Architectural Test Task Group Call – Minutes

Thur, 29Apr2021 8am Pacific → Daylight ← Time

See slide 6 for agenda

Antitrust Policy Notice

RISC-V International meetings involve participation by industry competitors, and it is the intention of RISC-V International to conduct all its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

Examples of types of actions that are prohibited at RISC-V International meetings and in connection with RISC-V International activities are described in the RISC-V International Regulations Article 7 available here: https://riscv.org/regulations/

If you have questions about these matters, please contact your company counsel.

RISC-V International Code of Conduct

RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

We are a transparent, collaborative community where all are welcomed, and all members are encouraged to participate.

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone.

https://riscv.org/risc-v-international-community-code-of-conduct/

Adminstrative Pointers

- Chair – Allen Baum allen.baum@esperantotech.com Co-chair – Bill McSpadden bill.mcspadden@seagate.com
- SIG Email sig-arch-test@lists.riscv.org
  - Notetakers: please send emails to allen.baum@esperantotech.com
- Meetings -Bi-monthly at 8am Pacific time on 2nd/4th Wednesdays.
  - See https://docs.google.com/spreadsheets/d/1L15_gHl5b2ApkcHVtpZyl4s_A7sgSrNN zoom link
- Documents, calendar, roster, etc. in
  - https://sites.google.com/a/riscv.org/risc-v-staff/home/tech-groups-cal
    https://drive.google.com/drive/folders/1DemKMAD3D0Ka1MeESRoVCJipSrwiUlEs
    (lifecycle in "policies/supporting docs" folder, gaps in "planning" folder, compliance specific in "compliance folder")
- Git repositories

←docs	riscv	→ tools
https://github.com/riscv/riscv-compliance/tree/master/doc/	tests	https://github.com/riscv/riscv-arch-test/
https://riscof.readthedocs.io/en/latest/index.html	riscof	https://gitlab.com/incoresemi/riscof/
https://riscv-isac.readthedocs.io/	ISA coverage	https://github.com/riscv_isac
https://riscv-ctg.readthedocs.io/	Test Gen.	https://github.com/riscv_ctg
https://github.com/riscv/riscv-config/tree/master/docs	YAML, WARL config	https://github.com/riscv/riscv-config/
https://github.com/rems-project/sail-riscv/tree/master/doc	Sail formal model	https://github.com/rems-project/sail-riscv/
https://github.com/riscv/groups/tree/main/Architecture-Test/minutes/	minutes	

- JIRA: https://jira.riscv.org/projects/CSC/issues/CSC-1?filter=allopenissues
- Sail annotated ISA spec: in https://github.com/rems-project/riscv-isa-manual/blob/sail/

README.SAIL	←how to annotate	annotated unpriv spec→	release/riscv-spec-sail-draft.pdf
release/riscv-spec-sail-draft.pdf	← annotated source	annotated priv spec→	release/riscv-privileged-sail-draft.pdf

  - https://us02web.zoom.us/rec/share/-XIYazzhIBbQoiZdarCfebdjxjDWiVhf-LxnuVrliN4Bc30yf17ztKkKDU4Og54b.fArPPqnuR-NiXpQU Tutorial
    Access Passcode: tHAR#5$V

SIG Charter

The Architectural Compatibility Test SIG is an umbrella group that will provide guidance, strategy and oversight for the development of tests used to help find incompatibilities with the RISC-V Architecture as a step in the Architectural Compatibility self-certification process

The group will:

- Guide Development of:
  - Architectural tests for RISC-V implementations covering ratified and in-flight specifications for
    - Architectural versions, standard extensions, and implementation options.
  - Tools and infrastructure to help identify architectural incompatibilities in implementations
- Work with LSM and Chairs for resources to get the above work done.
- Mentor or arrange for mentoring for the resources to get the above work done

Meeting Agenda

0. **Looking for more admins, maintainers for riscv-arch-test git repo !!**

I. Updates, Status, Progress:
   - I. Still looking for CI/Testing chair, Simulator SIG chair, ISA infrastructure HC chair
   - II. Many github issues closed (more will close when Riscof is deployed, more likely resolved but needs feedback)
   - III. ACT policy very nearly complete, waiver policies still under debate
   - IV. Chairs: B,V,Zk and Priv1.12 are top priorities
   - V. New DOD requirement: extensions with non-determinism (external events, timing, concurrency) need to define how those will be test in planning stage

