Compliance, Verification and Customization of RISC-V Cores and SoCs

Getting Started with RISC-V – North America Roadshow
Waltham (Boston) – Austin – Irvine – Milpitas (Bay Area) : April 2019

© Imperas Software Ltd.
“nobody designs a chip without simulation”,
at Imperas we believe that:
“nobody should develop embedded software without simulation”

Imperas develops simulators, tools, debuggers, modelling technology, and models to help embedded systems developers get their software running...
  • and hardware developers get their designs correct

10+ years, self funded, profitable, UK based, team with much EDA (simulators, verification), processors, and embedded experience

www.imperas.com/riscv
Agenda

• Compliance and verification
• Customization
• Summary
Agenda

• Compliance and verification
  • Why is compliance important for RISC-V?
  • RISC-V compliance checking
  • Verification

• Customization

• 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
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)

Notes on rv32i test suite:
• 54 hand coded directed tests (average 150 instructions each)
  • https://github.com/riscv/riscv-compliance/tree/master/riscv-test-suite/rv32i/src

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

Similar status available on RV32IM, RV32IMC, RV64I, RV64IM suites
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
  • very useful for users as provides coverage of custom instructions
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
- 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 – 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 ‘rubber stamp: RISC-V inside’
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
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.
April 2019
Two key issues in verification of SoCs based on open ISA processors:

1. Verification of the processor
2. Verification of the interface between the processor and the NoC/system bus

Needed:
- Reference model of the processor
- Test generation
- Processor instruction coverage measurement
Agenda

• Compliance and verification
• Customization
  • Why customize the ISA?
  • How to add custom features
• Summary
There are and will be many different RISC-V CPU developers:

- Deliver RTL IP as a business (like Arm, MIPS, just for RISC-V)
  - e.g. Andes, CloudBear, Codasip, Incore, Syntacore, ...
- Develop internal cores for SoCs
  - e.g. Nvidia, ...
- Develop for internal use but make available as open source
  - e.g. Western Digital, ...
- Develop and use open source as a business opportunity
  - e.g. SiFive, ...
- Download and use open source RTL in their products
  - e.g. Greenwaves, Google, ...
- ...

...
RISC-V CPU Adopters: (above & below the line)

New architectures (ML, IoT), arrays of processors, custom features, high performance, feature control (efficiency), large SoC design, architecture innovation, ...

Key requirement: ‘freedom to innovate’

Researchers, education, open source community, freely available, no license cost or restriction, ...

Key requirement: ‘free’
Many (above the line) adopters of RISC-V want to add their own custom extension instructions

• Traditional ISA choice has been hard if you want to add your own custom processor instructions to an ISA

• RISC-V as an open standard has specific regions of instruction decode space specifically allocated for users to add their own instructions

• Challenges
  • How to choose the processor IP starting point
  • How to add instructions to processor RTL
  • How to verify the complete RTL
  • How to evaluate effectiveness and performance gains of new instructions
  • How to enable software development utilizing the new instructions
Key Challenge of Optimizing Custom Instructions Requires New Methodology, Tools

- How to choose the processor IP starting point
- How to add instructions to processor RTL
- How to verify the complete RTL
- How to evaluate effectiveness and performance gains of new instructions
- How to enable software development utilizing the new instructions

- Extend instruction accurate simulation tools and models to support analysis and optimization of custom instructions
Agenda – Custom Instructions

• Who is adopting RISC-V and why...
• Adding custom instructions to RISC-V processors
• Custom instruction optimization flow
Custom Instruction Optimization Flow

• Show the flow used when designing new instructions to improve performance of applications running on a RISC-V processor
  • Allows evaluation and firming up of the instructions
  • Gets to the specification of the instructions needed to be implemented in RTL
• Introduce the technologies & tools needed for each stage

- Application software used for this walk-through is a character stream encoder, based on ChaCha20 encryption algorithm
  - Instruction Extensions to RISC-V courtesy of Cerberus Security Laboratories Ltd
  - https://cerberus-laboratories.com
Flow to add new custom instructions

1. Characterize C Application
2. Develop New Custom Instructions
3. Characterize New Instructions in Application
4. Release & Deploy
5. Optimize & Document model
The first thing needed is a simulator – think ISS, however …

- Simulator and model need to have ability to extend to allow custom instructions and new tools
- Ease of use also important
- Need to have a business-friendly open source license
Imperas Environment for Embedded Software Development, Debug & Test

Application Software & Operating System

Virtual Platform
- Peripheral
- Memory
- OVP CPU

JIT simulator engine

SlipStreamer API

Software Verification, Analysis & Profiling (VAP) tools
- Trace
- Profile
- Code coverage
- Memory monitor
- Protocol checker
- Assertion checkers

Multiprocessor / Multicore Debugger
- OS task tracing
- OS scheduler analysis
- Fault injection
- Function tracing
- Variable tracing
- ...

