Mutation-based Compliance Testing for RISC-V

Vладимир Herdt\textsuperscript{1,2}, Sören Tempel\textsuperscript{1}, Daniel Große\textsuperscript{2,3}, Rolf Drechsler\textsuperscript{1,2}

\textsuperscript{1}Group of Computer Architecture, University of Bremen, Germany
\textsuperscript{2}Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
\textsuperscript{3}Institute for Complex Systems, Johannes Kepler University Linz, Austria

tempel@uni-bremen.de
Background: RISC-V

- Open standard instruction set architecture (ISA)
- Implements a modular design consisting of:
  - A mandatory base standard
  - Optional extensions
- Example extensions: (A) Atomics, (M) Multiplication, ...
Background: Compliance Testing

**Problem:** Modular design can lead to fragmentation
   → Fragmentation can lead to compatibility problems

**Solution:** Compliance Testing

- Show that implementation adheres to specification
- RISC-V approach: Mainly hand-written test suite
- Available on GitHub: https://github.com/riscv/riscv-compliance
**Key Idea**

**Motivation:** Hand-written compliance tests may have coverage gaps
   → Tests may be insufficient to detect incompatibilities

**Key Idea:** Assess and improve quality of RISC-V compliance test suite

  **Assess:** Find coverage gaps in the test suite

  **Improve:** If gaps have been found, create new test cases
Approach: Mutation Testing

• Technique to evaluate quality of a test suite
• Based on modifications of tested code
  ■ Existing test suite should fail for mutant
  ■ If not: Coverage gap discovered in test suite
• Mutated an instruction set simulator (ISS)
• Created mutation classes tailored for ISS

```cpp
case Opcode::ADD: {
    auto rs1 = regs[instr.rs1()];
    auto rs2 = regs[instr.rs2()];

    // ----[ORIGINAL CODE]----
    // regs[instr.rd()] = rs1 + rs2;
    // ----[MUTATED CODE]----
    regs[instr.rd()] = rs1 - rs2;
    // ----[MUTATION END]----
    break;
}
```

**Figure:** Example mutation class.
Quality Assessment of Compliance Test Suite

Figure: Mutation-based assessment of test suite.

Results: Alive mutants constitute potential coverage gaps
→ Generate test cases to close coverage gaps
Test Generation (1)

**Approach:** Symbolic execution

→ Technique for discovering program paths

- Executed ISS with symbolic instruction
- Make registers, etc. unconstrained symbolic
- Look for difference in output behaviour

```cpp
case Opcode::ADD: {
    auto rs1 = regs[instr.rs1()];
    auto rs2 = regs[instr.rs2()];
    if (mutation_begin()) {
        regs[instr.rd()] = rs1 - rs2;
    } else {
        regs[instr.rd()] = rs1 + rs2;
    }
    mutation_end();
} break;
```

**Figure:** Paths through modified ISS.
Test Generation (2)

Output of symbolic execution solver:
- Single concrete instruction
- Concrete values for registers
- Concrete values for accessed memory words

⇒ **Transformation** into test case in RISC-V compliance test format
Figure: Complete overview of our mutation-based approach.
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>CT: Official</th>
<th>CT: Spec-based</th>
<th>Symbolic Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>#killed</td>
<td>2263</td>
<td>91</td>
<td>101</td>
</tr>
<tr>
<td>#alive</td>
<td>192</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>%alive</td>
<td>7.8%</td>
<td>52.6%</td>
<td>0%</td>
</tr>
<tr>
<td>runtime</td>
<td>4.14 h</td>
<td>25.12 h</td>
<td>2.40 h</td>
</tr>
</tbody>
</table>

**Table:** Results of the performed experiments.

- Comparison with prior work by Herdt et al. (Spec-based approach)
- Generated 2455 mutants specifically tailored for RISC-V ISS
Future Work

• Issues with platform-dependent code
  ▪ Mutants requiring specific memory addresses
• Focus on main execution unit
  ▪ Test virtual memory implementation?
• Enhance utilized mutation classes
Conclusion

Key Idea: Assess and improve quality of RISC-V compliance test suite

Assess: Alive mutants found $\rightarrow$ coverage gaps

Improve: Test case generation through symbolic execution

$\Rightarrow$ Evaluation indicates effectiveness of this approach

For more information visit: http://systemc-verification.org/risc-v