

UNITED STATES

SAN JOSE, CA, USA FEBRUARY 28 - MARCH 3, 2022

Tutorial: Introduction to the 5 levels of RISC-V processor verification

Simon Davidmann and Lee Moore, Imperas Software





Focus of this tutorial



- RISC-V is changing the options that SoC designers have in their tool kits
- RISC-V means many teams are designing new processors, or modifying source of processors
- For RISC-V anybody can be 'an architecture licensee'
- And every CPU needs verifying... in detail... (its not like buying in pre-verified IP)
- Many people are new to CPU DV for the first time
 - Traditionally done behind closed doors in commercial/proprietary companies
- This presentation aims to introduce the main approaches of RISC-V CPU DV
- And discusses pros and cons of the different approaches
- Also it introduces the main components needed in any RISC-V processor DV environment



RISC-V processor verification tutorial Items to be covered, main takeaways

mperas

- Introduction to RISC-V
- Overview of the issues when verifying the design of a RISC-V CPU
- Different approaches to verifying a RISC-V CPU
- Main components of a verification testbench
- Introduction to simulators for RISC-V CPUs
- Use of various virtual platform components in verification
- Discussion of different reference model requirements
- Introduction to instruction stream generators
- Status of RISC-V compliance and its relationship to verification
- Pointers to some useful architectural validation test suites
- Understanding a complete SystemVerilog testbench via a detailed walk through demonstration



Introduction to Imperas Involvement with RISC-V



- Imperas develops simulators, tools, debuggers, modeling technology, and models to help embedded systems developers
 get their software running...
- ...and hardware developers get their designs correct
- 14+ years, self funded, profitable, UK based, team with much EDA (simulators, verification), processors, and embedded experience
- Staff worked in Arm, MIPS, Tensilica, Cadence, Synopsys
- and in verification in EDA on development of Verilog, VCS, SystemVerilog, Verisity and their methodologies
- Started work with customers on RISC-V in 2017
- Contributed to RISC-V compliance since 2018, RISC-V DV since 2019
- Our RISC-V focus is CPU verification

Page 4

- We provide configurable reference models, the fastest highest quality simulators, advanced development tools and the absolute best solution for RISC-V hardware design verification
- 20+ of the leading RISC-V CPU developers use and rely on Imperas solutions



www.imperas.com www.OVPworld.org

© Imperas Software Ltd.





- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment

For each topic and item we will try to introduce, explain, even demo technologies and products that are available – to give you a feeling of current state-of-the-art

We will also introduce and walk through some open source solutions







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment









- RISC-V (pronounced "risk-five") is an open standard instruction set architecture (ISA) that began in 2010 and is based on established reduced instruction set computer (RISC) principles
- Unlike most other ISA designs, RISC-V is provided under open source licenses that do not require fees to use
- The project began in 2010 at the University of California, Berkeley, but now many current contributors are volunteers not affiliated with the university
- Unlike other academic designs which are typically optimized only for simplicity of exposition, the designers intended that the RISC-V instruction set be usable for practical computers



RISC-V == Freedom...



Freedoms enabled by RISC-V are a huge opportunity



RISC-V == Freedom...



Freedoms enabled by RISC-V are a huge opportunity

Freedoms enabled by RISC-V are a huge challenge for verification

the largest change in the industry since? ...



Challenges in RISC-V CPU DV



- Feature selection and choices require serious consideration due to implications of every choice
 - Experienced architecture teams know the costs associated with every feature
 - Every addition dramatically increases (doubles ?) verification & compounds verification complexity
 - Costs of simple added feature can be huge and unknown to inexperienced teams
 - Adds schedule, resources, quality costs == big risks...
- As of 2021, No off-the-shelf toolkit/products available for DV of processors
 - No EDA vendor has 'RISC-V CPU DV kit' product
 - There are in-house proprietary solutions in CPU developers... Intel, AMD, Arm, ...
 - Building your own adds schedule, resources, quality costs and risks
- Current SoC cost is 50% for HW DV (with CPUs bought in as proven IP)
 - Developing own CPU adds huge DV incremental schedule, resources, quality challenges







- Brief Introduction to RISC-V
- Processor DV project timeline
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment



A CPU HW DV project timeline



- Source/build/hire/allocate the expert team to do the work...
- Focus on what needs to be verified develop measurement metrics
- Develop Verification Plan (and measurement metrics)
- Determine EDA tools and models and methods to be used
- Simulation choices: open source, commercial, bespoke, SystemVerilog, UVM, FPGA, Emulation, ...
- Formal...
- Get tools, verification IP (VIP), testbenches, models in place
- Obtain tests, create tests
- Generate huge number of (constrained random) tests
- Verify, triage and fix issues, continue...
- ... continue while measuring until functional and code coverage metrics reached
- Benchmark, soak and integration testing







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment

During this section, several components will be mentioned... like ISS, and ISG – these may be introduced as we go, or may be discussed in more detail in later sections of this tutorial



mperas

Agenda

- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment





Agenda

- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment

Note that not all projects have the same requirements, schedule or verification needs – so each project's DV needs may / will differ







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment







- "if I can get a program to run then my DV is done... right?"
- "my DV challenge is sorted if I can get Linux to boot on my design..."
- Basically this level of DV is where developer feels if they can get their current compilation of their current program to run (through one path) - then their silicon design job is done
- This may be fine for test chips, research, academic, hobbyists, but NOT for products
- This approach is often due to lack of knowledge or interest in quality, ...



'Hello World' DV



- This requires either an HDL simulator, or an FPGA, or some silicon, and a test harness of some form to allow it to run programs...
- This is not DV!







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment



#1: Simple (results) check (1a)





Run RTL DUT in testbench

- (no real testbench)
- Just loads & runs the test program
- Each test program checks its results = go/no go test
 - Prints message to log
 - or writes bit to memory



#1: Simple (results) check (1b) (use e.g. riscvOVPsim ISS from GitHub)





FEBRUARY 28 - MARCH 3, 202

- Run RTL DUT in testbench
 - (no real testbench)
 - Just loads & runs the test program

Either

- Each test program checks its results = go/no go test
 - Prints message to log
 - or writes bit to memory
- Or, then run ISS, write log or signature file
 - Compare/diff file results (afterwards)
 - This is the approach taken by RISCV International for their architectural validation ("compliance tests")

01 2022

#1: Simple check (use e.g. riscvOVPsim ISS from GitHub)



Summary

- Very simple, needs basic ISS, and tool chains
 - Free ISS: <u>https://github.com/riscv-ovpsim</u>
 - Free compiler: https://github.com/Imperas/riscv-toolchains
- Basic bring up
- Good for simple test runs
- Basic functionality testing
- Still need accurate, configurable, version selectable, complete, reference model
- Not a robust DV solution







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment



#2: Entry Level DV: post-sim trace-compare (use e.g. riscvOVPsimPlus ISS from OVPworld)







Imperas riscvOVPsimPlus Reference Simulator

Page 31

Process

use random generator (ISG) to create tests

Imperas

- during simulation of ISS write trace log file
- during simulation of RTL write trace log file
- at the end of both runs, run logs through compare program to see differences / failures
- ISS: riscvOVPsimPlus includes Trace and **GDB** interface
 - Free ISS: https://www.ovpworld.org/riscvOVPsimPlus
- ISG: riscv-dv from Google Cloud / Chips Alliance
 - Free ISG: https://github.com/google/riscv-dv







Page 32

Process

mperas

- use random generator (ISG) to create tests
- during simulation of ISS write trace log file
- during simulation of RTL write trace log file
- at the end of both runs, run logs through compare program to see differences / failures
- ISS: riscvOVPsimPlus Includes Trace and **GDB** interface
 - Free ISS: https://www.ovpworld.org/riscvOVPsimPlus
- ISG: riscv-dv from Google Cloud / Chips Alliance
 - Free ISG: https://github.com/google/riscv-dv

#2: Entry Level DV: post-sim trace-compare (use e.g. riscvOVPsimPlus ISS from OVPworld)

Summary

- Compares files created after test runs
- Can be signature, logging, or instruction trace
- Can use random ISG as no need to know expected results...
- Usually the easiest method to implement (dependent on tracing formats)
 - Capture of program flow (monitor the PC)
 - Capture of program data (monitor the Registers)
- Potentially very large data files
- Potential for wasteful execution (if early failure)
- Will not work for on async events, control flow, or hardware real time effects, MP, OoO, multi-issue, ...
- Not a robust DV solution for commercial cores
- Can engage with Imperas for licenses of reference models and optional development to add customer own instructions, CSR, behaviors









- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment



#3: Industrial Quality Sync DV (sync-lock-step-compare)

- Tandem lockstep run both reference and DUT run together in lock step
- Not very complex to obtain, set up
- Compare PC, CSRs, GPRs, other internal state instruction by instruction
- No requirement on data saving
- No requirement on known good results in test
- Will not work for async events and control flow , ... – it is all about the data flow
- [OpenHW evolved into using async see later slides]

Coverage Driven Verification of OpenHW CORE-V Processors with Imperas RISC-V Golden Reference Model

Example flow:

Imperas



- Control block steps both CPUs, extracting data and comparing results
 - 1st Generation OpenHW flow (1H2020)



#3: Industrial Quality Sync DV (sync-lock-step-compare)

mperas

Summary

- Instruction by instruction lockstep comparison (excludes async events)
 - Comparison of execution flow
 - Comparison of program data
 - Comparison of programmers and internal state
- Immediate comparison
 - Allows for debug introspection at point of failure very powerful
 - Does not waste execution cycles after failure
- Will not work for async events, control flow, or hardware real time effects, ...
- Not too hard to develop & set up (depends on RTL DUT tracer features)
- Lock-Step / Compare is by far the best and most efficient approach
 - But does not address async events (see level #4)
- Need to engage with vendors such as Imperas for licenses of reference models and optional development to add customer own instructions, CSR, behaviors



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment



#4: Industrial Highest Quality Async DV (async-lock-step-compare)

- Builds on & extends Industrial Quality sync-lock-step-compare DV
- Adds focus on async capabilities
- Depending on design this can include: OoO, MP, debug mode, interrupts, multi-issue, ...
 - Example SystemVerilog Components
 - tracer: Reports instructions for checking and register writebacks
 - step_and_compare: Manages the reference model and checks functionality
 - interrupt_assert: Properties for interrupt coverage/checking
 - debug_assert: Properties for debug coverage/checking
- Typically hard, complex, and expensive to get working
 - Challenge is extracting async info from microarchitecture RTL pipeline
 - See latest developments with RVVI and ImperasDV



2nd generation CV32E40P OpenHW flow (2H2020) (Imperas model encapsulated in SystemVerilog)



#4: Industrial Highest Quality Async DV (async-lock-step-compare)



Summary

- Instruction by instruction lockstep comparison (includes async events)
 - Comparison of execution flow, of program data, of programmers and internal state
- Immediate comparison
 - Allows for debug introspection at point of failure very powerful
 - Does not waste execution cycles after failure
- Includes focus on async events, control flow, and hardware real time effects
- Can be hard to develop & set up (depends on RTL DUT tracer features and pipeline understanding)
 - See latest development for RVVI and ImperasDV
- Can be expensive in terms of time, resources, licenses and costs a lot per bug found
 - But the bugs are even more expensive if not found early enough...
- Lockstep / Compare is by far the best and most efficient approach (industry 'gold standard')



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
 - Digression into why we need standards...
- Components of RISC-V CPU DV environment


Challenges moving forward – the need for standards

- There are many different components needed:
 - DUT & its encapsulation
 - 'tracer' information
 - Control
 - Reference model & its encapsulation
 - Configuration
 - Comparisons
 - Synchronization
 - Asynchronous operations
 - Control
 - Functional coverage measurement & assertions
 - Test bench
 - Configuration
 - Overall control
 - Scoreboarding
 - Reporting / Logging
 - Tests (directed or generated)
 - Program linker scripts and binary file reader
- And each component has different interfaces and requirements on the interfaces



Imperas

2nd generation CV32E40P OpenHW flow (2H2020)



Challenges moving forward – the need for standards

- There are many different components needed:
 - DUT & its encapsulation
 - 'tracer' information
 - Control
 - Reference model & its encapsulation
 - Configuration
 - Comparisons
 - Synchronization
 - Asynchronous operations
 - Control
 - Functional coverage measurement & assertions
 - Test bench
 - Configuration
 - Overall control
 - Scoreboarding
 - Reporting / Logging
 - Tests (directed or generated)
 - Program linker scripts and binary file reader
- And each component has different interfaces and requirements on the interfaces





2nd generation CV32E40P OpenHW flow (2H2020)

It would be a disaster for RISC-V if every design team had to re-invent everything...

