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

实验作业

开课学期：2022 春季

课程名称：计算机组成原理（实验）

实验名称：Booth 乘法器设计

实验性质：综合设计型

实验学时：4 地点：

学生班级：计算机类 4 班

学生学号：200110428

学生姓名：杨杰睿

作业成绩：

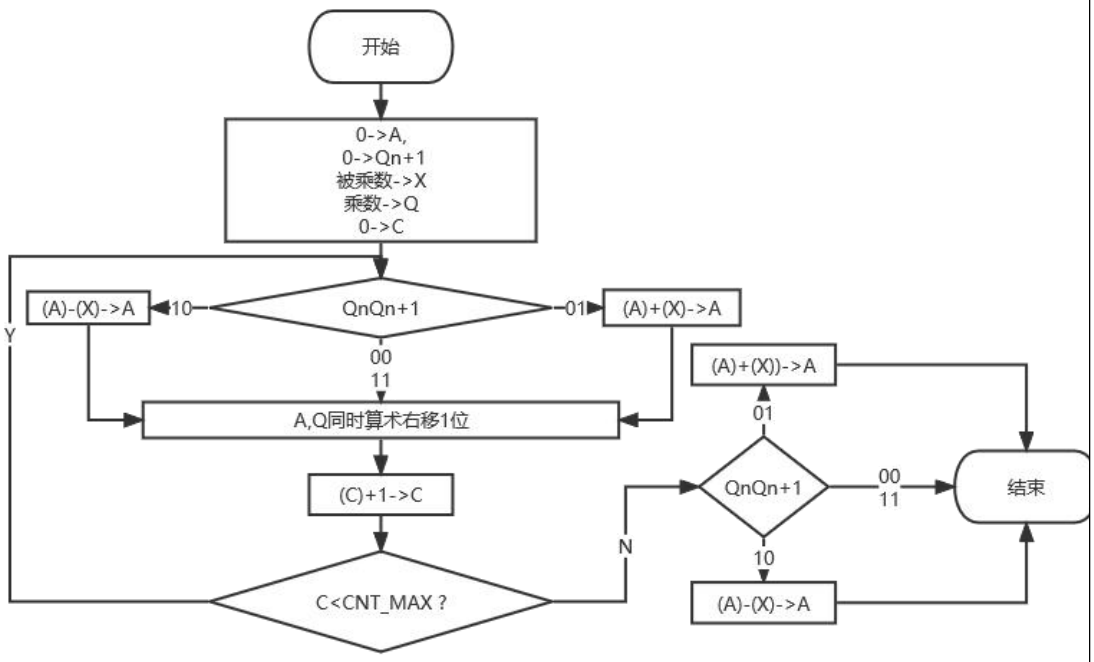
实验与创新实践教育中心制

2022 年 4 月

1、Booth 乘法器算法流程图

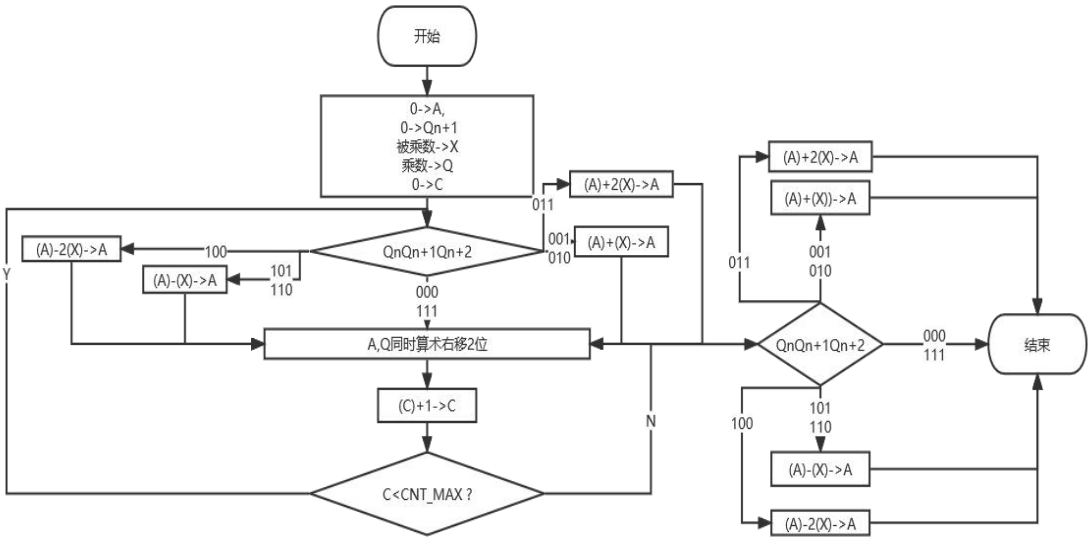
Booth 算法流程图：

booth.v



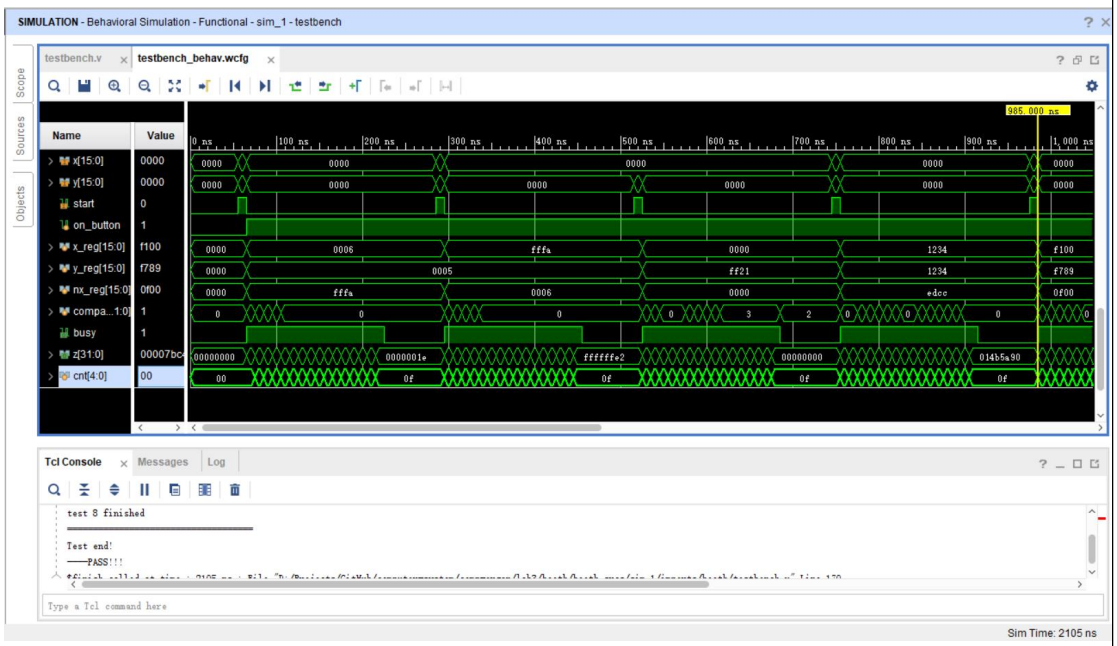
改进的 Booth 算法流程图：

booth2.v



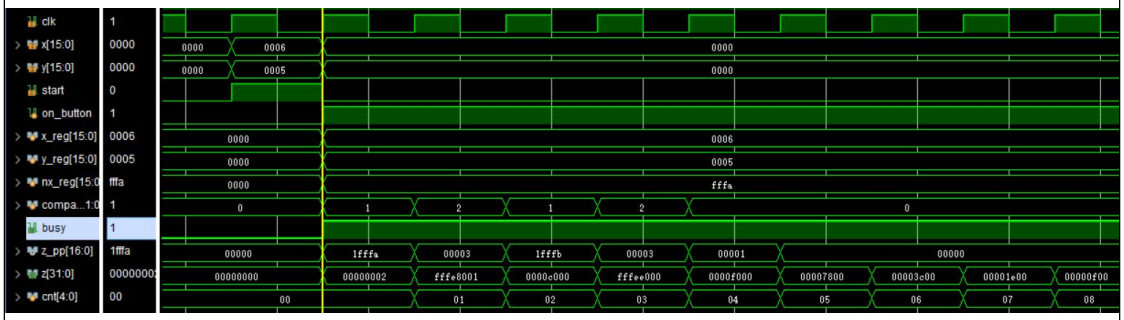
2、调试报告

booth.v 仿真通过，波形截图如下所示：



