Who Checks the Checker?

Enhancing Component-level Architectural SEU Fault Tolerance for End-to-End SoC Protection

ETH zürich

Michael Rogenmoser michaero@iis.ee.ethz.ch
Philippe Sauter phsauter@iis.ee.ethz.ch
Chen Wu chenwu@iis.ee.ethz.ch
Angelo Garofalo agarofalo@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SwissChips

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

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

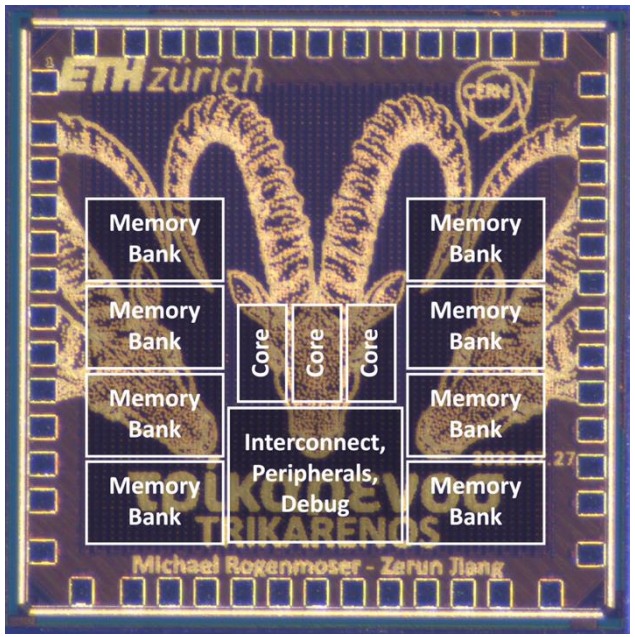
Motivation

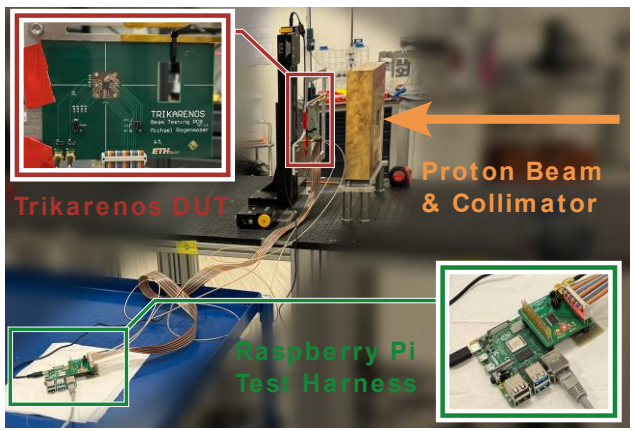
- **Compute in Space is increasing (New Space)**
  - Complex data processing
    - Communication satellites (routing, filtering, RF, …)
    - Earth observation satellites (pre-processing, compression, …)
  - Datacenters in space
- **Problem: Radiation in Space causes Faults in SoCs**
  - Bitflips throughout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

Existing Solutions

- **Radiation-hardened technology (node, cell design)**
  - Lags behind advanced commercial nodes
  - Expensive, lower performance, efficiency, less availability
- **Fine-grained Triple-Modular Redundancy (TMR)**
  - Triplicate everything & vote everywhere
  - 4-6x area required
- **Component-level protection methods**
  - Lockstep processors
  - ECC Memory
- **Gaps remaining in component-protected systems**
  - Some unprotected components
  - Voters & checkers

Trikarenos uncovered issues





- **RISC-V microcontroller**
- **Component protection**
  - Processor cores
  - Memory
- **Radiation Tests**
  - Confirmed protection mechanisms
  - Identified Gaps
- **Fault Injection Simulations**
  - Showed unprotected components affect operation
  - Highlight voters remain an issue

	Total		In Cores	
Termination Reason	Amount	Probability	Amount	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 %**

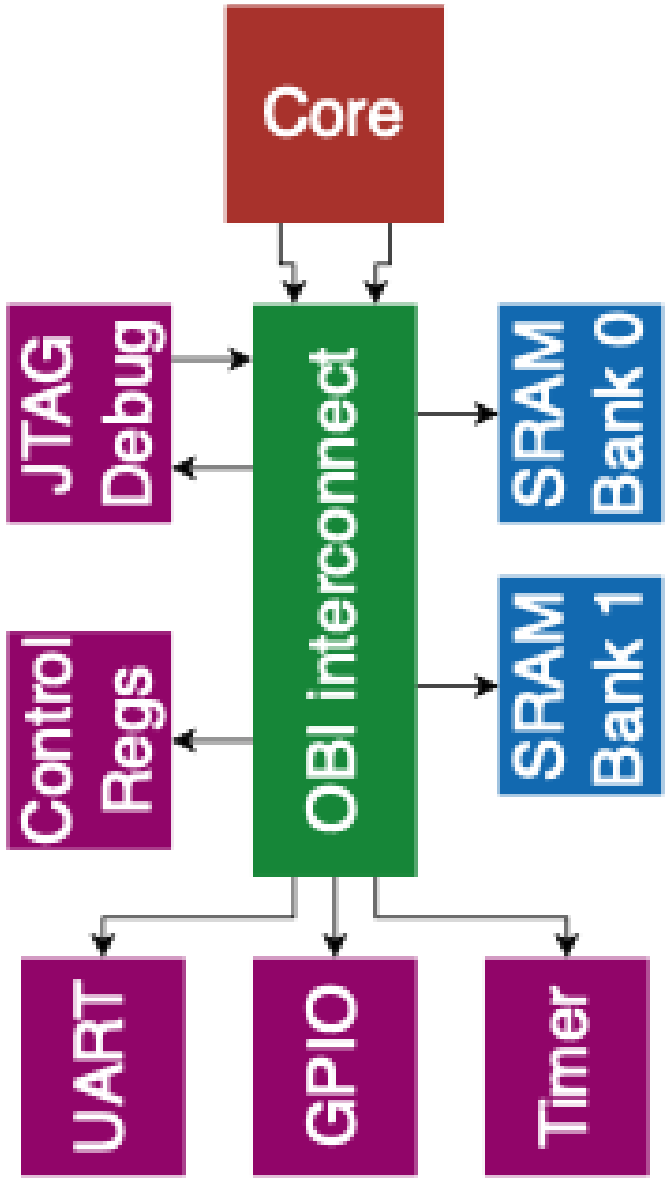
Components	Functional Errors			Fault / Injections
	Timeout	Exception	Incorrect	
Interconnect	613	22	26	661 / 5176
Debug Module	164	0	1	165 / 7379
Peripherals	26	12	13	51 / 41 867
Cores & Voters	19	2	0	21 / 35 934
Memory	0	6	0	6 / 9644
Total	**822**	**42**	**40**	**904 / 100 000**

