ETH zürich

New Projects at ETH Systems
(and why you should come do a Ph.D.)

Systems@ETH zürich

The Systems Group at ETH

About ETH

- Swiss Federal Polytechnic
- Founded 1854 (with EPFL)
- In central Zürich
- CHF1.8B/year
- Famous Alumni:
  - Albert Einstein
  - Wolfgang Pauli
  - Gernot Heiser ;)

Industrial Collaboration and Support

# A Selection of New/Ongoing Projects

- Barrelfish
- Runtime Verification
- Formal Hardware Modelling
- Strymon
- Enzian

The State of the Fish

- 7 architectures: OMAP44xx, ARMv7/GEM5, X-Gene 1, ARMv8/GEM5, Xeon Phi, x86-64, x86-32
- 42 applications + 51 test apps
- 9 languages
- 32 committers
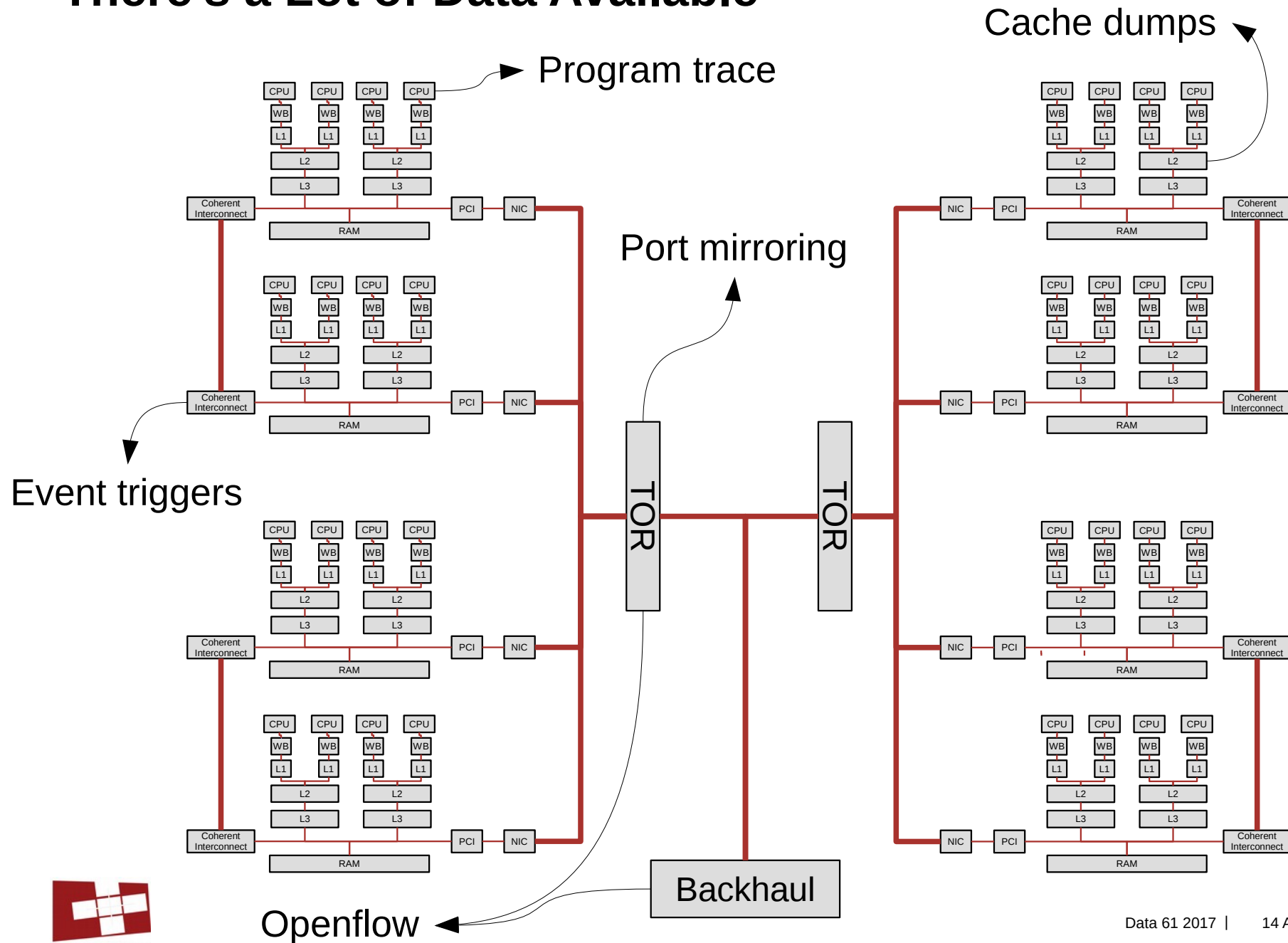- 9 years old
- > 1.1M lines of code

We're Building a *Large Program Collider*

tlbi alle3; dmb

*x = *y++; malloc();

0100111101001101
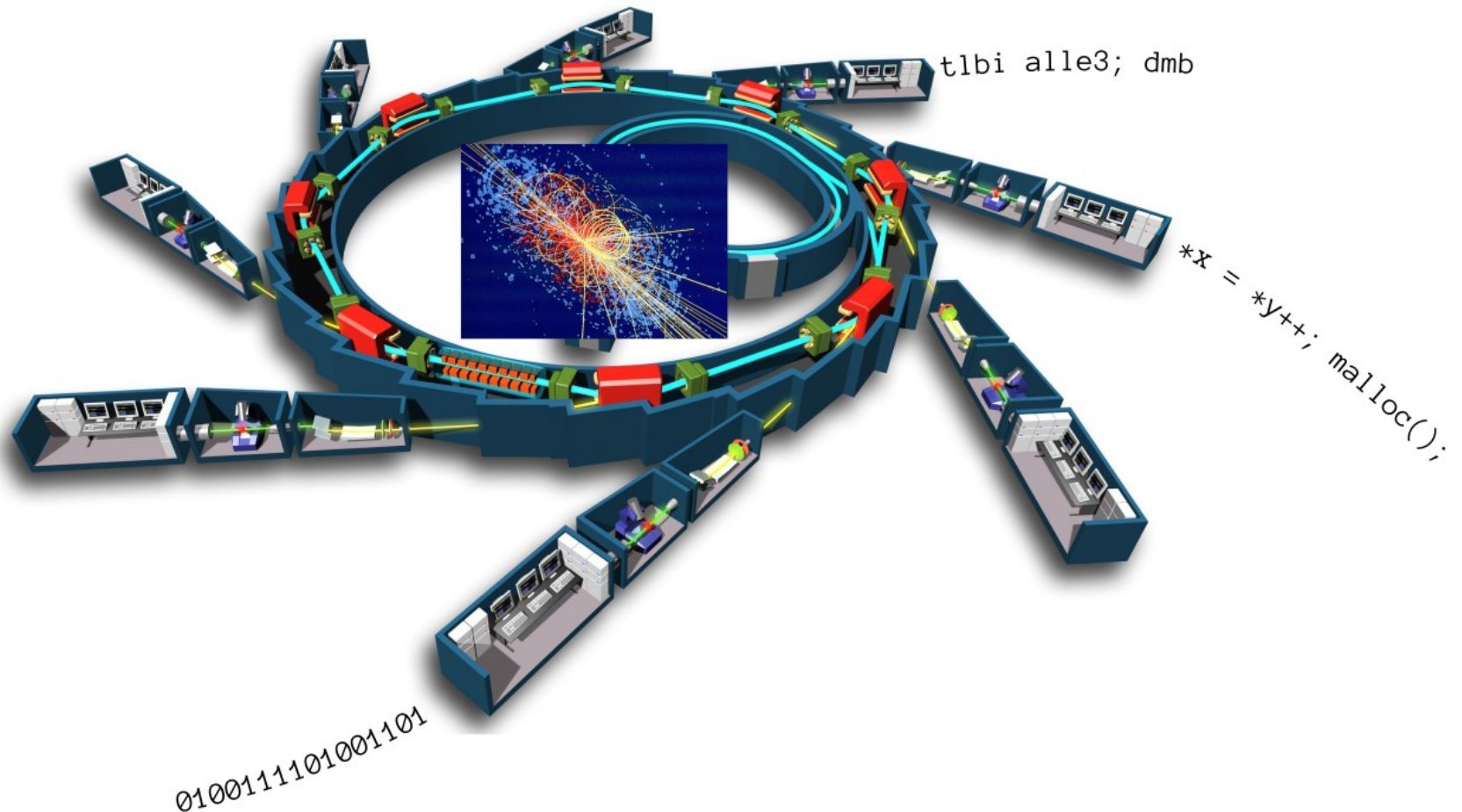
Images: CERN; Chaix & Morel et associés

