Lab3

Lab3
任务一：设计实现无符号数乘法器
模块代码
仿真代码
仿真结果
任务二：设计实现无符号数除法器
模块代码
仿真代码
仿真结果
任务三：设计实现浮点加法器
思考题：设计实现有符号数乘法器

任务一：设计实现无符号数乘法器

模块代码

```
`timescale 1ns / 1ps

module mul32(
    input clk,
    input rst,
    input [31: 0] multiplicand,
    input [31: 0] multiplier,
    input start,

    output [63: 0] product,
    output finish
    );
    reg [63: 0] ans; // 结果寄存器
    integer i; // 计数寄存器
    always @(posedge rst) begin // reset
        ans = 0;
        i = 0;
    end
    always @(posedge start) begin //开始乘法
        ans[31:0] = multiplier;
        i = 0;
    end
    always @(posedge clk) begin
        if (start && !rst && i < 32) begin
            /* 如果最低位为1则寄存器高位做一次加法 */
            if (product[0])
                ans[63:32] = ans[63:32] + multiplicand;
            ans = ans >> 1; /* 低位右移，Multiplier右移一位 */
            i = i + 1; /* 计数器加一 */
        end
    end
    assign product = ans; /* 结果赋值 */
```

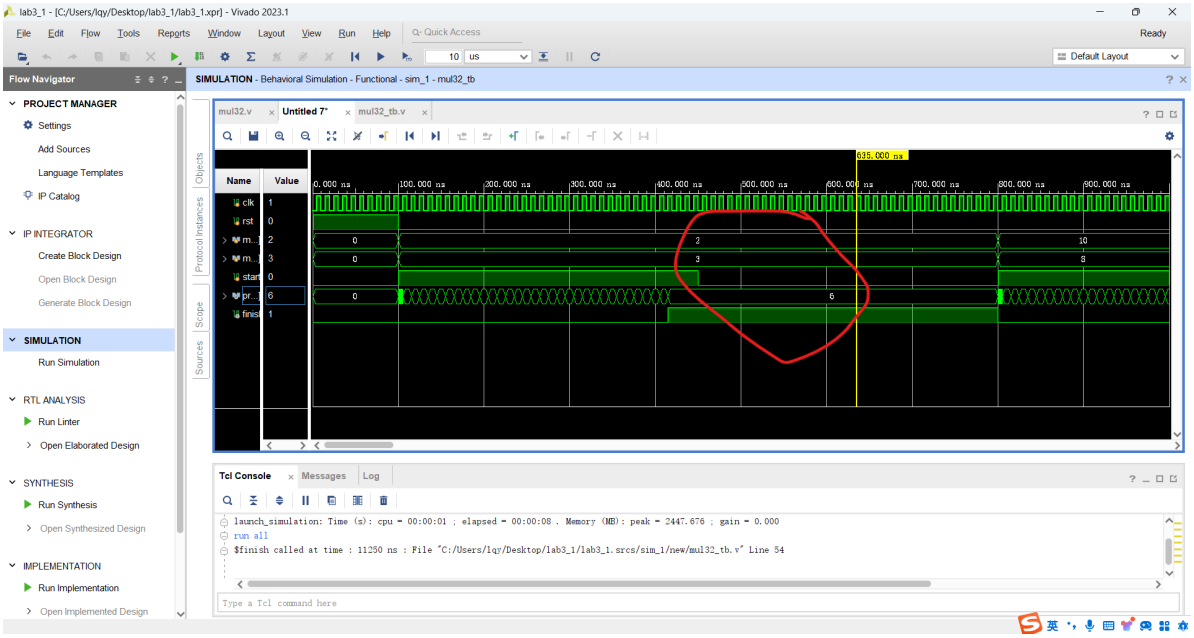
```
    assign finish = (i == 32); /* 运算完成状态赋值 */
endmodule
```

