**实验报告**

2021年05月11日 成绩：---------

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 刘爱兵 | 学号 | 19114801 | 班级 | 19计科2班 |
| 专业 | 计算机科学与技术 | 课程名称 | 计算机组成原理实验课 | 任课教师 | 吴云志 |
| 实验序号 | 02 | 实验名称 | 多功能ALU设计 | | |
| 实验时间 | 2021.05.11 | 机位号 | 18 | 实验设备号 | 21 |

1. **实验目的与要求**
2. 实验目的

学习多功能ALU的工作原理，掌握运算器的设计方法

掌握运用Verilog HDL语言进行行为描述与建模的技巧和方法

1. 实验要求

使用行为描述方式（case语句），编程实现多功能ALU模块，供后续CPU的设计使用。

编写验证模块验证ALU模块功能正确性（选择的测试数据要有代表性）

课前任务：编程、仿真、验证，确保逻辑正确性；

实验室任务：

1. 配置管脚：见下表。

|  |  |  |  |
| --- | --- | --- | --- |
|  | **信号** | **配置设备管脚** | **功能说明** |
|  | A、B | 32个逻辑开关 | 选择32位A或B数据 |
| 输入输出控制 | 按钮1~6 | 按钮1：按一下ALU\_OP自增一  按钮2：输入A  按钮3：输入B  按钮4：按下时，数码管显示当前操作数A  按钮5：按下时，数码管显示当前操作数B  按钮6：按下时，数码管显示运算结果 |
| 输出信号 | LED[31:0]  数码管 | 32个LED灯 | 显示运算结果F |
| 8段数码管 | 默认最高位显示运算符编码，最低两位显示零状态、溢出状态标识，中间其他位显示零 |

