Architectural Test SIG Call –Minutes

Thur, 13Jan2022 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.

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.

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. Zmmul: in public review
- II. Updated trap handler passed first simple test, handles nested traps/interrupts, multiple priv modes, more robust error checking
- III. Fixed riscof report errors, add retention capability (so doesn't clear artifacts between runs)
- IV. P-extension tests written, Sail support written but not merged

II. Next steps and Ongoing maintenance

1. Asynch Event Generator TG – a model proposal (see slides)
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-AAS 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. 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)
4. Dynamic Test Generation
   1. Related: how should we deal with 1GB test directories (FP
5. Discussion: starting a TG to precisely define the ABI for asynch event generation and a reference C-model to be used for Sail
6. Config YAML GUI interface demo

III. Future Agenda items

1. Maintenance updates to V2 to enable future tests
   a) Convert assertions to be out-of-line
   b) add assertion macros for FP, DP, Vreg to arch_test.h and test_format spec

Discussion

Status: see previous slide

Repo Maintenance
Incore : Work needed for repos: rename all "master" branches to "Main" branch to follow naming guidelines

Incore: Main branch currently allows pushs on ctg, isac, riscof, and riscv-arch-test, but shouldn't; only filing, approving and merging PRs should be allowed. This needs to be fixed
AI: talk to help@riscv.org to fix permissions

FP tests
Incore: FP F->D conversion coverpoints are wrong (assume D->F), needs fixing AI: fix

Interrupt Testing
Chair: Presentation of Asynch Event Generator proposal (see sides 10-16)
Q: can't we do this just by write xIP CSR?
A: We can, but only very specific circumstances. SW writeable xIP bits are not required, and cannot scheduled in advance, so can't interrupt user code, can't deal with message signaled interrupt, and this doesn't allow memory model testing

Simplifying Sail Reference Signature Generation
Incore: A docker image exists; it includes Sail, Spike, and toolchain.
It's 1.8GB, and there is a riscof plug to use it.
If you can't run docker, you can download repos, you can manually build the image an run it using a different plugin
<demo of docker image running tests>
Podman is an alternative that should allow running docker containers.
Couldn't get it running with the mount points in the image (that allows
- reading test files and YAML from outside the container, and
- writing test reports and coverage outside the container).
We need to find someone with podman experience to help.
Another option is singularity, which is not docker compatible so would need to be developed separately, and we need someone with singularity experience to help on that.

All the plug-ins and docs are available on the incore public repo

The docker image can be found here:

and the dockerfile for the same is her

The docker image can be found here:
https://gitlab.com/incoresemi/docker-images/container_registry/2205130
and the dockerfile for the same is here:
https://gitlab.com/incoresemi/docker-images/-/blob/master/compliance/Dockerfile.

Decisions & Action Items

Decisions ()

Outstanding Action Items

- find a different place to put coverage reports, e.g. google drive folder < Jenkins file preferred>
- Update all READMEs to point to branch <Incore?>
- Update standard trap handler code for added priv levels, custom exception handler registration, < Chair, initial tests pass>
- 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 AND < Chair >
- Develop plugs for podman as well as remote container < HC? >
- Fix tools (ctg, riscv-config, isac, riscof,) to require PRs for changes < help@riscv.org? >
- Set up a TG to define Async Event Generator specs (test interface, Model interface, generator SW that can interface to RTL and simulators, sample shims for Spike and Sail <chair>
- Fix FP D->F convert tests <IIT>?

