A Quantum Parallel Framework

For

Distributed Quantum Algorithm Execution, Architecture, Scheduling,

And

Industrial Case Studies Across Simulators and QPU Hardware

Dissertation Manuscript

Submitted to National University

School of Technology and Engineering

In Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

by

JUAN CARLOS SALCEDO

San Diego, California

February 2026

Abstract

With the arrival of robust error correcting quantum processors in the 5-year horizon, some sectors of the industry will require quantum computing knowledge and capabilities to stay competitive by integrating solutions with industrial grade closed loop applications. Current projections place mainstream quantum computing by 2030. Quantum computing technologies will be fundamental to maintaining competitive advantages in key technologies including cryptography, optimization, modeling quantum systems, molecular medicine, and image processing. This research develops a novel Quantum Parallel Framework (QPF) and builds the expertise to deploy quantum algorithms for distributed processing for industrial applications. This research addresses the need to develop mature quantum parallel algorithms executing in closed loop by using simulations and hardware for Quantum Processing Units (QPUs). The QPF hosts, schedules and synchronizes the execution of parallel quantum algorithms across simulators as well as IBM QPU hardware. The QPF framework is comprised of a Qiskit interface to manage communications to QPUs, C++ code and an OpenGL scene generation Graphical User Interface (GUI). The case study integrates the QPF manager framework with a Quantum Hadamard Edge Detector, Quantum Convolutional Neural Network, Quantum Charge Coupled Device, and Quantum Crypto Key Distribution algorithms. QPF parallelizes the execution of multiple QPU instances of the algorithm. Also, the QPF will compare to Quantum Interlin-q, a similar framework. In this research we explored alternate parallel processing methodologies that successfully and significantly yield better performance over classical computing. Although quantum technology is still maturing, the study provides an opportunity to explore complex distributed parallel quantum algorithms in preparation for mainstream quantum computing by 2030.

Acknowledgements

I want to thank God for all the gifts he has given me throughout my life including the thirst, curiosity, and drive to learn. I also want to thank my family, primarily my wife Maria Eugenia, and my children Juan Carlos, Victoria Eugenia, and Marcela Margarita for the unwavering support and love given in my pursuit of this doctoral degree. In spite of my research and profession playing an important part of my life, my family are my greatest project and achievement. I also want to thank my employer and in particular my supervisor Deepak Basandrai for the solid support, patience, and his unwavering interest in my studies. Finally, I want to thank my doctoral committee, including Dr. Khaled Ahmed - Chair, who with his mentorship helped me refine and expand the scope of my dissertation and provide a more complete research study with publication of an articles, and five test cases including cryptography, and quantum neural networks. Dr. Ahmed's feedback brough a higher level of excellence and robustness to my research. Dr. Saman Sarraf – Subject Matter Expert, brought a level of expertise that helped guide the software architecture. Dr. Sarraf also served as a guiding light providing concise direct feedback and the assurance that the direction of my technical solutions were on a sound path. Finally, Dr. Jason Pittman – Academic Reader, helped me architect a solid foundation for my research including the design of both the problem and purpose statements as well as the section for algorithm scalability. The guidance offered by Dr. Pittman was foundational to my research, and I am thankful for his patience and professionalism. In summary, I am grateful to my doctoral committee for their guidance in helping me polish my research and make corrections to the rudder of my doctoral studies with precise, constructive, and effective feedback.

Table of Contents

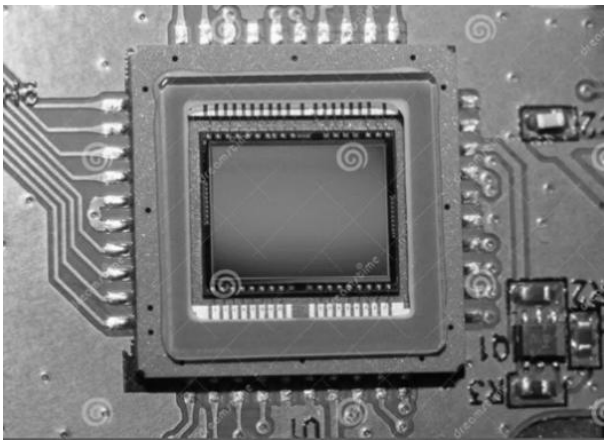
List of Tables

List of Figures

Chapter 1: Introduction

The challenge with current image processing algorithms is the sheer size of the input image and the requirement to process in real-time for some applications. The trend in camera Charge Coupled Devices (CCDs), for example, shows an ever-increasing pattern of large resolutions, increased pixel depth, and high refresh rates as shown in Figure 1. Modeling signal processing algorithms using classical computing technologies to accelerate integration of image processing is becoming more challenging as new CCD camera capabilities increase (Potma et al., 2021). Typical camera image resolutions are now in the 12-megapixel range or larger (Obaidat, 2020). Image processing of larger images and faster camera rates is driving new designs and posing new challenges for image processing, cryptography, and surveillance solutions.

Figure 1. *Olympus Camera Electro Optical CCD Photo Sensor.*



Note. Image shows a CCD integrated with the Olympus camera. From Dreamstime. December 4, 2023. 11. A sensitive matrix of the digital photo camera. https://www.dreamstime.com/stock-photo-sensitive-matrix-digital-photo-camera-olympus-image68322172.

For example, the Sony α9 camera has 24.4-megapixel resolution per digital image (Yan et al., 2023). The trend shows that some high-end CCD camera designs will have higher image resolutions, pixel depth, and faster refresh rate frequencies up to 500Hz (Potma et al., 2021).

Another example of capability trends for high-end CCD devices is the one developed by SmartSens, which unveiled the SC550XS ultra-high-resolution image sensor based on Complementary Metal-Oxide Semiconductors (CMOS), a 50-megapixel CCD featuring 1.0 micrometer pixels (GoPhotonics, 2023). Placing into perspective a modern camera of 4096x4096x32 bits per pixel operating at a refresh rate of 60Hz will generate an image stream of 4GB per second. The projection shows that in 2024 the CCD camera market was valued at 24.4 billion USD and is forecasted to reach 49.35 billion USD by the year 2031 with a growth of 6.9% Compound Annual Growth Rate (CAGR) (Verified Market Research, 2024). A common solution to accelerate image processing is to offload and distribute work to multiple Graphical Processing Units (GPUs). However, increasing the number of GPUs to manage larger images adds overhead time for image assembly and synchronization as each GPU is responsible for processing an image tile of the larger image (Muthukrishnan, 2022; Shukla & Vedula, 2023). Two challenges arise from multi-GPU memory management. One is the duplication of image data when all GPUs are treated as a single logical hardware entity. The second is the impact to scalability with increased image streams from GPU to Central Processing Unit (CPU), PCIe arbitration, and GPU to GPU data transfers (Dong & Peng, 2023).

An additional consideration with multi-GPU solutions is the carbon dioxide (CO_2) footprint of High-Performance Computing (HPC). Current HPC projections, including usage of GPUs, are projected to grow according to the US International Trade Commission, from 1.2 trillion gigabytes in 2010 to a staggering 175 trillion gigabytes by 2025 and beyond. The estimates show that by 2030, datacenters and HPC systems may account for up to 8% of the worldwide emissions if no changes are made (Yang et al., 2023).

For certain problem domains distributed quantum computing offers power consumption efficiency over classical computing (Meier & Yamasaki, 2023). This study explored an alternative processing approach to GPUs that may offer better performance upon arrival of more power self-error correcting QPUs. Current projections show that one of the biggest challenges in quantum computing is the capability of QPUs to correct or compensate for quantum computing errors. Much of the research in both quantum algorithm and quantum hardware design is to achieve quantum processors with an error free fidelity of 99.5 or better to be able to achieve the level of robustness necessary for mainstream applications (Bravyi et al., 2022; Zamora, 2022). Quantum engineering is a new emerging field focused on developing software and hardware to tackle error self-correction solutions and quantum processors with more quantum binary digits (Qubits) (Kop, 2025). Robust quantum computing will require the development of technologies including quantum processors with larger Qubit counts based on superconducting Qubits, self-error correcting logical Qubits acting as one, and surface code self-correcting schemes (Shafique et al., 2024). Research by the Quanta group at Massachusetts Institute of Technology (MIT) projects that quantum computing will be governed by Schoelkopf's Law, which states that quantum decoherence will be delayed by a factor of ten every three years for superconducting Qubits (Pitowsky, 2020). This implies that quantum computing programs will run longer before decoherence is reached. In addition, a literature review shows that within five years error correcting superconducting Qubits (AbuGhanem, 2024) will be integrated with mainstream computing solutions to address commercial useful applications (Vasconcelos, 2020).

Statement of the Problem

The problem addressed in this study was that a single QPU cannot solve distributed quantum computing applications that require more Qubits than available on a QPU, which limits the scalability and fidelity of image processing, cryptography, neural networks, error correction, and physics modeling (Davis et al., 2024; van Dijk et al., 2019). According to Ichikawa et al. (2023), in 2022 condensed physics applications were using more than 100 Qubits, which are approaching Amazon 105 Qubit QPU, and IBM 127 Qubit QPU (AbuGhanem, 2025). Additionally, the pervasiveness of noise effects increases as the complexity and depth of Quantum Circuit (QC) increases, which makes a case to partition an application into smaller shallow QC and distribute processing across several QPUs (Kan et al., 2024; Davis et al., 2023).

Distributed quantum algorithm processing is needed to develop applications to solve large scale QC algorithms, including decomposition into smaller problems for complex solutions that require more Qubits than available on a single QPU (Kan et al., 2024). This research developed a software framework for distributed parallel processing for modeling more Qubits than available in a single QPU, including performance improvement, maturation, and integration of QC algorithms for closed-loop applications.

The impact of this research will benefit applications that use the most Qubits including machine learning, condensed matter physics, quantum chemistry, and quantum error correction with projections of increased Qubit Quantum Volume (QV) (Ichikawa et al., 2023). This study will help with ongoing research to address development of QPUs with larger Qubit counts, and fault tolerance to remove the barrier that prohibits large scaling of Shor's, Glover's, Deutsch-Jozsa's, and other algorithms (Shafique et al., 2024). In general, this research benefits include faster search times and distributed processing to produce expedited solutions for molecular

medicine, drug discovery, artificial intelligence, machine learning, communications, cryptography, image processing, and physics modeling (Balamurugan et al., 2024; Shafique et al., 2024). Quantum Error Correction (QEC) based on machine learning (Melvin, 2022) will benefit as well by using more complex algorithms. Finally, the benefit of financial reduction of quantum application development is significant. As publication of this article, IBM charges $96 per minute for quantum processing, which can quickly increase to a proposition of tens of thousands of dollars (IBM Pricing, 2025).

Purpose of the Study

The purpose of this positivistic worldview quantitative constructive experimental research was to construct and validate a novel distributed parallel processing framework for quantum computing. More specifically, this research developed a framework to experimentally evaluate performance of distributed parallel QC algorithms under the impact of quantum noise modeling to provide QPU representative results (Grurl et al., 2023). This study is aimed at modeling distributed QC algorithms to expedite development, and maturation through experimentation to find which combination (of independent variables) including number of QPUs, Non-Uniform Memory Allocation (NUMA) strategy, CPU core affinity & priority, and OS scheduling offers the best performance (for dependent variables) including response time, Euclidian distance, probability of classification, cryptographic key generation rate, and probability of tamper detection.

The selection of test case QC algorithms covers a breath of distributed processing applications including a Quantum Hadamard Edge Detector (QHED), Quantum Convolutional Neural Network (QCNN), Quantum Charge Coupled Device (QCCD), Quantum Crypto Key

Distribution (QCKD), and Quantum Interlin-q (QILQ) for framework performance reference (Balamurugan et al., 2024; Shafique et al., 2024).

This research expanded work by Koch et al. (2019) with a closed loop framework capable of hosting distributed QC algorithms that collaborate processing data by working in parallel across several QPUs. The data collection methodology used probability sampling as each sample has a non-zero chance of being selected. Apriority power analysis used $\alpha < 0.05$ for statistical significance to reject Null hypotheses. The α-value demonstrates less than 1 in 20 chances of results occurring by chance (Hazra & Gogtay, 2016). Achieving $\alpha < 0.05$ per GPower analysis required a sample size of 14,428 for each dataset collected for each QC algorithm test case.

Introduction to Conceptual Framework

This quantitative experimental study investigated distributed framework feasibility, parallel processing performance, and reliability of results under quantum noise. The study tested combinations of independent variables and observed if there is a clear link between framework reliability, QC algorithm response time, and impact of quantum noise on quality of results for all QC test case algorithms.

Quantum theory was used to integrate how Qubits operate, including entanglement, superposition, interference, and measurement (Peres, 2002). Quantum mechanics governs and allows superposition uncertainty, which provides native parallel processing until measured (de Forges et al., 2024). Entanglement is a correlation between Qubits, where the state depends on the other regardless of distance. This attribute allows the creation of complex QC algorithms (Shafique et al., 2024). In addition, a consideration that must be mitigated by multiple shots is

wave collapse or decoherence, which can occur by external environment interference (Shafique et al., 2024).

The software framework used General System Theory (GST) principles for sub-system software model decomposition, modularity, and decoupled component layers to facilitate development of distributed software (Johnson, 2019). Framework scalability is configurable for key distributed execution (Levin, 2005). Adaptive parallel processing, inter-process client-server communications, and synchronization principles for multithreaded software drove the framework design including data assembly across processing elements (Han et al., 2024; Lindsay et al., 2021; Solarte-Martínez et al., 2021). The architecture included distributed HPC data partition principles for thread management and synchronization (Dong et al., 2024; Davis et al., 2023). The architecture is layered and hierarchical to facilitate responsibility delegation (Zhou & Preindl, 2023). This research models superconducting noise at Josephson tunnel junctions caused by quantum phase relaxation, radiation, readout effects, and temperature variations, which cause the collapse of a Qubit's wave function (Brunk & Lübbig, 1981). Data encoding errors and noise were modeled for QPU representative in family results (Dolciami, 2022; Iyengar et al., 2020).

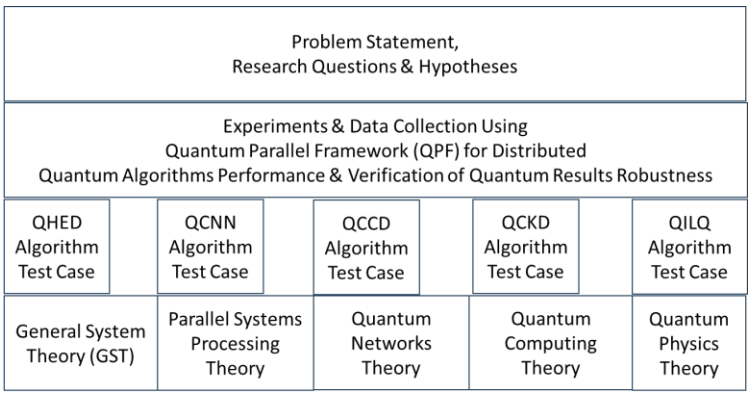
Introduction to Research Methodology and Design

The design of this quantitative study was constructive research used to experimentally test framework QC algorithm distribution, feasibility, parallel performance, and data quality under noisy conditions for five QC algorithm test domains. The design of the study was aligned with the problem statement to answer each of the three-research questions for framework feasibility, distributed QC algorithm performance, and effects of quantum noise on algorithms results (Grurl et al., 2023). Data collection was driven by a closed loop distributed parallel framework, which synchronized all thread processing for managing quantum QPUs. The

framework distributes work across all QPU elements including simulators, synchronizes frame completion, and calculates metrics for Measures of Effectiveness (MoE) for all five QC test experiments as shown in Figure 2.

The following Seminal works were used to develop the quantum distributed parallel framework including the five QC algorithm test cases. Seminal works used by this research include QC algorithms for factoring large prime numbers, which bring risks to Rivest–Shamir–Adleman (RSA) encryption (Shor, 1994). The QC algorithm principles developed by Shor (1994) were modeled in this study by the QCKD cryptographic test case. Works by Lov Gover on efficient search methodologies using the Walsh-Hadamard (W-H) transform provide an advantage over classical computer for exhaustive searches (Grover, 1988). The principles explored by research in Hadamard gates were employed in this research by the QHED edge detector and QCNN test cases. The Qiskit framework was referenced for salient QC examples including parallel processing of analog to digital conversions by the QCCD test case (Koch et al., 2019). A reference framework was used to compare this study performance. The comparison used QC CNOT gates for the reference and this research study to simplify the performance analysis for the QILQ test case. In addition, the Qiskit foundation covered by Koch et al. (2019) was integrated by the quantum framework developed for this study to interface to the QPU simulators and QPU hardware.

Figure 2. *Study Structured Experiments Across Five Salient Quantum Algorithm Domains.*



Note. The test space will be comprised of five distributed test case algorithms.

There was a degree of randomness driven by OS scheduling timing and variations in the algorithm for specific metrics. The datasets were designed to address each of the hypotheses and research questions by including QC algorithm distributed performance, reliability assessment, and data quality metrics under quantum noise conditions. At least 14,428 samples were collected for each of the five test case experiments as shown in Figure 2.

Research Questions

The research was aimed at presenting quantifiable results and collecting synthetic quantum simulator-generated data. Analysis of data demonstrated correlation between the experiments and the ability to produce a feasible stable quantum parallel framework, and performance improvements when compared to a single QC instance. Metrics MoE were generated specifically for each of the five QC algorithms to evaluate the impact of quantum noise on the quality of the data. Analysis was performed on data collected for all five QC test case algorithms to critically analyze and produce meaningful conclusions. The Research Questions (RQ) and their corresponding null $H1_0$, $H2_0$, $H3_0$, and alternate hypotheses $H1_a$, $H2_a$, and $H3_a$ were aligned with this research problem and purpose statement.

The seminal works cited support problem statement alignment using quantum physics foundation required to develop quantum computing systems including simulations required to collect the data and address the problem statement. The quantitative data collection for answering RQ1 included QC algorithm performance to address research feasibility to integrate a robust solution for closed loop operations. RQ2 was addressed by the collection of data to demonstrate that distributed parallelization of QC algorithms offers performance improvements. Finally, RQ3 was addressed by the collection of MoE metrics that addressed the quality of data under noisy conditions including Euclidian distance, probability of correct image classification, and probability of cryptographic key tampering. Figure 3 shows how the data collected for each of the five QC test case algorithms was used to support the three hypotheses and research questions to address the problem statement.

Figure 3. *Algorithm Support for Each of The Hypotheses H_1 through H_3.*

Algorithm	Metrics Used for Hypotheses Validation	Comments
QHED	• Parallel processing time (H1, H2,RQ1, RQ2,) • Image Euclidian Distance (H3, RQ3, RQ3, RQ3)	Quantum Hadamard Edge Detector Hypotheses & RQ validation: Time, Euclidian, Noise Modeling
QCNN	• Parallel processing time (H1, H2, RQ1, RQ2,) • Classifier Prediction Accuracy (H3, RQ3)	Quantum Convolutional Neural Network Hypotheses & RQ validation: Time, Prediction accuracy, Noise Modeling
QCCD	• Parallel processing time (H1, H2, RQ1, RQ2) • Image Euclidian Distance (H3, RQ3)	Quantum Charge Collection Device Hypotheses & RQ validation: Time, Euclidian, Noise Modeling
QCKD	• Parallel processing time (H1, H2, RQ1, RQ2) • Probability of tamper detection (H3, RQ3) • Key Generation Rate (KGR) (H3, RQ3)	Quantum Cryptography Key Distribution Algorithm Hypotheses & RQ validation: Time, KGR, Tamper probability, Noise does not appear to affect keys
QILQ	• Parallel processing time (H1, H2, RQ1, RQ2)	Compare QPF with Interlin-q to establish baseline performance metric

RQ1

Distributed quantum framework feasibility and reliability: How can a quantum distributed framework be designed and developed to maintain system reliability and stability when executing distributed parallel quantum algorithms over extended operational periods?

RQ2

Distributed quantum framework performance: What is the impact on performance response time gains by parallelizing a distributed QC algorithm in a simulation framework when compared to a single QC algorithm instance?

RQ3

Distributed quantum framework results in robustness under quantum noise conditions: How does the integration of quantum noise models affect the robustness and accuracy of distributed parallel QC algorithms within the proposed quantum simulation framework?

Hypotheses

The null hypotheses $H1_0$, $H2_0$, $H3_0$ addressed respectively the negation of the postulated premises for feasibility, performance, and validation of quantum results. The alternate hypotheses $H1_a$, $H2_a$, and $H3_a$ used MoE calculated from data collections for the five case studies as shown in Figure 2. The hypotheses addressed simulation feasibility, QC algorithm model processing performance, and the results of robustness under quantum noise conditions when compared to a reference model (Alvesson & Sandberg, 2013). The following are the MoE metrics utilized to evaluate the three hypotheses:

- QHED & QCCD models metrics include performance and Euclidian distance.

- QCNN metrics include performance and probability of correct classification.
- QCKD metrics include performance and probability of tamper detection and key generation rate.
- QILQ metrics include performance comparison to this study framework.

$H1_0$

Null hypothesis for distributed quantum framework shows no feasibility and reliability: Running multiple distributed parallel QC algorithms causes the quantum framework to crash at least once and not collect all samples in a single run.

$H1_a$

Distributed quantum framework feasibility and reliability: Running multiple distributed parallel QC algorithms does not affect the quantum framework reliability and ability to collect all samples in a single run.

$H2_0$

Null hypothesis for distributed quantum framework negligible performance: The performance response time from multiple distributed parallel QC algorithm instances working on a dataset shows no performance improvement when compared to the response times of a single QC algorithm instance working on the same dataset.

$H2_a$

Distributed quantum framework performance: The performance response time from multiple distributed parallel QC algorithm instances working on a dataset shows performance improvement when compared to the response times of a single QC algorithm instance working on the same dataset.

$H3_0$

The null hypothesis for distributed quantum framework results shows no quantum noise modeling robustness: The distributed QC algorithm models under quantum noise conditions will generate results that are not degraded when compared to the quantum results without noise.

$H3_a$

Distributed quantum framework results show quantum noise modeling robustness: The distributed QC algorithm under quantum noise conditions will generate results that are degraded when compared to the quantum results without noise.

Significance of Study

This research provided the organization with the opportunity to expand expertise in quantum computing and grow as technology becomes mainstream, including the following technology insertion examples. This research is important to the computer science community for research of distributed quantum networks, where it can be used to model QPU networks over a city or region including Space-Aerial-Terrestrial Quantum Networks (SATQNs) to support 6G and beyond communications (Zhang et al., 2025; Sebastian-Lombrana et al., 2025). In addition, this research will provide a software platform for maturation of QC algorithms at a reduced cost (IBM Pricing, 2025).

Several areas of science will also benefit from having the ability to execute quantum algorithms in an expedited fashion. The following is a summary of how various disciplines will be able to push science boundaries further with help from this research. Computer science will benefit from the ability to run more complex algorithms that may require more Qubits than

available on a single QPU (Salcedo & Ahmed, 2025). This study facilitates research into complex quantum algorithm design and optimization. Physicists will be able to develop more complex and higher fidelity models requiring a larger number of Qubits including modeling quantum mechanics, and cosmology (Miceli & McGuigan, 2019; Kaufman et al., 2019). Higher fidelity quantum molecular models and faster processing times will help molecular medicine and pharmaceutical research to develop vaccines faster (Vinod et al., 2025; Xu et al., 2025).

Cryptography will benefit from performance improvements when using parallel processing to expedite generation of cryptographic keys including integration of machine learning (ML) with distributed quantum processing to ensure AI training data is tamper resistant (Radanliev, 2024). Crypto research will benefit from more complex QCKD encryption algorithms requiring hardened cryptographic keys comprised of longer rotational strings to mitigate the quantum risks to RSA encryption (Moric et al., 2024; Dooms & Emerencia, 2025). This research can support environmentally friendlier solutions to reduce CO_2 generation by using quantum solutions where applicable (Meier & Yamasaki, 2023). Specifically, a phenomenon known as quantum tunneling reduces the consumption of power 100 to 1000 times when compared to classical computers (Balamurugan et al., 2024).

The telecommunication industry will benefit from expedited maturation and integration of Quantum Error (QE) techniques (Melvin, 2022). In general, for all disciplines using distributed quantum computing, this research expedites generation of test results and allows for exploration of larger test spaces in less time than the time incurred by a single QPU. Therefore, incurring a reduction of costs and faster deliverables deployment to market.

Definitions of Key Terms

Bloch Sphere

A Bloch is used to represent the state of a Qubit in 3D space with a state vector represented by magnitude, yaw, pitch, and roll. The Bloch sphere was introduced by Felix Bloch in 1946 to represent the state of a Qubit (Balamurugan et al., 2024).

Constructive & Destructive Interference

A Qubit can be represented as a wave function $\psi(x)$ with amplitude representing the state of the Qubit. The wave function of two distinct Qubits with the same frequency and distinct phases can affect each Qubit. If the phase of the wave is the same, the resulting wave can be described as constructive, as both wave amplitudes are added. On the other hand, if two Qubits have waves of the same frequency and amplitude, but distinct phases, the amplitudes subtract and the two waves describe destructive interference (Shafique et al., 2024).

Closed Loop System (Self Feedback System)

An iterative system designed to process data and feedback on the results of each cycle as inputs to the next iteration cycle until a goal is reached (Escudier, 2019). An example is an image processing object tracker, where the camera attitude (yaw, pitch, and roll) and position of the camera are adjusted to keep the object in the Field of View (FOV) frame by frame.

Distributed Quantum Computing

Distributed quantum computing is the paradigm of allocating a set of QPUs connected via a quantum network to solve a problem that would be otherwise require a single large capacity quantum computer that may not be available or feasible (Davis et al., 2024).

Euclidian Distance

The distance between two points in a plane or radiometric difference between two pixels (Divya, 2018).

Non-Uniform Memory Allocation

NUMA memory allocation strategy when enabled relies on memory allocation close to the CPU and core requesting memory. On the other hand, when NUMA is disabled, the OS interleaves allocation of memory based on the memory bank scheduled for allocation regardless of thread executing processor, which may increase memory access latency (Bai et al., 2025).

Pipeline Processing

A form of processing, analogous to a manufacturing production line, in which the time required to pass through some functional unit. For example, a floating-point Arithmetic Logic Unit (ALU) of a computer system is longer than the intervals at which data may enter, i.e. the functional unit performs its process in several steps (Butterfield et al., 2025).

Quantum Circuit

A logic circuit in which each gate is a quantum logic gate, i.e., a gate in which the role of a bit is played by a Qubit (Rennie & Law, 2025).

Quantum Entanglement

The probability of a Qubit state can be affected by the change of another Qubit state's probability. The phenomenon that occurs when two or more particles become correlated in such a way that the state of one Qubit is dependent on the state of the other Qubit, regardless of the distance between them (Shafique et al., 2024).

Quantum Image Encoding

There are several image encoding algorithms for storing the image in the QPU including the Flexible Representation of Quantum Images (FRQI), Novel Enhanced Quantum Representation (NEQR), and Quantum Boolean Image Processing (QBIP) (Mastriani, 2020; Chaduvula et al., 2024). A consideration in quantum image processing is that some quantum image encoding algorithms come with Quantum Probability Image Encoding (QPIE) errors. For example, FRQI and QBIP are non-exact image encoding algorithms, whether NEQR is an exact algorithm (Dolciami, 2022).

Quantum Interference

The external effects that change the stability of a Qubit to maintain its state can be due to external radiation, temperature variations, and wave interference. In the case of wave functions that describe quantum particles, waves that add to the amplitude are referred to as constructive, and waves that cancel are referred to as destructive. Interference allows the user to control processing by biasing desired results and canceling undesirable results using manipulation of wave magnitude and phase (Balamurugan et al., 2024).

Quantum Noise Modeling

The state of the art in quantum processing is still not perfect, and processing can be plagued by errors due to external conditions such as temperature variations in the dilution refrigerated environment. In addition, external factors like radiation and temperature variations can affect Qubit and collapse Qubit’s wave function (Dolciami, 2022).

Quantum Parallel Framework (QPF)

QPF is a software framework to be developed and integrated for distributed parallel processing and synchronization of QC algorithms running on QPU simulators or QPU hardware to generate the experiment data for this research.

Quantum Superposition

Superposition is Qubit's contextual indeterminism to maintain multiple concurrent non-interacting quantum states until measured (de Forges et al., 2024).

Quantum Wave Collapse

Wave collapse is defined as the Qubit's definite coalesced state upon measurement, which loses superposition at the point of sampling (Shafique et al., 2024).

Sensor CCD

A Charge Coupled Device (CCD) is responsible for capturing radiation for a camera and converting the analog signal for digital image processing (Ortiz & Oliver, 2004).

Summary

As quantum technology matures and increases capabilities in self-error correction and with QPU offerings of larger number of Qubits, the technology will become more mainstream with quantum algorithms offering tangible benefits to an organization. Available classical processing technologies are based on CPU and GPU solutions, which can be parallelized where each GPU is responsible for processing a subsection of the dataset. These solutions allow for effective parallelization of algorithms; however, the overhead for data movement and communication between CPU and GPUs to assemble the composite results starts to become problematic, especially as the number of GPUs increases. Also, a concern is that CO_2 footprint

projected for HPC may account for up to 8% of the worldwide emissions if no changes are made. A motivation for supplementing HPC with more power efficient quantum solutions is to change the 2030 carbon footprint projections trajectory. Some problem domains, including cryptography, optimizations, and quantum modeling, will benefit from more power efficient quantum computing.

Quantum theory principles provided the foundation for establishing constructive exploratory research to investigate distributed quantum computing. The instruments produced by this research were integrated with closed loop industrial simulations including development of the software framework required to conduct the experiment. The research also included the development of metrics to evaluate the robustness of QC algorithm results under quantum noise conditions. Quantum computing solutions offer opportunities to improve performance for some problems or even achieve quantum supremacy for specific problems. The driving force behind this research was to support early modeling of parallelization of distributed quantum algorithms to determine the configuration that will yield the best performance. Although the current state of the art of QPUs and image encoding methods exhibit errors, it is in the best interest of the organization to invest in research in core technologies to establish knowledge in this emerging domain. An additional objective of this research was to help the organization learn to integrate quantum computing into industrial solutions and maintain competitive advantage as quantum technology becomes mainstream by 2030.

Chapter 2: Literature Review

The purpose of this quantitative constructive research was to experimentally explore development and maturation of distributed quantum applications that cannot be solved by a single QPU due to the considerable number of Qubits required (Davis et al., 2024). Building large scale QPUs to achieve federated QC operating at quantum speeds and scale is still a challenge (Melvin, 2022). This research had the goal of developing a framework to host QC algorithms to understand the performance improvement of distributed parallel processing under the effects of quantum noise. This research expedites the generation of test results and allows for exploration of larger test spaces in less time than a single QPU could take. Therefore, incurring a reduction of costs and faster deliverables deployment to market. The research generated datasets based on five test cases to support the research questions and hypotheses including QC algorithms for QHED, QCNN, QCCD, QCKD, and QILQ.

The literature review is comprised of six sections starting with quantum computing theory, which lay the foundations for QC algorithm principles, superposition, entanglement, and wave decoherence to be modeled by this research (Peres, 2002). Market trends were analyzed for complexity of future sensor devices, which will drive image processing algorithms and quantum computing solutions (Potma et al., 2021; Obaidat, 2020; Yan et al., 2023). The literature review explored how complexity of future algorithms will need to be complemented, where possible, with more efficient quantum solutions including reduction of CO_2 (Meier & Yamasaki, 2023). Quantum challenges were explored as well including limitation of reduced number of Qubits per QPU, quantum noise, and quantum decoherence timing. This research will include the effects of these limitations, and provide a higher fidelity modeling capability (Pitowsky, 2020). Quantum scaling considerations were explored as well to identify how QC circuits can be partitioned and

scaled to support distributed parallel processing (Kan et al., 2024, Han et al., 2024; Chalumuri et al., 2022). The literature was also reviewed to identify scaling challenges for QPUs, and how Qubits may be integrated by this research to support Qubit interconnects and inter process synchronization (van Dijk et al., 2019). The review also included how the design of the quantum distributed framework will use finite state machine principles to guide each processing cycle, and facilitate future software updates with little impact to the software architecture (Wang & Zhang, 2024; Babakov & Barkalov, 2025). Also, current available frameworks were reviewed to identify which one can serve as the foundation for a distributed parallel architecture (Jimenez-Navajas et al., 2024; Han et al., 2024; Lindsay et al., 2021; Solarte-Martínez et al., 2021). Finally, framework requirements were identified as guidance for the distributed parallel framework design and development.

