

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KỸ THUẬT MÁY TÍNH**



**THIẾT KẾ HỆ THỐNG SỐ VỚI HDL – CE213.P21**

**ĐỀ TÀI**

**Thiết kế mạch nhân/chia dấu chấm động**  
**Floating-point kernel/split circuit**

**Giáo viên hướng dẫn:** Thạc sĩ Hồ Ngọc Diễm

**Sinh viên:**

Dương Hiền Gia Khang

22520610

*Hồ Chí Minh, ngày 28 tháng 5 năm 2025*

## Contents

|  |    |
|--|----|
| MỤC LỤC.....                               | 2  |
| Chương 1. TỔNG QUAN ĐỀ TÀI.....            | 4  |
| 1.1. Tổng quan.....                        | 4  |
| 1.2. Mục tiêu nghiên cứu .....             | 4  |
| 1.2.2. Mục tiêu cụ thể .....               | 4  |
| 1.3. Giới hạn của đề tài .....             | 5  |
| Chương 2. THIẾT KẾ GIẢI PHÁP ĐỀ XUẤT ..... | 6  |
| 2.1. Lưu đồ thiết kế thuật toán.....       | 6  |
| 2.1.1. Mạch nhân/chia dấu chấm động.....   | 6  |
| 2.2. Phần cứng .....                       | 9  |
| 2.3. Modules .....                         | 9  |
| 2.3.1. Module Unpack .....                 | 9  |
| 2.3.1.2. Module exponent_unit.....         | 14 |
| 2.3.1.3. Module mantissa_unit .....        | 19 |
| 2.3.1.4. Module normalization .....        | 24 |
| 2.3.1.5. Module rounding .....             | 34 |
| 2.3.1.6. Module fp-arithmetic.....         | 39 |
| TÀI LIỆU THAM KHẢO.....                    | 52 |

## MỤC LỤC

## DANH MỤC HÌNH

|   |    |
|---|----|
| Hình 1.1: Hệ thống nhà thông minh – Smart Home .....  | 7  |
| Hình 1.2: Hệ thống Smart Agriculture .....            | 8  |
| Hình 1.3: Hệ thống Autonomous Vehicles .....          | 9  |
| Hình 1.4: Hệ thống Revolution Healing care.....       | 10 |
| Hình 2.1: Sơ đồ khối kiến trúc hệ thống .....         | 12 |
| Hình 2.2: Thiết kế sơ đồ phần cứng hệ thống .....     | 13 |
| Hình 2.3: Kit Nucleo STM32F401RE .....                | 13 |
| Hình 2.4: Cảm biến Nhiệt độ và Độ ẩm Si7020 .....     | 15 |
| Hình 2.5: Quang trở GL5537 .....                      | 16 |
| Hình 2.6: Màn hình TFT LCD .....                      | 17 |
| Hình 2.7: Buzzer .....                                | 18 |
| Hình 2.8: Đèn LED .....                               | 19 |
| Hình 2.7: Sơ đồ thuật toán vận hành hệ thống.....     | 20 |
| Hình 3.1: Màn hình hiển thị thông số cảm bị .....     | 23 |
| Bảng 3.2: Kết quả thực nghiệm hệ thống cảnh báo ..... | 2  |

## Chương 1. TỔNG QUAN ĐỀ TÀI

### 1.1. Tổng quan

- Trong các hệ thống tính toán hiện đại, việc xử lý các phép toán số dấu chấm động là điều thiết yếu, đặc biệt trong các lĩnh vực yêu cầu độ chính xác cao như đồ họa máy tính, trí tuệ nhân tạo, xử lý tín hiệu số (DSP), và khoa học tính toán. Chuẩn IEEE-754 được sử dụng rộng rãi để biểu diễn và thực hiện các phép toán với số dấu chấm động trong cả phần mềm và phần cứng. Tuy nhiên, so với các phép toán số nguyên, việc thực hiện các phép toán dấu chấm động phức tạp hơn do phải xử lý phần dấu, số mũ, và phần định trị (mantissa).
- Đề tài "Thiết kế mạch nhân và chia số dấu chấm động" hướng đến việc tìm hiểu và xây dựng các khối phần cứng có khả năng thực hiện phép nhân và chia theo chuẩn IEEE-754 (single precision – 32 bit), từ đó mô phỏng và kiểm chứng hoạt động trên phần mềm chuyên dụng như ModelSim hoặc Vivado. Qua đó, nhóm củng cố và vận dụng kiến thức đã học về thiết kế mạch số, Verilog HDL, cũng như hiểu rõ hơn về cách các phép toán dấu chấm động được xử lý ở mức phần cứng.

### 1.2. Mục tiêu nghiên cứu

#### 1.2.1. Mục tiêu tổng quát

Xây dựng một mạch phần cứng thực hiện phép nhân và chia số dấu chấm động theo chuẩn IEEE-754 (32-bit single precision), có khả năng mô phỏng và kiểm thử chính xác bằng phần mềm, từ đó hiểu sâu hơn về cơ chế xử lý số dấu chấm động trong phần cứng.

#### 1.2.2. Mục tiêu cụ thể

- Tìm hiểu lý thuyết về chuẩn IEEE-754 và nguyên lý thực hiện phép nhân, chia số dấu chấm động.
- Thiết kế các mô-đun chức năng: tách số (unpack), cộng số mũ, nhân/chia phần định trị, chuẩn hóa (normalize), làm tròn (rounding), và xử lý các trường hợp đặc biệt (NaN, Infinity, Zero).
- Mô phỏng và kiểm thử mạch nhân/chia bằng Verilog trên phần mềm ModelSim hoặc Vivado.
- Đánh giá kết quả so với kỳ vọng (Expected Output) từ bộ test case.

- Đưa ra các giải pháp cải tiến dựa trên kết quả kiểm tra.

### 1.3. Giới hạn của đề tài

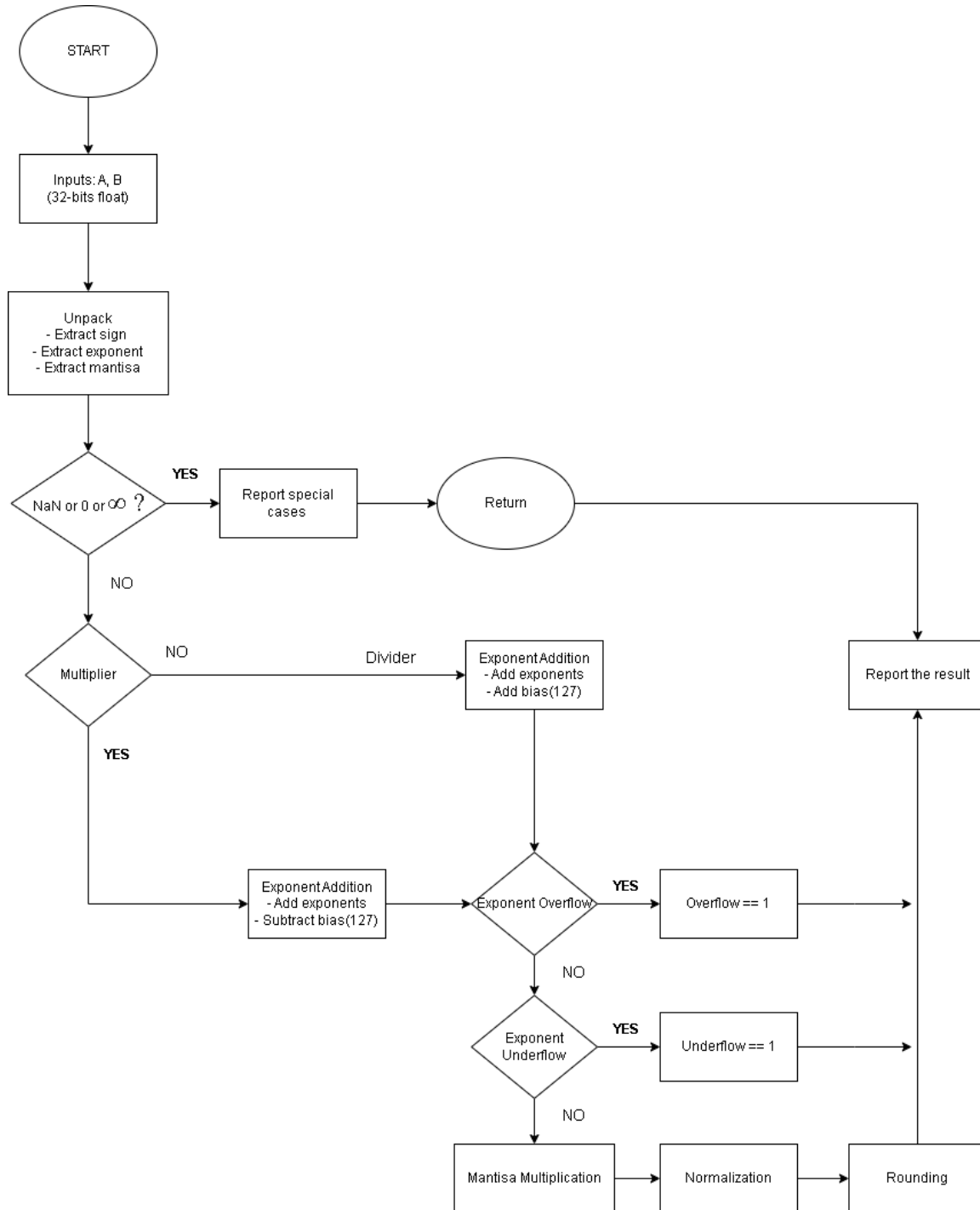
Do giới hạn về thời gian, nguồn lực và phạm vi kiến thức, đề tài thiết kế mạch nhân và chia số dấu chấm động chỉ tập trung vào một số nội dung cụ thể như sau:

- **Chỉ áp dụng cho chuẩn IEEE-754 single precision (32-bit):** Đề tài không mở rộng sang các chuẩn khác như double precision (64-bit) hay half precision (16-bit).
- **Chỉ thực hiện hai phép toán nhân và chia:** Các phép toán khác như cộng, trừ hoặc chuyển đổi giữa định dạng dấu chấm động và số nguyên không được thực hiện.
- **Không tối ưu hóa hiệu năng phần cứng:** Thiết kế chủ yếu nhằm đảm bảo độ chính xác và tính đúng đắn của phép toán, chưa tối ưu về tốc độ xử lý, diện tích mạch (area) hay mức tiêu thụ năng lượng.
- **Không tích hợp pipeline hoặc hỗ trợ song song:** Mạch được thiết kế hoạt động ở mức đơn chu kỳ hoặc đa chu kỳ đơn giản, không áp dụng kỹ thuật tăng hiệu năng như pipeline hay xử lý song song.
- **Giới hạn bộ kiểm thử (testbench):** Các test case sử dụng chủ yếu là các trường hợp cơ bản và một số trường hợp đặc biệt như zero, infinity, NaN, chưa bao phủ toàn bộ các trường hợp biên hoặc dữ liệu ngẫu nhiên lớn.

## Chương 2. THIẾT KẾ GIẢI PHÁP ĐỀ XUẤT

### 2.1. Lưu đồ thiết kế thuật toán

#### 2.1.1. Mạch nhân/chia dấu chấm động



Hình 1: Lưu đồ thuật toán mạch nhân dấu chấm động

Giải thích:

- **Unpack:** Giai đoạn này giải nén (unpack) các số dấu phẩy động thành các thành phần cấu tạo của chúng:
  - **Extract sign:** Trích xuất bit dấu (sign bit).
  - **Extract exponent:** Trích xuất phần mũ (exponent).
  - **Extract mantissa:** Trích xuất phần định trị (mantissa).
  
- **NaN or 0 or  $\infty$  ? (YES/NO):** Kiểm tra xem các số đầu vào có phải là các trường hợp đặc biệt như "Not a Number" (NaN), 0, hoặc vô cùng ( $\infty$ ) hay không.
  
- Kiểm tra xem đang thực hiện phép toán nhân hay chia hai số chấm động 32 bit.
  - Nếu  $op = 0$ , phép nhân
  - Nếu  $op = 1$ , phép chia
  
- **Exponent Addition:** Khi các số không phải là trường hợp đặc biệt:
  - **Add exponents:** Cộng các phần mũ của A và B.
  - **Add/Subtract bias (127):** Cộng/Trừ đi giá trị bias (độ lệch) là 127 – Add đối với phép chia và Subtract đối với phép nhân. Định dạng IEEE 754 single-precision (32-bit) có bias là 127.
  
- **Exponent overflow (YES/NO):** Kiểm tra xem có xảy ra tràn số ở phần mũ hay không.
  
- **Exponent underflow (YES/NO):** Kiểm tra xem có xảy ra dưới tràn số ở phần mũ hay không.
  
- **Mantissa Multiplication/Divider:**
  - **Multiply/Divide 24-bit mant:** 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.
  - **48-bit result:** Kết quả của phép nhân hai phần định trị 24-bit.
  - **27-bit result:** Kết quả của phép chia hai phần định trị 24-bit.

- **Normalization:** Chuẩn hóa kết quả sau phép nhân phần định trị để đảm bảo rằng phần định trị có dạng 1.xxxx... (đối với số chuẩn hóa) và điều chỉnh phần mũ tương ứng.
- **Rounding:** Làm tròn kết quả cuối cùng theo quy tắc làm tròn được định nghĩa trong chuẩn IEEE 754 (ví dụ: làm tròn đến số chẵn gần nhất).



## 2.2. Phần cứng

## 2.3. Modules

### 2.3.1. Module Unpack

```
module unpack(  
    input  [31:0] a, b,  
    output      sign_a, sign_b,  
    output [7:0] exp_a, exp_b,  
    output [23:0] mant_a, mant_b  
);  
    assign sign_a = a[31];  
    assign exp_a  = a[30:23];  
    assign mant_a = (exp_a == 0) ? {1'b0, a[22:0]} : {1'b1, a[22:0]};  
  
    assign sign_b = b[31];  
    assign exp_b  = b[30:23];  
    assign mant_b = (exp_b == 0) ? {1'b0, b[22:0]} : {1'b1, b[22:0]};  
endmodule
```

```
`timescale 1ns / 1ps  
  
module unpack_testbench;  
  
    // Khai báo các tín hiệu  
    reg [31:0] a, b;  
    wire sign_a, sign_b;  
    wire [7:0] exp_a, exp_b;  
    wire [23:0] mant_a, mant_b;  
  
    unpack uut (  
        .a(a),
```

```

.b(b),
.sign_a(sign_a),
.sign_b(sign_b),
.exp_a(exp_a),
.exp_b(exp_b),
.mant_a(mant_a),
.mant_b(mant_b)
);

initial begin
    // Test case 1:
    a = 32'h3F800000; // +1.0
    b = 32'h3FC00000; // +1.5
    #1;
    $display("Test case 1: So duong chuan");
    $display("a = %h (%f)", a, $bitstoreal(a));
    $display("b = %h (%f)", b, $bitstoreal(b));
    $display("sign_a = %b, exp_a = %h (%d), mant_a = %h", sign_a, exp_a, exp_a, mant_a);
    $display("sign_b = %b, exp_b = %h (%d), mant_b = %h", sign_b, exp_b, exp_b, mant_b);

    if (sign_a == 0 && exp_a == 8'h7F && mant_a == 24'h800000 &&
        sign_b == 0 && exp_b == 8'h7F && mant_b == 24'hC00000)
        $display("Test case 1 passed");
    else
        $display("Test case 1 failed");

    // Test case 2:
    a = 32'hBF800000; // -1.0
    b = 32'hC0D80000; // -6.5
    #1;
    $display("\nTest case 2: So am chuan");

```

```

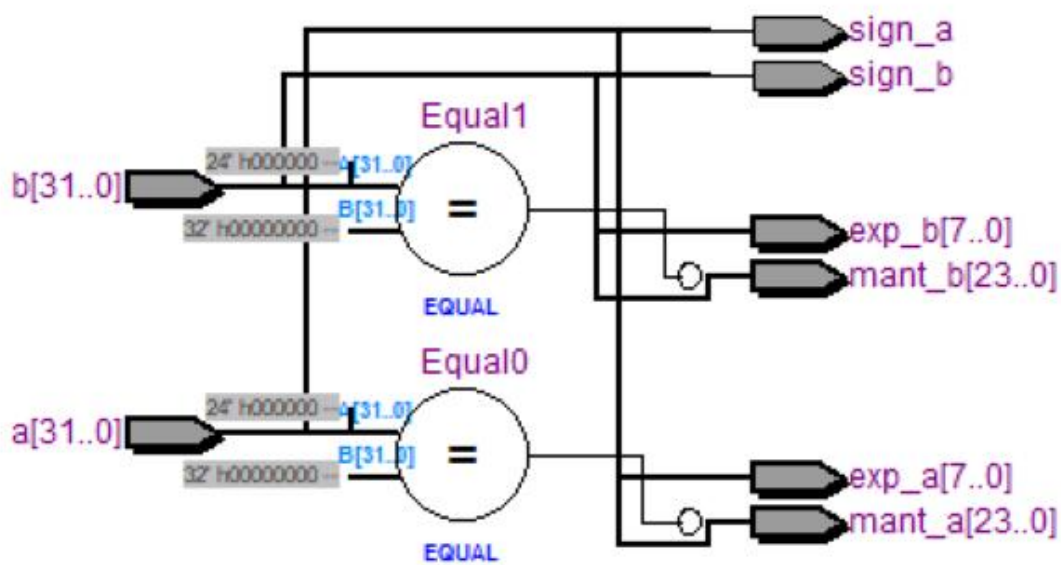
$display("a = %h (%f)", a, $bitstoreal(a));
$display("b = %h (%f)", b, $bitstoreal(b));
$display("sign_a = %b, exp_a = %h (%d), mant_a = %h", sign_a, exp_a, exp_a, mant_a);
$display("sign_b = %b, exp_b = %h (%d), mant_b = %h", sign_b, exp_b, exp_b, mant_b);

if (sign_a == 1 && exp_a == 8'h7F && mant_a == 24'h800000 &&
    sign_b == 1 && exp_b == 8'h81 && mant_b == 24'hd80000)
    $display("Test case 2 passed");
else
    $display("Test case 2 failed");

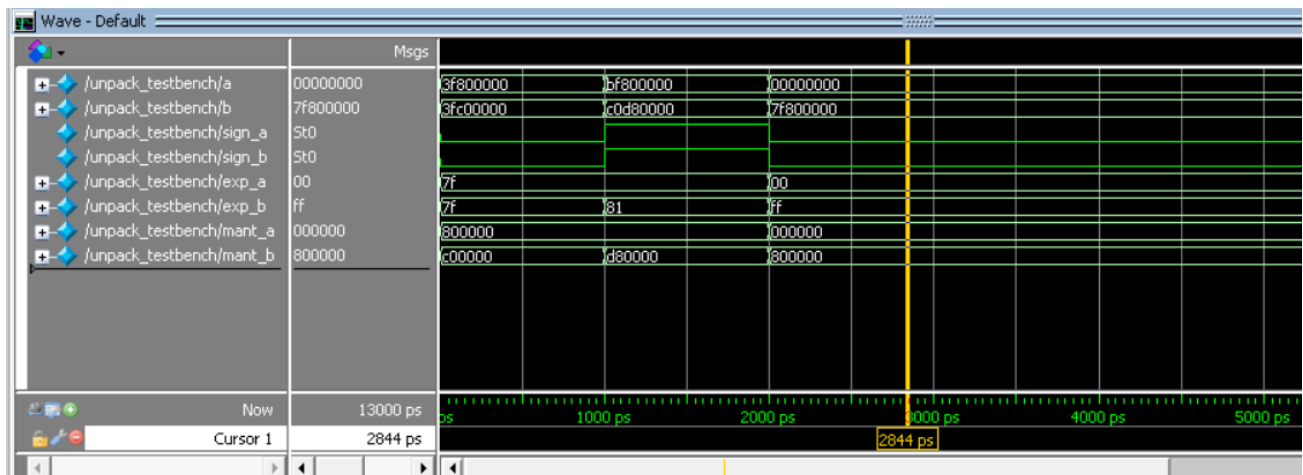
// Test case 3:
a = 32'h00000000; // +0.0
b = 32'h7F800000; // +?
#1;
$display("\nTest case 3: So khong va vo cuc");
$display("a = %h (%f)", a, $bitstoreal(a));
$display("b = %h (%f)", b, $bitstoreal(b));
$display("sign_a = %b, exp_a = %h (%d), mant_a = %h", sign_a, exp_a, exp_a, mant_a);
$display("sign_b = %b, exp_b = %h (%d), mant_b = %h", sign_b, exp_b, exp_b, mant_b);

if (sign_a == 0 && exp_a == 8'h00 && mant_a == 24'h000000 &&
    sign_b == 0 && exp_b == 8'hFF && mant_b == 24'h800000)
    $display("Test case 3 passed");
else
    $display("Test case 3 failed");
#10;
$finish;
end
endmodule

