透過視覺化可互動式程式碼映射工具展示量子程式預處理最佳化

Illuminating Quantum Profile-Guided Optimization via Interactive Source-Level

指導教授：涂嘉恆
專題成員：吳昱輝
開發工具：PyQt、SVGWrite
測試環境：Ubuntu 20.04 LTS

一、簡介：

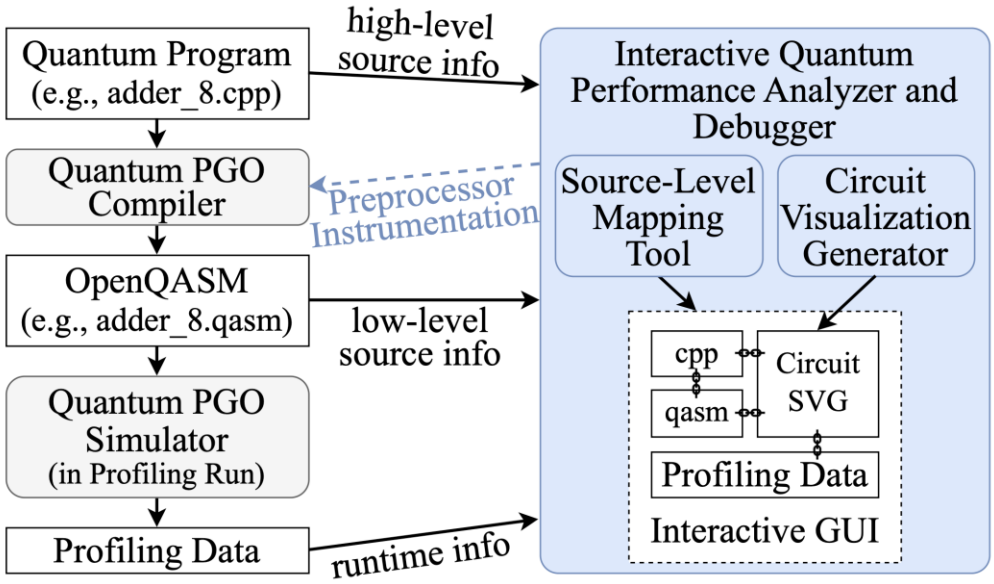
在本專題中使用 QCOR 作為量子程式的編譯器。編譯器的輸入是高階(high-level) 的程式語言，QCOR 是以 C++ 作為輸入，轉換過後輸出低階(low-level)的程式碼，目前實作是以 OpenQasm 作為輸出。當編譯器將高階語言轉換為低階程式碼時， 其中組合量子閘(composite gate) 會被解析為一組由硬體支援的量子閘。當開發人員想要透過量子模擬器來分析程式執行時間的時候，量子模擬器提供的時間分析是基於低階語言的量子閘，而開發人員面對的是高階的程式碼，所以開發人員分析時會受到編譯器的影響。所以我們開發了量子程式的分析和偵錯工具，讓使用者能以更有效率的方式分析程式。此工具分為兩個部分。

第一部分是 Source-level mapping tool，透過改進 QCOR 編譯器行為，在每次編譯器解析每一個 composite gate 時，在其低階語言的量子閘序列的結尾加入一個我們新增的量子閘，稱為 “comment” gate。透過 comment gate 就可以在連續的量子閘序列中判斷出對應的區塊。

第二部份是 Circuit visualization generation。當量子程式執行在量子模擬器上，會產生關於此程式的 profiling data。使用 SVGWrite 套件畫出量子閘執行的量子線路圖 (quantum circuit)，再透過 profiling data 內的時間資料標註每個量子閘在模擬器上的執行時間，提供效率分析用途。

最後使用 PyQt 視窗程式設計的框架，將以上兩種工具整合在 GUI 中，並且透過 Qt 提供的 SVG 物件辨識功能，連結量子線路和原始程式碼，提供使用者互動式的介面。

以下為架構圖：



二、測試結果：

左側顯示 C++ 的原始程式碼，右側顯示經編譯後產生的 OpenQasm 程式碼。下方是量子線路圖。透過滑鼠點擊不同區域會以底色標註該段程式碼的對應區塊。