Lab3: Build a Complete Embedded IA System (2025)

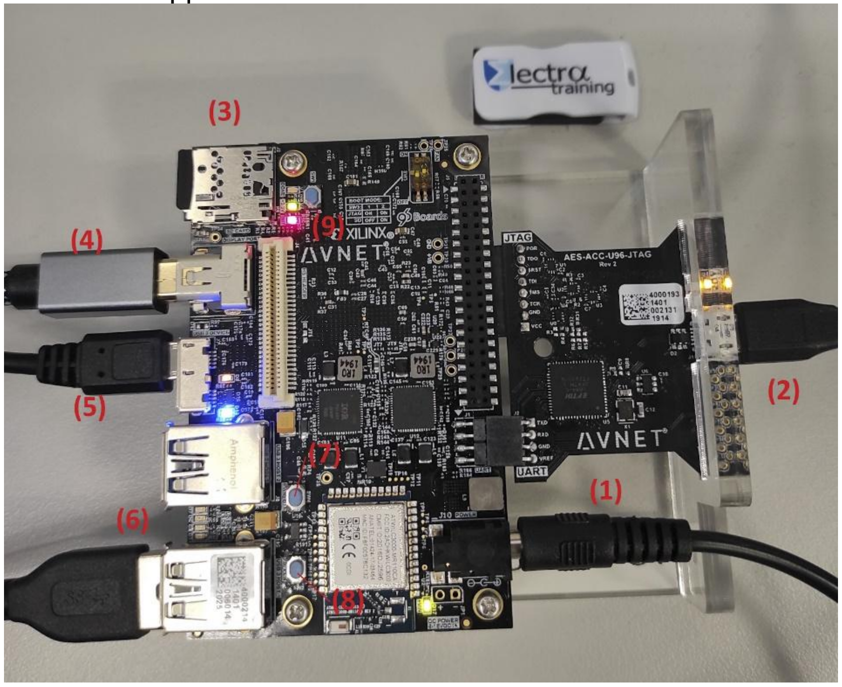
In this lab we will work on different aspects of optimizing implementations of Deep Learning models on an embedded board.

Part A. Setup the embedded Platform

This lab is based on hackester.io blog tutorial “Vitis-AI 1.4 Flow for Avnet VITIS Platforms” for the Ultra96v2 boards. https://www.hackster.io/AlbertaBeef/vitis-ai-1-4-flow-for-avnet-vitis-platforms-5ebc3f

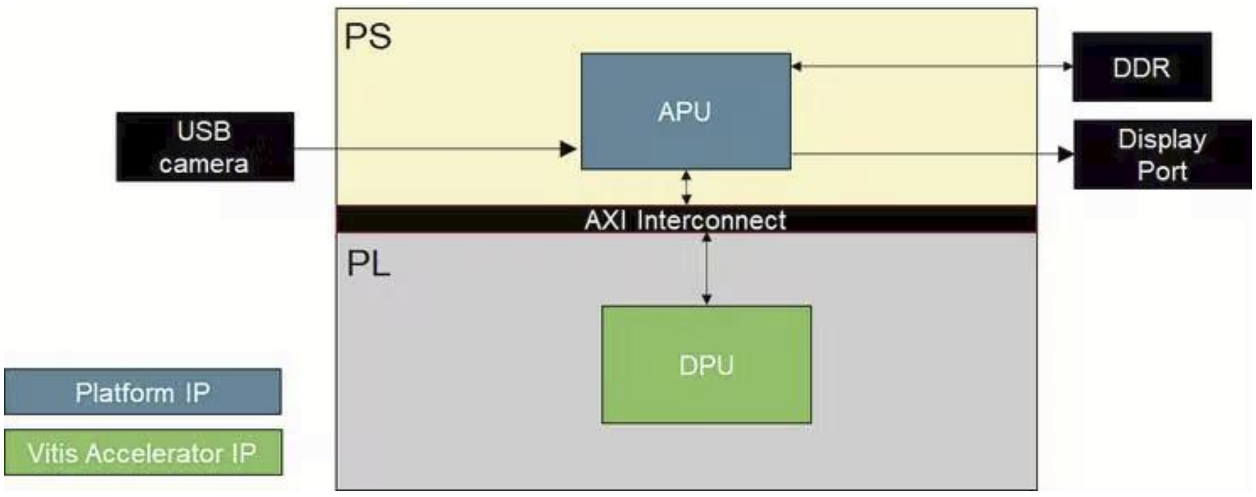
A.1 Introduction

These are detailed instructions for targeting the Xilinx Vitis-AI 1.4 flow into the Ultra96v2 development board. This guide will describe how to download and install the pre-built SD card images and execute the AI applications on the hardware.

