Aristotle University of Thessaloniki
Department of Electrical and Computer Engineering

Lab Coursework - Digital Hardware Systems 2

Stavros Spyridopoulos 10845

Table of Contents

1 Floating Point Multiplier

1.1 Main Module

The main module of the DUT is `fp_mult.sv`. It is called through the `fp_mult_top.sv` wrapper module. It processes and splits the provided inputs from the wrapper into variables that will be passed to the other modules.
The initial sign calculation, exponent addition and biasing, and mantissa multiplication also take place here, before being passed to the later modules. Also, to handle denormal numbers, the addition of the leading one is done to Normals before being passed to the later modules.

1.2 Normalization Module

All calculated values are passed to the normalization module `mult_norm` of `normalize_mult.sv`. All transformations of the pre-normalization mantissa and exponent are based on mantissa's MSB so are the values of the guard and sticky bits.
A `sticky` and a `guard` bit are outputted from the module to be used in the rounding module of the DUT.

- **Sticky:** If any discarded bit is 1, the true value is slightly larger than what can be exactly represented, so this information will influence the rounding decision.
- **Guard:** Indicates whether the true value is closer to the lower or upper representable numbers.

After this, the leading one is added to the post-normalization mantissa, resulting in a 24-bit value.

1.3 Rounding Module

In the `round_mult` module of the `round_mult.sv` file, the mantissa rounding is executed based on the mantissa's LSB, sign, guard and sticky values.
Rounding types are stored in an `enum` type, in the `defs.svh` header file. The flow enters a `case` where rounding is conducted based on the `rnd` input.
Finally the module either adds 1 to the mantissa or truncates the last bit.

1.4 Exception Module

In the `exception_mult.sv` file, two functions and the `mult_except` module can be found.

- The `num_interp` function interprets its input as a type of number, based on *Table 1* of the lab manual, and returns the type to the caller.
- The `z_num` function gets a type of number and returns its value on a variable of the caller.

The `mult_except` module handles all exceptions properly while treating *NaNs* and *Denormals* as infinities and zeroes respectively.

1.5 Wrapper Module

The `fp_mult_top` just creates an instance of the `fp_mult` module, handles resets, and updates the instance's values based on its inputs. The arguments passed to the `fp_mult` module have changed. The `clk` input has been removed since only combinational logic is used throughout the multiplication and the `rst` input has been removed since reset is controlled on the wrapper module.

2 Testbench

Each module was initially tested individually with a small number of inputs to verify proper response.
To meet the demands of the project, 2 types of cases were examined. All results were compared to the output of the `multiplication` function provided, and the comparison's result was printed on Questa's Transcript. All rounding modes were passed to the `multiplication` function as strings.

2.1 Random Tests

A fixed number of random values (10000) were set as inputs to the wrapper module inside a loop, where the rounding mode changed on every iteration. In case the result did not match the `multiplication` function's result, a flag was raised so the user can view only the total random cases result (PASS/FAIL).

2.2 Corner Case Tests

All 144 corner cases mentioned in the Lab Manual were tested inside 2 loops. Using a `corner_case_t` type enum on the `defs.svh` header file, the iteration throughout all of the cases was easy to operate using only loop iterators.
In case the result did not match the `multiplication` function's result, a flag was raised so the user can view only the total corner cases result (PASS/FAIL).

3 Assertions

3.1 Immediate Assertions

On the `test_status_bits` module, immediate SVA were used to check status bit interference based on the descriptions of *Table 3* of the Lab Manual.
All cases of assertion were only checked when `rst` if set to 1. This is the case because status bits are in the `x` state before the first calculation.

