CA_Lab1

K. Ram Mohan

COE19B055

Verilog Code:

```
// Module to compare exponent values of inputs
module CMP_EXP(ea, eb, sign_var);
   input[10:0]  ea, eb;
   output  reg sign_var;

   always @(*) begin
      if (ea>=eb) begin
         sign_var = 0;
      end
      if (ea<eb) begin
         sign_var = 1;
      end
   end
endmodule



// Module to swap inputs if necessary
module SWAP_M (ma, mb, sa, sb, ea, eb, sign_var, outa, outb, sa_out, sb_out, ea_out,
eb_out);
   input[52:0]  ma, mb;
   input sign_var;
   input sa, sb;
   input[10:0]  ea, eb;
```

```
output reg[52:0] outa, outb;

output reg sa_out, sb_out;

output reg[10:0] ea_out, eb_out;


always @(*) begin

   if (sign_var==1) begin

      outa = mb;

      outb = ma;

      sa_out = sb;

      sb_out = sa;

      ea_out = eb;

      eb_out = ea;

   end


   if (sign_var==0) begin

      if(ma>=mb) begin

         outa = ma;

         outb = mb;

         sa_out = sa;

         sb_out = sb;

         ea_out = ea;

         eb_out = eb;

      end


      if(ma < mb) begin

         outa = mb;

         outb = ma;

         sa_out = sb;
```

```
            sb_out = sa;
            ea_out = eb;
            eb_out = ea;
         end
      end
   end
endmodule


// Module to find difference of exponents
module DIFF_EXP (ea, eb, out);
   input[10:0]  ea, eb;
   output  reg[10:0] out;

   always @(*) begin
      if (ea>=eb) begin
         out = ea-eb;
      end
      if (ea<eb) begin
         out = eb-ea;
      end
   end
endmodule

// Module to shift smaller value
module RHT_SFT (mb, len, out);
   input[52:0]  mb;
   input[10:0]  len;
   output[52:0]  out;
```

```
    assign out = mb >> len;
endmodule

// Module for Full adder
module FA (in0, in1, cin, sum, cout);
    input in0, in1, cin;
    output sum, cout;
    wire c1, c2, c3;

    xor(sum, in0, in1, cin);
    and(c1, in0, in1);
    xor(c2, in0, in1);
    and(c3, c2, cin);
    or(cout, c1, c3);
endmodule

// Module to add same sign variables
module SAME_SIGN_ADD (ma, mb, out);
    input[53:0]  ma, mb;
    output[53:0]  out;

    wire[53:0] sum;
    wire[53:0] tmp;

    FA fa1(ma[0], mb[0], 1'b0,   sum[0], tmp[0]);
    FA fa2(ma[1], mb[1], tmp[0], sum[1], tmp[1]);
    FA fa3(ma[2], mb[2], tmp[1], sum[2], tmp[2]);
    FA fa4(ma[3], mb[3], tmp[2], sum[3], tmp[3]);
    FA fa5(ma[4], mb[4], tmp[3], sum[4], tmp[4]);
    FA fa6(ma[5], mb[5], tmp[4], sum[5], tmp[5]);
    FA fa7(ma[6], mb[6], tmp[5], sum[6], tmp[6]);
    FA fa8(ma[7], mb[7], tmp[6], sum[7], tmp[7]);
    FA fa9(ma[8], mb[8], tmp[7], sum[8], tmp[8]);
    FA fa10(ma[9], mb[9], tmp[8], sum[9], tmp[9]);
    FA fa11(ma[10], mb[10], tmp[9], sum[10], tmp[10]);
    FA fa12(ma[11], mb[11], tmp[10], sum[11], tmp[11]);
    FA fa13(ma[12], mb[12], tmp[11], sum[12], tmp[12]);

    FA fa14(ma[13], mb[13], tmp[12], sum[13], tmp[13]);
    FA fa15(ma[14], mb[14], tmp[13], sum[14], tmp[14]);
    FA fa16(ma[15], mb[15], tmp[14], sum[15], tmp[15]);
    FA fa17(ma[16], mb[16], tmp[15], sum[16], tmp[16]);
    FA fa18(ma[17], mb[17], tmp[16], sum[17], tmp[17]);
    FA fa19(ma[18], mb[18], tmp[17], sum[18], tmp[18]);

    FA fa20(ma[19], mb[19], tmp[19], sum[19], tmp[19]);
    FA fa21(ma[20], mb[20], tmp[17], sum[20], tmp[20]);
    FA fa22(ma[21], mb[21], tmp[20], sum[21], tmp[21]);
    FA fa23(ma[22], mb[22], tmp[21], sum[22], tmp[22]);
    FA fa24(ma[23], mb[23], tmp[22], sum[23], tmp[23]);
    FA fa25(ma[24], mb[24], tmp[23], sum[24], tmp[24]);

    FA fa26(ma[25], mb[25], tmp[24], sum[25], tmp[25]);
    FA fa27(ma[26], mb[26], tmp[25], sum[26], tmp[26]);
    FA fa28(ma[27], mb[27], tmp[26], sum[27], tmp[27]);
    FA fa29(ma[28], mb[28], tmp[27], sum[28], tmp[28]);
    FA fa30(ma[29], mb[29], tmp[28], sum[29], tmp[29]);
    FA fa31(ma[30], mb[30], tmp[29], sum[30], tmp[30]);

    FA fa32(ma[31], mb[31], tmp[30], sum[31], tmp[31]);
    FA fa33(ma[32], mb[32], tmp[31], sum[32], tmp[32]);
    FA fa34(ma[33], mb[33], tmp[32], sum[33], tmp[33]);
    FA fa35(ma[34], mb[34], tmp[33], sum[34], tmp[34]);
    FA fa36(ma[35], mb[35], tmp[34], sum[35], tmp[35]);
    FA fa37(ma[36], mb[36], tmp[35], sum[36], tmp[36]);

    FA fa38(ma[37], mb[37], tmp[36], sum[37], tmp[37]);
    FA fa39(ma[38], mb[38], tmp[37], sum[38], tmp[38]);
    FA fa40(ma[39], mb[39], tmp[38], sum[39], tmp[39]);
    FA fa41(ma[40], mb[40], tmp[39], sum[40], tmp[40]);
    FA fa42(ma[41], mb[41], tmp[40], sum[41], tmp[41]);
    FA fa43(ma[42], mb[42], tmp[41], sum[42], tmp[42]);

