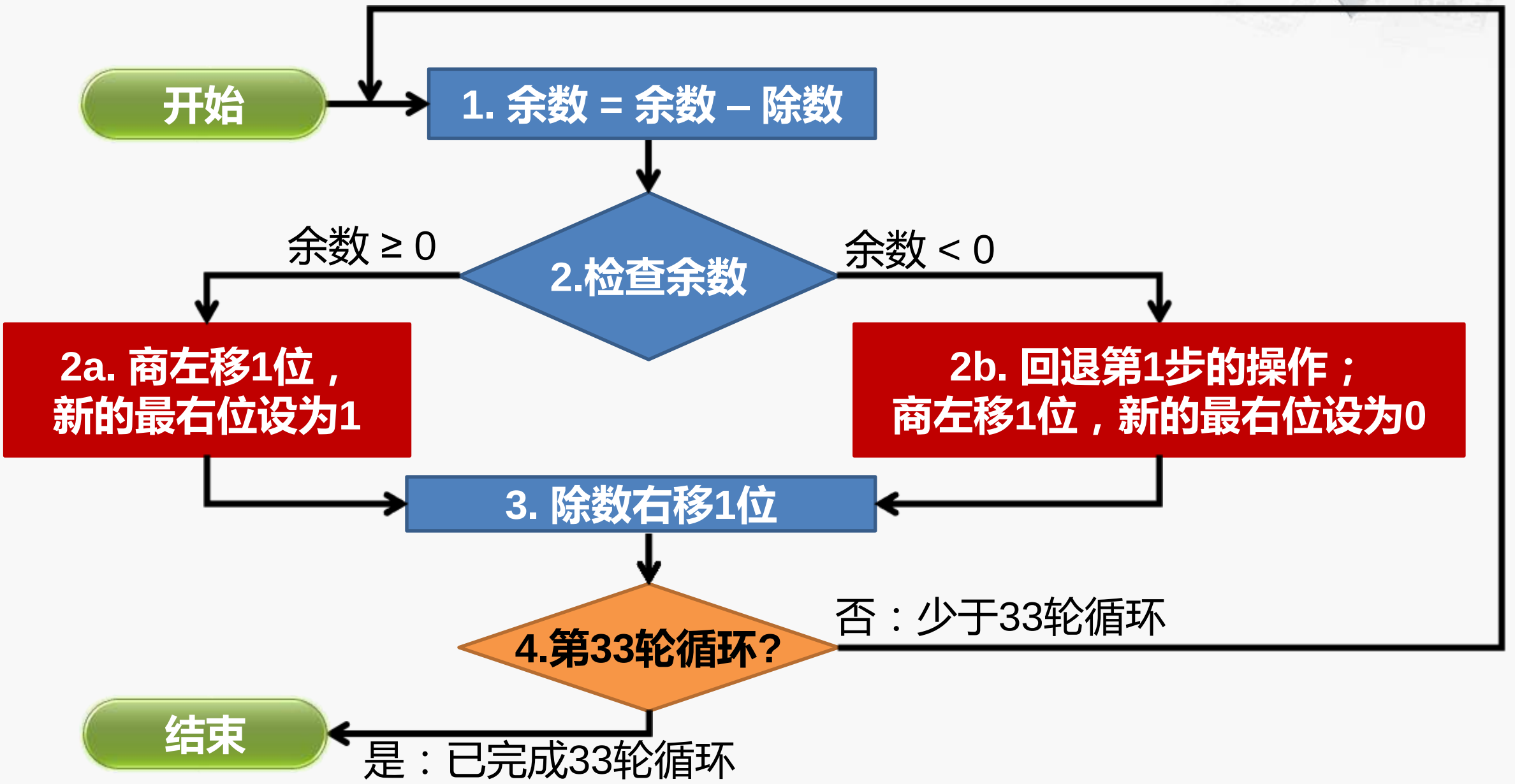
本节主题

除法器的实现

理解原理，会画除法器。

北京大学·慕课
计算机组成
制作人：陆俊林

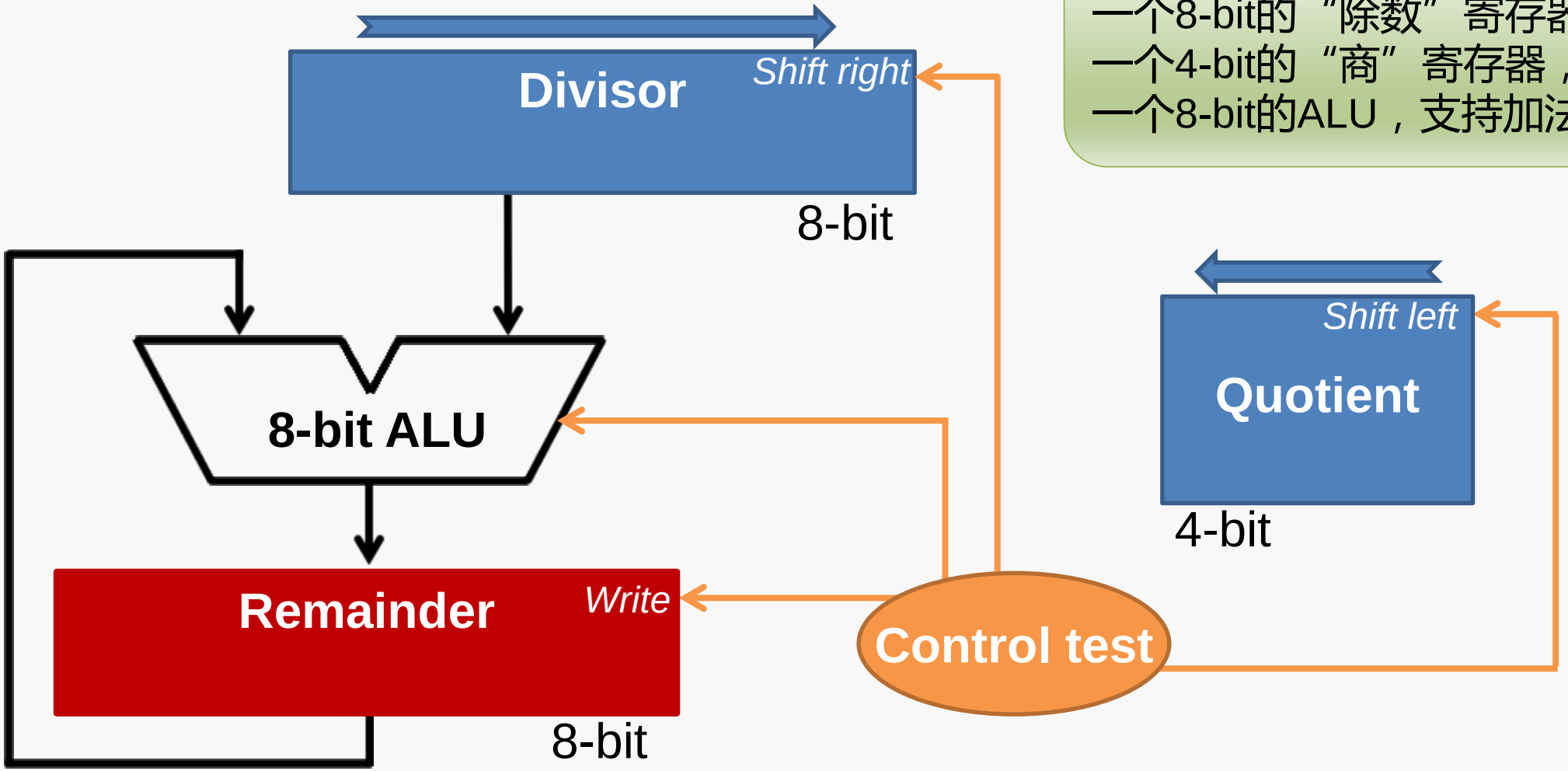
32-bit 除法器的工作流程图



4-bit 除法器的实现示例

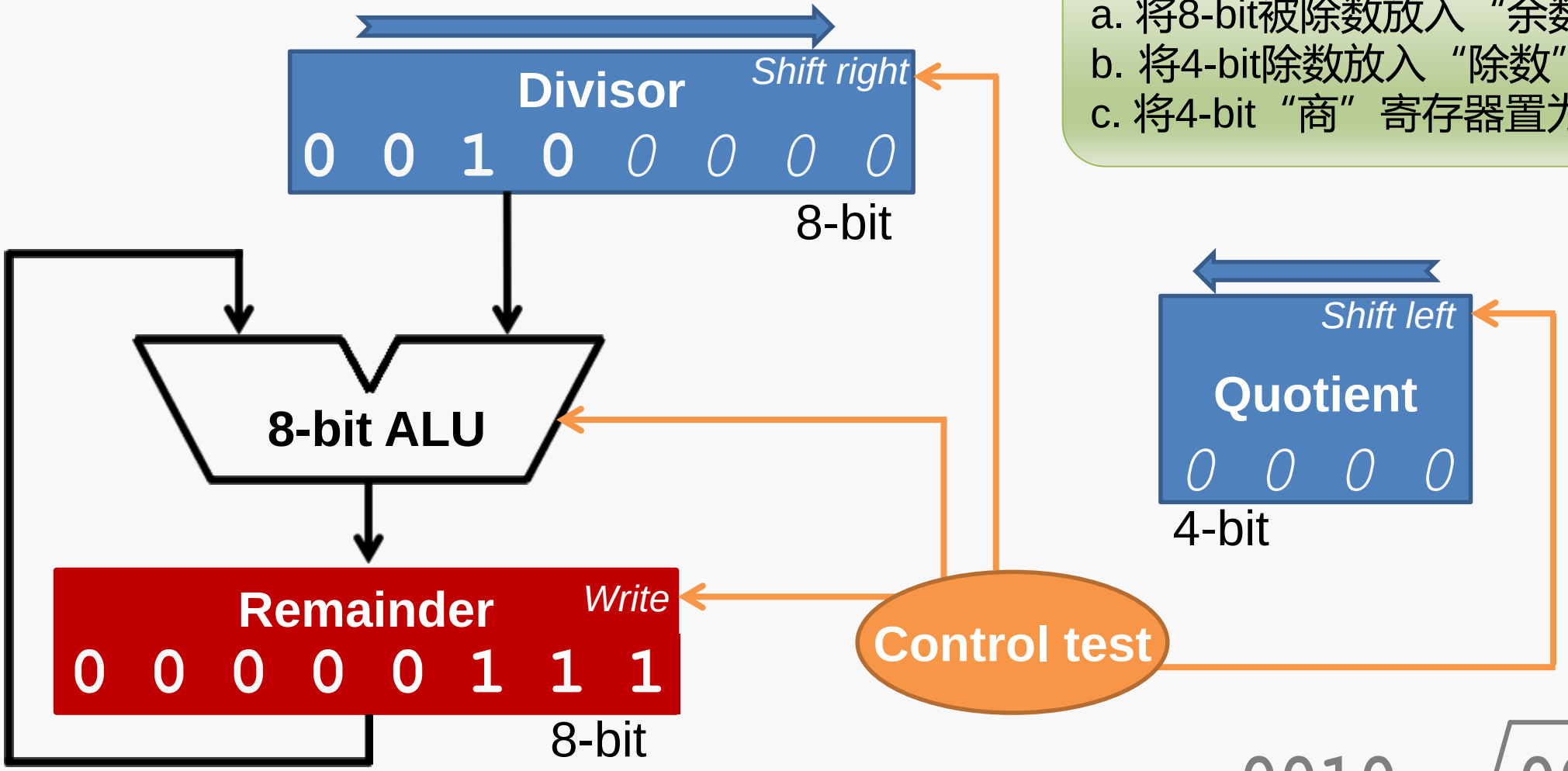
会画除法器。

