ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

Khoa Điện – Điện tử Bộ môn Điện tử

---------\*\*\*--------



**Báo Cáo Bài Tập Lớn**

**KỸ THUẬT SỐ NÂNG CAO**

THIẾT KẾ MỘT MÁY TÍNH

DẤU CHẤM ĐỘNG

**GVHD: ThS. Trịnh Vũ Đăng Nguyên**

**Sinh viên thực hiện:**

**Ngô Minh Nhân MSSV: 1813327**

**Hồ Bá Phước MSSV: 1813638**

**TP.HCM, tháng 6 năm 2021**

Phụ lục

[ĐỀ TÀI 3](#_Toc31569)

[1. Máy tính dấu chấm động 4](#_Toc10338)

[2.Bộ cộng trừ: 5](#_Toc4503)

[a. Sơ đồ: 5](#_Toc17453)

[b. Giải thích các khối: 6](#_Toc25468)

[c. Testbench và hình ảnh: 7](#_Toc6985)

[i. Cộng 7](#_Toc16869)

[ii. Trừ: 8](#_Toc30330)

[3.Bộ nhân: 9](#_Toc26667)

[a. Sơ đồ: 9](#_Toc2743)

[b. Giải thích các khối: 9](#_Toc2667)

[c. Testbench và hình ảnh: 10](#_Toc10937)

[4.Bộ chia: 12](#_Toc11067)

[a. Giải thích các khối: 13](#_Toc1636)

[b. Testbench – Hình ảnh: 14](#_Toc7183)

[5.Bộ chia cách 2 17](#_Toc28169)

[a. Giải thuật bộ chia hai số Floating point 17](#_Toc26078)

[b. Trường hợp đặc biệt 18](#_Toc264)

[c. Testbench và hình ảnh: 19](#_Toc958)

[6.Bộ nhân cách 2 21](#_Toc10862)

[a. Thuật toán nhân 21](#_Toc26514)

[b. Trường hợp đặc biệt 23](#_Toc9080)

[c. Testbench và hình ảnh: 24](#_Toc16643)

[7.Bộ chuyển đổi từ số floating point sang số thực 26](#_Toc13084)

[a. Giải thuật 26](#_Toc340)

[b. Testbench và hình ảnh: 28](#_Toc23314)

ĐỀ TÀI

PHẦN I: THIẾT KẾ MỘT MÁY TÍNH DẤU CHẤM ĐỘNG THỰC HIỆN CÁCPHÉP TOÁN (+, -, \*, /) GIỮA HAI SỐ FLOATING POINT (IEEE-754, SINGLEPRECISION, 32-BIT). (40%) (bắt buộc)

• Ngõ vào: 2 số float 32-bit (A, B) và 2-bit lựa chọn các phép toán (+, -, \*, /)

• Ngõ ra: Kết quả phép toán ở định dạng IEEE-754

• Lưu ý: Không được sử dụng các phép toán có sẵn trong Verilog/VHDL (như+, -,\*, / và <<) chỉ được sử dụng các lệnh logic (AND, OR, XOR, NOT)

PHẦN II: BONUS

• Thực hiện giải thuật thứ 2 cho các phép tính cộng, trừ (+, -): + 5% tổng kết

• Thực hiện giải thuật thứ 2 cho phép tính nhân \* : + 5% tổng kết

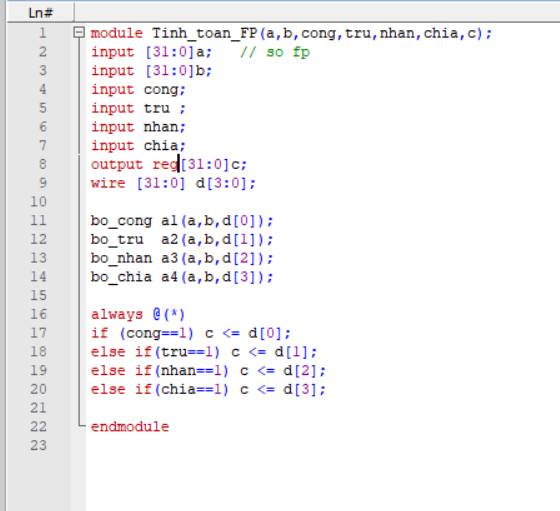
• Thực hiện giải thuật thứ 2 cho phép tính chia / : + 5% tổng kết

• Thực hiện phép lấy Căn bậc n: + 10% tổng kết

• Chuyển ngõ vào, ngõ ra thành dạng thập phân tương ứng (ex. 5.25\*10-5, -2.5\*103…):

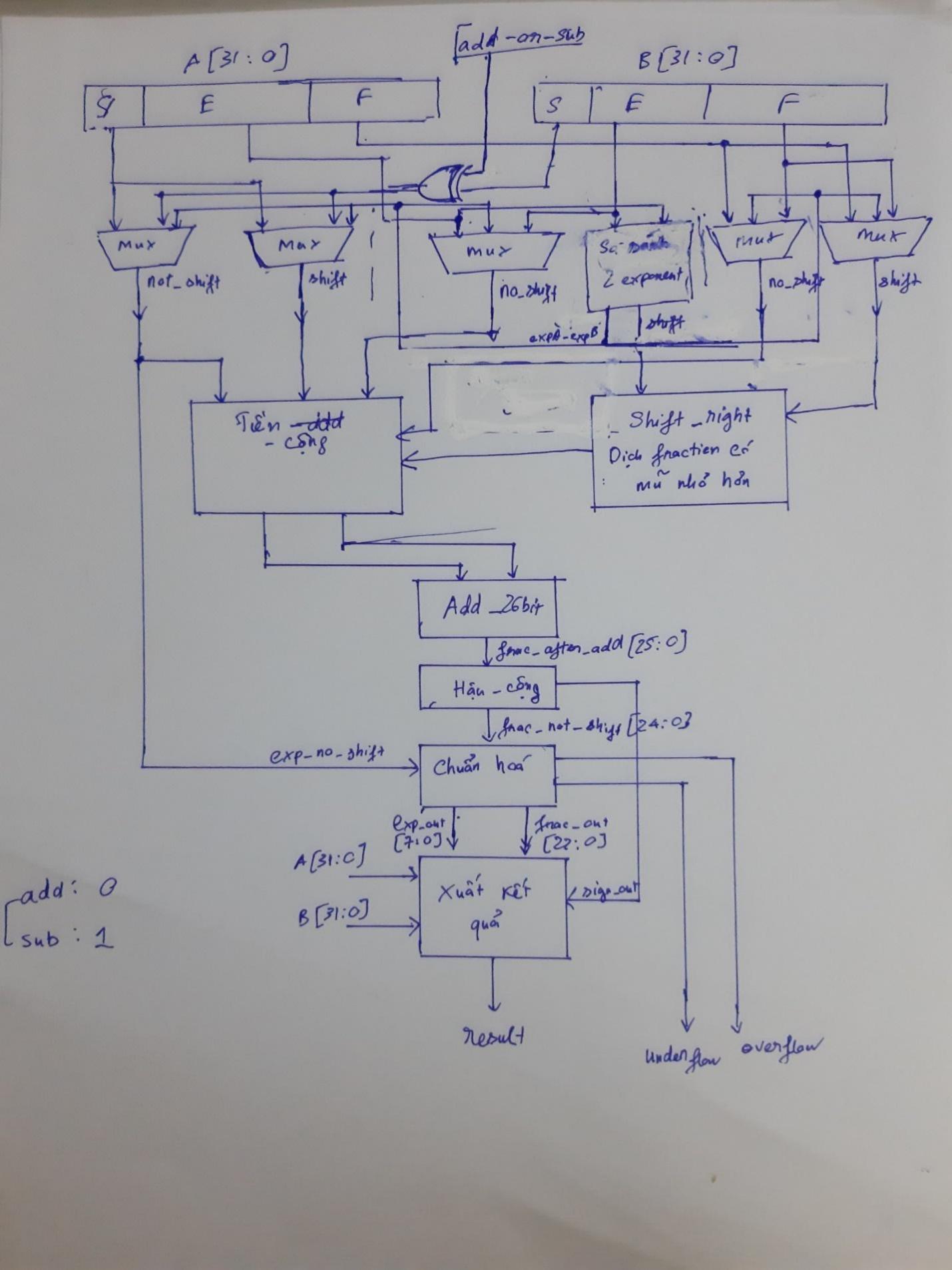
+ 10% tổng kết

1. Máy tính dấu chấm động



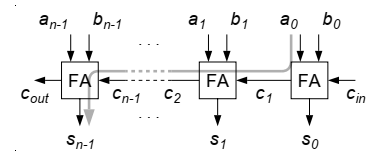
Module tính toán số chấm động

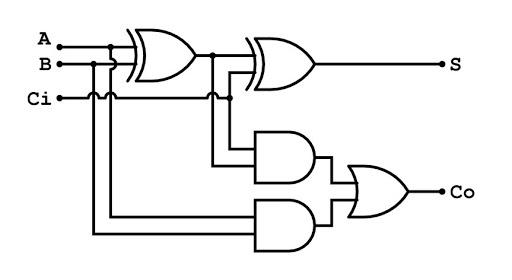
**2.Bộ cộng trừ:**

1. Sơ đồ: 
2. Giải thích các khối:

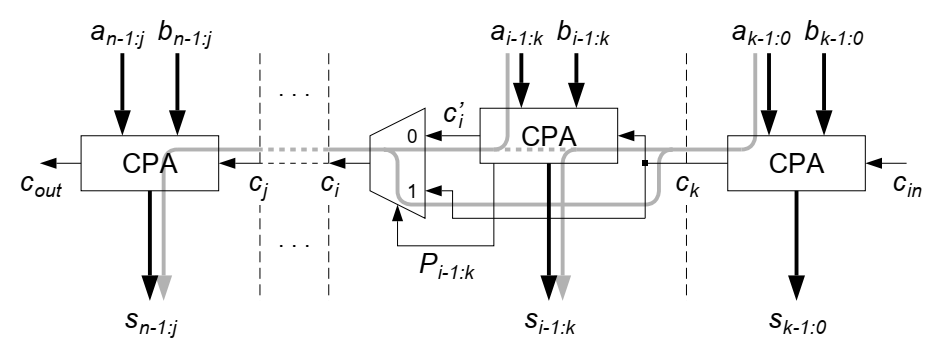
* Add\_or\_Sub: sử dụng XOR để xuất ra dấu để tính toán cộng (bit0) hoặc trừ(bit1)
* Mux 2-1: Tìm sign, fraction, exponent shift và không cần shift nhờ vào tín hiệu expA -expB ở bộ so sánh 2 exponent va tìm được giá trị shift
* Bộ so sánh: so sánh 2 số mũ của A và B bằng cách expA -expB và xuất giá trị hiệu của chúng, và tín hiệu (expA-expB) để biết số nào exponent lớn hơn
* Bộ dịch phải: để dịch phải fraction có số mũ nhỏ hơn
* Bộ tiền\_cộng: mở rộng 26bit đối với 2 fraction đã dịch phải và không dịch phải (24bit), kiểm tra nếu âm thì lấy bù 2 rồi mở rộng bit 1, dương thì mở rộng bit 0
* Bộ cộng: sử dụng 2 giải thuật: là Ripple Carry Adder (RCA) và Carry Skip Adder (CSKA)
  + Ripple Carry Adder (RCA):

Các Full – Adder nối tiếp nhau, Cin’ = Co



Với cấu trúc của Full \_Adder: 

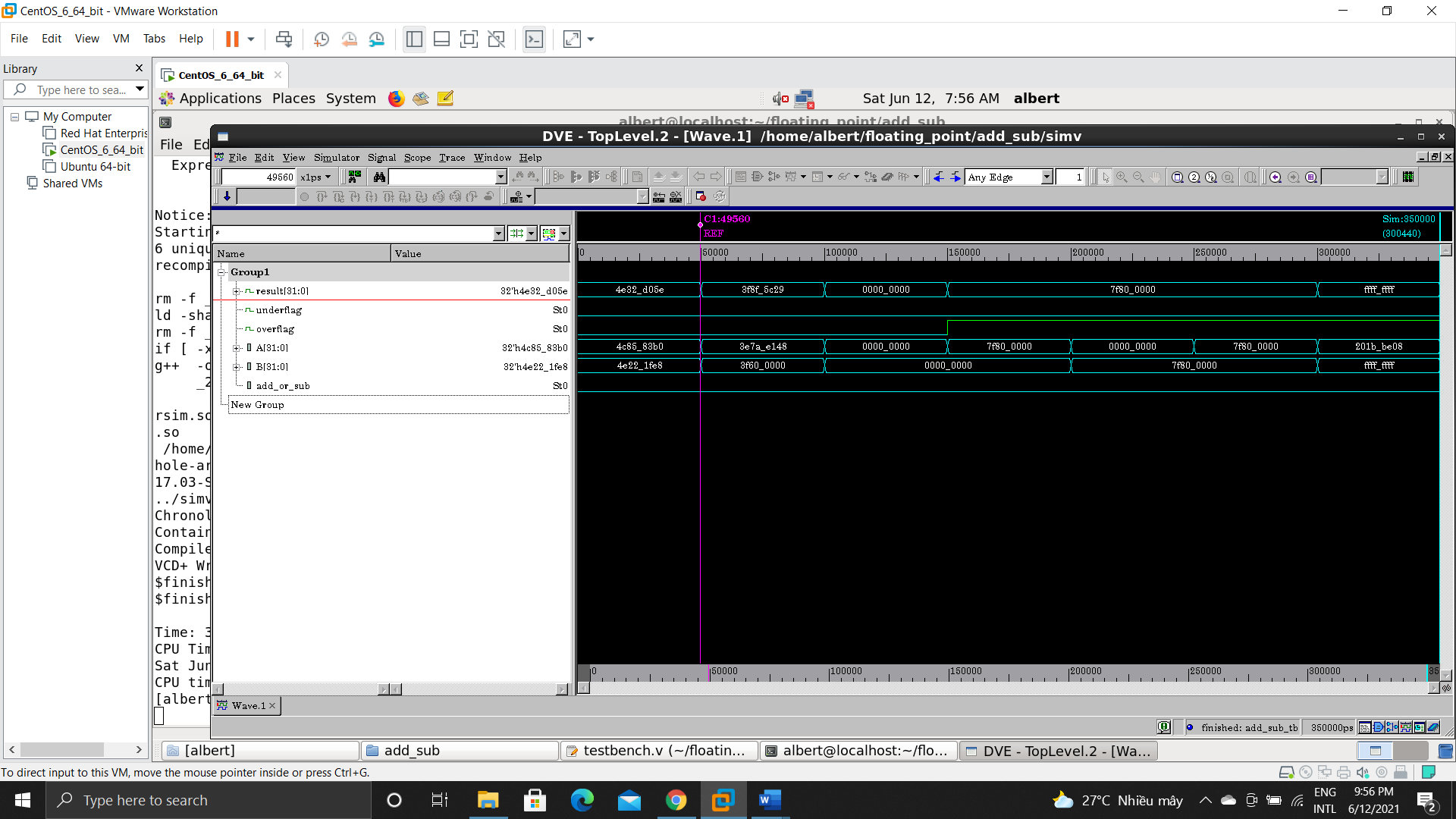
* + Carry Skip Adder (CSKA):



