ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

Khoa Điện – Điện tử Bộ môn Điện tử

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



BÁO CÁO XÂY DỰNG

BỘ CHIA HAI SỐ FLOATING POINT

LAB 3

**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**

**I/** Mục tiêu

Thiết kế bộ nhân hai số floating point

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

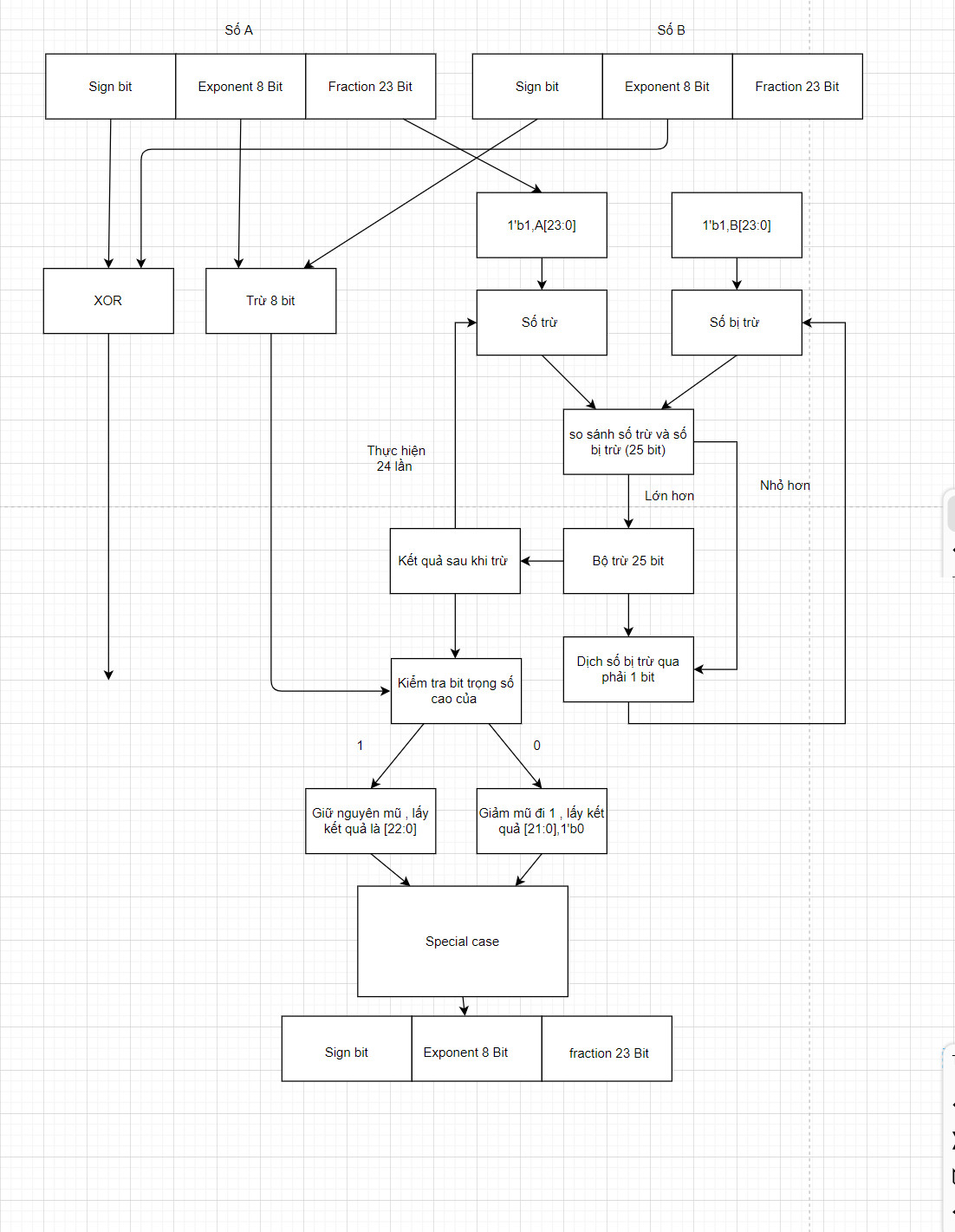
-Nếu là 1 thì phần faction của kết quả là từ bit [21:0] của biến kq và thêm 1 bit 0 ở bit trong số thấp giữ nguyên phần mũ

