

## Design

```
module RestoringDivider4Bit(
    input [3:0] dividend,    // Dividend (4-bit)
    input [4:0] divisor,     // Divisor (5-bit)
    output reg [3:0] quotient, // Quotient (4-bit)
    output reg [4:0] remainder // Remainder (5-bit)
);

    reg [4:0] accumulator;    // Accumulator for division
    reg [4:0] neg_divisor;    // Negative of the divisor

    assign neg_divisor = ~divisor + 1;

    always @* begin

        accumulator = 5'b0;
        quotient = dividend; // Initialize quotient
        remainder = 5'b0; // Initialize remainder

        for (int i = 0; i < 4; i = i + 1) begin

            {accumulator,quotient} = {accumulator,quotient} << 1;

            accumulator = accumulator + neg_divisor;

            if (!accumulator[4]) begin
                quotient[i] = 1'b1;
            end else begin
                quotient[i] = 1'b0;
                accumulator = accumulator + divisor;
            end
        end

        remainder = accumulator;
    end

endmodule
```

## Testbench

```
module Testbench;

    reg [3:0] dividend;
    reg [4:0] divisor;
    wire [3:0] quotient;
    wire [4:0] remainder;

    RestoringDivider4Bit divider (dividend,divisor,quotient,remainder);

    initial begin

        $dumpfile("dump.vcd");
        $dumpvars(1);

        // Test case 1: Divide 9 by 3
```

```
dividend = 4'b1001;
divisor = 5'b00011;

#10;

$display("Dividend: %d", dividend);
$display("Divisor: %d", divisor);
$display("Quotient: %d", quotient);
$display("Remainder: %d", remainder);

// Test case 2: Divide 15 by 4
dividend = 4'b1111;
divisor = 5'b00100;

#10;

$display("Dividend: %d", dividend);
$display("Divisor: %d", divisor);
$display("Quotient: %d", quotient);
$display("Remainder: %d", remainder);

end

endmodule
```