ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

Khoa Điện – Điện tử Bộ môn Điện tử

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



BÁO CÁO XÂY DỰNG

BỘ NHÂN 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**

Phụ lục

[I.MỤC TIÊU: 3](#_Toc23648)

[1. Thuật toán nhân 3](#_Toc27726)

[2. Trường hợp đặc biệt 5](#_Toc15463)

[II.CODE DESIGN – CODE TESTBENCH 7](#_Toc23507)

[1. Code cộng 1 bit full adder 7](#_Toc20178)

[2. Code cong 8 bit 7](#_Toc13180)

[3. Code công phần mũ của 2 số 8](#_Toc22397)

[4. Code trừ 1 bit 9](#_Toc5294)

[5. Mux 2 sang 1 32 bit 9](#_Toc17386)

[6. Mux 2 sang 1 23 bit 10](#_Toc6778)

[7.Mux 2 sang 1 8 bit 10](#_Toc24557)

[8.Mux 2 sang 1 49 bit 10](#_Toc4701)

[9.Code nhân faction 10](#_Toc6174)

[10.Code trường hợp đặt biệt 12](#_Toc32603)

[11.Code bộ nhân 13](#_Toc12565)

[III/Mô phỏng kết quả 15](#_Toc12995)

I.MỤC TIÊU:

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]

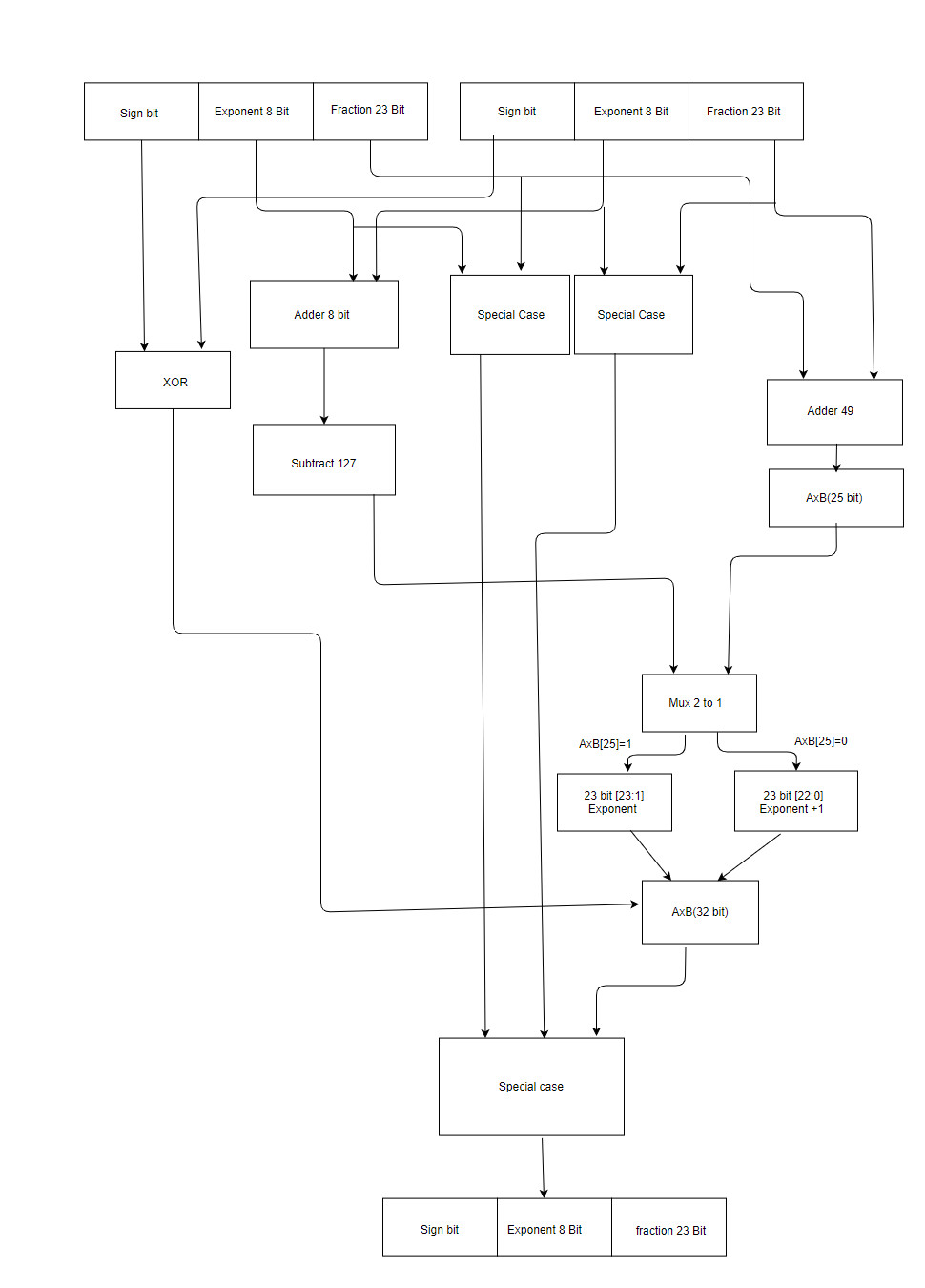
Vd: 0\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx

xxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx

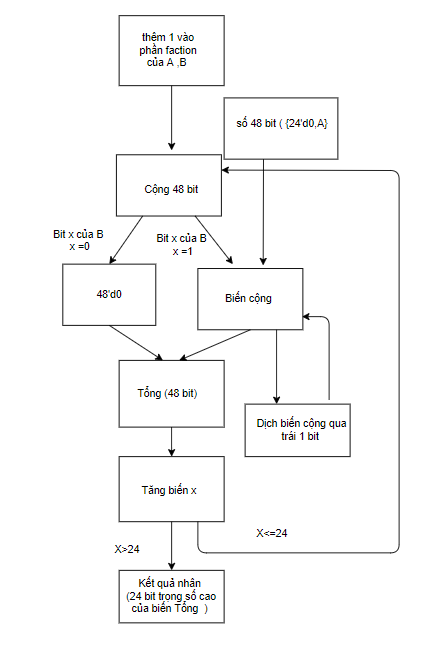
Kiểm tra bit 10 của mũ nếu là 1 thì kết quả là Zero

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

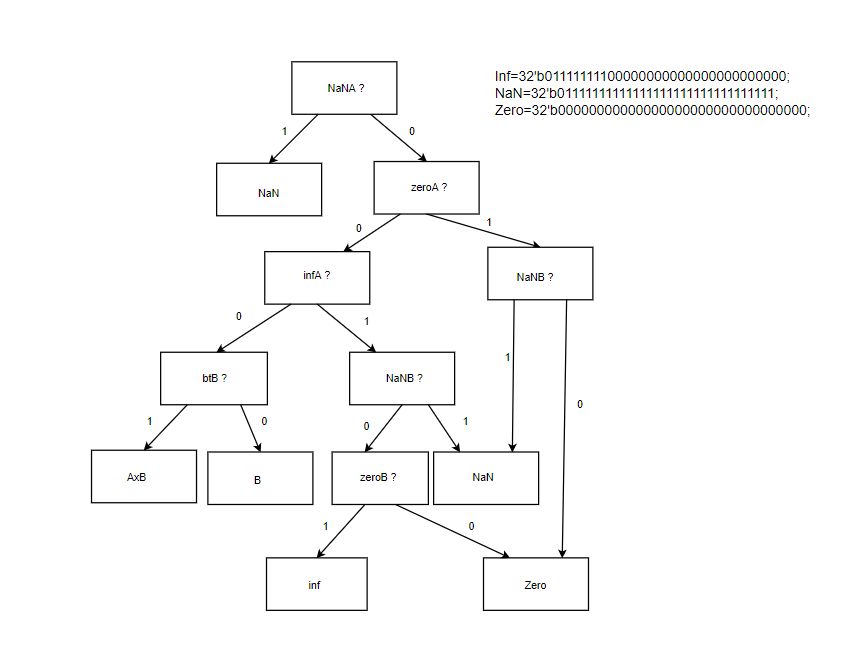
NaN=32'b01111111111111111111111111111111;

