
VLSI LAB REPORT | 2024

# (CO3098) LSI logic design

## Floating-point ALU
## Simulation report

**Submitted to: Mr. Huynh Phuc Nghi**

Submitted by: To Hoang Phong - 2112012
Vu Huynh Tan Phat - 2114391

Ho Chi Minh University of Technology (HCMUT)

The faculty of Computer Science and Engineering

| | |
|---|---|
| Author | Group 1 |
| Date | 2024/04/08 |
| Version | 1 |

# Contents

# 1 The github link to our project

Click here to visit our project

For testing the functional precision, we have developed a python program for auto functional verification.

The python program will calculate results directly by its algorithm and then transform them to IEEE754 32-bit floating-point numbers. Finally, another program will compare simulation results and python results.

**NOTE:** Because floating-point numbers in python are not represented in IEEE754 format, the precise of this program is quite unreliable. To be more accurate, we will also check the functional logic of our system by weitz's tool

# 2 AddOp verification

There are 4 cases to verify:

## 2.1 AddOp check in normal cases

### Check addition operator between 2 positive parameters

There were 10 test cases (no overflow/underflow cases and special cases):

```
12.5 5.25
20.25 12.75
20.84 12.9
125.7 24.92
2125.654 9924.123
123 456
2003 10
3.875 9.375
2.3 0.5
33554 52428
```

```
1: pass
2: pass
3: pass
4: pass
5: pass
6: pass
7: pass
8: pass
9: pass
10: pass
Pass 10/10 test cases
```

Figure 1: Test cases and auto test's result of two positive numbers' addition

```
At time                    0 : para1 = 41480000 + para2 = 40a80000 = out = 418e0000 - under_overflow = 0
At time                 1000 : para1 = 41a20000 + para2 = 414c0000 = out = 42040000 - under_overflow = 0
At time                 2000 : para1 = 41a6b851 + para2 = 414e6667 = out = 4206f5c3 - under_overflow = 0
At time                 3000 : para1 = 42fb6667 + para2 = 41c75c29 = out = 43169eb9 - under_overflow = 0
At time                 4000 : para1 = 4504da77 + para2 = 461b107d = out = 463c471b - under_overflow = 0
At time                 5000 : para1 = 42f60000 + para2 = 43e40000 = out = 4410c000 - under_overflow = 0
At time                 6000 : para1 = 44fa6000 + para2 = 41200000 = out = 44fba000 - under_overflow = 0
At time                 7000 : para1 = 40780000 + para2 = 41160000 = out = 41540000 - under_overflow = 0
At time                 8000 : para1 = 40133333 + para2 = 3f000000 = out = 40333333 - under_overflow = 0
At time                 9000 : para1 = 47031200 + para2 = 474ccc00 = out = 47a7ef00 - under_overflow = 0
```

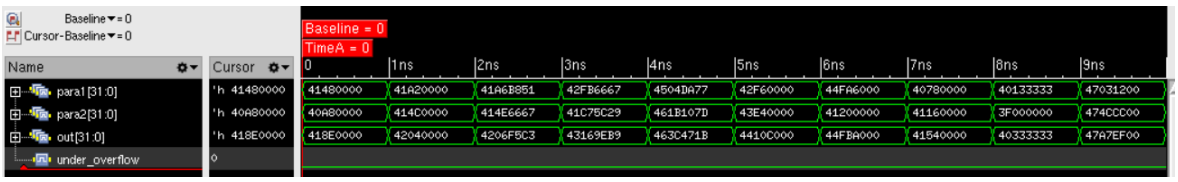



Figure 2: Console and waveform of two positive numbers' addition

**Description:** In testcase 1: parameter 1 is $12.5_d = 41480000_h$, parameter 2 is $5.25_d = 40a80000_h$ and their summary is $17.75_d = 418e0000_h$. Other test cases were explained similarly.

**Check addition operator between a positive parameter and a negative one**

There were 8 test cases (no overflow/underflow cases and special cases):

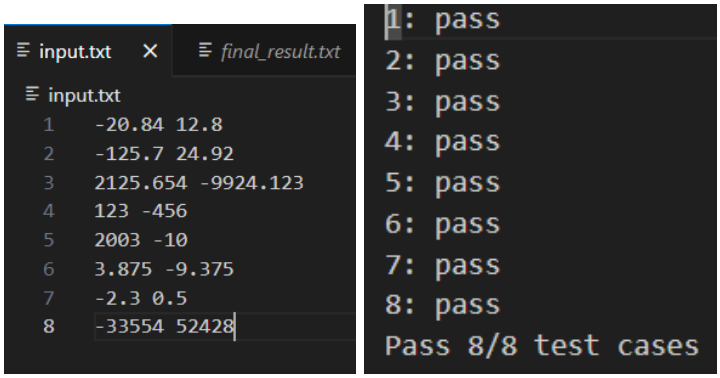



Figure 3: test cases and auto test's result of adding a positive number and a negative one

```
At time                0 : para1 = c1a20000 + para2 = 414c0000 = out = c0f00000 - under_overflow = 0
At time             1000 : para1 = c2fb6667 + para2 = 41c75c29 = out = c2c98f5d - under_overflow = 0
At time             2000 : para1 = 4504df00 + para2 = c61b1040 = out = c5f3b100 - under_overflow = 0
At time             3000 : para1 = 42f60000 + para2 = c3e40000 = out = c3a68000 - under_overflow = 0
At time             4000 : para1 = 44fa6000 + para2 = c1200000 = out = 44f92000 - under_overflow = 0
At time             5000 : para1 = 40780000 + para2 = c1160000 = out = c0b00000 - under_overflow = 0
At time             6000 : para1 = c00c0000 + para2 = 41270000 = out = 41040000 - under_overflow = 0
At time             7000 : para1 = c7031200 + para2 = 474ccc00 = out = 46937400 - under_overflow = 0
```

