哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2024 春季

课程名称：计算机组成原理（实验）

实验名称：原码除法器设计

实验性质：设计型

实验学时：4 地点：T2506

学生班级：22 级 4 班

学生学号：220110430

学生姓名：吴梓滔

作业成绩：

实验与创新实践教育中心制

2024 年 5 月

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。

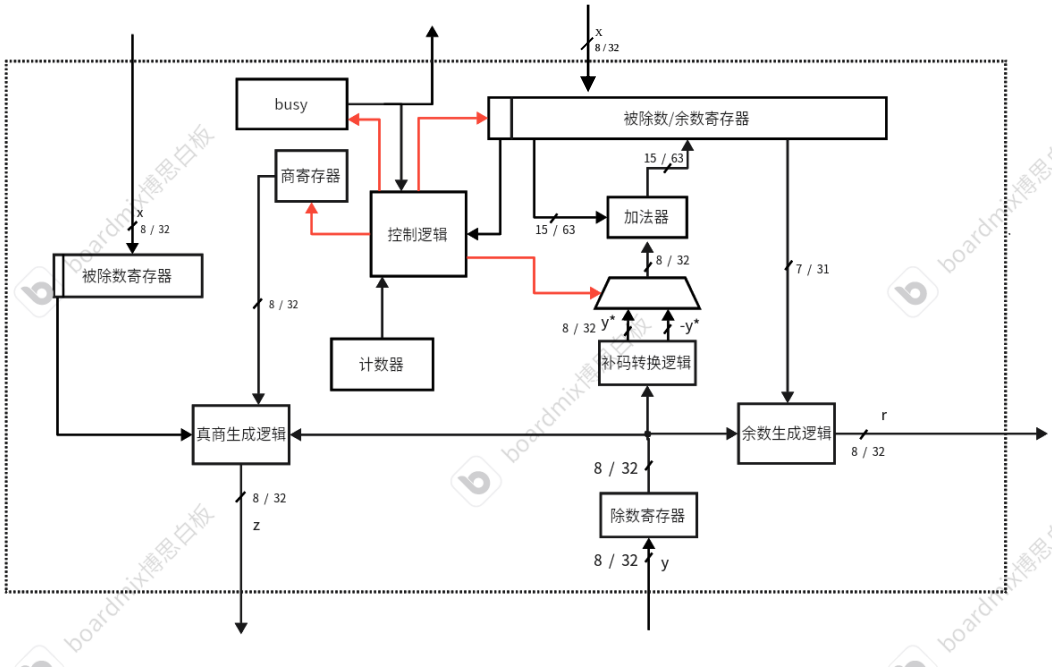

图 1　8 位/32 位除法器硬件模块框图

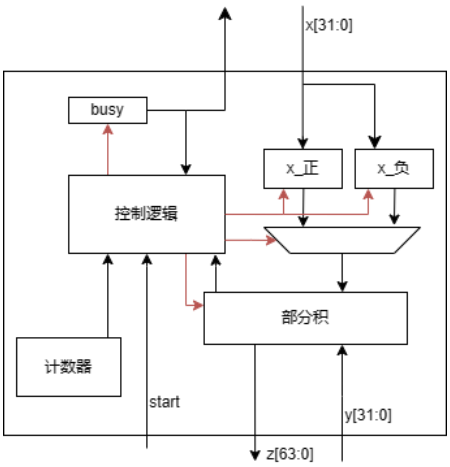

图 2 32 位乘法器硬件模块框图

1)　除法器功能（如图 1 所示）

a)　8 位/32 位除法器的硬件模块包括被除数、除数、商、余数、busy 的寄存器，计数器，加法器构成，一套封装起来的补码转换逻辑，输出的商和余数的生成逻辑和上商移位的控制逻辑组成。

b)　除数寄存器储存外部输入的除数，并将除数送入补码转换逻辑，生成两个正负

除数的补码信号。两个补码信号通过二路复用器，接受控制逻辑输出的信号控制，选择合适的一个与部分余数做加法。

c) 商寄存器受控制逻辑输出的信号控制，在每个时钟周期移位上商。最后的结果会送入真商的生成逻辑，和被除数、除数的最高位共同生成输出的商信号。

d) 计数器用于记录周期数，计数器的信号送入控制逻辑，用于控制逻辑的判断。

e) 被除数/余数寄存器接受控制逻辑的信号控制，每个周期会进行移位，并与除数的补码通过加法器相加，更新余数的值。同时最高位送入控制逻辑，用于控制逻辑的判断。余数寄存器的信号还与除数寄存器的信号共同生成最终输出的余数。

f) 被除数寄存器用于储存外部输入的被除数 x 信号，只用于参与输出的得数商信号的生成。

g) 控制逻辑读入计数器的信号、busy 信号和当前余数的最高位，用于判断下一上升沿余数寄存器、商寄存器的行为，同时会控制 busy 信号何时置 0。

2) 乘法器功能（如图 2 所示）

a) 乘法器的硬件模块包括部分积寄存器、计数器、控制逻辑、busy 的寄存器，以及被乘数 x 的正负补码寄存器。

b) x 的正负补码寄存器在 start 信号到来时，受控制逻辑控制，将外部的 x 信号读入并储存。之后持续输出，通过 2 路复用器，在控制逻辑控制下参与 booth 算法的执行。

c) 计数器输出信号接入到控制逻辑中，用于判断乘法的结束。

d) busy 信号受控制逻辑控制，一旦外部输入 start，就会受控制变成高电平。同时 busy 信号也输入控制逻辑中用于判断乘法仍在进行。

e) 部分积寄存器收到控制逻辑信号的控制，同时也输入到控制逻辑中用于 booth 算法的判断。在 start 信号到来后，部分积还用于存储输入的乘数 y。被乘数的正补码或者负补码可能输入并与部分积在低位相加。部分积寄存器直接输出到外部，在算法结束时，就输出了乘法计算的结果。

