2024

DESIGN AND VERIFICATION™

DVCON

CONFERENCE AND EXHIBITION

EUROPE

MUNICH, GERMANY

OCTOBER 15-16, 2024

BOSCH

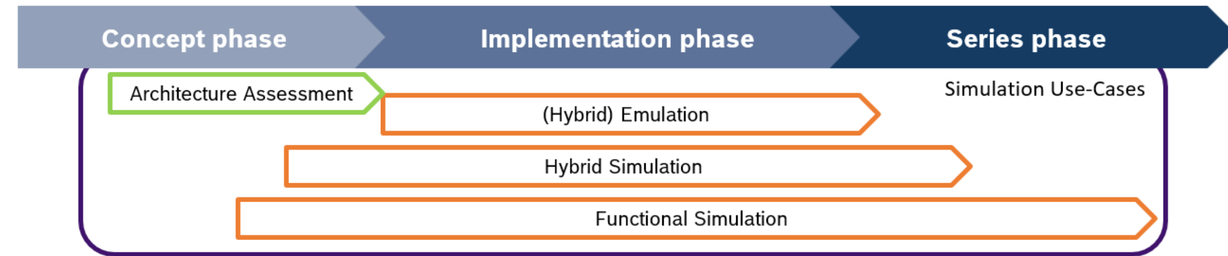
# A Software infrastructure for Hardware Performance Assessment


Ingo Feldner

Axel Sauer

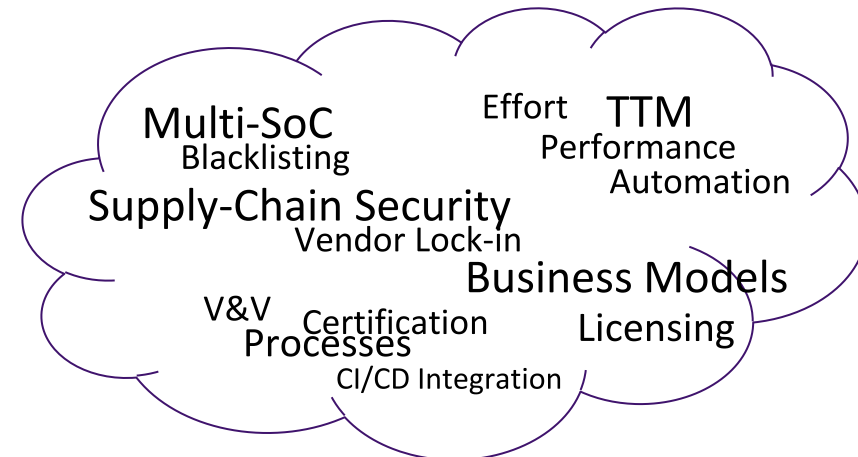
Tim Kraus

# Looking back…




- 15 years+ experience in applying models for SW development
- Covered various use-cases throughout the development cycle
- Added HW details for early IP verification
- Main model drawbacks encountered:
  - Availability
  - Cost
  - Mismatch to HW
- Performance Assessment poses even more challenges on application of simulation technology

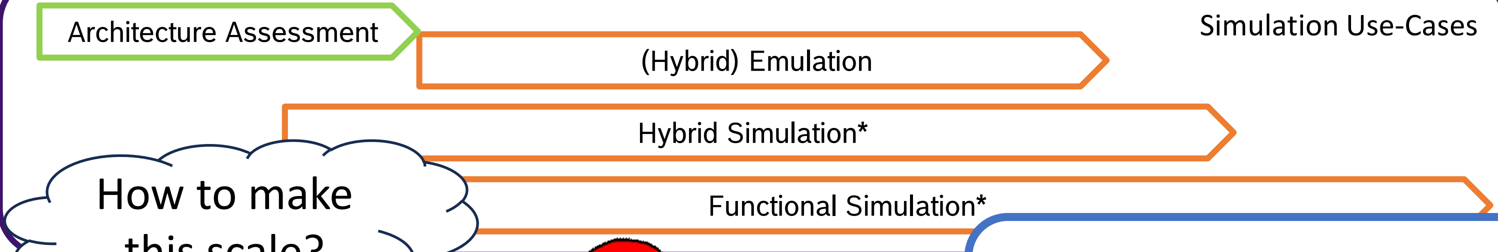
# It's not only the models

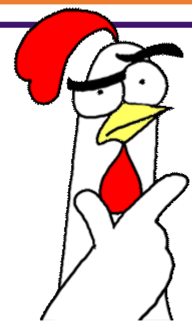



## Algorithm/Software

Increasing complexity, stagnating productivity
**Efficient programming required**

**Pre-Silicon:**
Early functional/timing Verification of embedded SW
**Post-Silicon:**
Increase observability and controllability, reduce cost
**Pre-Awarding:** Evaluate and ensure SW performance on different HW architectures



## Heterogeneous domain-specific Hardware

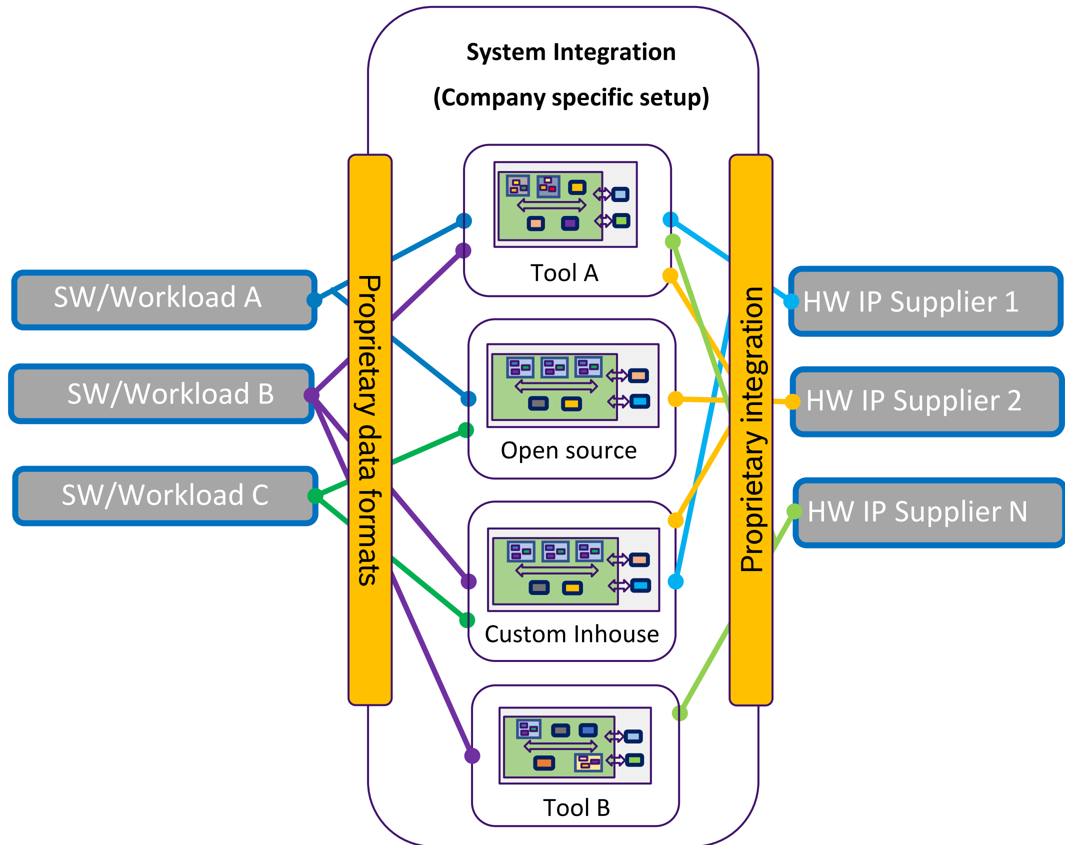
Specialization, Heterogeneity
**Dedicated HW optimization required**

Realistic HW assessment and comparison of suppliers



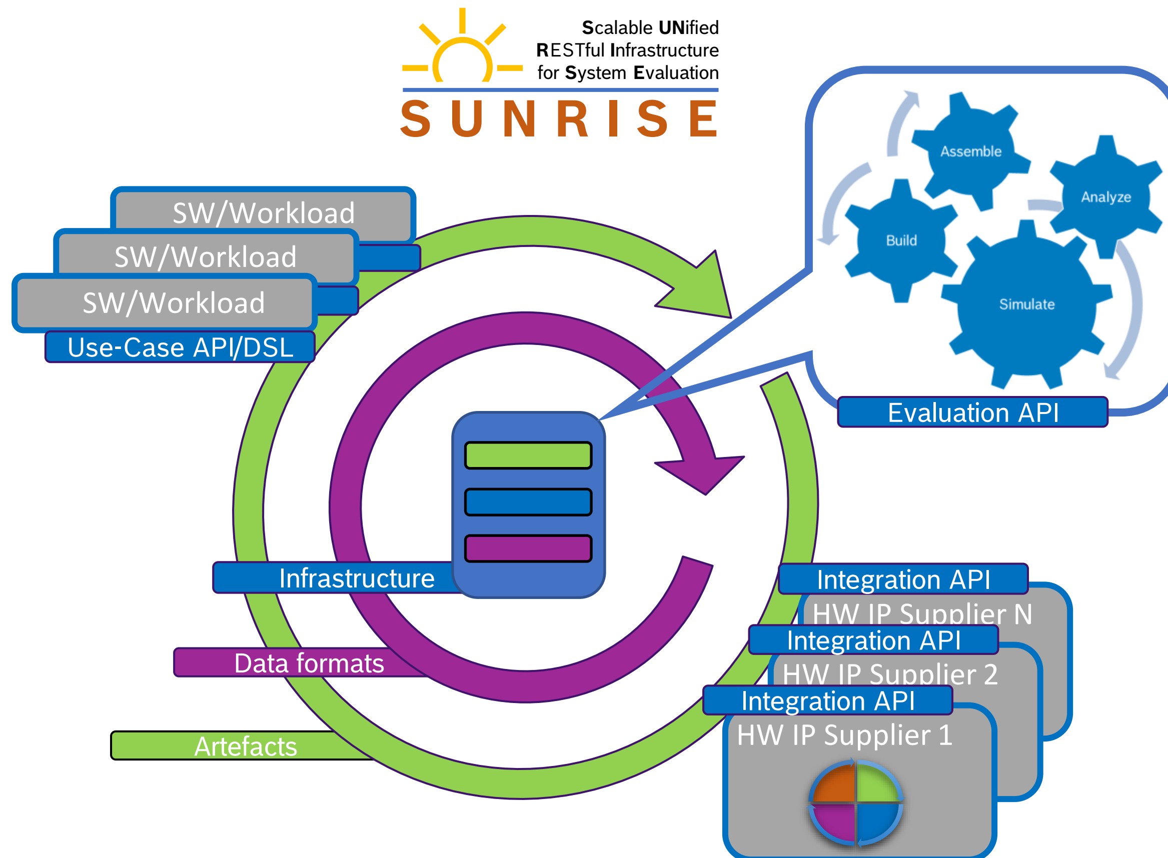
accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
MANNHEIM Flex KI
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Situation Today




