Wrangling Rogues: A Case Study on Managing Experimental Post-Moore Architectures

Will Powell
Jason Riedy
will.powell@cc.gatech.edu
jason.riedy@cc.gatech.edu
School of Computational Science and Engineering
Georgia Institute of Technology
Atlanta, Georgia

Jeffrey S. Young
Thomas M. Conte
jyoung9@gatech.edu
conte@gatech.edu
School of Computer Science
Georgia Institute of Technology
Atlanta, Georgia

Abstract

The Rogues Gallery is a new experimental testbed that is focused on tackling *rogue* architectures for the post-Moore era of computing. While some of these devices have roots in the embedded and high-performance computing spaces, managing current and emerging technologies provides a challenge for system administration that are not always foreseen in traditional data center environments.

We present an overview of the motivations and design of the initial Rogues Gallery testbed and cover some of the unique challenges that we have seen and foresee with upcoming hardware prototypes for future post-Moore research. Specifically, we cover networking, identity management, scheduling of resources, and tools and sensor access aspects of the Rogues Gallery along with techniques we have developed to manage these new platforms. We argue that current tools like the Slurm scheduler can support new rogues without major infrastructure changes.

CCS Concepts

• **Social and professional topics** → **Management of computing and information systems**; • **Hardware** → **Emerging technologies**; • **Computer systems organization** → *Architectures.*

ACM Reference Format:
Will Powell, Jason Riedy, Jeffrey S. Young, and Thomas M. Conte. 2019. Wrangling Rogues: A Case Study on Managing Experimental Post-Moore Architectures . In *Practice and Experience in Advanced Research Computing (PEARC '19), July 28-August 1, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3332186.3332223

1 Challenges for Post-Moore Testbeds

As we look to the end of easy and cost-effective transistor scaling, we enter the *post-Moore* era[33] and reach a turning point in computer system design and usage. Accelerators like GPUs have created a pronounced shift in the high-performance computing and machine learning application spaces, but there is a wide variety of possible architectural choices for the post-Moore era, including memory-centric, neuromorphic, quantum, and reversible computing. These revolutionary research fields combined with alternative materials-based approaches to silicon-based hardware have given us a bewildering array of options and *rogue* devices for the post-Moore era. However, there currently is limited guidance on how to evaluate this novel hardware for tomorrow's application needs.

Creating one-off testbeds for each new post-Moore technology increases the cost of evaluating these new technologies. We present an in-progress case study of a cohesive user environment that enables researchers to perform experiments across different novel architectures that may be in different stages of acceptance by the wider computing community. The ***Rogues Gallery*** is a new experimental testbed focusing on opening access to rogue architectures that may play a larger role in the Post-Moore era of computing. While some of these devices have roots in the embedded and high-performance computing spaces, managing current and emerging technologies provides a challenge for system administration that are not always foreseen in traditional data center environments. Our Rogues Gallery of novel technologies has produced research results and is being used in classes both at Georgia Tech as well as at other institutions.

The key lessons from this testbed (so far) are the following:

- **Invest in rogues, but realize some technology may be short-lived.** We cannot invest too much time in configuring and integrating a novel architecture when tools and software stacks may be in an unstable state of development. Rogues may be short-lived; they may not achieve their goals. Finding their limits is an important aspect of the Rogues Gallery. Early users of these platforms are technically sophisticated and relatively friendly, so not all components of the supporting infrastructure need be user friendly.
- **Physical hardware resources not dedicated to rogues should be kept to a minimum.** As we explain in Section 4, most functionality should be satisfied by VMs and containers rather than dedicating a physical server to manage each rogue piece of hardware.
- **Collaboration and commiseration is key.** Rogues need a community to succeed. If they cannot establish a community (users, vendors interested in providing updates, manageability from an IT perspective), they will disappear. We promote community with our testbed through a documentation wiki, mailing lists, and other communication channels for users as well as close contact with vendors, where appropriate.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PEARC '19, July 28-August 1, 2019, Chicago, IL, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7227-5/19/07…$15.00
https://doi.org/10.1145/3332186.3332223

- **Licensing and appropriate identity management are tough but necessary challenges.** Managing access to prototypes and licensed software are tricky for academic institutions, but we must work with collaborators like start-ups to provide a secure environment with appropriate separation of privileges to protect IP as necessary. Many software-defined network (SDN) environments assume endpoint implementations that may not be possible on first-generation hardware. Documentation restrictions present a particular challenge while trying to build a community.

Here we describe the initial design of the testbed as well as the high-level system management strategy that we use to maintain and tie together novel architectures. We use existing tools as much as possible; we need not re-invent the systems integration wheel to support early novel architectures, which may be few in number and specialized.

2 The Rogues Gallery

The Rogues Gallery was initiated by Georgia Tech's Center for Research into Novel Computing Hierarchies (CRNCH) in late 2017. The Gallery's focus is to acquire new and unique hardware (the rogues) from vendors, research labs, and start-ups and 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.