booth.v 乘法运算分析：

0ns-55ns：在计算开始前，各寄存器值均为 0，on_button 标记为 0



(1) 第 1 次乘法分析：

55ns-65ns：start 拉高一个周期，在这个周期各寄存器初始化，周期结束后完成赋值

65ns-75ns：on_button, busy 信号拉高，初始时截取 y 的前 15 位放入 z 的低 15 位的 z 的初始值 2，$[y_{n+1}\ y_n]=[0\ 1]$，加 $-x$ 的补码 $[f\ f\ f\ a]$。当前高 17 位部分和（双符号位）为 $z\_pp=[1\ f\ f\ f\ a]$，周期结束后，部分和加上原 z 值并完成右移 1 次，

进行符号扩展，得到$z=[f\ f\ f\ e\ 8\ 0\ 0\ 1]$

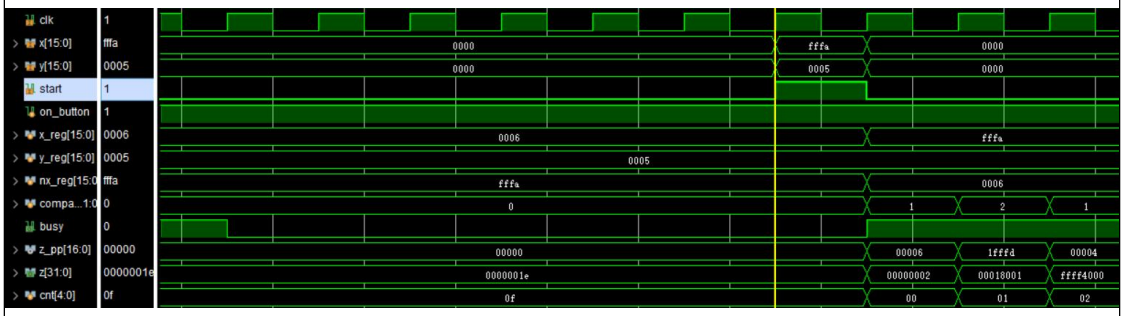
75ns-85ns：on_button, busy 信号高位，$[y_{n+1}\ y_n]=[1\ 0]$，加x的补码[0 0 0 6]。当前高 17 位部分和为$z_pp=[0\ 0\ 0\ 0\ 3]$，周期结束后部分和加上原 z 值并右移 1 位，符号扩展得到 z=[0 0 0 0 c 0 0 0]。

85ns-95ns：$[y_{n+1}\ y_n]=[0\ 1]$，加$-x$的补码$[f\ f\ f\ a]$。当前高 17 位部分和（双符号位）为$z_pp=[1\ f\ f\ f\ b]$，周期结束后，部分和加上原 z 值并完成右移 1 次，进行符号扩展，得到$z=[f\ f\ f\ e\ e\ 0\ 0\ 0]$。

95ns-105ns：[yn+1 yn]=[1 0]，加 x 的补码[0 0 0 6]。当前高 17 位部分和（双符号位）为 z_pp=[0 0 0 0 3]，周期结束后，部分和加上原 z 值并完成右移 1 次，进行符号扩展，得到 z=[0 0 0 0 f 0 0 0]。

105ns-225ns：[yn+1 yn]=[0 0]。当前 z 逐次右移，直到 15 次右移完成后，busy 信号拉低，最终得到 z=[0 0 0 0 0 0 1 e]。