II. Next steps and Ongoing maintenance
   1. Quick: test spec issues 188, 189, 190: spec clarifications & related fixes for K, fixes for overlapping extensions, 16b sig boundary bug,
   2. Discussion: Migration to Framework v.3.0 (riscof)
      a) Docker image for riscof proposal
   3. Discussion: testing methodology for SIGs/TGs needing external stimulus/observability "ports".
   4. Discussion: other steps for Migration to Framework v.3.0 (riscof). (blocking items):
      a) (Sail/Spike model updates, pipecleaning, N people have run it, testing all the "fixed in riscof" issues
      b) Review Pipecleaner tests:What do we need to do to exercise capabilities for Priv Mode tests
   5. Define report standard (see ACT policy doc here:
      https://docs.google.com/document/d/1ZKSLQ5HPT3E_CqTVOQlBPs7qJ9v1mpnMDXHhtOQJFcU
   6. Close github issues as a result of repo v2.1
   7. Maintenance updates to V2 to enable future tests
      a) update RVTEST_SIGUPD to keep automatically adjust base/hidden offset when offset>2K,
      b) Enable use Sail model results as the assertion value
      c) add assertion macros for FP, DP, Vreg to arch_test.h and test_format spec
      d) add trap handlers for S, VS modes
   8. Tests for non-deterministic result (see attached discussion in email)

What is Docker?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Why Docker?

- It gives RVI the means to standardize on the toolchains and other tools needed to run the Architectural Test Suite.
- It provides a consistent well-known environment in which to run.
- As needs change we can update the Docker Images that are provided. Updating toolchains, underlying OS, etc. as needed.
- Updating on the “customer” end for a new Docker Image is easier. Just do a a docker pull for the latest release. They do not have to worry about toolchains, other tools, etc.
- We do not need to worry about which toolchain/tools someone is using, just what version of the Docker Image they used.

Issues/Questions running in Docker

- How do you run the SIM under Docker?
  - unknown
- Do we package the Arch Tests in the Docker Image?
  - unknown
- Other concerns?
  - Supply chain attack?

Discussion

Passdowns, Status: see agenda (slide 6)

NextSteps
Quick: test spec issues 188, 189, 190
Issue#188 spec is missing explicit wording that test_case id be unique. Clarify in spec, update all K_unratified tests to make them unique. Also remove redundant test cases.
Issue# 189 instructions can now be part of 2 specs; format doesn't take that into account. Proposal is to modify RVTEST_ISA macro to allow multiple extension references
Issue# 190 Obsolete 16B sig boundary legacy: spec doesn't mandate initializing pad words. We can remove 16B length restriction, or modify all tests Prefer the former

Migration to Framework v.3.0 (riscof) **Docker image for riscof proposal**
Imperas - How do we use Docker in an existing RTL environment?
  Inspire - Can cross mount from docker to host or host to docker. .elf files can be seen
Imperas - Have to run verilog and arch tests dynamically. Dynamic signature generation
  Inspire - NO. Docker gives us control over what toolchains or OS the user needs.
Bristol - Naive question: what problem is it solving? Many tests already taped out with current method. Docker encourages building monoliths. <many people have same questions >
SimonFrasier - What are the dependencies?
Co-Chair - Example: python version
Incore - merge requests allow you to merge with a consistent set of tools
Chair - These are the things you are required to use.
Inspire - SAIL and Spike would also be in the image <note: Sail builds the Sail Cmodel
< in general: Docker solves a bunch of problems with CI >
< discussion of re-creation of Docker image is part of CI. can do? >
Incore - question for CM: does the toolchain group use Docker image?
  Ventana - sort of. Toolchain changes are frequent; we want people to work mainline if possible.
Inspire - < gave use model >
  Ventana - will present to toolchain group at next meeting
    <slide 9: Docker Questions>
Inspire - How do you run the SIM under Docker? (**Imperas** question again)
    Everything in the Docker image is publicly available in github(s)
Chair - don't understand the problem being solved. it still exists.
Imperas - the discussion Docker is about packaging tools. Do you *have* to use this tool version
Chair –Vendors can run any version, but if tests fail with a different version, on their own.
  ACT will document the versions used for CI regressions; tests reports document the versions used.