The primary goal of the Rogues Gallery is to make first-generation start-up or research lab prototypes available to a wide audience of researchers and developers as soon as it is moderately stable for testing, application development, and benchmarking. Examples of such cutting-edge hardware include Emu Technology's Chick, FPGA-based memory-centric platforms, field programmable analog devices (FPAAs), and others. This mirrors how companies like Intel and IBM are investigating novel hardware, Loihi and TrueNorth respectively, but the Rogues Gallery adds student and collaborator access.

The Rogues Gallery provides a coherent, managed environment for exposing new hardware and supporting software and tools to students, researchers, and collaborators. This environment allows users to perform critical architecture, systems, and HPC research, and enables them to train with new technologies so they can be more competitive in today's rapidly changing job market. As part of this goal, CRNCH and Georgia Tech are committed to partnering with vendors to define a flexible access management model that allows for a functional and distinctive research experience for both Georgia Tech and external users, while protecting sensitive intellectual property and technologies.

Not all rogues become long-term products. Some fade away within a few years (or are acquired by companies that fail to productize the technology). The overall infrastructure of a testbed focused on rogues must minimize up-front investment to limit the cost of "just trying out" new technology. As these early-access and prototype platforms change, the infrastructure must also adapt.

And even if the technology is fantastic, rogues that do not develop communities do not last. In our opinion, initial communities grow by easing use and access. Some systems, like the Emu Chick, require substantial thought around restructuring algorithms and data structures. Easing access permits eager users (e.g., students) to play with ideas. Georgia Tech has a history of providing such early access through efforts such as the Sony-Toshiba-IBM Center of Competence for the Cell Broadband Engine Processor[2], the NSF Keeneland GPU system[34], and work with early high core count Intel-based platforms[24].

3 Initial Rogues

Our initial Rogues Gallery shown in Figure 1 consists of various field programmable gate arrays (FPGAs), an Emu Chick, and a field-programmable analog array (FPAA). The gallery also currently includes power monitoring infrastructure for embedded-style systems (NVIDIA Tegra), but we will not go into depth on that more typical infrastructure.

FPGAs for Memory System Exploration Reconfigurable devices like FPGAs are not novel in themselves, but the gallery includes FPGAs for prototyping near-memory and in-network computing. Currently the Rogues Gallery includes

- two Nallatech 385 PCIe cards with mid-grade Intel Arria 10 FPGAs and 8 GiB of memory;
- a Nallatech 520N PCIe card with a Intel Stratix 10, 32 GiB of memory, and multiple 100 GBps network ports; and
- an end-of-lifed Micron AC-510 card that pairs an Xilinx Ultrascale 060 with a 4 GiB Hybrid Memory Cube (HMC).

While now reaching the end of official support, the HMC-based system has many local users and provided results for several recent publications [10, 11] focused on characterization and utilization of the 3D stacked memory component. All platforms are used for exploring new FPGA programming paradigms like OpenCL and domain-specific languages. Related projects like SuperStrider and MetaStrider also are exploring accelerating sparse and graph computations by pushing compute primitives to memory[29].

Emu Chick The Emu architecture (our Emu Chick box is shown in Figure 1) focuses on improved random-access bandwidth scalability by migrating lightweight *Gossamer* threads to data and emphasizing fine-grained memory access. A general Emu system consists of the following processing elements, as illustrated in Figure 3:

- A common *stationary* processor runs the operating system (Linux) and manages storage and network devices.
- *Nodelets* combine narrowly banked memory with several highly multi-threaded, cache-less *Gossamer* cores to provide a memory-centric environment for migrating threads.

These elements are combined into nodes that are connected by a RapidIO fabric. The current generation of Emu systems include one stationary processor for each of the eight nodelets contained within a node. A more detailed description of the Emu architecture is available elsewhere [6, 13]. 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, but that OS cannot be modified (at time of writing). Also, development users need administrative access to reset or reconfigure portions of the Chick. Integrating the Chick into a sufficiently secure infrastructure requires some creativity.

Figure 1: Initial rogues: FPGAs, the Emu Chick, and the field programmable analog array (FPAA)

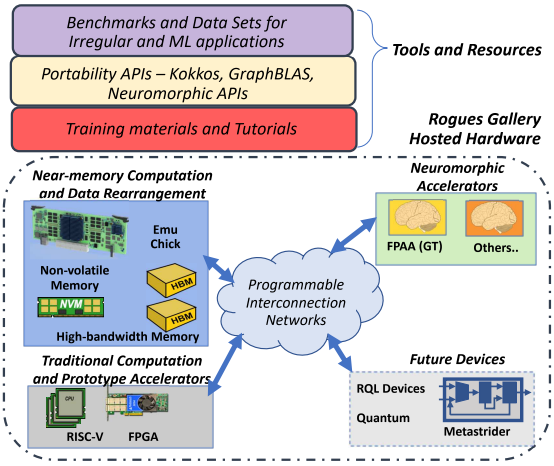

Figure 2: High-level overview of the CRNCH Rogues Gallery

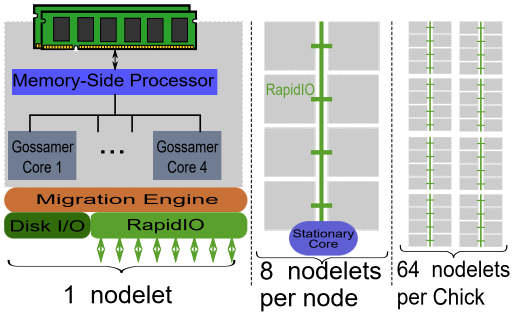

Figure 3: Emu architecture: The system consists of *stationary* processors for running the operating system and up to four *Gossamer* processors per nodelet tightly coupled to memory.

