



IEEE IMS DL:

Instrumentation and Measurement and Autonomous Driving



Prof. Ing. Sergio Saponara
Tel. +39 050 2217 602, Fax. +39 050 2217522

sergio.saponara@unipi.it



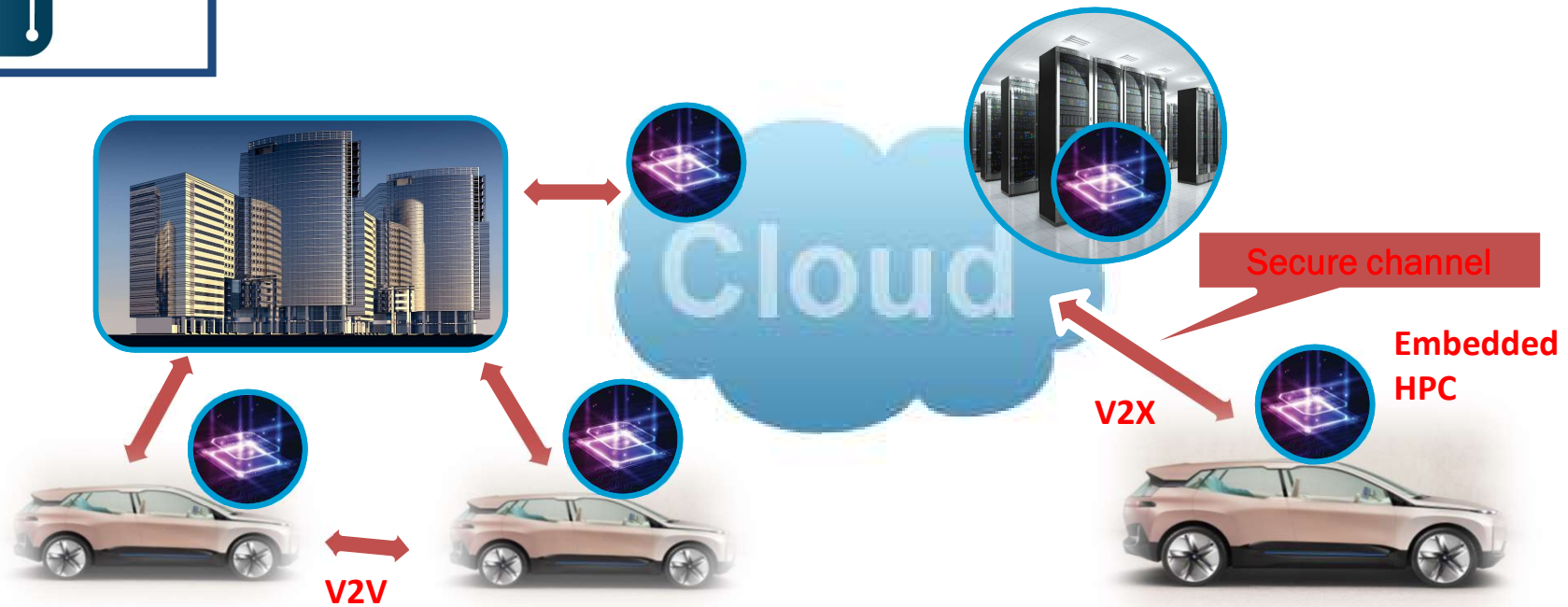
Outline

(EPI related slides, 20/80)

- Societal, economical, technical challenges of autonomous & connected vehicles and intelligent transport systems (ITS)
- Remote sensing (Radar, Lidar) in smart vehicle & ITS
- Sensing technology for navigation
- eHPC (embedded High Performance Computing) needs of autonomous and connected cars – the H2020 European Processor Initiative (EPI) project
- Arithmetic accuracy for DNN acceleration (Posits in EPI)
- Conclusions