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

II.CODE DESIGN – CODE TESTBENCH

1. Code cộng 1 bit full adder

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

input a , b , cin;

output s , cout;

wire a,b,cin;

wire c1,c0;

wire sum,cout,s ;

half\_adder half\_adder\_00(a , b , c0, sum );

half\_adder half\_adder\_01(sum , cin , c1 , s);

assign cout = c1 | c0;

endmodule

module half\_adder (a, b, c, s);

input a,b;

output s,c;

wire a,b,c,s;

assign s = a ^ b;

assign c = a &b;

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

1. Code cong 8 bit

module cong8bit(in1, in2, S, Cout);//bo cong 8bit

input [7:0] in1, in2;

output [8:0] S;

output Cout;

wire [8:1] temp;

reg Cout;

FA FA\_0(in1[0], in2[0], 1'b0, S[0], temp[1]);

FA FA\_1(in1[1],in2[1], temp[1], S[1], temp[2]);

FA FA\_2(in1[2], in2[2], temp[2], S[2], temp[3]);

FA FA\_3(in1[3], in2[3], temp[3], S[3], temp[4]);

FA FA\_4(in1[4], in2[4], temp[4], S[4], temp[5]);

FA FA\_5(in1[5], in2[5], temp[5], S[5], temp[6]);

FA FA\_6(in1[6], in2[6], temp[6],S[6], temp[7]);

FA FA\_7(in1[7], in2[7],temp[7], S[7], temp[8]);

assign Cout=temp[8];

assign S[8]=Cout;

endmodule

module FA (a, b, cin, S ,cout);

input a , b , cin;

output S , cout;

wire a,b,cin;

wire c1,c0;

wire sum,cout,S ;

half\_adder half\_adder\_00(a , b , c0, sum );

half\_adder half\_adder\_01(sum , cin , c1 , S);

assign cout = c1 | c0;

endmodule

module half\_adder (a, b, c, S);

input a,b;

output S,c;

wire a,b,c,S;

assign S = a ^ b;

assign c = a &b;

endmodule

module t\_cong8bit;

reg [7:0]in1,in2;

wire [8:0]sum;

wire cout;

cong8bit c8b\_1(in1,in2,sum,cout);

initial begin

#0

in1 = 8'b0000\_1111;

in2 = 8'b0000\_1100;

#100

in1 = 8'b1111\_1111;

in2 = 8'b1111\_1100;

#100;

end

endmodule

1. Code công phần mũ của 2 số

module Multi\_tru9bit(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

module Multi\_CongMu(A,B,ex);

input [31:0]A,B;

output [9:0]ex;

wire [8:0]ex1;

cong8bit a1(A[30:23],B[30:23],ex1[8:0],/\*\*/);

Multi\_tru9bit a2(ex1[8:0],9'b001111111,ex[9:0],/\*\*/); //ex(A) + ex(B) - 127

endmodule

module t\_Multi\_CongMu;

reg [31:0]A,B;

wire [9:0]ex;

Multi\_CongMu a(A,B,ex);

initial begin

#0

A=32'b0\_11111110\_11000000000000000000000;

B=32'b0\_11111110\_10000000000000000000000;

#100

A=32'b0\_00001111\_11000000000000000000000;

B=32'b0\_00010000\_00000000000000000000000;

#100

A=32'b0\_11110000\_11111111111111111111111;

B=32'b0\_00001111\_11111111111111111111111;

#100

A=32'b0\_11100000\_11000000000000000000011;

B=32'b0\_00000011\_11000000000000000000011;

#100

;

end

endmodule

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

assign A\_xor\_B= ((a==0)&&(b==1)||(a==1)&&(b==0))?1:0;

assign s= (((cin==0)&&(A\_xor\_B==1))||((cin==1)&&(A\_xor\_B==0)))?1:0;