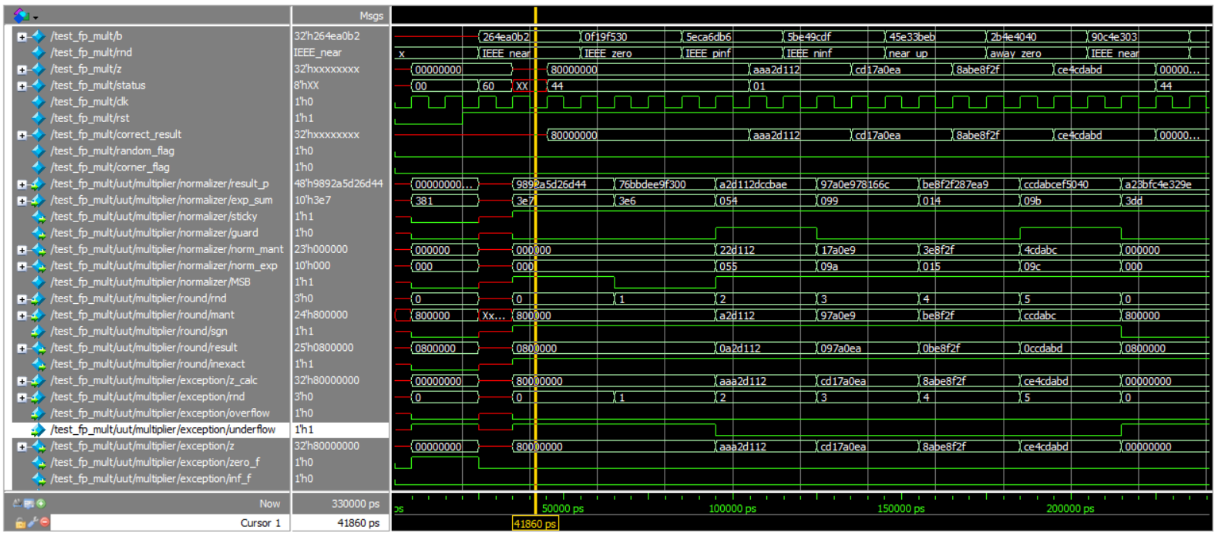
3.2 Concurrent Assertions

On the `test_status_z_combinations` module, concurrent SVA were used to check correlations between the inputs/outputs and the status bits at the positive edge of the clock.
All properties were stated and then asserted with the proper fail message.

3.3 SVA Testing

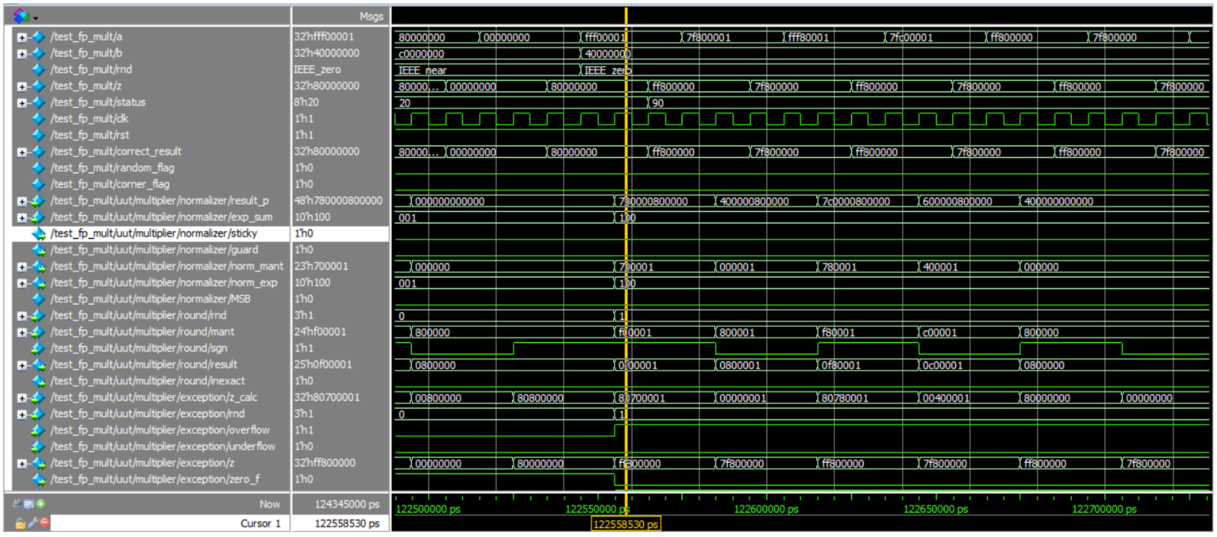
To test all assertions, `force` and `release` statements were used to trigger the undesired values. This code is commented out at the end of the `test_tb.sv` file. This code sample helped to verify that all asserted properties worked correctly. Results can be seen on (3).

