Compliance Methodology and Initial Results for RISC-V ISA Implementations

Lee Moore, Simon Davidmann, Larry Lapides
Imperas Software Ltd.
Agenda – RISC-V Compliance

• Compliance for RISC-V is extremely important
• Compliance is a thankless job that no one wants to do
• RISC-V Foundation’s compliance suite
• Experiences when checking compliance on user designs
• Summary
Agenda – RISC-V Compliance

• Compliance for RISC-V is extremely important
• Compliance is a thankless job that no one wants to do
• RISC-V Foundation’s compliance suite
• Experiences when checking compliance on user designs
• Summary
Compliance for RISC-V is important

Q: What is meant by “compliance”?  
A: The device works within the envelope of the agreed specifications

Q: Is there an easy process or path to follow to develop methodologies/tools for compliance?  
A: NO – all established ISAs are single company controlled and those companies work extremely hard on proprietary solutions to ensure that all designs that go out their door work correctly – so RISC-V has to pioneer compliance collectively

Q: What happens if the RISC-V industry builds devices that are not complying with specifications?  
A: Users cannot assume that tools like C compilers, operating systems, and application software will be transferable across devices and work correctly
Non-Compliance

• Non-compliance makes it very hard or impossible to benefit from the developing RISC-V ecosystem
  • Won’t be able to use GCC, LLVM, GDB, etc...

• Fragmentation is the result of non-compliance and would be a disaster for the RISC-V industry as it leads to incompatible processors
  • Locked in to one CPU, software not portable across different chips / boards

• If you build a non-compliant processor or simulator or tool
  • => IT IS NOT RISC-V
• Compliance for RISC-V is extremely important

• Compliance is a thankless job that no one wants to do

• RISC-V Foundation’s compliance suite

• Experiences when checking compliance on user designs

• Summary
Compliance is a thankless job that no one wants to do

- Compliance is the counterbalance to customization
- Customization is one of the key features of the RISC-V ISA, it is exciting!
- Not so for compliance; it is hard work, and your reward is to inform other people of mistakes in their work
- Plus, no one has had to do compliance testing for other ISAs
  - Previously all compliance testing done by the ISA vendors for closed ISAs
  - There is minimal industry knowledge of how best to do compliance
Compliance is a thankless job that no one wants to do

- The companies that do understand compliance do not (cannot?) share their test suites.
- Their tests are too closely bound to verification, and donating those tests would expose proprietary processor implementation details that provide product (processor IP) differentiation.
- The companies that most want compliance – SoC users, the system companies – are not yet a large part of the RISC-V community.
- The software ecosystem companies – software tools, software IP, OS – also want compliance.
  - Compliance enables leveraging of the investment in RISC-V.
  - Biggest contributors to compliance efforts to date are in this area.
Agenda – RISC-V Compliance

• Compliance for RISC-V is extremely important
• Compliance is a thankless job that no one wants to do
• RISC-V Foundation’s compliance suite
  • Short history
  • Current status
• Experiences when checking compliance on user designs
• Summary
The RISC-V Foundation’s Compliance Task Group

• Jun 2017 group started
  • Compliance testing is a testing technique which is done to validate whether the system developed meets the prescribed standards or not. It is **not design verification** testing
    • Compliance testing is looking for issues like missing registers, modes, instructions – not for bugs in RTL implementations...
  • The tests have to be written to ensure compliance/non-compliance is **observable in a test signature**. The signatures are published so that the user does not have to run a reference model and can compare the results of their target runs to the reference signature

• Jan 2018 initial rv32i test suite provided by Codasip
• Jun 2018 Imperas, Embecosm worked on GitHub, made repo public
• Oct 2018 Imperas improved test coverage, added new suites, ported 32bit riscv-tests
RISC-V Compliance Suite

