```
module fast_division (
  input [15:0] dividend,     // 16-bit dividend
  input [7:0] divisor,       // 8-bit divisor
  output reg [7:0] quotient,  // 8-bit quotient
  output reg [7:0] remainder, // 8-bit remainder
  output reg done            // Done flag
);

  reg [15:0] temp_dividend;   // Temporary register for dividend
  reg [7:0] temp_remainder;   // Temporary register for remainder
  reg [3:0] i;                // Loop variable

  always @(*) begin
    temp_dividend = dividend; // Assign initial value to the temporary dividend
    temp_remainder = 8'b0;    // Initialize temporary remainder to zero
    done = 0;                 // Set done flag to false

    if (divisor != 8'b0) begin // Check if divisor is non-zero to avoid division by zero
      quotient = 8'b0;        // Initialize quotient to zero

      for (i = 15; i >= 8; i = i - 1) begin // Start the loop from the most significant bit of dividend
        temp_remainder = (temp_remainder << 1) | temp_dividend[i]; // Shift remainder left by 1
and append current dividend bit

        if (temp_remainder >= divisor) begin // Check if remainder is greater than or equal to
divisor
          temp_remainder = temp_remainder - divisor; // Subtract divisor from the remainder
          quotient[i-8] = 1;               // Set the current quotient bit to 1
        end
      end

      remainder = temp_remainder; // Assign the final value of the temporary remainder to the
remainder output
      done = 1;                  // Set done flag to true to indicate division completion
    end
  end

endmodule
```