EE 7722—GPU Microarchitecture

EE 7722—GPU Microarchitecture

URL: `https://www.ece.lsu.edu/gp/`.

Offered by:

David M. Koppelman
3316R P.F. Taylor Hall, 578-5482, koppel@ece.lsu.edu, `https://www.ece.lsu.edu/koppel`
Office Hours: Monday - Friday, 14:00-15:00.

Prerequisites By Topic:

- Computer architecture.
- C++ and machine language programming.

Text

Papers, technical reports, etc. (Available online.)

Course Objectives

- Understand low-level *accelerator* organizations.
- Understand their application to for machine learning and scientific workloads.
- Be able to fine-tune accelerator codes based on this low-level knowledge.
- Understand issues and ideas being discussed for the next generation of accelerators . . .
  . . . including tensor processing (machine-learning, inference, training) accelerators.

Course Topics

- Performance Limiters (Floating Point, Bandwidth, etc.)
- Parallelism Fundamentals
- Distinction between processor types: CPU, many-core CPU, (many-thread) GPU, FPGA, large spatial processors, large systolic array.
- GPU Architecture and CUDA Programming
- GPU Microarchitecture, FP Units, Tensor Cores, etc.
- Tuning based on machine instruction analysis.
- Tensor Processing Units, machine learning accelerators.

Graded Material

Midterm Exam, 35%

Fifty minutes in-class.

Final Exam, 35%

Two hours in-class.

Yes, it's cumulative.

Homework, 30%

Written and computer assignments.

Lowest grade or unsubmitted assignment dropped.

Course Usefulness

Material in Course Needed For:

- Those designing the next generation of accelerator chips.
- Those writing back-ends and optimizers for machine-learning frameworks.
- Those writing low-energy machine learning programs.
- Those writing high-performance scientific programs.
- Those writing high-performance graphics programs.
- Those interested in future computer architectures.
- Compiler writers.

Course Resources

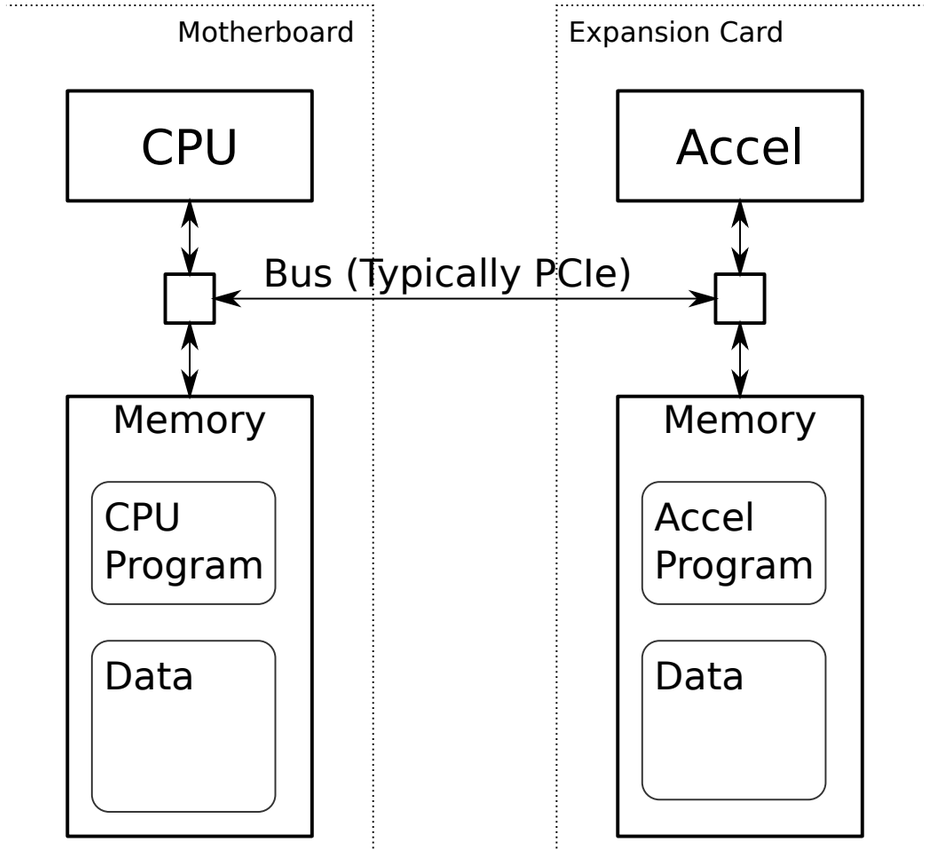
- Slides and other material via `https://www.ece.lsu.edu/gp/`
- Code examples in git repository `git://dmk.ece.lsu.edu/gp`
- Web site also has homework assignments, exams, grades, and other material.
- Announcements are on course home page and available as a Web (RSS) Feed.

