Debugging Quantum Circuits with Breakpoints

Palvit Garg(pgarg5), Harshwardhan Joshi(hjoshi2), Shawn Salekin (ssaleki)

Goal

The goal of the project is to implement debugging with breakpoints on quantum circuits that run on actual hardware (IBM Quantum), with the help of Qiskit API.

Previous Work

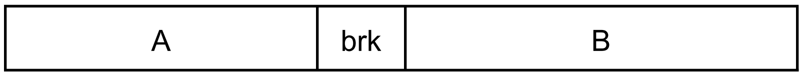
There was no breakpoint debugging implemented on real hardware – although in the previous years some work was done on simulated quantum circuits.

Problem Description

In classical computing, debugging with breakpoint means halting the program execution at any given point and freezing the program state. This allows programmers to examine the program internals mid-execution (as in, what variables contain which values, analyze control flow etc.) Unfortunately, in quantum computing, doing this is not possible. Qubits in a quantum circuit have probabilistic values (in superposition), and the act of measuring collapses the qubits to a deterministic state.

Breakpoint debugging is still needed for quantum computing, so our problem boils down to measuring at the breakpoint and preserving enough information to “recreate” that state and continue execution from that point.

Let’s assume we have a circuit with multiple qubits, and it has a breakpoint **brk**. The circuit before the breakpoint is called **A**, and after is called **B**. The target is to measure the state of qubits at point brk.

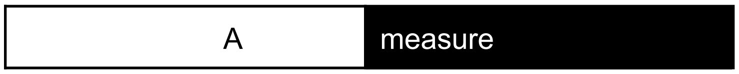


To “recreate”, i.e., synthesize an equivalent circuit right after it is measured, we thought of two possible methods.

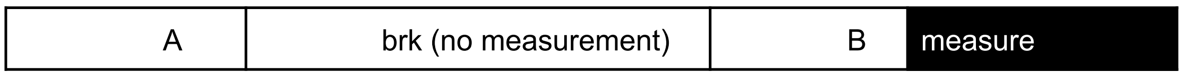
Approach 1

We make two circuits of varying length: one ends at the breakpoint, and the other stops at the original end without stopping at the breakpoint. Then we run both of these circuits.

Run 1:

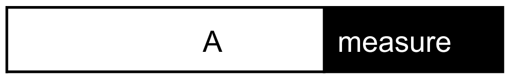


Run 2:



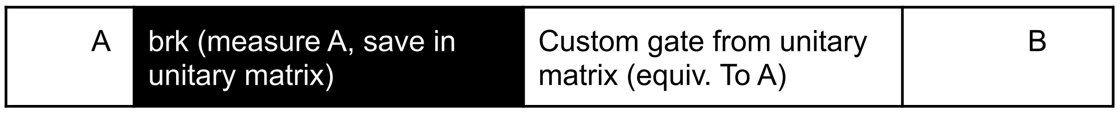
Approach 2
Create a custom gate using the unitary matrix that results from running circuit part A, save it somewhere. Then create a custom gate using the saved unitary matrix to append to circuit part B. This custom gate is equivalent to the circuit part A. Now run this custom gate and then part B.

Run 1:



Save the state of the circuit after this measurement in a Unitary Matrix.

Run 2:


A
brk (measure A, save in unitary matrix)
Custom gate from unitary matrix (equiv. To A)
B

Analysis of Approach 1

Rerun the circuit A without developer worrying about any extra logic will help them debug the circuit, but here the cost of running the circuit twice will be higher.

To implement this approach, we can make use of the `circuit_to_instruction` feature of the Qiskit converter library. We will be maintaining a queue of `circuit_to_instruction` objects between each breakpoint. This will help us reuse each component multiple times. i.e. First time, we will just run part A and measure. Then next time we will append A and B together and then measure finally. This approach can be extended to any number of breakpoints and qubits.

Analysis of Approach 2

Qiskit library lets us create a custom gate if we provide it with a unitary matrix. Thus, in step 1 our target is to run part A and get the unitary matrix from that. In the next step, we will use this unitary matrix to create a custom gate. In the next step, we will append the custom gate with part B. We will also be storing each section between any 2 breakpoints in a queue, so that we can append any component as per requirement.

Our understanding is since the unitary matrix will act as an equivalent of part A, it should take care of the entanglement(s). However, one drawback is that the unitary matrix can not preserve the phases of the state we measured.

Timeline

Note: this schedule is subject to changes depending on the final date release and project progression over the course of next month

Date	**Milestone**
11/01/2022	Implement approach 1
11/10/2022	Analyze feasibility of approach 2 implementation
11/26/2022	Finish implementation (approach 2 or if we find a better one)

URL

Github page: https://pages.github.ncsu.edu/ssaleki/qc_besteffort_breakpoint_debugging/

References

1. https://qiskit.org/documentation/tutorials/circuits_advanced/02_operators_overview.html
2. https://qiskit.org/documentation/stubs/qiskit.converters.circuit_to_instruction.html
3. https://github.ncsu.edu/fmuelle/qc19/tree/master/hw/hw6/m3/csc591-quantum-debugging
4. https://quantumcomputing.stackexchange.com/questions/27064/how-to-get-statevector-of-qubits-after-running-quantum-circuits-on-ibmq-real-har