

🔗 main ▾ [fpu / fadd / fadd.v](#)

[Go to file](#) [...](#)



Laplace-LT fadd修正

Latest commit cddf2d7 10 days ago [🕒 History](#)

👤 1 contributor

316 lines (294 sloc) | 10.5 KB

[Raw](#) [Blame](#) [🖨](#) [📄](#) [✎](#) [🗑](#)

```
1  `timescale 1us / 100ns
2  `default_nettype none
3
4  // module shift(
5  //     input wire [27:0] op,
6  //     input wire [7:0] shift,
7  //     output wire [27:0] result
8  // );
9  // wire [50:0] pre;
10 // assign pre = op >> shift;
11 // wire [27:0] shift_lt_26_result;
12 // assign shift_lt_26_result = {pre[50:24], |pre[23:0]};
13 // assign result = (shift > 8'd27) ? {27'd0, |op} : shift_lt_26_result;
14
15 // endmodule
16
17 module ZLC(
18     input wire [27:0] op,
19     output wire [4:0] out,
20     output wire [22:0] ans_shift_out
21 );
22 assign out = (op[27]) ? 5'd0 :
23             (op[26]) ? 5'd1 :
24             (op[25]) ? 5'd2 :
25             (op[24]) ? 5'd3 :
26             (op[23]) ? 5'd4 :
27             (op[22]) ? 5'd5 :
28             (op[21]) ? 5'd6 :
29             (op[20]) ? 5'd7 :
30             (op[19]) ? 5'd8 :
31             (op[18]) ? 5'd9 :
32             (op[17]) ? 5'd10 :
33             (op[16]) ? 5'd11 :
34             (op[15]) ? 5'd12 :
35             (op[14]) ? 5'd13 :
36             (op[13]) ? 5'd14 :
37             (op[12]) ? 5'd15 :
38             (op[11]) ? 5'd16 :
39             (op[10]) ? 5'd17 :
40             (op[9]) ? 5'd18 :
41             (op[8]) ? 5'd19 :
42             (op[7]) ? 5'd20 :
43             (op[6]) ? 5'd21 :
44             (op[5]) ? 5'd22 :
45             (op[4]) ? 5'd23 :
46             (op[3]) ? 5'd24 :
47             (op[2]) ? 5'd25 : 5'd28;
48 assign ans_shift_out = (op[27]) ? op[26:4] :
49             (op[26]) ? op[25:3] :
50             (op[25]) ? op[24:2] :
51             (op[24]) ? op[23:1] :
52             (op[23]) ? op[22:0] :
53             (op[22]) ? {op[21:0], 1'b0} :
54             (op[21]) ? {op[20:0], 2'b0} :
55             (op[20]) ? {op[19:0], 3'b0} :
56             (op[19]) ? {op[18:0], 4'b0} :
57             (op[18]) ? {op[17:0], 5'b0} :
58             (op[17]) ? {op[16:0], 6'b0} :
59             (op[16]) ? {op[15:0], 7'b0} :
60             (op[15]) ? {op[14:0], 8'b0} :
61             (op[14]) ? {op[13:0], 9'b0} :
62             (op[13]) ? {op[12:0], 10'b0} :
63             (op[12]) ? {op[11:0], 11'b0} :
64             (op[11]) ? {op[10:0], 12'b0} :
65             (op[10]) ? {op[9:0], 13'b0} :
```

```

66         (op[9]) ? {op[8:0], 14'b0} :
67         (op[8]) ? {op[7:0], 15'b0} :
68         (op[7]) ? {op[6:0], 16'b0} :
69         (op[6]) ? {op[5:0], 17'b0}:
70         (op[5]) ? {op[4:0], 18'b0} :
71         (op[4]) ? {op[3:0], 19'b0} :
72         (op[3]) ? {op[2:0], 20'b0} :
73         (op[2]) ? {op[1:0], 21'b0} : 23'd0;
74
75     endmodule
76
77     module fadd(
78         input wire [31:0] op1,
79         input wire [31:0] op2,
80         output reg [31:0] result,
81         input wire clk,
82         // output reg ready,
83         // output reg valid,//実質アンダーフロー検知
84         input wire reset
85     );
86
87     wire sig1;
88     wire sig2;
89     wire [7:0] exp1;
90     wire [7:0] exp2;
91     wire [27:0] fra1;
92     wire [27:0] fra2;
93     assign sig1 = op1[31];
94     assign sig2 = op2[31];
95     assign exp1 = op1[30:23];
96     assign exp2 = op2[30:23];
97     assign fra1 = (exp1 == 8'b0) ? {2'b00, op1[22:0], 3'b000} : {2'b01, op1[22:0], 3'b000};
98     assign fra2 = (exp2 == 8'b0) ? {2'b00, op2[22:0], 3'b000} : {2'b01, op2[22:0], 3'b000};
99
100    wire op1_is_abs_bigger;
101    assign op1_is_abs_bigger = (exp1 == exp2) ? (op1[22:0] > op2[22:0]) : (exp1 > exp2);
102
103    wire [7:0] shift_1;
104    wire [7:0] shift_2;
105    assign shift_1 = exp2 - exp1;//if op2 is bigger
106    assign shift_2 = exp1 - exp2;//if op1 is bigger
107
108    // wire [27:0] fra1_shifted;
109    // shift shift_mod_1(fra1, shift_1, fra1_shifed);
110    // wire [27:0] fra2_shifted;
111    // shift shift_mod_2(fra2, shift_2, fra2_shifted);
112
113    reg [27:0] op_big;
114    reg [27:0] op_small;
115    reg [7:0] exp_big;
116    reg sig_big;
117    reg sig_small;
118
119    wire [27:0] ans;
120    assign ans = (sig_big ^ sig_small) ? (op_big - op_small) : (op_big + op_small);
121    reg [27:0] ans_reg;
122    wire [4:0] zero_count;
123    wire [22:0] ans_shift;
124    reg [22:0] ans_shift_reg;
125    ZLC ZLC1(ans, zero_count, ans_shift);
126    wire marume_up;
127    assign marume_up = (~ans[27] && (ans[26] || ans[1]) && &ans[25:2]);
128
129    reg [7:0] exp_next;
130    reg sig_next;
131    reg [4:0] zero_count_reg;
132
133    wire [8:0] exp_next_zero;
134    assign exp_next_zero = {1'b0, exp_next};
135
136    wire [7:0] for_exp_next;
137    assign for_exp_next = {7'd0, marume_up};
138
139    wire [22:0] for_ZLC0_fra;
140    assign for_ZLC0_fra = {22'd0, |ans_reg[3:0]};
141    wire [22:0] ZLC0_fra;
142    assign ZLC0_fra = ans_shift_reg + for_ZLC0_fra;
143    wire [7:0] ZLC0_exp;
144    assign ZLC0_exp = exp_next + 8'd1;
145
146    wire [22:0] for_ZLC1_fra;
147    assign for_ZLC1_fra = {22'd0, |ans_reg[2:0]};
148    wire [22:0] ZLC1_fra;

