Dec 13, 2024

APAI2024 - LAB08
PULP-NE16

Authors: Davide Nadalini, Luka Macan, Lorenzo Lamberti, Francesco Conti
Contacts: d.nadalini@unibo.it, luka.macan@unibo.it, lorenzo.lamberti@unibo.it

---

Links: ***GitHub Link (code)*** ***GDOC link (assignment)***

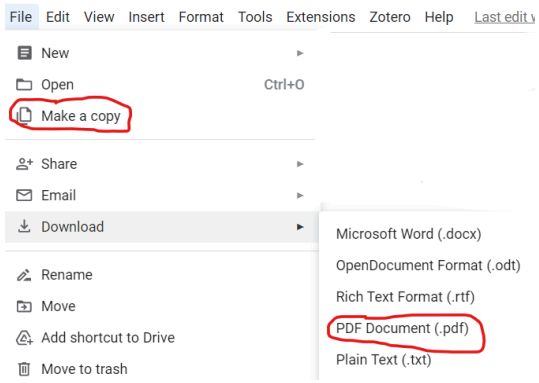
Summary

1. Subject(s):
    - NE16 accelerator
2. Programming Language: C
3. Lab duration: 3h
4. Assignment:
    - Time for delivery: 1 week
    - **Submission deadline:** Dec 27, 2024 at 14:30

How to deliver the assignment:

You will deliver ONLY THIS TEXT FILE, no code

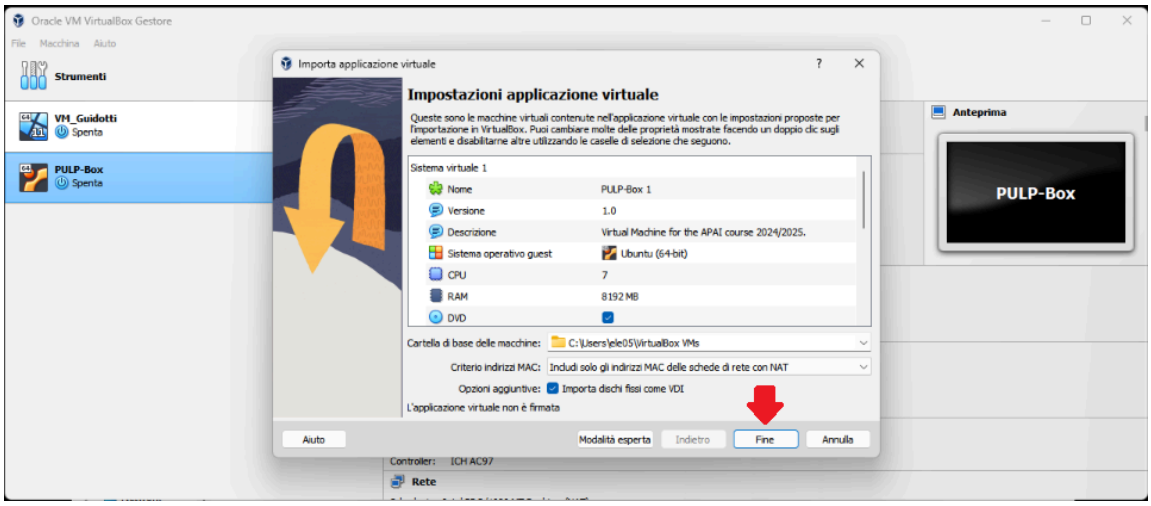
- Copy this Google doc to your drive so that you can modify it. (`File -> make a copy`)
- Fill the tasks on this Google doc.
- Export to PDF format.
- Rename the file to: LAB<number_of_the_lesson>_APAI_<your_name>.pdf
- Use Virtuale platform to load ONLY your .pdf file



