实验二 加法器设计

一、实验目的

熟悉 Vivado 的开发环境及开发流程。学会可配置 IP 核的设计、封装及调用方法。学会使用原理图方法编写功能模块。

通过实验，使学生加深对全加器的理解，并学会多位全加器的设计。

二、实验内容

封装并使用1位全加器IP核，完成拨码开关控制LED灯设计。使用Verilog HDL 语言结构化方法实现4位带进位标志的加法器add4。

三、实验步骤

1. 制作一个1位全加器fulladd1，封装 IP 核

二进制加法器必须考虑从低位到高位的进位问题，因此需要在做加法的时候，不仅做 a_i，b_i 两个数加法，还要加上从低位进来的进位位 c_i，计算之后，不仅得到本位的和 sum_i， 还要考虑向高位的进位位 c_{i+1}.这样的加法器称为全加器（区别于不考虑进位的半加器）。

表 1-1 是全加器的真值表。

表 1-1 全加器真值表

c_i	a_i	b_i	sum_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

根据表 1-1，大家不难得出全加器的逻辑表达式。请写出 sum_i 和 c_{i+1} 的逻辑表达式， 并将该表达式带入到下面的设计中。

```
module fulladd1(
    input a,            //被加数 ai
    input b,            //加数 bi
    input cin,          //低位进位位 ci
    output sum,         //和 sumi
    output cout         //高位进位位 ci+1
    );
```

```
//  请将 sum 和 cout 的逻辑表达式填写在这里
endmodule
```

可以用下面的仿真文件进行仿真，可得到图 1-1 所示的仿真波形图。

```
`timescale 1ns / 1ps

module fulladd1_sim( );
  //input
  reg a = 0,b = 0;
  reg cin= 0;    //必须为 0
  //output
  wire sum,cout;
  fulladd1 U1(.a(a),.b(b),.cin(cin),.sum(sum),.cout(cout));
  initial begin
  #20 begin a = 0;b = 0; cin= 1;end
  #20 begin a = 0;b = 1; cin= 0;end
  #20 begin a = 0;b = 1; cin= 1;end
  #20 begin a = 1;b = 0; cin= 0;end
  #20 begin a = 1;b = 0; cin= 1;end
  #20 begin a = 1;b = 1; cin= 0;end
  #20 begin a = 1;b = 1; cin= 1;end
  end
endmodule
```

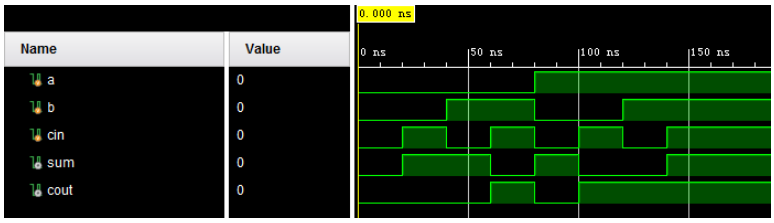


图 1-1 1 位加法器的仿真波形图

参照实验一的方法封装，生成1位全加器IP核。

2. 使用1位全加器IP核，完成拨码开关控制LED灯设计

1) 导入 IP 核

创建一个新的项目，在界面左侧的 Project Manager 中点击 Project Settings，打开 Project Settings 对话框，并转到 IP 项上。如图 1-2所示。

