

（深圳）

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： T2612

学生班级： 11

学生学号： 200111132

学生姓名： 吴桐

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** | |
|  | |
| 1. **调试报告** | |
| （仿真截图及时序分析，要求分析最少3次乘法运算）  1.仿真通过截图  2.波形分析  ①工作时间  工作周期为17个周期<20个周期，符合题目要求  第一个周期载入数据，后面十六个周期分别部分积加减16次（n+1），随后Z输出正确结果、  第一题中busy结束后为 5 \* 6 = 0x1e 结果正确；  ②每一步乘法波形分析  C:\Users\Wufisher\AppData\Roaming\Tencent\Users\1105889424\QQ\WinTemp\RichOle\3JTAO~[`@_DD$J8V0K%4%]D.png  由于32位Z如果波形展开的话实在太长了，后续就直接分析Z[31:0]的十六进制数来代表波形  这这里展示Z能够实现算数位移功能  1. x=6,y=5  第一个周期载入数据  直接将y加载到Z的低位上，并且完成x的正负补码储存  1.y第一位为1，部分积加上-x的补码 得到0xfffa0005  2.y[1:0]为01，向右算数位移后，部分积加上-X的补码 得到0x00030002  3.y[1:0]为10，向右算数位移后，部分积加上X的补码 得到0xfffb8001  4.y[1:0]为01，向右算数位移后，部分积加上-X的补码 得到0x0003c000  后续由于y为5较小，全是00，所以只有算数位移  得到结果：0x0000001e  2.x=0xfffa , y=0x0005  第一个周期载入数据后  1.y第一位为1，部分积加上-x的补码 得到z=0x00060005  2.y[1:0]为01，向右算数位移后，部分积加上-X的补码 z=0xfffd0002  3.y[1:0]为10，向右算数位移后，部分积加上X的补码 z=0x00048001  4.y[1:0]为01，向右算数位移后，部分积加上-X的补码z=0xfffc4000  后续由于y为5较小，全是00，所以每一步只有算数位移  得出结果 0xffffffe2  3.x=0 ，y=0xff21  第一步载入y数据到z低位  由于x=0，所以每一次无论如何判断，都是向右算数位移  低16位向右位移波形图  最终结果为 0  后续分析都符合算法 | |
| 1. **Booth乘法器算法流程图（改进版）** |
| 我觉得这样的流程图更清晰好看一些  指导书这个算法如果从高位定格则最后少了一个算数右移两位，或者一开始就已经默认已经算数右移两位做起（否则计算结果与之前一个公式矛盾）  我将+2^-2（-2yn-1 + yn + yn+1）[x]补直接在载入数据的时候就做了算数位移处理，防止代码有误解，所以在此说明 |
| 1. **调试报告** |
| 1.仿真通过  2.时间要求  以第一个用例为例子，花费8个工作时间周期出结果，符合要求  3.波形分析  用例1  我是直接在Z上操作的，X储存也是储存在可以直接加减的高位[31:14]里，  做了充分的符号扩展，详可看代码  Y为0101 ，符号补右0为 0101（0）  接下来的判断后三位为  010 010 之后全为0  从周期可以看出，  第一个周期+X  第二个周期位移两次，然后+x  后续周期全部为算数位移  最后得到正确结果  用例2  X=0xfffa  Y=0x0005  Y为0101 ，符号补右0为 0101（0）  接下来的判断后三位为  010 010 之后全为0  从周期可以看出，  第一个周期+X  第二个周期位移两次，然后+x  后续周期全部为算数位移  从图中可以看出计算两次后符号位显然是1，所以后续是带1算数符号右移  从波形图中看出符合预期  高位：  低位：  最终结果正确  用例3  .x=0  Y=0xff21  这里由于自身算法原因（补符号位），这里又是乘法运算，所以我直接在读入数据的时候，加入了判0的分支  如果X,Y其中有一个为0，则虽然busy8个周期，  但是输出结果一直为0  波形符合预期  最下面一行为Z输出  后续用例同理  C:\Users\Wufisher\AppData\Roaming\Tencent\Users\1105889424\QQ\WinTemp\RichOle\US]_3OL)%Z4)1DH`UDJBZLJ.png  每个展开后得到的算法波形以及最后结果和预想一致 |