ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

Khoa Điện – Điện tử Bộ môn Điện tử

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



BÁO CÁO XÂY DỰNG CHUYỂN SỐ FLOATING POINT THÀNH DẠNG THẬP PHÂN

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](#_Toc17504)

[1. Giải thuật 3](#_Toc25846)

[II. CODE DESIGN – CODE TESTBENCH 4](#_Toc17336)

[1. Code floating point sang số thực 4](#_Toc26838)

[2. Code trừ 8 bit 5](#_Toc27776)

[3. Code cộng 8 bit 6](#_Toc8933)

[4. Code cộng 24 bit 7](#_Toc11320)

[5. Code tìm phần nguyên 8](#_Toc30888)

[6. Code tìm phần thập phân 11](#_Toc20560)

[7. Code tính phần mũ 13](#_Toc28639)

[III. Mô phỏng kết quả 16](#_Toc17743)

1. **Mục tiêu**

Chuyển ngõ vào ngõ ra thành dạng thập phân tương ứng

Vd:

0\_0111\_1111\_0000\_0000\_0000\_0000\_0000\_000

+1\*10^(+0)

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 +

1. CODE DESIGN – CODE TESTBENCH
2. Code floating point sang số thực

module FP\_sang\_TP(a,dau,dau2,phannguyen,phanmu,b,c,c1);

input [31:0]a; // so fp

output [7:0]dau;

output [7:0]dau2;

output real phanmu;

output real phannguyen ;

wire [7:0]dau1,S;

wire [7:0]dau;

output wire [23:0]b; // chua gia tri nguyen

output wire [22:0]c;

output wire [22:0] c1; // chua gia tri mu

wire Cout;

assign dau=(a[31]==1'b0)?(8'b0010\_1011):(8'b0010\_1101); // xác d?nh dau

botru z(.in1(a[30:23]),.S(S), .Cout(Cout),.dau(dau1)); /// tinh phan 2^S và dau

tim\_phan\_nguyen z1( .mu(S), .dau(dau1),.a(a[22:0]),.b(b),.c(c));

tinh\_phan\_tp z2(.a(c),.b(c1));

tinhmu1 z3(.a(b),.b(phannguyen),.c(c1),.mu(phanmu),.mu1(dau2));

endmodule

module t\_FP\_sang\_TP;

real phanmu;

wire [7:0]dau;

wire [7:0]dau2;

reg [31:0]a;

real phannguyen;

wire [23:0]b; wire [22:0]c1;wire [22:0]c;

parameter time\_out = 100;

FP\_sang\_TP z(.a(a),.dau(dau),.phannguyen(phannguyen),.dau2(dau2),.phanmu(phanmu),.c(c),.b(b),.c1(c1));

initial $monitor($time," %b so %c%f \*10^ %c%d ",a,dau,phannguyen,dau2,phanmu );

initial begin

#0 a=32'b1\_1000\_1100\_1111\_0000\_0000\_0000\_0000\_000;

#10 a=32'b0\_1000\_0010\_1110\_0001\_0000\_0000\_0000\_000;

#10 a=32'b1\_0111\_1100\_1000\_0000\_0000\_0000\_0000\_000;

#10 a=33'b1\_1000\_1011\_1000\_0000\_0000\_0000\_0000\_000;

#10 a=33'b1\_1000\_0101\_1000\_0000\_0100\_0000\_0000\_000;

end

endmodule

1. Code trừ 8 bit

module botru(in1, S, Cout,dau);

input [7:0] in1;

output [7:0] S;

output Cout;

output [7:0]dau;

wire [7:0]in11,in22,in111;

wire Cout;

wire [7:0]dau;

assign in111=~in1;

assign in11=(in1>=(8'b0111\_1111))?in1:in111;

assign in22=(in1>=(8'b0111\_1111))?(8'b1000\_0001):(8'b1000\_0000);

cong\_8bit r(in11,in22,S,Cout);

assign dau=(in1>=(8'b0111\_1111))?(8'h2B):(8'h2D); // + -

endmodule

module t\_botru;

reg [7:0]in1;

wire [7:0]S;

wire Cout;

wire [7:0]dau;

parameter time\_out = 100;

botru z(.in1(in1),.S(S),.Cout(Cout),.dau(dau));

initial $monitor($time," so in1 %d , %c %d ,dau %b ", in1,dau,S,Cout );

initial begin

#0 in1=8'b1000\_0000;

#10 in1=8'b1000\_1010;

#10 in1=8'b0010\_1111;