仿真代码

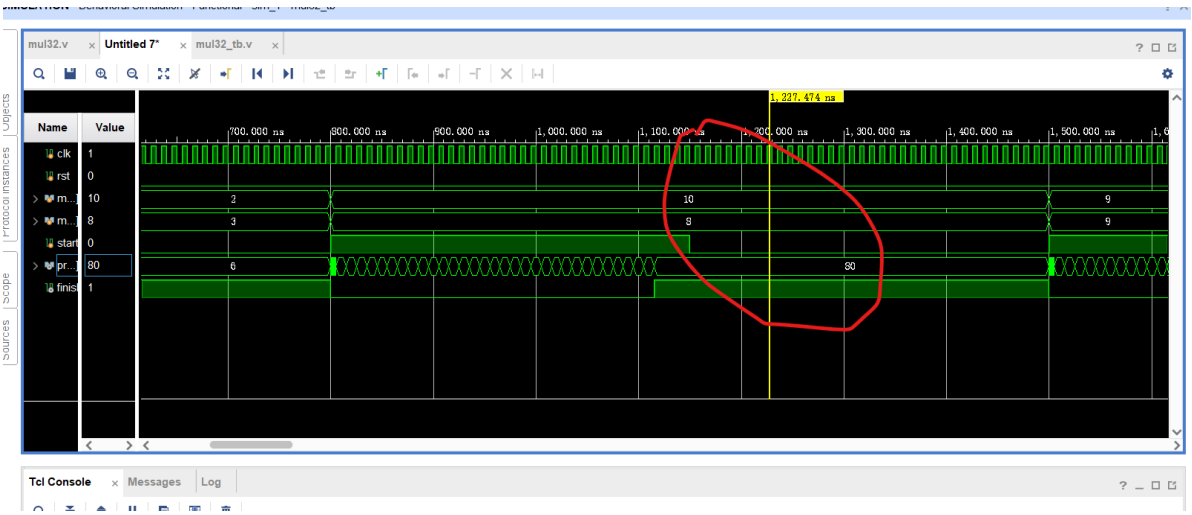
```
`timescale 1ns / 1ps

module mul32_tb();
    reg clk;
    reg rst;
    reg[31:0] multiplicand;
    reg[31:0] multiplier;
    reg start;
    wire[63:0] product;
    wire finish;

    initial begin
    clk = 0;
    rst = 1;
    multiplicand = 0;
    multiplier   = 0;
    start        = 0;
    #100
    rst = 0;
    start = 1;
    multiplicand = 32'd2;
    multiplier   = 32'd3;
    #350
    start = 0;

    #350
    start = 1;
    multiplicand = 32'd10;
    multiplier   = 32'd8;
    #350
    start = 0;

    #350
    start = 1;
    multiplicand = 32'd9;
    multiplier   = 32'd9;
    #350
    start = 0;

    #350
    start = 1;
    multiplicand = 32'd50;
    multiplier   = 32'd6;
    #350
    start = 0;

    #350
    start = 1;
    multiplicand = 32'd20240321;
    multiplier   = 32'd1931;
```

```
51      #350
52      start = 0;
53
54      #8000 $finish();
55      end
56
57      always #5 clk = ~clk;
58
59      mul32 mul32_u(
60          .clk(clk),
61          .rst(rst),
62          .multiplicand(multiplicand),
63          .multiplier(multiplier),
64          .start(start),
65          .product(product),
66          .finish(finish)
67      );
68  endmodule
```