European Processor Initiative

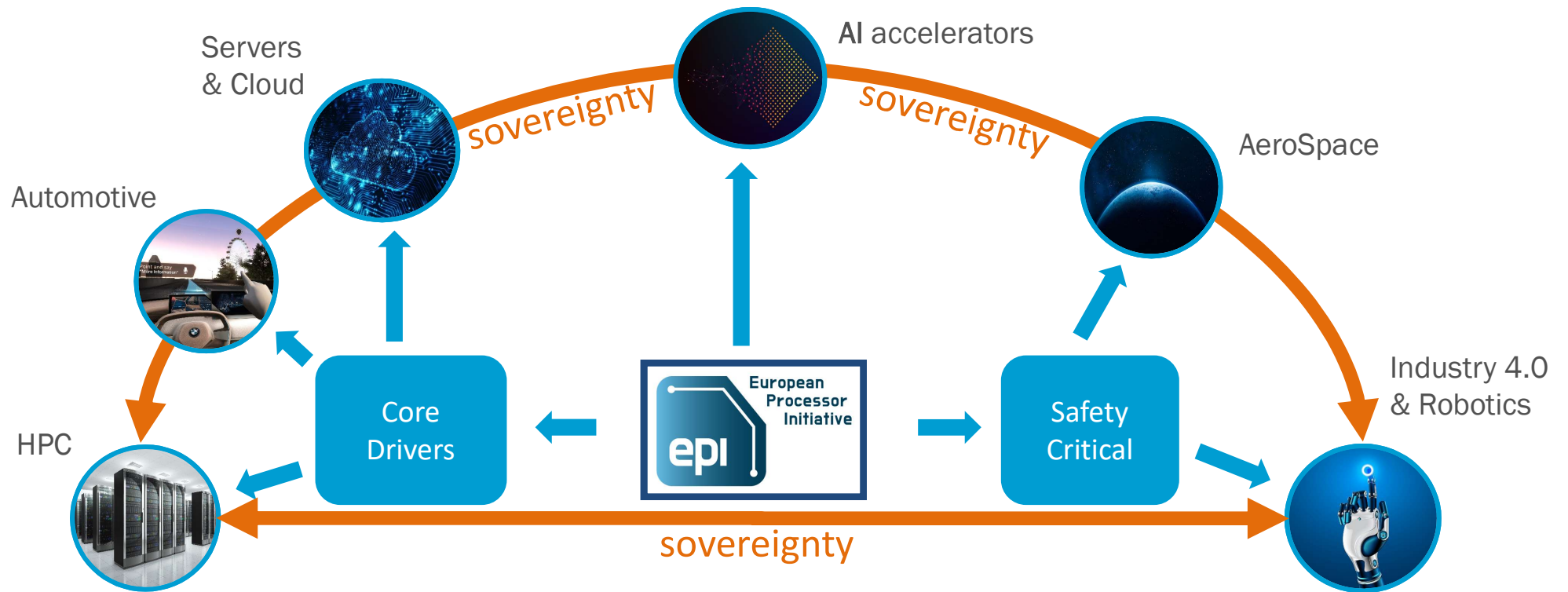
Enabling **TE**chnologies for sm**A**rt vehicles and **M**obility (EPI 120 M€ project in 5 years)



Copyright © European Processor Initiative 2019.










































ACES Vehicles & Mobility

Autonomous Connected Electrified Shared



Copyright © European Processor Initiative 2019.

