**中山大学数据科学与计算机学院本科生实验报告**

**（2019学年秋季学期）**

课程名称：**计算机组成原理实验**  任课教师： 郭雪梅 助教：丁文、汪庭葳

|  |  |  |  |
| --- | --- | --- | --- |
| 年级&班级 | **2019级04班** | 专业(方向) | **计算机科学与技术（超算方向）** |
| 学号 | **19335112** | 姓名 | **李钰** |
| 电话 | **19847352856** | Email | **1643589912@qq.com** |
| 开始日期 | **2020年10月 23日** | 完成日期 | **2020年11月6日星期五** |

**一、实验题目**

ALU功能的实现,以及符号位、标志位的扩展实现

1. **实验目的**

(1) 了解运算器的组成结构。

(2) 掌握运算器的工作原理。

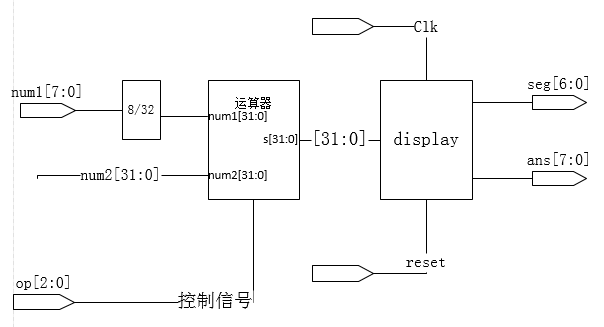
(3) 掌握数码管的工作原理与使用方法，学会IP核封装调用。

**三、实验内容**

**1.** 实验步骤

1. 创建display IP核
2. 创建调用IP核的工程，实现ALU功能
3. 扩展：加入符号位输出，以及可变宽度形式
4. 扩展：增加标志位输出
5. 实验原理

display IP核效果图 ALU输入输出形式 ALU效果图

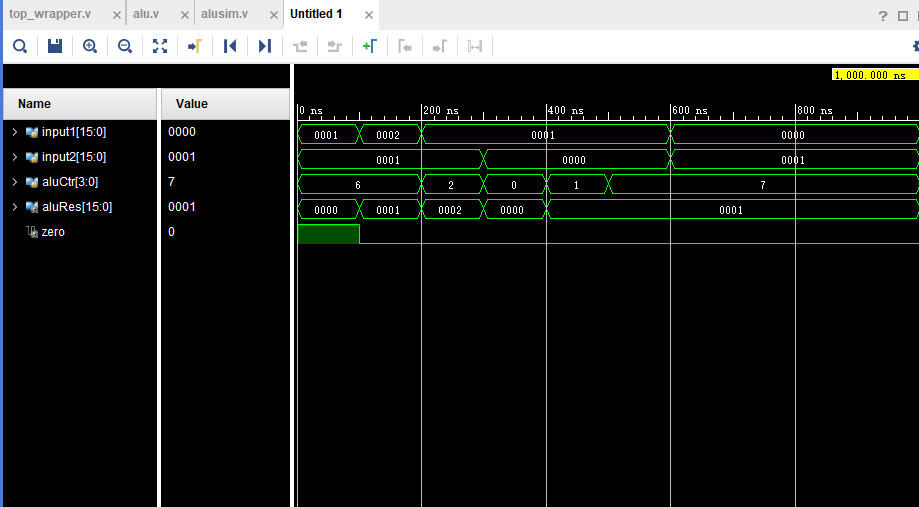


ALU功能图

|  |  |
| --- | --- |
| **0000** | **与** |
| **0001** | **或** |
| **1100** | **异或** |
| **0110** | **减** |
| **0010** | **加** |
| **0111** | **小于** |