（2）生成\*.bit文件，下载到HDU-XL-01教学开发板的FPGA中。

（3）完成板级调试。

（4）撰写实验报告。

1. **实验设计与程序代码**
2. 模块设计说明

（描述这个实验的设计方案，分几个模块，各模块的功能，各模块之间的连接关系，可附图）



1. 实验程序源代码及注释等

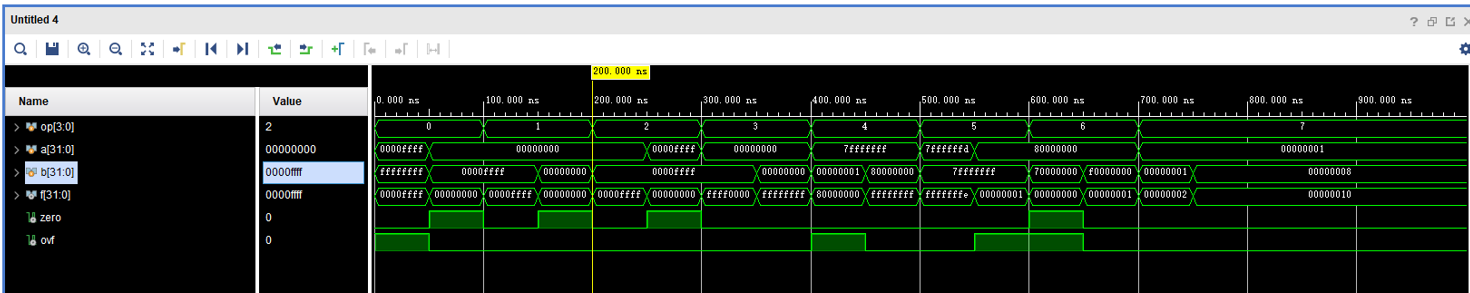
（实验各个模块的代码，包含功能注释）

|  |
| --- |
| 1. ALU.v   `timescale 1ns / 1ps  // Arithmetic Logic Unit，算术逻辑单元  module ALU(op, a, b, f, zero, ovf);  parameter N = 32; // bit witdh，数据位宽  input [3:0] op; // operator，运算符编码  input [N:1] a, b; // operands，左右操作数  output reg [N:1] f; // result，运算结果  reg cout = 0; // carry-out，最高位进位  always @\* case (op) // 判断运算符并进行运算  0: {cout, f} <= a & b;  1: {cout, f} <= a | b;  2: {cout, f} <= a ^ b;  3: {cout, f} <= ~(a | b);  4: {cout, f} <= a + b;  5: {cout, f} <= a - b;  6: {cout, f} <= a < b;  7: {cout, f} <= b << a;  default: {cout, f} <= 0;  endcase  output zero, ovf; // zero, overflow，零、溢出  assign zero = ~(|f); // 结果归约或、取反得到，即结果全零则为 1，否则为 0  assign ovf = a[N] ^ b[N] ^ f[N] ^ cout; // 由判溢公式计算（只对有符号数有意义）  // output neg, even; // negative, even(parity)，负数、偶数  // assign neg = f[N]; // 取结果最高位，补码符号位 0正 1负（只对有符号数有意义）  // assign even = ~^F; // 结果中 1 为偶数个则置 1，否则置 0  endmodule // ALU   1. Board.v   `timescale 1ns / 1ps  // 算术逻辑单元板极调试  module Board(sw, swb, led, clk, which, seg, enable);  output [1:32] led;  reg [3:0] alu\_op;  reg [31:0] alu\_a, alu\_b;  wire alu\_zero, alu\_ovf;  ALU ALU\_Instance(.op(alu\_op), .a(alu\_a), .b(alu\_b),  .f(led), .zero(alu\_zero), .ovf(alu\_ovf)); // LED：运算结果  input clk; // 数码管相关  output [2:0] which;  output [7:0] seg;  output reg enable = 1; // 默认开启数码管使能  reg [31:0] data;  Display Display\_Instance(.clk(clk), .data(data), .which(which), .seg(seg));  input [1:32] sw;  input [1:6] swb;  always @(posedge swb[1]) alu\_op <= alu\_op + 1; // SWB1：正向循环运算符编码  always @(posedge swb[2]) alu\_a <= sw; // SWB2：提交开关数据到左操作数  always @(posedge swb[3]) alu\_b <= sw; // SWB3：提交开关数据到右操作数  always @\* if (swb[4]) data <= alu\_a; // SWB4：按住时，数码管显示左操作数  else if (swb[5]) data <= alu\_b; // SWB5：按住时，数码管显示右操作数  else if (swb[6]) data <= led; // SWB6：按住时，数码管显示运算结果  else data <= {alu\_op, 23'b0, alu\_zero, 3'b0, alu\_ovf}; // 数码管默认最高位显示运算符编码，最低两位显示零状态、溢出状态标识  endmodule // Board   1. Display.v   `timescale 1ns / 1ps  // 8 位七段数码管扫描显示模块  module Display(clk, data, which, seg,  count, digit); // 调试接口  input clk; // 接入系统时钟  input [32:1] data; // 32 位显示数据  output reg [2:0] which = 0; // 片选编码（驱动哪一位数码管），低电平有效  output reg [7:0] seg; // 段选信号（点亮哪些笔划），低电平有效  output reg [10:0] count = 0; // 分频扫描，从左至右循环驱动每一位数码管  always @(posedge clk) count <= count + 1'b1;  always @(negedge clk) if (&count) which <= which + 1'b1;  output reg [3:0] digit; // 显示数据 片选得到 十六进制数码  always @\* case (which)  0: digit <= data[32:29]; // 最高位  1: digit <= data[28:25];  2: digit <= data[24:21];  3: digit <= data[20:17];  4: digit <= data[16:13];  5: digit <= data[12:09];  6: digit <= data[08:05];  7: digit <= data[04:01]; // 最低位  endcase  always @\* case (digit) // 十六进制数码 转换为 段选信号（a,b,c,...g,dp）  4'h0: seg <= 8'b0000\_0011; // 除 g、dp 外全亮，显示数码 0  4'h1: seg <= 8'b1001\_1111; // 仅 b、c 亮，显示数码 1  4'h2: seg <= 8'b0010\_0101;  4'h3: seg <= 8'b0000\_1101;  4'h4: seg <= 8'b1001\_1001;  4'h5: seg <= 8'b0100\_1001;  4'h6: seg <= 8'b0100\_0001;  4'h7: seg <= 8'b0001\_1111;  4'h8: seg <= 8'b0000\_0001;  4'h9: seg <= 8'b0000\_1001;  4'hA: seg <= 8'b0001\_0001;  4'hB: seg <= 8'b1100\_0001;  4'hC: seg <= 8'b0110\_0011;  4'hD: seg <= 8'b1000\_0101;  4'hE: seg <= 8'b0110\_0001;  4'hF: seg <= 8'b0111\_0001;  endcase  endmodule // Display |

