**实验7**

**Vivado使用**

# **实验介绍**

理论上，FPGA 从编程到下载实现预期功能的过程至少需要经历即 RTL 分析、综合以及实现这几个阶段。它们的简要介绍如下：

1. RTL 分析。RTL（Register Transfer Level，寄存器传输级）指通过描述寄存器到寄存器之间的逻辑功能来描述电路。RTL 分析过程则是将 HDL 硬件描述语言转换成逻辑门电路的过程。
2. 综合（Synthesis）。综合是指将 HDL 转换成较低层次电路结构的过程。低层次电路结构是 FPGA 内部存在的基本逻辑单元，包括查找表 LUT、触发器、RAM 等。这一步得到的电路图比 RTL 更为具体，结构上往往也十分不同。这是因为 FPGA 底层中并没有 RTL 生成的逻辑电路的实现，所以只能用其他逻辑资源进行代替。
3. 实现（Implementation）。综合后生成的电路只表示了逻辑资源之间虚拟的连接关系，并没有规定每个逻辑资源的实际位置以及连线长度等。实现就是一个将综合电路中逻辑资源位置以及连线长度确定的过程。

为了完成上面的步骤，我们需要借助一些专门的工具。从本次实验开始，我们将借助 Vivado 帮助完成 Verilog 的开发流程。在本次实验中，我们将了解到如何使用 Vivado 进行 RTL 分析与仿真操作。

# 实验内容

## 1 Vivado配置

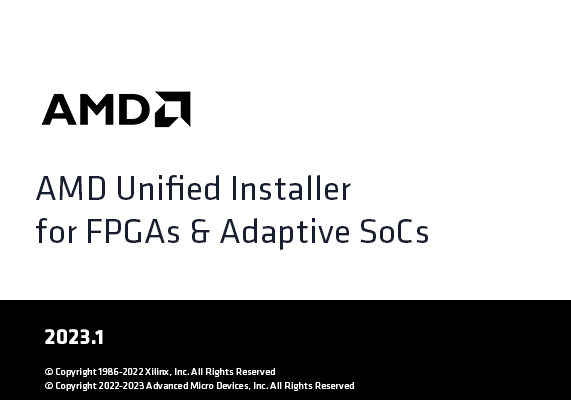
### 1.1 下载与安装

Vivado 是 FPGA 厂商 Xilinx（赛灵思）提供的、用于 FPGA 实现的工具。它可以将硬件描述语言所描述的硬件设计进行编译、综合与实现，将 FPGA 内部本身无序的各种逻辑资源配置成为有序的电路，进而实现数字系统功能。

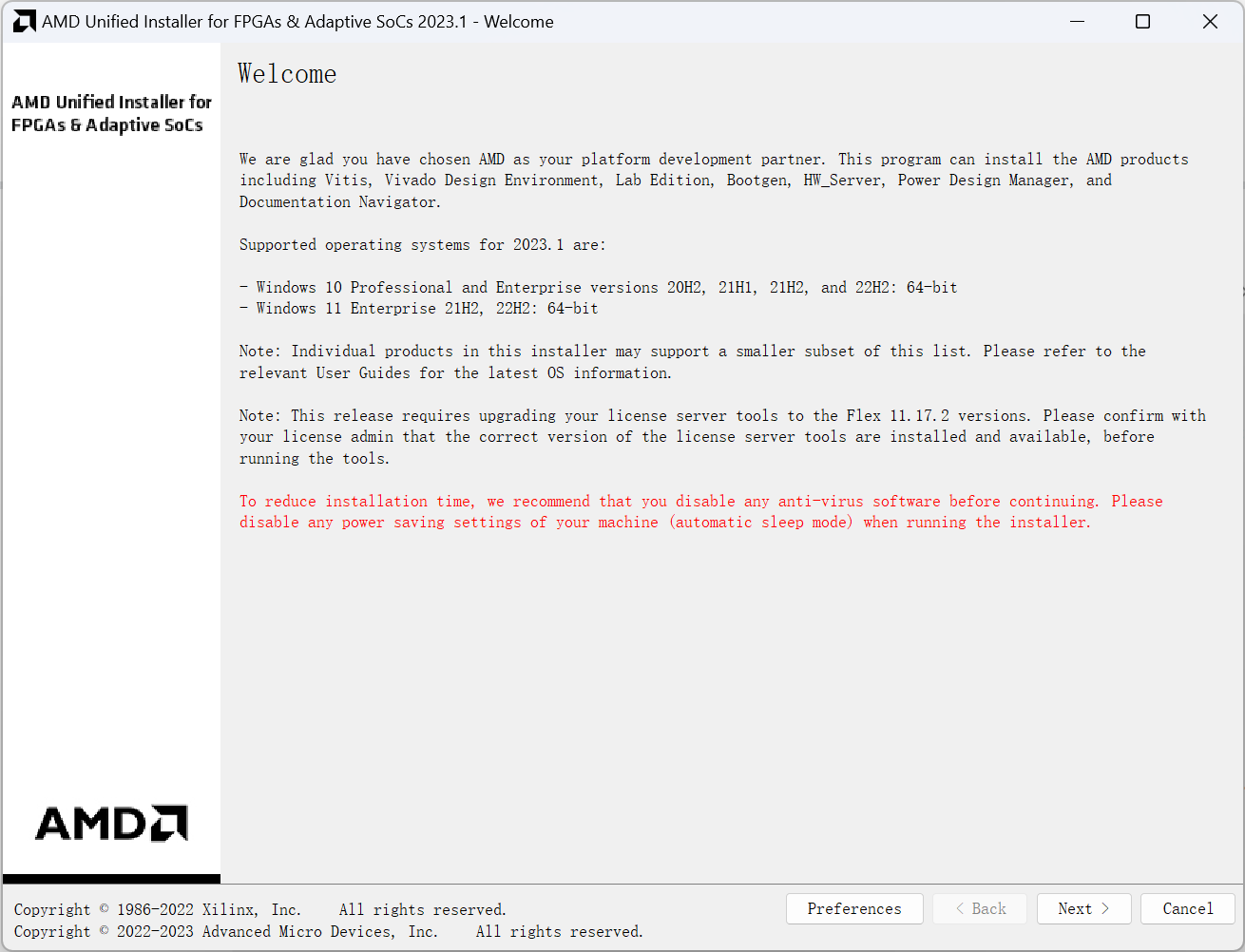
[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/vivado_logo.png)

如果你想在本地安装并使用 Vivado，请确保你是 Windows 或 Linux 系统。2023.1 版本的 Vivado 需要至少 240 G 的可用存储空间（最终占用 35G），2019.1 版本的 Vivado 需要至少 60 G 的可用存储空间（最终占用 20G）。此外，你也可以选择使用 Vlab 上虚拟机的 Vivado 进行后续实验流程。

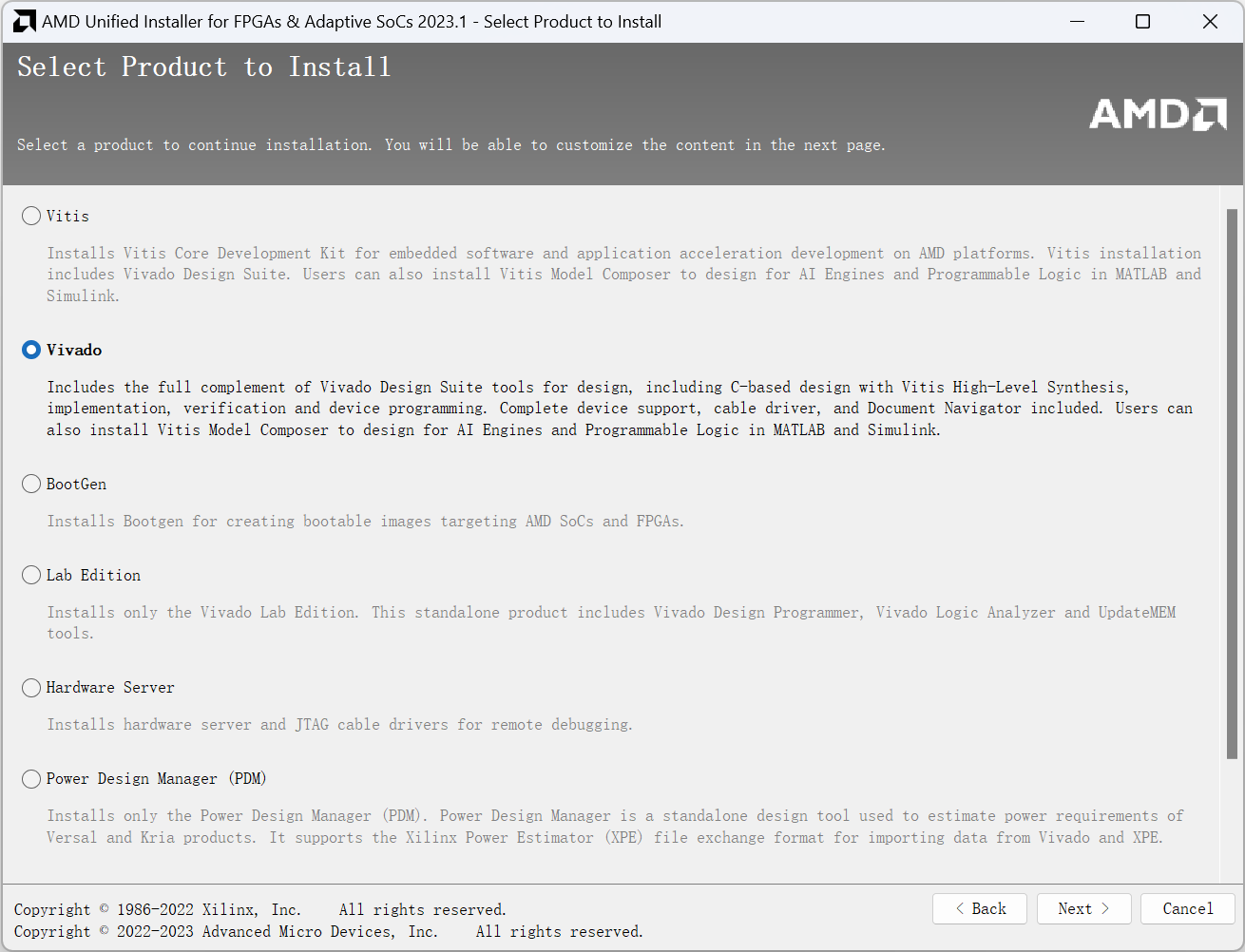
首先从官网获得2023.1版本的安装包。下载并解压完成后，单击文件夹中的 xsetup.exe 程序以开始安装。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_1.png)

