Quantum Computing and the Design of the Ultimate Accelerator

Moinuddin Qureshi, *Georgia Institute of Technology, Atlanta, GA, 30332, USA*
Swamit Tannu, *University of Wisconsin—Madison, Madison, WI, 53706, USA*

Quantum computers are domain-specific accelerators that can solve commercially important problems that are beyond the capability of conventional computers. Quantum computing is at an inflection point where small systems with a few dozen qubits have been demonstrated and the number of qubits is expected to increase to several hundred over the coming years. However, the qubit quality is likely to remain quite low making it difficult to execute programs reliably. Furthermore, scaling the designs to a large number of qubits would require specialized hardware that operates at cryogenic conditions. This article describes the compiler and hardware support for enabling reliable and scalable quantum computers.

QUANTUM COMPUTING AND THE ULTIMATE ACCELERATOR

Quantum computing can help solve problems ranging from developing new drugs to discovering novel materials to understanding photosynthesis. These important class of problems are currently intractable on a conventional computer because of the exponentially growing computational complexity. However, a quantum computer, which uses quantum properties itself to store and process data, can simulate molecules and materials in polynomial time. Recent advances in quantum bit (qubit) devices have propelled quantum computing from experimental physics research to an engineering challenge. Currently, several companies have demonstrated quantum computers ranging from 10 qubits to 72 qubits,[1–3] with larger systems (with thousands of qubits) expected within the next few years.[4]

While quantum computers are efficient at solving a class of problems, they are not suitable for solving general-purpose workloads (such as web-browsing, emails, number crunching, and big-data workloads). Therefore, a practical usage model for a quantum computer is to use it as an accelerator for the class of applications for which it is useful. While the concept of accelerators is becoming common in modern systems [such as graphical processing units (GPU) and neural processing units (NPU)], quantum computers are markedly different. For all known accelerators, once the speedup is obtained going to the acceleration paradigm, a linear increase in resources gives atmost a linear increase in speedup (for example, doubling the hardware for GPU/NPU would at most double the speedup). However, for a quantum computer, once we have N qubits, then just adding one more qubit can potentially double the computing power, so going from 50 qubits to 100 qubits could potentially increase the computing power of the quantum system by quadrillion fold. This exponential improvement in the compute capability with a small incremental increase in the hardware is what sets quantum computer apart as an *ultimate accelerator*.

The research in quantum computing ranges from quantum algorithms (that rely on mathematical properties of quantum states) to material scientists investigating devices that exhibit improved quantum behavior (physical properties of quantum states). In the middle part of this stack are computer systems that transform the mathematical properties of algorithms into the state changes in the physical devices. Unfortunately, the systems aspect of quantum computing (architecture and compiler) is less well studied compared to algorithms and devices. As the number of qubits increases, the control circuitry required to provide instructions to the qubits must scale for ensuring both performance and reliability. Similarly,

0272-1732 © 2021 IEEE
Digital Object Identifier 10.1109/MM.2021.3099196
Date of current version 14 September 2021.

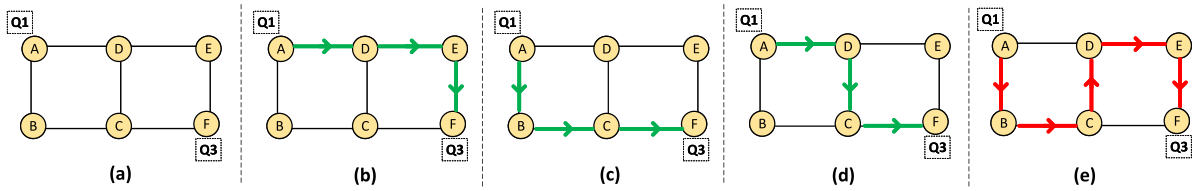

FIGURE 1. (a) Layout of a 6-qubit quantum computer. (b)–(e) Routes from A to F. (b), (c), and (d) have the identical number of swaps and (e) incurs higher swaps. An intelligent policy would choose one from (b), (c), and (d).

the software stack must adapt to new hardware limitations, such as limited connectivity, and imperfect qubit devices.[2] In this article, we describe the hardware and software challenges for enabling scalable quantum computers.

TWO ROADS DIVERGE: NISQ OR FAULT TOLERANCE?