#10 in1=8'b0111\_1111;

end

endmodule

1. Code cộng 8 bit

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 cong\_8bit( in1,in2 , S , Cout);

input [7:0]in1,in2;

output [7:0]S ;

output Cout;

wire [8:1] temp;

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 chia\_4bit(

input [3:0]a,

output [3:0]b,

output c

);

wire [3:0]a1;

assign a1=4'b0101;

FA FA[3:0](.a(a[3:0]), .b(a1[3:0]), .cin(1'b1), .s(b) ,.cout(c) );

endmodule

module t\_bocong;

reg [7:0]in1;

reg [7:0]in2;

wire [7:0]S;

wire Cout;

parameter time\_out = 100;

cong\_8bit z(.in1(in1),.in2(in2),.S(S),.Cout(Cout));

initial $monitor($time," so in1 %d , so in2 %d so s %d ,dau %b ", in1,in2,S,Cout );

initial begin

#0 in1=8'b1000\_0000;

in2=8'b1000\_0001;

#10 in1=8'b0110\_1010;

in2=8'b0000\_0111;

#10 in1=8'b0010\_1111;

in2=8'b0000\_0001;

#10 in1=8'b0111\_1111;

in2=8'b0000\_1101;

end

endmodule

1. Code cộng 24 bit

module cong\_24bit(in1,in2,sum,c);

output [24:0] sum;

output [24:0]c;

input [23:0] in1;

input [23:0] in2;

wire [23:0]cout;

FA FA[23:0] (in1[23:0], in2[23:0], {c[22:0],1'b0}, sum[23:0],c[23:0]);

assign cout = c[23];

assign sum[24] = cout;

endmodule

module t\_cong\_24;

reg [23:0]in1; reg [23:0]in2;

wire [24:0]sum;

wire c;

parameter time\_out = 100;

cong\_24bit z(.in1(in1),.in2(in2),.sum(sum),.c(c));

initial $monitor($time," so in1 %d so in2 %d , %d , %d ", in1,in2,sum,c );

initial begin

#0 in1=24'd5000\_000; in2=24'd2500\_000;

#10 in1=24'd1250\_000; in2=24'd100\_0000;

//#10 in1=8'b0010\_1111; in1=8'b1000\_0000;

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

end

endmodule

1. Code tìm phần nguyên

module tim\_phan\_nguyen (

input wire[7:0]mu,

input wire[7:0]dau,

input [22:0]a,

output reg [23:0]b,

output reg [22:0]c

);

always @(mu or a)

if( dau <= 8'h2B)

case (mu)

