互联网+班
《计算机组成与结构》课程
综合实验手册（V1.1 版）

Update: 2022-10-21

目 录

1　实验平台的安装和使用

1.1实验平台的安装

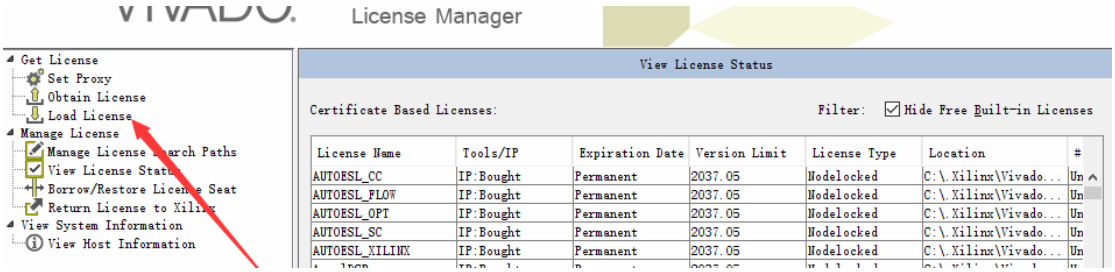
Step1：解压位于 XXX 的安装文件，文件很大，有 20G，需耐心等待。

Step2：在解压后的文件夹根目录中找到 Xsetup.exe 文件双击进行进行安装，安装时间依赖电脑性能，估计在半小时左右。有部分需要选择的地方按照如下所示的图片进行选择，没有图片的地方按照默认选择。

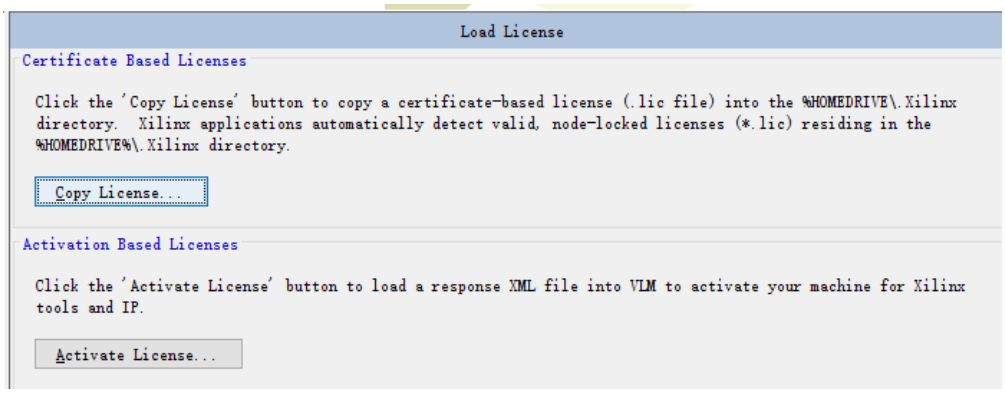
Step3：安装完成后会弹出 vivado License Manager 的程序，如下图所示。

如果没有自动弹出此程序，请在开始菜单->所有程序中选择这个程序打开。

Step4：加载证书文件。



双击如图箭头处的 Load License，会出现如下界面。打开 Copy License，选择在 XXXX 的 lic 文件，确定，就完成了实验平台的安装。



1.2实验平台的使用

在 vivado 中，有两种文件，一种是设计文件，一种是仿真文件，通过给设计文件加上一定的激励，就得到了仿真文件，仿真文件用于对电路进行仿真。
设计文件模块的编写如下：

```
module   设计模块名称(变量 1，变量 2，变量 3,.....);
    input  变量 1,变量 2.....;
   output  变量 m.....;
    语句 1;
    语句 2;
    .....
endmodule
```

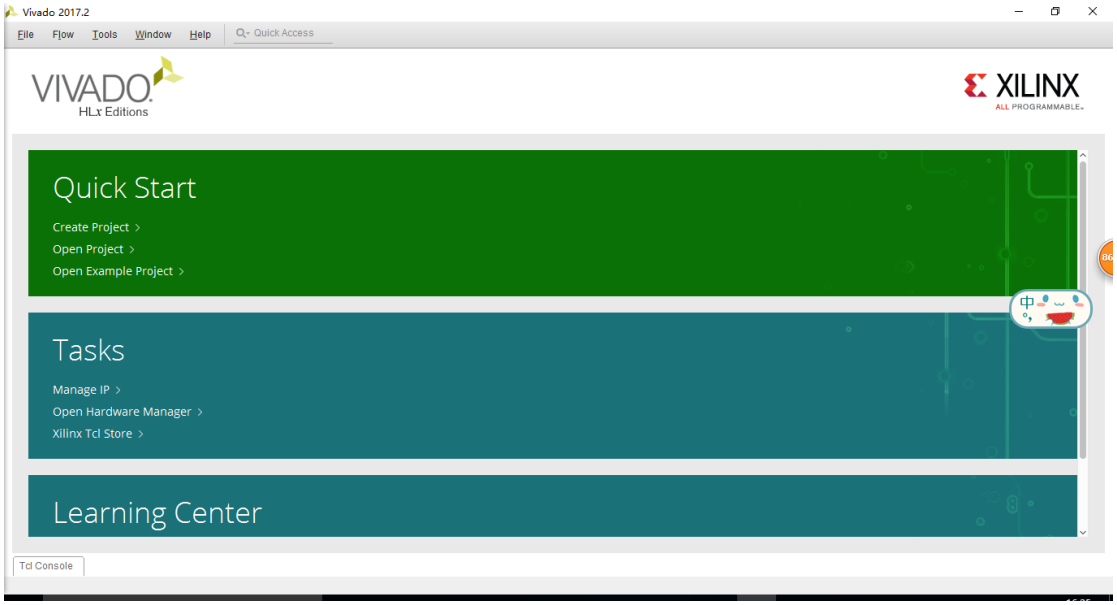
仿真文件模块的编写如下：

```
module  仿真文件模块名称 ;
    // Inputs
    变量类型  变量 1;
    变量类型  变量 2;
    .....
    // Outputs
    变量类型  变量 m;
    .....
    // Instantiate the Unit Under Test (UUT)
        设计模块类型   模块变量名(
            .形参 1（实参 1）,
            .....
            .形参 i(实参 i)      );
        语句 1；
        语句 2；
        .....
endmodule
```

