Architectural Test SIG Call –Minutes

Thur, 09Dec2021 8am Pacific → Standard ← Time

See slide 7 for agenda

RISC-V attendance

Only RISC-V Members May Attend

- Non-members are asked to please leave.
- Members share IP protection by virtue of their common membership agreement. Non-members being present jeopardizes that protection
- It is easy to become a member. Check out riscv.org/membership
- If you need work done between non-members or or other orgs and RISC-V, please use a joint working group (JWG).
  - used to allow non-members in SIGs but the SIGs purpose has changed.
- Please put your name and company (in parens after your name) as your zoom name. If you are an individual member just use the word “individual” instead of company name.
- Non-member guests may present to the group but should only stay for the presentation. Guests should leave for any follow on discussions.

RISC-V®

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®

Collaborative & Welcoming Community

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 are a continuous improvement organization. If you see something that can be improved, please tell us. help@riscv.org

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

https://riscv.org/community/community-code-of-conduct/

RISC-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

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-non-isa /riscv-arch-test/tree/master/doc	tests	https://github.com/riscv-non-isa/riscv-arch-test
https://github.com/riscv-software-src/riscof/tree/master/docs	riscof	https://github.com/riscv-software-src /riscof
https://github.com/riscv-software-src/riscv-ctg/tree/master/docs	Test Gen.	https://github.com/riscv-software-src /riscv-ctg
https://github.com/riscv-software-src/riscv-isac/tree/master/docs	YAML, WARL config	https://github.com/riscv-software-src /riscv-config/
https://github.com/riscv/sail-riscv/tree/master/doc	Sail formal model	https://github.com/riscv/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. RV32F and RV64D merged, RV32D now supported in Sail, RV64F coming along, SigUpd macro changes being tested
II. Bitmanip ratified, has toolchain and Sail support, support being added to isac, ctg, Sail support merged, tests still ongling
III. Crypto Scalar: ratified, waiting for updated toolchain support, PR to rename K_unratified to K and update ratified mnemonics filed
IV. Zfh, zfinx: Started working on arch-tests this week.
V. Zmmul: public review soon…
VI. Updated trap handler passed first simple test , will handle nested traps/interrupts, multiple priv modes, more robust error checking

II. Next steps and Ongoing maintenance

0. Meeting Schedule – this is the last meeting of the year
1. Updates to Current Spec – split into Test Guidelines and Vendor Interface spec
   Vendor Interface is primarily 3 required RVMODEL_ macros (DATA_BEGIN/END, and HALT) and various interrupt/debugmsg/boot code macros
   - Will be updated to add asynch testing (dummy interrupt and event generation device), and external debug ger)
   - this will require specific model interfaces, e.g. wired interrupts, debug messages, timer support)
2. Discussion: other steps for Migration to Framework v.3.0 (riscof). (blocking items):
   a) Reference signature docker image, local podman docker plugins, remote podman YAML2refsig-as-service implementation
   b) (Sail/Spike model updates, pipecleaning, N people have run it, testing all the “fixed in riscof” issues
   c) Review Pipecleaner tests: What do we need to do to exercise capabilities for Priv Mode tests
3. Dynamic Test Generation
   1. Related: how should we deal with 1GB test directories (FP
4. Discussion: starting a TG to precisely define the ABI for asynch event generation and a reference C-model to be used for Sail
5. Config YAML GUI interface demo

III. Future Agenda items

1. Maintenance updates to V2 to enable future tests

Discussion

Status: see previous slide

RVTEST spec discussion. Test format spec.

Chair: Walk through the various RVTEST macros
Imperas: I don't recall ever hearing about interrupt macros
Chair: They've been in the spec for years.
Will be needed when we start testing interrupts, specifically clearing architecturally defined interrupts (external, SW, Timer)

Discussion of Docker and reference signature generation

Chair: Problems with installing and SAIL.
One solution: run in a Docker image.
Incore - we already have the Docker image and plugins
Chair - issue issues with installation of Docker: needs sudo access
Other solutions:
- podman (sudo not needed, compatible w docker containers)
-singularity (open src , sudo not needed, not docker compatible
-reference signature generation as a service. (RSAAS)

RSAAS : Install docker container on a server
Send YAML file to the service, get signatures in return.

