EE2003 Assignment 2

Niranjan A. Kartha, EE21B095

P32. Restoring Division

Design a Verilog module that implements the restoring division algorithm. The module should take two 8-bit unsigned numbers (`dividend` and `divisor`) as inputs and produce an 8-bit quotient `quotient` and an 8-bit remainder `remainder`.

Implementation

Two extra inputs, `clk` and `reset` have been introduced in order to control the module. The `reset` bit should be set high for one clock cycle once the inputs to the module are loaded. An output bit, `ready`, is set high when the computation is complete.

Two extra registers, `state` and `rold`, are used. The `state` register keeps track of how many bits have been divided, and `rold` is used to left-rotate the `dividend` to append bits to the partial `remainder`.

After the inputs have been loaded, `quotient` and `remainder` become ready after 7 clock cycles.

Source code

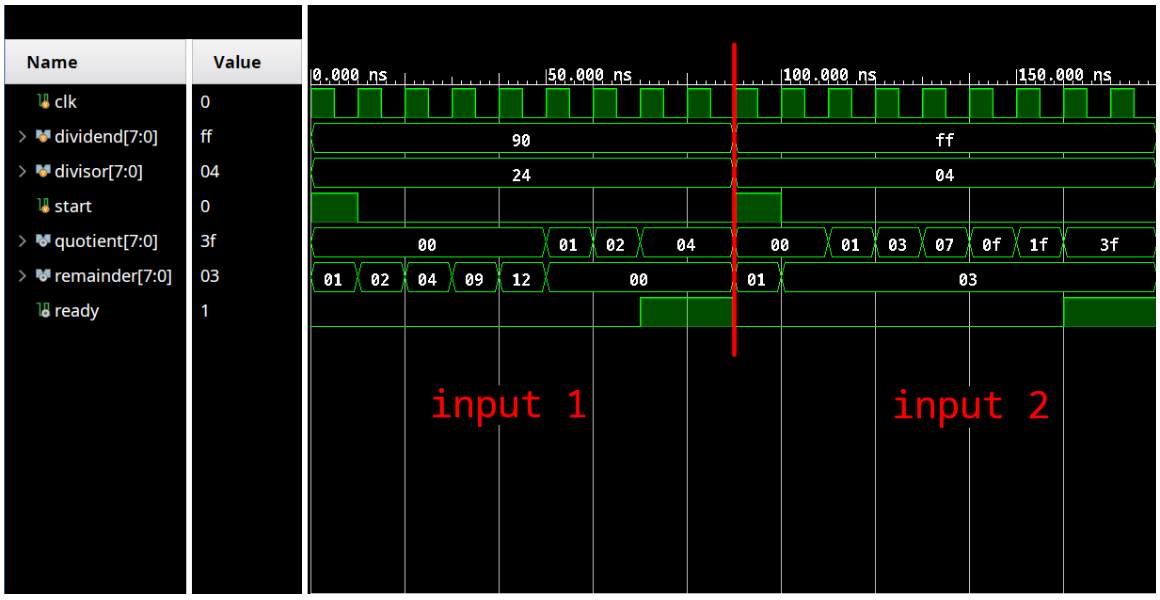
```verilog
`timescale 1ns / 1ps

module divider(
    input clk,
    input [7:0] dividend,
    input [7:0] divisor,
    input reset, // should be high for one clock cycle after giving args
    output reg [7:0] quotient,
    output reg [7:0] remainder,
    output reg ready // is set to high once calculations are complete
);
    reg [2:0] state; // internal state of the divider
    reg [7:0] rold; // to rotate the dividend left and append to remainder

    always @(posedge clk) begin
        if (reset == 1) begin
            ready = 0;
            state = 0;
            rold = dividend;
            quotient = 8'h00;
            remainder = 8'h00;
        end
        if (ready == 0) begin
            // append the leftmost bit of the dividend to remainder,
            // and left shift the dividend once
            remainder = {remainder[6:0], rold[7]};
            rold = {rold[6:0], 1'b0};

            // try subtracting
            remainder = remainder - divisor;
            // set quotient bit based on sign of result
            quotient = {quotient[6:0], ~remainder[7]};
            // rollback if result is negative
            if (remainder[7] == 1) remainder = remainder + divisor;

            // once the calculations are done, mark as ready
            if (state == 3'b111) ready = 1;
            // advance to the next state
            state = state + 1;
        end
    end
endmodule
```

Testbench

```
`timescale 1ns / 1ps

module divider_tb;
    reg clk = 1;
    always #5 clk = ~clk;

    reg [7:0] dividend;
    reg [7:0] divisor;
    reg reset = 0;

    wire [7:0] quotient;
    wire [7:0] remainder;
    wire ready;

    divider div(clk, dividend, divisor, reset, quotient, remainder, ready);

    initial begin
        dividend = 8'h90; // first input
        divisor = 8'h24;
        reset = 1;
        #10 reset = 0;
        #80 dividend = 8'hFF; // second input
        divisor = 8'h04;
        reset = 1;
        #10 reset = 0;
        #80 $finish;
    end
endmodule
```

Outputs

Input 1 shows 144/12 = 4 with remainder 0. Input 2 shows 255/4 = 63 with remainder 3.


Name
Value
clk
0
dividend[7:0]
ff
divisor[7:0]
04
start
0
quotient[7:0]
3f
remainder[7:0]
03
ready
1
0.000 ns
50.000 ns
100.000 ns
150.000 ns
90
ff
24
04
00
01
02
04
00
01
03
07
0f
1f
3f
01
02
04
09
12
00
01
03
input 1
input 2