

**ĐẠI HỌC QUỐC GIA TP.HCM**  
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**  
**KHOA ĐIỆN-ĐIỆN TỬ**

**BỘ MÔN ĐIỆN TỬ**



**BÁO CÁO BÀI TẬP LỚN**

**KỸ THUẬT SỐ NÂNG CAO**

*GVHD:* Trần Hoàng Linh

<b>Họ và tên</b>	<b>MSSV</b>
Nguyễn Trọng Tín	1814343
Dương Nhật Huy	1810162
Chung Hữu Nhân	1710215

# ĐỀ TÀI

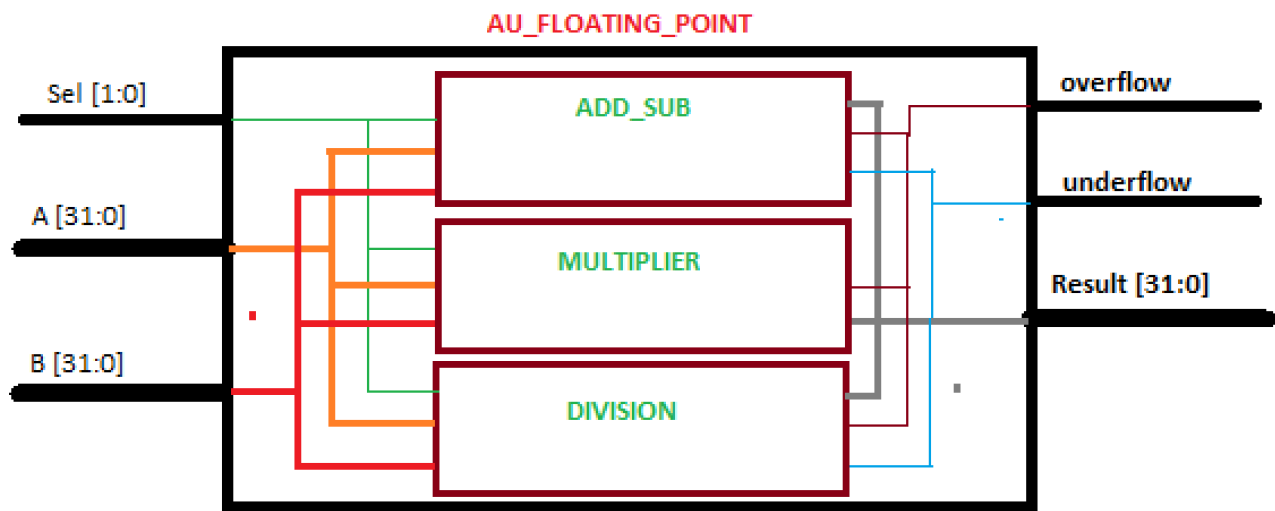
## **PHẦN I: THIẾT KẾ MỘT MÁY TÍNH DẤU CHẤM ĐỘNG THỰC HIỆN CÁC PHÉP TOÁN (+, -, \*, /) GIỮA HAI SỐ FLOATING POINT (IEEE-754, SINGLE PRECISION, 32-BIT). (40%) (bắt buộc)**

- 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)

## **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.  $5.25 \cdot 10^{-5}$ ,  $-2.5 \cdot 10^3 \dots$ ): + 10% tổng kết

Sơ đồ khối hệ thống máy tính dấu chấm động:



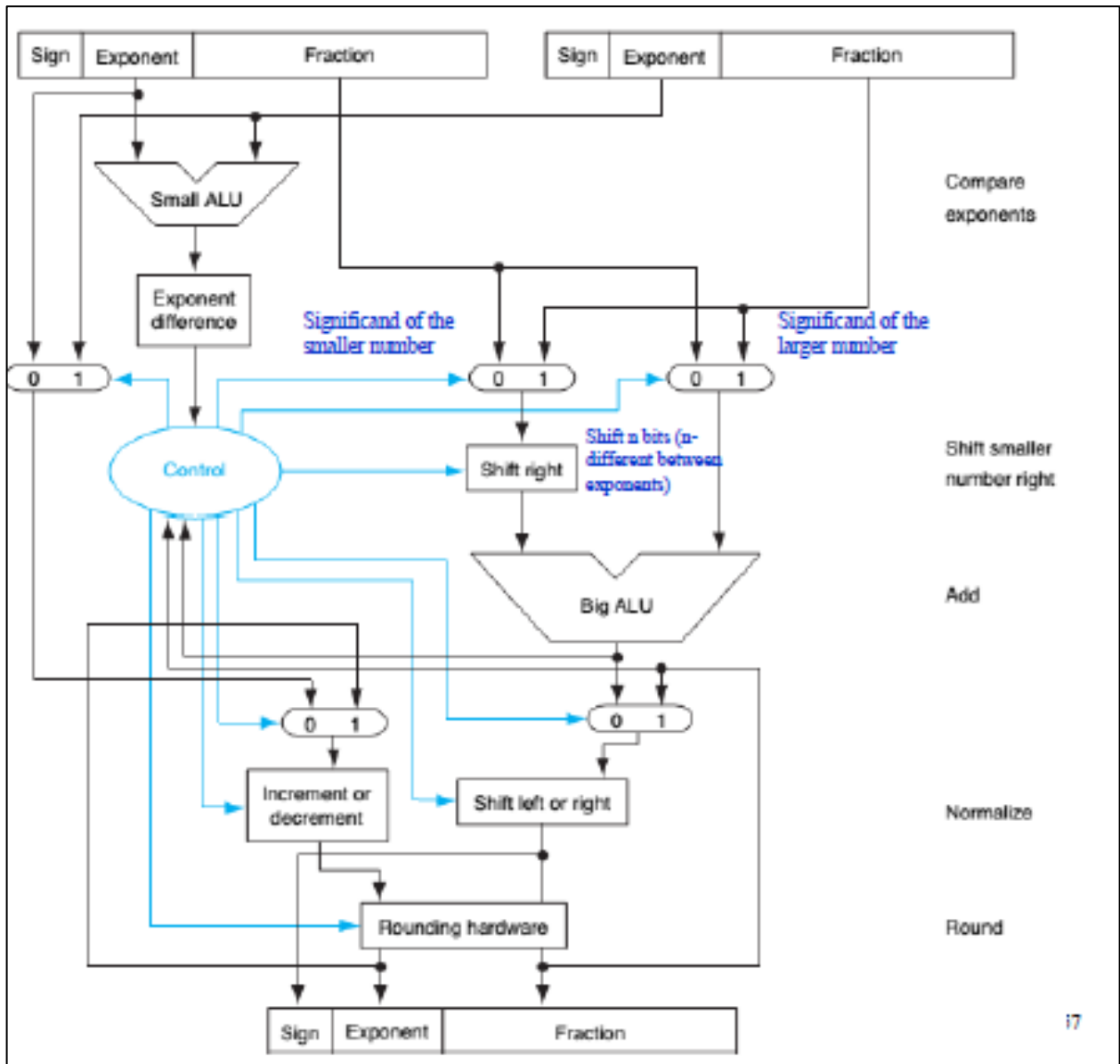
Top Module AU\_FLOATING\_POINT chứa 3 module nhỏ bên trong là ADD\_SUB, MULIPLIER, và DIVISION để thực hiện các phép tính công trừ nhân chia, lựa chọn phép tính dựa vào Sel:

MATH	SEL
Add	00
Sub	01
Multiplier	10
Division	11

# **PHẦN I: THIẾT KẾ MỘT MÁY TÍNH DẤU CHẤM ĐỘNG THỰC HIỆN CÁC PHÉP TOÁN (+, -, \*, /) GIỮA HAI SỐ FLOATING POINT (IEEE-754, SINGLE PRECISION, 32-BIT) – GIẢI THUẬT THỨ NHẤT**

## **I. Bộ cộng – trừ:**

### **1. Sơ đồ khối:**



## 2. Chi tiết giải thuật:

- Hai số ngõ vào 32-bit floating-point **float1**, **float2**, ngõ ra **result**.
- Kiểm tra trường hợp đặc biệt (zero, infinity, NaN...).
- Lấy mũ của số lớn làm số mũ chuẩn và trừ cho số bé để tìm số bit cần dịch để cân bằng phần phân số sao cho có cùng số mũ chuẩn.
- Cộng trừ hai phần phân số vừa dịch theo dấu của phép tính (thêm bit 1 ở bên trái thập phân để tiện chuẩn hóa kết quả).
- Tìm bit 1 đầu tiên của kết quả từ trái sang để chuẩn hóa theo IEEE-754, đồng thời đếm số lần dịch để cân bằng số mũ cuối của kết quả.
- Tính số mũ cuối và kiểm tra số mũ xong có nằm trong vùng giá trị không (-126 đến 127), nếu không thì kiểm tra xem là underflow hay overflow.
- Xuất ra kết quả cuối cùng.

## 3. Thực hiện phép tính

Vì theo chuẩn IEEE-754 thì phần dấu được biểu thị bằng bit riêng nên khi tính toán, phần phân số luôn được cho là dương nên ta sẽ xét đến tổ hợp của bit dấu và dấu của phép tính để biết phép tính được thực hiện như thế nào.

Dấu (+)	float1 > 0	float1 < 0
float2 > 0	float1 + float2	- float1 + float2
float2 < 0	float1 - float2	- float1 - float2

Dấu (-)	float1 > 0	float1 < 0
float2 > 0	float1 - float2	- float1 - float2
float2 < 0	float1 + float2	- float1 + float2

#### 4. Kết quả mô phỏng bộ cộng:

Msgs	/tb/float1	/tb/float2	/tb/K	/tb/result	/tb/overflow	/tb/underflow
4ea0...	4bf42400	3e0a233a	00000000	7f800000	00000000	7f800000
d04a...	4fcaa7e2	3e08b439	00000000	7f800000	7f800000	7f800000
01	00					
505e...	4fcb9c06	bc96bb98	00000000	7f800000		
St0						
St0						

⇒ Đúng tất cả các trường hợp.

#### 4. Kết quả mô phỏng bộ trừ:

Msgs	/tb/float1	/tb/float2	/tb/K	/tb/result	/tb/overflow	/tb/underflow
404c...	cbff95b0	4ea0eebb	00000000	7f800000	00000000	7f800000
7fffffff	d00f0d18	d04aa7e2	00000000	7f800000	7f800000	7f800000
01	01					
zzzzzzzz	500e8d4e	505ec5b9	80000000	7f800000		
StX						
StX						

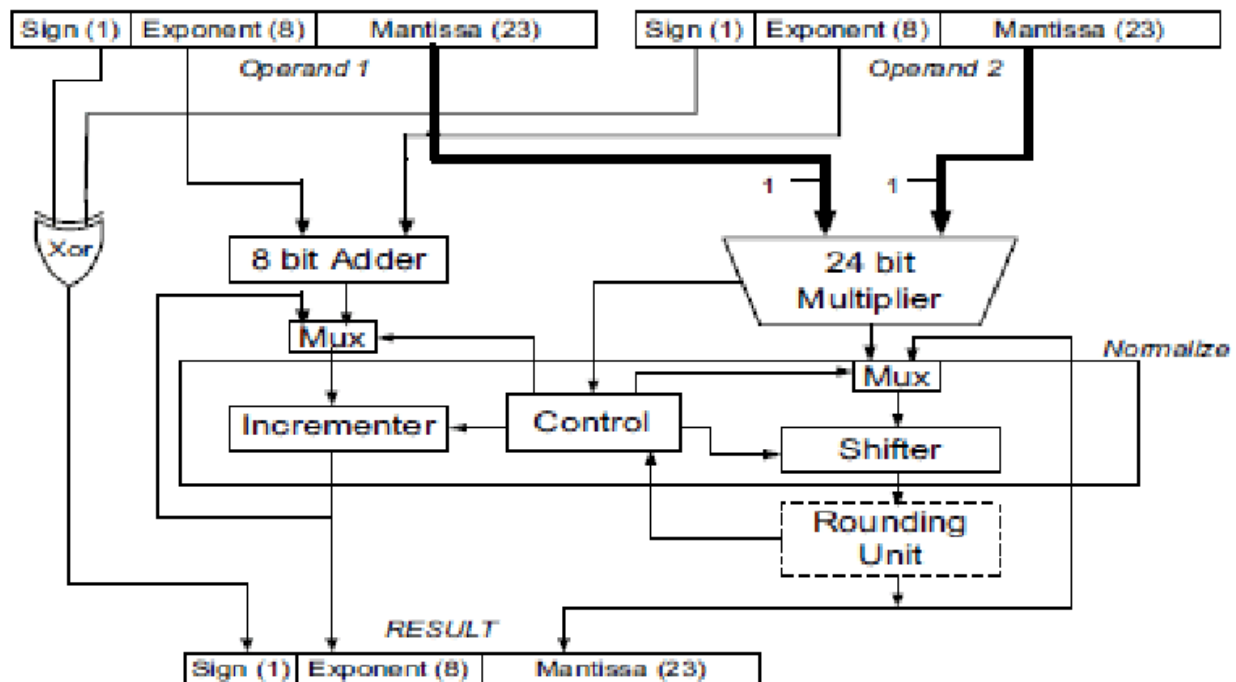
⇒ Đúng tất cả các trường hợp.

