- BNN_on_fpga
 - 项目概述
 - 网络结构
 - 硬件架构
 - 卷积模块
 - 最大池化层
 - 全连接模块
 - top模块
 - 项目总结

BNN_on_fpga

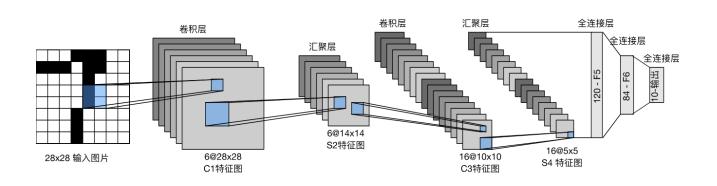
项目概述

这个项目是一个基于 FPGA 的二值神经网络(Binary Neural Network, BNN)加速器, 旨在加速 MNIST 数据集的手写数字识别任务,通过训练不同数据集,也可以用于其他任务,如fashionMNIST, cifar10(需要改通道数,重新调整时序)等。

二值化神经网络(BNN)是指仅使用+1和-1两个值来表示权重的神经网络,相比于全精度的神经网络,它可以使用加减这样的简单组合代替fp32的乘加来实现卷积操作,从而大大节省内存,减少计算开销。

本项目开源在了github上面,地址如下。 项目地址

网络结构



上图为经典的LeNet5网络,本项目为了减小复杂度,将三个全连接减少为一个,实际测试中并没有准确率下降。 lenet 2conv3fc 1bit量化,权重只为1或者-1

LeNet(

(conv1): BinaryConv2d(1, 6, kernel_size=(5, 5), stride=(1, 1), bias=False)

(pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)

(relu1): ReLU()

(conv2): BinaryConv2d(6, 12, kernel_size=(5, 5), stride=(1, 1), bias=False)

(pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)

(relu2): ReLU()

(fc1): BinaryLinear(in_features=192, out_features=10, bias=False)

)

硬件架构



conv1的输入是(1,28,28),输出通道为6,所以直接将并行度设置为6,第二个卷积层的输入通道为6,输出通道为12,所以一共是13次卷积操作。

卷积模块

首先是卷积的滑窗模块,由于卷积核尺寸是5*5,所以滑窗模块预存的是28*5=140个像素,需要支持两种模式,第一种模式的输入是28*28,第二种是在经过第一个卷积层和第二个池化层过后,输入变成了12*12,所以预存的像素点变成了12*5=60个,由于对于每五行,从上到下,从左到右读入mem中,所以第一个读入的点应该位于mem的最高位。

接下来是对该模块进行仿真,需要将图片转化为txt文件,每一行一个像素,用八位二进制无符号数表示。

接下来是卷积模块的核心部分,卷积相乘的部分,由于所有的权重都已经量化为了1bit,所以我们人为规定,1代表权重为1,0代表权重为-1,转化为硬件电路里面,便是为1时

加上这个激活值,为0时减掉这个激活值。激活值与权重对应相乘结束过后,进行流水线操作进行结果相加,一共是7级流水线。

执行流程如下

- 加载权重矩阵,一共是150个周期,因为并行度是6,所以是655=150
- 在滑窗模块加载完过后,还需要4个周期,流水线的第一次运行还需要7个周期,一 共就是161个周期。
- 在卷积计算完成的时候,因为输出是24**24**,所以在这里需要移位**24**24次,同时滑窗模块换行的时候,需要进行同步操作,一共换行23次,所以是23*4次,加起来就是829

仿真部分,硬件模拟软件,首先用pytorch生成一个卷积,权重全为-1,

```
conv = nn.Conv2d(1, 1, 5, 1, 0, bias=False)
conv.weight.data.fill_(-1)
output = conv(image)
```