f) 控制逻辑会接受 start 信号，计数器输出信号，busy 信号，以及部分积，用于判断 booth 算法的下一步操作，以及算法何时结束。

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。*

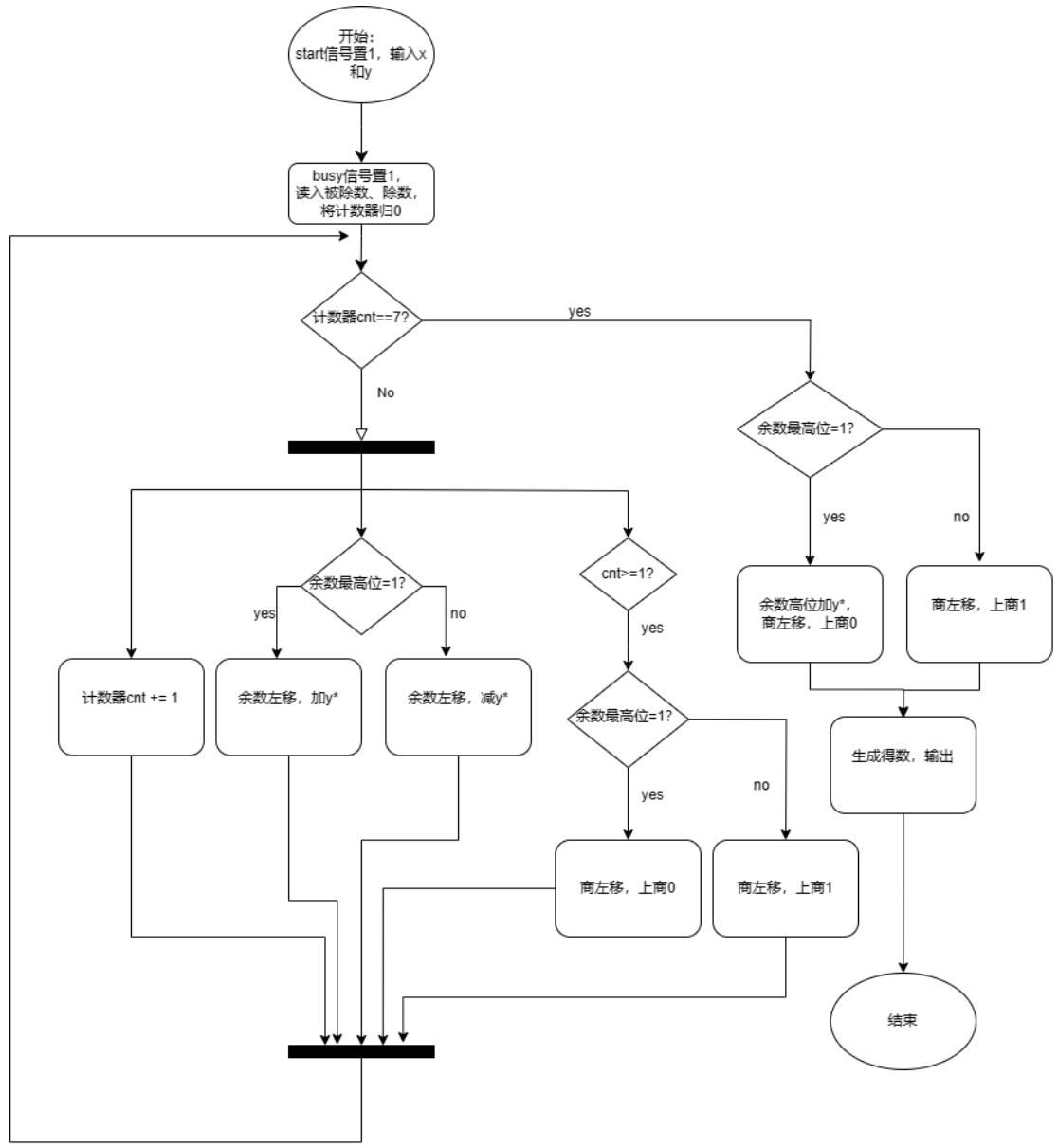

图 3 除法器算法流程图

以 8 位除法器为例。对于 32 为除法器，区别只在于计数器判断的条件为 cnt==31。

8 位除法器算法执行过程：

1) 开始时，start 信号置 1，并且给出 x 和 y 输入信号时，视为开始。
2) 下一周期，busy 信号置为 1，并且将输入的 x 和 y 保存到相应寄存器中，同时将计数器归 0。
3) 判断计数器的值是否为 7。
4) 如果计数器的值不为 7，则可以进入循环。同时做：计数器值加 1；判断余数最高位，若余数最高位为 1，将余数左移并加除数的绝对值，若余数最高位为 0，将余数左移并减去除数的绝对值；根据当前计数器的值，若计数器的值不为 0，则根据余数最高位，若余数最高位为 1，将商左移，上商 0，若余数最高位为 0，将商左移，上商 1。都做完后，回到(3)处判断。

5) 若计数器的值为 7，则进入最后一次操作。判断余数最高位，若余数最高位为 1，将余数加上 y*，并将商左移，上商 0。否则只将商左移，上商 1。之后生成得数并输出，包括将 busy 信号恢复成 0，算法结束。