Collide *instructions* at 0.99*c*, and observe the decay products.

There's a Lot of Data Available

Program trace

Cache dumps
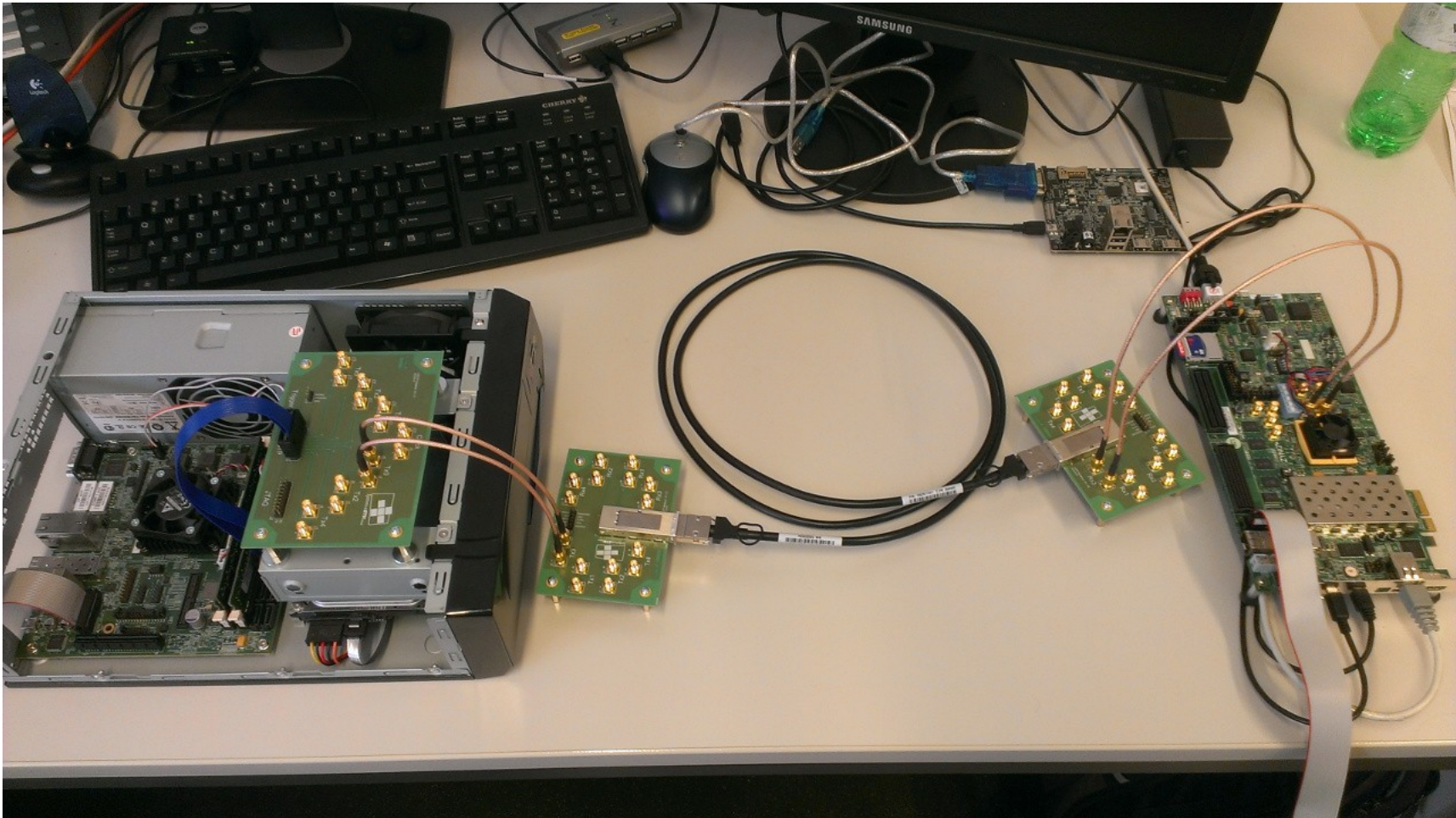
Port mirroring

Event triggers

TOR

TOR

Backhaul

Openflow

ARM High-Speed Serial Trace Port

Image: Teledyne Lecroy

- Streams from the *Embedded Trace Macrocell*.
- Cycle-accurate control flow + events @ 6GiB/s+
- Compatible with FPGA PHYs.
- Well-documented protocol.
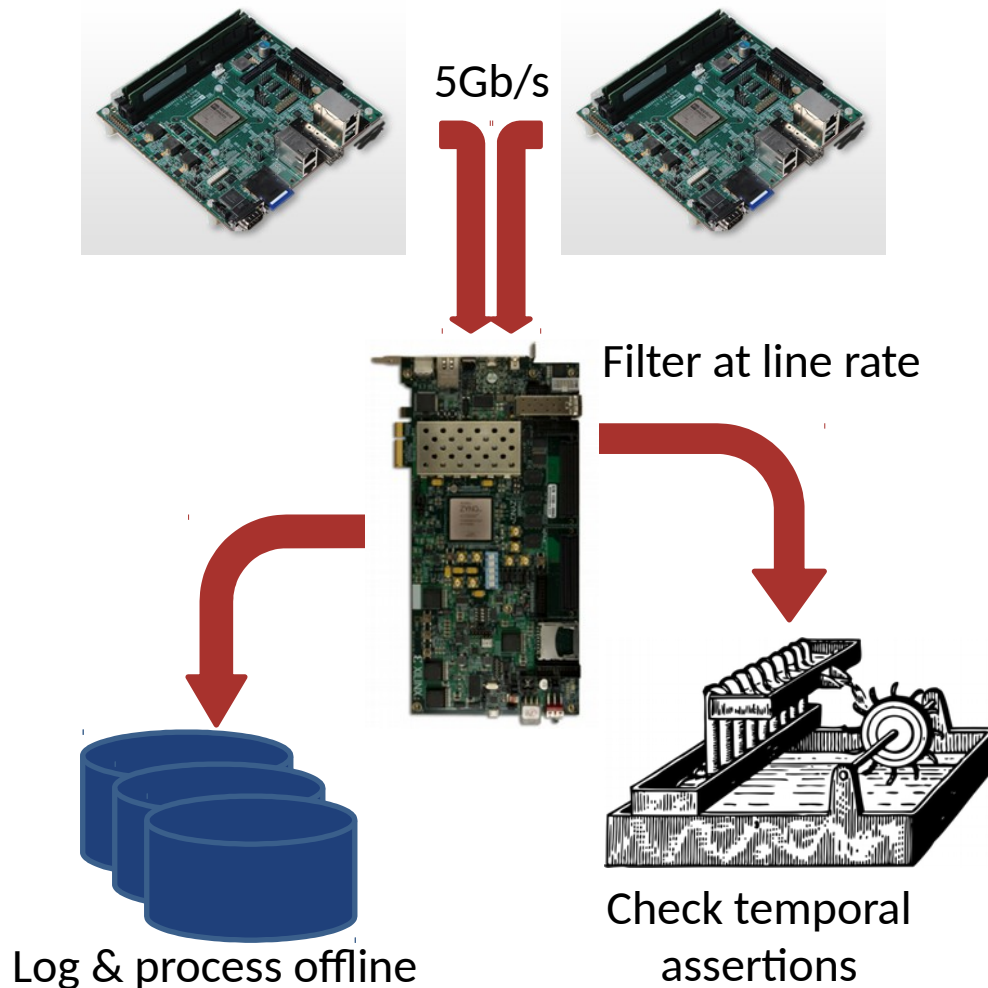 - Aurora 8/10
- Available on ARMv8

HSSTP Testbench

Systems@ETH Zürich
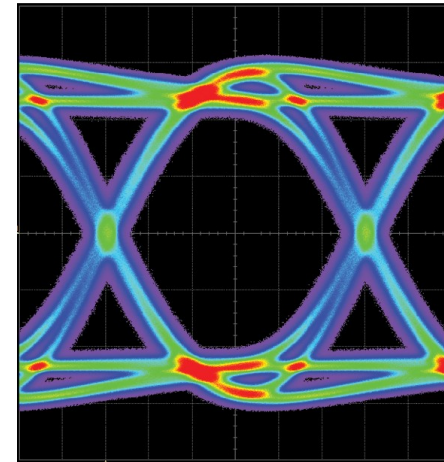
