1、系统功能详细设计

要求：描述系统主要功能，绘制硬件模块框图，并结合模块框图描述各模块之间的相互关系。*若完成了附加题，则需要绘制 Booth 乘法器硬件模块框图，并结合模块框图描述各模块之间的相互关系。

系统主要功能：基于恢复余数法，在不使用/、%符号的情况下，实现两个 8 位带符号整数原码（x 和 y）的除法运算（x ÷ y），并输出商（z）和余数（r）。具体功能为：

1.读取输入的两个 8 位整数原码 x 和 y（均使用原码形式）；

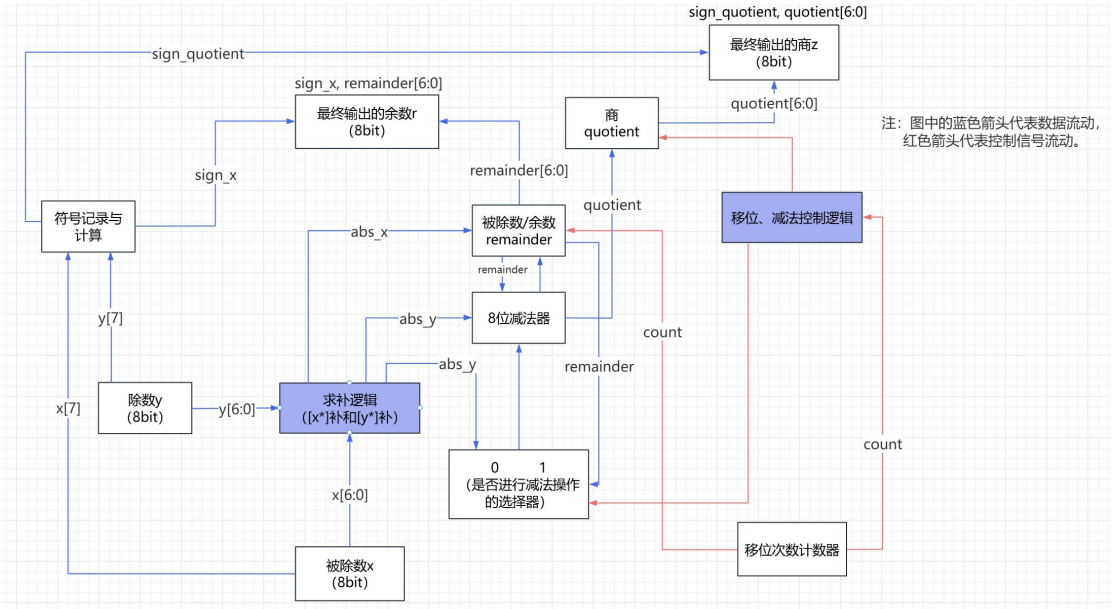
2.利用恢复余数法计算 x ÷ y 的商（z）和余数（r）；

3.处理输入输出的符号；

4.指示除法运算是否正在进行（busy 信号）；

5.当 rst 信号为高电平时可进行复位操作。

硬件框图如下：



模块间的关系：

1. 除数 y 模块

输入：接收除数 y 的 8 位数据。

输出：将除数 y 分成符号位 y[7]和数值部分 y[6:0]，分别传递给符号记录与计算模块和求补逻辑模块。

2. 被除数 x 模块

输入：接收被除数 x 的 8 位数据。

输出：将被除数 x 分成符号位 x[7]和数值部分 x[6:0]，分别传递给符号记录与计算模块和求补逻辑模块。

3. 符号记录与计算模块

输入：接收来自被除数 x 模块和除数 y 模块的两个符号位数据（x[7]、y[7]），

输出：记录 sign_x（被除数符号），并将其传入 r 模块作为最终输出的余数的符号位；

计算得出 sign_quotient（商的符号），并将其传入 z 模块作为最终输出的商的符号位。

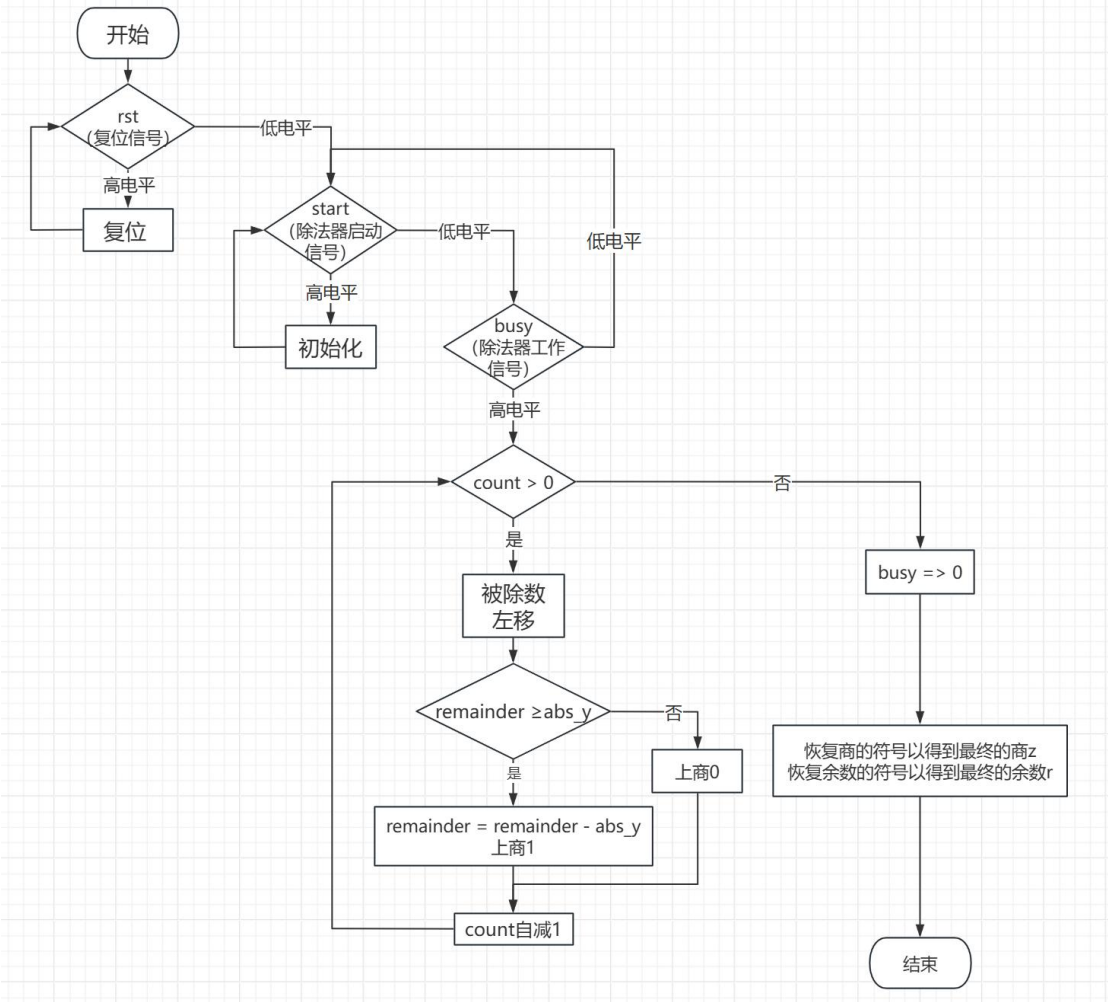
4. 求补逻辑模块

输入：接收来自被除数 x 模块的 x[6:0]和来自除数 y 模块 y[6:0]（即 x 和 y 的无符号数值部分）。
输出：计算并输出 x*的补码（abs_x）和 y*的补码（abs_y），将 abs_x 传入 remainder 模块，将 abs_y 传入 0/1 选择器模块、减法器模块。
