

THUYẾT TRÌNH THIẾT KẾ HỆ THỐNG SỐ VỚI HDL – CE213.P21 ĐỀ TÀI

Thiết kế mạch nhân/chia dấu chấm động

Giáo viên hướng dẫn: Thạc sĩ Hồ Ngọc Diễm

Sinh viên:

- Dương Hiên Gia Khang: 22520610
- Trần An Huy : 22520574



01

Tổng quan dự án

02

Lưu đồ thuật toán

03

Các module chính

04

Kết quả

Các mục chính

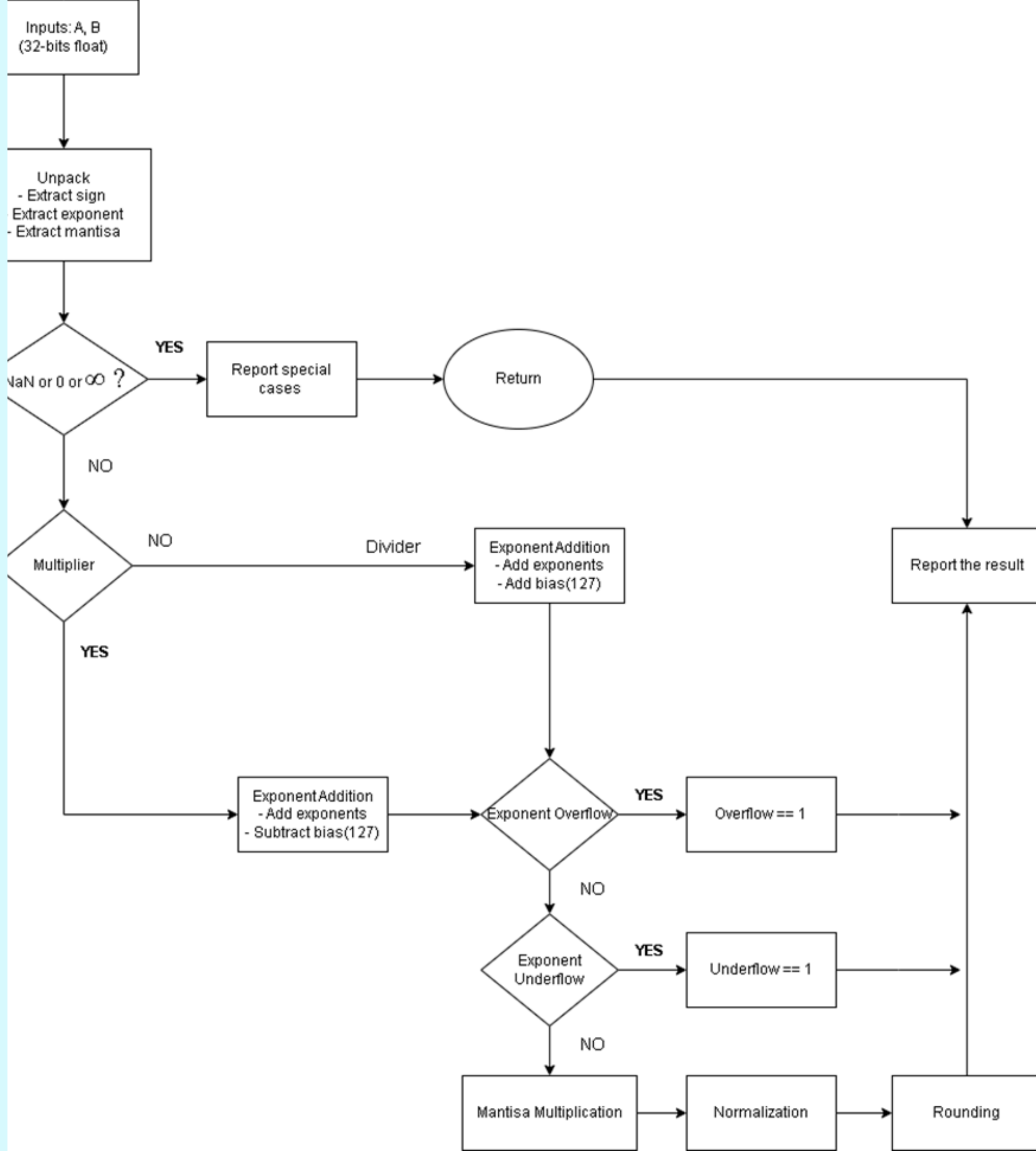
Floating point standard

IEEE 754 Floating Point Standard



