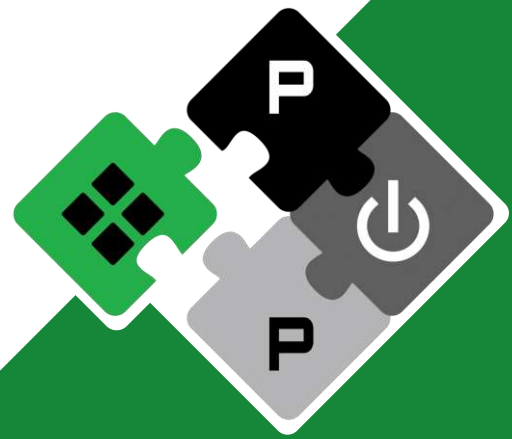
ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Security and safety using open-source hardware, The story so far.

Frank K. Gürkaynak kgf@iis.ee.ethz.ch



@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

PULP Platform

Open Source Hardware, the way it should be!

In 2020 at a workshop at HiPEAC I gave a talk

And at that time I said…

It has been 1621 days, what has changed?

- **Open source is still a tool to help tackle security issues**
  - It allows us to investigate the issues better
  - We can experiment with relevant systems easier
  - Share information more easily
  - Make changes and drive the developments openly (i.e. through RISC-V technical WGs)
- **But that was about security, how about safety and reliability**
  - Exactly the same arguments can be made for safety and reliability as well
  - In addition, several concerns/solutions for one has the potential to help the other
- **We no longer have to rely on vendor solutions, we can develop our own**

Unfortunately, this does not really solve the problems

Still need to develop our own solutions for safe, reliable & secure systems

Safety, Security and Reliability: Not same, but related

- **Security**
  - People with ill-intent can not circumvent the defined use of the system

Malicious people deliberately try to create errors

- **Reliability**
  - System continues to operate under environments that would cause errors

The environment randomly produces conditions that lead to errors

- **Safety**
  - The computing system operates as defined/expected

System continues to work correctly regardless of what happens

Solutions that work for one tend to help with the others as well

Knowing how things exactly work is vital

Security

- **From the "ZombieLoad" paper**

  From section 3.2, emphasis added for this presentation

  "While we identified some necessary building blocks to observe the leakage (cf. Section 5), we ***can only provide a hypothesis on why*** *the interaction of the building blocks leads to the observed leakage. As we could only observe data leakage on Intel CPUs, we assume that this is indeed an implementation issue (such as Meltdown) and not an issue with the underlying design (as with Spectre)."*

- **Closed implementations hide/abstract many secrets from users**

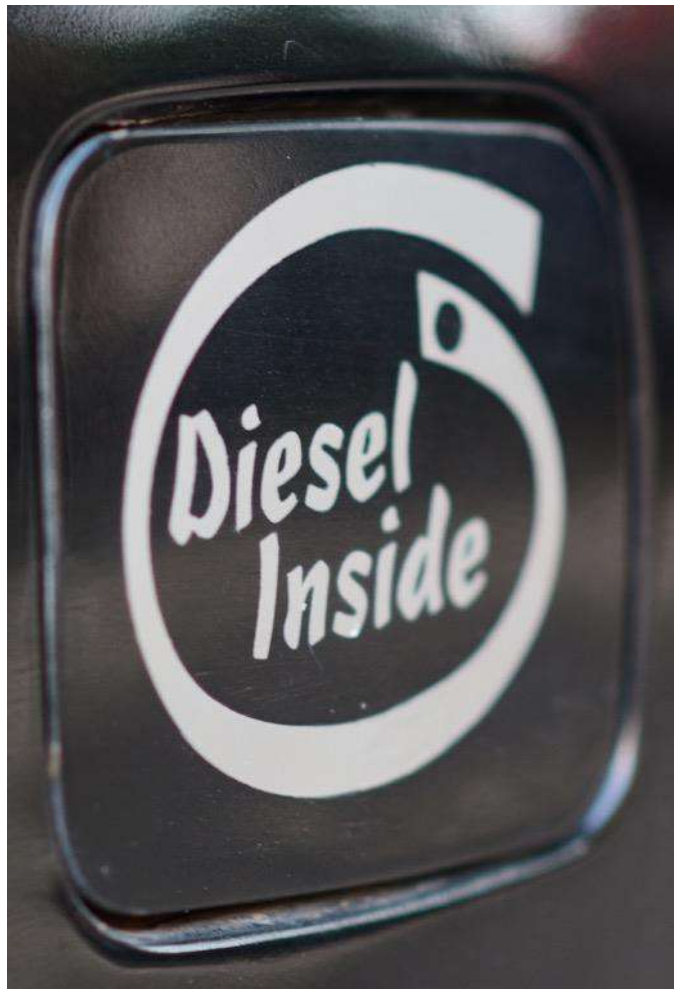
Ability to see inside and run experiments is vital for safety and security experts

M. Schwarz, M. Lipp, D. Moghimi, J. Van Bulck, J. Stecklina, T. Prescher, D. Gruss, "ZombieLoad: Cross-Privilege-Boundary Data Sampling", arXiv:1905.05726

How can open source HW help?

- **Know what is really inside**
- **More and independent verification of blocks**
- **Be able to experiment without constraints**
- **Share the information freely**
- **Fairer benchmarking**
- **Access to SoA systems to work on**

- **After all: Open-source SW has proven useful why should open source HW be different?**



Counterpoint: Just because it is open, bugs won't go away

- **Debian openssl -- predictable random number generator**

Security

  - On **2 May 2006**, a patch was applied to Debian sources to fix unitialized variables
  - These unitialized variables were intentionally used for the random numbers
  - … which generated seeds for (among others) RSA keys in the openSSL library
  - After the patch, there was very little entropy left. It was possible to guess (private) RSA keys
    - It took only 6 hours to generate all possible 4096 bit RSA keys using 32 Xeon cores.
  - The issue was discovered on **18 May 2008**. Almost exactly two years later

- **A serious security bug remained in plain sight for two years**
  - Although everything was open source and logged **nobody noticed!**

David Ahmed, "Two Years of Broken Crypto: Debian's Dress Rehearsal for a Global PKI Compromise", IEEE Security and Privacy, vol 6, pp 70-73, Sep/Oct 2008

RISC-V a free ISA to build SoA computer systems

- **It is FREE**
  - Everybody can build, sell, and make RISC-V cores available
  - The description is **FREE**, implementations can be FREE or proprietary
- **It is a modern design, no historical baggage**
  - Some common ISAs (ARM, Intel..) have been around for 20+ years
    Newer implementations, still need to be compatible to older designs.
  - RISC-V benefited form the mistakes made by others, cleaner design
  - Major design decisions have been properly motivated and explained
- **Reserved space for extensions, modular**
- **Open standard, you can help decide how it is developed**

Are RISC-V processors better than XYZ?

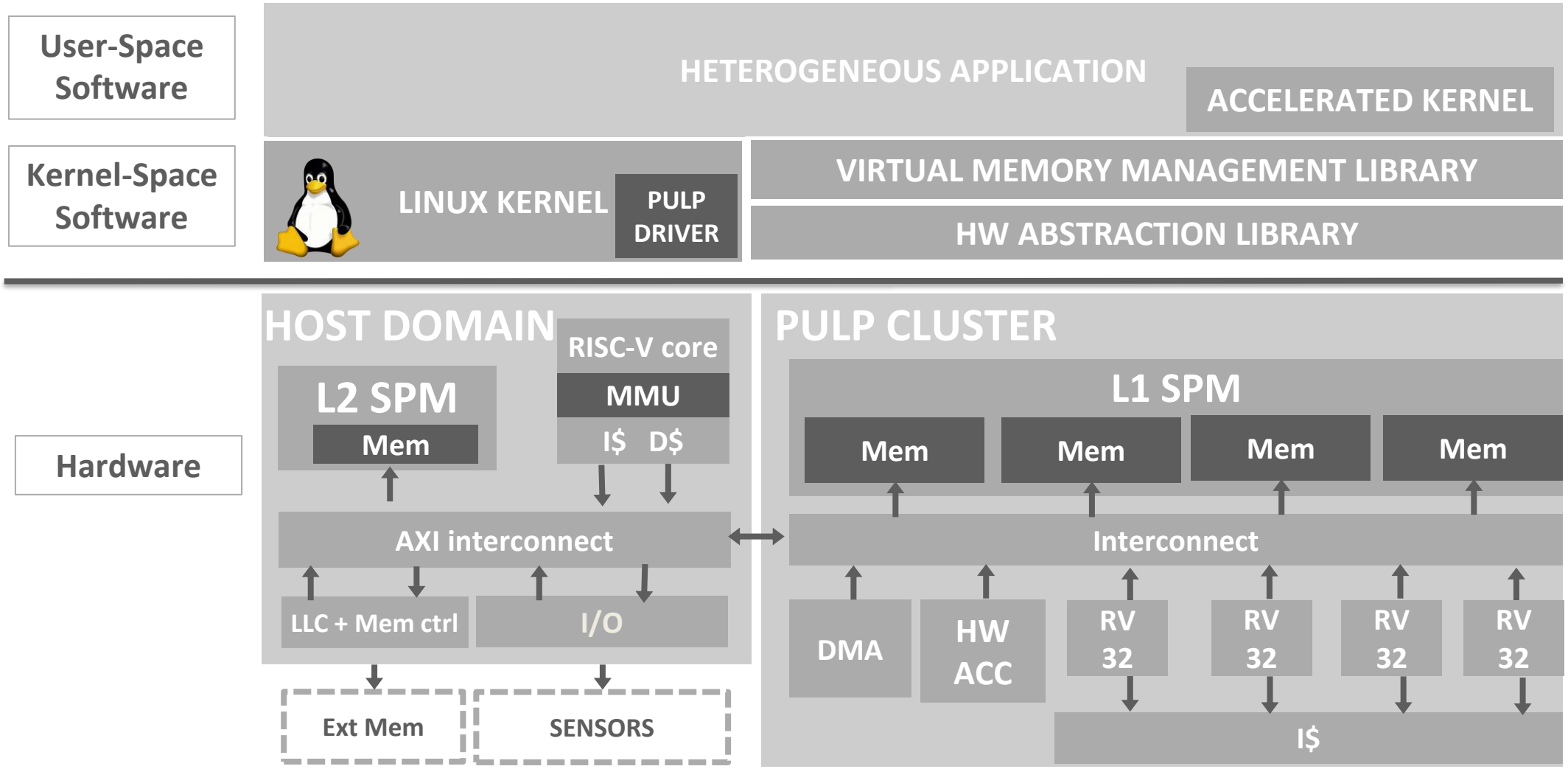
- **Actual performance depends on the implementation**
  - RISC-V does not specify implementation details (on purpose)
- **It is a modern design, should deliver comparable performance**
  - If implemented well, it should perform as good as other modern ISA implementations
  - In our experiments, we see no weaknesses when compared to other ISAs
  - It also is **not magically 2x better**
- **High-end processor performance is not much about ISA**
  - Implementation details like technology capabilities, memory hierarchy, pipelining, and power management are more important.

