# 实验十：FPGA 运行 LC3 系统

1. 实验目的：

1.1. 掌握配置生成用于 FPGA 的可综合代码的方法

1.2. 掌握 vivado 的安装与使用

1.3. 掌握 vivado 中 LC3 项目的搭建

1.4. 掌握烧录 FPGA 的流程

2. 实验内容：

2.1. 学习 LC3 中可综合代码的生成

2.2. 学习 Vivado 的安装

2.3. 学习 Vivado 项目的新建以及 LC3 代码的导入，以及 IP 核的导入

2.4. 学习使用 Vivado 烧录 LC3 到 FPGA 上的流程

3. 实验步骤：

3.1. 生成可综合的 LC3 代码

综合 (synthesis)就是将 Verilog 描述的 RTL 级的电路模型构造出**门级网表**的过程。综合只是个中间步骤，综合后生成的网表文件，就是由导线相互连接的寄存器传输级功能块 （像是触发器、算术逻辑单元和多路选择器等）组成的，但是并不是所有的硬件设计代码都是可综合的，比如一些用于仿真验证的子集，属于仿真验证语言，只在仿真时候使用，不能被综合成电路，因为没有相应的硬件元件与其对应，比如 Chisel 语法中的 printf，就是不可综合的，比如在 LC3 项目中，使用 DPI-C 实现的 Memory 模块，也是不可综合的。

在 LC3 设计时为什么不直接使用可综合的代码来设计 Memory 呢，这是因为 Memory 需要实现一些初始化逻辑，这部分逻辑使用不可综合的代码来实现更加简单。而当需要将 LC3 移动到 FPGA 上运行时，则需要对 Memory 做一些改动。

通过修改 Top.scala 中的 FPGAPlatform 变量为 true，再执行″ make verilog″，重新生成一遍 verilog 文件，在生成过程中可以观察到终端输出 FPGAPlatform = true 这样的信息，则生成的./build/TopMain.v 就是可综合的 Verilog 文件了。

在 Memory.scala 文件中，当 FPGAPlatform 为 true 时，memory 内部就会由原来的 RAMHelper 替换为 dual_mem，而进一步查看 dual_mem 这个模块的定义就会发现，这是一个空模块，除了定义了一些接口，内部没有任何逻辑。这是因为在将 LC3 烧录到 FPGA 上时，Memory 模块需要使用 FPGA 上的专门的 RAM 资源来实现，这就需要引入第三方的 IP 核，因此 dual_mem 只需要按照 IP 核的接口定义好，内部的具体逻辑封装在 IP 核中。

RAM (Random Access Memory, 随机存取存储器)，可以随时读写，而且速度很快，通常作为操作系统或其他正在运行中的程序的临时数据存储介质，也叫做内存。RAM 工作时可以随时从任何一个指定的地址写入（存入）或读出（取出）信息，RAM 的一个特点就是数据的易失性，即在断电时其中所有的数据都会丢失。

IP 核，全称知识产权核（Intellectual Property Core），是在集成电路的可重用设计方法学中，指某一方提供的、形式为逻辑单元、芯片设计的可重用模组。 IP 核通常已经通过了设计验证，设计人员以 IP 核为基础进行设计，

深圳大学微电子研究所

可以缩短设计所需的周期。可以理解成一个已经封装好了的函数，说明各个参数（接口）的含义与作用，内部具体实现可以不用关心。

### 3.2. Vivado 安装

由于 Vivado 安装需要占用的空间较大，且运行时对电脑性能有一定的需求，因此不建议在虚拟机中安装 Vivado，最好直接在电脑上安装。本实验中使用的 Vivado 版本是 Windows Vivado v2020.3，安装包在实验材料中，名为"Xilinx_Unified_2020.3_0407_2214"，将其解压后，可以看到其中有名为"xsetup.exe"的可执行文件，双击运行，可以看到如图 10.1 所示的对话框，这是表示有更新的版本，但这里直接点击 Continue 按钮跳过，就会看到如图 10.2 所示的界面，点击 Next 按钮。之后会看到让你选择需要安装的软件，如图 10.3 所示，选择 Vivado，然后点击 Next 按钮。然后进入如图 10.4 所示的界面，选择 Vivado HL System Edition 然后点击 Next。进图如图 10.5 所示的界面，保持默认值，直接点击 Next 按钮，之后进入图 10.6 所示的界面，把三个 I Agree 勾选后，点击 Next 按钮。之后会进入到如图 10.7 所示的界面，在红框中选择想要将 Vivado 安装在哪个路径下，选择好之后点击 Next 按钮。然后可以看到图 10.8 的界面，这是之前配置的一些总览，点击 Install 按钮，进入如图 10.9 所示的界面，这时 Vivado 就开始安装了，安装过程可能会比较久，请耐心等待，安装完后，在桌面上可以看到如图 10.10 所示的图标，双击打开后看到如图 10.11 所示的窗口，则表示 Vivado 安装成功。