* Bộ chuẩn hoá: Chuẩn hoá sau khi cộng trừ 2 số để trở thành 1.0010 dạng chuẩn, sử dụng bộ tìm\_bit1 để tìm vị trí số 1 đầu tiên sau khi cộng, rồi sau đó dùng dịch trái để dịch số sang trái
* Bộ xuất kết quả: Xuất giá trị cuối cùng sau khi tính toán, bên trong khối có xét các trường hợp đặc biệt trước khi xuất ra các kết quả cuối cùng

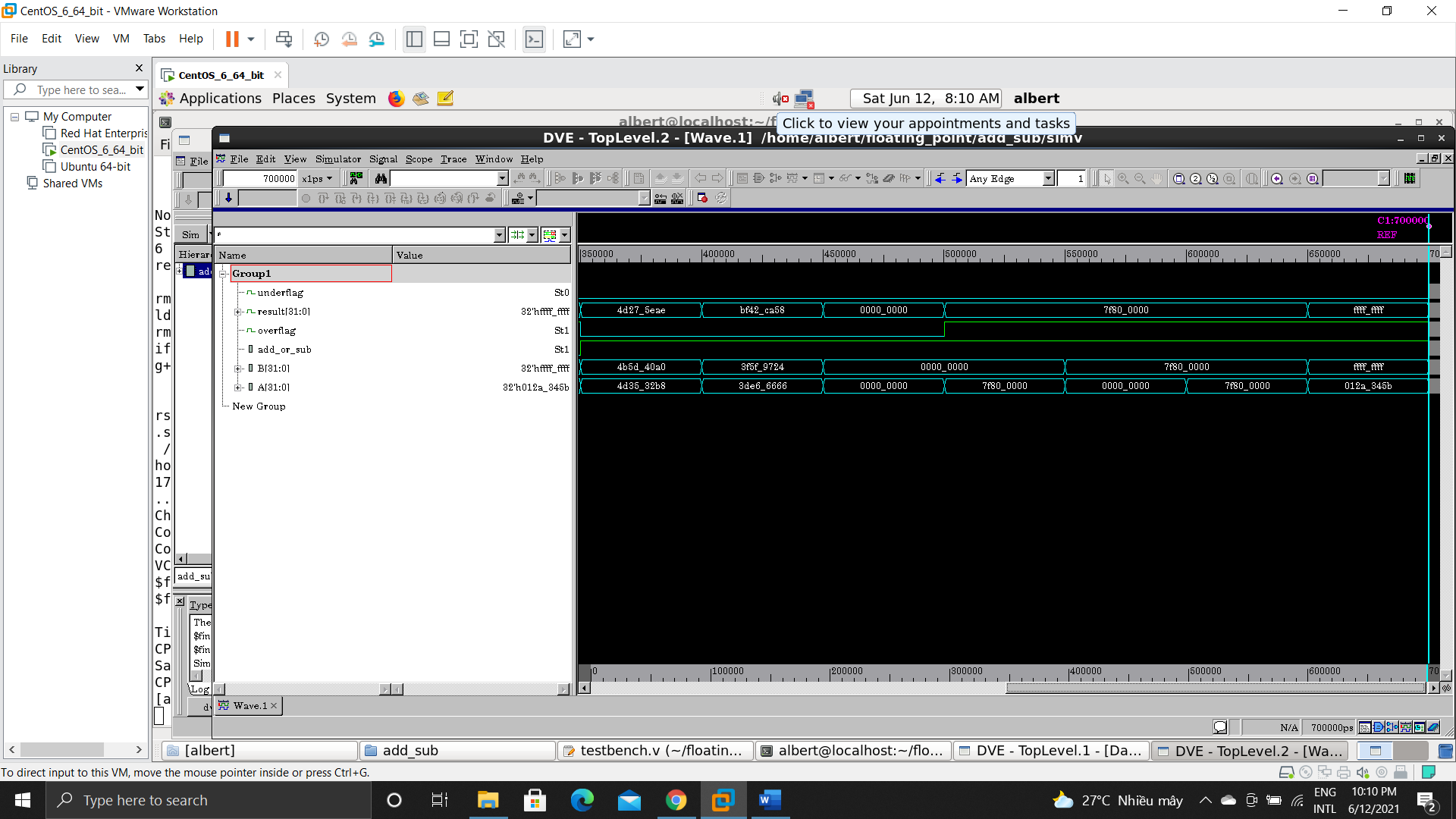
1. Testbench và hình ảnh:
2. Cộng

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | PHÉP TOÁN | IN1 | | IN2 | | KẾT QUẢ | | HÌNH |
| Decimal | FloatingPoint | Decimal | FloatingPoint | Decimal | FloatingPoint |
| 1 | Cộng | 7E7 | 0x4C8583B0 | 6.8E8 | 0x4E221FE8 | 0.75E9 | 0x4E32D05E | 1 |
| 2 | 0.245 | 0x3E7AE148 | 0.875 | 0x3F600000 | 1.12 | 0x3F8F5C29 |
| 3 | 0 | 0x00000000 | 0 | 0x00000000 | 0 | 0x00000000 |
| 4 | Inf | 0x7F800000 | 0 | 0x00000000 | Inf | 0x7F800000 |
| 5 | 0 | 0x00000000 | Inf | 0x7F800000 | Inf | 0x7F800000 |
| 6 | Inf | 0x7F800000 | Inf | 0x7F800000 | Inf | 0x7F800000 |
| 7 | 3.198 | 0x201BBE08 | NaN | 0x7FFFFFFF | NaN | 0x7FFFFFFF |

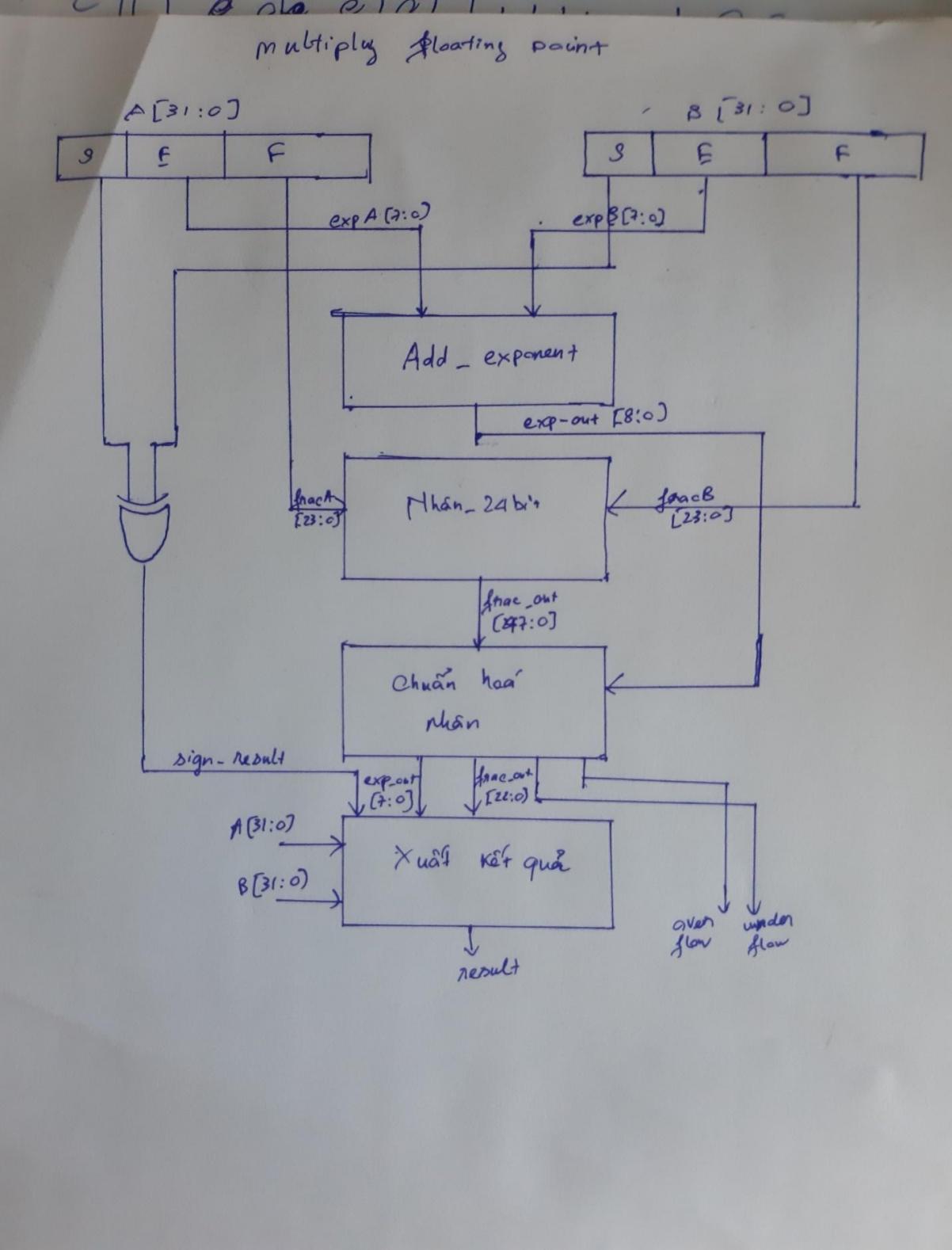
Hình ảnh:

1. Trừ:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | PHÉP TOÁN | IN1 | | IN2 | | KẾT QUẢ | | HÌNH |
| Decimal | FloatingPoint | Decimal | FloatingPoint | Decimal | FloatingPoint |
| 1 | Trừ | 19E7 | 0x4D3532B8 | 14.5E6 | 0x4B5D40A0 | 175.5E6 | 0x4D275EAE | 2 |
| 2 | 0.1125 | 0x3DE66666 | 0.8734 | 0x3F5F9724 | -0.7609 | 0xBF42CA58 |
| 3 | 0 | 0x00000000 | 0 | 0x00000000 | 0 | 0x00000000 |
| 4 | Inf | 0x7F800000 | 0 | 0x00000000 | Inf | 0x7F800000 |
| 5 | 0 | 0x00000000 | Inf | 0x7F800000 | Inf | 0x7F800000 |
| 6 | Inf | 0x7F800000 | Inf | 0x7F800000 | Inf | 0x7F800000 |
| 7 | 3.198 | 0x12A345B | NaN | 0x7FFFFFFF | NaN | 0x7FFFFFFF |

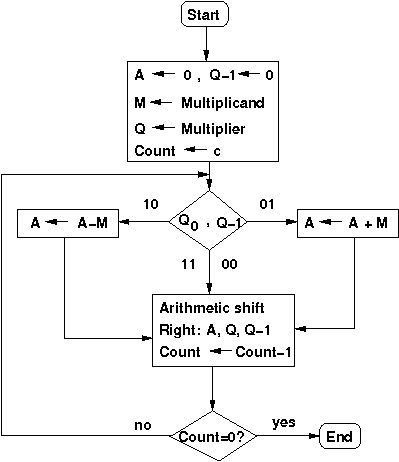
Hình ảnh

**3.Bộ nhân:**

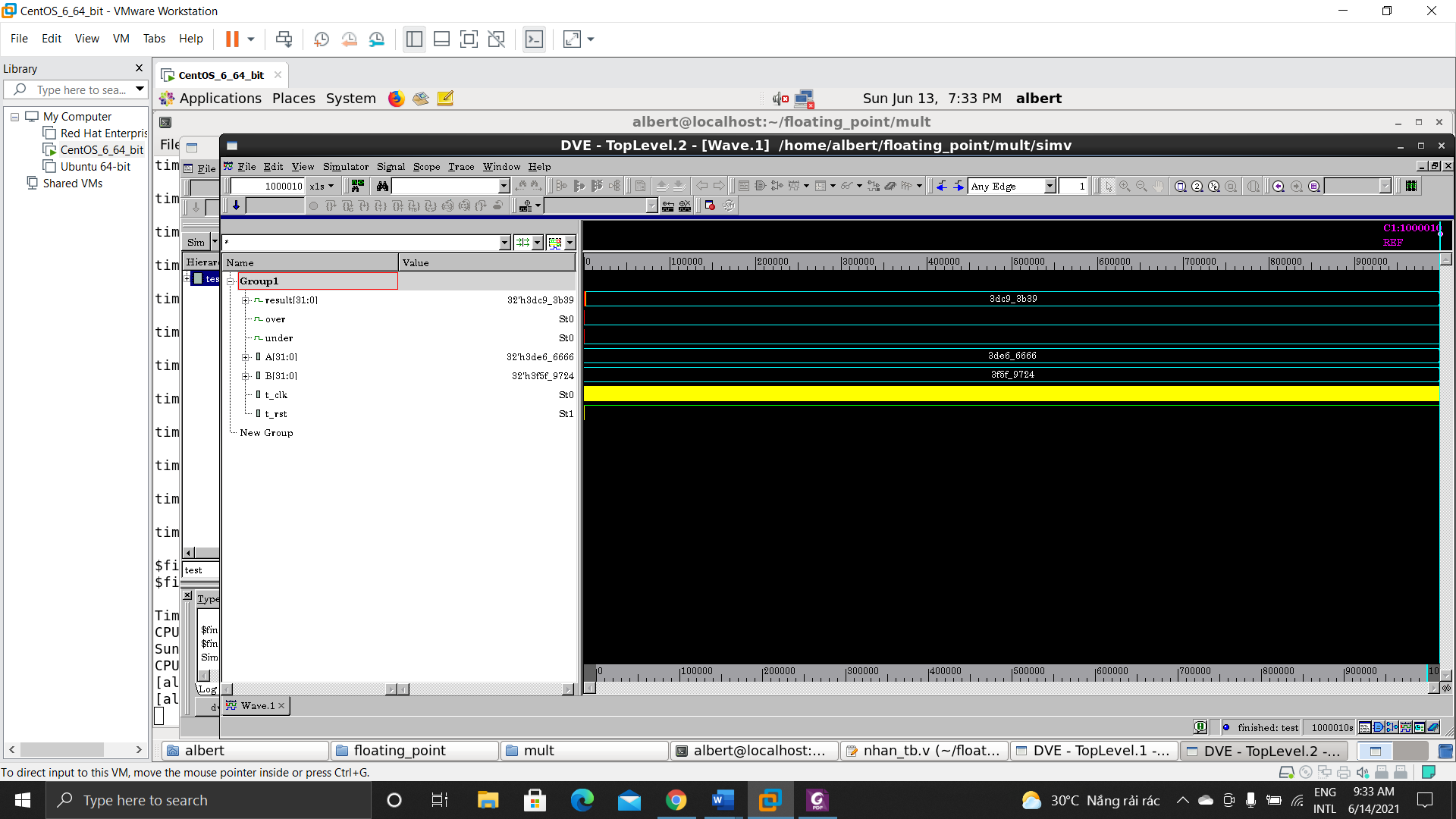
1. Sơ đồ:
2. Giải thích các khối:

* Khối cộng\_exponent: Khi nhân 2 số thì ta chỉ cần cộng 2 số mũ với nhau, ngõ ra là exp\_out 9bit với bit9 dùng để kiểm tra tràn, để đưa vào bộ chuẩn hoá kiểm tra tràn dưới (underflow)
* Bộ Chuẩn hoá: Chuẩn hoá kết quả sau khi tính toán xong, về số chuẩn hoá là 1.00010
* Bộ Nhân\_24bit: Nhân 2 fraction với nhau, sử dụng giải thuật phép nhân:
  + Giải thuật Booth:

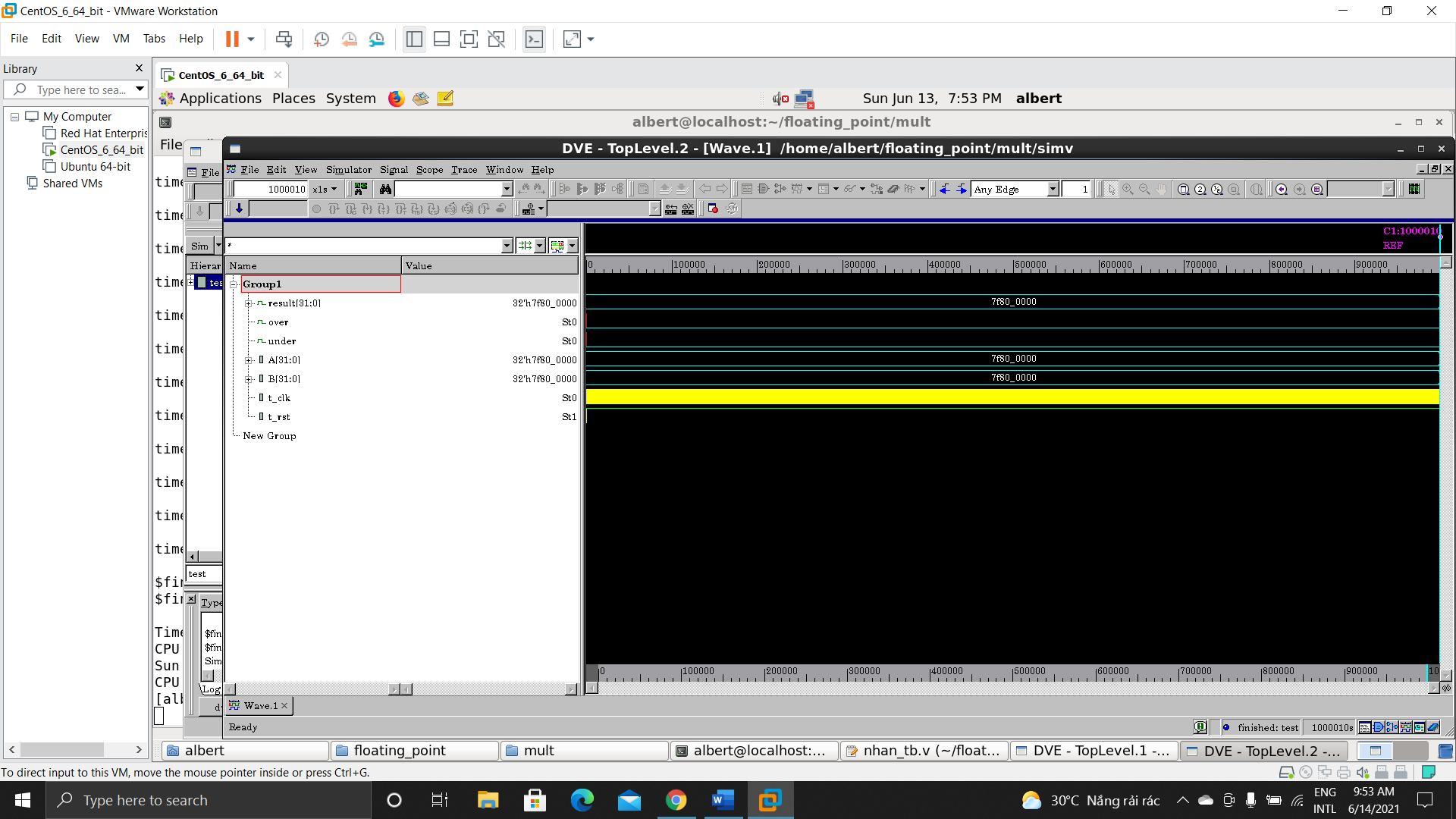
Sơ đồ giải thuật:

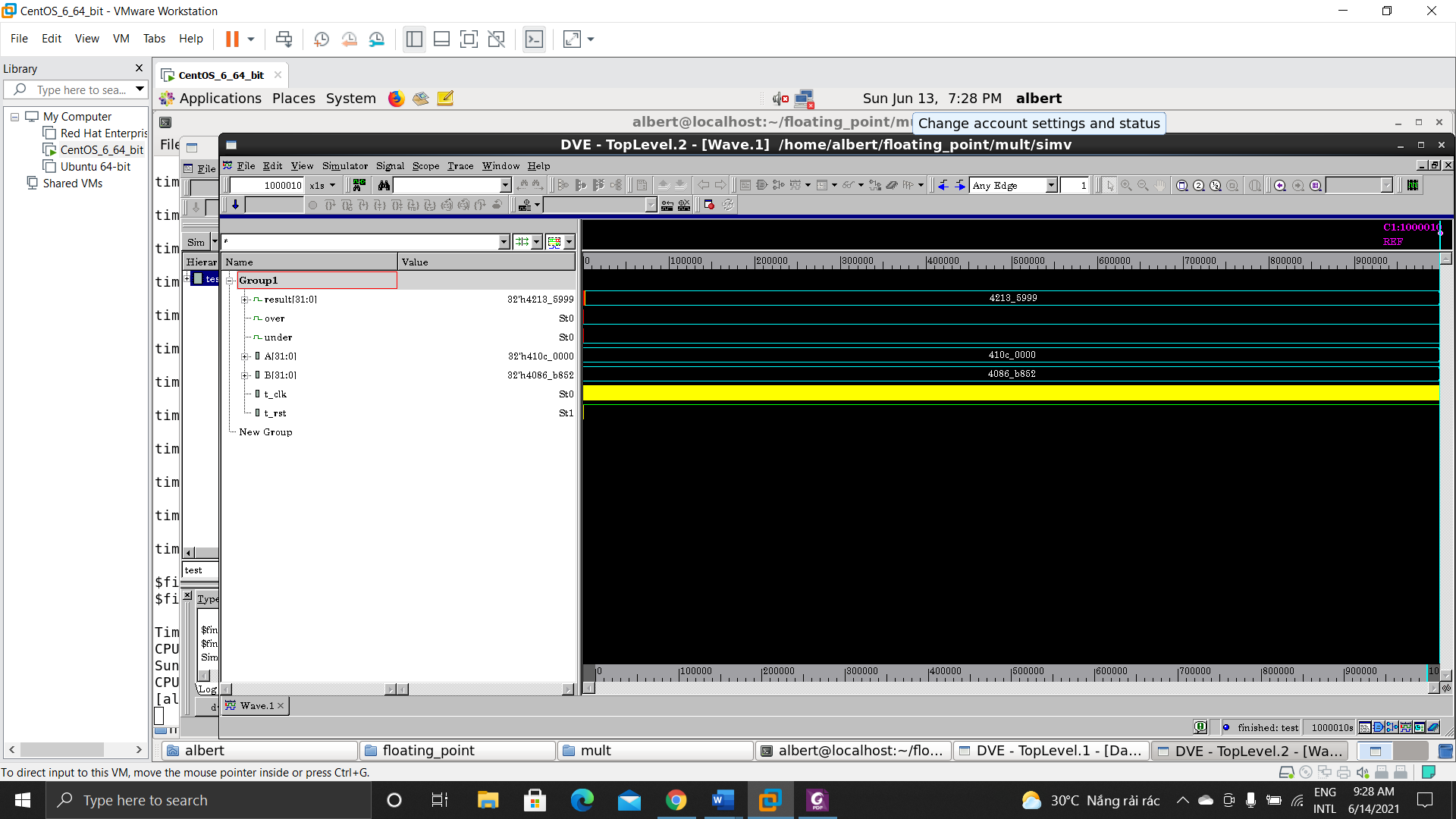
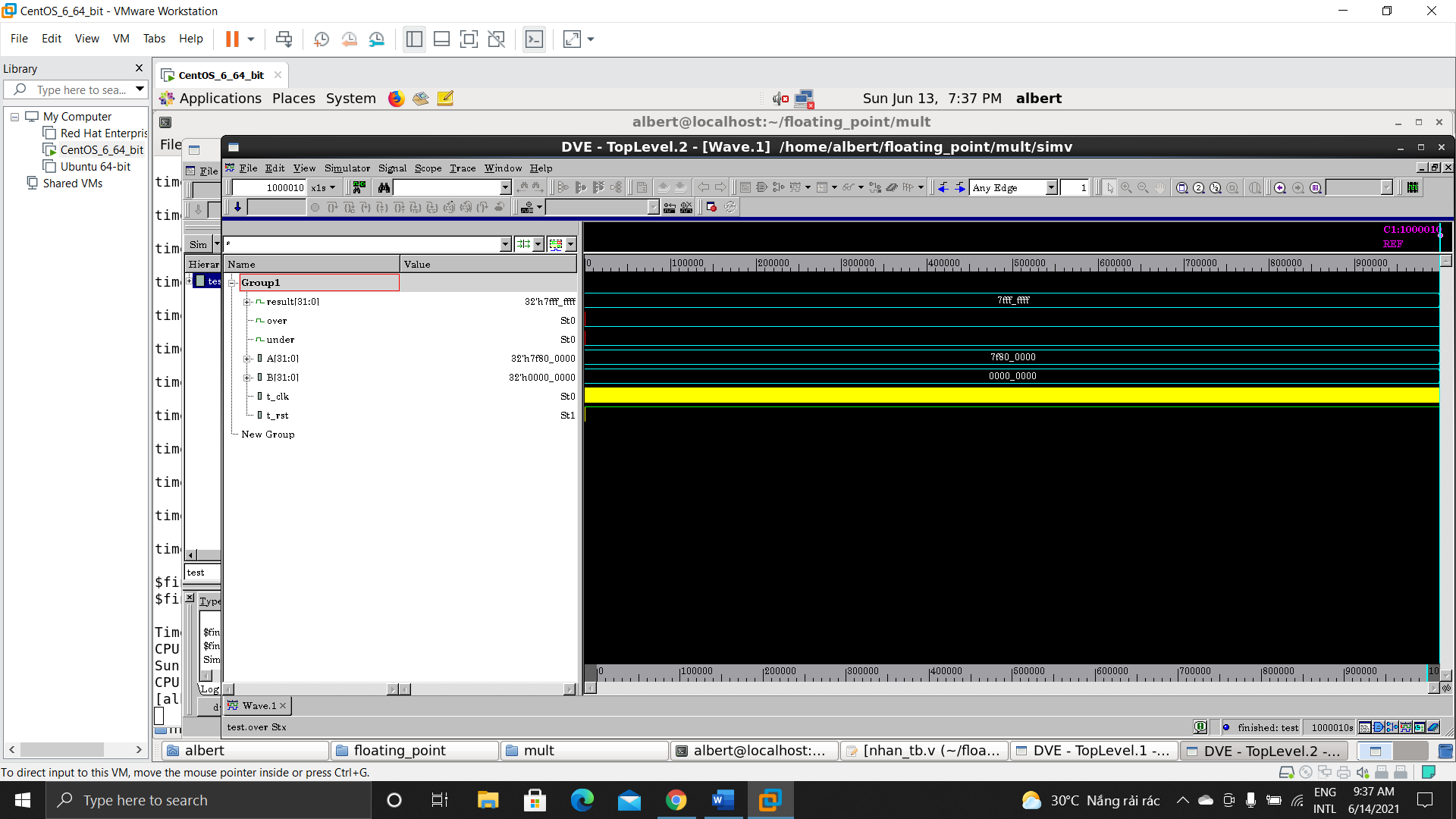
1. Testbench và hình ảnh:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | PHÉP TOÁN | IN1 | | IN2 | | KẾT QUẢ | | HÌNH |
| Decimal | FP | Decimal | FP | Decimal | FP |
| 1 | Nhân | 8.75 | 0x410C0000 | 4.21 | 0x4086B852 | 36.8375 | 0x42135999 | a |
| 2 | 0.1125 | 0x3DE66666 | 0.8734 | 0x3F5F9724 | 0.0982575 | 0x3DC93B39 | b |
| 3 | 0 | 0x00000000 | 0 | 0x00000000 | 0 | 0x00000000 | c |
| 4 | Inf | 0x7F800000 | 0 | 0x00000000 | Inf | 0x7F800000 | d |
| 5 | 0 | 0x00000000 | Inf | 0x7F800000 | Inf | 0x7F800000 | e |
| 6 | Inf | 0x7F800000 | Inf | 0x7F800000 | Inf | 0x7F800000 | f |
| 7 | 3.198 | 0x12A3456B | NaN | 0x7FFFFFFF | NaN | 0x7FFFFFFF | g |