4 Simulation Results



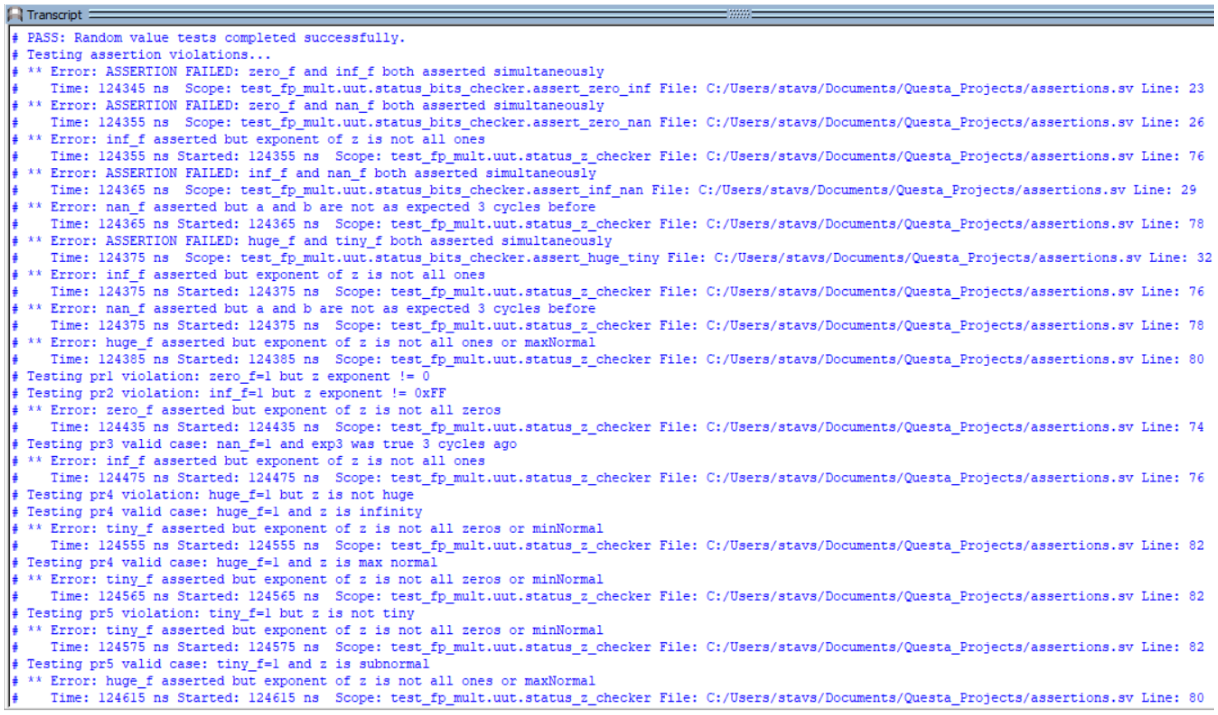
```
                                                      Msgs
/test_fp_mult/b                                       32'h264ea0b2
/test_fp_mult/rnd                                     IEEE_near
/test_fp_mult/z                                       32'hxxxxxxxx
/test_fp_mult/status                                  8'hXX
/test_fp_mult/clk                                     1'h0
/test_fp_mult/rst                                     1'h1
/test_fp_mult/correct_result                          32'hxxxxxxxx
/test_fp_mult/random_flag                             1'h0
/test_fp_mult/corner_flag                             1'h0
/test_fp_mult/uut/multiplier/normalizer/result_p      48'h9892a5d26d44
/test_fp_mult/uut/multiplier/normalizer/exp_sum       10'h3e7
/test_fp_mult/uut/multiplier/normalizer/sticky        1'h1
/test_fp_mult/uut/multiplier/normalizer/guard         1'h0
/test_fp_mult/uut/multiplier/normalizer/norm_mant     23'h000000
/test_fp_mult/uut/multiplier/normalizer/norm_exp      10'h000
/test_fp_mult/uut/multiplier/normalizer/MSB           1'h1
/test_fp_mult/uut/multiplier/round/rnd                3'h0
/test_fp_mult/uut/multiplier/round/mant               24'h800000
/test_fp_mult/uut/multiplier/round/sgn                1'h1
/test_fp_mult/uut/multiplier/round/result             25'h0800000
/test_fp_mult/uut/multiplier/round/inexact            1'h1
/test_fp_mult/uut/multiplier/exception/z_calc         32'h80000000
/test_fp_mult/uut/multiplier/exception/rnd            3'h0
/test_fp_mult/uut/multiplier/exception/overflow       1'h0
/test_fp_mult/uut/multiplier/exception/underflow      1'h1
/test_fp_mult/uut/multiplier/exception/z              32'h80000000
/test_fp_mult/uut/multiplier/exception/zero_f         1'h0
/test_fp_mult/uut/multiplier/exception/inf_f          1'h0
                                                 Now  330000 ps
                                            Cursor 1  41860 ps
```