Qubits are fickle as even a small perturbation in the environment can change the state of a qubit. The error rate for a qubit can be defined as a probability of undesired change in the qubit state. Errors in quantum computers can be classified into three broad categories: Coherence errors (state lost within a few tens of microseconds), gate errors (error rate of about 0.1%–1% per operation), and measurement errors (1%–5% per qubit readout). These errors significantly limit the length of computation that can be performed on a quantum computer without an error.

Quantum computers have a unique set of constraints. For example, copying of an unknown state of a qubit is not allowed due to the no-cloning theorem. Therefore, conventional techniques, such as triple modulo redundancy (TMR) and checkpointing are not viable. Quantum computers can be made resilient to errors by using specialized quantum error correction (QEC) codes. Unfortunately, QEC requires a large number of physical qubits (20–50$\times$) to encode one fault-tolerant bit. This 20–500$\times$ overhead for error correction may be acceptable when the quantum machines have thousands of qubits. However, near-term quantum machines will not have enough qubits to implement error correction. The path to fault-tolerant quantum computing (FTQC) is long, and we may not get there for at least a decade.

Figure 2 shows the computing model for the NISQ machines. In this model, the given program is run multiple times, and the output is stored in the log after each trial. As long as the correct results appear with nonnegligible probability, we can infer the correct results by analyzing the statistics of the log. The fidelity of NISQ machines is dictated not only by the quality of the qubit devices (coherence times and operation error rates) but also by the software solutions that orchestrate the computation on the devices. We now capture a few key challenges in the software stack for near-term quantum machines.

CHALLENGE 1: INTELLIGENT COMPILER MAPPINGS FOR RESTRICTED CONNECTIVITY

If a NISQ computer contains N qubits, then ideally, all the qubits will be connected to all other qubits. Such unrestricted connectivity would allow any two arbitrary qubits to get entangled. Unfortunately, such an organization would require approximately (N^2) links, which is impractical even for the 12–72 qubits machines that are available today. The links in a quantum machine are not just wires but resonators that operate at dedicated frequency, and having a large number of such circuits operate reliably on the chip is a difficult task. Therefore, almost all current quantum machines use a Mesh network (or a variant that allows diagonal connections). Such networks restrict that the movement of qubits can occur only between neighboring qubits. For example, for the hypothetical 6-qubit machine shown in Figure 1(a), there is no direct connection between qubits A and F. The communication between these qubits must happen via intermediate qubits. Such restrictions give rise to: 1) qubit-movement policy and 2) qubit-allocation policy.

Qubit-Movement Policy: This policy decides the route that should be used while moving the data from one location on the chip to another. Given that such movement is done using SWAP instructions between neighboring qubits, it is reasonable to select the route that minimizes the number of SWAP instructions. Figure 1(b)–(e) shows the four possible routes from A to F. The first three (b)–(d) require only 3 SWAPs, while (e) requires 4 SWAPs. The SWAP minimizing policy may arbitrarily pick one of the routes from (b)–(d).

Qubit-Allocation Policy: This policy decides the initial mapping of program qubits to the data qubits. For example, it is preferred that qubits that communicate

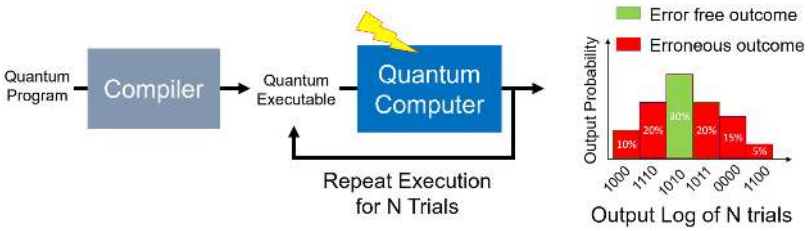

FIGURE 2. NISQ model for quantum computing.

frequently be placed near each other. For example, if we wanted to place four qubits on the machine shown in Figure 1(a), we would not keep these qubits on the four corners, and instead, we will try to use the middle two qubits (D and E), as doing so would minimize the SWAP instructions, required for communication. In fact, recent studies[5] have proposed such allocation policies based on minimizing the number of SWAP instructions.

