**float\_add.v**

module float\_add (

input [7:0] aIn,

input [7:0] bIn,

output reg [7:0] result

);

wire [7:0] big\_aOut, big\_bOut;

big\_number\_first step1 (

.aIn(aIn),

.bIn(bIn),

.aOut(big\_aOut),

.bOut(big\_bOut)

);

wire [2:0] aExp;

wire [4:0] aMan;

wire [2:0] bExp;

wire [4:0] bMan;

assign aExp = big\_aOut[7:5];

assign aMan = big\_aOut[4:0];

assign bExp = big\_bOut[7:5];

assign bMan = big\_bOut[4:0];

wire [2:0] distance1;

assign distance1 = aExp - bExp;

wire [2:0] bExpShift;

wire [4:0] bManShift;

assign bExpShift = bExp + distance1;

shifter step2 (

.in(bMan),

.distance(distance1),

.direction(1'b1),

.out(bManShift)

);

wire [4:0] adderSum;

wire adderCout;

adder step3 (

.A(aMan),

.B(bManShift),

.S(adderSum),

.Cout(adderCout)

);

wire [2:0] distance2;

assign distance2 = {2'b00, adderCout};

wire [2:0] resultExp;

wire [4:0] resultMan;

assign resultExp = bExpShift + adderCout;

shifter step4 (

.in(adderSum),

.distance(distance2),

.direction(1'b1),

.out(resultMan)

);

always @\* begin

if(bExpShift == 3'b111 && adderCout == 1'b1)

result = 8'b11111111;

else

begin

case(adderCout)

1'b0: result = {resultExp, resultMan};

1'b1: result = {resultExp, 1'b1, resultMan[3:0]};

endcase

end

end

endmodule

**big\_number\_first.v**

module big\_number\_first (

input [7:0] aIn,

input [7:0] bIn,

output reg [7:0] aOut,

output reg [7:0] bOut

);

wire [2:0] aExp;

wire [4:0] aMan;

wire [2:0] bExp;

wire [4:0] bMan;

assign aExp = aIn[7:5];

assign aMan = aIn[4:0];

assign bExp = bIn[7:5];

assign bMan = bIn[4:0];

always @\* begin

if((aExp < bExp) || (aExp == bExp && aMan < bMan))

begin

aOut = {bExp, bMan};

bOut = {aExp, aMan};

end

else

begin

aOut = {aExp, aMan};

bOut = {bExp, bMan};

end

end

endmodule

**shifter.v**

module shifter (

input [4:0] in,

input [2:0] distance,

input direction,

output reg [4:0] out

);

always @\* begin

case(direction)

1'b0: out = in << distance;

1'b1: out = in >> distance;

endcase

end

endmodule

**adder.v**

module adder(

input [4:0] A,

input [4:0] B,

output [4:0] S,

output Cout

);

wire Cin = 1'b0;

wire Cout0;

wire Cout1;

wire Cout2;

wire Cout3;

fullAdder a0 (

.A(A[0]),

.B(B[0]),

.Cin(Cin),

.Cout(Cout0),

.S(S[0])

);

fullAdder a1 (

.A(A[1]),

.B(B[1]),

.Cin(Cout0),

.Cout(Cout1),

.S(S[1])

);

fullAdder a2 (

.A(A[2]),

.B(B[2]),

.Cin(Cout1),

.Cout(Cout2),

.S(S[2])

);

fullAdder a3 (

.A(A[3]),

.B(B[3]),

.Cin(Cout2),

.Cout(Cout3),

.S(S[3])

);

fullAdder a4 (

.A(A[4]),

.B(B[4]),

.Cin(Cout3),

.Cout(Cout),

.S(S[4])

);

endmodule

**fullAdder.v**

module fullAdder (

input A,

input B,

input Cin,

output S,

output Cout

);

wire AxB;

assign AxB = A ^ B;

wire AxBaCin;

assign AxBaCin = (A ^ B) & Cin;

wire AaB;

assign AaB = A & B;

assign Cout = AaB | AxBaCin;

assign S = AxB ^ Cin;

endmodule

**float\_add\_tb.v**

module float\_add\_tb ();

reg [7:0] sim\_aIn;

reg [7:0] sim\_bIn;

wire [7:0] sim\_result;

float\_add test(

.aIn(sim\_aIn),

.bIn(sim\_bIn),

.result(sim\_result)

);

initial begin

sim\_aIn = 8'b00000000;

sim\_bIn = 8'b00000000;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b00000000);

sim\_aIn = 8'b00001000;

sim\_bIn = 8'b00000011;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b00001011);

sim\_aIn = 8'b00110001;

sim\_bIn = 8'b00001100;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b00110111);

sim\_aIn = 8'b10010010;

sim\_bIn = 8'b01011111;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b10011001);

sim\_aIn = 8'b00011110;

sim\_bIn = 8'b00011000;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b00111011);

sim\_aIn = 8'b11111110;

sim\_bIn = 8'b11111000;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b11111111);

sim\_aIn = 8'b11111111;

sim\_bIn = 8'b00111111;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b11111111);

sim\_aIn = 8'b11111110;

sim\_bIn = 8'b00111111;

