

# 使用 FPGA 实现简单的四则运算

## 环境要求

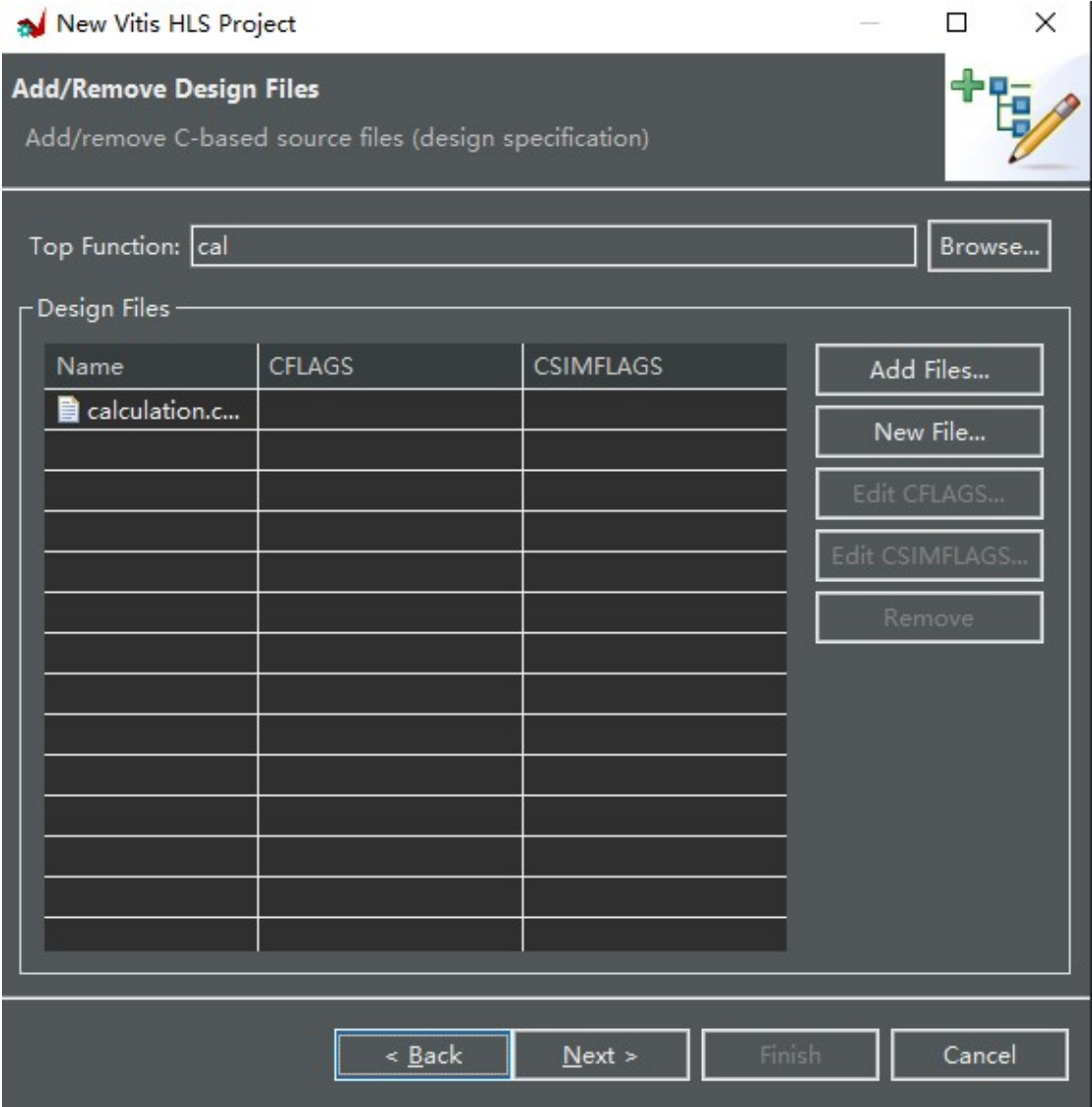
- 1、PYNQ-Z2 远程实验室服务或物理板卡
- 2、Vitis HLS
- 3、Vivado

