
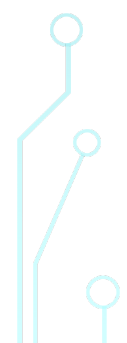






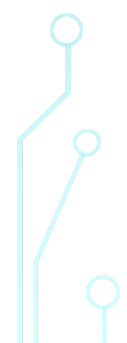
SEPTEMBER 2017



- Agenda
 - Introductions and thanks to Microsoft
 - Food/Pizza
 - Bell / GHZ / CHSH
 - Discussion
 - Presentations can be found at github.com/NYCQuantumComputing
 - Twitter [@NYCQuantum](https://twitter.com/NYCQuantum)
 - Looking for hosts, presenters, topics, suggestions
 - Bell's Theorem, Computational Complexity, Shor, Simon, Deutsch, Physics, modeling algorithms
- 
- 



RECAP 2017

- Github pointer to papers slides github.com/NYCQuantumComputing
 - March 2017 – Kickoff
 - April 2017 – Grover search, IBM's Quantum Experience, math behind Grover
 - UMD – Implementation of three bit search
 - Different oracles, two value search
 - May 2017 – Thanks to DWAVE for their technical presentation
 - June 2017 – Thanks to Chris Monroe from IONQJ
 - July 2017 – Quantum Entanglement
- 
- 
- 


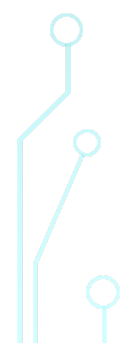
“FIRST QUANTUM COMPUTERS NEED SMART SOFTWARE”

- *Nature*, September 2017, Will Zeng, et.al. (Righetti)
- The article’s major points
 - Quantum computers will be implemented as hybrid system
 - QC needs open software
 - QC needs a community
- My add’l point – educate programmers





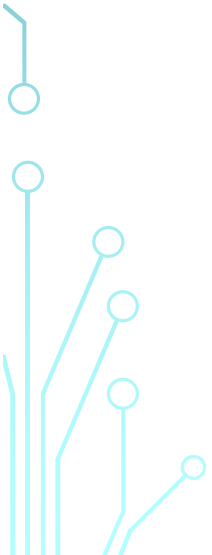
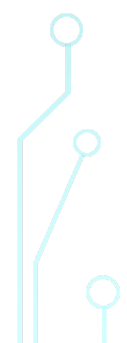
Quantum computers will be implemented as hybrid system

- “**These algorithms are generally designed for big, noiseless quantum computers, which are unlike the devices that will be available within the next five years.** These will have tens to thousands, not millions, of qubits, with little redundancy to correct for internal errors. They will calculate a limited range of things in a noisy way. For example, they will not be able to use Shor’s algorithm to find the prime factors of large numbers. So their use must be targeted: they will not always beat conventional computers. “
 - “These limitations can be overcome by building **quantum processors as ‘accelerators’** to boost the performance of conventional computers. A classical computer might, for example, optimize operations to compensate for noise in the quantum processor, or aggregate answers from sequences of short quantum programs. Such hybrid programming has been demonstrated in quantum chemistry³ and in optimization⁴”
- 
- 



QC needs open software

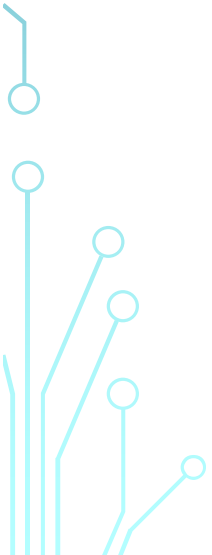
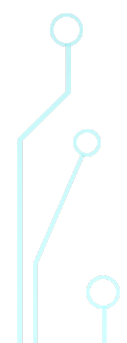


- “Different classical computers behave similarly enough to enable software written for one to run on others. **Early quantum computers will have their own nuances, and software for them will need to be bespoke.** When each operation and instruction matters, generalized solutions need to be optimized, and software and hardware designed concurrently. Algorithms must be discovered numerically rather than algebraically, and developed using simulators and software rather than pens and paper.
 - Easy programming interfaces are crucial to making quantum computers widely usable; examples include Quil and OpenQASM8 from IBM. More sophisticated options still need to be added, such as optimizations for specific types of processors. **Higher-level languages for writing and compiling quantum programs also need to be developed.**
- 
- 



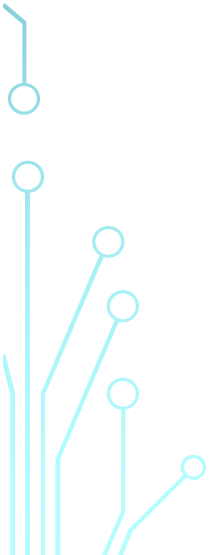
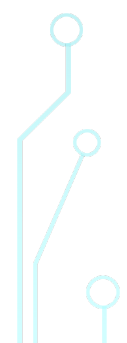
QC needs open software



- **“It is important that all these tools are open source.** Such a model was not available at the dawn of digital computing, but its power to speed innovation, as with Linux in the early days of the web, is essential for the quantum-programming community to grow quickly. We have made a start with our quantum-programming toolkit, Forest, which is written in Python, open source and accessible to anyone. It joins an exciting early ecosystem — much of it open source — developed by different academic and industrial research groups. Other examples are LIQ*U*i|> (embedded in F#), Scaffold (C++), Quipper (Haskell), QGL (Python), ProjectQ (Python), QCL, QuIDDPro and Chisel-Q (Scala). **Researchers must resist pressure to standardize tools prematurely or keep the high-level, exploratory parts of the programming stack proprietary.** “
- 
- 

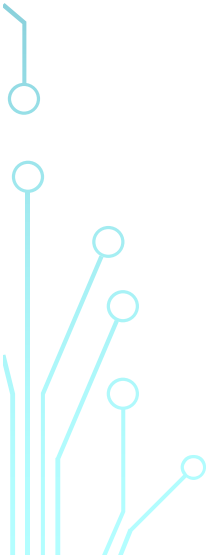
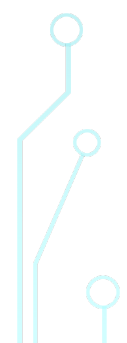


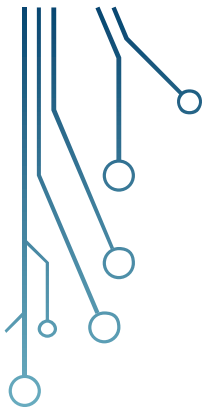
QC needs a community

- **“A new breed of quantum programmer is needed to study and implement quantum software — with a skillset between that of a quantum information theorist and a software engineer.** Such programmers will understand how quantum devices operate well enough to instruct them and minimize problems. They will be able to build usable software and will have a deep knowledge of the mathematics of quantum algorithms and computation.
- 
- 




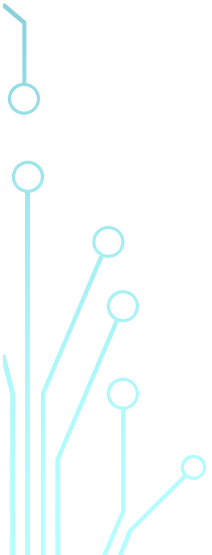
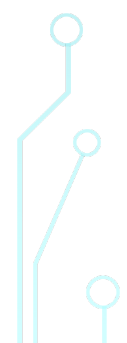
QC needs a community

- **Advanced kinds of education are needed to train this new breed. Several centres are well positioned to draw together the inter- disciplinary skills and tools needed to offer degrees in quantum-computer engineer- ing:** the Institute for Quantum Computing at the University of Waterloo in Canada, the Institute for Quantum Information and Matter at the California Institute of Technology in Pasadena, the quantum-engineering doctoral training centres in the United Kingdom, and QuSoft, the Dutch research centre for quantum software in Amsterdam. At Rigetti we have started a Junior Quantum Engineer programme for bachelor's degree students, which includes training in quantum programming. We have partnered with the Quantum Machine Learning accelerator at the Creative Destruction Lab (a technol- ogy-transfer centre that fosters start-ups) at the University of Toronto, Canada, to pro- vide access to and support for Forest and other programming tools.
- 
- 




Next Steps (from article)

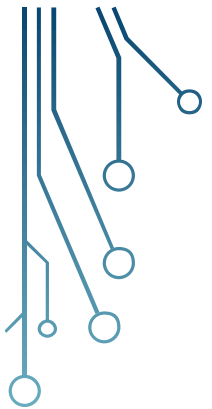


- It is crucial that research on quantum- computing algorithms is tied more closely to research on the software that's used to implement them.
 - First, funders should insist that **theoretical work is implemented in software** and, as much as possible, tested on hardware.
 - Second, **algorithm researchers must be explicit about the architecture they are targeting**. They must show evidence of how algorithms will be practically implemented on different noisy systems.
 - Third, funders and journal editors must establish **standard ways to assess algorithm performance and resource requirements**. This will enable hardware and software to improve together, and will sift out the most viable algorithms more quickly. Open-source tools should be used wherever possible, and publications should encourage the publication of code alongside results.
 - **Finally, the quantum-computing community must prioritize engagement with experts in areas such as simulation and machine learning. Quantum and classical programmers must collaborate more.**
- 
- 

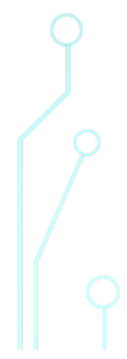


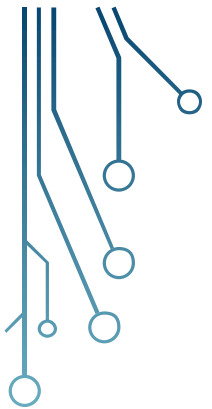
Educate the programmers

- How do existing, complex algorithms map into QC primitives?
 - Financial, chemistry, “large models”
 - What new algorithms can be created using QC?
 - Are there any other “tricks” in QC?
 - Grover search, Shor, entanglement
 - ***Nature, September 2017, Will Zeng, et.al. (Righetti)***
- 

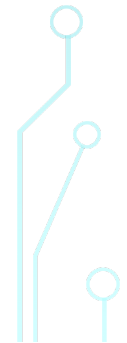
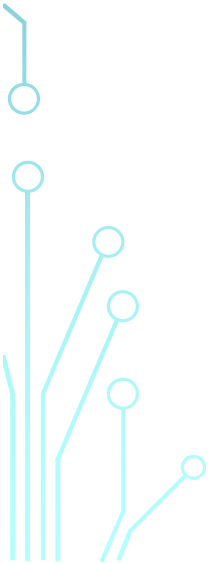


GHZ / CHSH / BELL





DISCUSSION





OCTOBER 2017 MEETING

- October 11, 2017 @ IBM NYC
 - Topics? Speakers?
- 
- 
- 