225ns-285ns：z 值保持不变，计数器 cnt 保持不变，busy 信号保持低位。



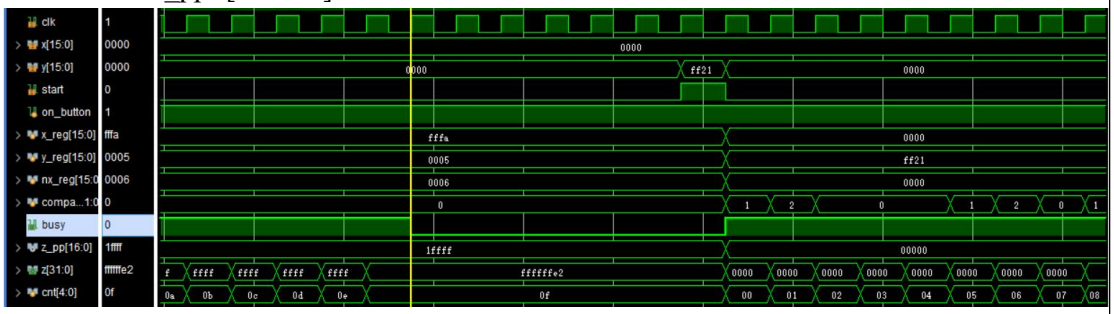
(2) 第 2 次乘法分析：

285ns-295ns：start 拉高一个周期，在这个周期各寄存器初始化，周期结束后完成赋值，得到初始化 z=[0 0 0 0 0 0 0 2]。

295ns-305ns：[yn+1 yn]=[0 1]，加-x 的补码[0 0 0 6]。当前高 17 位部分和（双符号位）为 z_pp=[0 0 0 0 6]，周期结束后，部分和加上原 z 值并完成右移 1 次，进行符

号扩展，得到 z=[0 0 0 1 8 0 0 1]。

305ns-315ns：[yn+1 yn]=[1 0]，加 x 的补码[f f f a]。当前高 17 位部分和（双符号位）为 z_pp=[1 f f f d]，周期结束后，部分和加上原 z 值并完成右移 1 次，进行符



号扩展，得到 z=[f f f f 4 0 0 0]。

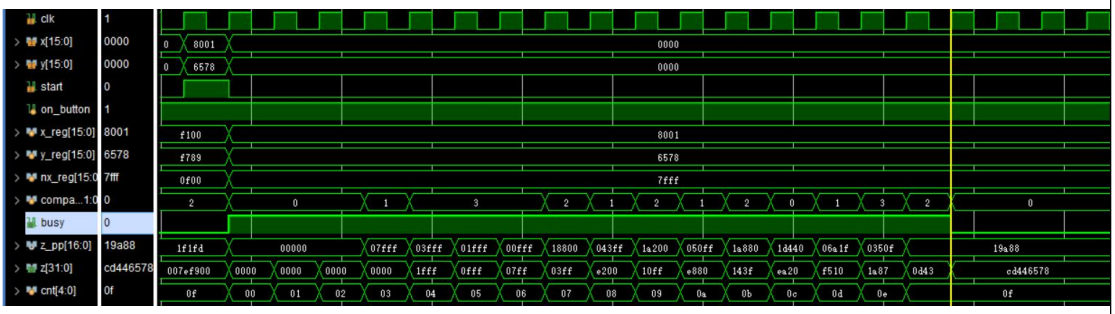
315ns-325ns：[yn+1 yn]=[0 1]，加-x 的补码[0 0 0 6]。当前高 17 位部分和（双符号位）为 z_pp=[0 0 0 0 4]，周期结束后，部分和加上原 z 值并完成右移 1 次，进行符号扩展，得到 z=[0 0 0 1 2 0 0 0]

325ns-335ns：[yn+1 yn]=[1 0]，加 x 的补码[f f f a]。当前高 17 位部分和（双符号位）为 z_pp=[1 f f f c]，周期结束后，部分和加上原 z 值并完成右移 1 次，进行符号扩展，得到 z=[f f f f 1 0 0 0]。

