S2C
EDA SOLUTIONS

Leveraging Transaction-Based Acceleration for High-Speed, High-Quality RISC-V Verification

Dehao, S2C

Agenda

01 RISC-V complexity and Verification Challenges

02 RISC-V Verification Interface (RVVI)

03 RVVI Limitations in Simulation-Only Environments

04 Transaction-Based Acceleration (TBA) as a Solution

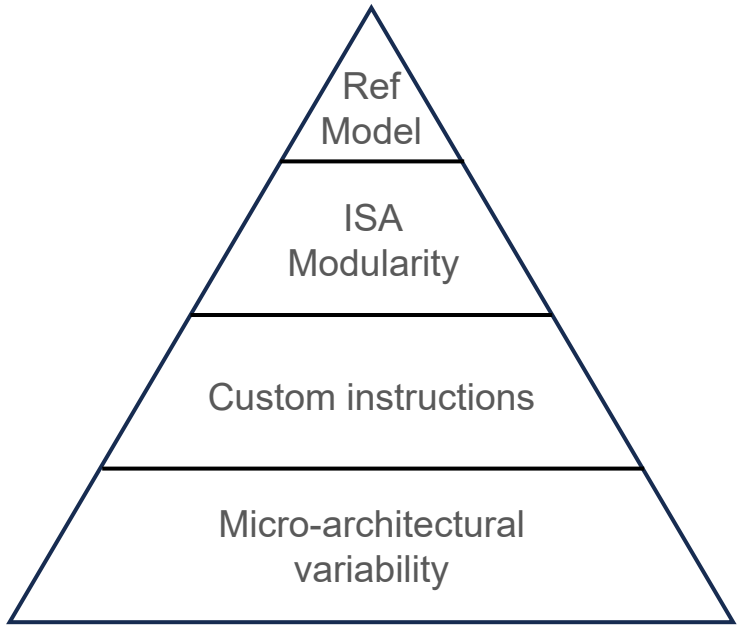
05 TBA applied to RVVI

06 Results and Observations

RISC-V complexity and Verification Challenges

- Key Complexity Factors of RISC-V
  - Microarchitectural variability
    - Complexifies the design
      - Memory Hierarchy Design
      - Pipeline and Performance Tuning
      - Power, Area, and Timing Constraints
- Custom instructions
  - Increase verification scope
  - Require thorough validation under all operating conditions
- ISA modularity
  - Complicates interoperability and toolchain validation
    - A program compiled for one RISC-V variant may not run on another that lacks required extensions
- No unified official reference model
  - Creates inconsistencies and uncertainty
  - Behavioral differences between implementations

$$challenge = complexity^{\left(\frac{1}{experience}\right)} * \log(1+t)$$

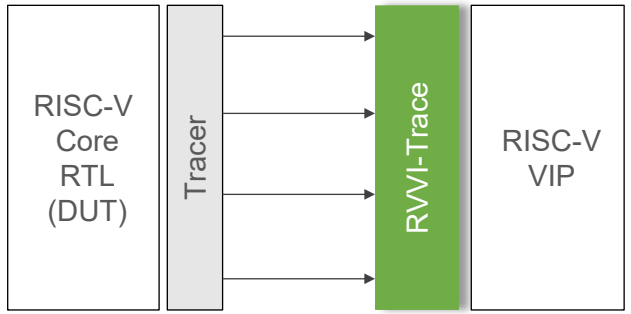


Verification challenges By Segment

How to address the complexity of RISC-V verification ?

- RISCV Verification Interface (RVVI)
  - Open standard interface developed to address the complexity of RISC-V verification
    - Provides standardized communication interface between the RTL design, testbench environment, and the RISC-V VIP
      - https://github.com/riscv-verification/RVVI
  - RVVI Key Components

RVVI-Trace



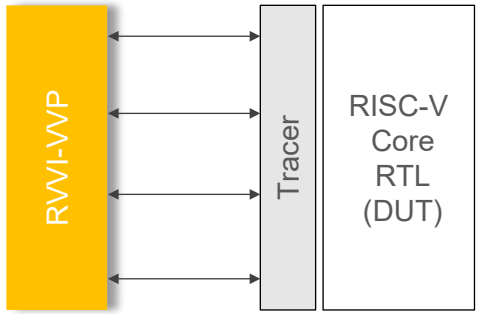
System Verilog interface (tracer) between RTL and testbench that records everything the RISC-V Core RTL (DUT) does while it runs

