ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

LAB07: Tiling on PULP part2

Davide Nadalini – d.nadalini@unibo.it
Lorenzo Lamberti – lorenzo.lamberti@unibo.it
Luka Macan – luka.macan@unibo.it
Francesco Conti – f.conti@unibo.it

Objective of the Class

Intro: Tiling

Tasks:

- Double Buffering
- Overlapping tiles with 3x3 convolutions

Deadline:

Dec 13th 2024

Programming Language: C

Lab duration: 3h

Assignment:

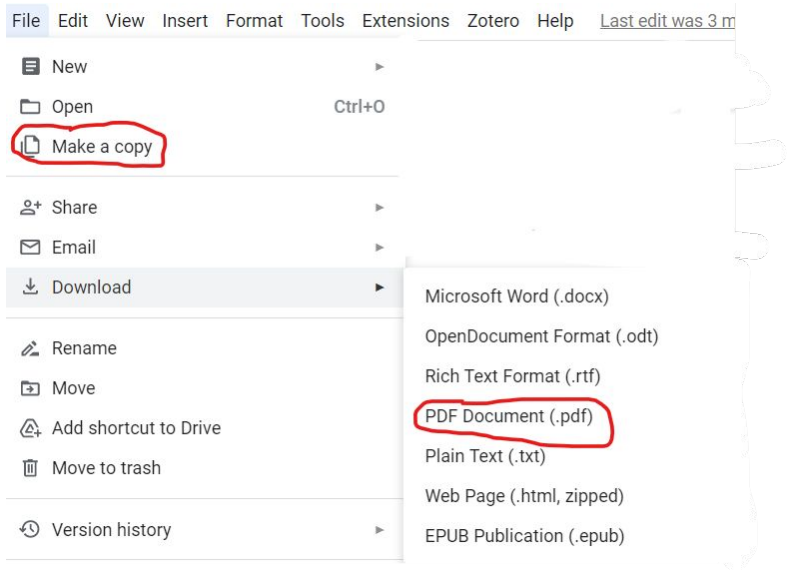
- Time for delivery: 2 weeks

The class is meant to be interactive: coding together, on your own, and do not be afraid to ask questions!

How to deliver the Assignment

You will deliver ONLY the GDOC assignment, no code

- Copy the 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



Opening the VM and VSCode

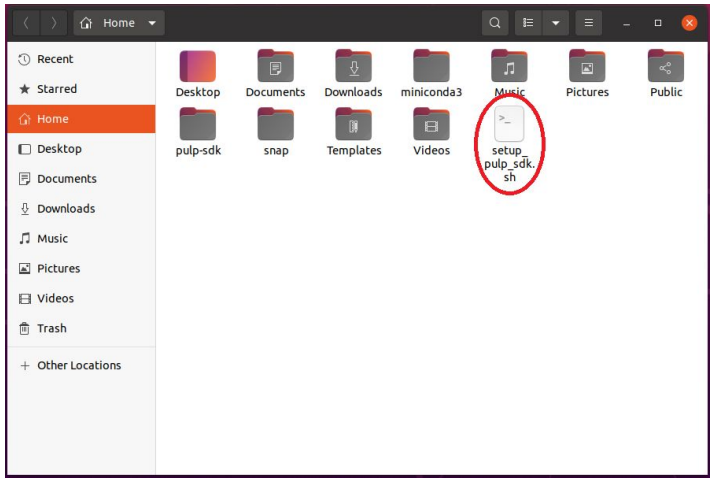
1. Open a terminal (right click – open a new terminal)

