DRDT 7.2: Front-end programmability

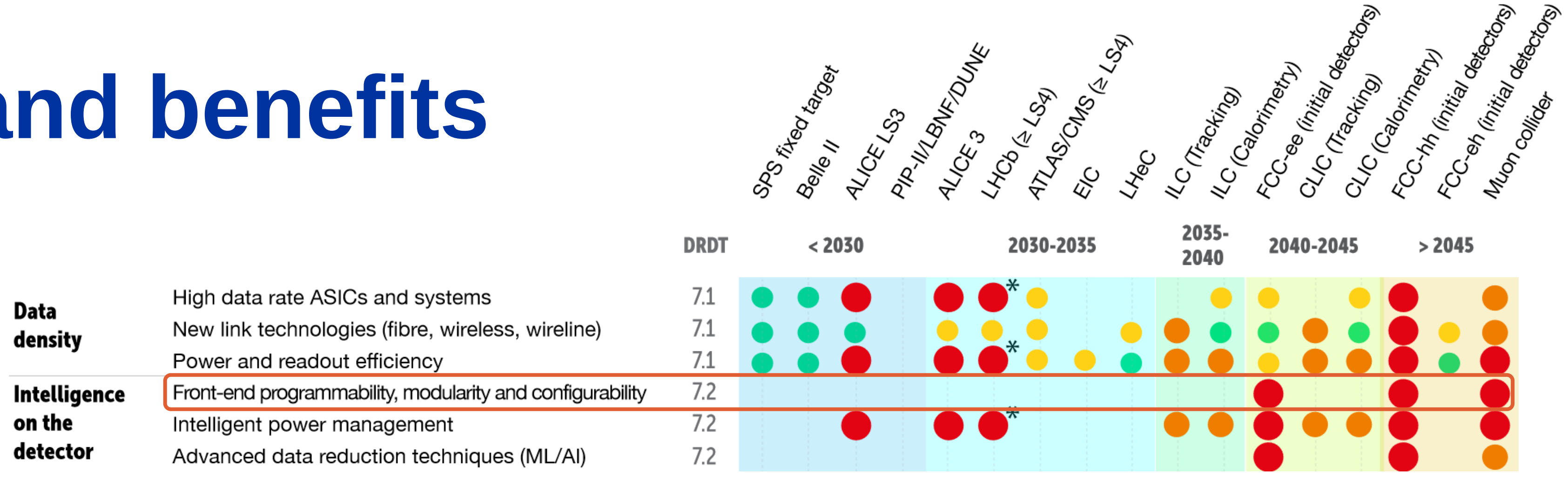
Marco Andorno (marco.andorno@cern.ch)

2023/03/14

Outline

Introduction

Motivations and benefits



		DRDT	< 2030				2030-2035					2035-2040		2040-2045			> 2045		
			SPS fixed target	Belle II	ALICE LS3	PIP-II/LBNF/DUNE	ALICE 3	LHCb (≥ LS4)	ATLAS/CMS (≥ LS4)	EIC	LHeC	ILC (Tracking)	ILC (Calorimetry)	FCC-ee (initial detectors)	CLIC (Tracking)	CLIC (Calorimetry)	FCC-hh (initial detectors)	FCC-eh (initial detectors)	Muon collider
Data density	High data rate ASICs and systems	7.1	🟢	🟢	🔴		🔴	🔴*	🟡				🟡	🟡		🟡	🔴		🟠
	New link technologies (fibre, wireless, wireline)	7.1	🟢	🟢	🟢		🟡	🟡	🟡		🟡	🟠	🟢	🟢	🟠	🟢	🔴	🟡	🟠
	Power and readout efficiency	7.1	🟢	🟢	🔴		🔴	🔴*	🟡	🟡	🟢	🟠	🟠	🟡	🟠	🟠	🔴	🟢	🔴
Intelligence on the detector	Front-end programmability, modularity and configurability	7.2												🔴			🔴		🔴
	Intelligent power management	7.2			🔴		🔴	🔴*				🟠	🟠	🔴	🟠	🟠	🔴		🔴
	Advanced data reduction techniques (ML/AI)	7.2												🔴			🔴		🟠

The increase in **design complexity** and **cost** in advanced nodes calls for a more efficient resource utilization and an abstract design methodology that employs programmability and modularity.