一个8-bit的“余数”寄存器
一个8-bit的“除数”寄存器，带右移功能
一个4-bit的“商”寄存器，带左移功能
一个8-bit的ALU，支持加法和减法运算



除法器的工作过程（0）

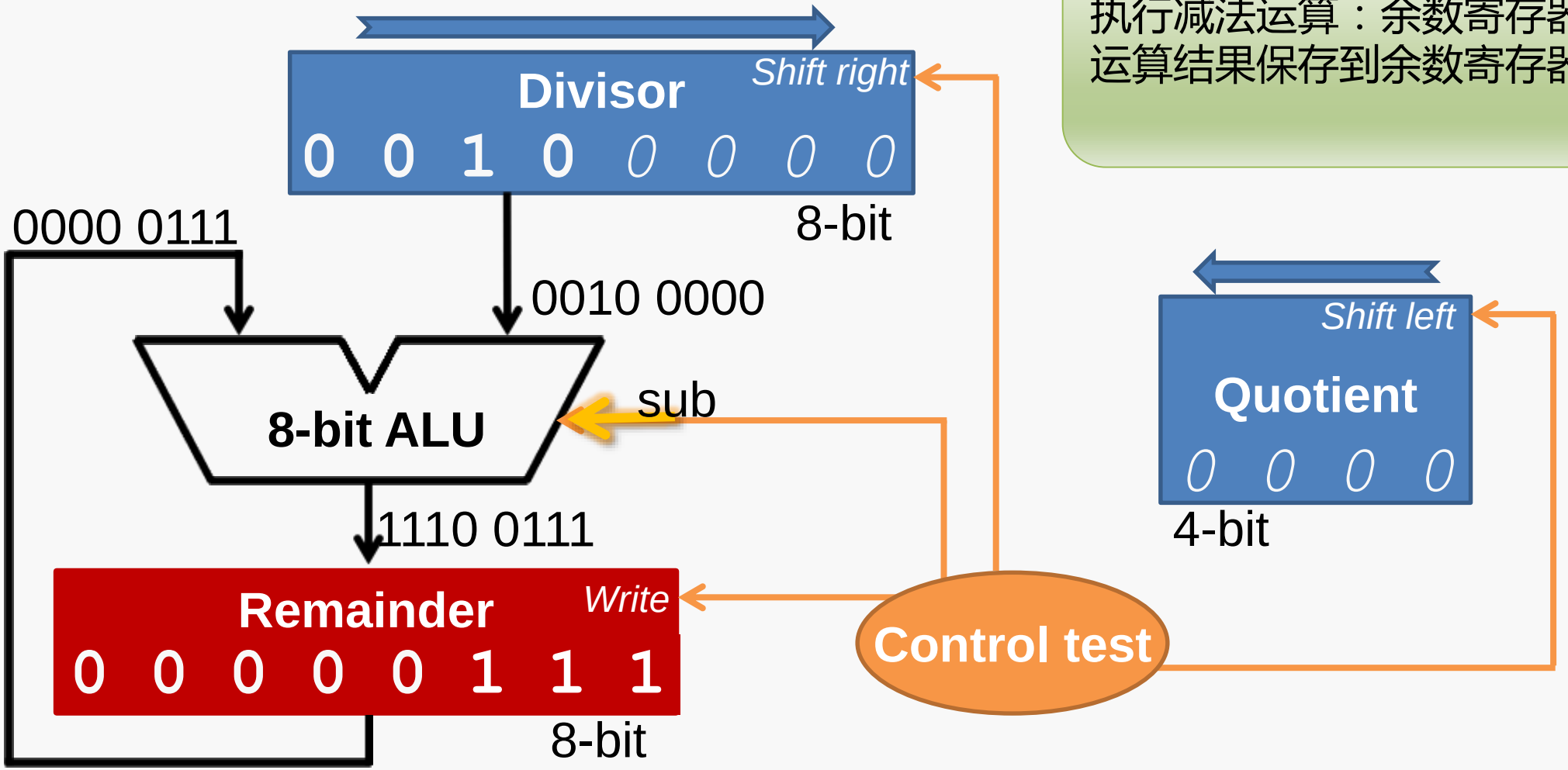
初始化：
a. 将8-bit被除数放入“余数”寄存器；
b. 将4-bit除数放入“除数”寄存器的高4-bit；
c. 将4-bit“商”寄存器置为零。



