Demo: First Demonstration of Real-Time Photonic-Electronic DNN Acceleration on SmartNICs

Zhizhen Zhong Mingran Yang Jay Lang Dirk Englund Manya Ghobadi

Massachusetts Institute of Technology

ABSTRACT

We demonstrate Lightning, a reconfigurable photonic-electronic deep learning smartNIC that serves real-time inference requests at 4.055 GHz compute frequency. To do so, Lightning uses a novel datapath to feed traffic from the NIC into its photonic computing cores without incurring digital data movement bottlenecks. Lightning achieves this by employing a *reconfigurable count-action* abstraction, which decouples the compute control plane from the data plane. The count-action abstraction counts the number of operations for each computation task in the Directed Acyclic Graph (DAG). It then triggers the execution of the next task(s) as soon as the previous task is finished without interrupting the dataflow. Our prototype shows that Lightning achieves 99.25% photonic MAC accuracy. When serving real-time inference requests, Lightning accelerates the end-to-end inference latency of the LeNet DNN by 9.4× and 6.6× compared to Nvidia P4 and A100 GPUs, respectively.

CCS CONCEPTS

• **Hardware** → **Networking hardware**; **Emerging optical and photonic technologies**; *Hardware accelerators*; *Reconfigurable logic applications*; • **Computer systems organization** → **Optical computing**; *Real-time system architecture*; *Analog computers*;

KEYWORDS

Photonic computing, Network hardware design, Computer architecture, Real-time AI, Machine learning inference

ACM Reference Format:

Zhizhen Zhong, Mingran Yang, Jay Lang, Dirk Englund, Manya Ghobadi. 2023. Demo: First Demonstration of Real-Time Photonic-Electronic DNN Acceleration on SmartNICs. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10, 2023, New York, NY, USA.* ACM, New York City, NY, USA, 3 pages. https://doi.org/10.1145/3603269.3610842

1 INTRODUCTION

Photonic computing is an emerging area with the potential to revolutionize the landscape of computing by leveraging lightwaves and optical devices to execute computations with high speed and energy efficiency within the analog domain [7, 9, 10, 13, 17]. Recent research efforts have showcased the potential of performing photonic computation at 100+ GHz frequency while consuming 40 atto Joules per operation [8–11, 14–16]. The Lightning paper [18] showed that data movement is a significant bottleneck in today's photonic computing approaches and proposed a reconfigurable photonic-electronic smartNIC with fast and energy-efficient *photonic vector dot product cores*. The key enabler in Lightning is a novel *reconfigurable count-action* abstraction that decouples the control and data planes of inference requests. This abstraction enables the datapath to keep track of the computation DAG of each inference request without interrupting the dataflow in and out of photonic multiplication cores [18].

In this demonstration, we demonstrate Lightning by serving real-time inference queries, performing photonic multiply-accumulate (MAC) operations at a frequency of 4.055 GHz. This demo provides an in-depth view of Lightning's hardware implementation. We further demonstrate a versatile benchmarking tool that allows developers from the SIGCOMM community to integrate photonic computing into their code base using a Python API. Our demonstration source code and documentation are publicly available at https://lightning.mit.edu.

2 IMPLEMENTATION

This section describes Lightning's implementation details complementary to our full paper [18]. Figure 1 illustrates our demonstration setup to serve real-time inference queries using photonic computing cores, and Figure 2 is a picture of our prototype.

DAC/ADC configurations. In Lightning, we use Xilinx's RF data converter IP (ADC/DAC) [5]. We configure the data converter IP to work at 4.055 Giga Samples per second (GS/s) data rate with a reference clock rate of 253.44 MHz. We enable four DACs and two ADCs, with multi-tile synchronization enabled. We set the DACs and ADCs to work in Nyquist Zone 1 bypassing all its internal mixers. We also configure the output current of the DACs to be 32 mA to provide the maximum output voltage. We configure the ADC to work in the DC coupling state to preserve the DC signal in the output of our photodetector.

100 Gbps CMAC configurations. We configure 100 Gbps CMAC IP [3] to support four 25.7812 Gbps lanes to be compatible with the ZCU111 board [6]. The CMAC core is configured to run at 156.25 MHz clock. For the receive port, we configure the maximum and minimum packet length to be 64 bytes and 9600 bytes, respectively. We also set the user interface of the CMAC to be AXI-Stream to be compatible with our datapath implementation.

DRAM configurations. We configure the DRAM [4] that is directly connected to our FPGA to work under a clock frequency of 333.25 MHz. Our DRAM supports 2666 Mega transactions per second, and each transaction is 64 bits, creating a data rate at ≈170 Gbps. We store the DNN parameters in DRAM, and stream the parameter data from the DRAM to our datapath through our DRAM controller. At every datapath clock cycle, the DRAM controller reads

ACM SIGCOMM '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0236-5/23/09…$15.00
https://doi.org/10.1145/3603269.3610842

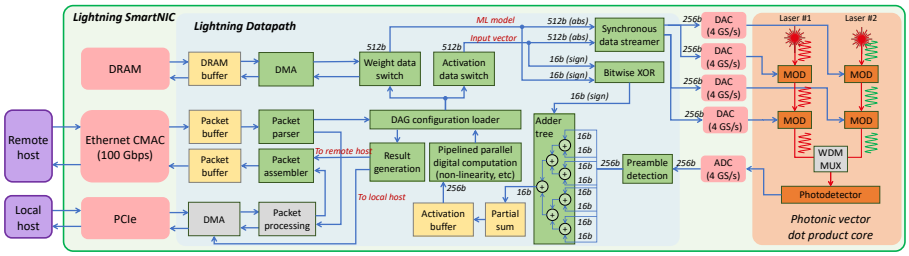

Figure 1: Lightning datapath implementation [18].

