**FPGA实践自选项目**

**HLS实现卷积神经网络MNIST手写数字识别**

1. 搭建pytorch的训练、量化、参数导出的完整工具链；
2. 纯python实现卷积神经网络用于算法硬件实现与数据对齐；
3. HLS实现AXI接口的卷积神经网络推理IP；
4. 部署在PYNQ上。

**环境：**

·支持pynq的板卡，无需DPU

·Vitis HLS

·Vivado

·pytorch

**1、python端软件工具链(src/sw文件夹中)**

**环境搭建:**

conda create -n torch python=3.11

pip3 install torch torchvision torchaudio（或者自己装GPU版本，本项目CPU版本足够）

pip3 install tqdm

**训练：**

详情看1\_train.py，训练结果导出为.pth文件

**量化：**

使用pytorch提供的int8量化

详情看2.0\_quant\_model.py，2.1\_get\_quant\_para.py

导出的参数除了量化后的权重，还有量化时用到的zeropoint和scale（HLS中没实现，实现可以增加准确率）

**参数导出：**

需要根据硬件所需要的形式导出参数

3.0\_hls\_weight.py

HLS实现的话，可以直接用C语言的数组，按维度顺序存放即可

**2、纯python实现卷积神经网络**

2.2\_quant\_test\_debug.py

使用六层for循环直接实现的卷积神经网络，脱离pytorch环境实现，方便debug

2.3\_quant\_test\_scale\_zp.py

带上scale和zp量化的实现，HLS中没用到

2.5\_quant\_test\_acc.py

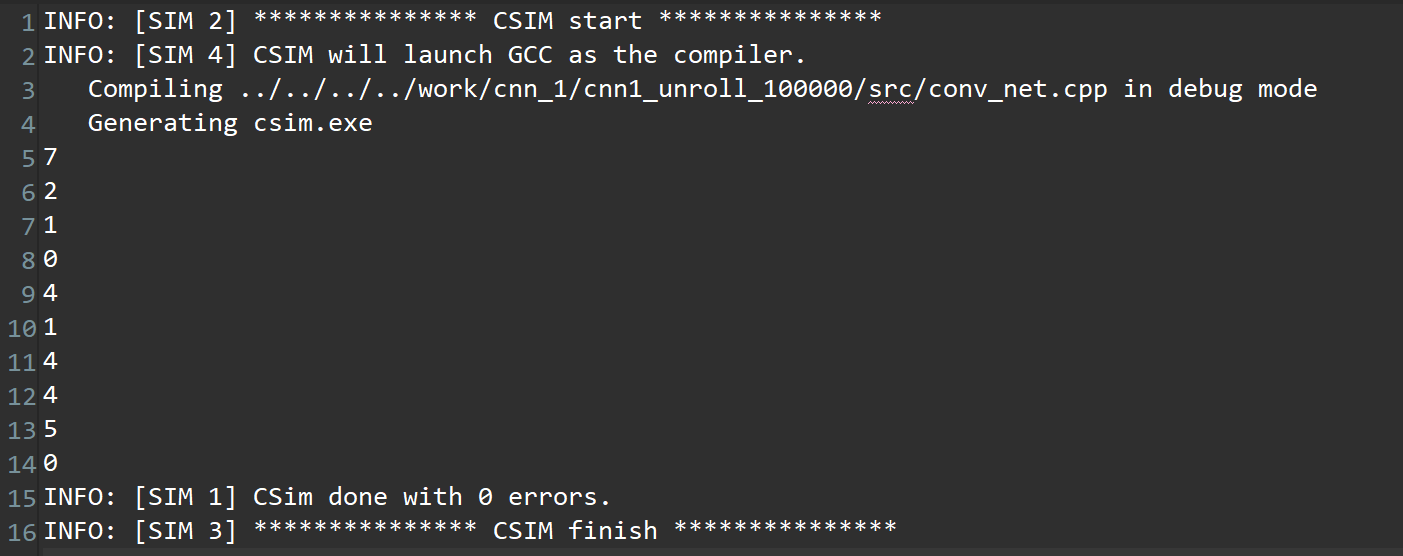
使用im2col实现的卷积神经网络加速，六层循环的实现太慢了，无法用于测试完整数据准确率，此实现主要用于测试量化前后准确率的变化。

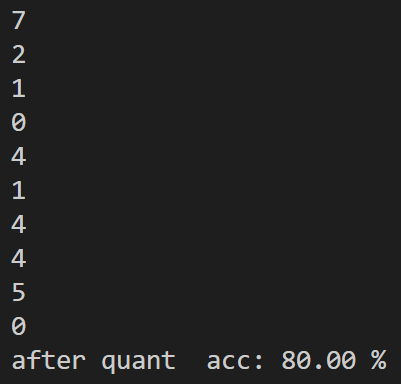
**3、HLS实现**

先将数据从DRAM中存入本地buffer，然后依次送入卷积、池化层，具体实现则是在六层循环的基础上仅加了unroll的优化，其他优化还没做。

接口为axi，从DDR中读取数据。

**仿真结果**

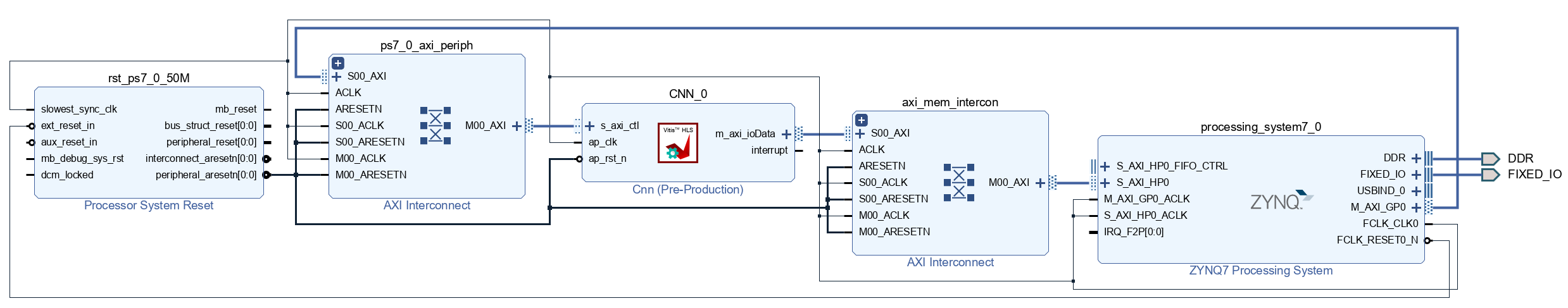




在HLS中仿真了十个数，结果都能对上，直接上板。

**4、PYNQ部署**

**BD框图**



软件代码可看jupyter



在50MHz时钟频率下，推理速度约为770FPS。

**5、总结**

本项目更多的是打通了从软件训练到硬件部署的全流程，其中的HLS实现优化没有做太多。

感兴趣的同学可以看我仓库中使用spinalHDL实现的卷积神经网络，性能更好，更符合硬件设计的思路。<https://github.com/androny1012/SpinalMNIST>