LSU EE 7722 **Homework 4** **Due: 29 April 2019**

In class we covered NVIDIA GPUs which are reasonably well suited to DNN (deep neural network) computations, especially the CC 7.X devices. The NVIDIA GPUs are successful commercial products and are the result of several generations of refinement and evolution. For that reason we can assume that the design is effective and that features work as intended for graphics and typical scientific workloads. For example, we can feel safe concluding that having a large number (2048!) of thread contexts per SM is a good way of hiding memory latency despite the huge area taken by the registers.

The three papers assigned towards the end of the semester (see the end of this assignment for a complete reference) describe designs for DNN accelerators of various maturity. The Google TPU described by Jouppi *et al* [3] is used in production so we can assume it is effective. It describes a straightforward way of implementing DNNs and one robust enough to handle production work and certainly devoid of complex ideas that would add to development time and have a risk of limited effectiveness.

Chen, Elmer, and Sze [1], introduce a CNN dataflow, row stationary, and compare it to some previously described dataflows. In their comparison methodology they configure implementations of each dataflow by optimizing modeled energy consumption. The idea is to obtain for comparison high-quality configurations of each dataflow under similar hardware constraints. The row-stationary dataflow is not revolutionary, but the equal-footing comparison methodology makes it interesting. Also, the paper is from a project to fabricate a working DNN accelerator chip, Eyeriss, and so one can expect that the design is workable.

The third paper, by Hegde *et al* [2], is the most interesting but also describes results which are less likely to be practical in the end. Their idea is to exploit repeated weights in a filter by computing $w_1(a_1 + a_2)$ instead of $w_1a_1 + w_2a_2$ when $w_1 = w_2$. The ostensible benefit is fewer multiplies and handling fewer weights. The big question is whether the effort needed to deliver the inputs in the right order to the now unique weights is greater than the benefits.

Problem 1: Compare the performance of the accelerators described in the three papers above, and for a GP100 GPU, as described below.

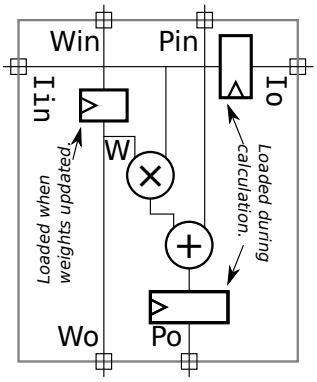
(*a*) Make the following unfair comparison: Find the peak operation rate, in units of multiply/add (accumulate) operations per second, for each of the three accelerator designs. Base this on the number of MADD (or MAC) units per chip and the clock frequency.

For each device show the peak MADD rate and explain how that was calculated. For UCNN show both peak MADD and also the peak number of input elements processed per second (which would be the MADD rate in other devices). In some cases the MADD (or equivalent) rate is provided in the publication. Show that number, but also calculate it from information provided in the publications, such as the clock frequency.

If a publication describes multiple variations pick a good one, and indicate which one you chose.

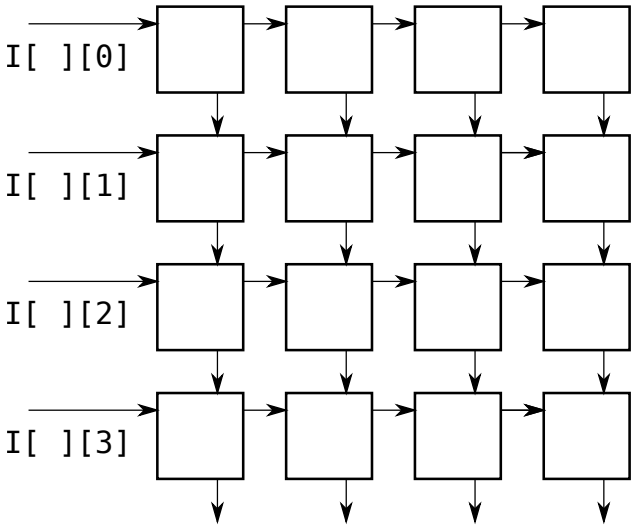
(*b*) Make some effort to compare the number for each device in a fair way. Take into account energy, precision, and anything else that sounds relevant. For example, one might compare them in terms of computation rate for a fixed amount of power or computation time for a fixed amount of energy.

Problem 2: The diagram below shows a possible design of a MAC, the elements that make up the Matrix Multiply Unit. An input activation, labeled Iin, enters on the left and leaves one cycle later on the right, at the port labeled Io. During calculation partial sums enter at the top (Pin) and exit one cycle later from the bottom. When new weights are being loaded (for example, when changing input channels) weights enter from the top and exit from the bottom. In the Google TPU the MACs are arranged into a 256×256 array, and so during calculation an input activation appearing at the input to a leftmost MAC will take 256 cycles to reach the rightmost MAC. Similarly it will take 256 cycles for a partial sum to reach the bottom, and it will take 256 cycles to load a new set of weights.

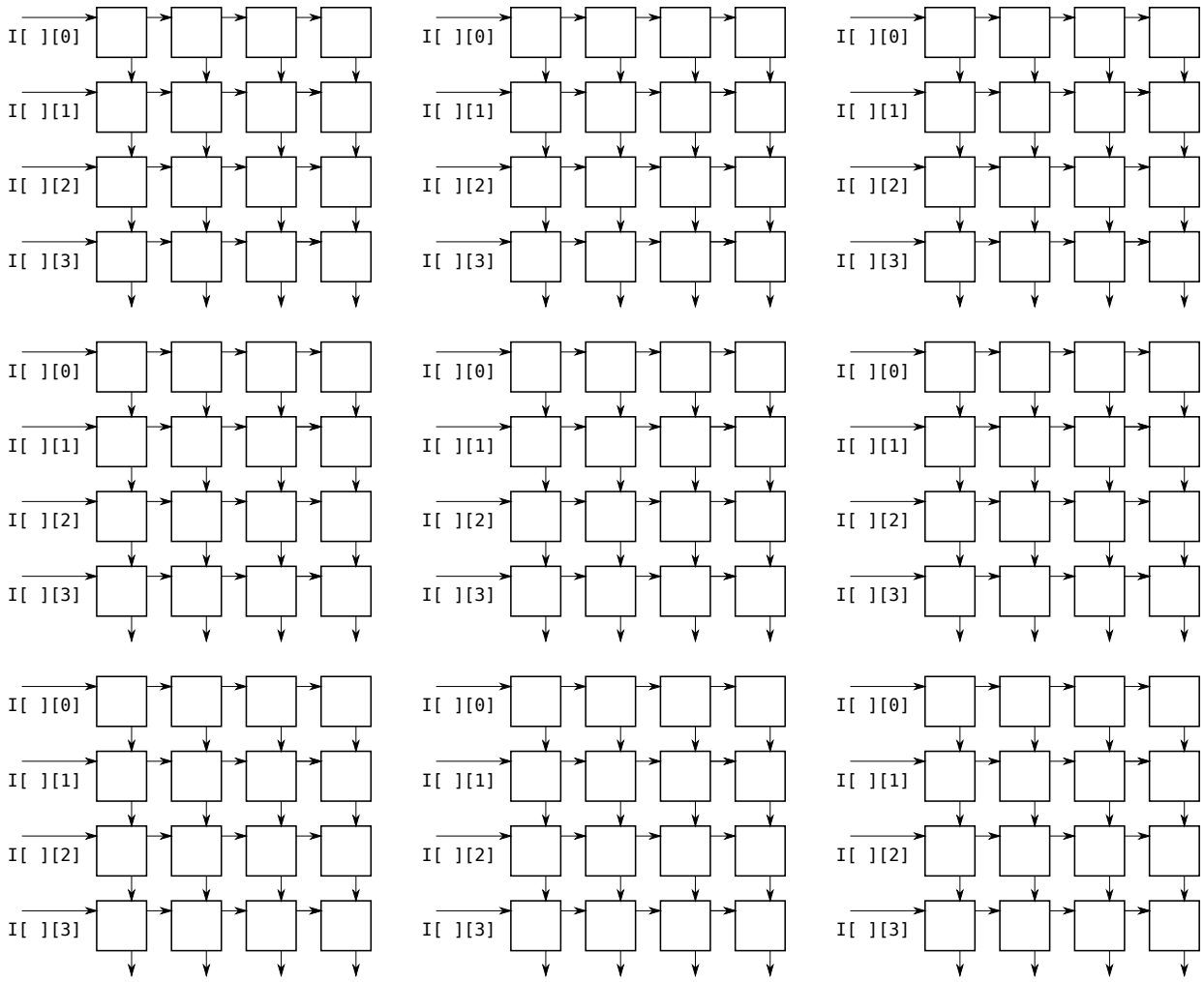


Illustrated below is an array of 4×4 mac units, with inputs partially labeled. Connections are shown for the inputs and partial sums, but not for the weights. The array is used to compute `O[n][x] += I[n][i] * W[i][x]`.

(*a*) Show the weights assigned to each MAC. Use $w_{1,2}$ to indicate `W[1][2]`.

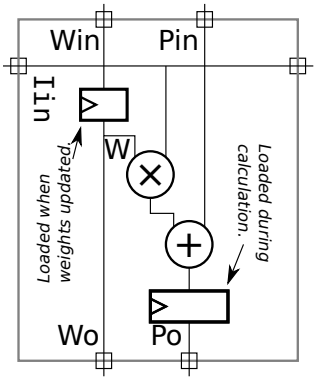


(*b*) Illustrated below are multiple copies of the array. The first one is for cycle 0, when $I_{0,0}$ is at the upper-left MAC. The second one is for cycle 1. The others can be labeled with cycles of your choosing. Show the input element being operated on in each active MAC at each clock cycle when multiplying a 4-element activation ($X = 4$) with a batch size of 2 for a fully connected layer. Use $I_{n,x}$ for input activation element x in batch n. There is no need to show every cycle if the pattern is obvious.

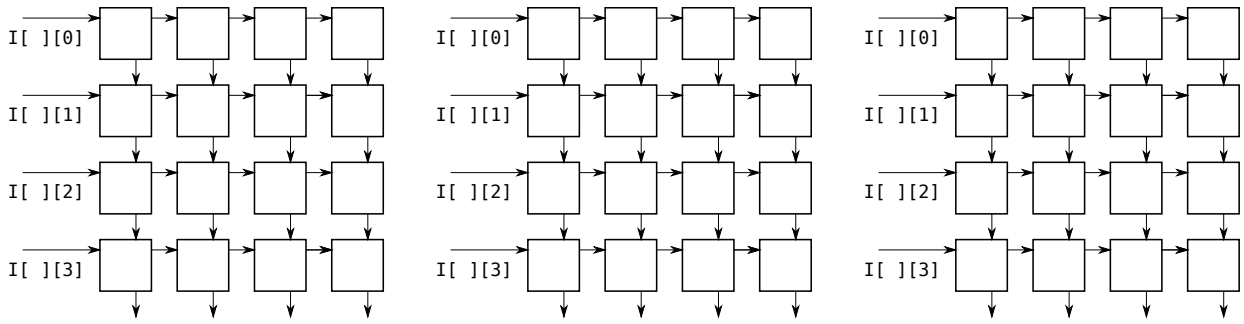


(*c*) How many cycles will it take to compute an activation of length $X \leq 256$ and a batch of size N?

Problem 3: Appearing below is an alternative MAC design in which the activation is not buffered. An input appearing at the leftmost MAC will be visible to the entire row. Such an organization requires fewer registers but cannot operate as fast (especially if there are 256 columns!).



(*a*) As with the previous problem, show the input element operated on, but only for the first three cycles.



(*b*) How many cycles will it take to compute an activation of length $X \leq 256$ and a batch of size N?

References:

[1] Chen, Y.-H., Emer, J., and Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Proceedings of the 43rd International Symposium on Computer Architecture* (Piscataway, NJ, USA, 2016), ISCA '16, IEEE Press, pp. 367–379. https://doi.org/10.1109/ISCA.2016.40.

[2] Hegde, K., Yu, J., Agrawal, R., Yan, M., Pellauer, M., and Fletcher, C. W. UCNN: Exploiting computational reuse in deep neural networks via weight repetition. In *Proceedings of the 45th Annual International Symposium on Computer Architecture* (Piscataway, NJ, USA, 2018), ISCA '18, IEEE Press, pp. 674–687. https://doi.org/10.1109/ISCA.2018.00062.

[3] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-l., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., and Yoon, D. H. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture* (New York, NY, USA, 2017), ISCA '17, ACM, pp. 1–12. http://doi.acm.org/10.1145/3079856.3080246.