02GQCOQ - INTEGRATED SYSTEMS ARCHITECTURE

A.A. 2020/2021

PROF. MAURIZIO MARTINA
PROF. GUIDO MASERA

LAB4

Multiplier verification with UVM

Group 18

https://github.com/wackozz/isa/tree/main/lab4

Federica BONGO	s292395
Stefano RIZZELLO	s288013
Francesco VACCA	s279895

Contents

1 Adder UVM Test

The goal of the laboratory is to familiarize with the Universal Verification Methodology (UVM), the industrial standard for HDL verification and testbench building.

UVM is articulated as a set of System Verilog classes whose purpose is to generate a set of random stimuli for a given device under test (DUT). The stimuli can be constrained under a custom set of rules, that can be modified without the need of re-building the entire testbench. To verify the DUT, its output is eventually compared to the output of a reference high-level model which is fed with the same input.

To get started, a set of provided `.sv` files is given, which is configured to test the behavioral model of an adder. The files were compiled through QuestaSim issuing the command `vlog -sv ../tb/top.sv` and letting the `top` testbench entity simulate for 4 µs. An excerpt of QuestaSim log file is reported in listing no. 1.

```
# adder: input A = 1212576424, input B = 3087079914, output OUT =    4689042
# adder: input A = 01001000010001100111001010101000, input B =
↪  10111000000000010001100111101010, output OUT =
↪  00000000010001111000110010010010
# refmod: input A =  1212576424, input B = -1207887382, output OUT =     4689042
# refmod: input A = 01001000010001100111001010101000, input B =
↪  10111000000000010001100111101010, output OUT =
↪  00000000010001111000110010010010
# UVM_INFO @ 3045: uvm_test_top.env_h.comp [Comparator Match]
[...]
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  106
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Comparator Match]   101
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [env]     1
```

Listing 1: Excerpt of QuestaSim log file, initial simulation

The transcript reports the output of both the adder and the reference model in integer and binary form. At each iteration the simulation informs the user if the output of the two is equal [Comparator Match] or not [Comparator Mismatch]. The last rows of the log file report a brief statistic of the match/mismatch cases and of the generated UVM alerts.

1.1 Testbench for modified adder

The adder behavioral model parallelism is now changed from 32 bit width to 64 bit. Moreover, it is now required for a A and B to satisfy the conditions in Equation 1.

$$\begin{cases} 100 < A < 1000 \\ 0 < B < A/10 \end{cases} \tag{1}$$

First, the dut_if.sv file was modified in order to extend A,B and Z from 32 to 64 bits. Then, to satisfy the conditions above, the code in packet_in.sv was edited as showed in 2. Adder output in packet_out.sv was also changed to long int.

Transcript for the modified simulation is reported in Listing no.3. In this case, binary output was converted to hexadecimal to be better visualized.

```
rand longint A;
rand longint B;

   constraint a_constraint {
      A inside {[100:1000]};
      B > 0;
      B < A/10;
    }
```

Listing 2: Constrains for the inputs

```
# adder: input A =                  766, input B =                   25, output
↪  OUT =                  791
# adder: input A = 00000000000002fe, input B = 0000000000000019, output OUT =
↪  0000000000000317
# refmod: input A =                  766, input B =                   25,
↪  output OUT =                  791
# refmod: input A = 00000000000002fe, input B = 0000000000000019, output OUT =
↪  0000000000000317
# UVM_INFO @ 3045: uvm_test_top.env_h.comp [Comparator Match]
[...]
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  106
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Comparator Match]   101
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [env]     1
```

Listing 3: Excerpt of QuestaSim log file, simulation with constrains

Mismatch test In order to test the correct function of the compactor and the reference model, the behavioral model of the adder was modified changing the `"+"` operator of the `adder.sv` with a `"-"` as showed below:

```
tr_out.data = tr_in.A - tr_in.B;
```

As expected, a `[Comparator mismatch]` was printed in transcript for every iteration (listing no.4.

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  207
# UVM_WARNING :  101
# UVM_ERROR :    1
# UVM_FATAL :    0
# ** Report counts by id
# [Comparator Mismatch]   101
# [MISCMP]   202
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [env]     2
```

Listing 4: Excerpt of QuestaSim log file, simulation with mismatched model and DUT

2 MBE-Dadda tree multiplier UVM Test

The MBE Dadda tree multiplier developed during Laboratory 2 was tested again using UVM methodology. To lower the number of `.hdl` files, the design was synthesized using Synopsis, with the netlist used as DUT. The system verilog files used in the previous point were modified as follows:

- The `dut_if.sv` interface was modified as before to set the input parallelism to 24 bits and the output parallelism to 48 bits;
- The `dut.sv` was modified to instanciate the MBE multiplier netlist as the device under test;
- The reference model `refmod.sv` was modified in order to realize a multiplication instead of an addiction;
- A,B in `packet_in.sv` were set to random 32bit-integer; the 8 MSB are truncated.
- `packet_out.sv` was modified to evaluate Z/data as a long integer.

QuestaSim log file of the simulation is reported in listings no. 5.

```
# MBE: input A =  4616872, input B =    72170, output OUT = work.DUT(fast)d
 ↪  333199652240
# MBE: input A = 010001100111001010101000, input B = 000000010001100111101010,
 ↪  output OUT = 000000000100110110010100001111110011010110010000
# refmod: input A =     4616872, input B =       72170, output OUT =
 ↪  333199652240
# refmod: input A = 00000000010001100111001010101000, input B =
 ↪  00000000000000010001100111101010, output OUT =
 ↪  0000000000000000000000000100110110010100001111110011010110010000
# UVM_INFO @ 3045: uvm_test_top.env_h.comp [Comparator Match]
[...]
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  106
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Comparator Match]   101
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [env]     1
```

Listing 5: Excerpt of QuestaSim log file, simulation with MBE tree multiplier

3 Floating Point multiplier UVM Test

The last step of the laboratory is to verify the behavior of the entire FP multiplier assigned during laboratory 2.

The main difference with respect to the previous designs is that now the device is composed of a set of input register and 4 pipeline stages. The total latency from input to output is of 5 clock cycles.

Pipeline emulation The `DUT.sv` is essentially a software emulation of a finite state machine. To mimic the delay of the pipeline, a series of dummy states where inserted between `INITIAL` and `SEND` states. Input data for the reference model is considered valid during `STAGE1` state; at the same time, the inputs are fed in to the the DUT. After 4 clock cycles (`STAGE2`, `STAGE3`, `STAGE4`, `SEND`) the output data of the DUT is considered valid and compared to the model.

Input constraints QuestaSim do not support the generation of random single precision floating point. To compensate for it, separate `rand bit` variables were allocated for sign (1), exponent(8) and mantissa(23) bits. Finally, the 32 bit float is rebuilt using the `X==sign,exponent,mantissa` constraint to join the fields. Reasonable constraints were applied to the exponent and mantissa according to the magnitude of the number of interests. The code of `packet_in.sv` is reported in listings no.6.

Display and comparison To print A, B and Z bits as single precision floating point and vice versa, the directive `$bitstoshortreral` and `$shortrealtobits` were employed, as reported in listing no.7. A transcript output is reported in listing no.8. The single mismatch detected is due to the presence of the two input registers, which are initialized to 0 when the reset is active.[1] The "power on reset" procedure is not synchronized with the IN/OUT trivial pipeline mechanism, so the first comparison produces mismatch as expected. The correct synchronization between input and output is showed in Figure 1.

Moreover, the directive `$bitstoshortreral` returned 0 when issued in `DUT.sv` and denormal number are dealt in. In order to double check the results, a simple python code exploiting the view property of numPy library was used to properly visualize binary output as float32 when needed.

```python
import numpy as np
print(np.int32(0b10111111100111011011101100111001).view(np.float32))
>>> -1.2322761
```

[1]The reset signal produced by the UVM structure is negated in input to the registers, as the designed registers reset is active low.

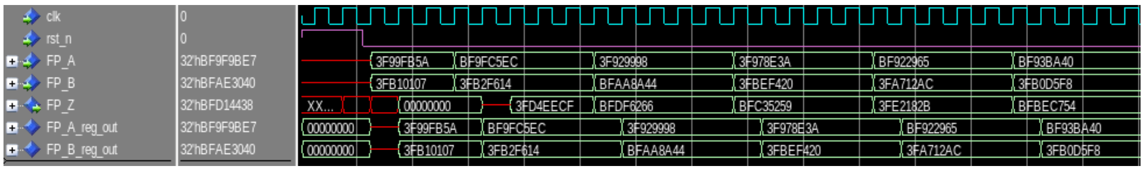

Figure 1: Snapshot of QuestaSim simulation. input (A,B) and output are synchronized.

```
rand bit [31:0] A;
rand bit [31:0] B;
rand bit sign_A;
rand bit sign_B;
rand bit [7:0] exponent_A;
rand bit [7:0] exponent_B;
rand bit [22:0] mantissa_A;
rand bit [22:0] mantissa_B;

constraint exponent_range_A {exponent_A > 100; exponent_A < 253;}
constraint exponent_range_B {exponent_B > 100; exponent_B < 253;}
constraint mantissa_A_range {mantissa_A < 2**21; mantissa_A > 2**20;}
constraint mantissa_B_range {mantissa_B < 2**22; mantissa_B > 2**21;}

constraint A_C {A == {sign_A, exponent_A, mantissa_A};}
constraint B_C {B == {sign_B, exponent_B, mantissa_B};}
```

Listing 6: Generation of random floating point numbers

```
tr_out.data = $shortrealtobits($bitstoshortreal(tr_in.A) *
 ↪  $bitstoshortreal(tr_in.B));
$display("refmod: input A = %b, \t input B = %b, \t output OUT = %b",tr_in.A,
 ↪  tr_in.B, tr_out.data);
$display("refmod: input A = %e, \t input B = %e, \t output OUT =
 ↪  %e",$bitstoshortreal(tr_in.A), $bitstoshortreal(tr_in.B),
 ↪  $bitstoshortreal(tr_out.data));
```

Listing 7: refmod.sv multiplication

```
# refmod: input A = 01001011000110000011110000101111,          input B =
 ↪  01011000001001000001110100010001,          output OUT =
 ↪  01100011110000110010111110101110
# refmod: input A = 9.976879e+06,          input B = 7.217790e+14,
 ↪  output OUT = 7.201102e+21
# fpmul: input A = 01001011000110000011110000101111, input B =
 ↪  01011000001001000001110100010001, output OUT =
 ↪  01100011110000110010111110101110
# fpmul: input A = 9.976879e+06, input B = 7.217790e+14, OUT_f = 7.201102e+21
# UVM_INFO @ 4975: uvm_test_top.env_h.comp [Comparator Match]
[...]
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  107
# UVM_WARNING :    1
# UVM_ERROR :    1
# UVM_FATAL :    0
# ** Report counts by id
# [Comparator Match]   100
# [Comparator Mismatch]     1
# [MISCMP]     2
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [env]     2
```

Listing 8: Excerpt of QuestaSim log file, simulation with floating point multiplier