However, even for a small machine with less than 50 qubits, these algorithms require significant compilation time (in the range of 30 min to several hours). As machines scale to hundreds of qubits, existing solutions for doing compiler mappings would incur impractically large compilation time. Therefore, an open challenge in this domain is to develop intelligent mapping algorithms that are effective at reducing SWAP operations for machines with hundreds (or thousands) of qubits while having low compilation time.

CHALLENGE 2: SOFTWARE THAT ADAPTS WITH HARDWARE QUALITY

The runtime of the quantum machine is responsible for qubit allocation (mapping the logical qubit to physical qubit), routing (inserting extra swaps to move the source qubit to the destination qubit for performing two-qubit operations), and qubit measurement. From the characterization reports provided by the cloud vendor, we observe a significant variation in error rates across qubits and links, with some qubits and links experiencing as much as 5× the error rates as the average qubits and links. Suppose the runtime is aware of the variation in error rates. In that case, it can perform the allocation, routing, and measurement in a manner such that most of the computation happens on the high fidelity qubits and links (the one with lower error rates), and we avoid the unreliable qubits and links (the ones with higher error rates). Our prior work[6] shows that such hardware error-rate-aware mapping and routing can significantly improve the fidelity of the quantum program compared to the state-of-the-art SWAP minimizing mapping.

We explain variation-aware mapping with an example. In Figure 3(a), if we want to map three program qubits to five physical qubits, we can pick any of three connected qubits (for example, Q_1 maps to A, Q_2 maps to B, and Q_3 maps to C), as placing qubits nearby results in SWAP minimization. Doing so assumes uniformity in the cost of performing SWAP operations. However, we can make the allocation policy aware of the variation in error rates. The variation-aware allocation policy will choose D, E, and A.

While we focus on exploiting the variation of only two-qubit operations, the idea can be generalized to account for variability in all gate operations, coherence errors, and measurement errors. Furthermore, there are other sources of errors, such as correlated errors[7] and measurement errors,[8] which can significantly vary across different devices on the machine. A key challenge for qubit mapping algorithms is to exploit the variation in errors from different modalities and perform mapping such that the overall fidelity of the quantum program is improved, all while doing it with manageable compilation time for a machine with hundreds of qubits. This is a difficult problem and is an area of active research in the architecture and compiler community.

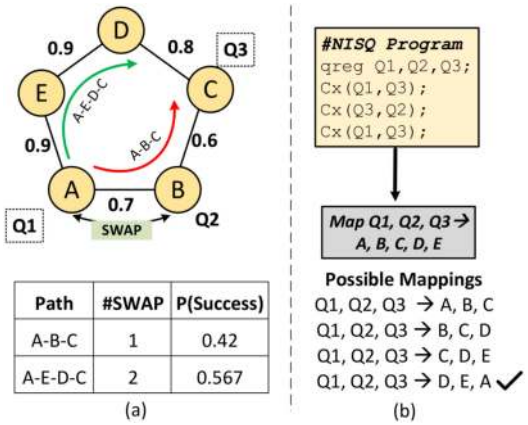

FIGURE 3. (a) A hypothetical quantum computer with five-qubits—the number on the edge denotes the success probability when that edge is used. Variation-aware qubit mapping (VQM) provides higher probability of success. (b) Variation-aware qubit allocation (VQA) helps with selecting the strongest links.

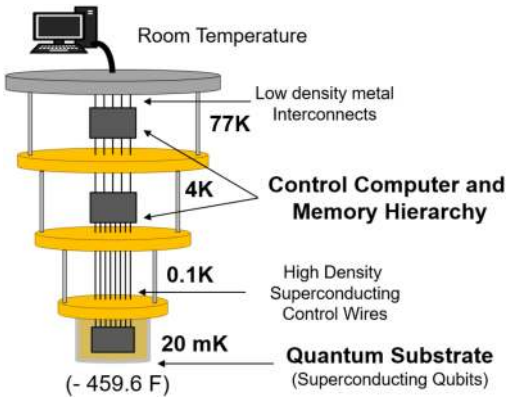

FIGURE 4. Thermal profile of scalable quantum computer.

CHALLENGE 3: SPECIALIZED CRYOGENIC HARDWARE FOR SCALABILITY

Scaling a quantum computer to a large number of qubits requires improvement in not only the qubit device technology but also the hardware architecture that interfaces with the qubit devices. Qubits are extremely fragile and can quickly decohere (lose their state) due to thermal noise. To protect the fickle quantum states, qubits are operated at the temperature of a few milli-Kelvins as shown in Figure 4. Unfortunately, the power budget at such a low temperature is quite small (10 μW).

To build a scalable quantum computer, a control processor needs to be placed close to the qubits at cryogenic temperatures because the control processor operated at room temperature cannot scale beyond a few hundred qubits due to a limited number of wires connecting the room temperature and the quantum substrate. The metal interconnects connecting room-temperature memory, and the cryogenic control processor would have thermal leakage due to the large temperature gradient between room temperature and control processor (about 300 K) that can destabilize the noise-sensitive qubits. The memory system required for the control processor would need significant capacity, which is impractical to provide at 4 K temperature, therefore the memory can be kept at cryogenic temperature (70–90 K). The host system is a regular processor operated at room temperature (300 K) that provides the executables to the control processor and the cryogenic memory system. Thus, a careful orchestration of work in a quantum computer between different thermal domains is critical to avoid thermal leakage and maintain the ultralow temperatures.

One of the challenges in enabling scalable quantum computers is the design of the control processor, which is responsible for providing the instructions to the qubits and for doing qubit measurements. An example of a cryogenic control processor is the recent *Horse Ridge* chip from Intel. As qubits require deterministic instruction supply, the instruction bandwidth must scale linearly with the number of qubits, and this becomes a significant bottleneck at low power budgets. We observe that the majority (99.99%) of the instructions in a quantum instruction stream corresponds to QEC. Our recent work proposes QEC substrate architecture[9] that uses a microcode based design to delegate the error correction responsibility to the hardware and reduces the instruction bandwidth requirement by several orders of magnitude. We investigate the organization of microcode structures for low area budget (few kilobytes) offered in technology that works at 4 K (Josephson Junction or JJ). While such architectures can provide high instruction bandwidth to large-scale quantum computers, they manage each qubit independently and do not provide synchronization in the state of different qubits. To ensure correctness, it important that multiple qubits that perform the same multiqubit operation must be synchronized to get correct results. Therefore, future architecture must provide microarchitecture designs that can efficiently perform synchronization.

The control processor will require substantial memory capacity to store quantum programs (which can have a footprint of several tens of gigabytes) and to store the results of syndrome measurements for error decoding and qubit tuning. Unfortunately, the memory technology that works at 4 K offers a capacity of only a few kilobytes. It is important to explore the feasibility of commodity DRAM at cryogenic temperatures (Cryo-DRAM).[10]

CHALLENGE 4: FAST ERROR DECODING FOR ENABLING FAULT TOLERANCE

Even with the architecture and memory to support the control and operation of a large number of qubits, enabling fault-tolerant quantum computers faces a critical obstacle: the development of a low-latency and accurate error-decoder. A fault-tolerant quantum computer (FTQC) needs to regularly refresh qubit by QEC within a short period of time (typically less than 1% of the coherence time period, so the latency is constrained to less than 1 μs).

QEC is way more challenging than classical error correction because the error rates of qubits are

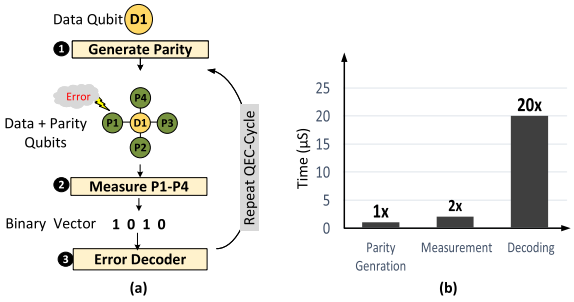

FIGURE 5. The framework for quantum error correction and the process of error decoding.

orders of magnitude higher, and their inherent properties impose many fundamental restrictions. For example, qubits cannot be copied and lose their quantum state when measured. To tackle these challenges, QEC codes encode *logical* qubits by using a set of *data* and *ancilla* qubits. The data qubits represent the quantum information, whereas the ancilla qubits periodically interact or entangle with the data qubits as per the QEC protocol and reveal information about errors on the data qubits upon measurement. This process is called syndrome measurement or extraction, and the output of ancilla measurements is called a *syndrome*. An *error decoder* processes the syndrome to locate and identify the type of errors on the data qubits. The process of error decoding and the relative latency are shown in Figure 5.

To be practical for implementation in an FTQC, a decoder[11] must satisfy three constraints: accuracy, latency, and scalability. The *accuracy constraint* suggests that the decoder must correctly identify all the errors (both in the qubit as well as measurement operations in the QEC cycles) with very high probability. The *latency constraint* requires decoders to correct errors within an error correction cycle to prevent any backlog or accumulation of errors. The *scalability constraint* indicates that it must be feasible to implement the decoder in FTQCs with thousands of logical qubits. To scale to such large FTQCs and simultaneously facilitate operation inside a constrained cryogenic environment, decoders must be hardware efficient. The proximity to the physical qubits necessitates the use of minimal hardware to ensure that the thermal heat generated from the decoding circuits does not introduce additional noise channels for the qubits. Therefore, hardware research needs to focus on the design of low-latency, accurate, and scalable decoders that can operate within the constraints of a cryogenic environment.

CHALLENGE 5: NOVEL HYBRID PARADIGMS COMBINING QUANTUM AND CLASSICAL

When a program is executed on a computer, we expect correct results. With FTQC, the quantum hardware is likely to produce the right results based on the strength of the error correction code. However, in the foreseeable future, when we are unable to implement FTQC, imperfect devices can cause erroneous results. This severely limits the length of the program that can be executed reliably on a near-term quantum computer, and for long programs, the likelihood of getting the correct results becomes vanishingly small.

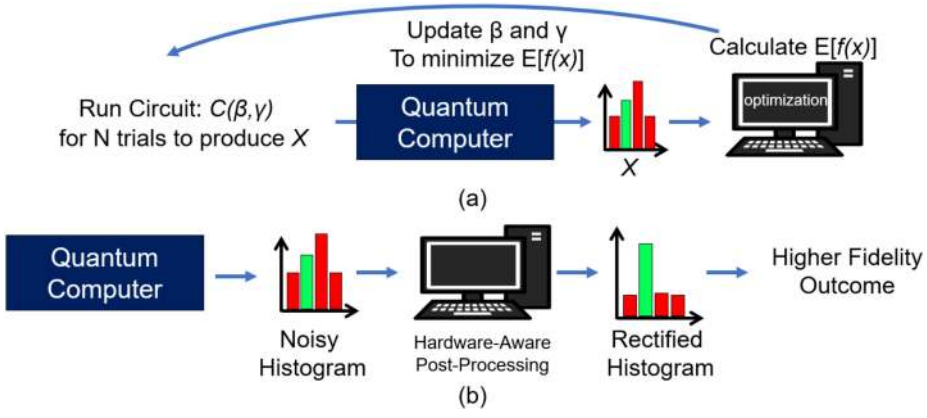

FIGURE 6. (a) Quantum-classical hybrid paradigm used for the QAOA problem. (b) Quantum-classical synergy framework via hardware-adaptive post-processing.

To extend the usefulness of near-term quantum machines, a new paradigm of hybrid quantum-classical model is emerging that combines both quantum machine and classical machine. Some applications, such as quantum approximate optimization algorithm (QAOA) and variational quantum algorithm (VQA) are amenable to such model of execution. Figure 6(a) shows the hybrid quantum-classical model of execution for QAOA, where the parameters of the algorithm are obtained from the quantum machine, then they are used to perform an optimization routine on the classical computer, and a new set of parameters are provided to the next iteration of the quantum machine. One of the key requirements for QAOA is to have a low latency of iteration, therefore, it is important to have low latency for both the classical optimizer and for the compilation time for the revised circuit for the next iteration. Unfortunately, currently only a restricted subset of applications are amenable to the hybrid quantum-classical paradigm and making this model applicable to a wider variety of applications will greatly improve the usefulness of near-term quantum computers.

WE ARE IN AN EXCITING TIME IN COMPUTER SYSTEMS, WHERE NEW PARADIGM OF PROCESSING UNITS SPECIALIZED FOR A SPECIFIC DOMAIN ARE BECOMING POPULAR. QUANTUM COMPUTERS ARE AN ACCELERATOR THAT CAN SOLVE HARD PROBLEMS THAT ARE BEYOND THE CAPABILITY OF CONVENTIONAL MACHINES.

Another approach to synergistically combine the capability of quantum computers and classical computer is to develop intelligent postprocessing algorithms. Figure 6(b) captures the flow of such an execution model. Currently, for NISQ machines, we run the program for a large number of trials, and we obtain a histogram that contains both correct answers and incorrect answers. The conventional wisdom is that the wrong answers produced by a quantum machine are arbitrary in nature—for example, occurring uniformly at random. However, in reality, we observe that there is a significant amount of structure in the wrong answers—for example, if the computation is only affected by a single measurement error, then the resulting incorrect answer will be one hamming distance away from the correct answer. We could develop intelligent postprocessing algorithms that can take hardware error behavior into account and rectify the histogram to increase the probability of the correct answer.

CONCLUSION

We are in an exciting time in computer systems, where new paradigm of processing units specialized for a specific domain are becoming popular. Quantum computers are an accelerator that can solve hard problems that are beyond the capability of conventional machines. However, to realize such machines, significant progress needs to be made not just by the quantum devices community and the quantum algorithms community but also by the computer systems community. We will need to develop intelligent software stack, the right hardware support, and new execution paradigms to fully harness the power of quantum computers. In some sense, the field is just getting started, and there is significant potential to make fundamental contributions. We cannot wait to see the progress that the quantum research community will make over the next decade.

REFERENCES

1. B. Villalonga *et al.*, "Establishing the quantum supremacy frontier with a 281 pflop/s simulation," 2019. [Online]. Available: https://arxiv.org/abs/1905.00444
2. I. B. M. Corporation, "Universal quantum computer development at IBM," 2017. Accessed: Apr. 2017. [Online]. Available: http://research.ibm.com/ibm-q/research/
3. Honeywell, "Honeywell quantum solutions," 2021. Accessed: Apr. 11, 2021. [Online]. Available: https://www.honeywell.com/us/en/company/quantum
4. A. Cho, "IBM promises 1000-Qubit quantum computer-A milestone-By 2023," 2021. Accessed: Apr. 11, 2021. [Online]. Available: https://www.sciencemag.org/news/2020/09/ibm-promises-1000-qubit-quantum-computer-milestone-2023
5. A. Zulehner, A. Paler, and R. Wille, "Efficient mapping of quantum circuits to the IBM QX architectures," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2018, pp. 1135–1138.
6. 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 *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2019, pp. 987–999.

7. S. S. Tannu and M. Qureshi, "Ensemble of diverse mappings: improving reliability of quantum computers by orchestrating dissimilar mistakes," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2019, pp. 253–265.
8. S. S. Tannu and M. K. Qureshi, "Mitigating measurement errors in quantum computers by exploiting state-dependent bias," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit.*, Oct. 2019, pp. 279–290.
9. S. S. Tannu, Z. A. Myers, P. J. Nair, D. M. Carmean, and M. K. Qureshi, "Taming the instruction bandwidth of quantum computers via hardware-managed error correction," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2017, pp. 679–691.
10. S. S. Tannu, D. M. Carmean, and M. K. Qureshi, "Cryogenic-dram based memory system for scalable quantum computers: A feasibility study," in *Proc. Int. Symp. Memory Syst.*, 2017, pp. 189–195.
11. P. Das *et al.*, "A scalable decoder micro-architecture for fault-tolerant quantum computing," 2020. [Online]. Available: https://arxiv.org/abs/2001.06598

MOINUDDIN QURESHI is a Professor of Computer Science with Georgia Institute of Technology, Atlanta, GA, USA. His research interests include computer architecture, hardware security, and quantum computing. He was a Research Scientist with IBM Yorktown (2007–2011), where he developed the caching algorithms for Power 7 Systems. Qureshi received a Ph.D. degree from the University of Texas at Austin in 2007. His research has been recognized with multiple best-paper awards, IEEE Top-Picks, and "Persistent Impact Prize" awards. Contact him at moin@gatech.edu.

SWAMIT TANNU is an Assistant Professor with the Computer Science Department, University of Wisconsin—Madison, Madison, WI, USA. His research interests include computer architecture, quantum computing, and emerging technologies. His research focuses on developing compiler and architectural abstractions for quantum computers and cryogenic accelerators. Tannu received a Ph.D. degree in electrical and computer engineering from Georgia Tech, Atlanta, GA, USA. Contact him at stannu@wisc.edu.