互联网+班《计算机组成与结构》课程
综合实验手册（V1.2 版）

Update: 2020-11-02

目 录

1　实验平台 ModelSim 的安装和使用

1.1实验平台的安装

1.1.1　说明

测试过的系统：win10 64bit

参考：

https://blog.csdn.net/weixin_42693097/article/details/90523692

1.1.2　下载

安装包大约 500MB，安装后大约 1.2GB

1)

百度云（非年费会员限速）链接：

https://pan.baidu.com/s/14ZxTLwfZKD_Qjjlk4PCkYQ

提取码：nxcm

2)

OneDrive（不限速，但需要梯子）链接：

https://1drv.ms/u/s!AhsQ6cw4ZjWRiJxDklImteAMxlBJOg?e=j4U541

提取码：nxcm

3)

百度云（非年费会员限速，有个小广告）链接：

https://pan.baidu.com/s/1k8f7dgQ2AYjssUDPTt3ToA

提取码：d36q

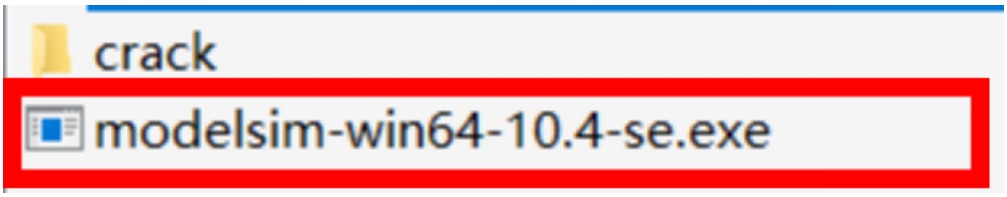
4)

在官网下载学生版，无需破解，需要注册，链接：

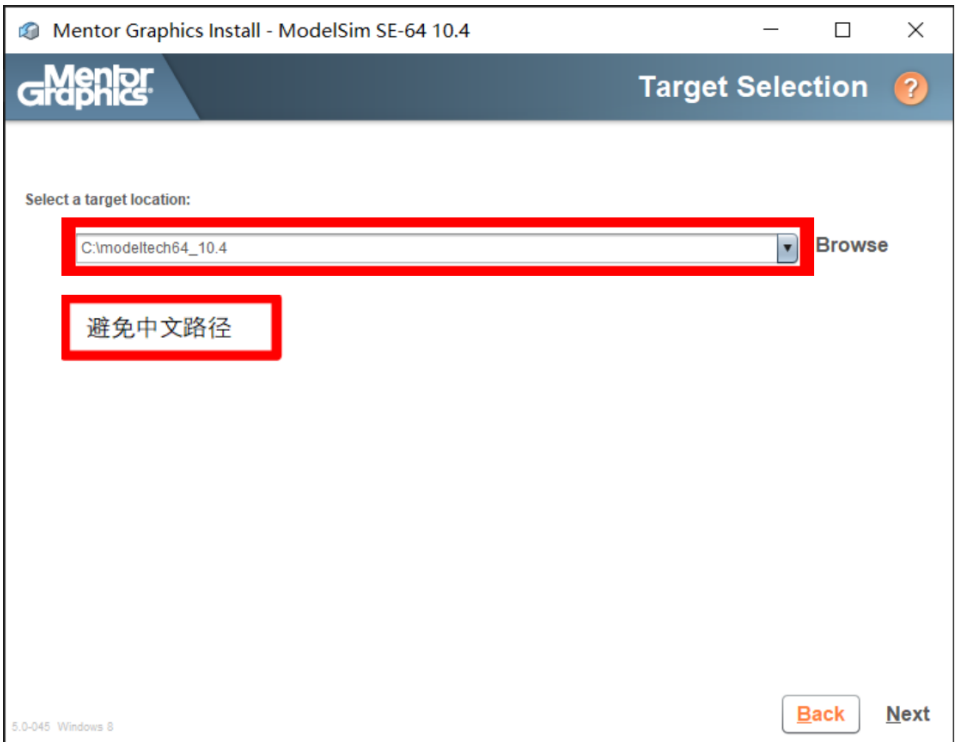
https://www.mentor.com/company/higher_ed/modelsim-student-edition