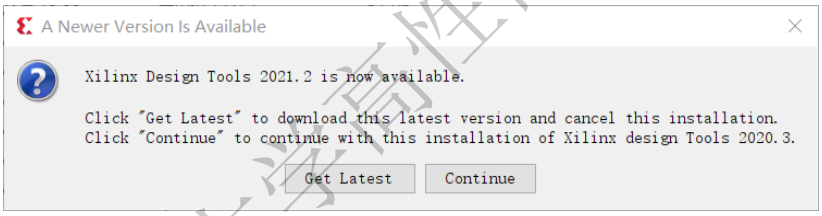



图 10.1

深圳大学高性能微处理器

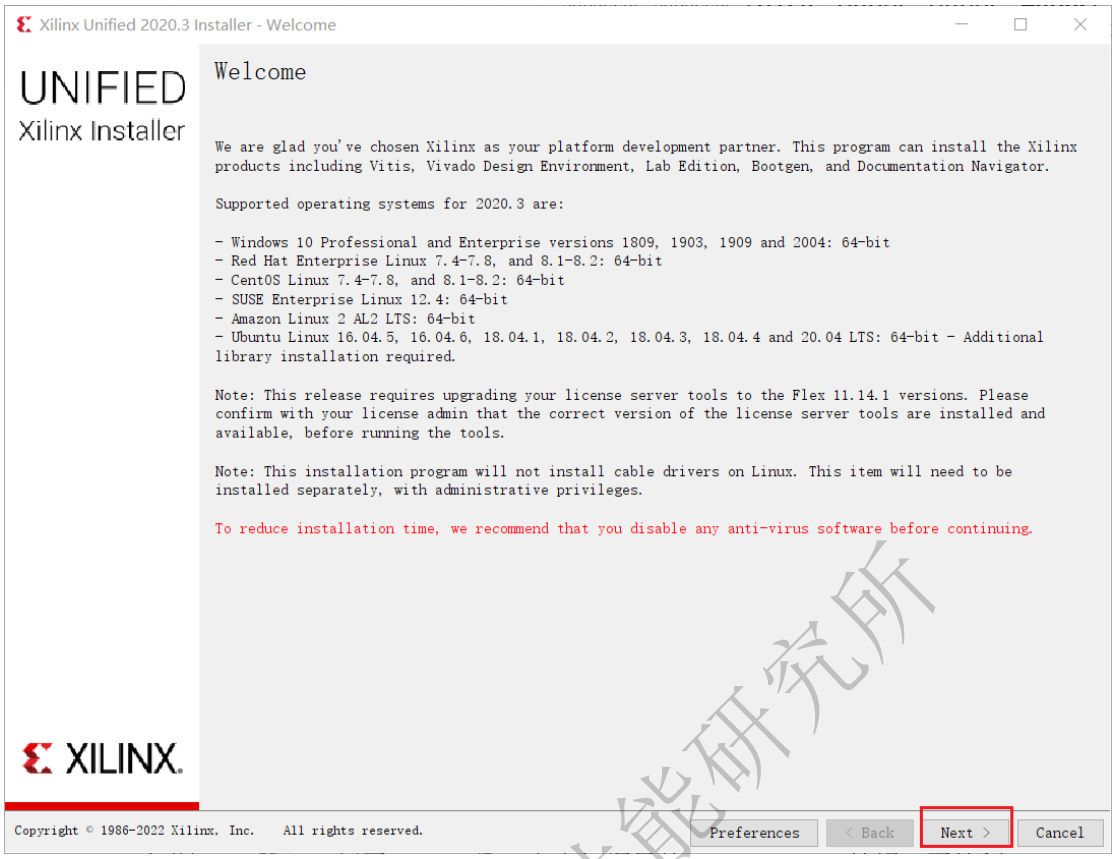



图 10.2

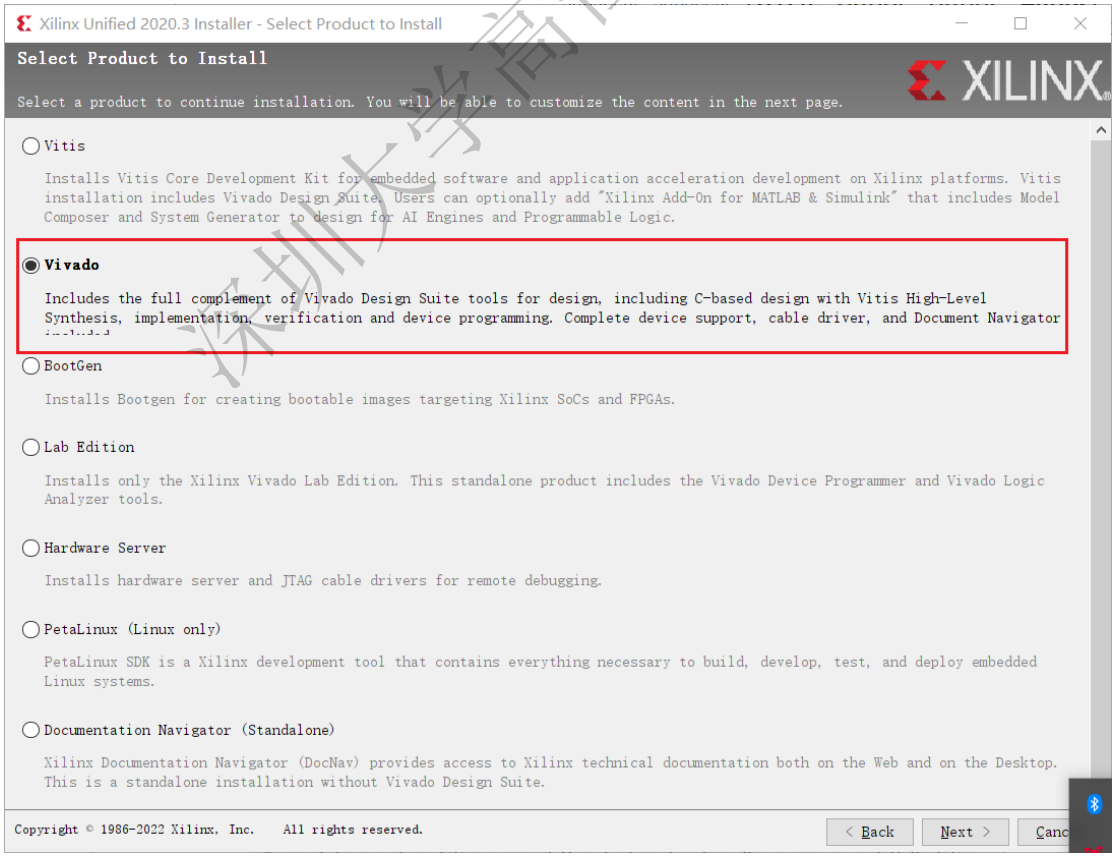



图 10.3

深圳大学高性能研究所


图 10.4

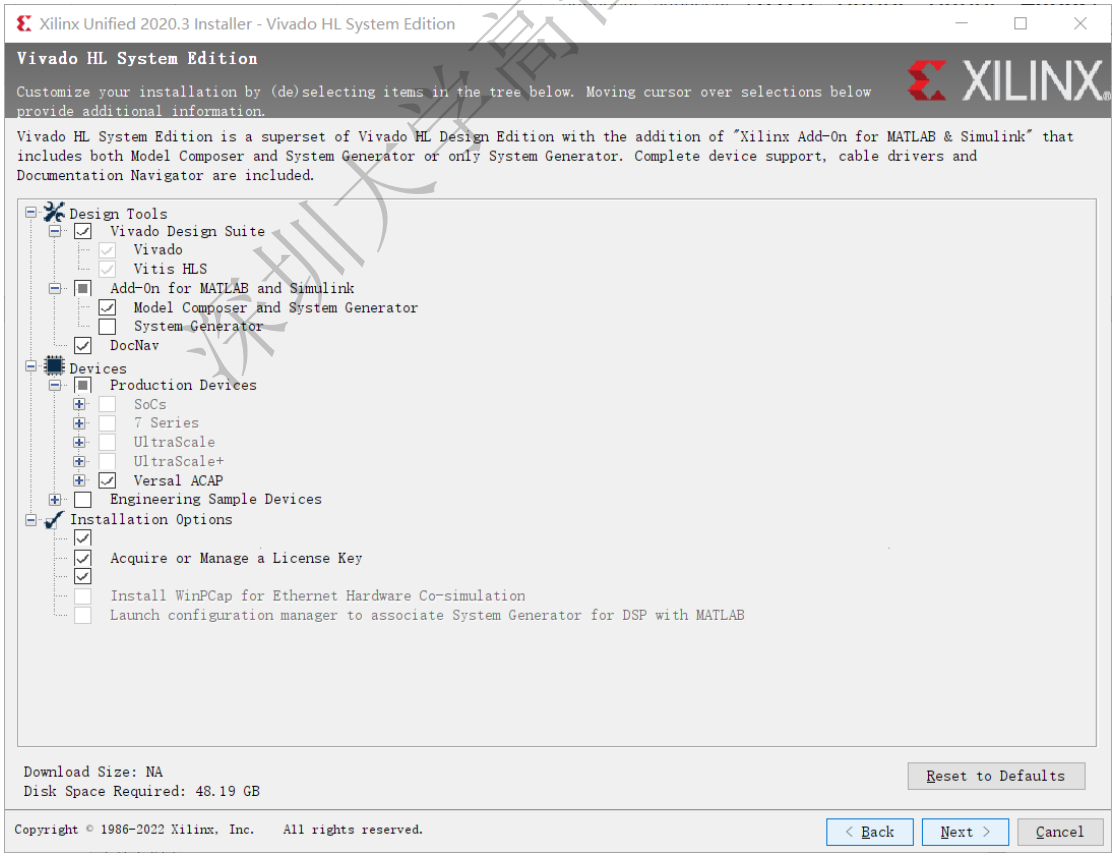



图 10.5

深圳大学高性能研究所

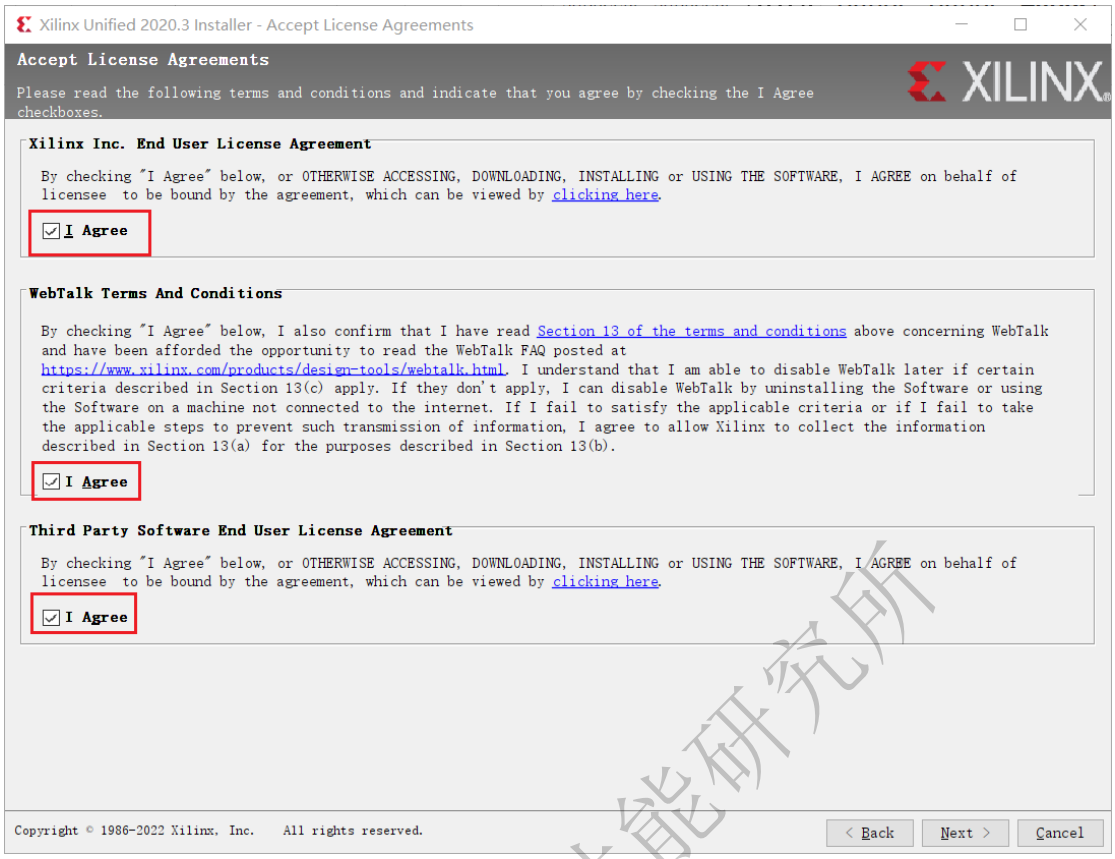



图 10.6

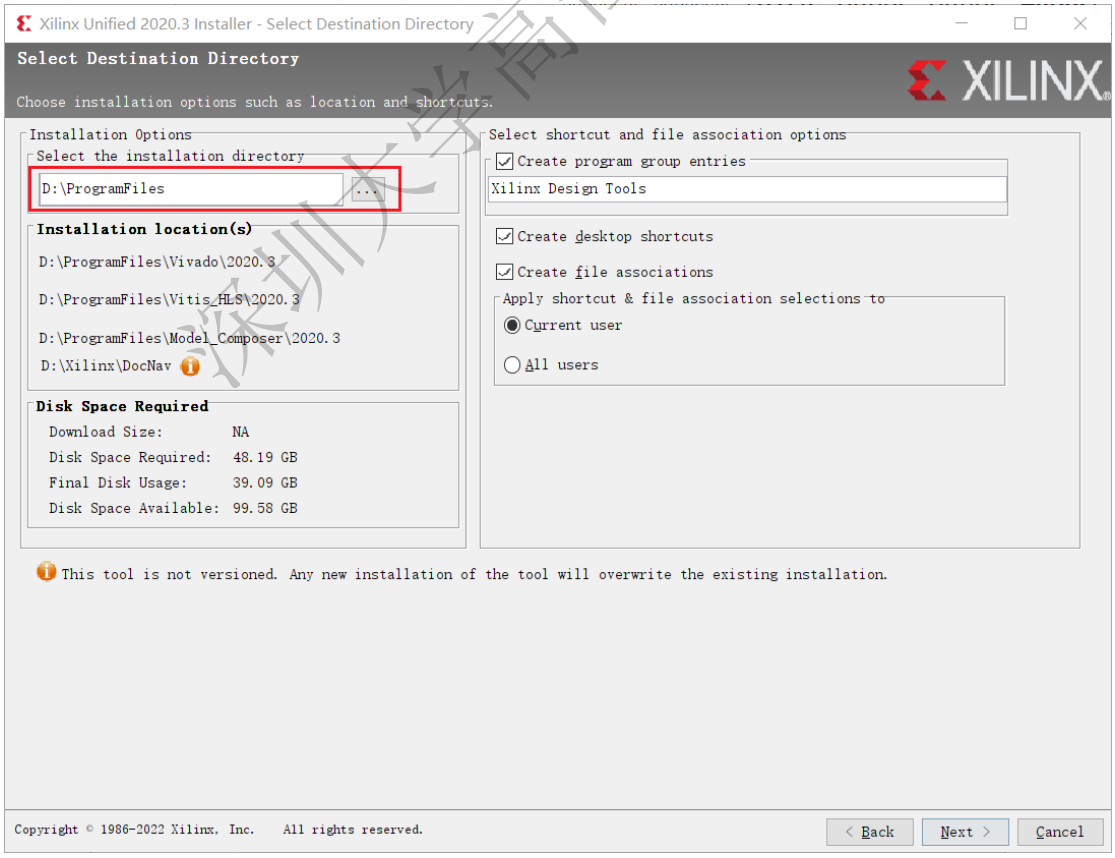



图 10.7

深圳大学高性能研究所

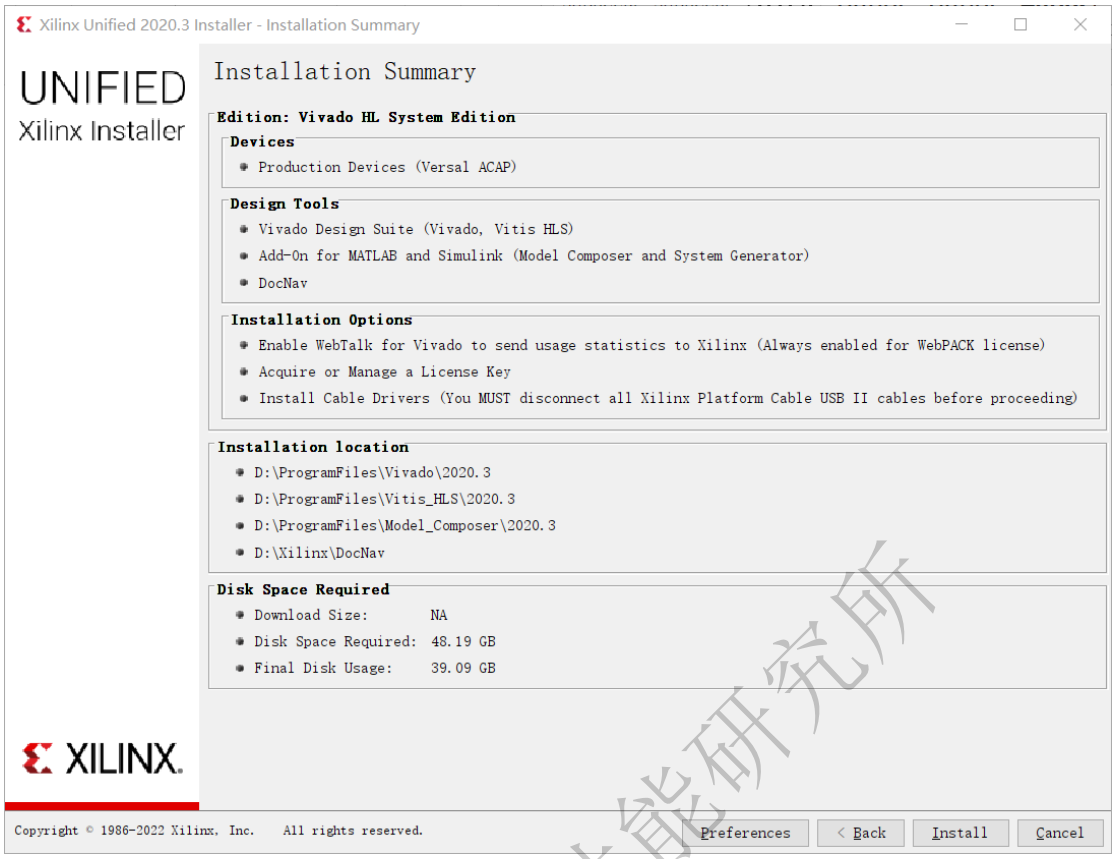



图 10.8


图 10.9

深圳大学高性能研究所


图 10.10

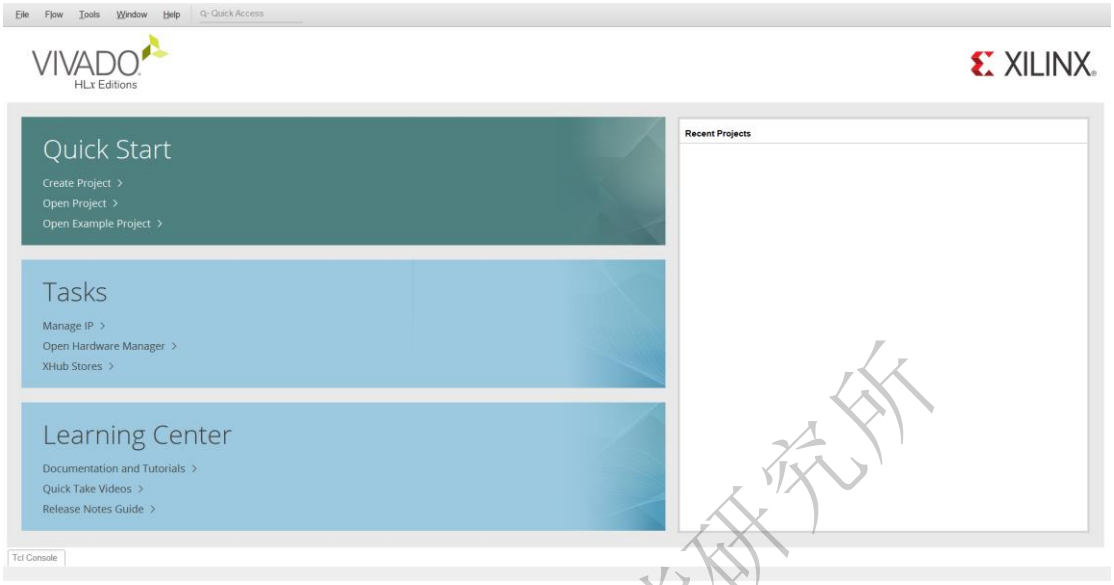



图 10.11

在 Vivado 安装完成后，还没有结束，因为 Vivado 是商业软件，因此还需要导入官方的授权 License 文件，才能够使用 Vivado 的完整功能，首先找到实验材料中的 Vivado2018.3_licence.lic 文件，如图 10.12 所示，在 Vivado 主窗口的菜单栏中选择 Help->Manage License，就会打开一个如图 10.13 所示的窗口。先在左侧选择 Load License 选项，之后在右侧点击 Copy License 按钮，在弹出的窗口中找到实验材料中的 Vivado2018.3_licence.lic 导入，Vivado 提示导入成功后，Vivado 就可以正常使用了。

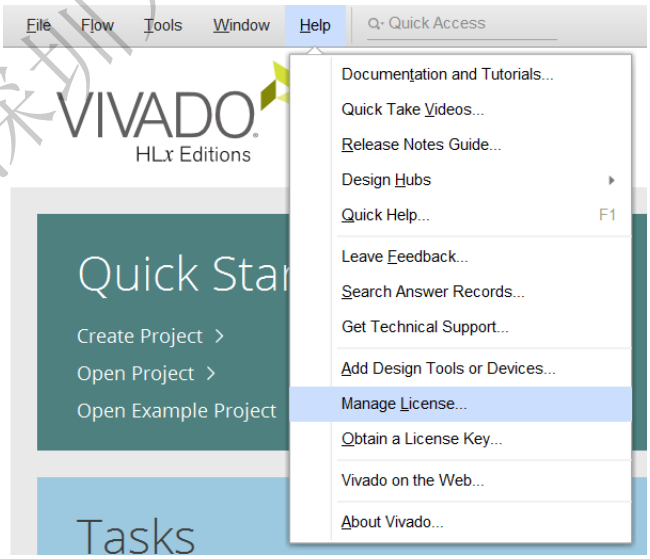



图 10.12

深圳大学微纳电子能研究所

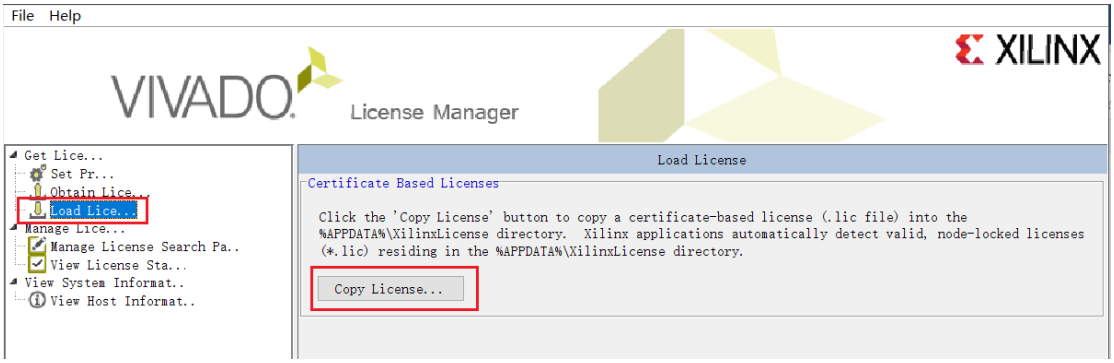



图 10.13

**实验十-任务一： 按照实验指导，成功安装 Vivado，并导入 license 文件**

3.3. 构建 LC3 的 Vivado 项目

如图 10.14 所示，打开 Vivado 后，点击红框里的 Create Project 按钮，开始创建一个新的 Vivado 工程。然后如图 10.15 所示，点击 Next 按钮。