335ns-455ns：[yn+1 yn]=[0 0]。当前 z 逐次右移，直到 15 次右移完成后，busy 信号拉低，最终得到 z=[f f f f f f e 2]。

455ns-515ns：z 值保持不变，计数器 cnt 保持不变，busy 信号保持低位。

(3) 第 5 次乘法分析（因此次执行了完整的 16 个周期计算，且含有 debug 过程记录附



后）

具体的分析过程同上两次乘法运算，下面简要总结过程中出现的计算情况：

- 从 1215ns-1375ns 共计 16 个时钟周期，busy 处于高位，总共完成完整的 16 个

周期的计算过程。

- 输入的 x，y 分别为[8 0 0 1]和[6 5 7 8]，经计算得的-x 为[7 f f f]（均为补码表示）
- 向量 comparator 即[yn+1 yn]在各个周期（16 个计算周期+剩余周期）的取值为：

  [0 0] => [0 0] => [0 0] => [0 1] => [1 1] => [1 1] => [1 1] => [1 0] =>

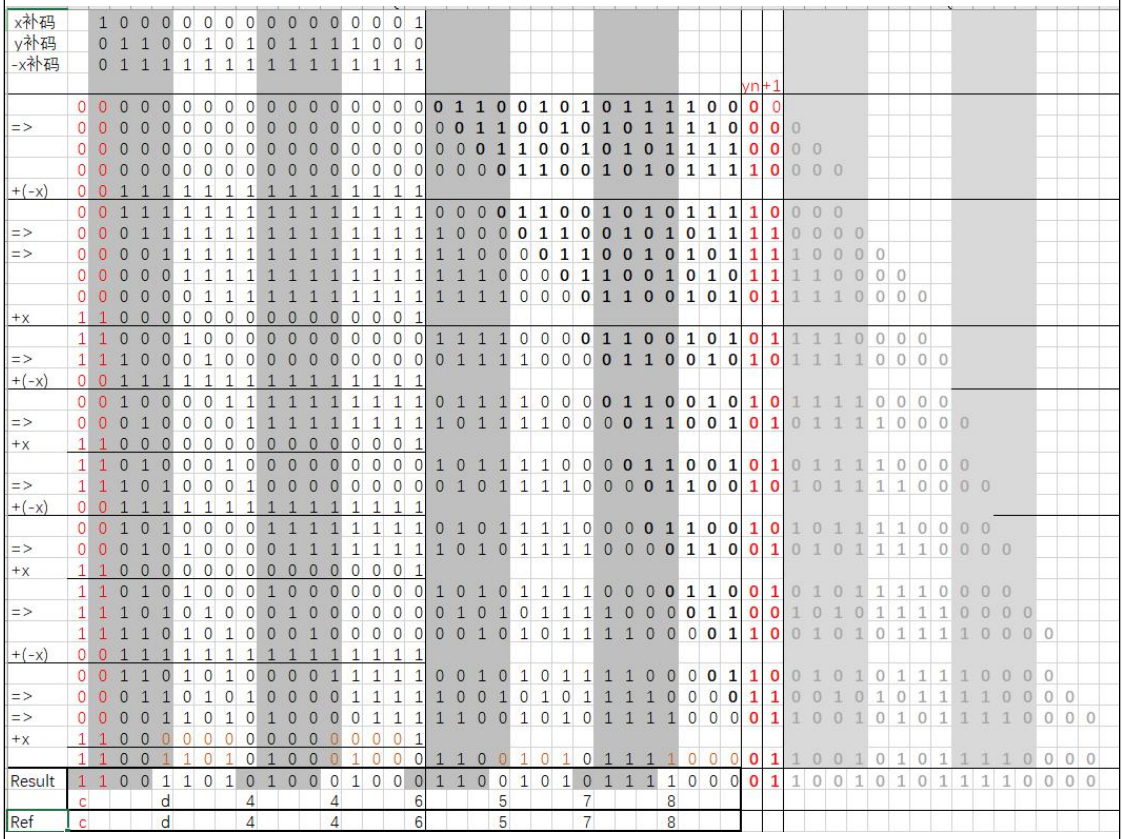
  [0 1] => [1 0] => [0 1] => [1 0] => [0 0] => [0 1] => [1 1] => [1 0] =>