Hardware Tracing for Correctness

5Gb/s

Filter at line rate

Log & process offline
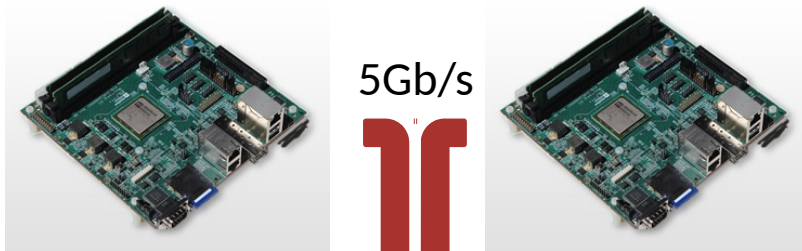
Check temporal assertions

Are HW operations right?

$$\exists va.\, va \to pa$$
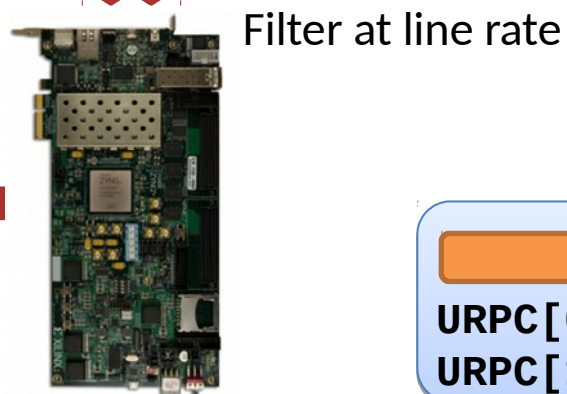
```
unmap(pa);
cleanDCache();
flushTLB();
```

$$\nexists va.\, va \to pa$$

- Real time pipeline trace on ARM.
- Can halt and inspect caches.
- HW has "errata" (bugs).
- Check that it actually works!
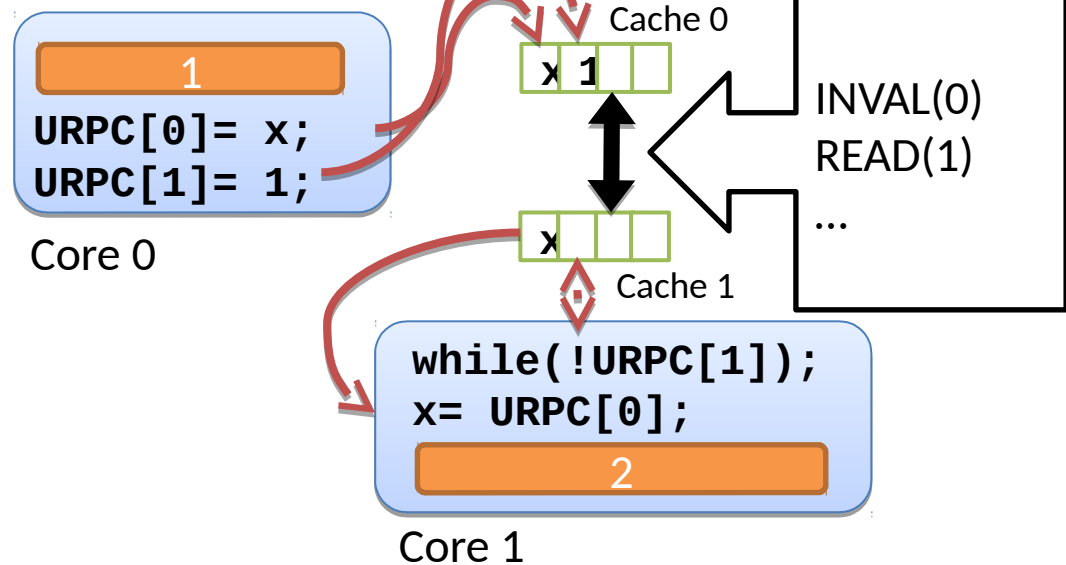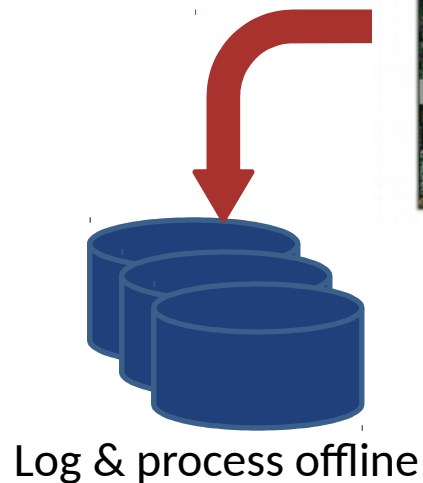- Catch transient and race bugs.