5. 被除数/余数 remainder 模块
输入：接收 abs_x、来自减法器模块的数据和来自移位次数计数器的 count 作为控制信号。
输出：不断将更新后的 remainder 数据（左移过程中的余数）传入减法器模块和 0/1 选择器模块，并将最终计算得到的去除了符号位的余数数据 remainder[6:0]传入 r 模块。
6. 移位次数计数器
输入：设置初始的计数值 count = 8。
输出：在每次移位操作后递减的 count 信号，用于控制移位、减法控制逻辑模块操作和 remainder 模块的左移次数。
7. 移位、减法控制逻辑模块
输入：接收来自移位次数计数器的 count 作为控制信号。
输出：控制 0/1 选择器模块的输出（是否使减法器模块进行减法操作）。
8. 0/1 选择器模块（控制减法器是否进行减法操作）
输入：接收 abs_y、remainder 和来自移位、减法控制逻辑模块的控制信号。
输出：根据 remainder 和 abs_y 的比较结果向减法器模块选择输出控制信号，以控制其是否进行减法操作。
9. 8 位减法器模块
输入：接收 remainder 和 abs_y。
输出：根据来自 0/1 控制器模块的信号决定是否对接收的 remainder 数据执行减法操作（remainder - abs_y），将更新后的的 remainder 数据传入 remainder 模块以构成循环，不断将更新后的的 quotient 数据传入 quotient 模块。
10. 商 quotient 模块
输入：接收 quotient 数据和来自移位次数计数器的 count 作为控制信号。
输出：将最终计算得到的去除了符号位的余数数据 quotient6:0]传入 z 模块。
11. 最终输出的商 z 模块
输入：接收 sign_quotient 和 quotient[6:0]。
输出：恢复商的符号，输出最终的商 z。
12. 最终输出的余数 r 模块
输入：接收 sign_x 和 remainder[6:0]。