7'd0: begin b[23:0]<={(22'b0),1'b1}; c[22:0]<=a[22:0]; end

7'd1: begin b[23:0]<={(22'b0),1'b1,a[22]}; c[22:0]<={a[21:0],1'b0}; end

7'd2: begin b[23:0]<={(21'b0),1'b1,a[22:21]}; c[22:0]<={a[20:0],2'b0}; end

7'd3: begin b[23:0]<={(20'b0),1'b1,a[22:20]}; c[22:0]<={a[19:0],3'b0}; end

7'd4: begin b[23:0]<={(19'b0),1'b1,a[22:19]}; c[22:0]<={a[18:0],4'b0}; end

7'd5: begin b[23:0]<={(18'b0),1'b1,a[22:18]}; c[22:0]<={a[17:0],5'b0}; end

7'd6: begin b[23:0]<={(17'b0),1'b1,a[22:17]}; c[22:0]<={a[16:0],6'b0}; end

7'd7: begin b[23:0]<={(16'b0),1'b1,a[22:16]}; c[22:0]<={a[15:0],7'b0}; end

7'd8: begin b[23:0]<={(15'b0),1'b1,a[22:15]}; c[22:0]<={a[14:0],8'b0}; end

7'd9: begin b[23:0]<={(14'b0),1'b1,a[22:14]}; c[22:0]<={a[13:0],9'b0}; end

7'd10: begin b[23:0]<={(13'b0),1'b1,a[22:13]}; c[22:0]<={a[12:0],10'b0}; end

7'd11: begin b[23:0]<={(12'b0),1'b1,a[22:12]}; c[22:0]<={a[11:0],11'b0}; end

7'd12: begin b[23:0]<={(11'b0),1'b1,a[22:11]}; c[22:0]<={a[10:0],12'b0}; end

7'd13: begin b[23:0]<={(10'b0),1'b1,a[22:10]}; c[22:0]<={a[9:0],13'b0}; end

7'd14: begin b[23:0]<={(9'b0),1'b1,a[22:9]}; c[22:0]<={a[8:0],14'b0}; end

7'd15: begin b[23:0]<={(8'b0),1'b1,a[22:8]}; c[22:0]<={a[7:0],15'b0}; end

7'd16: begin b[23:0]<={(7'b0),1'b1,a[22:7]}; c[22:0]<={a[6:0],16'b0}; end

7'd17: begin b[23:0]<={(6'b0),1'b1,a[22:6]}; c[22:0]<={a[5:0],17'b0}; end

7'd18: begin b[23:0]<={(5'b0),1'b1,a[22:5]}; c[22:0]<={a[4:0],18'b0}; end

7'd19: begin b[23:0]<={(4'b0),1'b1,a[22:4]}; c[22:0]<={a[3:0],19'b0}; end

7'd20: begin b[23:0]<={(3'b0),1'b1,a[22:3]}; c[22:0]<={a[2:0],20'b0}; end

7'd21: begin b[23:0]<={(2'b0),1'b1,a[22:2]}; c[22:0]<={a[1:0],21'b0}; end

7'd22: begin b[23:0]<={(1'b0),1'b1,a[22:1]}; c[22:0]<={a[0],22'b0}; end

7'd23: begin b[23:0]<={1'b1,a[22:0]}; c[22:0]<={23'b0}; end

default begin b[23:0]<={24'b1}; c[22:0]<={23'b0}; end

endcase

else case (mu)

7'd1: begin b[23:0]<={(24'b0)}; c[22:0]<={1'b1,a[22:1]}; end

7'd2: begin b[23:0]<={(24'b0)}; c[22:0]<={1'b0,1'b1,a[22:2]}; end

7'd3: begin b[23:0]<={(24'b0)}; c[22:0]<={2'b0,1'b1,a[22:3]}; end

7'd4: begin b[23:0]<={(24'b0)}; c[22:0]<={3'b0,1'b1,a[22:4]}; end

7'd5: begin b[23:0]<={(24'b0)}; c[22:0]<={4'b0,1'b1,a[22:5]}; end

7'd6: begin b[23:0]<={(24'b0)}; c[22:0]<={5'b0,1'b1,a[22:6]}; end

7'd7: begin b[23:0]<={(24'b0)}; c[22:0]<={6'b0,1'b1,a[22:7]}; end

7'd8: begin b[23:0]<={(24'b0)}; c[22:0]<={7'b0,1'b1,a[22:8]}; end

7'd9: begin b[23:0]<={(24'b0)}; c[22:0]<={8'b0,1'b1,a[22:9]}; end

7'd10: begin b[23:0]<={(24'b0)}; c[22:0]<={9'b0,1'b1,a[22:10]}; end

7'd11: begin b[23:0]<={(24'b0)}; c[22:0]<={10'b0,1'b1,a[22:11]}; end

7'd12: begin b[23:0]<={(24'b0)}; c[22:0]<={11'b0,1'b1,a[22:12]}; end

7'd13: begin b[23:0]<={(24'b0)}; c[22:0]<={12'b0,1'b1,a[22:13]}; end

7'd14: begin b[23:0]<={(24'b0)}; c[22:0]<={13'b0,1'b1,a[22:14]}; end

7'd15: begin b[23:0]<={(24'b0)}; c[22:0]<={14'b0,1'b1,a[22:15]}; end

7'd16: begin b[23:0]<={(24'b0)}; c[22:0]<={15'b0,1'b1,a[22:16]}; end

7'd17: begin b[23:0]<={(24'b0)}; c[22:0]<={16'b0,1'b1,a[22:17]}; end

7'd18: begin b[23:0]<={(24'b0)}; c[22:0]<={17'b0,1'b1,a[22:18]}; end

7'd19: begin b[23:0]<={(24'b0)}; c[22:0]<={18'b0,1'b1,a[22:19]}; end

7'd20: begin b[23:0]<={(24'b0)}; c[22:0]<={19'b0,1'b1,a[22:20]}; end

7'd21: begin b[23:0]<={(24'b0)}; c[22:0]<={20'b0,1'b1,a[22:21]}; end

7'd22: begin b[23:0]<={(24'b0)}; c[22:0]<={21'b0,1'b1,a[22]}; end

7'd23: begin b[23:0]<={24'b0}; c[22:0]<={22'b0,1'b1}; end

default begin b[23:0]<={24'b0}; c[22:0]<={23'b0}; end

endcase

endmodule

module t\_tim\_phan\_nguyen;

