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} :
```

```
                (op[9]) ? {op[8:0], 14'b0} :
                (op[8]) ? {op[7:0], 15'b0} :
                (op[7]) ? {op[6:0], 16'b0} :
                (op[6]) ? {op[5:0], 17'b0}:
                (op[5]) ? {op[4:0], 18'b0} :
                (op[4]) ? {op[3:0], 19'b0} :
                (op[3]) ? {op[2:0], 20'b0} :
                (op[2]) ? {op[1:0], 21'b0} : 23'd0;

endmodule

module fadd(
    input wire [31:0] op1,
    input wire [31:0] op2,
    output reg [31:0] result,
    input wire clk,
    // output reg ready,
    // output reg valid,//実質アンダーフロー検知
    input wire reset
);

wire sig1;
wire sig2;
wire [7:0] exp1;
wire [7:0] exp2;
wire [27:0] fra1;
wire [27:0] fra2;
assign sig1 = op1[31];
assign sig2 = op2[31];
assign exp1 = op1[30:23];
assign exp2 = op2[30:23];
assign fra1 = (exp1 == 8'b0) ? {2'b00, op1[22:0], 3'b000} : {2'b01, op1[22:0], 3'b000};
assign fra2 = (exp2 == 8'b0) ? {2'b00, op2[22:0], 3'b000} : {2'b01, op2[22:0], 3'b000};

wire op1_is_abs_bigger;
assign op1_is_abs_bigger = (exp1 == exp2) ? (op1[22:0] > op2[22:0]) : (exp1 > exp2);

wire [7:0] shift_1;
wire [7:0] shift_2;
assign shift_1 = exp2 - exp1;//if op2 is bigger
assign shift_2 = exp1 - exp2;//if op1 is bigger

// wire [27:0] fra1_shifted;
// shift shift_mod_1(fra1, shift_1, fra1_shifed);
// wire [27:0] fra2_shifted;
// shift shift_mod_2(fra2, shift_2, fra2_shifted);

reg [27:0] op_big;
reg [27:0] op_small;
reg [7:0] exp_big;
reg sig_big;
reg sig_small;

wire [27:0] ans;
assign ans = (sig_big ^ sig_small) ? (op_big - op_small) : (op_big + op_small);
reg [27:0] ans_reg;
wire [4:0] zero_count;
wire [22:0] ans_shift;
reg [22:0] ans_shift_reg;
ZLC ZLC1(ans, zero_count, ans_shift);
wire marume_up;
assign marume_up = (~ans[27] && (ans[26] || ans[1]) && &ans[25:2]);

reg [7:0] exp_next;
reg sig_next;
reg [4:0] zero_count_reg;

wire [8:0] exp_next_zero;
assign exp_next_zero = {1'b0, exp_next};

wire [7:0] for_exp_next;
assign for_exp_next = {7'd0, marume_up};

wire [22:0] for_ZLC0_fra;
assign for_ZLC0_fra = {22'd0, |ans_reg[3:0]};
wire [22:0] ZLC0_fra;
assign ZLC0_fra = ans_shift_reg + for_ZLC0_fra;
wire [7:0] ZLC0_exp;
assign ZLC0_exp = exp_next + 8'd1;

wire [22:0] for_ZLC1_fra;
assign for_ZLC1_fra = {22'd0, |ans_reg[2:0]};
wire [22:0] ZLC1_fra;
```

```
assign ZLC1_fra = ans_shift_reg + for_ZLC1_fra;
wire [7:0] ZLC1_exp;
assign ZLC1_exp = exp_next;

wire [22:0] for_ZLC2_fra;
assign for_ZLC2_fra = {22'd0, |ans_reg[1:0]};
wire [22:0] ZLC2_fra;
assign ZLC2_fra = ans_shift_reg + for_ZLC2_fra;
wire [8:0] ZLC2_exp;
assign ZLC2_exp = exp_next_zero - 9'd1;

wire [22:0] for_ZLC3_fra;
assign for_ZLC3_fra = {22'd0, ans_reg[0]};
wire [22:0] ZLC3_fra;
assign ZLC3_fra = ans_shift_reg + for_ZLC3_fra;
wire [8:0] ZLC3_exp;
assign ZLC3_exp = exp_next_zero - 9'd2;

wire [22:0] ZLC_lt3_fra;
assign ZLC_lt3_fra = ans_shift_reg;
wire [8:0] for_ZLC_lt3_exp;
assign for_ZLC_lt3_exp = {4'd0, zero_count_reg};
wire [8:0] for2_ZLC_lt3_exp;
assign for2_ZLC_lt3_exp = {8'd0, 1'b1};
wire [8:0] ZLC_lt3_exp;
assign ZLC_lt3_exp = exp_next_zero - for_ZLC_lt3_exp + for2_ZLC_lt3_exp;