It is not that cores from XYZ are insecure, unreliable

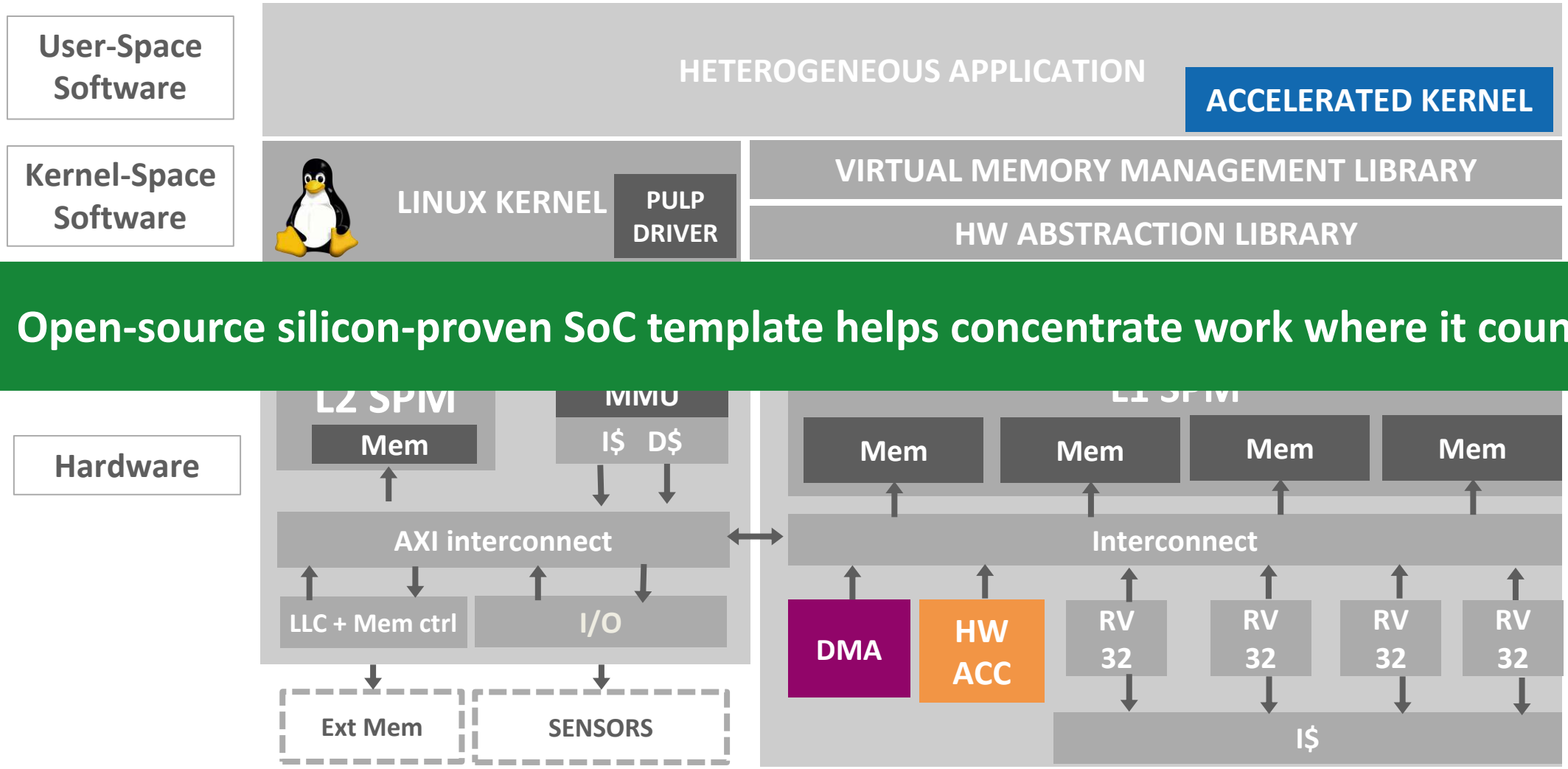
- **Most commercial processors have well thought out solutions**
  - In most likelihood better than anything we have in open-source hardware
- **But researchers do not always have access**
  - Work and insights can not be shared freely between researchers
  - Experimenting is limited, you work with what is given
  - Results and changes can not be verified independently

This is where we expect the most from open-source hardware: Access

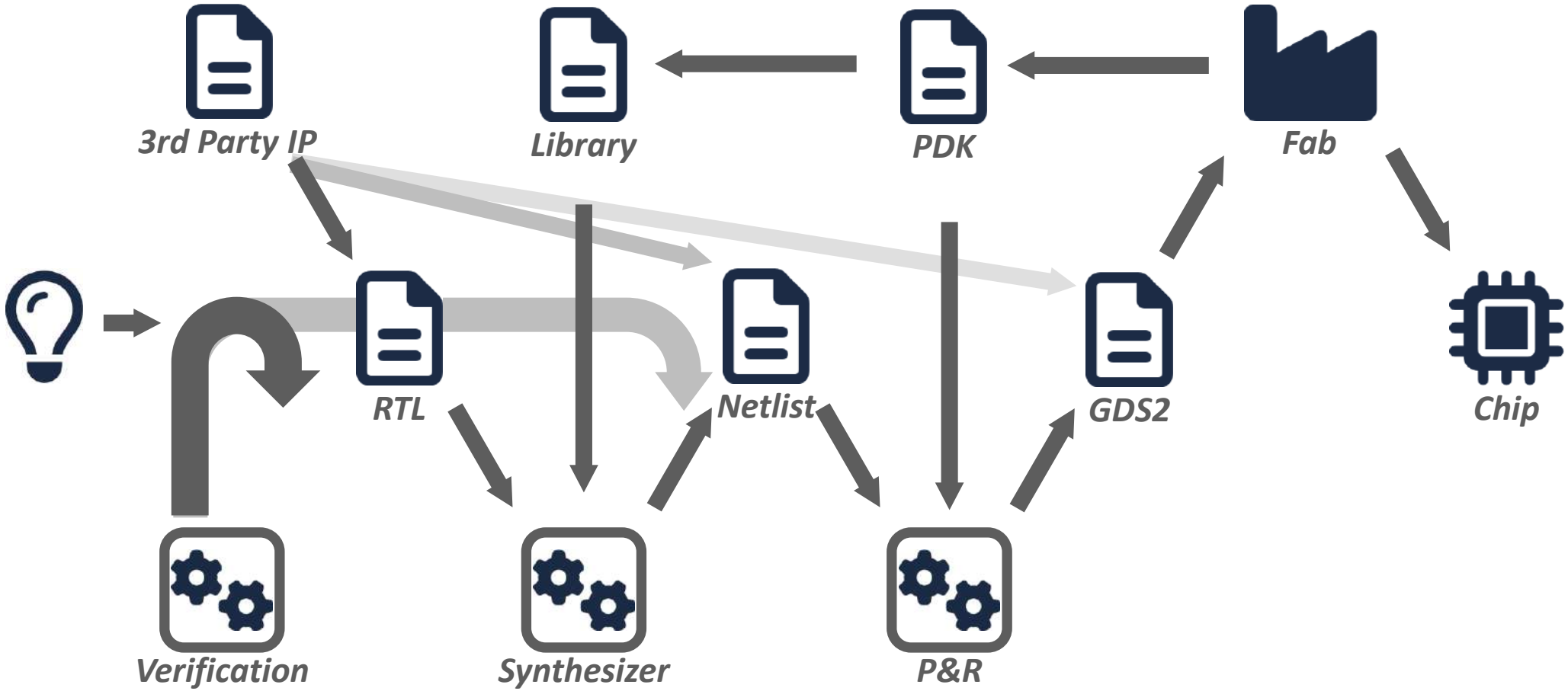
A processor core is but one part of a modern SoC



In a typical design, innovation is only in a limited scope

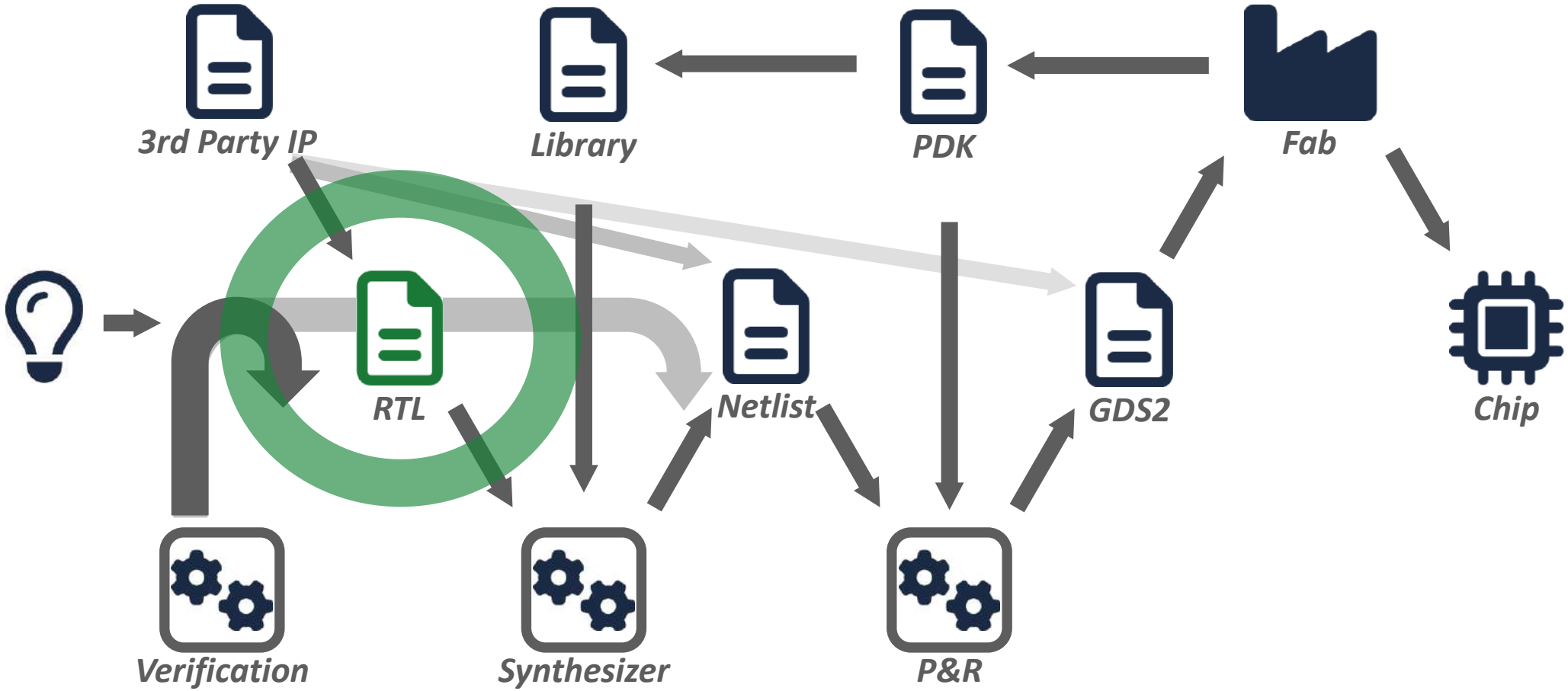


IC Design process is complicated with many stakeholders



Icons taken from free icons from fontawesome.com

Most of open source hardware is (currently) at RTL level



Icons taken from free icons from fontawesome.com

Reliability, Safety, Security are VERTICAL problems

Abstraction Layer	Example	Attacks	Security
Service	E-Voting servi	Legal challenges	
Users	Voters	Social engineering	
Application	Swiss Post	Bugs / backdoors in SW	
Algorithms / libraries	RSA / ope	Weaknesses in Algorithms	
Operating Systems	seL4	Privilege elevation	
Architecture	NXP - i.MX	Memory/cache attacks	
Microarchitecture	RISC-V/	Attacks on control flow	
Digital Electronics	Adders, ga	Side channel leakage	
Physics	Electrons, Qua n states	Environment	

Issues at ALL Levels

Open ce HW

RISC-V and open source HW are great to support..

- **Research in Reliability, Safety, Security**
  - In the last 5 years more and more work on RISC-V based systems
- **We need representative example systems to**
  - Investigate/expose flaws/problems
  - Find solutions to fix these issues
  - Properly understand the cost/benefit trade-offs of these solutions
- **But open-source and RISC-V alone are not sufficient**
  - Reliability, Safety and Security need solutions at all levels.
  - Open source can help but is not sufficient
    - False expectations can actually hinder development
  - There is still more work to be done

What has the PULP team done in these areas

Team of 100 people in ETH Zürich – University of Bologna

- **Research on open-source energy-efficient computing**

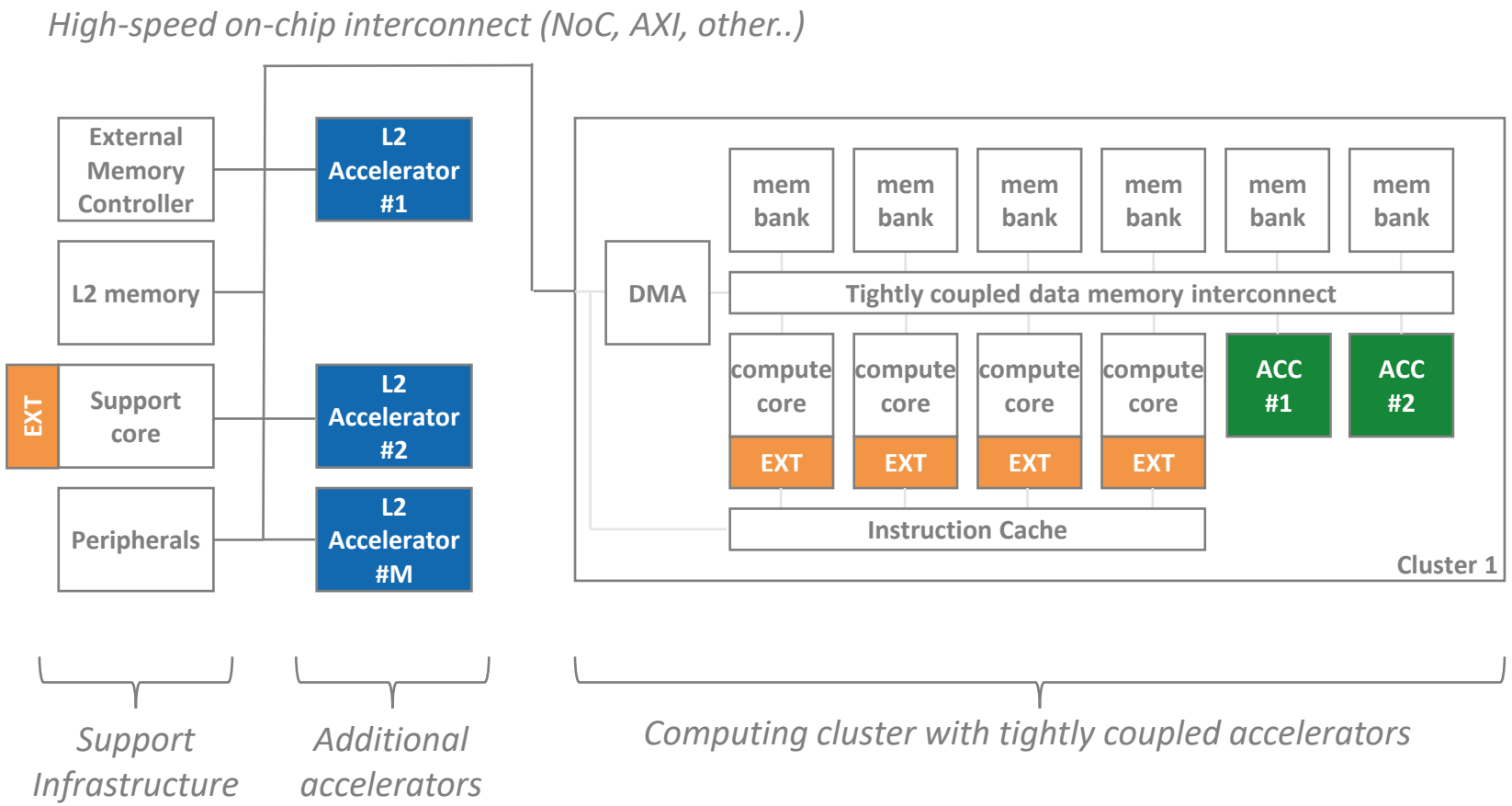
Our research focus: cluster-based many-core accelerators