```

```

149 assign ZLC1_fra = ans_shift_reg + for_ZLC1_fra;
150 wire [7:0] ZLC1_exp;
151 assign ZLC1_exp = exp_next;
152
153 wire [22:0] for_ZLC2_fra;
154 assign for_ZLC2_fra = {22'd0, |ans_reg[1:0]};
155 wire [22:0] ZLC2_fra;
156 assign ZLC2_fra = ans_shift_reg + for_ZLC2_fra;
157 wire [8:0] ZLC2_exp;
158 assign ZLC2_exp = exp_next_zero - 9'd1;
159
160 wire [22:0] for_ZLC3_fra;
161 assign for_ZLC3_fra = {22'd0, ans_reg[0]};
162 wire [22:0] ZLC3_fra;
163 assign ZLC3_fra = ans_shift_reg + for_ZLC3_fra;
164 wire [8:0] ZLC3_exp;
165 assign ZLC3_exp = exp_next_zero - 9'd2;
166
167 wire [22:0] ZLC_lt3_fra;
168 assign ZLC_lt3_fra = ans_shift_reg;
169 wire [8:0] for_ZLC_lt3_exp;
170 assign for_ZLC_lt3_exp = {4'd0, zero_count_reg};
171 wire [8:0] for2_ZLC_lt3_exp;
172 assign for2_ZLC_lt3_exp = {8'd0, 1'b1};
173 wire [8:0] ZLC_lt3_exp;
174 assign ZLC_lt3_exp = exp_next_zero - for_ZLC_lt3_exp + for2_ZLC_lt3_exp;
175
176
177 always @(posedge clk) begin
178     if (~reset) begin
179         result <= 32'd0;
180         // ready <= 1'b0;
181         // valid <= 1'b0;
182         op_big <= 28'd0;
183         op_small <= 28'd0;
184         exp_big <= 8'd0;
185         sig_big <= 1'b0;
186         sig_small <= 1'b0;
187         exp_next <= 8'b0;
188         sig_next <= 1'b0;
189         zero_count_reg <= 5'd0;
190     end else begin
191         if (op1_is_abs_bigger) begin
192             op_big <= fra1;
193             // op_small <= fra2_shifted;
194             exp_big <= exp1;
195             sig_big <= sig1;
196             sig_small <= sig2;
197             case (shift_2)
198                 8'd0 : op_small <= fra2;
199                 8'd1 : op_small <= fra2 >> 1;
200                 8'd2 : op_small <= fra2 >> 2;
201                 8'd3 : op_small <= fra2 >> 3;
202                 8'd4 : op_small <= fra2 >> 4;
203                 8'd5 : op_small <= fra2 >> 5;
204                 8'd6 : op_small <= fra2 >> 6;
205                 8'd7 : op_small <= fra2 >> 7;
206                 8'd8 : op_small <= fra2 >> 8;
207                 8'd9 : op_small <= fra2 >> 9;
208                 8'd10 : op_small <= fra2 >> 10;
209                 8'd11 : op_small <= fra2 >> 11;
210                 8'd12 : op_small <= fra2 >> 12;
211                 8'd13 : op_small <= fra2 >> 13;
212                 8'd14 : op_small <= fra2 >> 14;
213                 8'd15 : op_small <= fra2 >> 15;
214                 8'd16 : op_small <= fra2 >> 16;
215                 8'd17 : op_small <= fra2 >> 17;
216                 8'd18 : op_small <= fra2 >> 18;
217                 8'd19 : op_small <= fra2 >> 19;
218                 8'd20 : op_small <= fra2 >> 20;
219                 8'd21 : op_small <= fra2 >> 21;
220                 8'd22 : op_small <= fra2 >> 22;
221                 8'd23 : op_small <= fra2 >> 23;
222                 8'd24 : op_small <= fra2 >> 24;
223                 8'd25 : op_small <= fra2 >> 25;
224                 8'd26 : op_small <= fra2 >> 26;
225                 default : op_small <= {27'd0, |fra2};
226             endcase
227         end else begin
228             op_big <= fra2;
229             // op_small <= fra1_shifted;
230             exp_big <= exp2;
231             sig_big <= sig2;

