BOR3**ĐẠI HỌC QUỐC GIA TP. HCM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**BÀI TẬP LỚN MÔN HỌC**

**KỸ THUẬT SỐ NÂNG CAO**

***ĐỀ TÀI 3***

**THIẾT KẾ MỘT MÁY TÍNH DẤY CHẤM ĐỘNG THỰC HIỆN CÁC PHÉP TOÁN (+, -, \*, /) GIỮA HAI SỐ FLOATING POINT ( IEEE754, SINGLE PRECISION, 32-BIT)**

**LỚP A01, HK192**

**GVHD: TRẦN HOÀNG LINH**

**TRỊNH VŨ ĐĂNG NGUYÊN**

**SINH VIÊN THỰC HIỆN:**

**NGUYỄN HOÀNG BÁCH – 1710566**

**­­**

**TP. HỒ CHÍ MINH, NĂM 2020**

1. **Yêu cầu:**

**THIẾT KẾ MỘT MÁY TÍNH DẤY CHẤM ĐỘNG THỰC HIỆN CÁC PHÉP TOÁN (+, -, \*, /) GIỮA HAI SỐ FLOATING POINT ( IEEE754, SINGLE PRECISION, 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)**

1. **Cơ sở lý thuyết**
2. **Dấu chấm động (floating point number)**

* Trong máy tính, các số nhị phân phải được đưa về dạng chuẩn như sau:
* Số chấm động chính xác đơn (32 bits) chuẩn IEEE 754:  
  ---
* Sign: Bit dấu (1: số âm, 0: số dương).
* Exponent: Số mũ (Biểu diễn với dạng số quá K (Biased) K = 127 (2n-1- 1 = 28-1 - 1) với n là số bit lưu trữ.
* Significand (Fraction): Phần định trị (phần lẻ sau dấu chấm).
* Trường hợp đặc biệt :
* Tràn trên (overflow): trường hợp này xảy ra khi kích thước của số mũ lớn hơn kích thước giới hạn trên (số mũ dương).
* Tràn dưới (underflow): trường hợp này xảy ra khi kích thước của số mũ nhỏ hơn kích thước giới hạn dưới ( số mũ âm).

1. **Cộng/trừ 2 số floating point.**

* Chia module lớn thành 4 module nhỏ : dương A cộng B, dương A trừ B, âm A cộng B, âm A trừ B sau đó sử dụng bộ mux4 để chọn kết quả tương đương dựa vào dấu của A và B.
* Với mỗi module ta chia ra 2 số mũ nhỏ và số mũ lớn, dịch số mũ nhỏ với hiệu mũ 2 số để bằng số mũ lớn rồi thực hiện cộng/trừ phần trị 2 số sử dụng có module full adder và full subtracter tương ứng. Để cộng/trừ phần trị 2 số thì thêm 1 bit 1 vào trước 23 bit trị số. Chọn kết quả phần trị dựa vào cờ nhớ của phép cộng/trừ đó.
* Việc chọn số mũ tổng/hiệu cũng phụ thuộc vào cờ nhớ của phép tính trên.
* Về phần chọn dấu thì:

+ Module dương A cộng B thì dấu luôn dương.

+ Module âm A trừ B thì dấu luôn âm.

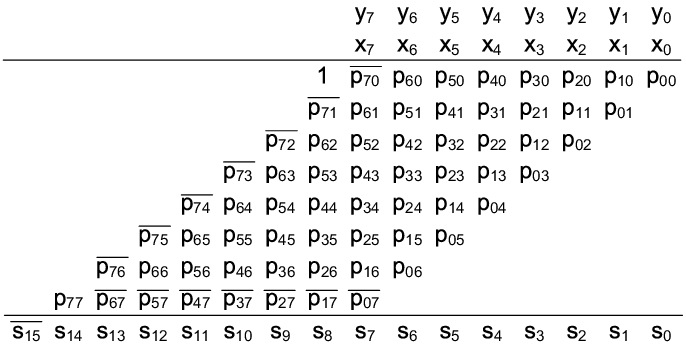
+ Module dương A trừ B thì chia ra 2 trường hợp: dấu sẽ là âm nếu mũ A nhỏ hơn mũ B, dấu là dương nếu mũ A lớn hơn mũ B.