## II. Bộ nhân:

Ngõ vào A, B là 2 số 32 bit

Ngõ ra là S là số 32 bit

### 1. Sơ đồ khối



## 2. Giải thuật:

Hai số hạng vào là A, B, kết quả là S

Bước 1: Kiểm tra bit dấu hai số hạng A, B. Nếu A, B cùng dấu, dấu của S là (+), Nếu A, B khác dấu, dấu của S là (-).

$$S[31] = A[31] \text{ xor } B[31]$$

Bước 2: Kiểm tra 2 các trường hợp đặc biệt của 2 số A, B.

Nếu  $A[30:23] = 11111111$  thì  $A = \infty$

Nếu  $A[30:0] = 0$  thì  $A = 0$

Tương tự với B

Bước 3: Nhân hai phần định trị (significant) của hai số hạng A, B

- A0, B0 chính là phần định trị của A, B và được thêm bit 1 phía trước
- Tiến hành phép nhân 24-bit của số A0 và B0. Sử dụng 23 bộ cộng 24 bit nối tiếp nhau.

Nếu  $B[0] = 1$  thì  $S0[0] = A$ , Nếu  $B[1] = 0$  thì  $S0[0] = 0$ .

Tại bộ cộng thứ 1, Nếu  $B[1] = 0$ , thì  $S0[1] = S0[0][23:1]$ , Nếu  $B[1] = 1$  thì  $S0[1] = S0[0][23:1] + A0$ . Cout[1] là bit nhớ của bộ cộng thứ 0

Tại bộ cộng thứ 2, nếu  $B[2] = 0$  thì  $S0[2] = \{Cout[1], S0[1][23:1]\}$ , Nếu  $B[2] = 1$  thì  $S0[2] = \{Cout[1], S0[1][23:1]\} + A0$

...

Tại bộ cộng thứ 23, Nếu  $B[23] = 0$ ,  $S0[23] = \{Cout[22], S0[22][23:1]\}$  Nếu  $B[23] = 1$  thì  $S0[23] = \{Cout[22], S0[22][23:1]\} + A0$ . Bit Cout[23] là bit nhớ bộ cộng thứ 23

- Làm tròn

Nếu Cout = 0. Kiểm tra bit  $S0[22][0]$ . Nếu bit này là bit 1 thì kết quả  $S1 = S0[23] + 1$ .

Nếu Cout = 1, Kiểm tra bit  $S0[23][0]$ , Nếu bit này là bit 1 thì kết quả  $S1 = S0[23] + 1$ .

Dịch tất cả các bit S1 sang phải 1 bit (Đưa về chuẩn hóa).

Kết quả cuối cùng loại bỏ bit đầu tiên của S1, đưa ra kết quả  $Sout[22:0] = S1[22:0]$ .

Bước 3: Thực hiện tính toán phần mũ (exponent)

Phần mũ của  $Sout[30:23] = A[30:23] + B[30:23] - 127 + Cout[23]$

Bước 4: Kiểm tra underflow, overflow, trường hợp số đặc biệt

- Nếu A hoặc B bằng 0 thì  $S = 0$ , ko cần kiểm tra tiếp.

- Nếu A, B đều khác 0, A hoặc B là oo thì  $S = oo$ .

- Nếu A, B đều ko phải số đặc biệt:

+ Nếu phần mũ  $Sout \geq 255$  thì kết quả là  $S = oo$  (overflow)

+ Nếu phần mũ  $Sout < 0$  thì kết quả là  $S = 0$  (underflow)

+ Nếu ko nằm trong tất cả các trường hợp trên thì  $S[30:0] = Sout[30:0]$ .



### 3. Mô phỏng

	Msgs				
S	32'h4e800000	32'h54e46cc7	32'h56592a16	32'h4e800000	
A	32'h4dfffffe	32'h52123123	32'h4fe06869	32'h4dfffffe	
B	32'h40000001	32'h42480000	32'h45f7bca0	32'h40000001	

Test 1:

Ngõ vào:

-  $A = 32'h52123123 = 156972400640$

-  $B = 32'h42480000 = 50$

Ngõ ra

-  $S = 32'h54e46cc7 = 7848620195840$

Kết quả thực tế  $S = 7848620032000$

S	32'h7f800000	32'h00000000	32'hff800000	32'hb0800001	32'h00000000	32'h7f800000
A	32'h7f000000	32'h00000000	32'hfe23842b	32'h00000001		32'h7f000000
B	32'h60000000	32'h13434234	32'h7f800000	32'hf0000000	32'h3f000000	32'h60000000

Test 2: Nếu A hoặc B = 0 thì kết quả phải là 0

Test 3: Nếu A, B khác 0, A hoặc B = oo thì kết quả là oo

Test 4: Nếu  $A \times B \leq 2^{-127}$  thì kết quả là 0 (underflow)

Test 5: Nếu  $A \times B \geq 2^{128}$  thì kết quả là oo (overflow)

### III. Bộ chia

#### 1. Giải thuật:

Bước 1: Kiểm tra bit dấu hai số hạng.

