Theoretical and Practical Aspects of Verification of Quantum Computers

Yehuda Naveh
IBM Research – Haifa
Haifa, Israel
naveh@il.ibm.com

Elham Kashefi
School of Informatics
University of Edinburgh
Edinburgh, UK and
CNRS LIP6
Universite Pierre et Marie Curie
Paris, France
ekashefi@inf.ed.ac.uk

James R. Wootton
University of Basel
Basel, Switzerland
james.wootton@unibas.ch

Koen Bertels
QuTech Research Centre
Delft University of Technology
Delft, The Netherlands
k.l.m.bertels@tudelft.nl

***Abstract*—Quantum computing is emerging at a meteoric pace from a pure academic field to a fully industrial framework. Rapid advances are happening both in the physical realisations of quantum chips, and in their potential software applications. In contrast, we are not seeing that rapid growth in the design and verification methodologies for scaled-up quantum machines. In this work we describe the field of verification of quantum computers. We discuss the underlying concepts of this field, its theoretical and practical challenges, and state-of-the-art approaches to addressing those challenges. The goal of this paper is to help facilitate early efforts to adapt and create verification methodologies for quantum computers and systems. Without such early efforts, a debilitating gap may form between the state-of-the-art of low level physical technologies for quantum computers, and our ability to build medium, large, and very large scale integrated quantum circuits (M/L/VLSIQ).**

***Keywords*—Quantum computing, verification, design automation, very large scale integrated quantum circuits, VLSIQ**

I. Introduction

Quantum computing has undergone an unprecedented shift in recent years. From pure academic sub-fields of physics and computer science, it has emerged as a viable computation scheme with prospects of being able to solve crucial and meaningful tasks in the not distant future [27]. Major advances in cooling techniques and material science have helped bring about this revolution, allowing us to reach coherence times three or four orders of magnitude longer than the basic computation time of a single quantum gate [39]. These advances in the physics of quantum computation have also triggered the development of many application schemes that can naturally exploit of the power of quantum computation [6].

In contrast to the huge advances in physical implementation and in applications of quantum computing, we do not see a similar surge of work in the fields of design and verification methodologies and automation for quantum computers. We feel that without such a surge, we may very rapidly find a debilitating gap between our advances in the low-level capabilities and our abilities to integrate and scale those capabilities into a fully functional system.

Looking at the history of design and verification methodology for classical computers, we see that many currently enshrined best practices evolved through lengthy processes of trial and error. They emerged from painful gaps in the ability of design and verification automation to keep pace with low-level technology advancements. When we look forward into a world with practical quantum computers, we should learn from this experience in two ways. First, we should understand that significant amounts of attention, resources, and rigour need to be devoted to design and verification methodologies as early in the process as possible, even if initially it appears that the actual hardware is small in number of bits and simple in its complexity. Second, we should closely look at the processes leading to our current classical design and verification methodologies, and attempt to learn and find their parallels in quantum computation. This will allow us to adopt the classical methodologies where they are relevant to the quantum case, even if major changes are needed on the way. It will also allow us to identify the gaps where entirely new quantum design and verification methodologies need to be invented. This will help us avoid stumbling into the same pitfalls that classical computing design and verification encountered in its long and pioneering history.

The purpose of this paper is hence to bring to the front of the design automation community the theory, the major challenges, and the state-of-the-art solutions, in the verification of quantum computers. As we will show, some of the verification aspects of quantum computers are inherently very different from the classical case. Still, the overall goal is the same — to verify that the computer provides the results according to its specification. This dichotomy between the similar goal and the sometimes very different challenges means that only wide expertise in classical verification methodologies, combined with deep levels of new research, will bring us to the point where we are fully ready for industrial-level verification tasks of quantum computers.

We organise the paper as follows: In the next section we provide broad background of the field of quantum computation. Section III dives into the specialised field of quantum error correction, which is one of the most pervasive topics in quantum computing and its verification. Section IV provides some of the theoretical tools for dealing with quantum verification, and state-of-the-art approaches in using those tools. Section V maps the methodologies for verification of quantum computers,

978-3-9819263-0-9/DATE18/©2018 EDAA

and draws initial analogues between classical and quantum methodologies. Section VI concludes this paper.

II. Background

Research on quantum computing started in 1982 when Richard Feynman suggested to use a quantum system to simulate another quantum system [18]. The basic idea is to exploit two fundamental phenomena of quantum mechanics, superposition and entanglement. Together, this results in a new computation scheme, allowing the polynomial solution of classes of problems that are currently intractable by classical computers [36].

Superposition A classical bit has two exclusive states, 0 or 1 and can only be in one state at any point in time. In contrast, the elementary unit of quantum computers, the quantum bit or qubit, can reside not only in a single basis state $|0\rangle$ or $|1\rangle$ but also in a superposition of both states, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. $\alpha, \beta \in \mathbb{C}$ are probability amplitudes that satisfy $|\alpha|^2 + |\beta|^2 = 1$. $|\alpha|^2$ and $|\beta|^2$ represent the probability of getting the measurement result $+1$ or -1, corresponding to states $|0\rangle$ or $|1\rangle$ respectively, when measuring the qubit in the computational basis. The act of measuring the qubit will project the state of the qubit onto one of the basis states, implying that a quantum state cannot be measured directly without losing the stored information.

Entanglement and quantum interference In classical computing, a system composed of n classical bits can only store and process one of the 2^n possible states at a time. However, in quantum computing, multiple qubits can be combined to form a single state, not separable into individual-qubit states. This is called entanglement. Together with superposition, this means that our most general state becomes $|\psi\rangle = \alpha_0|0\cdots00\rangle + \alpha_1|0\cdots01\rangle + \cdots + \alpha_{2^n-1}|1\ldots11\rangle$, where $\alpha_i \in \mathbb{C}$ and $\sum|\alpha_i|^2 = 1$. In contrast to classical probability theory, the α_i's are probability amplitudes, not probabilities, meaning that they can take negative values. A quantum algorithm may then sum up states with positive and negative amplitudes (or more generally, states with any phase difference between them) for any entangled set of bits, resulting in a smaller absolute value of the amplitude of the final state. This destructive interference phenomena, unheard of in classical computation — not even in probabilistic, analog, or parallel computations, is presumably what gives quantum algorithms their extra power.