© Imperas Software Ltd



Page 43

Why need standard interfaces?



- There are many blocks required in DV solutions
 - They all have different interfaces and these interfaces need defining
 - They may come from different developers / suppliers
 - They may be used in different projects and processor configurations / generations
- If standards exist, then verification IP can be created and licensed
- Goals when developing standard interfaces:
 - block re-use
 - common ways to do things
 - quick start-up
 - efficiency



Why adopt a standard?



- You have to use some interfaces
- No need to re-invent on your own they do not need to be proprietary
- RVVI is an open standard available on GitHub
- RVVI (and its predecessor) have already been flushed out and are in use
- There is no downside to adoption
- Upside to adoption: potentially make use of other tools / code
- What tools / technologies can potentially be (re)used
 - Encapsulation of reference models
 - Test benches & test bench components (including onward connection to reference models)
 - Functional coverage & assertions
 - Log file writer
 - Signature file writer (for RISC-V compliance testing)



RVVI: RISC-V Verification Interface History



- This RVVI work has evolved from over 2 years experience
 - Imperas, EM Micro, and SiLabs, ..., working with several RISC-V verification projects
 - Collaboration with OpenHW Group (<u>https://github.com/openhwgroup/core-v-verif</u>)
 - Re-usable test bench for Core-V range of open-source RISC-V cores
- Also... there was previously the RISC-V Formal Interface (RVFI) targeting formal tools
 - https://github.com/SymbioticEDA/riscv-formal
 - Interface for providing observation into a running core by streaming what is executing on the core (i.e. the basic trace data / functionality)
 - For quality RISC-V processor DV more is needed (than RVFI)
 - And each user needs to extend it with proprietary extensions (which is not the right approach...)
 - The RVVI-VLG interface has some parts very similar to the RVFI
 - RVVI-VLG can be thought of as an updated superset of RVFI



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
- Components of RISC-V CPU DV environment



#5 Evolving to use developing standards (RVVI)





- Focus is on developing standard interfaces between components
 - Allows reuse
 - Allows development of independent VIP
- Two main components to consider
 - DUT
 - Reference model



•

RVVI: RISC-V Verification Interface (driven by RISC-V DV usage)





<u>https://github.com/riscv-verification/RVVI</u> (Public Open Standard)

- RVVI-VLG
 - Verilog DUT interfaces
 - RVVI-VLG state streaming 'tracer' data
 - RVVI-VLG nets implementation dependent (Interrupts, Debug)
 - Handles multi-hart, multi-issue, Out-of-Order

	DV
RVVI-API	Reference Model

Test bench virtual peripherals RVVI-VPI

Page 49



- RVVI-API
 - Controls DV subsystem and reference model
 - C/C++
 - SystemVerilog
- RVVI-VPI (work-in-progress (Feb. 2022))
 - Virtual Peripheral Interfaces
 - timers, interrupts, debug, random, printer/uart, ...
 - Verilog and C macros & examples

SystemVerilog test bench using RVVI and components





VIRTUAL FEBRUARY 28 - MARCH 3, 2022

5 main CPU DV components







RTL DUT with 'tracer' interface



- The key component the DUT being tested
 - Includes memory model and bus interfaces
- Requires a 'tracer' to provide appropriate data to the test bench
- Requires control interface so test bench can step through events
- Quality of the 'tracer' determines the potential capabilities of the DV
- Can be RVVI, bespoke, or bespoke + extensions + control



Tests: Directed & Instruction Stream Generator



Directed Tests

- Generates test programs
- Usually using constrained random approach
- Most often obtain one:
 - Commercial such as Valtrix STING
 - Open source e.g. Google riscv-dv (written in SystemVerilog)
- Acquire suites of tests
 - Imperas make some available
- May require toolchains like GCC, LLVM for assemblers, linkers



Imperas Test Suites



- When verifying a CPU design you can never have enough tests...
- Imperas have developed a directed RISC-V test generator, instruction coverage measuring VIP, and a test qualifying mutating fault simulator to provide high quality test suites
- The generated tests suites are targeting architectural compatibility as defined in the RVIA architectural test working group coverage requirements
- There are currently over 50 free test suites, including
 - I,M,C,F,D,B,K,V,P
 - The provided vector test suite is one specific vector engine configuration
- The test suites are provided under an OVP open source license and are available free from: <u>https://github.com/riscv-ovpsim/imperas-riscv-tests</u>



Functional Coverage & assertions

Functional coverage measurement

- Normally written in SystemVerilog using covergroups, coverpoints, and assertions
- Targets specific measurements as required in verification plan
- Connects to RVVI-VLG from 'tracer'
- Typically includes
 - Standard ISA instruction extension measurement
 - Sequential instruction monitoring for e.g. hazards, etc.
 - Privilege model items such as interrupts, exceptions, debug mode
 - User specifics related to pipeline and micro-architecture
- Requires SystemVerilog simulator





Test bench / harness control, sequencing, compare (SystemVerilog, C or C++)

Test Bench / Harness



Test bench / harness control, sequencing, compare SystemVerilog

RVVI-VPI Test bench virtual peripherals

- Instances and connects all the subsystems
- Controls the stepping of events and instructions
- Connects the data & signals between the DUT and reference subsystem synchronizers and comparators
- Can have virtual peripherals such as uart for logging, or timers for asynchronous event / interrupt generation
- Can be in SystemVerilog for DUT in RTL
- Can be in C/C++ for DUT in C based compiled simulator
- Imperas provides templates in C/C++ and SystemVerilog



Page 56

© Imperas Software Ltd.

Reference model DV Subsystem



	DV
RVVI	RISCV reference model

Page 57



Three main components

- **RISC-V** reference model
- DV infrastructure to control model, etc.
- RVVI interface into test bench / harness
- Responsible for
 - Configuration of model
 - Comparisons between DUT and reference state
 - Synchronization due to pipeline affects
 - Asynchronous operations
 - Control of model



Reference model DV Subsystem





Page 58



• Three main components

RISC-V reference model

- DV infrastructure to control model, etc.
- RVVI interface into test bench / harness
- Responsible for
 - Configuration of model
 - Comparisons between DUT and reference state
 - Synchronization due to pipeline affects
 - Asynchronous operations
 - Control of model

Key component is Reference Model

- RISC-V is highly configurable & extendable
 - 200... Questions ?
- So it can get a little complicated





Example reference model: Imperas





http://www.imperas.com/riscv

- Imperas provides full RISC-V Specification envelope model
- Industrial quality model /simulator of RISC-V processors for use in compliance, verification and test development
- Complete, fully functional, configurable model / simulator
 - All 32bit and 64bit features of ratified User and Privilege RISC-V specs
 - Unprivileged versions 2.2, 20191213
 - Privilege versions 1.10, 1.11, 1.12
 - Vector extension, versions 0.7.1, 0.8, 0.9, 1.0
 - Bit Manipulation extension, versions 0.91, 0.92. 0.93, 1.0.0
 - Hypervisor version 0.6.1, 1.0.0
 - K-Crypto Scalar version 0.7.1, 1.0.0
 - Debug versions 0.13.2, 0.14, 1.0.0
 - P DSP/SIMD versions 0.5.2, 0.9.6
 - Zicbom, Zicbop, Zicboz, Zmmul, Zfh, Zfinx, Zce
 - Svnapot, Svpbmt, Svinval, Smstaten, Smepmp, ...
- Model source included under Apache 2.0 open source license
- Used as reference by :
 - Mellanox/Nvidia, Seagate, NSITEXE/Denso, Google Cloud, Chips Alliance, IowRISC, OpenHW Group, Andes, Valtrix, SiFive, Codasip, MIPS, Nagra/Kudelski, Silicon Labs, RISC-V Compliance Working Group, ...



Imperas is used as RISC-V Golden Reference Model

Page 60

© Imperas Software Ltd.

Imperas Model extensibility



- Separate source files and no duplication to ensure easy maintenance
- Imperas or user can develop the extension
- User extension source can be proprietary

Page 61

Imperas develops and maintains base model

- Base model implements RISC-V specification in full
- Fully configurable to select which ISA extensions
- Fully configurable to select which version of each ISA extension
 - Updated very regularly as ISA extension specification versions change
- Fully configurable for all RISC-V specification options
 - e.g. implemented optional CSRs, read only or read/write bits etc...

Imperas provides methodology to easily extend base model

- Templates to add new instructions
- Code fragments for adding functionality
- 100+ page user guide/reference manual with many examples
 - Includes example extended processor model



Imperas model is architected for easy extension & maintenance

© Imperas Software Ltd.



User feedback of Imperas as a reference



- "Andes is pleased to <u>certify the Imperas model and simulator</u> as a reference for the new Vector processor NX27V, and is already actively used by our mutual customers."
 - Charlie Hong-Men Su, CTO / EVP at Andes Technology Corp
- "We have <u>selected Imperas simulation tools and RISC-V models</u> for our design verification flow because of the <u>quality of the models and</u> <u>the ease of use</u> of the Imperas environment. Imperas reference model of the complete RISC-V specification, the ability to add our custom instructions to the model and their experience with processor RTL DV flows were also important to our decision."
 - Shlomit Weiss, Senior Vice President of Silicon Engineering at Mellanox / Nvidia
- "The OpenHW Group charter is to deliver high quality processor IP cores for our leading commercial members and open source community adoption. Central to this goal, the OpenHW Verification Task Group developed and published a DV test plan and implemented an open engineering-in-progress approach as we complete the verification tasks using the <u>Imperas golden RISC-V reference model</u>."
 - Rick O'Connor, Founder and CEO at OpenHW Group
- "Imperas are the pioneers in simulation technology and processor verification for RISC-V. Codasip is very proud of our rigorous approach to verification – <u>using Imperas as an important part of our quality process</u> furthers extend our differentiation. The Imperas independence, reputation and technical strength provides our customers with further reassurance in our 'best in class' RISC-V processors."
 - Philippe Luc, Verification Director Codasip
- "With this Imperas collaboration, our mutual customers will benefit from the availability of SiFive qualified models that are compatible with the mainstream EDA tool flows."
 - Phil Dworsky, Director, Strategic Alliances, SiFive







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
 - #6 using standards based DV products and VIP
 - Drill down into an available commercial RISC-V HW DV verification solution
 - Demonstration of ImperasDV in action on open source RISC-V cores
- Components of RISC-V CPU DV environment



A dedicated RISC-V CPU DV solution: ImperasDV from Imperas







Page 64

CPU DV test bench components mperas





CPU DV test bench components mperas



- Focus on:
 - DUT + 'tracer'
 - Test bench
 - DV subsystem



Page 66







- Encapsulates the reference model
 - Select model, use variant, configure,
- Includes DUT reference state storage
- Includes synchronization technology
 - Can run sync, async, interrupts, debug, multi-hart
- Includes comparison technology
- Includes Imperas instruction coverage
- Is configurable and traceable
- Can be used in C/C++ or SystemVerilog test bench / harness
 - Uses RVVI-API
- Very simple to use the 'smarts' are built-in



Test bench / harness control, sequencing, compare (SystemVerilog, C or C++)

Test Bench / Harness





- Instances and connects all the subsystems
- Controls the stepping of events and instructions
- Connects the data & signals between the DUT and reference subsystem synchronizers and comparators
- Can be in SystemVerilog for DUT in RTL
- Can be in C/C++ for DUT in C based compiled simulators
- Imperas provides templates in C/C++ and SystemVerilog
 - Expect users to extend and customize



ImperasDV setup





- Reference model setup
- Configuration of register and memory initialization
- Selection of what to compare (depends on DUT 'tracer' capabilities) :
 - PC, GPR, CSR, FPR, VR, decode, net, hart...
- Select capabilities:
 - sync-lock-step-compare or async-lock-step-compare
- Trace and logging set up
- Selection of built-in Imperas instruction coverage
- Choice of DV control options







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Drill down into an available commercial RISC-V HW DV verification solution
- Demonstration of ImperasDV in action on open source RISC-V cores
 - SystemVerilog test bench
- Components of RISC-V CPU DV environment



Demo: ImperasDV Core: IowRISC Ibex Simulator: SystemVerilog DV mode: sync-lock-step-compare



- Overview block diagram from RVVI github
- Walk through C/C++ rvvi.h and in doxygen introduce APIs: RVVI-VLG
- Walk through tracer code where it converts RVVI-VLG nets to -> RVVI-API
- Walk through SystemVerilog harness
 - Show init, config, main step loop
- Run example passes
- Run example fails, show trace, show in eGuiMPD and waveforms
- Show arch test suites
- Show instruction coverage



