

实验步骤

利用HLS设计加速系统包括3个阶段：

- 1. 利用HLS生成IP核；
- 2. 创建Block Design；
- 3. 系统的软硬件联调。

1. 利用HLS生成IP核

1.1 创建Vivado HLS项目

在桌面双击Vivado HLS快捷方式，或在开始菜单->Xilinx Design Tools->Vivado HLS打开Vivado HLS工具。点击“Quick Start”下方的“Create New Project”，或点击菜单栏中的File->New Project。然后，填写项目名称，并选择项目所在的路径，如图2-1所示。

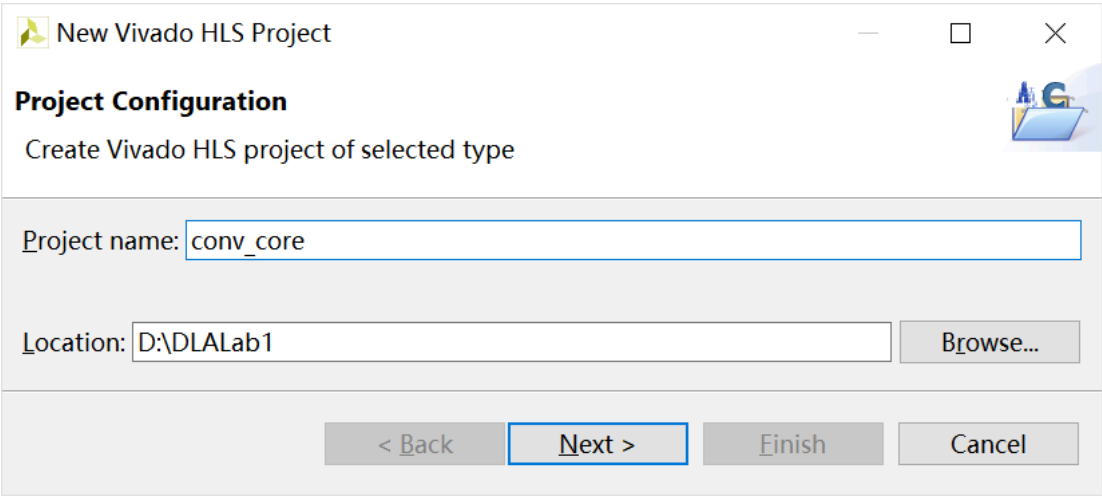


图2-1 创建Vivado HLS项目

点击Next，进入源文件添加窗口。在Top Function一栏内填写 `conv_core`，作为顶层函数名，如图2-2所示。

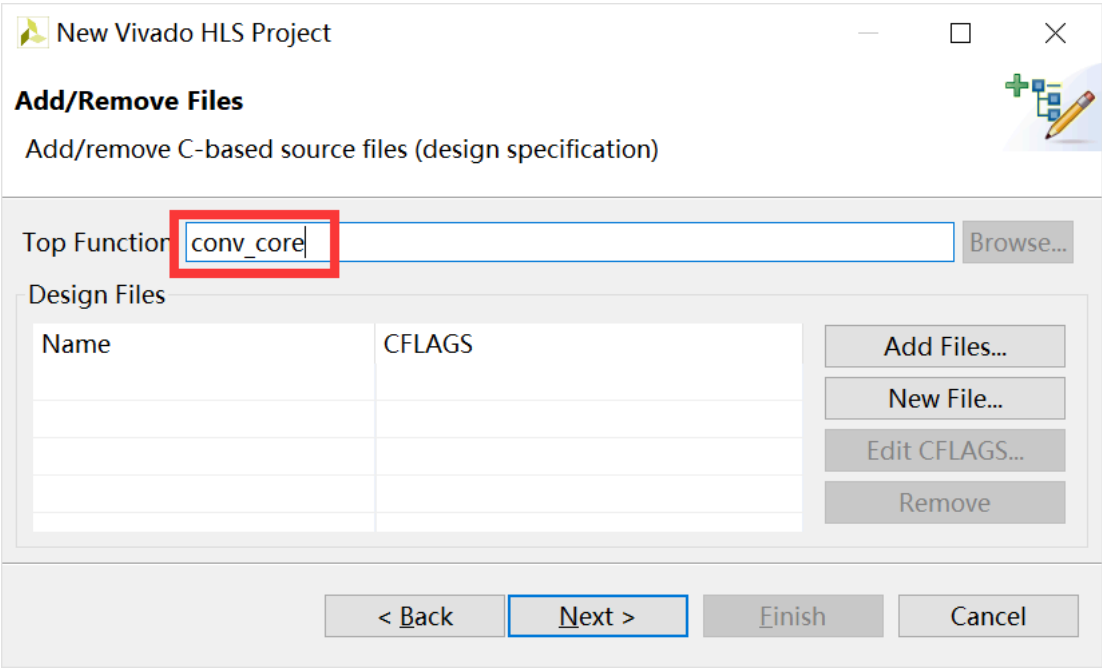


图2-2 填写Top Function名

点击Next，进入Test Bench的测试源文件添加窗口。此时暂不添加，继续点击Next，进入项目配置窗口。点击Part Selection线框内部的“...”按钮，打开器件选择窗口，输入xc7z020clg400-1，然后点击“OK”按钮，如图2-3所示。

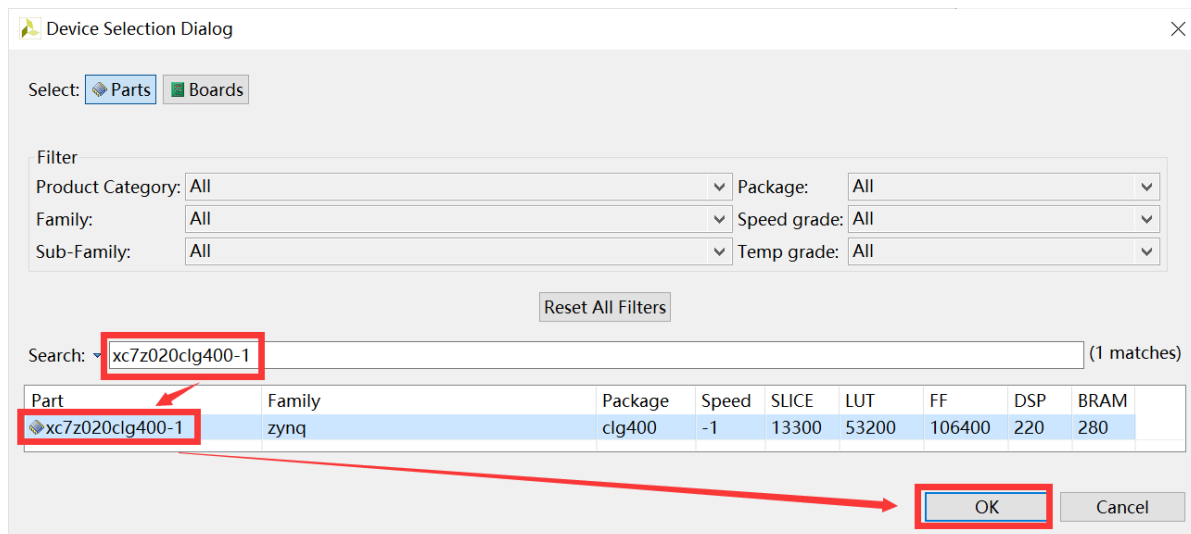


图2-3 选择器件

点击“Finish”按钮，完成项目的创建，进入主界面，如图2-4所示。



图2-4 Vivado HLS主界面

Vivado HLS主界面主要包括6个部分：1) 菜单栏和工具条。菜单栏和普通的IDE差不多，尤其接近Eclipse，这是因为Vivado HLS就是基于Eclipse开发的。工具栏包含了常用的功能按钮，如综合、仿真、打包IP核等。2) 视图切换按钮，用于在Debug、综合和分析3个视图之间切换。3) 项目管理窗口，用于管理项目源文件、测试源文件、IP核压缩包等文件。4) 工作区：用于编辑源码和调试。5) 控制台和报错信息窗口：用于查看报错信息和串口输出等。6) Outline窗口：用于显示项目相关的信息，包括顶层函数输入输出变量所对应的硬件模块的接口信号总线协议等。

1.2 添加源文件

将所提供的 conv_core.cpp、conv_core.h 和 main.cpp 三个源文件拷贝至工程目录下（D:\DLALab1\conv_core）。然后在项目管理窗口的Source图标处右键，选择Add File，选择刚才拷贝的除 main.cpp 之外的源文件，如图2-5所示。

