ASIC^2 ARCHITECTURES
SYSTEMS
INTELLIGENT COMPUTING
INTEGRATED CIRCUITS

Adi Hayon

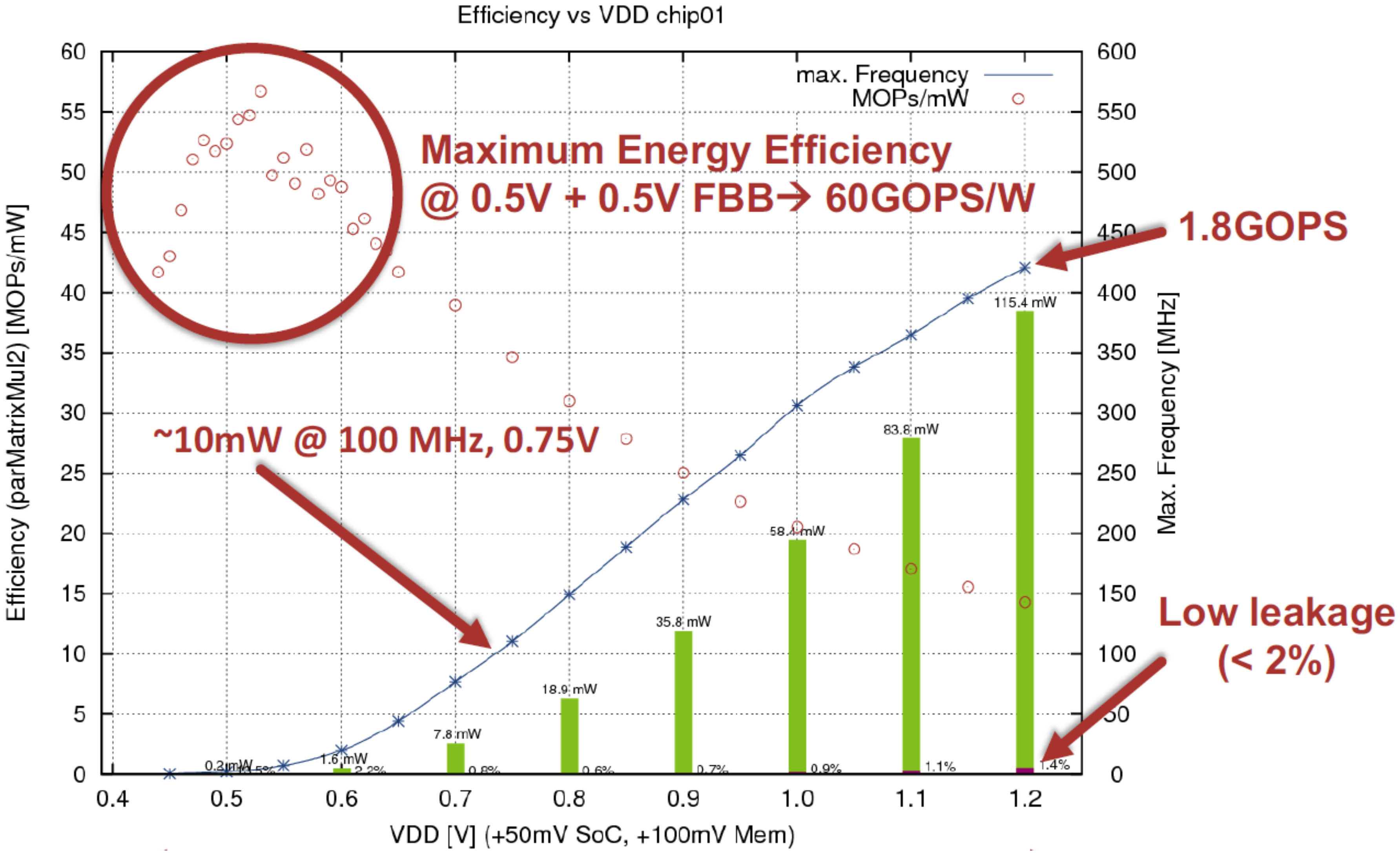
12/03/2018

The PULP Processor
Parallel-Ultra-Low-Power

Parallel Ultra Low Power (PULP)

- The project started in **2013** by Luca Benini
- A collaboration between University of Bologna and ETH Zurich
- The key goal is: **How to get the highest performance for the ENERGY consumed in a computing system.**

Energy efficiency is the key driver in the PULP project



Maximum Energy Efficiency @ 0.5V + 0.5V FBB→ 60GOPS/W

1.8GOPS

~10mW @ 100 MHz, 0.75V

Low leakage (< 2%)

PULP Goals

- Concentrating on **programmable** systems
  - Cannot have custom hardware, need to be scalable
- Making the system accessible to application developers
- **Scalable over a wide operating range**
  - Work just as well when processing 0.001 GOPS as 1000 GOPS
- **Don’t waste** idle energy
  - Eliminate sources where cores and systems are idly wasting energy
- Take advantage of **heterogeneous acceleration**
  - Allow an architecture where accelerators can be added efficiently

Why is Open Hardware Different than Open Software?

- From gnu.org www site:

  http://www.gnu.org/philosophy/free-hardware-designs.html
- **Software** is the operational part of a device that can be copied and changed in a computer
- **Hardware** is the operational part that can't be.
- You can not produce HW directly, you need
  - manufacturing plants
  - know-how
  - and volume

  to be able to manufacture HW **with reasonable cost**.

Open Hardware is a necessity, not an ideological crusade

- **The way we design ICs has changed, big part is now infrastructure**
  - Processors, peripherals, memory subsystems are now considered infrastructure
  - Very few (if any) groups design complete IC from scratch
  - High quality building blocks (IP) needed

Open Hardware is a necessity, not an ideological crusade

- **The way we design ICs has changed, big part is now infrastructure**
  - Processors, peripherals, memory subsystems are now considered infrastructure
  - Very few (if any) groups design complete IC from scratch
  - High quality building blocks (IP) needed
- **We need an easy and fast way to collaborate with people**
  - Currently complicated agreements have to be made between all partners
  - In many cases, too difficult for academia and SMEs

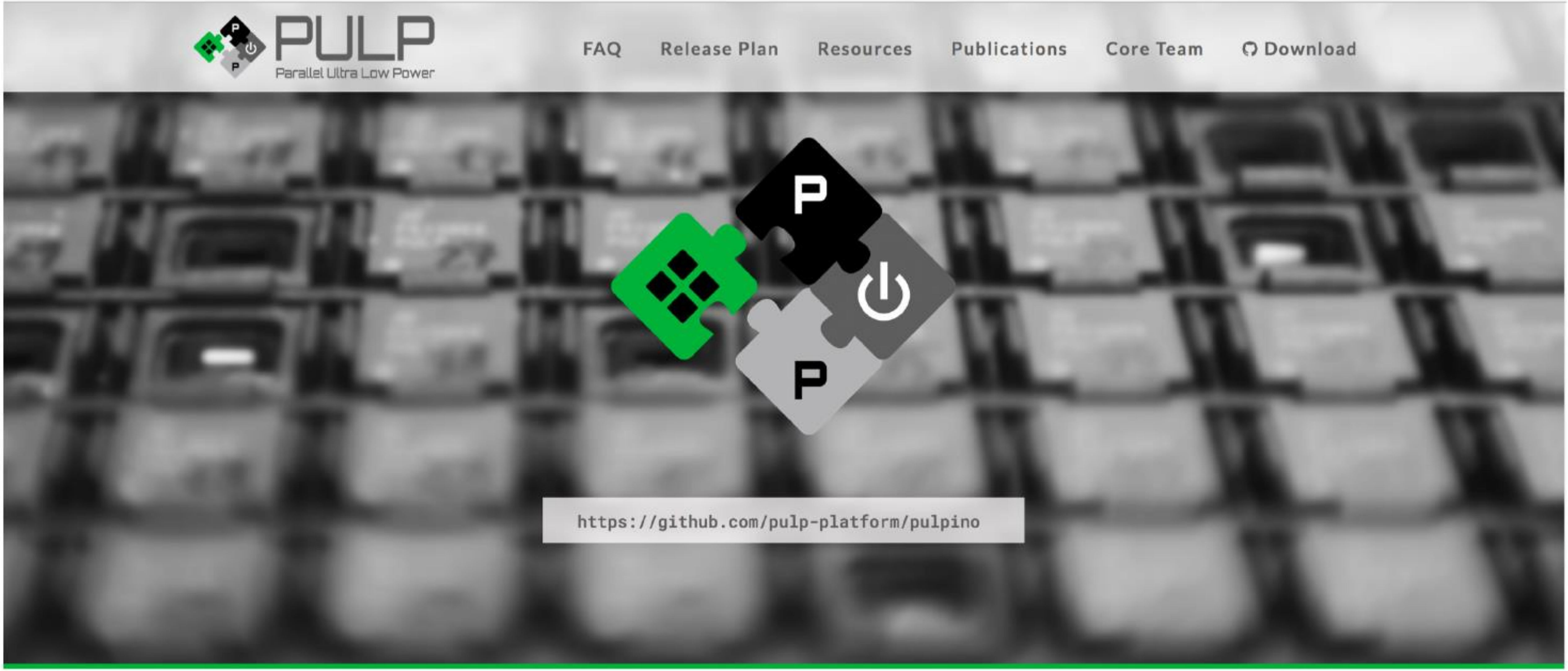
Open Hardware is a necessity, not an ideological crusade

- **The way we design ICs has changed, big part is now infrastructure**
  - Processors, peripherals, memory subsystems are now considered infrastructure
  - Very few (if any) groups design complete IC from scratch
  - High quality building blocks (IP) needed
- **We need an easy and fast way to collaborate with people**
  - Currently complicated agreements have to be made between all partners
  - In many cases, too difficult for academia and SMEs
- **Hardware is a critical for security, we need to ensure it is secure**
  - Being able to see what is really inside will improve security
  - Having a way to design open HW, will not prevent people from keeping secrets.

Current HW only supports security through obscurity

- **Systems are built on hardware blocks where you do not know what exactly is inside**
  - Open standards have proven themselves in SW. Why should HW be any different?
  - If you really want, you can still ‘obscure’ HW, but open HW gives you a choice!
  - Many bugs, features with unintentional consequences are hiding inside HW
- **Open HW will allow a larger community to verify building blocks**
  - Better verification, more reliable hardware

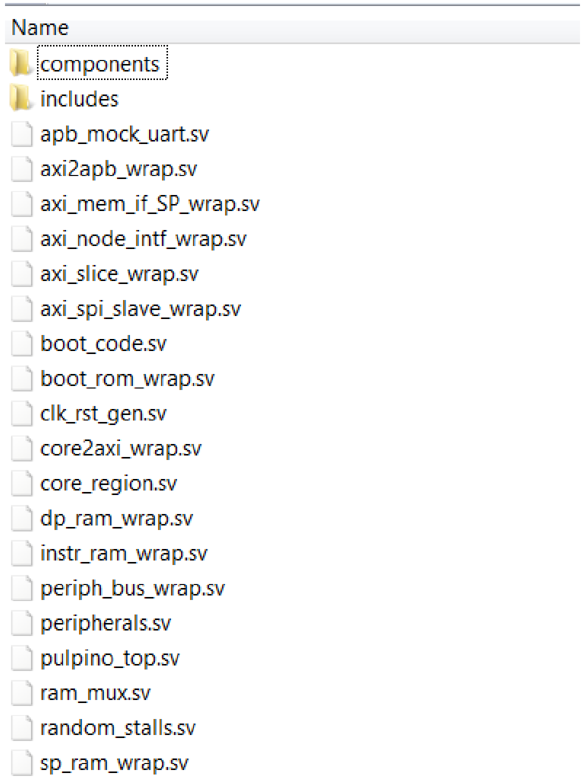
Open Hardware



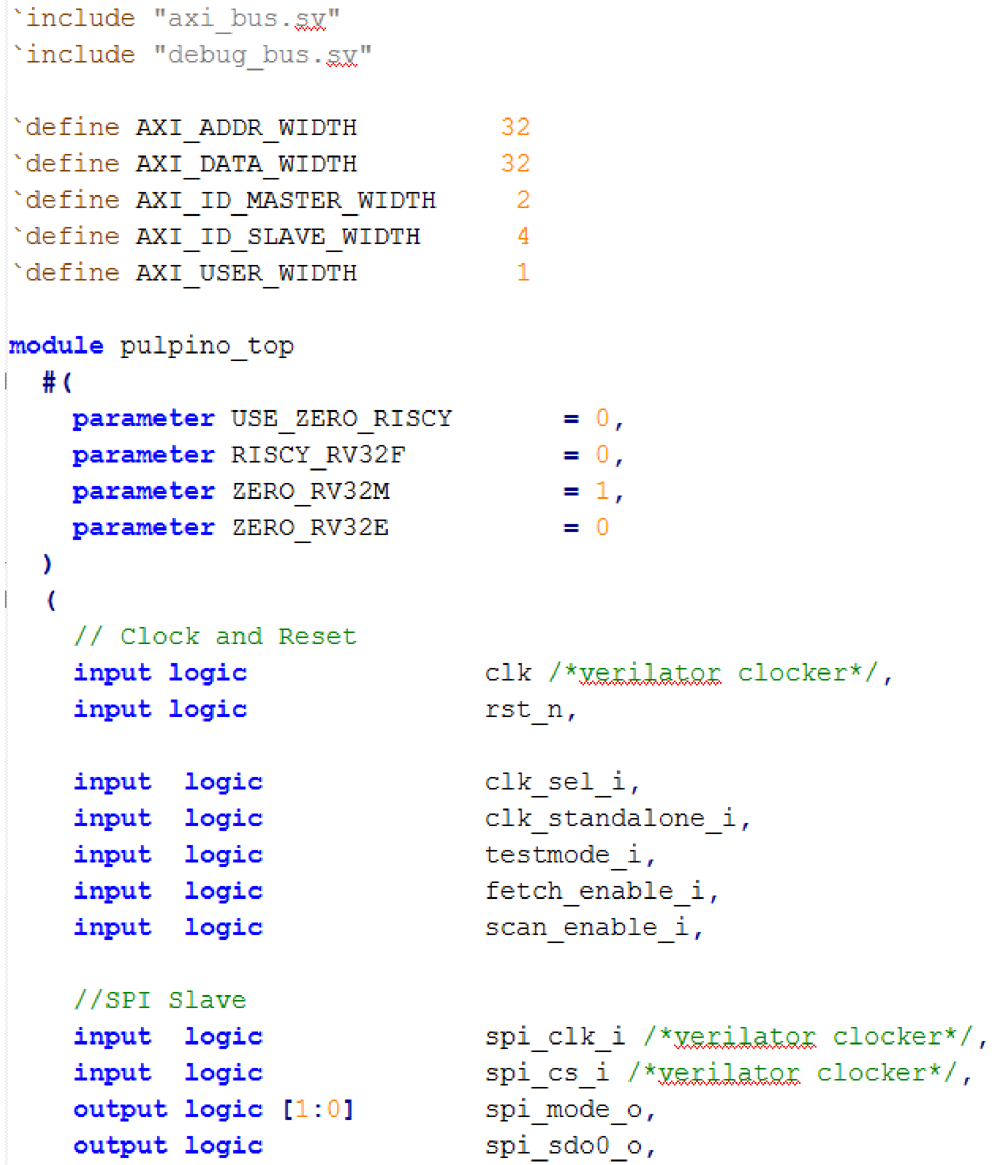
https://github.com/pulp-platform/pulpino

PULP Bundle

- ZIP file from github includes:
  - RTL Code



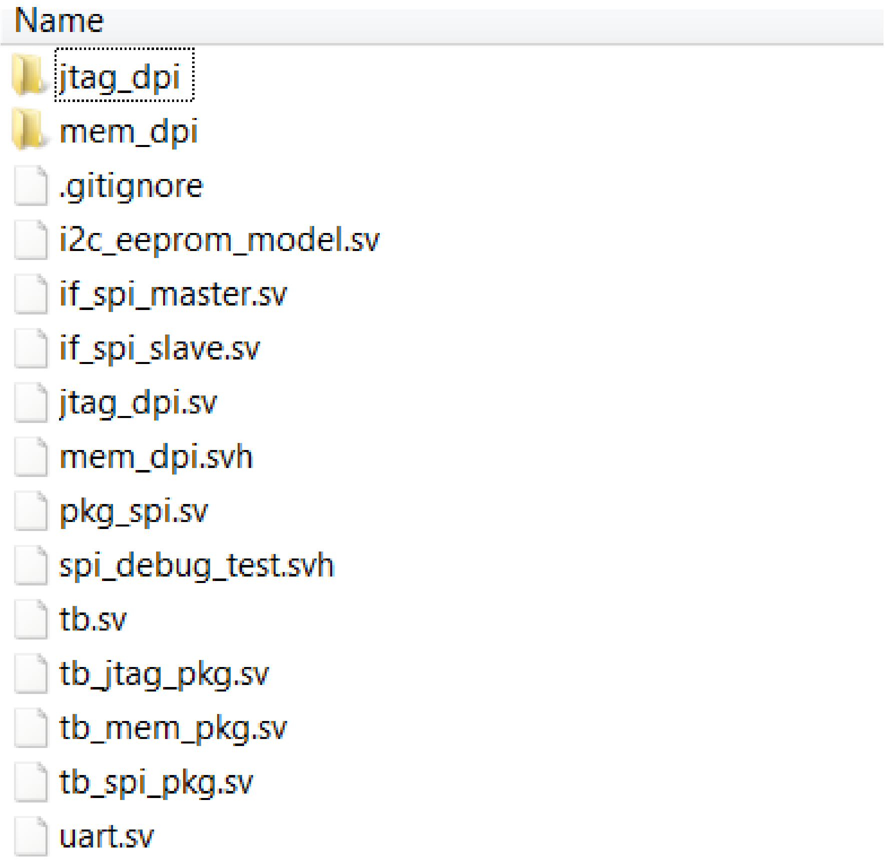
Name
- components
- includes
- apb_mock_uart.sv
- axi2apb_wrap.sv
- axi_mem_if_SP_wrap.sv
- axi_node_intf_wrap.sv
- axi_slice_wrap.sv
- axi_spi_slave_wrap.sv
- boot_code.sv
- boot_rom_wrap.sv
- clk_rst_gen.sv
- core2axi_wrap.sv
- core_region.sv
- dp_ram_wrap.sv
- instr_ram_wrap.sv
- periph_bus_wrap.sv
- peripherals.sv
- pulpino_top.sv
- ram_mux.sv
- random_stalls.sv
- sp_ram_wrap.sv



```
`include "axi_bus.sv"
`include "debug_bus.sv"

`define AXI_ADDR_WIDTH         32
`define AXI_DATA_WIDTH         32
`define AXI_ID_MASTER_WIDTH     2
`define AXI_ID_SLAVE_WIDTH      4
`define AXI_USER_WIDTH          1

module pulpino_top
  #(
    parameter USE_ZERO_RISCY       = 0,
    parameter RISCY_RV32F          = 0,
    parameter ZERO_RV32M           = 1,
    parameter ZERO_RV32E           = 0
  )
  (
    // Clock and Reset
    input logic                clk /*verilator clocker*/,
    input logic                rst_n,

    input  logic               clk_sel_i,
    input  logic               clk_standalone_i,
    input  logic               testmode_i,
    input  logic               fetch_enable_i,
    input  logic               scan_enable_i,

    //SPI Slave
    input  logic               spi_clk_i /*verilator clocker*/,
    input  logic               spi_cs_i /*verilator clocker*/,
    output logic [1:0]         spi_mode_o,
    output logic               spi_sdo0_o,
```