+ Module âm A cộng B thì dấu ngược lại với module dương A trừ B.

* Với mỗi module sẽ có 1 module nhỏ kiểm tra các trường hợp đặc biệt và xuất ra kết quả cuối cùng.

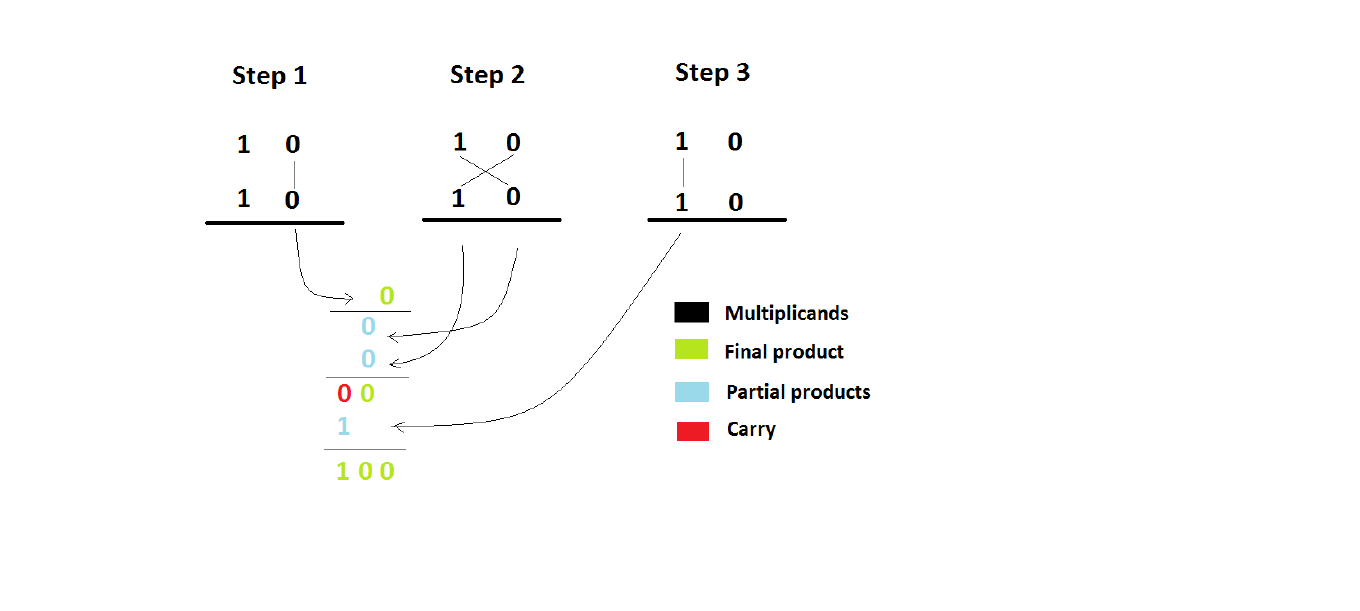
1. **Nhân 2 số floating point(2 phương pháp).**

* **Phương pháp shift +:**
* Module A nhân B sẽ thực hiện nhân 2 số 24 bit tương đương bit 1 ghép với 23 bit phần trị của 2 số A và B.
* Thực hiện phép nhân bằng cách cho 1 biến tổng có độ lớn 48 bit, sau đó sẽ cộng với trị A nếu bit 0 của B bằng 1,sau đó dịch trái trị A 1 bit rồi lại thực hiện cộng với tổng đó nếu bit 1 của B bằng 1 và tiếp tục cho đến hết trị B. Giá trị cần lấy là 23 bit từ bit 45 đến bit 23 của biến tổng đó bởi vì trừ ra 2 bit cuối do ghép thêm bit 1 vào phần trị của phép nhân.



