C5 SETI: Host Tools, Apptainer, Vitis AI and KV260

Course reference and personal-machine setup

Télécom Paris — Master SETI

2026–2027

This reference accompanies the lab handouts. Use each handout for its exact exercise commands, model filenames and stopping rules. Commands below identify their execution location. Examples use Bash; open `bash` before sourcing the course helpers if your login shell is Zsh.

Contents

1 The KV260 and its data path **2**

2 Choose the right environment **2**

3 Teaching-machine setup **3**

4 Prepare a personal computer **3**
4.1 Recommended route: Linux or a remote teaching host . . . 3
4.2 Obtain one pinned Vitis AI image, only if needed . . . 4

5 Enter the container and verify the tools **4**
5.1 Preferred lab launcher . . . 4
5.2 Optional manual entry when already familiar with Apptainer . . . 4

6 Download raw datasets once **5**

7 Understand the two KV260 software stacks **6**

8 Connect to the assigned board **6**
8.1 Address and account . . . 6
8.2 Read-only identification before deployment . . . 7

9 Mapping, numerical correctness and performance **7**
9.1 DPU mapping . . . 7
9.2 Measure distinct quantities . . . 7

10 HLS workflow and common failures **8**

11 Keep a reproducible, small working tree **8**

12 Official references **9**

A Board-side quick reference **9**
A.1 Bring-up, SSH and serial console . . . 9
A.2 Checks and measurements on the board . . . 9

1 The KV260 and its data path

The Kria K26 is a system-on-module built around a Zynq UltraScale+ MPSoC. The KV260 starter kit combines that module with a carrier board and its external connections. Within the device, the **processing system (PS)** contains Arm processors running Linux; the **programmable logic (PL)** implements an accelerator circuit. They cooperate through control interfaces and shared memory.

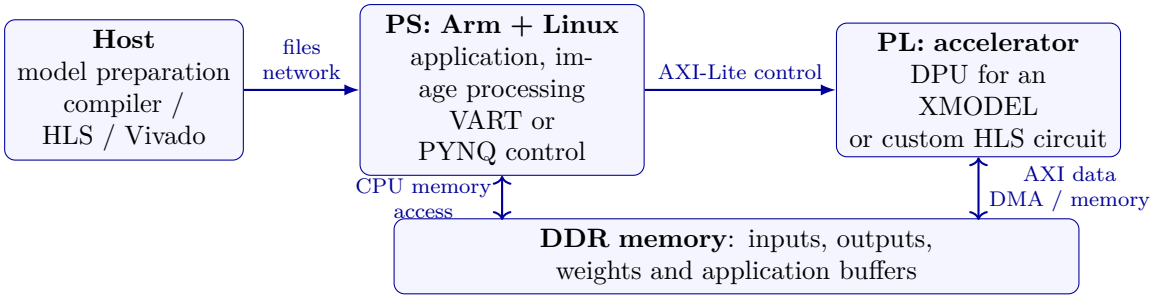


Figure 1: Control and image data follow different paths. Data buffers reside in DDR; FPGA circuits use AXI interfaces to access or transfer them. Ethernet carries host files, not an implicit connection into an accelerator's input port.

Training, quantization and compilation run on the host. On the board, the Arm application prepares inputs, invokes the matching runtime, and interprets outputs. The PL performs the accelerated operations. **AXI-Lite** is used for control registers; data interfaces transfer larger buffers. An **AXI DMA** connects memory buffers to a streaming accelerator. Moving data can matter as much as arithmetic, so input/output shapes, data types and numerical scales are part of the interface.

File or object	Role
`.bit`	FPGA configuration containing a particular circuit.
`.hwh`	Hardware description used by PYNQ, including IP/register information; must match the bitstream.
`.xmodel`	Compiled neural-network graph/weights for a compatible existing DPU; it is not an FPGA bitstream.
PYNQ overlay	Python-controlled hardware configuration and metadata, used with the matching application and interfaces.

TP01 is host-only. TP02–TP05 use the PetaLinux/VART image and its DPU. TP06–TP07 use Ubuntu/PYNQ with custom HLS accelerators. These are distinct software/hardware stacks; do not combine their images or exchange configuration files. Identify the running image and installed runtime using the read-only checks in Section 8.2; Section 8 gives connection accounts and the serial-console route. Then use the matching handout's preflight and numerical test, not a successful file copy alone. The KV260 User Guide (UG1089) describes the platform and interfaces.