FPAA: Field Programmable Analog Array The Field Programmable Analog Array (FPAA)[9], developed at Georgia Tech by Dr. Jennifer Hasler's group, is a combined analog and digital board that can implement many analog and neuromorphic architectures[12, 27]. The FPAA combines a driving 16-bit MSP430 microprocessor with a 2D array of ultra-low power processors consisting of floating-gate analog plus digital blocks. The exploration and development

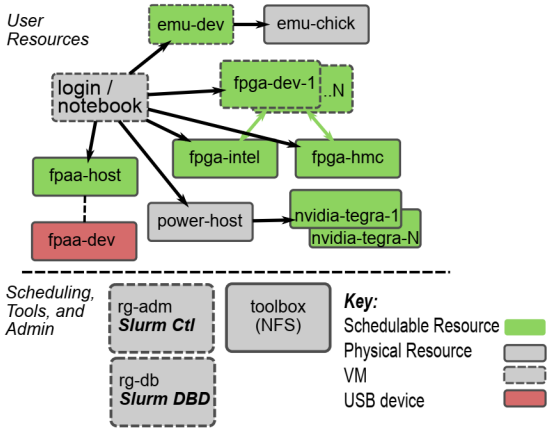

Figure 4: Overview of Rogues Gallery resources and management / network structure

platform is a USB-attached board with multiple analog input ports (Figure 1). All of the development tools for the FPAA and MSP430 are free software based on Scilab and XCos and are distributed in an Ubuntu-based virtual machine image. The FPAA's 2D array combines floating-gate analog units with programmable digital units along with its routing structure. Even manufactured with 350nm CMOS, the FPAA uses 23 μW to recognize the word "dark" in the TIMIT database[12]. 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 justify further research in mixed analog and digital hardware for multiple applications. As the development platform for FPAA is a remote USB device, it provides some challenges in terms of how to monitor, schedule, and maintain it in the Rogues Gallery.

4 Management Issues

Figure 4 shows the management and networking outline of the Rogues gallery. Here we discuss different components of the testbed from the identity management, scheduling, tools support, and networking viewpoints. As demonstrated in the figure, many of the tools and development platforms are hosted on virtual machines while tools are made available via containers where available, as is

the case with the Emu Chick simulator and toolchain as well as the FPAA development VM.

Management of hardware may be complicated because many of the rogues might be temporary additions to the testbed, especially if they fail to build a local userbase or if a company decides to discontinue tool and compiler support. Currently, we plan to reevaluate the hardware composition of the testbed each year and confer with users and industry and national lab colleagues to ensure sysadmin resources are focused on an keeping an up-to-date set of rogues and associated software stacks.

We emphasize using existing tools although sometimes through an extra level of indirection. For example, we cannot fully integrate the Emu Chick into the resource manager, Slurm[1], at the time of writing since each Chick node's OS image is immutable, but we can manage queue-based access to a VM used to access the Chick.

4.1 Software Support

Early hardware often is bound to a specific operating system and release. Tight dependencies make developing and supporting software difficult. For example, we need compatible versions of Slurm for managing resources, but a specific OS version may not have a compatible version available as a vendor package. Additionally, some platforms are best used for running code rather than compiling it, like the Chick's stationary cores. Maintaining many cross-compilation systems for many possible users would defeat our goal of lowering up-front work.

While we may have to support at least two versions of Slurm packages (Redhat and Ubuntu in our case), we find that Singularity, a lightweight user-level container system, provides a convenient way to wrap and deploy compilation environments. For example, we can maintain a single version of tools like the Emu compilers across multiple Ubuntu, Red Hat, *etc.* flavors without overloading the different VMs by sharing minimal development environment containers from a shared tools fileshare across the cluster. The compilation occurs inside the container as if in the target environment. With simple wrappers we also can permit use of the images without distributing them if necessary.

We provide Jupyter notebooks for external demonstrations and tutorials. We are working towards leveraging Kata containers[2] for limited *untrusted* / anonymous demonstration users. Data orchestration into and out of the rogues is our primary barrier.

4.2 Networking Structure

The networking structure is designed to protect sensitive intellectual property and licenses as well as provide general security. Networked devices are attached to firewalling switches that control host and port accesses. Generally inbound access is allowed in the directions of the arrows in Figure 4 and only for ssh. Outbound access generally is not restricted to allow fetching public data sets from online sources. The IT administrative systems (not shown) can access everything directly.

