Experimental Insights from the Rogues Gallery

Jeffrey S. Young*, Jason Riedy†, Thomas M. Conte*, Vivek Sarkar*, Prasanth Chatarasi*, Sriseshan Srikanth*
*School of Computer Science
Georgia Institute of Technology
Atlanta, GA
{jyoung9,conte,vsarkar,cprasanth,seshan}@gatech.edu
†School of Computational Science and Engineering
jason.riedy@cc.gatech.edu

I. Introduction to the Rogues Gallery

The Rogues Gallery is a new deployment for understanding next-generation hardware with a focus on unorthodox and uncommon technologies. This testbed project was initiated in 2017 in response to Rebooting Computing efforts and initiatives. The Gallery's focus is to acquire new and unique hardware (the rogues) from vendors, research labs, and start-ups and to make this hardware widely available to students, faculty, and industry collaborators within a managed data center environment. By exposing students and researchers to this set of unique hardware, we hope to foster cross-cutting discussions about hardware designs that will drive future performance improvements in computing long after the Moore's Law era of cheap transistors ends. We have defined an initial vision of the infrastructure and driving engineering challenges for such a testbed in a separate document, so here we present highlights of the first one to two years of *post-Moore* era research with the Rogues Gallery and give an indication of where we see future growth for this testbed and related efforts.

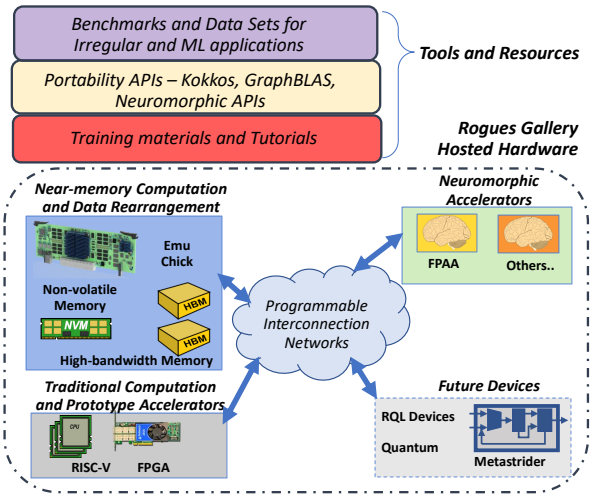

Fig. 1. High-level overview of the CRNCH Rogues Gallery

The important research insights from this testbed (so far) are the following:

- **Post-Moore computing research is built on a hierarchy of novel architectures with varying levels of software support.** Of the current rogues, the Emu has the most "developed" software stack, but this means that it still lacks critical libraries, compiler optimizations, and APIs for interfacing with target applications. Neuromorphic, quantum, and other more revolutionary architectures still lack much of the needed compiler and library infrastructure to map meaningful applications.
- **We can demonstrate small "wins" from technologies like the Emu Chick or Hybrid Memory Cube but these results indicate the need for deeper study and more focused engineering.** Our initial Emu results indicate a suitability for graph analysis, but we cannot yet run large graphs or at scale due to load imbalance and thread migration issues. Likewise, the packet-oriented nature of the Hybrid Memory Cube (HMC) interface allows for more focused latency and BW tradeoff studies, but applications research using a combined HMC and FPGA platform is limited by the lack of well-supported high-level synthesis (HLS) tools for FPGAs and similar programmable devices. Both of these early prototypes indicate a focused need for more research studies and engineering efforts.
- **Tools, runtimes, and benchmarking are an important challenge for post-Moore research that needs more research investment as an enabling infrastructure.** Tools like the Habanero programming model and runtime [1], Kokkos portability API [2], and benchmarking are "high-effort" investments. However, their respective benefits can justify the cost, if we can build sufficient communities around platforms and tools that are convincing to the research community and funding agencies.
- **Education and engagement with undergrads will be important elements to push post-Moore research into the forefront for students.** There is currently a strong push from industry and funding agencies for students and researchers to tackle machine learning topics on current-day architectures. We need to engage students in post-Moore computing efforts by bridging from current machine learning to topics that require future architectures. We detail one example of how we are working to map from machine learning applications to post-Moore architectures in Section VIII.

II. The Rogues Gallery

As Figure 1 demonstrates, the CRNCH Rogues Gallery initially encompasses five different areas of research: 1) **Traditional computation and accelerators** that include enabling technologies like FPGAs, RISC-V, and related HPC technologies that provide a good bridge to post-Moore hardware. 2) **Near-memory computation and data rearrangement technologies** like High-bandwidth memories (HBM), NVM (as typified by newer 3D XPoint devices), near-storage accelerators, and prototypes like the Emu Chick [3], [4]. 3) **Neuromorphic accelerators**, which extend our conception of machine learning accelerators to brain-inspired computing with lower power requirements and real-time processing power. 4) **Revolutionary Architectures** that include sparse accelerators like our local project, Strider, emerging quantum efforts, reversible, and thermodynamic computing. 5) **Tools and resources** that are critical to map today's algorithms and applications of interest to novel architectures, especially those that are truly revolutionary. We present examples of each of these categories from the Rogues Gallery in the following sections.

