项目介绍

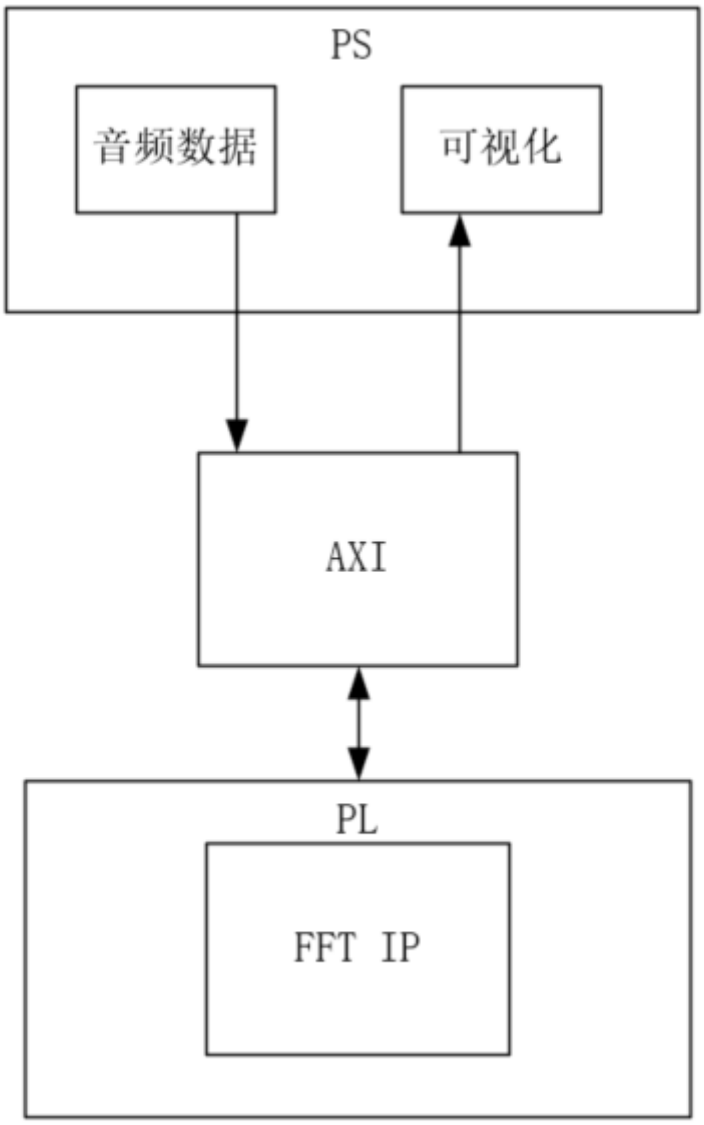
目标

- 使用Vitis HLS编写FFT算法，并利用HLS编译指令进行并行化处理，生成FFT的IP核部署到PL端
- 在PYNQ上将音频数据传输到PL端进行FFT硬件加速处理
- 最终将返回的逐帧音频数据在jupyter notebook上利用matplotlib和plotly动态可视化