- Fragmented simulation landscape hinders broad adoption due to proprietary data formats and interfaces
- Reduced effectiveness and efficiency with complex business setups and long contractual lead times
- Missing flexibility, high invest into proprietary vendor setup, resulting in high risk of lock-ins

# What do we need?




- Improved and highly automated integration on defined data formats
- Definition of secure execution, data access rights and collaboration models
- Support of multi-vendor, multi-tool setups tailored to customer needs
- Scalable business models fitting to on-demand use of simulation artefacts



accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
MANNHEIM FlexKI
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Seamless cross-vendor integration against open APIs/DSLs is key to scale simulation technology

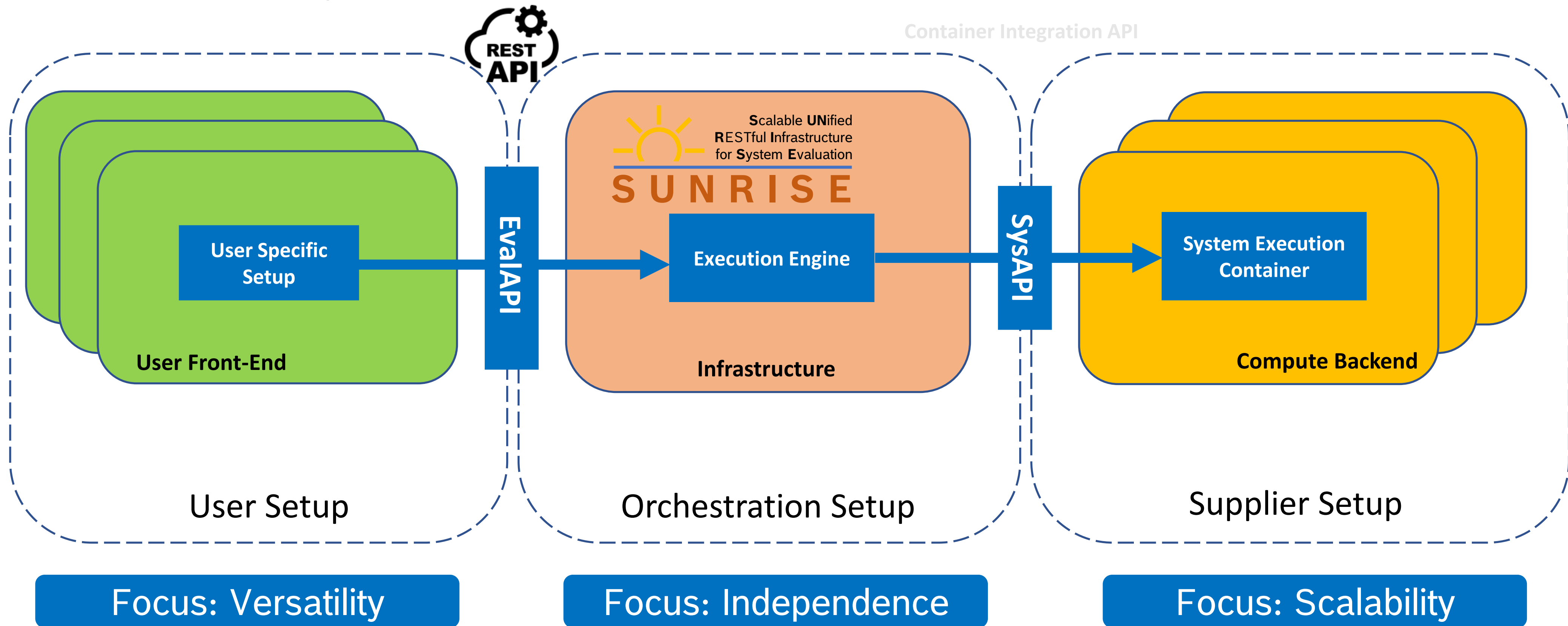
IMPORTANT:

*SUNRISE is not a tool, it‘s a set of APIs and data formats*



accellera SYSTEMS INITIATIVE

# The concept

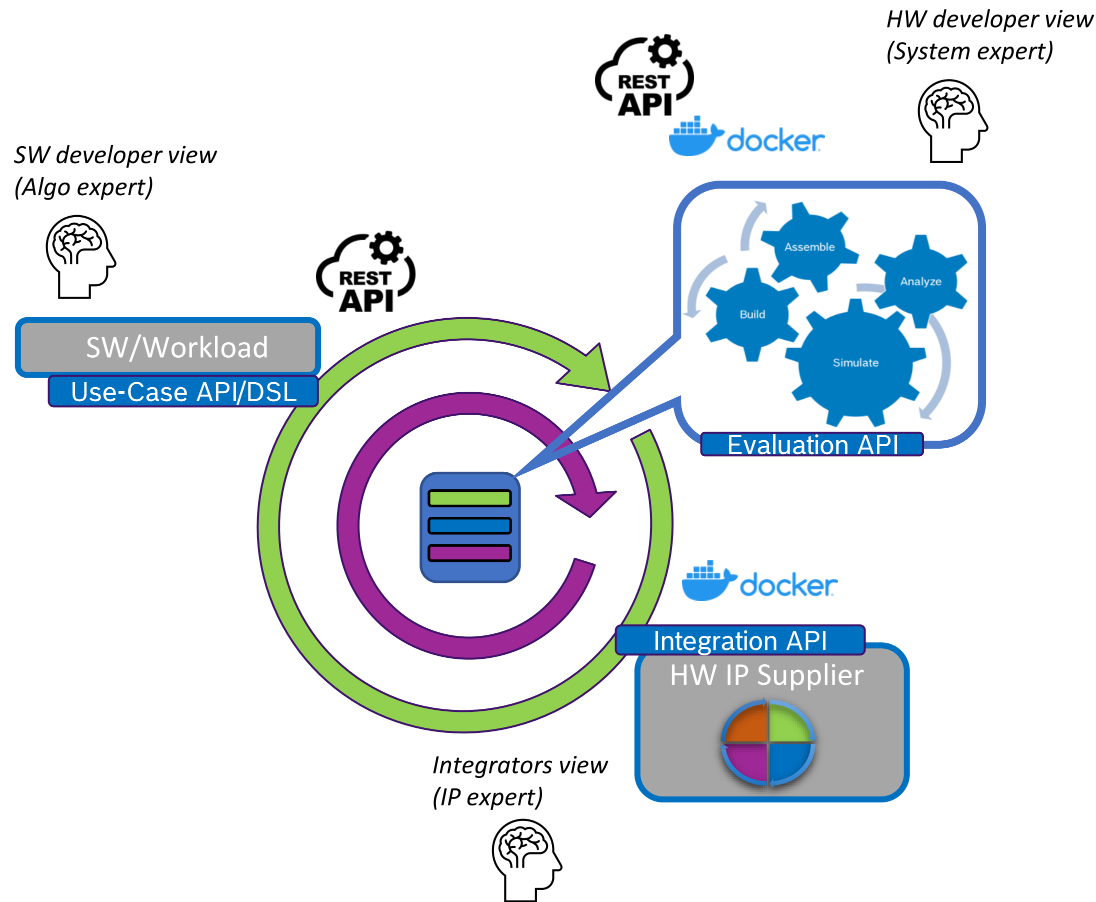



For more details on API and data formats attend SUNRISE paper presentation:
**Session 4D: Deployment of containerized simulations in an API-driven distributed infrastructure**



accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
MANNHEIM FlexKI
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# APIs and DSLs

