

Real Time additions to the CVA6 RISC-V Europe 2024

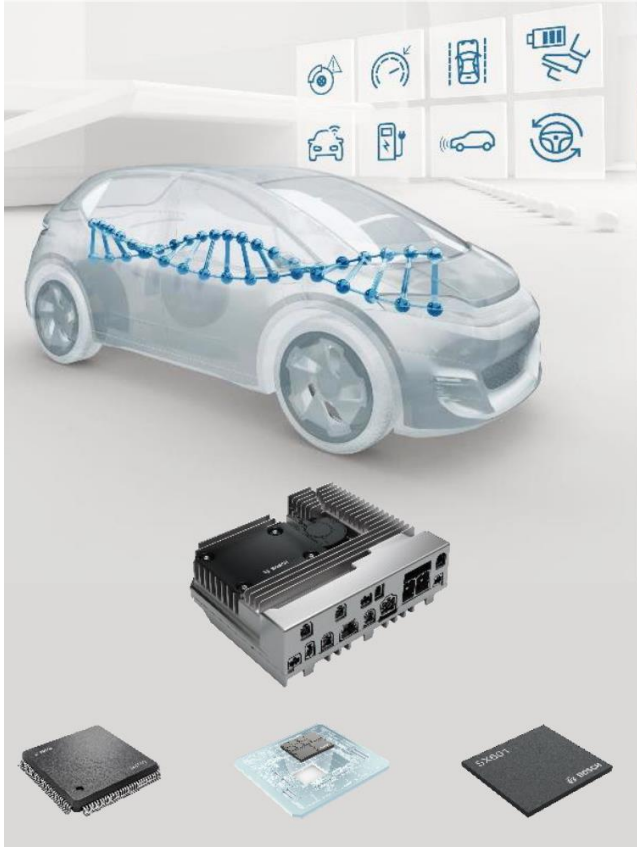
C. Allieux, O. Betschi, JC. Kircher, I. Schmid, G. Miet, R. Hardy,
N. Tribie

Bosch Mobility Electronics



Exploring RISC-V cores usage for Automotive Electronics

Software-defined vehicle



Electrification

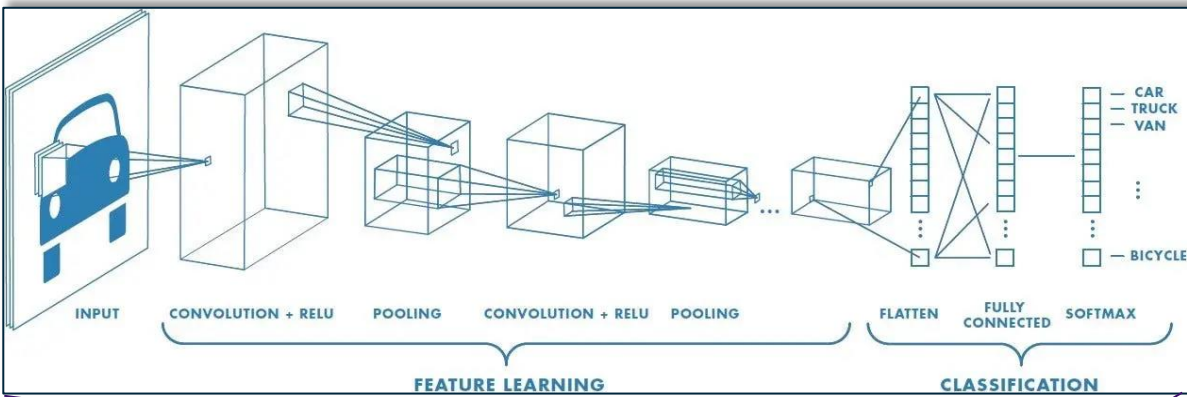
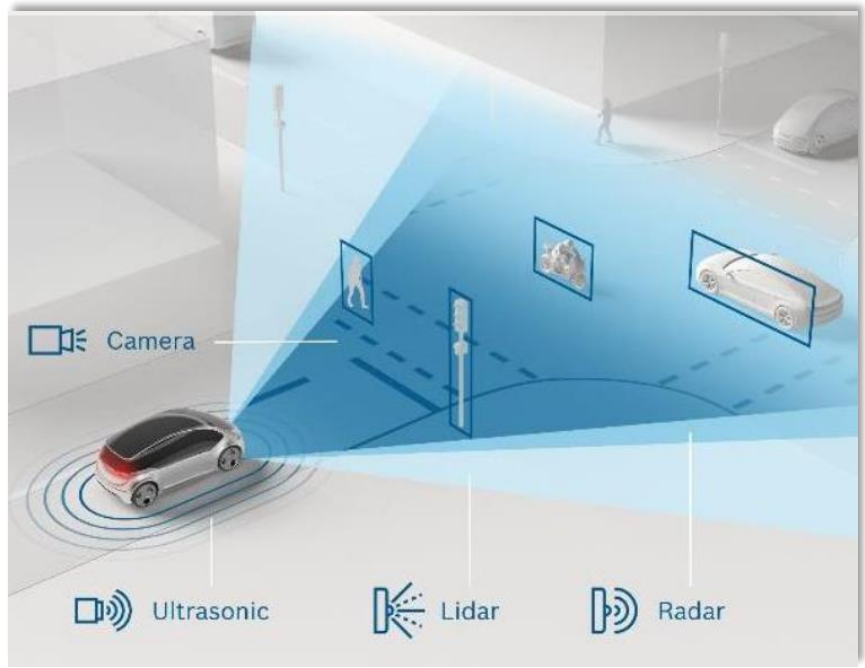


