BraggHLS

Maksim Levental
University of Chicago
Email: test@test.tes

Ryan Chard
Ecole Superieure
Nantes, France
Email: second@second.fr

Kyle Chard
and Ian Foster
Star Academy
San Francisco, California 99999-9999
Telephone: (800) 555–5555
Fax: (888) 555–5555

***Abstract*—In many experiment-driven scientific domains, such as high-energy physics, material science, and cosmology, very high data rate experiments impose hard constraints on the corresponding data acquisition systems: collected data must either be indiscriminately stored for post-processing and analysis, thereby necessitating large storage capacity, or accurately filtered in real-time, thereby necessitating low latency execution. Deep neural networks, effective in many other filtering tasks, have not been widely employed in such data acquisition systems, due to design and deployment difficulties. This paper presents an open source, lightweight, compiler framework `BraggHLS`, based on high-level synthesis techniques, for translating high-level representations of deep neural networks to low-level representations, suitable for deployment to near-sensor devices such as field-programmable gate arrays. We present a case study and evaluation of `BraggHLS` on a deep neural network for Bragg peak detection in the context of high-energy diffraction microscopy. We show `BraggHLS` is able to produce an implementation with a throughput 4.7µs/sample, which is approximately a 5x improvement over the existing implementation.**

I. Introduction

Very high data rates are observed and large datasets are, consequently, generated across a broad range of experiments in scientific domains such as high-energy physics, material science, and cosmology. For example, in high-energy physics, the LHCb detector, at the CERN Large Hadron Collider, is tasked with observing the trajectories of particles produced in proton-proton collisions at a rate of 40 million per second (i.e., 40 MHz) [1]. At a packet size of approximately 50kB (per collision), this implies a data rate of approximately 2TB/s. Ultimately, in combination with the other detectors, the LHC processes approximately 100EB of data a year. In materials science, High-Energy Diffraction Microscopy (HEDM) techniques, which provide non-destructive characterization of structure and its evolution in a broad class of single-crystal and polycrystalline materials, can have collection rates approaching 1 MHz [2], with a corresponding packet size of 80kB. In cosmology, the Square Kilometre Array, a radio telescope projected to be completed in 2024 and to be operational by 2027 [3], will sustain data rates in excess of 10 TB/s [4].

Naturally, for high data rate experiments, directly storing and distributing such large quantities of data to the associated research communities for further analysis is cost prohibitive. Thus, either compression (in the case of storage and transmission) or outright filtering is employed, i.e., only a small fraction of the most "interesting" data is selected at time of collection, with the remainder being permanently discarded. In this work we focus on the filtering approach. Note, that the tradeoff made in employing filtering should be clear: reduced storage at the cost of stronger latency constraints (on the filtering mechanisms). In addition, the risk of discarding significant data introduces accuracy (of the filtering mechanisms) as a critical new dimension of the data acquisition systems. Typically, these filtering mechanisms consist either of physics based models [5] or machine learning models [6]; in either case maximally efficient and effective use of the target hardware platform is tantamount to accuracy. Irrespective of the type of technique employed, almost universally, for the lowest latency constraint use cases (sub-microsecond), the technique is deployed to either field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs) [7]. The reason for this is only FPGAs and ASICs are flexible enough to satisfy the latency constraints for a wide range of techniques. Here we focus on FPGAs.

Deep neural networks (DNNs), a particular type of machine learning model, have been shown to be effective in many scientific and commercial domains due to their "representational capacity", i.e., a capacity to (approximately) represent diverse sets of mappings [8]. DNNs "learn" to represent a mapping over the course of "training", wherein they are iteratively applied to sample data while a "learning rule" periodically updates the parameters (weights) parameterize the DNN. In recent years they have been investigated for near real-time scientific use cases [9], [10], [11] but their use in the lowest latency use cases has been very limited [7]. The reasons for this are threefold:

1) DNNs, by virtue of being deep, are resource intensive, in terms of both memory (for the weights) and compute (floating point arithmetic), thereby preventing their deployment to FPGAs.
2) Graphics Processing Units (GPUs), the conventional target platforms for DNNs, until very recently, have not been performant enough for these very high data rate, very low latency, use cases (due to their low clock speeds and low peripheral bandwidth [12]).
3) DNNs are (typically) defined, trained, and distributed using high-level frameworks (such as PyTorch [13], TensorFlow [14], MXNet [15]), which abstract all implementation details from the user, thereby making porta-

bility of existing models (to e.g., FPGA) nigh impossible and reuse weights difficult.