Systems@ETH zürich

Hardware Tracing for Performance

5Gb/s

Filter at line rate

Log & process offline

- Should see N coherency messages.
- Do we?
  - The HW knows!

Is URPC optimal?

Cache 0

x 1

```
1
URPC[0]= x;
URPC[1]= 1;
```

Core 0

INVAL(0)
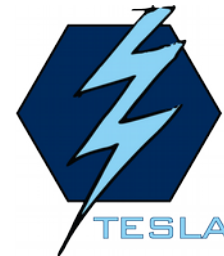READ(1)
...

x

Cache 1

```
while(!URPC[1]);
x= URPC[0];
2
```

Core 1

Properties to Check: Security

- Runtime verification is an established field.

- Lots of existing work to build on.

- What properties could we check efficiently?

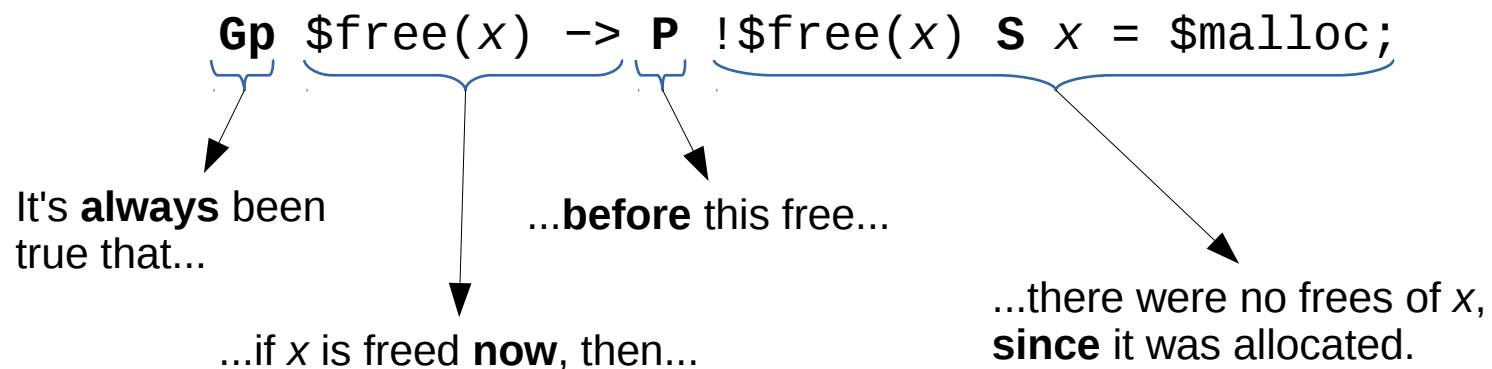- How could we map them to the filtering pipeline?


```
/* A very simple TESLA assertion. */
TESLA_WITHIN(example_syscall,
    previously(security_check(ANY(ptr),
            o, op) == 0));
```

http://www.cl.cam.ac.uk/research/security/ctsrd/tesla/

Properties to Check: Memory Management

```
void *a = malloc();
...
{a is still allocated}
free(a);
```

- Could we check this?

Gp $\$free(x) \rightarrow$ **P** $!\$free(x)$ **S** $x = \$malloc;$

It's **always** been true that...

...**before** this free...

...if x is freed **now**, then...

...there were no frees of x, **since** it was allocated.
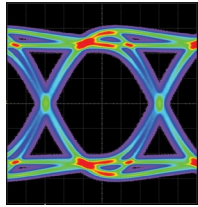
Checking LTL with Automata

This is a well-studied problem, and standard algorithms exist:

Gp $free(x) \rightarrow$ **P** !$free(x) **S** x = $malloc;
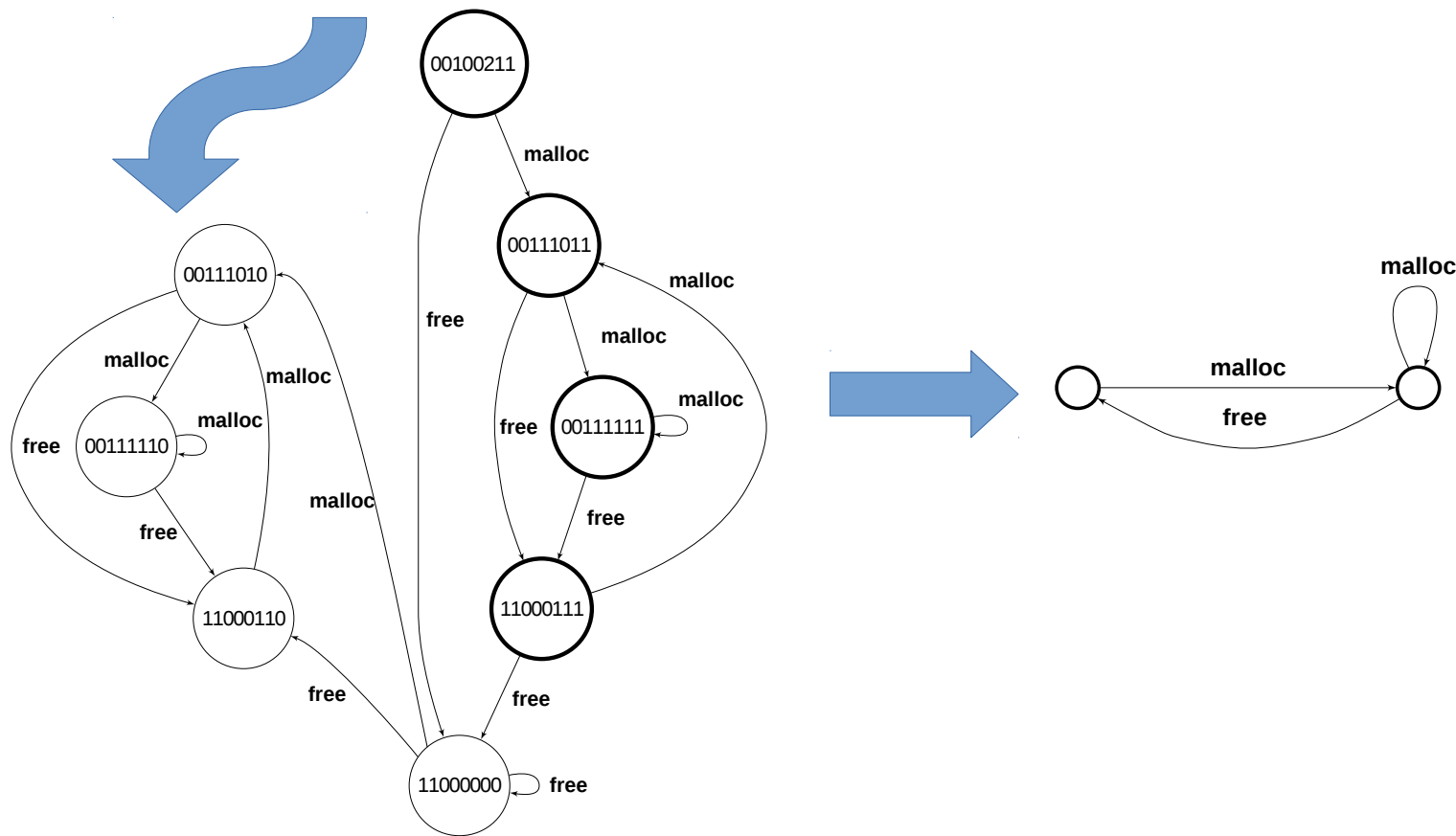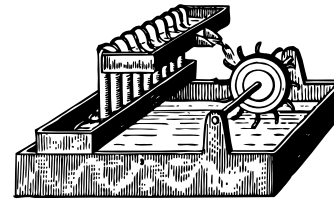
A Streaming Verification Engine
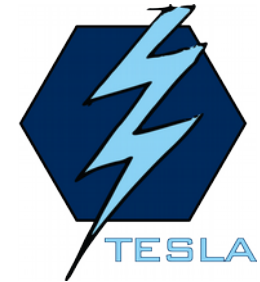
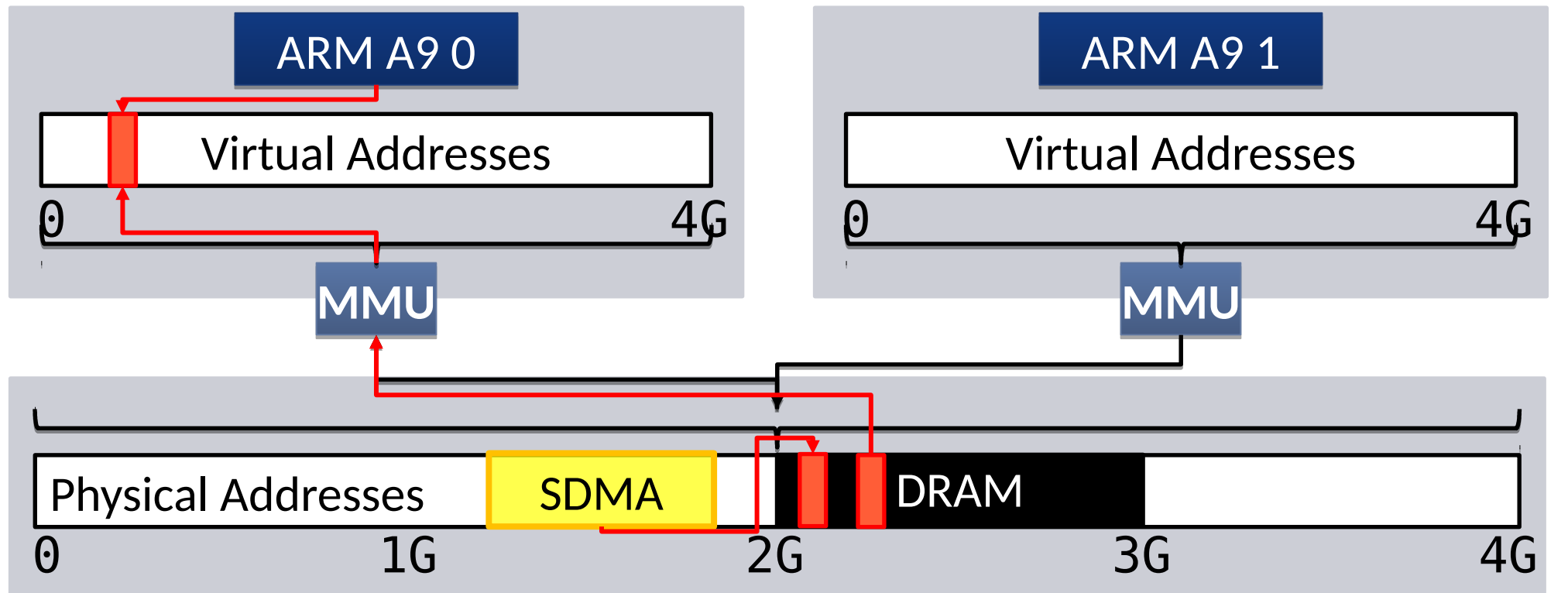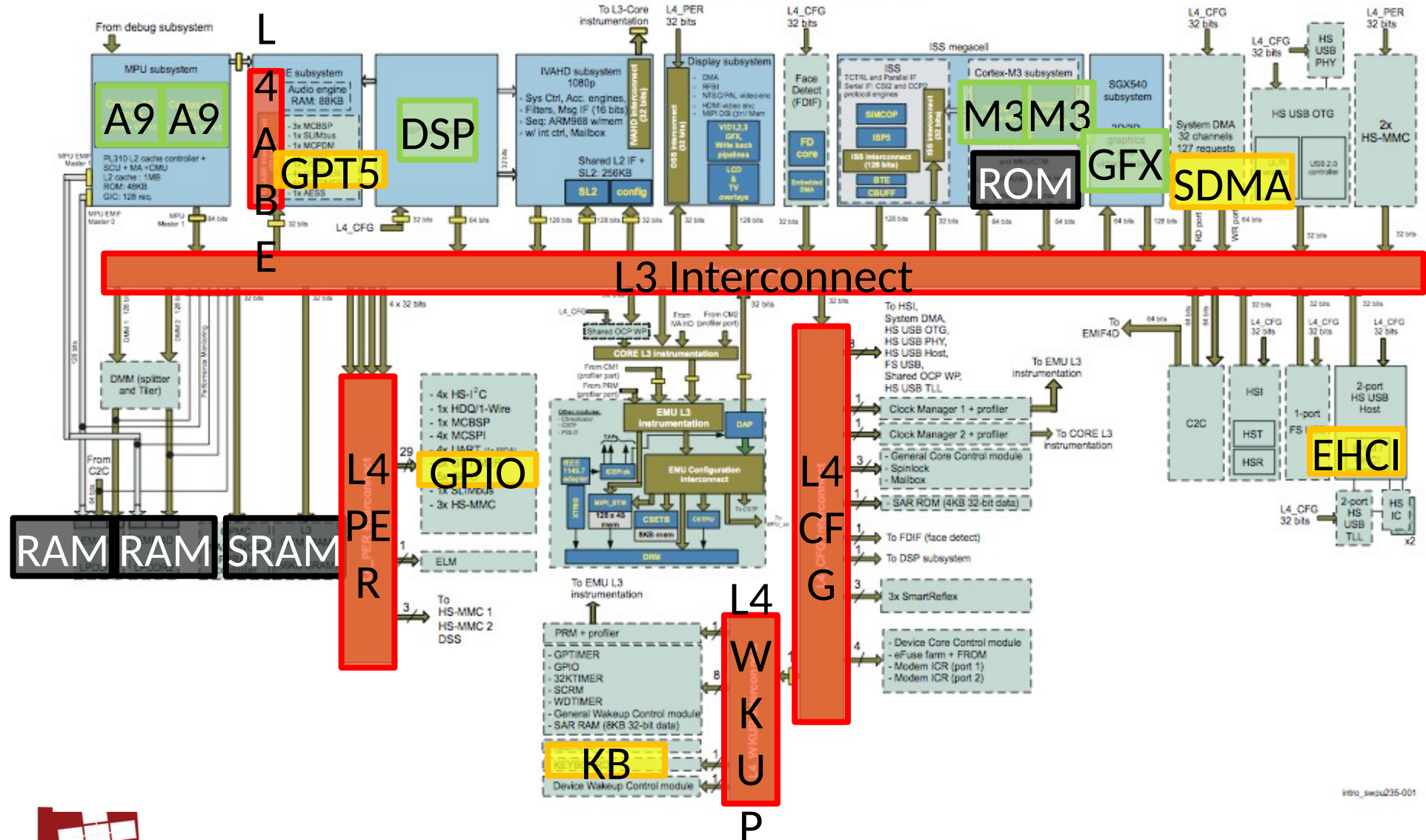| Sources	Capture	Processing	Properties
HSSTP	ETM Sequencer	Dataflow Engine	TESLA
Packet Capture	FPGA Capture	FPGA Offload	`malloc()` pairing
			Coherence correctness