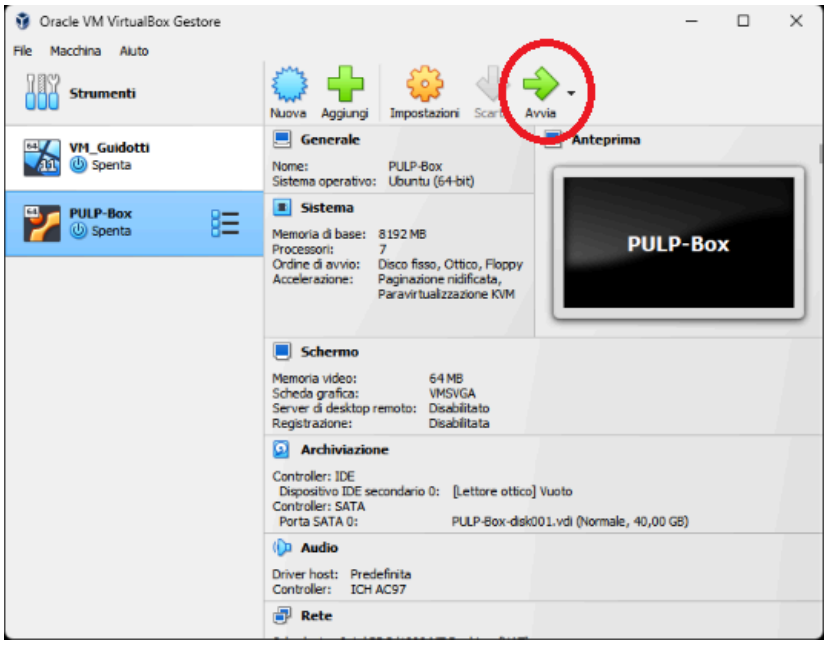
Setup

0. Access to the local VM and setup pulp-sdk

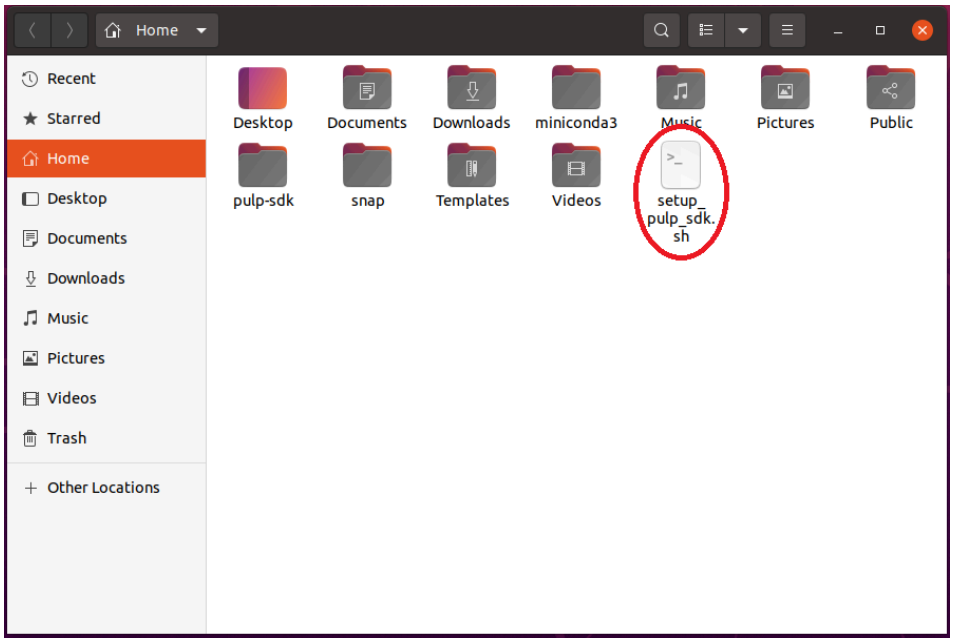
- On the lab's PCs, open the file explorer and go to This PC, C:/VM_Nadalini2
- Double click on PULP-box.ova
- VirtualBox opens, just click on "Fine"



- Wait for the VM to be imported
- Open the VM with "Avvia"



- Password is ‘pulp’
- Open a terminal (right click – open a new terminal)
- **Setup the PULP-SDK:** `source setup_pulp_sdk.sh`



- Clone GitHub repository of today’s lab: `git clone https://github.com/EEESlab/APAI24-LAB08-NE16`

- `cd APAI24-LAB08-NE16`
- `pip install mako numpy six`

How to run the code

1. Generate the parameters:

Command: `python parameters_generate.py`

Arguments:

```
--kernel-shape {1,3}, -ks {1,3}
                      Shape of the kernel. Choices: 1 or 3. Default: 3
--channels-in CIN, -cin CIN
                      Number of input channels. Default: 16
--channels-out COUT, -cout COUT
                      Number of output channels. Default: 32
--output-spatial-dimensions SPATIAL_DIMENSIONS, -osd SPATIAL_DIMENSIONS
                      Output spatial dimension. Default 3
```

`parameters_generate.py` script. We made some changes compared to the last lab. We changed the `--channels` flag into `--channels-in` and `--channels-out`.

Note: the arguments also have their shorthand versions, e.g., `--kernel-shape` is the same as `-ks`.