Autonomous driving

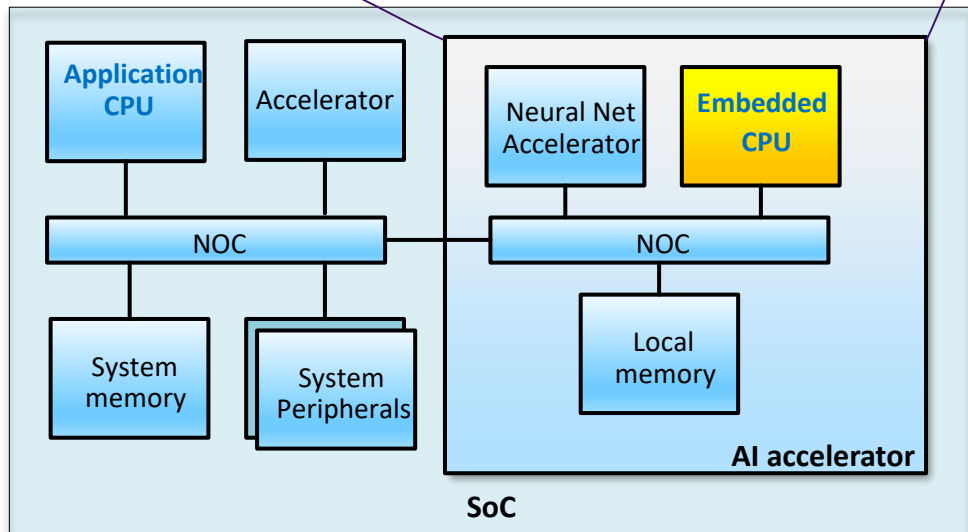


Demonstrator : RISC-V Core + Neural Network accelerator

ADAS* A.I : Camera/Radar Image classification



<https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/>



* : ADAS : Advanced Driver Assistance Systems

RISC-V Cores in our SoC demonstrator : CVxA6

CV64A6 "Host"

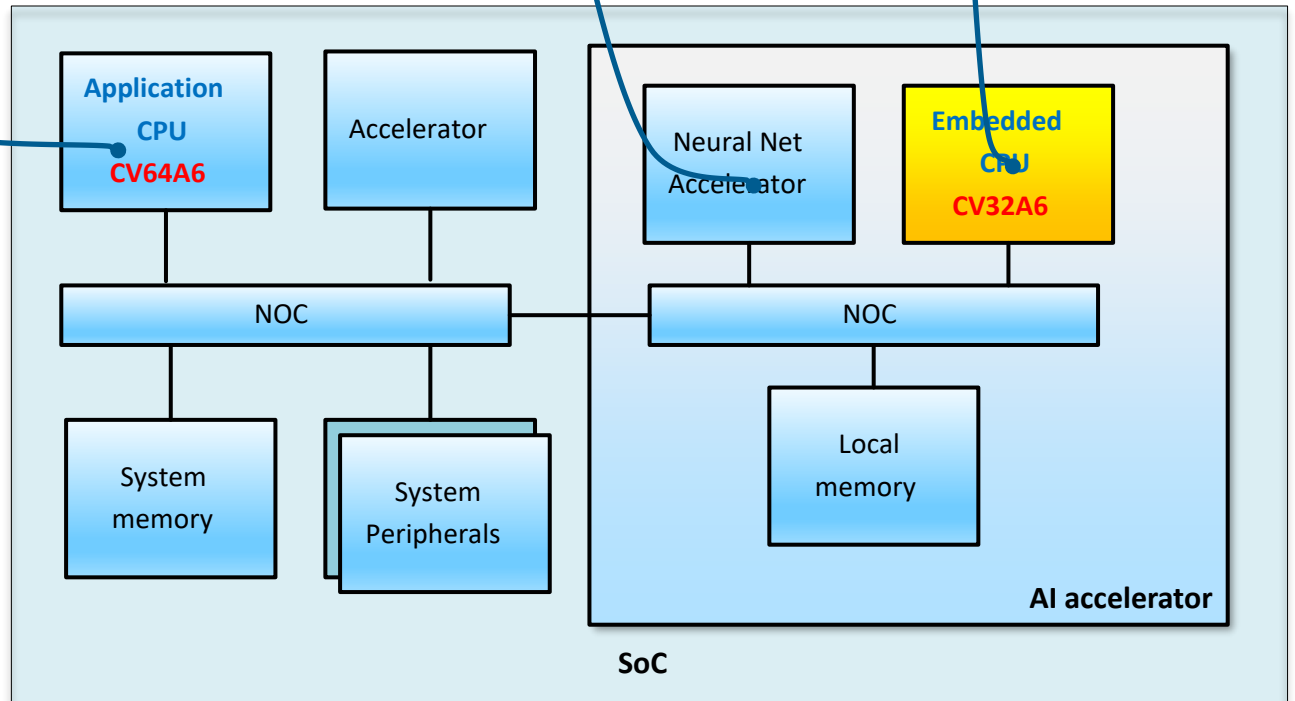
- High level application
- Possibly under Linux
- Communications

Neural Network Accelerator :

- CNNs computations
- Controlled by Embedded CPU

CV32A6 "embedded":

- Neural Net Acc. control
- CNNs Nonlinear operations,
- interface with application,
- Safety (SW part)



OPENHW GROUP RISC-V CVA6 Core :

- One of the most powerful RISC-V OSS Cores
- 6 stages pipeline, 32/64 bits IMACFD ISA
- Fully configurable
- Large user base : many OSS contributors
- 2.1 Coremark/MHz .. and growing
- 740 MHz achieved in GF 22 nm.

