ACM Journal on Emerging Technologies in Computing Systems

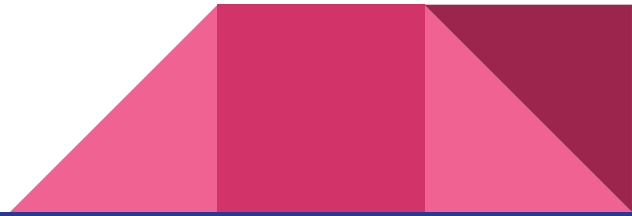
Extracting Success from IBM’s 20-Qubit Machines Using Error-Aware Compilation

Shin Nishio, Yulu Pan, Takahiko Satoh, Hideharu Amano, Rodney Van Meter

Presented by: Xuenan Wang

Outline

- Problem to solve
- Background
- Experiment
- Results and Analysis



The problem

"NISQ (Noisy, Intermediate-Scale Quantum) computing requires error mitigation to achieve meaningful computation."

The authors needed to develop a compilation tool in order to maximize the success probability of real-world subroutines such as an adder circuit.

The problem

- Quantum computers exist, and can surpass classical computers on a range of important problems.
- Full realization of quantum error correction remains out of reach.
- Need error-aware compilers for NISQ computing.

mapping of program qubits to machine qubits

"Not All Qubits Are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers" by Tannu and Qureshi

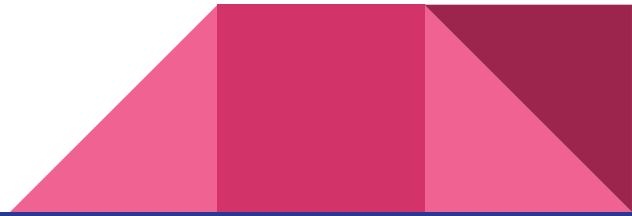
"Full-Stack, Real-System Quantum Computer Studies: Architectural Comparisons and Design Insights" by Murali et al.

setting some candidate path

solving a constraint equation

Background

1. Machine-level compilation
2. IBM's 20-qubits quantum processor
3. Error model
4. Tomography
5. Randomized benchmarking
6. Architecture-aware compilation



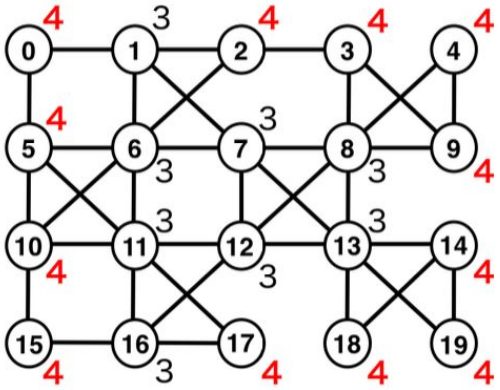
Machine-level compilation

1. Decomposition of high-level language constructs into a series of one- and two-qubit operations that can be executed on the target system;

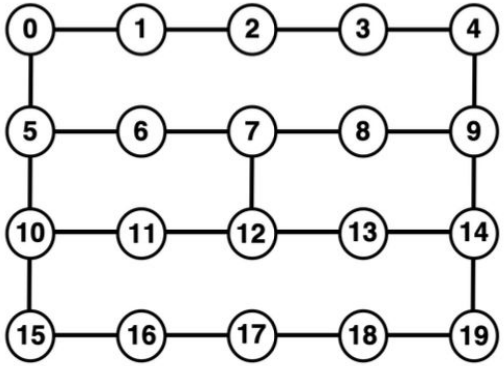
2. Mapping of the variables defined by the programmer to locations in the system, in tandem with generation of appropriate execution of gates between qubits unfortunately placed far apart;

3. Generation of low-level control for the hardware itself.

IBM's 20-qubits quantum processor



Tokyo



Poughkeepsie

Each vertex represents a qubit, and each edge indicates that a CNOT gate can be executed between the two qubits.