1. **实验仿真**
2. 仿真代码（仿真源代码）

|  |
| --- |
| `timescale 1ns / 1ps  // 算术逻辑单元仿真测试  module ALU\_Test();  reg [3:0] op;  reg [31:0] a, b; // input  wire [31:0] f;  wire zero, ovf; // output  ALU ALU\_Instance(.op(op), .a(a), .b(b), .f(f), .zero(zero), .ovf(ovf));  initial begin // #800 f zero ovf  op = 0; // bitwise logical and，按位逻辑与  a = 32'h0000\_ffff; b = 32'hffff\_ffff; #50; //  a = 32'h0000\_0000; b = 32'h0000\_ffff; #50; //  op = 1; // bitwise logical or，按位逻辑或  a = 32'h0000\_0000; b = 32'h0000\_ffff; #50; //  a = 32'h0000\_0000; b = 32'h0000\_0000; #50; //  op = 2; // bitwise logical exclusive or，按位逻辑异或  a = 32'h0000\_0000; b = 32'h0000\_ffff; #50; //  a = 32'h0000\_ffff; b = 32'h0000\_ffff; #50; //  op = 3; // bitwise logical nor，按位逻辑或非  a = 32'h0000\_0000; b = 32'h0000\_ffff; #50; //  a = 32'h0000\_0000; b = 32'h0000\_0000; #50; //  op = 4; // arithmetic signed add，算数有符号加法  a = 32'h7fff\_ffff; b = 32'h0000\_0001; #50; //  a = 32'h7fff\_ffff; b = 32'h8000\_0000; #50; //  op = 5; // arithmetic signed subtract，算数有符号减法  a = 32'h7fff\_fffd; b = 32'h7fff\_ffff; #50; //  a = 32'h8000\_0000; b = 32'h7fff\_ffff; #50; // 1  op = 6; // set on less than (unsigned)，无符号数小于置位（a < b 则结果为 1，否则为 0）  a = 32'h8000\_0000; b = 32'h7000\_0000; #50; //  a = 32'h8000\_0000; b = 32'hf000\_0000; #50; //  op = 7; // shift left logical variable (shift in zeros)，将 b 逻辑左移 a 位，右端补零  a = 32'h0000\_0001; b = 32'h0000\_0001; #50; //  a = 32'h0000\_0001; b = 32'h0000\_0008; #50; //  end  endmodule // ALU\_Test |

