Title: MIPS 32bit ALU Emulation using UVM

Team: Haranadh Chintapalli

Project Report

Objective:

To perform Emulation of the DUT which is “MIPS ALU 32bit” in a UVM environment

Techniques applied in achieving the Objective:
Server side:

UVM based verfication strategy consisting of

1.Stimulus generators : Created three types of test input generators

-Type1: Randomized generated test input(test_throw_random.sv)

-Type2: Settled generated test input

(traversing through each opcode-test_throw_settled.svh)

-Type3: Opcode targeted test input generator(test_throw_fixed.svh)

2.Driver: Driver is created and used to get the test input from the generators and passed to virtual bfm.

3.Monitor_input side, Monitor_Output side, Scoreboard: Monitors use the calls the write function of the Scoreboard for updating the test inputs, results.Scoreboard checks and compares the resuls and displays on the terminal.

4. Environment: environment encloses the above mentioned uvm components and connections of the ports using tlm fifo.

5. Created uvm_tests for the different types of tests mentioned in the generators

6. Created top_hvl module which calls the run_test method.

Emulator side:

1. Created a module for the Bus Function Model for access the pinlevel connectivity from the DUT.Created tasks for passing the inputs to the DUT.
2. Top_hdl module is created and included the instances of the BFM, DUT.

Emulation is performed by passing the test cases using the bfm by accessing the UVM environment on the Server side.

Results Achieved:

1.Successfully able to integrate the UVM test environment with the top_hdl environment and able to pass test cases from the Veloce Server to the Emulator using the bfm setup.

2.ALU operations are checked using scoreboard and Monitor setup.

3. Two million different test inputs are generated and ran on the emulator for 30mins. Below is the emulation Report for 100,000 Random Test cases executed:

Results:

```
#
================================================================================
#                        SIMULATION STATISTICS
#
================================================================================
# Simulation finished at time 1094275
#
# Total number of TBX clocks:                              206612242
# Total number of TBX clocks spent in HDL time advancement:        218855
# Total number of TBX clocks spent in HDL due to callee execution: 0
# Percentage TBX clocks spent in HDL time advance:                 0.11 %
# --------------------------------------------------------------------------
# Total CPU time (user mode):                               26.74 seconds.
# Total time spent:                                        114.30 seconds.
# --------------------------------------------------------------------------
```

Effort:

3.5 Weeks for total completion of the project.Understanding the BFM model and UVM are the main tasks involved.

Challenges Encountered:

1. Main challenge is the integration of the UVM test model with the top_hdl
2. Setting up the path variable for running the test cases
3. Finishing all the elements in the uvm model
4. Writing the Bus function model and passing signals from the UVM environment
5. Raise and Drop objection needed for sure
6. Clock has to be correctly set other wise end up in just UVM Report summary displaying otherthan the testcases running.In order to trigger the clock, initial value is a must.
7. Correctly configuring the ports.

Makefile and Environment need to run:

Instructions for running the project:

//

1. Creating the environmental variable for UVM files

command:

- export UVM_HOME=/pkgs/mentor/questa/10.3/questasim/verilog_src/uvm-1.1d

check with:

- echo $UVM_HOME

Next Run in Puresim mode first.

2. ///////////////Puresim Mode//

for calling 100000 random test cases use the following command.

command:

- make all MODE=puresim TEST=test_throw_random_uv TEST_CASES=100000

similarly for other generators(optional)

- make all MODE=puresim TEST=test_throw_settled_uv TEST_CASES=100000
- make all MODE=puresim TEST=test_throw_fixed_uv TEST_CASES=100000 TEST=Add

3.//

After choosing one from the above three types of tests proceed to run on velocesolo1

- make all MODE=veloce TEST=test_throw_random_uv TEST_CASES=100000

similarly for other generators(optional)

- make all MODE=veloce TEST=test_throw_settled_uv TEST_CASES=100000
- make all MODE=veloce TEST=test_throw_fixed_uv TEST_CASES=100000 TEST=Add

////////////////////done //

Pre Existing ideas/code leveraged:

- MIPS ALU Design leveraged from XUM Project MIPS32 core implementation.
- https://github.com/grantae/mips32r1_xum
- UVM code snippets from -UVM Primer by Ray Salemi.
- Ideas borrowed from Mentor Graphics videos on UVM Methodology.
- Took support for uvm integration.

TestBenches executed:

Created three types of test input generators

-Type1: Randomized generated test input(test_throw_random.sv)

-Type2: Settled generated test input -
(traversing through each opcode deterministically-test_throw_settled.svh)
-Type3: Opcode targeted test input generator(test_throw_fixed.svh)
Verifying the results for each of the opcodes used

•Operations: Add, Subtract, Multiply, And, Or, Nor, Xor, Shift, Count leading 1s/0s

Bugs Injected:

- Changed the functionality for the opcodes and spotted the buggy errors.

Bugs Found:

- Ex_Ov is not getting updated.
- Division module signals is not clear.

What more can be done if more time:

- Concentrated more on the DUT-stalling, flushing signals , Division algorithm.
- Coverage can be done.