2 Choose the right environment

Work	Host tools	Target, when required
TP01–TP05	Vitis AI 3.5 PyTorch container: training, NNDCT quantization, XIR and compiler	TP02–TP05: course PetaLinux/VART image with matching DPU
TP06–TP07	Vitis HLS and Vivado, installed separately; Python for data preparation	Course Ubuntu/Kria-PYNQ image, matching bitstream and hardware metadata

The Vitis AI container does not replace Vitis HLS/Vivado, and a normal PyTorch installation does not provide NNDCT or the DPU compiler. CPU-only training needs no host GPU. The published CPU image cannot gain GPU acceleration simply by adding an Apptainer GPU flag.

TP06 studies HLS Conv2D. TP07 builds the compact MNIST CNN. Keep each lab's generated headers, models and overlays together.

3 Teaching-machine setup

The sourced course helpers require Bash. In each new host terminal, type `bash` and press Enter as a separate action. Wait for its prompt before pasting any commands below; do not include the shell transition in a multiline paste. Scripts invoked with `bash script.sh` already select Bash. The example PATH below is for the teaching hosts only. After the Bash prompt appears, clone the student repository or extract a release lab archive. For a full checkout:

```
git clone https://gitlab.telecom-paris.fr/alaa.mazouz/c5_seti_students.git
cd c5_seti_students
export COURSE_ROOT="$(pwd -P)"
export PATH="/comelec/softs/bin:$PATH"
command -v apptainer
apptainer --version
```

IT's shared Vitis AI 3.5 CPU image is `/comelec/softs/opt/vitis-ai-pytorch-cpu_ubuntu2004-3.5.0.306.sif`. The lab helpers select it automatically; no `VAI_SIF` export is needed on a configured teaching machine. If you need an existing pinned image elsewhere, set its actual path before initialization:

```
export VAI_SIF=/absolute/path/to/existing-course-image.sif
test -r "$VAI_SIF"
apptainer inspect "$VAI_SIF"
sha256sum "$VAI_SIF"
```

An explicit `VAI_SIF` wins over the shared path. The helpers otherwise try the installed image above, the legacy shared location `/comelec/softs/share/vitis_ai/vitis-ai-pytorch-cpu.sif`, then a private copy at `/tmp/$USER/vitis-ai-pytorch-cpu.sif`. An unreadable image is an access problem: ask IT or choose a readable pinned image; do not replace the shared file. A temporary image can disappear after cleanup or reboot. Do not download another copy when a working one already exists.

4 Prepare a personal computer

4.1 Recommended route: Linux or a remote teaching host

Use an x86-64 Linux machine compatible with the pinned Vitis AI image. Install Git, Bash, OpenSSH client, a C/C++ build toolchain and Python through your operating system's package manager. Install Apptainer using its official instructions for your distribution; administrator policy and user-namespace support determine the permitted installation method. Check `uname -m`, disk space and `apptainer --version` before downloading. Keep room for the image, conversion/cache space, datasets and experiment outputs; inspect `df -h` rather than assuming a small home quota will suffice.

On Windows or macOS, the simplest course-compatible route is SSH to a configured Linux teaching host. A Linux VM can be used if it satisfies Apptainer and image architecture requirements. WSL2, Apple Silicon emulation, USB camera forwarding and board discovery through a VM are not validated course configurations. Access to a remote host does not imply access to the campus board network; use the institution's approved access route.

Install licensed Vitis HLS/Vivado separately for TP06–TP07, using the release and FPGA part specified by the selected handout. Accept the vendor licence terms and configure the required licence locally. Do not substitute the newest tool release without checking project compatibility. Ordinary C++ compilation alone does not establish HLS synthesis or RTL co-simulation correctness.

