Compliance Task Group Call – Minutes

Weds, 22 Apr 2020 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
- Progress:
  - Funding
  - Standard Default Trap Handler
  - ?
- Coverage for YAML-selected/configured tests, Google DV overview

Action Items from last meeting

1. Allen/Neel: start detailing steps needed to pipeclean misalign test, and any test that requires trap handling – prototype has been started
2. Simon: get pointers to Google DV (?) cover points to start ?
3. nothing needed
4. Allen: will attempt to contact jlucnagel to close #27 (need contact info)
5. Simon: will describe (& give a demonstration)? in 2 weeks (combine with #2?) (described, but no demo this week)

Discussion

- Issues
  - Issue#104 – fix fails because of extraneous blank line in reference (fixed)
  - Issue #105 LA is supposed to be a symbol (GCC accepts a constant)
    working with customer about compliance tests. Discovered Gcc v. Clang differences
    need to support both, so use lowest common denominator.
  - Issue #106 general use of pseudo instructions: Should we? Assembler will substitute compressed ops if compressed is available (& is required for RV64 platforms) –can it be turned off? (yes)
    But should divide test care about using LA? Clearly AUIPC test should
    Trap handler will expect specific alignments, can't allow substitutions/optimizations
  - Issue #107 naming of CSRs: old names are still in some tests, need to fix: SPTBR used instead of updated SATP in riscv_test.h
  - Issue #108 fix for #105 specifically used in implementation specific code
- Introduction: Uzi and Tom from IBM Research in Haifa, Israel. pre/post Si verification tools. Would like to contribute to the compliance suite effort. Their group deals with Tools for pre/post validation
- Funding for getting tests written:
  - Stephano: will be taken up at TSC next Tuesday (probably). Need a well-defined "statement of done". Calista (per Stephano) says funding should be available for this effort.
  - Need a description of what the value of " writing tests" vs. Statement of Work e.g.
    "Why is a compliance suite a good idea?
    No, looking for not just value, but scope
  - What about priv architecture?
    Priorities are unprivileged first (including RV64), then privileged
    Priv likely needs new framework (YAML description)
- Progress: Discussion of proposed default interrupt handler(s) (one per mode)
  - Stub that saves trap signature into separate area of signature block & returns
  - Signature: Vector+mode, xCAUSE, xEPC*, xMTVAL*, xIP**, intID***
    (*exceptions only, adjusted for loader values, **ints only, *** ext ints only)
  - Ints dismiss the interrupt; traps return past trap point
  - TBD: enabling Ecalls inside handler(e.g. to access mtime)
- Progress: riscof CSR tests - now takes YAML into account for RW tests.
- Presentation of YAML config (Simon)
  - Part1: risvOVPsim: configurable model donated by Imperas
  - Part2: riscv-config repository: (Shakti contributed): describes Risc-V options in YAML
  - Part3 trace->google cloud ->coverage (Google/Imperas)
  - Part4: riscvOVPsim now produces functional coverage report, and uses YAML directly

  Discussion of the difference between compliance and DV. (in response to question/observation by Uzi)

  Discussion of functional coverage based on test selector and parts-ot-test "selector". How do you measure coverage of tests when individual test cases are optionally executed (because of optional features being implemented or not) or configured differently (e.g. which bits of a CSR are writable).

  Must report tests individually. Which means that each individually selected part must have its own coverage metric. This is work….

Decisions & Action Items

Decisions

Action Items

Allen & Jeremy: add upstreamed gcc link into (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"

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	

Next Meeting Agenda (proposed, not final)

- Next steps
  - More tests? (and priorities)
    - Better coverage of existing tests vs. new tests (primarily priv level)
    - E.g. RV32i->rv64i -> ratified extensions -> priv modes
  - Coverage metrics? More detail on how to report cover of partially executed tests for configurable options
  - Transitioning to v.2?
  - Funding?
  - Get more repo maintainers
  - Other?

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**