```



Hình 2: RTL view của module Unpack



Hình 3: Waveform của module Unpack

```

# 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
# 1

```

Hình 4: Terminal của module Unpack

Giải thích:

Mỗi số IEEE – 754 sẽ có dạng:

| Sign (1 bit) | Exponent (8 bits) | Fraction (23 bits) |

- Nếu **exponent** == 0 → là **số denormalized (không chuẩn hóa)**, nên mantissa là 0.f → không thêm bit ẩn.
  - Nếu **exponent** ≠ 0 → là **normalized** → thêm 1. trước mantissa → ta thêm bit 1 vào MSB
- Test case1:
- 0x3F800000 → sign = 0, exponent = 127 (0x7F), mantissa = 0x000000 → chuẩn hóa → thêm 1. → 0x800000
  - 0x3FC00000 → sign = 0, exponent = 127 (0x7F), mantissa = 0x400000 → chuẩn hóa → 0xC00000
- Test case2:
- a = 0xBF800000 // -1.0 → sign = 1, exp = 127 (0x7F), mantissa = 0x000000 → +bit 1 → 0x800000
  - b = 0xC0D80000 // -6.75 → sign = 1, exp = 129 (0x81), mantissa = 0x580000 → +bit 1 → 0xD80000

- Test case3:
  - $a = 0x00000000$  //  $+0.0 \rightarrow \text{sign} = 0, \text{exp} = 0 \rightarrow$  không thêm 1  $\rightarrow$  mantissa =  $0x0000000$
  - $b = 0x7F800000$  //  $+\infty \rightarrow \text{sign} = 0, \text{exp} = 0xFF, \text{mantissa} = 0x0000000 \rightarrow$  là số vô cực, nhưng theo unpack, mantissa sẽ là  $\{1'b1, 23'b0\} \rightarrow 0x8000000$

### 2.3.1.2. Module exponent\_unit

```

module exponent_unit (
    input [7:0] exp_a,
    input [7:0] exp_b,
    input      op_sel,    // 0 for add (multiply op), 1 for subtract (divide op)
    output reg [7:0] exp_result,
    output reg overflow,
    output reg underflow
);
    reg signed [9:0] exp_temp;

    always @(*) begin
        if (op_sel == 1'b0) begin // Multiply operation: exp_a + exp_b - bias
            exp_temp = $signed({1'b0, exp_a}) + $signed({1'b0, exp_b}) - 10'sd127;
        end else begin // Divide operation: exp_a - exp_b + bias
            exp_temp = $signed({1'b0, exp_a}) - $signed({1'b0, exp_b}) + 10'sd127;
        end

        overflow = 0;
        underflow = 0;
    end

```

```

        if (exp_temp > 255) begin
            exp_result = 8'hFF;
            overflow = 1;
        end else if (exp_temp < 0) begin
            exp_result = 8'h00;
            underflow = 1;
        end else begin
            exp_result = exp_temp[7:0];
        end
    end
end
endmodule

```

```

`timescale 1ns / 1ps

```

```

module tb_exponent_unit;

```

```

    // Inputs

```

```

    reg [7:0] exp_a;

```

```

    reg [7:0] exp_b;

```

```

    reg    op_sel;

```

```

    // Outputs

```

```

    wire [7:0] exp_result;

```

```

    wire overflow;

```

```

    wire underflow;

```

```

    // Instantiate the Unit Under Test (UUT)

```

```

    exponent_unit uut (

```

```

        .exp_a(exp_a),

```

```

        .exp_b(exp_b),

```

```

        .op_sel(op_sel),

```

```

.exp_result(exp_result),
.overflow(overflow),
.underflow(underflow)
);

initial begin
    $display("Time\tOp\tExp_A\tExp_B\tResult\tOverflow\tUnderflow");

    // Test multiply:  $100 + 100 - 127 = 73$ 
    exp_a = 8'd100;
    exp_b = 8'd100;
    op_sel = 0;
    #10 $display("%0dns\tMUL\t%d\t%d\t%d\t%b\t%b", $time, exp_a, exp_b, exp_result,
overflow, underflow);

    // Test divide:  $200 - 100 + 127 = 227$ 
    exp_a = 8'd200;
    exp_b = 8'd100;
    op_sel = 1;
    #10 $display("%0dns\tDIV\t%d\t%d\t%d\t%b\t%b", $time, exp_a, exp_b, exp_result,
overflow, underflow);

    // Test overflow in multiply:  $200 + 200 - 127 = 273 (> 255)$ 
    exp_a = 8'd200;
    exp_b = 8'd200;
    op_sel = 0;
    #10 $display("%0dns\tMUL\t%d\t%d\t%d\t%b\t%b", $time, exp_a, exp_b, exp_result,
overflow, underflow);

    // Test underflow in divide:  $50 - 200 + 127 = -23 (< 0)$ 
    exp_a = 8'd50;

```



```

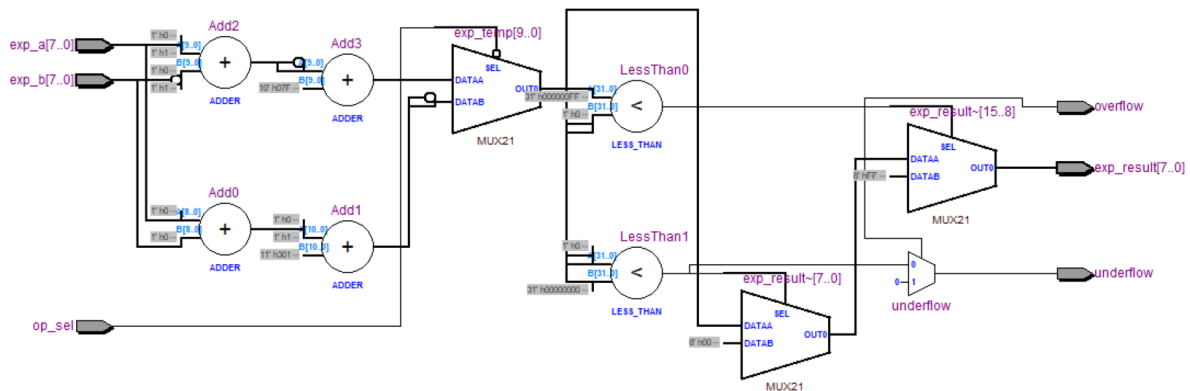
exp_b = 8'd200;
op_sel = 1;
#10 $display("%0dns\tDIV\t%d\t%d\t%d\t%b\t%b", $time, exp_a, exp_b, exp_result,
overflow, underflow);

// Test edge case: 127 + 127 - 127 = 127
exp_a = 8'd127;
exp_b = 8'd127;
op_sel = 0;
#10 $display("%0dns\tMUL\t%d\t%d\t%d\t%b\t%b", $time, exp_a, exp_b, exp_result,
overflow, underflow);

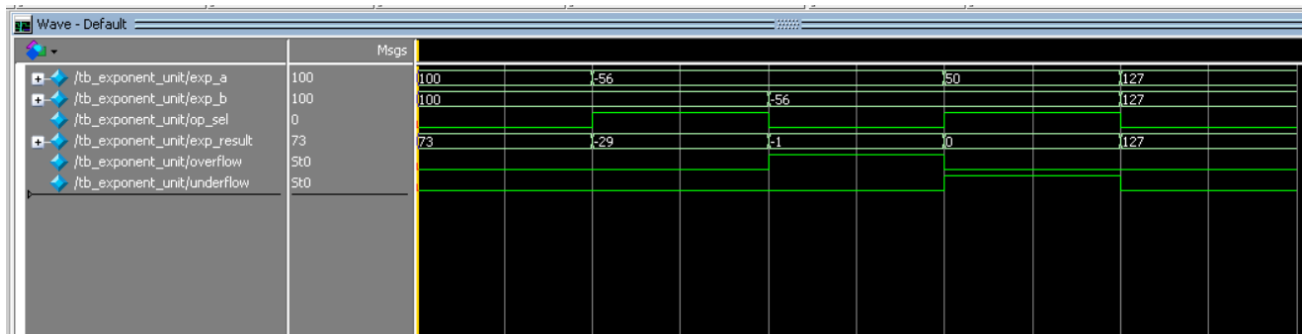
$finish;
end

endmodule