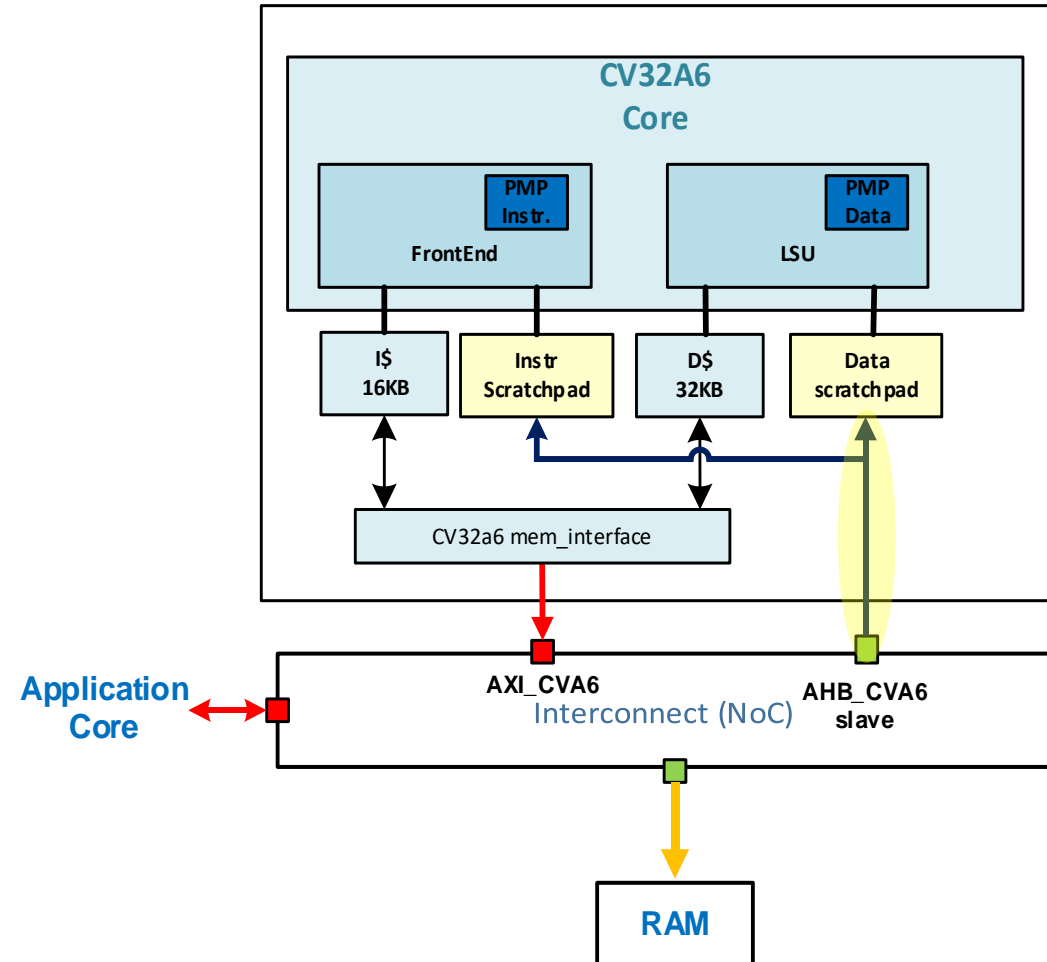
CV32A6 Core Real Time Additions : Scratchpads

Real time applications needs :

- execution determinism
- Critical sections execution time optimization :
 - Interrupt service (ISRs) latency reduction : stack, heap
 - DSP algorithms speed up
- Boot ROM replacement

Memory hierarchy changes:

- Add scratchpad memories, in // with caches:
 - **Instruction scratchpad** : critical routines, ISRs, boot code
 - **Data scratchpad** : stack, DSP coefficients
- Move **PMP** outside of **MMU** : keeping only memory protection
 - No Virtual => Physical Address translation timing uncertainty



