**组成原理实验课程第 4 次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | ALU模块实现 | | | 班级 | 李涛老师 |
| 学生姓名 | 孙蕗 | 学号 | 2112060 | 指导老师 | 董前琨 |
| 实验地点 | A308 | | 实验时间 | 2023.5.9 | |

1. **实验目的**
2. 熟悉MIPS指令集中的运算指令，学会对这些指令进行归纳分类。
3. 了解MIPS指令结构。
4. 熟悉并掌握ALU的原理、功能和设计。
5. 进一步加强运用verilog语言进行电路设计的能力。
6. 为后续设计cpu的实验打下基础。
7. **实验改进内容**

针对组成原理第四次的ALU实验进行改进，要求：

1. 将原有的操作码进行位压缩，调整操作码控制信号位宽为4位。
2. 操作码调整成4位之后，在原有11种运算的基础之上，自行补充3种不同类型的运算，操作码和运算自行选择，需要上实验箱验证计算结果。
3. 本次实验不用仿真波形，直接上实验箱验证即可。注意改进实验上实验箱验证时，操作码应该已经压缩到4位位宽。
4. 实验报告中的原理图就用图5.3即可，不再是顶层模块图。实验报告中应该有两个表，第一个表为验证实验初始的11种运算，表中列出操作码、操作数和运算结果；第二个表是改进实验后的11+3种运算的验证，表中列出操作码、操作数和运算结果。注意自行添加的三种运算还需要附上实验箱验证照片。
5. **实验设备**
6. 装有Xilinx Vivado的计算机一台。
7. LS-CPU-EXB-002教学系统实验箱一套。
8. **实验任务**
9. 学习MIPS指令集，熟知指令类型，了解指令功能和编码，归纳基础的ALU运算指令。
10. 归纳确定自己本次实验中准备实现的ALU运算，要求不实现定点乘除指令和浮点运算指令，要求至少实现5种ALU运算，其中要包含加减运算，其中减法在内部要转换为加法，与加法运算共同调用实验一里自己完成的加法模块去做。
11. 自行设计本次实验的方案，画出结构框图，大致结构框图如图5.1。图5.1中的操作码位数和类型请自行设计，可以设计为独热码（一位有效编码）或二进制编码。比如，设计方案中预定实现7种ALU运算，则操作码采用独热码，则需7bit数据，每位单独指示一种运算；若采用二进制编码，则只用3bit数据位即可，但在需ALU内部先进行解码，才能确定ALU作何种运算。

32位源操作数1

32位源操作数2

运算结果

操作码

图 5.1 ALU模块的大致框图

1. 根据设计的实验方案，使用verilog编写相应代码。
2. 对编写的代码进行仿真，得到正确的波形图。
3. 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图5.2。外围模块中需调用封装好的LCD触摸屏模块，显示ALU的两个源操作数、操作码和运算结果，并且需要利用触摸功能输入源操作数。操作码可以考虑用LCD触摸屏输入，也可以用拨码开关输入。

外围模块

来自FPGA板子上的输入

输出到FPGA板上进行展示

图 5.2 ALU设计实验的顶层模块大致框图

1. 将编写的代码进行综合布局布线，并下载到试验箱中的FPGA板子上进行演示。
2. **实验原理图**

ALU模块的原理图如下：



图 5.3 ALU的原理图

1. 加减运算原理图

加减运算最后都是调用实验一中的加法器做的：



图 5.4 加减法运算的原理图

1. **小于置位运算**

对于有符号比较的小于置位，是利用减法结果比较的，其比较的真值表如下：

表5.2 有符号比较小于置位的真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **源操作数1符号位** | | **源操作数2符号位** | | **结果符号位** | | **判断** | **slt结果** |
| **alu\_src1[31]** | | **alu\_src2[31]** | | **adder\_result[31]** | |
| 0 | 正数 | 0 | 正数 | 0 | 正数 | 正>正 | 0 |
| 0 | 正数 | 0 | 正数 | 1 | 负数 | 正<正 | 1 |
| 0 | 正数 | 1 | 负数 | X | 无关 | 正>负 | 0 |
| 1 | 负数 | 0 | 正数 | X | 无关 | 负<正 | 1 |
| 1 | 负数 | 1 | 负数 | 0 | 正数 | 负>负 | 0 |
| 1 | 负数 | 1 | 负数 | 1 | 负数 | 负<负 | 1 |