```



Hình 5: RTL view của module exponent\_unit



Hình 6: Waveform của module exponent\_unit

```

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

```

Hình 7: Terminal của module exponent\_unit

Giải thích:

- $\text{exp\_temp} = \text{\$signed}(\{1'b0, \text{exp\_a}\}) + \text{\$signed}(\{1'b0, \text{exp\_b}\}) - 10'sd127;$ 
  - Ghép thêm 1'b0 vào phía trước exp\_a, exp\_b để chuyển từ 8-bit không dấu sang 9-bit có dấu an toàn.
  - **Số mũ được cộng, rồi trừ bias (127)** để bù lại vì mỗi số ban đầu đã được cộng bias.
- $\text{exp\_temp} = \text{\$signed}(\{1'b0, \text{exp\_a}\}) - \text{\$signed}(\{1'b0, \text{exp\_b}\}) + 10'sd127;$ 
  - Số mũ được với nhau, sau đó cộng bias(127).
- Xử lý tràn:
  - Nếu  $\text{exp\_temp} > 255 \rightarrow$  tràn  $\rightarrow$  gán 0xFF, bật overflow.
  - Nếu  $\text{exp\_temp} < 0 \rightarrow$  thiếu  $\rightarrow$  gán 0x00, bật underflow.
  - Nếu trong khoảng  $[0, 255] \rightarrow$  bình thường  $\rightarrow$  cắt  $\text{exp\_temp}[7:0]$ .

Giải thích testbench

| Tes<br>t | Mô tả   | Kết quả mong đợi                       |
|----------|---|--|
| 1        | Nhân $100 \times 100 \rightarrow 100 + 100 - 127 = 73$  | Không tràn, không underflow            |
| 2        | Chia $200 / 100 \rightarrow 200 - 100 + 127 = 227$      | Không tràn, không underflow            |
| 3        | Nhân $200 \times 200 \rightarrow 400 - 127 = 273 > 255$ | Tràn ( $\text{overflow} = 1$ )         |
| 4        | Chia $50 / 200 \rightarrow 50 - 200 + 127 = -23$        | Dưới ngưỡng ( $\text{underflow} = 1$ ) |
| 5        | Nhân $127 \times 127 \rightarrow 127$                   | Trường hợp cân bằng                    |

### 2.3.1.3. Module mantissa\_unit

```

module mantissa_multiplier(
    input [23:0] mant_a,
    input [23:0] mant_b,
    output [47:0] mant_result
);
    assign mant_result = mant_a * mant_b;
endmodule

module mantissa_divider(
    input [23:0] mant_a,
    input [23:0] mant_b,
    output [26:0] mant_quotient
);

    wire [49:0] extended_mant_a;
    assign extended_mant_a = {mant_a, 26'b0};

```

```

wire [49:0] temp_quotient;

assign temp_quotient = (mant_b == 0) ? {50{1'b1}} : (extended_mant_a / mant_b) ; // Avoid
division by zero here, though FPU should catch it

assign mant_quotient = temp_quotient[26:0];

endmodule

```

```

module mantissa_unit (
    input [23:0] mant_a,
    input [23:0] mant_b,
    output [47:0] product_result,
    output [26:0] quotient_result
);

    mantissa_multiplier u_mm (
        .mant_a(mant_a),
        .mant_b(mant_b),
        .mant_result(product_result)
    );

    mantissa_divider u_md (
        .mant_a(mant_a),
        .mant_b(mant_b),
        .mant_quotient(quotient_result)
    );

endmodule

```

```

`timescale 1ns / 1ps
module tb_mantissa_unit;

    // Inputs

```

```

reg [23:0] mant_a;
reg [23:0] mant_b;

// Outputs
wire [47:0] product_result;
wire [26:0] quotient_result;

// Instantiate the Unit Under Test (UUT)
mantissa_unit uut (
    .mant_a(mant_a),
    .mant_b(mant_b),
    .product_result(product_result),
    .quotient_result(quotient_result)
);

// Helper task to print binary values in readable form
task print_results;
    input [23:0] a, b;
    input [47:0] prod;
    input [26:0] quot;
    begin
        $display("mant_a = %d, mant_b = %d", a, b);
        $display("→ Product = %d (0x%h)", prod, prod);
        $display("→ Quotient = %d (0x%h)", quot, quot);
        $display("-----");
    end
endtask

initial begin
    $display("Starting mantissa_unit testbench...");
    $display("-----");

```

```

// Case 1: Normal values
mant_a = 24'd1000000;
mant_b = 24'd500000;
#10 print_results(mant_a, mant_b, product_result, quotient_result);

// Case 2: mant_b = 1
mant_a = 24'd1234567;
mant_b = 24'd1;
#10 print_results(mant_a, mant_b, product_result, quotient_result);

// Case 3: mant_a = mant_b
mant_a = 24'd7654321;
mant_b = 24'd7654321;
#10 print_results(mant_a, mant_b, product_result, quotient_result);

// Case 4: mant_b = 0 (division by zero)
mant_a = 24'd1000;
mant_b = 24'd0;
#10 print_results(mant_a, mant_b, product_result, quotient_result);

// Case 5: Very small / very large
mant_a = 24'd1;
mant_b = 24'd10000000;
#10 print_results(mant_a, mant_b, product_result, quotient_result);

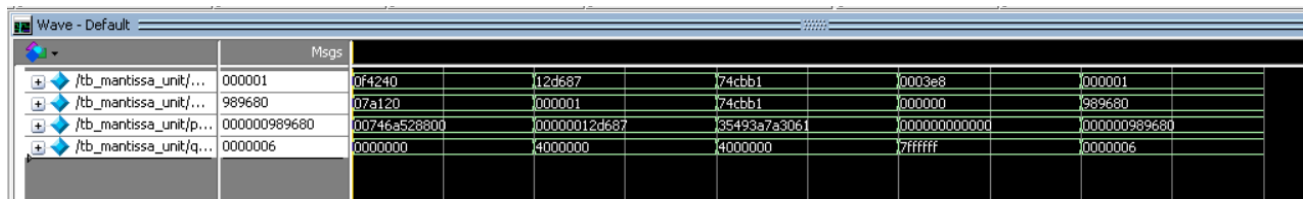
$display("Testbench finished.");
$finish;
end

endmodule

```



Hình 8: RTL view của module `mantissa_unit`



Hình 9: Waveform của module `mantissa_unit`

```

vSIM7> 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 (0x000000000000)
# ? 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.v(64)
# Time: 50 ns Iteration: 0 Instance: /tb_mantissa_unit

```

Hình 10: Terminal của module mantissa\_unit

Giải thích:

- (Phép nhân) mant\_result = mant\_a \* mant\_b;
- (Phép chia) mant\_result = mant\_a / mant\_b
  - mant\_a, mant\_b: là **định trị 24-bit** đã chuẩn hóa (bao gồm cả bit ẩn là 1 ở đầu).
  - Phần định trị (mantissa) mặc định là **1.xxxxxx** (nếu không phải số dưới chuẩn).
  - 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.

#### 2.3.1.4. Module normalization

```

module normalization(
    input    is_divide_op, // 0 for multiply, 1 for divide
    input [47:0] mult_mant_result, // Input from mantissa_multiplier
    input [26:0] div_mant_quotient, // Input from mantissa_divider (Q[26].Q[25:0])

```



```

input [7:0] exp_in,          // Current exponent sum/subtraction
output reg [26:0] normalized_mant, // 24-bit mantissa (1.F) + 3 GRS bits
output reg [7:0] normalized_exp,
output reg      norm_caused_underflow, // Flag if exponent becomes < 1 (denormal/zero)
output reg      norm_caused_overflow  // Flag if exponent becomes > FF
);

reg [7:0] shift_amount; // For more advanced normalization (not fully implemented here)

integer i;
reg found_msb;

always @(*) begin
    normalized_exp = exp_in;
    norm_caused_underflow = 1'b0;
    norm_caused_overflow = 1'b0;

    if (is_divide_op) begin
        if (div_mant_quotient == 0) begin
            normalized_mant = 27'b0;
            normalized_exp = 8'h00;
            norm_caused_underflow = 1'b1;
        end else if (div_mant_quotient[26] == 1'b1) begin
            normalized_mant = div_mant_quotient;
        end else begin
            if (normalized_exp == 0) begin
                normalized_mant = div_mant_quotient << 1;
                norm_caused_underflow = (exp_in <= 1);
            end else if (exp_in == 8'hFF) begin
                normalized_mant = div_mant_quotient;
                norm_caused_overflow = 1'b1;
            end
        end
    end
end