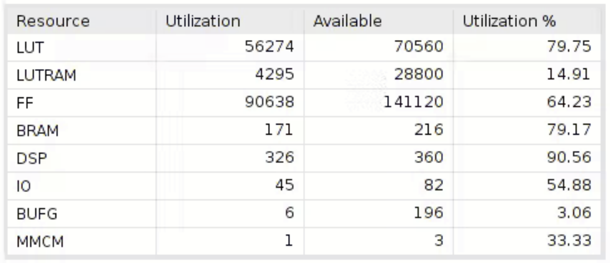


A.2. Design Overview

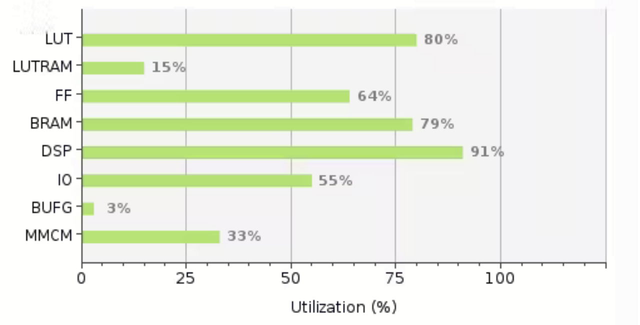
The following block diagram illustrates the hardware designs included in the pre-built images (the image from hackester.io)



These designs were built using the AMD Vitis flow with the following DPU (AMD Deep Learning Processing Unit) configurations for the Ultra96v2. One B2304 (low RAM usage) DPU running at 200MHz/400MHz. When the DPU is loaded, the total resource utilization of the FPGA devices is:



Resource	Utilization	Available	Utilization %
LUT	56274	70560	79.75
LUTRAM	4295	28800	14.91
FF	90638	141120	64.23
BRAM	171	216	79.17
DSP	326	360	90.56
IO	45	82	54.88
BUFG	6	196	3.06
MMCM	1	3	33.33



The system runs Petalinux (A Linux distribution based in Yocto specific for AMD-Xilinx) who uses the quad core cortex A53 processor from APU (Application Processor Unit) inside the PS (Processing System) part of MPSoC. The Programmable Logic Part of the MPSoC device is mainly used to deploy the DPU (Deep Learning Processing Unit) provided by AMD-Xilinx.

Part B. Execute the AI applications on hardware (Running Preconfigured Demos)

Step 1 - Create the SD card

Pre-built SD card images have been provided for the Ultra96-V2 Development Board (uz96v2_sbc_base). An image is already available at the file repository for the course (HPA4ML_Lab3_image.7z)

The SD card image contains the hardware design (BOOT.BIN, dpu.xclbin), as well as the petalinux images (boot.scr, image.ub, rootfs.tar.gz). It is provided in image (IMG) format, and contains two partitions:

- BOOT – partition of type FAT (size=400MB)
- ROOTFS – partition of type EXT4

The first BOOT partition was created with a size of 400MB, and contains the following files: BOOT.BIN, boot.scr, image.ub, init.sh, platform_desc.txt, dpu.xclbin, arch.json

The second ROOTFS partition contains the rootfs.tar.gz content, and is pre-installed with the Vitis-AI runtime packages, as well as the following directories:

- /home/root/install/vitis-ai-runtime-1.4.0, which includes - Vitis-AI 1.4 runtime packages
- /home/root/dpu_sw_optimize
- /home/root/Vitis-AI, which includes - pre-built VART samples - pre-built Vitis-AI-Library samples

Once downloaded, and extracted, the ".img" file can be programmed (burned) to a 16GB micro-SD card.