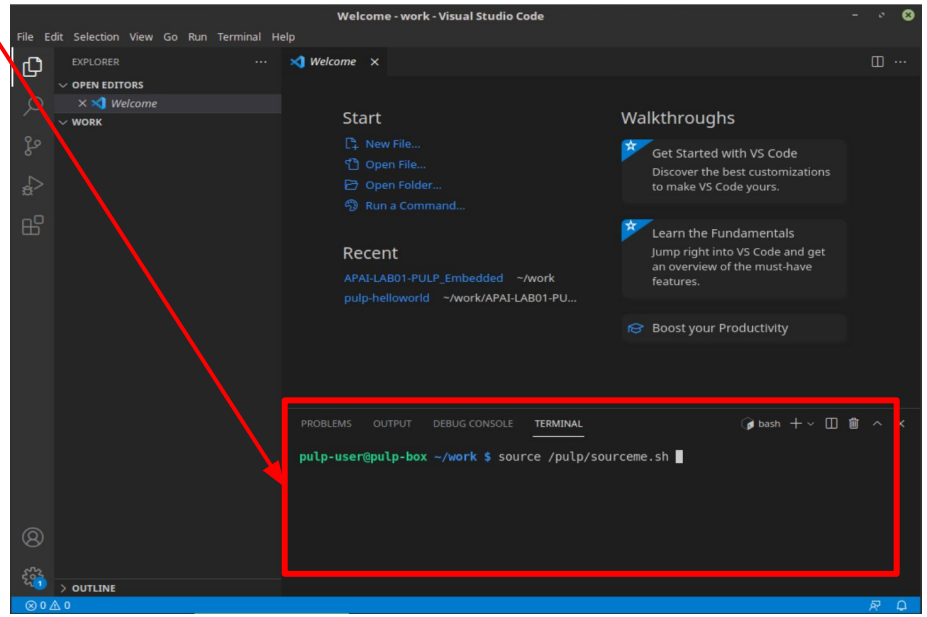
2. Open a text editor (For example “VSCode”):
   Now you can use the integrated terminal (open with CTRL+J) to run your applications!

```
$ code .
```

IMPORTANT: every time you open a new terminal to work on PULP, launch

```
$ source setup_pulp_sdk.sh
```





Getting Started

IMPORTANT: activate the pulp-sdk module file every time a new shell is open.

```
$ source setup_pulp_sdk.sh
```