$0010_{two} \overline{)\ 00000111_{two}}$

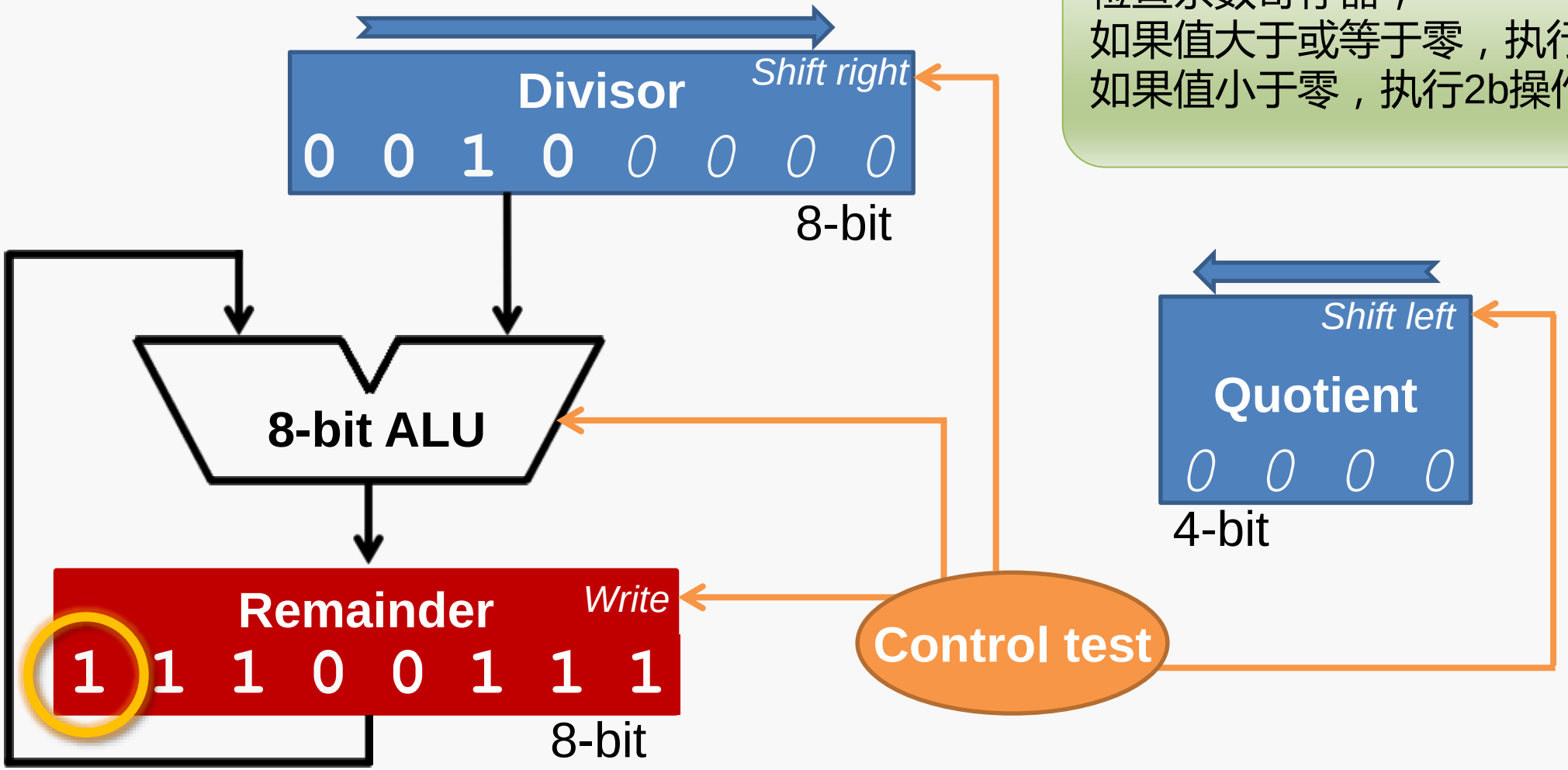
除法器的工作过程（1）

执行减法运算：余数寄存器 – 除数寄存器
运算结果保存到余数寄存器中



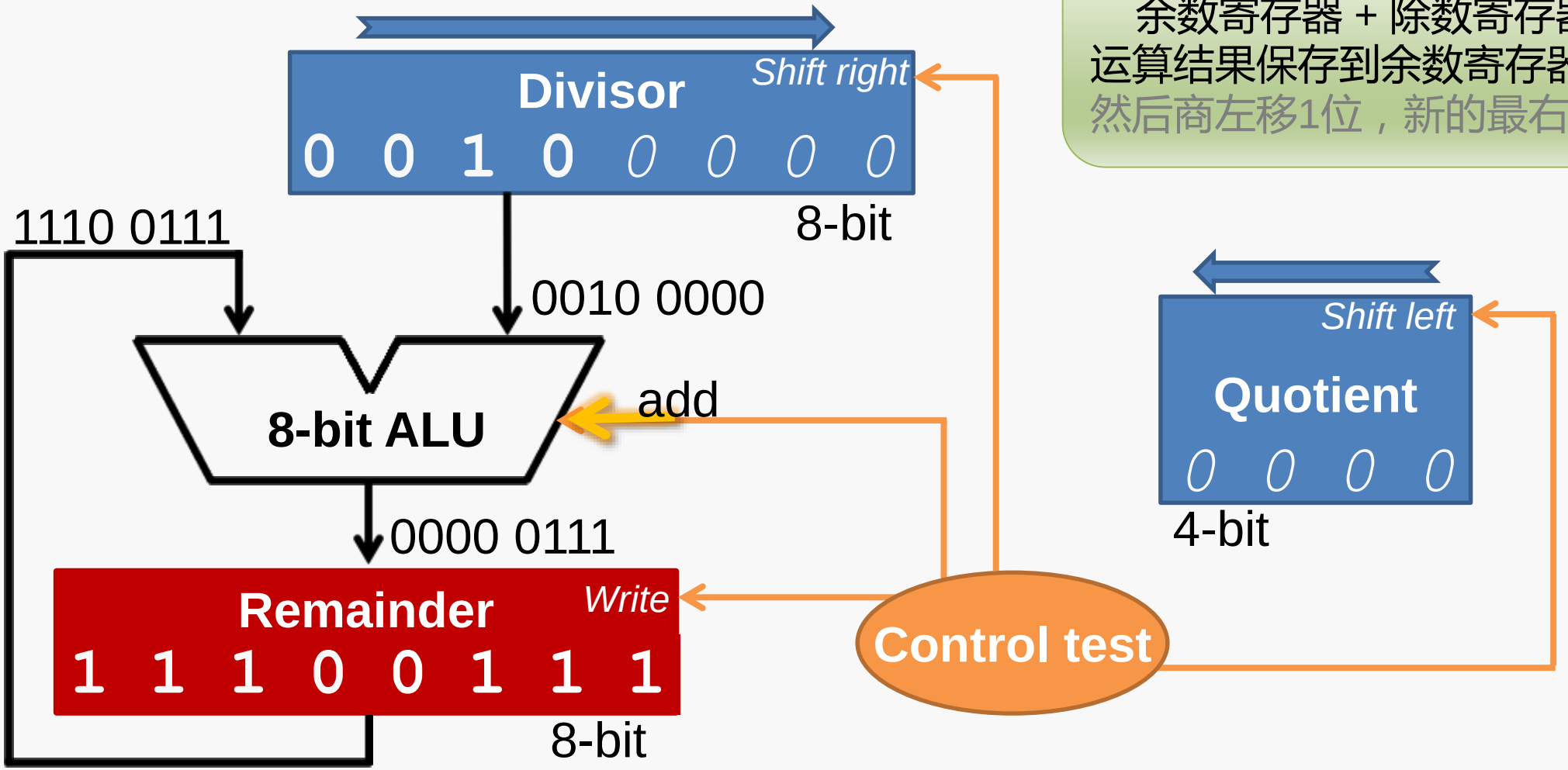
除法器的工作过程（2）

检查余数寄存器，
如果值大于或等于零，执行2a操作；
如果值小于零，执行2b操作。



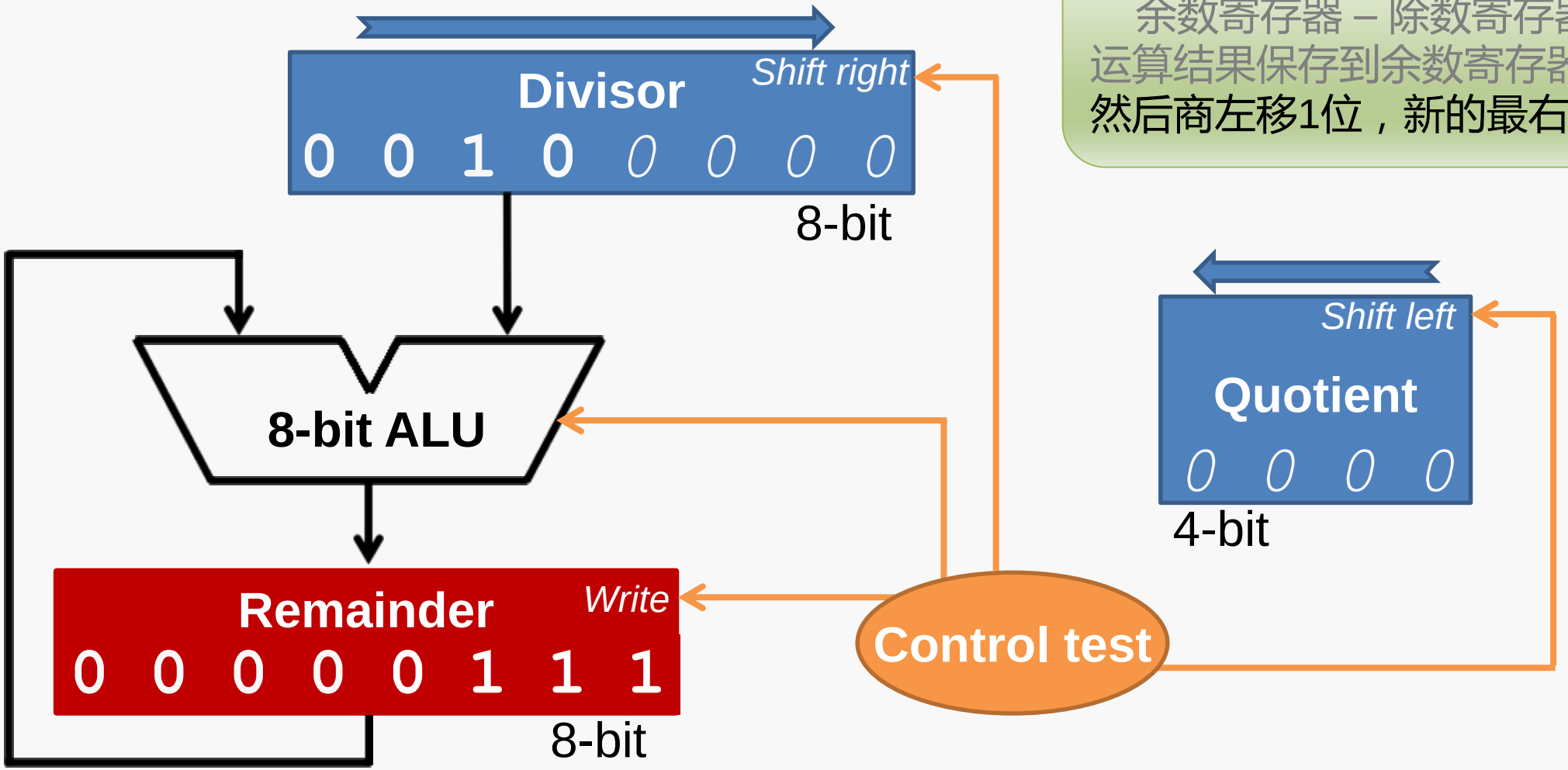
除法器的工作过程（2b）

首先回退第1步的操作，即执行加法运算：
余数寄存器 + 除数寄存器
运算结果保存到余数寄存器；
然后商左移1位，新的最右位设为0