A **quantum gate** can change the state of qubits. As the qubit state can be represented by a vector, a quantum gate, which is a reversible operation by nature, can be represented by a unitary matrix. The simplest quantum gate is that acting on a single qubit. For example, the identity and three Pauli matrices

$$I \equiv \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \quad X \equiv \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \quad Y \equiv \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} \quad Z \equiv \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \tag{1}$$

represent four common single-qubit gate operations: I represents a wait-gate operation applied to the qubit. X represents a bit-flip operation, as after being applied with an X gate, the qubit state transforms from $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to $|\psi'\rangle = \beta|0\rangle + \alpha|1\rangle$. Similarly, a Z gate represents a phase flip for the qubit, transforming its state to $|\psi\rangle = \alpha|0\rangle - \beta|1\rangle$.

A common multi-qubit gate is CNOT acting on two qubits. The second (target) qubit will bit-flip conditioned on the first (control) qubit:

$$\text{CNOT} \equiv \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}, \tag{2}$$

transforming the two-qubit entangled state $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ to $|\psi'\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|11\rangle + \delta|10\rangle$.

Any quantum gate acting on n qubits can be represented as a $2^n \times 2^n$ matrix. If a gate is applied on a subset of n qubits, all the 2^n complex numbers in the state will change accordingly [36]. This implies extremely large intrinsic parallelism. Just as for classical gates, there are also universal sets of quantum gates. An example is $\{H, T, \text{CNOT}\}$, where

$$H \equiv \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \quad \text{and} \quad T \equiv \begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}. \tag{3}$$

Hence, superposition, entanglement, and quantum interference, together with the possibility of operations through a universal set of gates, lay the foundation for the universality and the exponential speedup of quantum computer.

Even though the potential of quantum computing is huge, the Achilles heel of quantum technology is the fragility of the qubits. Through interaction between the qubits and the environment, the information in the qubits leaks into the environment, in a process called *decoherence*. The qubits' fragility is therefore one of the main challenges for building and using a quantum computer as this behaviour causes errors during computation. Quantum error correction (QEC) mechanisms are needed to make quantum computing fault-tolerant and is therefore a key component of any quantum computer architecture, as will be discussed in section III.

The challenges to build a circuit-model-based quantum computer – commonly referred to as the standard universal quantum computer – are huge. One of the main physics challenges is to increase the number of qubits per chip that can be entangled as well as to make their lifetimes longer and their operation fidelity higher. Several physical systems are being explored for quantum computing such as photons, quantum dots, trapped ions, and superconductors. Current state-of-the-art quantum chips contain around 10 qubits [21, 28, 16, 9, 48, 42]. As superconducting qubits are seen as one of the most promising technologies, we assume this for the rest of the paper [21, 28, 16, 13]. The engineering challenges focus on the technology necessary to provide high speed control logic in a way which is scalable, flexible, and with high fidelity. This paper focuses on the challenges related to errors appearing in the computation process, ways to avoid or correct them, and methods for verifying overall correctness after taking into account the inherent presence of errors.

III. Quantum Error Correction

A. Overview

Managing noise in quantum systems is arguably the biggest challenge for building fault-tolerant and scalable quantum

computers. Indeed, the majority of all resources in any fault-tolerant quantum computation will be dedicated to this task.

For example, let us consider the usual paradigm of quantum error correction [30]. This requires the repeated application of a process to extract syndrome information. The required process involves all qubits, and requires readout of a significant fraction of them. The repetition of this process must be constant, with each round completed as fast as possible. Furthermore, the number of *physical* qubits required will be vastly greater than the number of *logical* qubits: the quantum information of which we wish to encode reliably.

Implementing this process will be a massive technical challenge. As will the handling of the vast amounts of classical information that it extracts, which must be processed to determine what errors occurred and how to compensate for their effects. And all this computational effort is required to simply *store* the logical qubits.

To perform quantum computation with all additional physical qubits, we can use a method such as *code deformation* [11]. This makes slight modifications to the error correction procedure from one round to the next. The small changes do not affect the fault-tolerance of encoded qubits, but they do change the nature of the encoding. The cumulative effect can be to implement the basic operations required to process the encoded quantum information.

It is therefore no overstatement to assert that quantum error correction is essentially all that a fault-tolerant quantum computer will do. Computation can then be regarded as a bonus feature that we can trick the error correction process into implementing.

Of course, computation is more than just a side effect. It is the reason we are doing quantum error correction in the first place. Methods of verification based on the details of how the encoded logical qubits are processed are therefore necessary to ensure the high-level functionality of a quantum device. But to probe the vast majority of what a fault-tolerant quantum computer does, we can look to quantum error correction.

Many experiments have already been done based on tasks in this area. This includes proof-of-principle demonstrations of error detection [13, 31, 44, 41], as well as correction of a limited set of errors [28, 46]. More basic tests of the required techniques have also been performed for state preparation [37], syndrome measurement [40] and code deformation [47].

Despite these advances, no results have yet been published that demonstrate a logical qubit whose fidelity is higher than the physical qubits in which it is encoded. However, a necessary condition for this has been achieved: it has been shown that a logical *bit* encoded in a physical qubit can be protected using the techniques of quantum error correction [28, 46].

These experiments used a quantum implementation of the *repetition code*. Having no need to maintain arbitrary superpositions of the states $|0\rangle$ and $|1\rangle$, this is much simpler and more adaptable than most quantum error correcting codes. This allows it to be realised on devices too small, or whose connectivity is too limited, to implement other codes [21, 9]. Nevertheless, it uses all the basic techniques of quantum error correction, and so serves as a basic test of its effectiveness.

The results of these experiments allow us to probe the two standard assumptions used for quantum error correction: that the probability of an error on the logical information decays exponentially with the size of the code, and that it increases no faster than linearly with the number of error correction rounds.

