Mohammed Ali-Khan

1001722553

Maa2553

25 April 2025

CSE 5357 Advanced Digital Logic Design

Spring Semester 2025

Term Project

Half-Precision Floating-Point Adder Subtractor

Requirements Summary

This lab demonstrates the use of a 4x4 keyoad to input half-precision floating point numbers displayed in 16 bit binary display and adds the two numbers onto an LCD display.

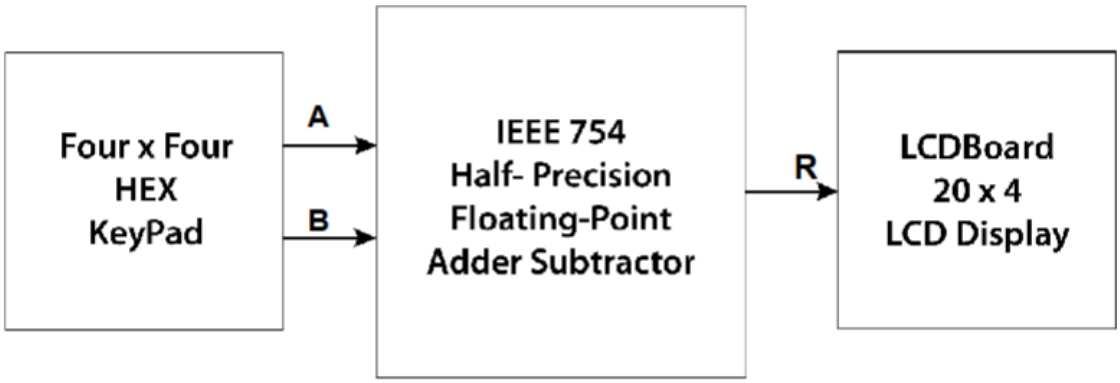


Figure 1 – Floating-Point Adder Subtractor with KeyPad and LCDBoard

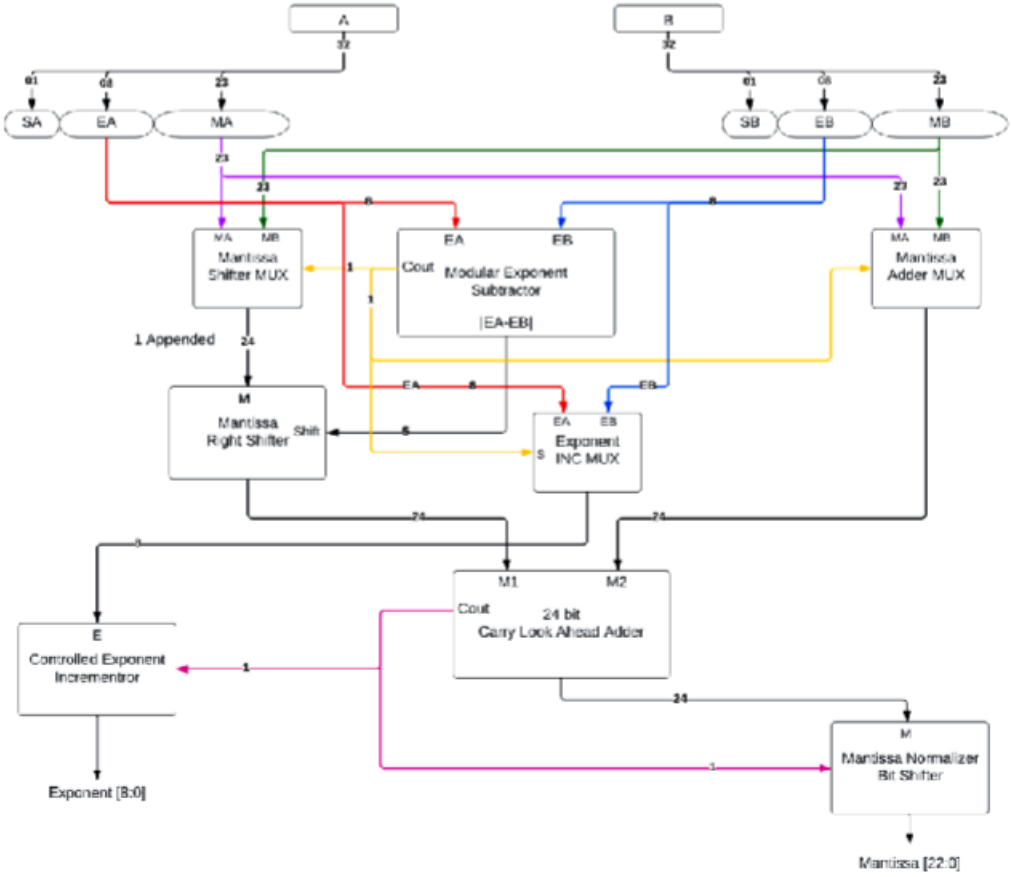
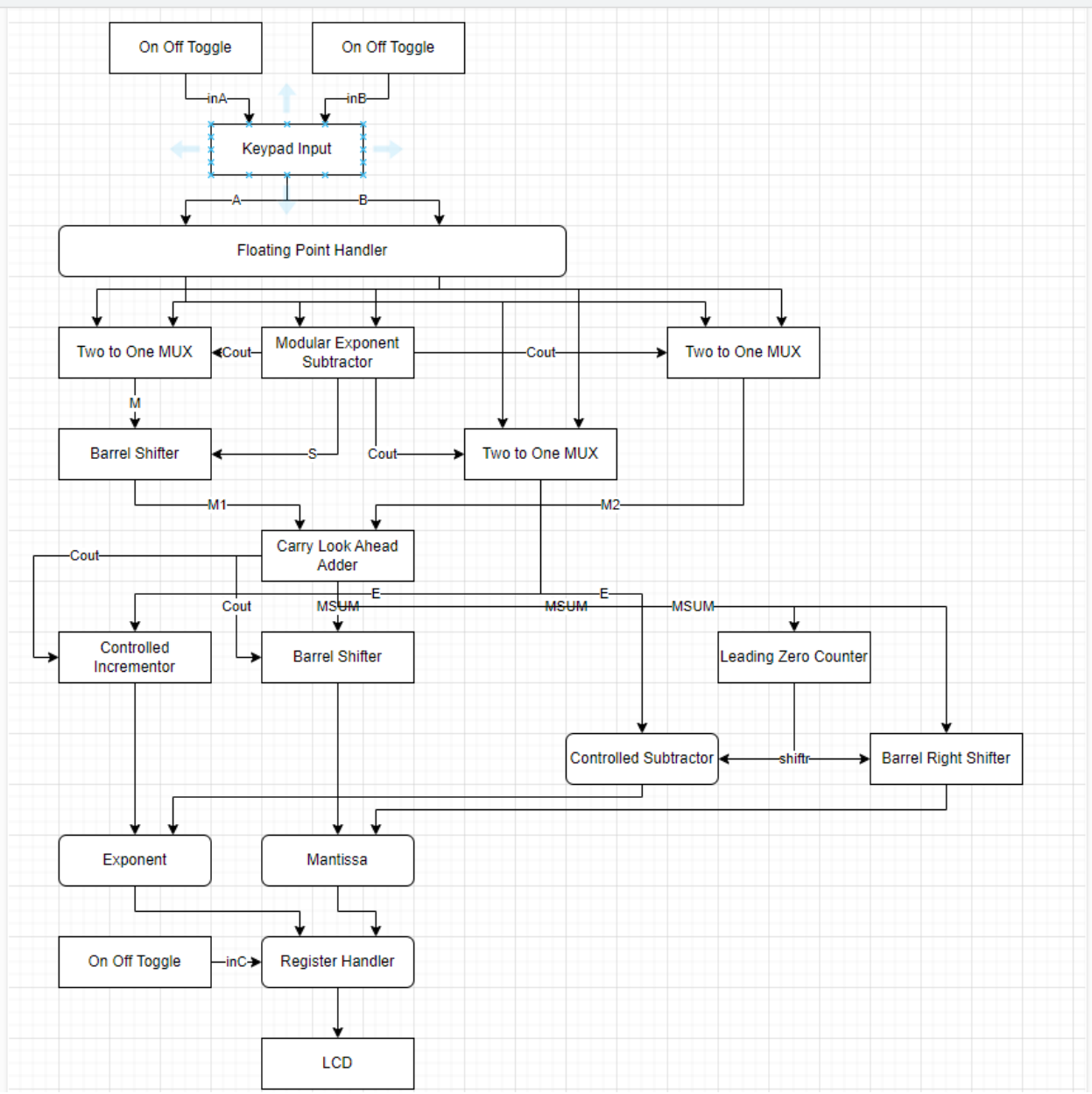


