From Components to Architecture: An End-to-End Approach to Soft-Error Tolerance

McCluskey PhD Competition Semifinals

Michael Rogenmoser michaero@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch

ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

PULP Platform
Open Source Hardware, the way it should be!

@pulp_platform
pulp-platform.org
youtube.com/pulp_platform

Motivation

- **Compute in Space is increasing (New Space)**
  - Complex data processing
    - Communication satellites (routing, filtering, RF, …)
    - Earth observation satellites (pre-processing, compression, …)
  - Datacenters in space
- **Problem: Radiation in Space causes Faults in SoCs**
  - Bitflips throughout the system
    - Single Event Upsets (SEUs)
    - Single Event Transients (SETs)
- **Space Processor Systems lag behind**
  - Old designs & technologies
  - Expensive & low performance/efficiency

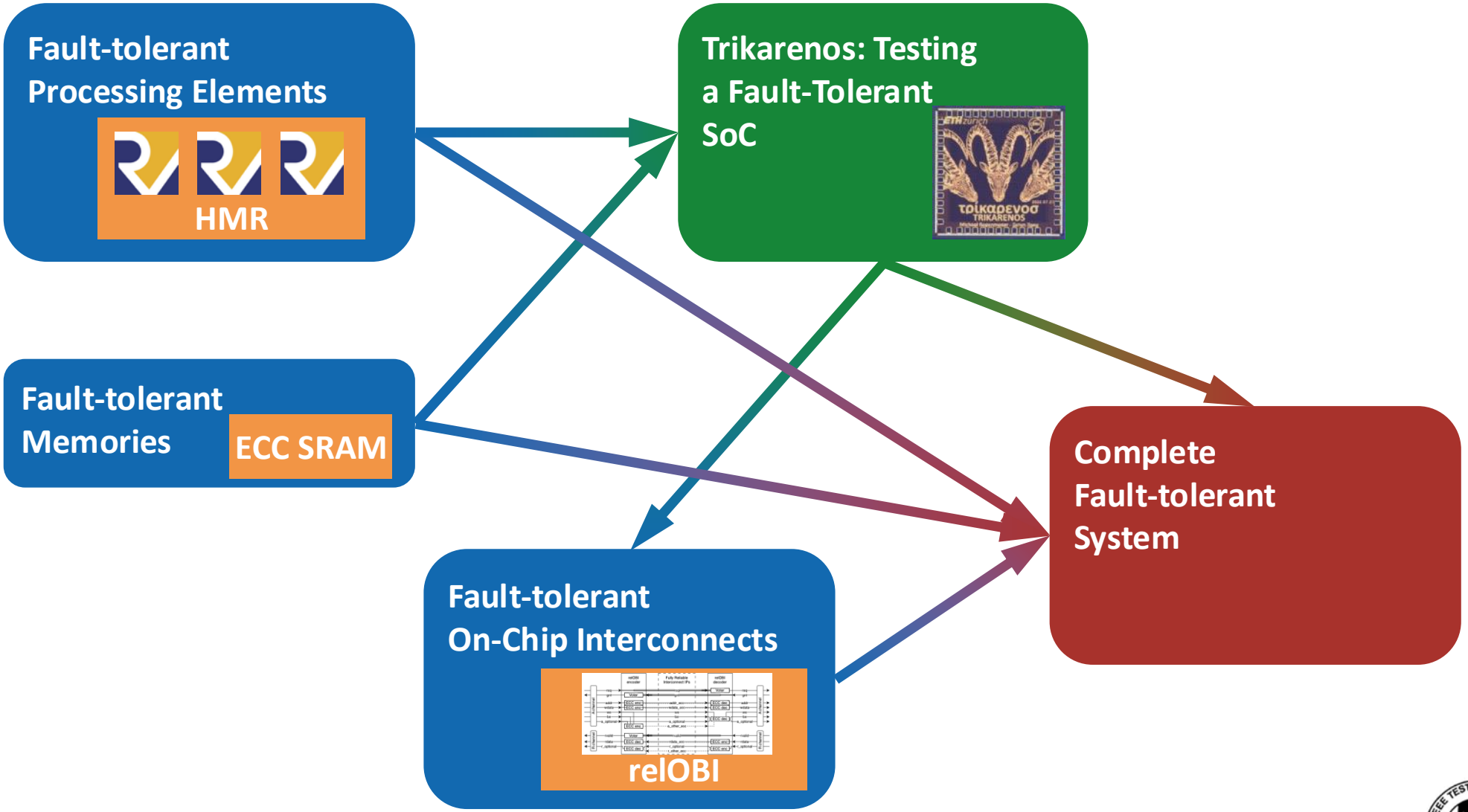
Existing Approaches

- **Commercial Space-grade Processor systems**
  - Rely on old, proven technology
    - Lags behind commercial processing
  - Rely on Radiation-Hardened technology stack
    - Lower performance & efficiency
- **Fine-grained Triple-Modular Redundancy (TMR)**
  - Triplicate everything & vote everywhere
    - 4-6x area required
- **Dual-/Triple-core Lockstep (DCLS/TCLS)**
  - May only detect faults or end in a degraded state
  - Rigid (inflexible when performance is needed)
  - Leaves part of the design unprotected (e.g., voters)
- **Error Correcting Codes**
  - Limited applicability