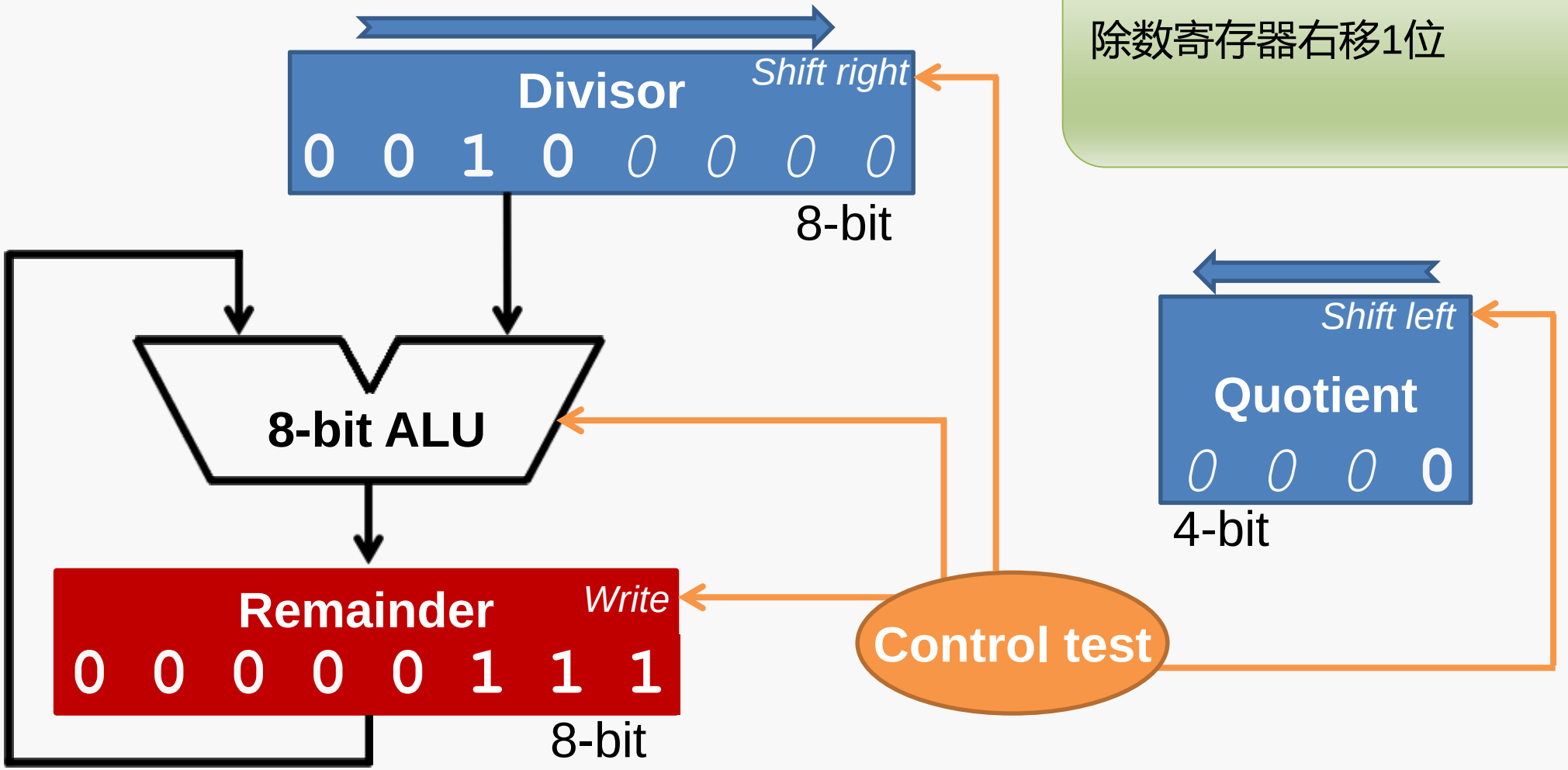
除法器的工作过程（2b）

首先回退第1步的操作，即执行加法运算：
余数寄存器 – 除数寄存器
运算结果保存到余数寄存器；
然后商左移1位，新的最右位设为0



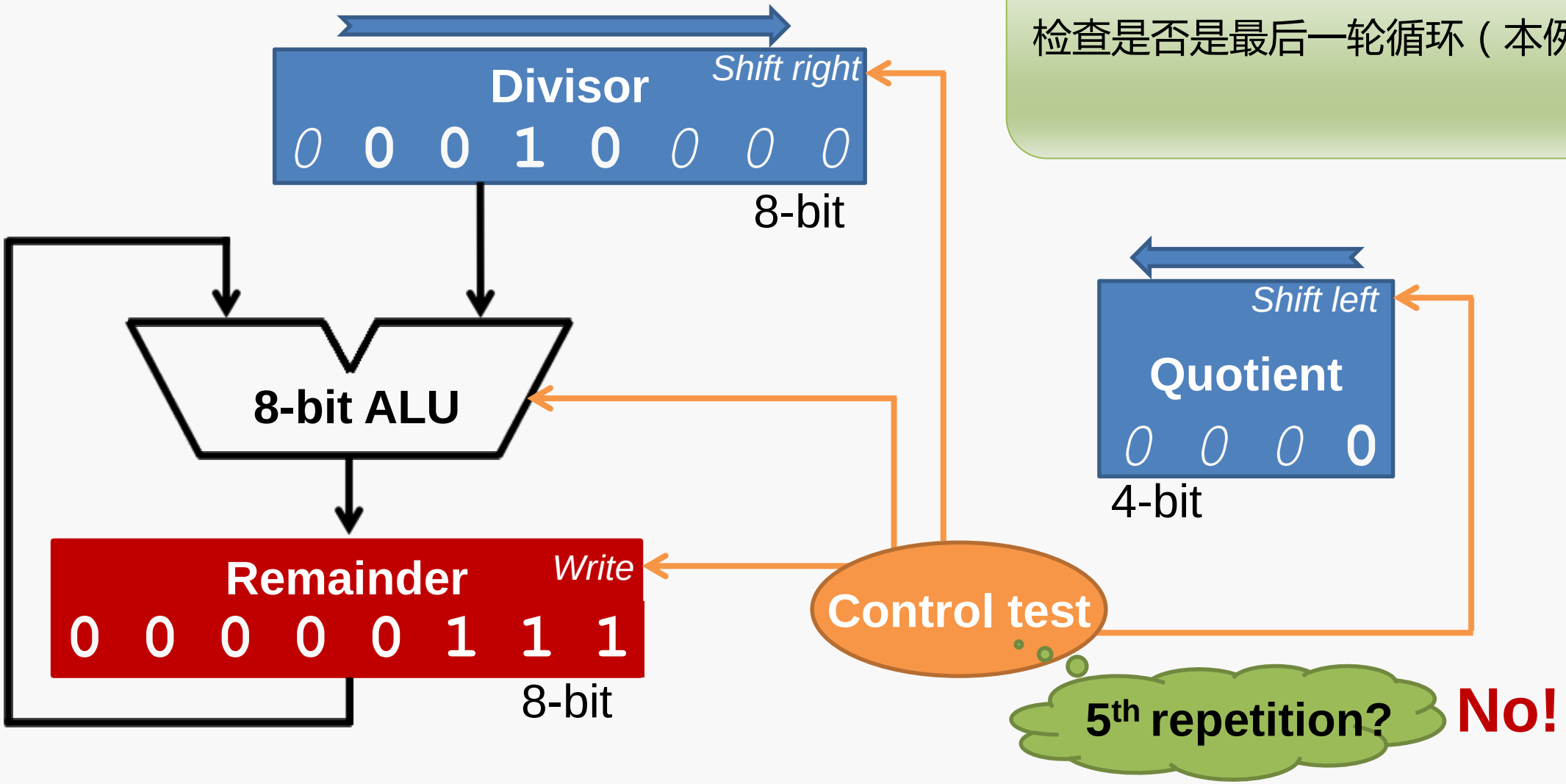
除法器的工作过程（3）

除数寄存器右移1位



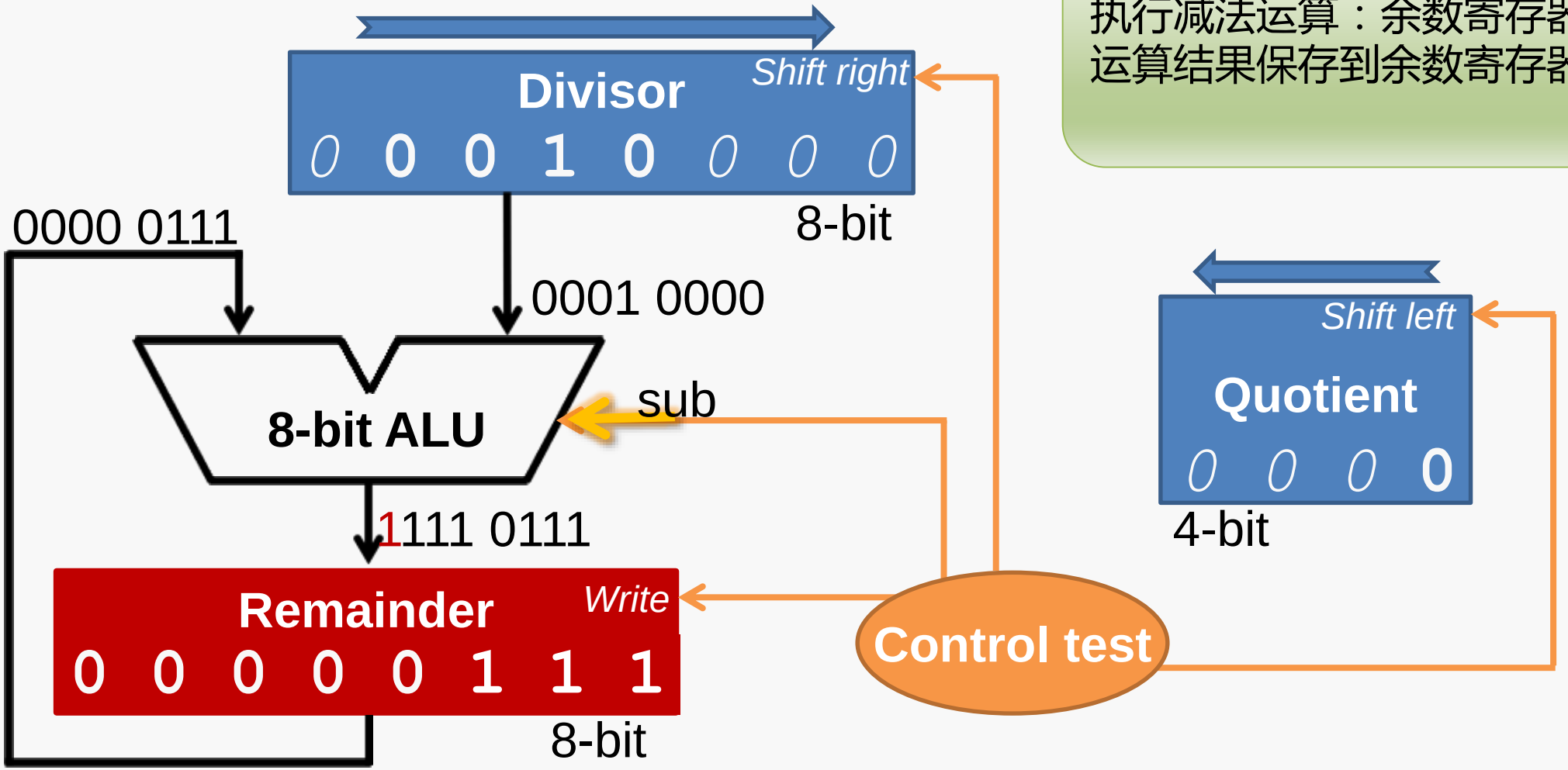
除法器的工作过程（4）

检查是否是最后一轮循环（本例为第五轮）



除法器的工作过程（1）　第二轮

执行减法运算：余数寄存器 – 除数寄存器
运算结果保存到余数寄存器中

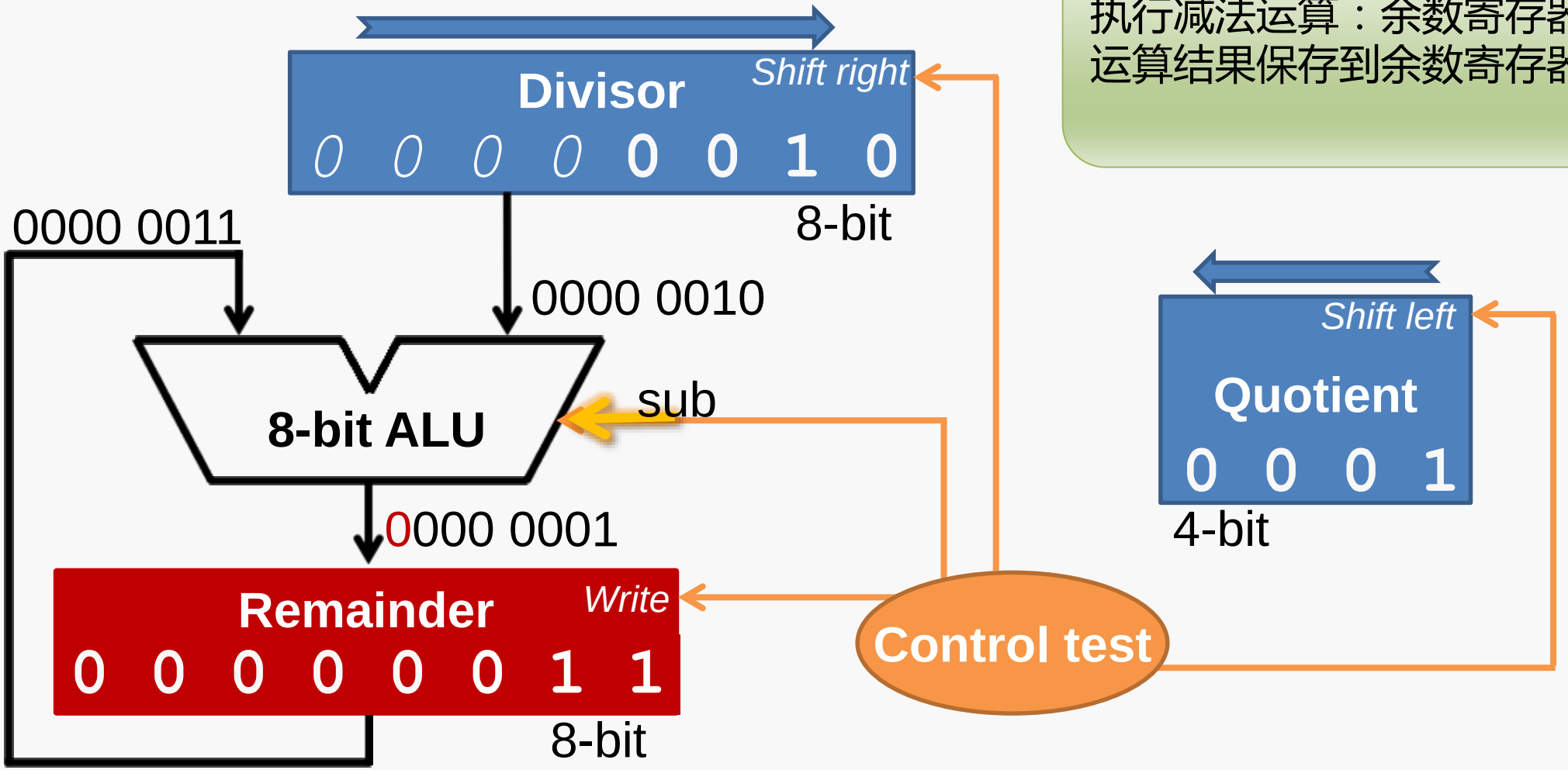


除法器的工作过程（第二轮～第四轮）

轮次	操作	商	除数	余数
二	1．余数=余数-除数	0000	0001 0000	1111 0111
	2b．余数=余数+除数，商左移补0	0000	0001 0000	0000 0111
	3．除数右移	0000	0000 1000	0000 0111
三	1．余数=余数-除数	0000	0000 1000	1111 1111
	2b．余数=余数+除数，商左移补0	0000	0000 1000	0000 0111
	3．除数右移	0000	0000 0100	0000 0111
四	1．余数=余数-除数	0000	0000 0100	0000 0011
	2a．商左移补1	0001	0000 0100	0000 0011
	3．除数右移	0001	0000 0010	0000 0011