RVVI-API

Initialization APIs	rvviRefInit() … rvviRefCsrCompareMask ()
Main Loop APIs	rvviRefPcGet() … rvviRefGprGet ()
Shutdown APIs	rvviRefShutdown()

A set of C/C++ API functions, defined in a C header and System Verilog package, and provided with a DPI wrapper for use in both C/C++ and System Verilog TB

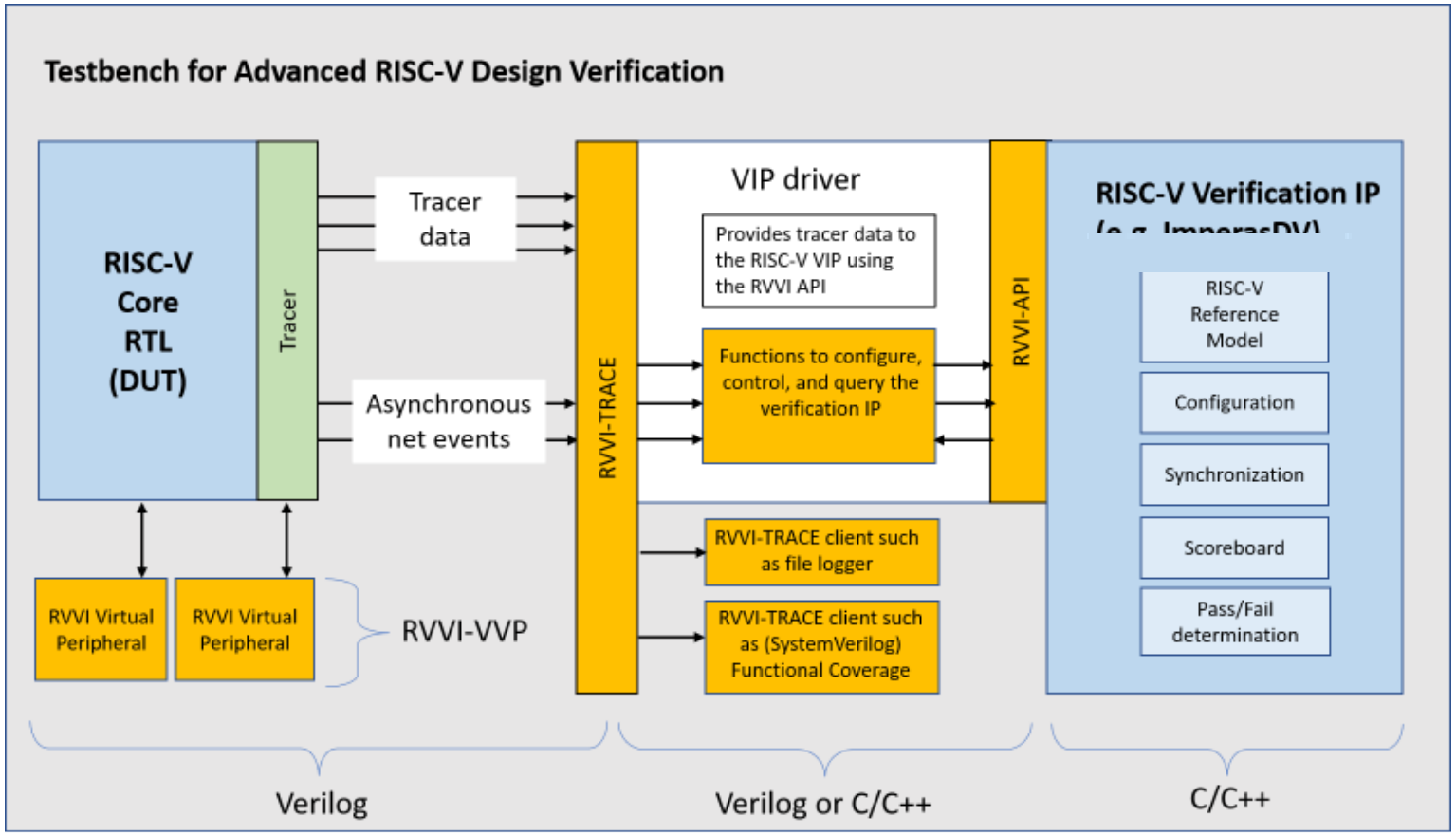
RVVI-VVP



A set of memory-mapped testbench components required for RISC-V compliance tests, defined to enable interaction with the processor and support usage in both C/C++ and SystemVerilog

How RVVI Helps ?

