同济大学

计算机科学与技术系

计算机组成原理课程设计实验报告

学　　号：1853790

姓　　名：庄镇华

专　　业：计算机科学与技术

授课老师：陈永生

目录

一、 实验内容

1. 了解 32 位有符号、无符号除法器的实现原理

2. 使用 $Verilog$ 实现一个 32 位有符号除法器和一个 32 位无符号除法器

二、 模块建模

2.1 功能描述

无符号除法器功能为：将两个 32 位无符号数相除，得到一个 32 位商和 32 位余数。本实验分别实现 32 位有符号和无符号除法器，结果为 32 位商 $quotient$ 和 32 位余数 $remainder$，分别存放在 CPU 的专用寄存器 LO 和 HI 中。除法器时钟信号下降沿时检查 $start$ 信号，有效时开始执行，执行除法指令时，$busy$ 标志位置 1。在执行除法指令时，任何情况下不产生算数异常，当除数为 0 时，运算结果未知，对除法器除数为 0 和溢出情况的发生通过汇编指令中其他指令进行检查和处理。

带符号除法器功能为：将两个 32 带符号数相除，得到一个 32 位商和余数，基本和无符号除法器类似，注意余数符号与被除数符号相同。

2.2 实现原理

无符号除法器：本次实验采用不恢复余数的方法构造除法器。恢复余数除法器的构造方法为：对于 32 位无符号除法，可将被除数 a 转换成高 32 位为 0 低 32 位是 a 的数 $temp_a$，在每个周期开始时 $temp_a$ 向左移动一位，最后一位补零，然后判断 $temp_a$ 的高 32 位是否大于等于除数 b，如是则 $temp_a$ 的高 32 位减去 b 并且加 1，得到的值赋给 $temp_a$，如果不是则直接进入下一步，执行结束后 $temp_a$ 的高 32 位即为余数，低 32 位即为商。

不恢复余数除法器构造方法为:不恢复余数即不管相减结果是正还是负,都把它写入 reg_r，若为负，下次迭代不是从中减去除数而是加上除数。其主要有以下几个步骤：1. 将除数向左移位到恰好大于被除数 2. 若余数为正：余数减去移位后除数；若余数为负：余数加上移位后除数；3. 若现余数为正，该位结果为 1，否则为 0，将除数向右移位一位 4. 重复 2，3，直到移位后除数小于原除数

带符号除法器：对于 32 位有符号除法，可先将有符号数转换成无符号数除法，根据被除数和除数的符号判断商的符号，被除数是负数时余数为负，否则为正。

2.3 模块代码

```
‘timescale 1ns / 1ps
module DIVU(
    input [31 : 0] dividend,
    input [31 : 0] divisor,
    input start,
    input clock,
    input reset,
    output [31 : 0] q,
    output [31 : 0] r,
    output reg busy
    );

        wire ready;
        reg [4 : 0] count;
        reg [31 : 0] reg_q;
        reg [31 : 0] reg_r;
        reg [31 : 0] reg_b;
        reg busy2, r_sign;
        assign ready = ~busy & busy2;
        wire [32 : 0] sub_add = r_sign ? ({reg_r, q[31]} + {1'b0,
           reg_b}) : ({reg_r, q[31]} - {1'b0, reg_b});
        assign r = r_sign ? reg_r + reg_b : reg_r;
        assign q = reg_q;
        always @(posedge clock or posedge reset) begin
        if (reset) begin
                count <= 5'b0;
                busy <= 0;
                busy2 <= 0;
        end else begin
                busy2 <= busy;
                if (start) begin
                        reg_r <= 32'b0;
```

```
                        r_sign <= 0;
                        reg_q <= dividend;
                        reg_b <= divisor;
                        count <= 5'b0;
                        busy <= 1'b1;
                end else if (busy) begin
                        reg_r <= sub_add[31 : 0];
                        r_sign <= sub_add[32];
                        reg_q <= {reg_q[30 : 0], ~sub_add[32]};
                        count <= count + 5'b1;
                        if (count == 5'b11111)
                                busy <= 0;
                end
        end
        end
endmodule
```

```
`timescale 1ns / 1ps
module DIV(
    input [31 : 0] dividend,
    input [31 : 0] divisor,
    input start,
    input clock,
    input reset,
    output [31 : 0] q,
    output [31 : 0] r,
    output reg busy
);

        wire ready;
        reg [4 : 0] count;
        reg [31 : 0] reg_q;
        reg [31 : 0] reg_r;
```

```
    reg [31 : 0] reg_b;
    wire [31 : 0] reg_r2;
    reg busy2, r_sign;
    assign ready = ~busy & busy2;
    wire [32 : 0] sub_add = r_sign ? ({reg_r, reg_q[31]} + {1'
       b0, reg_b}) : ({reg_r, reg_q[31]} - {1'b0, reg_b});
    assign reg_r2 = r_sign ? reg_r + reg_b : reg_r;
    assign r = dividend[31] ? (~reg_r2 + 1) : reg_r2;
assign q = (divisor[31] ^ dividend[31]) ? (~reg_q + 1) : reg_q
   ;
    always @ (posedge clock or posedge reset) begin
    if (reset) begin
            count <= 5'b0;
            busy <= 0;
            busy2 <= 0;
    end else begin
            busy2 <= busy;
            if (start) begin
                    reg_r <= 32'b0;
                    r_sign <= 0;
                    reg_q <= dividend[31] ? ~dividend + 1 :
                       dividend;
                    reg_b <= divisor[31] ? ~divisor + 1 :
                       divisor;
                    count <= 5'b0;
                    busy <= 1'b1;
            end else if (busy) begin
                    reg_r <= sub_add[31 : 0];
                    r_sign <= sub_add[32];
                    reg_q <= {reg_q[30 : 0], ~sub_add[32]};
                    count <= count + 5'b1;
                    if (count == 5'b11111)
```

```
                busy <= 0;
            end
        end
        end
endmodule
```