SimonFrasier - does SAIL do it this way? does it need to be rebuilt?
Chair - yes. SAIL models need to be built w/ YAML configuration when tests start.
Inspire - < discussion about how often the Docker image changes >
SimonFrasier - question: how does my YAML configure stuff in the Docker image?
Incore – Sail C model is built by tools packaged in the image. **Inspire** described it - the main purpose of Docker is as a reference for building SAIL, SPIKE, and toolchain.
Inspire - you can use the Docker image however you want. but this is how the sig-arch-tests group is building/running arch tests.
Imperas - how does work in the future? it's a dynamic process in the future.
  it needs to be easy for the user
  Inspire - it's a *reference*
  Ventana - it's an automated README. it's better than a text README
  Imperas - how is this better than checking a github build list
  Incore - it saves compile time and finding all the dependencies
  Incore2 - you can treat the Docker image as a remote machine.
Chair - my initial impression was that this was a way to deploy a toolset.
    Seems like this is not the case. See "Test Report Requirements" slide-
    We need to be able to specify what versions of tools were used in the test report.
    Test report should extract that automatically and print it in the test report>
  SimonFrasier - we will always be behind the head, using patches.
Inspire – Don't need to have absolute latest. In my project,
  I'm not updating my toolchain every week.
  See https://github.com/InspireSemi/dev_riscv
  ACT need tp be doing regression/
  Can we put ports into image to cause external host to run and get results back
    <likely not – highly dependent on sim tools>
  Or: use it to generate reference signature? Use this as a reference image and provide it as a baseline
  Only need to prove Ubuntu, Debian, Redhat and Centos
  Docker image, use a VM Just a way to be transparent - maybe just ubuntu - otherwise just a docker file i dockerhub debian and ubuntu, one and done.
  Sail is very hard to install, needs specific versions of tools

See: decisions on next slide

Decisions & Action Items

Decisions

Per the discussion in the last Architectural Test Meeting

1. A Docker image will be created based on Ubuntu for use in CI of the Architectural Test Suite. It will contain the cross-compiler, spike and sail for RiscV
2. RVI will publish this image on docker hub under the RVI name
3. 4 dockerfiles will also be created and published in an RVI github repo based on the following Oses
   a. Ubuntu
   b. CentOS
   c. Debian
   d. RedHat/Fedora
4. Both the docker hub and github repos will be for informational purposes only. We will accept input from the community, but will not be providing any support. They are provided as a resource for the RiscV community only.

Outstanding Action Items

NEW

Old

Chair : document target process for removing target environment files from riscv-compliance repo into a target repo and contact all model maintainers to inform them of the process and timeline. <restarted>

Chair: more brainstorming on handling nondeterminism, concurrency <discussion started on retrofitting riscof for concurrency>

Chair: need clarity on tool source/version report.

Inspire: add support for QEMU target <?>

Chair: get SAIL repo moved into a riscv repo <done, but for viewing only>

QC: extract bits of FAQ as guidelines for test writing <?>
Incore: Try YAML version of SAIL to see if it works <ongoing>