Thesis Goals

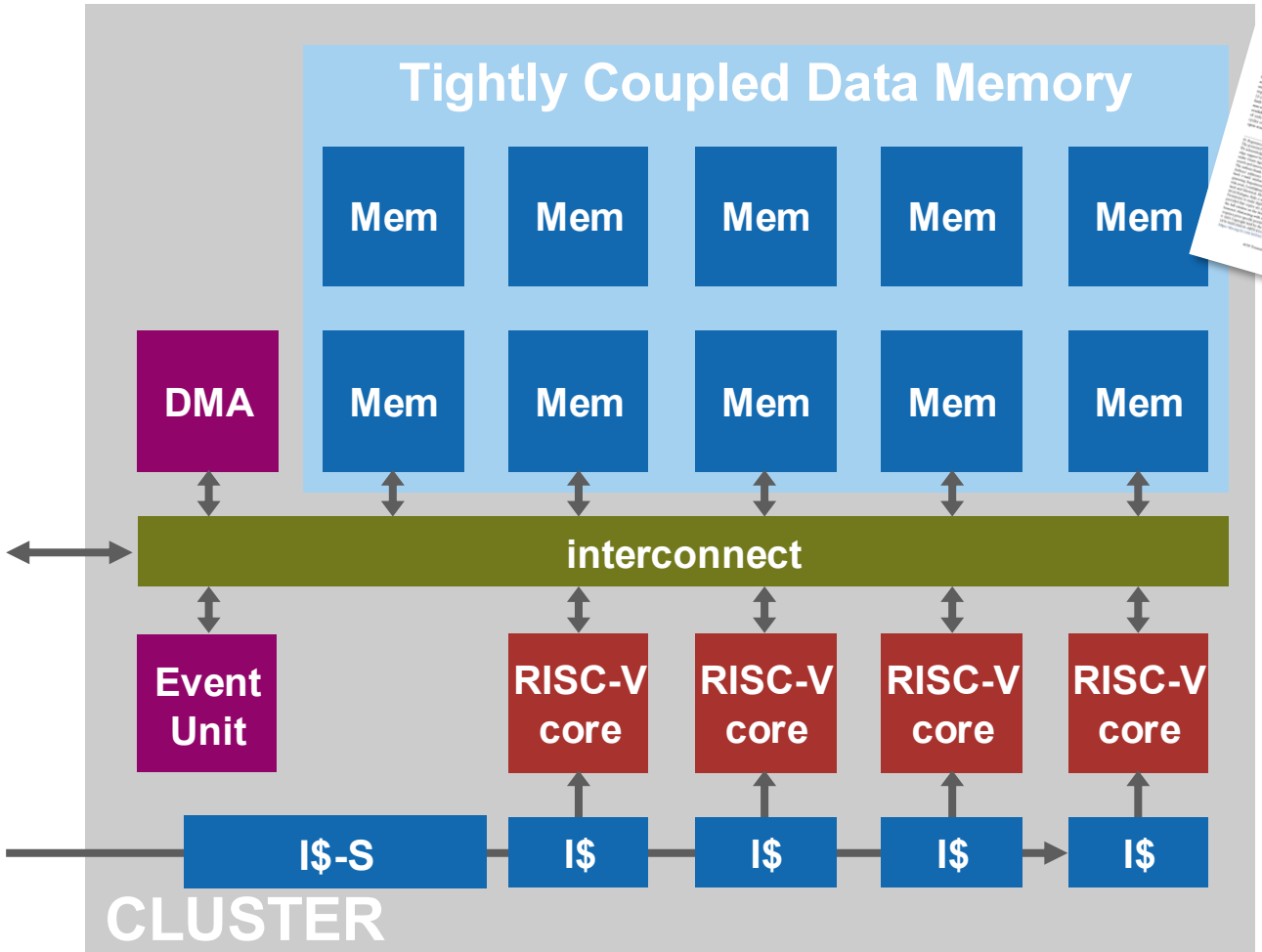
- **Design Space-ready processor system**
- **Leverage standard technologies**
  - No rad-hard cell requirements
  - Only TID validation required
- **Tolerant to Single faults (SEU, SET)**
- **Functional after most single faults**
  - Can correct for faults at runtime
  - No reboot required
- **RISC-V-based open-source designs**

Thesis contributions



The PULP Multicore Cluster foundation

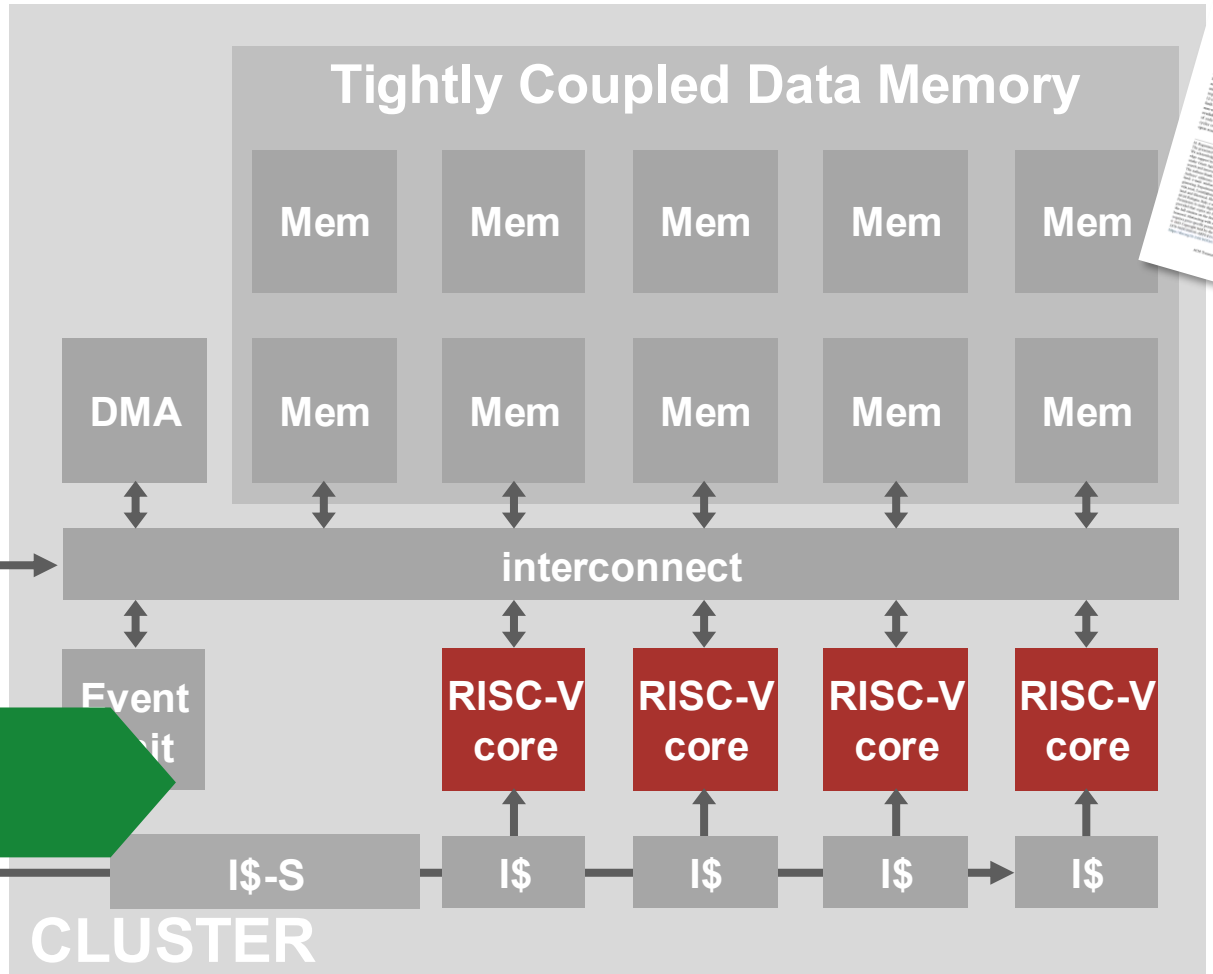
- **Multicore RISC-V processor cluster**
- **Local Scratchpad memory**
- **Low-latency interconnect**
- **Hierarchical Instruction Cache**
- **System peripherals**



The PULP Multicore Cluster foundation

- **Multicore RISC-V processor cluster**
- **Local Scratchpad memory**
- **Low-latency interconnect**
- **Hierarchical Instruction Cache**
- **System peripherals**