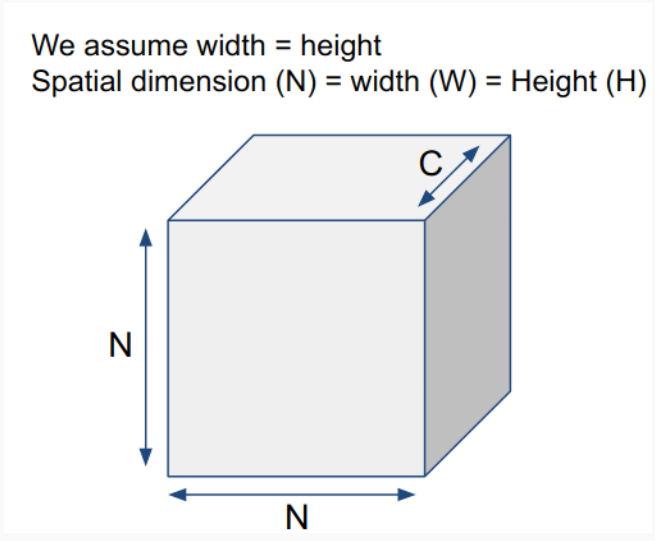
P.S. If you ever need information on the arguments, you can call the script with the `--help` argument like so:

```
python parameters_generate.py --help
```

2. Compile and run the code:

Command: `make clean all run`

“What is the spatial dimension?”