三、 测试模块建模

3.1 测试思路

首先通过 8 组既含边界条件又含有正常条件的测试案例得到结果，然后与 $VIVADO$ 自带的无符号、有符号除法器所得结果进行对比，最后即可验证编写的程序的正确与否。

3.2 测试代码

```
‘timescale 1ns / 1ps
module divu_tb;
        reg [31 : 0] a;
    reg [31 : 0] b;
    reg start;
    reg clk;
    reg rst;
    wire [31 : 0] q;
    wire [31 : 0] r;
    wire busy;

        wire [31 : 0] rc1;
        wire [31 : 0] rc2;

    integer intevel = 70;
        integer short = 1;

        DIVU uut(a, b, start, clk, rst, q, r, busy);
        assign rc1 = a / b;
```

```
assign rc2 = a % b;

    always  #1 clk = ~clk;

initial begin
    clk = 0;
    rst = 1;
    #5 rst = 0;

    a = 32'hfffffff0;
    b = 32'h00000005;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'b0;
    b = 32'hffffffff;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'd26;
    b = 32'd5;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'hffffffff;
    b = 32'hffffffff;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'h0000ffff;
    b = 32'hffff0000;
    start = 1'b1; #short start = 1'b0;
    #(intevel-short);
    a = 32'hffff0000;
    b = 32'h0000ffff;
    start = 1'b1; #short start = 1'b0;
```

```
        #(intevel - short);
        a = 32'hffff0000;
        b = 32'hfffffffe;
        start = 1'b1; #short start = 1'b0;
        #(intevel - short);
        a = 32'h00000001;
        b = 32'h0000000f;
        #50 $stop;
    end
endmodule
```

```
`timescale 1ns / 1ps
module div_tb;
        reg [31 : 0] a;
    reg [31 : 0] b;
    reg start;
    reg clk;
    reg rst;
    wire [31 : 0] q;
    wire [31 : 0] r;
    wire busy;

        wire [31 : 0] rc1;
        wire [31 : 0] rc2;

    integer intevel = 70;
        integer short = 1;

    DIV uut(a, b, start, clk, rst, q, r, busy);
        assign rc1 = $signed(a) / $signed(b);
    assign rc2 = $signed(a) % $signed(b);

        always  #1 clk = ~clk;
```