Bước 2: Kiểm tra các dạng đặc biệt ( infinity, 0 ).

Bước 3: Thực hiện phép trừ hai phần mũ ( exponent ).

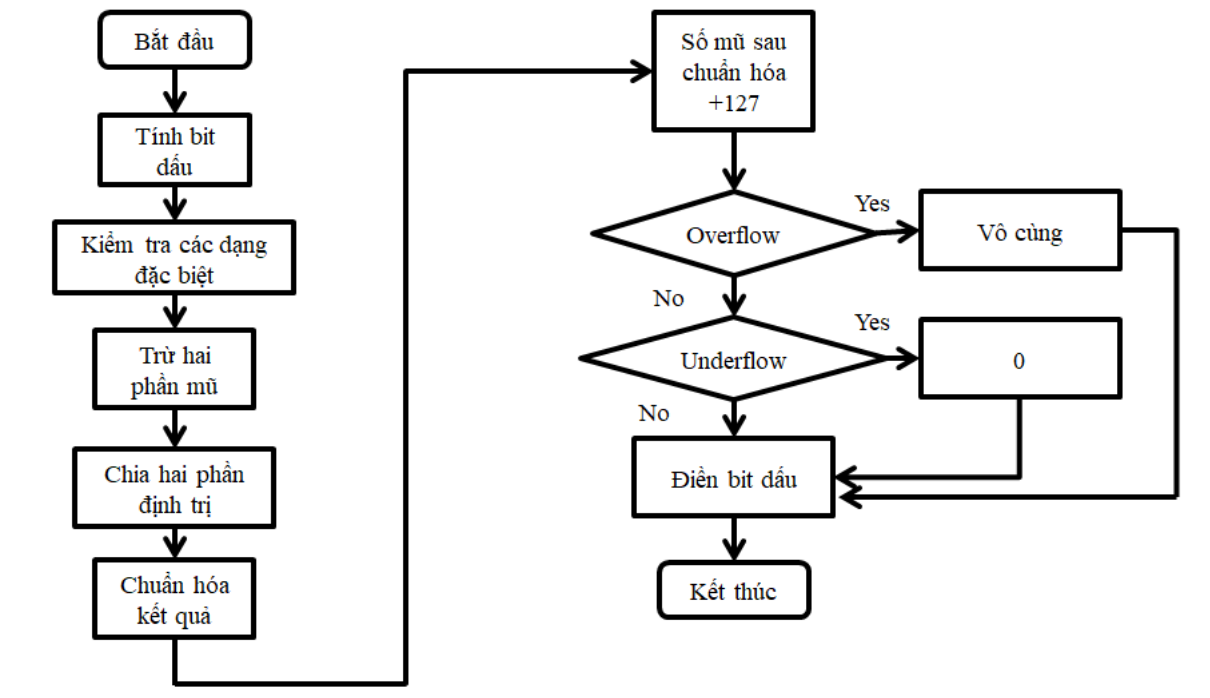
Bước 4: Chia hai phần định trị ( significant ) của hai số hạng.

Bước 5: Chuẩn hoá kết quả.

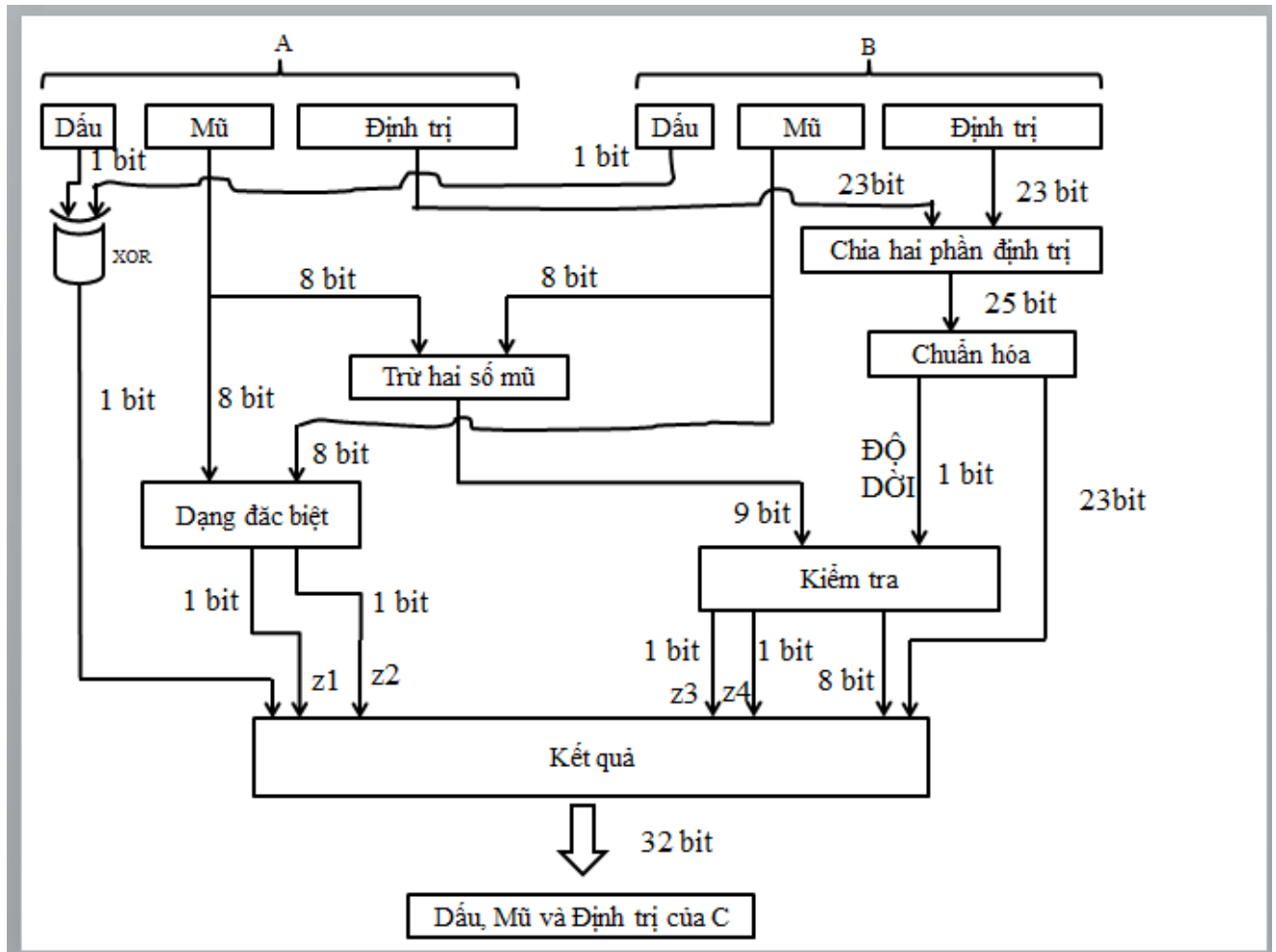
Bước 6: Tính số mũ thật sự và kiểm tra underflow, overflow.