## 实验步骤

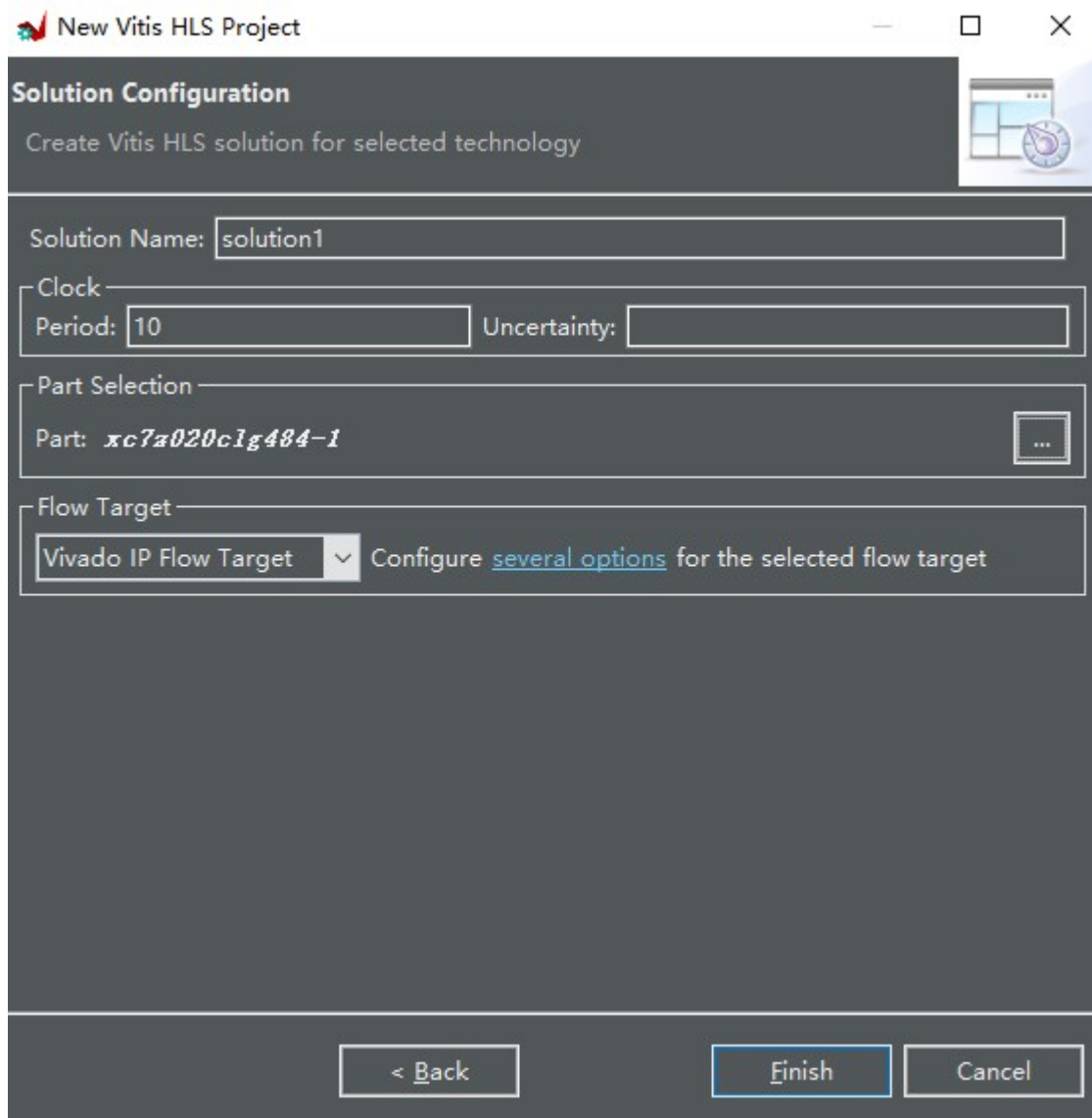
### 一、在 Vitis\_HLS 中设计 CALCULATE IP

#### 1.1 创建一个新项目

- 1. 打开 Vitis HLS 软件，点击 Create Project，创建一个新的项目
- 2. 在 Project name 输入项目名 calculation\_hls\_prj，点击 Browse 选择一个合适的目录位置，点击 Next
- 3. 点击 Add Files...，将 src 目录下的 calculation.cpp 添加到项目中
- 4. 给顶层函数设置为 cal



5. 下面进入到 Solution Configuration 界面，保持其他选项不变，在 Part Selection 栏最右侧点击 .... 字样的按钮，在 Search 栏的搜索框中输入 xc7z020clg484-1，即 PYNQ-Z2 板卡所使用的器件型号。



New Vitis HLS Project

Solution Configuration

Create Vitis HLS solution for selected technology

Solution Name:

Clock

Period:  Uncertainty:

Part Selection

Part: **xc7z020clg484-1**

Flow Target

< Back Finish Cancel

6. 点击 Finish，完成项目的创建

## 1.2 C-Synthesis

1. 下面，我们对设计进行 C 综合。在左下方的 Flow Navigator 中点击 Run C Synthesis，在弹出的 C Synthesis - Active Solution 窗口中保持各选项不变，点击 OK 开始综合
2. 等待数秒，Vitis HLS 会将其综合的各步骤的信息打印在 Console 中
3. 综合完成后，会弹出 Synthesis Summary(solution1) 窗口，我们可以在此看到 Vitis HLS 给出的时钟频率信息、时钟周期数和资源消耗等
4. 完成之后我们可以在左侧菜单里选择 solution1->impl->misc->drivers->mul\_v1\_0->src->xmul\_hw.h

```
calculation.cpp  Synthesis Summary(solution1)  xmul_hw.h x
15 // bit 31~0 - d[31:0] (Read/Write)
16 // 0x1c : reserved
17 // 0x20 : Data signal of c
18 // bit 31~0 - c[31:0] (Read)
19 // 0x24 : Control signal of c
20 // bit 0 - c_ap_vld (Read/COR)
21 // others - reserved
22 // 0x30 : Data signal of d
23 // bit 31~0 - d[31:0] (Read)
24 // 0x34 : Control signal of d
25 // bit 0 - d_ap_vld (Read/COR)
26 // others - reserved
27 // 0x40 : Data signal of e
28 // bit 31~0 - e[31:0] (Read)
29 // 0x44 : Control signal of e
30 // bit 0 - e_ap_vld (Read/COR)
31 // others - reserved
32 // 0x50 : Data signal of f
33 // bit 31~0 - f[31:0] (Read)
34 // 0x54 : Control signal of f
35 // bit 0 - f_ap_vld (Read/COR)
36 // others - reserved
37 // (SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake)
38
39 #define XMUL_CONTROL_ADDR_A_DATA 0x10
40 #define XMUL_CONTROL_BITS_A_DATA 32
41 #define XMUL_CONTROL_ADDR_B_DATA 0x18
42 #define XMUL_CONTROL_BITS_B_DATA 32
43 #define XMUL_CONTROL_ADDR_C_DATA 0x20
44 #define XMUL_CONTROL_BITS_C_DATA 32
45 #define XMUL_CONTROL_ADDR_C_CTRL 0x24
46 #define XMUL_CONTROL_ADDR_D_DATA 0x30
47 #define XMUL_CONTROL_BITS_D_DATA 32
48 #define XMUL_CONTROL_ADDR_D_CTRL 0x34
49 #define XMUL_CONTROL_ADDR_E_DATA 0x40
50 #define XMUL_CONTROL_BITS_E_DATA 32
51 #define XMUL_CONTROL_ADDR_E_CTRL 0x44
52 #define XMUL_CONTROL_ADDR_F_DATA 0x50
53 #define XMUL_CONTROL_BITS_F_DATA 32
54 #define XMUL_CONTROL_ADDR_F_CTRL 0x54
55
```

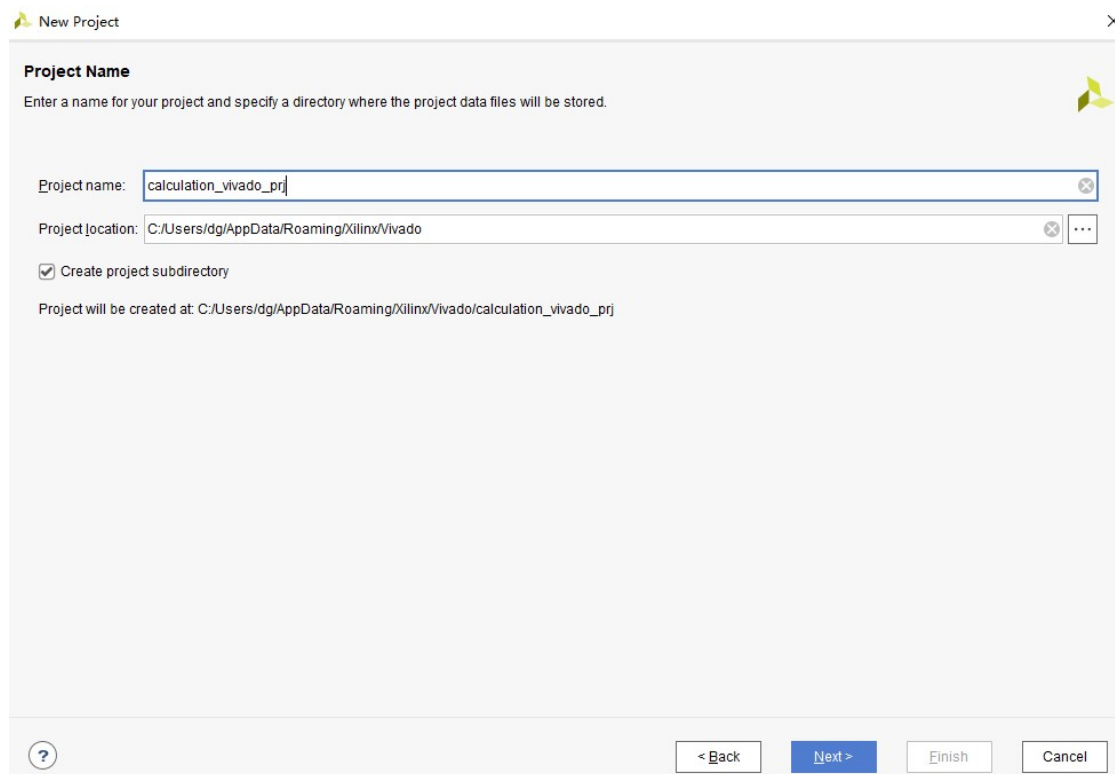
### 1.3 导出 RTL

1. 下面，我们对设计进行 RTL 导出。在左下方的 Flow Navigator 中点击 Export RTL，在弹出的 Export RTL 窗口中保持各选项不变，点击 OK 开始 RTL 的导出
2. 等待约半分钟，Console 中打印 Finished Export RTL/Implementation. 表明 RTL 设计已经导出完成
3. 为了后续使用的便利，将 export.zip 文件解压到其所在目录下，即得到一 export 文件夹
4. 至此，我们已经完成了 FIR 加速核的设计与导出

## 二、在 Vivado 中进行 IP 集成

### 2.1 创建一个新 Vivado 项目

1. 打开 Vivado 软件，点击 Create Project，创建一个新的项目，点击 Next
2. 在 Project name 输入项目名 calculation\_vivado\_prj，点击右侧的 ... 按钮选择一个合适的目录位置，点击 Next



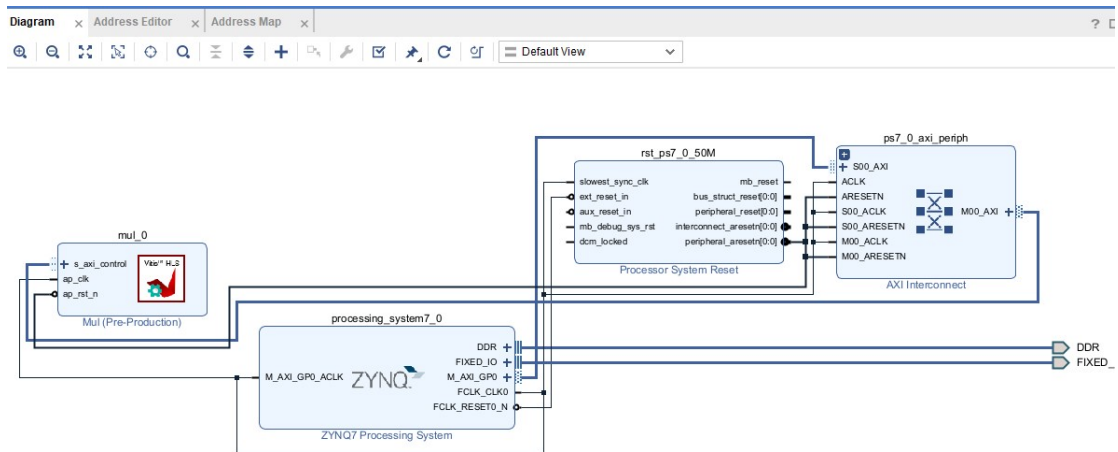
3. 进入 Project Type 界面，勾选上 Do not specify sources at this time，再点击 Next
4. 进入 Default Part 界面，在 Search 栏中搜索 xc7z020clg484-1，将其选中，再点击 Next
5. 点击 Finish 完成项目创建