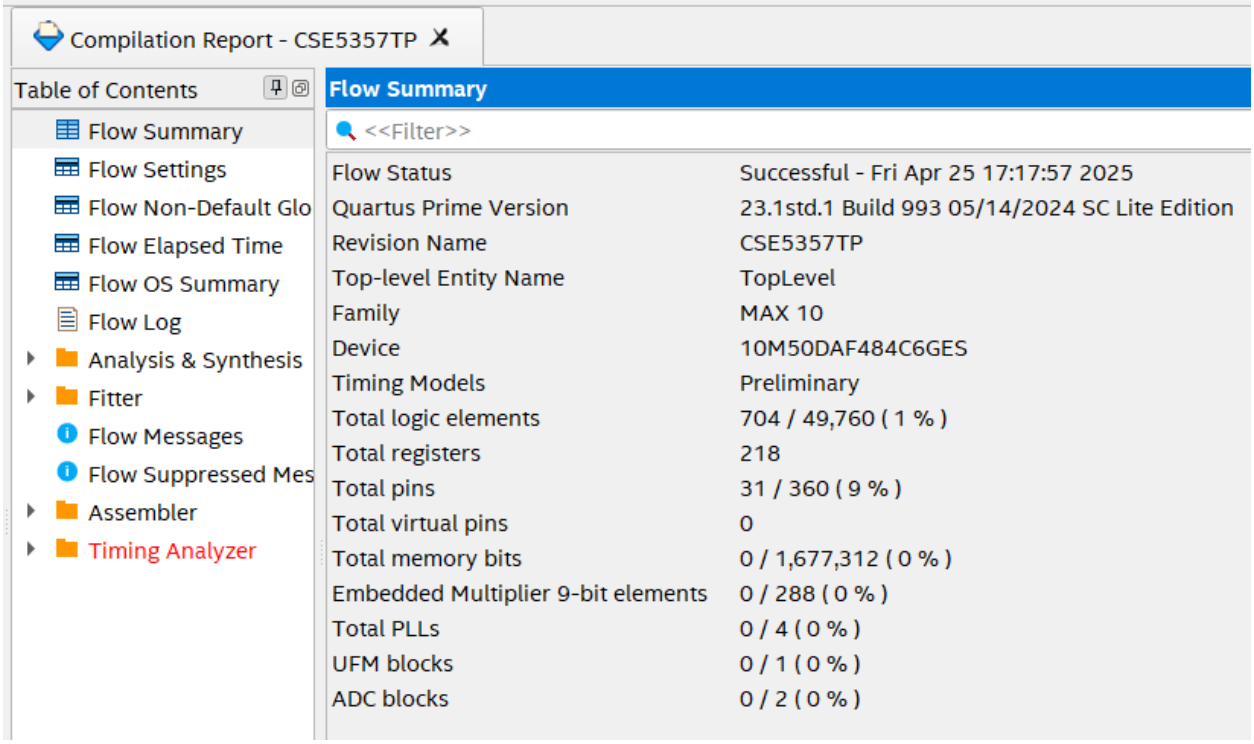
Figure 2 – Half-Precision Floating-Point Adder