Step 2 - Installing the Vitis-AI runtime packages

Some configuration steps only need to be performed once (after the first boot), including the following:
You can download the image including step 2 from Moodle page (HPA4ML_Lab3_image.7z). This step takes time and needs an internet connection.

2.1. Boot the target board with the micro-SD card that was created in the previous section
2.2. After boot, copy the dpu.xclbin file to the /usr/lib directory

```
$ cp /media/sd-mmcblk0p1/dpu.xclbin /usr/lib/.
```

2.3. Next, install the Vitis-AI 1.4 runtime packages

```
$ cd ~/install/vitis-ai-runtime-1.4.0
$ source ./setup.sh
```

This script will perform the following steps

- install the packages (taken from https://github.com/Xilinx/Vitis-AI/tree/v1.4/setup/mpsoc/VART/2021.1)
- modify the /etc/vart.conf to point to the /usr/lib/dpu.xclbin

2.4. Launch the dpu_sw_optimize.sh script

```
$ cd ~/dpu_sw_optimize/zynqmp
$ source ./zynqmp_dpu_optimize.sh
```

This script will perform the following steps:

- Auto resize SD card's second (EXT4) partition
- QoS configuration for DDR memory

2.5. Validate the Vitis-AI runtime with the `xdputil` utility.

```
$ xdputil query
```

EXERCISE: Explain the output of the xdputil command

Step 3 - Execute the AI applications on hardware

You will start here to run the examples

The steps described in this section need to be done after each boot in order to run the AI examples.

3.1. If running on the Ultra96-V2 development board, re-launch the dpu_sw_optimize.sh script to optimize the DDR's QoS configuration for the DisplayPort monitor (otherwise, you may experience glitches on the monitor).

```
$ cd ~/dpu_sw_optimize/zynqmp
$ source ./zynqmp_dpu_optimize.sh
```

This script will perform the following steps:

- Auto resize SD card's second (EXT4) partition
- QoS configuration for DDR memory

3.2. [Optional] Disable the dmesg verbose output:

```
$ dmesg -D
```

This can be re-enabled with the following:

```
$ dmesg -E
```

3.3. Define the DISPLAY environment variable

```
$ export DISPLAY=:0.0
```

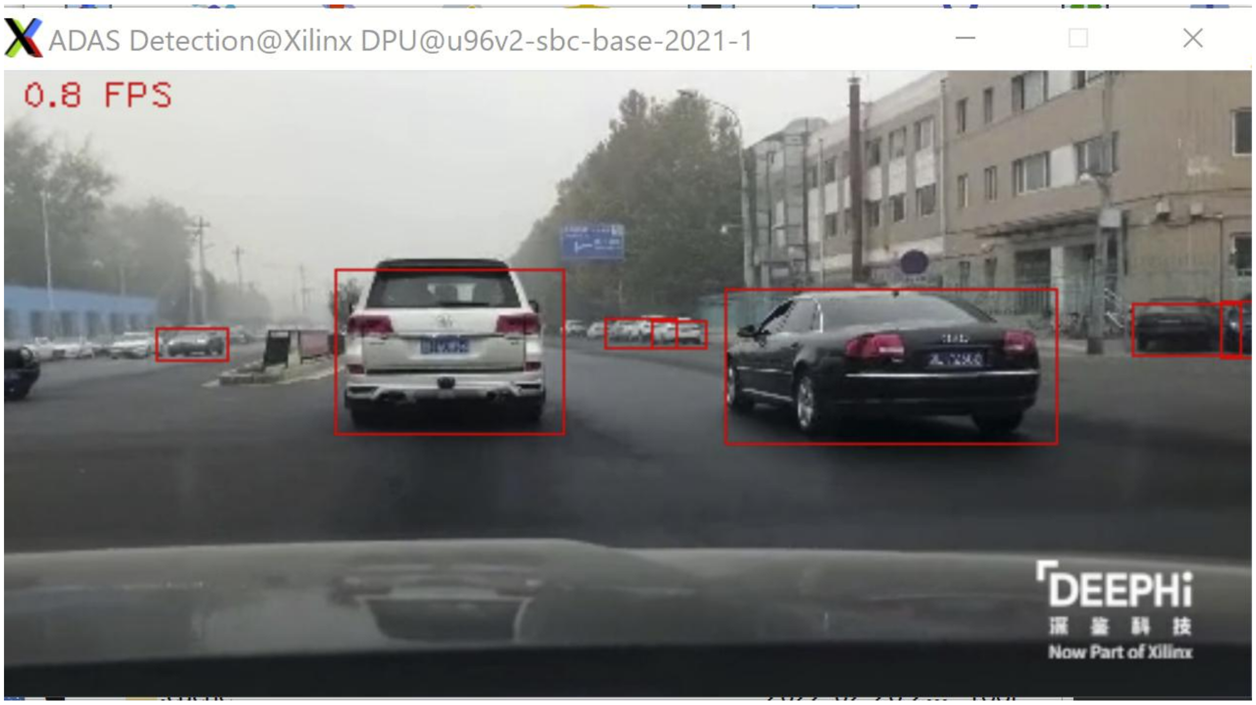
3.4. Change the resolution of the DP monitor to a lower resolution, such as 640x480

```
$ xrandr --output DP-1 --mode 640x480
```