输出：恢复余数的符号，输出最终的余数 r。

2、除法器算法流程

要求：绘制除法器算法流程图，并用文字详细描述算法执行过程。*若完成了附加题，则需要绘制乘法器算法流程图，并配以文字描述其执行过程。*

除法器算法流程图如下：



算法执行过程（基于恢复余数法）：

1. 复位：当 rst 信号为高电平时，系统复位，所有寄存器清零，busy 信号设为 0（低电平）。

2. 初始化：当 start 信号为高电平时，开始初始化

①busy 设为 1，表示除法器进入工作状态；

②分别用 sign_x 和 sign_y 记录被除数 x 和除数 y 的符号位（x[7]、y[7]）；

③计算商的符号 sign_quotient（为 x 和 y 符号位的异或结果）；

④计算得出 x*的补码和 y*的补码（$[x*]_{补}=\{1'b0,x[6:0]\}$， $[y*]_{补}=\{1'b0,y[6:0]\}$），并用 abs_x 和 abs_y 分别存储$[x*]_{补}$和$[y*]_{补}$；

⑤初始化商 quotient 和余数 remainder 为 0；

⑥设定移位计数 count 为 8。

3. 除法操作：在 busy 信号为高电平时，开始进行除法运算

（1）移位：当 count 大于 0 时，进行移位操作

①将被除数 abs_x 的最高位移入余数 remainder 的最低位（remainder = {remainder[6:0], abs_x[7]}），remainder 高 7 位左移一位；

②将被除数 abs_x 左移一位，低位补 0（abs_x = {abs_x[6:0], 1'b0}）；

③比较 remainder 和 abs_y 的大小关系：

a. 如果 remainder 大于等于 abs_y，说明余数为正，进行减法操作：remainder 减去 abs_y（remainder = remainder - abs_y），同时上商 1（quotient = {quotient[6:0], 1'b1}）；