Literature Review Search Strategies

The literature review research was conducted using the National University Library complemented by Google Scholar, Springer, ProQuest, IEEE Access, and IEEE Xplore. In addition, secondary sources including the arXiv database were used to complement primary search material. Articles, periodicals, and related material were searched with the peer reviewed attribute. Search years were narrowed between 2019 and 2025 with exceptions for seminal works, which date back to 1980s, 1990s and early 2000s. Table 1 shows the search criteria and research terms categories.

The literature review covered CCD market growth trends, quantum cryptography, and future camera designs and projections for increased camera resolutions, pixel depth, and higher refresh rates, which will directly challenge future image processing algorithms design and complexity (Potma et al., 2021; Obaidat, 2020).

The literature review evaluated the current landscape of quantum algorithm technology across five test case pillars applicable to image processing as shown on Figure 2. The test cases include edge detection, convolutional neural networks, charge-coupled devices, cryptographic key distribution applications, and multi-QPU synchronization solutions. The selection of the five test cases was based on breath of applications to show applicability of this research to various computer science domains. The literature review also included distributed and layered software development architectures to support the integration of quantum technology with industrial closed-loop systems (Fernandez-Conde et al., 2022; Zhou & Preindl, 2023). Finally, the literature review made the case for identifying the current challenges of reduced Qubit counts for available QPUs, which limits development of solutions to complex problems requiring more Qubits than available on a single QPU. This study also identified how the instruments developed by this research can be used to model, design, development, and integration of distributed closed-loop simulations (Davis et al., 2024; van Dijk et al., 2019).

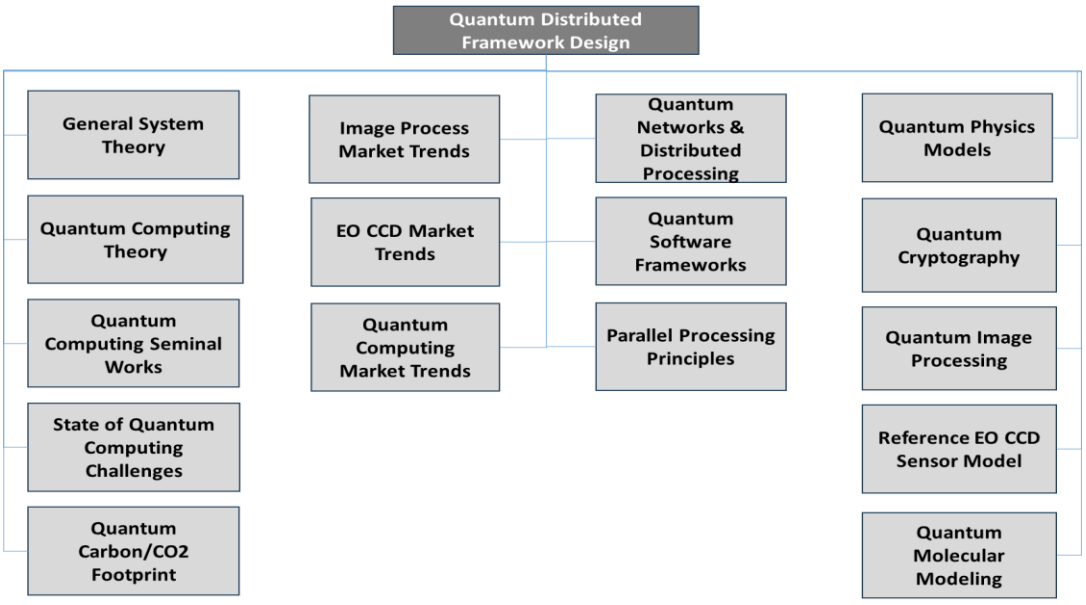
Table 1. *Search Criteria and Terms Used for Literature Review.*

Item	**Search Term Categories** **Peer Reviewed, Academic Journals, Past 5-years**	**Search Results**
1	CCD Sensors	408
2	Distributed Parallel Processing	775
3	Electro Optical CCD Sensors	6,841
4	Finite State Machine Applications	35
5	General System Theory	13,275
6	Image Processing Trends	82
7	Quantum Carbon - CO_2 Footprint	4
8	Quantum Computing Limitations	80
9	Quantum Computing Networks	775
10	Quantum Computing Performance	365
11	Quantum Computing Principles	367
12	Quantum Computing Trends	37
13	Quantum Convolutional Neural Networks	499

14	Quantum Cryptography	5,309
15	Quantum Image Processing	338
16	Quantum Molecular Modeling	579
17	Quantum Physics Modeling	107
18	Quantum Software Frameworks	25
	Total Articles	**29,901**

More than four hundred articles were reviewed; however, a taxonomy of the most relevant categories containing 86 articles referenced are listed in Figure 4. The exclusion criteria were quantum physics articles not applicable to quantum computing. The articles were organized in categories outlined by the taxonomy decomposition. The search criteria established the fundamentals of quantum computing principles for quantum image processing and concludes with some challenges including QPU hardware noise pervasiveness. The article organization was designed to serve as pillars to build the knowledge necessary to support the design, development, and integration of the instruments to generate the data required to address the hypotheses, research questions, and problem statements.

Figure 4. *Quantum Parallel Framework and Applications Review Taxonomy.*



Theoretical Framework

Quantum computing theory and General System Theory were the foundations for this research, and the software instruments developed to conduct the experiments to determine the feasibility, performance, robustness of a distributed quantum parallel framework. GST helped define the software architecture and relationships that comprised the communication and process synchronization of the various subsystems (Johnson, 2019). This section covers exploration of quantum computing foundations, quantum computing challenges, algorithm scalability & partitioning considerations, and parallel threaded processing & synchronization (Kan et al., 2024; Davis et al., 2023).

Quantum Computing Foundations

Fundamental understanding of the uncertainty of quantum mechanics, superposition, entanglement, and wave function collapse (decoherence) are required to develop QC algorithms for this research (Koch et al., 2021). Quantum computing is a concept that was first introduced in 1981 by Richard Feynman. Feynman proposed that a quantum computer could be used to simulate quantum systems and avoid the exponential resources needed by a classical computer (Sager-Smith, 2023). Quantum theory offers a volumetric model to explicitly predict the position in space where a particle is at any moment (Peres, 2002). This describes the uncertain behavior of QC circuits, which is based on Heisenberg's intrinsic irreproducibility principle of quantum experiments. Qubit quantum states are governed by Heisenberg's principles. For example, some Qubit implementations use photons to store states (Vasconcelos, 2020). For these cases Fourier analysis predicts a probabilistic location of a photon particle $1/\lambda = 1/p$, which requires a volumetric function $\Pi\, \Delta\, q^i\, \Delta\, p_i \gtrsim\ \Delta\, h^3$, where q is the photon's position, p the momentum, *i-th* the particle index, and $\Delta\, h^3$ the probabilistic differential volume where the photon is located

(Peres, 2002). Qubit uncertainty and quantum noise are the reasons QC algorithms require many iterations shots to provide a statistical outcome. A single iteration of a QC will not yield reliable results (Iyengar et al., 2020).

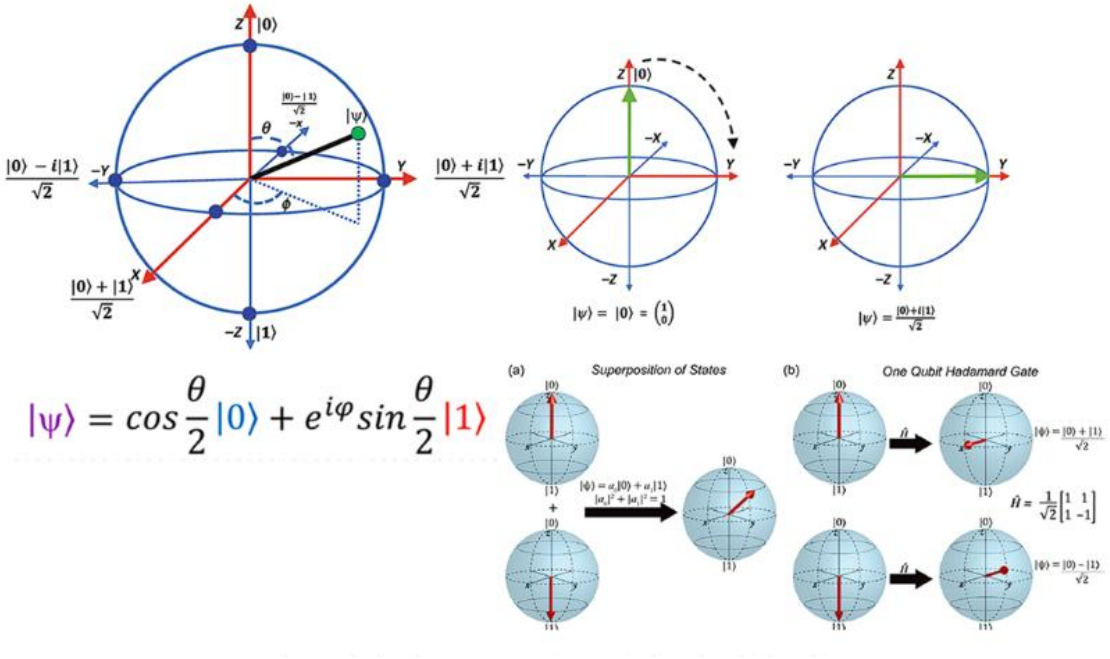
Qubit Principles

Qubit is the smallest storage mechanism a quantum computer uses to store binary digit information like a classical bit. The advantage a Qubit has is its capacity to store more information, and its ability to be at many superposition states at once (Sager-Smith, 2023). The superposition principle states that in a quantified world, a particle can be in two places at once, which is an impossibility in a macroscopic world. Superposition can be represented as a linear expression of the 0 and 1 states, where α and β are the probabilistic amplitudes of the wave function of the states 0 and 1, respectively (Peres, 2002). For a scenario of 2 bits and 2 Qubits, the possible numbers of states can be |00>, |01>, |10>, |11>. For N units, the total possible states are 2^N; therefore, for every extra Qubit, the storage capacity doubles for a QPU. An effective way of visualizing a Qubit state is by using a Bloch sphere, introduced by Felix Bloch in 1946 (Balamurugan et al., 2024). A quantum super positioned state can be represented by a random number on the surface of the Bloch sphere as shown in Figure 5. The north and south poles are the pure states for |0> and |1> respectively, which can be represented by the wave function equation $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ or, $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\varphi}{2}\, e^{i\varphi}|1\rangle$, $where\ 0 \leq \theta \leq \pi;\ 0 \leq \varphi \leq 2\pi$ (Balamurugan et al., 2024).

The state of a Qubit can be represented by a vector with origin at the center of the Bloch sphere, and magnitude pointing at the surface of the sphere. In addition, the α and β angles can also be used to encode information in a Qubit. Finally, this research superposition will be

implemented on a single Qubit by applying the Hadamard gate, named after Jacques Hadamard and Joseph L. Walsh (Shafique et al., 2024).

Figure 5. *Bloch Sphere Concepts for Single and Multiple Qubits.*



Note. Image shows a single Qubit Bloch sphere representation with example states. From Balamurugan, K. S., Sivakami A., Mathankumar M., Satya prasad, Y. J. D., & Ahmad, I. (2024). Quantum computing basics, applications, and future perspectives. Journal of Molecular Structure, 1308. https://doi.org/10.1016/j.molstruc.2024.137917.

Qubit Superposition Principles

The ability of a Qubit to model several states concurrently can be harvested for parallelism. Superposition is the main reason for the exponential power a QPU has over classical computing systems (Zhu et al., 2024). For practical purposes, a Hadamard gate is used in this research to place a Qubit in superposition. The advantage that a QPU has over a classical CPU is that it can measure the superposition states of all Qubits in parallel, instead of one by one like a classical computer (de Forges et al., 2024). Another example of quantum versus classical

computer advantages can be highlighted by Google Sycamore QPU. Sycamore is a 70-Qubit system, with a total theoretical memory address of 2^{70} or 1.1805×10^{21}, or one Exabyte (EX), which is more than the number of stars in the Milky Way. A recent experiment showed Sycamore to find a factor for a large number using 70 qubits in approximately 200 seconds, which would have taken 47 years to solve using Frontier, the most powerful supercomputer (Balamurugan et al., 2024).

Qubit Entanglement Principles

The term entanglement was coined in 1935, when Albert Einstein, Boris Podolsky and Nathan Rosen published a famous paper known as the EPR Paradox or Einstein-Podolsky-Rosen paradox (Balamurugan et al., 2024). Entanglement is the relationship established between two Qubits irrespective of distance (Bradley, 2023). Qubits can be entangled and separated by a few centimeters or large distances, and the entangled properties will be maintained. An interesting fact is that if two Qubits are entangled when measurement takes place, one Qubit will give its result and the other Qubit's result will be the opposite. This is because entanglement is achieved using a CNOT gate (Havlíček, et al., 2019). Entanglement allows for the integration and collaboration of several Qubits working as a group, which is required to exercise the QC to generate data for this research.

Qubit Constructive & Destructive Interference

A Qubit can be represented as a wave function $\psi(x)$ with amplitude representing the state of the Qubit as shown in Figure 5. The wave function of two distinct Qubits with the same frequency and distinct phases can affect each Qubit. If the phase of the wave is the same, the resulting wave can be described as constructive since both wave amplitudes are added. On the other hand, if two Qubits have waves of the same frequency and amplitude, but distinct phases,

the amplitudes subtract and the two waves describe destructive interference (Shafique et al., 2024). Constructive and destructive interference is the process by which a quantum Qubit state can be negated or favored prior to measurement (Balamurugan et al., 2024).

Quantum Computing Challenges

Although quantum computing offers tremendous potential, there are limitations, which this research identifies during distributed processing. Examples of challenges explored in this literature review cover quantum noise errors, delayed wave function stability collapse for complex QC algorithms, maximum number of Qubits limitation for QPU, quantum hardware challenges, and reliance on quantum simulations (Melvin, 2022; Davis et al., 2024; van Dijk et al., 2019).

Quantum Processing Errors

The state of the art in quantum processing is still not perfect, and processing can be plagued by errors due to external conditions such as temperature variations in the dilution refrigerated environment (Paracha et al., 2024). In addition, external factors like radiation and vibration can also affect a Qubit and collapse the wave function (Dolciami, 2022). Due to quantum noise, current algorithms rely on statistical outcomes. For example, for the Quantum Image Processing Algorithms (QIPA) image representation FRQI method, five million shots were required to obtain credible statistical representative results. For this research, a reduced number of 10 shots will be used initially to reduce process execution time and focus on integration of the QC algorithms and parallelization (Iyengar et al., 2020).

Wave Function Stability Collapse

The stability of the Qubit’s wave function can collapse due to external variations in the environment. The concept of Qubit de-coherence is a concern, where a quantum system can

interact with its surrounding environment and lose its quantum properties including superposition, entanglement, and interference (Dolciami, 2022). Coherence time can be thought as the real-time budget a QC algorithm has to complete all its processing before wave function collapse. Under normal circumstances some superconducting Qubits extend the maximum coherence time from 50ns to 500ns, which allows for longer processing times and more complex QC algorithms (Vasconcelos, 2020).

Maximum Number of Qubits Limitations

There is much research in quantum computing to address the development of QPUs with larger Qubit counts, and fault tolerance to remove the barrier that prohibits scaling Shor's, Glover's, Deutsch-Jozsa's and other algorithms (Shafique et al., 2024). Currently Qubits offer a fidelity of 99.5% with applied error correction. However, for a Qubit to be deemed robust and viable it must achieve an error correction fidelity of more than 99.5% accuracy. Error correction is a key barrier that must be conquered for building larger QPUs (Vasconcelos, 2020; Melvin, 2022).

Quantum Hardware Challenges

Performance of QPUs is not deterministic, and companies like IBM and Google are developing hardware designs and showing that various quality QPUs produce different results for the same algorithm (Iyengar et al., 2020). Although much progress has been made, there are several technologies that must be conquered to pave the way to mainstream quantum processing including super conducting processors, quantum photonics, Nuclear Magnetic Resonance (NMR), and trapped ion technologies (Vasconcelos, 2020).

Super Conducting Qubits Challenges. The fundamental concepts of a Qubit are rooted in an inductor and capacitor for harmonic oscillator circuits. In an analogous way, Qubits are

comprised of energy representing 0 and 1 by means of Josephson junctions that at superconducting temperatures function as linear inductors (Vasconcelos, 2020). Super conducting Qubits operate in the microwave range and can be fabricated by lithography, which is a big advantage over other approaches (Brunk & Lübbig, 1981). This research will use IBM cloud systems and simulators based on superconducting Qubits. Although superconducting Qubits extends the coherence time from 50ns to 500ns, which allows for longer processing times, 500ns is still a brief time when compared to classical computers. Extension of longer coherence times is the primary challenge facing superconducting Qubits (Vasconcelos, 2020).

Quantum Photonic Challenges. Ongoing research is focused on converting transducing photons from superconducting microwaves to optical photons, which are more impervious to the surrounding environment and will allow interconnection of QPUs across data centers or a city (Vasconcelos, 2020). An additional technology that offers a viable option for transmittal of quantum signals is thin film lithium niobate. For example, with 26.2 milliwatts of optical power, a measurement of 2.7±0.2% is required to increase the signal-to-noise ratio for phase measurement signals to perform reliably for quantum applications (Stokowski et al., 2023). Photonics will directly benefit Quantum Information Networks (QIN), which will allow for integration of networks connecting quantum devices over long distances (de Forges de Parny et al., 2023).

Nuclear Magnetic Resonance & Trapped Ions Challenges. There is ongoing research to build a QPU out of single atoms and single electrons. Research is focused on trying to solve specific problems in a way that is faster than classical law of physics allows; however, there is a scalability challenge that as quantum component scale up, so do the risks of losing quantum properties of the system (Vasconcelos, 2020).

Reliance on Quantum Simulations

A limitation with Quantum Image Processing Algorithms (QIPA) is that although image representation algorithms have been freely available, the computing community has relied heavily on quantum simulations with only a few algorithms being evaluated on actual QPU hardware (Iyengar et al., 2020). An additional limitation is that current QPU offerings are still in hundreds of Qubits, which limit the size of the data and algorithm complexity that can be processed (Davis et al., 2024; van Dijk et al., 2019). Future technologies such as superconducting processors and photonic communications will support distributed processing QPUs with many Qubits interconnected over large distances (de Forges de Parny et al., 2023). The reality, however, is that these technologies are at least five years from providing QPUs with Qubits counts that are in the thousands (Vasconcelos, 2020).

Quantum Noise Modeling Fidelity

A vital component of QPU simulation is the ability to emulate noise that will produce in-family results with the quality of data expected from QPU hardware. This research will include increasing the fidelity of the simulation by inclusion of gate noise, readout errors, and thermal decoherence relaxation noise (Sager-Smith, 2023). Several approaches are available to mitigate Quantum Error Correction (QEC), such as noise-aware folding techniques to better optimize the QC transpilation synthesis on QPU hardware. However, in spite of these approaches, noise is still present (Hour et al., 2024). This research will use quantum models downloaded from profiled IBM QPU hardware to provide a high-fidelity representation of the quality of the expected simulated results when compared to results from QPU hardware (Qiskit Ecosystem, 2025).

Quantum Gate Errors Model

A technique to mitigate gate readout errors enhances Zero-Noise Extrapolation (ZNE) by harnessing noise attributes on target QPUs hardware to fold circuits more efficiently. The ZNE technique achieves a 35% improvement on quantum simulators and a 31% improvement on QPU hardware. However, technology only mitigates QEC and has scaling challenges to QPU with more Qubits. Therefore, there is still a need to model gate errors to generate simulation results that are in family with expected results from QPU hardware (Hour et al., 2024).

Quantum Readout Errors

Quantum computers offer exponential computation advantages over some problem domains; however, a pervasive issue with current state-of-the-art QPU hardware is its proneness to readout errors (Paracha et al., 2024). One mitigation approach is in machine learning, where a training set is generated on a specific QPU hardware, and then transferred to a different QPU with a different readout noise profile. However, current technology cannot completely suppress noise presence and the need to model it in simulations is still required (Paracha et al., 2024). Other available options to mitigate quantum errors are based on ML, which could benefit from distributed processing (Melvin, 2022).

Thermal Relaxation Decoherence Noise Model

The current IBM QPU is based on superconducting Qubits, which rely on Josephson's tunnel junctions. The energy relaxation gap of small superconducting tunneled diodes generates thermal noise which affects the state and coherence duration of a Qubit. Approaches that rely on heat dissipation have been developed and implemented; however, the presence of thermal relaxation noise is still noticeable (Brunk & Lübbig, 1981).

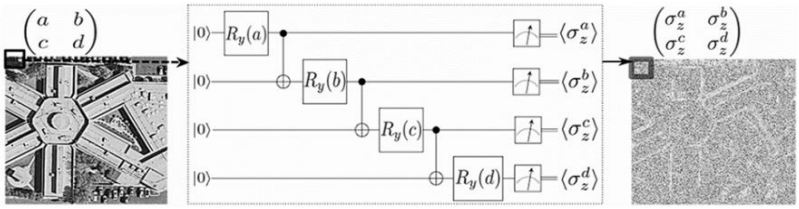
Scalability & Algorithm Partition Considerations

This section covers the impact of Qubit scalability on each of the algorithms, including the effects on performance. The instrument for distributed parallel framework developed for this research uses a pipeline architecture to allow for processing of complex data in stages, each breaking the problem into parallel processing units and passing the results to downstream stages (Gargees et al., 2020; Saxena et al., 2016). A pipeline architecture is how a processor executes mnemonic commands in parallel (Slide Serve, 2022). This mechanism in conjunction with data partitioning will provide each instance of a QC algorithm with a subset of the data to accelerate processing similar to how GPUs process large images in smaller tiles (Eilemann et al., 2009). For example, in the case of quantum image processing, each parallel 8x8 kernel was operating in its own QPU independently, thus maximizing parallel execution and expediting processing as shown in Figure 6 (Chalumuri et al., 2022). One consideration is that there may be data coupling, and algorithms will need to synchronize to resolve dependencies across parallel kernels (Bajpai et al., 2015). Another consideration is that a challenge with parallel distributed processing is scaling limitations that are only identified when large multiple instances of an algorithm are experimentally tested (Laguna et al., 2015).

In the case of QIPA, like Hadamard edge detection, the image was partitioned into tiles in QC to be processed in parallel across several QPUs. Some benefits of QIPA are that by using entanglement, parallelism, and superposition, pixels can be processed simultaneously and expedite image processing (Li et al., 2023). QIPA designs hold unique processing capabilities that outperform classical computers and have shown quantum supremacy in solving specific image processing problems (Shafique et al., 2024). The design of the quantum parallel

framework instrument was based on data partitioning techniques that minimize data coupling to avoid synchronization between QPUs to expedite processing.

Figure 6. Example QIPA Circuit *for 2x2 Kernel Function.*



Note. The image shows the application of a 2x2 kernel on the image starting in scan order at the upper left corner of the image and completing at the lower right corner of the image. From Chalumuri, A., Kune, R., Kannan, S., & Manoj, B. S. (2022). Quantum–Classical Image Processing for Scene Classification. IEEE Sensors Letters, Sensors Letters, IEEE, IEEE Sens. Lett, 6(6), 1–4. https://doi.org/10.1109/LSENS.2022.31732532.

QHED Algorithm Scalability

One consideration is that a larger number of Qubits could be used to increase the size of the kernel and process more pixels concurrently and reduce processing time (Chalumuri et al., 2022). A second consideration is to distribute image processing across many QPUs with each processing a smaller image tile like how GPU process tiles of a larger image (Eilemann et al., 2009). Another scaling option is to change the image encoding approach to increase or reduce color depth and expedite processing at the expense of pixel fidelity using the NEQR pixel encoding method (Mastriani, 2020).

QCNN Algorithm Scalability

A larger image kernel would allow processing of more pixels concurrently and hence reduce the overhead process synchronization between the Python interface codes and the QPU (Chalumuri et al., 2022).

QCCD Algorithm Scalability

The design of the QCCD analog to digital conversion model is based on a per pixel operation. An 8-bit adder circuit for summing two numbers will require 8 Qubits for each of two operands, for a total of 16 Qubits, plus 1 carry Qubit for a grand total of 17 Qubits. Hence, increasing the number of Qubits can benefit the pixel fidelity operation depth and increase the pixel dynamic range to manage more complex texture images (Mastriani, 2020). In addition, a scalability improvement would allow each QPU to process more than one pixel at a time, where the analog to digital conversion could be managed by several concurrent 8x8 kernels operating on the image hosted on the same QPU (Chalumuri et al., 2022). However, a limitation that must be considered is that the QPU simulator can only handle up to 30 Qubits.

QCKD Algorithm Salability

The operation of the QCKD model creates a rotating string of digits based on random drawings. Photons are polarized one at a time, creating a string of randomly rotated photons generated by the sender of the message. The receiver of the message will generate a similarly polarized, rotated string. The QCKD algorithm uses one Qubit to perform the rotation of the photon (Rasmusson & Barney, 2025). An improvement could be to parallelize the generation of the rotated string by processing more than one photon at a time (Kan et al., 2024).

Threaded Processing & Synchronization

The design of the QPF instruments to collect data for this research utilized Qiskit as the interface to the QPU (Koch et al., 2021). Each QC algorithm will be distributed to its own QPU in its own CPU core with multiple QC instances each working on their section of the data (Han et al., 2024). This capability provided the ability to model distributed processing using QPU simulators and mature QC algorithms before deploying to QPU hardware. Thread and

framework synchronization were achieved using inter process communications, semaphores, and shared memory instead of Qubit tunneling to facilitate debugging parallel programs (Lindsay et al., 2021; Laguna et al., 2015).

Review of Literature

A component of this literature review included trends of key technologies and applications in quantum computing including projection of arrival of quantum self-error correcting QPU's (Vasconcelos, 2020). The literature review starts with discussion of industry trends in the design and capabilities of CCD, which its pervasiveness will increase and drive complexities in image processing (Verified Market Research, 2024). The trends of larger image resolutions, deeper pixel color depth, and higher refresh rates are presenting challenges to image processing algorithms (Yan et al., 2023; Potma et al., 2021). These challenges are currently solved by distribution of image processing across multiple GPUs (Dong & Peng, 2023). However, HPC and GPU solutions have a cost including carbon generation. As a result, some applications can run more efficiently on a quantum solution (Yang et al., 2023).

The literature was reviewed for fundamental QC upon which more complex quantum algorithms can be built (Koch et al., 2019). For example, the application of the Hadamard gate to place a Qubit into superposition, and the application of the CNOT gate operation to place two Qubits into an entangled state (de Forges et al., 2024). The literature was reviewed to acquire a deeper understanding of the algorithms developed as experimental test cases integrated into the distributed quantum parallel framework instrument (Kan et al., 2024). The algorithms included in this review are QHED, QCNN, QCNN, QCCD, and QILQ. An important aspect reviewed was image Qubit encoding, and the various algorithms available including pros and cons (Mastriani, 2020; Dolciami, 2022). Synthetic quantum noise models were reviewed, which were integrated

with a simulated QPU to arrive at a predictive result that is representative of actual QPU hardware (Paracha et al., 2024).

Available quantum software frameworks were reviewed as well for their capability to be expanded to support distributed processing using multithreading across various CPU cores and across several QPU hardware (Dong et al., 2024). Quantum frameworks were evaluated for ample availability of QC algorithms examples that could be leveraged for this study including the five test case algorithms as shown in Figure 2. The literature was also reviewed for coverage of distributed and parallel processing architecture that facilitate the development and maturation of a distributed software framework instrument (Kan et al., 2024). Algorithms and CPU data locality and partitioning considerations were reviewed for reduction of data coupling to minimize synchronization points among multiple processing elements (Bai et al., 2025; Han et al., 2024). Finally, a section of finite state machines covers the architecture of how each of the processing threads managed the quantum interface as a finite state machine (Wang & Zhang, 2024; Kousiopoulos & Nikolaidis, 2024). State machine architecture offered and elegant and easy to modify transitions between processing threaded operations including synchronization, which facilitated development and integration of the distributed quantum parallel framework instrument.

Trends in Processing Complexity & Quantum Computing

Electro Optical CCD Camera Trends

In 2024, the CCD camera market was valued at 24.4 billion USD and is forecasted to reach 49.35 billion USD by the year 2031 with a growth of 6.9% Compound Annual Growth Rate) (CAGR). The growth of the market can be attributed to the increasing demand for high-resolution imagery in various applications such as manufacturing, security & surveillance,

medical & life sciences, and intelligent transportation systems based on Internet of Things (IoT) (Verified Market Research, 2024).

A modern camera CCD is characterized by having large image capabilities, such as the Sony α9 camera with 24.4-megapixel resolution per image (Yan et al., 2023). For example, some high-end CCD cameras will have higher display resolutions, pixel depth, and faster refresh frequencies up to 500Hz (Potma et al., 2021). Modern algorithms in some applications must be able to process a 4GB image in less than 16 milliseconds (one 60th of a second) to perform real-time object detection for applications that range from pattern recognition to modern self-driving vehicles (Potma et al., 2021).

Image Processing Trends

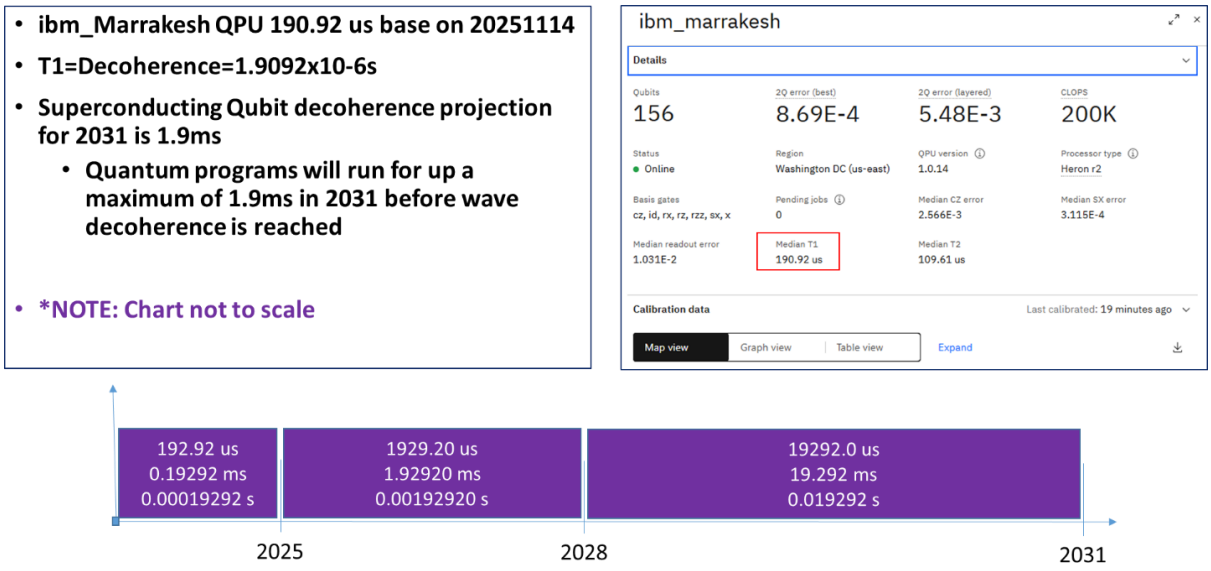
The trend in image processing has been on larger images, faster camera update rates, and image content with deeper pixel depths, particularly satellite imagery, including applications for science and surveillance (Yan et al., 2023; Potma et al., 2021; Verified Market Research, 2024). For example, satellite imagery presents challenges for signal process algorithms in terms of image data quantity and image refresh rate (Wang et al., 2022). Although the article is focused on satellite imagery, the concepts of processing and development of optimized methodologies for autonomous image collection and encryption are relevant. Therefore, exploration of alternate quantum computing solutions offers attractive options for large image processing (Chakraborty et al., 2022).