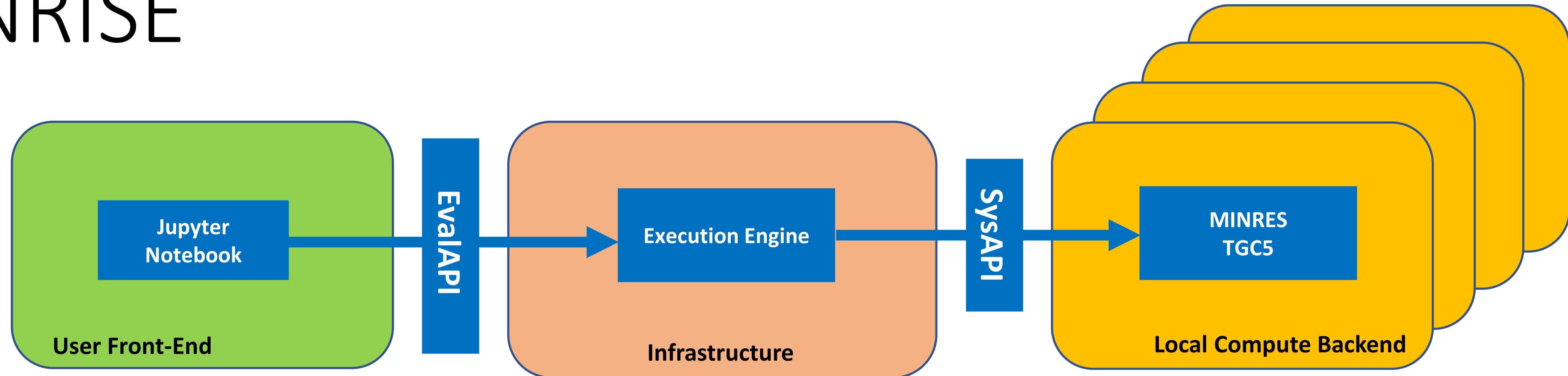



- Role-based approach for representing responsibilities
- REST APIs for exchange of data and web-based interaction
- Containers for defined integration into cloud and on-premise use-cases



accellera SYSTEMS INITIATIVE BOSCH TRISTAN RISC-V 2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Demo: Evaluation of MINRES TGC core with SUNRISE




- Demonstrating explicit calls to **EvalAPI** REST interface via the **SUNRISE Client** Python package
- Use-Case: Performance assessment of the MINRES RISC-V TGC5 VP core
  - Simulation container prepared by MINRES and integrated according to SysAPI requirements
  - Evaluation of performance done by RB with Jupyter Notebook as user frontend



accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Challenges

- Definition and acceptance of APIs and DSLs
- Secure Execution and data exchange
- Protection and consideration of IP rights
- Flexible business models and licensing
- Eligibility to access to results
- Co-existence of legacy setups (e.g., assembly, data formats)
- Separation of tooling and IP
- Debugging complex infrastructure setups

BOSCH

# What do we have?

- Integrator API:
  - Commercial and open-source integrations available
  - Formats for describing Systems and Configurations of platforms
  - Integration user guide
- Evaluation API:
  - Simulation of containerized platforms
  - Definition of basic result formats
  - On-premise and cloud services used

# What‘s next?

- RB is currently investigating possibilities for disclosure of SUNRISE
- Adding more partners willing to contribute and build community. Interested?
- Continue work on assembly and analysis APIs together with partners
- Realizing PoCs by adding more IPs to demonstrate scalability and efficiency

# Summary

- The goal of SUNRISE is to enable an efficient, scalable and technology-agnostic methodology by applying established SW methods to HW assessment
- Defined and reproducible ways of collaboration on premise and/or in the cloud
- SUNRISE applies modern SW techniques to improve the timely application of models and tools throughout the design process
- SUNRISE requires an open-source mindset focused on flexibility and ease of integration of existing IP and tool solutions



accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
MANNHEIM FlexKI
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Questions

This work has been developed in the project MANNHEIM-FlexKI. MANNHEIM-FlexKI is funded by the German Ministry of Education and Research (BMBF) (reference numbers: 01IS22086A-L). The authors are responsible for the content of this publication.

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 and including top-up funding by Federal Ministry of Education and Research, BMBF (Germany).

BOSCH

BOSCH

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024

SYNOPSYS®

# Efficient Workflow using Verilator for Processor Benchmarking in SystemC-based Automotive SoC Platforms

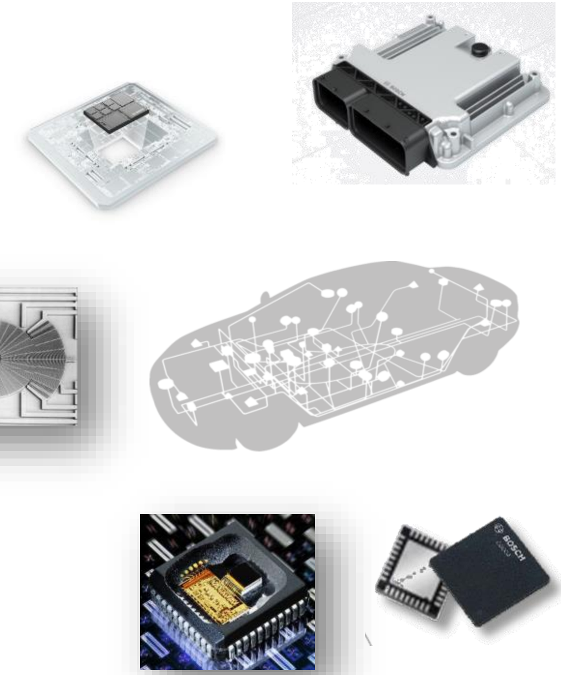
Johannes Sanwald, Andreas Mauderer, Mohammad Badawi, Javier Castillo, Jan-Hendrik Oetjens

Andreas Wieferink, Maryam Keeley, Tim Kogel

# Motivation & Goal

Motivation

- Increasing complexity of software in automotive edge devices
- Variety of processor IP through emergence of new architectures like RISC-V



Goal

- Rapid, efficient, and precise performance assessment and design exploration
- Usable with all processor cores, independent of vendor and architecture



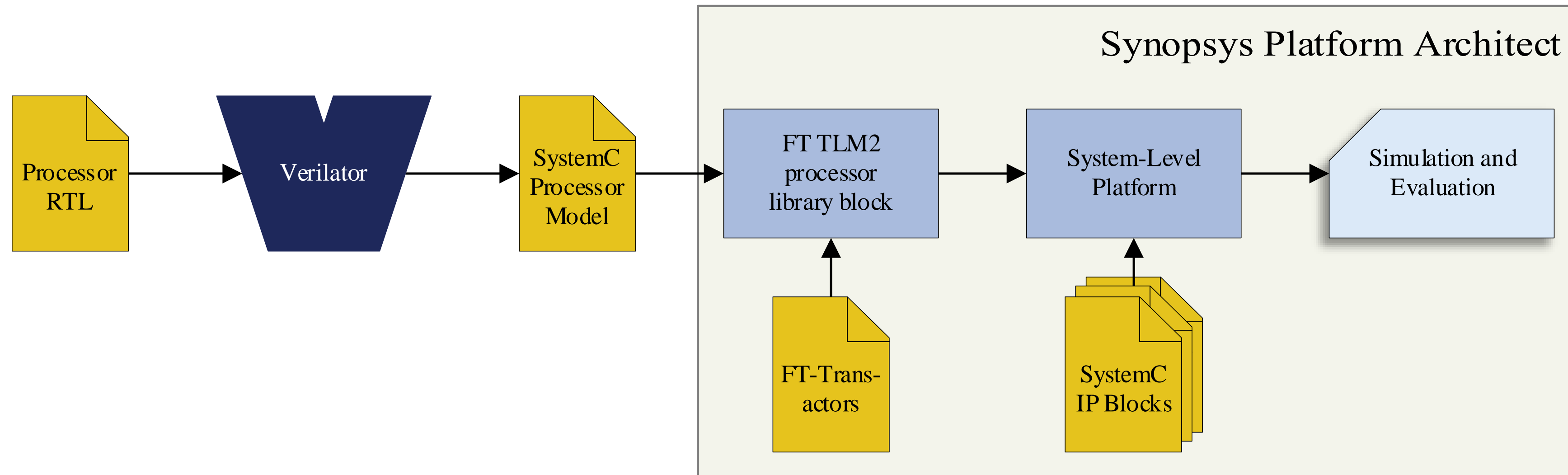
accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
SYNOPSYS®
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# State of the Art

- Full RTL simulation
  - ➕ Cycle accurate
  - ➖ High effort to integrate processors and to model peripherals, low simulation speed
- Co-simulation of RTL and SystemC
  - ➕ High accuracy, easier modeling of peripherals
  - ➖ High maintenance and integration effort for two simulation environments
- SystemC simulation of ISS or SystemC processor model
  - ➕ Established approach: Addressing fast, yet accurate, architecture exploration
  - ➕ Easy integration and modeling of peripherals
  - ➖ Processor integration still poses obstacles:
    - Interfaces not standardized, different TLM protocols, no accurate timing
    - Limited availability of cycle-accurate models

# Approach

- Workflow integrates verilated SystemC model of vendor supplied RTL
  - Addresses limited availability of vendor supplied SystemC models
  - Enables timing accurate communication
  - Compatible with all processors implementing AMBA-compliant interfaces

