2020-9-20

秦嘉余

191220088

1348288404@qq.com

数字电路实验二 加法器与ALU

目录

[1.实验目的 1](#_Toc52964640)

[2.实验原理 1](#_Toc52964641)

[3.实验环境/器材 1](#_Toc52964642)

[4.程序代码 2](#_Toc52964643)

[5.实验步骤 3](#_Toc52964644)

[6.测试方法 4](#_Toc52964645)

[7.实验结果 5](#_Toc52964646)

[8.实验中遇到的问题及解决办法 5](#_Toc52964647)

[9.对实验的思考 5](#_Toc52964648)

[10.思考题 5](#_Toc52964649)

1.实验目的

本次实验的目的是实现加法器和ALU这两种数字系统中的核心部件，学习简单ALU的设计方式，在quartus环境下利用verilog语言编写相关模拟程序，模拟仿真测试后写入DE-10开发板上。

2.实验原理

多位加法器可以利用移位加法器级联而成，而我们在verilog语言中无需关注这种基本的真值表达情况，只需要定义多个向量执行赋值语句即可，语言本身已经包含了处理的内容。

第一个器件加减法器中，加法直接相加即可，而减法需要将减数取反加一变成加法运算。同时根据一定的规则设定标志位即可。

第二个器件ALU的功能更加多样，包含了加法，减法，取反，位与，位或，位异或，大于判断，等于判断。除了加减法外，取反，位与，位或，位异或，等于判断操作均可直接赋值完成，语言本身已经可以处理。而大于判断由于我们输入的值代表为补码值，而我们定义的向量为无符号数，所以需要单独处理，不能直接比较大小。其中一种比较方法为把大于判断转化为减法判断是否大于0来判断。

经过上述的实验原理，我们已经可以设置加法器和ALU部件了

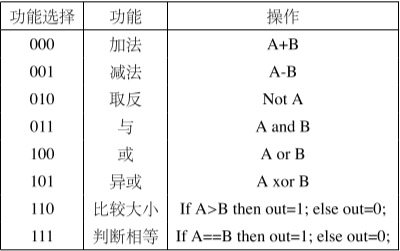
3.实验环境/器材

本次实验的环境为Quartus17.1版本

本次实验的器材为DE10 Standard开发板

4.程序代码

**本次实验需要实现加减法器以及ALU部件，由于ALU部件中已经包含加减法器的内容，所以以下内容仅展示ALU的部分。**

输入有四位输入A,B，选择功能端Op为3位，共有八种功能，如下图：

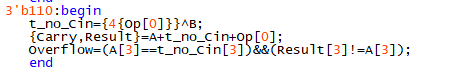
四位输出Result，一位标志位Carry，Overflow

代码在ALU.v文件中，由于篇幅问题，不在此处全文展示

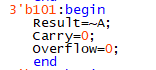
主要结构为case选择，对于每种情况执行不同操作，其中需要注意Op对于的Buttom为低电平有效，所以Op为反值



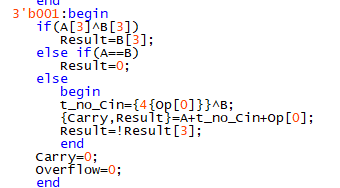
其中加法直接执行加法，利用连接符判断Carry是否进位，利用Overflow的符号位判断是否溢出。