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

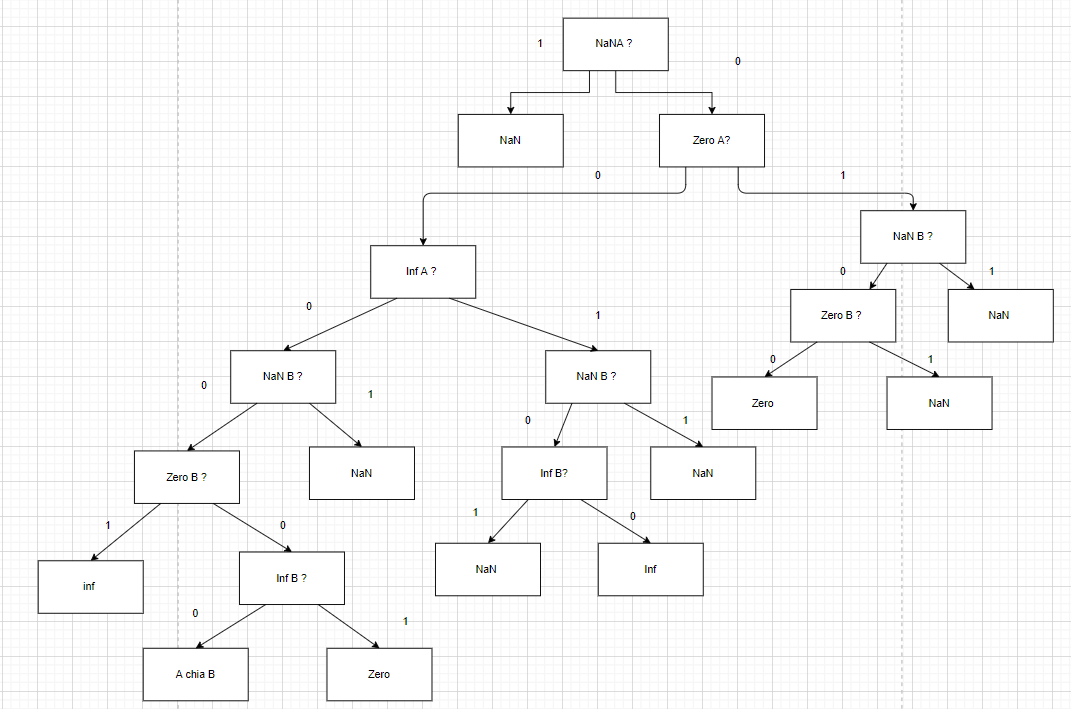
Nếu A là Inf B là Inf thì kết quả là NaN

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

II.CODE DESIGN – CODE TESTBENCH

1. Code trừ 1 bit

module Tru1bit(a,b,cin,s,cout);

input a,b,cin;

output s,cout;

wire A\_xor\_B;

wire not\_a, B\_or\_Cin,c1,c2;

/\* tinh s \*/

//xor

assign A\_xor\_B=a^b;

//xor

assign s=cin^A\_xor\_B;

/\* tinh cout \*/

//not

assign not\_a=!a;

//or

assign B\_or\_Cin=b|cin;

//and

assign c1=not\_a & B\_or\_Cin;

//and

assign c2=b & cin ;

//or

assign cout = c1|c2;

endmodule

module t\_Tru1bit;

reg a,b,cin;

wire s,cout;

Tru1bit easd(a,b,cin,s,cout);

initial begin

#0

a=0;b=0;cin=0;

#10

a=0;b=0;cin=1;

#10

a=0;b=1;cin=0;

