

## 实验二 2 选 1 多路选择器——IP 核的封装与调用

### 实验目的

通过实验，学会设计 1 位数据的 2 选 1 多路选择器，同时学会调用 IP 核的方法和通过 Block Design 设计电路的方法。

### 实验内容

分别用 Block Design 设计方法和添加源代码的方法设计 1 位数据的 2 选 1 多路选择器，并验证其正确性。当选择开关  $s = 0$  时，电路的输出值  $c$  就是  $a$ ；当选择开关  $s = 1$  时，电路的输出值  $c$  就是  $b$ 。

1 位 2 选 1 多路选择器真值表如下：

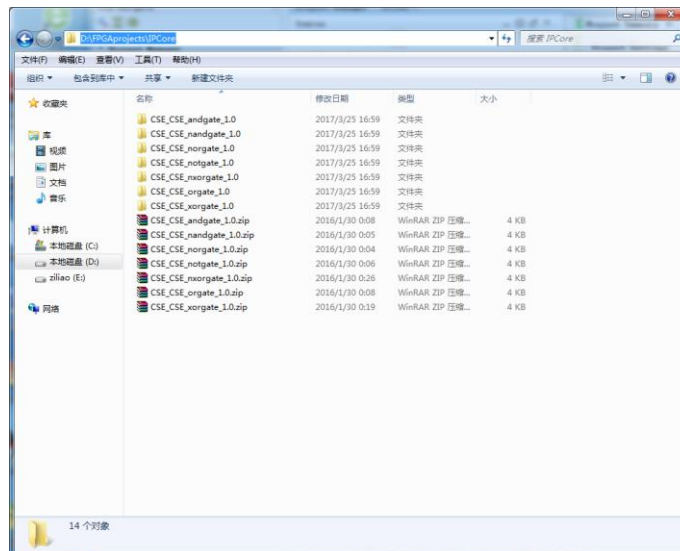
输入			输出
$a$	$b$	$s$	$c$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

### 实验步骤

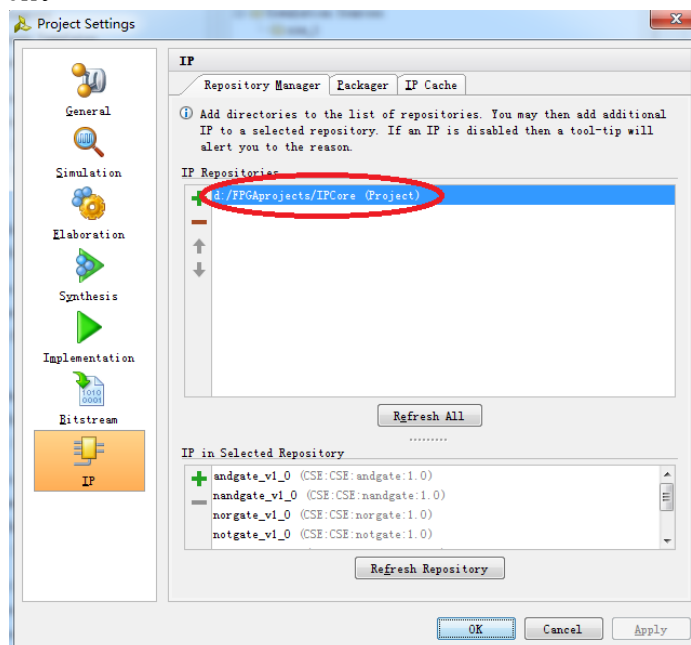
本次实验利用 3 种方法实现 2 选 1 多路选择器，分别是：Block Design，用 Verilog 语言的结构化描述设计，用 Verilog 语言的行为描述设计。具体步骤如下。

#### 一、Block Design

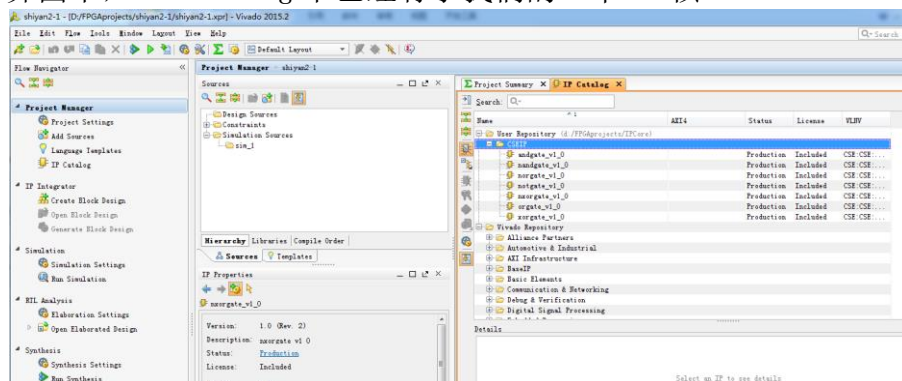
- 1、根据 1 位 2 选 1 多路选择器的功能画出电路图。
- 2、打开 vivado 开发环境，新建工程 shiyan2-1，得到空白的 Vivado 工程界面（注意：工程名称、路径不能含中文或特殊字符，N4 板卡型号为 XC7A100TCSG324-1, B3 板卡型号为 xc7a35tcpg236-1）。
- 3、导入 IP 核
  - 1) 创建 D:\FPGAprojects\IPCore 文件夹，将实验一中设计与封装的相关门电路的 IP 核文件（.zip 文件）全部拷贝到这个文件夹下并分别解压这些.zip 文件。