Bước 7: Điền bit dấu, số mũ và phần định trị (kết quả cuối cùng sẽ dựa vào kết quả phân kiểm tra các dạng đặc biệt và phần kiểm tra underflow, overflow).

#### 2. Lưu đồ giải thuật:



#### 3. Sơ đồ phần cứng:



#### 4. Giải thích giải thuật:

##### **Bước 1:**

+ Nếu bit dấu = 0 thì là số dương; = 1 thì là số âm.

A[31]	B[31]	Dấu
0 (dương)	0 (dương)	0 (dương)
0 (dương)	1 (âm)	1 (âm)
1 (âm)	0 (dương)	1 (âm)
1 (âm)	1 (âm)	0 (dương)

=> Dấu = A[31] XOR B[31];

- Đầu ra cổng XOR này sẽ đến khối “KẾT QUẢ” để xử lý.

## **Bước 2:**

- Lấy 8-bit phần mũ của số A, AND lại với nhau cho vào x1; nếu x1 bằng 1 thì 8-bit đều là 1 (A là vô cùng). Tương tự cho số B, gán kết quả tính được vào y1.

- Lấy 8-bit phần mũ của số A, OR lại với nhau; rồi đảo kết quả tính được cho vào x2; nếu x2 bằng 1 thì 8 bit đều là 0 (A là 0). Tương tự cho số B, gán kết quả tính được vào y2.

- Sẽ không xảy ra trường hợp x1 và x2 đồng thời bằng 1, tương tự cho y1 và y2.

- Nếu x1 và x2 đồng thời bằng 0 thì A là số hạng bất kỳ, tương tự cho B.

- Sẽ có 9 trường hợp xảy ra đối với mỗi cặp AB.

- Có 4 loại kết quả là vô cùng, Zero, ERROR và BT(bình thường).

- Gán 2 biến z1, z2 biểu diễn kết quả:

$z1z2 = 00$  (biểu diễn kết quả là Zero)

$z1z2 = 01$  (biểu diễn kết quả là BT)

$z1z2 = 10$  (biểu diễn kết quả là vô cùng)

$z1z2 = 11$  (biểu diễn kết quả là ERROR)

- Ta có bảng sau:

X1	X2	Y1	Y2	Kết quả	Z1	Z2
1	0	1	0	ERROR	1	1
1	0	0	1	Vô cùng	1	0
1	0	0	0	Vô cùng	1	0
0	1	1	0	Zero	0	0

0	1	0	1	ERROR	1	1
0	1	0	0	Zero	0	0
0	0	1	0	Zero	0	0
0	0	0	1	Vô cùng	1	0
0	0	0	0	BT	0	1

- Dùng bìa Karnaugh ta được:

$$z1 = y2 + x1$$

$$z2 = (\text{not } x1)(\text{not } x2)(\text{not } y1)(\text{not } y2) + x1y1 + x2y2$$

- Hai biến  $z1$ ,  $z2$  sẽ đưa đến khối “KẾT QUẢ” để xử lý.

### **Bước 3:**

- Mở rộng 2 số mũ lên 9-bit với bit thứ 9 mặc định bằng 0.

- Lấy 9-bit của A và B đưa vào bộ trừ 9 bit, kết quả 9 bit được đưa đến phần “KIỂM TRA”.

### **Bước 4:**

- Mở rộng phần định trị của A và B lên 25-bit với bit thứ 24 là 1, bit thứ 25 là 0.

- Dùng 1 thanh ghi D chứa thương của phép chia. D sẽ chứa 25-bit vì sau phần chuẩn hóa D sẽ dịch bit và loại bỏ bit.

- Dùng thanh ghi E để chứa kết quả (A-B).

- Sau đó trừ hai số A và B với nhau cho kết quả 25 bit:

+ Nếu bit thứ 25 của E bằng 1 nghĩa là A nhỏ hơn B, thì bit thứ 25 của D là 0; sau đó E sẽ chứa A và dịch trái 1 bit, bit cuối thêm vào sẽ là 0.

+ Còn nếu bit thứ 25 của E bằng 0 nghĩa là A lớn hơn hoặc bằng B thì bit thứ 25 của D là 1; sau đó E sẽ dịch trái 1 bit, bit cuối thêm vào sẽ là 0.

- Lần thứ hai thì dùng thanh ghi E trừ cho B.
- Làm tiếp thêm 23 lần nữa để thanh ghi D có đủ 25 bit.
- Kết quả thanh ghi D sẽ được đưa đến khối “CHUẨN HÓA” để xử lý.

#### **Bước 5:**

- Dùng thanh ghi F chứa 23-bit để chứa giá trị sau chuẩn hóa, các giá trị này sẽ được đưa đến khối “KẾT QUẢ” để xử lý.
- Dùng 1 bit DO\_DOI để tính số lần dịch bit, kết quả này được đưa đến khối “KIỂM TRA”.
- Nếu bit thứ 25 của D bằng 0, thì chắc chắn bit thứ 24 của D sẽ bằng 1.