#10

a=0;b=1;cin=1;

#10

a=1;b=0;cin=0;

#10

a=1;b=0;cin=1;

#10

a=1;b=1;cin=0;

#10

a=1;b=1;cin=1;

#10 ;

end

Endmodule

1. Code trừ 9 bit

module Div\_tru8bit(in1,in2,sub,cout); // in1 - in2 = {cout,sub}

input [8:0]in1,in2;

output [9:0]sub;

output cout;

wire [8:0] c;

Tru1bit T[8:0] (.a (in1[8:0]), .b (in2[8:0]), .cin ({c[7:0],1'b0}), .cout (c[8:0]), .s (sub[8:0]));

assign cout = c[8];

assign sub[9]=cout;

endmodule

1. Code cong 9 bit

module Div\_cong9bit(in1,in2,sum,cout);

input [9:0] in1;

input [9:0] in2;

output [9:0] sum;

output cout;

wire [9:0] c;

FA FA[9:0] (.in1 (in1[9:0]), .in2 (in2[9:0]), .cin ({c[8:0],1'b0}), .cout (c[9:0]), .sum (sum[9:0]));

assign cout = c[9];

endmodule

/\*\*/

1. Code trừ phần mũ

module Div\_TruMu(muA,muB,out);

input [7:0]muA,muB;

output [9:0]out;

wire [9:0]sub;

wire [8:0]muAA,muBB;

assign muAA[8:0]={1'b0,muA[7:0]};

assign muBB[8:0]={1'b0,muB[7:0]};

Div\_tru8bit a(muAA[8:0],muBB[8:0],sub[9:0],/\*\*/);

Div\_cong9bit b(sub[9:0],10'b0001111111,out[9:0],/\*\*/);

endmodule

module FA(in1,in2,cin,sum,cout); //cong1bit

input wire in1;

input wire in2;

input wire cin;

output wire sum;

output wire cout;

wire temp1, temp2, temp3;

xor(sum,in1,in2,cin);

and(temp1,in1,in2);

and(temp2,in1,cin);

and(temp3,in2,cin);

or(cout,temp1,temp2,temp3);

endmodule

module t\_FA;

reg in1,in2,cin;

wire sum,cout;

FA easd(in1,in2,cin,sum,cout);

initial begin

#0

in1=0;in2=0;cin=0;

#50

in1=0;in2=0;cin=1;

#50

in1=0;in2=1;cin=0;

#50

in1=0;in2=1;cin=1;

#50

in1=1;in2=0;cin=0;

#50

in1=1;in2=0;cin=1;

#50

in1=1;in2=1;cin=0;

#50

in1=1;in2=1;cin=1;

#50 ;

end

endmodule

/\*\*/

module t\_Div\_TruMu;

reg [7:0]muA,muB;

wire [9:0]out;

Div\_TruMu z(muA[7:0],muB[7:0],out[9:0]);

initial begin

#0

muA=8'b11111110;

muB=8'b11111100;

#100

muA=8'b00000010;

muB=8'b11111110;

#100;

end

endmodule

1. Code trừ 10 bit

module tru10bit(in1,in2,sub,cout); // in1 - in2 = {cout,sub}

input [9:0]in1,in2;

output [9:0]sub;

output cout;

wire [9:0] c;

Tru1bit T[9:0] (.a (in1[9:0]), .b (in2[9:0]), .cin ({c[8:0],1'b0}), .cout (c[9:0]), .s (sub[9:0]));

assign cout = c[9];

endmodule

module t\_tru10bit;

reg [9:0]in1,in2;

wire [9:0]sub;

wire cout;

tru10bit t8b\_1(in1,in2,sub,cout);

initial begin

#0

in1 = 9'b0000\_1111;

in2 = 9'b0000\_1100;

#100

in1 = 9'b1000\_0000;

in2 = 9'b1000\_0001;