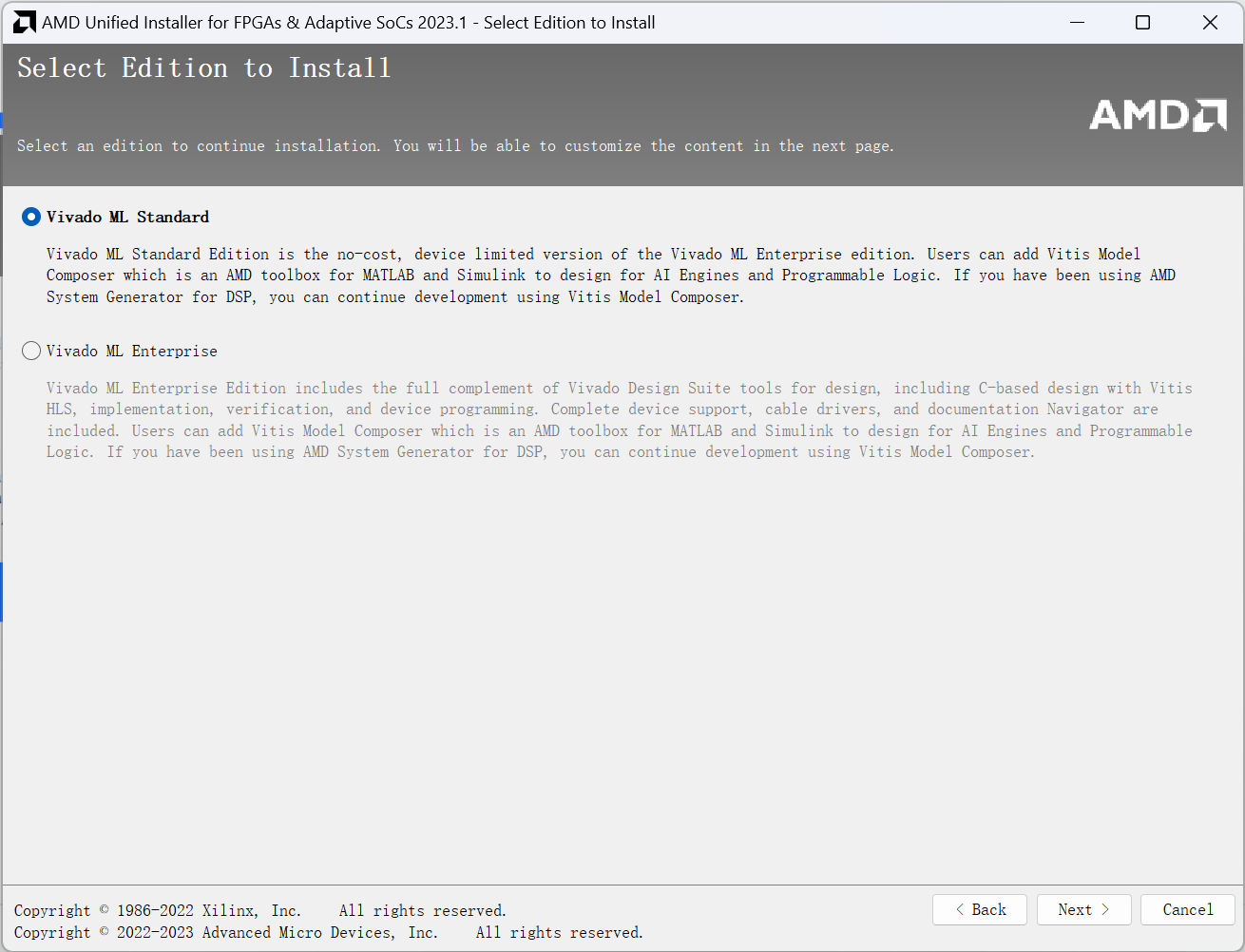
首先进入下图所示的界面。这里简单介绍了 2023.1 版本的 vivado 所支持的操作系统版本。单击 Next。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_2.png)

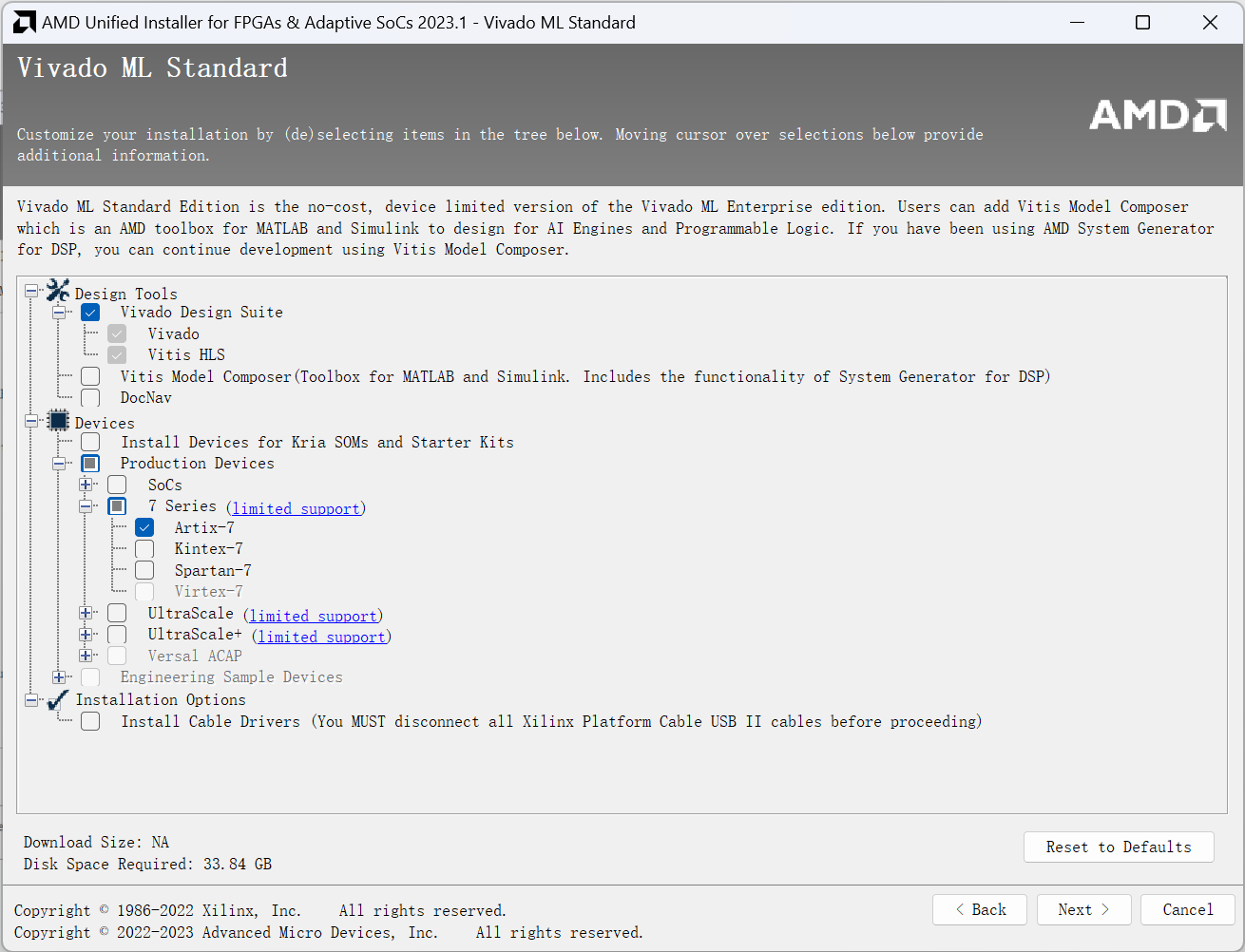
接下来是产品界面。单击选择 Vivado，再单击 Next 以继续安装。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_3.png)

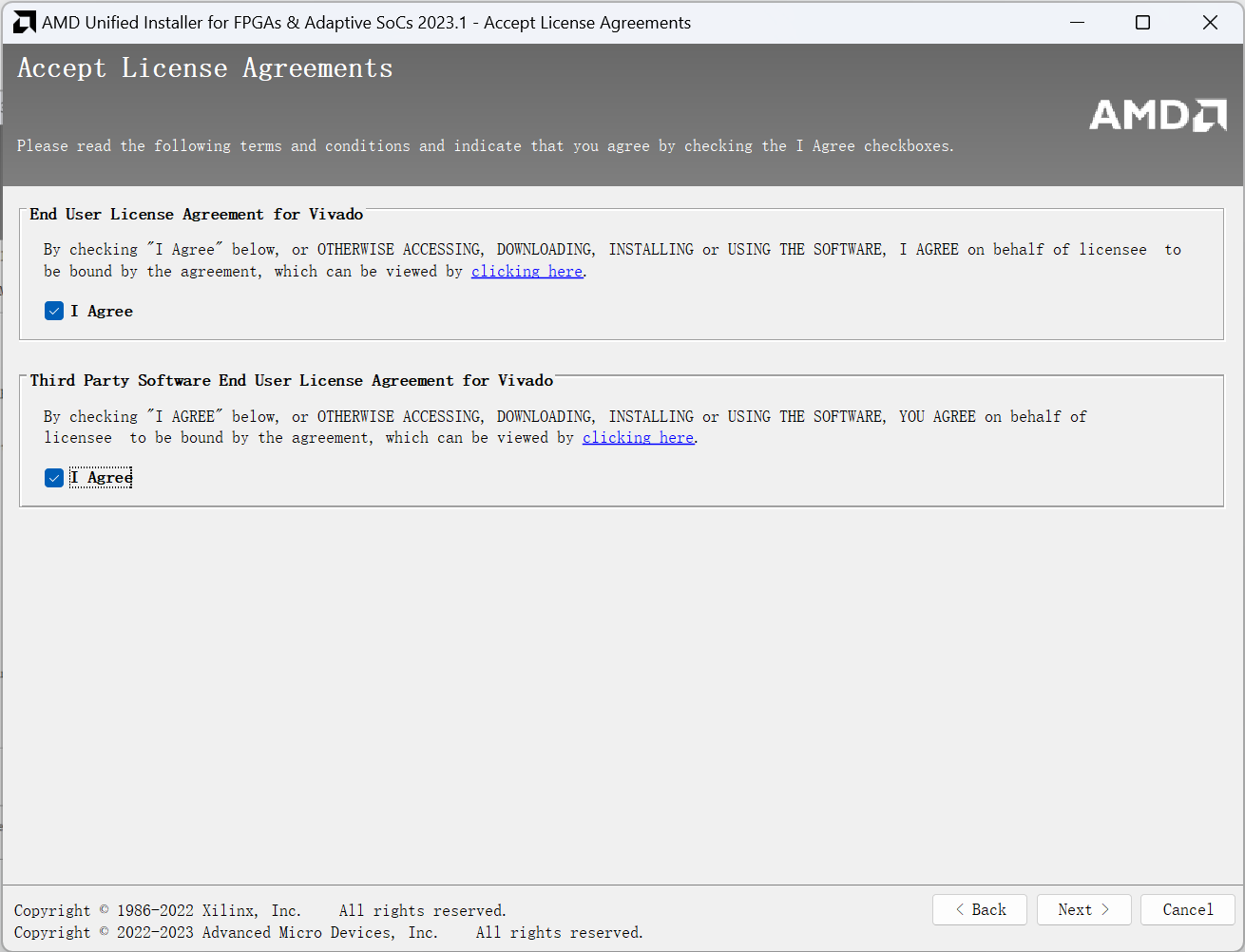
选择安装 Vivado 后，下面会选择安装标准版本（Standard）或者是企业版本（Enterprise）。二者的区别在于企业版本的 Vivado 额外支持高性能的芯片，比如 KU+15P。这里我们选择标准版（Standard）安装即可。单击 Next 以继续安装。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_4.png)

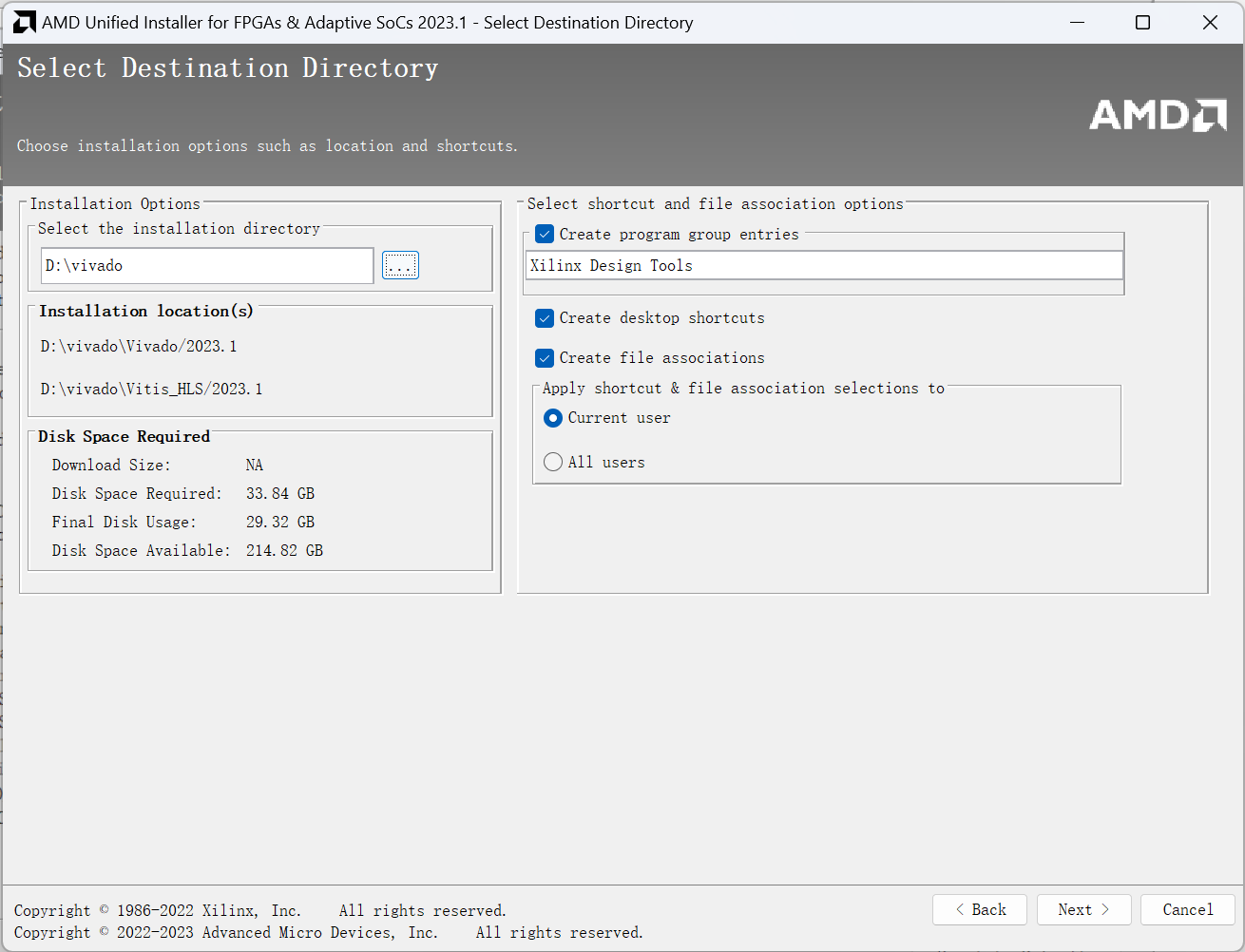
接下来的界面是选择安装组件。本学期数电实验以及下学期计组实验都只需要 A7 芯片即可，因此为了节省空间，可不装其他功能。设定好相关内容后，单击 Next 以继续安装。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_5.png)