- 2) 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。点击添加按钮, 如图添加 IP 核路径 D:\FPGAprojects\IPCore。点击 Apply, 然后点击 OK。

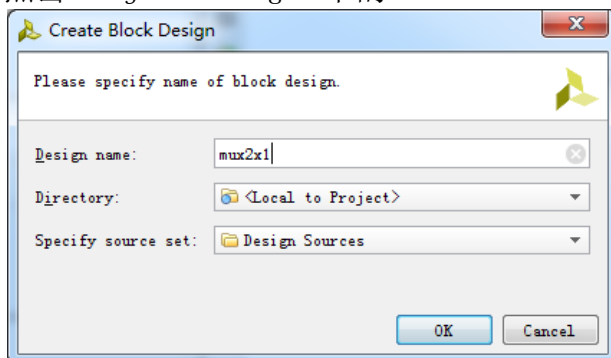


- 3) 点击 Project Manager 下的 IP Catalog, 我们会看到如图所示的界面中, IP Catalog 中已经有了我们的 7 个 IP 核。

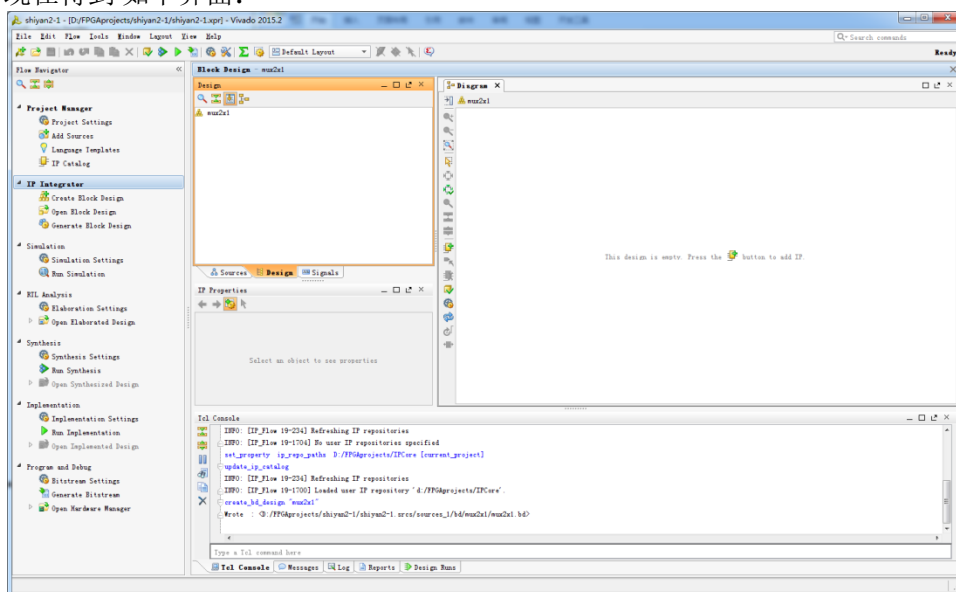


#### 4、创建 bd 设计文件

1) 点击 Project Manager 下的 Create Block Design。

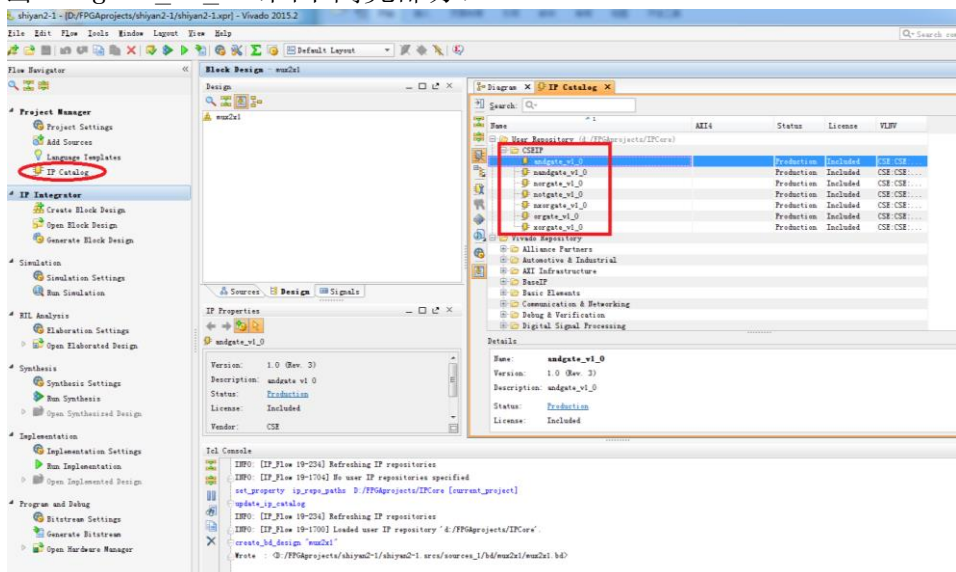


现在得到 如下界面：

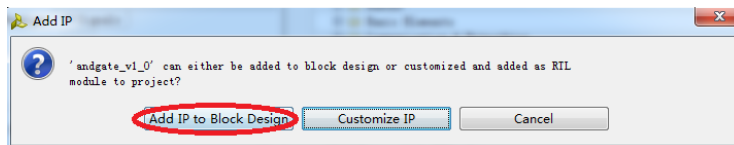


2) 放置电路

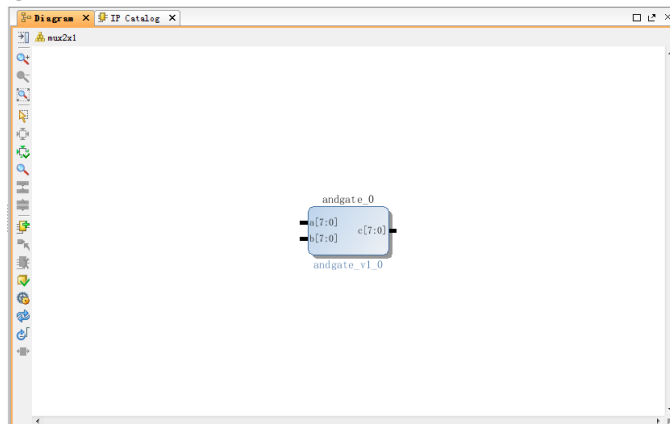
①点击 Project Manager 下的 IP Catalog，在如图所示的窗口中双击 andgate\_v1\_0（图中高亮部分）。



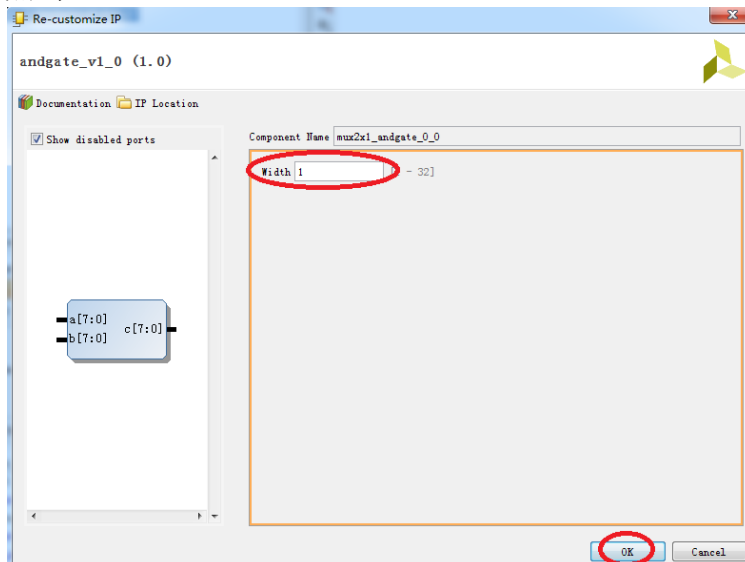
②在出现的如图 2-57 所示的对话框中选择 Add IP to Block Design。



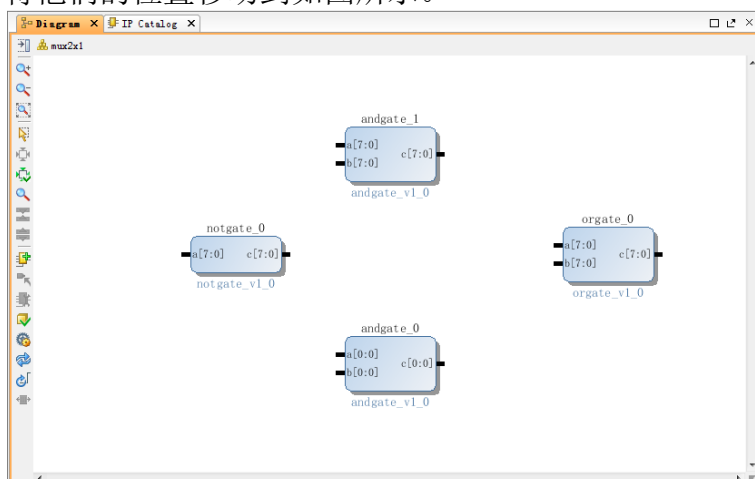
③此时界面上出现了所选择的 IP 核心，如图所示。



④双击图中的 andgate\_0 器件，按照图中设置数据宽度为 1，然后点击 OK。

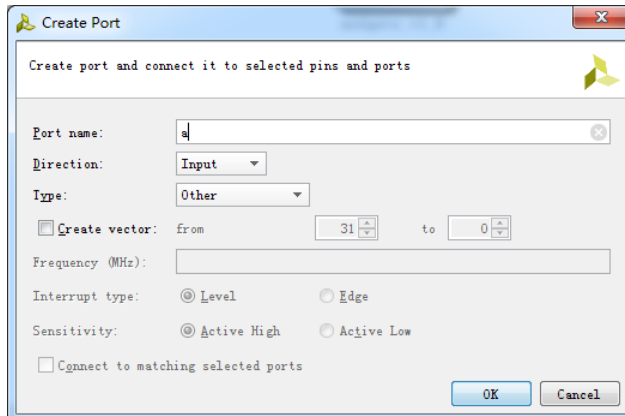


⑤这样我们就放置好了一个与门，按照这个方法再放置另一个与门，一个或门和一个非门，并设置好他们的数据宽度都是 1。放置好后，将他们的位置移动到如图所示。

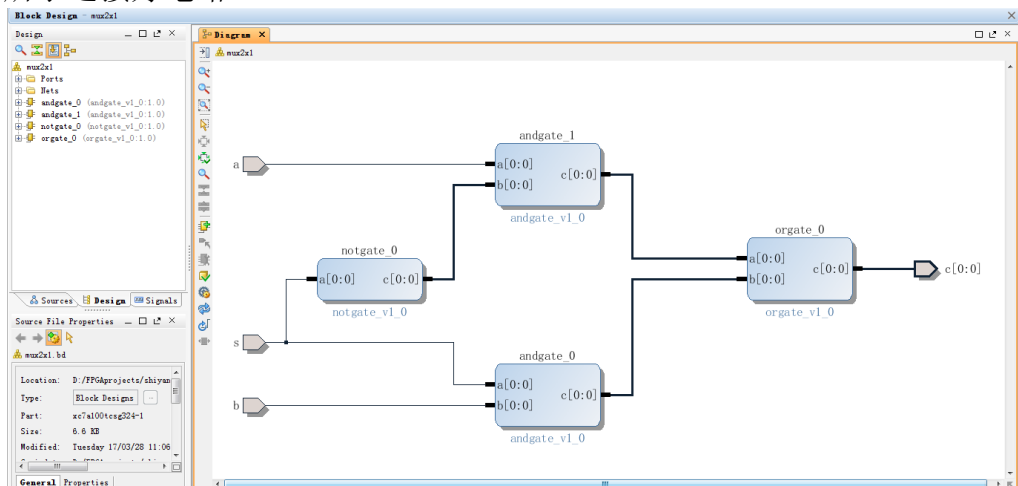


### 3) 放置输入输出端口

①在空白处右键点击，在弹出的菜单中选择 Create Port...。在弹出的对话框中，如下图所示设置输入端  $a$ ，输入端  $a$  即被添加。

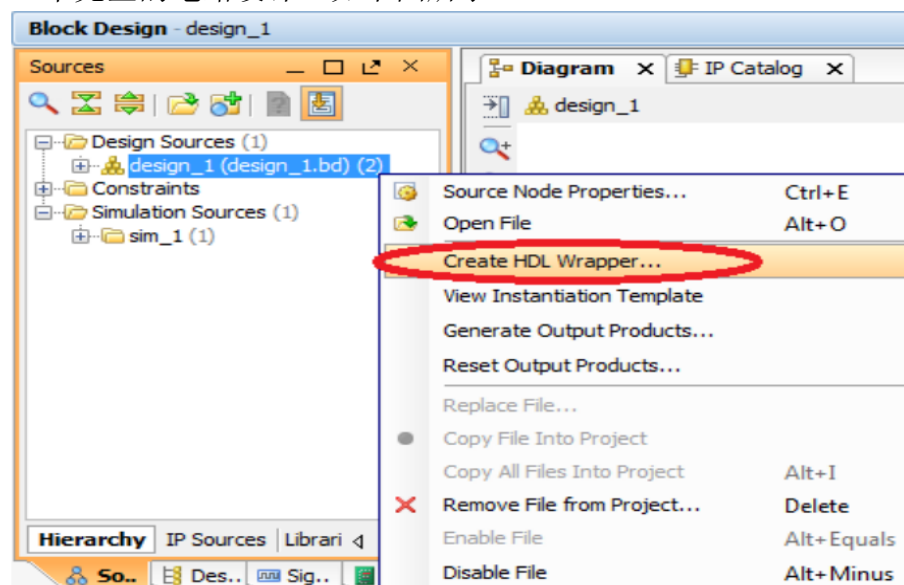


②按照上述方法再添加输入端  $b$ ，输入端  $s$  和输出端  $c$ 。并按照下图所示连接好电路。



### 4) 创建 HDL wrapper

连接好电路后需对 Block Design 进行打包，创建 HDL wrapper, 生成一个完整的电路设计。如下图所示。

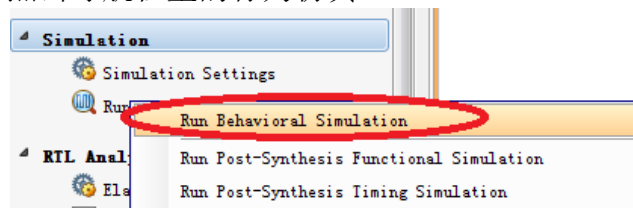


## 5、仿真验证

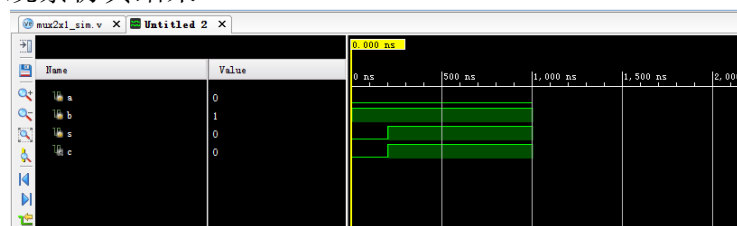
参考实验一的方法，建立仿真文件并进行行为仿真。

```
module mux2x1_sim();  
    // input  
    reg a=0;  
    reg b=1;  
    reg s=0;  
    //output  
    wire c;  
    mux2x1 u(.a(a),.b(b),.s(s),.c(c));  
    initial begin  
        #200 s=1;  
    end  
endmodule
```

点击导航栏里的行为仿真



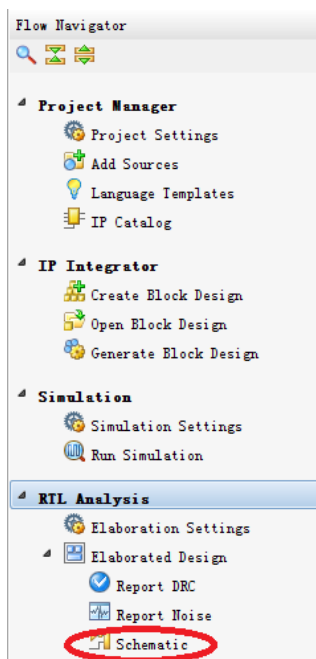
观察仿真结果



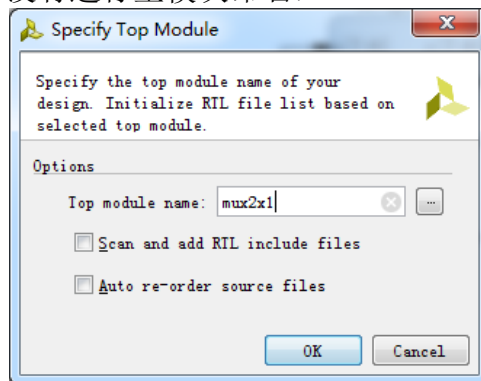
仿真结果显示，开关 $s = 0$ 时，电路的输出值 $c$ 就是 $a$ ，当开关 $s = 1$ 时，电路的输出值 $c$ 就是 $b$ ，设计正确。

## 6、RTL 分析

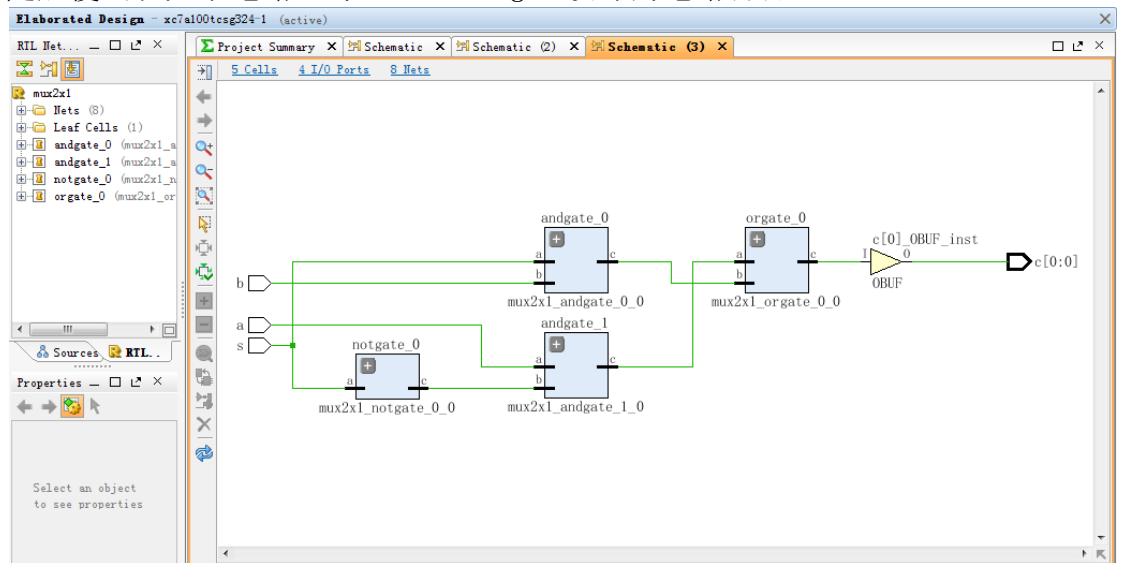
在导航栏里点击 RTL Analysis 下的 Elaborated Design 中的 Schematic。



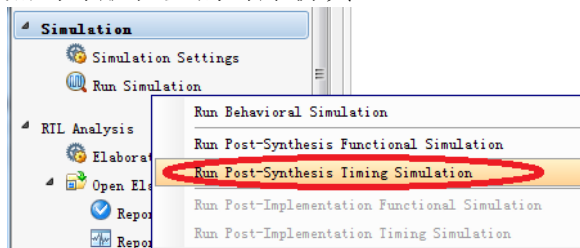
在弹出的对话框中输入主模块名字（因为在之前的 Block Design 中没有进行主模块命名）。



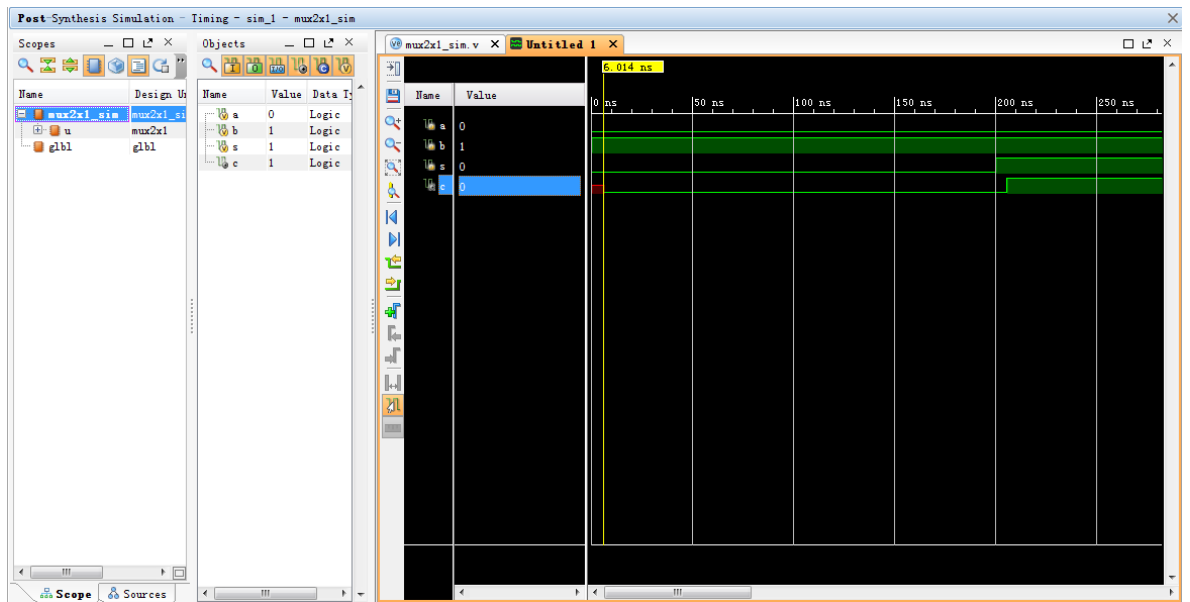
随后便出现如下电路，与 Block Design 设计的电路吻合。



7、综合，然后进行时序仿真，观察结果  
点击导航栏里的时序仿真



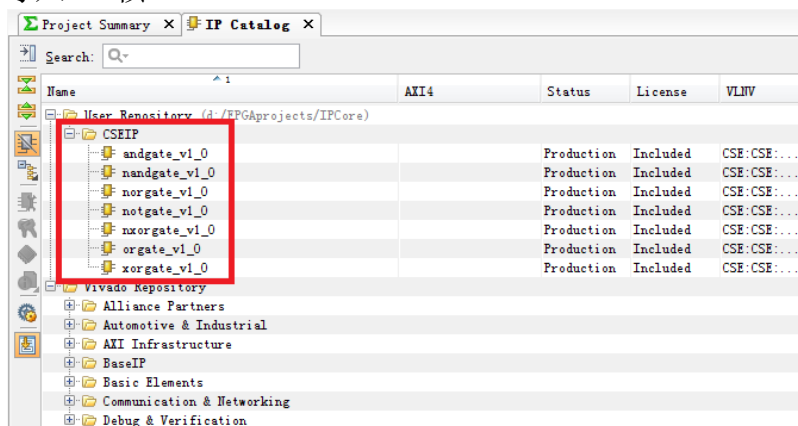
观察结果



8、实现、管脚分配、下载程序至开发板观察运行情况。

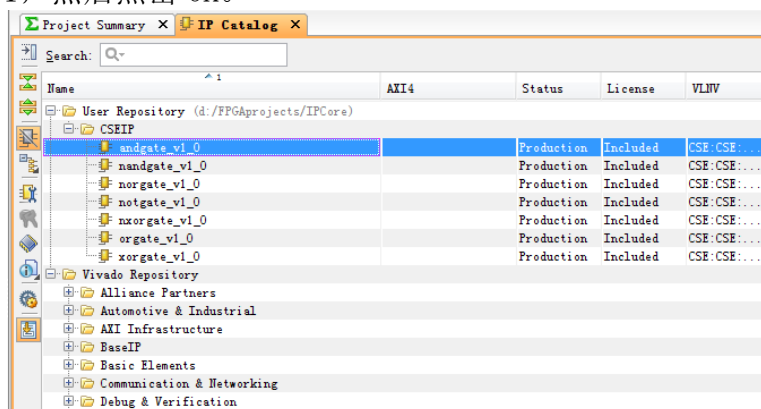
## 二、用 Verilog 语言的结构化描述设计

- 1、打开 vivado, 新建工程 shiyan2-2。
- 2、导入 IP 核。

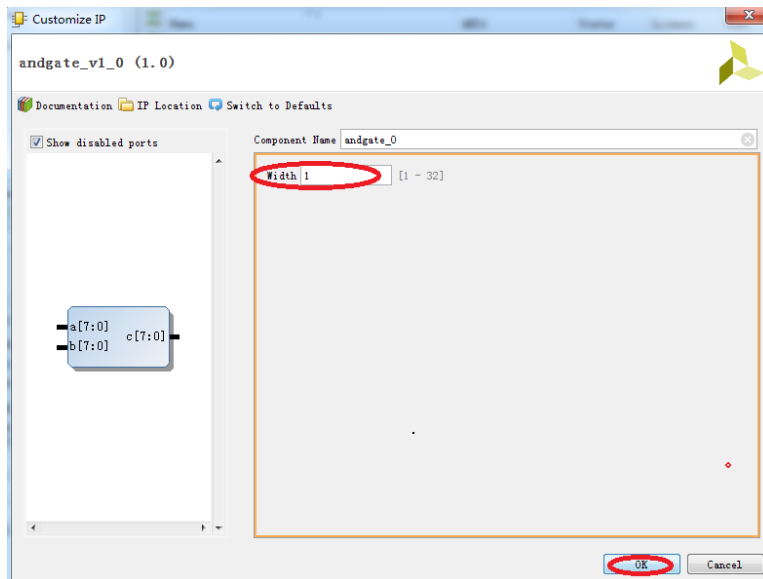


## 3、调用 IP 核

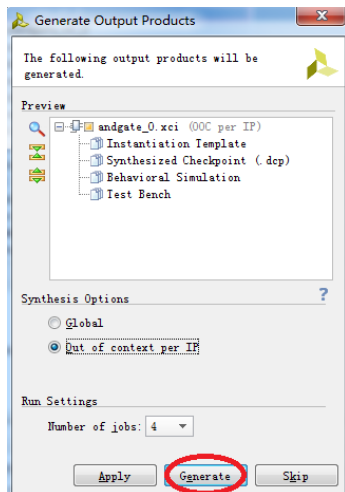
- 1) 点击 Project Manager 下的 IP Catalog, 在如图所示的窗口中双击 andgate\_v1\_0 (图中高亮部分), 在弹出的窗口中设置数据宽度为 1, 然后点击 OK。



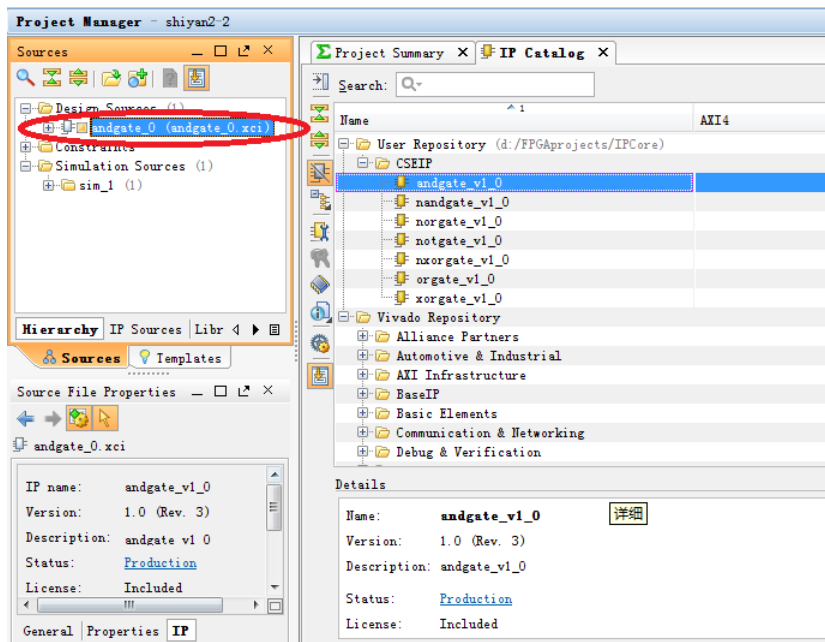




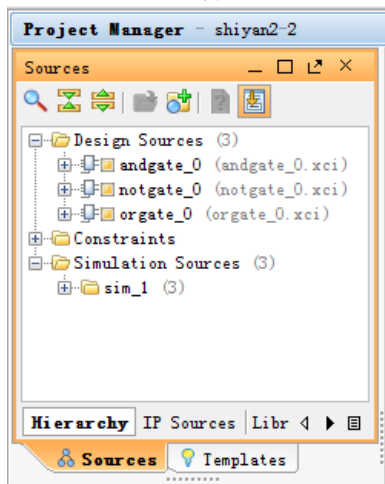
弹出如下图所示的 Generate Output Products 窗口。点击 Generate。



注：如果此时弹出系统安全警告，选择允许 vivado 相关程序的运行。在随后弹出的对话框中点击 OK。此时我们在 Project Manager 的 Sources 中可以看到刚加进去的 andgate\_0，如下图所示。



按照此方法再设置或门和非门。设置完后可以在 Project Manager 的 Sources 中看到如下图所示的情况。



#### 4、创建设计文件（mux2x1verilog.v）

添加设计文件并编写 mux2x1verilog 模块。

```
module mux2x1verilog(
input a,
input b,
input s,
output c
);
wire a1,b1,sel;
notgate_0 u0(.a(s),.c(sel));
andgate_0 u1(.a(a),.b(sel),.c(a1));
andgate_0 u2(.a(s),.b(b),.c(b1));
orgate_0 u3(.a(a1),.b(b1),.c(c));
endmodule
```

#### 5、添加仿真文件，并进行行为仿真验证其逻辑正确性

The screenshot displays the Icarus Verilog GUI. The top window shows the Verilog code for a 4-bit ripple-carry adder. The bottom window shows the simulation results in a waveform viewer. The waveform shows the output of the adder for a 4-bit input 'a' (0, 1, 1, 1) and a 4-bit input 'b' (1, 1, 1, 1). The output 's' (sum) is 1, 1, 1, 1, and the output 'c' (carry) is 1, 1, 1, 1. The simulation time is 0 ns to 300 ns.

The screenshot displays the 'Post-Synthesis Simulation - Timing - sim\_1 - mux2x1\_sim' window. The main area shows a timing diagram for signals 'a', 'b', 's', and 'glbl'. The time scale ranges from 0 ns to 35 ns. Signal 'a' is a constant logic 0. Signal 'b' is a constant logic 1. Signal 's' is a constant logic 0. Signal 'glbl' is a constant logic 1. The timing diagram is titled 'mux2x1verilog.v' and 'Untitled 2'.

### 三、用 Verilog 语言的行为描述设计

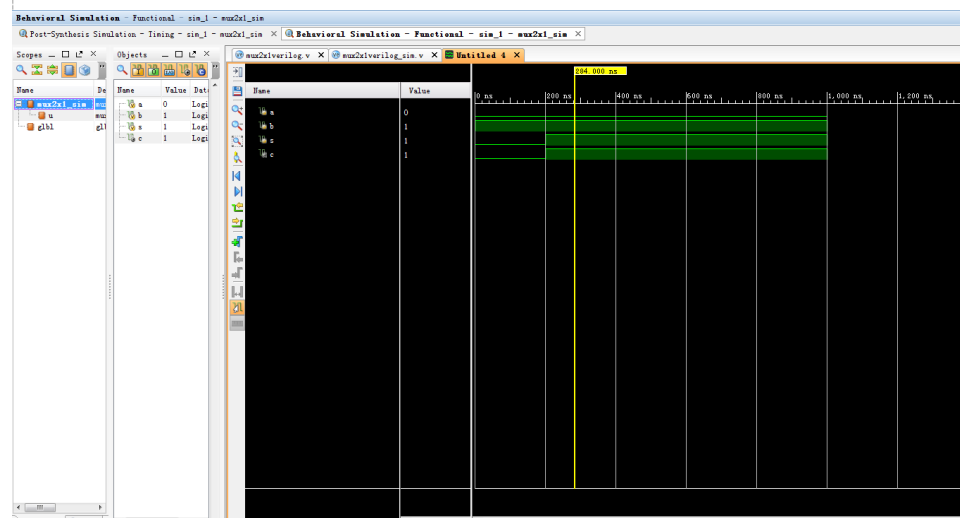
## 2、创建设计文件 (mux2x1verilog.v)

```
module mux2x1verilog(
input a,
input b,
input s,
output c
);
    assign c=(a&~s) | (b&s);
endmodule
```

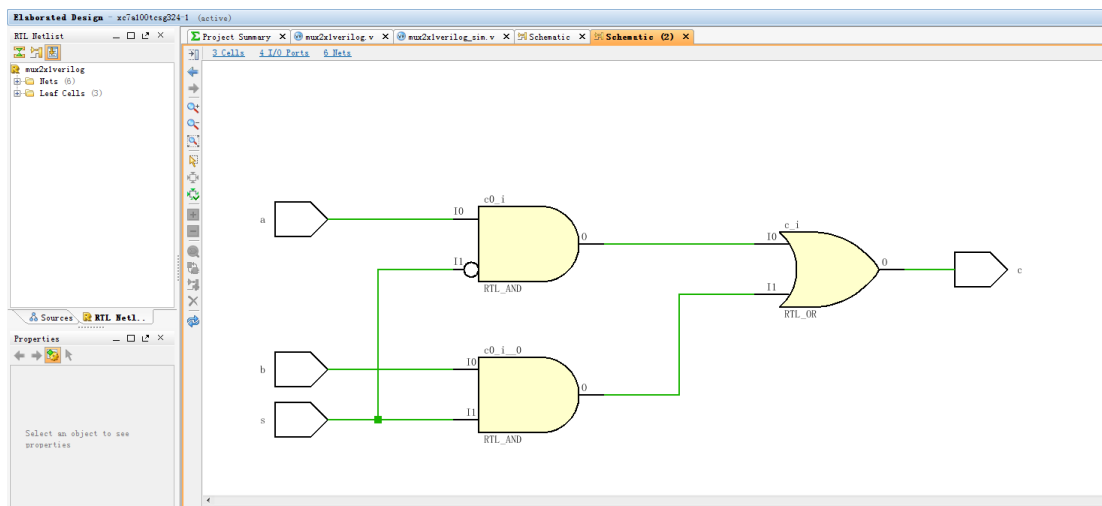
```

module mux2x1_sim ( );
// input
reg a=0;
reg b=1;
reg s=0;
//output
wire c;
mux2x1verilog u ( a(a), . b(b), . s(s), . c(c) );
initial begin
# 200 s=1;
end
endmodule

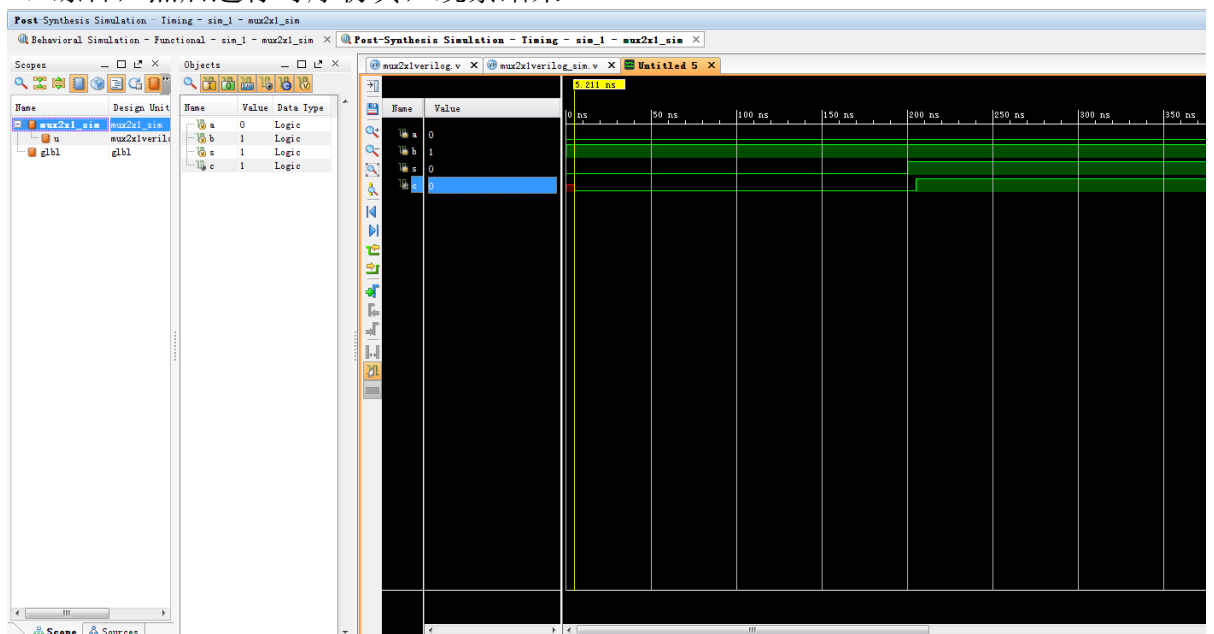
```



4、RTL 分析，与设计代码吻合，没有调用 IP 核。



5、综合，然后进行时序仿真，观察结果。



6、实现、管脚分配、下载程序至开发板观察运行情况。