**四、实验结果**

**1. ALU功能**

仿真波形

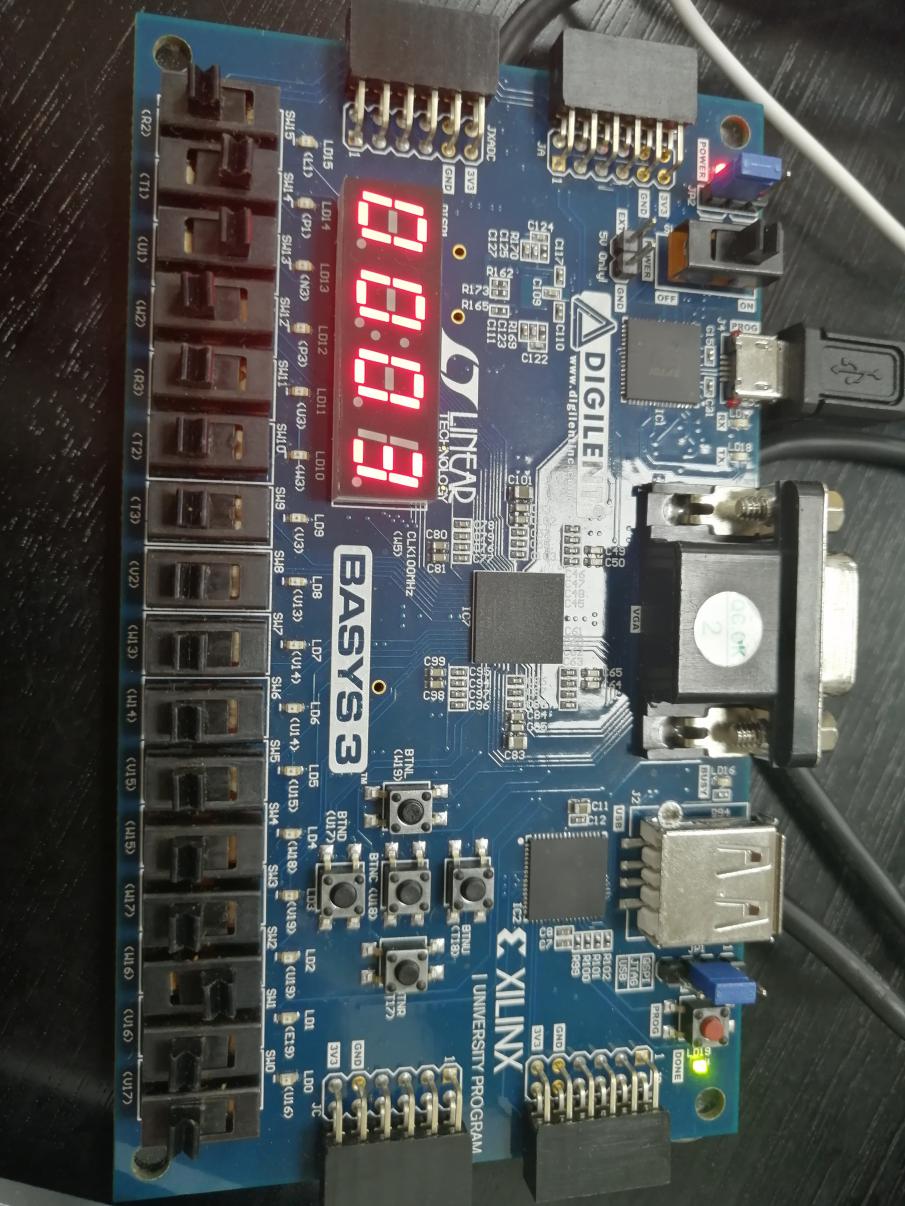
分析波形图

aluCtr = 6时 进行减法运算， 即0001 - 0001 = 0000； 0002 - 0001 = 0001

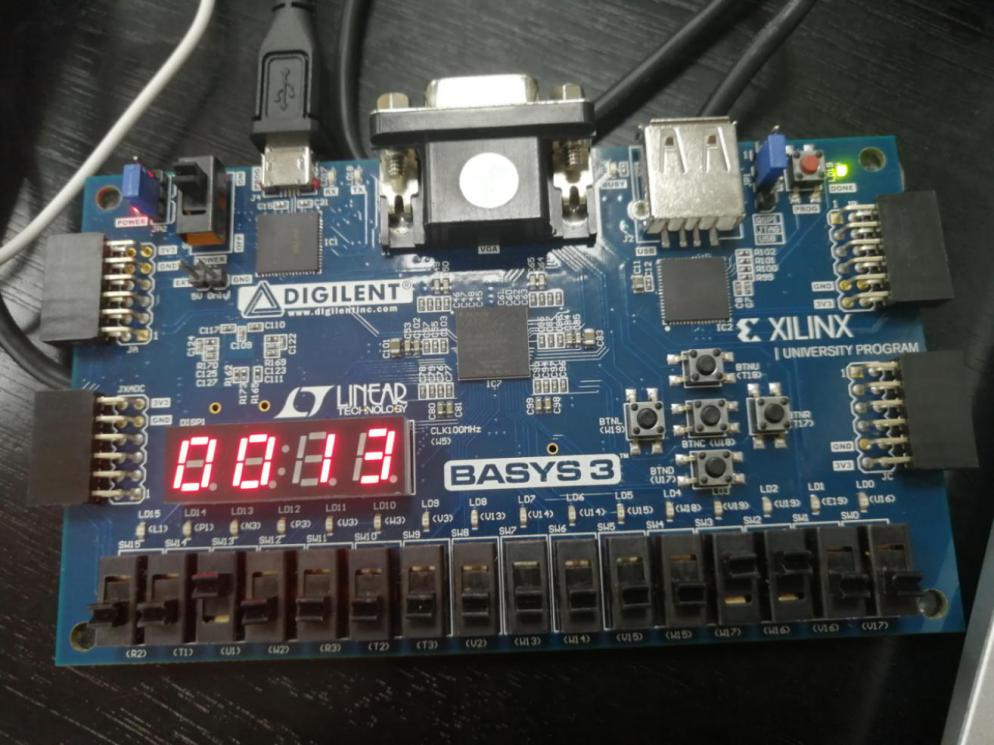
aluCtr = 2时 进行加法运算， 即0001 + 0001 = 0002

实物模拟

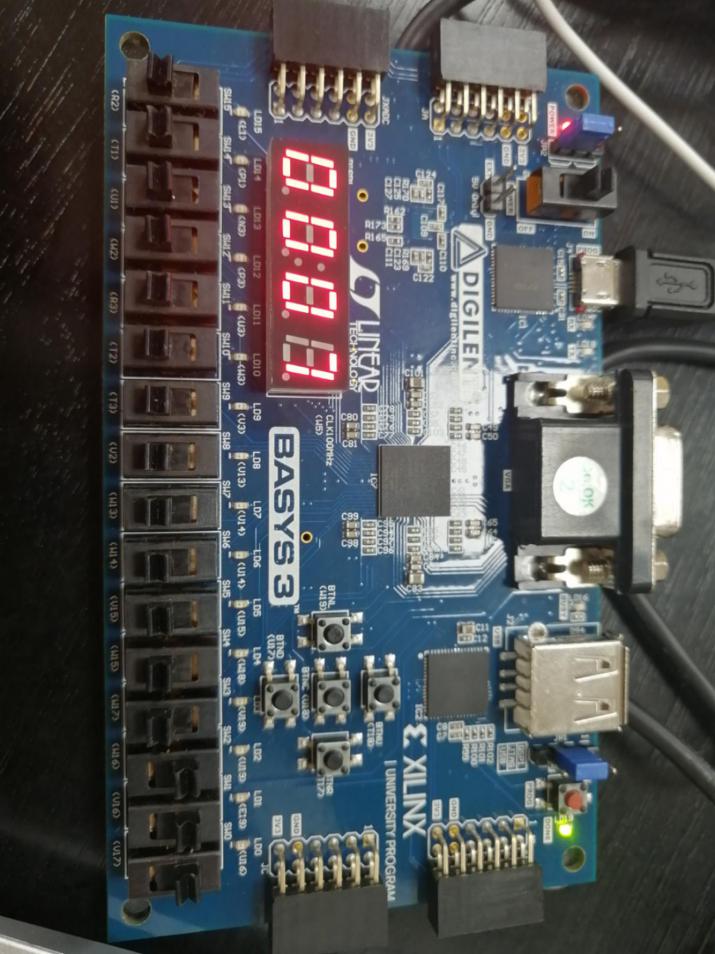
Input1初始设置为7



aluCtr = 0110 时，做减法运算， input2 = 4 ，最终结果为7 - 4 = 3



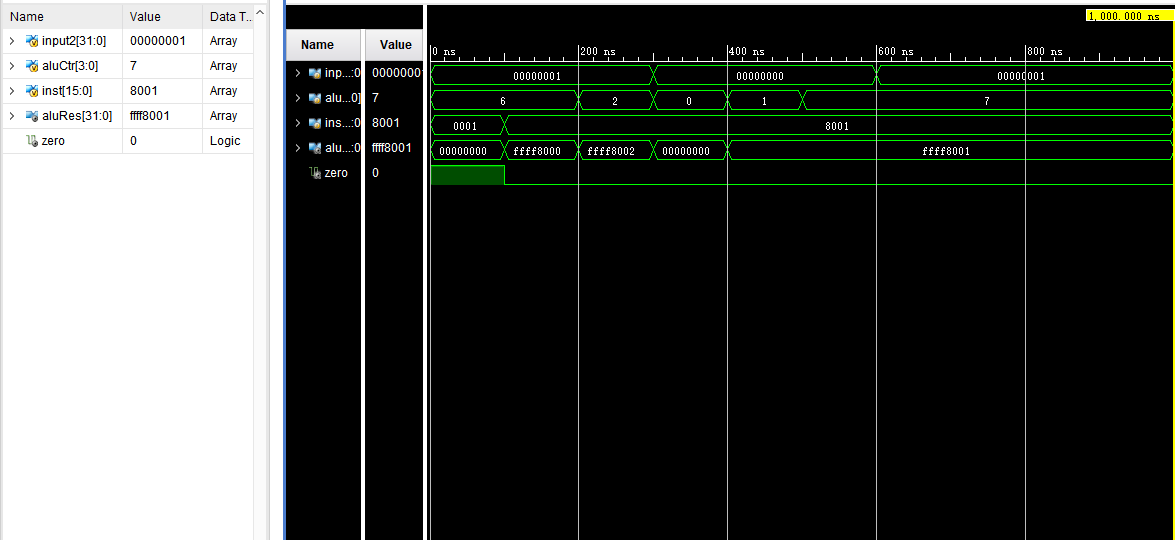
aluCtr = 0010时，做加法运算，input2 = 12， 结果为19， 十六进制显示13



AluCtr = 0000时，做与的运算，input2 为7，结果为7

扩展1 实现符号位扩展

仿真效果



扩展二 增加标志位输出

ZF, //0标志位, 运算结果为0(全零)则置1, 否则置0

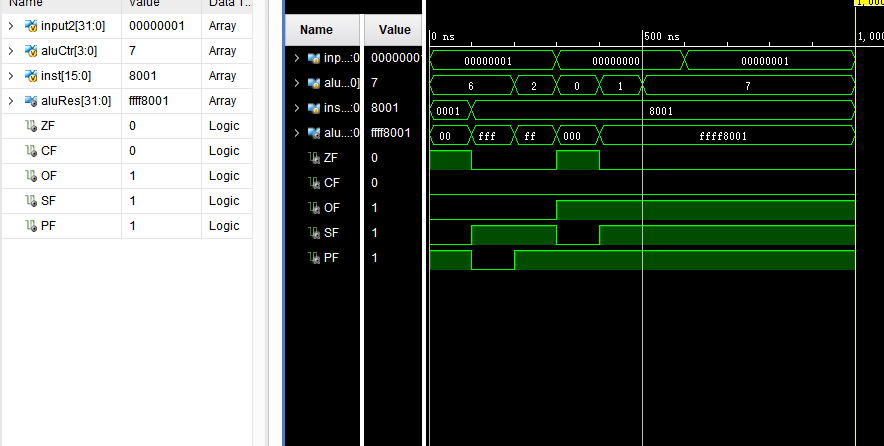
CF, //进借位标志位,

OF, //溢出标志位，对有符号数运算有意义，溢出则OF=1，否则为0

SF, //符号标志位，与F的最高位相同

PF; //奇偶标志位，F有奇数个1，则PF=1，否则为0

仿真如下



**五、实验感想**

通过这次实验，我掌握了数码管的构造原理，使用方法等；以及封装IP核的方法、调用IP核的相关操作，以及实现alu的基本功能，增添符号位扩展、标志位等功能。

做实验中遇到的问题：

① 调用IP核时，create wrapper 的是block design 之后的文件，生成的wrapper文件可以被当作头文件，做适当修改后应用。

② 仿真时，要把新增的仿真文件设置成头文件。

③ 实物模拟时，input1初始化的时候要在顶层文件中声明wire input input1，再做赋值，不然赋值会失败。

④ 生成约束文件时，所有引脚都需要fixed

⑤ 写标志位扩展时，要悉知标志位的计算方法

**附录（流程图，注释过的代码）：**

Ip核代码：module display(clk,data,sm\_wei,sm\_duan);

input clk;

input [15:0] data;

output [3:0] sm\_wei;

output [6:0] sm\_duan;

//分频

integer clk\_cnt;

reg clk\_400Hz;

always @(posedge clk)

if(clk\_cnt==32'd100000)

begin clk\_cnt <= 1'b0; clk\_400Hz <= ~clk\_400Hz;

end else clk\_cnt <= clk\_cnt + 1'b1;

//位控制

reg [3:0]wei\_ctrl=4'b1110;

always @(posedge clk\_400Hz)

wei\_ctrl <= {wei\_ctrl[2:0],wei\_ctrl[3]}; //位控制

reg [3:0]duan\_ctrl;

always @(wei\_ctrl or data)

case(wei\_ctrl)

4'b1110:duan\_ctrl=data[3:0];

4'b1101:duan\_ctrl=data[7:4];

4'b1011:duan\_ctrl=data[11:8];

4'b0111:duan\_ctrl=data[15:12];

default:duan\_ctrl=4'hf;

endcase

//解码模块

reg [6:0]duan;

always @(duan\_ctrl)

case(duan\_ctrl)

4'h0:duan=7'b100\_0000;//0

4'h1:duan=7'b111\_1001;//1

4'h2:duan=7'b010\_0100;//2

4'h3:duan=7'b011\_0000;//3

4'h4:duan=7'b001\_1001;//4

4'h5:duan=7'b001\_0010;//5

4'h6:duan=7'b000\_0010;//6

4'h7:duan=7'b111\_1000;//7

4'h8:duan=7'b000\_0000;//8

4'h9:duan=7'b001\_0000;//9

4'ha:duan=7'b000\_1000;//a

4'hb:duan=7'b000\_0011;//b

4'hc:duan=7'b100\_0110;//c

4'hd:duan=7'b010\_0001;//d

4'he:duan=7'b000\_0111;//e

4'hf:duan=7'b000\_1110;//f

// 4'hf:duan=7'b111\_1111;//不显示

default : duan = 7'b100\_0000;//0

endcase

//----------------------------------------------------------

assign sm\_wei = wei\_ctrl;

assign sm\_duan = duan;

endmodule

alu代码：

顶层文件：`timescale 1 ps / 1 ps

module top\_wrapper

(

input clk,

input reset, //复位信号（连接一个按键）

output [6:0] seg,//段码

output [3:0] sm\_wei,//哪个数码管

input [15:0] input2,

input [3:0] aluCtr

);

wire [15:0]input1;

//wire [15:0] input2;

assign input1 = 16'h0x7;

//assign input2[15:8] = {8{input2[7]}};

//wire aluCtr;

wire zero;

wire[15:0] aluRes;

alu alu(.input1(input1),

.input2(input2),

.aluCtr(aluCtr),

.zero(zero),

.aluRes(aluRes));

top top\_i

(.clk\_0(clk),

.data\_0(aluRes),

.sm\_duan\_0(seg),

.sm\_wei\_0(sm\_wei));

endmodule

//wire[15:0] expand;

// ALU控制信号线

// 实例化符号扩展模块

//signext signext(.inst(inst[15:0]), .data(expand));

//...............................实例化数码管显示模块

alu：module alu(

input [15:0] input1,

input [15:0] input2,

input [3:0] aluCtr,

output reg[15:0] aluRes,

output reg zero

);

always @(input1 or input2 or aluCtr) // 运算数或控制码变化时操作

begin

case(aluCtr)

4'b0110: // 减

begin

aluRes = input1 - input2;

if(aluRes == 0)

zero = 1;

else

zero = 0;

end

4'b0010: // 加

aluRes = input1 + input2;

4'b0000: // 与

aluRes = input1 & input2;

4'b0001: // 或

aluRes = input1 | input2;

4'b1100: // 异或

aluRes = ~(input1 | input2); //这里我认为应该是(~input1 & input2) | (input1 & ~input2)

4'b0111: // 小于设置

begin

if(input1<input2)

aluRes = 1;

end

default:

aluRes = 0;

endcase

end

endmodule

内容拓展1代码：

Verilog代码如下，细节见注释：

module signext(

input [15:0] inst, // 输入16位

output [31:0] data // 输出32位

);

// 根据符号补充符号位

// 如果符号位为1，则补充16个1，即16’h ffff

// 如果符号位为0，则补充16个0

assign data = inst[15:15]?{16'hffff,inst}:{16'h0000,inst};

endmodule

alu：

module alu(

input [15:0] inst,

input [31:0] input2,

input [3:0] aluCtr,

output reg[31:0] aluRes,

output reg zero

);

wire [31:0]input1;

always @(input1 or input2 or aluCtr)

// 运算数或控制码变化时操作

begin

case(aluCtr)

4'b0110: // 减

begin

aluRes = input1 - input2;

if(aluRes == 0)

zero = 1;

else

zero = 0;

end

4'b0010: // 加

aluRes = input1 + input2;

4'b0000: // 与

aluRes = input1 & input2;

4'b0001: // 或

aluRes = input1 | input2;

4'b1100: // 异或

aluRes = ~(input1 | input2);

4'b0111: // 小于设置

begin

if(input1<input2)

aluRes = 1;

end

default:

aluRes = 0;

endcase

end

signext signext(.inst(inst[15:0]), .data(input1));

endmodule

仿真：

module alusim;

// Inputs

reg [31:0] input2;

reg [3:0] aluCtr;

reg [15:0] inst; // 输入16 位

// Outputs

wire [31:0] aluRes;

wire zero;

// Instantiate the Unit Under Test (UUT)

alu uut (

.inst(inst),

.input2(input2),

.aluCtr(aluCtr),

.aluRes(aluRes),

.zero(zero)

);

initial begin

// Initialize Inputs

input2 = 1;

aluCtr = 4'b0110;

inst=16'b0000\_0000\_0000\_0001; //符号扩展的输入

#100;

input2 = 1;

aluCtr = 4'b0110;

inst=16'b1000\_0000\_0000\_0001;