由表5.2可得有符号32位比较小于置位运算结果表达式:

slt\_result = (alu\_src1[31] & ~alu\_src2[31]) | (~(alu\_src1[31]^alu\_src2[31]) & adder\_result[31])

对于32位无符号比较的小于置位，可在其高位前填0组合为33位正数的比较，即{1’b0, src1}和{1’b0, src2}的比较，最高位符号位为0。对比表5.2可知，对于正数的比较，只要减法结果的符号位为1，则表示小于。而33位正数相减，其结果的符号位最终可由32位加法的cout+1’b1得到，如图5.5。故无符号32位比较小于置位运算结果表达式为：sltu\_result = ~adder\_cout.



图 5.5 无符号32位数比较的原理图

1. **实验顶层模块框图**



图 5.6 ALU参考设计的顶层模块框图

1. **参考设计**
2. 做好预习：
3. 熟知指令类型，了解指令功能和编码；
4. 归纳基础的ALU运算指令，确定自己准备实现的ALU运算；
5. 设计本次实验的方案，列出准备实现的ALU运算和操作码的编码；
6. 在课前画好实验方案的设计框图，即补充完善图5.1；
7. 如果对FPGA板了解的话，可确定设计中与FPGA板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图5.2。
8. 实验实施：
9. 确认ALU模块的设计框图的正确性；
10. 编写verilog代码；
11. 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；
12. 完成调用ALU模块的外围模块的设计，并编写代码；
13. 对代码进行综合布局布线下载到试验箱里FPGA板上，进行上板验证。
14. 实验检查：
15. 完成上板验证后，让指导老师或助教进行检查，进行现场演示。先说明自己实现的ALU运算类型，按照检查人员的要求，对特定源操作数进行特定运算操作，检查运算结果的正确性，可对演示结果进行拍照作为实验报告结果一项的材料。
16. 实验报告的撰写：
17. 实验结束后，需按照规定的格式完成实验报告的撰写。
18. 本部分给出的ALU实验的参考设计方案中，共实现了12种运算，采用独热编码操作码(即ALU的控制信号)，对应关系如下：

表5.1 ALU的控制信号

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **控制信号** | | | | | | | | | | | | **ALU操作** |
| **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 无 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 加法 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 减法 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 有符号比较，小于置位 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 无符号比较，小于置位 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 按位与 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 按位或非 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 按位或 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 按位异或 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 逻辑左移 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 逻辑右移 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 算术右移 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 高位加载 |

1. **实验步骤**
2. 修改alu\_display.v代码，实现将原有的操作码进行位压缩，调整操作码控制信号位宽为4位。

|  |  |
| --- | --- |
|  | //-----{调用ALU模块}begin |
|  | reg [**3:0**] alu\_control; // ALU控制信号 |
|  | reg [31:0] alu\_src1; // ALU操作数1 |
|  | reg [31:0] alu\_src2; // ALU操作数2 |
|  | wire [31:0] alu\_result; // ALU结果 |