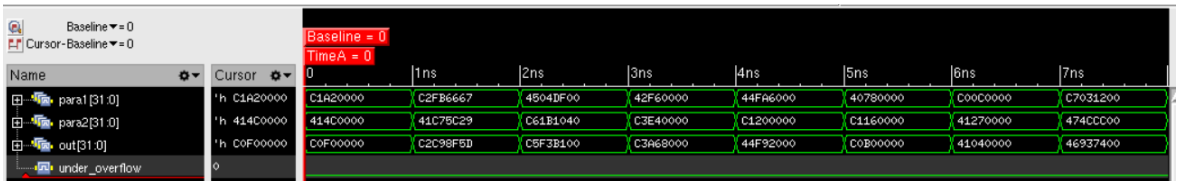



Figure 4: Console and waveform of adding a positive number and a negative one

**Description:** In testcase 2: parameter 1 is $-125.7_d = c2fb6667_h$, parameter 2 is $24.92_d = 41c75c29_h$ and their summary is $-100.78_d = c2c98f5d_h$. Other test cases were explained similarly.

### Check addition operator between 2 negative parameters

There were 6 test cases (no overflow/underflow cases and special cases):

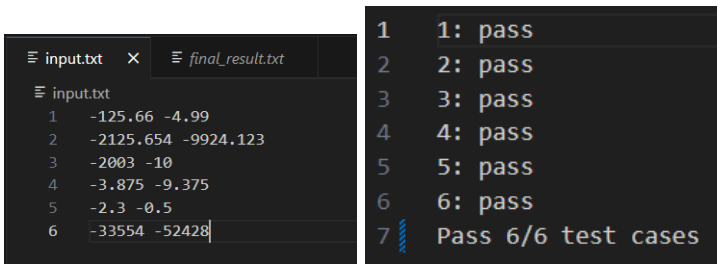
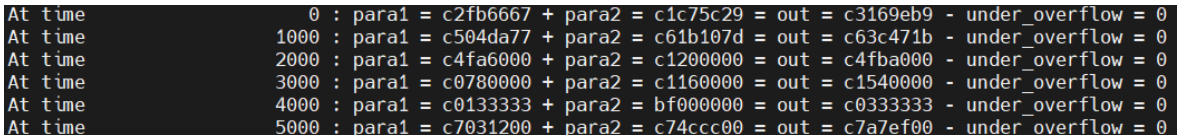



Figure 5: test cases and final result of two negative numbers' addition



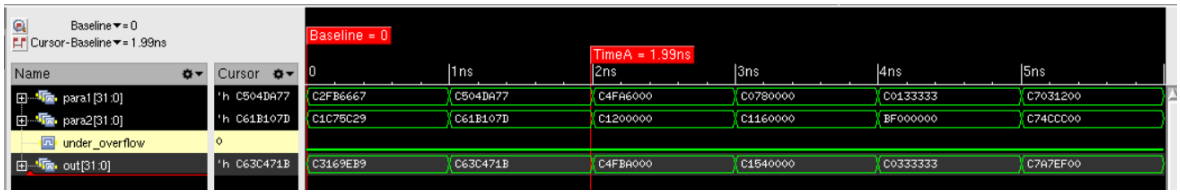




Figure 6: Console and waveform of two negative numbers' addition

**Description:** In testcase 4: parameter 1 is $-3.875_d = c0780000_h$, parameter 2 is $-9.375_d = c1160000_h$ and their summary is $13.25_d = c1540000_h$. Other test cases were explained similarly.

## 2.2 AddOp check in overflow cases

There were 2 test cases:

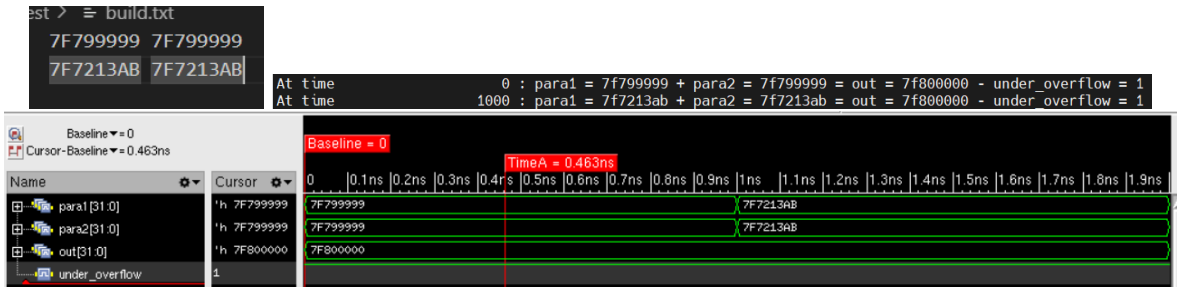



Figure 7: Console and waveform of addition overflow

**Description:** In testcase 1: parameter 1 is $7f799999_h \approx 3.318e+38_d$, parameter 2 is $7f799999_h \approx 3.318e+38_d$ and their summary is greater than the max IEEE754 number. Hence, the out value was $7f800000_h$. Another testcase was explained similarly.

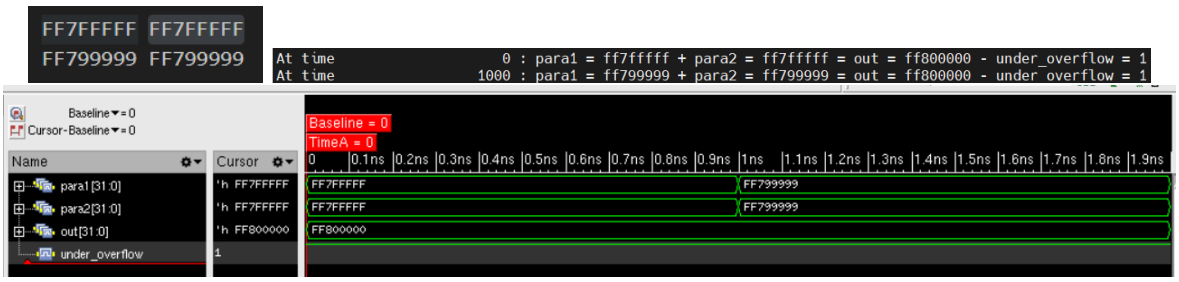



Figure 8: Console and waveform of addition underflow

## 2.3 AddOp check in underflow cases

There were 2 test cases:

**Description:** In testcase 1: parameter 1 is $ff7fffff_h \approx -3.4e+38_d$, parameter 2 is $ff7fffff_h \approx -3.4e+38_d$ and their summary is less than the min positive IEEE754 number. Another testcase was explained similarly.

