Final Report: BIRS Workshop 16w5029
Quantum Computer Science
April 17–22, 2016

Co-organizers: Michele Mosca, Martin Roetteler, and Peter Selinger

# 1 Objectives of the workshop

Shor's famous quantum algorithms for integer factoring and to compute discrete logarithms helped to kick-start the field of quantum computing. A topic that has received significantly less attention is that of actually translating quantum algorithms into elementary instructions that can then be carried out on a future large-scale quantum computer. This leads to questions about automation of the translation process, a question that gives rise to several challenges that arise naturally from there, for instance: how to program a quantum computer? How to model the underlying instructions? How to compile and optimize code that has been written in a high level of abstraction?

This interdisciplinary workshop aimed at making progress on these questions and brought together over 40 researchers from a diverse set of research communities: those working in quantum computing, and those working on the mathematical foundations of programming languages and their implementations. We anticipate a vibrant exchange of novel ideas, to tackle important problems such as the optimization of large-scale quantum algorithms, and their synthesis into the instruction set of a fault-tolerant quantum computer.

On the first day there was a very lively open problems debate, a summary of which is provided in this report. The problems ranged from big challenges that likely will remain open for some time, to concrete problems, some of which were tackled during the workshop. Besides the open problem discussion and a plethora of talks on recent topics of interests, salient features of the workshop were various software demo sessions in which some of the participants performed live demos of their software programs, and two tutorials about quantum programming languages.

We expect that the workshop will help to further establish quantum programming languages and circuit synthesis and optimization as a rapidly evolving research area in the intersection of quantum computing and programming languages.

# 2 Open problem discussion

Before the workshop we asked the participants to think of 1-2 open problems, seeking inspiration from Charles Kettering, the famous inventor (of such feats as the all-electric car ignition but also such doozies as leaded gasoline) who said that "a problem well stated is a problem half solved." We were very happy to see that the discussion spanned a wide range of open problems in quantum circuit synthesis, optimization, fault-tolerance, and programming and that the participants indeed brought to the table a mix of their favorite grand challenges, pet peeves, brain teasers, and conjectures. The resulting list is the result of the discussion and is loosely divided into the areas that were the main topics of discussion.

**Non-deterministic/probabilistic protocols** This includes questions about state distillation and circuits with fallback. Basic open questions included:

- Are non-deterministic protocols provably better than deterministic ones?
- How to parallelize non-deterministic protocols? How to deal with synchronization issues?
- Generally, how to cost the protocols? What are the relevant metrics besides circuit size, depth, and width? Can for instance probability of success be traded against some of the circuit-centric metrics in a meaningful way?
- How to characterize online vs. offline cost in a good way?
- How to capture the relative cost of gates? For instance when considering distillation which might give rise to more powerful gate sets which however might increase the offline computation part.
- What are the cost related to dimensionality / layout of the qubits, e.g., in 2D or 3D?

**Better representation of arithmetic operations** Under this umbrella there are mainly questions about how to implement various arithmetic functions under the constraints that quantum computing imposes. Basic open questions included:

- How to trade various metrics such as time and space in a good way when it comes to generating reversible circuits for arithmetic?
- How to implement functions that will be needed for many complex quantum algorithms such as trigonometric functions, inverse trigonometric functions, $1/x$, $1/\sqrt{x}$, etc.

**Lower bounds on circuit size**

- Can lower bounds on circuit size be found, e.g., in topological quantum computation?
- Lower bounds on $T$-depth, for example: can $THTHT$ be written in $T$-depth 2 with ancillas?
- What pure math do we need to answer such questions?
- Is there a provable link between $T$-depth and measurement depth?

**Lower bounds for permutation networks**

- It is known, for example, that a SWAP gate requires 3 CNOT gates, and a SWAP on 3 bits requires 6 CNOT gates. Can we show other lower bounds?
- Show lower bounds also for more general gate sets, say with Toffoli gates.

**Generators and relations**

- Can a presentation in terms of generators and relations be found for special gate sets, e.g. for Clifford+$T$?
- How about for classical reversible circuits?

**Size of quantum implementation of classical functions**

- For example, can we find a classical reversible function whose smallest classical circuit is exponentially larger than an equivalent Clifford+$T$ circuit?

**Compiling into topological models**

- How can one achieve this e.g. for the anyon model, where spaces are not tensor products?
- How can one combine anyons in a meaningful way?