```

```

232     sig_small <= sig1;
233     case (shift_1)
234         8'd0 : op_small <= fra1;
235         8'd1 : op_small <= fra1 >> 1;
236         8'd2 : op_small <= fra1 >> 2;
237         8'd3 : op_small <= fra1 >> 3;
238         8'd4 : op_small <= fra1 >> 4;
239         8'd5 : op_small <= fra1 >> 5;
240         8'd6 : op_small <= fra1 >> 6;
241         8'd7 : op_small <= fra1 >> 7;
242         8'd8 : op_small <= fra1 >> 8;
243         8'd9 : op_small <= fra1 >> 9;
244         8'd10 : op_small <= fra1 >> 10;
245         8'd11 : op_small <= fra1 >> 11;
246         8'd12 : op_small <= fra1 >> 12;
247         8'd13 : op_small <= fra1 >> 13;
248         8'd14 : op_small <= fra1 >> 14;
249         8'd15 : op_small <= fra1 >> 15;
250         8'd16 : op_small <= fra1 >> 16;
251         8'd17 : op_small <= fra1 >> 17;
252         8'd18 : op_small <= fra1 >> 18;
253         8'd19 : op_small <= fra1 >> 19;
254         8'd20 : op_small <= fra1 >> 20;
255         8'd21 : op_small <= fra1 >> 21;
256         8'd22 : op_small <= fra1 >> 22;
257         8'd23 : op_small <= fra1 >> 23;
258         8'd24 : op_small <= fra1 >> 24;
259         8'd25 : op_small <= fra1 >> 25;
260         8'd26 : op_small <= fra1 >> 26;
261         default : op_small <= {27'd0, |fra1};
262     endcase
263 end
264 ans_reg <= ans;
265 ans_shift_reg <= ans_shift;
266 exp_next <= (exp_big + for_exp_next);
267 sig_next <= sig_big;
268 zero_count_reg <= zero_count;
269 // if (ready) begin
270 //     ready <= 1'b0;
271 //     valid <= 1'b0;
272 // end
273 if (zero_count_reg == 5'd0) begin
274     result <= {sig_next, ZLC0_exp, ZLC0_fra};
275     // ready <= 1'b1;
276     // valid <= 1'b1;
277 end else if (zero_count_reg == 5'd1) begin
278     result <= {sig_next, ZLC1_exp, ZLC1_fra};
279     // ready <= 1'b1;
280     // valid <= 1'b1;
281 end else if (zero_count_reg == 5'd2) begin
282     if (ZLC2_exp[8]) begin
283         result <= {sig_next, 8'd0, ZLC2_fra}; //このfraに意味はない
284         // ready <= 1'b1;
285         // valid <= 1'b0;
286     end else begin
287         result <= {sig_next, ZLC2_exp[7:0], ZLC2_fra};
288         // ready <= 1'b1;
289         // valid <= 1'b1;
290     end
291 end else if (zero_count_reg == 5'd3) begin
292     if (ZLC3_exp[8]) begin
293         result <= {sig_next, 8'd0, ZLC3_fra};
294         // ready <= 1'b1;
295         // valid <= 1'b0;
296     end else begin
297         result <= {sig_next, ZLC3_exp[7:0], ZLC3_fra};
298         // ready <= 1'b1;
299         // valid <= 1'b1;
300     end
301 end else begin
302     if (ZLC_lt3_exp[8]) begin
303         result <= {sig_next, 8'd0, ZLC3_fra};
304         // ready <= 1'b1;
305         // valid <= 1'b0;
306     end else begin
307         result <= {sig_next, ZLC_lt3_exp[7:0], ZLC_lt3_fra};
308         // ready <= 1'b1;
309         // valid <= 1'b1;
310     end
311 end
312 end
313 end
314

```

```
315 endmodule
316 `default_nettype wire
```