Start with the processing elements



Protecting the Cores

Protecting the Cores

Protecting the Cores

Triple-Core Lock Step

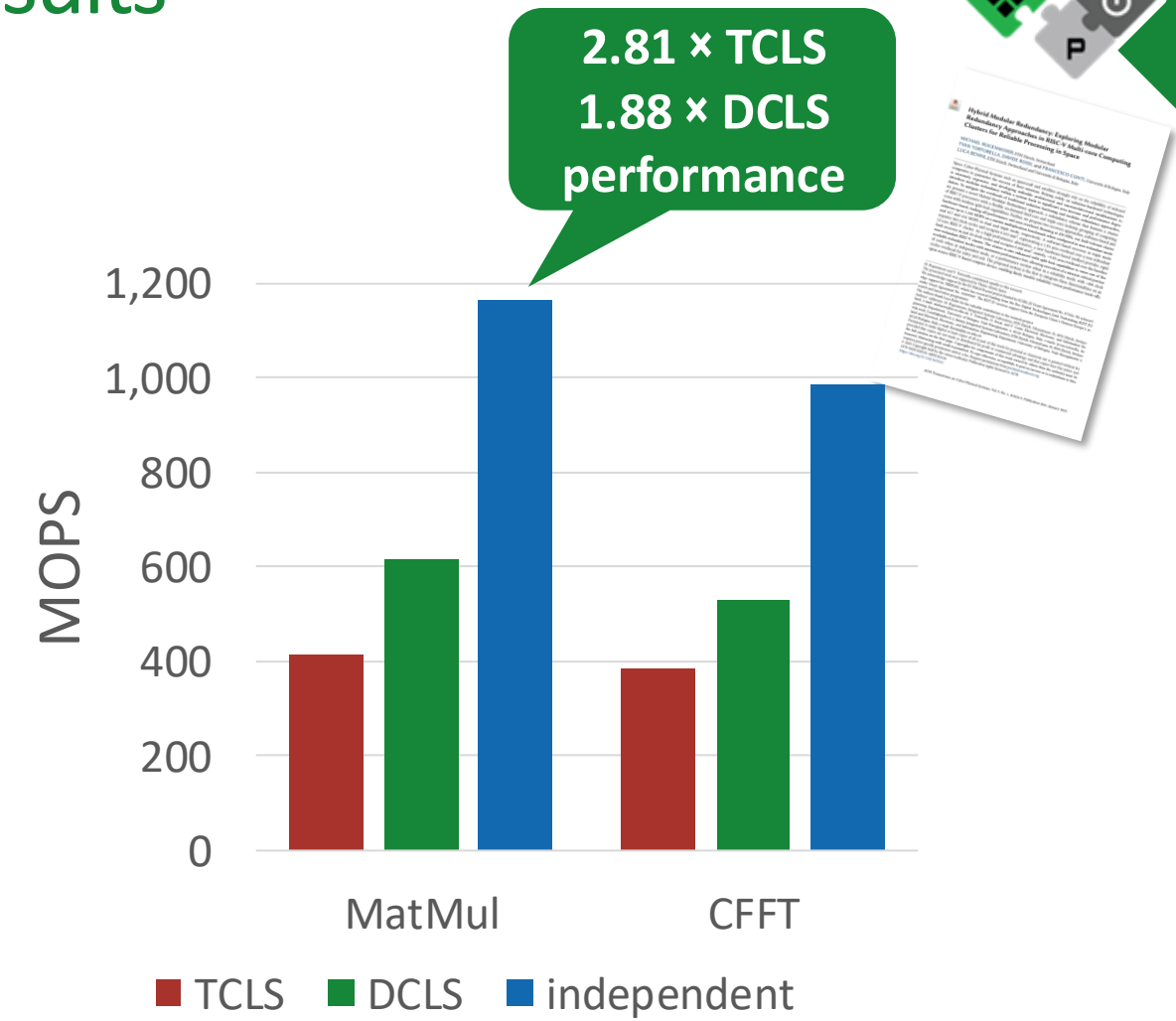
On-Demand Redundancy Grouping

On-Demand Redundancy Grouping

On-Demand Redundancy Grouping

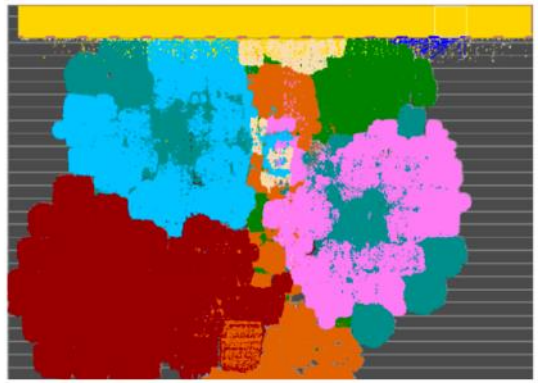
Hybrid Modular Redundancy Results

- **Enable independent, Dual-core Lockstep, & Triple-core Lockstep**
  - Re-configuration in 82-411 cycles
- **Software re-synchronization mechanism for Triple-core Lockstep**
  - ~363 cycle latency
  - No modification to cores required
- **Optional hardware backup & synchronization unit**
  - Direct register access required
  - Enable Dual-core & Triple-core correction & re-synchronization
  - ~24 cycles re-synchronization



Hybrid Modular Redundancy Implementation

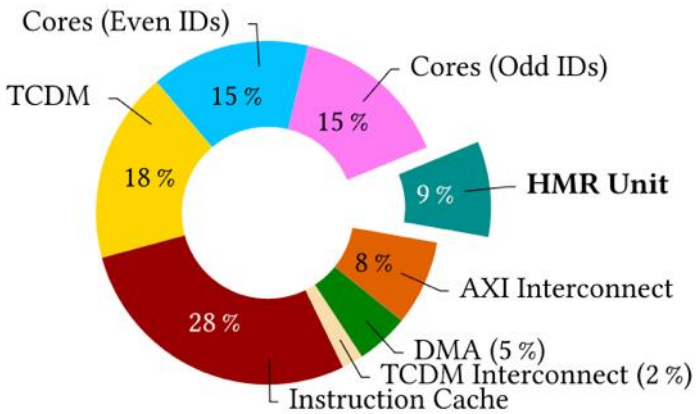
- **12-core cluster implemented in GF 22nm**
- **Minimal overhead for HMR Unit**
  - 0.3-1.3% using software recovery mechanisms
  - 8.4-9.4% adding hardware recovery unit
- **Up to 430MHz**
  - No impact wrt. baseline



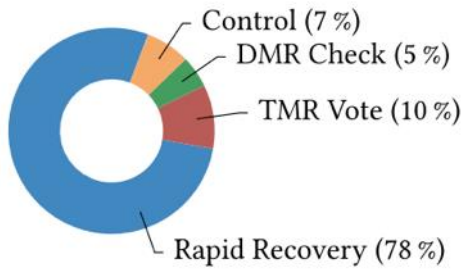
(a) Layout of the physical implementation of the fault-tolerant PULP cluster featuring full HMR and *rapid recovery*.

(b) PULP cluster area comparison in all available configurations.

PULP Cluster Area [mm^2]		Overhead
Baseline	0.604	-
DMR	0.605	0.3%
TMR	0.608	0.7%
HMR	0.612	1.3%
With Rapid Recovery		
DMR	0.654	8.4%
TMR	0.657	8.8%
HMR	0.660	9.4%