These above three barriers demand of a solution that can simultaneously translate a high-level representation of a DNN to a low-level representation, suitable for deployment to FPGA, while optimizing resource usage. In general, the task of *lowering* high-level representations of programs to lower-level representations (potentially, all the way to an architecture specific representation) is the domain of a compiler. Correspondingly, the task of *synthesizing* a program into a *register-transfer level* (RTL) *design*, rendered in a *hardware description language* (HDL), is the domain of high-level synthesis (HLS) [16]. While several such HLS tools exist [17], [18], [19], it turns out they struggle to effectively perform the necessary optimizations (despite, often, bundling robust optimizing compilers).

Recently, deep learning compilers (such as TVM [20], MLIR [21], and Glow [22]) have demonstrated the ability to dramatically improve inference latencies [23], training times [24], and memory usage [25] of DNNs. These compilers function by extracting intermediate-level representations (IRs) of the DNNs, from the representations produced by the frameworks, and performing various optimizations on those IRs (such as kernel fusion [26], vectorization [27], and memory planning [25]). The highly optimized IR is then used to generate code for various target architectures. Given the successes of these compilers, it's natural to wonder whether they can adapted to the task of sufficiently optimizing DNNs such that they might synthesized to RTL for deployment to FPGA (and thus suitable for the lowest latency use cases).

In this paper, we present `BraggHLS`[1], an open source, lightweight, compiler and HLS framework which can lower DNNs defined as PyTorch models to FPGA implementations. `BraggHLS` uses a combination of compiler and HLS techniques to compile the entire DNN into a *synchronous* module/circuit/design, thereby eliminating all synchronization resource overheads and achieving ultra-low latency. `BraggHLS` is general and supports a wide range of DNN layer types, and thus a wide range of DNNs, but we evaluate it on a DNN designed for identifying Bragg diffraction peaks. In summary our specific contributions include:

1) We discuss the challenges faced by a compiler and HLS tool in attempting to lower DNNs to ultra-low latency designs, including runtime costs incurred during design space exploration, challenges meeting resource and timing constraints during synthesis, placement, and routing.
2) We describe and implement a compiler framework, `BraggHLS`, which can effectively transform unoptimized, hardware-agnostic PyTorch models into ultra-low latency RTL designs suitable for deployment to Xilinx FPGAs. `BraggHLS` is thoroughly tested, open source, and available at https://github.com/makslevental/bragghls/.
3) We show that designs generated by `BraggHLS` achieve lower latency than Xilinx's state-of-the-art commercial HLS tool (Vitis HLS) for a variety of DNN layer types. In particular we show that `BraggHLS` can produce synthesizable designs that meeting placement, routing, and timing constraints where Vitis HLS cannot.

The rest of this paper is organized as follows: Section II reviews key concepts from compilers, scheduling, and FPGA synthesis, placement, and routing. Section III describes the `BraggHLS` compiler and HLS framework in detail. Section IV evaluates `BraggHLS`'s resource efficiency, scalability, and competitiveness with designs generated by Vitis HLS. Finally, Section V concludes with a summary, and related and future work.

[1]https://github.com/makslevental/bragghls

II. Background

III. `BraggHLS` compiler and HLS framework

IV. Evaluation

V. Conclusion

References

[1] V. Gligorov, "Real-time data analysis at the lhc: present and future," in *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, ser. Proceedings of Machine Learning Research, G. Cowan, C. Germain, I. Guyon, B. KÃ©gl, and D. Rousseau, Eds., vol. 42. Montreal, Canada: PMLR, 13 Dec 2015, pp. 1–18. [Online]. Available: https://proceedings.mlr.press/v42/glig14.html

[2] M. Hammer, K. Yoshii, and A. Miceli, "Strategies for on-chip digital data compression for x-ray pixel detectors," *Journal of Instrumentation*, vol. 16, no. 01, pp. P01 025–P01 025, jan 2021. [Online]. Available: https://doi.org/10.1088%2F1748-0221%2F16%2F01%2Fp01025

[3] J. McMullin, P. Diamond, M. Caiazzo, A. Casson, T. Cheetham, P. Dewdney, R. Laing, B. Lewis, A. Schinckel, L. Stringhetti *et al.*, "The square kilometre array project update," in *Ground-based and Airborne Telescopes IX*, vol. 12182. SPIE, 2022, pp. 263–271.

[4] K. Grainge, B. Alachkar, S. Amy, D. Barbosa, M. Bommineni, P. Boven, R. Braddock, J. Davis, P. Diwakar, V. Francis *et al.*, "Square kilometre array: The radio telescope of the xxi century," *Astronomy reports*, vol. 61, no. 4, pp. 288–296, 2017.

[5] "Comparison of particle selection algorithms for the LHCb Upgrade," 2020. [Online]. Available: https://cds.cern.ch/record/2746789

[6] V. V. Gligorov and M. Williams, "Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree," *Journal of Instrumentation*, vol. 8, no. 02, pp. P02 013–P02 013, feb 2013. [Online]. Available: https://doi.org/10.1088%2F1748-0221%2F8%2F02%2Fp02013

[7] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran, and Z. Wu, "Fast inference of deep neural networks in FPGAs for particle physics," *Journal of Instrumentation*, vol. 13, no. 07, pp. P07 027–P07 027, jul 2018. [Online]. Available: https://doi.org/10.1088%2F1748-0221%2F13%2F07%2Fp07027

[8] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[9] Z. Liu, T. Bicer, R. Kettimuthu, and I. Foster, "Deep learning accelerated light source experiments," in *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*. IEEE, 2019, pp. 20–28.

[10] R. M. Patton, J. T. Johnston, S. R. Young, C. D. Schuman, D. D. March, T. E. Potok, D. C. Rose, S.-H. Lim, T. P. Karnowski, M. A. Ziatdinov *et al.*, "167-pflops deep learning for electron microscopy: from learning physics to atomic manipulation," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 638–648.

[11] Y. Liu, K. P. Kelley, H. Funakubo, S. V. Kalinin, and M. Ziatdinov, "Exploring physics of ferroelectric domain walls in real time: deep learning enabled scanning probe microscopy," *Advanced Science*, p. 2203957, 2022.

[12] R. Aaij, J. Albrecht, M. Belous, P. Billoir, T. Boettcher, A. Brea Rodríguez, D. Vom Bruch, D. Cámpora Pérez, A. Casais Vidal, D. Craik *et al.*, "Allen: A high-level trigger on gpus for lhcb," *Computing and Software for big Science*, vol. 4, no. 1, pp. 1–11, 2020.

[13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016. [Online]. Available: https://arxiv.org/abs/1603.04467

[15] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015. [Online]. Available: https://arxiv.org/abs/1512.01274

[16] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, "A survey and evaluation of fpga high-level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1591–1604, 2016.

[17] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, T. Czajkowski, S. D. Brown, and J. H. Anderson, "Legup: An open-source high-level synthesis tool for fpga-based processor/accelerator systems," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 2, sep 2013. [Online]. Available: https://doi.org/10.1145/2514740

[18] Z. Zhang, Y. Fan, W. Jiang, G. Han, C. Yang, and J. Cong, *AutoPilot: A Platform-Based ESL Synthesis System*. Dordrecht: Springer Netherlands, 2008, pp. 99–112. [Online]. Available: https://doi.org/10.1007/978-1-4020-8588-8_6

[19] F. Ferrandi, V. G. Castellana, S. Curzel, P. Fezzardi, M. Fiorito, M. Lattuada, M. Minutoli, C. Pilato, and A. Tumeo, "Invited: Bambu: an open-source research framework for the high-level synthesis of complex applications," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1327–1330.

[20] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "{TVM}: An automated {End-to-End} optimizing compiler for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 578–594.

[21] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko, "Mlir: A compiler infrastructure for the end of moore's law," 2020. [Online]. Available: https://arxiv.org/abs/2002.11054

[22] N. Rotem, J. Fix, S. Abdulrasool, G. Catron, S. Deng, R. Dzhabarov, N. Gibson, J. Hegeman, M. Lele, R. Levenstein, J. Montgomery, B. Maher, S. Nadathur, J. Olesen, J. Park, A. Rakhov, M. Smelyanskiy, and M. Wang, "Glow: Graph lowering compiler techniques for neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1805.00907

[23] Y. Liu, Y. Wang, R. Yu, M. Li, V. Sharma, and Y. Wang, "Optimizing cnn model inference on cpus," 2018. [Online]. Available: https://arxiv.org/abs/1809.02697

[24] S. Zheng, R. Chen, Y. Jin, A. Wei, B. Wu, X. Li, S. Yan, and Y. Liang, "Neoflow: A flexible framework for enabling efficient compilation for high performance dnn training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 3220–3232, nov 2022.

[25] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," 2016. [Online]. Available: https://arxiv.org/abs/1604.06174

[26] A. Ashari, S. Tatikonda, M. Boehm, B. Reinwald, K. Campbell, J. Keenleyside, and P. Sadayappan, "On optimizing machine learning workloads via kernel fusion," vol. 50, no. 8, pp. 173–182, jan 2015. [Online]. Available: https://doi.org/10.1145/2858788.2688521

[27] S. Maleki, Y. Gao, M. J. Garzar, T. Wong, D. A. Padua *et al.*, "An evaluation of vectorizing compilers," in *2011 International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 2011, pp. 372–382.