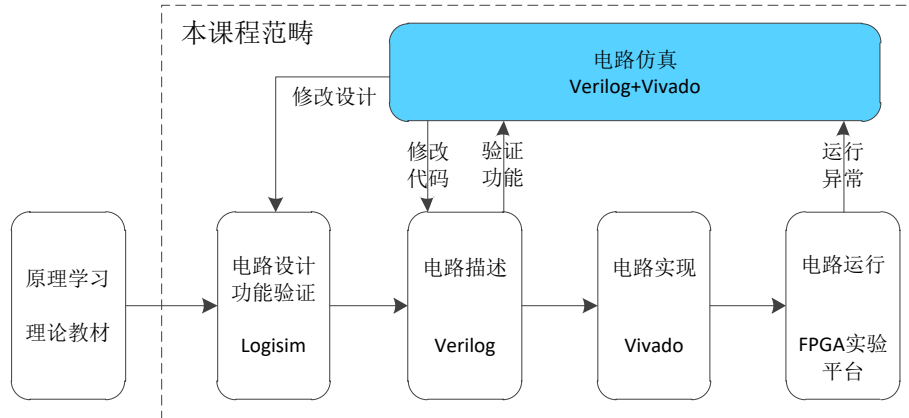


# 实验 05 使用 Vivado 进行仿真

## 简介



用 Verilog 完成电路的编码工作以后，如何确保电路功能正确呢？直接使用 Vivado 工具进行综合，并烧写到实验平台是一种最直接的办法，但是这种做法效率并不高，首先 Vivado 工具的综合过程比较慢，再者如果电路功能不正确，也无法快速定位问题所在，随着电路复杂度的增加，这一情况会更加凸显。因此我们更加推荐的是先通过 Vivado 提供的仿真功能对电路进行仿真。

用 Logisim 验证电路功能大体可分三个步骤：1. 在 Logisim 中完成电路的绘制工作；2. 为电路端口提供输入测试值；3. 在 Logisim 中观察电路运行状态（内部及输出信号）来确定电路功能是否正常。用 Vivado 进行仿真也需要上述 3 个步骤，只不过功能主体有所不同。如下表所示。

步骤	1. 电路设计	2. 构造输入	3. 观察输出
图例			
Logisim	原理图输入	鼠标点击	Logisim 中观察信号值
Vivado	Verilog 代码	Verilog 仿真文件	Vivado 中观察波形

## 实验目的

熟悉 Vivado 软件的下载、安装及使用

学习使用 Verilog 编写仿真文件

学习使用 Verilog 进行仿真，查看并分析波形文件

## 实验环境

PC 一台

vlab.ustc.edu.cn

Vivado 工具

## 实验步骤

### Step1. 下载并安装 Vivado 环境

用户可直接登录 VLAB 系统使用已经配置好的实验环境，如下图所示。为方便使用，也可为 Vivado 创建桌面快捷方式。

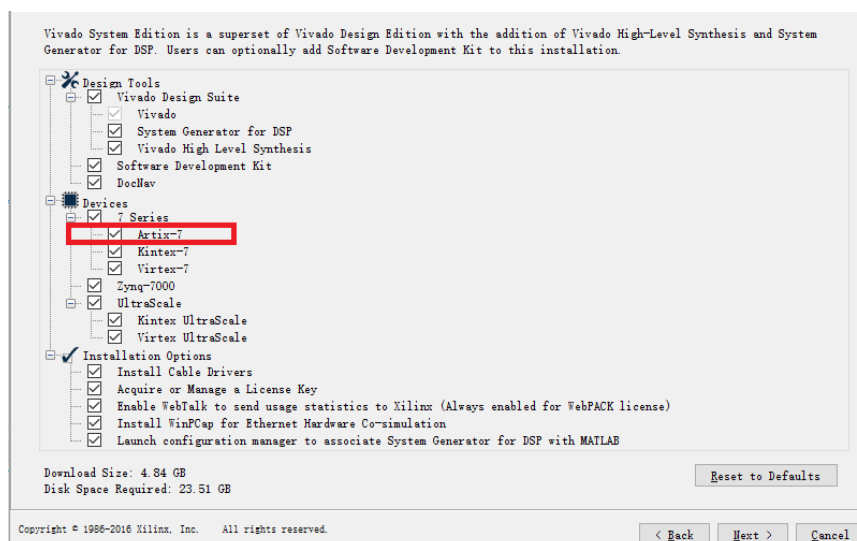


如用户希望自行安装，可登录赛灵思官方网站([www.xilinx.com](http://www.xilinx.com))或者我们的课程主页 ([vlab.ustc.edu.cn](http://vlab.ustc.edu.cn))，选择与用户操作系统相匹配的 vivado 版本，下载并安装。实验中心机房的电脑也已经预装了 Vivado 软件，可以直接使用。

在安装过程中，为减少软件占用的存储空间，我们可以只选择支持 Airtex-7 系列芯片，其它器件系列及功能能取消的全部取消勾选，

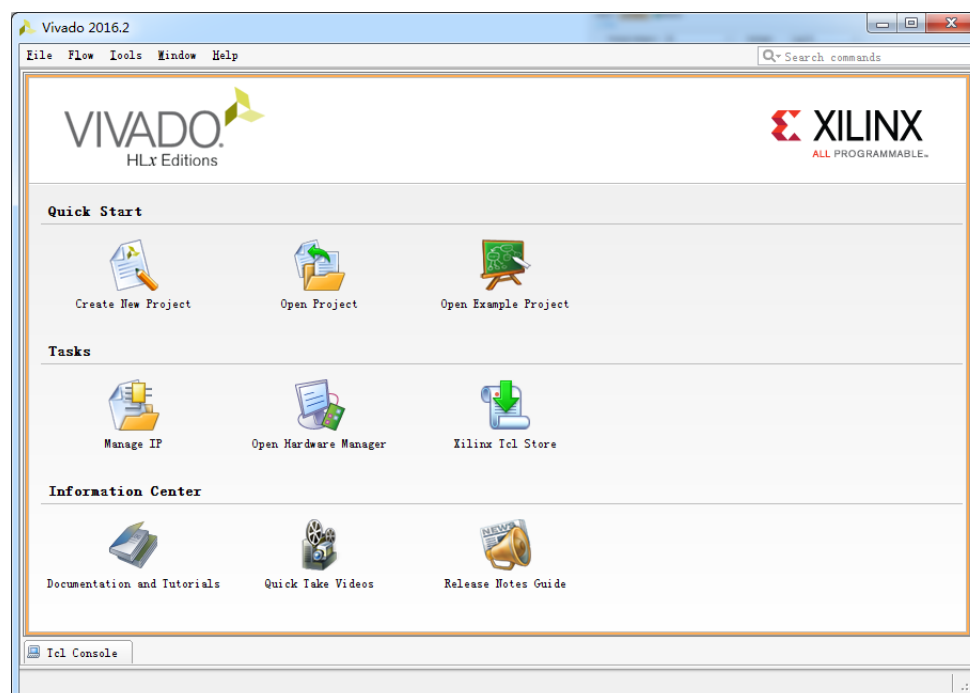
如下图所示，因为我们的目标实验平台采用的芯片是 Aritex-7 系列的一款。其它选项都选择默认选项即可。

特别注意:Vivado 的安装路径必须为不含中文和空格的纯英文路径。



## Step2. 建立 Vivado 工程

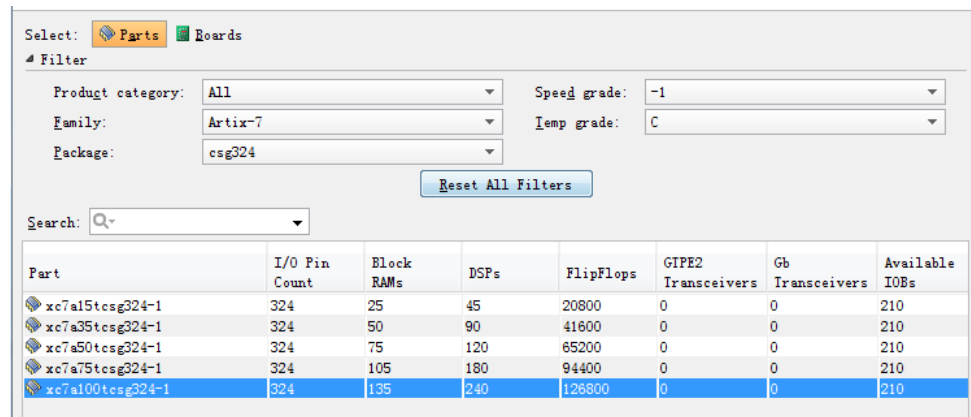
Vivado 的主界面如下图所示，可以通过这个界面快速的新建工程或者打开已有工程，另外，该页面右侧会显示最近使用工程的列表。



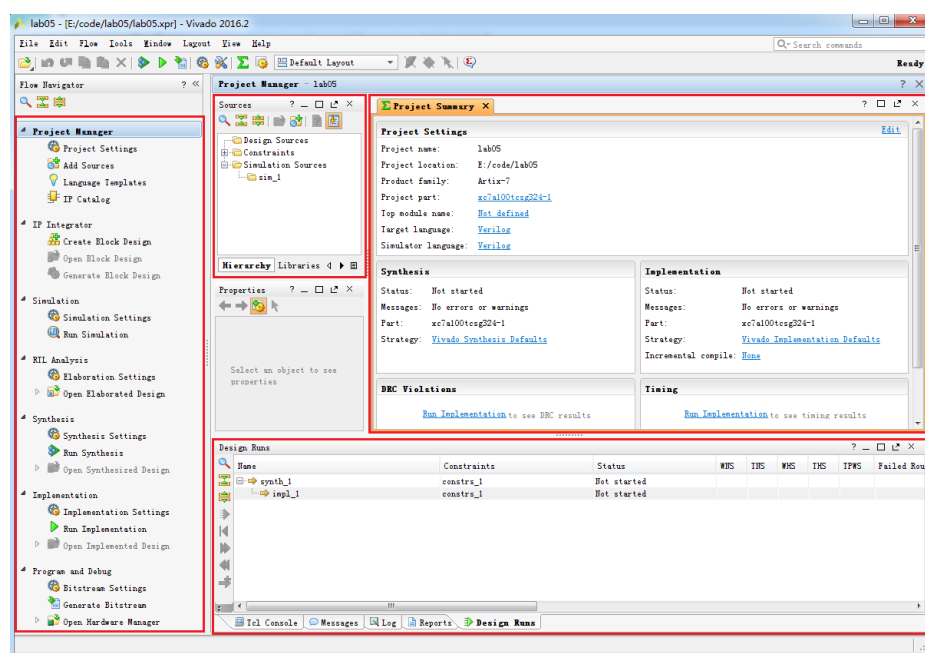
单击“Create New Project”以启动工程向导，按照向导提示建立工

程，一直点击 Next 完成工程的创建。

需要注意：工程路径应为不含空格的纯英文路径、“Default Part”页面选择 xc7a100tcsg324-1 型号的器件。

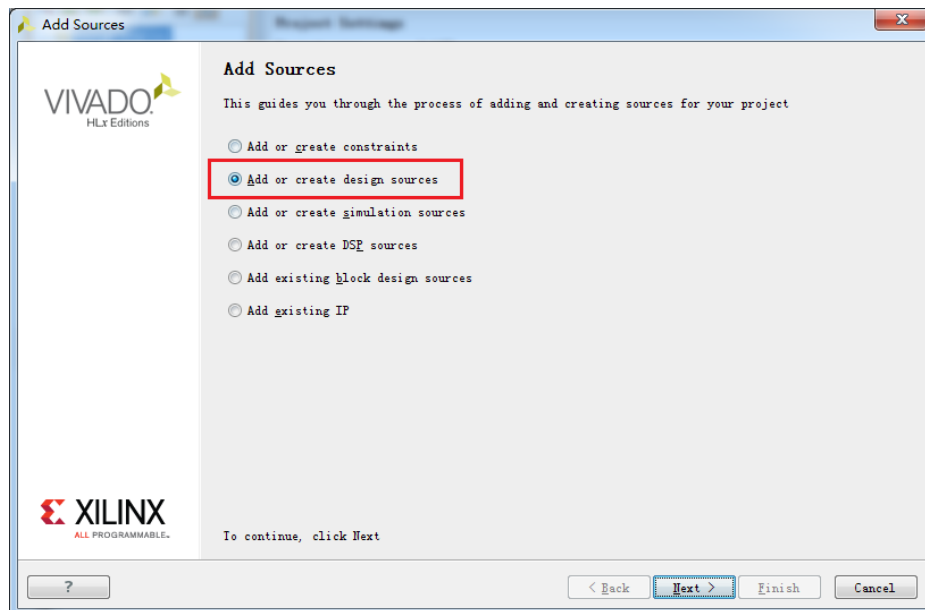


工程创建完毕后的界面如下图所示，主要包含 4 大区域，其中“Project Manager”为工程管理区域，可以完成添加代码、仿真、综合、烧写 FPGA 等一系列操作。“Sources”区域显示代码层级列表，可分为设计文件、约束文件和仿真文件三组，本次实验中会用到设计文件和仿真文件，约束文件在后续实验中会讲到。“Project Summary”显示工程的各种基本信息。“Design Runs”显示工程状态。

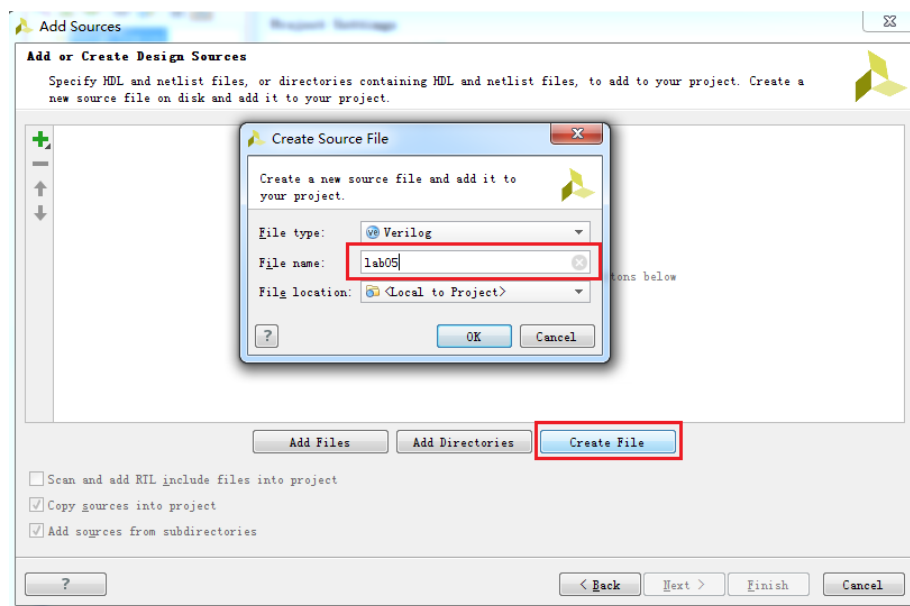


### Step3. 添加 Verilog 设计文件

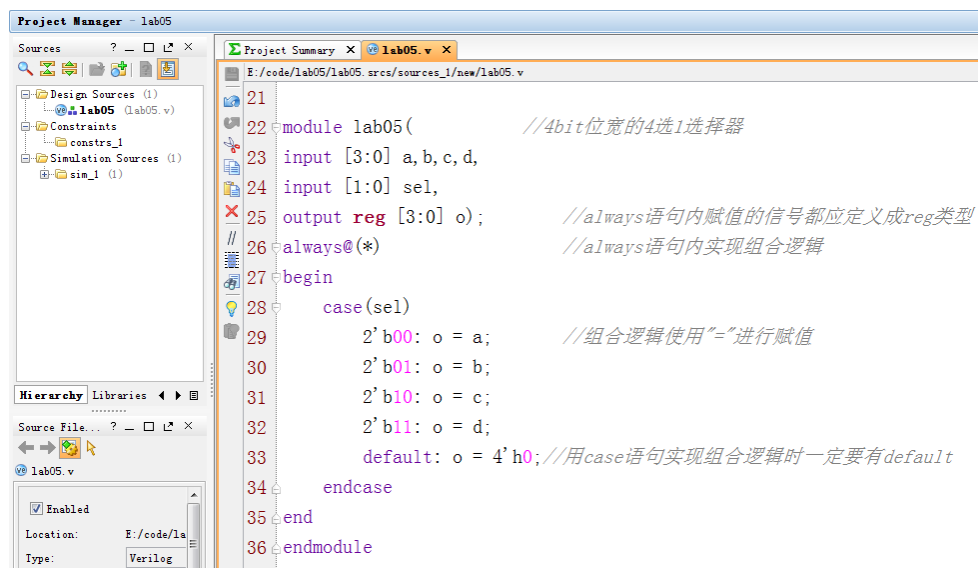
单击“Add Source”按钮添加文件，根据准备添加的文件类型选择对应的选项，包括：约束文件、设计文件、仿真文件等，此处我们需要添加 Verilog 设计文件，因此选择“Add or create design source”



点击“Create File”，输入文件名，并一直按确认键，完成 Verilog 源文件的创建。



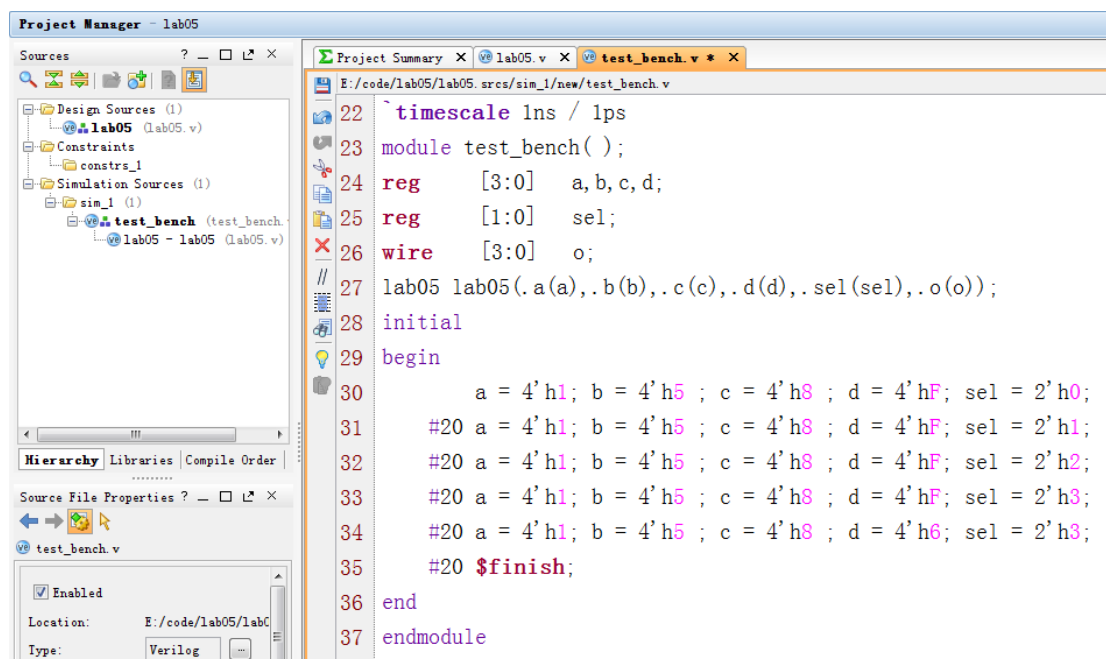
在生成的 Verilog 文件中，输入如下的代码。



该代码是一个 4bit 的四选一选择器，读者现在应该已经具备熟练阅读上述代码的能力，如果在阅读代码时存在困难，请重新学习前续实验。

#### Step4. 添加仿真文件

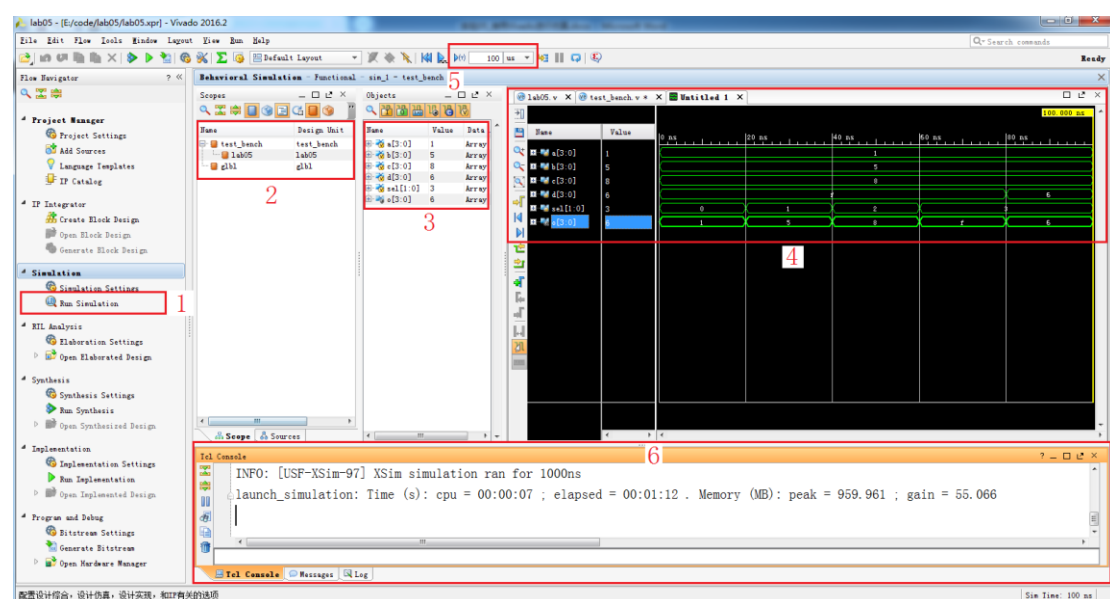
现在，我们要在 Vivado 中添加 Verilog 仿真测试文件，如下图所示。与 Step3 不同的是，在创建文件时，需要选择“Add or create simulation sources”选项。