## 2.4 Add a number with zero and add a number with its inversion

There were 3 test cases:

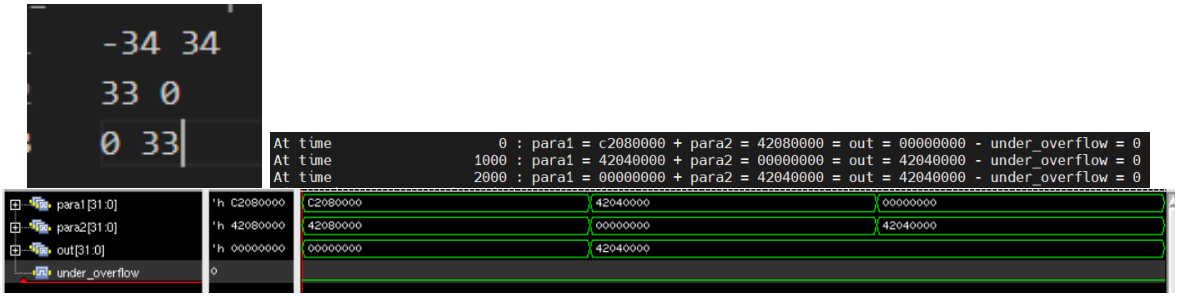



Figure 9: Console and waveform of adding with zero and with its inversion

**Description:** In above test cases, when a number plus with its inversion, the output would be zero and no underflow/overflow signal. When a number plus with zero, the output would be itself and no underflow/overflow signal.

# 3 MultiOp verification

There are 4 cases to verify:

## 3.1 MultiOp check in normal cases

**Check multiplication operator between 2 positive parameters**

There were 10 test cases (no overflow/underflow cases and special cases):

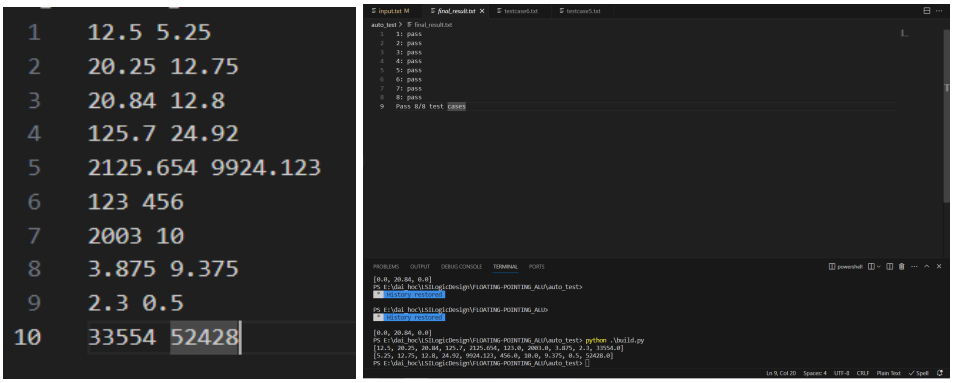



Figure 10: test cases and final result of multiplying two positive numbers

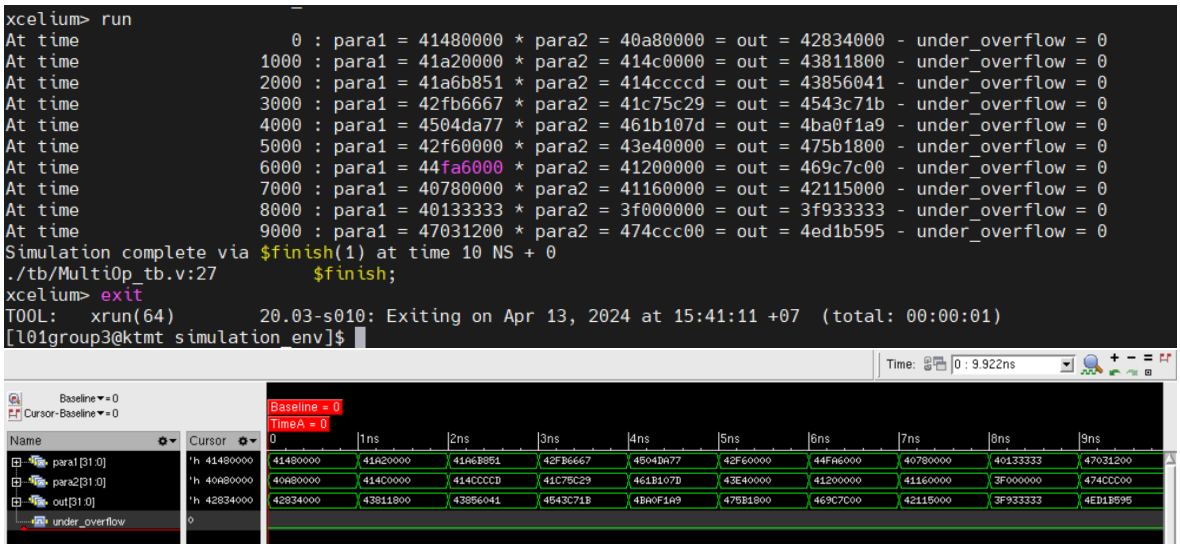



Figure 11: Console and waveform of multiplying two positive numbers

**Description:** In testcase 1: parameter 1 is $12.5_d = 41480000_h$, parameter 2 is $5.25_d = 40a80000_h$ and their product is $65.625_d = 42834000_h$. Other test cases were explained similarly.

