu_trd

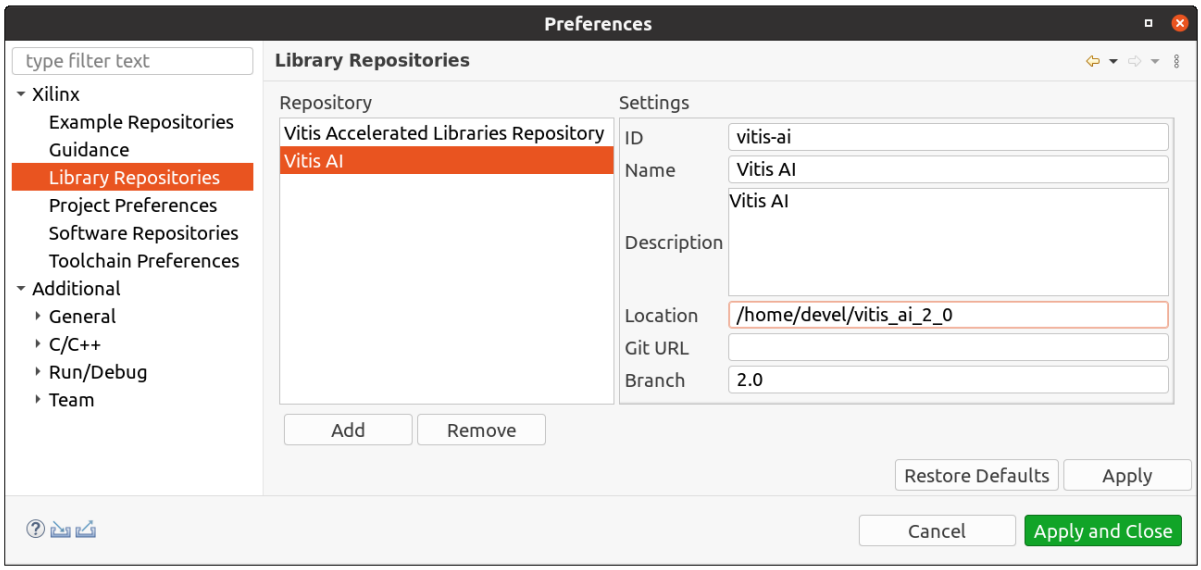
   Click on **Launch** button to start Vitis.

3. Add Vitis-AI Repository to Vitis:

   a. Open menu **Window→Preferences**.

   b. Go to **Library Repositories** tab.

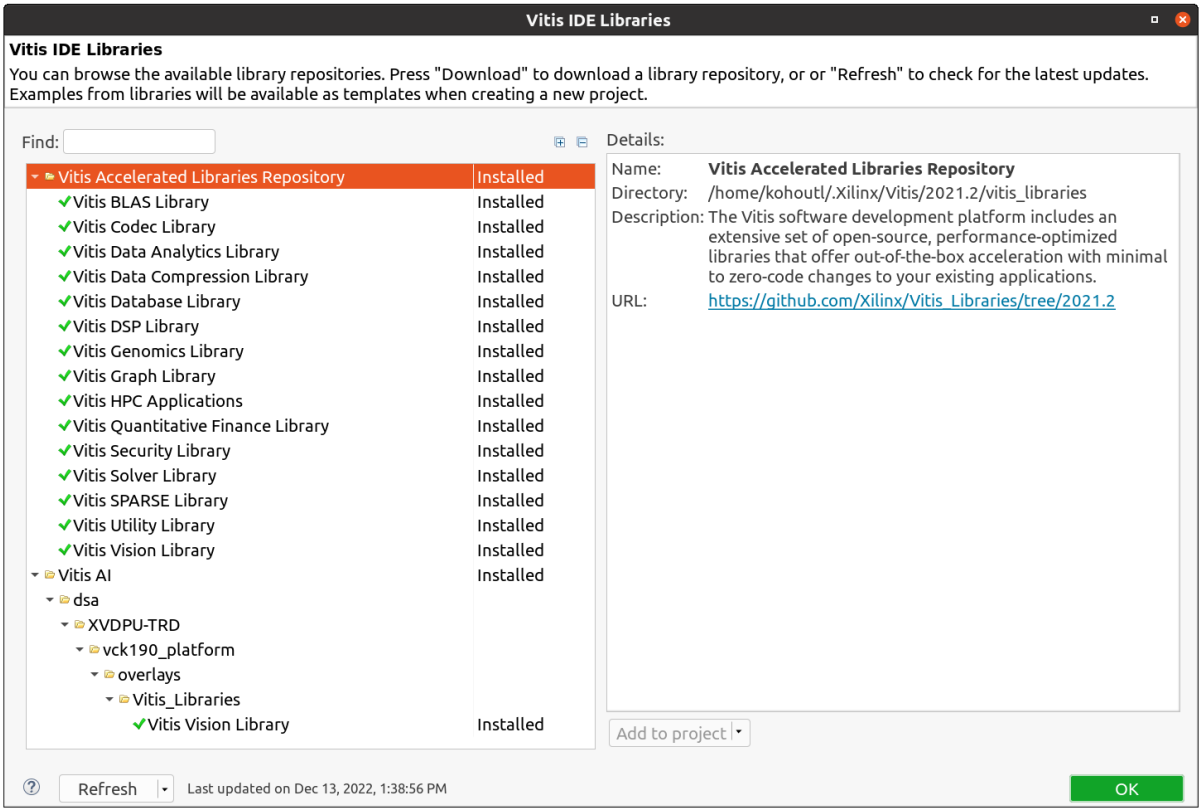
   c. Add Vitis-AI by clicking **Add** button and fill the form as shown below, use absolute path to your home folder in field **Location**.