Quantum Computing Trends

A challenge in quantum computing is the lack of capability of processors to self-correct or compensate for quantum computing errors (Shafique et al., 2024; Paracha et al., 2024; Melvin, 2022). Much of the research in both quantum algorithm and hardware design is to achieve

quantum processors with an error free fidelity of 99.5 or better to be able to achieve the level of robustness necessary for mainstream commercial applications (Bravyi et al., 2022). Continued reduction of transistors every 12 to 24 months is yielding devices that are reaching the limitations of classical physics. Hence there is an interest in quantum technologies exploration to solve the physics limits encountered by classical processor designs (Reddy & K R., 2024). Quantum engineering is a field that continues to evolve focused on developing software and hardware to tackle error self-correction solutions and quantum processors with more Qubits (Vasconcelos, 2020). The industry is addressing QPU error limitations by developing advanced quantum processors designs reliant on superconducting Qubits. Also, another research that will enhance Qubits reliability is development of logical Qubits (more than one Qubit) to self-error correct and function as one, and surface codes for self-correcting-error scheme algorithms (Vasconcelos, 2020). Quantum photonics are additional areas of research that will directly allow for QPU interconnection over larger distances and support distributed processing (Heshmati & Emami, 2023). Research at Massachusetts Institute of Technology (MIT) Quanta group projects that quantum computing will be governed by Schoelkopf s Law, which states that quantum decoherence will be delayed by a factor of ten every three years for superconducting Qubits. In other words, decoherence maximum time increases by a factor of ten an order of magnitude every three years. Therefore, QC will run for longer periods of time and handle more complex algorithms (Vasconcelos, 2020). Figure 7 shows how the progression of superconducting Qubits maximum decoherence is projected to reach 0.019292 seconds by 2031, which means that quantum algorithms will run for longer and be able to tackle more complex problems.

Figure 7. *Superconducting Qubits Maximum Process Time Before Wave Decoherence.*



Note. Figure shows Schoelkopf s Law projections for superconducting Qubit maxim processing time before wave function coherence is lost. Graphics derived from the IBM Quantum Platform. (Accessed November 14, 2025). https://quantum.cloud.ibm.com/?computer=ibm_marrakesh

Trends show that as quantum computing reaches mainstream, so will the integration of quantum cryptography, including quantum key generation to harden existing systems based on RSA (Solar et al., 2024). Classical cryptographic algorithms, like RSA, are based on computational difficulty, in which a quantum Shor's based algorithm can render RSA vulnerable (Sykot et al., 2025). Applications of quantum cryptography will touch all domains including databases, optimization problems, data analytics, planning, medical research, and steganography security (Solar et al., 2024). A projection shows that within five to ten years quantum computing will be integrated with mainstream solutions to address problems including cryptography, quantum physics modeling, algorithm optimization, and image processing (Vasconcelos, 2020; Miceli & McGuigan, 2019; Kaufman et al., 2019).

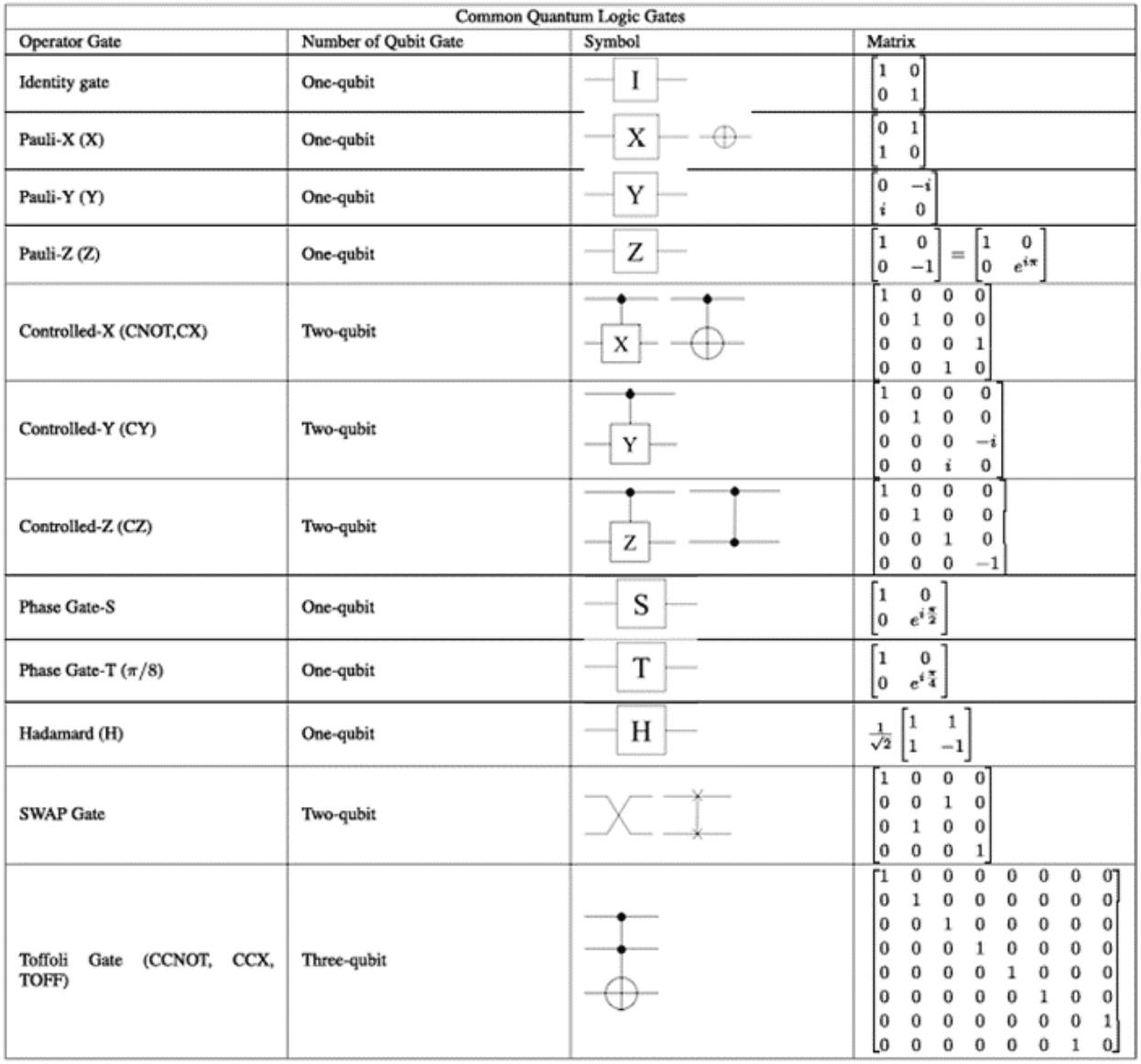
HPC Carbon Generation Concerns

A consideration with multi-GPU HPC solutions is the high carbon dioxide (CO_2) footprint. Current HPC projections, including usage of GPUs, are projected to grow according to the US International Trade Commission, from 1.2 trillion gigabytes in 2010 to a staggering 175 trillion gigabytes by 2025 and beyond. The estimates show that by 2030, datacenters and HPC systems may account for up to 8% of the worldwide emissions if no changes are made (Yang et al., 2023).

Quantum Circuits Fundamentals

The development of a quantum circuit program relies on basic functions applied to Qubits, which concludes with the measurement of the Qubit state at which point the superposition is collapsed, and the true state of the Qubit is revealed. Basic understanding of how quantum gates operate is fundamental to being able to develop quantum circuit algorithms (Shafique et al., 2024). Figure 8 summarizes the basic quantum gates used in all quantum circuit programs. A brief description of each fundamental gate is provided next.

Figure 8. *Fundamental Quantum Circuits.*



Common Quantum Logic Gates			
Operator Gate	Number of Qubit Gate	Symbol	Matrix
Identity gate	One-qubit	I	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Pauli-X (X)	One-qubit	X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	One-qubit	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	One-qubit	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{bmatrix}$
Controlled-X (CNOT,CX)	Two-qubit	X	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled-Y (CY)	Two-qubit	Y	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}$
Controlled-Z (CZ)	Two-qubit	Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Phase Gate-S	One-qubit	S	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix}$
Phase Gate-T ($\pi/8$)	One-qubit	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$
Hadamard (H)	One-qubit	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
SWAP Gate	Two-qubit		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli Gate (CCNOT, CCX, TOFF)	Three-qubit		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Note. The figure shows the fundamental building gates upon which more complex QC algorithms are built. From Shafique, M. A., Munir, A., & Latif, I. (2024). Quantum Computing: Circuits, Algorithms, and Applications. IEEE Access, Access, IEEE, 12, 22296–22314. https://doi.org/10.1109/ACCESS.2024.3362955

Identity Gate. The gate does not change the state of the Qubit. The primary use of the identity gate is to describe results from multi-Qubit circuits (Shafique et al., 2024).

Pauli X, Y, Z (NOT) Gates. The gate is the equivalent of a classical NOT gate and is useful to changing the phase and state of the Qubit for any of the three X, Y, and Z axes (Shafique et al., 2024).

Two Qubit Control Gates. The two gates comprise a control gate and the second gate with the function, where the first gate is used to control its operation. For example, a CNOT (Controlled NOT) circuit results of the second gate will be activated when the first gate reaches a state of one (Shafique et al., 2024).

Single Qubit T, S, Z Phase Shift Gates. The gates alter the phase of the Qubit without changing the amplitude of the state. There are three varieties of gates including the T type for changing phase, the S type for swapping two Qubits, and the Z type for applying a Pauli-Z operation. This gate can be used for constructive and destructive interference to favor or disfavor Qubit results (Shafique et al., 2024).

Single Qubit Hadamard Gate. The gate induces a superposition with an equal probability distribution for a Qubit (Shafique et al., 2024).

Two Qubit Swap Gate. The gate performs a swap operation between two Qubits (Shafique et al., 2024).

Three Qubit Toffoli Gate CCNOT Gate. The gate is also known as CCNOT Controlled-Controlled-NOT (CCNOT) gate with two control Qubits and one result Qubit. The third Qubit will be inverted if both the first and second control Qubits are one (Shafique et al., 2024).

Measurement Gate. Fundamental operation that extracts the result from the Qubit, which produces the collapse of a Qubit's wave function superposition state (Shafique et al., 2024).

Applicable Quantum Algorithms

Circa 1927 physicists realized that approximation of some of our physics models was incorrect. Classical mechanical models for particles were not effectively supported by experimentation results. The position and momentum of quantum particles could not be approximated (Peres, 2002). In addition, in experiments radiation waves behaved as if they were comprised of photon particles. The foundation of quantum computing is based on the Heisenberg uncertainty principle for describing matter at the quantum, where the location of a photon is described not by a point in space, but by a probabilistic quantum volumetric space (Peres, 2002). Qubits rely on the uncertainty of quantum to emulate superposition, which is the ability of a Qubit to be in multiple states concurrently until the point of measurement where the Qubit state settles (Zhu et al., 2024). Another foundational quantum physics concept is entanglement, where two or more quantum particles are associated and affect each other's state (Havlíček, et al., 2019). These foundational physics concepts are at the heart of the uncertainty of quantum algorithms and why QC algorithms need to be run multiple times to obtain a statistical representative outcome.

Quantum Image Encoding

Several image encoding algorithms were evaluated for storing the image in the QPU including the Flexible Representation of Quantum Images (FRQI), Novel Enhanced Quantum Representation (NEQR), and Quantum Boolean Image Processing (QBIP) (Mastriani, 2020; Munikote, 2024). Some quantum image encoding algorithms come with Quantum Probability Image Encoding (QPIE) errors. For example, FRQI and QBIP are non-exact image encoding algorithms, whether NEQR is an exact algorithm; however, quantum noise still affects pixel encoding (Dolciami, 2022).

Quantum Noise Modeling

The quantum error models integrated with the distributed parallel framework instrument increased the fidelity of the simulation by inclusion of gate noise, readout errors, and thermal decoherence relaxation noise. Error models were downloaded directly to the local simulator to use quantum noise models profiled from IBM QPU hardware. The results included modeled quantum noise, which provided a high-fidelity representation of the quality of the expected results when running on QPU hardware (Qiskit Ecosystem, 2021). Due to quantum errors, current quantum algorithms offer a statistical result based on a specified number of shots. For example, for image encoding five million shots were required to obtain credible statistical representative data providing a distribution of outcomes out of which the result was selected (Iyengar et al., 2020).

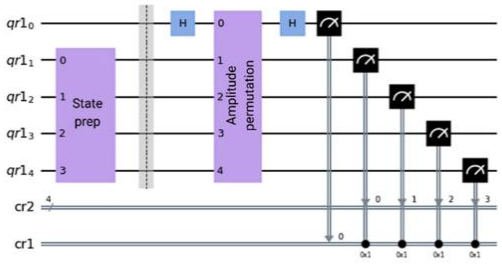
Quantum Hadamard Edge Detection Principles & Applications

An algorithm that is fundamental in image processing is line detection from where structures can be generated within the image, such as segments to identify image portions that may be of interest (Li et al., 2023). Line detection is the foundation for image segmentation, where the image can be analyzed for feature extraction, object detection, and tracking. The line detection algorithm filters pixels around the horizontal, vertical, +45°, and -45° directions. The algorithm uses a 3x3 neighbor kernel matrix for each of the four directions evaluated. With NEQR gray scale image representation algorithm, the time complexity of the quantum algorithm does not exceed a second order polynomial $\boldsymbol{O}(n^2 + q^2)$, where n represents a square image resolution in pixels in one axis and q the number of bits needed to represent binary gray scales. A second order polynomial offers an advantage over other quantum image encoding methods (Li et al., 2023).

In general, classical edge detection algorithms capitalize on the calculation of image gradients, i.e., identifying transitory pixels that change from low to high (or high to low) intensity values. Therefore, the worst-case time complexity for most of them is $\boldsymbol{O(n^2)}$. This implies that pixels must be computed individually to calculate the gradient. On the other hand, quantum edge detection algorithms, such as QSobel provide speedup of exponential complexity compared to classical edge detection algorithms (Ruan et al., 2021). However, there are extra steps to calculate pixel gradients, and copying the image adds inefficiencies to the QSobel algorithm. The Hadamard edge detector uses the property of the Hadamard-gate, and thus the QHED can achieve a time complexity of $\boldsymbol{O(1)}$ for algorithm execution not including state preparation and amplitude permutation, which is lower than $\boldsymbol{O(n^2)}$ incurred by the Sobel algorithm (Ruan et al., 2021). Therefore, the QHED algorithm yields an exponential speedup over classical edge detection and polynomial speedup over the QSobel algorithm (Yao et al., 2017).

The QHED encoder circuit uses the Quantum Probability Image encoding (QPIE) method to store the image (Qiskit Textbook, 2025). Figure 9 shows the encoder and QHED circuit for an 8x8 image. A QHED with 6 Qubits was selected to obtain the best performance over smaller or larger circuits. Although a smaller circuit comprised of 2 Qubits will yield the best quality results, a 6-Qubit circuit will provide a 230% performance improvement (Gultom & Amirullah, 2022).

Figure 9. *QPIE & QHED Circuit Algorithm for 5-Qubit 8x8 Pixel Kernel Image.*



Note. Figure for 5-Quibit pixel encoder and QHED QC algorithm. From Qiskitextbook2023. Various Authors. (Accessed August 6, 2025). Qiskit Textbook. Quantum edge detection. Github. https://github.com/Qiskit/textbook/blob/main/notebooks/ch-applications/quantum-edge-detection.ipynb

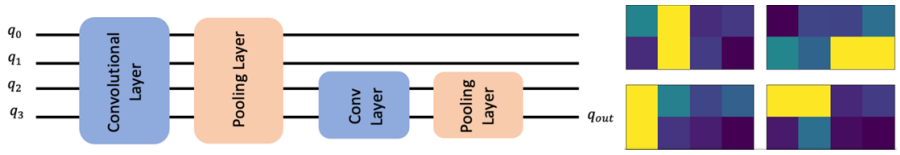
Quantum Convolutional Neural Network Principles & Applications

Convolutional Neural Networks (CNN) are adept at features and pattern recognition and object detection. Although there are variations of the architecture of CNN, the main processing phases of the CNN pipeline include the convolutional layer, where the image is convolved for feature extraction. Followed by the reshape phase, where feature images are reshaped and extracted. The next step is pooling where the feature images are organized followed by the final stage, where a classification prediction is made about the contents of the image as shown in Figure 10 (Cong et al., 2019; Zheng et al., 2025).

The first stage of a QCNN is to encode the image per one of several encoding methods (Mastriani, 2020; Munikote, 2024). The convolutional layer encodes the image feature vectors with neighbor nodes with their own feature vectors (Cong et al., 2019). The pooling layer serves as a hierarchical filter for features that are below the encoded thresholds (Wang et al., 2022). Image feature extraction is carried over by the convolutional and pooling layers. Quantum ML and QCNN offer a promising frontier for problems that are data bound and require significant

resources to process data. However, quantum hardware is still plagued by Noise Intermediate Scale Quantum (NISQ) that must be mitigated (Zhu et al., 2024; Melvin, 2022). An additional application of QCNN to mitigate current challenges with 3-Dimensional or higher data is with the integration of a multi-channel QCNN object detection algorithm (Roh et al., 2025).

Figure 10. *QCNN Circuit with Yellow Vertical and Horizontal Lines.*



Note. The figure shows the various stages of QCNN and horizontal and vertical pattern images on right. From Qiskit Ecosystem. Qiskit Machine Learning 0.8.2. (2025). The Quantum Convolutional Neural Network. Accessed Jan 6, 2025. https://qiskit-community.github.io/qiskit-machine-learning/tutorials/11_quantum_convolutional_neural_networks.html

CCD Sensor Concepts

A CCD is the sensor responsible for capturing image photon radiation, converting it to electrons, voltage, and finally to a digital signal (Konnik & Welsh, 2014). During the processing of the radiation received at the front end of a camera sensor, several noise factors affect the image including temporal noise, and dark current caused by heating of the CCD elements (Konnik & Welsh, 2014). CCD quantum modeling by this research will cover the last phase of the CCD model, including the analog voltage to digital signal conversion, as shown in Figure 11.

The process of sampling the image is not perfect, and some of the photon energy captured at each pixel photodetector is affected by various noise factors. Imperfections in the photodetector lead to the generation of noise in the measured signal (Ilari, 1990). There are several sources of noise spread across the components of the CCD sensor, including the main source of noise from dark current generated by the silicon substrate. An observation is that dark current noise increases with temperature and longer integration times (Sentenac et al., 2003). The

CCD models used for this research contain four top-level models including photon model, electron model, voltage model, and analog to digital model, each comprising finer detailed sub-models (Konnik & Welsh, 2014). A simplification of the CCD model will not include cryogenic cooling, which relies on a stream of cold liquid nitrogen across the camera sensor surface to improve detection for objects at longer ranges (Ortiz & Oliver, 2004). The rationale for selecting a CCD instead of a CMOS is because of the widespread use of CCDs over CMOS sensors. A CCD is founded on technology that has been well known and proven since the 1970's (Ortiz & Oliver, 2004). An example of advantages of CCD over CMOS is that a CCD device sample photodetector fill factor is 100% efficient versus 75% for CMOS (Gambheer & Bhat, 2023). CCDs are easier to model due to the high linearity of the device. Although a CCD generated image contains more noise than a CMOS image, the noise effects can be compensated or undone by signal processing algorithms (Ortiz & Oliver, 2004).

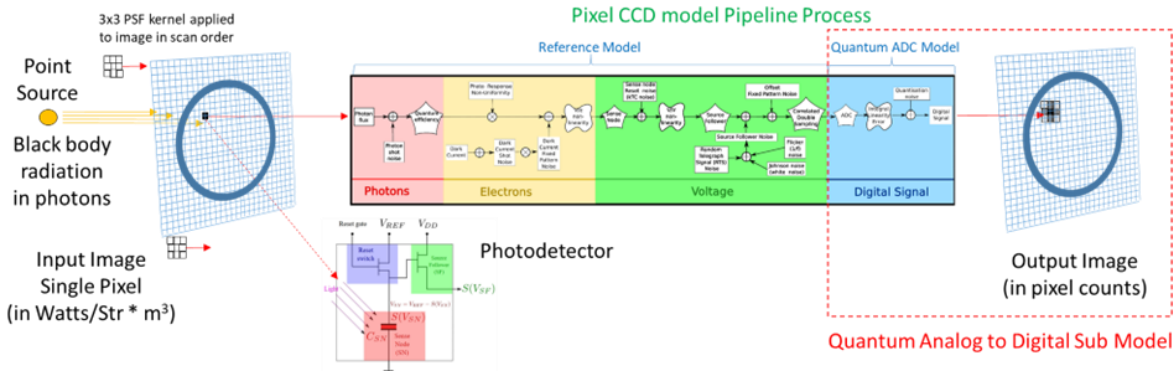
Quantum CCD Model

A CCD is responsible for collecting and digitizing an image for signal processing algorithms, which will perform object detection, discrimination, and classification. The QCCD case study of this research implemented the model for conversion of analog voltage to digital image (Konnik & Welsh, 2014). Current QPUs provide basic functions on Qubits and algorithms and are not designed to store substantial amounts of data. Therefore, the input and output images will be stored on the classical computer. Only a portion of the image to be processed in the QPU will be transferred and encoded in Qubits (Dolciami, 2022). A QPU does not have an Arithmetic Logic Unit (ALU); therefore, circuits for floating point arithmetic were reviewed for integration into the QCCD model pipeline. However, the designs yield noisy results that are not reliable and

limited to a maximum of 30 Qubits (limitation imposed by Qiskit quantum simulator) (Seidel et al., 2021).

Quantum programs are characterized by operating as a pipeline comprised of quantum circuits acting on Qubits (Balamurugan et al., 2024). In a similar fashion, the QCCD model takes the voltage read at each pixel and converts the pixel value to pixel digital counts ready for image processing as shown in Figure 11.

Figure 11. *The QCCD Model Converts Analog Signals to Digital Pixels.*



Note. CCD process pipeline stages. Image derived from Konnik, M., & Welsh, J. (2014). High-level numerical simulations of noise in CCD and CMOS photodetectors: review and tutorial.

Quantum Cryptography

Quantum cryptography principles are based on quantum particle attributes, which include the uncertainty of location of particles where they can simultaneously exist in more than one place at a time (Peres, 2002). Another important principle in cryptography is that photons are randomly generated in one or two quantum states. Therefore, a photon state cannot be measured without disturbing it, and certain aspects of a particle can be cloned, but not the entire particle (Rasmusson & Barney, 2025).

QCKD relies on the principles of quantum particles and is a method of securely exchanging cryptographic keys for encoding binary messages using the principles of quantum

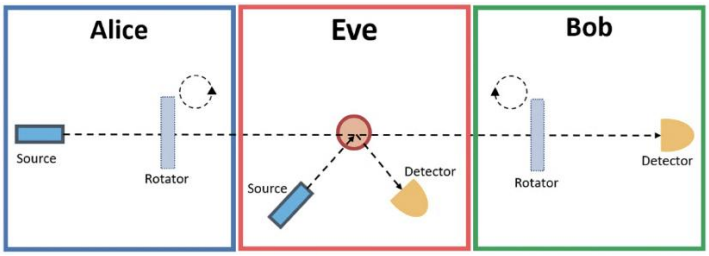
mechanics. The methodology relies on the inherent properties of quantum systems to detect eavesdropping, ensuring the security of the exchanged key. The action of measuring a Qubit collapses the state and makes it impossible to replicate. Therefore, QCKD offers unconditional security based on fundamental physical laws, not on the computational complexity of algorithms (Radanliev, 2024).

According to QuantumXChange, (2025), the methodology of generating quantum encrypted keys can be described as follows. Step 1. Sender transmits photons across a polarizer filter, which randomly applies one of four polarizations with a bit designation including Vertical (one bit), Horizontal (zero bit), 45 degrees right (one bit) and 45 degrees left (zero bit). Step 2. Photons travel to a receiver, which uses two beam splitters applied to the photon stream for horizontal-vertical and diagonal property extraction to read each photon polarization. An interesting fact is that the receiver does not know which beam splitter to use for each photon, and a guess must be made to select the beam splitter. Step 3. Upon photon stream transmittal and reception, the receiver tells the sender which beam splitter was randomly selected for each of the photons sent in the sequence. The sender then compares the information with the sequence of the polarizers used to send the key. The photons read with the incorrect beam splitter are discarded, and the resulting sequence becomes the cryptographic key. If during the key generation process, the photons are copied in any form by an eavesdropper, the state of the photon will be changed and detected by the end points. In other words, a photon cannot be read and then forwarded or copied to maintain its initial state (QuantumXChange, 2025).

For example, as shown in Figure 12 Alice in the transmitter encodes the message before sending it to Bob as the recipient. If an eavesdropper, named Eve, tries to listen in on the conversation, she must read each photon to read the secret. Eve must then copy and pass the

photon on to Bob, the message recipient (Rasmusson & Barney, 2025). The action of reading the photon makes Eve alter the photon's quantum state, which will introduce errors into the quantum key (Radanliev, 2024). The errors introduced by Eve reading the key will alert Alice and Bob that someone is eavesdropping, and the key has been compromised. Figure 12 shows a graphical representation of the encryption key generation process using photons, rotating polarizers, and beam splitters.

Figure 12. *QCKD Key Encoding Transmittal and Reception.*



Note. QCKD process representation. From Rasmusson, A.J., Barney, R. (April 19, 2025). Quantum Cryptography: Quantum Key Distribution. https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/awards/teach_me_qiskit_2018/quantum_cryptography_qkd/Quantum_Cryptography2.ipynb

An application reviewed as part of this literature includes the integration of quantum cryptography based on Rivest Shamir-Adleman (RSA) (Solar et al., 2024). The literature shows that algorithms integrated into current solutions must meet performance requirements to operate seamlessly with current deployed cryptographic solutions (Ahilan & Jeyam, 2022).

Another application of quantum cryptography using the key distribution approach is used in stenography, where information is hidden and spread among the pixel information of an image. For example, a secret image can be encrypted in a cover image, where the front cover image is only visible, and the encrypted image information is hidden and overlaid in the image pixels with only a successful decryption revealing the secret encoded image. Current

implementation of cryptography still relies on a hybrid solution of classical algorithms and quantum algorithms to harden a system (Sykot et al., 2024). This research demonstrated how cryptography using a QCKD algorithm can be distributed across multiple QPUs with processing time expedited and integrated with existing cryptographic solutions.

Quantum Interlin-q Framework

Interlin-q is a framework that offers distributed processing like the instrument that will be developed for this research; however, Interlin-q has a limitation of only being able to spawn a maximum of two Qubit QC algorithms (Parekh et al., 2021). Another limitation is that Interlin-q does not model quantum noise natively, which is required to provide realistic quantum simulated results (Yuan-Cheng et al., 2024). Other frameworks were evaluated, such as Cirq and Interlin-q, which provide an upper layer of abstraction and quantum circuits have to be ported to these platforms rather than integrated directly like Qiskit (Evans, 2021). This adds an extra layer of software complexity, which disqualified these frameworks to be used as the main QPU interface. In addition, Interlin-q has scaled limitations to two Qubits and cannot model larger QC circuit networks (Ferrari & Amoretti, 2024).

Quantum Software Frameworks

Several quantum development environments were reviewed as candidates for integration into a closed-loop simulation including Ocen, Cirq, Qiskit, Interlin-q, QMASM, Silq, PennyLane, and Project Q (Balamurugan et al., 2024). Although IBM's Qiskit framework is written with a python interface, it can still be incorporated as a module in a simulation developed in C++. The software framework architecture developed leveraged current technology and harnessed available solutions so that the focus could be kept on building the distributed software and not in re-developing algorithms for which there is already a solution (Koch et al., 2019).

Quantum Circuit Simulator. A quantum simulator was essential to the development of the distributed quantum parallel framework instrument to reduce costs since IBM charges $96 per minute of QPU quantum process time (IBM Pricing, 2025). Fortunately, the Qiskit SDK provides natively a QPU simulated environment in addition to the actual QPU hardware interface (Qiskit Ecosystem, 2021). The ability to run QC in a simulated environment allows for maturation and debugging of programs quicker in a controlled environment. A quantum simulator allowed for the development of quantum programs with a larger number of Qubits available on QPU by using distribution. One observation is that the quantum simulator may return results that are different from the ones produced by an actual QPU due to noise modeling fidelity (Shafique et al., 2024).

Several papers were evaluated for multi layered schedulers and parallel processing to use as concepts for the design, and integration of a distributed quantum simulation framework for this research (Fernandez-Conde et al., 2022; Zhou & Preindl, 2023). The design decision to develop a framework instead of integrating an existing one was based on two assumptions. One, a developed framework can be tailored specifically for this research. Second, a third-party framework may have to be purchased, and permission may be required to make use of the software, which may not be easily modifiable for integration (Ferrari & Amoretti, 2024). Algorithms operated as a pipeline and this characteristic was used by the distributed framework to parallelize processing. This attribute processed QC algorithms by treating them independently from other executing algorithms (Balamurugan et al., 2024).

Quantum Distributed and Parallel Processing

This research included GST principles for the design, development, and integration of a distributed quantum parallel framework comprised of interconnected subsystems to execute the

experiment and generate the required data (Johnson, 2019). The distributed quantum parallel framework provided the capability to seamlessly execute QC algorithm in simulators, or QPU hardware. The literature research provided a handful of quantum frameworks as listed in order of popularity including Qiskit, QML, Q#, D-Wave Ocean, OpenQASM, Cirq, Bracket AWS, Pennylane, Forest SDK, QCL, Strawberry Fields and Pytket. However, the selected foundation for this research expanded on Qiskit as it offers many quantum application examples including cloud interfaces to IBM QPU hardware (Jimenez-Navajas et al., 2024; IBM Pricing, 2025).

There were two synchronization alternatives considered for this research, one is at the CPU using shared memory and the other is using quantum entanglement. An innovative approach reviewed for distributed Qubit synchronization uses quantum link microwaves for quantum communications for entangled Qubits (Rached et al., 2025). Other quantum parallel processing frameworks, like Interln-q also offer process QPU synchronization using entanglement, which theoretically can offer a faster response time instead of shared memory at the CPU. However, the Interln-q framework is limited to maximum of two QPUs and has not been maintained since 2020 (Parekh et al., 2021). Another framework is Qiskit, which uses Open Multi-Processing (OpenMP); however, there are no primitives to synchronize threaded processing. The qTask framework supports quantum simulation only and does not interface to QPU hardware, nor does it provide primitives to optimize simulations for CPU cores and thread priorities (Huang, 2023). Another practical option is the TriQXNet framework, which is designed to model solar wind Disturbance Storm Time (DST) to estimate damage to infrastructure and Global Position Satellite (GPS); however, the design is not generalizable to other QC algorithms (Jahin et al., 2024).

The selected framework to develop the instruments for this research was Qiskit due to the ample material available including videos, tutorials, and proven QC algorithms and availability of IBM cloud QPUs. In addition, a factor considered was IBM, the developer of Qiskit, a company that has been working on quantum computing solutions for years (Qiskit Ecosystem, 2021; Koch et al., 2020; Jimenez-Navajas et al., 2024). This research builds and extends Qiskit capabilities to support distributed processing seamlessly using simulated QPU or actual QPU hardware (Kan et al., 2024).

Finite State Machines

The quantum distributed parallel framework design for execution of each quantum algorithm uses a pipeline processing technique for parallel processing similar to how a CPU executes the mnemonics command pipeline (Saxena et al., 2016). In addition, processing across the pipeline was broken down into states to facilitate task decomposition into simpler logic governed by state transitions (Wang & Zhang, 2024). All threads comprising the distributed quantum parallel framework operate using a common set of transitions states to facilitate distributed threaded synchronization (Babakov & Barkalov, 2025). For example, there is a state for frame start, frame process, frame assembly, frame transfer, frame complete, and no state (Salcedo & Ahmed, 2025). The finite state machine was instrumented with a common architecture to evaluate the performance for all threads regardless of the circuit models executing (Kousiopoulos & Nikolaidis, 2024).

Summary

This chapter started with the literature review strategies including search engines used to collect, compile, and categorize the articles to support this research. The theoretical framework was established including quantum computing physics foundations and quantum computing

challenges to address technology limitations, hardware challenges in building QPUs with large volumes of Qubits, and over reliance on quantum simulation. Scalability considerations were explored for the framework and the five QC test case algorithms to be developed for this research. The theoretical framework section concluded with thread processing and synchronization concepts that are at the core of the framework instrument, and ensure a reliable operation of the threaded software that drove the data collection for this research.

