TOGETHER FOR RISC-V TECHNOLOGY
& APPLICATIONS

RISC-V Summit Europe 2024

TRISTAN: Free Access Training on EDA tooling for RISC-V

09:00-11:30, Friday, 28.06.2024, Munich

TRISTAN has received funding from Chips Joint Undertaking (CHIPS-JU) under grant agreement nr. 101095947.

CHIPS JU receives support from the European Union's Horizon Europe's research and innovation programme and Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey

Virtual platforms and virtual prototypes for RISCV

Rocco Jonack, MINRES
Eyck Jentzsch, MINRES
Alexander-Eyck Jentzsch, MINRES

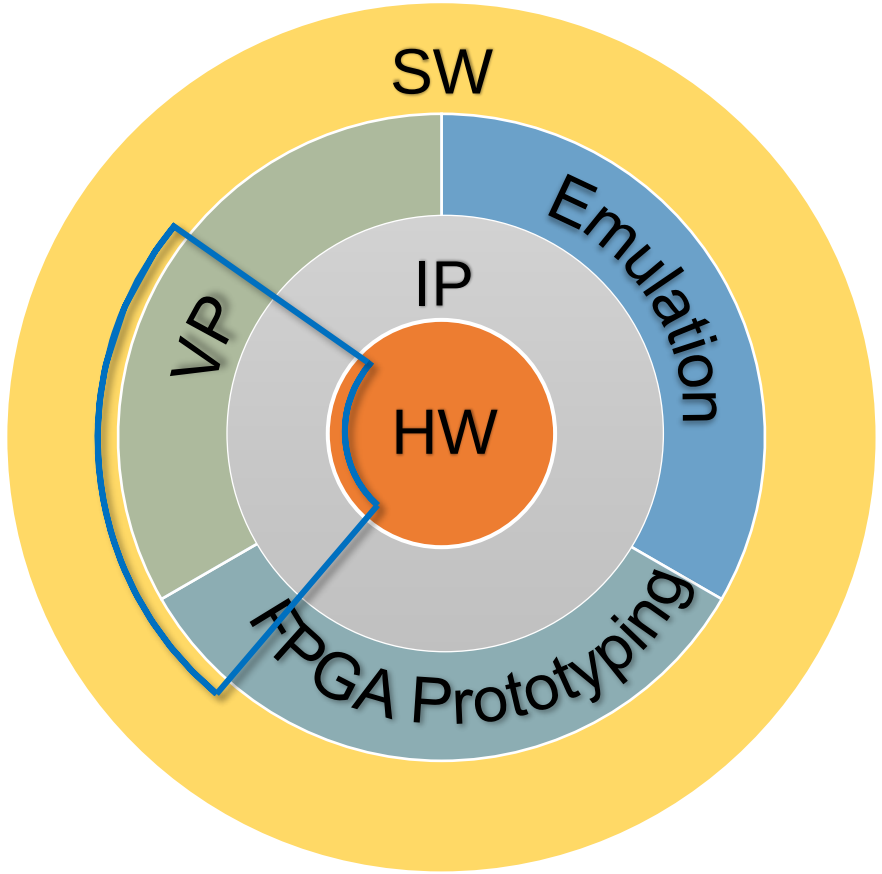
The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

MINRES introduction

- MINRES is an enabling company, dedicated to providing the expertise and solutions required for improving development productivity:
  - Methodology services + productivity IP
  - Design consulting services & value-added application engineering
  - ISO26262 compliant RISC-V RTL IP
  - Cloud based hybrid simulation solution (RTL/VP)
- MINRES Technologies GmbH is a privately-held, remote-first startup based in Germany

Trustworthy Software-Driven Hardware for the Intelligent Edge

- VP development is early in development cycle
  - IP reuse and 3rd party integration
  - Using productivity libraries
  - Tools can help, but plan carefully
  - SW development
  - SW drive requirements for SoC
- Early VP models build foundation for detailed models
  - Architecture analysis
  - Test preparation



VP environments and requirements

- Open-source environments
  - Qemu, GEM5, Renode
- Commercial tools
  - Virtualizer, Simics
- Customer environments
  - Larger customers
- Mixed environments
  - Hybrid solutions
  - IP specific solutions
  - Specialty solutions

- SW debug interface
  - Port to connect debugger
  - Memory viewing/loading
  - Interrupt monitoring
- Execution speed
  - Depends on use case
- Extensibility of prototypes
  - Which peripherals are modeled
- Model accuracy
  - No timing guarantees, but functional correctness