Thì  $F[22:0] = D[22:0]$ ,  $DO\_DOI = 1$ .

- Nếu bit thứ 25 của D bằng 1

Thì  $F[22:0] = D[23:1]$ ,  $DO\_DOI = 0$ .

#### **Bước 6:**

- Lấy kết quả trừ hai số mũ ở **Bước 3** trừ đi  $DO\_DOI$  ở **Bước 5**, ta được phần mũ sau chuẩn hóa. Nếu phần mũ sau chuẩn hóa  $> 127$  thì là overflow bit  $z3 = 1$ ; nếu phần mũ sau chuẩn hóa  $< -126$  thì là underflow bit  $z4 = 1$ .
- Lấy phần mũ sau chuẩn hóa cộng với 127 rồi đưa đến khối “KẾT QUẢ”. Kết quả này gần như sẽ là phần mũ thật sự của C nếu không bị overflow, underflow hoặc trùng với các trường hợp đặc biệt ở **Bước 2**.

#### **Bước 7:**

- Nếu  $z3 = 1$  thì kết quả chia A và B là vô cùng.
- Nếu  $z4 = 1$  thì kết quả chia là Zero.

- Hai biến z1, z2 cũng là đầu vào của khối “KẾT QUẢ” để quyết định kết quả cuối cùng của thanh ghi C.
- Nếu kết quả phép chia là Zero thì C có dạng x000 0000:
  - + Nếu bit Dấu âm thì là 1000 0000, còn bit Dấu dương thì là 0000 0000.
- Nếu kết quả phép chia là Vô cùng thì C có dạng xf80 0000:
  - + Nếu bit Dấu âm thì là ff80 0000, còn bit Dấu dương thì là 7f80 0000.
- Nếu kết quả phép chia là ERROR thì C là 1 giá trị cố định không phụ thuộc bit dấu, C là ffff ffff.
- Nếu kết quả phép chia là BT thì:
  - + Dấu của C sẽ là bit Dấu tính ở **Bước 1**.
  - + Mũ của C là phần mũ ngỏ ra của khối “KIỂM TRA” ở **Bước 6**.
  - + Định trị của C là phần ngỏ ra của khối “CHUẨN HÓA” ở **Bước 5**.

## 5. Mô phỏng:

1. a=32'h7f800002; // +  $\infty$

b=32'h00000003; // + 0

=> kết quả tính toán là +  $\infty$ : 7f80 0000 (do a và b khác dấu)

2. a=32'h7f800002; // +  $\infty$

b=32'h7f800003; // +  $\infty$

=> kết quả tính toán là ERROR: ffff ffff




(đặt giá trị mặc định cho ERROR là 32 bit đều là 1, bất kể bit dấu).

3. a=32'h7f800002; // +  $\infty$

b=32'h3f000000; // + 0.5

=> kết quả tính toán là  $+\infty$ : 7f80 0000 (do a và b khác dấu)

- Kết quả mô phỏng cho 1, 2, 3:

+  /tb_final/a	32'h05100000	7f800002		
+  /tb_final/b	32'h44500000	00000003	7f800003	3f000000
+  /tb_final/c	32'h00000000	7f800000	ffffff	7f800000

4. a=32'h00000004; // + 0

b=32'h00000003; // + 0

=> kết quả phép toán là ERROR: ffff ffff

5. a=32'h00000005; // + 0

b=32'h7f800003; //  $+\infty$




=> kết quả phép toán là + 0: 0000 0000

6. a=32'h00000006; // + 0

b=32'h3f000000; // + 0.5

=> kết quả phép toán là + 0: 0000 0000

- Kết quả mô phỏng 4, 5, 6:

+  /tb_final/a	32'h05100000	00000004	00000005	00000006
+  /tb_final/b	32'h44500000	00000003	7f800003	3f000000
+  /tb_final/c	32'h00000000	ffffff	00000000	

7. a=32'h3f000000; // + 0.5

b=32'h00000003; // + 0



=> kết quả phép toán là  $+\infty$ : 7f80 0000

8. `a=32'h3f000000; // + 0.5`

`b=32'h7f800003; // +  $\infty$`

=> kết quả phép toán là  $+0$ : 0000 0000

9. `a=32'h3f000000; // + 0.5`

`b=32'hbee00000; // - 0.4375`

=> kết quả phép toán là -1.14285714 : bf92 4925

10. `a=32'h40000000; // + 2`

`b=32'h40000000; // + 2`

=> kết quả phép toán là  $+1$ : 3f80 0000

- Kết quả mô phỏng 7, 8, 9, 10:

		Page			
+	/tb_final/a	32'h05100000	3f000000		40000000
+	/tb_final/b	32'h44500000	00000003	7f800003	bee00000
+	/tb_final/c	32'h00000000	7f800000	00000000	bf924924
					3f800000

11. Trường hợp kết quả phần mũ là overflow

`a=32'h45500000; // mũ 138 , đã cộng +127, phần định trị của a không nhỏ hơn b.`

`b=32'h05500000; // mũ 10 , đã cộng +127`

=> Hiệu 2 số mũ:  $128 > 127$

=> Kết quả là vô cùng. Do trong ví dụ a và b đều dương nên là  $+\infty$ : 7f80 0000

12. Trường hợp kết quả phần mũ là overflow nhưng sau khi chuẩn hóa thì không còn overflow nữa:

`a=32'h45000000; // mũ 138 , đã cộng +127, phần định trị của a nhỏ hơn b.`

b=32'h05400000; // mũ 10 , đã cộng +127.

=> Hiệu 2 số mũ là 128. Sau chuẩn hóa là 127 nên không còn overflow nữa.

