Compliance Task Group Call – Minutes

Weds, 13 May2020 8am Pacific →Daylight← Time

See slide 6 for discussions and action items

Charter

The Compliance Task Group will

- Develop compliance tests for RISC-V implementations, taking into account approved specifications for:
  - Architectural versions (e.g. RV32I, RV32E, RV64I, RV128I)
  - Standard Extensions (M,A,F,D,Q,L,C,B,J,T,P,V,N)
  - All spec'ed implementation options
    - (incl. MHSU modes, optional CSRs, optional CSR bits)
- Develop a method for selecting and configuring appropriate tests for a RISC-V implementation, taking into account:
  - Platform profile and Execution Environment (EE)
  - Implemented architecture, extensions, and options
- Develop a framework to apply the appropriate tests to an implementation and verify that it meets the standard
  - test result signature stored in memory will be compared to a golden model result signature

Adminstrative Pointers

- Chair – Allen Baum allen.baum@esperantotech.com
- Co-chair – Bill McSpadden bill.mcspadden@seagate.com
- TG Email tech-compliance@lists.riscv.org
  - Notetakers: please send emails to allen.baum@esperantotech.com
- Meetings -Bi-monthly at 8am Pacific time on 2nd/4th Wednesdays
  - See https://lists.riscv.org/g/tech-compliance/calendar entry for zoom link
- Documents, calendar, roster, etc. in https://lists.riscv.org/tech-compliance/
  see /documents, /calendars subdirectories
  - https://riscof.readthedocs.io/en/latest/ riscof
  - https://riscv-config.readthedocs.io/en/latest/ config: YAML and WARL spec
- Git repositories
  - https://github.com/riscv/riscv-compliance/
  - https://github.com/rsnikhil/Experimental_RISCV_Feature_Model
  - https://github.com/rsnikhil/Forvis_RISCV-ISA-Spec
  - https://gitlab.com/incoresemi/riscof (Shakti framework)

Meeting Agenda

- Updates
  - New: AUIPC coverage (current test doesn't propgate results to signature!)
  - New: change Makefile to turn assertions w/ parameter (instead of modifying file)
  - Funding RFQ review
  - Issue status – how do deal with "fixed in riscof" issues? – not discussed
- Progress:
  - Standard Default Trap Handler – nearly written,

Action Items from last meeting

Allen & Jeremy: add upstreamed gcc link (done)

- For each OS, there is a tar file of the compiler binaries and a link to the regression test results. Components are upstream top of tree from 17 April 2020. Provides binutils (inc linker, assembler), GCC compiler, GNU Debugger and newlib C library. (more details at https://www.embecosm.com/2020/04/17/up-to-date-risc-v-gcc-tool-chain-packages/)
- Centos6 Tarball
  https://buildbot.embecosm.com/job/riscv32-gcc-centos6/1/artifact/riscv32-embecosm-centos6-gcc-20200417.tar.gz
  Results: https://buildbot.embecosm.com/job/riscv32-gcc-centos6/1/
- Centos7 Tarball:
  https://buildbot.embecosm.com/job/riscv32-gcc-centos7/1/artifact/riscv32-embecosm-centos7-gcc-20200417.tar.gz
  Results: https://buildbot.embecosm.com/job/riscv32-gcc-centos7/1/
- Centos8 Tarball:
  https://buildbot.embecosm.com/job/riscv32-gcc-centos8/1/artifact/riscv32-embecosm-centos8-gcc-20200417.tar.gz
  Results: https://buildbot.embecosm.com/job/riscv32-gcc-centos8/1/

Simon, Allen: get slides that were shared.
done: slides added to https://lists.riscv.org/g/tech-compliance/files/Presentations. as

"Imperas RISCV compliance simulators.pdf"

Discussion

- How do we define compliance?
  - How we use it: To use Risc-v trademark, must pass compliance tests
  - Being compliant implies Interoperability of apps between implementations
  - Concern: the word "compliance" implies too much
    - Use ARMs compliance definition
  - Krste said ~ 5 years ago: RISCV will supply a set of compliance tests for an extension.
    - Compliance goal: did you read the spec correctly?
- Are notes from TSC meeting review coming? (yes, next day or so, AI: SC)
- Review of RFQ document
  - **Task section**
    - Q: What does 90% mean? (A: of coverage points)
    - Google open source coverage tools; needs more work to accurately spec coverage bins. E.g.: counts negative shifts, but they aren't possible. the bin(s) need to be cognizant of this.
    - Currently can't have different coverage for different instructions.
  - **Specifications Section**
    - Q: Do we really require an automated test generator?
      - I don't think we can generate volume/coverage tests without it in a timely manner
      - The purpose of the statement is to make clear that RISCV is not looking for hand-generated assembly code, and expects that without it the volume of tests won't be available in a timely manner. Note that the statement says "ideally", so not a strict requirement, but we should ask vendor how they will generate the tests; that will become a factor in choosing between vendors
    - Q: Need to ask for support in contract if e.g. tests turn out to be buggy?
    - Q: How do we know that a test passed? (A: test matches signature)
      - Vendor should test it again several models
      - We should do this as part of our acceptance criteria, but especially run it on SAIL golden reference; this will be added to RFQ (AI: TSC)
      - Does SAIL run under Windows? (unsure; it is supposed to run under MacOS)
      - RFQ needs to call out Linux deliverables (It does say work needs to be done on Linux)
    - These are directed tests (walked us through the test flow)
      - Tests are run with assertions on; save signature. Compare against other simulations.
      - Assertions only work on open code, will break when in a loop (like in trap handlers)
    - Coverage requirements seems overly simple:
      - Referred to: https://lists.riscv.org/g/tech-compliance/files/Review Documents/coverage model - v.2 draft.pdf includes forwarding as coverage point
        - That is DV, not compliance, and Verification suite is expensive to develop, e.g. ARM has spent years and millions of $ to develop.
        - But availability of result of one instruction on the immediately following op is architectural, and is even speced in several places
      - Someone needs to fill out accurate coverage bins. This would be to the contractor. Needs to be done for every instruction. (AI: SH, needs to be done by next Thu board meeting)
    - Why Apache 2.0 license as a deliverable?
      - Discussed by the steering committee
      - We don't want people fiddling w/ tests. Too easy to get 100% compliance.
        - Seems like a non-issue; they can't fiddle with our tests in repository
      - License needs to be "permissive",
        - NOT GPL; Have lawyers define which license (AI: SC)
- Request: Add line numbers of section numbers in RFQ (agreed; AI: SC)
- **Acceptance, Operating Mode, Staff, Reporting, Timeline, Resource Sections**:
  - no comments
- **Excluded additional items section**:
  - Mostly there to indicate we are paying for just tests and the coverage reports, and that users should not have to pay for any special tools to run them
  - Q: How will vendor generate coverage report? A: They'll likely need to purchase a tool license to run coverage and generate the report if they don't already have one, but cost is bundled into quote
- **Responses Section** :
  - RFQ should give examples of coverage and data propagation reports
  - See coverage report in repository

Decisions & Action Items

Decisions

- Modify Makefile for parameterizable assertions
- Fix AUIPC test to propagate results to signature
- Modify RFQ with:
  - Add more detailed coverage model
  - Number the document sections
  - Settle on license requirements with lawyers
  - Add testing on Sail formal model as a requirement
  - Add pointer to coverage docs on github as example of deliverable

Action Items

- Imperas will update Makefile to enable assertion on/off with Makefile parameter
- Imperas will fix AUIPC test to propagate results to signature
- SH will produce a more detail list of coverage metrics
- SC will number the document sections
- SC will ask lawyers for license recommendations
- SC will add running test on Sail model as a requirement
- SC will add a pointer to coverage doc on github as an example of what a coverage report should have.

Pull/Issue Status

Issue#	date	submitter	title	status	
#04	3-Jul-18	kasanovic	Section 2.3 Target Environment	Fixed in RISCOF	
#22	24-Nov-18	brouhaha	I-MISALIGN_LDST-01 assumes misaligned data access will trap		
#40	4-Feb-19	debs-sifive	Usage of tohost/fromhost should be removed		
#45	12-Feb-19	debs-sifive	Reorganization of test suites for code maintainability		
#63	13-Aug-19	jeremybennett	Global linker script is not appropriate		
#78	26-Jan-20	bobbl	RV_COMPLIANCE_HALT must contain SWSIG		
#90	11-Feb-20	towoe	Report target execution error		
#27	21-Dec-18	jlucnagel	Macros are checking side-effects	Fixed?	Close ?
#72	26-Oct-19	vogelpi	Allow for non-word aligned `mtvec`	deferred	needs v.2
#84	4-Feb-20	byllgrim	I-SW-01 corrupts .text region	Pull#104	Close?
#105	22-Apr-20	jeremybennett	Non-standard assembler usage	under discussion	
#106	22-Apr-20	jeremybennett	Use of pseudo instructions in compliance tests	under discussion	
#107	22-Apr-20	jeremybennett	Clang/LLVM doesn't support all CSRs used in compliance test suite	under discussion	
#108	22-Apr-20	bluewww	RI5CY's `compliance_io.h` fails to compile with clang	under discussion	
#109	06-May-20	Olofk	Swerv fails because parallel make	under discussion	

Next Meeting Agenda (proposed, not final)

- Coverage metrics?
  - how to report coverage of YAML-selected/configured tests/test cases
- Google DV overview
- Transitioning to v.2 framework
- Next tests and priorities)
  - Better coverage of existing tests vs. new tests (primarily priv level)
  - E.g. RV32i->rv64i -> ratified extensions -> priv modes
- Get more repo maintainers

Backup from previous discussions

Draft Test Coverage Proposal (unpriv)

Classes of things we want to test for

- Decode
  - Immediate – test all bits in either polarity will affect output
  - Register specifiers – test that changing any bit will affect output, ensure all regs are tested
  - Variations – test values of opcodes suffixes that have any string after a "." in its opcode
- Register combinations
  - Destructive (dest = either src) and non-destructive
  - Non-updating (i.e., targeting X0), or non-supplying (X0 as an input)
  - All registers (or immediate bit) should be used per instruction *category*
- Special and exception cases
  - Explicitly defined (e.g. shifts>=XLEN & RD=X0
  - Implicitly defined – corner cases
    - Maximal and minimal inputs, or creating maximal outputs
    - Inputs that special case outputs (mostly FP cases, also. shiftamt>=XLEN)
    - Outputs crossing value boundary (e.g. address cross word/page/superpage/VA boundary, FP crossing exponent boundary)

proposed coverage & categories

Arith[I],	*W1/0,crys*
Logical[I],	*W1/0*
Shift[I],	*W1/0/msk,+*
Auipc,Lui,	
Ld,St,	*W1/0, bndXing*
Br,	*W1/0, bndXing*
Jmp ,	*W1/0, bndXing*
Ebreak/ Ecall	

W1/0= walking 1/0

BndXing=: boundary crossing

This works for 32i base ops – what do we need to add for priv modes? Mem model? Sequential Dependencies? Other extensions?

Need a review of existing (non-RISC-V) compliance specs

Draft Test Coverage Proposal (more, incl priv)

- Forwarding: result of one op can be used as the source of the very next instruction
  - Need at least a case within and between instruction classes
- Changing non-reg state used by an op, immediately followed by op that uses it, e.g. :
  - changing the rounding mode for an FP op
  - writing into the instruction stream, followed by a fencei affecting the next ifetch
  - changing a page table entry or PMP entry, or SATP affecting the next access
  - changing xEPC or xSTATUS followed by xRET
  - changing MISA followed by any op enabled or disabled by it
  - changing xTVEC, xDELEG, xIE followed by a trap
  - write once behavior (PMP-lock)
- Ops that change non-reg status, immediately followed by op that tests it, e.g.:
  - FP status after an FP op
  - xSTATUS.FS,XS fields after FP, Vector or other coprocessor op
  - xCAUSE, xEPC, xTVAL, xPP after an interrupt or exception

RISCV-CONFIG

- Examples & definitions
  - https://github.com/riscv/riscv-config/tree/master/examples
  - https://github.com/riscv/riscv-config/tree/master/riscv_config/schemas
  - https://github.com/riscv/riscv-compliance/tree/master/riscv-ovpsim/config-yaml/examples
- Validator
  - https://github.com/riscv/riscv-config/blob/master/riscv_config/checker.py
- Example integration of converter (OVPsim)
  - https://github.com/riscv/riscv-compliance/tree/master/riscv-ovpsim/config-yaml
- WARL, YAML
  - https://riscv-config.readthedocs.io/en/latest/

RISCV-CONFIG WARL Syntax

WARL: {optional items in curly braces}

- dependency_fields: [list] – use this when legal/illegal values depend on other fields (in list)
- legal: [<warl-string>{,<warl-string>*}]
- wr_illegal: [<warl-string>{,<warl-string>*}] -> update_mode

where <warl-string> is either ”&” separated list of rangehi:rangelo lists

{*[dependency_value] ->*} field-name1[bit#hi:bit#lo] in [legal-range-list]

*{ & field-name2[bit#hi:bit#lo] in [legal-range] }**

or ”&” separated list of bitmasks

{*[dependency_value] ->*} field-name1[bit#hi:bit#lo] bitmask [mask, fixval]

{ & field-name2[bit#hi:bit#lo] bitmask [mask, fixval] }*

(can’t mix ranges and bitmasks)

RISCV-CONFIG WARL Example1

```
#         When base of mtvec depends on the mode field.
WARL:
  dependency_fields: [mtvec::mode]
  legal:
    - "[0] -> base[29:0] in [0x20000000, 0x20004000]”                  # can take only 2 fixed values when mode==0.
    - "[1] -> base[29:6] in [0x00000:0xF00000] & base[5:0] in [0x00]" # 256 byte aligned when mode==1
  wr_illegal:
    - "[0] -> unchanged”
    - "[1] wr_val in [0x2000000:0x4000000] -> 0x2000000”           # predefined value if write value is in this range
    - "[1] wr_val in [0x4000001:0x3FFFFFF] -> unchanged”           # predefined value  if write value is this range

#         When base of mtvec depends on the mode field. Using bitmask instead of range
WARL:
  dependency_fields: [mtvec::mode]
  legal:
    - "[0] -> base[29:0] in          [0x20000000, 0x20004000]"     # can take only 2 fixed values when mode==0.
    - "[1] -> base[29:0] bitmask [0x3FFFFFC0,  0x00000000]”        # 256 byte aligned when mode==1
  wr_illegal:
    - "[0] -> unchanged”                                           # no illegal for bitmask defined legal strings.

”
```

RISCV-CONFIG WARL Example2

```
# no dependencies. Mode field of mtvec can take only 2 legal values using range-descriptor
WARL:
  dependency_fields:
  legal:
    - "mode[1:0] in [0x0:0x1]                          # Range of 0 to 1 (inclusive)"
  wr_illegal:                                          # default to 0 if not a legal value
    - "0x00"

# no dependencies. using single-value-descriptors
WARL:
  dependency_fields:
  legal:
    - "mode[1:0] in [0x0,0x1]                          # also Range of 0 to 1 (inclusive)"
  wr_illegal:
    - "0x00”
```

- "[1] wr_val in [0x2000000:0x4000000] -> 0x2000000 & wr_val in [0x4000001:0x3FFFFFF] -> **unchanged**