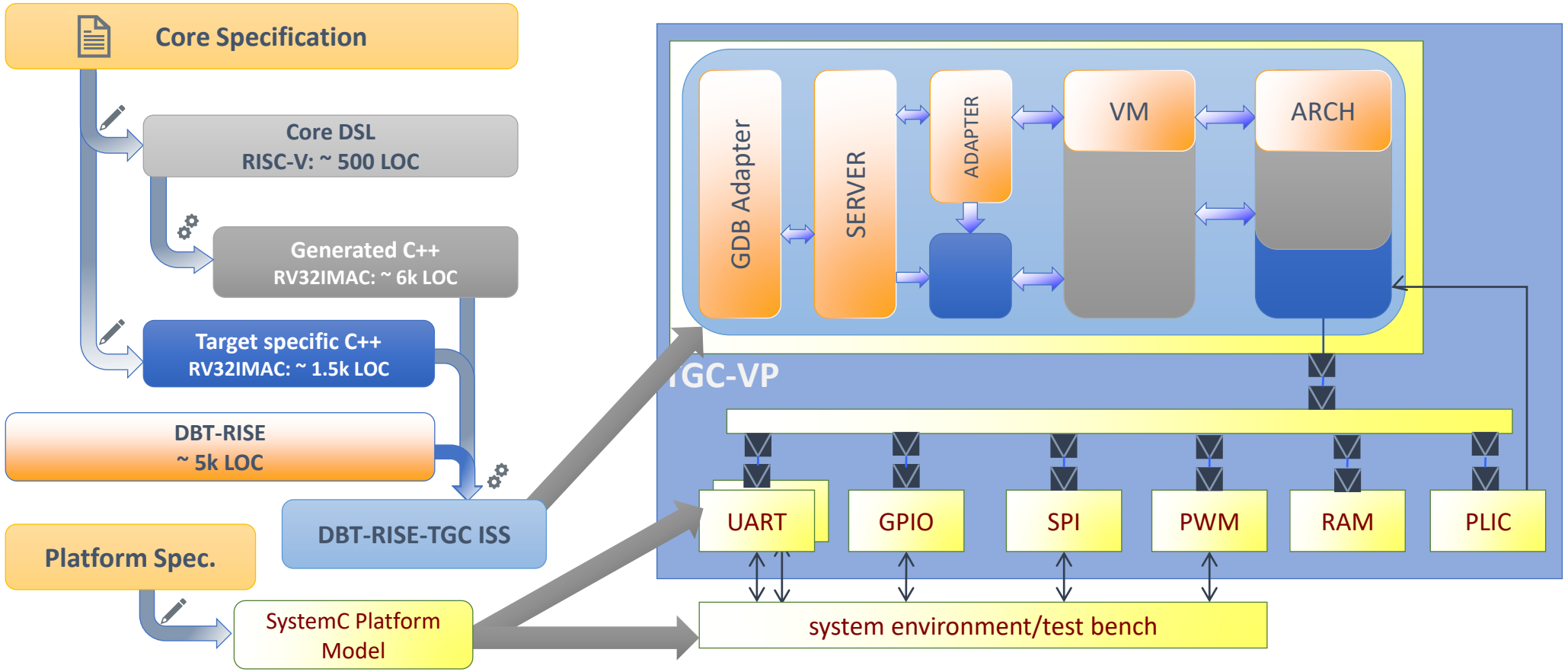
Architectural modeling

- Open-source environments
  - GEM5
- Commercial tools
  - PlatformArchitect
- Customer environments
  - Larger customers
- Mixed environments
  - Hybrid solutions
  - IP specific solutions
  - Specialty solutions

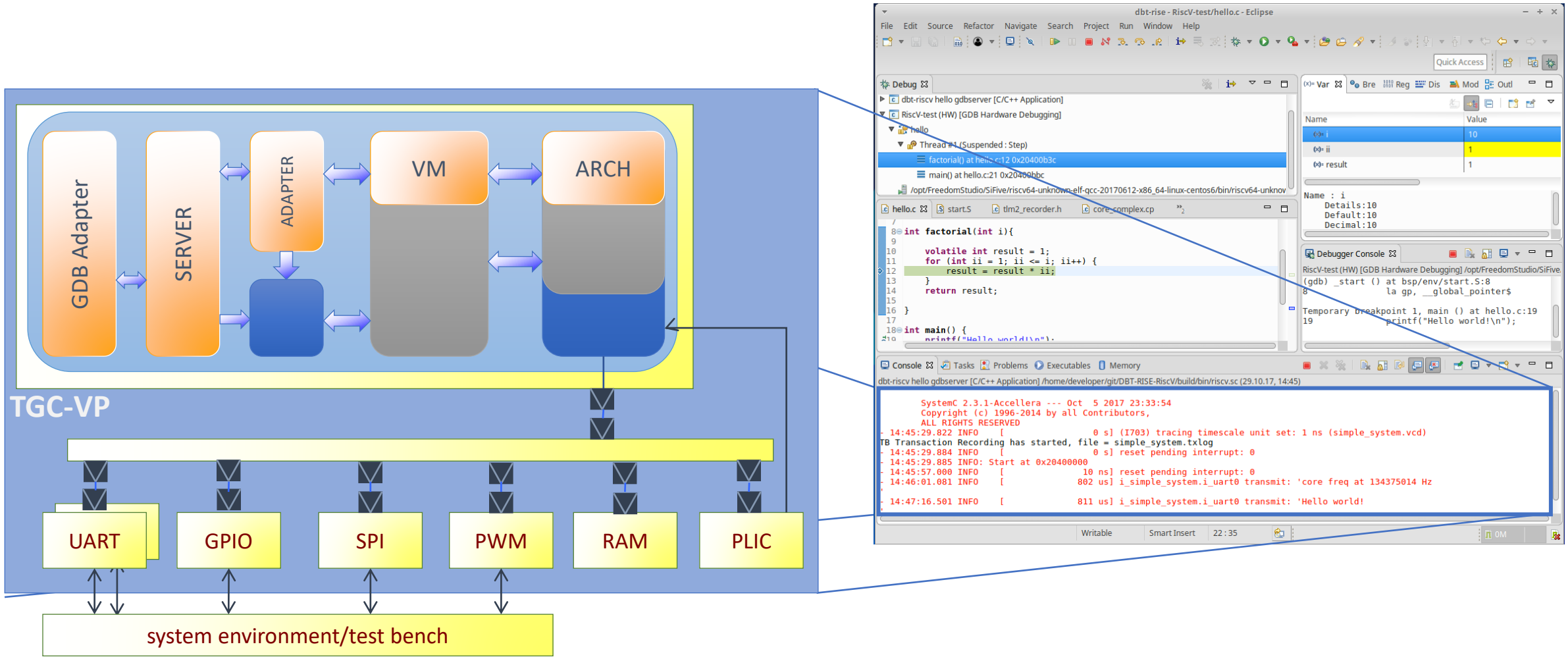
- Cycle accuracy within specified range
- Execution speed
- Extensibility and configurability of platform
- Reporting of results

DBT-RISE based platform

Open-Source Infrastructure for Dynamic Binary Translation (jit compiling) for ISS Generation
Different backends can be used