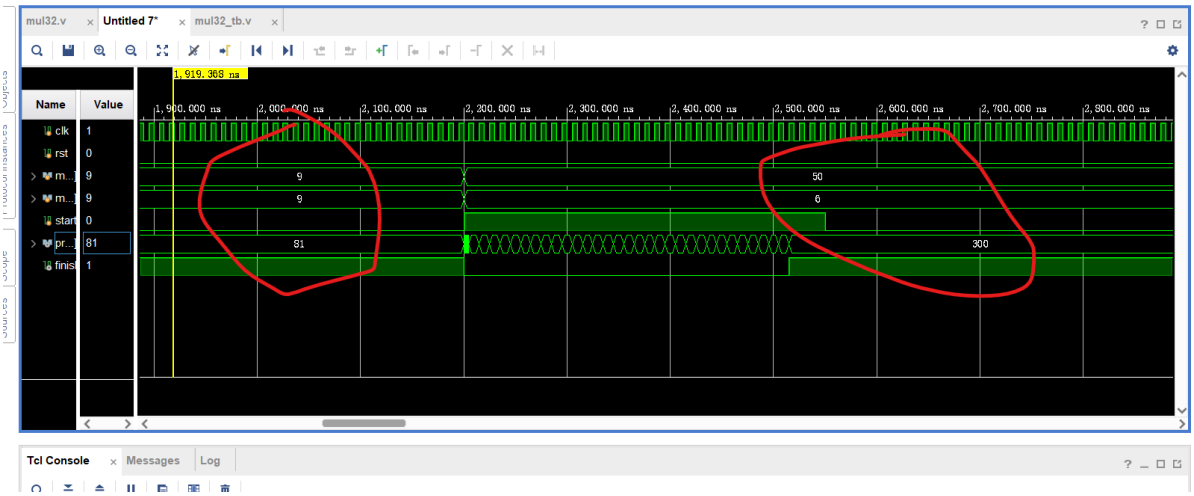
仿真结果



运算结果为$2 \times 3 = 6$，运算完成时`finish=1`



运算结果为$10 \times 8 = 80$，运算完成时`finish=1`



运算结果为$9 \times 9 = 81$，运算完成时`finish=1`

运算结果为$50 \times 6 = 300$，运算完成时`finish=1`

任务二：设计实现无符号数除法器

模块代码

```
`timescale 1ns / 1ps

module div32(
    input clk,
    input rst,
    input start,
    input [31:0] dividend,
    input [31:0] divisor,

    output [31:0] quotient,
    output [31:0] remainder,
    output finish
    );

    reg [31:0] Divisor;
    reg [63:0] ans;
    reg [32:0] tmp;
    integer i;

    /* Reset */
    always @(posedge rst) begin
       Divisor = 32'b1;
       ans = 64'b0;
       i = 0;
    end

    /* Start */
    always @(posedge start) begin
        Divisor = divisor;
        ans = {32'b0, dividend} << 1;
        i = 0;
    end
```

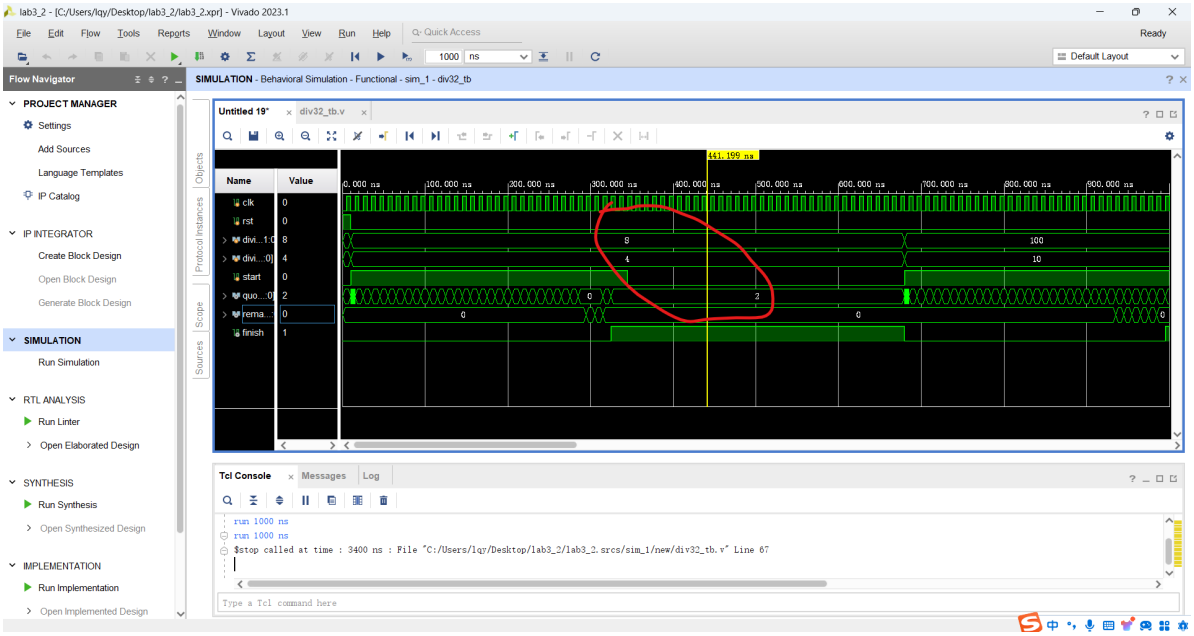
```
33
34      always @(posedge clk) begin
35          if (start && !rst && i < 32) begin
36              tmp = {1'b1, ans[63:32]} - {1'b0, Divisor};
37              /* dividend < divisor */
38              if (!tmp[32]) begin
39                  ans = ans << 1;
40                  ans[0] = 0;
41              end
42              /* dividend >= divisor */
43              else begin
44                  ans[63:32] = tmp[31:0];
45                  ans = ans << 1;
46                  ans[0] = 3;
47              end
48              i = i + 1; /* Step++ */
49          end
50      end
51      assign quotient = ans[31:0];
52      assign remainder = ans[63:32] >> 1;
53      assign finish = (i == 32);
54  endmodule
```