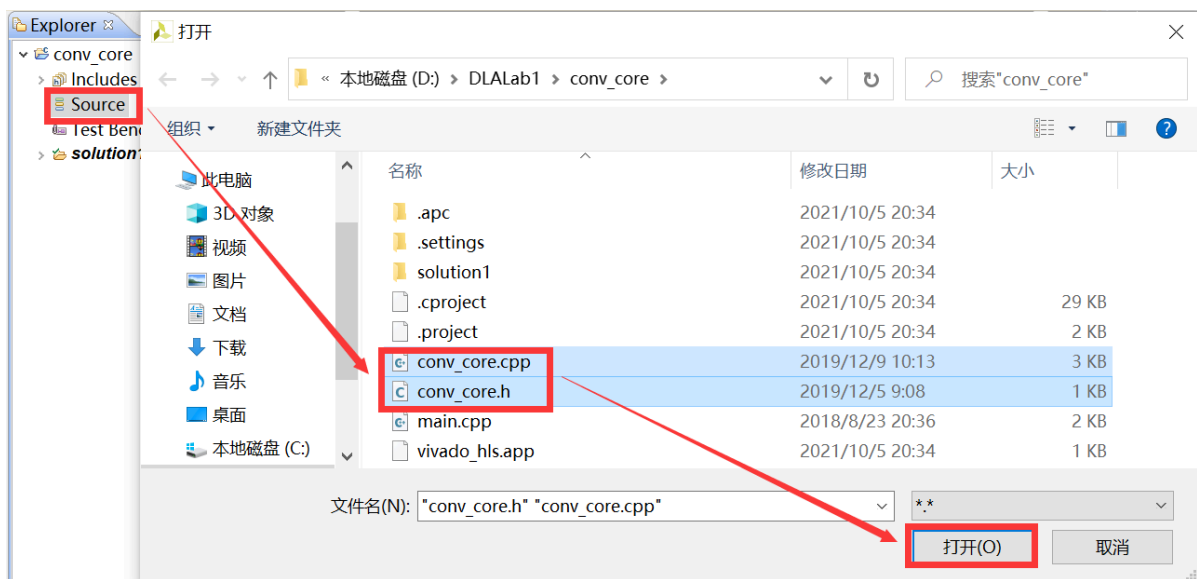


图2-5 添加源文件

双击Source下的源文件。可编辑其代码，如图2-6所示。

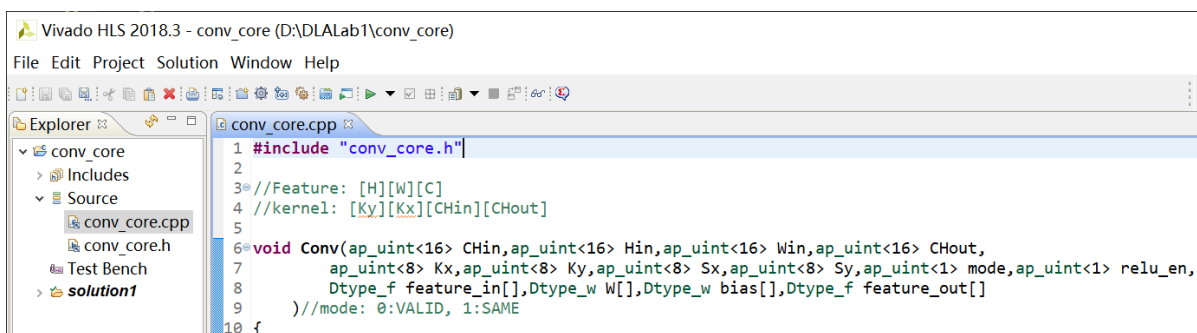


图2-6 查看或编辑源代码

类似地，在Test Bench图标处右键添加 main.cpp，如图2-7所示。

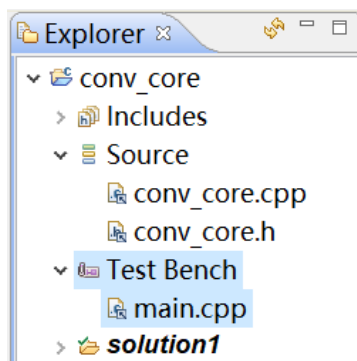


图2-7 添加测试源文件

1.3 C代码仿真/CSIM

点击Project->Run C Simulation或工具条中的CSim按钮，如图2-8所示。

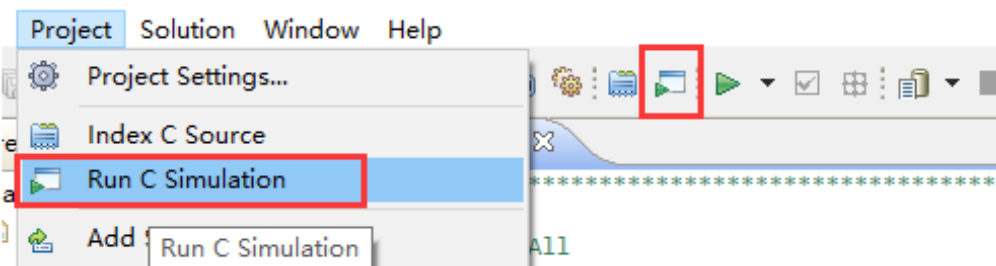
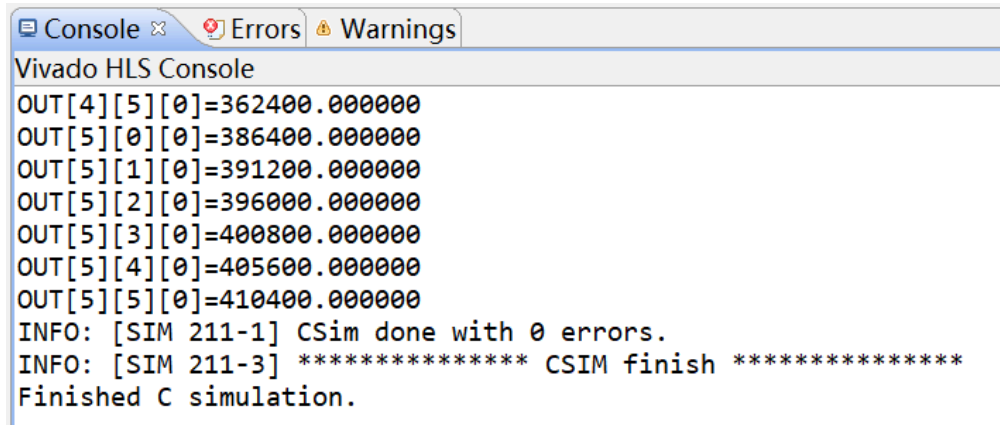


图2-8 点击进行CSim

点击CSim后，直接在弹出的窗口中点击“OK”按钮以开始C语言仿真。CSim是纯软件的仿真，仅用于验证代码功能的正确性。

CSim完毕后，工作区将弹出一个名为 `conv_core_csim.log` 的CSim日志文件以便查看仿真结果，也可在控制台窗口查看源码的编译过程及相应的测试结果，如图2-9所示。



```
Vivado HLS Console
OUT[4][5][0]=362400.000000
OUT[5][0][0]=386400.000000
OUT[5][1][0]=391200.000000
OUT[5][2][0]=396000.000000
OUT[5][3][0]=400800.000000
OUT[5][4][0]=405600.000000
OUT[5][5][0]=410400.000000
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
Finished C simulation.
```

图2-9 查看CSim结果

1.4 C代码调试

如果CSim时不能通过测试，则需要对代码进行debug。Vivado HLS工具的调试方法和Eclipse基本一致。

点击CSim按钮，在弹出的窗口中勾选Launch Debugger，如图2-10所示。

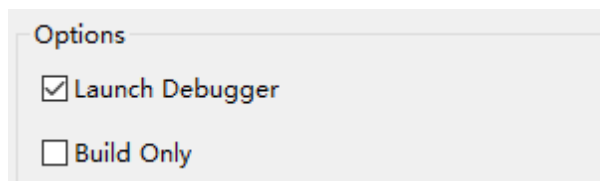


图2-10 开始debug

点击“OK”按钮，将自动进入跟Eclipse一样的Debug视图。可在代码行数的左侧空白处双击以添加断点，如图2-11所示。

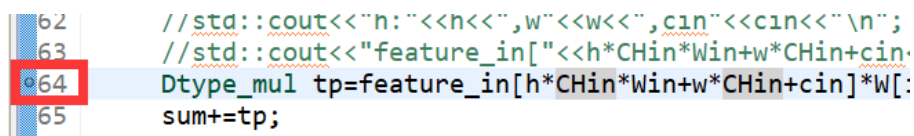


图2-11 添加断点

点击工具条中的Step Into(F5)或Step Over(F6)按钮，开始debug。此时，可在右上方的Variable窗口查看变量的值，如图2-12所示。

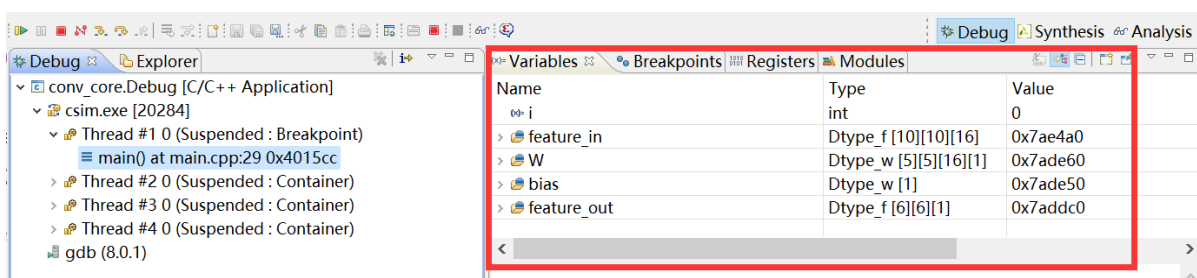


图2-12 debug时查看变量的值

debug完成后，点击图2-12左上角形如红色正方形的按钮，或用快捷键Ctrl+F2结束调试。此时，再点击图2-12右上角的Synthesis按钮，回到图2-4所示的主界面/综合界面。

1.5 综合/Synthesis

综合前，需要为 `conv_core` 项目添加Directive/原语，以指定顶层模块所对应的硬件模块的IO信号所使用的总线协议、综合时所使用的并行优化策略（如流水线、循环展开）等。

Directive的添加有2种方法。第一种方法是通过图形化界面添加。添加时，需要在Outline窗口处点击打开Directive标签页，并在相应的函数上右键添加Directive，如图2-13所示。

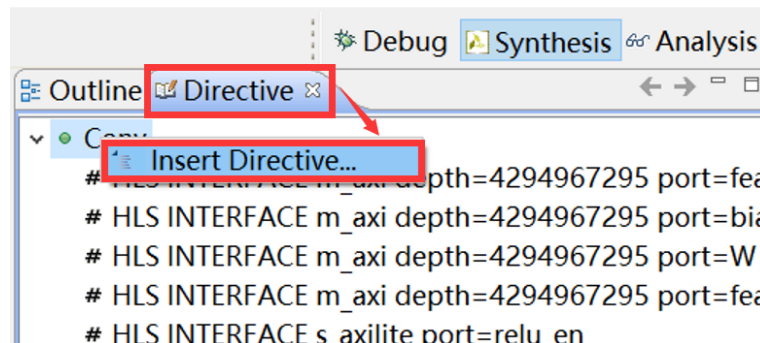


图2-13 用图形化界面添加Directive

第二种方法是直接在代码中插入形如 `#pragma HLS xxx` 的制导语句，如图2-14所示。

```
6 void Conv(ap_uint<16> Chin,ap_uint<16> Hin,ap_uint<16> Win,ap_uint<16> CHout,  
7         ap_uint<8> Kx,ap_uint<8> Ky,ap_uint<8> Sx,ap_uint<8> Sy,ap_uint<1> mode,ap_uint<1> relu_en,  
8         Dtype_f feature_in[],Dtype_w W[],Dtype_w bias[],Dtype_f feature_out[])  
9     //mode: 0:VALID, 1:SAME  
10 {  
11     // #pragma HLS PIPELINE enable flush  
12     #pragma HLS INTERFACE m_axi depth=4294967295 port=feature_out offset=slave  
13     #pragma HLS INTERFACE m_axi depth=4294967295 port=bias offset=slave  
14     #pragma HLS INTERFACE m_axi depth=4294967295 port=W offset=slave  
15     #pragma HLS INTERFACE m_axi depth=4294967295 port=feature_in offset=slave  
16     #pragma HLS INTERFACE s_axilite port=relu_en  
17     #pragma HLS INTERFACE s_axilite port=CHout  
18     #pragma HLS INTERFACE s_axilite port=Sx  
19     #pragma HLS INTERFACE s_axilite port=Hin  
20     #pragma HLS INTERFACE s_axilite port=Chin  
21     #pragma HLS INTERFACE s_axilite port=Kx  
22     #pragma HLS INTERFACE s_axilite port=mode  
23     #pragma HLS INTERFACE s_axilite port=Sy  
24     #pragma HLS INTERFACE s_axilite port=Ky  
25     #pragma HLS INTERFACE s_axilite port=Win  
26     #pragma HLS INTERFACE s_axilite port=return  
27 }
```