Constraints →

← Requirements

Formal Hardware Modelling

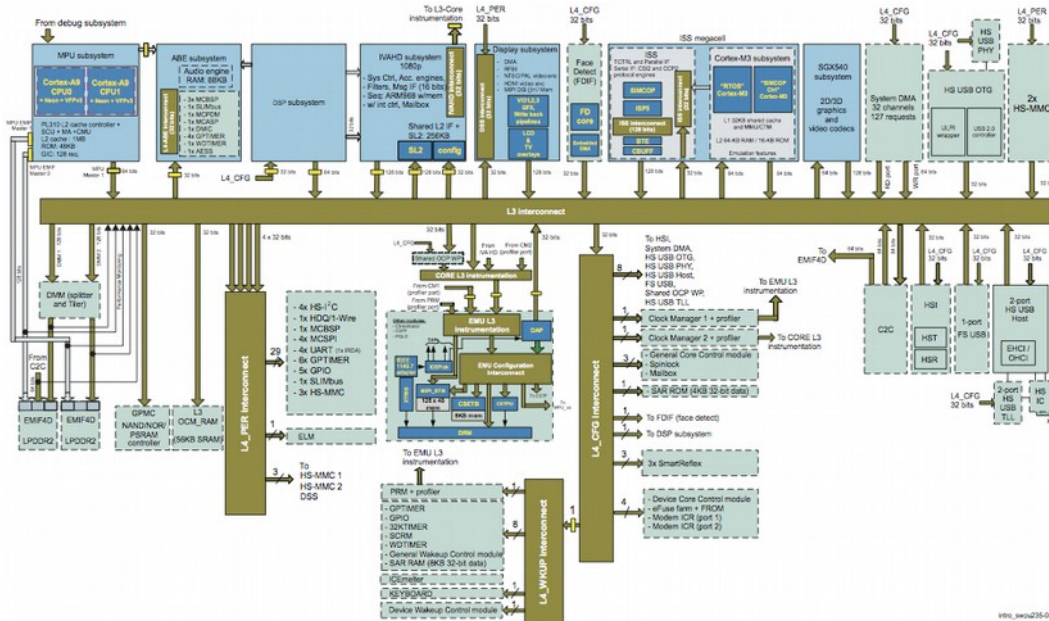Ti OMAP 4460 SoC

How the Computer *Actually* Looks

Your mobile phone... 5-10 years ago!

Decoding Nets

Node

| | map | | accept | |

Node

| | | accept | |

VA9:0
```
4001000
4001fff
```

PA9:0
```
8000000
bffffff
```

L3
```
8000000
bffffff
```

```
0000000
4000000
```
DRAM
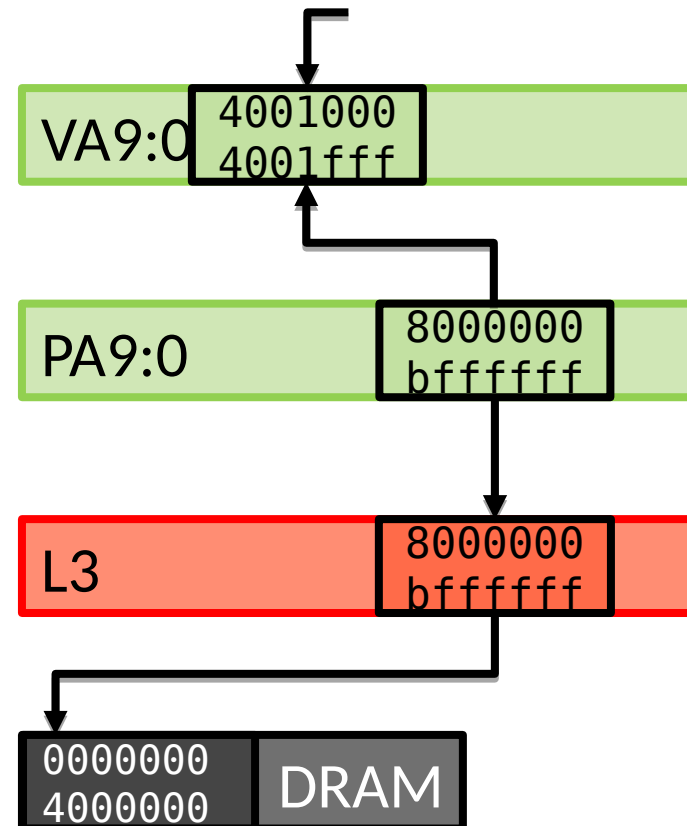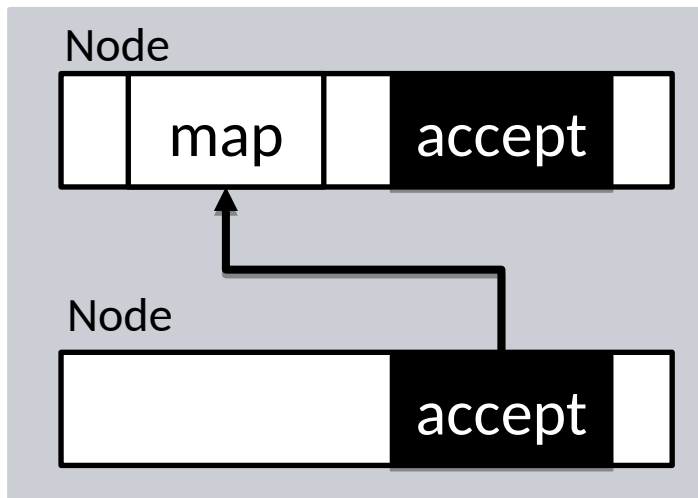
The OMAP4460 Decoding Net

$V_{A9:0}$ **is map** $[20000_3/12$ **to** $P_{A9:0}$ **at** $80000_3]$ $V_{A9:1}$ **is map** $[20000_3/12$ **to** $P_{A9:1}$ **at** $80000_3]$