The experiment of Ref. [28] considered two sizes of repetition code: one with a total of 5 physical qubits, and one with a total of 9. The effectiveness of these was then analysed over a total of eight syndrome measurement rounds. The results showed that the fidelity of the stored bit was significantly better than that of a bit stored in a single physical qubit for all cases. The limited number of code sizes meant that the exponential decay of failure rate with size could not be probed, though increasing size was found to decrease the failure rate significantly. The large number of rounds allowed the scaling of many rounds to be studied. It was found that the increase of logical error probability did not sharply deviate from linear scaling, but some evidence of faster than linear scaling was found. Since the dynamics of error processes are known to lead to non-Markovian effects, this result is not unexpected. However, it does serve to underline the importance of further experimental and theoretical studies of how such effects will affect quantum error correction.

The experiment of Ref. [46] considered the other extreme: Only a single round of syndrome measurement was applied before final readout. This is due to the fact that post-measurement access to a qubit can come with significant technical challenges. The measurement itself also requires long timescales in comparison to other operations. For example, measurements in the experiment of Ref. [28] took over 500 nanoseconds (ns), whereas other operations required no more than 50 ns. Such challenges will be addressed in due course, but current development of prototype quantum devices often focuses on expanding the number of qubits that can be achieved, rather than the number of rounds of measurements and classical feedback. Experiments with a single syndrome measurement round will therefore be more accessible in the short term.

Though the experiment of Ref. [46] used only a single syndrome measurement round, codes of many different sizes where studied, with total physical qubit numbers ranging from 5 to 15. The results are therefore useful to assess the decay of the logical error probability with code size d. The results from this experiment are shown in Fig. 1. The results are indeed consistent with an exponential decay though, again, results from a finite interval such as this are far from conclusive. As devices get ever larger, repetition codes of larger sizes must be probed to ensure that this exponential decay remains.

B. Single round repetition code

Let us now consider the single round repetition code experiment [46] in more detail, as a concrete example of a test of quantum error correction.

When implemented classically, the repetition code is the simplest form of error correction. The encoding step is achieved by simply copying the information across multiple qubits. The decoding step is done by majority voting.

Let us use d to represent the number of repetitions. Consider a $d = 5$ instance of this code with which we wish

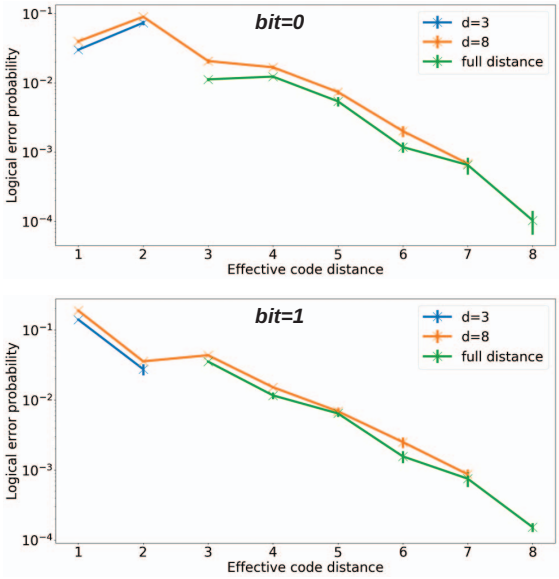

Figure 1. Logical error probabilities for encoded bit values 0 and 1. d is the actual repetition code size (see discussion in Section III-B).

to encode the bit value 0. This is done by preparing the string 00000. If errors cause some of these bits to flip, resulting in an output such as 01001, the 0 value is still in the majority. Decoding would then recover the stored value successfully. If one more error had occurred, however, giving an output such as 01101, the value 1 would be in the majority. This would cause the decoding to obtain the incorrect value of 1. Such a mistake is referred to as a logical error. The probability of logical errors occurring will be the main figure of merit used to assess the effectiveness of the code.

To minimise the probability of a logical error over an extended period of time, the decoding can be applied continuously. For a classical implementation, this can be done by continuously reading out the values of each bit, finding the minority and flipping to the majority value. This would not only give information useful in determining where errors have occurred, but also provide enough information to read out the stored information during each round. Such behaviour is not allowed in a quantum implementation. Though only a *bit* of information can be protected, we must still treat it as if it were a stored *qubit* in order to provide a test of the techniques of quantum error correction. Extracting information regarding a logical qubit during error correction would result in changes to the quantum state, which would themselves be errors. Quantum error correction must therefore be careful to extract information only about the errors, with no logical information revealed until final readout.

A process that can be used for this is shown in Fig. 2, which represents a quantum circuit. On the left hand side, all qubits are initialised in state $|0\rangle$. On the right hand side, the qubits are measured, resulting in an output of 0 or 1 for each.

The qubits are labelled alternately as *code qubits* and *ancilla qubits*. The former are those on which the bit value to be stored is repeated. This example shows a stored bit value 1, achieved by applying the bit-flip X gate to all code qubits.

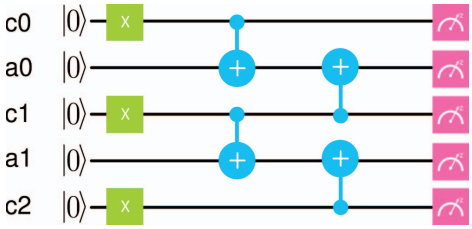

Figure 2. Circuit for quantum repetition code with five physical qubits. This example shows an encoded bit value of 1.

The ancilla qubits are left in state $|0\rangle$ in all cases.

To extract information about errors, two rounds of CNOT gates are applied. Here, the value of the 'target' qubit (that connected to the gate with a large '+' dot) is replaced by the XOR of the two inputs. The ancilla qubits serve as the target in all cases, with their two neighbouring code qubits as control. Given the initial $|0\rangle$ value of the ancilla bits, the end effect is for each ancilla state to represent the XOR of its neighbouring code qubits. By the nature of the repetition encoding, the code qubits should always have equal value when all is well, and so each such XOR should always return 0. Measuring an ancilla to be in state 1 is therefore a signature that an error has occurred. This is the syndrome information, which tells us about errors without revealing any of the encoded information. With this we can attempt to infer the error and mitigate its effects. This process can be repeated many times in general, but we consider only one round in this case.

