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

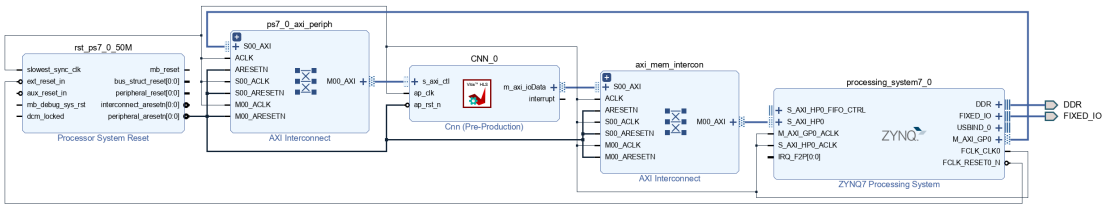
```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../work/cnn_1/cnn1_unroll_100000/src/conv_net.cpp in debug mode
   Generating csim.exe
7
2
1
0
4
1
4
4
5
0
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

```
7
2
1
0
4
1
4
4
5
0
after quant  acc: 80.00 %
```

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

```
:
import time
cnt = 0
num = 100
start = time.time()
for i in range(num):
    r          = test_data[i,:-1]
    label      = test_data[i,784]
    image_array_r = np.array(r,dtype='uint8')
    np.copyto(img_in_r,image_array_r)
    CNN_Init_EX()
    pred_out = np.zeros(1,dtype = np.int32)
    pred_out = img_out.copy()
#     print("The pred is ",pred_out[0])
#     print("The label is ",label)
    if(pred_out[0] == label):
        cnt += 1
end = time.time()
print(round((end-start)/num*1000,3),'ms/img')
print(cnt/num)
```

```
1.295 ms/img
0.89
```

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