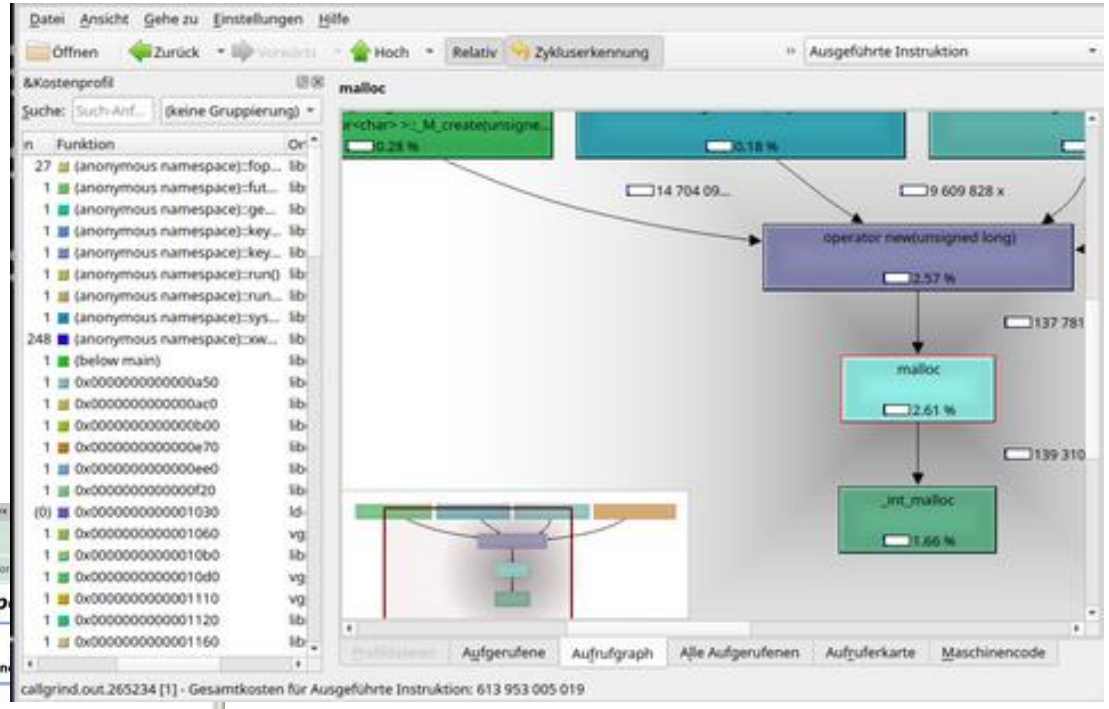


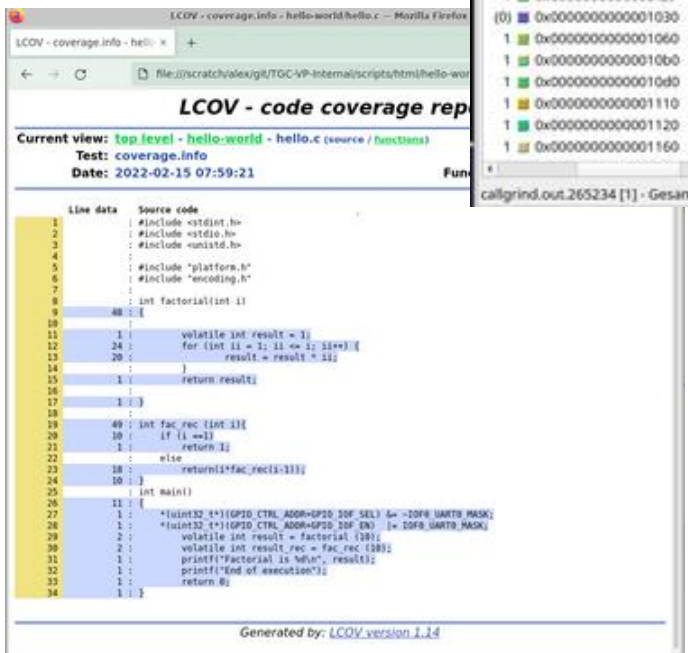
DBT-RISE RISC-V VP



DBT-RISE - Plugins

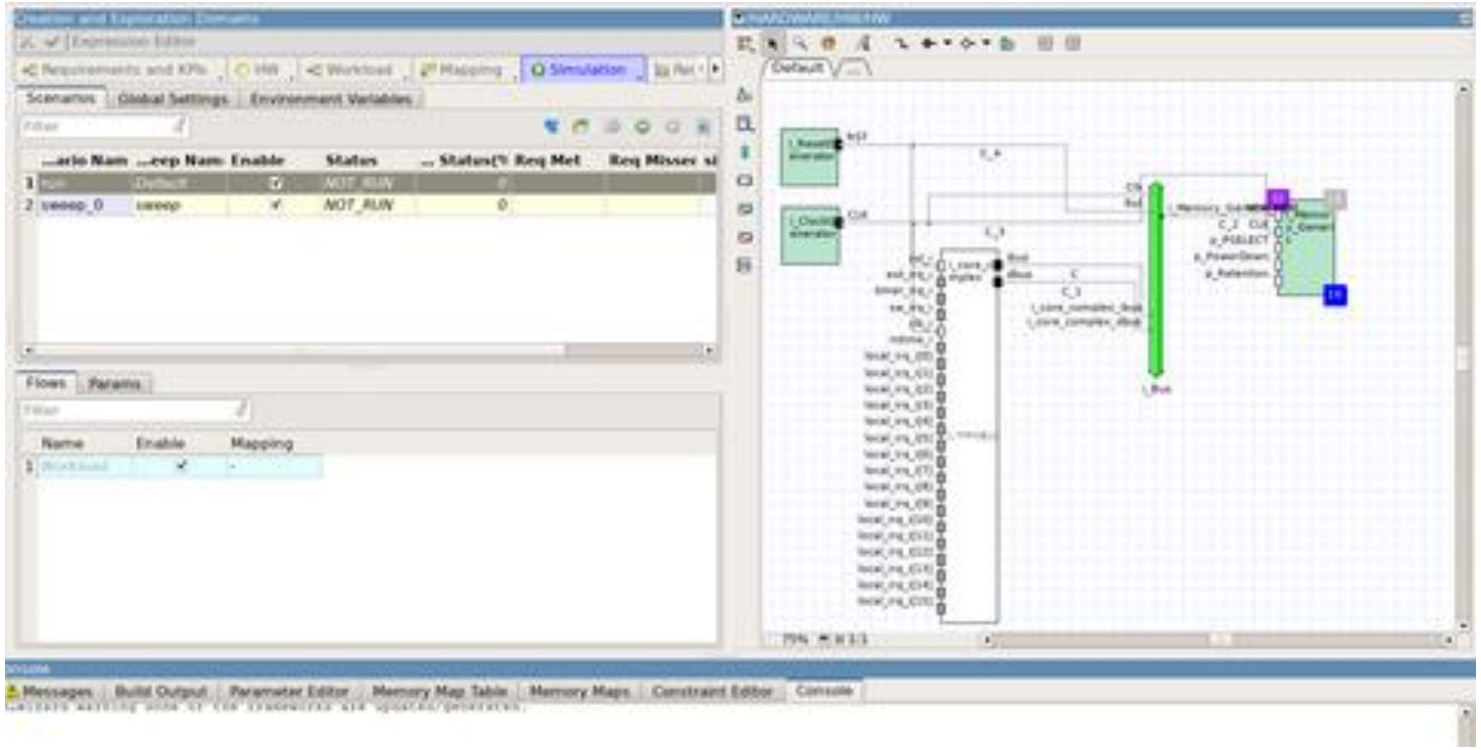
- Existing Plugin infrastructure
- For example:
  - Cycle Annotation
  - Instruction Tracing
    - Coverage Visualization with e.g. lcov
    - Profiling with callgrind
  - Register Dumping