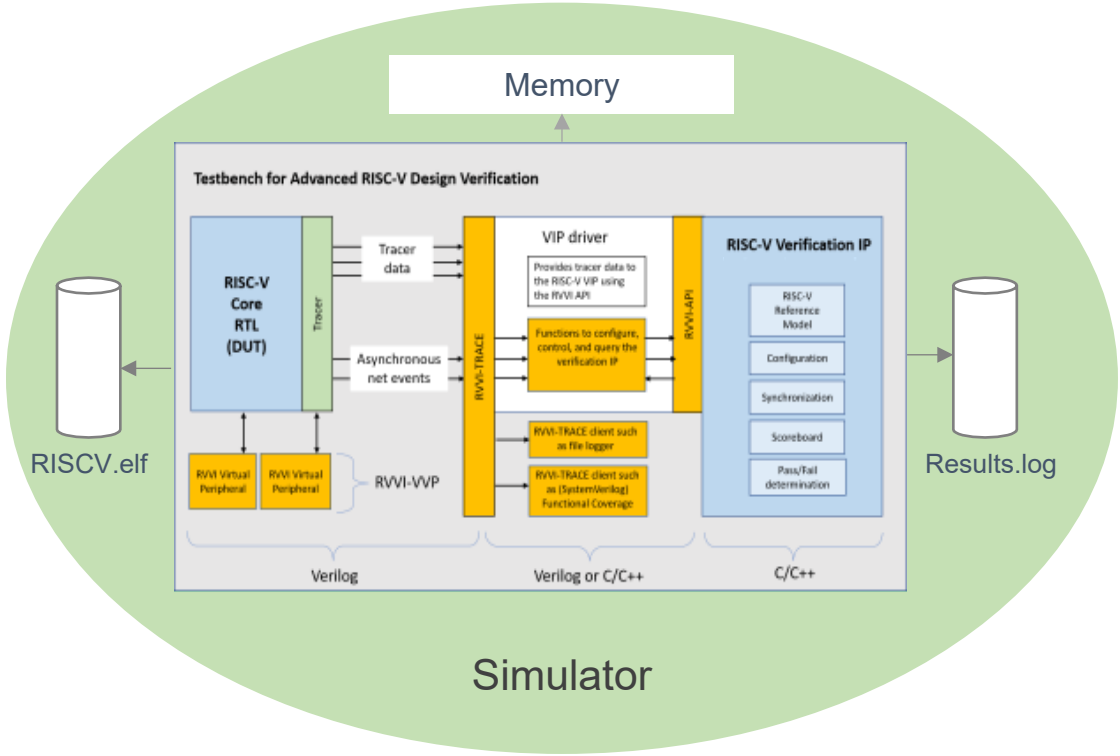
- Testbench structure with RVVI



- ✓ Instruction flow tracking with Tracer
- ✓ Out-of-Order & Speculative Execution
- ✓ Reference Model Comparison
- ✓ Reusable Testbenches
- ✓ ISA Compliance & Coverage
- ✓ Accelerates DV Timeline

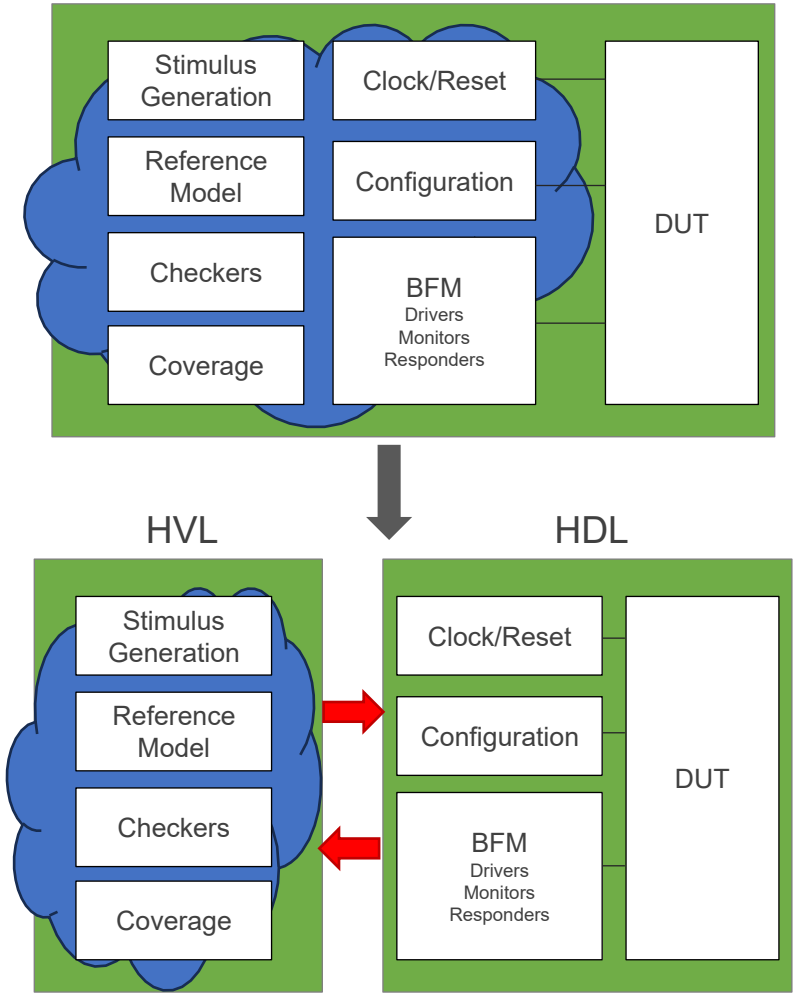
RVVI Limitations in Simulation-Only Environments