#100;

end

endmodule

1. Code chia 25 bit:

module chia\_25bit (a, b, ketqua);

input [23:0] a,b;

output [23:0] ketqua;

wire [23:0]C;

wire [24:0] ab1, ab2, ab3, ab4, ab5, ab6, ab7, ab8, ab9, ab10, ab11, ab12, ab13, ab14, ab15, ab16, ab17, ab18, ab19, ab20, ab21, ab22, ab23;

wire [24:0] frac[23:0];

sosanhfrac sosanh1 (.a({1'b0,a}), .b({1'b0,b}), .C1(C[23]));

assign frac[1]=C[23]?{1'b0,b}:25'b0;

tru\_25bit tru0 (.a({1'b0,a}),.b(frac[1]),.ab(ab1));

sosanhfrac sosanh2 (.a({ab1[23:0],1'b0}), .b({1'b0,b}), .C1(C[22]));

assign frac[2]=C[22]?{1'b0,b}:25'b0;

tru\_25bit tru1 (.a({ab1[23:0],1'b0}),.b(frac[2]),.ab(ab2) );

sosanhfrac sosanh3 (.a({ab2[23:0],1'b0}), .b({1'b0,b}), .C1(C[21]));

assign frac[3]=C[21]?{1'b0,b}:25'b0;

tru\_25bit tru2 (.a({ab2[23:0],1'b0}),.b(frac[3]),.ab(ab3));

sosanhfrac sosanh4 (.a({ab3[23:0],1'b0}), .b({1'b0,b}), .C1(C[20]));

assign frac[4]=C[20]?{1'b0,b}:25'b0;

tru\_25bit tru3 (.a({ab3[23:0],1'b0}),.b(frac[4]),.ab(ab4));

sosanhfrac sosanh5 (.a({ab4[23:0],1'b0}), .b({1'b0,b}), .C1(C[19]));

assign frac[5]=C[19]?{1'b0,b}:25'b0;

tru\_25bit tru4 (.a({ab4[23:0],1'b0}),.b(frac[5]),.ab(ab5));

sosanhfrac sosanh6 (.a({ab5[23:0],1'b0}), .b({1'b0,b}), .C1(C[18]));

assign frac[6]=C[18]?{1'b0,b}:25'b0;

tru\_25bit tru5 (.a({ab5[23:0],1'b0}),.b(frac[6]),.ab(ab6));

sosanhfrac sosanh7 (.a({ab6[23:0],1'b0}), .b({1'b0,b}), .C1(C[17]));

assign frac[7]=C[17]?{1'b0,b}:25'b0;

tru\_25bit tru6 (.a({ab6[23:0],1'b0}),.b(frac[7]),.ab(ab7));

sosanhfrac sosanh8 (.a({ab7[23:0],1'b0}), .b({1'b0,b}), .C1(C[16]));

assign frac[8]=C[16]?{1'b0,b}:25'b0;

tru\_25bit tru7 (.a({ab7[23:0],1'b0}),.b(frac[8]),.ab(ab8));

sosanhfrac sosanh9 (.a({ab8[23:0],1'b0}), .b({1'b0,b}), .C1(C[15]));

assign frac[9]=C[15]?{1'b0,b}:25'b0;

tru\_25bit tru8 (.a({ab8[23:0],1'b0}),.b(frac[9]),.ab(ab9));

sosanhfrac sosanh10 (.a({ab9[23:0],1'b0}), .b({1'b0,b}), .C1(C[14]));

assign frac[10]=C[14]?{1'b0,b}:25'b0;

tru\_25bit tru9 (.a({ab9[23:0],1'b0}),.b(frac[10]),.ab(ab10));

sosanhfrac sosanh11 (.a({ab10[23:0],1'b0}), .b({1'b0,b}), .C1(C[13]));

assign frac[11]=C[13]?{1'b0,b}:25'b0;

tru\_25bit tru10 (.a({ab10[23:0],1'b0}),.b(frac[11]),.ab(ab11));

sosanhfrac sosanh12 (.a({ab11[23:0],1'b0}), .b({1'b0,b}), .C1(C[12]));