Kết quả mô phỏng 11, 12:

	Msgs		
+ /tb_final/a	32'h05100000	45500000	45000000
+ /tb_final/b	32'h44500000	05500000	05400000
+ /tb_final/c	32'h00000000	7f800000	7f2aaaaa

13. Trường hợp kết quả phần mũ là underflow:

a=32'h05500000; // mũ 10 , đã cộng +127, phần định trị của a không nhỏ hơn b

b=32'h44d00000; // mũ 137 , đã cộng +127

=> Hiệu 2 số mũ  $-127 < -126$  nên là underflow

=> Kết quả là + 0: 0000 0000 (do a và b đều dương)

14. Trường hợp kết quả phần mũ không là underflow nhưng sau khi chuẩn hóa thì là underflow

a=32'h05100000; // mũ 10 , đã cộng +127, phần định trị của a nhỏ hơn b

b=32'h44500000; // mũ 136 , đã cộng +127

=> Hiệu 2 số mũ -126. Nhưng sau chuẩn hóa là  $-127 < -126$  nên là underflow

=> Kết quả là + 0: 0000 0000 (do a và b đều dương)

	Msgs		
+ /tb_final/a	32'h05100000	05500000	05100000
+ /tb_final/b	32'h44500000	44d00000	44500000
+ /tb_final/c	32'h00000000	00000000	

## **PHẦN II: BONUS.**

### **I. Thực hiện phép lấy căn bậc n của A. Với n ngõ vào là 1 số 8 bit, A là số 32 bit dấu chấm động.**

Sử dụng công thức hồi quy  $S = (S / A^{(n-1)} + S(n-1)) / n$  với S ban đầu được lựa chọn bất kì khác 0 .

Ngõ vào:

- A là số 32 bit
- n là số 8 bit
- clk là xung clock hệ thống

Ngõ ra:

- Error (1 bit) = 1 nếu  $n = 0$
- i (1 bit) = 1 nếu kết quả là số ảo
- S (32 bit) là kết quả căn bậc n của A

#### **1. Giải thuật**

Bước 1: Kiểm tra các trường hợp đặc biệt

Nếu A là 0 hoặc oo thì kết quả là  $S = A$ .

Nếu  $n = 0$  thì kết quả là  $S = oo$

Kiểm tra bit cuối của n và dấu của A. Nếu  $n[0] = 1$  (n là số lẻ) và S là số âm thì bật bit i (bit ảo) rồi chuyển A thành số dương.

Bước 2: Chuyển n , n - 1 thành số 32 bit dấu chấm động

Bước 3: Chọn S ban đầu (Mặc định là 1). Nếu S ban đầu làm cho  $S / A^{(n-1)}$

bị underflow thì phải chọn lại S (Ở đây sẽ lựa chọn S ban đầu lần lượt là căn 2, 2 căn 2, 4 căn 2, 8 căn 2 ,...)

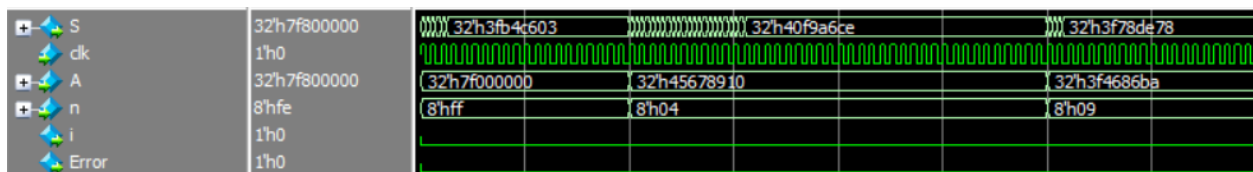
Bước 4: Phối hợp các bộ +,-,x,/ 32 bit dấu chấm động để tính

$$S_{out} = (S / A^{(n-1)} + S^{(n-1)}) / n$$

Bước 5: Sử dụng bộ flip flop cập nhật giá trị S kế tiếp  $S = S_{out}$

Bước 6: Quay về bước 4. Nếu  $S / A^{(n-1)}$  bị underflow hoặc ngõ vào thay đổi thì quay lại bước 3, Chọn lại S ban đầu

## 2. Mô phỏng



- Test 1:

Ngõ vào:

$$A = 32'h45678910 = 3704.56640625$$

$$n = 8'h04 = 4$$

Ngõ ra :

$$S = 40f9a6ce = 7.8016119$$

Kết quả thực tế 7.801611972

- Test 2:

Ngõ vào:

$$A = 32'h7f000000 = 2^{127}$$

$$n = 8'hff = 255$$

Ngõ ra:

$$S = 32'h3fb054f4 = 1.4122928381$$

$$\text{Thực tế } S = 1.412292793$$

- Test 3:

Ngõ vào:

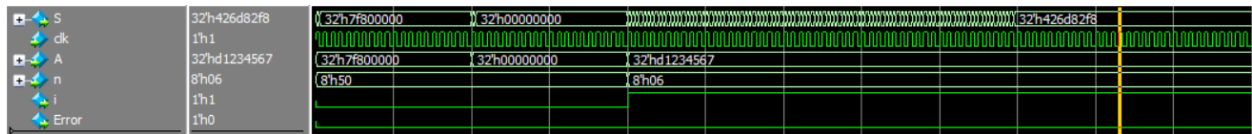
$$A = 32'h3f4686ba = 0.775493264198$$

$$n = 8'h09$$

Ngõ ra:

$$S = 3f78de78 = 0.972144603729$$

$$\text{Thực tế } S = 0.9721446539$$



- Test 4:

Ngõ vào:

$$A = 0 \text{ hoặc } oo$$

$$n = xx$$

Ngõ ra:

$$S = A;$$

- Test 5:

Ngõ vào:

$$A = 32'hxxxxxxxx;$$

$n = 0;$

Ngõ ra :

$S = 32'hxxxxxxxx$

$Error = 1;$

- Test 6:

Ngõ vào:

$A = 32'hd1234567 = -43827752960$

$n = 8'h06 = 6$

Ngõ ra :

$S = 32'h426d62f8 = 59.3778991699$

$i = 1;$

Kết quả thực tế  $S = 59,37790458i$

## **II. Chuyển ngõ vào ngõ ra thành dạng thập phân tương ứng**

Sử dụng thuật toán double dabble để chuyển đổi số nhị phân thành ký hiệu thập phân được mã hóa nhị phân (BCD). Nó còn được gọi là thuật toán shift-and-add- 3

Ngõ vào:

- A là số 32-bit dấu chấm động

- clk là xung clock hệ thống

Ngõ ra:

- Dau (1 bit).  $Dau = 1$  thì kết quả là số âm,  $Dau = 0$  thì kết quả là số âm

- S (32 bit) biểu diễn 8 số thập phân được mã hóa nhị phân (BCD)

-  $D_{sm10}$  (1 bit)

-  $sm10$  số mũ 10 của số nhị phân

### 1. Giải thuật:

Bước 1:

Ta cần lấy 8 số làm kết quả nên ta sẽ nhân A với 10 cho đến khi lớn hơn  $10^8$ .  $A0 = A \times 10^{sm} > 10^8$ .

