Automatic Design Space Algorithm for DNN Accelerators Under Total Resources Constraint

Akarsh Kumar, Elyssa Hofgard, Omri Lev
MIT EECS

Motivation

- Deep neural networks (DNNs) have revolutionized the field of machine learning.
- This has led to a need for energy-efficient hardware for both training and inference.
- The hardware design space for accelerators is quite large and exploring it by hand would be infeasible.

CONV Layers and Eyeriss Chip

One of the main operations in DNNs is the CONV layer (shown in Einsum notation below). A variety of mappings and dataflows for the CONV layer have been explored. The Eyeriss chip implements a row-stationary dataflow at the RF level to maximize data reuse and accumulation.

$$\mathbf{O}_{nmpq} = \left(\sum_{crs} \mathbf{I}_{nc(Up+r)(Uq+s)} \mathbf{F}_{mcrs}\right) + \mathbf{b}_m$$

Figure 1. CONV layer Einsum equation.

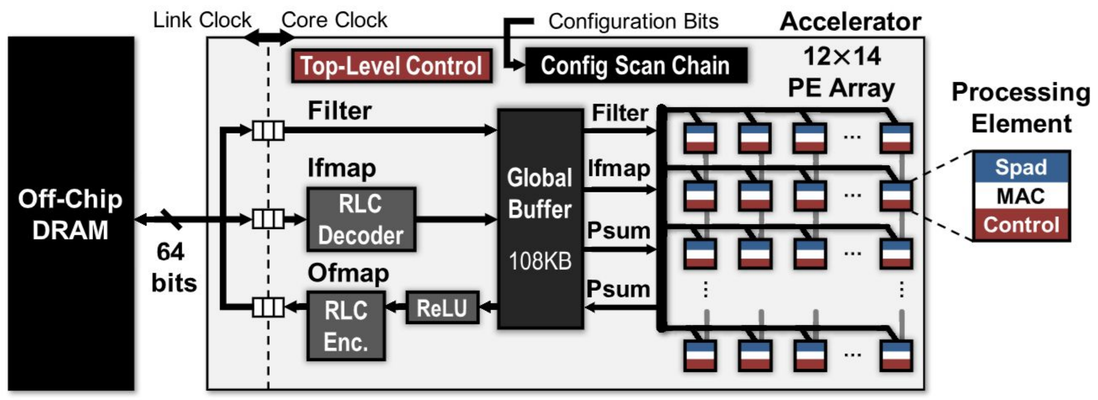

Figure 2. Eyeriss DNN accelerator.

Genetic Algorithm (GA)

Derivative-free optimization method

Algorithm 1 Genetic Algorithm

Initialization: Generate the initial population $\{x_1, \ldots, x_N\}$.
for iter $= 1, ..., G$ **do**
Get fitness f_i for each x_i
Select top K solutions as parents
Sample $N - 1$ children from the selected parents
Mutate children by perturbing each dimension with probability σ
Next generation is children + elite solution from previous generation unchanged
end for

Figure 3. Genetic algorithm.

Project Goals

We aim to find a configuration of $(\text{meshx}, \text{meshy})$ under the constraint $\text{meshx} \times \text{meshy} = 168$ and a configuration of the size of the global buffer and the scratchpads s.t. GFLOPs and Utilization are maximized, and Cycles, Energy, and EDP are minimized.

To do so, we build a tool using genetic algorithms with a customizable loss function.

Methodology

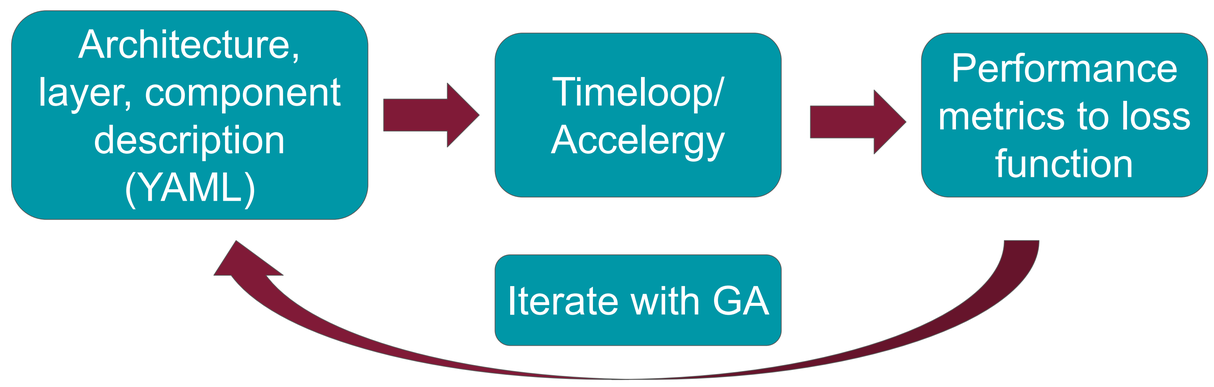

Figure 4. Optimization algorithm.

GA minimizes custom loss function

$$\text{loss} \triangleq a_1 \cdot \text{Cycles} + a_2 \cdot \text{Energy} - a_3 \cdot \text{GFLOPs} - a_4 \cdot \text{Utilization}$$

Results

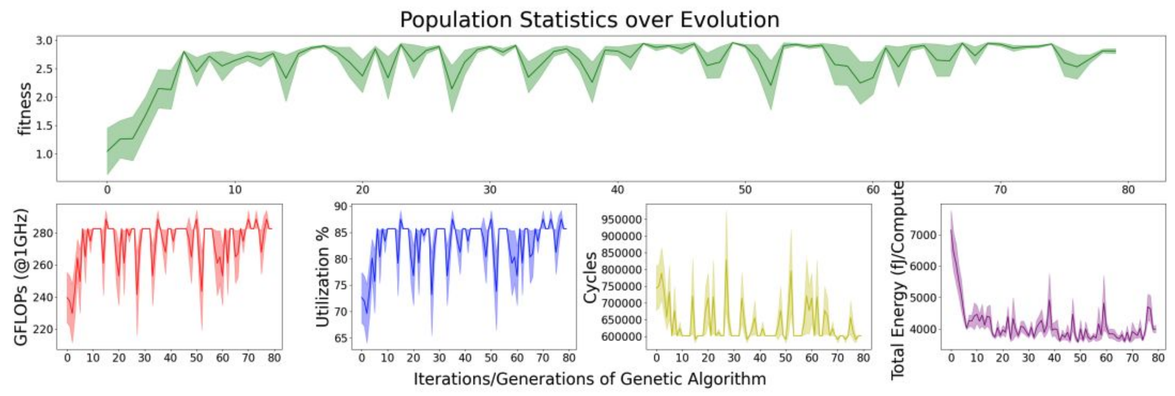

Figure 5. GA run results.

Design Spec	Original	Optimal
shared glb	16384×64	512×512
mesh	14×12	4×42
ifmap spad	12×16	6×64
weights spad	192×16	48×64
psum spad	16×16	8×128

Table 1. Original and optimal parameters.

Metric	Original	Optimal
GFLOPs (@1GHz)	188.48	**282.67**
Utilization (%)	57.14	**85.71**
Cycles	903168	**602112**
Energy (μJ)	720.99	**305.38**
EDP ($J \times$ cycle)	651.0	**184.0**

Table 2. Original and optimal metrics.

Conclusion & Further Steps

We found significantly better design solutions (in all metrics) using GA optimization. Because GAs are gradient-free, they are extremely scalable to large and messy search spaces and should be extended to a bigger portion of the design space of chips.

References

[1] "Timeloop: A tool for evaluating and designing deep learning accelerators." [Online]. Available: https://timeloop.csail.mit.edu
[2] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," Engineering, vol. 6, no. 3, pp. 264–274, 2020.
[3] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE journal of solid-state circuits, vol. 52, no. 1, pp. 127–138, 2016.
[4] S.-C. Kao and T. Krishna, "Gamma: Automating the hw mapping of dnn models on accelerators via genetic algorithm," ICCAD '20: Proceedings of the 39th International Conference on Computer-Aided Design, vol. 44, pp. 1–9, 2020.
[5] R. Machupalli, M. Hossain, and M. Mandal, "Review of asic accelerators for deep neural network," Microprocessors and Microsystems, vol. 89, p. 104441, 2022.
[6] C. Sakhuja, Z. Shi, and C. Lin, "Leveraging domain information for the efficient automated design of deep learning accelerators," in 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2023, pp. 287–301.
[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[8] L. C.-H. S. R. Tuli, Shikhar and N. K. Jha, "Codebench: A neural architecture and hardware accelerator co-design framework," arXiv preprint arXiv:2212.03965, 2022.
[9] J. Wang, M. Ge, B. Ding, Q. Xu, S. Chen, and Y. Kang, "Nicepim: Design space exploration for processing-in-memory dnn accelerators with 3d-stacked-dram," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023.
[10] Y. Yu, Y. Li, S. Che, N. K. Jha, and W. Zhang, "Software-defined design space exploration for an efficient dnn accelerator architecture," IEEE Transactions on Computers, vol. 70, no. 1, pp. 45–56, 2020.