assign frac[12]=C[12]?{1'b0,b}:25'b0;

tru\_25bit tru11 (.a({ab11[23:0],1'b0}),.b(frac[12]),.ab(ab12));

sosanhfrac sosanh13 (.a({ab12[23:0],1'b0}), .b({1'b0,b}), .C1(C[11]));

assign frac[13]=C[11]?{1'b0,b}:25'b0;

tru\_25bit tru12 (.a({ab12[23:0],1'b0}),.b(frac[13]),.ab(ab13));

sosanhfrac sosanh14 (.a({ab13[23:0],1'b0}), .b({1'b0,b}), .C1(C[10]));

assign frac[14]=C[10]?{1'b0,b}:25'b0;

tru\_25bit tru13 (.a({ab13[23:0],1'b0}),.b(frac[14]),.ab(ab14));

sosanhfrac sosanh15 (.a({ab14[23:0],1'b0}), .b({1'b0,b}), .C1(C[9]));

assign frac[15]=C[9]?{1'b0,b}:25'b0;

tru\_25bit tru14 (.a({ab14[23:0],1'b0}),.b(frac[15]),.ab(ab15));

sosanhfrac sosanh16 (.a({ab15[23:0],1'b0}), .b({1'b0,b}), .C1(C[8]));

assign frac[16]=C[8]?{1'b0,b}:25'b0;

tru\_25bit tru15 (.a({ab15[23:0],1'b0}),.b(frac[16]),.ab(ab16));

sosanhfrac sosanh17 (.a({ab16[23:0],1'b0}), .b({1'b0,b}), .C1(C[7]));

assign frac[17]=C[7]?{1'b0,b}:25'b0;

tru\_25bit tru16 (.a({ab16[23:0],1'b0}),.b(frac[17]),.ab(ab17));

sosanhfrac sosanh18 (.a({ab17[23:0],1'b0}), .b({1'b0,b}), .C1(C[6]));

assign frac[18]=C[6]?{1'b0,b}:25'b0;

tru\_25bit tru17 (.a({ab17[23:0],1'b0}),.b(frac[18]),.ab(ab18));

sosanhfrac sosanh19 (.a({ab18[23:0],1'b0}), .b({1'b0,b}), .C1(C[5]));

assign frac[19]=C[5]?{1'b0,b}:25'b0;

tru\_25bit tru18 (.a({ab18[23:0],1'b0}),.b(frac[19]),.ab(ab19));

sosanhfrac sosanh20 (.a({ab19[23:0],1'b0}), .b({1'b0,b}), .C1(C[4]));

assign frac[20]=C[4]?{1'b0,b}:25'b0;

tru\_25bit tru19 (.a({ab19[23:0],1'b0}),.b(frac[20]),.ab(ab20));

sosanhfrac sosanh21 (.a({ab20[23:0],1'b0}), .b({1'b0,b}), .C1(C[3]));

assign frac[21]=C[3]?{1'b0,b}:25'b0;

tru\_25bit tru20 (.a({ab20[23:0],1'b0}),.b(frac[21]),.ab(ab21));

sosanhfrac sosanh22 (.a({ab21[23:0],1'b0}), .b({1'b0,b}), .C1(C[2]));

assign frac[22]=C[2]?{1'b0,b}:25'b0;

tru\_25bit tru21 (.a({ab21[23:0],1'b0}),.b(frac[22]),.ab(ab22));

sosanhfrac sosanh23 (.a({ab22[23:0],1'b0}), .b({1'b0,b}), .C1(C[1]));

assign frac[23]=C[1]?{1'b0,b}:25'b0;

tru\_25bit tru22 (.a({ab22[23:0],1'b0}),.b(frac[23]),.ab(ab23));

sosanhfrac sosanh24 (.a({ab23[23:0],1'b0}), .b({1'b0,b}), .C1(C[0]));

assign ketqua = {C[23],C[22],C[21],C[20],C[19],C[18],C[17],C[16],C[15],C[14],C[13],C[12],C[11],C[10],C[9],C[8],C[7],C[6],C[5],C[4],C[3],C[2],C[1],C[0]};

endmodule

module tb\_chia\_25bit ();

reg [23:0] a,b;

wire [23:0] ketqua;

chia\_25bit chia (.a(a), .b(b), .ketqua(ketqua));

initial begin

#0;

a = 24'b1000\_0000\_0001\_0000\_0001\_1101 ; b = 24'b1000\_0000\_0000\_0100\_1011\_0000;