Eclipse IDE

© Imperas Software Ltd.
April 2019
Flow to add new custom instructions

Characterize C Application
- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling
Instruction Accurate simulation of C application

- Cross compiled C application targeting RV32IM
  - Character stream encoder, with ChaCha20 encryption algorithm
- IA simulation
  - Imperas RISC-V ISS with configurable model of RISC-V specification selecting RV32IM
- Semihosting
  - Enables bare metal application to very simply access host I/O
- Runs fast
  - Over 1 billion instructions a second (standard PC)
    - Linux and Windows supported host OS
Cycle Approximate simulation
C application

- Same C application
- IA simulation + timing Estimation (IA+E)
  - Includes annotated timing estimation for RV32IM processor
- Same simulation data results, different timing as now counting cycles
- Shows how long algorithm will take to execute

- Extends simulated time
  - Was 12.89 secs now takes 16.59 secs
Function Profile C Application

- Same C application
- IA+E simulation
- Sampled profiling with call stack analysis

- Shows proportion of time spent in each application function
  - 21.35% spent in `processLine`
Flow to add new custom instructions

Characterize C Application
- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

Develop New Custom Instructions
- Design Instructions
- Add to Application
- Add to Model
- Add Timing
Add custom instructions to application

- Inline assembly using new instructions replacing C code
- 4 new instructions
- Cross Compile using standard tool
- Run on IA simulator

Unimplemented instruction exception
- As the instructions have not yet been added to the simulator model
Add custom instructions to model

• Use standard Open Virtual Platforms (OVP) instruction modeling APIs to add new instructions (and optional state) as **new extension library**
  • Easy to extend decode table, add efficient behavioral JIT code
  • Optionally can call directly into user’s provided C function of behavior

• Compile and link model extension library

• Simulate IA with ISS plus standard model extended with new library

➤ Instruction count and simulated time have reduced (IA)
Cycle Approximate simulation including custom instructions

- IA simulation + timing annotation + custom instructions
  - Includes timing estimation for RV32IM processor
  - Need to add timing estimation for new custom instructions
- Simulate using C code application with inline assembler of custom extensions
- IA simulator + timing tool + custom extension instruction library

- See estimated improvement in throughput of application on new processor
  - Was 16.59 secs without custom instructions
  - **Now 9.21 secs with custom instructions**
Trace custom instructions

- Simulator has many trace features built in
- See new custom instructions in trace disassembly
- Can select when/where to turn trace on/off
  - Very efficient tracing
Debug custom instructions

- Imperas MPD is Eclipse based source code debug tool
- Can debug using source line or instruction level
- See new custom instructions and any new additional state registers
Function Profile custom instructions application

- IA simulation + timing annotation + custom instructions with sampled profiling

- Shows where slowest function is
  - Now much faster...

- Shows benefits of using custom instructions
  - processLine was 21.35% now 16.3%
Basic Block (BB) Profile custom instructions application

- IA simulation + timing annotation + custom instructions with detailed BB profiling
- Shows where expensive instruction sequences are
- Allows understanding of instruction performance
  - Useful for Compiler teams
  - Useful for Hardware teams
### Flow to add new custom instructions

<table>
<thead>
<tr>
<th>Characterize C Application</th>
<th>Develop New Custom Instructions</th>
<th>Characterize New Instructions in Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Instruction Accurate Simulation</td>
<td>• Design Instructions</td>
<td>• Instruction Accurate Simulation</td>
</tr>
<tr>
<td>• Trace / Debug</td>
<td>• Add to Application</td>
<td>• Trace / Debug</td>
</tr>
<tr>
<td>• Timing Simulation</td>
<td>• Add to Model</td>
<td>• Timing Simulation</td>
</tr>
<tr>
<td>• Function Timing / Profiling</td>
<td>• Add Timing</td>
<td>• Function Timing / Profiling</td>
</tr>
</tbody>
</table>
Flow to add new custom instructions

Characterize C Application
- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

Develop New Custom Instructions
- Design Instructions
- Add to Application
- Add to Model
- Add Timing

Characterize New Instructions in Application
- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

Optimize & Document model
- Instruction Coverage
- Line Coverage
- Instruction Performance
- Generate PDF model doc
Further tools for model developers

• Model source line coverage
  • To see how completely the tests exercise the model
Further tools for model developers (2)

- Custom instruction coverage
  - To see that there are tests for new instructions

- Custom instruction profile
  - See how long the simulator takes to execute each instruction
  - Use to focus speed up simulation of instructions
  - Enables improvement of speed of simulation runs
Document custom instructions

- Imperas tools automatically generate a processor model document PDF
- Includes all base model registers and any new registers
- Provides detailed documentation of new custom instructions
Flow to add new custom instructions

Characterize C Application
- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

Develop New Custom Instructions
- Design Instructions
- Add to Application
- Add to Model
- Add Timing

Characterize New Instructions in Application
- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

Release & Deploy
- Check RISC-V Compliance
- Use as reference for RTL Design Verification
- Use in Imperas/OVP Platforms, EPKs
  - Heterogeneous / Homogeneous
  - Multi-core, Many-core
- Imperas Multi-Processor Debug, VAP tools
- Port OS, RTOS (Linux, FreeRTOS…)
- Use in many simulation envs (inc. SystemC)
- Deliver to end users

Optimize & Document model
- Instruction Coverage
- Line Coverage
- Instruction Performance
- Generate PDF model doc

© Imperas Software Ltd. April 2019
Summary

• Compliance is a new challenge for open ISAs
• Additional verification tasks related to the open ISA processor
• Custom instructions are a key value proposition of RISC-V
  • Adding custom instructions requires solving the key challenge of how to optimize those instructions
• Instruction accurate (IA) simulation environment using IA models can be extended to enable custom instruction analysis and optimization
• Flow for optimizing custom instructions in RISC-V processors is being used in real designs
Thank You

• Visit www.imperas.com/riscv and www.OVPworld.org/riscv for more information

• RISC-V Foundation Compliance Suite, includes riscvOVPsim available:
  • https://github.com/riscv/riscv-compliance

Larry Lapides
VP Sales, Imperas Software
larryl@imperas.com