The benefits are:

- Smaller number of more flexible and capable ASICs.
- Data reduction
- Power reduction

Implementing programmability

Three main ingredients:

Software programmability

- Allows retargeting an ASIC for a different application
- Can change the algorithm within the same application

Standard interconnect bus

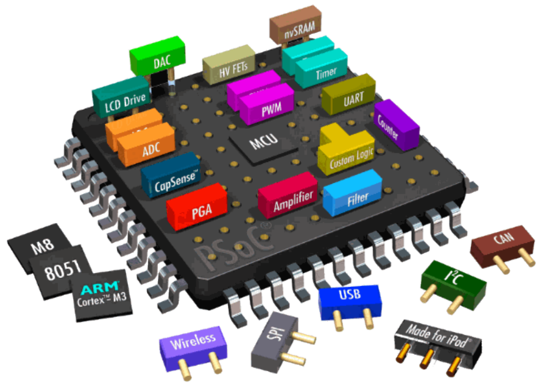
- Promotes collaborative work
- Ensures compatibility of blocks designed by different teams

IP blocks library

- Enables modularity with self-contained building blocks
- Helps design reusability

An SoC platform provides all this, leading to

- Shift from design for an application to **design for resources**
- **Faster turnaround time**, both for design and verification



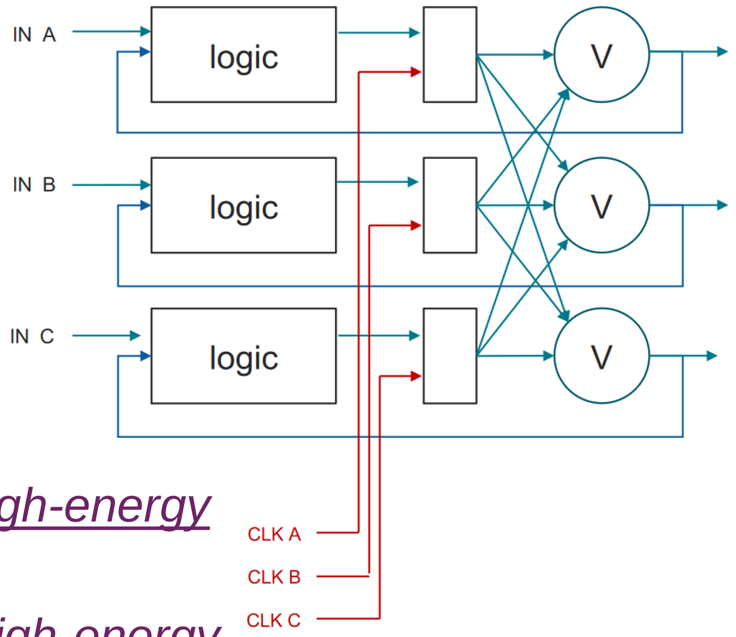
Main challenge: radiation tolerance

Processor core (1/3)

The most robust approach to harden a processor is **Triple Modular Redundancy (TMR)**: combinational logic, registers and clocks are triplicated and majority voted at the output.

- 👍 Simple to understand
- 👍 Maximum SEU protection
- 👍 Can be automatized

- 👎 Very large area and power overhead (> 3x)
- 👎 Can be difficult to implement correctly (depending on RTL and physical constraints)



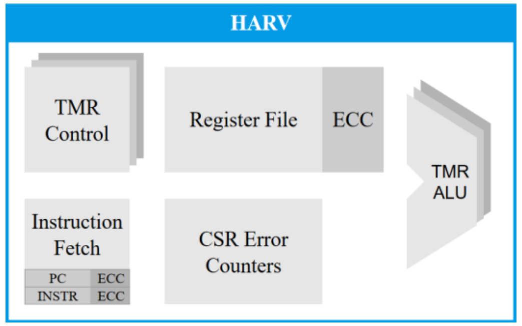
Examples of works using TMR:

- A. Walsemann et al., *STRV — a radiation hard RISC-V microprocessor for high-energy physics applications*, 2023, JINST 18
- M. Andorno et al., *Rad-hard RISC-V SoC and ASIP ecosystems studies for high-energy physics applications*, 2023, JINST 18
- A. E. Wilson and M. Wirthlin, *Neutron Radiation Testing of Fault Tolerant RISC-V Soft Processor on Xilinx SRAM-based FPGAs*, 2019, IEEE SCC

