2021

Verifying SPI with Universal Verification Methodology (UVM)

Yueting Zhao
zhao979@purdue.edu

PURDUE UNIVERSITY

System-on-Chip Extension Technologies (SoCET)

Verifying SPI with Universal Verification Methodology (UVM)

Yueting Mary Zhao

Advisors: Dr. Mark Johnson, Sarang Pramod, John Martinuk

Motivation and Objective

Testing and debugging are important steps in system-on-chip designs. The Universal Verification Methodology (UVM) provides a standardized method for verifying RTL blocks of integrated circuits. The purpose of this research project is to learn about the syntax and structure of UVM and to apply it to rigorously validate the functionalities of a recently developed SPI peripheral.

Why UVM?

- Standardized in the industry and well-formatted
- Scalable and reusable across projects
- Separated testbenches and tests
- Randomized test cases generated by software simulations
- Utilizes object-oriented programming concepts

Future Goals

- Inspect the current non-UVM testbench and come up test plans and checkers
- Implement testcases into the testbench
- Update the old UVM SPI testbench to meet the current SPI specification and functionality
- Design new UVM testbenches to test other RTL blocks in the design

UVM Testbench Structure

tb_top:
- Serves as the container to hold all other components
- Contains clock generation, design under test (DUT) and interface instantiation, invoking tests

UVM_test:
- Testcases used to check specific specifications and functionalities
- Contains factory registration, environment instantiation, sequences of tests

UVM_env:
- Defines configurations for different tests
- Contains scoreboard, agents, sub-environments, verification components instantiations and connections

UVM_agent
- Configurable, contains sequencer, driver, monitor

UVM_sequences:
- Object, created and destroyed based on requirements
- Randomizes transactions with constraints
- Puts together different variable values to create different scenarios

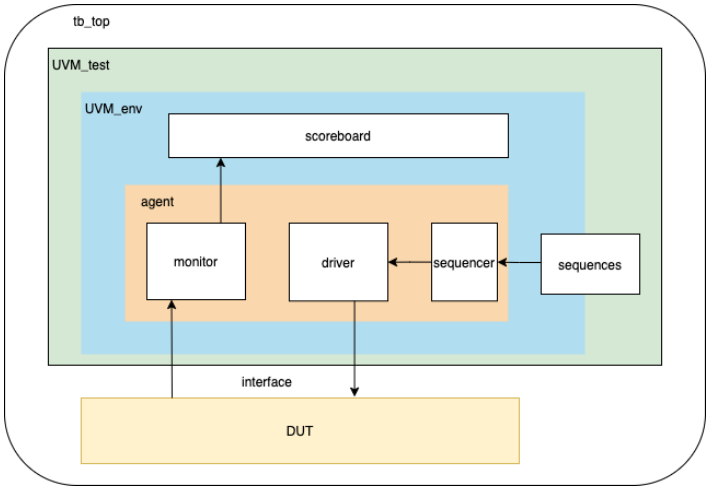
UVM_sequencer:
- Generates sequence transactions executed one-by-one
- Arbitrate sequences based on certain criteria

UVM_driver:
- Drives the signals to a particular interface and toggles the values
- Contains handshake signals with the sequencer

UVM_monitor:
- Collects signals from the design interface
- Send outputs to the scoreboard

UVM_scoreboard:
- Keeps track of the test results for all interfaces of the design
- Contains checkers to verify the functionalities

UVM Structure Overview



References

"UVM Tutorial for Beginners." ChipVerify, https://www.chipverify.com/uvm/uvm-tutorial.
"UVM Basics." Basic UVM | Universal Verification Methodology | Verification Academy, https://verificationacademy.com/courses/uvm-basics.

Acknowledgements

PURDUE UNIVERSITY Discovery Park — Encouraging undergraduate students for research

NAVSEA WARFARE CENTERS CRANE — Technical assistance and collaboration

INDIANA INNOVATION INSTITUTE — Funding through the ASSURE program