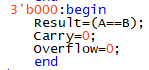
减法则将减数取反加一变为加法，其余相同



中间的取反，位与，位或，位异或操作均直接操作即可



大于判断，如果符号不同可以直接出结果，如果相等也可以直接出结果，如果符号相同则需要转化为减法，然后判断结果与0的大小

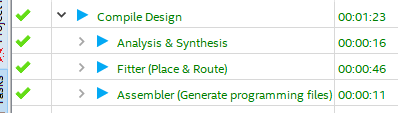


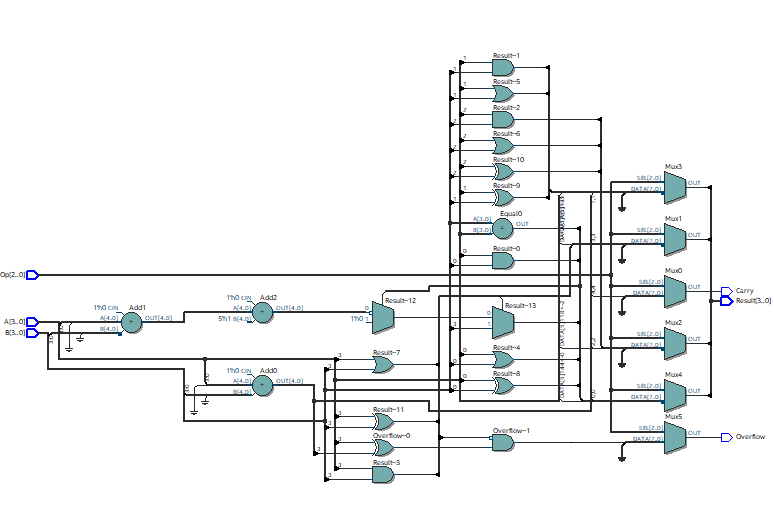
等于判断也直接操作即可

5.实验步骤

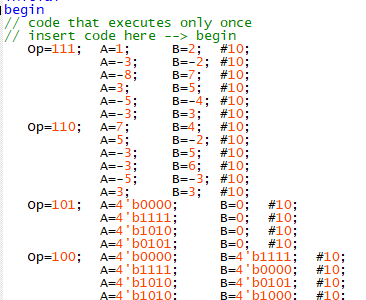
该部分主要展现是实验中各步骤

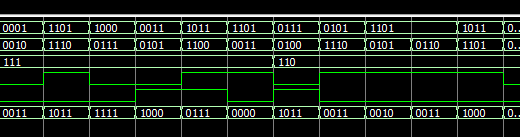
第一步：根据器件建立合适的项目文件，新建实验代码文件，键入相关代码，并且分析与综合，查看电路原理图，展示如下：





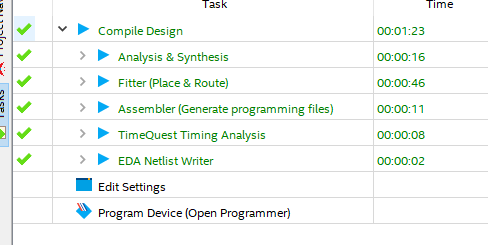
第二步：编写模拟测试代码，在Test Benches中链接相关模拟测试文件，最后利用RTL Simualtion查看波形图，测试程序是否正确，展示如下：

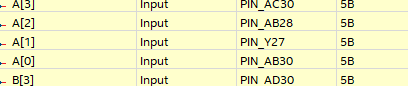
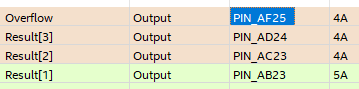




测试仿真代码仅展示部分，全文见ALU.vt文件

第三步：第二步正确后，将引脚对应到开发板的相应接口上，再编译全部项目，连接DE10 Standard开发板到电脑，将程序写入开发板上，展示如下：





第四步：写入开发板后，测试开发板输入输出是否正确，展示如下：

自此，该实验完成

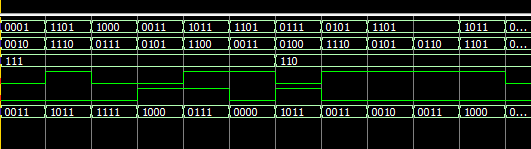
6.测试方法

本次实验采用测仿真仿真的方法， 编写的测试仿真代码如ALU.vt，在此由于篇幅限制不再展示。

其中分Op从111到000测试8种功能，每种功能选择有代表性的组合输入，测试了大部分有意义的情况

7.实验结果

仿真结果如下：



下载运行结果如下：

实验验收完成，与仿真结果一致

8.实验中遇到的问题及解决办法

在实验开始时，中对减法的标志位判断逻辑不太理解，在对4位二进制数的运算进行分析后了解了标志位判断的原理。

在比较大小的功能实现时，一开始打算直接判断，后来想到使用之前实现过的减法判断更为简便

9.对实验的思考

此时，PA也进行到有关ALU的部分，在对标志位已经运算本身的实现上，两者有一些区别，也由共通之处，这让人更加体会到硬件语言与普通语言的不同之处。

10.思考题

思考题1

需要比较A1，B1和运算结果的符号位，因为减法变成加法时才产生了溢出，此时的溢出时加法器产生的，而加法器的操作数是A1，B1，自然需要比较A1， B1与结果

思考题2：

不完全一样，其中方法一是正确的，因为以4位数为例，最小值为-8，最大值为7，这样的情况下， 如若把-8取反加一变为8，是超出了表示的范围的，所以Cin的1不能提前加到B取反的值上，应该在加法执行时一起加上，这样，溢出的产生是在A,B加法时产生的，而不是B取反加一时产生的，对Overflow判断时，不会因为t\_add\_Cin已经溢出而产生错误，这样才不会影响Overflow的判断

思考题3：

缩减运算时verilog的一种特殊语法，可以简单的对一个多位运算数执行自身的位操作，例如，| A等价于A[0] | A[1] | A[2] | A[3]，显然这种方法判断是否为0更加简单，不需要额外的比较器