Processor core (2/3)

Full TMR is not the only option:

- D. A. Santos et al., *Neutron Irradiation Testing and Analysis of a Fault-Tolerant RISC-V System-on-Chip*, 2022, IEEE DFT

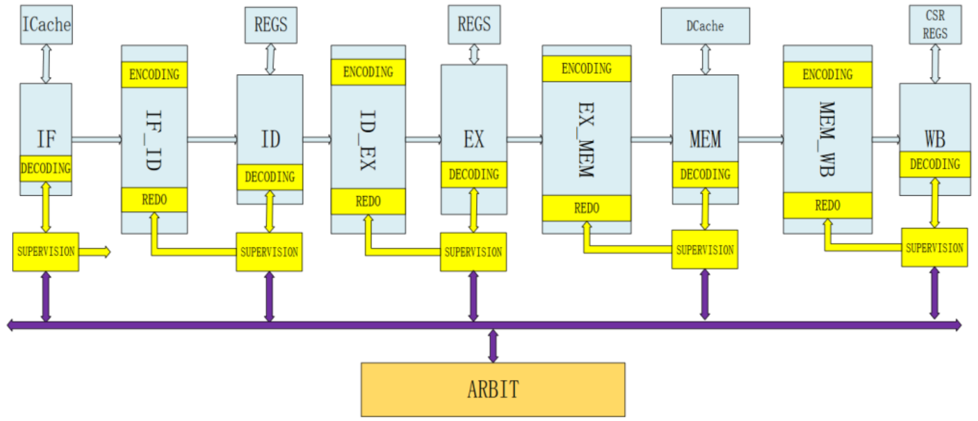


- Selective triplication
- ECC on registers
- Watchdog for critical failures

👍 Less area overhead
👎 Can accumulate errors

- J. Li et al., *DuckCore: A Fault-Tolerant Processor Core Architecture Based on the RISC-V ISA*, 2021, Electronics vol. 11

- Pipeline register encoding/decoding
- Pipeline rollback in case of error
- An arbiter decides the rollback strategy

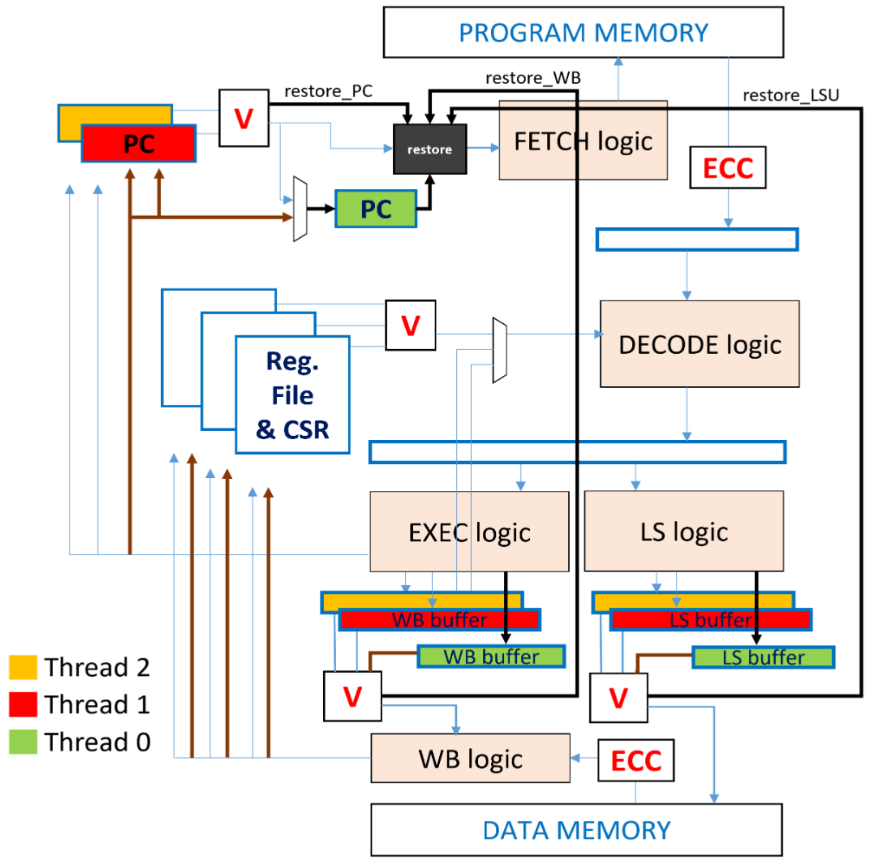


👍 Small area overhead

👎 Complex

Processor core (3/3)