- Performance overhead due to trace generation, comparison, and monitoring
- Simulation performance limits test throughput
  - Simple test may take too long to run
- Timing and Synchronization issues between DUT and reference model
- No system-level modeling or acceleration for full SoC workloads
  - Lack of infrastructure to simulate full SoC
    - booting OS
    - Initializing devices
    - Peripheral interactions
- Maintaining compatibility while extending RVVI for custom features can introduce maintenance and validation burdens



Transaction-Based Acceleration (TBA) as a Solution

- Communication between SW and HW is transaction-based not cycle-based
  - TLM (Transaction-Level Modeling) interfaces and proprietary APIs
  - Operations between SW and HW are processed in a higher level of abstraction
    - Example: DMA transfer, Memory R/W etc...
  - Reduces the amount of data that needs to be processed between SW and HW
    - Testbench in SW sends/receives multi-clock cycle transactions to/from BFM in HW
- SW/HW partitioning
  - HDL side is synthesized into the HW
    - Design (DUT)
    - BFMs (driver, monitor, responder)
    - Clock and Reset
  - Untimed Testbench is in SW simulation
    - Test control
    - Stimulus generation
    - Reference model
    - Checkers and coverage
- Throughput
  - Ratio of time spent in HW relative to the overall runtime spent
  - Throughput not limited by latency in transaction-based acceleration
    - Time spent in HW is maximized (HW can run for longer time until a transaction arrive)