    FA fa44(ma[43], mb[43], tmp[42], sum[43], tmp[43]);
    FA fa45(ma[44], mb[44], tmp[43], sum[44], tmp[44]);
    FA fa46(ma[45], mb[45], tmp[44], sum[45], tmp[45]);
    FA fa47(ma[46], mb[46], tmp[45], sum[46], tmp[46]);
    FA fa48(ma[47], mb[47], tmp[46], sum[47], tmp[47]);
    FA fa49(ma[48], mb[48], tmp[47], sum[48], tmp[48]);

    FA fa50(ma[49], mb[49], tmp[48], sum[49], tmp[49]);
    FA fa51(ma[50], mb[50], tmp[49], sum[50], tmp[50]);
    FA fa52(ma[51], mb[15], tmp[50], sum[51], tmp[51]);
    FA fa53(ma[52], mb[52], tmp[51], sum[52], tmp[52]);
    FA fa54(ma[53], mb[53], tmp[52], sum[53], tmp[53]);
```

```
   assign out = sum;

endmodule

// Module to add mantissas
module SUM_MANT (sa, sb, ma, mb, out, carry);
   input sa, sb;
   input[52:0]  ma, mb;

   output  reg carry;
   wire[53:0] sum;
   output  reg [52:0] out;

   reg[53:0] tmp1, tmp2, tmp_sum;
   reg[52:0] tmp;

   always @(*) begin
      if(sa==sb) begin
         tmp1 = {1'b0, ma};
         tmp2 = {1'b0, mb};
      end

      if(sa==1 & sb==0) begin
         tmp = ~mb+1;
         tmp1 = {1'b0, ma};
         tmp2 = {1'b0, tmp};
      end

      if(sa==0 & sb==1) begin
```

```
            tmp = ~mb+1;
            tmp1 = {1'b0, ma};
            tmp2 = {1'b0, tmp};
        end
    end
    SAME_SIGN_ADD same_sign_add1(tmp1, tmp2, sum);

    always @(*) begin
        if(sa==sb) begin
            carry = sum[53];
            out = sum[52:0];
        end
        if((sa==1 & sb==0)|(sa==0 & sb==1))begin
            if(sum[53]==1'b0) begin
                carry = 0;
                out = ~sum[52:0]+1;
            end
            if(sum[53]==1'b1)begin
                carry = 0;
                out = sum[52:0];
            end
        end
    end
endmodule

// Module to normalize the result
module NORM_RES (sum, carry, er, sr, mr, er_new);
    input[52:0]  sum;
    input carry, sr;
```

```
input[10:0]  er;

output  reg[51:0] mr;

output  reg[10:0] er_new;


reg[52:0] tmp;


always @(*) begin

   if(carry==1) begin

      tmp  = sum >> 1;

      tmp[52]  = carry;

      er_new = er + 1;


      mr = tmp[51:0];

   end

   else begin

      if(sum[52]==1) begin

         mr = sum[51:0];

         er_new = er;

      end

      else begin

         tmp = sum;

         er_new = er;

         while(tmp[52]!=1'b1) begin

            tmp  = tmp << 1;

            er_new = er_new - 1;

         end


         mr = tmp[51:0];

      end
      end
   end
endmodule



// Main module
module FP_ADDER(a, b, out);
   input[63:0]  a, b;
   output[63:0]  out;



   wire sign_var, max;


   wire[63:0] new_a, new_b;        // Variable for new inputs that r used after swapping small
one to b
   wire sa, sb;                   // Variables for storing signs of original inputs
   wire[10:0] ea, eb;             // Variables for storing expo of original inputs
   wire[52:0] ma, mb;             // Variables for storing mantissa of original inputs


   // Declaring output indvivdual variables
   wire sr;
   wire[10:0] er;
   wire[51:0] mr;
   wire[52:0] sum_mantissa;
   wire carry;
   wire[10:0] er_new;


   wire[10:0] diff_exp;           // Variable to store difference of exponents
   wire[52:0] ma_new, mb_new, mb_sft;
   wire sa_new, sb_new;
   wire[10:0] ea_new, eb_new;

   // Assigning the components to inputs to individual variables. Added one before mantissa
that is 1 before decimal
   assign sa = a[63];
   assign ea = a[62: 52];
   assign ma = {1'b1,a[51:0]};
   assign sb = b[63];
   assign eb = b[62: 52];
   assign mb = {1'b1,b[51:0]};



   // Finding the max value among inputs
   CMP_EXP cmp_exp1(ea, eb, sign_var);

   // Swapping the inputs(if necessary) such that min is in b
   SWAP_M swap_m1(ma, mb, sa, sb, ea, eb, sign_var, ma_new, mb_new, sa_new, sb_new,
ea_new, eb_new);

   // Finding difference of exponents
   DIFF_EXP diff_exp1(ea, eb, diff_exp);

   // Shifting the smaller mantissa based on difference of exponents
   RHT_SFT rht_sft1(mb_new, diff_exp, mb_sft);

   // Assigning sign and exponent to results
   assign er = ea_new;
```

```
    assign sr = sa_new;



    // Adding the two mantissa
    SUM_MANT sum_mant1(sa_new, sb_new, ma_new, mb_sft, sum_mantissa, carry);
    // Here sum_mantissa contains 53 bits including the explicit 1 before the decimal

    // Normalizing the result
    NORM_RES norm_res1(sum_mantissa, carry, er, sr, mr, er_new);



    assign out = {sr, er_new, mr};

endmodule
```