环境要求

- PYNQ-Z2
- Vitis HLS
- Vivado

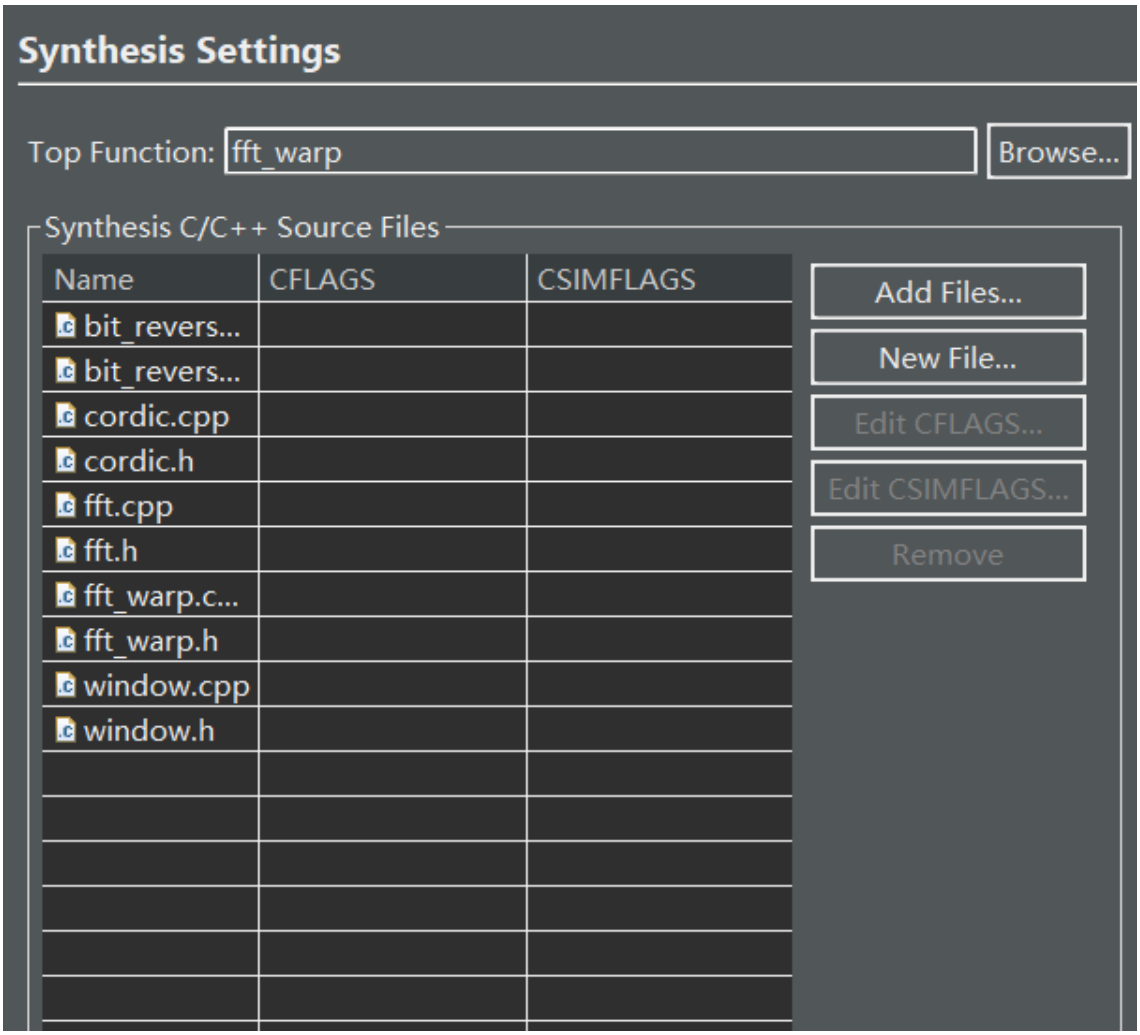
项目设计



项目步骤介绍

1. 在Vitis_HLS中设计FFT IP

- 打开Vitis HLS软件，点击Create Project，创建一个新的项目
- 在Project name输入项目名fft_hls，点击Browse选择一个合适的目录位置，点击Next
- 点击Add Files...，将src目录下的所有.cpp和.h文件添加到项目中



- 点击Top Function栏中的Browse按键，选择fft_wrap，这是我们进行综合时候的顶层函数，点击Next
- 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
- 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
- 导出RTL ，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. 在Vivado中进行IP集成

2.1 创建一个新Vivado项目

- 打开Vivado软件，点击Create Project，创建一个新的项目，点击Next
- 在Project name输入项目名fft_vivado_prj，点击右侧的 ... 按键选择一个合适的目录位置，点击Next

