**实验报告——ALU运算单元**

1. **实验设计**
2. 实验的**输入**、**输出**及对应的**管脚**：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **变量名** | **类型** | **描述** | **宽度** | **管脚** |
| **op** | input | 操作符 | **2** | **T8，U8** |
| **a** | input | 输入信号A | **4** | **J15，L16,M13,R15** |
| **b** | input | 输入信号B | **4** | **R17，T18，U18，R13** |
| **out** | output | 输出的F | **4** | **H17，K15，J13，N14** |
| **Cout** | output | 进位输出 | **1** | **R18** |

**备注：**

1. **value(a)=** Σ**a[i]\*2i(i=0,1,2,3)**
2. **以上表中的管脚顺序对应于相应变量的二进制表示的顺序为：从权重较小的位到权重较大的位（如： op[0]对应的是T8，op[1]对应的是U8）**
3. **半加器的设计**

**(1)输入：a(4-bit)， b(4-bit)**

**(2)返回值：out(4-bit)， Cout(1-bit)**

**(3)分析：**

1. a与b相加，若得到的和不足八位，高位上补零；
2. 得到的和的低四位为out；
3. 得到的和的最高位为Cout，原因是：若a与b相加有进位，那么和的最高位上为1，否则为0，恰好与Cout的取值一致，因此可以直接将和的最高位赋给Cout

­­

1. **半减器的设计**

**(1)输入：a(4-bit)，b(4-bit)**

**(2)返回值：out(4-bit)，Cout(1-bit)**

**(3)分析：**

1. a与b相减，若得到的差不足八位，高位上补零；
2. 得到的差的低四位为out；
3. 得到的差的最高位为Cout，原因是：若a与b相减有借位，那么和的最高位上为1，否则为0，恰好与Cout的取值一致，因此可以直接将差的最高位赋给Cout

1. **取反操作的设计**

**(1)输入：a(4-bit)**

**(2)返回值：out(4-bit)**

**(3)分析：**直接对a取反赋给out

1. **乘法器的设计**

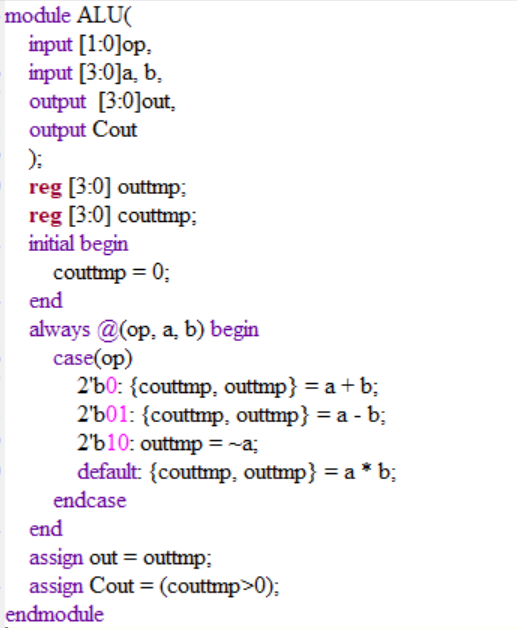
**(1)输入：a(4-bit)，b(4-bit)**

**(2)返回值：out(4-bit)，Cout(1-bit)**

**(3)分析：**

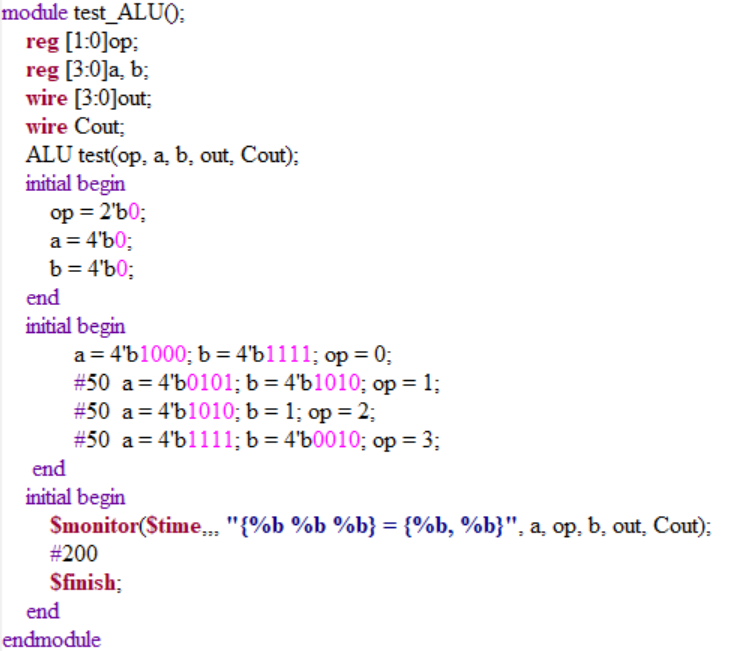
a与b相乘，得到的乘积最多有8位，低四位就是out，如果高四位上的值大于0，说明有进位，Cout为1，否则为0