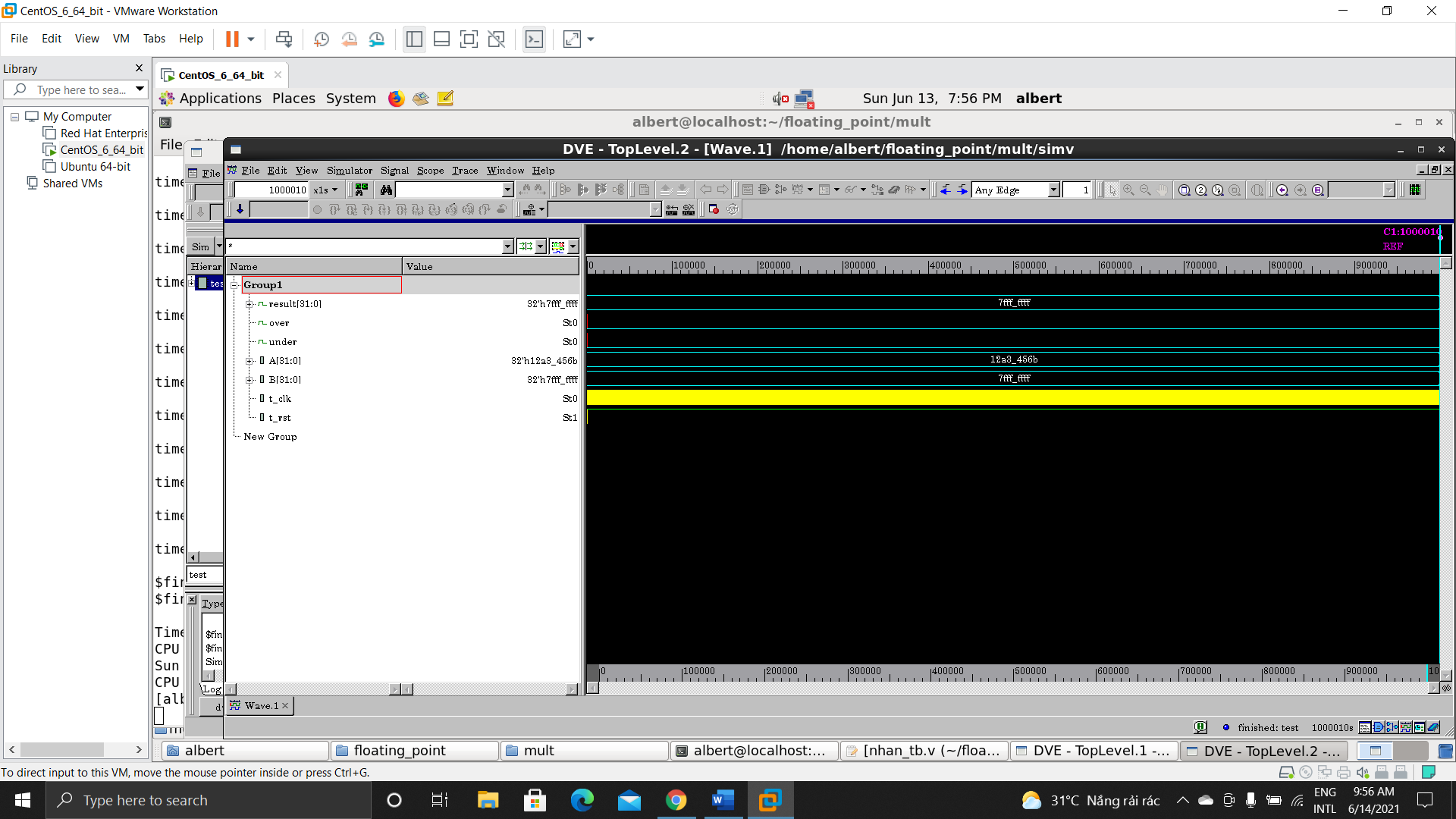
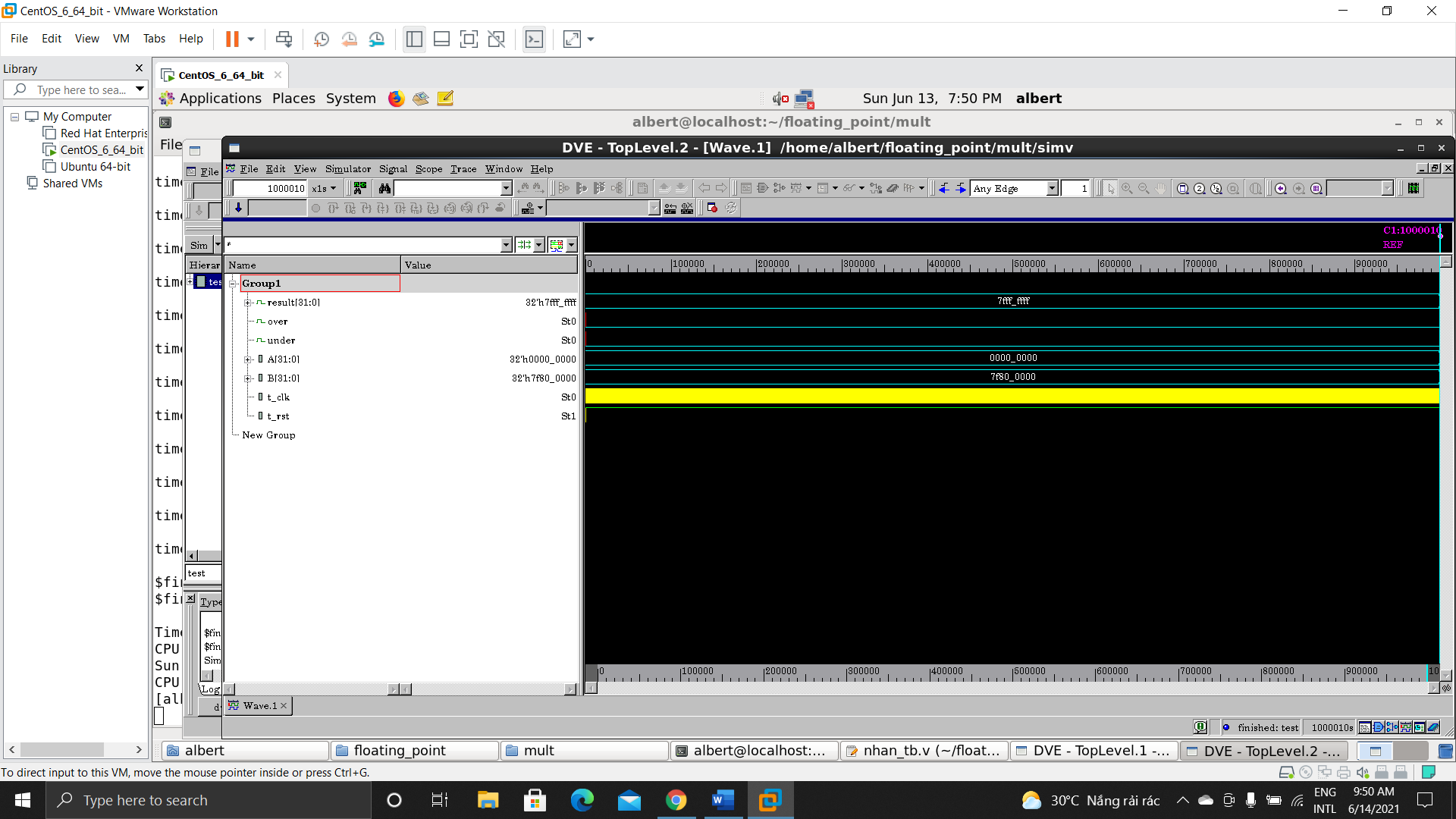
Hình ảnh:

a.

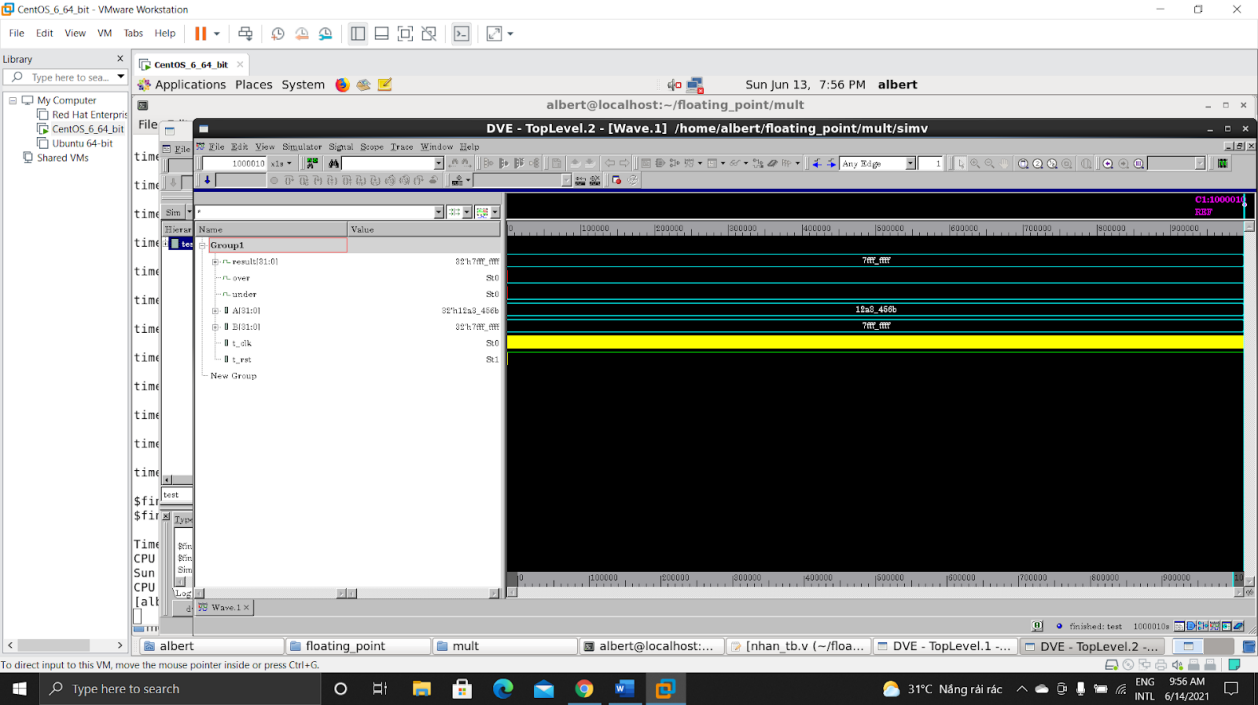
b.

c.

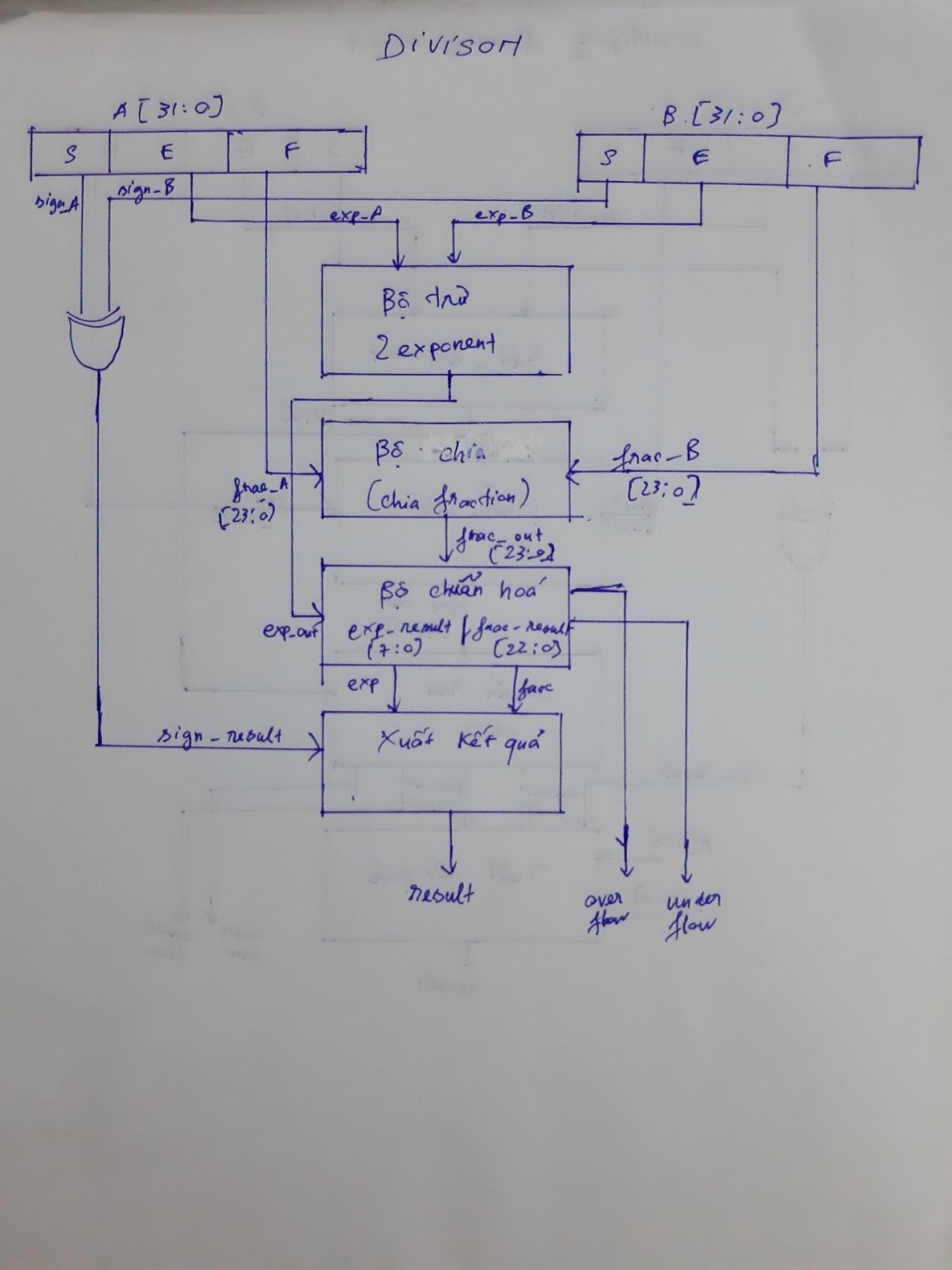
d.

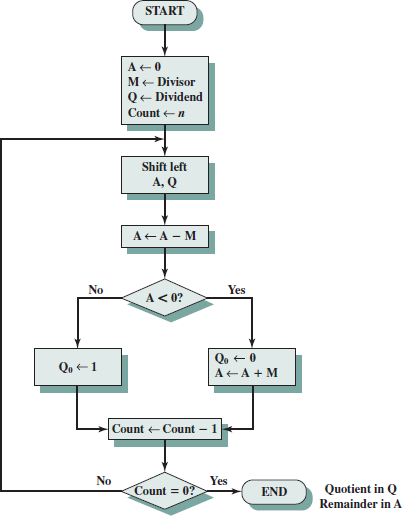
e.

f.

g. 

**4.Bộ chia:**





1. Giải thích các khối:

Bộ trừ: để tính toán số mũ bằng cách expA – expB, tìm được exp\_out.

