QDiff: Differential Testing of Quantum Software Stacks*

Jiyuan Wang, Qian Zhang, Harry Xu, and Miryung Kim

University of California, Los Angeles

*To be appeared in ASE 2021

Background knowledge

1. What is quantum computer and how developers use it
2. What is
    a. qubit
    b. quantum gate
    c. quantum circuit

Background knowledge

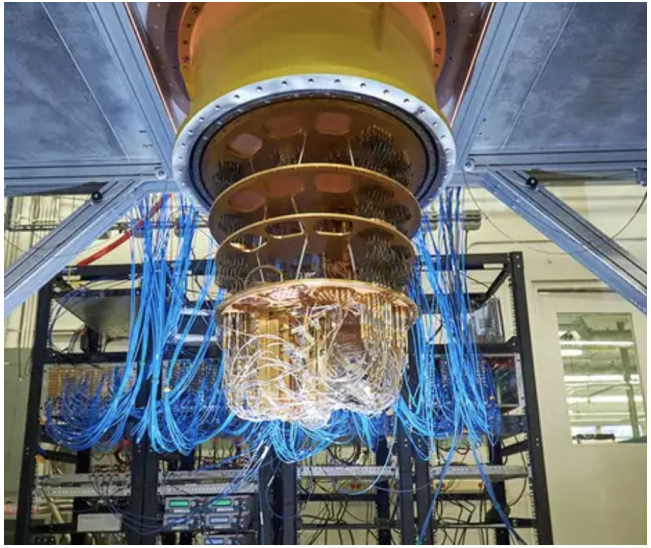
1. **What is quantum computer and how developers use it**
2. What is
   a. qubit
   b. quantum gate
   c. quantum circuit

What is quantum computer(QC)?

FPGA

GPU

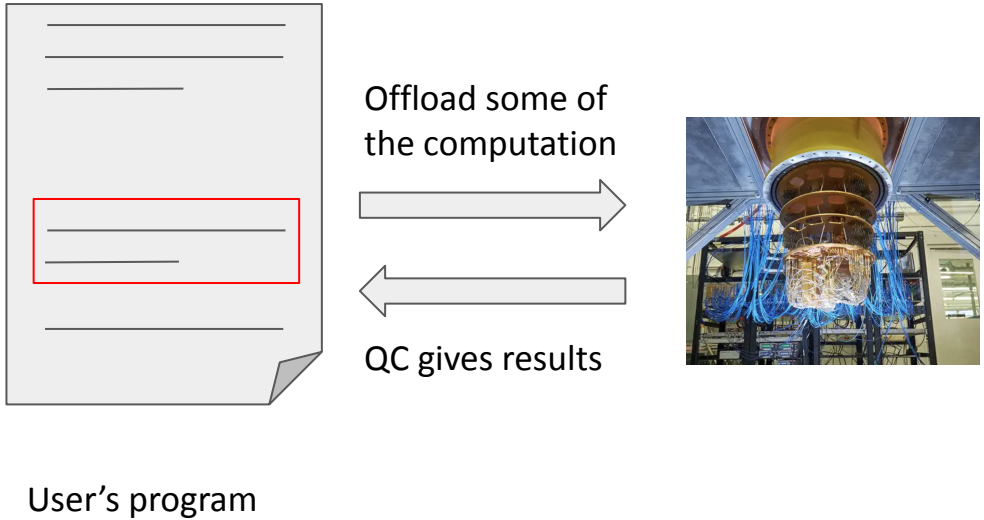
≒



Google’s Quantum Computer

From user’s perspective, Quantum computer is just a hardware accelerator

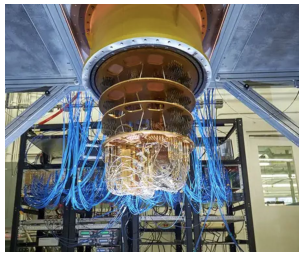
How people use it?



How people use it: Simon algorithm

Input : $f(x): \{0,1\}^n \rightarrow \{0,1\}^n$, $\exists S, f(x+S) = f(x)$

Output: S



Use QC to find $\mathbf{y_1}, \mathbf{y_2} \ldots \mathbf{y_n}$

$$\begin{cases} \mathbf{S} \cdot \mathbf{y_1} = 0 \\ \mathbf{S} \cdot \mathbf{y_2} = 0 \\ \ldots\ldots \\ \mathbf{S} \cdot \mathbf{y_n} = 0 \end{cases}$$

Use classical computer to determine **S** by Gaussian elimination

Performance improvement: $\Omega(2^{n/2})$ -> $O(n+n^3)$

Background knowledge

1. What is quantum computer and how developers use it
2. What is
    a. **qubit**
    b. quantum gate
    c. quantum circuit

What is qubit?

