Open Source RISC-V Processor Verification Platform

Tao Liu, Richard Ho, Udi Jonnalagadda
Agenda

- RISCV-DV: Open source RISC-V instruction generator
- YAML based end-to-end verification flow
- Case study: ibex core verification
- Roadmap
RISCV-DV
SV/UVM based open source RISC-V instruction generator
https://github.com/google/riscv-dv
Generate a complete random RISC-V assembly program for processor verification

RISCV-DV

- Random instructions
- Trap handlers
- Page tables
- Data and stack sections
RISCV-DV new features

Initial release @01/2019

- ISA: RV32IMC, RV64IMC
- M/S/U privileged mode
- Virtual address translation
- Regression suite
- Simulator: vcs, incisive
- ISS: spike
- Simple simulation flow

New features

+ ISA: F/D/A extensions
+ Debug mode support
+ Functional coverage model
+ Hint/illegal instruction testing
+ Handshake mechanism
+ Vectored interrupt handling
+ Bare program mode
+ Simulator: Questa, Metrics
+ ISS: ovpsim, whisper, sail-riscv
+ YAML based end-to-end flow
+ Improve performance
+ New regression suite
+ ...

Google Cloud
RISC-V-DV new features

Illegal/HINT instruction support

Type of illegal instructions:
- Reserved opcode
- Unsupported instruction
- CSR access to unsupported CSR
- Operation in the wrong privileged mode
...

Configurable memory regions

Use cases:
- Access various internal memories
- PMP verification
- Virtual address translation verification
Debug mode support

Debug mode test scenarios

- ebreak with dcsr.ebreakx enable/disable
- Enter debug mode through external debug request
- Access non debug mode CSR in debug mode
- ebreak in debug ROM
- Exception in debug ROM
- Single step debug mode
- ....

How to check the processor’s behavior if ISS cannot be put to debug mode the same time as RTL design?
Handshake mechanism

Handshake packet definition

<table>
<thead>
<tr>
<th>24 bits</th>
<th>8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>cmd</td>
</tr>
<tr>
<td>data</td>
<td>data</td>
</tr>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>

CORE_STATUS : 0  
TEST_RESULT: 1  
WRITE_GPR : 2  
WRITE_CSR : 3

Variable length data packets

Use cases:
- Debug mode testing
- Interrupt testing
- Memory access fault testing
- Self-checked tests
  ...

Testbench               Normal program          Debug ROM

debug request

check

CORE_STATUS: IN_DEBUG_MODE
WRITE_CSR: dpc
WRITE_CSR: dcsr
dret
Functional coverage flow

- Spike log
- Ovpsim log
- Whisper log

- Log2csv
- Log2csv
- Log2csv

- Trace CSV

- RISC-V ISA
  - SV functional coverage model

- Independent of test generation flow
- Fast coverage closure without RTL sim. No RTL tracer needed
- Flexible to support different ISS
Functional coverage model

- Cover all possible operands of each instruction
- Hazard conditions
- Corner cases like overflow, underflow, divide by zero
- Aligned/unaligned load/store
- Positive/negative immediate value
- Forward/backward branches, branch hit history
- Hint instruction
- Illegal instruction
- All opcode
- Access to all implemented privileged CSR
- Exception and interrupt
Generator performance improvement

Average test generation time (s)

35s

7s

Run large regression in parallel with small batches
YAML based end-to-end verification flow
An ecosystem for RISC-V processor verification
YAML based end-to-end verification flow

Add new RTL simulator, ISS, test to the flow simply by modifying the YAML configure file.
YAML based end-to-end verification flow

- iss: spike
  path_var: SPIKE_PATH
  cmd: >
      <path_var>/spike --isa=<variant> -l <elf> ...

- iss: ovpsim
  path_var: OVPSIM_PATH
  cmd: >
      <path_var>/riscvOVPsim.exe
      --controlfile <cfg_path>/riscvOVPsim.ic

- iss: whisper
  path_var: WHISPER_ISS
  cmd: >
      <path_var> <elf> --log --xlen <xlen> --isa <variant>

- import: <riscv_dv_root>/yaml/base_testlist.yaml

- test: riscv_machine_mode_rand_test
  description: >
      Machine mode random instruction test
  iterations: 2
  gen_test: riscv_instr_base_test
  gen_opts: >
      +instr_cnt=10000
      +num_of_sub_program=5
      +boot_mode=m
  rtl_test: core_base_test
...
YAML based end-to-end verification flow

# Run the generator with specific RTL simulator
python3 run.py --simulator vcs

# Run the generator with two ISS and cross compare the result
python3 run.py --iss ovpsim,spike

# Run the generator with RV64IMC variant
python3 run.py --target rv64imc

# Run a test with 10 iterations in verbose mode
python3 run.py --test_name riscv_rand_instr_test -i 10 -v
An ecosystem for RISC-V processor verification

RTL simulator

ISS

SPIKE

Pre-defined targets

RV32I

RV32IMC

RV64IMC

RV64GC

...
Case study: Ibex core verification

Source code: https://github.com/lowRISC/ibex/tree/master/dv/uvm

Ibex features
RV32IMC + Debug mode + User mode + PMP + ....
Case study: Ibex core verification

**Categories of found bugs**

- **Illegal/hint instruct...** 25.0%
- **Debug mode** 31.3%
- **Others** 6.3%
- **Pipeline issue** 6.3%
- **Memory access fault** 12.5%
- **Interrupt** 18.8%
Roadmap
Complete RISC-V verification platform

**Complete ISA support**
- Improve test library for supported ISA extensions.
- Vector extension
- Bitmanip extension

**Functional coverage model**
- Extend the current model to support more ISA extensions, privileged spec coverage etc.

**Reference testbench**
- UVM environment framework
- Test/sequence infrastructure
- Interface agent library
- Predefined targets for open source RISC-V processors
Machine Learning for Verification Research Platform

- Improve the generator controllability
- Enhance coverage model to provide better differentiation of coverage results
- Provide standard API to interact with ML models
- Benchmark platform for different ML algorithms
Unified configuration

A single configuration format for all flows
Backup Slides
Generator flow

- Generate program header
  - Privileged mode setup
  - Page table randomization
  - Initialization routine
  - Generate main/sub programs
  - Branch target assignment
- Generate data/stack section
  - Generate page tables
  - Generate intr/trap handler
  - Test completion section
  - Call stack randomization
  - Apply directed instructions
Benchmark flow

Processor candidates

Pulpino RI5CY:
4 stages, RV32-IMC, DSP extension

Pulpino Ariane:
6-stage, RV64-IMC, single issue, in-order, support M/S/U privileged levels

Merlin:
Open Source RV32I[C] CPU

ISS simulator

Spike

Benchmark metrics

Bug hunting capability, test coverage
Flow integration effort, performance

Ariane core architecture
Bugs found

- Cache line access racing
- Privileged CSR access
- FENCE operation failure
- Page fault handling
- Incorrect branch execution
- ALU corner case bug
Handshake packet format

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_DEBUG_MODE</td>
<td>0</td>
</tr>
<tr>
<td>TEST_PASS</td>
<td>1</td>
</tr>
<tr>
<td>MSTATUS</td>
<td>2</td>
</tr>
<tr>
<td>0x0000_1234</td>
<td></td>
</tr>
</tbody>
</table>

- **WRITE_GPR:** MSTATUS
- **TEST_RESULT:**