#5

$display("Output is %b, we expected %b", sim\_result, 8'b11111110);

$stop;

end

endmodule

**big\_number\_first\_tb.v**

module big\_number\_first\_tb ();

reg [7:0] sim\_aIn, sim\_bIn;

wire [7:0] sim\_aOut, sim\_bOut;

big\_number\_first test (

.aIn(sim\_aIn),

.bIn(sim\_bIn),

.aOut(sim\_aOut),

.bOut(sim\_bOut)

);

initial begin

sim\_aIn = 8'b00001000;

sim\_bIn = 8'b00000011;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_aIn, sim\_bIn);

sim\_aIn = 8'b00000011;

sim\_bIn = 8'b00001000;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_bIn, sim\_aIn);

sim\_aIn = 8'b00110001;

sim\_bIn = 8'b00001100;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_aIn, sim\_bIn);

sim\_aIn = 8'b00001100;

sim\_bIn = 8'b00110001;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_bIn, sim\_aIn);

sim\_aIn = 8'b11111110;

sim\_bIn = 8'b11111111;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_bIn, sim\_aIn);

sim\_aIn = 8'b11111111;

sim\_bIn = 8'b11111110;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_aIn, sim\_bIn);

sim\_aIn = 8'b11011111;

sim\_bIn = 8'b11111111;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_bIn, sim\_aIn);

sim\_aIn = 8'b11111111;

sim\_bIn = 8'b11011111;

#5

$display("Output is %b and %b, we expected %b and %b", sim\_aOut, sim\_bOut, sim\_aIn, sim\_bIn);

$stop;

end

endmodule

**shifter\_tb.v**

module shifter\_tb ();

reg [4:0] sim\_in;

reg [2:0] sim\_distance;

reg sim\_direction;

wire [4:0] sim\_out;

shifter test (

.in(sim\_in),

.distance(sim\_distance),

.direction(sim\_direction),

.out(sim\_out)

);

initial begin

sim\_in = 5'b11001;

sim\_distance = 3'b000;

sim\_direction = 1'b0;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b11001);

sim\_in = 5'b11001;

sim\_distance = 3'b001;

sim\_direction = 1'b0;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b10010);

sim\_in = 5'b11001;

sim\_distance = 3'b010;

sim\_direction = 1'b0;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b00100);

sim\_in = 5'b11001;

sim\_distance = 3'b011;

sim\_direction = 1'b0;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b01000);

sim\_in = 5'b11001;

sim\_distance = 3'b100;

sim\_direction = 1'b0;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b10000);

sim\_in = 5'b11001;

sim\_distance = 3'b101;

sim\_direction = 1'b0;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b00000);

sim\_in = 5'b11001;

sim\_distance = 3'b000;

sim\_direction = 1'b1;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b11001);

sim\_in = 5'b11001;

sim\_distance = 3'b001;

sim\_direction = 1'b1;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b01100);

sim\_in = 5'b11001;

sim\_distance = 3'b010;

sim\_direction = 1'b1;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b00110);

**shifter\_tb.v (continued)**

sim\_in = 5'b11001;

sim\_distance = 3'b011;

sim\_direction = 1'b1;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b00011);

sim\_in = 5'b11001;

sim\_distance = 3'b100;

sim\_direction = 1'b1;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b00001);

sim\_in = 5'b11001;

sim\_distance = 3'b101;

sim\_direction = 1'b1;

#5

$display("Output is %b, we expected %b", sim\_out, 5'b00000);

$stop;

end

endmodule

**adder\_tb.v**

module adder\_tb ();

reg [4:0] sim\_A;

reg [4:0] sim\_B;

wire [4:0] sim\_S;

wire sim\_Cout;

adder test (

.A(sim\_A),

.B(sim\_B),

.Cout(sim\_Cout),

.S(sim\_S)

);

initial begin

sim\_A = 5'b0;

sim\_B = 5'b0;

#5;

$display("Sum is %b, we expected %b", sim\_S, 5'b0);

$display("Cout is %b, we expected %b", sim\_Cout, 1'b0);

sim\_A = 5'b00001;

sim\_B = 5'b00010;

#5;

$display("Sum is %b, we expected %b", sim\_S, 5'b00011);

$display("Cout is %b, we expected %b", sim\_Cout, 1'b0);

sim\_A = 5'b00111;

sim\_B = 5'b00111;

#5;

$display("Sum is %b, we expected %b", sim\_S, 5'b01110);

$display("Cout is %b, we expected %b", sim\_Cout, 1'b0);

sim\_A = 5'b10000;

sim\_B = 5'b11000;

#5;

$display("Sum is %b, we expected %b", sim\_S, 5'b01000);

$display("Cout is %b, we expected %b", sim\_Cout, 1'b1);

$stop;

end

endmodule