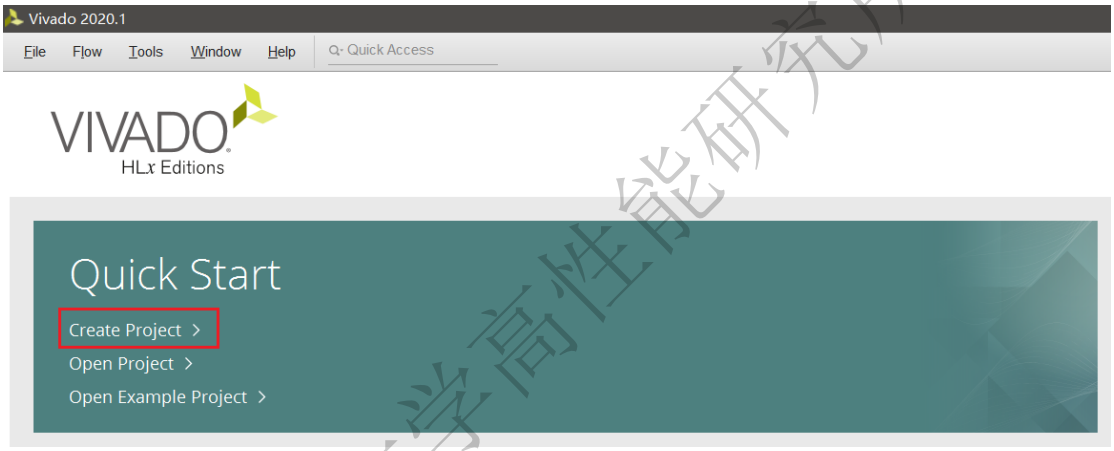



图 10.14 Vivado 创建新项目

深圳大学高性能研究院

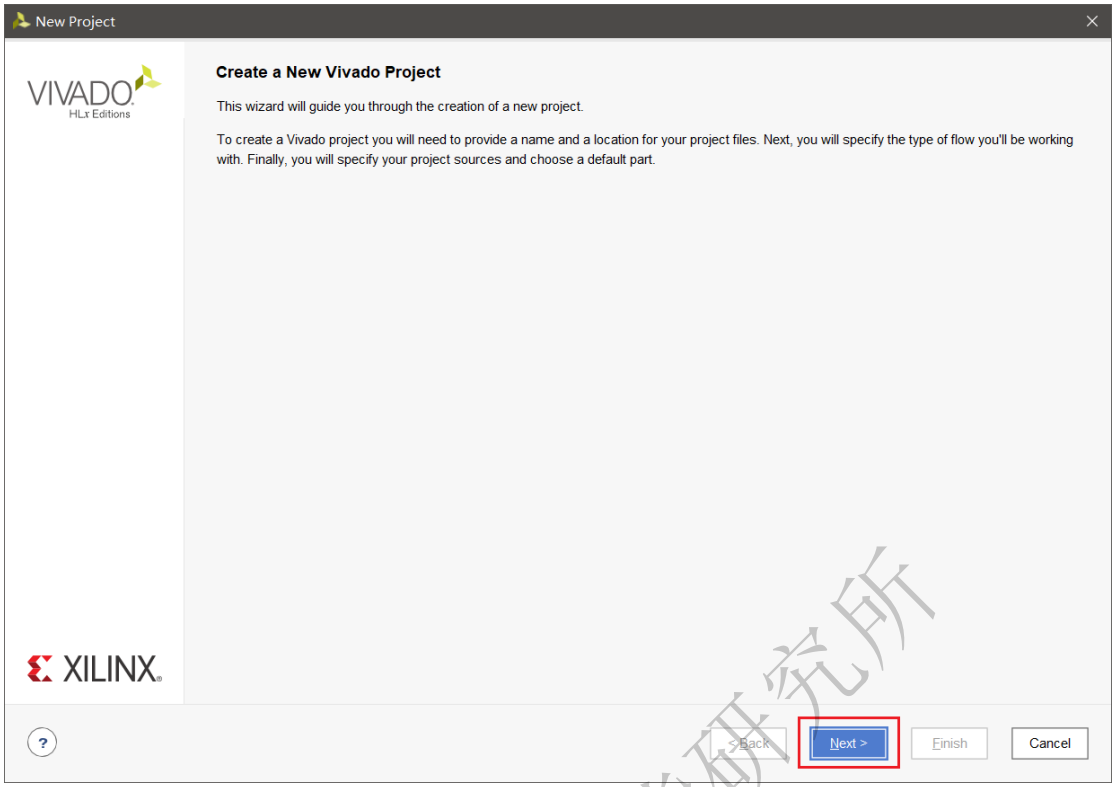



图 10.15

然后起一个项目名，比如图 10.16 中给项目命名为 lc3_fpg，接下来选择 RTL Project，如图 10.17 所示。

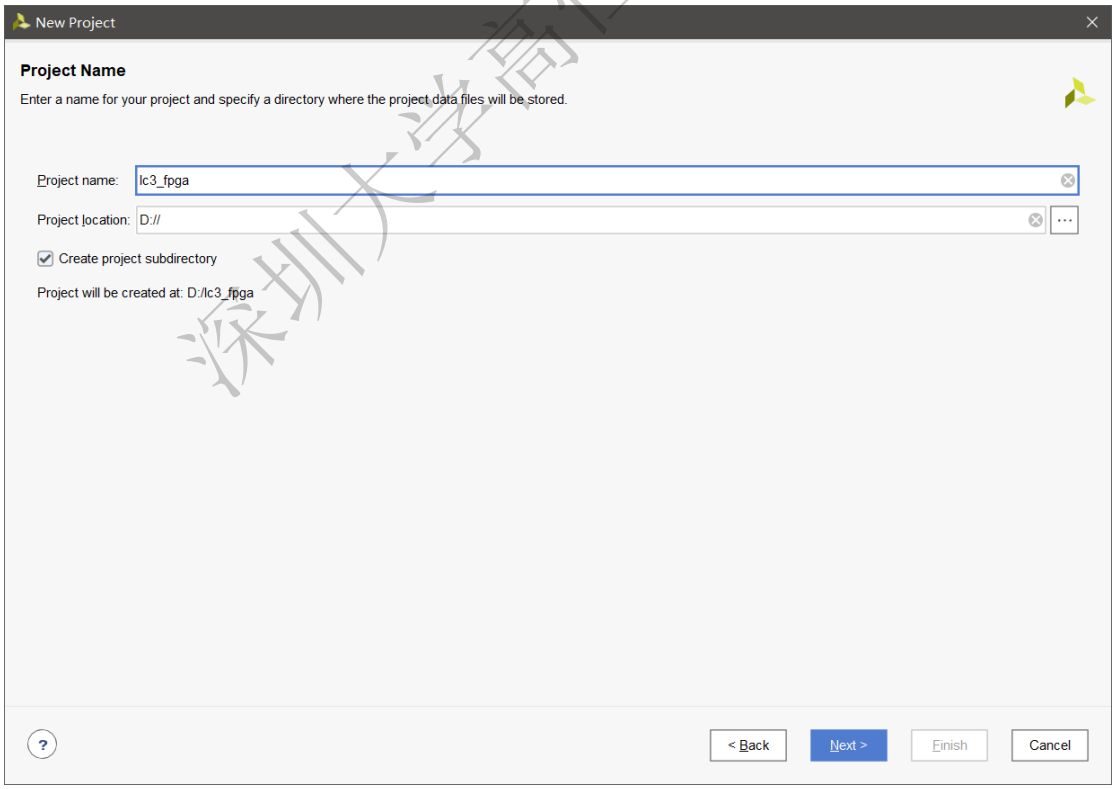



图 10.16

深圳大学高性能研究所

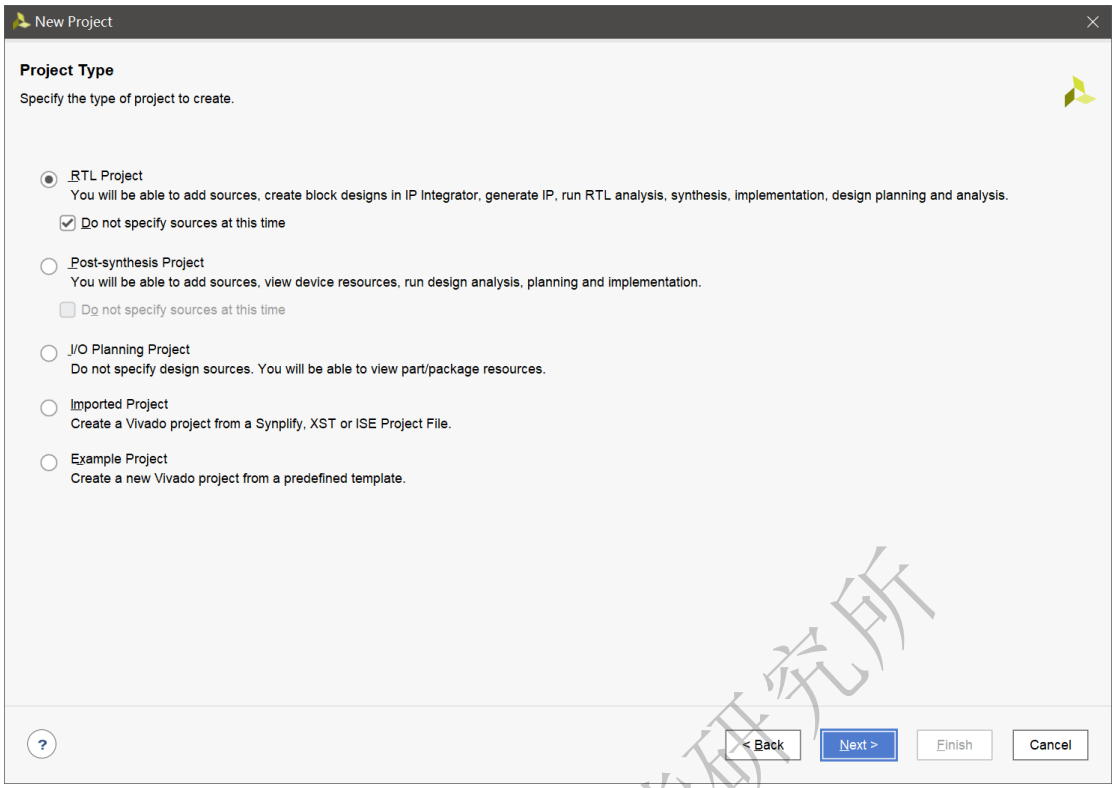



图 10.17

在如图 10.18 的窗口中，需要选择正确的 FPGA 芯片型号，实验中使用的芯片型号是 xc7a35tfgg484-2，如图 10.18 所示，可以通过选择红框中的选项来筛选掉多余的芯片型号。然后点击 Next，见到如图 10.19 所示的界面，点击 Finish 按钮即可，之后会来到 Vivado 的主要界面，如图 10.20 所示。

深圳大学高性能研究所

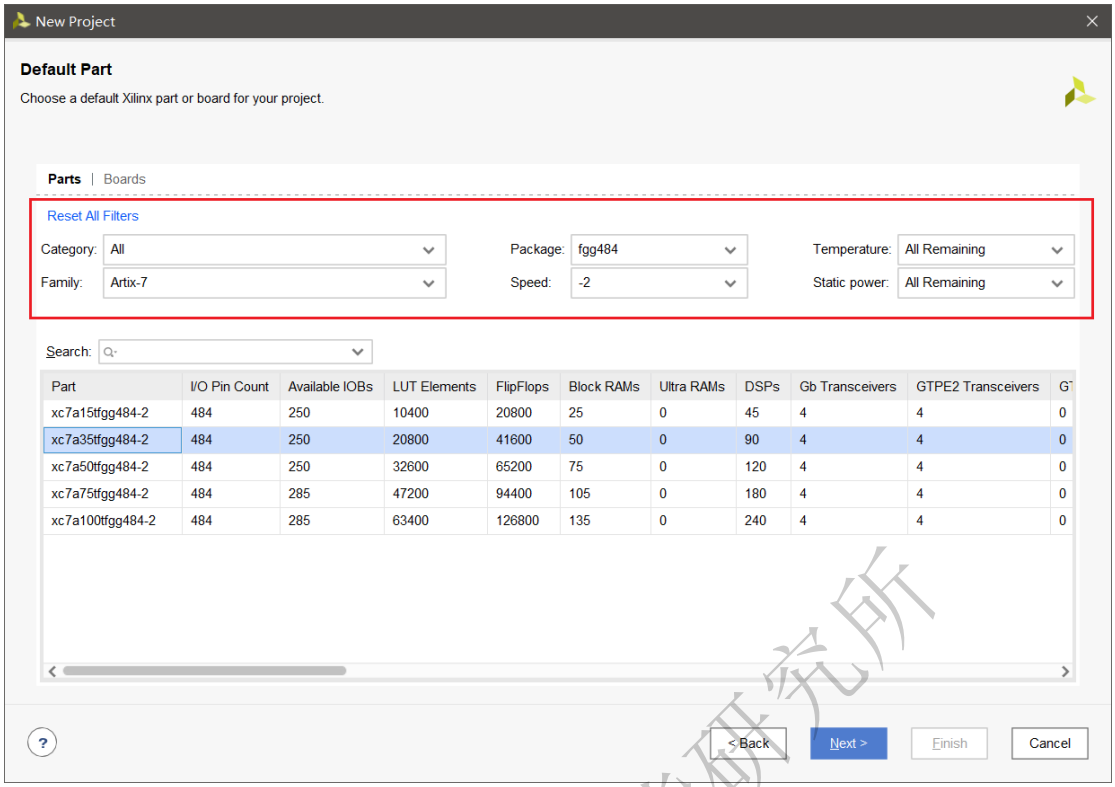



图 10.18

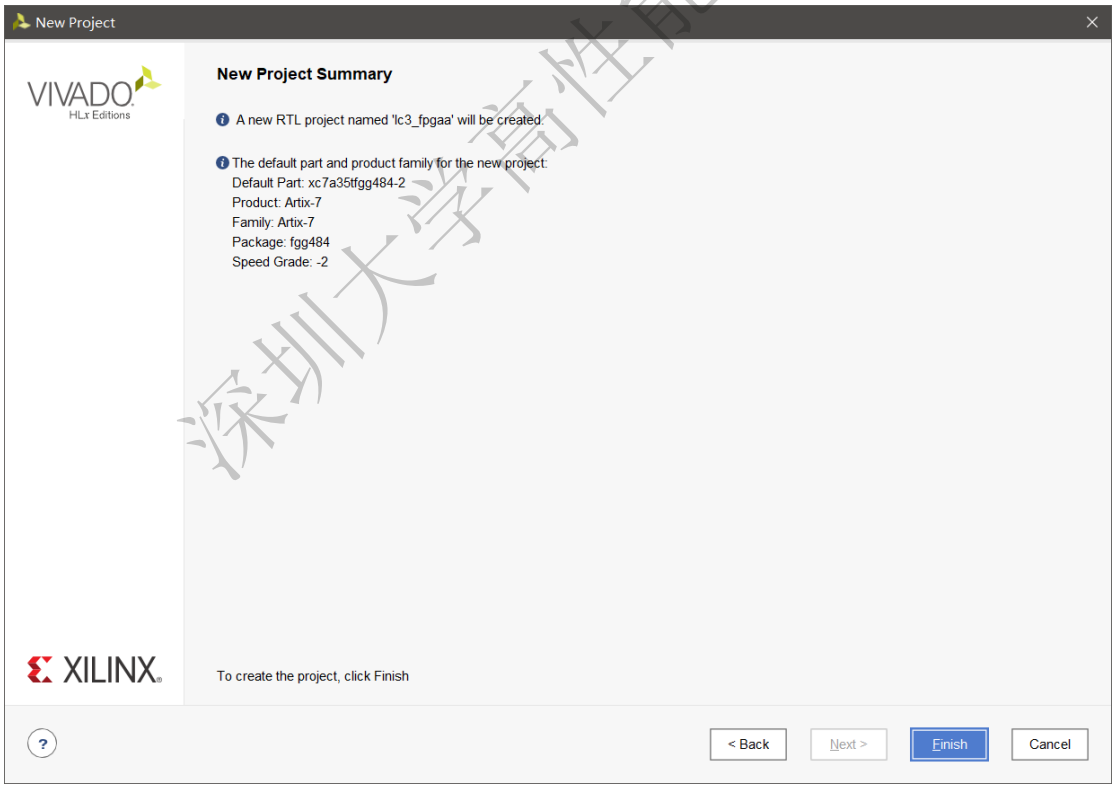



图 10.19

深圳大学高性能研究所

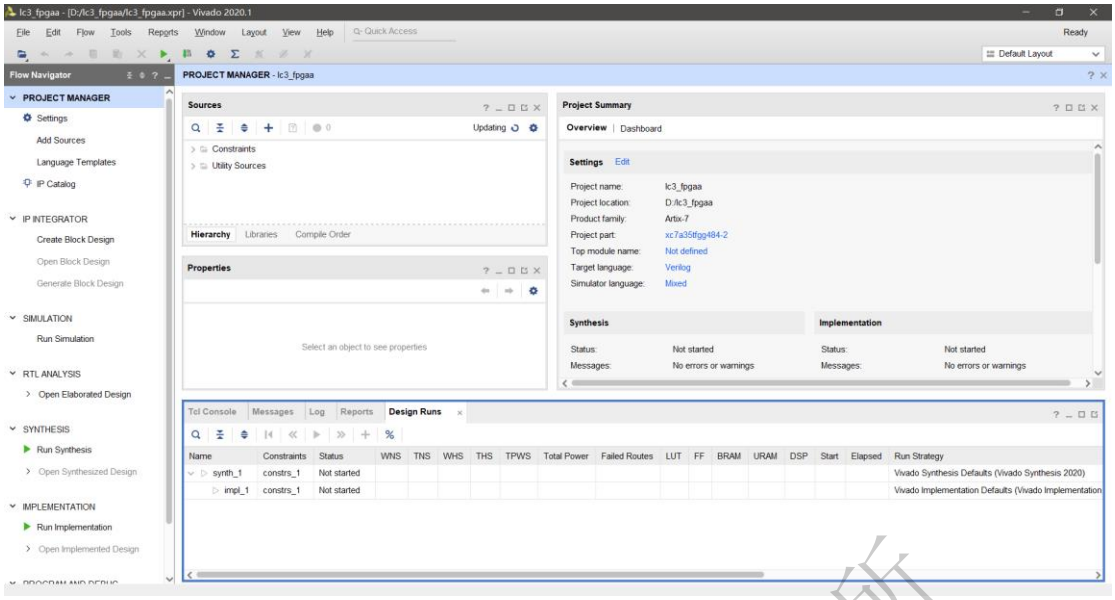



图 10.20

建立完项目后，记得将 TopMain.v 文件复制到刚建好的 Vivado 项目路径下，然后在项目中点击 Add Sources 按钮，将 TopMain.v 文件添加到项目中，如图 10.21-23 所示