```

```

end else begin
    normalized_mant = div_mant_quotient << 1;
    normalized_exp = exp_in - 1;
    if (normalized_exp == 0 && exp_in == 1) begin
        norm_caused_underflow = 1'b1;
    end
end
end
end else begin // Multiply operation
    if (mult_mant_result == 0) begin
        normalized_mant = 27'b0;
        normalized_exp = 8'h00;
        norm_caused_underflow = 1'b1;
    end else if (mult_mant_result[47]) begin
        normalized_mant = mult_mant_result[47:21];
        if (normalized_exp == 8'hFF) begin
            norm_caused_overflow = 1'b1;
        end else begin
            normalized_exp = exp_in + 1;
        end
    end else if (mult_mant_result[46]) begin
        normalized_mant = mult_mant_result[46:20];
    end else begin
        found_msb = 0;
        shift_amount = 0;
        normalized_mant = 27'b0;

        for (i = 45; i >= 20; i = i - 1) begin
            if (mult_mant_result[i] && !found_msb) begin
                shift_amount = 46 - i;
                found_msb = 1;
            end
        end
    end
end
end

```

```

        if ((exp_in < shift_amount) && (exp_in != 0)) begin
            norm_caused_underflow = 1'b1;
            normalized_exp = 0;
            normalized_mant = (mult_mant_result << shift_amount) >> (i - 19);
        end else if ((exp_in == 0) && (shift_amount > 0)) begin
            norm_caused_underflow = 1'b1;
            normalized_exp = 0;
            normalized_mant = 27'b0;
        end else begin
            normalized_mant = (mult_mant_result << shift_amount) >> 20;
            normalized_exp = exp_in - shift_amount;
            if ((normalized_exp == 0) && (exp_in >= 1) && (exp_in < shift_amount + 1))
begin
                norm_caused_underflow = 1'b1;
            end
        end
    end
end

    if ((!found_msb) && (mult_mant_result != 0)) begin
        norm_caused_underflow = 1'b1;
        normalized_exp = 8'h00;
        normalized_mant = 27'b0;
    end else if (mult_mant_result == 0) begin
        norm_caused_underflow = 1'b1;
        normalized_exp = 8'h00;
        normalized_mant = 27'b0;
    end
end
end
end

```

```

        if ((normalized_exp == 8'hFF) && (!norm_caused_overflow)) begin
            // Usually Inf/NaN state
        end else if ((normalized_exp == 8'h00) && (!norm_caused_underflow) && (exp_in > 0)) begin
            norm_caused_underflow = 1'b1;
        end
    end
endmodule

```

```

`timescale 1ns/1ps

```

```

module tb_normalization;

```

```

    // Inputs

```

```

    reg    is_divide_op;

```

```

    reg [47:0] mult_mant_result;

```

```

    reg [26:0] div_mant_quotient;

```

```

    reg [7:0] exp_in;

```

```

    // Outputs

```

```

    wire [26:0] normalized_mant;

```

```

    wire [7:0] normalized_exp;

```

```

    wire    norm_caused_underflow;

```

```

    wire    norm_caused_overflow;

```

```

    // DUT

```

```

    normalization uut (

```

```

        .is_divide_op(is_divide_op),

```

```

        .mult_mant_result(mult_mant_result),

```

```

        .div_mant_quotient(div_mant_quotient),

```

```

        .exp_in(exp_in),

```

```

        .normalized_mant(normalized_mant),
    );
endmodule

```

```

.normalized_exp(normalized_exp),
.norm_caused_underflow(norm_caused_underflow),
.norm_caused_overflow(norm_caused_overflow)
);

task print_result;
    input [47:0] mult_val;
    input [26:0] div_val;
    input [7:0] exp_val;
    input is_div;
    begin
        $display("----- %s Test -----", is_div ? "Divide" : "Multiply");
        $display("Mantissa Input : %h", is_div ? div_val : mult_val);
        $display("Exponent In    : %d", exp_val);
        #5;
        $display("Normalized Mant : %h", normalized_mant);
        $display("Normalized Exp  : %d", normalized_exp);
        $display("Underflow      : %b", norm_caused_underflow);
        $display("Overflow       : %b", norm_caused_overflow);
        $display("-----\n");
    end
endtask

initial begin
    $display("Running normalization testbench...\n");

    // Multiply case: MSB at bit 47 → expect exp + 1
    is_divide_op = 0;
    mult_mant_result = 48'h8000_0000_0000; // MSB=1 at 47
    div_mant_quotient = 0;
    exp_in = 8'd127;

```

```

print_result(mult_mant_result, div_mant_quotient, exp_in, is_divide_op);

// Multiply case: MSB at bit 46 → keep exp
mult_mant_result = 48'h4000_0000_0000; // MSB=1 at 46
print_result(mult_mant_result, div_mant_quotient, exp_in, is_divide_op);

// Multiply case: MSB at bit 43 → shift left 3 bits
mult_mant_result = 48'h1000_0000_0000; // MSB=1 at 43
print_result(mult_mant_result, div_mant_quotient, exp_in, is_divide_op);

// Multiply case: zero result → underflow
mult_mant_result = 48'd0;
print_result(mult_mant_result, div_mant_quotient, exp_in, is_divide_op);

// Divide case: already normalized
is_divide_op = 1;
div_mant_quotient = 27'h4000000; // bit 26 = 1
mult_mant_result = 0;
exp_in = 8'd127;
print_result(mult_mant_result, div_mant_quotient, exp_in, is_divide_op);

// Divide case: bit 26 = 0 → shift left
div_mant_quotient = 27'h2000000; // bit 25 = 1
print_result(mult_mant_result, div_mant_quotient, exp_in, is_divide_op);

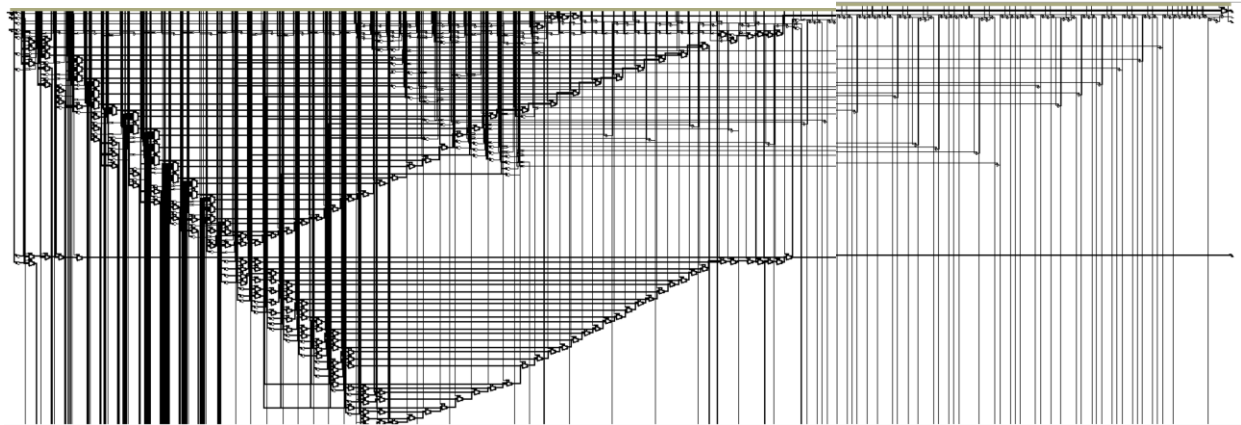
// Divide case: quotient = 0 → underflow
div_mant_quotient = 27'd0;
print_result(mult_mant_result, div_mant_quotient, exp_in, is_divide_op);

$display("Testbench finished.");
$finish;

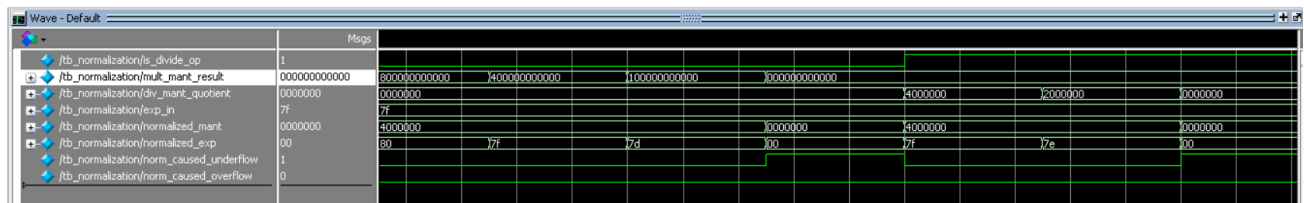
```

```
end
```

```
endmodule
```



Hình 11: RTL view của module normalization



Hình 12: Waveform của module normalization

|   |   |
|---|---|
| <pre> ----- Multiply Test ----- Mantissa Input  : 8000000000000 Exponent In     : 127 Normalized Mant : 4000000 Normalized Exp   : 128 Underflow       : 0 Overflow        : 0 -----  ----- Multiply Test ----- Mantissa Input  : 4000000000000 Exponent In     : 127 Normalized Mant : 4000000 Normalized Exp   : 127 Underflow       : 0 Overflow        : 0 -----  ----- Multiply Test ----- Mantissa Input  : 1000000000000 Exponent In     : 127 Normalized Mant : 4000000 Normalized Exp   : 125 Underflow       : 0 Overflow        : 0 -----  ----- Multiply Test ----- Mantissa Input  : 0000000000000 Exponent In     : 127 Normalized Mant : 0000000 Normalized Exp   : 0 Underflow       : 1 Overflow        : 0 ----- </pre> | <pre> ----- Divide Test ----- Mantissa Input  : 0000040000000 Exponent In     : 127 Normalized Mant : 4000000 Normalized Exp   : 127 Underflow       : 0 Overflow        : 0 -----  ----- Divide Test ----- Mantissa Input  : 0000020000000 Exponent In     : 127 Normalized Mant : 4000000 Normalized Exp   : 126 Underflow       : 0 Overflow        : 0 -----  ----- Divide Test ----- Mantissa Input  : 0000000000000 Exponent In     : 127 Normalized Mant : 0000000 Normalized Exp   : 0 Underflow       : 1 Overflow        : 0 ----- </pre> |
|---|---|

Hình 13: Waveform của module normalization

Giải thích:

Module này thực hiện **chuẩn hóa (normalization)** cho mantissa và exponent trong phép toán dấu phẩy động IEEE-754 sau khi:

- Nhân (multiply): kết quả mantissa có thể lớn hơn 24 bit, cần dịch phải.
- Chia (divide): mantissa thường nhỏ, cần dịch trái để đưa về dạng chuẩn 1.xxx.