Bước 2: S ban đầu gán là 0

Bước 3: Sử dụng thuật toán double dabble với số lần dịch chính là  $A0[30:23]$

- Kiểm tra  $S[31]$

+ Nếu  $S[31] = 0$  thì dịch tất cả các bit S sang trái 1 bit. Bit cuối cùng của S là bit  $A[23]$ .

+ Nếu  $S[31] = 1$  thì dịch tất cả các bit S sang phải 3 bit.  $sm1 = sm1 + 1$ .  $sm1$  ban đầu mặc định là 7 vì  $A0$  luôn là số  $\geq 10^8$

- Dịch tất cả các bit A sang trái 1 bit.

- Kiểm tra các nhóm 4 bit của S là  $[31:28], [27:24], \dots, [3:0]$ . Nếu nhóm 4 bit đó lớn hơn hoặc bằng 5 thì cộng thêm 3. Nếu không thì giữ nguyên

- Lặp lại các bước trên mỗi lần có xung clock với  $A0[30:23]$  lần

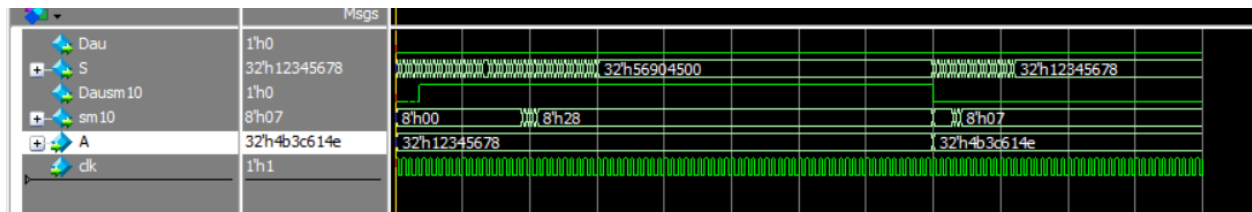
Bước 4: Tính số mũ  $10.sm10 = sm1 - sm$ .

Nếu  $sm10 < 0$  thì  $D_{sm10} = 1$ , nghịch đảo tất cả các bit  $sm10$  và cộng 1. (Bù 2)

Nếu  $sm10 \geq 0$  thì  $D_{sm10} = 0$ ;

Chuyển  $sm10$  sang dạng BCD.

## 2. Mô phỏng



Test 1:

Ngõ vào:  $A = 32'h12345678$  cùng với xung clock hệ thống

Ngõ ra:

- $Dau = 0$ ;
- $S = 32'h56904500$ ;
- $Dausm10 = 1$ ;
- $sm10 = 8'h28$ ;

Tức kết quả ngõ ra là  $5,6904500 \times 10^{-28}$

Kết quả thực tế  $5.6904566139 \times 10^{-28}$

Test 2:

Ngõ vào:  $A = 32'h4b3c614e$  cùng với xung clock hệ thống

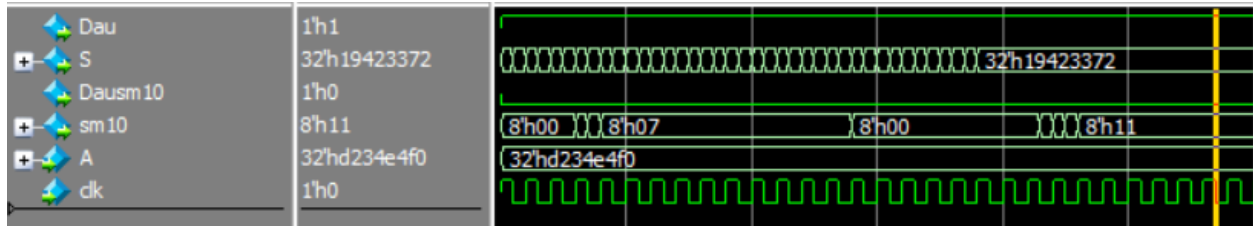
Ngõ ra:

- $Dau = 0$ ;
- $S = 32'h12345678$
- $Dausm10 = 0$ ;
- $sm10 = 8'h07$ .



Tức kết quả ngõ ra là  $1,2345678 \times 10^7$

Kết quả thực tế là  $1,2345678 \times 10^7$ .



Ngõ vào:  $A = 32'h\ d234e4f0$  cùng với xung clock hệ thống

Ngõ ra:

- $Dau = 1$ ;
- $S = 32'h19423372$ ;
- $Dausm10 = 0$ ;
- $sm10 = 8'h11$ ;

Tức kết quả ngõ ra là  $-1,9423372 \times 10^{11}$

Kết quả thực tế  $-1.94233761792 \times 10^{11}$

Qua nhiều bài test, độ chính xác khoảng 4,5 chữ số sau dấu phẩy . Có thể tăng độ chính xác bằng cách tăng lưu trữ của biến S nhiều hơn 32 bit

### **PHẦN III: CÔNG VIỆC TỪNG THÀNH VIÊN**

Họ và tên	Công việc
Nguyễn Trọng Tín	Bộ cộng trừ
Dương Nhật Huy	Bộ nhân và Bonus
Chung Hữu Nhân	Bộ chia