The number outside the circles indicate the eccentricity, or maximum distance to another qubit.

Error model

1. Single-Qubit Gate errors -> G
   a. Unitary bit flips
   b. Unitary phase flips

2. Bi-Qubit Gate errors(CNOT error) -> B
   a. One or both qubits value flips
   b. One or both qubits phase flips
   c. IMPORTANT, can propagate errors from one qubit to another

3. State Preparation and measurement(SPAM) errors -> S
   a. Can appear only once per qubit
   b. Less cumulative impact

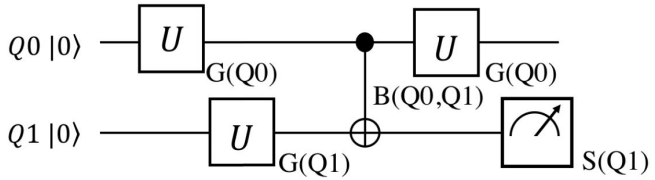


Figure 4: Errors and the gates with which they occur

Tomography

Two primary forms: **state tomography** & **process tomography**

Tells us how well we have done at creating our desired state, or how well a particular process (gate or set of gates) works, respectively.

The number of possible states naturally grows exponentially with the number of qubits. Need to test not just for bit flip errors, but also for phase flip errors.

Randomized benchmarking

1. Randomly select m gates from the *Clifford* group and arrange them in any order;
2. Select the *Clifford* gate (or gates, if performing RB on more than one qubit) that will reverse the operation of the entire preceding sequence of m *Clifford* gates, C_{m+1} execute this as the m+1th gate;
3. Measure the qubit(s). If the output state is not equal to the input state, then an error has occurred somewhere in the whole circuit;
4. Change the number m and perform steps 1 to 3 again.

Clifford gates are the elements of the *Clifford* group, a set of mathematical transformations which effect permutations of the Pauli operators.

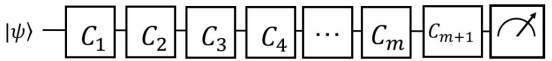


Figure 5: Randomized Benchmarking

$$C_{m+1} = \left(\prod_{i=1}^{m} C_i \right)^{\dagger}$$

In this way, it's possible to estimate the average value of errors per gate included in the gate set.

Architecture-aware compilation

Previously:
Focus on the impact on execution time.

Recently:
Pay attention to fidelity improvements.

Experiment

1. Long-distance CNOTs and making choices
   a. Path selection for remote CNOT
   b. Circuit selection for remote CNOT
2. Compiling complete programs
   a. Search space
   b. The optimization algorithm
   c. Subroutines for compilation algorithm
   d. Experimental evaluation of compilation

Long-distance CNOTs and making choices

WHY?

Each G, B, and S error above will have an error rate Ɛ dependent on type and location.

$$ESP = \prod_i (1 - \varepsilon_i).$$

Estimated Success Probability

Path selection for remote CNOT

Problem 1: CNOT Path Selection

The programmer wishes to execute the circuit shown in Fig. 6. When the starting point (control qubit) and the end point (target qubit) for the Bi-Qubit gate (CNOT) are not neighbors, which path is the highest fidelity?

Result 1

ESP was able to select the better of two routes with accuracy of 70% at 2 hops, 66.6% at 3 hops and 62.5% at 4 hops.

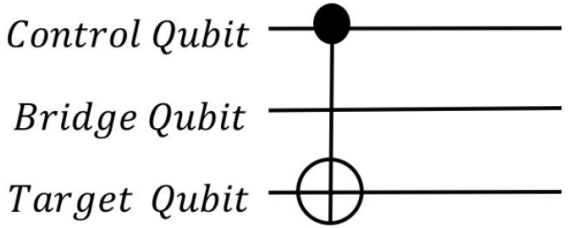
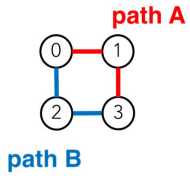
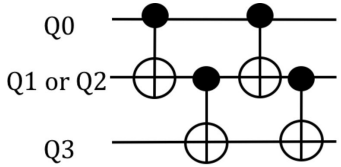