**Prove the number-theoretic hypotheses required for approximate synthesis**

- Termination seems to rely on strong assumptions about the distribution of primes. Can those be proven?

**Can we prove hardness of certain synthesis problems?** This question applies, for example, to the case of optimal synthesis of non-diagonal operators in Clifford+$T$.

- Can this be shown to be NP-complete?
- Synthesis of $V_x + V_z$, i.e., dropping the generator $V_y$ from the $V$ basis.

**Programming quantum computers**

- What is a good a common platform for implementation (like LLVM)?
- Can we implement ancilla management in a meaningful way (verification or "correct by construction" guarantees)?

**Recursive quantum programming**

- Is the principle of deferred measurement compatible with recursive quantum programming?
- How about a quantum stack? Would this make sense?

# 3 Presentation highlights

**Matt Amy: $T$-count optimization and Reed-Muller codes** Matt showed that minimizing $T$-count in $n$-qubit CNOT and $T$ quantum circuits reduces to minimum distance decoding of the length $2^n - 1$ punctured Reed-Muller code of order $n - 4$. A converse reduction also exists, providing strong evidence that no polynomial-time solution is possible. As a consequence, he derived an algorithm for the optimization of T-count in Clifford+T circuits which can utilize the wide variety of Reed-Muller decoders developed over the years, along with a new upper bound on the number of T gates required to implement a unitary over CNOT and $T$ gates. He further generalized this result to show that minimizing finer angle Z-rotations corresponds to decoding lower order binary Reed-Muller codes. This talk was based on joint work with Michele Mosca which is available at http://arxiv.org/abs/1601.07363.

**Holger Bock Axelsen: Reversible networks for sorting** Arguably, sorting is one of the most fundamental and important tasks in classical computing. Holger considered a model for reversible computing that is based on the programming language Janus and extensions thereof. This model is very natural for enforcing the reversibility constraints that have to be imposed on any program that is to be executed on a quantum computer. Sorting imposes some particular constraints when trying to implement it on a reversible computer, a very basic problem being that the $O(n \log n)$ cost that is classically obtainable from various recursive methods requires quite a bit of work in the reversible case as straightforward solutions either lead to large space overhead or lose the time advantage of the algorithm outright. Holder presented techniques that can be used to control both, the time and the space overhead of the reversible implementation in Janus, so that $O(n^2)$ can be achieved for an insertion sort based algorithm, and $O(n \log n)$ for a quicksort based algorithm.

**Anne Broadbent: How to verify a quantum computation** Anne's talk was based on her recent paper http://arxiv.org/abs/1509.09180. She gave a new interactive protocol for the verification of quantum computations in the regime of high computational complexity. Her results were given in the language of quantum interactive proof systems. Specifically, she showed that any language in BQP has a quantum interactive proof system with a polynomial-time classical verifier (who can also prepare random single-qubit pure states), and a quantum polynomial-time prover. Here, soundness is unconditional–i.e. it holds even for computationally unbounded provers. Compared to prior work, our technique does not require the encoding of the input or of the computation; instead, we rely on encryption of the input (together with a method to perform computations on encrypted inputs), and show that the random choice between three types of input (defining a computational run, versus two types of test runs) suffice. Anne also presented a new soundness analysis, based on a reduction to an entanglement-based protocol.

**Hillary Dawkins: Small codes for magic state distillation** Magic state distillation is a critical component in leading proposals for fault-tolerant quantum computation. Relatively little is known, however, about how to construct a magic state distillation routine or, more specifically, which stabilizer codes are suitable for the task. While transversality of a non-Clifford gate within a code

often leads to efficient distillation routines, it appears to not be a necessary condition. In her talk, Hillary examined a number of small stabilizer codes and highlight a handful of which displaying interesting, albeit inefficient, distillation behaviour. Many of these distill noisy states right up to the boundary of the known undistillable region, while some distill toward non-stabilizer states that have not previously been considered. As a consequence of recent results, this implies that there is a family of quantum states that enable universality if and only if they exhibit contextuality with respect to stabilizer measurements. This talk was based on http://arxiv.org/abs/1512.04765.

**Simon Devitt: Topological circuit optimization** Simon discussed several aspects of topological circuit optimization. He started from the fundamental Clifford + $T$ optimization that occurs at higher levels and then moved into what is required for surface codes and 3D Raussendorf codes to integrate the algorithmic structure, ICM representation of a high level circuit and when required classical information in ancillary protocols becomes available. He also talked about probabilistic gate decomposition and what practical downsides they introduce. He outlined a first attempt at this that puts a preference on state distillation protocols and how they are integrated into large scale algorithms. Moreover, the talk was the theoretical counterpart to a software demo that Simone gave in the demo session.

**Vlad Gheorghiu: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3** Vlad presented joint work which investigated the cost of Grover's quantum search algorithm when used in the context of pre-image attacks on the SHA-2 and SHA-3 families of hash functions. The cost model assumes that the attack is run on a surface code based fault-tolerant quantum computer. The estimates rely on a time-area metric that costs the number of logical qubits times the depth of the circuit in units of surface code cycles. As a surface code cycle involves a significant classical processing stage, our cost estimates allow for crude, but direct, comparisons of classical and quantum algorithms. Vlad exhibited a $T$-optimized circuit for a pre-image attack on SHA-256 that is approximately 2149 surface code cycles deep and requires approximately 213 logical qubits. This yields an overall cost of 2162 logical-qubit-cycles. Likewise, we exhibit a T-optimized SHA3-256 circuit that is approximately 2146 surface code cycles deep and requires approximately 216 logical qubits for a total cost of, again, 2162 logical-qubit-cycles. Both attacks require on the order of 2128 queries in a quantum black-box model, hence his results suggest that executing these attacks may be as much as 17 billion times more expensive than one would expect from the simple query analysis. The talk was based on http://arxiv.org/abs/1603.09383

**Markus Grassl: Reversible circuit synthesis based on stabilizer chains** Stabilizer chains are a basic tool in computational group theory. Markus discussed how stabilizer chains and the corresponding transversals can be used to decompose reversible functions into elementary gates such as NOT, CNOT, and Toffoli. He showed several heuristics that allow finding factorizations for any reversible function of reasonable length, though not minimal in general.

**Jungsang Kim: Scalable quantum computing architectures based on trapped ions** Besides being one of the most pristine qubits, the trapped ion technology provides novel ways to gener-

ate entanglement between ion qubits in the system. A technologically feasible architecture could enable construction of ion trap quantum computer systems on which a range of useful quantum algorithms can be effectively executed. Jungsang discussed a scalable system architecture, a preliminary studies on the performance simulation on such an architecture, and experimental progress towards realization of a prototype system.

**Vadym Kliuchnikov: A framework for approximating qubit unitaries** Vadym presented an algorithm for efficiently approximating of qubit unitaries over gate sets derived from totally definite quaternion algebras. It achieves $\epsilon$-approximations using circuits of length $O(\log(1/\epsilon))$, which is asymptotically optimal. The algorithm achieves the same quality of approximation as previously-known algorithms for Clifford+$T$ [arXiv:1212.6253], V-basis [arXiv:1303.1411] and Clifford+$\pi/12$ [arXiv:1409.3552], running on average in time polynomial in $O(\log(1/\epsilon))$ (conditional on a number-theoretic conjecture). The algorithm that Vadym presented is the first such algorithm that works for a wide range of gate sets and provides insight into what should constitute a "good" gate set for a fault-tolerant quantum computer. The talk was based on joint work with Alex Bocharov, Martin Roetteler, and Jon Yard: http://arxiv.org/abs/1510.03888.

**Sandy Kutin: Circuit diagrams with $\langle q|pic\rangle$** Sandy's talk introduced $\langle q|pic\rangle$, which is a system for preparing circuit diagrams in LaTeX, with an emphasis on diagrams used in quantum computing. The user prepares a description of the circuit in the human-readable "$\langle q|pic\rangle$ language": a Python program then converts this into LaTeX code using the TikZ graphics package. Sandy gave a brief overview of the capabilities of $\langle q|pic\rangle$ through a series of examples. The talk was based on joint work with Tom Draper.

**Olivia Di Matteo: Parallelizing quantum circuit synthesis** Olivia presented a method for exact circuit synthesis based on deterministic walks. She applied this method to construct a parallel framework for circuit synthesis, and implemented one such version which performs optimal $T$-count synthesis over the Clifford+$T$ gate set. She presented several examples where parallelization offers a significant speedup on the run time, and breaks previous records for maximum achievable $T$-count.

**Alex Parent: Garbage collection for reversible circuit compilation** When compiling high level programs into reversible circuits in a space efficient way, it is desirable to allow for intermediate cleanup and ancilla reuse. Alex presented a cleanup method based on tracking data dependencies and mutation within a program. Tracking is done using a representation that he calls a "Mutable Dependency Diagram" (MDD). Using this representation, he presented an algorithm for intermediate (eager) cleanup. An incremental cleanup strategy related to Bennett's pebble game was also presented. The MDD representation using the eager cleanup strategy has been implemented as a reversible circuit compiler (REVS). Alex also discussed numerical results based on this implementation.

**Ori Parzanchevski: Trees, buildings, and navigation in the unitary group** Ori explained that several breakthroughs in the analysis of quantum gates in $U(2)$ can be linked to the tree structure of certain subgroups of $U(2)$. When one moves to higher dimensions, these trees are replaced by Bruhat-Tits buildings, which are simplicial complexes with fascinating geometry. Ori explained what these are and how they relate to navigating the group $U(n)$.

**Mathias Soeken: Ancilla-free reversible logic synthesis using symbolic methods** Mathias addressed the problem that optimum synthesis is a difficult task with respect to the number of lines, due to the properties of reversibility. For an irreversible Boolean function it is coNP-hard to find an optimum embedding, i.e., a reversible function with the minimum number of additional lines. Synthesis algorithms exists that obtain from an optimum embedding ancilla-free reversible circuits which have as many circuit lines as variables in the reversible function. However, so far all implementations for such synthesis algorithms require exponential time and space since they operate on the truth table representation of the function. In his talk, he presented alternative implementations of the algorithms based on symbolic methods, which allow to run the algorithm in less space and time for some functions. He also showed that high quality synthesis results for large circuits can be obtained using these implementations.

**Himanshu Thapliyal: Synthesis of quaternary quantum circuits using optimized gate realizations** In this talk, Himanshu proposed a new, enhanced synthesis method for quaternary quantum circuits. He described an algorithm which achieves improved performance results by making use of 11 newly defined quaternary Galois field (QGF) expansions (for a total of 21 QGF expansions). He showed that this algorithm achieves QGFSOP minimization with the assistance of a pseudo-Kronecker Galois field decision diagram (QGFDD). He also discussed performance evaluations against existing works in the literature, showing that his proposed method achieves an average QGFSOP expression product term savings of 32.66%. Toffoli gate realizations from the proposed gate library were found to achieve a minimum quantum cost savings of 39%. Other composite gate realizations either outperformed or matched existing realizations in terms of quantum cost.

**Benoît Valiron: Automated, parametric gate count of quantum programs** In this talk, Benoît turned to the question of logical resource estimation for quantum programs in the quantum programming language Quipper. In the current implementation, gatecount can be obtained automatically for fixed size of circuits. Benoît presented a static analysis tool for obtaining resource estimation parametric on the parameters fed to the program. He discussed the conditions for this gate count to be in closed form.

**Nathan Wiebe: Resource estimates for simulating Nitrogen fixation on a quantum computer** The context for Nathan's talk was the observation that within the last few years improved algorithms, circuits and error bounds have led to substantial reductions in the costs of quantum chemistry simulation. Despite this progress two issues remain. First, the circuits have not been decomposed into a discrete gate set which makes the cost estimates unrealistic for a fault tolerant quantum computer. Second, given the success of classical methods such as DMRG and DFT,

it is unclear whether a practical application exists for these simulation algorithms. In this talk, Nathan addressed both these problems by providing complete cost estimates for simulating the active space of Nitrogenase, which is a catalyst responsible for Nitrogen fixation that cannot be efficiently simulated to within chemical precision using known techniques. He further showed a new family of circuits for simulating quantum chemistry that can dramatically reduce the depth of the simulation.

**Mingsheng Ying: Toward automatic verification of quantum programs** Mingsheng's talk addressed the question of how to build automatic tools for verifying correctness of quantum programs. This is motivated by the observation that programming is error-prone, and especially so when programming a quantum computer or designing quantum communication protocols, because human intuition is much better adapted to the classical world than to the quantum world. He talked about a theorem prover for for verification of quantum programs, based on the logic for verification of both partial correctness and total correctness of quantum programs from his TOPLAS'2011 paper. He also described ongoing work on techniques for invariant generation and synthesis of ranking functions for quantum programs.

# 4 Summary of the software demos

**Matt Amy: ReVer** In this software demo, Matt demonstrated a formally verified, optimizing reversible circuit compiler called ReVer. Although a variety of tools exist which can compile classical, irreversible code into reversible circuits, little effort had previously been spent on verifying these tools, potentially allowing compiler errors to cause incorrect results and negatively affect resource estimation. Rever compiles the simple ML-like language Revs to reversible circuits, with optimizations to reduce the number of ancilla bits used. It uses the dependently typed language F* to verify with machine-checked proofs that ReVer compiles circuits that operate correctly with respect to the input program and cleans all temporary bits. This demo was based on joint work with Martin Roetteler and Krysta Svore, http://arxiv.org/abs/1603.01635.

**Simon Devitt: meQuanics** Simon gave a demonstration of meQuanics, a game for optimizing topological quantum circuits that he is currently developing. He also reported on his current a Kickstarter campaign to crowd-source meQuanics. Each puzzle in meQuanics represents a real quantum algorithm, which needs to be optimized for a quantum computer to be realized with real quantum devices. A fewer number of devices and a shorter run time increase our chance to build a real quantum computer. Reducing the resources necessary for a quantum computer is one of the most urgent and important problems to make this technology a reality. See also https://qis1.ex.nii.ac.jp/mequanics/en/.

**Vlad Gheorghiu: Quantum++** Vlad demonstrated Quantum++, a general-purpose multi-threaded quantum computing library written in C++11 and composed solely of header files. The library is not restricted to qubit systems or specific quantum information processing tasks, but

is capable of simulating arbitrary quantum processes. Vlad also commented on the main design factors that were taken in consideration: ease of use, portability, and performance. See http://arxiv.org/abs/1412.4704.

**Blake Johnson: QGL** Blake gave a demonstration of the QGL programming language, an abstract language with Python-like programming constructs, which can be compiled to existing hardware and used to execute programs (experiments to physicists) on today's qubit systems. He argued that while quantum computers will need high-level, scalable programming languages in the upcoming decades, we need abstract, low-level programming languages now for the sophisticated experiments being performed with current quantum systems. One of the challenges addressed by QGL is that our quantum systems and associated experiments with tens of qubits have outstripped manual specification of the waveforms required to execute these sophisticated experiments. These experiments require sequences of 1000s of gates comprising 10s of waveforms with precise, synchronized execution across the qubits. Further, at the current time, qubit systems are a "sea of gates" over which many architectures are being designed and analyzed and every qubit needs a control signal at every clock cycle. In addition, we do not have the luxury of Von Neumann and other standard architecture concepts that greatly simplify the design of compilers, yet we desire the same abstract programming paradigms. Blake argued that the QGL programming language tackles this challenge. Further, QGL also accommodates the "plug and play" of hardware components (e.g., AWGs and digitizers) with standard driver APIs.

# 5 Tutorials

**Peter Selinger: A tutorial on Quipper** Quipper is a functional programming language for quantum computing. In this tutorial, Peter discussed Quipper's design rationale and gave a practical demonstration.

**Dave Wecker (tutorial given by Martin Roetteler): The LIQUi$|\rangle$ simulator tutorial** Martin presented a tutorial on LIQ$Ui|\rangle$, Microsoft's quantum simulator package that was recently released to the public for academic use. LIQ$Ui|\rangle$ provides a modular software architecture for the simulation of quantum algorithms. It provides a high level interface and is independent of a specific quantum architecture. It runs on Windows, Linux, and OSX as a provided executable with built-in examples and sample scripts as well as a development environment (using Visual Studio or mono, also freely available) that allows the user to compile their own quantum algorithms into an executable. The package includes a User's Manual as well as over 700 pages of API documentation.

This tutorial focused on a number of points. Martin discussed how to obtain and install the package from http://stationq.github.io/Liquid/, and how to obtain and install Visual Studio and mono as development environments. He then showed how to get the system running using built in examples (e.g., Shor's algorithm), how to draw circuits (e.g., Teleport), and he gave a step-by-step demonstration of how to edit, compile, and run circuits. He also discussed the use of scripting (e.g., controlling and creating Quantum Chemistry tests) and gave an overview of the documentation (User's Manual, API docs, Videos, GitHub community).