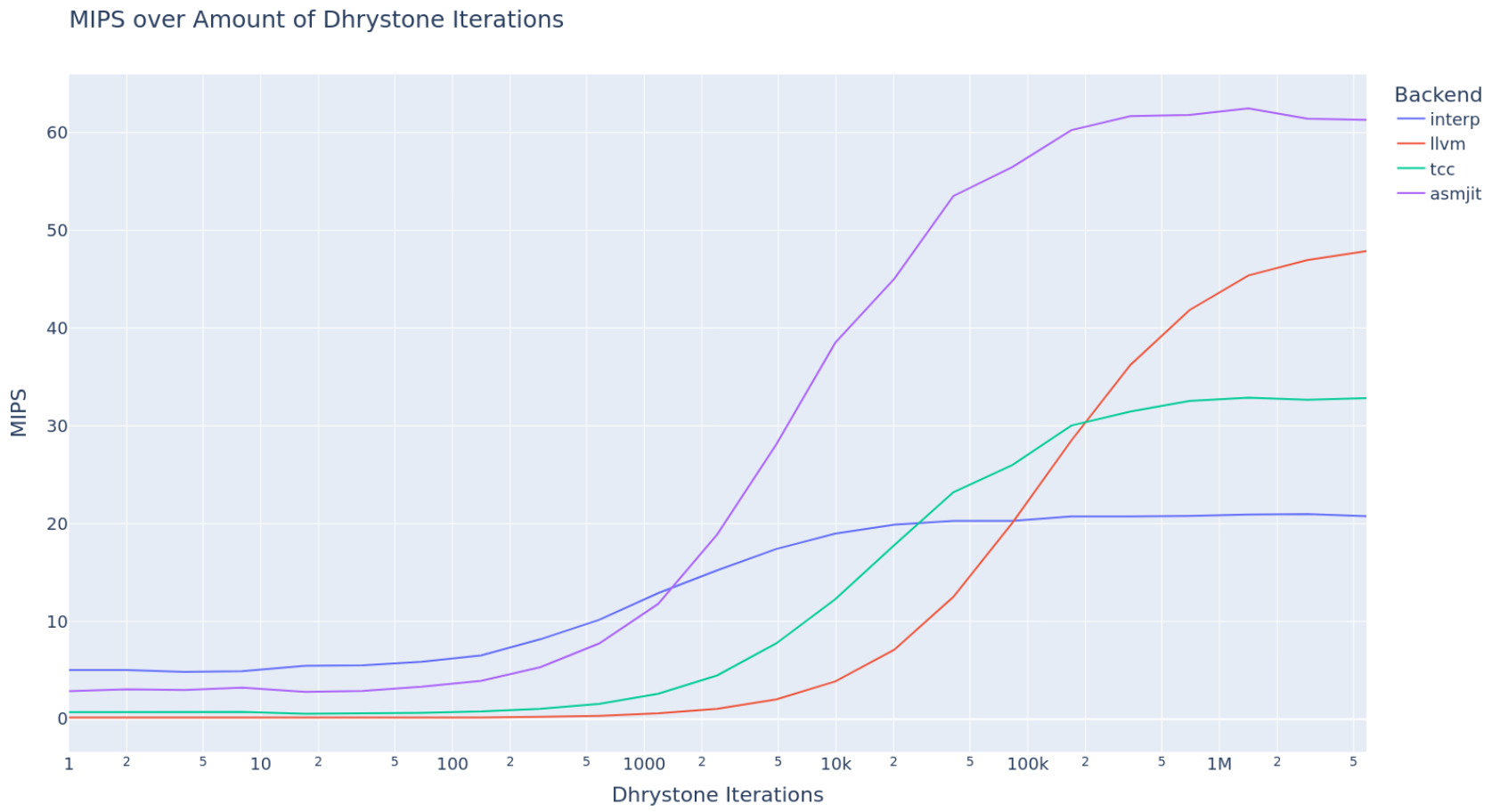
DBT-RISE - Integration

- Works in any TLM2 based tool environment, e.g. Platform Architect

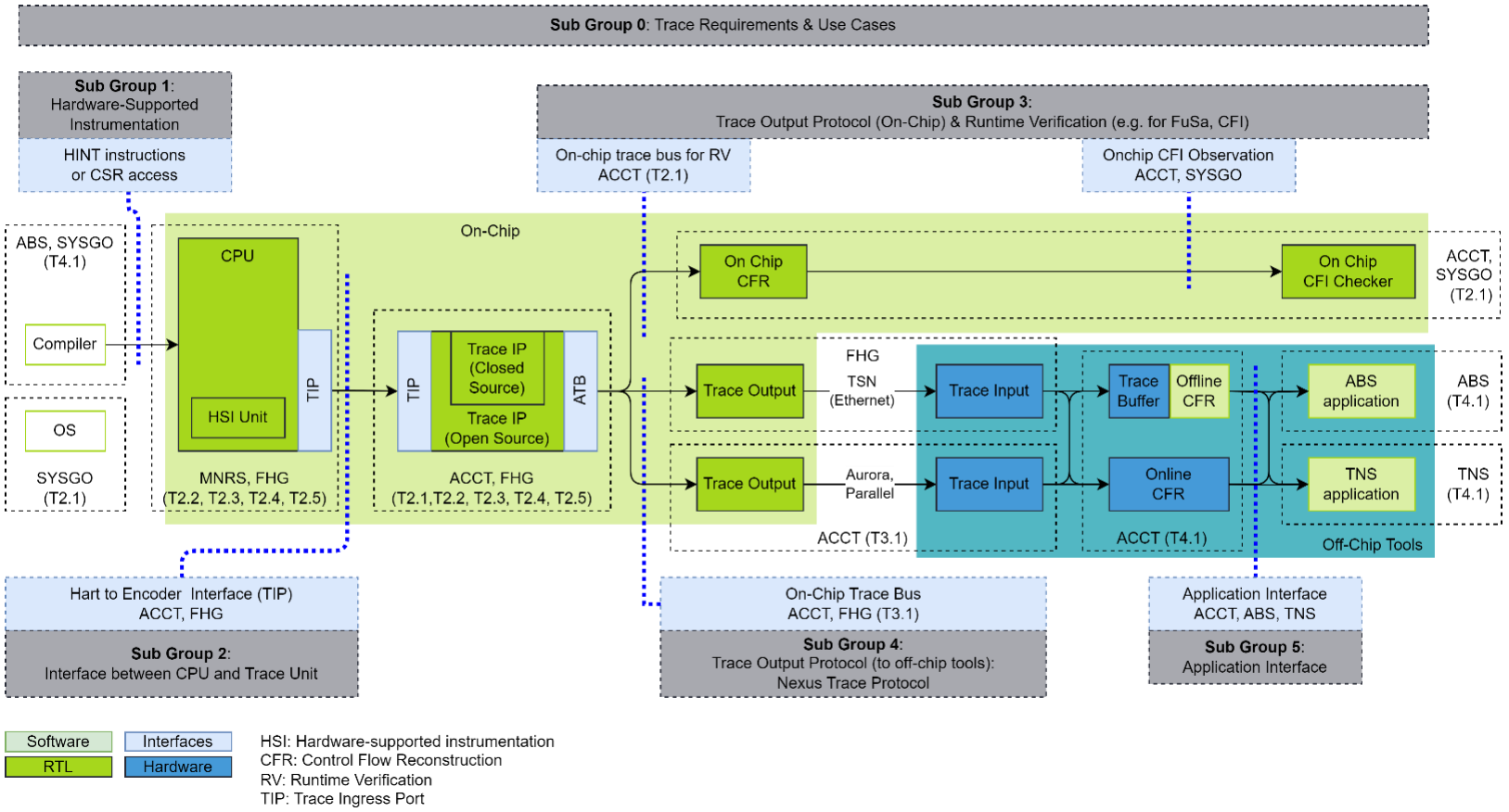


DBT-RISE - Backends

- Backend defines execution speed
  - Each curve shows a different backend
- Speed is measured as MIPS as function of iterations over benchmark Dhystone
- JIT techniques optimize SW sections which are executed repeatedly



C-Trace Development within TRISTAN



AbsInt

ACCEMIC TECHNOLOGIES

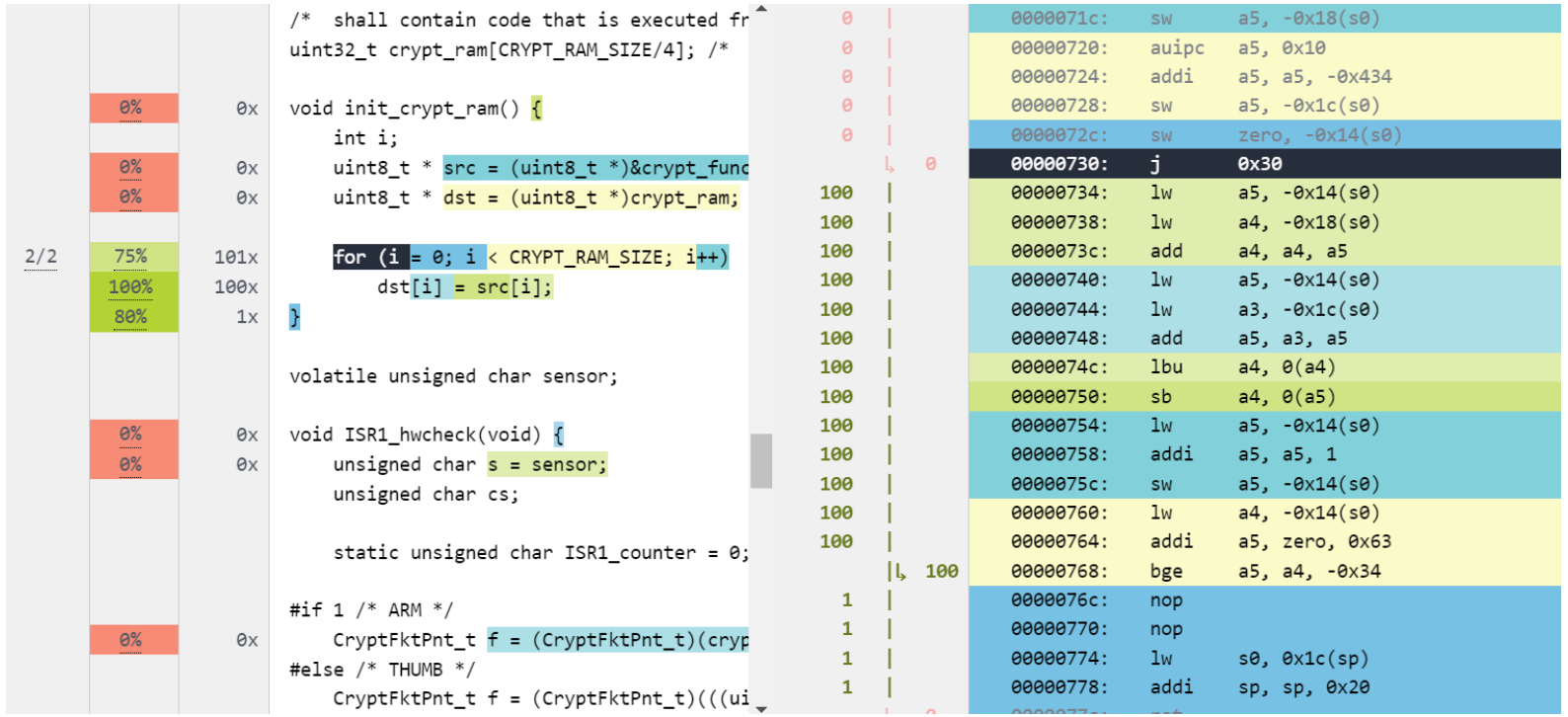
Fraunhofer IPMS

MINRES TECHNOLOGIES

TENSOR EMBEDDED GMBH

SYSGO EMBEDDING INNOVATIONS

Code Coverage Measurement



```
                    /*  shall contain code that is executed fr
                    uint32_t crypt_ram[CRYPT_RAM_SIZE/4]; /*

      0%      0x    void init_crypt_ram() {
                        int i;
      0%      0x        uint8_t * src = (uint8_t *)&crypt_func
      0%      0x        uint8_t * dst = (uint8_t *)crypt_ram;

2/2   75%     101x        for (i = 0; i < CRYPT_RAM_SIZE; i++)
      100%    100x            dst[i] = src[i];
      80%     1x    }

                    volatile unsigned char sensor;

      0%      0x    void ISR1_hwcheck(void) {
      0%      0x        unsigned char s = sensor;
                        unsigned char cs;

                        static unsigned char ISR1_counter = 0;

                    #if 1 /* ARM */
      0%      0x        CryptFktPnt_t f = (CryptFktPnt_t)(cryp
                    #else /* THUMB */
                        CryptFktPnt_t f = (CryptFktPnt_t)(((ui
```