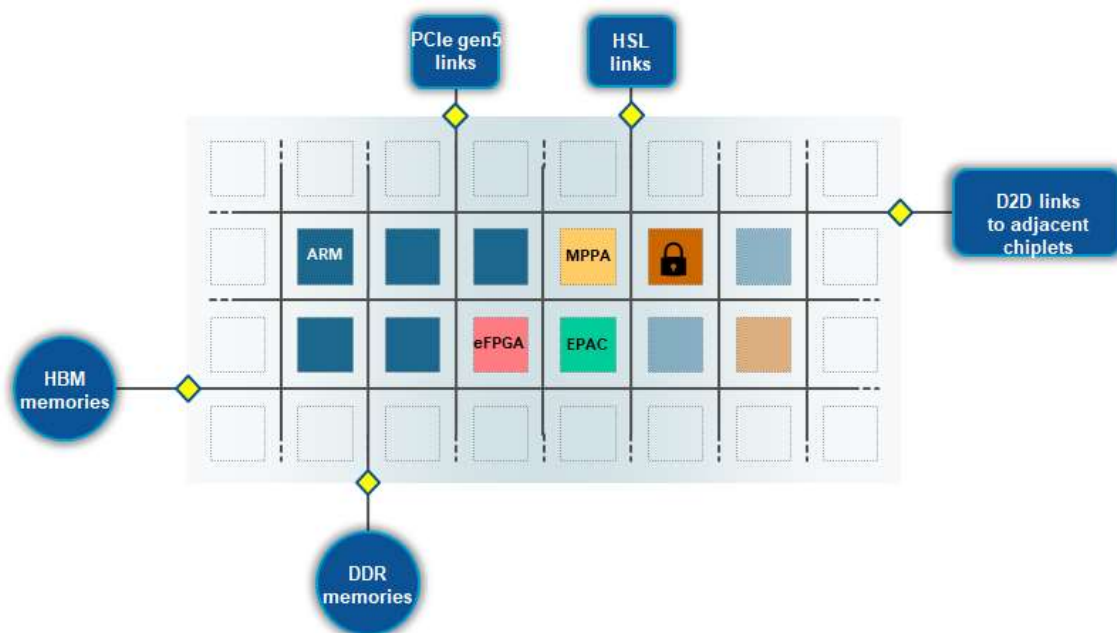
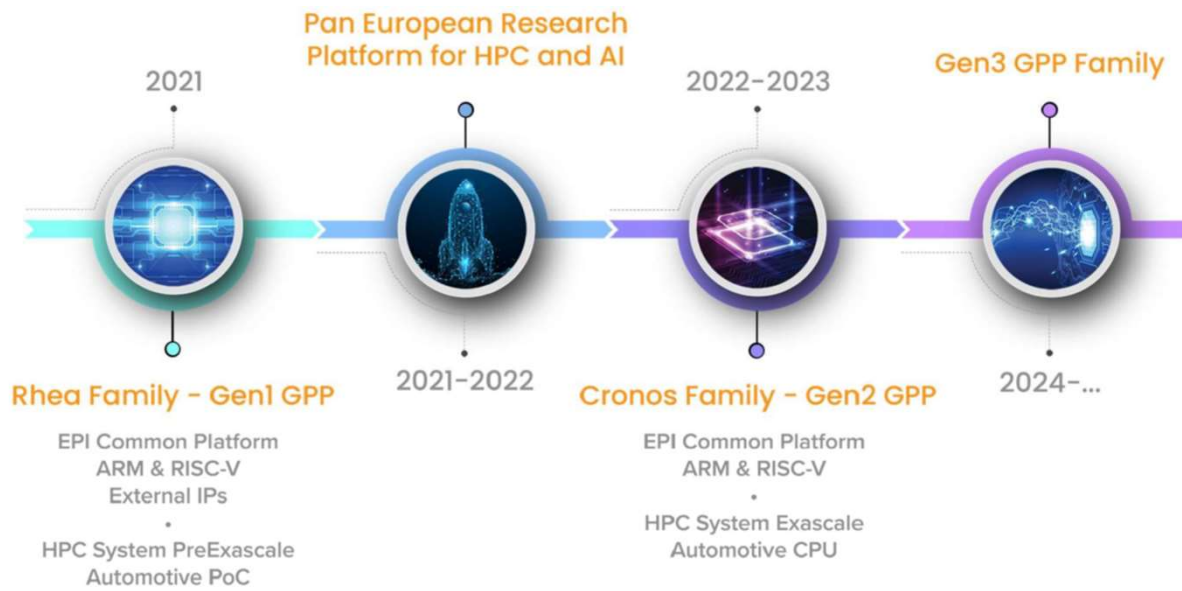
EPI partners & HW/SW eco-system

   <p>Full HPC Environment for the Reference Platform</p>  	    <p>Co-design exploration space</p>    	   <p>Automotive eHPC software support</p>    	 <p>Programming tools & Libraries: LLVM/GCC with OpenMP; OpenMPI; FFTW; BLIS; OpenBLAS, ...</p>   
<p>Security, Low-level software, power management</p>  	   <p>Linux Operating System</p>    		
    	<p>EPI Processor</p>	<p>EPI Reference Hardware</p>   	