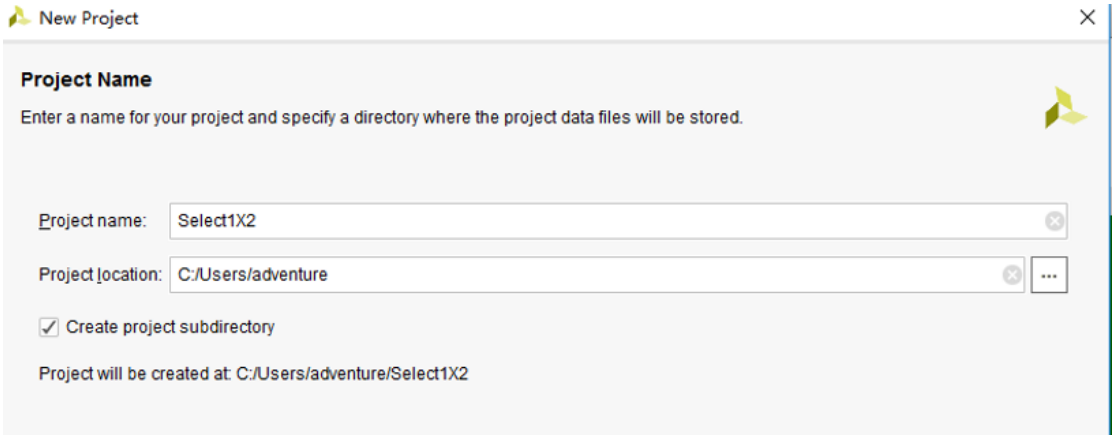
其中，仿真文件中使用设计文件，即绿色部分，如下：

door uut (.x(x),.y(y),.z(z),.f(f));

- Step1：打开 vivado 软件（注意是 vivado，不是 vivado HTL），会进入到如图所示界面。

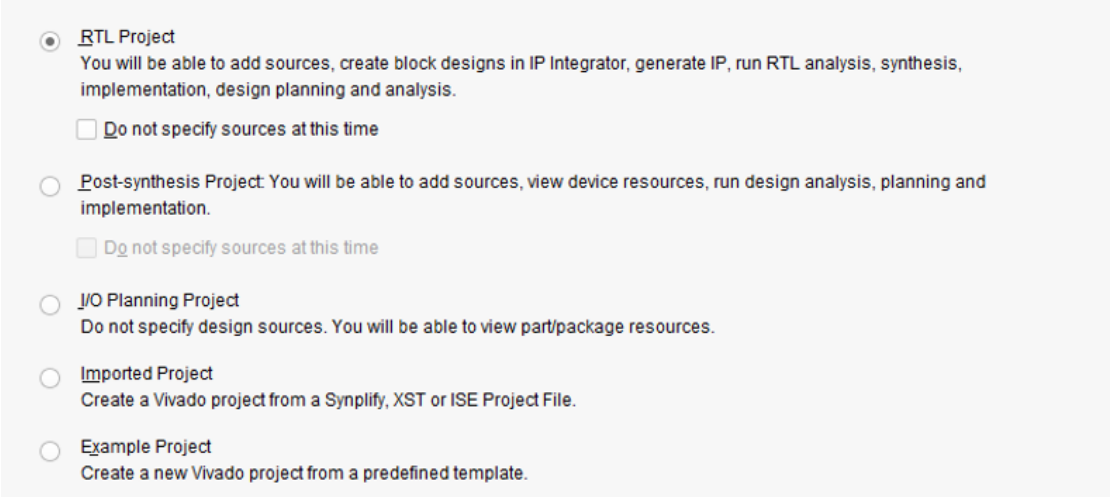


- Step2：建立工程，点击 Quick Start 的 Create Project 按钮，可以快速创建工程。基本所有的设置都可以使用默认的设置。快速建立工程的基本步骤是：命名工程，选择工程类别，选择仿真版的型号，最后建立工程。命名工程如下图所示：