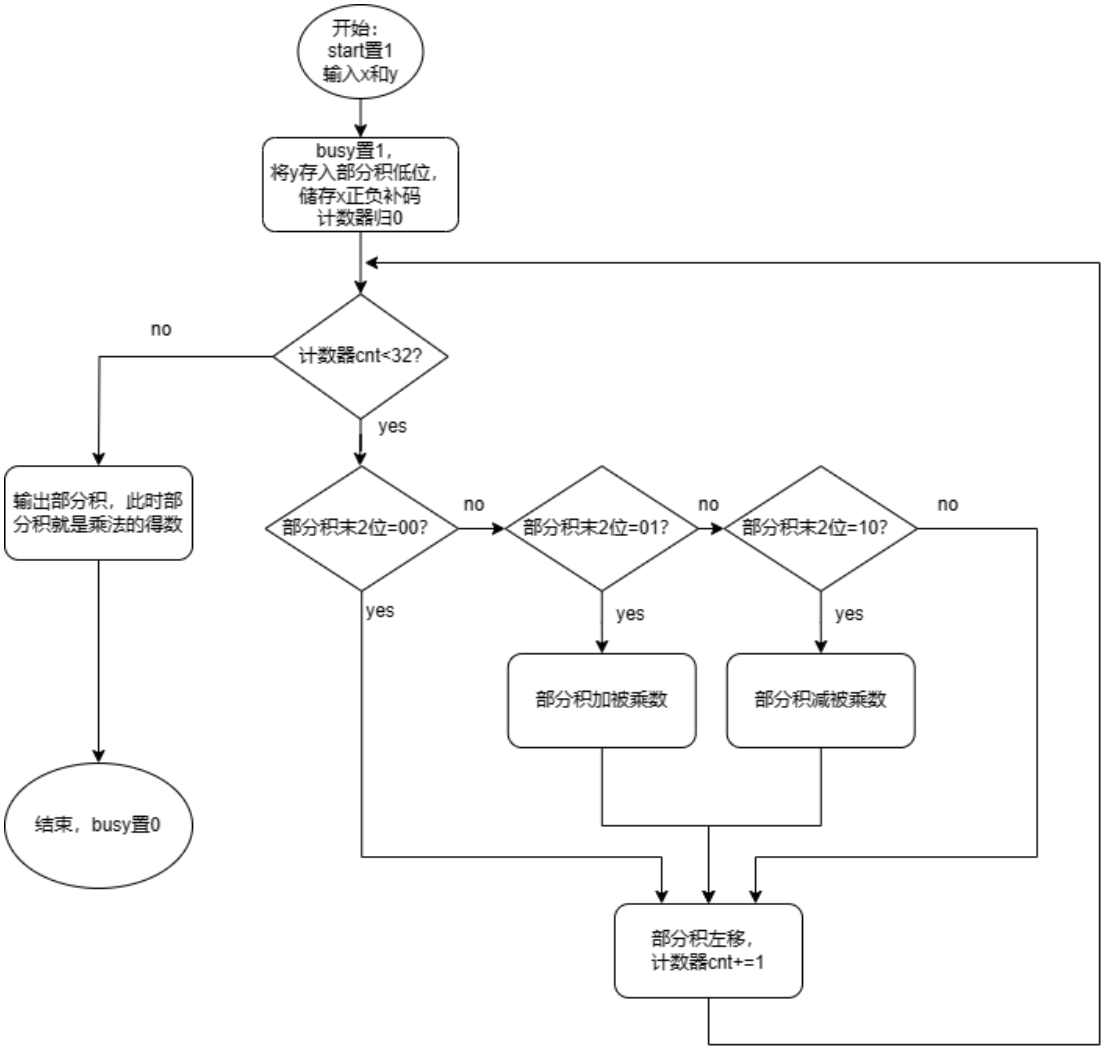

图 4 乘法器算法流程图

32 位乘法器算法执行过程：

1) 开始时，start 置 1 并输入 x 和 y 信号，视为算法开始。
2) 下一周期，busy 信号置 1，将 y 信号直接储存到部分积寄存器的低位，将 x 正负补码分别储存，同时将计数器归 0。
3) 判断计数器 cnt 的值是否小于 32。
4) 若计数器的值小于 32，则进入循环。判断部分积加上一个辅助标志位，所得到的末 2 位的值。若为 01，则需要部分积加被乘数。若为 10，则需要部分积减去被乘数。若为 00 或 11，则不需要额外操作。之后，都将部分积左移，同时计数器的值 cnt 加 1。之后回到步骤（3）。
5) 若计数器的值已经为 32，那么此时说明乘法运算已经完成，部分积就是乘法最后的得数。可以将 busy 置 0，算法结束。