Figure 6: Circuit A, a CNOT gate skipping across another qubit

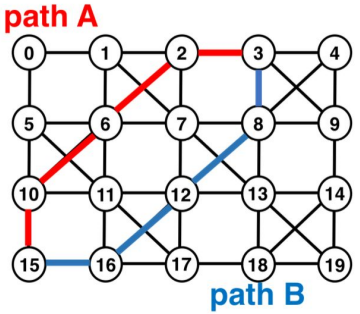
Path selection for remote CNOT



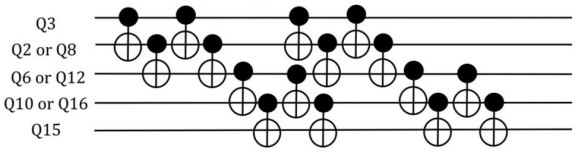
(a) 2-hop path selection



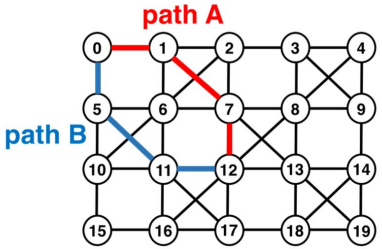
(b) 2-hop path circuit



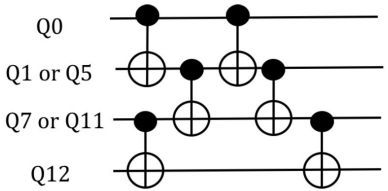
(a) 4-hop path selection



(b) 4-hop path circuit



(a) 3-hop path selection



(b) 3-hop path circuit

Circuit selection for remote CNOT

Problem 2: Circuit Selection

The programmer wishes to execute the circuit shown in Fig. 6. When the CNOT gate is executed with remote qubits, there are multiple theoretically equivalent circuits as shown in Fig. 10.
However, considering errors, these are not necessarily equivalent. Which circuit has the highest fidelity?

Result 2

ESP was able to select the optimal circuit among three candidates with accuracy of 40%.

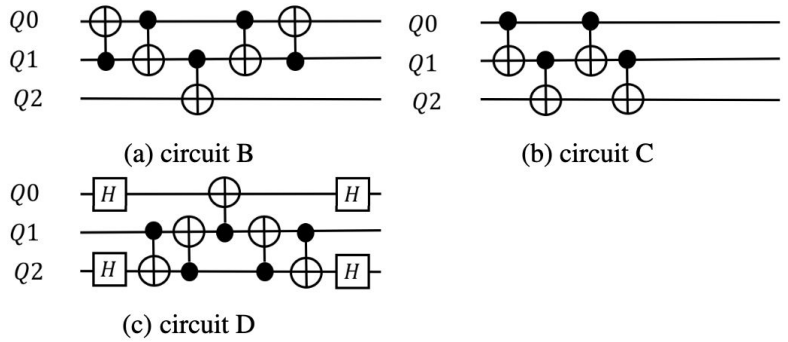
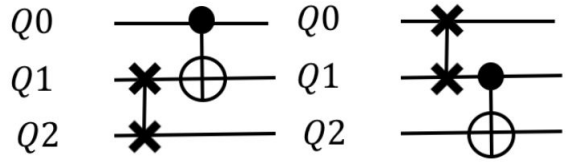


Figure 10: Circuits equivalent to the circuit shown in Fig. 6

Circuit selection for remote CNOT



(a) SWAP Q1, Q2 (b) SWAP Q0, Q1

Figure 12: Circuits including SWAP gates

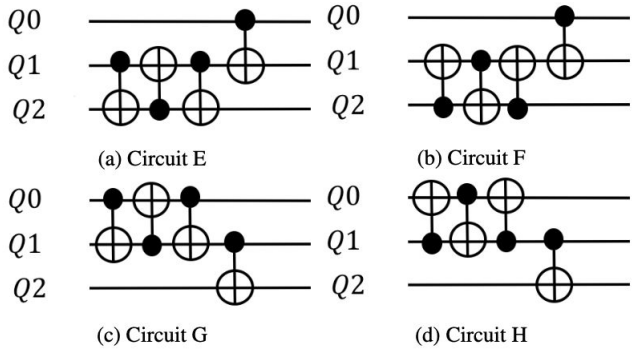


Figure 13: Circuits equivalent to Fig. 12. SWAP gate can be implemented with three CNOT gates.