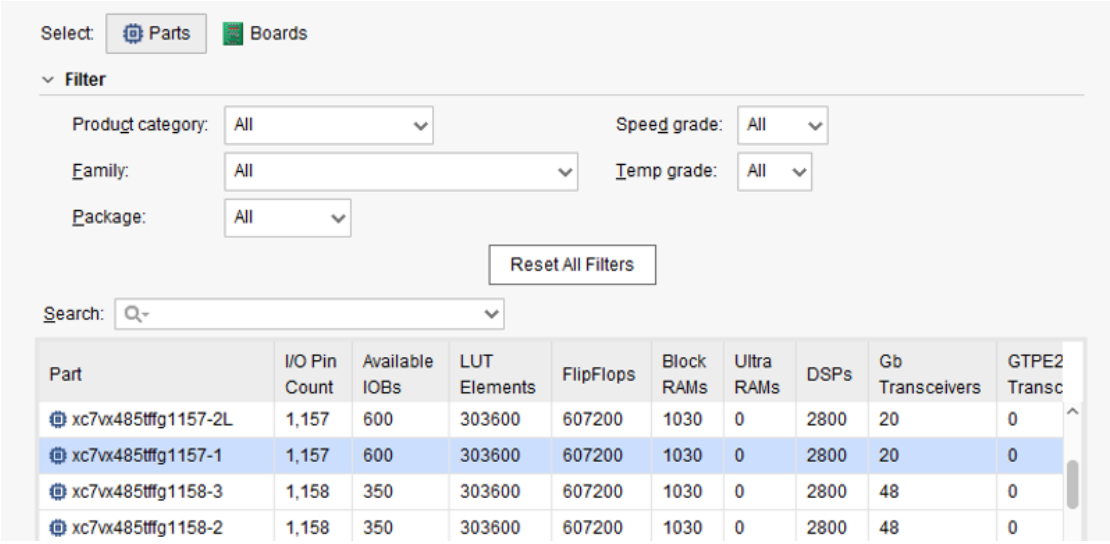
在“Create project subdirectory”处勾选，就表示在已选位置基础上创建一个子目录，不勾就表示不创建。一般情况下按大类来分，比如米尔的文件夹，zingsk 的文件夹，zybo 的文件夹等，所以最好是勾选上。

选择工程类别时，选择 RTL 项目，如下图所示：



最好勾选 RTL Project 下的 Do not specify sources at this time，勾选之后可以在创建工程时跳过创建源文件的过程，加快创建速度。如果不勾选，就会进入具体设置，有硬件语言的类型，ip 的选择等等。此处建议勾选，因为这些可以在工程中设置，没有必要提前设置。

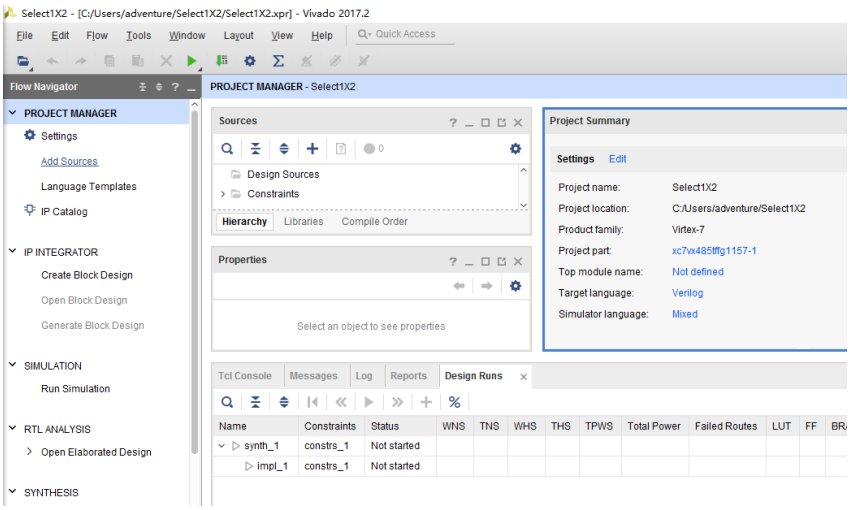
最后是选择仿真版的型号。在不需要仿真版时可以随便选一个，之后用到仿真版的时候再进行修改。选择仿真版的界面如下图所示：



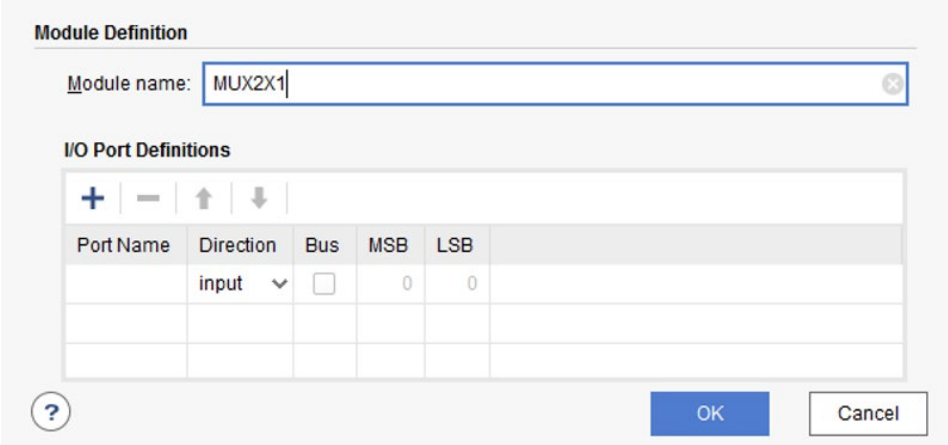
完成之后，创建工程就完成了。

- Step3：设计文件的编写。

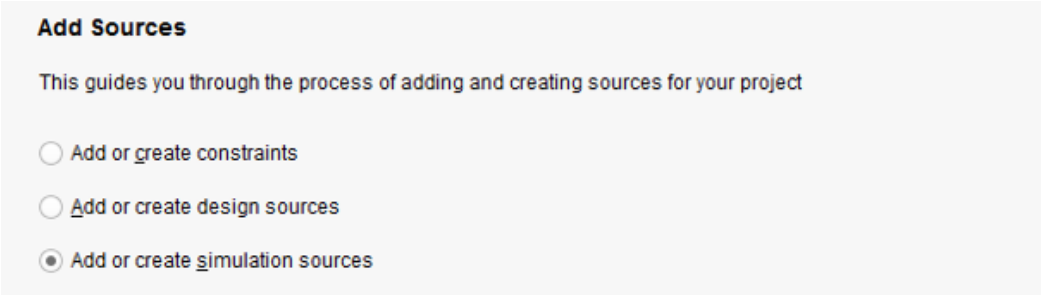
首先我们要新建一个设计文件



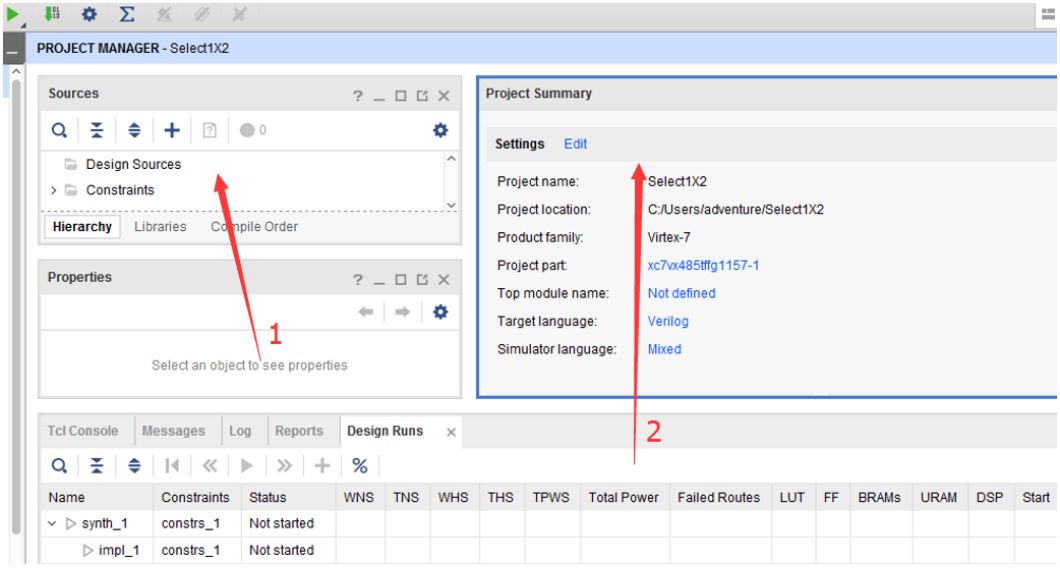
新建工程后会出现上图所示的界面，点击 Project manager 中的 add sources 就可以新建文件。新建文件的过程是给新文件命名，I/O 接口定义（可跳过），选择文件类型。



上图包含了文件命名和 I/O 接口定义，输入了文件名之后可以直接点选 OK。



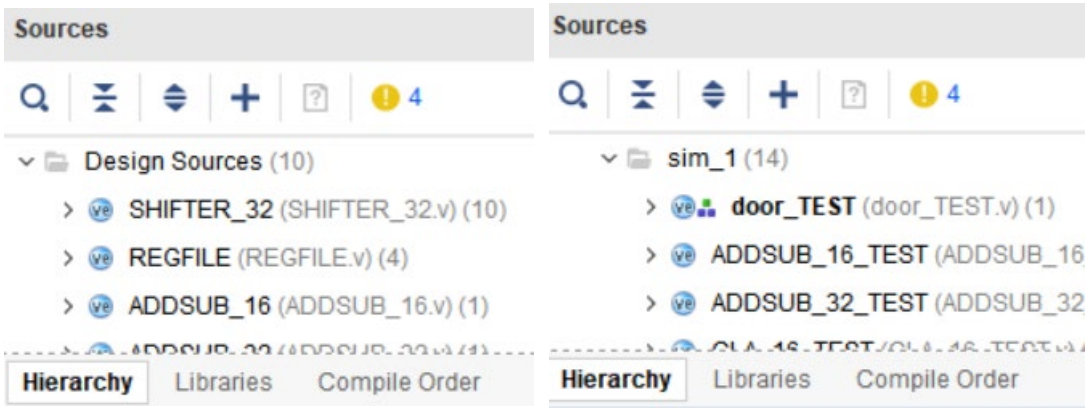
上图是选择文件类型的界面，第一个选项是同时生成设计文件和仿真文件，第二个选项是生成设计文件，第三个选项是生成仿真文件，在这里我们选择第二个文件类型，点击确认就新建了工程。



新建工程后就可以在如上图所示的 1 区域找到新建的文件，双击新建的文件后就可以在如上图所示 2 区域编辑文件。将实例代码复制到对应文件就完成了项目的编写。

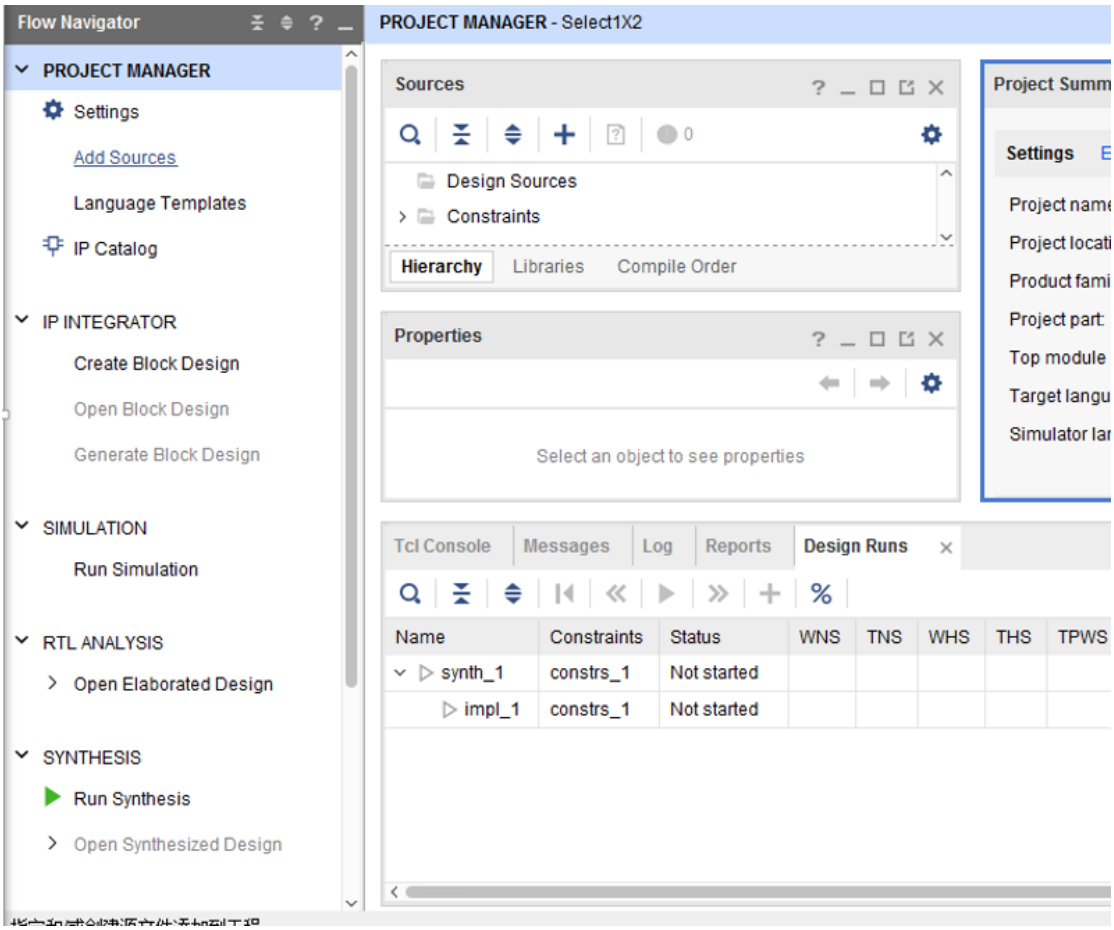
- Step3：RTL 图和仿真。

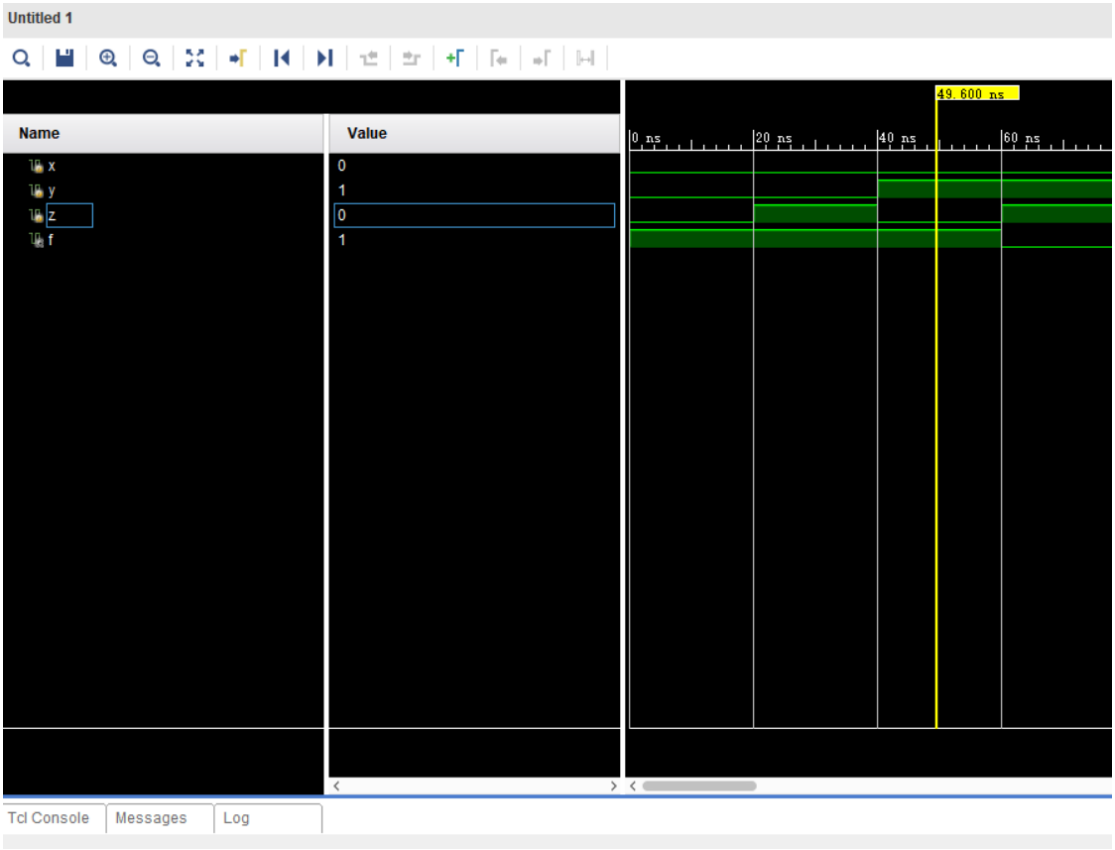
  如图所示，在 Source 区有三个文件夹，分别是 Design Sources，Constraints 和 Simulation Sources，Design Sources 放置了所有的设计代码，Simulation Sources 放置了所有的测试代码。