  [0 0] ……（此后全[0 0]）

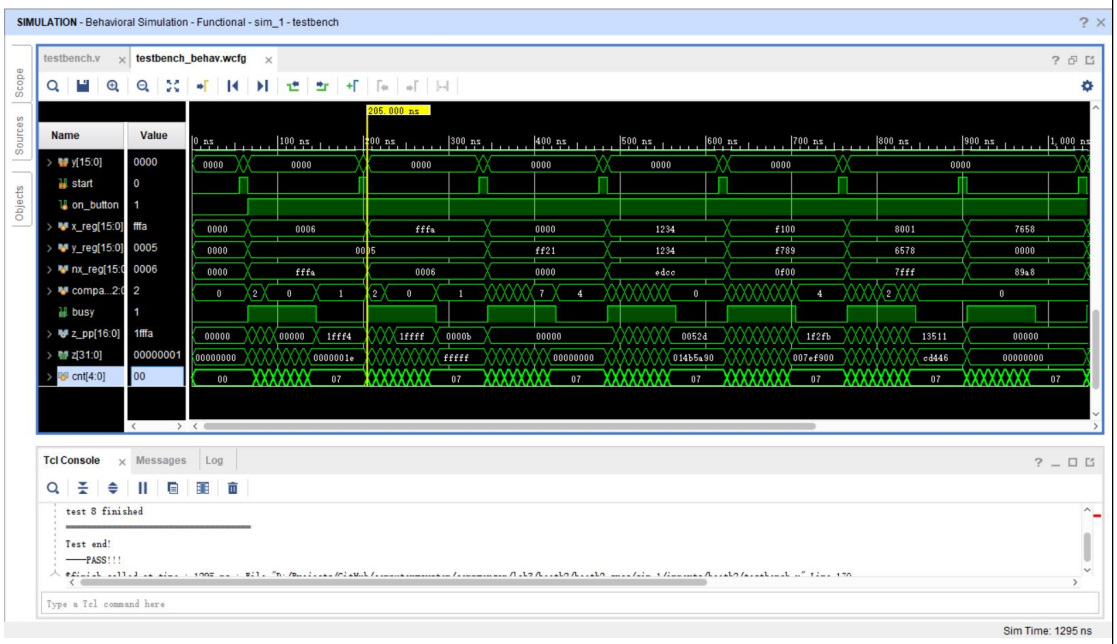
- 对于取值为[0 0]和[1 1]的周期，z 右移 1 位，并进行符号扩展
- 对于取值为[0 1]的周期，部分和 z_pp 加上-x，与 z 的高 17 位相加，并完成 z 的右移和符号扩展
- 对于取值为[1 0]的周期，部分和 z_pp 加上 x，与 z 的高 17 位相加， 并完成 z 的右移和符号扩展

波形图如上所示，需要特别指出的是，在第 16 个周期，即 cnt=15 时（因为 0 时占据了一个周期），z 在完成计算后，不再进行右移，因为 z 的低位从起初是截取的 y 的高 15 位存放的，在计算过程中，当 y 的高 15 位被完全移出 z 后，计算完成，总共需要右移 15 次，故第 16 个周期时，在不应当继续右移。最终可以计算出 z 的正确答案为 z=[c d 4 4 6 5 7 8]。