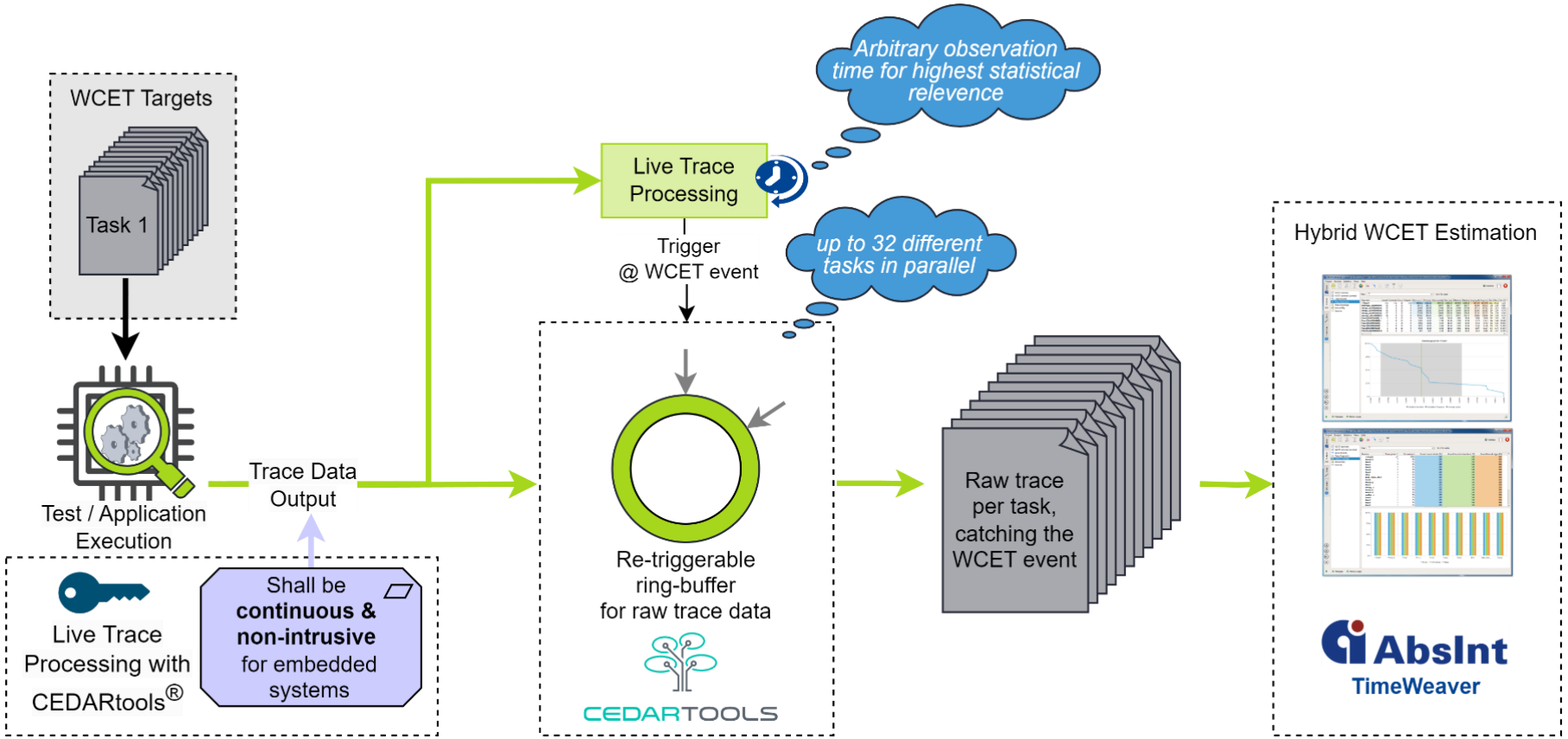
```
0     |     0000071c:  sw      a5, -0x18(s0)
0     |     00000720:  auipc   a5, 0x10
0     |     00000724:  addi    a5, a5, -0x434
0     |     00000728:  sw      a5, -0x1c(s0)
0     |     0000072c:  sw      zero, -0x14(s0)
      ↳ 0   00000730:  j       0x30
100   |     00000734:  lw      a5, -0x14(s0)
100   |     00000738:  lw      a4, -0x18(s0)
100   |     0000073c:  add     a4, a4, a5
100   |     00000740:  lw      a5, -0x14(s0)
100   |     00000744:  lw      a3, -0x1c(s0)
100   |     00000748:  add     a5, a3, a5
100   |     0000074c:  lbu     a4, 0(a4)
100   |     00000750:  sb      a4, 0(a5)
100   |     00000754:  lw      a5, -0x14(s0)
100   |     00000758:  addi    a5, a5, 1
100   |     0000075c:  sw      a5, -0x14(s0)
100   |     00000760:  lw      a4, -0x14(s0)
100   |     00000764:  addi    a5, zero, 0x63
      |↳ 100 00000768:  bge     a5, a4, -0x34
1     |     0000076c:  nop
1     |     00000770:  nop
1     |     00000774:  lw      s0, 0x1c(sp)
1     |     00000778:  addi    sp, sp, 0x20
```