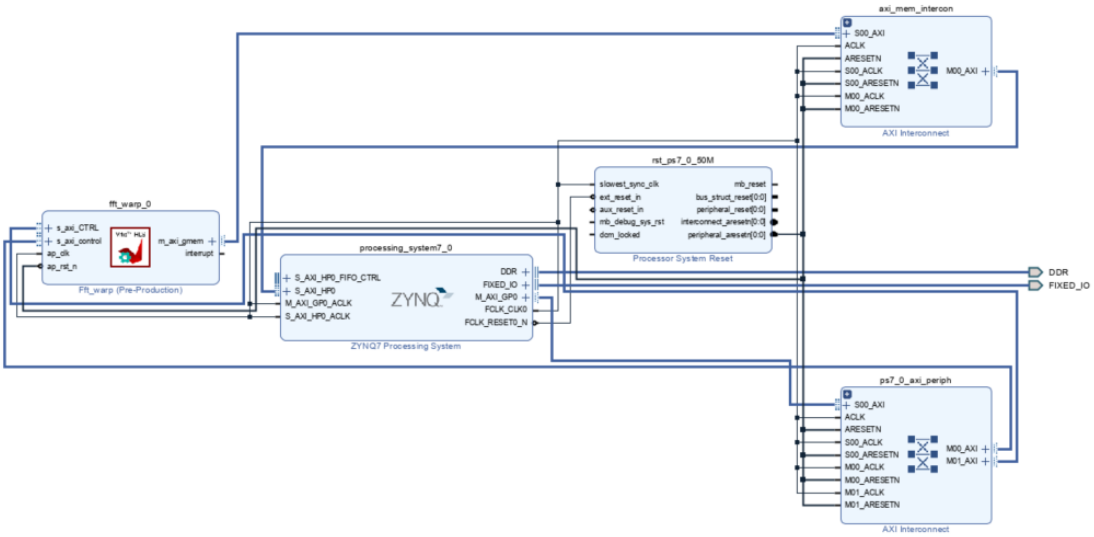
- 进入Project Type界面，勾选上Do not specify sources at this time，再点击Next
- 进入Default Part界面，在Search栏中搜索xc7z020clg484-1，将其选中，再点击Next

2.2 导入IP

- 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
- 将左侧的Project Settings中展开IP栏目，选中Repository项，点击右侧面板中的 + 按键，在弹出窗口中选择刚才解压出来IP，即\fft_hls_prj\solution1\impl\export，在点击Select

2.3 创建Block Design

- 左侧的FlowNavigator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
- 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
- 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
- 我们需要对上述Processing System进行配置，添加一个HP端口双击Diagram中的processing_system7_0模块，弹出Re-customize IP窗口, 在左侧Page Navigator中选择PS-PL Configuration页面，展开右侧选项中的HP Slave AXI Interface，勾选上S AXI HP0 interface选项点击OK
- 点击Diagam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Fft_wrap以将其添加到设计中
- 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
- 自动连线之后会得到如下的设计



- 在Diagram上侧的工具栏中点击勾形图标Validate Design，对设计进行验证
- 在左侧的Source > Design Sources > design_1选项上右键，选择Generate Output Products
- 在左侧的Source > Design Sources > design_1选项上右键，选择Create HDL Wrapper，在弹出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

- 在左侧的Flow Navigator中选择Run Synthesis，在弹出窗口中保持选择不变并选择OK
- . 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
- 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