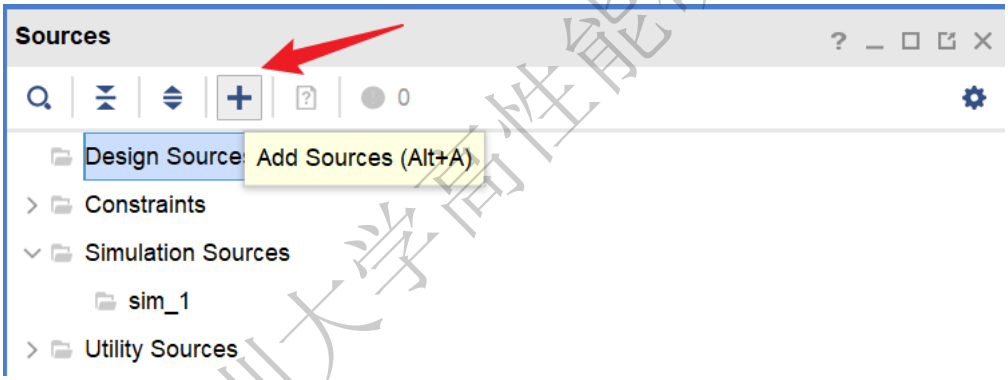



图 10.21

深圳大学高性能研究所

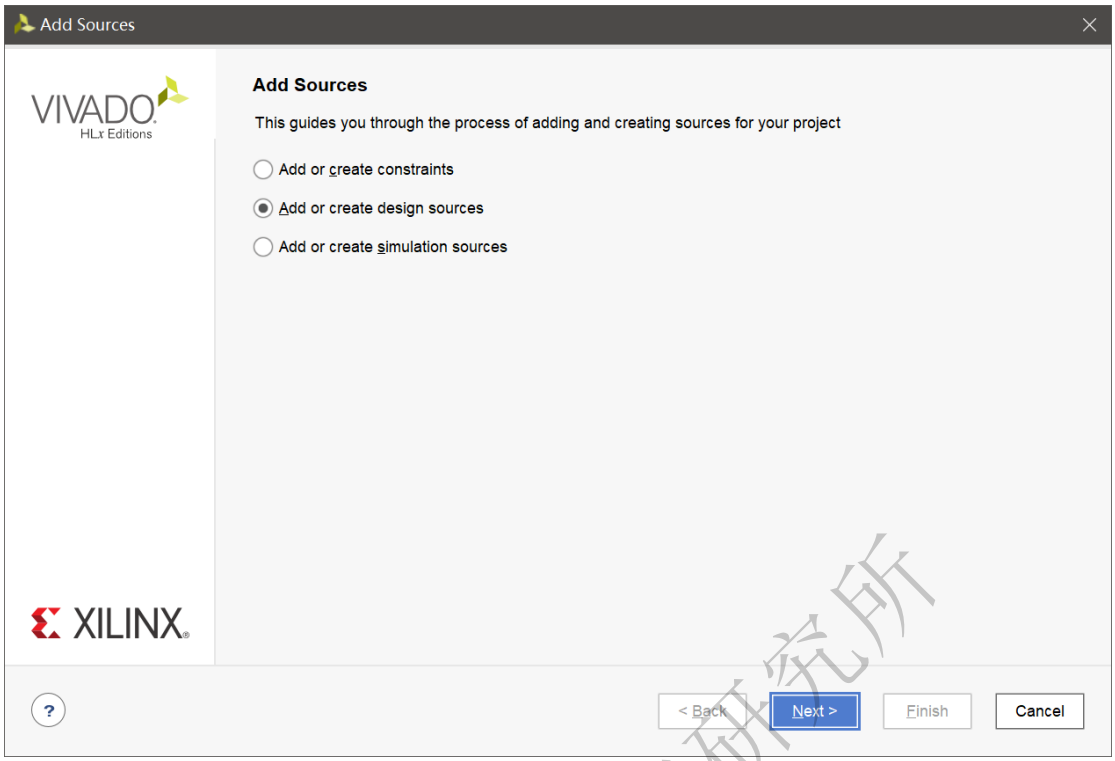



图 10.22

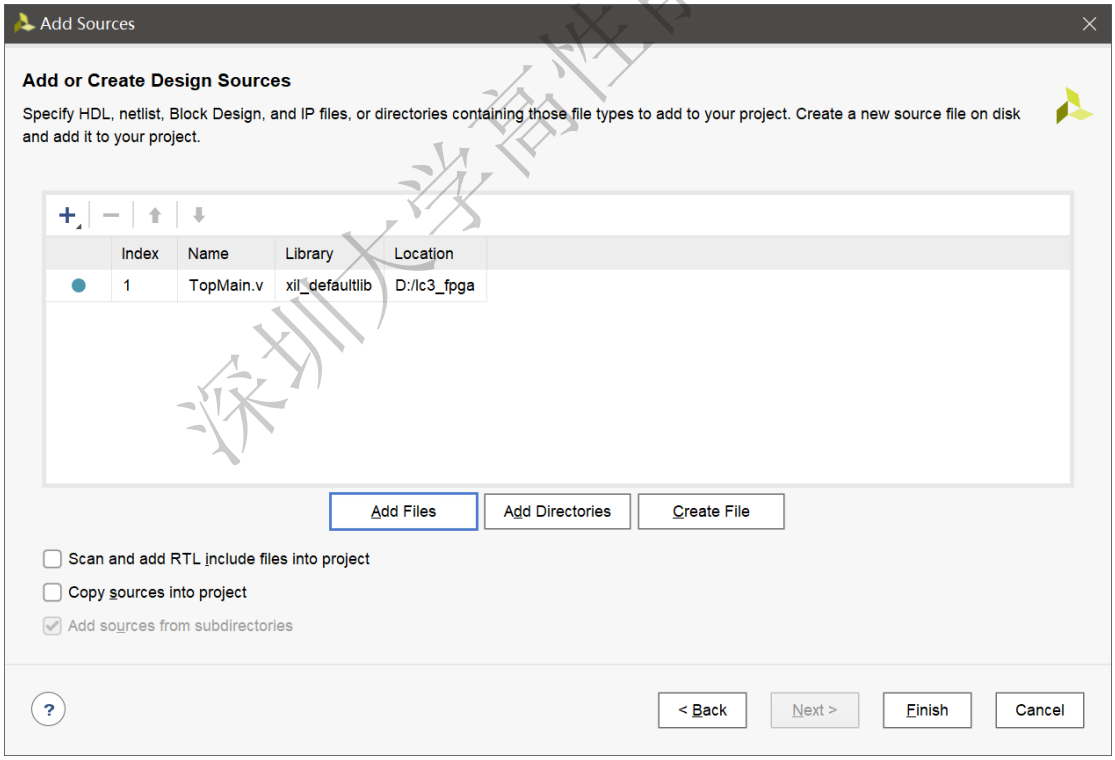



图 10.23

导入成功后，你可以在 vivado 的 Sources 窗口中看到 LC3 的几个基本模块，如图 10.24 所示，但是可以看到在 memory 中 dual_mem 是一个没有找到对应模块的状态。接下来就要在 vivado 中添加对应的 ip 核。

深圳大学高性能研究所

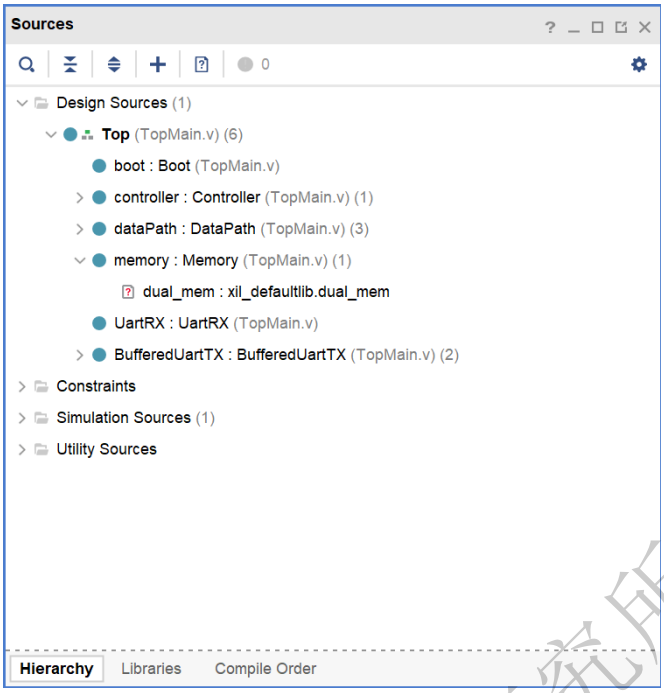



图 10.24

添加 ip 核，首先点击 Flow Navigator 窗口中的 IP Catalog 按钮，在弹出的窗口中搜索 Block Memory，选择 Block Memory Generator，如图 10.25-10.26 所示。

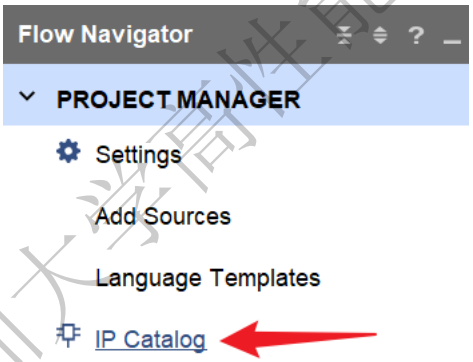



图 10.25

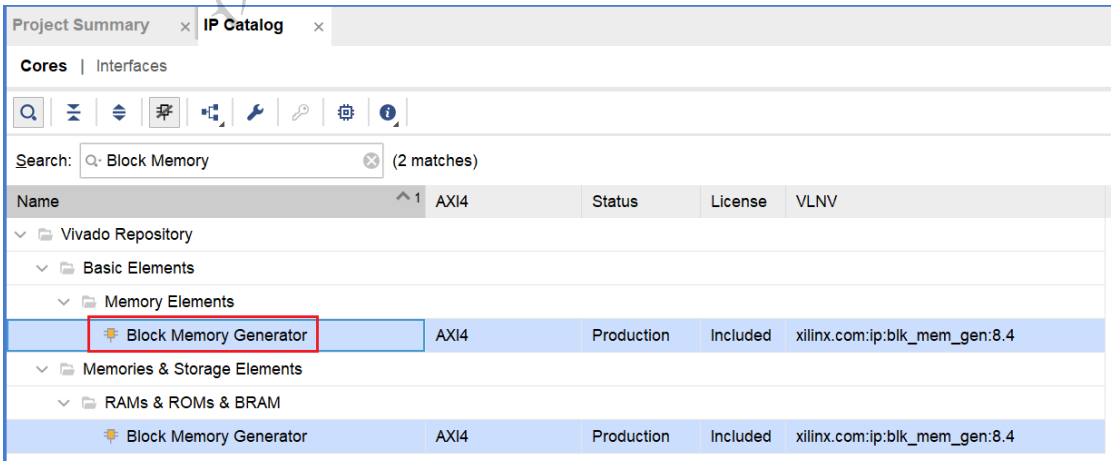



图 10.26

接下来会弹出 dual_mem IP 核的配置窗口，按照图 10.27-10.30 所示，配置好 IP

深圳大学高性能计算所

核，注意在图 10.30 中，红框框选住的地方，需要传入一个用于内存初始化的 init.coe 文件，这个文件就在 chsiel_lc3 项目的目录下可以找到，这个文件在某种程度上替代了 LC3 项目中不可综合的内存初始化逻辑，在 FPGA 运行时，会通过 init.coe 文件，将 LC3 中一些固定的 TRAP 指令程序初始化到内存中，保证程序的正确运行。