4.2 Obtain one pinned Vitis AI image, only if needed

The course uses the CPU PyTorch image tagged `ubuntu2004-3.5.0.306`. On your Linux host, choose writable storage; unlike `/tmp`, the example below can survive reboot:

```
mkdir -p "$HOME/c5-tools" "$HOME/c5-scratch/apptainer-cache" \
  "$HOME/c5-scratch/apptainer-tmp"
export APPTAINER_CACHEDIR="$HOME/c5-scratch/apptainer-cache"
export APPTAINER_TMPDIR="$HOME/c5-scratch/apptainer-tmp"
export VAI_SIF="$HOME/c5-tools/vitis-ai-pytorch-cpu.sif"
# Run only when VAI_SIF does not already contain a verified image.
test ! -e "$VAI_SIF" && apptainer pull "$VAI_SIF" \
  docker://xilinx/vitis-ai-pytorch-cpu:ubuntu2004-3.5.0.306
apptainer inspect "$VAI_SIF"
sha256sum "$VAI_SIF"
```

Record the checksum with your experiment. A checksum you compute yourself records identity; authenticity requires comparison with a trusted course-supplied checksum. If interrupted, inspect the incomplete file and preserve or rename it before retrying; do not overwrite an existing verified image. Do not install packages into the shared image. AMD also documents a Docker workflow, but course launcher scripts use Apptainer; a manual Docker setup requires equivalent mounts, working directory and environment activation.

5 Enter the container and verify the tools

5.1 Preferred lab launcher

On the host, TP02 illustrates the launcher pattern:

```
cd "$COURSE_ROOT/TP02_quantization"
source scripts/00_source_env.sh
bash scripts/02_enter_container.sh
```

Wait for `[TP02] Container ready`, then run the following in that container shell:

```
# container
python3 -c 'import sys, torch, pytorch_nndct, xir; print(sys.version); print(torch.
    __version__)'
command -v vai_c_xir
conda list pytorch
```

Follow the selected lab README for its launcher name: TP05 uses `scripts/enter_container.sh`. Launchers set the lab working directory and activate `vitis-ai-pytorch`. The reference toolchain uses Python 3.8/PyTorch 1.13.1; record actual versions. Import success is a prerequisite, not evidence that a model compiles or runs on hardware.

5.2 Optional manual entry when already familiar with Apptainer

On the host, with `COURSE_ROOT` and `VAI_SIF` set:

```
export C5_DATA="${C5_DATA:-$COURSE_ROOT/datasets}"
mkdir -p "$C5_DATA"
apptainer exec --bind "$COURSE_ROOT:$COURSE_ROOT" \
  --bind "$C5_DATA:$C5_DATA" --pwd "$COURSE_ROOT" \
  "$VAI_SIF" bash
```

After the container prompt appears:

```
# container
source /opt/vitis_ai/conda/etc/profile.d/conda.sh
conda activate vitis-ai-pytorch
python3 -c 'import torch, pytorch_nndct, xir; print(torch.__version__)'
```

A bind exposes host files at the indicated container path; writes there persist on the host. If a dataset is a symlink, its target must also be visible. For a preinstalled external read-only source, bind its real directory read-only too; the course launchers handle the configured `C5_DATASETS` source. Use `exit` to return to the host. Do not start a nested container from a container shell.

6 Download raw datasets once

Setting	Meaning
`C5_DATA`	Explicit absolute writable raw-data root; overrides discovery.
Course checkout	Default `datasets/` beside the lab folders.
Standalone lab	Fallback `${XDG_CACHE_HOME:-$HOME/.cache}/c5/datasets`; adjacent extracted labs can share a sibling `datasets/`.
`C5_DATASETS`	Optional preinstalled source, normally `/comelec/softs/share/c5_datasets`; separate from the writable root.

IT provides the SIF, not the datasets. The shared dataset source above is optional. For TP01–TP03, prepare FashionMNIST and CIFAR-10 with the instructor's verified instructions and reuse a persistent per-user `C5_DATA`; MNIST and RPS are for later labs or optional choices. Set overrides before sourcing the lab environment.

Keep the existing lab preparation scripts as the default. If the official FashionMNIST download is slow or fails, this verified fallback runs from the TP01 folder inside its activated container, after the host environment initialization has created `data/` links:

```
# container, TP01 folder; fallback only
python3 - <<'PY'
from torchvision import datasets as d
mirror = ('https://gitlab.telecom-paris.fr/api/v4/projects/10956/'
          'packages/generic/c5-datasets/2026.11.16/')
d.FashionMNIST.mirrors = [mirror] + d.FashionMNIST.mirrors
d.FashionMNIST('data', download=True)
PY
```

The override lasts only for that Python process. It preserves the official mirrors as fallbacks and torchvision's original MD5 checks. The package contains the MIT license (2017 Zalando SE), attribution and SHA-256 checksums. Existing complete data need no new download; incomplete/corrupt user data require diagnosis.