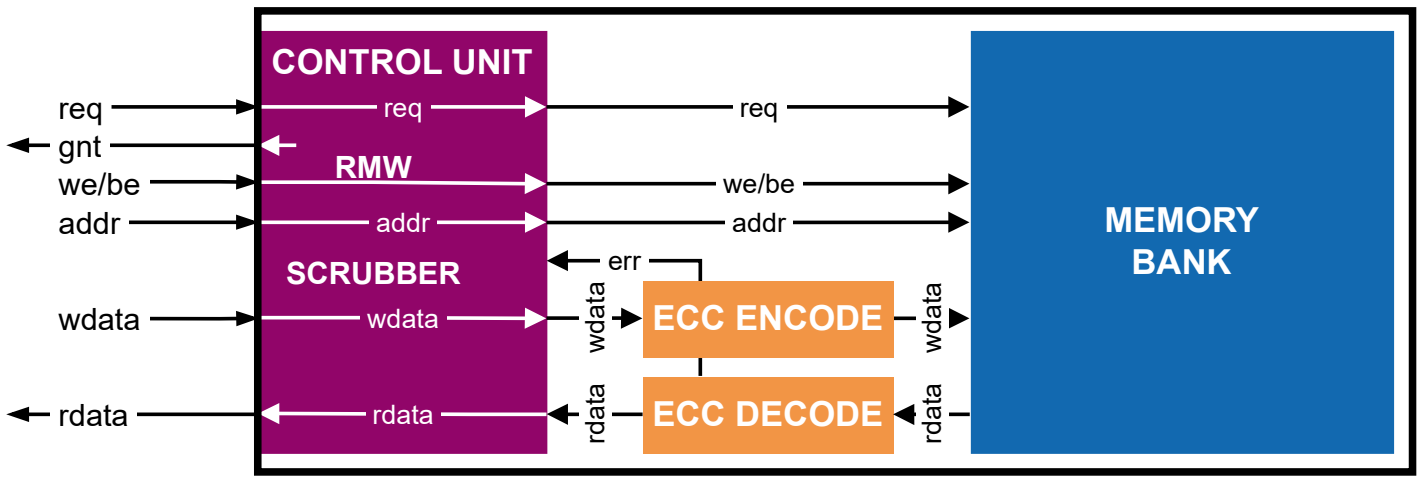
(c) Area breakdown of the fault-tolerant PULP cluster featuring full HMR and *rapid recovery*, with the same color as in the physical implementation layout.



(d) HMR unit area breakdown.

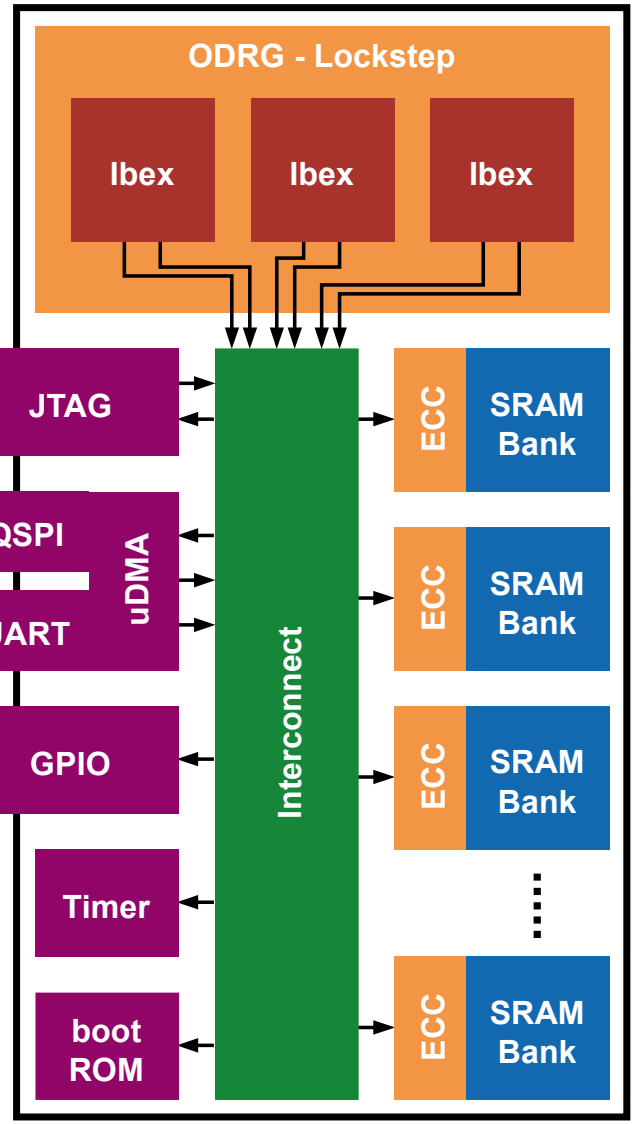
ECC memory

- **Hsiao code for efficient encoding**
  - Single Error Correction, Double Error Detection
- **32bit word stored as 39bit with parity**
- **Read-modify-write for efficient byte-wise access**
- **Scrubber for continuous correction**
  - Avoids latent errors causing uncorrectable error



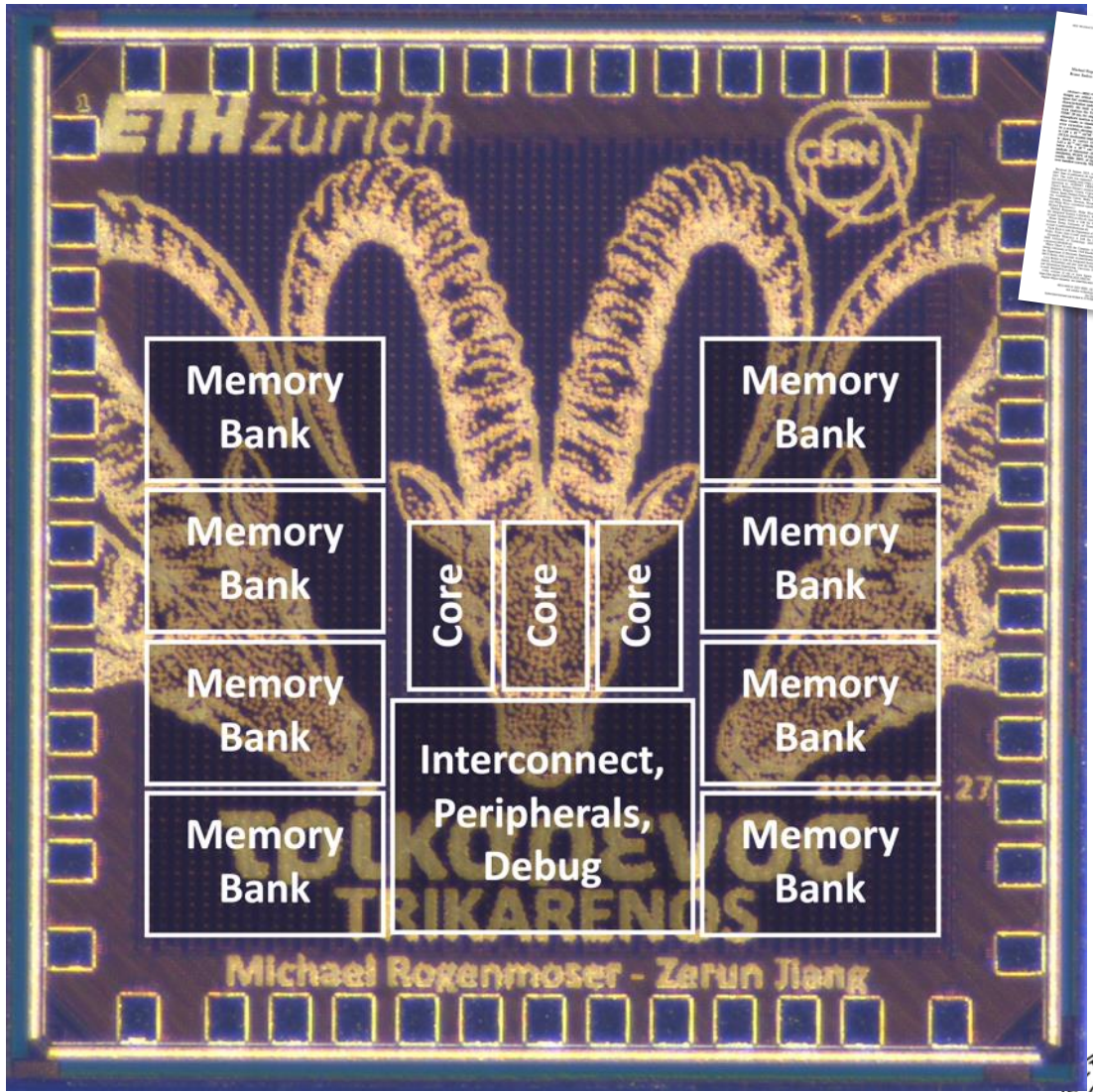
Trikarenos Design

- **PULPissimo-based SoC Design**
  - Modified for reliability
- **Three lockstepped cores with voting**
- **ECC-protected memory**
- **Peripherals**
  - UART
  - QSPI
  - GPIO
- **JTAG for programming and internal access**
- **Low-latency interconnect**