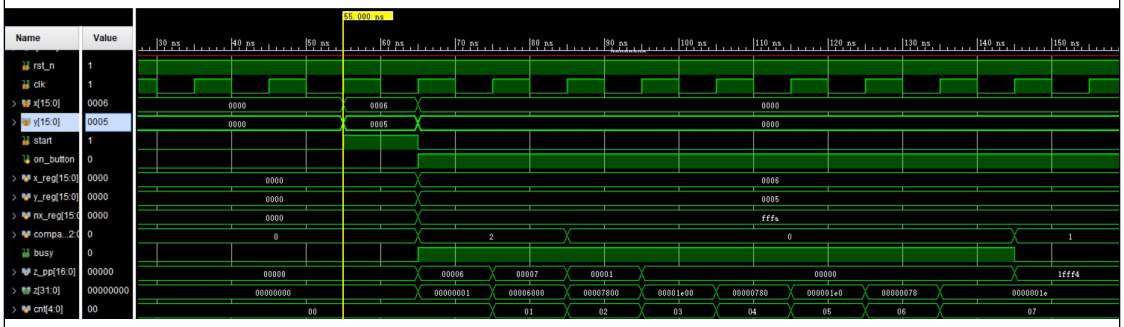
下附 debug 过程的手算记录，代码此前因最后 1 个周期没有计算部分和 z_pp 导致了错误，检查计算过程后更正了代码：



booth2.v 仿真通过，波形截图如下：



booth2.v 乘法运算分析：



(1) 第 1 次乘法分析

0-55ns：在 rst_n 高位后，第一次 start 启动前，寄存器的值均置 0

55ns-65ns：start 高位，这个周期中 start 的高有效使得 x_reg 和 y_reg 在该周期结束后被赋值，z 作为线网类型其低位在这一周期结束后通过 z_reg 赋值与 y 的高 14 位相同。

65ns-75ns；busy 高位，乘法器开始正式计算，z 低位的值在这一周期被赋值为 y 的高 14 位即（二进制）[0000 0000 0000 0101]的前 14 位，为（十六进制）[00000001]。[yn+2 yn+1 yn]=[0 1 0]，z_pp 保存 z[30:14]与 x 的双符号位补码之和，在这一周期结束后，z 的值高位由部分和 z_pp 右移 2 位后填充，低位为原 z[13:2]，在这周期

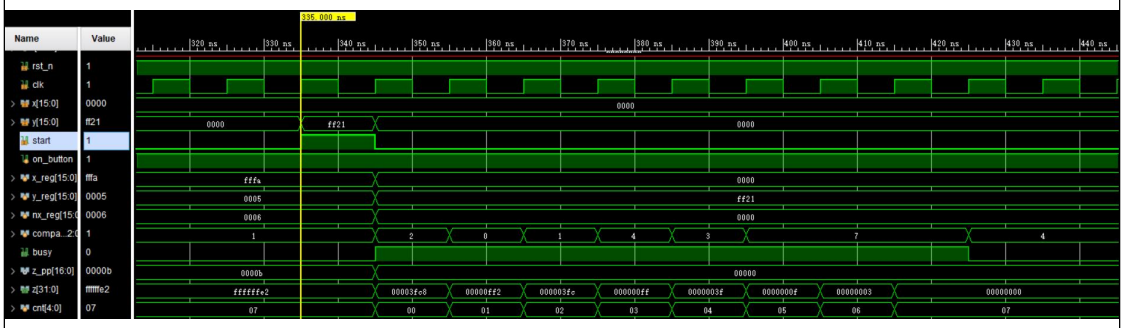
结束后，z=[00006000]。

75ns-85ns：busy 高位，[yn+2 yn+1 yn]=[0 1 0]，z_pp 保存 z[30:14]与 x 的双符号位补码之和，在这一周期结束后，z 的值高位由部分和 z_pp 右移 2 位后填充，低位为原 z[13:2]，在这周期结束后，z=[0 0 0 0 7 8 0 0]

85ns-145ns：busy 高位，[yn+2 yn+1 yn]=[0 0 0]，z_pp 保存 z[30:14]与 x 的双符号位补码之和，在每一周期结束后，z 的值高位均由部分和 z_pp 右移 2 位后填充，低位为原 z[13:2]，在这 6 个周期结束后，z=[0 0 0 0 0 0 1 e]

145ns-195ns：busy 低位，计算已经在上一周期结束，z 保持不变，[yn+2 yn+1 yn]收到算法的影响会进行一次计算，之后保持不变。计数器保持不变。

