FAST DIVISION ALGORITHM: RESTORING DIVISION ALGORITHM VERILOG CODE

```
module newton_raphson_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] reciprocal;
  reg [31:0] estimate;
  reg [31:0] product;
  reg [31:0] correction;
  reg [31:0] next_estimate;

  always @(posedge clk or negedge reset) begin
    if (!reset) begin
      quotient <= 32'b0;
      remainder <= 32'b0;
      done <= 1'b0;
    end else begin
      if (divisor == 0) begin
        done <= 1'b1;
        // Division by zero, handle accordingly
      end else begin
```

```
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

if (dividend_abs == 0) begin
  quotient <= 32'b0;
  remainder <= 32'b0;
  done <= 1'b1;
end else begin
  reciprocal <= 32'h7FFFFFFF / divisor_abs; // Initial approximation for reciprocal
  estimate <= reciprocal * 2; // Initial approximation for quotient

  // Perform iterative Newton-Raphson division
  repeat (6) begin
    product <= divisor_abs * estimate;
    correction <= product >> 31; // Correction term for next estimate
    next_estimate <= estimate + correction;
    estimate <= next_estimate;
  end

  // Calculate quotient and remainder
  quotient <= dividend_abs * estimate;
  remainder <= dividend_abs - (quotient * divisor_abs);
```

```
        // Apply sign to quotient and remainder
        if (dividend[31] ^ divisor[31]) begin
          quotient <= -quotient;
          remainder <= -remainder;
        end

        done <= 1'b1;
       end
     end
   end
 end

endmodule
```