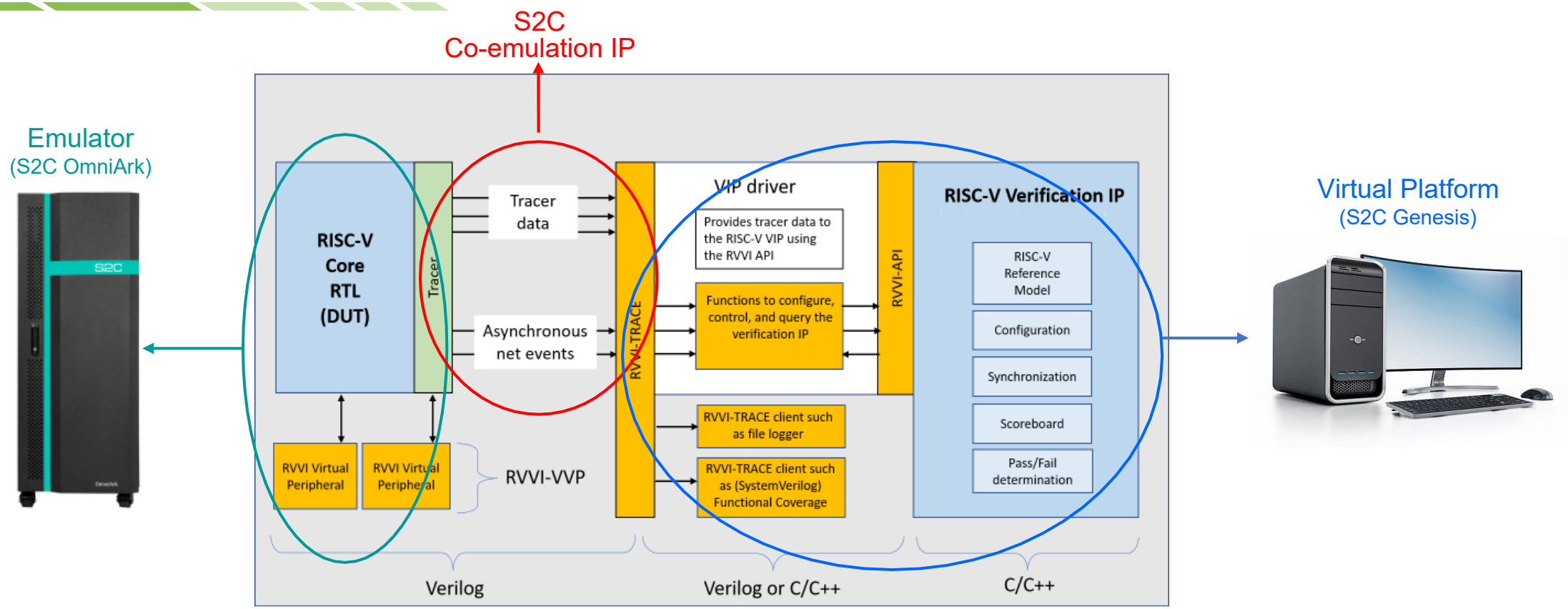


Benefits of Transaction-Based Acceleration

- Faster execution than traditional RTL simulation
  - 10-1000x speedup vs. RTL simulation
    - Multi-clock cycle behavior between HW and SW instead of cycle-by-cycle execution
  - Overall verification time is reduced thus accelerating the TTM
  - Faster regression testing
- Early software development & firmware validation
  - Concurrent hardware/software co-design
    - Software team can test drivers/firmware before silicon is ready

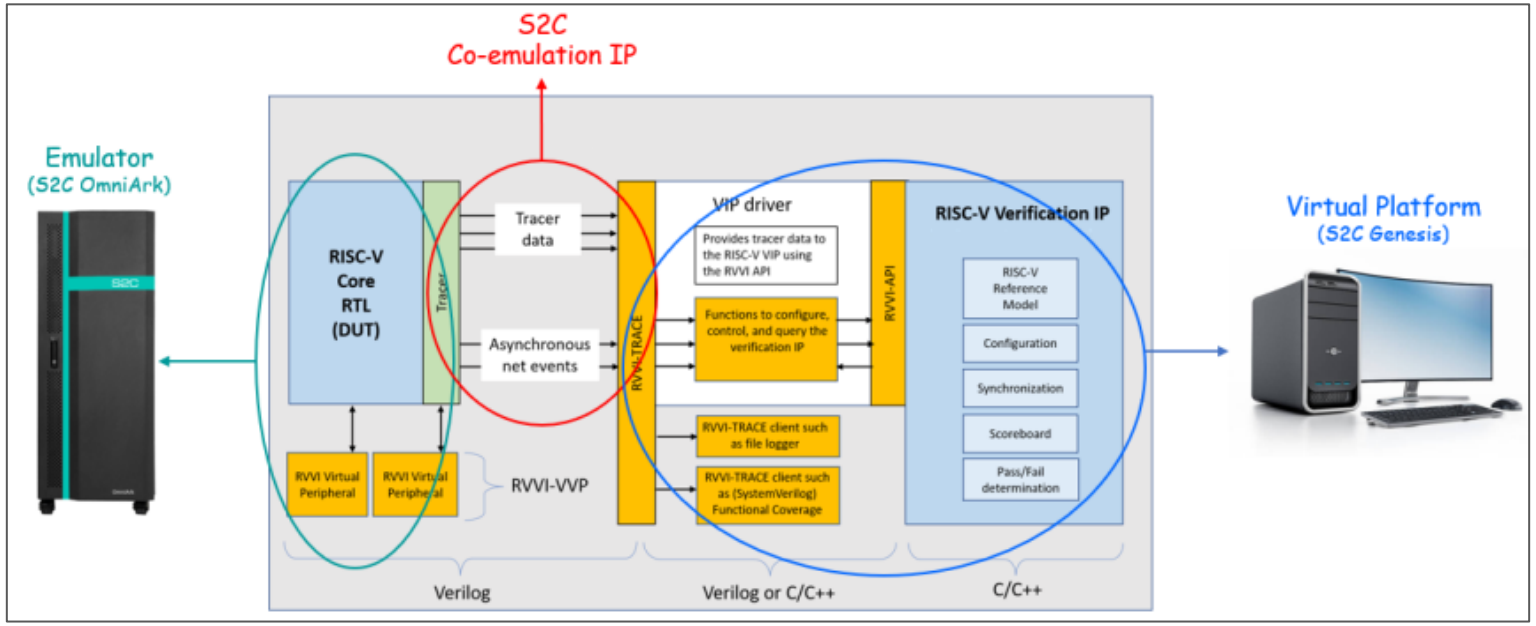
- Scalability for large designs
  - Efficient for SoCs/ASICs
    - Simulating complex systems that are too large or slow to simulate at the RTL
- Verification and Debug efficiency
  - Debug at the transaction level
    - No need to trace thousands of signals across millions of cycles
  - Automate error detection
    - Built-in protocol checkers check for protocol violations
    - Reference models in SW compare against actual RTL outputs
  - Coverage collector & analysis
    - Coverage metrics in SW ensure verification completeness