If the official CIFAR-10 download is slow, this separate verified mirror uses the unchanged torchvision archive, MD5 verification, extraction and persistent cache. Run it inside the activated course container after normal setup:

```
# container; optional CIFAR-10 fallback
python3 - <<'PY'
import os
from torchvision.datasets import CIFAR10
CIFAR10.url = ('https://gitlab.telecom-paris.fr/api/v4/projects/10956/'
               'packages/generic/c5-datasets/2026.11.16-cifar10/'
               'cifar-10-python.tar.gz')
CIFAR10(os.environ['C5_DATA'], download=True)
PY
```

This URL override lasts only for this process. Normal lab scripts retain the official source and reuse the same raw files across TP01–TP03. If GitLab is unavailable, follow the normal official-source instructions. The instructor authorized the mirror; the original redistribution terms remain unresolved, and no new license is asserted.

For optional full-course preparation, inside the activated Vitis AI environment and from the full course root (this can download datasets):

```
bash scripts/prepare_datasets.sh
```

```
python3 tools/shared_datasets/profile_datasets.py \
  --output work/dataset_profiles.json
```

TP05 prepares RPS through its own preparation command and reuses cached raw archives/extraction. Existing user data must stay in place. A corrupt archive must be diagnosed, not deleted or overwritten automatically. Keep lab-specific splits, calibration choices, checkpoints, bundles and reports within their own experiment directories; shared raw data does not mean shared train/test decisions. Never use the held-out test set for calibration or hyperparameter selection. Standalone archives include their required local utilities; follow their README instead of assuming root-only preparation commands exist.

7 Understand the two KV260 software stacks

The host Vitis AI 3.5 toolchain and the historical course PetaLinux Vitis AI 3.0 runtime are different environments. Host compilation is not proof of compatibility. For TP02–TP05, compare the compiled model's architecture/fingerprint, tensor layouts, batch size and runtime requirements with the actual board. The supplied DPU target fingerprint is `0x101000056010407`; this is a compilation configuration, not a live measurement of your board.

TP06–TP07 use Ubuntu/PYNQ and custom HLS overlays. Their `.bit` and `.hwh` files must come from the same design build, and the Python app, register map, DMA buffers and numerical scales must match. A PYNQ overlay is not an XMODEL. Do not change SD images, firmware or accelerator applications outside the matching documented deployment procedure.

8 Connect to the assigned board

8.1 Address and account

Each board answers on the network to a name of the form `maquette-xx`: use the name of the board you are given (the `KV260-xx` label on a board is an inventory number, not its network name). The examples use `maquette-07`; replace it by the name of your board, or by its IP address.

```
# Host: TP02--TP05, course PetaLinux image
export BOARD=root@maquette-07
# Host: TP06--TP07, course Ubuntu/PYNQ image
export BOARD=ubuntu@maquette-07
# Alternative, when the lab's SD image/account is known:
export BOARD=maquette-07
export BOARD_USER=ubuntu
# Direct address, when assigned/confirmed:
export BOARD=ubuntu@192.0.2.57
```

The address 192.0.2.57 is a documentation example, not a usable board assignment. An account explicitly included in `BOARD` takes precedence over `BOARD_USER`; otherwise the override or lab-specific image account is used. Generic helpers do not guess root when no image account is known.

On the host, check name resolution with `getent hosts maquette-07`, then connect using the appropriate account. Verify the host-key fingerprint through an independent trusted channel before accepting it. After a known re-image, confirm the new key before removing only that board's stale known-host entry. Never disable host-key checking globally. Passwords or private SSH keys do not belong in scripts or Git.

For an assigned board that cannot be reached, inspect its serial-console connection (the board-side appendix gives the course procedure). Device names and permissions vary; identify the correct interface and serial parameters for the image instead of opening a wildcard that may match several devices. Read the board's address with `ip -4 addr`; do not scan for unassigned boards or change network settings.