One convenient aspect of completely unusual architectures under heavy development, like the Emu Chick, is that they do not present serious attack targets. Even if someone illicitly accessed the Chick, they would not find many tools to do damage. However, access is managed to prevent leaking system specifics to inappropriate parties.

4.3 Identity Management

The Rogues Gallery has two types of identity management that are tied in with its notion of scheduling unique devices. At a high-level, new users request access to the testbed via a publicly available webform tied to the local administrative mailing list. Once approved, users are set up with a new username and given access to specific subsets of resources. For example, we currently have groups for neuromorphic computing, reconfigurable devices, and the Emu Chick. Each has sub-groups for normal users and admins (with sudo access to related physical devices and VMs). This setup integrates with Georgia Tech's campus authentication system (CAS) as well as our robust in-house role (GRS) and entitlement system (GTED). We leverage Georgia Tech's campus identity system. Currently the benefits of requiring users to have Georgia Tech logins outweighs the effort in setting up those identities, primarily for access control by POSIX groups and file system permissions.

More challenging is identity management for systems that have a limited idea of what a user might be. For example, while the Emu Chick currently has a basic Yocto-generated Linux OS[25] that runs on the controller node and compute nodes, it does not currently support the subset of packages needed to integrate with our LDAP-based authentication system for users. When the Emu Chick initially arrived, it only supported root access. At writing, the Chick still cannot be integrated into an LDAP environment. This is not uncommon for early hardware. Likewise, embedded style devices like standalone FPGA boards (such as the Arria 10 devkit) typically run a micro-kernel and have a limited concept of simultaneous user access or limited access to the FPGA fabric. These devices need to be accessible only through a fully featured front-end that can manage user identity. In the case of different FPGAs in the same system, this needs coupled to OS-level isolation like Linux control groups. For machines like the Chick, network firewall rules permit access only from a front-end VM.

4.4 Scheduling and Management of Resources

Currently, access to several of the Rogues Gallery resources are maintained using soft scheduling via the group's mailing list and other communication channels. Some similar external sites use calendaring systems. Resource slots are represented as conference rooms to be scheduled. These work well for small numbers of users and resources and require essentially no setup. The looser mechanisms do require a small, friendly user community.

As our user base grows, we are bringing the resources under control of the Slurm resource manager. Slurm already supports generalized resources (GRES) to handle licenses and device allocation. We use a single Slurm controller for all the devices and rely on features and Slurm's support for heterogeneous jobs[3]. Currently with a small number of each type of hardware, we have not needed to worry about careful queue controls or complex solutions for co-scheduling heterogeneous resources.

Some systems are relatively easy to bring under Slurm control. The FPGA development and hardware systems are a somewhat typical case of allocating licenses and accelerator nodes. Our only

[1] https://slurm.schedmd.com/
[2] https://katacontainers.io/
[3] https://slurm.schedmd.com/heterogeneous_jobs.html

complication is separating the development machines from the hardware machines; users may need one category or both. Many researchers start up needing only the compilation and development tools. Others have their generated FPGA bitstreams and only need the hardware for experiments. And some will need both for rapid final development turn-around. The resources need to be scheduled both separately and together, again a use already supported by Slurm. The `pam_slurm_adopt` module can clean up cross-connections for mutual allocations. This helps optimize usage for FPGAs and the FPGA development tools.

Other systems like the Emu Chick are more complicated. While Slurm cannot run on the Emu Chick directly, it can manage allocating the Chick's nodes via generic resources (GRES) on the Emu development node. Slurm controls access to the front-end VM. This still requires cooperation between users; we cannot control access to individual nodes without more effort than it is worth. Also, users need root access to the control node to reboot individual nodes. Ideally, Slurm could reconfigure the Chick between the eight single nodes and one eight-node configurations, but that is not stable enough to automate. But we can manage multi-tenant access to the Chick, one user per node, along side single-user access to the entire Chick through different Slurm partitions. This is similar to sharing a multi-core server while still permitting whole-node jobs. These are issues with many first-generation hardware platforms. The risks are worth the benefits in an academic environment.

And then there are USB-connected devices like the FPAAs (Section 3), which pose challenges for sharing and scheduling. Our *plan* is to manage access to the FPAA host (a Raspberry Pi) as with other accelerated systems. The current tools assume a direct USB connection, and we are experimenting with tunneling USB/IP[4] from the Pi to the tools running in a VM on the user's machine. Conveniently, the Pi also supports remote USB power toggling via `uhubctl`[5] for hard FPAA resets and provides analog FPAA inputs via a USB digital to analog converter (DAC) of moderate resolution.

4.5 Tool support

VMs for tools and users are currently provisioned via a RHEV 4.2 cluster of 16 host nodes, with 4 hosts specifically serving research VMs like those used by the Rogues Gallery. The standard image for most of our testbed research is Ubuntu 18.04.5 LTS with one or two CentOS VMs as needed for specific toolsets.

Early hardware may require its own, specific OS and dependency stack. New hardware companies rarely can invest in wide software support. Virtual machines and light-weight containers come to the rescue. We use Singularity[19] to package the Emu toolset and VirtualBox[4] for the FPAA tools.

Singularity wraps the Emu compiler and simulator in a manner convenient for command-line build systems. We can offload compilation and simulation to laptops and other platforms that are less resource-constrained than the VMs. Only users with access to the Emu development node have access to the Singularity image. Because of Emu's rapid development, we do not worry about old versions in the wild after students graduate; they provide no useful information outside of what is publicly available.

The Georgia Tech FPAA tools[3] currently consist of a graphical environment using Scilab and Xcos. These need USB access to the FPAA, so an environment like VirtualBox is more appropriate to encompass the GUI, compiler tools, simulation, and example code. As mentioned in Section 4.4, we are experimenting with methods for remote USB access. Here a major unsolved aspect is providing appropriate physical test inputs to an FPAA device as has been done in previous work focused on acoustic classifiers [27]. While we can replay existing audio samples into the FPAA via a DAC converter, we would like to eventually enable an experimental station where students can work with real-world inputs such as environmental sounds.

We manage FPGA and other licensed tools on a mid-range, dual-socket Intel Xeon system (IBM System x3650 M4) with 192GiB of RAM, a 250GB NVMe drive, and a 500GB ZFS raidz-1 volume. Resources remotely NFS mount the specific tools using autofs. This does require administrative support for deploying new tools, but managing a single shared host is simpler than managing tools on each individual resource. Currently tool versioning is provided by a top-level symbolic link in each resource's top-level directory while keeping older versions accessible when useful (e.g., */usr/local/emu* points to the current Emu compiler toolchain).

4.6 Monitoring the Rogues Gallery

An instance of OpenNMS monitors the Rogues Gallery. Alerts are sent to support personnel regarding any change in the availability of systems in case something unexpected/unscheduled occurs. We also use monitored power distribution units for reports/trends as well as alerts if power usage gets dangerously high (intentionally or not).

However, things are complicated with some of the more novel systems in the Rogues Gallery. For example, with the Emu Chick, we would like to monitor the system management board but not be notified every time a user resets an internal node unless there is an issue. We are investigating how to use tools like ssh-ping to supplement OpenNMS queries for the management node without interfering with other system uses. The FPAA provides a similar challenge in that it is a USB device attached to a related physical host. We would like to check the USB device's online status without interfering with any active activity and also communicate the availability of the resource to our Slurm scheduler. Our current target for performing this type of monitoring is to extend LBNL's Node Health Check [17] script for Slurm to support monitoring (and possibly restarting) specific USB devices.

Monitoring individual FPGA resources without disturbing running experiments similarly is complicated. Currently these devices must be managed as either USB or PCIe devices using node-health scripts. Platforms with an embedded ARM core like Xilinx's Zync board or the Intel Arria10 DevKit provide a basic Linux-enabled core, but it is not clear that these on-board processors can host meaningful monitoring and/or perform communication with a global monitor or scheduler.

5 Reproducibility and Replicability

Many major research venues push for reproducible, or at least replicable, experiments. In general this is a wonderful direction. For early, novel platforms, this may not be easily achievable. Some of our platforms, like the Emu Chick, fix show-stopping bugs with frequent

[4] http://usbip.sourceforge.net/
[5] https://github.com/mvp/uhubctl

software and firmware releases. Reverting to earlier versions to reproduce experiments requires system-wide pain and in many cases is just not feasible.

Other Emu Chick installations have used GT-developed benchmarks and test codes like Emu-STREAM and pointer chasing benchmarks[6], and this has helped Emu Technology identify hardware and software issues. Other test codes that have been developed for the Rogues Gallery are not generally available do to their one-off nature for a specific hardware, firmware, and experimental setup that has changed drastically during the hardware's deployment. With the Emu, the platform's software interface and usage API is changing as well, in part due to research undertaken over the past year using the Rogues Gallery and due to vendor upgrades.

The high-initial-effort "Review of Computational Results Artifacts" option in [15] still is possible and worthwhile. Instilling these ideas in students will pay off over the long run but does require more initial effort than is generally allowed for with academic experimentation and publication cycles. This balancing act for replicating results and meeting deadlines with constantly evolving hardware is an open issue for further research.

6 Education, Outreach, and Collaboration

One benefit to hosting the Rogues Gallery at a university is integrating the Gallery into education. Our initial undergraduate research class, part of the Vertically Integrated Projects program[28], provides novel architecture access for early computing and engineering students[7]. The students are engaged and self-organized into groups focused on the FPAA, the Emu Chick, and integrating quantum simulators like Qiskit[1]. These are students with little initial knowledge of parallel computing, but we hope the experience with novel architectures will prepare them for a wider, post-Moore computing landscape.

As stated previously, no rogue can survive without a community. People need to learn about the platforms and kick the platforms' tires. One mechanism is through organized tutorials. In April 2018 we held a neuromorphic workshop combining invited speakers with hands-on FPAA demonstrations. In April 2019, we presented a tutorial at ASPLOS[22] focused on optimizing for the Emu Chick. And in July 2019, we presented a similar tutorial at PEARC[23]. We also organize sessions at scientific computing conferences that bring together potential users, the Rogues Gallery, and other test beds. Tutorial and presentation materials are made available through our external site https://crnch-rg.gitlab.io and official center site https://crnch.gatech.edu.

Another way to build community is through collaboration with existing high-performance computing and data analysis groups. There are active projects to explore both Kokkos[7] and the GraphBLAS[18] on the Emu Chick. The Emu PGAS model is sufficiently unusual that these are not direct ports but re-working of implementation layers to match the architecture. The FPAA and other neuromorphic-capable platforms present programming challenges that are being addressed by collaborators, such as researchers at UT Knoxville working on the TENNLab framework[20]. We also anticipate a growing collaboration between other Department of Energy (DoE) architecture test beds including CENATE[31] at Pacific Northwest National Lab, ExCL[8] at Oak Ridge National Lab, and Sandia HAAPS[9]. With a limited amount of personnel and funding resources to tackle post-Moore computing hardware and research, we believe that each of these centers can help to fulfill different but overlapping research agendas and can coordinate commonalities within academic, industry, and government userbases.

7 Future Plans

Clear future plans include extending the infrastructure and making new system integration easier. Collaborating with other roguelike centers, including CENATE and ExCL could expose abstractions useful across all such efforts including common scheduling and user management techniques that can also preserve security and data protections. Additionally, solutions to the monitoring issues in Section 4.6 are crucial for enabling research for a primarily remote userbase. Finally, we must collaborate to establish community standards about reproducibility for changing (and possibly disappearing) novel computing platforms.

We are currently looking to adopt new platforms and combine them with upcoming technology like large-scale nonvolatile memory (*e.g.* Intel Octane DC[16]) and programmable networking, as shown in Figure 2. A recent NSF award at Georgia Tech for a general-purpose CPU and GPU cluster with XSEDE[32] integration opens more pathways for exposing the Rogues Gallery infrastructure to a wider community through common user authentication and advertising the Rogues Gallery as a community resources. Integrating the Globus tools with the rogue architectures will be an interesting challenge but will permit larger scaling and data movement to the host VMs and shared storage that support tools and front-end testing.

Handling of Sensitive Data Sets Many interesting applications for these novel platforms involve sensitive data in areas like corporate networks[26] and health care[5] A secure infrastructure to pipe sensitive data through our testbed would benefit both application and hardware / software designers. In this environment, application designers can learn what upcoming hardware could be useful while hardware and software designers would discover which of their assumptions apply to specific, real-world application areas.

Current VLAN, VXLAN, and overlay network methods combined with some assurance for wiping data may suffice for non-regulated industry applications. Per-VM memory and storage encryption combined with the Science DMZ design[8] could assist with data transfer. However, for some levels of certification, particularly in health care, this may not be feasible. Identifying specific roadblocks could help new platform developers engineer around them and possibly provide regulatory agencies guidance otherwise unavailable.

Future Post-Moore Hardware We also have not discussed the deployment of *far-off* devices like those in the quantum computing space in our current testbed. There are upcoming devices in the Noisy Intermediate-Scale Quantum (NISQ) [21] category which we are investigating. Many existing quantum devices require extensive physical hosting requirements that are out of reach for nearly all facilities. However, we envision hosting common sets of quantum tools like those provided by the ProjectQ team [30] and engaging

[6] https://github.com/ehein6/emu-microbench
[7] http://www.vip.gatech.edu/teams/rogues-gallery
[8] https://excl.ornl.gov/
[9] https://www.sandia.gov/asc/computational_systems/HAAPS.html

with smaller quantum start-ups to provide remote access for researchers, perhaps allowing our testbed to become a gateway for this hardware or even a *Rogues Grid*.

8 Summary

The CRNCH Rogues Gallery at Georgia Tech lets researchers experiment with new and novel architectures in a managed but evolving testbed environment. Reducing the intellectual cost of trying new systems enables new and sometimes unexpected applications that can be mapped onto a post-Moore platform. This low barrier to experimentation is supplemented by a growing community that can help with prototyping and porting software and that can help to give vendors feedback on developing new post-Moore hardware.

Although there are multiple vectors for growth and improved infrastructure with the Rogues Gallery, the testbed has already led to some early successes. These positive outcomes include several published academic papers [10, 11, 14, 29, 35], support for PhD thesis research, ongoing collaborations with external academic, industry, and government users, and a job offer from one of the rogue startups for at least one of our PhD students at Georgia Tech. We look forward to new infrastructure developments, student-focused activities like the Rogues Gallery VIP class, and further collaborations with other post-Moore architecture and system evaluation labs to help drive the next phase of the Rogues Gallery's evolution.

Acknowledgments

This work is supported in part by Micron's and Intel's hardware donations, the NSF XScala (ACI-1339745) and SuperSTARLU (OAC-1710371) projects, and IARPA. Sandia National Laboratory provides student support for the Rogues Gallery VIP undergraduate research class. Thanks also to colleagues at GTRI including David Ediger and Jason Poovey and users like Vipin Sachdeva for ideas and assistance with scheduling and FPGA support using Slurm. In addition, thanks to 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. We thank reviewers for their attentive comments.

References

[1] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernádez, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari, Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunho Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreda Rodríguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O'Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyanov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Mingi Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylour, Kenso Trabing, Matthew Treinish, Wes Turner, Desiree Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562110

[2] David A. Bader and Virat Agarwal. 2007. FFTC: Fastest Fourier Transform for the IBM Cell Broadband Engine. *Lecture Notes in Computer Science* (2007), 172–184. https://doi.org/10.1007/978-3-540-77220-0_19

[3] Michelle Collins, Jennifer Hasler, and Suma George. 2016. An Open-Source Tool Set Enabling Analog-Digital-Software Co-Design. *Journal of Low Power Electronics and Applications* 6, 1 (2016). https://doi.org/10.3390/jlpea6010003

[4] Pradyumna Dash. 2013. Getting started with Oracle VM VirtualBox.

[5] Jon Duke. 2018. *Precision Medicine at Georgia Tech: Introduction to the Health Data Analytics Platform.* Center for Heath Analytics and Informatics Seminar Series. Georgia Institute of Technology. http://hdl.handle.net/1853/59350

[6] Timothy Dysart, Peter Kogge, Martin Deneroff, Eric Bovell, Preston Briggs, Jay Brockman, Kenneth Jacobsen, Yujen Juan, Shannon Kuntz, and Richard Lethin. 2016. Highly scalable near memory processing with migrating threads on the Emu system architecture. In *Workshop on Irregular Applications: Architecture and Algorithms (IA3)*. IEEE, 2–9.

[7] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. 2014. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *J. Parallel and Distrib. Comput.* 74, 12 (2014), 3202 – 3216. https://doi.org/10.1016/j.jpdc.2014.07.003 Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[8] Dart Eli, Rotman Lauren, Tierney Brian, Hester Mary, and Zurawski Jason. 2014. The Science DMZ: A network design pattern for data-intensive science. *Scientific Programming* 22, 2 (2014), 173–185. https://doi.org/10.3233/SPR-140382

[9] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R. Wunderlich, S. Nease, and S. Ramakrishnan. 2016. A Programmable and Configurable Mixed-Mode FPAA SoC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24, 6 (June 2016), 2253–2261. https://doi.org/10.1109/TVLSI.2015.2504119

[10] R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, and H. Kim. 2017. Demystifying the characteristics of 3D-stacked memories: A case study for Hybrid Memory Cube. In *2017 IEEE International Symposium on Workload Characterization (IISWC)*. 66–75. https://doi.org/10.1109/IISWC.2017.8167757

[11] R. Hadidi, B. Asgari, J. Young, B. Ahmad Mudassar, K. Garg, T. Krishna, and H. Kim. 2018. 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)*. 99–108. https://doi.org/10.1109/ISPASS.2018.00018

[12] Jennifer Hasler and Harry Marr. 2013. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in Neuroscience* 7 (2013), 118. https://doi.org/10.3389/fnins.2013.00118

[13] Eric Hein, Tom Conte, Jeffrey S. Young, Srinivas Eswar, Jiajia Li, Patrick Lavin, Richard Vuduc, and Jason Riedy. 2018. An Initial Characterization of the Emu Chick. In *The Eighth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES)*.

[14] Eric Hein, Srinivas Eswar, Abdurrahman Yaşar, Jiajia Li, Jeffrey S. Young, Thomas M. Conte, Ümit V. Çatalyürek, Rich Vuduc, Jason Riedy, and Bora Uçar. 2018. Programming Strategies for Irregular Algorithms on the Emu Chick. *CoRR* (2018). arXiv:cs.DC/1901.02775 http://arxiv.org/abs/1901.02775v1

[15] Michael A. Heroux. 2015. Editorial: ACM TOMS Replicated Computational Results Initiative. *ACM Trans. Math. Softw.* 41, 3, Article 13 (June 2015), 5 pages. https://doi.org/10.1145/2743015

[16] Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R. Dulloor, Jishen Zhao, and Steven Swanson. 2019. Basic Performance Measurements of the Intel Optane DC Persistent Memory Module. *CoRR* (2019). arXiv:cs.DC/1903.05714 http://arxiv.org/abs/1903.05714v2

[17] Michael Jennings. 2018. Node Health Check. https://github.com/mej/nhc.

[18] Jeremy Kepner, Peter Aaltonen, David Bader, Aydın Buluc, Franz Franchetti, John Gilbert, Dylan Hutchison, Manoj Kumar, Andrew Lumsdaine, Henning Meyerhenke, Scott McMillan, Jose Moreira, John D. Owens, Carl Yang, Marcin Zalewski, and Timothy Mattson. 2016. Mathematical Foundations of the Graphblas. *CoRR* (2016). arXiv:cs.MS/1606.05790 http://arxiv.org/abs/1606.05790v2

[19] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. 2017. Singularity: Scientific containers for mobility of compute. *PLOS ONE* 12, 5 (May 2017), e0177459. https://doi.org/10.1371/journal.pone.0177459

[20] J. S. Plank, C. D. Schuman, G. Bruer, M. E. Dean, and G. S. Rose. 2018. The TENNLab Exploratory Neuromorphic Computing Framework. *IEEE Letters of the Computer Society* 1, 2 (July-Dec 2018), 17–20. https://doi.org/10.1109/LOCS.2018.2885976

[21] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. https://doi.org/10.22331/q-2018-08-06-79

[22] E. Jason Riedy and Jeffrey S. Young. 2019. Programming Novel Architectures in the Post-Moore Era with The Rogues Gallery. In *24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. Providence, RI.

[23] E. Jason Riedy and Jeffrey S. Young. 2019. Programming Novel Architectures in the Post-Moore Era with The Rogues Gallery. In *Practice and Experience in Advanced Research Computing (PEARC)*. Chicago, IL.

[24] Jason Riedy, Henning Meyerhenke, David A. Bader, David Ediger, and Timothy G. Mattson. 2012. Analysis of Streaming Social Networks and Graphs on Multicore Architectures. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan. https://doi.org/10.1109/ICASSP.2012.6289126

[25] Otavio Salvador and Daiane Angolini. 2014. *Embedded Linux Development with Yocto Project.* Packt Publishing.

[26] Ted E. Senator, Henry G. Goldberg, Alex Memory, William T. Young, Brad Rees, Robert Pierce, Daniel Huang, Matthew Reardon, David A. Bader, Edmond Chow, Irfan Essa, Joshua Jones, Vinay Bettadapura, Duen Horng Chau, Oded Green, Oguz Kaya, Anita Zakrzewska, Erica Briscoe, Rudolph IV L. Mappus, Robert McColl, Lora Weiss, Thomas G. Dietterich, Alan Fern, Weng-Keen Wong, Shubhomoy Das, Andrew Emmott, Jed Irvine, Jay-Yoon Lee, Danai Koutra, Christos Faloutsos, Daniel Corkill, Lisa Friedland, Amanda Gentzel, and David Jensen. 2013. Detecting Insider Threats in a Real Corporate Database of Computer Usage Activity. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. ACM, New York, NY, USA, 1393–1401. https://doi.org/10.1145/2487575.2488213

[27] S. Shah, C. N. Teague, O. T. Inan, and J. Hasler. 2016. A proof-of-concept classifier for acoustic signals from the knee joint on a FPAA. In *2016 IEEE SENSORS*. 1–3. https://doi.org/10.1109/ICSENS.2016.7808748

[28] J. Sonnenberg-Klein, Randal T. Abler, Edward J. Coyle, and Ha Hang Ai. 2017. Multidisciplinary Vertically Integrated Teams: Social Network Analysis of Peer Evaluations for Vertically Integrated Projects (VIP) Program Teams. In *2017 ASEE Annual Conference & Exposition*. ASEE Conferences, Columbus, Ohio, 12. https://peer.asee.org/28697

[29] S. Srikanth, T. M. Conte, E. P. DeBenedictis, and J. Cook. 2017. The Superstrider Architecture: Integrating Logic and Memory Towards Non-Von Neumann Computing. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*. 1–8. https://doi.org/10.1109/ICRC.2017.8123669

[30] Damian S Steiger, Thomas Häner, and Matthias Troyer. 2018. ProjectQ: an open source software framework for quantum computing. *Quantum* 2 (2018), 49.

[31] Nathan R. Tallent, Kevin J. Barker, Roberto Gioiosa, Andres Marquez, Gokcen Kestor, Leon Song, Antonino Tumeo, Darren J. Kerbyson, and Adolfy Hoisie. 2016. Assessing Advanced Technology in CENATE. *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)* (Aug 2016). https://doi.org/10.1109/nas.2016.7549392

[32] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, and et al. 2014. XSEDE: Accelerating Scientific Discovery. *Computing in Science & Engineering* 16, 5 (Sept. 2014), 62–74. https://doi.org/10.1109/mcse.2014.80

[33] Jeffrey S. Vetter, Erik P. DeBenedictis, and Thomas M. Conte. 2017. Architectures for the Post-Moore Era. *IEEE Micro* 37, 4 (2017), 6–8. https://doi.org/10.1109/mm.2017.3211127

[34] Jeffrey S Vetter, Richard Glassbrook, Jack Dongarra, Karsten Schwan, Bruce Loftis, Stephen McNally, Jeremy Meredith, James Rogers, Philip Roth, Kyle Spafford, et al. 2011. Keeneland: Bringing heterogeneous GPU computing to the computational science community. *Computing in Science & Engineering* 13, 5 (2011), 90–95.

[35] Jeffrey Young, Eric Hein, Srinivas Eswar, Patrick Lavin, Jiajia Li, Jason Riedy, Richard Vuduc, and Tom Conte. 2018. *A Microbenchmark Characterization of the Emu Chick.* Technical Report. arXiv:cs.DC/arXiv:1809.07696 https://arxiv.org/abs/1809.07696