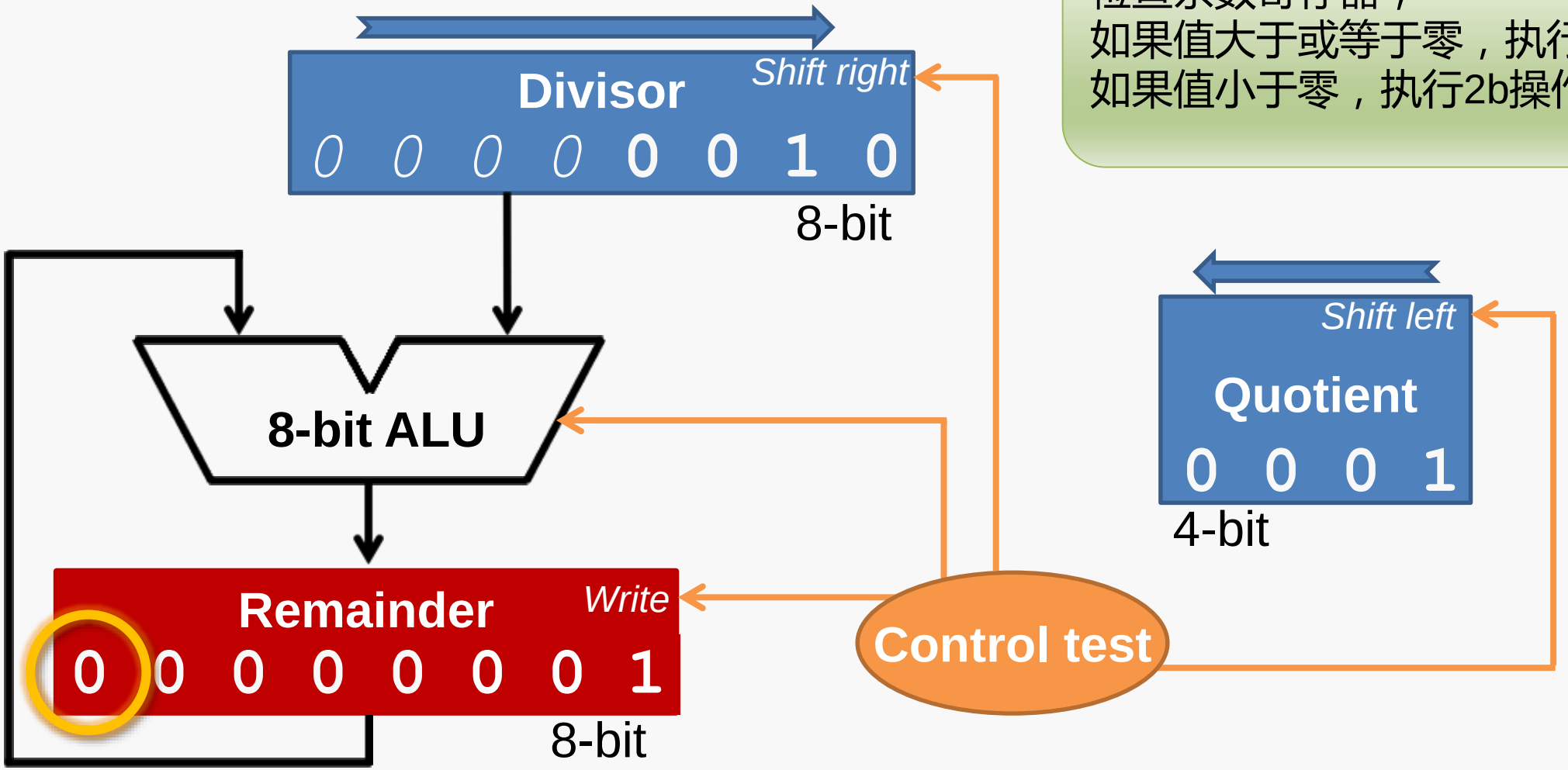
除法器的工作过程（1）　第五轮

执行减法运算：余数寄存器 – 除数寄存器
运算结果保存到余数寄存器中



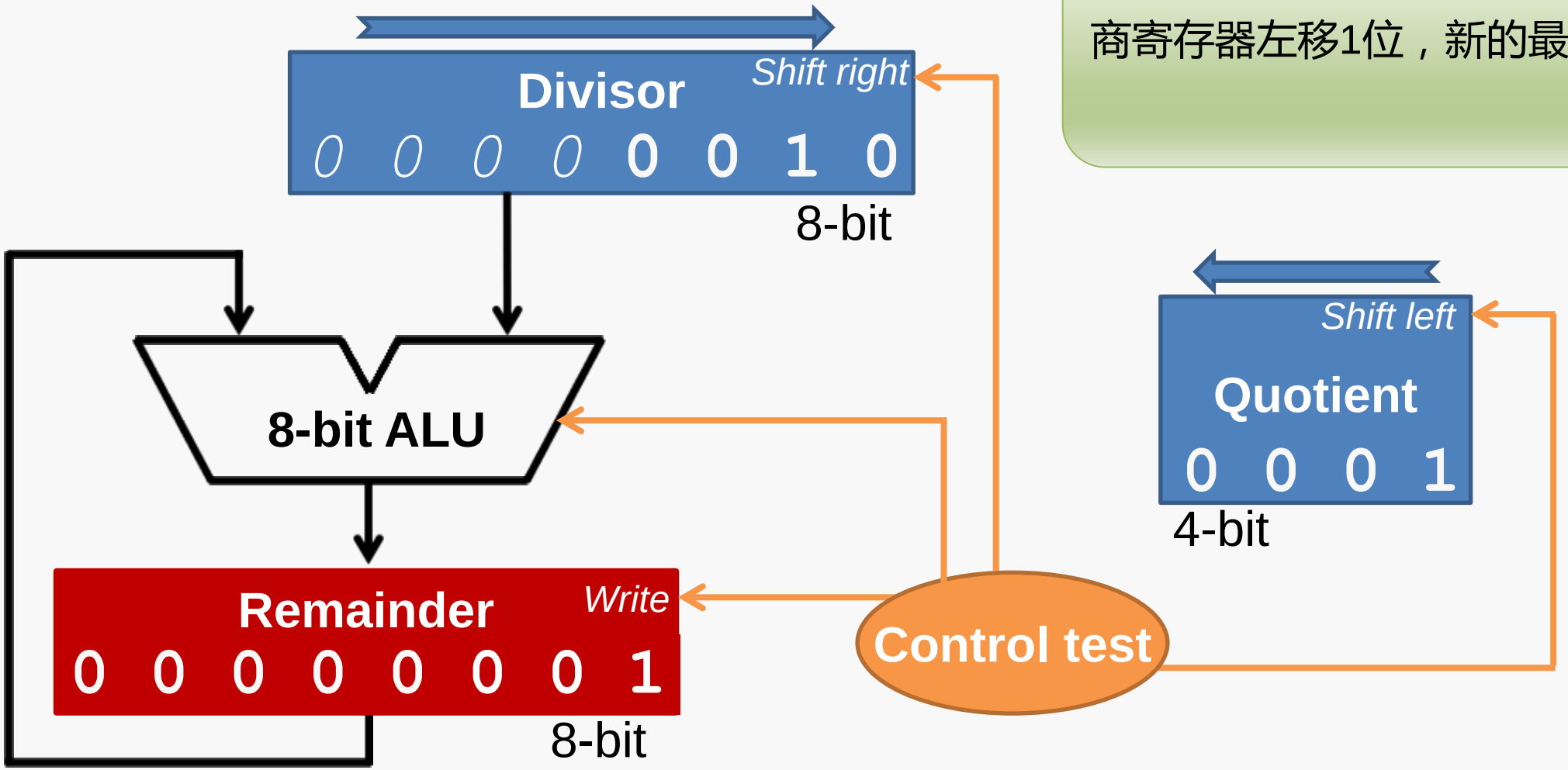
除法器的工作过程（2）　第五轮

检查余数寄存器，
如果值大于或等于零，执行2a操作；
如果值小于零，执行2b操作。



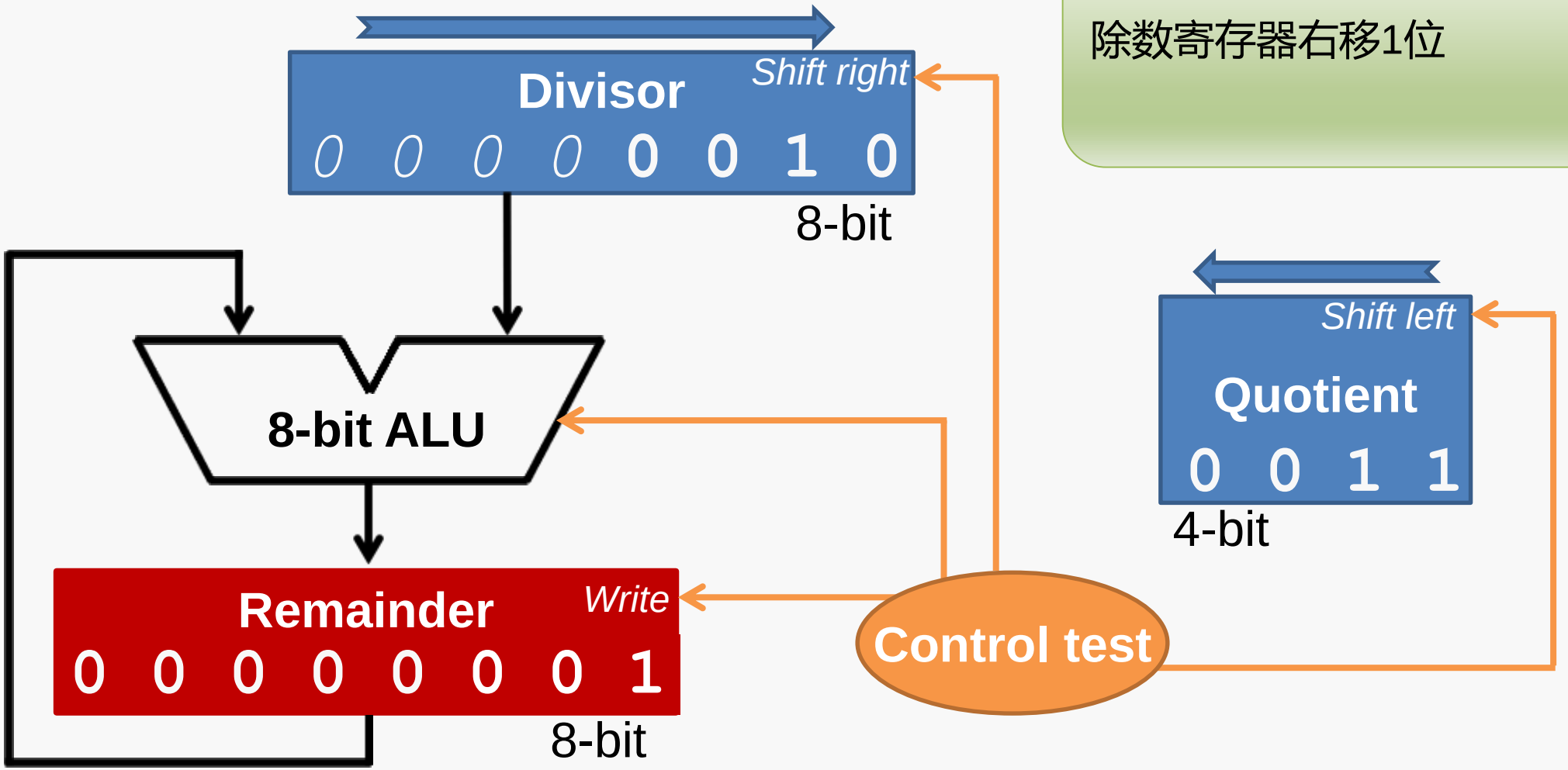