**Check multiplication operator between a positive parameter and a negative one**

There were 8 test cases (no overflow/underflow cases and special cases):

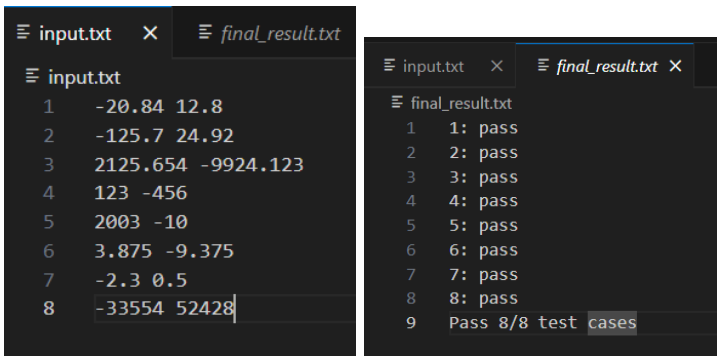



Figure 12: test cases and final result of multiplying positive and negative numbers

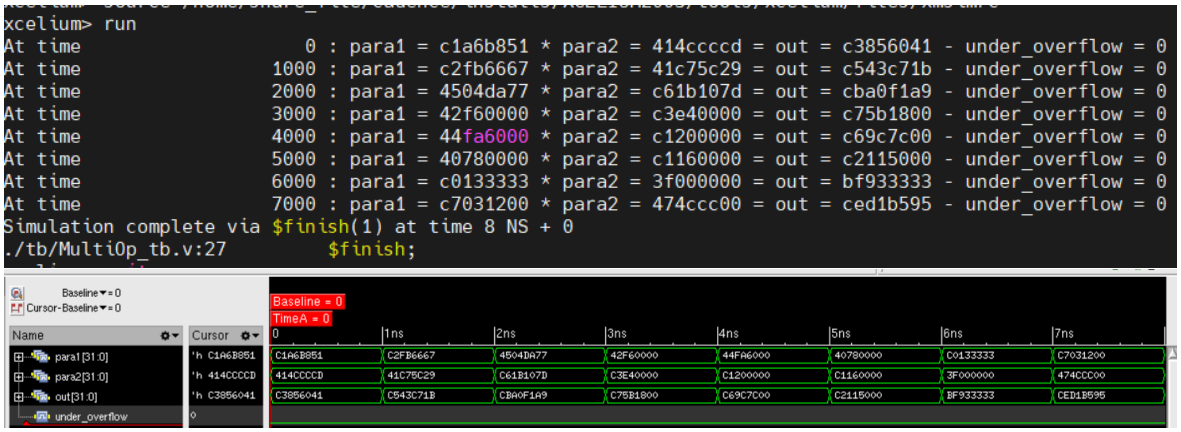



Figure 13: Console and waveform of multiplying two positive numbers

**Description:** In testcase 1: parameter 1 is $-20.84_d = c1a6b851_h$, parameter 2 is $12.8_d = 414ccccd_h$ and their product is $-266.752_d = c3856041_h$. Other test cases were explained similarly.

**Check multiplication operator between 2 negative parameters**

There were 6 test cases (no overflow/underflow cases and special cases):

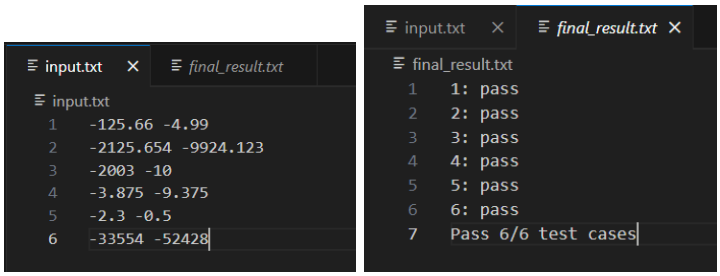



Figure 14: test cases and final result of multiplying two negative numbers

**Description:** In testcase 4: parameter 1 is $-3.875_d = c0780000_h$, parameter 2 is $-9.375_d = c1160000_h$ and their product is $36.328125_d = 42115000_h$. Other test cases were explained similarly.

## 3.2 MultiOp check in overflow cases

There were 2 test cases:

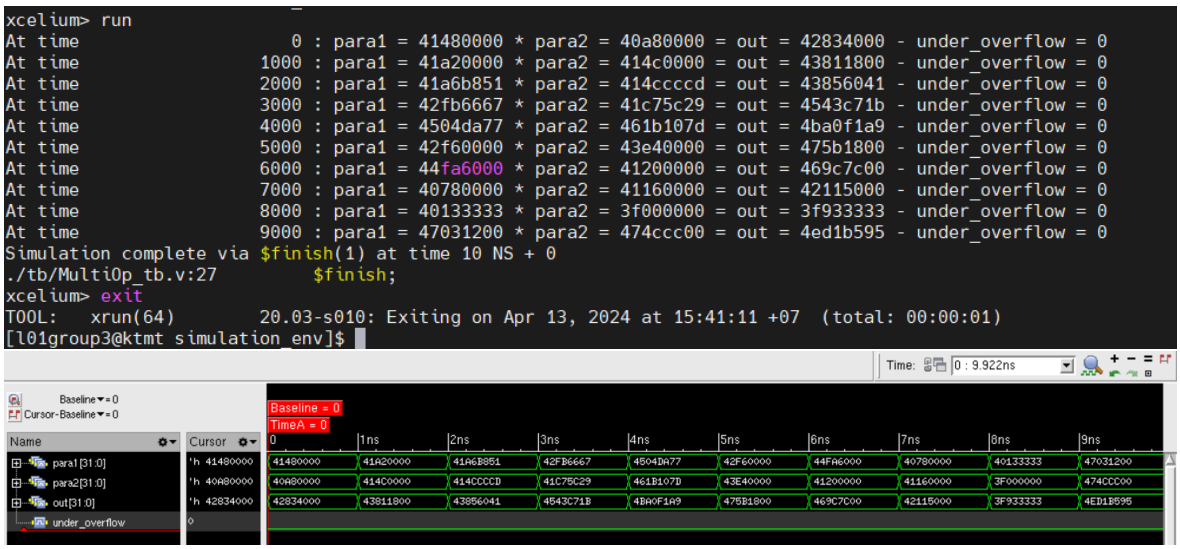



Figure 15: Console and waveform of multiplying two positive numbers

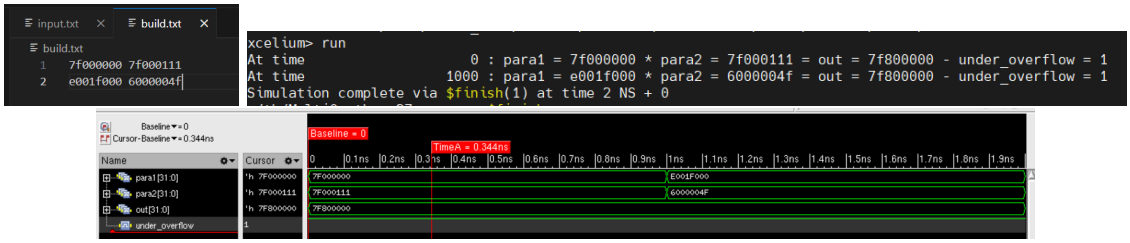



Figure 16: test cases and final result of multiplication overflow

**Description:** In testcase 1: parameter 1 is $7f000000_h = 2_d^{127}$, parameter 2 is $7f000111_h = 2^{127}(1 + 2^{-15} + 2^{-19} + 2^{-23})_d$ and their product is greater than the max IEEE754 number. Hence, the out value was $7f800000_h$. Another testcase was explained similarly, but in a negative result.

## 3.3 MultiOp check in underflow cases

There were 2 test cases:

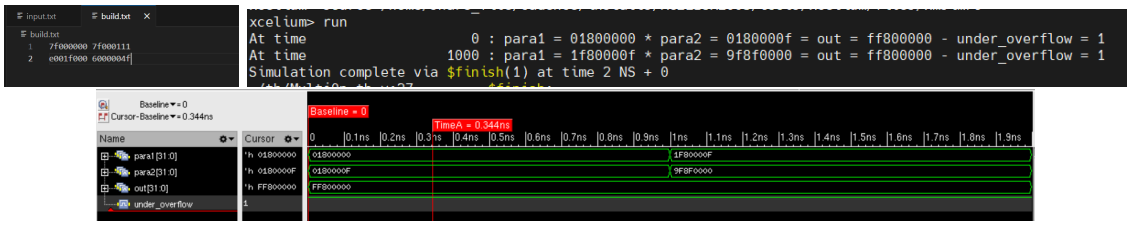



Figure 17: test cases and final result of multiplication underflow

**Description:** In testcase 1: parameter 1 is $01800000_h = 2_d^{-124}$, parameter 2 is $0180000f_h = 2^{-124}(1 + 2^{-23} + 2^{-22} + 2^{-21} + 2^{-20})_d$ and their product is less than the min positive IEEE754 number. Another testcase was explained similarly, but in a negative result.

## 3.4 Multiply a number with zero

There were 3 test cases:

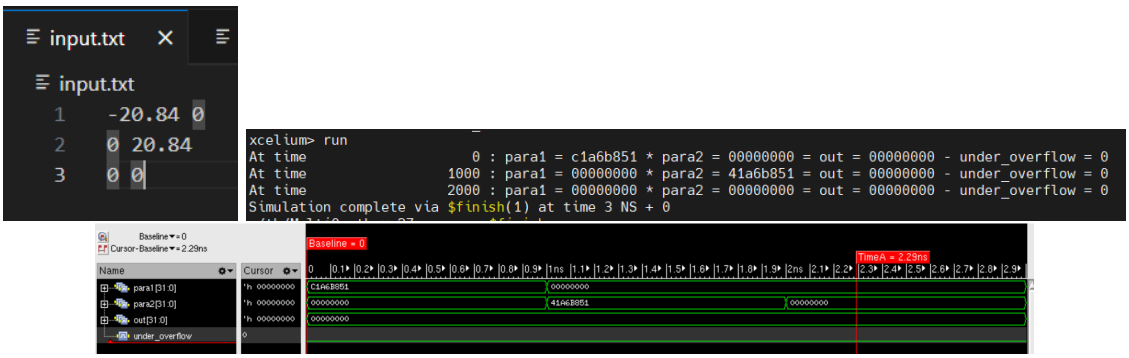



Figure 18: test cases and final result of multiplying with zero

**Description:** In above test cases, one of two or both parameters wew zero, so the out value was zero and no underflow/overflow signal.

# 4 SubOp verification

There are 4 cases to verify:

## 4.1 SubOp check in normal cases

### Check subtraction operator between 2 positive parameters

There were 8 test cases (no overflow/underflow cases and special cases):