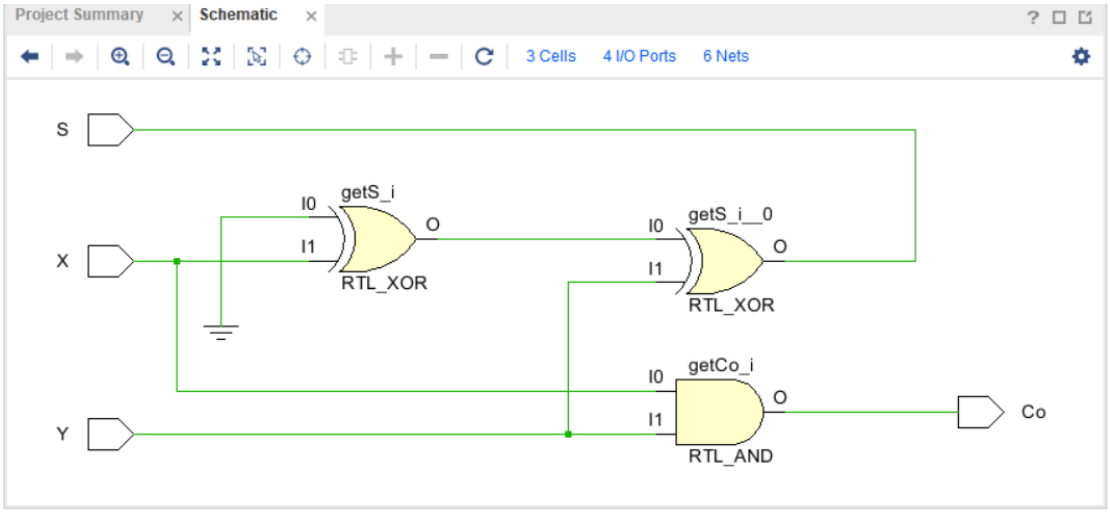
类似于 c 语言的 main 函数，测试文件也有顶层文件的说法，只有设置成顶层测试文件的测试文件才能够进行测试，可以右击你想测试的测试文件选择 Set As Top，将测试文件测试成顶层文件进行测试。特别的，当只有一个测试文件时，该测试文件会被默认成顶层文件，不需要进行多余的设置。

如下图所示，点击左侧的 RTL ANALYSIS 的 open Elaborated Design 就可以看到 RTL 图，如果存在仿真代码，则可点击 Run Simulation 的 run behavioral simulation 看到仿真图。





仿真图例如下图所示，这是一个仿真的波形图，这个界面的最右端为输入和输出的变量名称，中间为当前时间变量的准确值，我们可以左键波形图的任意位置来获得任意位置的变量的准确值。



如上图所示，这就是模仿出来的 RTL 图，可以清楚的看到电路的组织结构。

2 实验内容

2.1总体要求

本课程实验手册总共分为 10 个实验，分别是基本门电路的实现、多路选择器的实现、译码器实现、加减法器的实现、移位器的实现、寄存器堆的实现、存储器的整体封装、ALU 的整体封装、实现支持异常和中断的单周期 CPU 和解决数据冒险的带流水线的 CPU。

2.1.1 总体要求

（1） 代码模块名、变量名的命名要求规范，基本做到达意，便于阅读理解
（2） 重要代码要有注释说明（使用代码块注释和代码行注释）

2.1.2 检查说明

过程中检查：本课程实验，会在中间设置几个检查点，具体时间根据课程进度另行通知。

期末最终检查：期末结课前后安排最终检查，重点检查最后两个实验，用自己的电脑，接到投影仪上，汇报课程实验的 PPT 和代码及运行情况。突出自己设计思路、完成的创新点及难点，接受代码的随机解读抽查。同时，在 PPT 中说明同组两人的分工情况。

2.2各任务要求（包括模块代码和仿真测试代码）

2.2.1 实验一 门电路的实现（10 月 15 日前完成）

门电路的电路图如图 2-1 所示：

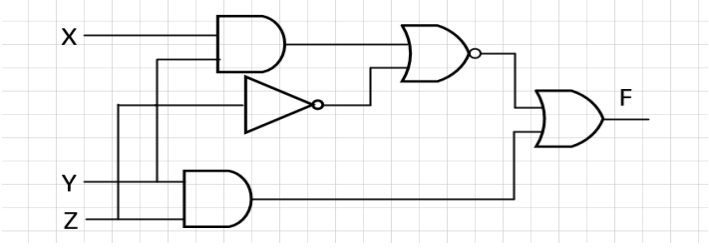


图 2-1 有门电路组成的电路图

需完成设计代码和仿真代码的实现，要求仿真代码实现所有真值的覆盖，并在提交报告时给出设计代码、仿真代码、RTL 图和仿真波形图。

2.2.2 实验二 多路选择器（10 月 15 日前完成）

包括 5 位二选一选择器，32 位二选一选择器，32 位 4 选一选择器，模块命名格式和每个选择器的输入输出参数要求分别为：

- 5 位二选一选择器（MUX2X5） 模块参数列表：module MUX2X5 (A0,A1,S,Y);
- 32 位二选一选择器（MUX2X32） 模块参数列表：module MUX2X32 (A0,A1,S,Y);
- 32 位四选一选择器（MUX4X32） 模块参数列表：module MUX4X32 (A0,A1,A2,A3,S,Y);