由上述仿真代码可以看出，Verilog 仿真文件与 Verilog 设计文件有些不同。第一，仿真文件不需要输入输出信号，所有的信号都是模块的内部信号。第二，在仿真文件内对被测试模块进行实例化，并对被测试模块构造输入信号。第三，仿真文件只用于仿真，最终不会被综合成电路，会经常用到“initial”等 Verilog 设计文件中不会用到的关键字或语法，这些语法很多是不可综合的。

## Step5. 波形仿真

点击“Run Simulation”运行仿真工具，会出现如下图所示的界面，其中“Scopes”区域显示的是各实例化模块的层级关系。“Objects”区域显示的是选中模块的所有端口和内部信号。最右侧是波形显示区域，用户可以使用鼠标从“Objects”区域拖拽想要观察的信号。最上方是仿真控制按钮，用户可以通过点击此按钮进行设定时长的电路仿真。最下方是信息显示区域，当代码出现语法错误，无法进行仿真时，用户可以在此查找错误信息。



通过观察波形我们可以发现，该电路的仿真波形符合四选一选择

器的行为特性，Verilog 代码设计正确。

现在，请读者关闭波形仿真窗口，打开前面的 Verilog 设计文件，将其中的 “input [1:0] sel,” 改成 “input sel,” 重新进行仿真，观察波形结果。我们会发现其波形不符合四选一选择器的行为。实际上，信号位宽不匹配是 Verilog 代码编写过程中经常出现的一个错误，这类错误不会导致语法错误，阅读代码时也不容易发现，但通过仿真工具可以比较容易的定位到。

## Step6. Verilog 仿真文件常用语法

在 Vivado 中新建一个工程，加入下面的仿真文件，进行仿真，观察各信号的波形

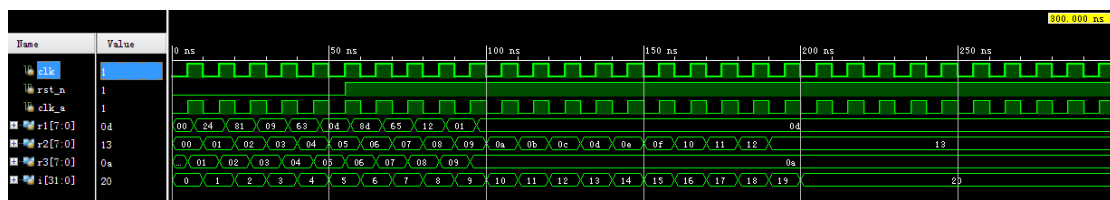
```
module test_bench();
reg clk, rst_n, clk_a;
reg [7:0] r1, r2, r3;
integer i;
initial clk = 0;
always #5 clk = ~clk;
initial
begin
    rst_n = 0;
    #55 rst_n = 1;
    #245 $stop;
end
initial
begin
    clk_a = 0;
    forever #5 clk_a = ~clk_a;
end
initial
begin
    r1 = 0;
    repeat (10)
    begin
        @(posedge clk);
        #2 r1 = $random%256;
```



```

        end
    end
    initial
    begin
        for(i=0;i<20;i=i+1)
        begin
            r2 = i;#10;
        end
    end
    initial
    begin
        r3=0;
        while(r3<10)
        begin
            @(posedge clk);
            r3 = r3 +1;
        end
    end
end
endmodule

```



上述代码中包含了一些新的关键字和语法结构，下面我们对其进行说明。

**initial:** 该关键字与 always 同为过程语句关键字，但与 always 不同的是，initial 语句只执行一次，initial 语句在模拟开始时执行，其语法结构为：

*initial* (时序控制) 过程语句

过程语句：过程语句可以是赋值语句、条件表达式、循环表达式、触发事件、顺序过程语句、并列过程语句等，如：

```

a = b;
if()...;else ...;
for()...;
posedge b;
begin....end;

```

*fork...join;*

**时序控制：**一般用在 `always`、`initial` 关键字后面，或者过程语句内部，常用的时序控制语句有时延控制、电平敏感事件控制和边沿触发事件控制三种。时延控制语句形式为“`#n`”，用于实现 `n` 个时间单位的延时，常用在过程语句内部；电平敏感事件控制语法格式为“`always @ (a, b, c) ...`”，`a, b, c` 三个信号的电平变化时会执行其后的过程语句，一般用来实现组合逻辑，更简洁的一种写法是“`always @ (*) ...`”；边沿敏感事件控制语法格式为“`always@(posedge clk or negedge rst_n)...`”，表示在“`clk`”信号的上升沿或“`rst_n`”信号的下降沿时执行，需要用到表示边沿事件的“`posedge`”“`negedge`”两个关键字。

**循环控制：**在过程语句中可以通过循环语句实现循环控制，主要包括 `forever`、`repeat`、`while`、`for` 四种，具体用法可以在上述代码中学习。

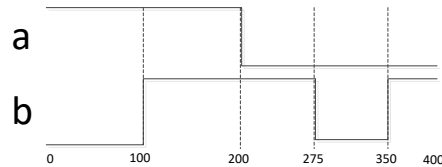
**系统函数：**在 Verilog 仿真文件中支持调用一些系统函数，以提高仿真效率，调用格式为：`$函数名`，如 `$random` 表示生成一个随机数，`$finish` 表示仿真结束，`$stop` 表示停止仿真，`$fopen`、`$fclose` 用于打开和关闭文件，`$fwrite`、`$fwriteb` 等表示写入文件，Verilog 语法中支持的系统函数有很多，本文作为 Verilog 语法的入门介绍，不再一一列举。

本文对 Verilog 仿真中用到的基本语法进行了介绍，熟练掌握这些语法，可以完成绝大多数 Verilog 仿真文件的编写，如果读者想深入学习 Verilog 语法或者通过改进仿真文件提高仿真效率，可自行查

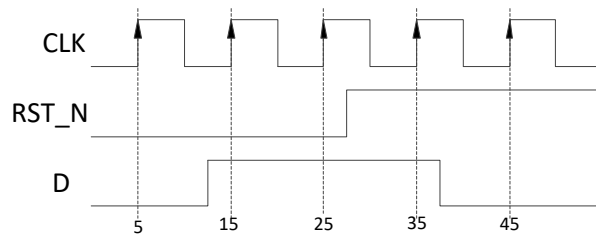
阅相关的 Verilog 语法书籍。

## 实验练习

**题目 1.** 请编写 Verilog 仿真文件，生成如下图所示的波形，并在 Vivado 中进行仿真。



**题目 2.** 请编写 Verilog 仿真文件，生成如下图所示的波形，并在 Vivado 中进行仿真。



**题目 3.** 利用题目 2 中的信号作为以下代码的输入，在 Vivado 中对其进行仿真，并观察仿真波形。

```
module d_ff_r(  
    input clk, rst_n, d,  
    output reg q);  
    always@(posedge clk)  
    begin  
        if(rst_n==0)  
            q <= 1'b0;  
        else  
            q <= d;  
        end  
    end  
endmodule
```

**题目 4.** 设计一个 3-8 译码器，编写仿真测试文件，在 Vivado 中对其进行仿真。要求仿真时遍历所有的输入情况组合，给出源代码和仿真截图。

## 总结与思考

1. 请总结本次实验的收获
2. 请评价本次实验的难易程度
3. 请评价本次实验的任务量
4. 请为本次实验提供改进建议