1. 仿真波形（运行仿真时波形截图）



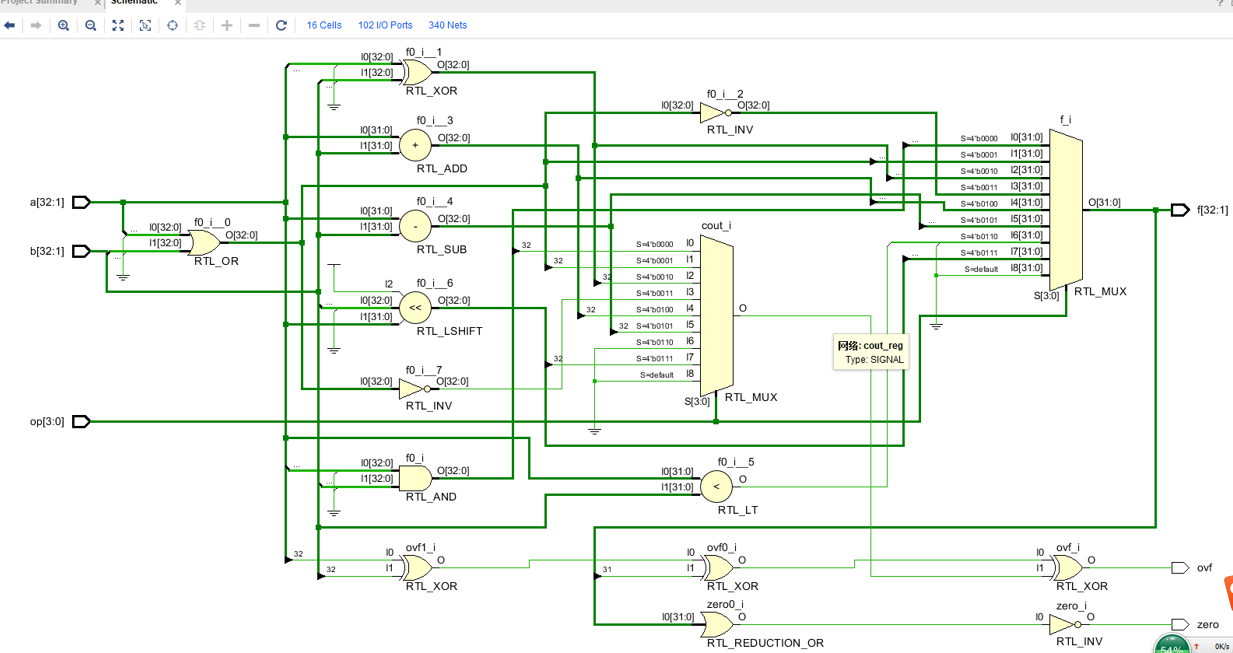
1. 仿真结果分析（对仿真波形进行分析）

时钟周期开始后，每100ns操作数OP自增一次，到800ns停止。

输出值F根据操作码和输入的A、B来计算。

1. **电路图**

（开发工具中显示的电路模块图）



1. **引脚配置**

（引脚约束文件的内容，描述主要配置情况）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **信号** | **配置设备管脚** | **功能说明** |
|  | A、B | 32个逻辑开关 | 选择32位A或B数据 |
| 输入输出控制 | 按钮1~6 | 按钮1：按一下ALU\_OP自增一  按钮2：输入A  按钮3：输入B  按钮4：按下时，数码管显示当前操作数A  按钮5：按下时，数码管显示当前操作数B  按钮6：按下时，数码管显示运算结果 |
| 输出信号 | LED[31:0]  数码管 | 32个LED灯 | 显示运算结果F |
| 8段数码管 | 默认最高位显示运算符编码，最低两位显示零状态、溢出状态标识，中间其他位显示零 |