III. Traditional Architectures - Reconfigurable Computing

Field Programmable Gate Arrays are not post-Moore architectures in themselves, but they can enable the evaluation of novel memory systems and accelerators like the neuromorphic DANNA implementation [5]. Recent work with the Rogues Gallery has focused on compilation tools for FPGAs and evaluating memory systems like the Hybrid Memory Cube [6], which has been made available for researchers as part of a Micron-supported Xilinx FPGA and HMC module called the AC-510. This platform has enabled several recent publications [7], [8] that are focused on low-level characterizations of this type of 3D stacked memory. As Figure 2 shows, the packet-oriented nature of HMC requests allows for packaging multiple DRAM read requests into a larger payload that can then be sent to the memory with related trade-offs in latency or improved overall bandwidth utilization of the serialized link. While Micron has recently moved away from future HMC designs, the similarities of HMC accesses with traditional networking payloads and the abstracted nature of the HMC's DRAM interface may play a role in future post-Moore successors to High-bandwidth memory (HBM) including those enabled by packet-oriented technologies like Gen-Z [9].

More traditional FPGA research results have included the usage of Intel and Xilinx platforms for research into runtimes and compiler frameworks. Most notably, standard Arria 10 boards provide an opportunity to evaluate joint industry and academic research on the Temporal to Spatial (T2S) programming system [10] that allows for decoupling of different spatial optimizations and orientations from the functionality of an accelerated kernel like a matrix-multiply operation. Results from this study show that dense tensor algebra kernels (GEMM, Tensor-Times Matrix, etc.) can easily be mapped to an Intel FPGA platform with a 1.76X speedup

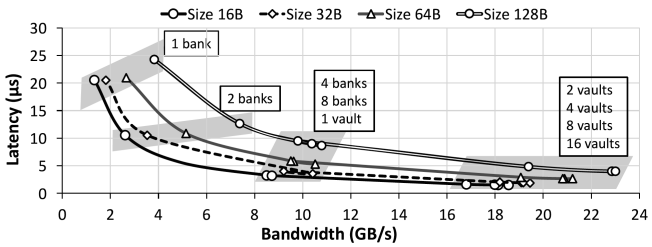

Fig. 2. Characterization of the AC-510 FPGA+HMC platform shows clear trade-offs for latency versus bandwidth as packet sizes change. [8]

for GEMM over an NDRange-based OpenCL implementation and up to 88% of a hand-tuned HDL implementation.

Ongoing work in this space is looking at mapping machine learning and high-performance computing kernels to reconfigurable platforms using compiler optimizations and systems like T2S and related research projects like OpenARC [11] and Chisel to reduce the overhead of testing new accelerators from that required with hand-tuned HDL or OpenCL codes.

IV. Near-memory Computation

We envision that near-memory processing architectures will increasingly migrate computation near high-bandwidth and non-volatile memories as in previous work [12], [13], [14], [15] and will be supplemented by an evolving set of near-storage processors like the recently announced Western Digital SweRV RISC-V core. In our local testbed, we have been focused on characterizing and evaluating optimizations for current near-memory systems like HMC and novel architectures like the Emu system. The Emu architecture focuses on improving random-access bandwidth scalability by migrating lightweight *Gossamer* threads to data and by emphasizing fine-grained memory access. We leave the more detailed description of this system to other related work [3], [16], [4], [17]. Our currently evolving prototype contains eight nodes that are each organized into eight gossamer cores (also referred to as a nodelet) and that are clocked at 175 MHz with DDR4 DRAM memory that is clocked at 1600MHz. The current generation of Emu system, the Emu Chick, includes one stationary processor for each of the eight nodelets contained within a node. The Emu Chick provides an interesting use case as an intermediate-level rogue in that it has a basic Linux-based OS on the stationary cores, a detailed single-threaded simulator, and support for Cilk++ and some basic Python functionality. While there are many missing libraries for the Emu architecture, students have used the testbed to perform characterization experiments, run graph-oriented benchmarks, and test basic machine learning algorithms using a SciKit-Learn interface.

Detailed microbenchmark characterization for this architecture has been explored in [18], so we present two sample results that demonstrate initial "wins" for irregular algorithm design. Figure 3 shows a custom pointer chasing benchmark that was designed to test the irregular access capabilities of the Emu system in a more fine-grained manner than the

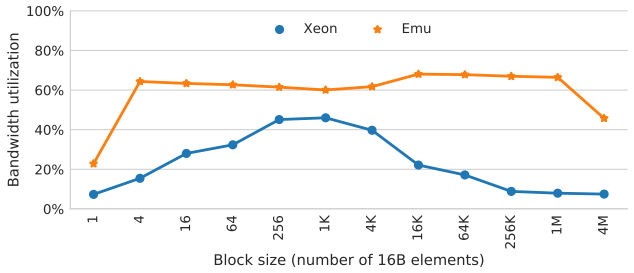

Fig. 3. Bandwidth utilization of pointer chasing, compared between Sandy Bridge Xeon and Emu (64 nodelets) [18]