(2) 第 3 次乘法分析



335-345ns：本次 start 启动前，寄存器的值保持上一周期结果不变，在 start 拉高位之后，各寄存器的值将在本周期结束后更新。

345ns-425ns：start 高位，这 8 个周期中[yn+2 yn+1 yn]变化如下所示：

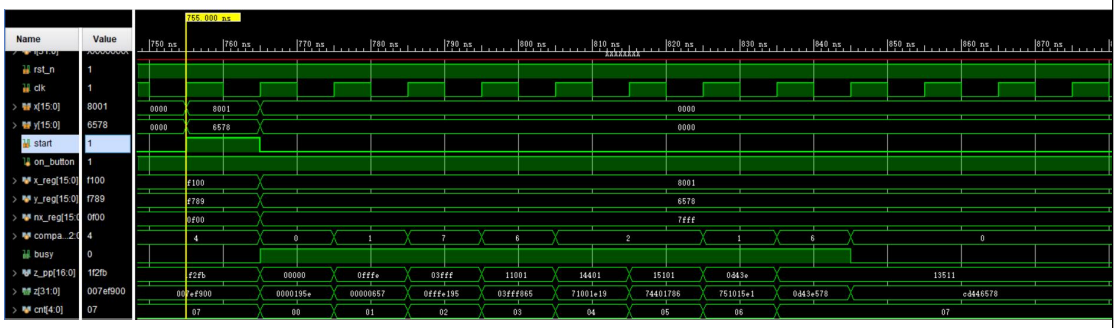
[0 1 0] => [0 0 0] => [0 0 1] => [1 0 0] => [0 1 1] => [1 1 1] => [1 1 1]

但受到 x 取值为 0 的影响，部分积 z_pp 始终为 0，在进行计算（如[0 1 0]为 x 的 1 倍加法，[0 0 1]为-x 的 2 倍加法）的过程中，实际上相当于始终每个周期将 z 的值友谊 2 位，最终得到 z=[0 0 0 0 0 0 0 0]

425ns-475ns：busy 低位，计算已经在上一周期结束，z 保持不变，[yn+2 yn+1 yn]收到算法的影响会进行一次计算，之后保持不变。计数器保持 7 不变。

(3) 第 5 次乘法分析

755ns-765ns：本次 start 启动前，寄存器的值保持上一周期结果不变，在 start 高位之后，各寄存器的值将在本周期结束后更新。



765ns-775ns：[yn+2 yn+1 yn]=[0 0 0]，z_pp= z[30:14],z= {z_pp[16], z_pp[16], z_pp, z[13:2]},最终 z=[0000195e]

775ns-785ns：[yn+2 yn+1 yn]=[0 0 1]，z_pp= {nx_reg[15], nx_reg} + {nx_reg[15], nx_reg} + z[30:14], z= {z_pp[16], z_pp[16], z_pp, z[13:2]}，最终 z=[00000657]

785ns-795ns：[yn+2 yn+1 yn]=[1 1 1], z_pp = z[30:14], z= {z_pp[16], z_pp[16], z_pp, z[13:2]}，最终 z=[0fffe195]

795ns-805ns: [yn+2 yn+1 yn]=[1 1 0], z_pp = {x_reg[15], x_reg} + {x_reg[15], x_reg} + z[30:14]，最终 z=[03fff865]

805ns-815ns 及 815ns-825ns: [yn+2 yn+1 yn]=[0 1 0], z_pp= {x_reg[15], x_reg} + z[30:14]，z= {z_pp[16], z_pp[16], z_pp, z[13:2]}，最终 z=[74401786]

825ns-835ns: [yn+2 yn+1 yn]=[0 0 1], z_pp={nx_reg[15], nx_reg} + {nx_reg[15], nx_reg} + z[30:14]，最终 z=[751015e1]

835ns-845ns: [yn+2 yn+1 yn]=[1 1 0], z_pp={x_reg[15], x_reg} + {x_reg[15], x_reg} + z[30:14]，最终 z=[0d43e578]

845ns-855ns:在上一周期结束后计算出 z=[cd446578]，在本周期开始前完成赋值

剩余周期各寄存器保持不变，直到下一次 start 高位