CS5863: Introuction to Program Analysis and Optimization

Dynamic Quantum Network Optimization

Kartheek Tammana, Kushagra Gupta, Rishit D

Indian Institute of Technology, Hyderabad

Table of contents

Problem Statement

Introduction

- Quantum programs serve as a gateway to realize quantum algorithms alongside classical interpretations. In most programs, there is considerable interleaving of classical and quantum operations which are highly dependent on each other.

Introduction

- Quantum programs serve as a gateway to realize quantum algorithms alongside classical interpretations. In most programs, there is considerable interleaving of classical and quantum operations which are highly dependent on each other.
- In most cases, the flow of quantum programs is as follows:
  1. Fetch quantum circuit from program and optimize appropriately.
  2. Load the circuit on the quantum device and execute it.
  3. Fetch measurement results from the device and run classical post-processing.
  4. Update the quantum circuit with the classical results and repeat.

  These models are especially useful for variational quantum algorithms (eg, to realize quantum neural networks) and error-correcting codes.

Motivation

- Although deferred measurement is a well-known technique, it fails when the succeeding circuit involves re-evaluation of measured qubits. In mathematical terms, the combined circuit is *not reversible*.

Motivation

- Although deferred measurement is a well-known technique, it fails when the succeeding circuit involves re-evaluation of measured qubits. In mathematical terms, the combined circuit is *not reversible*.
- Most transpilers including `Qiskit` [1] and `cuda-quantum` [2] fail to recognize common patterns across branches and completely ignore optimization *across* measurements.

Motivation

- Although deferred measurement is a well-known technique, it fails when the succeeding circuit involves re-evaluation of measured qubits. In mathematical terms, the combined circuit is *not reversible*.
- Most transpilers including `Qiskit` [1] and `cuda-quantum` [2] fail to recognize common patterns across branches and completely ignore optimization *across* measurements.
- Assuming we know the branch values, simple unrolling on the aforementioned transpilers greatly improves gate count and circuit size. This shows that there is a lot of scope for improvement in existing transpilers, although only a small fraction of this gap may be closed.

Problem Statement

- Given a quantum circuit with mid-circuit measurements, we aim to optimize the circuit by integrating existing classical and quantum techniques including deferred measurement, branch expansion, constant propagation, constant folding and a novel pass for *measurements serving a single-branch*.

Problem Statement

- Given a quantum circuit with mid-circuit measurements, we aim to optimize the circuit by integrating existing classical and quantum techniques including deferred measurement, branch expansion, constant propagation, constant folding and a novel pass for *measurements serving a single-branch*.
- We aim to investigate the effect of these optimizations and the time taken to perform these optimizations, ie, the tradeoff between optimization time and circuit size.

Problem Statement

- Given a quantum circuit with mid-circuit measurements, we aim to optimize the circuit by integrating existing classical and quantum techniques including deferred measurement, branch expansion, constant propagation, constant folding and a novel pass for *measurements serving a single-branch*.
- We aim to investigate the effect of these optimizations and the time taken to perform these optimizations, ie, the tradeoff between optimization time and circuit size.
- We would also like to look at alternate circuit formulations, specifically *probabilistic quantum circuits* [3] which forgo correctness in favor of removing measurements.

Problem Statement

- Given a quantum circuit with mid-circuit measurements, we aim to optimize the circuit by integrating existing classical and quantum techniques including deferred measurement, branch expansion, constant propagation, constant folding and a novel pass for *measurements serving a single-branch*.
- We aim to investigate the effect of these optimizations and the time taken to perform these optimizations, ie, the tradeoff between optimization time and circuit size.
- We would also like to look at alternate circuit formulations, specifically *probabilistic quantum circuits* [3] which forgo correctness in favor of removing measurements.
- We chiefly use the `Qiskit` transpiler as a baseline and use their API to introduce our optimizations and perform analysis.

Work Done

Work Done

- We have discussed the basic of quantum computing and optimizations in such circuits.

Work Done

- We have discussed the basic of quantum computing and optimizations in such circuits.
- We have gone through a few quantum compilers, chiefly `Qiskit` and `cuda-quantum` and the optimization passes they perform. As mentioned before, there is limited scope for mid-circuit measurement optimizations in these compilers.

Work Done

- We have discussed the basic of quantum computing and optimizations in such circuits.
- We have gone through a few quantum compilers, chiefly `Qiskit` and `cuda-quantum` and the optimization passes they perform. As mentioned before, there is limited scope for mid-circuit measurement optimizations in these compilers.
- We have gone through the following techniques that we will be using:
    1. *Constant Folding:* As per Hoare optimizatioms we can fold constants as well propagate entanglement across registers through *entanglement assertions* and *triviality conditions*. Although, this has been seen in `Qiskit`, we attempt use constant folding across branches and use measurements to create new constants. [4]
    2. *Branch Expansion*: We can expand each branch of the circuit upto a certain depth and internally optimize each branch. We will attempt to use this conjunction with our novel pass. [5]

Novel Pass

The key idea is to assume a branch to fail and combine the circuit before branch and after branch specifically for registers that do not involve the measured qubit. We broadly divide our pass into two phases with our initial configuration of the following form.

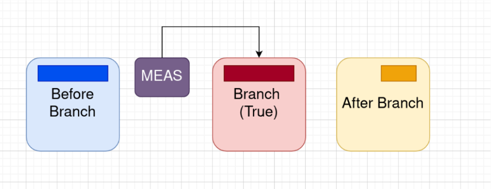

Figure 1: Initial configuration of the circuit

Novel Pass - Split

- We first split the *after-branch* subcircuit into parts that are dependent on the measured qubit and those that are not. In other words, the subcircuit *after-branch-1* which is not dependent commutes with the measurement.

Novel Pass - Split

- We first split the *after-branch* subcircuit into parts that are dependent on the measured qubit and those that are not. In other words, the subcircuit *after-branch-1* which is not dependent commutes with the measurement.
- We also select a certain depth of the *before-branch* subcircuit to combined later. This depth is selected heurstically.

Novel Pass - Split

- We first split the *after-branch* subcircuit into parts that are dependent on the measured qubit and those that are not. In other words, the subcircuit *after-branch-1* which is not dependent commutes with the measurement.
- We also select a certain depth of the *before-branch* subcircuit to combined later. This depth is selected heurstically.

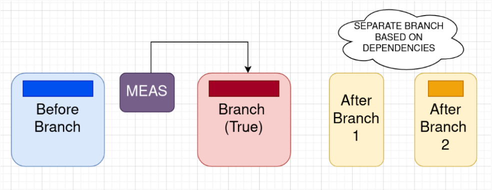

Figure 2: Spliting Phase

Novel Pass - Recombine

- We assume the branch to fail and combine *before-branch* with *after-branch-1*. Note that the measurement result is unchanged and the circuit is still reversible.

Novel Pass - Recombine

- We assume the branch to fail and combine *before-branch* with *after-branch-1*. Note that the measurement result is unchanged and the circuit is still reversible.
- If our assumption is false, we simply reverse *after-branch-1* and prefix it to the true branch, followed by the remaining *after-branch* subcircuit.

Novel Pass - Recombine

- We assume the branch to fail and combine *before-branch* with *after-branch-1*. Note that the measurement result is unchanged and the circuit is still reversible.
- If our assumption is false, we simply reverse *after-branch-1* and prefix it to the true branch, followed by the remaining *after-branch* subcircuit.
- Each of the combinations is optimized to the highest degree to reduce the circuit size. In most worst-cases, we recover the original circuit size but do better on average.

Novel Pass - Recombine

- We assume the branch to fail and combine *before-branch* with *after-branch-1*. Note that the measurement result is unchanged and the circuit is still reversible.
- If our assumption is false, we simply reverse *after-branch-1* and prefix it to the true branch, followed by the remaining *after-branch* subcircuit.
- Each of the combinations is optimized to the highest degree to reduce the circuit size. In most worst-cases, we recover the original circuit size but do better on average.
- We can make use of constant-folding to propagate set qubits and cancel gates across the measurement if necessary.

Novel Pass - Recombine

- Note that false-branches can also be used but only if it commutes with the measurement. This is a special that may be checked at compile time (although it seems to be expensive).

Novel Pass - Recombine

- Note that false-branches can also be used but only if it commutes with the measurement. This is a special that may be checked at compile time (although it seems to be expensive).
- Further recursion is possible with the above assumption but the tradeoff weighs negatively after every recursion.

Novel Pass - Recombine

- Note that false-branches can also be used but only if it commutes with the measurement. This is a special that may be checked at compile time (although it seems to be expensive).
- Further recursion is possible with the above assumption but the tradeoff weighs negatively after every recursion.

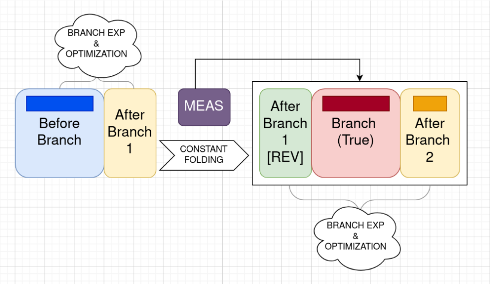

Figure 3: Recombination Phase

Deliverables & Timeline

Deliverables

- (MVP) An investigation on our proposed pass based on the `Qiskit` transpiler on certain toy examples referenced in bibliography.
- (MVP) A simple implementation of the proposed pass on `Qiskit` transpiler involving *Branch Expansion* and our novel pass.
- A MLIR-driven optimization pass for `cuda-quantum` which will be used to all aforementioned optimizations.

Timeline & Contributions

- *Week 1*:
  - Kartheek: Literature reading and investigation/verification of optimizations
  - Kushagra: Qiskit transpiler overview and sample passes
  - Rishit: Integrated pseudocode for proposed pass, MLIR pass overview
- *Week 2*:
  - Kartheek & Kushagra: Implement Qiskit pass & experiments
  - Rishit: Implement MLIR pass

References i

Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta.
Quantum computing with Qiskit, 2024.

The CUDA-Q development team.
CUDA-Q.

Yanbin Chen, Innocenzo Fulginiti, and Christian B. Mendl.
Reducing mid-circuit measurements via probabilistic circuits.
In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, page 952–958. IEEE, September 2024.

- Thomas Häner, Torsten Hoefler, and Matthias Troyer.
  Assertion-based optimization of quantum programs.
  Proceedings of the ACM on Programming Languages, 4(OOPSLA):1–20, November 2020.
- Yanbin Chen.
  Unleashing optimizations in dynamic circuits through branch expansion, 2025.