Demonstration / Walkthrough of ImperasDV Commercial RISC-V CPU DV solution Lee Moore







Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Drill down into an available commercial RISC-V HW DV verification solution
- Demonstration of ImperasDV in action on open source RISC-V cores
 - SystemVerilog test bench
- Components of RISC-V CPU DV environment







🐸 Applications 🛛 🐄 workspace - Im	rperasOV 🔜 📦 RVVI IRISC-V Venticatio 📧							
	In the Case of the Party of the Case of th	RVVI (RISC-V Verification Interface): pclude/host/rvvi/rvvi-apl.h File Reference – Mozilla Firefox						
← → C	3 file:///home/moore/imperas/doc/api/rvvi/	itml/rvvi-api_8h.html	② ☆					
imperas	Local Documentation	RVVI (RISC-V Verification Interfac	e)					
mperas Software Ltd. Open Vir	rtual Platforms API Reference docume	tation.						
Main Page Files *			Qr Search					
RVVI (RISC-V Vertication interface)	bool_t rvviRefInsBinComp Compare retired instr	re (uint32_t hartid) iction bytes between reference and DUT. More						
Ψ Fielist Ψ include	bool_t rvviRefPcCompare Compare program co	1						
* 000 * 000	bool_t rvviRefCsrCompare Compare a CSR value							
- Const	bool_t rvviRefCsrsCompar Compare all CSR valu	bool_t rvviRefCsrsCompare (uint32_t hartId) Compare all CSR values between DUT and the the reference model. More						
	bool_t rvviRefVrsCompare Compare all RVV vect	bool_t rvviRefVrsCompare (uint32_t hartid) Compare all RVV vector register values between reference and DUT. More						
	bool_t rvviRefFprsCompar Compare all floating p	bool_t rvviRefFprsCompare (uint32_t hartId) Compare all floating point register values between reference and DUT. More						
	uint64_t rvviRefGprGet (uint Read a OPR value fro	2_t hartid, uint32_t index) 1 a hart in the reference model. More						
	uint32_t rvviRefGprsWritten Read a GPR written m	Get (uint32_t hartId) ask from the last rvviRefEventStep. More						
	uint64_t rvviRefPcGet (uint3 Return the program of	_t hartid) unter of a hart in the reference model. More						
	uint64_t rvviRefCsrGet (uint3 Read a CSR value from	2_t hartid, uint32_t index) a hart in the reference model. More						
	uint64_t rvviRefInsBinGet (u Return the binary rep	nt32_t hartid) esentation of the previously retired instruction. More						
	uint64_t rvviRefFprGet (uint Read a floating point	2_t hartId, uint32_t index) egister value from a hart in the reference model. More						
	void rvviRefVrGet (uint3) Read a RVV vector res	_t hartid, uint32_t index, void *data, uint32_t size) ister value from a hart in the reference model. More						
	void rvviDutBusWrite (u Notify RVVI that the D	nt32_t hartid, uint64_t address, uint64_t value, uint32_t byteEnableMask) JT has been written to memory. More						
	void rvviRefMemoryWrit	e (uint32_t hartid, uint64_t address, uint64_t data, uint32_t size)						

🐸 Applications 🖙 workspace - imperastiv... 🍯 (RVVI (RISC-V Venticatio). 🗵 Terminal

workspace - ImperasDV/lbex/designTop/systemverilog/lbex_testbench.sv - Imperas eGui

File Edit Source Refactor Navigate Search Project Run Window Help

1098

Ę	2	1.	Ξ.	5	9	۰.	15:31	74	Lee	Mo	011
								18	-	-	ŝ

A THE R. M. P. D.

imperas

		6 - 6 - 5 - 5	Carl State Carl		AD ALL AL	Cherry and the second s			THE THE PART OF
Project Explorer 🕴 🥥 🖓 🚔 👘 🗍 🖓	🗉 🖻 rvvi-apiti	the ique ivvi	weigte-invertie	velgeselv 2	vig2log.sv	in iber_rwi_top.av	ibex_testbench.sv 🕸	lbex_alu.sv	
nperasDV	(129 Jf (f	atali begin				10 N			
r 💼 lbex	131 end	uplay(" due to fatal er	norial and and nan	-Tatal errors and 10	d warrungs. unur", arr	cmt, warm_cmt);			
+ 👜 designTop	153 etas 173 501	Begin Solavi" with ABM errors	and and sections	whether ant warm	cotti				
In e +	114 mt	the second second be		later to ma service					
systemvanlog	116 enthunc	tion	ne only call to st	Initian In the bestden	10 10				
F 💩 verilator	110 //////		ummunu		aaaa				
🖌 🔐 Imperas	138 // Test 340 // T	bench control: his task implements the	- total - and - company -	loog					
* 🕞 ImpProprietary	141 task To	CntrlLaap;	o o stra constante a c	2240					
* D-source	143 1 = 0	1							
* 🗁 host	144 // In 145 11 11	reviversionCheck('RVVI	API_VERSION1) begin	UN DUT					
* (Brivi	148 Sdi 147 fat	splay "Mm @ t=b0t: Expe al out():	cting RVVI API ver	nion 90d.", štime, "	RVVI_API_VERSION);				
wig2api.sv	148 wid	0.000111							
I viozina sv	158 17 18	rvviRefInit(text_progra	e sif, "lostisc.dy	pworld.org", "Data W	V3ZIC*) tegin				
* - ImpPublic	151 9di 152 fat	splayi™w @ t≁Witi rvvi al.outii:	RefInit failed", 1	timeli					
x mining	153 end	19030							
w de heat	155 void*	irvviRefCsrSetVolatilei	tb_hart_1d, 32.000	00_0000111 // cycla					11/
T AL DAG	151 void'	(rvviRefCarSeiVoIatile)	th_hart_1d, 12'500	00_0C02)); // insnet					
a Great and b	156 // In 159 ortin	itialize DUT							
Part an mh	111	and the second second for							
in orn-apcorn	161 // St 163 fanw	art step and compare to er begin: Lnop	00						
* is source	163								
* 🗁 nost	165 dot	Step: // returns on los	truction retiremen	t					
* (2000)	10 11	(1=188) TD_ing_external	= 1i						
nvvi-pkg.sv	118. 1149 Los	t = rvviAefInsBinGet10)							
iii rvxi-vig.sv	120 14	(init = ECAL) begin	01. 0241117 01144	102					
/ Group_and_Checks	172 1	T IFINISH ON ECALL beg	tiu	14.					
Control_Files	21178	oreak: nd							
D checkEnvironment.sh	175 end	11000							
defines.QUESTA.Makefile.include	172								
in defines.TARGETS.Include	175 void*	IrvviRetShutdown())							
iii defines XCEUUM Makefile include	100 termi 101 cottask	nate sim[]; // ThCotritoon							
🔤 install.pkg	163								
Makefile.common.include	104 // Matc	hdog timer			mm				
README.txt	185 6lways 185 +++tb	<pre>#(posedge th_CU() begin cyclms;</pre>	the star						
e bests	10 11 11	b cycles >= TIMEOUT CVC	LESI begin	incontrol 117. Atom					
I README.txt	189 18	tal_outil;	a tratter miner t	second of the second					
	192 If ie	rr_ont == MAX_ERRSI beg	din .						
	101 \$d	taplay["vvvillPvill he tal_out();	g t=901: PATAL: t	is many errors/usil)	"+ Stime};				
	114 end	and sector to							
	155 end //)	NUL							the second se

Insert

143 + 11 + 4721
TC 15:52 👣 Lee Moore 10 10

* _ H. X

😬 Applications 🛛 🕫	Serminal St Terminal			N 2 1 1
	Ter	rminal to a Social	Tecninal Tecninal	
m	File Edit View Search Terminal He	ep	File Edit View Search Terminal Help	
Subbletv GBs	ecoro(Lov64761-8		2xfb x5.00001234 -> 00002200 EST_6_100_0000000, "C3430300 scml x5.x5.~072 ".x5-000003000 2xfb 100. "ref%ect/cpv", 5-000000000000000000000000000000000000	65, 85, -872
	50		<pre>#E7.0.187.00000004."80028337 3x1 x8.8x20 ",x8=00020000, Info 187: "xeFRoot/cpu", dx800000000000004(begin_txstcode+290): Machine 80038337 1x1 x 1xFu x8 ffffffff → 60020000</pre>	r8, 8×20
			<pre>EXT_0_IBR_0000002ER_F00430033_BSE1 _ x6_x6_4F0F000000004, Info_IBE1 'ref%uc/cpu', outonouoocoocoosse(begin_testcode+294): Machine 00430333_a061 _ x Info_X6_00000060 -> 00000004 WT_0_IBR_0000006(</pre>	18, 18, 4
ette system			Info 188: 'reffect/cps', Bubbebbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb	15, ŵ(16)
			Info 190: "refenct/cpu", 6x000000000000000000000000000000000000	15, 6x2
		0	<pre>2xfb 1911 'refRuct/cpu', tw000000000004(begin.testcode+2a0): Machine cb028295 addi > 2xfb eE 00002380 -> 00002080 PFT B 102 00000108 'res200000 bu' af mo20 'r af mo2000000000000000000000000000000000000</pre>	15, 15, -846
ikuna			<pre>Info 190: 'refRect/cpu', Sob0000000000000(hegin_hastcode-254): Machine 00000337 lui x Info x6 percent > percente Pri x 20 percent > percente Pri x 20 percente</pre>	×6, 6x26
			INFo 195: 'YefWoot/cpa', b=D000000000000000000000000000000000000	(B. 10, 1
			Info 194: 'refFbot/cps', 0x860000000000178(begin_tastcode-Jac]: Machine 80632023 Nar a Martine 20000074, MolMORDA andi, Mila - standardada - standardada - standardada - standardada - standardada - st	15, 0(16)
1907.005			InFo 196: 'reFEnct/cpu', End0000000000074(begin_textcode-3b6): Machine 00100303 addi = 1 2xFu =0 00002000 -> 0000000).	(5, 20, 1
Weinsetze			<pre>FET, 0, 194, 00000378, "D00000337 lot s6, 0v20 ", x8=00030000,, Info 196: 'reffect/cpe', tw0000000000578(begin,testcode=254): Machine 00020337 loi Info s6 00020008 -> 00020000</pre>	×8, 8×29
题		U	ET. 6. 107, NEWBOOKS, ONDERGOS SW. 85, 0(18) 1976 183: "Influencing," Guidescondergonalistic begin textrade-3583: Machine 00032023 multi ET. 6. 185, NewBooks, "Offlowed" Facce 1976 186. "Influencing," Subproductioned Mid/Learne Factored-3571: Marking Offlowed Facce	$(\Sigma, \Phi(\pi E))$
			WET, D. 150, 00000404, "D0100103 addi. x3, x5, 1 ",x3=00000001, Info 156: "refExec/cpu", Endependence0000464(begin_textcode=3c5): Wachine D0100103 addi. Tofo. x5 (Tffuerc. x benefact).	12, 10, 1
CARDONNE			EXT, 5, 306, 50000458, "DOCOMONY scall EXT, 5, 306, 50000458, "DOCOMONY scall Info 250: "refmac/cps", Extendenceseses(begin.testcode=2c4): Machine 00000073 scall	
			1076 mega 00000144 00000400 1070 megane 00000000 -> 00000000	
			(1/ ibes_textberch.TbEntriloop.Loop @ tr2630es: &EALL) TeFn	
			Info Imperandov VERIFICATION REPORT: Info Instruction retires : 100	
			Info Exceptions : i Info Mismutifes : 0	
			Info Sats / Compares. Info PC 200 / 200	
			Info Instruction : 200 / 200	
			1470 SPR 250 / 5400 1470 SSR 0 / 0	
			14fo FFR 1 0 / 9 16fo V8 - 0 / 9	
			Info Total compares : 6800 Info	
			Test PASSED with D errors and D versings.	
			Simulation complete via Sfinish(1) at time 1630 %5 + 0 /designTop/systemwerllop/ikex_testbench.sv:125 = 3finish; // this should be the only co	All to Sfinish in the testbeeck
			stellano esit	

7001: srue(64) 20-05-5010: Exiting on Feb 04, 2022 at 15:51:18 0MT (total: 00:00:04) Moore@ins6476:-/Dems/ImperasDV/Ibes/systemwrileg5 []