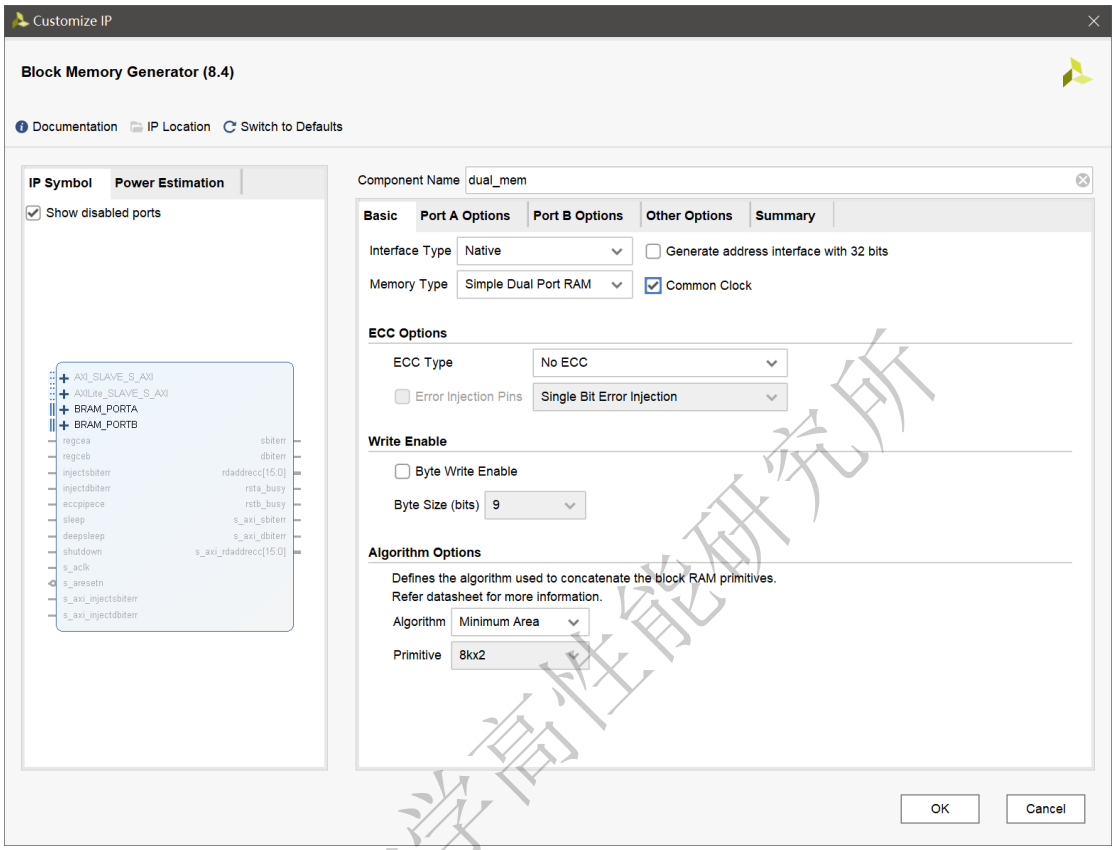



图 10.27

深圳大学高性能研究所

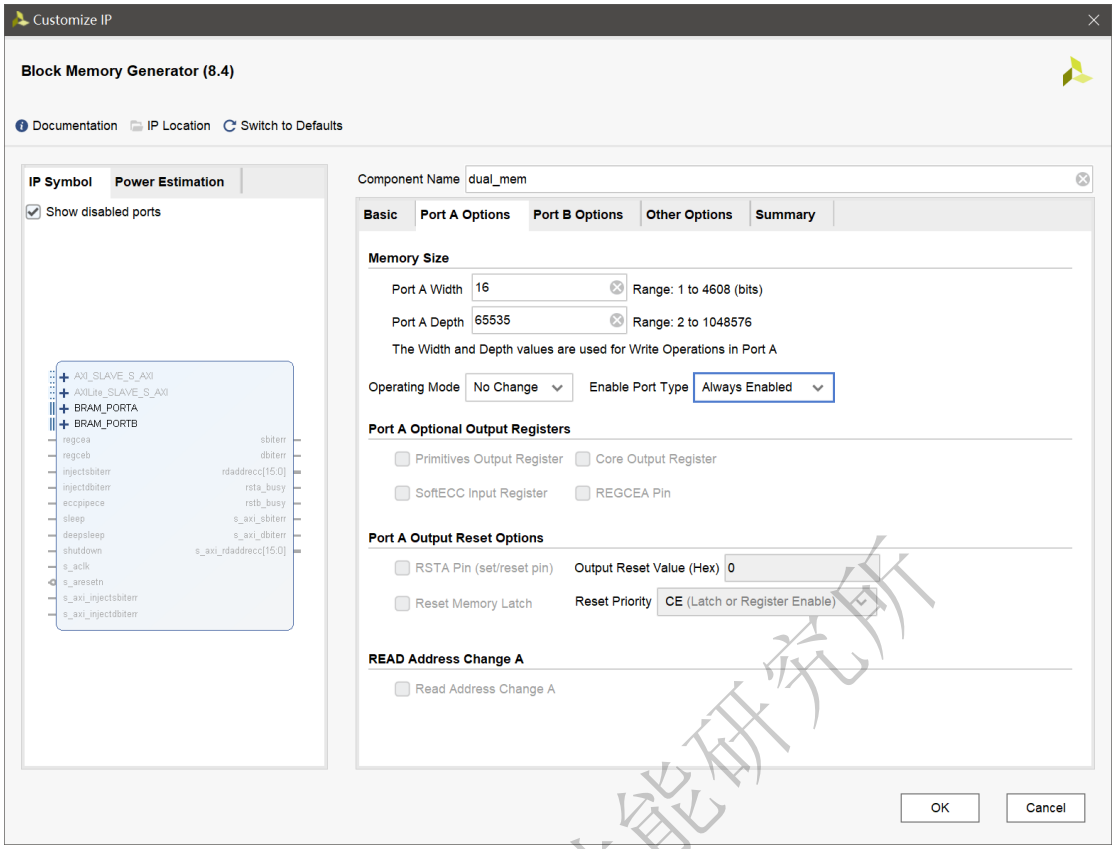



图 10.28

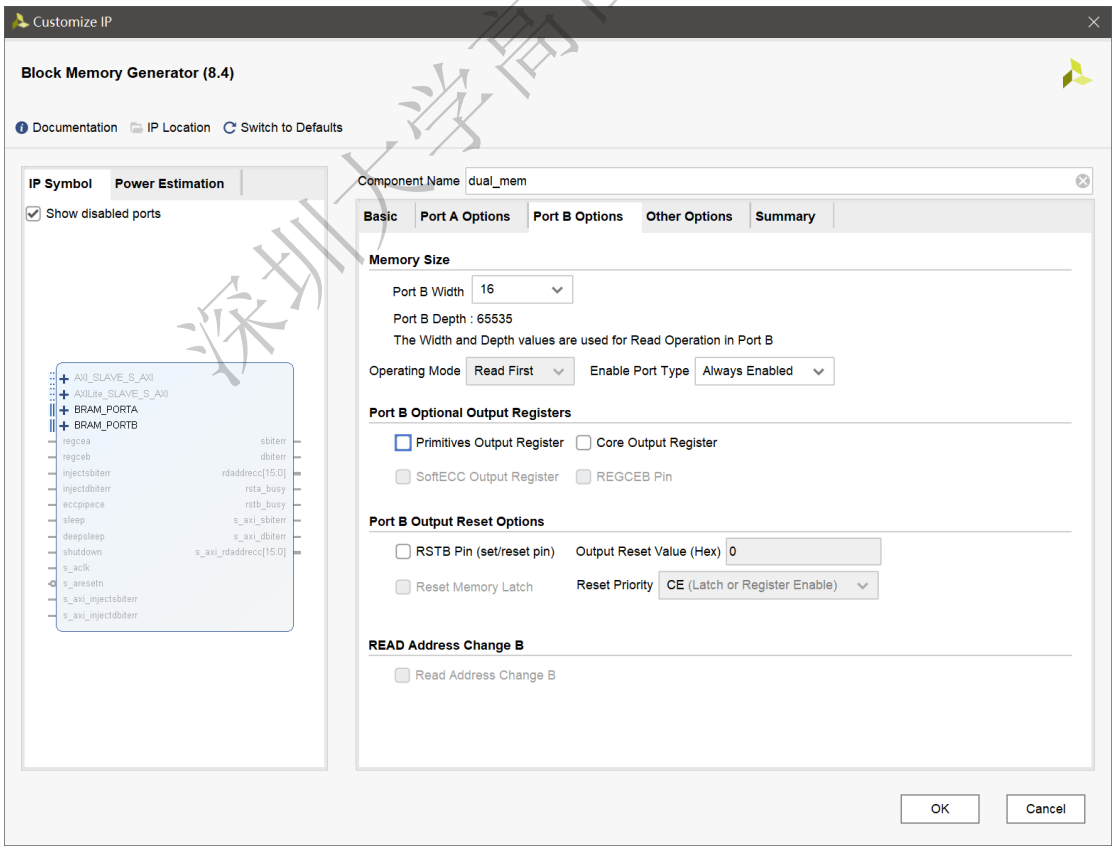



图 10.29

深圳大学高性能研究所

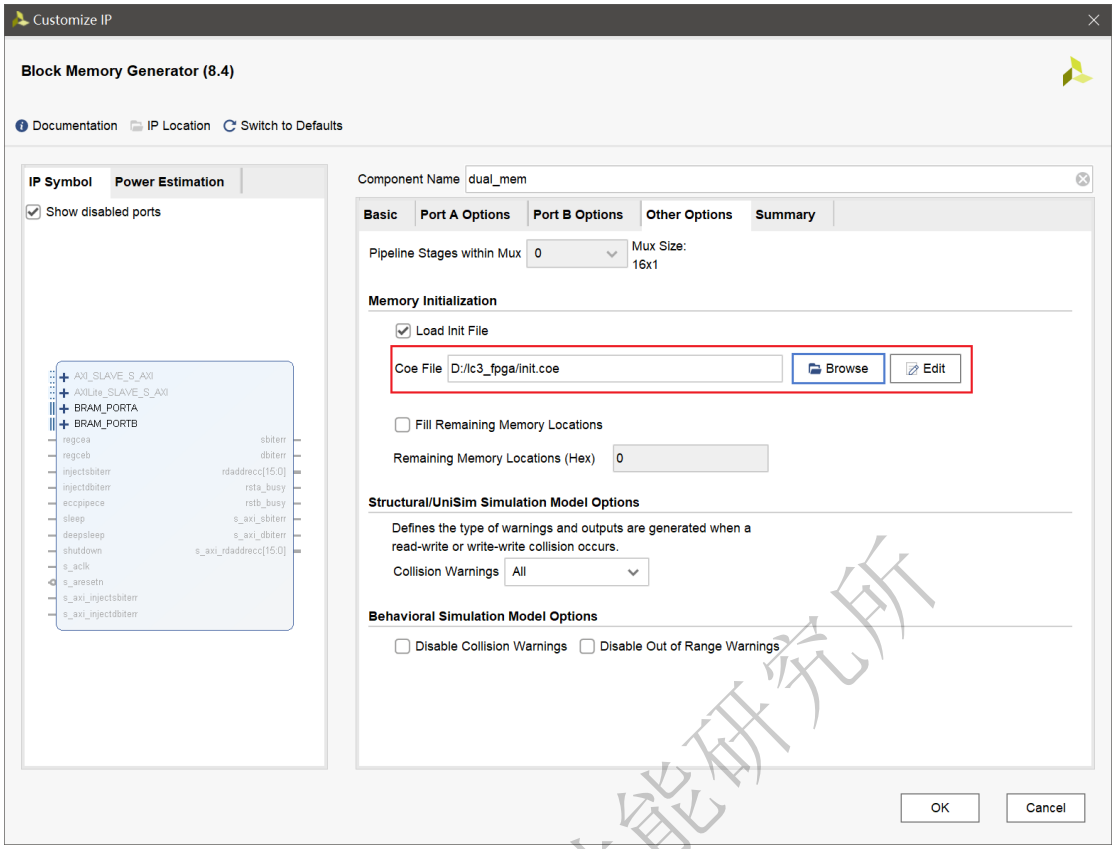



图 10.30

配置完成后点击 OK 按钮，会弹出如图 10.31 所示的窗口，点击 Generate 按钮然后等待界面右上角，如图 10.32 所示，这代表正在生成 ip 核，等待一段时间后，右上角变为图 10.33 所示时，代表 ip 核生成完毕，这可能会需要较长的时间，请耐心等待。

深圳大学高性能研究所

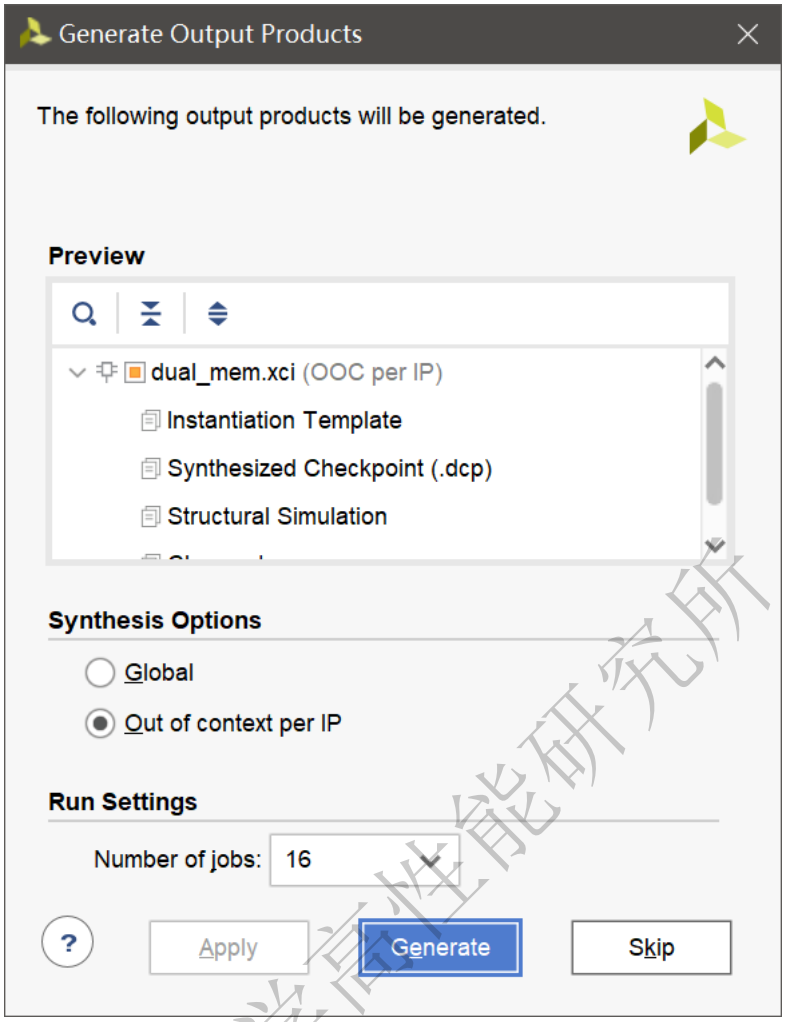



图 10.31

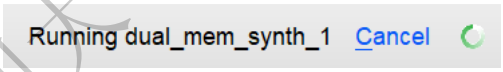



图 10.32


图 10.33

编译完成后在 Sources 窗口下点击切换到 IP Sources 标签页，如图 10.34 所示，可以看到产生的 ram，点开 dual_mem_stub.v 可以看到，我们之前 chisel 中定义的 dual_men 接口与生成的 ram 接口是一致的，与此同时，在 Hierachy 标签页中，dual_mem 的图标已经改变了。如图 10.35 所示。

深圳大学高性能研究所

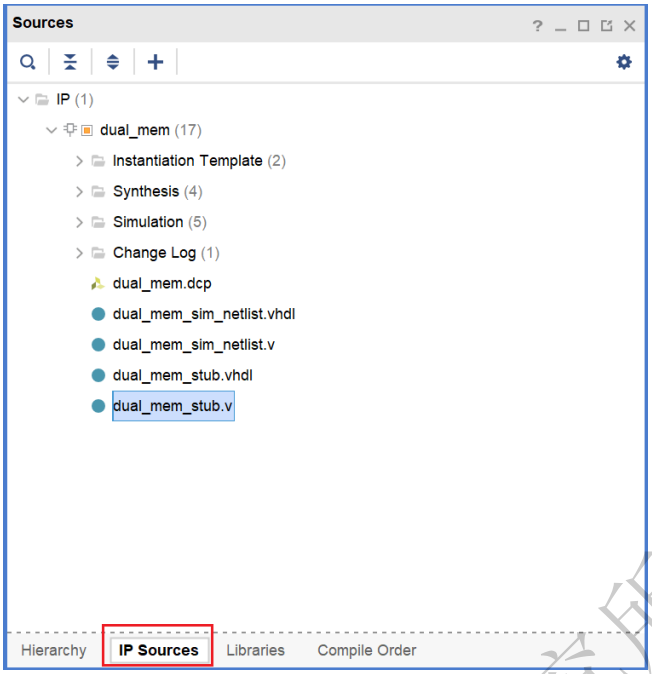



图 10.34

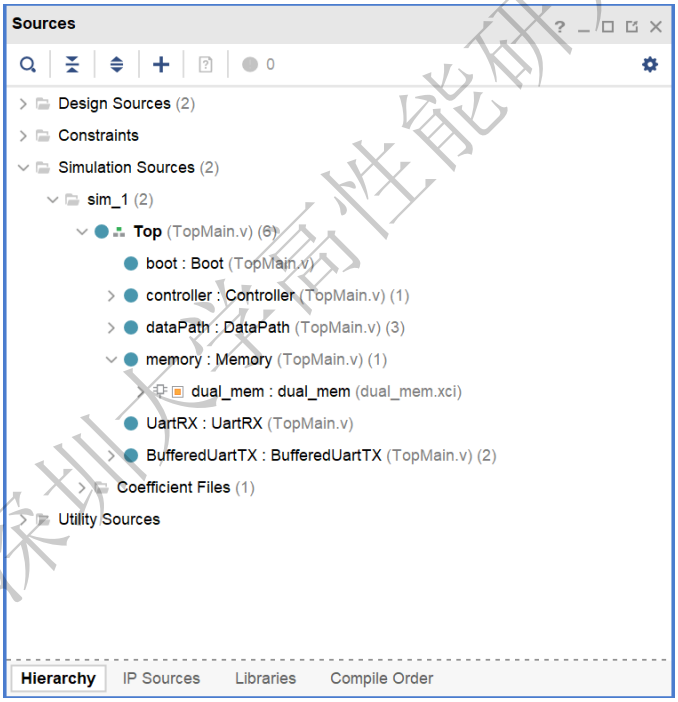



图 10.35

接下来我们要为 vivado 项目添加一个约束文件，这一步与添加 Source 文件类似，只是由选择 Add or create design sources 变为了 Add or create constraints，如图 10.36-10.37 所示。这个文件主要是用来将 LC3 的时钟、复位接口和 Uart 接口与 FPGA 上对应的引脚连接，注意这里我们并没有将 reset 按钮映射到 FPGA 的 reset 按钮上，而是映射到了 KEY0 按钮上。因为如果映射到 reset 按钮上，只有 reset 按钮一直处于按下的状态，系统才会正常工作。后半部分的功能主要是为了生成的 bit 流文件转换成固化文件后能够适用于 4bit 位宽 SPI 通信的 flash 器件，这个不必了解，只要写上即可，代码如下：

深圳大学高性能研究所

lc3.xdc

```
1 create_clock -period 20.000 -name clk [get_ports clock]
 2 set_property -dict {PACKAGE_PIN R4 IOSTANDARD LVCMOS33}
[get_ports clock]
 3 set_property -dict {PACKAGE_PIN T1 IOSTANDARD LVCMOS33}
[get_ports reset]
 4 set_property -dict {PACKAGE_PIN U5 IOSTANDARD LVCMOS33}
[get_ports io_uart_rxd]
 5 set_property -dict {PACKAGE_PIN T6 IOSTANDARD LVCMOS33}
[get_ports io_uart_txd]
 6
 7 set_property CFGBVS VCCO [current_design]
 8 set_property CONFIG_VOLTAGE 3.3 [current_design]
 9 set_property BITSTREAM.GENERAL.COMPRESS true [current_design]
10 set_property BITSTREAM.CONFIG.CONFIGRATE 50 [current_design]
11 set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
12 set_property BITSTREAM.CONFIG.SPI_FALL_EDGE Yes [current_design]
```

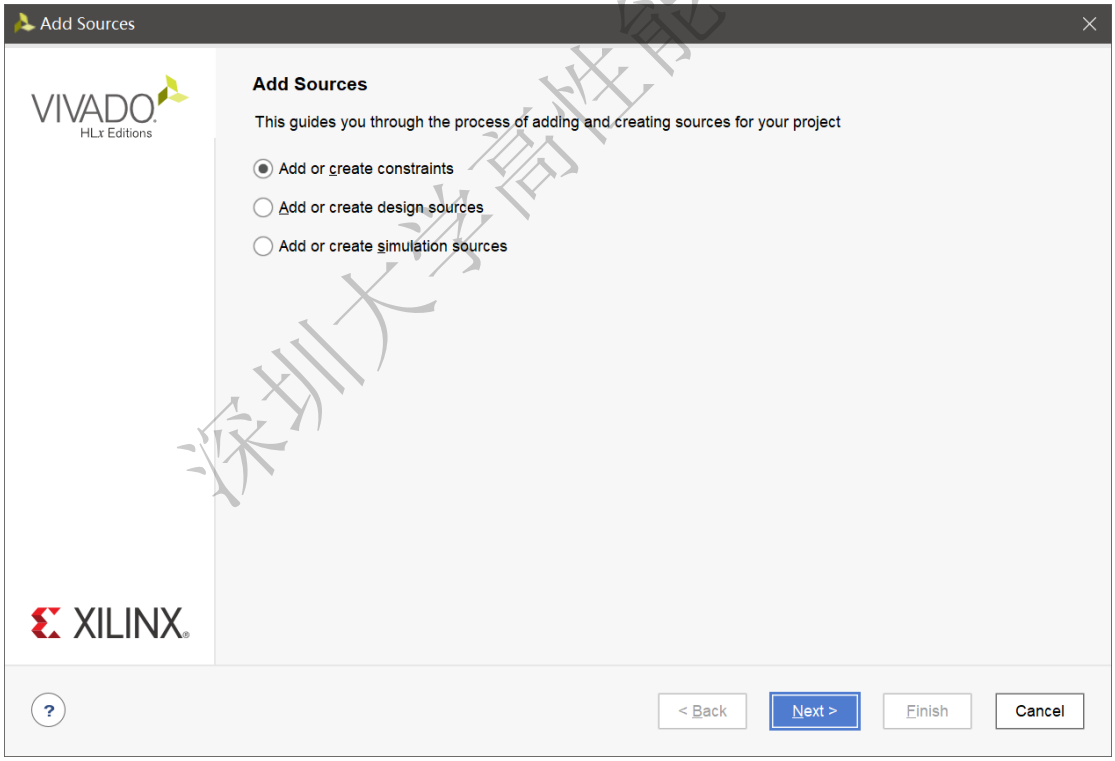



图 10.36

深圳大学高性能研究所

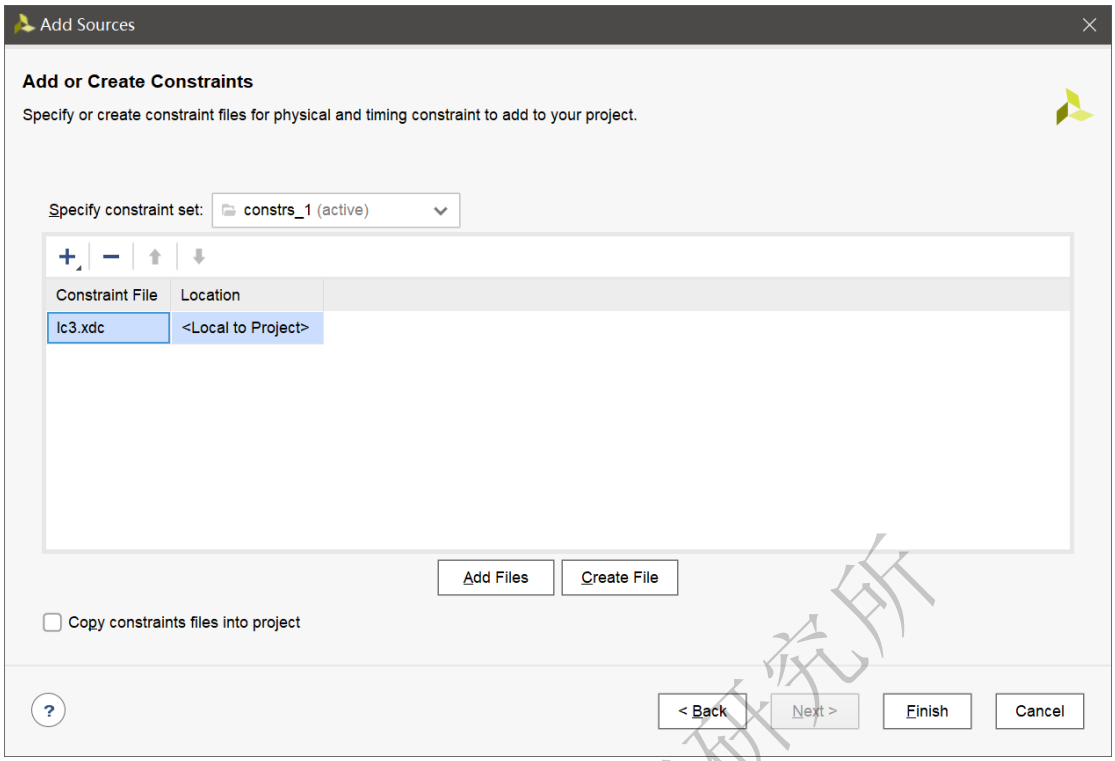
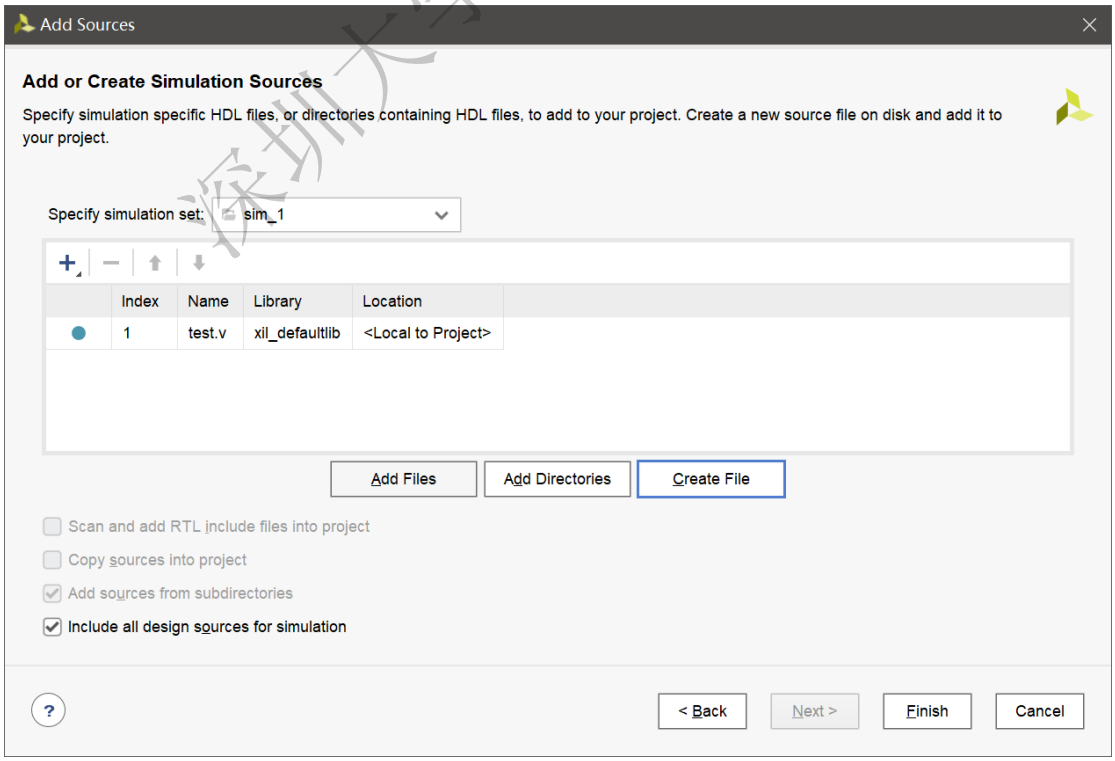



图 10.37

接下来我们要再在 vivado 中指定一个顶层，将我们的 Top 模块再包一层，这个顶层文件只在 vivado 做仿真时会使用，并不会真正被烧录到 FPGA 中。由于本实验不会在 Vivado 中运行仿真，因此只需要将这个文件添加进去即可。 和之前添加 TopMain.v 一样，只不过这次我们选择 Add or create simulation sources 选项。如图 10.38 所示，点击 Create File，然后给文件命名，点击 Finish 按钮。




深圳大学智能研究所

图 10.38

test.v 的代码如下，添加完 test.v 文件后，在 Sources 窗口中大概能看到如图 10.39 所示的组织结构。

test.v

```
1 `timescale 1ns / 1ps // 代表仿真时间单位/时间精度，这里代表的是 1ns 是基础的时间单位，而时间单位
最多可以精确到 1ps，例如使用 verilog 中的延时语句，#1 代表延时 1ns，1ps 表示延时最多可以精确到小数点后
3 位，即 0.0001ns
2 module test();
3         reg sys_clk;
4         reg sys_rst_n;
5         wire txd; // 连接 UART 接口
6           initial begin // 给时钟赋初值，reset 信号最开始是有效的，在 100ns 后 reset 信号撤销
7                 sys_clk = 1'b0;
8                 sys_rst_n = 1'b1;
9                 #100
10                sys_rst_n = 1'b0;
11        end
12        always #10 sys_clk = ~sys_clk; // 设置时钟每 10ns 反转一次，则一个时钟周期是 20ns
13
14        Top top( // 实例化 LC3 顶层模块
15                .clock(sys_clk),
16                .reset(sys_rst_n),
17                .io_uart_rxd(1'b0),
18                .io_uart_txd(txd)
19        );
20 endmodule
```

深圳大学高性能研究所

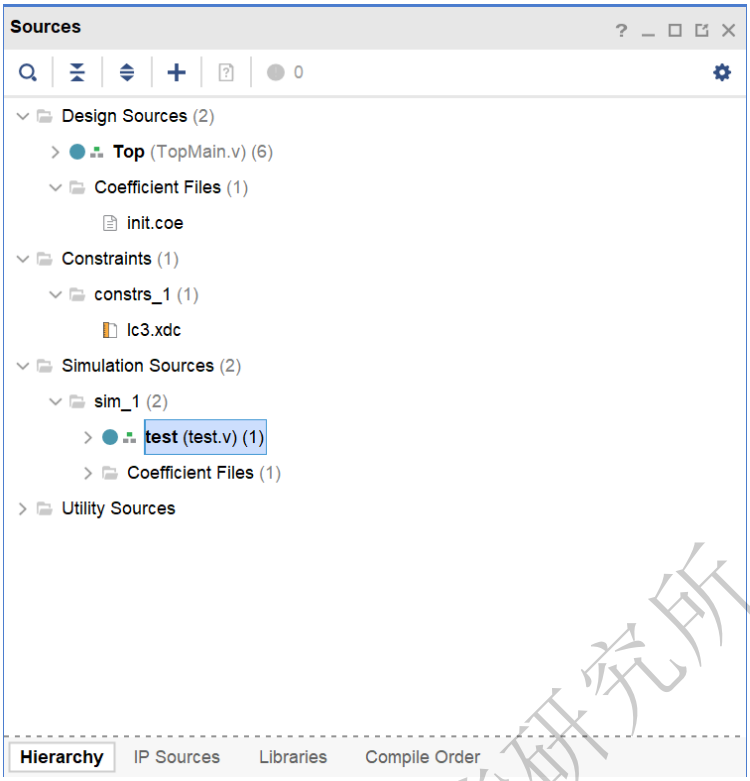
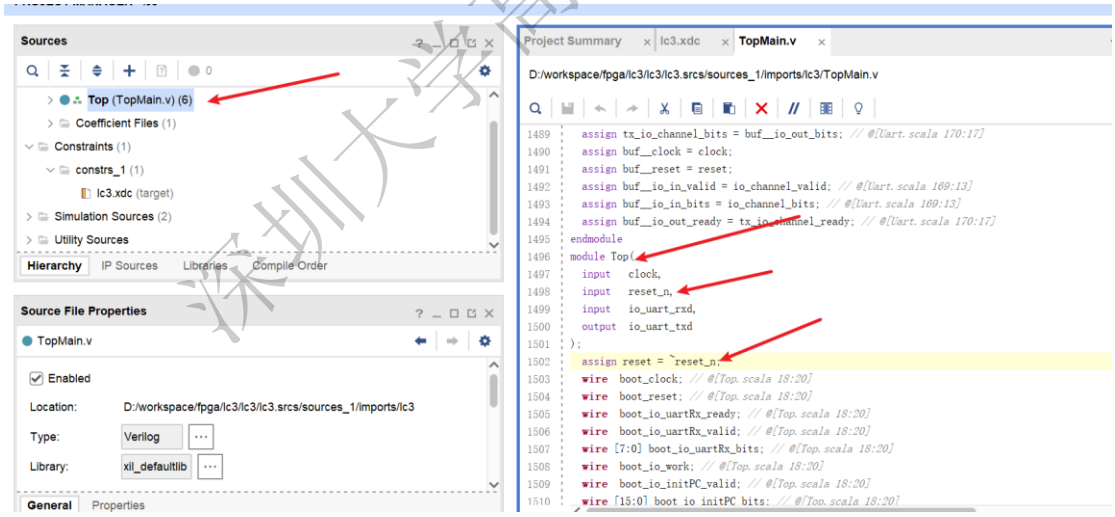



图 10.39

至此一个 LC3 的 Vivado 项目已经基本构建完成。

注意，此处需要手动修改 TopMain.v 的复位信号，将其取反：具体操作如下：




第 498 行将 reset 名字改为 reset_n，然后在第 1502 行中将 reset_n 信号取反赋值给 reset。同时，**需要在 xdc 约束文件中将 reset 引脚的名字改为 reset_n**。（这里的代码行数不一定相同，请找到对应的代码位置。）

## 3.4. 将 LC3 烧录到 FPGA 板卡上

在 Vivado 项目搭建完成后，接着要对硬件电路进行综合，生成用于烧录到 FPGA 上的文件，点击 Flow Navigator 窗口中的 Generate Bitstream 按钮，如图

深圳大学高性能研究所

10.40 所示，综合生成 bit 文件，弹出的对话框中不用修改配置，点击 OK 即可，如图 10.41 所示。这部分过程比较久，请耐心等待，可以看 vivado 右上角来判断状态。生成完成后会弹出对话框，选择 Generate Memory Configuration on File，然后点 OK，如果一不小心关闭了这个对话框，也可以在菜单栏的 Tools 菜单中找到。

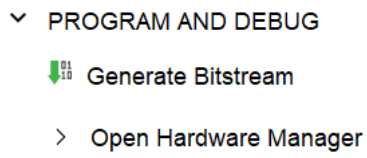



图 10.40

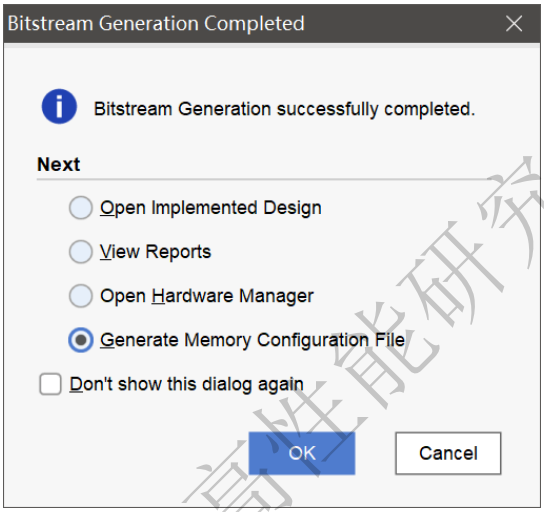



图 10.41

接着按照图 10.42 格式进行配置，点击 OK 后，看到弹出对话框，表示 mcs 文件已经生成成功。

深圳大学高性能研究所

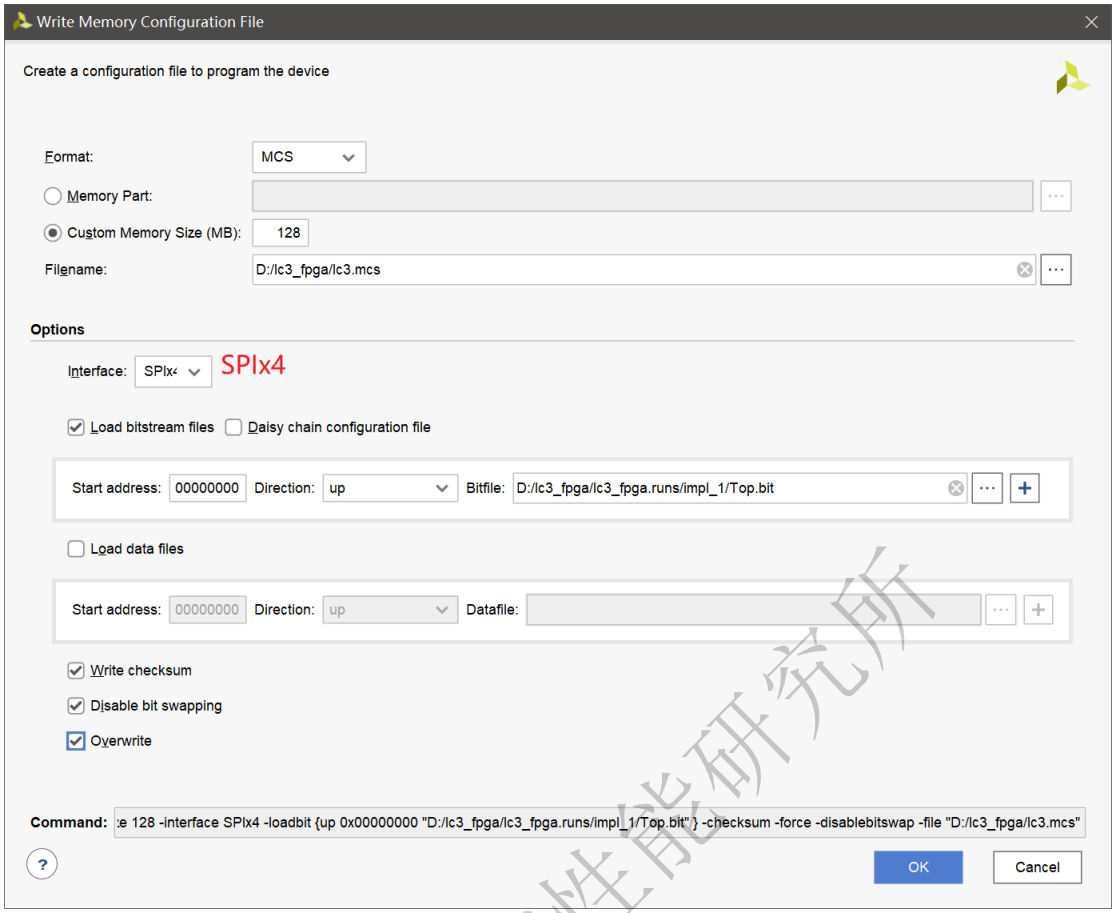



图 10.42

接下来就要正式开始烧录了，首先我们需要连接好开发板，首先我们需要连接 JTAG 下载接口，通过这个接口我们将我们的程序烧录到 FPGA 上自带的 Flash 中，这样在开机后 FPGA 就会从 Flash 中运行我们烧录好的程序。

其次一个系统必须要有输入输出，因此我们还需要连接 UART 接口，同时在电脑上安装串口调试助手，来传输和接收我们的程序输入输出。在连接完成后按下蓝色开关，给 FPGA 上电。具体的连接如图 10.43 所示。

深圳小眼睛性能研究所

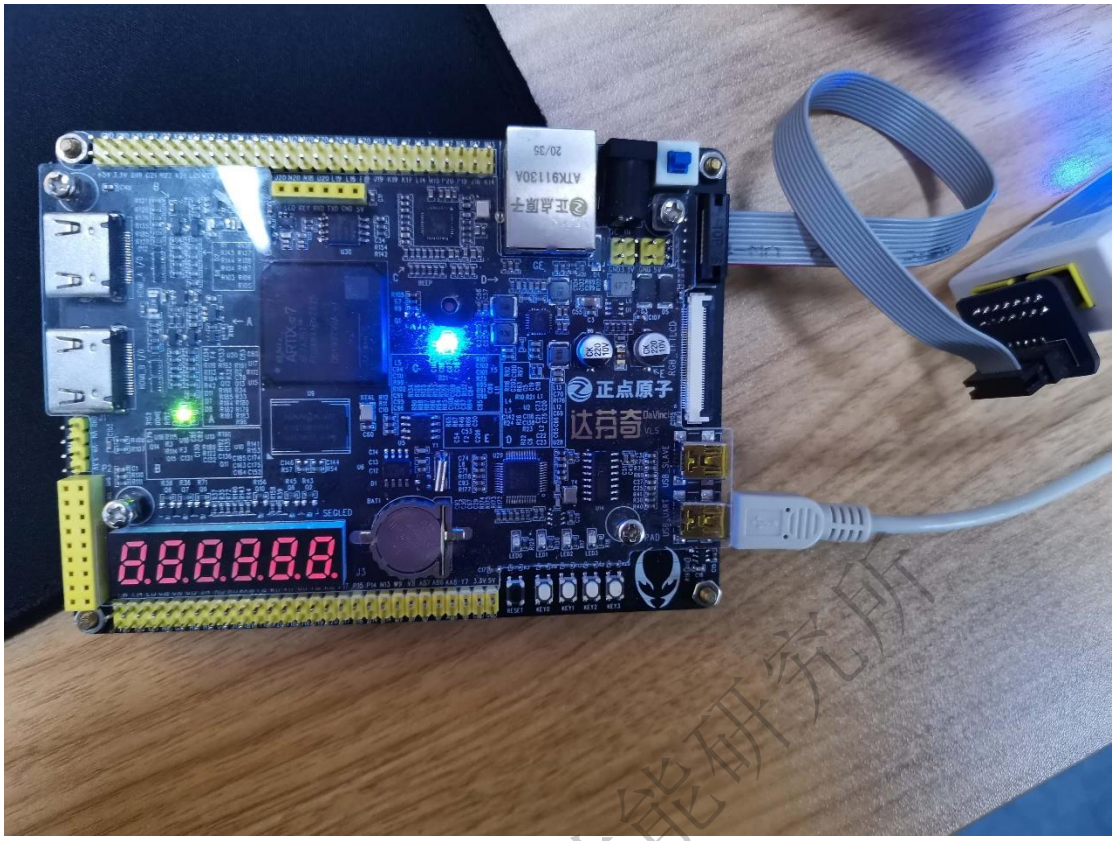



图 10.43

接下来在 Flow Navigator 窗口中点击 Open Hardware Manager 按钮，如图 10.44 所示。点击 Hardware 窗口中的 Auto Connect 按钮。看到如图 10.46 所示的画面，代表已经连接成功了。

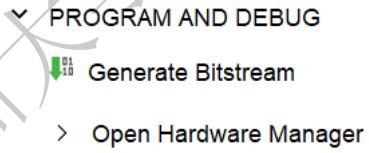



图 10.44

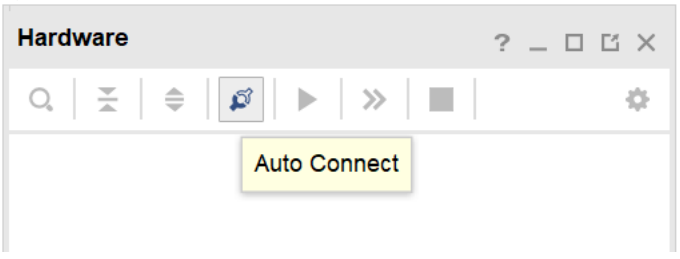



图 10.45

深圳大学高性能研究所

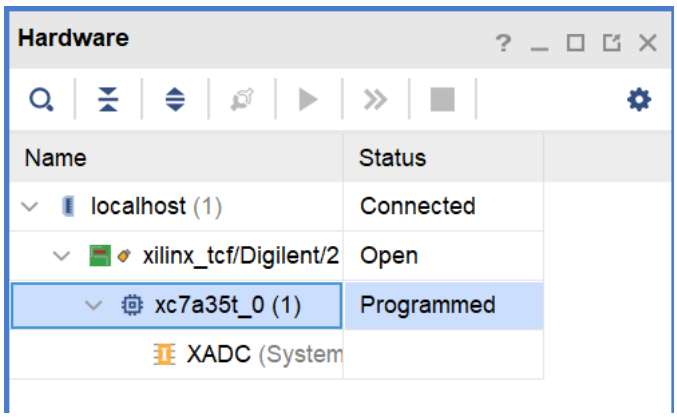



图 10.46

接下来我们要在项目中为开发板添加一个固化 Flash 部件，选中芯片右键选择 Add Configuration Memory Device。如图 10.47 所示。然后根据图 10.48 所示，选择 FPGA 板卡上的 Flash 芯片型号。

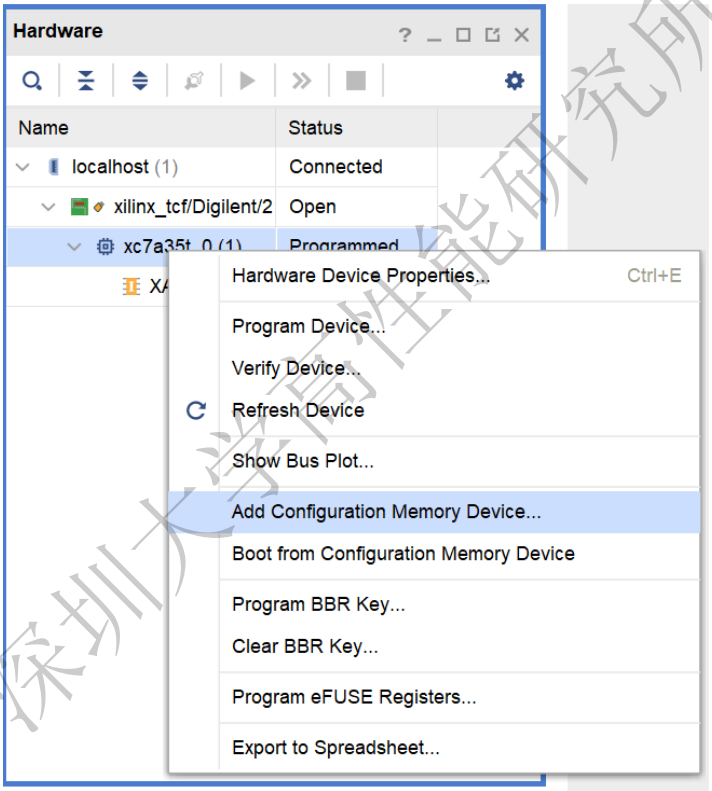



图 10.47

深圳大学高性能研究所

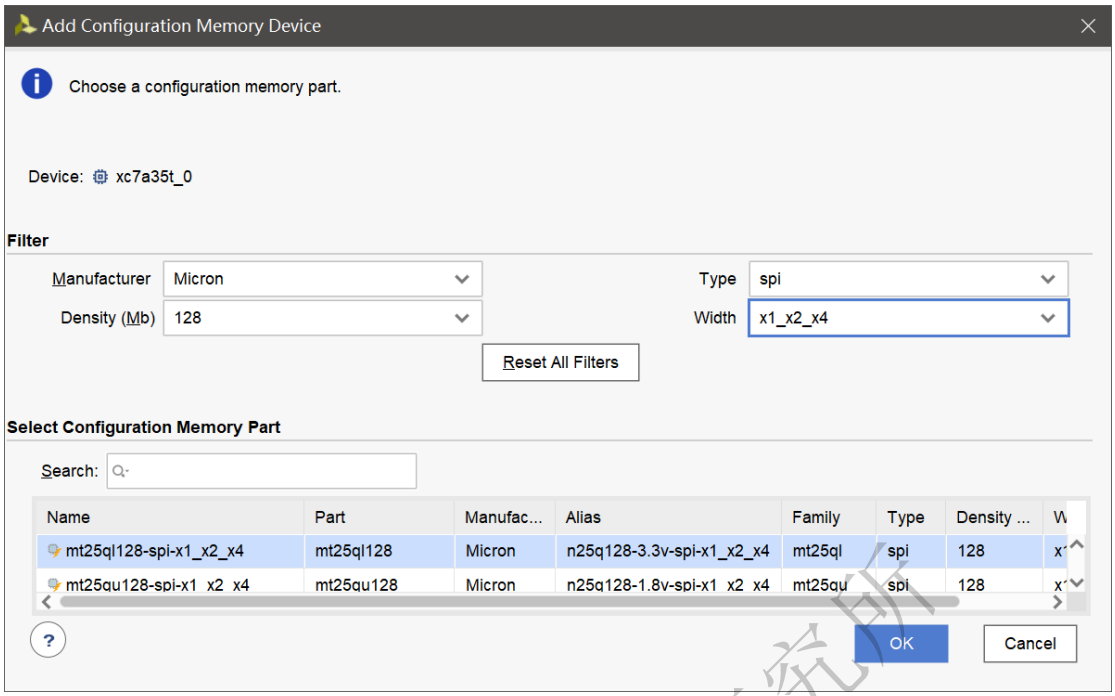



图 10.48

然后会询问你是否要烧写新添加的 Flash，选择 OK，如果不小心关闭了对话框，在新添加的 Flash 上右键，选择 Program Configuration Memory Device。

