```
`timescale 1ns / 1ps

module fpu_mul (
    input  wire        clk,
    input  wire        rst,
    input  wire        sign_a,
    input  wire [10:0] exp_a,
    input  wire [52:0] mant_a,    // 1.M (includes implicit 1)
    input  wire        sign_b,
    input  wire [10:0] exp_b,
    input  wire [52:0] mant_b,    // 1.M (includes implicit 1)
    output reg         result_sign,
    output reg  [10:0] result_exp,
    output reg  [52:0] result_mant,
    output reg         ready
);

    // Internal registers
    reg [105:0] product;  // 53x53 = 106 bits
    reg [10:0]  raw_exp;
    reg [52:0]  final_mant;

    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;

            // 1. Compute result sign
            result_sign = sign_a ^ sign_b;

            // 2. Compute raw exponent
            raw_exp = exp_a + exp_b - 11'd1023;

            // 3. Multiply mantissas
            product = mant_a * mant_b;  // 106-bit product

            // 4. Normalize result
            if (product[105]) begin
                final_mant = product[105:53];     // Already normalized
                result_exp = raw_exp + 1;
            end else begin
                final_mant = product[104:52];     // Needs normalization
                result_exp = raw_exp;
            end

            // 5. Handle underflow (flush to zero)
            if (result_exp <= 0) begin
                result_exp  = 0;
                result_mant = 0;
                result_sign = 0;
            end else begin
                result_mant = final_mant;
            end

            ready <= 1;
        end
    end
endmodule
```