1. 修改alu.v代码，实现将原有的操作码进行位压缩，调整操作码控制信号位宽为4位。

|  |  |
| --- | --- |
|  | module alu( |
|  | input [**3:0**] alu\_control, // ALU控制信号 |
|  | input [31:0] alu\_src1, // ALU操作数1,为补码 |
|  | input [31:0] alu\_src2, // ALU操作数2，为补码 |
|  | output [31:0] alu\_result // ALU结果 |
|  | ); |
|  |  |
|  | // ALU控制信号，独热码 |
|  | wire alu\_nand; //按位与非 |
|  | wire alu\_lli; //低位加载 |
|  | wire alu\_not; //按位取反 |
|  |  |
|  | wire alu\_add; //加法操作 |
|  | wire alu\_sub; //减法操作 |
|  | wire alu\_slt; //有符号比较，小于置位，复用加法器做减法 |
|  | wire alu\_sltu; //无符号比较，小于置位，复用加法器做减法 |
|  | wire alu\_and; //按位与 |
|  | wire alu\_nor; //按位或非 |
|  | wire alu\_or; //按位或 |
|  | wire alu\_xor; //按位异或 |
|  | wire alu\_sll; //逻辑左移 |
|  | wire alu\_srl; //逻辑右移 |
|  | wire alu\_sra; //算术右移 |
|  | wire alu\_lui; //高位加载 |
|  |  |
|  | **assign alu\_nand = alu\_control == 4'b1110 ? 1 : 0;** |
|  | **assign alu\_lli = alu\_control == 4'b1101 ? 1 : 0;** |
|  | **assign alu\_not = alu\_control == 4'b1100 ? 1 : 0;** |
|  |  |
|  | **assign alu\_add = alu\_control == 4'b1011 ? 1 : 0;** |
|  | **assign alu\_sub = alu\_control == 4'b1010 ? 1 : 0;** |
|  | **assign alu\_slt = alu\_control == 4'b1001 ? 1 : 0;** |
|  | **assign alu\_sltu = alu\_control == 4'b1000 ? 1 : 0;** |
|  | **assign alu\_and = alu\_control == 4'b0111 ? 1 : 0;** |
|  | **assign alu\_nor = alu\_control == 4'b0110 ? 1 : 0;** |
|  | **assign alu\_or = alu\_control == 4'b0101 ? 1 : 0;** |
|  | **assign alu\_xor = alu\_control == 4'b0100 ? 1 : 0;** |
|  | **assign alu\_sll = alu\_control == 4'b0011 ? 1 : 0;** |
|  | **assign alu\_srl = alu\_control == 4'b0010 ? 1 : 0;** |
|  | **assign alu\_sra = alu\_control == 4'b0001 ? 1 : 0;** |
|  | **assign alu\_lui = alu\_control == 4'b0000 ? 1 : 0;** |
|  |  |
|  | wire[31:0] nand\_result; |
|  | wire[31:0] lli\_result; |
|  | wire[31:0] not\_result; |
|  |  |
|  | wire [31:0] add\_sub\_result; |
|  | wire [31:0] slt\_result; |
|  | wire [31:0] sltu\_result; |
|  | wire [31:0] and\_result; |
|  | wire [31:0] nor\_result; |
|  | wire [31:0] or\_result; |
|  | wire [31:0] xor\_result; |
|  | wire [31:0] sll\_result; |
|  | wire [31:0] srl\_result; |
|  | wire [31:0] sra\_result; |
|  | wire [31:0] lui\_result; |
|  |  |
|  | assign nand\_result=~and\_result;//按位与非 |
|  | assign lli\_result = {16'd0,alu\_src2[31:16]};//低位加载 |
|  | assign not\_result=~alu\_src2; //按位取反的结果 |
|  |  |
|  | assign and\_result = alu\_src1 & alu\_src2; // 与结果为两数按位与 |
|  | assign or\_result = alu\_src1 | alu\_src2; // 或结果为两数按位或 |
|  | assign nor\_result = ~or\_result; // 或非结果为或结果按位取反 |
|  | assign xor\_result = alu\_src1 ^ alu\_src2; // 异或结果为两数按位异或 |
|  | assign lui\_result = {alu\_src2[15:0], 16'd0}; // 立即数装载结果为立即数移位至高半字 |