Figure 1: Random Inputs Results - Time Zero



```
                                                      Msgs
/test_fp_mult/a                                       32'hfff00001
/test_fp_mult/b                                       32'h40000000
/test_fp_mult/rnd                                     IEEE_zero
/test_fp_mult/z                                       32'h80000000
/test_fp_mult/status                                  8'h20
/test_fp_mult/clk                                     1'h1
/test_fp_mult/rst                                     1'h1
/test_fp_mult/correct_result                          32'h80000000
/test_fp_mult/random_flag                             1'h0
/test_fp_mult/corner_flag                             1'h0
/test_fp_mult/uut/multiplier/normalizer/result_p      48'h780000800000
/test_fp_mult/uut/multiplier/normalizer/exp_sum       10'h100
/test_fp_mult/uut/multiplier/normalizer/sticky        1'h0
/test_fp_mult/uut/multiplier/normalizer/guard         1'h0
/test_fp_mult/uut/multiplier/normalizer/norm_mant     23'h700001
/test_fp_mult/uut/multiplier/normalizer/norm_exp      10'h100
/test_fp_mult/uut/multiplier/normalizer/MSB           1'h0
/test_fp_mult/uut/multiplier/round/rnd                3'h1
/test_fp_mult/uut/multiplier/round/mant               24'hf00001
/test_fp_mult/uut/multiplier/round/sgn                1'h1
/test_fp_mult/uut/multiplier/round/result             25'h0f00001
/test_fp_mult/uut/multiplier/round/inexact            1'h0
/test_fp_mult/uut/multiplier/exception/z_calc         32'h80700001
/test_fp_mult/uut/multiplier/exception/rnd            3'h1
/test_fp_mult/uut/multiplier/exception/overflow       1'h1
/test_fp_mult/uut/multiplier/exception/underflow      1'h0
/test_fp_mult/uut/multiplier/exception/z              32'hff800000
/test_fp_mult/uut/multiplier/exception/zero_f         1'h0
                                                 Now  124345000 ps
                                            Cursor 1  122558530 ps
```

Figure 2: Corner Cases Inputs Results



```
# PASS: Random value tests completed successfully.
# Testing assertion violations...
# ** Error: ASSERTION FAILED: zero_f and inf_f both asserted simultaneously
#    Time: 124345 ns  Scope: test_fp_mult.uut.status_bits_checker.assert_zero_inf File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 23
# ** Error: ASSERTION FAILED: zero_f and nan_f both asserted simultaneously
#    Time: 124355 ns  Scope: test_fp_mult.uut.status_bits_checker.assert_zero_nan File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 26
# ** Error: inf_f asserted but exponent of z is not all ones
#    Time: 124355 ns Started: 124355 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 76
# ** Error: ASSERTION FAILED: inf_f and nan_f both asserted simultaneously
#    Time: 124365 ns  Scope: test_fp_mult.uut.status_bits_checker.assert_inf_nan File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 29
# ** Error: nan_f asserted but a and b are not as expected 3 cycles before
#    Time: 124365 ns Started: 124365 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 78
# ** Error: ASSERTION FAILED: huge_f and tiny_f both asserted simultaneously
#    Time: 124375 ns  Scope: test_fp_mult.uut.status_bits_checker.assert_huge_tiny File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 32
# ** Error: inf_f asserted but exponent of z is not all ones
#    Time: 124375 ns Started: 124375 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 76
# ** Error: nan_f asserted but a and b are not as expected 3 cycles before
#    Time: 124375 ns Started: 124375 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 78
# ** Error: huge_f asserted but exponent of z is not all ones or maxNormal
#    Time: 124385 ns Started: 124385 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 80
# Testing pr1 violation: zero_f=1 but z exponent != 0
# Testing pr2 violation: inf_f=1 but z exponent != 0xFF
# ** Error: zero_f asserted but exponent of z is not all zeros
#    Time: 124435 ns Started: 124435 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 74
# Testing pr3 valid case: nan_f=1 and exp3 was true 3 cycles ago
# ** Error: inf_f asserted but exponent of z is not all ones
#    Time: 124475 ns Started: 124475 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 76
# Testing pr4 violation: huge_f=1 but z is not huge
# Testing pr4 valid case: huge_f=1 and z is infinity
# ** Error: tiny_f asserted but exponent of z is not all zeros or minNormal
#    Time: 124555 ns Started: 124555 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 82
# Testing pr4 valid case: huge_f=1 and z is max normal
# ** Error: tiny_f asserted but exponent of z is not all zeros or minNormal
#    Time: 124565 ns Started: 124565 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 82
# Testing pr5 violation: tiny_f=1 but z is not tiny
# ** Error: tiny_f asserted but exponent of z is not all zeros or minNormal
#    Time: 124575 ns Started: 124575 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 82
# Testing pr5 valid case: tiny_f=1 and z is subnormal
# ** Error: huge_f asserted but exponent of z is not all ones or maxNormal
#    Time: 124615 ns Started: 124615 ns  Scope: test_fp_mult.uut.status_z_checker File: C:/Users/stavs/Documents/Questa_Projects/assertions.sv Line: 80
```

Figure 3: Assertion Failures on Forced Inputs