b. 如果 remainder 小于 abs_y，上商 0（quotient = {quotient[6:0], 1'b0}）；

④移位计数 count 减 1；

（2）当 count 等于 0 时，说明左移次数已达要求，除法操作结束。此时将 busy 信号设为 0，表示除法操作完成。

4. 得到最终结果：

①将 sign_x 作为最终余数结果 r 的符号位，remainder 的低 7 位作为其数值部分（r <= {sign_x,remainder[6:0]}）；

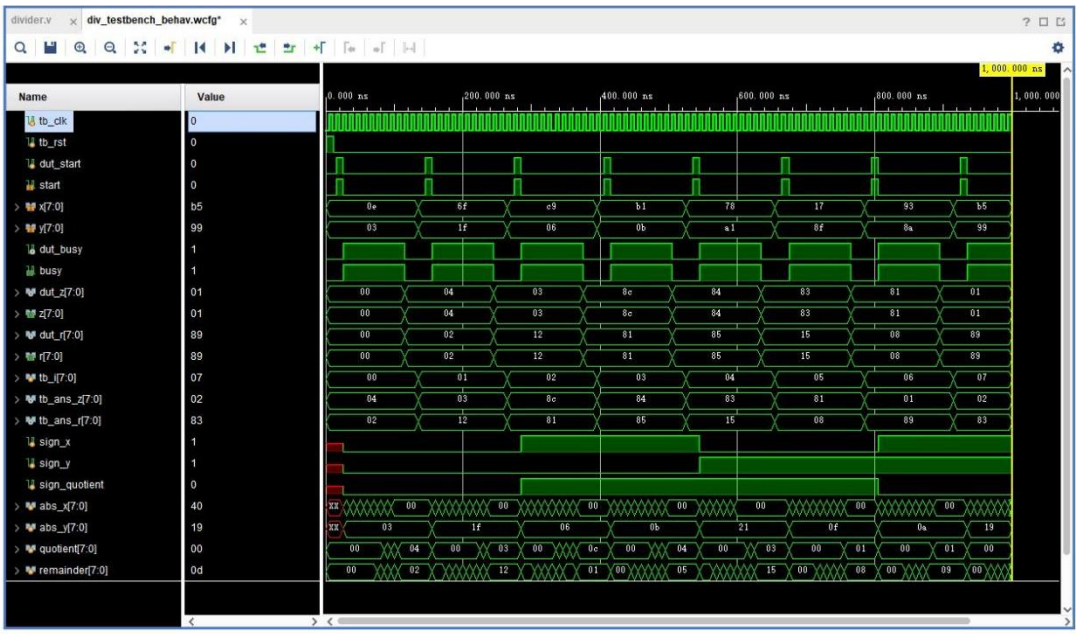
②将 sign_quotient 作为最终商结果 z 的符号位，quotient 的低 7 位作为其数值部分（z <= {sign_quotient,quotient[6:0]}）。

5. 结束。

3、调试报告

要求：至少分析 2 个不同的测试用例，且必须包含完整的仿真波形截图及详细的时序分析。*若完成了附加题，则需要再分析 2 个不同的乘法测试用例，且必须包含完整的仿真波形截图及详细的时序分析。

完整波形图如下：

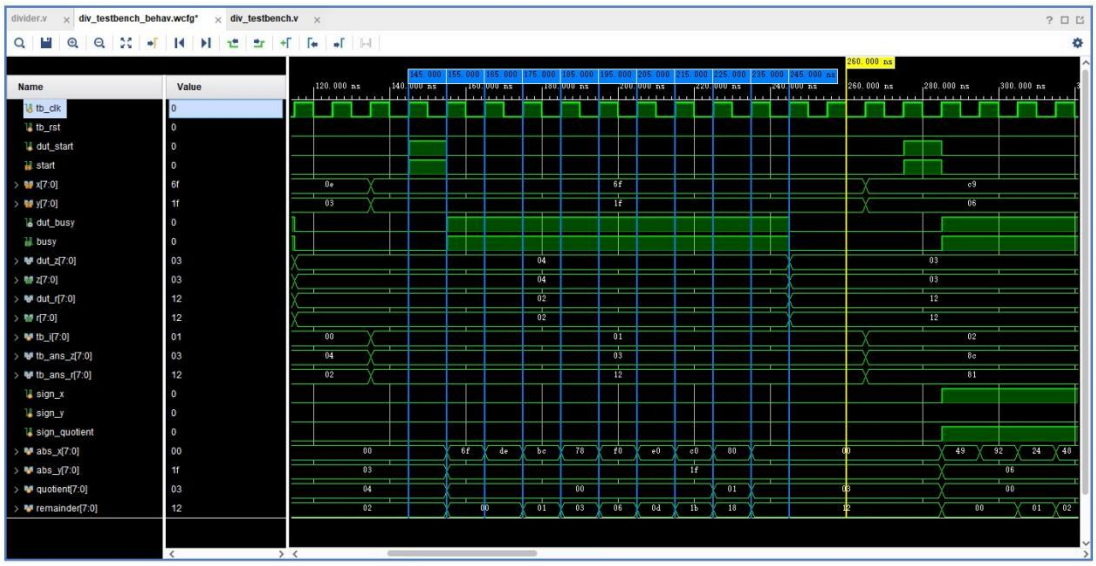


信号说明：时钟信号 clk、复位信号 rst、除法器启动信号 start、除法器工作信号 busy。sign_x 用于记录被除数 x 的符号，sign_y 用于记录除数 y 的符号，sign_quotient 用于记录商的符号，abs_x 用于记录 x*的补码以及左移过程中的与余数相关的过程值，abs_y 用于记录 y*的补码，count 用于记录移位次数。

