

# Automated Verification Flow of a Highly Configurable RISC-V Core Family

Stanislaw Kaushanski

MINRES Technologies GmbH



# MINRES Technologies

- Est. 2012 in Munich, Germany
- RISC-V International member since 2017
- We provide Software-Driven Hardware for the Intelligent Edge
- MINRES is an enabling company, dedicated to providing the expertise and solutions required for improving development productivity



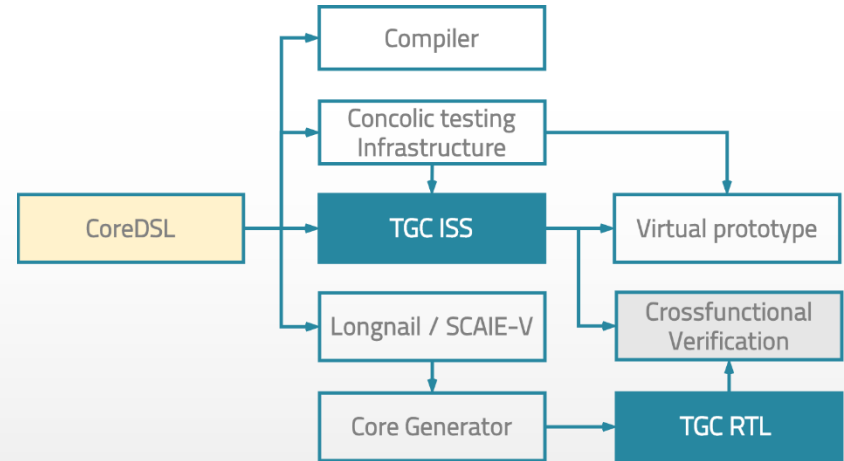
# TGC RISC-V Family

	Embedded systems	IoT or edge applications
Low area & power	TGC_A RV32E 3 stages	
RTOS capable	TGC_B RV32I 3 stages	TGC_C RV32IMC 4 stages
Tight hw/sw coupling		TGC_D RV32IMC 4 stages CLIC, PMP
High frequency		TGC_E RV32IMC 5 stages CLIC, PMP

- The Good Core (TGC)
- Highly flexible, scalable and extendable
- Standard configurations as starting points
- Easy combinations of features and options
  - Different bus interfaces
  - Interrupt controllers
  - Processor caches
  - Custom instructions
  - Safety features (lockstep, GPRs parity bits, ECC)
  - Security (physical memory protection)

# Single Source of Truth

- CoreDSL: Domain-specific language to model processor cores at the level of their *instruction set architecture* (ISA)
- Automatically generated:
  - Accurate ISS reference model
  - Configuration for random stimuli generation
  - List of instructions for coverage collection
  - Properties for formal verification
  - Artifacts for toolchain compatibility
  - Hardware for custom instructions



# CoreDSL

## Open Source Specification & Frontend

### Contents:

- Architectural state
  - Implementation parameter definition
  - General purpose register file
  - Single register with attribute
- Instructions
  - Instruction name
  - Specification of instruction encoding
  - Functional behavior

Simple definition of custom instructions using C-like syntax

```
Core My32bitRISCVCore {  
    architectural_state {  
        unsigned int    REG_LEN = 32;  
        unsigned int    XLEN = 32;  
        register unsigned<XLEN> X[REG_LEN];  
        register unsigned<XLEN> PC [[is_pc]];  
    }  
    instructions {  
        LUI {  
            encoding: imm[31:12] :: rd[4:0] :: 7'b0110111;  
            behavior: if (rd != 0) X[rd] = imm;  
        }  
    }  
}
```

