Qiskit Advocate Mentorship Program

Timeline debugger for the Qiskit transpiler

Mentor: Kevin Krsulich

Mentess: Aboulkhair Foda , Harshit Gupta

Progress

Developed two working prototypes for the transpiler timeline debugger

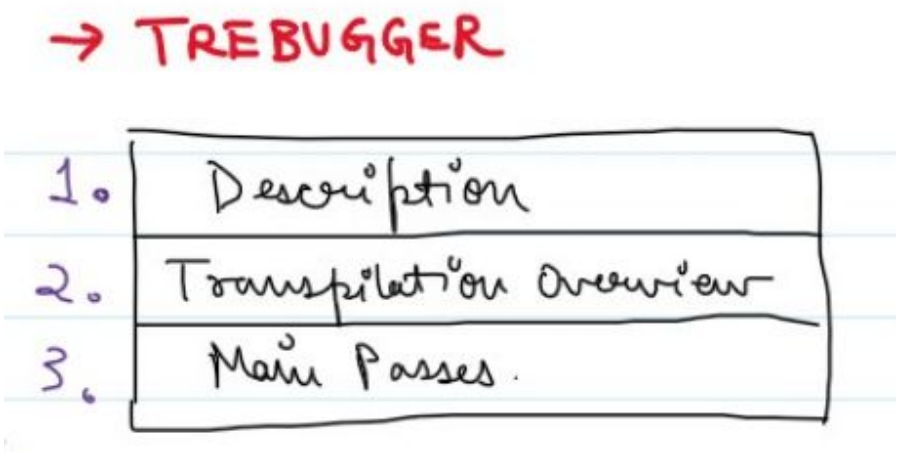
Enables users to see how and what changes are happening while transpiling a single quantum circuit

- Incorporates circuit images in debugger as circuit goes through the transpilation process
- Allows user to analyse **visual diffs** of back to back transpilation passes
- Parses **Logs** emitted by each transpiler pass to see a greater level of detail during execution
- Highlights changes in circuit stats and property set for each pass

Currently working on making the debugger efficient in terms of memory and CPU utilization

Finalised Design

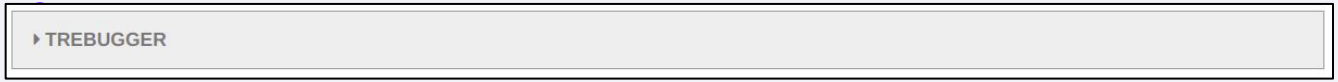
- Our final debugger consolidates both prototypes into one. It is being developed as a jupyter widget and consists of three main components :
  - **Description** of the circuit, backend and transpiler arguments provided during transpilation
  - **Transpilation Overview** for the final circuit which provides a quick summary of the complete process
  - **Main Passes** of the transpiler, where each component contains a description of circuit state (depth < 100), properties, logs of pass and documentation.



Color Scheme of project is inspired by that of Qiskit

1. Description Panel

- **Collapsed View**



- **Expanded View**

- The description panel is developed for a quick glance at the circuit transpilation options

- It serves as a point for the user to see what are the parameters involved in transpilation of their circuit