d. Click **Apply and Close**

NOTE: Field "Location" says that the Vitis-AI repository from GITHUB has been already cloned into ~/vitis_ai_2_0 folder during the stage of the Petalinux configuration in Section 3.2. 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 that the library is correctly added, it appears in Libraries. Open menu **Xilinx→Libraries...**. You can see there just added Vitis-AI library marked as *Installed*.



4. Create a Vitis-AI design for our **te0820_15_240** custom platform.

   a. Select **File→New→Application Project…**. Click **Next**. Skip welcome page if it is shown.

   b. Click on **+ Add** icon and select the custom extensible platform **te0820_15_240_pfm** located in directory:

      ~/work/te0820_15_240/test_board_pfm/te0820_15_240_pfm/export/te0820_15_240_pfm.

   c. Click **Next**.

   d. In **Application Project Details** window type into **Application project name**:

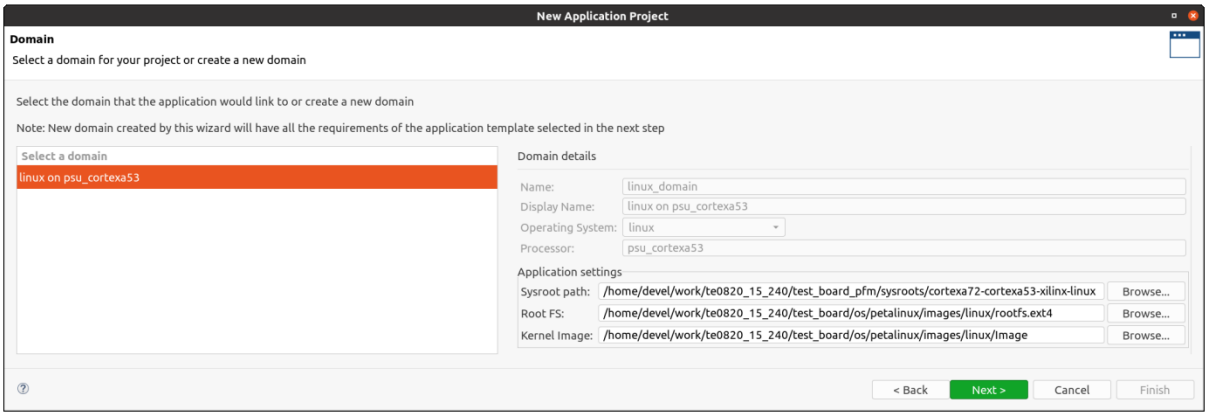
      dpu_trd

   e. Click **Next**.

   f. In **Domain window** type (or select by browse):