在 rst 信号为低电平时（12ns 时）进行初始化：busy、count、quotient、remainder、r、z 信号均初始化为 0。

（具体分析详见下页）

（1）分析 test1：111 /31 =3 ... 18



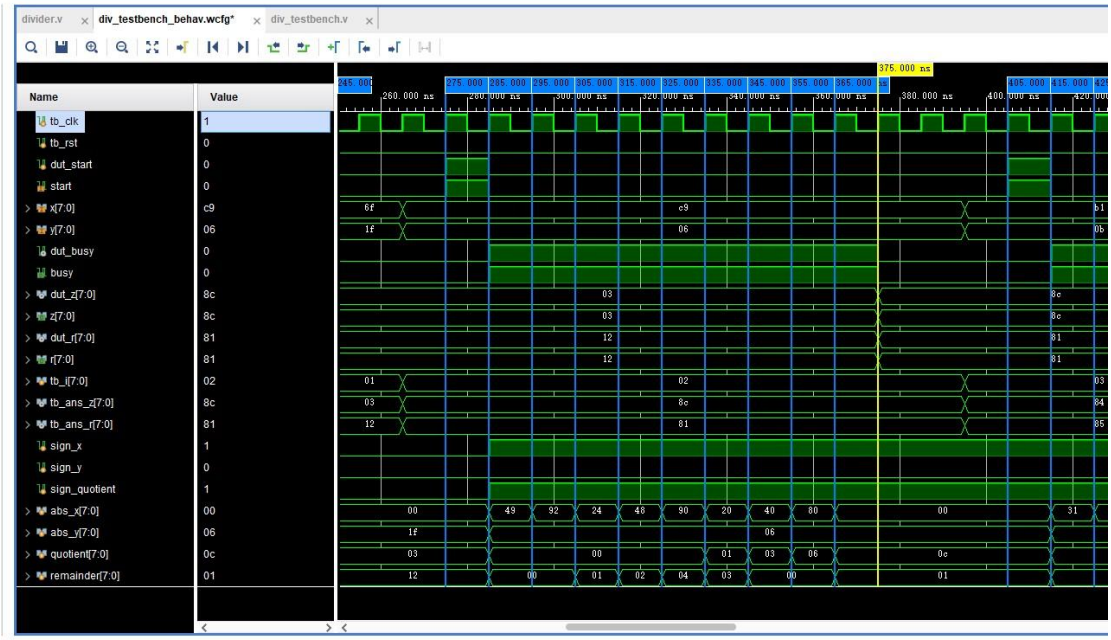
在 145ns 时，start 信号变为高电平，x、y 的输入值有效（test1 情况下十六进制下 x=6f，y=1f；十进制下 x=111，y=31）。在下一个时钟上升沿信号到来（155ns）后，busy 信号变为高电平，sign_x 赋值为 x[7]（test1 情况下为 0）、 sign_y 赋值为 y[7]（test1 情况下为 0），sign_quotient 则由 x[7]和 y[7]异或得到（test1 情况下为 0），abs_x 赋值为{1'b0,x[6:0]}（此时用于记录 x*的补码，test1 情况下为 6f），abs_y 赋值为{1'b0,y[6:0]}（test1 情况下为 1f），count 赋值为 8，接下来进入 test1 的计算阶段（155ns 至 245ns）。

从 155ns 至 245ns 间，每一个时钟上升沿信号到来时，依照恢复余数法的思路，在利用 count 判断移位次数是否已达要求后，利用 remainder 不断记录左移过程中余数（符号位未最终确定）的值，利用 abs_x 不断记录左移过程中被除数 x 的值，利用 quotient 不断记录计算过程中商（符号位未最终确定）的值。每次左移后 count 会减一。