- M. Barbirotta, *Evaluation of Dynamic Triple Modular Redundancy in an Interleaved-Multi-Threading RISC-V Core*, 2022, Journal of Low Power Electronics and Applications



- Dynamic TMR: interleaved multi-threading (temporal + spatial redundancy) with voters
- Third thread loaded in case of error to repeat execution of a stage

👍 Smaller area and power footprint w.r.t TMR

👍 Can be generalized to more complex architectures

👎 Complex to implement

👎 Works well only for low SEU rate (e.g. in space)

Interconnect bus

- The need is to prevent SETs (Single Event Transients) from being sampled by the registers.
- The decision to harden or not the interconnect depends on the clock frequency and the expected SET rate.
- Little to no literature on the topic, space probably doesn't need this. Are high-energy physics requirements unique in this sense?

Possible approaches:

Full TMR
Still possible and used.

👍 Simple to implement

👎 The number of wires could lead to unroutable designs

Encoding
Commonly used to reduce the number of wires.

👍 Significant routing resource saving (46% less wires for each 32-bit bus with Hamming(13,8) as in our work with APB-RT)

👎 Area penalty for encoders/decoders

Memories (1/2)

- “Brute force” approach: make a **flip-flop memory with full TMR**.
- Error correction comes automatically with the voted output.
- If the voters provide an “error” signal, the memory can be clock gated.

👍 Most protection

👍 Simple to implement

👍 Technology independent

👎 Very large area overhead (~250k gates for 2 kB memory, 32-bit word voting) mainly due to voters and error signal generation (latches wouldn’t help)

➡ Suitable only for small configuration or instruction memories.

Memories (2/2)

- Another option is to use an SRAM block and harden it with either:

Triplication
A. Walsemann et al.

👍 Best protection also against multi-bit upsets

👎 More than 3x area penalty

Encoding
R. C. Goerl, An efficient EDAC approach for handling multiple bit upsets in memory array, 2018, Microelectronics Reliability

👍 Small area penalty (just few bits per word)

👎 Susceptible to multi-bit upsets (encoding can usually only fix 1 bit per word)

- In any case, a **refreshing algorithm** is needed to correct errors. It needs to be fast enough in relation with clock frequency and expected SEU rate.

- Different techniques for program/data memory (e.g. one might accept corrupted data but not instructions).

Additional challenges

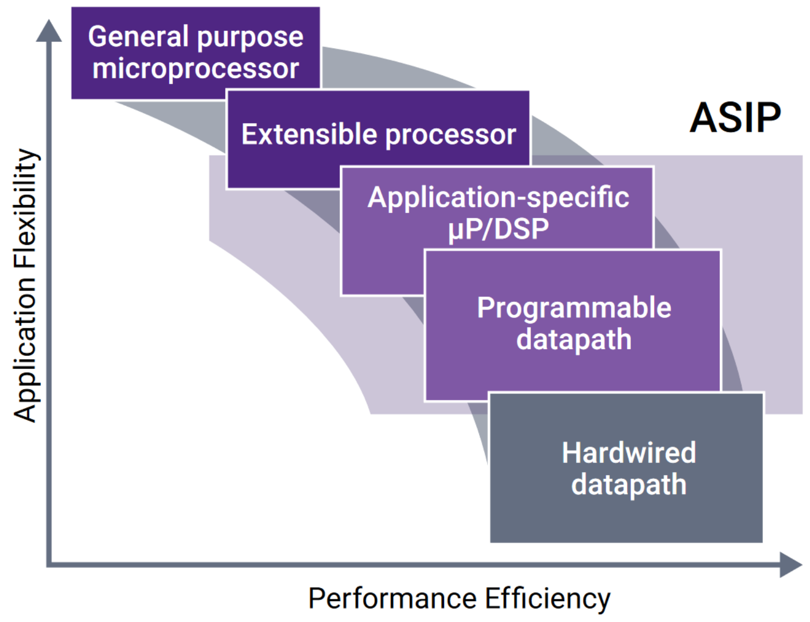
Data processing

A general-purpose microprocessor is good for control applications and light computing loads. But what about heavy data-processing tasks?