For the lab of today, we need to:

```
$ pip install mako numpy six
```

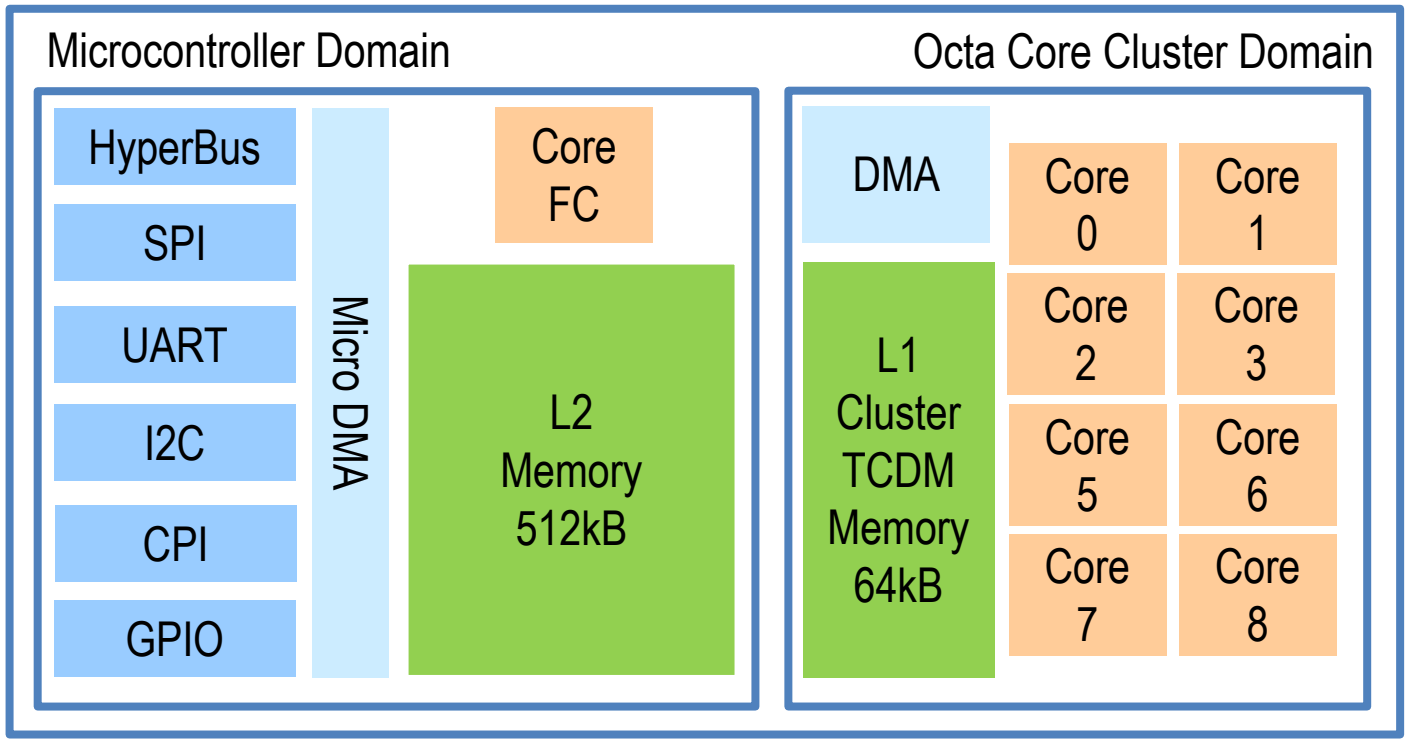
HOW TO RUN THE CODE:

```
$ git clone https://github.com/EEESlab/APAI24-LAB07-PULP-Tiling-part2
$ cd APAI24-LAB07-PULP-Tiling-part2
$ python3 parameters_generate.py --kernel-shape=<add_here>
--channels=<add_here> --output-spatial_dimensions=<add_here>
$ make clean all run
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

TASK4: double buffering

PULP Platform: today we focus on the 8-cores cluster

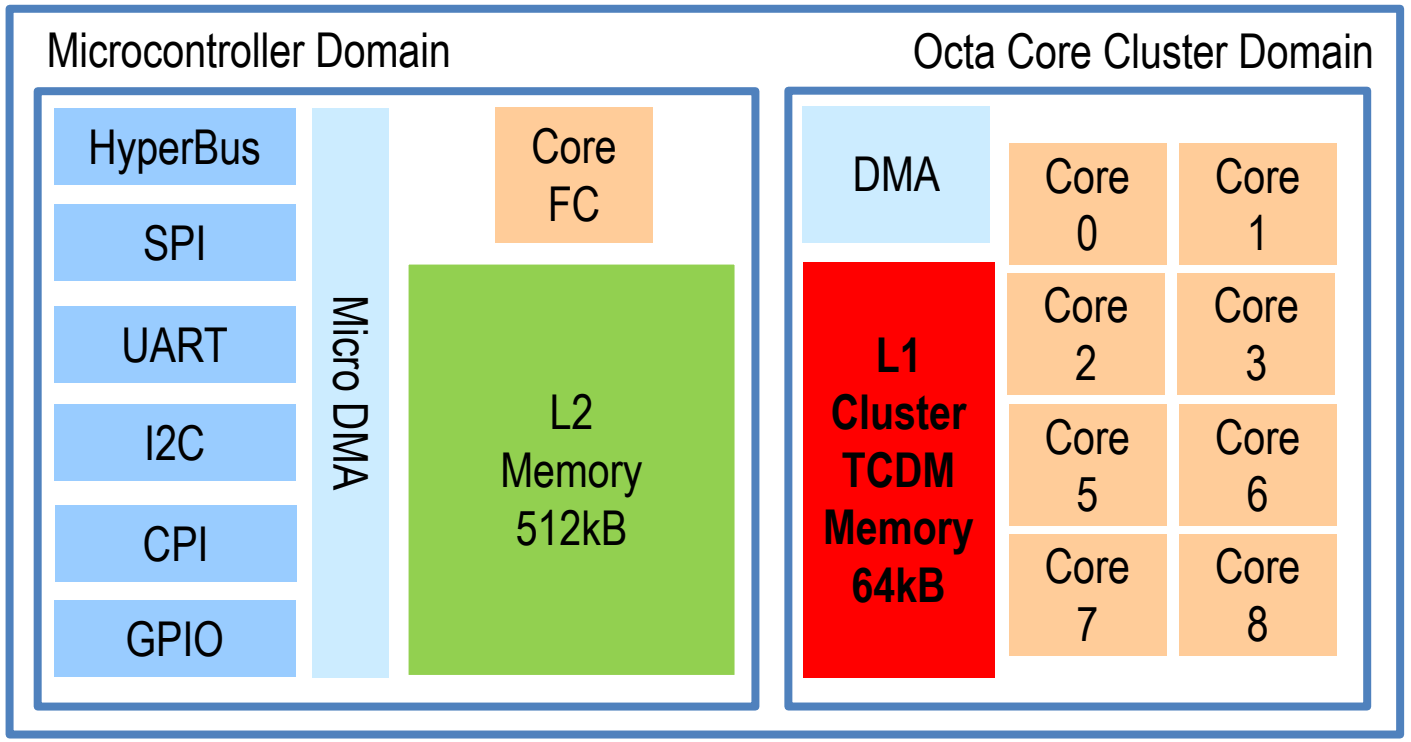