#100

input2 = 1;

aluCtr = 4'b0010;

#100

input2 = 0;

aluCtr = 4'b0000;

#100

input2 = 0;

aluCtr = 4'b0001;

#100

input2 = 0;

aluCtr = 4'b0111;

#100

input2 = 1;

aluCtr = 4'b0111;

end

endmodule

内容拓展2代码：

alu修改：

module alu(

input [15:0] inst,

input [31:0] input2,

input [3:0] aluCtr,

output reg[31:0] aluRes,

output reg

ZF,CF,OF,SF,PF

);

wire [31:0]input1;

always @(input1 or input2 or aluCtr)

// 运算数或控制码变化时操作

begin

case(aluCtr)

4'b0110: // 减

begin

{CF,aluRes}= input1 - input2;

if(aluRes == 0)

ZF = 1;

else

ZF = 0;//全为0，则ZF=1

OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式

SF = aluRes[31];//符号标志,取F的最高位

PF = ~^ aluRes;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0

end

4'b0010: // 加

begin

{CF,aluRes} = input1 + input2;

if(aluRes == 0)

ZF = 1;

else

ZF = 0;//全为0，则ZF=1

OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式

SF = aluRes[31];//符号标志,取F的最高位

PF = ~^ aluRes;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0

end

4'b0000: // 与

begin

aluRes = input1 & input2;

if(aluRes == 0)

ZF = 1;

else

ZF = 0; //全为0，则ZF=1

OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式

SF = aluRes[31];//符号标志,取F的最高位

PF = ~^ aluRes;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0

end

4'b0001: // 或

begin

aluRes = input1 | input2;

if(aluRes == 0)

ZF = 1;

else

ZF = 0;//全为0，则ZF=1

OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式

SF = aluRes[31];//符号标志,取F的最高位

PF = ~^ aluRes;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0

end

4'b1100: // 异或

begin

aluRes = ~(input1 | input2);

if(aluRes == 0)

ZF = 1;

else

ZF = 0;//全为0，则ZF=1

OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式

SF = aluRes[31];//符号标志,取F的最高位

PF = ~^ aluRes;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0

end

4'b0111: // 小于设置

begin

if(input1<input2)

aluRes = 1;

ZF = 0;//全为0，则ZF=1

OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式

SF = aluRes[31];//符号标志,取F的最高位

PF = ~^ aluRes;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0

end

default:

begin

aluRes = 0;

ZF = 0;//全为0，则ZF=1

OF = input1[31]^input2[31]^aluRes[31]^aluCtr;//溢出标志公式

SF = aluRes[31];//符号标志,取F的最高位

PF = ~^ aluRes;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0

end

endcase

end

signext signext(.inst(inst[15:0]), .data(input1));

endmodule

仿真：  
module alusim;

// Inputs

reg [31:0] input2;

reg [3:0] aluCtr;

reg [15:0] inst; // 输入16 位

// Outputs

wire [31:0] aluRes;

wire ZF;

wire CF;

wire OF;

wire SF;

wire PF;

// Instantiate the Unit Under Test (UUT)

alu uut (

.inst(inst),

.input2(input2),

.aluCtr(aluCtr),

.aluRes(aluRes),

.ZF(ZF),

.CF(CF),

.OF(OF),

.SF(SF),

.PF(PF)

);

initial begin

// Initialize Inputs

input2 = 1;

aluCtr = 4'b0110;

inst=16'b0000\_0000\_0000\_0001; //符号扩展的输入

#100;

input2 = 1;

aluCtr = 4'b0110;

inst=16'b1000\_0000\_0000\_0001;

#100

input2 = 1;

aluCtr = 4'b0010;

#100

input2 = 0;

aluCtr = 4'b0000;

#100

input2 = 0;

aluCtr = 4'b0001;

#100

input2 = 0;

aluCtr = 4'b0111;

#100

input2 = 1;

aluCtr = 4'b0111;

end

endmodule