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。

图 5(a)　32 位除法器测试用例 1

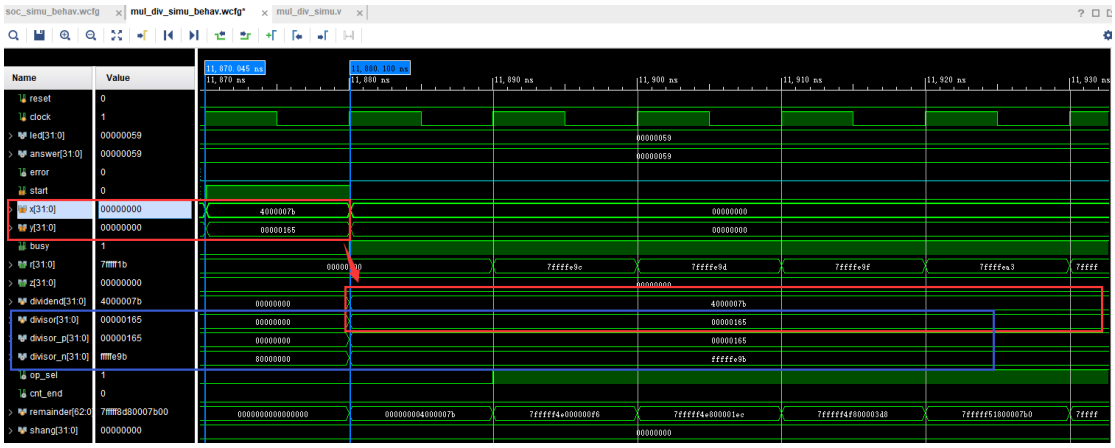

图 5(b) 对输入信号的储存

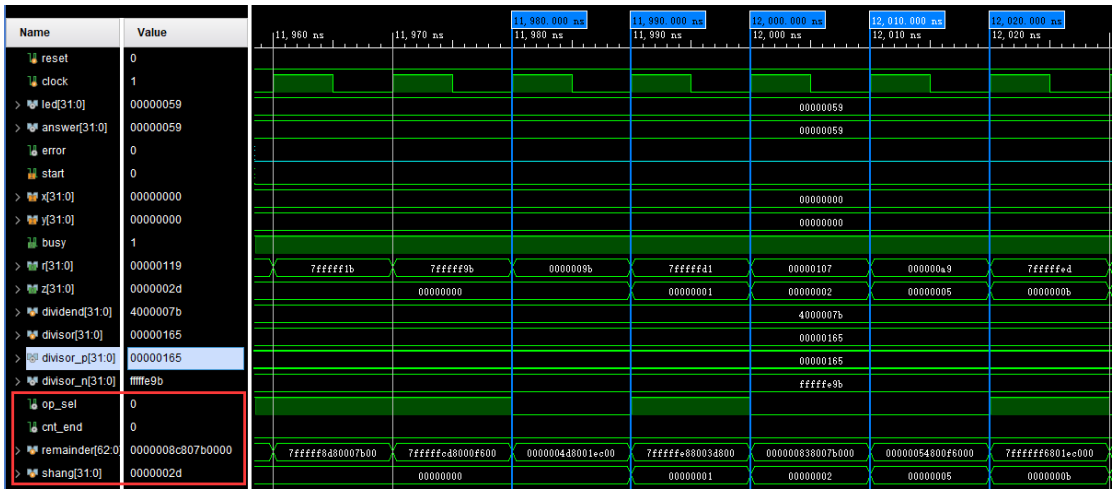

图 5(c) 加减交替法上商

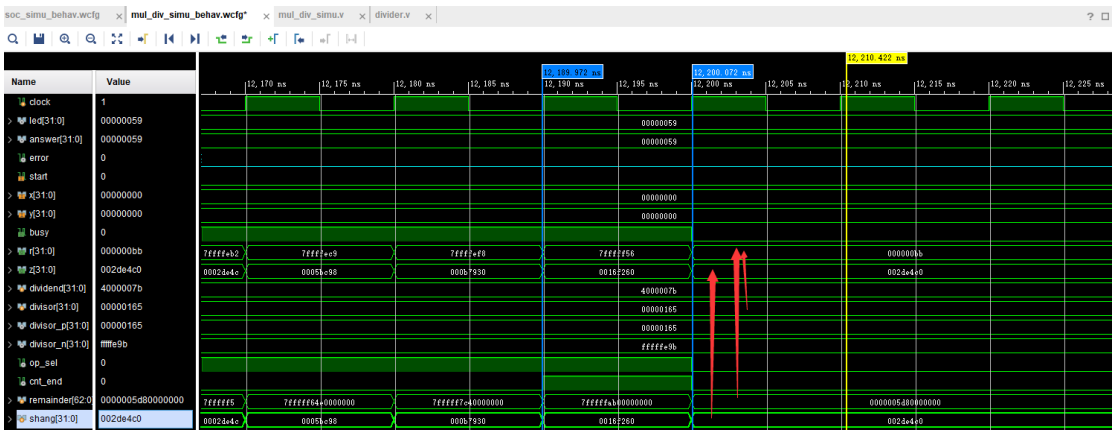

图 5(d) 计算完成的时刻

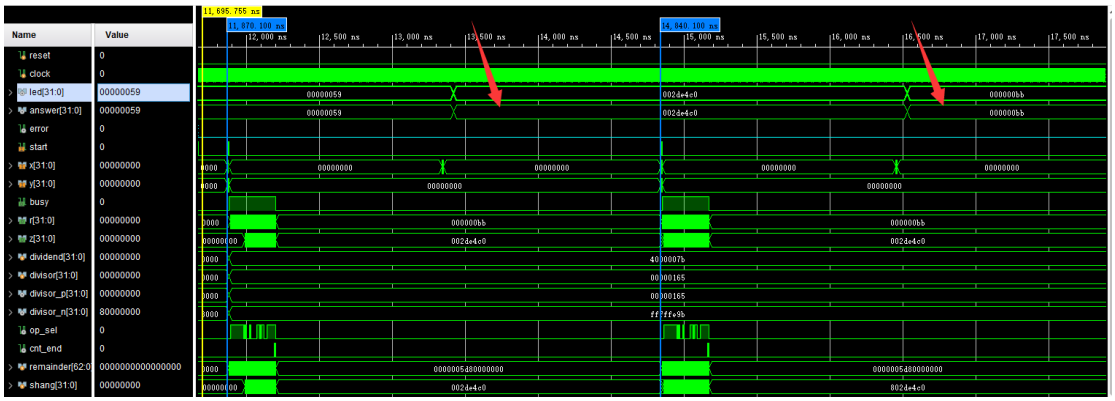

图 5(e) 计算结果正确

仿真波形中，所涉及到的信号有：被除数寄存器 dividend，除数原码的寄存器 divisor，部分余数的寄存器 remainder，商寄存器 shang，最终结果的信号为余数 r 和商 z。过程中辅助判断的信号 op_sel，指示当前余数的符号，0 表示当前余数为正。

1) 在第 1 个标记点的时钟上升沿处，start 信号更新为 1，被除数和除数的输入信号 x 和 y 更新成对应值。

2) 在第 2 个标记点的时钟上升沿处，start 信号更新为 0，x 和 y 信号更新为 0，这三个输入持续一个周期。在该时刻，寄存器 dividend 和 divisor 分别更新成 x 和 y 的值。并维持到下一次有有效输入信号时。在一次完整计算过程中，不会再变化。如图 5(b)所示。与此同时，被除数的值也存储到部分余数 remainder 寄存器的低位。同时，除数存储到 divisor 后，divisor_p 和 divisor_n 会同时更新成除数的正负值补码。如图 5(b)中蓝色框所示。

3) 在图 5（b）中，op_sel 信号有一次上升沿。op_sel 信号为 0 时，remainder 应减去余数的绝对值。在 op_sel 信号上升沿的时刻，也是 clk 信号的上升沿。此时 remainder 信号从 4000007b 变成 7fffff4e000000f6，正好是在高位部分经移位并加上了 divisor_n 信号，最低 8 位从 7b 变成 f6，也恰好是左移一位得到的结果，该步骤符合预期。说明余数部分的加减交替法在正常进行。

4) 在 op_sel 信号上升沿的时刻，shang 仍保持为 0，根据设计，op_sel 信号为 0 时，shang 会在最低位上 1，否则上 0，但是在计算的第一次移位时，不进行上商操作。因此

在该时钟上升沿，shang 保持全 0，符合预期。在图 5(b)内之后的每次时钟上升沿，op_sel 都为 1，shang 每次都移位并上商 0，保持全 0，符合预期。

5) 在图 5(c)中，展示了本次计算中间过程的几个时钟上升沿。在标记的第一个时钟上升沿到来时，op_sel=1,此时 shang 保持为 0，remaind 经移位并加上除数的绝对值，计算结果经验证是正确的。第二个时钟上升沿到来时，由于 op_sel=0,shang 信号左移并在最低位上商 1,同时余数更新。第三个时钟上升沿到来时,由于 op_sel=1, shang 信号左移并上商 0，从 1 变为 2，符合预期，同时余数更新。在第四个时钟上升沿到来时，op_sel=0, shang 信号左移并上商 1，从 2 变为 5，符合预期。因此该过程中，shang 和 remainder 信号均按照加减交替法的预期变化。