PULP Bundle

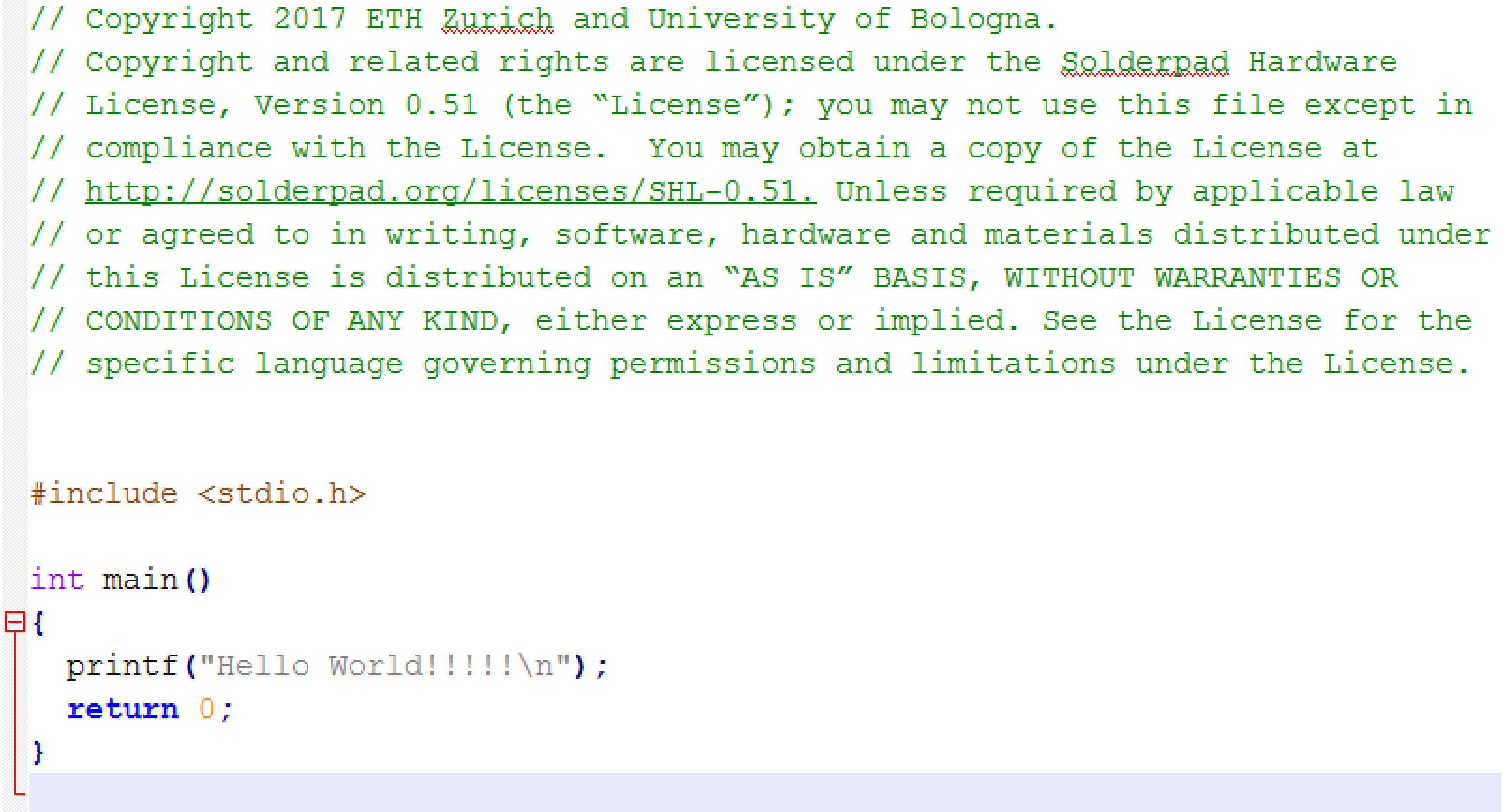
- ZIP file from github includes:
  - RTL Code
  - Testbench



```
Name
jtag_dpi
mem_dpi
.gitignore
i2c_eeprom_model.sv
if_spi_master.sv
if_spi_slave.sv
jtag_dpi.sv
mem_dpi.svh
pkg_spi.sv
spi_debug_test.svh
tb.sv
tb_jtag_pkg.sv
tb_mem_pkg.sv
tb_spi_pkg.sv
uart.sv
```

PULP Bundle

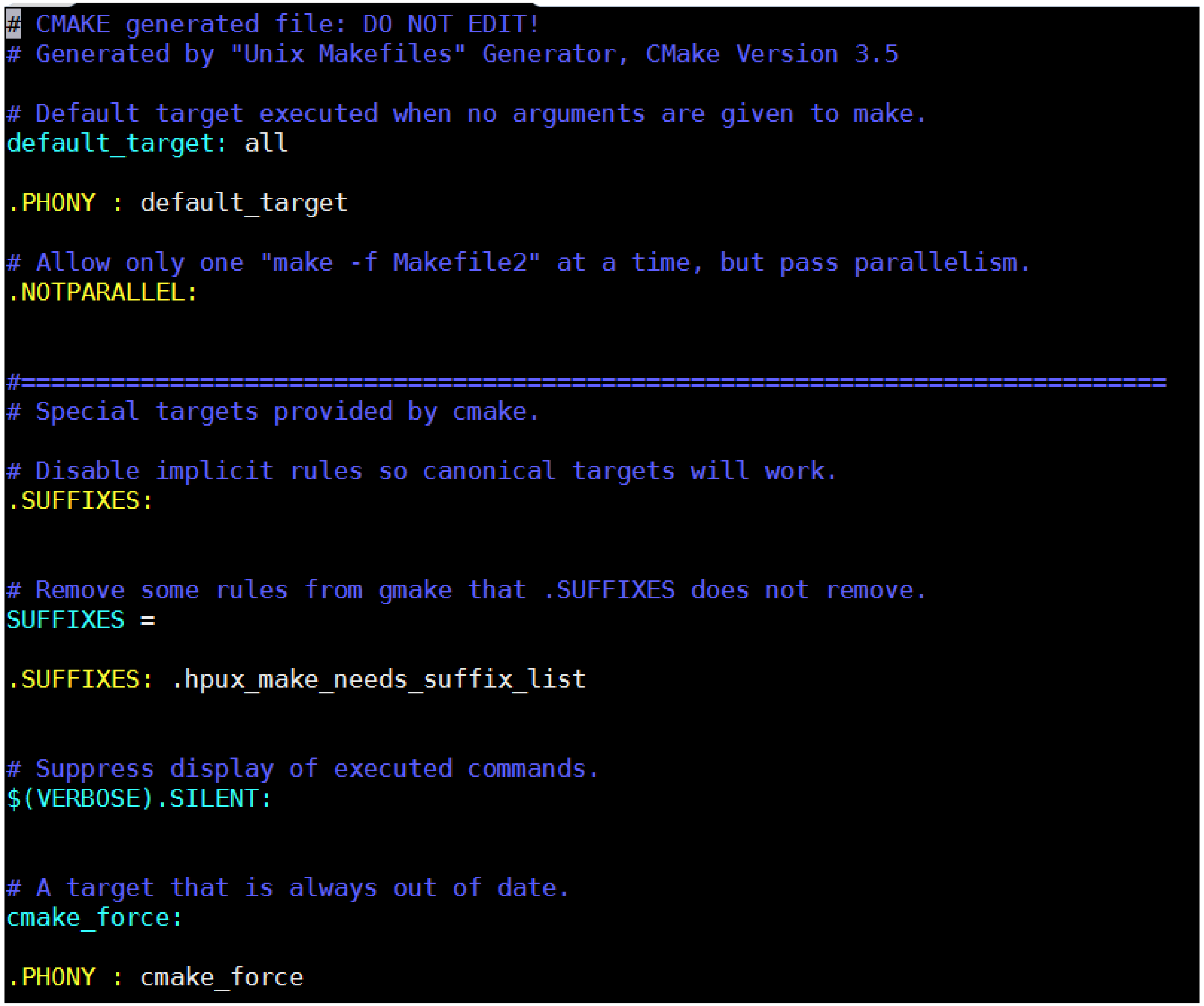
- ZIP file from github includes:
  - RTL Code
  - Testbench
  - Example C code



```
// Copyright 2017 ETH Zurich and University of Bologna.
// Copyright and related rights are licensed under the Solderpad Hardware
// License, Version 0.51 (the “License”); you may not use this file except in
// compliance with the License.  You may obtain a copy of the License at
// http://solderpad.org/licenses/SHL-0.51. Unless required by applicable law
// or agreed to in writing, software, hardware and materials distributed under
// this License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR
// CONDITIONS OF ANY KIND, either express or implied. See the License for the
// specific language governing permissions and limitations under the License.


#include <stdio.h>

int main()
{
  printf("Hello World!!!!!\n");
  return 0;
}
```

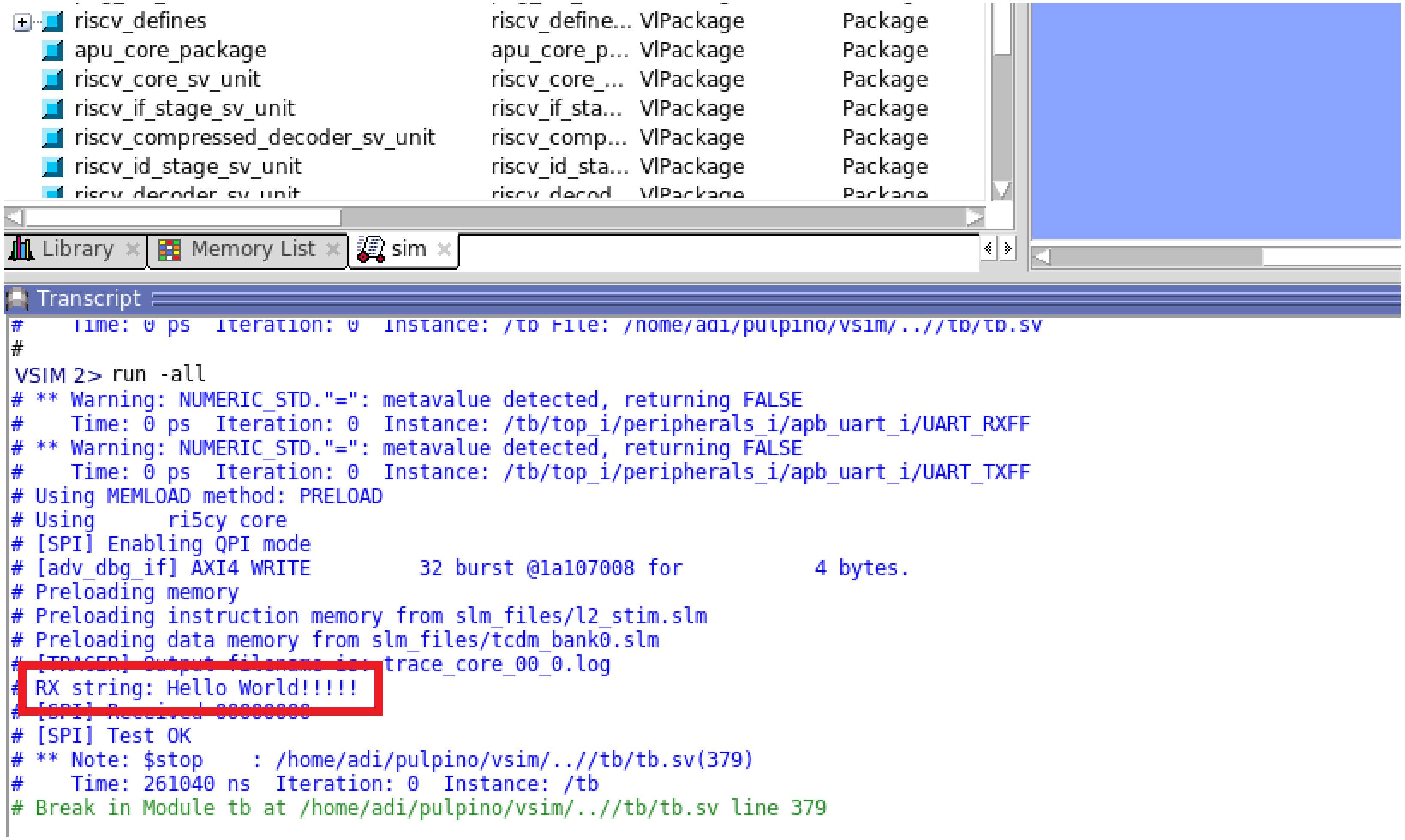
PULP Bundle

- ZIP file from github includes:
  - RTL Code
  - Testbench
  - Example C code
  - Makefile



```
# CMAKE generated file: DO NOT EDIT!
# Generated by "Unix Makefiles" Generator, CMake Version 3.5

# Default target executed when no arguments are given to make.
default_target: all

.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:


#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:


# Remove some rules from gmake that .SUFFIXES does not remove.
SUFFIXES =

.SUFFIXES: .hpux_make_needs_suffix_list


# Suppress display of executed commands.
$(VERBOSE).SILENT:


# A target that is always out of date.
cmake_force:

.PHONY : cmake_force
```

Modelsim Simulation



riscv_defines	riscv_define...	VlPackage	Package
apu_core_package	apu_core_p...	VlPackage	Package
riscv_core_sv_unit	riscv_core_...	VlPackage	Package
riscv_if_stage_sv_unit	riscv_if_sta...	VlPackage	Package
riscv_compressed_decoder_sv_unit	riscv_comp...	VlPackage	Package
riscv_id_stage_sv_unit	riscv_id_sta...	VlPackage	Package
riscv_decoder_sv_unit	riscv_decod...	VlPackage	Package

```
#    Time: 0 ps  Iteration: 0  Instance: /tb File: /home/adi/pulpino/vsim/..//tb/tb.sv
#
VSIM 2> run -all
# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE
#    Time: 0 ps  Iteration: 0  Instance: /tb/top_i/peripherals_i/apb_uart_i/UART_RXFF
# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE
#    Time: 0 ps  Iteration: 0  Instance: /tb/top_i/peripherals_i/apb_uart_i/UART_TXFF
# Using MEMLOAD method: PRELOAD
# Using      ri5cy core
# [SPI] Enabling QPI mode
# [adv_dbg_if] AXI4 WRITE         32 burst @1a107008 for           4 bytes.
# Preloading memory
# Preloading instruction memory from slm_files/l2_stim.slm
# Preloading data memory from slm_files/tcdm_bank0.slm
# [TRACER] Output filename is: trace_core_00_0.log
# RX string: Hello World!!!!!
# [SPI] Received 00000000
# [SPI] Test OK
# ** Note: $stop    : /home/adi/pulpino/vsim/..//tb/tb.sv(379)
#    Time: 261040 ns  Iteration: 0  Instance: /tb
# Break in Module tb at /home/adi/pulpino/vsim/..//tb/tb.sv line 379
```

The PULP family explained

RISC-V Cores

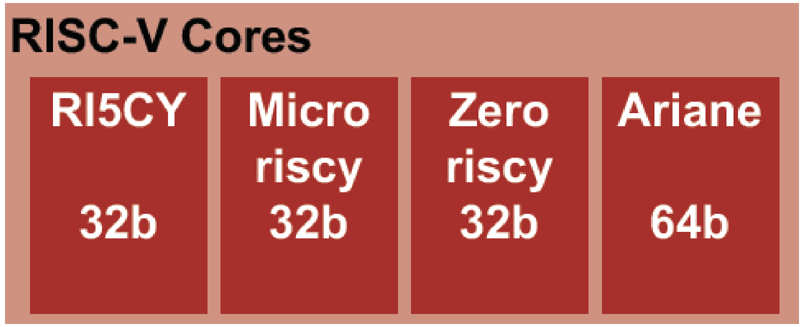
Peripherals

Interconnect

Platforms

Accelerators

We have developed several optimized RISC-V cores



RISC-V Cores

RI5CY 32b

Micro riscy 32b

Zero riscy 32b

Ariane 64b

We have also been working on hardware accelerators

RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

We have our own peripherals and interconnect solutions

RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

Peripherals

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

By combining these components we get PULP platforms

RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

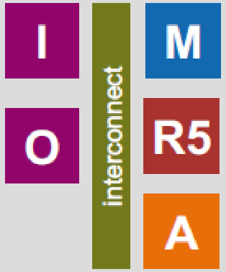
Peripherals

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Platforms



Single Core

- PULPino
- PULPissimo

Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

Our main research is on Near-Threshold Multi-Core Systems

RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

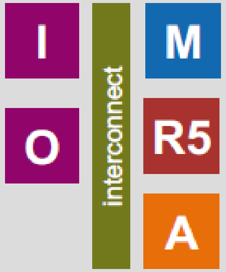
Peripherals

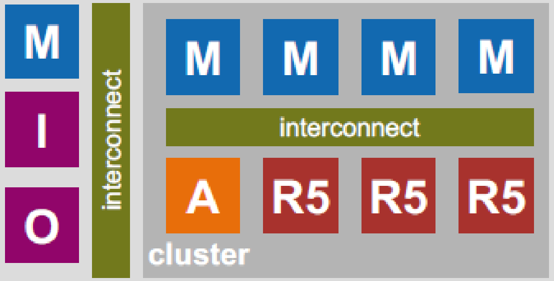
- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Platforms





Single Core

- PULPino
- PULPissimo

Multi-core

- Fulmine
- Mr. Wolf

Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

Finally for HPC applications we have multi-cluster systems

RISC-V Cores

RI5CY 32b	Micro riscy 32b	Zero riscy 32b	Ariane 64b

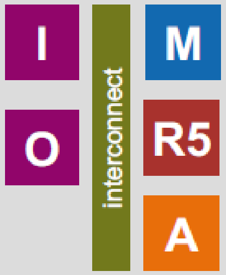
Peripherals

JTAG	SPI
UART	I2S
DMA	GPIO

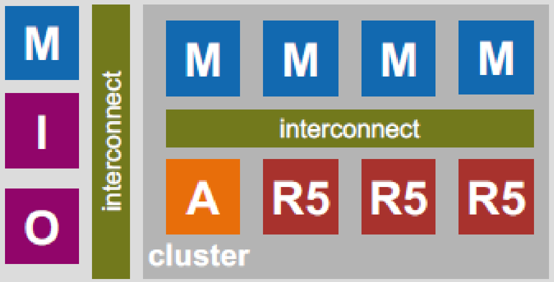
Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