# 开启比特流压缩，优化 .bit 文件大小

|  |
| --- |
| set\_property BITSTREAM.GENERAL.COMPRESS TRUE [current\_design]  # Switch，开关  set\_property PULLDOWN true [get\_ports sw]  set\_property IOSTANDARD LVCMOS18 [get\_ports sw]  set\_property PACKAGE\_PIN T3 [get\_ports {sw[1]}]  set\_property PACKAGE\_PIN U3 [get\_ports {sw[2]}]  set\_property PACKAGE\_PIN T4 [get\_ports {sw[3]}]  set\_property PACKAGE\_PIN V3 [get\_ports {sw[4]}]  set\_property PACKAGE\_PIN V4 [get\_ports {sw[5]}]  set\_property PACKAGE\_PIN W4 [get\_ports {sw[6]}]  set\_property PACKAGE\_PIN Y4 [get\_ports {sw[7]}]  set\_property PACKAGE\_PIN Y6 [get\_ports {sw[8]}]  set\_property PACKAGE\_PIN W7 [get\_ports {sw[9]}]  set\_property PACKAGE\_PIN Y8 [get\_ports {sw[10]}]  set\_property PACKAGE\_PIN Y7 [get\_ports {sw[11]}]  set\_property PACKAGE\_PIN T1 [get\_ports {sw[12]}]  set\_property PACKAGE\_PIN U1 [get\_ports {sw[13]}]  set\_property PACKAGE\_PIN U2 [get\_ports {sw[14]}]  set\_property PACKAGE\_PIN W1 [get\_ports {sw[15]}]  set\_property PACKAGE\_PIN W2 [get\_ports {sw[16]}]  set\_property PACKAGE\_PIN Y1 [get\_ports {sw[17]}]  set\_property PACKAGE\_PIN AA1 [get\_ports {sw[18]}]  set\_property PACKAGE\_PIN V2 [get\_ports {sw[19]}]  set\_property PACKAGE\_PIN Y2 [get\_ports {sw[20]}]  set\_property PACKAGE\_PIN AB1 [get\_ports {sw[21]}]  set\_property PACKAGE\_PIN AB2 [get\_ports {sw[22]}]  set\_property PACKAGE\_PIN AB3 [get\_ports {sw[23]}]  set\_property PACKAGE\_PIN AB5 [get\_ports {sw[24]}]  set\_property PACKAGE\_PIN AA6 [get\_ports {sw[25]}]  set\_property PACKAGE\_PIN R2 [get\_ports {sw[26]}]  set\_property PACKAGE\_PIN R3 [get\_ports {sw[27]}]  set\_property PACKAGE\_PIN T6 [get\_ports {sw[28]}]  set\_property PACKAGE\_PIN R6 [get\_ports {sw[29]}]  set\_property PACKAGE\_PIN U7 [get\_ports {sw[30]}]  set\_property PACKAGE\_PIN AB7 [get\_ports {sw[31]}]  set\_property PACKAGE\_PIN AB8 [get\_ports {sw[32]}]  # Switch Button，按钮  set\_property IOSTANDARD LVCMOS18 [get\_ports swb]  set\_property PACKAGE\_PIN R4 [get\_ports {swb[1]}]  set\_property PACKAGE\_PIN AA4 [get\_ports {swb[2]}]  set\_property PACKAGE\_PIN AB6 [get\_ports {swb[3]}]  set\_property PACKAGE\_PIN T5 [get\_ports {swb[4]}]  set\_property PACKAGE\_PIN V8 [get\_ports {swb[5]}]  set\_property PACKAGE\_PIN AA8 [get\_ports {swb[6]}]  # LED  set\_property IOSTANDARD LVCMOS18 [get\_ports led]  set\_property PACKAGE\_PIN R1 [get\_ports {led[1]}]  set\_property PACKAGE\_PIN P2 [get\_ports {led[2]}]  set\_property PACKAGE\_PIN P1 [get\_ports {led[3]}]  set\_property PACKAGE\_PIN N2 [get\_ports {led[4]}]  set\_property PACKAGE\_PIN M1 [get\_ports {led[5]}]  set\_property PACKAGE\_PIN M2 [get\_ports {led[6]}]  set\_property PACKAGE\_PIN L1 [get\_ports {led[7]}]  set\_property PACKAGE\_PIN J2 [get\_ports {led[8]}]  set\_property PACKAGE\_PIN G1 [get\_ports {led[9]}]  