实验五 在vivado中进行运算器的设计

**一、实验目的**

1、掌握一位全加器的工作原理和逻辑功能

2、掌握串行进位加法器的工作原理和进位延迟

3、掌握减法器的实现原理

4、掌握加减法器的设计方法

5、掌握ALU基本原理及在CPU中的作用

6、掌握ALU的设计方法

**二、实验设备**

1、装有vivado的计算机 1台

2、EGO1开发板 1块

**三、实验任务**

1、采用原理图方式设计4位加/减法器

2、实现4位ALU及应用设计

**四、实验原理**

**（一）、1位全加器的设计**

三个输入位:数据位Ai和 Bi，低位进位输入Ci。二个输出位:全加和Si，进位输出Ci+1

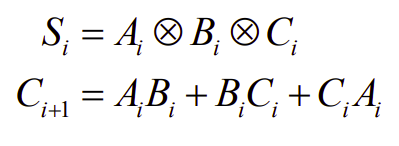
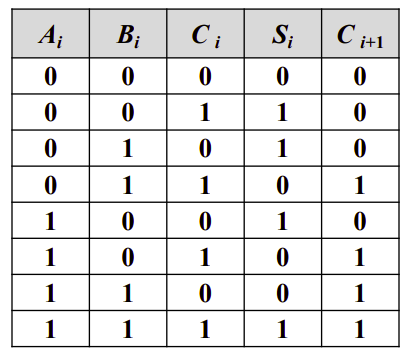


图1

根据一位全加器的输入输出关系,得到电路图如下，在vivado中可以用对应的verilog的结构描述法进行结构描述。或者用数据流法进行描述。

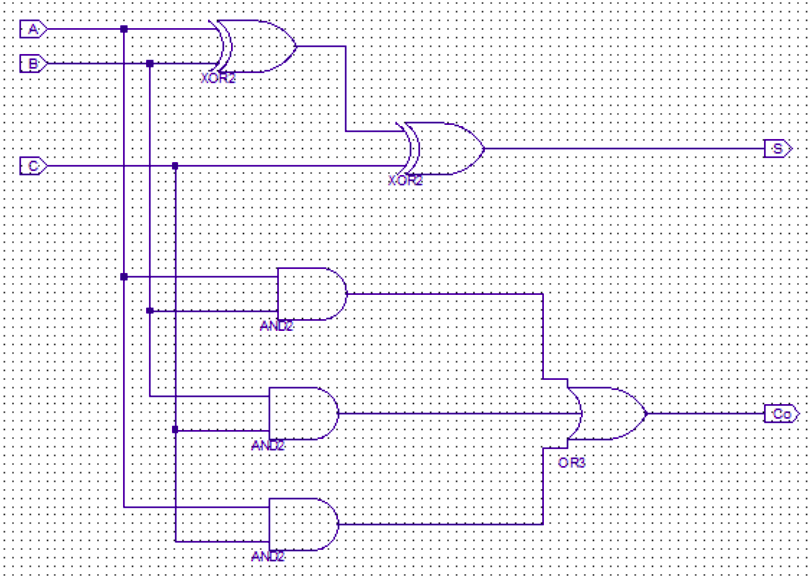


图2

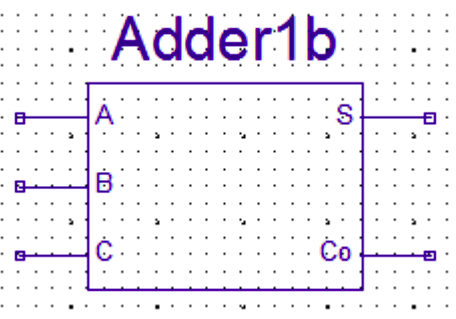


图3

**（二）、四位串行加法器的设计**

多位串行进位加法器

由一位全加器将进位串接构成。低位进位C0为0，Ci为高位进位输出

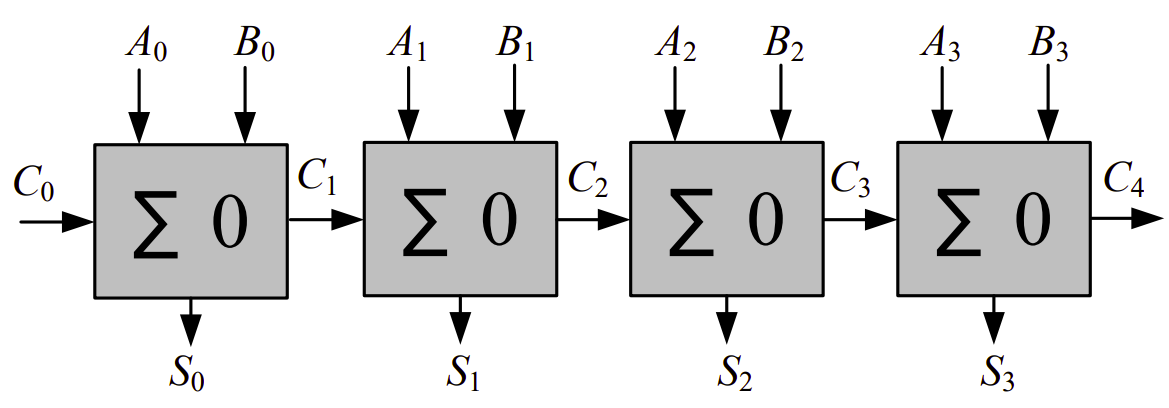


图4

根据四位加法器的输入输出关系, 得到电路图如下，在vivado中可以用对应的verilog的结构描述法进行结构描述。或者用数据流法进行描述。

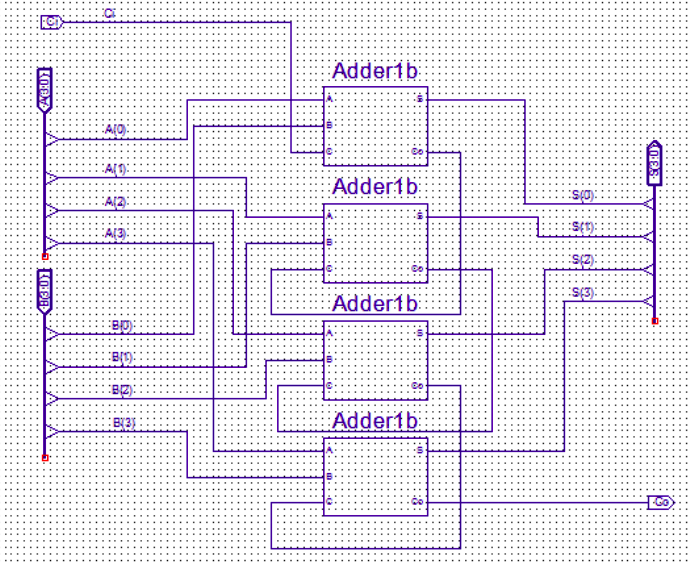


图5

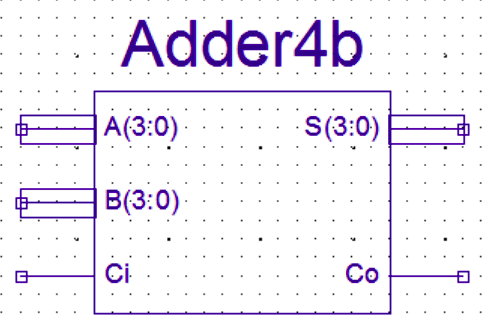


图6

**（三）、一位加减法器的设计**

1位加减法器

用负数补码加法实现,减数当作负数求补码，共用加法器。用“异或”门控制求反,低位进位C0为1。根据一位减法器的输入输出关系, 得到电路图如下，在vivado中可以用对应的verilog的结构描述法进行结构描述。或者用数据流法进行描述。

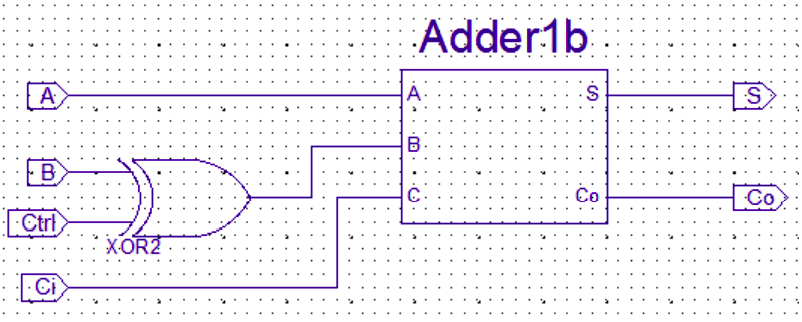


图7

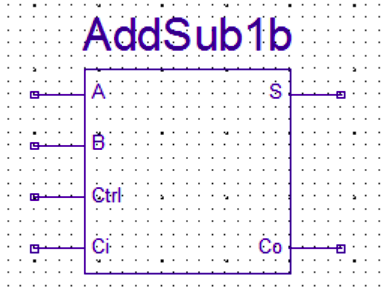


图8

**（四）、多位串行进位全加/全减器的设计原理**

八位串行进位全加/全减器：如下图是采用一位全加器串联而成的八位串行进位全加/全减器。用负数补码加法实现,减数当作负数求补码，共用加法器。用“异或”门控制求反,低位进位C0为1时，进行减法运行（补码加）。低位进位C0为0时，进行加法运行（补码加）。

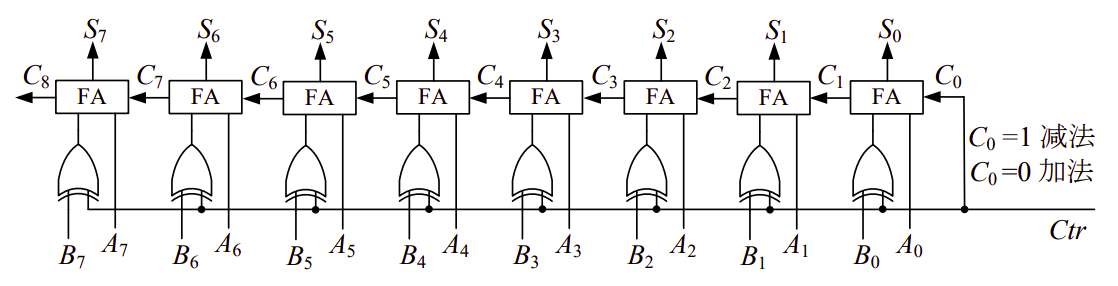
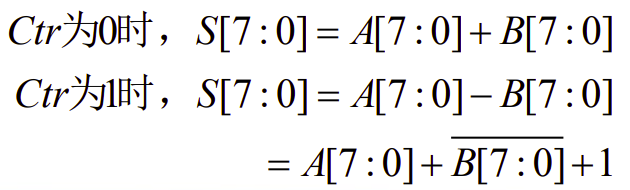


图9



根据**四位减法器**的输入输出关系,得到该**4位加减法器电路图**如下，在vivado中可以用对应的verilog的结构描述法进行结构描述。

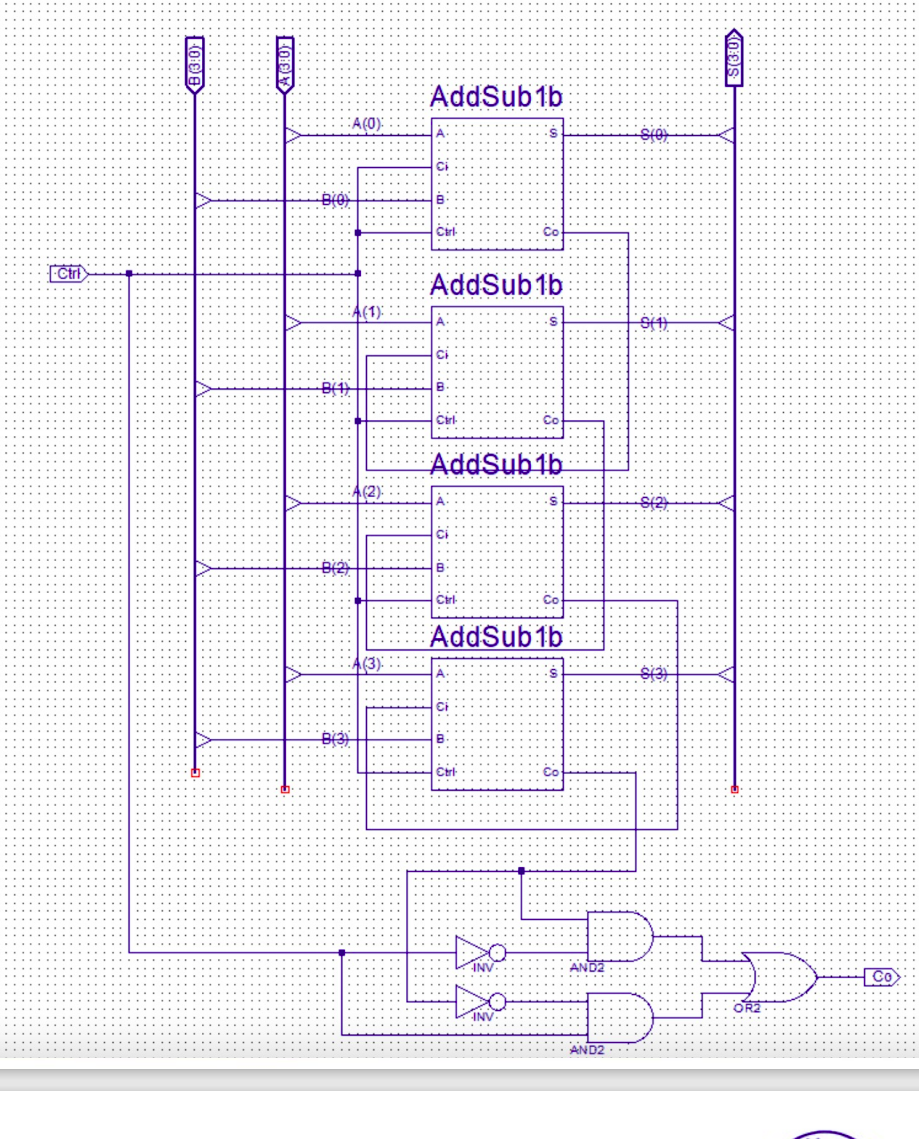


图10



图11

仿真前面的实验，需要我们自己添加不同的激励输入信号，进行波形仿真。得到不同的波形图。

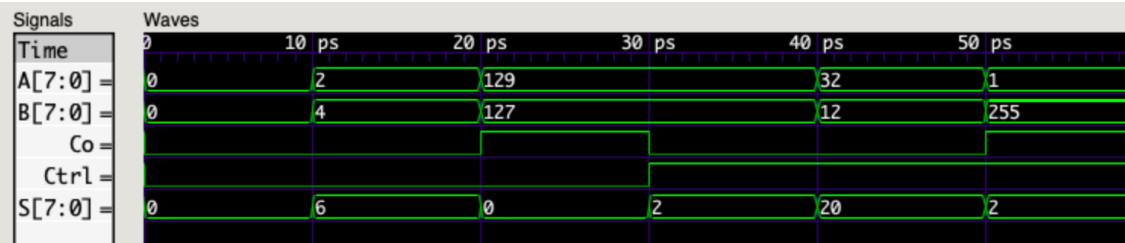


图12

**（五）、用多路选择器来实现四位ALU设计原理**

已知四位与门模块构成的电路图如下，在vivado中可以用对应的verilog的结构描述法进行结构描述。或者用数据流法进行描述。

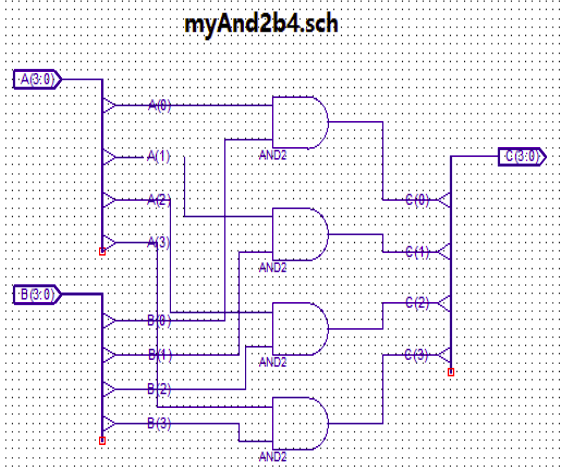


图13

已知四位或门模块构成的电路图如下。在vivado中可以用对应的verilog的结构描述法进行结构描述。或者用数据流法进行描述。

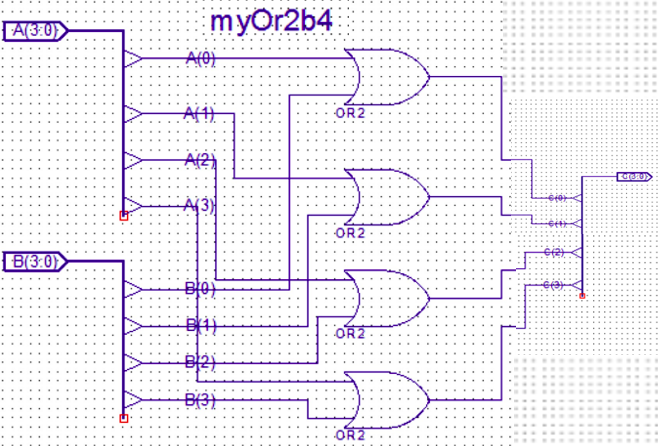


图14

在已有上面两个模块的情况下，根据输入输出关系,后端采用多路选择器构成四位ALU的电路图如下图15所示，在vivado中可以用对应的verilog的结构描述法进行结构描述。或者用数据流法进行描述。

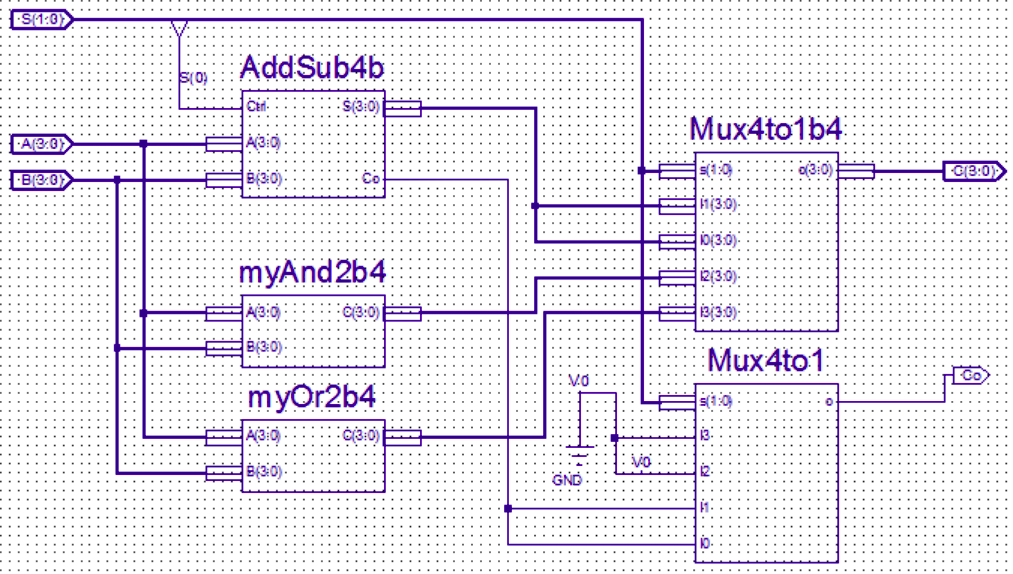


图15

输入对应的激励，仿真出该电路对应的输出结果，思考仿真结果是否与实际分析得到的结果一致。

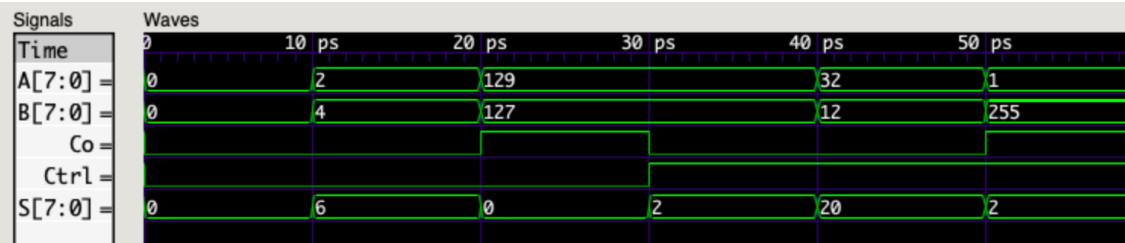


图16

**（六）、**结合EGO1开发板上的sw0、sw1……sw7这8个开关，开发板上的8个七段显示器。将上述电路中的A[3:0]绑定到开关sw0、sw1 、sw2、sw3上面。将上述电路中的B[3:0]绑定到开关sw5、sw6 、sw7、sw8上面。输出Co和S[3:0]绑定到5个放光二极管上去。例如：绑定到D1、D2、D3、D4和D5这5个发光二极管上去。在项目中生成bit文件，然后下载到开发板中。拨动开关sw0、sw1……sw7，设定2个数人数A和B的值，观察5个发光二极管的亮灭的值。看看对应的二进制数之间是否有加法的对应关系。

**五、具体的实验步骤**

**** 在vivado中进行设计，通常包含下面几个步骤：通过编写 HDL 文件的方式创建 Vivado 设计；建立仿真；通过 I/O Planing 添加管脚约束；通过编写约束文件添加管脚约束； 添加时序约束；生成 Bitstream 文件；将生成的 Bitstream 文件下载到 FPGA 开发板里。具体步骤描述如下。

1. **在 Vivado 中创建一个新的空项目。**

步骤如下：

1、启动 Vivado 2017 开发工具后，软件初始界面如下图所示：

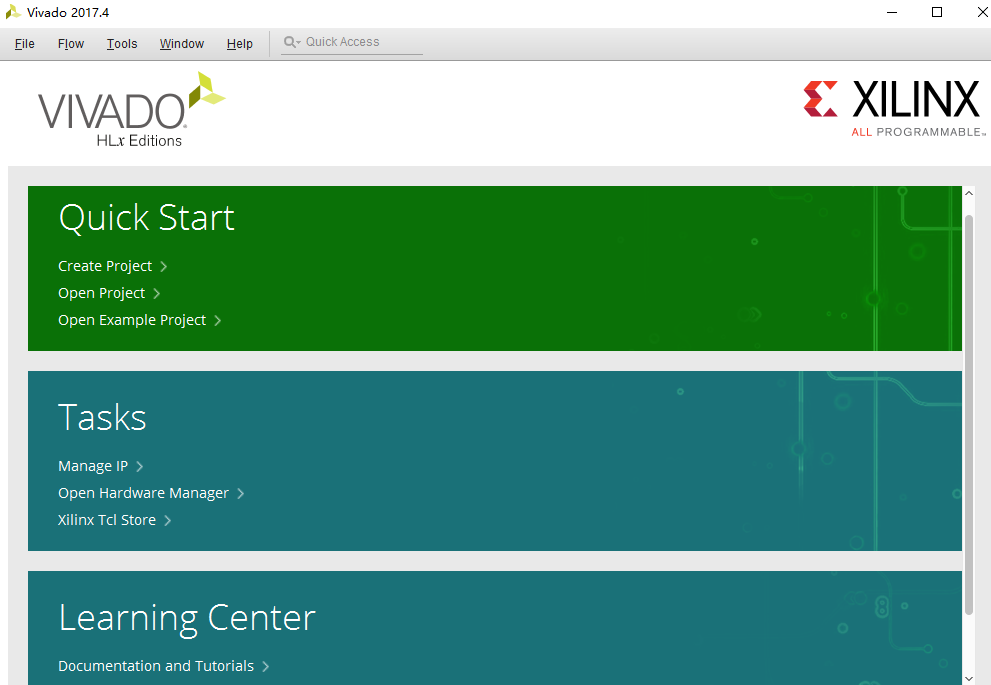


图17

2、单击上述界面中 Create New Project 图标，弹出新建工程向导，点击 Next。

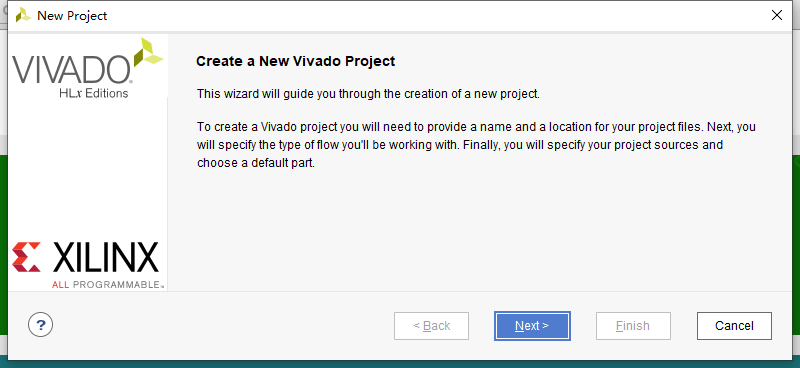


图18

3、输入工程名称、选择工程存储路径，并勾选 Create project subdirectory 选项，为工程在指定存储路径下建立独立的文件夹。设置完成后，点击 Next。

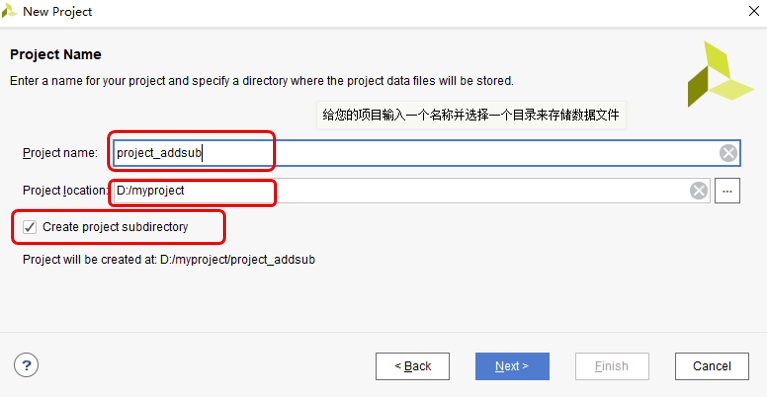
注意：工程名称和存储路径中不能出现中文和空格，建议工程名称以字母、数字、下划线来 组成。

图19

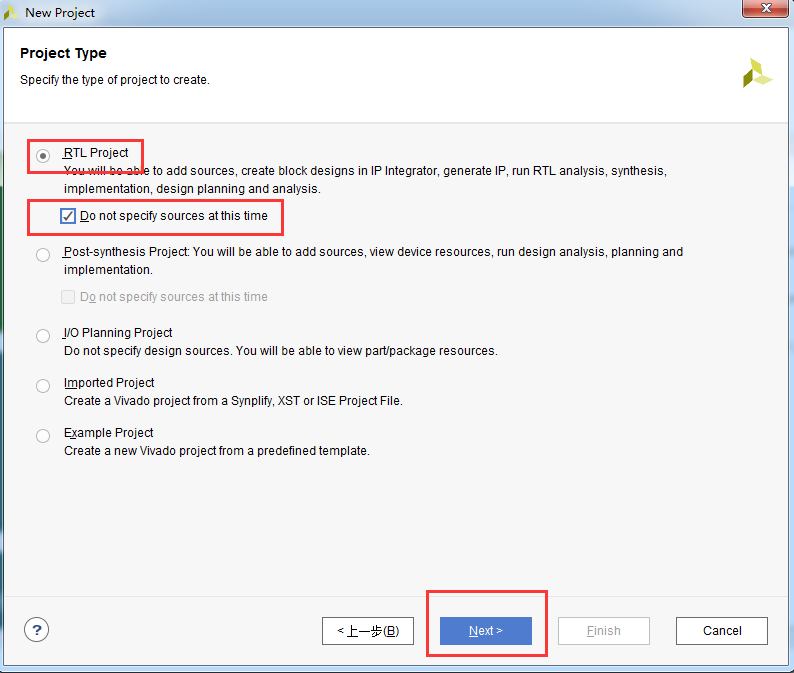
4、选择 RTL Project 一项，并勾选 Do not specify sources at this time，勾选该选项是为了跳过在新建工程的过程中添加设计源文件。点击 Next。

图20

5、根据使用的 FPGA 开发平台，选择对应的 FPGA 目标器件。 在本手册中，以 Xilinx 数模混合口袋实验室 为例，FPGA 采用 Artix-7 XC7A35T-1CSG324-C 的器件，即 Family 和 Subfamily 均为 Artix-7，封装形式（Package）为 CSG324，速度等级（Speed grade）为-1，温度等级（Temp Grade） 为 C）。点击 Next。

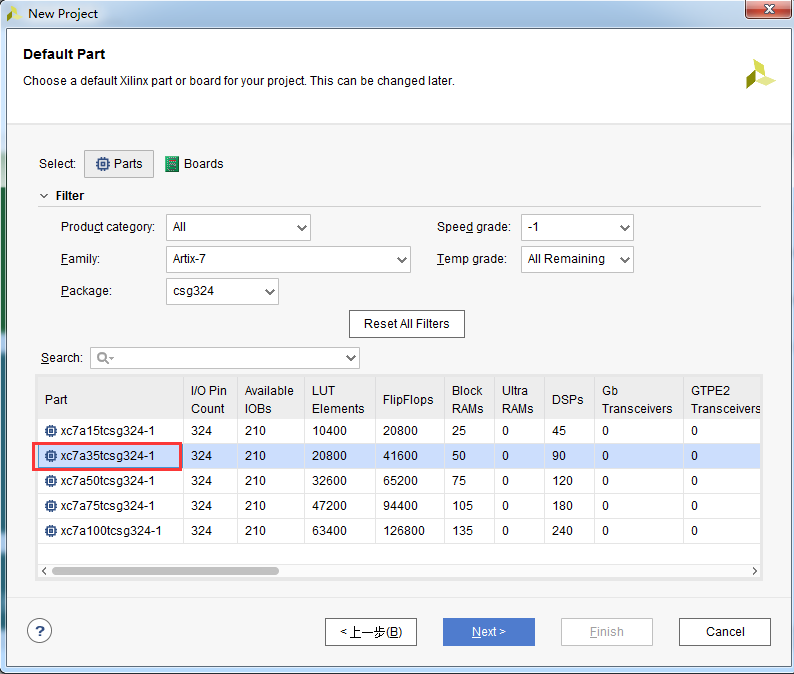


图21

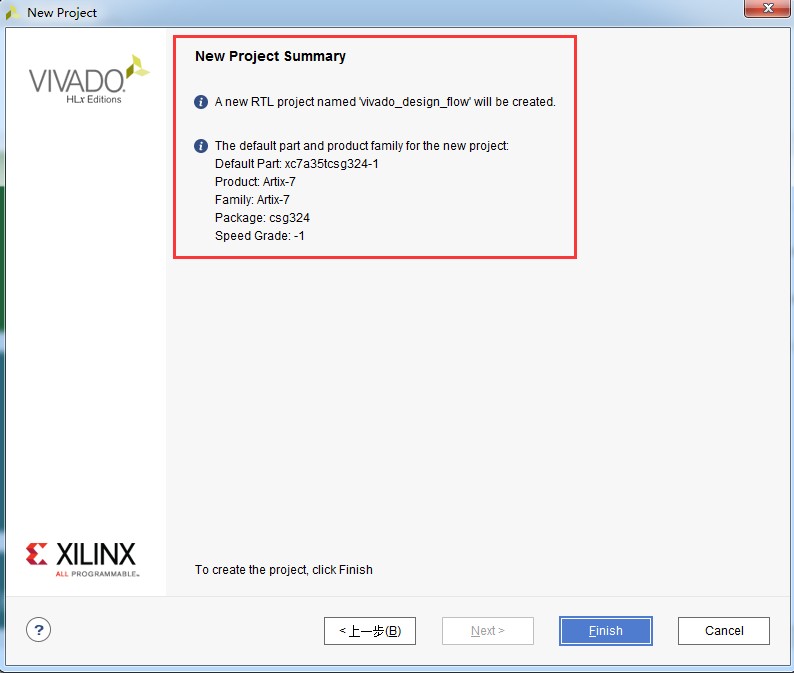
6、确认相关信息与设计所用的的 FPGA 器件信息是否一致，一致请点击 Finish，不一致，请返回上一步修改。

图22

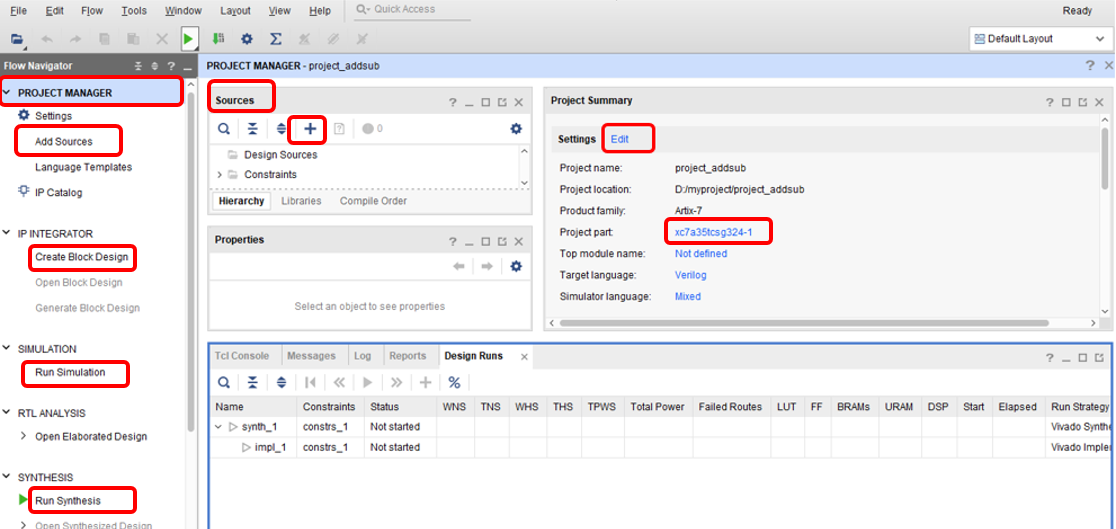
7、得到如下的空白 Vivado 工程界面，完成空白工程新建。

图23

**（二）、设计文件输入**

1、如下图所示，点击 Flow Navigator 下的 Project Manager->Add Sources 或中间 Sources 中的对话框打开设计文件导入添加对话框。

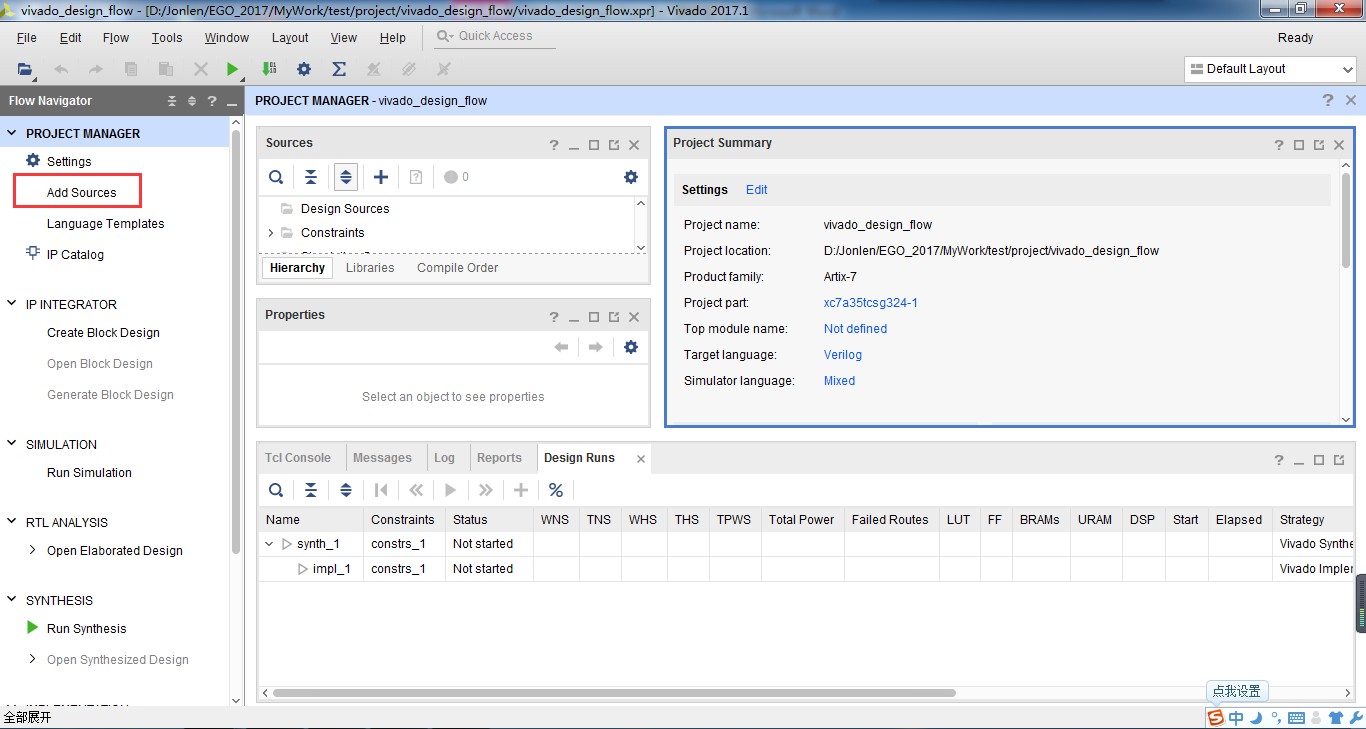


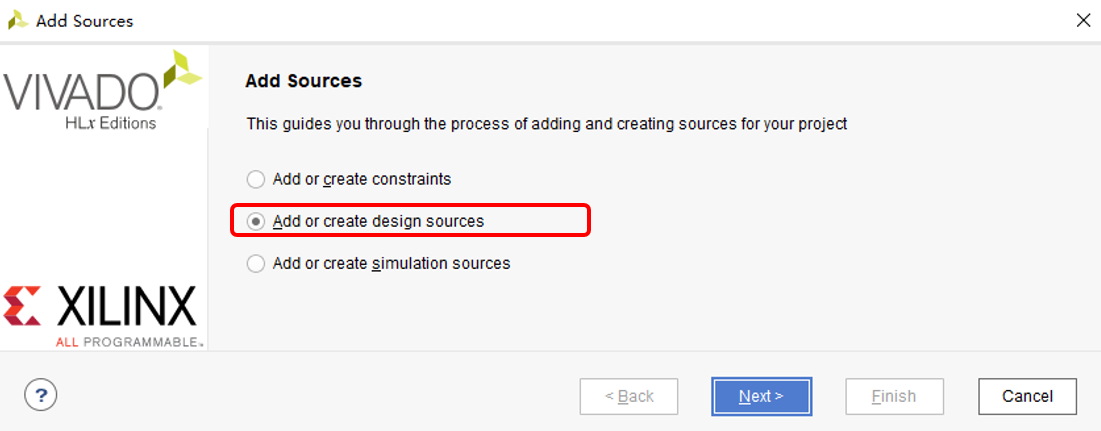
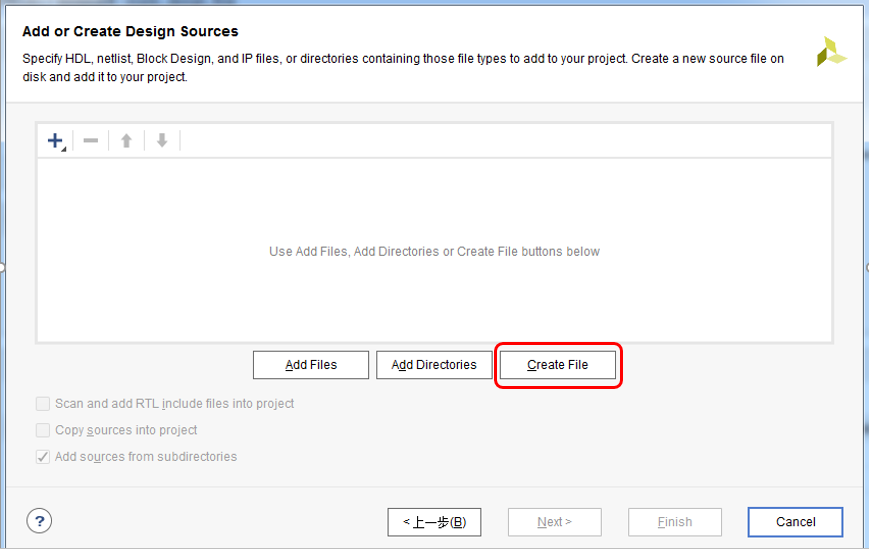
图24  
2、选择第二项 Add or Create Design Sources，用来添加或新建 Verilog 或 VHDL 源文件，点击Next。

图25

3、如果有现有的 V/VHD 文件，可以通过 Add Files 一项添加。在这里，我们要新建文件，所以选择 Create File 一项。

图26

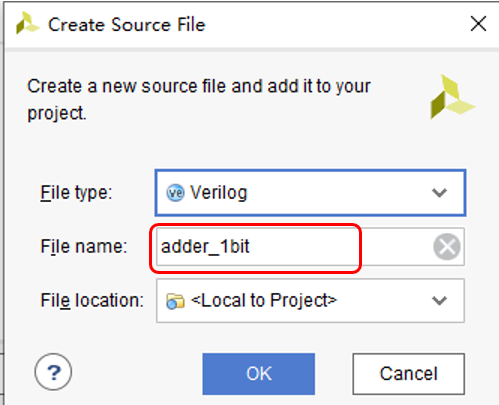
4、在 Create Source File 中输入 File Name，点击 OK。注：名称中不可出现中文和空格。

图27

5、点击 Finish。

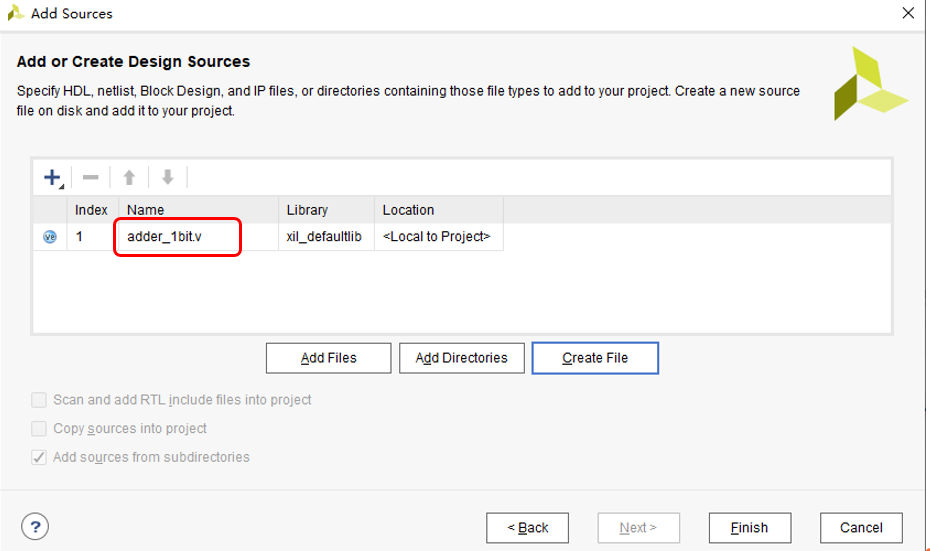
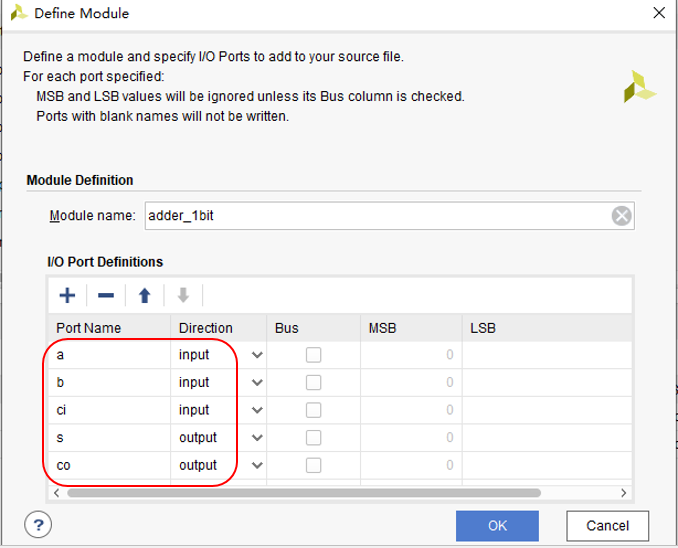


图28

6、在弹出的 Define Module 中的 I/O Port Definition，输入设计模块所需的端口，并设置端口方向，如果端口为总线型，勾选 Bus 选项，并通过 MSB 和 LSB 确定总线宽度。完成后点 OK。图29

注：输入值和输出值，实际宽度与代码中一致，也可在代码中修改。

7、新建的设计文件（此处为 adder\_1bit.v）即存在于 Sources 中的 Design Sources 中。双击

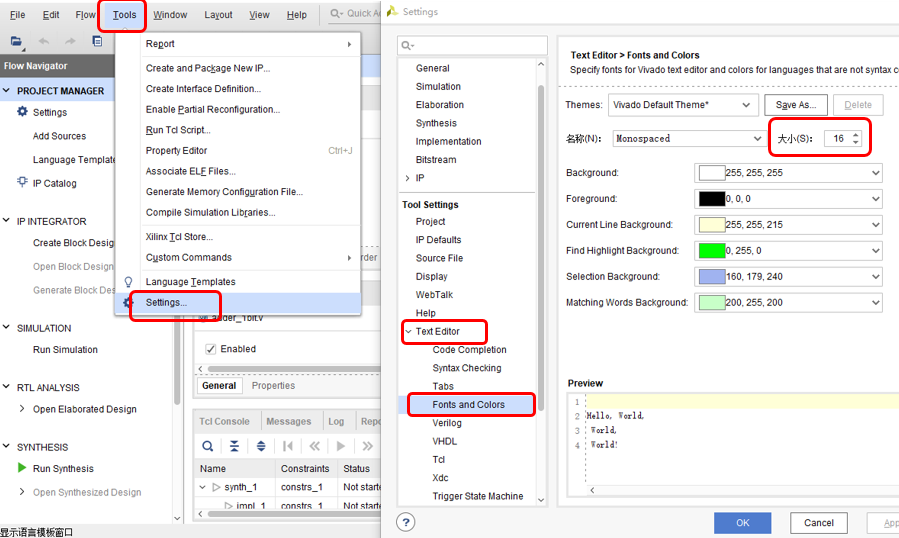
打开该文件，输入相应的设计代码。

图30

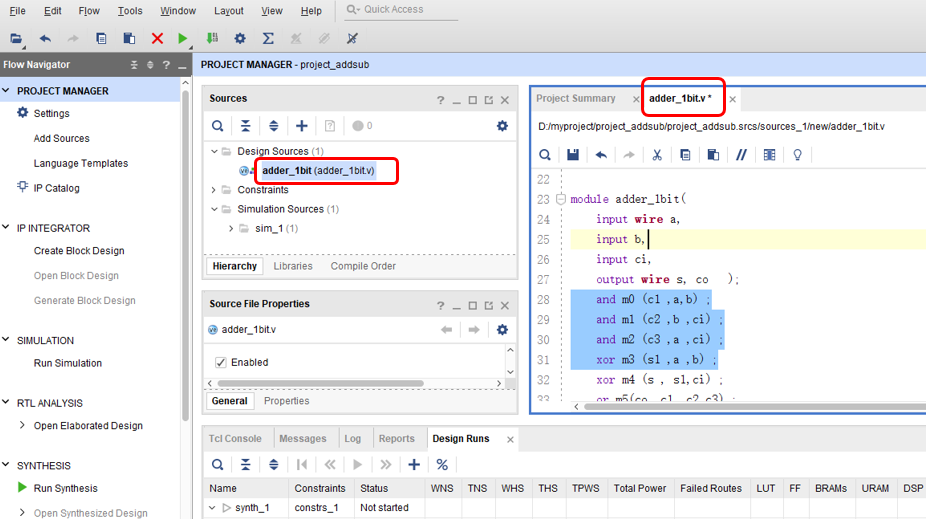


图31

设计文件adder\_1bit.v中的关键的verilog代码如下：

`timescale 1ns / 1ps

module adder\_1bit( input wire a, input b, input ci, output wire s, co );

and m0 (c1 ,a,b) ;

and m1 (c2 ,b ,ci) ;

and m2 (c3 ,a ,ci) ;

xor m3 (s1 ,a ,b) ;

xor m4 (s , s1,ci) ;

or m5(co ,c1 ,c2,c3) ;

endmodule

**（三）、利用 Vivado 进行功能仿真**

1.创建激励测试文件，在 Source 中右击选择 Add Source

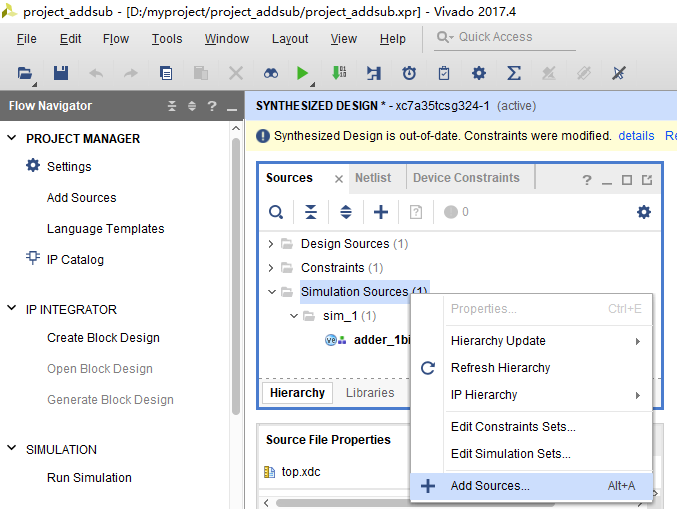


图32

2、在 Add Source 界面中选择第三项 Add or Create Simulation Source,点击 Next。

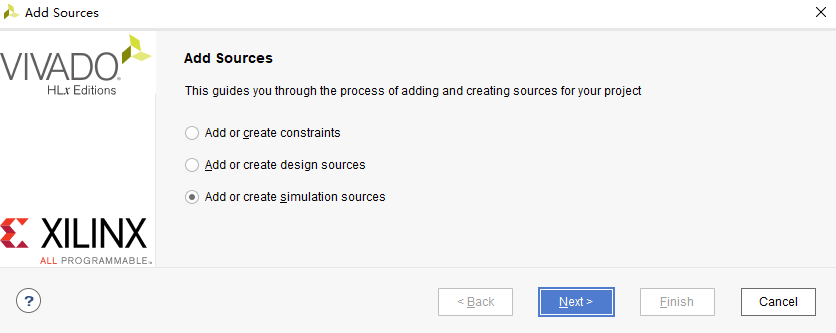


图33

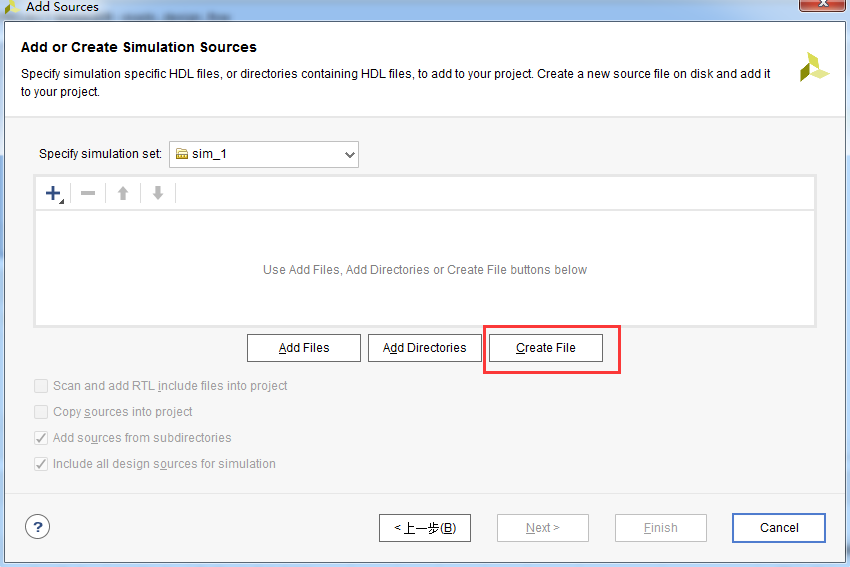
3、选择Create File 创建一个仿真激励文件

图34

4、输入激励文件名称，点击 OK。



图35

5、确认添加完成之后点击 Finish，因为是激励文件不需要对外端口，所以直接Port 部分直接空着，点击ok。

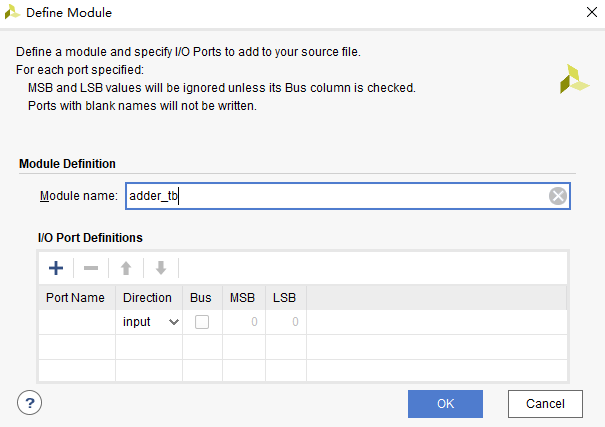


图36

5、在 Source 下双击打开空白的激励测试文件，完成对将要仿真的 module 的实例化和激励代码的编写，如下图和下述代码所示。

`timescale 1ns / 1ps module adder\_tb( );

reg a; reg b; reg ci;

wire s;

wire co;

adder\_1bit u0( .a(a), .b(b), .ci(ci) , .s(s), .co(co) );

initial

begin

a = 1'b0;

b = 1'b0;

ci = 1'b0;

#100;

a = 1'b0;

b = 1'b0;

ci = 1'b1;

#100;

a = 1'b0;

b = 1'b1;

ci = 1'b0;

#100;

a = 1'b0;

b = 1'b1;

ci = 1'b1;

#100;

a = 1'b1;

b = 1'b0;

ci = 1'b0;

#100;

a = 1'b1;

b = 1'b0;

ci = 1'b1;

#100;

a = 1'b1;

b = 1'b1;

ci = 1'b0;

#100;

a = 1'b1;

b = 1'b1;

ci = 1'b1;

#100;

a = 1'b0;

b = 1'b0;

ci = 1'b0;

#100;

end

endmodule

激励文件完成之后，工程目录如下图所示。

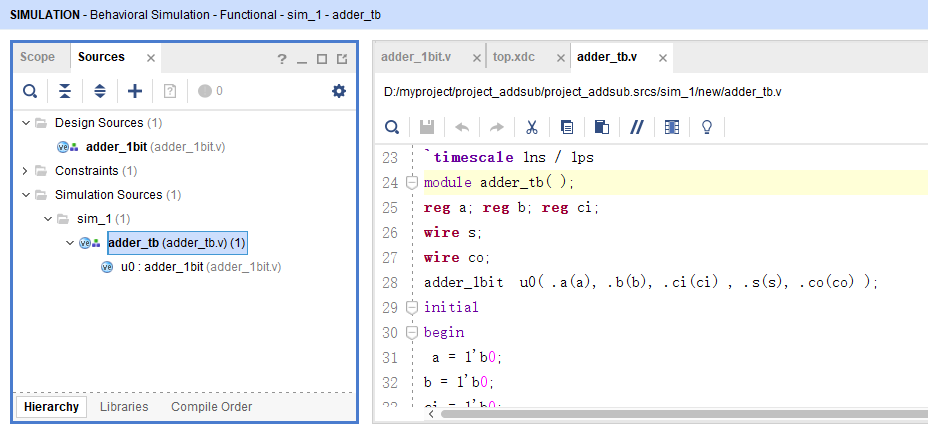


图37

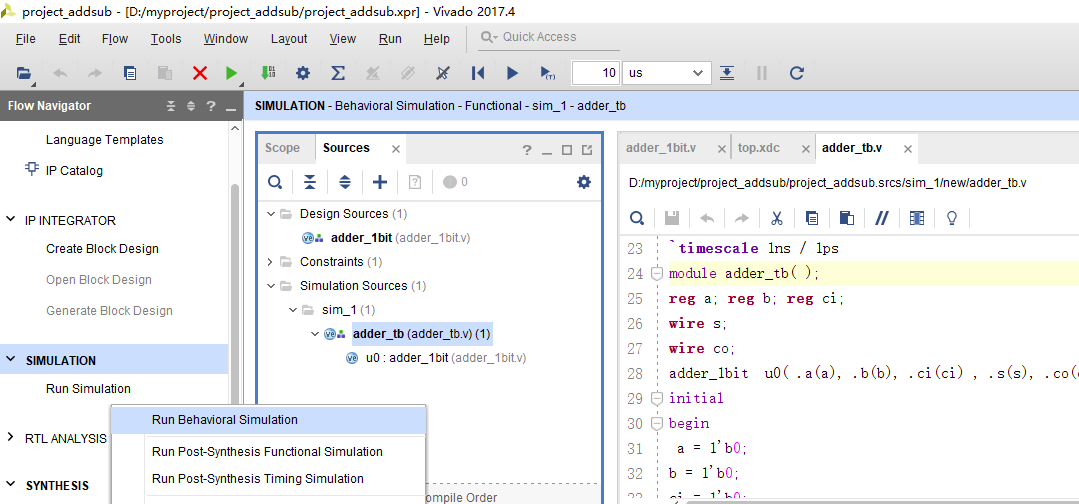
6、此时，进入仿真。在左侧 Flow Navigator 中点击 Simulation 下的 Run Simulation 选项，并选择 Run Behavioral Simulation 一项，进入仿真界面。

图38

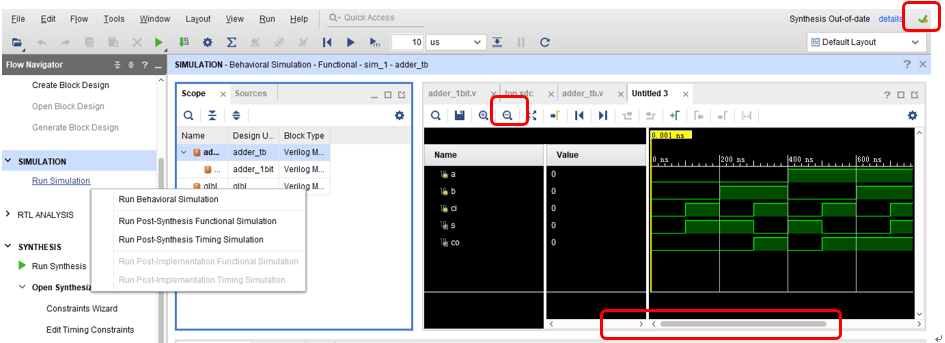
7、下图所示为仿真界面。

图39

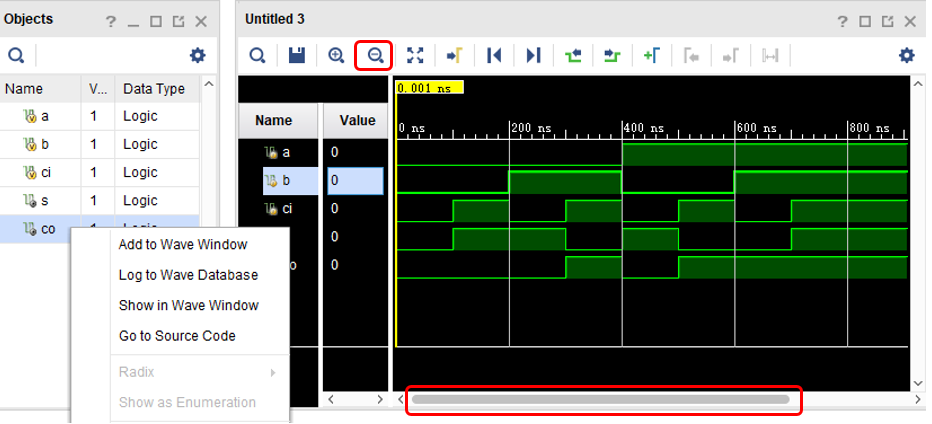
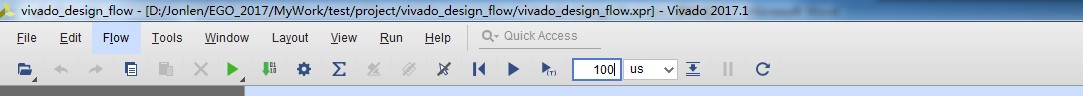
可通过左侧 Scope 一栏中的目录结构定位到设计者想要查看的 module 内部寄存器，在 Objects 对应的信号名称上右击选择 Add To Wave Window，将信号加入波形图中。因为窗口已有信号， 此操作不需要进行。 

图40

可通过选择工具栏中的如下选项来进行波形的仿真时间控制。如下工具条，分别是复位波形（即清空现有波形）、运行仿真、运行特定时长的仿真、仿真时长设置、仿真时长单位、单步运行、暂停……

8、最终得到的仿真效果图如下。核对波形与预设的逻辑功能是否一致。仿真完成。

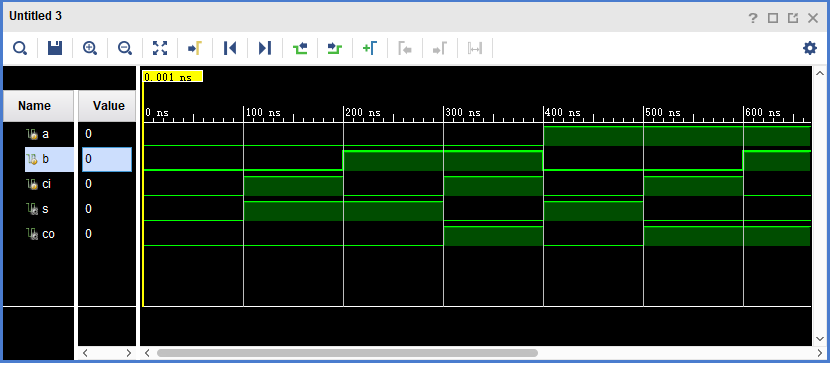


图41

**（四）、添加引脚约束**

添加引脚约束文件，有两种方法可以添加约束文件，一是可利用 Vivado 中 IO planning 功能，二是可以直接新建 XDC 的约束文件，手动输入约束命令。

此处建议：方法一了解。实验时采用方法二进行引脚约束。只要采用两种方法中的一种方法进行引脚约束就可以了。如果两种约束都采用，出现约束冲突的话，系统会报错。

1. **先来看第一种方法，利用 IO planning进行引脚约束（建议了解即可）。**

a.1、点击 Flow Navigator 中 Synthesis 中的 Run Synthesis，先对工程进行综合。此种方法只有在进行了综合后，才能出现相应的菜单，才能进行引脚约束。所以要先综合，然后引脚约束。后面的方法二就不需要。在进行综合时，如果电路文件比较大，综合时间需要1分钟左右的时间，注意观察屏幕右上方，如果有绿色的小圆圈在旋转，表示系统正在进行综合，请等待。等屏幕右上方的绿色小圆圈变成对号时，表示综合完成了。如下图红框所示。其实，后面的实现环节和生成bit流文件等操作，都需要等待。注意屏幕右上方拐角处的绿色小圆圈变成对号时，才表示对应的环节完成，可以进入下一个环节了。

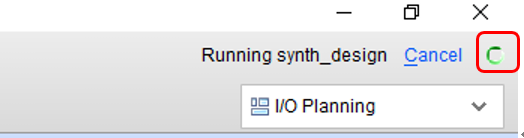
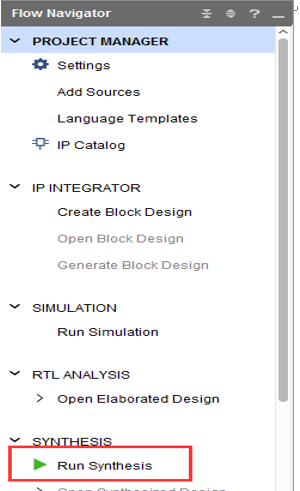


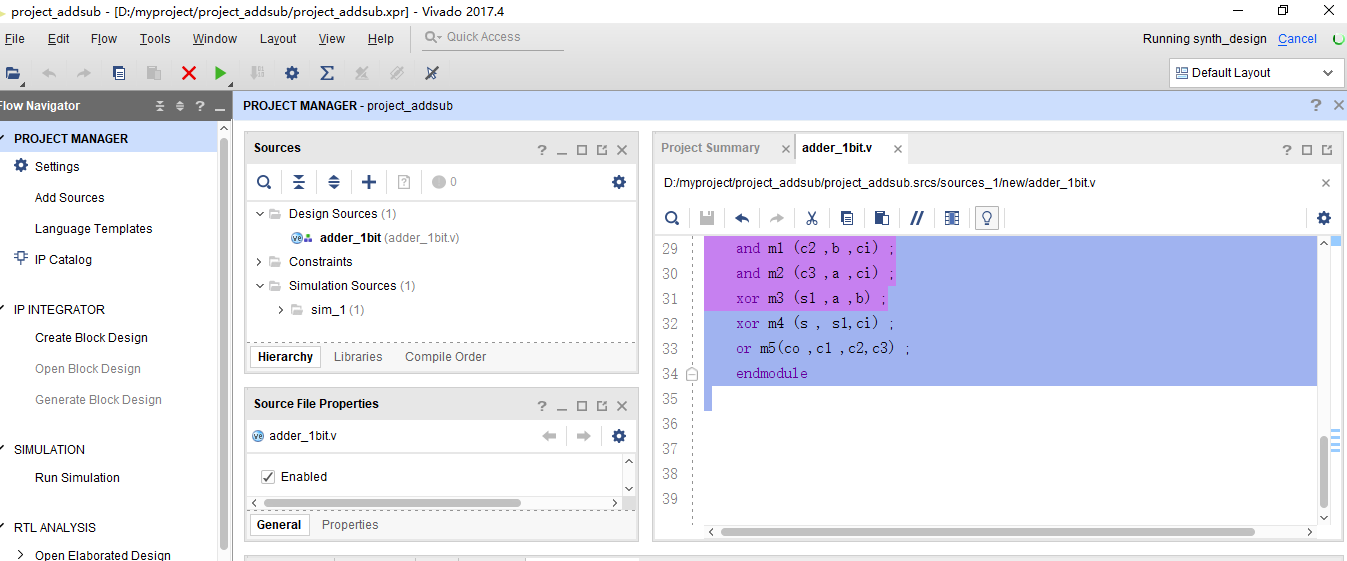
图42

图43

a.2、综合完成之后，选择 Open Synthesized Design，打开综合结果。

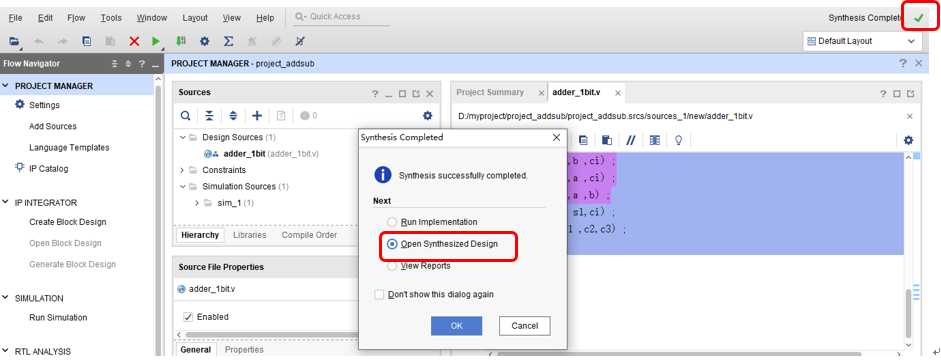


图44

a.3、此时应看到如下界面，如果没出现如下界面，在图示位置layout 中选择 IO Ports 一项。

图45

a.4、在右下方的选项卡中切换到 I/O ports 一栏，并在对应的信号后，输入对应的 FPGA 管脚标号（或将信号拖拽到右上方 Package 图中对应的管脚上），并指定 I/O std。 具体的 FPGA 约束管脚和 IO 电平标准，可参考对应板卡的用户手册或原理图）。

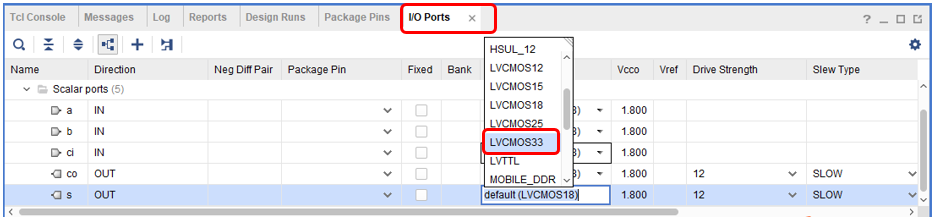


图46

a.5、完成之后，点击左上方工具栏中的保存按钮，工程提示新建 XDC 文件或选择工程中已有的 XDC 文件。在这里，我们要 Create a new file，输入 File name，点击 OK 完成约束过程。

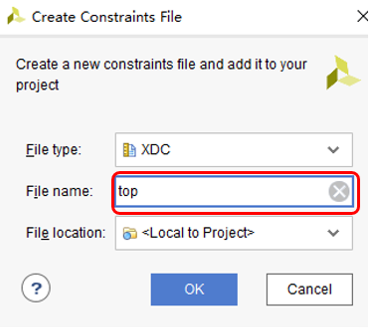


图47

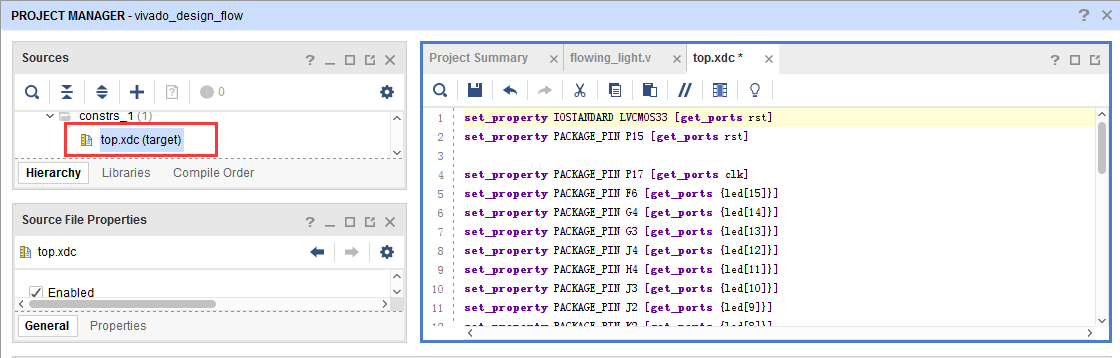
a.6、此时，在 Sources 下 Constraints 中会找到新建的 XDC 文件。

图48

1. **利用第二种方法添加约束文件。（建议采用此方法）**

b.1、点击 Add Sources，选择第一项 Add or Create Constraints 一项，点击 Next。

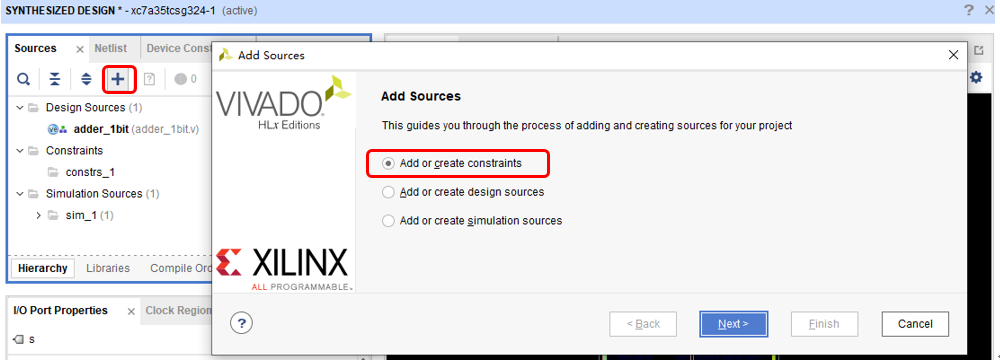


图 49

b.2、点击 Create File，新建一个 XDC 文件，输入 XDC 文件名，点击 OK。点击 Finish。

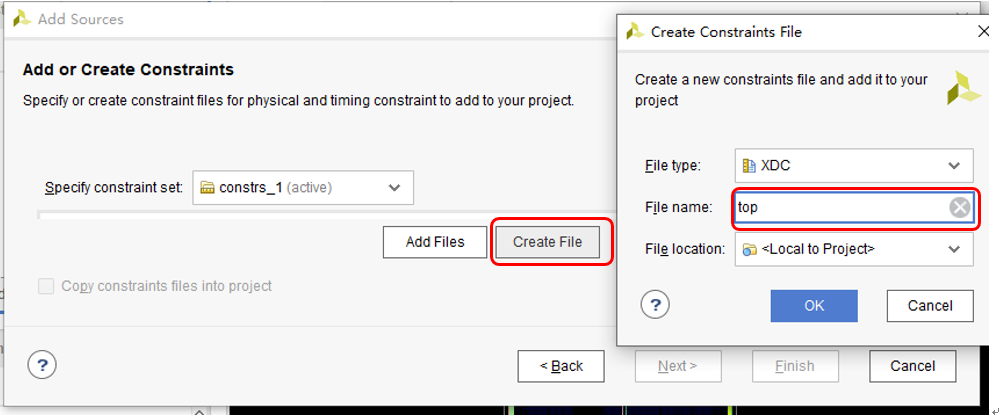


图50

b.3、双击打开新建好的 XDC 文件，并按照如下规则，输入相应的 FPGA 管脚约束信息和电平标准。

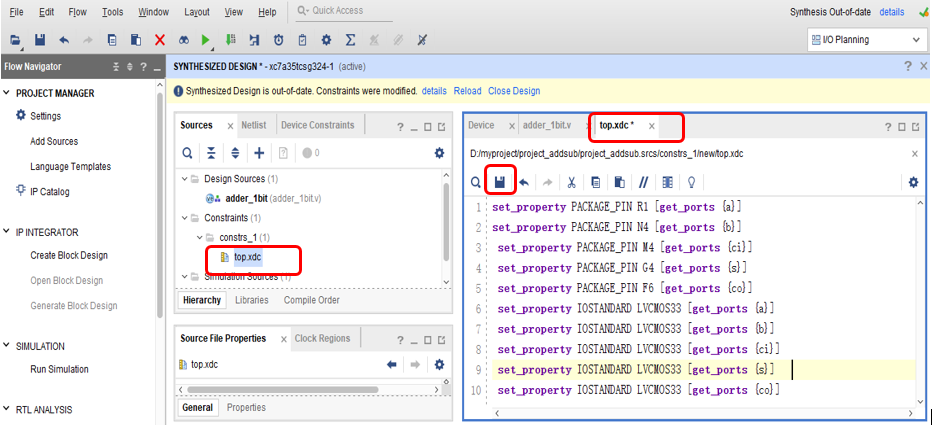
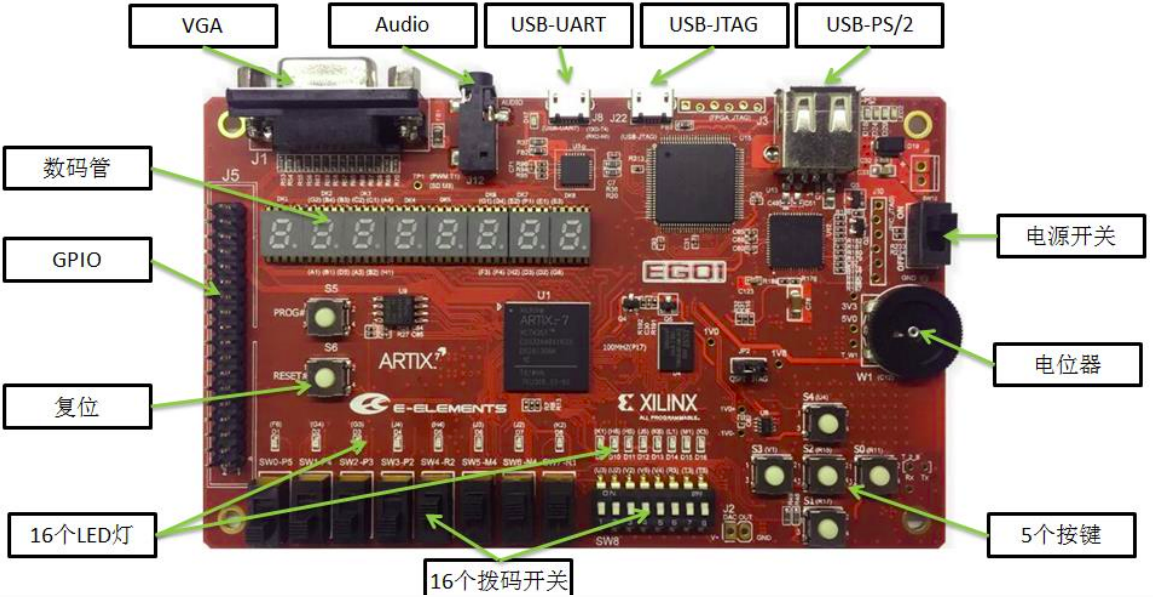
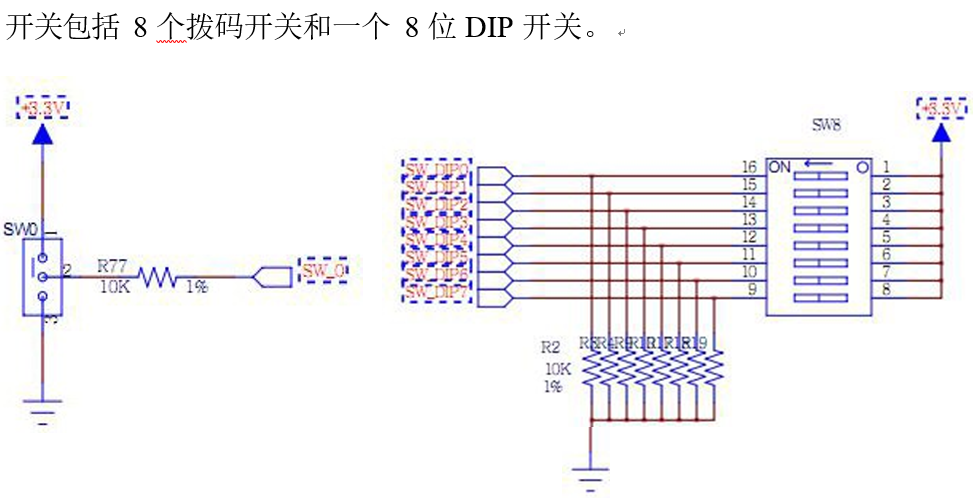
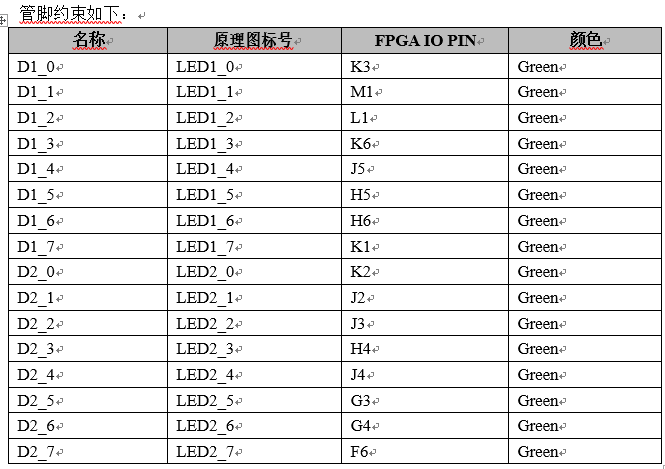
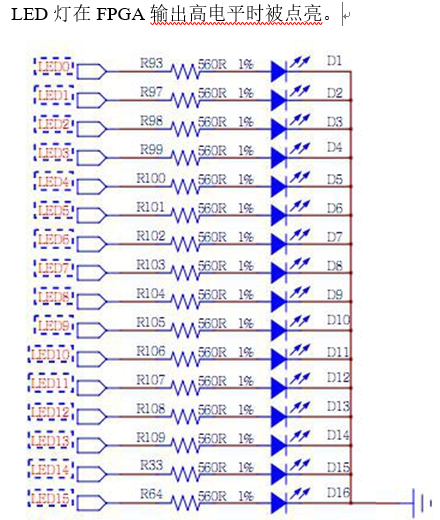


图51

图52

图53图54

图55

根据上面的与开发板有关的内容介绍，写出对应的约束文件。这个约束文件表示将输入信号a、b、ci和输出信号s、co绑定到FPGA可编程芯片对应的引脚上，分别绑定到引脚R1、N4、M4、G4、F6上。根据开发板的资料，可以看出FPGA引脚R1、N4、M4连接了开发板上的开关。FPGA引脚G4、F6连接了开发板上的LED灯。测试时，我们通过对应的开关输入信号进入FPGA内部电路。FPGA内部电路的输出通过LED灯进行显示。

set\_property PACKAGE\_PIN R1 [get\_ports {a}]

set\_property PACKAGE\_PIN N4 [get\_ports {b}]

set\_property PACKAGE\_PIN M4 [get\_ports {ci}]

set\_property PACKAGE\_PIN G4 [get\_ports {s}]

set\_property PACKAGE\_PIN F6 [get\_ports {co}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b}]

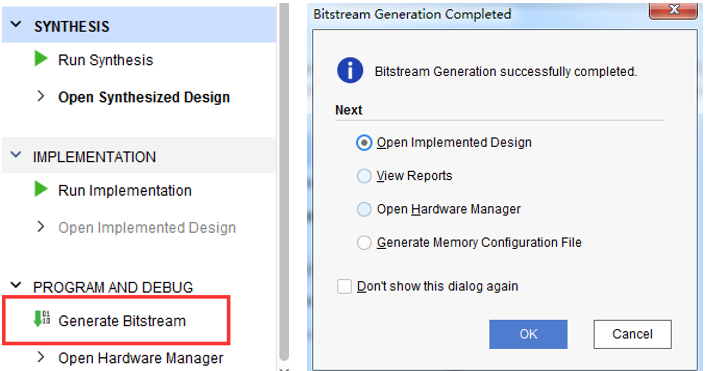
set\_property IOSTANDARD LVCMOS33 [get\_ports {ci}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {co}]

**（五）、工程实现**

1、在 Flow Navigator 中点击 Program and Debug 下的 Generate Bitstream 选项，工程会自动完成综合、实现、Bit 文件生成过程，完成之后，可点击 Open Implemented Design 来查看工程实现结果。

图56

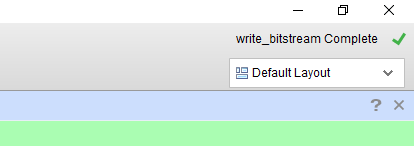
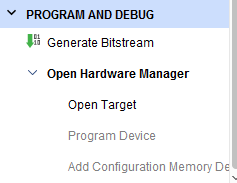
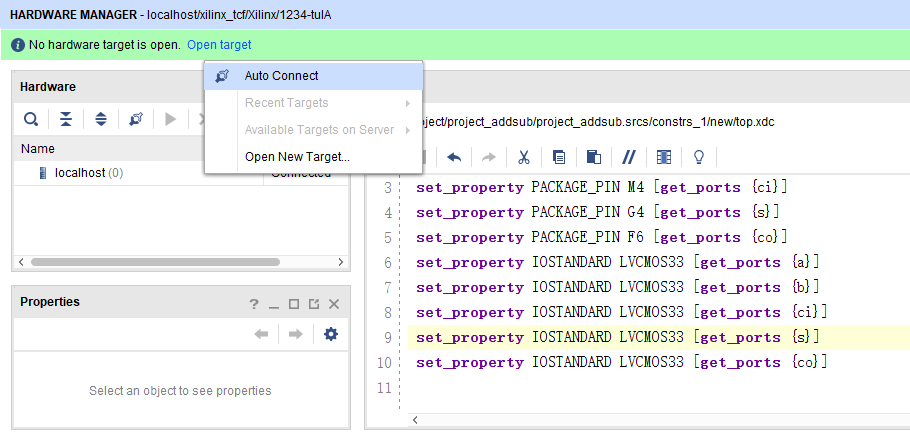