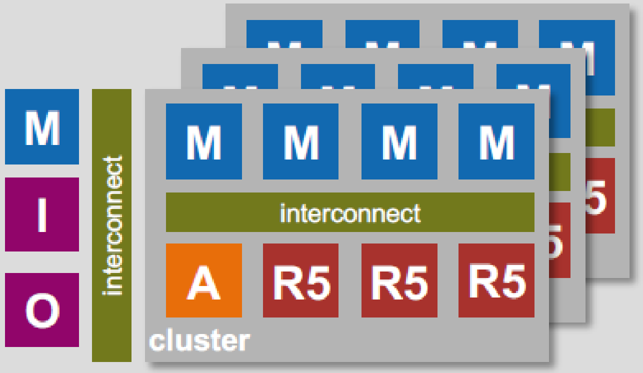
Platforms



Single Core
- PULPino
- PULPissimo



Multi-core
- Fulmine
- Mr. Wolf



Multi-cluster
- Hero

IOT → HPC

Accelerators

HWCE (convolution)	Neurostream (ML)	HWCrypt (crypto)	PULPO (1st order opt)

Eventually we plan to release ALL we did on PULP

RISC-V Cores

- RI5CY 32b — Open
- Micro riscy 32b — Open
- Zero riscy 32b — Open
- Ariane 64b — Open

Peripherals

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

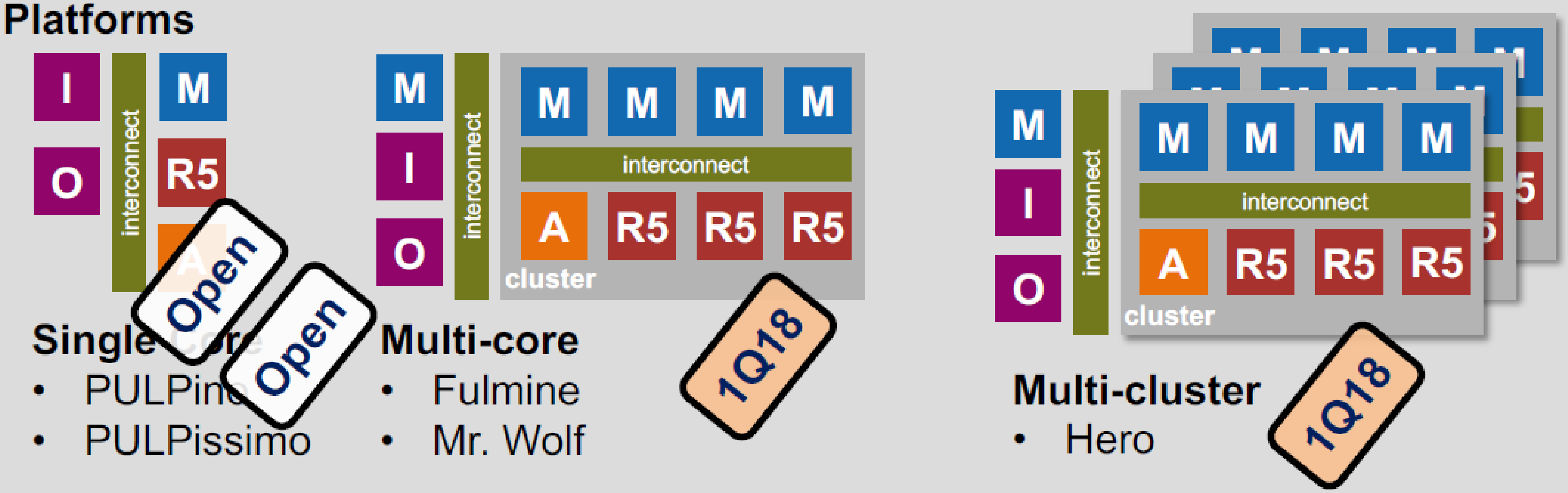
Open

Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Open

Platforms



Single Core
- PULPino
- PULPissimo

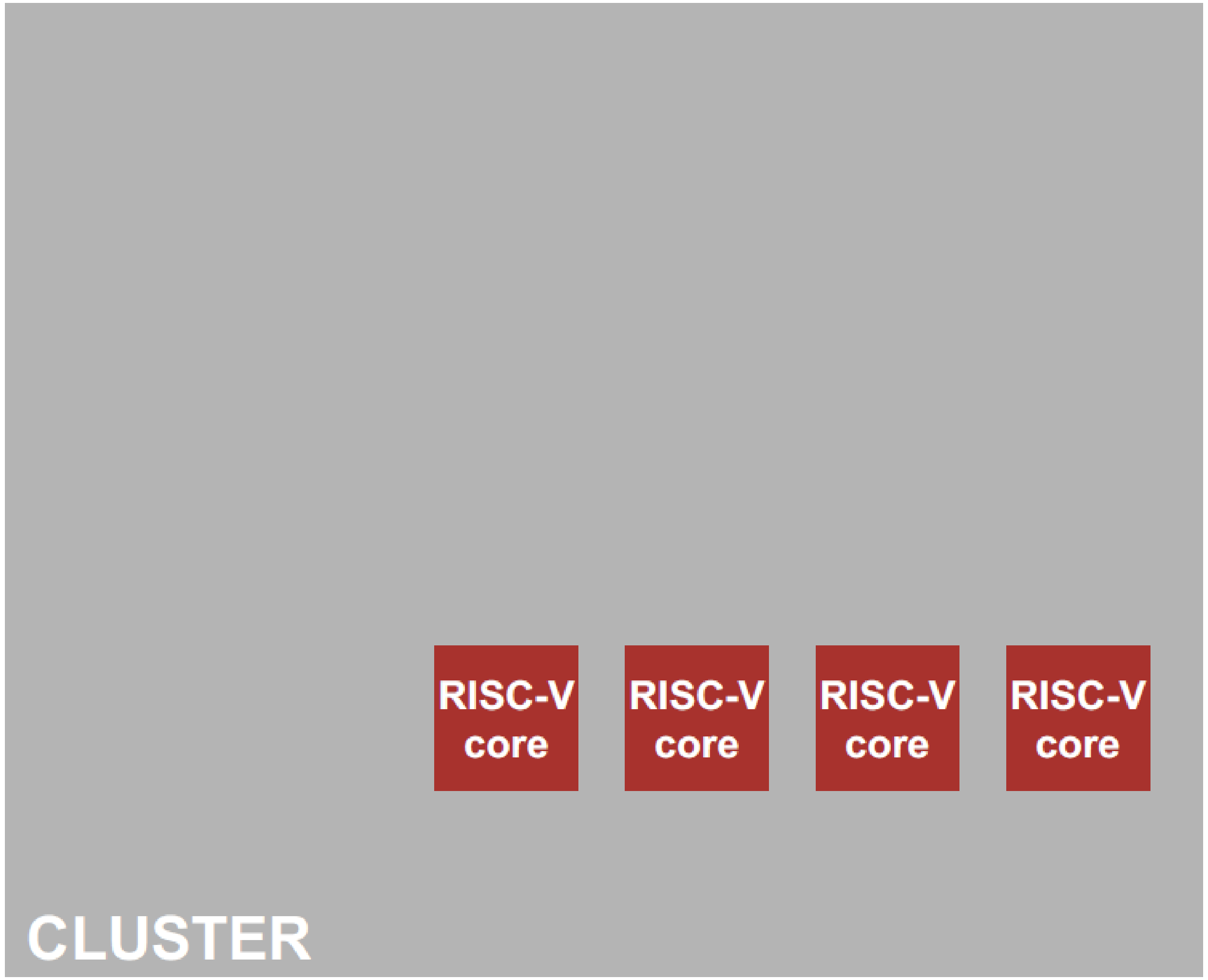
Multi-core
- Fulmine
- Mr. Wolf

Multi-cluster
- Hero

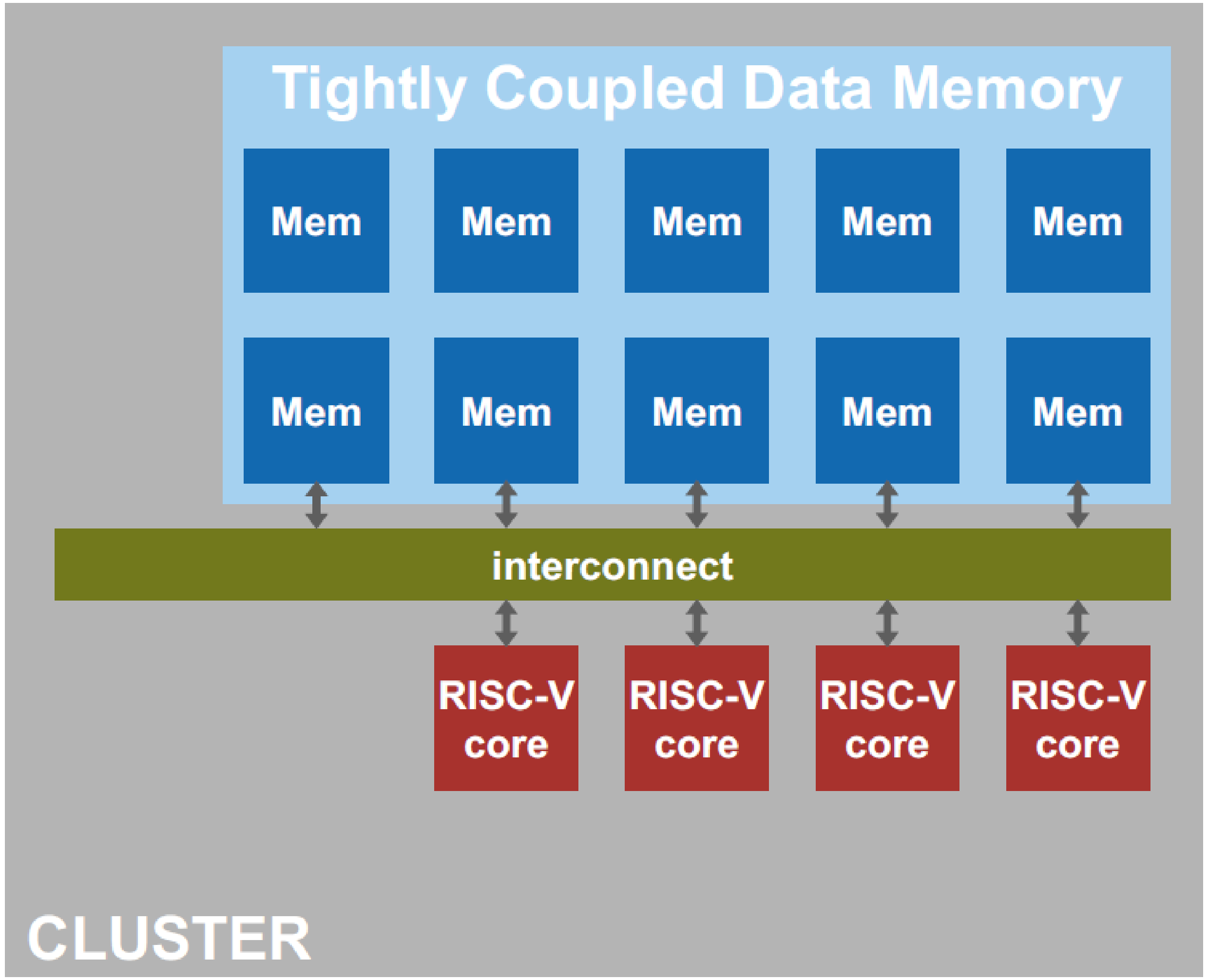
Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

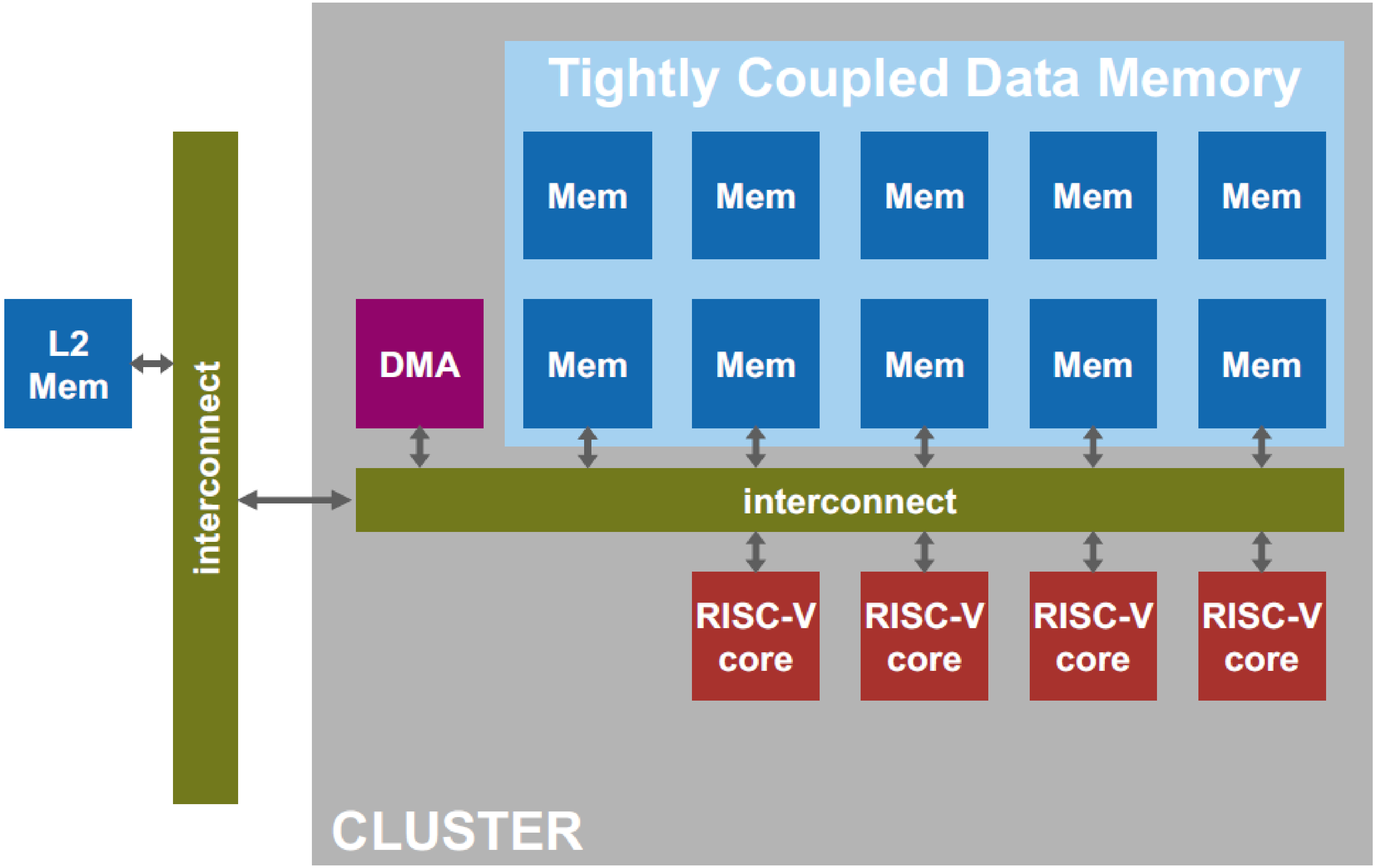
PULP cluster contains multiple RISC-V cores



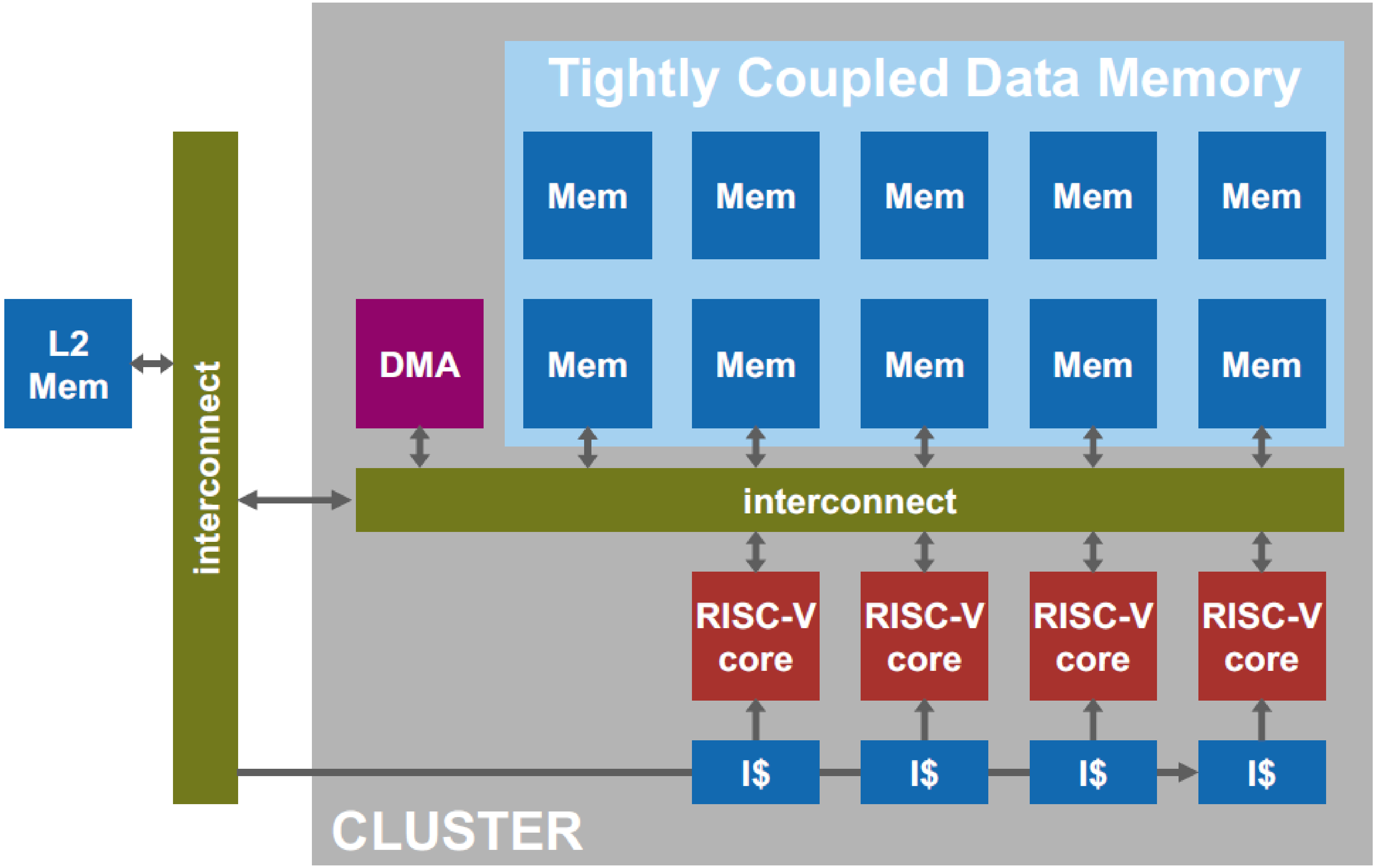
All cores can access all memory banks in the cluster