Classical bit: 0/1

Qubit: $\alpha|0\rangle+\beta|1\rangle$, $|\alpha|^2+|\beta|^2=1$

Measurement (read):

$\alpha|0\rangle+\beta|1\rangle \xRightarrow{M} |\alpha|^2 \rightarrow 0,\ |\beta|^2 \rightarrow 1$

What is qubit: T1 time

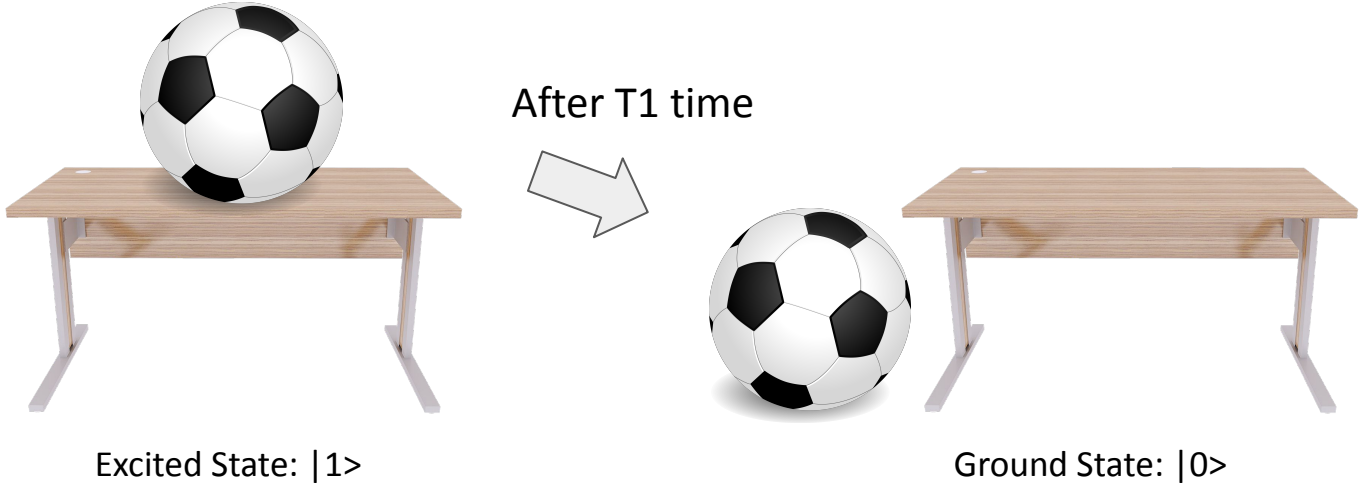
Where does |1> and |0> come from?

QC uses excited state to represent |1> and ground state to represent |0>

Excited State: |1>

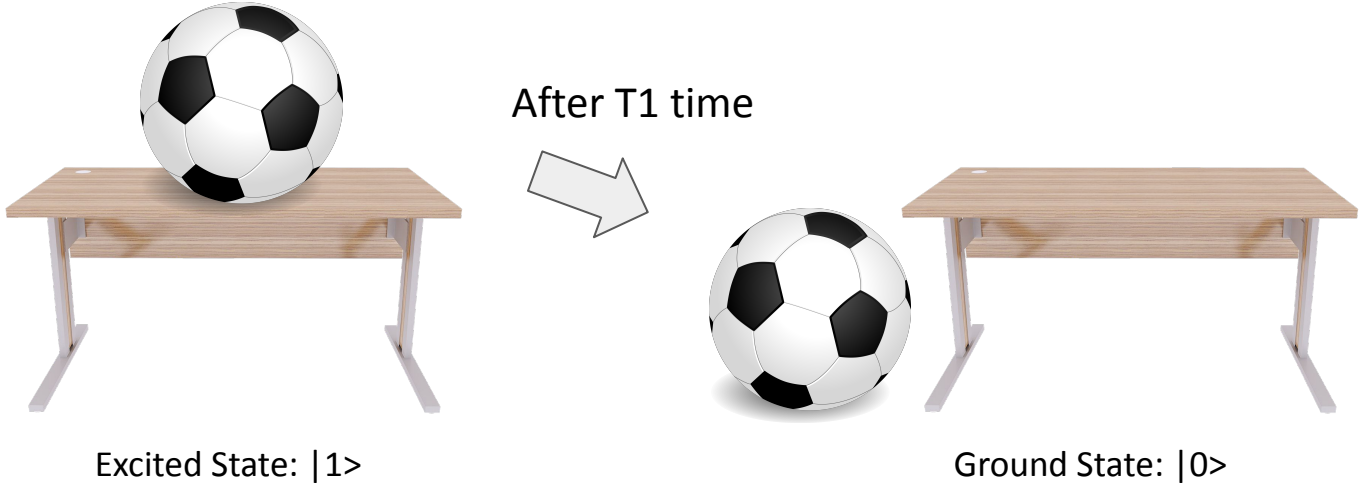
Ground State: |0>

What is qubit: T1 time



Qubit will gradually decay from high energy state (excited state |1>) to low energy state (ground state |0>)

What is qubit: T1 time



Qubit will become unreliable after T1 time

Circuit execution time should be less than T1 time!

Background knowledge

1. What is quantum computer and how developers use it
2. What is
    a. qubit
    b. quantum gate
    c. quantum circuit

What is quantum gate?

Classical gate: not, or, and, xor, …

Quantum gate: X (not), CNOT (xor), ….

X gate:

$\alpha|0\rangle+\beta|1\rangle \Longrightarrow \boxed{X} \Longrightarrow \beta|0\rangle+\alpha|1\rangle$

CNOT gate:

$(a|0\rangle+b|1\rangle)(c|0\rangle+d|1\rangle) \Longrightarrow (ac|00\rangle+ad|01\rangle)(bc|11\rangle+bd|10\rangle)$

What is quantum gate: error rate

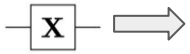
Error rate of CNOT (~1%) >> error rate of 1-qubit gate (<0.1%)[1]

2-qubit gate dominates the error rate. If one quantum circuit has much more 2-qubit gates than its equivalent versions, then it may produce different results.