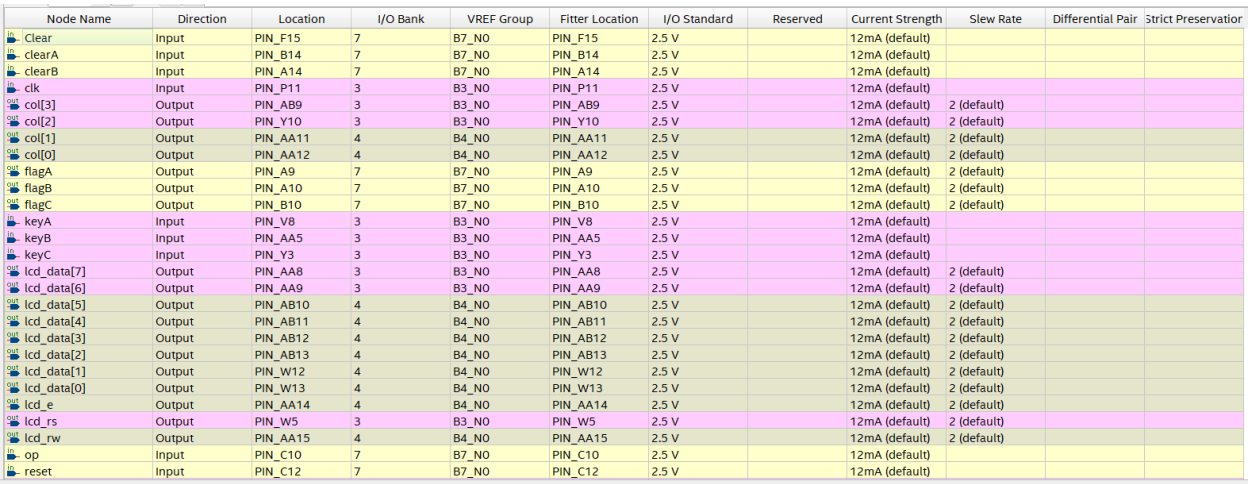
Organization Diagram



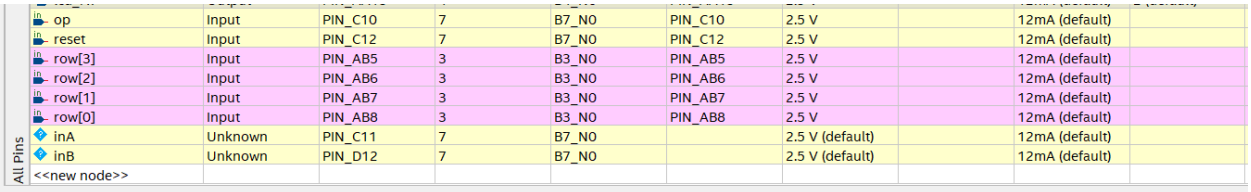
Test Results

Summary





Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
Clear	Input	PIN_F15	7	B7_N0	PIN_F15	2.5 V		12mA (default)			
clearA	Input	PIN_B14	7	B7_N0	PIN_B14	2.5 V		12mA (default)			
clearB	Input	PIN_A14	7	B7_N0	PIN_A14	2.5 V		12mA (default)			
clk	Input	PIN_P11	3	B3_N0	PIN_P11	2.5 V		12mA (default)			
col[3]	Output	PIN_AB9	3	B3_N0	PIN_AB9	2.5 V		12mA (default)	2 (default)		
col[2]	Output	PIN_Y10	3	B3_N0	PIN_Y10	2.5 V		12mA (default)	2 (default)		
col[1]	Output	PIN_AA11	4	B4_N0	PIN_AA11	2.5 V		12mA (default)	2 (default)		
col[0]	Output	PIN_AA12	4	B4_N0	PIN_AA12	2.5 V		12mA (default)	2 (default)		
flagA	Output	PIN_A9	7	B7_N0	PIN_A9	2.5 V		12mA (default)	2 (default)		
flagB	Output	PIN_A10	7	B7_N0	PIN_A10	2.5 V		12mA (default)	2 (default)		
flagC	Output	PIN_B10	7	B7_N0	PIN_B10	2.5 V		12mA (default)	2 (default)		
keyA	Input	PIN_V8	3	B3_N0	PIN_V8	2.5 V		12mA (default)			
keyB	Input	PIN_AA5	3	B3_N0	PIN_AA5	2.5 V		12mA (default)			
keyC	Input	PIN_Y3	3	B3_N0	PIN_Y3	2.5 V		12mA (default)			
lcd_data[7]	Output	PIN_AA8	3	B3_N0	PIN_AA8	2.5 V		12mA (default)	2 (default)		
lcd_data[6]	Output	PIN_AA9	3	B3_N0	PIN_AA9	2.5 V		12mA (default)	2 (default)		
lcd_data[5]	Output	PIN_AB10	4	B4_N0	PIN_AB10	2.5 V		12mA (default)	2 (default)		
lcd_data[4]	Output	PIN_AB11	4	B4_N0	PIN_AB11	2.5 V		12mA (default)	2 (default)		
lcd_data[3]	Output	PIN_AB12	4	B4_N0	PIN_AB12	2.5 V		12mA (default)	2 (default)		
lcd_data[2]	Output	PIN_AB13	4	B4_N0	PIN_AB13	2.5 V		12mA (default)	2 (default)		
lcd_data[1]	Output	PIN_W12	4	B4_N0	PIN_W12	2.5 V		12mA (default)	2 (default)		
lcd_data[0]	Output	PIN_W13	4	B4_N0	PIN_W13	2.5 V		12mA (default)	2 (default)		
lcd_e	Output	PIN_AA14	4	B4_N0	PIN_AA14	2.5 V		12mA (default)	2 (default)		
lcd_rs	Output	PIN_W5	3	B3_N0	PIN_W5	2.5 V		12mA (default)	2 (default)		
lcd_rw	Output	PIN_AA15	4	B4_N0	PIN_AA15	2.5 V		12mA (default)	2 (default)		
op	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V		12mA (default)			
reset	Input	PIN_C12	7	B7_N0	PIN_C12	2.5 V		12mA (default)			