$$\text{number} = (-1)^s * (1.m) * 2^{e-127}$$

Lưu đồ thuật toán





Các module chính

01

Unpack
module

02

Exponent unit
module

03

Mantissa unit
module

04

Normalization
module

05

Rounding
module

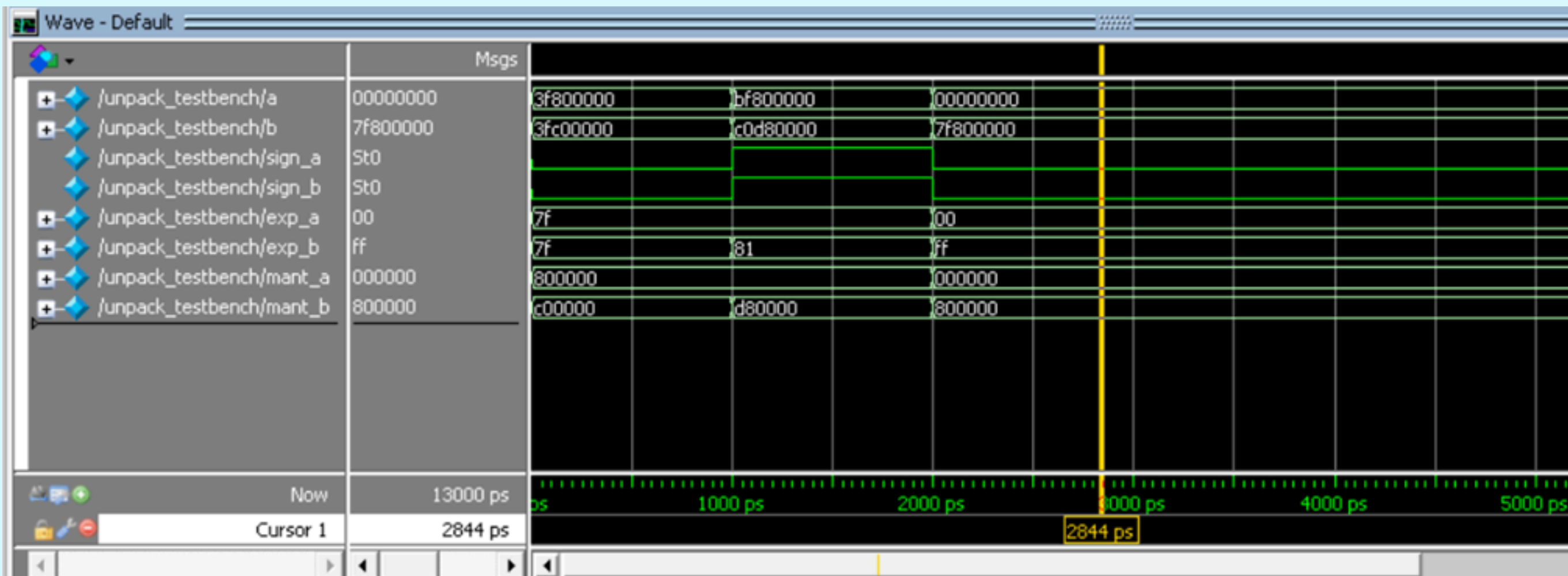
06

fp_arithmetic
module

Unpack module

Unpack: Giải nén (unpack) các số dấu phẩy động:

- Extract sign: sign bit.
- Extract exponent: exponent.
- Extract mantissa: mantissa.