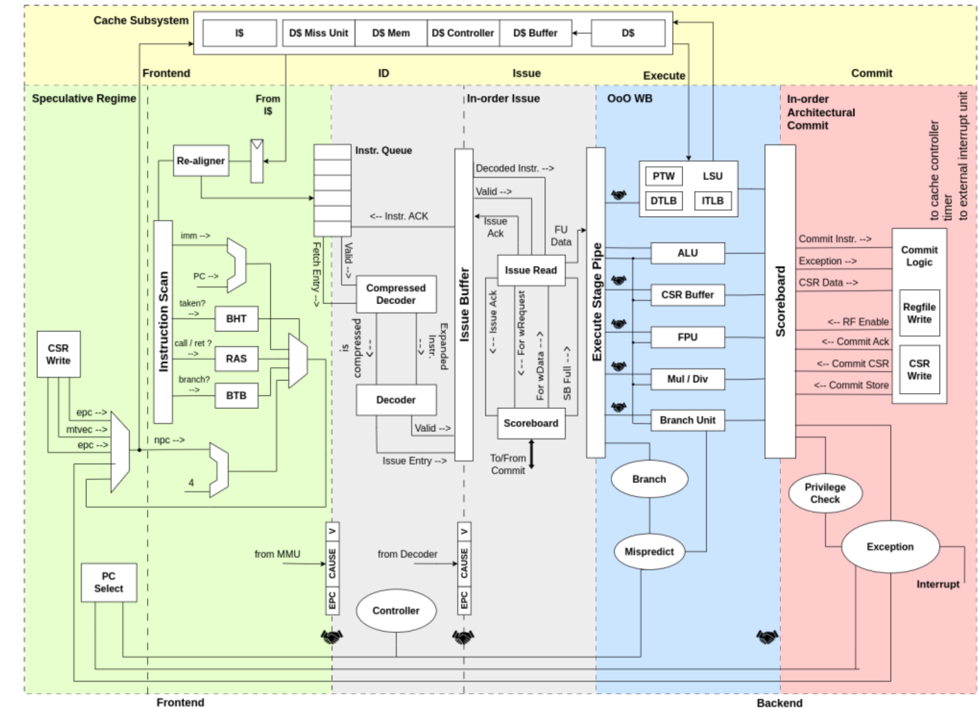





accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
SYNOPSYS®
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Application Example – CVA6 Dhrystone Benchmarking

- Open-source RISC-V core, maintained by OpenHW Group
- RV64-IMAC, 6 stage in-order pipeline
- I- and D-caches, with write-through policy
- AXI5 Interface for connection to memory and system
- Dhrystone Benchmark as proof of concept




"CVA6 RISC-V CPU," OpenHW Group, [Online]. Available: https://github.com/openhwgroup/cva6. [Accessed 22 June 2024]



accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
SYNOPSYS®
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Step 1 – Verilator and CVA6

- CVA6 exposes SystemVerilog packed structs as interface. We had to create a wrapper to expose the AMBA AXI bus signals as dedicated pins instead.
- Use Verilator to generate a SystemC model of the RISC-V core CVA6

```
// AXI types
parameter type axi_ar_chan_t = struct packed {
  logic [CVA6Cfg.AxiIdWidth-1:0]   id;
  logic [CVA6Cfg.AxiAddrWidth-1:0] addr;
  axi_pkg::len_t                   len;
  axi_pkg::size_t                  size;
  axi_pkg::burst_t                 burst;
  logic                            lock;
  axi_pkg::cache_t                 cache;
  axi_pkg::prot_t                  prot;
  axi_pkg::qos_t                   qos;
  axi_pkg::region_t                region;
  logic [CVA6Cfg.AxiUserWidth-1:0] user;
},
parameter type axi_aw_chan_t = struct packed {
  logic [CVA6Cfg.AxiIdWidth-1:0]   id;
  logic [CVA6Cfg.AxiAddrWidth-1:0] addr;
  axi_pkg::len_t                   len;
  axi_pkg::size_t                  size;
  axi_pkg::burst_t                 burst;
  logic                            lock;
  axi_pkg::cache_t                 cache;
  axi_pkg::prot_t                  prot;
  axi_pkg::qos_t                   qos;
  axi_pkg::region_t                region;
  axi_pkg::atop_t                  atop;
  logic [CVA6Cfg.AxiUserWidth-1:0] user;
},
```

```
////output noc_req_t noc_req_o,
//axi_aw_chan_t aw;
output logic [CVA6Cfg.AxiIdWidth-1:0]   axi_aw_id,
output logic [CVA6Cfg.AxiAddrWidth-1:0] axi_aw_addr,
output axi_pkg::len_t                   axi_aw_len,
output axi_pkg::size_t                  axi_aw_size,
output axi_pkg::burst_t                 axi_aw_burst,
output logic                            axi_aw_lock,
output axi_pkg::cache_t                 axi_aw_cache,
output axi_pkg::prot_t                  axi_aw_prot,
output axi_pkg::qos_t                   axi_aw_qos,
output axi_pkg::region_t                axi_aw_region,
output axi_pkg::atop_t                  axi_aw_atop,
output logic [CVA6Cfg.AxiUserWidth-1:0] axi_aw_user,
// end axi_aw_chan_t aw,
output logic          axi_aw_valid,
//axi_w_chan_t  w,
output logic [CVA6Cfg.AxiDataWidth-1:0]     axi_w_data,
output logic [(CVA6Cfg.AxiDataWidth/8)-1:0] axi_w_strb,
output logic                                axi_w_last,
output logic [CVA6Cfg.AxiUserWidth-1:0]     axi_w_user,
//end axi_w_chan_t  w,
output logic          axi_w_valid,
output logic          axi_b_ready,
//axi_ar_chan_t ar,
output logic [CVA6Cfg.AxiIdWidth-1:0]   axi_ar_id,
output logic [CVA6Cfg.AxiAddrWidth-1:0] axi_ar_addr,
output axi_pkg::len_t                   axi_ar_len,
output axi_pkg::size_t                  axi_ar_size,
output axi_pkg::burst_t                 axi_ar_burst,
output logic                            axi_ar_lock,
output axi_pkg::cache_t                 axi_ar_cache,
output axi_pkg::prot_t                  axi_ar_prot,
output axi_pkg::qos_t                   axi_ar_qos,
output axi_pkg::region_t                axi_ar_region,
output logic [CVA6Cfg.AxiUserWidth-1:0] axi_ar_user,
```

# Step 2 – Synopsys Platform Architect Integration

1. Raw SystemC Import

Pin Accurate SystemC Processor Model

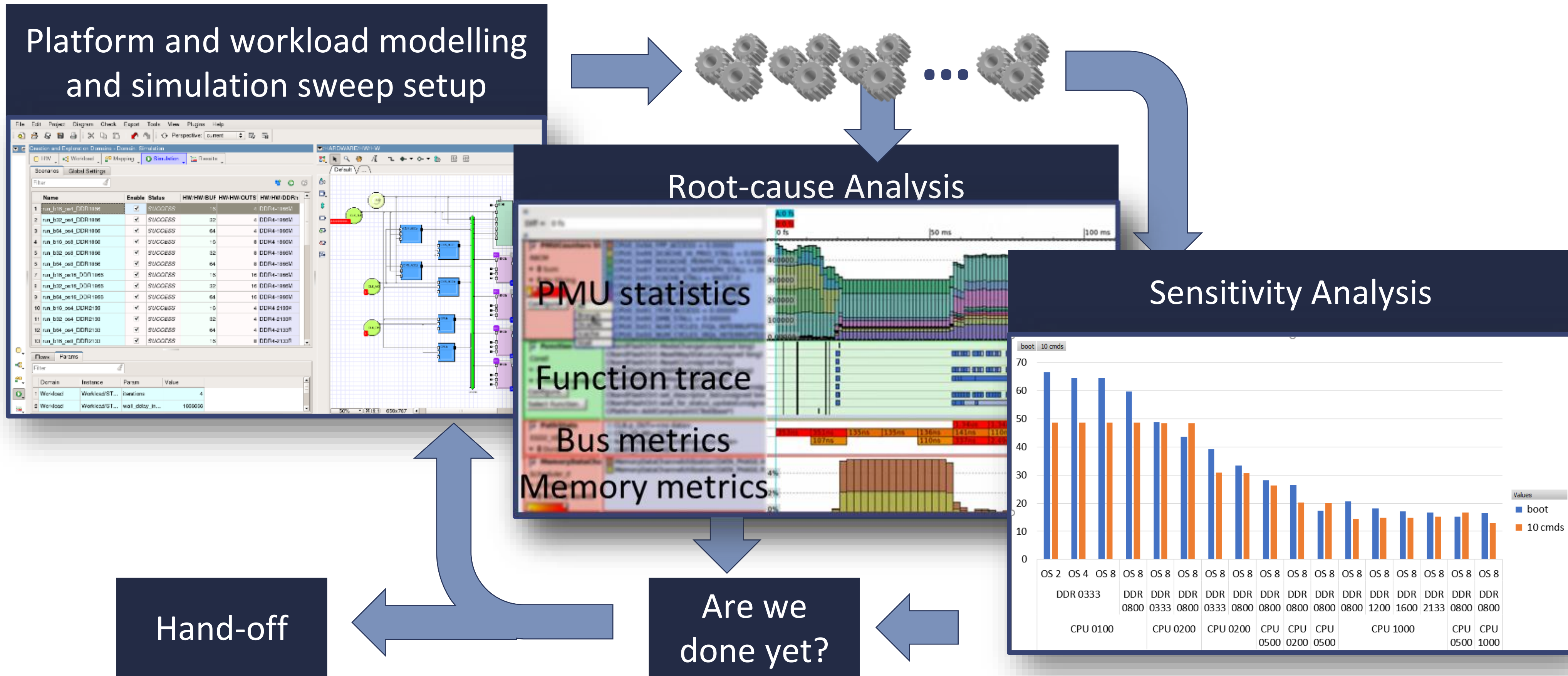
2. Automatic Block Refinement

FT TLM2 Processor Model



accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
SYNOPSYS®
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