卷积完的结果如下

```
torch.Size([1, 24, 24])
             0,
tensor([[[
                   0,
                         0, 0,
                                       0,
                                              0,
                                                     0,
                                                           0,
                                                                  0,
                                                                         0,
             0,
                   0,
                         0,
                               0,
                                       0,
                                              0,
                                                     0,
                                                           0,
                                                                  0,
                                                                         0,
             0,
                   0,
                         0,
                                0],
                                 0,
                                                           0,
             0,
                   0,
                          0,
                                       0,
                                              0,
                                                     0,
                                                                -3.
                                                                     -21,
                 -57, -183, -316, -473, -481, -629, -758, -869,
           -39.
          -795, -629, -374, -127],
                                 0,
                                     -30, -66, -160, -314, -487, -728,
                   0,
                          0,
          -963, -1140, -1365, -1581, -1710, -1637, -1785, -1903, -1956, -1747,
         -1549, -1130, -633, -191],
                                    -317, -606, -953, -1360, -1737, -1993,
                          0,
                             -49,
                   0,
         -2228, -2405, -2630, -2844, -2813, -2569, -2546, -2467, -2308, -2006,
         -1726, -1225, -672, -191],
                             -67, -554, -1096, -1696, -2356, -2968, -3258,
                   0,
                          0,
         -3438, -3544, -3763, -3965, -3681, -3239, -3034, -2708, -2308, -2006,
         -1726, -1225, -672, -191],
                          0,
                             -67, -634, -1332, -2039, -2952, -3817, -4232,
                    0,
         -4267, -4266, -4275, -4378, -3889, -3436, -3231, -2862, -2308, -2006,
         -1726, -1225, -672, -191],
                          0, -67, -634, -1346, -2054, -3121, -4236, -4723.
                   0,
         -4726, -4706, -4435, -4152, -3416, -2955, -2602, -2104, -1439, -1185,
```

同时用硬件进行仿真,结果打印在控制台上面,如下图所示

-629 -1785 -2546 -3034 -3231 -2602 -1446 -685 -306 -506 -1189 -1974 -2789 -3578 -4119 -3968 -3266 -2451 -1553 -615 -83 -21 -728 -1993 -3258 -4232 -4723 -4598 -1905 -1393 -811 -357 -799 -1837 -3938 -4034 -3422 -2384 -1177 - 39 - 963 - 2228 - 3438 - 4267 - 4726 - 4386 - 2428 - 1975 - 1391 - 867 - 412 - 507 - 1327 - 2552 - 3586 - 4181 - 3913 - 3048 - 1823 -183 -1365 -2630 -4275 -4435 -3698 -2946 -2582 -2897 -3038 -2702 -2189 -1779 -1874 -2623 -3552 -4089 -681 -681 -144 -316 -1581 -2844 -3965 -4378 -4152 -3079 -1753 -2286 -2933 -3102 -3028 -2905 -3095 -3571 -3997 -4032 -3431 -2295 -1082 -295 -11 -473 -1710 -2813 -3681 -3889 -3416 -2181 -1148 -774 -1456 -2343 -2955 -3788 -4106 -4257 -4177 -3764 -2899 -1661 -623 -89 -481 -1637 -2569 -3239 -3436 -2955 -1799 -867 -466 -919 -1788 -2589 -3340 -4037 -4638 -4554 -4034 -3292 -2266 -1075 -290 -758 -1903 -2467 -2762 -2862 -2104 -959 -395 -155 -145 -575 -1267 -2082 -2796 -3337 -3186 -2496 -1681 -966 -281 -20 - 869 - 1956 - 2308 - 2308 - 2308 - 1439 - 352 0 0 0 - 25 - 202 - 642 - 1208 - 1670 - 2077 - 1978 - 1538 - 972 - 510 - 78 0 0 -821 -1747 -2006 -2006 -2006 -1185 -259 0 0 0 -27 -214 -527 -736 -918 -891 -704 -391 -182 -374 -633 -672 -672 -672 -298 -39 0 0 - 314 - 1360 - 2356 - 2952 - 3121 - 2946 - 1911 - 915 - 319 - 150 0 0 - 138 - 693 - 1842 - 3098 - 3846 - 3708 - 3153 -57 -1140 -2405 -3544 -4266 -4706 -2988 -2840 -2574 -2006 -1482 -997 -1092 -1955 -3149 -3939 -4151 -3498 -2404 -1194 0 - 66 6 - 606 - 1096 - 1332 - 1346 - 1280 - 740 - 250 - 14 0 0 0 0 0 - 89 - 750 - 1907 - 3013 - 3013 - 2924 - 2352 -160 -953 -1696 -2039 -2054 -1894 -1101 -358 -15 0 0 0 0 -24 -326 -1240 -2478 -3463 -3439 -3137 -37 -487 -1737 -2968 -3817 -4236 -4144 -3095 -1899 -1050 -628 -236 -35 0 -399 -398 -1206 -2437 -3570 -2437 -3570 -2437 -3570 -2862