1. 修改alu.v代码，实现按位与非，低位加载，按位取反3种功能

|  |  |
| --- | --- |
|  | module alu( |
|  | input [3:0] alu\_control, // ALU控制信号 |
|  | input [31:0] alu\_src1, // ALU操作数1,为补码 |
|  | input [31:0] alu\_src2, // ALU操作数2，为补码 |
|  | output [31:0] alu\_result // ALU结果 |
|  | ); |
|  |  |
|  | // ALU控制信号，独热码 |
|  | **wire alu\_nand; //按位与非** |
|  | **wire alu\_lli; //低位加载** |
|  | **wire alu\_not; //按位取反** |
|  |  |
|  | wire alu\_add; //加法操作 |
|  | wire alu\_sub; //减法操作 |
|  | wire alu\_slt; //有符号比较，小于置位，复用加法器做减法 |
|  | wire alu\_sltu; //无符号比较，小于置位，复用加法器做减法 |
|  | wire alu\_and; //按位与 |
|  | wire alu\_nor; //按位或非 |
|  | wire alu\_or; //按位或 |
|  | wire alu\_xor; //按位异或 |
|  | wire alu\_sll; //逻辑左移 |
|  | wire alu\_srl; //逻辑右移 |
|  | wire alu\_sra; //算术右移 |
|  | wire alu\_lui; //高位加载 |
|  |  |
|  | **assign alu\_nand = alu\_control == 4'b1110 ? 1 : 0;** |
|  | **assign alu\_lli = alu\_control == 4'b1101 ? 1 : 0;** |
|  | **assign alu\_not = alu\_control == 4'b1100 ? 1 : 0;** |
|  |  |
|  | assign alu\_add = alu\_control == 4'b1011 ? 1 : 0; |
|  | assign alu\_sub = alu\_control == 4'b1010 ? 1 : 0; |
|  | assign alu\_slt = alu\_control == 4'b1001 ? 1 : 0; |
|  | assign alu\_sltu = alu\_control == 4'b1000 ? 1 : 0; |
|  | assign alu\_and = alu\_control == 4'b0111 ? 1 : 0; |
|  | assign alu\_nor = alu\_control == 4'b0110 ? 1 : 0; |
|  | assign alu\_or = alu\_control == 4'b0101 ? 1 : 0; |
|  | assign alu\_xor = alu\_control == 4'b0100 ? 1 : 0; |
|  | assign alu\_sll = alu\_control == 4'b0011 ? 1 : 0; |
|  | assign alu\_srl = alu\_control == 4'b0010 ? 1 : 0; |
|  | assign alu\_sra = alu\_control == 4'b0001 ? 1 : 0; |
|  | assign alu\_lui = alu\_control == 4'b0000 ? 1 : 0; |
|  |  |
|  | **wire[31:0] nand\_result;** |
|  | **wire[31:0] lli\_result;** |
|  | **wire[31:0] not\_result;** |
|  |  |
|  | wire [31:0] add\_sub\_result; |
|  | wire [31:0] slt\_result; |
|  | wire [31:0] sltu\_result; |
|  | wire [31:0] and\_result; |
|  | wire [31:0] nor\_result; |
|  | wire [31:0] or\_result; |
|  | wire [31:0] xor\_result; |
|  | wire [31:0] sll\_result; |
|  | wire [31:0] srl\_result; |
|  | wire [31:0] sra\_result; |
|  | wire [31:0] lui\_result; |
|  |  |
|  | **assign nand\_result=~and\_result;//按位与非** |
|  | **assign lli\_result = {16'd0,alu\_src2[31:16]};//低位加载** |
|  | **assign not\_result=~alu\_src2; //按位取反的结果** |
|  |  |
|  | assign and\_result = alu\_src1 & alu\_src2; // 与结果为两数按位与 |
|  | assign or\_result = alu\_src1 | alu\_src2; // 或结果为两数按位或 |
|  | assign nor\_result = ~or\_result; // 或非结果为或结果按位取反 |
|  | assign xor\_result = alu\_src1 ^ alu\_src2; // 异或结果为两数按位异或 |
|  | assign lui\_result = {alu\_src2[15:0], 16'd0}; // 立即数装载结果为立即数移位至高半字 |

