Architectural Test Task Group Call – Minutes

Thur, 08Jul2021 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/

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

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 Thursdays.
  - 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, arch-test specific in “information->content->arch-test”)
- 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-admin-docs/architecture-test/	minutes, charter	

- 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

(next) Meeting Agenda

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

I. Updates, Status, Progress:

  I. F/D tests will be done soon
  II. BitManip is starting; when done, K tests will also be complete
  III. RIOS labs working on Vector tests & Sail Support
  IV. PLCT working to add CSR configuration to SAIL
  V. Zce development Partner identified
  VI. A Sail maintainer has been identified

II. Next steps and Ongoing maintenance

  1. Discussion: other steps for Migration to Framework v.3.0 (riscof). (blocking items):
     a) Reorg of arch-test repo <done>
     b) Coverage reports: dev/vendor, archive in repo or not?
     c) (Sail/Spike model updates, pipecleaning, N people have run it, testing all the “fixed in riscof” issues
     d) Review Pipecleaner tests:What do we need to do to exercise capabilities for Priv Mode tests
  2. Discussion: testing methodology for SIGs/TGs needing external stimulus/observability "ports".
  3. 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 of Sail model results as the assertion value
     c) Convert assertions to be out-of-line
     d) add assertion macros for FP, DP, Vreg to arch_test.h and test_format spec
     e) add trap handlers for S, VS modes
  4. Tests for non-deterministic result (see attached discussion in email)
     a) Provide a reference RTL test fixture (as opposed to SW functional model). See. JIRA CSC-6
     b) Define hooks for concurrency tests
  5. Specific Arch-Teest Policy/Process Gaps:

Discussion

Status: F/D tests nearly done. SAIL bug found: No F/D tests supported on RV32I
BitManip is starting up (IITM)
K has some bitmanip instructions; the tests for those are complete, except entropy source ops (for which only CSR access is testable). See:
https://github.com/riscv/riscv-arch-test/blob/master/riscv-test-suite/rv64i_m/K_unratified/Makefrag for the list of all K_unratified tests
RIOS labs is writing Vector tests and Sail support
Zce tests will be done by Nambi
Priv tests are waived; Proof of concept will be running an OS.

Riscof discussion (Neel Gala and S Pawan Kumar, Incore Semi):
Test repo will be split into test-suite, framework **[AI]**

Chair: why split off the plugins into a separate repo?
(plugins are examples of the model specific code implementors must provide)
people don't want to clone extra repos. These sound more like docs.
Incore: rationale is if Spike or SAIL interfaces change, plugins will be out of date.
Incore: we can keep them in the test repo if desired. **[AI]**
NOTE: RISCOF & riscv-config are installed with pip install, not by cloning
Chair: plugins currently only functional simulators - do we need an RTL example?
??: Remove tests stats - its an artifact.
Chair: test stats are reports generated when tests are developed, just as test reports are generated when tests are run. Reports are pushed in a test report repo (see ACT requirements policy). We should do something similar when tests are accepted into the repo – but where?

PMgr : the riscv/ GitHub organization is being split into additional orgs to simplify things. At this time, the Arch Tests will move to riscv-software per this proposal:
https://drive.google.com/drive/u/1/folders/1y8nq63-8EDVyRRN4pQbAbRcWw6o-Kjy5

Incore: there are a total of 6 repos that have an effect on arch-test:
tools: (ctg, isac, riscv-config, riscof),, tests, plugins? (see above)
note that only developers need to look at ctg, or isa, and the framework+riscv-config are installed with pip.

Note: this doesn't include the DUT, reference simulator, and the toolchain, which are always needed

For the transition: all this is in a branch. **[AI]**
Put a disclaimer on all READMEs to start using the branch**[AI]**

RVTEST_ISA macro will change in various tests because they're incompatible with riscof (minor formatting issue) **[AI]**

Incore : how do we handle added tests during transition? Concern is people not moving to Riscof

Chair: Concerned mostly about ops/tests shared between extensions.
They have to be treated differently: duplicated subdirectories now, testcase conditions afterwards.