6) 图 5(d)显示计算结束的时刻。在最后一个周期里，cnt_end 信号亮起，表明移位次数够，余数不会再移位，在接下来这个时钟上升沿，商的移位上商操作仍然进行，余数只检查当前余数是否为负数，如果是负数会最后加一次除数绝对值。在结束时刻，busy 信号由 1 变成 0，shang 信号仍然完成左移并上 0 的操作，从 0016f260 变成 002de4c0，正好是左移一位，符合预期。remainder 信号正好是加上了 divisor_p 的结果。

7) 输出信号 z 与 r 会随 shang 和 remainder 同步变化。shang 信号的低位送入 z 信号，z 信号的最高位由被除数 dividend、除数 divisor 两个寄存器的最高 1 位异或得到，符合预期。remainder 信号除最高位外，剩余最高 31 位和 divisor 最高位共同组成 r 信号，同样符合预期。因此，该过程正确地完成了计算结果的输出。测试模块验证后，error 信号为 0，说明计算结果正确。

8) 图 5(e)显示，测试模块对该计算进行了两次测试，分别测试商和余数输出是否正确。商和余数均与 answer 信号一致，error 信号一直为 0，本次计算结果正确。

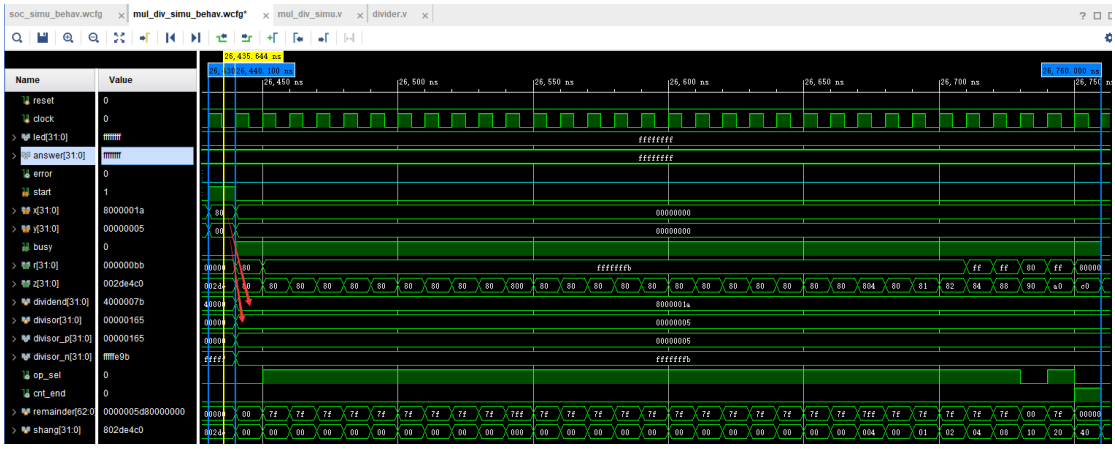

图 6(a)　32 位除法器测试用例 2

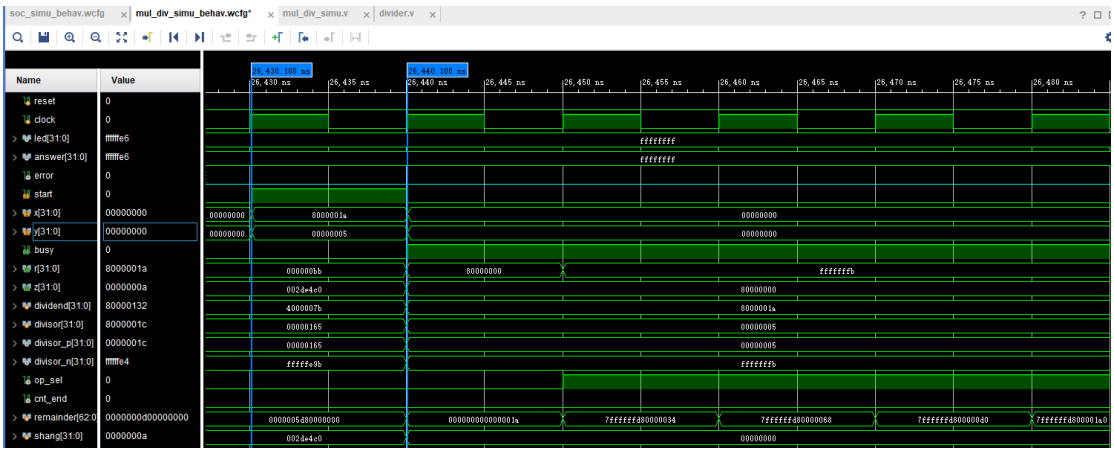

图 6(b) 开始时余数和商的变化

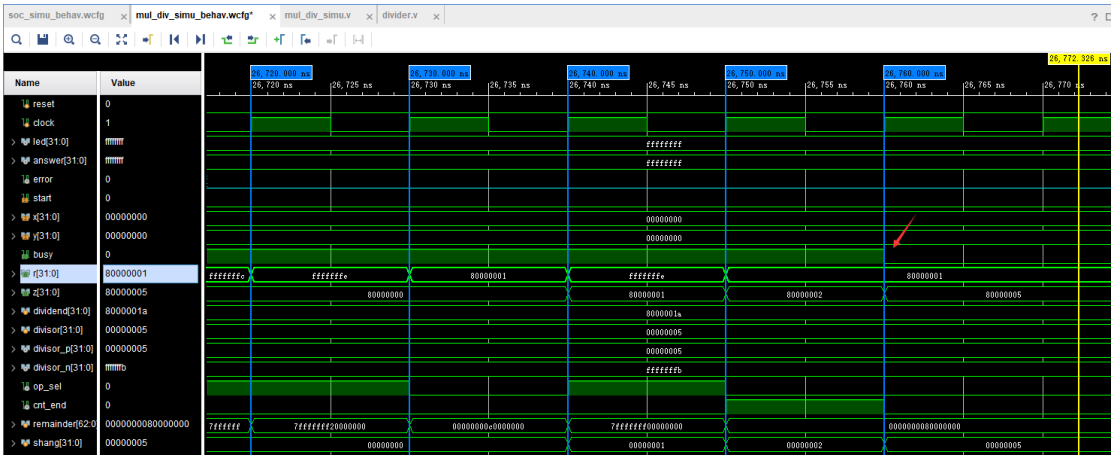

图 6(c) 加减交替法过程和结束时刻

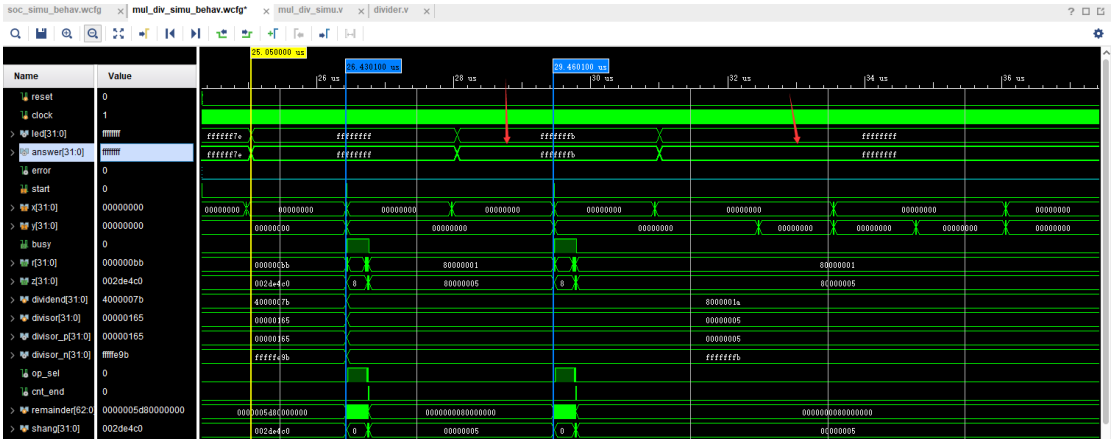

图 6(d) 计算结果正确

仿真波形中涉及到的信号与上一用例相同。

1) 图 6(a)中标记了开始的那一个周期，start 信号为 1，并在 x 和 y 中给到有效输入。在标记的第二个时钟上升沿后，busy 立即更新为 1，同时 x 和 y 的值分别保存到 dividend 寄存器和 divisor 寄存器中，divisor_p 和 divisor_n 同步更新成正确的信号。