Applications 🔠 Console - SimVision	🦒 Design Browser 1 - Sim (E) Terminal	S Terminal	Waveform 1 - Similian			「日日日 1557 tu Lee
	Design Browner 7 - 51mVIsion		*	Terminal		1.0
TTE THE DHO V B+ BR	Yow gelect Egalare Singlation Windows Help		cade file Edit View Search To	erminal Belp		
58		Waveform 1 - SimVis	ion		(denominally	
le gat yer trajore forget Singleton Windows	Bits				câdence	
Stall on KDDX 3	· · · · · · · · · · · · · · · · · · ·			👹 - 📥 Bella 🦄	光風點照而用	
Sanath Shanay (Sanata) at all all San					AND THE REAL PROPERTY AND A	
				[]		
The second secon	5 11 · · · · · · · · · · · · · · · · · ·			Drs: Eth	0:190m = 35 0	
G Backer-0	Busine + G				THE REPORT OF THE	
R Dam 0+ Dam 0+	0 10ra 20ru 30ru 40ru 50ru	Būra pūra	60ma 60ma 100ma 110ma	120m (130m (140m (150m (150m	na (170na 180na	
	[1 + 1 + 2 + 1 + 3]					
	$[1 \times 1 \times 64 \text{ bits}]$					
a solution 11 x 1	Administrative and the Administrative Science of the Science of th					
B to orderfolioi 'd 1	0				945	
a s vaid 0.0] [1 ± 1.	[1 x 1 bit]					
B & x wb[0.0] [1 x 1*	[1 x 1 x 32 bits]					
B₁ × wdata[0:0] [1 x 1•	[1 x 1 x 32 x 32 bits]					
⊨	[1 x 32 x 32 bits]					
b � ×_wdata[0][0] 'h 000+	00000000_0000000_0000000_00000000_000	000000_00000000_00	00000_0000000_0000000_00000	000_0000000_00000000_0000000	00000000_00000000_0	
æ 🧆 ≪ata[0][0][S1] 'h 000•	0000000					
а 🐢 •ata[0][0][90] '⊁ 000+	0000000					
⊞ 😘 •ata[0][0][29] 'h 000•	0000000					
p 🐝 •ata[0][0][28] 'h 000►	0000000					
в 💁 •ata(0][0][27] 'л 000•	0000000					
a • •ata[0][0][26] 'h 000+	0000000					
в 💁 <ata[0][0][25] 'h="" 000∙<="" td=""><td>0000000</td><td></td><td></td><td></td><td></td><td></td></ata[0][0][25]>	0000000					
B • • • • • • • • • • • • • • • • • • •	0000000					
B	0000000					
	0000000					
	0000000					
	00000000					
e e estelolloll 181 1 h 000	00000000					
e este Ollol 171 h 000	0000000					
a ata[0][0][11] 'h 000	0000000					
ata(0)/0// 15	0000000					
ata(0)[0][14] 'h 000+	00000000					
в 💊 «ata[0][0][13] Ун 000»	00000000					
a 💁 •ata[0][0][12] 'h 000+	0000000					
в 🗣 •ata[0][0][11] 'н 000•	0000000					
	55			j so pao jaja pao pa	0 240 250 253m 🖂	
					1. object selected	

loore



🔁 😳 🖾 🕝 🖾 17:25 📬 Lee Moore

* _ # %

12

Terretital



35 49 **Malastra**

Ville	This Make	Official de la	Theories	Harley.					
	A ANALYSIA		CONTRACTOR OF CONTRACTOR	meth					
DET O	45 00001234 ->-	00002200	a	10 C		0.000			
Shi w	ine, economico, "ca	Augury 200	11 13, 12, -P.	ante pare	104007803	COL.	Concernment of		and the second
lefe.	at Appender	20001000	And the second second	editor rear	reserved)	satura.	CONCERED AN	10.1	101.601.0
897.0.	187 Millional Ton	000037 Jul	al. 5x20		RUMADORN				
lefp 3	#J: 'ref&pot/cpu	·	000000000000000000000000000000000000000	poin tent	code+2901:	Haching	00000007 14	10	16.5120
1efe	AB TITTTTTT ->	00020500							
RET. D.	186,06000028,*00	400013 add	10, 10, 10, 1	÷.,	18-0003500	4			
INFO 1	IBS: 'refficet/cps	. 0x00000	0000000003/8/ 54	egin test	1000-2541	Machine	00430333 88	101	0.5, (5,4)
Infp .	16 00022000 ->.	00025004	244 (BROM)	Sermes	Constraints.				
HET, O,	189,00000366,"00	632823.54	35,8(15)	N	1115				
Info 1	IBS: "refRoot/spa	*, 8x20000	00000000003dc(bi	egin_test	code+398):	Hachine	80632602 ma	A) ()	15,4(16)
HET, O,	190,00000340, "90	002257 840	pi +5, 6+2	51	x5+0000238	Deres			
Infp 3	199: 'refepot/cpa	r, expeese	eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee	egin_teot	code+29c):	MACTINE	00002207 B1	idpr -	15,003
INFO.	A5 00002000 ->	00002360				2			
58.5 , W.	193, 00000004, "CD	SURDER ADD	11 15, 15, -01	ale real	contractory in the local sectory of the local secto	No. of Concession, Name		100	2012
Tella .	at MMMMMerce	NORTHING STREET	concentration of the	elinere electronic	constraint.	10000100	Charlense an	10.1	K2, K2, -9
	THE ADDRESS TOP	CONST 100	10.000		-	0.000			
Teffer 1	100: Profilmer Area	C Subbble	Unine Sector Sector	Ante Ante	radia [nd] :	Barbins	A4406337 7.	a	16.0100
1650	¥8. 00020004	00020000		-g	eres (res)				
RET. 40	183.000003ec."00	434013 add	1.11.11		+8-0000000	6			
tefp 1	193: 'refeset/cee	. subsect		cule test	code+24811	MICTORE	00030315 48	iei.	15, 15, 8
lefp.	36 00600000 ·>	00000006							
RET. D.	194, 6666637'6, "66	632023 Sel	000035,00051	Sec. A.	ana 1997				
Info 3	194: 'refficet/cpu	", 8x80000	ecococcoccorb(b)	egir teet	code+lac):	Rectone	66532622 Te	677	15,4(18)
RET, D.	105,00000374, "08	386255 466	11 15,10,1	10. CM	15-000008	1			
Infp 3	195: "refRoot/cpa	*, \$100000	000000000174(b)	egin_test	code+350):	Rachine	D0100293 Ad	1013	15, 10, 1
1450	15 00002000 (>	00000001							
RET, 0,	106,00606378,100	606037 Jun	x6,8x30	1.1.1.1	*8-9993996	0			
THLE 1	196: 'refficet/spe	", 9v90000	00000000318(84	egan, test	20081284)	Machine	00056333 31	18	18,8128
Infp	×6 00600008 ->	00030000	1	20 m 23					
827, 0,	187,00000374,"00	632823 Se	15,0(10)	ine edd		10.00		212	en anna
anre a	THE ADDREADE AND	famor for	and a second sec	egan_cent	cose+one).	Pacture	annual an		10,0(10)
Terfor 1	DE- 'raffret /res	La De DE DE	CONTRACTOR ADD/ NO	Ante Part	rednaller's-	mehine	pressure r.		
897.6.	150 Decembers "be	OPPORT AND	1 13.15.1		*3.0000000	1		100	
Infp. 3	DD: 'reffeot/cpu	. Gubbbbb	00000000404754	egin tent	code+3c01;	Nachine	00100100 ad	nei.	12,16,1
1nFo	N3 TTTTRECE ->	00000001							10.00
ENE, D.	296, 66666466, *66	0000073 ecs	11						
1850 2	DD: 'refwest/cps	*, expected	00000000406(14	egin test	0000+20431	Machine	00000073 65	111	
Info	matatan 3000188	0 -> 86663	1000	0520220388					
2117-0	Hept 00000144 -	> 00000468	S						
Info .	ricause 100000000	-> 200000	-00						
000.44	sem_testbeech.TbC	strilloop.1	.cop @ t+2638#1	E: EDALL!					
INTE -	Second and a second	UTTER OF	We.						
atro a	Inperaspe vestroit	ALCON REPO	200						
lefe .	Logardines	1100	4						
THE .	Wignorches	1.1							
lefe	Sats / Copperson								
1000	No.	1.1.1	788 / 788						
Infe	Instruction		206 / 200						
tefe	LTN .	2.2	200 / 6405						
Info	CSP		07.0						
1810	FFS	1.1	0/0						
Info	18		0/0						
Tefo	TUEAL COME	eres :	6880						
Info -									
test t	HARRY WITH & HIT	ters and th	WATTOTAL.						
1202									

Simulation resplets via Sfinish(i) at time 2020 AS + 0 .//ResignTop/systemwerilop/ibex.testbench.sx:125 Bfinish; // this should be the only call to Sfinish in the testbench seplime east TOOL: x:res(04) 20.05-s010: Exiting on Feb 04, 2022 at 17:23:34 GMT (total: 00:00:04) moorngline5476:-/Dems/ImperasDW/Ibes/Systemwerilog& []

17:29 to Lee Moore

.tt. 36

Terminal

Terminal. File Edit View Search Terminal Help 000r003nx8478:-\$ Souther state

CHILL SPECIMIN

T



2 1003 146 CONTRACTOR OF

and the second s	File Edit View Search Terminal Help
	Linding snapshot verklib ibes testbench by
	ensim: "w.DEPODDD: This Systemverileg design is simulated as per TEEE 1000-2000 Systemverileg simulation semartics. Use -disable sem2000 option for turking o PF Sv 2000 minulation semartics. stelling- specto /home/nds/samece/installs/REELING2000/tools/scellum/files/smminrs
	<pre>intribut ris lest.testienth.th_intrl@tabes isading test_program /bome/mocre/Demo/Demo/Demo/Demo/Demo/Demo/Demo/Lest/systemwerllog/werk/1-ADD-01.kes info (DR_SF) Target 'refRoot/cpo' has object file read from '/bome/mocre/Demo/Demo/Demo/Demo/Demo/Demo/Demo/Dem</pre>
	111 ibas rastbooch. TSCntrlicop.icop # t=26180n: EFAL1 1870. 1870. Doperandy VERFICATION GEPERT: 1870. ISCratice retires : 100. 1870. Exceptions : 1

Terrettal.

1670	Instruction retires	- 1	190		
Inf a	Exceptions	-	1		
INFR-	Misnatches	1			16.52
hfp.	Sets / Compares				lat .
INFE-	PC	1	200	1	200
lefe-	Instruction	-	200	1	290
1870	079	1	200	1	6400
lefo .	423			2	
infin :	1111	1	0	7	
infp.	VR .		0	1	
tef o	Tital compares	÷1 –	1000		
info -					

INFo (ITV.MD) Reading file /home/more/Deme/ImperasDV/IDex/Systemverileg/I-ADD-01.basic.coverage.yuml Info (ICV_RD) Reading file /home/moore/Demo/ImperanDV/Ibes/systemserilog/I-AMD-61.basic.comerage.yaml

10fb (ICV_RD) Reading file /None/moore/Deno/ImperasDV/Ibex/Systemwerilog/1-8L7-82 basic coverage yand Info (ID/ 6D) Reading file /home/home/bone/ImperasDv/lben/hystemverilog/1-U4-Di.tasic.coverage.yanl

10Fe (IDX.MD) Reading file /home/more/Deme/ImperasDv/IDex/systemverileg/I-D4-D1.basic.coverage.yaml

Info (IDV WCR) Writing coverage report collate.basic.coverage.tet

INTO ADDRESSANDARESSAND INFO TOTAL INSTRUCTION COVERAGE : RV001

INTO THRESHOLD 11

Info Instructions counted : 180

122

- INTO UNIQUE INStructions | 15/30 | 33.33%
- Info Coverage points hit : \$803/2548 : 18.88%
- 1870 ------

Text R&SSED with 8 errors and 9 wernings.