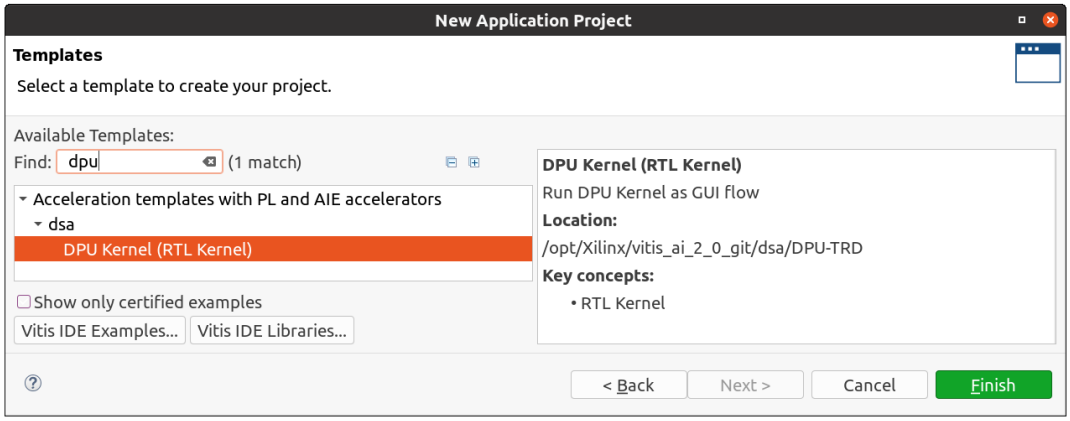
Field	Value
Sysroot path	~/work/te0820_15_240/test_board_pfm/sysroots/cortexa72-cortexa53-xilinx-linux
Root FS	~/work/te0820_15_240/test_board/os/petalinux/images/linux/rootfs.ext4
Kernel Image	~/work/te0820_15_240/test_board/os/petalinux/images/linux/Image

   g. Click **Next**.



   h. In the **dsa** folder, select: **DPU Kernel (RTL Kernel)**.

   i. Click **Finish**. New project template is created now.



5. Configure project and DPU:

   a. In dpu_trd window switch **Active build configuration** from Emulation-SW to **Hardware**.

   b. Configure DPU to utilize internal Ultra RAMs instead of Block RAMs. Open file

   dpu_trd_kernels/src/prj/Vitis/dpu_conf.vh

   and change line 37 from

   ```
   `define URAM_DISABLE
   ```

   to

   ```
   `define URAM_ENABLE
   ```

   NOTE: this step applies to FPGAs with Ultra RAMs, which is the case of the Xilinx zcu04-ev device.

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

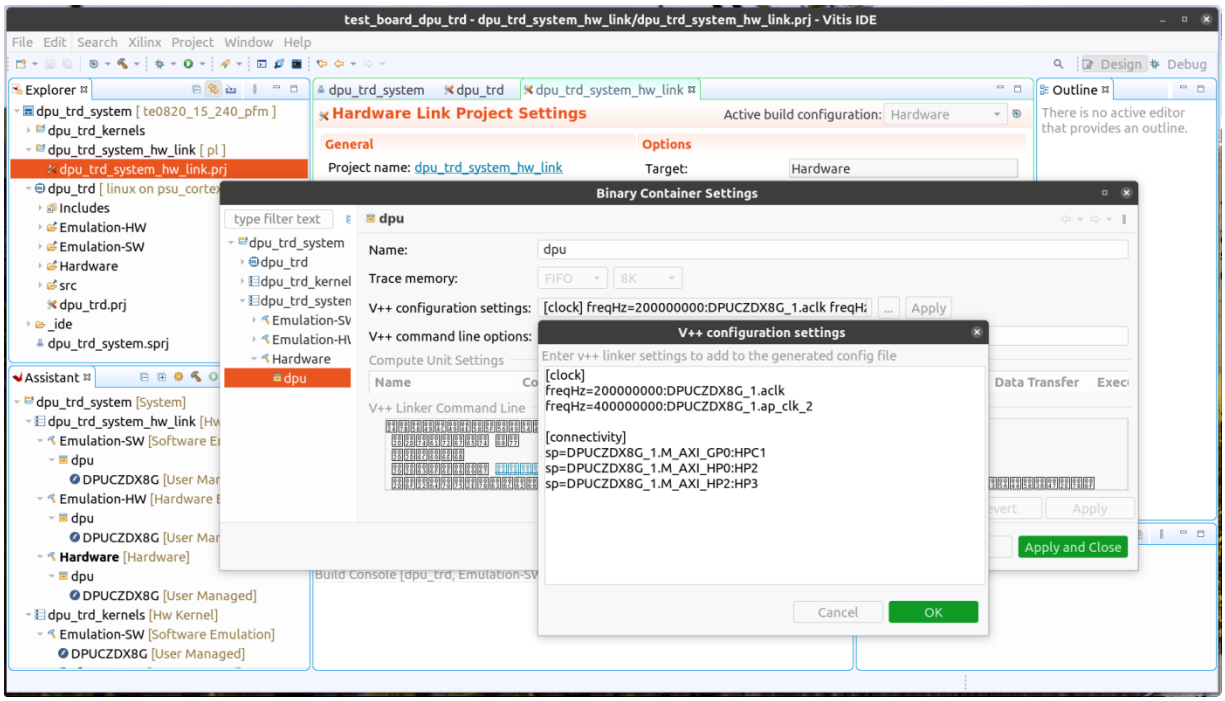
   c. Go to **dpu_trd_system_hw_link** and open **dpu_trd_system_hw_link.prj** file. Remove **sfm_xrt_top** kernel from binary container by right clicking on it and choosing remove. **Reduce number of DPU kernels to one**.