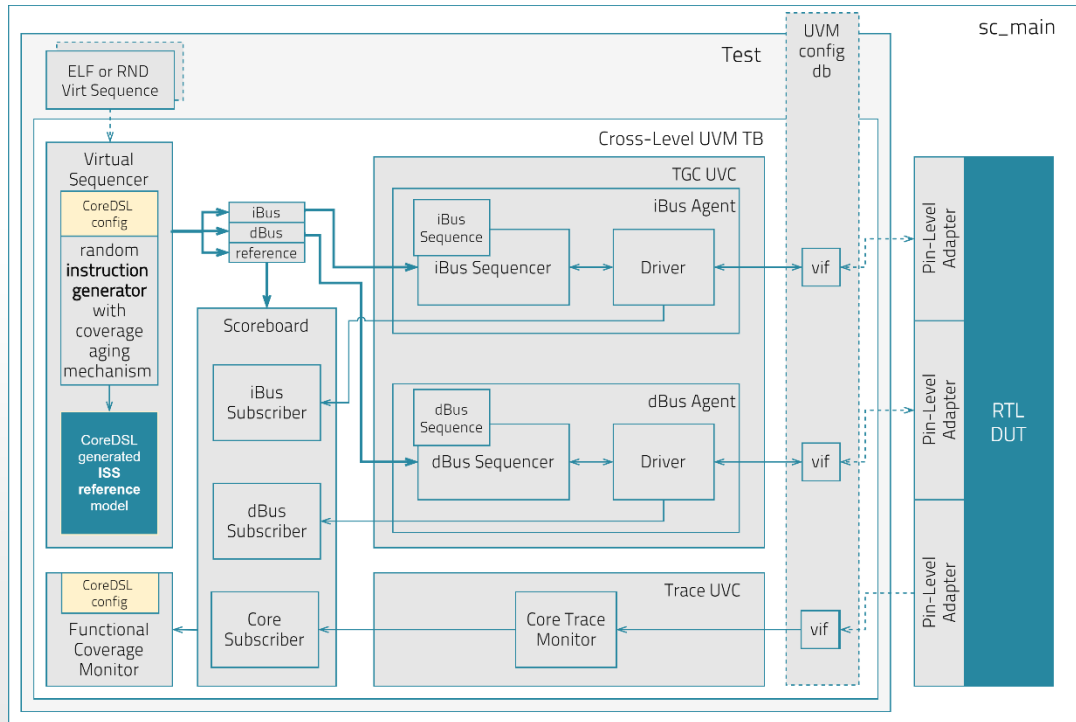
# Cross-Level TB Overview

- UVM-SystemC Testbench

- Seamless integration of generated components
- Instruction Generator sends random instructions to ISS and RTL
- ISS behavior and state compared with RTL results in the Scoreboard

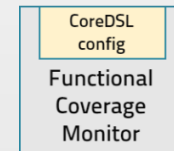
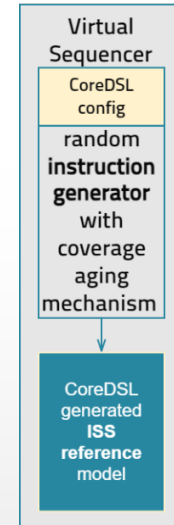
- TB operation modes:

- Pseudo-random instruction generation with aging based feedback
- Load and execute ELF file



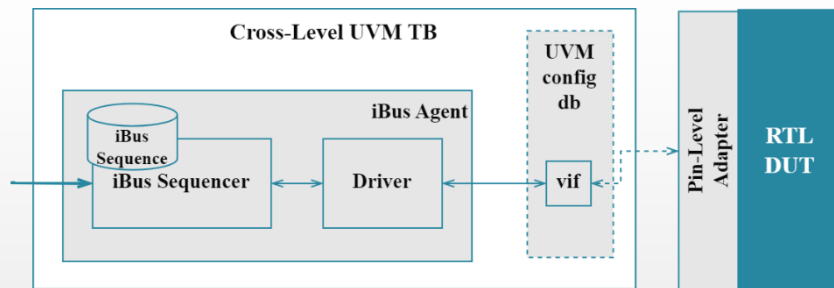
# CoreDSL: Generated Components

- Virtual sequencer:
  - Instruction generator + ISS reference model
  - Instruction generator dynamically adjusts instruction frequency for optimized coverage
  - Coverage aging mechanism speeds up coverage achievement
- Instruction accurate ISS model
  - DBT-RISE infrastructure is the basis for the reference model
- Functional coverage monitor:
  - Utilizes information from CoreDSL description to accurately track and report coverage metrics



# UVM-SystemC TB Agents

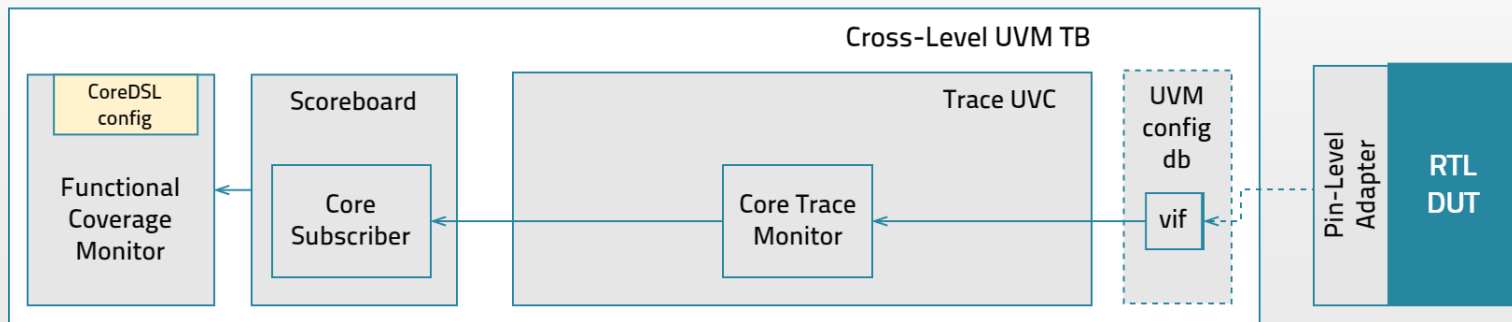
- iBus and dBus agents: sequencer + driver
  - Connected to the DUT through a virtual interface (vif)
  - The DUT initiates instruction fetches as well as data bus accesses over vif
- vif enables communication without being tied to a specific implementation
  - DUT can be exchanged without changing the interface itself
  - Simulation engine can be exchanged
    - Verilator via SystemC wrapper and Pin-Level adapter
    - SystemVerilog Simulator via UVM Connect
    - Hybrid simulation with FPGA using RAVEN