EPI
27 PARTNERS



EPI Roadmap & Architecture

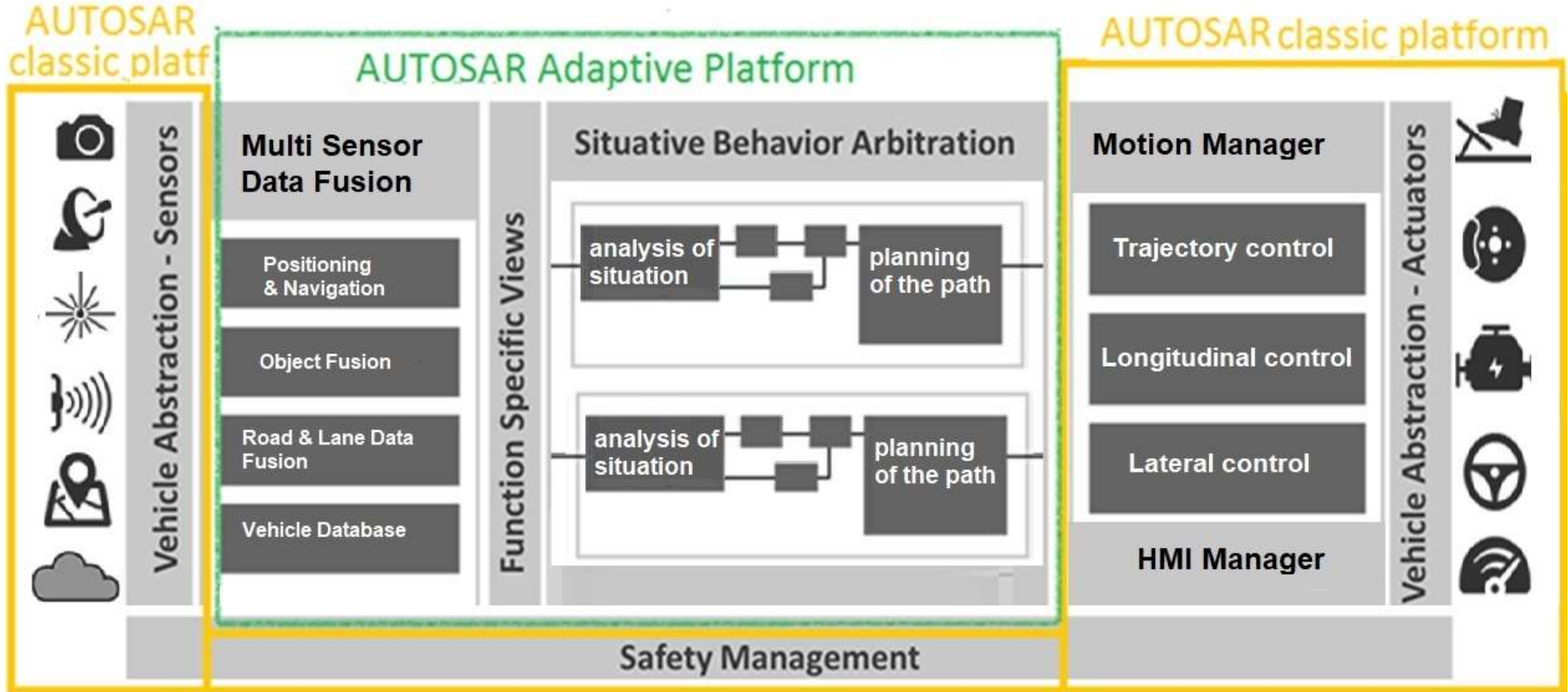


- EPAC - EPI Accelerator  
- MPPA - Multi-Purpose Processing Array
- eFPGA - embedded FPGA 
- Cryptographic ASIC (EU Sovereignty) 

EPI chip in 6 nm technology



EPI enables AUTOSAR adaptive platform



New eHPC ECU: Safe&secure MCU with high-SIL controlling EPI-like number crunchers (multi-core 64b GPP + accelerators)

