```
`timescale 1ns / 1ps

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

    reg [105:0] dividend_ext;
    reg [52:0]  quotient;
    reg [10:0]  raw_exp;

    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;

            // Sign
            result_sign = sign_a ^ sign_b;

            // Exponent
            raw_exp = exp_a - exp_b + 11'd1023;

            // Divide mantissas with precision
            dividend_ext = {mant_a, 53'b0};   // 106-bit numerator
            quotient     = dividend_ext / mant_b; // Result is 53-bit

            // Normalize quotient
            if (quotient[52]) begin
                result_mant = quotient;
                result_exp  = raw_exp;
            end else begin
                result_mant = quotient << 1;
```

```
            result_exp  = raw_exp - 1;
        end

        // Handle underflow
        if (result_exp <= 0) begin
            result_exp  = 0;
            result_mant = 0;
        end

        // Handle overflow
        if (result_exp >= 11'b11111111111) begin
            result_exp  = 11'b11111111111;
            result_mant = 0;
        end

        ready <= 1;
    end
  end
endmodule
```