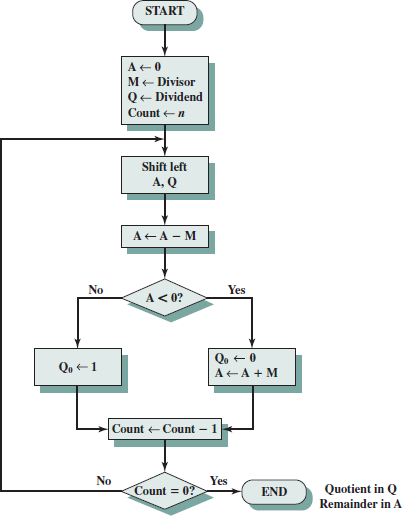
Bộ chia: dung để tính toán fraction, thực hiên bằng giải thuật

Bộ chuẩn hoá: Chuẩn hoá số sau khi tính toán fraction và exponent, đồng thời xuất cờ báo tràn trên – tràn dưới

Bộ xuất kết quả: xuất ra kết quả cuối cùng

Giải thuật chia: Sử dụng restoring (đơn giản hơn non-restoring)

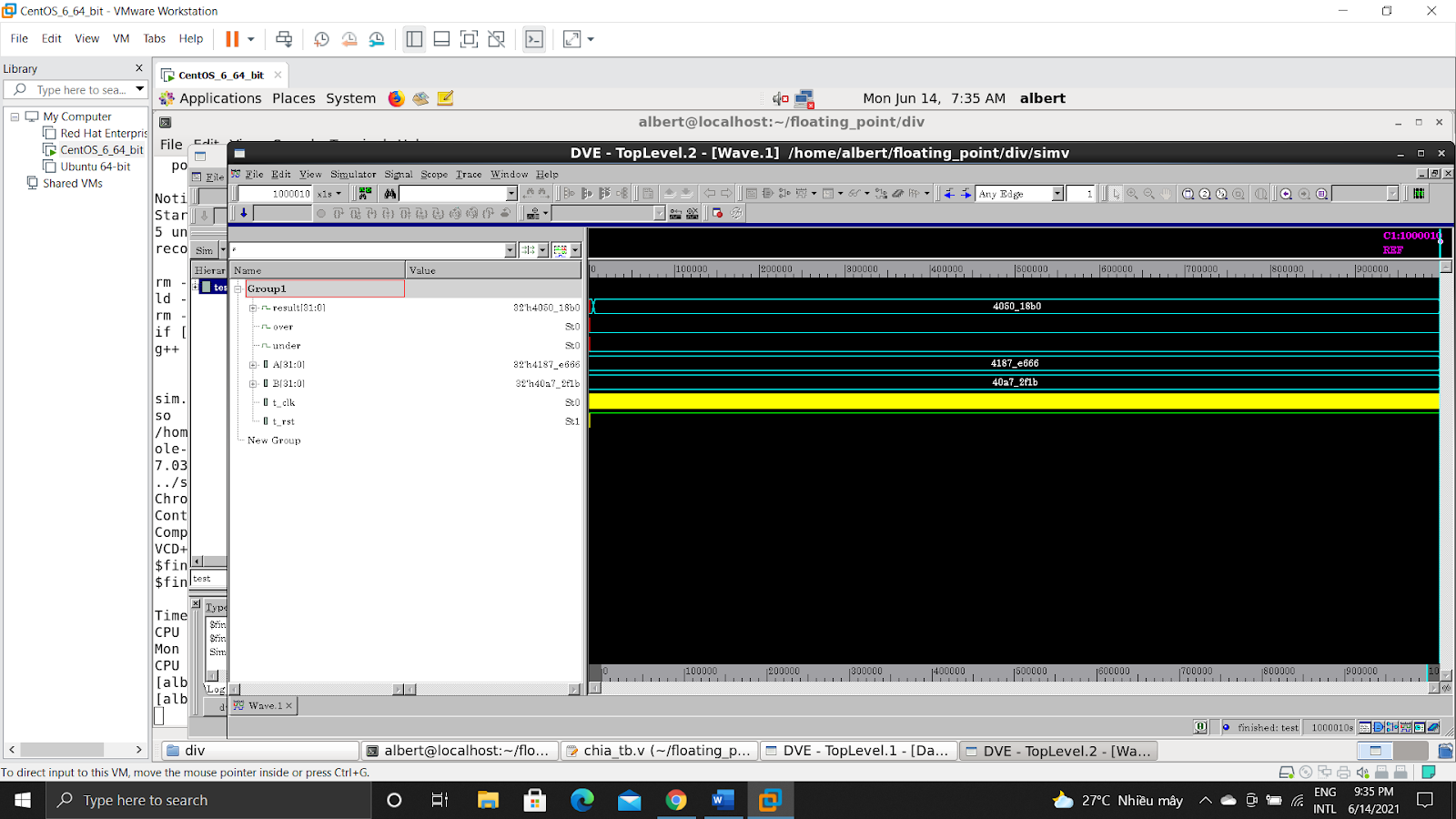
Sơ đồ giải thuật:



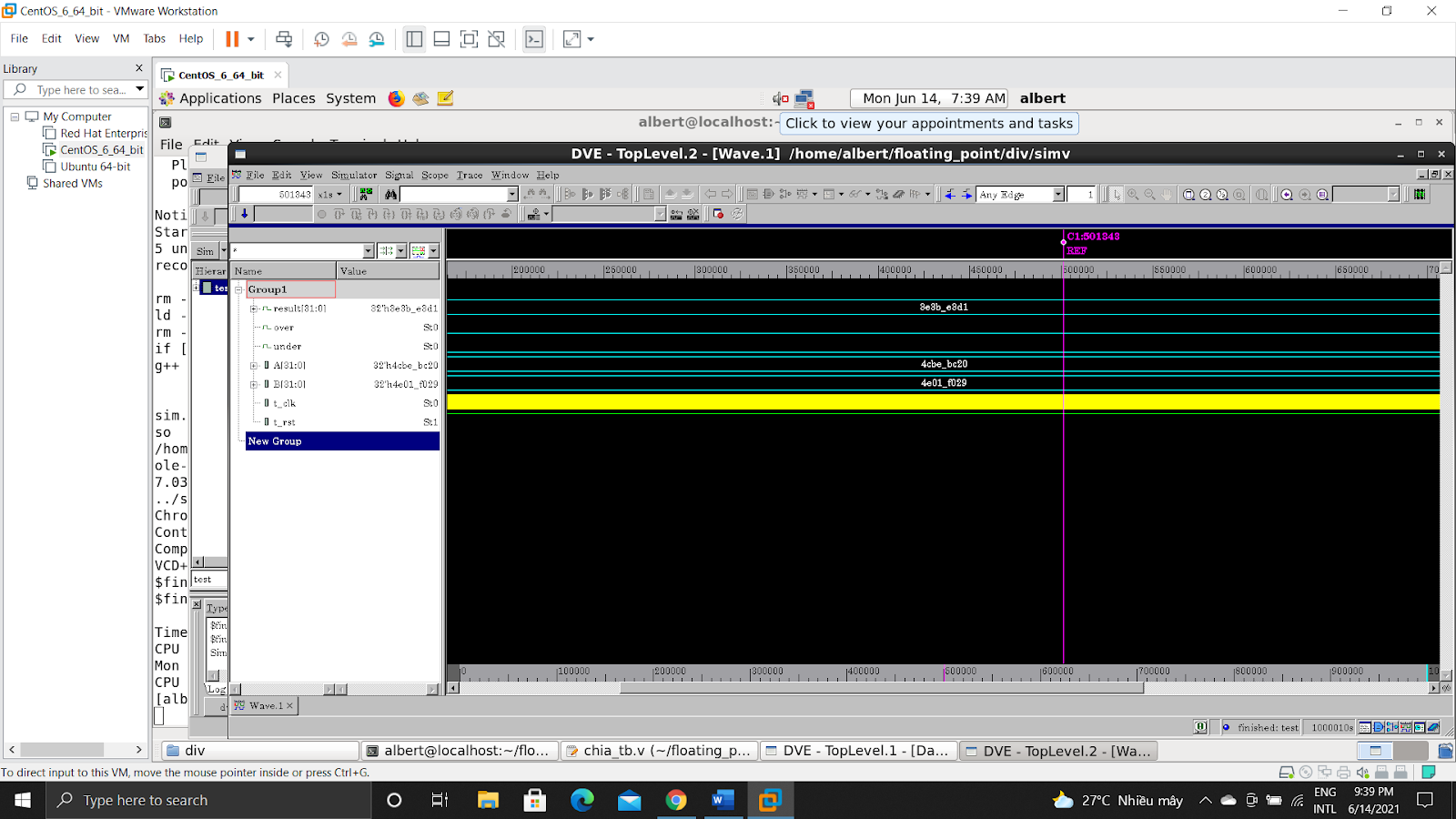
1. Testbench – Hình ảnh:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | PHÉP TOÁN | IN1 | | IN2 | | KẾT QUẢ | | HÌNH |
| Decimal | FP | Decimal | FP | Decimal | FP |
| 1 | Chia | 16.9875 | 0x4187E666 | 5.2245 | 0x40A72F1B | 3.2515 | 0x405018B0 | a |
| 2 | 10E7 | 0x4CBEBC20 | 5.45E8 | 0x4E01F029 | 0.18348 | 0x3E3BE3D1 | b |
| 3 | 0 | 0x00000000 | 0 | 0x00000000 | NaN | 0x7FFFFFFF | c |
| 4 | Inf | 0x7F800000 | 0 | 0x00000000 | Inf | 0x7F800000 | d |
| 5 | 0 | 0x00000000 | Inf | 0x7F800000 | Inf | 0x00000000 | e |
| 6 | Inf | 0x7F800000 | Inf | 0x7F800000 | NaN | 0x7FFFFFFF | f |
| 7 | 98.7654 | 0x42C587E3 | NaN | 0x7FFFFFFF | NaN | 0x7FFFFFFF | g |

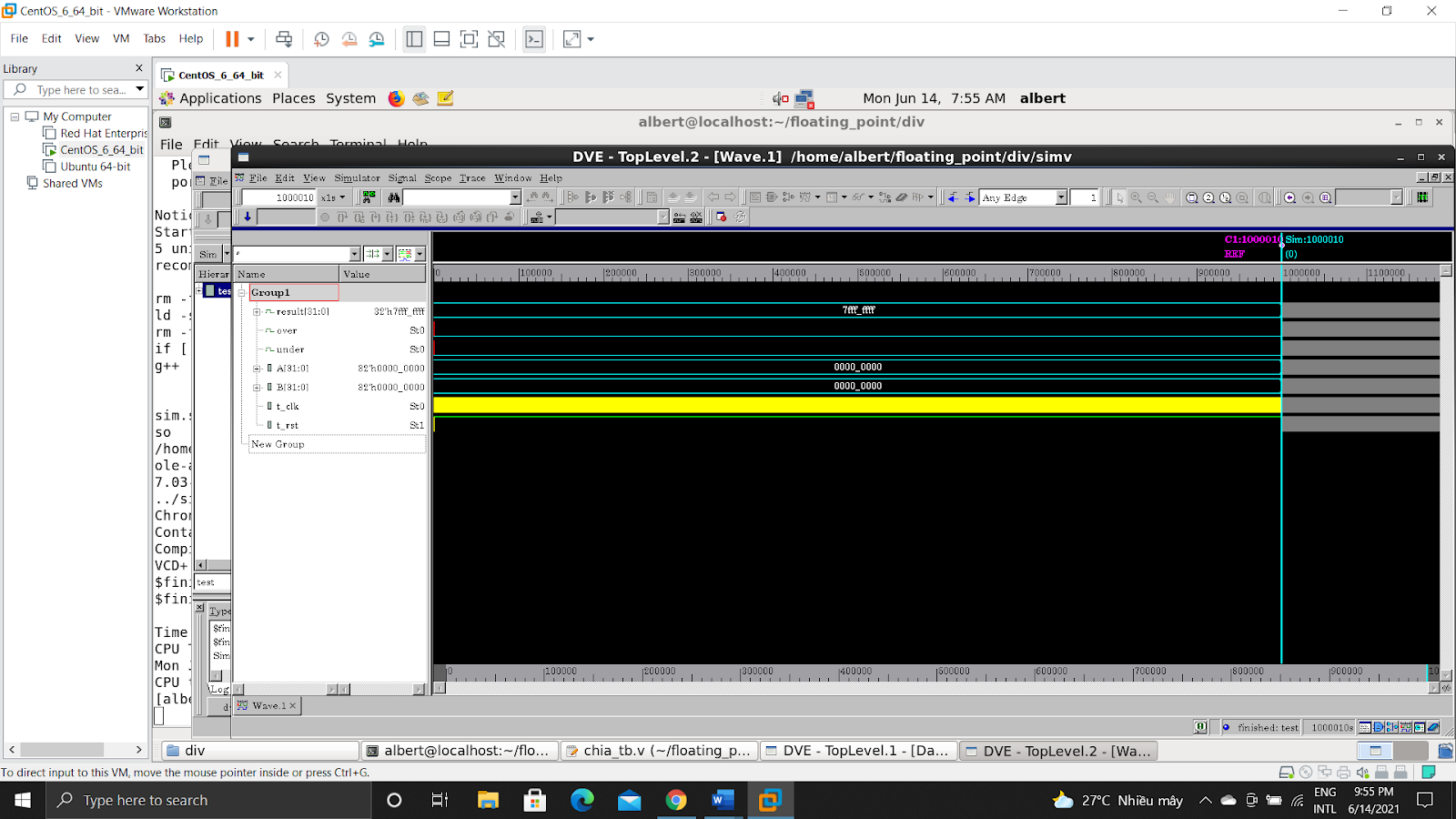
a.



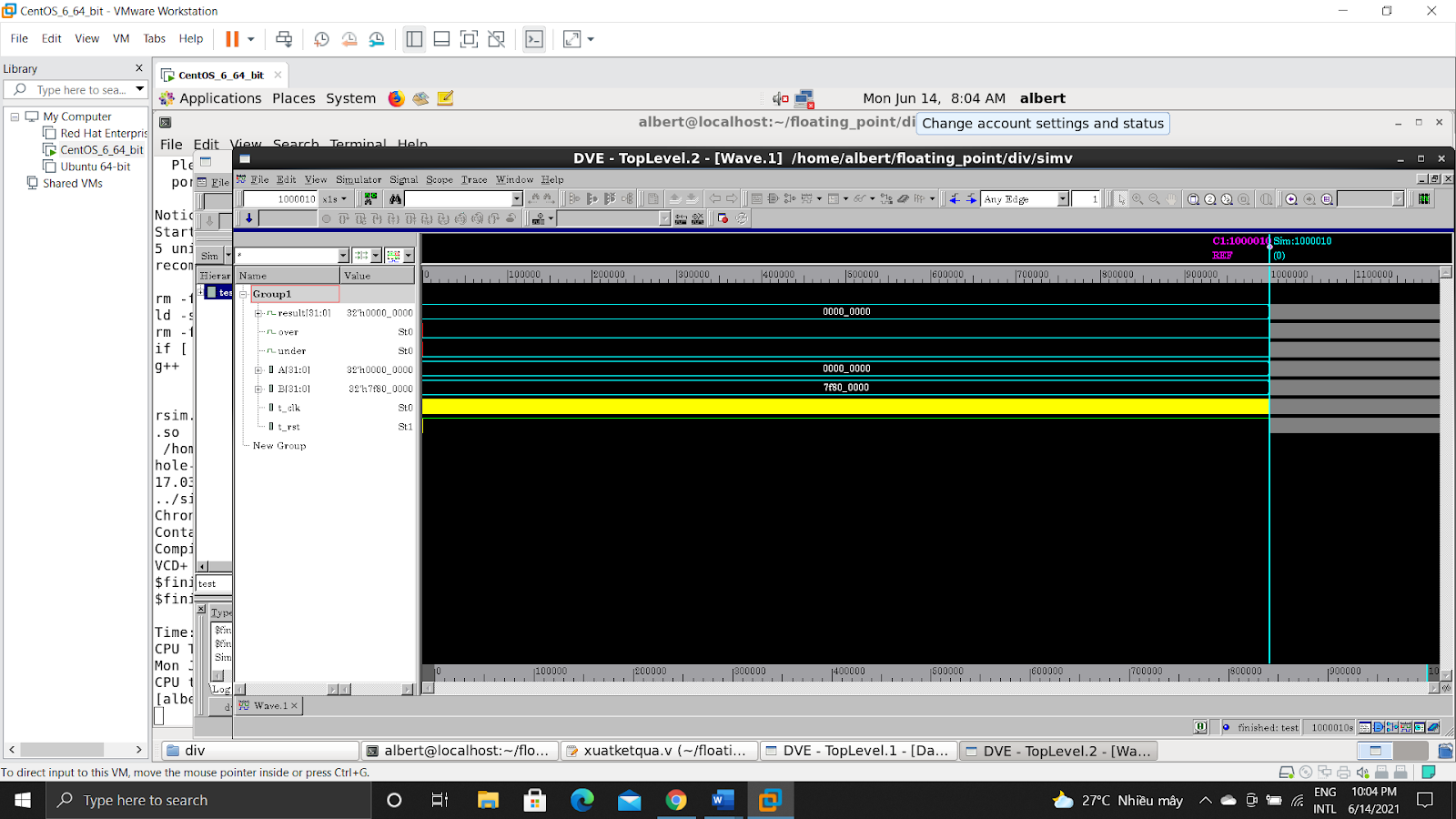
b.