The literature review showed that trends in CCD photo sensor design point to larger resolutions, more colors, and faster refresh rates, supporting the idea of alternate image processing solutions for GPU and HPC technologies. Along these lines, classical HPC processor studies show that if there is no change of course in processor design and power consumption, the projection is that 8% of carbon emissions will be due to HPC by 2030. Quantum technologies offer an option to reduce carbon emissions for some applications, and this research presented the opportunity to explore quantum algorithms that may be more efficient than classical HPC solutions.

Quantum hardware design trends were reviewed for challenges in superconducting quantum processors, quantum photonics, magnetic resonance, and trapped ions. These technologies are still in their infancy, and results are plagued by noise or by quantum wave de-coherence caused by temperature or radiation, which results in invalid outcomes. In addition, current QPU technology has not produced reliable quantum processors with more than 1,000 Qubits, which is estimated to be the minimum required Qubits to achieve self-error correction. Another challenge is networking of various QPU located across the same building or separated by large distances. Despite challenges, quantum computing keeps moving forward with breakthroughs as technology matures.

This literature review also provided the theoretical framework to obtain the knowledge to develop a software instrument for hosting distributed processing. The instrument was used to support performance evaluation of QC algorithms for which more Qubits are required than available in a QPU. Various QIPA were reviewed as good candidates to be integrated including charge collection devices, cryptographic algorithms, convolutional neural networks, and line edge detection. Quantum processing is not perfect, and the technology is still in the maturation phase, and some QPUs are more dependable than others, which affects the results. More specifically, the chapter includes a literature review that describes the methodology and mathematical theory for developing several quantum QHED, QCNN, QCCD, QCKD, and QILQ circuit models. These five QC test case models were used to develop the fundamental practical applications used to evaluate the feasibility and robustness of distributed quantum algorithms when integrated into a closed loop solution.

Distributed processing literature was reviewed to evaluate the landscape of available work that could be leveraged for this study. There is plenty of literature for theory and development of QC models and quantum circuit software architecture. However, there is no information for how multi-threaded layered simulations can be distributed, parallelized, and integrated to improve performance. Also, there are no frameworks to provide a viable solution to closed-loop applications requiring more Qubits than available on a single QPU. This research provided a novel framework for development and integration of distributed parallel quantum algorithms, which can be hosted on a simulated QPU prior to deployment to actual QPU hardware. This research fills a gap in literature that will extend further studies in quantum algorithm design, development, and integration. Finite state machine material and pipeline processing articles were reviewed to help partition and modularize the quantum parallel

framework design. The finite state machine concept facilitated software design and maintenance, and pipeline concepts were used to develop the parallel framework.

In conclusion, this research served as the foundation for distributed processing, parallel quantum model performance studies, and the impact of noise on algorithm performance as quantum computing technologies mature, become more dependable, and reach mainstream deployment. The literature research laid out the foundations for experiment instruments development that generated data to support the hypotheses and answer the research questions.

Chapter 3: Research Method

The problem addressed in this study was that a single QPU cannot solve distributed quantum computing applications that require more Qubits than available on a QPU, which limits the scalability and fidelity of image processing, cryptography, neural networks, error correction, and physics modeling (Davis et al., 2024; van Dijk et al., 2019; Davis et al., 2023). According to Ichikawa et al. (2023), in 2022 condensed physics applications were using more than 100 Qubits, which are approaching Amazon 105 Qubit QPU, and IBM 127 Qubit QPU (AbuGhanem, 2025). Additionally, the pervasiveness of noise effects aggravates as the complexity and depth of a Quantum Circuit (QC) increases, which makes a case to partition an algorithm into smaller shallow QCs and distribute processing across several QPUs (Kan et al., 2024). Distributed quantum algorithm processing is needed to develop applications to solve large scale QC algorithms, including decomposition into smaller problems for complex solutions that require more Qubits than available on a single QPU (Davis et al., 2023; Kan et al., 2024). Therefore, to address these literature gaps, this study developed a novel software framework instrument to support research of QC distributed processing through experimentation.

The purpose of this positivistic worldview quantitative constructive research was to build and validate a novel distributed parallel processing framework for quantum computing. More specifically, this research developed a framework instrument to experimentally evaluate performance of distributed parallel QC algorithms under the impact of quantum noise and provide QPU representative results. This study aimed at modeling distributed QC algorithm execution to expedite development, and maturation through experimentation to find which combination (of independent variables) including number of QPUs, Non-Uniform Memory Allocation (NUMA) strategy, CPU core affinity & priority, and OS scheduling offers the best

performance (for dependent variables) including response time, Euclidian distance, probability of classification, cryptographic Key Generation Rate (KGR), and probability of tamper detection. The selection of test case QC algorithms covered several domains for distributed processing applications including a Quantum Hadamard Edge Detector (QHED), Quantum Convolutional Neural Network (QCNN), Quantum Charge Coupled Device (QCCD), Quantum Crypto Key Distribution (QCKD), and Quantum Interlin-q (QILQ) for framework performance reference (Balamurugan et al., 2024; Shafique et al., 2024).

The rest of this chapter is organized as follows to provide more detail into the research methodology, the experiment population data, and sample size. A detailed breakdown of the various data collection sets is provided. The materials of instrumentation used to generate and collect data are discussed, including the Quantum Parallel Framework (QPF) software to perform the experiment. A brief description of the pilot test of the feasibility and validity of the QPF instruments is addressed. Dependent and independent variables are described as well as the purpose of their design. The study procedures' methodology is explained in more detail too. In addition, the data analysis approach to test hypotheses and answer research questions using the five QC test cases are explained. Finally, Measures of Effectiveness (MoE) metrics for each of the five QC test cases are explained as well as experiment limitations and delimitations.

Research Methodology and Design

Constructive research requires the development of instruments for a specific domain to collect data (Crnkovic, 2010; Lassenius et al., 2001). A quantitative study can be performed using experimental, non-experimental, correlational, and descriptive frameworks to develop hypotheses that may help reveal numerical patterns from the data analysis (Bloomfield & Fisher, 2019; Creswell & Creswell, 2017). However, this constructive research focused on

experimentally exploring the distribution of QC algorithms over several QPUs and the cause and effects on feasibility and performance of the system. This true experimental research, unlike correlational study, generated data to test the impact of dependent variables affecting distributed performance and data quality (Gilleland, 2030; Creswell & Creswell, 2017). A Quantum Parallel Framework (QPF) instrument designed and developed specifically for this research to generate the experiment data required to test hypotheses, research questions and address the research problem (Reubens, 2016). The research also required the use of pre generated reference datasets to use in comparisons with experimental data to evaluate causal effects of independent variables (Creswell & Creswell, 2017).

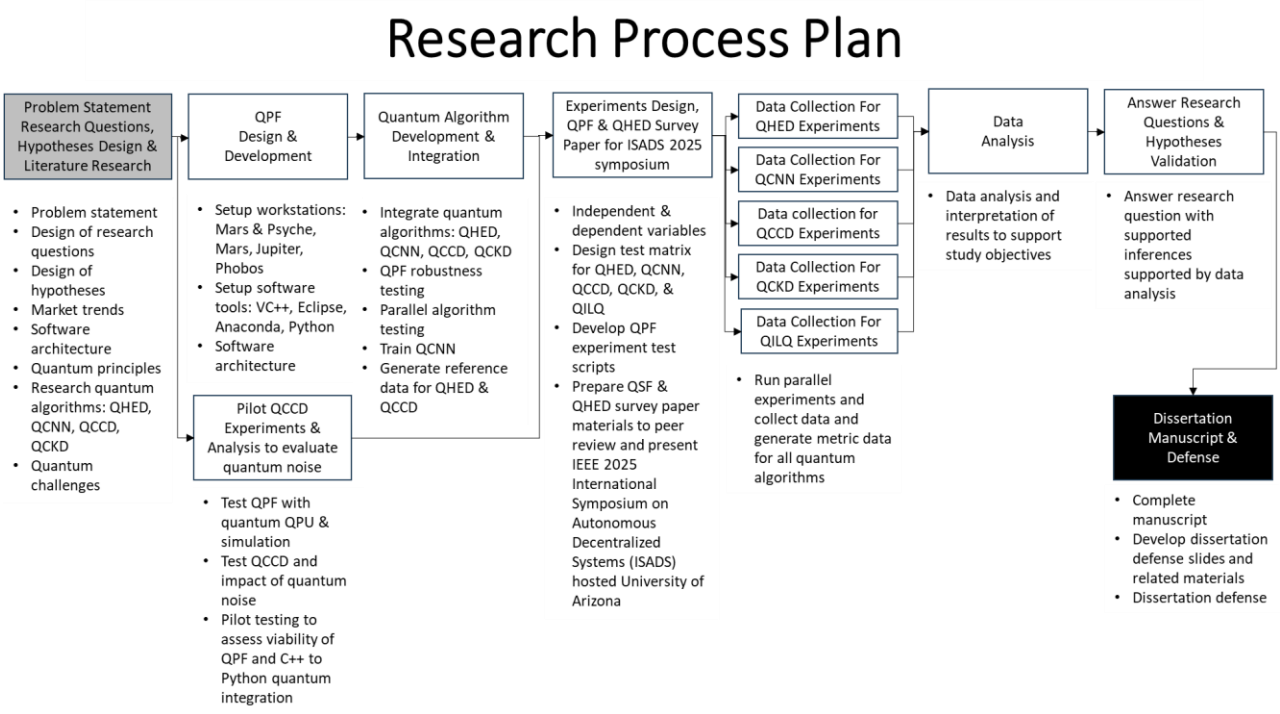
The design of the study is aligned with the problem statement to answer each of the three-research questions for framework feasibility, distributed QC algorithm performance, and effects of quantum noise on algorithm results. Experimental data collection was driven by a closed loop distributed QPF, which synchronized all QPU processing for hosting the QHED, QCNN, QCCD, QCKD, and QILQ circuit models. The QPF framework distributed work across all QPU elements including simulators, synchronized frame completion, and calculated metrics for MoE for all five QC test case experiments.

Research Study Process Plan

The study process plan begins with the problem statement from the associated literature research. The study was then followed by research to design and develop the distributed QPF software instruments. Based on literature research regarding quantum noise prevalence, an initial pilot experiment was designed to test the QPF instrument interface to QPU hardware and perform initial assessments on generated data containing quantum noise. In addition, the pilot experiment was designed to determine the feasibility and reliability of the QPF data collection

instrument. The next phases included the design and integration of QC algorithms models. A test matrix was designed to account for all independent and dependent variables to drive experiments for all QC algorithms. A test matrix was designed to address the problem statement, research questions, and hypotheses. Data collection experiments followed the integration of all QC algorithms. Data analysis and interpretation of results took place next to build the supporting data required to test the hypotheses. The next phase of the process was to interpret the results of the data analysis and provide inferences required to answer the research questions. Figure 13 shows this research study in chronological order, starting from left to right and culminating in the thesis manuscript completion and dissertation defense.

Figure 13. *Research Process Plan with Detailed Stages Culminating in Dissertation Defense.*



Population and Sample

The data distribution was a parametric normal distribution since more than one hundred samples were collected (Sil et al., 2019). Comparison of reference datasets to quantum synthetic datasets was performed using statistical inference based on metrics described in this chapter (Biswas & Shlizerman, 2022). A two t-test approach was used to compare baseline data to experiment collected data to evaluate if expected results improve or degrade. A value of $\alpha <$ 0.05 was used as the cut-off for statistical significance to reject Null hypotheses. The α-value demonstrates that there are less than 1 in 20 chances of results occurring by chance (Hazra & Gogtay, 2016). Achieving $\alpha < 0.05$ required a sample size of at least 14,428. The power sample size analysis was performed with the GPower tool.

There were seventeen synthetic datasets generated to support the analysis and derive conclusions, which are reconciled as follows. There were eight datasets required by experiments for data collection for QHED and QCCD. In addition, nine experiment datasets were required for the QCNN, QCKD, and QILQ algorithm models as shown in Figure 14. All research datasets were unpaired as the results of one dataset did not affect the other (Sil et al., 2019). The determination of datasets was derived from the data size combinations for each of the five QC test cases. The experiment used a random sampling method for the quantum data generated results as quantum noise tends to be random (Dolciami, 2022). Algorithm performance timing used random samples since the OS scheduling timing and task context switch are non-deterministic (Fernandez-Conde et al., 2022).

Figure 14. *Test Matrix with Samples per Algorithm to Achieve Statistical Significance.*

Name of Instrument(s) Image Resolution OR Data Matrix Size Algorithm	Rows	X	Cols	Minimum Required Sample Size (Based on G*Power or other Analysis)	Sample Size (# Participants Responded)	Complete Data Set(s) (# Participants with Complete Dat QPF Frames Required to achieve minimum required sample size
QHED	32		32	14,428	14432	902 frames
QCNN	4		8	14,428	14432	451 frames
QCCD	32		32	14,428	15360	15 frames
QCKD	4		4	14,428	14432	902 frames
QHED	512		512	14,428	16384	1 frame
QCNN	64		128	14,428	16384	2 frames
QCCD	64		64	14,428	16384	4 frames
QCKD	64		64	14,428	16384	4 frames
QHED	1024		1024	14,428	16384	1 frame
QCNN	1024		1024	14,428	32768	1 frame
QCCD	128		128	14,428	16384	2 frames
QCKD	128		128	14,428	16384	1 frame
QHED	2048		2048	14,428	65536	1 frame
QCNN	256		512	14,428	131072	1 frame
QCCD	256		256	14,428	65536	2 frames
QCKD	256		256	14,428	65536	1 frame
QILQ	32		32	14,428	14428	25 frames

Note. Column labeled ‘Sample Size’ contains the total number of samples collected for each QC algorithm. Five test cases yielded a total of 17 datasets.

Materials of Instrumentation

The primary data collection instrument was the QPF, a multithreaded framework, which hosts all QC algorithms in QPU simulators. Also, the QPF includes an OpenGL display GUI as shown in Figure 15. An initial pilot experiment test was performed on the QPF framework to evaluate reliability and to assess the impact of quantum noise on validity of the QPF instrument. The pilot test included interfacing to a cloud QPU and running a QC in actual hardware. The model selected for the initial noise assessment was the QCCD. The resulting quantum generated noisy image was compared to the classical reference image without noise to quantify the impact. The pilot test was able to complete successful execution on IBM Kyiv, a cloud QPU. The reliability and validity tests included running 30 pixels of the Analog to Digital conversion QCCD algorithm, with each pixel executing 14,428 shots to obtain a statistical representative

value for the analog voltage to digital (AtoD) image conversion. The reliability and validity run reconciled as 14,428 shots times 30 pixels, giving 432,840 runs shots, which performed the AtoD radiometric conversion without crashing the QPF instrument.

The design of the QPF instrument is based on a hierarchical modular scheduling model (Fernandez-Conde et al., 2022). When the QPF is scheduling QC algorithms to be run on actual QPU hardware, then the QPF will host and manage the threaded interfaces for each QPU. The QPF also includes the instrumentation to collect the data required as shown in Figure 15. The QPF collects experimental data including per frame performance timing, per pixel generated Euclidian distance metrics, image classification prediction, KGR, and cryptographic key tamper detection. Additional instruments used were common tools, such as Qiskit, IBM SPSS, Rapid Miner Studio, gnuplot, and Excel.

The languages selected for development of the instrument were C/C++ and Python. The development included two hierarchical executives. The primary main executive manages the top-level simulation and GUI display. The secondary executive coordinates QPU thread interface scheduling & synchronization, and data collection. The GUI is founded on two cross platform portable libraries including GLUT (Graphics Library Utility Toolkit, 2024) version 3.4.0 for managing window rendering primitives, and GLUI (Graphics Library User Interface, 2024) version 2.36 for user controls. The software was developed for cross platforms using Microsoft Windows 10 and Linux Mint 22. However, the platform for running the research experiments was Linux Mint 22 using Z840 Hewlett Packard (HP) 2 Xeon E5-2670 @ 2.60GHz processors with 16-cores and 128 GB RAM. Linux GCC version 14 was used for development and debugging. A secondary environment used for integration was Microsoft Visual Studio version 2022 to support seamless portable development across platforms. The following software

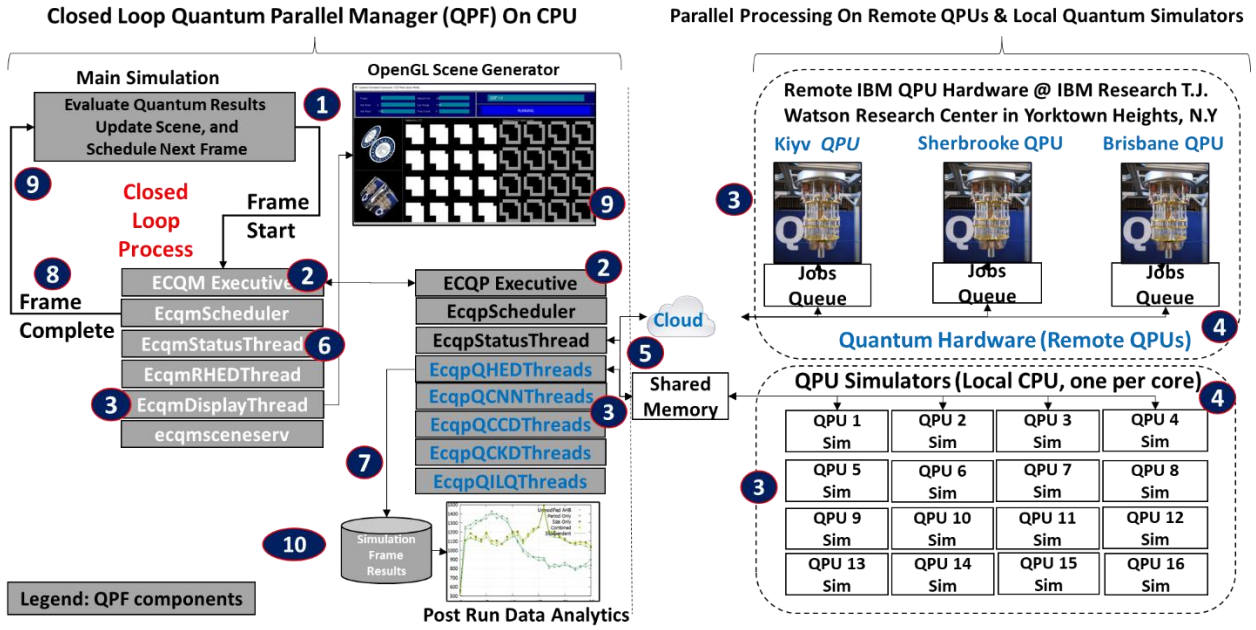
packages used are common to both Windows and Linux including Eclipse version 2021, Qiskit version 1.2.4, Spyder version 5.5.1, and Python 3.7. Rapid Miner (AI Studio) version 2026.0.2 was used for data analysis.

The criteria for selecting C/C++ as the main simulation programming language was the efficiency and performance of compiled binary code. C/C++ produces the least amount of machine language instructions when compared to other languages (Farooq et al., 2014). The design distributes processing across many QPUs, which can be optimized in C/C++ and Python with priorities and CPU core affinities to support optimized applications (Abeni et al., 2019). The primary Executive Controller for Quantum Management (ECQM) is written in C/C++. However, the interface to the quantum computer is managed by a secondary Executive Controller for Quantum Processing (ECQP). The ECQP is written in Python for seamless integration with IBM's Qiskit quantum programming SDK framework as shown in Figure 15.

The Python development environment is Anaconda distribution version 3.0. Anaconda is used to help manage installation of various Qiskit package configurations, and to capitalize on Anaconda tools like Spyder IDE and Jupyter Notebook. Figure 15 shows a system decomposition of the two QPF executives managing the execution of QC algorithms. Processing of the QPF framework is divided into two main processing subsystems, including classical CPU and QPU processing. On the QPU hardware side, the project uses cloud IBM quantum processing hardware ranging from QPUs with 127 to 156 Qubits. Three IBM cloud QPUs are available including IBM Kyiv, IBM Sherbrooke, and IBM Brisbane. The QPF can assign specific QPU for processing or select the QPU with less processing load. When running in a QPU simulator, all instances of the simulated QPUs are hosted locally on the workstation's CPU cores (with one QPU per core to maximize parallel performance). The structure of the data

collection files is comprised of fixed width data records in ASCII text format to facilitate analysis using tools like Excel, gnuplot, and Rapid Miner Studio. A console log is also included to record the processing events of both executives. Output images for the QHED, QCNN, and QCCD models are stored in binary format comprised of 16-bit unsigned integers.

Figure 15. *General System Theory and QPF Instrument Decomposition & Processing Order.*

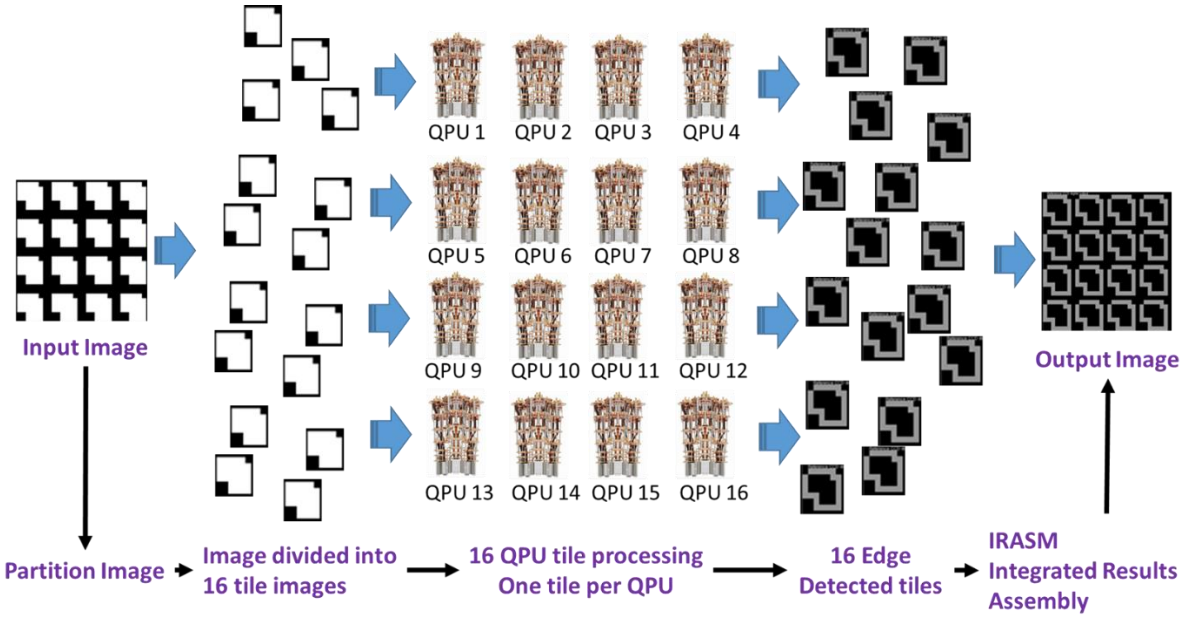


Note. The figure shows all five QC algorithms hosted by the ECQP executive; however, typically only one type of algorithm is exercised for each QC test case.

This methodology describes how the QPF partitions, distributes, process, and assembles results for each of the QC test cases. The process used by the QPF partitions the input data into smaller equally sized tiles (data subsets) and distributes each of the tiles one per QPU. Each QPU process its corresponding tile in parallel with the other QPUs. The final stage includes Integrated Results Assembly (IRASM) of the tiles into a composite image as shown on Figure 16 (Bai et al. 2025; Han & Wang, 2024). The approach was derived from techniques used to parallelize processing across multiple GPUs. This methodology is similar to how a large image is

partitioned into smaller portions distributed across several GPUs to expedite processing (Eilemann et al., 2009).

Figure 16. *QPF Process Pipeline for 32x32 Edge Detection Distributed Over 16 QPUs.*

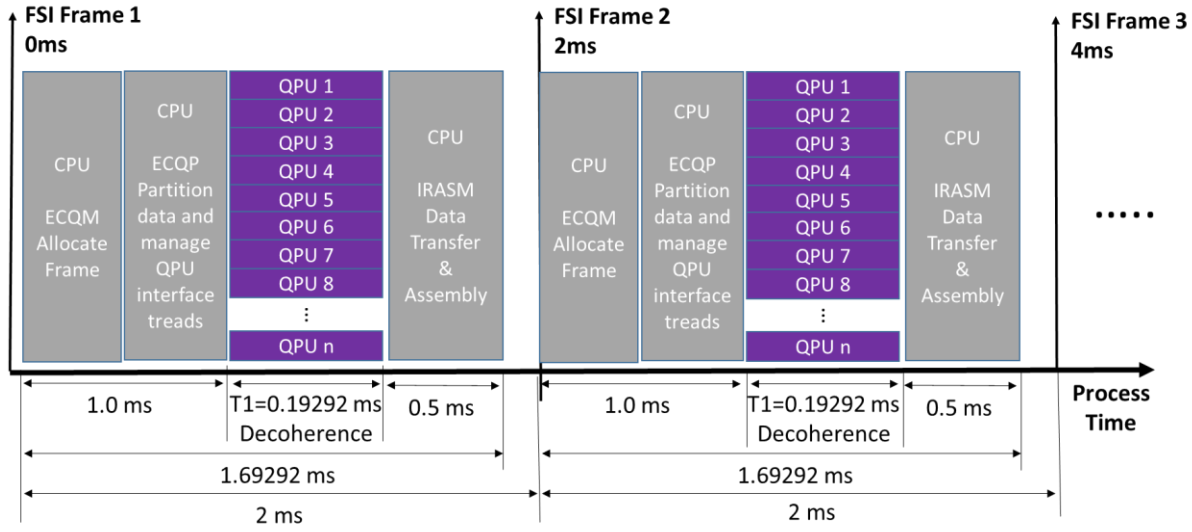


Note. Example process distribution methodology for larger images or data for distributed parallel processing, and assembly of results. The QPUs can be simulated or use actual QPU hardware.

Process synchronization is handled by two hierarchical executive schedulers. Appendix E describes the interfaces structure used for message passing and synchronization between executives. The quantum executive ECQP distributes the data to all QPUs and waits for completion of each QPU by using event semaphores. The top executive ECQM manages overall frame processing and synchronization based on messages sent and received from ECQP quantum scheduler. The ECQP quantum executive only completes its processing when all QPUs have finished, which at this point a message is sent to the ECQM executive for frame completion, and to start preparation for processing the next frame. An example of a process sequence for a notional 500Hz clock interrupt is shown next on Figure 17. The sequence includes reception of

Frame Start Interrupt (FSI) by ECQM, quantum task scheduling by ECQP, and distributed QC parallel processing by several QPUs. The processing is limited by a decoherence budget of 0.19292ms before wave function collapse is reached. Finally, the last processing stage includes IRASM for all QCs, and frame complete message sent from ECQP to ECQM.

Figure 17. *QPF Process Synchronization for a 500Hz Interrupt Clock Cycle and 16 QPUs.*



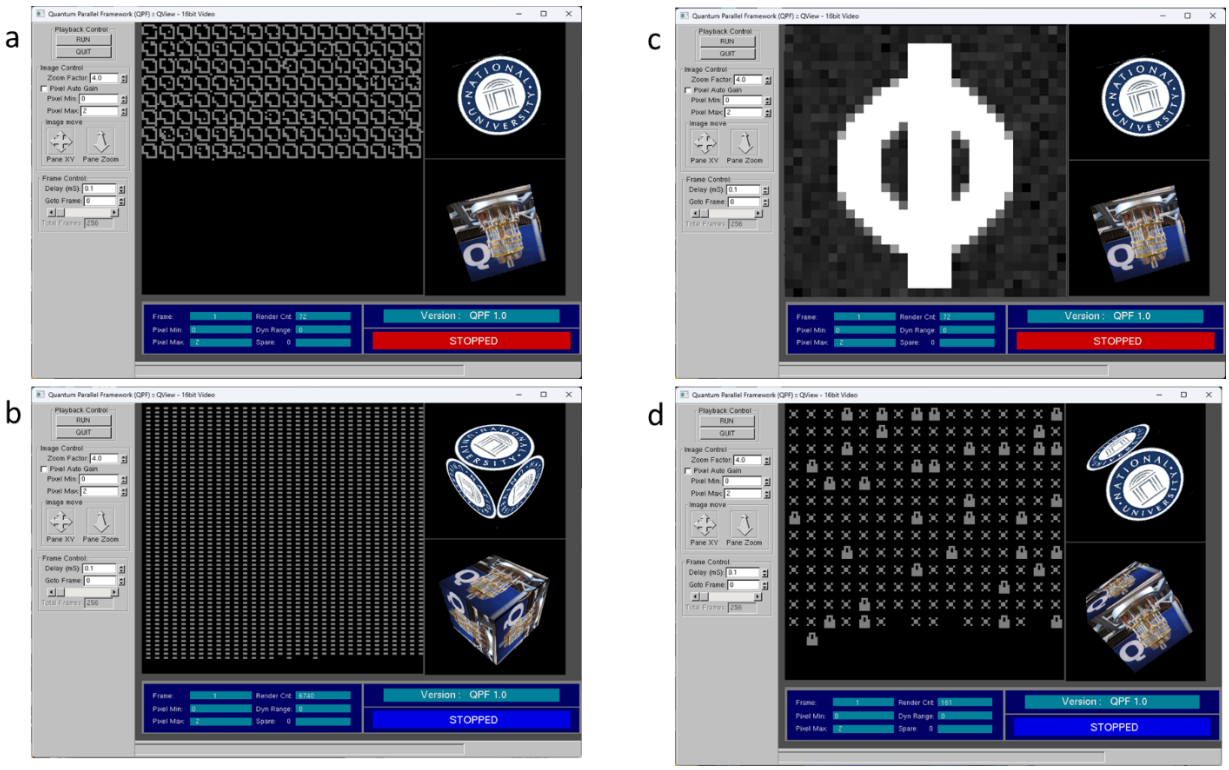
Note 1. Notional 500Hz FSI outlines the start of the processing cycle. Assuming direct Ethernet connection between CPU and QPU (non-cloud-based connection). T1 shows the maximum time a QC can execute before reaching wave function decoherence.

Note 2. The example shows a self-imposed timing budget of 1.0ms for ECQM and ECQP, and of 0.5ms for IRASM. Max QPUs are n=16. The T1 maximum decoherence time is from IBM Marrakech QPU. From IBM Quantum Platform. (Accessed November 14, 2025). https://quantum.cloud.ibm.com/?computer=ibm_marrakesh

The QPF is instrumented with an OpenGL display for generating imagery to provide the user with a runtime state of the algorithm. The following section briefly describes the displays shown for each of the example four test case displays for QHED, QCNN, QCCD, and QCKD algorithms for a 128x128 input data set. The QHED, QCNN, and QCCD algorithms are image-based processing and generate an image as shown on Figure 18a through c. The QCKD

algorithm takes as input a rotational string matrix and produces cryptographic keys for each of the matrix entries. Figure 18a shows the results of the 8x8 pixel kernel edge detected input shape computed by the QHED. Figure 18b shows the progress of the QCNN as the distributed parallel process works through the 2x8 kernel input test image. Figure 18c shows a no-noise output image, where each pixel has been converted from voltage to a digital signal by the QCCD ADC algorithm. Finally, Figure 18d shows the progress and status of the QCKD algorithm with a lock icon symbolizing no tamper detected and an X symbol for tamper detection. The QILQ test case does not have a display capability as the test purpose is to compare raw performance between the Interlin-q and QPF frameworks.