- **Cores**: 1 + 8
- **On-chip Memories**
  - A level 2 Memory, shared among all cores
  - A level 1 Memory, shared by the 8-cores cluster
- **cluster-DMA:** A multi-channel 1D/2D DMA, controlling the transactions between the L2 and L1 memories
- **micro-DMA**: A smart, lightweight and completely autonomous DMA () capable of handling complex I/O scheme
  - **Bus+Peripherals:** HyperBus, I2S, CPI, timers, SPI, GPIOs, etc…

NB: this is the architecture you find on the nano-drone!

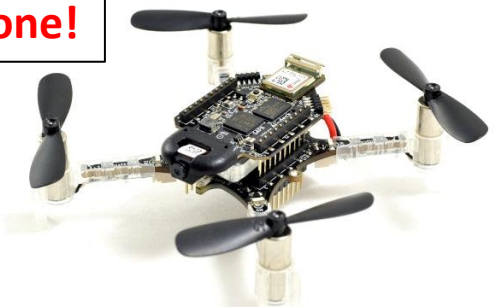
GitHub HW Project: https://github.com/pulp-platform/pulp
HW Documentation:
https://raw.githubusercontent.com/pulp-platform/pulp/master/doc/datasheet.pdf

PULP Platform: today we focus on the 8-cores cluster



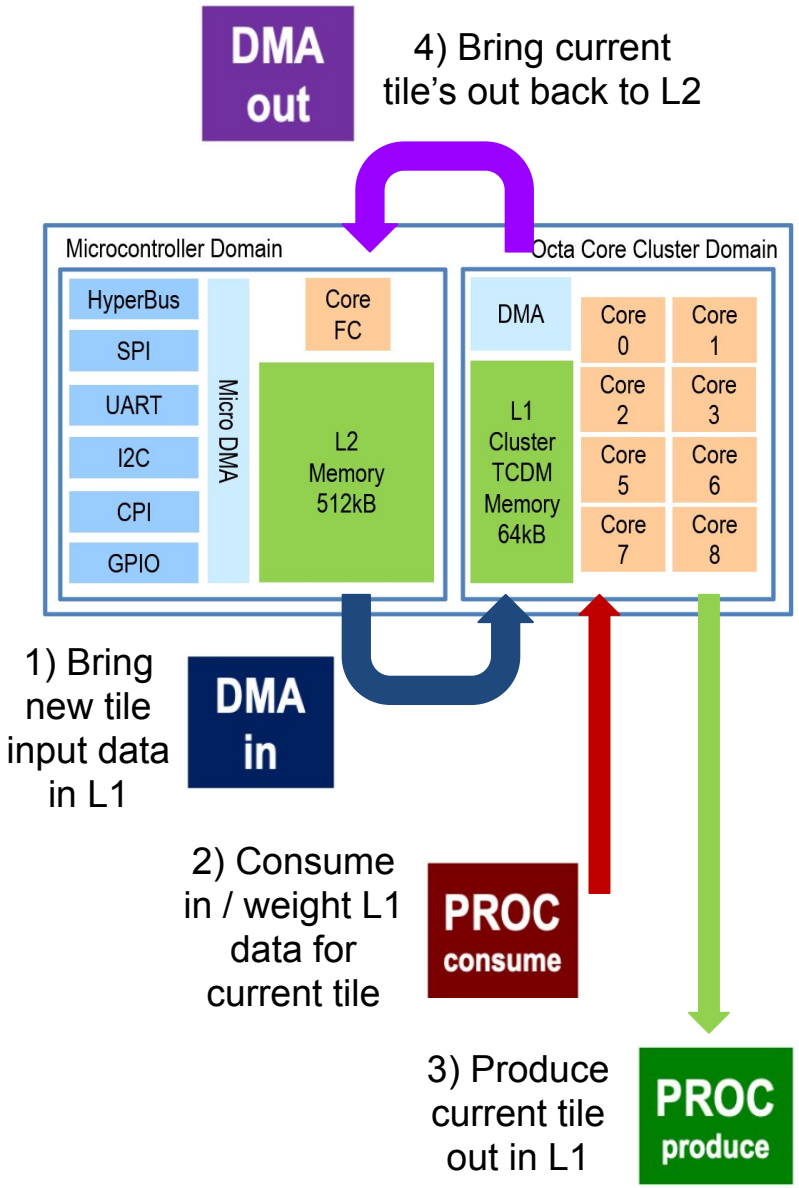
- **Cores**: 1 + 8
- **On-chip Memories**
  - A level 2 Memory, shared among all cores
  - A level 1 Memory, shared by the 8-cores cluster
- **cluster-DMA:** A multi-channel 1D/2D DMA, controlling the transactions between the L2 and L1 memories
- **micro-DMA**: A smart, lightweight and completely autonomous DMA () capable of handling complex I/O scheme
