# P1实验报告：关于4/8/32位加法器、GPR以及ALU等模块

140613 14061075 修闽珂

1. 4/8/32位加法器

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[k:0], k = 3/7/31 | I | 第一个加数 |
| B[k:0] , k = 3/7/31 | I | 第二个加数 |
| C0 | I | 初始进位 |
| SUM[k:0] , k = 3/7/31 | O | 两数相加之和 |
| Overflow | O | 溢出标志位  Overflow=C8 |

1. 这三个加法器本质上是类似的。一位加法器中所有的输入输出量都只有一位。四级加法器中每一位都用一个一级加法器运算，每一位都产生一个进位，这个进位用一个wire型变量储存。后来的8位加法器由两个四位加法器构成，每四位产生一个进位，由一个2位的wire型储存。32位加法器由4个八位加法器级联而成。
2. 加法器中我使用了always，其实可以不用的。主要是对always 的本质还不了解。原本还想用for循环来实现，但是对for循环的本质尚未彻底领悟，最终尝试失败只得放弃。
3. ALU模块

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ALU\_DA[31:0] | I | 参与ALU计算的第一个值 |
| ALU\_DB[31:0] | I | 参与ALU计算的第二个值 |
| ALUOp[2:0] | I | ALU功能的选择信号  000：ALU进行加法运算  001：ALU做减法运算  010：ALU进行与运算  100：ALU进行或运算  101：ALU进行异或运算 |
| ALU\_DC[31：0] | O | ALU的计算结果 |
| ALU\_Zero | O | 1：ALU\_DC为0  0：ALU\_DC不为0 |

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加运算 | ALU\_DC= ALU\_DA+ ALU\_DB |
| 2 | 减运算 | ALU\_DC= ALU\_DA- ALU\_DB |
| 3 | 与运算 | ALU\_DC= ALU\_DA& ALU\_DB |
| 4 | 或运算 | ALU\_DC= ALU\_DA|ALU\_DB |
| 5 | 异或运算 | ALU\_DC= ALU\_DA^ALU\_DB |

1. 对ALUOp进行选择，当ALUOp为某某某时，对ALU\_DA和ALU\_DB进行相应操作。对ALU\_DC进行选择，当它为0或1时相应ALU\_Zero进行赋值。选择用 ? : 实现。
2. GPR模块

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | MIPS-C处理器时钟 |
| Reset | I | 复位信号 |
| RS1[4:0] | I | 读寄存器文件时的第1个寄存器下标 |
| RS2[4:0] | I | 读寄存器文件时的第2个寄存器下标 |
| RD[4:0] | I | 写寄存器文件时的寄存器下标 |
| RegWrite | I | 寄存器文件写使能 |
| WData[31:0] | I | 寄存器文件写入数据 |
| RData1[31:0] | O | 读寄存器文件时的第1个寄存器的输出 |
| RData2[31:0] | O | 读寄存器文件时的第2个寄存器的输出 |

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读寄存器 | RData1输出RS1[4:0]所寻址的寄存器。  RData2输出RS2[4:0]所寻址的寄存器。 |
| 2 | 写寄存器 | 当时钟上升沿到来时，并且RegWrite或Reset有效时，WData被写入RD[4:0]所寻址的寄存器 |

1. 设置一个32\*32的寄存器data，data中的数据data[RS1] data[RS2]能被读出，前提是这个地址有数字。Data[RD]会被写入新的数据，在时钟信号的上升沿来到时对RegWrite 和 Reset进行选择，如果reset是有效的则所有数据赋值为0；RegWrite有效则data[RD] 被赋值为WData. 赋值操作被放在always中；RS1和RS2用assign来赋值。
2. 经过在mooc上看视频，我了解到并改正了我对题目的误读之处。
3. 为什么减法操作可以通过加法操作来完成？

计算机中的减法通过加上一个数字的补码来实现。减数变为二进制后求反码再加一得到补码，在字长范围内作加法，超出的那一位去掉，剩下的码所代表的数字就是减法得到的结果。