接下来要选择刚才生成的 mcs 文件，还有和 mcs 文件在同一目录下的 prm 文件，按照图 10.49 配置完成后点击 OK，就会开始烧录，烧录完成后会弹出窗口，此时在 Hardware 窗口中可以看到多出了一个 Flash 的图标。如图 10.50 所示。

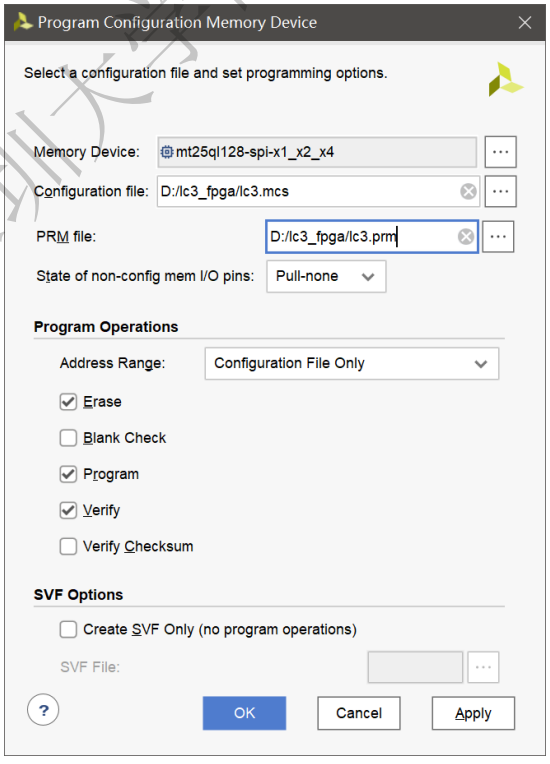



图 10.49

深圳大学高性能计算研究所

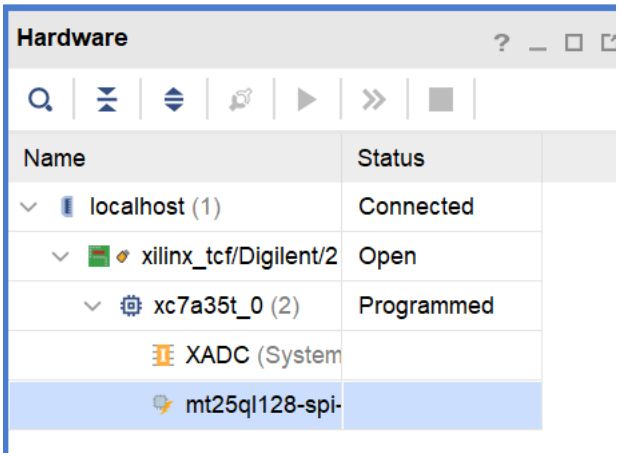



图 10.50

接下来我们打开串口调试助手，在 FPGA 上电启动的情况下，我们选中 FPGA 对应的串口（不同的电脑可能对应的串口号不同，指导中的是 COM4），然后配置好对应的波特率等参数，选择发送文件，然后将希望运行的程序的 obj 文件通过串口传输给 FPGA，其中的 LC3 启动程序会自动接收你想要运行的程序，将它存入 RAM 中，然后开始执行这里可以使用 chisel_lc3 中的 dummy.obj 文件。图 10.51 是一个最简单的程序，它通过串口输出 Hello!

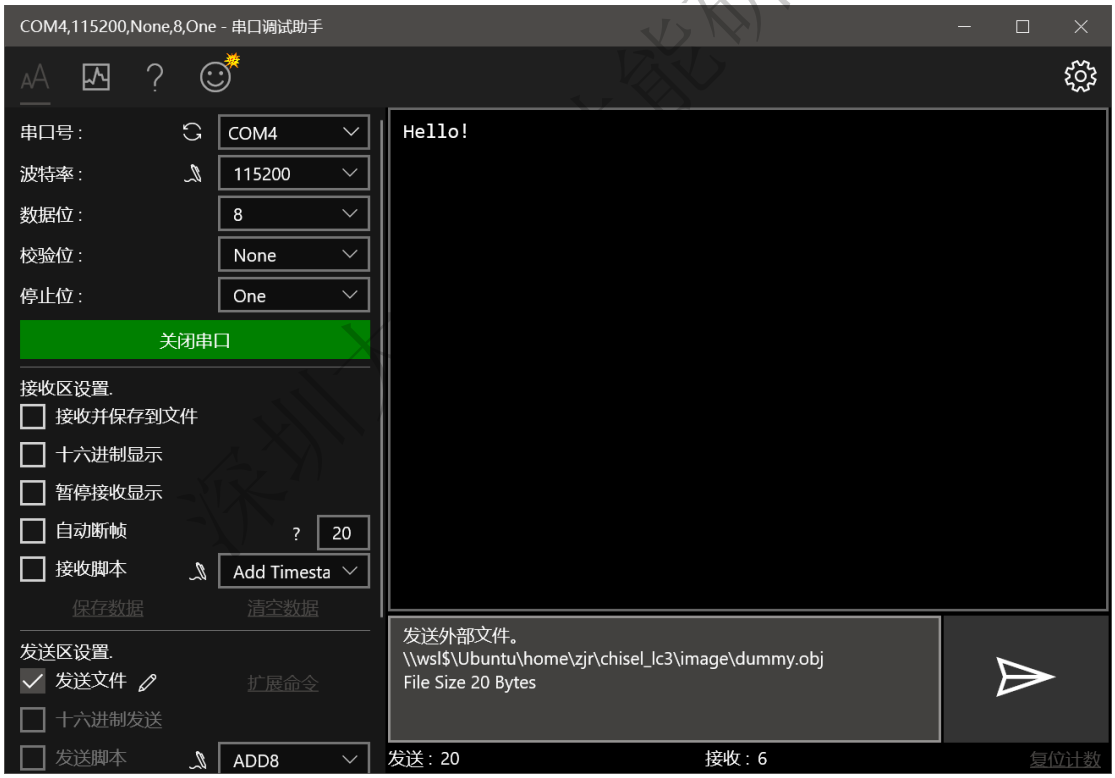



图 10.51

当然我们也可以运行一些带交互的程序，下图首先给 FPGA 传输一个计算机系统实验四里的 MIN 小游戏程序，然后在通过串口输入与 LC3 交互，实现游玩游戏的过程。这个程序的 obj 文件也可以在实验材料中找到。运行结果如图 10.52-10.53 所示。

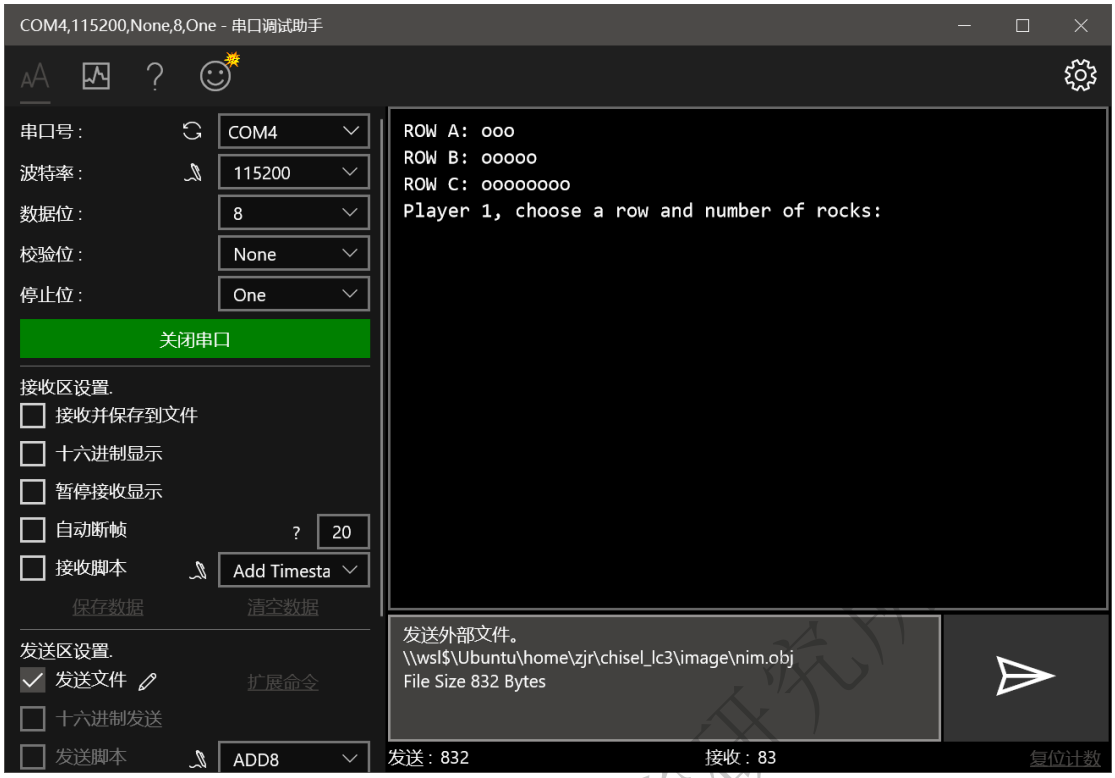



图 10.52

深圳大学高性能计算研究所

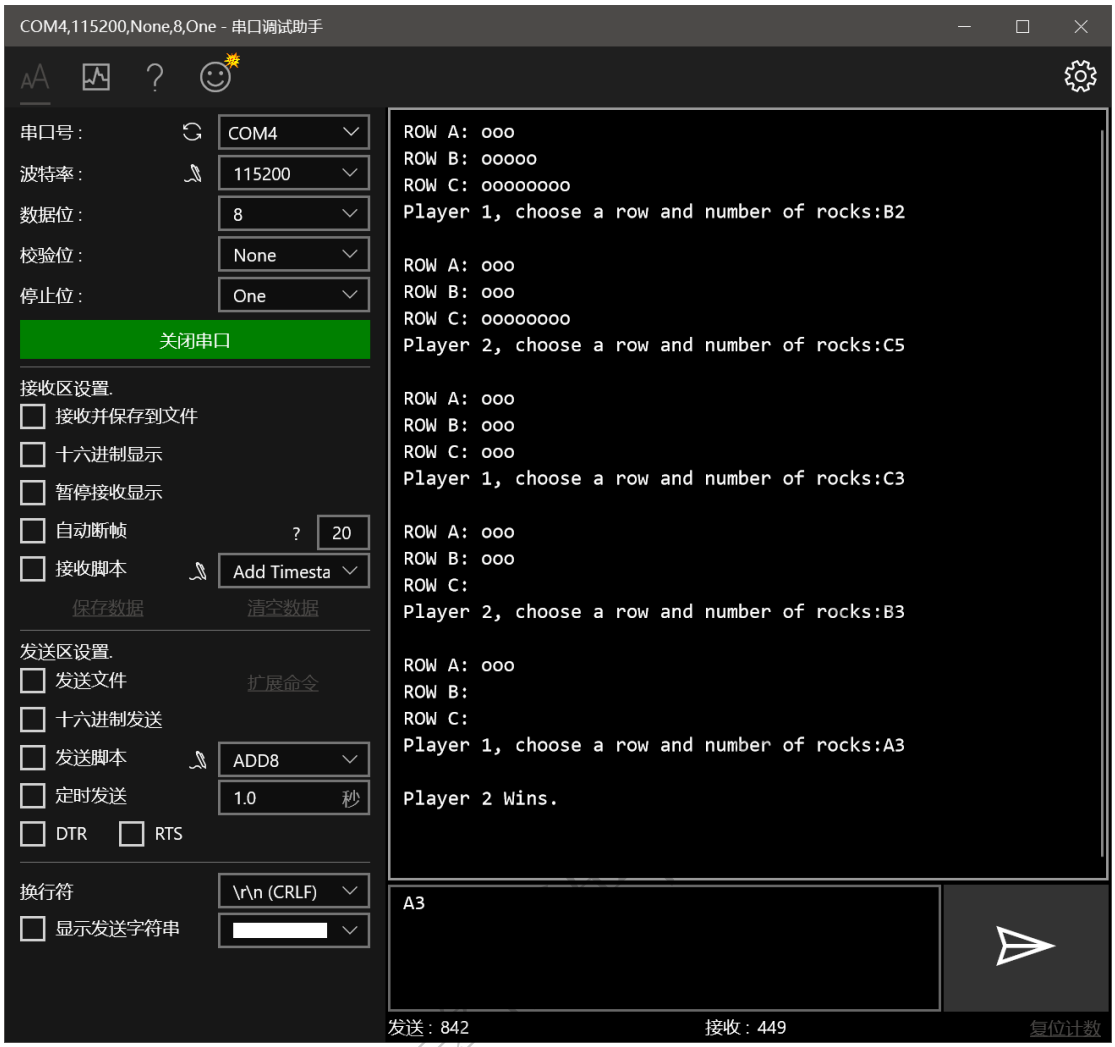



图 10.53

**实验十-任务二：按照实验指导，成功在 FPGA 上运行 LC3 系统，并运行 dummy.obj 程序，在串口调试助手中观察到输出**

附：

其他同学的经验分享，大家可以对照参考。

1. 修改 Top.scala 中的 FPGAPlatform 变量为 true 时,Replace_Mem 也要修改为 true，不然 Memory 内部的模块不会被替换为 dual_mem，
2. 进行串口调试时最后没看到输出，一开始以为是串口驱动没装，但后来装了也还是没有输出，然后按复位键也没反应，后面才知道讲义中给的 xdc 文件中约束到的是 FGPA 的 T1 管脚，对应的按键是 key0（这个没看具体的电路还真不知道），前面我一直按的是 reset 键所以没反应；
3. 在 Vivado 中导入 TopMain.v 的文件后，在配置 ip 核的过程中报错，这个一开始工程文件在 D 盘时怎么弄都报错，最后换到 C 盘中就解决了（虽然还是不知道具体的原因，可能是个人电脑的问题）；