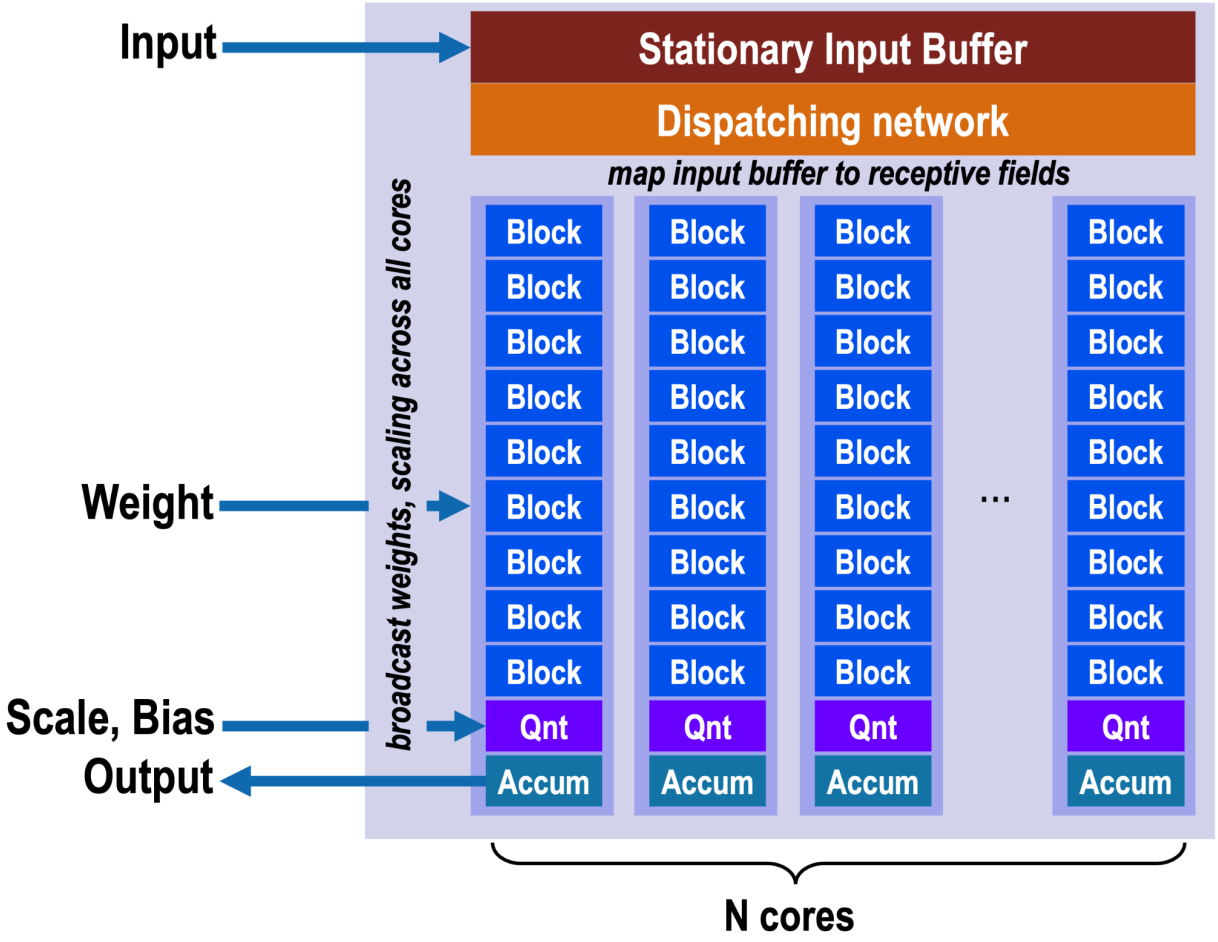
LAB: THEORETICAL PART

Here are the lecture slides on NE16

Recap: architecture overview

NE16 is a flexible weight precision Neural Engine.

Key ideas: many cores, flexible weights bitwidth, programmable operation size implemented through tiling.



A **core** is responsible for computing 1 pixel of output data.

- Each Core contains 9 blocks.
- A block contains 16 1-bit × 8-bit multipliers that are used to multiply 1-bit of the weights with 8-bits of the input.
- The results of all the multipliers from all the blocks in one core are then summed up into one value that gets stored in the accumulator.
- The accumulated results can be optionally normalized and requantized to 8 bits.

In our case: the specific configuration of NE16 contains **9 cores**, each one having **9 blocks,** that work on **ICP=16** input channels and produce **OCP=32** output channels.

Exercise 1: NE16 theoretical maximal performance

In this exercise, we will compute the theoretical maximal performance of NE16 in terms of computational capabilities in *MAC/cycle*. This is done simply by looking at the amount of MAC operations that can be done in a system in one cycle.

With our example of the NE16, we will count the number of *1b x 8b MAC* operations that can be done. As your first exercise, fill out the table below with the right performance for each component in the NE16.

Component	**Perf** (1b x 8b MAC/cycle)
Block	
Core	
NE16	

In the following exercise, where we will measure performance on the simulator, the performance will be expressed in **8-bit MAC/cycle**. For a comparison number, let's calculate the 8-bit MAC/cycle performance by simply dividing the previously calculated performance by *8*.

Questions:

A) NE16 theoretical maximal performance (8-bit MAC/cycle): **[answer]**

Computational modes: 3x3 and 1x1 convolution

NE16 was specifically designed for the convolution operation of kernel size equal to 3x3, but it also supports two other modes: pointwise convolution (kernel size equal to 1x1) and depthwise convolution of kernel size equal to 3x3. Here, we will quickly recap the 3x3 and 1x1 convolution modes, covered in the lectures in more detail.

In the **3x3 mode**, the NE16 maps the inputs in a sliding-window fashion from the input buffer to each block. The weights are then streamed in 1 bit at a time and broadcast across cores.

In the **1x1 mode**, the NE16 maps only one input pixel onto a core and broadcasts it on *weight-bitwidth* blocks in that core. In this mode, the weights are still broadcast across cores, but the whole value is fetched, not 1 bit at a time.

For more details on the modes and how they are mapped onto internal buffers of NE16, consult the lectures from pages 26 to 43.

Exercise 2: Theoretical maximal performance - 3x3 vs. 1x1 mode

The theoretical maximal performance we calculated might change with the mode we choose because of the inner workings of the accelerator. In this exercise, you will fill out the table below with the same theoretical maximal performance in 1b x 8b MAC/cycle as in the previous exercise.

	3x3 mode	**1x1 mode**
Component	**Perf** (1b x 8b MAC/cycle)	**Perf** (1b x 8b MAC/cycle)
Block		
Core		
NE16		

The same as in the previous exercise, let's calculate the theoretical maximal performance in 8-bit MAC/cycle:

Questions:

A) NE16 *3x3 mode* theoretical maximal performance (8-bit MAC/cycle): **[answer]**
B) NE16 *1x1 mode* theoretical maximal performance (8-bit MAC/cycle): **[answer]**
C) How does the maximal performance change between 3x3 and 1x1 mode? **[answer]**

LAB: Hands-on

Each of the following tasks consists of profiling the network execution on the NE16 accelerator to understand how the performance varies when a parameter is changed (kernel size, Cin, Cout, Output spatial dimension). We use the GVSoC simulator to do all the measurements.