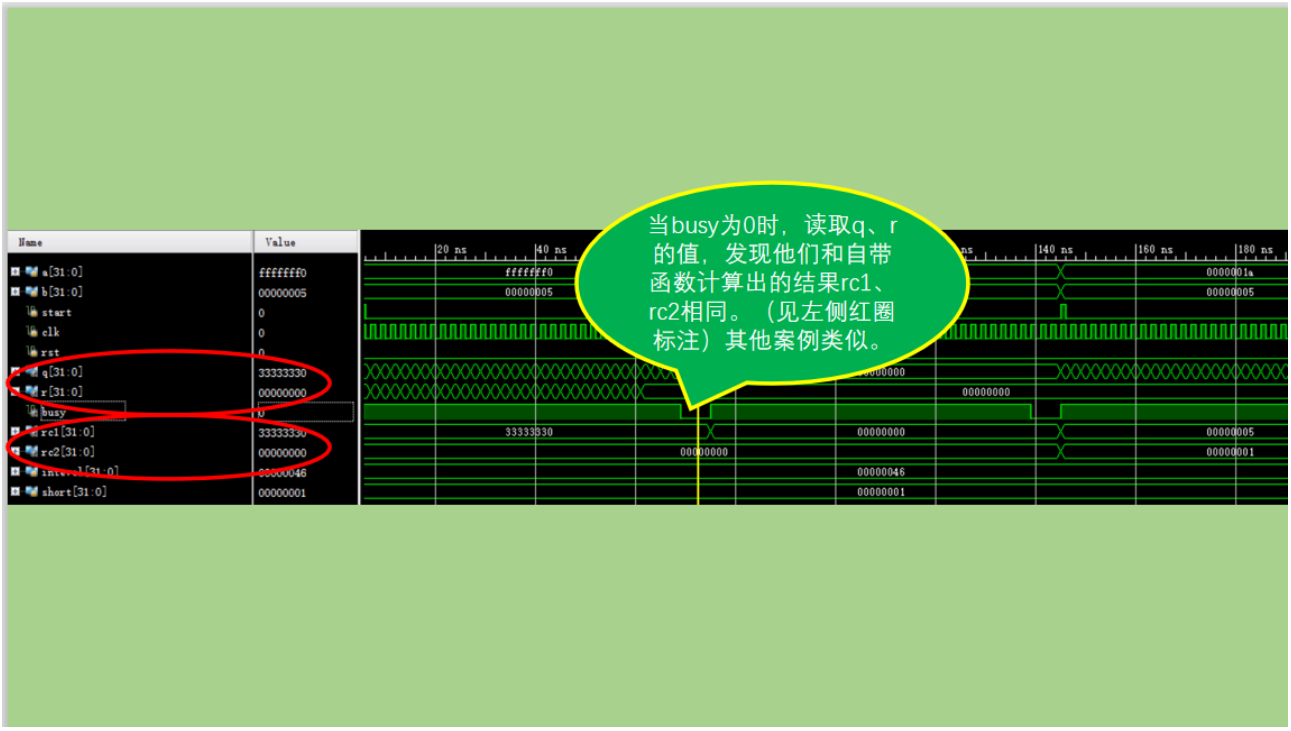
```
initial begin
    clk = 0;
    rst = 1;
    #5 rst = 0;

    a = 32'hfffffff0;
    b = 32'h00000005;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'b0;
    b = 32'hffffffff;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'd26;
    b = 32'd5;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'hffffffff;
    b = 32'hffffffff;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'h0000ffff;
    b = 32'hffff0000;
    start = 1'b1; #short start = 1'b0;
    #(intevel-short);
    a = 32'hffff0000;
    b = 32'h0000ffff;
    start = 1'b1; #short start = 1'b0;
    #(intevel - short);
    a = 32'hffff0000;
    b = 32'hfffffffe;
```

```
        start = 1'b1; #short start = 1'b0;
        #(intevel - short);
        a = 32'h00000001;
        b = 32'h0000000f;
        #50 $stop;
    end
endmodule
```