• It is ‘work in progress’
• Two components
  • Test suites
    • Each suite focuses on a feature set of the RISC-V envelope
    • Initial focus is instructions, user mode spec, e.g. rv32i, rv32im, rv32imc, rv64i, ...
    • Awaiting RISC-V platform specifications to subset privilege spec, before starting privilege suites
  • Framework
    • Make, bash, and scripts
      • Encapsulate compiler tools, linkers, simulators, and targets (Devices Under Test)
    • Includes simulator: as example target, and to generate reference signatures
    • Run: Select suite and target
      • Runs each test, target produces signatures, compares to saved golden reference signature
• Available: www.github.com/riscv/riscv-compliance
Compliance Suite Status (rv32i)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Decode Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>lb</td>
<td>100%</td>
</tr>
<tr>
<td>lh</td>
<td>100%</td>
</tr>
<tr>
<td>lv</td>
<td>100%</td>
</tr>
<tr>
<td>lbu</td>
<td>100%</td>
</tr>
<tr>
<td>lhu</td>
<td>100%</td>
</tr>
<tr>
<td>sb</td>
<td>100%</td>
</tr>
<tr>
<td>sh</td>
<td>100%</td>
</tr>
<tr>
<td>sw</td>
<td>100%</td>
</tr>
<tr>
<td>sl</td>
<td>100%</td>
</tr>
<tr>
<td>sli</td>
<td>100%</td>
</tr>
<tr>
<td>srl</td>
<td>100%</td>
</tr>
<tr>
<td>srl1</td>
<td>100%</td>
</tr>
<tr>
<td>sra</td>
<td>100%</td>
</tr>
<tr>
<td>srai</td>
<td>100%</td>
</tr>
<tr>
<td>add</td>
<td>100%</td>
</tr>
<tr>
<td>addi</td>
<td>100%</td>
</tr>
<tr>
<td>sub</td>
<td>100%</td>
</tr>
<tr>
<td>lui</td>
<td>100%</td>
</tr>
<tr>
<td>auipc</td>
<td>100%</td>
</tr>
<tr>
<td>xor</td>
<td>100%</td>
</tr>
<tr>
<td>xori</td>
<td>100%</td>
</tr>
<tr>
<td>or</td>
<td>100%</td>
</tr>
<tr>
<td>ori</td>
<td>100%</td>
</tr>
<tr>
<td>and</td>
<td>100%</td>
</tr>
<tr>
<td>andi</td>
<td>100%</td>
</tr>
<tr>
<td>slt</td>
<td>100%</td>
</tr>
<tr>
<td>sltu</td>
<td>100%</td>
</tr>
<tr>
<td>j</td>
<td>100%</td>
</tr>
<tr>
<td>jr</td>
<td>100%</td>
</tr>
<tr>
<td>mv</td>
<td>100%</td>
</tr>
<tr>
<td>beq</td>
<td>95%</td>
</tr>
<tr>
<td>bne</td>
<td>95%</td>
</tr>
<tr>
<td>blt</td>
<td>95%</td>
</tr>
<tr>
<td>bge</td>
<td>95%</td>
</tr>
<tr>
<td>bltu</td>
<td>95%</td>
</tr>
<tr>
<td>bgeu</td>
<td>95%</td>
</tr>
<tr>
<td>slti</td>
<td>77%</td>
</tr>
<tr>
<td>sltiu</td>
<td>77%</td>
</tr>
<tr>
<td>fence</td>
<td>0%</td>
</tr>
<tr>
<td>fence.i</td>
<td>0%</td>
</tr>
<tr>
<td>scall</td>
<td>0%</td>
</tr>
<tr>
<td>sbreak</td>
<td>0%</td>
</tr>
<tr>
<td>rdcycle</td>
<td>0%</td>
</tr>
<tr>
<td>rdcycleh</td>
<td>0%</td>
</tr>
<tr>
<td>rdtimem</td>
<td>0%</td>
</tr>
<tr>
<td>rdtimem</td>
<td>0%</td>
</tr>
<tr>
<td>rdnstret</td>
<td>0%</td>
</tr>
<tr>
<td>rdinstreth</td>
<td>0%</td>
</tr>
</tbody>
</table>

Notes on rv32i test suite:
- 54 hand coded directed tests (average 150 instructions each)

Notes on Decode Coverage:
- Decode Coverage: observe changes on all bits of legal decodes
- Decode Coverage data from the Imperas Fault Simulation Coverage tool
  - ran 478,390 simulations in 308 secs

Need more tests
Investigate (more tests?)
no tests yet (fence.i not in 2.3 RV32I)
pseudos+CSR, no tests yet (not in 2.3 RV32I)
Coverage Metrics for Compliance Tests

- Coverage metrics used in RTL design verification are *not applicable* as they are often functional and connected to specific microarchitecture (RTL).
- Imperas’ Code Coverage Tool provides coverage of model source, which is useful to see how much of model is exercised by tests but *does not* show how much of specification is covered.
- Imperas’ Fault Simulation Coverage tool provides *Instruction Decode Coverage*:
  - explores the decodes of the instructions and mutates legal bits
  - detects that there is a test that stimulates & observes each bit
  - tool is very fast, runs in parallel, provides other metrics including data coverage
  - Also valuable outside of compliance as *provides coverage of custom instructions*
Test Suite Status

- https://github.com/riscv/riscv-compliance/tree/master/riscv-test-suite

Currently there are twelve test suites checked into this repository.