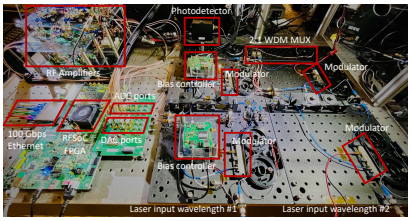

Figure 2: Photo of our demonstration setup [18].

Figure 3: Lightning's photonic computing developer kit.

512 bits DNN parameter data and forwards it to other modules in our datapath. Note that our current prototype use two DACs running at 4.055 GS/s to stream DNN parameters, creating a data rate at 4.055 GS/s × 8b/S × 2 = 64.88 Gbps. Upgrading to higher number of photonic parallelism or frequency requires higher data rates. Therefore, it will require more DRAMs or high-bandwidth memory (HBM).

3 DEMONSTRATION

We first demonstrate the performance of Lightning serving real-time inference requests using its 100 Gbps network interface. Then, we present an open-source photonic computing developer kit with a Python API to lower the barrier to entry.

Inference serving with Lightning. We first demonstrate how to serve inference queries with Lightning. The end-to-end latency reflects the time from the moment the request arrives until the moment the inference result packet leaves the system. We perform real-time inference on a handwriting recognition DNN called LeNet-300-100 [12] using the Lightning smartNIC. Our experimental results show Lightning can accelerate inference serve time by 9.4× and 6.6× compared to Nvidia P4 and A100 GPUs, respectively.

Assembling "plug-and-play" open-source photonic developer kit. To lower the barrier of entry into photonic computing, we demo the detailed assembly steps to build a developer kit using the same devices as in our prototype (shown in Figure 3). The developer kit is designed to be "plug-and-play" such that a developer without deep knowledge in photonics and FPGAs can get started easily.

Benckmarking photonic MAC operations. We then demonstrate our customized Python API on Lightning's FPGA using PYNQ [2] and QICK [1] libraries to provide programmers an easy-to-use, direct interface for photonic computing. We use this Python API to perform photonic MAC operations on unsigned fixed-point 8-bit numbers, demonstrating 99.25% photonic MAC accuracy. This Python API enables developers to prototype and benchmark applications (e.g., video encoding, data encryption, etc.) beyond machine learning, without the need to construct a fully-fledged hardware design. Listing 1 shows a code snippet of using Python API to perform a MAC operation using Lightning.

```
from lightning import LightningCompute, LightningConfig
from qick import QickSoC
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm

# Configure the firmware of the FPGA
soc = QickSoC()
soccfg = soc
ltng = LightningCompute(soccfg, soc, LightningConfig)

# automatic calibration on four modulators
fit_mod_0 = ltng.calibration(target_mod = 0)
fit_mod_1 = ltng.calibration(target_mod = 1)
fit_mod_2 = ltng.calibration(target_mod = 2)
fit_mod_3 = ltng.calibration(target_mod = 3)
fittings = [fit_mod_0, fit_mod_1, fit_mod_2, fit_mod_3]

# configure input values
LightningConfig["mod_0"] = input_0
LightningConfig["mod_1"] = input_1
LightningConfig["mod_2"] = input_2
LightningConfig["mod_3"] = input_3

# check if the photonic computing result agrees with the truth
truth = input_0 * input_1 + input_2 * input_3
ltng_result = ltng.photonic_computing(fittings, LightningConfig)
print("The truth is:", truth, "Lightning result is:", ltng_result)
```

Listing 1: Performing photonic MAC with our Python API.

Automatic calibration of the photonic components. To perform computation in the photonic domain where digital data is represented in light intensity, it is important to derive a transfer function that encodes a digital number into the modulator input voltage and then decodes the measured photodetector output voltage back into the digital domain. We demonstrate the process to calibrate the system before performing photonic computing on the Lightning system. We believe this automatic calibration functionality facilitates a broader range of community members to be more involved with photonic computing.

Acknowledgement. The authors are supported by DARPA FastNICs 4202290027, Air Force AI Accelerator, ARPA-E ENLITENED PINE DE-AR0000843, NSF CNS-2008624, NSF SHF-2107244, NSF ASCENT-2023468, NSF CAREER-2144766, NSF PPoSS-2217099, NSF CNS-2211382, NSF FuSe-TG-2235466, Sloan fellowship FG-2022-18504, the U.S. Army Research Office through the Institute for Soldier Nanotechnologies (ISN) (W911NF-18-2-0048), and the NSF Center for Quantum Networks.

REFERENCES

[1] 2022. QICK: Quantum Instrumentation Control Kit . https://github.com/openquantumhardware/qick.

[2] 2022. RFSOC-PYNQ . http://www.rfsoc-pynq.io/.

[3] 2022. UltraScale+ Devices Integrated 100G Ethernet Subsystem v3.1. https://docs.xilinx.com/v/u/en-US/pg203-cmac-usplus.

[4] 2022. UltraScale™ architecture-based FPGAs Memory IP core. https://www.xilinx.com/content/dam/xilinx/support/documents/ip_documentation/ultrascale_memory_ip/v1_4/pg150-ultrascale-memory-ip.pdf.

[5] 2022. Zynq UltraScale+ RFSoC RF Data Converter v2.6 Gen 1/2/3 LogiCORE IP Product Guide. https://docs.xilinx.com/v/u/en-US/pg269-rf-data-converter.

[6] 2022. Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit. https://www.xilinx.com/products/boards-and-kits/zcu111.html,.

[7] Devin Coldewey. 2021. Lightmatter's photonic AI ambitions light up an $80M B round. https://techcrunch.com/2021/05/06/lightmatters-photonic-ai-ambitions-light-up-an-80m-b-round/.

[8] J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. S. Raja, J. Liu, C. D. Wright, A. Sebastian, T. J. Kippenberg, W. H. P. Pernice, and H. Bhaskaran. 2021. Parallel convolutional processing using an integrated photonic tensor core. *Nature* 589, 7840 (2021), 52–58. https://doi.org/10.1038/s41586-020-03070-1

[9] Heedong Goh and Andrea Alù. 2022. Nonlocal Scatterer for Compact Wave-Based Analog Computing. *Phys. Rev. Lett.* 128 (Feb 2022), 073201. Issue 7. https://doi.org/10.1103/PhysRevLett.128.073201

[10] Ryan Hamerly, Liane Bernstein, Alexander Sludds, Marin Soljačić, and Dirk Englund. 2019. Large-scale optical neural networks based on photoelectric multiplication. *Physical Review X* 9, 2 (2019), 021032.

[11] Philip Jacobson, Mizuki Shirao, Kerry Yu, Guan-Lin Su, and Ming C. Wu. 2022. Hybrid Convolutional Optoelectronic Reservoir Computing for Image Recognition. *Journal of Lightwave Technology* 40, 3 (2022), 692–699. https://doi.org/10.1109/JLT.2021.3124520

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[13] Microsoft. 2023. Project AIM (Analog Iterative Machine). https://www.microsoft.com/en-us/research/project/aim/.

[14] Alexander Sludds, Saumil Bandyopadhyay, Zaijun Chen, Zhizhen Zhong, Jared Cochrane, Liane Bernstein, Darius Bunandar, P Ben Dixon, Scott Hamilton, Matthew Streshinsky, Ari Novack, Tom Baehr-Jones, Michael Hochberg, Manya Ghobadi, Ryan Hamerly, and Dirk Englund. 2022. Delocalized Photonic Deep Learning on the Internet's Edge. *Science* 378, 6617 (2022), 270–276. https://doi.org/10.1126/science.abq8271

[15] Alexander Sludds, Ryan Hamerly, Saumil Bandyopadhyay, Zhizhen Zhong, Zaijun Chen, Liane Bernstein, Manya Ghobadi, and Dirk Englund. 2022. Demonstration of WDM-Enabled Ultralow-Energy Photonic Edge Computing, In Optical Fiber Communication Conference (OFC) 2022. *Optical Fiber Communication Conference (OFC) 2022*, Th3A.3. https://doi.org/10.1364/OFC.2022.Th3A.3

[16] Xingyuan Xu, Mengxi Tan, Bill Corcoran, Jiayang Wu, Andreas Boes, Thach G Nguyen, Sai T Chu, Brent E Little, Damien G Hicks, Roberto Morandotti, et al. 2021. 11 TOPS photonic convolutional accelerator for optical neural networks. *Nature* 589, 7840 (2021), 44–51.

[17] Javier Yanes. 2020. Optical Computing: Solving Problems at the Speed of Light. https://www.bbvaopenmind.com/en/technology/future/optical-computing-solving-problems-at-the-speed-of-light/.

[18] Zhizhen Zhong, Mingran Yang, Jay Lang, Christian Williams, Liam Kronman, Alexander Sludds, Homa Esfahanizadeh, Dirk Englund, and Manya Ghobadi. 2023. Lightning: A Reconfigurable Photonic-Electronic SmartNIC for Fast and Energy-Efficient Inference. In *Proceedings of the 2023 ACM SIGCOMM 2023 Conference.*

A ENVIRONMENT AND RESOURCES NEEDED

To enable the live demo, we will package our experimental setup and all essential devices (shown in Figure 2) and move it to the conference site. Except the devices we can move from MIT, we also need the following resources.

- 110V standard power outlet.
- office table with size at least 48 inches × 24 inches.
- internet access with wired Ethernet.
- two computer monitors (we can provide ourselves).
- a desktop computer running Windows or Ubuntu system (we can provide ourselves).