Integrated System Architecture

Lab 4 Report

Academic Year 2021-2022

Professors:	Masera Guido	
	Martina Maurizio	
	Valpreda Emanuele	
	Walid Walid	
Students:	Lagostina Lorenzo	288019
	Lanza Mauro	290766
	Tudisco Antonio	285433
Group Number:	5	

Contents

1 Adder Tests

1.1 Overview

In this laboratory experience, we are tested our components using the UVM-testebench. Since some components needed to be limited to avoid the overflow/underflow, all tests were done with and without constraints to see how those components worked.

1.2 Tests without Constraints

Firstly, we tried to simulate the adder given to us with the laboratory material.
Using the commands written in the laboratory description we were able to see a match between the output of the design under test (DUT) and the expected results.

```
# 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
```

1.3 Tests with Constraints

We set a constraint for the inputs.
In order to do that, we added:

constraint my_range A inside [100 : 1000]; $B < 10 * A$; $B \geq 0$;

Then, we verified the correctness of the result:

```
# adder: input A =          779,
input B =         5067,
output OUT =          5846
# adder: input A = 00000000000000000000001100001011,
input B = 00000000000000000001001111001011,
output OUT = 00000000000000000001011011010110
# refmod: input A =          779,
input B =         5067,
output OUT =          5846
# refmod: input A = 00000000000000000000001100001011,
input B = 00000000000000000001001111001011,
output OUT = 00000000000000000001011011010110
# UVM_INFO @ 3015: uvm_test_top.env_h.comp [Comparator Match]
# adder: input A =          531,
input B =         2670,
output OUT =          3201
# adder: input A = 00000000000000000000001000010011,
input B = 00000000000000000000101001101110,
output OUT = 00000000000000000000110010000001
# refmod: input A =          531,
input B =         2670,
output OUT =          3201
# refmod: input A = 00000000000000000000001000010011,
input B = 00000000000000000000101001101110,
output OUT = 00000000000000000000110010000001
# UVM_INFO @ 3045: uvm_test_top.env_h.comp [Comparator Match]

# ** 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
```

Then, we changed the expected operation to verify the mismatch between the DUT and the expected results. To do that, it's necessary to edit the *refmod.sv* file in the following way:

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

The result is the following one:

```
# adder: input A =          531,
input B =        2670,
output OUT =         5846
# adder: input A = 00000000000000000000001000010011,
input B = 00000000000000000000101001101110,
output OUT = 00000000000000000001011011010110
# refmod: input A =          531,
input B =         2670,
output OUT =        -2139
# refmod: input A = 00000000000000000000001000010011,
input B = 00000000000000000000101001101110,
output OUT = 11111111111111111111011110100101
# UVM_INFO @ 3045: reporter [MISCMP] Miscompare for exp.data:
lhs = 'hffffff7a5 : rhs = 'hc81
# UVM_INFO @ 3045: reporter [MISCMP] 1 Miscompare(s) for object
tr@825 vs. exp@617
# UVM_WARNING @ 3045: uvm_test_top.env_h.comp [Comparator Mismatch]

# --- 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
```

2 Multiplier Tests

2.1 MBE-DADDA Tree Multiplier

In this part of the laboratory, we had to verify the correctness of the MBE-DADDA Multiplier.
So, first of all, we had to define a constraint in order to evaluate just the positive numbers.

```
constraint my_range { A inside {[0: 30000]}; B inside {[0: 30000]};};
```

It's also necessary to change the definition in the *refmod.sv* file.

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

The results are the following ones:

```
# multiplier: input A =        29352,
input B =        6634,
output OUT =              194721168
# multiplier: input A = 00000000000000000111001010101000,
input B = 00000000000000000001100111101010,
output OUT = 00000000000000000000000000000000000000101110011011001101011001
# refmod: input A =         29352,
input B =          6634,
output OUT =   194721168
# refmod: input A = 00000000000000000111001010101000,
input B = 00000000000000000001100111101010,
output OUT = 00001011100110110011010110010000
# 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
```

2.2 Floating Point Multiplier

For this part we had to test the Floating Point Multiplier defined in the second laboratory, to do that it's necessary to check the normal numbers (exponent result

is in the range [-127, +127]). To control it, we put a range for the bias exponent in the range [64, 190], and so the unbiased exponent for each input is in [-63, 63].
So in the packet in file, we have written:

```
rand bit[31:0] A;
rand bit[31:0] B;

constraint my_range { A[30:23] inside {[64 : 190]};
    B[30:23] inside {[64 : 190]};}
```

Then, in the *refmod.sv* file, we defined the following variable in order to simulate the 2-levels of pipe behaviour.

```
packet_in tr_in;
packet_out tr_out;
bit[31:0] tr_pipe1A;
bit[31:0] tr_pipe1B;
bit[31:0] tr_pipe2A;
bit[31:0] tr_pipe2B;


shortreal A1;
shortreal B1;
shortreal res;
```

In the same file, we added:

```
in.get(tr_in);
tr_pipe1A <= tr_in.A;
tr_pipe1B <= tr_in.B;
tr_pipe2A <= tr_pipe1A;
tr_pipe2B <= tr_pipe1B;



A1 = $bitstoshortreal(tr_pipe2A);
B1 = $bitstoshortreal(tr_pipe2B);
res = A1 * B1;
tr_out.data = $shortrealtobits(res);
$display("refmod: input A = %f, input B = %f,
output OUT = %f",A1, B1, res);
$display("refmod: input A = %b, input B = %b,
output OUT = %b",tr_pipe2A , tr_pipe2B , tr_out.data);
```

In the *DUT.sv* file, we had to declare some signal for the 2-levels pipe.

```
bit[31:0] tr_pipe1A;
bit[31:0] tr_pipe1B;
bit[31:0] tr_pipe2A;
bit[31:0] tr_pipe2B;
```

```
shortreal A1;
shortreal B1;
shortreal res;
```

And in order to print the output of the design under test, we added:

```
tr_pipe1A <= in_inter.A;
tr_pipe1B <= in_inter.B;
tr_pipe2A <= tr_pipe1A;
tr_pipe2B <= tr_pipe1B;



A1 = $bitstoshortreal(tr_pipe2A);
B1 = $bitstoshortreal(tr_pipe2B);
res = $bitstoshortreal(out_inter.data);
$display("mult: input A = %f, input B = %f, output OUT = %f",
A1, B1,res)
$display("mult: input A = %b, input B = %b, output OUT = %b",
tr_pipe2A, tr_pipe2B ,out_inter.data);
out_inter.valid <= 1;
```

The result is the following one:

```
# mult: input A = 0.000000,
input B = -2125274337752645632.000000,
output OUT = -381535744.000000
# mult: input A = 00101111010001010110001100111111,
input B = 11011101111010111111001111101110,
output OUT = 11001101101101011110111000110000
# refmod: input A = 0.000000,
input B = -2125274337752645632.000000,
output OUT = -381535729.508579
# refmod: input A = 00101111010001010110001100111111,
input B = 11011101111010111111001111101110,
output OUT = 11001101101101011110111000110000
# 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
```