#### Input

- is\_divide\_op: chọn phép chia (1) hoặc nhân (0)
- mult\_mant\_result: kết quả phép nhân (48-bit)
- div\_mant\_quotient: kết quả phép chia (27-bit)



- exp\_in: exponent đã được tính toán từ trước

### Output

- normalized\_mant: 27-bit mantissa đã chuẩn hóa (gồm cả 3 bit GRS)
- normalized\_exp: exponent sau chuẩn hóa
- norm\_caused\_underflow: true nếu exponent dưới 0
- norm\_caused\_overflow: true nếu exponent vượt quá 255

## 2. Tóm tắt thuật toán chuẩn hóa

### - Nhân (is\_divide\_op = 0)

- Nếu bit MSB là 1 tại bit 47 → shift phải 1, cộng exp +1
- Nếu MSB ở bit 46 → giữ nguyên
- Nếu MSB thấp hơn: tìm vị trí 1, dịch trái, trừ exponent
- Nếu không có bit 1 → kết quả là 0 → underflow

### - Chia (is\_divide\_op = 1)

- Nếu bit 26 (MSB của mantissa chia) là 1 → đã chuẩn
- Nếu nhỏ hơn, dịch trái 1 bit, trừ exponent
- Nếu mant\_b = 0 từ trước đó → không tính normalization (được xử lý ngoài)

Testbench:

| # | Trường hợp          | Input              | Kết quả mong đợi                              |
|---|---------------------|--------------------|---|
| 1 | Nhân – MSB ở bit 47 | 48'h8000_0000_0000 | Phải shift phải 1 bit và <b>tăng exponent</b> |
| 2 | Nhân – MSB ở bit 46 | 48'h4000_0000_0000 | Không cần shift, giữ nguyên exponent          |
| 3 | Nhân – MSB ở bit 43 | 48'h1000_0000_0000 | Dịch trái 3 bit, <b>giảm exponent</b>         |
| 4 | Nhân – Mantissa = 0 | 48'd0              | Kết quả là 0, underflow = 1                   |
| 5 | Chia – bit 26 = 1   | 27'h4000000        | Đã chuẩn, giữ nguyên exponent                 |
| 6 | Chia – bit 25 = 1   | 27'h2000000        | Dịch trái 1 bit, <b>giảm exponent</b>         |
| 7 | Chia – Quotient = 0 | 27'd0              | Kết quả là 0, underflow = 1                   |

### 2.3.1.5. Module rounding

```
module rounding(
    input [26:0] normalized_mant, // 1.F + GRS (bit 26 is hidden '1', bits 2:0 are G,R,S)
    input [7:0] normalized_exp,
    output reg [22:0] rounded_mant, // Final 23-bit fraction
    output reg [7:0] rounded_exp,
    output reg round_causes_overflow // To indicate if exponent needs to be FF
);

reg [24:0] temp_mant_to_round; // Holds {1'b0, 1.F} from normalized_mant for rounding
reg G, R, S, LSB;

always @(*) begin
    rounded_exp = normalized_exp; // Default
    round_causes_overflow = 1'b0;

    LSB = normalized_mant[3];
    G = normalized_mant[2];
    R = normalized_mant[1];
    S = normalized_mant[0];

    temp_mant_to_round = {1'b0, normalized_mant[26:3]}; // 25 bits total
    if (G && (R || S || LSB)) begin
        temp_mant_to_round = temp_mant_to_round + 1;
    end

    if (temp_mant_to_round[24]) begin
        rounded_mant = 23'h000000;
    end
end
```

```

        if (rounded_exp == 8'hFE) begin // Max normal exponent before Inf
            rounded_exp = 8'hFF; // Becomes Infinity
            round_causes_overflow = 1'b1;
        end else if (rounded_exp == 8'hFF) begin // Already Inf/NaN exp
            round_causes_overflow = 1'b1; // Stays Inf/NaN
        end else begin
            rounded_exp = rounded_exp + 1;
        end
    end else begin
        rounded_mant = temp_mant_to_round[22:0];
    end

    if (rounded_exp == 0 && normalized_exp > 0 && !round_causes_overflow) begin
        end
    end
endmodule

```

```

`timescale 1ns/1ps

```

```

module tb_rounding;
    reg [26:0] normalized_mant;
    reg [7:0] normalized_exp;
    wire [22:0] rounded_mant;
    wire [7:0] rounded_exp;

    rounding uut (
        .normalized_mant(normalized_mant),
        .normalized_exp(normalized_exp),
        .rounded_mant(rounded_mant),
        .rounded_exp(rounded_exp)
    );
endmodule

```

```

initial begin

    $display("Time\tNorm_Mant\tNorm_Exp\tRounded_Mant\tRounded_Exp");

    // Test case 1: No rounding, G=0
    normalized_mant = 27'b00000000000000000000000000000000; // All zero GRS
    normalized_exp = 8'd10;
    #10;
    $display("%0\t\t%b\t%d\t%h\t%d", $time, normalized_mant, normalized_exp, rounded_mant,
rounded_exp);

    // Test case 2: G=1, R=0, S=0 ? round up (tie, round to even)
    normalized_mant = {24'b00000000000000000000000000000001, 3'b100}; // mantissa LSB = 1, G=1,
R=0, S=0
    normalized_exp = 8'd10;
    #10;
    $display("%0\t\t%b\t%d\t%h\t%d", $time, normalized_mant, normalized_exp, rounded_mant,
rounded_exp);

    // Test case 3: G=1, R=1, S=0 ? round up
    normalized_mant = {24'b00000000000000000000000000000010, 3'b110}; // G=1,R=1,S=0
    normalized_exp = 8'd10;
    #10;
    $display("%0\t\t%b\t%d\t%h\t%d", $time, normalized_mant, normalized_exp, rounded_mant,
rounded_exp);

    // Test case 4: G=1, R=0, S=1 ? round up
    normalized_mant = {24'b00000000000000000000000000000011, 3'b101}; // G=1,R=0,S=1
    normalized_exp = 8'd10;
    #10;
    $display("%0\t\t%b\t%d\t%h\t%d", $time, normalized_mant, normalized_exp, rounded_mant,
rounded_exp);

```

```

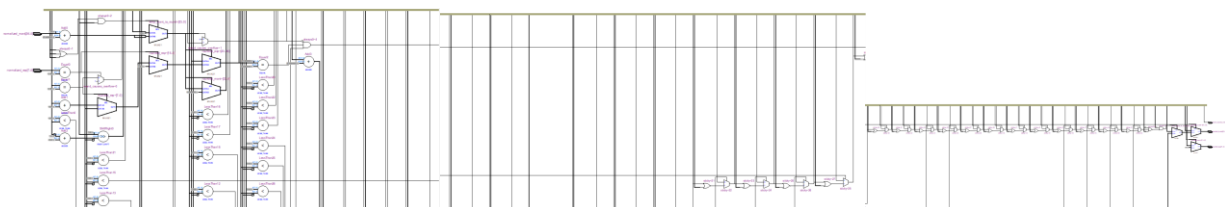
rounded_exp);

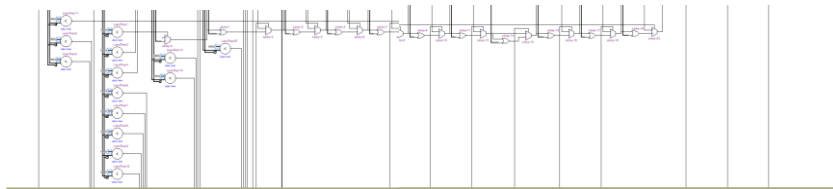
// Test case 5: G=0, R=1, S=1 ? no round up (G=0)
normalized_mant = {24'b000000000000000000000000100, 3'b011}; // G=0,R=1,S=1
normalized_exp = 8'd10;
#10;
$display("%0t\t%b\t%d\t%h\t%d", $time, normalized_mant, normalized_exp, rounded_mant,
rounded_exp);

// Test case 6: Check overflow mantissa after rounding
normalized_mant = {23'b111111111111111111111111, 1'b1, 3'b100}; // Mantissa all 1 +
G=1,R=0,S=0
normalized_exp = 8'd100;
#10;
$display("%0t\t%b\t%d\t%h\t%d", $time, normalized_mant, normalized_exp, rounded_mant,
rounded_exp);

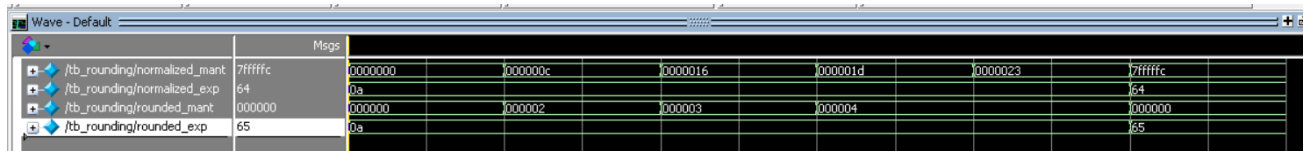
$finish;
end
endmodule

```





Hình 14: RTL view của module rounding



Hình 15: Waveform của module rounding

```
# Time   Norm_Mant      Norm_Exp      Rounded_Mant  Rounded_Exp
# 10000  00000000000000000000000000000000      10      000000      10
# 20000  000000000000000000000000000001100      10      000002      10
# 30000  0000000000000000000000000000010110      10      000003      10
# 40000  0000000000000000000000000000011101      10      000004      10
# 50000  00000000000000000000000000000100011      10      000004      10
# 60000  111111111111111111111111111111100      100      000000      101
# ** Note: $finish      : E:/HK2_nam3/HDL/FLT_MULTIPLEXER/rounder_tb.v(55)
#   Time: 60 ns  Iteration: 0  Instance: /tb_rounding
```

Hình 16: Waveform của module mantissa\_unit

Giải thích:

**Input:**

- normalized\_mant (27-bit) = 24-bit mantissa + 3 bit GRS.
- normalized\_exp = số mũ tương ứng.
- 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.