TBA applied to RVVI



- Testbench partitioning:
  - Virtual Platform (e.g., S2C Genesis) hosts the reference model the RVVI checker and coverage
  - RISC-V core runs on the emulator (e.g., S2C OmniArk)
- Co-emulation IP uses TBA to establish the communication channels
  - Synchronization between OmniArk emulator and Genesis Virtual Platform to maintain timing accuracy
  - Maintaining RVVI's cycle accuracy and consistency in split setups
  - Carrying RVVI trace info between ISS and OmniArk emulator

TBA+RVVI Flow



Design Partitioning → **Instantiate Co-emulation** → **Hardware setup** → **Connection** → **Compile** → **Run** → **Debug & Analysis**

Design Partitioning	Instantiate Co-emulation	Hardware setup	Connection	Compile	Run	Debug & Analysis
Separate synthesizable DUT from non-synthesizable components	Instantiate the co-emulation IP on both HW and SW	Configure the HW and setup the communication channels	Connect virtual platform to Co-emulation IP SW and connect DUT to the co-emulation IP HW	Compile DUT for HW and testbench for SW. Load the design onto the HW	Execute the testcases at faster speed with host-hardware interaction	Use logs, trace tools and waveform viewers to validate the behavior. Optimize and iterate to improve the performance

Results and Observations

- Use Case: video processing (320×180)

Metric	RVVI in Simulation-Only	TBA + RVVI in Co-Emulation
Frames processed per second	~0.2–0.5 FPS	~25–60 FPS (can meet)
Time to decode 1 min video	~1.5 to 4 hours	~30–60 seconds
Instructions per frame (est.)	~30M	Same
RVVI trace validation time (entire)	~2–3 hours	~2–3 minutes
Simulation runtime	~3–6 hours (total sim + RVVI)	~2–3 minutes (total, trace included)
Trace misalignment issues	High, due to peripheral latency	Low, trace aligned in hardware-assisted RVVI
Interrupt/CSR Sync Bugs Detected (avg)	Often missed unless specifically trapped	Captured through synchronized RVVI trace
Cycle-Level Alignment Accuracy	±10–500 cycles	±0–1 cycles

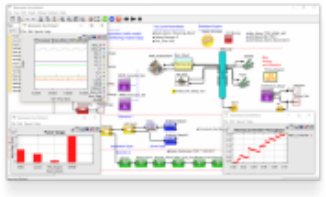
- RVVI + TBA enables both speed and quality
- RVVI trace integrity maintained in emulation
- Debug and coverage gains
- Reusability of environment for other RISC-V cores or configurations
- TBA enables RVVI to scale efficiently to emulator-based environments
- Maintains compliance while accelerating validation

S2C Global Organization

Who We Are

- A **worldwide leader** of functional verification solutions for today's innovative designs
- **20 years of success** in delivering rapid SoC prototyping solutions
- **600+ customers** worldwide
- **300+ employees** dedicated to functional verification solutions
- Direct offices in San Jose, Seoul, Tokyo, Shanghai, Shenzhen, Beijing, Xian

S2C Product Families



Genesis Architect

Early RTL Verification

Early Modeling

PegaSim Simulation

HW Debug

Claryti Debugger

OmniArk Emulation

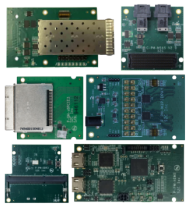
Prodigy Prototyping

EDA Cloud

RTL Regression

Heterogeneous Verification Platform

System Validation



90+ Prototype Ready IP
Speed Bridge
VIP

S2C
EDA SOLUTIONS

Thank you!