Investigation on Quantum Computers

Zaher Mohammed Faruk

**FINAL YEAR BSC PROJECT REPORT 2023/2024**

This report is presented in partial fulfilment of the requirements for the award of the degree of B.Sc. with Honours in Physics

1. Abstract

This project focuses on comparing and evaluating the capabilities of two leading quantum computing platforms: IBM Quantum, which uses superconducting qubits, and Xanadu’s photonic quantum systems, accessed through the Pennylane programming framework (a Python library) [1]. The primary goal is to investigate their performance across a range of algorithms, focusing on factors such as computational efficiency, accuracy, and error rates.

The study emphasises the contrasting approaches of photonic and superconducting qubit technologies, analysing their effectiveness in terms of scalability, hardware robustness, and algorithmic precision [2]. Additionally, it examines how tools like Pennylane enable seamless programming and simulation, bridging the gap between theoretical quantum concepts and real-world implementation. Through a detailed comparison of outcomes, this work aims to shed light on the current state of quantum computing platforms and their future potential in advancing scientific and industrial applications [3].

1. Acknowledgements

We sincerely thank the following people and organisations for their invaluable support, which were instrumental making this scientific report possible:

- Dr Mark Hughes: His endless support and in-depth knowledge on Quantum Computing assisted me heavily.

- IBM: We are deeply thankful to IBM Quantum for providing access to their state-of-the-art quantum computing platform. Their advanced superconducting qubit technology enabled us to execute and benchmark quantum algorithms effectively. Additionally, we appreciate the extensive documentation and support available through the IBM Quantum platform, which significantly enhanced our understanding and implementation of quantum circuits.

- Xanadu Quantum Technologies: We express our gratitude to Xanadu Quantum Technololgies for granting access to their photonic quantum computing platform and for providing the powerful Pennylane framework. The flexibility and user-friendliness of Pennylane allowed us to efficiently program and simulate quantum systems, while its compatibility with various quantum backends proved invaluable in conducting comparative studies.

1. Introduction

Over the past decade, quantum computing—a field intersecting physics, computer science, mathematics, and engineering—has seen remarkable progress [4]. Diverse quantum technologies lie at the heart of this evolution, each presenting novel solutions to complex computational challenges that classical systems struggle to address. Superconducting qubits and photonic quantum computing stand out as two leading paradigms. IBM Quantum is at the forefront with superconducting qubits, while Xanadu leads in photonic quantum computing. These cutting-edge platforms are expanding the frontiers of computational capability and scalability [2], allowing researchers to delve into previously inaccessible quantum algorithms and applications.

IBM Quantum is notable for its durable superconducting qubit architecture, characterised by high coherence times, accurate gate operations, and an extensive suite of quantum programming tools provided through the Qiskit platform [5]. The IBM Quantum cloud provides access to a wide array of quantum processors, facilitating detailed testing of quantum circuits and enabling users to benchmark performance and investigate novel algorithms [6]. In contrast, Xanadu's photonic quantum systems, available through their Pennylane framework, offer a different approach by leveraging continuous-variable quantum computing [7]. This photonic platform is particularly well-suited for tasks like Gaussian Boson Sampling, which has notable applications in graph theory and machine learning [8].

Our research investigates and compares the performance of IBM Quantum and Xanadu Quantum Technologies’s photonic quantum systems through practical programming and execution of quantum algorithms. By utilising Qiskit for IBM Quantum and Pennylane for Xanadu, we explore their respective strengths and limitations in terms of computational efficiency, scalability, and accuracy. This comparative study highlights the distinct hardware architectures, programming paradigms, and error-correction strategies of each system while assessing their practical applications in solving quantum problems.

This work also delves into the growing utility of software frameworks like Pennylane, which bridge theoretical quantum computing concepts with real-world implementations [9]. Pennylane supports hybrid quantum-classical workflows, enabling the smooth execution of quantum algorithms on both photonic and qubit-based systems, thus making it an essential tool for this research. Likewise, Qiskit’s advanced simulator and hardware interfaces provide rigorous testing of quantum programs before deploying them on physical qubits [10].