|  |  |
| --- | --- |
|  | // 选择相应结果输出 |
|  | **assign alu\_result = alu\_nand ? nand\_result :** |
|  | **alu\_lli ? lli\_result :** |
|  | **alu\_not ? not\_result :** |
|  | (alu\_add|alu\_sub) ? add\_sub\_result[31:0] : |
|  | alu\_slt ? slt\_result : |
|  | alu\_sltu ? sltu\_result : |
|  | alu\_and ? and\_result : |
|  | alu\_nor ? nor\_result : |
|  | alu\_or ? or\_result : |
|  | alu\_xor ? xor\_result : |
|  | alu\_sll ? sll\_result : |
|  | alu\_srl ? srl\_result : |
|  | alu\_sra ? sra\_result : |
|  | alu\_lui ? lui\_result : |
|  | 32'd0; |

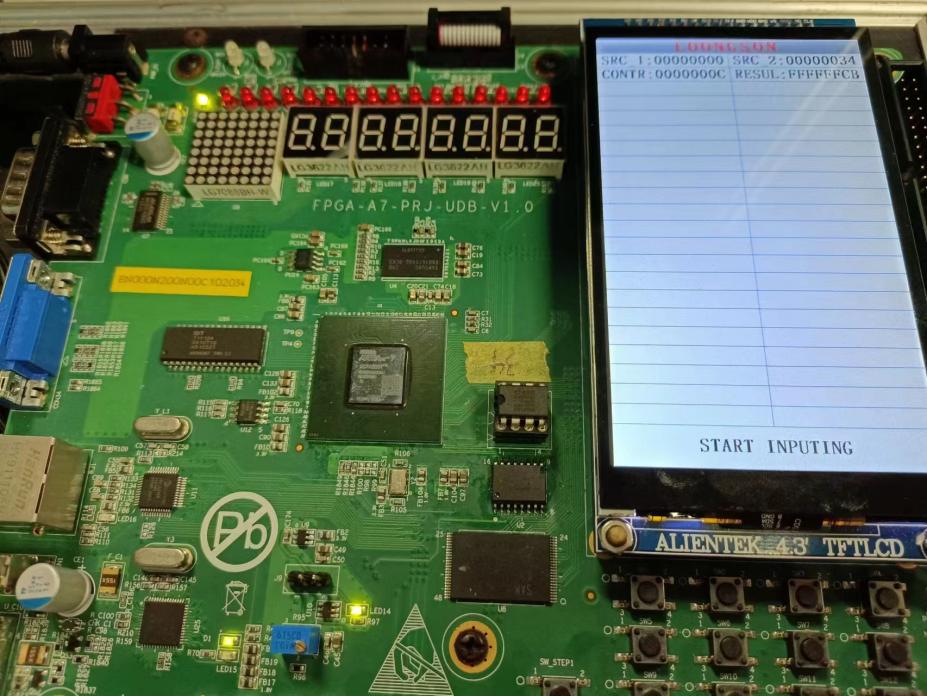
1. **实验结果分析**
2. 实验手册上的代码

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **操作** | **ALU操作数1alu\_src1** | **ALU操作数2alu\_src2** | **ALU控制信号alu\_control** | **ALU结果alu\_result** |
| 加法 | 11111111 | 22222222 | 00000800 | 33333333 |
| 减法 | 22222222 | 11111111 | 00000400 | 11111111 |
| 有符号比较，小于置位（正数） | 00000000 | 11111111 | 00000200 | 00000001 |
| 有符号比较，小于置位（负数） | 00000002 | FFFFFFFD | 00000200 | 00000000 |
| 无符号比较，小于置位 | 00000000 | 11111111 | 00000100 | 00000001 |
| 按位与 | 10010011 | 11111111 | 00000080 | 10010011 |
| 按位或非 | 10010011 | 11111111 | 00000040 | EEEEEEEE |
| 按位或 | 10010011 | 11111111 | 00000020 | 11111111 |
| 按位异或 | 10010011 | 11111111 | 00000010 | 01101100 |
| 逻辑左移 | 00000004 | 00000004 | 00000008 | 00000040 |
| 逻辑右移 | 00000001 | 00000004 | 00000004 | 00000002 |
| 算术右移 | 00000002 | FF000000 | 00000002 | FFC00000 |
| 高位加载 | 22223333 | 00002222 | 00000001 | 22220000 |