1. IBM quantum computer, https://quantum-computing.ibm.com/

What is quantum gate: unitary matrix

Qubit: $\alpha|0\rangle+\beta|1\rangle \Rightarrow (\alpha, \beta)^T$

Quantum gate: $(\alpha, \beta)^T \Rightarrow$  $\Rightarrow (\beta, \alpha)^T$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Quantum gate should ensure $|\alpha|^2+|\beta|^2=1$! $\Rightarrow$ unitary matrix

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

What is quantum gate: unitary matrix

Matrix of gate sequence = multiplication of each unitary matrix (quantum gate)

$$HXH = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} = Z$$

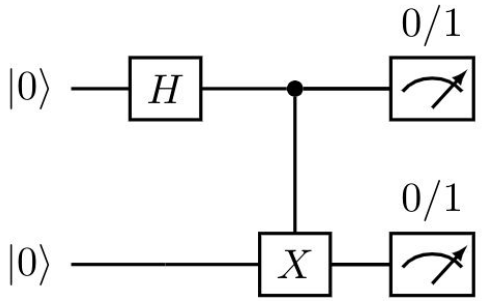
Two quantum gate sequences are equivalent as long as they yield same matrix representation!

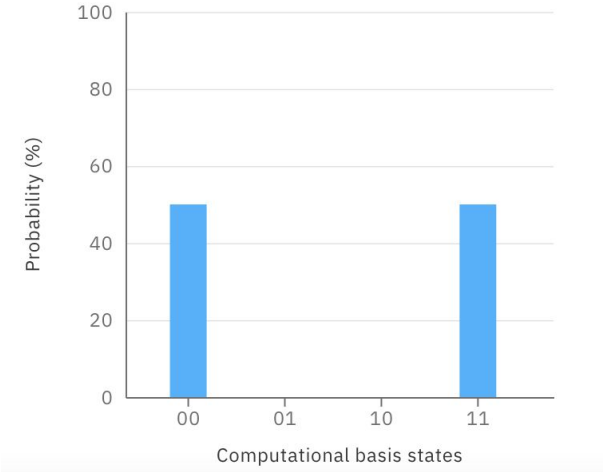
Background knowledge

1. What is quantum computer and how developers use it
2. What is
   a. qubit
   b. quantum gate
   c. quantum circuit

What is quantum circuit?

Quantum circuit = qubits + quantum gate sequence + measurement





For most of the time, we will run and measure the circuits multiple times to ensure the correctness. The result becomes a distribution.

Summary of Quantum basics

1. Two quantum gate sequences are equivalent as long as they yield same matrix representation
2. Developers need to repeatedly run and measure the result of quantum circuits, and result becomes a distribution
3. The execution time of circuit should be less than T1 time
4. The circuit cannot contain too many 2-qubit gates

QDiff: Differential Testing of Quantum Software Stacks

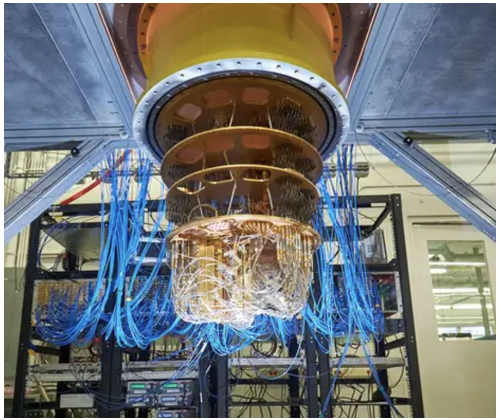
Jiyuan Wang, Qian Zhang, Harry Xu, and Miryung Kim

University of California, Los Angeles

People are using hardware accelerators

FPGA

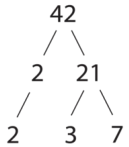
GPU



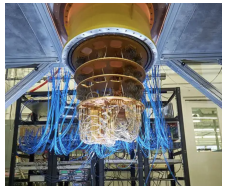
Quantum computer

QC is a promising hardware accelerator

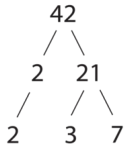
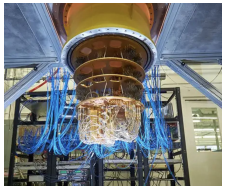

