AFRL-RI-RS-TR-2014-038

HYBRID TECHNIQUES FOR QUANTUM CIRCUIT SIMULATION

UNIVERSITY OF MICHIGAN

FEBRUARY 2014

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2014-038 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /
PAUL M. ALSING
Work Unit Manager

/ S /
MARK H. LINDERMAN
Technical Advisor, Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
FEB 2014	FINAL TECHNICAL REPORT	NOV 2010 – SEP 2013

4. TITLE AND SUBTITLE

HYBRID TECHNIQUES FOR QUANTUM CIRCUIT SIMULATION

5a. CONTRACT NUMBER
FA8750-11-2-0043

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Igor L. Markov, John P. Hayes

5d. PROJECT NUMBER
T2QC

5e. TASK NUMBER
MI

5f. WORK UNIT NUMBER
CH

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Michigan, EECS Department
2260 Hayward St.
Ann Arbor, MI 48109 - 2121

8. PERFORMING ORGANIZATION REPORT NUMBER

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

Air Force Research Laboratory/RITA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2014-038

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88ABW-2014-0595
Date Cleared: 19 FEB 2014

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This is the final report for our project "Hybrid Techniques for Quantum Circuit Simulation". We propose new computing technologies for simulating quantum-mechanical phenomena to aid in the development of quantum computers. We contribute new algorithms and software tools to simulate certain classes of quantum circuits with improved efficiency. Such circuits arise when enriching arbitrary quantum circuits with quantum error-correction codes, and may also be generated by restructuring more generic quantum circuits to improve the efficiency of quantum simulation. The project particularly explores extensions of the so-called stabilizer formalism to allow arbitrary quantum gates. Compared to prior work, the resulting computation is more efficient and more amenable to parallel processing. Empirical results are presented on a variety of quantum circuit benchmarks. We also show how to apply stochastic logic to the simulation of quantum circuits.

15. SUBJECT TERMS

Quantum information processing, quantum simulation, stabilizer circuits, algorithms, software

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**PAUL M. ALSING**
U	U	U	SAR	28	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 SUMMARY

We are exploring computing technologies for simulating quantum-mechanical phenomena and developing software tools to aid in the development of quantum computers. The project seeks new algorithms and software tools to simulate certain classes of quantum circuits with improved efficiency. Such circuits arise when enriching arbitrary quantum circuits with quantum error-correction codes, and may also be generated by restructuring more generic quantum circuits to improve the efficiency of quantum simulation. The project is structured around extensions of the so-called stabilizer formalism, making the resulting computation more efficient and amenable to parallel processing.

2 INTRODUCTION

The work reported includes algorithmic techniques and methodologies to simulate quantum circuits with improved efficiency. Our key objective was to develop and evaluate new principles, algorithms and software for high-performance simulation of quantum circuits – a widely accepted computational model for quantum computation and communication

The construction of computer algorithms and software models that simulate physical systems plays a fundamental role in all branches of science and engineering, and plays a special role in attempts to achieve competitive advantage over non-quantum techniques. In particular, it was observed in the 1980s that the important task of simulating quantum-mechanical processes on a standard computer requires an extraordinary amount of computer memory and runtime. Such observations gave rise to the notion of quantum computing, where quantum mechanics itself is used to simulate naturally-occurring quantum properties and phenomena. The key insight is to replace the familiar 0 and 1 bits of conventional computing with information units called qubits (quantum bits) that capture quantum states of elementary particles or atomic nuclei. By operating on qubits, a quantum computer can, in principle, process exponentially more data than a classical computer in a similar number of steps. Existing quantum algorithms exponentially outperform best known techniques for key tasks in code-breaking [1] [2]. In addition to quantum computation, quantum communication exhibits unique properties, such as automatic detection of attempts to eavesdrop. High-speed quantum communication systems have been built by the National Institute of Standards and Technology, several DARPA contractors (notably, BBN Technologies) and start-ups in the US and Europe. Some of these systems are currently available commercially, and others are operated 24x7 for research purposes and as testbeds for network protocol development.

The most serious fundamental obstacle to practical quantum information processing known today is the inherent instability of qubits. This obstacle is traditionally addressed by quantum error-correction techniques [3] [4], which are generally available but require significant overhead and require adaptation to individual quantum circuits. In order to scale quantum information processing, including computation and communication, to large and complex systems, quantum

device operation is compactly captured by the abstract formalism of quantum circuits [5]. These circuits consist of interconnected quantum gates that act on qubits. They can be composed in a hierarchical manner, using design techniques similar to those used in digital logic design. However, as quantum circuits offer a much broader range of information-processing possibilities, their design and evaluation entail a dramatic increase of complexity, requiring new levels of sophistication in design algorithms and tools [6]. A particularly important class of design tools performs simulation of quantum circuits on conventional workstations, i.e., these tools produce representative outputs of ideal quantum circuits on particular inputs, but without requiring quantum hardware.

Quantum simulation tools typically consist of a front-end and a back-end [7]. The front-end facilitates the development of quantum software and the back-end acts as a temporary replacement of (hardware) quantum processing units to run such software. Once quantum hardware is available, it can be used in conjunction with pre-existing front-end to run the accumulated software with increased efficiency. In some cases, quantum simulation can demonstrate that quantum software does not bring competitive advantage to quantum hardware over conventional CPUs. This project develops new mathematical and algorithmic concepts for efficient simulation of quantum circuits. These concepts are being implemented in software and will help evaluating architectures for error-corrected quantum communication and computation.

We have also begun to investigate a highly unconventional tool for simulating quantum circuits that employs stochastic computing (SC) [8] [9]. SC processes information in the form of random bit-streams that are interpreted as probabilities and are implemented by relatively simple logical structures. Preliminary results demonstrate that the technique is feasible, and that useful trade-offs between model size, run-time and accuracy can be achieved.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

This effort builds upon the software infrastructure we developed as part of previous AFRL-funded research. Unlike, other quantum-circuit simulators, stabilizer-based techniques can process the so-called stabilizer gates (Figure 1) very efficiently.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad Hadamard = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad Phase = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

Figure 1. Controlled-CNOT, Hadamard and Phase gates.

The stabilizer formalism is a simulation method involving the *Heisenberg model* [8] often used by physicists to describe atomic-scale phenomena. *In this model, one keeps track of the symmetries of an object rather than represent the object explicitly*. In the context of quantum-circuit simulation, this model represents quantum states by their symmetries, rather than

complex-valued vectors and amplitudes. The symmetries are *Pauli operators* (tensor products of the one-qubit Pauli matrices shown in Figure 2 along with the identity) for which these states are 1-eigenvectors, i.e., the Pauli operators that *stabilize* such states. Algebraically, these symmetries form a *group* structure, which can be specified compactly by *group generators* [9]. Therefore, the formalism represents an n-qubit pure quantum state $|\psi\rangle$ by its *stabilizer group* $S(|\psi\rangle)$ – the group of n-qubit Pauli operators that stabilizes$|\psi\rangle$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Figure 2. The one-qubit Pauli operators.

Theorem 1. *For an n-qubit pure state* $|\psi\rangle$ *and* $k \leq n$, $S(|\psi\rangle) \cong \mathbb{Z}_2^k$. *If* $k = n$, $|\psi\rangle$ *is specified uniquely by* $S(|\psi\rangle)$ *and is called a stabilizer state.*
Proof. See [10] for details.

$S(|\psi\rangle)$ is specified by only $\log_2 2^n = n$ irredundant stabilizer generators. Therefore, an arbitrary n-qubit stabilizer state can be represented by a *stabilizer matrix* $\boldsymbol{\mathcal{M}}$ whose rows represent a set of generators $R_1, R_2, \ldots, R_n$ for $S(|\psi\rangle)$. (Hence we use the terms generator set and stabilizer matrix interchangeably.) Since each R_i is a string of n Pauli literals (X, Y, Z or I), the size of the matrix is $n \times n$. Each R_i contains a ± 1 leading phase which is stored separately using a binary vector of size n. Thus, the storage cost for $\boldsymbol{\mathcal{M}}$ is $\Theta(n^2)$, which is an exponential improvement over the $O(2^n)$ cost often encountered in vector-based representations. Tables 1 and 2 list all one and two-qubit stabilizer states along with their generator sets.

Table 1. One-qubit stabilizer states and their Pauli-matrix stabilizer. Shorthand notation lists only the normalized amplitudes of the basis states.

Stabilizer state	Shorthand notation	Pauli generator	Stabilizer state	Shorthand notation	Pauli generator
$(\|0\rangle + \|1\rangle)/\sqrt{2}$	$1, 1$	X	$(\|0\rangle - \|1\rangle)/\sqrt{2}$	$1, -1$	-X
$(\|0\rangle + i\,\|1\rangle)/\sqrt{2}$	$1, i$	Y	$(\|0\rangle - i\,\|1\rangle)/\sqrt{2}$	$1, -i$	-Y
$\|\mathbf{0}\rangle$	$\mathbf{1, 0}$	**Z**	$\|\mathbf{1}\rangle$	$\mathbf{0, 1}$	**-Z**

Table 2. Sixty two-qubit stabilizer states and their corresponding Pauli generators.