Problem 2′: Circuit Selection (including SWAP)

Perform the same circuit as Problem 2 with additional circuit options, including permitting the relocation of a qubit, using the SWAP gate shown in Fig. 12. Which circuit is the highest fidelity?

Result 2′

ESP was able to select the optimum circuit from among seven candidates 25% of the time. Fig.14 shows the fidelity of selected circuit relative to the other candidate circuits.

Circuit selection for remote CNOT

Problem 3: Circuit Selection (on Poughkeepsie)

Perform the same circuit as Problem2âĂŹ on the other 20-qubit processor called Poughkeepsie. Is there any change in the reliability of circuit selection by ESP?

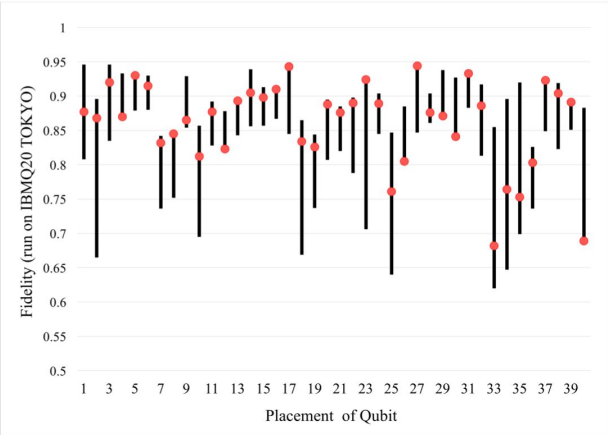
Result 3

ESP was able to select the optimum circuit from among seven candidates 43% of the time. Fig.15 shows the fidelity of selected circuit relative to the other candidate circuits.

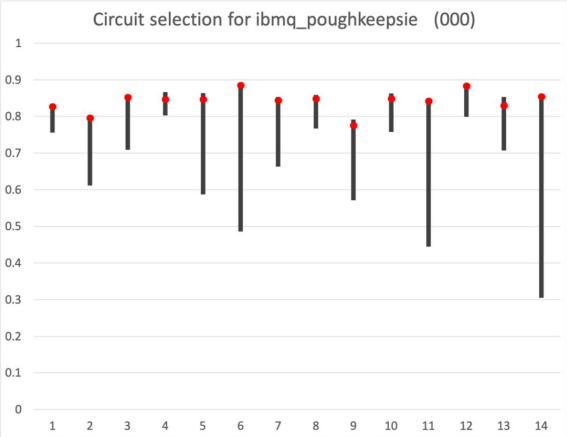
Table 1: the performance specifications of two IBMQ 20-qubit systems named Tokyo and Poughkeepsie

	tokyo (2nd gen 20-qubit system)	poughkeepsie (3rd gen 20-qubit system)
Mean of Two-qubit (CNOT) error rates $\times 10^{-2}$	2.84	2.25
best	1.47	1.11
worst	7.12	6.11
Mean of Single-qubit error rates $\times 10^{-3}$	1.99	1.07
best	0.64	0.52
worst	6.09	2.77