#10;

a = 24'b1011\_0100\_1001\_0000\_0110\_0000 ; b = 24'b1011\_0100\_1001\_0000\_0110\_0000;

#10;

a = 24'b1001\_0100\_0001\_0010\_1010\_1010 ; b = 24'b1100\_1001\_0111\_1000\_1101\_0100;

#10;

a = 24'b1111\_0100\_1001\_0010\_1110\_0000 ; b = 24'b1100\_1001\_0100\_1000\_0010\_0000;

#10;

a = 24'b1011\_0100\_1001\_0000\_0110\_0000 ; b = 24'b1101\_1001\_0100\_0011\_1111\_0000;

#10;

end

endmodule

1. Code mux 2 sang 1

module mux2to1kq(s,in1,in2,out); //s=0 out=in1 // s=1 out=in2

input wire s;

input wire [31:0]in1,in2;

output wire [31:0]out;

assign out = (s)? in2 : in1;

endmodule

module t\_mux2to1kq;

reg s;

reg [31:0]in1,in2;

wire [31:0]out;

mux2to1kq m1(s,in1[31:0],in2[31:0],out[31:0]);

initial begin

#0

s=0;

in1=32'b0\_10000110\_01101010101010101010101;

in2=32'b0\_10000010\_00110011001100110011001;

#100

in1=32'b0\_10000110\_00000000101000111101011;

in2=32'b0\_10000001\_01010101010101010101010;

#100

s=1;

in1=32'b0\_10000110\_01001100110011001100110;

in2=32'b0\_10000001\_10101010101010101010101;

#100

in1=32'b0\_10000110\_01101010101010101010101;

in2=32'b0\_10000010\_11001100110011001100110;

#100;

end

endmodule

1. Code special case

module SpecCase(in,inf,NaN,zero,bt); // chi xet duong // ko xet dau

input wire [31:0]in;

output wire inf,NaN,zero,bt;

wire [7:0]mu;

wire [22:0]frac;

assign mu[7:0]=in[30:23];

assign frac[22:0]=in[22:0];

assign zero=(mu[7:0]==8'b0000\_0000) ? 1'b1 : 1'b0;

assign inf=(mu[7:0]==8'b1111\_1111) ? ((frac[22:0]==23'b0)? 1'b1 : 1'b0) : 1'b0;

assign NaN=(mu[7:0]==8'b1111\_1111) ? ((frac[22:0]!=23'b0)? 1'b1 : 1'b0) : 1'b0;

assign bt= ~zero && ~inf && ~NaN;

endmodule

module t\_SpecCase;

reg [31:0]in;

wire inf,NaN,zero;

SpecCase s1(in,inf,NaN,zero,bt);

initial begin

#0 in = 32'b0\_00000000\_00000000000000000000000; // zero

#100 in = 32'b0\_00000000\_11110000000000000000000; // zero

#100 in = 32'b1\_00000000\_11110000000000000000000; // zero

#100 in = 32'b0\_11111111\_00000000000000000000000; // inf

#100 in = 32'b1\_11111111\_00011000000000000000000; // NaN

#100 in = 32'b0\_11111111\_11111111111111111111111; // NaN

#100 in = 32'b0\_11111111\_11111110000000000001111; // NaN

#100 in = 32'b0\_11000001\_11111111111111111111111; // bt

#100 in = 32'b1\_00000000\_00000000000000000000000; // -zero

#100 in = 32'b1\_11111111\_00000000000000000000000; // -inf

#100 in = 32'b1\_11111111\_11111111111111111111111; // NaN

#100 in = 32'b1\_11111111\_11111110000000000001111; // NaN

#100 in = 32'b1\_11000001\_11111111111111111111111; // -bt

#100;