图2-14 直接在代码中添加Directive

!!! hint "小提示 💡"

在图2-14中，每一个制导语句都指定了一个端口的总线类型、主从属性以及允许的最大数据深度等信息。例如，第一行指定了 `feature_out` 将作为AXI总线的从端口，且最大深度为4294967295。

接着，可用同样的方法设置并行优化策略。

!!! info "补充说明 📖"

此处暂不添加优化语句。在后续的实验中，将会再讲解如何优化。

Directive设置完毕，接下来需要对C代码进行综合，以生成RTL电路。

点击Project->Project Settings->Synthesis，点击Top Function右边的Browse，选择 `conv_core` 作为Top Function，点击“OK”按钮，如图2-15所示。

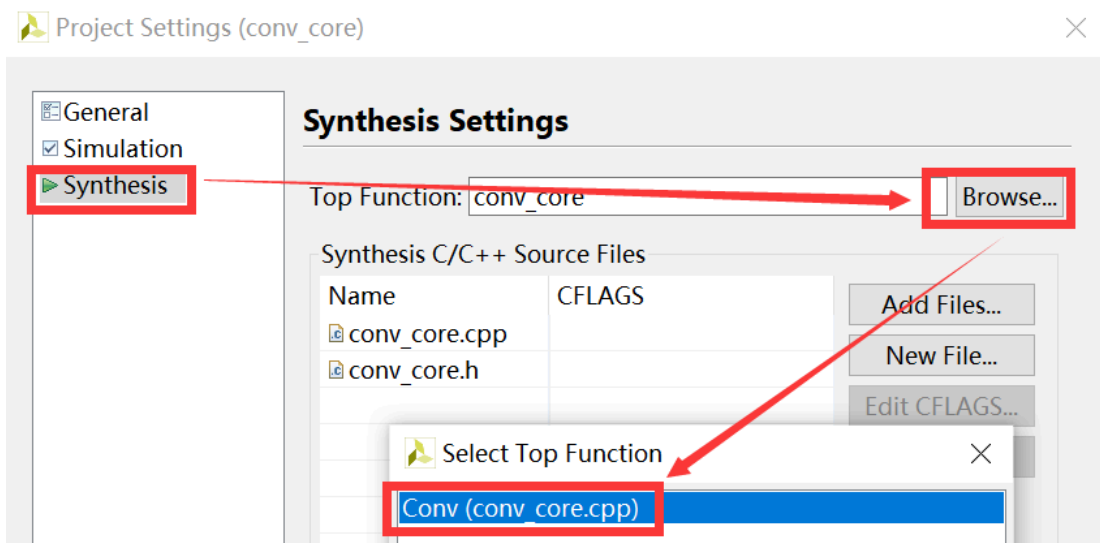


图2-15 选择Top Function

点击菜单栏的Solution->Run C Synthesis->Active Solution，或点击工具栏中形如绿色三角形的综合按钮，如图2-16所示。

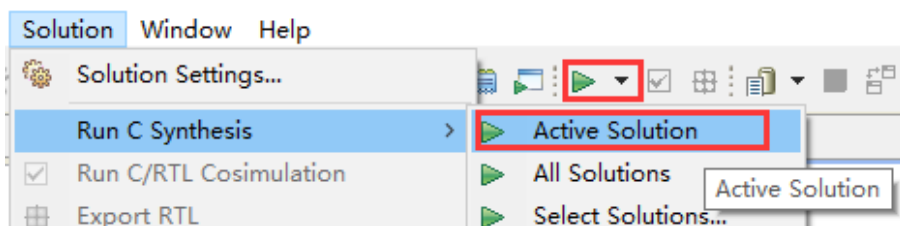


图2-16 点击以进行综合

综合完成后会自动显示综合报告。报告中包含了预估性能、预估资源情况、电路时延、模块接口等信息，如图2-17所示。

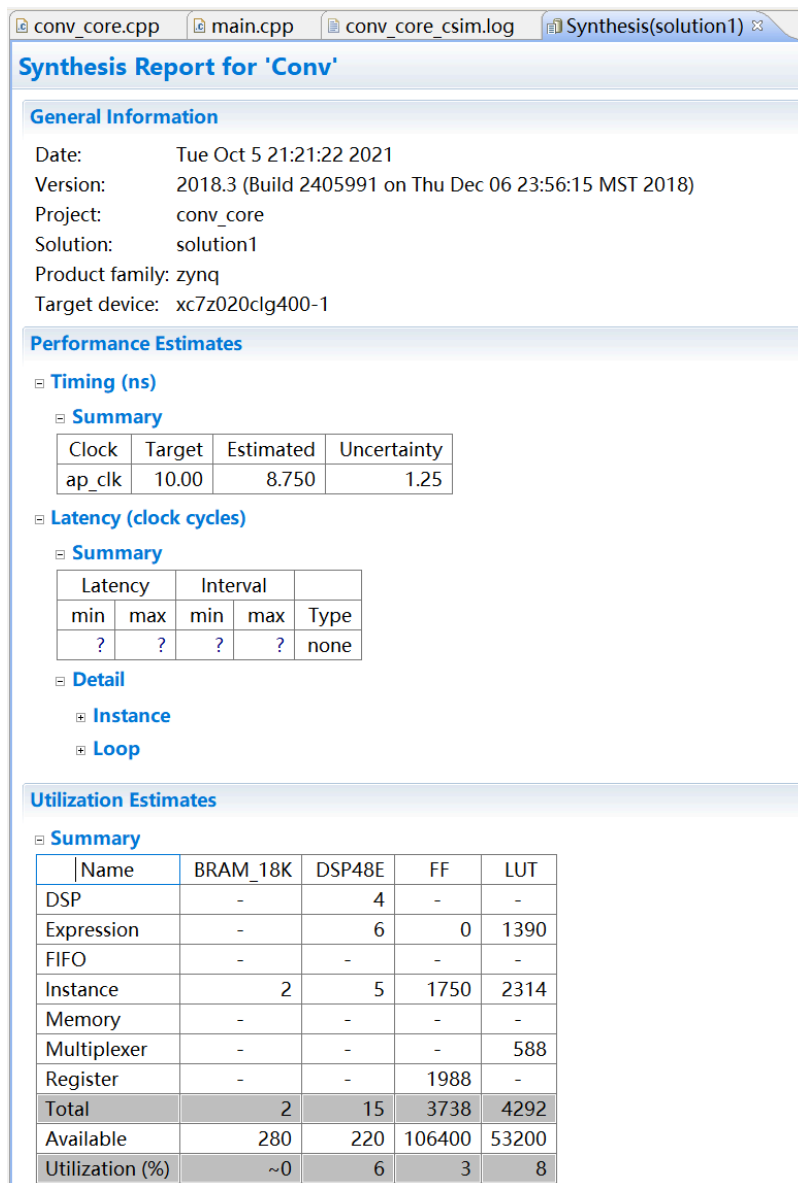


图2-17 综合报告

1.6 C/RTL协同仿真

运行C/RTL Co-Sim时，会调用HDL仿真工具。

点击Solution->Run C/RTL Cosimulation，或点击工具条中形如对号的按钮，如图2-18所示。

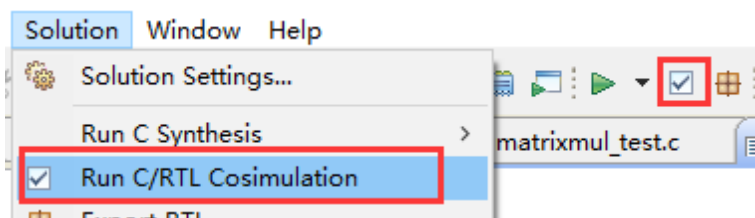


图2-18 点击进行C/RTL联合仿真

在弹出的窗口中选中仿真工具为Vivado Simulator，语言选中Verilog，Dump Trace选择all，如图2-19所示。

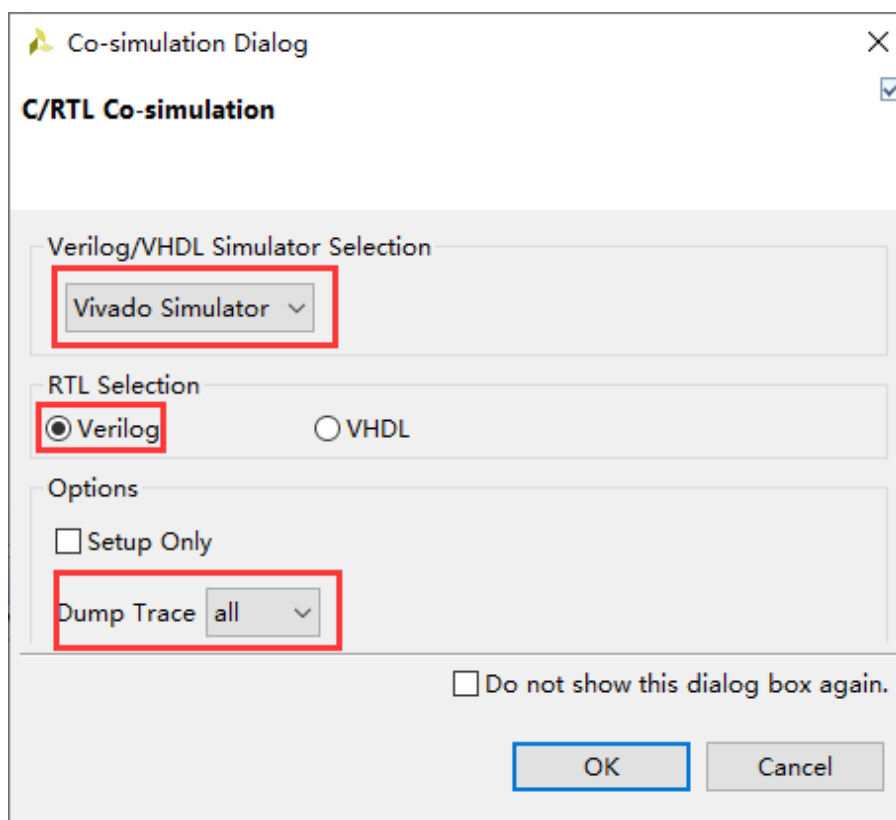


图2-19 联合仿真设置

仿真报告显示了仿真的结果以及时延等信息，如图2-20所示。

Cosimulation Report for 'matrixmul'