- Non-intrusive
- Continuous

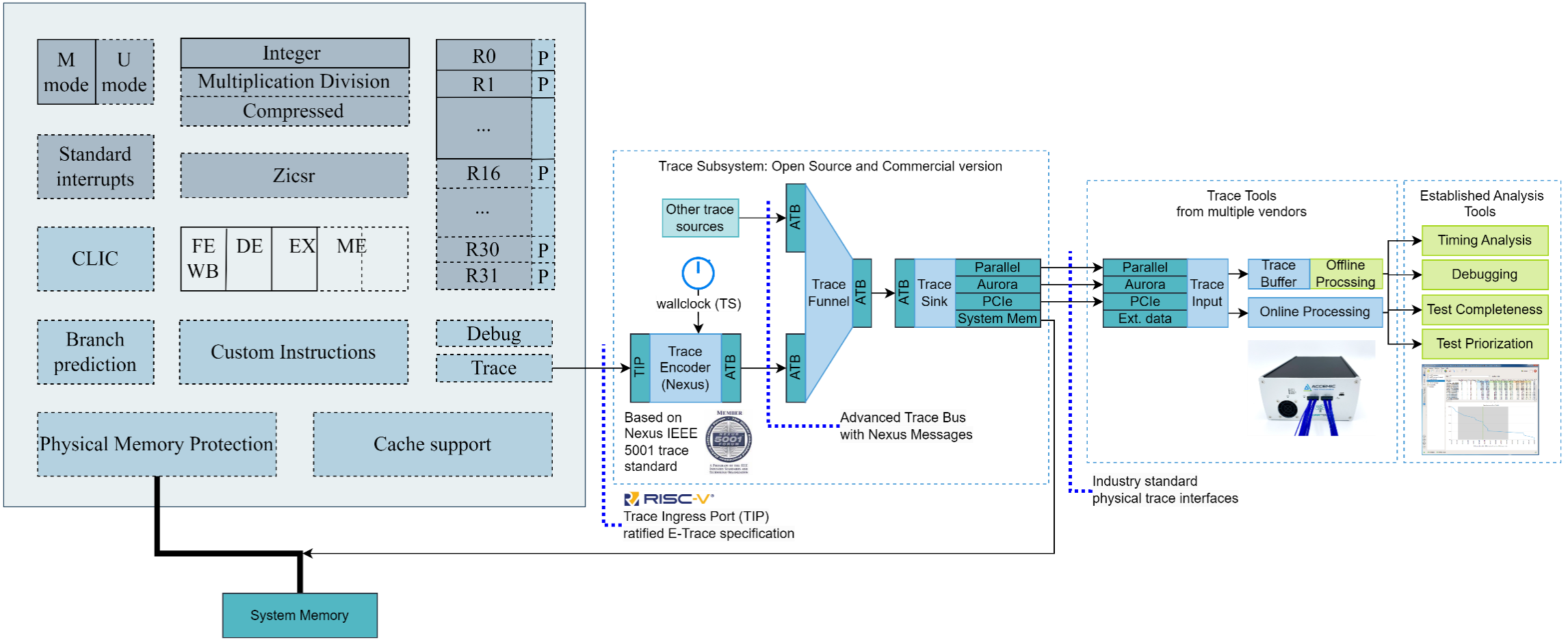
Code Coverage
measured by RISC-V Trace IP
on MINRES TGC core

Application / User Story

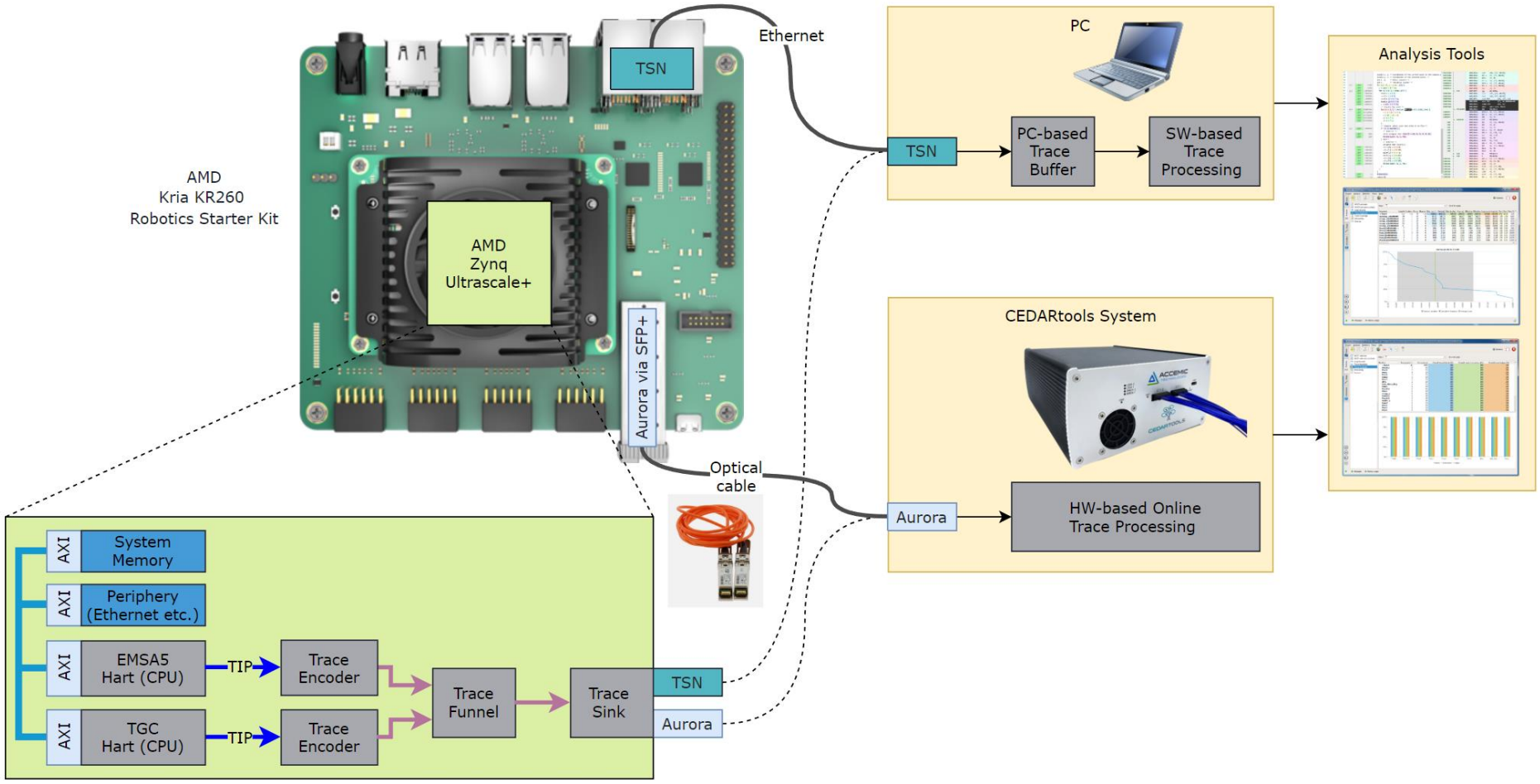
Timing Verification



C-Trace for MINRES TGC



C-Trace Demonstrator



What is SystemC Components Lib (SCC)