Documentation

Riscof docs are here:

https://riscof.readthedocs.io/
https://gitlab.com/incoresemi/riscof/-/tree/master/docs/source
These use sphinx to create documentation - fairly language dependent (sphinx is python specific?)
example: https://riscv-isac.readthedocs.io/en/stable/code.html#module-riscv_isac.fp_dataset

Chair : What format are docs in? And do they have to be put in certain directories?
Incore showed the docs/source and how things get rendered.
python code can be written so that documentation can be automatically derived.
Sphinx does this conversion. Strong for python code.
Do we need to do docs in asciidoc?
PMgr: How far do we take documentation to asciidoc? I'll come back with an answer. **[AI]**

CoChair: should we standardize on in-line documentation tool?
PMgr : someone should broach this with SW HC **[AI]**

Decisions & Action Items

Decisions

- Riscof will be added as an in an arch-test branch ahead of the switch over (all existing tests will be backwards compatible)

- separately, a branch of the arch-test repo will be created with all the directory changes needed for the switchover to riscof.

- all READMEs will be updated to indicate when the switchover will happen, and point to the riscof branch and encourage/warn users to start using it

- In (very) roughly 3 months, we will merge in branch.

Outstanding Action Items

- create framework riscv repo <done!>
- Add branch for reconfigured riscof arch-test repo & migrate riscof to new framework repo <Neel>
- Fix uses of RVTEST_ISA macro in various tests (formatting incompatibility with riscof and update spec <Neel>
- Contact SW HC & DOC SIG to determine an inline comment->doc tool flow, and determine if docs (as opposed to ISA specs) must be .adoc, or could be .pdf or .hmtl <Allen, Jeff-in progress>
- Settle where plugins will go, and the migration process <Allen, Neel, ++, in progress>
- Update all READMEs to point to branch <Neel, Pawan?>
- Set agenda item to decide how to handle coverage reports <Allen>

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
#193	18-Jun-21	odarcy1	CHECK_XLEN macro will silently pass	N/A	Close on 31Jul unless objections
#107	22-Apr-20	jeremybennett	Clang/LLVM doesn't support all CSRs used in compliance test suite	under discussion	-will close in 3wks; all uses deprecated
#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
#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

Tests merged into the ACT test_suite repo must :

- conform to the current format spec (macros, labels, directory structure)
  - including framework-readable configurations - i.e. which ISA extension it will be tested with (using Test_Case macro parameter equations) for each test case
- use only files that are part of the defined support files in the repository, including standard trap handlers
  - TBD: how to install test specific (not model specific) handlers
- Be able to be loaded, initialized, run, signal completion, and have signature results extracted from memory by a/the framework
- run using the SAIL model and not fail any tests
- generate signature values either
  - directly from an instruction result (that can be saved & compared with DUT/sim)
  - by comparing an instruction result with a configuration-independent value range embedded in the test code (e.g. saving above, below, within)
  - by comparing an instruction result with a configuration-independent list of values (e.g saving matches or mismatched)
    - (it can be useful to also return a histogram of value indices that matched)
- Store each signature value into a unique memory location in a signature region that is
  - delimited by standard macros embedded in the test which can be communicated to the test framework
  - pre-initialized to values that are guaranteed not to be produced by a test
- have defined coverage goals in a machine readable form that can be mechanically verified
- improve coverage (compared to existing tests) as measured and reported by a coverage tool (e.g. ISAC)
- use only standard instructions (and fixed size per architecture macros, e.g. LI, LA are allowed)
- be commented in test_case header (ideally listing coverpoint covered)

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

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 to be used inside tests based on the YAML configuration
- Include the compliance trap handler(s), & handle its (separate) signature area(s)
- 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 for test generation
    - 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

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 Compatibility Test 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 Compatibility Test 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 Compatibility Test 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

Meeting 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 unilaterly. 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, ...