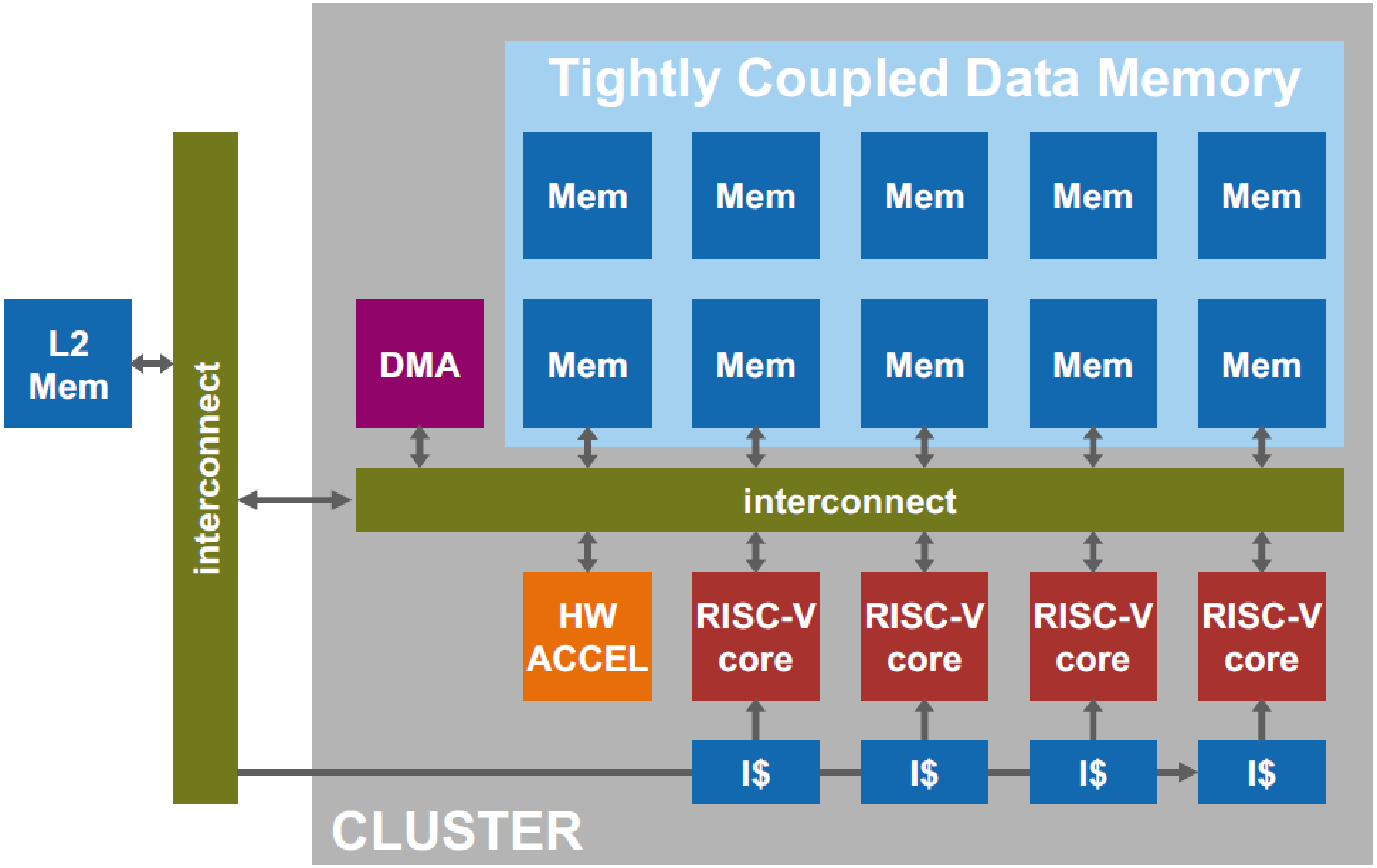
Data is copied from a higher level through DMA



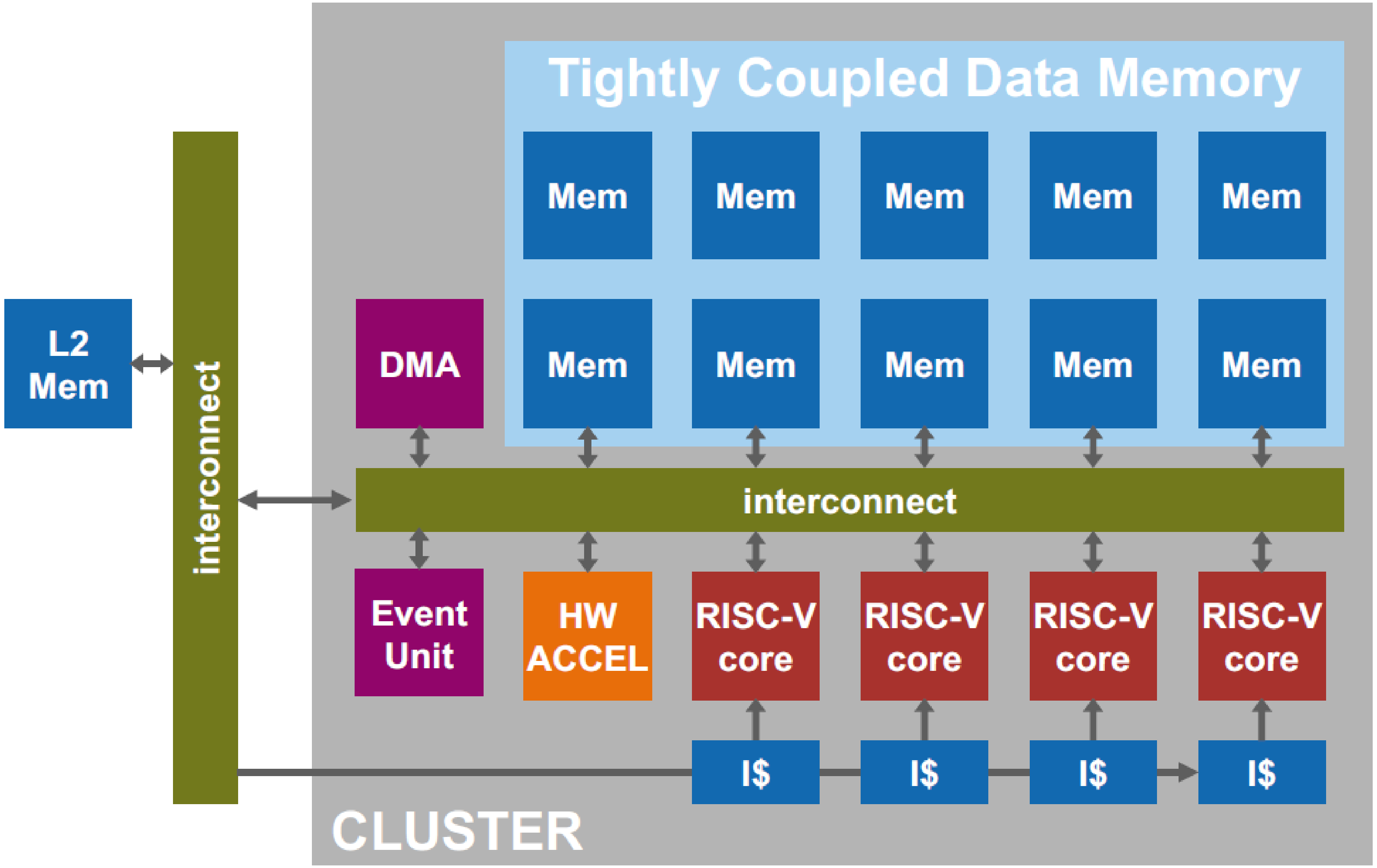
There is a (shared) instruction cache that fetches from L2



Hardware Accelerators can be added to the cluster

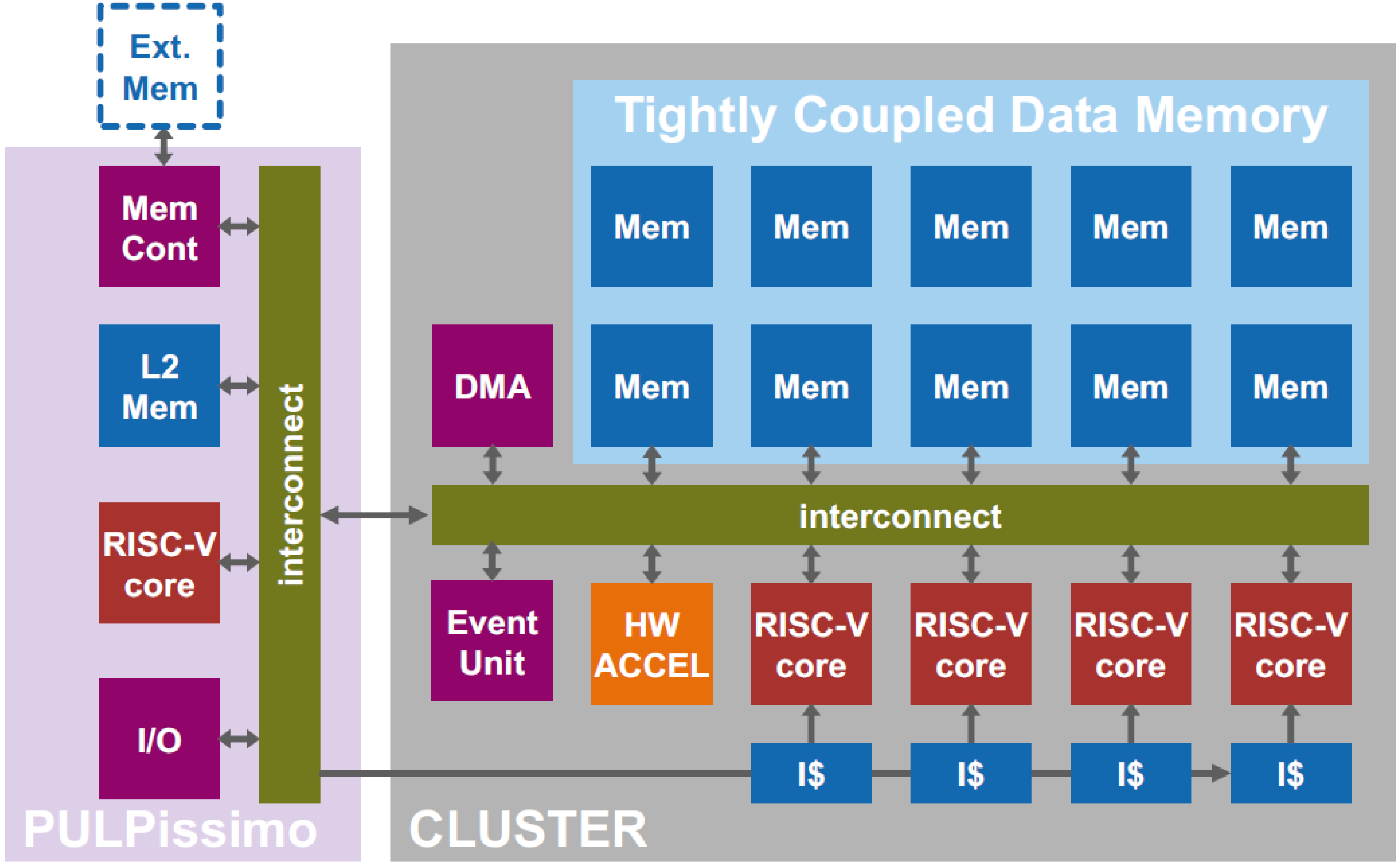


Event unit to manage resources (fast sleep/wakeup)

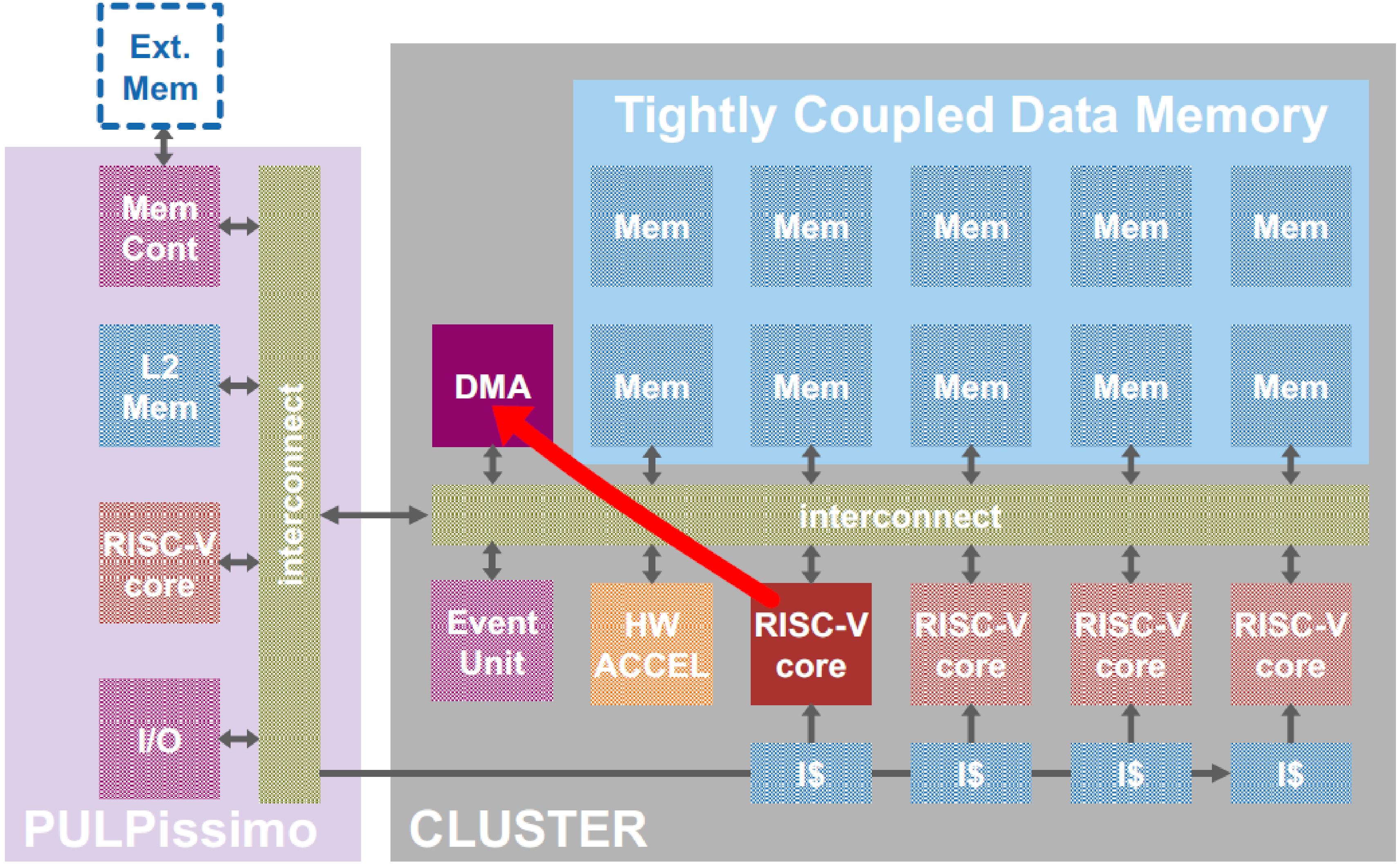


PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

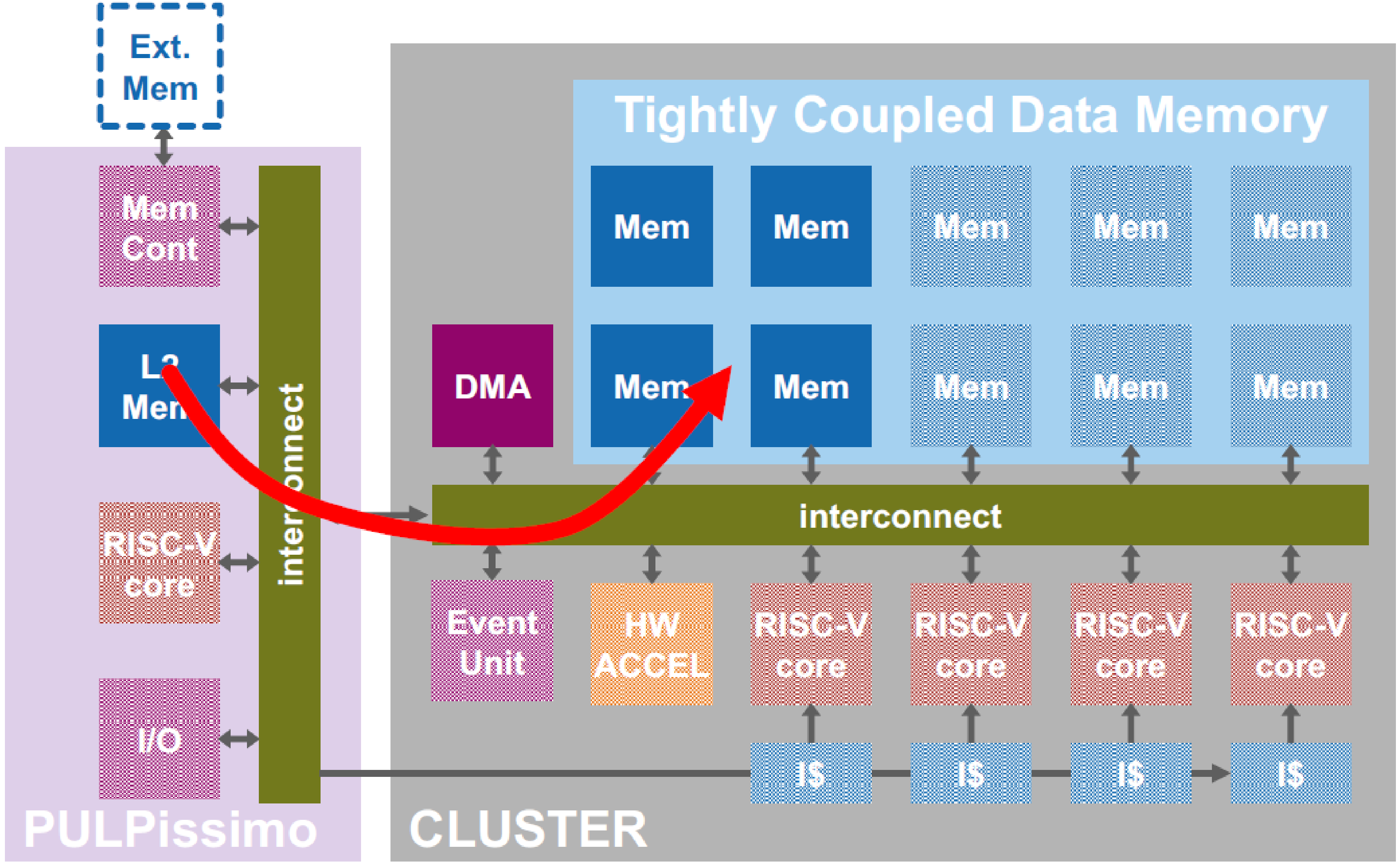
An additional microcontroller system (PULPissimo) for I/O