除法器的工作过程（2a） 第五轮

商寄存器左移1位，新的最右位设为1



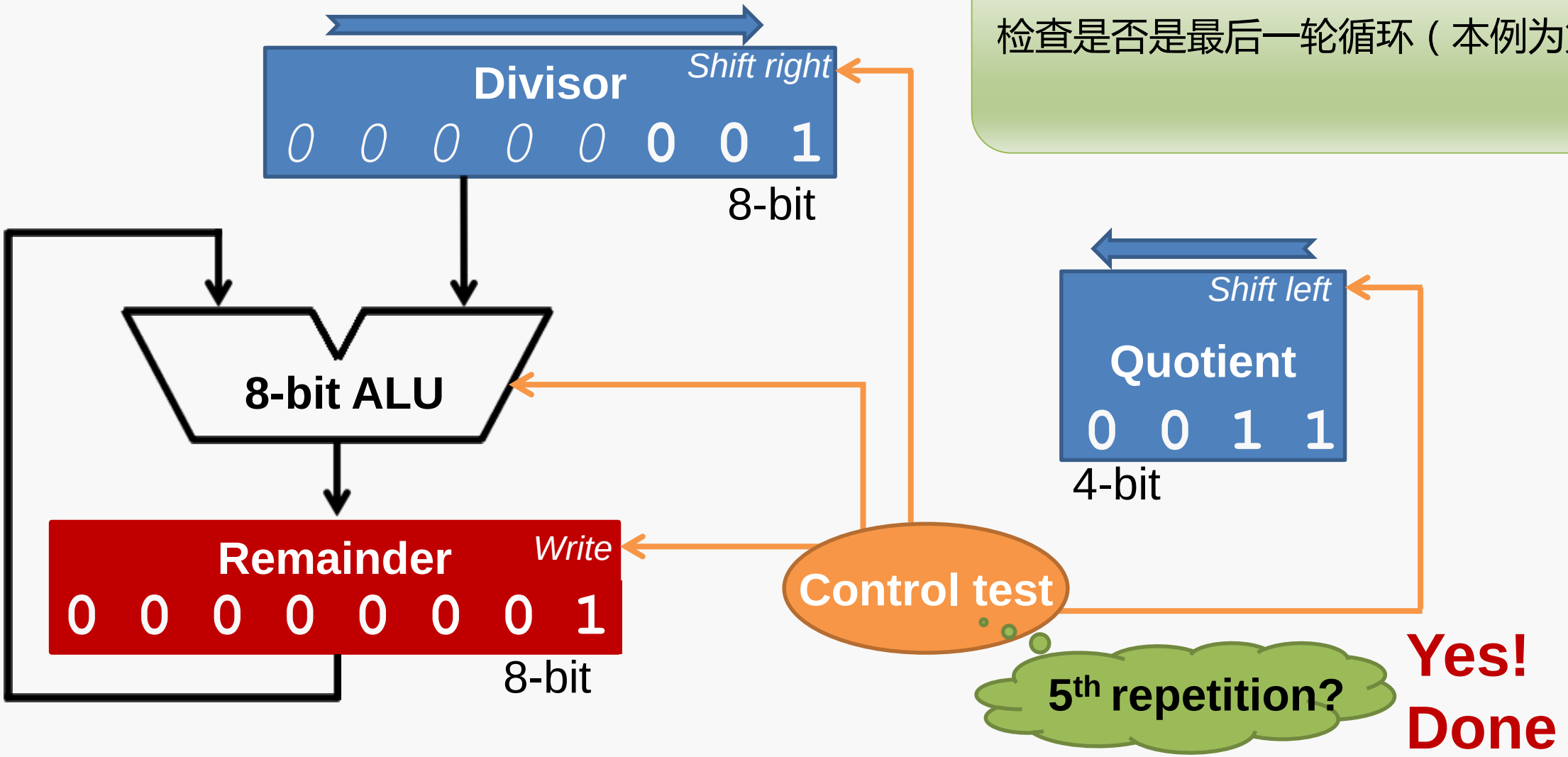
除法器的工作过程（3）　第五轮

除数寄存器右移1位



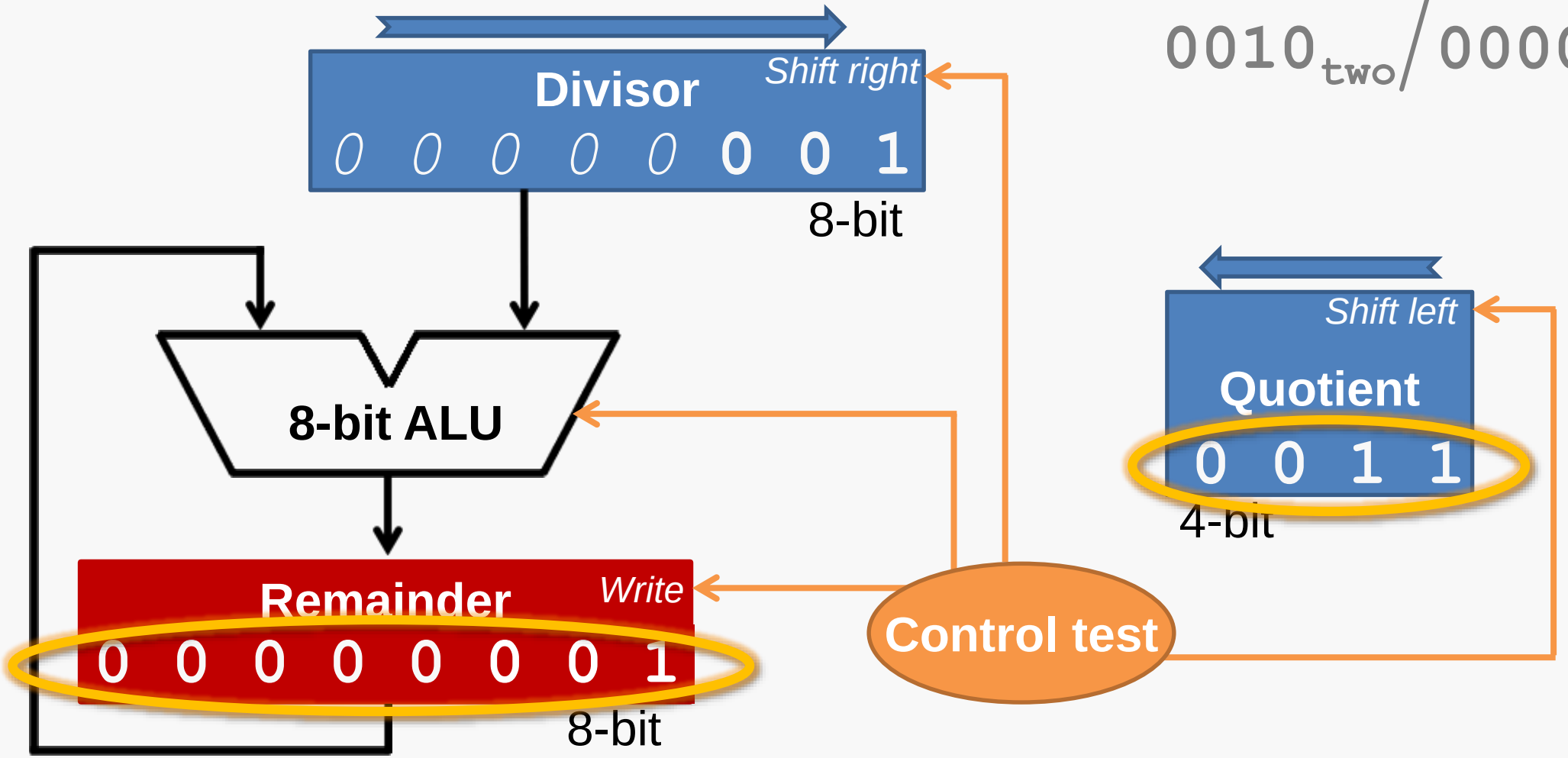
除法器的工作过程（4）　第五轮

检查是否是最后一轮循环（本例为第五轮）



除法器的运算结果