Contributions

- **Complete fault-tolerant RISC-V-based MCU SoC**
  - Dedicated architectural fault tolerance methods for each component
  - End-to-end: Addressing all gaps in the system
- **Overlapping fault-tolerance method**
  - Ensuring error detection and correction without rad-hard gates
- **Area and timing analysis of incremental configurations**
  - Leveraging open-source frameworks
- **Fault tolerance validation and analysis**
  - Fault injection simulation for different fault types
  - Incrementally adding protections for tradeoff analysis

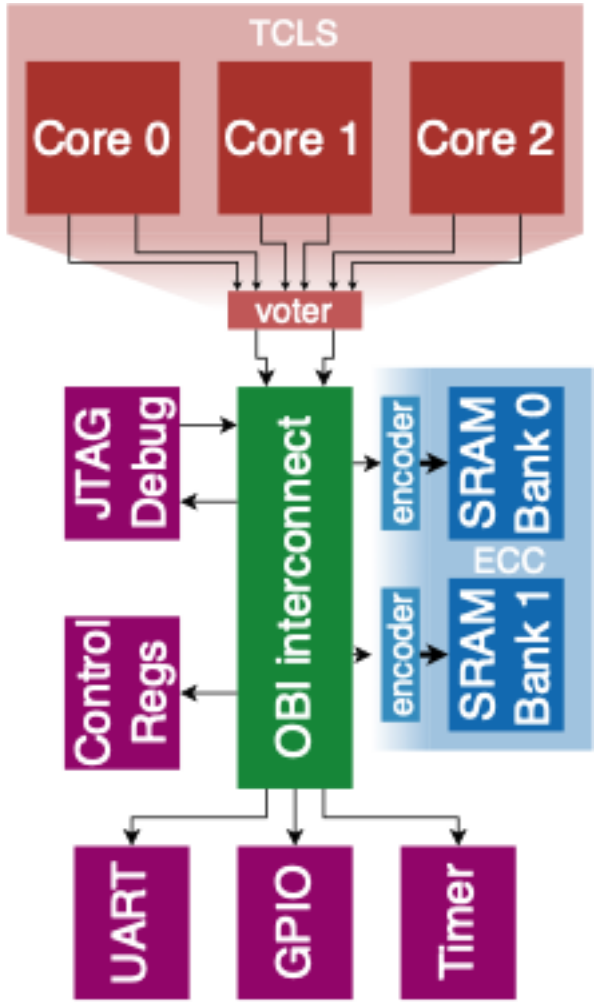
The croc MCU

- **Processor Core**
  - RISC-V
  - 2-stage in order
- **On-chip SRAM memory**
  - Instruction and Data bank
- **OBI Interconnect**
  - Low-latency
- **Peripherals**
  - System configuration registers
  - I/O peripherals (UART, GPIO)
  - JTAG RISC-V Debug module