→ Specialized **hardware accelerators** help offload the CPU.

This approach comes with its own challenges:

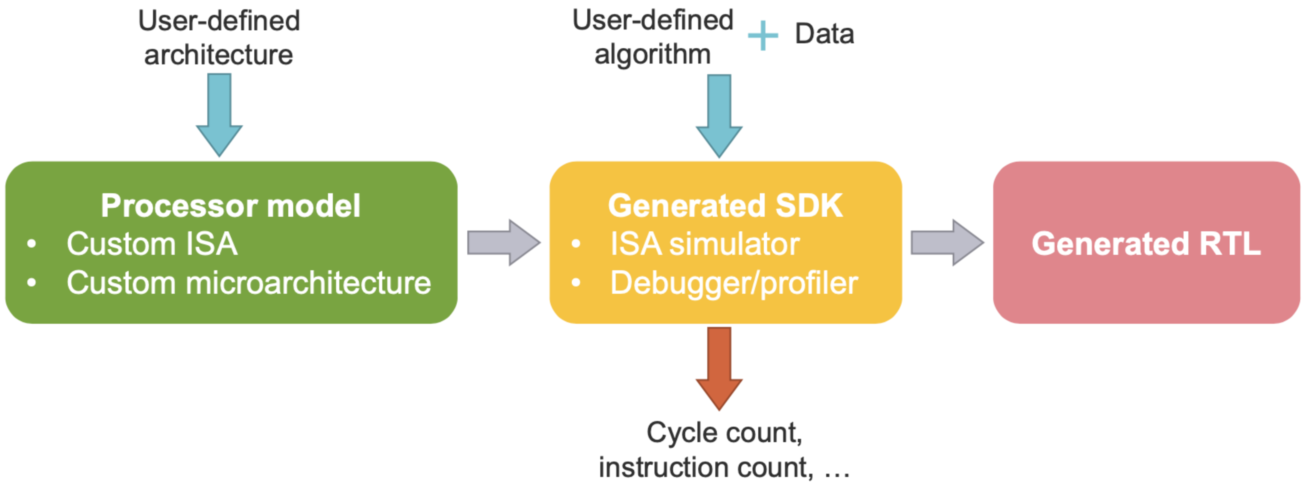
- **Dependence on the application**: hard to reuse on very different applications
- Need for **architectural study**: will the data processing take place in the pixel array, in the periphery, in a separate chip?



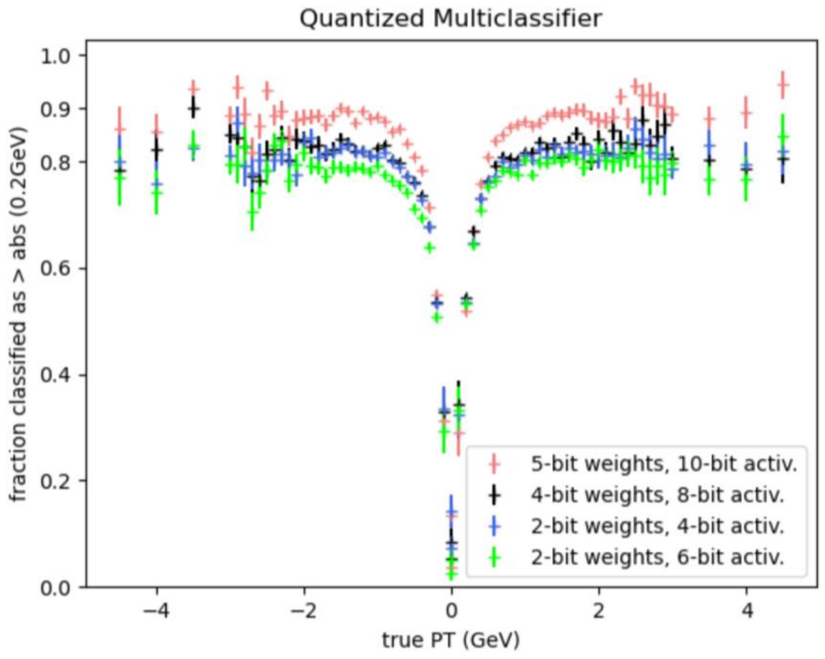
Data processing approaches

- **Application-Specific Instruction set Processors (ASIP)**: tradeoff between flexibility of general-purpose and performance of custom logic.

