哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

计算机组成原理实验指导书

马世禹

目录

实验1 原码除法器设计

1、实验目的

- 理解恢复余数除法器的原理，掌握恢复余数除法器的设计方法。
- 理解加减交替除法器的原理，掌握加减交替除法器的设计方法。

2、实验内容

- 设计一个8bit的整数原码恢复余数除法器，输入、输出均采用8bit原码表示。
- 设计一个8bit的整数原码加减交替除法器，输入、输出均采用8bit原码表示。

本实验实现时可采用Verilog的"+"运算符，但不可直接使用"/"和"%"运算符，需要使用恢复余数算法或加减交替算法实现除法器。

建议不要使用"-"实现减法，而采用补码加的方式。

3、实验原理

原码除法，符号位为单独处理

设$[x]_{原} = x_0x_1x_2 \dots x_n$

$[y]_{原} = y_0y_1y_2 \dots y_n$

$$[\frac{x}{y}]_{原} = (x_0 \oplus y_0)\frac{x_1x_2 \dots x_n}{y_1y_2 \dots y_n}$$

式中 $x_1x_2 \dots x_n$为x的绝对值，记作x^*；

$y_1y_2 \dots y_n$为y的绝对值，记作y^*；

即商的符号由被除数和除数的符号位"异或"运算求得，商的值由被除数和除数绝对值相除$\left(\frac{x^*}{y^*}\right)$求得。

3.1 补码

整数补码定义如下：

$$[x]_{补} = \begin{cases} 0, x & 2^n > x \geq 0 \\ 2^{n+1} + x & 0 > x \geq -2^n \ (mod 2^{n+1}) \end{cases}$$

式中x为真值，n为整数的位数；

示例

当x＝+1010时，$x_{补} = 01010$

当x＝-1010时，$x_{补} = 2^5 - 1 = 100000 - 1010 = 10110$

3.2 恢复余数法

算法步骤：

- 符号位单独处理，分别取除数和被除数绝对值进行运算（和原码两位乘一样，参与运算的是绝对值的补码）；
- 若余数（被除数）为正，表示够减，商上1，左移一位，减去$[y]_{补}$；

  若余数（被除数）为负，表示不够减，商上0，恢复余数（加上$[y]_{补}$），左移一位，

  加上 $[-y]_{补}$；
- 重复上一步骤n次；
- 若最后一步余数为负，需要恢复余数，否则不需要；

示例

被除数：x＝13（1101）、除数：y＝6（0110）

	被除数（余数）	商	操作
	00001101		
(1)	<- 00011010	0	左移一位
	+11010		加[-y]补
	11011	0	余数为负
	+ 0110		恢复余数，加[y]补
	000011010		
(2)	<- 00110100		左移一位
	+11010		加[-y]补
	11101	0	余数为负
	+ 0110		恢复余数，加[y]补
	000110100		
(3)	<- 01101000		左移一位
	+11010		加[-y]补
	00000	1	余数为正
(4)	<- 00010000		左移一位
	+11010		加[-y]补
	11011	0	余数为负
	+ 0110		恢复余数，加[y]补
余数	00001	00010	商

由以上运算可知：商为0010，余数为0001

3.3 加减交替法

算法步骤：

- 符号位单独处理，分别取除数和被除数绝对值进行运算（和原码两位乘一样，参与运算的是绝对值的补码）；
- 若余数（被除数）为正，商上1，左移一位，减去$[y]_{补}$；

  若余数（被除数）为负，商上0，左移一位，加上$[y]_{补}$；
- 重复上一步骤n次；

示例

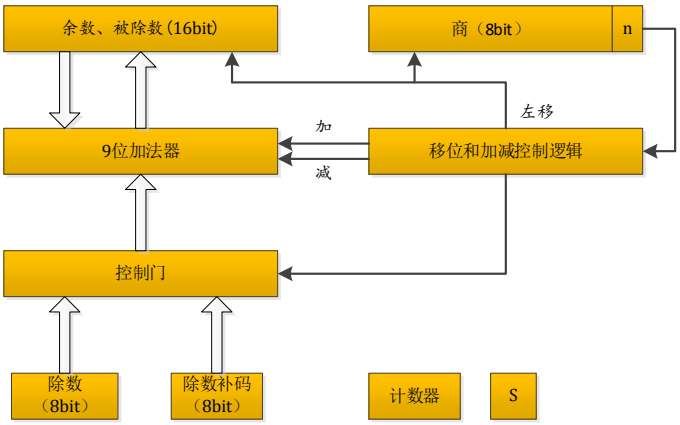
被除数：13（1101）、除数：6（0110）

	被除数（余数）	商	操作
	00001101		
(1)	<- 00011010	0	左移一位
	+11010		加[-y]补
	11011	0	余数为负
(2)	<-101110100		左移一位
	+ 0110		加[y]补
	11101	0	余数为负
(3)	<-110101000		左移一位
	+ 0110		加[y]补
	00000	1	余数为正
(4)	<- 00010000		左移一位
	+11010		加[-y]补
	11011	0	余数为负
	+ 0110		加[y]补
余数	00001	00010	商

由以上运算可知：商为0010，余数为0001

3.4 实现框图

以加减交替法为例：



除法开始前，商寄存器清0，计数器中存放除数的位数n。

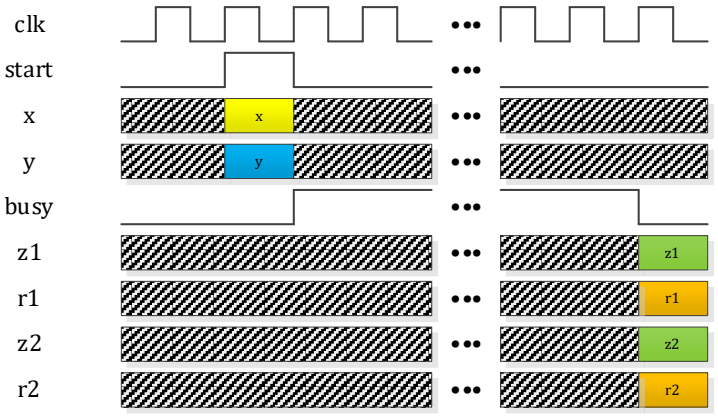
除法开始后，先通过异或运算求出商符S，然后根据商的末尾，判断是加除数还是减除数，之后再左移，这样重复n次后，即得到运算结果。

注：测试条件将满足0< |除数| ≤ |被除数|

3.5 除法器的 I/O定义

属性	名称	含义	位宽
输入	clk	时钟	1
输入	x	被除数	8
输入	y	除数	8
输入	start	输入就绪信号	1
输出	z1	恢复余数法-商	8
输出	r1	恢复余数法-余数	8
输出	z2	加减交替法-商	8
输出	r2	加减交替法-余数	8
输出	busy	忙标志信号	1

3.6 除法器的时序



如上图所示，当start信号拉高时，x、y信号有效 且只会存在一个周期，除法器接受到start时需要 立即拉高busy信号，直到除法器工作完成时拉低busy，同时输出运算结果z1、z2、r1、r2，并保持至少一个周期。

注意

请注意，本实验要求，busy的周期不得超过10，即除法的运算周期不得过长，否则将被判定为无效实现。

移位和加操作可以在同一周期内完成，请充分利用好组合逻辑和assign语句。

4、实验步骤

4.1 项目框架

top (top.v) —— 顶层模块，模拟输入输出并进行正确性检查

div_rr (div_rr.v) —— 恢复余数法模块（需要完成）

div_as (div_as.v) —— 加减交替法模块（需要完成）

4.2 具体步骤

完成 div_rr.v

div_as.v