Memory needs for autonomous cars

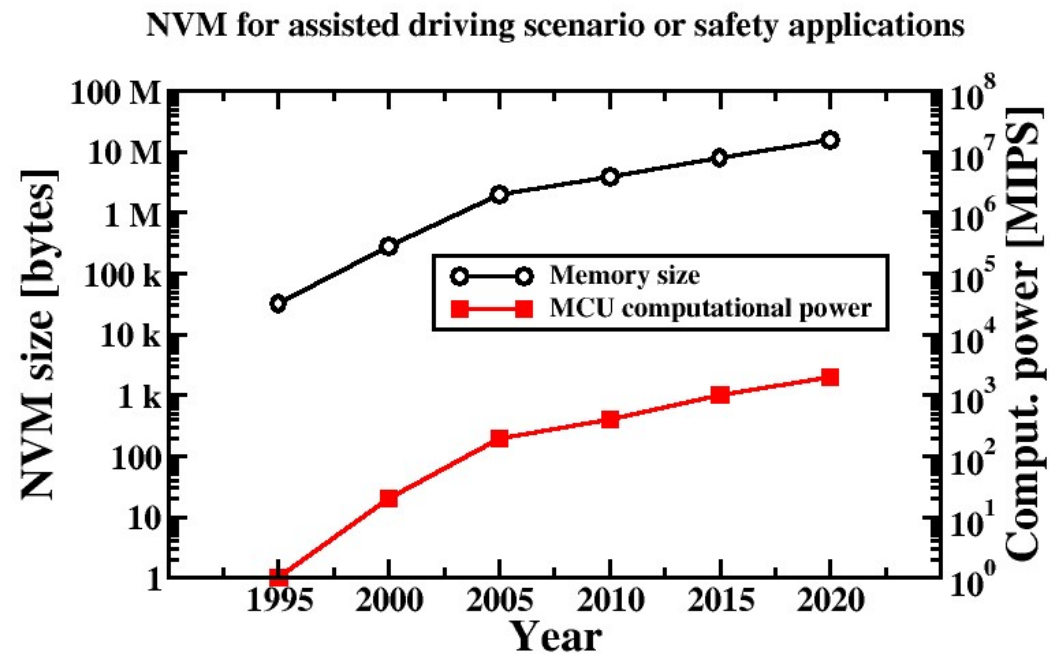
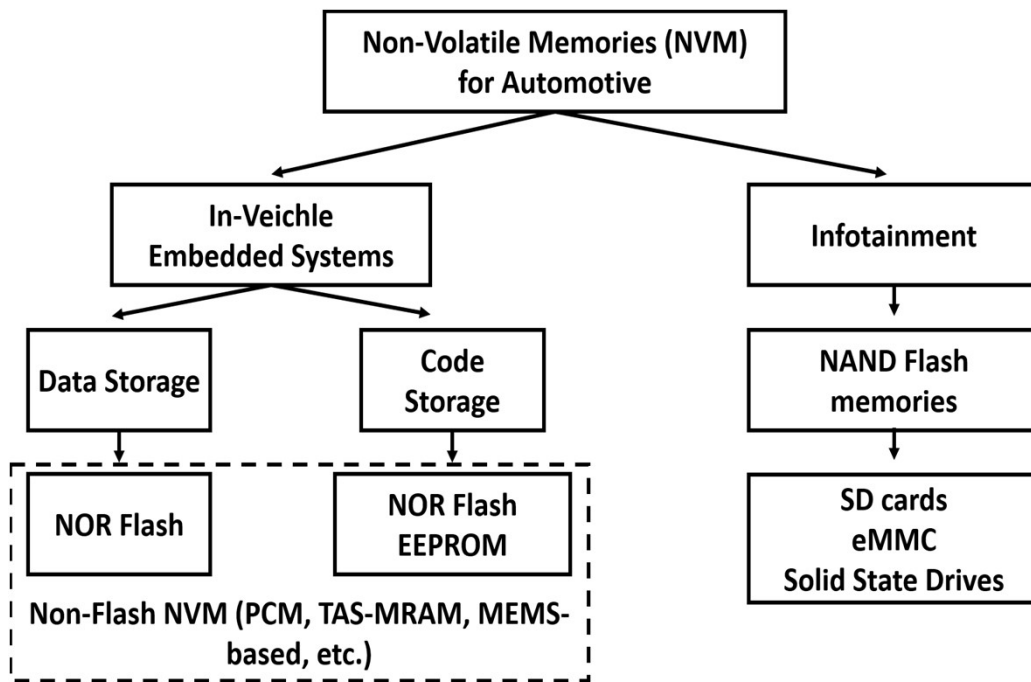
AVG INTERNET USER	~1.5 GB OF TRAFFIC PER DAY
SMART HOSPITAL	3,000 GB PER DAY
AUTONOMOUS VEHICLES	4,000 GB PER DAY... EACH
AIRPLANE DATA	40,000 GB PER DAY
SMART FACTORY	1,000,000 GB PER DAY

TECHNICAL
DATA

SOCIETAL &
CROWDSOURCED

PERSONAL
DATA

Memory needs and trends for assisted driving



Parameter	EEPROM	NOR Flash Code Storage	NOR Flash Data Storage	PCM	MEMS-based	RRAM CBRAM	TAS-MRAM
Endurance	500k	10k – 100k	500k – 1M	>1M	>1M	100k	>1M
Data Retention	>10 yrs/125 °C	10 yrs/125 °C	>10 yrs/125 °C	10 yrs/85 °C	>10 yrs/125°C	10 yrs/85 °C	>10 yrs/125 °C
Power consumption	Low	Low	Low	High (Write)	Low	Low	High
Read Latency	20 – 50 ns	< 20 ns	< 20 ns	> 20 ns	> 100 ns	> 20 ns	50 – 100 ns
Cost per bit	Medium/High	Medium	Medium	Low	High	Low	High

Outline

- Societal, economical, technical challenges of autonomous & connected vehicles and intelligent transport systems (ITS)
- Remote sensing (Radar, Lidar) in smart vehicle & ITS
- Sensing technology for navigation
- eHPC (embedded High Performance Computing) needs of autonomous and connected cars – the H2020 European Processor Initiative (EPI) project
- Arithmetic accuracy for DNN acceleration (Posits in EPI)
- Conclusions

Motivations for alternatives to float in ML & DNN

- In Automotive Applications, Machine Learning (ML) and Deep Neural Networks (DNNs) must run in vehicle, relying on internet connection and remote services can not be mandatory
- we need both HPC on-board the vehicle, and/or more efficient representation of the information
- The representation chosen for real numbers has a high impact on the synthesized hardware (cores, SoC accelerators, etc.)
- Novel **posit** format as alternative to float (posit library developed in Pisa: the cppPosit library)
- Floating-point representation (IEEE-754) has some limitations:
 - The support to unnormalized numbers is tricky (needs more HW)
 - Too many representations wasted for Not-A-Number
 - Uses the same number of bits for the mantissa, both for small and large numbers (and this is inefficient)