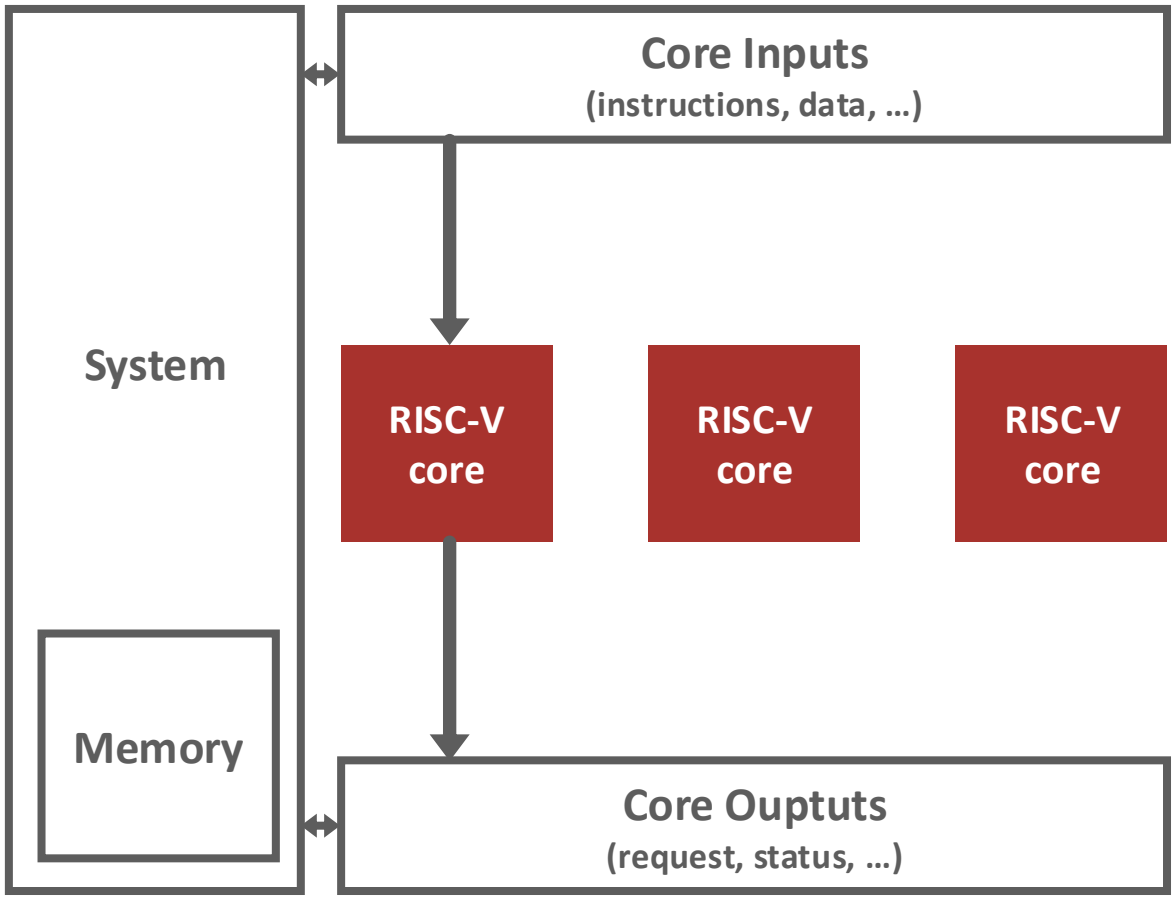


Adding fault-tolerance to croc

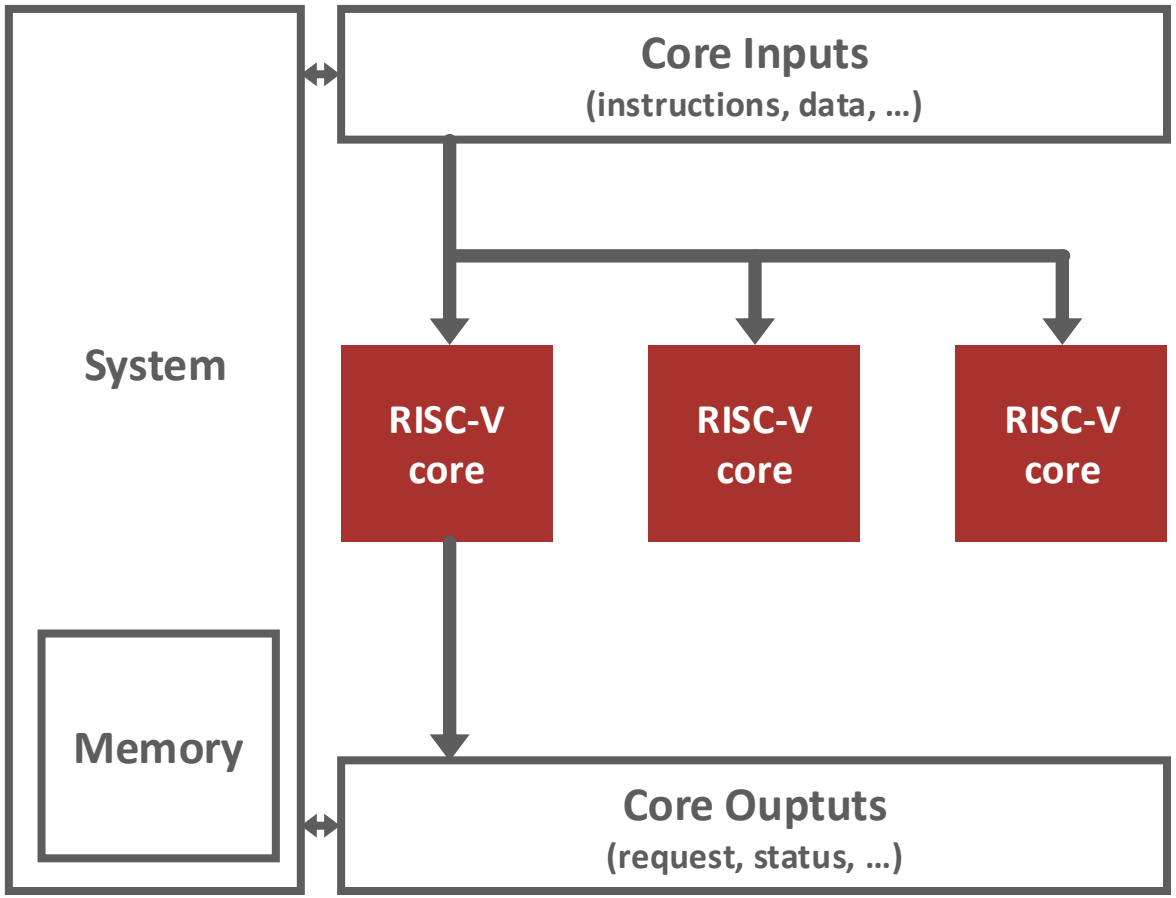
- **ECC Memory**
  - Efficient protection for static data
- **Lockstep Processor Cores**
  - Ensure correction for processing data