仿真代码

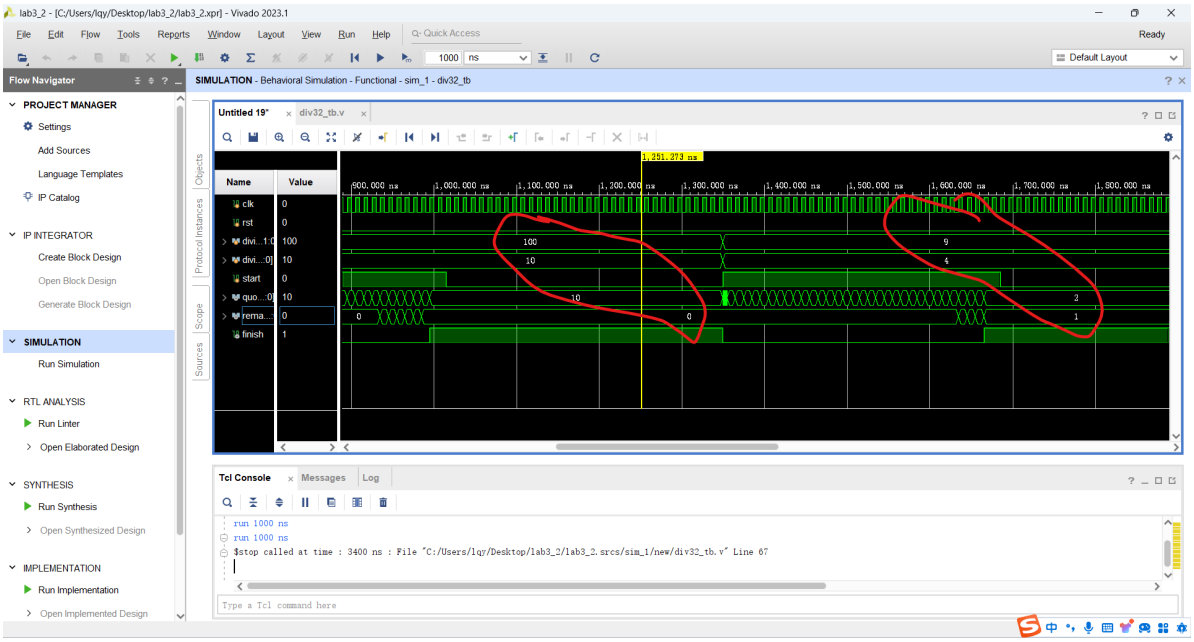
```
1   `timescale 1ns / 1ps
2
3   module div32_tb();
4         reg clk;
5         reg rst;
6         reg [31:0] dividend;
7         reg [31:0] divisor;
8         reg start;
9
10        wire [31:0] quotient;
11        wire [31:0] remainder;
12        wire finish;
13        div32   u_div(
14           .clk(clk),
15           .rst(rst),
16           .dividend(dividend),
17           .divisor(divisor),
18           .start(start),
19           .quotient(quotient),
20           .remainder(remainder),
21           .finish(finish)
22        );
23        always #5 clk = ~clk;
24
25        initial begin
26         clk =0;
27         rst = 1;
28         start = 0;
29         dividend = 32'd0;
30         divisor  = 32'd0;
```

```
        #10
        rst = 0;
        start = 1;
        dividend = 32'd8;
        divisor  = 32'd4;
        #335
        start = 0;

        #335
        start = 1;
        dividend = 32'd100;
        divisor  = 32'd10;
        #335
        start = 0;

        #335
        start = 1;
        dividend = 32'd9;
        divisor  = 32'd4;
        #335
        start = 0;

        #340
        start = 1;
        dividend = 32'd100;
        divisor  = 32'd99;
        #350
        start = 0;

        #340
        start = 1;
        dividend = 32'd20240321;
        divisor = 32'd1931;
        #335
        start = 0;

        #350 $stop();

      end

endmodule
```