Simulation complete via Sfinish(1) at time 2000 MS + 0 ./designTop/systemwarilog/itex restherch swills Brinish; // this should be the only call to Sfinish in the textbeech sceline exit T00L: srun(64) 20.03-eDi8: Emiting on Feb 84, 2022 at 17:27:36 GMT (total: 00:00:02) neere@lsw8478--/Demo/ImperasDV/Ibex/Systemverilog5 grep -+ "Doverage points hit" ".tat collate basic coverage.tet:1182:1nfo Coverage points hit : 505/3540 : 10.005 1-ADD-D1.Basis_coverage_txt_3182:10fb__toverage_points hit i 253/2540 : 0.06% 1-AMD-DI, basic coverage tot 2182:2nfo Coverage points hit : 253/2540 : 0.60% 1-5LT-53 basic coverage txt:3382:3xf0 Coverage points tot | 185/2540 | 7.44% 3-15-01.basir.coverage.tst:1182:Info Coverage points hit : 105/2540 : 7.685 2-54-02 basic coverage tot 2382 2470 Coverage points hit 154/2540 8.08% noore@lasH436:-/Deno/InperasDV/Ibes/systemverilog5

Demo: ImperasDV Core: IowRISC Ibex Simulator: SystemVerilog DV mode: sync-lock-step-compare



- Overview block diagram from RVVI github
- Walk through C/C++ rvvi.h and in doxygen introduce APIs: RVVI-VLG
- Walk through tracer code where it converts RVVI-VLG nets to -> RVVI-API
- Walk through SystemVerilog harness
 - Show init, config, main step loop
- Run example passes
- Run example fails, show trace, show in eGuiMPD and waveforms
- Show arch test suites
- Show instruction coverage







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Functional Coverage
 - Instruction Set Simulators



RISC-V RISC-V International's compliance tests



- RISC-V International has been working on compliance testing since 2018
- Status (Feb 2022):
 - Test suites for basic un-priv older ratified extensions I, M, C, etc
 - Simple framework for running DUT and provides signatures for comparison
 - Working on new framework to run in a post simulation signature compare mode
 - Encapsulates sail model, uses yaml configuration, does not provide build in reference signatures
 - Process is self-certification
 - You run the tests on your DUT and declare it is RISC-V compliant



RISC-V International Compliance Special Interest Group - Charter (aka compliance working group)







RISC-V International Compliance Special Interest Group (2) (aka compliance working group)

mperas





RISC-V International Compliance Special Interest Group (3) – Mailing List (aka compliance working group)

mperas





RISC-V International Compliance Special Interest Group (4) - GitHub (aka compliance working group)



II dus-not-tia//ke/-arch-bed_lies				
Home During H. B. Manners H. G. An	an Elevant II an Olsa,	n Ende		
	P mage + P 2 formitte	0.000 mm m m	ta Cant	Mount
		() (In the subscriptle) is set of the set of	(D) 216 commits	We development a
	an amazantan	samewrith or with credy incredenting.	Twisting	· vocision
	H 40	testering old ross-peoplering tittly will rais door and right.	10.month age	1 250 years
	monapato	munistrati America (C. S. Jur. 2014). April March April 1993	to months age	A ALTER .
	B Hortegel	Setting of the state in the backward products.	3 salathc age	9135an
	In this set where	terrar (U. No	10 receiptings.	Revenue (ii)
	 movies and 	service day file	+ in other large	D 441 (1915
	B meretale	Facilitating projection and	Territory	without
	Dec dec	appended spins for PV/UG7, SGC/20,7 mm/r)	3 ecotits ago.	1.11.0004004
	B papers	Decluptions makeline	() moto aga	
	B Ownersonwi	farger (in WVEP), (F. (Parille), many	Transfer age	Packager
	D International	types from	12 months age	the state of the state
	3 IDMANDAMO4	aposted laterate that are contently used by term	iù monta aga	
	TE COPYNOND	shift present dataal to the Welcher	Kynes ogs.	CONTRACTOR
	D convisor	and present densities the WADAC	Owninge	
	3 Manufin	aided layered if fogde to aid tech	ii suth ip	- 20
	D Hallefie Periods	tore had	i) motivage	+ 11
	The second s	inclusion have been been as the fact that have been as a second second second	Taxable also	
NRISC-V	D TEADOCOM	money will be print will not the mean and all sold		



Page 90

RISC-V International Compliance Special Interest Group (5) – GitHub – test suites (aka compliance working group)



P restor - show with hard show it	uit salle/	Kotolfik Addrile*
• ++++++++++++++++++++++++++++++++++++		ninima #12516-2021 (@He
M 447	Ka Kutazi, W Devala mene	Liebbe
 S2compted 	Participant in Dougrand Resid Course-Officer and Development	210000
a ang a	is the care strip to be and be	
	Automatical and a second second	1.0
D weekstoon	CREATEN THATA	1344200
RISC-V Test S	i uites eCoolthe Officient waterialism subsets of the RISC-V upprovinged KA. The tests unitity b	alizer the Sw2 frame
RISC-V Test S	Suites eCoordine of Newerl waternicky subsets of the RSIC-V sciptivilegied RA. The tests unitity b	alizer Ster Sed Auroral
READWOINNE RISC-V Test S The feature grouped law spectrumon cristing names positive?	Suites eC on the different extension subsets of the RISC-V upproleged RA. The tests unitity to with "Jensified" redicate that tests for extensions that have not yet been railfed by m	alize the Sed format
READWOINNE RISC-V Test S The feats are grouped loss specification. Origitary names peethed The courses opent (2010)	contresting of the text and a subsets of the RIC-V soprovleged Kit, The text and by the with "Jenzified" indicate that tests for extensions that have writed been calified by the millionary of the texts available in this subs is generated through RICCOF and is available	olicia Sie Teib farmat VI. de ferrer Conerrigo Report.
READWORNE RISC-V Test S The heat are gatapediate aperitation Chattery names positive The coverage operit (w/m There had have been gen	Suites ef on the Officer (waterials wheels of the RISC-V opproleged KA. The tests unitally b with "JumaBlod" indicate that tests for antensions that have writed biom sailfed by to inflammat of the tests available in this sube is generated through RISCOF and is available ended using the oper sevene Compatibility Test Generater from InCom Services Judge	olizer She Sed harriet VI. de forer: Coverage Report. 11 analebte et: 1775,
RESOURCESS RISC-V Test S The least are prospections apartication. Creating runnic posthap The coursign oppart (with These least tree been per The selevenie cignature A	Suites ef on the offleering extension subsets of the BISC-V upprivileged KA. The tests unticly to with "Jumatilited" indicate that tests for antensoris that have with yet been calified by th millionnal; of the tests available in this sub- is generated through ROCCE and is available extend owing the open sources Compatibility Test Generated from InCome Demiconductor re-generated using CAL or CHIL.	olizer Ste Sed Azmai VI. 1 analetie et 175,
READVEINE RISC-V Test S The feature grouped loss specification Crisicitery numes positive The course ge capacit do the These features are cigantance a Test directory with the ", critication	Suites et on the officer (extension schedu of the BISC-V opproleged KA. The tests study to with "_emailfood" indicate that tests for antensions that have not yet bloom calified by its millionest; of the tests available in this sube is generated through RB2CP and is available ended using the oper second Compatibility Test Generates from InCom Demiconductor re-generated using LAL or CPHIL annethed" post-faced cale test scalars for extensions which there not been withed duct	olizer She Sed Aurren VI. de foren: Coneraço August. n analable et 1757, ang dad la and mar
READVENIES RISC-V Test S The lease are proportions apostoation Oractary runnic position The courage operation of The courage operation of The reference organizes of The reference organizes of Constany structure	Suites et on the different extension subsets of the BISC-V soprovleged BA. The tests strate to with "excelled" indicate that tests for extensions that have wright been callfed by in inflormer; of the tests evaluation in this rate is generated through BCCCF and is well extend origin the oper source. Compatibility Test Generator from InCore Servicesductor regenerated using SAL or SPAL. envertised" post-failed call of SAL or SPAL.	alizer Ste Sed farmat (M. Is foren Colempo Report. n analable at 1705, any diable at 1705,



Page 91

RISC-V International Compliance Special Interest Group (6) – GitHub – test suites (aka compliance working group)

mperas

P mater + may arch fait / way test subs / rd	2, n/	Conteinte Ageinte
Recipitant is when case strings for twand two		(Herzu in 22 Oct 2021 (S) History
- • c	Compliance Take Group stranged to Autolecture Test SrG man docs and	10 months agai
	the rotant case strings for file and how	4 etatititaga
- BB - 1	To the soler case 3. In the BREST, GRE, means used in the shift	4 months ago
Muraniten	thursference, actual for sec32 tests	4 morths200
N	Compliance Test Online changed to Architecture Test STG in all docs and	ogs princes di
Directal	Compliance fails Group charged to Architecture failt SIG in all Bolz and	10 mentre apo
1 0%/itgt	corrected string "EBREAN" in to dring matrix to "ECAUL" for equils $\tau_{\rm eff}$	5 morths apo
D gitgrare	rhj-tratu (*154)	14 minthsage



Imperas Test Suites



- When verifying a CPU design you can never have enough tests...
- Imperas have developed a directed RISC-V test generator, instruction coverage measuring VIP, and a test qualifying mutating fault simulator to provide high quality test suites
- The generated tests suites are targeting architectural compatibility as defined in the RVIA architectural test working group coverage requirements
- There are currently over 50 free test suites, including
 - I,M,C,F,D,B,K,V,P
 - The provided vector test suite is one specific vector engine configuration
- The test suites are provided under an OVP open source license and are available free from: <u>https://github.com/riscv-ovpsim/imperas-riscv-tests</u>





This download contains a binary of the Imperas configurable reference simulator, a text framework to run the simulator or your device-under-test, and tests to run on several targets. Use the tests to check ISA compliance of your targets.

Bremen Univ. 'Fuzz' tests



Closing the RISC-V Compliance Gap: Looking from the Negative Testing Side*

.

- + 160%

Vladimir Herdt¹ Daniel Große^{1,2} Rolf Drechsler^{1,2} ¹Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany ²Institute of Computer Science, University of Bremen, 28359 Bremen, Germany {vherdt.grosse.drechsle}@informatik.uni-bremen.de

Abstract—Compliance testing for RISC-V is very important. Therefore, an official hand-written compliance test-suite is being actively developed. However, besides requiring significant manual effort, it focuses on positive testing (the implemented instructions work as expected) only and neglects negative testing (consider illegal instructions to also ensure that no additional/unexpected behavior is accidentally added). This leaves a large gap in compliance testing.

In this paper we propose a fuzzing-based test-suite generation approach to close this gap. We found new bugs in several RISC-V simulators including *riscvOVPsim* from Imperas which is the official reference simulator for compliance testing.

I. INTRODUCTION

An Instruction Set Architecture (ISA) defines the interface between the Hardware (HW) of a processor and the Software (SW). While as a consequence, the format of a SW binary running on that represents the output of the test result and is dumped at the end of the test execution. For compliance testing, these signatures are compared against golden reference signatures (obtained by running the test-suite on a reference simulator). A separate sub test-suite is developed for the RISC-V base ISA as well as for each standard ISA extension. Besides the significant manual effort for the maintenance, the compliance test-suite focuses on positive testing only, i.e. to show that the implemented instructions work as expected. However, it neglects negative testing, i.e. to consider illegal instructions to also ensure that no additional/unexpected behavior is accidentally added. This leaves open a large gap in compliance testing.

Contribution: In this paper we propose a fuzzing-based test-suite generation approach to close this gap. We leverage state-of-theart fuzzing techniques (based on LLVM *libFuzzer*) to iteratively generate test-cases which are executed on a RISC-V simulator and