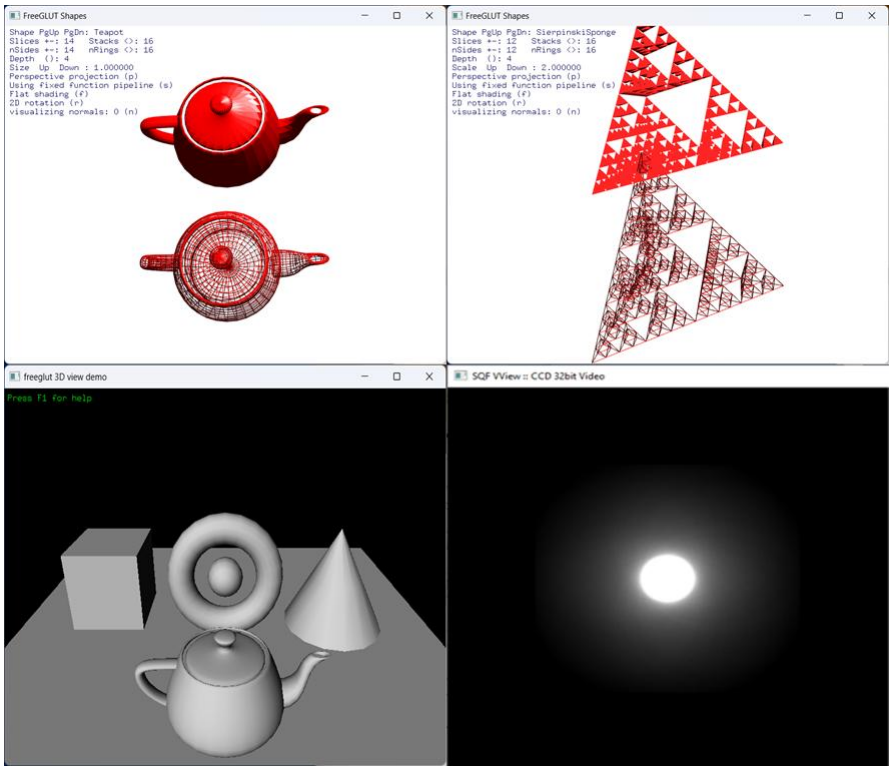
Figure 18. *Displays Showing Results for QHED, QCNN, QCCD, and QCKD Test Cases.*



Note. Figure 18a shows results for QHED, Figure 18b shows results for QCCD, Figure 18c shows results for QCCD, and Figure 18d shows results for QCKD.

QPF is instrumented with the OpenGL 3D graphics rendering engine, a well-known standard, used in many video games and scientific applications. OpenGL can render geometric and pixel imagery to visualize input and results of the quantum algorithms as well as act as a dynamic scene generator as shown in Figure 19.

Figure 19. QPF *OpenGL Capabilities are Used to Generate Input or Output Imagery.*



Note. OpenGL scene generation rendering capabilities include 3D geometry in upper top and lower left windows and pixel drawings as shown in the lower left window for a point source modeled after a 25x25 Point Spread Function (PSF) applied by the reference CCD.

Operational Definitions of Variables

Dependent Variables

The constructive research study experimentally tested the feasibility and performance of distributed processing on QC algorithms across several QPUs. There are five dependent

variables whose effects were analyzed including distributed QC performance time in [seconds], applicable to all QC algorithms (Bai et al. 2025; Han & Wang, 2024). Euclidian distance in [pixel intensity counts], which measures image divergence for algorithms QHED and QCCD (Divya, 2018; Qiskit Ecosystem, 2025; Konnik & Welsh, 2014). Classifier image prediction accuracy in [percentage], applicable to QCNN (Qiskit Ecosystem, 2025). Probability of tamper detection in [percentage] and Key Generation Rate (KGR) in [keys/second], applicable to QCKD (Qiskit Ecosystem, 2025). Upon dataset analysis, the hypotheses tested for QC distribution feasibility, QC distribution performance response time, and quantum noise effects on QC results. Table 2 shows a summary of QC algorithm test cases, independent variables and the effect observed during the experiment.

Table 2. *Dependent Variables Effects Observed per QC Algorithm Test Case.*

ALGORITHM TEST CASE	DEPENDENT VARIABLES	EFFECTS OBSERVED
QHED	• Pixel Euclidian Distance • Distributed performance	Measures the effects of quantum noise on edge detection and image distortion including pixel flipping due to quantum readout noise
QCNN	• Image Classification • Distributed performance	Measures test image neural network classification accuracy in presence of quantum readout noise, which adds pixel flipping distortions to image
QCCD	• Pixel Euclidian Distance • Distributed performance	Measures the effects of quantum noise on coupled charge collected image distortion including pixel flipping due to quantum readout noise
QCKD	• Tamper Detection • Key Generation Rate • Distributed performance	Measures the effects of quantum noise on cryptographic key distribution rotational strings and tamper detection under the effects of quantum readout noise. KGR measures the impact of rotational string length.
QILQ	• Framework Performance • Distributed performance	Comparison of QPF to Interlin-q framework performance. Distributed performance over 1 to 16 QPU configurations.

Note. All five QC test cases share a common independent variable for distributed performance for 1, 4, 8, and 16 QPU configurations.

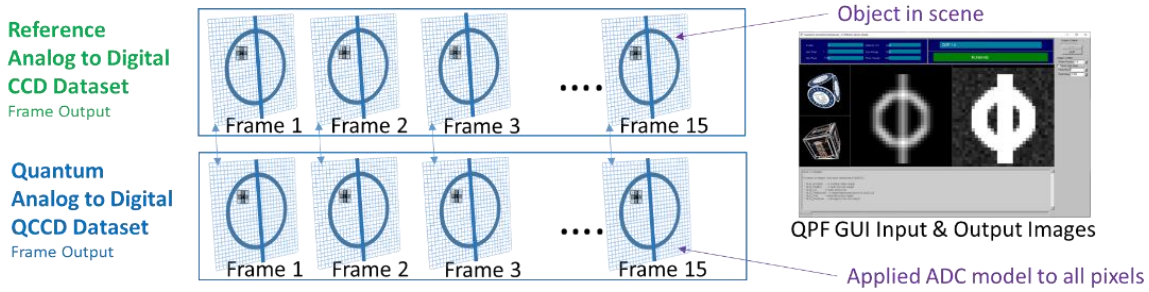
QPU Performance Time for All Quantum Algorithms

The processing time is fundamental to this research for evaluating the effectiveness of distribution of a QC algorithm over several QPUs and the effects on processing response time (Bai et al. 2025; Han & Wang, 2024). QPF is instrumented to collect per frame timing at the algorithm level, thread finite state machine level, and executive level to obtain an aggregate of the time to process a frame cycle.

QHED and QCCD Image Euclidian Distance (Noisy vs Non-Noisy Image)

A pixel's count values are used by Euclidian distance metrics to compare the quantum image results against reference models for QHED and QCCD. The Euclidian distance measures the effects of quantum noise on divergence for a quantum generated image when compared to a classical generated reference image (Divya, 2018). An average Euclidian distance is computed and normalized for the entire image with a 0.0 result signifying perfect match between test image versus reference image, and 1.0 value denoting complete divergence between the two images. Figure 20 shows a depiction of the Euclidian distance image comparison approach.

Figure 20. *Euclidian Distance Metric Methodology for 32x32 QHED and QCCD.*



Pixel Euclidian distance used for Per pixel comparison metric, where p is reference pixel irradiance and q is the corresponding quantum pixel: $d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$.

QCNN Classifier Prediction Performance (Noisy vs Non-Noisy Image)

The metric used for the QCNN evaluates the accuracy of prediction of the QCNN classifier to determine if the test image has a horizontal or vertical line contained in the image. The accuracy of classification prediction ranges from 0 to 100% (Qiskit Ecosystem, 2025). The measurement of accuracy is done under the influence of quantum noise.

QCKD Key Tamper Detection Robustness (Key Length Analysis)

The premise of the QCKD algorithm test is to determine if a key has been compromised or interdicted by an eavesdropper. The logic of the test interdicts the QCKD algorithm 100% of the time to verify if the key recipient can detect if the key has been compromised (Qiskit Ecosystem, 2025). A successful test determines that the key has been compromised, a true positive. A failure or false positive (Type I Error) does not detect that the key has been compromised and deems the key incorrectly as secure. The measurement of accuracy is done under the influence of quantum noise.

Independent Variables

Independent variables were selected based on works that identify performance impact to software applications, which is at the heart of this research. Independent variables include data size with image resolution for image algorithms (Obaidat, 2020; Yan et al., 2023), NUMA memory strategy (Bai et al., 2025; Han & Wang, 2024), thread priority (Basha et al., 2023), OS load balancing (thread CPU core affinity), and distributed 1, 4, 8, and 16 QPU configurations (Bai et al., 2025; Kan et al., 2024; Lindsay et al., 2021).

Tables 3 through 7 show the independent variable matrices for each of the five QC test cases. Independent variables drive the configuration of the QC algorithms to evaluate the impact on performance and impact of noise on the dependent variables. All five QC test cases are driven

by a group of common independent variables for noise type, data size, RAM NUMA strategy, OS thread priority, OS load balance, and number of QPUs to distribute QC algorithms.

The common set of independent variables were designed to obtain performance and quality of QC results under various configurations for obtaining mathematical patterns from where conclusions can be made to support hypotheses and answer research questions. In addition, algorithms have specific independent variables as listed in Tables 3 through 7. The independent variables in the common configuration were optimized to reduce QC response time. The noise type configures the QPU to enable quantum noise modeling. The noise type can be fast quantum gate readout (type 16), or default noise model (type 20), faster quantum gate (type 22), and tailored to favor performance based on algorithm type. The data size denotes the image size in pixels, or the input data matrix size. RAM NUMA is enabled to force allocation to memory banks as close as possible to the QC CPU core. OS thread priority is set for Realtime (RT) processing to favor performance and pre-empt other processing of lesser priority. A load balance of 1 tells the OS to relocate threads to CPU cores, which have the probability of completing the work faster. A load balance of type 2 denotes user assigned load balancing, allocating one QC per CPU core using the modulus function. An explicit load balance of CORE x, where x is a core number, denotes user assigned CPU core affinity to execute QC without changing core during run.

Table 3 contains the independent variable data collection matrix to drive the experiment for the QHED test case. Noise type 16 enables Qubit gate and readout noise models, which manifest as pixel flip errors (Mastriani, 2020; Dolciami, 2022).

Table 3. *QHED Independent Variables Data Collection Matrix Template.*

QHED INDEPENDENT VARIABLES								
NOISE TYPE	*DATA SIZE (PIX)*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	QC PERFORMANCE TIME (SEC) *1 QPU BASELINE*	*4 QPU*	*8 QPU*	*16 QPU*
16	**32x32**	**Enable**	**RT**	**OS 1**				
16	**512x512**	**Enable**	**RT**	**OS 1**				
16	**1024x1024**	**Enable**	**RT**	**OS 1**				
16	**2048x2048**	**Enable**	**RT**	**OS 1**				

Note. Independent variables in *italics and purple text*.

Table 4 contains the independent variable data collection matrix to drive the experiment for the QCNN test case. Noise type 20 enables Qubit readout and default QPU noise modeling, which manifests in test image corruption due to Qubit pixel encoding errors (Mastriani, 2020; Dolciami, 2022).

Table 4. *QCNN Independent Variables Data Collection Matrix Template.*

QCNN INDEPENDENT VARIABLES								
NOISE TYPE	*DATA SIZE (PIX)*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	QC PERFORMANCE TIME (SEC) *1 QPU BASELINE*	*4 QPU*	*8 QPU*	*16 QPU*
20	**4x8**	**Enable**	**RT**	**OS 1**				
20	**64x128**	**Enable**	**RT**	**OS 1**				
20	**128x256**	**Enable**	**RT**	**OS 1**				
20	**256x512**	**Enable**	**RT**	**OS 1**				

Note. Independent variables shown in *italics and purple text*.

Table 5 contains the independent variable data collection matrix to drive the experiment for the QCCD test case. The QCCD experiment was designed to collect data for reduced image sizes from 32x32 up to 256x256 to expedite time taken to process larger images using the QPU

simulator. Long processing times for the QCCD will not be a problem when running on actual QPU hardware. Noise type 22 enables Qubit faster readout error and QPU noise modeling, which manifests as pixel flips due to Qubit pixel encoding (Mastriani, 2020; Dolciami, 2022).

Table 5. *QCCD Independent Variables Data Collection Matrix Template.*

QCCD INDEPENDENT VARIABLES								
NOISE TYPE	*DATA SIZE (PIX)*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	QC PERFORMANCE TIME (SEC) *1 QPU BASELINE*	*4 QPU*	*8 QPU*	*16 QPU*
22	**32x32**	**Enable**	**RT**	**OS 1**				
22	**64x64**	**Enable**	**RT**	**OS 1**				
22	**128x128**	**Enable**	**RT**	**OS 1**				
22	**256x256**	**Enable**	**RT**	**OS 1**				

Note. Independent variables shown in *italics and purple text*.

Table 6 contains the independent variable data collection matrix to drive the experiment for the QCKD test case. In addition, a QCKD has the independent variable for rotation string lengths of 10, 20, 33, and 64 characters to be exercised for each of the four input matrices 4x4, 64x64, 128x128, and 256x256. Noise 20 enables Qubit readout and default QPU noise modeling, which manifests as random draws during the rotational string generation.

Table 6. *QCKD Independent Variables Data Collection Matrix Template.*

QCKD INDEPENDENT VARIABLES									
NOISE TYPE	*DATA SIZE (STR)*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	*ROTATE STRING*	QC PERFORMANCE TIME (SEC) *1 QPU BASELINE*	*4 QPU*	*8 QPU*	*16 QPU*
20	**4x4**	**Enable**	**RT**	**OS 1**	**10,20, 33,64**				
20	**64x64**	**Enable**	**RT**	**OS 1**	**“**				
20	**128x128**	**Enable**	**RT**	**OS 1**	**“**				
20	**256x256**	**Enable**	**RT**	**OS 1**	**“**				

Note. Independent variables shown in *italics and purple text*.

Table 7 contains the independent variable data collection matrix to drive the experiment for the QILQ test case. Noise type 20 enables Qubit readout and default QPU noise modeling; however, the QILQ algorithm is only evaluated for performance by this study and does not consider noise effects on quality of the CNOT gate output.

Table 7. *QILQ Independent Variables Data Collection Matrix Template.*

QILQ INDEPENDENT VARIABLES								
NOISE TYPE	*DATA SIZE (GATE)*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	QC PERFORMANCE TIME (SEC) *2 QPU BASELINE*	*4 QPU*	*8 QPU*	*16 QPU*
20	**2**	**Enable**	**RT 2**	**OS 1**				
20	**2**	**Enable**	**RT 2**	**OS 2**				
20	**2**	**Enable**	**RT 2**	**CORE 2**				

Note. Independent variables shown in *italics and purple text*.

Table 8 contains the dependent variable data collection matrix to drive the experiment for the QHED test case. The dependent variables for the QHED are frame average Euclidian distance and performance percentage improvement from baseline for 4, 8, and 16 QPU configurations. Noise type 16 enables Qubit gate and readout noise models.

Table 8. *QHED Dependent Variables Data Collection Matrix Template.*

QHED DEPENDENT VARIABLES											
NT	**DATA SIZE (PIX)**	**RAM NUMA**	**TR PRI**	**LOAD BALANCE**	*AVERAGE FRAME EUCLIDIAN DISTANCE (PIX)* **1 QPU BASELINE**	**4 QPU**	**8 QPU**	**16 QPU**	*PERFORMNCE IMPROVE % FROM BASELINE* **4 QPU**	**8 QPU**	**16QPU**
16	**32x32**	**Enable**	**RT**	**OS 1**							
16	**512x512**	**Enable**	**RT**	**OS 1**							
16	**1024x1024**	**Enable**	**RT**	**OS 1**							
16	**2048x2048**	**Enable**	**RT**	**OS 1**							

Note. NT Noise Type. TR = thread. PRI=priority. Dependent variables shown in *italics and blue text*.

Table 9 contains the dependent variable data collection matrix to drive the experiment for the QCNN test case. The dependent variables for the QCNN are frame average Euclidian distance and performance percentage improvement from baseline for 4, 8, and 16 QPU configurations. Noise type 20 enables Qubit readout and default QPU noise modeling.

Table 9. *QCNN Dependent Variables Data Collection Matrix Template.*

QCNN DEPENDENT VARIABLES											
NT	DATA	RAM	TR	LOAD	*AVERAGE FRAME EUCLIDIAN DISTANCE (PIX)*				*PERFORMANCE IMPROVE % FROM BASELINE*		
	SIZE (PIX)	NUMA	PRI	BALANCE	1 QPU BASELINE	4 QPU	8 QPU	16 QPU	4 QPU	8 QPU	16QPU
20	4x8	Enable	RT	OS 1							
20	64x128	Enable	RT	OS 1							
20	128x256	Enable	RT	OS 1							
20	256x512	Enable	RT	OS 1							

Note. NT Noise Type. TR = thread. PRI=priority. Dependent variables shown in *italics and blue text*.

Table 10 contains the image classification accuracy for the QCNN algorithm's ability to correctly identify horizontal or vertical line patterns in a test image. The table also includes performance improvement variable from baseline for 4, 8, and 16 QPU configurations. Noise type 20 enables Qubit readout and default QPU noise modeling.

Table 10. *QCNN Dependent Variables Data Collection Matrix Template.*

QCNN DEPENDENT VARIABLES											
DATA	RAM	TR	LOAD	*FRAME AVE CLASSIFICATION ACCURACY %*				*PERFORMANCE IMPROVE % FROM BASELINE*			
SIZE (PIX)	NUMA	PRI	BALANCE	1 QPU BASELINE	4 QPU	8 QPU	16 QPU	4QPU	8QPU	16QPU	
4x8	Enable	RT	OS 1								
64x128	Enable	RT	OS 1								
128x256	Enable	RT	OS 1								
256x512	Enable	RT	OS 1								

Note. NT Noise Type. TR = thread. PRI=priority. Dependent variables shown in *italics and blue text*.

Table 11 contains the dependent variable data collection matrix to drive the experiment for the QCCD test case. The dependent variables for the QCCD are frame average Euclidian distance, and performance percentage improvement from baseline for 4, 8, and 16 QPU configurations. Noise type 22 enables Qubit faster readout error QPU noise modeling.

Table 11. *QCCD Dependent Variables Data Collection Matrix Template.*

QCCD DEPENDENT VARIABLES											
NT	**DATA**	**RAM**	**TR**	**LOAD**	***AVERAGE FRAME EUCLIDIAN DISTANCE (PIX)***				***PERFORMANCE IMPROVE % FROM BASELINE***		
	SIZE (PIX)	**NUMA**	**PRI**	**BALANCE**	**1 QPU BASELINE**	**4 QPU**	**8 QPU**	**16 QPU**	**4QPU**	**8QPU**	**16QPU**
22	**32x32**	**Enable**	**RT**	**OS 1**							
22	**64x64**	**Enable**	**RT**	**OS 1**							
22	**128x128**	**Enable**	**RT**	**OS 1**							
22	**256x256**	**Enable**	**RT**	**OS 1**							

Note. NT Noise Type. TR = thread. PRI=priority. Dependent variables shown in *italics and blue text*.

Table 12 contains the dependent variable data collection matrix to drive the experiment for the QCKD test case. The dependent variables for the QCKD are frame performance, tamper detection probability, KGR, and performance percentage improvement from baseline for 4, 8, and 16 QPU configurations. Noise type 20 enables Qubit readout and default QPU noise modeling. Each of the four QC test cases will be run for rotation string lengths of 10, 20, 33 and 64 characters.

Table 12. *QCKD Dependent Variables Data Collection Matrix Template.*

QCKD DEPENDENT VARIABLES										
DATA	**RAM**	**TR**	**LOAD**	**ROT**	***TAMPER***			***PERFORMANCE IMPROVE % FROM BASELINE***		
SIZE (STR)	**NUMA**	**PRI**	**BALANCE**	**STR**	***DETECT %***	**1 QPU BASELINE *KGR (KEY/SEC)***	**4 QPU *KGR (KEY/SEC)***	**4QPU**	**8QPU**	**16QPU**
4x4	**Enable**	**RT**	**OS 1**							
64x64	**Enable**	**RT**	**OS 1**							
128x128	**Enable**	**RT**	**OS 1**							
256x256	**Enable**	**RT**	**OS 1**							

Note. Noise Type. Noise type 20. TR = thread. PRI=priority. STR=string. ROT=rotation. STR=string. Dependent variables shown in *italics and blue text and blue text*.

Table 13 contains the dependent variable data collection matrix to drive the experiment for the QILQ test case. The dependent variable for the QILQ is performance percentage improvement from baseline for 1, 4, 8, and 16 QPU configurations. The scope of the QILQ test case is to compare QPF to the Interlin-q distributed framework to evaluate performance. Noise type 20 enables Qubit readout and default QPU noise modeling; however, noise is not considered in the analysis of this test case and only included to ensure the noise model is operational.

Table 13. *QILQ Dependent Variables Data Collection Matrix Template.*

QILQ DEPENDENT VARIABLES											
CNOT	**RAM**	**TR**	**LOAD**	***QC PERFORMANCE TIME (SEC)***				***PERFORMANCE IMPROVE % FROM BASELINE***			
GATE	**NUMA**	**PRI**	**BALANCE**	**1 QPU BASELINE**	**4 QPU**	**8 QPU**	**16 QPU**	**1QPU**	**4QPU**	**8QPU**	**16QPU**
2	**Enable**	**RT 2**	**OS 1**								
2	**Enable**	**RT 2**	**OS 2**								
2	**Enable**	**RT 2**	**CORE 2**								

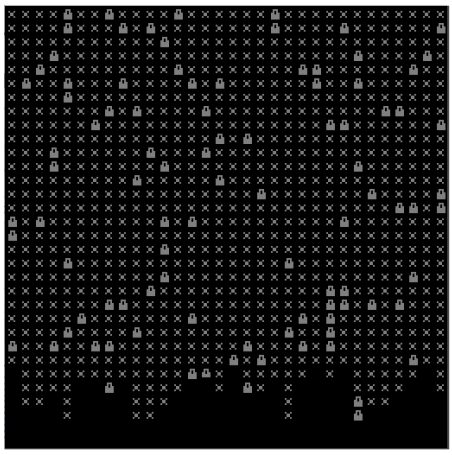
Note. NT Noise Type 20. TR = thread. PRI=priority. STR=string. Dependent variables shown in *italics and blue text*.

Data Size

The QHED model relies on images of various resolutions to help evaluate the robustness of the algorithms including 32x32, 512x512, 1024x1024, and 2048x2048 pixels. The selection of an increasing scalable size allows the assessment of the algorithm performance from small to larger CCD sensors available in the market (Potma et al. 2021). The QCCD model uses reduced image sizes of 32x32, 64x64, 128x128, and 256x256 pixels to mitigate long times incurred when processing larger images using QPU simulators. According to Gultom & Amirullah, (2022), an 8x8 image is represented with a 6-Qubit QC, which yields the best performance. All images are broken down into smaller 8x8 pixel image tiles to process the image similar to how a multi-GPU processes a large image (Eilemann et al., 2009).

The QCNN inputs are comprised of 2x8 pixel test images, where they are all stitched in a larger image for each of the executing QPUs to process as a tile in a parallel fashion. The selection size of the 2x8 is based on the actual design of the QC algorithm. The intent is to evaluate the design and performance of the algorithm pristine without needing to modify the size of the images (Qiskit Ecosystem, 2025). Using input matrices for rotation strings, the QCKD model generates key matrixes comprised of 4x4, 32x32, 64x64, 256x256, and 512x512 keys. In this case, the GUI is used to provide status to users and does not render an image. The data sizes are designed to evaluate the performance and robustness of the QCKD algorithm across various processing loads. Each QCKD algorithm will be distributed in parallel with its own QPU (Gultom & Amirullah, 2022). Figure 21 shows a 32x32 QCKD generated array of test results represented by a GUI. The lock icon represents unsuccessful detection of tampered state (Type I errors), and X represents successful key tamper detection.

Figure 21. 32x32 *Crypto Key Matrix with X for Tamper Detection.*



Note. The QCKD algorithm is tested by interdicting 100% of the time and testing for detection.

NUMA RAM Allocation Strategy

NUMA memory allocation strategy, when enabled on a workstation Basic Input Output System (BIOS), relies on memory allocation close to the CPU core requesting memory. Data locality plays a significant role in multi-processor multi-core platforms as memory can be allocated across memory banks that are not in proximity with the thread and core owning the data (Bai et al. 2025; Han & Wang, 2024). When NUMA is disabled, the OS interleaves allocation of memory based on the memory bank scheduled for allocation regardless of thread executing CPU core, which may increase memory access latency. For example, if the memory is allocated on a memory bank owned by CPU 1, but the allocating thread has affinity with a core on CPU 0, then there will be increased memory access latency (Bai et al., 2025). The increased latency is incurred by moving data across the Quick Path Interconnect (QPI) Bridge between CPU 0 and 1 as shown in Figure 22.

Thread Priority

The priority of threads can be changed to a higher or lower value to favor processing during OS thread scheduling. Those threads that have higher priority will obtain longer processor quantum time slices (Basha et al., 2023). In this case quantum refers to the OS time allocated to process a thread before the next context switch. A note is that when running with actual QPU hardware, the processing takes place at the remote QPU processor. However, all interfacing and data management to extract data from the QPU takes place at a local CPU core. Therefore, a higher priority improves the performance of the QPU interface thread to expedite data management and storage locally. When a QPU is simulated on a CPU core, priority will improve performance of the QPU simulation over other competing threads (Basha et al., 2023).

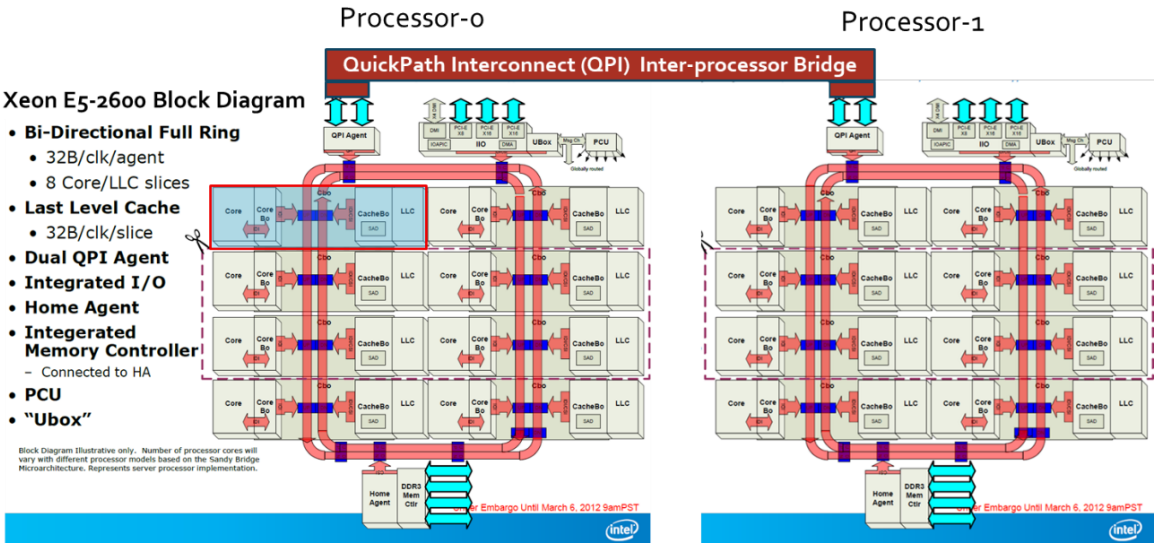
Thread Load Balancing (CPU core affinity)

The load balancing is related to the CPU core affinity a thread is associated with. If the OS provides the load balance, the OS assigns processing on CPU cores that can complete processing quicker (Bai et al. 2025; Han & Wang, 2024). This causes threads to hop from core to core, which may be advantageous for some algorithms, but detrimental to others. For example, for threads where memory access is important, it may be beneficial to assign a thread to cores that are closer to the CPU memory bank interface (Bai et al. 2025). In addition, assigning a thread to be fixed to a particular core also allows some algorithms to benefit from Level L1, L2, and L3 caching, which will expedite memory access to local cache instead of reaching out to main memory banks. An example of core level access is shown on Figure 22 by the blue box on processor 0 on core 0 (Anand Tech, 2024).

Number of QPUs

The number of QPUs is the variable that determines how many QPU's to use to distribute processing for the QC algorithm. The current configurations include 1, 4, 8, and 16 QPUs, which will distribute uncoupled data (with no dependencies between data) (Bai et al., 2025; Kan et al., 2024; Lindsay et al., 2021). The QPF framework will distribute the QC processing equally among the QPU's. The selection of up to 16 maximum was based on the current number of CPU cores available on the Z840 HP with two Xeon ES-2570 Haswell processors per Figure 22.

Figure 22. *Xeon ES-2570 Haswell 16 Core 2-Processor QPI Architecture.*



Note. Two processors representative of the HP Z840 platform were used to collect sample data for this research. From Anand Tech. (2024 15 April). Dual sandy Bridge for Servers. https://www.anandtech.com/Show/Index/5553?cPage=5&all=False&sort=0&page=1&slug=the-xeon-e52600-dual-sandybridge-for-servers.

Study Procedures

One of the early tasks of this constructive research was to design and develop the QPF framework instrument per GST theory and integrate the five QC test case algorithms (Johnson, 2019). Upon system integration of all QPF QC algorithms, the software was ready to generate

experimental data. Figure 15 shows how the data collection is generated by the execution of a ten-stage procedure to process each data frame, execute the QC algorithm models hosted by the QPF, collect results, calculate per frame metrics, and display the results. The QPF instrumentation codes are stored on a Git repository, as well as the test artifacts analysis generated per Appendix C.

Dataset Collection Design

Datasets were collected for each of the five test case QC algorithms for combinations of 1, 4, 8 and 16 QPUs. The 1 QPU configuration represents the baseline non-optimized reference dataset. For the Interlin-q QC, only 2 QPUs were used as it is the maximum configuration it supports. However, on the QPF side, the QILQ algorithm executed for 1, 4, 8 and 16 QPU configurations to compare the scalability and performance against the 2 QPU Interlin-q QC baseline. The independent variables to trade included QPUs, OS load balancing, thread CPU affinity & priority, and NUMA RAM allocation strategy. For QHED, QCCD, and QCNN, the data size input represents image pixels. For QCKD, the data size input represents rotational strings. For QILQ, the data size input represents the number of samples collected for performance comparison to the QPF instrument.

Data Collection & Acquisition

Stage 1: Data Injection Processing

The methodology of this experiment can be described in a ten-step process starting with the input data, image, or cryptography, provided by the ECQM and ECQP executives, which schedules the selected QC algorithm for execution. Any combination of the five QC algorithms can be set for execution based on system configuration. However, this research study only schedules one QC algorithm at a time to generate the required experiment data. In addition, a

pipeline for image processing can be established by sequential execution of QCCD, QHED, and QCNN algorithms. This would model image processing from photon reception at the QCCD model to QHED image processing, and QCNN object detection and classification.

Stage 2: Task Scheduling of Frame Finite State Machine

The ECQM executive also coordinates execution of the frame state machine process, and coordinates simulation execution for the run. The QPU is managed by a cloud connection, which serves as the input and output mechanism for coordinating execution of the QC model. If running QPU simulators locally, then the interface used is shared memory.

Stage 3: Task Scheduling of Quantum Circuit Model

This step includes execution of the QC on QPU hardware or simulator and execution blocks until results are returned. All results are stored on file by the thread managing the execution of a QC. The ECQP executive receives input data prior to application of the QC algorithm for image processing, or key generation if the cryptographic QCKD model is executed. If image processing is required, then the input image is sent to the QHED, QCNN, or QCCD algorithms for processing. If cryptographic data is required, then the data is sent to the QCKD algorithm for processing. The QILQ QC algorithm is processed in the same fashion as the other algorithms. The QPU executes the QC algorithm and returns results to the CPU.

Stage 4: Generate Quantum Dataset

During the execution of the QC algorithm, the QPF monitors the execution on the QPU and coordinates the management of shots (number of times a QC algorithm is executed), and stores shot results for each QPU.

Stage 5: Return Quantum Data to CPU

Upon generation of results by the QPU, the QPF framework collects run data, measures runtime process timing, calculates metrics, and stores results to disk. Processing the QPF framework blocks until all QC algorithms complete execution for that data frame. All data generated is saved to files on a frame-by-frame basis. Upon processing completion by the QC algorithms, the radiometric content of generated images is compared to pixel by pixel and frame by frame using Euclidian distance for QHED and QCCD. For QCNN, the accuracy of horizontal and vertical line prediction classifiers is computed. When executing the QCKD algorithm, the tamper detection probability, KGR, and algorithm timing are generated. The data artifacts generated at this stage include images or matrix of cryptographic keys for each processing cycle, including raw inputs, generated results, and metrics.

Stage 6: Wait to Process Next Command from CPU Executive

Process synchronization is managed by the ECQM, and ECQP executives to ensure a frame processing is deemed complete when all QC interface threads have finished their processing.

Stage 7: Storage of Frame Generated Data

All data generated for each QPU frame is stored in text files for frame runtime metrics and post-run performance analysis.

Stage 8: Frame Complete

A complete frame message is sent from the ECQP to the ECQM executive when all QC interface threads have finished processing, including frame data storage.