//tinh cout

assign not\_a=!a;

assign B\_or\_Cin = ((b==0)&&(cin==0))?0:1;

assign c1=((not\_a==1)&&(B\_or\_Cin==1))?1:0;

assign c2=((b==1)&&(cin==1))?1:0;

assign cout = ((c1==0)&&(c2==0))?0:1;

endmodule

1. Mux 2 sang 1 32 bit

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

1. Mux 2 sang 1 23 bit

module mux2to123bit(s,in1,in2,out); //s=0 out=in1 // s=1 out=in2

input wire s;

input wire [22:0]in1,in2;

output wire [22:0]out;

assign out = (s)? in1 : in2;

endmodule

7.Mux 2 sang 1 8 bit

module mux2to1(s,in1,in2,out); //s=0 out=in1 // s=1 out=in2

input wire s;

input wire [7:0]in1,in2;

output wire [7:0]out;

assign out = (s)? in1 : in2;

endmodule

8.Mux 2 sang 1 49 bit

module mux2to148bit(s,in1,in2,out); //s=0 out=in1 // s=1 out=in2

input wire s;

input wire [48:0]in1,in2;

output wire [48:0]out;

assign out = (s)? in1 : in2;

endmodule

9.Code nhân faction

module nhan\_24bit(a,b,ketquatruoc,ketqua);

input [23:0] a,b;

output [47:0] ketquatruoc;

output [24:0] ketqua;

wire [47:0]X[24:1];

wire [47:0]S[23:0];

assign X[1] = b[0]?{24'b0,a}:48'b0;

assign X[2] = b[1]?{23'b0,a,1'b0}:48'b0;

adder\_48bit add1(.in1(X[1]), .in2(X[2]), .Sout(S[1]), .Cout());

assign X[3] = b[2]?{22'b0,a,2'b0}:48'b0;

adder\_48bit add2(.in1(S[1]), .in2(X[3]), .Sout(S[2]), .Cout());

assign X[4] = b[3]?{21'b0,a,3'b0}:48'b0;

adder\_48bit add3(.in1(S[2]), .in2(X[4]), .Sout(S[3]), .Cout());

assign X[5] = b[4]?{20'b0,a,4'b0}:48'b0;

adder\_48bit add4(.in1(S[3]), .in2(X[5]), .Sout(S[4]), .Cout());

assign X[6] = b[5]?{19'b0,a,5'b0}:48'b0;

adder\_48bit add5(.in1(S[4]), .in2(X[6]), .Sout(S[5]), .Cout());

assign X[7] = b[6]?{18'b0,a,6'b0}:48'b0;

adder\_48bit add6(.in1(S[5]), .in2(X[7]), .Sout(S[6]), .Cout());

assign X[8] = b[7]?{17'b0,a,7'b0}:48'b0;

adder\_48bit add7(.in1(S[6]), .in2(X[8]), .Sout(S[7]), .Cout());

assign X[9] = b[8]?{16'b0,a,8'b0}:48'b0;

adder\_48bit add8(.in1(S[7]), .in2(X[9]), .Sout(S[8]), .Cout());

assign X[10] = b[9]?{15'b0,a,9'b0}:48'b0;

adder\_48bit add9(.in1(S[8]), .in2(X[10]), .Sout(S[9]), .Cout());

assign X[11] = b[10]?{14'b0,a,10'b0}:48'b0;

adder\_48bit add10(.in1(S[9]), .in2(X[11]), .Sout(S[10]), .Cout());

assign X[12] = b[11]?{13'b0,a,11'b0}:48'b0;

adder\_48bit add11(.in1(S[10]), .in2(X[12]), .Sout(S[11]), .Cout());

assign X[13] = b[12]?{12'b0,a,12'b0}:48'b0;

adder\_48bit add12(.in1(S[11]), .in2(X[13]), .Sout(S[12]), .Cout());

assign X[14] = b[13]?{11'b0,a,13'b0}:48'b0;

adder\_48bit add13(.in1(S[12]), .in2(X[14]), .Sout(S[13]), .Cout());

assign X[15] = b[14]?{10'b0,a,14'b0}:48'b0;

adder\_48bit add14(.in1(S[13]), .in2(X[15]), .Sout(S[14]), .Cout());

assign X[16] = b[15]?{9'b0,a,15'b0}:48'b0;