## 2.2 导入 IP

1. 我们需要首先将从 Vitis HLS 中导出的 IP 导入到 Vivado 中，点击左侧窗口 Flow Navigator 中的 Settings 选项，弹出 Settings 窗口
2. 将左侧的 Project Settings 中展开 IP 栏目，选中 Repository 项，点击右侧面板中的 + 按钮，在弹出窗口中选择刚才解压出来 IP，即 \fir\_hls\_prj\solution1\impl\export，再点击 Select
3. 可以看到对应的 IP 已经被成功添加到了工程中，在两个窗口中依次单击 OK 来关闭这些窗口

## 2.3 创建 Block Design

1. 下面我们创建一个 Block Design，利用 Vivado 的 IP 集成功能来构建完整系统。在左侧的 Flow Navigator 中点击 IP INTEGRATOR > Create Block Design，在弹出的 Create Block Design 窗口中保持各选项不变，设计名称使用默认的 design\_1，点击 OK 创建 Block Design
2. 在出现的 Diagram 窗口中点击上方的 + 按钮，会弹出一个搜索框，在输入栏中键入 zynq，双击备选项中出现的 ZYNQ7 Processing System，即可将该 IP 添加到设计中
3. 在窗口上方会出现蓝色下划线提示 Run Block Automation，单击该区域弹出对应窗口，我们保持默认设置不变，直接点击 OK
4. 点击 Diagram 窗口上方的 + 按钮，搜索 fir，可以看到我们刚才导入的 IP 已经可以使用了，双击 Mul 以将其添加到设计中
5. 下面我们对设计进行自动连线。点击窗口上方的蓝色下划线提示 Run Connection Automation，弹出对应窗口，将左侧 All Automation 选项勾选上，再点击 OK
6. 系统将根据对应接口自动进行连线，我们可以得到如下图的设计



7. 在 Diagram 上侧的工具栏中点击勾形图标 Validate Design，对设计进行验证
8. 在左侧的 Source > Design Sources > design\_1 选项上右键，选择 Create HDL Wrapper，在弹出的窗口中保持选项不变并点击 OK，完成后可以看到在 design\_1.bd 上层嵌套了一层 design\_1\_wrapper.v 文件

## 2.4 综合与生成比特流

1. 在左侧的 Flow Navigator 中选择 Run Synthesis，在弹出的窗口中保持选择不变并选择 OK
2. 综合完成后，会弹出 Synthesis Completed 窗口，在 Next 栏中保持默认的 Run Implementation 选项，并点击 OK，如果出现新弹窗，同样保持默认选项并点击 OK 即可
3. Implementation 结束后，会弹出 Implementation Completed 窗口，在 Next 栏中选择 Generate Bitstream 选项，并点击 OK，如果出现新弹窗，同样保持默认选项并点击 OK 即可
4. 比特流生成后，会弹出 Bitstream Generation Completed 窗口，我们直接点击 Cancel 即可
5. 至此，我们已经完成了硬件部分的设计与导出

## 三、构建 PYNQ 设计

### 3.1 提取 bit 与 hwh 文件

### 3.2 访问 Jupyter

1. 请先完成 PYNQ 远程实验室的账号注册与 Jupyter 访问
2. 登录 Jupyter 界面，点击界面右上方的 upload 按钮，将以下文件上传到开发板上
3. 部署与运行 Overlay 运行结果如下

```
In [44]: # 用FPGA实现简单的四则运算 加减乘整除
# 作者: 哈尔滨工业大学 (深圳)陈舒扬 厦门大学 黄婧
```

```
In [45]: from pynq import Overlay
overlay = Overlay("./calculation.bit")
cal= overlay.mul_0
```

```
In [46]: cal.write(0x10,8)
cal.write(0x18,3)
cal.read(0x20)
```

```
Out[46]: 11
```

```
In [47]: cal.read(0x30)
```

```
Out[47]: 5
```

```
In [48]: cal.read(0x40)
```

```
Out[48]: 24
```

```
In [49]: cal.read(0x50)
```

```
Out[49]: 2
```

```
In [50]: # 用FPGA实现简单的四则运算 加减乘整除
# 作者: 哈尔滨工业大学 (深圳)陈舒扬 厦门大学 黄婧
```

```
In [51]: from pynq import Overlay
overlay = Overlay("./calculation.bit")
cal= overlay.mul_0
```

```
In [52]: cal.write(0x10,15)
cal.write(0x18,3)
cal.read(0x20)
```

```
Out[52]: 18
```

```
In [53]: cal.read(0x30)
```

```
Out[53]: 12
```

```
In [54]: cal.read(0x40)
```

```
Out[54]: 45
```

```
In [55]: cal.read(0x50)
```

```
Out[55]: 5
```