set\_property PACKAGE\_PIN E1 [get\_ports {led[10]}]  set\_property PACKAGE\_PIN D2 [get\_ports {led[11]}]  set\_property PACKAGE\_PIN A1 [get\_ports {led[12]}]  set\_property PACKAGE\_PIN L3 [get\_ports {led[13]}]  set\_property PACKAGE\_PIN G3 [get\_ports {led[14]}]  set\_property PACKAGE\_PIN K4 [get\_ports {led[15]}]  set\_property PACKAGE\_PIN G4 [get\_ports {led[16]}]  set\_property PACKAGE\_PIN K1 [get\_ports {led[17]}]  set\_property PACKAGE\_PIN J1 [get\_ports {led[18]}]  set\_property PACKAGE\_PIN H2 [get\_ports {led[19]}]  set\_property PACKAGE\_PIN G2 [get\_ports {led[20]}]  set\_property PACKAGE\_PIN F1 [get\_ports {led[21]}]  set\_property PACKAGE\_PIN E2 [get\_ports {led[22]}]  set\_property PACKAGE\_PIN D1 [get\_ports {led[23]}]  set\_property PACKAGE\_PIN B1 [get\_ports {led[24]}]  set\_property PACKAGE\_PIN B2 [get\_ports {led[25]}]  set\_property PACKAGE\_PIN N3 [get\_ports {led[26]}]  set\_property PACKAGE\_PIN M3 [get\_ports {led[27]}]  set\_property PACKAGE\_PIN K3 [get\_ports {led[28]}]  set\_property PACKAGE\_PIN H3 [get\_ports {led[29]}]  set\_property PACKAGE\_PIN N4 [get\_ports {led[30]}]  set\_property PACKAGE\_PIN L4 [get\_ports {led[31]}]  set\_property PACKAGE\_PIN J4 [get\_ports {led[32]}]  # 数码管相关  set\_property IOSTANDARD LVCMOS18 [get\_ports seg]  set\_property PACKAGE\_PIN H19 [get\_ports {seg[7]}]  set\_property PACKAGE\_PIN G20 [get\_ports {seg[6]}]  set\_property PACKAGE\_PIN J22 [get\_ports {seg[5]}]  set\_property PACKAGE\_PIN K22 [get\_ports {seg[4]}]  set\_property PACKAGE\_PIN K21 [get\_ports {seg[3]}]  set\_property PACKAGE\_PIN H20 [get\_ports {seg[2]}]  set\_property PACKAGE\_PIN H22 [get\_ports {seg[1]}]  set\_property PACKAGE\_PIN J21 [get\_ports {seg[0]}]  set\_property IOSTANDARD LVCMOS18 [get\_ports which]  set\_property PACKAGE\_PIN N22 [get\_ports {which[0]}]  set\_property PACKAGE\_PIN M21 [get\_ports {which[1]}]  set\_property PACKAGE\_PIN M22 [get\_ports {which[2]}]  set\_property -dict {IOSTANDARD LVCMOS18 PACKAGE\_PIN L21} [get\_ports enable]  set\_property -dict {IOSTANDARD LVCMOS18 PACKAGE\_PIN H4} [get\_ports clk]  # [Place 30-574] Poor placement for routing between an IO pin and BUFG.If this  # sub optimal condition is acceptable for this design, you may use the  # CLOCK\_DEDICATED\_ROUTE constraint in the .xdc file to demote this message to a  # WARNING. However, the use of this override is highly discouraged.  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets clk\_IBUF]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[1]]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[2]]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[3]]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[4]]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[5]]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[6]] |