2) 在计算开始后，通过图 6(b)可以看出，计算开始时被除数的低 31 位也存储到了 remainder 信号中。在计算开始的第一个时钟上升沿，op_sel=0，因此 remainder 移位并加上了 divisor_n 的值，符合预期。第一次移位时，不进行上商，shang 信号保

持位 0，同样符合预期。此后的几个周期，remainder 的移位和加法操作经验证，符合预期。每次上商的值都是 0，因此 shang 保持 0，也符合预期。

3) 在图 6(c)中展示了改用例临近结束时的一段运行过程。标记了 5 个时钟上升沿。在第 2 个时钟上升沿时，op_sel=1,上商 0，remainder 信号经更新后变成正数。在第 3 个时钟上升沿，op_sel=0，shang 信号进行左移，最低位上商 1，于是变成 1，符合预期。remainder 更新后为负数。第 4 个时钟上升沿时，op_sel=1,因此 shang 仅左移 1 位，变成 2，符合预期。可以说明，余数和商都按照加减交替法的预期变化。

4) 图 6(c)中标记的第 4、5 个时钟上升沿间，是最后一个周期。此时 cnt_end =1，按照预期，本次不再对余数进行移位，若余数为负，会有额外操作，余数为正，则保持不变。仍然进行最后一次移位和上商。在第 5 个时钟上升沿后，remainder 信号保持不变，由于 op_sel 信号为 0，shang 从 2 变成 5，符合预期。

5) 在结束时刻，busy 信号变为 0。此时，shang 信号和 divisor、dividend 的最高位共同生成输出的商 z，最高位变为 1，符合预期。输出的余数 r 由 remainder 信号和 dividend 信号的最高位共同生成，原本 remainder 中仅有的一位 1 恰好位于 r 的最低位，r 最高位由 dividend 最高位决定，符合预期。

6) 图 6(d)显示，测试模块进行了两次测试，分别检查了商和余数的值。测试模块使用补码，除法器输出原码。商和余数均与 answer 的值相等，error 信号为 0，本次计算结果正确。