1. 改过的代码

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **操作** | **ALU操作数1alu\_src1** | **ALU操作数2alu\_src2** | **ALU控制信号alu\_control** | **ALU结果alu\_result** |
| 加法 | 11111111 | 22222222 | 0000000B | 33333333 |
| 减法 | 22222222 | 11111111 | 0000000A | 11111111 |
| 有符号比较，小于置位（正数） | 00000000 | 11111111 | 00000009 | 00000001 |
| 有符号比较，小于置位（负数） | 00000002 | FFFFFFFD | 00000009 | 00000000 |
| 无符号比较，小于置位 | 00000000 | 11111111 | 00000008 | 00000001 |
| 按位与 | 10010011 | 11111111 | 00000007 | 10010011 |
| 按位或非 | 10010011 | 11111111 | 00000006 | EEEEEEEE |
| 按位或 | 10010011 | 11111111 | 00000005 | 11111111 |
| 按位异或 | 10010011 | 11111111 | 00000004 | 01101100 |
| 逻辑左移 | 00000004 | 00000004 | 00000003 | 00000040 |
| 逻辑右移 | 00000001 | 00000004 | 00000002 | 00000002 |
| 算术右移 | 00000002 | FF000000 | 00000001 | FFC00000 |
| 高位加载 | 22223333 | 00002222 | 00000000 | 22220000 |
| 按位取反 | 00000000 | 00000034 | 0000000C | FFFFFFCB |
| 低位加载 | 00000000 | 12345678 | 0000000D | 00001234 |
| 按位与非 | 00001100 | 00001010 | 0000000E | FFFFEFFF |