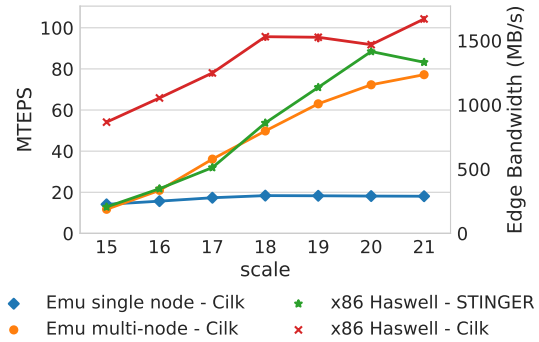

Fig. 4. Streaming BFS on the Emu Chick versus x86 Cilk and STINGER for balanced Erdös-Rényi) graphs. Scale: $\log_2$ of the number of vertices. [19]

traditional GUPS benchmark. Blocks of data in a linked list are increasingly shuffled to simulate different levels of random access, and the CPU system (a Sandy Bridge box) exhibits a somewhat expected bandwidth utilization trend line that follows the caching and paging trend lines up to 16K sized blocks. At the same time, the Emu is able to achieve a relatively consistent bandwidth utilization result for all but the smallest and largest block sizes.

The results in Figure 4 compare Emu single node (8 cores) and multi-node (64 cores) with a Haswell server system running a streaming BFS application using either Cilk or a recent state-of-the-art STINGER [20] implementation. Despite the differences in clock rate and memory speed for the Emu and CPU systems, the Chick is able to achieve similar rates of edge traversal for the input balanced graphs. This result indicates that at least for small amounts of data, the cacheless Emu architecture can perform as well as a heavily optimized CPU-based system. Note that the Emu currently does not perform nearly as well for unbalanced RMAT-style graphs due to load imbalance issues with the current code implementation.

Interestingly, both of these positive results were somewhat limited in their impact for the Emu architecture due to a later analysis [18] that showed that the current Chick prototype is compute-bound due to the underlying FPGA boards that are used to construct its fabric and gossamer core architecture. This further analysis demonstrates that **we need to be careful in extrapolating larger trends for post-Moore architectures from small "wins" that show positive progress for key applications**.

A. *Compiler-oriented optimizations*

Even though the Emu system is designed to improve the performance of data-sensitive workloads exhibiting weak-locality, the thread migrations across nodelets can also hamper the performance if overhead from the thread migration dominates the benefits achieved through the migration. Also, frequent thread creations can hamper the performance because thread creation on a remote nodelet results in migrating to that remote nodelet and spawning locally [21], [16].

Recent work using the Rogues Gallery Emu Chick system [22] measures the total number of migrations arising from a set of popular graph applications such as Bellman-Ford's algorithm for the single-source shortest path problem (SSSP), triangle counting, and conductance, using an in-house simulation environment of the Emu prototype. These measurements showed that many unnecessary and redundant thread migrations occurred because of naive algorithmic implementations. To address these unnecessary thread migrations, two high-level compiler optimizations, loop fusion and edge flipping, and one low-level compiler transformation that incorporates hardware support for remote atomic updates were explored to address overheads arising from thread migration, creation, and atomic operations.

A preliminary evaluation of these compiler transformations was performed by manually applying them for SSSP, triangle counting, and conductance over a set of RMAT graphs from Graph500. RMAT graphs (edges of these graphs are generated randomly with a power-law distribution) were used with the number of vertices running from 2^6 to 2^{14} (scale 6 through 14) and average degree 16 as specified by Graph500 [23]. These graphs were generated using the utilities present in the STINGER framework [24]. This evaluation targeted a single node of the Emu hardware prototype, and summary results are presented for the combined set of optimizations on the three types of input graphs in Figure 5. Applying all of thes optimizations results in an overall geometric mean reduction of 22.12% in thread migrations [22]. This preliminary study clearly motivates further exploration of the implementation of automatic compiler transformations to alleviate these thread migration overheads arising from running graph applications on the Emu system.

V. Neuromorphic

The Field Programmable Analog Array (FPAA) [25] implements a combined analog and digital board that can implement many analog and neuromorphic architectures [26], [27]. The FPAA combines a 2D array of ultra-low-power floating-gate analog plus digital blocks with a driving 16-bit MSP430 microprocessor. The exploration and development platform consists of a USB-attached board with multiple analog input ports (Figure 6).

For a device manufactured on a 350nm CMOS process, the FPAA provides a very low-power platform for neuromorphic, machine learning, and classification tasks. For example, the

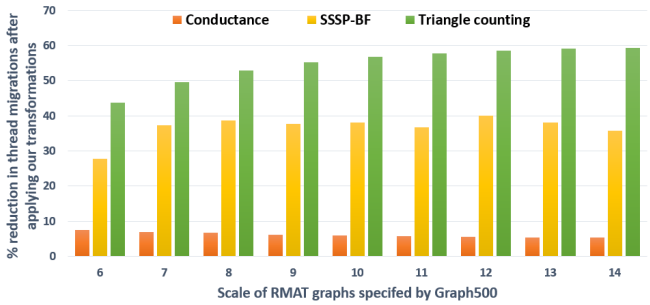

Fig. 5. Experimental evaluation (i.e., %reduction in thread migrations) of three graph algorithms (Conductance, SSSP-BF and Triangle counting) on the RMAT graphs from scales 6 to 14 specified by Graph500. Transformations applied on the algorithms: Conductance/SSSP-BF/Triangle counting: (Node fusion)/(Edge flipping and Remote updates)/ (Remote updates). The evaluation is done a single node (8 nodelets) of the Emu system [22].

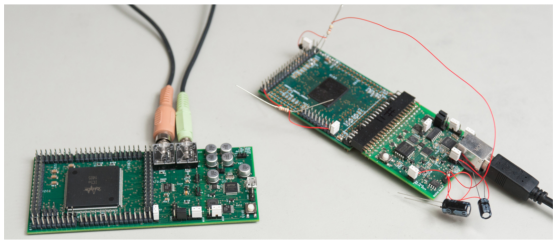

Fig. 6. The Field Programmable Analog Array (FPAA)

FPAA uses 23 μW to recognize the word "dark" in the TIMIT database [26]. Similarly, classifying acoustic signals from a knee joint requires 15.29 μW [27]. Both of these types of computations are performed in real time, so these power savings translate directly to energy savings and help to provide justification for further research in mixed analog and digital hardware for multiple applications.

Our current focus for research with the FPAA platform is focused on extending existing classification and spiking neural network examples to support higher level neuromorphic APIs like the TennLab exploratory framework [28] and Sandia National Lab's N2A [29]. Currently the FPAA is primarily programmed with a graphical interface using Scilab and XCos tools, so we initially need to provide a mapping from high-level neuromorphic APIs to the modular and macro-block programming environment that is used to create mixed analog and digital designs on the FPAA. While this initial engineering effort may not be groundbreaking in terms of post-Moore research, it is a critical effort to bring this novel hardware to a wider community of potential users.

VI. Revolutionary Architectures

The final category of post-Moore devices are those that are revolutionary in terms of upending the traditional von Neumann model for computing. While we group reversible, thermodynamic, quantum, and some specialized accelerators in this category, we currently are investigating two types of revolutionary architectures, sparse non-von Neumann accelerators and quantum computing enabling software and algorithms.

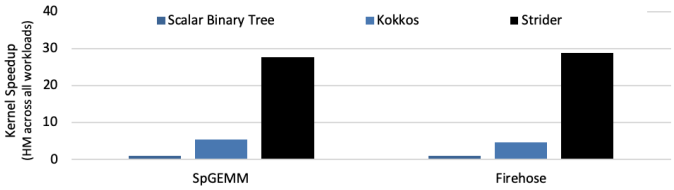

Fig. 7. Intelligent and dynamic memory-centric marshalling of sparse data with the Strider architecture significantly improves DRAM performance and results in over a magnitude of speedup for sparse reductions [31], [32].

A. Accelerators for Sparse Data

Conventional architectures are inefficient for a class of emerging workloads that exhibit low locality of reference. These sparse data applications suffer from a high cache miss ratio and expose DRAM (main memory) latency to the critical path. Recent research [30], [31], [32] has revealed that DRAM-centric sparse representations, algorithms and architectures have the potential of improving the state-of-the-art performance and energy by an order of magnitude. In particular, these works have focused on the sparse reduction kernel, wherein an associative operation is applied to the values of two or more key-value pairs that share the same key. For example, in generalized sparse matrix-matrix multiplication (SpGEMM, used in a variety of domains including graph analytics and HPC [33]), the accumulation phase is nothing but a sparse reduction problem where the "keys" are the matrix indices of the non-zero partial products ("values") that need to be summed (reduction operator).

Figure 7 demonstrates the tremendous speedup achieved due to using such an approach with our custom architecture, Strider, when compared with software binary tree and Kokkos-based approaches to the same problem. Speedups of up to 30x are measured as an average across a variety of SpGEMM-based workloads as well as for Firehose, a cybersecurity benchmark that models soft real time events. While these results are obtained via implementations in cycle-accurate simulators, FPGA synthesis of the underlying hardware operations [34] suggests that we could pursue a future integration of such accelerators for irregular data streams in the reconfigurable arena of the Rogues Gallery (Section III).

B. Quantum

The Rogues Gallery focuses on the programming and systems level of quantum computing while leveraging both local [35] and remote "Noisy Intermediate-Scale Quantum" devices (NISQ) [36] systems for intermediate results. The strenuous physical requirements for hosting quantum systems limits their physical deployment, so we focus more on easing intermediate access steps for algorithm development and helping users access available remote testbeds.

Programs for NISQ devices must adapt to the noise and errors [37]. This currently requires multiple runs along with adapting both algorithms and hardware mappings. The Rogues Gallery framework already recognizes systems where stability is an issue and extends directly to quantum sampling.

Beyond all of the systems issues lie the educational aspects. Quantum computing requires many mental pivots from classical computing. Section VIII briefly outlines our undergraduate efforts for quantum platforms leveraging existing frameworks like Qiskit[38] and building on existing training programs like NSF's EPicQC quantum computing tutorials and summer schools.

VII. Software Resources

Making new hardware available is a key component of a successful post-Moore testbed, but we argue that the tools and benchmarks made available are critical to build an interested community around specific architectures.

A. Benchmarking from Micro to Macro

Initial research into post-Moore architectures has resulted in a rich set of microbenchmarks that can be used and modified to test new architectures. For instance, characterizations of the Emu Chick system at our institute has led to a novel pointer-chasing benchmark that can be run on CPU-based systems as well as on the Emu platform, streaming graph benchmarks, sparse-matrix vector microkernels, and related graph and sparse microbenchmarks from related characterizations [17]. Likewise, local tensor libraries like ParTI [39] have variants for traditional systems and the Emu system and are supported by growing, collaborative datasets like the FROSTT tensor repository [40].

Figure 8 shows results for traditional HPC-oriented systems with the newly released Spatter benchmark suite [41], which enables users to test variations of gather and scatter operations and to characterize and evaluate patterns of indexed accesses that occur commonly in sparse algorithms and HPC applications. These results show the correlation between the sparsity of accesses, where the sparsity of 16 is equivalent to accessing one out of every 16 elements, and the effective bandwidth of these accesses when compared to a more traditional STREAM benchmark. While some current systems like the KNL struggle with reasonably sparse gather operations, newer GPUs like the V100 and AVX-enabled CPUs like the Skylake can effectively perform gather operations with high amounts of performance.

Our current focus is on extending Spatter by adding new backends for post-Moore architectures, including the cacheless Emu architecture, FPGAs with OpenCL, Metastrider, and neuromorphic accelerators. We anticipate that these cacheless architectures may be much better than traditional CPU- and GPU-based platforms at performing these types of critical memory accesses.

B. Portability Tools and Libraries

In addition to benchmarking efforts, there are early efforts to explore both the Kokkos performance portability API [42] and the GraphBLAS [43] on the Emu Chick in collaboration with other labs and universities. These engineering efforts contribute to the overall community around a tool and enable a larger number of applications to be tested on a novel post-Moore architecture. For instance, the ongoing development of an Emu Cilk backend for Kokkos would enable the execution of many of the Mantevo mini-apps and other Kokkos-enabled applications that have been ported by researchers at Sandia National Labs and other DoE-associated laboratories. It is highly unlikely that graduate students could port this same set of codes to the Emu Chick by generating custom Cilk kernels for a set of mini-apps.

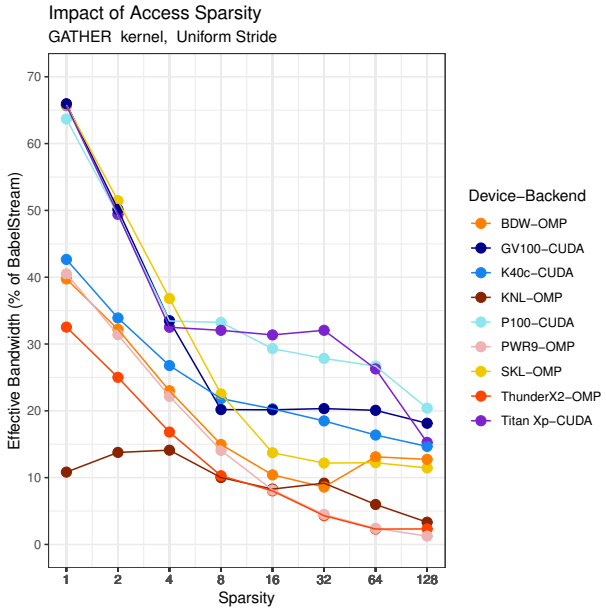

Fig. 8. Evaluating CUDA and OpenMP Gather with the Spatter benchmark suite. [41]

Likewise, we look to build on related external libraries and APIs for neuromorphic computing to enable a wider audience for neuromorphic hardware like the FPAA. With a limited number of researchers in the post-Moore computing space and many high-effort infrastructure pieces to build, we should look to grow our community's research impact by building on and improving existing tools and frameworks where at all possible.

VIII. Education and Outreach

One important aspect of the Rogues Gallery is ensuring that knowledge of novel platforms grows beyond the labs from which they come. Many new ideas are tested only locally, educating a only few graduate students about a platform's benefits and drawbacks. Making the platforms more widely available combined with providing tutorial and educational material should accelerate novel computing research. We also organize sessions at scientific computing conferences that bring together potential users, the Rogues Gallery, and other test beds.

A. Training and Demonstration

In April 2018 we held a neuromorphic workshop combining invited speakers with *hands-on* FPAA demonstrations. The hands-on portion required physical attendance. Participants organized into small groups, each of which had a FPAA to set up and use.

Our more recent approach focuses on remote access to our platforms. Rogues Gallery recently set up a JupyterLab[1] environment for tutorials. These allow active participation for those who want it, and a pre-made demonstration for those who would rather listen and read. The pre-made portion also is useful when the venue's network is unreliable. Even without making new Jupyter kernels, the environment permits editing code, running compilers, visualizing results, and even shell access. This proved useful for tutorials run at ASPLOS 2019 and PEARC 2019.

Tutorial and presentation material is made available through our institutional website and a separate Gitlab website[2]. Soon we hope these tutorials can be run entirely remotely, which would ease access from classes.

B. Education and Undergraduate Research

The authors had experience with novel platforms as undergraduates back when computing was not as homogeneous as it is now. We understand the benefits of exposing undergraduates to more than the few dominant platforms. One benefit to hosting the Rogues Gallery at a university is integrating the Gallery into undergraduate education.

Our initial undergraduate research class, part of the Vertically Integrated Projects program [44], provides novel architecture access for early computing and engineering students. The students are engaged and self-organized into groups focused on the FPAA, the Emu Chick, and integrating quantum simulators like Qiskit [38]. The students interested in quantum computing have learned initial skills from tutorials provided by efforts like the NSF EPiQC program and have identified the need for both diagrammatic and programmatic expression of quantum algorithms. These undergraduate students are *excited* to use novel architectures and provide feedback on how their preparation does or does not match the expectations in many platforms' documentation. Additionally, many of our microbenchmarking and initial engineering efforts provide a good entry point for undergraduate students looking to get involved with post-Moore computing research.

We also provide access to external graduate students at multiple universities. So far the access has been mainly limited to final projects for parallel computing classes, but we anticipate providing more materials that can extend the reach of post-Moore computing into traditional classwork.

IX. Looking to the Future of post-Moore Testbeds

We provide an overview of recent research results that have been enabled by the Rogues Gallery testbed to provide a sampling of what near-term and future research for post-Moore computing is ongoing at our institute. As with other larger architectural testbeds like CENATE [45] at Pacific Northwest National Lab, ExCL[3] at Oak Ridge National Lab, and Sandia HAAPS[4], we are focused on different aspects of the post-Moore computing landscape with a slightly different but overlapping audience of industry collaborators, researchers, and students.

To make sense of this broad research landscape we have attempted to organize the emerging candidate architectures into a basic classification scheme focused on how "near-term" a potential architecture might be. We propose a few initial lessons learned that reinforce our common need to investigate new post-Moore candidate architectures at a deep level while supplementing them with a research organizaton built around tools, benchmarking, and common APIs. Finally, we propose that efforts like our own undergraduate post-Moore course and large-scale education and tutorial efforts like those supported by NSF (e.g., EPiQC) are critical to growing a thriving community around post-Moore architecture testbeds.

[1]https://jupyterlab.readthedocs.io/en/stable/

[2]https://gitlab.com/crnch-rg

[3]https://excl.ornl.gov/

[4]https://www.sandia.gov/asc/computational_systems/HAAPS.html

Acknowledgments

This work is supported in part by Micron's and Intel's hardware donations, the NSF XScala project (ACI-1339745), the NSF SuperSTARLU project (OAC-1710371), and IARPA. We also thank Eric Hein, Janice McMahon and the rest of the team at Emu for their assistance in setting up and supporting the Emu Chick system for our users. Special thanks goes to Dr. Jennifer Hasler for her support of the FPAA and related training and tutorial materials for students.

Sandia National Laboratory supports the Rogues Gallery VIP undergraduate research class, and parts of this research have been funded through the Laboratory Directed Research and Development (LDRD) program at Sandia. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

[1] D. Majeti and V. Sarkar, "Heterogeneous Habanero-C (H2C): A portable programming model for heterogeneous processors," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, May 2015, pp. 708–717.

[2] H. Carter Edwards, C. R. Trott, and D. Sunderland, "Kokkos," *J. Parallel Distrib. Comput.*, vol. 74, no. 12, pp. 3202–3216, Dec. 2014.

[3] T. Dysart, P. Kogge, M. Deneroff, E. Bovell, P. Briggs, J. Brockman, K. Jacobsen, Y. Juan, S. Kuntz, and R. Lethin, "Highly scalable near memory processing with migrating threads on the Emu system architecture," in *Workshop on Irregular Applications: Architecture and Algorithms (IA3)*. IEEE, 2016, pp. 2–9.

[4] J. S. Young, E. Hein, S. Eswar, P. Lavin, J. Li, J. Riedy, R. Vuduc, and T. M. Conte, "A microbenchmark characterization of the emu chick," *CoRR*, 2018.

[5] J. P. Mitchell, M. E. Dean, G. R. Bruer, J. S. Plank, and G. S. Rose, "DANNA 2: Dynamic adaptive neural network arrays," in *Proceedings of the International Conference on Neuromorphic Systems*, ser. ICONS '18. New York, NY, USA: ACM, 2018, pp. 10:1–10:6.

[6] *Hybrid Memory Cube Specification 1.0*, Hybrid Memory Cube Consortium, 2013.

[7] R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, and H. Kim, "Demystifying the characteristics of 3D-stacked memories: A case study for Hybrid Memory Cube," in *2017 IEEE International Symposium on Workload Characterization (IISWC)*, Oct 2017, pp. 66–75.

[8] R. Hadidi, B. Asgari, J. Young, B. A. Mudassar, K. Garg, T. Krishna, and H. Kim, "Performance implications of NoCs on 3D-stacked memories: Insights from the Hybrid Memory Cube," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2018, pp. 99–108.

[9] G.-Z. Consortium, "Gen-Z overview (whitepaper)," *Gen-Z (online)*, 2016. [Online]. Available: http://genzconsortium.org/wp-content/uploads/2016/11/Gen-Z-Overview-V1.pdf

[10] N. Srivastava, H. Rong, P. Barua, G. Feng, H. Cao, Z. Zhang, D. Albonesi, V. Sarkar, W. Chen, P. Petersen *et al.*, "T2S-Tensor: Productively generating high-performance spatial hardware for dense tensor computations," in *The 27th IEEE International Symposium On Field-Programmable Custom Computing Machines (FCCM)*, 2019.

[11] S. Lee, J. Kim, and J. S. Vetter, "OpenACC to FPGA: A framework for directive-based high-performance reconfigurable computing," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 544–554.

[12] S. Lloyd and M. Gokhale, "In-memory data rearrangement for irregular, data-intensive computing," *Computer*, vol. 48, no. 8, pp. 18–25, Aug 2015.

[13] J. C. Beard, "The sparse data reduction engine: Chopping sparse data one byte at a time," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '17. New York, NY, USA: ACM, 2017, pp. 34–48.

[14] M. Shantharam, K. Iwabuchi, P. Cicotti, L. Carrington, M. Gokhale, and R. Pearce, "Performance evaluation of scale-free graph algorithms in low latency non-volatile memory," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2017, pp. 1021–1028.

[15] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proceedings of the 53rd Annual Design Automation Conference*, ser. DAC '16. New York, NY, USA: ACM, 2016, pp. 173:1–173:6.

[16] E. Hein, T. Conte, J. S. Young, S. Eswar, J. Li, P. Lavin, R. Vuduc, and J. Riedy, "An initial characterization of the Emu Chick," in *The Eighth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES)*, May 2018.

[17] M. Belviranli, S. Lee, and J. S. Vetter, "Designing algorithms for the EMU migrating-threads-based architecture," *High Performance Extreme Computing Conference 2018*, 2018.

[18] J. Young, E. R. Hein, S. Eswar, P. Lavin, J. Li, E. J. Riedy, R. W. Vuduc, and T. Conte, "A microbenchmark characterization of the Emu Chick," *CoRR*, vol. abs/1809.07696, 2018.

[19] E. R. Hein, S. Eswar, A. Yasar, J. Li, J. S. Young, T. M. Conte, Ü. V. Çatalyürek, R. Vuduc, E. J. Riedy, and B. Uçar, "Programming strategies for irregular algorithms on the Emu Chick," *CoRR*, vol. abs/1901.02775, 2019.

[20] J. Riedy, H. Meyerhenke, D. A. Bader, D. Ediger, and T. G. Mattson, "Analysis of streaming social networks and graphs on multicore architectures," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012.

[21] P. M. Kogge and S. K. Kuntz, "A Case for Migrating Execution for Irregular Applications," in *Proceedings of the Seventh Workshop on Irregular Applications: Architectures and Algorithms*, ser. IA3'17. New York, NY, USA: ACM, 2017, pp. 6:1–6:8.

[22] P. Chatarasi and V. Sarkar, "A Preliminary Study of Compiler Transformations for Graph Applications on the Emu System," in *Proceedings of the Workshop on Memory Centric High Performance Computing*, ser. MCHPC'18. New York, NY, USA: ACM, 2018, pp. 37–44.

[23] D. A. Bader, J. Berry, S. Kahan, R. Murphy, E. J. Riedy, and J. Willcock, "Graph500 Benchmark 1 (search) Version 1.2," Graph500 Steering Committee, Tech. Rep., Sep. 2011.

[24] D. Ediger, R. McColl, J. Riedy, and D. A. Bader, "STINGER: High performance data structure for streaming graphs," in *2012 IEEE Conference on High Performance Extreme Computing*, Sept 2012, pp. 1–5.

[25] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R. Wunderlich, S. Nease, and S. Ramakrishnan, "A programmable and configurable mixed-mode FPAA SoC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2253–2261, June 2016.

[26] J. Hasler and H. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in Neuroscience*, vol. 7, p. 118, 2013.

[27] S. Shah, C. N. Teague, O. T. Inan, and J. Hasler, "A proof-of-concept classifier for acoustic signals from the knee joint on a FPAA," in *2016 IEEE SENSORS*, Oct. 2016, pp. 1–3.

[28] J. S. Plank, C. D. Schuman, G. Bruer, M. E. Dean, and G. S. Rose, "The TENNLab exploratory neuromorphic computing framework," *IEEE Letters of the Computer Society*, vol. 1, pp. 17–20, 2018.

[29] F. Rothganger, C. Warrender, D. Trumbo, and J. Aimone, "N2A: a computational tool for modeling from neurons to algorithms," *Frontiers in Neural Circuits*, vol. 8, p. 1, 2014.

[30] S. Srikanth, T. M. Conte, E. P. DeBenedictis, and J. Cook, "The superstrider architecture: Integrating logic and memory towards non-von neumann computing," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, Nov. 2017, pp. 1–8.

[31] E. P. DeBenedictis, J. Cook, S. Srikanth, and T. M. Conte, "Superstrider associative array architecture: Approved for unlimited unclassified release: Sand2017-7089 c," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017, pp. 1–7.

[32] S. Srikanth, A. Jain, J. Lennon, T. Conte, E. DeBenedictis, and J. Cook, "Metastrider: Architectures for scalable memory centric reduction of sparse data streams," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2019.

[33] Y. Nagasaka, S. Matsuoka, A. Azad, and A. Buluç, "High-performance sparse matrix-matrix products on intel knl and multicore architectures," *arXiv preprint arXiv:1804.01698*, 2018.

[34] A. Jain, S. Srikanth, E. P. DeBenedictis, and T. Krishna, "Merge network for a non-Von Neumann accumulate accelerator in a 3D chip," in *2018 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2018, pp. 1–11.

[35] M. Ghadimi, V. Blūms, B. G. Norton, P. M. Fisher, S. C. Connell, J. M. Amini, C. Volin, H. Hayden, C.-S. Pai, D. Kielpinski, M. Lobino, and E. W. Streed, "Scalable ion–photon quantum interface based on integrated diffractive mirrors," *npj Quantum Information*, vol. 3, no. 1, p. 4, Jan 2017.

[36] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[37] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19. New York, NY, USA: ACM, 2019, pp. 987–999.

[38] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernádez, J. Carballo-Franquis, A. Chen, C.-F. Chen, J. M. Chow, A. D. Córcoles-Gonzales, A. J. Cross, A. Cross, J. Cruz-Benito, C. Culver, S. D. L. P. González, E. D. L. Torre, D. Ding, E. Dumitrescu, I. Duran, P. Eendebak, M. Everitt, I. F. Sertage, A. Frisch, A. Fuhrer, J. Gambetta, B. G. Gago, J. Gomez-Mosquera, D. Greenberg, I. Hamamura, V. Havlicek, J. Hellmers, Ł. Herok, H. Horii, S. Hu, T. Imamichi, T. Itoko, A. Javadi-Abhari, N. Kanazawa, A. Karazeev, K. Krsulich, P. Liu, Y. Luh, Y. Maeng, M. Marques, F. J. Martín-Fernández, D. T. McClure, D. McKay, S. Meesala, A. Mezzacapo, N. Moll, D. M. Rodríguez, G. Nannicini, P. Nation, P. Ollitrault, L. J. O'Riordan, H. Paik, J. Pérez, A. Phan, M. Pistoia, V. Prutyanov, M. Reuter, J. Rice, A. R. Davila, R. H. P. Rudy, M. Ryu, N. Sathaye, C. Schnabel, E. Schoute, K. Setia, Y. Shi, A. Silva, Y. Siraichi, S. Sivarajah, J. A. Smolin, M. Soeken, H. Takahashi, I. Tavernelli, C. Taylor, P. Taylour, K. Trabing, M. Treinish, W. Turner, D. Vogt-Lee, C. Vuillot, J. A. Wildstrom, J. Wilson, E. Winston, C. Wood, S. Wood, S. Wörner, I. Y. Akhalwaya, and C. Zoufal, "Qiskit: An open-source framework for quantum computing," 2019.

[39] "ParTI github," https://github.com/hpcgarage/ParTI, 2018.

[40] S. Smith, J. W. Choi, J. Li, R. Vuduc, J. Park, X. Liu, and G. Karypis, "FROSTT: The formidable repository of open sparse tensors and tools," http://frostt.io/, 2017.

[41] P. Lavin, J. Riedy, R. Vuduc, and J. Young, "Spatter: A customizable scatter/gather benchmark," http://spatter.io/, 2019. [Online]. Available: http://spatter.io/

[42] H. C. Edwards, C. R. Trott, and D. Sunderland, "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns," *Journal of Parallel and Distributed Computing*, vol. 74, no. 12,

pp. 3202 – 3216, 2014, domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[43] J. Kepner, P. Aaltonen, D. Bader, A. Buluc, F. Franchetti, J. Gilbert, D. Hutchison, M. Kumar, A. Lumsdaine, H. Meyerhenke, S. McMillan, J. Moreira, J. D. Owens, C. Yang, M. Zalewski, and T. Mattson, "Mathematical foundations of the GraphBLAS," *CoRR*, 2016.

[44] J. Sonnenberg-Klein, R. T. Abler, E. J. Coyle, and H. H. Ai, "Multi-disciplinary vertically integrated teams: Social network analysis of peer evaluations for Vertically Integrated Projects (VIP) program teams," in *2017 ASEE Annual Conference & Exposition*. Columbus, Ohio: ASEE Conferences, June 2017.

[45] N. R. Tallent, K. J. Barker, R. Gioiosa, A. Marquez, G. Kestor, L. Song, A. Tumeo, D. J. Kerbyson, and A. Hoisie, "Assessing advanced technology in CENATE," *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, Aug 2016.