可以看到、结果是一样的、说明卷积模块的功能是没有问题的。

最大池化层

池化层的尺寸是2*2,首先缓存两行输入数据,接着通过一个二级流水线,先让下面的两个像素点分别与对应的第一行的像素点比大小,然后设置一个寄存器保存最大值,然后将该寄存器的两个值进行大小比较,最终输出最大值。

仿真方法与卷积模块结果一样

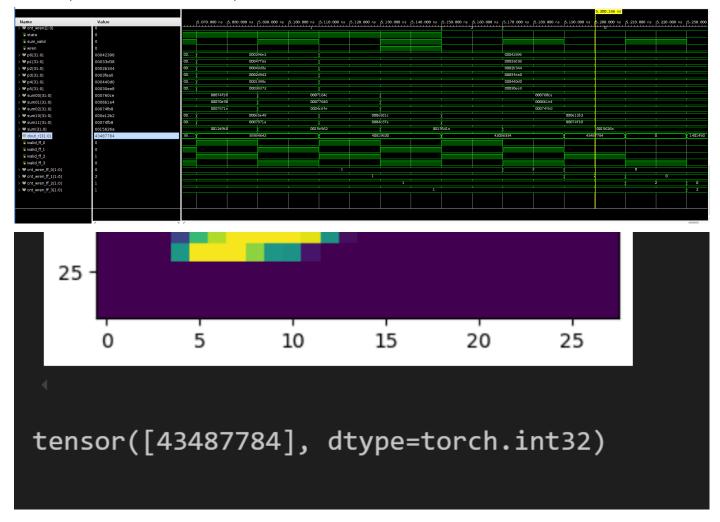
```
tensor([[[
               0,
                      0,
                              0,
                                      0,
                                             0,
                                                     0,
                                                             0,
                                                                    0,
                                                                            0,
                                                                                   0,
                      0],
               0.
                            -30,
                                  -160,
                                          -487,
                                                 -963, -1365, -1637, -1785, -1747,
               0,
                      0,
           -1130,
                   -191],
                           -554, -1696, -2968, -3438, -3763, -3239, -2708, -2006,
               0,
                   -191],
          -1225,
                           -604, -1894, -4144, -4207, -3079, -1799,
                      0,
             -95,
                      0],
                            -80,
               0,
                      0,
                                  -358, -1899, -2936, -1753,
                                                                 -466,
                                                                        -155,
                                                                                   0,
                      0],
               0,
               0,
                      0,
                              0,
                                      0,
                                          -628, -1975, -2286,
                                                                 -919,
                                                                        -145,
                                                                                   0,
               0,
                      0],
               0,
                      0,
                              0,
                                      0,
                                           -35,
                                                 -867, -2189, -2589, -1267,
                                                                                -214,
               0,
                      0],
                                      0,
                                                 -412, -1779, -3758, -2796,
               0,
                      0,
                              0,
                                                                                -736,
               0,
                      0],
                                          -398, -1327, -2623, -4034, -2496,
               0,
                      0,
                              0,
               0,
                      0],
                    -18,
                           -431, -1240, -2437, -3586, -3431, -2266,
               0,
                                                                                -182,
                      0],
               0,
```