Stage 9: Generate Frame Metrics & Generate Dynamic Next Frame Data

The results of the completed frame are evaluated, and based on the outcome, the attributes of the next frame to be generated are created. For example, if an image processing object tracker is closing the loop around the image, the tracker would calculate the camera angle to be used for the next frame and any dynamic model applied to the object being tracked. A new image scene would be generated based on the camera position and attitude using the OpenGL graphics subsystem. If the QC algorithm executing in closed loop is the QCKD algorithm, the results of the prior frame key would be evaluated. Attributes for the next frame would be generated and applied to produce the next key. For example, if an image is being encrypted, each frame could be encrypted with a different key to improve the robustness of the encryption and the overall security hardened stance. An increase in the key length could be changed to produce a hardened key that would be more difficult to interdict.

Stage 10: Perform Post-Run Analysis

Post run data analysis includes plotting of Key Performance Parameters (KPP) to evaluate the overall performance of the simulation including metrics for frame processing times, Euclidian distance, accuracy of image classification, KGR, and probability of key tampering for encryption.

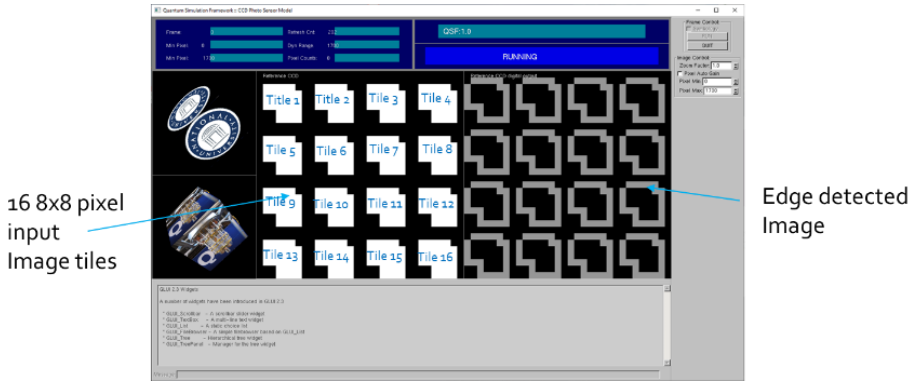
Data Preprocessing

Both executive controllers, ECQM and ECQP, generate per frame timing, image data, or cryptographic key data, which requires preprocessing data for each algorithm prior to a closed loop run. The following sections outline the specific preprocessing required for each of the QC algorithm models.

QHED Data Preparation

The data preparation includes generation of reference edge detected imagery for the resolution under study including 32x32, 512x512, 1024x1024, and 2048x2048 pixels. The instrument used to generate the reference image is the same QPF with quantum noise disabled so that perfect imagery can be generated. Figure 23 shows how a 32x32 image is broken into 8x8 pixel tiles to be processed independently by each QPU. QHED processing allocates one QHED quantum algorithm per tile for each of the 16 QPUs. Therefore, for a 32x32 pixel image, the entire image is processed in parallel. Although the image shown is relevant to the QHED, the same tile processing methodology is used for multiple data sizes for QCNN, QCCD, QILQ, and QCKD input rotation strings.

Figure 23. *QHED Input Image on Left and Output Edge Detected Image on Right.*



Note. Image represents a 32x32 pixel image comprised of 16 solid tiles as shown on the left.

QCNN Data Preparation

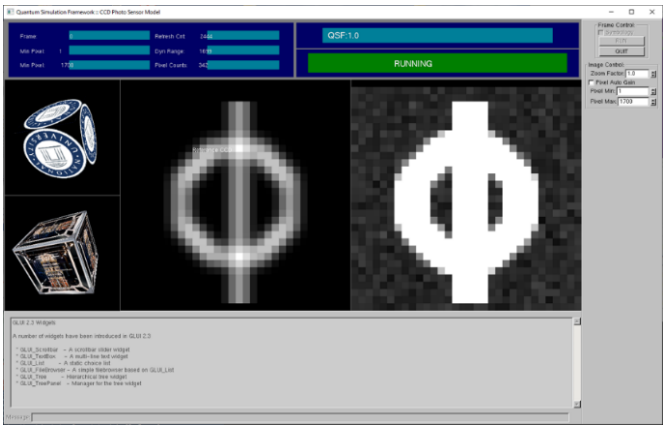
The quantum neural network requires training data to identify horizontal and vertical patterns in the test image. The QPF instrument is used to train the QCNN as part of the initialization by using a configuration variable to enable training upon QPF startup. Upon each

QPU's QCNN training complete, the train data features are saved in binary files so that future runs can load the trained QCNN feature map without the need to incur the cost of training every time the QPF is started. The objective function served as an initial evaluation of the trained data's accuracy. Each QPU generated its own objective function for its training dataset to evaluate the accuracy of the prediction using 200 epoch training cycles, which can be changed if needed. The selection of 200 epochs cycles was based on the original configuration for the QCNN algorithm. The intent of this research was to integrate QC algorithms from pristine models and minimize changes to the algorithm to facilitate reference comparisons. An observation is that Linux and MS Windows save binary data in different formats and are not interchangeable; therefore, each platform will have to be trained individually.

QCCD Data Preparation

The QCCD data preparation includes generation of reference CCD model imagery for resolutions including 32x32, 64x64, 128x128, and 256x256 pixels. The QPF generates the reference CCD image without quantum noise present. The reference image was generated in the CPU as shown on the right image in Figure 24. The original CCD model is written in MATLAB and ported to C++ code for generation of CCD reference imagery, and to provide a more seamless integration with closed loop simulations (Konnik & Welsh, 2014).

Figure 24. *CCD Reference Voltage Pixels on Left with Reference ADC Conversion on Right.*



Note. Reference 32x32 pixel CCD image on right has no quantum noise included.

QCKD Data Preparation

The QCKD does not require the pre-generation of data to be used during the run. The only assumption the QCKD test software makes is that the QCKD is interdicted 100% of the time, and it is left to the QCKD quantum generated key to be checked to determine effective interdiction detection (QiskitEcosystem, 2025).

Model Fit Evaluation & Diagnostics - Reference Models

Quantum Edge Detection Model

Edge detection relies on transitions in the image from one to zero and from zero to one to identify the perimeter boundary of an image. The Hadamard edge detector model performance improvement surpasses the QSobel edge detection algorithm with polynomial improvement and exponential improvements over classical algorithms (Yao et al., 2017).

The QHED algorithm with noise modeling included is evaluated (fit compared) against an ideal reference image without noise to provide the impact of quantum noise on the results.

The metric used to show radiometric traceability and performance of the QHED results to the reference model is Euclidian distance, which produces a numerical divergence value from the reference image (Divya, 2018). The Euclidian distance metrics was used to validate the radiometric image results to the reference model (Peng, 2014). The formula for Euclidian distance is shown in equation 3-1.

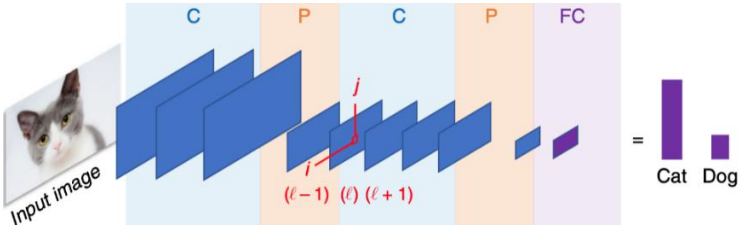
$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \quad (3\text{-}1)$$

The second metric used was the performance of the QC algorithm over 1, 4, 8, and 16 QPU configurations to evaluate algorithm response time improvements.

Quantum Convolutional Neural Network

Quantum convolutional neural networks, like their classical counterparts, rely on pixel gradients (low to high and high to low) transitions to detect features in the image. The pixelated input image is stored in Qiskit's ZFeatureMap from where it is processed by the convolutional and pooling layers. For example, for a trained QCNN to detect cats and dogs, the input image passes through a series of alternating convolutional (C), and Pooling (P) layers, where different layers detect different image features. As shown in Figure 25, the *l-th* layer recognizes features and patterns along the *i-j* plane. The pooling layer reduces the image size footprint, allowing for less computational cost and fewer learning parameters required. The Fully Connected (FC) layer provides predictions of object classification. For this example, the classification determination shows if the detected image contains a cat or dog (Qiskit Ecosystem, 2025).

Figure 25. *QCNN Cat-Dog Classifier Schematic.*

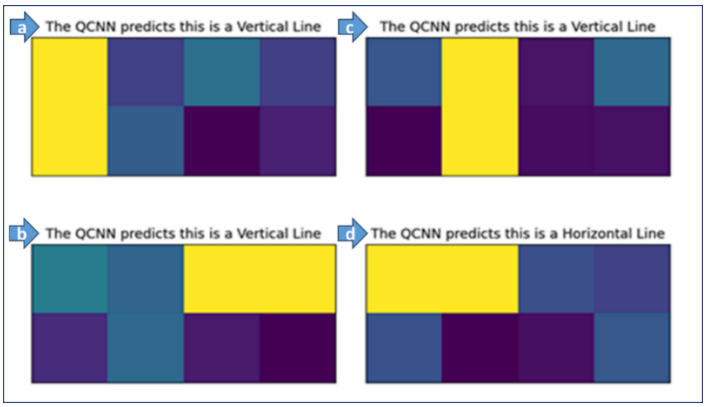


Note. QCNN cat-dog classifier schematic. From Qiskit Ecosystem. Qiskit Machine Learning 0.8.2. (2025). The Quantum Convolutional Neural Network. Accessed Jan 6, 2025. https://qiskit-community.github.io/qiskit-machine-learning/tutorials/11_quantum_convolutional_neural_networks.html.

The QCNN algorithm is evaluated (fit compared) with a probability representing the accuracy of pattern detection for classifier's prediction for horizontal and vertical lines in the test image. The test data is comprised of 2x8 pixel images containing vertical and horizontal yellow lines (QiskitEcosystem, 2025). The QCNN algorithm is instrumented to calculate the prediction automatically for every frame processed. Finally, the third metric used is the performance of the QC algorithm over 1, 4, 8, and 16 QPU configurations to determine the best response time improvements.

Figures 26a through 26d shows examples of four predictions described in the texts above each of the yellow line. Figure 26a prediction for the vertical line in the upper left quadrant is correctly classified as a vertical line. Figure 26b prediction is incorrect for a vertical line as truth shows a horizontal line. Figure 26c prediction for the vertical line is correctly classified as a vertical line. Finally, Figure 26d prediction for the horizontal line is correctly classified as truth shows a horizontal line.

Figure 26. *QCNN Classifier Prediction for Horizontal and Vertical Lines in Test Image.*



Note. QCNN classifier horizontal and vertical results follows: Figure 25a prediction is correct, Figure 25b prediction is incorrect, Figure 25c prediction is correct, and Figure25d prediction is correct. From Qiskit Ecosystem. Qiskit Machine Learning 0.8.2. (2025). The Quantum Convolutional Neural Network. Accessed Jan 6, 2025. https://qiskit-community.github.io/qiskit-machine-learning/tutorials/11_quantum_convolutional_neural_networks.html.

A key component of QCNN image processing is the Fast Fourier Transform (FFT), applied during the convolution (C) stage. The FFT is used to convert data from the time domain to the frequency domain to facilitate processing. A QFFT has an advantage as it operates at logarithmic complexity versus quadratic complexity for the classical FFT algorithm as shown in Figure 27 (Yan et al., 2023).

Figure 27. *Computation Complexity Between FFT and QFFT Algorithms.*

	Computational complexity
FFT	$O(NL^2 \log_2 L^2)$
QFFT+RAM	$O(NL^2) + O(L^2 \log_2 L^2)$
QFFT+qRAM	$O(L^2 \log_2 L^2)$

Note. Fast Fourier Transform (FFT) classical and quantum complexity comparison. From Asaka, R., Sakai, K., & Yahagi, R. (2019). Quantum circuit for the fast Fourier transform. https://doi.org/10.1007/s11128-020-02776-5

Quantum Coupled Collection Device Analog to Digital Quantization Models

The QCCD sensor algorithm executes a subset of the operations defined by the reference CCD sensor model. The purpose of execution of a simplified QCCD sensor algorithm is to facilitate initial integration and investigation of results robustness. The QCCD algorithm is evaluated (fit compared) against a reference CCD model image using Euclidian distance to determine the measure of divergence or likeness (Divya, 2018). The second metric used is the performance of the algorithm over 1, 4, 8, and 16 QPU configurations to determine the best response time improvements.

The following operations are executed by the reference CCD sensor model including photon to charge modeling, charge to voltage modeling, and voltage to digital conversion. The QCCD sensor model converts the analog voltage from the upstream models including addition of integral linear errors, and addition of quantization noise. The final stage of the ADC conversion to a digital signal is governed by the pixel depth of the CCD camera sensor (Konnik & Welsh, 2014). The total voltage calculated out of the voltage-to-voltage non-linearity models is calculated by the ADC and rounded to produce a digitized signal pixel value per equation 3-2.

$$I_{DN} = round(A_{ADC} - I_{total.V}) \qquad (3\text{-}2)$$

Where $I_{total.V} = (V_{ADC.V} - I_V)$ represents the total voltage accumulated during frame integration cycle.

Integral Linear Model. The ADC linear model can be calculated as follows from equation 3-3. For simplicity, the non-linear ADC model is not considered in this research as literature considers it an optional model (Konnik & Welsh, 2014).

$$A_{ADC} = \frac{1}{K_{ADC}} \qquad (3\text{-}3)$$

Quantization Noise Model. The ADC linear model can be calculated as follows from equation 3-4. The noise quantization (analog voltage to digital conversion) is an additive value.

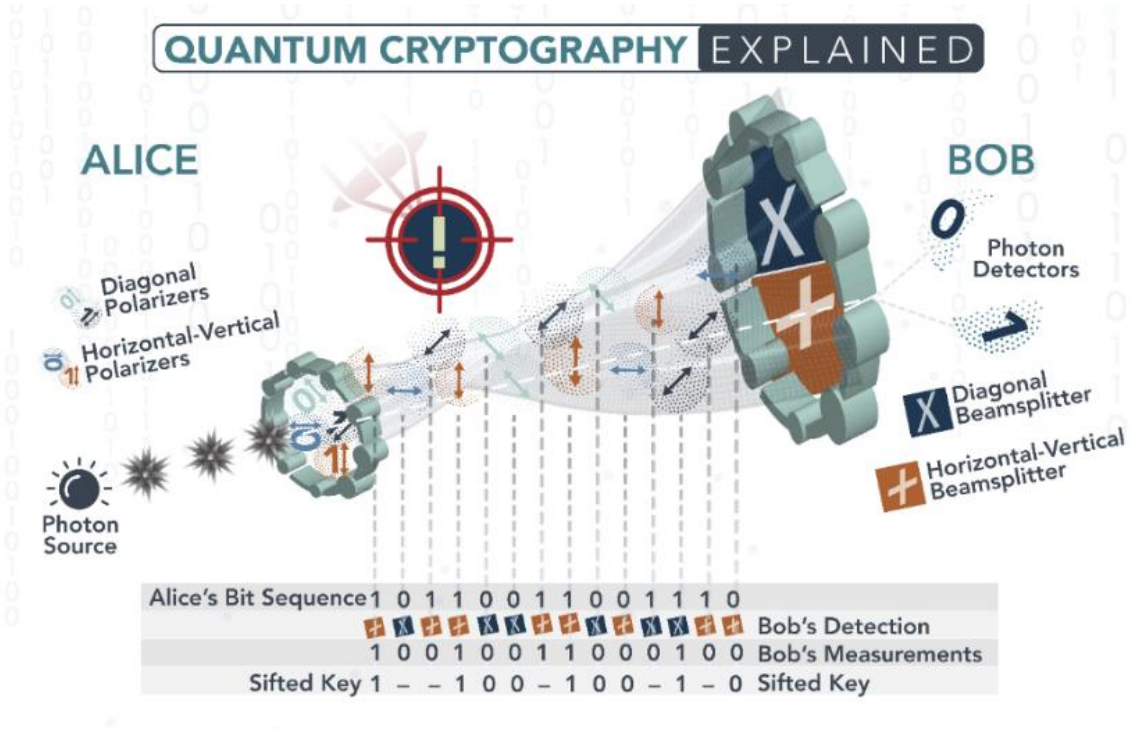
$$\sigma_{\mathrm{ADC}} = \sqrt{\frac{q_{ADC}^2}{12}} \qquad (3\text{-}4)$$

Quantized Pixel Depth Dynamic Range Considerations. The dynamic range of a photodetector is defined as the ratio of the maximum achievable signal (proportional to light) to the sensor noise. For example, the theoretical dynamic range pixel resolution for a 16-bit per pixel image is 2^{16} or 0 to 65,535 colors. However, given thermal losses and noise considerations, the effective dynamic range for a 16-bit photodetector may end up being 2^{14} or 16,384 (Gambheer & Bhat, 2023). Therefore, to facilitate initial integration of the QCCD ADC algorithm and reduce the number of Qubits required to encode the image, the pixel depth used in this research was 2^8 for 8-bit with 256 shade variations of wavelength for a single-color image.

Quantum Cryptography Key Distribution Model

The methodology of QCKD relies on generation of a rotating string using random draws and encoding the outcome by both the sender and receiver of the cryptographic key. The process requires the sender and receiver to exchange the random rotating strings generated and traverse the bit string comparing and noting the bits the match (QiskitEcosystem, 2025). The resulting matching shifted key bits become the cryptographic distributed key. The QCKD generation process is depicted in Figure 28.

Figure 28. *QCKD Polarization and Beam Splitters to Change and Read Photons Polarity.*



Note. Key generation depiction with polarizers and rotation string bit sequence. QuantumXChange. Quantum Cryptography, Explained. (Accessed May 6, 2025). https://quantumxc.com/blog/quantum-cryptography-explained/.

The strength of the QCKD methodology relies on the power of quantum mechanics, where a Qubit cannot be measured and then reproduced to have the same pristine state it had prior to measurement to hide interdiction (Rasmusson & Barney, 2025). The QCKD algorithm uses a rotator device as shown in Figure 28 to polarize the photons through it to generate a random state. The randomly rotated photon is model by a Qubit as the rotation string is generated. Diagonal and horizontal beam splitters are used at the receiving end to extract and randomly generate the receiver's rotation string. The power of the QCKD algorithm is that it can be integrated with current encryption technology as the key generation is one of the few components to be integrated including the photon encoder at the source as well as the beam splitters at the receiving end (QuantumXChange, 2025).

The QCKD algorithm was evaluated for its effectiveness to detect eavesdropping and interdiction detection. The algorithm developed to evaluate the QCKD was designed to always attempt to interdict or eavesdrop on the transmittal of the QCKD from Alice to Bob. This facilitates evaluating the algorithm as the truth is represented by transmittal of the QCKD Qubits from Alice to Bob under a 100% interdicted environment (Qiskit Ecosystem, 2025). The metric is the probability of key interdiction tamper detection. The robustness of the QCKD was evaluated with a rotating string configuration from weakest to strongest using a 10, 20, 33, and 64 string length. The 33-character string length is the baseline used in the original pristine QCKD configuration. The rotation string lengths of 10, 20, and 64 characters were chosen to investigate tamper detection from less hardened to most hardened key configurations. Finally, the performance of the algorithm was evaluated over 1, 4, 8, and 16 QPU configurations to determine the best response time improvements.

Statistical Analysis

The objective of the statistical analysis is to obtain numerical patterns from the observed data so that further insight can be gleaned. The scope of the statistical analysis was to determine the effects of quantum noise on four of the QC test cases including QHED, QCNN, QCCD, and QCKD. The QILQ algorithm was not included because this test case is used to measure performance comparisons to the QPF instrument, and the effects of quantum noise are ignored for this study. Descriptive statistics in conjunction with MS Excel and Rapid Miner - AI Studio (Altair, 2026) were used to generate the analysis and graphs for histograms, scatter plots, and radial plots.

Analysis Variables

The dependent variables selected for analysis were comprised of common variables for all algorithms including performance, and QC specific variables for each of the four test cases. For all QC test cases the impact of noise was collected for various input data sizes. For the QHED and for the QCCD test cases the Euclidian distance was analyzed. For the QCNN test case the probability of classification was selected. For the QCKD, the probability of tamper detection and KGR were selected. Finally, for all algorithms the following descriptive statistics were used including minimum, maximum, average, standard deviation, and variance.

Data Analysis

Data collection was performed by the QPF instrument hosting the QC algorithms. The collection of data included values for all dependent and independent variables as well as generated imagery. All data was collected in text format for ease of manipulation and analysis except for the binary imagery. For all QC test case algorithms, performance was collected by the QPF instrument during the completion of each frame processing cycle. Data was collected for each independent QPU and an average calculated for all frame performance times. In the case of image processing, the input data frame was the image. In the case of cryptography, the input data frame was the matrix of rotational strings used to calculate the QCKD cryptographic keys (Rasmusson & Barney, 2025). For all five QC test cases, the anticipated distributed QC algorithm performance is a reduction in response time since each QPU has less data to process (Kan et al., 2024; Han et al., 2024).

Data sampling for QCNN comprised the probability of correct identification for vertical and horizontal line patterns in the test image. An average probability of correct image identification was generated for all frames processed. The expected probability of correct image

classification was forecasted to diverge from the baseline probability due to quantum noise impact on image encoding (Paracha et al., 2024; Dolciami, 2020).

Euclidian distance data sampling for QHED and QCCD algorithms is generated by performing a pixel-by-pixel irradiance comparison and calculating an overall average Euclidian distance for the entire frame (Divya, 2018). In addition, the QPF instrument has the option to save the pixel-by-pixel Euclidian distance for all pixels in an image. The expected Euclidian distance showed divergence from the reference image caused by quantum readout noise affecting image encoding due to pixel flipping errors (Paracha et al., 2024; Dolciami, 2020).

In summary for all QC test case algorithms, the analysis was expected to reveal numerical patterns from the performance timing, accuracy of correct image identification, average Euclidean distance, probability of tamper detection, and KGR metric. The analysis from the numerical patterns provided supporting data to test the hypothesis and answer the research questions.

Null Hypotheses $H1_0$, $H2_0$, and $H3_0$ Testing

The null hypotheses were tested as follows. $H1_0$ – Null hypothesis for technology that is not feasible was tested and rejected by reliable processing of the QPF instrument with generation of results without QPF crashing. $H2_0$ - Distributed parallel QC algorithm performance improvement was tested and rejected if QC algorithms show performance improvement under distributed processing. Finally, $H3_0$ – Quantum noise effects do not impact quality robustness of data was tested and rejected by quantification of impact on quality of data under the presence of quantum noise.

Answering research questions RQ1, RQ2, and RQ3 was based on the analysis of tests of the corresponding hypotheses as described next. The analysis tested the hypotheses to

demonstrate a viable simulation environment ($H1_a$), simulation performance timing ($H2_a$), and robustness of quantum noise results ($H3_a$). The following sections describes how data analysis was used to test each of the hypotheses.

Hypothesis *$H1_a$* - Viable simulation infrastructure required was tested by the development of the closed-loop QPF software instrument and its ability to integrate C++ with Python QC interfaces to the QPU. The intent of the QPF instrument was to provide a reliable experimentation platform required to collect data and test the hypothesis.

Hypothesis *$H2_a$* - Quantum algorithm models' performance was tested by a collection of distributed QPU for each of the five QC algorithm test cases with comparison to the 1 QPU baseline. The 1 QPU test case was used to demonstrate that the 4, 8, and 16 QPU distributed parallel configuration offers performance improvement over the baseline.

Hypothesis *$H3_a$* - Quantum algorithm models results robustness validation was tested by analysis of the radiometric content of the image for QHED and QCCD. The analysis measured the impact of quantum noise modeling on the quality of the data when compared to a reference image using Euclidian distance. For QCNN, the hypothesis was tested by the impact of noise measured by comparing the accuracy of classifier prediction to identify horizontal and vertical lines in test images. Finally, for QCKD, the hypothesis was tested by analysis of the probability of cryptographic key tamper detection for various rotation strings including 10, 20, 33, and 64-characters. The length of 33 characters was the pristine baseline case provided with the reference QCKD algorithm. The lengths of 10 and 20-character keys evaluated the effects of a weaker cryptographic key. The 64-character key length was used to evaluate the cryptographic key improved hardness (Rasmusson & Barney, 2025).

Assumptions

This research assumes the self-error correction horizon is five years away; however, some pessimistic projections place the arrival ten years away (Vasconcelos 2020). Regardless, investment in growing knowledge of quantum computing, where supremacy over classical computing will provide an organization with strategic advantage, is worth the investment. Salient domains to strategically position an organization for arrival of mainstream QPUs include cryptography, quantum system modeling, optimization problems, physics modeling, and image processing (AbuGhanem, 2025).

The research also assumes that quantum RAM will be integrated in future hardware as well to reduce QPU to CPU traffic and maximize quantum processing performance. Current QPUs only offer raw Qubit computing resources, which, given their limited number, can only store the minimal information necessary to carry the computation with results still needing classical computer RAM storage. Having the capability to host Quantum Random Access Memory (QRAM) within the QPU will be fundamental to improving QPU performance. QRAM is an encoding method to convert classical data to quantum states, to emulate random access memory for QPUs. Hosting QRAM locally at the QPU will reduce data transfers between the CPU and QPU and maximize QPU performance (Asaka et al., 2019).

Finally, this research assumes that quantum computing will reach mainstream, and economies of scale will reduce the cost of quantum processing and the infrastructure required to integrate QPUs. The deployment of quantum technology to mainstream markets will accelerate the spread and maturation of quantum computing and its applications, making this research more relevant (AbuGhanem, 2025).

Limitations

A limitation is that noise modeling is based on actual QPU profiles generated from IBM QPU hardware. The fidelity of the quantum noise model approximates the quality of data that could be expected if the algorithm were run on QPU hardware. However, the results are an estimate that would need to be verified for the five QC test cases when running on QPU hardware.

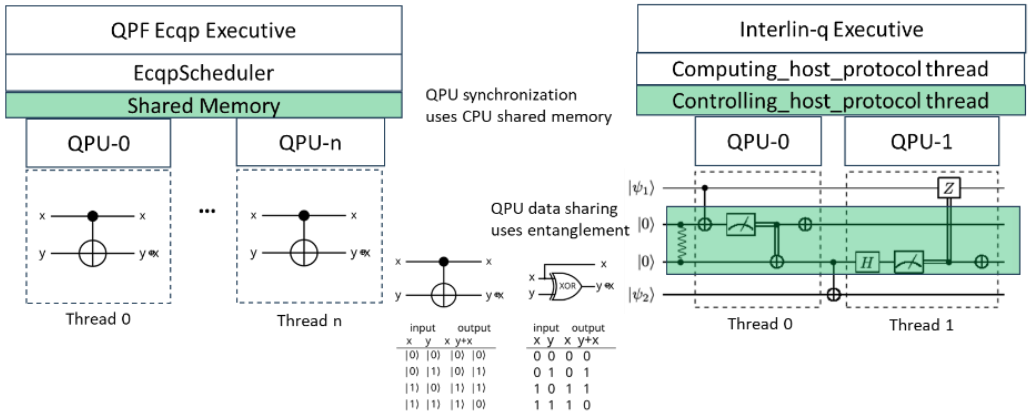
A second imitation with the QPF is that QPU synchronization is performed at the CPU level instead of using QPU-to-QPU entanglement, which would offer a faster QC to QC synchronization mechanism. Distributed parallel QPU processing is still limited by large data movements and increased CPU and inter QPU synchronization. Therefore, the same limitations that affect classical HPC solutions like multi-GPU implementations, also affect QPU solutions including Peripheral Component Interconnection Express (PCIe) bus interface arbitration bottlenecks (Harbrecht & Zaspel, 2018; Chen et al., 2018).

A third limitation is that the QPF approach for inter QPU data sharing has a disadvantage when compared to the Interlin-q framework QPU-QPU entanglement. Interlin-q can provide faster QPU-QPU data exchanges for distributed processing. Figure 29 shows the implementation of the CNOT gate on both QPF and Interlin-q to highlight the similarities and differences in architecture. A limitation of the QPF data sharing architecture is that it takes place at the CPU scheduler, which is the owner of all QPU interface threads. The data synchronization and sharing across multiple executing quantum circuits takes place on the classical CPU and does not entangle circuits across QPUs (Parekh et al., 2021). On the other hand, on Interlin-q, QPU interface thread synchronization and management take place that the CPU, but data sharing occurs directly at the circuit level using quantum entanglement. QPF and Interlin-q QPU

synchronization and data sharing use different philosophies; however, there are similarities in their architecture including executives, and a middle layer for QPU management.

Both methodologies have advantages and disadvantages, and these are relative to the algorithm implemented. For example, an algorithm like QCNN which relies heavily on training data stored on the classical CPU RAM, the QPF will offer better performance due to CPU training data proximity in cache (Bai et al., 2025). However, for QC algorithms that do not rely on data stored on the classical CPU memory, Interlin-q may have an edge. For example, for distributed Shor's algorithm, Quantum Phase Estimation (QPE), and accelerated Variational Quantum Eigen solver (VQE) Interlin-q may provide better performance results (Parekh et al., 2021).

Figure 29. *QPF & Interlin-q Architecture and CNOT Implementation Comparison.*



Note. Inter-Qubit synchronization methodologies for QPF and Interln-q. Derived from Parekh, R., Ricciardi, A., Darwish, A. and DiAdamo, S. (2021). "Quantum Algorithms and Simulation for Parallel and Distributed Quantum Computing," 2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS), St. Louis, MO, USA, 2021, pp. 9-19, doi: 10.1109/QCS54837.2021.00005.

Delimitations

A delimitation with this research is that QC algorithms are treated as unitary and are not broken down into sub-QC algorithms that could be executed as smaller circuits spread across several QPU's. The decision to not break QC algorithms into smaller components was done to facilitate comparison of results between distributed and non-distributed baseline QC algorithm configurations. Another delimitation is that parallel executing QC algorithms do not share data between QPU's to reduce data interdependency and optimize response time. In other words, a parallel executing QC algorithm runs without requiring data from other QPUs. This provides maximum performance as QC algorithms execute to completion the data without synchronization interruptions.

Another delimitation is that all data generated for the study was based on quantum simulated QPUs except for the initial pilot experiment. The pilot experiment was used to evaluate the robustness of the QPU hardware interface using the QCCD model. The initial pilot experiment was also used to test the QPU data reliability under realistic quantum noise conditions. The initial data collection used the QCCD model to collect data and to evaluate the impact of quantum noise on results. Although the QPF instrument uses simulated QPUs, the architecture can be configured to switch interfaces seamlessly to actual QPU hardware. The delimitation to rely on QPU simulation was imposed to mitigate the expensive cost of IBM cloud QPU services (IBM Pricing, 2025).

Ethical Assurances (Primary Data Collection)

This research did not include human subjects as part of the data collection process. Ethical assurances for this project were reviewed by the National University (NU) Institutional Review Board (IRB) and approved on May 30, 2024, with NU IRB number IRB-FY23-24-1128.

Summary

The chapter established the research method foundation including the applied concepts for quantum computing principles, distributed parallel processing, and the development of the experiment QPF instrument required to run the closed loop simulation to generate all experiment data. The methodology for carrying out the experiment was explained in detail, including the ten steps of the QPF instrument. Dependent and independent variables were identified along with the experiment design, including the combination of image resolution for QHED, QCNN, and QCCD, and input rotation string matrix sizes for QCKD. Additional independent variables were designed to drive the experiment including NUMA RAM allocation strategy, thread priority, thread CPU core affinity, and OS load balancing. Data preprocessing designs were addressed for each of the QC test case algorithms. The chapter also addressed QC algorithm models that require the generation of reference data to compare the performance. Model fitness evaluation or comparison to a reference model was described as well for each of the five QC test case algorithms. Analysis design was described on how it supports the underlying testing of hypotheses to provide answers to the research questions. Limitations were addressed, including the usage of simulated data and the selection of number of QPUs used to model the distributed parallelism. In summary, the chapter provided the design of the experiment methodology, instrumentation mechanisms, model evaluation, and analysis approach to provide robust testing of the hypotheses and generate the necessary test artifacts to answer the research questions.