2.2.3 实验三 译码器（10 月 15 日前完成）

请用代码实现带使能端的 2-4 译码器、5-32 译码器电路，模块分别命名为：
module DEC2T4E (I0, I1, En, Y0, Y1, Y2, Y3)和
module DEC5T32E (I, En, Y)
其中 I 信号为 5 位宽，Y 为 32 位宽

2.2.4 实验四 加减法器（10 月 15 日前完成）

请用代码实现该 4 位超前进位加法器的门级电路，模块命名为：
module CLA_4 (X, Y, Cin, S, Cout)
其中 X，Y 和 S 信号均为 4 位宽，Cin 和 Cout 为 1 位宽

2.2.5 实验五 移位器（10 月 15 日前完成）

请用代码实现 32 位移位器，支持算术和逻辑左移、右移操作，模块命名为：
module SHIFTER_32(X, Sa, Arith, Right, Sh)
其中 ：X 为 32 位的输入；Sa 为 5 位的移位量；Arith 是算术或逻辑移位标识，1 为算术移位，0 为逻辑移位；Right 为右移或左移标识，1 为右移，0 为左移；Sh 为 32 位的结果输出。

2.2.6 实验六 寄存器堆（10 月 25 日前完成）

模块名：module REGFILE (Ra, Rb, D, Wr, We, Clk, Clrn, Qa, Qb)
参见教材 P108。

2.2.7 实验七 ALU 封装（10 月 25 日前完成）

ALU 需完成 9 种运算即可，具体完成的功能如表 2-1 所示。

表 2-1：ALU 功能真值表

Aluc	功能描述
x000	add（加）
x100	sub（减）
x001	and（与）
x101	or（或）
x010	xor（异或）
x110	lui（设置高位）
0011	sll（左移）
0111	srl（右移）
1111	sra（算术右移）

模块名：module ALU(

```
input [31:0] X,
input [31:0] Y,
input [3:0] Aluc,
output [31:0] R,
output        Z
);
```

2.2.8 实验八 存储器（10 月 25 日前完成）

包括指令存储器和数据存储器，参见教材 P109-110：

module INSTMEM (Addr, Inst)

module DATAMEM (Addr, Din, Clk, We, Dout)

2.2.9 实验九 单周期 CPU （计分实验，11 月 10 日前完成）

基础要求，实现 14 条指令，见教材表 5-10。

可以扩展实现 lui 指令。

2.2.10 实验十 支持简单异常和中断处理的单周期 CPU（计分实验二选一之一，11 月 10 日前）

2.2.11 实验十一 带流水线的 CPU（计分实验二选一之二，11 月 30 日前）

基本要求包括：写操作提前半周期、内部前推

可以扩展实现：lw 指令的数据冒险，以及其他控制冒险

3 实验一门电路的实现指导

3.1 相关知识

3.1.1 注释符

Verilog HDL 语言允许插入注释，标明程序代码功能、修改、版本等信息，以增强程序的可阅读性和帮助管理文档。Verilog HDL 有两种注释方式：

- 单行注释：单行注释以“//”开始，Verilog HDL 忽略从此处到行尾的内容；
- 多行注释：多行注释以“/*”开始，到“*/”结束，Verilog 忽略其中的注释内容。

3.1.2 标识符和转义符

在 Verilog HDL 中，标识符（Identifier）被用来命令信号名、模块名、参数名等。它可以使任意一组字母、数字、$符号和_符号的组合。应该注意的是，标识符的字符区分大小写，并且第一个字符必须是字母或者下划线。

Verilog HDL 规定了转义标识符（Escaped Identifier）。采用转义字符可以在一条标识符中包含任何可打印的字符。转义标识符以“\”（反斜线）符号开头，以空白符结尾（空白可以是 一个空格、一个制表符或者换行符）。

3.1.3 关键字

Verilog HDL 语言内部已经使用的词称为关键字或保留字，它是 Verilog

HDL 语言的内部专用词，是事先定义好的确认符，用来组织语言结构的。需要注意的是，在 Verilog HDL 中，保留字都是小写的。

❖ 与门、或门以及同类门单元

在 Verilog HDL 编程中实例化此类门单元需要用下列关键字中的一个作为实例化的模块名：

and(与)，nand(与非)，nor(或非)，or(或)，xor(异或)，xnor(异或非)

此类门电路的使用如下：

and and1 (tmp_1, x, y，z);

其中，“and1”是门单元“and”的一个实例化接口名称，第一个参数 tmp_1 是输出变量，即输入变量 x，y，z 进行 and 的结果。输出变量只有 1 个，输入变量多于两个。其余此类门电路的使用类似。