How do we work: Initiate a DMA transfer

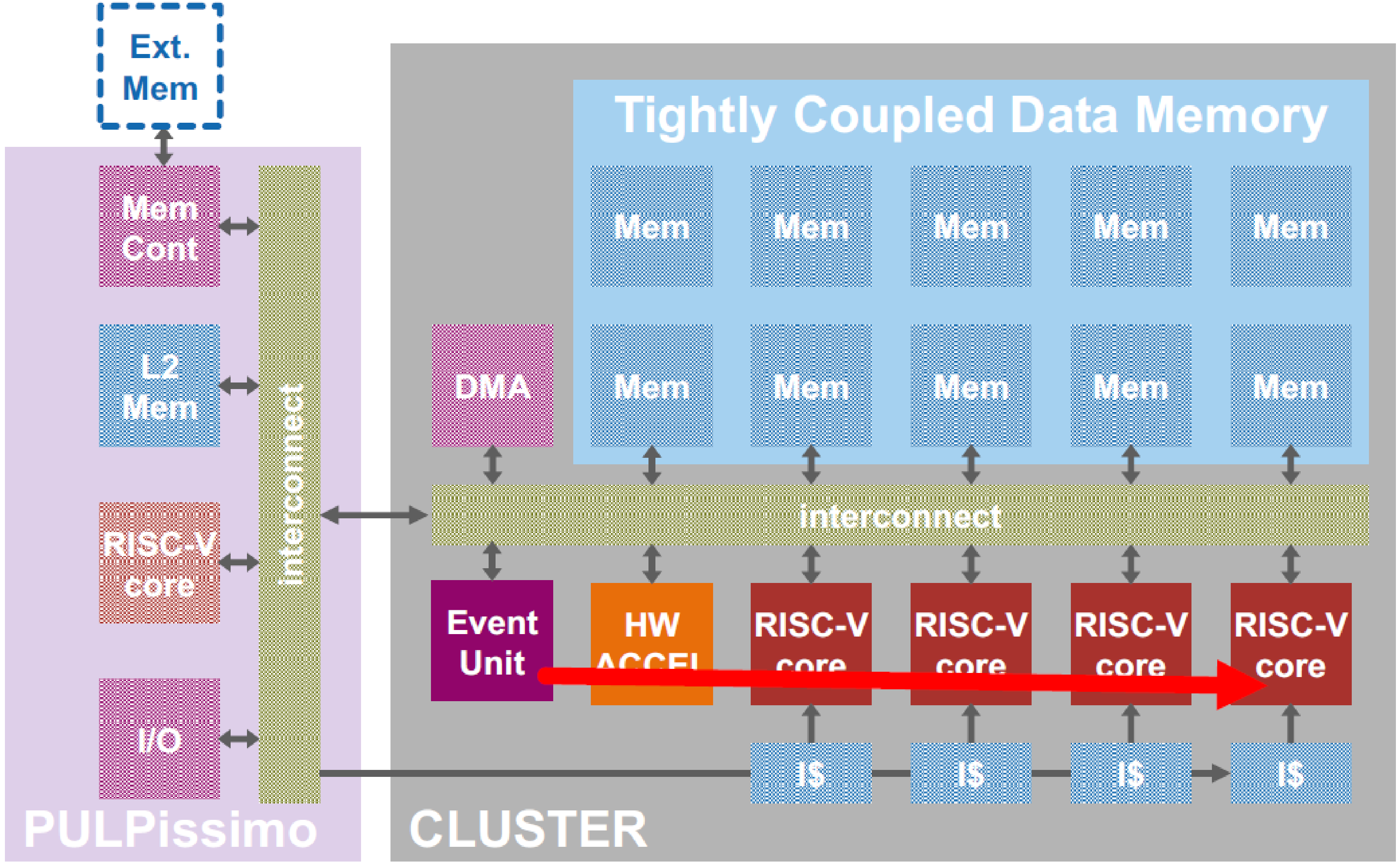


Data copied from L2 into TCDM

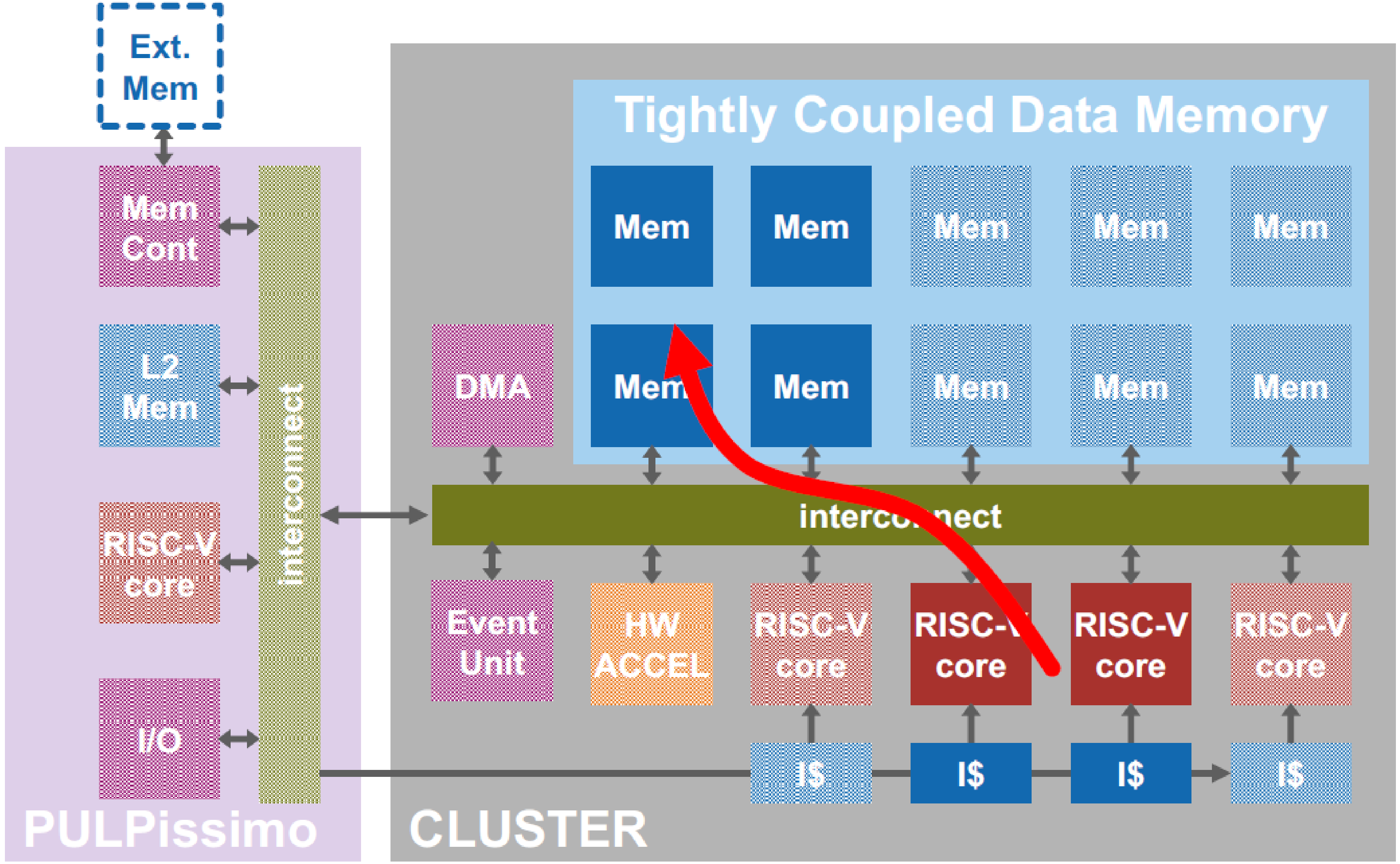


PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

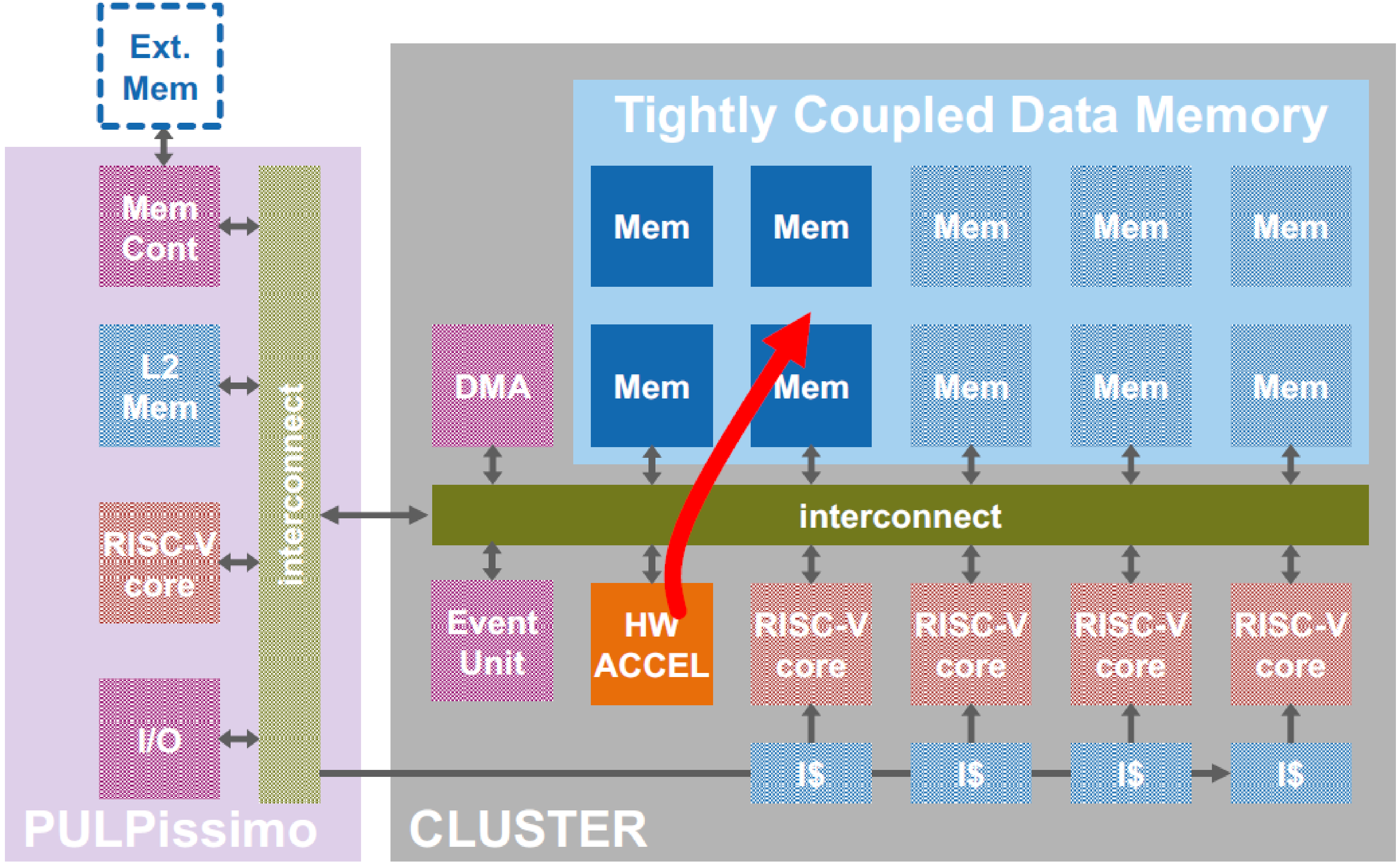
Once data is transferred, event unit notifies cores/accel



Cores can work on the data transferred

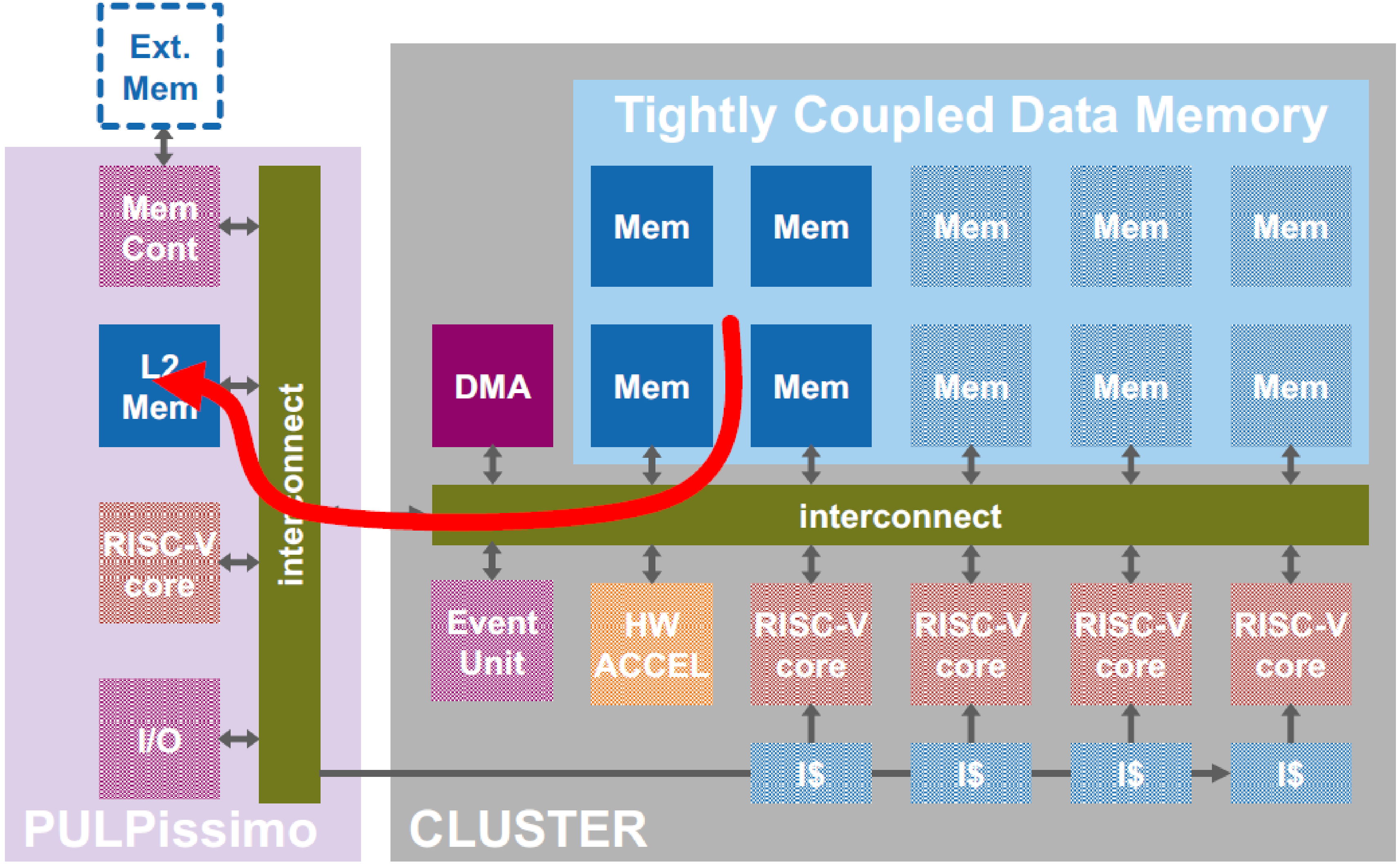


Or accelerators

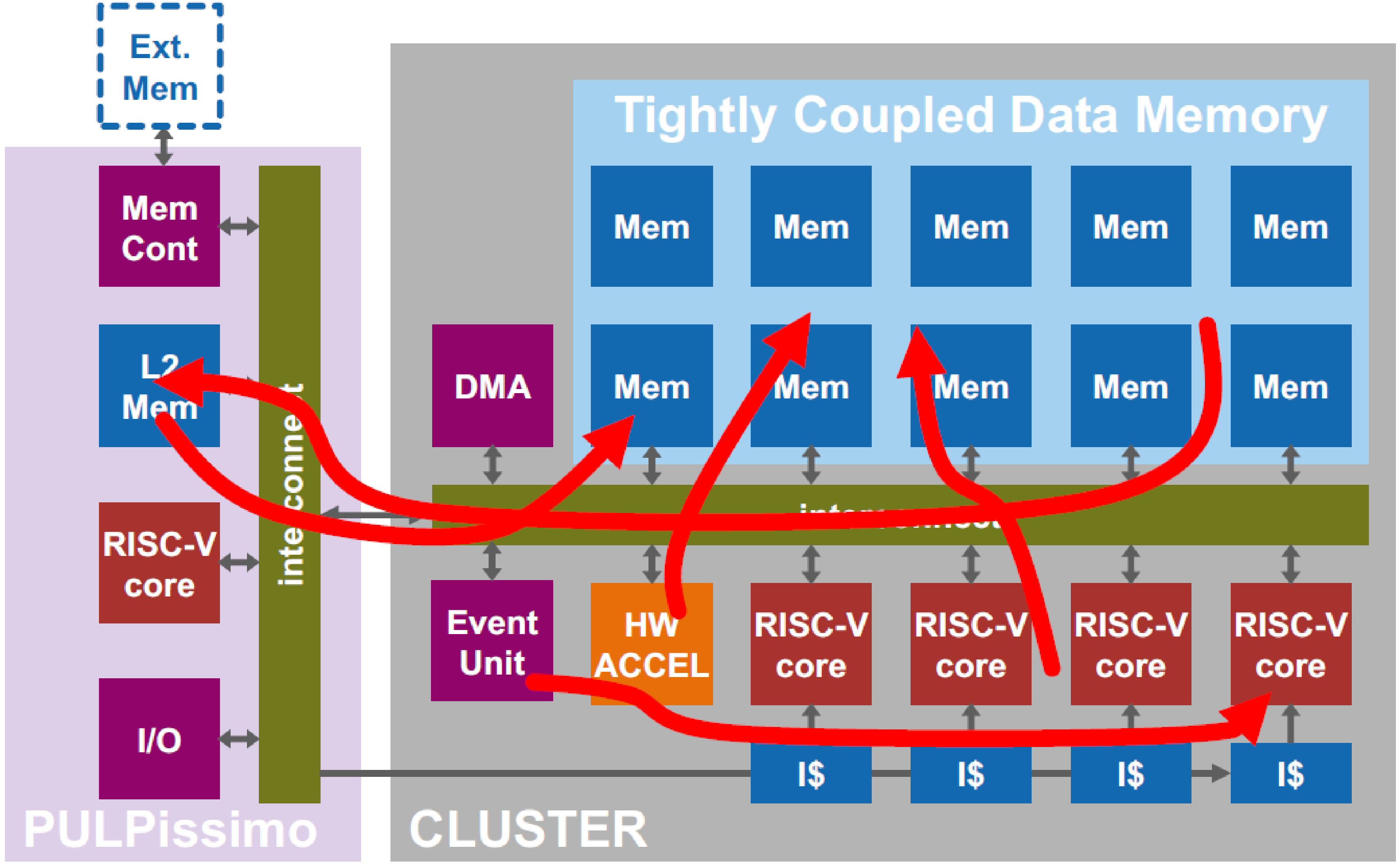


PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

Once our work is done, DMA copies data away



DMA data copies and processing actually work in parallel



RISC-V cores under development

32 bit			64 bit
Low Cost Core	Core with DSP enhancements	Floating-point capable Core	Linux capable Core
▪ **Zero-riscy** ▪ RV32-ICM ▪ **Micro-riscy** ▪ RV32-CE	▪ **RI5CY** ▪ RV32-ICMX ▪ SIMD ▪ HW loops ▪ Bit Man ▪ Fixed point	▪ **RI5CY + FPU** ▪ RV32-ICMFX	▪ **Ariane** ▪ RV64-IC(MA) ▪ Full privileged specification