Unpack module

```
# Test case 1: So duong chuan
# a = 3f800000 (0.000000)
# b = 3fc00000 (0.000000)
# sign_a = 0, exp_a = 7f (127), mant_a = 800000
# sign_b = 0, exp_b = 7f (127), mant_b = c00000
# Test case 1 passed
#
# Test case 2: So am chuan
# a = bf800000 (0.000000)
# b = c0d80000 (0.000000)
# sign_a = 1, exp_a = 7f (127), mant_a = 800000
# sign_b = 1, exp_b = 81 (129), mant_b = d80000
# Test case 2 passed
#
# Test case 3: So khong va vo cuc
# a = 00000000 (0.000000)
# b = 7f800000 (0.000000)
# sign_a = 0, exp_a = 00 ( 0), mant_a = 000000
# sign_b = 0, exp_b = ff (255), mant_b = 800000
# Test case 3 passed
# ** Note: $finish      : E:/HK2_nam3/HDL/FLT_MULTIPLEXER/Unpack_tb.v(75)
#      Time: 13 ns  Iteration: 0  Instance: /unpack_testbench
```

Exponent_unit module

Cộng phần Exponent của 2 số sau đó trừ/cộng vào phần bias

Wave - Default									
	Msgs								
/tb_exponent_unit/exp_a	100	100		-56			50		127
/tb_exponent_unit/exp_b	100	100			-56				127
/tb_exponent_unit/op_sel	0								
/tb_exponent_unit/exp_result	73	73		-29	-1	0			127
/tb_exponent_unit/overflow	St0								
/tb_exponent_unit/underflow	St0								

```
VSIM 4> run -all
```

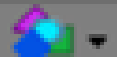








```
# Time  Op      Exp_A  Exp_B  Result  Overflow  Underflow
# 10ns  MUL      100    100    73      0         0
# 20ns  DIV      200    100    227     0         0
# 30ns  MUL      200    200    255     1         0
# 40ns  DIV      50     200    0       0         1
# 50ns  MUL      127    127    127     0         0
# ** Note: $finish      : E:/HK2_nam3/HDL/FLT_MULTIPLEXER/Exponent_tb.v(58)
#   Time: 50 ns  Iteration: 0  Instance: /tb_exponent_unit
# 1
# Break in Module tb_exponent_unit at E:/HK2_nam3/HDL/FLT_MULTIPLEXER/Exponent_tb.v line 58
```


Mantissa_unit module

```
VSIM 7> run -all
# Starting mantissa_unit testbench...
# -----
# mant_a = 1000000, mant_b = 500000
# ? Product = 500000000000 (0x00746a528800)
# ? Quotient = 0 (0x00000000)
# -----
# mant_a = 1234567, mant_b = 1
# ? Product = 1234567 (0x00000012d687)
# ? Quotient = 67108864 (0x40000000)
# -----
# mant_a = 7654321, mant_b = 7654321
# ? Product = 58588629971041 (0x35493a7a3061)
# ? Quotient = 67108864 (0x40000000)
# -----
# mant_a = 1000, mant_b = 0
# ? Product = 0 (0x00000000000000)
# ? Quotient = 134217727 (0x7fffffff)
# -----
# mant_a = 1, mant_b = 10000000
# ? Product = 10000000 (0x0000000989680)
# ? Quotient = 6 (0x00000006)
# -----
# Testbench finished.
# ** Note: $finish : E:/HK2_nam3/HDL/FLT_MULTIPLEXER/mantissa_unit_tb.
# Time: 50 ns Iteration: 0 Instance: /tb_mantissa_unit
```

Nhân/Chia các phần định trị. Phần định trị 32-bit có 23 bit ẩn và 1 bit ẩn (implicit 1), tổng cộng 24 bit cho độ chính xác:

- Nếu dùng phép nhân thì kết quả sẽ là 48 bit
- Ngược lại, phép chia sẽ có kết quả là 27 bit
- Khi nhân/chia hai số, ta giữ bit ẩn, và sau khi nhân thì kết quả cần chuẩn hóa và làm tròn lại tại các module normalization, rounding.