Innovation factors

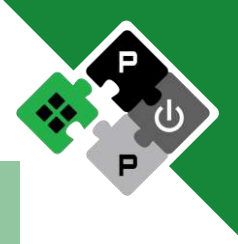
Extensions to processor cores
- Explore new extensions
- Efficient implementations

Shared-memory Accelerators
- Domain specific
- Local memory

Multiple computing clusters
- Communication
- Synchronization



We have created a sandbox to design System on Chips



RISC-V Cores and Vector Units

RI5CY *CV32E*	Zero R *Ibex*	Snitch	Spatz	Ariane *CVA6*	ARA
RV32	RV32	RV32	RVV	RV64	RVV

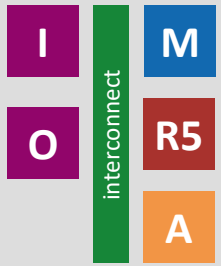
Peripherals

JTAG, SPI, UART, I2S, DMA, GPIO

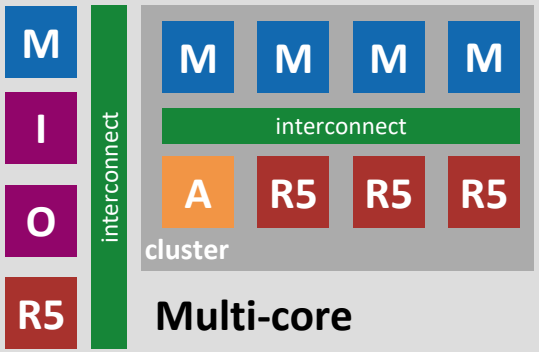
Interconnects

LIC, HCI, APB, FlooNoC, AXI4

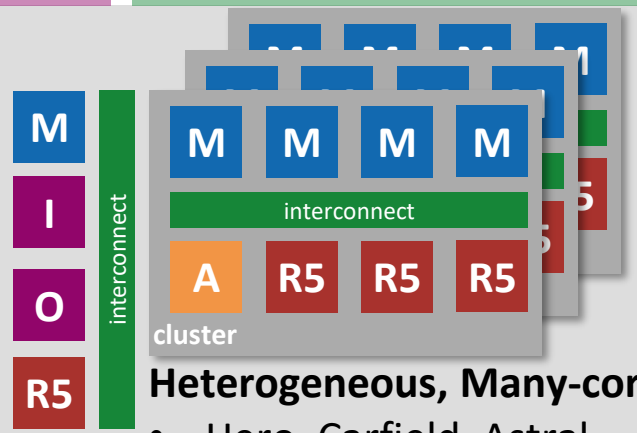
Platforms



Single core
- PULPino, PULPissimo
- Cheshire



Multi-core
- OpenPULP
- ControlPULP



Heterogeneous, Many-core
- Hero, Carfield, Astral
- Occamy, Mempool

IOT → **HPC**

Accelerators and ISA extensions

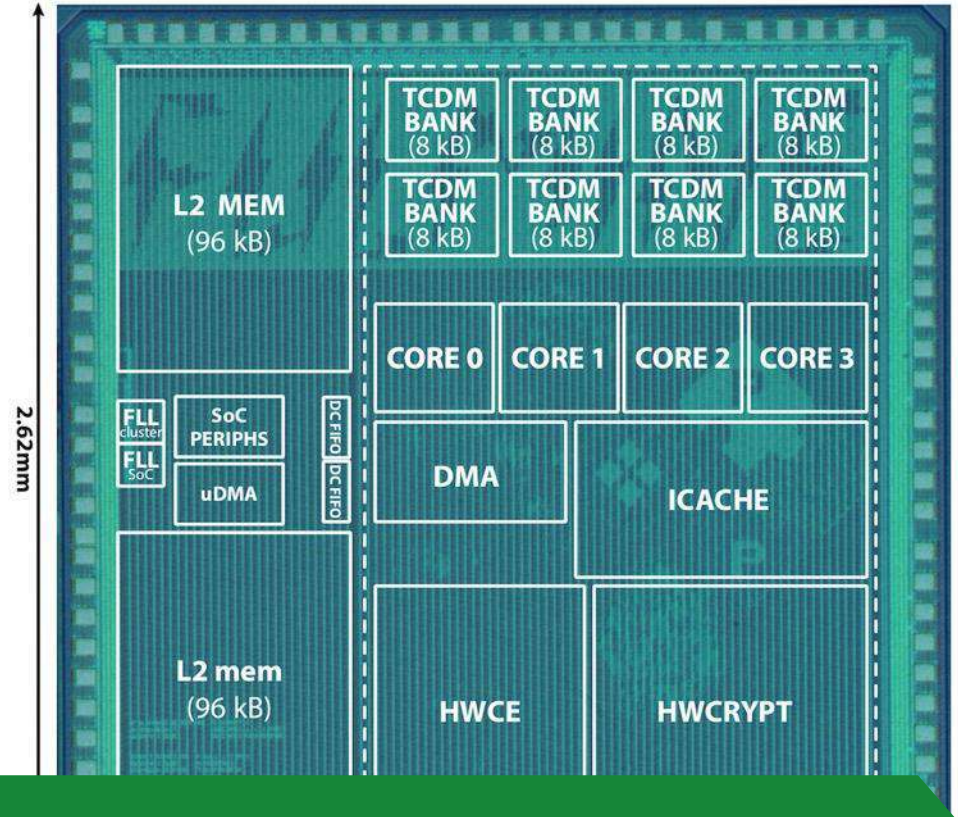
XpulpNN, ITA, RBE, NEUREKA, FFT, REDMULE (…sor)

https://github.com/pulp-platform

Accelerating cryptographic functions

Security

- **Key challenge: I/O bandwidth**
  - Not so difficult to design fast crypto HW
  - Need to match the rest of the system
  - Bandwidth to memory/bus the issue
- **Fulmine (UMC65)**
  - 2 TCDM ports 64bits/cycle
  - AES unit (2 rounds/cycle)
    - 0.38 cpb (8 kByte block); Intel Xeon AES-NI 1.18 cpb
    - @0.8V and 84 MHz, 1.76 Gbit/s, 120 pJ per byte (chip)
  - Also SHA3 unit and other accelerators



Let's look at how the accelerator works a bit more in detail

F. Conti et al., "An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 9, pp. 2481-2494, Sept. 2017.

For a typical AES accelerator (encryption)

Security

- **You need**
  - 128 bits of input data (plaintext)
  - 10 rounds of operation (AES-128)
- **This data will need**
  - 128 bits of key (AES-128)
- **And produce**
  - 128 bit of output data (ciphertext)
- **Output dependence**
  - Some cryptographic modes
    - CFB, OFB, CBC

- **What are your options?**
  - How many bits you process per step
    - 1/2/4/8/16/32/64/128
    - Let's take max:128, one round every step
  - How many rounds you process per clock
    - 1/2/5/10
    - Let's take: Two rounds every clock cycle
- **What is the I/O?**
  - Assuming the key is taken care of
  - We need 256 bits every 5 cycles or 51.2 bits/cycle

You can easily get 10Gb/s as long as you can transfer 50+ bits/cycle

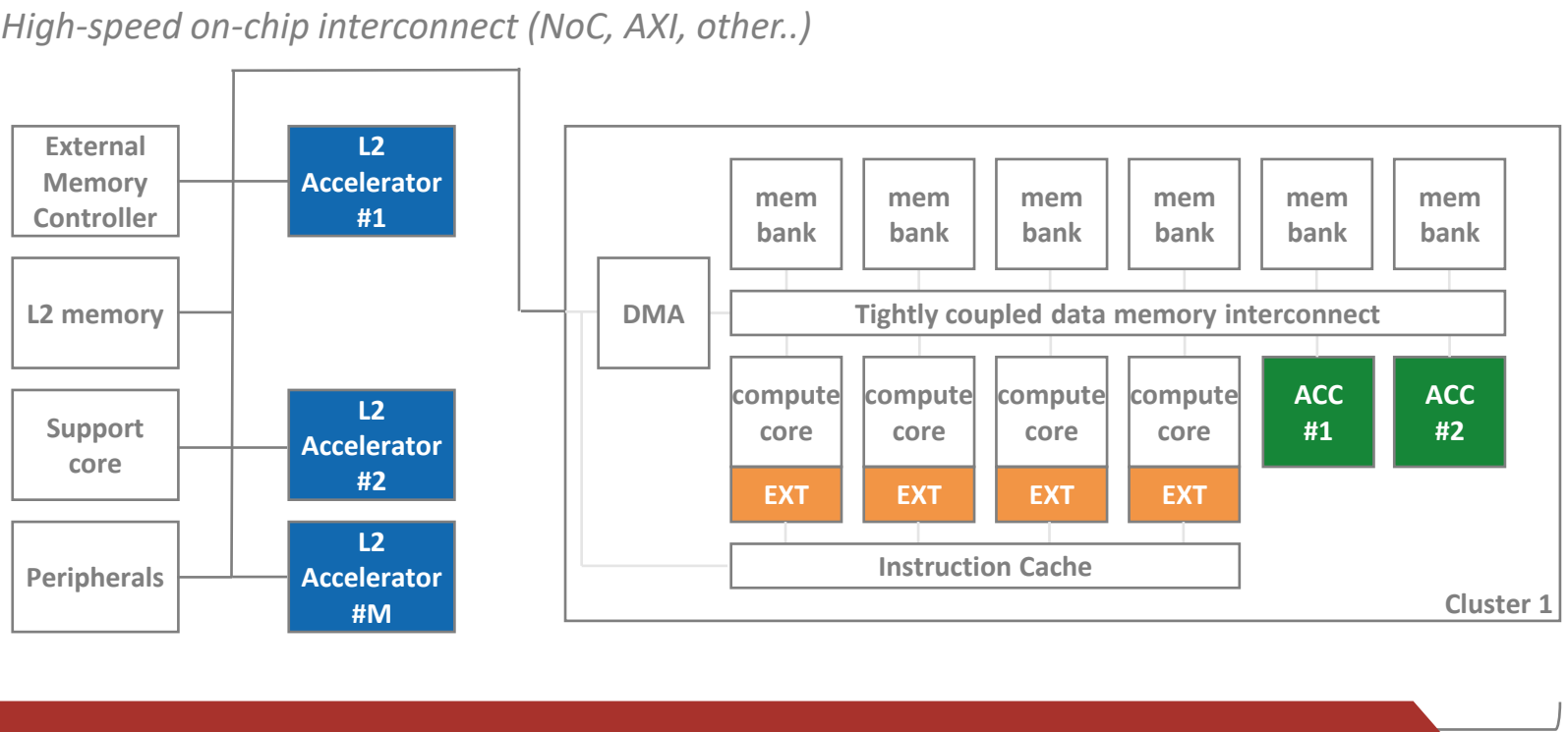
Notice that it is the I/O bandwidth that is your issue

How you can accelerate

Extensions to processor cores
- Limited by the processor bit-width
- Vector could help, but is large
- People mostly want this

Shared-memory Accelerators
- Can adapt memory I/O to your needs
- This is extremely efficient

Independent Accelerators
- Communication (Bus I/O)
- Synchronization
- Compute units are fairly small