Classical: $O(2^n)$




Classical: $O(e^{P(\log N)})$

Quantum: $O(2^{\sqrt{n}})$

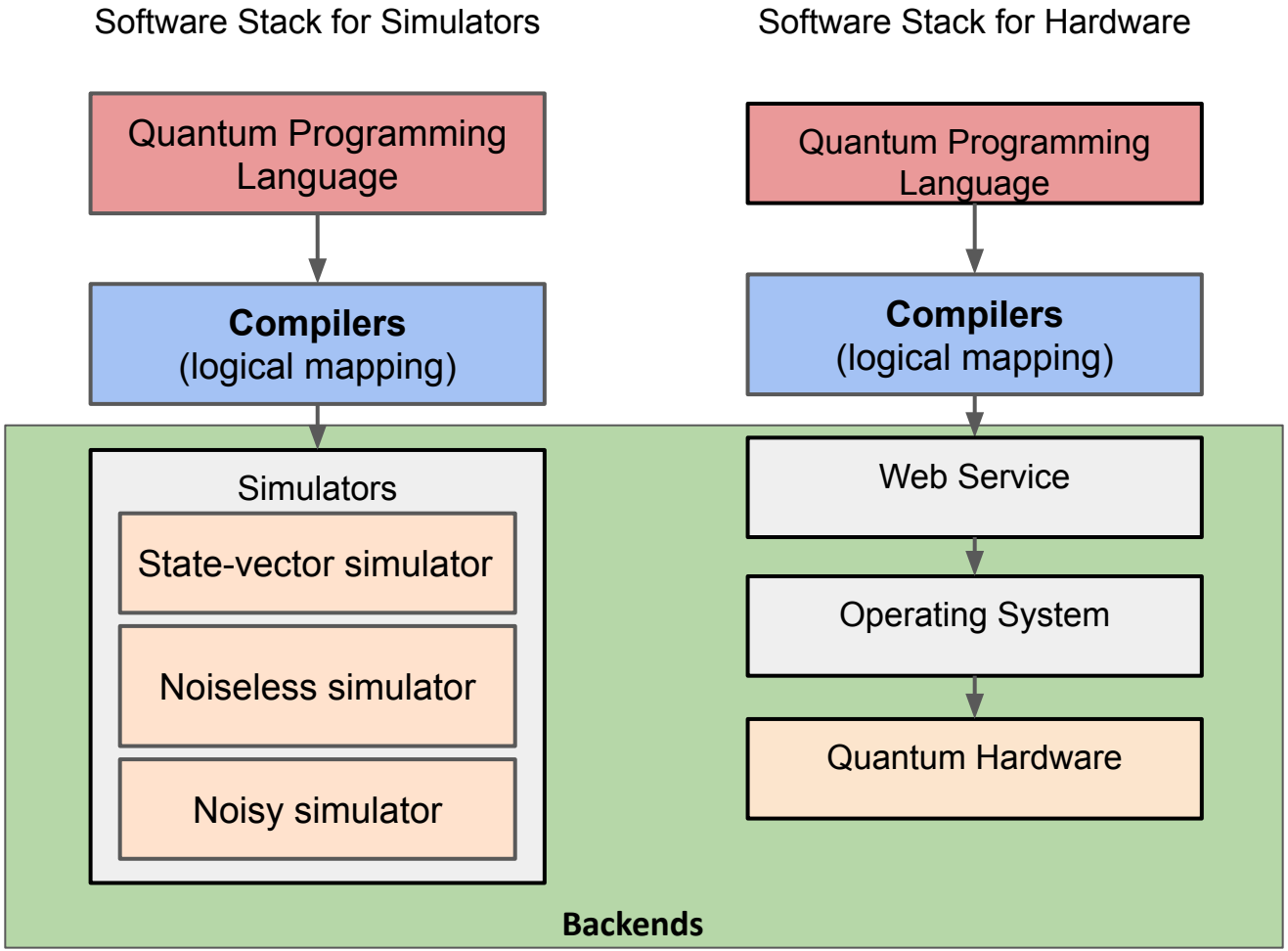



Quantum: O(P(logN))

Grover algorithm can speed up unstructured search with quantum computer

Shor algorithm can speed up integer factorization with quantum computer

Quantum Software Stacks (QSS) are becoming publicly available.