reg [7:0]mu;reg [7:0]dau;

reg [22:0]a;

wire [23:0]b;

wire [22:0]c;

parameter time\_out = 100;

tim\_phan\_nguyen z(.mu(mu),.dau(dau),.a(a),.b(b),.c(c));

initial $monitor($time," so mu %d , %b , %b , %b ", mu,a,b,c );

initial begin

#0 dau=8'h2B; mu=8'b0000\_0100; a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_0110;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2B;mu=8'b0000\_0111;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1000;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1001;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1010;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1011;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1100;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1101;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1110;a=23'b100\_0001\_1111\_0000\_0000\_0000;

#10 dau=8'h2D;mu=8'b0000\_1111;a=23'b100\_0001\_1111\_0000\_0000\_0000;

end

endmodule

1. Code tìm phần thập phân

module tinh\_phan\_tp(

input wire [22:0]a,

output [22:0]b

);

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

wire [23:0]in2[24:0];

wire [24:0]c;

assign in2[0]=(a[22]==1)?(24'd500\_0000):(24'd0); //0

cong\_24bit cong0(.in1(24'b0),.in2(in2[0]),.sum(sum[0]),.c(c[0]));

assign in2[1]=(a[21]==1)?(24'd250\_0000):(24'd0); //1

cong\_24bit cong1(.in1(sum[0]),.in2(in2[1]),.sum(sum[1]),.c(c[1]));

assign in2[2]=(a[20]==1)?(24'd125\_0000):(24'd0);

cong\_24bit cong2(.in1(sum[1]),.in2(in2[2]),.sum(sum[2]),.c(c[2]));

assign in2[3]=(a[19]==1)?(24'd62\_5000):(24'd0);//3

cong\_24bit cong3(.in1(sum[2]),.in2(in2[3]),.sum(sum[3]),.c(c[3]));

assign in2[4]=(a[18]==1)?(24'd31\_2500):(24'd0);//4

cong\_24bit cong4(.in1(sum[3]),.in2(in2[4]),.sum(sum[4]),.c(c[4]));

assign in2[5]=(a[17]==1)?(24'd15\_6250):(24'd0); //5

cong\_24bit cong5(.in1(sum[4]),.in2(in2[5]),.sum(sum[5]),.c(c[5]));

assign in2[6]=(a[16]==1)?(24'd7\_8125):(24'd0);//6

cong\_24bit cong6(.in1(sum[5]),.in2(in2[6]),.sum(sum[6]),.c(c[6]));

assign in2[7]=(a[15]==1)?(24'd3\_9062):(24'd0);

cong\_24bit cong7(.in1(sum[6]),.in2(in2[7]),.sum(sum[7]),.c(c[7]));

assign in2[8]=(a[14]==1)?(24'd1\_9531):(24'd0);//8

cong\_24bit cong8(.in1(sum[7]),.in2(in2[8]),.sum(sum[8]),.c(c[8]));

assign in2[9]=(a[13]==1)?(24'd9765):(24'd0);

cong\_24bit cong9(.in1(sum[8]),.in2(in2[9]),.sum(sum[9]),.c(c[9]));

assign in2[10]=(a[12]==1)?(24'd4882):(24'd0);

cong\_24bit cong10(.in1(sum[9]),.in2(in2[10]),.sum(sum[10]),.c(c[10]));

assign in2[11]=(a[11]==1)?(24'd2441):(24'd0);

cong\_24bit cong11(.in1(sum[10]),.in2(in2[11]),.sum(sum[11]),.c(c[11]));

assign in2[12]=(a[10]==1)?(24'd1220):(24'd0);

cong\_24bit cong12(.in1(sum[11]),.in2(in2[12]),.sum(sum[12]),.c(c[12]));

assign in2[13]=(a[9]==1)?(24'd610):(24'd0);

cong\_24bit cong13(.in1(sum[12]),.in2(in2[13]),.sum(sum[13]),.c(c[13]));

assign in2[14]=(a[8]==1)?(24'd305):(24'd0);

cong\_24bit cong14(.in1(sum[13]),.in2(in2[14]),.sum(sum[14]),.c(c[14]));

assign in2[15]=(a[7]==1)?(24'd152):(24'd0);

cong\_24bit cong15(.in1(sum[14]),.in2(in2[15]),.sum(sum[15]),.c(c[15]));

assign in2[16]=(a[6]==1)?(24'd76):(24'd0);