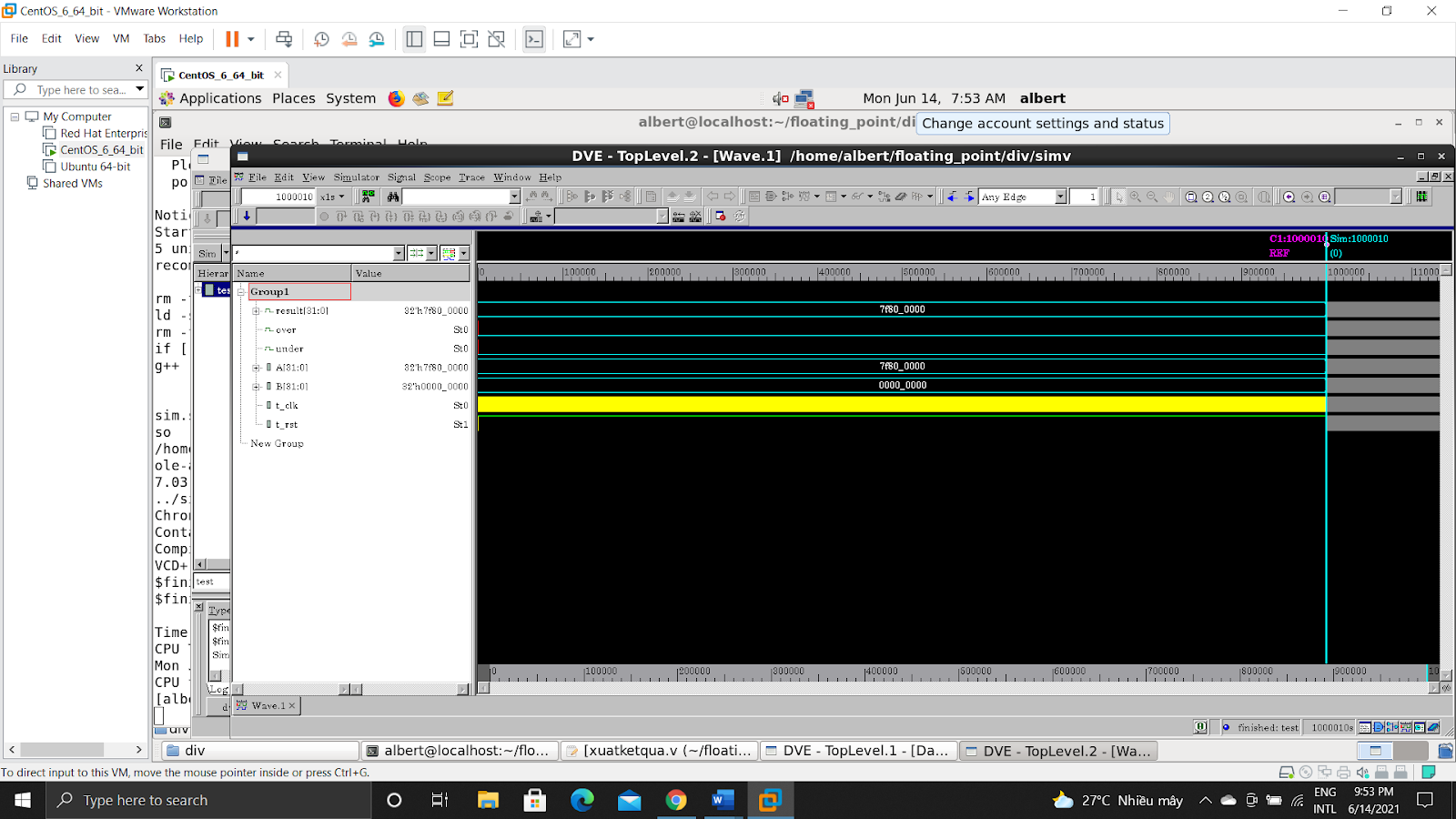
 c.



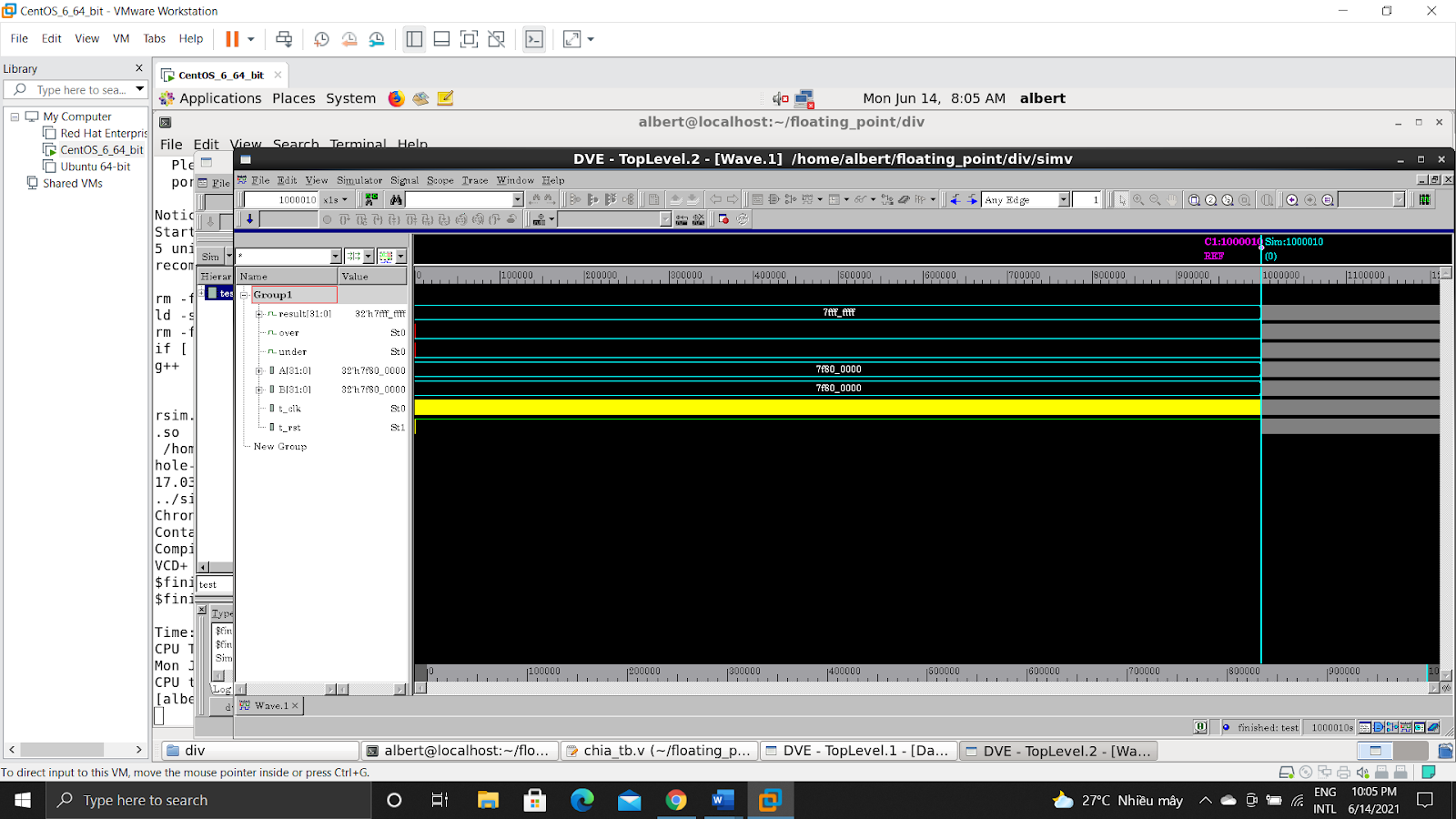
d.



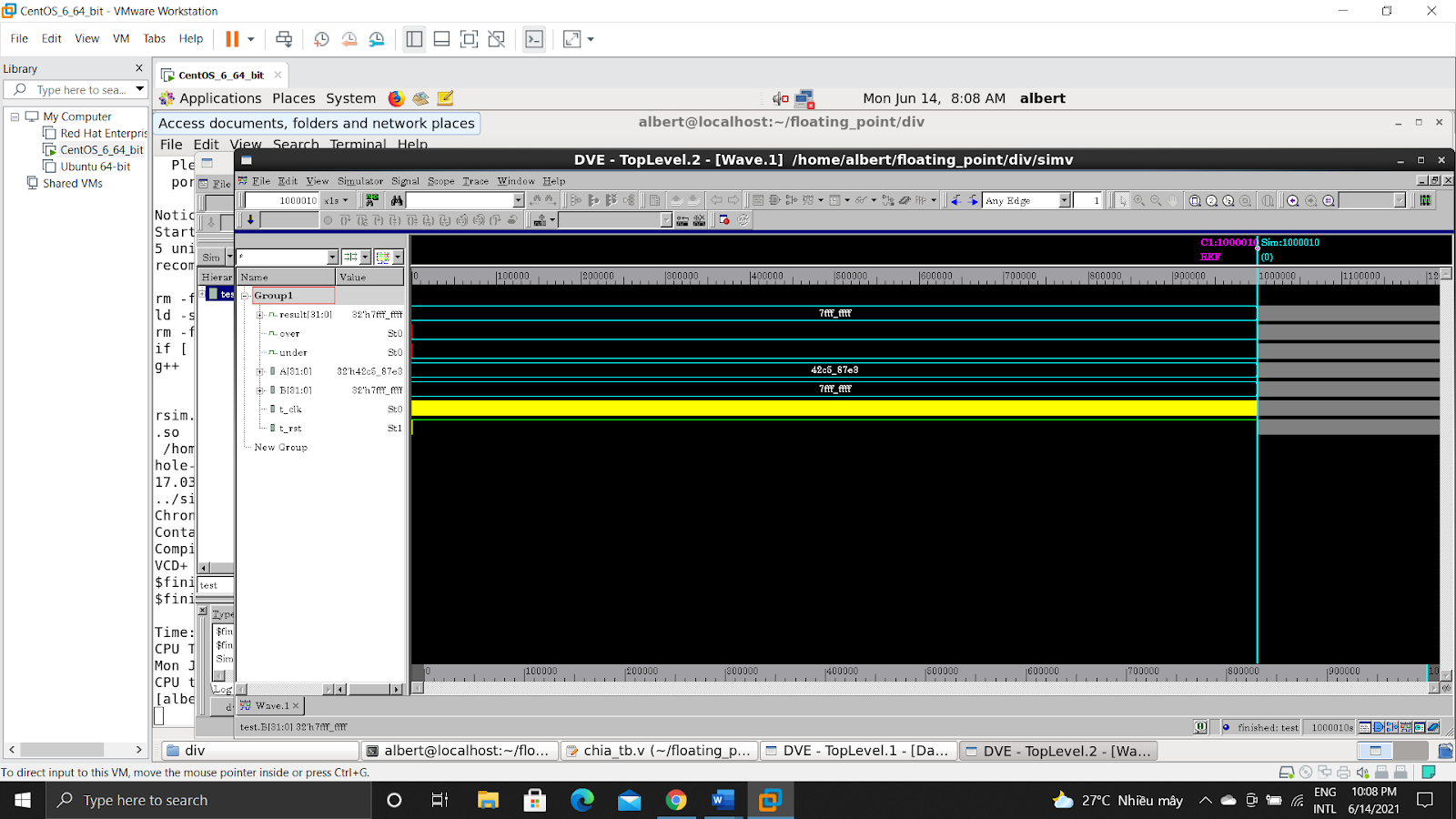
e.



f.



g.



**5.Bộ chia cách 2**

1. Giải thuật bộ chia hai số Floating point

Hai số floating point 32 bit 1 bit thể hiện dấu 8 bit thể hiện số mũ 24 bit thể hiện giá trị

Kiểm tra thử hai số có thuộc trường hợp đặc biết như số có mũ là 0 , số có mũ là vô cùng có giá tri là 23 bit 0 ,số có mũ là vô cùng có giá tri là 23 bit 1 .

Tính bit dấu xor bit 32 của 2 số .

Tính số mũ lấy phần mũ của số a ( số đầu tiên ) trừ đi phần mũ của số b (số thứ hai) và cộng thêm 127.

Thêm 1 bit 1 vào phần faction của 2 số tạo thành số số 24 bit .

So sánh a và b

- Nếu a lớn hơn hoặc bằng b thì bit x ( giảm từ 23 xuống 0 ) của biến kq là 1 thực hiện trừ 23 bit a trừ b , phần dư lưu vào biến fac .

- Nếu a nhỏ hơn b thì bit x ( giảm từ 23 xuống 0 ) của biến kq là 0 , a lưu vào biến fac .

Dịch a qua trái 1 bit và tiếp tục so sánh với b , thực hiện quá trình này 24 lần .