Result

RTL	Status	Latency			Interval		
		min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	277	277	277	NA	NA	NA

图2-20 联合仿真报告

还可点击工具条中的波形按钮以查看波形，此时将打开Vivado的仿真波形界面，如图2-21所示。

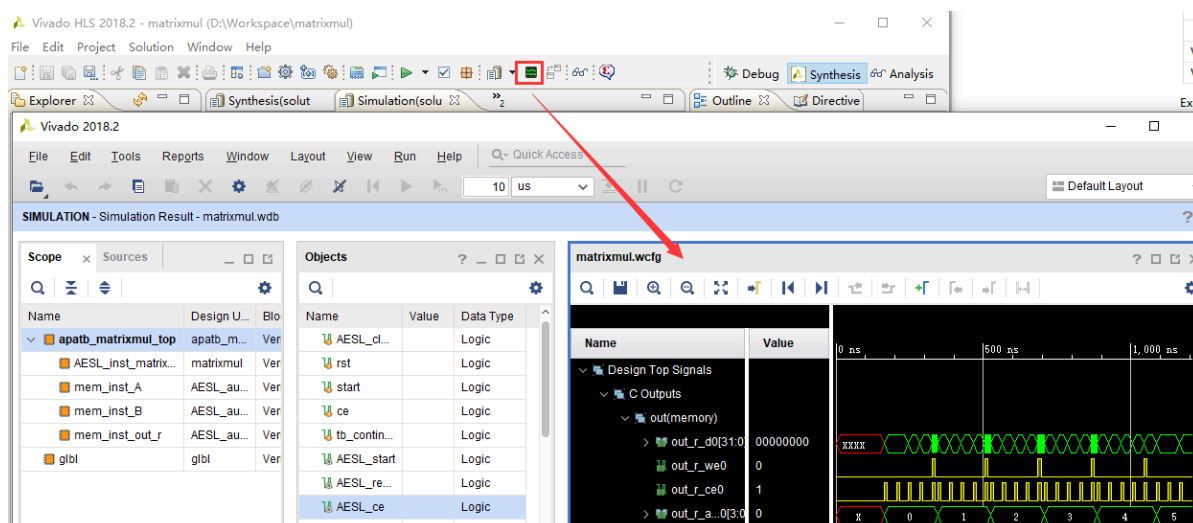


图2-21 查看仿真波形

!!! info "补充说明" 🚩

对于本课程实验，可忽略C-RTL协同仿真的步骤。

1.7 导出RTL并打包成IP核/Export RTL

点击菜单栏的Solution->Export RTL，或点击工具条中形如棕色箱子的按钮，如图2-22所示。

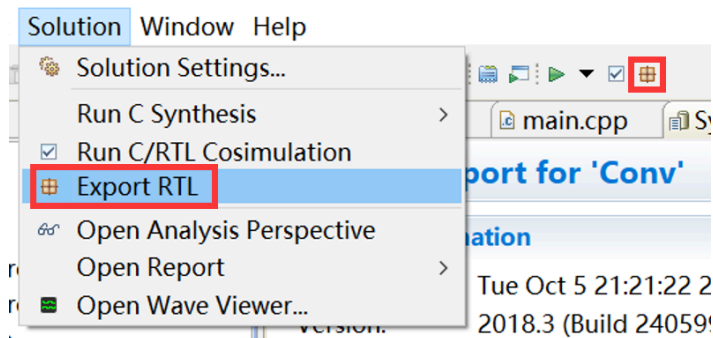


图2-22 点击以导出RTL

随后将弹出导出IP核的设置窗口，保持默认即可，点击“OK”按钮。

IP核导出成功后，可在主界面/综合界面左侧的工程目录Solution1->impl->ip中找到.zip格式的IP核，如图2-23所示。

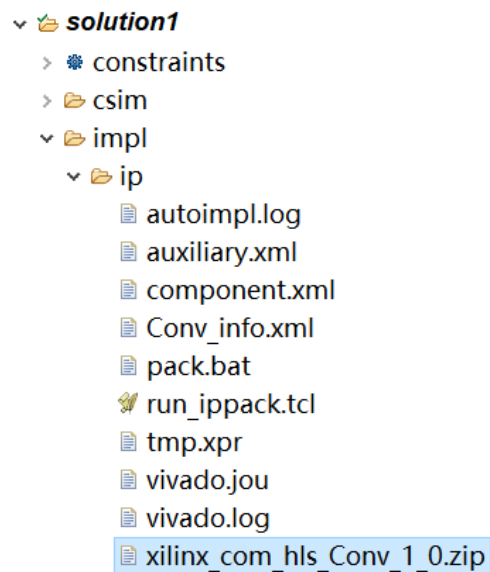


图2-23 查看导出的IP核

2. 创建Block Design

2.1 创建Vivado项目

在桌面双击Vivado快捷方式，或在开始菜单->Xilinx Design Tools->Vivado打开Vivado工具。点击“Quick Start”下方的“Create Project”，或点击菜单栏中的File->Project->New。然后，填写项目名称，并选择项目所在的路径，如图2-27所示。

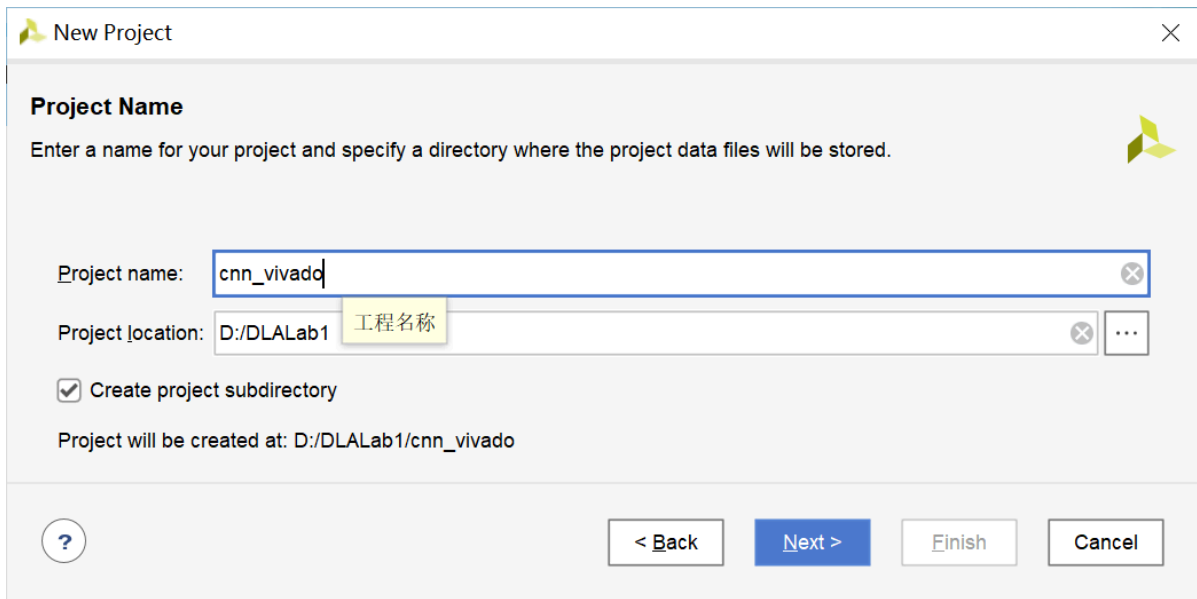


图2-27 创建Vivado项目

点击“Next”按钮，勾选“Do not specify sources at this time”，然后继续点击Next按钮。然后在搜索框中输入 xc7z020clg400-1，如图2-28所示。

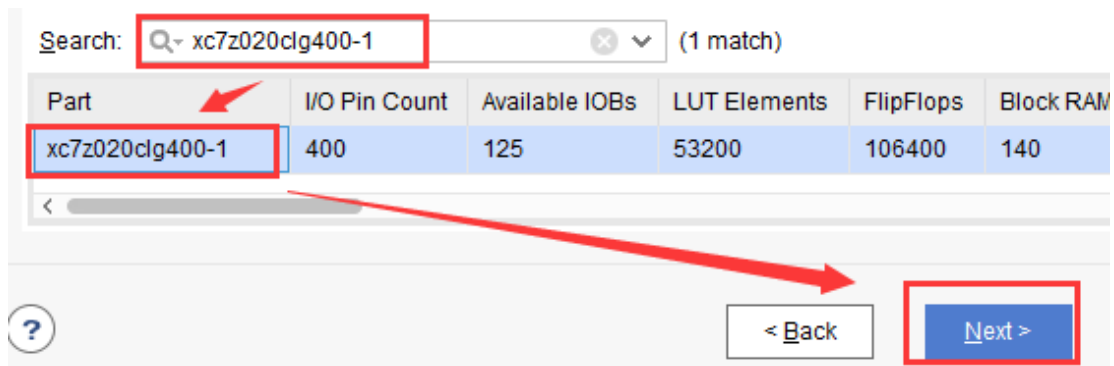


图2-28 输入PYNQ-Z2的器件型号

一直点击“Next”按钮，最后点击“Finish”按钮，完成Vivado工程的创建。

2.2 添加IP核

点击左侧Flow Navigator->Project Manager->Settings，在IP->Repository中直接添加上文的HLS工程路径，如图2-29所示。