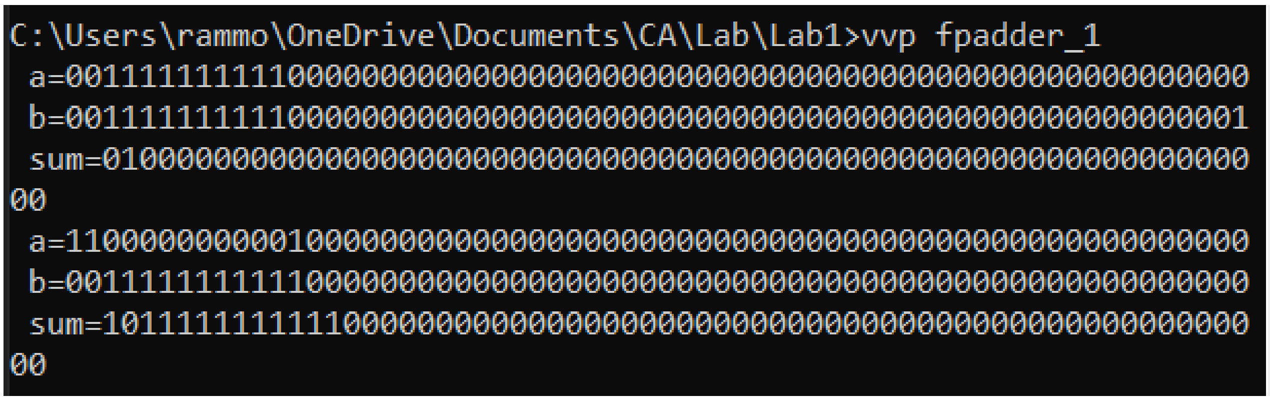
```
// Test bench
module FP_ADDER_TB;
    reg[63:0] a, b;

    wire[63:0] out;
    FP_ADDER fp_adder1(a, b, out);


    initial begin
        // a=1; b=1.0000000000000002
        #0
a=64'b0011111111110000000000000000000000000000000000000000000000000000;
b=64'b0011111111110000000000000000000000000000000000000000000000000001;


        // a=1; b=-3.5
        #10
a=64'b1100000000001000000000000000000000000000000000000000000000000000;
b=64'b0011111111111000000000000000000000000000000000000000000000000000;
    end


    initial begin
        $monitor(" a=%b b=%b sum=%b", a, b, out);
    end


endmodule
```

Output:

Given

i) a=1; b=1.0000000000000002

ii) a=1; b=-3.5



```
C:\Users\rammo\OneDrive\Documents\CA\Lab\Lab1>vvp fpadder_1
 a=0011111111110000000000000000000000000000000000000000000000000000
 b=0011111111110000000000000000000000000000000000000000000000000001
 sum=01000000000000000000000000000000000000000000000000000000000000
00
 a=1100000000001000000000000000000000000000000000000000000000000000
 b=0011111111111000000000000000000000000000000000000000000000000000
 sum=10111111111110000000000000000000000000000000000000000000000000
00
```