  A workflow has been demonstrated implementing a clustering algorithm in: M. Andorno et al., *Rad-hard RISC-V SoC and ASIP ecosystems studies for high-energy physics applications*, 2023, JINST 18

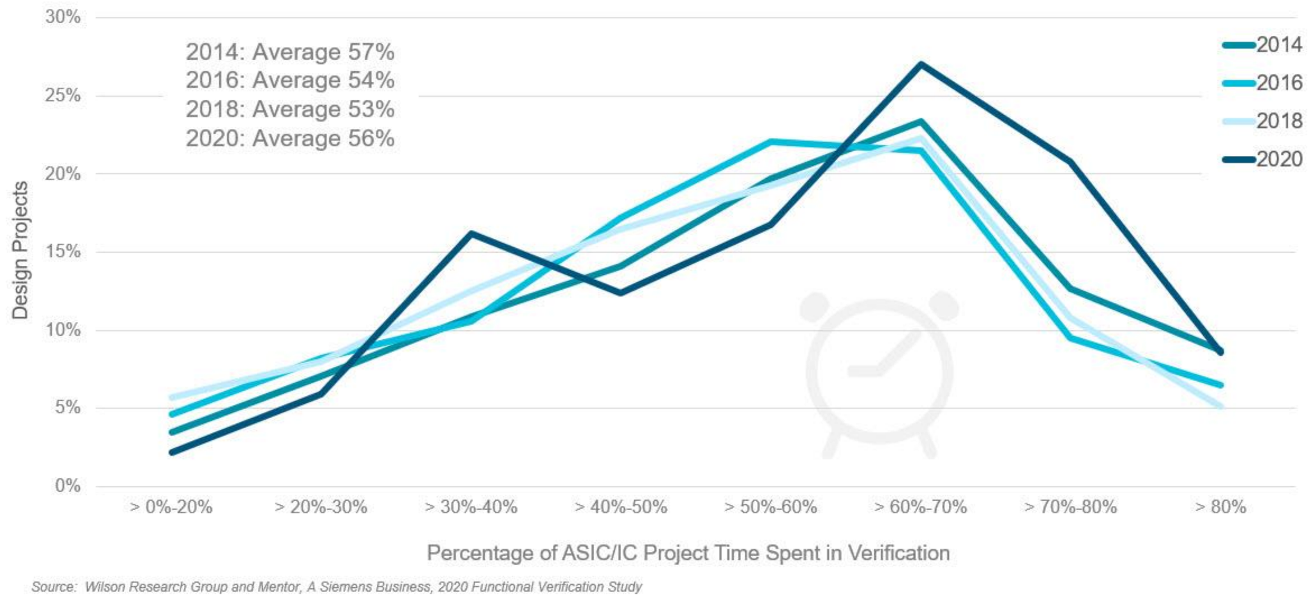


- **Machine learning accelerators**, as shown in the work of F. Fahim et al. presented at the last 28 nm Forum, can be used to implement data reduction, filtering and processing on-chip. A high-level synthesis and ASIC implementation has been presented, using a multi-classifier to cluster and filter hit data.



Verification

- Verification of SoCs and programmable hardware blocks is a challenge: it’s impossible to cover 100% of possible combinations of instructions/data/memory usage.



- Verification is a must and needs to go alongside design during the project timeline (not after).

- More information will be given in tomorrow’s presentation by Adithya Pulli

Wrap-up

Summary and open points

- On-chip programmability will provide more flexible, powerful and cost effective ASICs.
- Many challenges to be faced concerning radiation tolerance:
  - What’s the best strategy to harden a processor?
  - Do you need to harden the interconnect? How?
  - What memory architectures are the most suitable? How do you protect the memory?
- How to tackle heavy data-processing requirements?
- What’s the best strategy for SoC verification?

→ Many open points, lots of interesting work to be done to get an optimal solution.

Thank you!

home.cern