After the syndrome measurements, final readout is performed. This includes a measurement of the code qubits, and therefore extracts the encoded logical information as well as syndrome information.

The whole process shown in Fig. 2 for a $d = 3$ code can be straightforwardly scaled up to larger cases. Note that the code requires d code qubits and $d - 1$ ancilla qubits, and so $2d - 1$ in all.

For a scalable fault-tolerant quantum computer, the decoding process will be done with fast algorithms which allow for constant feedback and control on the code qubits. However, due to the limited size of the problem in the experiment of Ref. [46], lookup table decoding can be used instead. This table can be populated with experimental data, allowing decoding that is tailored to the exact nature of noise in the device.

Specifically, a large number of repetition code instances are run for the purpose of populating the lookup table. This consists of large number of runs in which the bit value 0 is encoded, and the same for an encoded 1. In each case, the number of times each possible $2d - 1$ bit output occurs is noted. Assuming that two possible bit values occur with equal probability, this information directly forms the lookup table.

Whenever an instance of the repetition code is run later, and its output needs to be decoded, we simply look at which encoded value had that output occur the most in the lookup table data. This is then assumed to be the value that was encoded for the current instance. If this is not correct, the output of the decoding is not the same as the original encoded value. This is a logical error. By running many repetition code samples, we can estimate the probability of this occurring.

For example, the output 00000 is more likely to occur for a $d = 3$ code with encoded 0 (for which it represents the case of no errors) than for encoded 1 (for which it would result from errors on all code qubits). This result is therefore decoded as 0. This will most likely be the correct decoding, but since it is possible for the 00000 output to occur for an encoded 1, this decoding will sometimes lead to logical errors.

It should be noted that this decoding procedure could also be performed using only the outputs of the code qubits. This could lead us to question why the ancilla qubits are required at all. The answer is simply that it would not be an instance of quantum error correction otherwise. With the full syndrome measurement round in place, the process contains all the techniques of quantum error correction. Implementation and analysis of this then allows us to verify that these processes work as they should, and that the imperfections in the qubits, measurements, and gates do not cause fatal problems within the scope that the experiment considers.

The main point to verify is that the logical error probability decays exponentially with system size. The reason why this is expected can be seen from considering only the code qubits. Errors must flip the majority of bits, at least $\lceil d/2 \rceil$, in order to confuse the decoding. If such errors occur independently with probability p, the probability of such events scales with $p^{\lfloor d/2 \rfloor}$ (note that this is a slight oversimplification for clarity, neglecting entropic effects). Though detailed analysis is left to Ref. [46], such exponential scaling can be seen in Fig. 1, obtained from a 16 qubit IBM device [26].

Fig. 1 also contains additional data not presented in Ref. [46]. This data is available from the source code for the experiment, available at [45]. It shows the effect of truncating a code. For example, consider a $d = 9$ code run in the normal way. However, when it comes to decoding the final readout, the results for the last code and ancilla qubit are ignored. This effectively leads to a $d = 7$ code. Ignoring the next two qubits would give a $d = 5$ code, and so on. This allows us to compare a normally implemented code with a given d with a larger one truncated down to d.

It can be expected that the truncated code would perform slightly worse. This is due to the final code qubit being involved in a CNOT whose results are not used. The negative effects of its imperfections would then outweigh any positive effects of the information it obtains. However, if the truncated code were to perform significantly worse, it would raise questions as to how noise is distributed through the code by the CNOT gates. The presence of such non-local effects could be highly dangerous for quantum error correction.

Fortunately, the results in Fig. 1 suggest the former. Results from a truncated $d = 8$ code and a truncated $d = 3$ code are shown. They show good agreement to each other at those points at which they coincide, as well as good agreement to results from full codes.

The description here of quantum error correction and its verification has been brief and qualitative. More details can be found in the references given, but many more details are also yet to be fixed. There is much that can be explored, and new methodologies to construct. For members of the classical verification community who wish to contribute, exploring the source code of this repetition code experiment could be a good place to start [45].

IV. Theory of Quantum Verification

Global Vision

Over the next five to ten years we can expect to see a state of change as quantum technologies become part of the mainstream computing landscape. Quantum computing machines are likely to enter the market with high variability in terms of architectures and capacities. Adopting and applying such a highly variable and novel technology may be both costly and risky for any individual company or research group. This is exacerbated by the fact that the quantum approach has an acute verification and validation problem. First, since classical computations cannot scale-up to the computational power of quantum mechanics, verifying the correctness of a quantum-mediated computation is challenging. Second, the underlying quantum structure resists classical certification analysis. The required experiments are not beyond our reach and have been implemented, but these are quantum experiments simulating quantum theory. So even if we assume the correctness of quantum theory, we are not able to verify the experimental result due to the superior computational capacity of quantum systems. *What makes quantum not classical, makes its verification not classical either*. Solving this challenge is a key milestone on the way to make the translation from theory to practice possible.

***Encrypted Verification**:* The ability to compute with encrypted data, while hiding the underlying function, has opened a new approach toward verification, through the detection of a cheating server [10, 5, 38, 20, 8, 22]. When a user wants to compute the solution to a classical problem in NP, he or she can efficiently verify the result provided by a server. But a dishonest server is not so easy to detect in other cases such as quantum simulation [1] or Bounded-error Quantum Polynomial time (BQP) problems [14]. The challenge is to mimic a similar construction where an efficiently testable witness can guarantee the correctness of the entire computation. We have shown, as a proof-of-principle, that one can *bootstrap* a small quantum device to test a bigger one [20]. The path forward is to adapt these generic verification techniques to the specific architecture and constraints of various hardware platform that are emerging as quantum technology matures.

Generally speaking there are three levels of (quantum) verification, all of which are important and require to be tailor-made for the task in hand. The first level is testing the devices in a setting where we trust the parties involved. Techniques for randomised benchmarking and certain hypothesis testing belong to this part. The second level is when we want to verify a computation or certain fundamental property without trusting our devices. Post-hoc verification and hypothesis testing for quantum advantage scenarios belong here. Finally, one could exploit verification techniques based on performing a hidden computation. In this view, verification defines an interface platform that enables one to deal at the same time with the practical experimental restrictions as well as effect of noise on the desired applications.

***Interactive Proof System**:* In trying to address the verification challange we return to complexity theory. The primary

complexity class that we are interested in is BQP, which is the class of problems that can be solved efficiently by a quantum computer. We are willing to allow our notion of verification to include interactive communications between the user and the device, leading to a family of protocols known as an *interactive-proof system*. Such a protocol consists of two entities: a *verifier* and a *prover*. The verifier is a Bounded-error Probabilistic Polynomial time (BPP) machine, whereas the prover has unbounded computational power. Given a problem for which the verifier wants to check a reported solution, the verifier and the prover interact for a number of rounds which is polynomial in the size of the input to the problem. At the end of this interaction, the verifier should accept a valid solution with high probability, and reject, with high probability, an invalid solution. The class of problems which admit such a protocol is denoted Interactive Polynomial time (IP). It is known that BQP $\subseteq$ IP, which means that every problem which can be efficiently solved by a quantum computer admits an interactive-proof system. One would be tempted to think that this solves the question of verification, however, the situation is more subtle. Recall that in IP, the prover is computationally unbounded, whereas for our purposes we would require the prover to be restricted to BQP computations. Hence, the question that we would like answered and, arguably, the main open problem concerning quantum verification is the following:

(Verifiability of BQP computations). *Does every problem in BQP admit an interactive-proof system in which the prover is restricted to BQP computations?*

This complexity theoretic formulation of the problem was considered by Gottesman, Aaronson and Vazirani [2, 43] and, in fact, Scott Aaronson has offered a 25$ prize for its resolution [2]. While, as of yet, the question remains open, one does arrive at a positive answer through slight alterations of the interactive-proof system. Specifically, if the verifier interacts with two or more BQP-restricted provers, instead of one, and the provers are not allowed to communicate with each other during the protocol, then it is possible to efficiently verify arbitrary BQP computations [38, 23, 25, 33, 19, 34, 12]. Alternatively, in the single-prover setting, if we allow the verifier to have a constant-size quantum computer and the ability to send/receive quantum states to/from the prover then it is again possible to verify all polynomial-time quantum computations [5, 20, 22]. Note that in this case, while the verifier is no longer fully classical, its computational capability is still restricted to BPP since simulating a constant-size quantum computer can be done in constant time. These scenarios are depicted in Figures 5.

The primary technique that has been employed in most, thought not all, of these settings, to achieve verification, is known as *blindness*. This entails delegating a computation to the provers in such a way that they cannot distinguish this computation from any other of the same size, unconditionally. In other words, the provers would not be able to differentiate among the different computations even if they had unbounded computational power. Intuitively, verification then follows by having most of these computations be *tests* or *traps* which the verifier can check. If the provers attempt to deviate they will have a high chance of triggering these traps and prompt the verifier to reject.

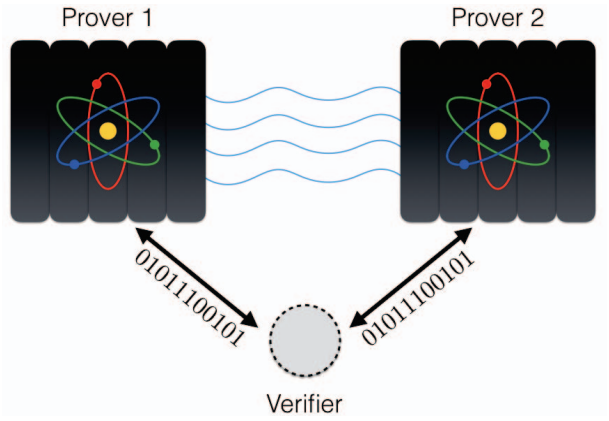

Figure 3. Classical verifier interacting with two entangled but non-communicating quantum provers

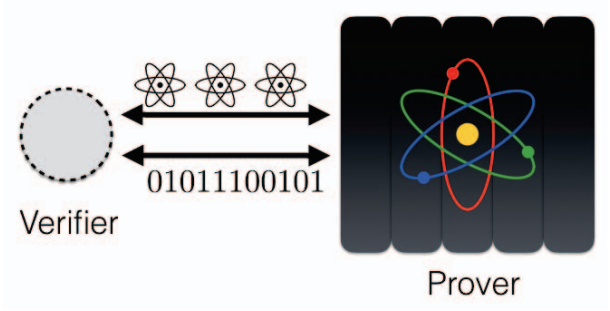

Figure 4. Verifier with the ability to prepare or measure constant-size quantum states interacting with a single quantum prover

Figure 5. Models for verifiable quantum computation

A full review of a number of verification protocols could be found in [22]. While none of these achieve the ultimate goal of the field, which is to have a classical client verify the computation performed by a single quantum server, each protocol provides a unique approach for performing verification and has its own advantages and disadvantages. These protocols combine elements from a multitude of areas including: cryptography, complexity theory, error correction, and the theory of quantum correlations. Proof-of-concept experiments for some of these protocols have already been realised. What all of the surveyed approaches have in common, is that none of them are based on computational assumptions. In other words, they all perform verification unconditionally. However, recently, there have been attempts to reduce the verifier's requirements by incorporating computational assumptions as well. What this means is that the protocols operate under the assumption that certain problems are intractable for quantum computers. We can therefore see a new direction emerging in developing protocols based on computational assumptions. This could very well lead to the first single-prover verification protocol with a classical client.

V. Verification Methodologies

The verification of practical quantum computers opens up an entirely new field of research. The theoretical ideas and experimental approaches, as outlined in this paper, will remain

at the basis of this field. However, as we are entering an era in which the scale and complexity of quantum hardware is expected to grow at a rapid pace, much more seemingly mundane aspects would also need to be addressed. In this section, we follow the evolution of classical verification methodologies, and view the possible quantum methodologies in light of this classical evolution.

At the lowest level, each qubit and each gate need to be verified for correct functionality. In other words, we need to verify that the quantum state is maintained, or more realistically, to determine average times by which the quantum state loses coherency. A set of generic methods which are designed to examine and verify the quantum state directly go under the name of **quantum tomography** [15]. These are well-defined, mathematically rigorous techniques of evaluating the quantum state across the entire state space. Unfortunately, this type of techniques is likely to remain purely academic (or at least confined to the design stages where only a very small number of bits and gates are of interest), because any application of these techniques requires an exponential number of resources.

Going one level above quantum tomography is **randomised benchmarking** [29, 32]. Here sequences of quantum gates are applied in such a way that their end result is known (typically, the last gate applied is used to reverse the entire sequence so the end result is just the initial state). Technically, this can be achieved by using gates chosen from a restricted set of gates which allow exact and tractable calculation of the result of the sequence [3]. By applying increasingly longer sequences and checking whether end-of-test is as expected as a function of sequence length, one may calculate the typical time scales by which coherency is lost (and hence results not corresponding with calculations). Moreover, by designing sequences with some specific properties (e.g., higher percentage of gates of a particular type), one can gain more structured insights into the behaviour of specific gates or combination of gates.

Today, random benchmarking is generally designed in a manual fashion. The designer wishes to test a particular aspect of the design, whether in the hardware implementation of qubits, or the physical process of applying gates, and designs benchmarking sequences which would relate to the issue under hand. Also, continuous validation of the chips and gates is performed by carefully designed sequences of random benchmarks. We can view this process as reminiscent of directed testing of classical hardware designs. Here also, the designer or low-level verification engineer designs specialised sets of tests to examine specific prone to bugs behaviours. This activity, by nature, requires a large amount of effort by experts. In addition, such manual design of tests may still miss some areas of the design that have not been thought of. In classical computing, the field of **constraint-based random test generation** [4, 35] has emerged to address precisely these two concerns. Here, the expert is given a much higher-level modelling language to model his or her intentions. Then software tools, based on sophisticated algorithms, automatically generate large numbers of random sequences, all biased towards the hints taken from the user, but cover a very large set of diverse possibilities within those biases. We certainly expect the important field of quantum random benchmarking to evolve similarly. This will reduce much of the required expert knowledge to a one-time effort in building the model and in constructing an input language for specifying constraints and wishes. After that, the expert would just need to specify their wishes for a particular set of runs, and let the machine produce large and smart sets of random benchmarks all conforming to those wishes.

A natural next step in the evolution of constraint-based automatic random benchmarking is the definition of tests at higher-levels than the specification of gate sequences applied to one or two qubits in parallel. Once we deal with a number of parallel sequences, all applied to different qubits on the same chip, we can start thinking in terms of program constructs. Then a modelling language for those constructs should be in place. Such a language should be designed with two aspects in mind. It should have the ability to resemble common programming idioms in order to be able to check realistic scenarios. But even more importantly, it should have the ability to naturally specify common or expected prone-to-bugs scenarios. For example, ability to easily specify some specific interaction between two or more parallel sequences running on adjacent qubits. Moreover, the language should be flexible enough to be able to adopt to newly discovered scenarios, while all the time being at sufficiently high level for the designer to be readily able to specify the scenario he or she has in mind. The combination of a **high-level test specification language** and an automatic constraint-based random test generator would allow for major sets of benchmarks to run with very high efficiency. This is because designer time would now be used much more effectively in specifying the test scenarios, so many more scenarios can be created. In addition, the automatic generator would make sure not to waste too much of its tests on scenarios which were already covered by other tests.

This brings us to the notions of **coverage** and **coverage measurement** [24]. Once the designer is accustomed to thinking in high-level modelling constructs, then not only he or she can specify their intentions for test scenarios in those constructs, but can also measure how well have the tests covered their entire set of wishes. Of course, the modelling of coverage measures need not be at the same level as the test-specification constructs. For example, the designer may specify the test such that random sequences of gates rapidly exercise a number of qubits in the chip, and can even specify a large percentage of CNOT gates in the sequences, but then measure coverage at the gate (or even the pulse) level, checking, e.g., that a CNOT gate was applied simultaneously to all possible combinations of adjacent pairs of qubits on the chip. Once coverage models become part of the methodology, they can become increasingly more sophisticated, as the team learns from experience about additional, possibly more subtle, prone-to-bugs scenarios. Additionally, once coverage-based test generation is in place, substantial shifts in overall verification methodology may take place. For example, most verification efforts at some point in time may be shifted to one particular hard-to-cover (but important) coverage hole, rather than continue across the board verification. Additionally, verification time-lines may change and become more dynamic, driven by coverage goals of the overall project, rather than by extrapolation from past experience.

In the random-benchmarking scheme described above, simple end-of-test checking is the tool for ensuring correctness. However, once testing methodology becomes more complex,

one can think of more elaborate checking mechanisms. Most importantly is **dynamic checking**. Here, one can envision checking mechanisms which are inserted within the sequence implementing the tests and being checked online as part of the sequence. This may allow for much higher flexibility in designing the tests, in more powerful identification of issues (i.e., wrong behaviour can be found even if would have been masked before reaching end of test), and in their faster identification. However, compared to the other concepts outlined above, implementation of on-line checking mechanisms may prove to be more challenging. This is because the previous concepts are supported by only software design and implementation, while mechanisms for online checking may require interfering with the actual gate transmission mechanisms, or even the hardware chip.

A higher level of checking can be thought of as **formal verification** [17]. Here, a one-time model is created, specifying all known relationships between static and dynamic properties of the chip. Once this model is expressed in a formal mathematical language, mathematical properties of the design (including the gate implementation) can be checked against this model. Dynamically, properties may be modelled per 'cycle' of the design (i.e., before and after each application of a gate). Depending on the sophistication of the model, complex scenarios can be analysed for covering wide areas of the design functional space without running any actual physical gate. If applicable, this method can prove to be especially powerful in the quantum world, where real-life experiments tend to be noisy and not necessarily conclusive. However, in order for such formal methods to have any added benefit, very detailed mathematical models of the chip must be created. Those mathematical models must be created in a way that any calculation of properties would still be tractable. This by itself can prove to be tricky for quantum computation scenarios, as the computation itself is non-tractable. So clearly, the mathematical model must be at some level of approximation, useful enough to prove non-trivial properties of the design, but abstract enough to be able to disregard details leading to intractability. Still, if such a model can be found, its prospects in supporting formal verification of the design may be high. This is because, at least for the time being, we are expected to deal with a relatively small number of qubits and gate sequences compared to the millions of bits dealt with in classical formal verification. This means that we may be able to expect run-times of formal verification tools which may be much shorter than the sometimes prohibitive times reached when verifying classical hardware.

Quantum hardware is today simulated at two different, and extreme levels. First is the physical level, used to aid in actual physical design of the chip. Second is the qubit and gate level, which allow for simulating any given program on a classical machine, provided of course the number of qubits is small enough to allow for the exponential calculation inherent in the simulation (or, alternatively, resort to approximate simulation). One main reason for such simulation is in order to understand physical noise mechanisms in the hardware. Hence, noise models are incorporated into the simulator, and are then simulated in order to check how well such models correspond with the actual hardware behaviour. A large body of knowledge in modelling and analysing **soft errors** in classical computers may be relevant here [7]. While soft errors happen on very rare occasions in classical machines, modelling their behaviour and in particular their effect (whether transient or not) on the overall program running on the machine is critical for the machine's operation. Such harm-analysis of soft errors may prove relevant to the simulation analysis of decoherence and other types of noise in quantum machines, both with and without error correction mechanisms.

Finally, today's quantum simulators are adequate for present day situation of a relatively small number of qubits. This is because the correctness of the logic itself can be inferred by design and manual inspection. However, once the number of qubits becomes sufficiently large, and their connectivity sufficiently complex, **logic-level simulators** will need to be built. Here, the simulator will accept the design plan as an input, and tests will need to be created and simulated over the design plan. The output of such logic-level simulator is then passed to a checker designed to check whether the simulated behaviour is as specified. Building such logic-level simulators and checkers requires a few levels of formality. For example, the specification of what the combined hardware and gates are expected to do needs to be expressed in such a way that an automatic checker could use. While such logic-level simulators are commonly built and run for classical hardware, and vast amount of experience has been obtained in writing and checking classical logic-level specifications, not all of this knowledge can be taken as-is to the quantum regime. As one simple example, the quantum hardware by itself does not implement the logic. For this dynamic gates are applied over the circuit. So the question of how to define correctness of the quantum chip is an interesting research area by itself. This research must be completed before we can expect to build reasonable logic-level simulators and checking mechanisms — in turn, an important prerequisite for building medium, large, and very large scale integrated quantum circuits, or M/L/VLSIQ.

VI. Conclusions

We have shown in this work the entire spectrum of thought related to verification of quantum computers. Going from the pure theoretical aspects, namely, of whether and how a result obtained on a quantum machine can be independently verified, through computational aspects and insights obtained from actual experimentation on a quantum computer, and all the way to the methodologies needed for the verification of scaled-up quantum computers. In all parts of this spectrum, the unknowns overshadow the known. This is what gives the topic a huge potential for ground-breaking research. Important results in this field are likely to influence both the theoretical thinking, and the practicality and quality of future quantum computers. With very strong teams working at the forefront of research both in the quantum computing community, and the design automation community, we hope that this paper will help facilitate cross-education and collaboration between the teams, resulting in rapid and high quality advancements in the crucially important field of quantum verification.

Acknowledgements

JRW is supported by the SNSF through the NCCR QSIT. JRW acknowledges use of the IBM Q experience for this work. The views expressed are those of the author and do not reflect

the official policy or position of IBM or the IBM Q experience team. KB acknowledges funding by Intel in the context of the QuTech-Intel collaboration. YN is grateful to Yael Ben-Haim, Jay Gambetta, Gil Shurek, John Smolin, and Chris Wood for many important discussions.

REFERENCES

[1] S. Aaronson and A. Arkhipov. "The computational complexity of linear optics". In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM. 2011, pp. 333–342.

[2] Scott Aaronson. *The Aaronson $25.00 Prize*. http://www.scottaaronson.com/blog/?p=284.

[3] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Phys. Rev. A* 70.5 (2004), p. 052328. eprint: arXiv:quant-ph/0406196.

[4] Allon Adir et al. "Genesys-pro: Innovations in test program generation for functional processor verification". In: *IEEE Design & Test of Computers* 21.2 (2004), pp. 84–93.

[5] D. Aharonov, M. Ben-Or, and E. Eban. "Interactive Proofs For Quantum Computations". In: *ICS*. 2010, p. 453.

[6] *Applications of quantum computing*. https://www.research.ibm.com/ibm-q/learn/quantum-computing-applications/. Accessed: 2017-11-30.

[7] Eli Arbel et al. "Automated detection and verification of parity-protected memory elements". In: *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press. 2014, pp. 1–8.

[8] S. Barz et al. "Towards the experimental verification of a quantum computer". In: *Submitted to Nature* (2013).

[9] X. Bermudez *et al.* "Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation". arXiv:1705.02771. 2017.

[10] A. Broadbent, J. Fitzsimons, and E. Kashefi. "Universal Blind Quantum Computing". In: *FOCS*. 2009.

[11] Benjamin J. Brown et al. "Poking Holes and Cutting Corners to Achieve Clifford Gates with the Surface Code". In: *Phys. Rev. X* 7 (2 May 2017), p. 021029. DOI: 10.1103/PhysRevX.7.021029.

[12] Andrea Coladangelo et al. "Verifier-on-a-Leash: new schemes for verifiable delegated quantum computation, with quasilinear resources". In: *arXiv preprint arXiv:1708.07359* (2017).

[13] A.D. Córcoles et al. "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits". In: *Nature Communications* (Apr. 2015), p. 6979. DOI: 10.1038/ncomms7979. URL: https://www.nature.com/articles/ncomms7979.

[14] D. Aharonov, V. Jones, and Z. Landau. "A Polynomial Quantum Algorithm for Approximating the Jones Polynomial". In: *STOC*. 2006.

[15] G. Mauro Dariano, Matteo G. A. Paris, and Massimiliano F. Sacchi. "Quantum Tomography". In: *Advances in Imaging and Electron Physics* 128 (2003), p. 205.

[16] G de Lange et al. "Realization of microwave quantum circuits using hybrid superconducting-semiconducting nanowire Josephson elements". In: *Physical Review Letters* 115.12, art.nr. 127002 (2015). harvest, pp. 1–5. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.115.127002.

[17] Rolf Drechsler et al. *Advanced formal verification*. Vol. 122. Springer, 2004.

[18] Richard Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6 (June 1982), pp. 467–488.

[19] Joseph F Fitzsimons and Michal Hajdušek. "Post hoc verification of quantum computation". In: *arXiv preprint arXiv:1512.04375* (2015).

[20] Joseph F. Fitzsimons and Elham Kashefi. "Unconditionally verifiable blind quantum computation". In: *Phys. Rev. A* 96 (2017).

[21] Jay M. Gambetta, Jerry M. Chow, and Matthias Steffen. "Building logical qubits in a superconducting quantum computing system". In: *npj Quantum Information* 3.1 (2017), p. 2. ISSN: 2056-6387. DOI: 10.1038/s41534-016-0004-0.

[22] Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. "Verification of quantum computation: An overview of existing approaches". In: *arXiv preprint arXiv:1709.06984* (2017).

[23] Alexandru Gheorghiu, Elham Kashefi, and Petros Wallden. "Robustness and device independence of verifiable blind quantum computing". In: *New Journal of Physics* 17.8 (2015), p. 083040.

[24] Raanan Grinwald et al. "User defined coverage—a tool supported methodology for design verification". In: *Proceedings of the 35th annual Design Automation Conference*. ACM. 1998, pp. 158–163.

[25] Michal Hajdušek, Carlos A Pérez-Delgado, and Joseph F Fitzsimons. "Device-independent verifiable blind quantum computation". In: *arXiv preprint arXiv:1502.02563* (2015).

[26] IBM QX team. "ibmqx3 backend specification". GitHub repository, accessed August 2017. 2017. URL: https://ibm.biz/qiskit-ibmqx3.

[27] A. Kandala et al. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". In: *Nature* 549 (2017), p. 242.

[28] Julian Kelly et al. "State preservation by repetitive error detection in a superconducting quantum circuit". In: *Nature* 519.7541 (2015), pp. 66–69.

[29] E. Knill et al. "Randomized benchmarking of quantum gates". In: *Phys. Rev. A* 77 (1 Jan. 2008), p. 012307. DOI: 10.1103/PhysRevA.77.012307. URL: https://link.aps.org/doi/10.1103/PhysRevA.77.012307.

[30] D. A. Lidar and T. A. Brun, eds. *Quantum Error Correction*. Cambride, UK: Cambridge University Press, 2013.

[31] N. M. Linke *et al.* "Fault-tolerant quantum error detection". arXiv:1611.06946. 2016.

[32] Easwar Magesan, J. M. Gambetta, and Joseph Emerson. "Scalable and Robust Randomized Benchmarking of Quantum Processes". In: *Phys. Rev. Lett.* 106 (18 May 2011), p. 180504. DOI: 10.1103/PhysRevLett.106.180504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.106.180504.

[33] Matthew McKague. "Interactive Proofs for BQP via Self-Tested Graph States". In: *Theory of Computing* 12.3 (2016), pp. 1–42. DOI: 10.4086/toc.2016.v012a003. URL: http://www.theoryofcomputing.org/articles/v012a003.

[34] Anand Natarajan and Thomas Vidick. “Robust self-testing of many-qubit states”. In: *arXiv preprint arXiv:1610.03574* (2016).
[35] Yehuda Naveh et al. “Constraint-based random stimuli generation for hardware verification”. In: *AI magazine* 28.3 (2007), p. 13.
[36] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. New York, NY, USA: Cambridge University Press, 2011.
[37] D. Nigg et al. “Quantum computations on a topologically encoded qubit”. In: *Science* 345.6194 (2014), pp. 302–305. DOI: 10.1126/science.1253742.
[38] B. Reichardt, F. Unger, and U. Vazirani. “Classical command of quantum systems”. In: *Nature* 496.7446 (2013), p. 456.
[39] Chad Rigetti et al. “Superconducting qubit in a waveguide cavity with a coherence time approaching 0.1 ms”. In: *Physical Review B* 86.10 (2012), p. 100506.
[40] Maika Takita et al. “Demonstration of Weight-Four Parity Measurements in the Surface Code Architecture”. In: *Phys. Rev. Lett.* 117 (21 Nov. 2016), p. 210505. DOI: 10.1103/PhysRevLett.117.210505.
[41] Maika Takita et al. “Experimental Demonstration of Fault-Tolerant State Preparation with Superconducting Qubits”. Oct. 2017. URL: https://link.aps.org/doi/10.1103/PhysRevLett.119.180501.
[42] L. M. K. Vandersypen et al. “Interfacing spin qubits in quantum dots and donors—hot, dense, and coherent”. In: *International journal of tourism sciences* 3.1 (2017), p. 34. ISSN: 2056-6387. DOI: 10.1038/s41534-017-0038-y. URL: https://publications.rwth-aachen.de/record/707467.
[43] Umesh Vazirani. *Workshop on the Computational Worldview and the Sciences*. http://users.cms.caltech.edu/~schulman/Workshops/CS-Lens-2/report-comp-worldview.pdf. 2007.
[44] C. Vuillot. “Error detection is already helpful on the IBM 5Q chip”. arXiv:1705.08957. 2017.
[45] J. R. Wootton. “Repetition code for up to 15 qubits”. GitHub repository, accessed August 2017. 2017. URL: https://github.com/QISKit/qiskit-tutorial/blob/master/2_quantum_information/repetition_code.ipynb.
[46] J. R. Wootton and Loss D. “A repetition code of 15 qubits”. arXiv:1709.00990. 2017.
[47] James R Wootton. “Demonstrating non-Abelian braiding of surface code defects in a five qubit experiment”. In: *Quantum Science and Technology* 2.1 (2017), p. 015006.
[48] X. Yao et al. “Observation of eight-photon entanglement”. In: *Nature Photon.* 6 (2012).