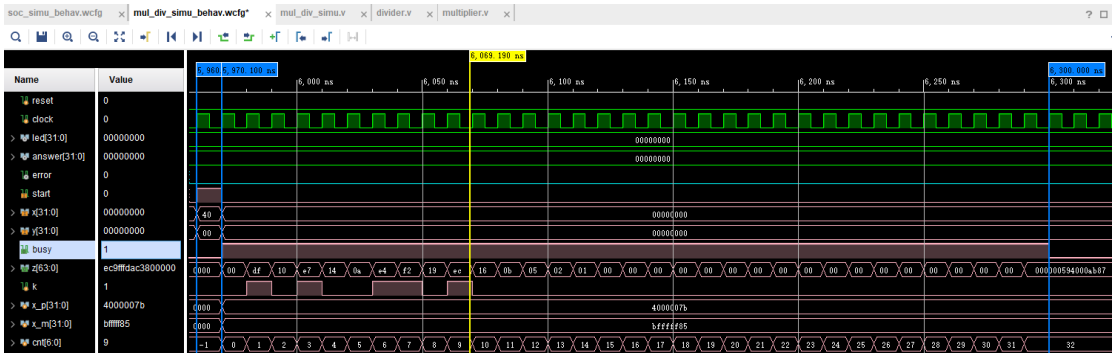

图 7(a)　32 位乘法器测试用例 1

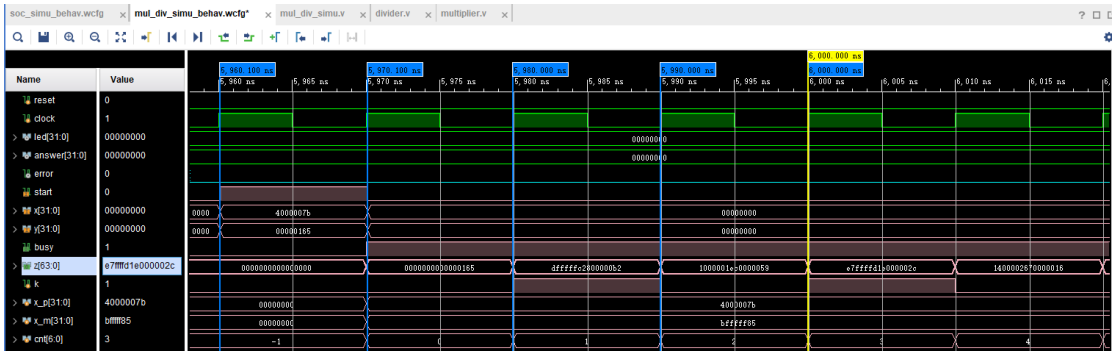

图 7(b)　开始时数的储存和运算

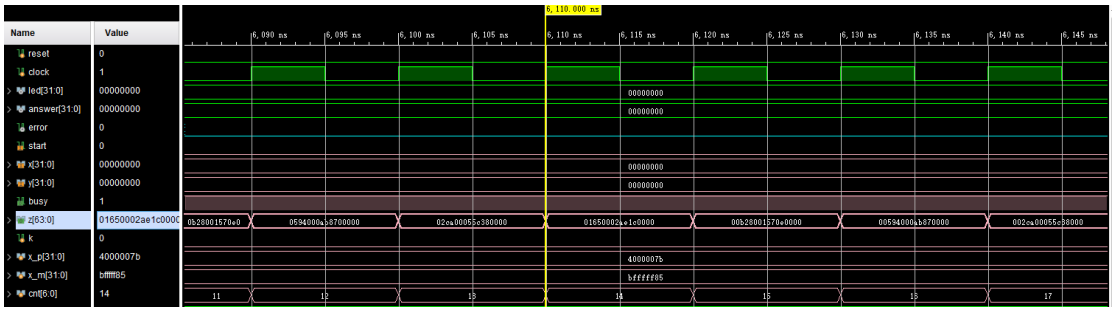

图 7(c) 计算过程中的移位

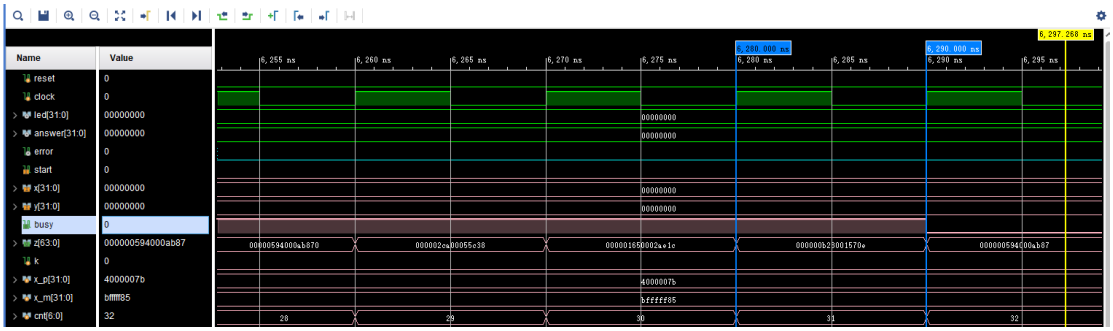

图 7(d) 计算结束的时刻

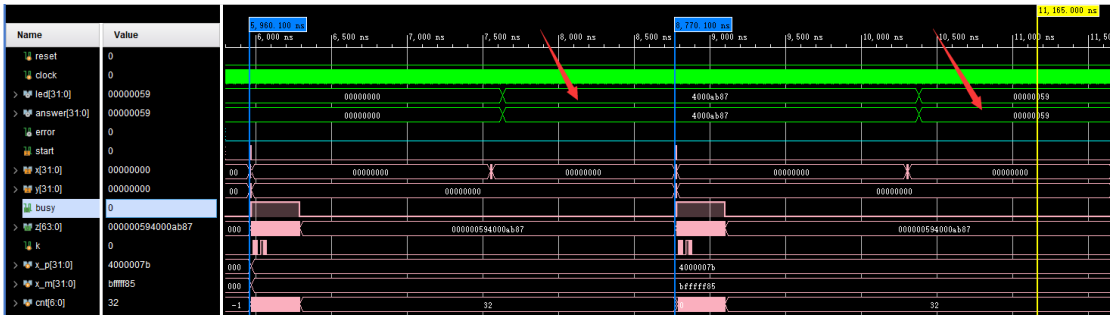

图 7(e) 计算结果正确

仿真波形中，所涉及到的信号有：输入信号 start，被乘数 x，乘数 y，输出 busy 信号，输出的乘积信号 z，同时 z 也是计算过程中的部分积寄存器。辅助位 k，计数器 cnt，以及被乘数正负补码的寄存器 x_p 和 x_m。

1) 在图 7(a)标记的第 1 个时钟上升沿处，start 置 1，x 和 y 给有效信号。持续一周期后，start、x、y 均置 0，同时 busy 信号置 1，x_p 和 x_m 储存被乘数的正负补码。在 busy 信号为 1 的过程中，部分积 z 和辅助位 k 每个周期都会进行更新。

2) 如图 7(b)所示，在标记的第 2 个时钟上升沿，即开始信号输入后，乘数 y 立即存储到部分积的低 32 位，同时 x 的正负补码存储到 x_p 和 x_m 中，计数器 cnt 设置为 0，辅助位 k 初始值 0。在图中第 3 个上升沿，booth 算法开始执行。根据预期，由于部分积连同辅助位的低 2 位为 10，应该减去被乘数，因此部分积加上 x_m 并右移。经验证，部分积的变化符合预期。同时，部分积的最低位右移后给到辅助位，因此 k 更新为 1，符合预期。

3) 在图 7(b)第 4 个时钟上升沿时，部分积连同辅助位的低 2 位为 01，因此应加上被乘数，并右移。经验证，部分积的变化符合预期。同时，部分积最低位右移后给到辅

助位，也符合预期。

4) 图 7(c)展示了计算过程中，cnt=14 左右时的一段计算过程。在连续几次操作中，辅助位位 0，部分积最低位也为 0，因此几次操作均为直接右移，部分积的变化符合预期。

5) 根据设计，在 cnt=31 时，会进行最后一次操作。cnt=32 时，计算结束，busy 会同步置 0。在图 7(d)标记的第 1 个时钟上升沿，cnt=30，部分积连同辅助位的末 2 位为 00，因此部分积直接右移，部分积的变化符合预期。cnt=31 时，部分积连同辅助位的末 2 位为 00，再次进行右移，符合预期。此时 z 存储的就已经是运算结果。

6) 图 7(e)显示了测试模块进行的测试，进行了两次计算，分别检查低 32 位和高 32 位的结果。z 最终的输出值与 answer 相同，error 信号位 0，计算结果正确。

