```verilog
`timescale 1ns / 1ps

module fpu_addsub (
    input  wire        clk,
    input  wire        rst,
    input  wire        op,              // 0 = add, 1 = subtract
    input  wire        sign_a,
    input  wire [10:0] exp_a,
    input  wire [52:0] mant_a,          // includes implicit 1 (total 53 bits)
    input  wire        sign_b,
    input  wire [10:0] exp_b,
    input  wire [52:0] mant_b,          // includes implicit 1 (total 53 bits)
    output reg         result_sign,
    output reg  [10:0] result_exp,
    output reg  [52:0] result_mant,
    output reg         ready
);

    // Internal variables
    reg [10:0] exp_diff;
    reg [52:0] aligned_mant_a, aligned_mant_b;
    reg [53:0] mant_sum;
    reg [53:0] mant_diff;
    reg        sign_b_eff;
    reg [10:0] exp_max;
    reg [5:0]  shift_amt;
    reg [53:0] mant_norm;
    integer    i;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            result_sign <= 0;
            result_exp  <= 0;
            result_mant <= 0;
            ready       <= 0;
        end else begin
            ready <= 0;

            // Effective sign for operand B (flip if subtract)
            sign_b_eff = (op) ? ~sign_b : sign_b;

            // Exponent alignment
            if (exp_a > exp_b) begin
                exp_diff       = exp_a - exp_b;
                aligned_mant_a = mant_a;
                aligned_mant_b = mant_b >> exp_diff;
            exp_max        = exp_a;
        end else begin
            exp_diff       = exp_b - exp_a;
            aligned_mant_a = mant_a >> exp_diff;
            aligned_mant_b = mant_b;
            exp_max        = exp_b;
        end

        // ADDITION (same signs)
        if (sign_a == sign_b_eff) begin
            mant_sum = {1'b0, aligned_mant_a} + {1'b0, aligned_mant_b};
            if (mant_sum[53]) begin
                result_mant = mant_sum[53:1];
                result_exp  = exp_max + 1;
            end else begin
                result_mant = mant_sum[52:0];
                result_exp  = exp_max;
            end
            result_sign = sign_a;
        end

        // SUBTRACTION (different signs)
        else begin
            if (aligned_mant_a >= aligned_mant_b) begin
                mant_diff   = {1'b0, aligned_mant_a} - {1'b0, aligned_mant_b};
                result_sign = sign_a;
            end else begin
                mant_diff   = {1'b0, aligned_mant_b} - {1'b0, aligned_mant_a};
                result_sign = sign_b_eff;
            end

            // Normalize the result
            mant_norm = mant_diff;
            shift_amt = 0;
            for (i = 53; i >= 0; i = i - 1) begin
                if (!shift_amt && mant_norm[i]) begin
                    shift_amt = 53 - i;
                end
            end
            result_mant = mant_norm << shift_amt;
            result_exp  = (exp_max > shift_amt) ? (exp_max - shift_amt) : 0;

            // Optional: If result is zero, sign is positive
            if (mant_diff == 0)
                result_sign = 0;
        end
```

```
            ready <= 1;
        end
    end
endmodule
```