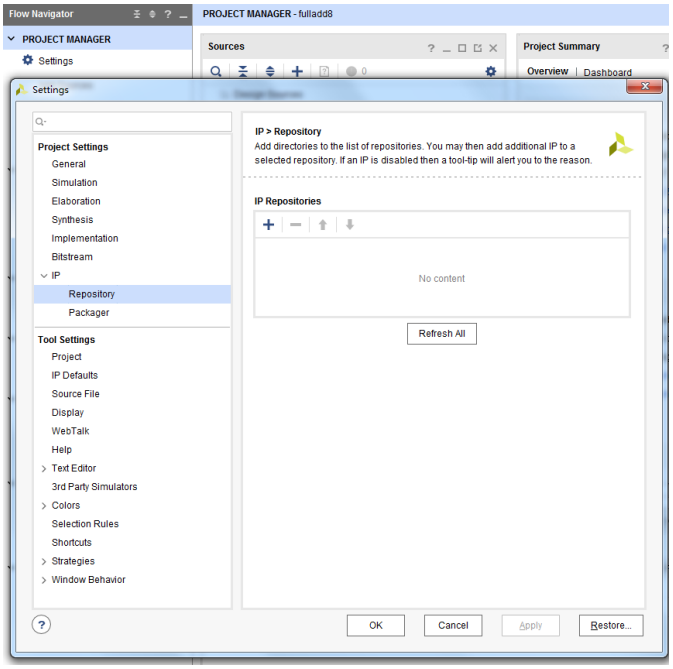


图 1-2 项目设置中添加IP 核

点 +，找到下面的目录：D:\digit_FPGA \IPcore。如图 1-3 所示，选择 IPcore 的文件夹，点击 Select。

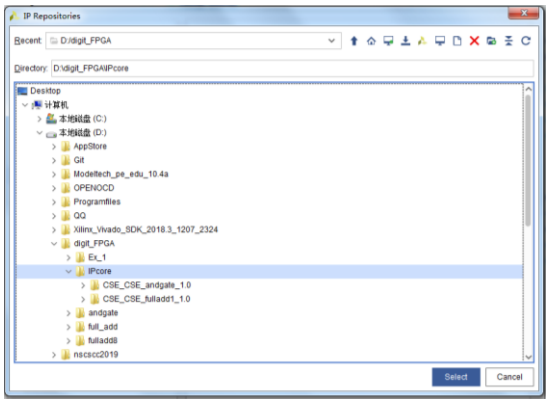


图 1-3 选择IP 核的位置

系统弹出如图 1-4 所示的 Add Repository 对话框，点击OK。

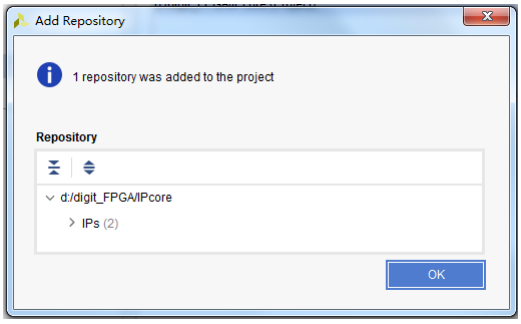


图 1-4 Add Repository 对话框

现在回到了如图 1-5 所示的 Project Settings 对话框。点击 Apply，然后点击 OK。

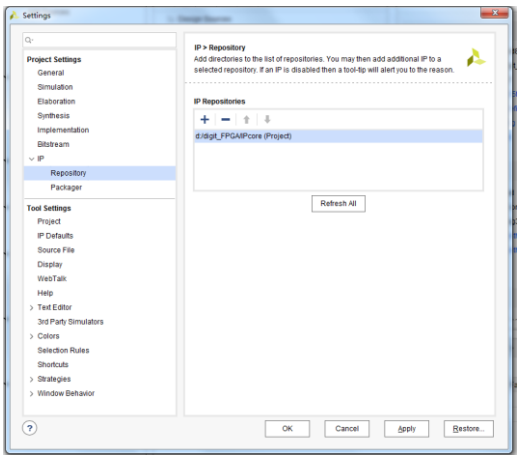


图 1-5　　导入IP 核后的Project Settings 对话框

现在点击Project Manager 下的IP Catalog，就会看到如图 1-6 所示的界面中，IP Catalog里已经有了自己设计的IP核。

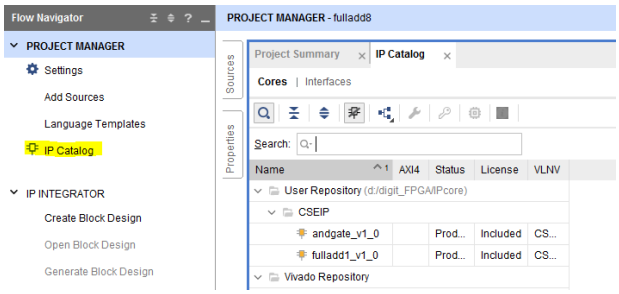


图 1-6　　IP Catalog 中的 IP核

2)　创建 bd 设计文件

点击 Flow Navigator中的IP Integrator-> Create Block Design，打开图 1-7 所示的对话框，设置后点击 **OK**。

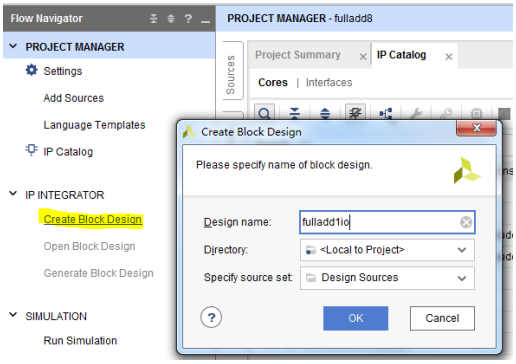


图 1-7　创建 bd 文件

现在得到了图 1-8 所示的界面。

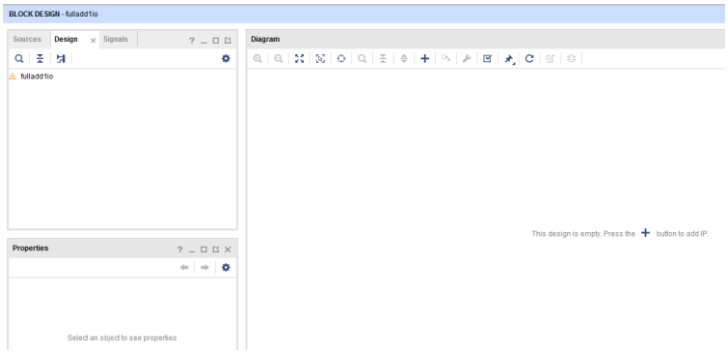


图 1-8　完成 bd 文件创建

3)　放置基本门电路

现在点击Project Manager下的IP Catalog，在如图 1-9 所示的窗口中双击要添加的IP核名称。

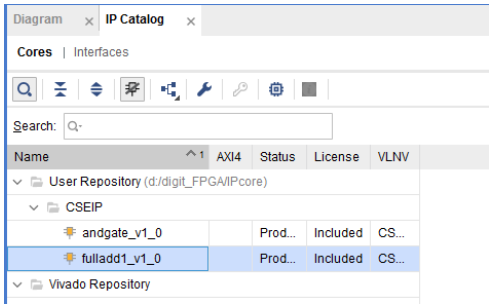


图 1-9　选择 IP 核

在出现的如图 1-10 所示的对话框中选择 **Add IP to Block Design**。

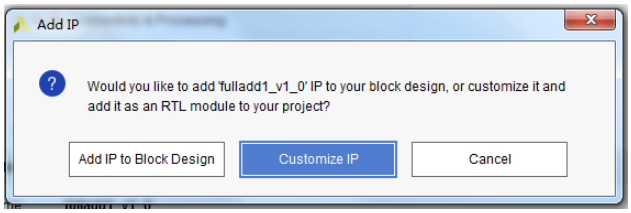


图 1-10　添加 IP 核

此时界面上出现了所选择的 IP 核，如图 1-11 所示。

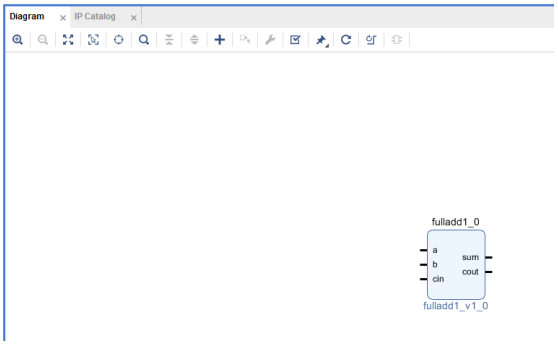


图 1-11　放置一个 IP 核后

4） **放置输入输出端口**

在图 1-11 中空白处右键点击，在弹出的菜单中选择 **Create Port...**。打开如所示的对话框，按照图 1-12 所示设置，添加端口名称，设置端口方向。

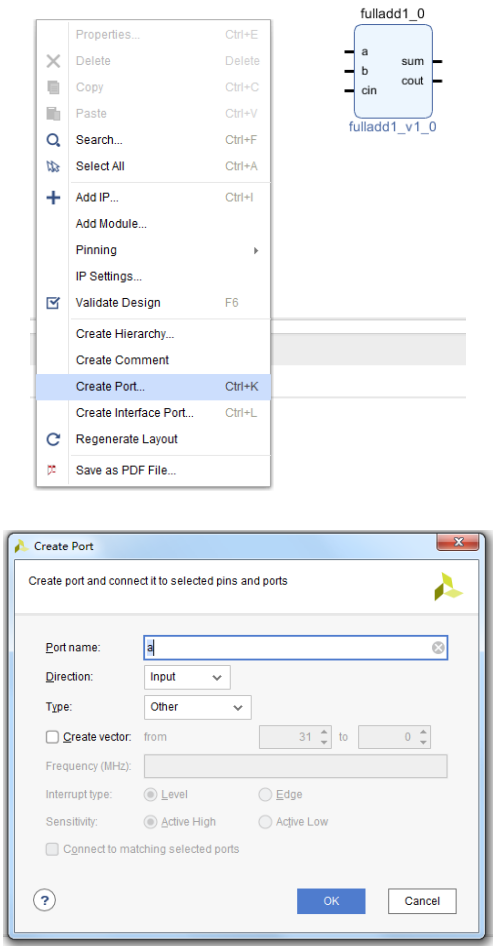


图 1-12 添加端口

按照上述方法在增加输入端 a , b , cin 和输出端 sum , cout 。并按照图 1-13 所示连接好电路

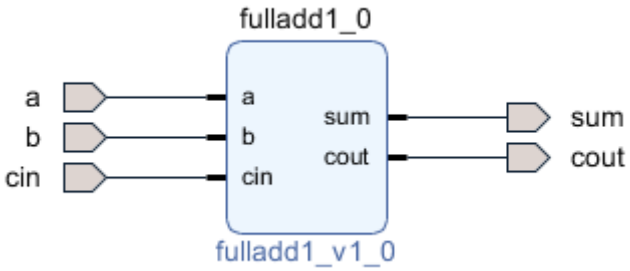


图 1-13 连接好的电路

5） **生成输出文件**

在 Project Manager 的 Source 界面中，右键点击添加的IP核，在弹出的菜单中选择 Generate Output Products...，之后在弹出的如图 1-14 所示的 Generate Output Products 对话框）中选择Global，点击 Generate。

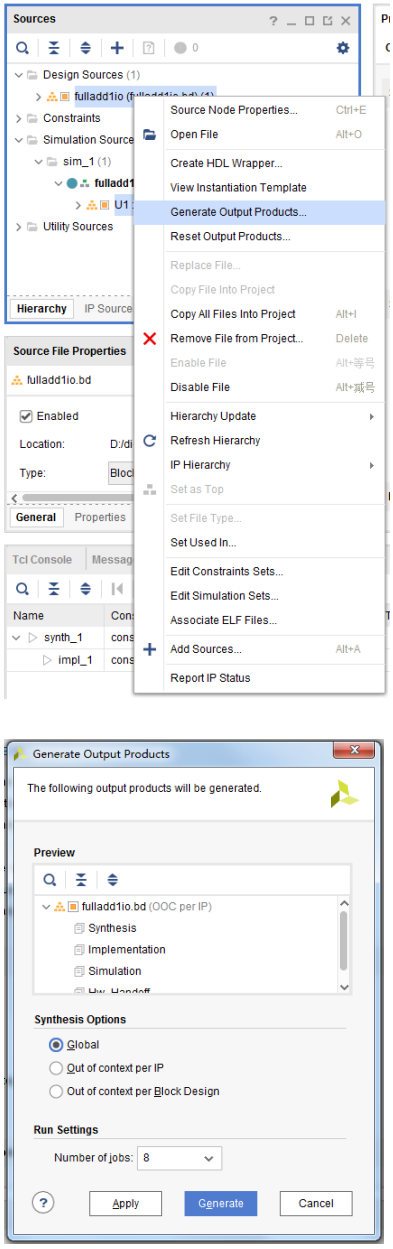


图 1-14 生成输出文件

6） 生成 HDL 文件

再次在 Project Manager 的 Source 界面中，右键点击 fulladd1io，在弹出的菜单中选择 Create HDL Wrapper...，之后在弹出的如图1-15 所示的Create HDL Wrapper 对话框中选择Let Vivado Manage wrapper and auto-update，点击 **OK**。

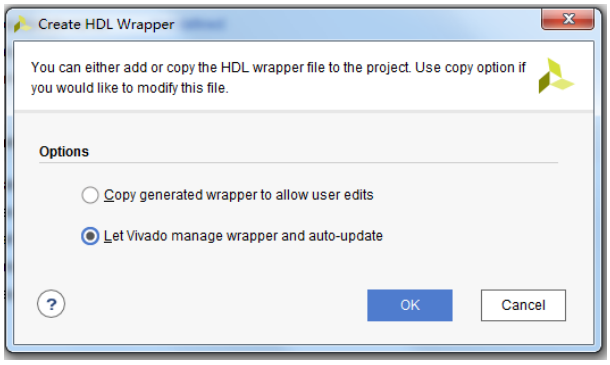


图 1-15 生成 HDL 文件

7) 使用约束文件fulladdio_ip.xdc进行管脚约束文件配置、综合、实现、生成比特流文件，下载到板子上进行验证。

表 1-2 管脚分配表

信号	部件	管脚	信号	部件	管脚
cin	SW7	R1	cout	D7	J2
a	SW6	N4	sum	D8	K2
b	SW5	M4			

3. 用结构化描述方法设计4位加法器

4 位加法器是由上述设计的 4 个 1 位全加器通过串联组成的，其框图如图 1-16 所示。最低位进位置为 c0，每一位的 cin 输入来源于低位的 cout 输出，最高位的 cout 作为加法运算的进位位。

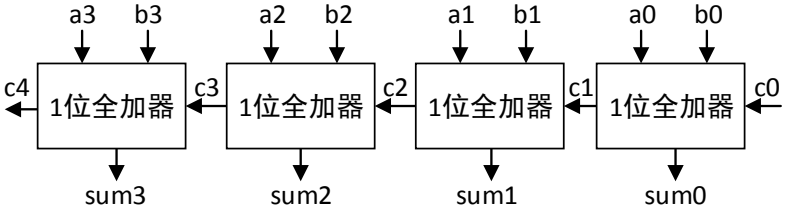


图 1-16 4位加法器框图

创建新文件add4.v，同时选择 Add Sources 添加fulladd1.v源文件。在新建的add4.v源文件中多次实例化1位全加器，扩展成4位全加器。

```
module add4(
  input [3:0] a,          //被加数
  input [3:0] b,          //加数
  input cin,              //低位进位
  output cout,            //进位
  output [3:0] sum        //和
  );
  wire [2:0] scout;   //用作保存中间进位
  fulladd1 f0(a[0],b[0],cin,sum[0],scout[0]);//调用方法1，注意变量
```

顺序必须严格对应子模块代码中声明模块输入输出的顺序

……　　//参考上一行的模块调用语句完善此部分

```
endmodule
```

可以用下面的仿真文件进行仿真，可得到图 1-17 所示的仿真波形图。

```
`timescale 1ns / 1ps

module add4_sim( );
    //input
    reg [3:0] a = 4'd7;
    reg [3:0] b = 4'd6;
    reg cin= 0;        //必须为 0
    //output
    wire [3:0]sum;
    wire cout;
    add4 U(.a(a),.b(b),.cin(cin),.sum(sum),.cout(cout)); //调用方法2，每一变量的调用格式为.[子模块变量名]([当前模块变量名])
    initial begin
      #20 begin a = 4'd15;b = 4'd1;end  //加法溢出
      #20 begin a = 4'd13;b = 4'd2;end
    end
endmodule
```