❖ 非门

和与/非类门单元相反，非门具有一个输入端口，以及一个或多个输出端口。在在 Verilog HDL 编程中实例化此类门单元需要用下列关键字中的一个作为实例化的模块名：

not(非)

此类门电路的使用如下：

not not1 (tmp_2, x);

其中，“not1”是门单元“not”的一个实例化接口名称，第一个参数 tmp_2 输出变量，即输入变量 x 进行 not 的结果。

3.1.4 数值

Verilog HDL 有四种基本的逻辑数值状态，用数字或字符表达数字电路中传送的逻辑状态和存储信息。Verilog HDL 逻辑数值中，x 和 z 都不区分大小写。也就是说 ，0x1z 和值 0X1Z 是等同的。

Verilog HDL 中的四值电平逻辑如下表：

状态	含义
0	低电平、逻辑 0、“假”
1	高电平，逻辑 1 或“真”
X 或 x	不确定或未知的逻辑状态
Z 或 Z	高阻态

3.1.5 仿真文件中的变量类型

reg：reg 定义的是寄存器类型，通过赋值语句可以改变寄存器储存的值，reg 型变量常用来表示用于“always”模块内的指定信号，在“always”块内被赋值的每一个信号都必须定义成 reg 型。

定义 reg 变量的格式为”reg ([n – 1:0]) 变量名 1,变量名 2,…,变量名 I;”其中当宽度为 1 时，括号中的内容可以去掉，n 代表是一个几位的变量，比如说声明一个 32 位的寄存器类型变量，格式为 reg [31:0] a;声明一个一位的寄存器变量为 reg b;

wire：wire 型变量通常是用来表示单个门驱动或连续赋值语句驱动的网络型数据，是用于连接器件单元，不能直接进行赋值，而必须使用 wire 或者 reg 来赋值，定义的方式和 reg 相同。比如说 wire a; a = 1 这种做法是错误的，而必须使用 “reg a = 1; wire b = a;wire c = b; ”。

3.1.6 阻塞赋值

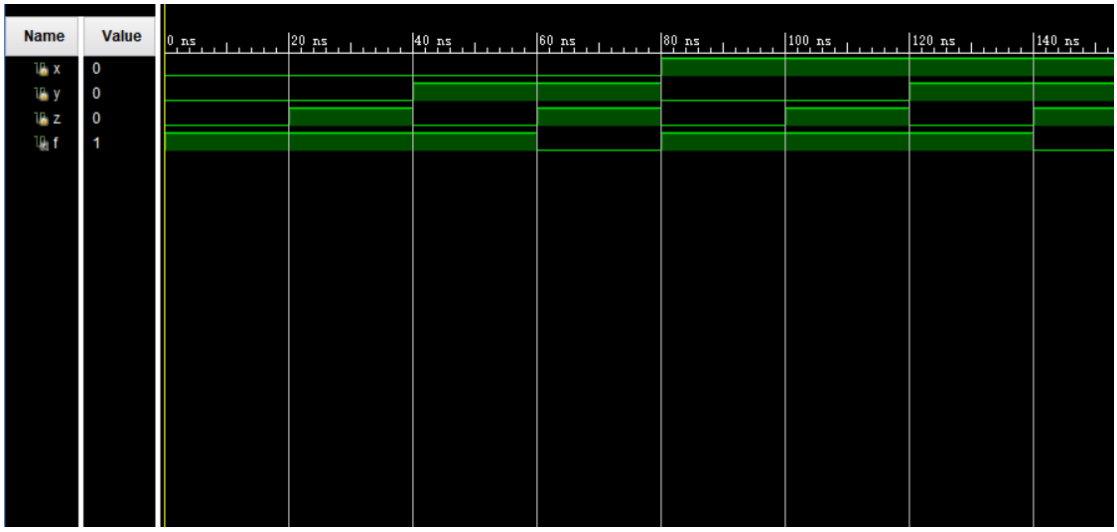
阻塞是指在同一个 always 块中，其后面的赋值语句从概念上是 在前一条赋值语句结束后开始赋值的。非阻塞语句的执行过程是：首先计算语句块内部所有右边表达式的值，然后完成对左边寄存器变量的赋值操作。

阻塞赋值： begin B=A; C=B+1; end 上述代码先将 A 的值赋值给 B，C 的值是 A+1；	非阻塞赋值： begin B<=A; C<=B+1; end 上述代码的最终结果是：将 A 赋值给了 B，但是 C 的值是 B 原来的值+1。因为最先计是的是右边的表达式。

```
module door_TEST;
    // Inputs
    reg x,y,z;    // 定义三个输入变量，为模块的三个输入，定义成 reg 赋值
    // Outputs
    wire f;       // 定义输出变量，使用 wire 作为变量类型
    //wire S0, S1,S2;      // 定义三个中间变量
     // Instantiate the Unit Under Test (UUT)
    door uut (.x(x),.y(y),.z(z),.f(f));          //应用设计模块
    initial begin
        // Initialize Inputs
        x = 0;           //把 x，y，z 的初始值设置为 0
        y = 0;
        z = 0;
        // Wait 20 ns for global reset to finish
        #20;           //等待 20ns 改变 x,z,y 参数的值
        x = 0;
        y = 0;
        z = 1;
        #20;
        x = 0;
        y = 1;
```

```
        z = 0;
        ......
        ......
        #20;
        x = 1;
        y = 1;
        z = 1;
    end
endmodule
```

仿真图如下：



3.3 实验要求

写出下图所示门电路的逻辑表达式，并进行实验操作，给出其仿真图。