- SCC is an open-source productivity library
- As such it provides common functions, components and modules often needed in SystemC based models
- In-use at customers of MINRES and others
- Relies on C++11
- SystemC version >= 2.3.1
- Utilizes SystemC CCI

SCC Elements

- Common build system
- PySysC binding
- Modular structure

Module	Content
common	SystemC independent functions and components
sysc	Common SystemC and TLM related functionalities and components
components	VP components, mainly for LT modeling
bus_interfaces	TLM2.0 protocol abstractions and sockets as well as AT drivers, targets, and pin-level adapter

PySysC Usage

- Using Python for structural composition and runtime control
  - No preparation of shared libraries to be integrated
  - No need to have the sources of the code, even 3rd party binary only libraries can be used
  - Allows introspection of the interfaces and adaptation when needed
- Optional common functions in models allow convenient integration
- Standardization efforts within Accellera

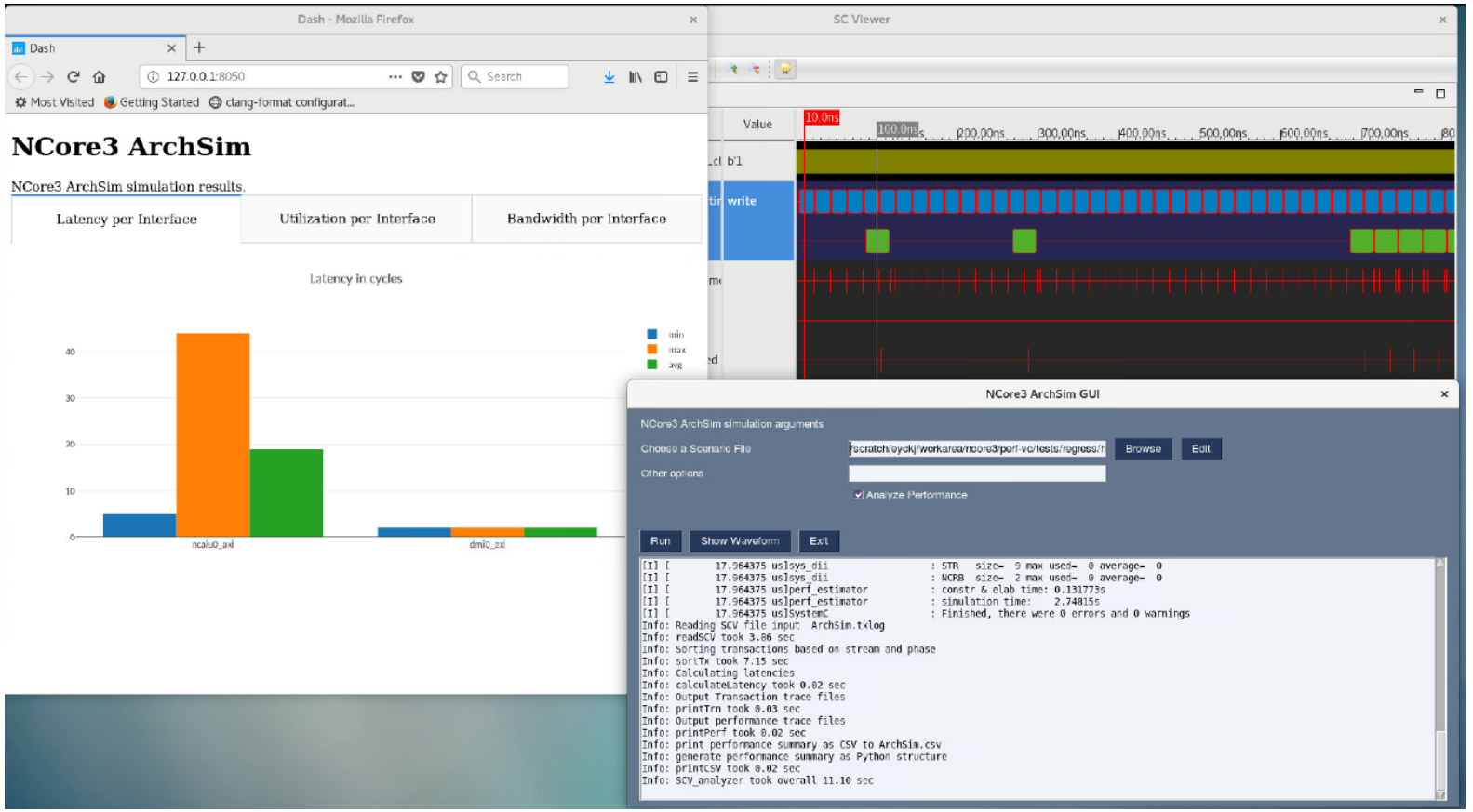
Result analysis and Visualization

Analysis of simulation results

- The goal of model simulations is the result analysis
- Type of analysis depends on accuracy of model
  - Latency, bandwidth only with cycle accurate/approximate models
  - Cache statistics only when caches are modeled
- Common, open-source formats for tracing are important
  - VCD for signals
  - Transaction tracing
- Reuse of existing frameworks for visualization, post processing and dashboarding
  - Dash, OpenSearch

Dashboards

- Trace analysis output can be used by open-source visualization tools like dash
- Python libraries allow simple analysis and even simulation control interfaces



- SystemC Components Library (SCC)
  https://github.com/Minres/SystemC-Components
- PySysC: Python bindings for SystemC, adopted by Accellera
  https://github.com/Minres/PySysC/
- CoreDSL: a language to describe ISAs for ISS generation and HLS of RTL implementation
  https://minres.github.io/CoreDSL/

Open-soure offerings

- DBT-RISE: a library for rapid implementation of ISS/VP using dynamic binary translation
  https://git.minres.com/DBT-RISE/
- DBT-RISE-RISCV: application of CoreDSL & DBT-RISE for RISCV
  https://github.com/Minres/DBT-RISE-RISCV
- Model code generation for VP based on industry standards like SystemRDL
  https://github.com/Minres/RDL-Editor
- Utility tools & libraries for VP modeling
  https://github.com/VP-Vibes/VPV-Peripherals

TRISTAN
RISC-V

TOGETHER FOR
RISC-V TECHNOLOGY
& APPLICATIONS

TRISTAN webpage

TRISTAN LinkedIn

TRISTAN Unified Access Page

TRISTAN has received funding from Chips Joint Undertaking (CHIPS-JU) under grant agreement nr. 101095947.

CHIPS JU receives support from the European Union's Horizon Europe's research and innovation programme and Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey