**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：汪庭葳、刘洋旗

|  |  |  |  |
| --- | --- | --- | --- |
| 年级&班级 | **2018级1班** | 专业(方向) | **计算机类** |
| 学号 | **18308045** | 姓名 | **谷正阳** |
| 电话 | **13355426001** | Email | [**Guzy0324@163.com**](mailto:Guzy0324@163.com) |
| 开始日期 | **2019.11.6** | 完成日期 | **2019.11.6** |

**一、实验题目**

计算机结构与组成 实验5

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

**一.创建display IP 核**

1.创建项目，首先打开vivado，在vivado左上侧菜单栏点击File菜单，选择New Project，

点击Next，设置项目名称和相应的路径，一直点击Next到Add Source界面，添加display模块的源代码文件。（xc7a35tcpg236-1）

参考源文件代码：

//数码管 ip 核

//数码管 ip 核

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]}; //段控制 reg [3:0]duan\_ctrl;

always @(wei\_ctrl)

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 [7:0]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

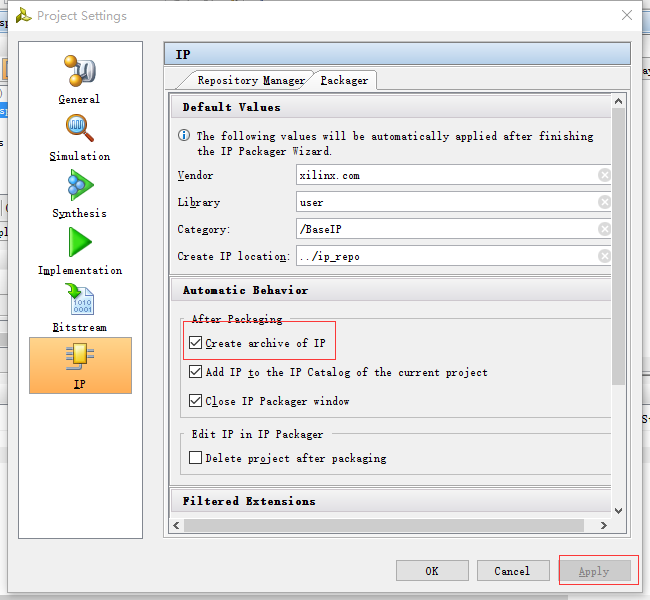
//----------------------------------------------------------

assign sm\_wei = wei\_ctrl;

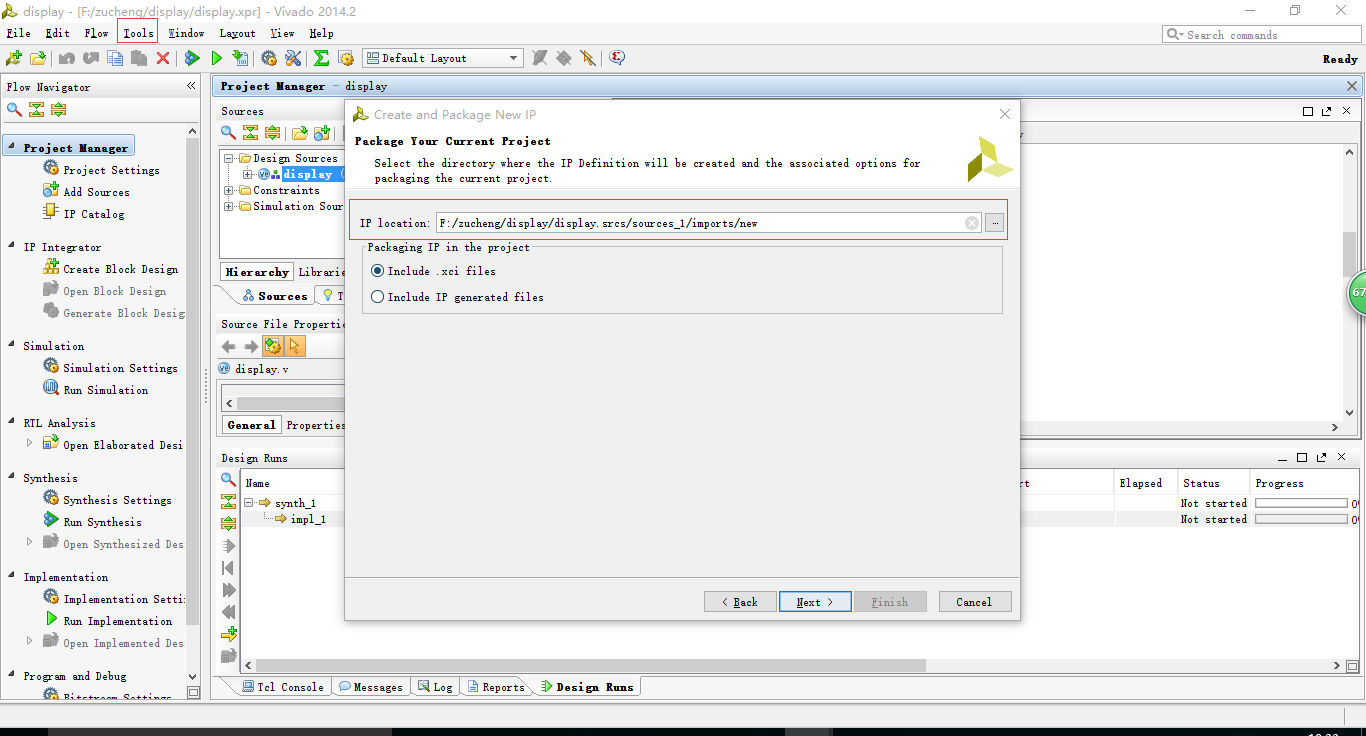
assign sm\_duan = duan;

endmodule

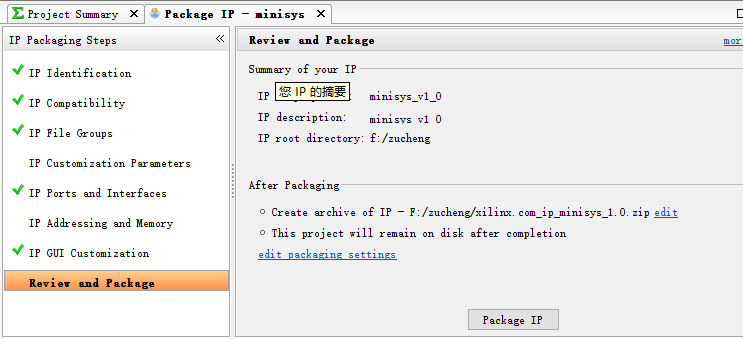
2.将display模块代码封装成ip核。点击Project Settings菜单，弹出对话框如下图，按下图方式设置后点击Apply，然后点击OK。其中Category为IP核所在的上级目录，当添加IP核时需要在该目录下找到相应的IP核。



设置成功后，点击Tools，选择Create and Package IP,一直点击next按键，选择IP核保存的路径，最后生成IP核。



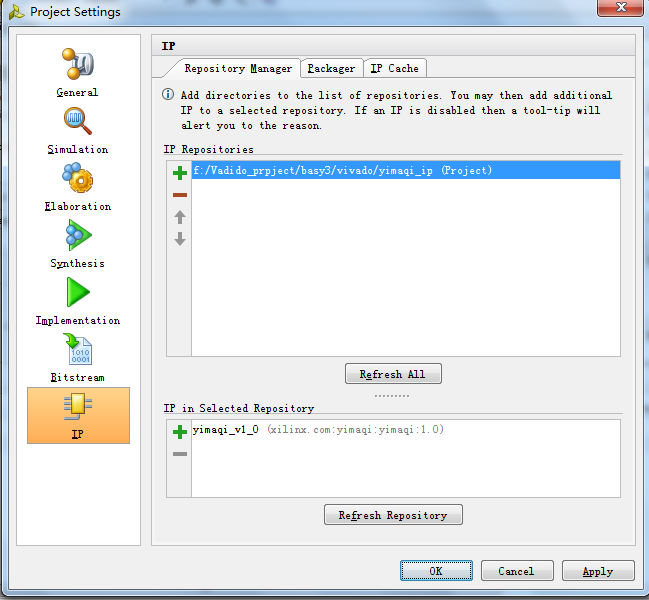
生成后会弹出该IP核的详细情况页面，点击Review and Package选项，然后点击Package IP，IP核就封装完成。



ip核的例化，在source下面有一个IP Source选项卡点击进去，在点击Instantiation Template里的.veo文件就是verilog例化文件了,默认名称disply\_0.veo.

创建调用ip核的工程

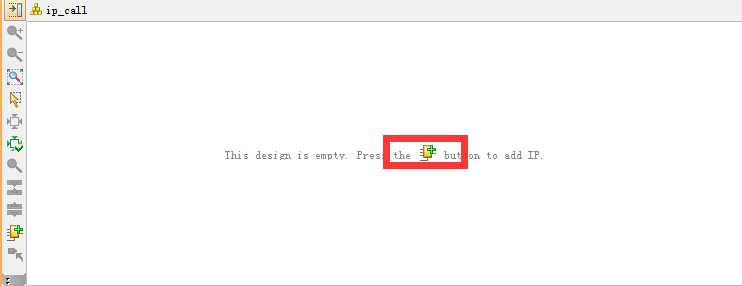
新建一个工程，则点击project setting->IP,在该界面将之前设计的ip核所在工程路径添加到“ip repositories”中，点击ok。



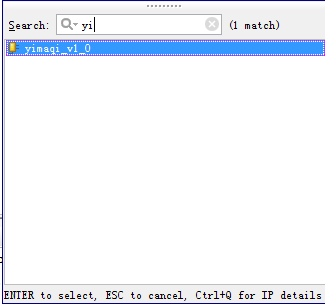
创建以及ip核的系统：

在ip integrator选项中，点击create block design，创建之后出现以下界面,

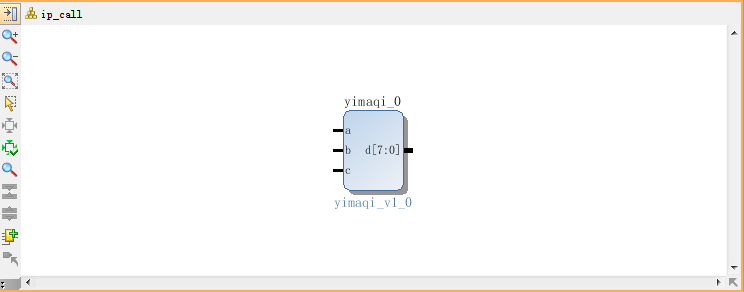
点击红框内的add ip



↓↓↓

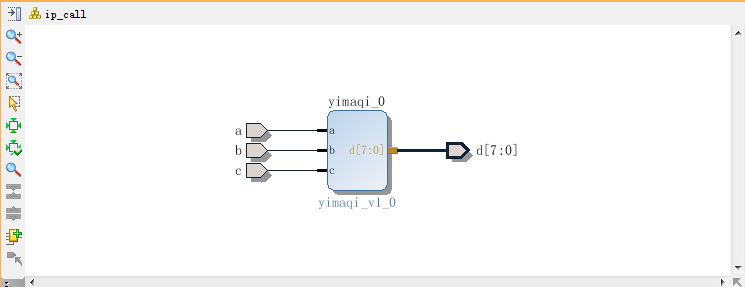
出现下面界面，在搜索一栏输入之前设计的ip核显示名字，双击，将ip核添加进去

↓↓↓



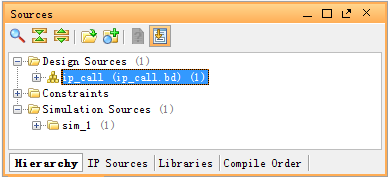
↓↓↓

接下来将引脚引出，右击a-> make external,则将a引出，其他端口类似。



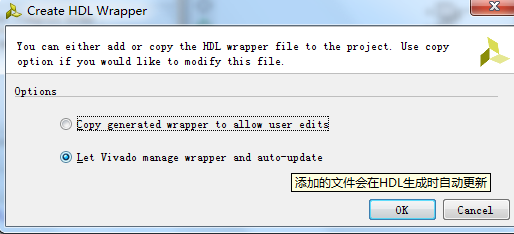
↓↓↓

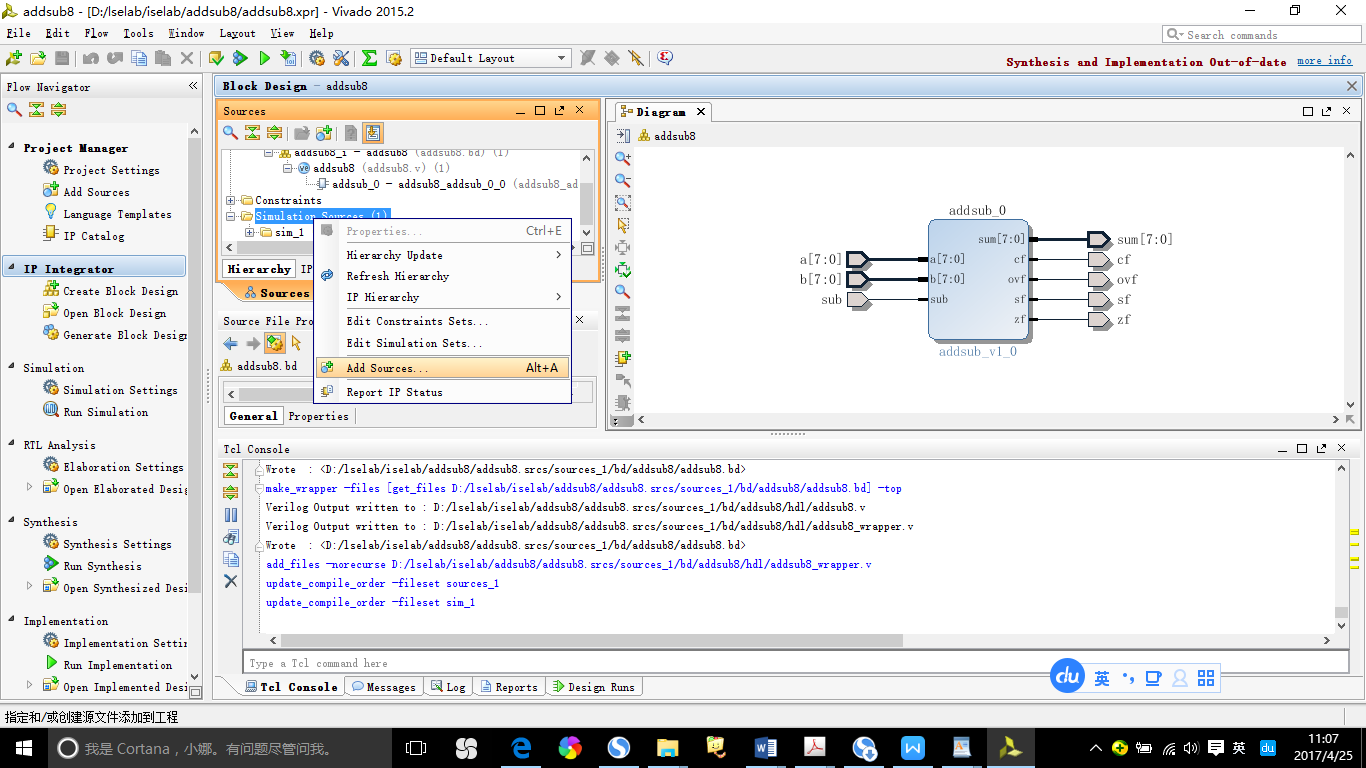
在source窗口下，右击->选择create HDL wrapper,



↓↓↓

点击ok，则生成文件内调用了译码器ip核





**二.创建ALU 模块、top模块Verilog HDL文件和仿真文件**

**（1）创建Verilog HDL源设计文件，如下图**

Verilog代码如下，细节见注释：

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);

4'b0111: // 小于设置

begin

if(input1<input2)

aluRes = 1;

end

default:

aluRes = 0;

endcase

end

endmodule

**（2）添加顶层设计文件**

**顶层文件**

module top(

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

);

// ALU信号线

wire zero;

wire[15:0] aluRes;

//wire[15:0] expand;

// ALU控制信号线

wire[3:0] aluCtr;

// 实例化ALU模块

alu alu(.input1(input1),

.input2(input2),

.aluCtr(aluCtr),

.zero(zero),

.aluRes(aluRes));

// 。。。。。。。。。。。。。。。。。。。。。。。。。。实例化符号扩展模块

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

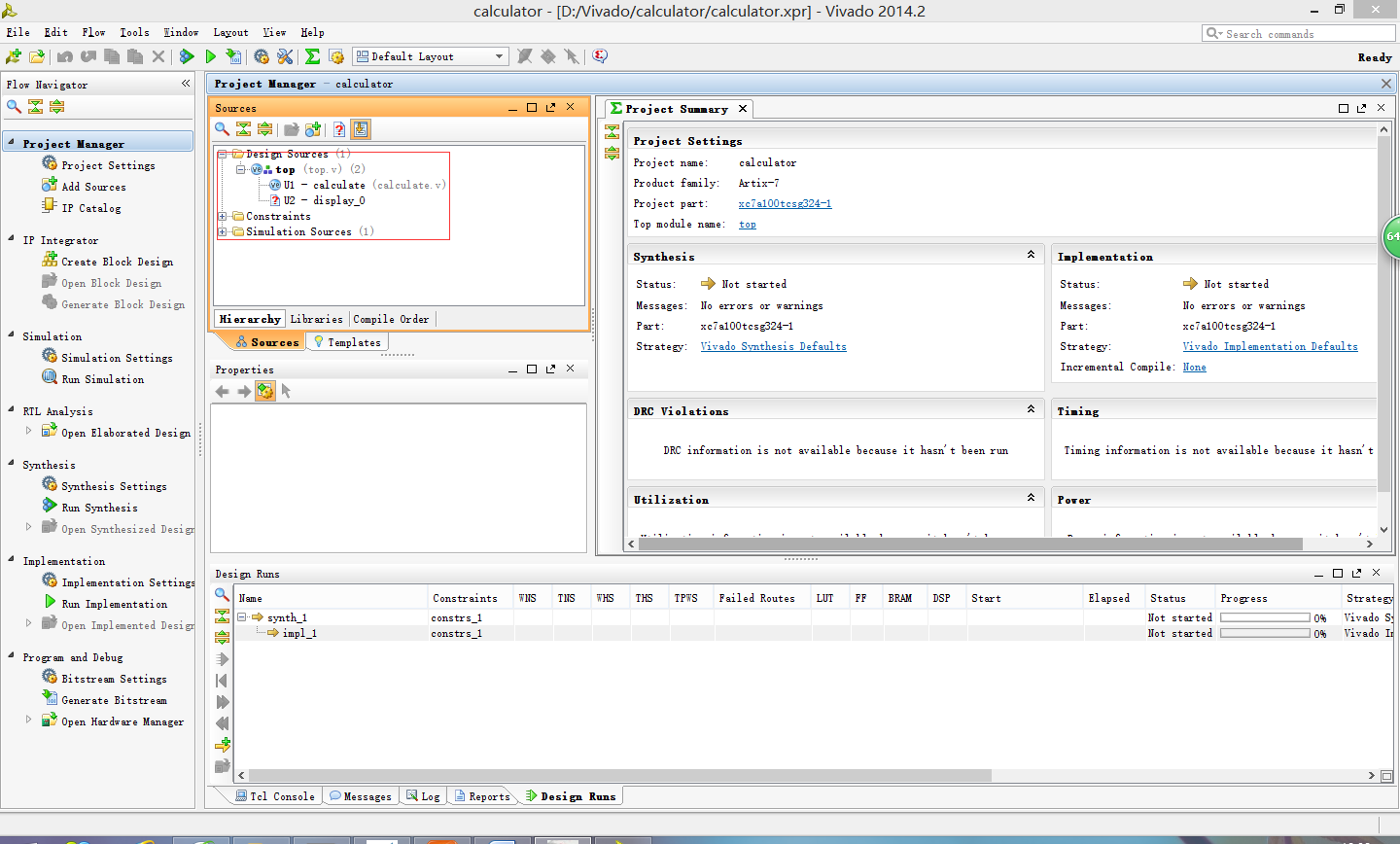
//...............................实例化数码管显示模块

display disp1 (.clk(clk),.sm\_wei(sm\_wei),.data(aluRes[15:0]),.sm\_duan(seg));

);

endmodule

创建完成后，可以看到display模块的代码缺失，如下图所示：



添加display模块的IP核。点击Project Settings然后点击IP，点击Add Repository将生成的display模块的IP核所在的目录添加到其中，默认的情况下，display模块的IP核在该模块项目的文件夹下，只需要将display模块的项目添加到其中就能找到相应的IP核，然后点击OK。

如果你已经在里工程添加好三个设计源文件，那么你可以一步步点击‘Run Synthesis’>=‘Implementation’>=‘Generate Bitstream’:

**(3)添加仿真文件**

仿真代码

module alusim;

// Inputs

reg [15:0] input1;

reg [15:0] input2;

reg [3:0] aluCtr;

// Outputs

wire [15:0] aluRes;

wire zero;

// Instantiate the Unit Under Test (UUT)

alu uut (

.input1(input1),

.input2(input2),

.aluCtr(aluCtr),

.aluRes(aluRes),

.zero(zero)

);

initial begin

// Initialize Inputs

input1 = 1;

input2 = 1;

aluCtr = 4'b0110;

#100;

input1 = 2;

input2 = 1;

aluCtr = 4'b0110;

#100

input1 = 1;

input2 = 1;

aluCtr = 4'b0010;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0000;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0001;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0111;

#100

input1 = 0;

input2 = 1;

aluCtr = 4'b0111;

end

endmodule

编译综合无错后观察仿真波形。

**三、进一步扩展**：

1. 把程序改成可变宽度形式，并加入标志位输出，比如可参照上一次实验。
2. **FPGA实现， 创建约束文件**

1）把数据宽度设定为16位，把input1操作数设置为常数。运算结果通过数码管输出。比如在文件中加入:

assign input1=0x7;

2）连接引脚：用SW模拟输入运算数据（比如input2）以及aluCtr操作选择，sw0~sw7作为16位input2的低8位输入，input2的高8位全部接sw8。

连接数码管an[0]-an[3]段码选择,段码seg[0]-seg[6].

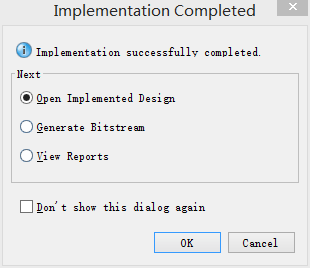
3）因为运算器支持“加、减、与、或、非”5种运算，需要3位（8个操作）。将sw15~sw13输入到aluCtr作为运算器的控制信号；

4）将计算16位结果显示到数码管显示器上，显示器由4个同阳极7段数码管组成，显示器显示的是十六进制。

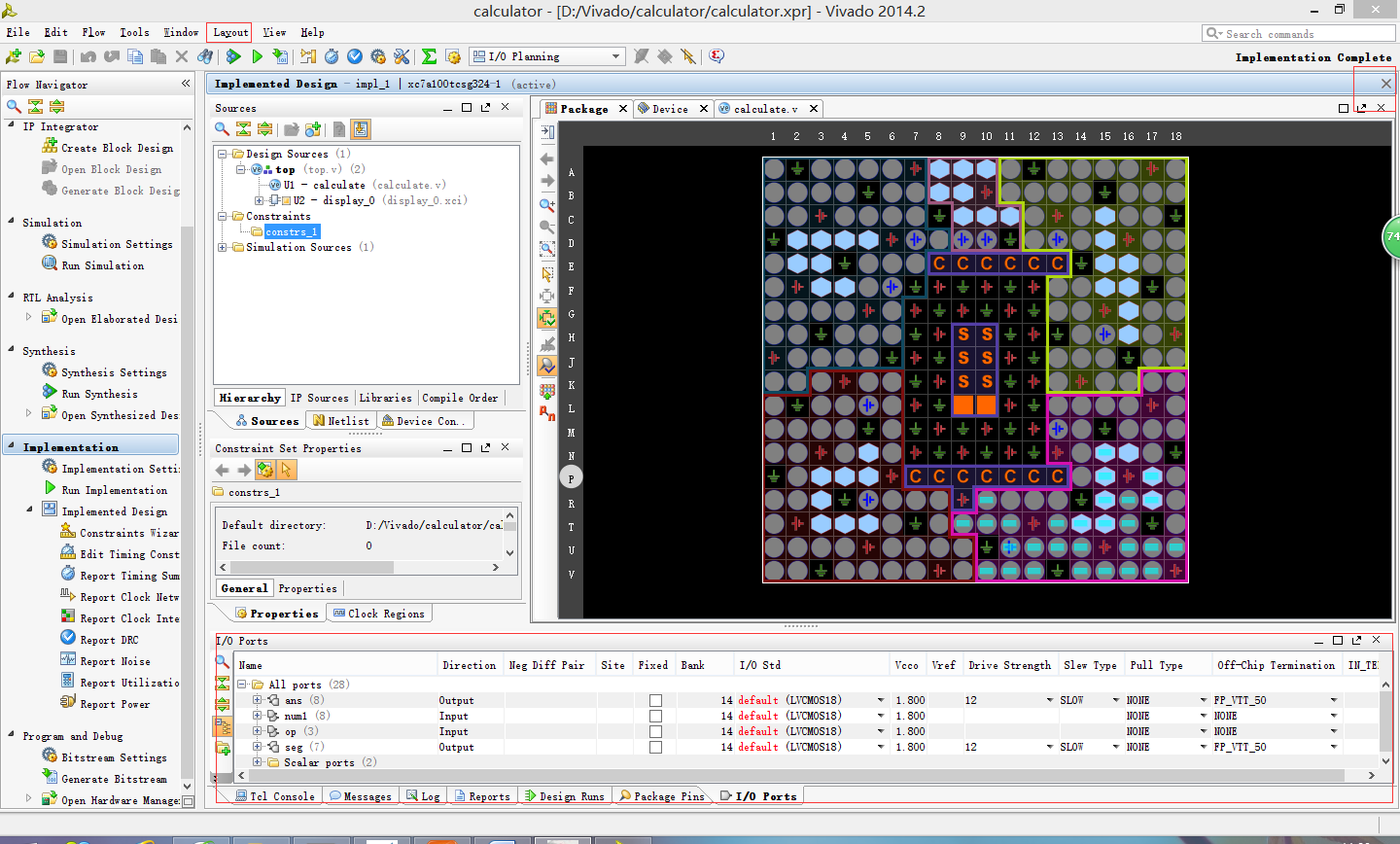
在实现之前我们需要指定FPGA使用过的SW和LED的引脚。

看Basys3 的原理图去选择FPGA的管脚。

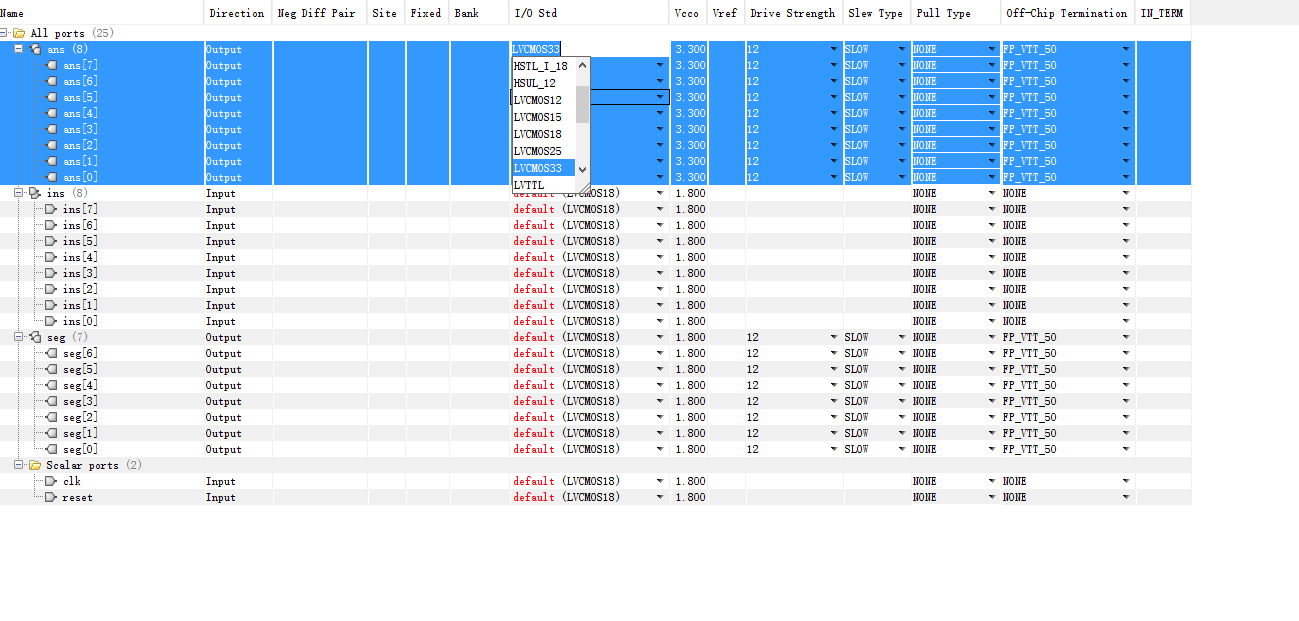
点击Run Synthesis对项目源文件进行综合，检查项目是否有语法错误。无误后弹出框的Run Implementation然后点击OK，或者直接点击Cancel，然后选择左侧菜单栏Implementation菜单栏下的Run Implementation选项进行实施项目活动的设计合成网表，完成后会有如下图所示的提示：



点击OK，选择Layout菜单下的I/O Planning进行I/O端口的管脚绑定。

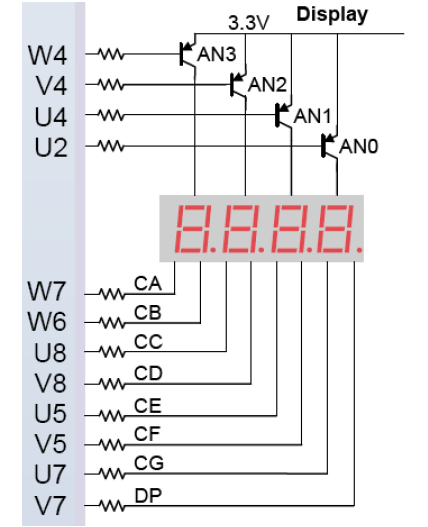


首先将所有的端口的电压改为3.3V（在IO std下选择LVCMOS33）。



其中clk所要绑定的管脚为w5，reset所要绑定的管脚为U18。绑定完成后需要关闭该Implemented Design界面，关闭时会弹出提示框询问是否保存已绑定的管件生成的约束文件，选择保存，该步骤完成。

5.点击Generate Bitstream，生成相应的比特流文件，生成成功后，将开发板连接到电脑USB接口，打开开关，然后点击Open target，选择开发板后，点击Program device，然后将比特流文件下载到开发板，根据不同的输入观察显示的内容。



数码管约束参考：系统默认主频率为100MHz

set\_property PACKAGE\_PIN U2 [get\_ports {an[0]}]

set\_property PACKAGE\_PIN U4 [get\_ports {an[1]}]

set\_property PACKAGE\_PIN V4 [get\_ports {an[2]}]

set\_property PACKAGE\_PIN W4 [get\_ports {an[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[0]}]

set\_property PACKAGE\_PIN U7 [get\_ports {seg[6]}]

set\_property PACKAGE\_PIN V5 [get\_ports {seg[5]}]

set\_property PACKAGE\_PIN U5 [get\_ports {seg[4]}]

set\_property PACKAGE\_PIN V8 [get\_ports {seg[3]}]

set\_property PACKAGE\_PIN U8 [get\_ports {seg[2]}]

set\_property PACKAGE\_PIN W6 [get\_ports {seg[1]}]

set\_property PACKAGE\_PIN W7 [get\_ports {seg[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[0]}]

set\_property PACKAGE\_PIN W5 [get\_ports clk]

set\_property PACKAGE\_PIN U18 [get\_ports reset]

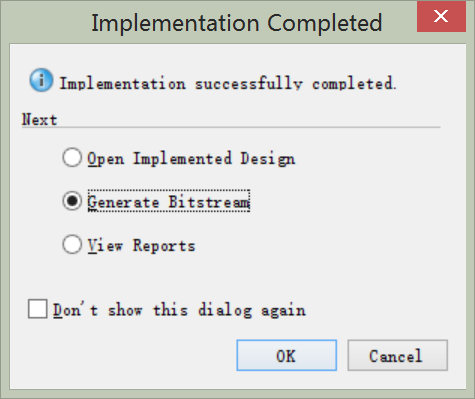
set\_property IOSTANDARD LVCMOS33 [get\_ports reset]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

**四. RTL分析，观察综合后的原理图.**

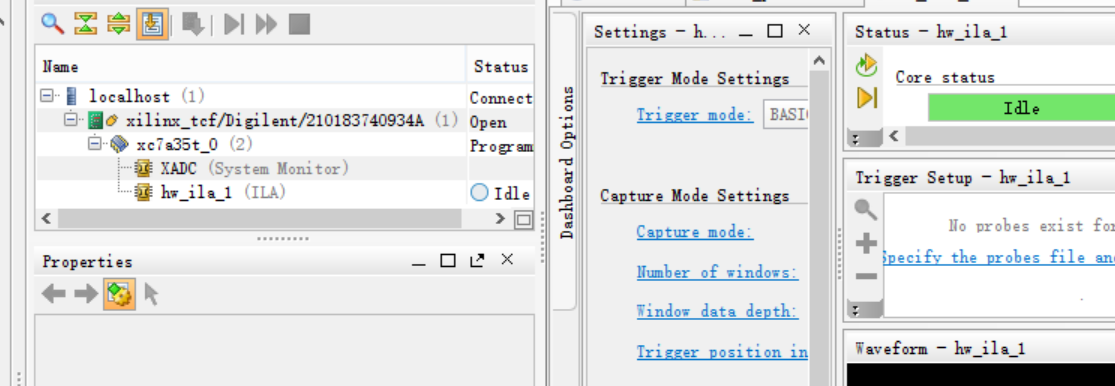
**五. 生成比特流**

 点击生成比特流，见下图：



**六. 下载程序**

 （1）与硬件相连



 （2）下载程序

**七. 实物演示**

比如设置 aluCtr 为’b0010，第二个操作数高 8 位为’b0000\_0000，低 8 位为8’b00000001结果如下图：对应计算为：1+0x07=-0x08

alures=8



比如设置 aluCtr 为’b0110 ，第二个操作数低 8 位为8’b00000101 alures=7-5=2



**八、选作：32位运算参考代码设计input1为16位，运算中加入符号扩展，扩展为32位参与运算，并进行仿真。**

符号扩展模块

将16位有符号数扩展成32位有符号数，只需要在16位数前面补足符号即可。



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

**Basys3开发板**

Basys3开发板的通用I/O设备电路图如图1所示：

**1** 电源电路

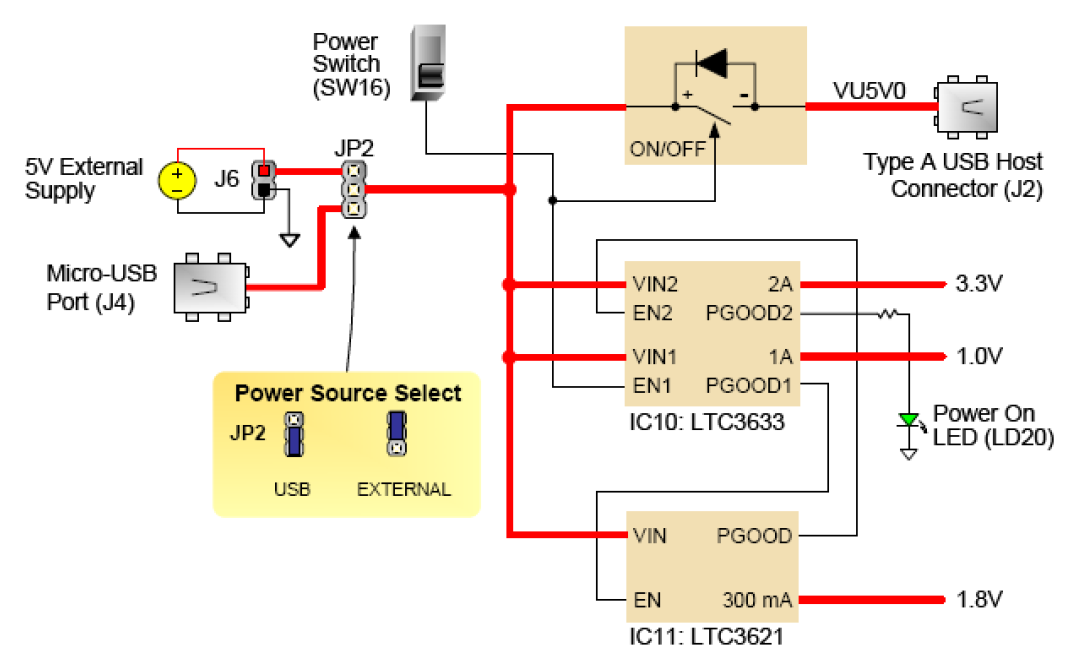
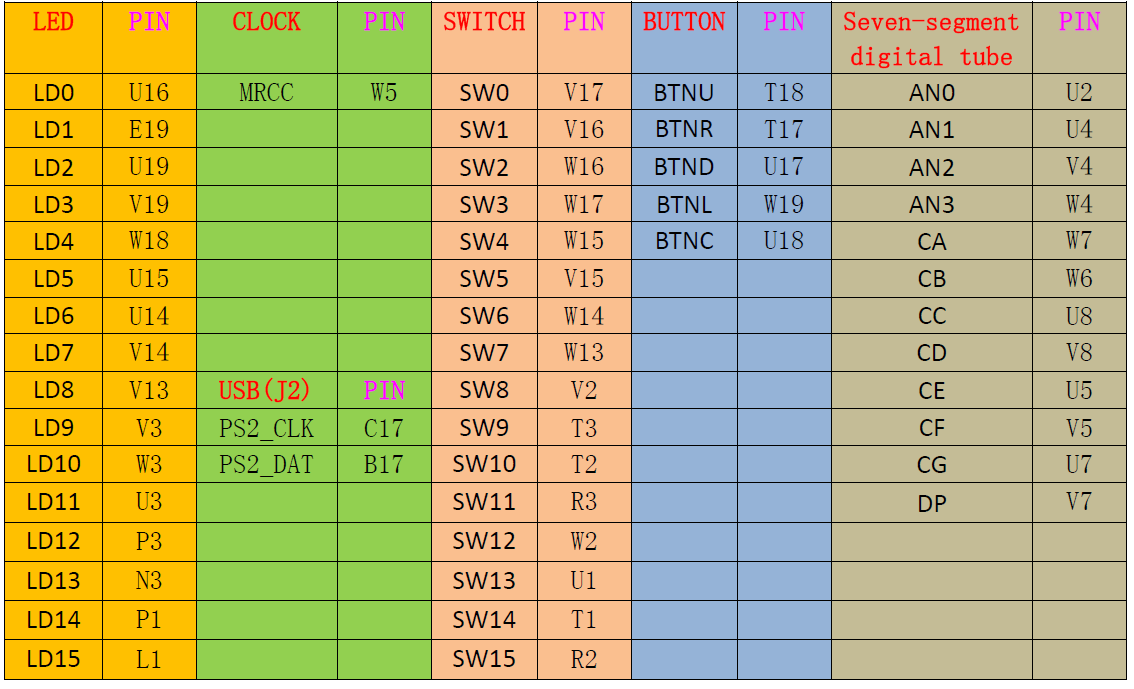


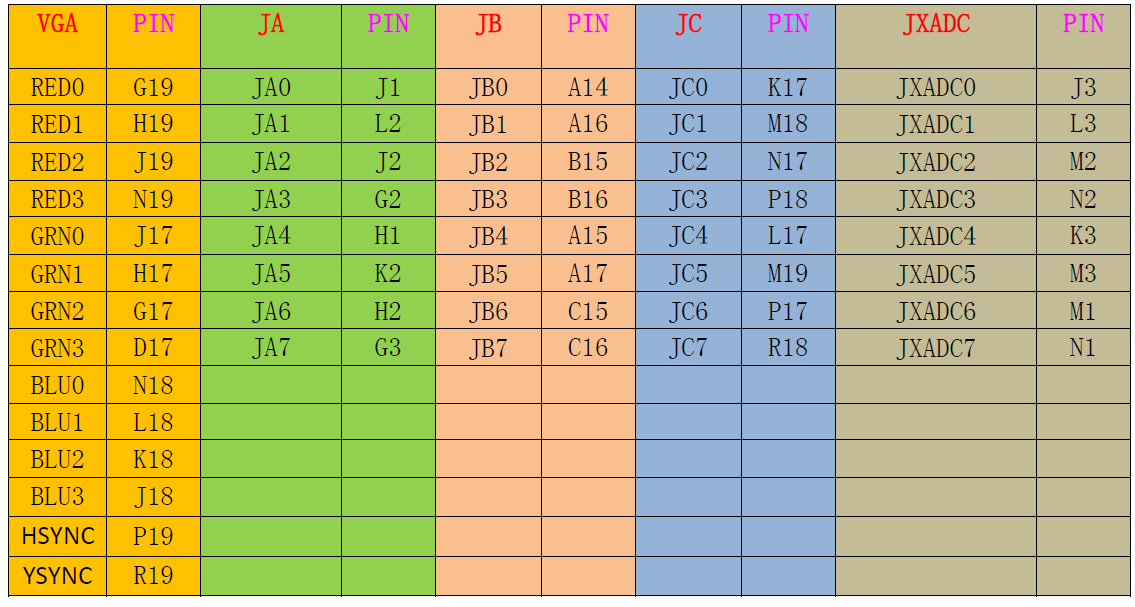
图1 USB接口电路

|  |  |
| --- | --- |
| **LED**灯电路 | 拨码开关电路 |
| 按键电路 | 数码管电路 |

1. 系统默认主频率为100MHz

引脚分配表格：





**实验背景知识**

1. **LED 数码管基础知识**

****

数码管原理图

在数码管上显示数字就是将相应的段位点亮组成要显示的数字，共阳数码管的码值表如下所示，‘1’代表相应的管脚输出高电平，不点亮相应段位，‘0’代表相应的管脚输出低电平，点亮相应段位。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 显示 | a | b | c | d | e | f | g | dp(h) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

段码：gfedcba

* parameter zero = 7'b100\_0000;
* parameter one = 7'b111\_1001;
* parameter two = 7'b010\_0100;
* parameter three = 7'b011\_0000;
* parameter four = 7'b001\_1001;
* parameter five = 7'b001\_0010;
* parameter six = 7'b000\_0010;
* parameter seven = 7'b111\_1000;
* parameter eight = 7'b000\_0000;
* parameter nine = 7'b001\_0000;

**2．动态数码管原理**

数码管显示部分的电路如图所示。我们使用的是一个四位带小数点的七段共阳极数码管，当我们相应的输出脚为低电平时，该段位的LED点亮。位选位也是低电平选通。

****

动态数码管显示的原理是：每次选通其中一位，送出这位要显示的内容，然后一段时间后选通下一位送出对应数据，4 个数码管这样依次选通并送出相应的数据，结束后再重复进行。这样只要选通时间选取的合适，由于人眼的视觉暂留，数码管看起来就是连续显示的。

**2.** 实验原理

本次实验的原理图如图所示：



|  |  |
| --- | --- |
|  |  |

运算器实验原理图

ALU主要执行5种操作：与，或，加，减，小于设置。这五种操作可以使用四位的编码表示：0000，0001，0010，1110，0111。指令不同，则对应的ALU运算不同，所以该模块需要根据指令来控制ALU进行正确的运算。

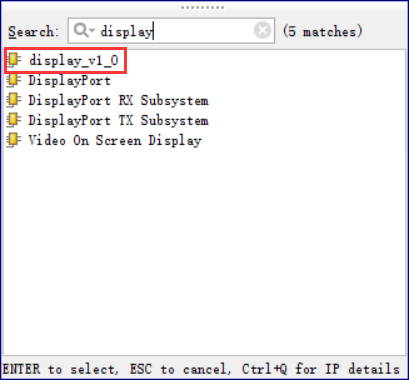
根据aluCtr的不同进行相应的运算操作，将操作结果赋值到输出变量。具体的运算规则如下表所示：

表1运算规则

|  |  |
| --- | --- |
| 操作码aluCtr | 运算结果 |
| 0000 | And 与 |
| 0001 | Or 或 |
| 0010 | Add 加 |
| 1110 | Sub 减 |
| 0111 | slt小于设置 |
| 0011 | not |
| 0100 | xor |
| 其他 | 32'h00000000; |

**四、实验结果**

实验1：ip核

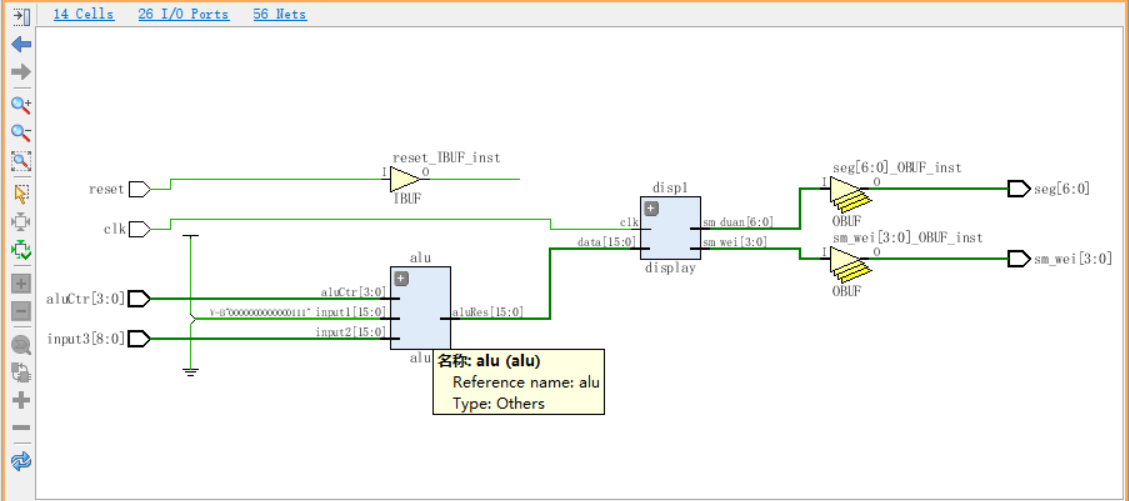


实验2：仿真

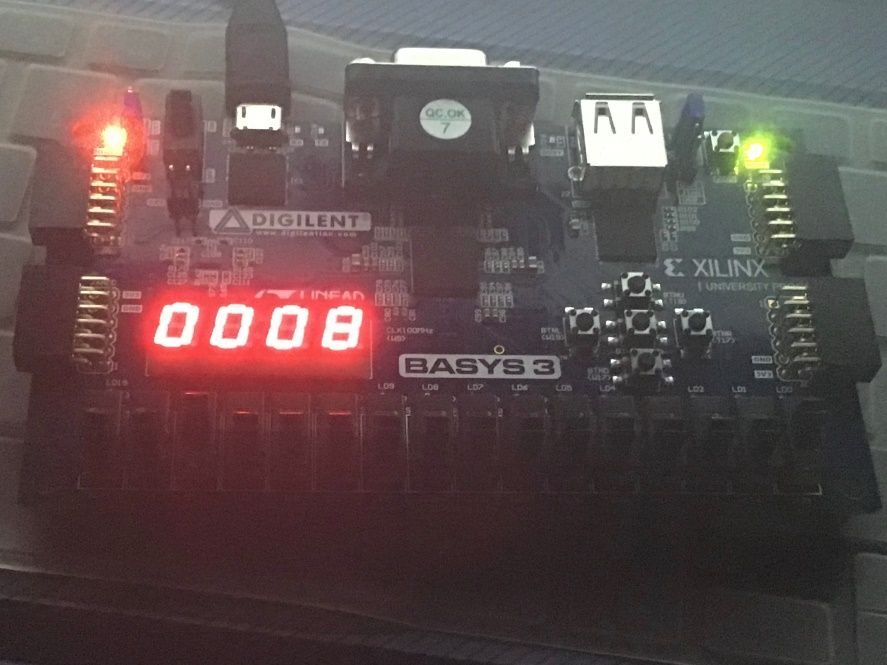


其中0x1 – 0x1 = 0x0，0x2 – 0x1 = 0x1，0x1 + 0x1 = 0x2，0x1 & 0x0 = 0x0，0x1 | 0x0 = 0x1，0x1 < 0x0 = 0x0，0x0 < 0x1 = 0x1

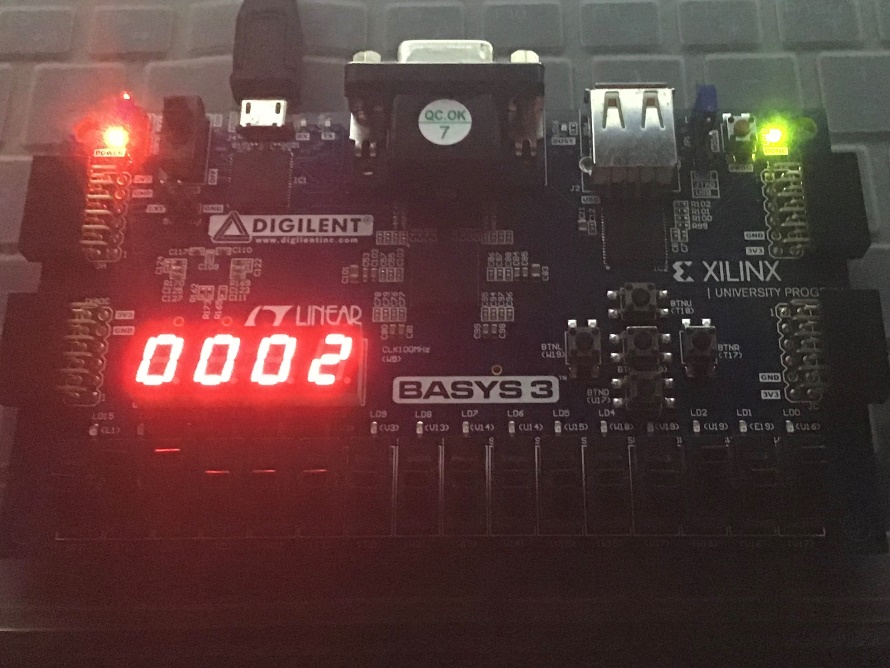
实验3：RTL分析



实验4：烧板

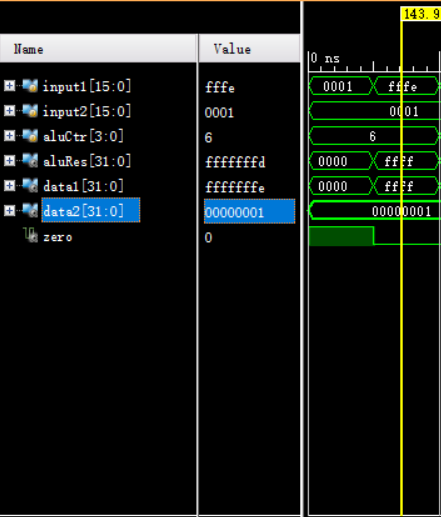


0x1 + 0x7 = 0x8



0x7 - 0x5 = 0x2

实验5：选作对input1，input2 sign-extend然后32位运算



其中0xfffe扩展变为0xfffffffe，0x1扩展仍为0x1，0xfffffffe - 0x1 = 0xfffffffd

**五、实验感想**

ip核类似库函数，可以更方便的调用设计。使用verilog语言实现alu并不困难。

**附录（流程图，注释过的代码）：**

**实验1，2，3，4：**

**top：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/06 15:21:51

// Design Name:

// Module Name: top

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module top(

input clk,

input reset, //复位信号（连接一个按键）

input [3:0] aluCtr,

//input [15:0] input1,

//input [15:0] input2,

input [8:0] input3,

output [6:0] seg,//段码

output [3:0] sm\_wei//哪个数码管

);

// ALU信号线

wire zero;

wire[15:0] aluRes;

//wire[15:0] expand;

// ALU控制信号线

wire [15:0] input1;

wire [15:0] input2;

// 实例化ALU模块

assign input1 = 16'h0x7;

assign input2[15:9] = {7{input3[8]}};

assign input2[8:0] = input3[8:0];

alu alu(.input1(input1),

.input2(input2),

.aluCtr(aluCtr),

.zero(zero),

.aluRes(aluRes));

// 。。。。。。。。。。。。。。。。。。。。。。。。。。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

display disp1 (.clk(clk),.sm\_wei(sm\_wei),.data(aluRes[15:0]),.sm\_duan(seg));

endmodule

**alu：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/06 14:56:40

// Design Name:

// Module Name: alu

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

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;

4'b0111: // 小于设置

begin

if(input1<input2)

aluRes = 1;

else

aluRes = 0;

end

default:

aluRes = 0;

endcase

end

endmodule

**display：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/06 15:23:12

// Design Name:

// Module Name: display

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

//数码管 ip 核

//数码管 ip 核

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]};

//段控制

reg [3:0]duan\_ctrl;

always @(wei\_ctrl, 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 [7:0]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

**constraint：**

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

set\_property PACKAGE\_PIN W5 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports reset]

set\_property PACKAGE\_PIN U18 [get\_ports reset]

set\_property IOSTANDARD LVCMOS33 [get\_ports {aluCtr[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {aluCtr[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {aluCtr[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {aluCtr[0]}]

set\_property PACKAGE\_PIN R2 [get\_ports {aluCtr[3]}]

set\_property PACKAGE\_PIN T1 [get\_ports {aluCtr[2]}]

set\_property PACKAGE\_PIN U1 [get\_ports {aluCtr[1]}]

set\_property PACKAGE\_PIN W2 [get\_ports {aluCtr[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {input3[0]}]

set\_property PACKAGE\_PIN V2 [get\_ports {input3[8]}]

set\_property PACKAGE\_PIN W13 [get\_ports {input3[7]}]

set\_property PACKAGE\_PIN W14 [get\_ports {input3[6]}]

set\_property PACKAGE\_PIN V15 [get\_ports {input3[5]}]

set\_property PACKAGE\_PIN W15 [get\_ports {input3[4]}]

set\_property PACKAGE\_PIN W17 [get\_ports {input3[3]}]

set\_property PACKAGE\_PIN W16 [get\_ports {input3[2]}]

set\_property PACKAGE\_PIN V16 [get\_ports {input3[1]}]

set\_property PACKAGE\_PIN V17 [get\_ports {input3[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[0]}]

set\_property PACKAGE\_PIN U7 [get\_ports {seg[6]}]

set\_property PACKAGE\_PIN V5 [get\_ports {seg[5]}]

set\_property PACKAGE\_PIN U5 [get\_ports {seg[4]}]

set\_property PACKAGE\_PIN V8 [get\_ports {seg[3]}]

set\_property PACKAGE\_PIN U8 [get\_ports {seg[2]}]

set\_property PACKAGE\_PIN W6 [get\_ports {seg[1]}]

set\_property PACKAGE\_PIN W7 [get\_ports {seg[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {sm\_wei[0]}]

set\_property PACKAGE\_PIN U2 [get\_ports {sm\_wei[0]}]

set\_property PACKAGE\_PIN U4 [get\_ports {sm\_wei[1]}]

set\_property PACKAGE\_PIN V4 [get\_ports {sm\_wei[2]}]

set\_property PACKAGE\_PIN W4 [get\_ports {sm\_wei[3]}]

**alusim：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/06 15:04:49

// Design Name:

// Module Name: alusim

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module alusim;

// Inputs

reg [15:0] input1;

reg [15:0] input2;

reg [3:0] aluCtr;

// Outputs

wire [15:0] aluRes;

wire zero;

// Instantiate the Unit Under Test (UUT)

alu uut (

.input1(input1),

.input2(input2),

.aluCtr(aluCtr),

.aluRes(aluRes),

.zero(zero)

);

initial begin

// Initialize Inputs

input1 = 1;

input2 = 1;

aluCtr = 4'b0110;

#100;

input1 = 2;

input2 = 1;

aluCtr = 4'b0110;

#100

input1 = 1;

input2 = 1;

aluCtr = 4'b0010;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0000;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0001;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0111;

#100

input1 = 0;

input2 = 1;

aluCtr = 4'b0111;

end

endmodule

**实验5：**

**top：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/06 15:21:51

// Design Name:

// Module Name: top

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module top(

input clk,

input reset, //复位信号（连接一个按键）

input [3:0] aluCtr,

//input [15:0] input1,

//input [15:0] input2,

input [8:0] input3,

output [6:0] seg,//段码

output [3:0] sm\_wei//哪个数码管

);

// ALU信号线

wire zero;

wire[31:0] aluRes;

//wire[15:0] expand;

// ALU控制信号线

wire [15:0] input1;

wire [15:0] input2;

// 实例化ALU模块

assign input1 = 16'h0x7;

assign input2[15:9] = {7{input3[8]}};

assign input2[8:0] = input3[8:0];

wire [31:0] data1;

wire [31:0] data2;

signext signext1(.inst(input1[15:0]), .data(data1));

signext signext2(.inst(input2[15:0]), .data(data2));

alu alu(.input1(data1),

.input2(data2),

.aluCtr(aluCtr),

.zero(zero),

.aluRes(aluRes));

// 。。。。。。。。。。。。。。。。。。。。。。。。。。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

display disp1 (.clk(clk),.sm\_wei(sm\_wei),.data(aluRes[15:0]),.sm\_duan(seg));

endmodule

**signext：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/09 00:52:15

// Design Name:

// Module Name: signext

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

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：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/06 14:56:40

// Design Name:

// Module Name: alu

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module alu(

input [31:0] input1,

input [31:0] input2,

input [3:0] aluCtr,

output reg[31:0] 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;

4'b0111: // 小于设置

begin

if(input1<input2)

aluRes = 1;

else

aluRes = 0;

end

default:

aluRes = 0;

endcase

end

endmodule

**alusim：**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/11/06 15:04:49

// Design Name:

// Module Name: alusim

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module alusim;

// Inputs

reg [15:0] input1;

reg [15:0] input2;

reg [3:0] aluCtr;

// Outputs

wire [31:0] aluRes;

wire [31:0] data1;

wire [31:0] data2;

wire zero;

// Instantiate the Unit Under Test (UUT)

signext uut1 (

.inst(input1),

.data(data1)

);

signext uut2 (

.inst(input2),

.data(data2)

);

alu uut3 (

.input1(data1),

.input2(data2),

.aluCtr(aluCtr),

.aluRes(aluRes),

.zero(zero)

);

initial begin

// Initialize Inputs

input1 = 1;

input2 = 1;

aluCtr = 4'b0110;

#100;

input1 = 16'hfffe;

input2 = 1;

aluCtr = 4'b0110;

#100

input1 = 1;

input2 = 1;

aluCtr = 4'b0010;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0000;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0001;

#100

input1 = 1;

input2 = 0;

aluCtr = 4'b0111;

#100

input1 = 0;

input2 = 1;

aluCtr = 4'b0111;

end

endmodule