Ultimately, this study aims to shed light on the current capabilities of IBM Quantum and Xanadu Quantum Technologies, providing a detailed analysis of their respective advantages and challenges. By comparing their performance across a range of computational tasks, we aim to contribute to the broader understanding of quantum computing technologies, offering insights into their potential roles in scientific and industrial applications.

1. Theory and Computational Approach

**Solving the QUBO Problem with Quantum Computing**

The Quadratic Unconstrained Binary Optimi**s**ation (QUBO) problem is a type of optimi**s**ation problem**.** It involves finding the optimal solution to a quadratic function where the variables are binary, meaning they can only take the values 0 or 1. The goal is to minimise (or maximise) this function without any additional constraints on the variables. QUBO problems are particularly notable because they can represent a wide variety of combinatorial optimisation issues, making them highly versatile and applicable to numerous real-world scenarios. Their importance has surged with the advent of quantum computing, which promises more efficient solutions to these complex problems [11] .

Mathematical Formulation of QUBO

Mathematically, the QUBO problem is expressed as follows:

Minimize f(x) = ∑i ai xi + ∑i < j bij xi xj

Where:

* xi are the binary variables, i,e., xi ∈ {0,1}
* ai are the coefficients for the linear terms
* bij are the coefficients for the quadratic terms

Matrix Representation

The QUBO problem can also be formulated in matrix notation. Let Q be a symmetric matrix where Qii = ai and Qij = bij for i ≠ j. The function then becomes:

f(x) = xTQx

Where:

* x is a vector of binary variables
* Q is a symmetric matrix containing the coefficients [12]

Example of QUBO function:

Consider a simple QUBO problem with three binary variables x1, x2, and x3:

f(x1, x2, x3) = - x1 – x2 – x3 + 2x1x2 + 2x1x3 + 2x2x3

This can be represented in the Matrix form as:

The objective function is:

f(x) = (x1 x2 x3)

**Solving the QUBO Problem with Quantum Computing**

**Xanadu Quantum Technologies**

**Solving the QUBO Problem with a simulation of a Quantum Computer via PennyLane:**

The QUBO Problem was solved using, the “default.qubit” device on PennyLane which is simulator for testing and development purposes. It doesn't directly connect to any specific Xanadu quantum hardware. For actual Xanadu hardware to be used, specifying a device such as "X8" or "Borealis”. These devices are part of Xanadu's photonic quantum computing systems. The QUBO problem was solved via this method, and the code was broken down into various sections each representing a different part of the QUBO function.

The code broken down into it’s various steps:

1) The Hamiltonian

H = 0.5 \* qml.Identity(1) + \

0.5 \* qml.PauliZ(1) @ qml.PauliZ(4) + \

0.5 \* qml.PauliZ(2) @ qml.PauliZ(3) + \

0.5 \* qml.PauliZ(3) @ qml.PauliZ(5) + \

0.5 \* qml.PauliZ(4) @ qml.PauliZ(4)

print(H)

The Hamiltonian (H), is defined as a combination of the tensor products of the Pauli – Z operators and the identity operator:

Here:

* I is the identity operator
* Zi represents the Pauli – Z operator acting on the qubit i
* ZiZj represnts a coupling term between the qubits i and j

This Hamiltonian encodes the QUBO problem. The coefficients of ZiZj represent the interaction weights, and the terms correspond to local biases. These interactions determine the energy landscape of the system, with the goal being to find the configuration of qubits that minimises H, which corresponds to solving the QUBO problem. The eigenvalues of H represent possible energies of the system, and the ground state (lowest energy eigenvalue) corresponds to the optimal solution of the QUBO problem [13].

2) Device Setup

dev = qml.device("default.qubit", wires=H.wires)

The device default.qubit simulates a quantum computer using qubits. The number of qubits is determined by the number of wires in the Hamiltonian H.

3) Defining the Circuit

@qml.qnode(dev)

def circuit(params):

for param, wire in zip(params, H.wires):

qml.RY(param, wires=wire)

return qml.expval(H)

Mathematical Intepretation:

The quantum circuit applies a series of parameterised rotations on each qubit:

Where the following equation is the rotation operator around the Y axis of the Bloch sphere:

After the applying the rotations, the expectation value of H is measured:

The rotations prepare a quantum state that depends on the parameters θ. The expectation value corresponds to the energy of the system for the given parameters. The goal is to minimise by optimising the parameters [14].

For the first equation, ∣ψ⟩ is the quantum state after the circuit has applied transformations, ⨂i is the tensor product applied across all qubits and it signifies that is applied independently to each qubit, is the rotation operator for the i-th qubit which performs a rotation by the angle θi around the Y – axis of the Bloch sphere, ∣0⟩ is the initial state of the qubit, which is the "ground state" in quantum computing. For the second equation RY (θ) is the rotation operator that rotates a qubit by angle θ around the Y – axis [15].

4) Optimisation Loop

params = np.random.rand(len(H.wires))

opt = qml.AdamOptimizer(stepsize=0.5)

epochs = 200

for epoch in range(epochs):

params = opt.step(circuit, params)

The optimisation process adjusts the parameters {} to minimise the expectation value of H:

The Adam oprimiser updates the prameters using gradient based optimisation:

=

Where η is the learning rate (stepsize) [16].

5) Sampling from an Optimised Circuit

dev = qml.device("default.qubit", wires=H.wires, shots=1)

@qml.qnode(dev)

def circuit(params):

for param, wire in zip(params, H.wires):

qml.RY(param, wires=wire)

return qml.sample()

Here, the circuit is modified to sample from the quantum state prepared by the optimised parameters {}. Each sample corresponds to a bitstring that represents a potential solution to the QUBO problem. The sampling process corresponds to a measurement in the computational basis. The probabilities of obtaining each bitstring are determined by the squared amplitudes of the quantum state:

In the context of quantum mechanics, this represents a projective measurement. The sampled bitstring with the lowest energy (as evaluated by H) corresponds to the optimal or near-optimal solution to the QUBO problem.

**Solving the QUBO Problem Using Real Quantum Hardware via PennyLane**

The QUBO problem was solved using the X8 device on PennyLane, which is part of Xanadu's photonic quantum computing systems. Unlike the default.qubit device, which is a simulator for testing and development purposes, the X8 device utilises real quantum hardware. This device leverages photonic qubits to perform quantum computations. The QUBO problem was addressed via this method, and the code was broken down into various sections, each representing a different part of the QUBO function. By utilising the X8 hardware, we achieve quantum computational benefits that are not possible with classical computers, demonstrating the potential of photonic quantum computing in solving complex optimisation problems [17].

**A comprehensive analysis of the mathematical framework underlying the X8 CPU architecture:**

The X8 architecture leverages advanced photonic integrated circuits (PICs) to manipulate and control light at the quantum level. These PICs are fabricated using silicon nitride (Si3N4) waveguides, which offer low propagation loss and high nonlinearities, essential for efficient quantum operations. The device integrates multiple components such as beam splitters, phase shifters, and squeezing operations to perform complex quantum computations.

At the core of the X8 system is the use of squeezed states of light, represented mathematically as:

Where S(r) is the squeezing operator defined by:

Here, r is the squeezing parameter, a is the annihilation operator, is the creation operator. These squeezed states have reduced noise in one quadrature, and increased noise in the conjugate quadrature, which enhances the sensitivity of quantum measurements. The architecture employs temporal division multiplexing (TDM) to increase the number of modes, represented as n, and enhance the computational power without increasing the physical footprint of the device. The quantum state evolution is described by the unitary operator U:

Where are the phase parameters, and are the Hamiltonians of individual components such as beam splitters and phase shifters. The X8 device operates at cryogenic temperatures (approximately 0.1 K) to minimise thermal noise and maximise coherence times. Superconducting nanowire single-photon detectors (SNSPDs) are used to detect single photons with high efficiency and low dark counts, ensuring accurate readout of quantum states.