	State	Gen'tors	∠	State	Gen'tors	∠	State	Gen'tors	∠	State	Gen'tors	∠
Separable	$1,1,1,1$	IX, XI	$\pi/3$	$1,-1,1,-1$	-IX, XI	$\pi/3$	$1,1,-1,-1$	IX, -XI	$\pi/3$	$1,-1,-1,1$	-IX, -XI	$\pi/3$
	$1,1,i,i$	IX, YI	$\pi/3$	$1,-1,i,-i$	-IX, YI	$\pi/3$	$1,1,-i,-i$	IX, -YI	$\pi/3$	$1,-1,-i,i$	-IX, -YI	$\pi/3$
	$1,1,0,0$	IX, ZI	$\pi/4$	$1,-1,0,0$	-IX, ZI	$\pi/4$	$0,0,1,1$	IX, -ZI	$\perp$	$0,0,1,-1$	-IX, -ZI	$\perp$
	$1,i,1,i$	IY, XI	$\pi/3$	$1,-i,1,-i$	-IY, XI	$\pi/3$	$1,i,-1,-i$	IY, -XI	$\pi/3$	$1,-i,-1,i$	-IY, -XI	$\pi/3$
	$1,i,i,-1$	IY, YI	$\pi/3$	$1,-i,i,1$	-IY, YI	$\pi/3$	$1,i,-i,1$	IY, -YI	$\pi/3$	$1,-i,-i,-1$	-IY, -YI	$\pi/3$
	$1,i,0,0$	IY, ZI	$\pi/4$	$1,-i,0,0$	-IY, ZI	$\pi/4$	$0,0,1,i$	IY, -ZI	$\perp$	$0,0,1,-i$	-IY, -ZI	$\perp$
	$1,0,1,0$	IZ, XI	$\pi/4$	$0,1,0,1$	-IZ, XI	$\perp$	$1,0,-1,0$	IZ, -XI	$\pi/4$	$0,1,0,-1$	-IZ, -XI	$\perp$
	$1,0,i,0$	IZ, YI	$\pi/4$	$0,1,0,i$	-IZ, YI	$\perp$	$1,0,-i,0$	IZ, -YI	$\pi/4$	$0,1,0,-i$	-IZ, -YI	$\perp$
	$\mathbf{1,0,0,0}$	**IZ, ZI**	0	$\mathbf{0,1,0,0}$	**-IZ, ZI**	$\perp$	$\mathbf{0,0,1,0}$	**IZ, -ZI**	$\perp$	$\mathbf{0,0,0,1}$	**-IZ, -ZI**	$\perp$
Entangled	$0,1,1,0$	XX, YY	$\perp$	$1,0,0,-1$	-XX, YY	$\pi/4$	$1,0,0,1$	XX, -YY	$\pi/4$	$0,1,-1,0$	-XX, -YY	$\perp$
	$1,0,0,i$	XY, YX	$\pi/4$	$0,1,i,0$	-XY, YX	$\perp$	$0,1,-i,0$	XY, -YX	$\perp$	$1,0,0,-i$	-XY, -YX	$\pi/4$
	$1,1,1,-1$	XZ, ZX	$\pi/3$	$1,1,-1,1$	-XZ, ZX	$\pi/3$	$1,-1,1,1$	XZ, -ZX	$\pi/3$	$1,-1,-1,-1$	-XZ, -ZX	$\pi/3$
	$1,i,1,-i$	XZ, ZY	$\pi/3$	$1,i,-1,i$	-XZ, ZY	$\pi/3$	$1,-i,1,i$	XZ, -ZY	$\pi/3$	$1,-i,-1,-i$	-XZ, -ZY	$\pi/3$
	$1,1,i,-i$	YZ, ZX	$\pi/3$	$1,1,-i,i$	-YZ, ZX	$\pi/3$	$1,-1,i,i$	YZ, -ZX	$\pi/3$	$1,-1,-i,-i$	-YZ, -ZX	$\pi/3$
	$1,i,i,1$	YZ, ZY	$\pi/3$	$1,i,-i,-1$	-YZ, ZY	$\pi/3$	$1,-i,i,-1$	YZ, -ZY	$\pi/3$	$1,-i,-i,1$	-YZ, -ZY	$\pi/3$

Note that in Table 2, the shorthand notation represents a stabilizer state as $\alpha_1, \alpha_2, \alpha_3\alpha_4$where α_i are the normalized amplitudes of the basis states. The basis states are emphasized in bold. The first column lists states whose generators do not include an upfront minus sign, and other columns introduce the signs. A sign change creates an orthogonal vector. Therefore, each row of the table gives an orthogonal basis. The cells in dark grey indicate stabilizer states with four non-zero basis amplitudes, i.e., $\alpha_i \neq 0 \; \forall \; i$. The $\angle$ column indicates the angle between that state and $|00\rangle$, which has twelve nearest-neighbor states (light gray) and 15 orthogonal states ($\perp$).

Stabilizer circuits, which consist exclusively of stabilizer and measurement gates, can be simulated in polynomial time and space on conventional computers because such gates conjugate the Pauli group onto itself. Thus, to simulate a stabilizer gate, one simply updates the stabilizer matrix using a finite set of rules for mapping Pauli literals to Pauli literals [3] [11] [12]. Stabilizer simulation is limited by the following constrains: (*i*) the initial state of the quantum system must be a computational-basis or stabilizer state and (*ii*) the quantum gates that are applied to the system must be stabilizer gates.

Our approach was to generalize stabilizer simulation by relaxing the above constraints. This included the design of a novel data structure, called a *stabilizer frame*, which can be used to represent an arbitrary quantum state as a superposition of stabilizer states. It is important to note that given two physical quantum states, there is no quantum-mechanical operation to combine them in a linear combination – such operations are specific to the simulation context. Generic simulation algorithms typically can process arbitrary linear combinations, but algorithms that are limited to certain quantum states usually have trouble with linear combinations.

Stabilizer frames facilitate simulation of almost-stabilizer circuits, i.e., circuits that contain a small number of non-stabilizer gates. To understand the advantages and limitations of our approach, we conducted a comprehensive analysis of the geometry of stabilizer states ("On the geometry of stabilizer states," QIC 2014) [10]. This research allowed us to identify the following advantages when using stabilizer-state superpositions to simulate quantum circuits are:

- Large stabilizer subcircuits, such as those encountered in quantum error correcting circuits and fault-tolerant architectures, are simulated with high efficiency.
- Superpositions can be restructured and compressed on the fly during simulation to reduce resource requirements.
- Operations performed on such superpositions can be computed in parallel and lend themselves to distributed or asynchronous processing. The superpositions can be partitioned between processors, and gates can be applied in parallel to these partitions.

Therefore, the stabilizer-based techniques, algorithms and tools we developed can handle arbitrary gates beyond the stabilizer formalism, but incur certain penalties for non-stabilizer gates. Furthermore, we designed additional techniques to moderate these penalties through algorithmic innovation, optimized software implementations and parallel processing ("Quipu: high-performance simulation of quantum circuits via stabilizer frames," ICCD 2013).

4 RESULTS AND DISCUSSION

Three main accomplishments of this project deserve special identification: (*i*) our software improvements to QuIDDPro (developed under prior support from DARPA and AFRL), (*ii*) our results on the geometry of stabilizer sates, and (*iii*) our design of new software tools for simulation of quantum circuits and fault-tolerant quantum architectures.

4.1 Development and maintenance of software tools

The goal pursued in this part of the overall effort was to design a compact data structure for representing and transforming quantum circuits within QuIDDPro in order to improve overall runtime and memory performance when simulating quantum circuits. At the gate-level, our data structure is designed to use various storage schemes that range from highly compact bit-representations to explicit matrices depending on the quantum gate that is to be stored in memory. It also provides I/O functions and can be used as an interface mechanism between external synthesis and simulation tools.

Data decomposition. Our data structures specify the information required to represent and analyze quantum circuits. This data structure also provides access to low-level constructs that represent quantum gates in the most compact way possible given the specifications of each gate. Quantum circuits describe a particular computation using a sequence of quantum gates that act on a set of qubits. Hence, we employ a data structure consisting of an array of quantuum gates. Each gate is represented using one of three data structures—compact, elaborate or explicit. Note that these data structures are not disjoint and some build upon simpler data structures, ensuring efficiency in common cases, while also representing less frequently occurring gates.

- *Case 1*: the gate can be specified using a 64-bitfield representation that contains the necessary qubit wire information only, i.e., the operator matrix is not stored. We call this the compact_quantum_gate data structure. Most named quantum gates (e.g. NOT, CNOT, PHASE, etc.) that act on up to three-qubits will fall into this category.

- *Case 2*: the gate representation requires more than 64-bits. We use a larger elaborate_quantum_gate data structure that stores all qubit wire information for named gates of any size by reusing the 64-bit scheme described in Case 1. The matrix operator itself is not stored. Most large named gates that act on more than three qubits will fall under this category.
- *Case 3*: the gate is represented explicitly using a matrix of complex numbers and the qubit wire information is stored using the schemes from either Case 1 or Case 2 depending on the amount of wire information that needs to be stored. We call this the explicit_quantum_gate data structure. Unnamed and user-defined gates fall under this category.