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

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

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

   [connectivity]
   sp=DPUCZDX8G_1.M_AXI_GP0:HPC1
   sp=DPUCZDX8G_1.M_AXI_HP0:HP2
   sp=DPUCZDX8G_1.M_AXI_HP2:HP3
   ```



6. Build DPU_TRD application

   a. In *Explorer* section of Vitis IDE, click on **dpu_trd_system[te0820_15_240_pfm]** to select it.

   b. Right Click on **dpu_trd_system[te0820_15_240_pfm]** and select in the opened sub-menu **Build project**.

7. Run the board with the DPU_TRD design.

   a. Write **sd_card.img** to SD card using SD card reader. The image file is an output product of the Vitis compilation and packaging phase. It is located in directory **~/work/te0820_15_240/test_board_dpu_trd/dpu_trd_system/Hardware/package/**

   NOTE: For writing the image file to the SD card can be used BalenaEtcher tool downloadable from https://www.balena.io/etcher. The tool is available for Windows, Linux or MAC platforms.

   b. Boot the board and open a terminal connection, USB UART or SSH.

   - UART connection via USB UART/JTAG (connector J8 on the board). On your host machine identify the device to be connected (COM port on Windows or ttyUSB device on Linux). On Windows machine it can be any port. On Linux machine are USB serial ports counted from zero. As the board provides two virtual devices and the second one is the UART, the device will be **/dev/ttyUSB1** (in case that only one board is connected). To connect the board use PUTTY, for instance. The settings are:
     - Baud rate – 115200
     - Data bits – 8
     - Stop bits – 1
     - Parity – none
     - Flow control – none

     NOTE: If you want to forward graphical applications started from this terminal to the board monitor connected via display port, you should also set DISPLAY variable correctly (export DISPLAY=:0.0).

   - SSH connection via Ethernet (preffered). To find out the board IP address use UART connection, execute command:

     ```
     ifconfig
     ```

     Or from Linux host machine, you can use *arp-scan* command. Example:

     ```
     sudo arp-scan -l -I enp4s0
     Interface: enp4s0, type: EN10MB, MAC: c4:6e:1f:01:b9:0d, IPv4:
     10.42.0.1
     Starting arp-scan 1.9.7 with 256 hosts
     (https://github.com/royhills/arp-scan)
     10.42.0.102       80:1f:12:d0:7d:0a   Microchip Technology Inc.

     1 packets received by filter, 0 packets dropped by kernel
     Ending arp-scan 1.9.7: 256 hosts scanned in 1.997 seconds (128.19
     hosts/sec). 1 responded
     ```

     NOTE: The SSH connection should have X11 forwarding activated, if you want to forward graphical applications to your host machine. You should also set DISPLAY variable correctly (export DISPLAY=:10.0).

c. Resize the second partition of the SD card to its maximal size, from the board terminal use command *resize-part*:

```
resize-part /dev/mmcblk1p2
/dev/mmcblk1p2
Warning: Partition /dev/mmcblk1p2 is being used. Are you sure you want to
continue?
Yes/No? yes
End?  [4295MB]? 100%
Information: You may need to update /etc/fstab.