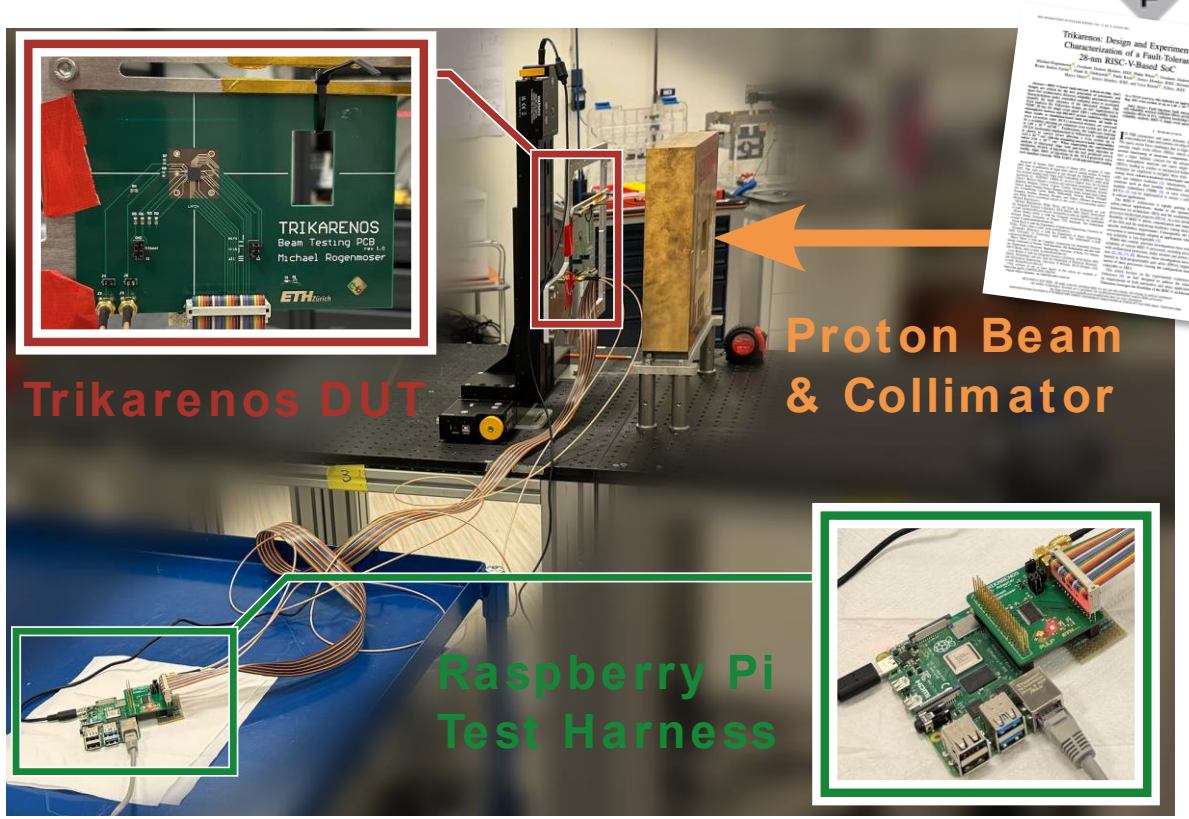
Trikarenos implementation

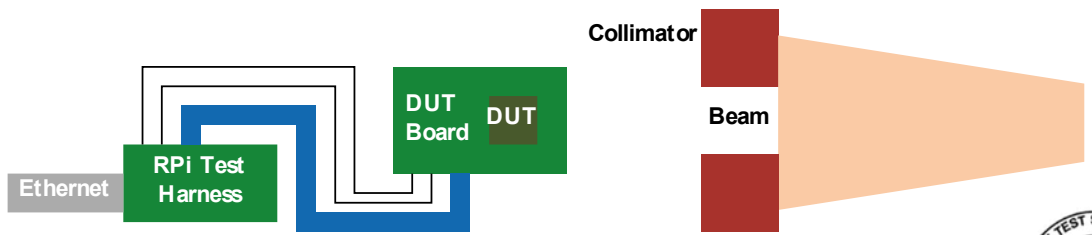
- **Implemented in TSMC 28nm**
  - TID tolerance previously investigated
- **Standard cells**
  - No hardened cells
- **Standard flows**
  - No additional protections (clock/reset tree)
- **Physical separation for cores**
  - Ensures single particle does not cause SEU in multiple cores
- **250MHz target, operating at 125MHz**
  - 0.9V core, 1.8V I/O



Experimental Setup

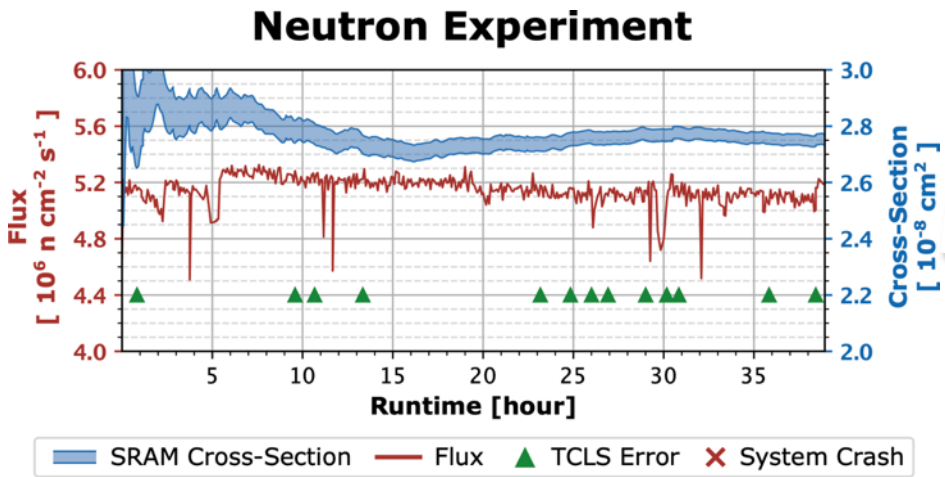
- **Trikarenos standalone on PCB**
- **Raspberry Pi**
  - For programming and monitoring
  - Stores data
  - Ethernet to transmit outside & control
- **Application on Trikarenos**
  - Coremark
  - Register operations to accumulate errors and detect TCLS correction
  - GPIOs
    - Heartbeat
    - Exception signalling

