Compliance Task Group Call – Agenda

Weds, Nov13, 2019 8am Pacific →Standard← Time

Charter

The Compliance Task Group will

- Develop a framework for RISC-V tests, 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 method 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 – Stuart Hoad stuart.hoad@microchip.com
- TG membership- Sue Leininger sue@riscv.org
  - Send email to her - you must have a lists.riscv.org login
- TG Email tech-compliance@lists.riscv.org
  - Notetakers: please send emails to allen.baum@esperantotech.com
- Meetings -Bi-monthly at 9am Pacific time on 2nd/4th Wednesdays
  - Location is https://zoom.us/j/6213886723
- 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)

Attendees

• Allen Baum	(Esperanto)
• Lee Moore	(Imperas)
• Bill Mcspadden	(Seagate)
• Greg Wright	(Qualcomm)
• Henrik Gustafsson	(Qualcomm)
• Neel Gala	(IIT Madras)
• Nilesh Asher	(Esperanto)
• S Pawan Kumar	(IIT Madras)
• Sergey Vasiliev	(Syntacore)
• Simon Davidmann	(Imperas)
• Stuart Hoad	(Microchip)

Meeting Agenda (in order of Priority)

1. Pull requests
   - 65: Test Format spec: needs reviewing. Volunteers?
   - 71: Added the ability to select a single test from a suite of tests
     - Approved but pulled for resubmission
   - 75: Added tests and coverage report information, modified makefiles, licenses
2. Summit announcement
3. RISCOF review:

Discussion

1. Pulls
   - 65: affects are large, needs approval vote not just pull reviews (review in off-cycle meeting?)
   - 71: no issues
   - 75:
     - tests added (more C-ext tests needed)
     - test suite restructured to match current Priv spec
     - coverage directory w/results added,
       - Coverage tools checked in yet
       - doesn't affect compliance tests, so OK.
       - Currently on a continuous integration system at Imperas (32i,m,c only)
       - More cover points need to be added over time – do we need more coverage? Cover points
2. Announcement: see slide 11 for possible changes
3. Riscof
   - No one has reviewed it last 2 weeks – not good
   - Next meeting: Bill and Stuart have agreed to try it out
   - Simon:python more error prone; maintainablility is the issue
     - Current framework hass static signatures only
     - Works well for what have –but for multihart, vectors, priv, needs to be dynamic
   - If we can't approve riscof before summit, we will need to modify slide appropriately
   - RV64+RV43: can run both, but if a core supports both, need to add initialization code to ensure you're in the right environment

Decisions & Action Items

Decisions

- Pull 65: review in meeting and ask for approval vote
- Pull 71, 75: merged

Action Items

- Allen: Schedule review and vote of testFormat spec
- Everybody: review pull 65; schedule an off-cycle meeting to review?

Backup from previous discussions

Future Discussions

1. Summit announcement TestFormatSpec (slide 11-12)
2. Coverage metric (slide 13)
3. Email discussion#5 nonconforming extension support
4. WARL definition – (slides 14-16)
   https://riscv-config.readthedocs.io/en/latest/yaml-specs.html#warl-field-restriction-proposal

Foundation Expectations

- Objective: publish compliance test 1.0 and finish the public review **before** the RISC-V summit in Dec. Shorter term is pre-1.0 by EO Q3
- Scope: publish tests and expected results run from the executable RISC-V formal specs -- make sure that all formal specs agree with each other
  - (Note: this approach will not work for priv spec)
- Minimal acceptance criteria is RV32Imc and RV64Imc
- Allen will focus on driving the task group to make this happen
- Nikhil will be tasked to ask all formal spec groups to commit their executable model support in the riscv-compliance repository
- Silviu and Yunsup will make the {compliance manager} CFP happen. They just need to understand what help is needed.

Draft Announcement:
Testing RISC-V Architectural Compliance

- A v0.1 of the RISC-V Compliance Framework is now available here:
  https://github.com/riscv/riscv-compliance/
  - Enables *dynamically* comparing signatures of two arbitrary models
    - e.g. your implementation vs. (e.g. RISC-V CSAIL formal model, ovpsim, spike, etc.)
  - *Can be configured to match your choices of architectural options*
    - *E.g. extensions implemented, HW unaligned access support, CSRs and WARL fields*
  - Currently covers RV32I/RV64 IMC,CSR,FENCEI (unprivileged spec) only
    - With more to come, and with better coverage

Not implemented in current framework; available in riscof

The RISC-V spec allows many (architectural) implementation choices, so

- A new repository has been created to describe implementation configurations that the Framework will use to select & configure tests:
  https://riscv-config.readthedocs.io/en/latest/

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/

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

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 [0x4000001:0x3FFFFFFF] -> 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 [0x4000001:0x3FFFFFFF] -> **unchanged**