REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
20-07-2019	Final Report	21-Dec-2018 - 20-Jun-2019

4. TITLE AND SUBTITLE
Final Report: Quantum Utilities for Integrated Characterization

5a. CONTRACT NUMBER

5b. GRANT NUMBER
W911NF-19-P-0007

5c. PROGRAM ELEMENT NUMBER
665502

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES

Quantum Benchmark
111 Concord St

New Haven, CT 06512 -0000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
74541-PH-ST1.1

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Ian Hincks
UU	UU	UU	UU		19b. TELEPHONE NUMBER: 519-998-7938

RPPR Final Report
as of 29-Aug-2019

Agency Code:

Proposal Number: 74541PHST1 **Agreement Number: W911NF-19-P-0007**
INVESTIGATOR(S):

Name: Ian Hincks
Email: ihincks@quantumbenchmark.com
Phone Number: 5199987938
Principal: Y

Organization: **Quantum Benchmark**
Address: 111 Concord St, New Haven, CT 065120000
Country: USA

DUNS Number: 203747191 EIN:
Report Date: 20-Jul-2019 Date Received: 20-Jul-2019
Final Report for Period Beginning 21-Dec-2018 and Ending 20-Jun-2019
Title: Quantum Utilities for Integrated Characterization
Begin Performance Period: 21-Dec-2018 **End Performance Period:** 20-Jun-2019
Report Term: 0-Other

Submitted By: Ian Hincks Email: ihincks@quantumbenchmark.com
Phone: (519) 998-7938

Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: 0 **STEM Participants:** 0

Major Goals: The goals summarized below are as stated in the original proposal. The overall performance of a gate-based quantum information processing device is determined by its ability to execute low-level gates native to its architecture with low error rates. Specifically, error rates must be low in the context of quantum circuits that require many gates to be executed simultaneously. High-fidelity one- and two-qubit operations have been demonstrated in a variety of physical implementations of multi-qubit systems. These operations demonstrate key components of a quantum information processing system for either physical or logical quantum operations. At the same time, existing experimental methods of evaluating one- and two-qubit gate fidelities suffer from two key limitations.

(i) First, they do not capture cross-talk or spatially correlated errors, which are fundamental obstacles to ensuring the reliability of a multi-qubit system. Cross-talk and spatially correlated errors are emerging as a dominant error source in multi-qubit quantum information processing systems and so design cycles need to be centered around eliminating such errors.

(ii) Second, existing quantum characterization, verification, and validation (QCVV) protocols are experimentally implemented in an ad-hoc way due to the lack of subject matter experts in many experimental laboratories and the tendency to under-specify key implementation details in research publications.

Quantum Benchmark Inc. (Quantum Benchmark) is developing fully scalable QCVV protocols, including Cycle Benchmarking and Pauli Noise Reconstruction, as a software package to be interfaced with quantum hardware. These tools overcome the first limitation by directly characterizing the noise acting on an entire quantum information processor. The abstract protocols are completely scalable and platform-agnostic.

However, prior to the Phase I STTR project, the tools were developed had several key limitations that prevented them from meeting the desired requirements. For the software framework to be platform agnostic, easy to use, and foolproof, it needed to meet the following conditions.

1) The set of quantum operations should be user-configurable. There are many different sets of native quantum operations in different experimental architectures.

2) The framework should allow quantum operations to be specified in terms of gates, Hamiltonians, and generators.

3) The memory footprint for multi-qubit circuits should be small to ensure that the software tools are practical.

4) The generation of experimental circuits and analysis of the results should be fast enough to characterize noise on short time scales, enable adaptive QCVV protocols, and allow rapid iterations of design cycles.

5) The framework should allow users to customize the set of QCVV protocols to be implemented on a quantum information processor.

6) The software interface between the QCVV tool and the hardware controller for the quantum computer system need to be specified to allow straightforward implementation of the protocol and communication of associated data to between the tool and the experimental hardware.

Therefore, the major goal of the Phase I project were to develop a software framework to be used in a Phase II research effort, enabling software tools that implement Cycle Benchmarking and Pauli Noise Reconstruction in addition to standard QCVV protocols.

This goal is broken into the following list of research tasks that address the required conditions listed above.

1) Evaluate the versatility, functionality, and ease of use of the initial software tools, called True-Q, developed by Quantum Benchmark.

2) Develop, create, and refine data types for quantum gates, Hamiltonians, quantum states, quantum measurements, and quantum circuits that are dimension- and platform-agnostic.

3) Develop a framework and interface for users to configure the quantum operations and the order in which QCVV protocols are implemented.

4) Create data interface for communicating circuits to experimental control computers (or intermediaries) and receiving experimental results.

5) Create functions to generate families of circuits for tens of qubits to illustrate scalability of the framework.

6) Evaluate the memory footprint for multi-qubit circuits to determine that software tools built upon the software framework can be scalable.

7) Determine throughput of the data interface to evaluate the requirements for tune-up routines to be feasible.

8) Verify that the software framework is versatile enough to describe the native operations in an existing quantum information processing experiment.

Accomplishments: Major Activities

The major activities undertaken during this project are itemized below. Further descriptions of these activities are detailed in subsequent sections.

1) Assessing existing software capabilities by interfacing it with third party hardware ourselves
2) Assessing existing software capabilities by receiving feedback from a subcontractor (Duke University) who interfaced it with their own hardware
3) Brainstorming and prototyping configuration tools and quantum type design
4) Developing new software framework in front-end and back-end of our software
5) Assessing capabilities of new software

Specific Objectives

Accomplishments of specific objectives are listed with respect to the Research Tasks stated in Major Goals. A Gantt chart detailing the timeline of these tasks can be found in the Attachments section.

1) Evaluate the versatility, functionality, and ease of use of the initial software tools, called True-Q, developed by Quantum Benchmark.

We accomplished this task through internal review and consultations with users of the software. We determined the limitations of the existing software and identified how resources should be diverted to various tasks. In particular, we concluded the following

- A standard quantum hardware device specification was needed to address configurability
- A new type framework was required to support a unified circuit generation and analysis structure
- Aspects of the user interface needed to be streamlined and simplified

2) Develop, create, and refine data types for quantum gates, Hamiltonians, quantum states, quantum measurements, and quantum circuits that are dimension- and platform-agnostic.

We developed minimal type specifications for quantum objects relevant to QCVV, and implemented them as a part of our new software framework. Types associated with quantum circuit descriptions can be ordered in the following hierarchy:

CircuitCollection > Circuit > Cycle > Gate.

Here, a Gate is an abstract or native operation to be performed by a quantum device, a Cycle is a round of parallel gates found in a Circuit, and a CircuitCollection is a collection of circuits to be performed on a quantum device.

The types above are user-facing and live on the front end. Following the development of data types for the front-end, data types for the C++ backend were planned implemented. A sparse representation of quantum operators in terms of Weyl operators was chosen, and the following types were chosen:

- ModVec – vectors of integers modulo some dimension
- Weyl – a complex coefficient combined with a pair of ModVecs (for X and Z resp.)
- Gate – A set of Weyls
- GateList – A map from qubit labels to Gates
- Cycle – A special case of GateList that supports factorization, products, and tensoring

These formats were constructed to enable efficient randomized compiling of quantum circuits, which will also serve as the base for all characterization protocols in our library.

3) Develop a framework and interface for users to configure the quantum operations and the order in which QCVV protocols are implemented.

We developed a standard to specify the native operations of a quantum device. Important aspects of these configuration options include (a) the ability to specify qudit dimension, (b) the ability to choose a single-register decomposition mode, e.g. ZXZXZ for qubits, (c) the ability to alias gate operation types under user defined strings, (d) the ability to conveniently specify connectivity graphs of multi-register operations, (e) the ability to specify which registers are blocked from activity when an operation is taking place on other registers, (f) the ability to compactly specify multi-qudit gates naturally through graph connectivities, for example, the Molmer-Sorensen gate on the complete graph. An example of a configuration file is found in the Attachments section.

4) Create data interface for communicating circuits to experimental control computers (or intermediaries) and receiving experimental results.

This task was completed in two ways. In the first, an interface between our software and the IBM Q API was programmed. This interface allows us to create circuits in our format, eg. through any of our protocol generation functions, and send these circuits to IBMs quantum devices, which subsequently return data that is converted into our circuit result format. This interface was made robust to communication problems, and supports the automatic batching of large experiments into smaller runs. Data was successfully collected and analyzed through this interface.

The second was done through our contractor, Duke university. They were provided with a copy of our software with

the objective of interfacing it to their quantum hardware (trapped ions) and providing feedback to Quantum Benchmark related to the usability, versitility, and performance of the software. As major revisions to the framework of our software were underway at the time, the copy they received did not include the new framework detailed in this document. Their feedback was used to inform design decisions necessary to the completion of the objectives of Phase I.

5) Create functions to generate families of circuits for tens of qubits to illustrate scalability of the framework.

Due to the generality of the type framework described in (2), we were able to use Randomized Compiling (RC) as a centralized primitive for all randomized benchmarking and related protocols, including:

- Standard Randomized Benchmarking
- Interleaved Randomized Benchmarking
- Unitarity and Leakage Benchmarking
- Cycle Benchmarking
- Noise Reconstruction
- Quantum Capacity

Notably, these protocols are able to generate circuits in both isolated and simultaneous modes, with mixed twirling groups. This enables sophisticated cross-talk diagnostics.

6) Evaluate the memory footprint for multi-qubit circuits to determine that software tools built upon the software framework can be scalable.

Memory tests were run by generating circuits varying numbers of qubits. In all cases, memory usage was not observed to be a bottleneck. In particular, for all protocols, we were able to generate circuits on 500 qubits using a standard desktop computer without encountering excessive memory usage, thereby successfully concluding this Task. This number was not chosen as an upper bound to what the framework is capable of, but rather, as a ceiling for the number of high-quality qubits in devices expected to be available in the next two years.

7) Determine throughput of the data interface to evaluate the requirements for tune-up routines to be feasible.

Our interface to the IBMQ device determined that, for this provider, the primary bottleneck to feasible tune-up routines is the latency between submitting a set of quantum circuits as a job to a device and receiving the results. The turnaround time can be between minutes and tens of minutes depending on the availability of the device, which is inadequate for meaningful tune-up.

8) Verify that the software framework is versatile enough to describe the native operations in an existing quantum information processing experiment.

Our interface to the IBM-Q system demonstrated that our sofware framework is capable of describing their native operations, and performing QCVV protocols on their devices, and processing the resulting information.

Significant Results including major findings, developments, and conclusions

We were able to create and implement a unified framework for the generation and analysis of all QCVV protocols. The framework was made possible by the realization that all protocols can be largely defined in terms of RC acting on a protocol-dependent base circuit, with additional contextual information.

We were also able to create a standard for specifying the native operations of

Training Opportunities: Nothing to Report

Results Dissemination: Nothing to Report

RPPR Final Report
as of 29-Aug-2019

Honors and Awards: Nothing to Report

Protocol Activity Status:

Technology Transfer: Nothing to Report

PARTICIPANTS:

Participant Type: PD/PI
Participant: Ian Norman Hincks
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Other Professional
Participant: Dar Wilbur Dahlen
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

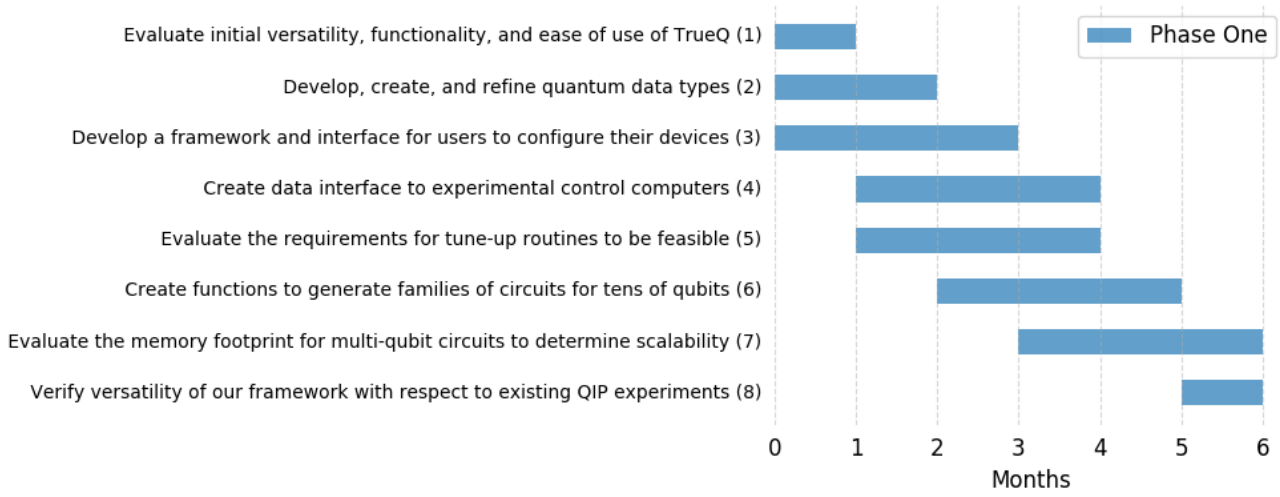
Participant Type: Faculty
Participant: Steven Flammia
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Faculty
Participant: Joseph Emerson
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Faculty
Participant: Joel Wallman
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

RPPR Final Report
as of 29-Aug-2019

Gantt Chart of Research Tasks



Quantum Device Configuration

The following is an example of a configuration file. It lists which gates the device can natively perform (allowing both parameterized and static operations), and specifies which subsets of registers are allowed to perform each operation.

```
Name: TestDevice
Dimension: 2
N_Systems: 4
Mode: ZXZXZ

Gate Z(phi):
  Hamiltonian:
     - ['Z', phi]
  Involving:
     (0,) : (1,)
     (1,) : (0, 2)
     (2,) : (1, 3)
     (3,) : (2,)
Gate X90:
  Hamiltonian:
     - ['X', 90]
  Involving:
     (0,) : (1,)
     (1,) : (0, 2)
Gate CNOT:
  Matrix:
    - [1, 0, 0, 0]
    - [0, 1, 0, 0]
    - [0, 0, 0, 1]
    - [0, 0, 1, 0]
  Involving:
    (0, 1) : (2, 3)
    (1, 2) : (0, 3)
    (2, 3) : (0, 1)
Gate FER(a,b,c,d,e):
  Matrix:
    - [1, 0,                          0,                            0        ]
    - [0, exp(1j*(b)) * cos(a/2),   exp(1j*(b+c)) * sin(a/2),   0        ]
    - [0, exp(1j*(b+d)) * sin(a/2), exp(1j*(b+c+d)) * cos(a/2), 0        ]
    - [0, 0,                          0,                            exp(1j*e)]
  Involving:
    (0, 1) : (2, 3)
    (1, 2) : (0, 3)
    (2, 3) : (0, 1)
```