1. **思考与探索**
2. 实验结果记录：

（实验操作的过程及结果记录）

时钟周期开始后，每100ns操作数OP自增一次，到800ns停止。

输出值F根据操作码和输入的A、B来计算。

1. 实验结论

（分析实验结果，给出实验结论）

实验结果符合ALU计算逻辑。

1. 问题与解决方案：

（整个实验过程中发生了什么问题？你是如何解决的。）

代码补充遇到困难，经过和同学交流合作完成代码。

4、思考与探索

（1）记录实验结果到表中，分析结果是否符合预期；如果不符合，请分析原因。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A=7FFFFFFD  B=7FFFFFFF | A=703097F9  B=7754D8B0 | A=9DDC1C91  B=E7759683 | A=83B2DAC2  B=31D27BF7 |
|  | F=7FFFFFFD  ZF=0 OF=0 | F=70109060  ZF=0 OF=0 | F=85541881  ZF=0 OF=1 | F=01925AC2  ZF=0 OF=1 |
|  | F=7FFFFFFF  ZF=0 OF=0 | F=7774DFF9  ZF=0 OF=0 | F=FFFD9F93  ZF=0 OF=1 | F=B3F2FBF7  ZF=0 OF=0 |
|  | F=00000020  ZF=0 OF=0 | F=07644F49  ZF=0 OF=0 | F=7AA98712  ZF=0 OF=0 | F=B260A135  ZF=0 OF=0 |
|  | F=80000000  ZF=0 OF=0 | F=888B2006  ZF=0 OF=0 | F=0002606C  ZF=0 OF=1 | F=4C0D0408  ZF=0 OF=0 |
| A加B | F=FFFFFFFC  ZF=0 OF=1 | F=E78570A9  ZF=0 OF=1 | F=85516814  ZF=0 OF=0 | F=B58556B9  ZF=0 OF=0 |
| A减B | F=FFFFFFFE  ZF=0 OF=0 | F=F8DBBF49  ZF=0 OF=0 | F=B666810E  ZF=0 OF=0 | F=51E05ECB  ZF=0 OF=1 |
| A<B置1 | F=00000001  ZF=0 OF=0 | F=00000001  ZF=0 OF=0 | F=00000001  ZF=0 OF=0 | F=00000000  ZF=1 OF=1 |
| B<<A | F=00000000  ZF=1 OF=0 | F=00000000  ZF=1 OF=0 | F=00000000  ZF=1 OF=0 | F=00000000  ZF=1 OF=1 |

1. 研究MIPS的核心指令集，结合本实验所实现的ALU，分析该ALU能否实现MIPS核心指令集的所有指令？如果不能，它还需要哪些运算功能？

答：不能。

add/addu:把两个定点寄存器的内容相加add $1,$2,$3($1=$2+$3);u为不带符号加。

addi/addiu:把一个寄存器的内容加上一个立即数add $1,$2,#3($1=$2+3);u为不带符号加。   
sub/subu：把两个定点寄存器的内容相减。   
div/divu:两个定点寄存器的内容相除。   
mul/mulu:两个定点寄存器的内容相乘。   
and/andi:与运算，两个寄存器中的内容相与and $1,$2,$3($1=$2 & $3);i为立即数。   
or/ori:或运算。   
xor/xori:异或运算。   
beq/beqz/benz/bne:条件转移eq相等，z零，ne不等。

（3）从你的实验结果中能否对以下问题得出结论：slt操作中“A＜B”，是对有符号数的比较还是无符号数的比较？如果不能，请考虑对A=32’b0、B=32’bFFFF\_ FFFF做slt操作实验，得出该问题的结论。

答：无符号数。

F=00000000

ZF=0

OF=1

（4）80X86CPU除了ZF和OF，还有其他标志位，试理解80X86的3个标志位SF、CF和PF的作用和逻辑，将这3个标志位实现到MIPS的多功能ALU中，并完成板级调试。：

SF：符号标志，SF=F[31]。

PF：奇偶标志，运算结果中有偶数个“1”，则PF=1；有奇数个“1”，则PF=0。

CF：进位/借位标志，它对无符号数运算有效。

当进行加法运算时，C32=1，则CF=1； C32=0，则CF=0；

当进行减法运算时，C32=1，则CF=0；C32=0，则CF=1。

1. 实验中的ALU实现了逻辑左移操作sll，考虑逻辑右移和算术右移操作如何实现？考虑为何MIPS指令系统中没有算术左移指令？

答：逻辑右移：右移一次，最高位补0

算数右移：右移一次，最高位保持不变，最低位送入CF

（6）谈谈你在实验中碰到了哪些问题？又是如何解决的？

代码补充遇到困难，经过和同学交流合作完成代码。