仿真后的时序图如下所示：

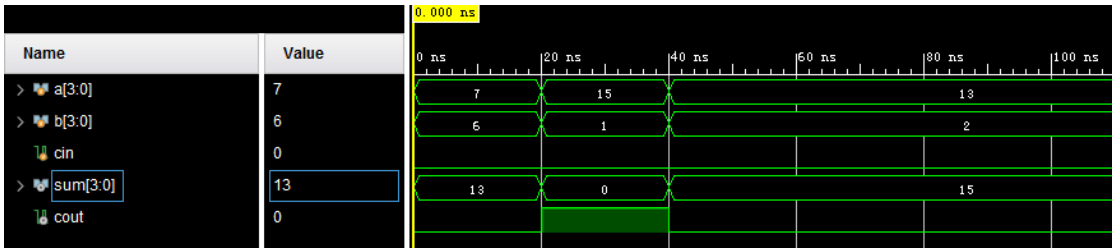
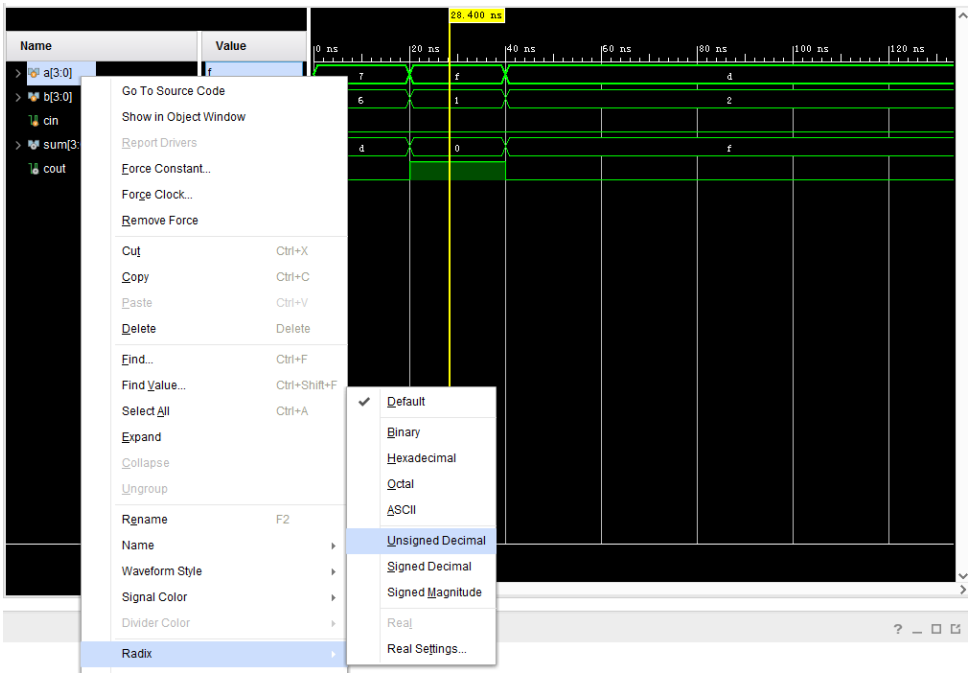


图 1-17 4 位加法器的仿真波形图

注：时序图中的数字请使用十进制显示。仿真波形可能默认使用十六进制显示数值，切换的方法是右键点击时序图最左侧的**Name**一栏中的变量名，在菜单中点击**Radix**，在弹出的菜单中将进制从**Default**切换到**Unsigned Decimal**(无符号十进制数)，这样能更直观的看到加法器逻辑是否正确。



使用给出的约束文件add4.xdc完成管脚分配。

表 1-3 4 位全加器管脚分配表

信号	部件	管脚	信号	部件	管脚
a[3]	SW0	P5	b[3]	SW4	R2
a[2]	SW1	P4	b[2]	SW5	M4
a[1]	SW2	P3	b[1]	SW6	N4
a[0]	SW3	P2	b[0]	SW7	R1
sum [3]	D5	H4	cin	SW8-1	U3
sum [2]	D6	J3	cout	D1	F6
sum [1]	D7	J2			
sum [0]	D8	K2			