1 dt 6

		A Manual Andrews	Automatica Automatica and a Automatica and and			
Automatic Automatic Interesting of automatic Due (now 2) Interesting of automatic	€) → @ @	A https://github.com/riscv-ovpsim/in	peras-mucv-tests/bidb/maiv/riscv-test-suite/rv32vi/basic-coverage.txt	(90%) ···· 🖾 🕁	± IN (D @ #* Ξ	
Number legend wind is kest Number legend Number legend Image: Section S		Ay, 1 contributor		R		
Impres Number Impres Num Impres		anan lines (2003 alic) 402 KB		200 Bate / 0		
		1 Imperes HIDC-V Instruction Coverage	Report 5			
		<pre>datestion : MAN instructions: 12 vest_via vest_via </pre>				
		n stan				
View 2/2 : 200 - 200 View 2/2 : 200 V		neg. 12/3.	188.00%			
Image 20 Image 20 Image 20 Image 20 Image 20 Image 20 <tr< td=""><td></td><td>- sign Arz : pos.som</td><td></td><td></td><td></td><td></td></tr<>		- sign Arz : pos.som				
Wit 40		11 1mm 2/2 1 µ30 .000				
1 40 47 40.000 1 47.0 80.800 1 10.000 10.000 1 </td <td></td> <td>LL VBF</td> <td></td> <td></td> <td></td> <td></td>		LL VBF				
14 14/2 16/2 13/8 14/2 16/2 13/1 14/2 16/2 13/1 14/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 16/2 13/2 12/2 12/2 13/2 12/2 12/2 13/2 12/2 12/2 13/2 12/2 12/2 13/2 12/2 12/2 13/2 12/2 12/2 13/2 12/2 12/2 13/2 12/2 12/2 13/2 12/2 12		10 0/1 1	8.30% ZERO			
1 47.1 47.1 48.4 MA 1 47.2 48.4 MA 1 47.4 48.4 MA 1 48.4 MA 48.4 MA 1 47.4 48.4 MA 1 47.4 48.4 MA 1 47.4 48.4 MA 1 47.4 48.4 MA 1		······································	Los orts			
1 121.4 100.400 142.3 100.400 143.4 407.5 100.400 143.4 407.5 100.400 143.4 407.5 100.400 143.4 100.5 100.400 143.4 100.5 100.400 143.4 100.5 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 143.4 100.400 100.400 144.4 101.4 100.400 145.4 100.400 100.400 145.4 100.400 100.400 146.4 101.4 100.400		(i) v13 8/3 ;	100.00%			
visit visit visit visit visit visit visit <		413 E1/1	195.000			
113 312, 103.000 117 312, 103.000 118 312, 103.000 119 312, 103.000 110 312, 103.000 111 312, 103.000 112 312, 103.000 113 312, 103.000 114 312, 103.000 117 312, 103.000 118 312, 103.000 119, 000,000 <tr< td=""><td></td><td>11 VIA 8/2 1</td><td>las am</td><td></td><td></td><td></td></tr<>		11 VIA 8/2 1	las am			
$ \begin{array}{ccccccc} 1 & 13 & 1 & 10 & .00 \\ 1 & 17 & 1 & 10 & .00 \\ 1 & 13 & 0 & 1 & 10 & .00 \\ 1 & 13 & 0 & 1 & 10 & .00 \\ 1 & 13 & 10 & .00 \\ 1 & 13 & 10 & .00 \\ 1 & 13 & 10 & .00 \\ 1 & 13 & 10 & .00 \\ 1 & 13 & 10 & .00 \\ 1 & 13 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 1 & 10 & .00 \\ 1 & 1 & 1 & 1 & 1 & 10 & .00 \\ 1$		10 y28 3/2 (109.50%			
121 $9/2$ 100.000 172 $0/2$ 100.000 173 $21/4$ 100.000 170 $21/4$ 100.000 170 $21/4$ 100.000 170 $21/4$ 100.000 170 $21/4$ 100.000 172 $21/4$ 100.000 172 $21/4$ 100.000 172 $21/4$ 100.000 172 $21/4$ 100.000 172 $21/4$ 100.000 172 $21/4$ 100.000 172 $21/4$ 100.000 172 $11/4$ 100.000 173 100.000 174 $11/4$ 100.000 174 $11/4$ 100.000 174 $11/4$ 100.000 174 100.000 100.000 174 100.000 100.000 174 100.000 100.000 174 100.000 100.000 174 100		10 918 HB/L 1	100.00%			
472 472 $100.$ 472 $124.$ $100.$ 472 $124.$ $100.$ 472 $124.$ $100.$ 472 $124.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $120.$ $100.$ 472 $100.$ $100.$ 472 $100.$ $100.$ 474 $120.$ $100.$ <td< td=""><td></td><td>13 938 8/1 1</td><td>100.09%</td><td></td><td></td><td></td></td<>		13 938 8/1 1	100.09%			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		10 V19 B/3 :	100_005			
V25 3/2 2 100-000 V25 0/2 0/2 0/2 0/2 V25 0/2 0/2 0/2 0/2 0/2 V26 0/2 0/2 0/2 0/2 0/2 V26 0/2 0/2 0/2 0/2 0/2 V26 0/2 1/2 0/2 0/2 0/2 V27 0/2 1/2 0/2 0/2 0/2 V28 0/2 1/2 0/2 0/2 0/2 V20 0/2 1/2 0/2 0/2 0/2 V21 1/2 0/2 0/2 <td< td=""><td></td><td>10 V2 0/3</td><td>109-00%</td><td></td><td></td><td></td></td<>		10 V2 0/3	109-00%			
92 92 100.000 92 32 100.000 924 32 100.000 924 32 100.000 924 32 100.000 924 32 100.000 927 32 100.000 928 32 100.000 929 32 100.000 920 32 100.000 921 32 100.000 928 32 100.000 929 32 100.000 920 32 100.000 921 32 100.000 924 32 100.000 924 32 100.000 925 100.000 100.000 92 32 100.000 92 100.000 100.000 93 100.000 100.000 94 52 100.000 95 100.000 100.000 94 52 100.000 95 100.000 100.000 95 100.00		11 Y25 B/5 1	199. Mile			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		10 922 0/3.1	10a. am.			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		123 3/3	100.000			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		11 120 8/2 1	100.000			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		12 128 972 1	102.00%			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		10 927 B/L 1	189.00%			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		V28 21/1 :	100.00%			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$			102 00%			
v21 3/3 1.000,0005 v4 21/2 1.000,0005 v5 3/4 1.000,0005 v4 0.1/2 1.000,0005 v5 3/4 1.000,0005 v6 0.1/2 1.000,0005 v7 3.1/2 1.000,0005 v8 0.1/2 1.000,0005 v9 0.1/2 1.000,0005 v9 0.1/2 1.000,0005		11 Y26 9/5 1	100.005			
v4 F1/5 100.00% v5 11/5 100.00% v8 0./2 100.00% v9 0./2 100.00% v9 0./2 100.00% v9 0./2 100.00% v9 0./2 100.00%		(i) V24 8/3.1	109.005			
YD 13/2 1000,0005 YD 0/2 1000,0005		10 V4 ELG 1	100.00%			
10 17 17 18 100 000 10 10 10 10 100			109_00%			
		17 17 8/5 1	Los des			
		40 40 30/1 :	196 - 90%		1000	





- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Functional Coverage
 - Instruction Set Simulators







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Google Cloud riscv-dv (SystemVerilog open source)
 - OpenHW Group force-riscv (C++ open source)
 - Valtrix STING (commercial)
 - Functional Coverage
 - Instruction Set Simulators



CPU DV test bench components mperas





Constrained Random Instruction Stream Test Generators (ISG)









Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Google Cloud riscv-dv (SystemVerilog open source)
 - OpenHW Group force-riscv (C++ open source)
 - Valtrix STING (commercial)
 - Functional Coverage
 - Instruction Set Simulators



Key Features



01

02

03

Randomness

Randomize everything: instruction, ordering, program structure, privileged mode setting, exceptions..

Architecture Aware

The generated program should be able to hit the corner cases of the processor architectural features.

Performance

The instruction generator should be scalable to generate a large program in a short period of time.

24

Extendability

Easy to add new instruction sequences, custom instruction extension, custom CSR etc.

From Google Cloud presentation 2019 RISC-V Summit



Randomness

Instruction level randomization

Cover all possible operands and immediate values of each instruction Example: Arithmetic overflow, divide by zero, long branch, exceptions etc.

Sequence level randomization

Maximize the possibility of instruction orders and dependencies





Program level randomization

Random privileged mode setting, page table organization, program calls



Google RISC-V Instruction Stream Generation

- High quality SystemVerilog UVM DV infrastructure
- Open source
- Drives a RISC-V core through corner cases and pushes it to the limit



Imperas worked on this with Google Cloud & Metrics through 2019-2020

mperas

- Uses a post-sim trace-compare methodology
- Uses Imperas riscvOVPsim as the reference



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Google Cloud riscv-dv (SystemVerilog open source)
 - OpenHW Group force-riscv (C++ open source)
 - Valtrix STING (commercial)
 - Functional Coverage
 - Instruction Set Simulators



OpenHW Group force-riscv



- Developed initially by Futurewei
- Open source C++
- https://github.com/openhwgroup/force-riscv
- Initial focus on RV64, recently working on RV32
- Not yet key part of OpenHW flows for core-v cores



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Google Cloud riscv-dv (SystemVerilog open source)
 - OpenHW Group force-riscv (C++ open source)
 - Valtrix STING (commercial)
 - Functional Coverage
 - Instruction Set Simulators



<) → @ @	A https://www.valtrix.in/sting/	回 ···· 回 ☆	± IN 00 00 11 [°] Ξ
	Valtrix Systems	Home Company • Products • Services Blog Contact	
	STING - A Versatile Design Verification Platform		
	STING, the flagship product of Valtrix Systems, is a bare metal verification of IP/SoC implementations. The software stack consists which can be configured into a portable program as per the needs simulation. FPGA prototypes, emulation or silicon and execute the programs or requests for.	software specially designed to serve as a platform for the design s of test generators, checkers, device drivers and a light-weight kernel s of the verification environment. The program can seamlessly boot on a constrained random, directed or coverage based tests that the user	
	The highly portable stimulus is controlled by a rich file based test s for every test parameter so that every test condition can be mappe	pecification scheme. High level of controllability is provided to the user d to a particular test configuration.	
	STING is developed with a vision to solve problems commonly se methodologies and practices in the industry whilst providing in Designed for scalability and extensibility, companies can make full	een in design verification and system validation. It embodies the best novative solutions for the unique challenges in specific ecosystem use of it across the spectrum of embedded, client and server SoCs.	
	IMPORTANT FEATURES		
	Stable and deterministic kernet with a tiny memory and instruction footprint ideal for simulation environments	Extramely fast test generation and execution to cover a large amount of verification space in a small amount of time	
	Run the exactly same portable stimulus on simulation, FPGA prototype , emulation or silicon without any change	Extensive hardware support including ARMv8, RISC-V and USB. Check the section below for details	
	Generates extremely tight sequences of code for faster closure on coverage	Interspersed directed and random testing for better coverage under different levels of stress	
	Configuration file based input to control kernel setup, test generation and execution	Special kernel and library APIs for design verification available to test developers to write stimulus generators	
	Clock, power, memory and interrupt management support provided by	Support for slandard verification algorithms is available with the library of	





- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Functional Coverage
 - Instruction Set Simulators



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Functional Coverage
 - Imperas Built-in Instruction Coverage
 - SystemVerilog Covergroups, Coverpoints, Assertions
 - Instruction Set Simulators



Imperas Instruction Coverage



- Built-in as 'extension library' to Imperas models
- Works with all Imperas simulators and RISC-V models
- Focus is for measuring 'architectural validation tests'
 - i.e. measurement of basic architectural operations
 - Not intended for measuring HW DV testing (use SystemVerilog for that)
- Controlled from command line tailor options for each run
 - Outputs .txt file, and .yaml file of data
- After individual runs, collates data into 'suite measured coverage'
- Shows what has, and has not, been covered
- Selectable coverage focus
 - --extensions, --instructions
 - --mnemonic, --basic, --extended



ImperasDV Instruction Coverage





- Reference model setup
- Configuration of register and memory initialization
- Selection of what to compare (depends on DUT 'tracer' capabilities) :
 - PC, GPR, CSR, FPR, VR, decode, net, hart...
- Select capabilities:
 - sync-lock-step-compare or async-lock-step-compare
- Trace and logging set up
- Selection of built-in Imperas instruction coverage
- Choice of DV control options



ImperasDV Instruction Coverage



Page 116



- Reference model setup
- Configuration of register and memory initialization
- Selection of what to compare (depends on DUT 'tracer' capabilities) :
 - PC, GPR, CSR, FPR, VR, decode, net, hart...
- Select capabilities:
 - sync-lock-step-compare or async-lock-step-compare
- Trace and logging set up
- Selection of built-in Imperas instruction coverage
- Choice of DV control options



ImperasDV includes built-in Imperas Instruction Coverage


Historopsim / imperas-ristor-tests - Percenter

(b) East: ① Journal A Think The Section ② Advances ③ Advances ③ Properties ③ Market ③ Security 22 (respired)

P x2022012 + P 7 minut	um 🖉 0 tage	Go to file Ade Sik * CON++	About
🗮 duncangneham-Impanas upd	telle to villollol113	poliar 24 daysage (S) Ricerterty	No description, website, or topics provoded.
📷 (teci ovpelini pluk	uplies to (0520) 12	2+ days app	12 32 mint
TTCs digitite	Leaderty for callocade 12	24 staje spa	© iendvg.
In tac-target	upitiate to y20020112	24 thun kpc	A a due
nici-tatt-ett/	update to v20011010	4 months aga	Deleverer
In techteb-sute	update to \30020112	14 days age	increases
Completions	Location to v20022012	24 stays with	Chains a real ablance
D Maxia	upitiate to y20020112	34 days age	
Aserenisie	website for VEIIIII0112	24 steps aga	Packages
D READING ond	updata 19-000011117	Lincoln app	Pie dadoegos published Publiki your finit partage
SEADWEINE		×.	Contributors ±
Imperas RISC architectural	-V riscvOVPsim refer validation tests	ence simulator and	damongrafiam-Impress Denter Croner Impress institut Officia Account () rock-implies
compling, no tidaling, and	no external dependencies, it just works-		sanwadi

In the RISC-V world, Imperas simulators are used by most companies and organizations that are serious about getting quality RTL vorking and signed off. They use it as an architectural reference and many use it as the golden simulator to valify their R7L in a hardware design verification methodology, imperes simulator technology and verification P is the technology to use to obtain high quality results and to verify RTL works as expected.