8.2 Read-only identification before deployment

After connecting to the assigned board, run the checks for its image:

```
# On either board image:
uname -a
cat /etc/os-release
# PetaLinux/DPU image, if installed:
xdputil query
xdputil status
# Ubuntu/PYNQ image:
python3 -c 'import pynq; print(pynq.__version__)'
```

Record the outputs. On the board, enter `bash` (or `exec bash -l`) before handout commands using `source`; a noninteractive SSH command needs an explicit `bash -lc` wrapper. The current directory is explicit in `source ./setup_env.sh`. An absent command is a missing prerequisite, not a reason to load arbitrary firmware. Use the handout's deployment wrapper to copy the complete matching bundle, including evaluation inputs and metadata. Default board directories are lab-specific beneath `/home/root/c5/` or `/home/ubuntu/c5/`; consult the selected lab's environment file for its override. Copying only an XMODEL or bitstream can omit required preprocessing or runtime files.

9 Mapping, numerical correctness and performance

9.1 DPU mapping

Use the lab's XIR inspection/checker and preserve compiler diagnostics for both PTQ and QAT models. The strict TP02 gate requires batch one and exactly one DPU neural-computation partition. It permits only the specifically checked terminal DPU-output `fix2float` conversion with valid topology and matching shapes; this does not allow arbitrary CPU neural-network operations. Input constants and boundary conversion must not be confused with CPU-assigned layers.

Parameter counts, nonzero counts and dense MACs are static model quantities; XMODEL bytes and partitioning describe compiled artifacts. They are not measured FPGA LUT/FF/BRAM/DSP utilization. Models on the same fixed DPU reuse its hardware; MAC savings alone do not establish fabric-resource savings.

Check input dtype, tensor layout (for example NHWC), quantization scale/fix-point metadata, output order and class mapping against the shipped runner. Do not substitute a generic zero-input timing harness for accuracy evaluation. Preprocessing and integer rounding must match the lab implementation.

9.2 Measure distinct quantities

Use the provided lab profiling scripts and their documented arguments. On a compatible DPU board, `xdputil benchmark --help` describes that installed version's benchmark interface. A subgraph benchmark omits application preprocessing, camera capture, rendering and usually other end-to-end work. Record warm-up, timed iterations, batch size, thread count and included work when reporting latency or throughput.

Where installed, `vaitrace` and the Vitis AI Profiler can reveal CPU/DPU execution and transfer costs; use the lab wrapper because tracing options and permissions vary with runtime version. A trace can perturb timing. For CPU comparisons, keep dataset, batch, threads and timing scope fixed. Unstructured zeros do not automatically reduce dense MACs or runtime; a structured smaller model can reduce dense work.

Report accuracy and macro recall with their metric names and data split; they are not interchangeable. For camera demos, preserve the documented ROI, resize, channel order and normalization. Missing-camera fallback can test an image path, but does not establish camera latency or live FPS.

Energy per inference is measured energy over the measured number of inferences. Average power divided by throughput is equivalent only for the same measurement interval and workload. State sensor scope, sampling method and whether idle power was subtracted. The course PetaLinux reading is labelled `SOM total power`; it excludes the carrier and USB camera. For the helpers' mean-power estimate, $P_{\text{dyn}} = \max(P_{\text{load}} - P_{\text{idle}}, 0)$ and $E_{\text{dyn}} \simeq 1000 P_{\text{dyn}} t/N$ in mJ/input, using the same measured workload interval t and counted inputs N. The result is baseline-subtracted SOM energy, not isolated DPU energy. Helper headings labelled `board power` refer to this SOM sensor. Keep raw samples, sample counts, units and interval metadata; do not combine power from one workload with camera FPS from another. If the image has no supported sensor access, report energy as unavailable; do not invent a reading or substitute a datasheet estimate for a measurement.

10 HLS workflow and common failures

Run HLS/Vivado commands on a host where those licensed tools are installed, separately from the AI container. From the selected TP06/TP07 folder, source the supplied helper before the handout's CSIM command:

```
source hls/setup_csim_env.sh
command -v vitis_hls
command -v vivado
```

The helper cleans compiler search paths and handles the teaching-machine linker setup. In particular, empty or relative compiler-path entries can cause GCC to interpret a local `specs/` directory as a spec file. Use the supplied helper rather than renaming exercise directories or patching system GCC. CSIM checks C/C++ behavior; synthesis checks hardware generation; RTL co-simulation checks generated RTL against the testbench; only real deployment checks the board integration.