1. 自行添加的三种运算附实验箱验证照片
2. 按位取反



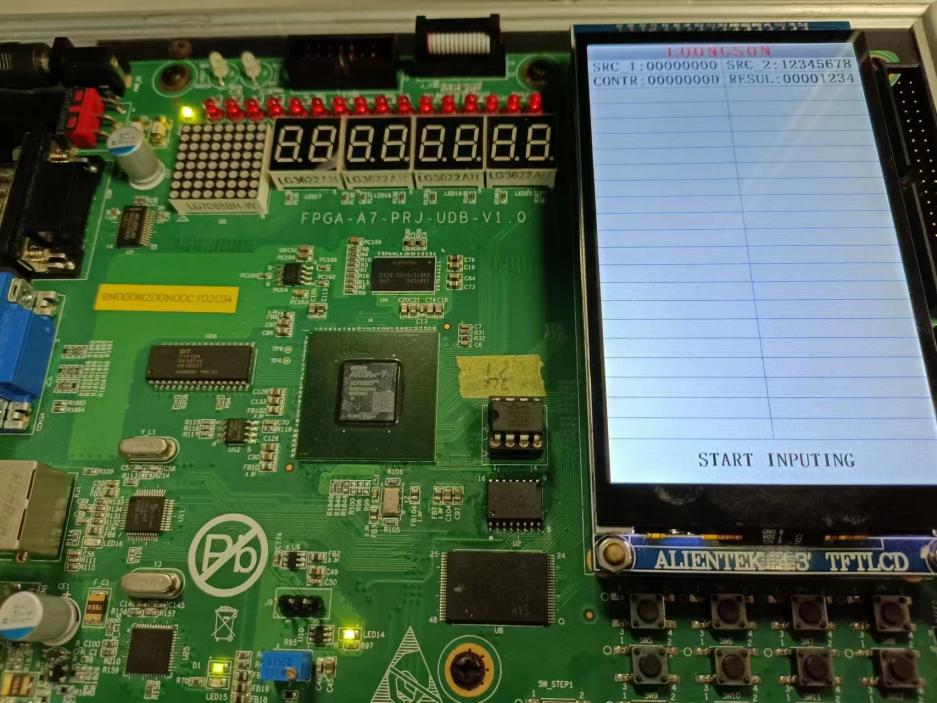
alu\_src1=00000000，alu\_src2=00000034。

alu\_control=0000000C，按位取反。

16进制alu\_src2按位取反的结果是FFFFFFCB。

显示alu\_result=FFFFFFCB。结果正确

1. 低位加载



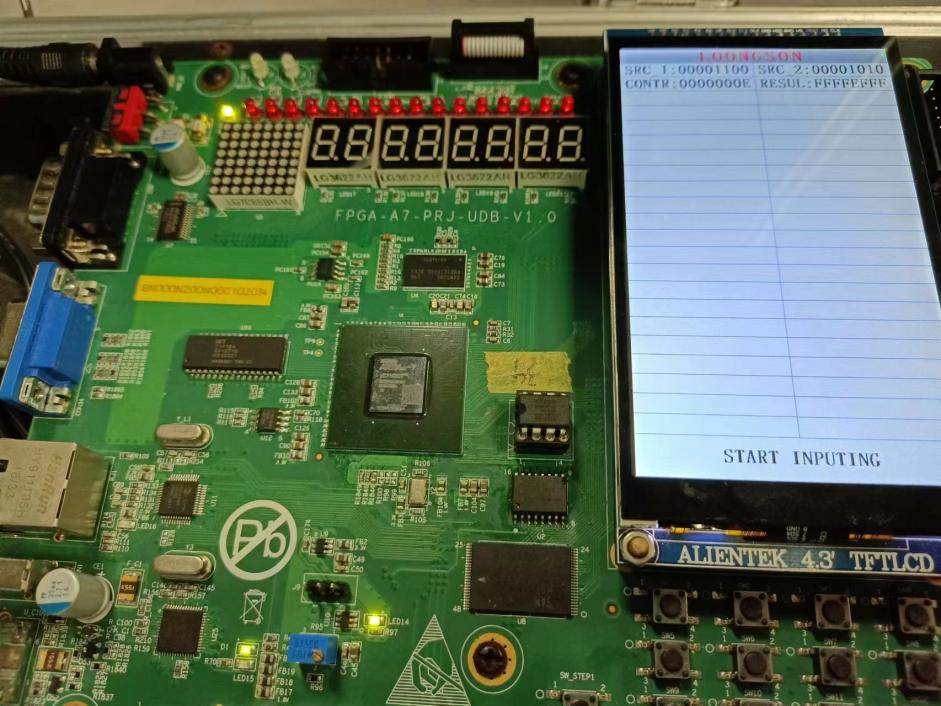
alu\_src1=00000000，alu\_src2=12345678。

alu\_control=0000000D，低位加载。

alu\_src2低位加载的结果是00001234。

显示alu\_result=00001234。结果正确

1. 按位与非



alu\_src1=00001100，alu\_src2=00001010。

alu\_control=0000000E，按位与非。

16进制alu\_src1和alu\_src2按位与非的结果是FFFFEFFF。

显示alu\_result=FFFFEFFF。结果正确

1. **总结感想**
2. 通过本次实验，对ALU的指令集的功能与编码有了更深入的了解。在讨论、学习中，学会了如何编写实现一条指令：如何定义、如何计算、如何显示结果等。
3. 通过代码修改，实现将原有的操作码进行位压缩，调整操作码控制信号位宽为4位。
4. 添加了按位与非，低位加载，按位取反3种功能，通过与、非运算并加入新指令，实现3种功能。