  - **Bus+Peripherals:** HyperBus, I2S, CPI, timers, SPI, GPIOs, etc…

NB: this is the architecture you find on the nano-drone!

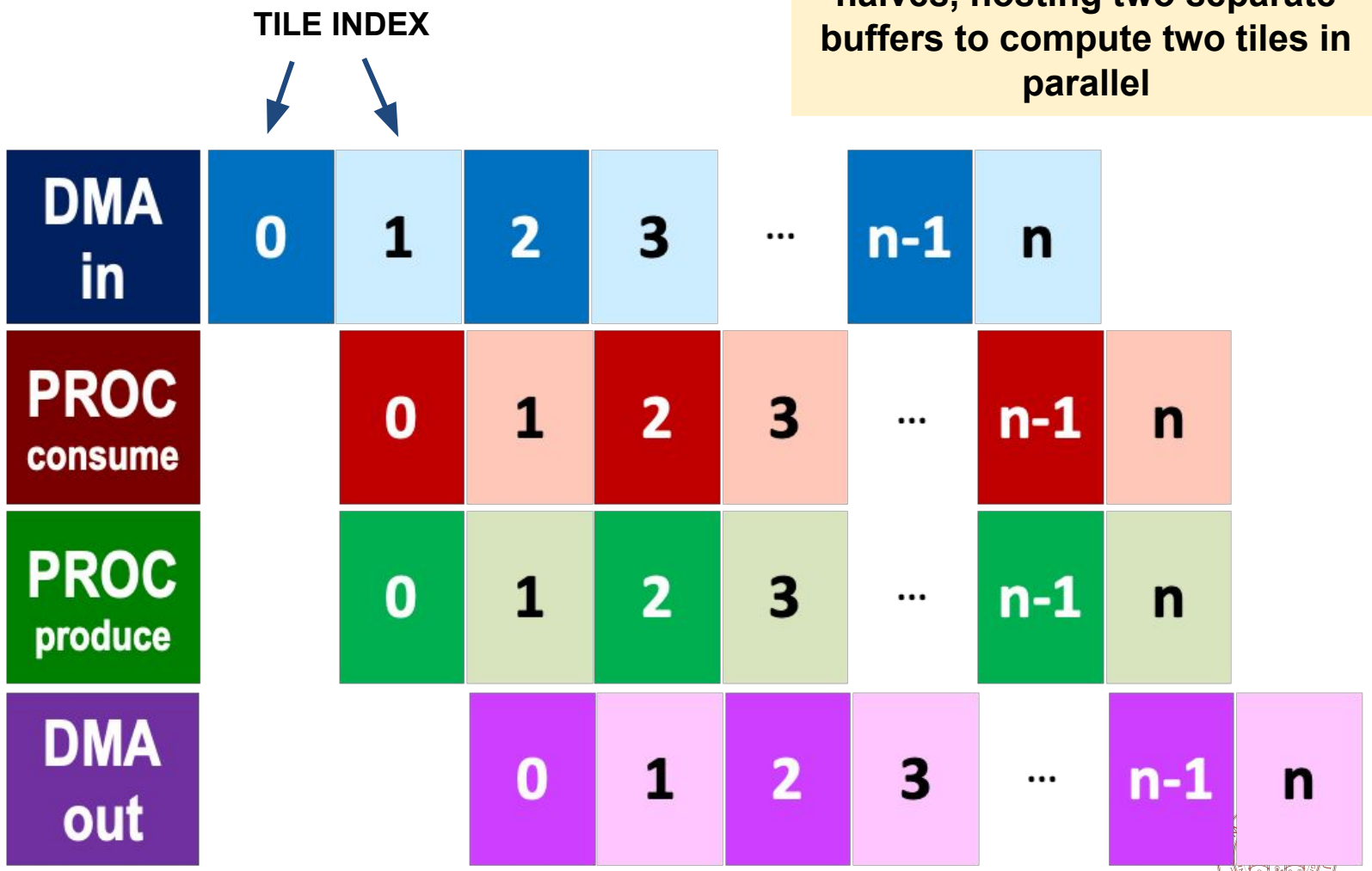


GitHub HW Project: https://github.com/pulp-platform/pulp
HW Documentation:
https://raw.githubusercontent.com/pulp-platform/pulp/master/doc/datasheet.pdf

LAB06: Tiling from L2 to L1 with double buffering



DOUBLE BUFFERING: L1 memory is divided in two halves, hosting two separate buffers to compute two tiles in parallel



EX4: find maximum dimensions of layers fitting L1 without tiling

Prerequisites:

```
source setup_pulp_sdk.sh
```

Run the code:

```
1.  python3 parameters_generate.py --kernel-shape=<add_here> --channels=<add_here> --output-spatial_dimensions=<add_here>
2.  make clean all run
```

Follow the assignment document.

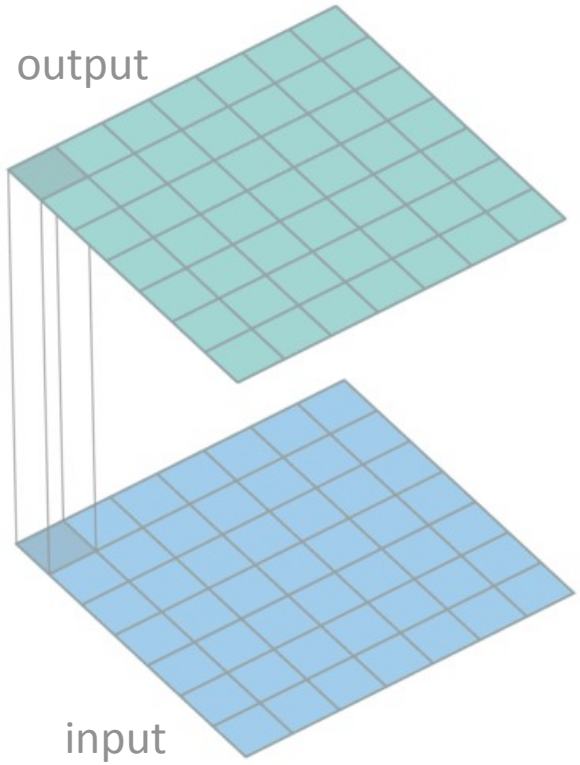
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

TASK5: conv3x3 and overlapping tiles

Case study: 3x3 conv2D

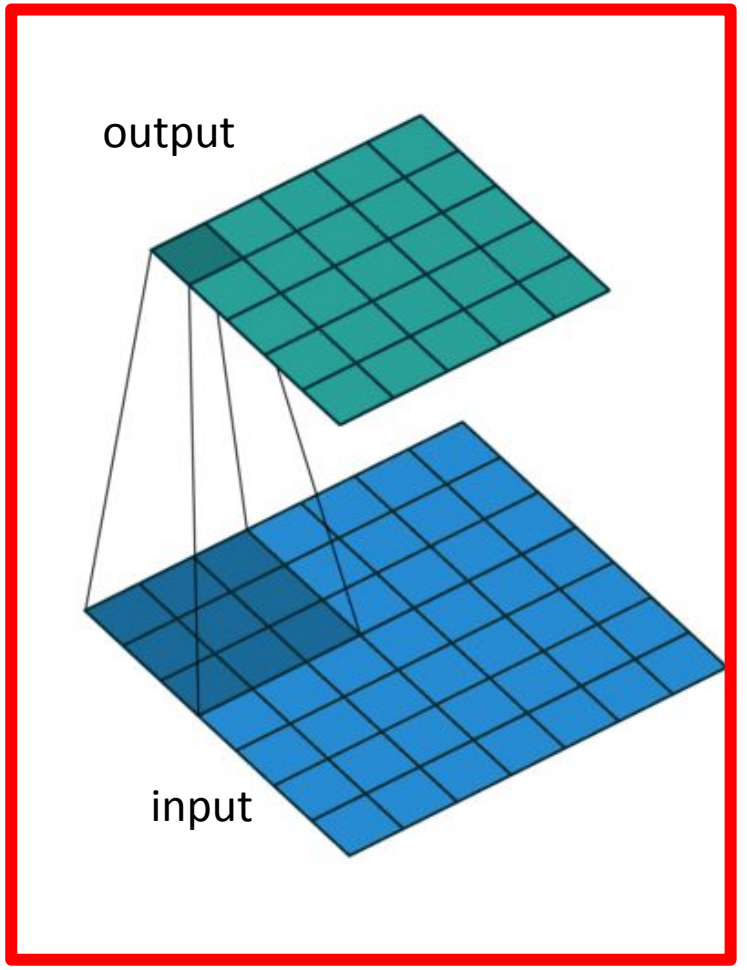
Task5.1: cov3x3

- conv1x1 the spatial size between input and output does not change!
- With conv3x3 it changes. Find out how!



1x1 Convolution

lab06



3x3 Convolution

Used in lab07!

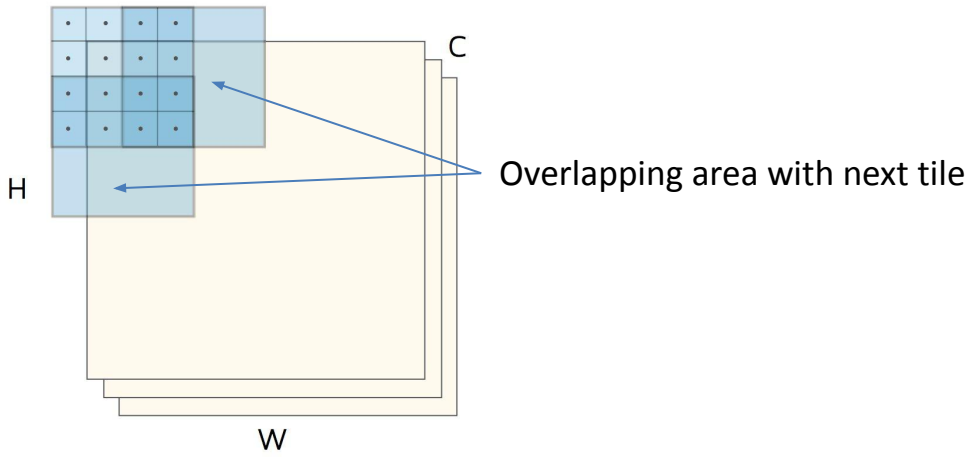
Case study: 3x3 conv2D

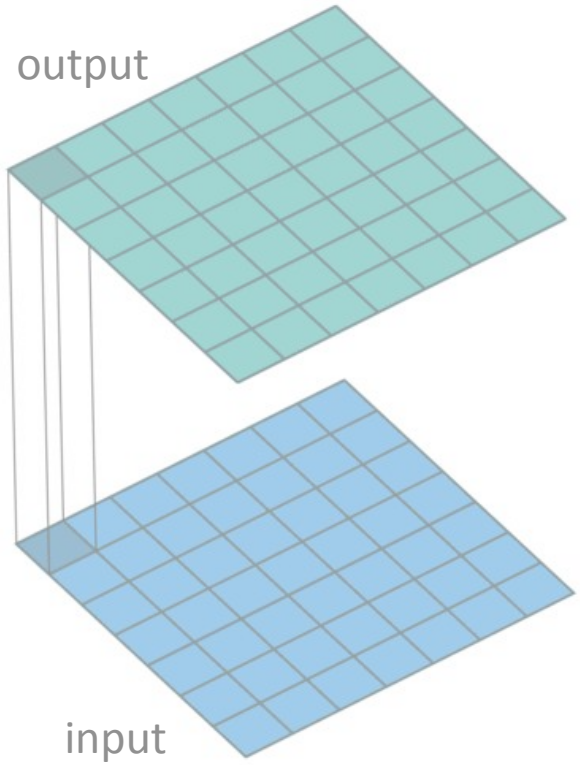