Architectural Event Generator Proposal v0.1

Arch-Test SIG

Allen Baum, Chair

Rationale

- To properly test the architectural handling of interrupts, tests need to be able to arbitrarily cause interrupts to be generated
  - Timer interrupts can be generated by Mcode, though there can be large timing uncertainty between implementations that make this difficult to reproduce on a reference model (e.g. timer phase and granularity) unless implemented by test framework
  - Software Interrupts can be generated with limited coverage or using platform specific parameters and external support with much lower timing uncertainty
  - External interrupts cannot be generated without external support with low timing uncertainty
- There are several extensions in progress for external interrupt:
  AIA, Fast Interrupts, CLINT,
  so test support is needed
- In addition, support is needed for generated asynchronous reads and writes to test the memory model
- Since this has a “work product”, this will be a non-ISA TG under the ACT SIG
- The charter will include
  - Implementation agnostic Hart interface Spec
  - A SW model of the pseudo device that can be integrated with an RTL model
  - Two example shims to integrate it as Sail and Spike external functions

High Level Pseudo-Device Proposal

- Develop a “trick box” artificial IO device that can generate both interrupt wires and MMIO reads and writes
- Tests will write to the device to schedule
  - which interrupt wires (or wires) are asserted to the DUT,
  - when they are asserted or de-asserted
- Tests will write to the device to schedule
  - when memory transactions are made to the DUT,
  - the physical address to be used,
  - the type of memory transactions (e.g. read, write, other?)
  - the data to be written (in case of a write)
  - The data that is read (in case of a read)

  Note: this is also used to deliver message signaled interrupts

Diving Deeper

- The pseudo-device has a C-compatible function interface
  - it is called every cycle by the DUT, and monitors the the external memory bus
  - The vendor is responsible for interfacing this to their DUT at a DUT specific PA
  - The inputs to the function are:
    - Physical address (assume XLEN, zero extended)
    - WtData (XLEN) –
      - interrupt signal set/clr mask (*Note: interrupts are level sensitive positive polarity only)*
      - MSI address
      - TimingDelta, operation (wired interrupt set/clr/read/write/other
    - RdData (XLEN) – primarily used for debugging
    - #InstRet on this cycle (4b max, zero extended) *note: the eTrace extension needs this also*

Diving Deeper Still

- Every write to the device pushes write data onto a specific FIFO, depending write address offset
- There are 3 fifos at 3 separate offsets. The DUT will ignore writes to other offsets (base device address is vendor supplied in config YAML)
  - *Address/Mask Fifo*
  - *WtData Fifo*
  - *Control Fifo: TimingDelta and Operation, IntAck?*
- Read to other offsets will just return internal state from the fifos for debugging.
- Writing the Control Fifo arms the oldest unarmed event in the address FIFO (and the WtData Fifo, if the operation is a write)
- Each cycle, the called function will decrement the InstRet counter of all entries in the TimingDelta field.
- When the InstRet field value decrements to (or below) zero, the operation is issued from the operation field with the parameters from Address/Mask FIFO, and optionally the WtData macro if the operation uses it, and pops them off the top.
  - Writes with TimingDelta of 0 will be performed immediately – use to Ack/clear an interrupt?