Kết quả thu được biến kq 24 bit

Xét bit 23 của biến kq :

-Nếu là 1 thì phần faction của kết quả là từ bit [22:0] của biến kq và phàn mũ giảm 1

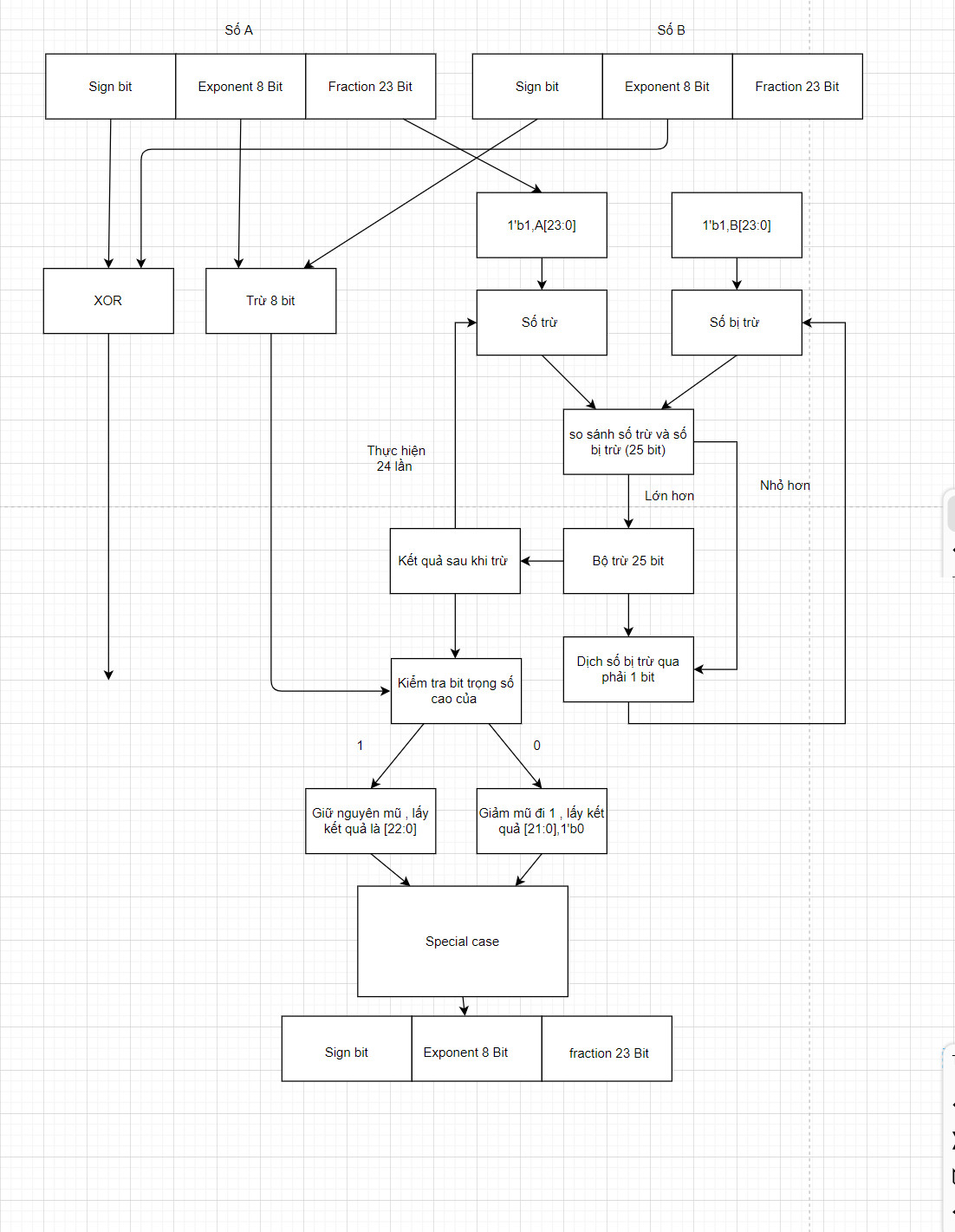
-Nếu là 1 thì phần faction của kết quả là từ bit [21:0] của biến kq và thêm 1 bit 0 ở bit trong số thấp giữ nguyên phần mũ

Kiểm tra bit 10 của mũ nếu là 1 thì kết quả là Zero

Kiểm tra bit 9 của mũ nếu là 1 thì kết quả là Inf

Kiểm tra trường hợp đặc biệt

Kết quả được số floating point 32 bit



1. Trường hợp đặc biệt

Kiểm tra thử hai số có thuộc trường hợp đặc biết như : số có 32 bit 0 ( zero ), số có phần mũ là 8 bit 1 và phần faction có 23 bit 0 (inf) ,số có phần mũ là 8 bit 1 và phần faction có 23 bit 1 (NaN)

Inf=32'b01111111100000000000000000000000

Zero=32'b00000000000000000000000000000000

NaN=32'b01111111111111111111111111111111;

Nếu 1 trong hai số là NaN thì kết quả là NaN

Nếu A là Zero B là Zero thì kết quả là NaN

Nếu A là Zero B là Inf thì kết quả là Zero

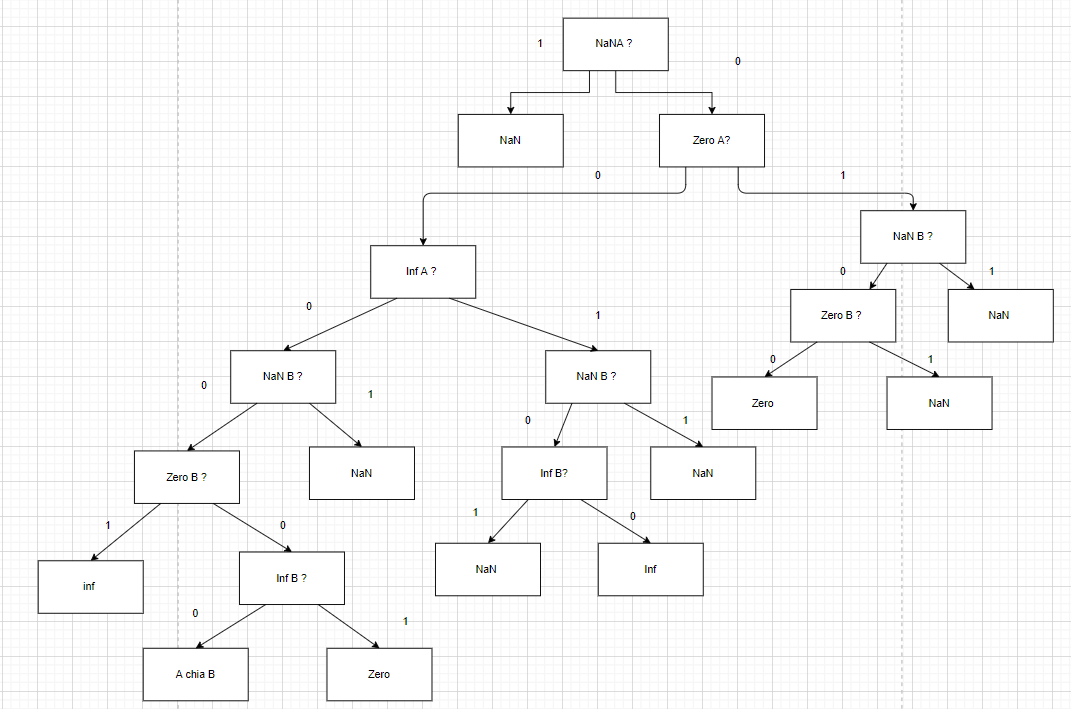
Nếu A là Inf B là Inf thì kết quả là NaN

Nếu A là Inf B là Zero thì kết quả là Inf

Nếu A bình thường B là Zero thì kết quả là Inf

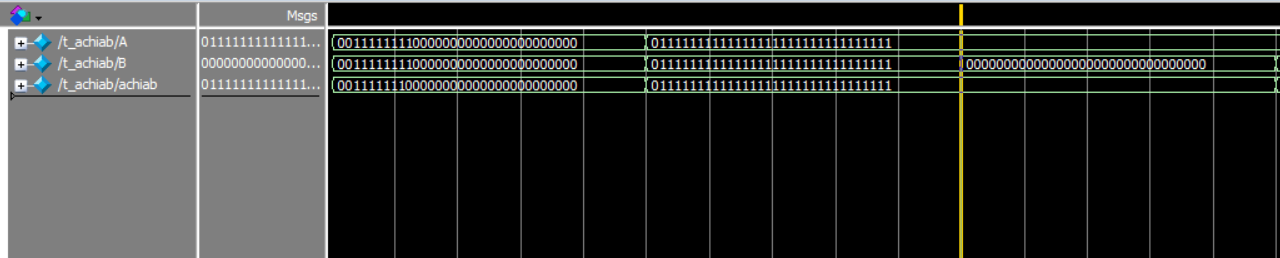
Nếu A bình thường B là Inf thì kết quả là zero

Nếu A bình thường B bình thường thì kết quả là a chia b



Mô hình khối xét trường hợp đặt biệt

1. Testbench và hình ảnh:





#0 B=32'b0\_01111111\_10000000000000000000000;

A=32'b0\_01111111\_10000000000000000000000;

#10 A=32'b01111111111111111111111111111111; //NaN

B=32'b01111111111111111111111111111111; //NaN

#10 B=32'b00000000000000000000000000000000; //+Zero

#10 B=32'b10000000000000000000000000000000; //-Zero

#10 B=32'b01111111100000000000000000000000; //+Inf

#10 B=32'b11111111100000000000000000000000; //-Inf

#10 B=32'b01000001010001011000011110010100;//+12.3456

#10 B=32'b11000001010001011000011110010100;//-12.3456

6.Bộ nhân cách 2

1. Thuật toán nhân

Hai số floating point 32 bit 1 bit thể hiện dấu 8 bit thể hiện số mũ 24 bit thể hiện giá trị

Kiểm tra thử hai số có thuộc trường hợp đặc biết như số có mũ là 0 , số có mũ là vô cùng có giá tri là 23 bit 0 ,số có mũ là vô cùng có giá tri là 23 bit 1 .

Tính bit dấu xor bit 32 của 2 số .

Tính số mũ cộng 2 số mũ của 2 số lại và trừ 127.

Thêm 1 bit 1 vào phần faction của 2 số tạo thành số số 24 bit .

Kiểm tra nếu bit 0 của số phần faction của số A ( số đầu tiên ) là 0 hay 1 .

-Nếu là 1 thì tạo 1 biến 48 bit sum 24 bit đầu là 0 24 bit sau là phần faction của B (số thứ hai ) (24’d0,B).

-Nếu là 0 thì tạo 1 biến 48 bit sum 48 bit đầu là 0.

Lần lượt kiểm tra biến x ( tăng từ 1 đến 23 )của số phần faction của số A ( số đầu tiên ) là 0 hay 1

- Nếu là 1 thì sum cộng 48 bit với số C 48 bit {24’d0, A,1’b0}.

- Nếu là 0 thì sum cộng 48 bit với số 48’d0.

Tăng x lên 1 đơn vị và dịch biến C qua trái 1 bit lặp lại tới khi x > 23 thì dừng lại

Lấy 25 bit trọng số cao của biến sum là phần 1 bit dư và 24 bit chứ giá trị của kết quả

Xét bit 48 của biến sum ( bit dư )

- Nếu là 1 có nghĩa là sau khi cộng 48 bit có dư nên mũ sẽ tăng thêm 1 và faction của kết quà là từ bit [23:1]

Vd: 1\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx

xxxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxx

- Nếu là 0 có nghĩa là sau khi cộng 48 bit không có dư nên mũ không đổi và faction của kết quà là từ bit [22:0]

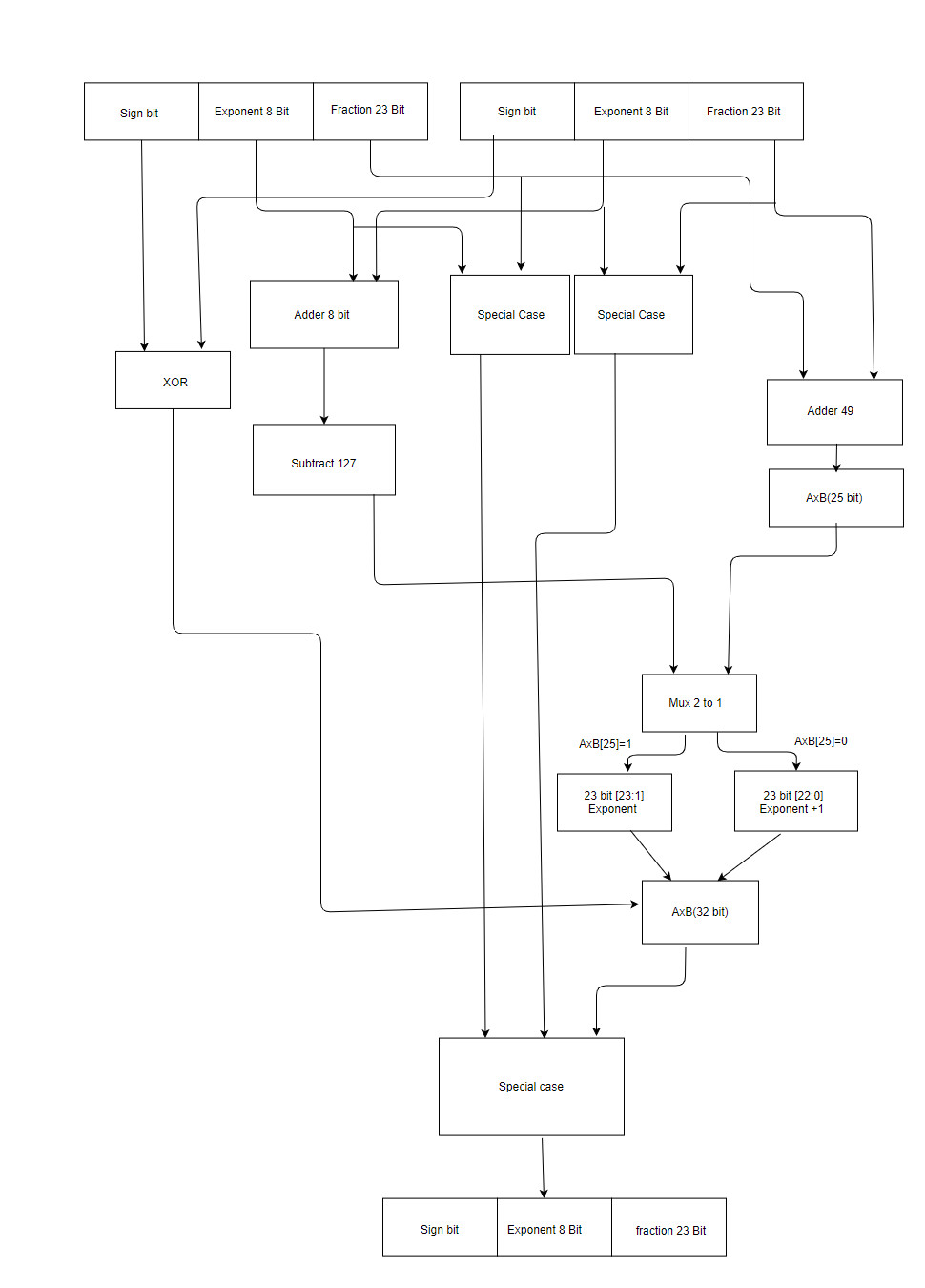
Vd: 0\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx

xxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx

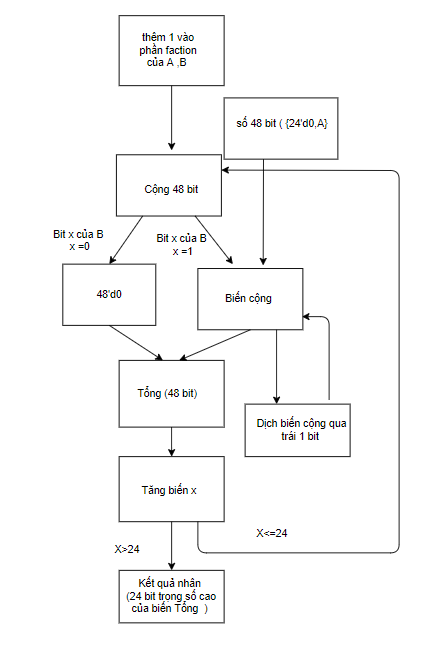
Kiểm tra bit 10 của mũ nếu là 1 thì kết quả là Zero