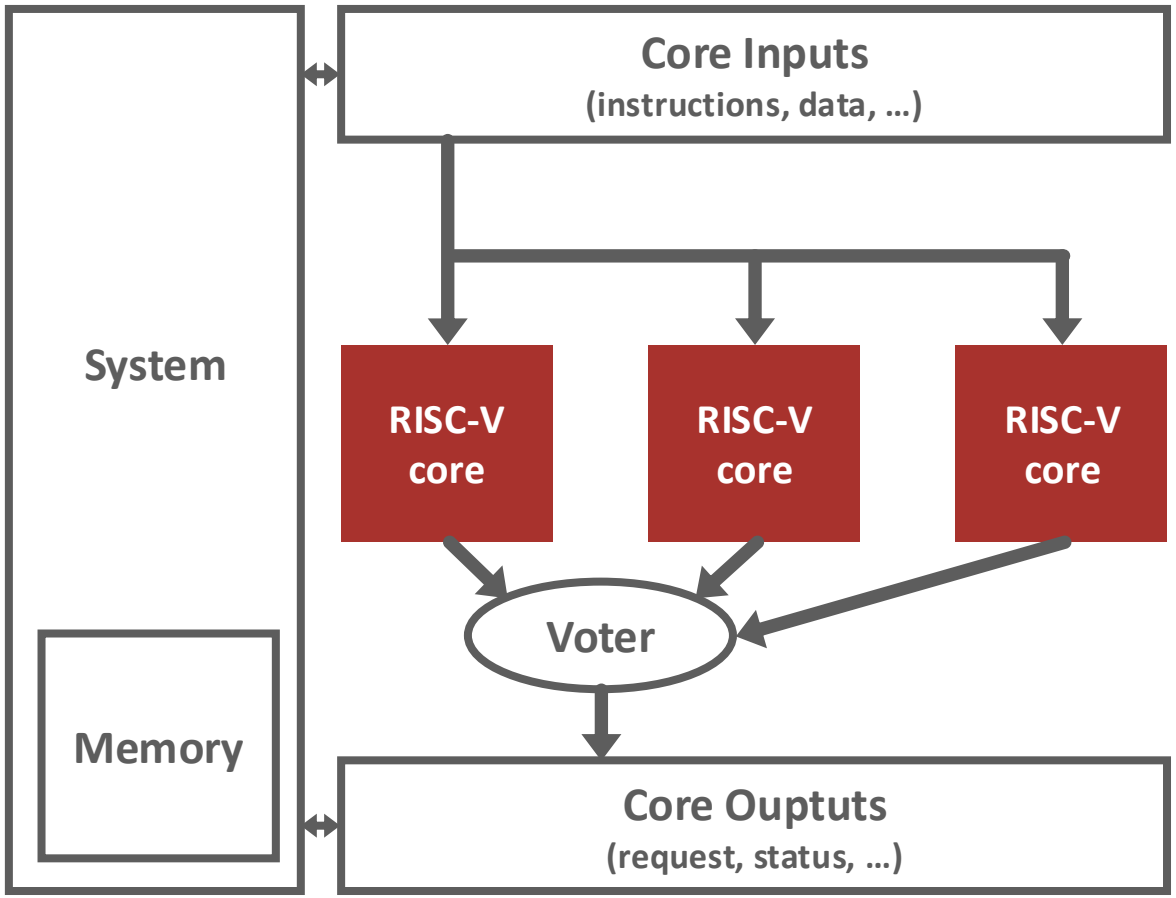
Protecting the Cores



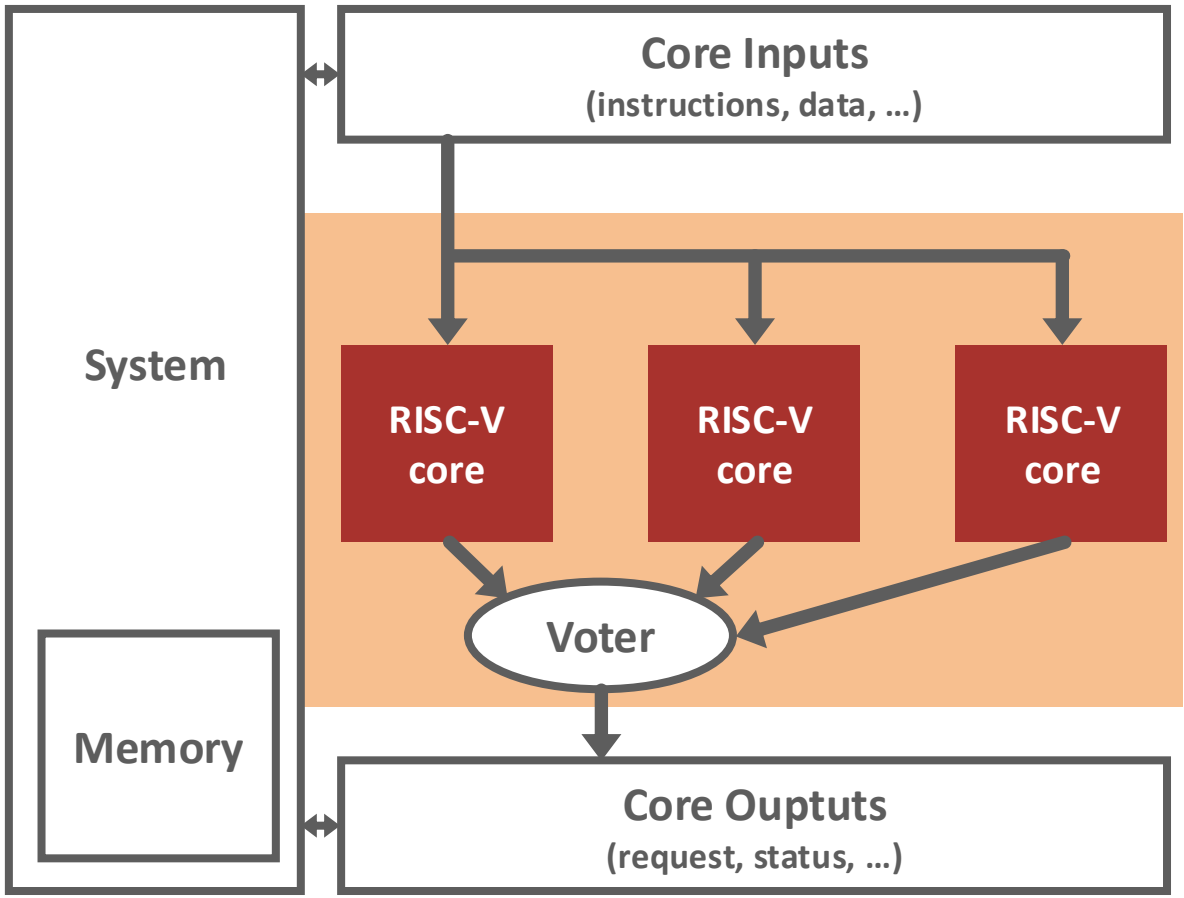
Protecting the Cores



Protecting the Cores

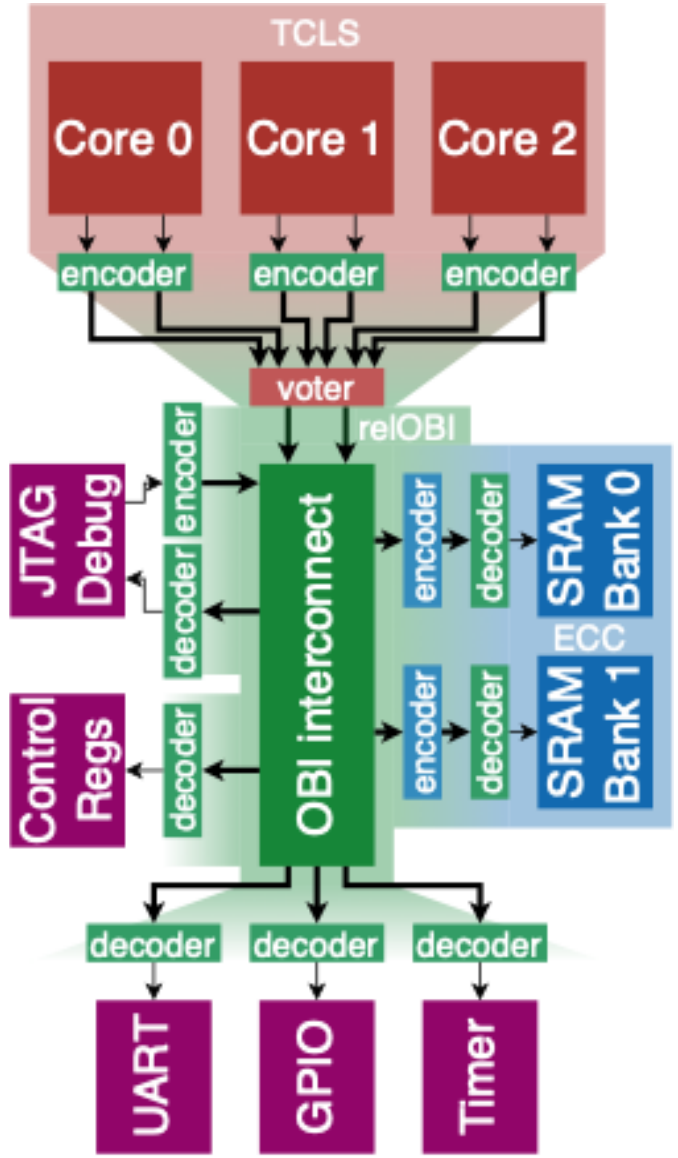


Triple-Core Lock Step



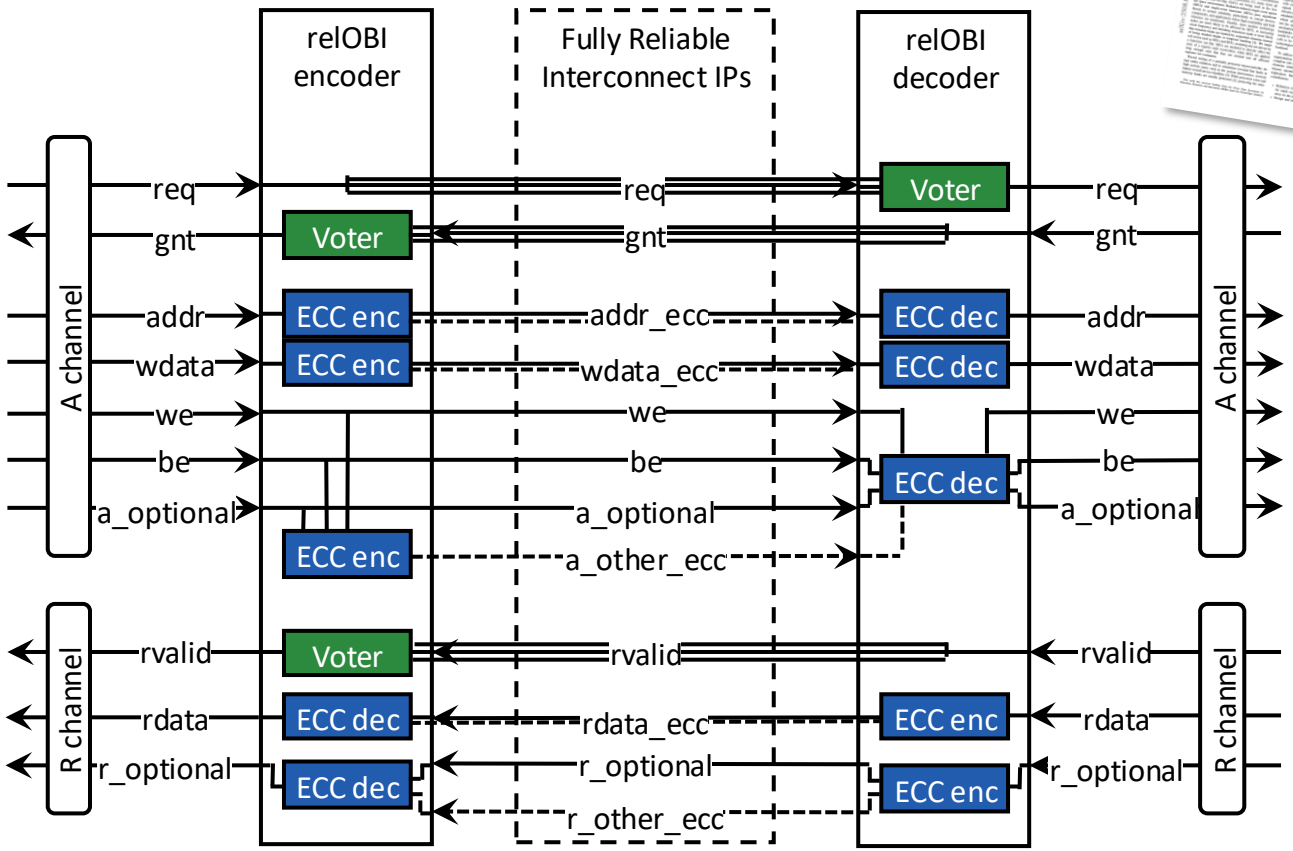
Adding fault-tolerance to croc

- **ECC Memory**
  - Efficient protection for static data
- **Lockstep Processor Cores**
  - Ensure correction for processing data
- **Reliable Interconnect**
  - relOBI for protecting transfers