The following is a fist of some of the companies and organization: that rely on imperas simulators as their RISC-V reference:

Melanov/Nikika, Seagate, NSI/TEXE/Denso, Google, Cloud, Chips Alliance, IowR/SC, OpenHW Group, Andes, Valibix, Nagra/Sudelski, Silicon Labs, Incore Semi, FUSC-V Compliance Working Group, Symbiotic EDA, Thales, Hersoldt Cyber, 111/0....

This download contains a binary of the Impetes configurable reference simulator, a test harnework to run the simulator or your device-under-test, and tests to run on several targets. Use the tests to shock ISA compliance of your device.

This Imperasitest framework has formed the basis of the RISC-V international intoxicity! Compliance Working Group's

100	ana phinist
sele	iges
t out	kagos sublished your finit parnage
ant	rbuters ±
¢.	dunxengrafiden-Impress Denter, Orsheim
-	Inserving Insperso Official Appoint
8	uraca-nadingua

Languages

♦ < \$1.4%</p>
♦ Associate 19.2% · Shell 1.11 - Other 1.95

Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Functional Coverage
 - Imperas Built-in Instruction Coverage
 - SystemVerilog Covergroups, Coverpoints, Assertions
 - Instruction Set Simulators



CPU DV test bench components SystemVerilog Functional Coverage





Recall there needs to functional coverage connected to the 'tracer'



To measure what the architecture and micro-architecture is doing...

CPU DV test bench components SystemVerilog Functional Coverage (2)





- And we saw earlier in the detailed ImperasDV walkthrough the connections from the 'tracer' via the RVVI
 - From the standard RVVI-VLG there are reusable source 'clients' that connect the 'tracer' to other components



CPU DV test bench components SystemVerilog Functional Coverage (3)





• We can connect a functional coverage subsystem in the same way



Connecting RVVI to Covergroups

mperas

- VLG2COV connects to 'tracer' and is informed when events like an instruction retires or interrupt is taken
- On these events VLG2COV
 - calls 'sample' in coverage class
 - Which then calls the appropriate covergroup sample functions

```
100
         covergroup ins cg with function sample(string ins str);
 387
             option.per instance = 1;
 3.02
             cp asm : covergoint get asm enumiins stri;
 383
        endgroup
 384
 3.05
         function new();
 3.65
             Lts cg = new();
 3.87
         endfunction
 108
 105
         function void sample(input string decode);
 110
             string ins_str, op[4], key, val;
 111
             int num = $sscanf (decode, "%s %s %s %s %s %s", ins_str, op[0], op[1], op[2], op[3]);
 117
             ins cg.sample(ins str);
         endfunction
 114
 115 endclass
 -116
       covergroup sub og with function samplelins t ins).
296
           option per instance = 1;
297
                    : coverpoint get upr name(ins.ups[0].val, ins.ups[0].key, "sub");
           CD rd
798
                     i coverpoint get ger name(ins.egs[1].val, ins.egs[1].key, "sub");
           CD_rsl
799
                    : coverpoint get_ggr_nome(ins.ogs[2].val, ins.ogs[2].key, "sub"):
           CD /52.
808
       endorpup-
1001
802
       covergroup sw cg with function sample(ins t ins);
           option.per instance = 1;
12
894
                    : coverpoint get gur name(ins.ops[0].val, ins.ops[0].key, "sw");
           CD FS2
805
                    : coverpoint get gpr mamelins.ops[1].val, ins.ops[1].key, "sw"];
855
                   : coverpaint get ins(ins.sps[2].key, "sw") {
           CD 100
887
               bins neg = ([5:-1]);
800
               bins zero = {0}
809
                bins pos = {[1:5]};
818
           . ).
811
       endgroup
012
013
        covergroup wfi og with function sample(ins t ins);
814
           option per instance = 1:
815
           cp_asm : coverpoint ins.asm == MFI {
B16
                ignore bins zero = {0};
1117
018
       endgroup
819
112.9
       covergroup sor og with function sample(iss t ins);
821
           option per instance = 1;
102
                   : coverpoint get gpr_mamefins.ops[0].val. ins.ops[0].key, "xor"):
           cp rd
87.1
                   : coverpoint get ger name(ins.ops[1].val, ins.ops[1].key, "xor");
           cp ral
824
                    i coverpoint get ger name(ins.eps[2].val, ins.eps[2].key, "xon");
           CD_FS2
       endgroup
```



RVVI Functional Coverage



- So the use of a standard interface from DUT 'tracer' to testbench means reusable standard components can be developed
 - A key one is a SystemVerilog Functional Coverage sub system
- And for RISC-V standard extensions they can be provided as SystemVerilog source
 - And then extended for DV specifics of the specific micro-architecture
 - Such as pipeline issues, hazards, assertions design specific items
- Imperas has an example available (Feb2022) and will shortly release others
 - (contact Imperas for more information)



SystemVerilog simulators provide Coverage reports

Imperas

Class Type Class Type Converge	 None 	iGoosi Type	Ceverage -	10eM	- Mini Goak	Skilos Web.d	leij Merge Ivelano	en live a	A CONTRACTOR		
Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Methods, go Method	🕁 📕 Mater, entr jokgits	en jent over group	71.21								
Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold Market wold Market wold Market wold Arrowsky me Market wold <	gi 😹 THYE anti_ og	FRO(308_30	116.004	- 39	0 L05.00M	· · · · · · · · · · · · · · · · · · ·		##C0			
Image: status Image: status<	TYPE G. R. KD	DBC MIX, 00-	100.004	· 00	0 CLOL IN 4		1	4.16(1)			
Image: class type Im	ALL THE ADD OF	Files and the			0.0000	5	1.5	ALC: NO. 10			
Image: Sector	e a Weater of	Fact and the	10.00	- 50	č –		1.1	andi b			
Image: Project of the state of the stat	5 A THEAR.IS	FORN BLIEF, MIL	110.004	1 10	0 120.000	/		witeco P	-		
i Tref and you Tref and you Disc 100 0000 Disc 100 0000<	A # TYPE # 40	Fight #10_00.	100 809	32	D LUB SOM	1		ie8400			
Image: State of the set	S A THE ST ST	risev_linit_to.	110,000		0 1.05-364		1.1	erte(1)			
Total International Control of the set of the	S S WEIN TO	Figh_Hid_ID.	6.204	- 10	0.000			with(23)			
Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million Million <td>a a The state of</td> <td>FOR FREE A</td> <td>100.000</td> <td></td> <td>0.100.004</td> <td>5</td> <td></td> <td>e ##())</td> <td></td> <td></td> <td></td>	a a The state of	FOR FREE A	100.000		0.100.004	5		e ##())			
S TWO 5 F Comparing * Name Class Type Cowrage Coal % of Ocal Status Mcbuded * Name Class Type Cowrage Coal % of Ocal Status Mcbuded * Tetrat, g * TYPE add, g * CVP add cg: cp : cp : n1 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add cg: cp : cp : n1 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add cg: cp : cp : n1 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add cg: cp : cp : n2 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add cg: cp : n1 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add cg: cp : n1 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add cg: cp : p : n2 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add cg: cp : p : n32 mscv_inst_co. 100.00% 100 100.00% * Tetrat, g * CVP add_ cg: cp : p : n32 mscv_inst_co. 100.00%	THE SOL OF	The Court of the	10.00		1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	V		100			
Image Class Type Coverage Could two fload Status Included Image Image Image Class Type Coverage Could two fload Status Included Image I	as 🔳 THME OF , SE	. Coverproups						1328311	www.eacata		WWW DADAGE
a. Treton, c. a. Treton, c. b. Treton, c. 100,00% 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% b. Treton, c. a. Treton, c. a. Treton, c. a. Treton, c. 100,00% 100,00% b. Treton, c. a. Treton, c. b. Treton, c. a. Tret	at TYPE and, og	* Name				Class Type	Coverage	Coal	16 of Goul	Satus	Includied
a T TYPE and, rg a T TYPE and, rg <td< td=""><td>H THE DT_CD</td><td>- Aney may paping instration</td><td>10</td><td></td><td></td><td>3930215346</td><td>16.249</td><td></td><td>0206803.5</td><td></td><td>- Contraction</td></td<>	H THE DT_CD	- Aney may paping instration	10			3930215346	16.249		0206803.5		- Contraction
a) Tree and and the set of the set	E I THE and, sg	a TYPE and on	80			risty instruct.	109.009	100	100.00%	-	1
a) TWE HL (2) a) C VP add (2) (2) (52 rscv inst (c). 100.00% 100.100.00% b) T TWE HL (2) a) C VP add (2) (2) (1) (1) rscv inst (c). 100.00% 100.100.00% c) T TWE HL (2) a) C VP add (2) (2) (2) (1) rscv inst (c). 100.00% 100.100.00% c) T TWE HL (2) a) C VP add (2) (2) (2) (1) rscv inst (c). 100.00% 100.100.00% c) T TWE HL (2) a) C VP add (2) (2) (2) (1) rscv inst (c). 100.00% 100.100.00% c) T TWE HL (2) a) C VP add (2) (2) (2) (2) (2) (2) (2) (2) (2) (2)	ALL TYPE OF O	CVP and renter mil				many must co.	100.009	100	1 100 00%		
a Tref un, g a CVP asd_cg: tp_ttl niscv_inst_co. 100.00% 100 100.00% a Tref wil, g a CVP asd_cg: tp_ttl niscv_inst_co. 100.00% 100 100.00% a Tref wil, g a CVP asd_cg: tp_ttl niscv_inst_co. 100.00% 100 100.00% a Tref wil, g a CVP asd_cg: tp_ttl_sign niscv_inst_co. 100.00% 100 100.00% a Tref wil, g a CVP asd_cg: tp_ttl_sign niscv_inst_co. 100.00% 100 100.00% a Tref wil, g a CVP asd_cg: tp_ttl_sign niscv_inst_co. 100.00% 100 100.00% a Tref wil, g a CVP asd_cg: tp_gr_f tarzard niscv_inst_co. 100.00% 100 100.00% a Tref tars, g a CVP asd_cg: tp_gr_f tarzard niscv_inst_co. 100.00% 100 100.00% a Tref tars, g a CVP asd_cg: tp_gr_f sgn_ctoss niscv_inst_co. 100.00% 100 100.00% a Tref tars, g a CROSS asti_cg: cp_ sgn_ctoss niscv_inst_co. 100.00% 100 100.00% a Tref tars, g a Tref tardi_cg	a TYPE IT CO.	U CVP and rm rs rs7				dany inch co.	100.008	100	100.00%		- 2
a Trettive_org a) CVP add_cg::p max max <td< td=""><td>A M THE HEAL OF</td><td>at a CMP and control at</td><td></td><td></td><td></td><td>nery least on</td><td>100.000</td><td>100</td><td>100.000</td><td></td><td></td></td<>	A M THE HEAL OF	at a CMP and control at				nery least on	100.000	100	100.000		
a Tretale, aj a) CVP add cg: tp: m2 kgn macv mat cc. 100 00% 100 00% a) Tretale, aj a) CVP add cg: tp: m2 kgn macv mat cc. 100 00% 100 100 00% a) Tretale, aj a) CVP add cg: tp: m2 kgn macv mat cc. 100 00% 100 100 00% a) Tretale, aj a) CVP add cg: tp: fairzard macv mat cc. 100 00% 100 100 00% a) Tretale, aj a) CROSS adt cg: tp: fairzard macv mat cc. 100 00% 100 100 00% a) Tretale, aj a) CROSS adt cg: tp: sign cross macv imat cc. 100 00% 100 100 00% a) Tretale, aj a) CROSS adt cg: tp: sign cross macv imat cc. 100 100 00% a) Tretale, aj a) Tretale, aj a) CROSS adt cg: tp: sign cross macv imat cc. 100 100 00% a) Tretale, aj a) Tretale, aj macv imat cc. 100 100 00% av	a THE HALL OF	Chill Chill and Chilling Chill				user_mag_ov.	100.007	100	100 000	-	
Image: No. 100 Image:	the second second	T CAN 930 cft th ur edu				maca wear cor	100.004	100	100.004	-	¥.
all Trettisking all CVP add opting to group t	TYPE DOL AN	CVP and cg.cp_rs2_egn				nace must co.	100.004	100	100.004		V
S at Tertison, rg S at CVP add_cg:rg, gpr_harzard riscv_instr_ct. 100.00% 100 100.00% S at Tertison, rg S at CQ:rg, ggr_harzard riscv_instr_ct. 100.00% 100 100.00% S at Tertison, rg S at Tertison, rg S at Tertison, rg S at Tertison, rg 100.00% 100 100.00% S at Tertison, rg S at Tertison, rg S at Tertison, rg S at Tertison, rg 100.00% 100 100.00% S at Tertison, rg S at Tertison, rg S at Tertison, rg S at Tertison, rg 100.00% 100 100.00%	a TYPE bile op	al de la construction de la c				riscy_insit_co.	100.009	100	100.00%	1	
a) at Teres, all a) at CROSS add_cg:cp_sign_ctross nscv_instr_co. 100.00% 100 100.00% a) at Teres, all a) at Teres, all a) at Teres, all nscv_instr_co. 100.00% 100 100.00% a) at Teres, all a) at Teres, all nscv_instr_co. 100.00% 100 100.00%	S J THE LIGHT_TO	🗉 🖬 CVP add_cg:cp_gpr_harzard				risty_instr_co	100.00%	100	100.00%		
I THE NUT I I I THE SUB_CD I I I THE SUB_CD I I I I I I I I I I I I I I I I I	al TYPE D. DB	🖃 🔟 CROSS and ogrop sign cross				risev instr co.,	100.009	100	100.00%	0	
a) 🖬 TYPE addi cg nscv insiz co. 99.55% 100 99.55%	a Title or	-i d TYPE sub co				riscy insit co.	100.004	100	100.004	1	
	1.	TYPE and CU				riscy insit co.	99.599	100	99 554		1
1 TYPE W CE		N TYPE IN CO				risov litest co.	90.000	100	1000		
A TYPE autor on they instrong souther 100 90.000		al of TYPE mant on				many insit on		100	90.07	-	1
2 TYPE on an interview of 100,00% 100,00%		a) d Type sea on				dary least on-	100.009	100	100.004		
TYPE of 2m		TYPE of AN				ripey inph ch.	100.008	100	100,008	-	
2 We at an		al al TVIE of co				risey insit on	100.005	100	100.000	_	
		ar a mit as ug				HOLE HIRE CO.		100	10110010	-	Y
Starting under under 100 occure		TI TI ALLER BURL CD				uses unst cor	0.004	100	0.004		V
a THE SH og OSC 100.00% 100 100.00%		A TYPE SIN_CG				ISCY_HIST_CO.	180.00%	106	100.00%	1	1
(a) → TVPE stl_cg nscv_inst_co 100.00% 100 100.00% √		(s) TYPE sti_cg				necv_inst_co.	109.004	100	100.00%	0	- C
🛓 📶 TYPE kor_tig riskt_co 100.00% 100 100.00%		A) I TYPE KOF_EG				risca insit co.	100.00%	100	100.00%	5	
a) 100.00% 100 100.00% 100 100.00% 100 100.00%		🔊 🗾 TYPE or .cg				risev instr co.	100.004	100	100.004	-	
		- TYPE and cd				riscy instricts.	100.00%	100	100.00%	X	
w # TYPE yon cp nscv inst co., 100.00% 100 100.00%		H I TYPE XON CO				risty inst co.	109.004	100	100.004	2	
1 d TYPE of cn racy instruction 100 00% 100 100 00%		+i at TYPE of cn				riscy inst co.	105.00%	100	100.00%	k.	