- Kiểm tra tràn :

- Nếu temp\_mant[24] = 1 → số bị tròn lên thành **2.0**:
- Dịch phải 1 bit.
- Tăng số mũ lên 1.

Testbench:

| Case | G R S | LSB   | Làm tròn? | Ghi chú   |
|------|-------|-------|-----------|---|
| 1    | 0 0 0 | 0     | Không     | Không cần làm tròn  |
| 2    | 1 0 0 | 1     | Có        | Tie + LSB = 1 → làm tròn lên (even)                       |
| 3    | 1 1 0 | 0     | Có        | G=1 và R=1  |
| 4    | 1 0 1 | 0     | Có        | G=1 và S=1  |
| 5    | 0 1 1 | x     | Không     | G=0 → không xét   |
| 6    | 1 0 0 | LSB=1 | Có        | Tròn lên gây <b>tràn mantissa</b> → dịch phải, tăng số mũ |

### 2.3.1.6. Module fp-arithmetic

```

module fp_arithmetic(
    input [31:0] a_in,
    input [31:0] b_in,
    input      op_sel, // 0 for multiply, 1 for divide
    output [31:0] result_out

```

```

);

// Unpack stage
wire sign_a, sign_b;
wire [7:0] exp_a, exp_b;
wire [23:0] mant_a, mant_b;

unpack u_unpack (
    .a(a_in), .b(b_in),
    .sign_a(sign_a), .sign_b(sign_b),
    .exp_a(exp_a), .exp_b(exp_b),
    .mant_a(mant_a), .mant_b(mant_b)
);

// Sign logic
wire sign_result = sign_a ^ sign_b;

// Special case detection (logic remains similar to previous fp_unit)
wire is_zero_a = (a_in[30:0] == 31'b0);
wire is_zero_b = (b_in[30:0] == 31'b0);
wire is_inf_a = (exp_a == 8'hFF) && (mant_a[22:0] == 0);
wire is_inf_b = (exp_b == 8'hFF) && (mant_b[22:0] == 0);
wire is_nan_a = (exp_a == 8'hFF) && (mant_a[22:0] != 0);
wire is_nan_b = (exp_b == 8'hFF) && (mant_b[22:0] != 0);

wire op_is_multiply = ~op_sel;
wire op_is_divide = op_sel;

wire R_is_nan, R_is_inf, R_is_zero;
// ... (logic R_is_nan, R_is_inf, R_is_zero nh? trong fp_unit tr??c) ...
assign R_is_nan = is_nan_a || is_nan_b ||

```



```

        (op_is_multiply && ((is_inf_a && is_zero_b) || (is_inf_b && is_zero_a))) ||
        (op_is_divide && ((is_zero_a && is_zero_b) || (is_inf_a && is_inf_b)));

    assign R_is_inf = (op_is_multiply && (is_inf_a || is_inf_b) && !( (is_inf_a && is_zero_b) ||
(is_inf_b && is_zero_a) )) ||
        (op_is_divide && ( (is_inf_a && !is_inf_b && !is_zero_b && !is_nan_b) ||
        (!is_inf_a && !is_zero_a && !is_nan_a && is_zero_b) ) );

    assign R_is_zero = (op_is_multiply && (is_zero_a || is_zero_b) && !( (is_inf_a && is_zero_b) ||
(is_inf_b && is_zero_a) )) ||
        (op_is_divide && ( (is_zero_a && !is_zero_b && !is_inf_b && !is_nan_b) ||
        (!is_inf_a && !is_nan_a && is_inf_b) ) );

    reg final_sign;

// Exponent Unit
    wire [7:0] current_exp_calc_res;
    wire current_exp_ovf, current_exp_udf;

    exponent_unit u_exp_unit (
        .exp_a(exp_a),
        .exp_b(exp_b),
        .op_sel(op_sel), // Pass op_sel to select add or subtract
        .exp_result(current_exp_calc_res),
        .overflow(current_exp_ovf),
        .underflow(current_exp_udf)
    );

// Mantissa Unit
    wire [47:0] mant_prod_val;
    wire [26:0] mant_div_val;

```

```

mantissa_unit u_mant_unit (
    .mant_a(mant_a),
    .mant_b(mant_b),
    .product_result(mant_prod_val),
    .quotient_result(mant_div_val)
);

// Normalization stage
wire [26:0] norm_mant;
wire [7:0] norm_exp;
wire      norm_udf, norm_ovf;

normalization u_norm (
    .is_divide_op(op_sel),
    .mult_mant_result(mant_prod_val), // Pass product from mantissa_unit
    .div_mant_quotient(mant_div_val), // Pass quotient from mantissa_unit
    .exp_in(current_exp_calc_res), // Pass result from exponent_unit
    .normalized_mant(norm_mant),
    .normalized_exp(norm_exp),
    .norm_caused_underflow(norm_udf),
    .norm_caused_overflow(norm_ovf)
);

// Rounding stage
wire [22:0] round_mant;
wire [7:0] round_exp;
wire      round_ovf;

rounding u_round (
    .normalized_mant(norm_mant),
    .normalized_exp(norm_exp),

```

```

        .rounded_mant(round_mant),
        .rounded_exp(round_exp),
        .round_causes_overflow(round_ovf)
    );

// Final Result Assembly
reg [31:0] result_temp;

always @(*) begin
    if (R_is_nan) begin
        result_temp = 32'h7FC00000; // Quiet NaN
    end else if (R_is_inf || current_exp_ovf || norm_ovf || round_ovf) begin
        final_sign = (op_is_divide && !is_inf_a && !is_nan_a && is_zero_b) ? sign_a : sign_result;
        result_temp = {final_sign, 8'hFF, 23'h000000};
    end else if (R_is_zero || current_exp_udf || norm_udf) begin
        result_temp = {sign_result, 8'h00, 23'h000000};
    end else begin
        result_temp = {sign_result, round_exp, round_mant};
    end
end

assign result_out = result_temp;

endmodule

```

---

```

`timescale 1ns / 1ps

module fp_arithmetic_tb;

    // Inputs
    reg [31:0] a_in;
    reg [31:0] b_in;

```

```

reg      op_sel; // 0 = MUL, 1 = DIV

// Output
wire [31:0] result_out;

// Instantiate the Unit Under Test (UUT)
fp_arithmetic uut (
    .a_in(a_in),
    .b_in(b_in),
    .op_sel(op_sel),
    .result_out(result_out)
);

// Task for displaying result
task test_case(input [31:0] a, input [31:0] b, input op, input [31:0] expected, input [255:0]
description);
    begin
        a_in = a;
        b_in = b;
        op_sel = op;
        #10; // Wait for operation

        $display("Test: %s", description);
        $display("A = %h | B = %h | OP = %s | Result = %h | Expected = %h | %s\n",
            a, b, (op ? "DIV" : "MUL"), result_out, expected,
            (result_out === expected ? "PASS" : "FAIL"));
    end
endtask

// Begin tests
initial begin