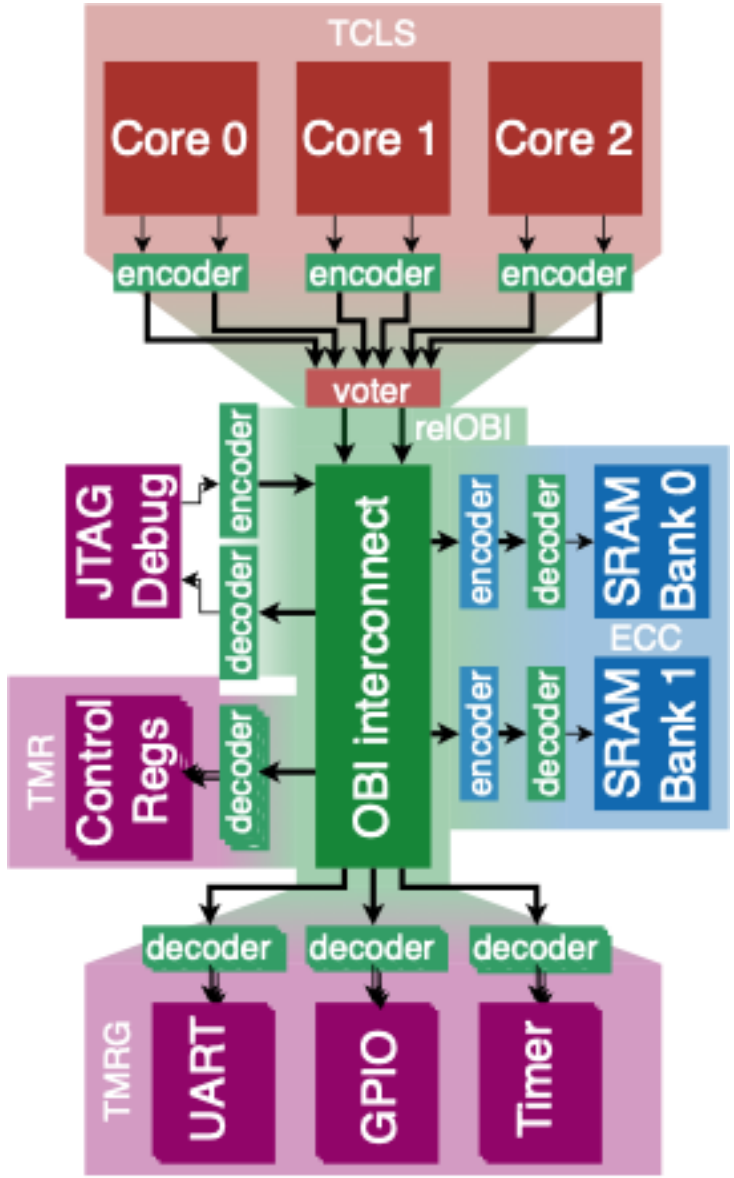
Reliable Interconnect Design

- **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



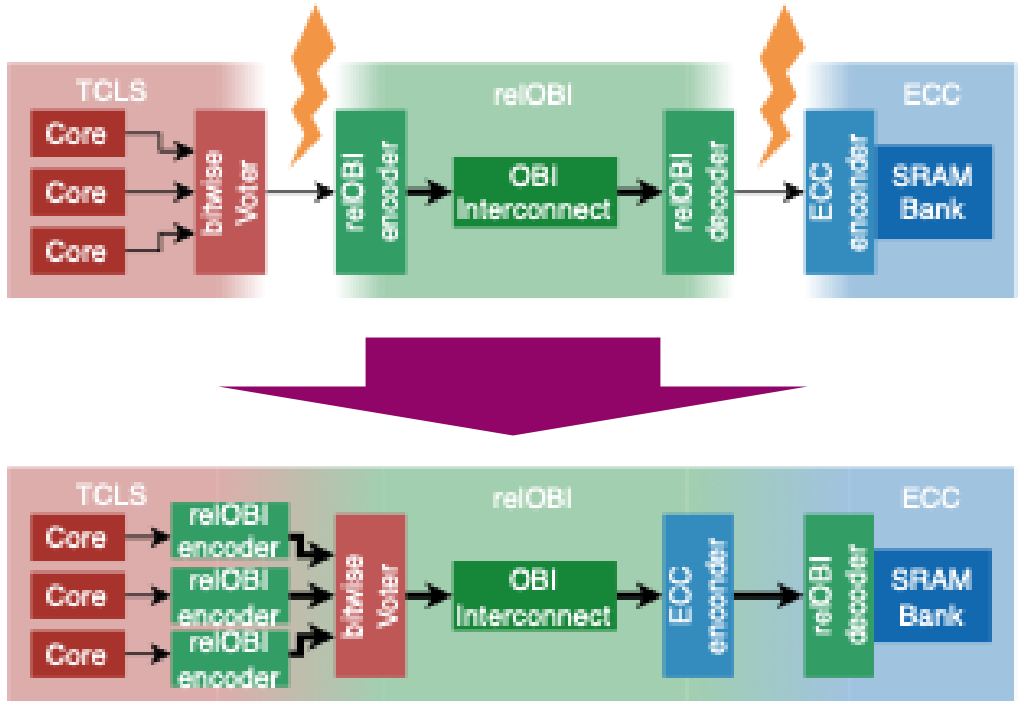
Adding fault-tolerance to croc

- **ECC Memory**
  - Efficient protection for static data
- **Lockstep Processor Cores**
  - Ensure correction for processing data
- **Reliable Interconnect**
  - relOBI for protecting transfers
- **Fine-grained TMR for Peripherals**
  - Classic approach for remaining components
  - Can be improved with bespoke approach



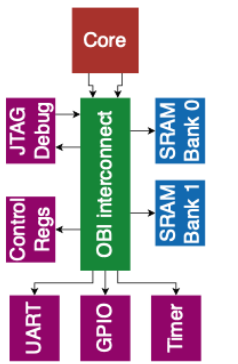
Who Checks the Checker?

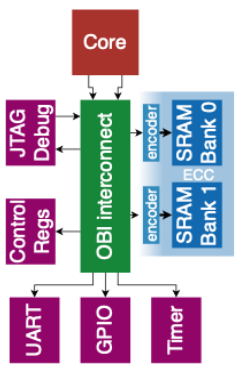
- **Main domains in system now protected**
  - Cores, memory, interconnect, peripherals
- **No protection in between!**
  - Connections vulnerable
  - Voters vulnerable
  - En-/decoders vulnerable
- **Idea: overlap the protection domains**
  - Encode Interconnect before voting cores
  - Encode for memory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

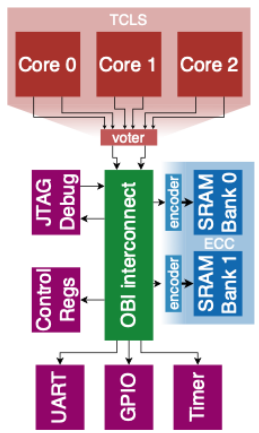


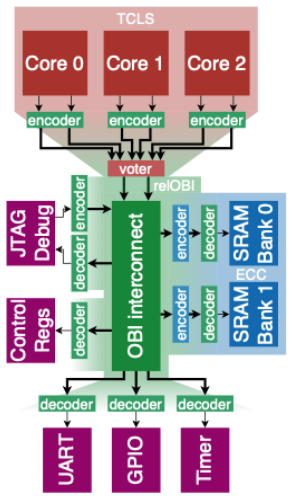
Evaluate Incremental Configurations

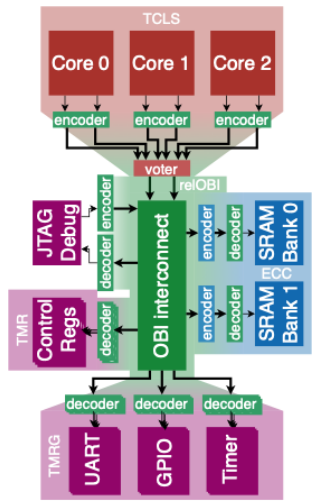
- **Iterative exploration, impact of each fault-tolerance addition**