Kiểm tra bit 9 của mũ nếu là 1 thì kết quả là Inf

Kết quả được số floating point 32 bit



Mô hình bộ nhân 2 số floating point



Mô hình khối nhân faction

1. Trường hợp đặc biệt

Kiểm tra thử hai số có thuộc trường hợp đặc biết như : số có 32 bit 0 ( zero ), số có phần mũ là 8 bit 1 và phần faction có 23 bit 0 (inf) ,số có phần mũ là 8 bit 1 và phần faction có 23 bit 1 (NaN)

Inf=32'b01111111100000000000000000000000

Zero=32'b00000000000000000000000000000000

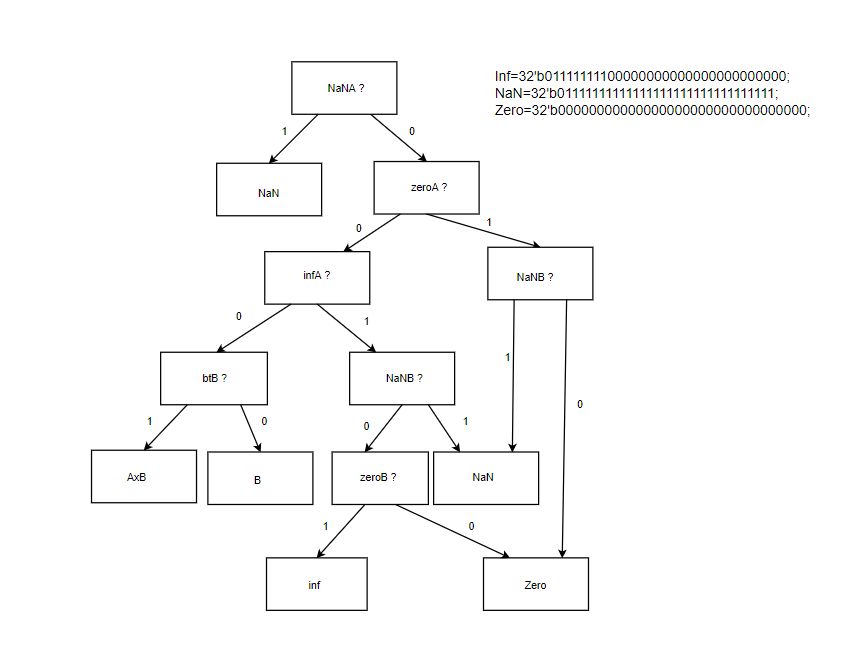
NaN=32'b01111111111111111111111111111111;

Nếu 1 trong hai số là NaN . thì kết quả là một số NaN

Nếu 1 số là zero và 1 số là inf thì kết quả là zero

Nếu 2 số là zero thì kết quả là zero

Nếu 2 số là inf thì kết quả là inf



Mô hình khối xét trường hợp đặt biệt

1. Testbench và hình ảnh:

#0

A=32'b0\_01111111\_00000000000000000000000; //1

B=32'b0\_01111111\_00000000000000000000000; //1

#10

A=32'b1100\_1011\_1111\_1111\_1001\_0101\_1011\_0000; B=32'b1101\_0000\_0000\_1111\_0000\_1101\_0001\_1000;

#10

A=32'b0\_01111111\_00000000000000000000000; //0.02345

B=32'b0\_01111010\_00011011000111011001001; //0.03456

#10

A=32'b0\_01111111\_10000000000000000000000; //0.02345

B=32'b0\_01111111\_11000000000000000000000; //0.03456

#10 A=32'b0\_11111111\_11000000000000000000000;

B=32'b0\_11111111\_10000000000000000000000;

#10 A=32'b0\_00000001\_11000000000000000000000;

B=32'b1\_00000001\_00000000000000000000000;

#10 A=32'b0\_11111110\_11000000000000000000000; // ex(A) + ex(B) -127 = 8'b 1111\_1111

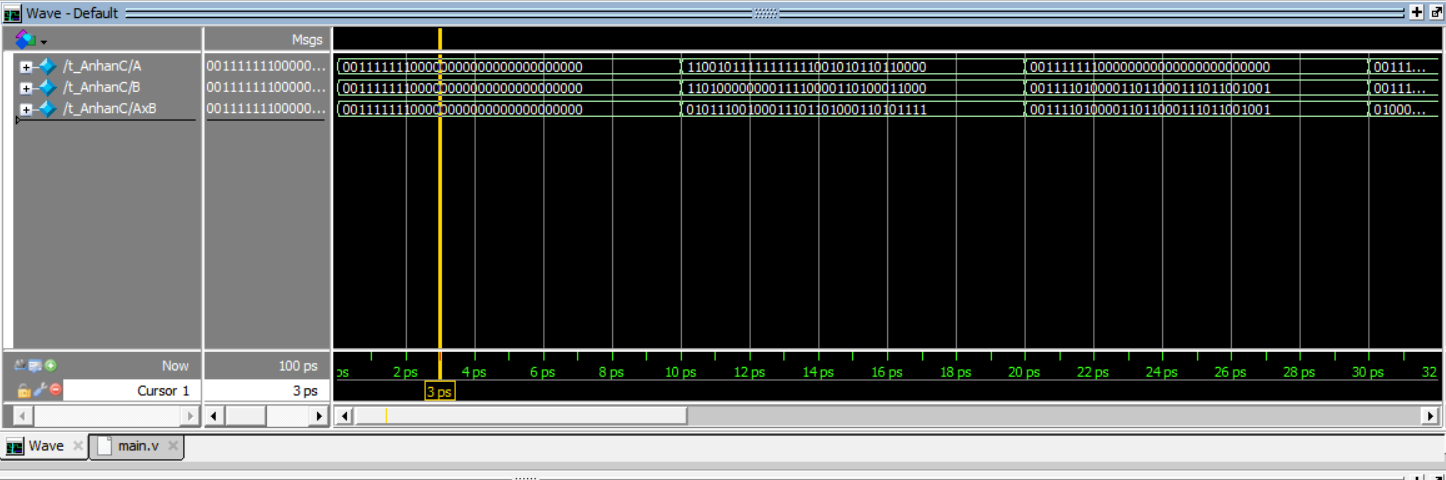
B=32'b1\_10000000\_00000000000000000000000;

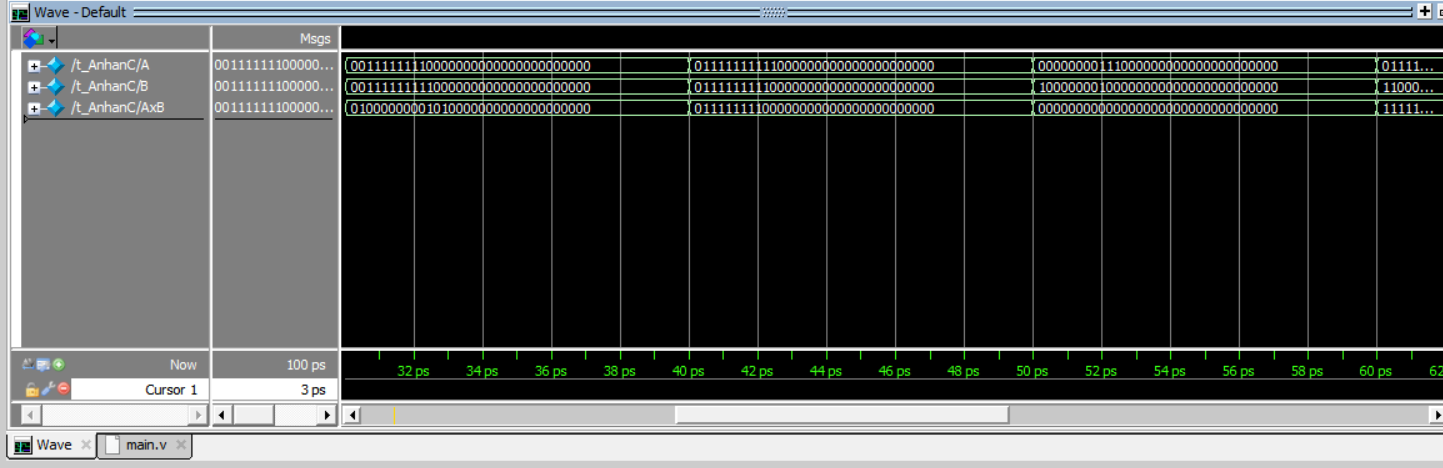
#10 A=32'b1\_11111110\_11111111111111111111111;

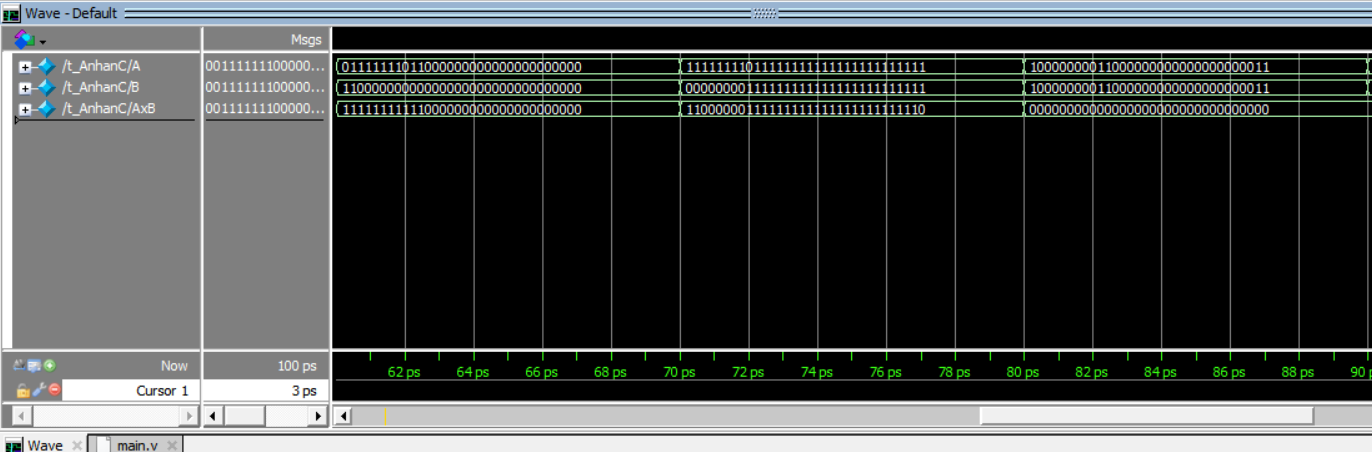
B=32'b0\_00000001\_11111111111111111111111;

#10 A=32'b1\_00000000\_11000000000000000000011;

B=32'b1\_00000000\_11000000000000000000011;







7.Bộ chuyển đổi từ số floating point sang số thực

1. Giải thuật

- Lấy phần mũ của số floating point so sánh với 127

+ Nếu phần mũ lớn hơn thì lấy phần mũ trừ 127 ta được x và biến

y =8'h2B (+)

+ Nếu phần mũ bé hơn thì lấy 127 trừ phần mũ ta được x và biến

y =8'h2D (-)

- Thêm 1 bit 1 vào đầu của phần faction t được 1 số 24 bit

- Nếu y =8'h2B (+) thì đặt biến a là phần nguyên là từ bit 24 qua phải x bit của phần faction . Phần còn lại của faction là b phần thập phân .

- Nếu y =8'h2B (-) thì đặt biến a là phần nguyên là 0 . Biến b phần thập phân thêm x bit 0 trước phần faction .

Vd:

y =8'h2B (+) , phần faction là 1\_0000\_1000\_1111\_0000\_0000\_111 ,

x là 5

a = 24’b0000\_0000\_0000\_0000\_0001\_0000

b = 24’b1000\_1111\_0000\_0000\_1110\_0000

y =8'h2D (-) , phần faction là 1\_0000\_1000\_1111\_0000\_0000\_111 ,

x là 5

a = 24’b0000\_0000\_0000\_0000\_0000\_0000

b = 24’b0000\_0100\_0100\_0111\_1000\_0000

- Từ a và b tính phần thập phân và phần nguyên sau đó cộng lại với nhau biến sum

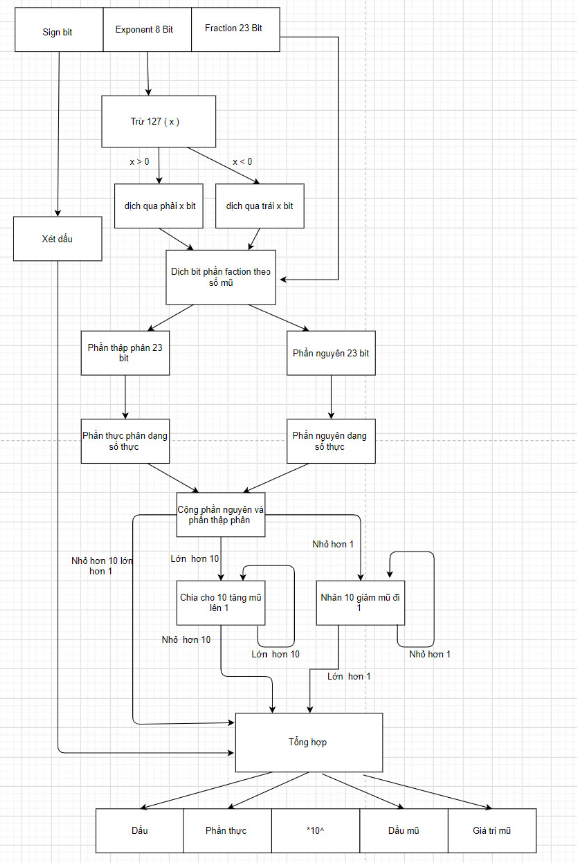
+ Nếu sum >10 thì chia sum cho 10 và tăng biến mũ lên 1 biến dấu của mũ là +

+ Nếu sum <1 thì nhân sum cho 10 và tăng biến mũ lên 1 biến dấu của mũ là -

- Dựa vào bit 32 của số floating point xác dịnh dấu của số tương ứng

+ 1 thì dấu là -

+ 0 thì dấu là +



Sơ đồ khối bộ đổi từ số floating point sang số thực

1. Testbench và hình ảnh:

#0 a=24'b0001\_0000\_0000\_0000\_0000\_1010; c=23'b100\_1100\_0100\_1011\_0100\_0000;

#10 a=24'b0000\_0000\_0000\_0000\_1010\_0100;c=23'b0000\_0000\_0000\_0000\_0000\_0000;

#10 a=24'b0000\_0000\_0000\_0000\_0000\_0000;c=23'b100\_1100\_0100\_1011\_0100\_0000;

#10 a=24'b0000\_0000\_0000\_0010\_1000\_0000;

#10 a=24'b0000\_0000\_0000\_0000\_0000\_0000;c=23'b000\_0000\_0000\_0000\_0000\_1000;

//#10 in1=8'b0111\_1111; in1=8'b1000\_0000;

