Final Project: Your First DNN Accelerator!

Model Detailed Information:

Layer	Type	IFM Size	Input Channel	OFM Size	Output Channel	Kernel Size	Stride	Padding (zero)	ReLU	Q_{IN}	Q_{OUT}	Q_W
Layer-1	Conv	32^2	1	32^2	32	5×5	1	2	√	7	5	7
Layer-2	Max Pooling	32^2	32	16^2	32	2×2	2	0	×	5	5	/
Layer-3	Conv	16^2	32	16^2	64	3×3	1	1	√	5	5	8
Layer-4	Conv	16^2	64	8^2	64	3×3	2	1	√	5	5	8
Layer-5	Conv	8^2	64	4^2	128	3×3	2	1	√	5	5	8
Layer-6	Average Pooling	4^2	128	1^2	128	4×4	1	0	×	5	5	/
Layer-7	FC	1	128	1	10	1×1	/	/	×	5	5	6

This model is an MNIST-based (10 categories) simple DNN with all the data quantized in linear 8bit with 2^{-Q} scaling. The inputs of the model are normalized to 0~1 (so Q_{IN} of the 1st layer is 7). The final results of the model are 10 bytes (i.e. the outputs of the FC layer, no need to consider the Softmax function). The amount of all the weights in this model is only 128KB, and you can store them all in the BRAM. For convenience, bias and BN function are not used. Therefore, it is a very simple linear calculation for you to implement the DNN accelerator. The quantization method for the next layer is ***floor(x)*** (not *round* or *ceil*).

Requirements:

1. Firstly, you need to develop a golden model in a high-level language (C, C++, Python, etc.) to generate the current results as what we provided.
2. Implement an FPGA-based DNN accelerator in the PL part of our development board.
3. The operation frequency is at least 100MHz.
4. You are allowed to store all the weights on board. But the total capacity of BRAM should be less than 200KB. You are required to figure out some solutions to overwrite the out-of-date data.
5. A parallel design is required. The performance should be more than 64 MACs per cycle with an average utilization of more than 80% in total. (How to evaluate the utilization: # busy cycles / # total cycles).

Reports:

1. Prove your high-level model results match all the provided results (software);
2. Prove your hardware results match all the provided results (simulation results);
3. Your architecture introduction with detailed figures and descriptions. And you also need to describe what's your parallel strategy (dataflow, data mapping, etc.)
4. Show your module design hierarchy.
5. An STA report to claim your frequency is no less than 100MHz;
6. Power report, resources utilization report (find them in the VIVADO tool) of the implementation version.
7. Provide the end-to-end runtime of the whole model (or the scope of model you implemented).
8. Claim the interface of your DPU design on the PL side.
9. Introduce your system framework and submit a video if you have successfully run your design on board.
10. **<u>You will present your work in the last course (December 23, 2020)</u>**

附

课程评价标准(每个档位中所列举的要求，满足其中一项即可)

第一档（课程分数95-100）：
（1）Final Project 通过整个Model测试，并上板
（2）Final Project 通过整个Model测试，并具有一定创新性，并完成仿真验证

第二档（无作业不提交前提下，课程分数不少于90分）
（1）Final Project 通过整个Model测试，并仿真
（2）Final Project 通过单层卷积测试，并上板

第三档（课程分数<90，最后总成绩结合作业、出勤情况）
（1）Final Project 通过单层卷积测试，并仿真

最后展示会对各组同学所设计的加速器各项指标评分，前三名有额外加分
（评价指标：计算层数↑，频率↑、处理时间↓、功耗↓，是否上板）