SH: write up coverage taxonomy

Pull/Issue Status

Issue#	Date	submitter	title	status	comments
#4	03-Jul-2018	Kasanovic	Section 2.3 Target Environment	Fixed in riscof	Will be closed in V3
#22	24-Nov-18	brouhaha	I-MISALIGN_LDST-01 assumes misaligned data access will trap	^	HW misalign support not configurable
#40	4-Feb-19	debs-sifive	Usage of tohost/fromhost should be removed	|	now
#142	17-Nov-20	subhajit26	Not able to run compliance test for rv32E device and RV32E ISA	RV32E only	Not RV32EC or RV32EM
#146-9	01-Dec-20	Imperas	Test I EBREAK,ECALL, MISALIGN_JMP/LDST, OpenHW	v	HW misalign support not configurable
#107	22-Apr-20	jeremybennett	Clang/LLVM doesn't support all CSRs used in compliance test suite	under discussion	-can we add an alias?
#115	06-jun-20	adchd	How to support on-board execution?	under discussion	
pull#129	31-jul-20	nmeum	sail-riscv-ocaml: Disable RVC extension on all devices not using it	In process	Who can review this?
pull#184	15-apr-21	dansmathers	Updating http reference for constr	In process	Approved, needs merge
#116	06-jun-20	simon5656	loss of 64bit test infrastucture	under discussion	Will close in a week unless objections
#119	17-jun-20	allenjbaum	Missing RV32i/RV64i test: Fence	Test has been written	Close when RFQ test is merged
#188	26-Apr-21	neelgala	Updates required in K_unratified tests to be compatible with current RISCOF		
#189	26-Apr-21	neelgala	Proposal to enhance the RVTEST_ISA macro		
#190	26-Apr-21	neelgala	The 16-byte signature boundary issue		

JIRA Status

Issue#	Date	submitter	title	status	comments
IT-1	27Aug/20	Allen Baum	Need to modify the description of compliance in https://riscv.org/technical/specifications/	done	
IT-4	01/Sep/20	Allen Baum	Add Jira link to TG home pages	done	
CSC-1	20/Aug/20	Ken Dockser	Come up with names for the tests suites that we are creating		1st step done
CSC-2	20/Aug/20	Ken Dockser	Produce concise text to explain the Architecture Tests intent and Limits	done	Written, needs pull req
CSC-3	20/Aug/20	Ken Dockser	Come up with an internal goal for what we wish to accomplish with the Architectural Tests		Not written
CSC-4	20/Aug/20	Ken Dockser	Develop a roadmap for all the different categories of test suites that will need to be created		Not written
CSC-5	20/Aug/20	Ken Dockser	Develop a roadmap for releases of single-instruction Architecture Tests		Not written
CSC-6	20/Aug/20	Ken Dockser	Develop a reference RTL test fixture that can stimulate and check the CPU under test		Needs more discussion

BACKUP

Test Acceptance Criteria – draft3

Tests must:

- conform to current standard of test spec (macros, labels, directory structure)
- use only files that are part of the defined support files in the repository
- run in framework
- run in SAIL and not fail any tests
- generate a valid signature using SAIL (that can be saved & compared with DUT/sim)
- Report that test results propagate to signature
- have a clear configuration - i.e. which ISA extension it can be used with
- improve coverage (compared to existing tests) as measured and reported by ISAC
- use only standard instructions (fixed size per architecture LI, LA allowed)
- be commented in test_case header

Tests that are otherwise accepted, but depend on tools or simulators that have not be upstreamed must be put into a <Ext-Name_beta>/ directory instead of <Ext-Name>/

Test Report Requirements

- Architecture test version policy (mechanism TBD by ACT group)
  - Architectural test reports are generated by the framework and include:
    - The YAML description of the features and options implemented (for v3 of the framework)
      - This will include the ISA string, and options that claim to be implemented
      - The vendor and implementation IDs that the part will report
    - Test pass/fail reports
    - Version numbers at the point of extension ratification of
      - Toolchain used to compile tests,
      - reference model used,
      - Architecture Compatibility Test (ACT) suite, and riscv-config tool (for v3 of the framework)
- Vendors who self-certify generate pull request into arch-test-reports github repo
  - Report is stored in subdirectory <vendor_name> with filename <instanceID>-<date>.html
  - Additional file that describes the buildsteps used with filename <instanceID>-<date>.buildsteps
    - List source and version of tools, simulator, etc
      - same items as listed in ACT README that describes tools and versions ACTs were tested with
    - Failures found using different versions is not a valid reason for waiver unless reproducible with standard
- Each release version of ACT will document, in the repository doc/VERSION.md file, the versions of toolchain utilities used for the version of the tests in that repository
  - detail the steps to run, source of tools, and there version numbers

Framework Requirements – first cut

The framework must:

- Use the TestFormat spec and macros described therein
  - (which must work - including assertions)
- Choose test cases according to equations that reference the YAML configuration
- Define macro variables that can be used inside tests based on the YAML configuration
- Include the compliance trap handler, & handle its (separate) signature area
- Load, initialize, and run selected tests between two selected models, extract the signatures, compare results, and write out a report file
- Exist in a riscv github repo, with a more than one maintainer.
- Be easy to get running, e.g.:
  - run under a variety of OSes with the minimum number of distro specific tools.
  - Not require sudo privileges
- Have the ability to measure and report coverage
  - Coverage specification is a separate file
  - Could be a separate app

Non-determinism in Architectural Tests

The RV architecture defines optional and model/µarch defined behavior.
This implication: there are tests that have multiple correct answers. E.g.:

- Misaligned accesses: can be handled in HW, by "invisible" traps w/ either misaligned or illegal access causes, and do it differently for the same op accessing the same address at different times (e.g. if the 2nd half was in the TLB or not)
- Unordered Vector Reduce ops: (different results depending on ordering & cancellation)
- Tests involving concurrency will have different results depending on microarchitectural state, speculation, or timing between concurrent threads (e.g. modifying page table entry without fencing)

From the point of view of ACTs, there are 2 (& sometimes more) legal answers. The golden model only generates one. Possible mechanisms to test include:

- Modify (if necessary) & configure reference model to generate each legal result, run it with each config, & accept either result from the DUT (e.g. misalign or un-fenced PTE modification)
- Provide specific handlers for optional traps
- Use self-testing tests(compare with list or range of allowed outcomes from litmus tests)
- Avoid tests that can generate non-deterministic results
- Ultimately: develop new frameworks that can handle concurrency along with reference models that can generate all legal outcomes
- It is the responsibility of the TG that develops an extension to develop the strategy for testing features and extensions that can have nondeterministic results

Architectural Test Rationale – Intent and Limits

RISC-V Architectural Tests are an evolving set of tests that are created to help ensure that SW written for a given RISC-V Profile will run on all implementations that comply with that profile.

These tests also help ensure that the implementer has both understood and implemented the specification.

The RISC-V Architectural Tests test suite is a minimal filter. Passing the tests and having the results approved by RISC-V International is a prerequisite to licensing the RISC-V trademarks in connection with the design.

Passing the RISC-V Architectural Tests does ***not*** mean that the design complies with the RISC-V Architecture. These are only a basic set of tests.

The RISC-V Architectural Tests are ***not*** a substitute for rigorous design verification; it is the responsibility of the implementer to deploy extensive testing.

To be added to the riscv/riscv-compliance/doc/ directory as "RISC-V Architectural Test Rationale" satisfying Jira CSC-2

TGs under the SIG

- IF you’re creating work product, you should be a TG
- If changing requirements, plans ABIs, etc
  - Test plan==SOW
- The Architectural Testing Task Group will define and maintain specifications for
  - test formats
  - test-benches and frameworks needed for
    - privilege testing privilege testing,
    - Concurrency/ Memory model testing
    - Asynchronous event testing (interrupts)
    - Nondeterministic tests
  - ISA test coverage goals
  - test tools (e.g. coverage, generators)
- The Architectural Testing Task Group will maintain the appropriate GitHub:
  - tests for the individual ISA extensions
  - issues related to the tests
  - the operation and issues related to the framework
- The Architectural Testing Task Group will
  - work with the different privilege and un-privilege ISA extension Task Groups
    - to help them write test plans/specs for the ISA tests
    - to help them work with the sub-contractors (IITMadras, RIOS, CAS, etc) to deliver the tests
  - assess quality of delivered tests and be maintainer for the test GitHub

Conventions

- We don’t solve problems or detailed topics in most meetings unless specified in the agenda because we don’t often have enough time to do so and it is more efficient to do so offline and/or in email. We identify items and send folks off to do the work and come back with solutions or proposals.
- If some policy, org, extension, etc. can be doing things in a better way, help us make it better. Do not change or not abide by the item unillaterly. Instead let's work together to make it better.
- Please conduct meetings that accommodates the virtual and broad geographical nature of our teams. This includes meeting times, repeating questions before you answer, at appropriate times polling attendees, guide people to interact in a way that has attendees taking turns speaking, ...