1. **代码实现(如下图)，其中couttmp中是做加、减、乘法运算时高四位，outtmp是低四位。如果运算结束后，couttmp的值大于0，表示有进位，Cout=1，否则为0**

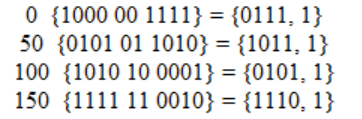
****

**二、仿真波形**

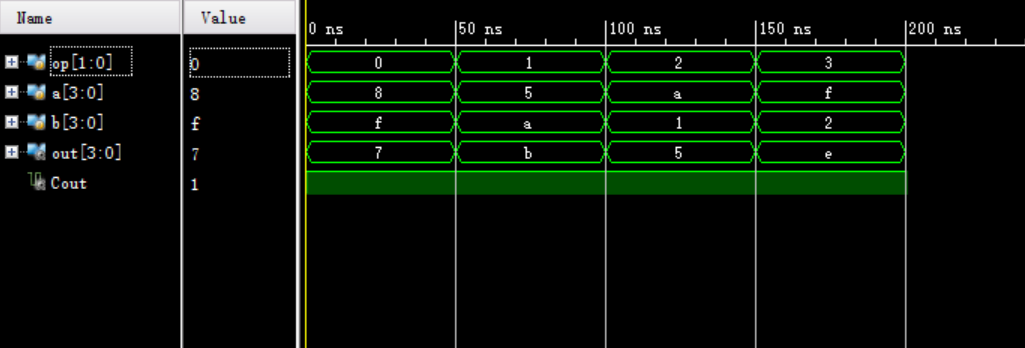
1. **测试代码：**

****

**2. 我针对四种运算，选取了四组数据进行分别测试，测试数据在控制台的输出如下：(格式为$time {a op b}={out, Cout})**

****

**3. 对应的仿真波形：**



**三、实验中遇到的问题和解决思路**

在此次实验中我遇到了好多问题，大部分都是因为对vivado设计流程不熟悉导致的，下面列出来的是我认为比较有意义的两个问题，也算是经验和教训吧：

1. **[Synth 8-2576] procedural assignment to a non-register led is not permitted**

这个是verilog的语法错误，经过查找一些教程，我对这个问题的初步的理解如下：

(1)概念：**连续赋值、过程赋值**

**连续赋值：**在连续赋值语句中，表达式右侧的计算结果可以立即更新表达式的左侧

**过程赋值：**表达式右侧的计算结果在某种条件的触发（如时钟的上升沿）下放到一个变量中

(2)概念：**wire类型、reg类型**

**wire：**表示直通，只要输入有变化，输出马上无条件地反映

**reg：**表示一定要有触发，输出才会反应输入

(3)**连续赋值和过程赋值的区别**：

|  |  |
| --- | --- |
| **连续赋值** | **过程赋值** |
| 主要赋值给nets | 主要赋值给reg变量 |
| 有关键词assign标识 | 没有assign标识 |
| **不可以用于always和initial等过程块中（上面ERROR的原因）** | 可以用在always和initial等过程块中 |
| 只于右端的计算结果有关 | 赋值是有“过程”的，即与上下文其他语句有关 |

(4)**wire和reg的区别：**

|  |  |
| --- | --- |
| **Wire** | **reg** |
| 不保存状态，它的值可以随时改变，不受时钟信号限制 | 可以保存输出状态。状态改变通常在下一个时钟信号边沿翻转时进行 |
| 可以用assign赋值 | 不可以用assign这种过程赋值的方式赋值 |
| input一定是wire类型 | output两种类型皆可 |

1. **在烧制FPGA时，已经用数据线连接了电路板，但是始终无法打开，查了好多资料，最后才发现是电源没有打开，白白耗费了很多时间**

**四、实验中的收获**

1.了解了一些verilog的基础语法

2.了解了vivado的设计流程和FPGA的烧制过程

3.对于时序电路和组合电路，从硬件方面有了更加底层的认识

**五、实验的感想**

这次实验中接触到的所有内容都是之前没有了解过的，所以中间过程有点艰辛，bug百出，但同时收获很多，做完之后也很有成就感，在此过程中，很感谢助教的耐心答疑。