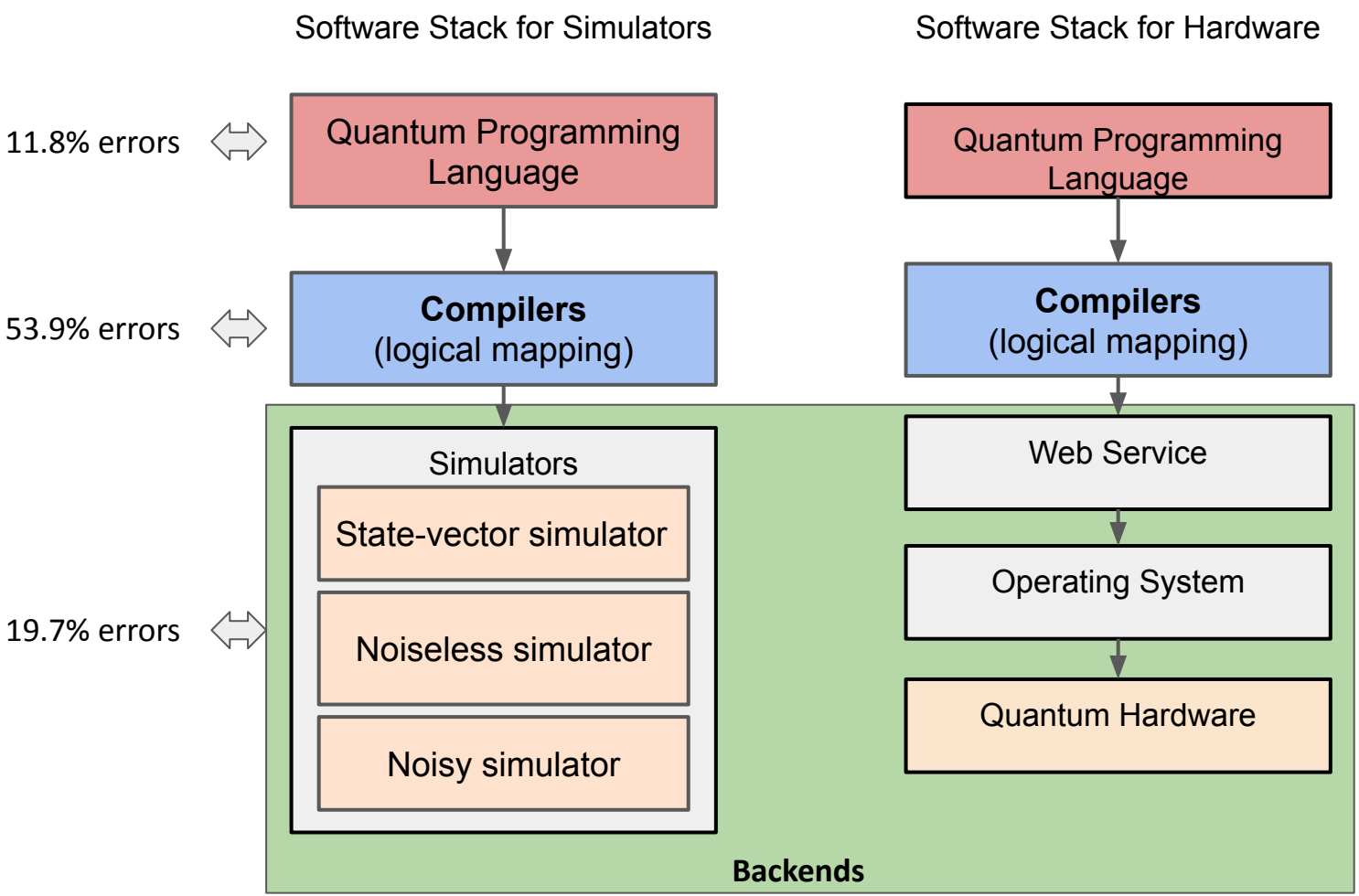

Software Stack for Simulators
Quantum Programming Language
Compilers (logical mapping)
Simulators
State-vector simulator
Noiseless simulator
Noisy simulator
Software Stack for Hardware
Quantum Programming Language
Compilers (logical mapping)
Web Service
Operating System
Quantum Hardware
Backends

Study of real world QSS errors: errors happen in all components of QSS

- We studied 76 bugs posted on **StackOverflow** and **Github** regarding Qiksit, Cirq, and Pyquil.

Survey Statistics	
Keywords Searched	Quantum framework bugs, error, unexpected behavior
Posts Studied in total	76 posts
Common Fault Types	4

Error Types	Example
Bugs in Different Simulation Backends	Pyquil crashes on PyQvm with controlled gate
Bugs in API implementations of Quantum Gates	Qiskit crashes on cnot() gate
Bugs in Compiler Setting	Qiskit has bugs in compiler optimization level
Bugs in installation / interaction with other tools	Qiskit installation fails with Python 3.9


Software Stack for Simulators
Software Stack for Hardware
11.8% errors
Quantum Programming Language
Quantum Programming Language
53.9% errors
Compilers (logical mapping)
Compilers (logical mapping)
Simulators
State-vector simulator
19.7% errors
Noiseless simulator
Noisy simulator
Web Service
Operating System
Quantum Hardware
Backends

Errors in QSS are hard to identified due to...

Challenge 1: inherent probabilistic nature of quantum makes error hard to identified

Challenge 2: there are very few quantum programs out here

Challenge 3: quantum computer resources are expensive

Errors in QSS are hard to identified due to...

Challenge 1: inherent probabilistic nature of quantum makes error hard to identified	→	Need to find a way to identify the correctness of program outcome
Challenge 2: there are very few quantum programs out here	→	Need to find a way to diversify test program
Challenge 3: quantum computer resources are expensive	→	Need to avoid worthless invocation of quantum hardware

QDiff’s solution

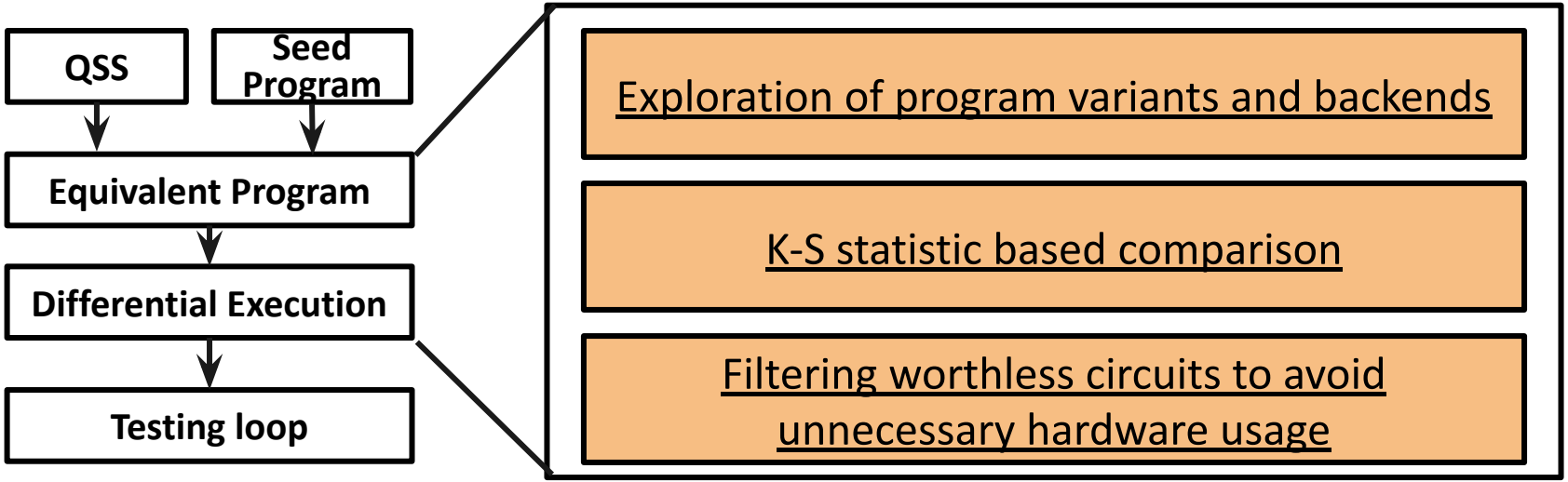
Challenge 1: inherent probabilistic nature of quantum makes error hard to identified	→	Differential testing with K-S based distribution comparison to identify errors
Challenge 2: there are very few quantum programs out here	→	Equivalent transformation and mutation to generate quantum programs
Challenge 3: quantum computer resources are expensive	→	With two rules, T1 time and 2-qubit gate number, to avoid worthless invocation of QC

Takeaway

- As a first step, **QDiff** enables differential testing on **QSS** with
  - Equivalent quantum program generation
  - Multiple checking including **Kolmogorov–Smirnov test**[1] based results comparison
  - Filtering mechanism to speed up the test oracle and avoid worthless hardware usage
- We found four software crashes in the **Cirq** and **Pyquil** simulations**,** and 29 divergence cases with 2 possible root causes on **IBM Quantum Computers**

1. Kolmogorov A. Sulla determinazione empirica di una legge di distribuzione. G. Inst. Ital. Attuari. 1933, **4**: 83.

QDiff approach overview



We found four software crashes in the **Cirq** and **Pyquil** simulations, and 29 divergence cases with 2 possible root causes on **IBM Quantum Computers**

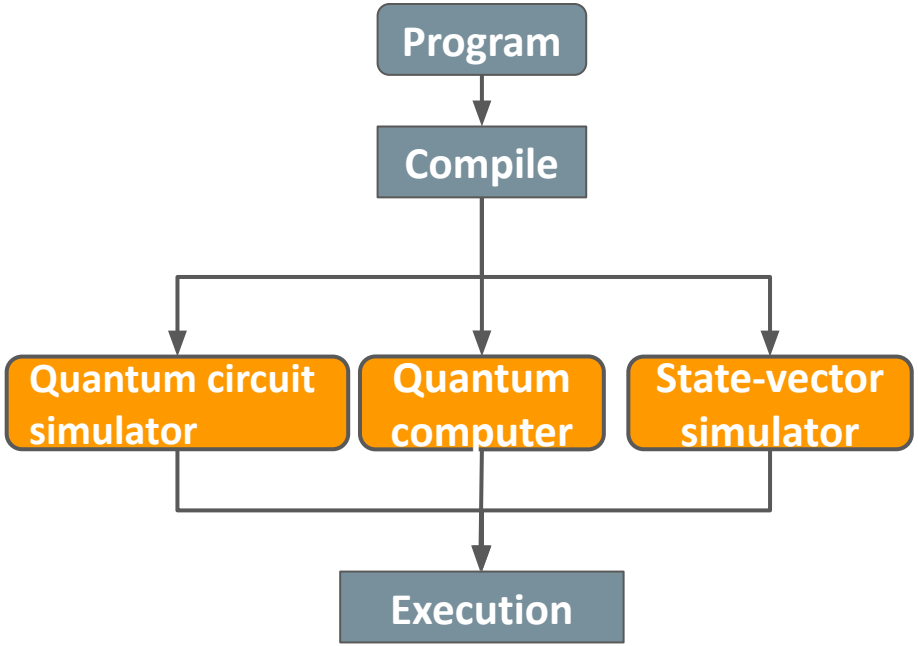
Challenge 1: Equivalent program transformation

- Equivalent Gate Transformation

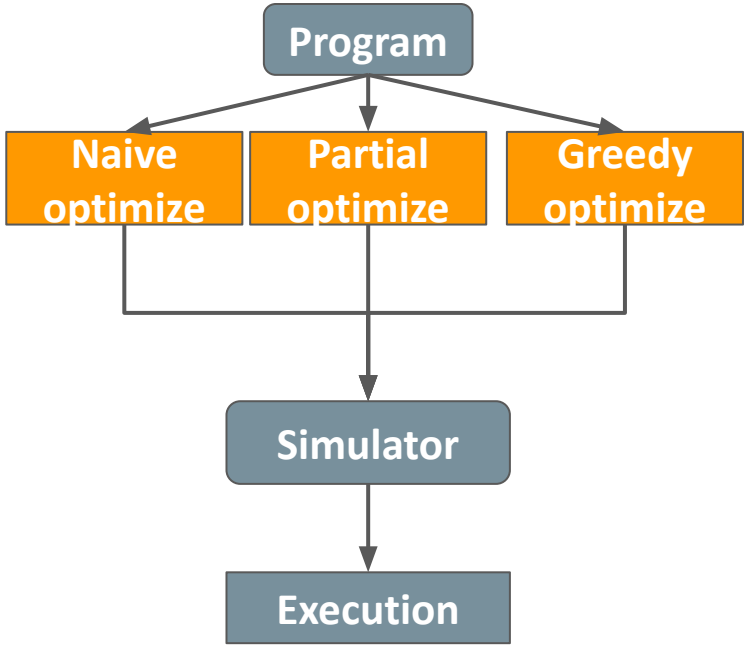
Rule ID	Original Gate	Equivalent Gate
G1	SWAP(p,q)	CNOT(p,q)CNOT(q,p)CNOT(p,q)
G2	S(p)	T(p)T(p)
G3	X(p)	H(p)S(p)S(p)H(p)
G4	Z(p)	S(p)S(p)
G5	CZ(p,q)	H(q)CZ(p,q)H(q)
G6	CZ(p,q)CZ(p,q)	Merged to Identity Matrix
G7	CCNOT(p,q)	6 CNOT gates with 9 one-qubit gate
…	…	…

Challenge 1: Equivalent program transformation

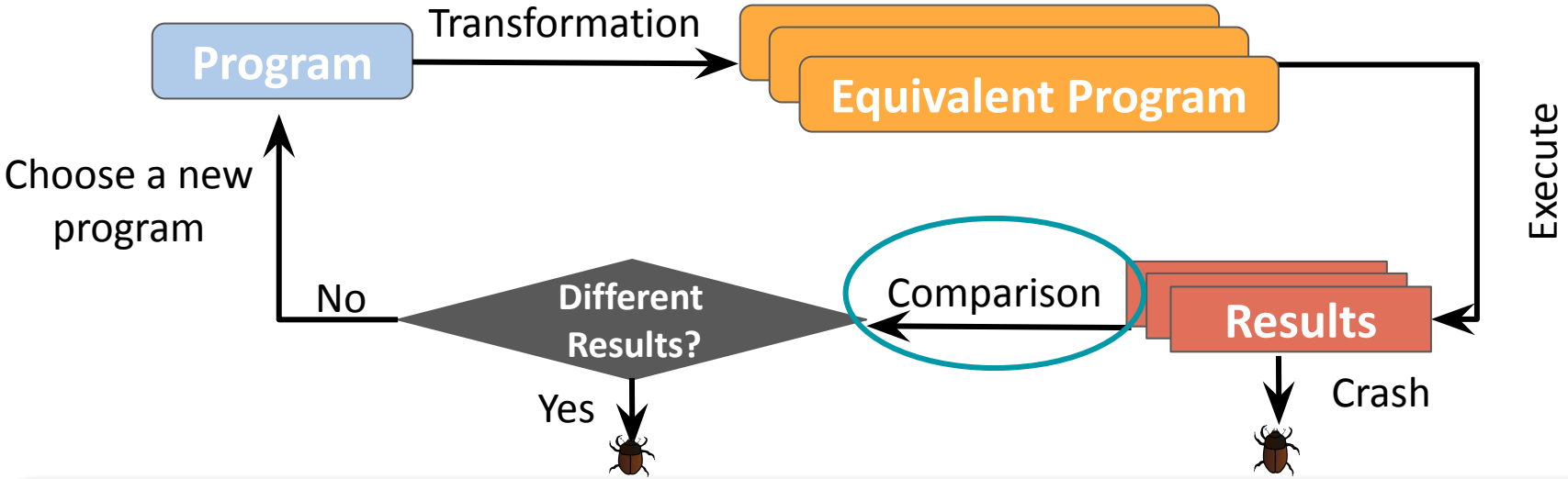
- Backends Exploration



- Compiler Settings Exploration



Challenge 2: Differential testing and comparison



QDiff Uses K-S statistic[1] to compare the distributions. We report difference if K-S value is larger than a threshold.

1. Kolmogorov A. Sulla determinazione empirica di una legge di distribuzione. G. Inst. Ital. Attuari. 1933, **4**: 83.

Challenge 2: Differential testing and comparison

- **Cumulative Possibility** $S(x) = \Sigma\, i/n$ for $i \leq x$
  - n is the total trail number. In the example, n=1000
- **K-S Statistic** $D = \max|S_{N1}(x)-S_{N2}(x)|$
  - D=0.06 in the example.
- Report difference if **D** is larger than a predefined threshold
- Measurement trail number n is determined by p-value and threshold t

Measurement Distribution	'00'	'01'	'10'	'11'
N1	250	250	250	250
N2	247	247	253	253

Cumulative Possibility	'00'	'01'	'10'	'11'
S1	0.250	0.500	0.750	1.00
S2	0.247	0.494	0.747	1.00

Challenge 3: Filtering mechanism to speed up testing and avoid worthless QC invocations

- We obviet the invocation of two kinds of worthless circuits:
  - If circuit execution time is likely larger than T1 time
  - If circuit has too many 2-qubit gates

Quantum Circuits

Analysis 2-qubit gate number / circuit depth and execution time

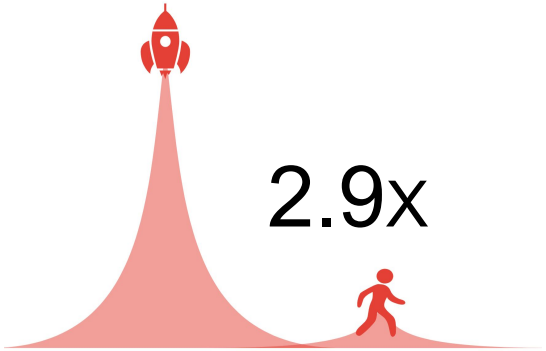
Execute on IBM hardware

Evaluation: Generation & Speed up & Results

730

14799

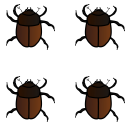
We generate 730 variants with semantic modifying mutations, and 14799 circuits with equivalent transformation

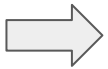


We reduce 66% unnecessary quantum hardware invocation

We found 4 crashes with simulation and 29 divergences with quantum hardware

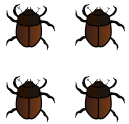
Evaluation: Results Analysis

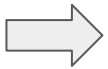




4 crashes happens with quantum simulation. 1 is in simulator backends, 2 are in compiler settings, and 1 is in API implementations of high level quantum gates. They have been confirmed with QSS developers.

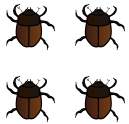
Evaluation: Results Analysis





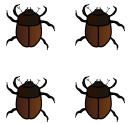
```
qvm = get_qc('3q-qvm')
try:
    qvm.run(Program())
except:
    print("empty")
…..
qvm.run(Program(H(1)))
```

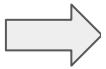
Evaluation: Results Analysis

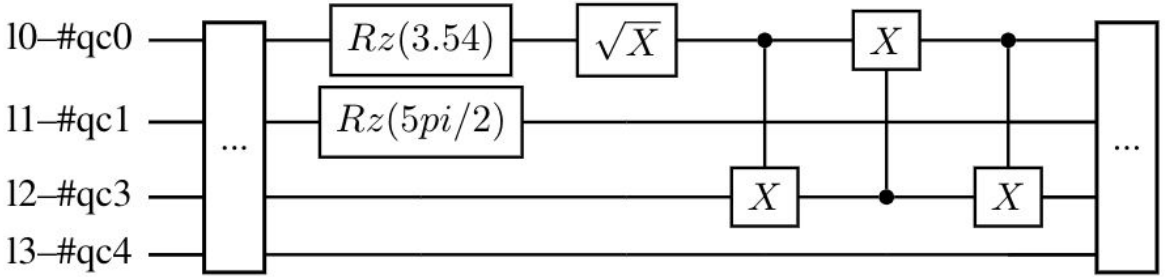


We manually analysis 29 divergence cases in hardware executions. We conclude 2 possible root causes for 25 of them (9 are related with qubit dephasing, 16 are related with 2-qubit gate mapping). For the rest of 4, we can't determine the root cause.

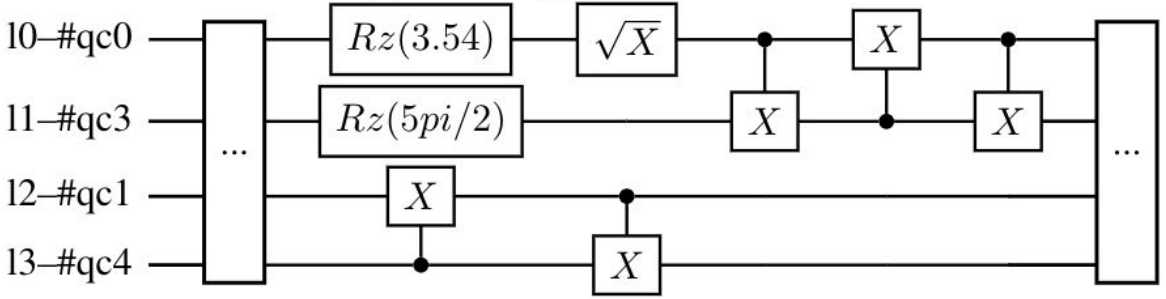
Evaluation: Results Analysis







(a) ”Divergent” circuits



(b) ”Good” circuits

QDiff: Differential Testing of Quantum Software Stacks

Jiyuan Wang, Qian Zhang, Harry Xu, and Miryung Kim

University of California, Los Angeles

- We adapt differential testing to quantum Software Stacks
- QDiff combines:
  - Automated generation of equivalent programs with gate transformation, backend and compiler setting exploration to reflect real world error types;
  - K-S based comparison to report the diff of quantum program's results

Future direction

- We are investigating better transformations and mutations for *quantum hardware*
- QDiff generates semantically equivalent programs with a limited scope of transformation, which are very likely to exercise the *same* and a *limited* part of quantum framework.
  - We will investigate test adequacy criteria appropriate for QSS to guide test program generation and mutations
- We will investigate how to test *quantum programs* and how to *debug* the found bugs automatically