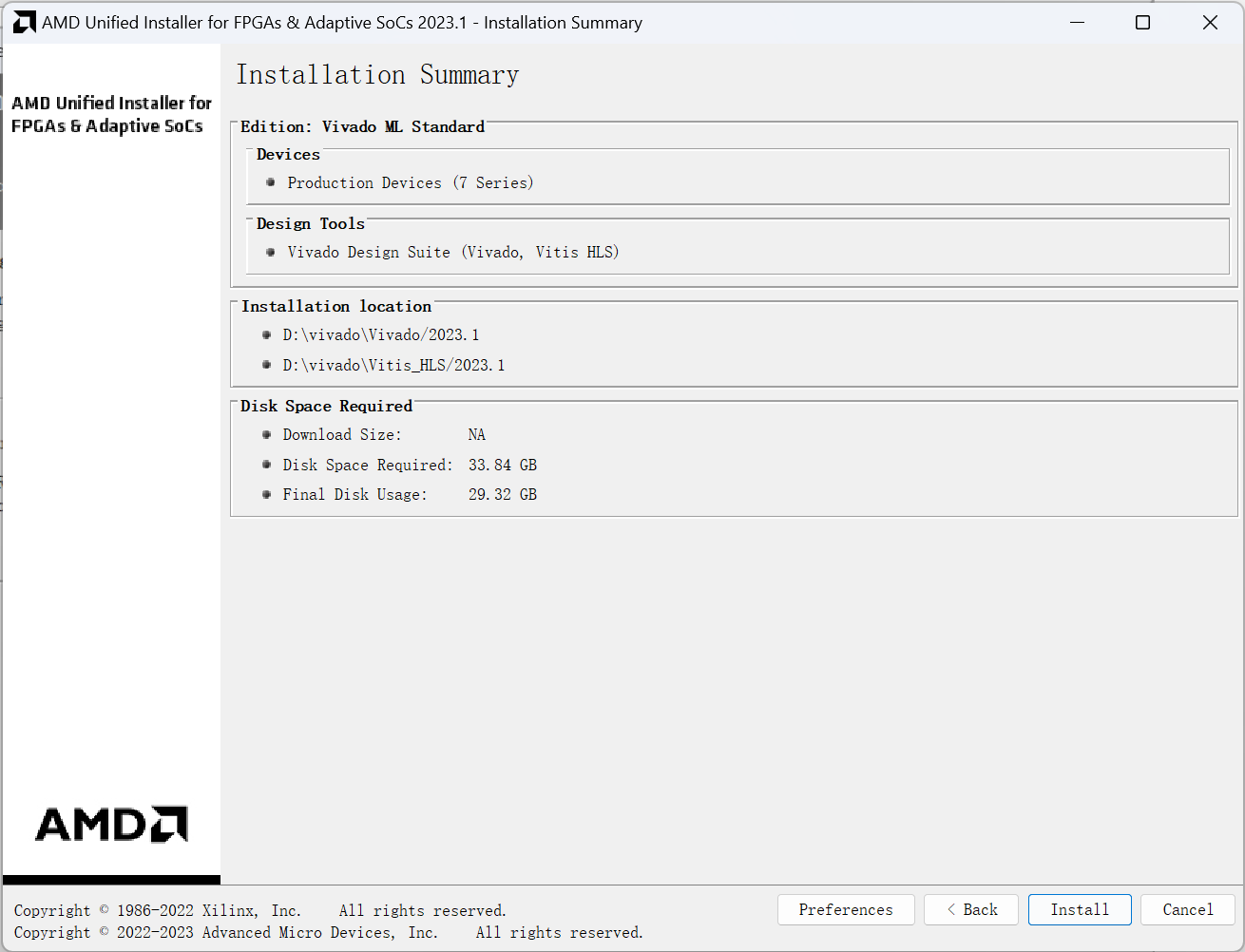
接下来需要同意许可协议。设定好相关内容后，单击 Next 以继续安装。

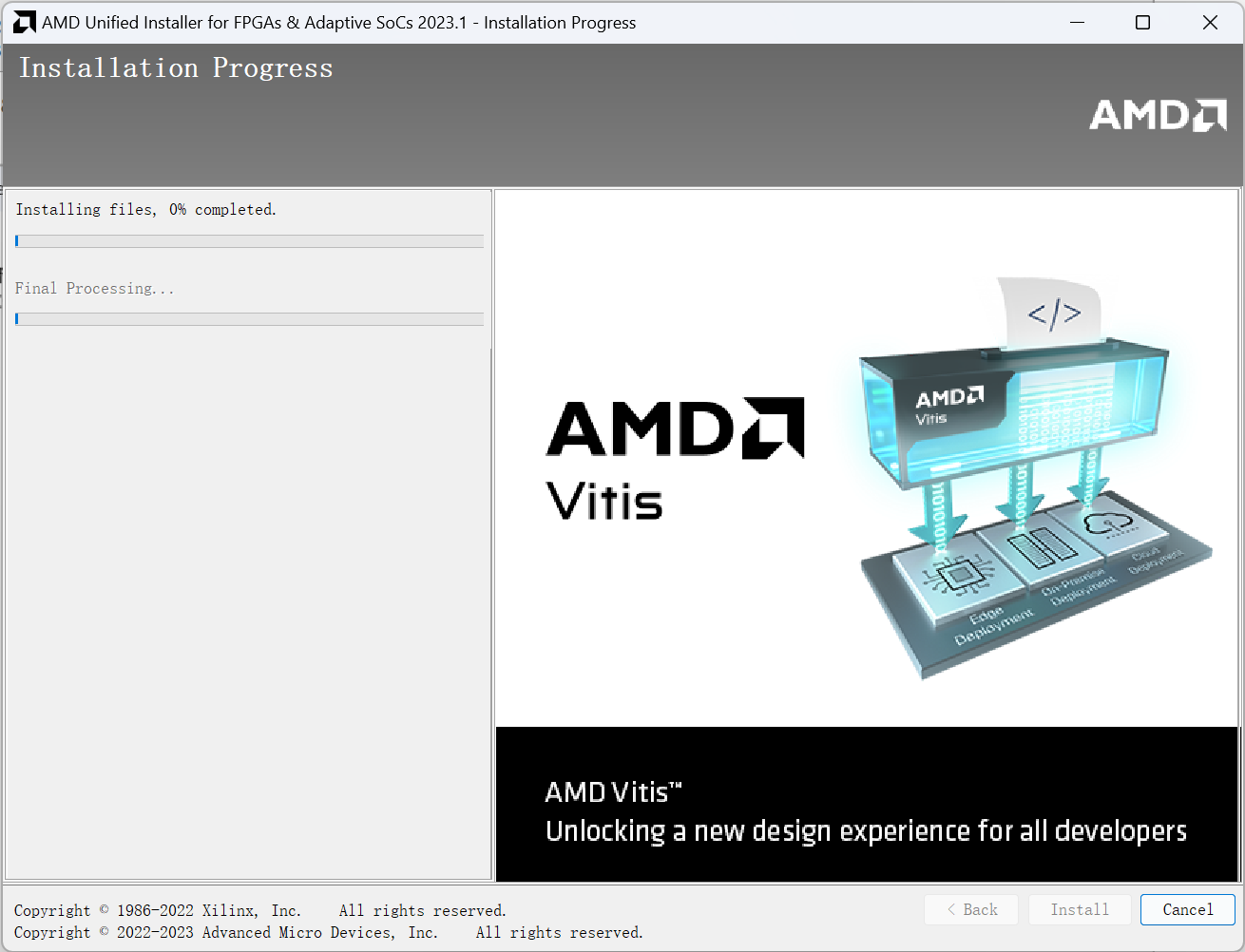
[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_6.png)

接下来是选定安装目录。此时的安装大小超出了 30G，因此不建议直接安装到 C 盘。你可以选择容量充足的位置进行安装。注意：安装目录中不能出现中文与空格字符。设定好相关内容后，单击 Next 以继续安装。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_7.png)

最后是安装概览。检查安装信息无误后，单击 Install 以开始安装。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_8.png)

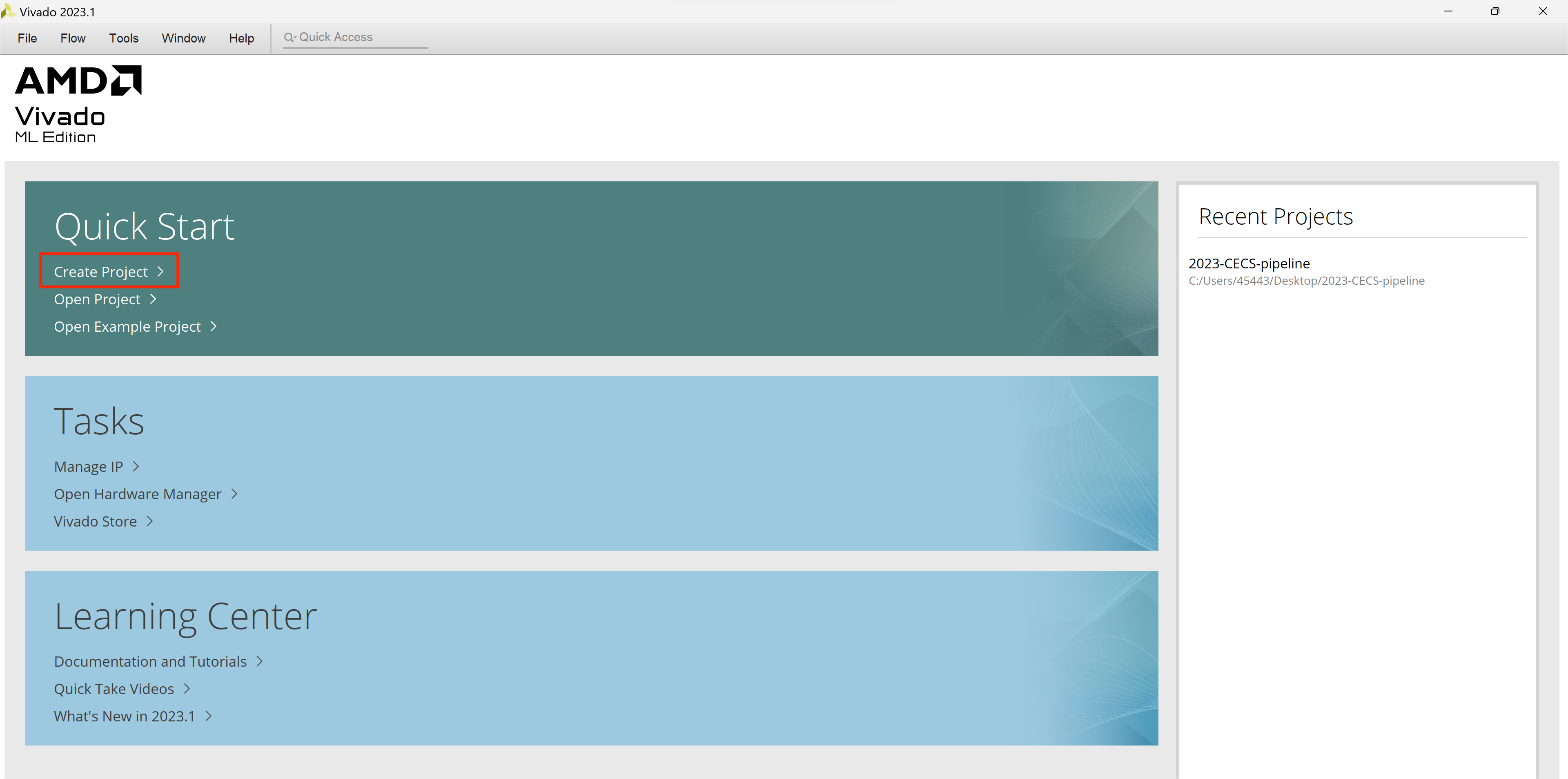
[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_9.png)