Circuit selection for remote CNOT



Tokyo



Poughkeepsie

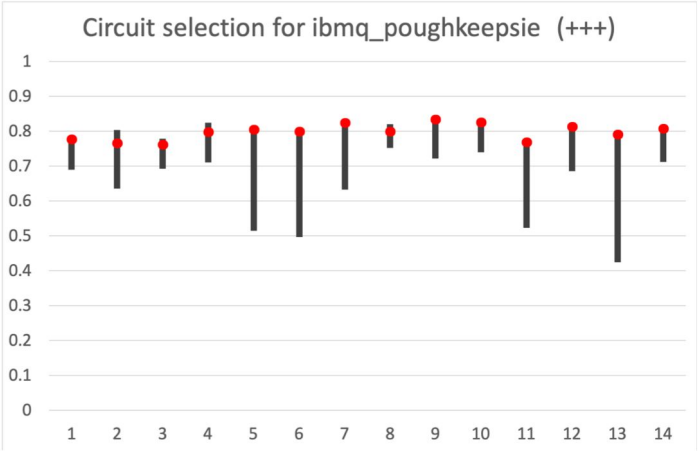
Circuit selection for remote CNOT

Problem 3′: Circuit Selection ($|+++\rangle$)

Perform the same circuit as Problem 3 but with different initial input states on Poughkeepsie. Is there any change in the reliability of circuit selection by ESP?

Result 3′

ESP was able to select the optimum circuit from among seven candidates 36% of the time. Fig.16 shows the fidelity of selected circuit relative to the other candidate circuits.



Compiling complete programs

“A general purpose compiler, however, is required to combine such realizations in order to deal with much more complex quantum circuits with many CNOT gates.”

“The compiler must be able to distinguish more reliable realizations from the others.”

$$ESP(C) = \prod_{g \in C} ESP(g) = \prod_{g \in C} (1 - \varepsilon_g)$$

Search space

The search space of this problem can be visualized as a collection of hypercubes. Each hypercube consists of a complete set of execution states, and there is one corresponding hypercube for each qubit mapping. There are two types of transitions between the states: gate execution and swap insertion.

The optimization algorithm

Algorithm 1 Compilation Algorithm

```
1:  function COMPILE(gates, topology)
2:      N ← |gates|
3:      S_0 ← initialize states with INITIAL_MAPPING(gates, topology)
4:      for i = 0 upto N (exclusive) do
5:          S_{i+1} ← {}
6:          for all state ∈ S_i do
7:              for all g ∈ gates do
8:                  if dependency of g is satisfied then
9:                      qubits ← state.mapping[g.qubits]
10:                     swap ← BEST_SWAP(qubits, topology)
11:                     state' ← state with swap inserted and g executed
12:                     state'.esp ← state.esp × ESP(swap) × ESP(g)
13:                     UPDATE_SCORE(state', gates, topology)
14:                     S_{i+1} ← S_{i+1} ∪ {state'}
15:                 end if
16:             end for
17:         end for
18:         S_{i+1} ← top-B states of S_{i+1}
19:     end for
20:     return the best state ∈ S_N
21: end function
```

Beam-search based heuristic optimization algorithm.

Time Complexity:

$$O(N^2M + N^3B)$$

Subroutines for compilation algorithm

Algorithm 3 Greatest Connecting Edge Mapping

function INITIAL_MAPPING($gates$,$topology$)
  extract guest graph $G_V = (V_V, E_V)$ from $gates$
  $Mapped = \{\}$
  $(v_1, v_2) \leftarrow$ greatest edge $\in E_V$
  $(q_1, q_2) \leftarrow$ lowest edge $\in topology$
  $M \leftarrow \{v_1 \mapsto q_1, v_2 \mapsto q_2\}$
  $Mapped \leftarrow Mapped \cup \{v_1, v_2\}$
  $E_V \leftarrow E_V \setminus \{(v_1, v_2)\}$
  while not all endpoints in E_V are mapped **do**
    pick the greatest edge $(v, v') \in E_V$ where $v \in S$ and $v' \notin S$
    if there is a free physical qubit adjacent to $M[v]$ **then**
      pick the lowest edge $(q, q') \in topology$ where $q = M[v]$ and q' is a free qubit
      $M \leftarrow M \cup \{v' \mapsto q'\}$
      $S \leftarrow S \cup \{v'\}$
    end if
    $E_V \leftarrow E_V \setminus \{(v, v')\}$
  end while
  for all $v \in V_V \setminus Mapped$ **do**
    pick a random free physical qubit $q \in topology$
    $M \leftarrow M \cup \{v \mapsto q\}$
  end for
  return M
end function