$P_{A9:0}, P_{A9:1}$ **are map** $[40138_3/12$ **to** GPT **at** $0]$ **over** $L3$ V_{DSP} **is over** P_{DSP}

P_{DSP} **is map** $[1d3e_3/12$ **to** GPT **at** $0]$ **over** $L3$ $L2_{M3}$ **is map** $[0_{30}$ **to** $L3$ **at** $80000_3]$

V_{M3}, V_{M3} **are over** $L1_{M3}$ $L1_{M3}$ **is map** $[0_{28}$ **to** $MIF]$

RAM_{M3} **is accept** $[55020_3/16]$ $L4$ **is map** $[49038_3/12$ **to** GPT **at** $0]$

ROM_{M3} **is accept** $[55000_3/14]$ GPT **is accept** $[0/12]$

MIF **is map** $[0 - 5\texttt{fffffff}$ **to** $L2_{M3}, 55000_3/14$ **to** $RAM_{M3}, 55020_3/16$ **to** $ROM_{M3}]$

$L3$ **is map** $[49000_3/24$ **to** $L4$ **at** $40100_3, 55000_3/12$ **to** $MIF]$ **accept** $[80000_3/30]$

Using the model

- ## Static Configuration:
  - We can now generate the kernel page tables directly from the formal spec.
- ## Dynamic Discovery and Reconfiguration:
  - The SKB can be populated at runtime – extend the model as hardware is discovered.
- ## Scheduling:
  - We collaborate with the DB research group on operator scheduling – this work needs the model data.

Strymon: Online Datacenter Analytics and Management

Inputs	Functions	Outputs
Network topology	Cross-layer state overview	Look-up queries
Performance traces	Fault diagnosis and prevention	Complex analytics
Configuration data	Automatic management and configuration	What-if scenarios
Monitoring data	Resource optimization Capacity planning Invariant verification	Policy enforcement
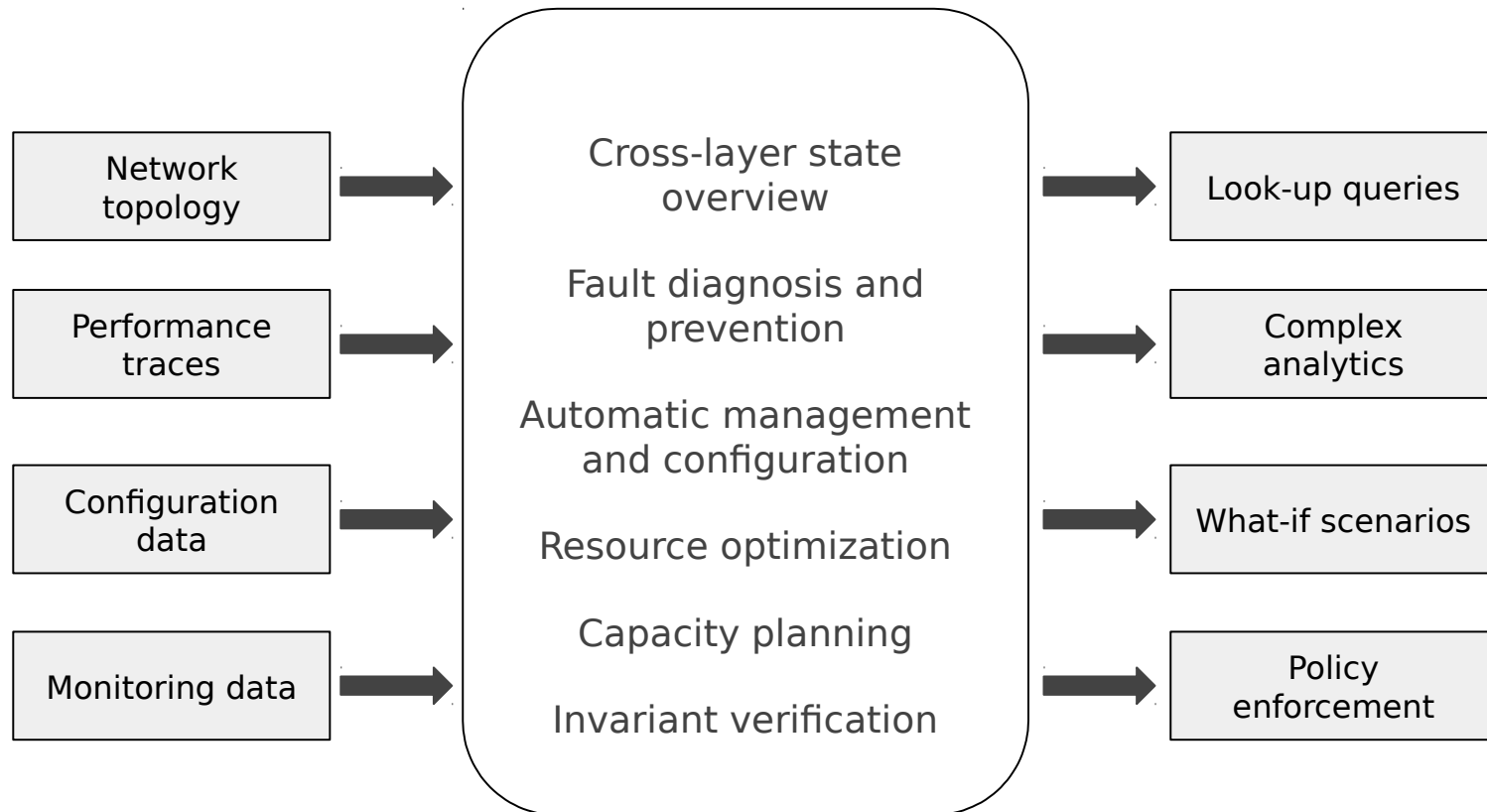
Support: aMaDEUS   FNS|SNF FONDS NATIONAL SUISSE SCHWEIZERISCHER NATIONALFONDS FONDO NAZIONALE SVIZZERO SWISS NATIONAL SCIENCE FOUNDATION   Google