I	Integer instructions (frozen)
E	Reduced number of registers
M	Multiplication and Division (frozen)
A	Atomic instructions (frozen)
F	Single-Precision Floating-Point (frozen)
D	Double-Precision Floating-Point (frozen)
C	Compressed Instructions (frozen)
X	Non Standard Extensions

PULP Open-Source Releases and External Contributions

1. **February 2016**
   First release of **PULPino**, our single-core microcontroller

2. **May 2016**
   Toolchain and compiler for our RISC-V implementation (**RI5CY**), DSP extensions

3. **August 2017**
   PULPino updates, new cores Zero-riscy and Micro-riscy, **FPU**, toolchain updates

4. **February 2018**
   PULPissimo, ARIANE, PULP

5. **A bit later in 2018**
   PULP, HERO

PULP Success

- **Many companies (we know of) are actively using PULP**
  - They value that it is **silicon proven**
  - They like that it uses a **permissive open source license**

Companies with announced products, business
Companies that use PULP internally or for training
Companies exploring opportunities

Companies that are using/evaluating PULP

- GreenWaves Technologies
- Dolphin
- IQ Analog (14nm chips)
- Embecosm
- lowRISC
- Mentor Graphics
- Cadence Design Systems
- ST Microelectronics (IT,F)
- Micron
- SIAE Microelectronica
- Advanced Circuit Pursuit
- NXP
- Shanghai Xidian Technology
- SCS Zurich
- IMT technologies
- **Google**
- Microsemi
- Arduino
- RacyICs

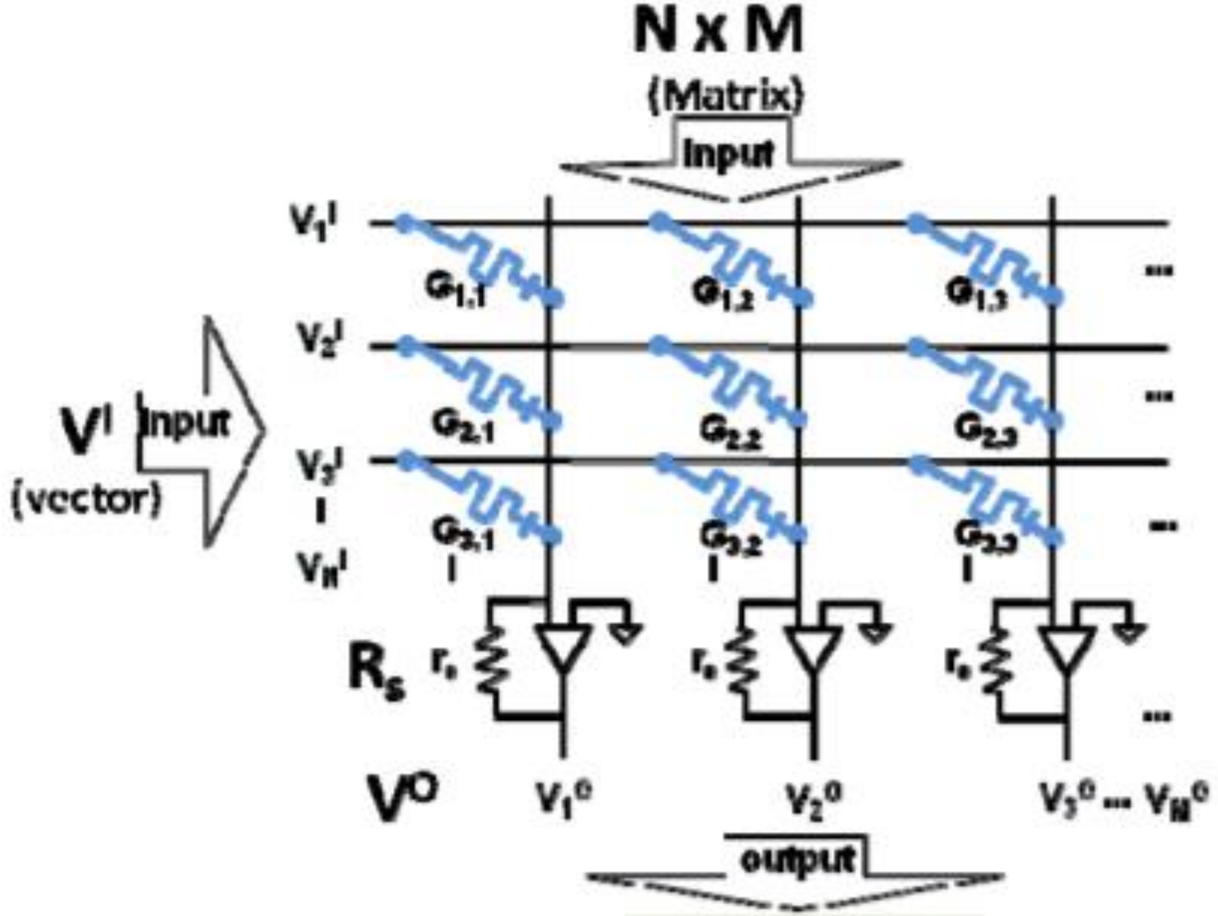
Research Centers/Universities using PULP

- Stanford
- Cambridge
- UCLA
- CEA/LETI
- EPFL
- National Chia Tung University
- Politecnico di Milano
- Politecnico di Torino
- Universita Roma I
- Instituto Superior Tecnico – U. de Lisboa
- Fondazione Bruno Kessler
- Zagreb HER
- Universita di Genova
- Istanbul Technical U.
- RWTH Aachen
- Lund
- USI – Lugano
- Bar-Ilan
- TU-Kaiserslautern
- TU-Graz
- UC San Diego
- CSEM
- IBM Research

A Few Words about My Project

- Design new application-dedicated logic elements, that will be energy efficient, and integrate them in the PULP processor:
  - A CCLO (Configurable Combinational Logic Operator) unit
    - enables the realization of application specific operations and software customization. For example, LUTs
  - A dot-product unit, which calculates matrix and vector multiplication
    - Uses a memristors crossbar
    - Useful for machine-learning applications

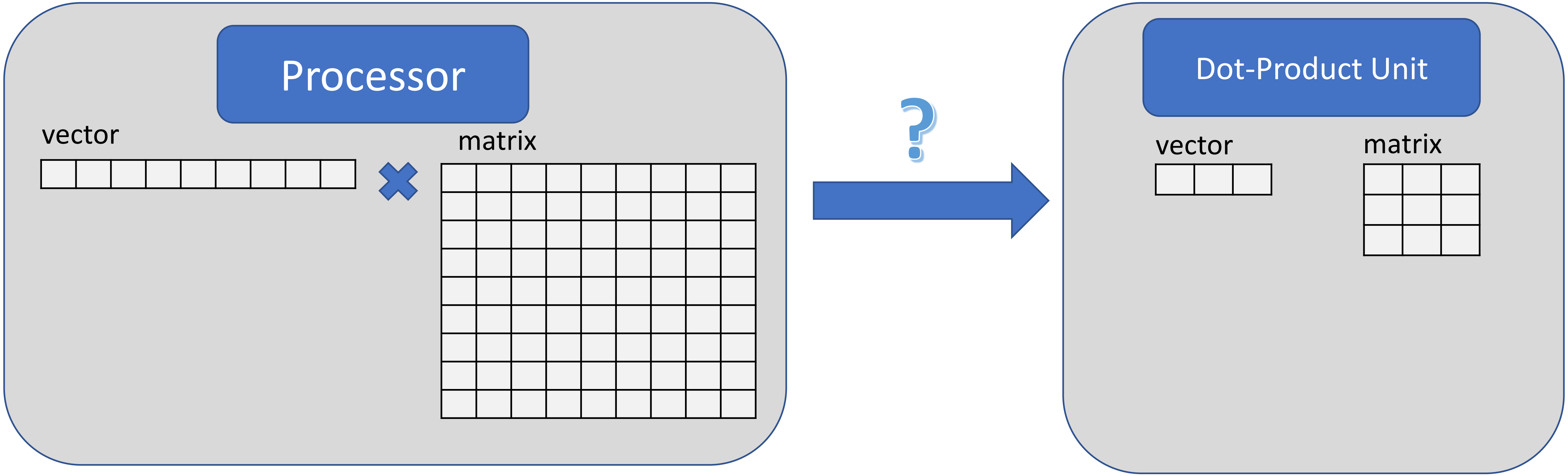
Calculating Multiplication using a Crossbar



$$V^o = V^I G R_S$$

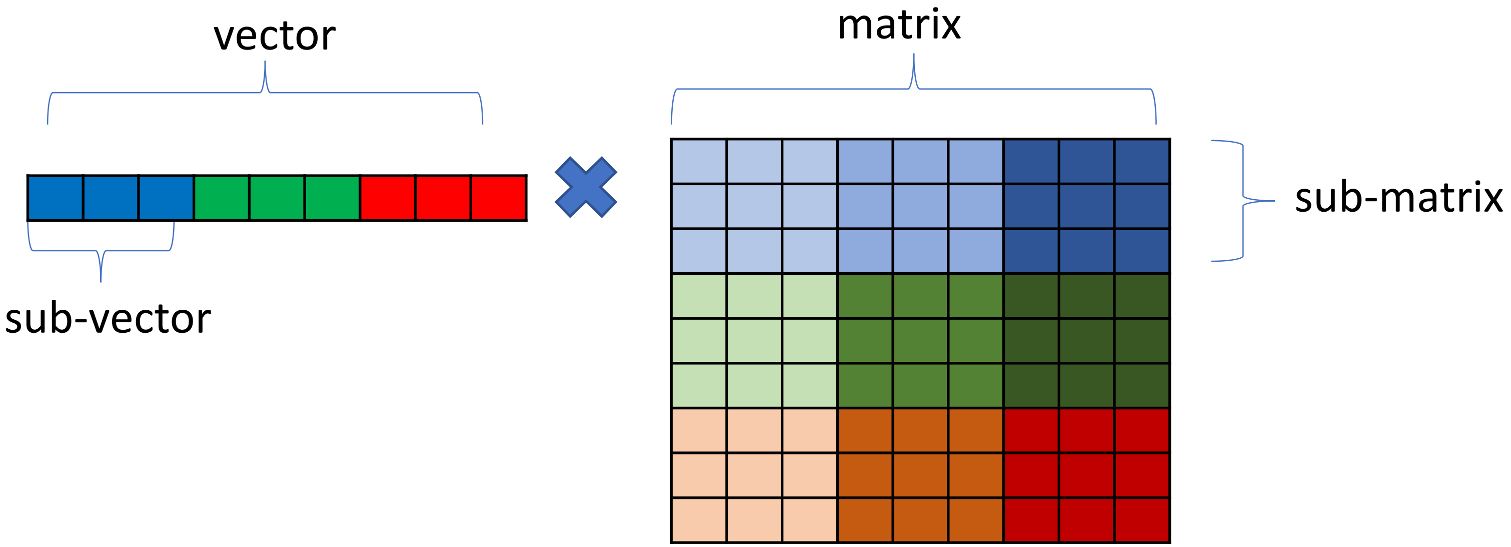
The Problem: Multiplying Large Vector and Large Matrix

- The Dot-Product unit is not large enough to store large vectors and large matrices.



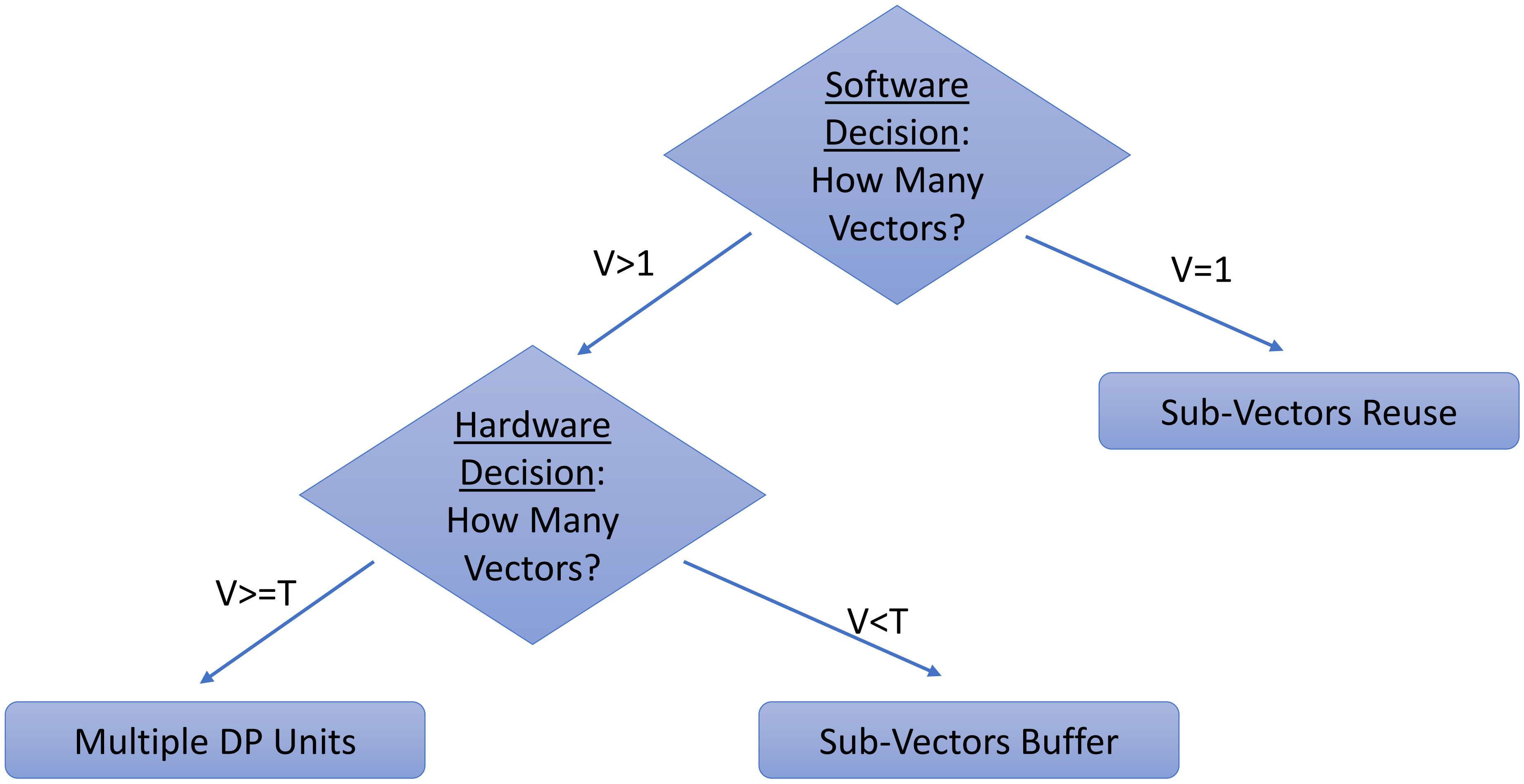
The Solution: Dividing The Vector & Matrix

- Dividing the vector to sub-vectors, and the matrix to sub-matrices:



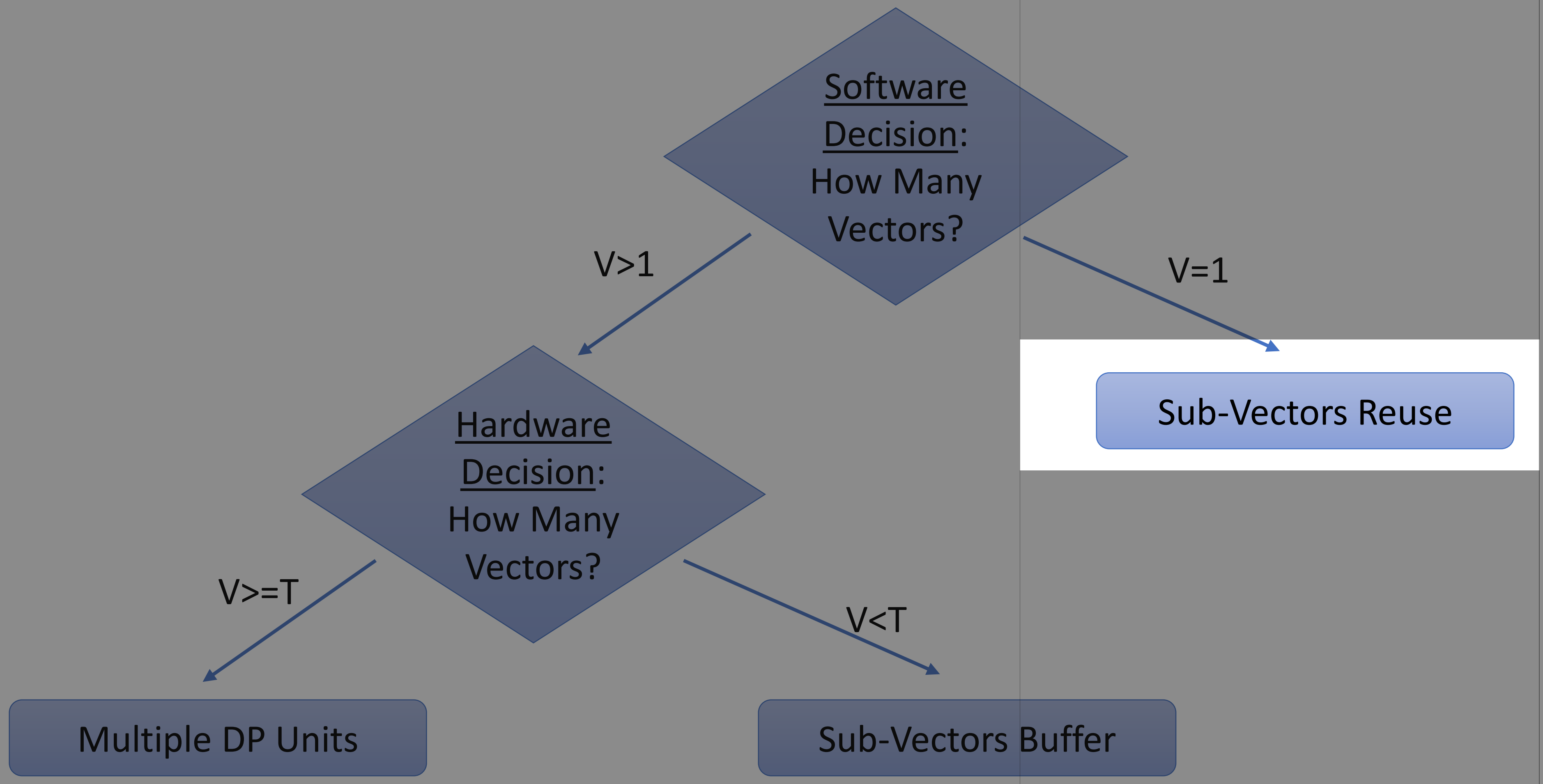
- The blue sub-matrices are multiplied by the blue sub-vector, the green ones with the green vector, etc.
- Additional calculations (summing the results) are performed in the processor.