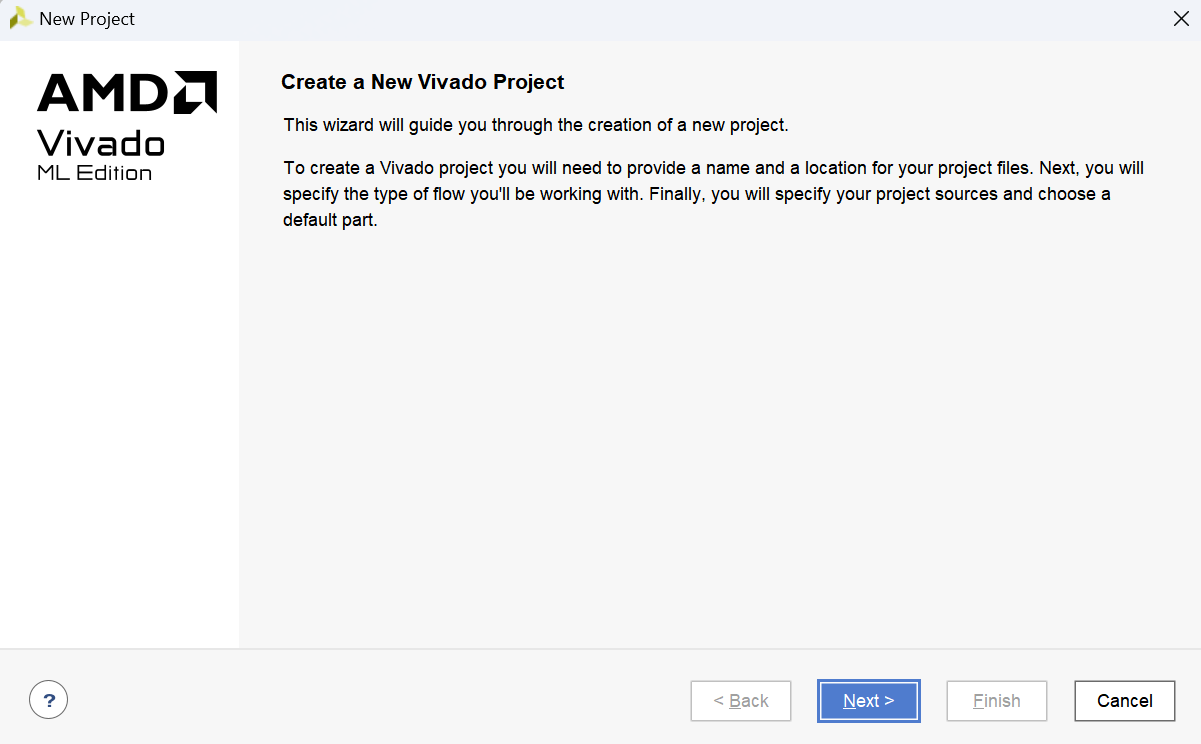
当弹出下面的对话框时，表明安装结束。此时点击桌面上的 Vivado 2023.1 图标即可启动 Vivado。

[](https://soc.ustc.edu.cn/Digital/lab0/figs/vivado/v2023_10.png)

### 1.2 项目创建

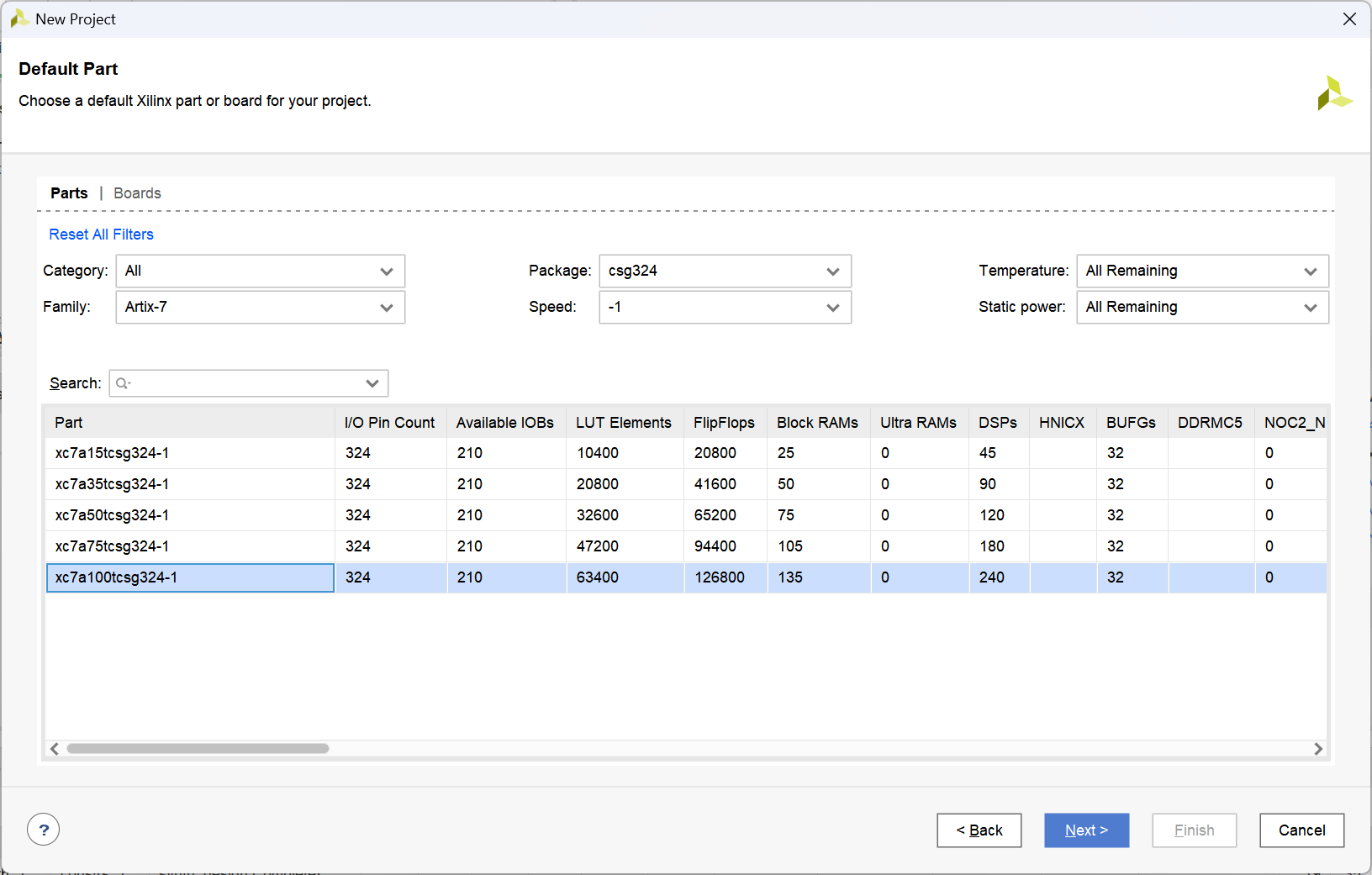
打开 Vivado2023.1，点击 Create Project 新建一个 Vivado 工程。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/vivado_start.png)

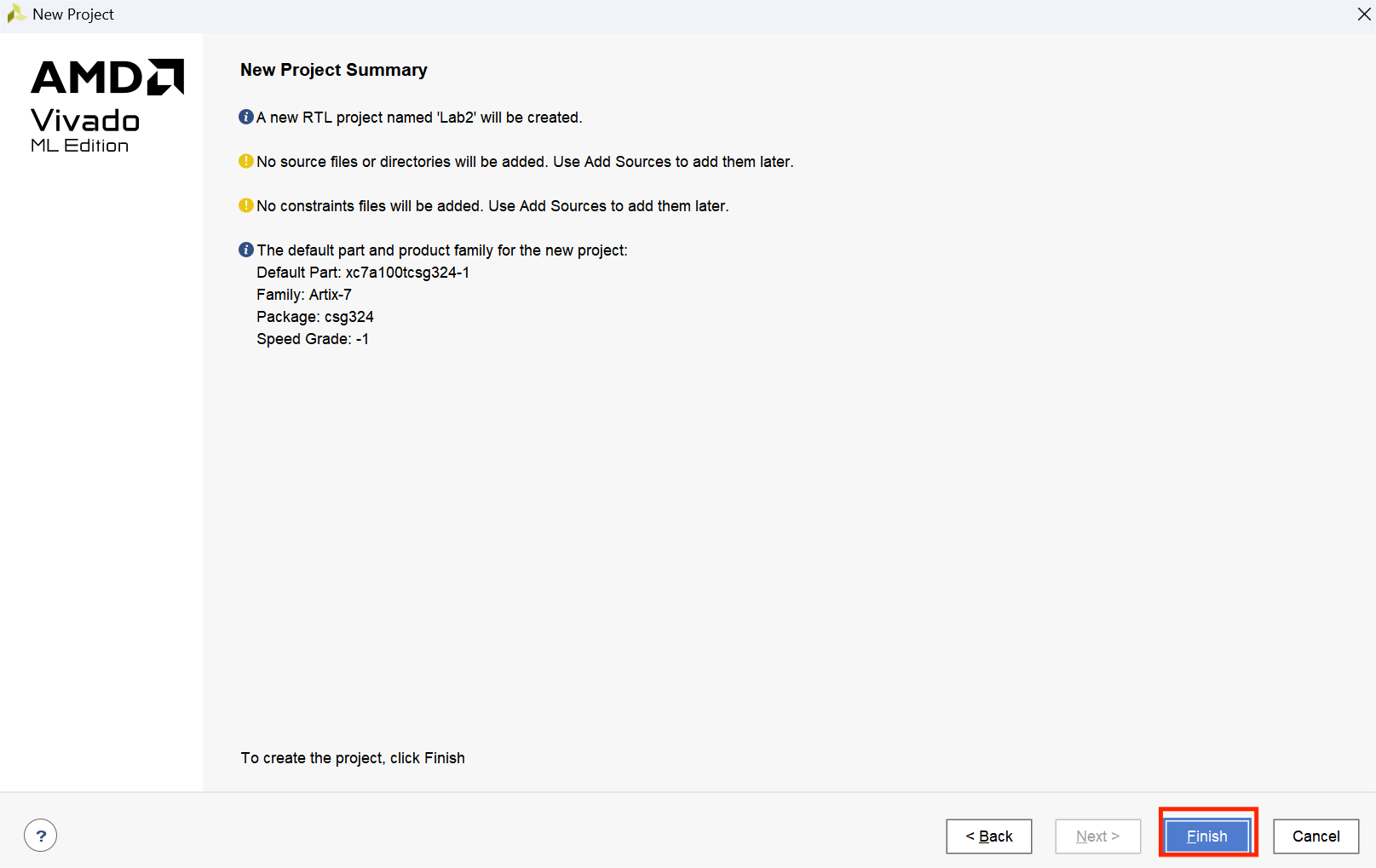
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/create_proj.png)

点击 Next，填写项目名称与项目路径。与安装 Vivado 时类似，请确保你的项目路径与项目名称中不包含中文或空格，否则可能带来意想不到的问题。

设置好项目名称后，一直选择 Next（无需更改其他选项），直到遇到选择芯片型号（Default Part）的界面，按照下图所示的配置就可以找到我们使用的芯片型号：xc7a100tcsg324-1。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/search_part.png)

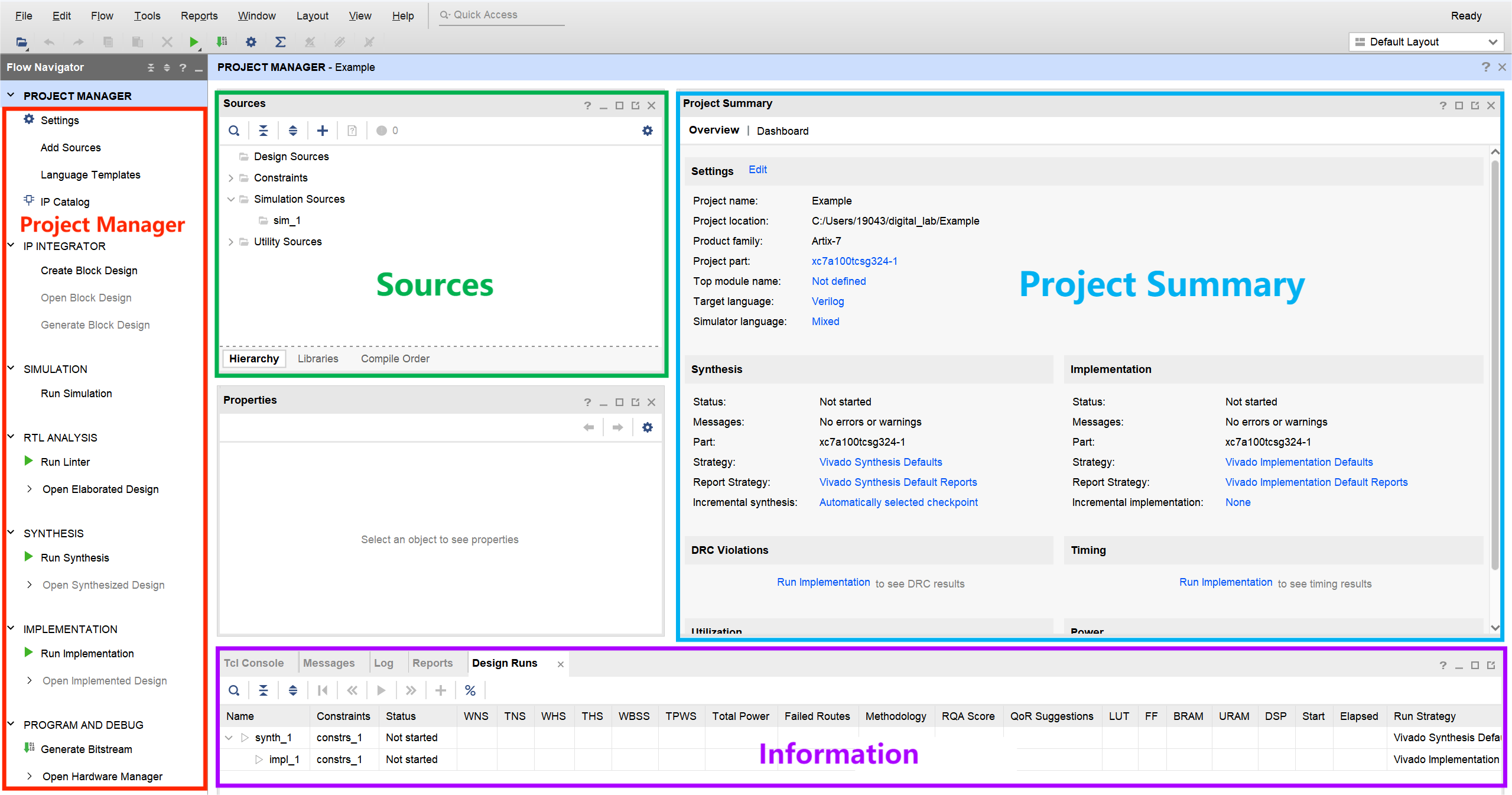
之后继续选择 Next ，最后点击 Finish 就创建好了一个新的工程项目。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/finish.png)

## 2 Vivado开发

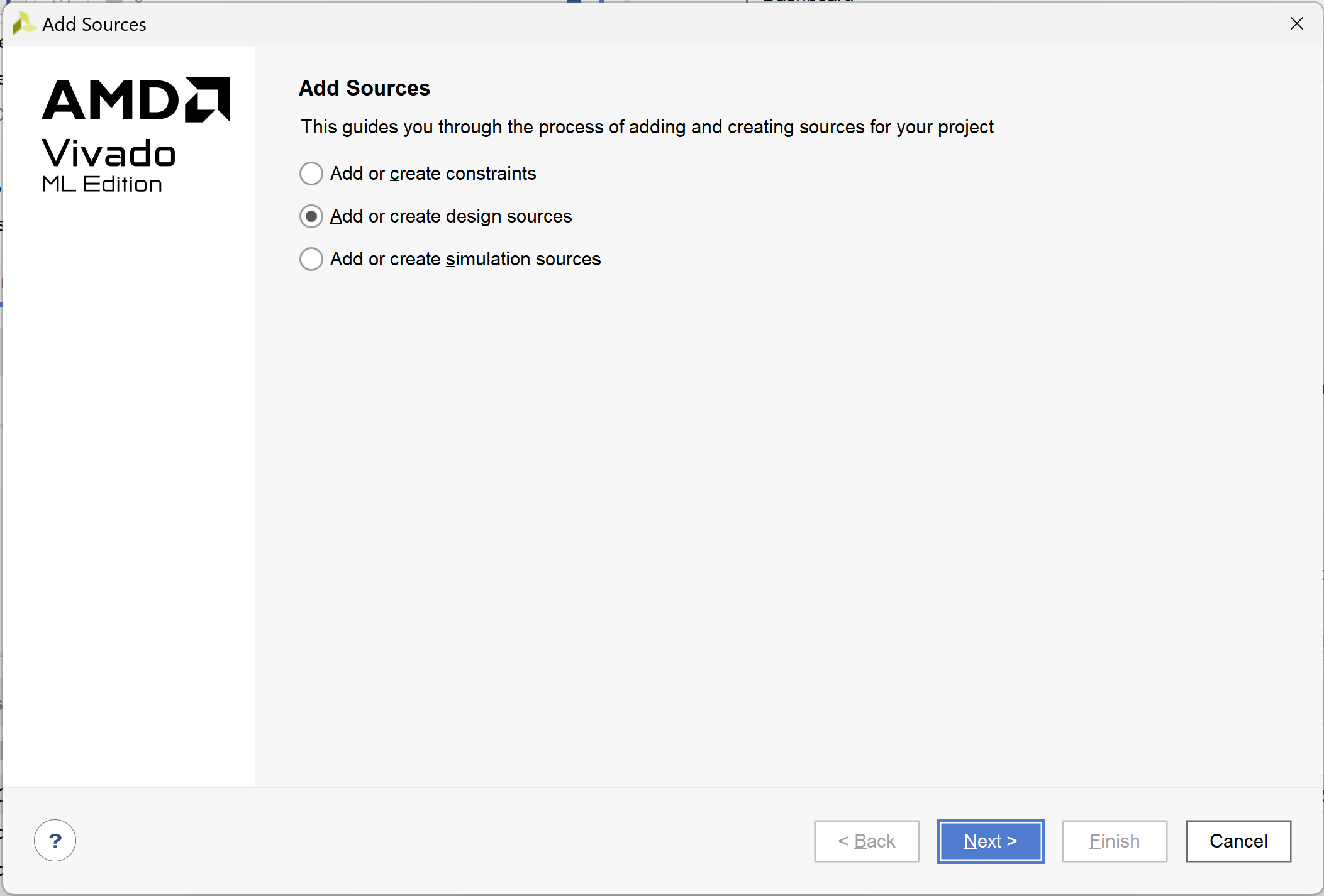
### 2.1 代码编写

打开 Vivado，你将进入下图所示的界面：

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/vivado_base.png)

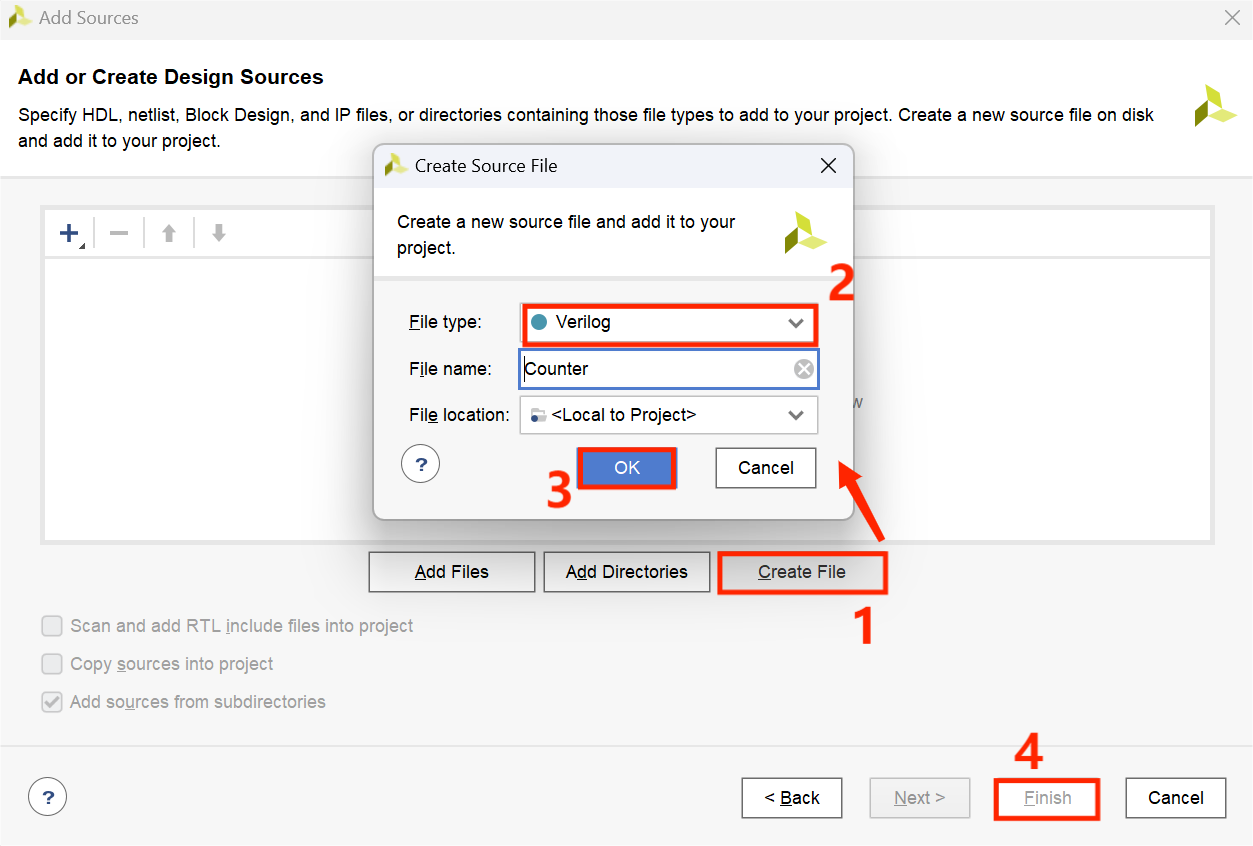
项目界面主要包含 4 大区域。其中 Project Manager 为工程管理窗口，可以完成添加代码、仿真、综合、烧写 FPGA 等一系列操作；Sources 窗口显示代码层级列表，分为设计文件、约束文件和仿真文件三组；Project Summary 窗口显示工程的各种基本信息；Information 窗口显示各种项目的信息参数。

目前我们的项目内还空空如也。因此，我们首先需要创建一个设计文件。在 Project Manager 窗口中点击 Add Sources 可以打开下图所示的窗口。可供添加的文件类型包括：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 sources”。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/add_design.png)