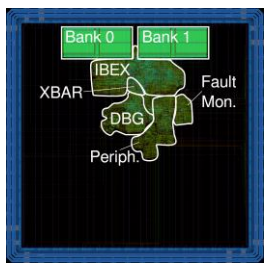


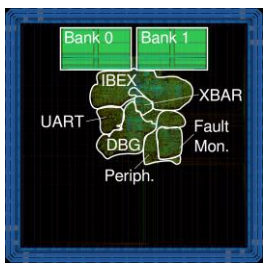


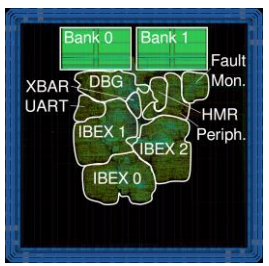


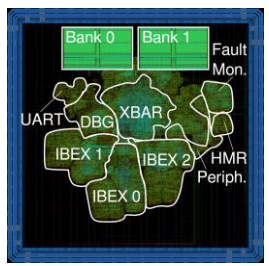


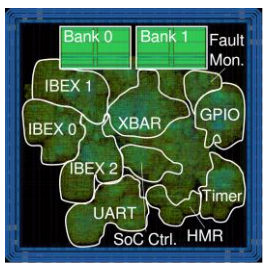












Evaluation & SotA comparison

- **Fault-tolerance analysis with fault injection**
  - VC_Z01X for fault simulations (single faults using VCS simulator)
  - Simulating representative workload (Coremark + peripheral access)
  - Testing multiple configurations
    - FLOP faults (RTL): bitflip in all stateholding elements (SEU)
    - FLOP faults (RTL): bitflip in stateholding elements excluding SRAM (SEU)
    - PRIM faults (netlist): bitflip in all logic excluding SRAM (SET)
- **SotA: Fine-grained triplication using TMRG**
  - Open-source tool
  - Triplicates the entire circuit
  - Voters insertion configured after every flipflop

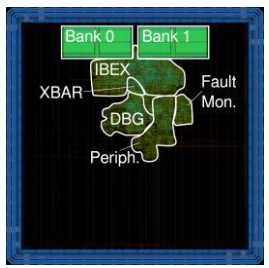
Implementation Results - Area

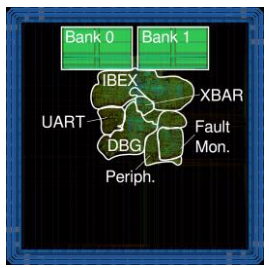
1.28x better than SotA!

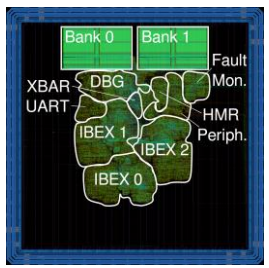
- **Implementation Overhead**
  - Synthesis with Yosys
    - Ensure protection methods are not optimized away
  - Place and Route with OpenRoad
  - Using IHP130 Open-source PDK

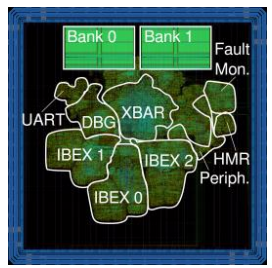
POST-LAYOUT IMPLEMENTATION RESULTS FOR C CFGS.

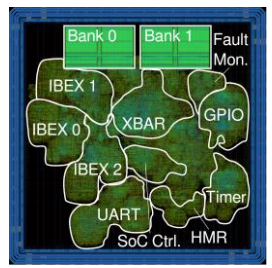
Cfg.	Used Core Area @ 60 MHz	Max. achieved Frequency	Regist	Area Overhead
0	1.057 mm^2 / 146 kGE	112 MHz	5113	1.00 ×
1	1.286 mm^2 / 177 kGE	108 MHz	5254	1.22 ×
2	1.758 mm^2 / 242 kGE	95 MHz	9358	1.66 ×
3	2.171 mm^2 / 299 kGE	65 MHz	10 175	2.05 ×
4	2.867 mm^2 / 395 kGE	62 MHz	13 799	2.71 ×
TMRG	3.676 mm^2 / 507 kGE	70 MHz	11 964	3.48 ×

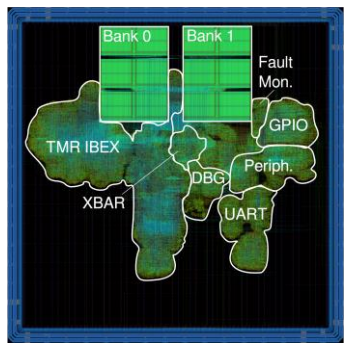




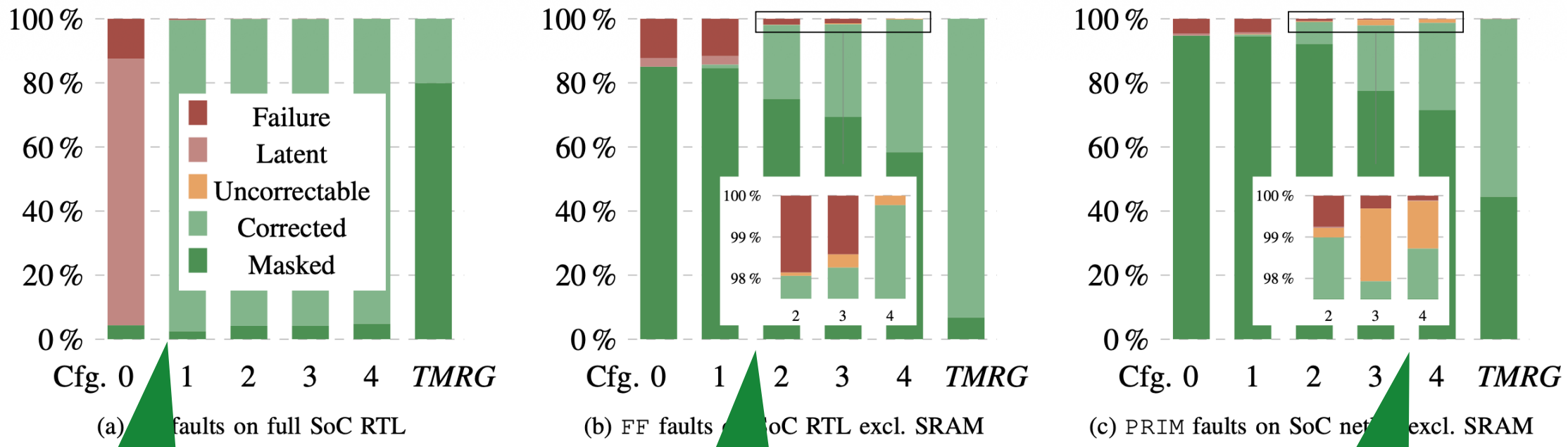








Fault Injection Results



(a) faults on full SoC RTL

(b) FF faults oC RTL excl. SRAM

(c) PRIM faults on SoC net excl. SRAM

Most faults injected in SRAM (>90%)

ECC SRAM addresses main problem (97%)

Second largest fault source in Cores

Large component (84% of remaining)