Multiplication Flow



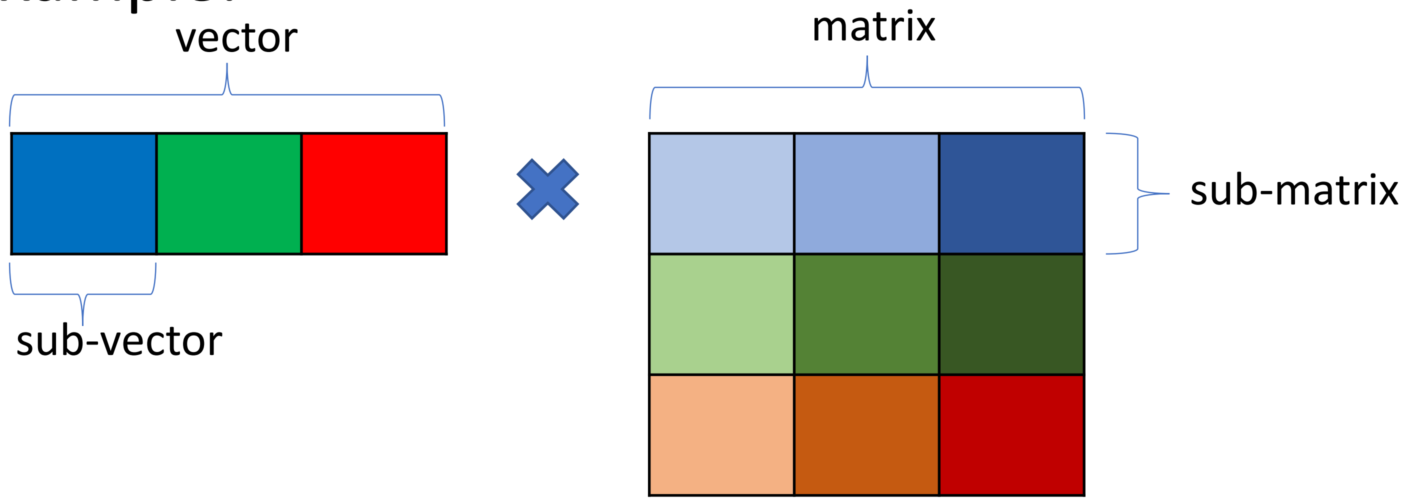
Software Decision: How Many Vectors?
V>1
V=1
Sub-Vectors Reuse
Hardware Decision: How Many Vectors?
V>=T
V<T
Multiple DP Units
Sub-Vectors Buffer

Multiplication Flow


Software Decision: How Many Vectors?
V>1
V=1
Sub-Vectors Reuse
Hardware Decision: How Many Vectors?
V>=T
V<T
Multiple DP Units
Sub-Vectors Buffer

One Vector: Sub-Vectors Reuse

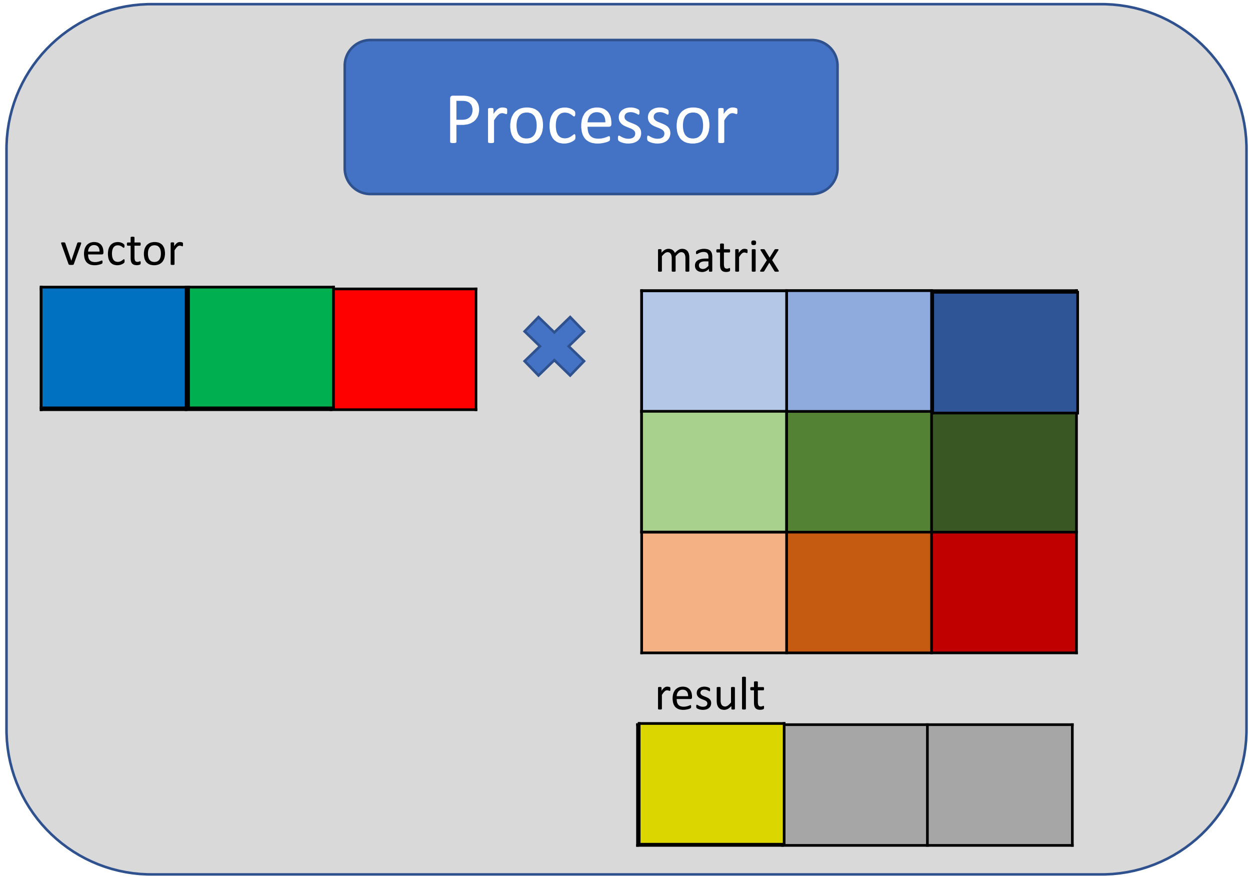
- A simple example:

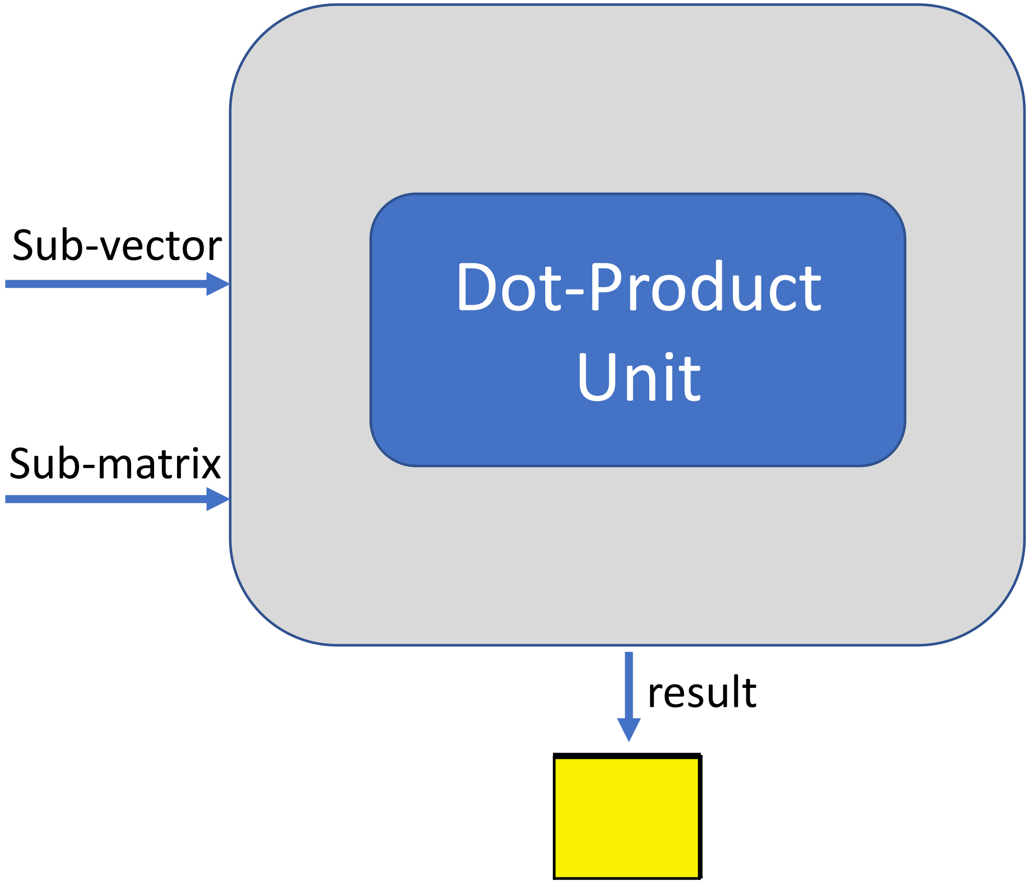


- The blue sub-matrices are multiplied by the blue sub-vector, the green ones with the green vector, etc.
- Each sub-matrix is used once. Each sub-vector is used 3 times.

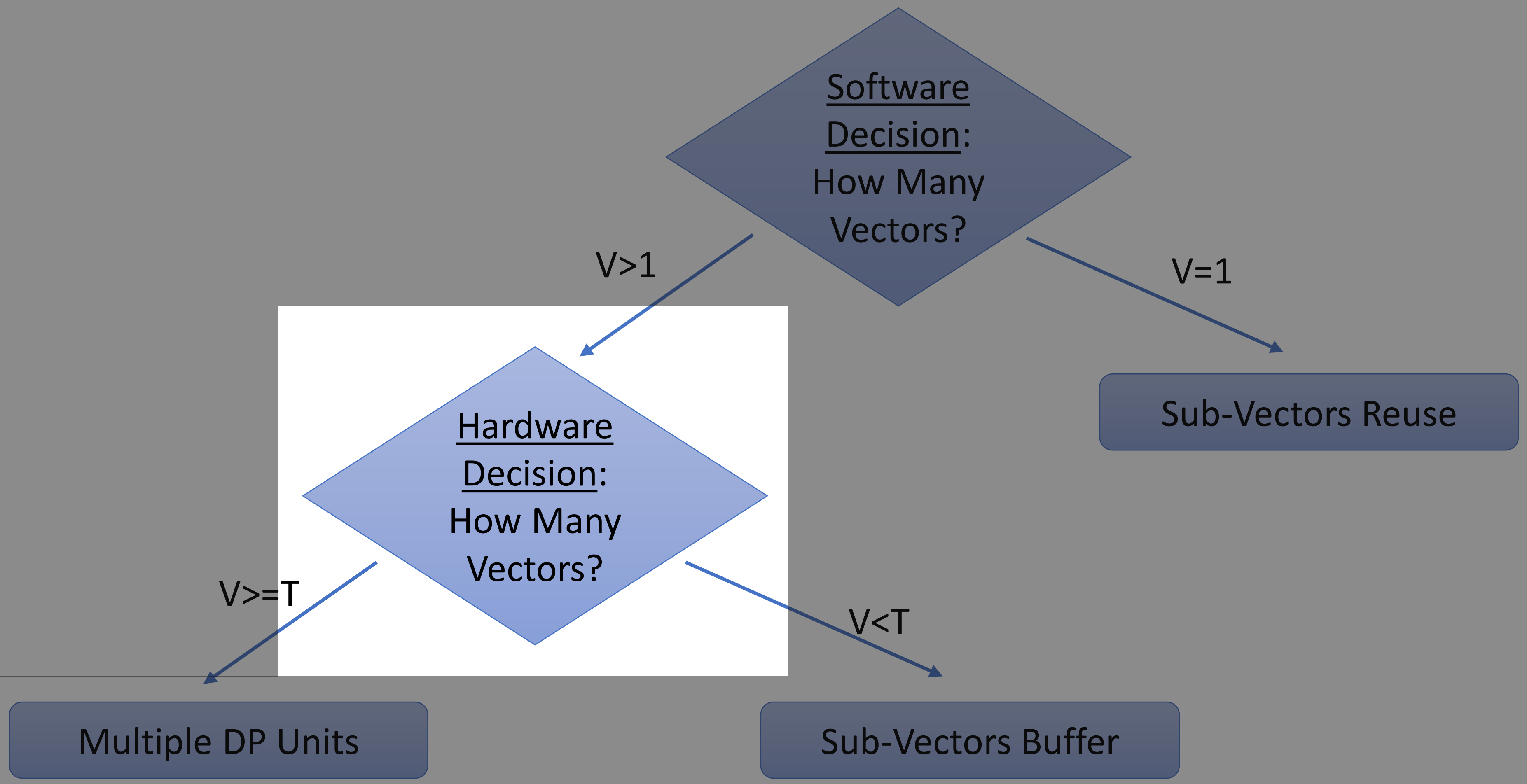
One Vector: Sub-Vectors Reuse

- We first write a sub-vector and then all the relevant row sub-matrices:



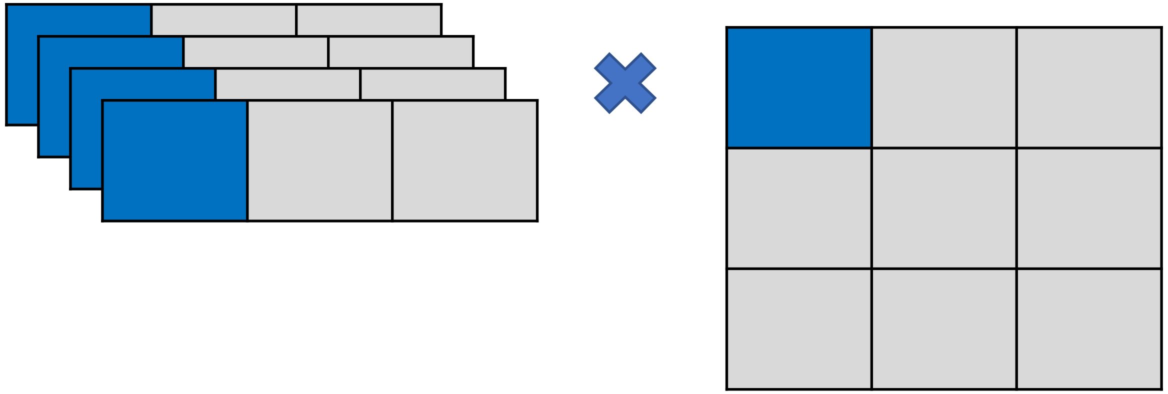


Multiplication Flow


Software Decision: How Many Vectors?
V>1
V=1
Sub-Vectors Reuse
Hardware Decision: How Many Vectors?
V>=T
V<T
Multiple DP Units
Sub-Vectors Buffer