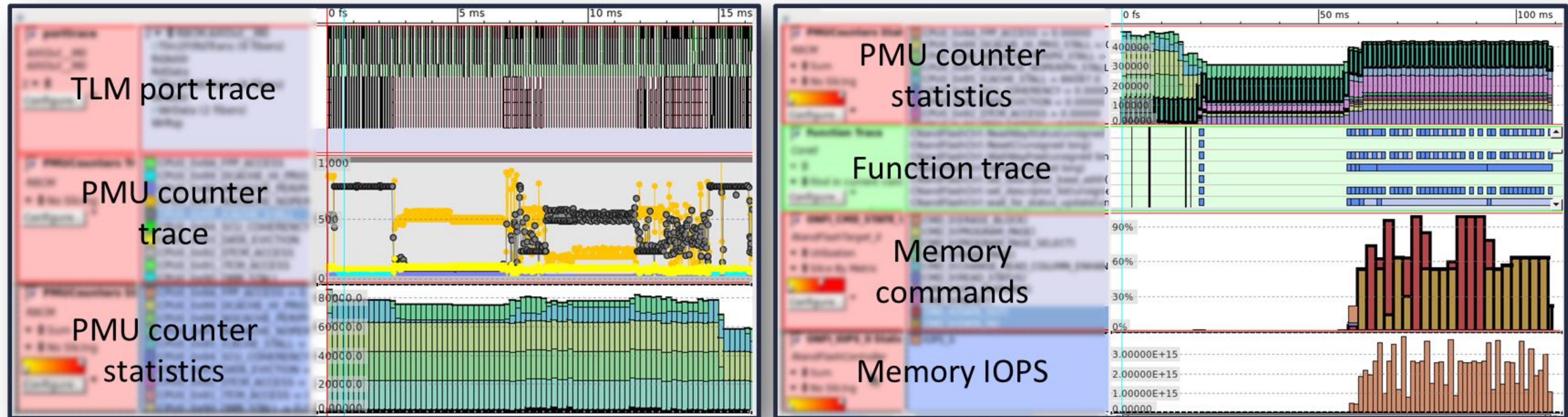
# Step 3 – Iterative Architecture Optimization Flow






accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
SYNOPSYS®
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Step 3 – Quantitative Performance Analysis and Optimization



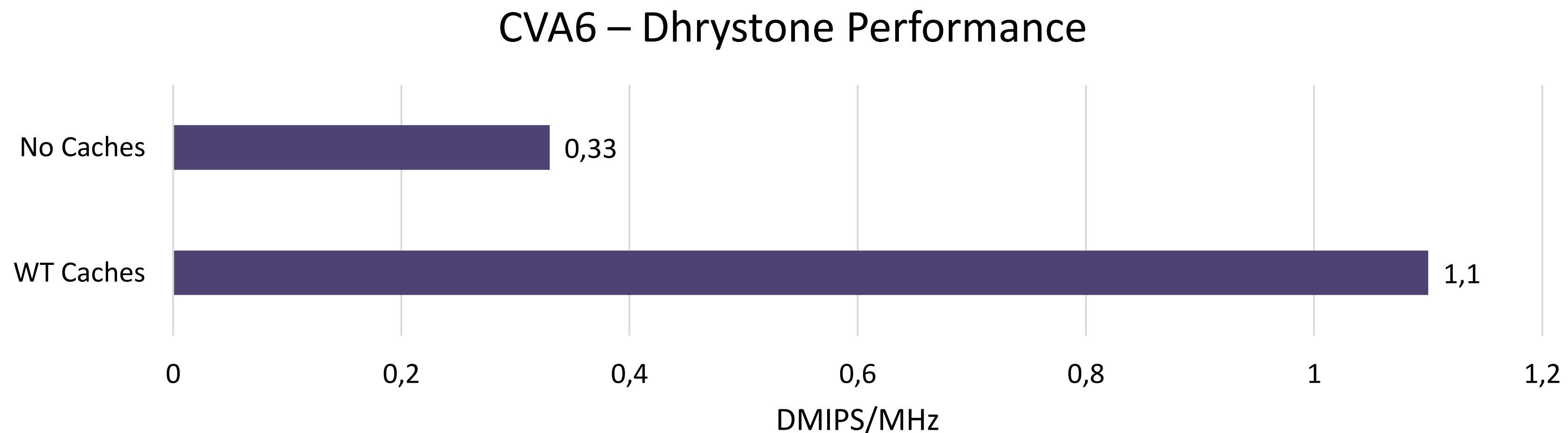

- Root-cause analysis of performance issues using traces and statistics
- Interpretation of analysis views from CPU performance counters, interconnect, memory and flash subsystem



accellera SYSTEMS INITIATIVE
BOSCH
TRISTAN RISC-V
SYNOPSYS®
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Application Example – Dhrystone Results

| CVA6 Configuration | Cycle Count for 10 Runs | DMIPS/MHz |
|---|---|---|
| No Caches | 17165 | 0,33 |
| Write-through Cache | 5167 | 1,10 |






accellera SYSTEMS INITIATIVE BOSCH TRISTAN RISC-V SYNOPSYS® 2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Summary

- Proposal of an approach for rapid and efficient architecture exploration
- SystemC as simulation environment because of its efficiency and ease of modeling for complex peripherals
- Verilator used for generating SystemC processor models with timing-accurate interfaces from vendor supplied RTL
- Synopsys Platform Architect as SystemC integration and simulation platform, which provides pin-level to TLM transactors, peripheral IP blocks, and simulation and analysis tools



accellera SYSTEMS INITIATIVE BOSCH TRISTAN RISC-V SYNOPSYS® 2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Outlook

- Study further processors and instruction set architectures
- Model full automotive system-architecture and compare with equivalent RTL simulation
- Integrate breakpoints and instruction stepping debug features

# Questions

## Acknowledgement

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 and including top-up funding by Federal Ministry of Education and Research, BMBF (Germany).

BOSCH

SYNOPSYS®

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
OCTOBER 15-16, 2024

# Developing performance models using SystemC

Rocco Jonack, MINRES Technologies

Eyck Jentzsch, MINRES Technologies

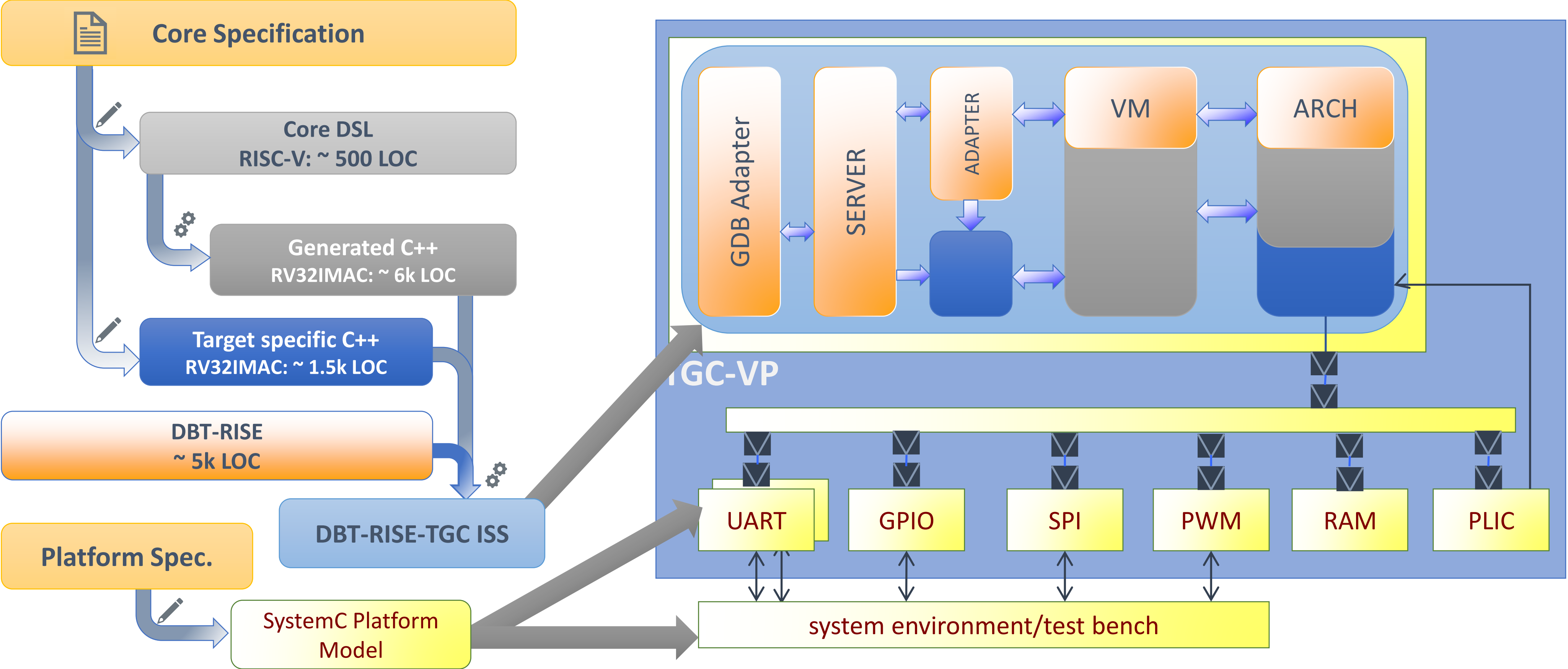
# Implementing ISS using DBT-RISE

- ***D***ynamic ***B***inary ***T***ranslation - ***R***etargetable ***IS***S ***E***nvironment
- An Open-Source C++ environment to implement instruction set simulators (ISS) e.g. using CoreDSL
- DBT-RISE-CORE contains the core elements of DBT-RISE and as such is intended to be part of a target project
- Different backends can be used to adapt to requirements
- Plugins system allows easy extension
- Easy to embed into SystemC based models