The X8 architecture by leveraging these advanced technologies and mathematical principles, achieves quantum computational benefits that are not possible with classical computers, demonstrating the potential of photonic quantum computing in solving complex optimisation problems [18].

**An indepth look into the hardware behind the X8 CPU:**

A diagram of a programmable transformation

AI-generated content may be incorrect.

Figure 1: Quantum Circuit showcasing Xanadu's X8 Chip

This features eight quantum modes, numbered 0 to 7, which are organised into four pairs: (0,4), (1,5), (2,6), and (3,7). Each pair undergoes a process known as two-mode squeezing, facilitated by an S2 gate, which generates entangled photon pairs. Initially, the quantum modes—comprising signal photons (with a wavelength of 1549.3 nm) and idler photons (1536.6 nm)—begin in a vacuum state. The S2 gate then manipulates these states into a two-mode squeezed vacuum (TMSV) state, with the squeezing parameter r determining the outcome. If 𝑟 is 0, the state remains a vacuum; if 𝑟 is 1, a TMSV state is formed. To maintain phase coherence, the phase parameter 𝜙 is set to 0.

Following this, a 4×4 unitary transformation is carried out using an interferometer built from Mach-Zehnder (MZ) interferometers and rotation gates, which operate on both signal and idler modes. Lastly, photon-number resolving detectors measure the number of photons present in each quantum mode.

The eight modes intialised in the vaccum state, undergo squeezing with parameters r. These modes are then entangled using a fixed two-mode U(2) transformation, which acts like a 50/50 beam splitter with a set input phase, resulting in two-mode squeezing at the output. Next, programmable four-mode rotation gates (SU(4) transformation) are applied to each four-mode subspace, represented by the U4 boxes in the diagram. Finally, each of the eight modes is measured individually in the Fock basis [19].

**Strawberry Fields and PennyLane**

PennyLane is a software library for quantum computing, quantum machine learning, and quantum chemistry. It provides an interface to a range of quantum hardware and simulators, making it easier to implement variational quantum algorithms by combining quantum circuits with classical machine learning techniques. A key feature of PennyLane is its use of quantum gradients, allowing users to optimise quantum circuits with classical gradient-based methods. The library supports multiple quantum backends, including IBM Q, Rigetti Forest, and Xanadu’s Strawberry Fields, enabling seamless integration with different quantum platforms. Built on Python, PennyLane is easy to use and works well with other scientific computing and machine learning libraries [9].

Strawberry Fields, developed by Xanadu, is a specialised software platform for continuous-variable quantum computing using photonic systems. It provides tools for simulating and executing quantum circuits that involve Gaussian and non-Gaussian states and operations. Rooted in quantum optics, it leverages photonic qubits, where information is encoded in the quantum states of light, such as squeezed and coherent states. Strawberry Fields allows users to design, optimise, and run quantum circuits on both simulated and real photonic quantum hardware. With a Python library and built-in support for PennyLane, it facilitates the development of hybrid quantum-classical algorithms, making it a valuable tool for research in quantum machine learning and quantum chemistry [20].

References

[1] IBM Quantum. 2024. IBM Quantum. [accessed 2024 Nov 27]. <https://quantum.ibm.com/functions>.

[2] Xanadu | Research. (2024). Xanadu.ai. <https://www.xanadu.ai/research>.

[3] Quantum circuits. (2024). Pennylane.ai. <https://docs.pennylane.ai/en/stable/introduction/circuits.html>.

[4] IBM Quantum Experience - Dashboard. (n.d.). IBM Quantum Experience. <https://quantum-computing.ibm.com/>.