2. Transpilation Overview

- **Panel View**

Transpilation Overview

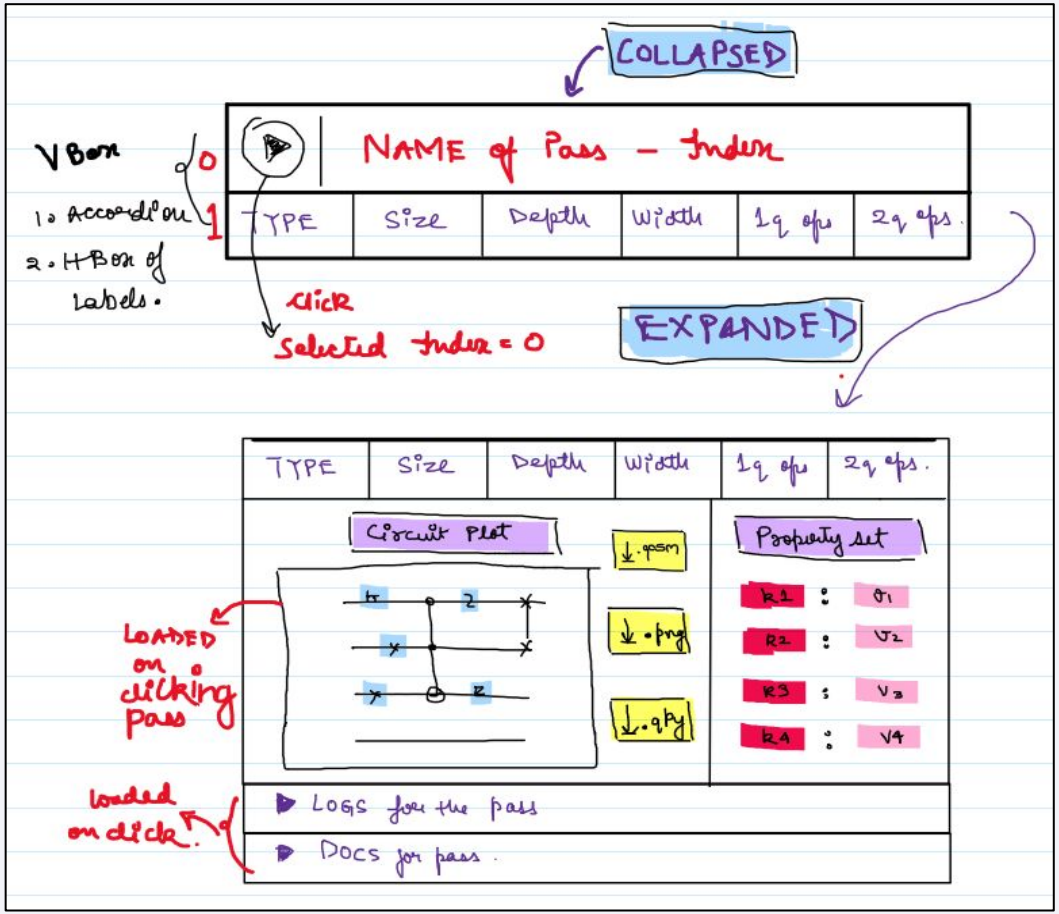
Transformation Passes	Analysis Passes
9	4
Initial depth	Initial Op Count
6	15
Final depth	Final Op Count
81	100

- This panel allows users to see a brief summary of the transpilation process

- Depth and Op count are the two main parameters chosen for overview panel

- Subsequent Transformation and Analysis passes are color coded in the debugger according to the **color scheme** presented

3. Main Passes - Work in Progress



- Each component of main pass panel contains information about each transpiler pass

- We aim to provide two levels of debugging - basic and advanced

- For the basic debugger, pass component would only contain the name of the pass and necessary properties

- The advanced debugger would contain four sub components highlighting changes being made

Circuit Properties

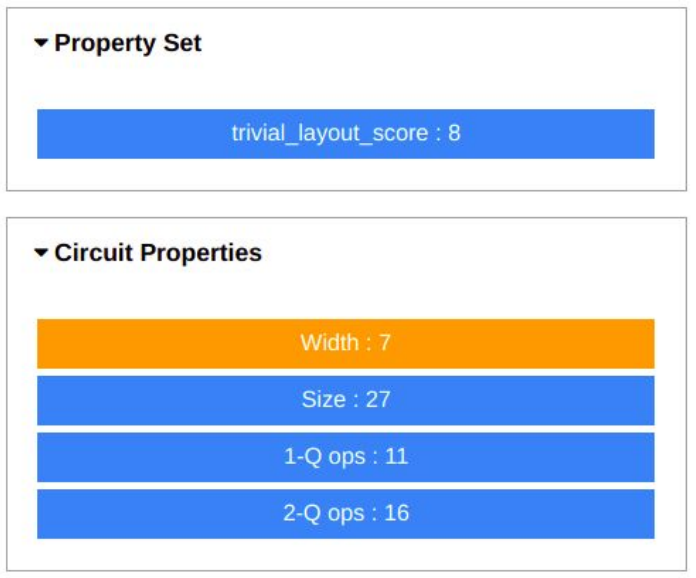
Concept

- There exist two collapsible headings in which highlight the **property set** and the **circuit properties** after each pass

- The changes in properties are highlighted in the code by changing the color of the property value which has been changed or added to the set

- Properties having very large values are truncated to the basic information like commutation set

- This allows users to find which passes are responsible for major changes in the circuits and their property sets



Circuit State

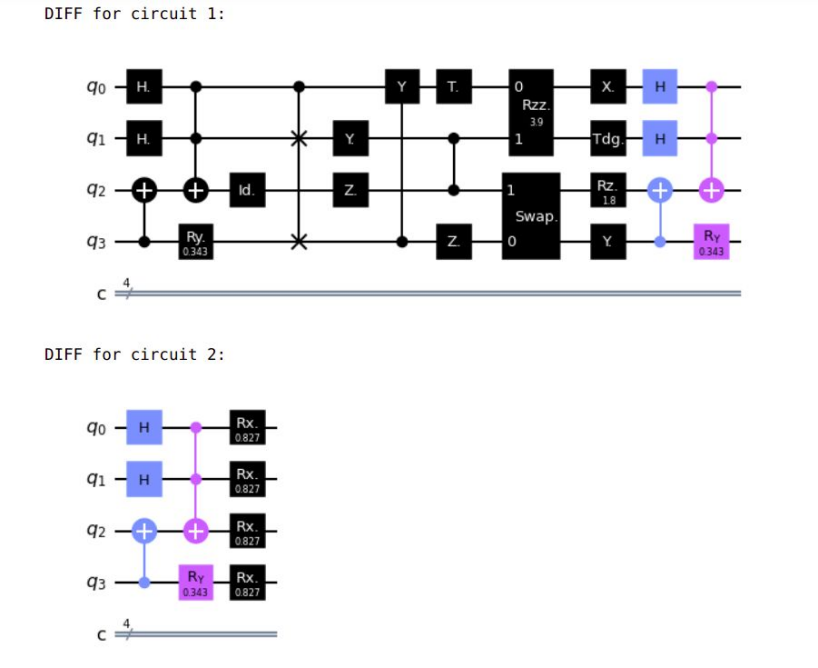
Concept

- Qiskit visualization provides the draw('mpl') method for circuit visualization but can not **visually differentiate** between two circuits using it

- Our debugger uses the DAG representation of circuit and finds the **Longest Common Subsequence** between two successive DAGs

- This enables us to define **visual diffs** for two circuits of consecutive passes and thus helps to analyse what changed in our circuit and how

- Users also have the opportunity to download the state and further inspect the circuit diff



Transpiler Pass Logs

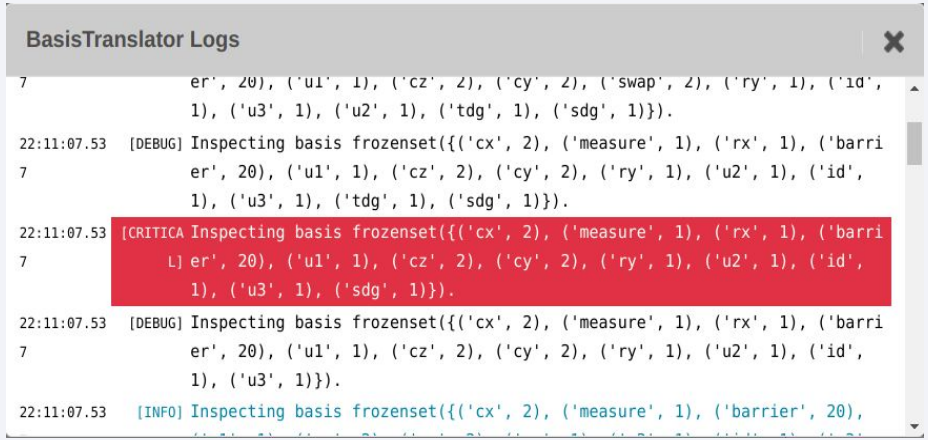
Concept

- The qiskit transpiler associates **Log** entries which are emitted during the execution of each transpiler pass

- There exists a collapsible header containing all the relevant entries of the transpiler pass

- Each log entry is color coded according to the level of severity of the log i.e. INFO, WARNING, etc.

- This enables user to **see what decisions** transpiler is making while executing a particular pass



Transpiler Pass Docs

Concept

- To let users know the motif of each pass, the documentation block is provided as a collapsible header

- Seeing the documentation block allows a user to **understand the goal** of a particular transpiler pass

- Each doc string is formatted accordingly to highlight the explanation, arguments and return values of each pass, if any

ConsolidateBlocks

Replace each block of consecutive gates by a single Unitary node.

Pass to consolidate sequences of uninterrupted gates acting on the same qubits into a Unitary node, to be resynthesized later, to a potentially more optimal subcircuit.

Notes:

This pass assumes that the 'blocks_list' property that it reads is given such that blocks are in topological order. The blocks are collected by a previous pass, such as `Collect2qBlocks`.

```
ConsolidateBlocks.run(dag)
```

Run the ConsolidateBlocks pass on `dag`.

Iterate over each block and replace it with an equivalent Unitary on the same wires.

Please follow the updates of our project at :

https://github.com/kdk/qiskit-timeline-debugger

Thank you!