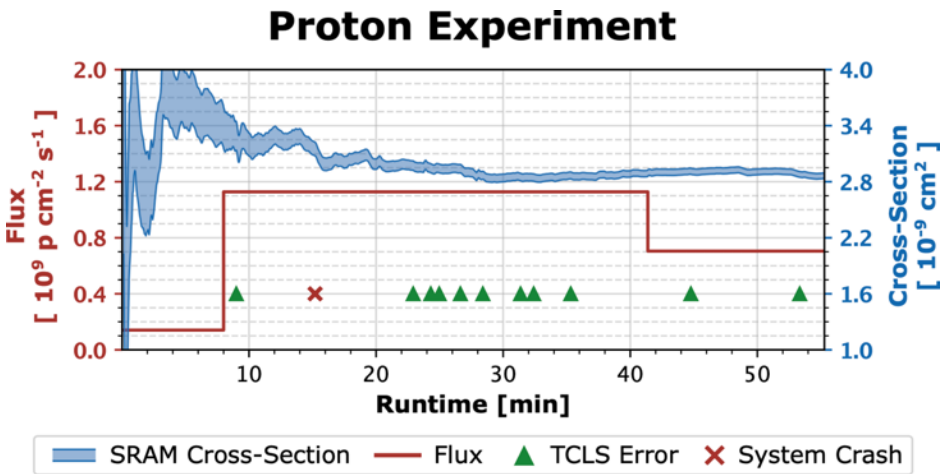




Trikarenos Results

- **Atmospheric Neutrons (ChipIR)**
  - 18’786 corrected SRAM errors
    - $(1.08 \pm 0.01) \times 10^{-14}$ cm^2 bit^{-1} SRAM cross-section
  - 13 corrected Lockstep faults
    - $(2.55 \pm 0.68) \times 10^{-11}$ cm^2 cross-section
  - 0 system faults
- **200 MeV Protons (HollandPTC)**
  - 8’249 corrected SRAM errors
    - $(1.12 \pm 0.01) \times 10^{-15}$ cm^2 bit^{-1} SRAM cross-section
  - 11 corrected Lockstep faults
    - $(5.25 \pm 1.51) \times 10^{-12}$ cm^2 cross-section
  - 1 system fault





SpaceX Transporter-13 launch, March 15th 2025

Carried our first PULP chip Trikarenos to space

Aboard the ALICE experiment by ARIS

Find the missing pieces!

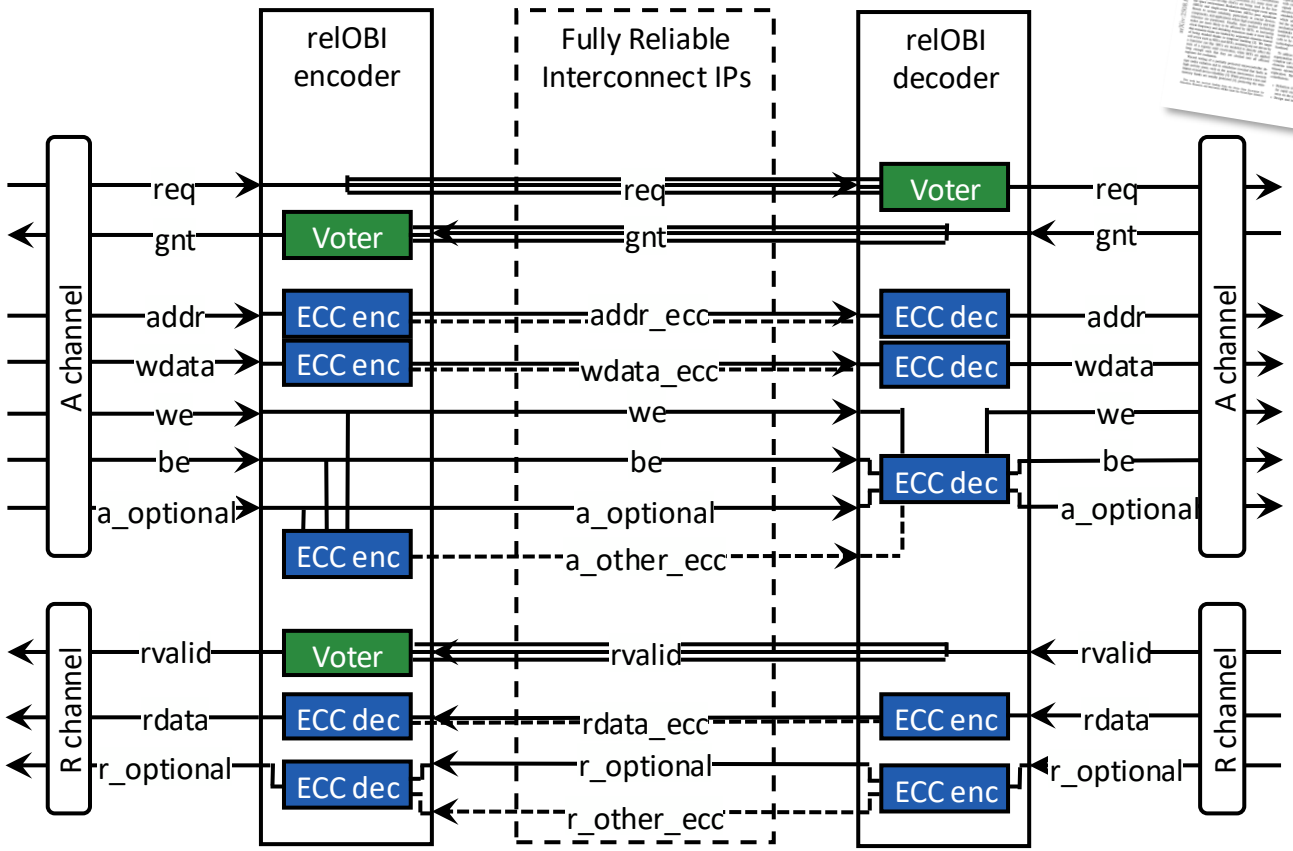
- **Fault injection simulation**
  - Inject SEUs in all FFs of the design
  - Random selection
- **0.9% indicate failure**
- **Which design partition causes failures?**
- **Interconnect is the main vulnerability!**

Termination Reason	Total Amount	Total Probability	In Cores Amount	In Cores Probability
Correct Termination	**99 096**	**99.10 %**	**34 524**	**100.00 %**
Correct	18 958	18.96 %	7642	22.14 %
TCLS	12 283	12.28 %	11 806	34.20 %
Latent Non-Propagating	61 514	61.51 %	13 805	39.99 %
Latent Propagating	6341	6.34 %	1271	3.68 %
Functional Error	**904**	**0.90 %**	**0**	**0.00 %**
Timeout	822	0.82 %	0	0.00 %
Exception	42	0.04 %	0	0.00 %
Incorrect	40	0.04 %	0	0.00 %
Total	**100 000**	**100.00 %**	**34 524**	**100.00 %**

Components	Functional Errors Timeout	Exception	Incorrect	Fault / Injections
Interconnect	613	22	26	661 / 5176
Debug Module	164	0	1	165 / 7379
Peripherals	26	12	13	51 / 41 867
Cores & Voters	19	2	0	21 / 35 934
Memory	0	6	0	6 / 9644
Total	**822**	**42**	**40**	**904 / 100 000**