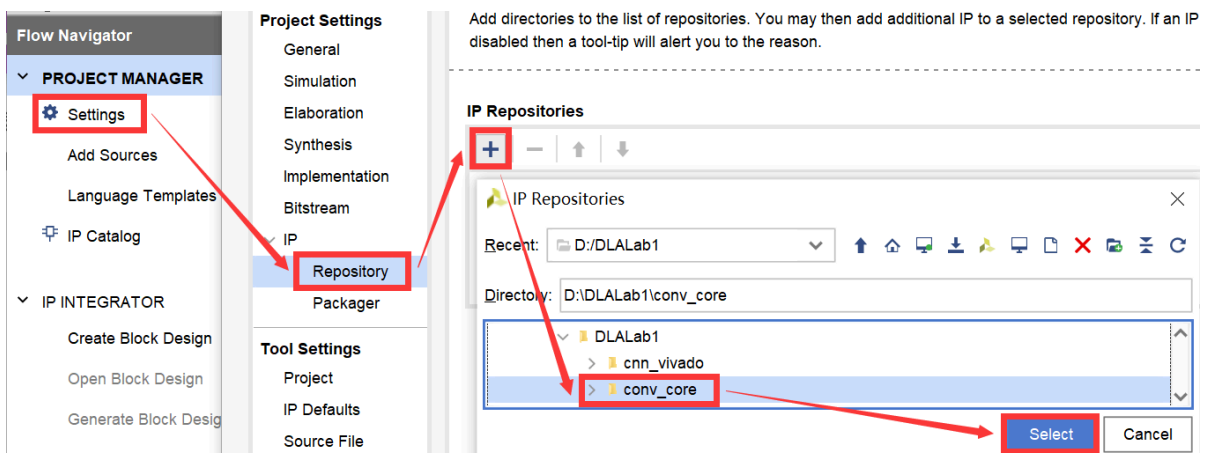


图2-29 添加IP核

添加完成后，将弹出确认窗口。在该窗口中，可查看IP核，如图2-30所示。

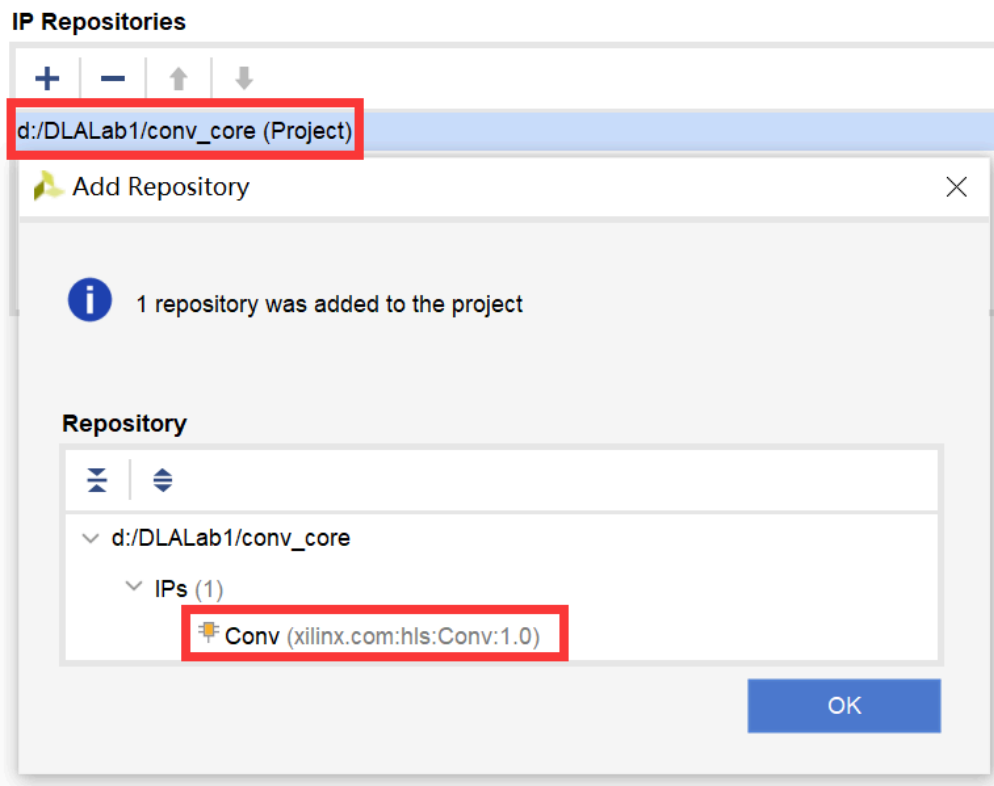


图2-30 添加成功后的窗口

点击“OK”按钮，回到Vivado主界面。

将实验包中的 pool_core_ip.zip 池化IP核解压后，用类似的方法添加入工程。

2.3 搭建Block Design电路

点击IP Integrator->Create Block Design，并点击“OK”按钮以新建Block Design，如图2-31所示。

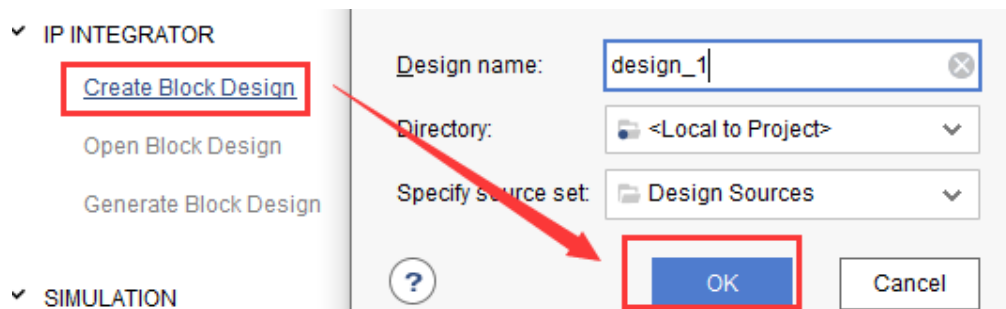


图2-31 新建Block Design

在右上方工作区中的Diagram内，点击加号添加ZYNQ IP核，然后点击Run Block Automation，如图2-32所示。

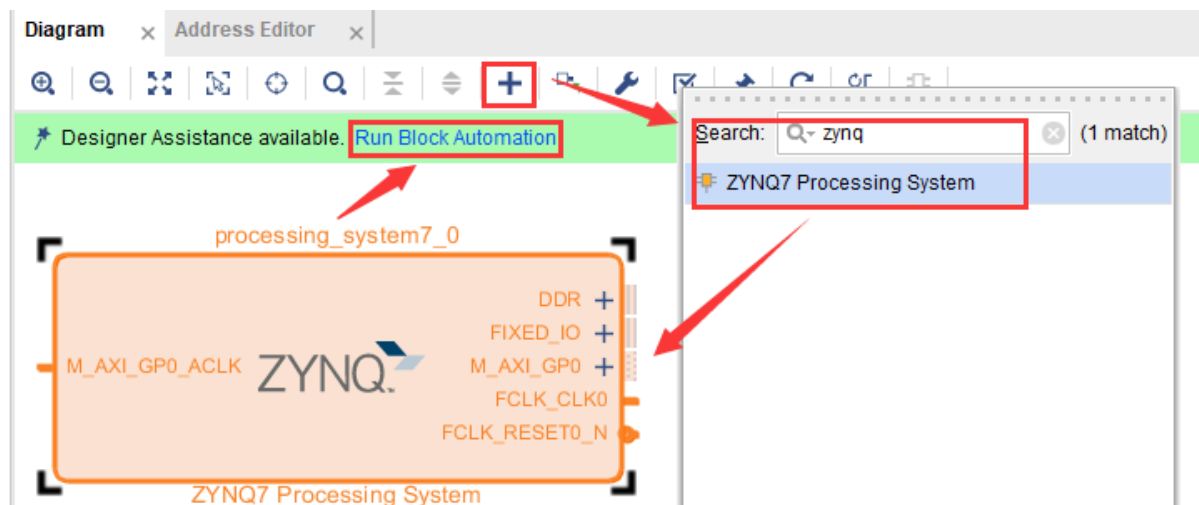


图2-32 添加ZYNQ IP核

随后将弹出一个窗口，保持默认设置，点击“OK”按钮即可。

接下来，需要为ZYNQ的PS端添加相应的AXI总线接口，以连接卷积IP核和池化IP核。

双击ZYNQ IP核，为ZYNQ添加S_AXI_HP0、S_AXI_HP1两个高性能AXI从端的总线接口，如图2-33所示。

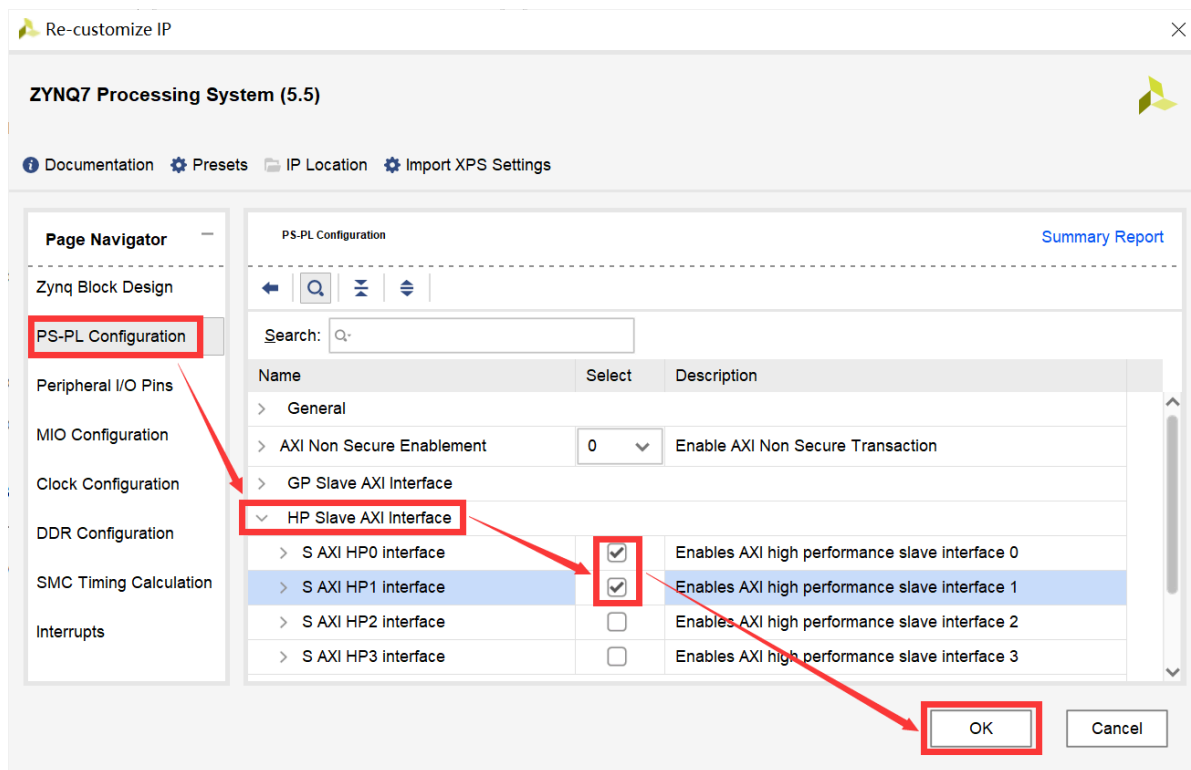