Algorithm 4 Find The Best Swap Sequence

function BEST_SWAP($qubits$,$topology$)
  if $qubits.len() == 1$ **then**
    return []
  end if
  $ESP_{max} \leftarrow 0$
  $swap_{max} \leftarrow []$
  for all $(q_0, q_1) \in topology$ **do**
    $swap_0 \leftarrow$ find the swap sequence between $qubits[0]$ and q_0 with greatest $ESP(swap_0)$
    $swap_1 \leftarrow$ find the swap sequence between $qubits[1]$ and q_1 with greatest $ESP(swap_1)$
    $ESP \leftarrow ESP(swap_0) \times ESP(swap_1) \times ESP(\text{CNOT over } q_0 \text{ and } q_1)$
    if $ESP > ESP_{max}$ **then**
      $ESP_{max} \leftarrow ESP$
      $swap_{max} \leftarrow swap_0 + swap_1$
    end if
  end for
  return $swap_{max}$
end function

Algorithm 2 Update Score of a State

function UPDATE_SCORE($state$,$gates$,$topology$)
  $score \leftarrow state.esp$
  for all $g \in gates$ **do**
    if g has not been executed **then**
      $swap \leftarrow$ BEST_SWAP($state.mapping$,g,$topology$)
      $score \leftarrow score \times ESP(swap) \times ESP(g)$
    end if
  end for
  $state'.score \leftarrow score$
end function

Experimental evaluation of compilation

"We cannot decide whether the circuit succeeded or failed from one shot of measurement. Instead, we have to focus on how close the empirical probability distribution sampled by running the compiled circuit on a NISQ machine multiple times and the ideal distribution an imaginary noiseless quantum computer will produce are."

Define KL divergence:

$$D_{KL}(P_{ideal}||P_{empirical}) = \sum_{x} P_{ideal}(x) \log \frac{P_{ideal}(x)}{P_{empirical}(x)}$$

The lower KL divergence is, the better the quality of the compiled circuit is.

Experimental evaluation of compilation

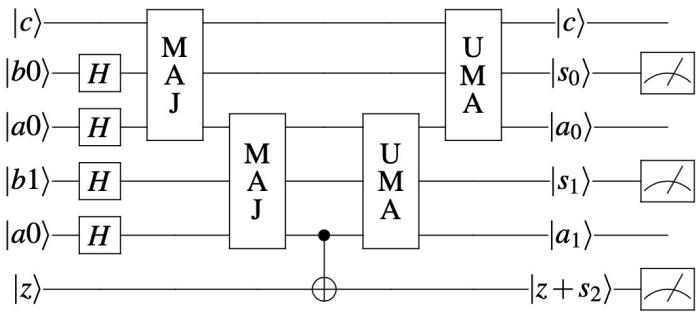
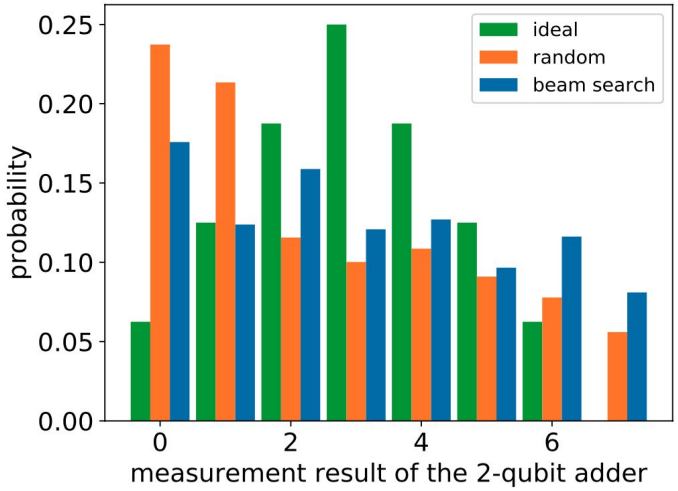
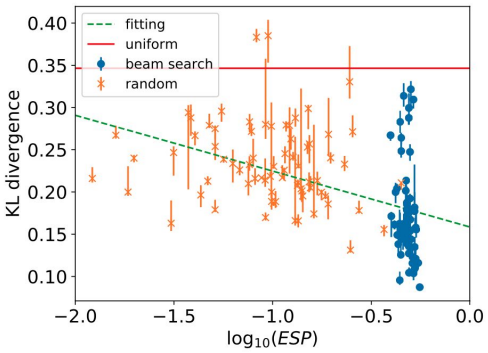


