Verification of the CVA6
Open Source Core

2022-05-04
Jean-Roch COULON - Thales DIS (INVIA)
in collaboration with Thales TRT and ECC
CVA6 is a RISC-V implementation

- Designed by ETH Zürich
- Open Source, maintained by OpenHW Group
- Application processor, System Verilog
Today CVA6 maturity is TRL3-4

Many features and configurations

- no A
- no F
- F, no D
- no C
- no PMP
- synchronous
- active high
- no H
- no user bit
- no counter
- 16kB 4ways
- 4kB 3ways
- no cache
- 16kB 4ways
- 32kB 8ways
- no CV-X-IF
- rv64
- no F
- no D
- sv32
- sv0
- H
- synchronous
- active low
- user bits
- Counters
- WB Cache
- reset
- WT
- Dcache
- CVA6 RISC-V configuration
- cfg
- Dcache
- CV-X-IF
- ASIC
- FPGA

www.thalesgroup.com
CVA6 is a good climber but…

- Many features
- Multi configurations
  - Big: 500 k gates
  - Small: 80 k gates
- Multi targets
  - FPGA
  - ASIC

... cannot be industrialized as it is

- Functional bugs
  - Compromise product industrialization
  - Create security vulnerabilities and safety issues
Verification cost

- Verification more costly than design implementation!
RISC-V to reuse verification

- All RISC-V processors follow the same Instruction Set Architecture
  - Unprivileged ISA, Privilege Architecture, External Debug Support

- Related Design Verification Plans are identical

- Verification methodology can be re-used by using same:
  - Verification repository, e.g. OpenHW Group
  - Reference model, e.g. Spike
  - Random test generation, e.g. riscv-dv
  - UVM agents
  - Test regression suites, e.g. riscv-arch-test
Can be reused for different processors!

- RISC-V Design spec
- DV plan
- DV plan
- Micro archi Design spec
- Test random generation riscv-dv
- Functional coverage RVFI
- UVM
- Check vs Reference model Spike
- Fix design bugs
Collaborations

- **Thales is the main CVA6 industrial contributor**
  - OpenHW Group member
  - Contributions in Design, Verification and OS support
  - Committer in OpenHW Group GitHub repositories

- **But…**
  - Sharing a same verification framework makes sense
  - If an organization plans to improve the maturity of a RISC-V processor (could be different from CVA6), collaboration is welcome!
Verification infrastructure
First test bench - Testharness

Support digital simulation in platform environment

**CVA6 Processor**
- Copro if
- RVFI if
- Copro
- RVFI tracer

**DM_TOP**

**rstGen**

**Peripherals**
- Timer
- UART
- SPI
- ETH

**simJTAG**

**dmiJTAG**

**simDTM**

**AXI**

**ROM**

**RAM**

**CLINT**

**PLIC**

**THALES GROUP**

**Template : 87211168-GRP-EN-003**

**THALES**

© Thales 2020 All rights reserved.
Second test bench - UVM

Support UVM verification environment

- CVA6 Processor
- AXI
- RAM
- Copro if
- RVFI if
- Uvmt_cvxif_if
- Uvmt_rvfi_if
- Uvmt_clknrst_if
Verification flow

- riscv-compliance
- riscv-tests
- riscv-arch-test
- Linux
- riscv-dv
  - Random Instruction Generator

- Job Launcher
- riscv.elf

- CVA6 Testharness
- Spike
  - RISC-V Instruction Set Simulator
- CVA6 UVM

- Trace and Compare
- code coverage

DUT:
- CVA6 Risc-V RTL
- CVA6 UVM

Testharness

LLVM

GCC
Pull Request workflow

Web hook

GitLab

CVA6 CI

GitHub Actions

THALES CI

RUNNER

Tests OK

Looks good to me

master branch
Continuous Integration checks

- Test Testharness, UVM versus Verilator, VCS
- Checks 22 CVA6 HW configurations (in progress)
- 40 first Mcycles with Verilator, about 1h
- On ASIC, max frequency and Silicon Area
- 32-bit verification
- Test open source regression suite
- For Genesys 2

- 22 CVA6 HW configurations (in progress)
- 40 first Mcycles with Verilator, about 1h
- On ASIC, max frequency and Silicon Area
- 32-bit verification
- Test open source regression suite
- For Genesys 2
Dashboard

**Update .gitlab-ci/scripts/merge_job_reports.py, .gitlab-ci/c...**

Authored by Yannick Casamatta | Pipeline #1461392 triggered by "parent_pipeline"

**ASIC Synthesis**
Synthesis estimation on fake Techno

**Smoke test vcs-uvm,spike**
In this section we challenge all target architecture with all simulator

- `rv64gc vcs-uvm assembly test tests/isa/rv64ui/add.s` - PASS
- `rv64gc vcs-uvm assembly test tests/isa/rv64ui/add.s` - PASS
- `rv64gc vcs-uvm assembly test tests/riscv-compliance/riscv-suite/rv32i/src/i-ADD-01.s` - PASS
- `rv64gc vcs-uvm assembly test tests/custom/hello_world/custom_test_template.s` - PASS
- `rv64gc vcs-uvm c test tests/custom/hello_world/hello_world.c` - PASS
- `rv32imac vcs-uvm assembly test tests/riscv-compliance/riscv-test-suite/rv32i/src/i-ADD-01.s` - PASS
- `rv32imac vcs-uvm assembly test tests/isa/rv32ui/add.s` - PASS

**Compliance**
This is the regression Compliance

**Riscv-test P**
This is the regression riscv-test (physical)

**Smoke test vcs-testharness,spike**
In this section we challenge all target architecture with all simulator
Third test bench – FPGA

- Support Linux 64 and 32 bits execution
- Connect FPGA to run Linux boot during CI (in progress)
Both RTL and verification are Open Source
Repositories

- **CVA6 requirement specification**

- **CVA6**
  - RTL, testharness test bench and FPGA generation scripts
    - [https://github.com/openhwgroup/cva6](https://github.com/openhwgroup/cva6)

- **core-v-verif**
  - UVM test bench, random test generation, continuous integration
    - [https://github.com/openhwgroup/core-v-verif](https://github.com/openhwgroup/core-v-verif)
CVA6 for Security Certification

Security Certification requests
- Right description of the test
  - functionalities / level of the test and test coverage
- Tests results
- At high level, certification body shall replay itself the test

Quality certification requests to monitor continuously results

CVA6 open source repository and continuous integration
- RTL and test codes
- Verification infrastructure to produce the test coverage
THANK YOU!

Thales DIS (INVIA)