Compliance Task Group Call – Minutes

Weds, Oct 09, 2019 8am Pacific →Daylight← Time

See slides 6-7 for summary

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
- 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)
- Grant Martin (Cadence)
- Neel Gala (IIT Madras)
- Simon Davidmann (Imperas)
- Stuart Hoad (Microchip)
- Bill Mcspadden (Seagate)
- Lee Moore (Imperas)
- Paul Donahue (Ventana)

Meeting Agenda (in order of Priority)

1. Pull requests
2. Status: RISCOF status, Imperas, Google riscv_dv {slide 6,8}
3. Foundation expectations {see slide 9-10}
4. Email discussion#5 nonconforming extension support
   - continue discussion – but not now
   - Reference model support for YAML
5. WARL syntax, RISC_CONFIG ,{see slide 12-16}
   https://github.com/neelgala/riscv-config/tree/master/warl_proposal

Discussion

1. Ibex pull request - accepted
2. Status updates:
   - RISCOF (more detail in slide 8)
     - Now 3 repositories: Framework(with tests), configurator, targets
     - Only doc directory is missing (TestSpec will need updating to match newest structure and macros)
     - Google is happy with YAML configuration spec
     - models integrated: 3 of 6 formal models now in target repo Csail, ovpsim, & spike working,
     - targets integrated : SiFive fails Verilator compile, Ibex & ri5cy too new
     - google is happy with YAML structure
   - Imperas has tested coverage of current tests, is working with google for better measurements
     - Working with google on UVM testbench coverage – easy to add coverage points as coverage spec is created
     - Current test suites have ppor coverage (based on that)
     - Google will open source their coverage reporting
     - Has written new 32i tests with improved coverage based on this, not in repository yet

Coverage: how to define address coverage?

by effective address bit flipping (so entire platform defined address range should be covered?
or by offset bit flipping?

Similar issue for branch addresses

Discussion was that effective address coverage was veering into verification rather than compliance
offset coverage was doable (e.g flip bits in both base and offset to keep effective address range small)
(unclear how to handle branches since base is PC….)

Conclusions & Action Items

Decisions

- Next meeting will be a live demo of riscof
- Following meeting will be a feedback session

Action Items

- Allen will try again check with formal modelling group to see if they can (easily) define trapping behaviour.
- Neel et. Al: Prepare a live demo for next meeting
- Everybody: if there is a specific thing you want to see in the demo, please send it to Neel
- Everybody: play with riscof !
- Allen will resurrect his coverage document
  (see slide XXX)

RISCOF Status Detail

Targets and framework (incl. tests) are now in separate repositories.

RISCOF: https://gitlab.com/incoresemi/riscof . YAML based framework, test-pool and relevant docs.

RISCOF-Targets: https://gitlab.com/incoresemi/riscof-plugins plugins for various targets

Differences from current github riscv-compliance and riscof:

- Test preamble significantly minimized. (see here)
- Preamble can add custom boot-code/libraries, etc through RVMODEL_BOOT macro
- No reference signature: obly comparison between golden and DUT models
- RISCOF generates 2 separate elfs for golden DUT models.
  - differ only in MODEL specific features : boot-code, termination sequence, etc. Tests not modified for either of the models.
- RISCOF allows parallel execution of the leveraging pmake (a.k.a make -j<jobs
- RISCOF uses the new directory structure
- RISCOF automatically generates and maintains a YAML database of all the tests

Merging RISCOF to riscv-compliance on github:

- RISCOF now is a complete python framework available as a pypi project: here.
  Repo gets shipped as a complete package, installed via **riscof pypi** as a command line tool
  cloning/updating the repo not nee whenand building tool.
- Should only require adding the current docs (readme, license, testpec-format) to RISCOF docs folder
- riscof-plugins/targets will continue to remain separate from RISCOF
- Sifive is missing (issues with verilator compilation)
- ibex, ri5cy are missing– only added recently.

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.

RFP for Compliance Engineer

There is quite a bit of work involved with getting the compliance test suite repository in shape for an actual public release for RV32i and RV64i (with a slight stretch goal of A and M extensions. The foundation is looking at getting some funding to get that accomplished.
The Compliance TG (that is us) is responsible for coming up with the tasks that will need to be performed for this.

An initial stake in the ground:

Prerequisites: We have a documented

- test spec format
- coverage metrics – this is the gating item, in my opinion

The compliance engineer will

- restructure the repository to meet the structure documented in the test spec format
  RISCOF has done this. Please check here
- convert tests to use the macros documented in the test spec format
  This too is available here
- modify any other collateral to match the new structure if necessary
  RISCOF removed un-necessary preambles from riscv_test.h. , and ported spike, vovpsim and one shakti-core within RISCOF.
  Migration of current RISC-V targets on github to RISCOF should have happened
- Provide documentation and an example of how to download the respository and run the compliance framework
  RISCOF has documentation here ; Feedback is appreciated.
- add (to) tests to meet coverage metrics (gated by coverage spec)

Proposal is to use RISCOF as the starting base

2Bi. Instruction Test Coverage

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	

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

- 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
  - https://riscv-config.readthedocs.io/en/latest/yaml-specs.html
- YAML Rev2 CSR definitions & WARL proposal
  - https://github.com/neelgala/riscv-config/tree/master/warl_proposal

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: [list of warl-string]
- wr_illegal: [list of warl-string] -> update_mode

where warl-string is:

{*[dependency_value] ->}* field-name[bit#hi:bit#lo] in [legal-values]

*{ & field-name[bit#hi:bit#lo] in [legal-values] }**

or

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

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

RISCV-CONFIG WARL Syntax2

```
#         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 Syntax2

```
# 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**

Backup for previous discussions