SLOW DIVISION ALGORITHM: RESTORING DIVISION ALGORITHM VERILOG CODE:

```
module restoring_division (
  input wire clk,
  input wire reset,
  input wire [31:0] dividend,
  input wire [31:0] divisor,
  output reg [31:0] quotient,
  output reg [31:0] remainder,
  output reg done
);

  reg [31:0] dividend_abs;
  reg [31:0] divisor_abs;
  reg [31:0] quotient_abs;
  reg [31:0] remainder_abs;
  reg [31:0] shift_reg;
  reg [31:0] next_shift_reg;
  reg [5:0] count;
  reg is_negative;

  always @(posedge clk or negedge reset) begin
    if (!reset) begin
      shift_reg <= 32'b0;
      next_shift_reg <= 32'b0;
      quotient <= 32'b0;
      remainder <= 32'b0;
      count <= 6'b0;
      is_negative <= 1'b0;
      done <= 1'b0;
    end else begin
```

```
if (count == 0) begin
  done <= 1'b1;
end else begin
  done <= 1'b0;
end

// Convert dividend and divisor to absolute values
if (dividend[31] == 1'b1) begin
  dividend_abs <= -dividend;
end else begin
  dividend_abs <= dividend;
end

if (divisor[31] == 1'b1) begin
  divisor_abs <= -divisor;
end else begin
  divisor_abs <= divisor;
end

// Initialize variables
if (count == 6'b0) begin
  quotient_abs <= 32'b0;
  remainder_abs <= dividend_abs;
  shift_reg <= dividend_abs;
  is_negative <= dividend[31] ^ divisor[31];
end

// Perform division steps
if (remainder_abs >= 0) begin
  next_shift_reg = shift_reg - divisor_abs;
  if (next_shift_reg[31] == 1'b1) begin
```

```
        quotient_abs[count] <= 1'b0;
      end else begin
        quotient_abs[count] <= 1'b1;
        shift_reg <= next_shift_reg;
      end
      remainder_abs <= next_shift_reg;
    end else begin
      quotient_abs[count] <= 1'b0;
      remainder_abs <= shift_reg;
    end

    // Update count
    count <= count + 1;
  end
 end

 // Assign outputs
 assign quotient = is_negative ? -quotient_abs : quotient_abs;
 assign remainder = remainder_abs;

endmodule
```