adder\_48bit add15(.in1(S[14]), .in2(X[16]), .Sout(S[15]), .Cout());

assign X[17] = b[16]?{8'b0,a,16'b0}:48'b0;

adder\_48bit add16(.in1(S[15]), .in2(X[17]), .Sout(S[16]), .Cout());

assign X[18] = b[17]?{7'b0,a,17'b0}:48'b0;

adder\_48bit add17(.in1(S[16]), .in2(X[18]), .Sout(S[17]), .Cout());

assign X[19]= b[18]?{6'b0,a,18'b0}:48'b0;

adder\_48bit add18(.in1(S[17]), .in2(X[19]), .Sout(S[18]), .Cout());

assign X[20] = b[19]?{5'b0,a,19'b0}:48'b0;

adder\_48bit add19(.in1(S[18]), .in2(X[20]), .Sout(S[19]), .Cout());

assign X[21] = b[20]?{4'b0,a,20'b0}:48'b0;

adder\_48bit add20(.in1(S[19]), .in2(X[21]), .Sout(S[20]), .Cout());

assign X[22] = b[21]?{3'b0,a,21'b0}:48'b0;

adder\_48bit add21(.in1(S[20]), .in2(X[22]), .Sout(S[21]), .Cout());

assign X[23] = b[22]?{2'b0,a,22'b0}:48'b0;

adder\_48bit add22(.in1(S[21]), .in2(X[23]), .Sout(S[22]), .Cout());

assign X[24] = b[23]?{1'b0,a,23'b0}:48'b0;

adder\_48bit add23(.in1(S[22]), .in2(X[24]), .Sout(S[23]), .Cout());

assign ketquatruoc = S[23];

assign ketqua = ketquatruoc[47:23]; //25 bit???

endmodule

module adder\_48bit(in1, in2, Sout, Cout);

input [47:0] in1,in2;

output [47:0] Sout;

output Cout;

wire [47:0] S;

wire [48:1] temp\_c;

FA FA[47:0](in1[47:0],in2[47:0],{temp\_c[47:1],1'b0},S[47:0],{Cout,temp\_c[47:1]});

assign Sout = S;

endmodule

module t\_nhanfac;

reg [23:0]A,B;

wire [47:0]AxB;

wire [24:0]ketqua;

nhan\_24bit m7(A[23:0],B[23:0],AxB[47:0],ketqua[24:0]);

initial begin

#0

B=24'b1000\_0000\_0000\_0000\_0000\_0000;

A=24'b1110\_1111\_0000\_1101\_0001\_1000;

#10

A=24'b1000\_0000\_0000\_0000\_0000\_0000;

B=24'b1000\_1111\_0000\_1101\_0001\_1000;

#10

A=24'b 1011\_1001\_1101\_0110\_1101\_1111;

B=24'b 0110\_1110\_1111\_0110\_1011\_0000;

#10

A=24'b 1111\_0011\_1001\_0110\_0011\_1000;

B=24'b 1100\_0001\_1100\_1001\_0011\_0110;

#10

A=24'b 0100\_1100\_0011\_1111\_0110\_0010;

B=24'b 1000\_1011\_1000\_0011\_1000\_0011;

#10

A=24'b 0000\_0000\_0111\_0111\_0001\_1010;

B=24'b 0000\_0000\_0011\_1100\_1111\_1110;

#10

A=24'b 0000\_0000\_0000\_0111\_1100\_0011;

B=24'b 0000\_0000\_0001\_1000\_0111\_1101;

#100

A=24'b 1111\_0000\_0000\_0110\_0011\_1000;

B=24'b 1000\_0000\_0000\_0001\_0011\_0110;

#100

A=24'b 0100\_1100\_0011\_1111\_0110\_0010;

B=24'b 0000\_1011\_1000\_0011\_1000\_0011;

#100

A=24'b 0000\_0001\_0111\_0111\_0001\_1010;

B=24'b 0000\_0010\_0011\_1100\_1111\_1110;

#100

A=24'b 0000\_0000\_1000\_0111\_1100\_0011;

B=24'b 0000\_0000\_1101\_1000\_0111\_1101;