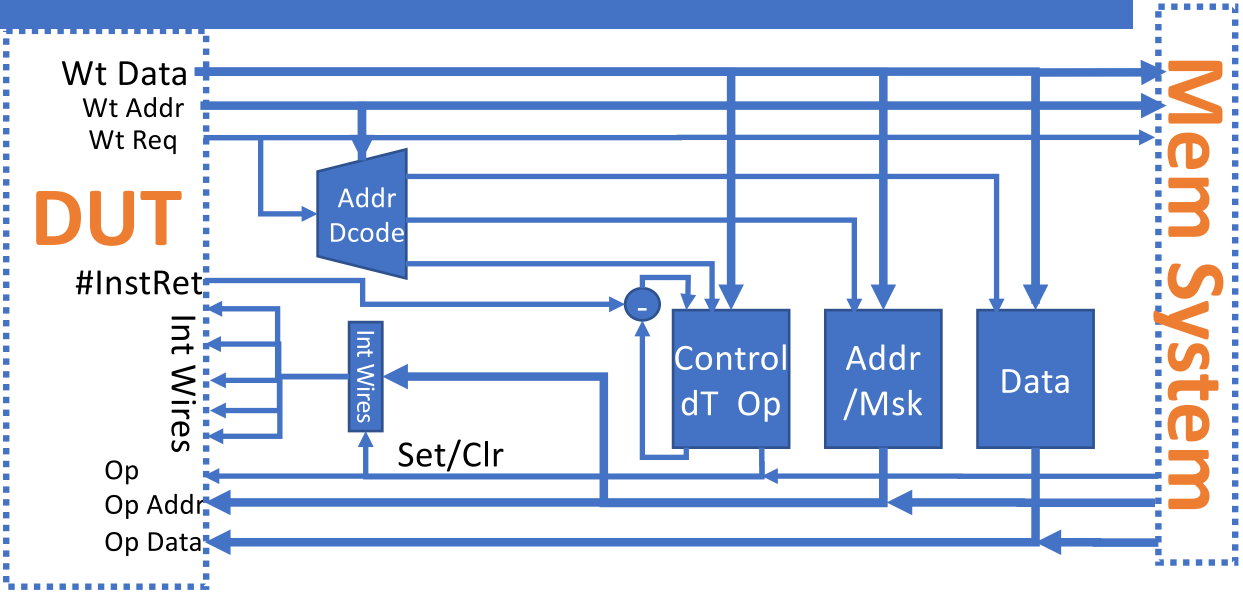
Note: fifos can be written in arbitrary order except controls must be written after the parameters it uses, so you can write multiple Masks/Addresses, multiple WtDatas, and finally multiple Controls. Controls that are written without any unarmed addresses/masks will be silently ignored.

External Asynch Event Support

- Strawman C interface (and hart interface)
  - RVMODEL_ASYNCH_EVENTGen(Instret[3:0], wt_cmd[??], wt_addr[phys_addr_sz], wt_data[XLEN-1:0], interrupt_out[XLEN-1:0], cmd_in[??], addr_in [phys_addr_sz], data_in [XLEN-1:0])
- Strawman ACT Macro Interface
  - RVMODEL_ASYNCH_EVENT_ADDR(BaseReg, AddrReg)
    - Assembly language would store 1 word at a platform specific mmio address
  - RVMODEL_ASYNCH_EVENT_DATA(BaseReg, DataReg)
    - Assembly language would store 1 word at a platform specific mmio addresses
  - RVMODEL_ASYNCH_EVENT_CMD(BaseReg, CmdReg, Delta, Cmd, [ResultReg])
    - Assembly language would construct and load or store 1 word at a platform specific mmio addresses
    - Delta and Cmd could be a constant or a register – TBD

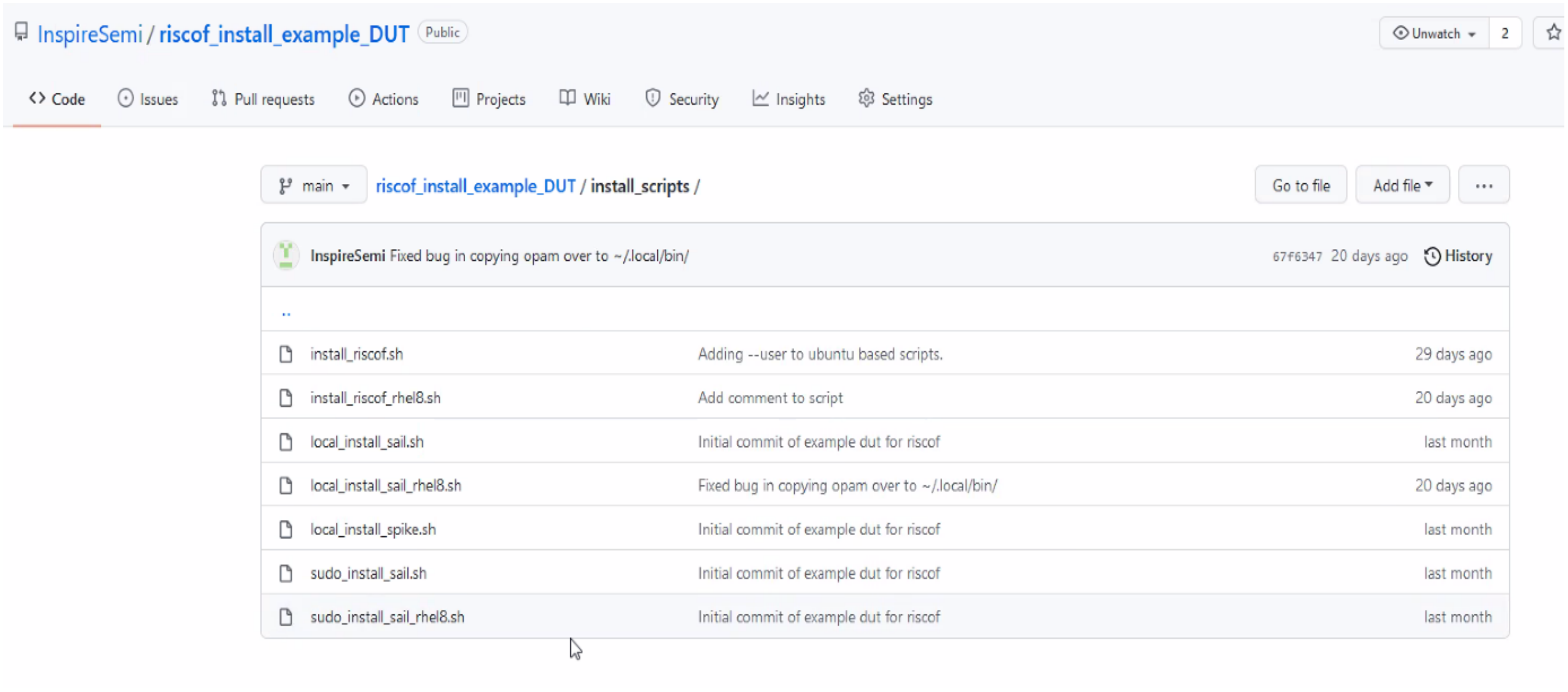
Questions

- *How should multiple TimingDeltas simultaneously becoming <=0 be handled?*
  - *E.g.Issue at most one wired & 1 write?*
  - *Just one at a time in fifo order?*
- *Should we allow events to be taken out of middle of the fifo or go strictly in order?*
- *Which operations are needed besides Rd and Wt? (e.g.inval, snoop…)*
  - *Is a single XLEN width write adequate?*
- *Precisely what should the read/write interface be*
  - *e.g. AXI subset?*
  - *Do we need an explicit mux to interface with memory?*



BACKUP

Example riscof repo



riscof_install_example_DUT / install_scripts /

InspireSemi Fixed bug in copying opam over to ~/.local/bin/ 67f6347 20 days ago

..

install_riscof.sh	Adding --user to ubuntu based scripts.	29 days ago
install_riscof_rhel8.sh	Add comment to script	20 days ago
local_install_sail.sh	Initial commit of example dut for riscof	last month
local_install_sail_rhel8.sh	Fixed bug in copying opam over to ~/.local/bin/	20 days ago
local_install_spike.sh	Initial commit of example dut for riscof	last month
sudo_install_sail.sh	Initial commit of example dut for riscof	last month
sudo_install_sail_rhel8.sh	Initial commit of example dut for riscof	last month

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
pull#225	08-dec-21	Phthinh	update the K extension for the V.1.0.0 ratified spec	Needs review	Looks good to go
#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
#220	20-oct-21	Davidharrismc	F tests		Add new F tests to makefile so it works OOB
pull#226	17-dec-21	liweiwei90	add support for cbo.zero in cmo extension	Needs changes	

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? (can't test the trap is correct then)
- 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>/