Imperas - for our vector tests: 237MB references sign. 30 hour run time
ViceChair RTL sims?
Imperas No . Instruction Set Simulator, for a single vector configuration
Chair We will be running this on Sail which is likely a slower ISS
Note that we are not requiring RSAAS;
Reference signatures can still be run on your own machines
Two possible saving graces:
- compatibility tests may not be as extensive as Imperas DV test
- only needs to be run once per implementation

Other RSAAS issues
Chair – vendors would be sending out the YAML describing what they're building; They may want to keep that private.
Incore - whenever ref signatures are sent out, need to also need to get a set of other artifacts to say what tests were run, and other stuff
Chair: ACT policy requires that, but we don't need to get that from the container. The tests signatures will identify themselves by file name

Chair - question about 100MB signature files: have [Imperas] tried hashing the signatures? a vectorized hash function?
Imperas - We've done a POC. but we don't have a need for it yet.
The signatures are not the real problem.
The problem is size of data in the asm file.
Source for code is 2GB which results in 250MB of signatures.
We had issues breaking the toolchain as well as other parts.

Chair This is a a single configuration out of ..90? !!

Vector is likely to be a worst case

AI: track down Resources that can host file systems with that much (temporary) user space and can host very long runs. We will know our requirements better when ACTs for vector are submitted.

Decisions & Action Items

Decisions (from last meeting)

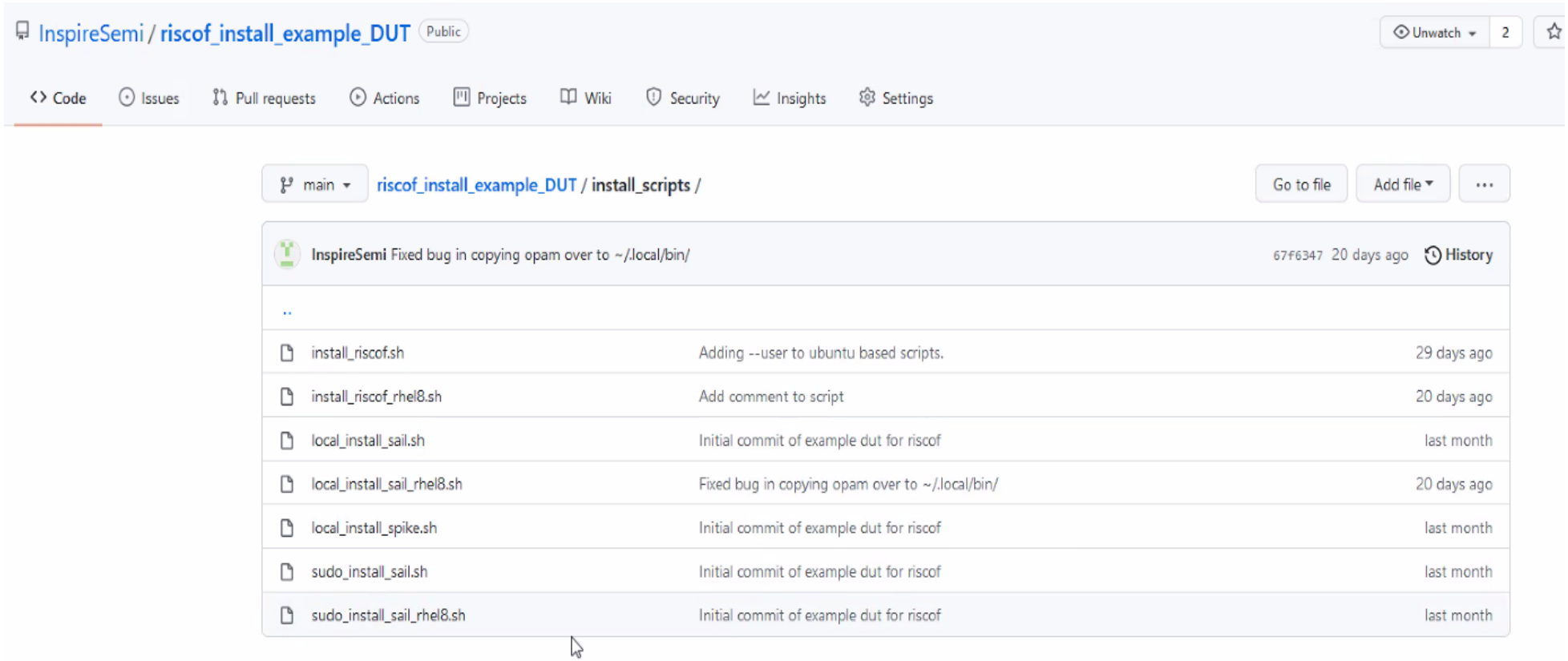
- Revised Decision : Sail interface to Asynch event generator will be an MMIO write
  - Address is implementation dependent.
    Easier to implement in RTL
- Decision : Sail will trigger event generation when instret > delay+(instret@Store of) Delay
- Decision: test format spec will be split into 2 pieces:
  RISC-V_ACT_MODEL_INTERFACE_SPECIFICATION
  RISC-V_ACT_TEST_DEVELOPERS_GUIDE
  the interface spec is subject to non-ISA spec ratification