Chapter 4: Findings

The problem addressed by this study was that a single QPU cannot solve distributed quantum computing applications that require more Qubits than available on a QPU, which limits the scalability and fidelity of image processing, cryptography, neural networks, error correction, and physics modeling applications (Davis et al., 2024; van Dijk et al., 2019; Davis et al., 2023). According to Ichikawa et al. (2023), in 2022 condensed physics applications were using more than 100 Qubits, which are approaching Amazon 105 Qubit QPU, and IBM 127 Qubit QPU (AbuGhanem, 2025). Additionally, the pervasiveness of noise effects increases as the complexity and depth of Quantum Circuit (QC) increases, which makes a case to partition and distribute an algorithm into smaller shallow QCs and spread processing across several QPUs (Kan et al., 2024). Distributed quantum algorithm processing is needed to develop applications to solve large scale QC algorithms, including decomposition into smaller problems for complex solutions that require more Qubits than available on a single QPU (Kan et al., 2024). This research developed a software framework instrument for distributed parallel processing for modeling more Qubits than available in a single QPU, including performance improvement, maturation, and integration of QC algorithms for closed-loop applications.

The purpose of this positivistic worldview quantitative constructive research is to build and validate a novel distributed parallel processing framework for quantum computing. More specifically, this research developed a framework to experimentally evaluate performance of distributed parallel QC algorithms under the impact of quantum noise to provide QPU representative results. This study aimed at modeling distributed QC algorithm execution to expedite development, and maturation through experimentation to find which combination (of independent variables) included number of QPUs, Non-Uniform Memory Allocation (NUMA)

strategy, CPU core affinity & priority, and OS scheduling offers the best performance. The (dependent variables) included response time, Euclidian distance, probability of classification, cryptographic Key Generation Rate (KGR), and probability of tamper detection. The selection of test case QC algorithms covers several domains for distributed processing applications including a Quantum Hadamard Edge Detector (QHED), Quantum Convolutional Neural Network (QCNN), Quantum Charge Coupled Device (QCCD), Quantum Crypto Key Distribution (QCKD), and Quantum Interlin-q (QILQ) for framework performance reference (Balamurugan et al., 2024; Shafique et al., 2024).

The rest of this chapter is organized by providing a description of this research plan, followed by results analysis to test the hypothesis, evaluation of the findings, research questions, research limitations, and concluding summary.

Data Preprocessing and Modeling Process Diagram

The first phase of this quantitative research plan was to identify the problem statement and the purpose and collect the body of literature work to support this study. Next, research took place to identify the GST concepts to support the design, development, and integration of the QPF data collection instrument (Johnson, 2019). The rationale for selection of the QC test algorithms was to provide a breath of relevant computer science domains for which this research can provide tangible results and practical applications (AbuGhanem, 2025).

The QPF instruments were tested next using a pilot experiment for reliability and validity on MS Windows and Linux platforms using QPU hardware and simulators to verify the instrument’s robustness under different environments. The pilot experiment also evaluated the impact of quantum noise on the results generated, and the interface logic to the IBM QPUs cloud connection. Non interrupted extensive testing of the QPF instrument was performed for hours to

ensure the interface logic to the QPUs is reliable. The next stage was comprised of development and integration of the QC parallel algorithms as shown in Figure 13.

The next stage included the design of the test space matrices for the five QC algorithm test cases for combinations of independent variables to generate data to reveal numerical pattern correlations to the dependent variables (Creswell & Creswell, 2017). The QC experiments were designed to provide a broader perspective on distribution and parallelization of quantum algorithms across four domains including QHED, QCNN, QCCD, and QCKD. The QILQ test case was used as reference to compare the QPF instrument performance to an existing parallel framework (Parekh et al., 2021).

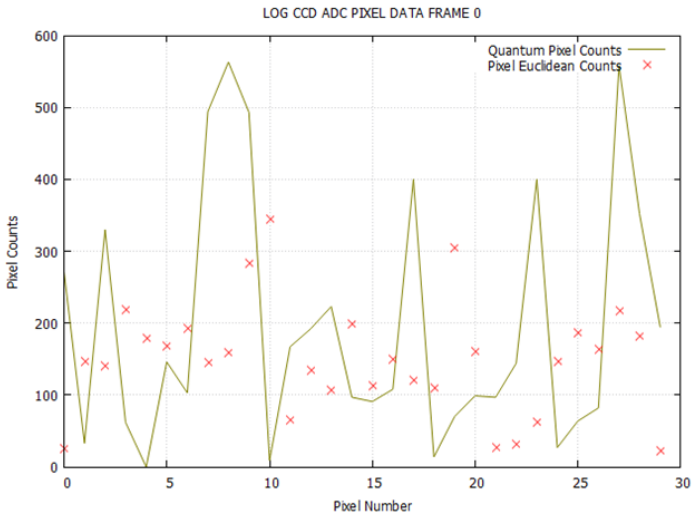
The following stage included the design, development, and execution of the experiment for running each of the five QC algorithm test cases. The data analysis phase included performing metric calculations to support the analysis stage with the expectation of identification of numerical patterns to be used to test the hypothesis and answer the research questions. The plan's last stage included final revisions and approval of the manuscript and preparation of all material used for the dissertation defense.

QFP Instrument Data Reliability and Validation

The pilot exploration phase included evaluation of the quality and robustness of the QCDD quantum algorithm performance to provide an initial assessment of expected QPF instrument reliability and results of validity. Figure 30 shows the Euclidian distance metric plotted for each of the 30 pixels calculated on QPU hardware. The experiment includes 14,428 shots executed for the QCDD for each of the 30 pixels processed. For identical images compared, the Euclidean distance should be zero for all pixels (Divya, 2018). The expected

results were Euclidian distances for all 30 pixels to be in the range of a few counts or tens of counts, or zero. However, instead the pilot experiment showed pixels with Euclidian distance ranging from 0 to 550 counts, which provides and assessment of how quantum noise affects image encoding and how it may affect image processing (Dolciami, 2022).

Figure 30. *Euclidian Distance of Theoretical ADC versus QPU Generated Pixels.*



Note. Euclidian distance shows divergence of pixels by pixel between quantum image and reference. A Perfect match between quantum and reference generated image would show a Euclidian distance of zero for all pixels.

Ten experimental runs were performed with each yielding equivalent results, which outline the impact of quantum noise on the QPU and non-repeatable results. This phase outlined the necessity of modeling quantum noise on the QPF using QPU simulation to provide a higher results fidelity. The pilot runs reconciled as 14,428 shots times 30 pixels giving 432,840 runs (shots), which performed the QCCD radiometric Analog to Digital Conversion (ADC) without crashing the QPF instrument. Overall, the pilot experiment was able to demonstrate the QPF instrument's feasibility, reliability, and validity to generate and collect all experiment data.

In addition, to strengthen the posture of this dissertation, a technical article paper was developed for distributed “Quantum Parallel Processing Framework for Image Processing Applications.” The article was presented on July 22, 2025, at the IEEE International Symposium on Autonomous Decentralized Systems (ISADS) hosted at the University of Arizona (Salcedo & Ahmed, 2025). The article received positive feedback from the academic community with interest for collaboration in future research.

Statistical Analysis

The objective of this section is to obtain numerical patterns from analysis for the impact of noise. The selected statistical method used distribution analysis of dependent variables under the effects of quantum noise. The selected tools, to facilitate analysis are Rapid Miner - AI Studio (Altair, 2026) and MS Excel, which include descriptive statistics to glean minimum, maximum, average, standard deviation, and variance. The algorithms analyzed included the QHED, QCNN, QCCD, and QCKD. The data sets were pre-processed by removal of the first seven lines of data to remove the header, and allow for ease of data consumption. All datasets used for the statistical analysis were imported and pre-processed format with removal of data not required for the analysis. The tools utilized for the analysis included graphs of the independent variables. A challenge in this statistical analysis exercise includes the ability to discern any new meaningful and statistically significant numerical patterns, which may be interpreted incorrectly or may not be significant to the overall operation of the QC test cases.

Model Validation and Parameter Tuning

The parameters selected for analysis were the independent variables including Euclidian distance, image classification, KGR, and probability of tamper detection. For all QC algorithms tested, the results were compared to the corresponding baseline to determine the impact of noise

over the duration of the run. The QILQ test case was not included in the statistical analysis since the purpose of this test case was to evaluate performance improvement of the QPF and not consider noise. Although quantum noise at the Qubit level is the same for all QC test cases, the noise effects manifest differently for each algorithm. For example, for the QHED and QCCD, noise manifests as distortions in the image caused by pixel flips due to readout errors (Dolciami, 2022; Mastriani, 2020). For QCNN, the effects of noise negatively impact classification of correctly identifying the image in the test pattern. For QCKD, interestingly and unexpected, the QC algorithm is impervious to noise as QCKD relies on random Qubit flips during the generation of the QCKD key (Rasmusson & Barney, 2025).

QHED Data Modeling

The initial statistical analysis yielded poor results since the dynamic range of the pixel data was limited to only 1-bit color depth. This is a limitation of the design of the QHED algorithm, where pixels are monochromatic, i.e., black (0), or white (1). Several approaches were attempted including clustering, and data distribution, which yielded little information on data patterns. Therefore, a simple descriptive statistical analysis was performed. The QHED QC algorithm was impacted by quantum noise with distortion on the image caused by pixel flips incurred by the Qubit readout errors (Dolciami, 2022; Mastriani, 2020). For a 512x512 image the total number of pixels impacted by distortions due to quantum noise was 5,553 out of a total of 262,144 pixels. The analysis shows that overall QPUs can be affected by noise. There was no meaningful graph to plot due to the large number of data points; Therefore, a brief descriptive statistics analysis was provided as shown on Table 14, from where an initial assessment of image quality was gleaned. The percentage of pixels affected by quantum noise was 2%, which may be sufficient for image processing algorithms. In spite, of image distortions, image processing

algorithms may be impervious to quantum noise for some applications since the image will already be noisy caused by sampling of the CCD device. For example, the QHED algorithm is still able to perform edge detection in the presence of quantum noise as shown in Figure 36. The rationale is that in real-world applications image processing algorithms are expected to encounter distortions in the image. For example, all EO sensors, CCD and CMOS, add various forms of image distortions including temporal noise, dark noise, bad pixels, fixed pattern noise, responsivity effects, and optical distortion that affect the quality of the image (Konnik & Welsh, 2014).

Table 14. *QHED Descriptive Statistics.*

MIN	MAX	AVERAGE	STDEV.S	STDEV.P	VAR	VAR.S	VAR.P
0	1	0.021183	0.143994	0.143994	0.020734	0.020734	0.020734

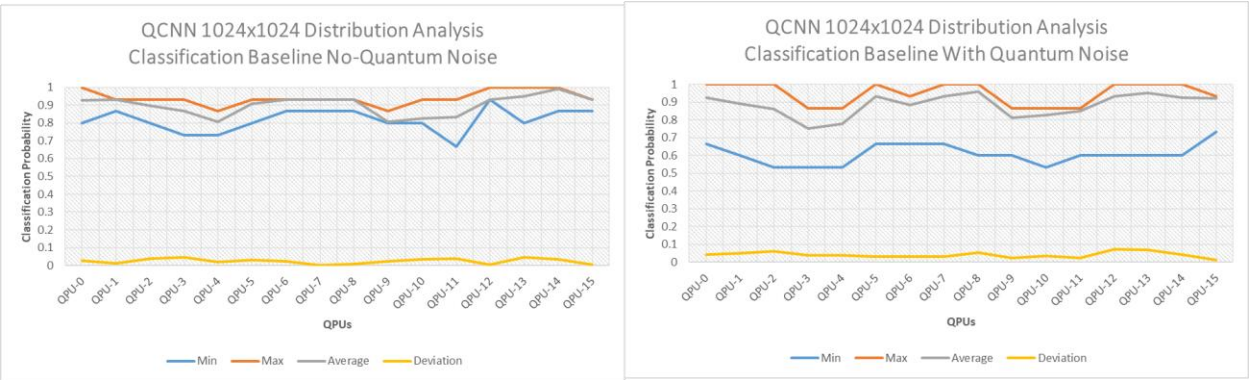
Note. The reference data set used for this analysis was 20251214.001.

QCNN Data Modeling

The QCNN QC algorithm is impacted by quantum noise in its ability to correctly identify the test pattern in the image. The challenges encountered by the QC algorithm are rooted in pixel flips caused by Qubit readout errors (Dolciami, 2022; Mastriani, 2020). As shown on Figure 31, the QC algorithm shows performance classification variability in the left image, which is exacerbated by quantum noise on the image on the right. Classification performance was analyzed for all 16 QPUs. The analysis showed that all QPUs can be affected by noise. Statistical analysis was performed for all resolutions including 32x32, 256x256, 1024x1024 and 2048x2048, with similar conclusions as described for the 1024x1024 test case. An observation is

that the classification accuracy without quantum noise was 90%, and dropped to 88% under the presence of quantum noise.

Figure 31. *QCNN Probability of Correct Classification Statistical Distribution Analysis.*



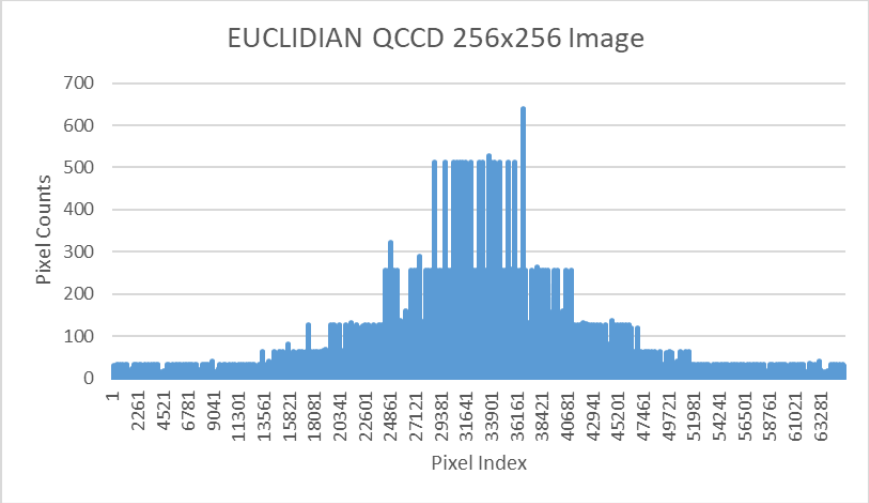
Note. The reference data set used for this analysis was 20251214.001.

QCCD Data Modeling

The QCCD QC algorithm was impacted by quantum noise shown as pixel flips incurred by Qubit readout errors (Dolciami, 2022; Mastriani, 2020). For a 1024x1024 image the total number of pixels impacted by encoding distortions were 5,553 out of a total of 262,146 pixels. The analysis showed that overall QPUs can be affected by noise. An interesting insight is that the quantum noise appeared to affect pixels with larger values more than pixels with smaller pixel counts. The center of the image per Figure 24 is where the majority of the signal pixels are located, hence the clustering of pixel values around image pixel index 27,121 to 40,681 as shown on Figure 32. An observation is that the ADC model of the QCCD uses a quantum adder circuit, which encodes a pixel value as an 8-bit binary number, which are prone to Qubit encoding errors. Therefore, the larger the pixel value, the more Qubits used to encode a number, the

greater probability of encountering pixel encoding errors, which leads to a larger Euclidian distance (Dolciami, 2022; Mastriani, 2020).

Figure 32. *QCCD Euclidian Statistical Distribution Analysis.*

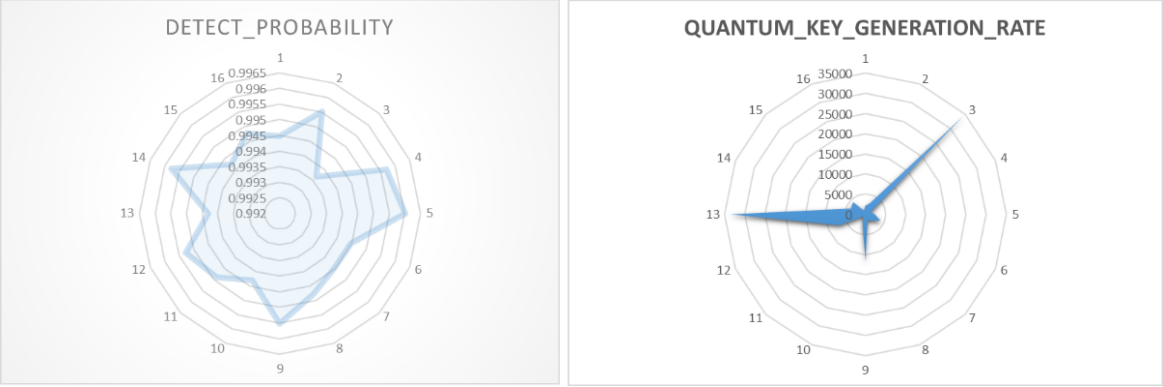


Note. The reference data set used for this analysis was 20251214.001.

QCKD Data Modeling

The QCKD QC algorithm was not impacted by quantum noise since the key cryptographic key relies on random Qubit flips to generate a random number. As shown on Figure 33, the QC algorithm showed variations when the data is analyzed across 16 QPUs. The analysis showed that overall QPUs can be affected by noise. An interesting observation regarding the key generation rate is that the OS scheduling favored QPU 3, 9, and 13, with QPU 13 outperforming every other QPU in completing the assigned tasking.

Figure 33. *QCKD Probability Tamper Detection Statistical Analysis.*



Note. The reference data set used for this analysis was 20251214.001.

Results

The results were organized by research questions and hypotheses and included the analysis for the five QC algorithm test cases. Analysis was broken down by QC algorithm test cases including QHED, QCNN, QCCD, QCKD, and QILQ. The analysis provided quantitative support to test the null and alternate hypotheses and provided a foundation for answering the research questions. This constructive research provided experiments and correlations between independent and dependent variables to draw conclusions from the numerical test patterns. From these results, the hypotheses were tested for quantum instruments development feasibility, distributed parallel QC algorithm evaluation and impact of quantum noise on quality of results (Dolciami, 2022; Iyengar et al., 2020). The QPF instruments software and test artifacts data were stored on a repository per Appendix C.

All QC algorithms share a common set of independent variables as listed on chapter 3 tables 3 through 13. The selection of independent variables was designed to obtain performance and quality of QC results under various configurations with the purpose of obtaining

mathematical patterns from where conclusions can be made to support hypotheses and answer research questions. In addition, algorithms have specific independent variables including Euclidian distance, probability of correct image classification, KGR, and tamper detection probabilities. The independent variables in the common configuration were optimized to reduce QC response time. The noise type configures the QPU to enable quantum noise modeling for the QC. The noise type can be fast quantum gate, readout (type 16), or default noise model (type 20), faster quantum gate (type 22). Noise models are tailored to favor performance based on algorithm type. The data size denotes the image size in pixels, or the input data matrix size. RAM NUMA is enabled to force allocation to memory banks as close as possible to the allocating QC CPU core. OS thread priority is set for Realtime (RT) processing to favor performance and pre-empt other processing of lesser priority. A load balance of 1 tells the OS to relocate threads to CPU cores, which have the probability of completing the work faster. A load balance of type 2 denotes user assigned load balancing, allocating one QC per CPU core using the modulus function. An explicit load balance of CORE x, where x is a core number, denotes user assigning CPU core affinity to execute QC without changing core during run.

The analysis showed that NUMA memory allocation strategy provides the best performance as shown on tables 16 through 29 as all memory allocated for any QPU interface thread is pulled from the processors' closest memory banks (Bai et al. 2025; Han & Wang, 2024). QPU interface thread priority set to real-time benefited the thread execution time as preferential OS quantum time slices was provided to the thread (Basha et al., 2023). In many QC test cases, load balancing provided by the OS allowed for better performance as the thread was moved during context switching to a CPU core that completed processing quicker (Bai et al. 2025; Han & Wang, 2024).

QHED Data Analysis

The following analysis is provided for the data generated by the test matrix in Figure 15. Analysis showed that with NUMA enabled, memory allocation time was reduced since allocation took place closer to the processor core requesting memory instead of the OS interleaving allocation between processor memory banks (Bai et al. 2025; Han & Wang, 2024). Real-time thread priority favored the OS quantum time slice to allow thread to run more deterministically (Basha et al., 2023).

Table 15 shows the performance timing collected with the 4 QPU configuration, showing the best performance over the 1 QPU baseline. The 8 and 16 QPU configurations also demonstrated performance improvement over the 1 QPU baseline. Noise type 16 enabled Qubit gate and readout noise models, which manifested as pixel flip errors (Mastriani, 2020; Dolciami, 2022).

Table 15. *QHED Performance Data Collection.*

QHED INDEPENDENT VARIABLES							Dataset: 20250306.001	
NOISE TYPE	*DATA SIZE*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	PERFORMANCE TIME(SEC) *1 QPU BASE*	*4 QPU*	*8 QPU*	*16 QPU*
16	32x32	Enable	RT	OS 1	152.1004393	**88.13984513**	111.4303517	122.0894825
16	512x512	Enable	RT	OS 1	3264.225616	**2184.446763**	2715.83718	2755.105819
16	1024x1024	Enable	RT	OS 1	1199.957851	**850.6096139**	1099.635228	1071.929961
16	2048x2048	Enable	RT	OS 1	2506.752055	**1682.131569**	2194.635295	2227.087846

Note. Best performance for 4 QPU configuration shown. Independent variables shown in *italics and purple text*. Best performance in **bold**.

Table 16 shows the average Euclidian distance when compared to a no quantum noise reference image. The lowest Euclidian distance was 0.020507813 for 32x32 image on 8 QPUs, and the highest 0.021065235 for 1024x1024 image on 4 QPUs showing that nor the number of QPUs spawned nor image size affected the quality of the image significantly.

Table 16. *QHED Euclidian Distance and Performance Improvement Data Collection.*

				QHED DEPENDENT VARIABLES				Dataset: 20250306.001		
DATA	RAM	TR	LOAD	*FRAME QVERAGE EUCLIDIAN DISTANCE (PIXELS)*				*PERFORMANCE IMPROVE % FROM BASELINE*		
SIZE	NUMA	PRI	BALA	1 QPU BASE	4 QPU	8 QPU	16 QPU	4QPU	8QPU	16QPU
32x32	Enable	RT	OS 1	0.025390625	0.021484375	0.020507813	0.025390625	**42%**	27%	20%
512x512	Enable	RT	OS 1	0.020983582	0.020984879	0.021018524	0.020976028	**33%**	17%	16%
1024x1024	Enable	RT	OS 1	0.021007729	0.021065235	0.020942497	0.021011257	**29%**	8%	11%
2048x2048	Enable	RT	OS 1	0.020982361	0.021007299	0.020967436	0.020932007	**33%**	12%	11%

Note. Best performance for 4 QPU configuration shown. Legend: TR=thread, PRI=priority, BALA=balance. Dependent variables shown in *italics and blue text*. Best performance in **bold**.

In summary, analysis showed that the QPU configuration that yielded the best performance was 4 QPUs, with 8 and 16 QPUs also outperforming the 1 QPU baseline.

Figure 34 shows the performance improvement gained by 4, 8, and 16 QPUs over the 1 QPU baseline with the best performance achieved by the 4 QPU configuration. The results were consistent with the 4 QPU configuration showing the best performance for all resolutions.

Figure 34. *QHED Performance Timing Plots.*

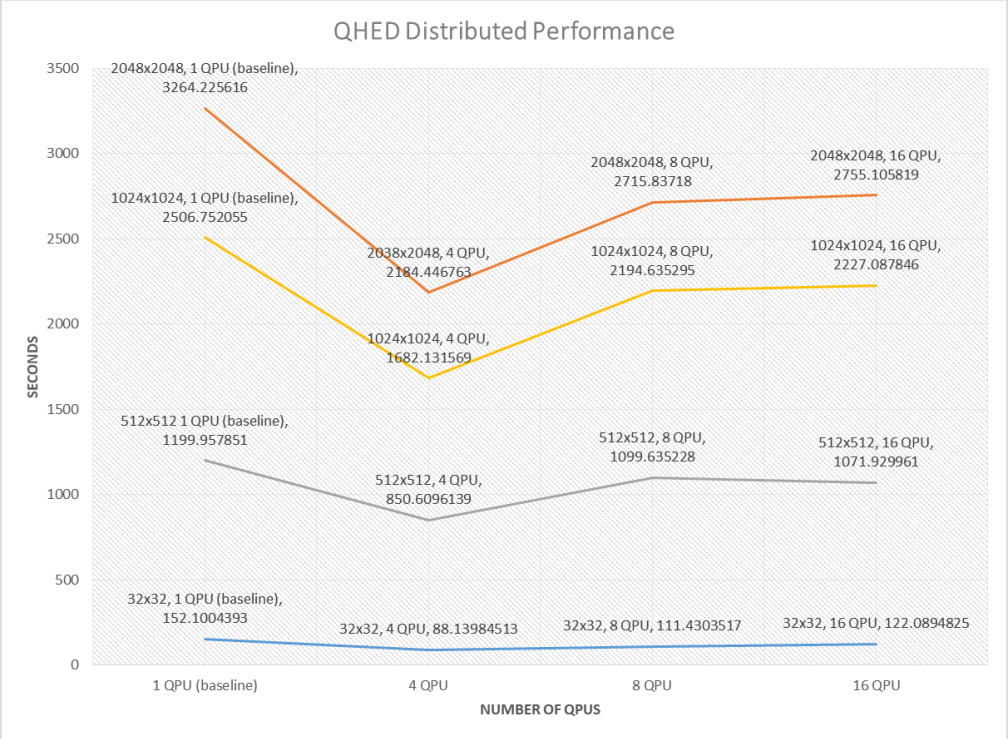
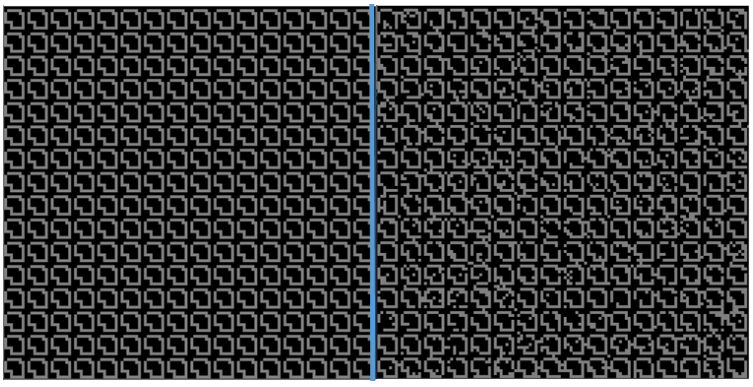


Figure 35 shows the visual distortion of the image caused by quantum readout noise manifested as pixel flips due to pixel encoding errors (Mastriani, 2020).

Figure 35. *QHED Reference on Left vs Readout Noise on Right.*



Note. Readout noise shown as a flip in Qubit states, which results in a pixel flips on the right image. 128x128 pixel image shown

An additional metric used was the 2-group t-test for comparison of the QHED quantum and reference images, which assessed the divergence of the means between the two images as shown in Table 17. The t-test was also used to determine the performance improvement between the 1 QPU baseline and 4, 8, and 16 QPU datasets. In addition, the t-test produced a value of $\alpha < 0.05$, which was used to reject the $H1_0$ hypotheses for distributed performance improvements not significant.

Table 17. *T-Test for QHED 32x32 for 15,360 Samples Used for H_2 Tests.*

T-Test Description	*BASELINE_TIME*	*TIME*
Mean	0.148712728	0.118837
Variance	2.98347E-05	0.001446
Observations	15360	15360
Hypothesized Mean Difference	0	
df	1040	
t Stat	24.59540053	
P(T<=t) one-tail	5.8744E-106	
t Critical one-tail	1.646320104	
P(T<=t) two-tail	1.1749E-105	
t Critical two-tail	1.962247626	

Note. Mean and variance provide a metric for the impact of read out noise.

Statistical significance was achieved with a cutoff value of $\alpha = 5\%$, which required a dataset of at least 14,428 samples. The hypothesized mean difference of zero assumed there were no performance improvements between the baseline reference case and the experimental data per Table 17. The test used for this analysis was a one tail test to verify if QC distributions over multi QPUs would yield results that increased performance from the baseline 1 QPU dataset. The sample size assured there were less than 1 in 20 chances the empirical data was generated randomly. The samples used in the t-test calculation were values generated for each pixel, which reconciled as 32x32 pixels per image times 15 frames, which yielded 15,360-pixel samples. The

t-test showed that the pixel data for the reference and quantum generated empirical image were statistically different. The quantum and reference pixel mean, and variance provided a comparison metric to evaluate the impact of quantum noise. In summary, the differences in means and variance showed a definite performance improvement from the 1 QPU dataset.

QCNN Data Analysis

NUMA enabled allocation benefited the performance of the QCNN algorithms since all memory was allocated closer to the calling QPU interface thread core. Therefore, reducing the time for accessing test data from memory (Bai et al. 2025; Han & Wang, 2024). A QPU interface thread priority set to real-time provided preferential CPU cycles, which yielded more deterministic timing and better performance (Basha et al., 2023). In this case, thread load balancing specified by the user instead of OS offered better performance. Setting a QPU affinity to a particular core allowed the thread to capitalize on core's Last Level Cache (LLC) for training data lookups, which improved performance (Bai et al. 2025; Han & Wang, 2024).

Table 18 shows the performance timing collected with 8 and 16 QPUs showing the best percentage performance over the 1 QPU baseline. The 4 QPU configuration also demonstrated performance percentage improvement over the 1 QPU baseline. Noise type 20 enabled Qubit readout and default QPU noise modeling, which manifested in test image corruption due to Qubit pixel encoding errors (Mastriani, 2020; Dolciami, 2022). Overall, the best performance was split between 8 and 16 QPUs.

Table 18. *QCNN Performance Data Collection.*

QCNN INDEPENDENT VARIABLES							Dataset: 20250629.001	
NOISE TYPE	*DATA SIZE*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	QC PERFORMANCE TIME (SEC)			
					1 QPU BASE	*4 QPU*	*8 QPU*	*16 QPU*
20	4x8	Enable	RT	OS 1	41706.46813	27097.01852	27609.0943	**26849.43938**
20	64x128	Enable	RT	OS 1	47564.31025	31255.2016	**30104.56546**	30353.12688
20	128x256	Enable	RT	OS 1	94965.08297	61989.84762	60333.38915	**60917.49558**
20	256x512	Enable	RT	OS 1	380787.6495	248675.7281	**239381.3476**	244582.9338

Note. Independent variables shown in *italics and purple text*. Best performance for 4 QPU configuration shown in **bold**.

Table 19 shows increasing to 4 QPUs or more offers improved performance over the 1 QPU baseline. The analysis showed that 8 and 16 QPU configuration also outperformed the 1 QPU baseline. Table 19 shows the percentage of performance improvement by 4, 8, and 16 QPUs with a best performance of 37% achieved by the 8 QPU configuration.

Table 19. *QCNN Frame Classification Accuracy and Performance Improvement.*

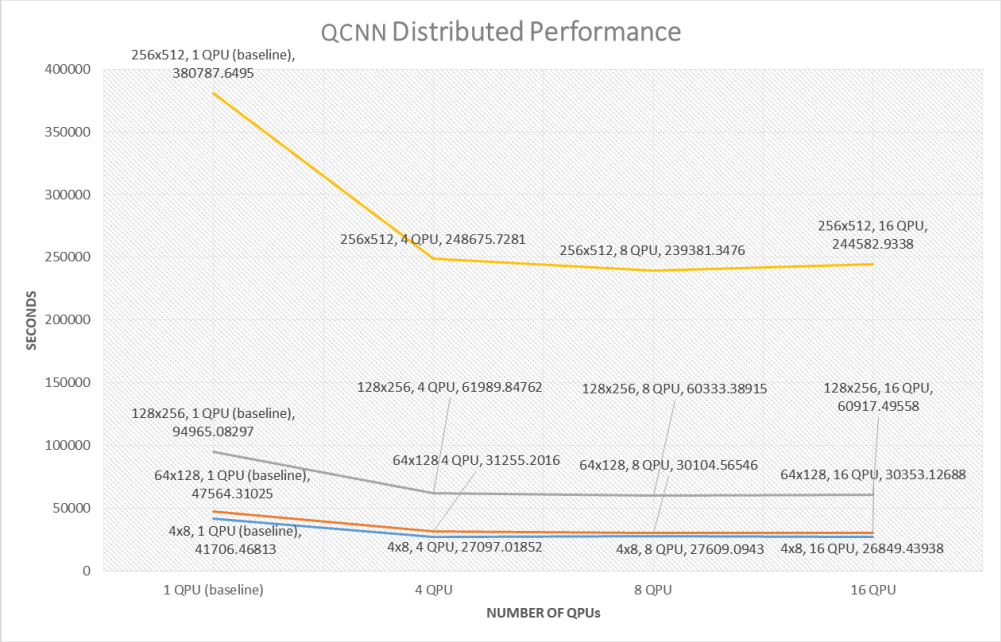
QCNN DEPENDENT VARIABLES										Dataset: 20250306.001
DATA SIZE	RAM NUMA	TR PRI	LOAD BALANCE	*FRAME AVE CLASSIFICATION ACCURACY %*				*PERFORMANCE IMPROVE % FROM BASELINE*		
				1 QPU BASE	4 QPU	8 QPU	16 QPU	4QPU	8QPU	16QPU
4x8	Enable	RT	OS 1	0.857945307	0.825720621	0.868144863	0.868144863	35%	34%	**36%**
64x128	Enable	RT	OS 1	0.868144863	0.933333333	0.933333333	0.933333333	34%	**37%**	36%
128x256	Enable	RT	OS 1	0.933333333	0.866666667	0.933333333	0.933333333	35%	**36%**	**36%**
256x512	Enable	RT	OS 1	0.866666667	0.933333333	0.866666667	0.933333333	35%	**37%**	36%

Note. Best performance for 8 QPU configuration shown in **bold**. Legend: TR=thread, PRI=priority. Dependent variables shown in *italics and blue text*.

