实验 2：运算器 ALU 实验，熟悉 Vivado 环境

刘泓尊 2018011446 计 84

一、实验目的

1. 熟悉硬件描述语言；
2. 熟悉开发环境 Vivado，了解硬件系统开发的基本过程；
3. 掌握 ALU 基本设计方法和简单运算器的数据传送通路；
4. 验证 ALU 的功能；

二、实验过程

操作码：

将 1-10 的 4 位二进制表示分别作为 Add, Sub, And, Or, Xor, Not, Sll, Srl, Sra, Rol 的操作码；

加减操作与进位、溢出标志：

加减操作直接使用 Verilog 自带的运算。

进位标志使用语句 {cf, result} = a + b; 实现，使得完成加法运算的同时可以记录进位。减法同理。这是因为 Verilog 中算术表达式长度由最长的操作数决定，在赋值语句下，算术操作结果的长度由左端目标长度决定。如果左端目标长度比右侧表达式长度长，那么任何溢出的位都将存储在左侧的高位中。

逻辑运算操作：

逻辑运算直接使用 Verilog 自带的逻辑运算。

移位操作：

逻辑移位使用 << 和 >> ；算数右移需要注意使用 result = $signed(a) >>> b; 而不是 result = a >>> b; 因为 verilog 默认 reg 是无符号的；循环移位使用 result = (a >> (16 - b)) | (a << b);即可，分别将对应位置赋给目标位置。

符号标志和零标志：

符号标志直接取 result 的最高位；零标志检测 result 是否为 0 即可。

状态机：

S0 状态下读取 a 的值；S1 状态下读取 b 的值；S2 状态下读取 op 并将结果显示到 led, S3 状态下输出 flag. 需要注意 op 的值在示例代码中是一直在读取的，所以每次拨码开关改变都会引起结果的改变，只是没有显示到 led 上；同时注意 flag 各个位对应哪个操作码。

三、实验结果记录

提交到 GitLab 并通过了实验平台的评测，结果如下：

实验过程中进行的操作与结果数据：

输入数据			实际输出		一致性
操作码	操作数 A	操作数 B	运算结果	标志位 {cf, zf, sf, vf}	
0001(Add)	0100 0001 1110 0100	0111 1011 1001 0001	1011 1101 0111 0101	{0, 0, 1, 1}	一致
0010(Sub)	0110 0100 1000 0111	1000 1010 0001 0110	1101 1010 0111 0001	{1, 0, 1, 1}	一致
0011(And)	0000 1000 1100 0000	0000 0000 1000 1100	0000 0000 1000 0000	{0, 0, 0, 0}	一致
0100(Or)	0000 1000 1100 0000	0000 0000 1000 1100	0000 1000 1100 1100	{0, 0, 0, 0}	一致
0101(Xor)	0000 1000 1100 0000	0000 0000 1000 1100	0000 1000 0100 1100	{0, 0, 0, 0}	一致
0110(Not)	0000 1000 1100 0000	----	1111 0111 0011 1111	{0, 0, 1, 0}	一致
0111(Sll)	0000 0000 0000 0001	0000 0000 0000 0010	0000 0000 0000 0100	{0, 0, 0, 0}	一致
1000(Srl)	1000 0000 0000 0000	0000 0000 0000 0010	0010 0000 0000 0000	{0, 0, 0, 0}	一致
1001(Sra)	1000 0000 0000 0000	0000 0000 0000 0111	1111 1111 0000 0000	{0, 0, 1, 0}	一致
1010(Rol)	0000 1100 0011 0000	0000 0000 0000 1000	0011 0000 0000 1100	{0, 0, 0, 0}	一致

三、思考题

1. ALU 进行算术逻辑运算所使用的电路是组合逻辑电路还是时序逻辑电路？

使用的是**组合逻辑电路**，任意时刻的输出仅仅取决于该时刻的输入，和电路的状态无关，也不需要记录中间状态。

2. 如果给定了 A 和 B 的初值，且每次运算完后结果都写入到 B 中，再进行下次运算。这样一个带暂存功能的 ALU 要增加一些什么电路来实现？

该功能需要时序逻辑来实现，因为电路的输出取决于当前状态（即暂存的 B）。在输出端后加一个 16 位**寄存器**，保存新的运算结果作为 B。每次 ALU 从寄存器获取 B 的值。寄存器的初值在一开始设为 B 的初值即可。

四、实验小结

本次实验从事后来看比较简单，但是我入门 verilog 还是花了一些精力。此外我还在计算溢出符号时因为不熟悉语言特性，写了很冗长的代码。但是后来经过

查阅手册，发现 verilog 可以实现类似{cf, result} = a + b;的语句，这大大精简了我的代码（虽然到后面得知不需要实现 cf）。多尝试、多查资料是我最大的收获。

感谢老师和助教在微信群中的悉心指导和耐心答疑！您的支持让我少走了很多弯路。