图57

2、点击 Flow Navigator 中 Open Hardware Manager 一项，进入硬件编程管理界面。

图58

3、在提示的信息中，选择 Open Hardware Manager（或在 Flow Navigator 中展开 Hardware Manager，点击 Open Target）。选择Auto Connect连接到板卡。

图59

4、 连接成功后，在目标芯片上右击，选择 “Program Device”。在弹出的对话框中 “Bitstream

File” 一栏已经自动加载本工程生成的比特流文件，点击 “Program” 对 FPGA 芯片进行编程。

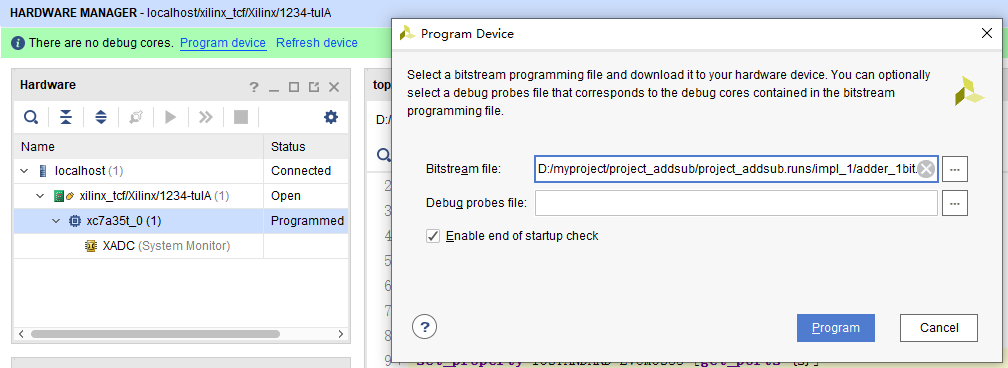
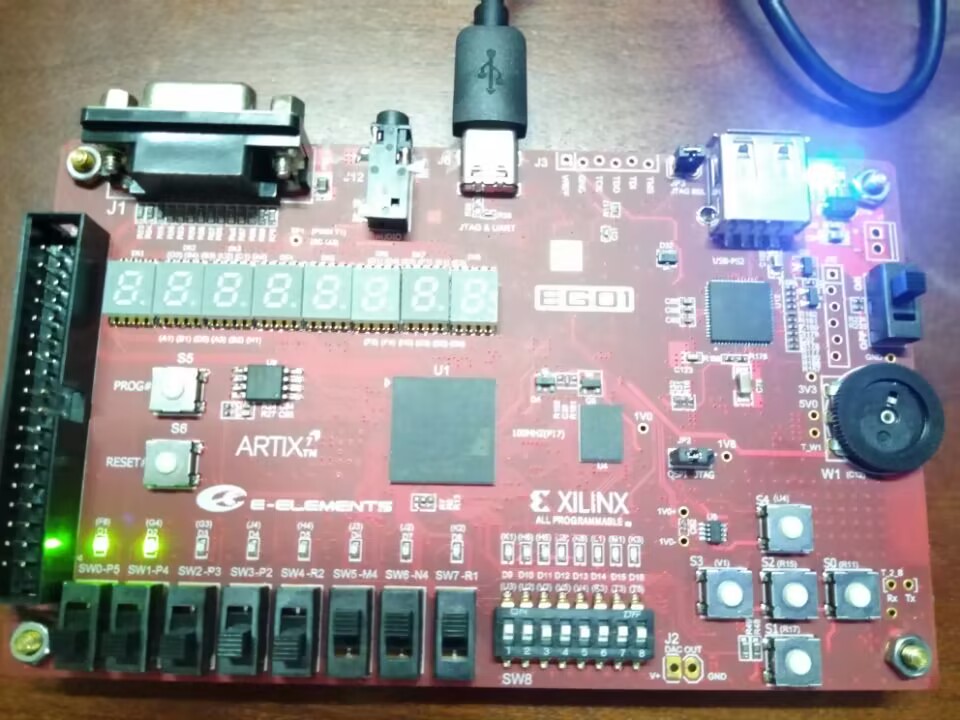
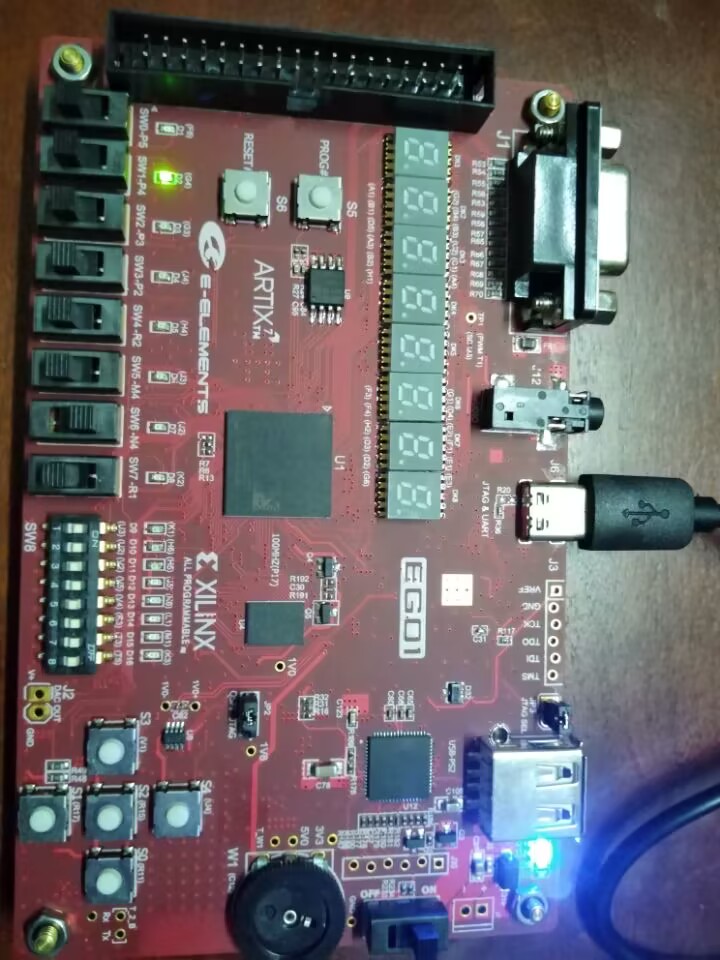


图60

5、下载完成后，在板子上观察实验结果。拨动下方的开关，看到开发板左方的两个led灯的亮灭在变化。开关的输入与led灯的输出规律符合一位加法器的运算特点。

图61

图62

**（六）采用四个一位的全加器和附加逻辑设计两个四位数的加/减法器。**

采用四个一位的全加器和附加逻辑，根据图10设计两个四位数的加/减法器。要求编写与此电路对应的verilog代码。编写激励输入文件，激励文件要求放入实验报告。进行仿真，仿真截图放入实验报告。生成比特流文件，进行下载，下载后进行运行，只需要一张运行照片放入实验报告。

1、请在下面补充与图10对应的verilog代码。

module addsub\_4bit（……）;

请在此处补充对应的verilog代码，要求在实验前完成。进实验室时会检查。

endmodule

2、补充下面仿真时需要的激励输入文件：

module addsub\_4bit\_tb（……）;

请在此处补充对应的verilog代码，要求在实验前完成。进实验室时会检查。

endmodule

**（七）采用与逻辑、或逻辑和数据选择器设计两个四位数的加/减法器**

根据图15，采用与逻辑、或逻辑和四选一数据选择器设计两个四位数的加/减法器。要求编写与此电路对应的verilog代码。编写激励输入文件，激励文件要求放入实验报告。进行仿真，仿真截图放入实验报告。生成比特流文件，进行下载，下载后进行运行，只需要一张运行照片放入实验报告。

1、请在下面补充与图15对应的verilog代码。

module addsub\_4bitmux（……）;

请在此处补充对应的verilog代码，要求在实验前完成。进实验室时会检查。

endmodule

2、补充下面仿真时需要的激励输入文件：

module addsub\_4bitmux\_tb（……）;

请在此处补充对应的verilog代码，要求在实验前完成。进实验室时会检查。

endmodule

**六、实验报告要求和思考**

1、本次实验需要写入实验报告。

2、每张截图不要超过1M。对于超过1M的照片,通过QQ截图等方式进行照片浏览截图，让每张照片截图后的体积降到1M以下。

3、要求每个实验的实验报告内容，包含所有的截图在内尽量不要超过5M。实验五到实验八放在一个word文档内（不要超过20M）进行提交。也可以提交PDF格式。

4、实验报告要求简明扼要，关键的verilog设计代码、激励输入、仿真截图和下载后的运行照片这四样不能少，其它的内容简单介绍就可以了（有实验报告模板文档）。

5、思考：如果要求将2个四位数运算的结果，在LED灯上显示的同时，运算结果同时送给开发板上方的2个七段显示器来显示对应的十进制值，verilog代码需要怎么怎么修改。引脚约束文件应该怎么写。

6、思考：如果设计四位运算器的溢出判断电路，应该怎么设计？

**七、附录说明：**

由于vivado中不支持直接输入原理图，可以通过将上述的原理图映射为verilog代码的方式创建 Vivado 中verilog代码文件，然后进行综合。例如：

一位加法器的verilog设计代码可以写成下面的样式：

module adder\_1bit(input wire a, b, ci ,output wire s, co);

and m0 (c1 ,a,b) ;

and m1 (c2 ,b ,ci) ;

and m2 (c3 ,a ,ci) ;

xor m3 (s1 ,a ,b) ;

xor m4 (s , s1,ci) ;

or m5(co ,c1 ,c2,c3) ;

endmodule

如果进行仿真，每个同学自己的激励输入文件怎么编写。可以将自己的激励输入文件和对应的仿真截图（通过截屏的方式获得），放到实验报告中。

除了结构描述法外，一位加法器也可以采用verilog代码中的数据流法进行描述，代码类似为：

module Add1b(

input a,

input b,

input ci,

output s,

output co

);

assign s = a ^ b ^ ci;

assign co = (a&b) | (a&ci) | (b&ci);

endmodule

加/减法器也可以采用verilog代码中的数据流法进行描述，代码类似为：

module AddSub #( parameter WIDTH = 8 )

// WIDTH should be greater than 1

(

input [WIDTH-1:0] A,

input [WIDTH-1:0] B,

input Ctrl,

output[WIDTH-1:0] S,

output Co

);

wire[WIDTH:0] C\_temp;

assign C\_temp[0] = Ctrl;

wire[WIDTH-1:0] S\_temp;

wire[WIDTH-1:0] Bo;

assign Bo = {WIDTH{Ctrl}} ^ B;

genvar i;

generate

for(i = 0; i < WIDTH; i = i+1) begin

Add1b u0 ( .a(A[i]), .b(Bo[i]), .ci(C\_temp[i]), .s(S\_temp[i]), .co(C\_temp[i+1]) );

end

endgenerate

assign Co = 1'b0 == Ctrl ? C\_temp[WIDTH] : ~C\_temp[WIDTH]; // deal with carry bit when subtracting

assign S = S\_temp;

endmodule

对应的测试输入文件可以写成类似这样：

`timescale 1ps/1ps

module AddSub\_tb();

reg[7:0] A;

reg[7:0] B;

reg Ctrl;

wire[7:0] S;

wire Co;

AddSub #(.WIDTH(8)) m0 (.A(A), .B(B), .Ctrl(Ctrl), .S(S), .Co(Co));

initial begin

$dumpfile("AddSub.vcd");

$dumpvars(1, AddSub\_tb);

A = 0; B = 0; Ctrl = 0;

#10 A = 8'b0000\_0010; B = 8'b0000\_0100;

#10 A = 8'b1000\_0001; B = 8'b0111\_1111;

#10 Ctrl = 1;

#10 A = 8'b0010\_0000; B = 8'b0000\_1100;

#10 A = 8'b0000\_0001; B = 8'b1111\_1111;

#10;

end

endmodule