resize2fs 1.45.6 (20-Mar-2020)
Filesystem at /dev/mmcblk1p2 is mounted on /media/sd-mmcblk1p2; on-line
resizing required
old_desc_blocks = 1, new_desc_blocks = 2
The filesystem on /dev/mmcblk1p2 is now 3539968 (4k) blocks long.
```

d. Initialize HDMI output on the TE0701 carrier board to display X11 desktop on the monitor. The HDMI output is fixed at 1920x1080p60 resolution.

- Copy prebuilt SW application for HDMI output from the attached package to board. Connect the board using SFTP and copy :

  hdmio/hdmio.elf to **/mnt/sd-mmcblk1p1**/

  Source codes of the application are also part of the attached ZIP file.

- Make file **hdmio.elf** executable, from the board terminal execute:

```
chmod +x /mnt/sd-mmcblk1p1/hdmio.elf
```

- Configure the *X11* server to start the *hdmio.elf* application automatically on when it starts. Connect the board using SFTP and copy from the enclosed ZIP file to the board:

  hdmio/01hdmio.sh to **/etc/X11/Xsession.d**

- Make file **01hdmio.sh** executable, from the board terminal execute:

```
chmod +x /etc/X11/Xsession.d/01hdmio.sh
```

- Configure *X11* server to use a *Simple Frame Buffer*. On the board modify file **/etc/X11/xorg.conf**:

```
Section "InputDevice"
  Identifier "System Mouse"
  Driver "mouse"
  Option "Device" "/dev/input/mouse0"
EndSection

Section "InputDevice"
  Identifier "System Keyboard"
  Driver "kbd"
  Option "Device" "/dev/input/event0"
EndSection

Section "Screen"
  Identifier "DefaultScreen"
  Device "/dev/fb0"
  DefaultDepth 24
EndSection

Section "Device"
  Identifier "myfb"
  Driver "fbdev"
  Option "fbdev" "/dev/fb0"
EndSection
```

The modified file is included within the attached ZIP file in folder **hdmio**. On the next boot, the Petalinux desktop will be displayed on the monitor. To reboot the board you can use command

```
reboot
```

8. Run the *facedetect* demo using DPU. The input data are taken from the HDMI input on the AVNET FMC card a processed data are shown on the monitor connected via HDMI output of the AVNET FMC card.

   a. Copy the precompiled executable file of the *facedetect demo* and the corresponding model for the DPU from the enclosed ZIP package. Connect the board using SFTP and copy to the board folder:

      facedetect/ to **/home/root**

      Source codes of the *facedetect* application are also part of the attached ZIP file.

   b. Make the *facedetect* SW application executable, from the board terminal execute:

      ```
      chmod +x /home/root/facedetect/hdmi_facedet2.elf
      ```

   c. Copy scripts optimizing the data transfers between the DPU and the memory controller from the PC to the home folder on the board. These scripts are available in attached ZIP file. To copy them use SFTP:

      dpu_sw_optimize to **/home/root**

   d. Switch back to the board terminal.

   e. Optimize the data transfers between the DPU and the memory controller, execute:

      ```
      cd /home/root/dpu_sw_optimize/zynqmp
      chmod -R +x *
      ./zynqmp_dpu_optimize.sh
      ```

      NOTE: The scripts are modified compare to scripts provided by Xilinx.

   f. Set needed environment variable pointing the DPU firmware.

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

   g. Connect the HDMI input on the AVNET FMC card. It can be an HDMI camera or an output from a PC. In the case of PC output, just play some video where there are people.

   h. Connect an HDMI monitor to the HDMI output of the AVNET FMC card. The output video signal is 1920x1080p60.

   i. Run **facedetect** demo

      ```
      cd /home/root/facedetect
      ./hdmi_facedet2.elf
      ```

      The result is shown on the monitor.

5 VCU Decoder

The platform includes H.264/H.265 Video Codec Unit (VCU), which is configured to use the decoder. The encoder is disabled, because the currently used FPGA does not have enough free resources for it. To test the decoder, follow these steps.

1. Upload any MP4 H.264 encoded video to the board. Connect the board using SFTP and copy the video file to the board folder **/home/root/**.

The video can be downloaded here, for instance:

http://distribution.bbb3d.renderfarming.net/video/mp4/bbb_sunflower_1080p_60fps_normal.mp4.

The name of the example video is **bbb_sunflower_1080p_60fps_normal.mp4**.

If the board has the Internet connection, the video file can be downloaded directly to the board, from the board terminal execute:

```
cd /home/root
wget http://distribution.bbb3d.renderfarming.net/video/mp4/bbb_sunflower_1080p_60fps_normal.mp4
```

2. Inspect the video file on the board, use command *gst-discover-1.0*:

```
cd /home/root
gst-discoverer-1.0 bbb_sunflower_1080p_60fps_normal.mp4
Analyzing file:///home/root/bbb_sunflower_1080p_60fps_normal.mp4
Done discovering file:///home/root/bbb_sunflower_1080p_60fps_normal.mp4
Missing plugins

Topology:
  container: Quicktime
    audio: AC-3 (ATSC A/52)
    audio: MPEG-1 Layer 3 (MP3)
    video: H.264 (High Profile)

Properties:
  Duration: 0:10:34.533333333
  Seekable: yes
  Live: no
  Tags:
      audio codec: MPEG-1 audio
      maximum bitrate: 165120
      bitrate: 160000
      datetime: 2013-12-16T17:59:32Z
      title: Big Buck Bunny, Sunflower version
      composer: Sacha Goedegebure
      artist: Blender Foundation 2008, Janus Bager Kristensen 2013
      comment: Creative Commons Attribution 3.0 -
http://bbb3d.renderfarming.net
      genre: Animation
      QT atom: buffer of 39 bytes
      container format: ISO MP4/M4A
      has crc: false
      channel mode: joint-stereo
      video codec: H.264 / AVC
```

3. Play the video

   a. Petalinux desktop

```
export DISPLAY=:0.0

gst-launch-1.0 filesrc location=bbb_sunflower_1080p_60fps_normal.mp4 ! \
               qtdemux name=demux demux.video_0 ! h264parse ! \
               omxh264dec ! queue ! videoconvert! Autovideosink
```

   b. X11 forwarding via SSH

```
export DISPLAY=:10.0

gst-launch-1.0 filesrc location=bbb_sunflower_1080p_60fps_normal.mp4 ! \
               qtdemux name=demux demux.video_0 ! h264parse ! \
               omxh264dec ! queue ! videoconvert! Autovideosink
```

Both methods are too slow because of the output video software processing after the VCU decodes the data. To get the true performance of the VCU, displaying of the decoded video data has to be suppressed.

```
gst-launch-1.0 filesrc location=bbb_sunflower_1080p_60fps_normal.mp4 ! \
               qtdemux name=demux demux.video_0 ! h264parse ! \
               omxh264dec ! queue ! videoconvert! \
               fpsdisplaysink text-overlay=0 video-sink=fakevideosink -v
```

6 Automations and Optimizations

Some steps can be automatized to avoid doing them after each boot.

6.1 QoS

To automatize starting the scripts from the **/home/root/dpu_sw_optimize** folder follow next steps:

1. Copy file **sripts/qos.sh** from the provided ZIP file to the **/etc/init.d/** folder on the board, use SFTP. Make it executable:

```
chmod +x /etc/init.d/qos.sh
```

2. Create a link of the **qos.sh** file for the runlevel 5.

```
ln -s /etc/init.d/qos.sh /etc/rc5.d/S99qos
```

6.2 Monitor Optimizations

A monitor connected via the HDMI port on the TE0701 (Petalinux desktop) will go into sleep mode when keyboard and mouse are in idle state. To disable this behavior append an extra section to the file **/etc/X11/xorg.conf**:

```
Section "ServerFlags"
    Option "BlankTime" "0"
EndSection
```

The modified file is included within the attached ZIP file in folder hdmio.

6.3 DPU Firmware

To set a variable pointing the DPU firmware automatically during the boot time, copy file **sripts/dpu_fw.sh** from the provided ZIP file to the **/etc/profile.d/** folder on the board, use SFTP. Make the file executable:

```
chmod +x /etc/profile.d/dpu_fw.sh
```

7 Used Resources

Resource	Utilization		Available	Utilization %	
	All	DPU		All	DPU
LUT	73128	48255	87840	83.25	54.94
LUTRAM	9304	5595	57600	16.15	9.71
FF	138250	97563	175680	78.69	55.53
BRAM	99	84	128	77.34	65.63
URAM	48	46	48	100.00	95.83
DSP	710	690	728	97.53	94.78
IO	62	0	252	24.60	0.00
BUFG	13	0	352	3.69	0.00
MMCM	3	0	4	75.00	0.00

8 Power Consumption

Conditions of the power consumption measurement:

- Measured with wattmeter Electrobock EMF-1
  - Voltage range: 190-276 VAC (accuracy +/-1%)
  - Current range: 0.01-16A (+/-1%)
  - Power range: 0.2-4416 W (+/-1%)
- Measurement covers these components:
  - Power adapter.
  - Connected to the Ethernet.
  - USB connections: keyboard, mouse.
  - Connected to the HDMI monitor, used resolution is 1920x1080p60 (HDMI out on TE0701 - Petalinux desktop).
  - Connected to the HDMI monitor, used resolution is 1920x1080p60 (HDMI out on AVNET FMC card – *facedetect* demo out).
  - Running demo: *facedetect*.
  - Connected UART/JTAG USB.

The power consumption was measured in two situations:

1. The whole system was running without the *facedetect* demo: **11.5 W**.
2. Running the *facedetect* demo: **14.8 W**.

9 Package Content

```
├── dpu_sw_optimize
│   └── zynqmp
├── facedetect
│   ├── densebox_640_360
│   └── hdmi_facedet2.elf
├── hdmio
│   ├── hdmio.elf
│   ├── xorg.conf
│   └── 01hdmio.sh
├── ip_lib
│   ├── axis_vid_det_1_0
│   ├── hblank_det_1_0
│   ├── im_hdmi_in
│   ├── im_hdmi_out
│   └── video_io_to_hdmi
├── petalinux
│   └── project-spec
│       ├── meta-user
│       │   └── conf
│       │       └── user-rootfsconfig
│       ├── recipes-bsp
│       │   ├── device-tree
│       │   └── uboot-device-tree
│       └── recipes-vitis-ai
├── scripts
│   ├── dpu_fw.sh
│   └── qos.sh
├── src
│   ├── hdmi_facedet2_system.ide.zip
│   └── hdmio_system.ide.zip
└── vivado
    └── te0820_EV_fast_track_vcu_vdma.tcl
```

10 References

[1]. Trenz Electronic, „MPSoC Module with Xilinx Zynq UltraScale+ ZU4EV-1I, 2 GByte DDR4 SDRAM, 4 x 5 cm,“ [Online]. Available: https://shop.trenz-electronic.de/en/TE0820-05-4DI21MA-MPSoC-Module-with-Xilinx-Zynq-UltraScale-ZU4EV-1I-2-GByte-DDR4-SDRAM-4-x-5-cm.

[2]. Trenz Electronic, „Carrier Board for Trenz Electronic 4 x 5 Modules,“ [Online]. Available: https://shop.trenz-electronic.de/en/TE0701-06-Carrier-Board-for-Trenz-Electronic-4-x-5-Modules.

[3]. AVNET, „AES-FMC-HDMI-CAM-G,“ [Online]. Available: https://www.avnet.com/shop/us/products/avnet-engineering-services/aes-fmc-hdmi-cam-g-3074457345635221625/.

[4]. Trenz Electronic, UTIA, „TE0820 test board Vitis AI Tutorial,“ [Online]. Available: https://wiki.trenz-electronic.de/display/PD/TE0820+test+board+Vitis+AI+Tutorial.

Disclaimer

This disclaimer is not a license and does not grant any rights to the materials distributed herewith. Except as otherwise provided in a valid license issued to you by UTIA AV CR v.v.i., and to the maximum extent permitted by applicable law:

(1) THIS APPLICATION NOTE AND RELATED MATERIALS LISTED IN THIS PACKAGE CONTENT ARE MADE AVAILABLE "AS IS" AND WITH ALL FAULTS, AND UTIA AV CR V.V.I. HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and

(2) UTIA AV CR v.v.i. shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under or in connection with these materials, including for any direct, or any indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or UTIA AV CR v.v.i. had been advised of the possibility of the same.

Critical Applications:
UTIA AV CR v.v.i. products are not designed or intended to be fail-safe, or for use in any application requiring fail-safe performance, such as life-support or safety devices or systems, Class III medical devices, nuclear facilities, applications related to the deployment of airbags, or any other applications that could lead to death, personal injury, or severe property or environmental damage (individually and collectively, "Critical Applications"). Customer assumes the sole risk and liability of any use of UTIA AV CR v.v.i. products in Critical Applications, subject only to applicable laws and regulations governing limitations on product liability.