INFO: [Wavedata 42-604] Simulation restarted run 30 us

run 50	u o												
picture	e read over												
maxpool	ling comple	te											
	0 :	0	0	0	0	0	0	0	0	0	0	0	0
	1 :	0	0	-30	-160	- 487	-963	-1365	-1637	-1785	-1747	-1130	-191
	2 :	0	0	-554	-1696	- 2968	-3438	-3763	-3239	- 2708	-2006	-1225	-191
	3 :	0	0	-604	-1894	-4144	- 4207	- 3079	-1799	- 959	- 259	- 95	0
	4 :	0	0	-80	-358	-1899	- 2936	-1753	- 466	-155	0	0	0
	5 :	0	0	0	0	-628	-1975	-2286	-919	-145	0	0	0
	6 :	0	0	0	0	-35	-867	-2189	- 2589	-1267	-214	0	0
	7 :	0	0	0	0	0	-412	-1779	-3758	- 2796	-736	0	0
	8 :	0	0	0	- 24	-398	-1327	- 2623	- 4034	- 2496	-704	0	0
	9 :	0	-18	- 431	-1240	- 2437	-3586	-3431	-2266	-966	-182	0	0
	10 :	-191	-1113	- 2497	-3439	-3422	-2404	-1082	- 290	-2	0	0	0
	11 :	-191	-1095	- 2066	-2004	-1177	- 404	-11	0	0	0	0	0
	12 :\$fin	ish called	at time : 1	.1560 ns : Fil	e "/home/cu	rry/code/curi	ry_code_summa	ay/rtl_works/	/BNN_on_fpga/	/rtl/maxpool_	tb.v" Line 1	.09	

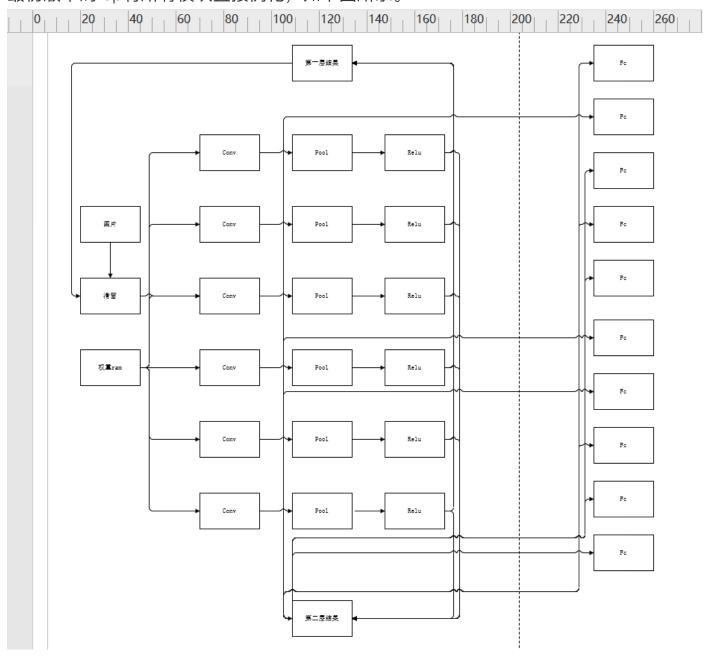
全连接模块

由于网络第二层一共有12个输出通道,feature map大小是16,所以全连接层一共接收12个输入,流水线级接受16次,最终4级累加到最后结果。



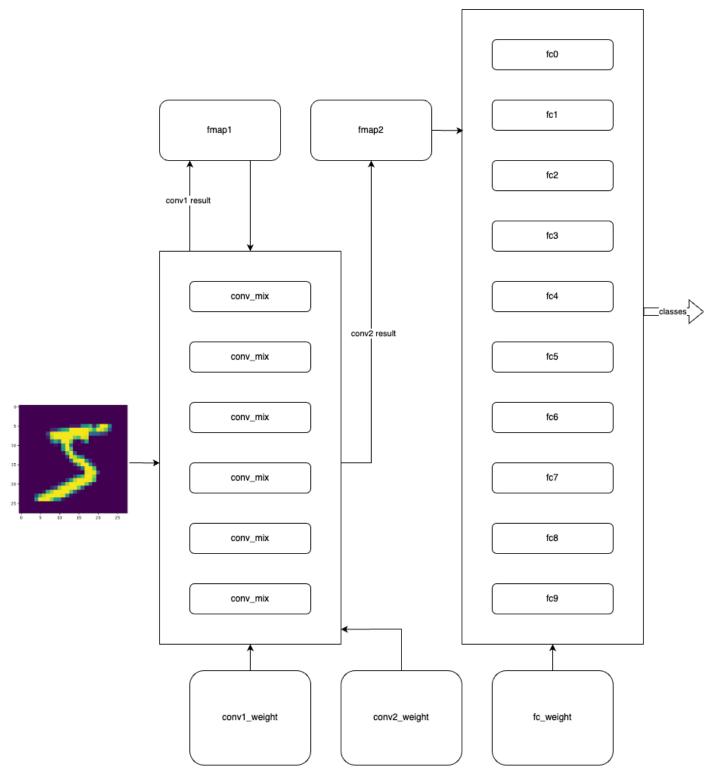
top模块

最初版本的top将所有模块直接例化,如下图所示。



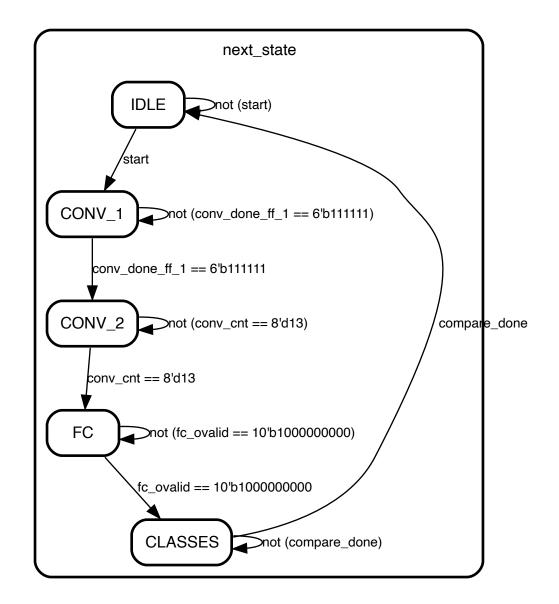
在实际过程中调度过于复杂,波形信号有几百条,最终决定做一个化繁为简的操作,将 卷积模块,池化模块,ReLU模块进行合并,通过再多添加流水线实现卷积后的池化和激 活操作。

更改过后的架构图如图所示。



第二版例化六个融合卷积模块,10个全连接,最后再通过五级流水线输出最大值以及对应类别。

权重被存在了ROM当中,将pth文件的权重导出为coe文件,到BRAM的IP核配置界面进行加载,一开始权重晚于地址一个clk读出,但是由于IP核是在综合过后加入仿真的,因此存在一定的偏移和延迟,于是进行了打拍处理。 运算过程的状态机如图所示。



最后我们实现了对数字的分类,如图所示。输入数字是5,最终全连接得到的输入如下

> Wfc_out_reg[9:0][31:0] -29588,1822 -29588,18224,-11862,-1.1906,274446,-75610,118808,-121240,-199346,-221848

与pytorch得到的输出一致,如图所示

tensor([[-221848, -199346, -121240, 118808, -75610, 274446, -111906, -11862, 18224, -29588]], dtype=torch.int32)

对应分出的类别打印在控制台上,如图所示

classes[0] = x	
classes[1] = x	
classes[2] = x	
classes[3] = x	
classes[4] = x	
classes[5] = 1	
classes[6] = x	
classes[7] = x	
classes[8] = x	
classes[9] = x	
classes[10] = x	

项目总结

这个项目verilog代码一共有7000多行,已经是一个比较大的工程了,在此之前,本人也未独自完成一个神经网络加速器的全部流程,因此在项目执行过程中遇到了很多问题。

- 最初的验证方案未确定完全,后期多次修改Jupyter里面的python代码进行打印调试,浪费了很多时间。
- 最初对于卷积的运算顺序并未考虑清楚,比如在第二层卷积运算完全进行通道相加的时候出现顺序错误。
- 因为本人并未掌握UVM验证方法学,所以没有通过打印checkpoint的方法进行调试,实际执行时仍然采用用verilog写tb,然后看波形的方法,消耗了大量时间。