Outstanding Action Items

- config structure needs translators to/from ACT YAML <added to Dev partners spreadsheet>
- find a different place to put coverage reports, e.g. google drive folder < Jenkins file preferred>
- Develop a C-model of an asynch event generator and a Sail+Spike interface to it <set up a TG under Simulator SIG?>
- Update all READMEs to point to branch <Incore?>
- Update standard trap handler code for added priv levels, custom exception handler registration, < Chair, under review>
- 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 < Chair, Jeff-in progress>
- Look for and setup ref-signature-as a service site using docker image of Sail and tests < Chair >
- Develop plugs for podman as well as remote container < incore? >

BACKUP

Example riscof repo



External Asynch Event Support

- Why?
  - We want to be able to test events like: interrupts, concurrent reads & writes
  - These events would inject interrupts (wired and msg signaled), modify memory
- 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 implement
    - We will provide one as a Sail external function in C, and Spike
  - The ACT SIG (or a TG under it) will develop an implementation template as a C model
- Strawman C interface
  - RVMODEL_EVENT(instret, delta, eventtype_num)
  - RVMODEL_MEMWRT(instret, delta, size, address data) ←note that this could be a Message Signalled interrupt
  - RVMODEL_MEMRD(instret, delta, size, address)
    - Assembly language implementation are 1-3 successive writes; events are scheduled on the last one
    - Vendor code is called every cycle, and asserts events to the model when curr instret>=orig_instret+delta (instret is global variable, sampled on final MMIO write)
    - Possible schedule multiple events with appropriate deltas, even enabling simultaneous events + rd/write} (but only single rd/wr per cycle)
  - RVMODEL_EVT_STATUS() – returns values read by RVMODEL_MEMRD()
- Strawman ACT Macro Interface
  - RVMODEL_EVENT(typ_evtnum_ delta_srcreg)
    - Assembly language would store 1 word at a platform specific address/CSR: edgelvl/polaritytype/evt_num encoded in high half, delta in lower)
  - RVMODEL_MEMWRT(addr_srcreg, data_srcreg, sz_delta_srcreg) ←note that this could be a Message Signaled interrupt
    - Assembly language would store 3 words at platform specific mmio addresses/CSRs: address, data, sz+delta
  - RVMODEL_MEMRD(addr_srcreg,typ_ sz_delta_srcreg)
    - Assembly language would store 3 words at platform specific address or into 3 CSRs: address, data, type+sz+delta (type is ownership/temporal, etc)

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
#146-9	01-Dec-20	Imperas	Test I EBREAK,ECALL, MISALIGN_JMP/LDST, OpenHW	|	HW misalign support not configurable
#189	26-Apr-21	neelgala	Proposal to enhance the RVTEST_ISA macro	v	
#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
#190	26-Apr-21	neelgala	The 16-byte signature boundary issue		
#203	24-Aug-21	Allenjbaum	Fence test has poor coverage		Specifically: test fm bits are ignored
#211	19-sep-21	Neelgala	default rvtest_data should be 16-bytes		
#214	05-oct-21	Allenjbaum	Test Format Spec doesn't specify the order of line in the signature file		Spec clarification
#218	18-oct-21	abukharmeh	Makefile appears to be broken for OPENHW targets		
#220	20-oct-21	Davidharrismc	F tests		Add new F tests to makefile so it works OOB
#223	23-nov-21	fborisovskii	fcsr reset value		Clr sticky status before tests, add coverage

JIRA Status

Issue#	Date	submitter	title	status	comments
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		This is the /test coverpoint YAML
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

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

Framework Requirements

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

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>/