在此期间（155ns 至 245ns），remainder 的值依次为 00（155ns 时）、00（165ns 时）、01（175ns 时）、03（185ns 时）、06（195ns 时）、0d（205ns 时）、1b（215ns 时）、18（225ns 时）、12（235ns 时）、12（245ns 时）；abs_x 的值依次为 6f（155ns 时）、de（165ns 时）、bc（175ns 时）、78（185ns 时）、f0（195ns 时）、e0（205ns 时）、c0（215ns 时）、80（225ns 时）、00（235ns 时）、00（245ns 时）；quotient 的值依次为 00（155ns 时）、00（165ns 时）、00（175ns 时）、00（185ns 时）、00（195ns 时）、00（205ns 时）、00（215ns 时）、01（225ns 时）、03（235ns 时）、03（245ns 时）。

245ns 时，时钟上升沿信号到来，将 busy 信号由高电平恢复为低电平，等待下一次数据输入。同时，test1 的计算过程结束，将 quotient 的值连同之前所得的符号位 sign_quotient 截取给 z 作为商（z 赋值为{sign_quotient,quotient[6:0]}），将 remainder 的值连同之前所得的符号位 sign_x 截取给 r 作为余数（r 赋值为{sign_x,remainder[6:0]}）。此时 z 的值即为计算所得的商的结果：03，而正确的商结果 dut_z 也为 03，经比较二者相同（转为十进制后均为 3，符合题意）；此时 r 的值即为计算所得的余数的结果：12，而正确的余数结果 dut_r 也为 12，经比较二者相同（转为十进制后均为 18，符合题意），综上说明 test1 计算过程正确，可通过测试用例 test1。

（2）分析 test2：-73 /6 = -12 ... -1



在 275ns 时，start 信号变为高电平，x、y 的输入值有效（test2 情况下十六进制下 x=c9，y=06；十进制下 x= -73，y=6）。在下一个时钟上升沿信号到来（285ns）后，busy 信号变为高电平，sign_x 赋值为 x[7]（test2 情况下为 1）、 sign_y 赋值为 y[7]（test2 情况下为 0），sign_quotient 则由 x[7]和 y[7]异或得到（test2 情况下为 1），abs_x 赋值为{1'b0,x[6:0]}（此时用于记录 x*的补码，test2 情况下为 49），abs_y 赋值为{1'b0,y[6:0]}（test2 情况下为 06），count 赋值为 8，接下来进入 test2 的计算阶段（285ns 至 375ns）。

从 285ns 至 375ns 间，每一个时钟上升沿信号到来时，依照恢复余数法的思路，在利用 count 判断移位次数是否已达要求后，利用 remainder 不断记录左移过程中余数（符号位未最终确定）的值，利用 abs_x 不断记录左移过程中被除数 x 的值，利用 quotient 不断记录计算过程中商（符号位未最终确定）的值。每次左移后 count 会减一。

在此期间（285ns 至 375ns），remainder 的值依次为 00（285ns 时）、00（295ns 时）、01（305ns 时）、02（315ns 时）、04（325ns 时）、03（335ns 时）、00（345ns 时）、00（355ns 时）、01（365ns 时）、01（375ns 时）；abs_x 的值依次为 49（285ns 时）、92（295ns 时）、24（305ns 时）、48（315ns 时）、90（325ns 时）、20（335ns 时）、40（345ns 时）、80（355ns 时）、00（365ns 时）、00（375ns 时）；quotient 的值依次为 00（285ns 时）、00（295ns 时）、00（305ns 时）、00（315ns 时）、00（325ns 时）、01（335ns 时）、03（345ns 时）、06（355ns 时）、0c（365ns 时）、0c（375ns 时）。

375ns 时，时钟上升沿信号到来，test2 的计算过程结束，将 quotient 的值连同之前所得的符号位 sign_quotient 截取给 z 作为商（z 赋值为{sign_quotient,quotient[6:0]}），将 remainder 的值连同之前所得的符号位 sign_x 截取给 r 作为余数（r 赋值为{sign_x,remainder[6:0]}）。此时 z 的值即为计算所得的商的结果：8c，而正确的商结果 dut_z 也为 8c，经比较二者相同（转为十进制后均为-12，符合题意）；此时 r 的值即为

计算所得的余数的结果：81，而正确的余数结果 dut_r 也为 81，经比较二者相同（转为十进制后均为-1，符合题意），综上说明 test2 计算过程正确，可通过测试用例 test2。同时，将 busy 信号由高电平恢复为低电平，等待下一次数据输入。