Computing Industry Looking for Alternatives

- Intel/Google BFLOAT16 (equivalent to a standard single-precision floating-point value with a truncated mantissa field). Basically, they are less precise than fp16, but with a range similar to fp32. Supported in Google cloud TPU and TensorFlow and Intel AI processors
- Intel flexpoint (16bits size aiming at equivalent fp32 accuracy)
- NVIDIA (e.g. concurrent execution of Floating Point and Integer Instructions in the new Turing SM; from Fp32/Fp16 and INT32 to INT8 and INT4 precision modes for inferencing workloads that can tolerate quantization)
- Tesla FSD chip (Neural processing units use 8-bit by 8-bit integer multiply and a 32-bit integer addition)
- Transprecision computing proposed in state of art (e.g. Greenwaves)

The Novel Posit Format

- Proposed by John Gustafson in 2017
- It can be viewed as a compressed floating-point format, which deserves more mantissa bits for low number and less for large numbers, within a fixed-length format
- No-need to use un-normalized floats (so, no extra-hardware wasted to handle this exception)
- Only one representation wasted for Not-A-Real (NAR)
- Posit numbers use an interesting encoding which allows, to compare two posits, to reuse the same circuit used to compare two integers in 2's complement already present in the ALU

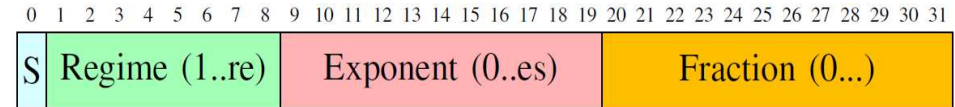


Fig. 1. An example of Posit data type.

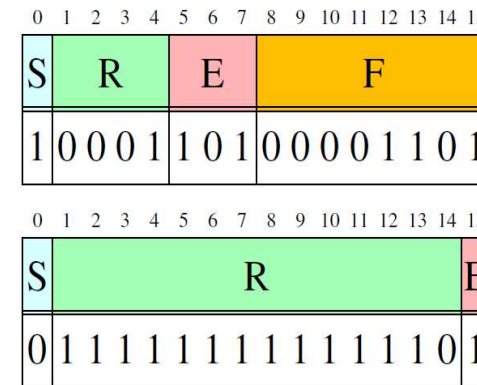
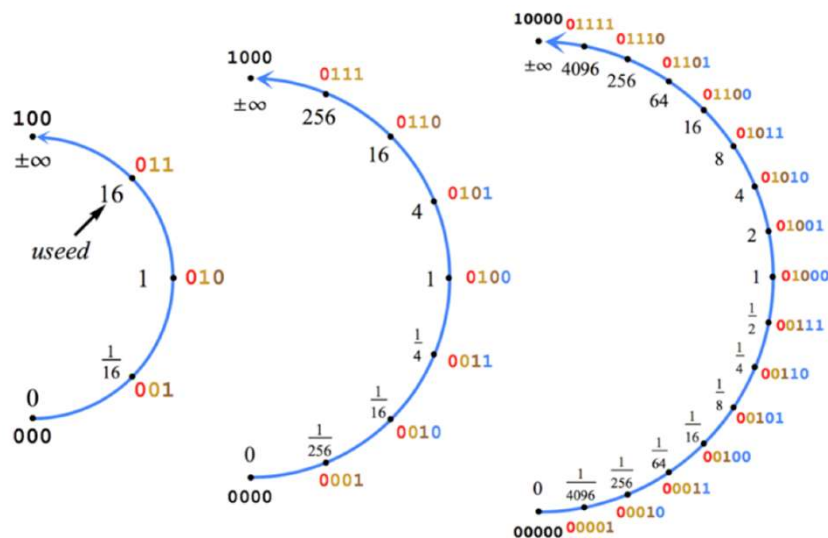


Fig. 2. Two examples of 16-bit Posit with 3 bits for exponent (es=3). In the upper the numerical value is: $-256^{-3} \cdot 2^5 \cdot (1 + 13/256)$ (13/256 is the value of the fraction, $1 + 13/256$ is the value of the mantissa). The final value is therefore $-1.907348 \times 10^{-6} \cdot (1 + 13/256) \cong -2.0042 \times 10^{-6}$. In the lower the numerical value is: $+256^{+12} \cdot 2^4 \cdot (1 + 0)$ (since the fractional part of the mantissa is missing, we set it to zero). The final value is therefore $2^{16} \cdot 2^4 \cong 1.2676506 \times 10^{30}$. The second example allows to clarify that: i) the fractional part can be missing, ii) the exponent field can be shorter than its maximum size (in that case the missing bits are assumed zero: the exponent 4 comes from the reconstructed exponent field 100).



