Architectural Test SIG Call –Minutes

Thur, 23Sep2021 8am Pacific → Daylight ← Time

See slide 6 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

←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/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 pull request has been made, RV32D nowsupported in Sail
   II. BitManip tests almost ready.

II. Next steps and Ongoing maintenance

1. Using Docker to ease reference signature generation
2. Discussion: testing methodology for SIGs/TGs needing external stimulus/observability "ports". (see slide 9 proposal)
3. Riscof plugin generation
4. Riscof: Makefile -> Python plugin support code
5. Discussion: ACT and errata policy
6. 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
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 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
8. 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

Discussion

Status:
Administrative pointers have changed. (slide 5)
F/D tests: tests ready, but no Sail RV32D support . See pull#205:
https://github.com/riscv/sail-riscv/issues/96 .
Bitmanip: Tests almost done. SAIL model support is incomplete, so no coverage rpt
Vector Presentation from RIOS at Sept 15 Forum
Sail repo now transferred to riscv repo

1a. Docker security issues :
Bristol: : can we trust what is in the image, reported docker security issues
Inspire: "recipe" can be examined; everything in the image is already required with the exception of the OS (e.g. Ubuntu).
Incore: In addition, image configured to disable external access (internet/ethernet)
Otherwise, only the Docker engine (daemon) is new, installed by user
1b. open source issues:
All pieces of the docker image, and the Docker engine itself are open source
1c. Permission issues:
Docker doesn't have to be run as root, but can't mount arbitrary file systems
Incore: we need to pass data both directions:
YAML config files, ELF tests in, reference_signature out
Many/most file systems are support, but not necessarily all
If you have sudo privileges, you don't have to worry about it.
1d. Docker image size
Incore: 1.8GB is what i've generated. Built in layers.
Chair: we could support multiple images, all inclusive or one that needs to rebuild
Pawan: example: can download the binaries for toolchain, or you can download the source code and re-build. Solves the dependency problem.

2. External event ABI
Two interrupt extensions are coming. need support.
Macro ABI is good enough for current testing framework
Not good enough for DV though
Events are scheduled
<Discussion about scheduler for external event and how it might affect signature.>
What is the interface to Sail that can trigger these events? Is there one?
Interrupts - should be interrupt events
SAIL: Jessica Clarke and Martin Burger (maintainer) are main contacts.
Bristol: gave a short description of SAIL. There is a RISC-V model that is run inside of an IO environment (whether its memory, interrupt, entropy, etc.)
Sail step function can call the external environment to assert the events.

Q: how do we trigger a future event? Dbg trigger, hpmcounter, cycle delta, instret delta,
Chair: triggers are optional, hpm counters are optional, cycle is variable, time is variable
Instret counter is deterministic at one level, but if multiple instructions are retired per cycle, it won't be exact – but should be in range 0..max#of insts retired/cycle
Event trigger defined as (instret>=curr_instret+param)
Q: What should the Sail instruction interface?
MMIO write is too variable; CSR write should be a synchronizing event, and Sail can easily detect it

Decisions & Action Items

Decisions (from last meeting)

- Decision: We will use an elf->ref-signature rootless Docker image
- Decision : There will be plug-ins for both local Sail install or Docker image
- Decision : Sail interface will be a CSR write
  - TBD: which CSR.
    Stake in the ground: largest custom RW CSR: 8FF
- Decision : Sail will trigger event generation when instret > (instret@CSR_wt_time + CSR_write_param)
  - -TBD: max# of schedule event

Outstanding Action Items

- Add example plugin, scripts into repo (especially for docker) <done, https://gitlab.com/incoresemi/riscof-plugins/-/tree/master/sail_cSim>
- Migration tool to be added to riscof repo <done>
- DUT artifacts to be separated from SAIL artifacts in riscof <done>
- Fix uses of RVTEST_ISA macro in various tests (formatting incompatibility with riscof and update spec <done>
- 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>
- 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>
- Marc's example plugin to be added to riscof repo <Marc,Neel> (with updated documentation)

Cadence Support for OSes

2020-2022 Cadence Compute Platform Roadmap

Arch	OS Name	OS Version	Base Releases		
			2020	2021	2022
x86_64	RHEL	6.5+			
		7.4+			
		8			
	SLES	11 SP4			
		12			
		15			
	CentOS*	6.5+			
		7.4+			
		8			
	Windows	Windows 10			
		Server 2012			
		Server 2016			
		Server 2019			
IBM POWER	RHEL LE	7.2+			
		8			
Arm v8	RHEL	7.5+			
		8			

Supported Selected products Not supported Dropped

2021 base releases:
- x86_64
  - RHEL 7.4 as baseline
  - EOL SLES 11 support
  - EOL RHEL 6 support
  - EOL Windows Server 2012
  - Add Windows Server 2019
  - No CentOS 8 support (see Red Hat announcement)

2022 base releases:
- x86_64
  - Add SLES 15

* Cadence supports CentOS, but disclaims any liability for any errors or bugs in CentOS

BACKUP

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)
  - RVMODEL_MEMRD(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
#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
#203	24-Aug-21	Allenjbaum	Fence test has poor coverage		Specifically: test fm bits are ignored

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

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>/