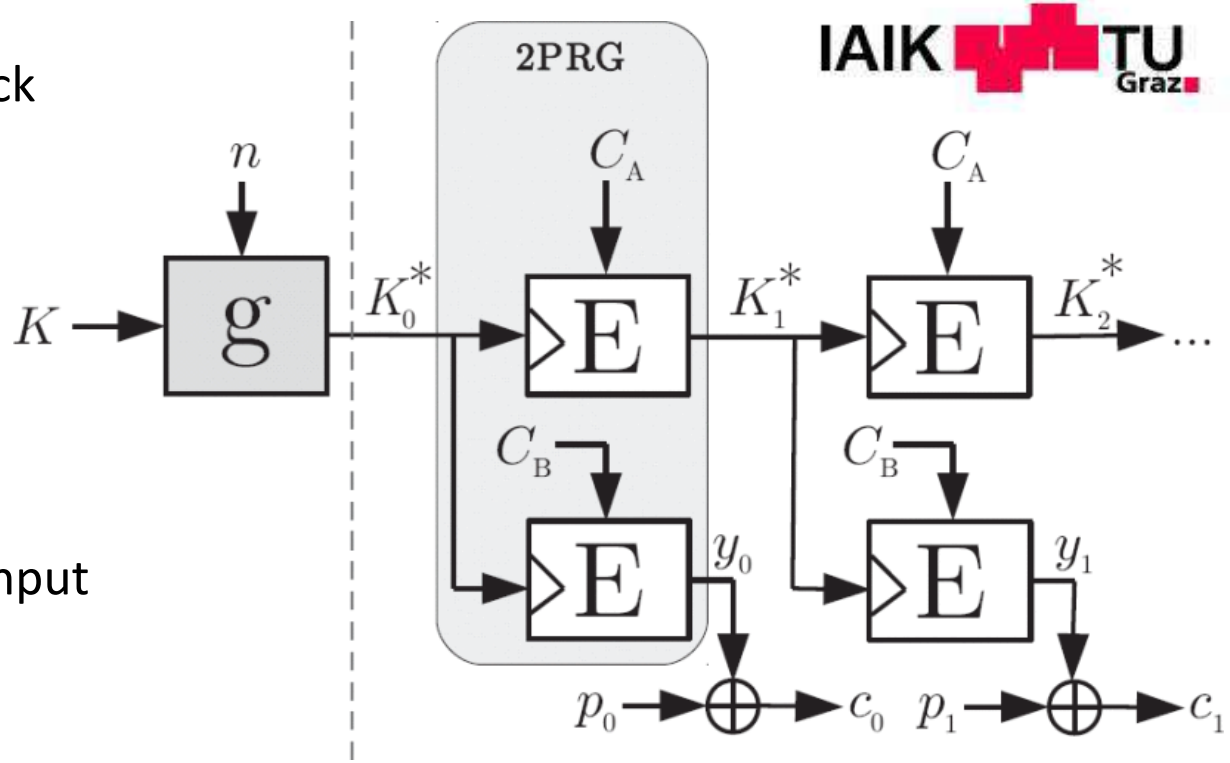


For most cases, shared memory accelerators are very efficient, but overlooked

Leakage resilient cryptography

Security

- **Reduce Attack surface**
  - A new key (**K***) is generated per data block
- **Encryption example (2PRG)**
  - **E** function is AES
  - **g** finite field multiplication with 1st order masking
  - Max throughput 5.29 Gbit/s @ 256 MHz
  - Needs **2x Block ciphers** for same throughput
- **Strong side channel resilience within IoT Power budget**
  - Implemented and tested in **Fulmine** (last slide)



Robert Schilling, Thomas Unterluggauer, Stefan Mangard, Frank Gürkaynak, Michael Muehlberghuber, Luca Benini, "High-Speed ASIC Implementations of Leakage-Resilient Cryptography", DATE 2018

Attacks against the control flow

Security

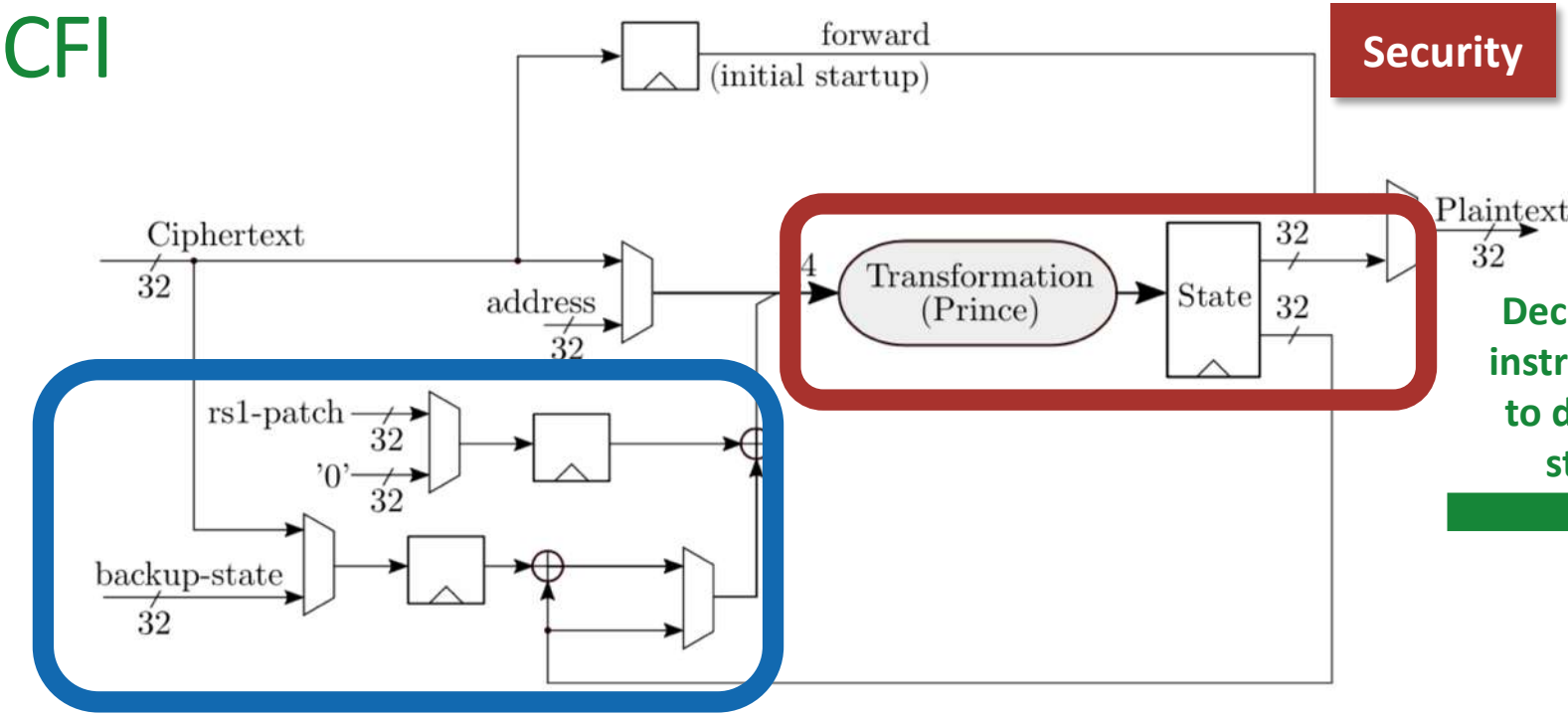
- **Can be realized in both HW and SW**
  - A successful attack on a processor changes the order of executed instructions
  - Can be used to execute malicious code or jump over security checks
- **HW attacks can be realized by controlling environment**
  - Clock or voltage glitches
  - Injecting electromagnetic pulses
- **Small IoT devices more vulnerable**
  - They operate in potentially hostile environment
  - Have less resources to withstand attacks from a capable adversary

Sponge based CFI

Security

Encrypted instructions from memory

Ability to patch the state for branches/jumps



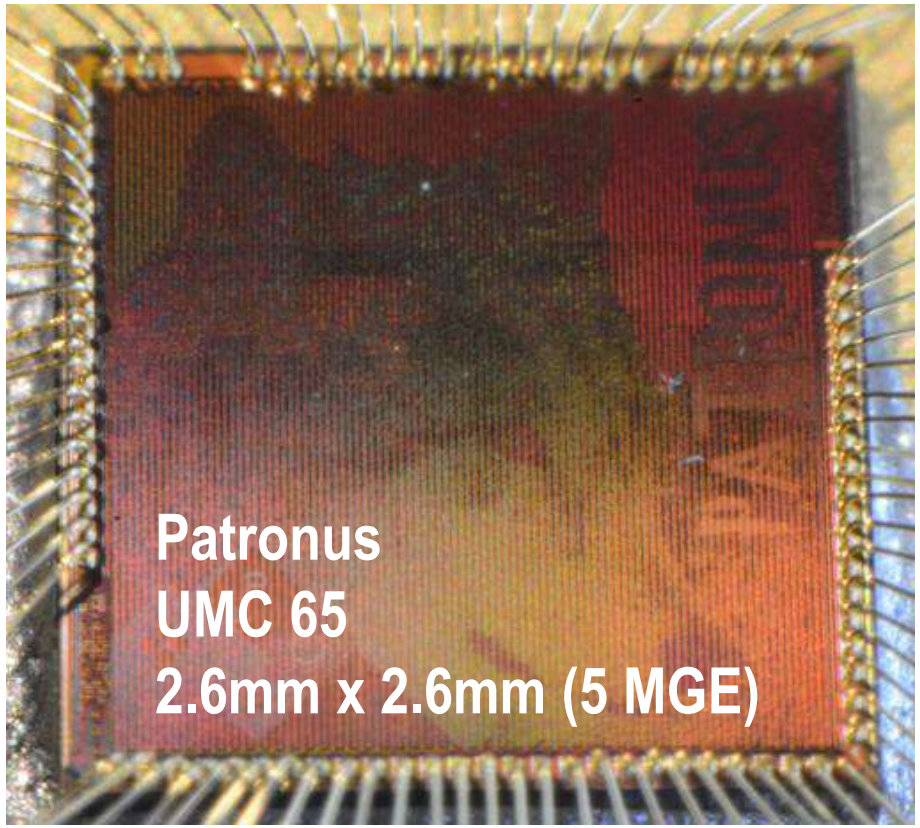
Decrypted instructions to decode stage

- **Sponge based construction to decrypt instructions**
  - AEE Light with 32 bit **state** and 32 bit capacity in APE mode
  - Used **Prince** for permutation allowing single cycle execution
- **Attacker has to change instructions and state at the same time**

IAIK TU Graz

Patronus: RISC-V system with CFI

- **Additional pipeline stage in Ibex for decryption**
  - LLVM based compilation flow
- **Only 25-35% power/area overhead**
- **Additional instructions for branches added as instruction set extensions**
- **About 10% runtime overhead due to patches**
- **Probability of illegal instruction trap when instruction altered**
  - 91.51% within 1 cycle
  - 99.19% within 2 cycles
  - **99.95%** within 3 cycles



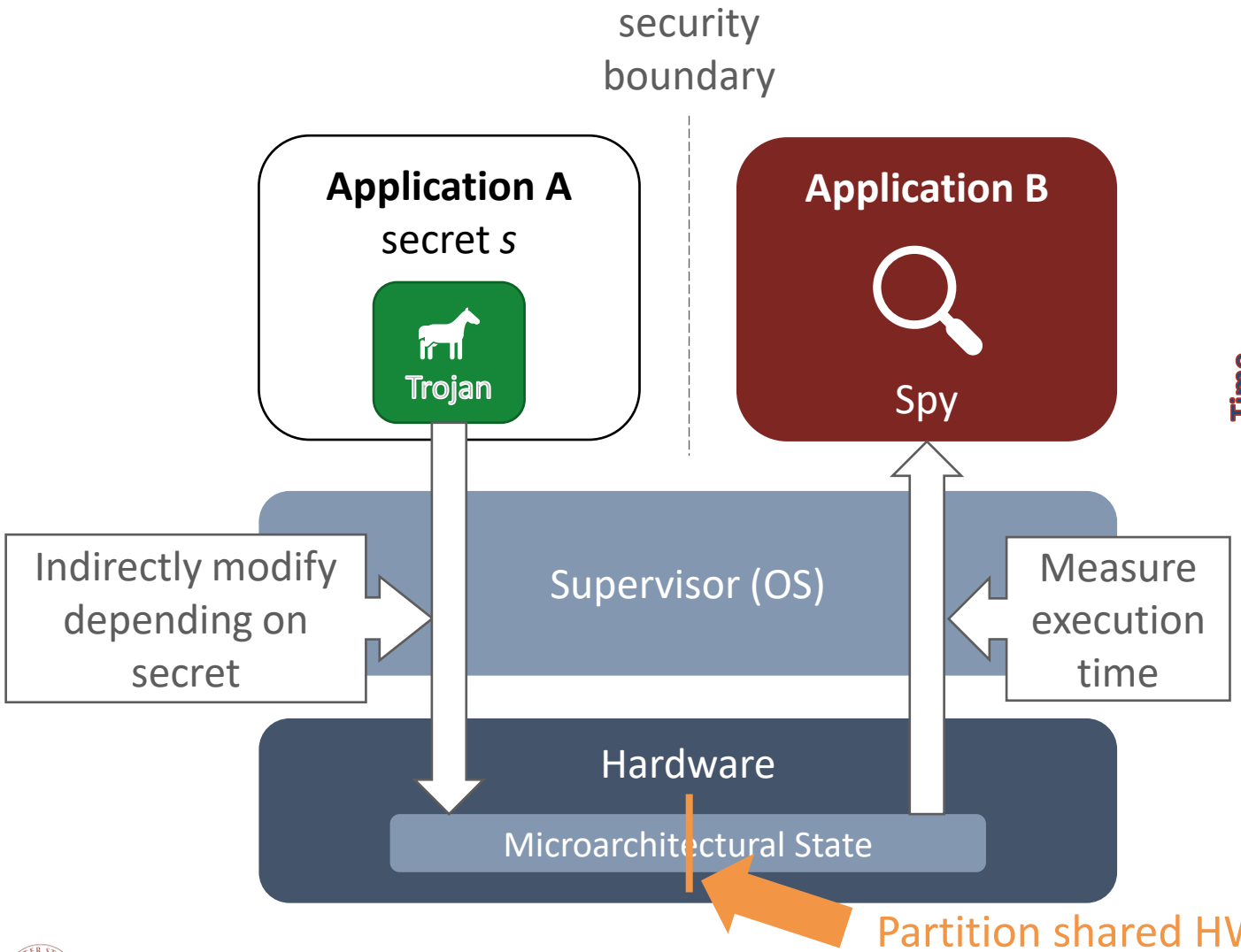
Mario Werner, Thomas Unterluggauer, David Schaffenrath, Stefan Mangard, "Sponge-Based Control-Flow Protection for IoT devices", 2018 IEEE European Symposium on Security and Privacy