Wave - Default									
		Msgs							
	  /tb_mantissa_unit/...	000001	0f4240	12d687	74cbb1	0003e8	000001		
	  /tb_mantissa_unit/...	989680	07a120	000001	74cbb1	000000	989680		
	  /tb_mantissa_unit/p...	000000989680	00746a528800	00000012d687	35493a7a3061	000000000000	000000989680		
	  /tb_mantissa_unit/q...	0000006	0000000	4000000	4000000	7fffffff	0000006		

Normalization module

```
----- Multiply Test -----
Mantissa Input : 800000000000
Exponent In    : 127
Normalized Mant : 40000000
Normalized Exp  : 128
Underflow      : 0
Overflow       : 0
-----
```

```
----- Multiply Test -----
Mantissa Input : 400000000000
Exponent In    : 127
Normalized Mant : 40000000
Normalized Exp  : 127
Underflow      : 0
Overflow       : 0
-----
```

```
----- Multiply Test -----
Mantissa Input : 100000000000
Exponent In    : 127
Normalized Mant : 40000000
Normalized Exp  : 125
Underflow      : 0
Overflow       : 0
-----
```

```
----- Multiply Test -----
Mantissa Input : 000000000000
Exponent In    : 127
Normalized Mant : 00000000
Normalized Exp  : 0
Underflow      : 1
Overflow       : 0
-----
```

```
----- Divide Test -----
Mantissa Input : 000004000000
Exponent In    : 127
Normalized Mant : 40000000
Normalized Exp  : 127
Underflow      : 0
Overflow       : 0
-----
```

```
----- Divide Test -----
Mantissa Input : 000002000000
Exponent In    : 127
Normalized Mant : 40000000
Normalized Exp  : 126
Underflow      : 0
Overflow       : 0
-----
```

```
----- Divide Test -----
Mantissa Input : 000000000000
Exponent In    : 127
Normalized Mant : 00000000
Normalized Exp  : 0
Underflow      : 1
Overflow       : 0
-----
```

- Nhân ($\text{is_divide_op} = 0$)
 - Nếu bit MSB là 1 tại bit 47 \rightarrow shift phải 1, cộng $\text{exp} + 1$
 - Nếu MSB ở bit 46 \rightarrow giữ nguyên
 - Nếu MSB thấp hơn: tìm vị trí 1, dịch trái, trừ exponent
 - Nếu không có bit 1 \rightarrow kết quả là 0 \rightarrow underflow
- Chia ($\text{is_divide_op} = 1$)
 - Nếu bit 26 (MSB của mantissa chia) là 1 \rightarrow đã chuẩn
 - Nếu nhỏ hơn, dịch trái 1 bit, trừ exponent
 - Nếu $\text{mant_b} = 0$ từ trước đó \rightarrow không tính normalization (được xử lý ngoài)

Rounding module

```
# Time   Norm_Mant      Norm_Exp      Rounded_Mant  Rounded_Exp
# 10000  00000000000000000000000000000000      10      000000      10
# 20000  0000000000000000000000000000000001100      10      000002      10
# 30000  00000000000000000000000000000000010110      10      000003      10
# 40000  00000000000000000000000000000000011101      10      000004      10
# 50000  000000000000000000000000000000000100011      10      000004      10
# 60000  11111111111111111111111111111111100      100      000000      101
# ** Note: $finish      : E:/HK2_nam3/HDL/FLT_MULTIPLEXER/rounder_tb.v(55)
#   Time: 60 ns  Iteration: 0  Instance: /tb_rounding
# 1
```

Wave - Default		Msgs							
<div> <div>/tb_rounding/normalized_mant</div> <div>/tb_rounding/normalized_exp</div> <div>/tb_rounding/rounded_mant</div> <div>/tb_rounding/rounded_exp</div> </div>	7ffffc	00000000	0000000c	0000016	000001d	0000023	7ffffc		
	64	0a					64		
	000000	000000	000002	000003	000004		000000		
	65	0a					65		

- Chuẩn IEEE-754 yêu cầu:

- Định trị chỉ có 23 bit.