Symptom	First check
Apptainer not found	Host installation/PATH; a broken launcher symlink is not an installed runtime.
No torch/NNDCT import	Activate `vitis-ai-pytorch` inside the correct image; inspect Python path.
Dataset redownload or missing path	Same absolute `C5_DATA`, integrity and bind visibility, including symlink targets.
Compiler success but board rejection	Actual DPU fingerprint, runtime version, model layout and complete bundle.
SSH authentication failure	Assigned hostname, SD-image account and approved access; no guessed suffixes.
PYNQ load/DMA failure	Matched bit/hwh pair, app/register contract, buffer shapes and runtime prerequisites.
No camera	Permissions/device selection and documented image fallback; do not report live-camera results.

11 Keep a reproducible, small working tree

Record release tag, tool versions, image checksum, dataset identity, seed, preprocessing, calibration subset and exact commands. Preserve logs outside the publishable source. Never commit raw data, container images, newly trained checkpoints, XMODELs, HLS/Vivado builds or credentials. Supplied reference teaching inputs are an explicit exception, not permission to add new experiment outputs.

Before cleanup, inspect `du -sh` on the exact paths you own. Remove only disposable outputs you can reproduce, after retaining useful measurements. Preserve shared raw data, supplied assets, working images and other users' files. Temporary storage is machine-local; arrange persistent storage for work you need after a session.

12 Official references

- Apptainer quick start and installation routes.
- Apptainer bind mounts and filesystem visibility.
- Vitis AI 3.5 host installation: Docker is AMD's documented host route; the course supplies Apptainer launchers.
- UG1414, Vitis AI 3.5 containers and framework environments. Use documentation matching the pinned release and installed board image.
- AMD `platformstats`: the power option (`-p`) that the lab energy profilers sample prints the SOM power utilization.

A Board-side quick reference

The board commands of the handouts, gathered in one place. The handouts remain the reference for their arguments and outputs; use these commands only on the board you are given.

A.1 Bring-up, SSH and serial console

Before switching a board on, check that its microSD card holds the image of the lab (PetaLinux for TP02–TP05, Ubuntu/PYNQ for TP06–TP07; otherwise verify the required image before booting), that its Ethernet cable goes to the lab network and its micro-USB cable to the lab machine. The micro-USB port carries the serial console (and JTAG): it works without network and shows the boot messages.

```
# Host
getent hosts maquette-07 # the address behind the name
ssh root@maquette-07 # ubuntu@maquette-07 on the Ubuntu/PYNQ image
ssh-copy-id root@maquette-07 # optional key login (ssh-keygen first if you have no key)
# Host, when the name does not answer
ls /dev/serial/by-id/ # one board: four interfaces
tio /dev/serial/by-id/<name>-if01-port0 # the console, 115200 baud; leave with ctrl-t then q
# Board console: log in with the account of the image, then
ip -4 addr show eth0 # the address follows "inet"
```

`Permission denied` on the serial device: your account is not in the `dialout` group. No `inet` line on `eth0`: no network (cable, switch). After a confirmed re-image, remove that board's old key only: `ssh-keygen -R maquette-07`.

A.2 Checks and measurements on the board

Question	Command, as the labs use it	What it gives
Active accelerator application	`sudo xmutil listapps` (Ubuntu/PYNQ)	The installed applications and the active slot. The TP06–TP07 runtime check stops while the SmartCam application is active: reboot the board first.
Raw DPU throughput	`xdputil benchmark model.xmodel -i -1 1` (TP02; the last number is the thread count)	The XMODEL in a C++ VART loop: no preprocessing, no Python.
DPU time per run	`python3 vai_trace_fpga.py ...`, the lab wrapper of `vaitrace`	Column `HW_RT` of its summary.
Power	`xmutil xlnx_platformstats -p` (as root; `sudo` on Ubuntu), sampled by `profile_energy_fpga.py`	AMD describes this reading as the SOM power utilization: report it as SOM power, not whole-board power.