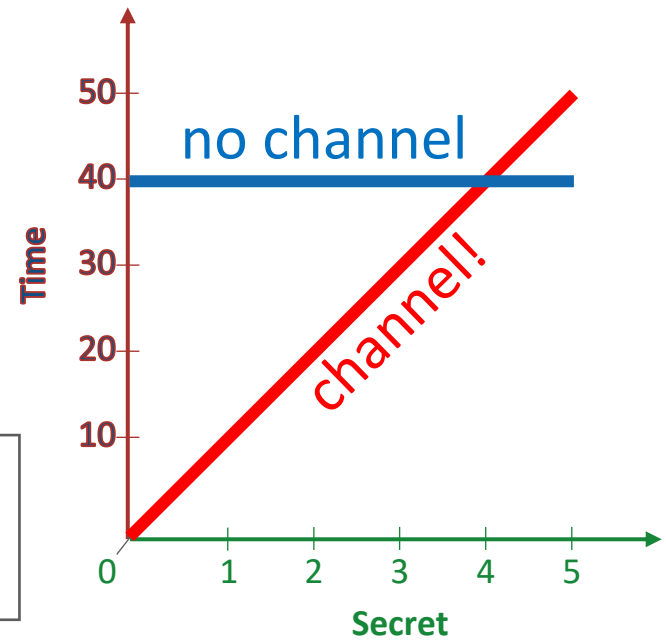
Securing covert channels

- **Several attacks are based on passing information between tasks**

- **Covert channels are used to pass information between tasks**
  - Most channels are based on state of hardware that is retained between task switches
  - Branch prediction history, caches, reorder buffers
- **Attacks can be mitigated by 'securing' covert channels**

Timing Channel



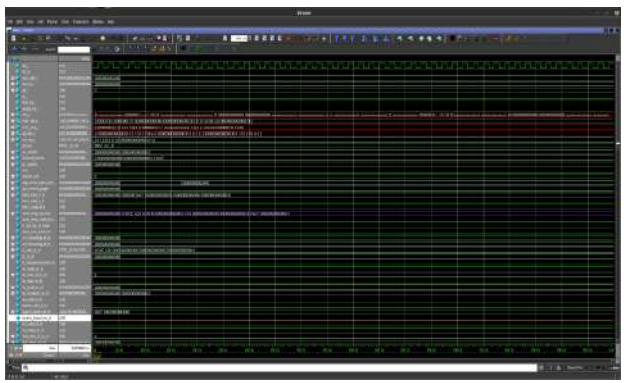


CVA6 (Ariane)

Security

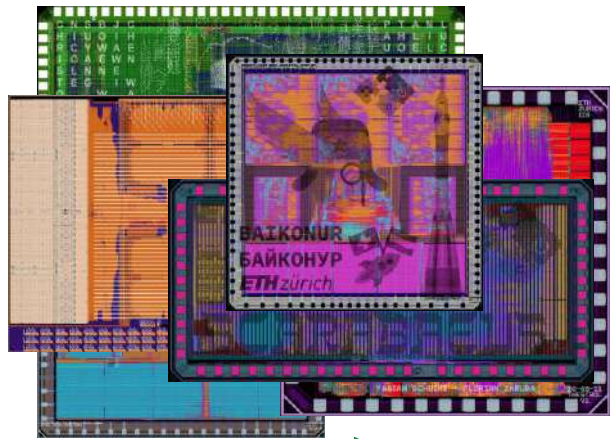
- **Open-source 64-bit application-class RISC-V processor**
- **Boots Linux (or seL4)**
- **Developed by PULP team at ETH**
- **Now owned and maintained by OpenHW Group**
- **Widely used in academia and industry**

RTL Simulation



FPGA Emulation

ASIC



speed, turn-around time, cost

Timing Channels on CVA6

Security

Application

Channel bench

+

OS

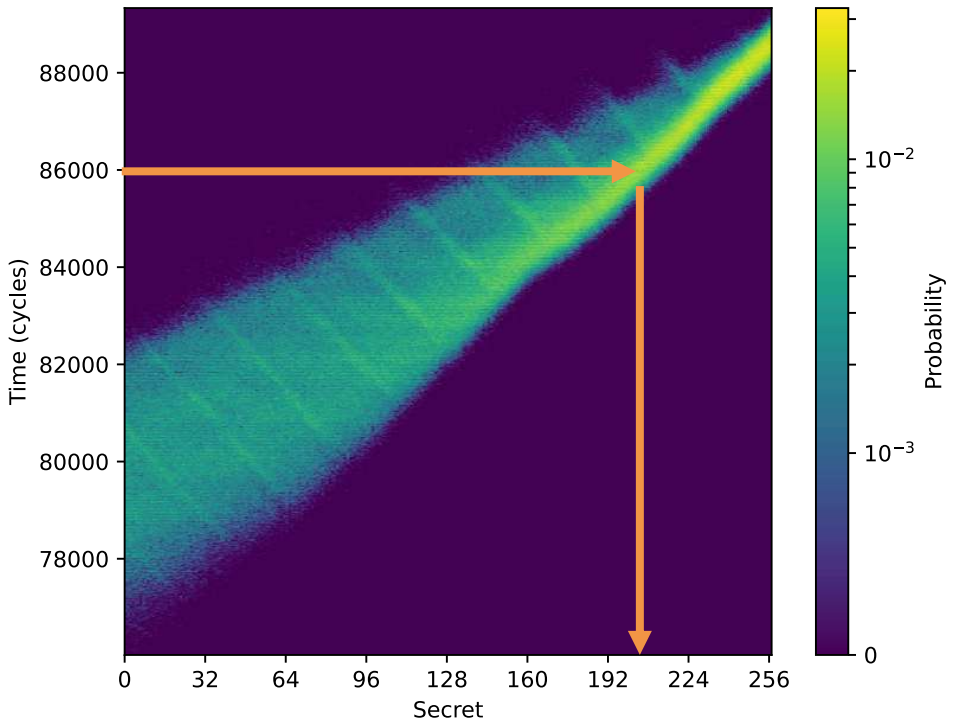
seL4

Security. Performance. Proof.