```

```

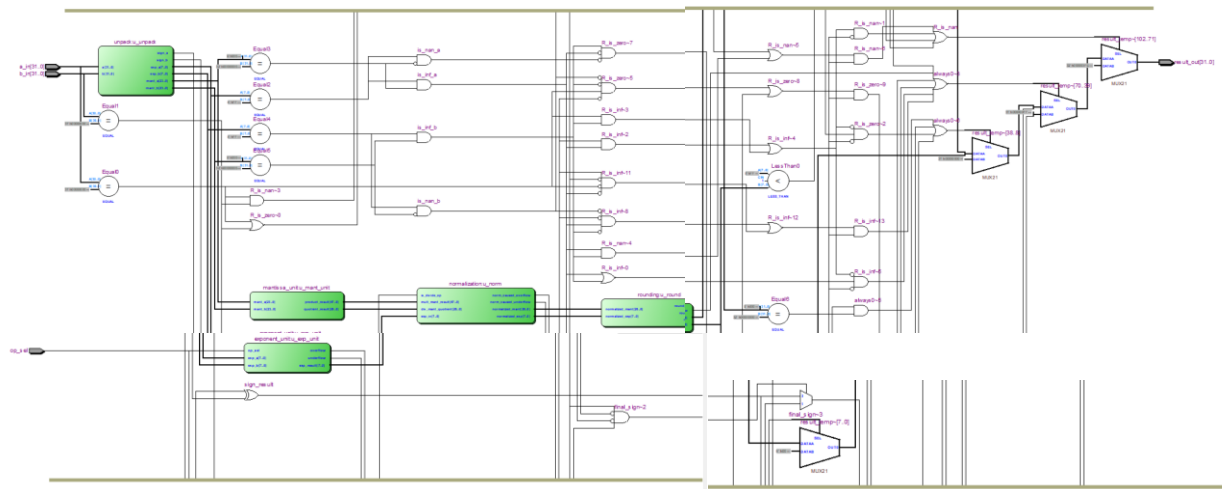
$display("Starting fp_arithmetic testbench...\n");

// Multiplication tests
test_case(32'h3f800000, 32'h3fc00000, 0, 32'h3fc00000, "1.0 * 1.5");
test_case(32'h40000000, 32'h3f000000, 0, 32'h3f800000, "2.0 * 0.5");
test_case(32'hc0400000, 32'h40800000, 0, 32'hc1400000, "-3.0 * 4.0");
test_case(32'h00000000, 32'h40a00000, 0, 32'h00000000, "0.0 * 5.0");
test_case(32'h7f800000, 32'h3f800000, 0, 32'h7f800000, "+Inf * 1.0");
test_case(32'h7f800000, 32'h00000000, 0, 32'h7fc00000, "+Inf * 0.0 = NaN");
test_case(32'h7fc12345, 32'h3f800000, 0, 32'h7fc00000, "NaN * 1.0");





// Division tests
test_case(32'h3f800000, 32'h40000000, 1, 32'h3f000000, "1.0 / 2.0 = 0.5");
test_case(32'h40400000, 32'h40000000, 1, 32'h3fc00000, "3.0 / 2.0 = 1.5");
test_case(32'hc0a00000, 32'h40800000, 1, 32'hbfa00000, "-5.0 / 4.0 = -1.25");
test_case(32'h00000000, 32'h3f800000, 1, 32'h00000000, "0.0 / 1.0 = 0.0");
test_case(32'h3f800000, 32'h00000000, 1, 32'h7fc00000, "1.0 / 0.0 = NaN");
test_case(32'h7f800000, 32'h3f800000, 1, 32'h7f800000, "Inf / 1.0 = Inf");
test_case(32'h7f800000, 32'h7f800000, 1, 32'h7fc00000, "Inf / Inf = NaN");

$display("All tests finished.");
$finish;
end
endmodule

```



Hình 17: RTL view của module fp\_arithmetic

|   |                        | Msgs     |          |          |          |          |          |          |  |          |  |  |
|---|------------------------|----------|----------|----------|----------|----------|----------|----------|--|----------|--|--|
| +  | /fp_arithmetic_tb/a_in | 3f800000 | 3f800000 | 40000000 | c0400000 | 00000000 | 7f800000 |          |  | 7fc12345 |  |  |
| +  | /fp_arithmetic_tb/b_in | 3fc00000 | 3fc00000 | 3f000000 | 40800000 | 40a00000 | 3f800000 | 00000000 |  | 3f800000 |  |  |
| +  | /fp_arithmetic_tb/r... | 3fc00000 | 3f800000 | c1400000 | 00000000 | 7f800000 |          | 7fc00000 |  |          |  |  |
|    | /fp_arithmetic_tb/o... | 0        |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |
|   |                        |          |          |          |          |          |          |          |  |          |  |  |

```

# 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

```

Hình 19: Terminal của module *fp\_arithmetic*

## 1. Mô tả các Modules

### *fp\_arithmetic*

- **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.
- **Các đầu vào/ra:**
    - `a_in`, `b_in`: Số IEEE-754 32-bit đầu vào.
    - `op_sel`: 0 = multiply, 1 = divide.
    - `result_out`: Kết quả IEEE-754 32-bit.
- 

#### **unpack**

- Tách:
    - `sign`: bit 31
    - `exponent`: bits 30-23
    - `mantissa`: bits 22-0, thêm ẩn 1 nếu là số chuẩn.
- 

#### **exponent\_unit**

- **Tính toán số mũ kết quả:**
    - Nhân:  $\text{exp\_a} + \text{exp\_b} - \text{bias}$  (127)
    - Chia:  $\text{exp\_a} - \text{exp\_b} + \text{bias}$
  - **Báo cáo:**
    - overflow: nếu số mũ  $> 255$
    - underflow: nếu số mũ  $< 0$
- 

#### **mantissa\_unit**

- **Nhân hoặc chia phần trị**
    - $\text{mant\_a} * \text{mant\_b}$  (48-bit)
    - $(\text{mant\_a} \ll 23) / \text{mant\_b}$  (27-bit): Dịch trái để giữ độ chính xác trước khi chia.
- 

#### **normalization**

- **Chuẩn hóa phần trị:**
  - Đưa dạng về 1.F nếu cần dịch bit.
  - Cập nhật số mũ tương ứng.



- Gắn cờ nếu kết quả dẫn đến overflow hoặc underflow.

## rounding

- **Làm tròn phần trị**
  - Sử dụng **Guard, Round, Sticky bits** (GRS)
  - Quy tắc: Round to Nearest, Ties to Even
  - Kiểm tra overflow do làm tròn (ví dụ, 1.11111...1 làm tròn thành 10.0 -> tăng số mũ)

## 2. Giải thích Testbench: fp\_arithmetic\_tb.v

### Cấu trúc:

- Khởi tạo tín hiệu đầu vào a\_in, b\_in, op\_sel
- Module fp\_arithmetic được instantiatd.
- Sử dụng task test\_case để:
  - Đưa đầu vào
  - Chờ xử lý
  - So sánh kết quả với giá trị mong đợi
  - In ra kết quả (PASS/FAIL)

### Test      Kết quả mong đợi

1.0 \* 1.5    1.5 → 0x3fc00000

2.0 \* 0.5    1.0 → 0x3f800000

-3.0 \* 4.0   -12.0 → 0xc1400000

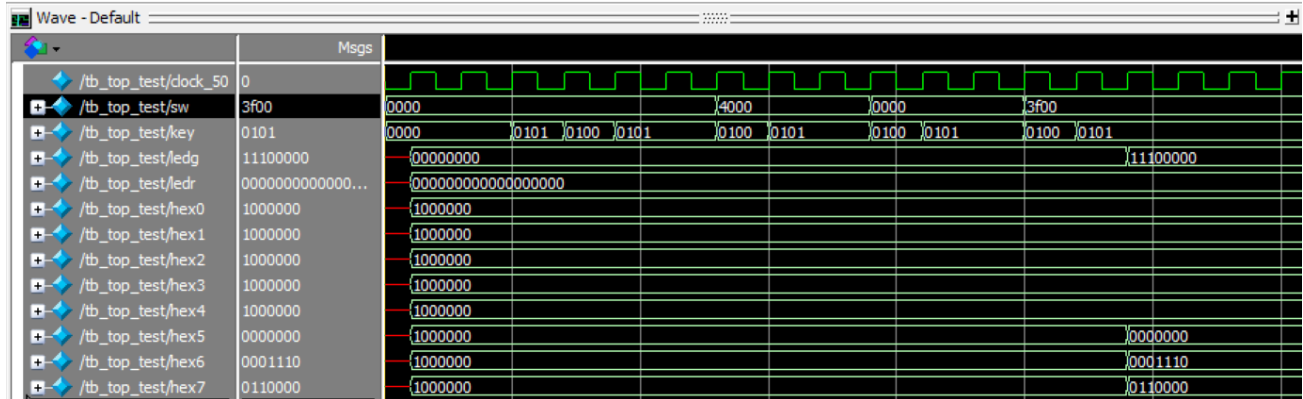
+Inf \* 0.0   NaN → 0x7fc00000

3.0 / 2.0    1.5 → 0x3fc00000

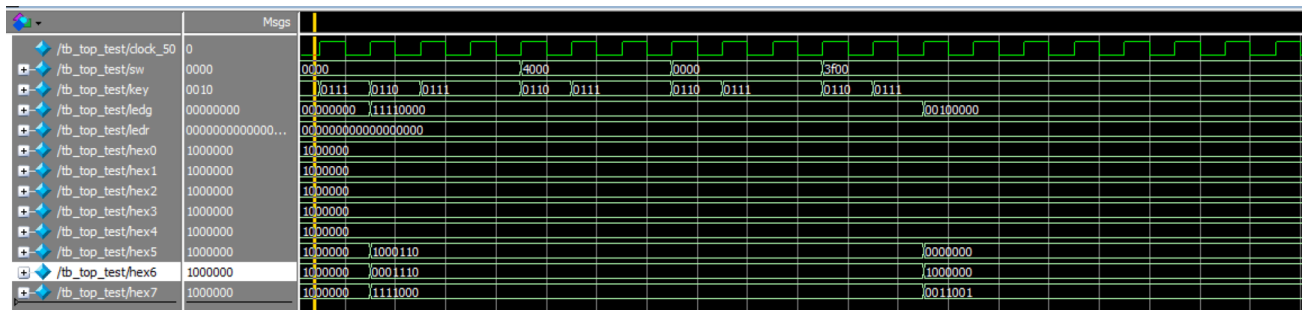
1.0 / 0.0    +Inf or NaN

### 3. testbench trên kit ( input 16 bit high and low cho 32 bit input)

Vd: A= 0x40000000; B= 0x3f000000; phép mult; đáp án sẽ là 0x3f800000 ; hiển thị led đúng kết quả.



Vd: A= 0x40000000; B= 0x3f000000; phép div; đáp án sẽ là 0x3f800000 ; hiển thị led đúng kết quả.





## **TÀI LIỆU THAM KHẢO**