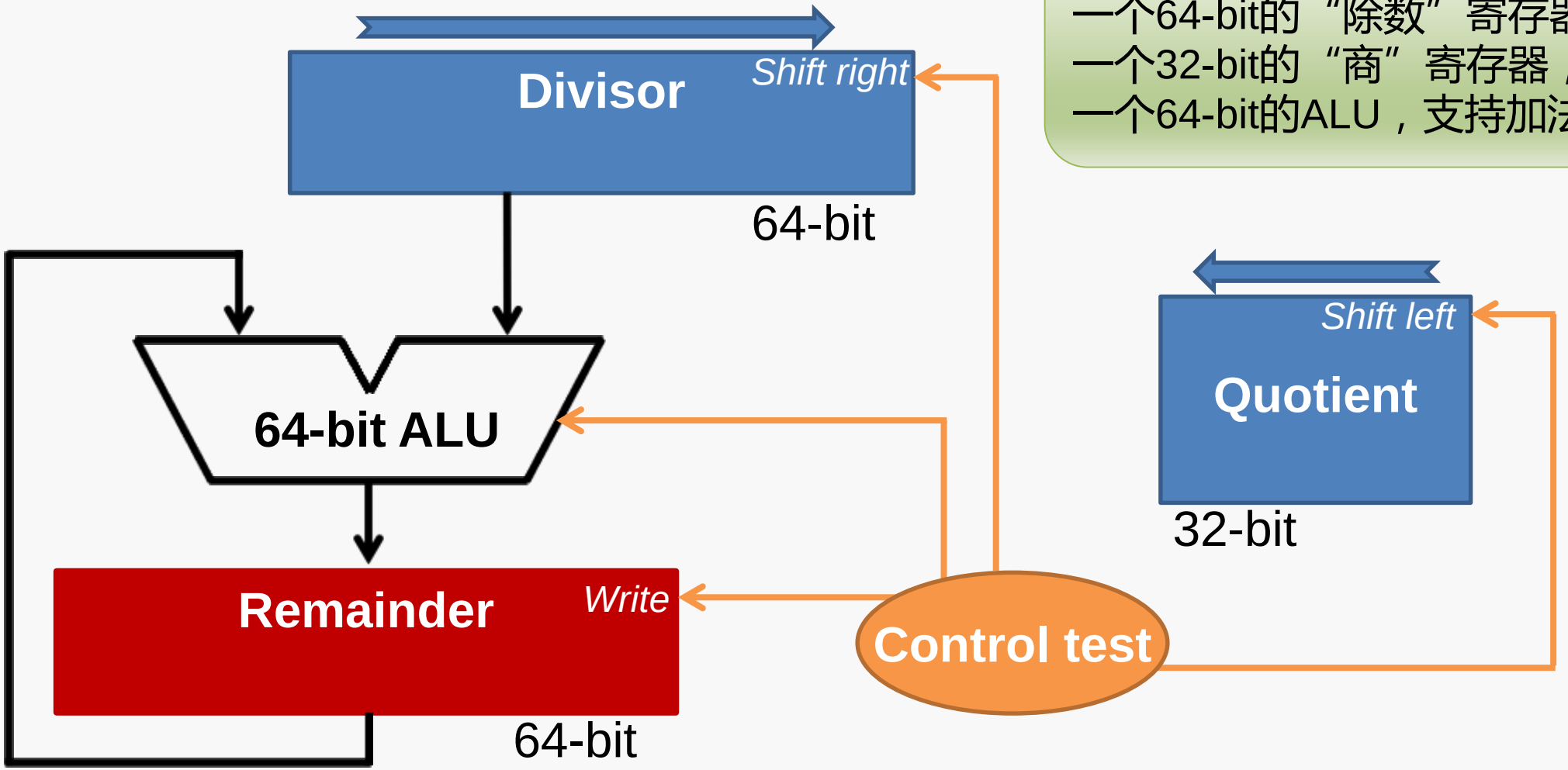
$0010_{two} \overline{)\ 00000111_{two}}$



32-bit 除法器的实现

会画除法器。（注意位数）
各寄存器的移位需掌握

一个64-bit的“余数”寄存器
一个64-bit的“除数”寄存器，带右移功能
一个32-bit的“商”寄存器，带左移功能
一个64-bit的ALU，支持加法和减法运算



本节小结

除法器的实现

北京大学·慕课
计算机组成
制作人：陆俊林