The `cpp-Posit` Library developed in Pisa

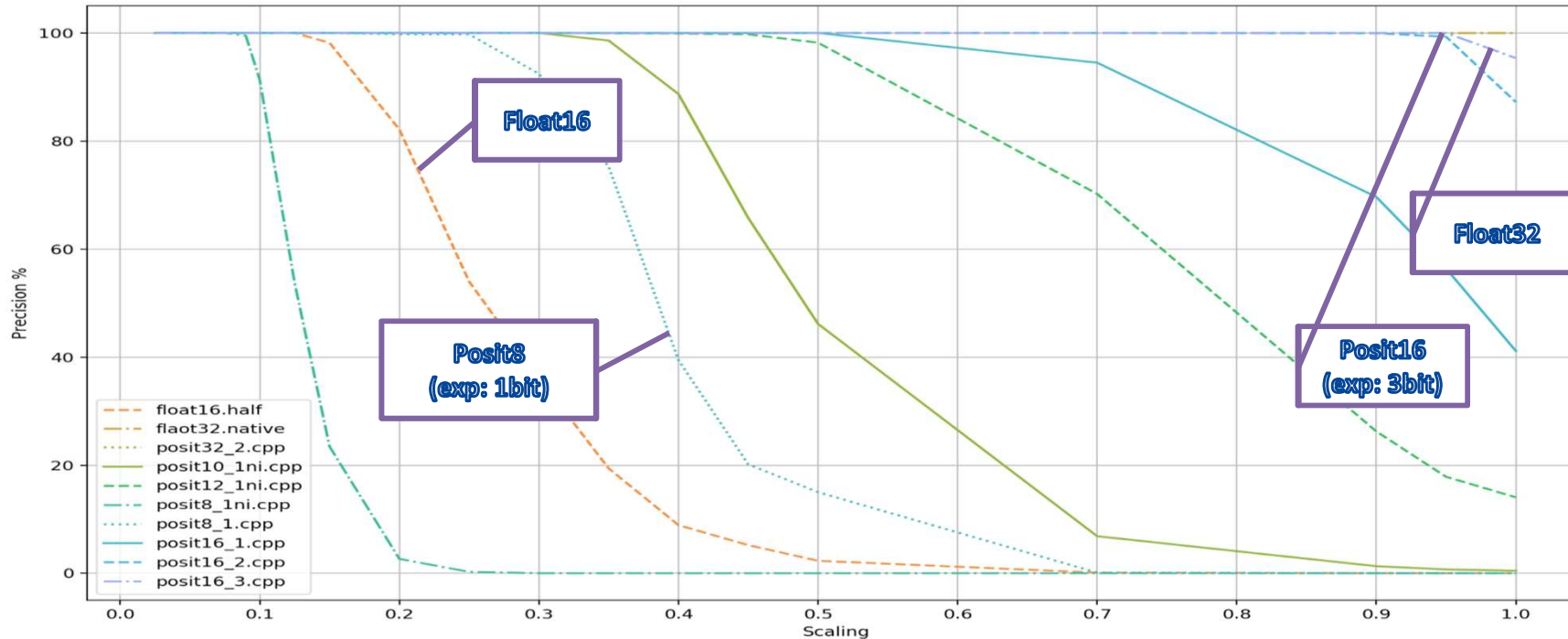
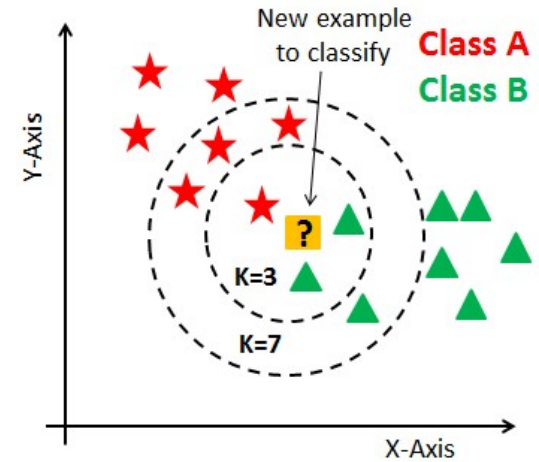
- State-Of-The-Art Posit library, developed in Pisa
- Very efficient (written in C++, fully exploiting templates and several features of the C++14 standard)
- Emulates a Posit Processing Unit (PPU) using, either
 - The FPU and the ALU, or
 - The ALU alone (the FPU is emulated using softfloat)
- Supports **TABULATED POSITS** (using look-up-tables, for posit having total length ≤ 14 bit): this speedup the library, a mandatory feature to train DNNs
- Next goals (ongoing activities):
 - Exact Dot Product: **main goal 1**
 - High Level Synthesis in FPGA/ SoC Accelerator: **main goal 2**

Are Posits Really Better Than Floats?