always @(posedge clk) begin
    if (~reset) begin
        result <= 32'd0;
        // ready <= 1'b0;
        // valid <= 1'b0;
        op_big <= 28'd0;
        op_small <= 28'd0;
        exp_big <= 8'd0;
        sig_big <= 1'b0;
        sig_small <= 1'b0;
        exp_next <= 8'b0;
        sig_next <= 1'b0;
        zero_count_reg <= 5'd0;
    end else begin
        if (op1_is_abs_bigger) begin
            op_big <= fra1;
            // op_small <= fra2_shifted;
            exp_big <= exp1;
            sig_big <= sig1;
            sig_small <= sig2;
            case (shift_2)
                8'd0 : op_small <= fra2;
                8'd1 : op_small <= fra2 >> 1;
                8'd2 : op_small <= fra2 >> 2;
                8'd3 : op_small <= fra2 >> 3;
                8'd4 : op_small <= fra2 >> 4;
                8'd5 : op_small <= fra2 >> 5;
                8'd6 : op_small <= fra2 >> 6;
                8'd7 : op_small <= fra2 >> 7;
                8'd8 : op_small <= fra2 >> 8;
                8'd9 : op_small <= fra2 >> 9;
                8'd10 : op_small <= fra2 >> 10;
                8'd11 : op_small <= fra2 >> 11;
                8'd12 : op_small <= fra2 >> 12;
                8'd13 : op_small <= fra2 >> 13;
                8'd14 : op_small <= fra2 >> 14;
                8'd15 : op_small <= fra2 >> 15;
                8'd16 : op_small <= fra2 >> 16;
                8'd17 : op_small <= fra2 >> 17;
                8'd18 : op_small <= fra2 >> 18;
                8'd19 : op_small <= fra2 >> 19;
                8'd20 : op_small <= fra2 >> 20;
                8'd21 : op_small <= fra2 >> 21;
                8'd22 : op_small <= fra2 >> 22;
                8'd23 : op_small <= fra2 >> 23;
                8'd24 : op_small <= fra2 >> 24;
                8'd25 : op_small <= fra2 >> 25;
                8'd26 : op_small <= fra2 >> 26;
                default : op_small <= {27'd0, |fra2};
            endcase
        end else begin
            op_big <= fra2;
            // op_small <= fra1_shifted;
            exp_big <= exp2;
            sig_big <= sig2;
```

```
            sig_small <= sig1;
            case (shift_1)
                8'd0 : op_small <= fra1;
                8'd1 : op_small <= fra1 >> 1;
                8'd2 : op_small <= fra1 >> 2;
                8'd3 : op_small <= fra1 >> 3;
                8'd4 : op_small <= fra1 >> 4;
                8'd5 : op_small <= fra1 >> 5;
                8'd6 : op_small <= fra1 >> 6;
                8'd7 : op_small <= fra1 >> 7;
                8'd8 : op_small <= fra1 >> 8;
                8'd9 : op_small <= fra1 >> 9;
                8'd10 : op_small <= fra1 >> 10;
                8'd11 : op_small <= fra1 >> 11;
                8'd12 : op_small <= fra1 >> 12;
                8'd13 : op_small <= fra1 >> 13;
                8'd14 : op_small <= fra1 >> 14;
                8'd15 : op_small <= fra1 >> 15;
                8'd16 : op_small <= fra1 >> 16;
                8'd17 : op_small <= fra1 >> 17;
                8'd18 : op_small <= fra1 >> 18;
                8'd19 : op_small <= fra1 >> 19;
                8'd20 : op_small <= fra1 >> 20;
                8'd21 : op_small <= fra1 >> 21;
                8'd22 : op_small <= fra1 >> 22;
                8'd23 : op_small <= fra1 >> 23;
                8'd24 : op_small <= fra1 >> 24;
                8'd25 : op_small <= fra1 >> 25;
                8'd26 : op_small <= fra1 >> 26;
                default : op_small <= {27'd0, |fra1};
            endcase
        end
        ans_reg <= ans;
        ans_shift_reg <= ans_shift;
        exp_next <= (exp_big + for_exp_next);
        sig_next <= sig_big;
        zero_count_reg <= zero_count;
        // if (ready) begin
        //     ready <= 1'b0;
        //     valid <= 1'b0;
        // end
        if (zero_count_reg == 5'd0) begin
            result <= {sig_next, ZLC0_exp, ZLC0_fra};
            // ready <= 1'b1;
            // valid <= 1'b1;
        end else if (zero_count_reg == 5'd1) begin
            result <= {sig_next, ZLC1_exp, ZLC1_fra};
            // ready <= 1'b1;
            // valid <= 1'b1;
        end else if (zero_count_reg == 5'd2) begin
            if (ZLC2_exp[8]) begin
                result <= {sig_next, 8'd0, ZLC2_fra};//ここのfraに意味はない
                // ready <= 1'b1;
                // valid <= 1'b0;
            end else begin
                result <= {sig_next, ZLC2_exp[7:0], ZLC2_fra};
                // ready <= 1'b1;
                // valid <= 1'b1;
            end
        end else if (zero_count_reg == 5'd3) begin
            if (ZLC3_exp[8]) begin
                result <= {sig_next, 8'd0, ZLC3_fra};
                // ready <= 1'b1;
                // valid <= 1'b0;
            end else begin
                result <= {sig_next, ZLC3_exp[7:0], ZLC3_fra};
                // ready <= 1'b1;
                // valid <= 1'b1;
            end
        end else begin
            if (ZLC_lt3_exp[8]) begin
                result <= {sig_next, 8'd0, ZLC3_fra};
                // ready <= 1'b1;
                // valid <= 1'b0;
            end else begin
                result <= {sig_next, ZLC_lt3_exp[7:0], ZLC_lt3_fra};
                // ready <= 1'b1;
                // valid <= 1'b1;
            end
        end
    end
end

```

```
endmodule
`default_nettype wire
```