* Phần mũ của phép nhân thì sẽ thực hiện tổng hoặc hiệu tương ứng với dấu của mũ A và B.
* Phần dấu của phép nhân thì bằng A XOR B.
* Sau đó đưa kết quả cuối qua 1 module kiếm tra trường hợp đặc biệt rồi xuất ra kết quả cuối.
* **Phương pháp nhân vedic:**
* Phương pháp này sẽ dùng chung phần tìm dấu và tìm mũ với phương pháp shift +
* Phần tìm trị thì sẽ sử dụng phương pháp vedic:

+ Đầu tiên xây dựng module nhân vedic 2 bit:



**+** Sau đó sử dụng các module cộng và module nhân vedic 2 bit để tạo module vedic 4 bit và tương tự cho đến khi tạo ra module nhân vedic 32 bit.

**+** Truyền 24 bit gồm bit 1 ghép với 23 bit phần trị của A và B tương ứng vào, phần thiếu thì ghép thêm bit 0 và thực hiện phép nhân vedic trên module nhân vedic 32 bit, ta được kết quả là 48 bit đầu của phép nhân trên tương đương với nhân 2 số 24 bit.

**+** Sau đó ta lấy phần trị cho phép nhân vedic là 23 bit từ 45 đến 23 hoặc từ 46 đến 24 phụ thuộc vào bit thứ 47.

* Sau đó đưa kết quả cuối qua 1 module kiếm tra trường hợp đặc biệt rồi xuất ra kết quả cuối.

1. **Chia 2 số floating point( sử dụng phương pháp shift -).**

* Module A chia B được truyền vào 2 số 24 bit gồm 1 bit 1 ghép với 23 bit phần trị của A và B tương ứng và kết quả ra là 25 bit. Lấy A trừ B , bit thứ 24 của kết quả là 1 hay 0 phụ thuộc vào cờ mượn của phép trừ. Để thực hiện tiếp thì lấy biến có giá trị bằng kết quả phép trừ hoặc A phụ thuộc vào cờ mượn. Sau đó lấy biến đó dịch sang trái 1 bit rồi thực hiện phép trừ biến với B, bit thứ 23 của kết quả là 1 hay 0 phụ thuộc vào cờ mượn phép trừ, và dần dần cho tới khi có bit thứ 0 của kết quả. Sau đó lấy phần trị của phép chia là 23 bit từ bit thứ 22 đến 0 hoặc bit thứ 23 đến 1 phụ thuộc vào bit thứ 24 của kết quả trên.
* Phần mũ của phép chia thì sẽ thực hiện tổng hoặc hiệu tương ứng với dấu của mũ A và B và bit thứ 24 của kết quả trên.
* Phần dấu của phép chia thì bằng A XOR B.

1. **Sơ đồ giải thuật**
2. **Bộ cộng/trừ.**

Trường hợp đặc biệt?

Tìm phần mũ tổng/hiệu

Cộng/trừ phần định trị

Tìm độ lệch giữa phần mũ 2 số

Dịch số có mũ nhỏ sang phải với số lần bằng độ lệch mũ

Y

N

1. **Bộ nhân/ chia tương tự.**
2. **Sơ đồ phần cứng.**
3. **Bộ cộng/trừ.**

F

S

Trường hợp đặc biệt

Bộ cộng/trừ(ALU lớn)

Dịch phải theo lệch mũ

0 1

0 1

Độ lệch mũ

Bộ ALU nhỏ

E

F

S

F

E

S

E

1. **Bộ nhân.**

XOR

Small ALU

E

F

E

S

F

S

Bộ nhân

Trường hợp đặc biệt

E

S

F

1. **Bộ chia**

E

XOR

Small ALU

E

F

S

F

S

Bộ chia

Trường hợp đặc biệt

E

S

F

1. **Mô phỏng**
2. **Module cộng.**

* Trường hợp 1: số bất kì.