More info: http://strymon.systems.ethz.ch

Data Management

Distributed Stream Processing

Graph Algorithms

Real-time sessonization

Incremental network routing

Online critical path analysis

Provenance and interactive explanations

Streaming fault-tolerance and data durability
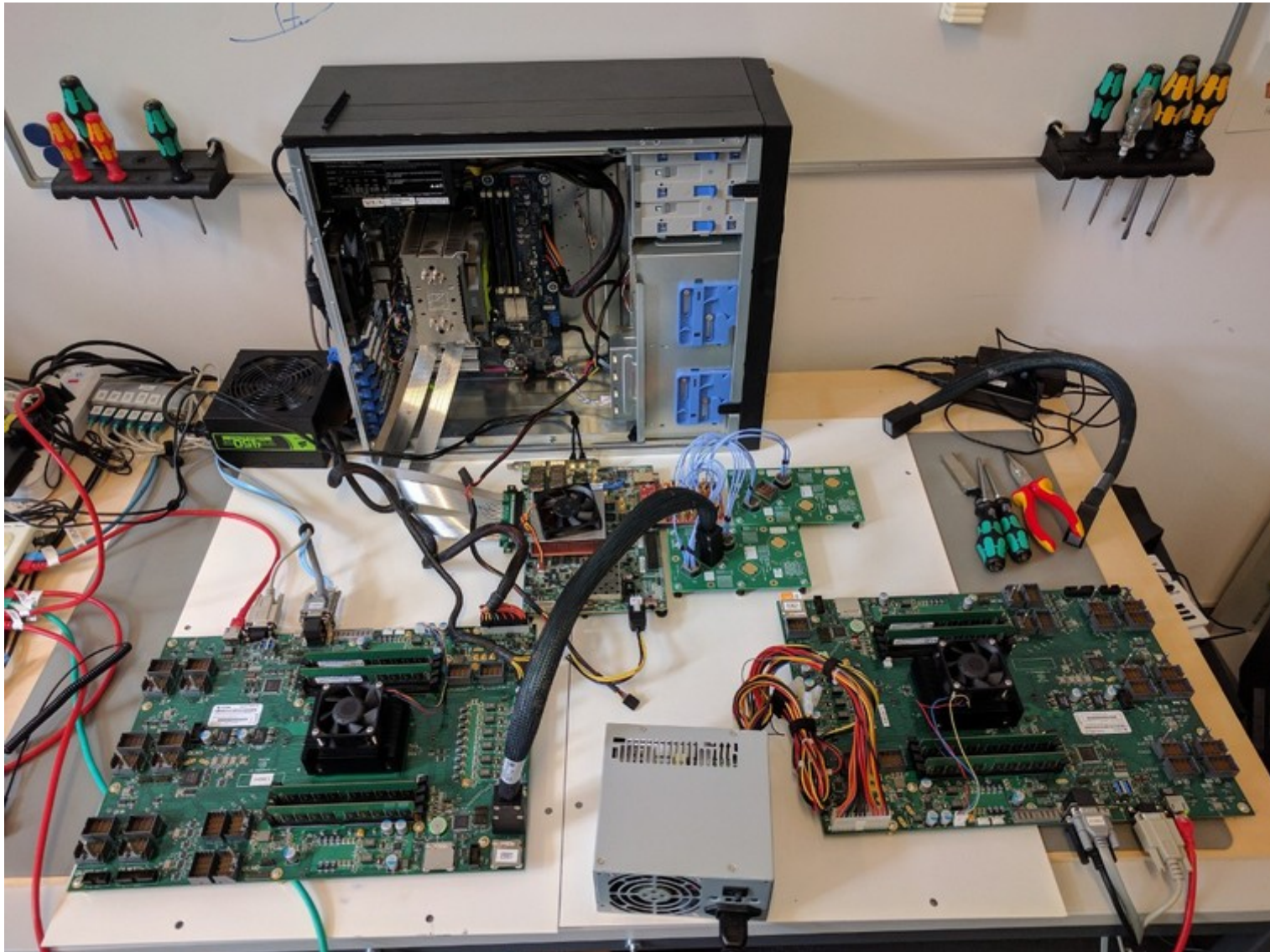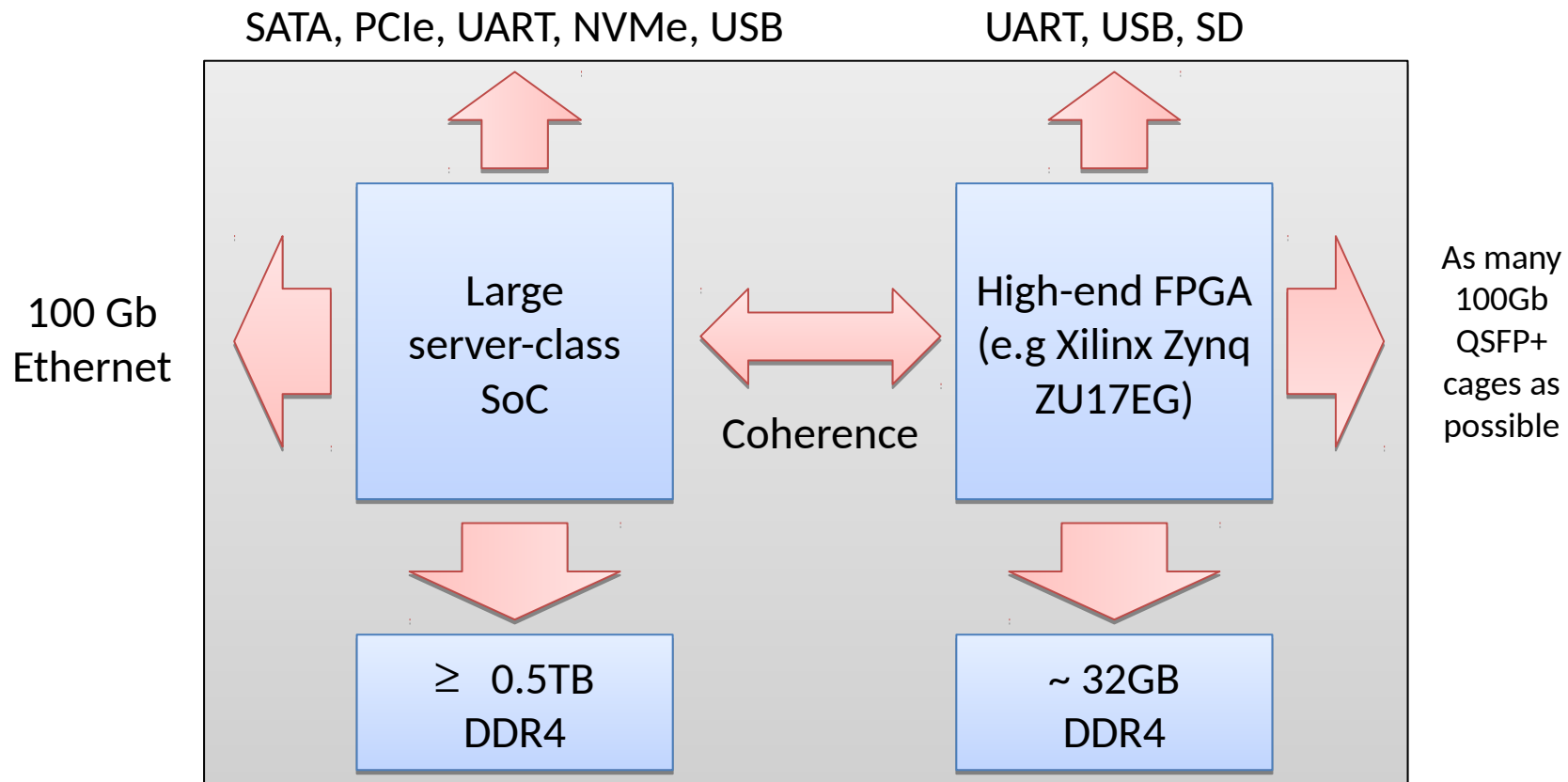
Automatic re-scaling of distributed dataflows

TensorFlow

Apache Flink

Spark

Contact: Vasia Kalavri (kalavriv@inf.ethz.ch)

Enzian: Building Understandable Hardware

Enzian: Sketch

Living and Working in Zürich

Living and Working in Zürich