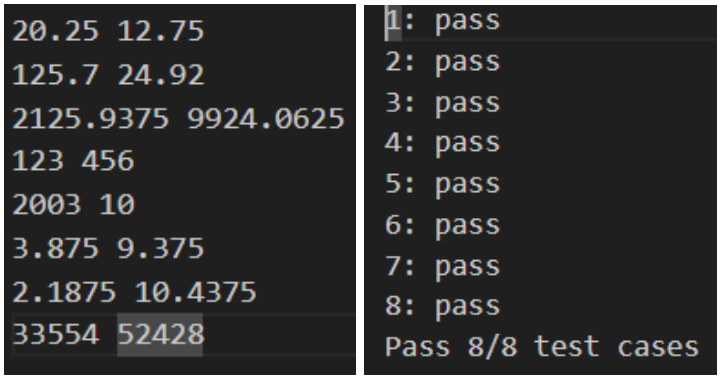



Figure 19: test cases and final result of two positive numbers' subtraction

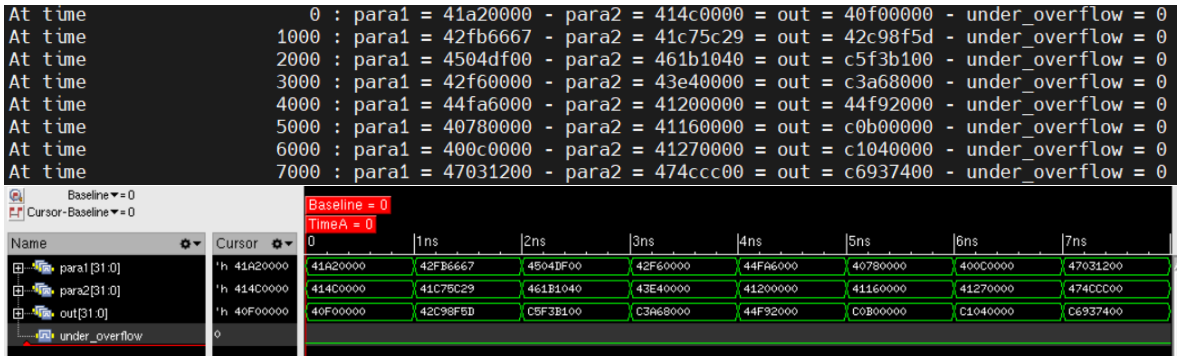



Figure 20: Console and waveform of two positive numbers' subtraction

**Description:** In testcase 1: parameter 1 is $20.25_d = 41a20000_h$, parameter 2 is $12.75_d = 414c0000_h$ and their difference is $7.5_d = 40f00000_h$. Other test cases

were explained similarly.

**Check subtraction operator between a positive parameter and a negative one**

There were 8 test cases (no overflow/underflow cases and special cases):

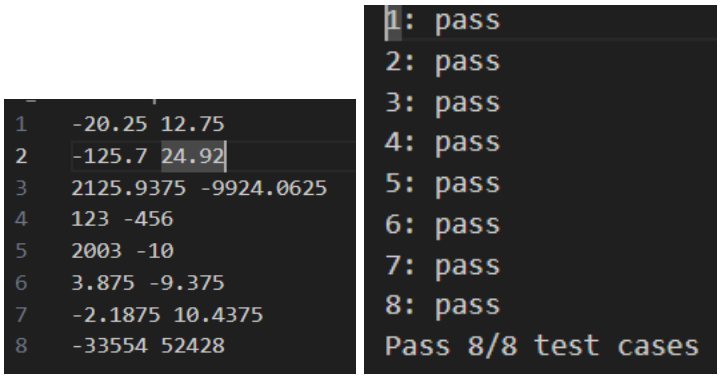



Figure 21: test cases and final result of one positive number and one negative number subtraction

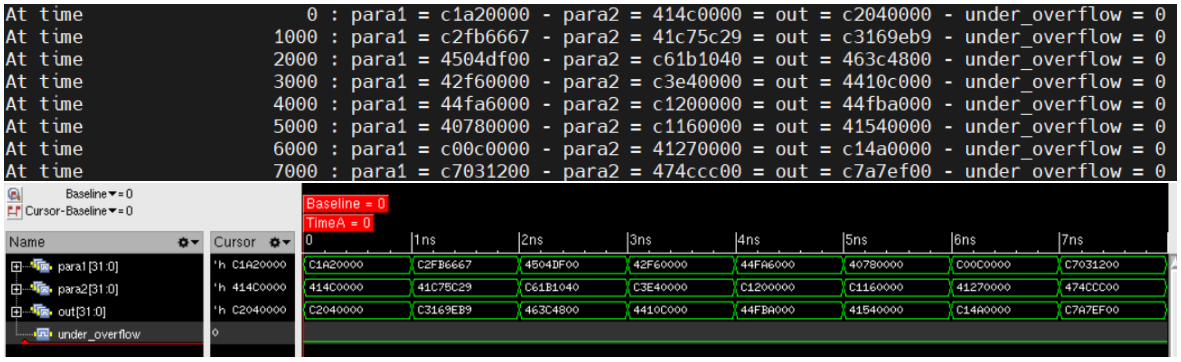



Figure 22: Console and waveform of one positive number and one negative number subtraction

**Description:** In testcase 2: parameter 1 is $-125.7_d = c2fb6667_h$, parameter 2 is $24.92_d = 41c75c29_h$ and their difference is $-150.62_d = c3169eb9_h$. Other test cases were explained similarly.

### Check subtraction operator between 2 negative parameters

There were 9 test cases (no overflow/underflow cases and special cases):