约束文件用于明确模块的输入输出端口与开发板上物理端口的对应关系，设计文件是我们电路模块的具体设计，仿真文件则用于对待测试模块进行仿真测试。

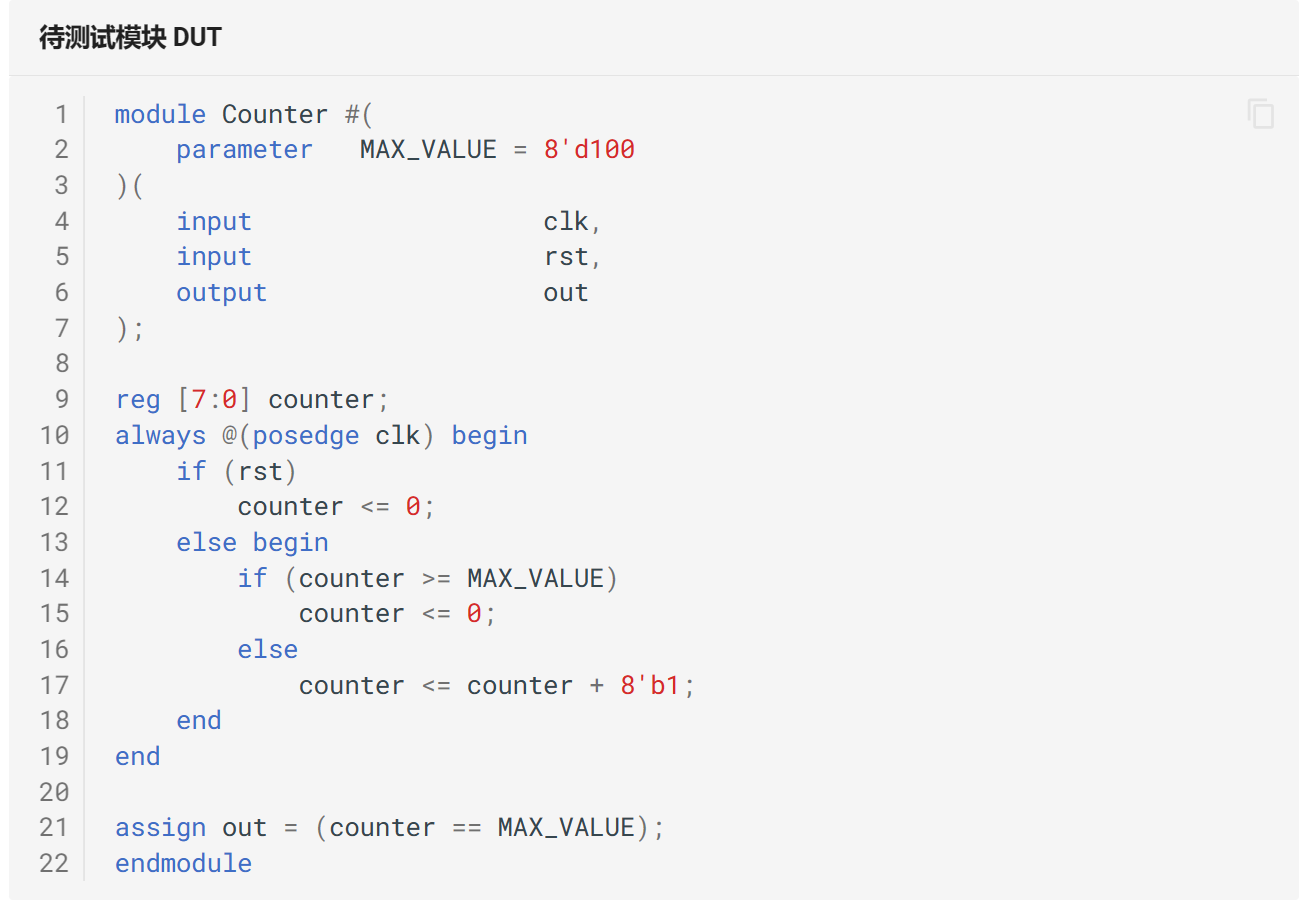
单击 Next，再单击 Create File，我们便可以创建一个全新的设计文件。这里可以设置文件类型与文件名，其中文件类型包括：Verilog、Verilog Header、SystemVerilog、VHDL 和 Memory File。在这里以及后续的实验中，我们都选择 Verilog 作为设计语言。文件名可以简单命名为 "Counter" 或其他你喜欢的名字。完成后单击 OK，再单击 Finish 关闭界面。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/add_design_2.png)

最后，在弹出的 Design Module 窗口中单击 OK，再单击 YES 即可完成文件创建。

在添加文件时，点击 Create File 旁边的 Add Files，就可以把其他位置的设计文件包含入项目了。需要注意的是，这里并不能将对应位置的设计文件真正复制入项目文件夹，而只是进行了目录添加。也就是说，在其他项目也包含了这个设计文件的情况下，如果在本项目中修改这个文件，那么其他项目中的这个文件也会被修改。因此，一种比较好的做法是先将这些文件复制到本项目的目录中，再进行文件添加操作。

创建完成后，在 Sources 窗口中我们便可以看到创建好的 Counter.v 文件。双击这个文件就可以在编辑器中编辑它了。在打开的编辑器中，输入我们之前的代码：



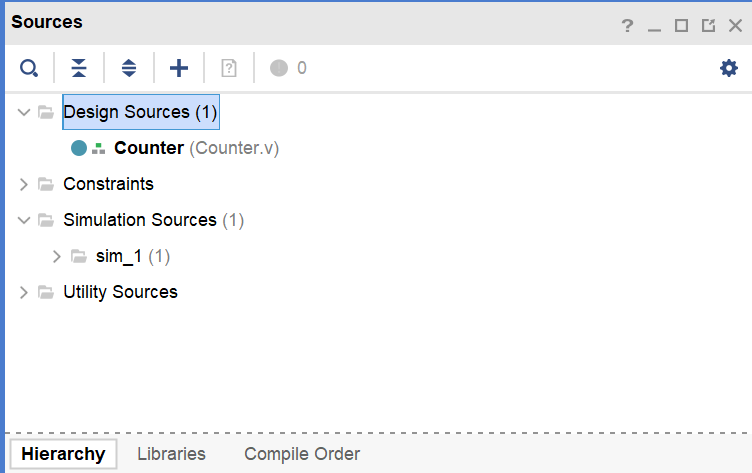
### 2.2 电路检查与RTL

现在我们的项目中已经有了一个设计文件 Counter.v。接下来，我们将进入分析阶段。请确保你的目标设计文件已经被正确设定为 Top 文件（Set as Top）。

由于不同文件的模块存在互相例化的情况，因此 Vivado 规定：只对手动规定的 Top 文件执行用户操作，这样可以保证我们能够一直对想要的模块进行分析（画电路图、运行检查器、仿真、综合、实现等）。所以，在进入分析阶段前都要记得把待分析模块设为顶层模块。

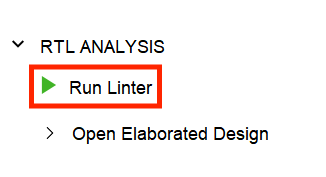
你可以在 Sources 窗口中右键某一模块，在弹出的窗口中选择 Set as Top 将其设定为 Top 模块。Vivado 会自动将项目创建的第一个文件设定为 Top 文件。如果该文件出现语法错误或不可用，则 Top 文件会空缺。

正常情况下，此时你的 Sources 窗口应当是如下的结构：

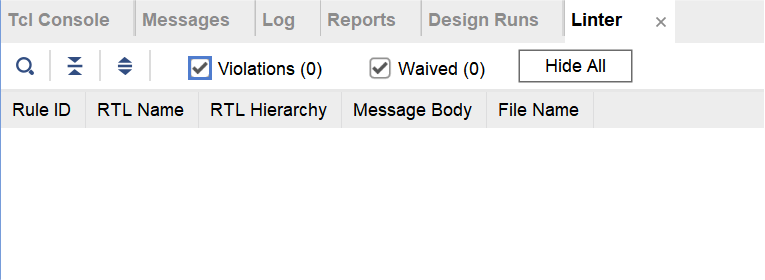
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/Settop.png)

Counter 模块的名称被加粗，代表其已经被设定为 Top 文件。

如果你使用的是 2023.1 版本的 Vivado，在左侧 Project Manager 窗口中可以看到 Run Linter 按钮。这是 2023.1 版本 Vivado 新增的代码检查器，可以检查出大部分潜在的问题。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/linter_2.png)

点击 Run Linter 后会在界面下方打开一个 Linter 窗口。如果这里没有弹出 Warnings 以及 Critical Warnings 则表明目前的设计没有明显的问题。

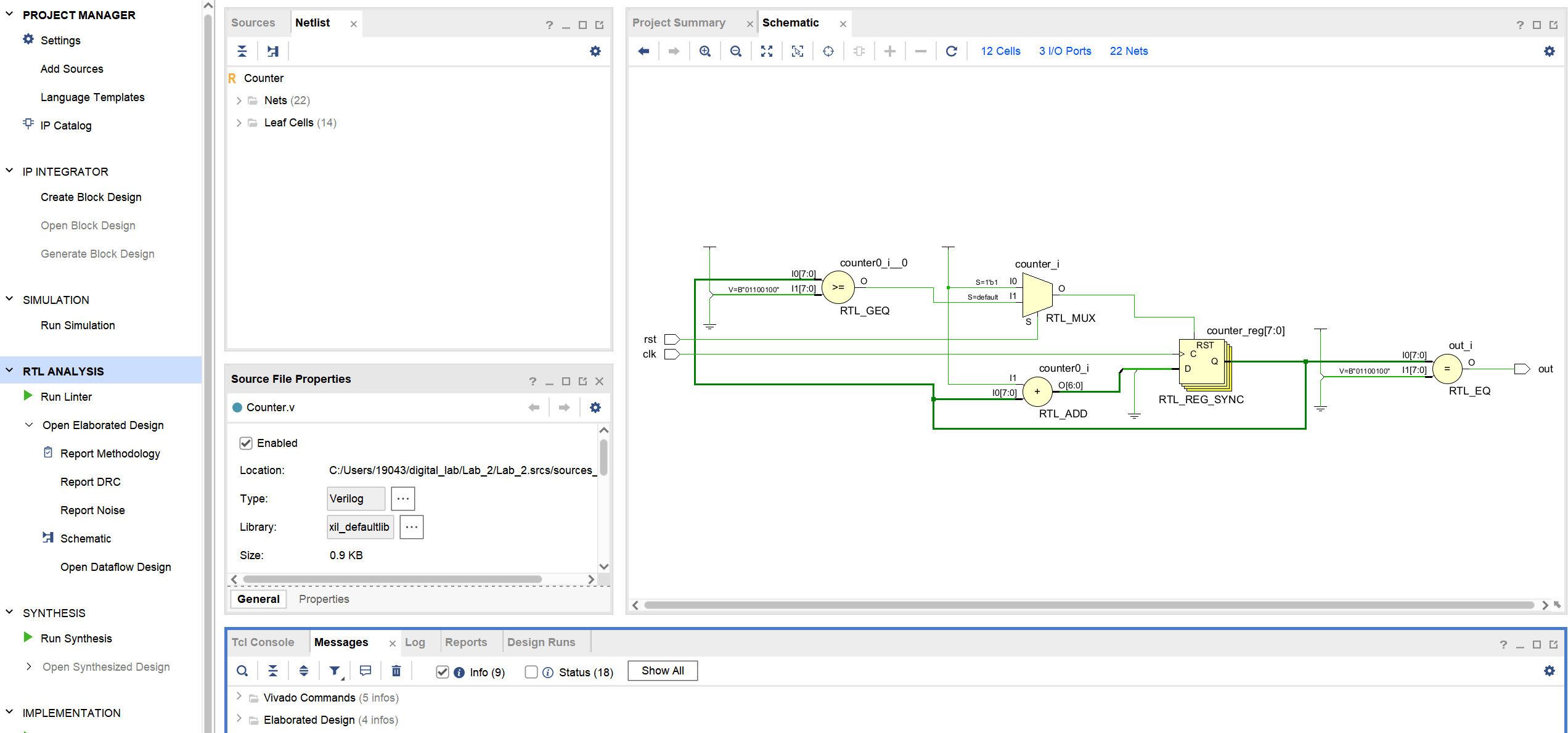
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/linter.png)

当使用一个比较复杂的设计时，Linter 很有可能报出以下两个警告：

* 提示信号没有被读：表示这个信号的某几位虽然接进了某个模块，但在这个模块中并没有使用。
* 提示信号没有被用：表示这个信号虽然被声明了，但是并没有使用。或者说信号的某几位接了常值，而这个常值的位宽小于这个信号的位宽。

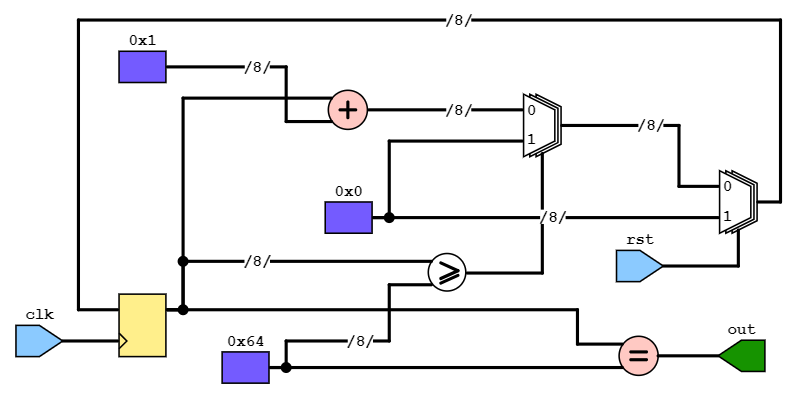
以上两个警告可能是真的代码错误，也很有可能是“杞人忧天”。这时一定要仔细核对出错信息，如果发现所有的信息都是“杞人忧天”，那么就可以进行下一步了。

接下来，点击左侧的 Open Elaborated Design ，画出设计电路图：

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/rtl.png)

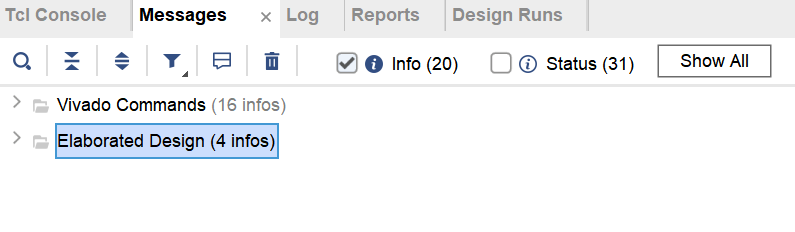
这里可以看到电路初步的样子，主要由一个寄存器、一个加法器、两个比较器和一个选择器组成。

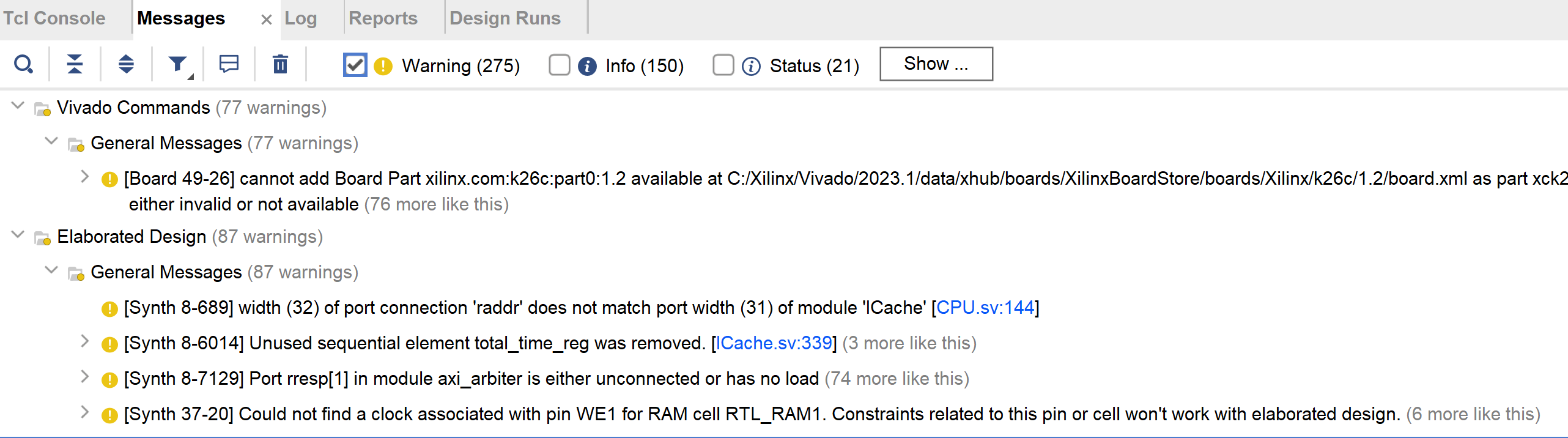
如果你已经配置好了 TerosHDL 插件，则 Counter.v 模块自动生成的电路结构如下图所示：

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/teroshdl.png)

不难发现，RTL 电路与 TerosHDL 生成的电路图有诸多相似之处，而 TerosHDL 中多出的比较器实际上已经被封装在了 RTL 的寄存器模块中。

除了画出电路图外，分析阶段更重要的意义在于打开下方的 Messages 窗口，解决 Elaborated Design 文件夹下所有带具体文件行数的 Warnings。这些具体的警告是千万不可以被忽视的，因为它们是影响硬件设计的重要问题。对于一些很冗长且没有标出具体行数的 Warnings，则可以选择忽视。

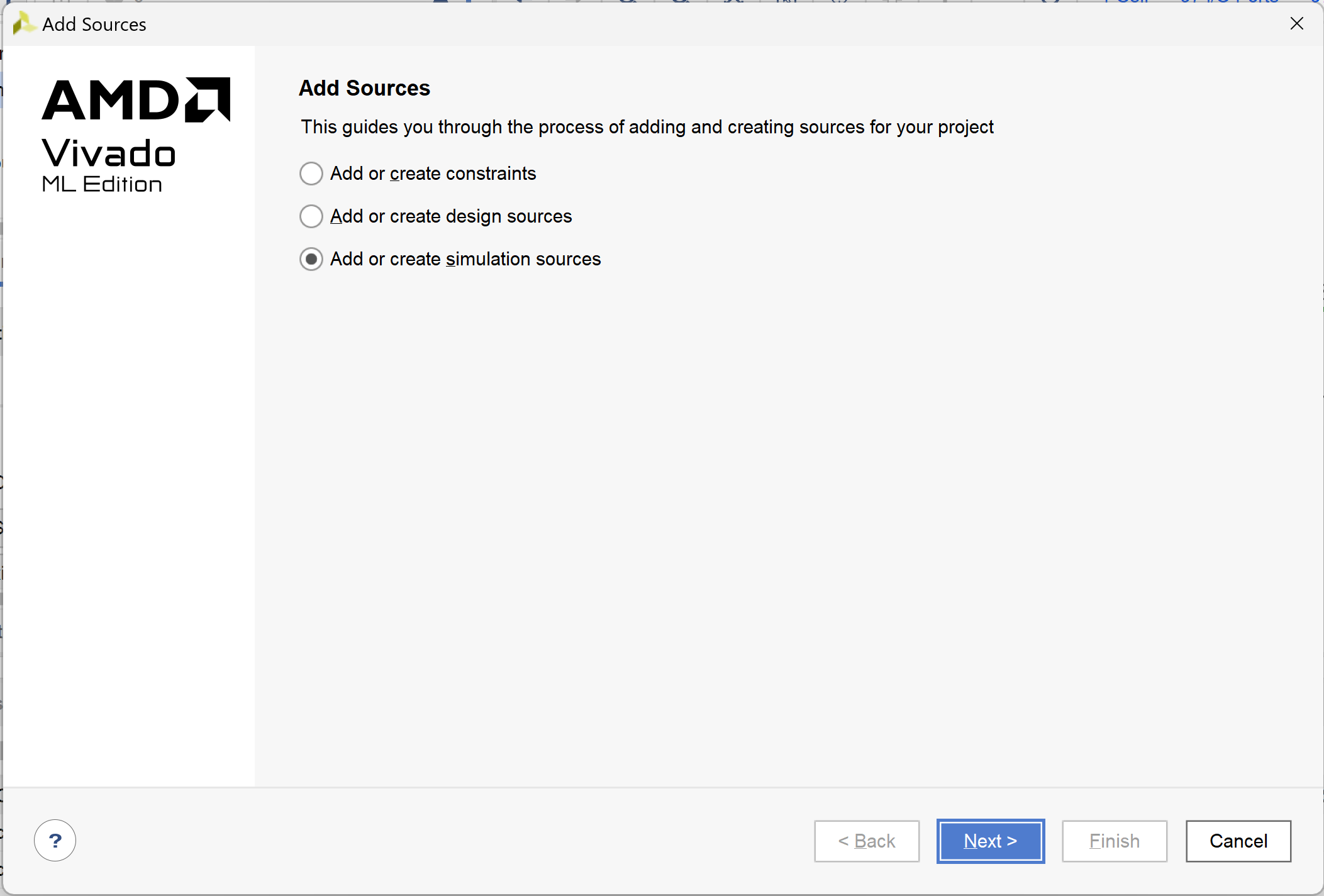
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/elaborate_1.png)

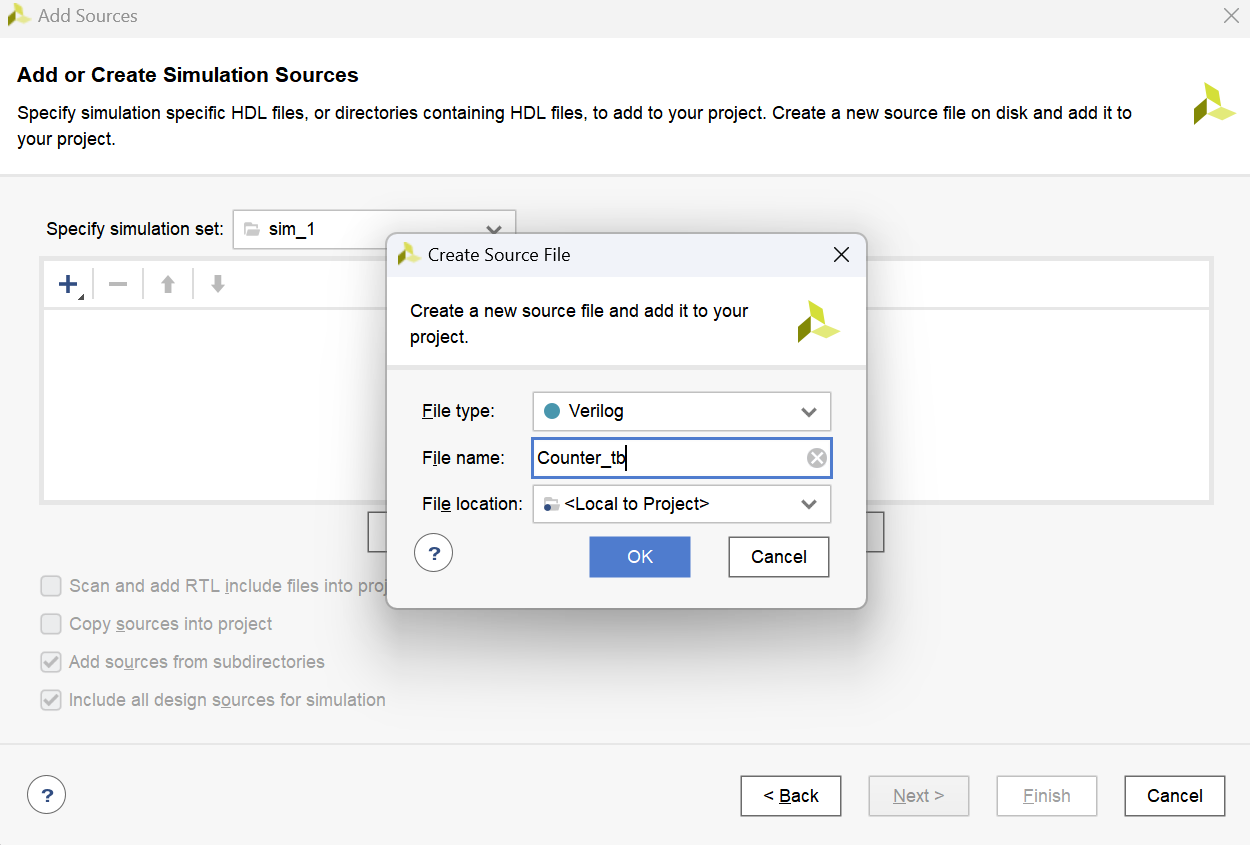
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/elaborate_2.png)