A :decimal 240

Binary 01000011011100000000000000000000

Hex 0x43700000

B : decimal 303.1

Binary 01000011100101111000110011001101

Hex 0x43978ccd

S : decimal 543.1

Binary 01000100000001111100011001100110

Hex 0x4407c666



* Trường hợp đặc biệt:

A :decimal inf

Binary 01111111100000000000000000000000

Hex 0x7f800000

B :decimal inf

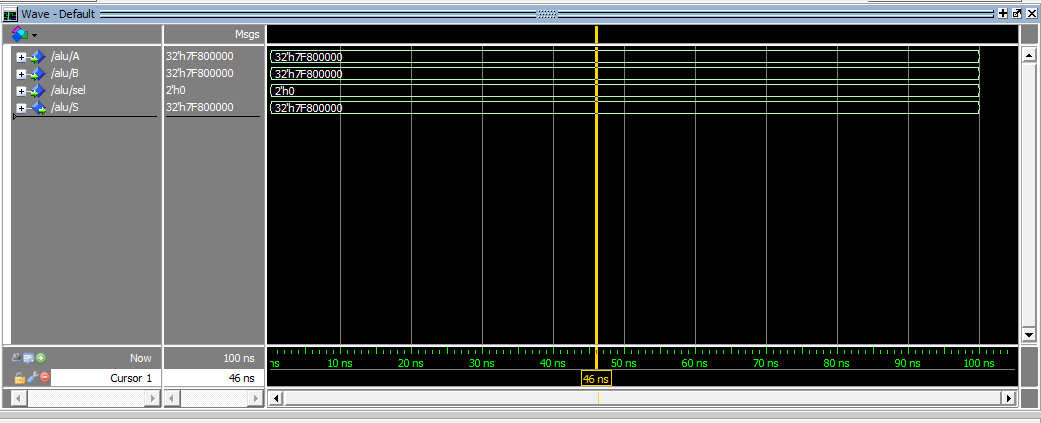
Binary 01111111100000000000000000000000

Hex 0x7f800000

S :decimal inf

Binary 01111111100000000000000000000000

Hex 0x7f800000



1. **Module trừ.**

* Trường hợp 1: số bất kì

A : decimal -240

Binary 11000011011100000000000000000000

Hex 0xc3700000

B : decimal 303.1

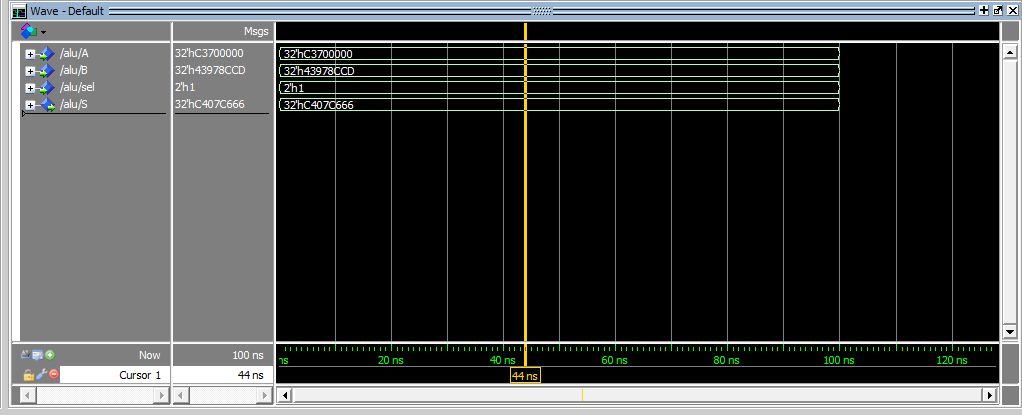
Binary 01000011100101111000110011001101

Hex 0x43978ccd

S :decimal -543.1

Binary 11000100000001111100011001100110

Hex 0xc407c666



* Trường hợp đặc biệt:

A :decimal NaN

Binary 01111111111111111111111111111111

Hex 0x7fffffff

B :decimal 100

Binary 01000010110010000000000000000000

Hex 0x42c80000

S :decimal NaN

Binary 01111111111111111111111111111111

Hex 0x7fffffff

1. **Module nhân**

* **Shift +:**
* Trường hợp 1: số bất kì

A :decimal -200

Binary 11000011010010000000000000000000

Hex 0xc3480000

B :decimal 1.5

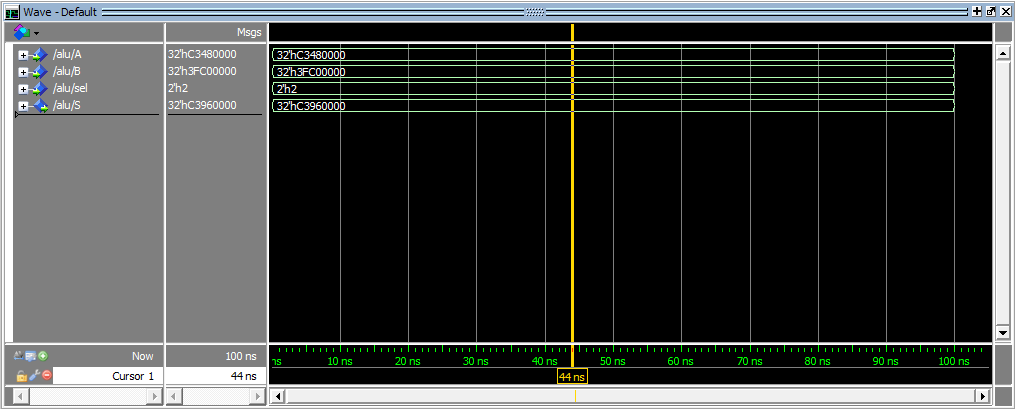
Binary 00111111110000000000000000000000

Hex 0x3fc00000

S :decimal -300

Binary 11000011100101100000000000000000

Hex 0xc3960000



* Trường hợp đặc biệt:

A :decimal -inf

Binary 11111111100000000000000000000000

Hex 0xff800000

B :decimal inf

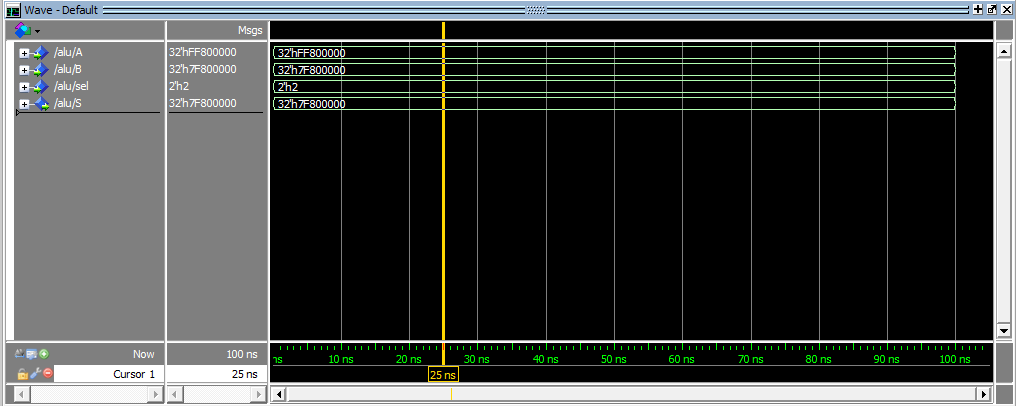
Binary 01111111100000000000000000000000

Hex 0x7f800000

S : decimal inf

Binary 01111111100000000000000000000000

Hex 0x7f800000



* **Vedic:**
* Trường hợp 1: số bất kì

A :decimal -200

Binary 11000011010010000000000000000000

Hex 0xc3480000

B :decimal -500

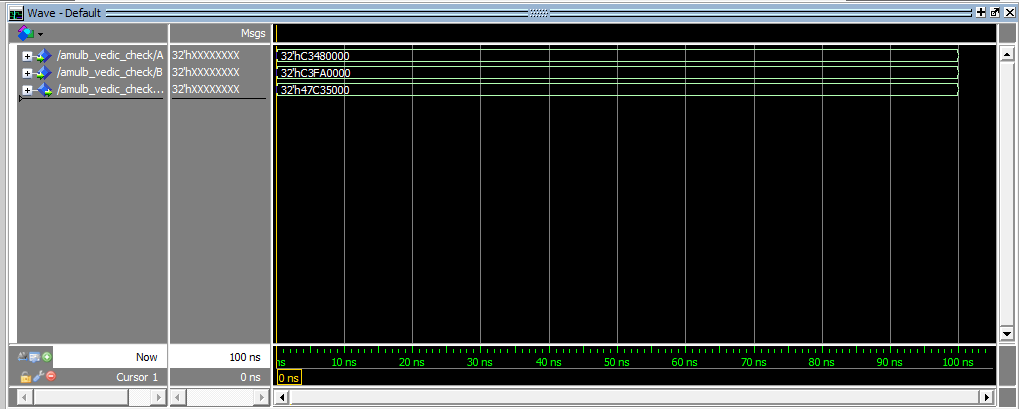
Binary 11000011111110100000000000000000

Hex 0xc3fa0000

S :decimal 100000

Binary 01000111110000110101000000000000

Hex 0x47c35000



1. **Module chia**

* Trường hợp 1: số bất kì.

A :decimal 143

Binary 01000011000011110000000000000000

Hex 0x430f0000

B :decimal -32

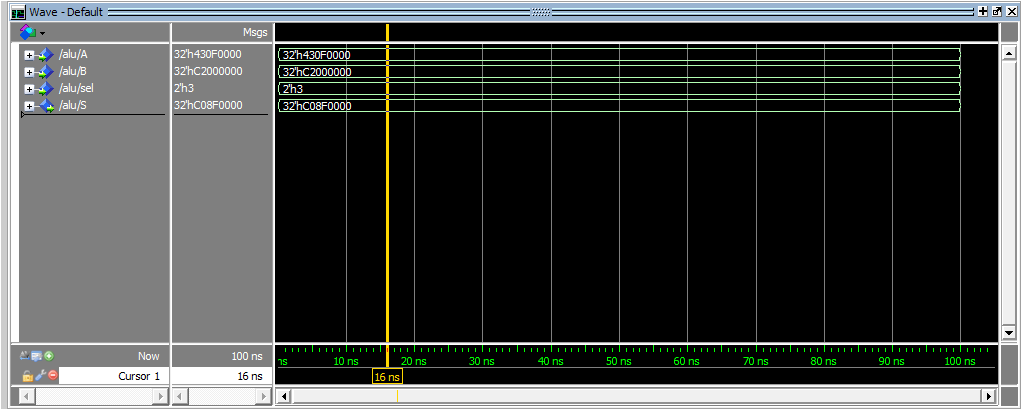
Binary 11000010000000000000000000000000

Hex 0xc2000000

S :decimal -4.46875

Binary 11000000100011110000000000000000

Hex 0xc08f0000



* Trường hợp đặc biệt

A :decimal 143

Binary 01000011000011110000000000000000

Hex 0x430f0000

B :decimal 0

Binary 00000000000000000000000000000000

Hex 0x00000000

S :decimal NaN

Binary 01111111111111111111111111111111

Hex 0x7fffffff