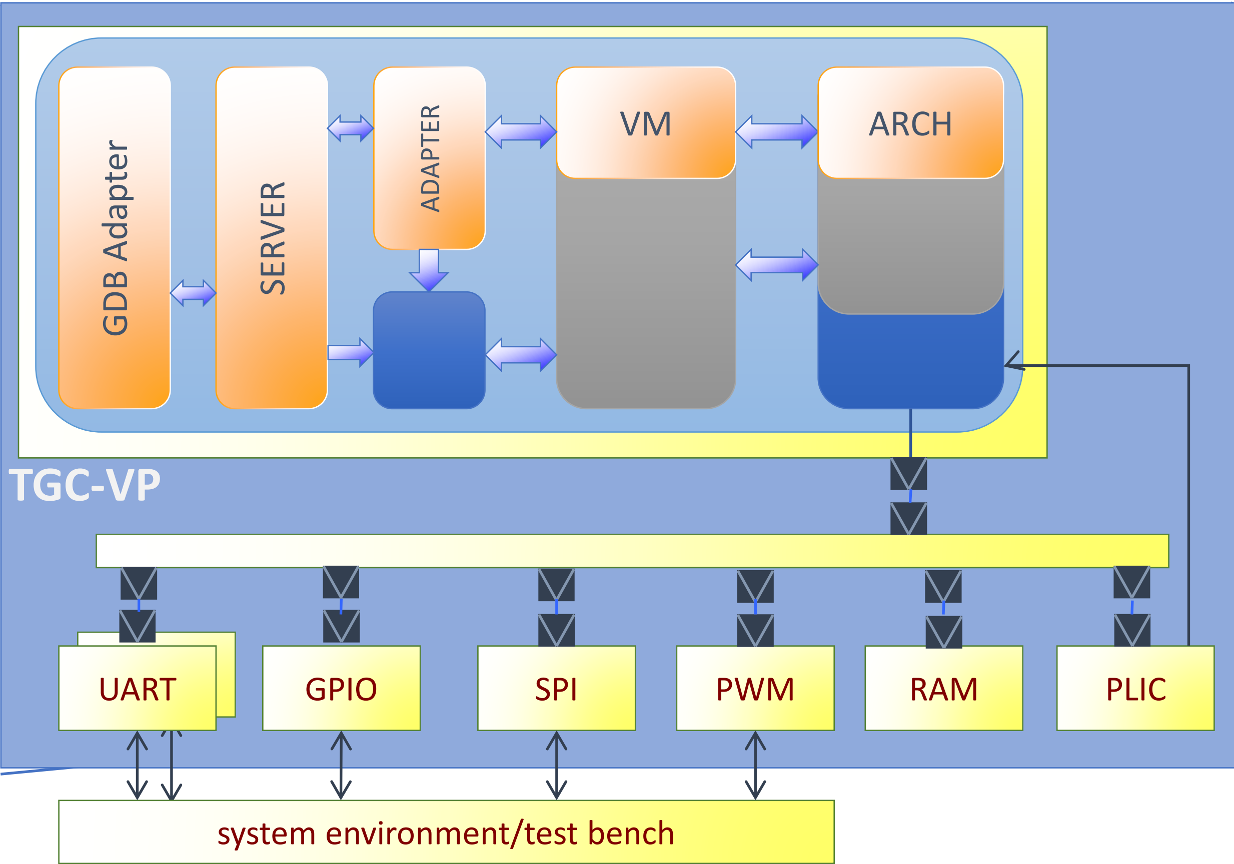

MINRES Technologies GmbH Confidential

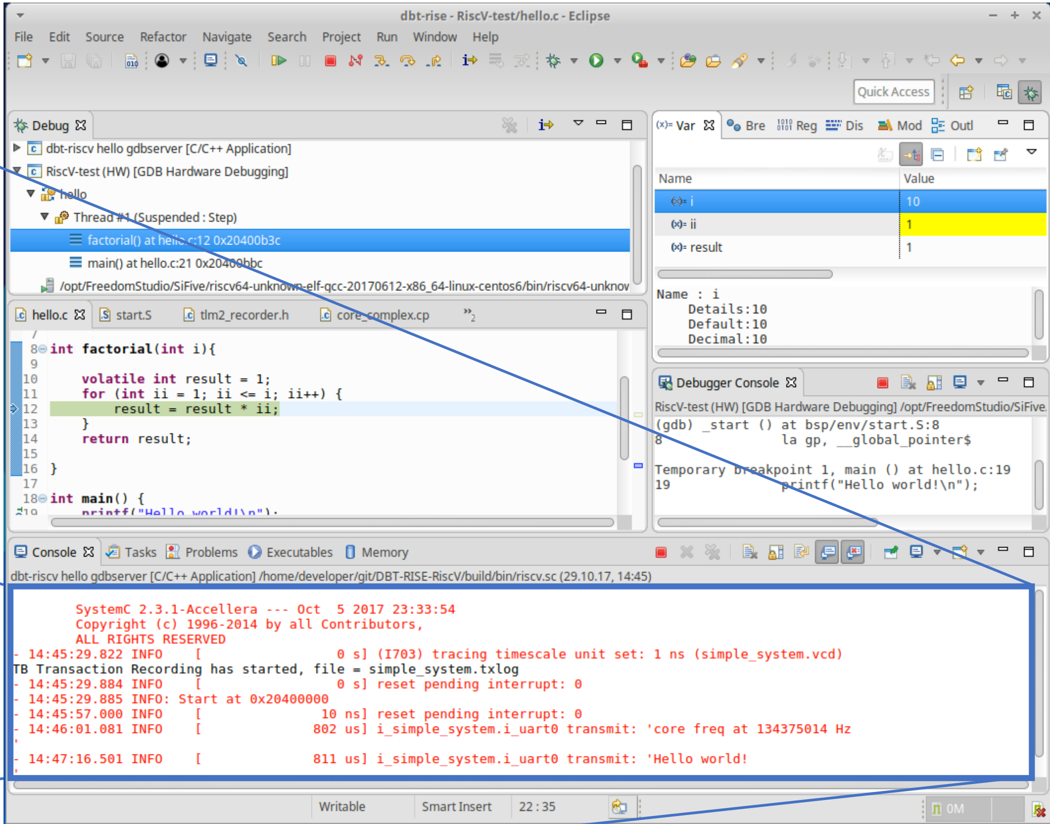
# DBT-RISE based Platform

Open-Source Infrastructure for Dynamic Binary Translation (jit compiling) for ISS