cong\_24bit cong16(.in1(sum[15]),.in2(in2[16]),.sum(sum[16]),.c(c[16]));

assign in2[17]=(a[5]==1)?(24'd38):(24'd0);

cong\_24bit cong17(.in1(sum[16]),.in2(in2[17]),.sum(sum[17]),.c(c[17]));

assign in2[18]=(a[4]==1)?(24'd19):(24'd0);

cong\_24bit cong18(.in1(sum[17]),.in2(in2[18]),.sum(sum[18]),.c(c[18]));

assign in2[19]=(a[3]==1)?(24'd9):(24'd0);

cong\_24bit cong19(.in1(sum[18]),.in2(in2[19]),.sum(sum[19]),.c(c[19]));

assign in2[20]=(a[2]==1)?(24'd4):(24'd0);

cong\_24bit cong20(.in1(sum[19]),.in2(in2[20]),.sum(sum[20]),.c(c[20]));

assign in2[21]=(a[1]==1)?(24'd2):(24'd0);

cong\_24bit cong21(.in1(sum[20]),.in2(in2[21]),.sum(sum[21]),.c(c[21]));

assign in2[22]=(a[0]==1)?(24'd1):(24'd0);

cong\_24bit cong22(.in1(sum[21]),.in2(in2[22]),.sum(sum[22]),.c(c[22]));

assign b= sum[22];

endmodule

module t\_tinh\_phan\_tp;

reg [23:0]a;

wire [23:0]b;

parameter time\_out = 100;

tinh\_phan\_tp da(.a(a),.b(b));

initial $monitor($time," so a %b so %d ", a,b );

initial begin

#0 a=24'b1000\_0000\_0000\_0000\_0000\_0000;

#10 a=24'b1111\_1111\_1111\_1111\_1111\_1111;

#10 a=24'b0000\_0000\_0000\_0000\_0000\_1111;

//#10 in1=8'b0010\_1111; in1=8'b1000\_0000;

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

end

endmodule

1. Code tính phần mũ

module tinhmu1(a,b,c,mu,mu1);

input wire[23:0]a; // gia tri nguyen

input wire [22:0]c;

output real mu; // mu

output real b; // phan nguyen

output wire [7:0]mu1;

real b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11;

real g,m11,mu2,mu3,mu4,mu5,mu6,mu7,mu8,mu9,mu10,mu11;

assign mu1=(a==0)?(8'b0010\_1101):(8'b0010\_1011);

assign b1=((a>=24'd10))?((a\*0.1)+c\*0.00000001):a+c\*0.0000001;

assign mu11=((a>=24'd10))?(1):0;

assign b2=((b1>=10))?((b1\*0.1)):b1;

assign mu2=((b1>=24'd10))?(mu11+1):mu11;

assign b3=((b2>=10))?((b2\*0.1)):b2;

assign mu3=((b2>=24'd10))?(mu2+1):mu2;

assign b4=((b3>=10))?((b3\*0.1)):b3;

assign mu4=((b3>=24'd10))?(mu3+1):mu3;

assign b5=((b4>=10))?((b4\*0.1)):b4;

assign mu5=((b4>=24'd10))?(mu4+1):mu4; //

assign b6=((a==0))?((b1\*10)):b5;

assign mu6=((a==0))?(mu11+1):mu5;

assign b7=((b6<1))?((b6\*10)):b6;

assign mu7=((b6<1))?(mu6+1):mu6;

assign b8=((b7<1))?((b7\*10)):b7;

assign mu8=((b7<1))?(mu7+1):mu7;

assign b9=((b8<1))?((b8\*10)):b8;

assign mu9=((b8<1))?(mu8+1):mu8;

assign b10=((b9<1))?((b9\*10)):b9;

assign mu10=((b9<1))?(mu9+1):mu9;

assign b=((b10<1))?((b10\*10)):b10;

assign mu=((b10<1))?(mu10+1):mu10;

//assign b=((b11<1))?((b11\*10)):b11;

//assign mu=((b11<1))?(mu11+1):mu11;

//assign b=b10;

//assign mu=mu10;

endmodule

module t\_tinhmu1;

reg [23:0]a;

real b;

reg [22:0] c;

real mu;

wire [7:0]mu1;

parameter time\_out = 100;

tinhmu1 da(.a(a),.b(b),.c(c),.mu(mu),.mu1(mu1));

initial $monitor($time," so a %b %b la %f\*10^(%c%f) ", a,c,b,mu1,mu );

initial begin

#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;

end

endmodule

1. Mô phỏng kết quả