+

Hardware Platform

$N = 10^6$



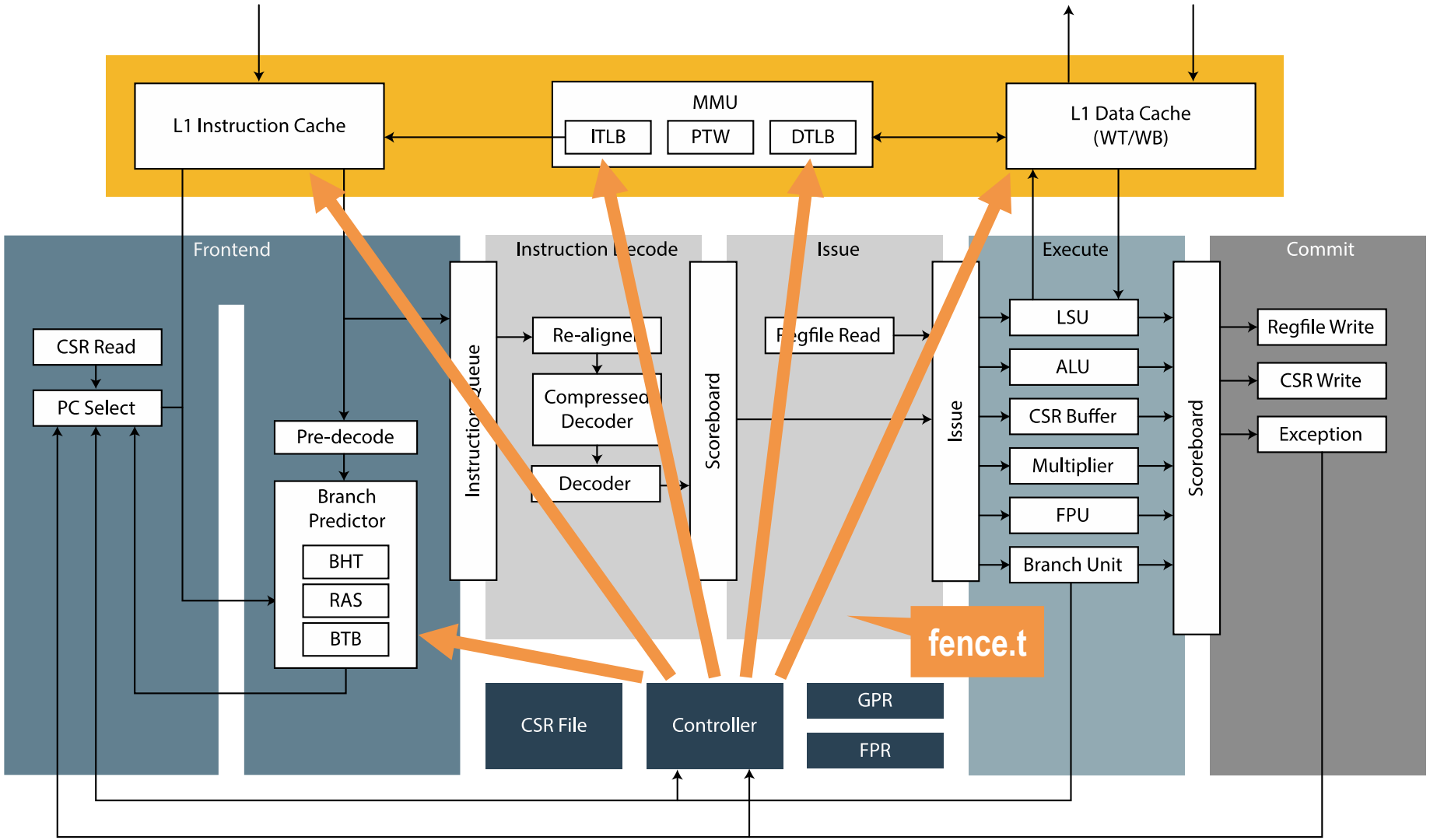
$M = 1667$ mb

$M_0 = 0.5$ mb

Characterises noise

fence.t instruction to clear microarchitectural state

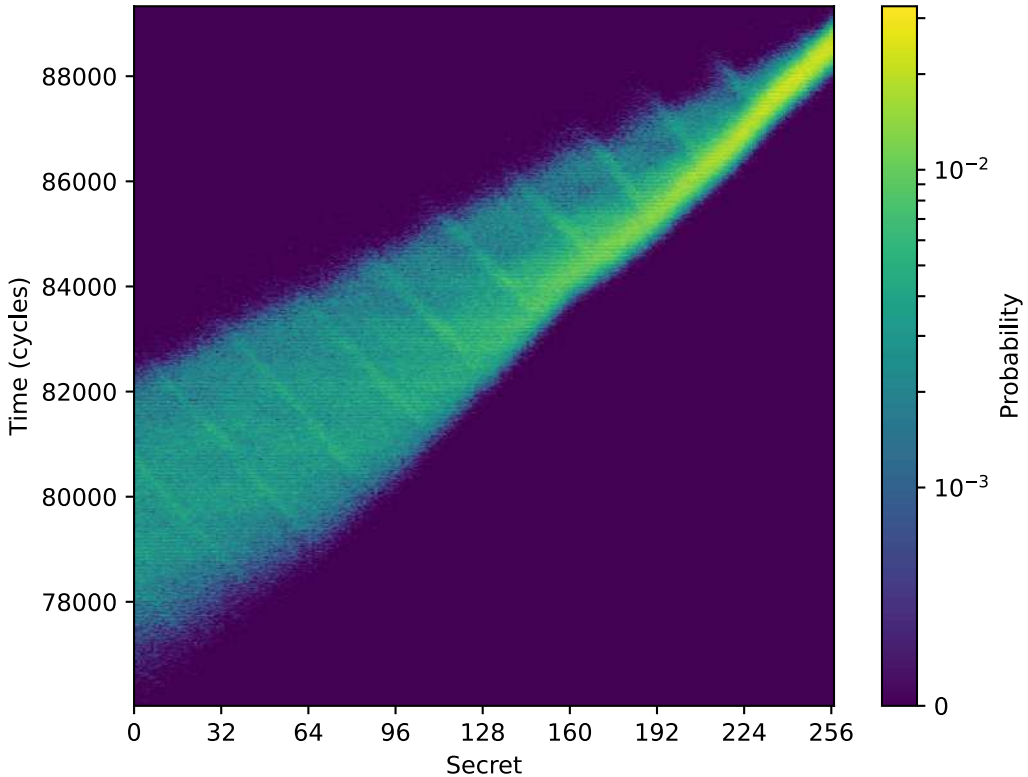
Security



How the Microreset affects timing channels

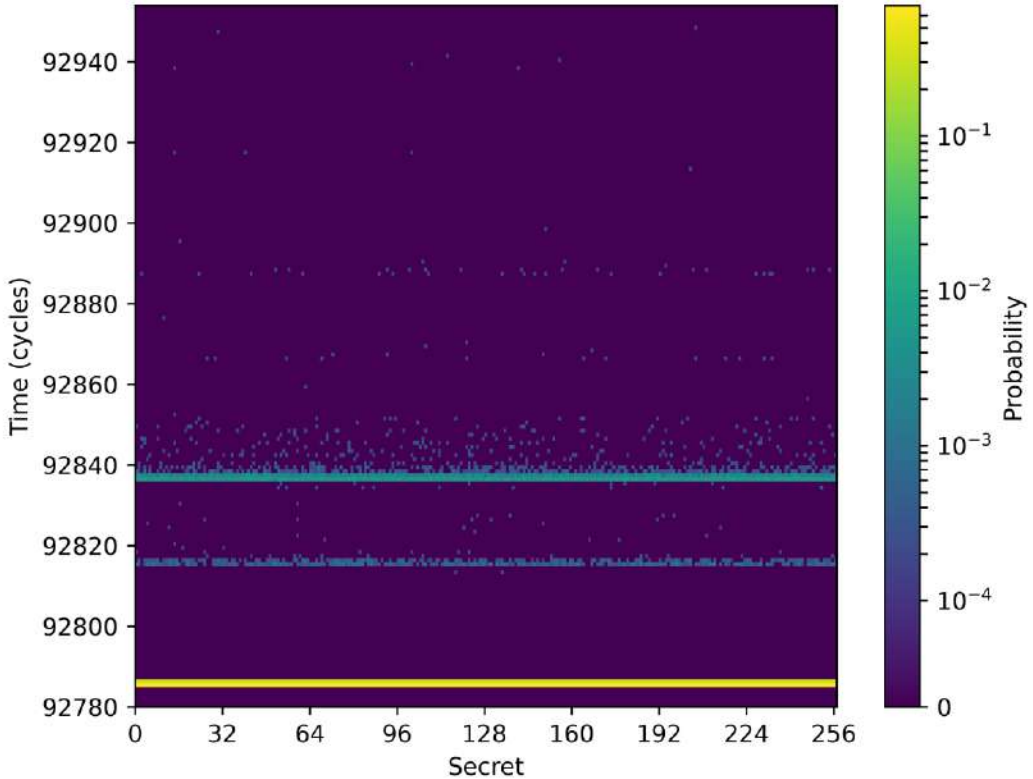
Security

Unmitigated



N = 10^6, M = 1667.3 mb, M_0 = 0.5 mb

fence.t (Microreset)



N = 10^6, M = 21.7 mb, M_0 = 27.8 mb

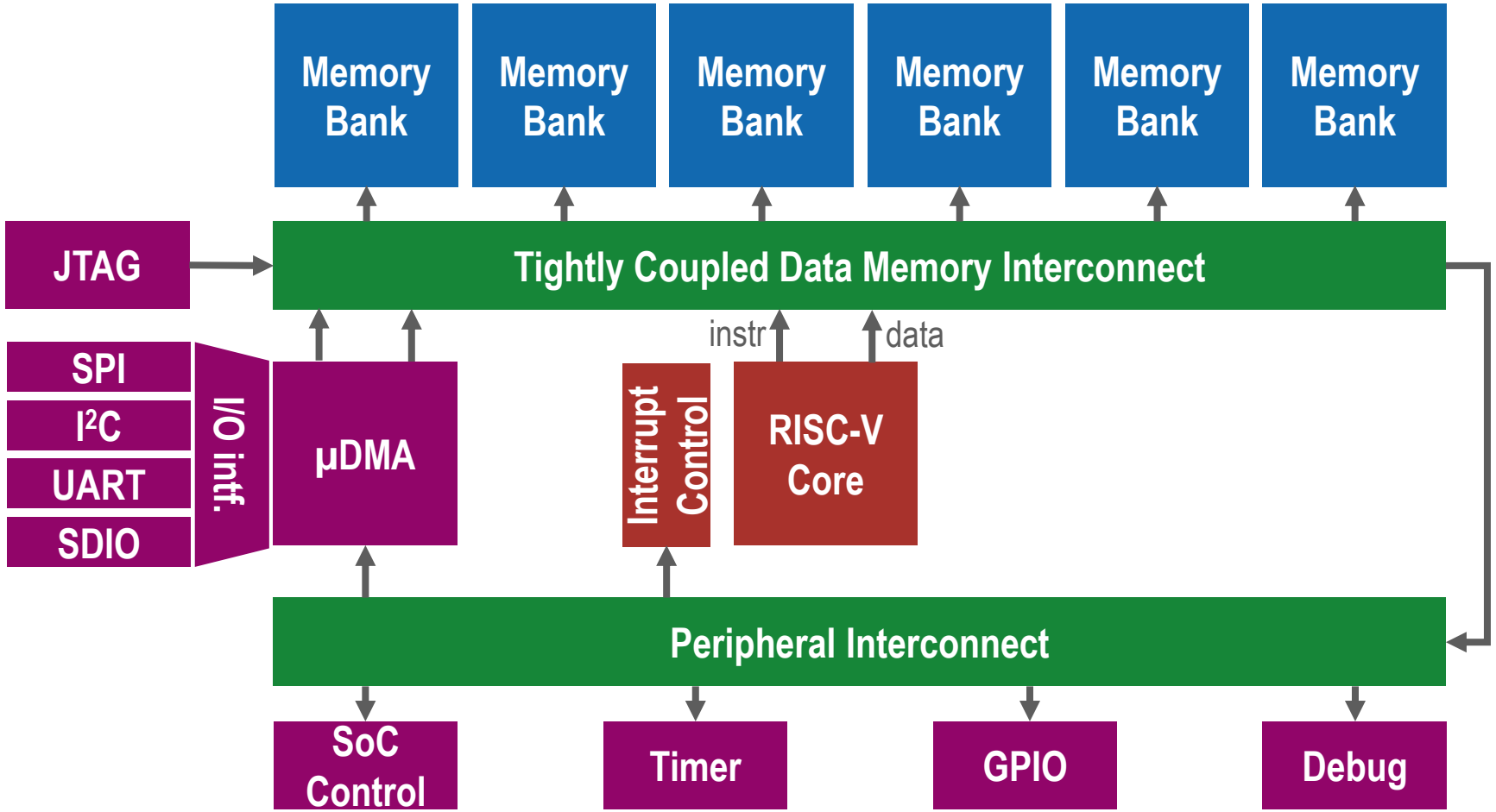
When particles play tricks on you

Reliability

- **There are environments that are not friendly to Integrated Circuits**
  - Space
  - High-energy physics experiments
- **Particles can cause all kinds of issues**
  - Flip bits in memory / FFs
  - Affect timing and corrupt next state calculations
  - Destroy part of the circuit
- **More random (unlike for example fault attacks)**
  - It is about statistics, and statistics can help us
  - Particle hits affect also circuits in the vicinity (spatial separation can help)
  - Redundancy is the key approach. Find a balance between redundancy and performance