[5] Welcome to Qiskit’s documentation! — Qiskit 0.7 documentation. 2019. Qiskitorg. <https://qiskit.org/documentation/>.

[6] Quantum Computing. 2021 Feb 9. IBM Research. [accessed 2024 Nov 27]. <https://research.ibm.com/quantum>.

[7] Aqeeb I, Arka, Kumar B, Kim R, Kim H, Rao Doppa J, Pande P. 2020. HeM3D: Heterogeneous Manycore Architecture Based on Monolithic 3D Vertical Integration. ACM Transactions on Design Automation of Electronic Systems. doi:https://doi.org/10.1145/3424239. [accessed 2023 Dec 6]. <https://arxiv.org/ftp/arxiv/papers/2012/2012.00102.pdf>.

[8] dougfinke. Quantum Computing Report - Market Analysis, News & Resources. Quantum Computing Report. <https://quantumcomputingreport.com/>.

[9] PennyLane. pennylaneai. <https://pennylane.ai/>.

[10] Quantum. Quantum. <https://quantum-journal.org/>.

[11] Mathworks, “Mel Frequency Cepstral Coefficients,” Mathworks.com, 2020, doi: <https://doi.org/1061245463.woff2>.

[12] F. Glover, G. Kochenberger, and Y. Du, “Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models,” 4OR, vol. 17, no. 4, pp. 335–371, Nov. 2019, doi: <https://doi.org/10.1007/s10288-019-00424-y>.

[13]“Introduction to Quantum Computing | PennyLane Codebook,” Pennylane.ai, 2025. <https://pennylane.ai/codebook/introduction-to-quantum-computing?form=MG0AV3> (accessed Jan. 16, 2025).

[14] “qml.Rot,” Pennylane.ai, 2025. <https://docs.pennylane.ai/en/stable/code/api/pennylane.Rot.html> (accessed Jan. 20, 2025).

[15]“RYGate | IBM Quantum Documentation,” IBM Quantum Documentation, 2024. <https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.RYGate>.

[16] J. S. John, “AdamD: Improved bias-correction in Adam,” arXiv.org, 2021. <https://arxiv.org/abs/2110.10828> (accessed Jan. 20, 2025).

[17] “Xanadu | X-Series,” Xanadu.ai, 2025. <https://xanadu.ai/products/x-series/?form=MG0AV3> (accessed Feb. 06, 2025).

[18] A. Ranjan, T. Patel, H. Gandhi, D. Silver, W. Cutler, and D. Tiwari, “Experimental Evaluation of Xanadu X8 Photonic Quantum Computer: Error Measurement, Characterization and Implications,” Zenodo (CERN European Organization for Nuclear Research), pp. 1–13, Nov. 2023, doi: <https://doi.org/10.1145/3581784.3607058>.

‌[19] J. M. Arrazola et al., “Quantum circuits with many photons on a programmable nanophotonic chip,” Nature, vol. 591, no. 7848, pp. 54–60, Mar. 2021, doi: <https://doi.org/10.1038/s41586-021-03202-1>.

[20] “Strawberry Fields,” Strawberryfields.ai, 2023. <https://strawberryfields.ai/> (accessed Feb. 14, 2025).

‌

‌

‌

‌

‌

‌

‌

**Appendix**

**QUBO Problem solved with various devices**

Simulation of a Quantum Computer (device: default.qubit):

**A screenshot of a computer

AI-generated content may be incorrect.**A screenshot of a computer program

AI-generated content may be incorrect.

A white rectangular object with black text

AI-generated content may be incorrect.

Executing Quantum Algorithms on Xanadu's X8 Hardware (device: strawberryfields.X8):

A screenshot of a computer program

AI-generated content may be incorrect.

Executing Quantum Algorithms on Xanadu's Borealis Hardware (device: borealis):

A screenshot of a computer program

AI-generated content may be incorrect.

Executing Quantum Algorithms on Xanadu's Aurora Hardware (device: Aurora):

A screenshot of a computer program

AI-generated content may be incorrect.