图2-33 为ZYNQ添加AXI总线接口

点击Diagram中的加号，添加矩阵乘法IP核，如图2-34所示。

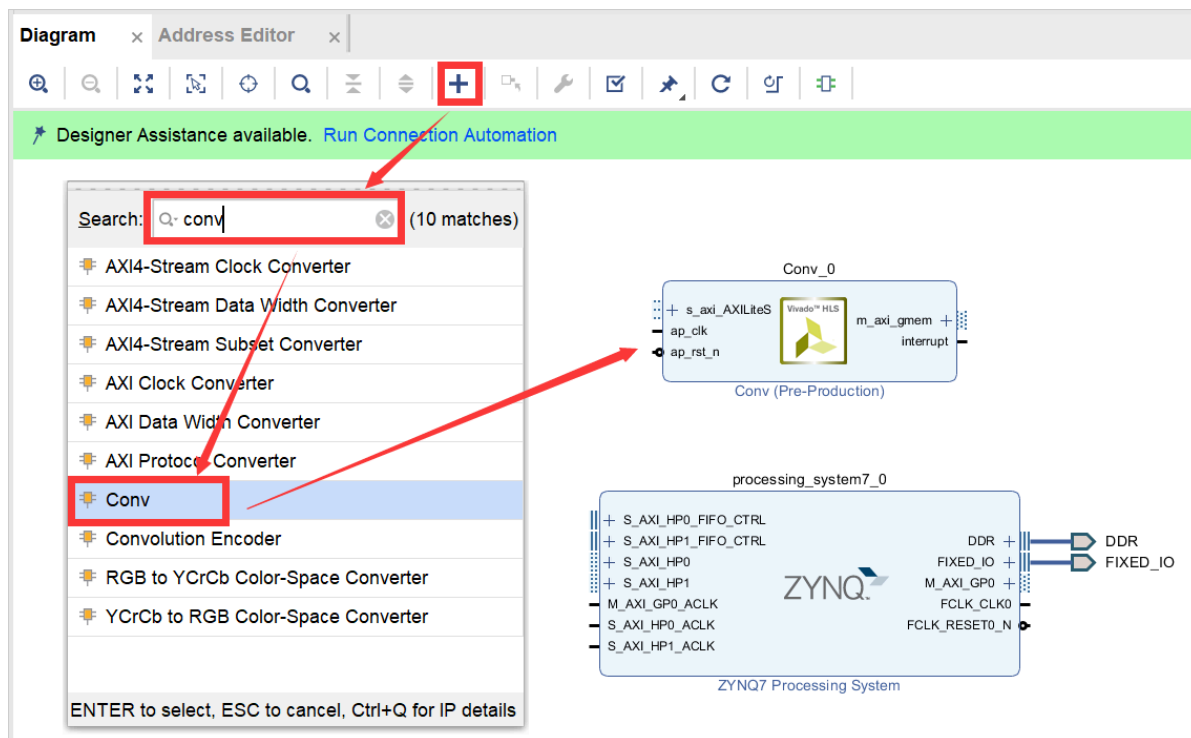


图2-34 添加卷积IP核

点击Run Connection Automation，并勾选除S_AXI_HP1以外的所有复选框，随后点击“OK”按钮，如图2-35所示。

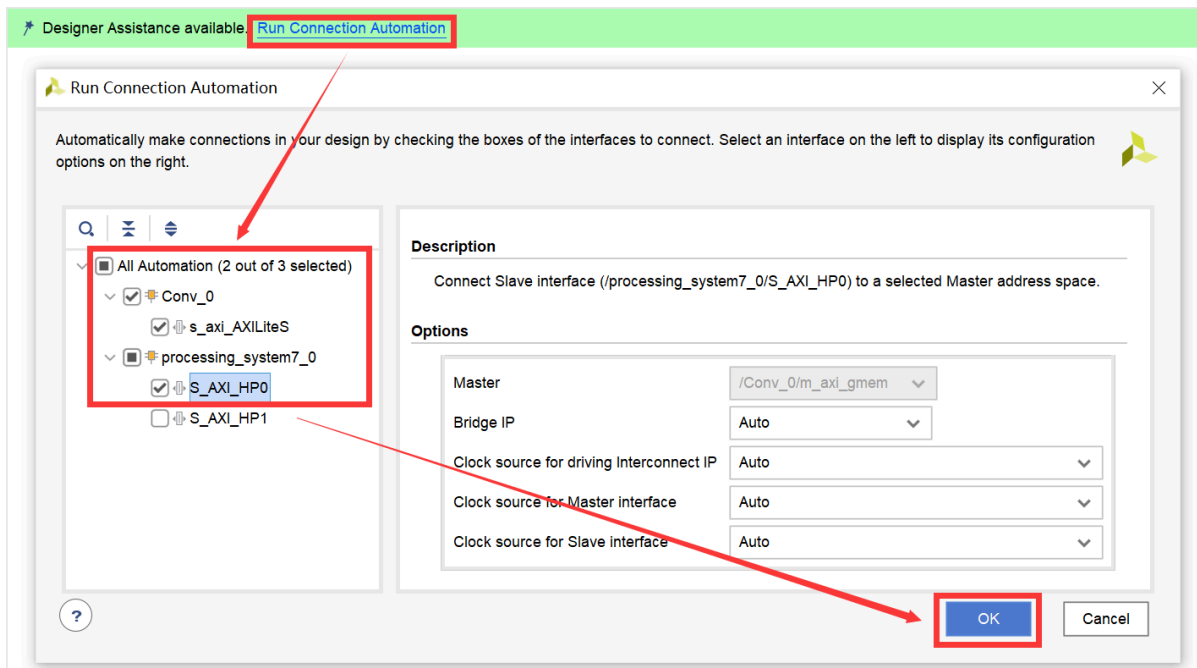


图2-35 自动连接ZYNQ IP核与卷积IP核

通常自动布线得到的电路模块图都较不美观，不利于观察和分析电路。可点击圆形箭头进行重新布线，如图2-36所示。

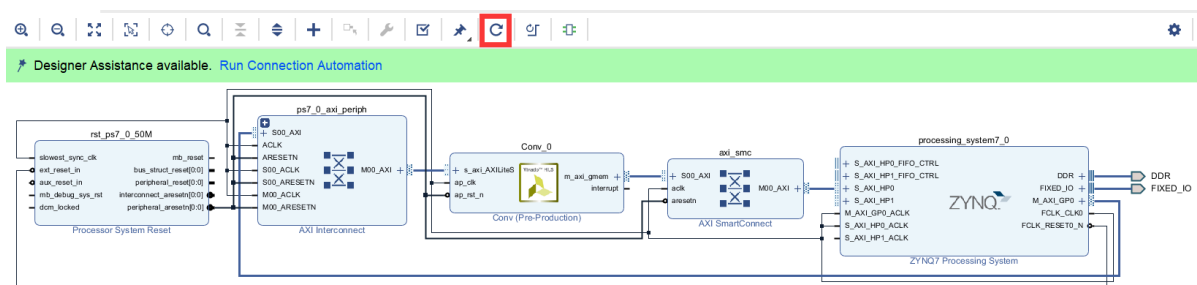


图2-36 重新布线

同样的，使用如图2-34所示的方法，在搜索框中输入 `pool`，添加池化IP核。

然后，双击打开 **AXI Interconnect** IP核的配置窗口，将其Master接口的数量从1改成2，如图2-37所示。

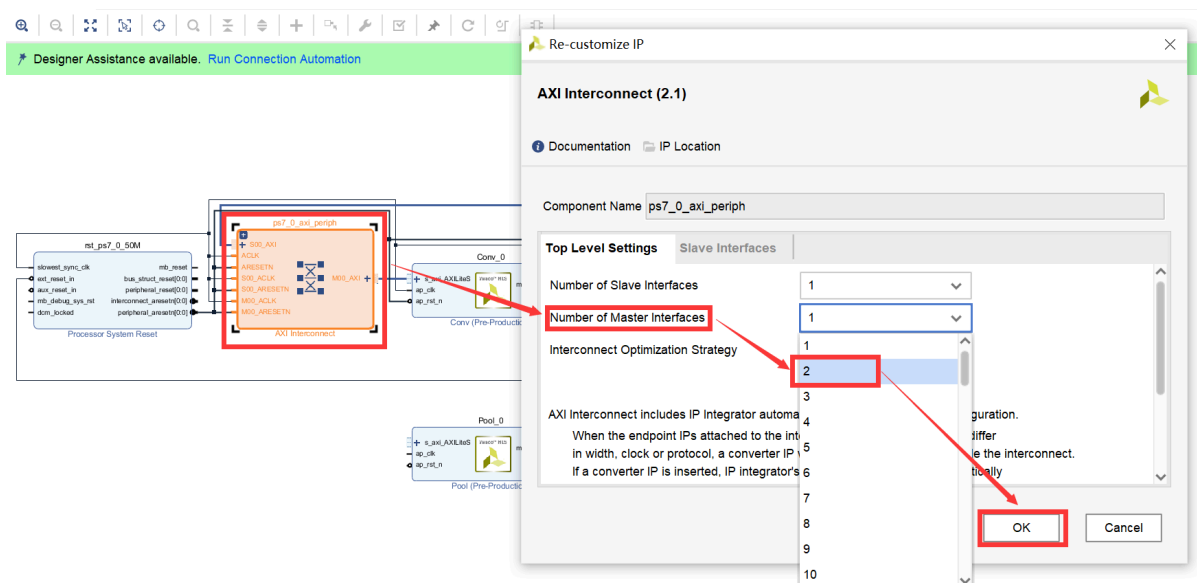


图2-37 增加AXI互联IP核的Master接口

使用Ctrl+C和Ctrl+V快捷键，复制一个AXI SmartConnect IP核，并将AXI Interconnect IP核、池化IP核、新复制的AXI SmartConnect IP核与ZYNQ IP核的S_AXI_HP1接口进行连接，如图2-38所示。