Take our 32bit Micro controller - PULPissimo

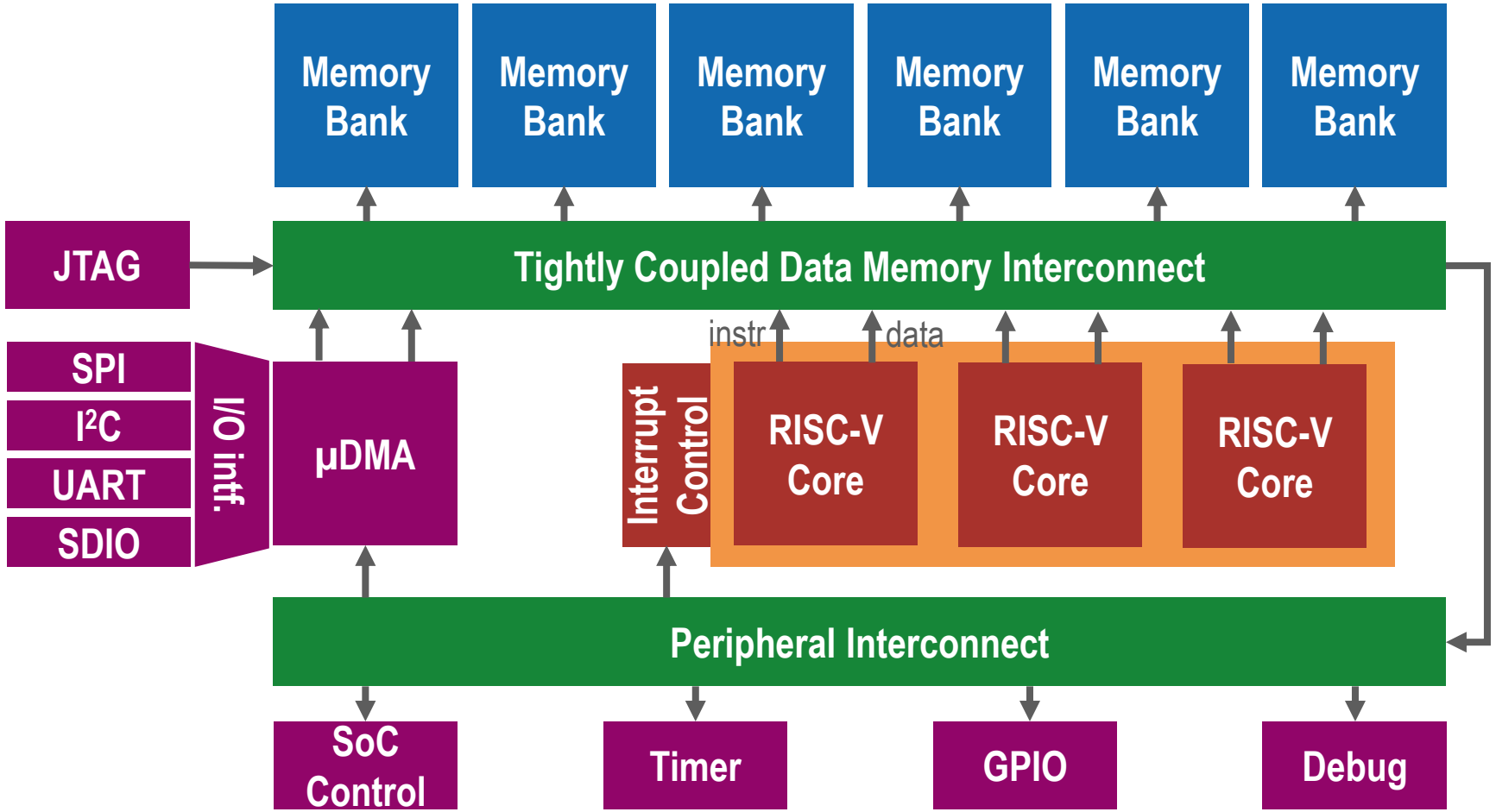
Reliability



https://github.com/pulp-platform/pulpissimo

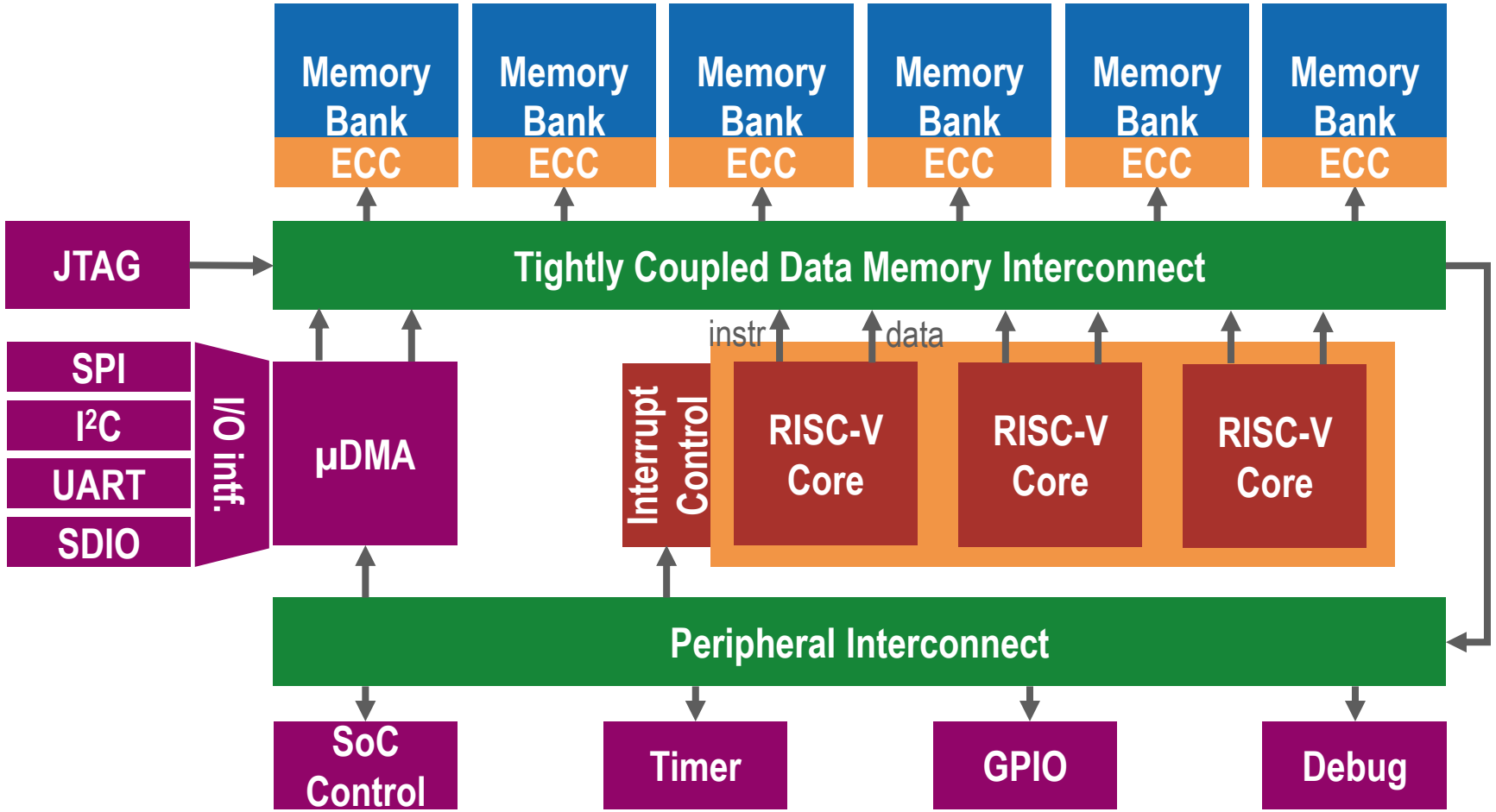
Trikarenos – PULPissimo with Reliability

Reliability



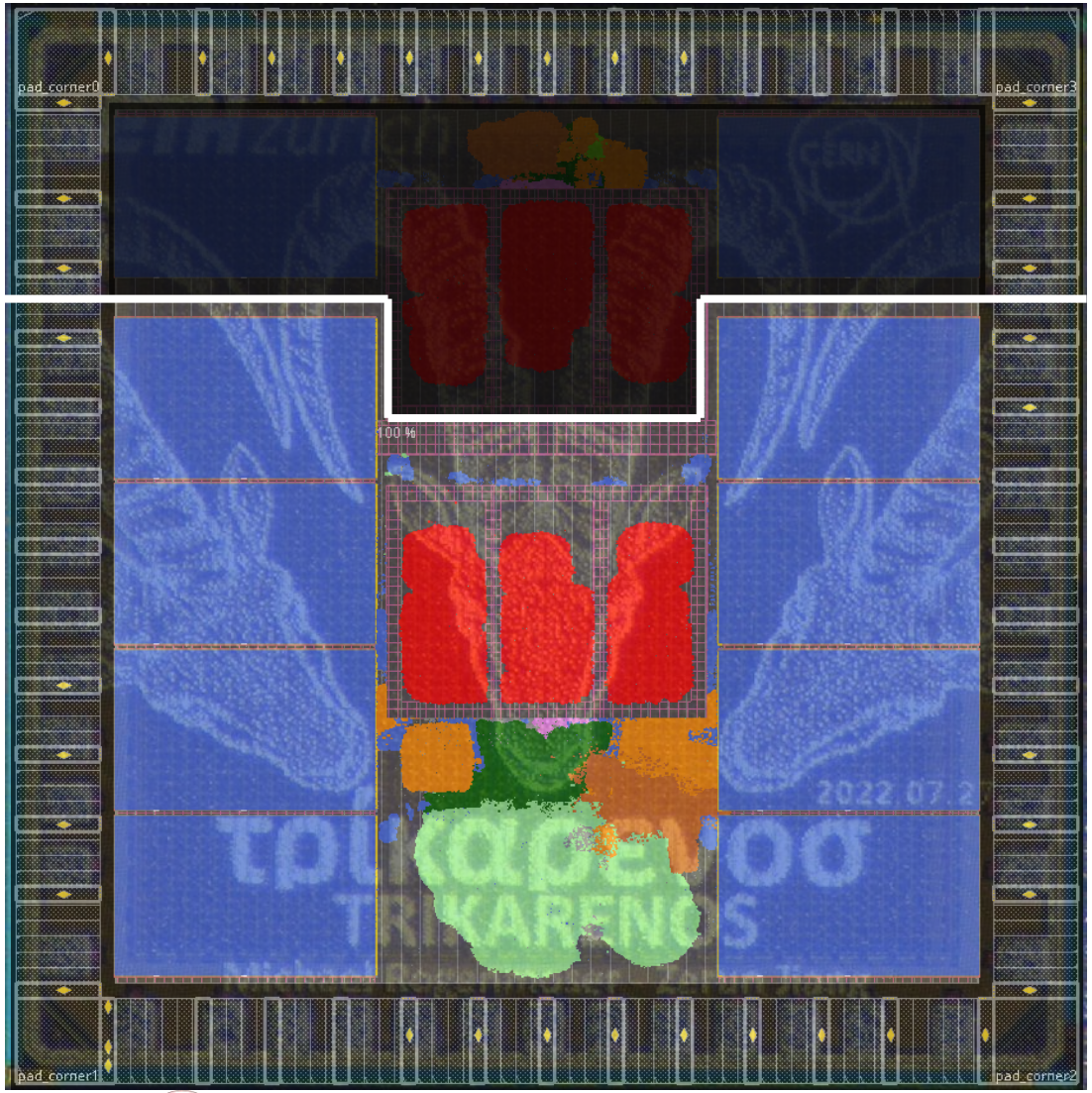
Trikarenos – PULPissimo with Reliability

Reliability



Trikarenos – ASIC implementation in TSMC28

Reliability



- **2 mm² @250 MHz**
- **3 separate Ibex cores**
- **256 KiB Memory in 8 word-interleaved banks**
- **Legend:**
  - Cores
  - HMR Unit
  - Memory (w/ ECC en-/decode)
  - Interconnect
  - Debugger
  - Logging & control registers, ROM, …

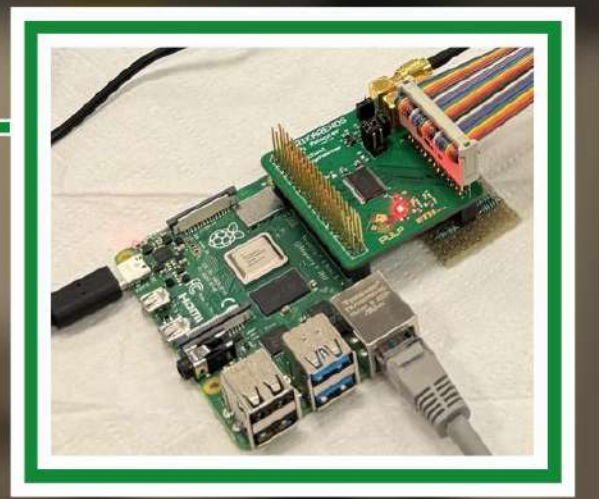
Experiments

Reliability

Trikarenos DUT

Proton Beam
& Collimator

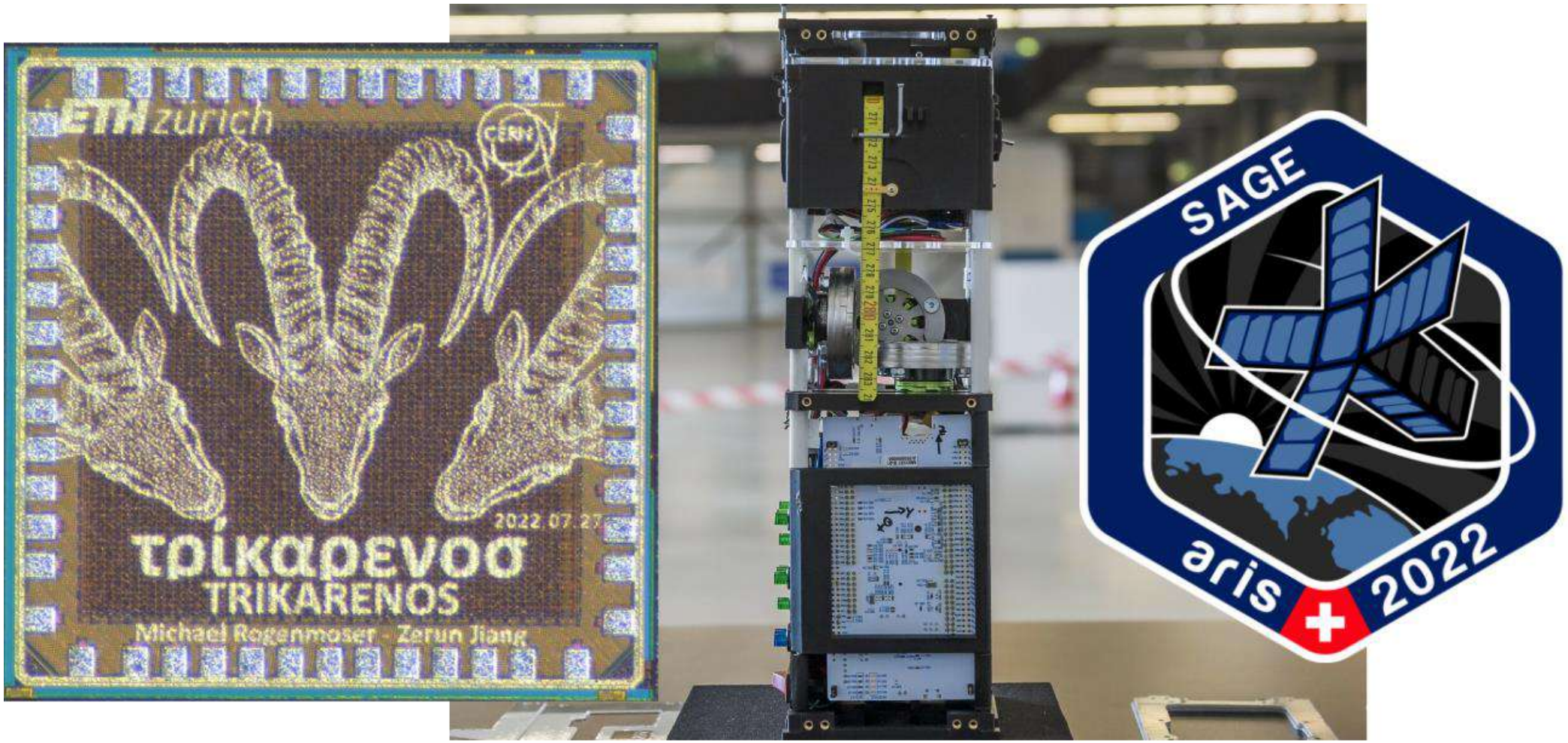
Raspberry Pi
Test Harness



Tested at ChipIR with the help of University of Twente
And hollandPTC with the help of TU Delft