end

endmodule

1. Code bộ nhân

module achiab(A,B,KQ); // Chi Xet Duong

input [31:0]A,B;

output [31:0]KQ;

wire [31:0]Inf,Zero,NaN;

wire [9:0]ex,ex1;

wire [22:0]tpAB;

wire [23:0]Thuong;

wire [31:0]p1,kqAB;

wire [7:0]exAB;

wire [23:0] A1,B1;

wire Sign;

wire [31:0]o1,o2,o3,o4,o5,o6,o7,o8,o9,KQ\_NoSign,achiab;

/\*\*/

assign Inf=32'b01111111100000000000000000000000; //inf

assign Zero=32'b00000000000000000000000000000000;

assign NaN=32'b01111111111111111111111111111111;

/\*Tru Mu\*/

Div\_TruMu a1(A[30:23],B[30:23],ex[9:0]); // mu a tru mu b cong 127

/\*Giam Mu Di 1\*/

tru10bit a2(ex[9:0],10'b1,ex1[9:0]);

/\*Chia Thap Phan\*/

assign A1[23:0]={1'b1,A[22:0]};

assign B1[23:0]={1'b1,B[22:0]};

chia\_25bit a3(A1[23:0],B1[23:0],Thuong[23:0]);

/\*Dich Thap Phan\*/

mux2to1kq a4(Thuong[23],{Thuong[21:0],1'b0},Thuong[22:0],tpAB[22:0]);

/\*Xac Dinh Mu\*/

mux2to1kq a5(Thuong[23],ex1[7:0],ex[7:0],exAB[7:0]);

/\*A Chia B\*/

assign kqAB[31:0]={1'b0,exAB[7:0],tpAB[22:0]};

/\*ex[9]=1 Zero | ex[8]=1 Inf | aChiab\*/

mux2to1kq a6(ex[8],kqAB[31:0],Inf[31:0],p1[31:0]);

mux2to1kq a7(ex[9],p1[31:0],Zero[31:0],achiab[31:0]);

/\*\*/

SpecCase s1(A[31:0],infA,nanA,zeroA,btA);

SpecCase s2(B[31:0],infB,nanB,zeroB,btB);

xor b1(Sign,A[31],B[31]);

/\*Truong Hop Dac Biet\*/

mux2to1kq m1(zeroB,Zero[31:0],NaN[31:0],o1[31:0]);

mux2to1kq m2(nanB,o1[31:0],NaN[31:0],o2[31:0]);

mux2to1kq m3(infB,Inf[31:0],NaN[31:0],o3[31:0]);

mux2to1kq m4(nanB,o3[31:0],NaN[31:0],o4[31:0]);

mux2to1kq m5(infB,achiab[31:0],Zero[31:0],o5[31:0]);

mux2to1kq m6(zeroB,o5[31:0],Inf[31:0],o6[31:0]);

mux2to1kq m7(nanB,o6[31:0],NaN[31:0],o7[31:0]);

mux2to1kq m8(infA,o7[31:0],o4[31:0],o8[31:0]);

mux2to1kq m9(zeroA,o8[31:0],o2[31:0],o9[31:0]);

mux2to1kq m0(nanA,o9[31:0],NaN[31:0],KQ\_NoSign[31:0]);

assign KQ[31:0]={Sign,KQ\_NoSign[30:0]};

endmodule

/\*\*/

module t\_achiab;

reg [31:0]A,B;

wire [31:0]achiab;

achiab az(A,B,achiab);

initial begin

#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

#10

A=32'b0\_01111111\_01001100110011001100110;

B=32'b0\_01111111\_01001100110011001100110;

#10

A=32'b0\_10000011\_00000000000000000000000;

B=32'b0\_01111111\_00000000000000000000000;

