Compliance Task Group Call – Minutes

Weds, Oct 23, 2019 8am Pacific →Daylight← 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)
- Neel Gala (IIT Madras)
- Bill Mcspadden (Seagate)
- Eyck Jentzsch (Minres)
- Henrik Gustafsson (Qualcomm)
- Lee Moore (Imperas)
- Ken Dockser (Qualcomm
- Ondrej Peroutka (Codasip)
- Premysl Vaclavik (Codasip)
- Alan Saw (Codasip)
- Simon Davidmann (Imperas)
- S Pawan Kumar (IIT Madras)
- Stuart Hoad (Microchip)
- Ilja Stepanov (Syntacore)

Meeting Agenda (in order of Priority)

1. Pull requests
2. New coverage status
3. Status: RISCOF demo:

    1. Installation and setup.
    2. Directory Structure.
    3. Necessary structure of tests i.e presence of specific macros and compatibility with sig-gen
    4. Dbgen working explanation and advantages.
    5. Directives - (Need and Working)
    6. Demonstration and Walkthrough of dbgen.
    7. Demo of Addition of test to riscof
    8. riscv-config integration and need.
    9. Preamble structure and model specific code possibilities.
    10. Plugin model explanation and walkthrough.
    11. Flexibility and custom support with plugins.
    12. Riscof working brief and changes since the last proposal.
    13. Riscof demonstration and walkthrough.
    14. Report generation explanation.
    15. Further improvements possible.

Discussion

1. Pull req accepted: adding params to sim
2. Coverage
   - latest results presented
   - google tools ties to UVM testbench, more tests coming
3. Riscof

- Plugins: DUT and Ref, assertion and req. YAML (platform
- Auto gen of db, more defaults, more examples, sanity and coverage, makefile gen, make plugin creation
- Can stop at various points: stop after yaml gen, etc
- Use pyenv python 3.7, pip/pip3
- Plugins now in separate repositories
- Gitclone or pip version – database not available in pip version!, so use git version otherwise – sphinx has docs
- Database gen is manual, moving forward more automated, does sanity checks
  - If no condition, what happens? (always runs)
  - Would it be easier to use previous condition?
- Plugins: needs to provide input YAML, set of routines to init, build, sim DUT as python func,
  choose SDK, compiler, etc, all in python
- "—setup" option generates template including default macro template, sample linker,

Decisions & Action Items

Decisions

Action Items

- Next meeting will discuss experiences of using the framework
- **Everyone should review presentation and try out the framework (Ideally with their implementation)**

New Coverage Status

test	covergrp	old	new
Bxx	branch_hit	100	---
ADDI,ANDI,JAL,SLTIx,XORI	imm_sign	100	---
Bxx,JALR,LW,OrI		50	unch
LBx,LHx		50	unch
LBx,LUI,SLTx	rd	40-97	97
JALx		6-9	unch
CSRx, AUIPC		22-60	unch
JALR	rd_link	50	Unch
ADDX,ANDX,AUIPC,LUI,Orx, SLLx,SRAx,SUB,XOR	rd_ sign	100	---
SLTxx	result	100	---

test	covergrp	old	new
ADDx,ANDx,Bxx,Orx,SLTx, SLLx,SRx,SUB,XORx	rs1	31-69	97
CSRRx		31-53	unch
JALR	rs1_link	50	unch
ADDx,ANDx,Bxx,Orx,SLTx, SLLx,SRx,SUB,XORx	rs1_sign	100	unch
ADDx,ANDx,Bxx,Orx,SLTx, SLLx,SRx,SUB,XORx	rs2	81	97
SLL,SRx	rs2_sign	50	unch
ADDx,ANDx,Bxx,Orx,SLTx, SUB,XORx		100	unch

Future Discussions

1. TestFormatSpec
2. Coverage metric
3. Email discussion#5 nonconforming extension support
4. WARL definition

Backup for previous discussions

Previous: RISCOF Status

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.
  Rrepo 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.

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

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.

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**