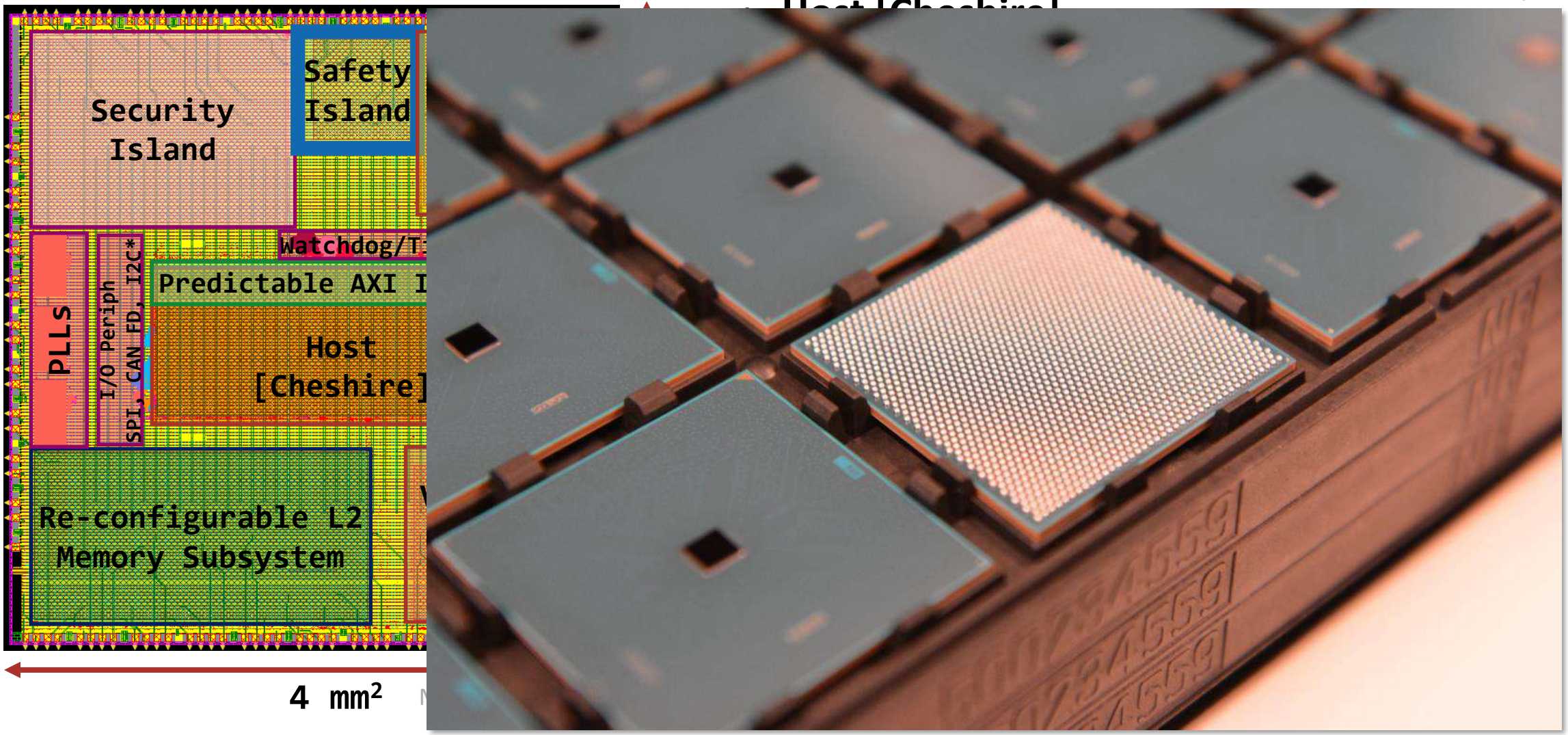
What is next for Trikarenos? Actually go to Space

Reliability



Carfield SoC Flooplan – Arrived this Tuesday

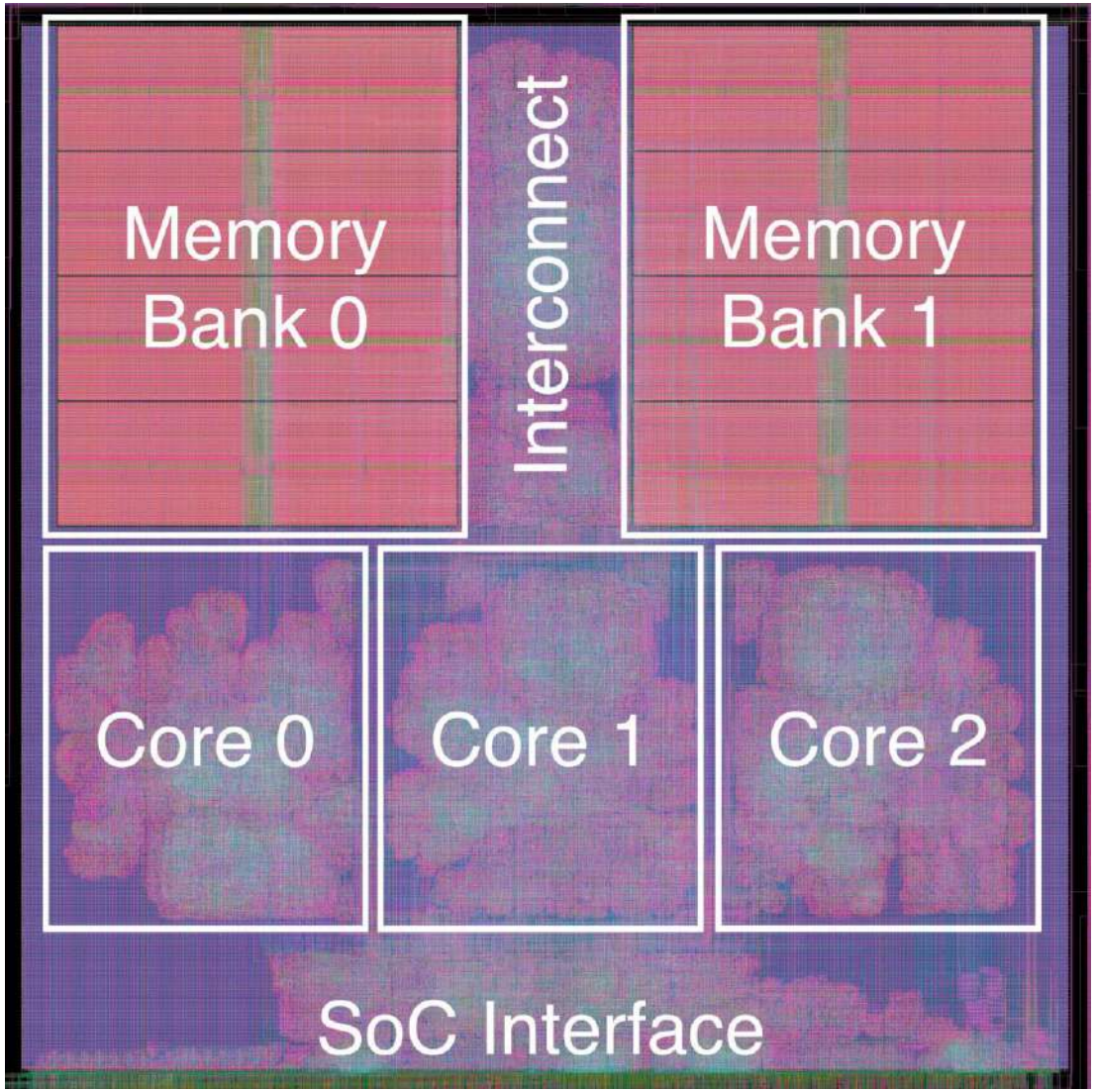
Safety



The safety island in Carfield implemented in Intel16

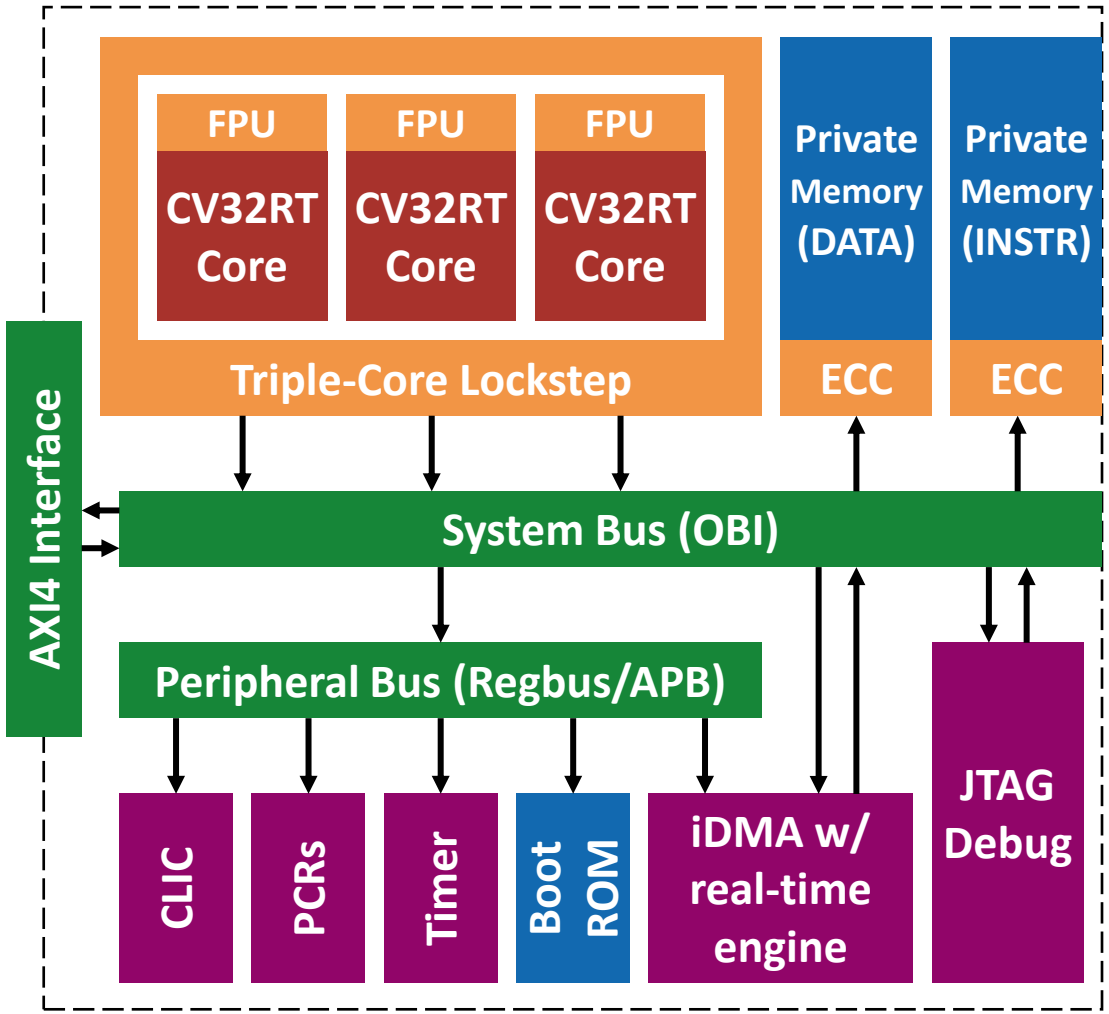
Safety

- **SentryCore Configuration**
  - CV32RT, FPU (32bit), CLIC
  - TCLS, ECC Memory
  - 64 bit AXI interface, no DMA
  - Total 128 KiB ECC-protected Memory
- **Physically separated TCLS Cores**
  - 20 µm margins
  - Avoids multi-bit error from single particle
- **Implementation Results**
  - Clock Frequency: ***500 MHz***
  - Area: ***0.42 mm²***
  - Power (preliminary): ***50-70 mW***



Pre-demonstrator to show what can be achieved

Safety



- **Independent DMA unit**
  - Transfer of data in and out of the mega-IP
- **Triple-Core Lockstep**
  - Majority Voters on all outputs
- **ECC-protected Memory**
  - 39-32 Hsiao Code for single error correction, double error detection
- **CV32RT [Balas et. al., 2023]**
  - CV32E40P-based real-time core
- **Core-Local Interrupt Controller (CLIC)**
- **fastIRQ extension**

How open source helped us in our projects

- **Start from a working system, no need to reinvent everything**
  - Silicon proven SoC templates available
- **Easy to extend with accelerators and custom blocks/memories**
  - Platforms designed for heterogeneous acceleration (Occamy, Mempool)
- **Not limited by previous design choices**
  - Everything can be adapted and changed
- **Full source code allows you to observe/record everything**
  - Not limited to available performance counters/timers, build add/your own
- **Possible to exploit the results commercially**

Most projects we have involve security, safety, reliability

- **There is much more to be done**
  - We have made a decent start
  - Different SoC templates available to implement a variety of solutions
  - And we have some great ideas
- **We need partners to reach our goals**
  - We are not experts in security, safety, reliability
  - We need partners to co-operate and learn from
- **Open source hardware helps reach these goals**
  - Easier to collaborate, less roadblocks
  - Collaborations do no block commercial exploitation (permissive licensing)

The future is bright