- Yes!
- UNIPi has performed comparisons on both Machine Learning (K-NN) and Deep Neural Networks for Image Classification (we extended the tiny-DNN C++ library)
- We have found that, on a K-NN application (see next slide):
 - a 16-bit posit is as accurate as a 32-bit float (*single precision*)
 - an 8-bit posit is much better than a 16-bit float (*half precision*).
- On an DNN used for image classification:
 - a 10-bit posit is as accurate as a 32-bit float (>98.5% of correct classification)
 - a 8-bit posit is able to provide a very high accuracy (>97%)
- Both cppPosit-based K-NN and tiny-DNN libraries have been selected as WP1 benchmark applications (they support both floats and posits)

The Cpp-Posit based K-NN Library

- The K-NN algorithm searches for the K points in a dataset that are the closest to a given query point
- It can be computed in an exact or approximated manner.
 - Implemented the approximated NN, using floats and posits
 - Compared the two formats on two standard benchmarks: Fashion Mnist 784 Euclidean & SIFT-128-Euclidean



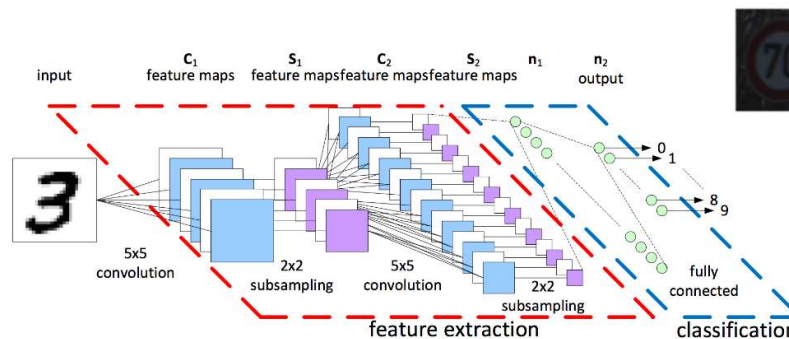
The scaling factor rescales the dynamic range of the original dataset, without affecting relative dynamic. Scale 1.0: original dataset. For a given scaling factor, the higher the precision, the better

Experiments with Deep-Neural Networks

- We integrated the cppPosit library with tiny-DNN open source C++ lib
- A posit12 DNN reaches the same accuracy of the float32 counterpart
- To speedup the learning phase, we tabulated the posits (LUT)
- Acceptable performance can even be attained using an 8-bit

Data Type (tot_bits, exp_bits)	Accuracy on 10,000 images
Float32	98,88%
Posit16,2	98,88%
Posit14,2	98,85%
Posit12,2	98,66%
Posit10,0	98,69%
Posit8,0	97,24%

Type	Accuracy
Float32	94.0%
Posit16,0	94.0%
Posit14,0	94.0%
Posit12,0	94.0%
Posit10,0	94.0%
Posit8,0	93.8%



- **MNIST** dataset: 10 classes, 10,000 samples
- Convolutional Neural Network
- Similar results obtained on **CIFAR10**.
- Currently investigating the **ImageNet** dataset, using the AlexNet pre-trained network

Posit Processing Unit (PPU) vs FPU

- Lower memory footprint (on RAM, on disk)
- Higher bandwidth & lower power consumption
- More cache-friendly (due to the use of shorter data)
- More suited for vectorization (again, shorter data means more data on registers at the same time – see ARM SVE)

A Posit Processing Unit (PPU) can be synthesised e.g. using the Vivado toolkit: the cppPosit library allows automatic HDL code generation starting from C++ code

An alternative is a LUT tabulated implementation of a PPU, for posits with max 8/10 b

**Memory needs to store the single
LUT as a function of X
(total number of bits of the Posit)**

Total bits (X)	Storage type bits (b)	Per-table occupation
8	8	64KB
10	16	2MB
12	16	32MB
14	16	512MB
16	16	8GB

Posits for DNN/ML: conclusions

- Posits have the potential to overcome most of the float issues in Machine Learning and DNN computing
- They allow to reduce the bandwidth bottleneck problem during read/write from/to RAM
- Have beneficial effects on vectorizable applications, since data are generally shorter
- They are more cache friendly, every time a posit8 can replace a float16, a posit16 a float32 and a posit32 a float64 (i.e., in most of the applications)
- A posit library developed at UniPI (cppPosit)
- Tested on K-NN and DNN benchmarks
- Activity ongoing:
 - Test on additional datasets/applications
 - Recompile on ARM-64 SVE simulator
 - Software implementation of the Exact Dot Product
 - Hardware PPU by high level synthesis

Thanks for your attention



Prof. Ing. Sergio Saponara
Tel./Fax +39 050 2217602 /522



sergio.saponara@unipi.it