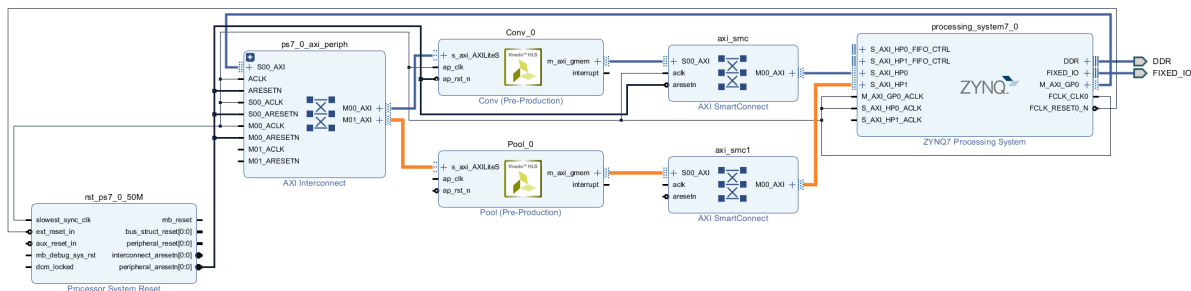


图2-38 连接池化IP核

再次点击Run Connection Automation，并勾选所有复选框，完成剩余的电路连接。

!!! warning "注意"

自动连接完毕后，如果发现axi_smc1 IP核的复位信号悬空，则需要手动将其与axi_smc IP核的复位信号连接起来。

再次点击如图2-36所示的圆形箭头以重新布局，最终得到如图2-39所示的电路模块图。

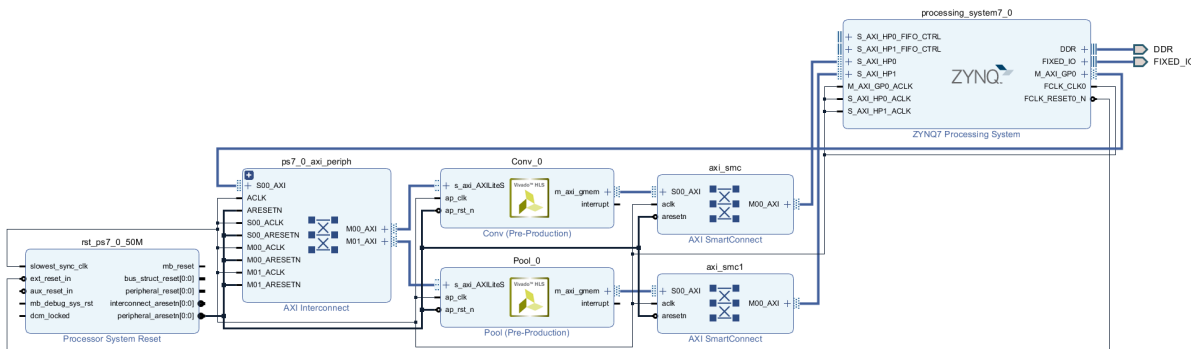


图2-39 本实验的CNN加速器电路模块图

所有连接完成后，点击进入Diagram右侧的Address Editor标签页，并点击Auto Assign Address按钮，从而为IP核分配地址空间，如图2-40所示。

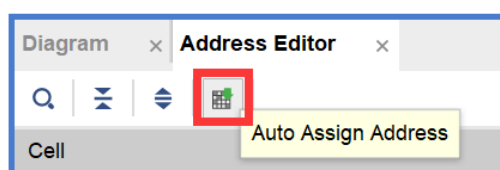


图2-40 分配地址空间

此时，将弹出对话框提示地址已经分配成功，点击OK按钮即可。分配完成后，Auto Assign Address按钮将变成灰色。

!!! info "补充说明"

若打开Address Editor标签页时，Auto Assign Address按钮已经是灰色，说明地址已自动分配完成。

点击回到Diagram标签页，点击Validate Design按钮，如图2-41所示。

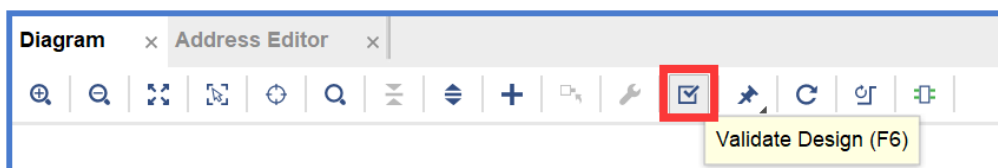


图2-41 检查Block Design是否有错误

随后，Vivado将自动检查Block Design的正确性。若Block Design正确无误，将弹出对话框提示 **Validation successful**。否则将显示 **Critical Warning** 或者 **Error** 信息，此时需要根据这些信息修正设计中的问题或错误。

Validation成功后，点击菜单栏的保存按钮，或用快捷键 **Ctrl+S** 保存Block Design。

2.4 生成比特流并导出Overlay

在源文件管理窗口中的Block Design文件上右键，点击 **Create HDL Wrapper**，如图2-42所示。

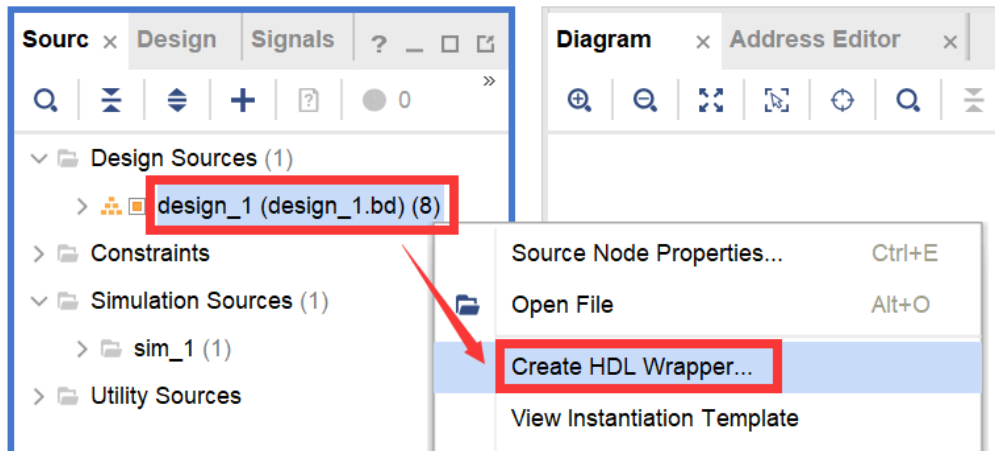


图2-42 为Block Design添加HDL顶层文件

随后将弹出“Create HDL Wrapper”窗口，保持默认设置，然后点击“OK”按钮即可。

接下来，需要生成整个硬件设计的比特流文件。

点击菜单栏中的绿色箭头，生成比特流文件，如图2-43所示。如果希望加快比特流的生成速率，可以将“Number of jobs”设置为最大值。

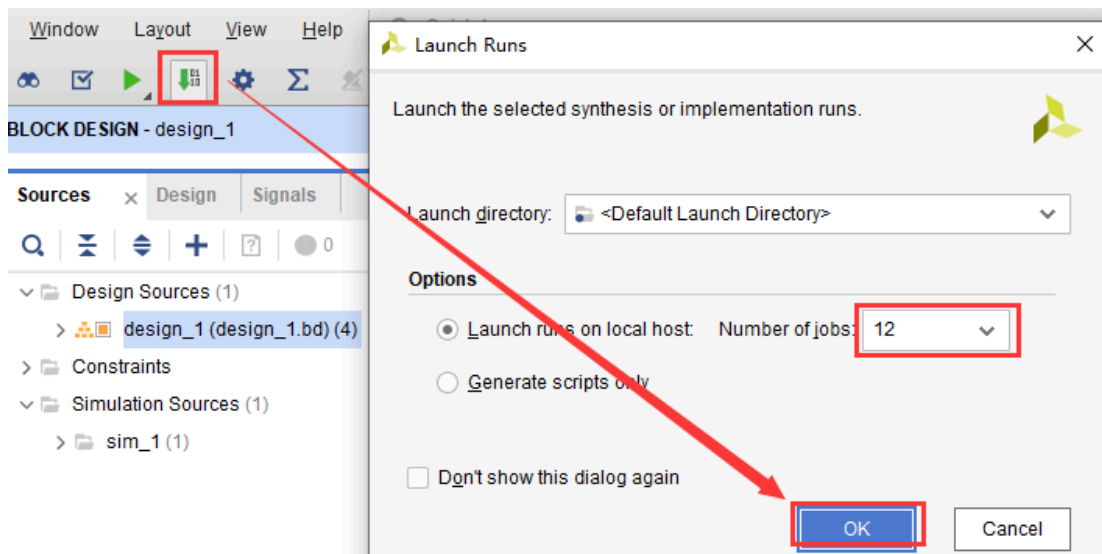


图2-43 生成比特流

耐心等待比特流生成完毕，然后点击菜单栏的File->Export->Export Block Design，将Block Design导出到 **D:/DLALab1/cnn_vivado** 目录，如图2-44所示。