op	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V		12mA (default)	
reset	Input	PIN_C12	7	B7_N0	PIN_C12	2.5 V		12mA (default)	
row[3]	Input	PIN_AB5	3	B3_N0	PIN_AB5	2.5 V		12mA (default)	
row[2]	Input	PIN_AB6	3	B3_N0	PIN_AB6	2.5 V		12mA (default)	
row[1]	Input	PIN_AB7	3	B3_N0	PIN_AB7	2.5 V		12mA (default)	
row[0]	Input	PIN_AB8	3	B3_N0	PIN_AB8	2.5 V		12mA (default)	
inA	Unknown	PIN_C11	7	B7_N0		2.5 V (default)		12mA (default)	
inB	Unknown	PIN_D12	7	B7_N0		2.5 V (default)		12mA (default)	
<<new node>>									

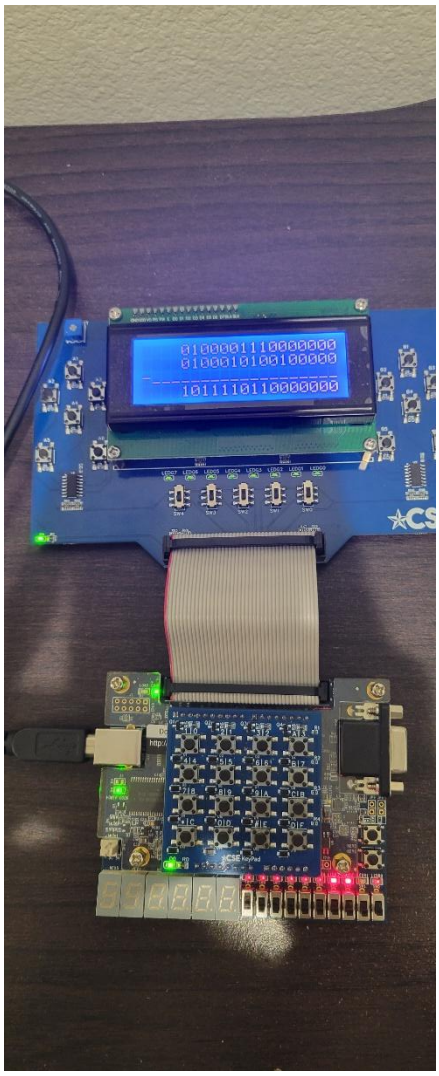
All Pins

Demonstration

The following link is to the demo video from my personal google drive. Cuts were made to facilitate keypad inputs (keypresses on same column are hard to register). Additionally, pictures have also been provided showcasing each test case.

https://drive.google.com/file/d/1WyDM4RBCj7r2rQ4dV-JmP8vEceatxAiJ/view?usp=sharing

1. 3.75 + 5.125
   a. 
2. 3.75 + -5.125
3. 3.75 – 5.125
   a. 

a. 

4. 9.5 + 0.5625

a. 

5. 125.125 + 12.0625

a. 

a. 

6. 1000 + 1049

b.

Verilog Code

Top Level shown below. Minor changes, if any, are made to other modules. LeadingZeroCounter and BarrelRightShifter added for handling subtraction. Changes made to CarryLookAheadAdder to allow for 11 bit addition.