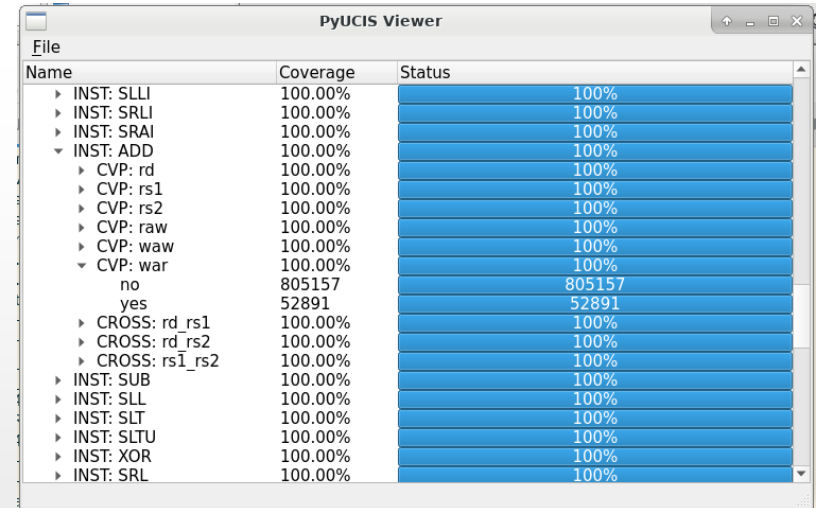
# UVM-SystemC TB Agents

- Trace interface maps the internal state of the core
  - Register values
  - Program counter
  - Traps
- The scoreboard analyzes and compares the iBus, dBus, and trace monitor sequences against the reference.



# Functional Coverage

- Coverage Monitor:
  - Defines coverage for different instruction types
  - Coverpoints: coverage criteria for instructions
    - Parameters
    - Dependencies
    - Hazards
  - Covergroups: Summarize coverage information
- Functional coverage provides:
  - Parameter toggling frequency
  - Cross-coverage analysis
  - Identification of data hazards



The screenshot shows the PyUCIS Viewer application window. It displays a table with three columns: Name, Coverage, and Status. The table lists various instruction types and their coverage percentages, along with their status values. The status values are either 100% or specific counts.

Name	Coverage	Status
▶ INST: SLLI	100.00%	100%
▶ INST: SRLI	100.00%	100%
▶ INST: SRAI	100.00%	100%
▼ INST: ADD	100.00%	100%
▶ CVP: rd	100.00%	100%
▶ CVP: rs1	100.00%	100%
▶ CVP: rs2	100.00%	100%
▶ CVP: raw	100.00%	100%
▶ CVP: waw	100.00%	100%
▼ CVP: war	100.00%	100%
no	805157	805157
yes	52891	52891
▶ CROSS: rd_rs1	100.00%	100%
▶ CROSS: rd_rs2	100.00%	100%
▶ CROSS: rs1_rs2	100.00%	100%
▶ INST: SUB	100.00%	100%
▶ INST: SLL	100.00%	100%
▶ INST: SLT	100.00%	100%
▶ INST: SLTU	100.00%	100%
▶ INST: XOR	100.00%	100%
▶ INST: SRL	100.00%	100%

# Conclusion

- Highly effective solution for detecting design issues including control and data hazards
- Automated testing of various cores without manual adjustments
- Automated extension with custom instructions
- Does not require forward progress guarantee – better testing of hazards
- Modular design allows for easy adaptation and reuse
  - Same testbench is being reused for 64bit cores
- Ensures accurate behavior of the ISS
- Not limited to the TGC core family

# Links & References

- TGC Family: <https://www.minres.com/products/the-good-folk-series>
- CoreDSL: [www.git.com/minres/CoreDSL](http://www.git.com/minres/CoreDSL)
- TGC-ISS: <https://git.minres.com/TGFS/TGC-ISS>
- SCC: <https://github.com/Minres/SystemC-Components>
- RAVEN: <https://www.minres.com/products/hybrid-simulation>

