# Architectural Test Task Group Call – Minutes

Thur, 12Aug2021 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 2$^{nd}$/4$^{th}$ 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/stable/ | riscof | https://github.com/riscv/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/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 Passcode: tHAR#5$V | | |

# Meeting Agenda

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

I.    Updates, Status, Progress:
- I.    Sail transfer is still ongoing. , but we do have a maintainer that can review and merge PRs
- II.   There is a preliminary code that uses YAML to generate Sail CSR access code.

## II. Next steps and Ongoing maintenance

1. Riscof plugin generation
2. Riscof: Makefile -> Python plugin support code
3. Discussion: testing methodology for SIGs/TGs needing external stimulus/observability "ports". (see slide 9 proposal)
4. Discussion: ACT and errata policy
5. 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) Gaps: missing D support in RV32, Sail CSIM compilation issues,
   - c) Review Pipecleaner tests: What do we need to do to exercise capabilities for Priv Mode tests
6. 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
7. 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
8. Specific Arch-Test Policy/Process Gaps:
   - a) Identify Tool providers, e.g. coverage model, test generation for new features/extensions

# Discussion

1.  Marc Karasek presenting: RISCOF port example

Migration from old style to RISCOF
  Created an "example_dut" directory  with riscof_example_dut.py  and all req'd  scripts
  This is an example template for converting from a Makefile to the .py file
   It   does not install the RISCV toolchain;
   It *does*.    install SAIL and everything else, e.g. local_install_sail.sh.
   Not tested with Docker.
   It still needs to be but into the repo
**ViceChair**:  Are there steps that require sysadmin privileges?
   **Inspire**: sudo_install_sail.sh  is the only step that needs to be run by sysadmin
**Chair**: This needs a README before we put it into a repo
**ViceChair**: Where repo should it go into?
   **Inspire:**  I propose yet another repo
**InCore**  Use RISCOF plugin method.
                 Updated docs for building your own plugin:
                    https://riscof.readthedocs.io/en/plugin-doc-update/plugins
                 Will be adding one more migration plugin by end of week:
                    https://gitlab.com/incoresemi/riscof-plugins
       < SAIL compilation "TEST_CASE_1 error discussion , fixed>)
**Chair**:  Did you use about "riscof –setup"
      **Inspire**:  Yes, used to initialize    riscof_example_dut.py
**Inspire**:  DUT artifacts should be separated from SAIL artifacts. **(AI)**
**InCore**    .py file will be added to riscof documentation. **(AI)**
**Chair**:  Is the way to look at
     changes from current framework → RISCOF framework
     is primarily changing from Makefile → RISCOF plugins.?
      **InCore**   yes. < documentation examples of those changes is up & ready for use>

2. new migration plugin tool

**Incore**: There is a new tool that will convert existing Makefile.include into riscof_<DUT>.py.
Caveats: assumes a particular coding style in the makefile (e.g. which $ variables correspond to which parameters)

**Inspire**:  The problem with this tool:  it's a half-step,  not really a migration process.
    People won't move entirely to native python/YAML; they'll just stop at Makefile.ini changes.
     **InCore**  the intent is conversion would only be used for a "short" (3-6 months) period of time.
     **Chair**:  it's a migration helper tool
     **Inspire**:  it really isn't a helper tool.  Nothing is migrated; we need to force python transition.
     **Chair**:  it would be interesting to compare results from Marc & incore


3. Other updates:

**InCore**   IITM FP tests take a very long time:  50,000 coverpoints
Coverpoints for fmadd.s: 58254 (with IBM b1 and B15 models)
Coverpoints for fmul.s: 6500
coverpoints for FP tests described here:
        https://riscv-isac.readthedocs.io/en/stable/code.html#module-riscv_isac.fp_dataset


Sail does not support D-ext on RV32
BitManip having problems - tests written can't compile on Spike or Sail
RV32E E/M/C tests now available, being reviewed.

**Syntacore**:       vector testing:  is there development wrt vector tests?
                          can we go over extensions in flight and the arch tests status?
     **Chair**:        RIOS labs is working on tests and Sail support. Full status can be found here
https://docs.google.com/spreadsheets/d/1qzu6b9kgADGjaa5fd1Qla7b9gCMOaEnGO5bUVu2oPys/edit#gid=156613795

# Decisions & Action Items

## Decisions

**Outstanding Action Items**

- Add example plugin and scripts into repo

- 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>

- Update all READMEs to point to branch <Neel, Pawan?>

- Update standard trap handler code for added priv levels, custom exception handler registration, <Allen, under review>

- DUT artifacts to be separated from SAIL artifacts in riscof <Neel,Pawan>

- Migration tool to be added to riscof repo <pawan>

- Marc's example plugin to be added to riscof repo <Marc,Neel> (with updated documentation)

# External Event ABI

- Why?
  - We want to be able to test events like: interrupts, concurrent reads & writes
  - These events would inject interrupts, modify memory at some future point

- What?
  - From at test perspective, these are model-specific macros that invoke vendor provided code
  - From an RTL perspective, this would look like a write to a specific MMIO "trick box" that RTL testbenches implements

- Possibilities
  - RVMODEL_ASSERT_INT(int, edgelevel, polarity, Trigger)
    - Int is a bitmask for simultaneous interrupts
    - Edgelevel, Polarity are masks for the type of signaling
    - Trigger is what initiates the event (e.g. #cycles from the write, debug trigger event, instret offset)
  - RVMODEL_MEMWRT(address, data, event)

- Questions
  - Should there be a pseudo-randomized Trigger?
  - What events should be standardized?
  - Do we need deassertion macros, Read macros? (we actually have specific interrupt deassertion macros now).
  - Are there another type of Event we should consider?

# 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 |
| #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 |
| pull#199 | 01-Aug-21 | bilalsakhawat | Fix for issue #142 , Adds RV32EC, EM tests | | Wait for RV32E spec? rename unratified |
| #119 | 17-jun-20 | allenjbaum | Missing RV32i/RV64i test: Fence | Test has been written | Close when RFQ test is merged |
| #189 | 26-Apr-21 | neelgala | Proposal to enhance the RVTEST_ISA macro | | |
| #190 | 26-Apr-21 | neelgala | The 16-byte signature boundary issue | | |
| Pull#201 | 17-Aug-21 | Liweiwei90 | Update K-ext tests | | Updates for spec changes, improved Sbox coverage |

# 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 | Will become ACT policy |
| **CSC-3** | 20/Aug/20 | Ken Dockser | Come up with an internal goal for what we wish to accomplish with the Architectural Tests | | Will become ACT policy |
| **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, ...