CSE5357P > TopLevel.sv > ...

```
module TopLevel (
    input clk,
    input reset,
    input op,
    input Clear,
    input keyA, keyB, keyC,
    input clearA, clearB,
    input [3:0] row,
    output flagA, flagB, flagC,
    output [3:0] col,
    output [7:0] lcd_data,
    output lcd_rs,
    output lcd_rw,
    output lcd_e
);

    // input button handler
    logic inA, inB, inC;

    // input logic reg
    reg [15:0] A;
    reg [15:0] B;
    reg [15:0] C;
    wire [1:0] Operation;
    wire [15:0] keypad_out;
    wire [3:0] keypad_value;
    wire trig;

    // output logic reg
    reg [31:0] opA, opB;
    reg [4:0] Exponent;
    reg [10:0] Mantissa;

    // floating point internal reg
    reg SA, SB;
    reg [4:0] EA, EB;
    reg [9:0] MA, MB;

    // exponent sub reg
    reg [3:0] shift;
    reg S;

    // CLA adder reg
    reg [10:0] M, M1, M2, MSUM;
    reg[4:0] E;
    reg Cout;

    // reg for subtraction or negative
    reg [10:0] M3;
    reg [3:0] shiftr;
    reg [4:0] dec, exp;
    reg[10:0] minused, maxused;
```

CSE5357P > TopLevel.sv > ...

```
module TopLevel (
    // assign plus or minus
    assign Operation = op ? 2'b01 : 2'b10;

    // split floating point
    assign SA = A[15];
    assign SB = B[15] ^ ~op;
    assign EA = A[14:10];
    assign EB = B[14:10];
    assign MA = A[9:0];
    assign MB = B[9:0];

    // button handlers
    OnOffToggle input_a (
        .OnOff(keyA),
        .IN(1'b1),
        .OUT(inA)
    );

    OnOffToggle input_b (
        .OnOff(keyB),
        .IN(1'b1),
        .OUT(inB)
    );

    OnOffToggle input_c (
        .OnOff(keyC),
        .IN(1'b1),
        .OUT(inC)
    );

    // subtract exponents
    ModularExponentSubtractor #(.N(5)) modular_exponent_subtrac
        .A(EA),
        .B(EB),
        .Out(shift),
        .C(S)
    );

    // shift mantissa
    Two2OneMux #(.N(11)) mantissa_shifter_mux (
        .R({1'b1, MB}),
        .B({1'b1, MA}),
        .Cout(S),
        .Out(M)
    );

    // select higher mantissa
    Two2OneMux #(.N(11)) mantissa_adder_mux (
        .R({1'b1, MA}),
        .B({1'b1, MB}),
        .Cout(S),
        .Out(M2)
    );
```

CSE5357P > TopLevel.sv > ...

```
1   module TopLevel (
108     // select exponent
109     Two2OneMux #(.N(5)) exponent_inc_mux (
110         .R(EA),
111         .B(EB),
112         .Cout(S),
113         .Out(E)
114     );
115
116     // shift lower mantissa
117     BarrelShifter mantissa_right_shifter(
118         .A(M),
119         .B(shift[3:0]),
120         .Y(M1)
121     );
122
123     // handle negative addition
124     assign M3 = (SA ^ SB) ? ~M1 + 1 : M1;
125
126     // add mantissas
127     CarryLookAheadAdder CLAA(
128         .A(M3),
129         .B(M2),
130         .c_in(1'b0), // optional, usually 0
131         .S(MSUM),
132         .c_out(Cout)
133     );
134
135     // count leading zero for subtraction
136     LeadingZeroCounter #(.N(11)) zero_counter (
137         .A(MSUM),
138         .Count(shiftr)
139     );
140
141     // increment exponent
142     ControlledIncrementor controlled_exponent_incr
143         .A(E),
144         .Select(Cout),
145         .S(exp)
146     );
147
148     // adjust exponent for subtraction
149     assign dec = E - {1'b0, shiftr};
150
151     // adjust mantissa for addition
152     BarrelShifter mantissa_normalizer_bit_shifter(
153         .A(MSUM),
154         .B(Cout),
155         .Y(maxused)
156     );
157
```