Accelerator:

A specialized processor designed to work alongside a general-purpose processor (CPU).

Work is split between the different devices . . .

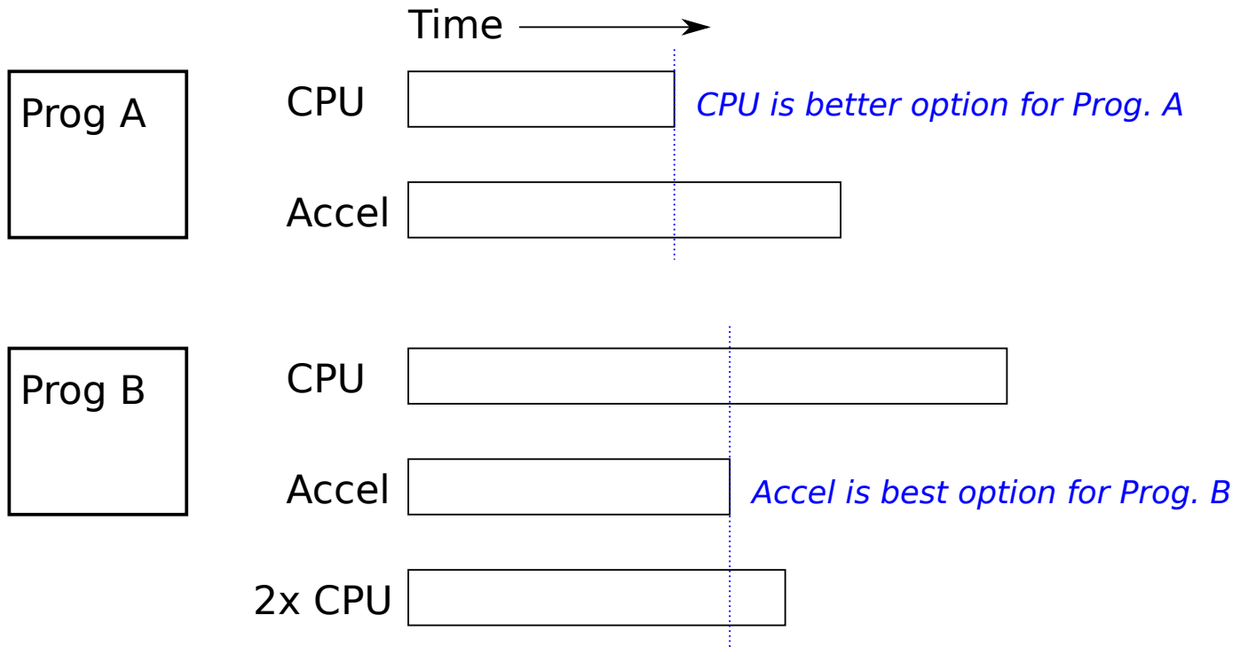
. . . each does what it's best at.



-7

Accelerator Benefit

Accelerator can execute some code faster than CPU.



Program B is faster on the accelerator . . .
. . . but for Program A the accelerator hurts performance.

For Program B, two CPUs almost as good as one CPU plus one accelerator.

EE 7722 Lecture Transparency. Formatted 12:18, 22 October 2026 from lsli00-TeXize.
-8

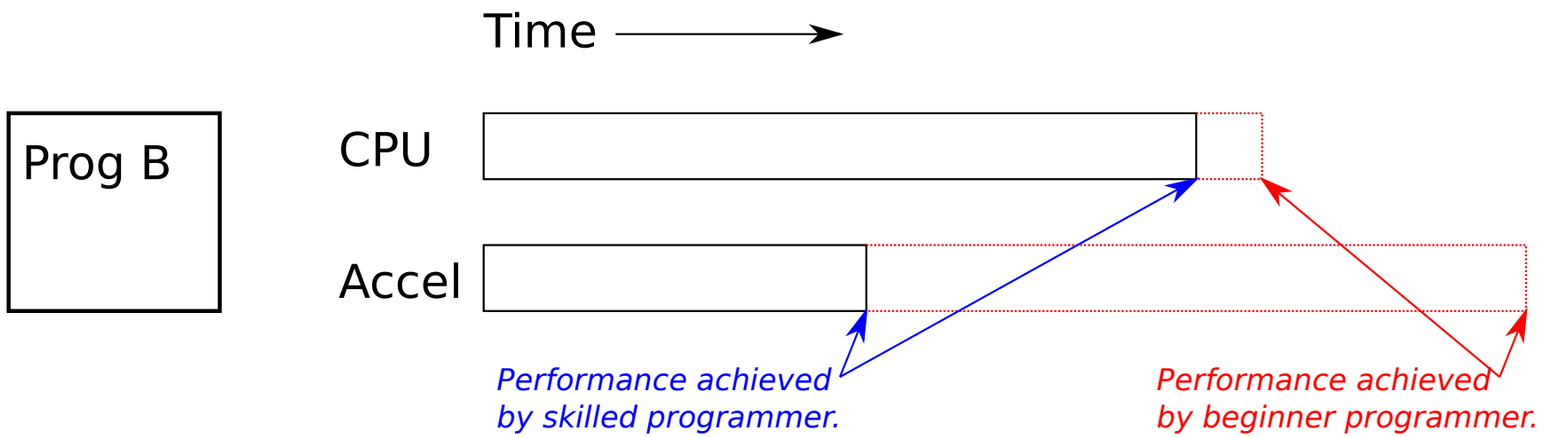
Accelerator Programming Challenge

CPUs are more forgiving.

Not paying attention to things may cost a few percent.

Accelerators must be programmed with care.

Not paying attention to things can result in much worse performance.



Common Accelerator Types

- *GPU (Graphics Processing Unit)* — *E.g.*, NVIDIA Blackwell B100 [11]
- *Large Systolic Array Accelerator* — *E.g.*, Google Trillium TPU v6e [4]
- *Mesh-Connected Medium Core Accelerator* — *E.g.*, Meta MTIA 2i, [1], Tesla DOJO [14]
- *FPGA Accelerator* — *E.g.*, Nallatech PCIe-180
- *Processor-In-Memory Digital Accelerator* — Commercial UPMEM [3] — nearing commercialization Samsung Aquabolt-XL [3, 8] — research [12].
- *Processor-In-Memory ReRAM Analog Accelerator* — Research area [13].
- *Many-Core Processor* — *E.g.*, Intel Xeon Phi (Discontinued)

Example:

> "I remember way back when our computer was taking four days to compute the 48-hour forecast, so we bought a system with 3 accelerators: an NVIDIA K20c, a Xeon Phi, and a Nallatech board, all of these were given work that would have been performed by the general-purpose CPU, an Intel i7."

Typical Acceleratable Code

Input:

`w`, a $n_{oi} \times n_{ii}$ matrix.

`i`, a $n_b \times n_{ii}$ matrix.

`o`, a $n_b \times n_{oi}$ matrix.

```
for ( int b=0; b<nb; b++ )
  for ( int oi=0; oi<noi; oi++ )
    for ( int ii=0; ii<nii; ii++ )
      o[b,oi] += w[oi,ii] * i[b,ii]
```

Number of multiply/add (madd) operations: $n_b\, n_{oi}\, n_{ii}$.

The value of `nb`, `noi`, and `nii` can drastically change the nature of the computational demands of this code.

GPU (Graphics Processing Unit):
A processor designed to execute a class of programs that includes 3D graphics, scientific computation, and machine learning using a large number of threads.

A Brief History

GPUs originally designed *only* for 3D graphics [2].

Large economies of scale made them cheap.

Resourceful scientific users used GPUs by disguising their work as 3D graphics [15].

GPU makers started supporting scientific and other non-graphical work.

GPUs evolved into a second kind of processor, with 3D graphics just one application.

Current Status

Always used in systems needing 3D graphics, from cell phones to workstations.

Typically used for ML *training* workloads.

Often used for scientific computation, but acceptance had been slow due to difficulty of porting code.

GPU Product Examples

- NVIDIA RTX 6000 Pro — High-end GPU for home use.
- NVIDIA Blackwell B100 — High-end GPU for non-graphical computing...
  ... including half-precision FP support for deep learning applications.
- AMD Radeon R9 — High-end GPU for home use.

Large Systolic Array Accelerators:
An accelerator that uses a systolic arrays that each take up a major fraction of the chip area. A typical minimum size might be a 64×64 array (which consists of 4096 multiply/accumulate (MAC) units).

Key Characteristics

These are very efficient since the inputs a MAC unit are connected to nearby MAC units . . .
. . . reducing communication and intermediate storage costs.

Efficiency is lost if problem size smaller than array.

Commercial Example

Google's TPUs. TPU v1 has one 256×256 array [7], TPU v2 through v5 each had several 128×128 arrays [6, 10, 5], and v6e (Trillium) has two 256×256 arrays.

Brief History

Talked about at least since late 1970s [9]. . .
. . . with many proposed designs [16].

FPGA (Field-Programmable Gate Array):
A chip that can be programmed to mimic a specific piece of digital hardware.

Suppose you have a design for a digital circuit.

You can fabricate an *ASIC* version . . .
. . . and wait months for delivery . . .
. . . and pay $100k for the first chip, and pennies for the second, etc.

Or you can download the design into an FPGA. . .
. . . and have your part in seconds . . .
. . . and pay a few dollars for the first chip, and for the second, etc.

ASICs v. FPGAs

ASICs used when:

- A large number of chips are needed.
- The design is not expected to change over the product lifetime.
- The highest performance is needed.
- The design is very large.

  Examples: cell phone signal processor, Bitcoin mining rig.

FPGAs used when:

- The design may change frequently or require updates.
- A part is needed quickly, perhaps as a prototype.

  Example: WiFi router.

FPGA Accelerator:
An accelerator which creates custom hardware, at the time a program is run, which should execute parts of the code more efficiently than any general-purpose device.

Using an FPGA Accelerator

Programming an FPGA accelerator can be more like hardware design.

Methods to simplify programming are still in the research stage.

They work best for specialized applications . . .
. . . rather than as an accelerator in a general-use facility.

Few high-performance computing facilities have FPGA accelerators.

Accelerators and this Course

GPU Accelerators

These will be covered because of their maturity and success.

Tensor Processing / Machine Learning Accelerators

These will be covered because of current interest and rapid advancement.

FPGA Accelerators

Not covered because they are very different than GPUs and many-cores ...
... and because so far they are used in much more specialized settings.

Those interested in FPGA acceleration might look for high-level synthesis courses here.

GPU, CPU Raw Performance Numbers

GPU: NVIDIA K20x (2012)
SP FP, 3950 GFLOPS; DP FP, 1310 GFLOPS; Off-Chip Bandwidth 250 GB/s

GPU: NVIDIA P100 (2016)
SP FP, 4761 GFLOPS; DP FP, 2381 GFLOPS; Off-Chip Bandwidth 549 GB/s

GPU: NVIDIA V100 (2018)
SP FP, 7066 GFLOPS; DP FP, 3533 GFLOPS; Off-Chip Bandwidth 898 GB/s

Many Core: Xeon Phi 7120 P
SP FP, 2416 GFLOPS; DP FP 1208 GFLOPS, Off-Chip Bandwidth 352 GB/s

CPU: Intel Xeon E7-8870. (2011) (Ten cores, 2.4 GHz, 128b vector insn.)
SP FP, 96 GFLOPS; DP FP 48 GFLOPS, Off-Chip Bandwidth ? GB/s

CPU: Intel Xeon W-2195. (2017) (18 cores, 2.3 GHz, 512b vector insn.)
SP FP, 1325 GFLOPS; DP FP 662 GFLOPS, Off-Chip Bandwidth 85.3 GB/s

GPU v. CPU

GPU can execute floating-point operations at a higher rate . . .
. . . because it has more floating point units.

GPU can read and write memory at a higher rate.

CPUs are easier to program.

CPUs can run certain programs faster.

CPU/GPU Similarities

Both run programs.

Machine languages are similar.

Both have instructions for integer and FP operations.

High-end chips are roughly same area, and draw same power.

GPU Incidental (and diminishing) Differences

Special hardware for *texture fetch and filtering* operations.

Special hardware for *interpolation*.

Trend is less specialized hardware.

GPU Important Performance Spec Differences

GPUs can do more FP operations per second.

GPUs can transfer more data per second.

GPU Programmability Differences

Extensive tuning required to achieve performance.

GPUs perform poorly on certain problems, regardless of tuning.

GPUs succeeded where many failed: Establishing a new computer architecture.

CPUs today share similar architecture.

Very good example of a one-size-fits-all device . . .
. . . same chip in business, scientific computation, server, etc.

Exceptions are minor: embedded.

In the past specialized architectures could be successful.

Cray supercomputers.

But in most cases, different or specialized architectures failed:

Database machines.

LISP machines.

Specialized architectures failed because:

They were not that much faster.

Were very expensive to develop.

Market was small and so engineering big part of cost.

But two specialized architectures *have* succeeded: GPUs and TPUs.

GPUs: First, their birth.

3D computer graphics is compute intensive. (Think frame rate.) Need.

Graphics computation is well structured.

Facilitates development of specialized processor.

Amount of code relatively small.

Can be isolated in libraries.

Large market: Home computers, game consoles.

Can amortize development costs.

Their evolution to non-graphical use.

Computational characteristics of 3D graphics code shared by other types of programs.

GPUs could easily be modified to support non-graphical use.

References:

[1] Coburn, J., Tang, C., Asal, S. A., Agrawal, N., Chinta, R., Dixit, H., Dodds, B., Dwarakapuram, S., Firoozshahian, A., Gao, C., Gondkar, K., Graf, T., Hu, J., Huang, J., Hughes, S., Hutchin, A., Jakka, B., Chen, G. J., Kalyanaraman, I., Kamath, A., Kansal, P., Kazi, E., Levenstein, R., Maddury, M., Mastro, A., Medaiyese, S., Modi, P., Montgomery, J., Nadathur, S., Nagpal, A., Narasimha, A., Naumov, M., Ozer, E., Park, J., Ramani, P., Reddy, H., Reiss, D., Roy, D., Sekar, S., Sharma, A., Shetty, P., Sukumaran-Rajam, A., Tal, E., Tsai, M., Varshini, S., Wareing, R., Wu, O., Xie, X., Yang, J., Yu, H., Zargar, T., Zeng, Z., Zhang, F., Matthews, A., Jiao, X., Zhang, J., Menage, E., Stokke, T. E., and Sourouri, M. Meta's second generation AI chip: Model-chip co-design and productionization experiences. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture* (New York, NY, USA, 2025), ISCA '25, Association for Computing Machinery, p. 16891702. https://doi.org/10.1145/3695053.3731409.

[2] Dally, W. J., Keckler, S. W., and Kirk, D. B. Evolution of the graphics processing unit (GPU). *IEEE Micro 41*, 06 (Nov 2021), 42–51. http://dx.doi.org/10.1109/MM.2021.3113475.

[3] Gomez-Luna, J., Hajj, I. E., Fernandez, I., Giannoula, C., Oliveira, G. F., and Mutlu, O. Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory architecture, 2022. https://arxiv.org/abs/2105.03814.

[4] Jouppi, N., Kurian, G., Li, S., Ma, P., Nagarajan, R., Nai, L., Patil, N., Subramanian, S., Swing, A., Towles, B., Young, C., Zhou, X., Zhou, Z., and Patterson, D. A. TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*

(New York, NY, USA, 2023), ISCA '23, Association for Computing Machinery. https://doi.org/10.1145/3579371.3589350.

[5] Jouppi, N. P., Hyun Yoon, D., Ashcraft, M., Gottscho, M., Jablin, T. B., Kurian, G., Laudon, J., Li, S., Ma, P., Ma, X., Norrie, T., Patil, N., Prasad, S., Young, C., Zhou, Z., and Patterson, D. Ten lessons from three generations shaped googles TPUv4i : Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (2021), pp. 1–14. http://dx.doi.org/10.1109/ISCA52012.2021.00010.

[6] Jouppi, N. P., Yoon, D. H., Kurian, G., Li, S., Patil, N., Laudon, J., Young, C., and Patterson, D. A domain-specific supercomputer for training deep neural networks. *Commun. ACM 63*, 7 (June 2020), 6778. https://doi.org/10.1145/3360307.

[7] 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.

[8] Kim, B., Cha, S., Park, S., Lee, J., Lee, S., Kang, S.-h., So, J., Kim, K., Jung, J., Lee, J.-G., Lee, S., Paik, Y., Kim, H., Kim, J.-S., Lee, W.-J., Ro, Y., Cho, Y., Kim, J. H., Song, J., Yu, J., Lee, S., Cho, J., and Sohn, K. The Breakthrough Memory Solutions for Improved Performance on LLM Inference. *IEEE Micro 44*, 03 (May 2024), 40–48. https://doi.ieeecomputersociety.org/10.1109/MM.2024.3375352.

[9] Kung. Why systolic architectures? *Computer 15*, 1 (1982), 37–46. http://dx.doi.org/10.1109/MC.1982.1653825.

[10] Norrie, T., Patil, N., Yoon, D. H., Kurian, G., Li, S., Laudon, J., Young, C., Jouppi, N., and Patterson, D. The design process for Google's training chips: TPUv2 and TPUv3. *IEEE Micro 41*, 2 (2021), 56–63. http://dx.doi.org/10.1109/MM.2021.3058217.

[11] NVIDIA Corporation. NVIDIA Blackwell architecture technical brief. Tech. rep., NVIDIA Corporation, 2024.

[12] Park, J., Choi, J., Kyung, K., Kim, M. J., Kwon, Y., Kim, N. S., and Ahn, J. H. AttAcc! unleashing the power of PIM for batched transformer-based generative model inference. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (New York, NY, USA, 2024), ASPLOS '24, Association for Computing Machinery, p. 103119. https://doi.org/10.1145/3620665.3640422.

[13] Sun, Z., Pedretti, G., Mannocci, P., Ambrosi, E., Bricalli, A., and Ielmini, D. Time complexity of in-memory solution of linear systems. *IEEE Transactions on Electron Devices 67*, 7 (2020), 2945–2951. http://dx.doi.org/10.1109/TED.2020.2992435.

[14] Talpes, E., Sarma, D. D., Williams, D., Arora, S., Kunjan, T., Floering, B., Jalote, A., Hsiong, C., Poorna, C., Samant, V., Sicilia, J., Nivarti, A. K., Ramachandran, R., Fischer, T., Herzberg, B., McGee, B., Venkataramanan, G., and Banon, P. The microarchitecture of DOJO, Tesla's exa-scale computer. *IEEE Micro 43*, 3 (2023), 31–39. http://dx.doi.org/10.1109/MM.2023.3258906.

[15] Thompson, C. J., Hahn, S., and Oskin, M. Using modern graphics architectures for general-purpose computing: a framework and analysis. In *Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture* (Washington, DC, USA, 2002), MICRO 35, IEEE Computer Society Press, p. 306317.

[16] Xu, R., Ma, S., Guo, Y., and Li, D. A survey of design and optimization for systolic array-based DNN accelerators. *ACM Comput. Surv. 56*, 1 (Aug 2023). https://doi.org/10.1145/3604802.