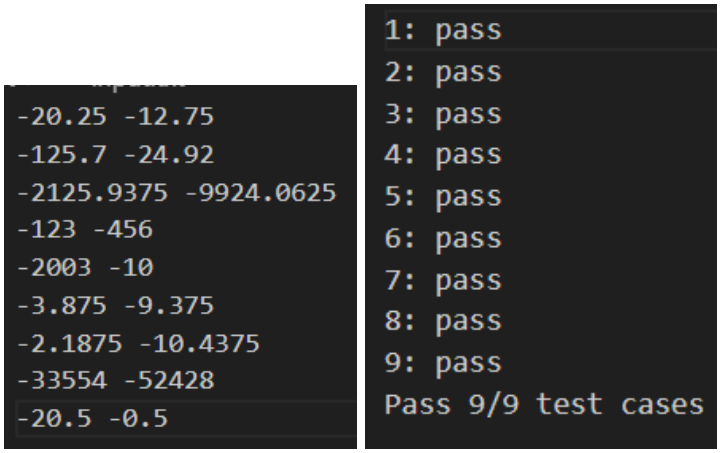



Figure 23: test cases and final result of two negative numbers' subtraction

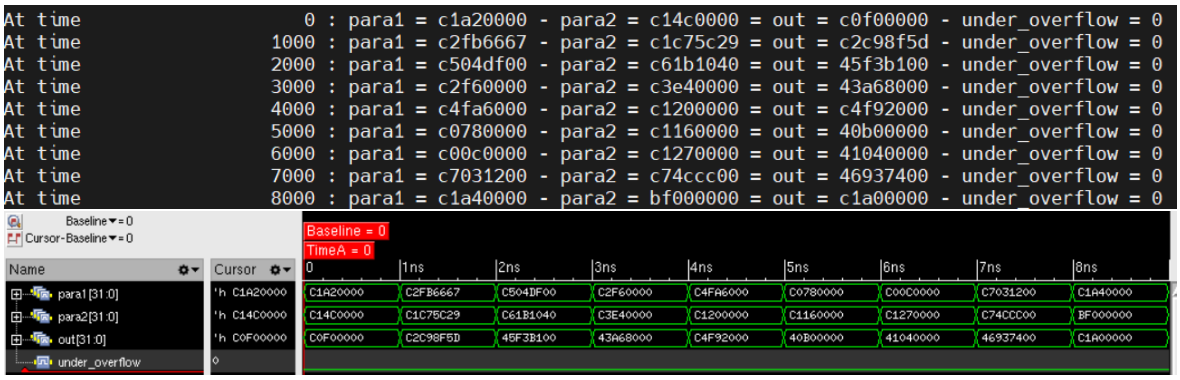



Figure 24: Console and waveform of two negative numbers' subtraction

**Description:** In testcase 6: parameter 1 is $-3.875_d = c0780000_h$, parameter 2 is $-9.375_d = c1160000_h$ and their product is $5.5_d = 40b00000_h$. Other test cases were explained similarly.

## 4.2 SubOp check in overflow cases

There were 2 test cases:

Figure 25: test cases of subtraction overflow

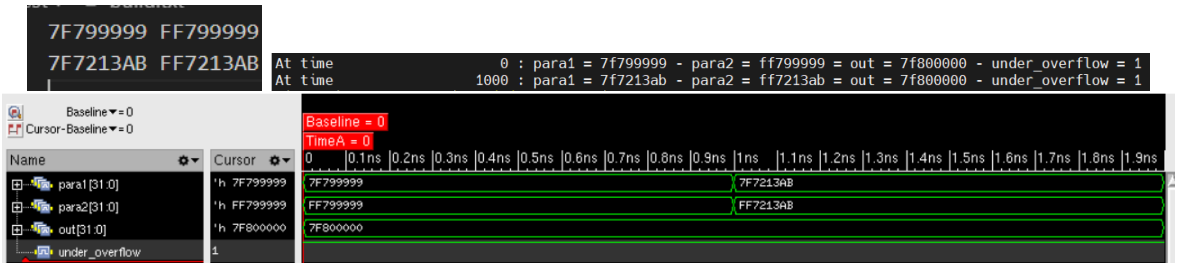



Figure 26: Console and waveform of subtraction overflow

**Description:** In testcase 1: parameter 1 is $7f799999_h \approx 3.318e + 38_d$, parameter 2 is $ff799999_h \approx -3.318e + 38_d$ and their difference is greater than the max IEEE754 number. Hence, the out value was $7f800000_h$. Another testcase was explained similarly.

## 4.3 SubOp check in underflow cases

There were 2 test cases:

Figure 27: test cases of subtraction underflow

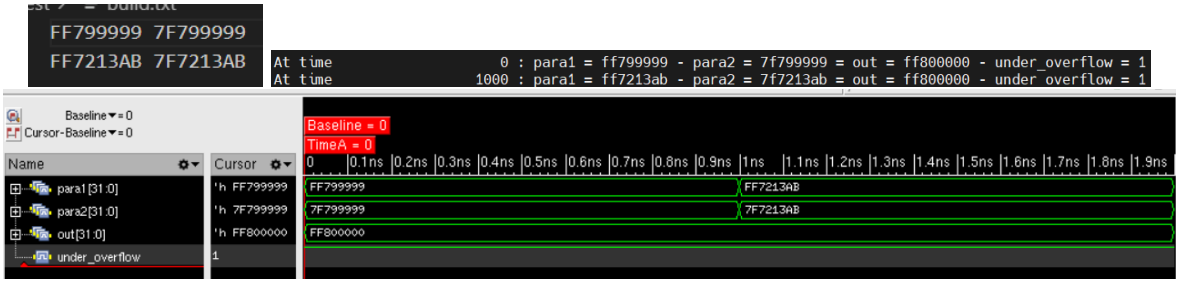



Figure 28: Console and waveform of subtraction underflow

**Description:** In testcase 1: parameter 1 is $ff799999_h \approx -3.318e + 38_d$, parameter 2 is $7f799999_h \approx 3.318e + 38_d$ and their difference is less than the min IEEE754 number. Hence, the out value was $ff800000_h$. Another testcase was explained similarly.

