Project 3: UVM Testbench

Jacob Boline

November 11, 2018

1 Introduction

This project was an introduction to UVM design and simulation. Using the EDA playground website we converted the hello world testbench into an adder testbench. The code for this project can be found at https://www.edaplayground.com/x/4gPg.

2 Implementation

The DUT was fairly simple and was modified to add an a, b, and, cout, and cmd signal. Then if the command was a 0 it add, else it subtracted. In addition, the DUT displayed the command, values of a & b, the answer of the function, and the carry out value.

The sequence file was modified to produce random values between 0 and 1 for command, and random values between 0 and 256 for a & b.

Finally, the driver was modified to connect the sequencer and DUT ports for cmd, a, and b.

3 Output

```
1  # KERNEL: UVM_INFO @ 0: reporter [RNTST] Running test my_test...
2  # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 15: reporter [DUT] Received
       cmd = 1, a = 188, b =  67
3  # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 15: reporter [DUT] Result
       ans = 121, cout = 0
4  # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 25: reporter [DUT] Received
       cmd = 0, a = 243, b = 120
5  # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 25: reporter [DUT] Result
       ans = 107, cout = 1
6  # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 35: reporter [DUT] Received
       cmd = 0, a =  96, b = 225
7  # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 35: reporter [DUT] Result
       ans =  65, cout = 1
8  # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 45: reporter [DUT] Received
       cmd = 1, a =   3, b = 126
9  # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 45: reporter [DUT] Result
       ans = 133, cout = 1
10 # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 55: reporter [DUT] Received
       cmd = 1, a = 221, b =  80
```

```
11 # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 55: reporter [DUT] Result
       ans = 141, cout = 0
12 # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 65: reporter [DUT] Received
       cmd = 0, a = 238, b =  87
13 # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 65: reporter [DUT] Result
       ans =  69, cout = 1
14 # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 75: reporter [DUT] Received
       cmd = 0, a =  53, b = 146
15 # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 75: reporter [DUT] Result
       ans = 199, cout = 0
16 # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 85: reporter [DUT] Received
       cmd = 1, a = 178, b =   1
17 # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 85: reporter [DUT] Result
       ans = 177, cout = 0
18 # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 95: reporter [DUT] Received
       cmd = 1, a = 102, b = 165
19 # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 95: reporter [DUT] Result
       ans = 193, cout = 1
20 # KERNEL: UVM_INFO /home/runner/design.sv(24) @ 105: reporter [DUT] Received
        cmd = 0, a =  81, b = 126
21 # KERNEL: UVM_INFO /home/runner/design.sv(25) @ 105: reporter [DUT] Result
       ans = 207, cout = 0
22 # KERNEL: UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_objection.svh
       (1271) @ 105: reporter [TEST_DONE] 'run' phase is ready to proceed to the
        'extract' phase
23 # KERNEL: UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_report_server
       .svh(856) @ 105: reporter [UVM/REPORT/SERVER]
24 # KERNEL: --- UVM Report Summary ---
25 # KERNEL:
26 # KERNEL: ** Report counts by severity
27 # KERNEL: UVM_INFO :     23
28 # KERNEL: UVM_WARNING :  0
29 # KERNEL: UVM_ERROR :    0
30 # KERNEL: UVM_FATAL :    0
31 # KERNEL: ** Report counts by id
32 # KERNEL: [DUT]                  20
33 # KERNEL: [RNTST]                1
34 # KERNEL: [TEST_DONE]            1
35 # KERNEL: [UVM/RELNOTES]         1
```

4 Conclusion

From this exercise we learned the basic structure of a UVM test bench, and modified it to interface a basic adder. By adding other test fixtures to the design we can begin to learn more about UVM.