If you are looking to check compliance of RV32I in user mode then run the suites: RV32I and RV32UI

Test suites status:

Pretty Solid:

- RV32I (originally developed by Codasip, updated significantly by Imperas to improve coverage)
  - 54 focused tests, using the correct style/macros, excellent coverage of most instructions
  - no coverage of fence, scall, sbreak, pseudo and csr instructions
- RV32IM (developed by Imperas)
  - 7 focused tests, using the correct style/macros, excellent coverage
- RV32IMC (developed by Imperas)
  - 24 focused tests, using the correct style/macros
- RV64I (developed by Imperas)
  - 8 focused tests, using the correct style/macros
- RV64IM (developed by Imperas)
  - 3 focused tests, using the correct style/macros

Work in progress (user mode):

- RV32UI (from github/riscv-tests with poor coverage, Ported by Imperas)
Challenges for writing tests for RISC-V and compliance suite developers

• How to write the tests
  • Need to white box walk through to examine detailed operation

• How to explore when things don’t go as planned

• Requirements on a simulator to assist with test development
  • Needs full implementation of specifications
  • Needs full configurability to all options of specification
    • Envelopes of legal subset choices
  • Full trace to see all resources affected by all instructions
  • Need access to comprehensive debug
  • Has to be able to be encapsulated, and controlled in external environment
  • ...

© Imperas Software Ltd.
riscvOVPsim simulator

• Instruction Accurate simulator using high performance Just-in-Time Code Morphing that executes RISC-V binaries and runs on Linux/Windows PC
  • Runs very fast, 1,000 MIPS
• Industrial quality for use in test development, software development, compliance testing and design verification
  • Includes capabilities to perform RISC-V compliance signature generation
• Maintained and supported by Imperas Software (www.imperas.com)
• Simulator restricted to single processor model and fully populated memory
  • Loads cross compiled ELF binaries
  • Uses semi-hosting to provide easy access to host I/O
  • Single executable (includes model, memory, platform, simulator, tools)
• Includes Apache 2.0 Open Source model of complete RISC-V ISA envelope model of 2.2, 2.3, 1.10, 1.11 revisions of the RISC-V Foundation specifications (not hypervisor/vectors are not stable)
• FREE. No license keys or license management
Issues to be sorted by RISC-V Foundation’s Compliance Working Group

• Detailed specifications needed
  • the structure of the Compliance Suite Framework (and how used)
  • what content of test should be (how written)
  • list of all the tests needed in each test suite (what is covered of specification)

• Formal process for RISC-V Foundation to allow user to claim their processor is a RISC-V processor
  • i.e. process to explore and record ‘is it a RISC-V compliant design?’
  • the ‘certification stamp: RISC-V inside’
Users of the Test Suites

• Very hard to really know usage of the tests suites from GitHub
  • 46 unique cloners (download, use)
  • 23 forks of repo (download, use & make changes)

• From questions on GitHub & Task Group & emails these people are (public) users:
  • Tools/Service:
    • Imperas
    • Embecosm
    • Bluespec
    • Galois
    • Valtrix
    • XtremeEDA Corporation
  • Processors:
    • Andes
    • Microsemi
    • ETH (PULP)
    • Incore (Shakti)
    • Codasip
    • Syntacore
  • Groups:
    • Formal TG with Forvis

• Plus all the RISC-V Summit 2018 soft-core competitors...

• If you are a user – let us know - send an email to Simon Davidmann: simond@imperas.com
STING is a design verification tool developed by Valtrix Technologies for RISC-V based CPU and SoC implementations.

STING generates architecturally correct random, directed and control graph flow based instruction sequences for testing the functionality and architectural compliance of the target implementation.

STING implements a checking mechanism where the results from running a test on DUT and a functionally accurate ISA simulator are compared.

Imperas’ riscvOVPsim is used in the checking flow for its superior ISA modelling, verification & configuration capabilities.