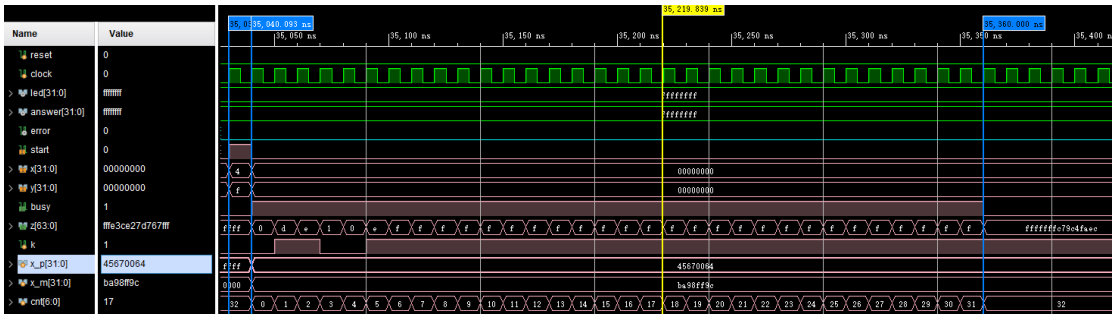

图 8(a)　32 位乘法器测试用例 2

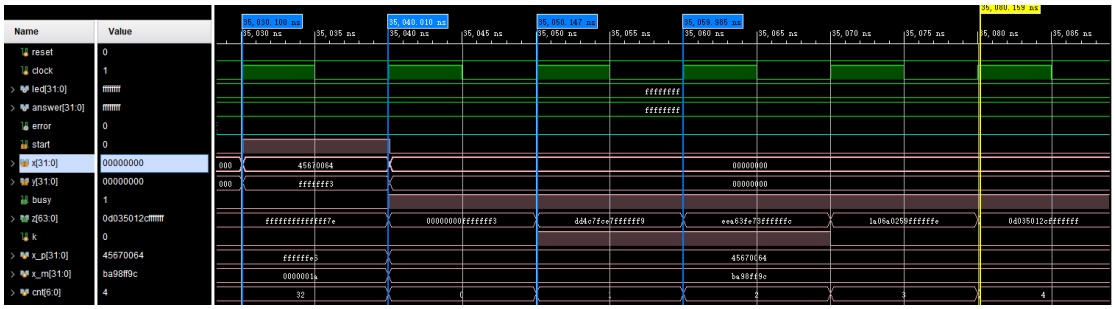

图 8(b)　开始时数的储存和运算

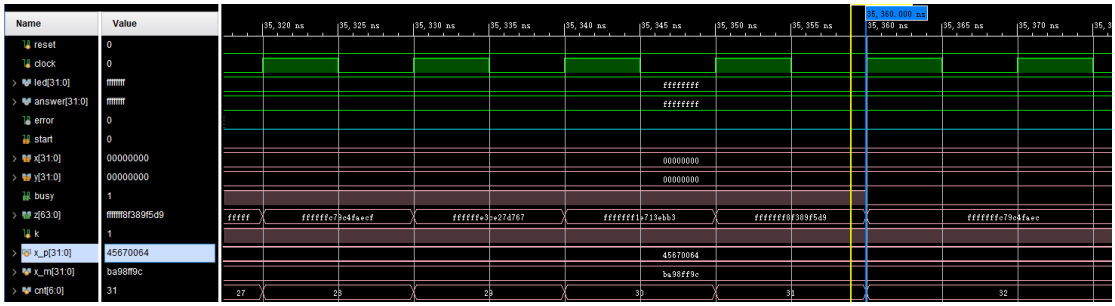

图 8(c) 计算结束的时刻

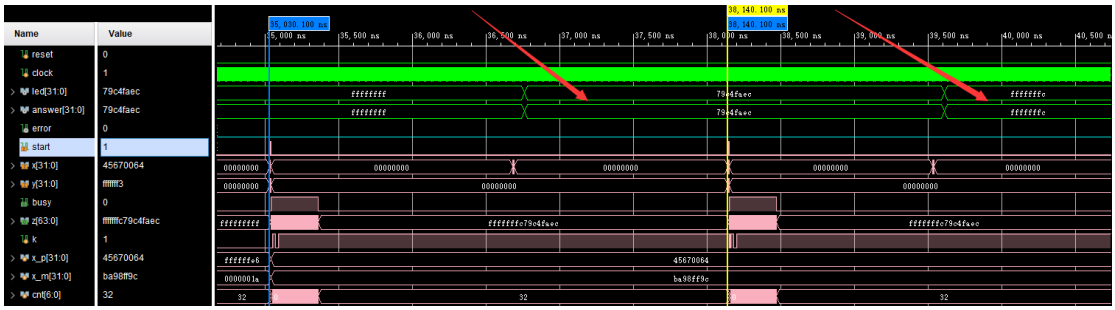

图 8(d) 计算结果正确

仿真波形中涉及到的信号与上一用例相同。

1) 第 1 个标记的时钟上升沿到来时，start 信号更新为 1，同时 x 和 y 给有效输入，持续一个周期。第 2 个时钟上升沿时，x 的值立刻以正负值储存到 x_p 和 x_m 寄存器中，同时 y 的值储存到部分积的低 32 位，计数器 cnt 归 0，辅助位初始值为 k。
2) 图 8(b)中第 3 个标记的时钟上升沿到来时，部分积连同辅助位的末 2 位为 10，此时部分积应在高位减去被乘数并算数右移。因此 z 的值应加上 x_m，并算数右移。经验证，z 的变化符合预期。在第 4 个标记的时钟上升沿，部分积连同辅助位的末 2 位为 11，此时部分积直接右移。经验证，z 的变化符合预期。
3) 如图 8(c)所示，在 cnt=31 时，进行最后一次操作，部分积连同辅助位的末 2 位为 11，部分积直接进行算术右移。经验证，部分积的变化符合预期。之后 cnt 更新至 32，busy 同步变为 0，此时 z 存储的就是最后的乘积。
4) 图 8(d) 显示了测试模块进行的测试，测试模块进行了两次相同输入的运输，分别检查乘积的低 32 位和高 32 位。busy 从 1 变成 0 后，z 的低 32 位与高 32 位分别于 2 个阶段的 answer 相同，error 为 0，说明计算结果正确。