Compliance WG
July 19 Meeting

Agenda:

Pull together a bullet list requirements / deliverables we'd like a dev resource be responsible for.

Don't worry about format / structure or anything - a raw list is fine.

Input will be used to create a draft RFP we can use to solicit proposals for this work.

Attendees

- Allen (Esperanto) [Chair]
- SimonD (Imperas)
- Lee Moore (Imperas)
- Ben Selfridge (Galois)
- Dmitri (Syntacore)
- Neel Gala (IIT Madras)
- Radek (Codasip)
- Stuart Hoad (Microsemi)
- Ken Docker (Qualcomm)
- Jeremy Bennett (Embecosm)
- Ramprasad Chandrasekaran (Xtreme-EDA)

Progress

- Imperas has updated all the RV32I tests to use the macros to show detailed logging and assertions
  - they now adhere to the current compliance test framework format

- Imperas has added a suite of RV32IM tests and their reference signatures
  - so there are now 2 compliance suites RV32I and RV32IM
  - both fully adhering to the specified format
- All checked into the groups github –
  - all can be downloaded and used

Discussion: What do we need them to do?

- Use Codasip approach of instruction by instruction testing as a starting point
  - They are not exhaustive – e.g doesn't check intermediate values
  - Priv spec is more challenging
- We can specify boundary conditions regarding how to generate test values
  - Look at superoptimization techniques and Plackett-Burnham technique?
- Use list on github riscv/riscv-compliance/doc/#the-test-suitesfor guidlines
- What is compliance? We don't want a verification test!
  - Representative subset ?
  - Or more exhaustive
- **Group needs to specify what a test should look like – structure and guidelines**
  - We have examples now, but not a philosophy of how test should work
  - Current approach is multi-instruction sequence that doesn't test intermediate results – is that OK?
- Need coverage in golden model to ensure test suite completeness
  - Galois golden model has coverage reporting built in, waiting for DARPA approval to release
  - Imperas golden model also has coverage reporting built in
  - E.g source instruction coverage, value coverage, spec coverage, reg permutations (ref to what Verisity did for terminology),
  - Can we use other reference models that report coverage of the test suite e.g. github.com/jerralph/riscv-vip ?
    - Reports value min/max ranges, and all bits flipped
  - → **But we want coverage of spec with respect to test, <u>not</u> coverage of test with respect to HW** ←
- See https://docs.google.com/document/d/1mDOjdz2wy7aI9-vBB6nm-aPVm4O6TPSqqX7-0HwFaj8/edit
- Should we Add torture tests, converting them to the appropriate format?
- Should we propose a compliance process, not just tests, and submit to Foundation?
  - Do rules or a process exist, beyond paying dues to the foundation?
  - Nothing written down yet

Answers, short:

- Who: Foundation Decides
  - But we will provide suggested requirements, see "**Who**" slide
- What: We provide guidance
  - See "**What**" slide
- Where: N/A
- When: Foundation decides
  - But we need to send them something in the next week or so
- How: We will provide guidelines
  - See "**How**" slide
- How much: Foundation decides
  - But we will provide suggested amount
- Why: because vendors are claiming compliance, when they aren't

Conclusion

- We want someone to write tests
  - Start with list in https://github.com/riscv/riscv-compliance/tree/master/doc#the-test-suites
  - Jeremy will help Allen to modify it appropriately
  - Add a separate document that describes test structure and guidelines, see https://docs.google.com/document/d/1mDOjdz2wy7aI9-vBB6nm-aPVm4O6TPSqqX7-0HwFaj8/edit#
  - Maybe the can re-code torture tests for the compliance framework?
  - If we can figure out how to describe it, tools for test generation are also valuable
- To evaluate these tests, formal models need to track coverage
  - Specifically, this tests how much of the spec is being tested by the test suite
  - Not how much of an implementation is being exercised!
- We need guidance on compliance testing from Foundation
  - E.g. need to see the licensing docs, since there may be dependencies affecting us
  - Many current implementations say they are RISCV, but are not compliant!
  - Currently, being a Foundation member seems to be the only requirement
  - The compliance group should propose a process to the Foundation

Who (from email)

We need someone with:

- expertise in
  - embedded processor instruction set architectures
  - language: bash, python, linux, C, assembler
  - tools: make, gcc, gdb, github
  - running leading commercial verilog simulators
  - usage of ISS and IA simulators
- Has some knowledge of verilog RTL design verification
- Ideally has working experience on processor, IP, RTL, testing, models, simulators
- Should be available soonish
- Should have access to development environment, linux pc
- Budget initially a project of ~ $100k for some 3-4 months work (50-80 days), not necessarily full time, -
  - Probably no need to be a budget for tools as commercial vendors would lend them.
- Assume that this working group would guide and specify project and monitor and manage it

What: (from Github riscv-compliance)

- Tests are grouped into different functional test suites targeting the different subsets of the full RISC-V specifications. There will be ISA and privilege suites.
- Currently there is one test suite: the RV32I (developed by Codasip).
- Test suites will be developed in this priority order:
  - RV32I
  - RV64I
  - RV32IM
  - RV64IM
  - RV32IC
  - RV64IC
  - RV32IA
  - RV64IA
  - RV32IF
  - RV64IF
  - RV32ID
  - RV64ID
  - RV32E
  - RV32EC
  - RV32EA
  - RV32EF
  - RV32ED
- Eventually Priv Spec, adding U , N, and S, w/ ALL combinations of optional features

How: Test Requirements (from Google Docs)

- *Instruction decode – basic functionality to ensure that the instruction is properly decoded*
  - *Make sure that each instruction bit that can vary is exercised and that it has a visible effect*
    - *Use each register for each input and output (don't need to cover all permutations)*
    - *Immediate values: Use various values so that each bits have been hi and low*
      - *Vary combinations of bits to ensure bits given correct weight*
    - *FP: Apply each rounding mode where available within the instruction*
  - *Handling special registers (e.g., X0) for source and/or destination*
  - *Register combinations – various legal combinations where the same register is used for multiple sources and/or destinations.*
  - *?? Need to specify reference assembler or assembled code?*
- *Instruction behavior – ensure the instruction executes as defined*
  - *Exercise all combinations of input types for floating point (e.g., infinity, Nan, subnormal, zero, normal etc.)*
  - *Exercise inputs producing corner cases (e.g., overflow, just before overflow, etc.)*
  - *Exercise cases producing all the floating-point exceptions possible for each instruction, from a variety of approaches*
    - *In FP: e.g., due to round, carry propagation w/o round, and underflow*
    - *In Vector: e.g., exceptions on one lane at a time*
  - *Ordering corner cases (e.g., vector reduction with operands chosen to give different results based on the order)*
  - *Machine state - Cover all cases of machine state (e.g., CSR bits) that impacts instruction behavior*
    - *Rounding and clip modes for fixed point (all combinations)*
    - *Rounding modes for floating point*
    - *Others (e.g., vector configurations)*
- *Some sections will need more extensive tests (e.g., floating point, vector) than others (e.g., basic integer)*