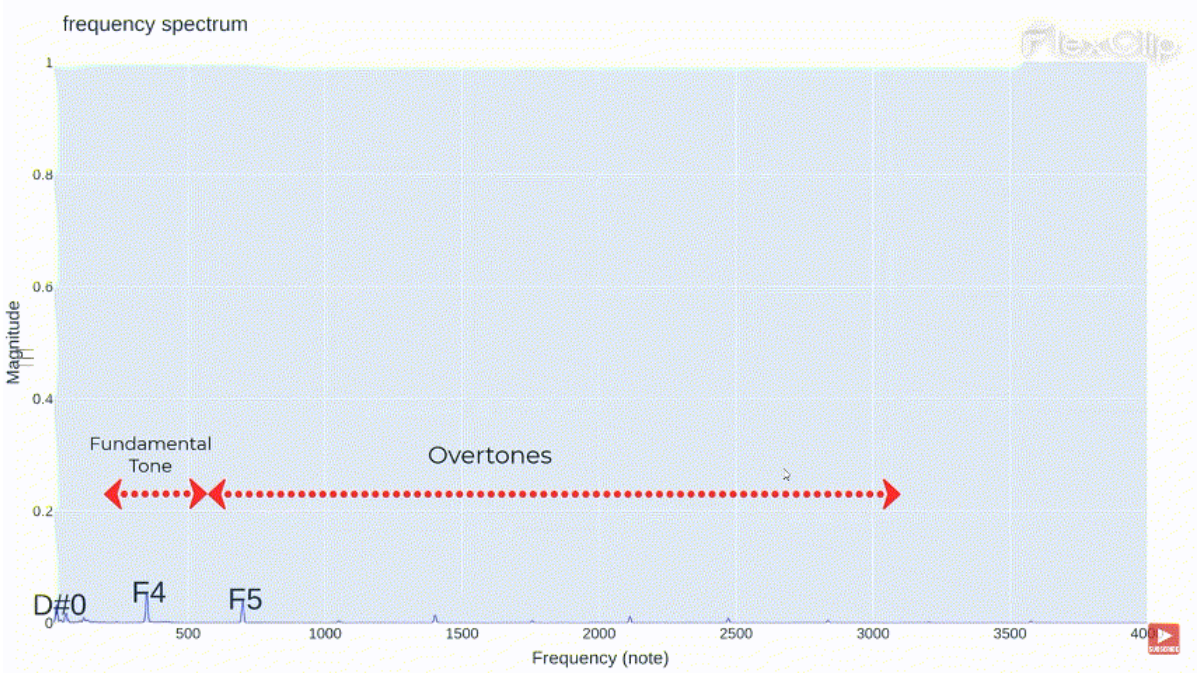
3. 构建PYNQ设计

3.1 提取bit与hwh文件

- 在文件管理器中访问 \ffr_vivado_prj\ffr_vivado_prj.runs\impl_1 目录，该目录下的design_1_wrapper.bit文件即为生成的比特流文件，将其复制到自己的文件夹中保存，并重命名为fft.bit
- 在文件管理器中访问\fft_vivado_prj\fft_vivado_prj.gen\sources_1\bd\design_1\hw_handoff 目录，其中的design_1.hwh即为我们需要的hardware handoff

  文件，将其复制到自己的文件夹中保存，并重命名为fft.hwh

项目预期结果



项目心得

经过本次为期10天的FPGA暑期学校的学习与实践，我们组从零基础到熟悉基本的FPGA项目设计流程，并动手尝试着去实现一个自选项目，虽然最后没有达到预期的效果，但是在这个过程中我们收获非常多。首先是在项目设计之前应该了解硬件资源和算法的可优化程度，当成熟的设计思想与可行的算法但是遇到受限的硬件资源也无计可施。在开发过程中，我们原计划是打算实时处理音频（大概一首歌的3，4分钟长短），并实时可视化。但是当我们在HLS中设计算法时，发现仅仅处理10秒的音频都需要耗费非常长的时间，所以我们不得不尝试对算法进行优化，加入一些编译优化的指令，但由于我们对编译指令

的理解不够，使用基本就是各种报错，非常崩溃。在我们尝试了无数次之后，时间紧迫，最后一次在我们认为可以达到预期效果的优化时，C综合却报了一个未知的错误，至今都还没解决。最后因为时间紧迫，我们不得不退一步使用相对慢一些的版本，导出IP后，立即上板验证。最后在PYNQ中实际跑起来发现，最后生成动图运行时间实在是太长了，项目没有达到预期结果。

虽然最终的结果，没有跑出来，但是还是学到非常多，非常感谢暑期学校的主办方和各位老师提供这个机会，同时也希望暑期学校在未来越办越好！