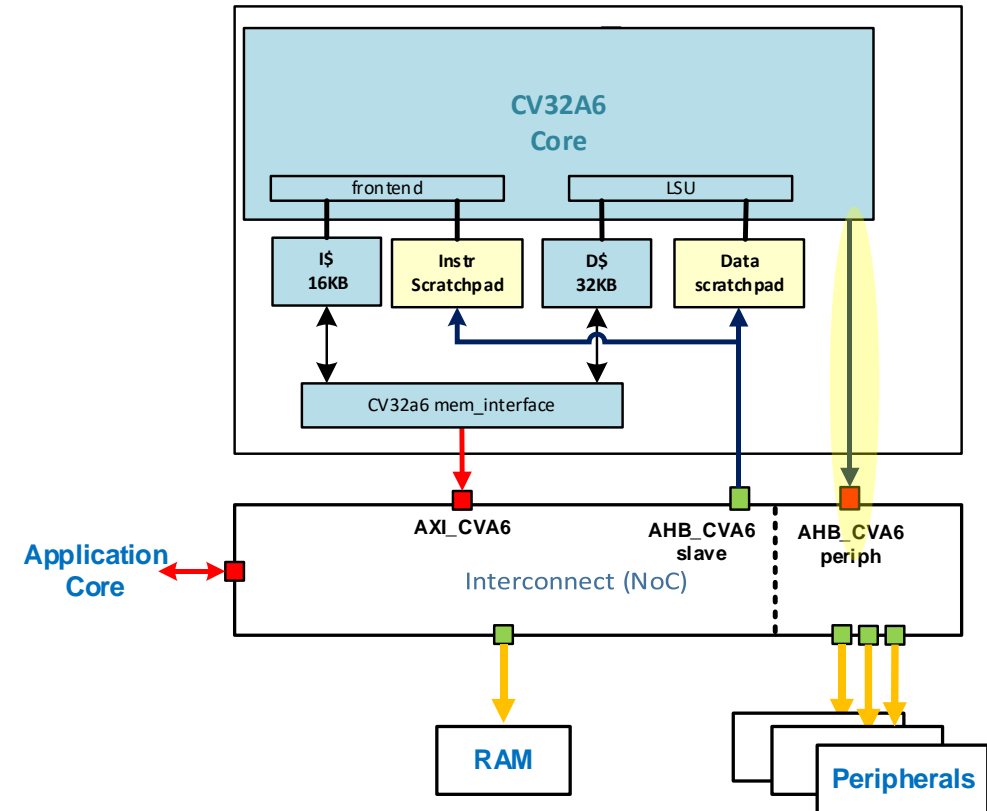
CV32A6 Core Real Time Additions : Peripheral bus

Real Time applications Needs :

- Isolate CPU Transactions categories :
 - “Data plane” (Memory) : High Bandwidth, bursts, OoO,
 - “Control plane” (Peripherals): Low Bandwidth, single transactions
- Back compatibility with other CPU cores offering this feature
 - Goal : CVA6 as a “plug and play” replacement

Solution : Additional peripheral interface (AHB)



- Global NoC simplification and area reduction,
- Functional safety simplification : split memory / peripherals protections







Resulting CVA6 Address map

Principle : keep it as simple as possible !



Instructions: Read access from:

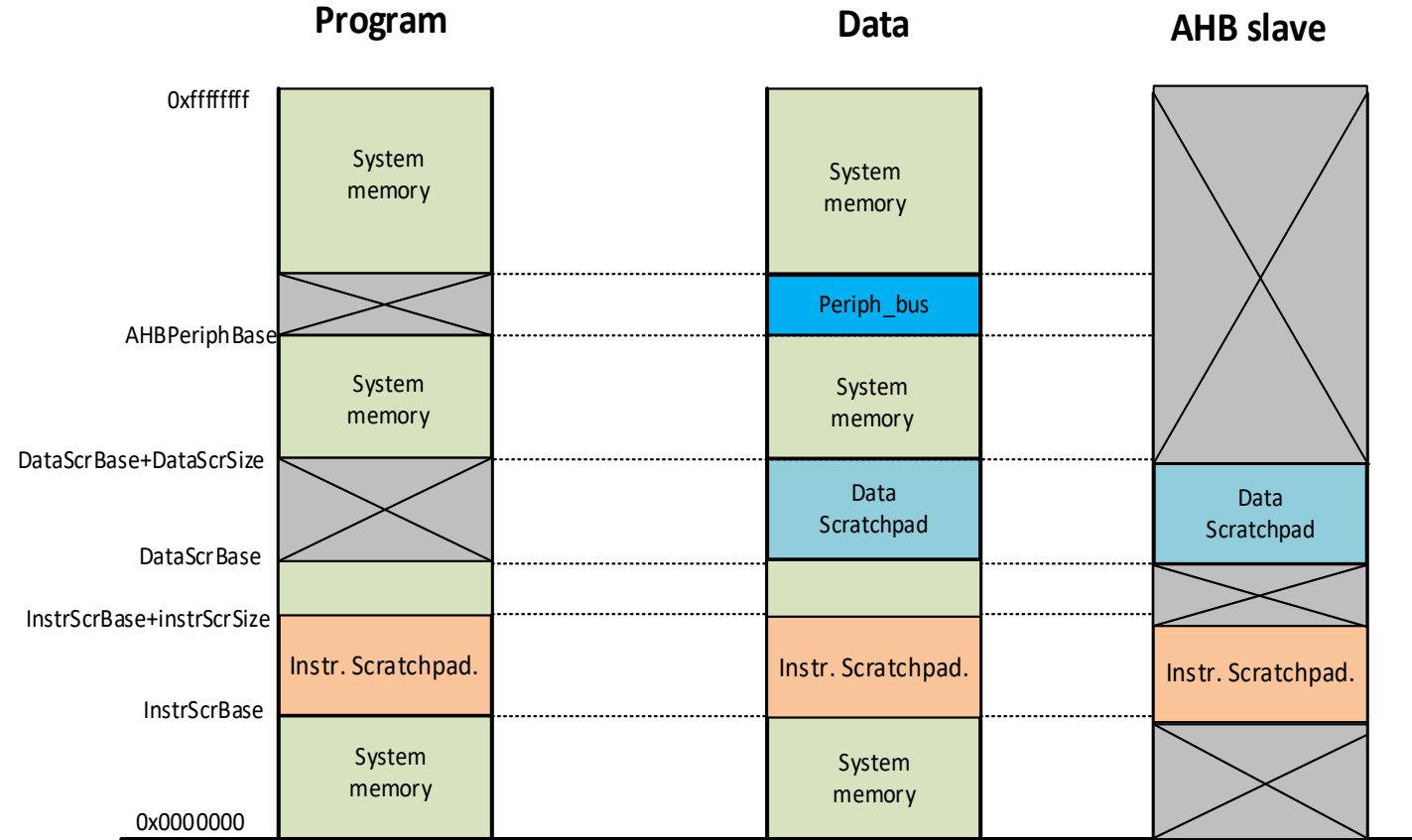
-  Main system memory
-  Instruction scratchpads

Data: Read / Writes accesses :

-  Main system memory
-  Data scratchpad
-  Instructions scratchpads
-  Peripheral bus

New external AHB interface : R/W accesses

-  Data scratchpad (preload by a host)
-  Instr. scratchpad (same use)



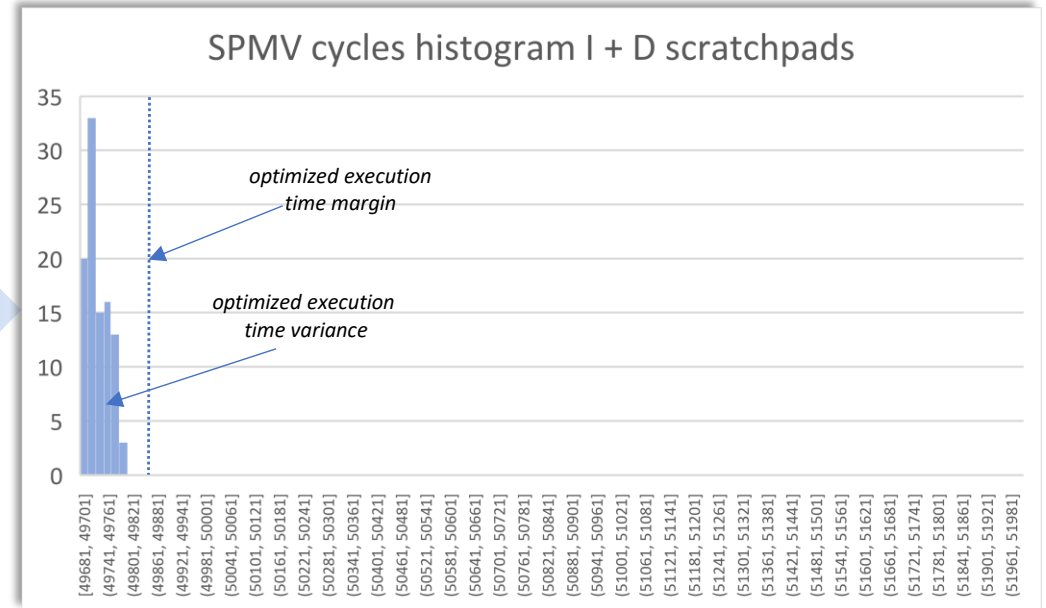
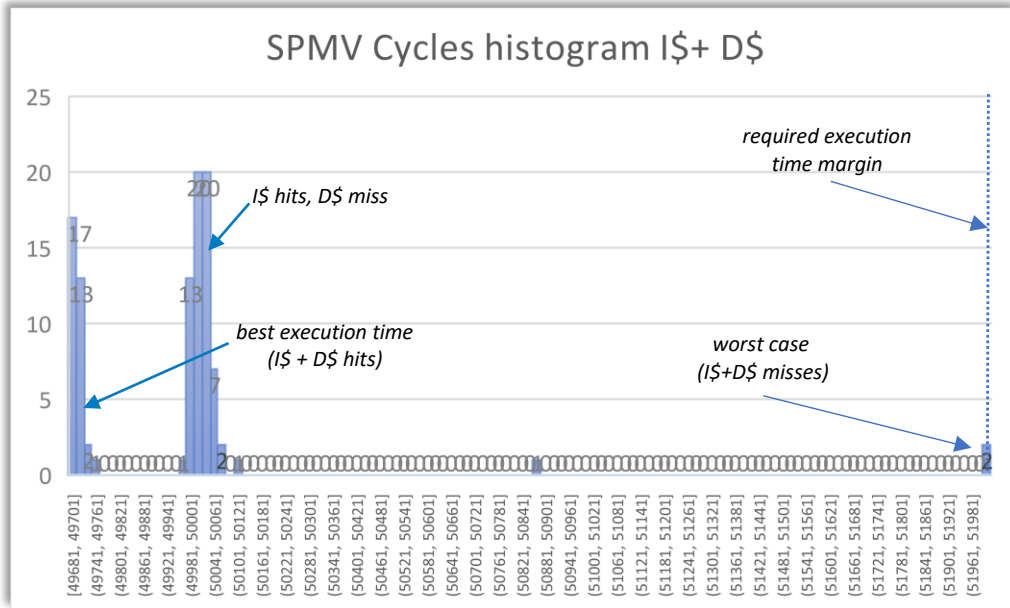
Physical addresses only :

- for latency determinism
- PMP kept for safety / debug

Early Benchmarks results

- Scratchpads: Execution determinism (SPMV benchmark) :

- I\$ + D\$: 7% execution time variance / cycle
- I + D Scratchpads : 0.3 % execution variance (remaining : branch prediction,..)

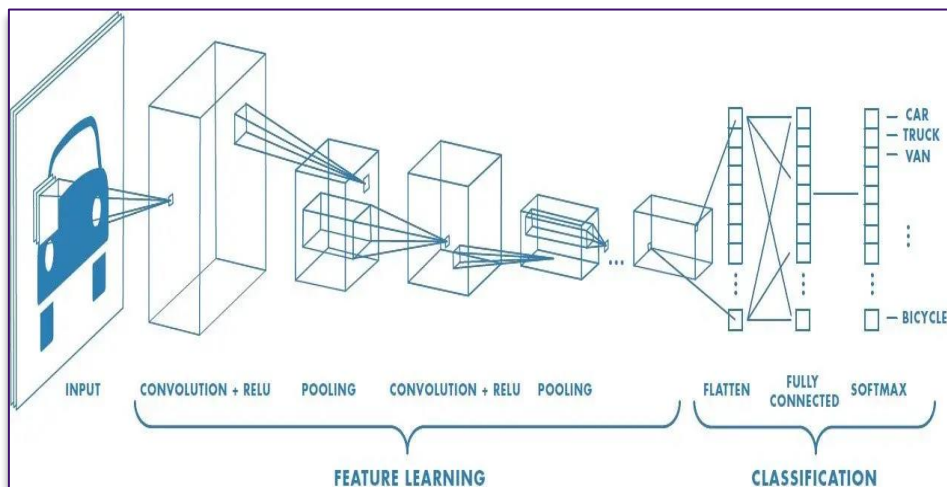


- Scratchpads : Acceleration : under development :

- Memories still running with 1 wait state (WS) , due to integration issues
- No improvement yet versus I\$ + D\$
- 0 WS achievable



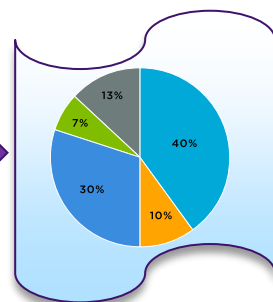
Exploiting the RISC-V customizable ISA



our CNN application

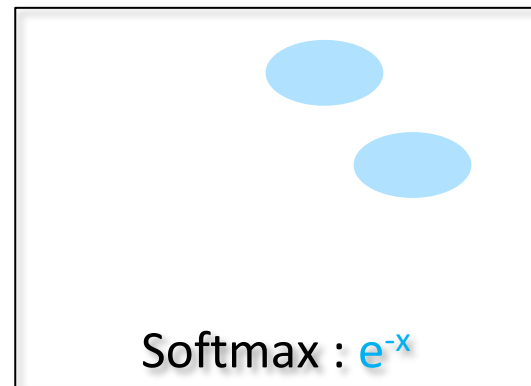
[Src1]

Profiling



Profiling report

extract most time consuming routines



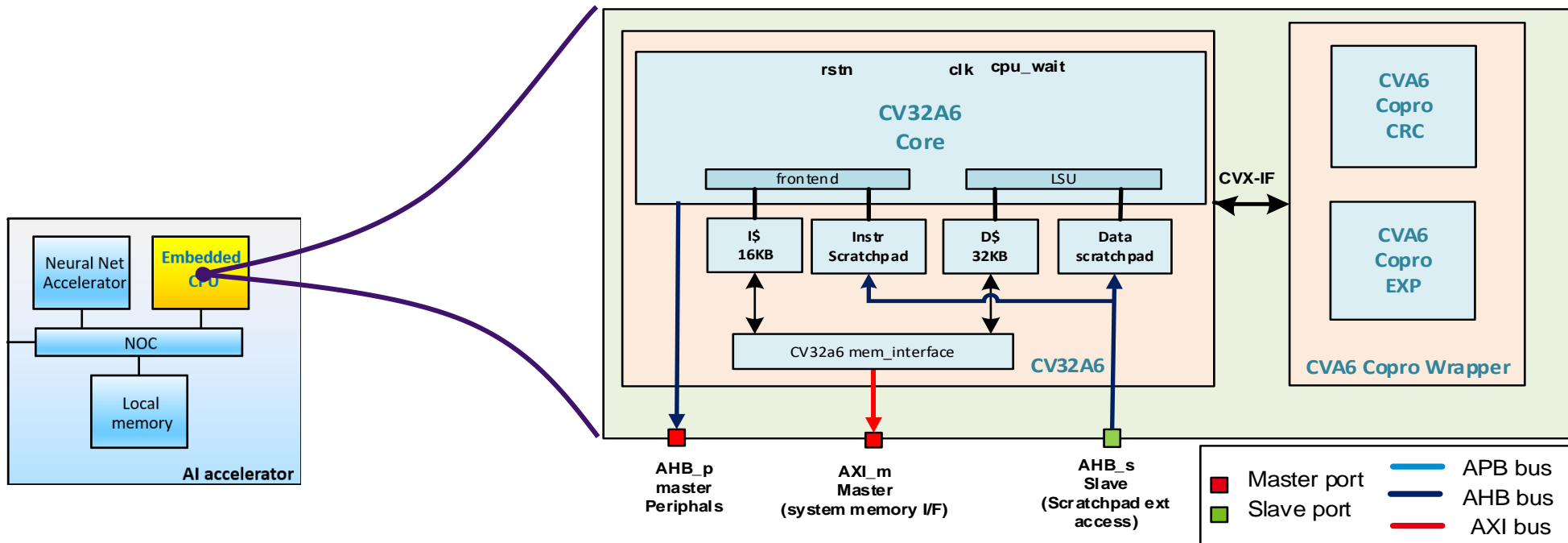
CRC codes	Generator polynomials
CRC-4	$x^4 + x^3 + x^2 + x + 1$
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-24	$x^{24} + x^{23} + x^6 + x^5 + x + 1$

CRC : inner loop

Softmax: Performed by CPU (Not NN accel) . e^{-x} is the most time costly

CRC: CNN accelerator workload description checksum (for safety)
inner loop is the most time costly when running on CVA6

Exploiting the RISC-V customizable ISA on CVA6



Custom ISA extension : CRC8, 16, 32 and EXPNEG instructions

```
CRC{8,16,32} rd, rs1, rs2,rs3
```

```
EXPNEG rd,rs1 ; rd = Exp(-rs1) fixed point
```

Implemented as CVA6 Coprocessors, using the CV-X-IF interface : (defined by OpenHWGroup)

- 2x coprocessors for **CRC** and **EXP** : standalone development, no core RTL changes required
- Custom wrapper to support 2 coprocessors on a single CV-X-IF
- Concept of a **coprocessor library**, to serve a whole range of applications



Coprocessors : Performance Predictions



- Accelerated Software Routines Examples:

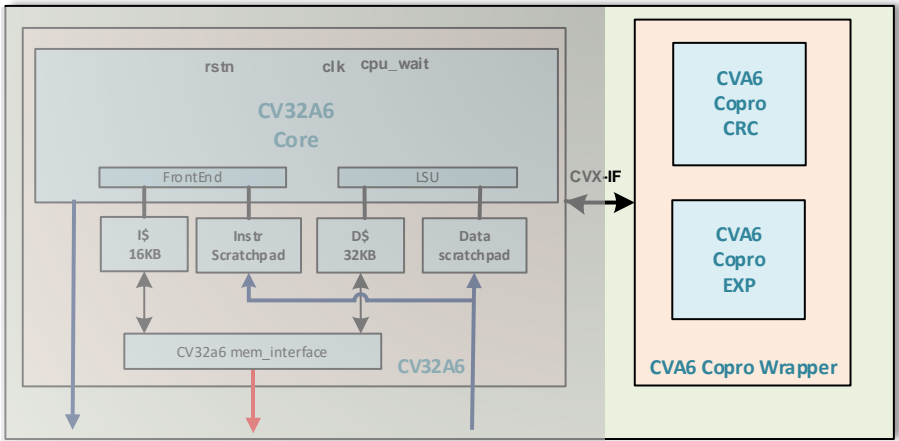
- **Functional safety** : Data block CRC computation
- **A.I (CNN)** : Softmax computation

- Evaluation using Spike simulator :

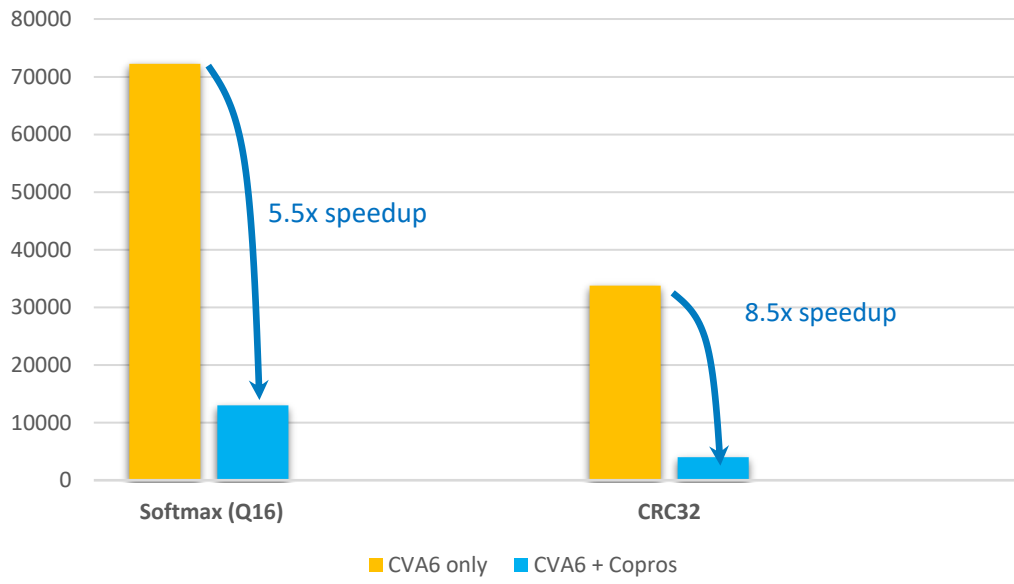
1. Baseline : a standard RISC-V processor implementation
2. With our custom instructions :
 - **CRC32**
 - fixed point **EXPNEG (Exp(-x))** in Q16 fixed point
 -