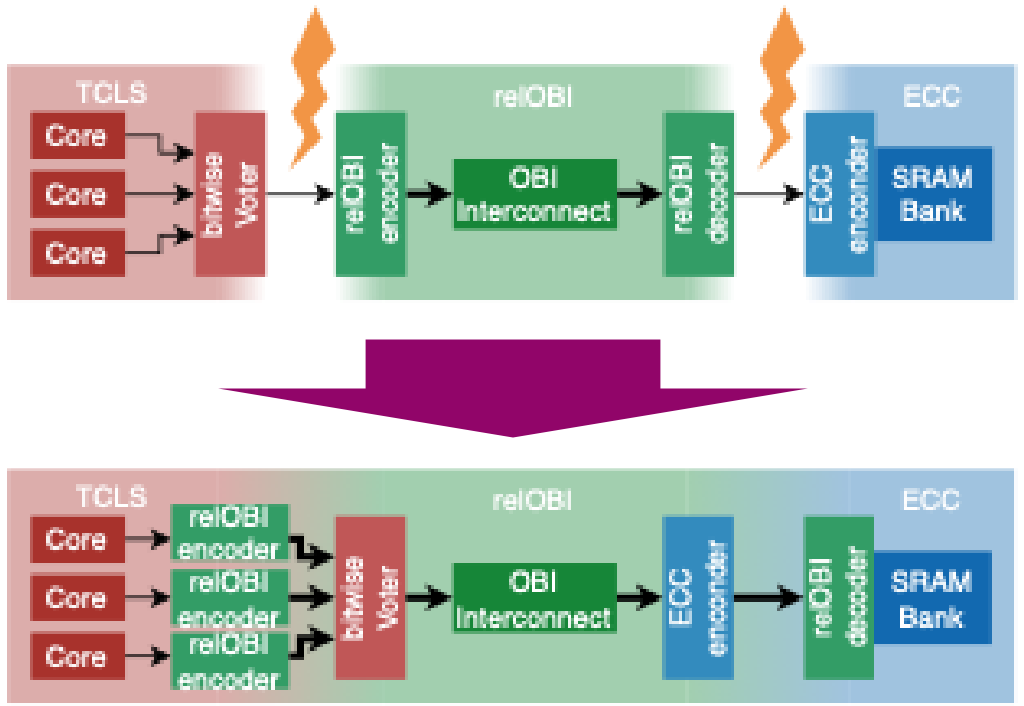
Final configuration addresses all issues

Similar to SotA fine-grained TMR

Who Checks the Checker?

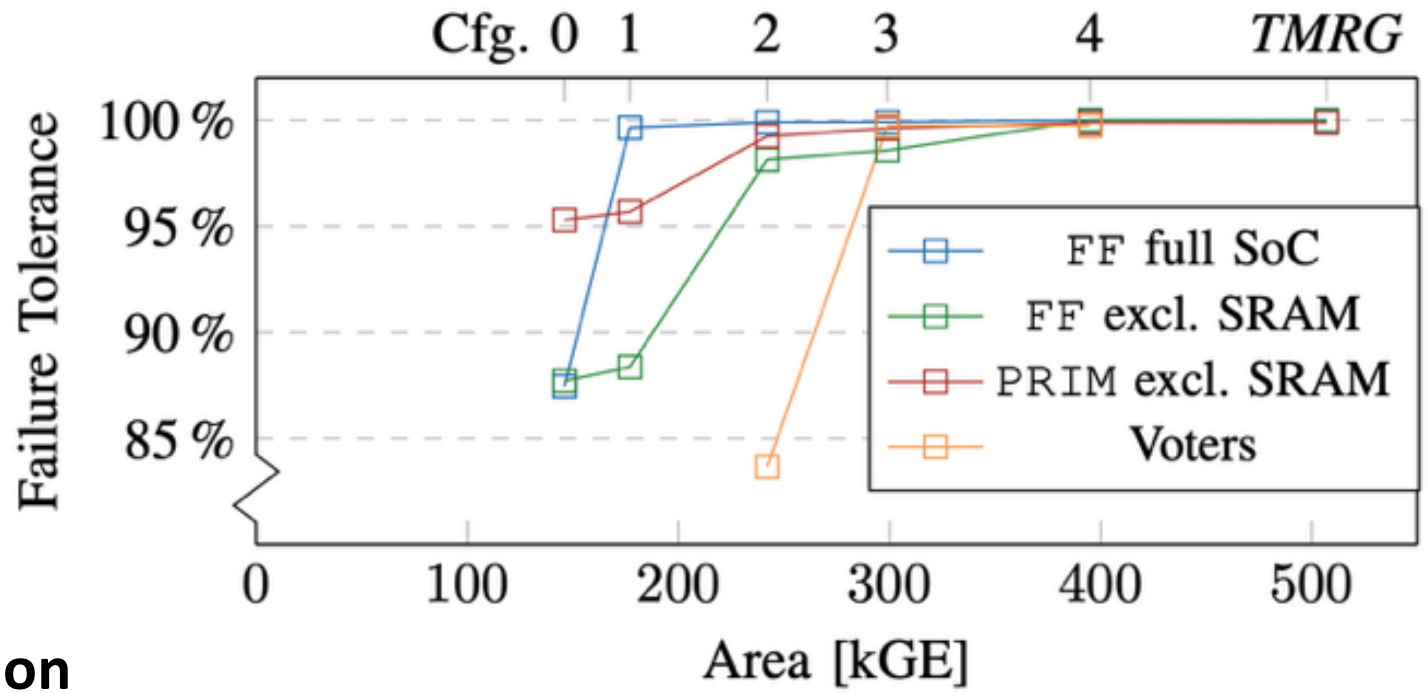
- **Does the checker overlap work?**
- **Targeted fault-injection in voter, encoder, decoder**

- **Non-overlapped design**
  - 16.33% failure
  - 1.59% latent faults
- **Overlapped design**
  - 44.44% corrected
  - 0.26% failure – all for an unused signal



Pareto Analysis

- **Comparing vulnerability and area across configurations**
- **Each configuration on pareto-front**
  - More tolerance with limited area overhead
- **Final fully-protected configuration better than SotA**



Conclusion

- **Fully protected RISC-V Microcontroller**
  - Tailored protection approaches
- **Overlapping protection mechanisms**
  - Ensure end-to-end fault-tolerance
- **Detailed implementation analysis**
  - 22% lower area overhead than SotA
- **Fault injection vulnerability analysis**
  - 97% of faults addressed with first method
  - Confirmed methods protect design
- **Incremental analysis for tailored tolerance levels**

Michael Rogenmoser michaero@iis.ee.ethz.ch
Philippe Sauter phsauter@iis.ee.ethz.ch
Chen Wu chenwu@iis.ee.ethz.ch
Angelo Garofalo agarofalo@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch