Compliance Task Group Call – Minutes

Weds, 27 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
  - Issue status – how do deal with “fixed in riscof” issues? – not discussed
  - Other status (also see progress)
    - Summit Call for Presentations
- Progress:
  - Standard Default Trap Handler
  - RFQ
  - Makefile assertion parameter
  - AUIPC
- Discussion:
  - Moving forward with V.2
  - Coverage…

Action Items from last meeting

Allen:	Standard Default Trap Handler – under review
Stephano:	RFQ – released
Lee:	Makefile assertion parameter - merged
Lee:	AUIPC fix - still under development

Discussion

Updates

Issue status – how do deal with "fixed in riscof" issues? – not discussed

Coverage:

SH gave update. Long coverage email thread - Reference the email thread.
Code coverage of model, vs. functional coverage (which includes values)
Lengthy discussion about "what is coverage" and how do define it, and differentiating it from DV. Example is cross product of src1 x src2 x dest.

Lengthier discussion about usefulness of "code coverage" (of reference model).
Allen: usefulness of SAIL code coverage: run entire suite, examine code coverage - it's a guide to see if corner cases are missing
Imperas: Unless the Sail model labels (or structures) code precisely to determine which configuration a line of code applies to, it's impossible to tell if if code is covered or not - and that is non-trivial
Allen: OK, understood and agreed

Emphasis must be on "functional coverage".
One way to implement functional coverage: post-process trace file to see if all cases covered.
Not a SystemVerilog-specific method. Google does its functional coverage this way.

(general agreement): defining cases for a particular test is also non-trivial...
Do we cover register values? Required min, max, pos, neg (and zero).
Need to investigate other interesting cases wrt to the instruction (e.g. page crossing for Ld/St)

Imperas: We need a spreadsheet that has a row for each of the 46 RV32I instructions, and lists all things that will be checked. Which registers for rs1, rs2, rd? Interesting values for registers. This can then be discussed about what we're going to cover.

Where are we putting documents?
https://lists.riscv.org/g/tech-compliance/files/Review Documents/ folder
Can we put Google docs for the files?
– no, they live in docs.google.com walled garden

RISCOF: Moving forward with V.2:

Python (re: Riscof) Short discussion at start about Python.
Imperas: we avoid python for production code because it is so version dependent and many tool installations require particular versions which may be incompatible with purchased tool installations
Imperas: if we can keep with Python 3.3.0, then we can probably work with Cadence/Mentor/Synopsys.
Riscof: need directed actions as to what is needed.
Updating macros are hard to track - there are bugs in some of them

Makefile assertion parameter:
Pull request for the fix has been done and merged.

AUIPC fix done? (TBD)

RFQ: It's available. Close to what we gave feedback on.
Some changes (e.g. SAIL mode req.)l, but missing a few cosmetic items

2020 Summit: Everyone should have gotten the call for speakers:
https://riscvsummit2020.c4p.tensubs.com/
The plan is in-person, but there is a distinct possibility it may be virtual
Both Allen and Simon are session chairs (of different sessions….)

Decisions & Action Items

Decisions

We cannot rely on model code coverage for coverage

because we can't map implementation options to lines of formal model code)

We need to ensure that riscof is portable across OSes

That may mean backing off Python version (possibly to v3.3)

Action Items

Imperas will produce a coverage CSV in a few weeks,

AB will turn coverage draft spec into CSV

SH and KD, will add files regarding coverage

AB will make sure that formal model can be run in different environments to ensure that RFQ responders can test their code against it

Allen to talk to Stephano to possibly set up google docs folder or define a format that we can insert comments on.

Riscof and Imperas will coordinate to make sure updates match newest spec

Imperas will file a pull request for a fix that has coverage (and possible an issue?)

Everybody: if you have work or product to present, submit to 2020 Risc-V Summit !

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~~	~~closed~~	~~fixed~~
#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~~	~~closed~~	~~fixed~~
#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	
~~#110~~	~~19-May-20~~	~~imperas~~	~~adding make target to define whether assertions are on or off~~	~~Closed~~	~~Pull 112 merged~~

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**