#10

A=32'b0\_10000011\_00010110101110000101001;

B=32'b0\_10000000\_01001100110011001100110;

#10

B=32'b0\_10000011\_00010110101110000101001;

A=32'b0\_10000000\_01001100110011001100110;

#10

A=32'b0\_01111110\_00000000000000000000000;

B=32'b0\_10000000\_01000000000000000000000;

#10

B=32'b0\_01111110\_00000000000000000000000;

A=32'b0\_10000000\_01000000000000000000000;

#10

B=32'b0\_01111110\_00000000000000000000000;

A=32'b0\_01111110\_00000000000000000000000;

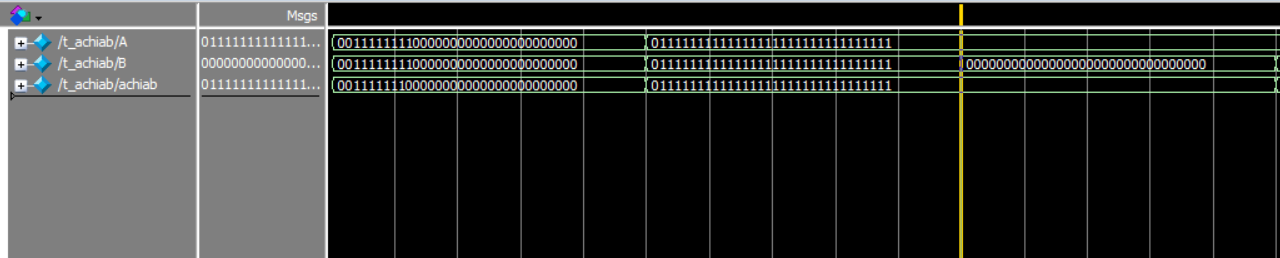
#10

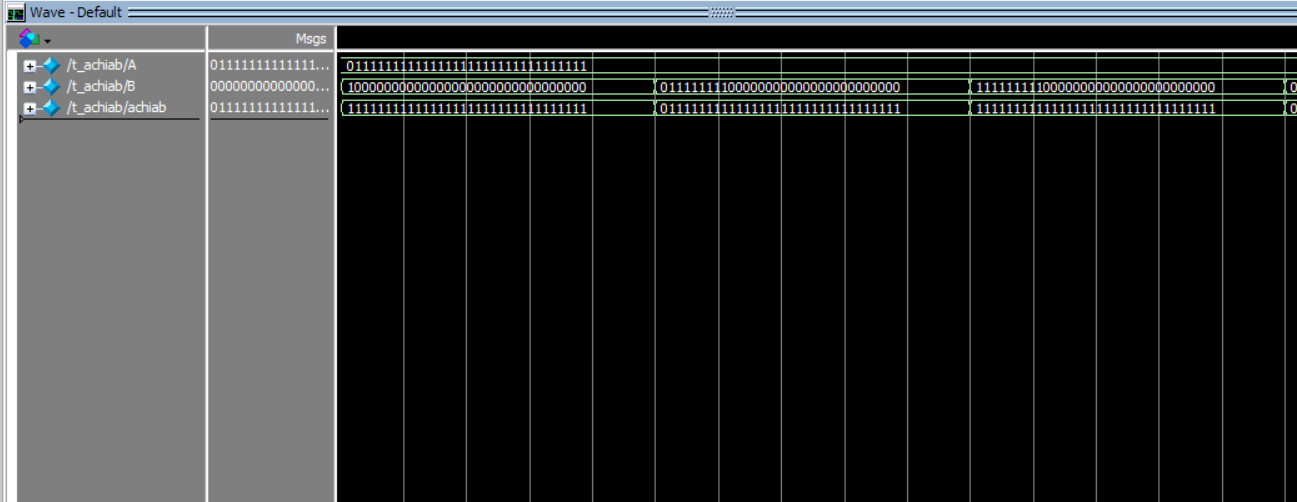
;

end

endmodule

III/Mô phỏng kết quả





#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