3.5. Launch the C++ based VART sample applications

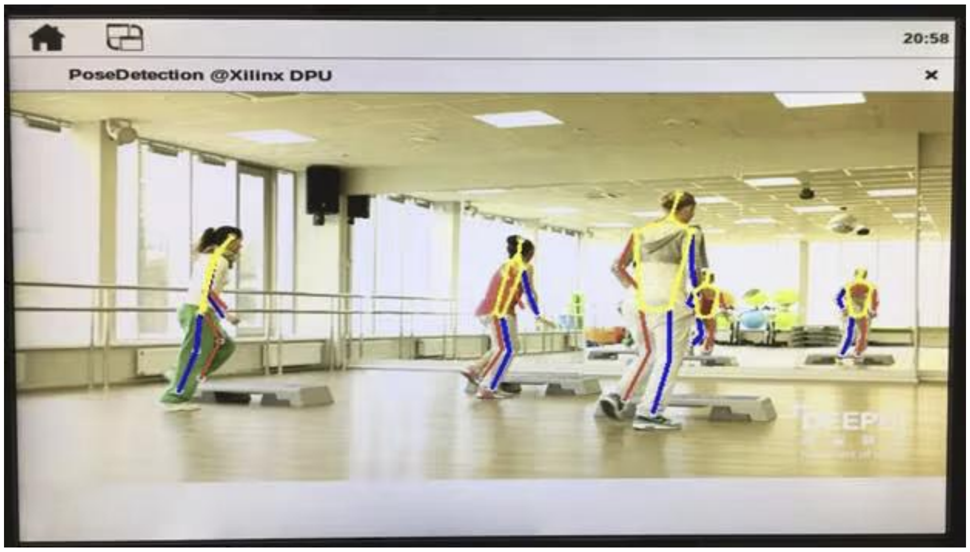
3.5.a. Launch the adas_detection application

```
$ cd ~/Vitis-AI/demo/VART/adas_detection

$ ./adas_detection ./video/adas.avi
/usr/share/vitis_ai_library/models/yolov3_adas_pruned_0_9/yolov3_adas_
pruned_0_9.xmodel
```