仿真结果

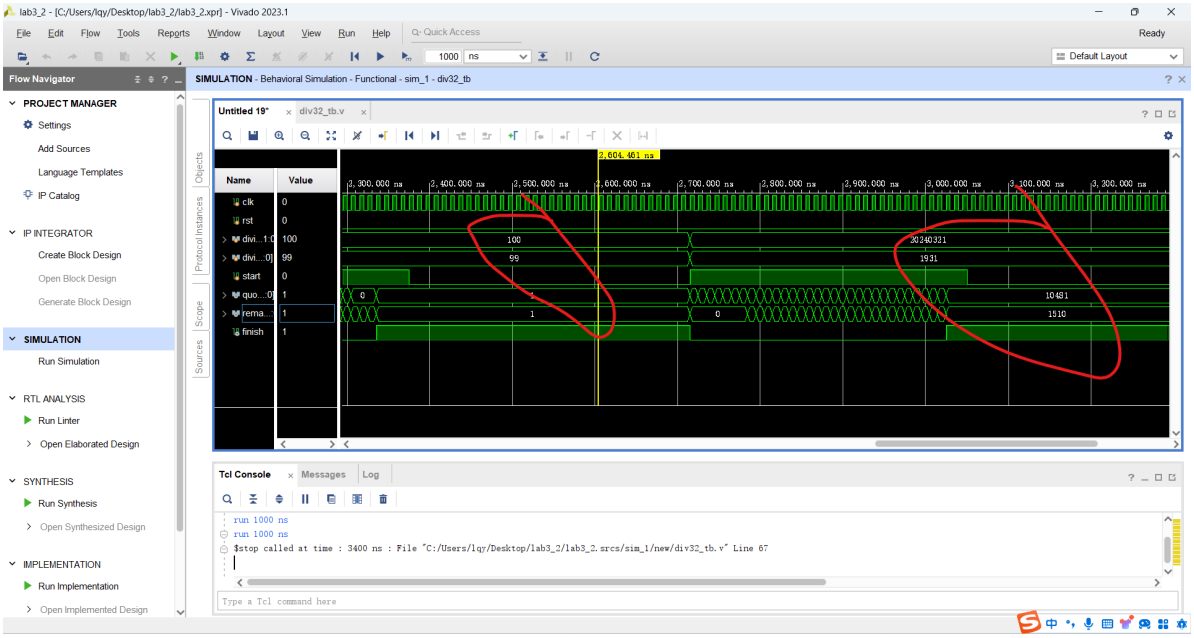


$$8 \div 4 = 2 \cdots 0$$



$$100 \div 10 = 10 \cdots 0$$

$$9 \div 4 = 2 \cdots 1$$



$$100 \div 99 = 1 \cdots 1$$

任务三：设计实现浮点加法器

来不及做了，已知根据思路将浮点数按照固定位分别提取出符号位`[31]`、指数`[30:23]`和尾数`[22:0]`，然后按照下列思路进行计算：

1. 对齐：将小指数往大指数对齐，同时对尾数进行右移
2. 尾数加减法：根据符号位确定尾数做加法还是减法
3. 规格化：对步骤2得到的结果进行规格化，对尾数进行适当右移并增加指数，同时判断是否有溢出

代码还没写完捏

思考题：设计实现有符号数乘法器

设计思路：设计一个子模块用于将有符号数转化为符号位和无符号数，然后利用无符号数乘法器模块代码，直接计算转化后的无符号数乘法，最后再根据符号位确定结果的符号位，再将结果转化回有符号数