四、 实验结果

4.1 无符号除法

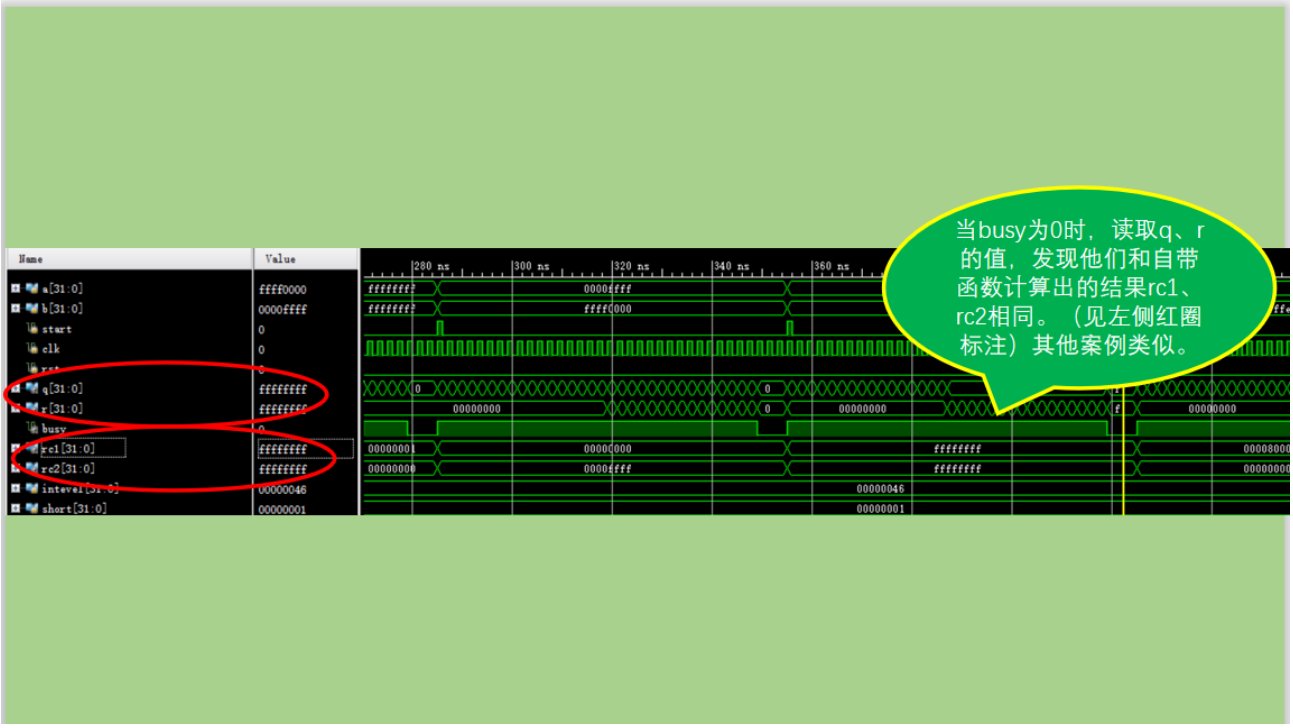


a、b 分别代表被除数和除数，q、r 分别代表自己编写的模块运算结果商和余数，$rc1$、$rc2$ 分别代表 $VIVADO$ 自带函数运算结果商和余数。

	a	b	q	r	$rc1$	$rc2$
第 1 组	$fffffff0$	00000005	33333330	00000000	33333330	00000000
第 2 组	00000000	$ffffffff$	00000000	00000000	00000000	00000000
第 3 组	$0000001a$	00000005	00000005	00000001	00000005	00000001
第 4 组	$ffffffff$	$ffffffff$	00000001	00000000	00000001	00000000
第 5 组	$0000ffff$	$ffff0000$	00000000	$0000ffff$	00000000	$0000ffff$
第 6 组	$ffff0000$	$0000ffff$	00010000	00000000	00010000	00000000
第 7 组	$ffff0000$	$fffffffe$	00000000	$ffff0000$	00000000	$ffff0000$
第 8 组	00000001	$0000000f$	00000000	00000001	00000000	00000001

表 1: 无符号除法

4.2 带符号除法



a、b 分别代表被除数和除数，q、r 分别代表自己编写的模块运算结果商和余数，$rc1$、$rc2$ 分别代表 $VIVADO$ 自带函数运算结果商和余数。

	a	b	q	r	$rc1$	$rc2$
第 1 组	$fffffff0$	00000005	$fffffffd$	$ffffffff$	$fffffffd$	$ffffffff$
第 2 组	00000000	$ffffffff$	00000000	00000000	00000000	00000000
第 3 组	$0000001a$	00000005	00000005	00000001	00000005	00000001
第 4 组	$ffffffff$	$ffffffff$	00000001	00000000	00000001	00000000
第 5 组	$0000ffff$	$ffff0000$	00000000	$0000ffff$	00000000	$0000ffff$
第 6 组	$ffff0000$	$0000ffff$	$ffffffff$	$ffffffff$	$ffffffff$	$ffffffff$
第 7 组	$ffff0000$	$fffffffe$	00008000	00000000	00008000	00000000
第 8 组	00000001	$0000000f$	00000000	00000001	00000000	00000001

表 2: 带符号除法