Exercise 3: Simulation measurements

Kernel size (-ks)	Input channels (-cin)	Output channels (-cout)	Output spatial dimensions (-osd)	MAC/cycles	Latency [cycles]
1	16 (default)	32 (default)	3 (default)		
3	16 (default)	32 (default)	3 (default)		

Questions:

A. Compare the measured numbers to the theoretical maximal performance from previous exercise. What percentage does the measured performance achieve compared to it? **[answer]**

B. What is more performant in terms of MAC/cycles between 1x1 and 3x3 kernel size? **[answer]**

Exercise 4: Varying layer size

Task 4.1: different input channels

Kernel size (-ks)	Input channels (-cin)	Output channels (-cout)	Output spatial dimensions (-osd)	MAC/cycles	Latency [cycles]
3 (default)	1	32 (default)	3 (default)		
3 (default)	8	32 (default)	3 (default)		
3 (default)	16	32 (default)	3 (default)		
3 (default)	32	32 (default)	3 (default)		
3 (default)	64	32 (default)	3 (default)		

Questions:

A. What is the tile size for NE16? (answer from theory) **[answer]**
B. Do you get better MAC/cycles performance either when the input size is smaller or bigger than NE16's tile size? Why? **[answer]**
C. What is the input size that gets the best MAC/cycle performance? **[answer]**

Task 4.2: what is the best input channel size?

Kernel size (-ks)	Input channels (-cin)	Output channels (-cout)	Output spatial dimensions (-osd)	MAC/cycles	Latency [cycles]
3 (default)	16	32 (default)	3 (default)		
3 (default)	17	32 (default)	3 (default)		
3 (default)	32	32 (default)	3 (default)		

3 (default)	33	32 (default)	3 (default)		

Questions:

A. You should see that performance dramatically increases when the output channels are a multiple of 16. Why? **[answer]**

Task 4.3: what is the best output spatial dimension?

Kernel size (-ks)	Input channels (-cin)	Output channels (-cout)	Output spatial dimensions (-osd)	MAC/cycles	Latency [cycles]
3 (default)	16 (default)	32 (default)	3		
3 (default)	16 (default)	32 (default)	4		
3 (default)	16 (default)	32 (default)	5		
3 (default)	16 (default)	32 (default)	6		
3 (default)	16 (default)	32 (default)	7		
3 (default)	16 (default)	32 (default)	8		
3 (default)	16 (default)	32 (default)	9		

Questions:

A. You should see that performance dramatically increases when the output spatial dimension is a multiple of 3. Why? **[answer]**

Exercise 5 (optional): try NE16 on a real board

Access the VM with GAP-SDK on the PCs of LAB1.

Once GAP-SDK is sourced and ready:

```
cd gap9_ne16_example/
python generate_onnx.py
```

In this example, we deploy a model with:

```
Input: 16x28x28
Conv0 (not to be profiled): 1x16x1x1
Conv1 (to be profiled): 32x16x3x3
Output: 32x26x26
```

Run the code to load and execute the model on GAP9 and fill the first row of the :

```
python run_convolution_on_board.py
```

Task 5.1: performance on the real hardware?

S6__conv1_Conv	Cycles	Operations	MAC/cycle
GAP9			

Questions:

A. Is the performance close to the ones previously obtained on GVSoC? **[answer]**

Exercise 6 (optional): look into NE16 states

Go back to GVSoC (usual VM). There is a way to print out very detailed logs of the accelerator, allowing you to understand the execution phases (=states).

State counting

In the proceeding tasks, you will have to measure the number of states executed for different layers. To ease the counting, we created a script `count_states.sh`.

Quick usage guide:

- If you want to use it on a file, call the script like this:

```
./count_states.sh ne16.log
```

- Or you can pipe the output of the simulation directly to the script:

```
make all run runner_args=”--trace=ne16” | ./count_states.sh
```

You are required to do measurements while only changing the **Cin** parameter. We expect you to execute the commands like this:

```
python parameters_generate.py -cin 1
make all run runner_args="--trace=ne16" > ne16.log
./count_states.sh ne16.log
```

Varying input channel: We want to see how the number of states changes with varying the input channel size. Fill out the table below with measurements.
Notice how different states change below the input channel subtile size (below 16) and above it.

Kernel shape (-ks)	Input channels (-cin)	Output channels (-cout)	Output spatial dimensions (-osd)	MAC/ cycles	Latency (cycles)	Number of states				
						LOAD	MATRIX VEC	NQ_MULT	NQ_BIAS	STREAMOUT
3 (default)	1	32 (default)	3 (default)							
3 (default)	8	32 (default)	3 (default)							
3 (default)	16	32 (default)	3 (default)							
3 (default)	32	32 (default)	3 (default)							
3 (default)	48	32 (default)	3 (default)							