- Coprocessors are still under development

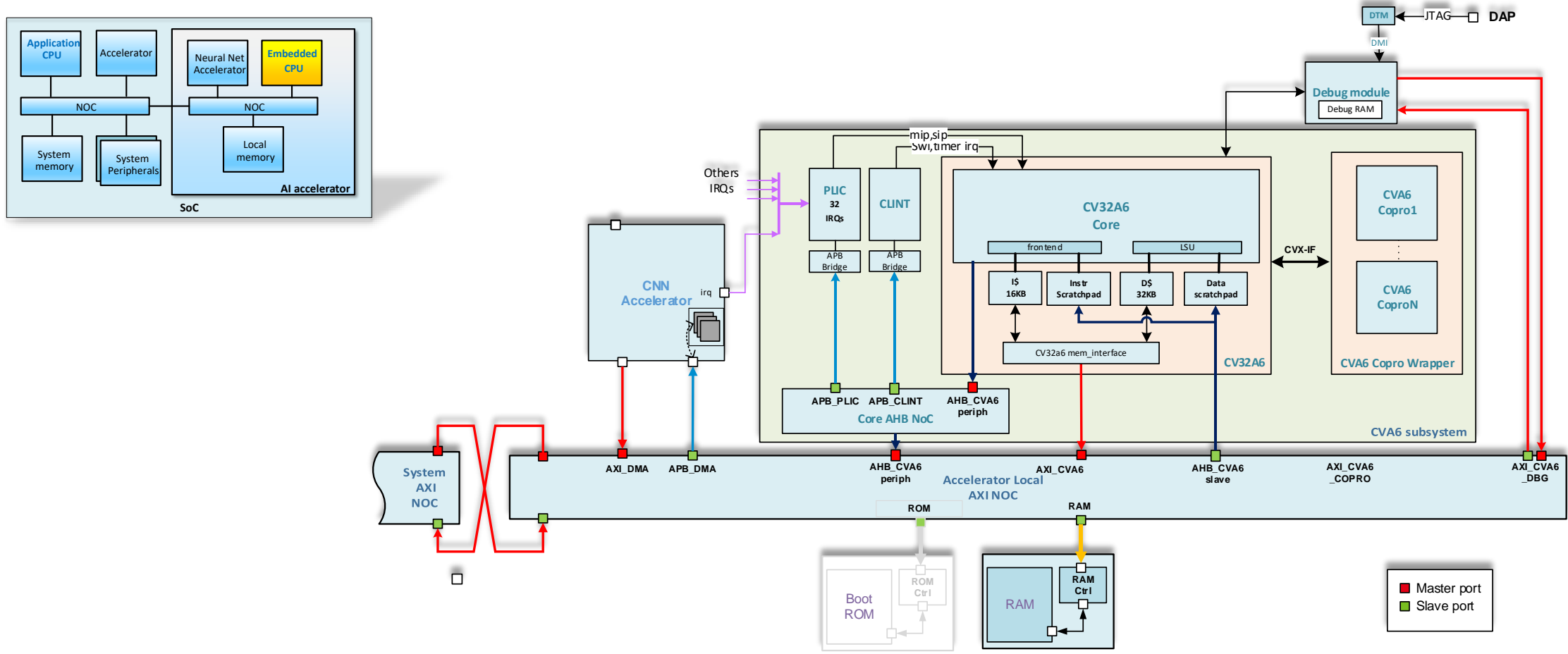
- execution speed up is fully predictable



Execution time [Cycles]



Putting it all together : TRISTAN Automotive AI Demonstrator



Acknowledgments



Together for RISC-V Technology and ApplicationS

This work is supported by the TRISTAN project funded by the Chips Joint Undertaking (Chips JU) under grant agreements 101095947

Contact : nicolas.tribie@fr.bosch.com



Thank you !

Questions ?



Peripheral Bus : Rationale



Pros :

- Transactions to/from peripherals directly routed to this bus, w/o querying the caches
 - Can be done with PMA (Physical Memory Attributes), but kept independent for greater flexibility
- Transactions to/from peripherals are **single requests** : **Simpler interconnect** (AHB instead of AXI)
- **Peripherals transfers** won't interfere with I\$ / D\$ accesses to/from the system memory
 - could be done with additional AXI IDs : LSU changes required (complex)
- For **safety critical** applications : “control plane” separated for “data plane”:
 - Possibly with different protections levels,
 - Split data / control NoCs,
 - Minimize clock domain crossings (fast memory, slow peripherals).
- **Back compatibility** with commercial CPU cores (used in our projects)



Cons:

- Requires a core interface change (+AHB master) : breaks back-compatibility (top level I/Os change),
- Internal LSU Address decoding path becomes a bit longer (possibility becoming the critical path in the core).
- Peripherals not seen from debug module master port : Use a small access routine in debug ROM (abstract function)