We implemented this data structure as the interface mechanism within QuIDDPro to facilitate synthesizing and simulating quantum circuits. Specifically, we used this data structure to implement a native quantum compiler (q-compiler) in QuIDDPro. Analogous to the initial phase of classical compilers, the q-compiler creates the intermediate representation that is to be transformed in later phases of the design flow. The q-compiler first parses the quantum circuit specification input file and decides which compact representation to use on a per-gate basis. Our data structure is then used to instantiate the intermediate representation and to keep track of the subsequent circuit transformations. Since our proposed data structure is compact by design, it is possible to load the entire quantum circuit representation in memory at once thereby removing several potential sources of inefficiency. The current version of QuIDDPro (3.8) features a compile batch-mode option, which acts as a q-compiler and outputs one intermediate-representation file for each state-vector manipulated in the high-level quantum program (density matrices are not currently supported). Figure 3 shows the intermediate representation for the quantum Fourier transform circuit that is instantiated from the successful compilation of the high-level source file. The circuit representation can be used by external design tools as required, e.g. for building a graphical representation for the circuit.

Simulation of quantum circuits. After generating quantum circuits from a high-level QuIDDPro program, it may be useful to simulate such a circuit. Given the multitude of available algorithms for simulation, one should first analyze the circuit's structure to determine the most appropriate simulation techniques. For example, in the case of quantum stabilizer circuits, the software infrastructure can employ efficient algorithms from [11]. In fact, our data structure is already used by two different simulator back-ends---one specific to stabilizer circuits and one entirely generic, which we call QuIDDPro-Lite (QPLite). Note that our work takes on a more holistic approach to quantum circuit simulation by being sensitive to the type of circuits in each case. In the worst case, the exponential cost associated with simulation of arbitrary quantum gates will limit the range of problems that the infrastructure can support. However, the infrastructure is flexible enough such that it can be used to simulate most practical quantum circuits and even allows simulation of hybrid computing platforms that incorporate both quantum and classical constructs. Our experiments showed that pre-compiling and simulating a quantum circuit via QPLite is up to 4X faster than using the native QuIDDPro simulator.

4.2 On the geometry of stabilizer states

The goal pursued in this part of the overall effort was to provide a theoretical analysis of the geometric properties of stabilizer states, and design new efficient algorithms for computing fundamental geometric operations such as inner products and wedge products. The main contributions of this work are:

- We quantified the distribution of angles between pairs of stabilizer states and characterized nearest-neighbor stabilizer states.
- We studied linearly-dependent sets of stabilizer states and showed that every linearly-dependent triplet of such states that are non-parallel to each other includes two pairs of nearest neighbors and one pair of orthogonal states. We illustrate such triplets in Figure 4.

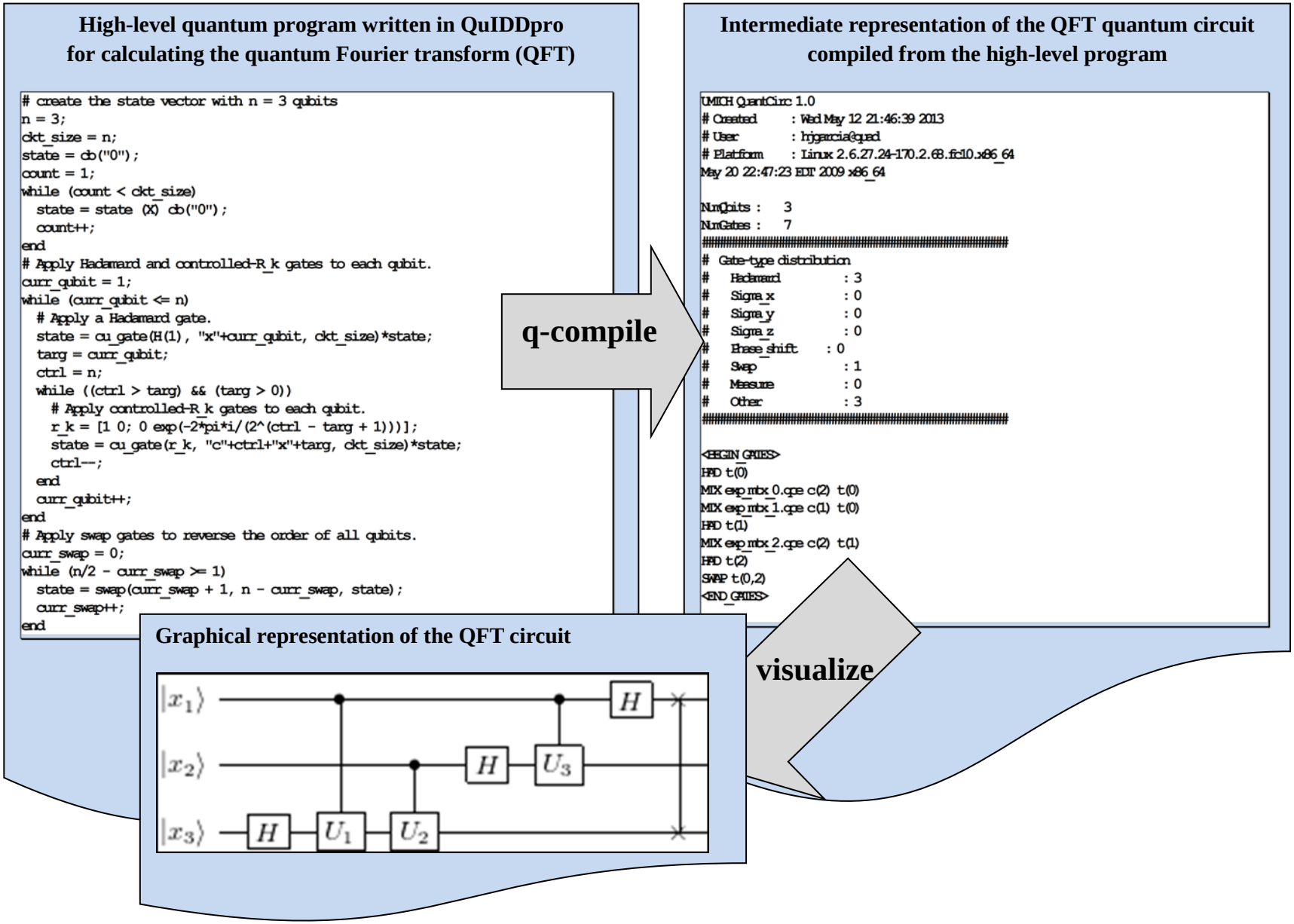


Figure 3. Sample output of the intermediate circuit representation compiled from high-level program source code.

- We also described an orthogonalization procedure for stabilizer states that exploits this rather uniform nearest-neighbor structure.
- We showed that: (*i*) for any n-qubit stabilizer state $|\psi\rangle$, there are at least $5(2n-1)$ states $|\varphi\rangle$ such that $|\psi \wedge \varphi\rangle$ is a stabilizer bivector, and (*ii*) the norm of the wedge product between any two stabilizer states is $\sqrt{1-2^{-k}}$, where $0 \leq k \leq n$.
- We explored the approximation of non-stabilizer states by single stabilizer states and short superpositions of stabilizer states.
- We exploited our analysis of the geometry of stabilizer states to design algorithms for the following fundamental tasks: (*i*) synthesizing new canonical stabilizer circuits that reduce the number of gates required for QECC encoding procedures, (*ii*) computing the inner product between stabilizer states, and (*iii*) computing stabilizer bivectors.

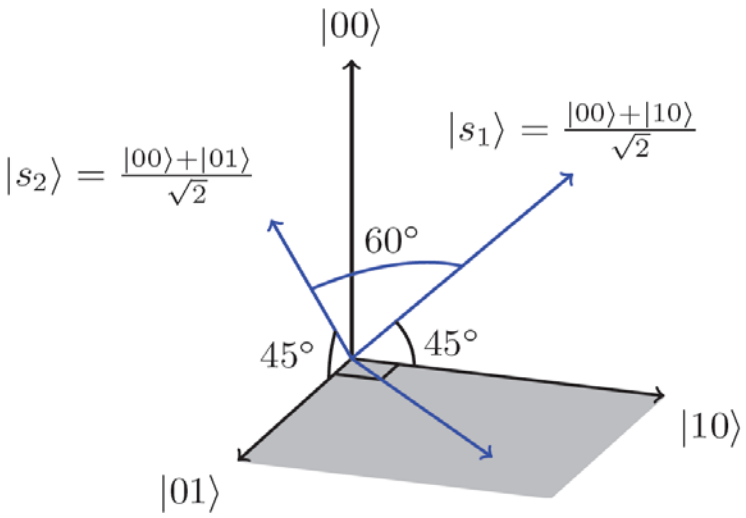


Figure 4. The angle between any stabilizer state and its nearest neighbors is 45^o and the distance is $\sqrt{2-\sqrt{2}}$.

In Figure 4, the angle between any stabilizer state and its nearest neighbors is 45^o and the distance is $\sqrt{2-\sqrt{2}}$. Here, $|s_1\rangle$ is a nearest neighbor of both $|00\rangle$ and $|10\rangle$. Similarly, $|s_2\rangle$ is a nearest neighbor of $|00\rangle$ and $|01\rangle$. The angle between these two nearest neighbors of $|00\rangle$ is 60^o. Consider the linearly-dependent triplets $\{|00\rangle, |10\rangle, |s_1\rangle\}$ and $\{|00\rangle, |01\rangle, |s_2\rangle\}$. Each set contains two pairs of nearest neighbors and one pair of orthogonal states.

Geometric properties of stabilizer states. We define *nearest neighbor* of an n-qubit stabilizer state $|\psi\rangle$ as a state $|\phi\rangle$ such that $|\langle\psi|\phi\rangle| = 1/\sqrt{2}$, the largest possible value $\neq 1$. We prove that every stabilizer state has exactly $4(2^n-1)$ nearest neighbors and generalize this result to the case of k-neighbors, where $0 < k \leq n$. We used these results to quantify the distribution of angles between any one stabilizer state and all other stabilizer states as shown in Table 3.

Table 3. Distribution of inner products (angles) between any one n-qubit stabilizer state and all other stabilizer states for $n \in \{1, 2, \ldots, 7\}$.

n	$\mathcal{N}(n)$	$\mathcal{L}_n(k)/(\mathcal{N}(n)-1)$							
		$k=1$ (45.00°)	$k=2$ (60.00°)	$k=3$ (69.30°)	$k=4$ (75.52°)	$k=5$ (79.82°)	$k=6$ (82.82°)	$k=7$ (84.93°)	$\perp$ (90.00°)
1	6	80%							20%
2	60	20.34%	54.24%						25.42%
3	1080	2.59%	20.76%	47.45%					29.19%
4	36720	0.16%	3.05%	20.92%	44.62%				31.25%
5	2423520	0.01%	0.20%	3.27%	20.96%	43.27%			32.29%
6	315057600	$\approx 0\%$	0.01%	0.23%	3.39%	20.97%	42.60%		32.81%
7	81284860800	$\approx 0\%$	$\approx 0\%$	0.01%	0.24%	3.44%	20.97%	42.27%	33.07%

In Table 3 $\mathcal{N}(n)$ is the total number of n-qubit stabilizer states. The last column indicates the ratio of orthogonal ($\perp$) states. Table 3 shows that, for sufficiently large n, $1/3$ of all stabilizer states are orthogonal to $|\psi\rangle$ and the fraction of k-neighbors tends to zero for $0 < k < n - 4$. Thus, $2/3$ of all stabilizer states are oblique to $|\psi\rangle$ and this fraction is dominated by the k-neighbors, where $n - 4 \leq k \leq n$. These findings suggest a rather uniform geometric structure for stabilizer states. Detailed theorems and proofs describing these results are included in our published manuscript [10].

Embedding of stabilizer geometry in the Hilbert space. We also describe how the discrete embedding of stabilizer geometry in Hilbert space complicates several natural geometric tasks. As described above, our results on the geometric properties of stabilizer states imply that there are significantly more stabilizer states than the dimension of the Hilbert space in which they are embedded, and that they are arranged in a fairly uniform pattern. These factors suggest that, if one seeks a stabilizer state closest to a given arbitrary quantum state, local search appears a promising candidate. To the contrary, in Section 4.2 of [10] show that local search does not guarantee finding such stabilizer states. The second natural task we consider is representing or approximating a given arbitrary quantum state by a short linear combination of stabilizer states. Again, having considerably more stabilizer states than the dimension of the Hilbert space seems to suggest a positive result. However, in Section 4.3 of [10] we demonstrate a family of quantum states that exhibits asymptotic orthogonality to all stabilizer states and can thus be neither represented nor approximated by short superpositions of stabilizer states. Furthermore, as the following proposition shows, the maximal radius of any 2^n-dimensional ball centered at a point on the unit sphere that does not contain any n-qubit stabilizer states cannot exceed $\sqrt{2}$, but approaches $\sqrt{2}$ as $n \to \infty$.

Proposition 2. *Consider 2^n-dimensional balls centered at a point on the unit sphere that do not contain any n-qubit stabilizer states in their interior. The radius of such balls cannot exceed $\sqrt{2}$, but approaches $\sqrt{2}$ as $n \to \infty$.*

Proof. Consider an arbitrary ball B with radius $\sqrt{2}$ and centered on the unit sphere as shown in Figure 5. B covers half of the unit sphere. Let $|s\rangle$ be a stabilizer state that does not lie on the boundary of B. Then, either $|s\rangle$ or $-|s\rangle$ is inside B. Furthermore, observe that the intersection of the unit sphere and the boundary of B is contained in a hyperplane of dimension $n - 1$. If all

stabilizer states were contained there, they would not have included a single basis set. However, since stabilizer states contain the computational basis, they cannot all be contained in that intersection. An asymptotic lower bound for the radius of B is obtained using the family of stabilizer-evading states of the form $\left(\frac{|00\rangle+|01\rangle+|10\rangle}{\sqrt{3}}\right)^{\otimes n}$ (Theorem 4.2 in [10]). These states are asymptotically orthogonal to all n-qubit stabilizer states ($n \to \infty$). Therefore, the distance to their closest stabilizer state approaches $\sqrt{2}$.

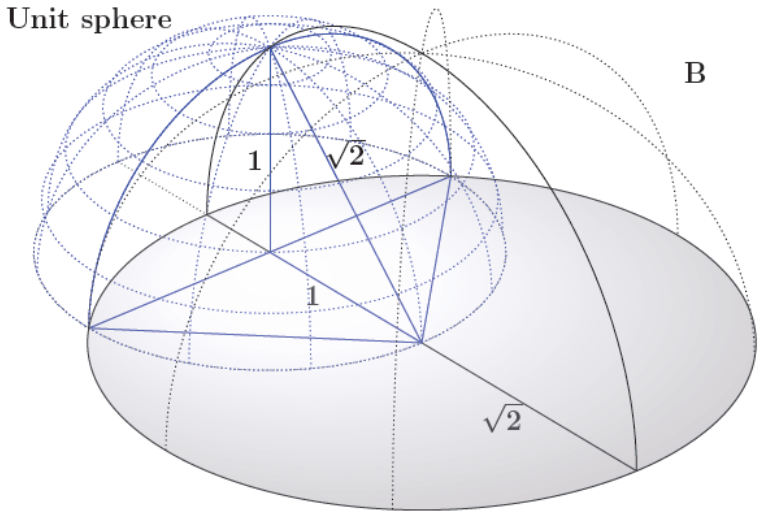


Figure 5. A ball B with radius $\sqrt{2}$ centered on the unit sphere covers half of the unit sphere. Every such ball contains at least one stabilizer state (in most cases, half of all stabilizer states).

Computational geometry of stabilizer states. Angles between stabilizer states were discussed in [11], where the authors describe possible values for such angles and outline an inner-product computation that involves the synthesis of a basis-normalization stabilizer circuit that maps a stabilizer state to a computational basis state. We observed that this circuit-synthesis procedure is the computational bottleneck of the algorithm and thus any improvements to this synthesis process translate into increased performance of the overall algorithm. It was also shown in [11] that, for any unitary stabilizer circuit, there exists an equivalent block-structured canonical circuit that applies a block of Hadamard (H) gates only, followed by a block of CNOT (C) only, then a block of Phase (P) gates only, and so on in the 7-block sequence H-C-P-C-P-C-H. The number of gates in any H and P block is at most $2n$ and $4n$, respectively, so the total number of gates for such canonical circuits is dominated by the C blocks, which contain $O(n^2 log\, n)$ gates. Therefore, reducing the number of C blocks (or those involving other controlled operations) leads to more compact circuits that minimize the build-up of decoherence when implementing QECC encoding procedures. Using an alternative representation for stabilizer states, the work in [13] proves the existence of canonical circuits with the shorter sequence H-C-X-P-CZ, where the X and CZ blocks consist of NOT and Controlled-Z gates, respectively. Such circuits contain only two controlled-op blocks (as compared to three C blocks in the 7-block sequence from [11]) but no algorithms are available in literature to synthesize these circuits for an arbitrary stabilizer state. We described an algorithm for synthesizing canonical circuits with the block sequence H-

C-CZ-P-H. Our circuits are therefore close to the smallest known circuits proposed in [13]. Our canonical circuits improve the computation of inner products, helping us outperform the inner-product algorithm based on non-canonical circuits proposed in [14] as show in Figure 6. Furthermore, we leveraged our circuit-synthesis technique and inner-product algorithm to: (*i*) compute stabilizer bivectors efficiently and (*ii*) derive orthogonalization procedure for linear combinations stabilizer states. These algorithms rely on the stabilizer-matrix representation for quantum states and are described in detail in [10].

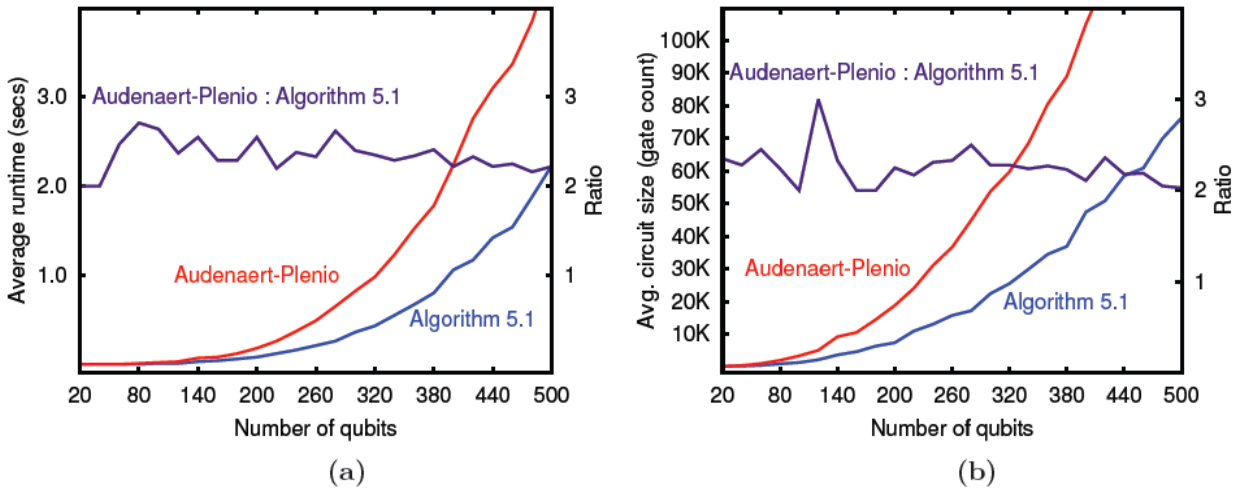


Figure 6. (a) Runtime and (b) circuit-size comparisons between our circuit-synthesis algorirthm (Algorithm 5.1) and the circuit synthesis portion of the Audenaert-Plenio inner-product algorithm.

Figure 6 shows that on average, our algoirthm runs roughly twice as fast and produces canonical circuits that contain less than half as many gates. Furthermore, the Audenaert-Plenio circuits are not canonical.

4.3 Engineering stabilizer-based simulation of generic quantum circuits

The goals pursued in this part of the overall effort were to leverage our results on the geometry of stabilizer states (Section 4.1) in order to design novel data structures and algorithms for simulation of generic quantum circuits via the stabilizer formalism. The stabilizer gates by themselves do not form a universal set for quantum computation [11] [12]. However, the Hadamard and Toffoli (TOF) gates do [15]. To simulate TOF and other non-stabilizer gates, we extend the formalism to include the representation of arbitrary quantum states as superpositions of stabilizer states.

Observation 3. Given an n-qubit stabilizer state $|\psi\rangle$, there exists an orthonormal basis including $|\psi\rangle$ and consisting entirely of stabilizer states. One such basis is obtained directly from the stabilizer representation of $|\psi\rangle$ by changing the signs of an arbitrary, non-empty subset of generators of $S(|\psi\rangle)$, i.e., by modifying the phase vector of the stabilizer matrix for $|\psi\rangle$. Thus, one can produce $2^n - 1$ additional orthogonal stabilizer states. Such states, together with $|\psi\rangle$, form an orthonormal basis. This basis is unambiguously specified by a single stabilizer state, and any one basis state specifies the same basis.

Definition 4. An n-qubit *stabilizer frame* $\mathcal{F}$ is a set of $k \leq 2^n$ stabilizer states $\{|\psi_j\rangle\}_{j=1}^{k}$ that forms an orthogonal subspace basis in the Hilbert space. We represent $\mathcal{F}$ by a pair consisting of (*i*) a stabilizer matrix $\mathcal{M}$ and (*ii*) a set of distinct *phase vectors* $\{\sigma_j\}_{j=1}^{k}$, where $\sigma_j \in \{\pm 1\}^n$. We use $\mathcal{M}^{\sigma_j}$ to denote the ordered assignment of the elements in σ_j as the (± 1)-phases of the rows in $\mathcal{M}$. Therefore, state $|\psi_j\rangle$ is represented by $\mathcal{M}^{\sigma_j}$. The size of the frame, which we denote by $|\mathcal{F}|$, is equal to k.

Each phase vector σ_j can be viewed as a binary (0-1) encoding of the integer index that denotes the respective basis vector. Thus, when dealing with 64 qubits or less, a phase vector can be compactly represented by a 64-bit integer (modern CPUs also support 128-bit integers). To represent an arbitrary state $|\Psi\rangle$ using $\mathcal{F}$, one additionally maintains a vector of complex amplitudes $a = (a_1, \ldots, a_k)$, which corresponds to the decomposition of $|\Psi\rangle$ in the basis $\{|\psi_j\rangle\}_{j=1}^{k}$ defined by $\mathcal{F}$, i.e., $|\Psi\rangle = \sum_{j=1}^{k} a_j |\psi_j\rangle$ and $\sum_{j=1}^{k} |a_j|^2 = 1$ (see Figure 7). Observe that each a_j forms a pair with phase vector σ_j in $\mathcal{F}$ since $|\psi_j\rangle \equiv \mathcal{M}^{\sigma_j}$. Any stabilizer state can be viewed as a one-element frame.

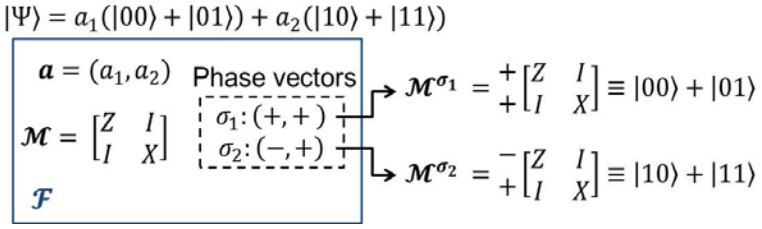


Figure 7. Example of a stabilizer frame $\mathcal{F}$ that represents $|\Psi\rangle$. While $|\Psi\rangle$ is composed of four computational-basis amplitudes, its frame representation has only two phase vectors.

Stabilizer frames are the fundamental data structure in our technique. Prior work on simulation of non-stabilizer gates using the stabilizer formalism can be found in [11] where the authors represent a quantum state as a sum of $O(4^{2dk})$ density-matrix terms while simulating k non-stabilizer operations acting on d distinct qubits. In contrast, the number of states in our technique is $O(2^k)$ although we do not handle density matrices and perform more sophisticated bookkeeping including an $O(n^2)$ algorithm to compute the *global phase of a stabilizer state*. Since the stabilizer formalism simulates gates via their action-by-conjugation, global phases are not maintained. This is not an issue when dealing with a single stabilizer state since a global phase does not affect the statistics of measurement. However, such phases must be maintained when dealing with stabilizer-state superpositions, where global phases become relative.

We designed the following frame operations, which are efficient in the size of the stabilizer frame.

- *Frame rotation* – facilitates simulation of stabilizer gates. The procedure updates the stabilizer matrix and phase vectors associated with $\mathcal{F}$ according to the mapping rules specified by the stabilizer formalism. Each element in the vector of amplitudes

$(a_1, \ldots, a_k)$ is also updated using our global-phase computation algorithm. The runtime of this operation is $O(n^2 + n|\mathcal{F}|)$.

- *Frame cofactoring* – facilitates simulation of measurements and non-stabilizer operations such as Toffoli and controlled-rotation gates. This procedure updates the stabilizer matrix associated with $\mathcal{F}$. Then iterates over each phase vector in $\mathcal{F}$ and permutes the corresponding phases in order to generate additional phase vectors corresponding to *cofactor states*, which are output states obtained after performing computational-basis measurements. As in the case of stabilizer gates, this operation is linear in the number of phase vectors. However, by the end of the operation, the number of phase vectors (states) in $\mathcal{F}$ will have grown by a (worst case) factor of four in the case of both Toffoli and controlled-rotation gates. For an arbitrary n-qubit stabilizer frame $\mathcal{F}$, the number of phase vectors is upper bounded by 2^n, the number of possible ± 1 permutations.

Our simulation technique, implemented in our software package Quipu, admits a universal gate set for quantum computation [15].

Multiframe simulation. Although a single frame is sufficient to represent a stabilizer-state superposition, one can sometimes tame the exponential growth of states in $|\psi\rangle$ by constructing a *multiframe representation*. Such a representation cuts down the total number of states required to represent $|\psi\rangle$ by at least a half, thus improving the scalability of our technique. Our experiments showed that, when simulating ripple-carry adders, the number of states in $|\psi\rangle$ grows linearly when multiframes are used but exponentially when a single frame is used. To this end, we introduce an additional frame operation called *coalescing* whose output is a list of n-qubit frames $\{\mathcal{F}'_1, \mathcal{F}'_2, \ldots, \mathcal{F}'_s\}$ (i.e., a multiframe) that together represent the same superposition as the original input frame $\mathcal{F}$. Figure 8 shows an example.

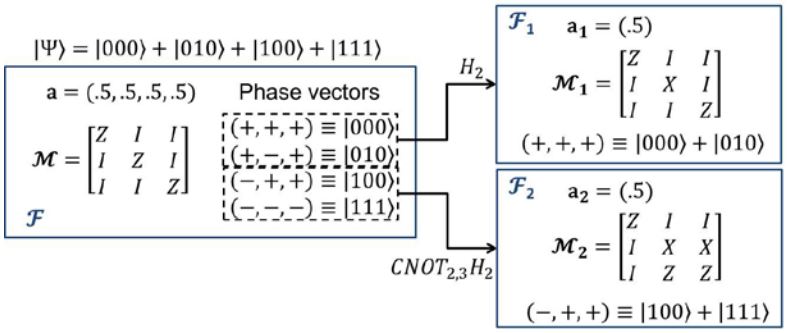


Figure 8. Example of how a multiframe representation is derived from a single-frame representation.

To obtain $\mathcal{F}_1$ in Figure 8, we conjugate the second column of $\mathcal{M}$ by a Hadamard gate and coalesce the first two phase vectors from $\mathcal{F}$. To obtain $\mathcal{F}_2$, conjugate the second column $\mathcal{M}$ by Hadamard, then conjugate the second and third columns by CNOT, and coalesce the last two phase vectors in $\mathcal{F}$.

The overall simulation flow of Quipu is shown in Figure 9. Each of the steps in the flow is described in greater detail in [16] along with additional background material.

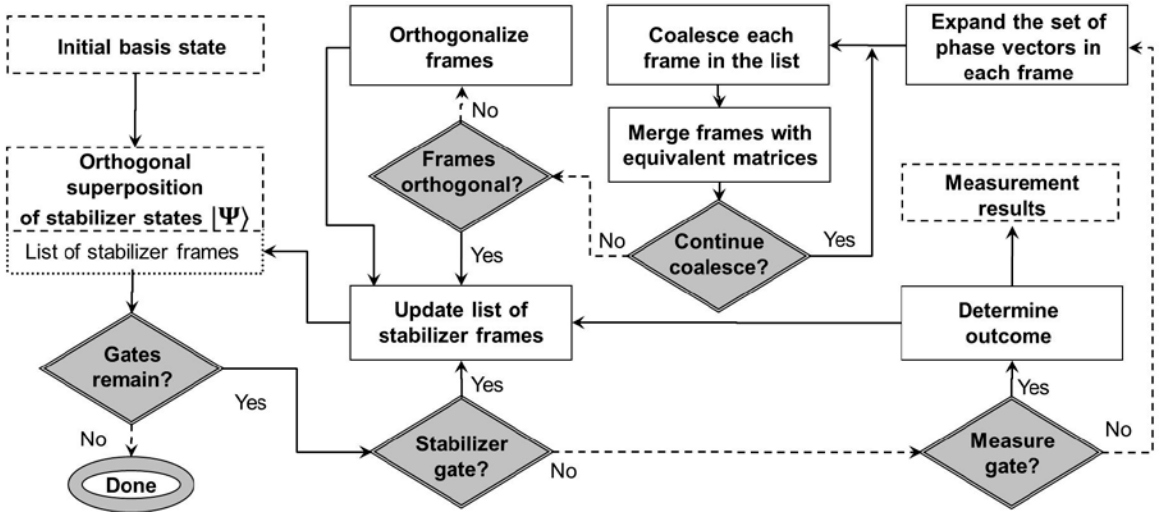
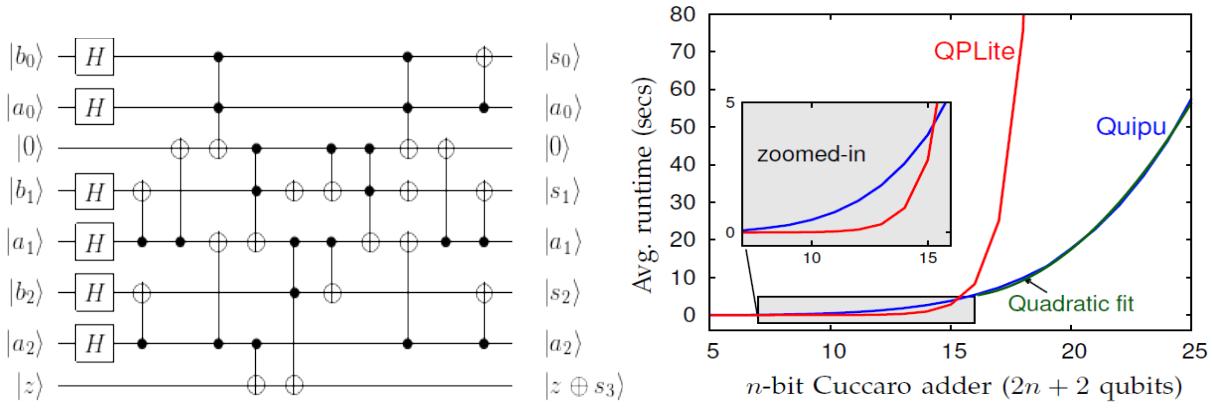


Figure 9. Overall simulation flow for Quipu.

Unlike other techniques based on compact representations of quantum states (e.g., using BDD data structures [7]), most frame-based operations are inherently parallel and lend themselves to multi-threaded implementation. The only step in Figure 9 that presents a bottleneck for a parallel implementation is the orthogonalization procedure, which requires communication across frames. All other processes can be executed on large subsets of frames independently.

Experimental results. Figure 10 shows that Quipu simulates certain quantum arithmetic circuits in polynomial time and space for input states consisting of equal superpositions of computational-basis states. On such instances, known linear-algebraic simulation techniques, such as the (state-of-the-art) BDD-based simulator QuIDDPro, take exponential time. We simulated quantum Fourier transform and quantum fault-tolerant circuits with Quipu, and the results demonstrate that our stabilizer-based technique leads to orders-of-magnitude improvement in runtime and memory as compared to QuIDDPro (Figure 11). While our technique uses more sophisticated mathematics and quantum-state modeling, it is significantly easier to implement and optimize. In particular, our multithreaded implementation of Quipu exhibited a 2X speed up on a four-core server. Additional results including comparisons with stabilizer circuits and quantum fault-tolerant circuits are covered in [16].



(a) **(b)**

Figure 10. (a) Ripple-carry (Cuccaro) adder for 3-bit numbers [17]. (b) Average runtime and memory needed by Quipu and QuIDDPro (QPLite) to simulate n-bit Cuccaro adders.

Figure 10(a) illustrates the Ripple-carry (Cuccaro) adder for 3-bit numbers [17]. Figure 10(b) shows the average runtime and memory needed by Quipu and QuIDDPro (QPLite) to simulate n-bit Cuccaro adders after a superposition of all computational basis states is obtained using a block of Hadamard gates. The quadratic function $f(x) = 0.5248x^2 - 15.815x + 123.86$ fits Quipu's curve with $R^2 = .9986$.

In Figure 11, we show the average runtime and memory needed by Quipu (single-threaded and multi-threaded) and QuIDDPro (QPLite) to simulate n-qubit QFT circuits, which contain $n(n + 1)/2$ gates. We used the input state for all benchmarks.

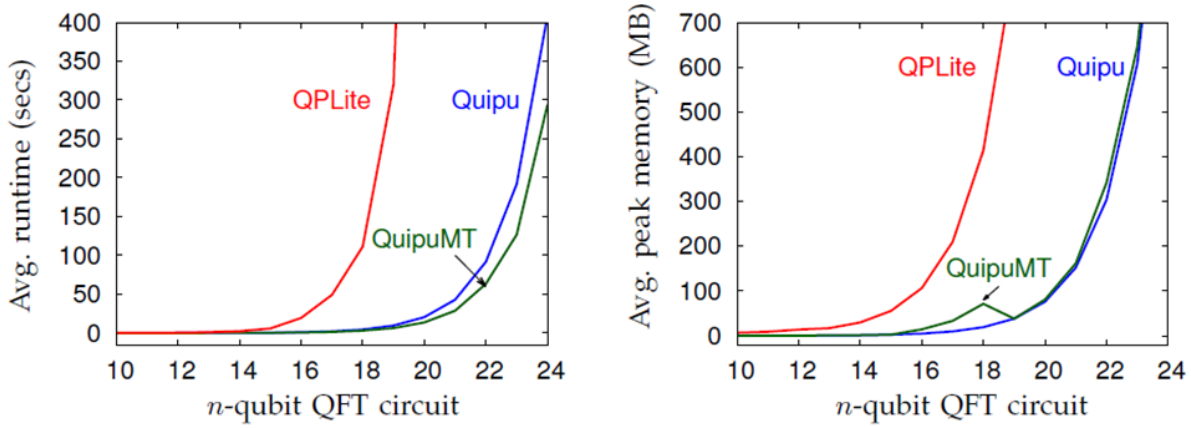


Figure 11. Average runtime and memory needed by Quipu (single-threaded and multi-threaded) and QuIDDPro (QPLite) to simulate n-qubit QFT circuits, which contain $n(n + 1)/2$ gates.

4.4 Simulating quantum circuits via stochastic computing.

We have investigated the use of an unconventional technique called stochastic computing (SC) for the simulation of quantum circuits [18] [19]. SC processes information represented by

random bit-streams of 0s and 1s. The bit-streams are called stochastic numbers (SNs) and are interpreted as probabilities. An n-bit SN N containing m 1s has the numerical value $p = m/n$, which is the probability of a randomly-observed bit of N being 1. In their basic unipolar form, SNs approximate numbers lying in the real interval [0,1].

The main attraction of SC is that arithmetic operations can be implemented by simple logic circuits; see Figure 12. For example, multiplication of two n-bit SNs p_1 and p_2 to form the product $p_1 \times p_2$ can be realized in n clock cycles by a two-input AND gate, provided p_1 and p_2 are from uncorrelated sources. Addition is implemented by a multiplexer in the scaled form $(p_1 + p_2)/2$, which ensures that the sum lies in the probability range [0,1]. Note that the scaling factor is supplied by an independent SN r of value 1/2, i.e., a purely random bit-stream. The remaining two circuits convert numbers between ordinary binary form N and stochastic form $p = N/2k$. They serve to interface conventional and stochastic circuits, and also to re-randomize SNs that have become unduly correlated. The (pseudo) random number generator in Figure 12c is typically implemented by a linear feedback shift register (LFSR) using well-known methods. Signed numbers can be handled by "bipolar" notation, which maps the SN range from [0,1] to [−1,1]. This results in minor changes to the component in Figure 12; for example, an XNOR gate rather than an AND gate is needed to perform bipolar multiplication; a multiplexer continues to serve as a scaled adder.

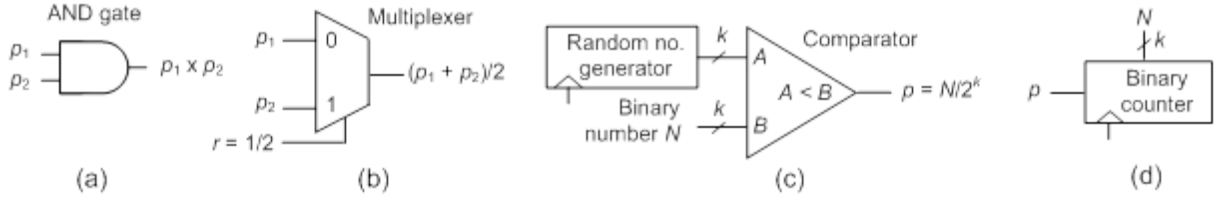


Figure 12. A selection of (unipolar) SC components: (a) multiplier, (b) scaled adder, (c) binary-to-stochastic converter, and (d) stochastic-to-binary converter.

Stochastic computing has some drawbacks that have limited it to a rather narrow range of applications. These include long computing times, inaccuracies due to bit-stream fluctuations, and the need for scaling. Recently, however, some very promising new applications for SC have emerged, notably decoding LDPC codes and image processing, which suggest that SC is much more practical than previously believed [18]. A major concern with SC has been that run-time, as measured by the number of clock cycles for a computation step, tends to grow exponentially with precision. Doubling bit-stream length from n to $2n$ increases precision by only 1 bit. However, SC can be much faster than it appears for several reasons: the basic clock cycle needed for operations like add and multiply is very short, SC tasks can often be parallelized, and precision can be varied on-the-fly (progressive precision).

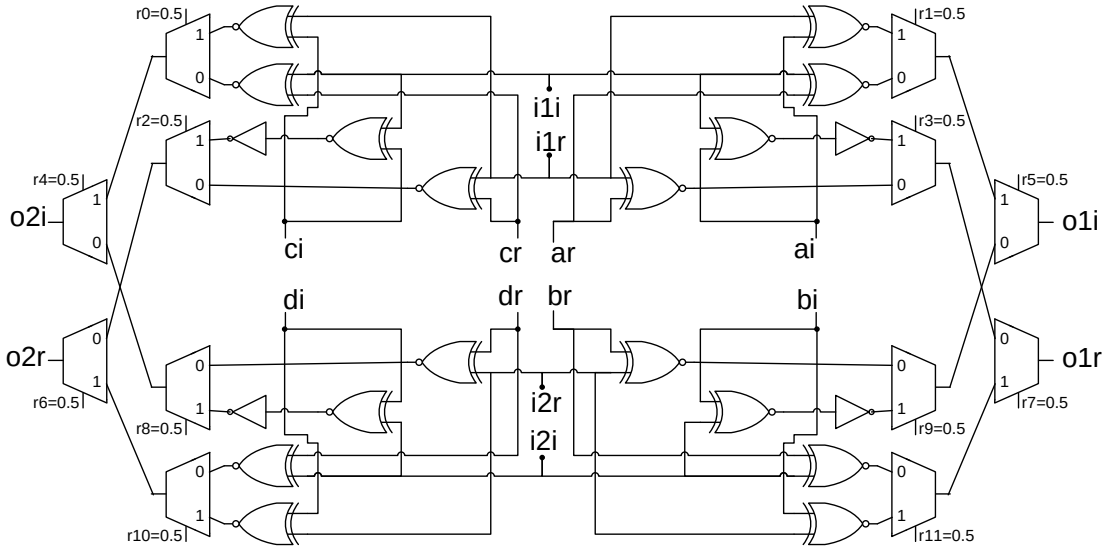


Figure 13. Stochastic circuit to simulate multiplication of a complex vector (i_1, i_2) by a complex matrix (*a,b;c,d*).

Modeling quantum circuits. To model a quantum circuit, we need to represent the quantum state $(a, b)^{\mathrm{T}}$ by SNs, where each of the probability amplitudes a and b is a complex number with real and imaginary parts n_r and n_i, respectively. The probability property $|a|^2 + |b|^2 = 1$ constrains both n_r and n_i to lie in the real interval [–1,1], which is precisely the range of the bipolar SNs. Hence, we use two such SNs to represent one complex probability amplitude, and 2^{n+2} SNs to denote an n-qubit state. To model a quantum gate operation, we map it to a conventional logic circuit that processes the SNs in an appropriate way.

A quantum gate G corresponds to a $2^n \times 2^n$ unitary matrix M, and application of G requires multiplying the current state of the quantum circuit by M. For $n = 1$, we get

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} i^1 \\ i^2 \end{pmatrix} = \begin{pmatrix} o^1 \\ o^2 \end{pmatrix}$$

This is converted to SC form in which each complex number is represented by an SN pair (n_r,n_i):

$$ai^1 + bi^2 = o^1$$
$$ci^1 + di^2 = o^2$$
$$(a_r, a_i)(i_r^1, i_i^1) + (b_r, b_i)(i_r^2, i_i^2) = (o_r^1, o_i^1)$$
$$(c_r, c_i)(i_r^1, i_i^1) + (d_r, d_i)(i_r^2, i_i^2) = (o_r^2, o_i^2)$$
$$o_r^1 = (a_r i_r^1 - a_i i_i^1) + (\mathrm{b}_r i_r^2 - b_i i_i^2)$$
$$o_r^2 = (c_r i_r^1 - c_i i_i^1) + (d_r i_\mathrm{r}^2 - d_i i_i^2)$$
$$o_i^1 = (a_r i_i^1 + \mathrm{a}_i i_\mathrm{r}^1) + (\mathrm{b}_r i_\mathrm{i}^2 + b_i i_\mathrm{r}^2)$$
$$o_i^2 = (c_r i_i^1 + c_i i_r^1) + (d_r i_\mathrm{i}^2 + d_i i_r^2)$$

Matrix multiplication is implemented by bipolar multipliers (XNOR gates) and adders (multiplexers); see Figure 13. Four multipliers and two adders are required for one complex multiplication. A total of $2^n - 1$ stochastic adders are used for all real components of the state vector; the same number of adders is used for the imaginary part. If the adders are arranged in a tree, the number of adders on all paths is n, and the resulting numbers must be multiplied by a compensating factor of 2^n.

Experimental results. We applied the foregoing SC-based simulation approach to quantum circuits of several representative types [18]. These include a class of stabilizer circuits called GHZ*n* (Greenberger-Horne-Zeilinger) circuits, which produce entangled states with a desired number of qubits n. Each target quantum circuit was automatically mapped to a stochastic version in VHDL, synthesized from the VHDL, and then transferred to a commercial FPGA platform. Further infrastructure, such as random number generators and counters needed for format conversion (Figure 12) was also synthesized.

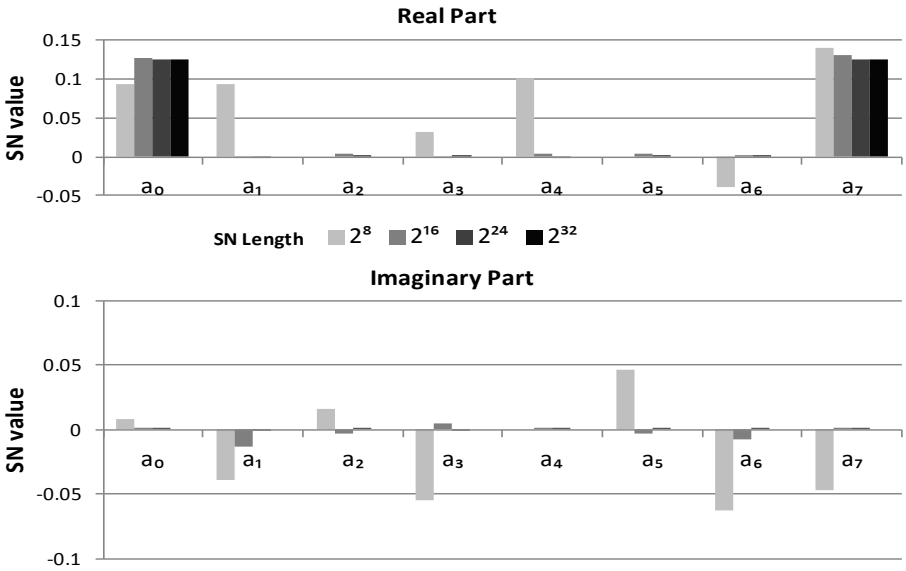


Figure 14. Stochastic simulation of GHZ3 demonstrating progressive precision.

Figure 14 shows the results of an experiment in simulating GHZ3. This circuit has three qubits, and therefore eight amplitudes. GHZ3 is applied to the input state $|000\rangle$ to produce the entangled output state $(1/\sqrt{2}, 0, 0, 0, 0, 0, 0, 1/\sqrt{2})^T$. The stochastic model for GHZ3 computes the state $(a_0, \ldots, a_7)^T$ represented by 16 SNs, the real and the imaginary parts of each amplitude a_i. Due to the scaling inherent in stochastic addition, all entries are divided by $2^3 = 8$. Therefore, the outcome of the simulation with no approximation error is $a_{ref} = (0.125, 0, 0, 0, 0, 0, 0, 0.125)^T$ where $0.125 = (1/\sqrt{2}) / 4\sqrt{2}$. Figure 14 shows the output values for various SN lengths. Here "a0r" stands for the real component of amplitude a_0. The bar corresponding to bit-length $n = 2^8$ is less

than 0.1, whereas the other three bars are close to the correct value of 0.125. It can be seen that the values obtained for $n = 2^8$ bear little resemblance to the exact value a_{ref}, the values for $n = 2^{16}$ are relatively close to a_{ref}, and the values for $n = 2^{24}$ and $n = 2^{32}$ perfectly match a_{ref}. This can be quantified by defining the average error as the Euclidian distance between the obtained distribution and a_{ref}. For $n = 2^8$, 2^{16}, 2^{24} and 2^{32}, the error is 0.03662, 0.00360, 0.00019 and 0, respectively. As expected very long SNs are needed for high accuracy, but precision increases in progressive fashion.

The above results also demonstrate that the simulation complexity as measured by the number of SC components needed can be orders of magnitude less than that of a classical version. The main limitation of the SC approach is the very long run-times required to achieve adequate accuracy. We have shown, however, that this problem can be mitigated by exploiting SC's progressive precision property. Other optimizations seem possible to further reduce model size and run-time, and are the t of on-going research.

5 CONCLUSIONS

Our research advanced the state of the art in simulating quantum circuits. In particular, we have further optimized the QuIDDPro software developed under prior support from DARPA and AFRL and continued to maintain it. We have also significantly broadened theory and algorithmic techniques necessary to extend the stabilizer formalism and encompass the simulation of non-stabilizer gates, while enjoying the efficiency of the stabilizer formalism on circuits with heavy quantum error correction. Developing such algorithms in full detail and implementing them as reusable software components laid foundation for high-performance simulation tools for quantum circuits dominated by stabilizer gates. Computational experiments indicate that our techniques outperform state-of-the-art simulators for practical instances of quantum arithmetic, quantum fault tolerant and quantum Fourier transform circuits.

Through the duration of our project we have cultivated productive relations with other research groups in the industry and academia. Our software QuIDDPro has been downloaded a number of times for educational and research purposes. It is also commonly used in scholarly publications for benchmarking purposes. We have coauthored an extensive journal paper on the geometry of stabilizer states with Dr. Andrew Cross from IBM (formerly at SAIC). PI Markov coauthored several papers with Dr. Mehdi Saeedi from USC, including one in Phys Rev A on speeding up Shor's algorithm. In 2012, PI Markov visited the Institute for Quantum Computing at Waterloo by invitation, presented research covered in this report, and had discussions with several researchers including the director of the institute Prof. Michele Mosca, as well as Dr. Dmitri Maslov (currently a Program Director at the National Science Foundation). PI Markov was also invited to visit Microsoft Research Faculty Summit in Redmond in 2013, gave a related presentation to the QuArC group, and interacted with Dr. Krysta Svore, Dr. Burton Smith, Dr. Dave Wecker and a number of other Microsoft researchers, as well as several visiting academic researchers (Prof. Umesh Vazirani, Prof. Scott Aaronson, Prof. Al Aho, Prof. Joe Traub). We are

continuing technical interactions with the QuArC group at Microsoft Research, including benchmarking of our new simulator software against their software. Univ. of Michigan Technology Transfer specialists are currently engaged with Microsoft Business Development specialists regarding potential licensing of our algorithms and tools to Microsoft.

6 PUBLICATIONS

Published or accepted for publication (numbering is consistent with References below)

[8] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 2, pp. 92:1-92:19, 2013.

[9] A. Paler, J. Kinseher, I. Polian and J. P. Hayes, “Approximate simulation of circuits with probabilistic behavior,” *Proc. 16th Symposium on Defect and Fault Tolerance in VLSI and Nano Systems* (DFT), pp. 95–100, New York, 2013.

[12] H. J. Garcia and I. L. Markov, "On the geometry of stabilizer states," *Quantum Information and Computation*, vol. 14, no. 7-8, pp. 683-720, 2014.

[18] H. J. Garcia and I. L. Markov, "Quipu: high-performance simulation of quantum circuits via stabilizer frames," in *ACM/IEEE International Conference on Computer Design* (ICCD), pp. 404-410, Asheville, 2013.

[20] H. J. Garcia and I. L. Markov, " Spinto: high-performance energy minimization in spin glasses," in *ACM/IEEE Design Automation and Test in Europe Conference* (DATE), pp. 160-165, Dresden, 2010.

[21] I. L. Markov and M. Saeedi, “Faster quantum number-factoring via circuit synthesis,” *APS Physical Review A*, vol. 87, no. 012310, 2013.

Under review

[22] H. J. Garcia and I. L. Markov, "Simulation of quantum circuits via stabilizer frames," *IEEE Transactions on Computers*, 2013.

7 REFERENCES

[1] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996. arXiv:quant-ph/9605043.

[2] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484-1509, 1997. arXiv:quant-ph/9508027.

[3] D. Gottesman, "Stabilizer codes and quantum error correction," arXiv:quant-ph/9705052, 1997.

[4] J. Preskill, "Fault Tolerant Quantum Computation," in *Introduction to Quantum Computation*, World Scientific, 1998. arXiv:quant-ph/9712048.

[5] M. Oskin, F. T. Chong and I. L. Chuang, "A practical architecture for reliable quantum computers," *IEEE Computer,* vol. 35, no. 1, pp. 79-87, 2002.

[6] K. M. Svore, A. V. Aho, A. W. Cross, I. L. Chuang and I. L. Markov, "A layered software architecture for quantum computing design tools," *IEEE Computer,* vol. 39, no. 1, pp. 74-83, 2006.

[7] G. F. Viamontes, I. L. Markov and J. P. Hayes, Quantum Circuit Simulation, Springer, 2009.

[8] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded Computing Systems,* vol. 12, no. 2, pp. 92:1-92:19, 2013.

[9] A. Paler, J. Kinseher, I. Polian and J. P. Hayes, "Approximate simulation of circuits with probabilistic behavior," in *Sixteenth Symposium on Defect and Fault Tolerance in VLSI and Nano Systems (DTF),* New York, 2013.

[10] D. Gottesman, "The Heisenberg representation of quantum computers," in *Proceedings of the twenty-second International Colloquium on Group Theoretical Methods in Physics*, 1998. arXiv:quant-ph/9807006.

[11] T. W. Hungerford, Algrebra, Springer, 1974.

[12] H. J. Garcia and I. L. Markov, "On the geometry of stabilizer states," *Quantum Information and Computation,* vol. 14, no. 7-8, pp. 683-720, 2014.

[13] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A*, vol. 70, no. 052328, 2004. arXiv:quant-ph/0406196.

[14] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[15] M. V. den Nest, "Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond," *Quantum Information and Computation*, vol. 10, pp. 258-271, 2010. arXiv:0811.0898.

[16] K. M. R. Audenaert and M. B. Plenio, "Entanglement on mixed stabiliser states: normal forms and reduction procedures," *New Journal of Physics*, vol. 7, no. 170, 2005. arXiv:quant-ph/0505036.

[17] D. Aharonov, "A simple proof that Toffoli and Hadamard are quantum universal," arXiv:quant-ph/0301040, 2003.

[18] H. J. Garcia and I. L. Markov, "Quipu: High-performance simulation of quantum circuits via stabilizer frames," in *IEEE International Conference on Computer Design*, Asheville, 2013.

[19] S. A. Cuccaro and e. al, "A new quantum ripple-carry addition circuit," arXiv:quant-ph/0410184, 2004.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AFRL – Air Force Research Laboratory

BDD – Binary Decision Diagram

DARPA – Defense Advanced Research Projects Agency

H – Hadamard gate

CNOT – controlled NOT gate

CZ – controlled-Z gate

P – phase gate

QECC – Quantum Error Correcting Codes

SC – stochastic computing

SN – stochastic number

$\boldsymbol{\mathcal{M}}$ – stabilizer matrix

$\boldsymbol{\mathcal{F}}$ – stabilizer frame

X – Pauli-X matrix (operator)

Y – Pauli-Y matrix (operator)

Z – Pauli-Z matrix (operator)