CSE5357P > TopLevel.sv > ...

```
1   module TopLevel (
157
158     // adjust mantissa for subtraction
159     BarrelRightShifter mantissa_normalizer_right_shifter(
160         .A(MSUM),
161         .B(shiftr),
162         .Y(minused)
163     );
164
165     // output assignment
166     assign Exponent = (SA ^ SB) ? dec : exp;
167     assign Mantissa = (SA ^ SB) ? minused : maxused;
168
169     // input from keypad
170     KeypadInput #(.DIGITS(4)) keypad (
171         .clk(clk),
172         .reset(Clear),
173         .row(row),
174         .col(col),
175         .out(keypad_out),
176         .value(keypad_value),
177         .trig(trig)
178     );
179
180     // logic for input output reg
181     reg [15:0] regA, regB, regC;
182
183     always @(posedge clk) begin
184         if (reset) begin
185             regA <= 0;
186             regB <= 0;
187             regC <= 0;
188         end else begin
189             if (clearA)
190                 regA <= 0;
191             else if (inA)
192                 regA <= keypad_out;
193
194             if (clearB)
195                 regB <= 0;
196             else if (inB)
197                 regB <= keypad_out;
198
199             if (!inC)
200                 regC <= 0;
201             else if (inC)
202                 regC <= {(SA ^ SB), Exponent, Mantissa[9:0]};
203         end
204     end
205
206     assign A = regA;
207     assign B = regB;
208     assign C = regC;
```

```
209
210     assign flagA = inA;
211     assign flagB = inB;
212     assign flagC = inC;
213
214     // send data to LCD
215     LCD #(
216         .WIDTH(16),
217         .DIGITS(5),
218         .FLOAT(0),
219         .MODE(1),
220         .LINES(4),
221         .CHARS(20)
222     ) lcd_unit (
223         .clk(clk),
224         .lcd_data(lcd_data),
225         .lcd_rs(lcd_rs),
226         .lcd_rw(lcd_rw),
227         .lcd_e(lcd_e),
228         .lcd_reset(reset),
229         .A(A),
230         .B(B),
231         .C(C),
232         .Operation(Operation)
233     );
234
235 endmodule
236
```

CSE5357P > LeadingZeroCounter.sv > ...

```
1  module LeadingZeroCounter #(parameter N = 11)(
2      input wire [N-1:0] A,
3      output reg [3:0] Count
4  );
5      integer i;
6      reg found;
7
8      always @(*) begin
9          Count = 0;
10         found = 0;
11         for (i = N-1; i >= 0; i = i - 1) begin
12             if (!found && A[i] == 1'b1) begin
13                 Count = N - 1 - i;
14                 found = 1;
15             end
16         end
17     end
18 endmodule
19
```

CSE5357P > BarrelRightShifter.sv > BarrelRightShifter

```
1  module BarrelRightShifter(
2      input [10:0] A,
3      input [3:0] B,
4      output [10:0] Y
5  );
6
7      wire [10:0] shift8, shift4, shift2, shift1;
8
9      Two2OneMuxAB shiftEight (
10         .Sel(B[3]),
11         .A({A[2:0], 8'b0}),
12         .B(A),
13         .Out(shift8)
14     );
15
16     Two2OneMuxAB shiftFour (
17         .Sel(B[2]),
18         .A({shift8[4:0], 4'b0}),
19         .B(shift8),
20         .Out(shift4)
21     );
22
23     Two2OneMuxAB shiftTwo (
24         .Sel(B[1]),
25         .A({shift4[8:0], 2'b0}),
26         .B(shift4),
27         .Out(shift2)
28     );
29
30     Two2OneMuxAB shiftOne (
31         .Sel(B[0]),
32         .A({shift2[9:0], 1'b0}),
33         .B(shift2),
34         .Out(shift1)
35     );
36
37     assign Y = shift1;
38
39 endmodule
```

CSE5357P > CarryLookAheadAdder.sv > CarryLookAheadAdder

```
module CarryLookAheadAdder(
    input  logic [10:0] A,
    input  logic [10:0] B,
    input  logic        c_in,
    output logic [10:0] S,
    output logic        c_out
);

    wire [3:0] S0, S1;
    wire [2:0] S2;
    wire       c4, c8;
    wire       G0, P0, G1, P1, G2, P2;

    CLA_4bit cla0 (
        .A(A[3:0]), .B(B[3:0]), .c_in(c_in),
        .S(S0), .c_out(c4), .G_out(G0), .P_out(P0)
    );

    CLA_4bit cla1 (
        .A(A[7:4]), .B(B[7:4]), .c_in(c4),
        .S(S1), .c_out(c8), .G_out(G1), .P_out(P1)
    );

    CLA_3bit cla2 (
        .A(A[10:8]), .B(B[10:8]), .c_in(c8),
        .S(S2), .c_out(c_out), .G_out(G2), .P_out(P2)
    );

    assign S = {S2, S1, S0};

endmodule

```