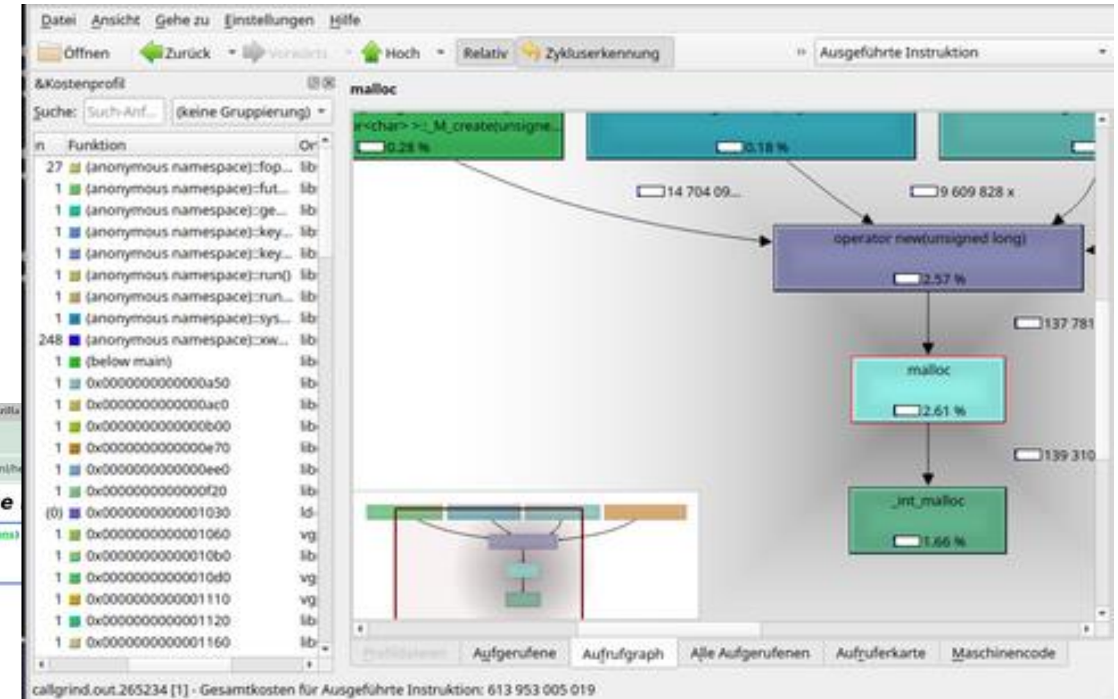
# DBT-RISE RISC-V VP



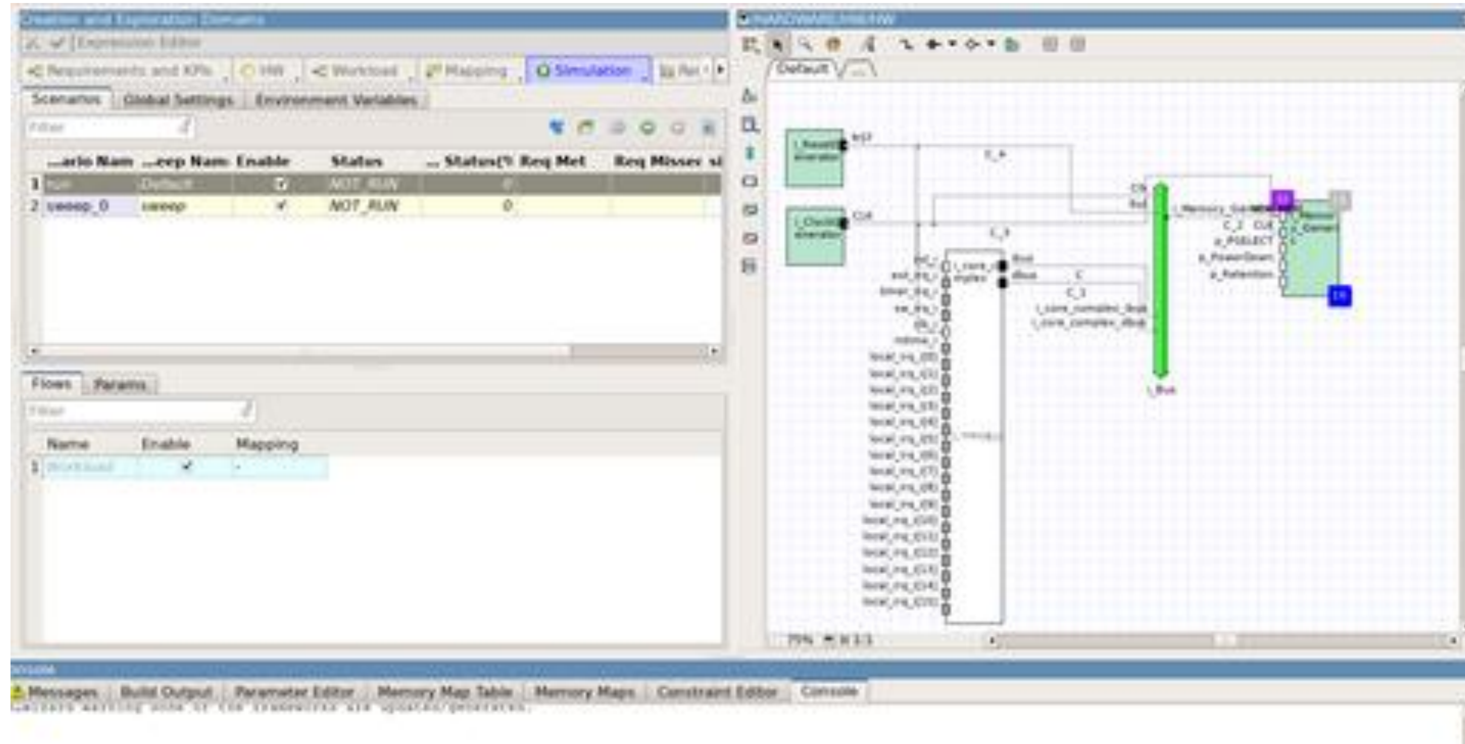



# DBT-RISE - Plugins

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



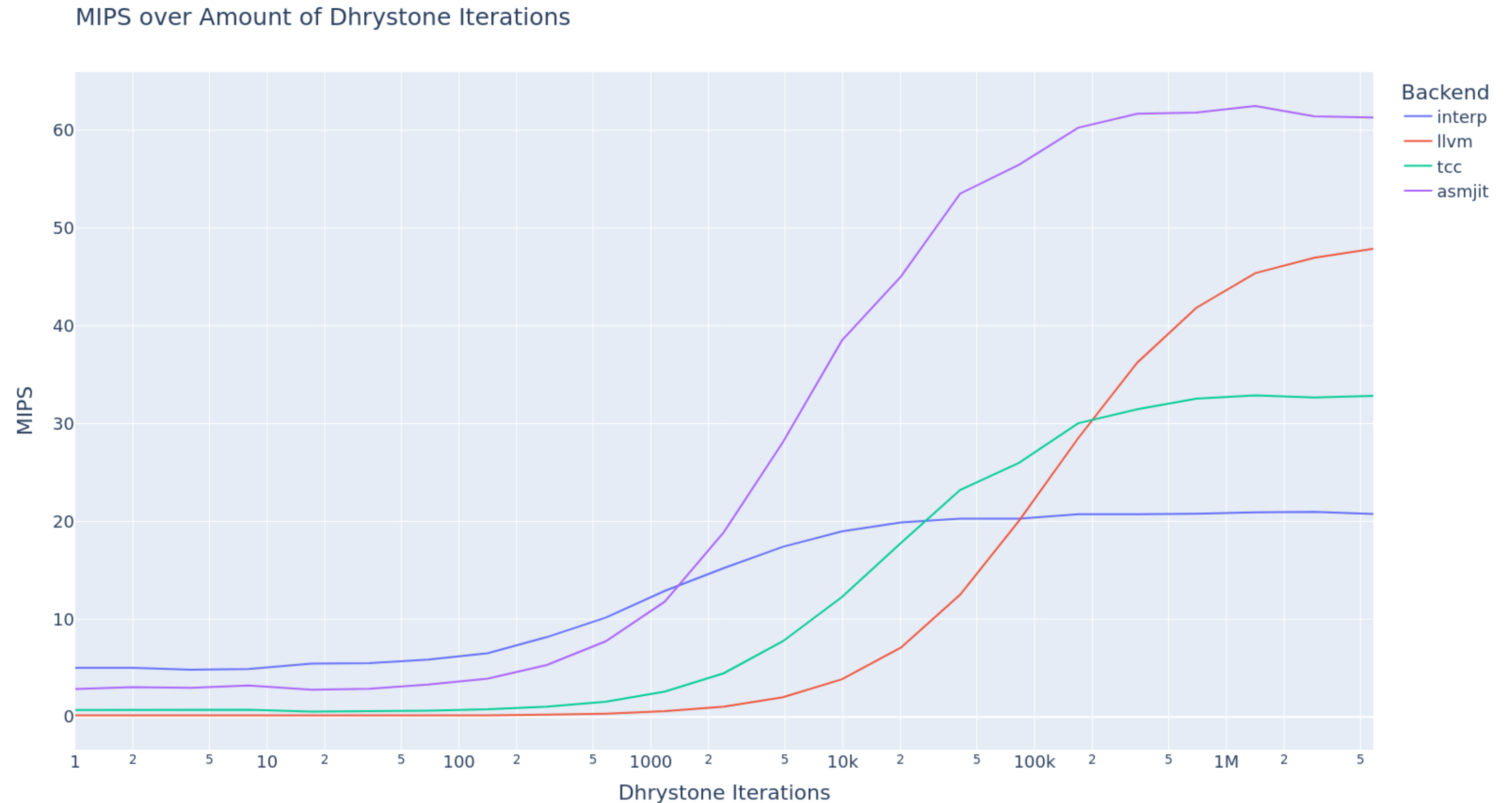
# DBT-RISE - Integration




- Works in any TLM2 based tool environment, e.g. Platform Architect
- Plugins allow tailored integration