### 4.4 Subtract a number with zero - zero with a number - a number with itself

There were 3 test cases:

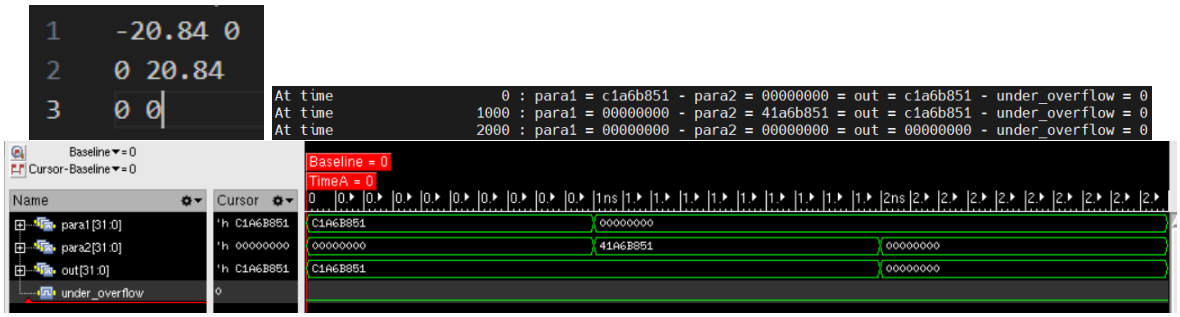



Figure 29: Console and waveform of of subtracting with zero - zero with a number - with itself

**Description:** In above test cases, when a number subtract with itself, the output would be zero and no underflow/overflow signal. When a number subtract with zero, the output would itself and no underflow/overflow signal. When zero subtract with a number, the output would be inversion of that number bug and no underflow/overflow signal.

## 5 FP_ALU verification

The above section verified all operations thoroughly, so in this section we just add ALU_op input, and verify one more signal (zero)

### 5.1 Check the accuracy of multiplexers

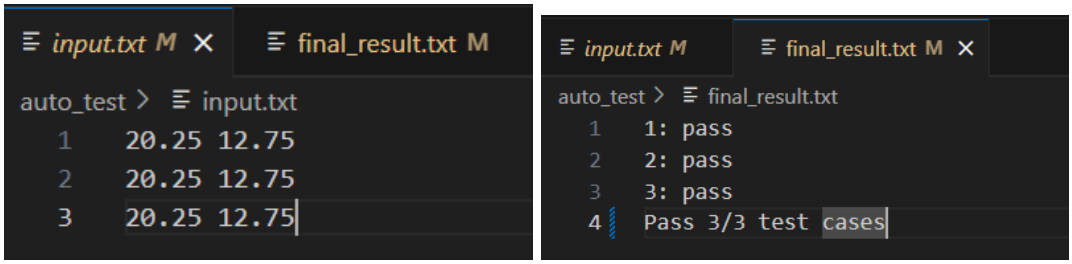



Figure 30: testcases and final result of checking the accuracy of multiplexers

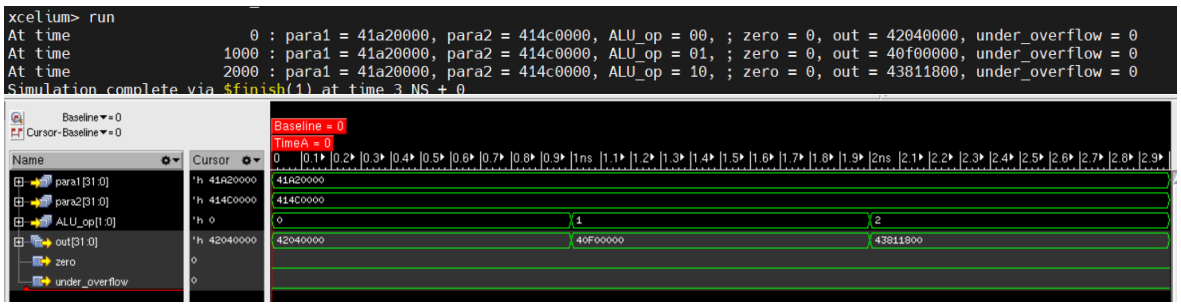



Figure 31: testcases and final result of checking the accuracy of multiplexers

Description: Testcase 1, 2, 3 respectively are the result of AddOp, SubOp and MultiOp. When the input 'ALU_op' = 0, the output 'out' = result of AddOp. When the input 'ALU_op' = 1, the output 'out' = result of SubOp. When the input 'ALU_op' = 2, the output 'out' = result of MultiOp.

## 5.2 Check 'zero' signal

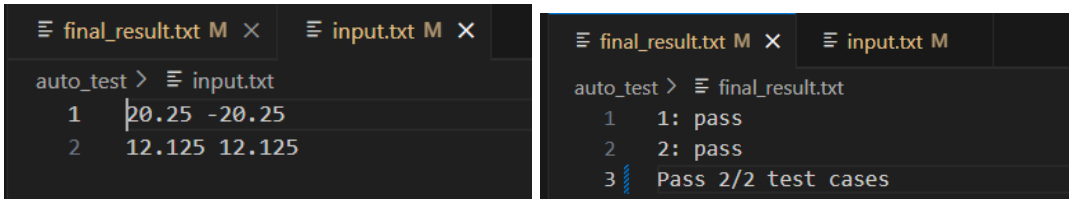



Figure 32: testcases and final result of checking 'zero' signal

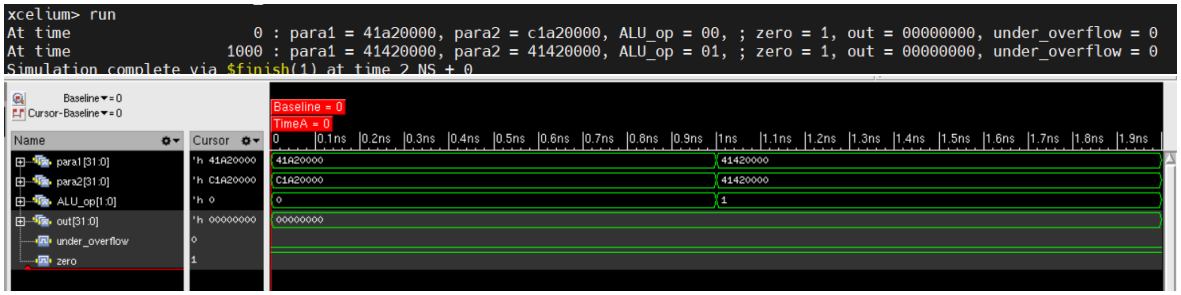



Figure 33: testcases and final result of checking 'zero' signal

Description: When the output 'out' equal to zero, the output 'zero' will be triggered to one.