#100

;

end

endmodule

10.Code trường hợp đặt biệt

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

11.Code bộ nhân

module mult(A,B,C);

input wire[31:0]A,B;

output wire [31:0]C;

wire dauC,dauA,dauB;

wire [7:0] munho,muout,muC,muA,muB;

wire [47:0]cout;

wire [22:0]tpAB; // bien de luu gia tri sau khi dich bit

wire [24:0]tpAB\_cd; // 1 bit chua du 23 bit chua kq A\*B

wire [31:0]Inf,C1,kqAB,Zero;

wire [7:0]exAB;

wire [8:0]ex1;wire [7:0]tru;

wire [9:0]ex;

wire [23:0]gtA,gtB;

wire infA,nanA,zeroA,btA,infB,nanB,zeroB,btB,Sign;

wire [31:0]NaN,KQ\_NoSign,o1,o2,o3,o4,o5,o6,AxB,AxB1;

assign dauA = A[31];

assign muA = A[30:23];

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

assign dauB = B[31];

assign muB = B[30:23];

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

assign Inf=32'b01111111100000000000000000000000; //inf

assign Zero=32'b00000000000000000000000000000000;

assign NaN=32'b01111111111111111111111111111111;

assign tru= 8'b0000\_0001;

assign Sign = A[31]^B[31];

Multi\_CongMu n1(A[31:0],B[31:0],ex[9:0]);

// Nhan Thap Phan

nhan\_24bit k2(gtA,gtB,cout,tpAB\_cd);

//assign tpAB\_cd[24:0]=out[47:23];

// Dich Thap Phan

mux2to123bit k3(tpAB\_cd[24],tpAB\_cd[23:1],tpAB\_cd[22:0],tpAB[22:0]);

cong8bit h1(ex[7:0],tru,ex1[7:0]);

// Xac Dinh Mu

mux2to1 k4(tpAB\_cd[24],ex1[7:0],ex[7:0],exAB[7:0]); // neu co them bit du tang them so mu

// A Nhan B

assign kqAB[31:0]={Sign,exAB[7:0],tpAB[22:0]}; // bien de dat kq cuoi cung

/\* ex[9]=1\_Zero | ex[8]=1\_Inf | AxB \*/

//mux2to1kq m5(ex[8],kqAB[31:0],Inf[31:0],o1[31:0]); //ex[8]=1 => e(A) + e(B) trÃ n

//mux2to1kq m6(ex[9],o1[31:0],Zero[31:0],AxB[31:0]); //ex[9]=1 => ex(AB)<0

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

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

mux2to1kq m52(ex[8],kqAB[31:0],Inf[31:0],C1[31:0]); // neu tran tren thi ra infinity

mux2to1kq m62(ex[9],C1[31:0],Zero[31:0],AxB[31:0]); //neu tran duoi thi zero

assign o1=btB?AxB1:B;

mux2to1kq m1(btB,B[31:0],AxB[31:0],o1[31:0]);

mux2to1kq m2(zeroB,Inf[31:0],Zero[31:0],o2[31:0]);

mux2to1kq m3(nanB,o2[31:0],NaN[31:0],o3[31:0]);

mux2to1kq m4(infA,o1[31:0],o3[31:0],o4[31:0]);

mux2to1kq m9(nanB,Zero[31:0],NaN[31:0],o5[31:0]);

mux2to1kq m10(zeroA,o4[31:0],o5[31:0],o6[31:0]);

mux2to1kq m7(nanA,o6[31:0],NaN[31:0],KQ\_NoSign[31:0]);

assign C[31:0]=KQ\_NoSign[31:0];

endmodule

module t\_AnhanC;

reg [31:0]A,B;

wire [31:0]AxB;

mult m7(A[31:0],B[31:0],AxB[31:0]);

initial begin

#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;

#10

A=32'b1\_00000001\_11000000000000000000000;

B=32'b1\_00000010\_11000000000000000000000;

#10

A=32'b0\_10000000\_00101100001010001111011; // 2.345

B=32'b0\_10000000\_10111010010111100011011; // 3.456

#100

;

end

endmodule

III/Mô phỏng kết quả