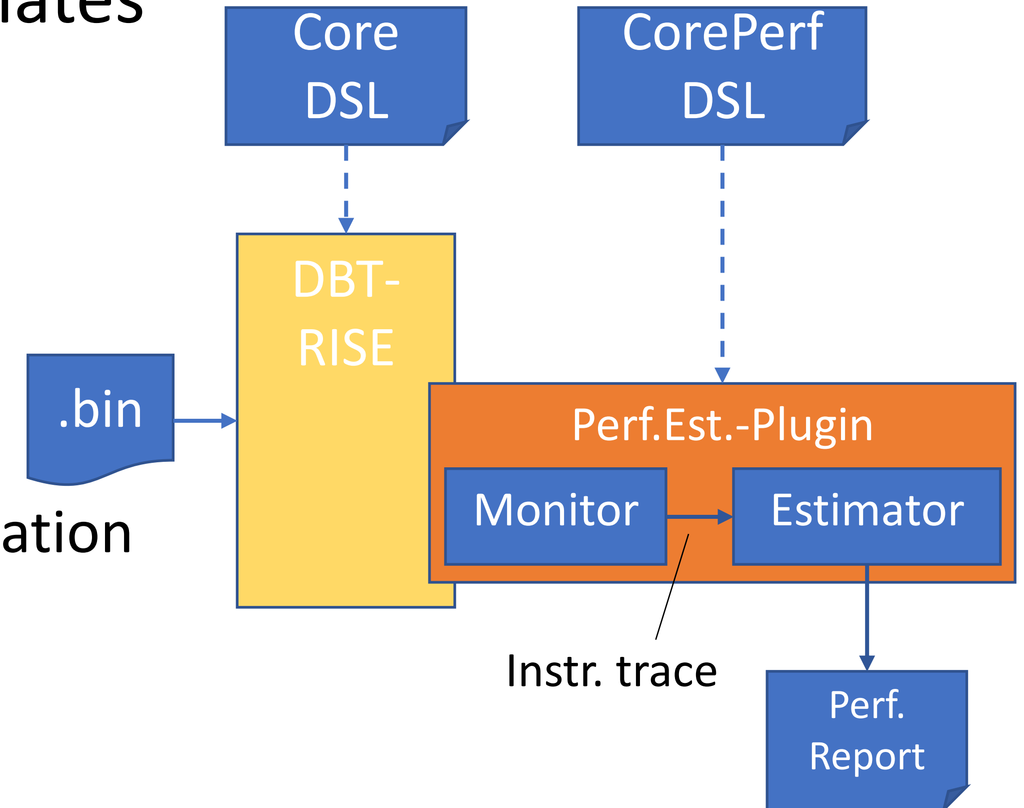
# DBT-RISE - Backends

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



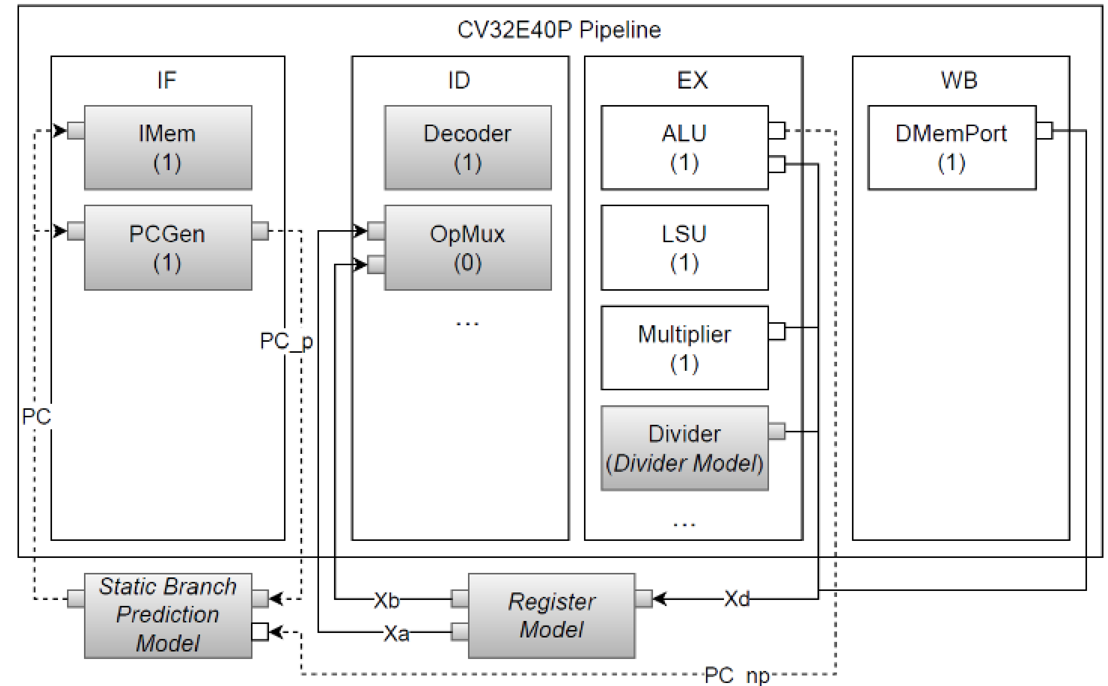
# Performance Estimation

- No accurate performance (timing) estimates for ISS
  - ISA-level model: Correct functional behavior
  - Microarchitecture not considered
- PerformanceEstimator-Plugin
  - Observes instruction trace
  - Estimates timing based on microarch. information
  - Retarget via CorePerfDSL
  - Accuracy >99% @ up to 24 MIPS



# CorePerfDSL

- Compact structural description
  - Non-functional
  - Focus on flexibility
  - *External models* to represent dynamic components (e.g. caches, branch pred.)
- Instruction-mapping to match components to instruction types
- Generator transforms to:
  - Single “max-plus” *scheduling function* for each instruction type
  - *Timing variables* to represent state of pipeline





accellera SYSTEMS INITIATIVE
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Interconnect Models

- TLM to allow interoperability
- SystemC Components library comes with AT level implementations of common on-chip protocols
- CCI for configuration
  - Provides a standard layer, which some tools build upon
- SCV or LWTR for transaction recording
  - Some waveform tools for visualization are available that build on top of them
  - Text based analysis possible based on structured format
- SCC library for common elements and logging format

# Memory Modeling

- Generic testbench component with AXI slave port
  - Configurable latency
  - No explicit behavior
  - Can be connected once modeled
- DRAMSys for improved accuracy in terms of performance

# System Composition using PySysC

- Python Binding for SystemC
- Allows to compose systems using Python
- Beyond support for structural construction, simulation control and dynamic model parametrization should be supported
- Due to broad availability of Python integrations plenty of libraries can be used and combined
  - Computational models using numpy/scipy etc.
  - UIs and cockpits using GTK, wxWidgets or Qt

# PySysC Example

1. Instantiation of a module
2. Instantiation of a templated module
3. Named signal connection
4. TLM2.0 socket connection
5. Simulation run

```
from cppyy import gbl as cpp
from cppyy.gbl import sc_core
from pysysc.structural import Connection, Signal, Module, Simulation
# loading required libraries
...
# instantiating modules
clk_gen = Module(cpp.ClkGen).create("clk_gen")                    ## (1)
initiator = Module(cpp.Initiator).create("initiator")
memories = [Module(cpp.Memory).create(name)
                for name in ["mem0", "mem1", "mem2", "mem3"]]
router = Module(cpp.Router[4]).create("router")                   ## (2)
# creating connections
clk = Signal("clk")
        .src(clk_gen.clk_o)
        .sink(initiator.clk_i)
        .sink(router.clk_i)                                       ## (3)
[clk.sink(m.clk_i) for m in memories]
Connection()
        .src(initiator.socket)
        .sink(router.target_socket)                               ## (4)
[Connection()
        .src(router.initiator_socket.at(idx))
        .sink(m.socket)
        for idx,m in enumerate(memories)]
# run simulation
sc_core.sc_start()                                                ## (5)
```

# Tracing

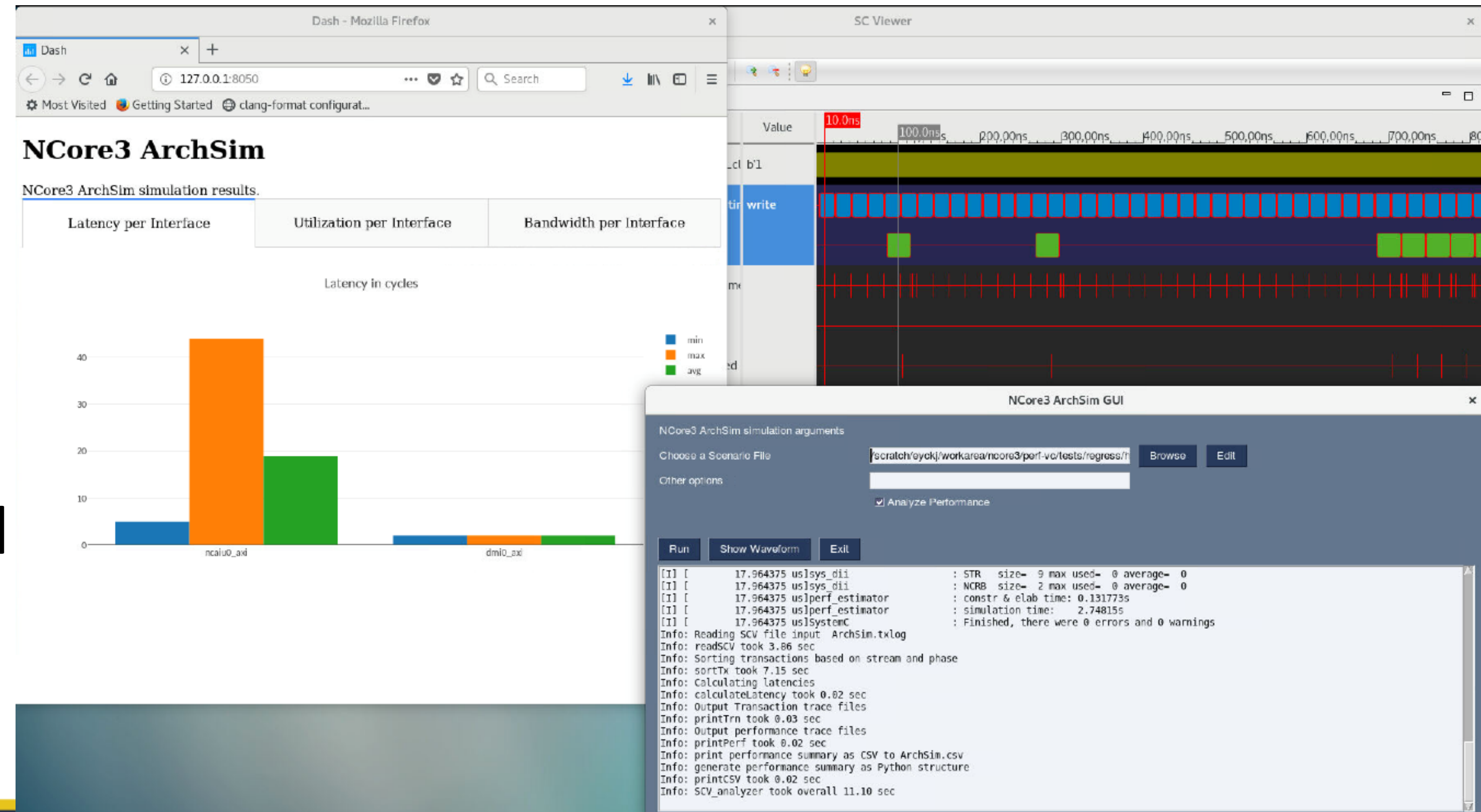
- Comprehensive tracing allows thorough analysis
- Choose the right formats
  - Waveform tracing using efficient implementation and FST format
  - Transaction tracing using Light weight transaction recording (LWTR)
- Python allows easy post-simulation analysis
  - Pyfst, cbor2 to read FST & LWTR recordings
  - Numpy, Pandas to analyze the trace events
  - Plotly/Dash to visualize

# 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, FTS for signals
  - Transaction tracing using LWTR
- 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



# Correlation

- Correlation against RTl models
- Waveform Analysis Language (WAL) to the rescue
  - Allows to abstract from signals to transactions

# Open-soure offerings I

- 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 II

- 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

# Questions?