- Nhưng để làm tròn chính xác, ta cần thêm 3 bit phụ: G, R, S (Guard, Round, Sticky).

- rounding là bước quyết định cắt hay làm tròn lên dựa trên các bit phụ này.

-Quy tắc làm tròn (round-to-nearest-even):

if (G && (R || S || LSB))

temp_mant = temp_mant + 1;

-Làm tròn lên khi :

- G=1 và (R hoặc S khác 0) → có phần dư đáng kể.

- Hoặc tie (G=1, R=S=0) và LSB hiện tại = 1 → làm tròn thành số chẵn.

fp_arithmetic module

```
# A = 3f800000 | B = 3fc00000 | OP = MUL | Result = 3fc00000 | Expected = 3fc00000 | PASS
#
# Test:                2.0 * 0.5
# A = 40000000 | B = 3f000000 | OP = MUL | Result = 3f800000 | Expected = 3f800000 | PASS
#
# Test:                -3.0 * 4.0
# A = c0400000 | B = 40800000 | OP = MUL | Result = c1400000 | Expected = c1400000 | PASS
#
# Test:                0.0 * 5.0
# A = 00000000 | B = 40a00000 | OP = MUL | Result = 00000000 | Expected = 00000000 | PASS
#
# Test:                +Inf * 1.0
# A = 7f800000 | B = 3f800000 | OP = MUL | Result = 7f800000 | Expected = 7f800000 | PASS
#
# Test:                +Inf * 0.0 = NaN
# A = 7f800000 | B = 00000000 | OP = MUL | Result = 7fc00000 | Expected = 7fc00000 | PASS
#
# Test:                NaN * 1.0
# A = 7fc12345 | B = 3f800000 | OP = MUL | Result = 7fc00000 | Expected = 7fc00000 | PASS
#
# Test:                1.0 / 2.0 = 0.5
# A = 3f800000 | B = 40000000 | OP = DIV | Result = 3f000000 | Expected = 3f000000 | PASS
#
# Test:                3.0 / 2.0 = 1.5
# A = 40400000 | B = 40000000 | OP = DIV | Result = 3fc00000 | Expected = 3fc00000 | PASS
#
# Test:                -5.0 / 4.0 = -1.25
# A = c0a00000 | B = 40800000 | OP = DIV | Result = bfa00000 | Expected = bfa00000 | PASS
#
# Test:                0.0 / 1.0 = 0.0
# A = 00000000 | B = 3f800000 | OP = DIV | Result = 00000000 | Expected = 00000000 | PASS
#
# Test:                1.0 / 0.0 = NaN
# A = 3f800000 | B = 00000000 | OP = DIV | Result = 7f800000 | Expected = 7fc00000 | FAIL
#
# Test:                Inf / 1.0 = Inf
# A = 7f800000 | B = 3f800000 | OP = DIV | Result = 7f800000 | Expected = 7f800000 | PASS
#
# Test:                Inf / Inf = NaN
# A = 7f800000 | B = 7f800000 | OP = DIV | Result = 7fc00000 | Expected = 7fc00000 | PASS
```

[illegible]

3f800000	40400000	c0a00000	00000000	3f800000	7f800000		
40000000		40800000	3f800000	00000000	3f800000	7f800000	
3f000000	3fc00000	bfa00000	00000000	7f800000		7fc00000	

Chức năng: Module chính để thực hiện phép nhân hoặc chia hai số dấu phẩy động 32-bit theo chuẩn IEEE-754.

Cấu trúc gồm 5 giai đoạn:

1. Unpack: Tách dấu, số mũ, phần trị.
2. Exponent Unit: Tính số mũ mới (cộng hoặc trừ và trừ bias).
3. Mantissa Unit: Thực hiện nhân hoặc chia phần trị.
4. Normalization: Đưa kết quả phần trị về dạng chuẩn 1.F.
5. Rounding: Làm tròn kết quả và điều chỉnh số mũ nếu cần.