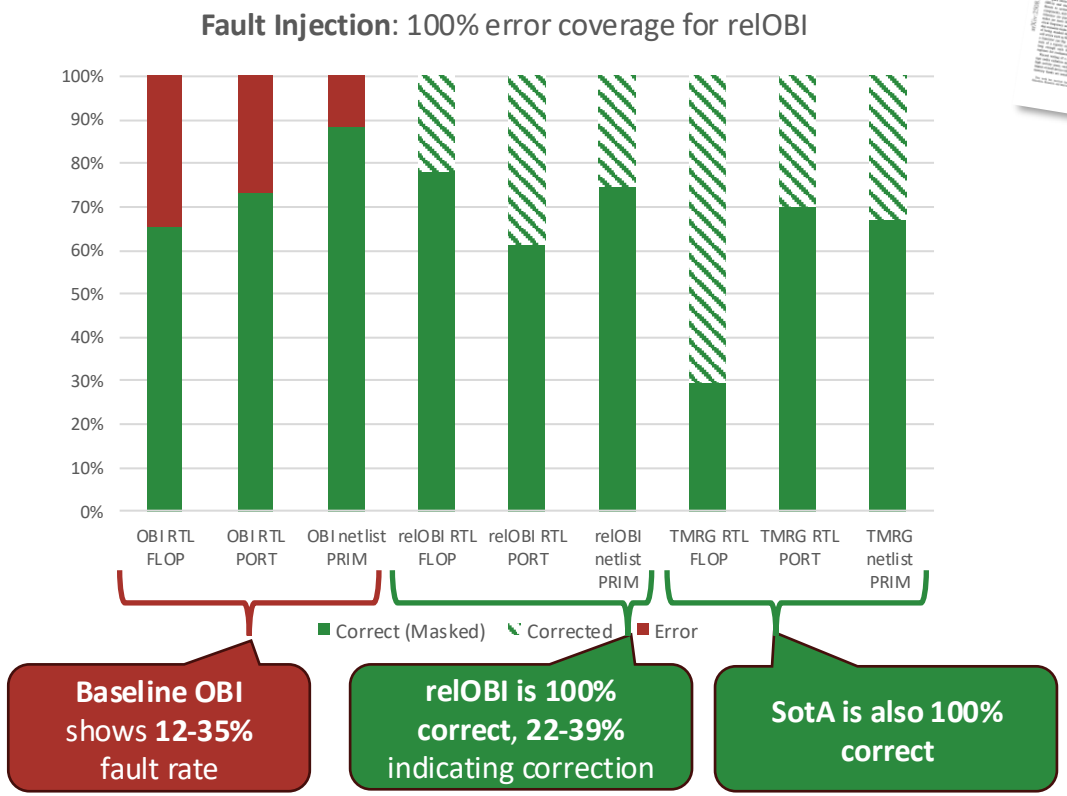
Reliable Interconnect Design

- **Open Bus Interface (OBI)**
  - SRAM-like with flow control
- **TMR for Handshake signals**
  - Minimal handling for critical signals
- **ECC for data signals**
  - More compact for transport
  - Re-use of encoding in memory
- **En- & decoding**
  - Interfacing with existing components
- **Reliable interconnect IP**
  - In-flight correction



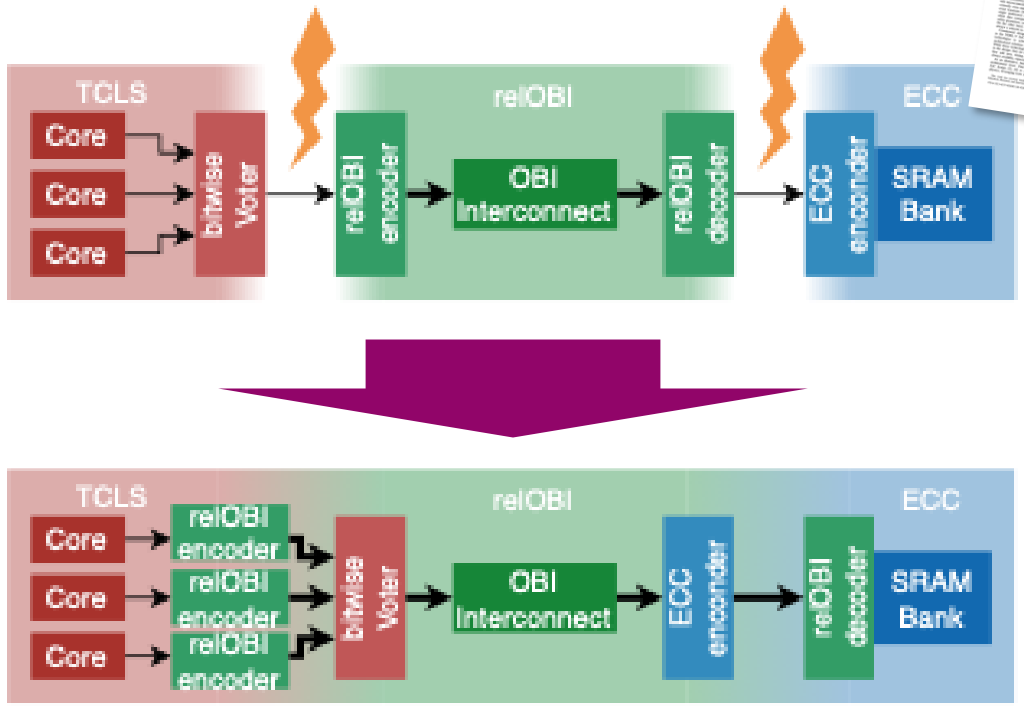
Reliable Interconnect Results

- **Fault injection into crossbar design**
  - All registers in RTL (FLOP)
  - All ports in RTL (PORT)
  - All primitives in synthesized netlist (PRIM)
- **Synthesized in TMSC 7nm**
  - 2.6x area overhead
  - 1.4x timing overhead
- **SotA: Fine-grained TMR**
  - Full design triplicated
  - Voters inserted following registers
  - 4.8x area overhead
  - 1.4x timing overhead



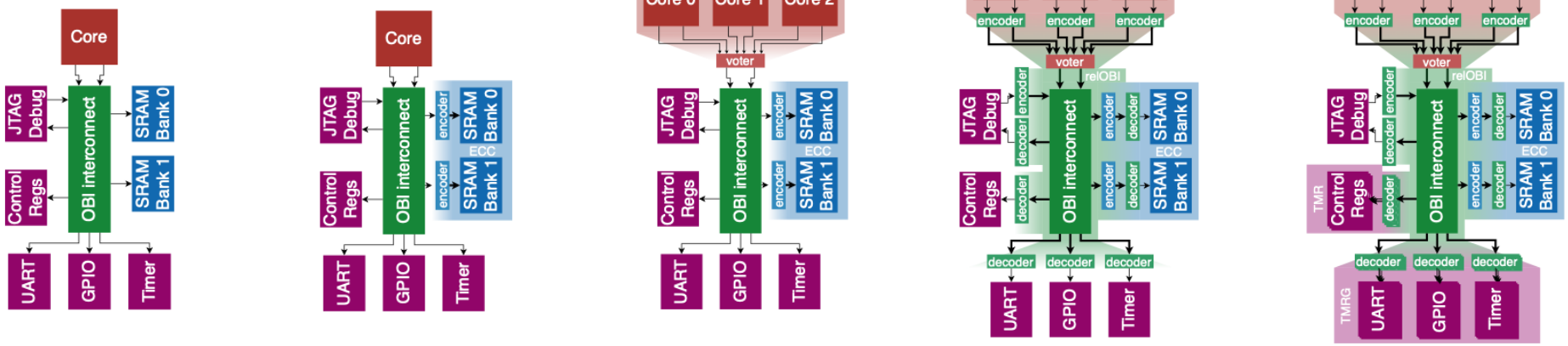
Who Checks the Checker?