Coverage reports from Siemens/Mentor Questa



© Imperas Software Ltd.

Metrics: includes top level overview dashboard







Allows management overview of status of verification

Page 125

© Imperas Software Ltd.

Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Functional Coverage
 - Instruction Set Simulators



Agenda



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
 - Compliance Tests and other Test Suites
 - Instruction Stream Generators
 - Functional Coverage
 - Instruction Set Simulators
 - Free riscvOVPsim (github)
 - Free riscvOVPsimPlus (ovpworld.org)
 - Commercial M*SIM (imperas.com)
 - Commercial M*SDK (imperas.com)
 - Commercial ImperasDV (imperas.com)



<u>.</u>		Gilliak - risev-avgsim/inpe	ras-riscv fests - Modila Firefee	an (1997)		- 65	() <u>e</u> 1		
O GitHub - riscv-ovpsim/ ← → C D	X O Imperas-riscv-tests/riscv x O B https://github.com//iscv-	mperas-riscv-tests/risc- × N riscvOVPsir	mPlus_User_E 🛪 🎂 Welcome Page Open \	◎×+ (305.) 回☆	± m	. 🗆 🔹	ť	=)
O Code 🔿 lisues	i 🗅 Pull requests 💿 Actions 🔟 Proje	da © Secunty ⊠ Inaighta						1	
	P main - P I branch 00	lege	Go to file 🚽 Oode +	About No description, website, or topics					
	Imperas Merge pul request #7	from riscs-oxpain/updateOVPsim 🚞	reessos: 21 days ago 🕤 17 commite	provided.					
	🖿 ristv-ovpsim-plus	update risco-oxpaini version	21 days agu	11 Feadme					
	risce-expérn	update race expelin version	21 dags ago						
	🖿 Histy-targat	update mice-oxpain version	21 days age	Releases					
	tiacy-leaf-env	update nius-bypaini vension	21 days agri	No. reteaure poolaries					
	nary-best-suite	update nazy-ovpeint vension	321 daya aga	Packages					
	ChangeLog.md	update risov-oxpains vention	21 days agu	We uncharger published					
	C) Mahatla	aditificnal information: update MakoSia and	I FIEADME last month						
	C README.md	type fixed	20 days ago	Contributors 2					
	README.md			meras Imperas Github Account					
	Importe DISC	V ricevOVDeim refere	noo cimulator and	😫 risev-ovpsim					
	architectural	-v riscvovesini relere	ince simulator and						
	architectural	andation tests		Languages					
	riscvOVPsim is released by simulators for advanced proc compiling, no fiddling, and no	imperas based on their 12+ years of developing assor architectures. It is a free closed source si external dependencies, II just works.	C DR.2% Assembly 48.8% C++ 3.0% Objective-C 0.3% Maileftle 1.2%						
	In the RISC-V world, Imperas quality RTL working and sign to verify their RTL in a hardwo the technology to use to obtain	Cother 0.3%							
	The following is a list of some reference:	of the companies and organizations that rely o	in Imperas simulators as their RISC-V						
	We use optional third-party enab	rics cookies to understand how you use GitHub.com	so we can build better products. Learn more.	Accept Feged					

Imperas Tools for Embedded Software Development, Debug & Test





Imperas



		importan 1950 M Saladherra Impiriety - Emilian	1	at 2 water (Eq. 172-35-3-142) - Arryand Parya (He			7 - D X
Ĩ	🖮 Imperas RISC-V Solutio :: 🗴	+	File Edit View Search Terminal Help	Andres NDS FrankTIDSt			
Rabhun	ା ← → ୯ ଜା	🔏 https://www.imperas.com/impera	51000000000000000000000000000000000000	wides_mas_recentors []			
	🖌 ther Login 🖉 Contact Us	SEARCH!					
	(*100055550) * V * 5105565	aconstr.					
The Syst		Home Products ~					
	Impera	S					
-							
4	Revolutionizing	Embedded Software					
Home							
	Imperas RIS	SC-V Solutions					
VCS man							
			Terrsinal		0. B 18 C		
File Edit	View Search Terminal Help nt shell						
cd -/simo	nd/force-riscv/force-riscv	/; clear ; ls -ltr ; ls -ltr	/utils/regression/master_run.py				1
tail -50	nutput/regression/regressi	on_summary.log ; echo ""; ech	a **				
is -itr o	utput/regression/*/*/*.ELF						
<pre>cd ~/simo ./GEN.sh;</pre>	nd/force-riscv; clear; ./C ls -ltr	LEAN.sh; source setup.sh; is	ltr; cat ./GEN.sh				
cat ./RUN ./RUN.sh	.sh						
# differe	nt shell						
cd /home/s	simond/riscv/valtrix; iset	up; clear; ls -ltr tests/*.el	r				
clear+ is	etun: cd /scratch/simond/d	enn: ls -ltr					
cd /scrat	ch/simond/demo/riscv Andes	N25_FreeRTOS; ls -ltr; firef	ox Andes N25 NX25 FreeRT05.jpg &				
cd /scrat	ch/simond/demo/riscv_5iFiv	eFU540_Linux; ls -ltr ; firef	ox Linux_SiFiveFU540.jpg &				
cd /scrat	eFU540.sh # root sitive ch/simond/demo//Hetero_ARM	_RISCV_NeuralNetwork; ls -ltr	; firefox 0.NeuralNetwork_Platform.jpg 1.	NeuralNetwork_Alexnet.jpg &			
cd /scrat	ch/simond/demo/Hetero_ARM_	RISCV_NeuralNetwork/harness;	/RUN_Hetero_ARM_RISCV_NeuralNetwork_Alexn	ot.sh			
*demoread	me.txt" 66L, 2975C written			63,3	95%		
					1	-	
				Application	ns 🕒 Imperas 95524 🛄 (Termi	nal 📴 Termatal 🔡	e2-usergip-17

ISS: Summary



- Only mentioned Imperas simulators (as we use daily, our customers rely on them , and we understand their quality)
 - There are are others...
 - There are many open source grad. student project simulators... go search github...
 - RISC-V has a formal model under development 'sail'
 - There is also spike from Berkeley for architectural exploration
 - And there are full system software emulators like gemu
- Free: github: riscvOVPsim.exe
 - Model selection configuration
 - Signature, logs, coverage
 - Includes rv32I tests
 - Useful for compliance tests
- Free: ovpworld.org (needs registration): riscvOVPsimPlus.exe
 - As riscvOVPsimPlus
 - trace, debug
 - Useful for post-sim-trace-compare, e.g. in Google riscv-dv
- Commercial from Imperas the industry leader in processor based simulation solutions
 - Full range of ISS, virtual platforms, full system emulation, fixed & extendable platforms
 - 50 reference platforms, 250+ peripheral components, 300+ processor models







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Other components of RISC-V CPU DV environments
- Summary





- Congratulations... on getting to the end of this tutorial
- if you got this far and still have some energy...
 - send us an email (<u>info@imperas.com</u>) with
 - Subject: imperasdv at dvcon22 tutorial
 - With: comments on this tutorial
 - and we will send the first 50 of you one of our ImperasDV drinking mugs



We covered...



- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
- Components of RISC-V CPU DV environment







- Brief Introduction to RISC-V
- RISC-V CPU HW DV approaches
 - #0 "hello world" test
 - #1 self checking tests (e.g. Berkeley torture tests pre2018)
 - #2 Post simulation trace log file compare (e.g. Google riscv-dv 2019)
 - #3 sync-lock-step-compare (e.g. CV32E40P in OpenHW 1H 2020)
 - #4 async-lock-step-compare (e.g. CV32E40P in OpenHW 2H H2020)
 - #5 test bench use of standards (RVVI) (e.g. CV32E40X/S in OpenHW 2021)
 - #6 using standards based DV products and VIP (ImperasDV)
 - And had a 35 minute walk through



We covered (3)

- Components of RISC-V CPU DV environment
 - Compliance Tests and other Test Suites
 - RISC-V architectural test suites
 - Imperas architectural test suites
 - Bremen fuzz testing
 - Instruction Stream Generators
 - Google riscv-v
 - OpenHW force-riscv
 - Valtrix STING
 - Functional Coverage
 - Imperas build-in instruction coverage
 - SystemVerilog covergroups and coverpoints
 - Instruction Set Simulators & tools
 - Free riscvOVPsim (github)
 - Free riscvOVPsimPlus (ovpworld.org)
 - Commercial M*SIM (imperas.com)
 - Commercial M*SDK (imperas.com)
 - Commercial ImperasDV (imperas.com)





What we did not talk about...



- SystemVerilog encapsulation of Imperas models (yes we do that)
- Using ISS with RTL emulators
 - Including hybrid simulation (yes we do that)
- Formal tools
 - (no we don't do these...)







- The open standard ISA of RISC-V offers many design freedoms
 - Many standard extensions and configuration options plus custom instructions
- The key verification requirements are to detect discrepancies with efficient debug
- The open standard RVVI offers a framework for verification reuse with support for both open-source and commercial tools
 - RISC-V Verification Interface
 - <u>https://github.com/riscv-verification/RVVI</u>
- Lockstep / Compare is by far the best and most efficient approach (industry 'gold standard')
 - https://www.imperas.com/imperasdv



Thank You!



info@imperas.com <u>www.imperas.com</u> <u>www.imperas.com/ImperasDV</u>

www.OVPworld.org





Questions?



Page 141

© Imperas Software Ltd.

Q1 2022