Task5.1: cov3x3

- conv1x1 the spatial size between input and output does not change!
- With conv3x3 it changes. Find out how!

Task5.2: overlapping tiles

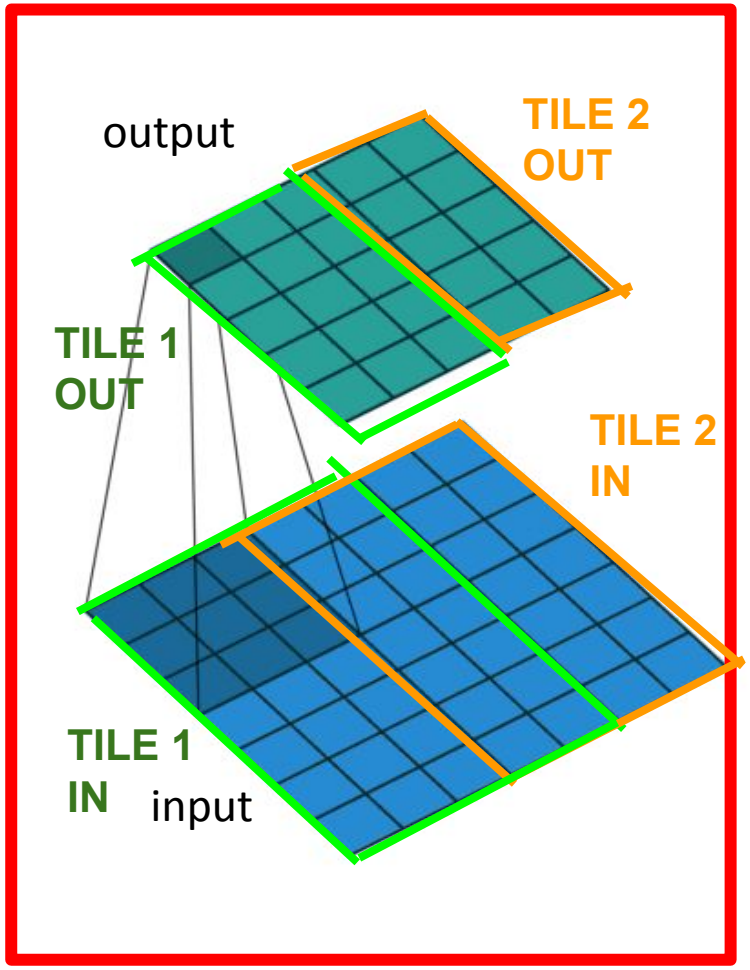
With 3x3 convolutions adjacent tiles overlap a bit, so we load the same piece of input twice.





1x1 Convolution

lab06



3x3 Convolution

Used in lab07!

EX3: Tiling layer

Prerequisites:

```
source setup_pulp_sdk.sh
```

Run the code:

```
1.  python3 parameters_generate.py --kernel-shape=<add_here> --channels=<add_here> --output-spatial_dimensions=<add_here>
2.  make clean all run
```

Follow the assignment document.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEI – Università di Bologna

www.unibo.it