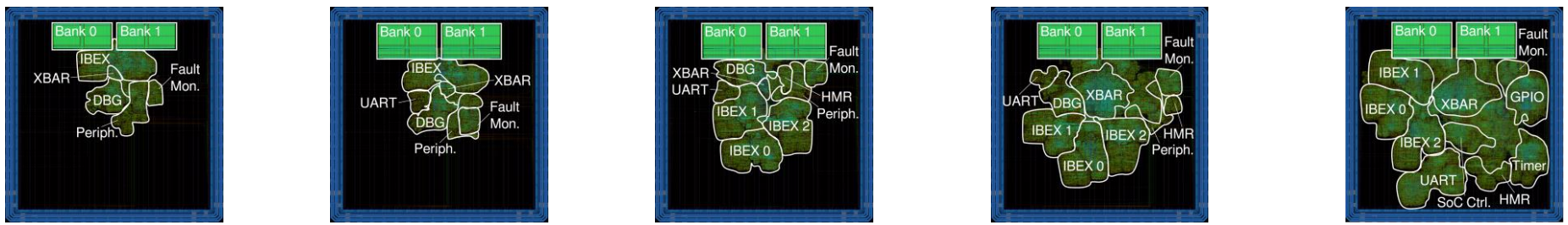
- **Main domains in system now protected**
  - Cores, memory, interconnect
- **No protection in between!**
  - Connections vulnerable
  - Voters vulnerable
  - En-/decoders vulnerable
- **Idea: overlap the protection domains**
  - Encode Interconnect before voting cores
  - Encode for memory before decoding interco.
- **Intermediate faults handled by overlapping protections!**



Who Checks the Checker

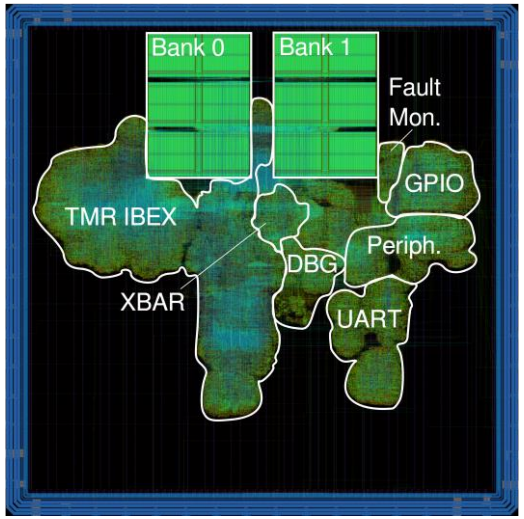
- **Test in an open-source MCU design, combining all previous elements**
- **Ensure strict overlap of protection domains**
- **Iterative exploration**

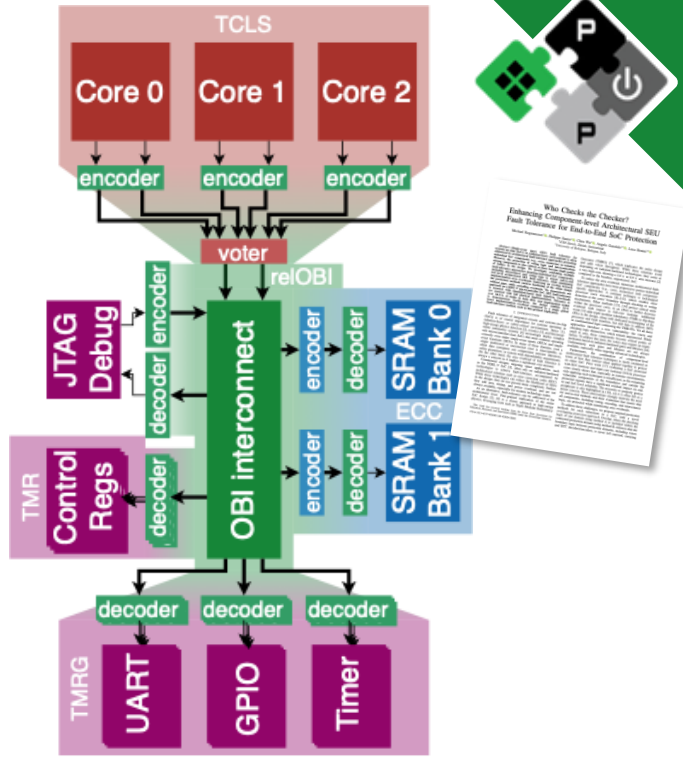


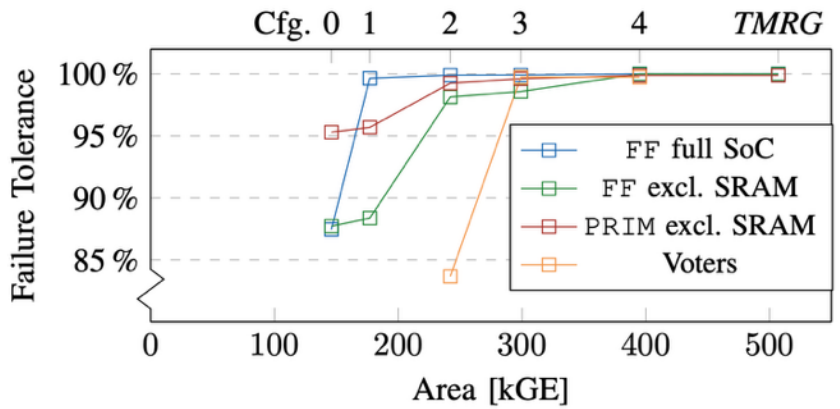


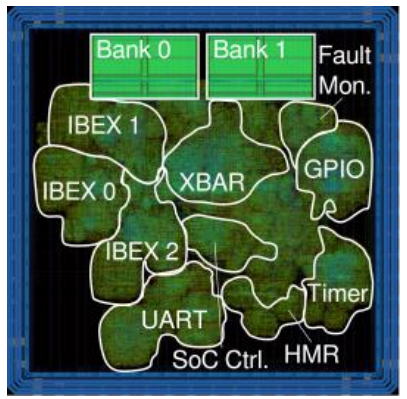
Who Checks the Checker

- **Compare fully protected design to fine-grained TMR**
- **Equivalent fault coverage (99.9%)**
- **2.71x area overhead**
  - 3.48x for fine-grained TMR
- **Voters confirmed protected**
- **All configurations improve tolerance**









Conclusion

- **Hybrid Modular Redundancy**
  - Configurable Fault-tolerance for Cores
- **Trikarenos**
  - Tapeout tested under Radiation
- **relOBI**
  - Reliable interconnect addressing gaps
- **Who checks the checker**
  - Close all remaining gaps in architectural protection

➢ SEU fault-tolerance across full SoC
➢ Node-agnostic architectural methods

Michael Rogenmoser michaero@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch

HMR Re-synchronization