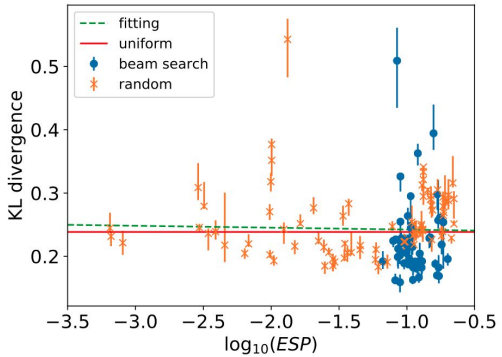
Figure 17: Cucarro's adder circuit of 2-qubit inputs and carry-in ($|c\rangle$) and carry-out ($|z\rangle$). MAJ (MAJority) gate consists of one Toffoli gate and two CNOT gates and computes carry bit by determining the majority of inputs are $|1\rangle$ state. UMA (UnMajority and Add) is also made of one Toffoli and two CNOTs and computes the addition of this digit on $|b\rangle$, uncomputing the other qubits. We denote the i'th bit of the addition as $|s_i\rangle$ in the figure. Toffoli gates are made of 6 CNOT gates and 9 single qubit gates. Since we give $|0\rangle$ for $|c\rangle$ and $|z\rangle$, the measurement of this circuit gives you the distribution of the added value.



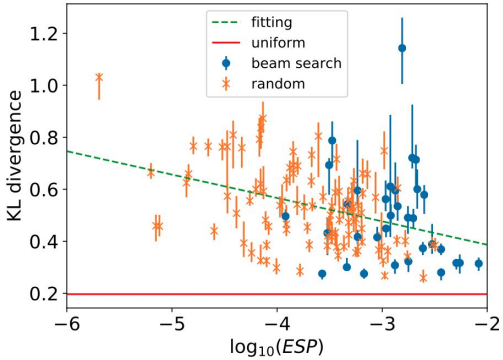
Experimental evaluation of compilation



(a) 1-qubit adder



(b) 2-qubit adder



(c) 4-qubit adder

Linear regression:
1-qubit adder = -0.475
2-qubit adder = -0.0279
4-qubit adder, negative correlation, but absolute value of KL divergence is much worse.

Impacts

Proposed two reliability metrics for quantum gates and circuits: Estimated Success Probability for use during compilation and KL divergence for assessing results.

Designed and implemented an error-aware quantum compiling system.

Could contribute to future quantum architectural design.

WHO should care & WHY

Quantum computer designers
Quantum algorithm developers
Quantum computing scientists

This paper exhibited a working error-aware compilation, which is crucial for future quantum computer design. The paradigm described in the paper can be incorporated directly into standard release of current quantum systems.

Is it believable?

Yes.

ACM Journal

24 citations for pre-print version(since May.2019)

4 citations for published version(since May.2020)

Limitations

1. Even in relatively simple case of selecting a two-hop path, the best success rate is only 70%.

2. Relation between ESP and KL divergence vanished on complex circuit.

3. Not considered memory errors.

Questions?

Thank you