Example of Design Verification Use-case: Using riscvOVPsim with Valtrix STING Instruction Generator

1. Build STING image
2. Run on riscvOVPsim
3. Collect the results of the test; Populate checkpoints; Record events
4. Rebuild the image with the reference results embedded into it
5. Run the rebuilt image on DUT with checking enabled. Any mismatch in test results between DUT and riscvOVPsim reported and debugged

Reference Results

STING ELF

DUT

Model DUT/CPU Capabilities in STING Configurations

© Imperas Software Ltd. 26-Feb-19
Agenda – RISC-V Compliance

• Compliance for RISC-V is extremely important
• Compliance is a thankless job that no one wants to do
• RISC-V Foundation’s compliance suite

• Compliance Experience
  • Imperas experience of examining compliance on various designs
  • Examples of Fails...
  • Recommendations

• Summary
Imperas: Exploring designs

Note: these notes are Imperas exploring designs brought into Imperas, not a discussion of what our customers are doing with our tools & test suites

• Many candidate DUTs (Device-Under-Test) = customers, partners, & for fun...
  • proprietary RTL, open source RTL
  • FPGA
  • Silicon
  • Simulators

• Process – goal is to load .elf file, run, write signature, compare with golden reference

• Develop encapsulation of DUT
  • so can run public GitHub Compliance Suite – and – Imperas internal compliance test suites

• Different levels of complexity and challenge
  • ISA Simulators relatively easy (as can read/write files)
  • RTL simulators a little harder, 2 approaches
    • Memory read/write
    • GDBserver – control as if hardware, inject, extract data
  • Hardware (FPGA, silicon)
    • Pod, JTAG, GDBserver type approaches to get data in/out

© Imperas Software Ltd.
• Over 10+ years, Imperas has learnt how to test processors/models
  • Expertise in 12+ ISAs: x86, Arm, MIPS, ARC, Renesas, Power, Xilinx, Altera, RISC-V, ...
  • Developed testing approach with sophisticated regression test environment
    • Tests: directed white box, litmus, random, constrained random, benchmark, OS, ...
• Instructions are easy, the issues are always with the ‘privilege’ modes
• Imperas customers can license access to Imperas internal tests (inc. RISC-V suite)

• Built & productized tools to assist
  • Configurable simulator
    • so very easy to get models into specific correct states (hard with virtual memory and muti-processor)
    • control & visibility for 3rd party execution validation (RTL, simulator, ...)
  • Model code coverage (source)
  • Instruction coverage (functionality coverage)
  • Instruction decode coverage (test coverage)
  • Instruction profiling (simulation speed)
  • Methodology for user custom instructions as well as standard instructions
What did we find...

• Many candidate DUTs (Device-Under-Test) = customers, partners, & for fun...
  • proprietary RTL, open source RTL
  • FPGA
  • Silicon
  • Simulators
What did we find...

- Many candidate DUTs (Device-Under-Test) = customers, partners, & for fun...
  - proprietary RTL, open source RTL
  - FPGA
  - Silicon
  - Simulators
- Missing registers
- Missing instructions
- Floating point mode change issues
- PMP implementation issues
- ...

© Imperas Software Ltd.
Recommendations for compliance checking

Note: this is for compliance checking, not design verification...

• Clone www.GitHub.com/riscv/riscv-compliance

• Encapsulate your DUT target (in your cloned local version of test suite)
  • Use as starting point one of the example targets that are being added

• Get the initial rv32i suite running (and comparing to golden references)
  • add others are they are available and as appropriate

• Run & check whenever you modify your design (add to your regression suite)

• Update your clone/fork and run & check when there are changes to www.GitHub.com/riscv/riscv-compliance
• Compliance for RISC-V is extremely important
• Compliance is a thankless job that no one wants to do
• RISC-V Foundation’s compliance suite

• Compliance Experience
  • Imperas experience of examining compliance on various designs
  • Examples of Fails...
  • Recommendations

• Summary
Summary

• Compliance – it is a work in progress
  • Initial suites & framework can be used NOW
    • Users are telling us the tests are finding issues for them
  • Ongoing work on RV32 A, F, D user mode tests
  • Need to start compliance suite tests of privilege mode & platforms
  • New ISA extensions (e.g. Vector, BitManip, DSP, …) will need to add compliance test suites

• Understanding & adoption has started

• Need to consider compliance as fundamental part of methodology
  • Continuously...
Thanks

• More Information:
  • www.imperas.com/riscv
  • www.OVPworld.org/riscv
  • www.github.com/riscv/riscv-compliance
  • simond@imperas.com
  • moore@imperas.com