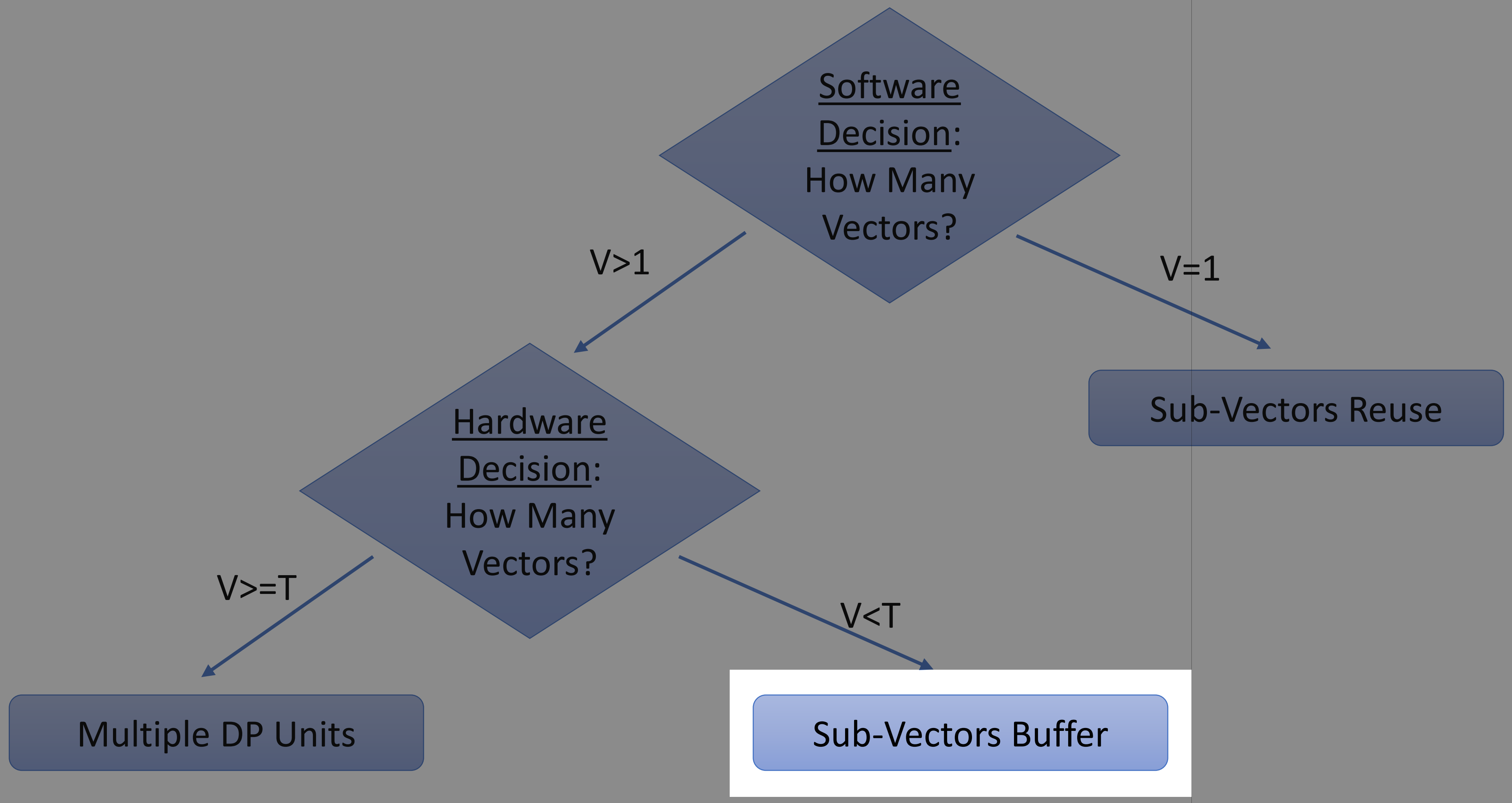
Multiple Vectors: Sub-Matrices Reuse

- When having multiple vectors, each sub-matrix is used more than once.
- In the following example, the blue sub-matrix is multiplied by all the blue sub-vectors:



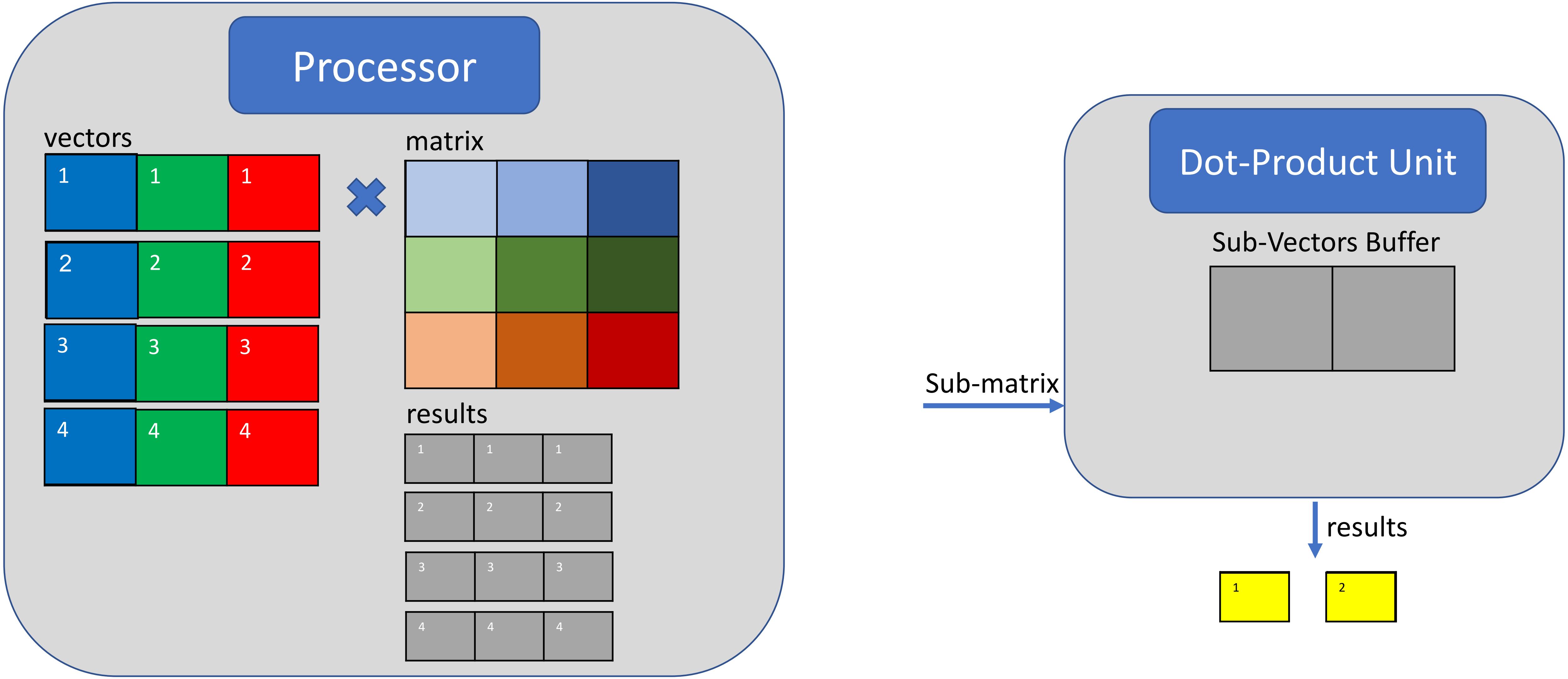
- Since matrices are larger than vectors, we’d rather reuse sub-matrices rather then reuse sub-vectors, like before.

Multiplication Flow


Software Decision: How Many Vectors?
V>1
V=1
Sub-Vectors Reuse
Hardware Decision: How Many Vectors?
V>=T
V<T
Multiple DP Units
Sub-Vectors Buffer

Sub-Vectors Buffer

- We first write a sub-matrix and then all the relevant sub-vectors:

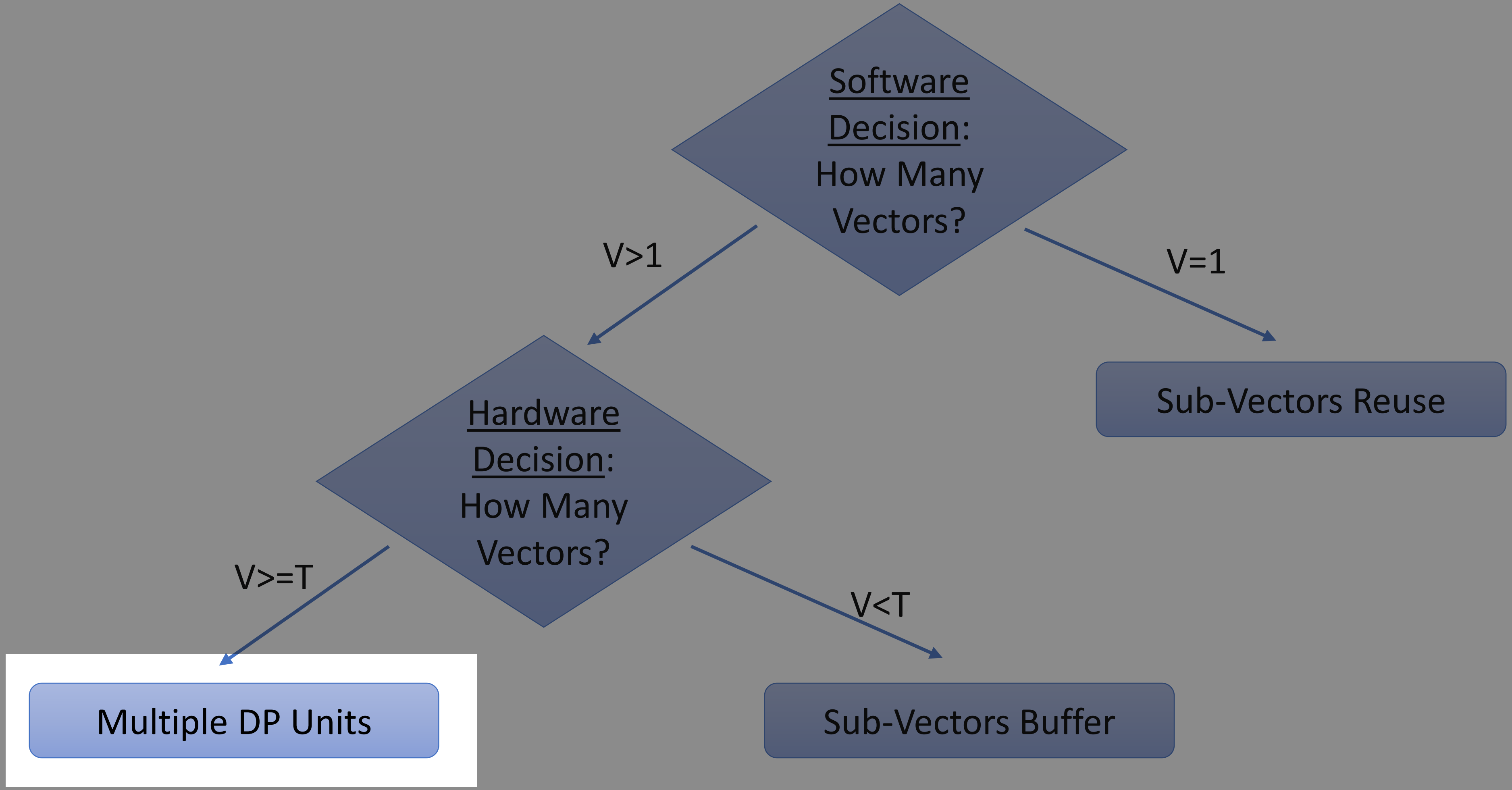


Sub-Vectors Buffer - Issues

- When the number of vectors isn’t a multiplication of the buffer size, the last group of vectors will take only part of the buffer. The reused sub-vectors in the next sub-matrix are not sorted, therefore the results should be placed in the right location:

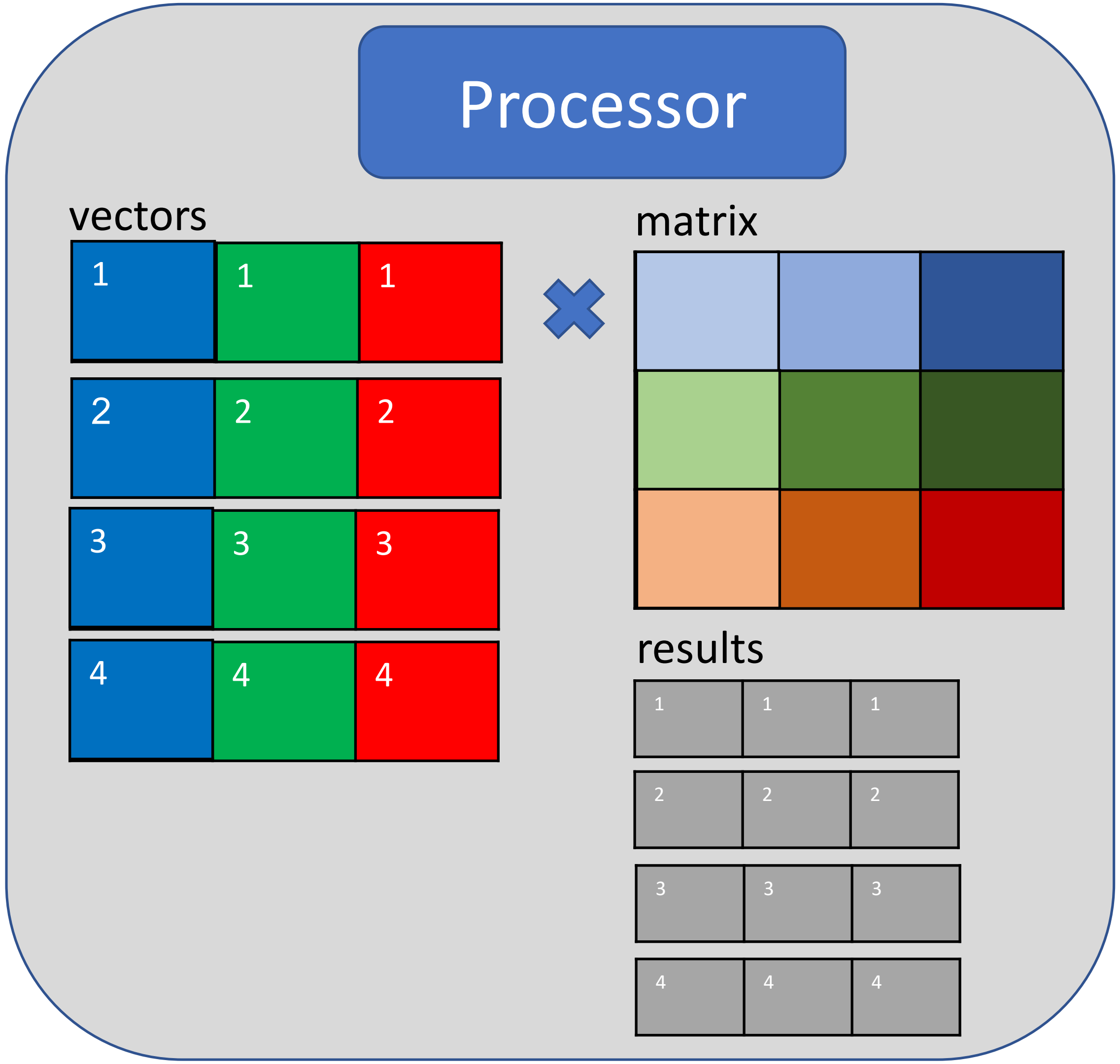
vectors
1
1
1
2
2
2
3
3
3
Sub-Vectors Buffer

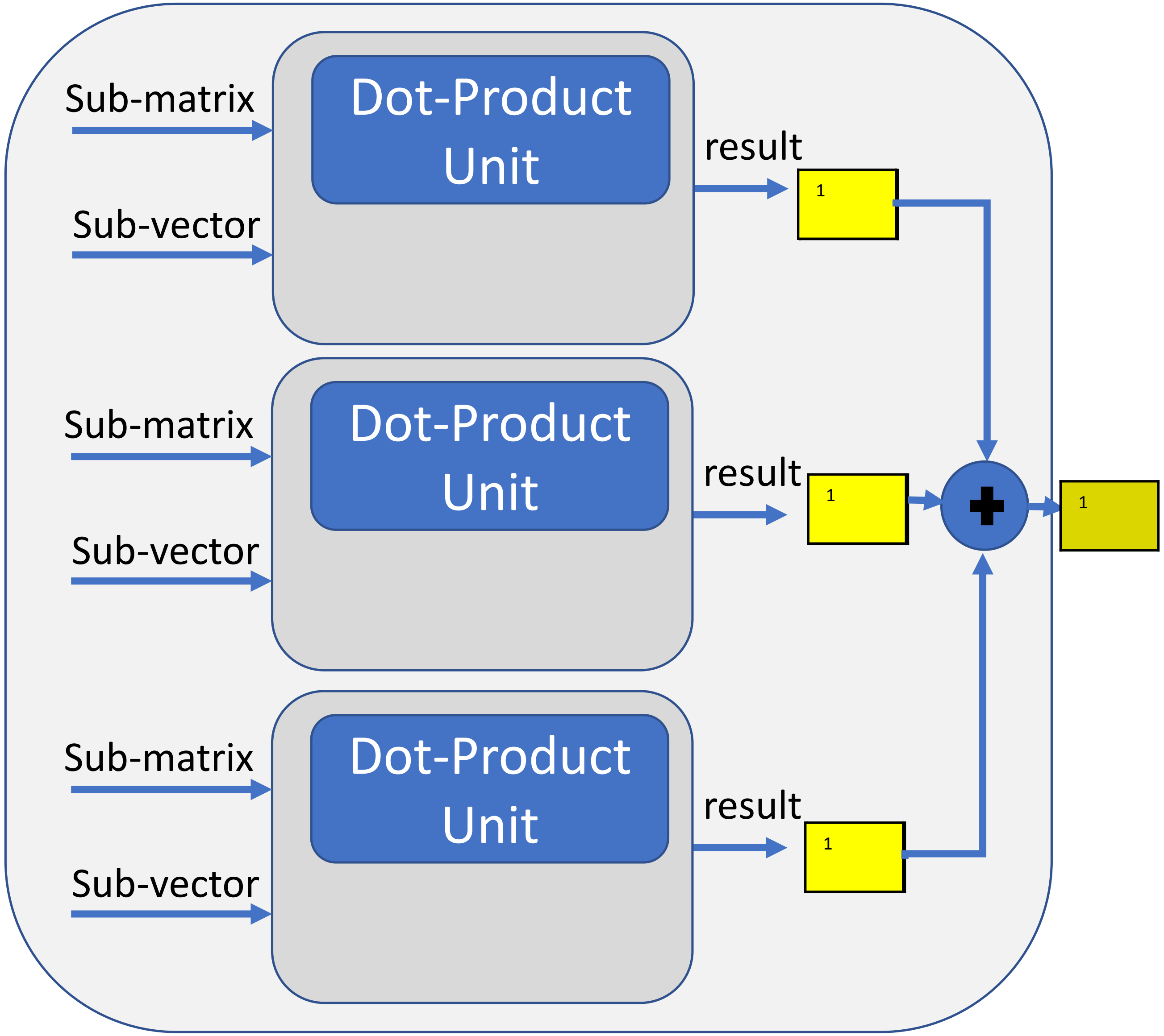
Multiplication Flow


Software Decision: How Many Vectors?
V>1
V=1
Sub-Vectors Reuse
Hardware Decision: How Many Vectors?
V>=T
V<T
Multiple DP Units
Sub-Vectors Buffer

Multiple DP Units

- We first write a sub-matrix and then all the relevant sub-vectors:





What’s Next?

- Developing an accelerator “by the book”
  - Using the accelerator DMA
- Cache prefetching
- Reducing the number of add cycles
- Real performance analysis

Thank You!