### 2.3 仿真与波形图

现在进入到最为核心的仿真环节。Vivado 允许我们通过编写仿真文件进行测试，观察特定的波形图以了解电路的工作情况。

首先，我们需要创建仿真资源文件。习惯上，我们在需要测试的模块名后加上 \_tb 后缀来表示这是模块的 testbench 仿真文件。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/add_sim.png)

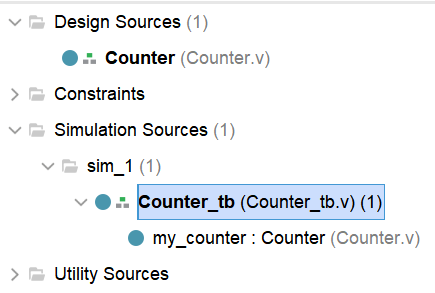
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/add_tb.png)

现在你可以参考我们之前的教程编写仿真文件了。我们给出的参考如下：



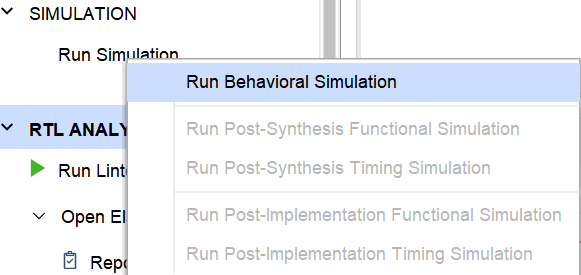
按照之前的教程，一个仿真文件应当包括：激励（Stimulus Block）、输出校验（Output Checker）和被测模块（Design Under Test，DUT）。在这里，输入激励对应着第 5-11 行的 initial 和 always 语句。我们通过这些语句确定了 clk 和 rst 信号的变化逻辑。在 initial 块中，我们首先令 clk 为 0，令 rst 为 1，这代表初始的复位操作。在测试开始前进行复位是保证测试正确性的重要步骤。经过 50 个时间单位后，我们令 rst 为 0。之后就是正常的测试流程了。第 11 行的 always 语句产生了一个周期为 20 个时间单位的时钟信号。接下来的 13-17 行我们例化了待测试模块 Counter，并用变量 out 获取该模块的输出。

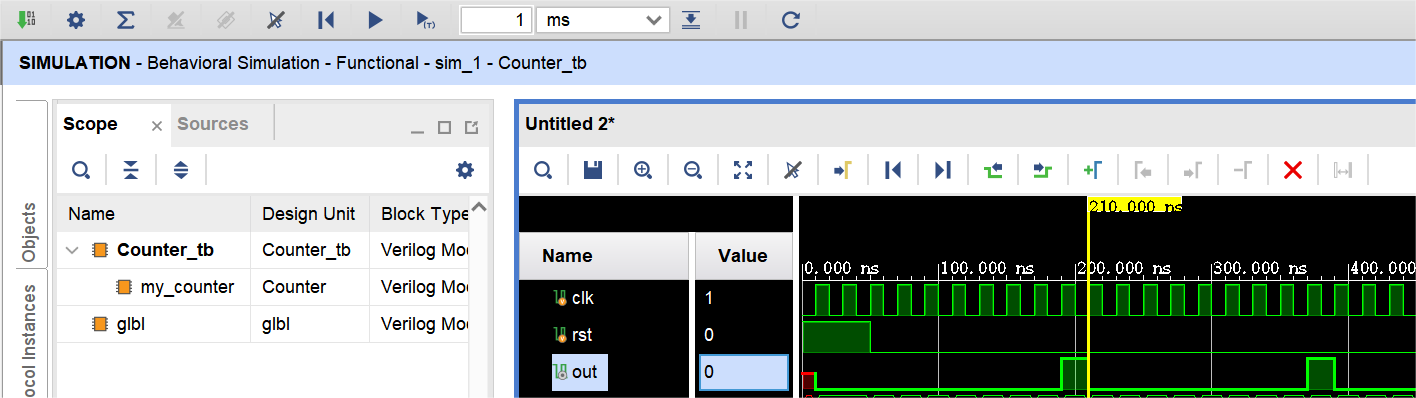
编写完仿真文件后，不要忘记在 Sources 窗口中 Simulation Sources 文件夹下将我们的仿真文件设为 Top。理论上，现在的文件结构应当如下图所示：

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/tb_struc.png)

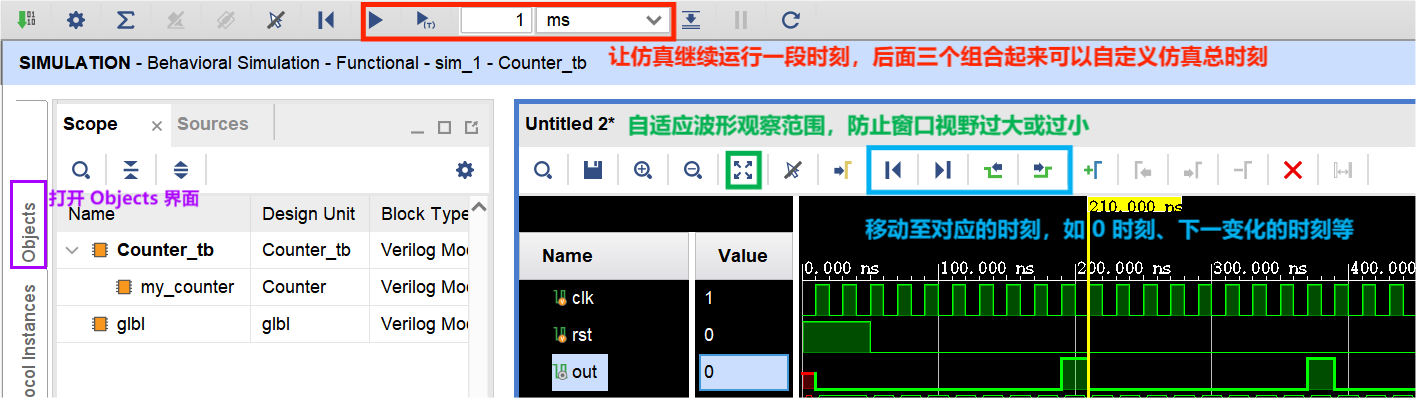
其中仿真文件 Counter\_tb.v 中例化了 Counter.v 模块，因此在文件目录中表现为包含的形式。

接下来，点击左侧 SIMULATION 栏下的 Run Simulation 即可开始仿真。你将看到下图所示的窗口。

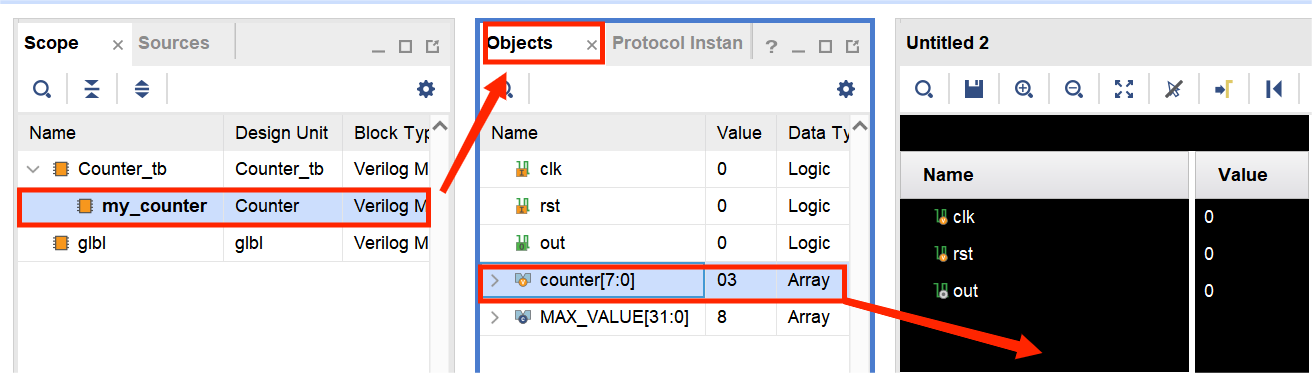
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/behav_sim.png)

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/sim_window.png)

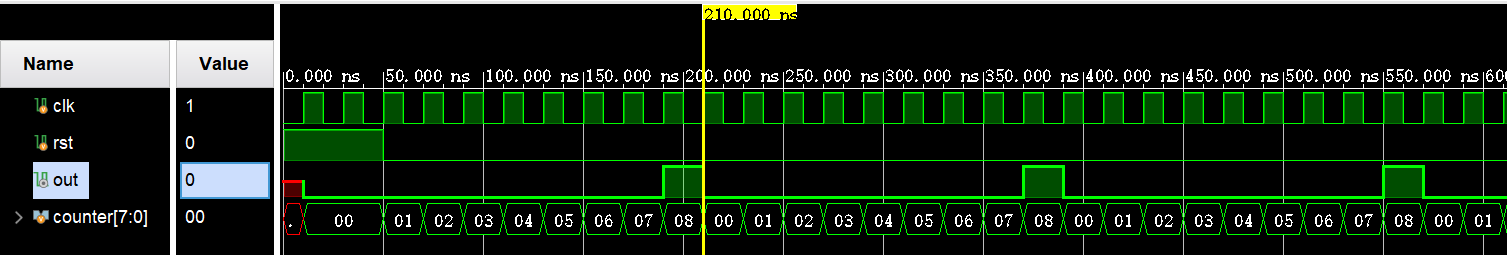
我们来简要介绍一下上图中的基本元素。左侧为模块与元素窗口，用于管理项目中不同层次模块内的信号；右侧为仿真波形窗口，用于显示各个时刻下不同信号的值。下图标注了一些常用的快捷按钮：

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/sim_window_1.png)

你可以点击左侧 Scope 选项卡中的模块名选中需要查看的模块，再在中间的 Objects 选项卡中选择模块内部信号，将其拖拽到波形图的 Name 列中，随后重新运行仿真以查看内部特定信号的波形。

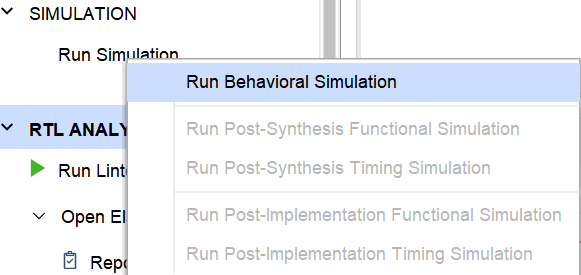
[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/add_signals.png)

例如，我们将 Counter 模块中的 counter 变量拖动到波形图中，重新运行仿真后得到的波形图如下图所示：

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/result.png)

从图中可以看到，每经过 9 个时钟周期 out 就会发出一次信号，这是符合我们的设计预期的。

在点击 Run Simulation 的时候，事实上运行的是 behaviour simulation，我们称之为行为仿真。

[](https://soc.ustc.edu.cn/Digital/lab2/figs/simulation/behav_sim.png)

行为仿真是一种在逻辑上近乎正确，而在物理上可能出现问题的仿真，它假定设计的最长组合逻辑通路的延时一定小于时钟周期，因此它并没有考虑到逻辑延迟大于时钟周期的情况。如果想要考虑逻辑延迟，那么就需要等到实现电路后运行时序仿真。由于 Vivado 编译器会将信号重命名或元件重排，因此时序仿真很有可能如同「天书」，我们一般还是以看行为仿真为主要的纠错手段。

# 思考与练习

1. 这是我们在实验教程中展示的计数器代码：



现在，我们需要对其进行一点改进。

1. 现在的计数器复位值是 0，最小值是 0，最大值是 MAX\_VALUE。请修改代码，使得计数器的最小值也可以由模块参数 MIN\_VALUE 指定。
2. 在(1)的基础上，增加一个 1bit 输入信号 enable 用于控制计数器的工作状态。要求：enable 为高电平时，计数器在 MIN\_VALUE 至 MAX\_VALUE 之间正常工作；enable 为低电平时，counter 变量复位并保持在 0。从 enable 变为低电平开始到 counter 复位并保持在 0 的间隔不超过三个时钟周期。此外，rst 信号的优先级应当高于 enable 信号。

你需要结合仿真波形证明自己设计的正确性。