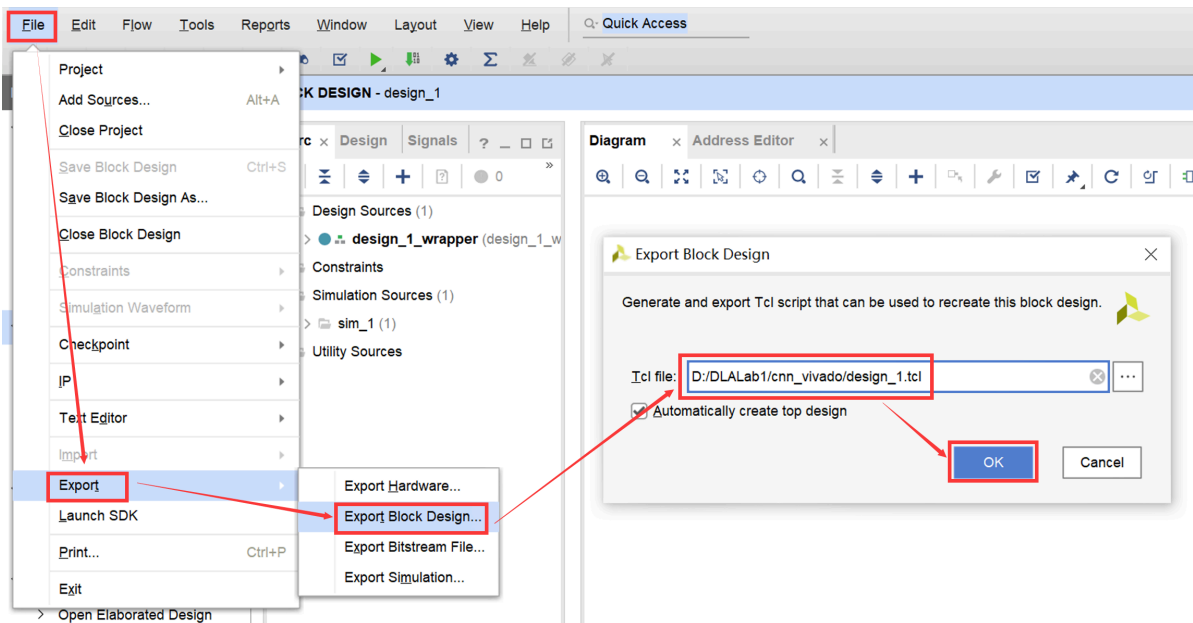


图2-44 导出Block Design的.tcl文件

类似地，点击菜单栏的File->Export->Export Bitstream File，将生成的比特流文件导出到 `D:/DLALab1/cnn_vivado` 目录。

将上面导出的.tcl和.bit文件分别重命名为 `mnist_cnn.tcl` 和 `mnist_cnn.bit`，并将其拷贝到实验包的 `mnist_cnn.zip` 压缩包中，如图2-45所示。

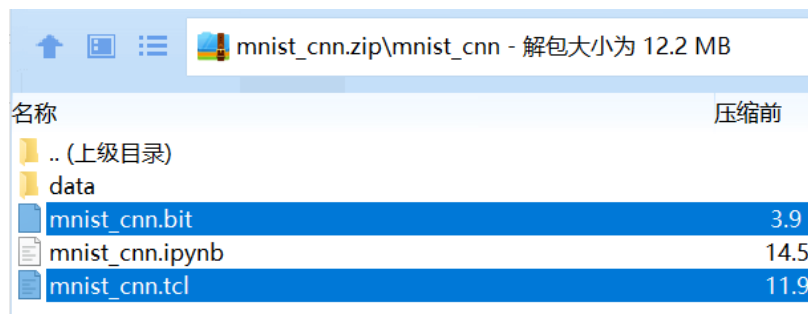


图2-45 拷贝Overlay到压缩包

!!! note "记笔记 📌"

在PYNQ开发中，`.tcl` 文件和 `.bit` 文件在一起合称Overlay。

3. 上板测试

3.1 上传测试包

本课程实验以远程访问的形式使用PYNQ-Z2平台——校园网内的PC可通过Web浏览器访问 `10.249.12.21:<port>`，以密码 `xilinx` 登录Jupyter Notebook。

点击Jupyter网页右侧的 `Upload` 按钮，上传 `mnist_cnn.zip` 压缩包，如图2-46所示。

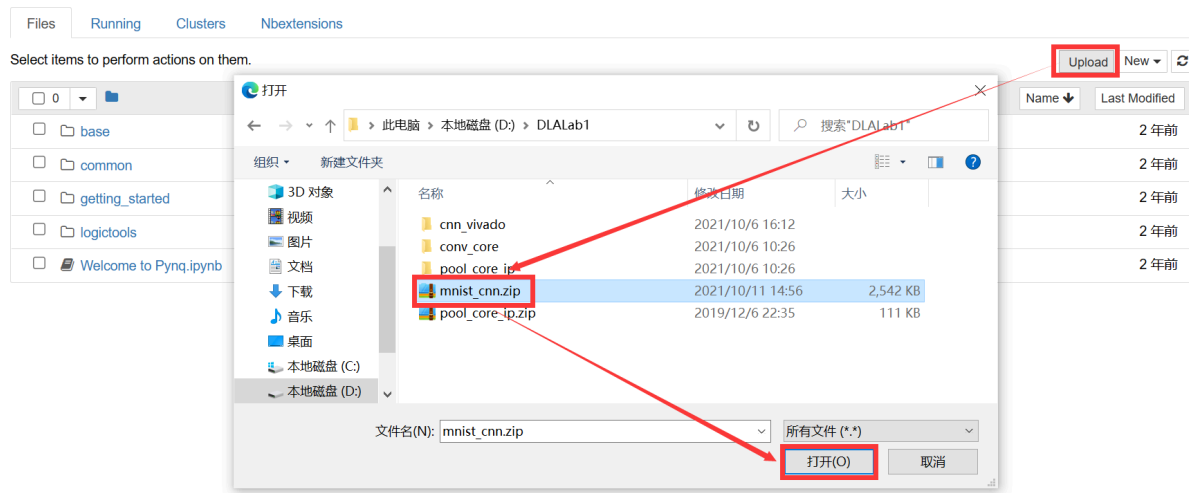


图2-46 上传测试包

点击文件右侧蓝色的“Upload”按钮，开始上传，如图2-47所示。

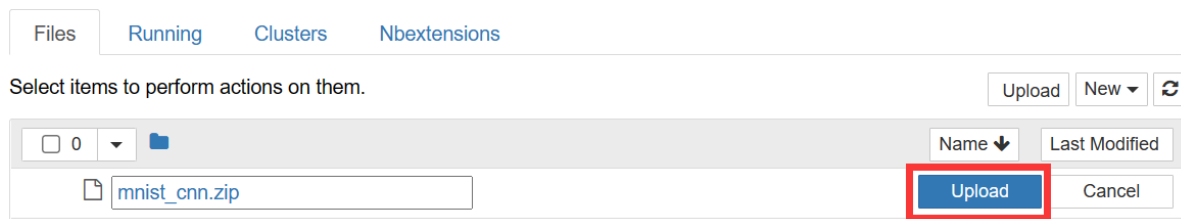


图2-47 上传包含测试文件的.zip压缩包

!!! info "莫心急 🤖"

有时网络不稳定可能导致上传失败。若上传失败，请多尝试几次。

展开 Upload 按钮右侧的 New，并点击 Terminal 以新建一个命令行终端，如图2-48所示。

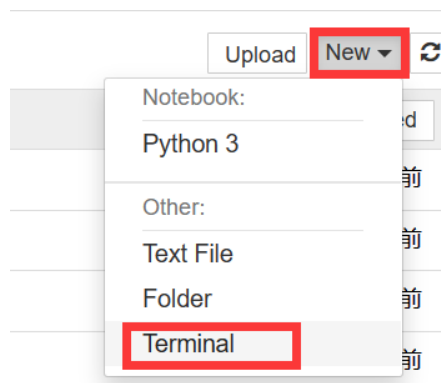


图2-48 新建Linux终端

在终端中依次输入以下命令，以解压刚才上传的 mnist_cnn.zip 压缩包。

```
$> cd jupyter_notebooks/  
$> unzip mnist_cnn.zip
```

!!! note "记笔记 📝"

在终端中，若遇到需要输入长串字符的情形，为避免误输入，请多使用 Tab 键。

  使用方法：

     先手动输入前几个字符，再按下`Tab`键，此时将自动补全。若没有补全，说明当前目录至少存在2个文件，

   且这些文件的文件名的前几个字符均相同。此时，再手动多输入几个字符，再按`Tab`键即可。

3.2 运行测试

解压完毕后，回到Jupyter，点击进入 `mnist_cnn` 文件夹。

打开 `mnist_cnn.ipynb`，并点击菜单栏的Cell->Run All开始运行测试。如果所设计的IP核功能正确，则程序应当能够正确识别出 `mnist_cnn/data/` 目录下名为 `1.jpg` 的手写数字图片，将输出如图2-49所示的结果。

5. CNN硬件推导

```
In [5]: image1 = cv2.imread("./data/1.jpg", cv2.IMREAD_GRAYSCALE).astype(np.float32)
for r in range(IN_HEIGHT1):
    for c in range(IN_WIDTH1):
        for ch in range(IN_CH1):
            image[r][c][ch] = (255 - image1[r][c])/255

print("Finish reading image.")

# Conv1
hwConv(conv_ip, KERNEL_W1, KERNEL_W1, STRIDE1, STRIDE1, 1, RELU_EN1, image, W_conv1, b_conv1, h_conv1)
hwPool(pool_ip, KERNEL_W11, KERNEL_W11, MODE11, h_conv1, h_pool1)
# Conv2
hwConv(conv_ip, KERNEL_W2, KERNEL_W2, STRIDE2, STRIDE2, 1, RELU_EN2, h_pool1, W_conv2, b_conv2, h_conv2)
hwPool(pool_ip, KERNEL_W21, KERNEL_W21, MODE21, h_conv2, h_pool2)
# FC1
hwConv(conv_ip, KERNEL_W3, KERNEL_W3, STRIDE3, STRIDE3, 0, RELU_EN3, h_pool2, W_fc1, b_fc1, h_fc1)
# FC2
hwConv(conv_ip, KERNEL_W4, KERNEL_W4, STRIDE4, STRIDE4, 0, RELU_EN4, h_fc1, W_fc2, b_fc2, h_fc2)

MAX = h_fc2[0][0][0]
result = 0
for ch in range(1, OUT_CH4):
    if (h_fc2[0][0][ch] > MAX):
        MAX = h_fc2[0][0][ch]
        result = ch

print("The image was recognized as " + str(result))

Finish reading image.
The image was recognized as 2
```

图2-49 `mnist_cnn.ipynb` 的运行结果

!!! info "补充说明 😊"

感兴趣的同学可用Windows下的“画图”软件擦除 `1.jpg`，并绘制其他数字；然后将图片上传回 `mnist_cnn/data` 文件夹，重新运行图2-49中的“5.CNN硬件推导”。修改 `1.jpg` 的时候注意不要更改图片的尺寸。