1.1.3　安装及关键选项

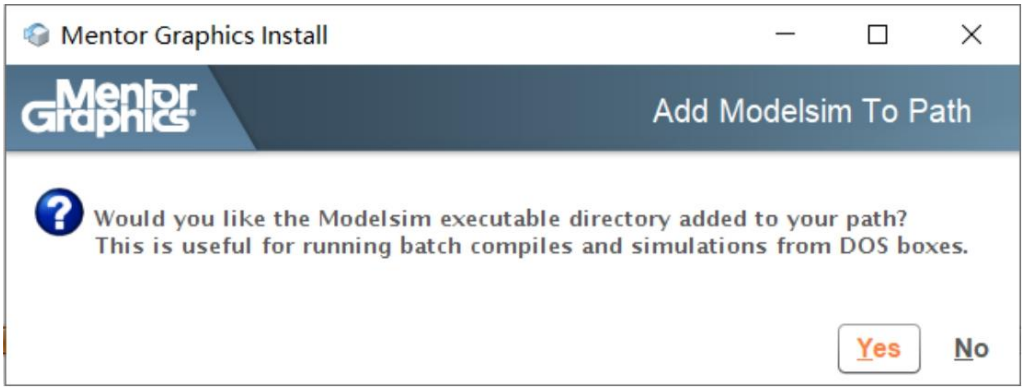
1)　双击运行安装软件



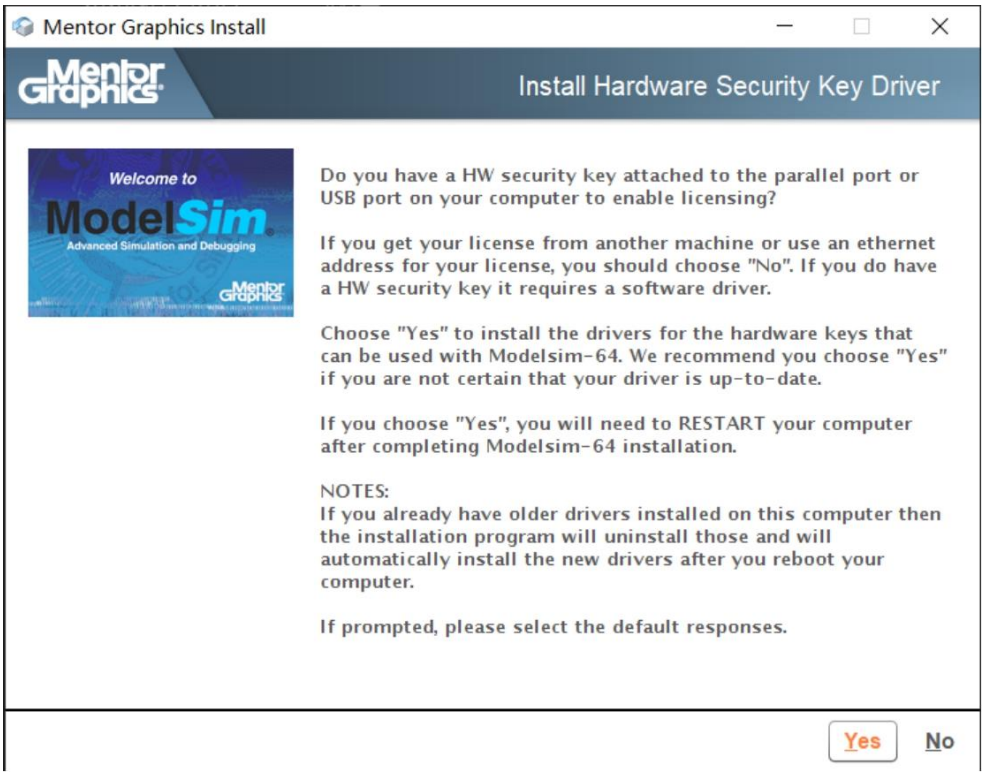
2)　选择安装路径，应避免中文



3) 添加环境变量，尽量选择 yes



4) 硬件安全 Key 驱动，点击 no

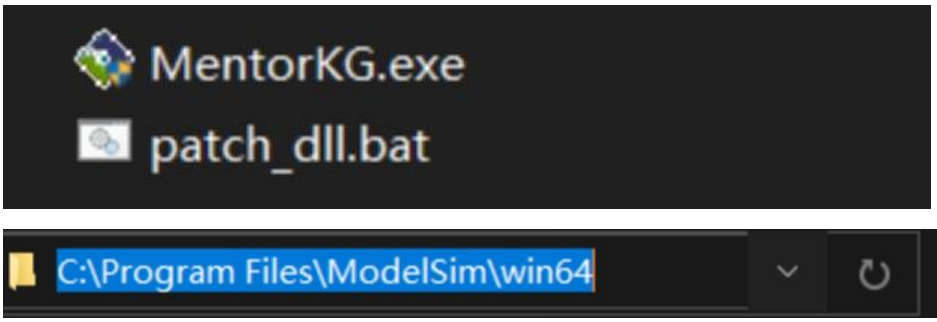


5) 安装完成

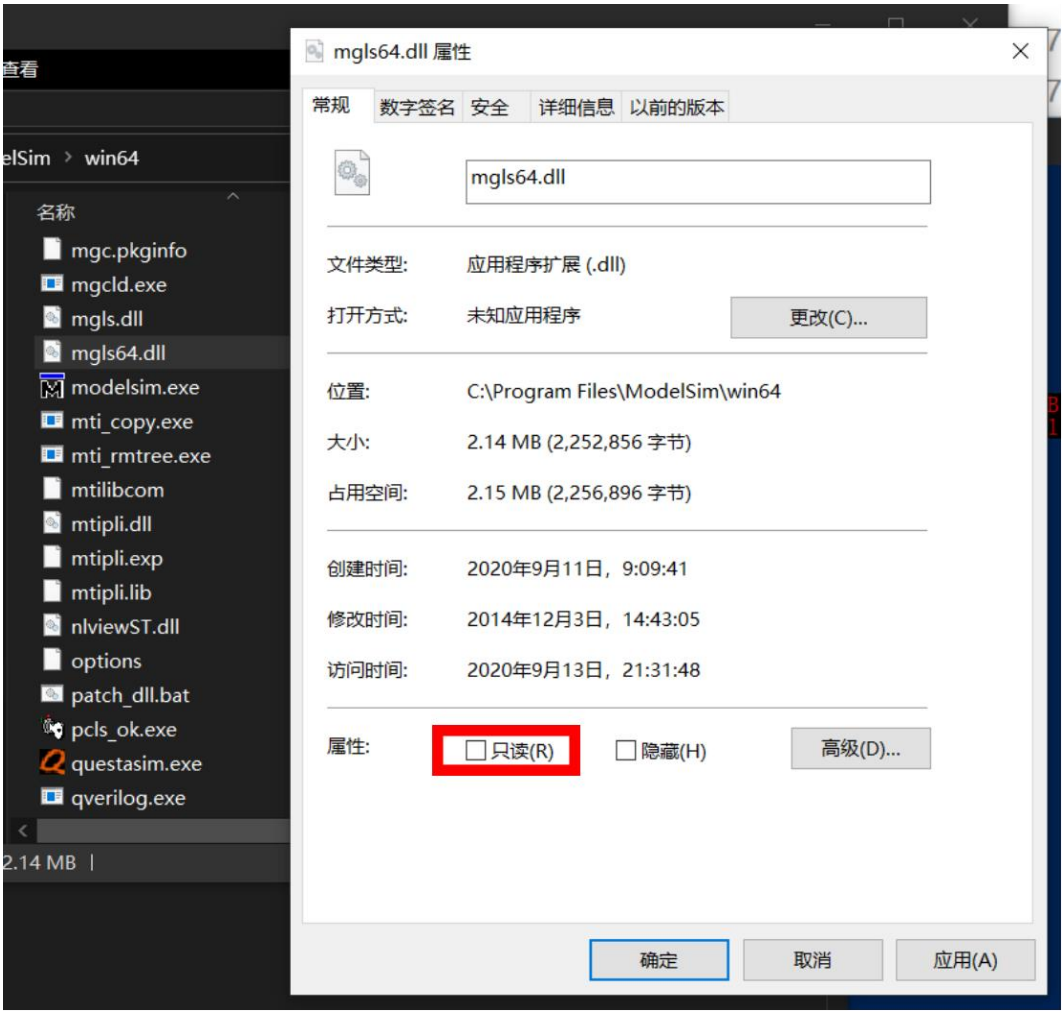
1.1.4 破解

1) Crack 文件夹中的这两个文件复制到安装目录的 win64 文件夹下面，如有重名，直

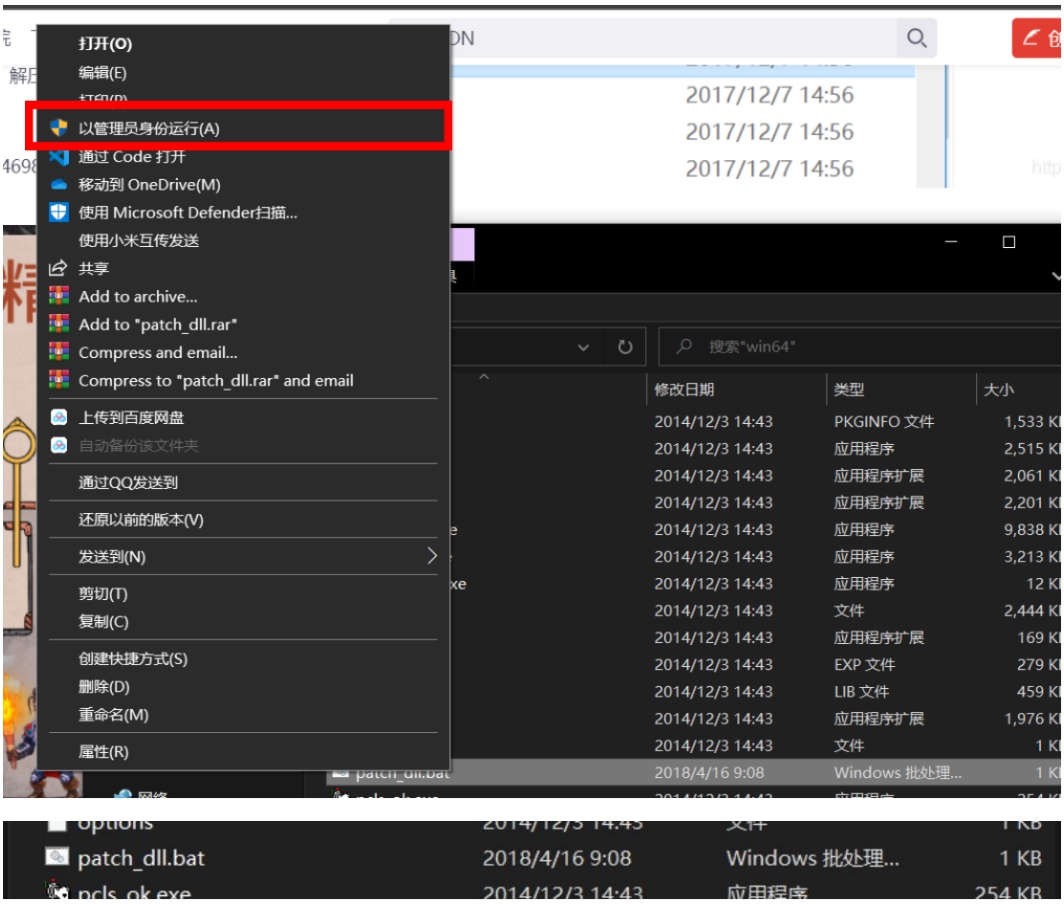
接覆盖



2) 修改 mgls64.dll 的读写属性，去掉只读复选框



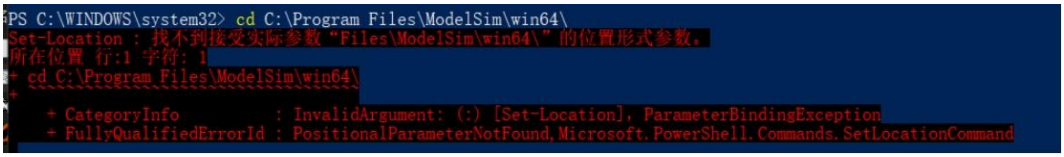
3) 以管理员身份运行 patch_dll.bat。如果命令行窗口一闪而过，或者没有出现步骤 7 的情况，请继续步骤 4

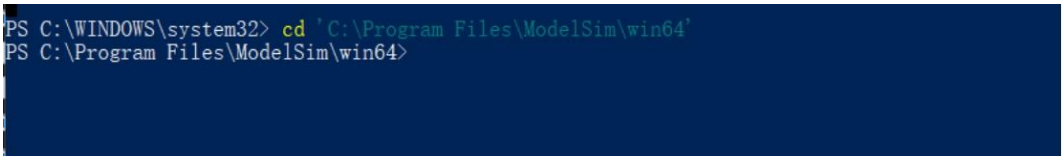


4) 右击开始，使用管理员打开 PowerShell

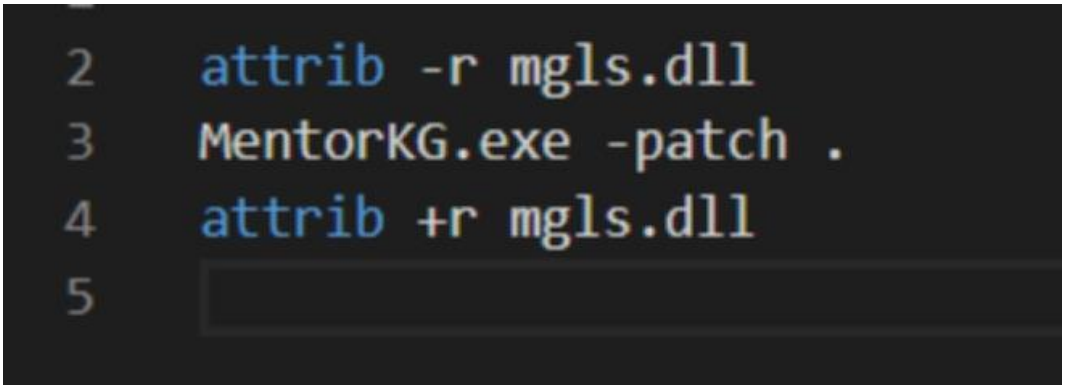
5) 进入 ModelSim 安装目录的 win64 目录，注意路径最好嵌在单引号内。如果没有单引号，无法进入带空格的路径。请手动输入下面的命令，如果直接复制下面这条命令至 powershell，单引号会被过滤掉。

cd ‘C:\Program Files\ModelSim\win64’



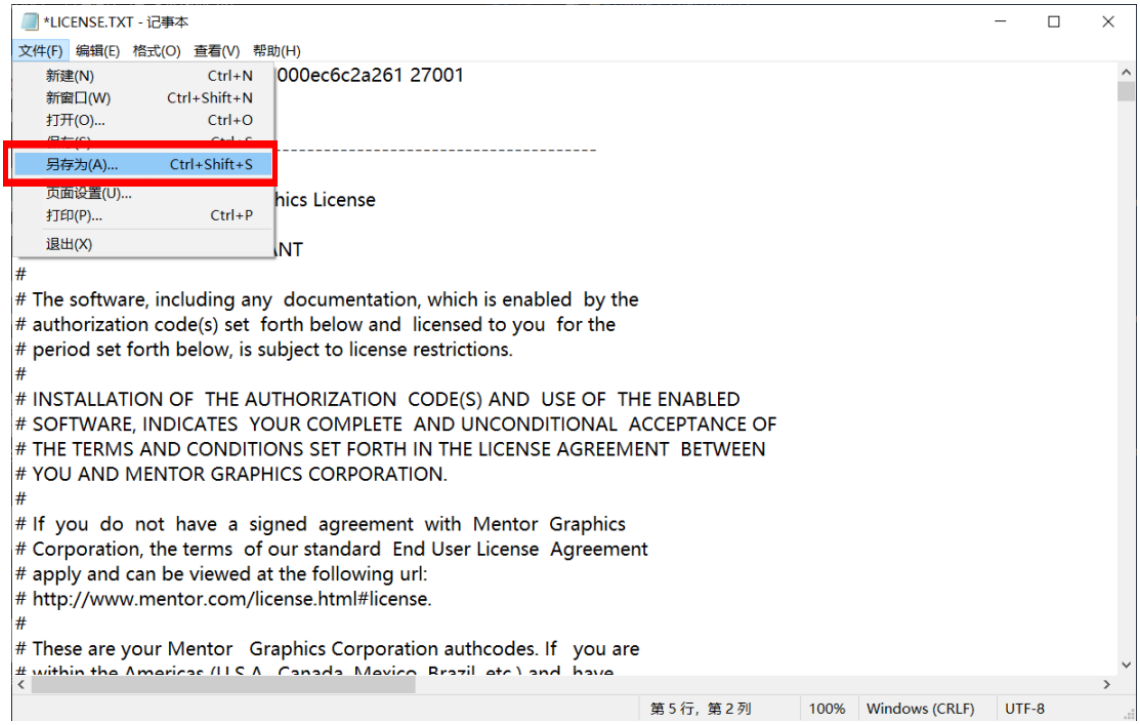


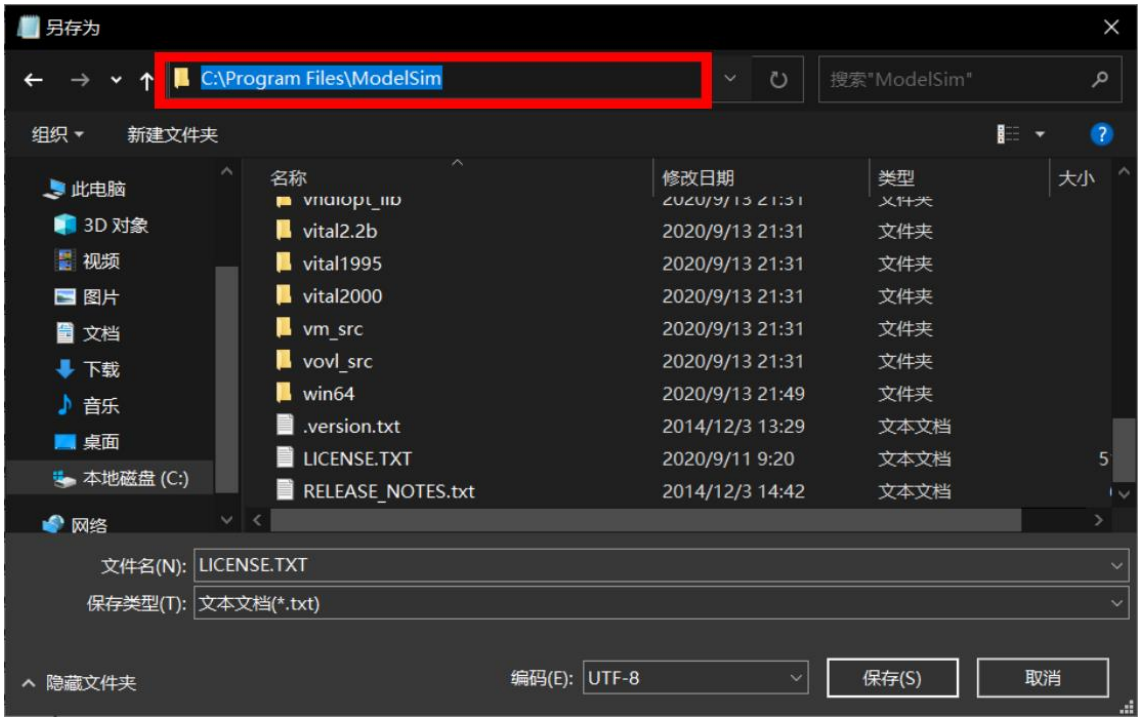
6) 依次执行以下命令，注意第二行的英文句号



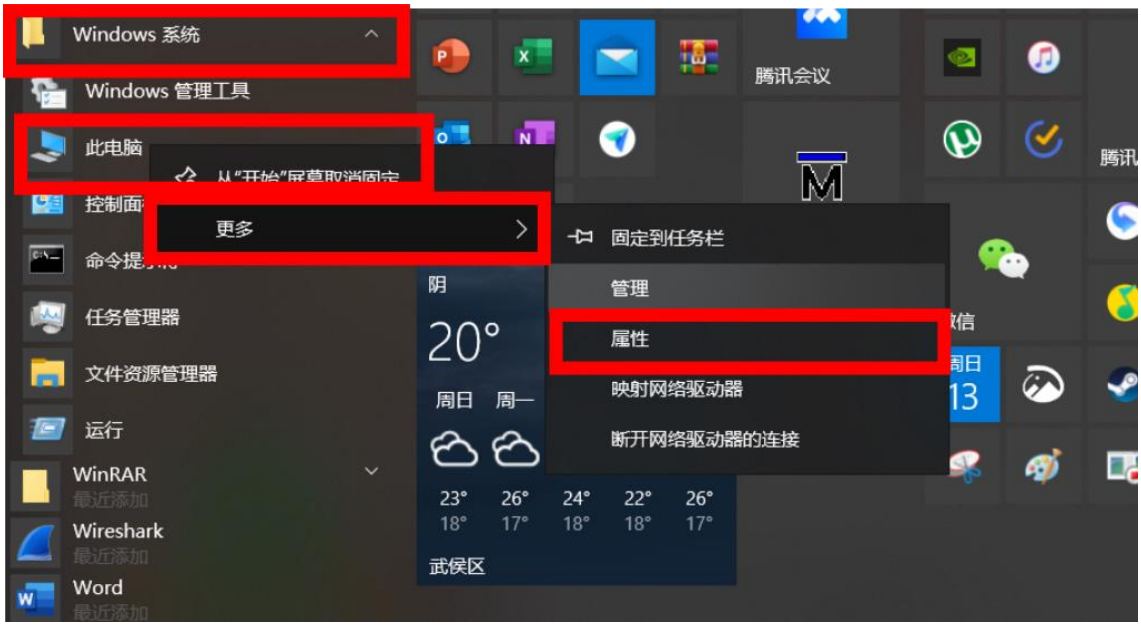

```
attrib -r mgls.dll
MentorKG.exe -patch .
attrib +r mgls.dll
```

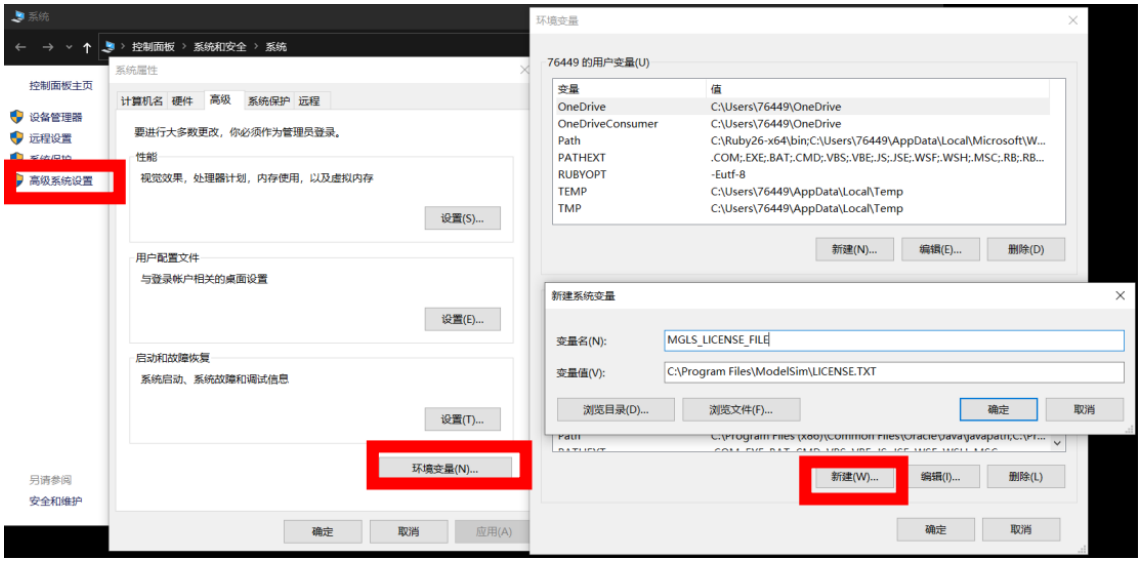
7) 执行第二条命令时会稍微卡顿，并弹出记事本。将记事本另存到 ModelSim 的安装目录。如有重名，直接覆盖。





8) 新建环境变量（环境变量在“我的电脑－属性－高级系统设置－环境变量”），变量名为 MGLS_LICENSE_FILE，变量值为上面 LICENSE.TXT 文件的路径，如我的是 C:\Program Files\Modelsim\LICENSE.TXT





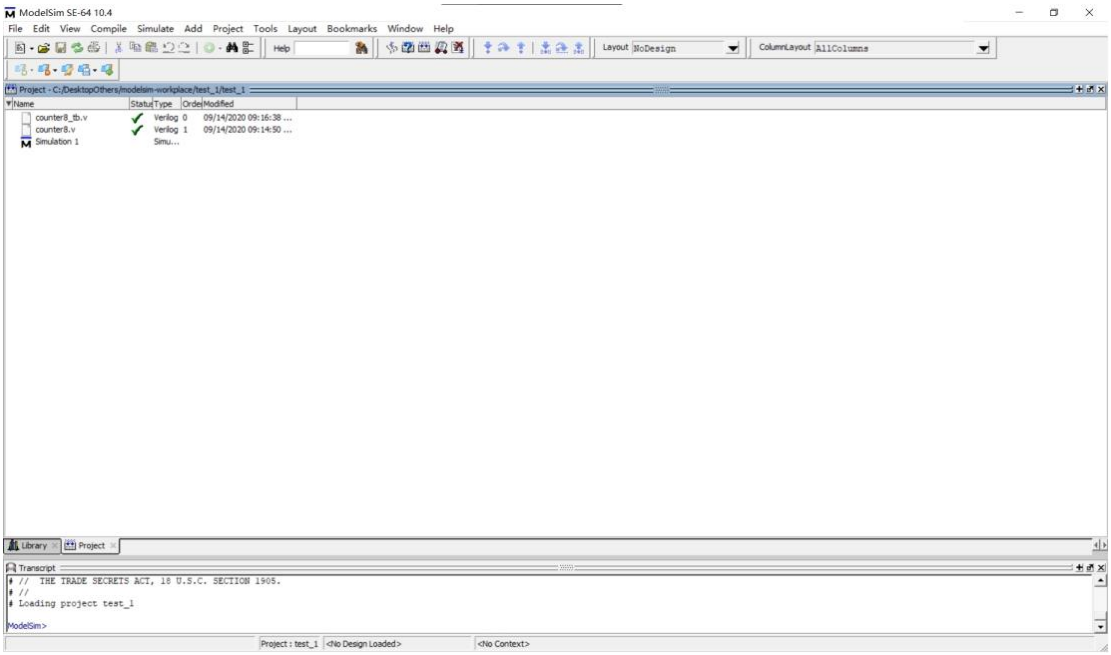
1.2实验平台的使用

语言：Verilog HDL
参考：
https://mp.weixin.qq.com/s?__biz=MzAwOTU5NjY5OQ==&mid=401524002&idx=1&sn=a7c6c8307a807b65260c44caf201f676&scene=6#wechat_redirect

1.2.1 新建项目与代码编写

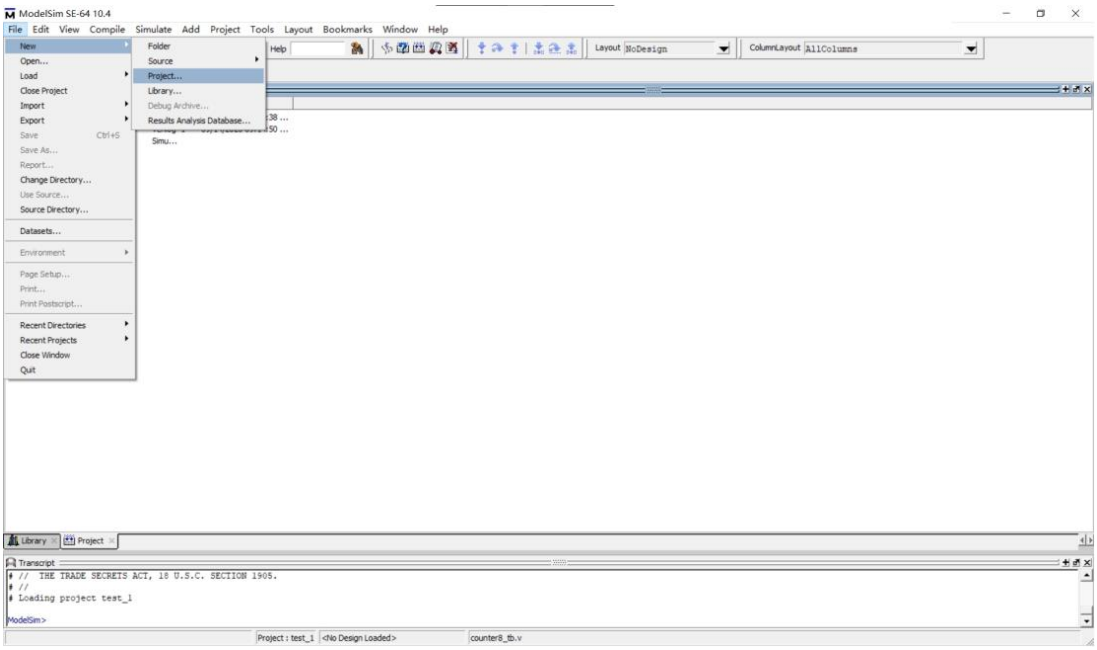
1. 打开软件

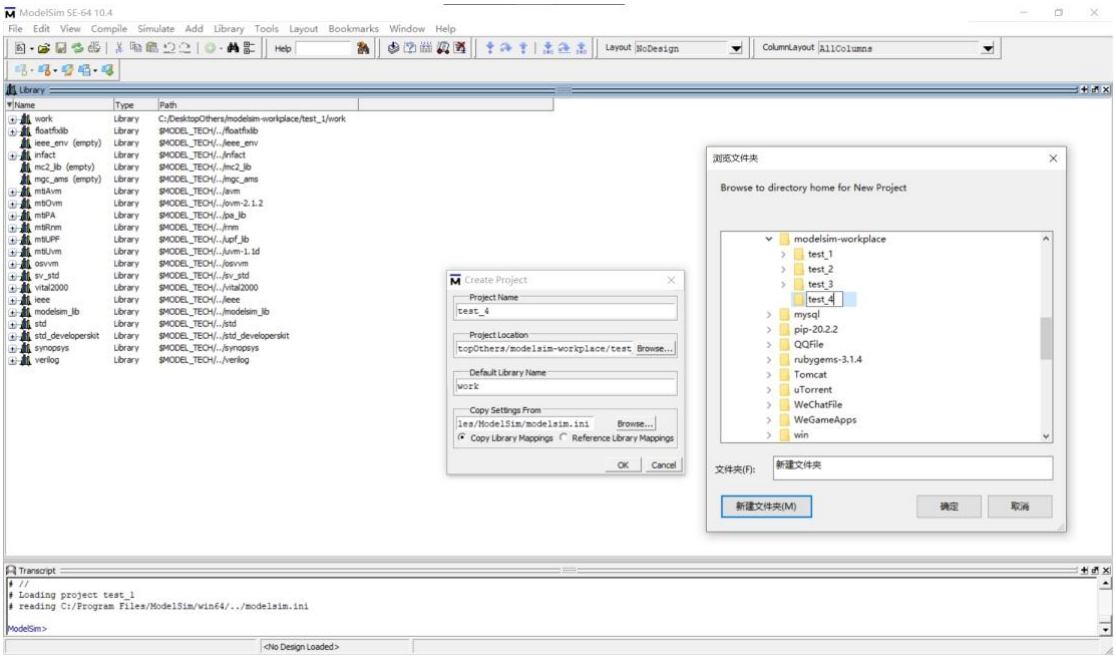
打开 ModelSim 软件，进入主界面



2. 新建项目

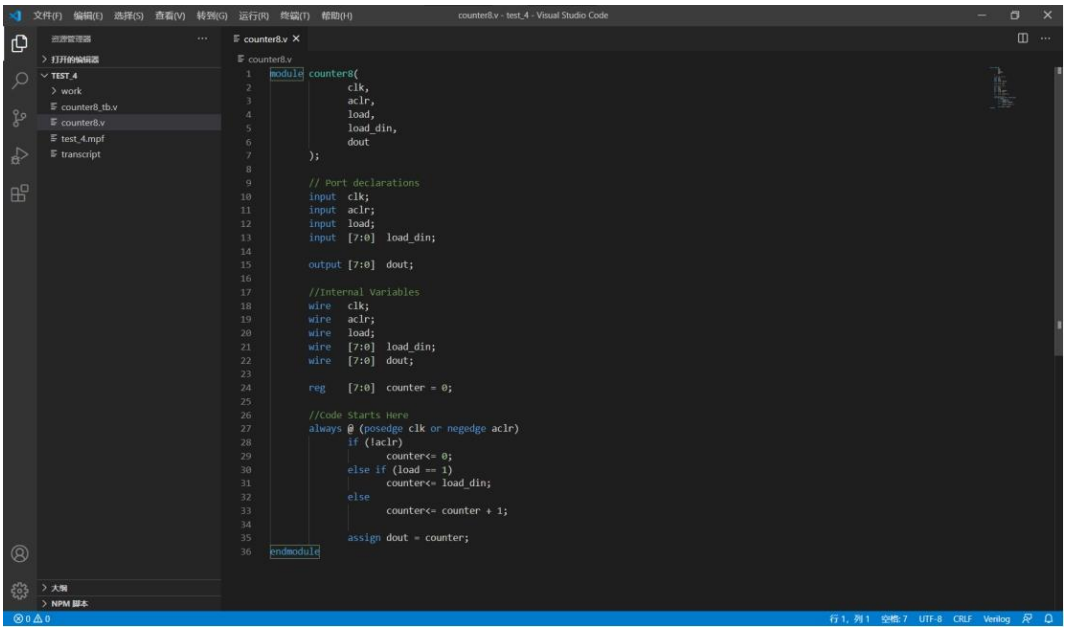
进入 ModelSim 主窗口后，选择 File 菜单下的“New->Project”，新建一个项目。在弹出的对话框中，给该工程命名并指定一个存放的路径，应避免出现中文



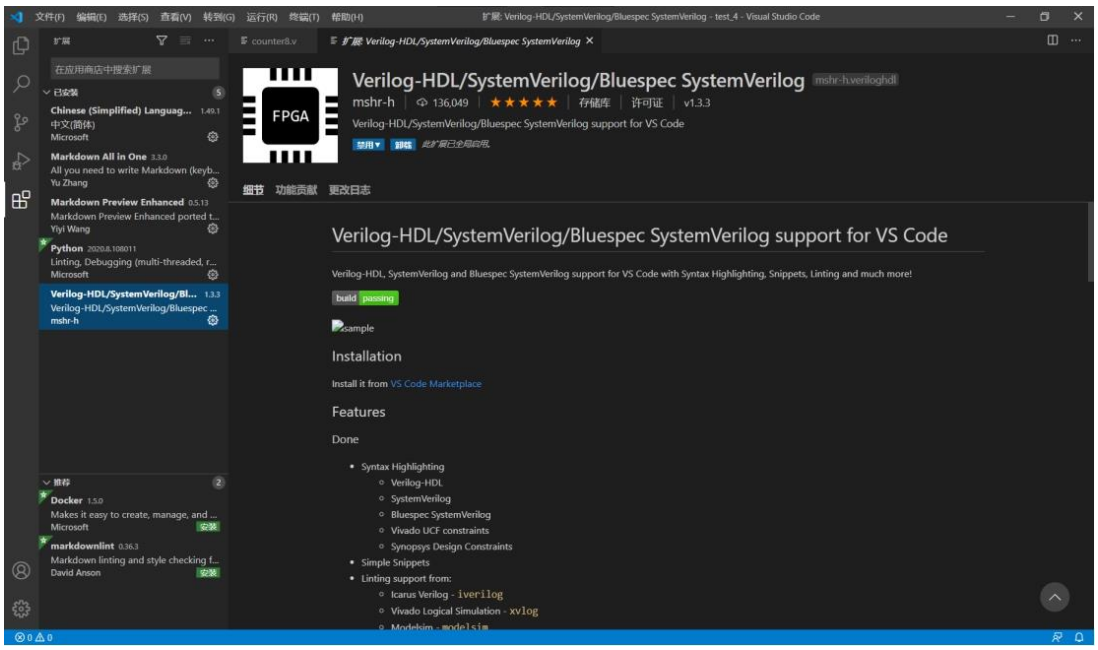


3. 进行开发

在新建的目录中，使用别的编辑器进行开发。例如 VS Code



若使用 VS Code 进行开发，建议下载下图所示的插件




```
// counter8.v
    module counter8(
                clk,
                aclr,
                load,
                load_din,
                dout
            );

            // Port declarations
            input  clk;
            input  aclr;
            input  load;
            input  [7:0]  load_din;

            output [7:0]  dout;

            //Internal Variables
            wire   clk;
            wire   aclr;
            wire   load;
            wire   [7:0]  load_din;
            wire   [7:0]  dout;

            reg    [7:0]  counter = 0;

            //Code Starts Here
```

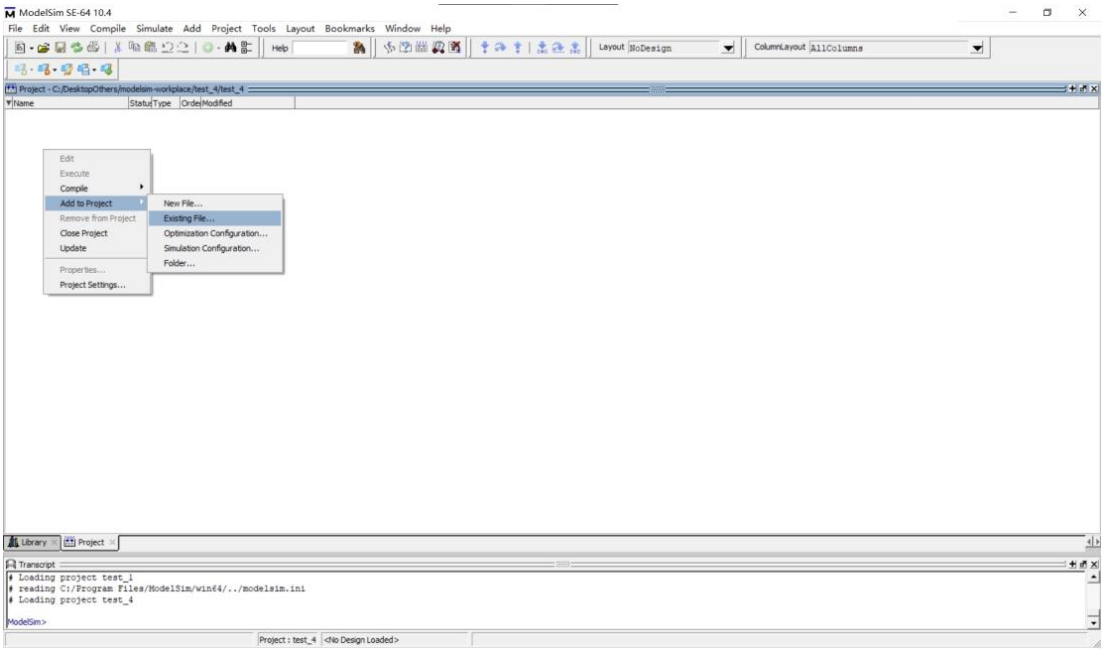
```
        always @ (posedge clk or negedge aclr)
                if (!aclr)
                        counter<= 0;
                else if (load == 1)
                        counter<= load_din;
                else
                        counter<= counter + 1;

                assign dout = counter;
    endmodule

// counter8_tb.v
    `timescale 1ns/1ns       //注意最前面的符号是数字键“1”左边的反引号，不是单引号
    module test_counter8;
            reg    clk;
            reg    aclr;
            reg    load;
            reg    [7:0]  load_din;
            wire   [7:0]  dout;

            initial
            begin
                        clk = 0;
                        aclr = 1;
                        load = 0;
                        load_din =0;
                #120    aclr =0;
                #40     aclr =1;
                #20     load =1;
                        load_din =100;
                #20     load =0;
                #100    $stop;    //可以不添加这个仿真结束的系统任务
            end

            always #10 clk = ~clk;

            counter8 U(
                        .clk(clk),
                        .aclr(aclr),
                        .load(load),
                        .load_din(load_din),
                        .dout(dout)
                );
```

```
endmodule
```

4. 添加文件

开发完成后，将新的文件加入刚才的项目中。

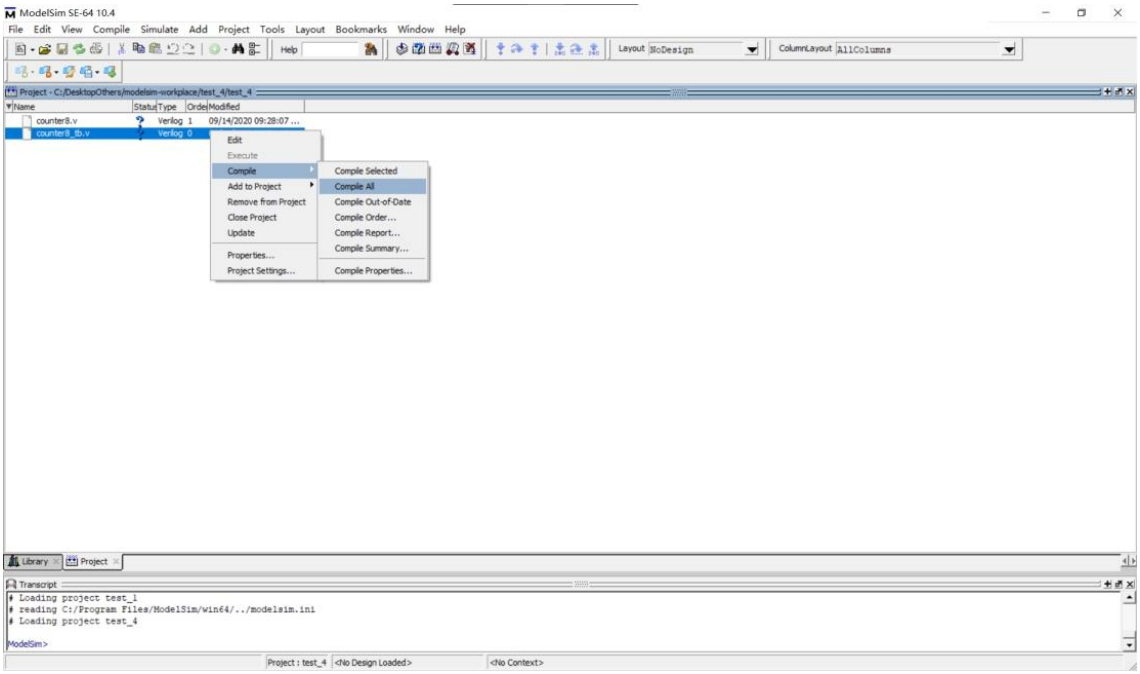
右击工作区（右击空白部分、文件也都可以），“Add to Project->Existing File”



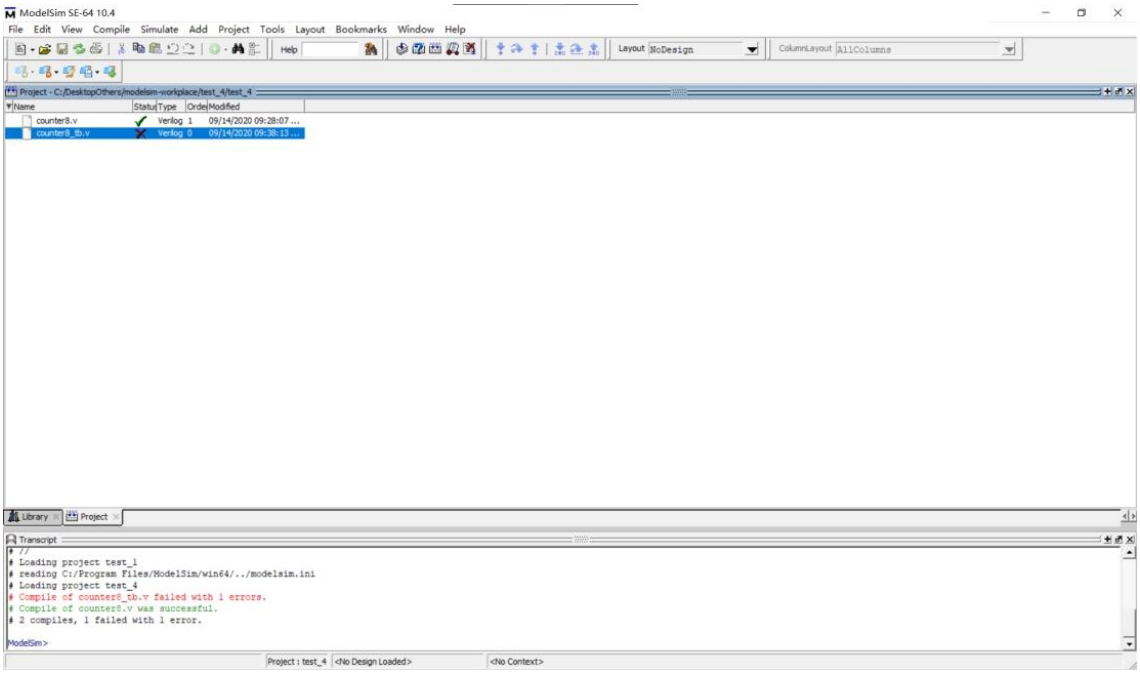
可以同时选中一个或多个文件。

1.2.2 编译

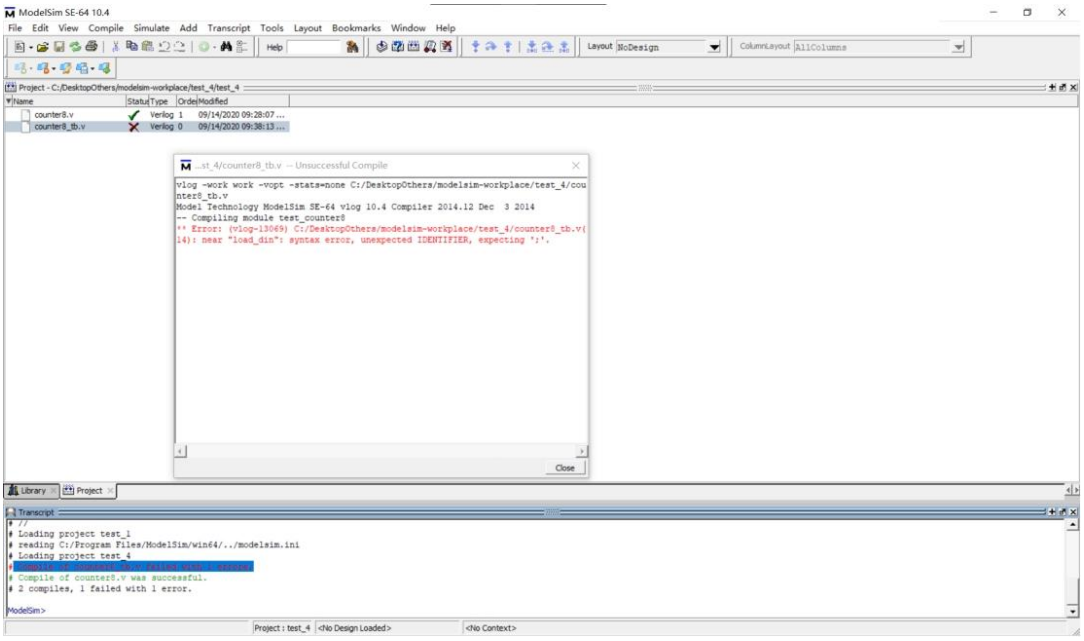
右击工作区，“Compile->Compile All”。也可以选择某个文件后，单独编译一个文件，“Compile->Compile Selected”



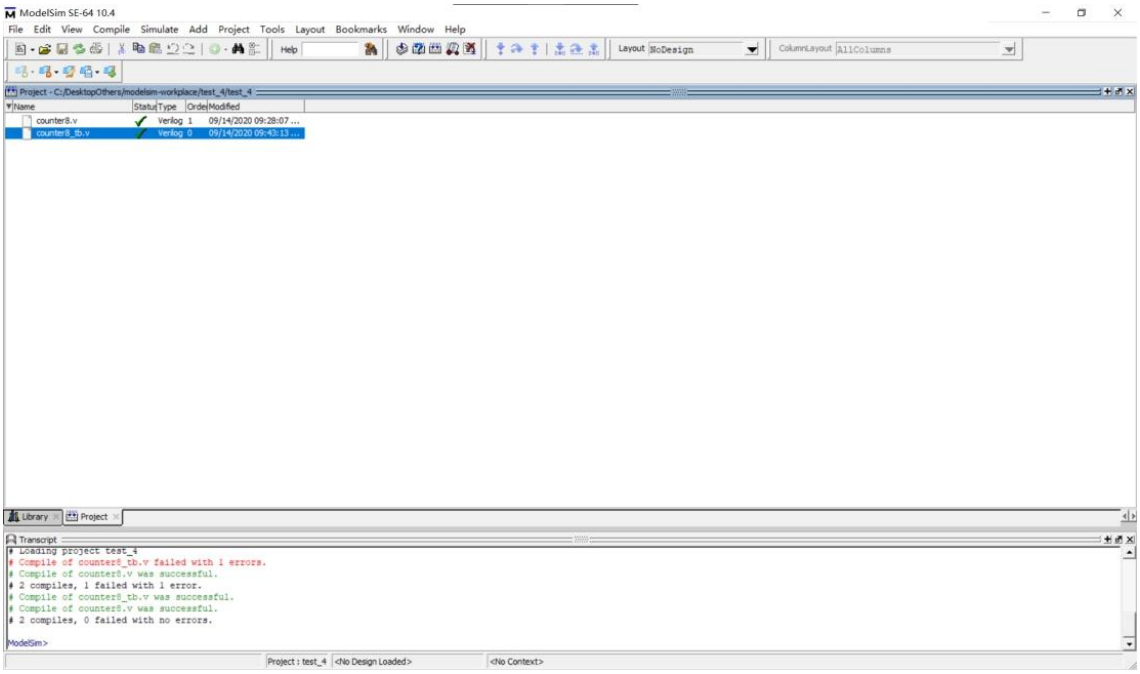
编译能拦截语法和部分逻辑问题



双击控制台红色提示部分，可以看到具体的错误提示

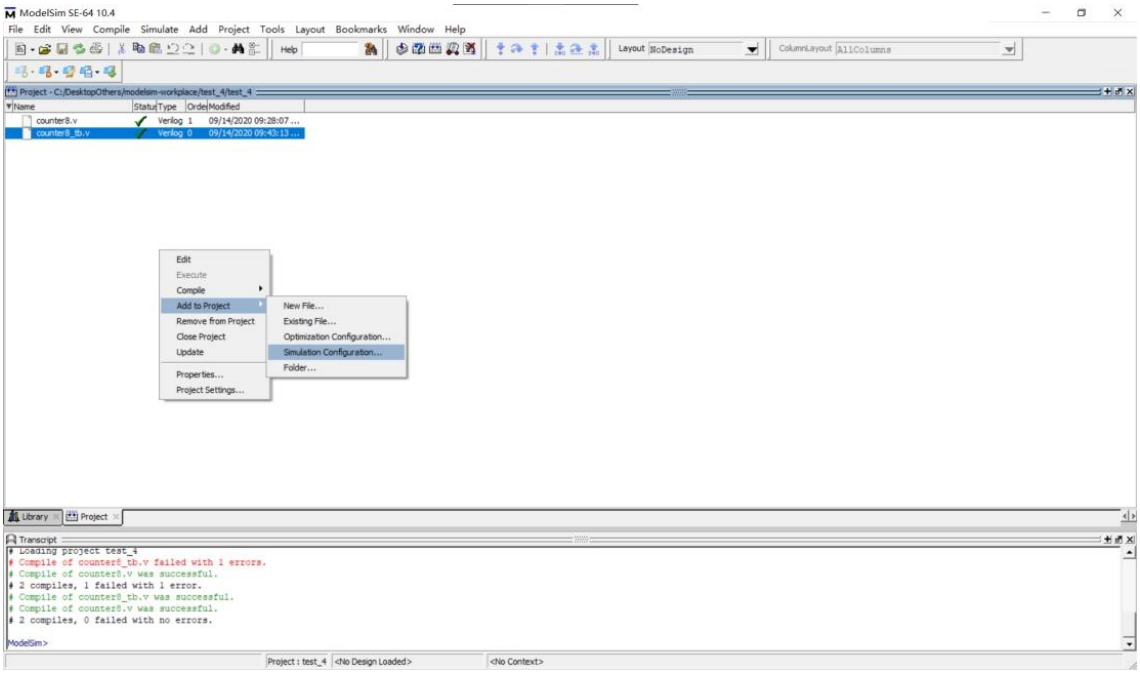


修改完成后重新编译，可以看到文件的 status 都是绿色的对勾

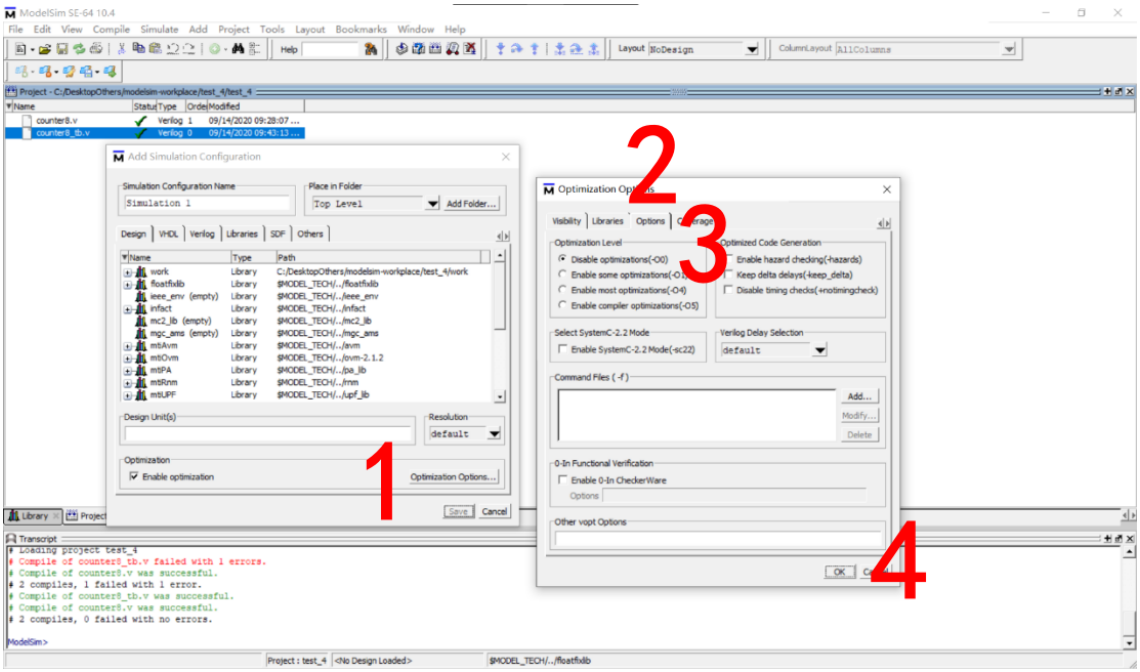


1.2.3 添加仿真配置文件

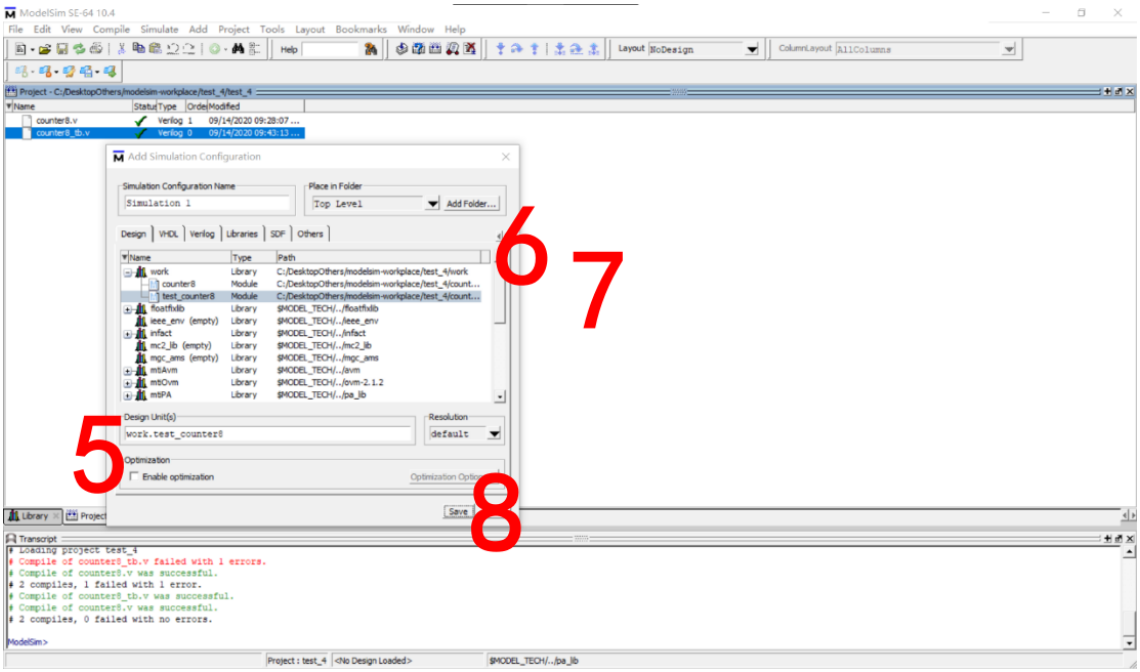
只有新建项目或项目发生重大改变后才需要新建、修改仿真配置文件。右击工作区，“Add to Project->Simulation Configuration”



1) 点击右下角“Optimization Options”
2) 打开选项卡“Options”
3) 选择“Optimization Level”为“Disable optimizations(-O0)”
4) 点击右下角“OK”

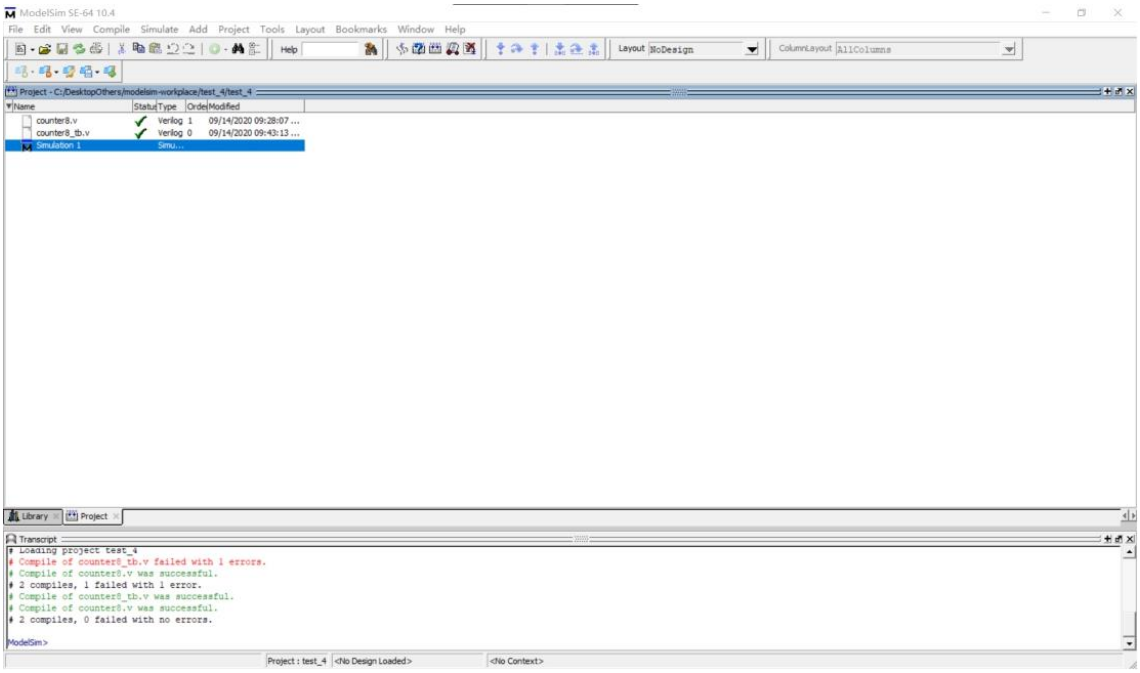


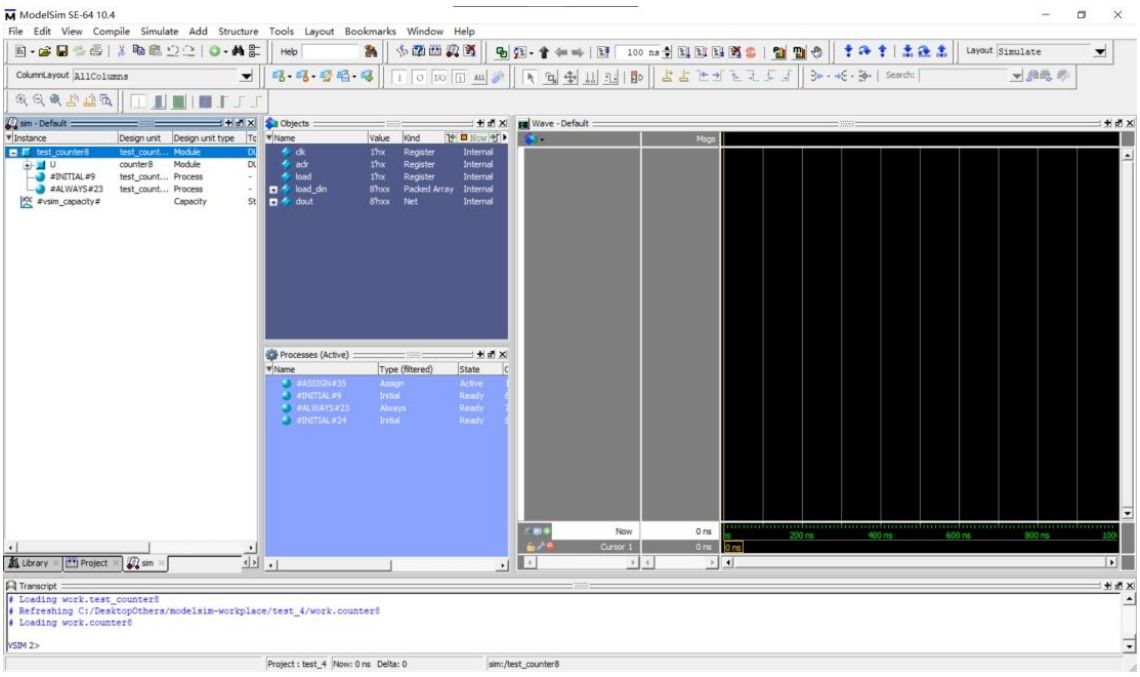
5) 再去掉复选框“Enable optimization”
6) 展开“work”目录
7) 选择 Test Bench 文件，这个例子中是“test_counter8”
8) 点击右下角“Save”
9) 如果不关闭优化选项的话，有时候 ModelSim 软件会报错，导致不能正常进行仿真



1.2.4 进入仿真界面

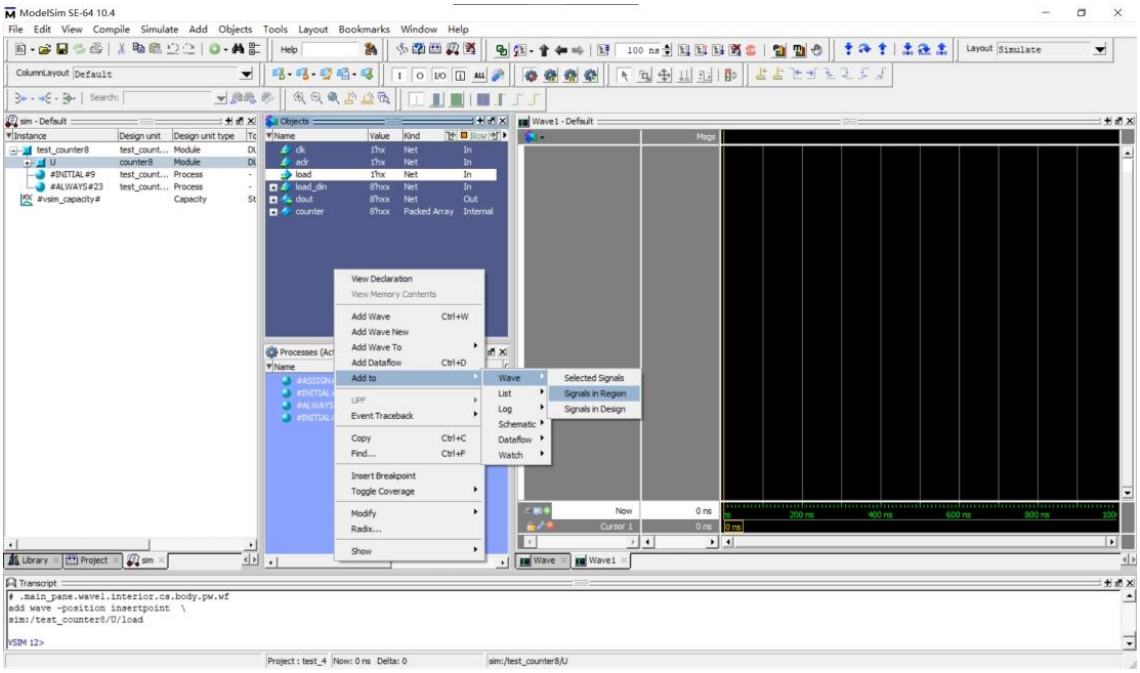
双击新建的仿真配置文件

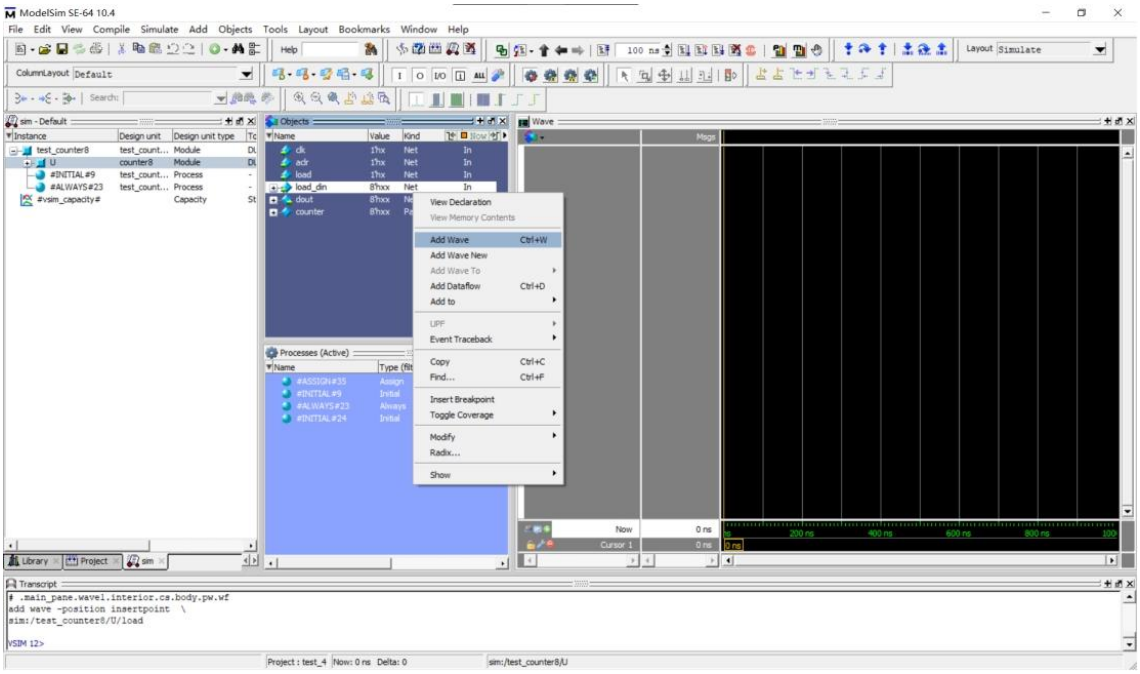




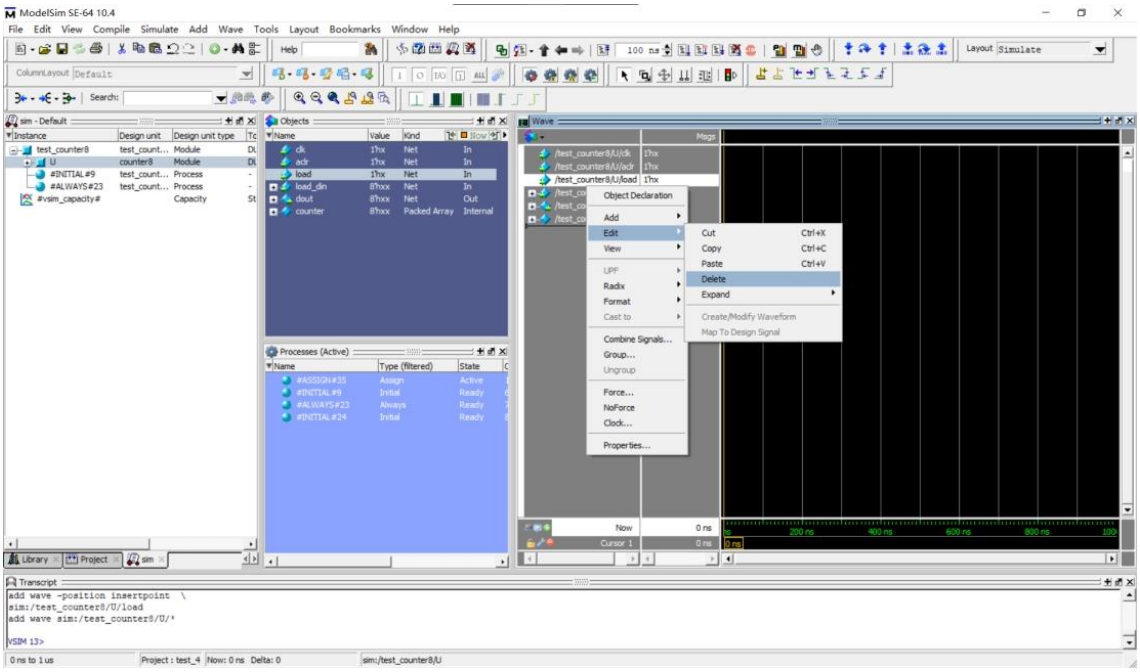
1.2.5 根据需要添加波形显示

1) 在左侧“sim”选项卡中，选择需要添加波形显示的模块
2) 右击“Objects”的空白部分，“Add to->Wave->Signal in Region”，以添加该模块所有信号量。也可以右击某个信号量，“Add Wave”。

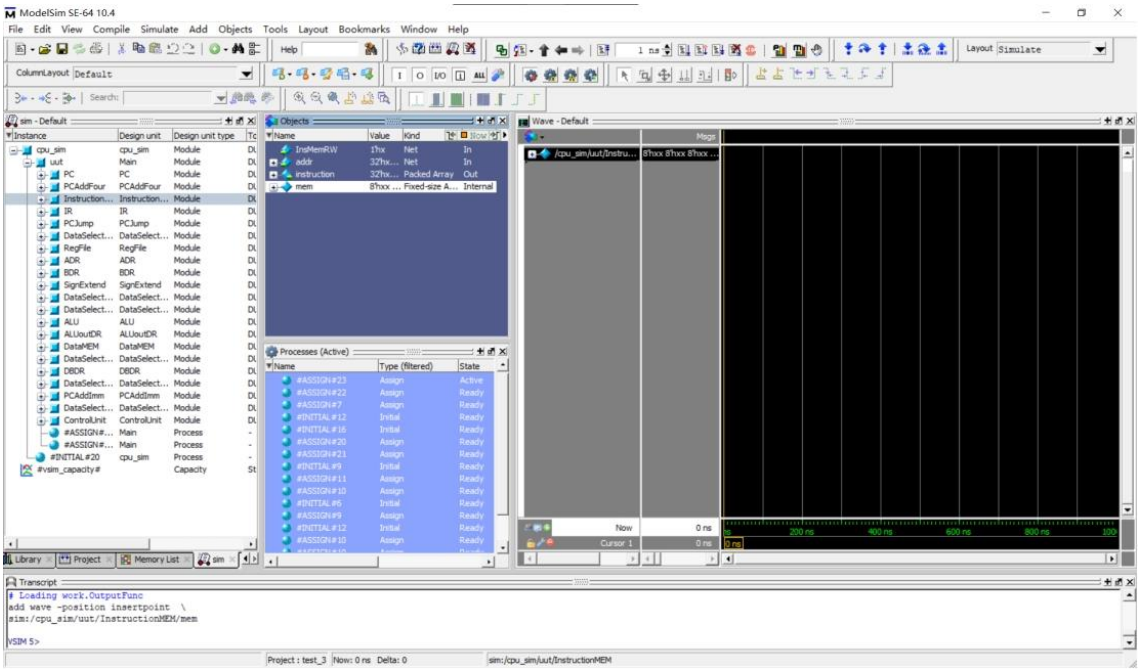




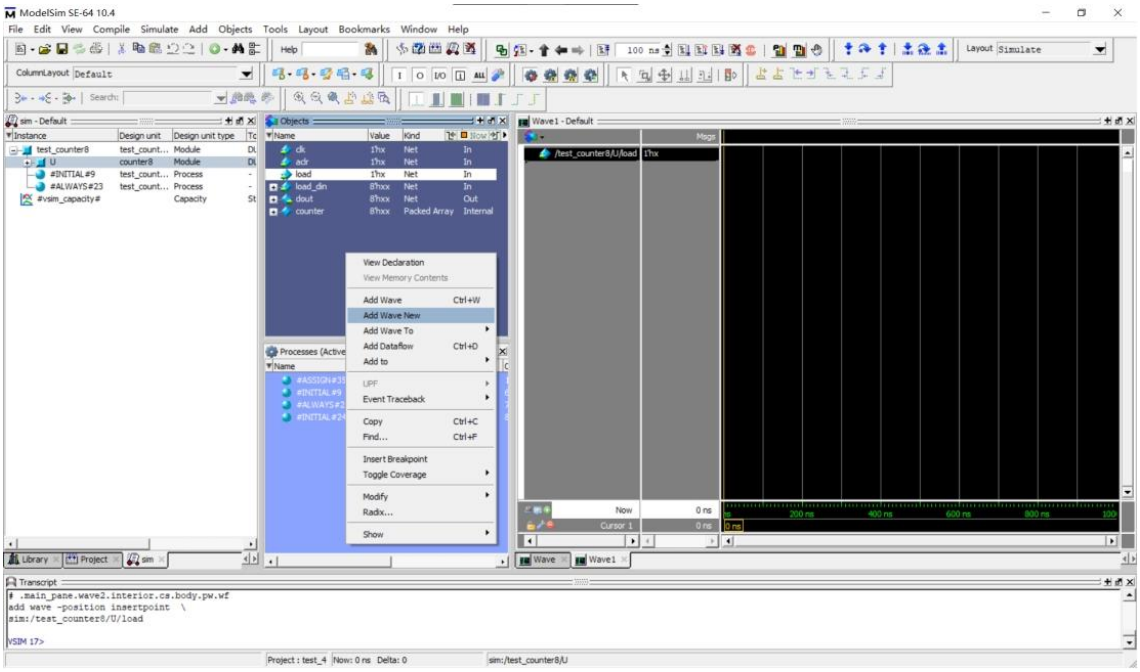
3) 一个信号量可以重复被添加波形显示，可以右击示波器某个信号，对其进行编辑



4) 如果有多个模块，可以同时将不同模块的信号量添加到一个示波器

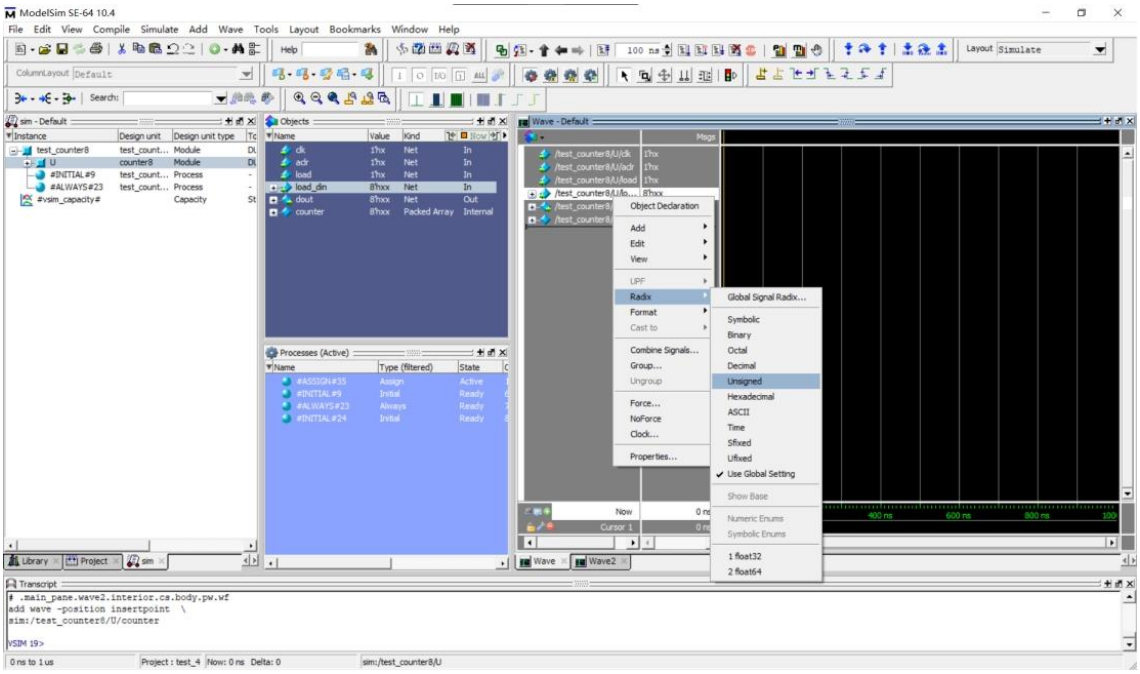


5) 如果示波器不够大，也可以新建第二个示波器。右击“Objects”的空白部分，“Add Wave New”



1.2.6 根据需要修改信号类型

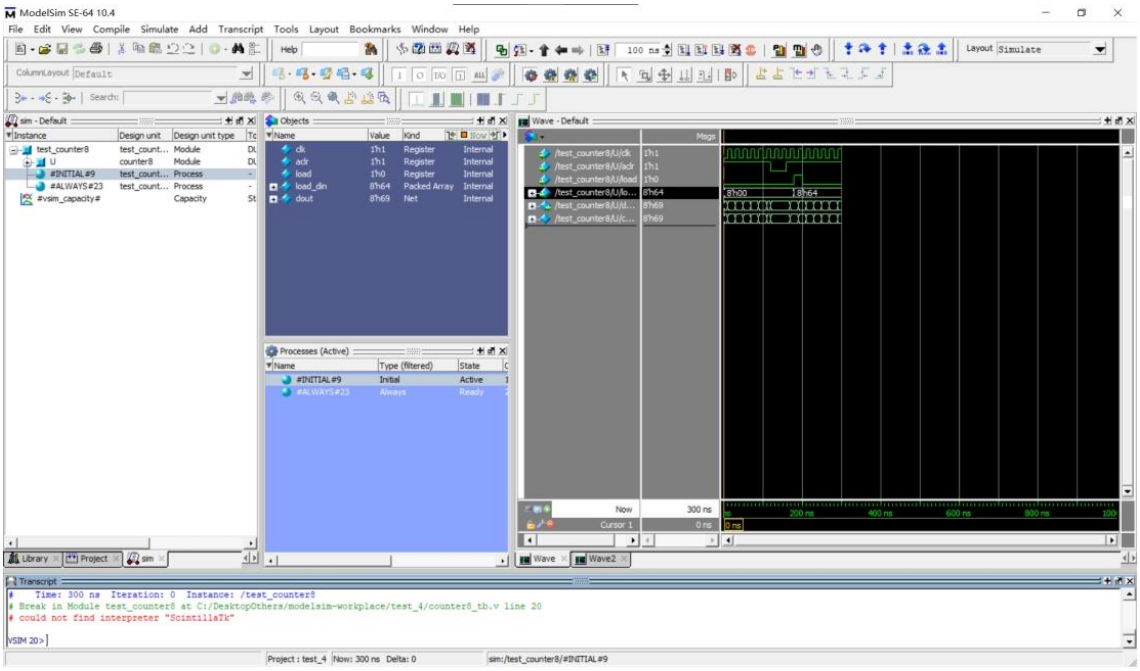
右击示波器某个信号，“Radix->……”



1.2.7 开始仿真

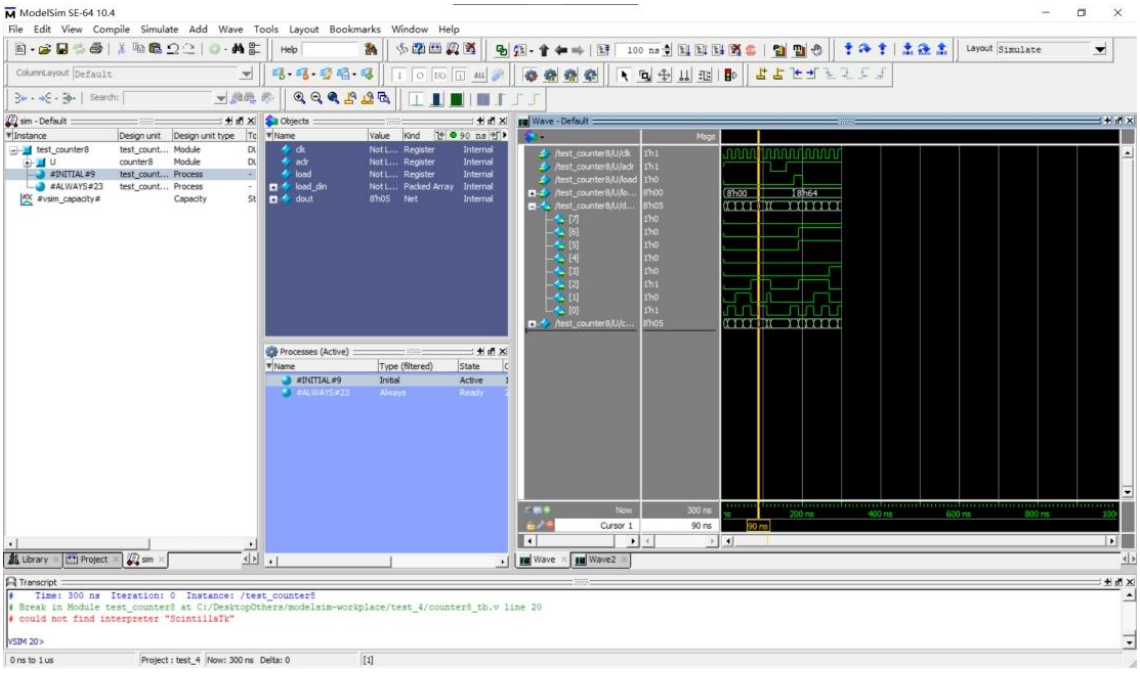
在控制台输入开始仿真命令“run 1us”。时间根据自己设计而定。仿真结束后，在示波器上出现了波形显示。如果想要新添加信号，需要重新仿真（步骤 12）后才能看到

新的波形。

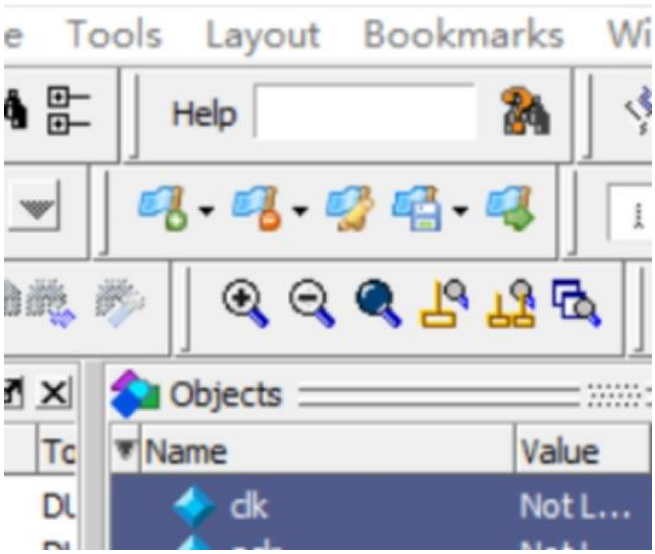


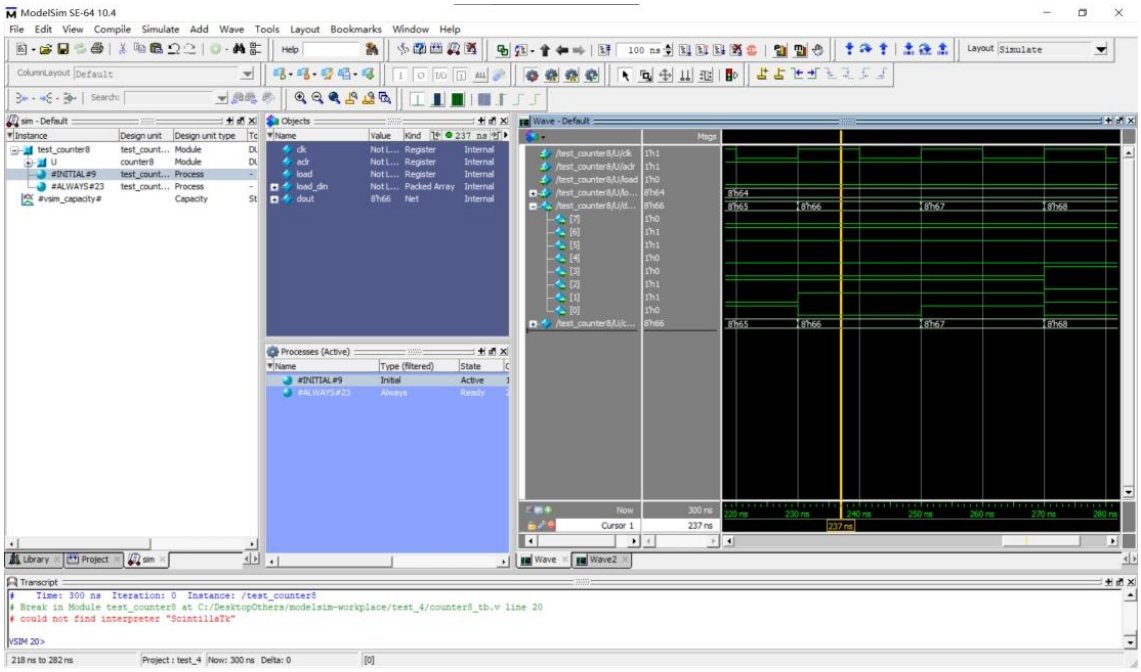
1.2.8 查看波形

1) 点击示波器，左右滑动，可以看到某时刻数值



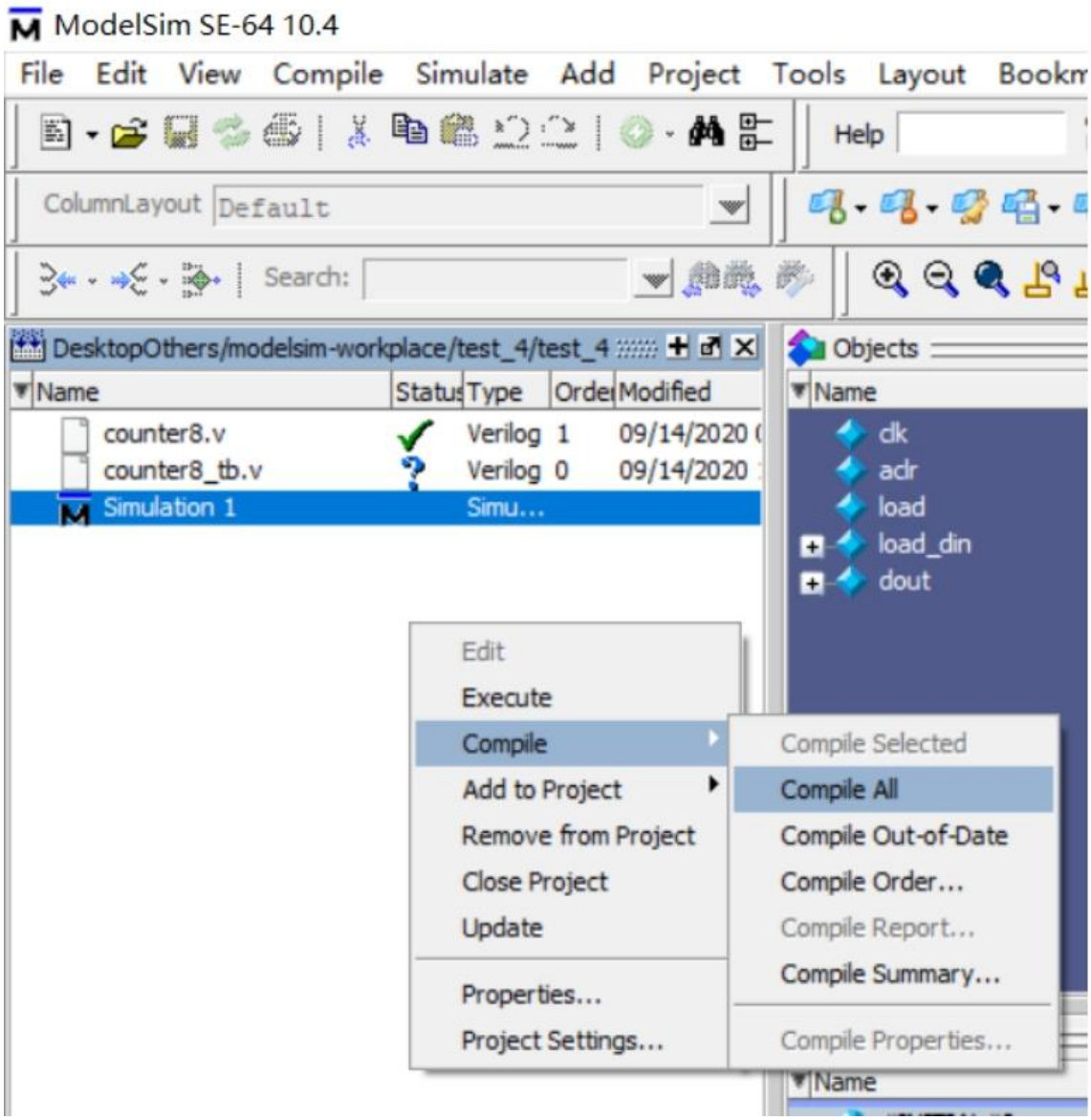
2) 先点击示波器以选中焦点，再点击工具栏的放大镜，可以缩小放大时间轴。



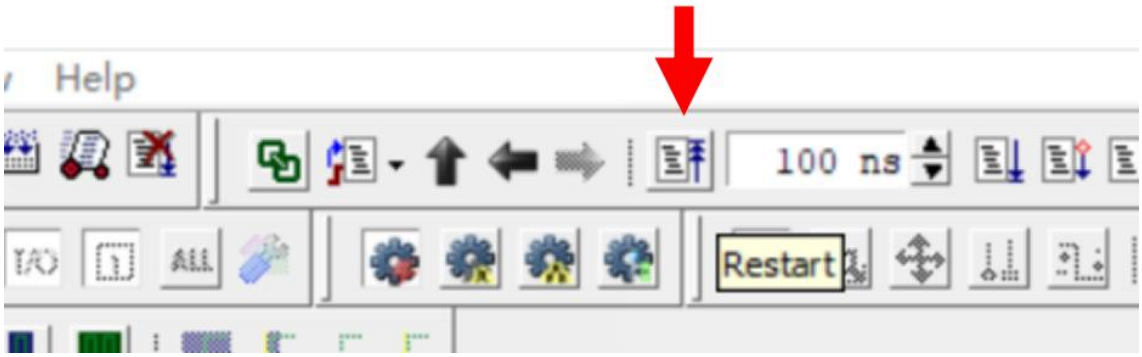


1.2.9 修改、编译、重新仿真

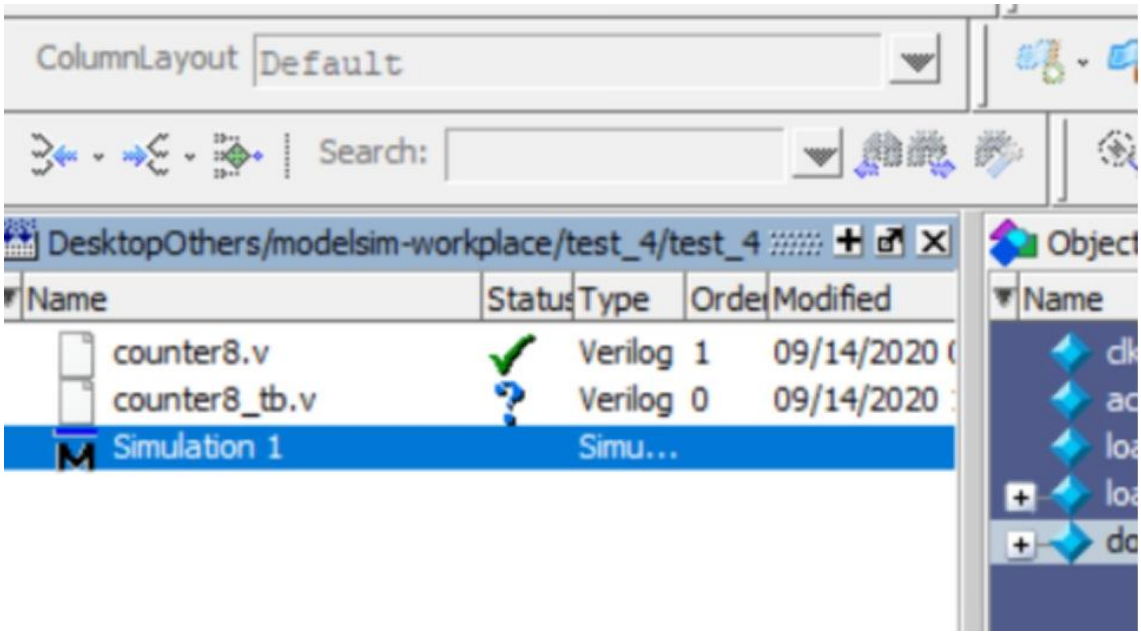
1) 如果出现逻辑错误，需要打开编辑器修改代码。
2) 修改完成后，选中左侧“Project”选项卡，可以看到被修改的文件“Status”变为了问号，需要重新编译（步骤 5）。



3) 点击工具栏的“Restart”按钮后，再进行仿真



4) 也可以双击仿真配置文件从头开始仿真（不建议，除非工程发生了重大改变）。即重新选择信号量及数据类型。



5) 退出仿真，控制台命令，“quit -sim”

1.2.10 打开已有项目

1) 菜单栏，“File->Open…”

2) 找到项目目录，选择文件类型，“Project Files (*.mpf)”

3) 打开 mpf 文件即可打开现有项目

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：

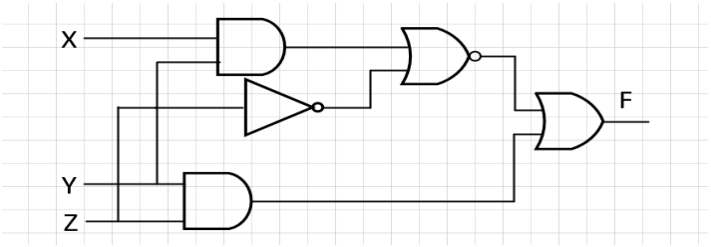


图 2-1 有门电路组成的电路图

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。

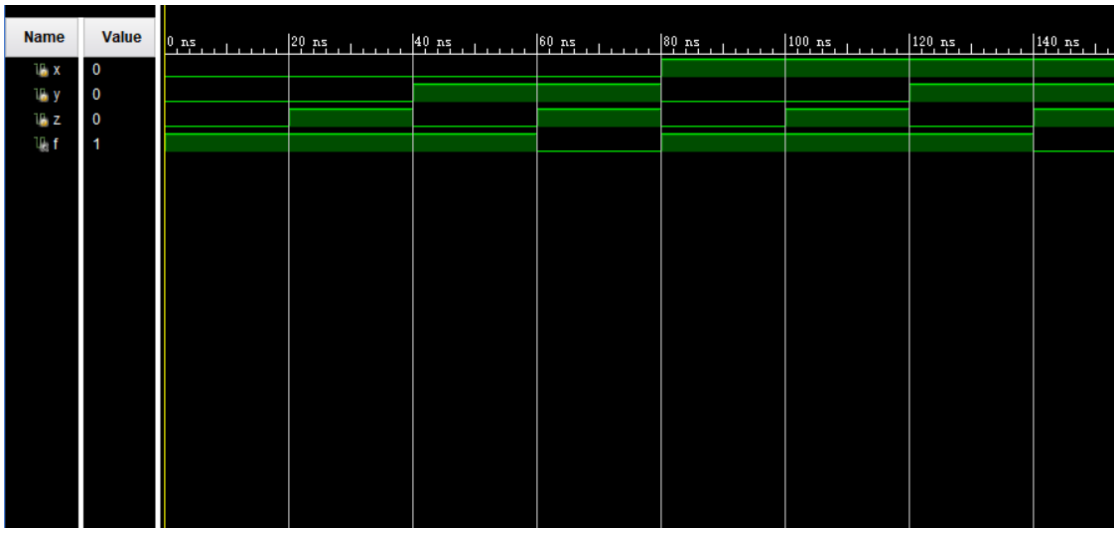
3.2 实验演示

门电路如右图所示，则其设计文件如下：

```
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。