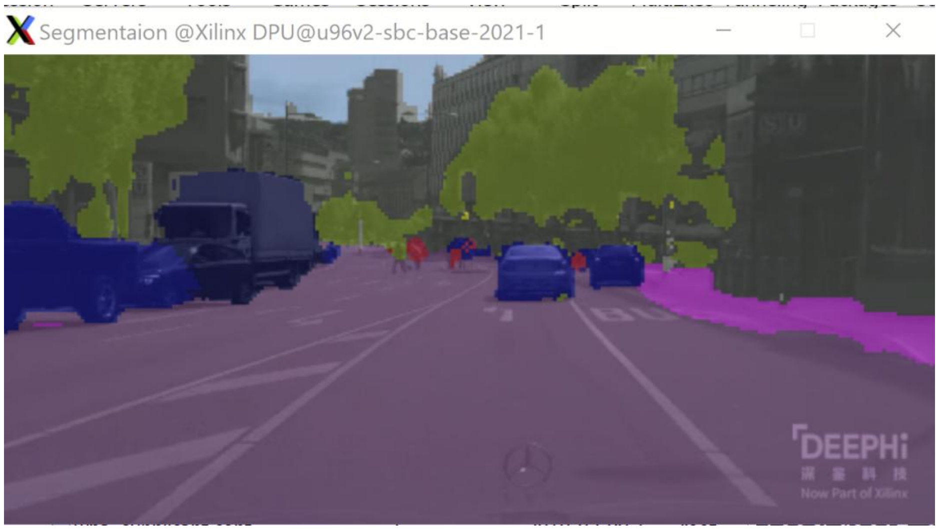
3.5.b. Launch the pose_detection application

```
$ cd ~/Vitis-AI/demo/VART/pose_detection

$ ./pose_detection ./video/pose.mp4
/usr/share/vitis_ai_library/models/sp_net/sp_net.xmodel
/usr/share/vitis_ai_library/models/ssd_pedestrian_pruned_0_97/ssd_pede
strian_pruned_0_97.xmodel
```



3.5.c. Launch the segmentation application

```
$ cd ~/Vitis-AI/demo/VART/segmentation

$ ./segmentation ./video/traffic.mp4
/usr/share/vitis_ai_library/models/fpn/fpn.xmodel
```



3.6. Launch the python based VART sample applications

3.6a.1. Launch the resnet50_mt_py application

```
$ cd ~/Vitis-AI/demo/VART/resnet50_mt_py
$ python3 ./resnet50.py 8
/usr/share/vitis_ai_library/models/resnet50/resnet50.xmodel
```

The python script will apply the resnet50 to a batch of images. When done, you will see a result similar to the following (for Ultra96-V2):

```
FPS=27.00, total frames = 2880.00 , time=106.653120 seconds
```

3.6b. Launch the inception_v1_mt_py application

```
$ cd ~/Vitis-AI/demo/VART/inception_v1_mt_py
$ python3 ./inception_v1.py 8
/usr/share/vitis_ai_library/models/inception_v1/inception_v1.xmodel
```

The python script will apply the resnet50 to a batch of images. When done, you will see a result similar to the following (for Ultra96-V2):

```
71.98 FPS
```

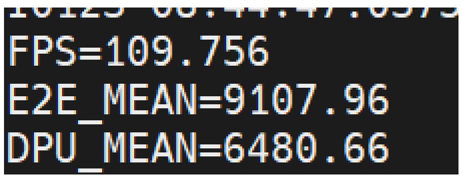
3.7. Launch the Vitis-AI-Library based sample applications

a. Launch the face_detect application with both variants of the densebox model (specify “0” as second argument, to specify the USB camera)

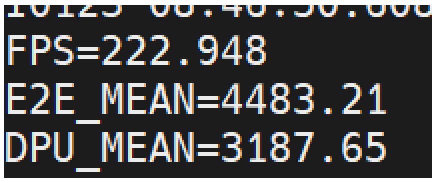
```
$ cd ~/Vitis-AI/demo/Vitis-AI-Library/samples/facedetect
$ ./test_video_facedetect densebox_640_360 0
$ ./test_video_facedetect densebox_320_320 0
```

b. Compare the performance of each variant of the densebox models

```
$ ./test_performance_facedetect densebox_640_360
./test_performance_facedetect.list
$ ./test_performance_facedetect densebox_320_320
./test_performance_facedetect.list
```



```
FPS=109.756
E2E_MEAN=9107.96
DPU_MEAN=6480.66
```



```
FPS=222.948
E2E_MEAN=4483.21
DPU_MEAN=3187.65
```

3.8. Launch the Xilinx Developer demo

a. Launch the "Point Cloud 3D detection" demo, with output showing images with 3D annotations

```
cd ~/xilinx_developer/ppdemo1216
./demo ./ppd/vlist.txt ./ppd/ 1
```

b. Launch the "Point Cloud 3D detection" demo, with output showing annotated point cloud data

```
./demo ./ppd/vlist.txt ./ppd/ 2
./demo ./ppd/vlist.txt ./ppd/ 4
```