Figure 36 shows the performance improvement by the 4, 8, and 16 QPUs over the 1 QPU baseline with the best performance achieved by the 16 QPU configuration. The 16 QPU configuration offered the best performance for all input image sizes. The 4 and 8 QPU

configuration showed marginal lower gains when compared to the 16 QPU configuration. All 4, 8, and 16 QPU configurations showed performance improvement when compared to the 1 QPU baseline.

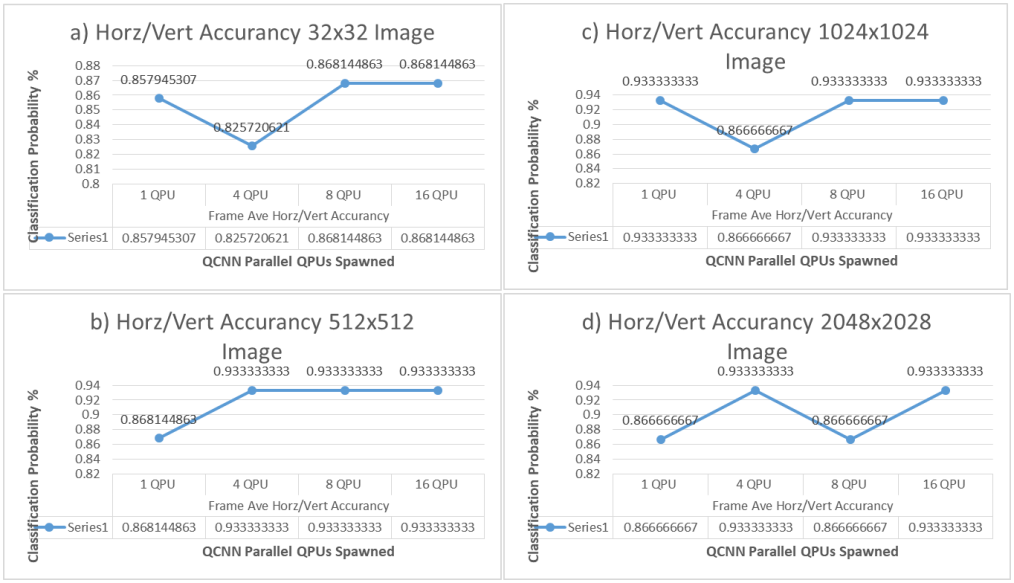
Figure 36. *QCNN Performance Timing Plots.*



The reference classifier prediction accuracy in absence of quantum noise was 93% for correctly identifying horizontal and vertical line patterns in an image (QiskitEcosystem, 2025). The QCNN experiment was optimized to reduce processing time and thus only one configuration of quantum noise was modeled including Qubit readout and default QPU noise modeling, which manifested in test image corruption due to Qubit pixel encoding errors (Mastriani, 2020; Dolciami, 2022). The impact of image size showed linearly across all QPU configurations with the 256x512 image taking the longest time, which could be mitigated by distributing the image processing across more than 16 QPUs.

Typical detection accuracies on collected data ranged from 82% to as high as 93% with a negative impact on trained data by presence of quantum noise on the test data as shown in Figure 37. The analysis showed that for a small 32x32 image the impact of noise appeared to be more significant as shown in Figure 38a. For images of medium size including Figures 37b and Figure 37c, the accuracy of the QCNN improved and reached 93% and stayed as low as 86% accuracy, which appears to correlate with the larger number of QPUs allocated. An exception is the large image 2048x2048 on Figure 37d, which showed an accuracy of 93% for 4 and 16 QPUs, and a degraded prediction accuracy of 86% for the 1 and 8 QPU configuration. The variations were due to the quantum error model included in the simulation (Mastriani, 2020; Dolciami, 2022).

Figure 37. *QCNN Classifier Prediction Accuracy Plots.*



QCCD Data Analysis

NUMA enabled allocation strategy yielded the best performance since each QPU interface thread did not have to go over the QPI inter-processor interconnect bridge and incur the cost of accessing image memory allocated on the other CPU memory banks (Bai et al. 2025; Han & Wang, 2024). The QCCD algorithm applied the ADC model to each pixel, and thus the overall

memory access time for this algorithm was considerable. QPU interface thread priority set to real-time provided a thread preferential OS quantum time slice yielding better performance (Basha et al., 2023). Finally, letting the OS provide the processor load balancing also yielded the best performance since the OS placed the thread at a core whose processing load was forecasted to complete quicker (Bai et al. 2025; Han & Wang, 2024).

Table 20 shows the raw performance data collected for the QCCD test case over the 1 QPU configuration baseline. Noise type 22 enabled Qubit faster readout error QPU noise modeling, which manifested as pixel flips due to Qubit pixel encoding errors (Mastriani, 2020; Dolciami, 2022). Smaller image resolutions were collected to expedite the experiment. The resolutions collected included 32x32, 64x64, 128x128, and 256x256 pixels. For the 1 QPU 32x32 pixel image, the data collection took 27,309 seconds, or 7.6 hours. The larger image 1 QPU 256x256 data collection took 218,476 seconds or 60.1 hours.

Table 20. *QCCD Performance Data Collection.*

QCCD INDEPENDENT VARIABLES					Dataset: 20250627.001 Dataset:20251123.001-004			
NOISE TYPE	*DATA SIZE*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	QC PERFORMANCE TIME (SEC) *1 QPU BASE*	*4 QPU*	*8 QPU*	*16 QPU*
22	32x32	Enable	RT	OS 1	27309.54288	5507.764928	**2308.3096**	2309.125227
22	64x64	Enable	RT	OS 1	54619.08577	14315.43853	6285.104969	6288.524864
22	128x128	Enable	RT	OS 1	109238.1715	22051.31186	9761.123594	9676.747681
22	256x256	Enable	RT	OS 1	218476.3431	40134.09255	28198.97893	17887.90361

Note. Independent variables shown in *italics and purple text*. Best performance in **bold**.

Table 21 shows that the 8 QPU configuration yields the best performance with 91.55% performance improvements above 4 QPUs also surpassing the 1 QPU baseline. Analysis of

Euclidian distance assessed the impact of quantum noise on the ADC conversion model (Divya, 2018; Mastriani, 2020). Table 21 shows the average Euclidian distance when compared to a no quantum noise reference image. The average Euclidian distance divergence ranged from 13 to 18 out of 255 maximum counts per pixel for an 8-bit image. The lowest Euclidian distance was 3.042785645 for 4 QPUs and the highest 18.95898438 pixels for 8 QPUs.

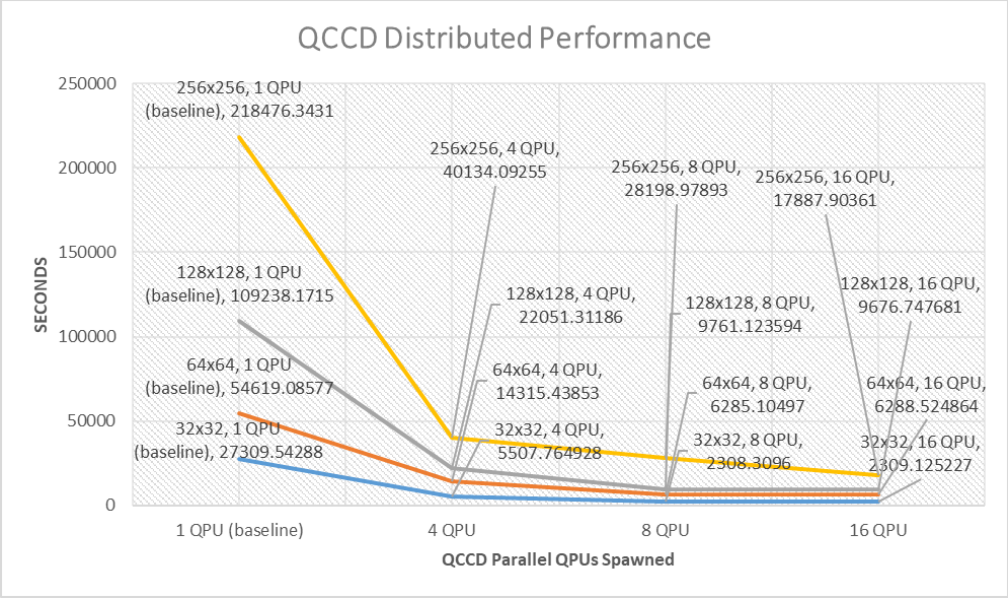
Table 21. *QCCD Euclidian Distance and Performance Improvement Data Collection.*

				QCCD DEPENDENT VARIABLES				Dataset: 20250306.001 Dataset: 20251123.001-004		
DATA	RAM	TR	LOAD	*FRAME AVERAGE EUCLIDIAN DISTANCE (PIXELS)*				*PERFORMANCE IMPROVE % FROM BASELINE*		
SIZE	NUMA	PRI	BALANCE	1 QPU BASE	4 QPU	8 QPU	16 QPU	4QPU	8QPU	16QPU
32x32	Enable	RT	OS 1	13.92773438	10.99609375	13.34570313	18.95898438	79.83%	**91.55%**	91.54%
64x64	Enable	RT	OS 1	12.78123123	6.09765625	7.616210938	6.529541016	73.79%	**88.49%**	88.49%
128x128	Enable	RT	OS 1	8.242342223	3.042785645	3.312988281	3.311401367	79.81%	91.06%	**91.14%**
256x256	Enable	RT	OS 1	11.63445343	4.073318481	3.983230591	3.983230591	81.63%	87.09%	**91.81%**

Note. Best performance for 8 QPU configuration in bold. Used noise model 22. Legend: TR=thread, PRI=priority. Dependent variables shown in *italics and blue text*. Best performance in **bold**.

Figure 38 shows the performance improvement for 4, 8, and 16 QPUs over the 1 QPU baseline with the best performance achieved by the 16 QPU configuration. Interestingly, the performance between 8 and 16 QPUs for 32x32 and 64x64 resolutions was comparable with 8 QPUs showing marginal improvements over the 16 QPU configuration. The results showed consistent scalability up to 16 QPUs. For the QCCD test case, increasing the number of QPUs showed a scaling in performance over 1 and 4 QPUs with marginal improvement from 8 to 16 QPU configurations.

Figure 38. *QCCD Performance Plot.*



Analysis of the t-test median between the QCCD computed image and the reference CCD image assessed the quality of the quantum generated image as shown in Table 22.

Table 22. *T-Test for QCCD 32x32 for 15,360 and Two Variables to Test H_1 & H_3.*

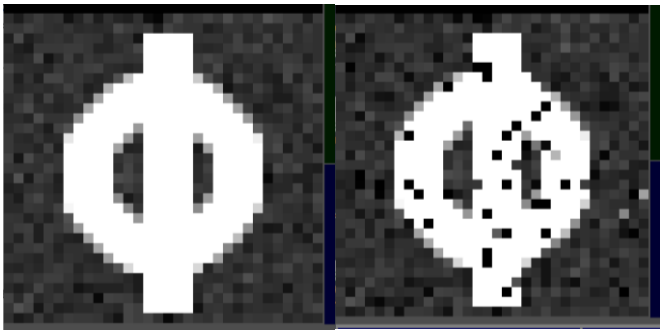
T-Test Description	*PIXEL*	*PIXEL_REFERENCE*
Mean	74.51855	85.16503906
Variance	13334.04	10594.65993
Observations	15360	15360
Hypothesized Mean Difference	0	
df	2020	
t Stat	-2.2024	
P(T<=t) one-tail	0.013875	
t Critical one-tail	1.645608	
P(T<=t) two-tail	0.02775	
t Critical two-tail	1.961139	

Note. Mean and variance provide a metric for the impact of read out noise.

The assumption was that the comparison of the sampled and measured tests sets had unequal variances. The test utilized for this analysis was a one tail test to verify if the impact of quantum noise on the results was greater than the (non-noise) baseline. The hypothesized mean difference of zero assumed there was no performance improvement between the baseline case and the experimental data. Analysis of the t-test mean between the QCCD ADC image and the reference CCD ADC image assessed the statistical significance of the generated empirical imagery. The t-test showed that the means for pixel values for the reference image and the quantum empirical image were statistically different. This showed that the samples collected for the QCCD test met the statistically significant minimum of 14,428 samples to achieve 1 in 20 probability results occurring by chance (Hazra & Gogtay, 2016).

Figure 39 shows that the distortion on the image caused by the quantum readout noise appeared as pixel flips due to quantum errors in the image encoding QC (Dolciami 2022; Mastriani 2020).

Figure 39. *Reference 32x32 CCD ADC on Left versus QCCD ADC Output on Right.*



Note. Quantum readout noise manifests as pixel flips on the right image.

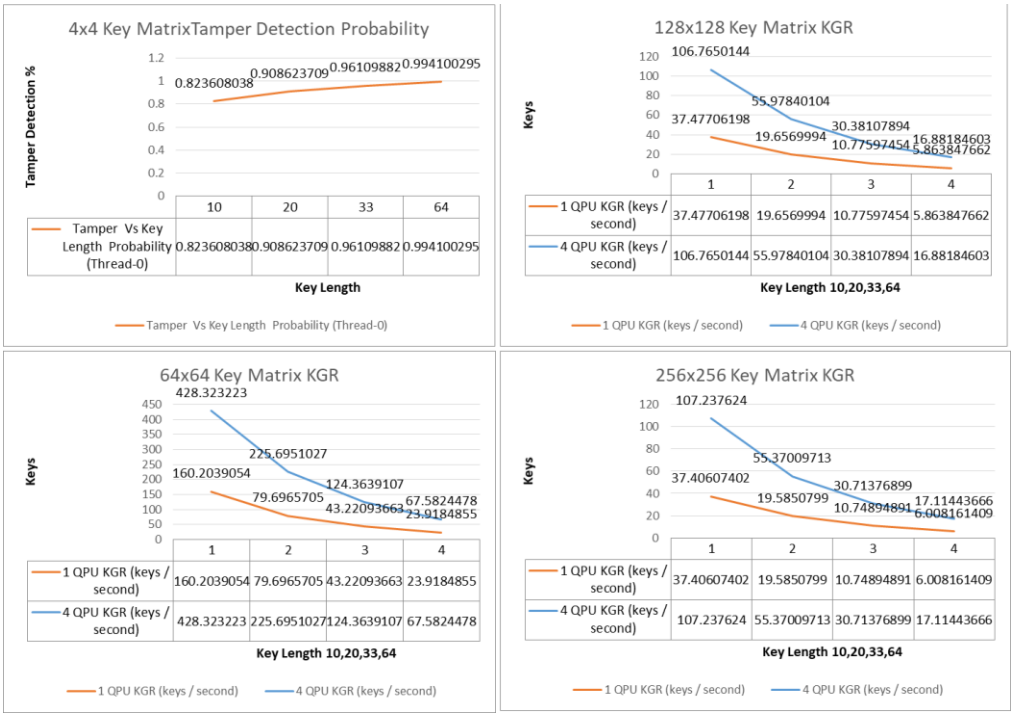
QCKD and KGR Data Analysis

The experiment data collection and analysis included matrices comprised of QCKD rotational strings for 4x4, 64x64, 128x128, and 256x256 datasets. The testing methodology for

testing QCKD encryption used tampering of the cryptographic key 100% of the time with the hope that the QCKD configuration would catch the eavesdropper interdiction.

Figure 40 shows a summarized comparison between a 1 QPU versus a 4 QPU KGR. The KGR was computed for the four QCKD rotational input matrices including 32x32, 64x64, 128x128, and 256x256 sizes. The KGR was directly proportional to the performance of the distribution of the algorithms. In other words, the faster the QC algorithm executed, the greater the KGR (Bai et al. 2025; Han & Wang, 2024). KGR was inversely proportional to the length of the rotational string. The longer the rotational string, the longer it took to process and generate the QCKD.

Figure 40. *QCKD Key Generation Rate vs QPU Plots for 1-GPU versus 4-GPUs.*



QCKD 4x4 Data Analysis

Table 23 shows the raw performance data collected for the QCKD 4x4 input matrix test case.

Table 23. *QCKD Performance Data Collection for 4x4 Rotation Matrix.*

QCKD INDEPENDENT VARIABLES								Dataset: 20250712.001	
NOISE TYPE	*DATA SIZE*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	*ROT STR*	QC PERFORMANCE TIME (SEC) *1 QPU BASE*	*4 QPU*	*8 QPU*	*16 QPU*
20	4x4	Enable	RT	OS 1	10	387.7700202	141.9573596	**139.9325974**	142.6738551
20	4x4	Enable	RT	OS 1	20	744.5988464	260.4430082	**198.1555686**	267.0101416
20	4x4	Enable	RT	OS 1	33	1353.063831	481.4080701	**476.8703172**	483.9669518
20	4x4	Enable	RT	OS 1	64	2429.211632	867.4797764	**862.3295996**	873.2533221

Note. Noise type 20. ROT=rotation. STR=string. Independent variables in *italics and purple text*. Best performance in **bold**.

Table 24 shows the performance improvement for 4, 8, and 16 QPUs over the 1 QPU baseline configuration. All QPU configurations showed improvement over the baseline with 8 QPUs demonstrating more consistent performance across the four rotation string lengths with 64-73% improvement. Although the performance improvement from baseline was over 60%, the scalability performance gains were marginal between 4, 8 and 16 QPUs. Tamper detection improved as the rotational string length increased; thus, yielding a hardened cryptographic stance. KGR dropped as expected with longer processing times for longer rotational strings.

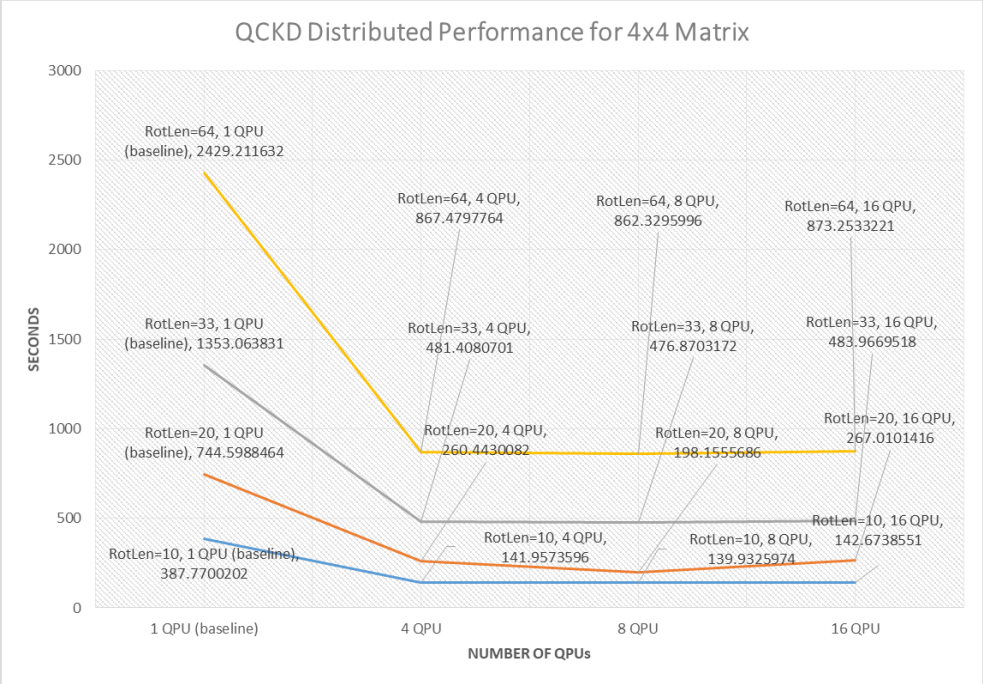
Table 24. *QCKD Tamper Detection and Performance Improvement for 4x4 Matrix.*

QCKD DEPENDENT VARIABLES								Dataset: 20250306.001		
DATA SIZE	RAM NUMA	TR PRI	LOAD BALANCE	ROT STR	*TAMPER DETECT %*	1 QPU BASE *KGR (KEY/SEC)*	4 QPU *KGR (KEY/SEC)*	*PERFORMANCE IMPROVE % FROM BASELINE* 4QPU	8QPU	16QPU
4x4	Enable	RT	OS 1	10	0.00%	37.20762113	101.6361536	63%	**64%**	63%
4x4	Enable	RT	OS 1	20	8.50%	19.37687665	55.39791642	65%	**73%**	64%
4x4	Enable	RT	OS 1	33	13.75%	10.66320721	29.97041574	65%	**65%**	64%
4x4	Enable	RT	OS 1	64	17.05%	5.939375478	16.63208802	64%	**65%**	64%

Note. Noise type 20. Legend: TR=thread, PRI=priority, ROT=rotation string, STR=string. Independent variables shown in *italics and blue text*. Best performance in **bold**.

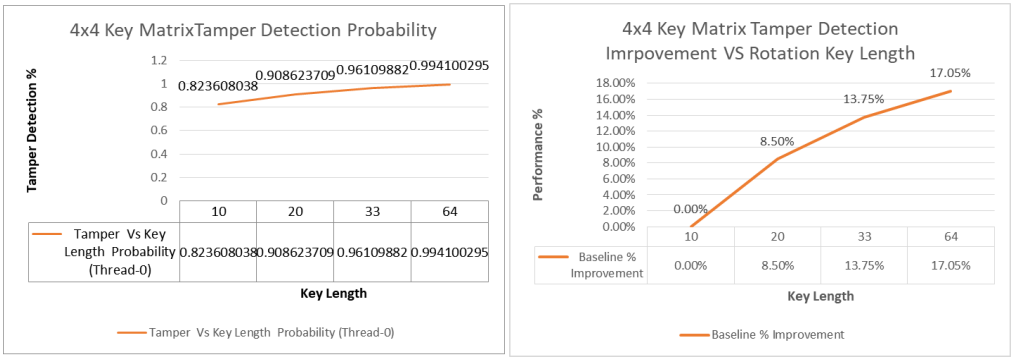
Figure 41 shows that the QPU configuration that yields the best performance is 8 QPUs over all other QPU configurations. The 4 and 16 QPU configurations outperformed the 1 QPU configuration as well. The performance improvements between 4, 8, and 16 QPU configurations were marginal and can be said to be flat as shown in Figure 41.

Figure 41. *QCKD Performance and Key Length Plots for 4x4 Rotation Matrix.*



Analysis of the probability of QCKD key tampering showed that the longer the length of the rotating string for both the source (Alice) and recipient (Bob), the more difficult it was for the eavesdropper (Eve) to tamper the cryptographic key without detection (Peres, 2002). The analysis showed that for a 10-character rotation string, the expected probability of detecting tampering was 82%. For the 20-character rotation string, the probability of detecting tampering was 90%. For a 33-character rotation string, the probability of detecting tampering was 96%, and for a 64-character key the probability of detecting tampering was 99%. In summary, increasing the length of the rotational string from 10 characters to 64 increased the tamper detection by 17.05% as shown in Figure 42.

Figure 42. *QCKD Tamper Detection vs Key Length Plots for 4x4 Matrix.*



QCKD 64x64 Data Analysis

Table 25 shows the raw performance data collected for the QCKD 64x64 input matrix test case.

Table 25. *QCKD Performance Data Collection for 64x64 Matrix.*

QCKD INDEPENDENT VARIABLES						Dataset: 20250712.001			
NOISE TYPE	*DATA SIZE*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	*ROT STR*	QC PERFORMANCE TIME (SEC) *1 QPU BASE*	*4 QPU*	*8 QPU*	*16 QPU*
20	64x64	Enable	RT	OS 1	10	102.2696667	**38.25148654**	38.9511559	39.50426555
20	64x64	Enable	RT	OS 1	20	205.5797369	**72.59351134**	73.97575045	75.41522717
20	64x64	Enable	RT	OS 1	33	379.0755424	**131.7423995**	138.3218246	138.4218345
20	64x64	Enable	RT	OS 1	64	684.9932032	**242.4298103**	244.3881373	246.1021891

Note. Noise type 20. Independent variables shown in *italics and blue text*. Best performance in **bold**.

Table 26 shows the performance improvement percentage for 4, 8, and 16 QPUs over the 1 QPU baseline configuration. Although the performance improvement from baseline was over 61%, the scalability performance gains were marginal between 4, 8 and 16 QPUs. Tamper detection improved as the length of the rotational string used to generate the cryptographic key increased; thus, yielding a hardened cryptographic stance. KGR dropped as expected with longer processing times required for longer rotational strings.

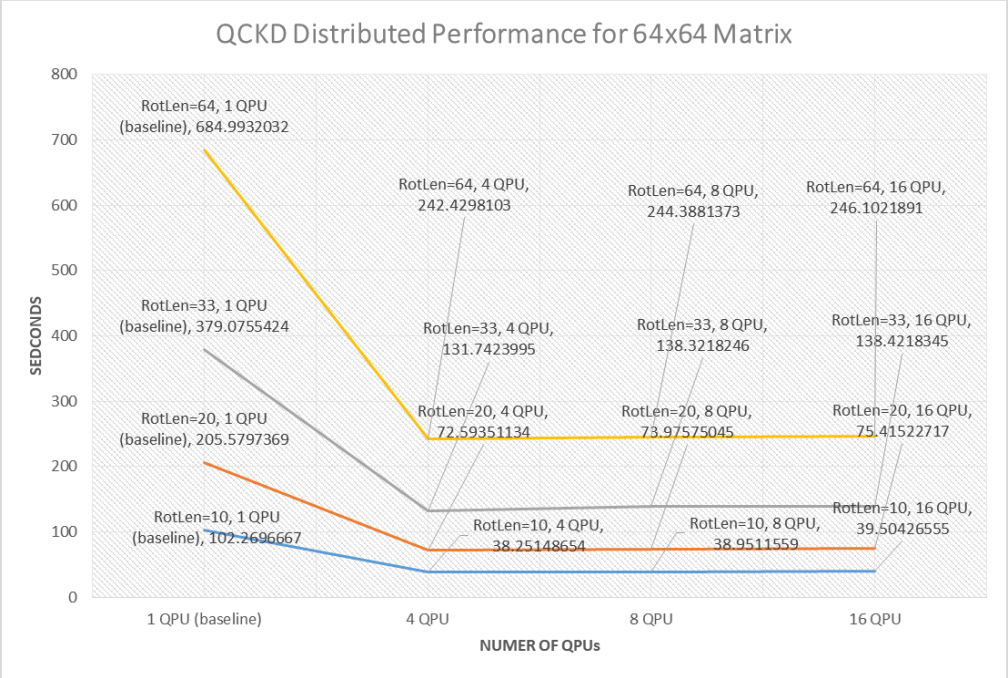
Table 26. *QCKD Tamper Detection and Performance Improvement for 64x64 Matrix.*

QCKD DEPENDENT VARIABLES								Dataset: 20250306.001		
DATA SIZE	**RAM NUMA**	**TR PRI**	**LOAD BALANCE**	**ROT STR**	*TAMPER DETECT %*	**1 QPU BASE** *KGR (KEY/SEC)*	**4 QPU** *KGR (KEY/SEC)*	*PERFORMANCE IMPROVE % FROM BASELINE* **4QPU**	**8QPU**	**16QPU**
64x64	Enable	RT	OS 1	10	0.827555339	0.00%	160.2039054	**63%**	64%	63%
64x64	Enable	RT	OS 1	20	0.907552083	8.00%	79.6965705	**65%**	64%	64%
64x64	Enable	RT	OS 1	33	0.961425781	13.39%	43.22093663	**65%**	65%	64%
64x64	Enable	RT	OS 1	64	**0.995117188**	16.76%	23.9184855	**64%**	65%	64%

Note. Noise type 20. Legend: TR=thread, PRI= priority, STR=string. Dependent variables shown in *italics and blue text*. Best performance in **bold**.

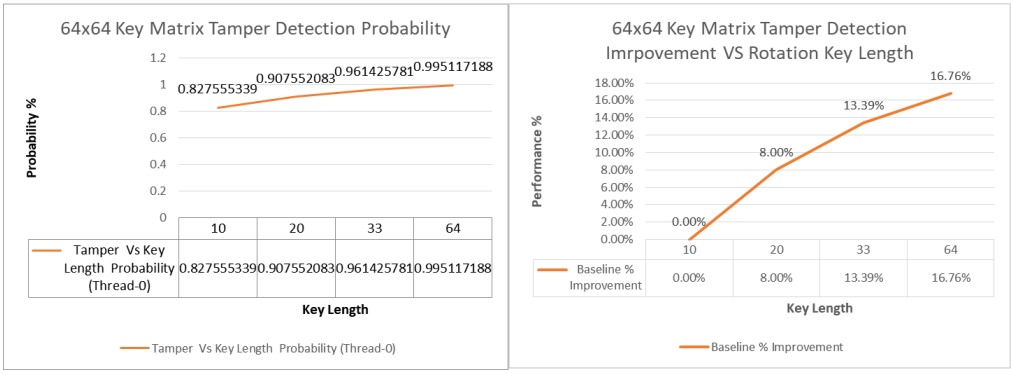
The QPU configurations that yield the best performance were 4 QPU or greater. For configurations above 4 QPUs, the performance gains were marginal as shown in Figure 43.

Figure 43. *QCKD Performance and Key Length Plots for 64x64 Matrix.*



As expected, with larger rotation string lengths, the hardness of the QCKD algorithm increased (Rasmusson & Barney, 2025). Increasing the length of the rotational string from 10 characters to 64 increased tamper detection by 16.76% as shown in Figure 44.

Figure 44. *QCKD Tamper Detection vs Key Length Plots for 64x64 Matrix.*



QCKD 128x128 Data Analysis

Table 27 shows the raw performance data collected for the QCKD 128x128 input matrix test case. The configuration that showed the best performance was 4 QPUs.

Table 27. *QCKD Performance Data Collection for 128x128 Matrix.*

QCKD INDEPENDENT VARIABLES						Dataset: 20250712.001			
NOISE TYPE	*DATA SIZE*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	*ROT STR*	QC PERFORMANCE TIME (SEC) *1 QPU BASE*	*4 QPU*	*8 QPU*	*16 QPU*
20	128x128	Enable	RT	OS 1	10	437.1740775	**153.4585097**	156.8971603	160.0970409
20	128x128	Enable	RT	OS 1	20	833.4944549	**292.6843157**	298.4534085	307.7934244
20	128x128	Enable	RT	OS 1	33	1520.419331	**539.28302**	551.1052172	562.2111058
20	128x128	Enable	RT	OS 1	64	2794.069857	**970.5099766**	982.6016026	997.7079868

Note. Noise type 20. Legend: ROT=rotational, STR=string. Independent variables shown in *italics and purple text*. Best performance in **bold**.

Table 28 shows the performance improvement percentage for the 4, 8, and 16 QPUs over the 1 QPU baseline configuration. Although the performance improvement from baseline was over 63%, the scalability performance gains were marginal between 4, 8 and 16 QPUs. Tamper detection improved as the length of the rotational string used to generate the cryptographic key increased; thus, yielding a hardened cryptographic stance. KGR dropped as expected with longer processing times required for longer rotational strings.

Table 28. *QCKD Tamper Detection and Performance Improvement for 128x128 Matrix.*

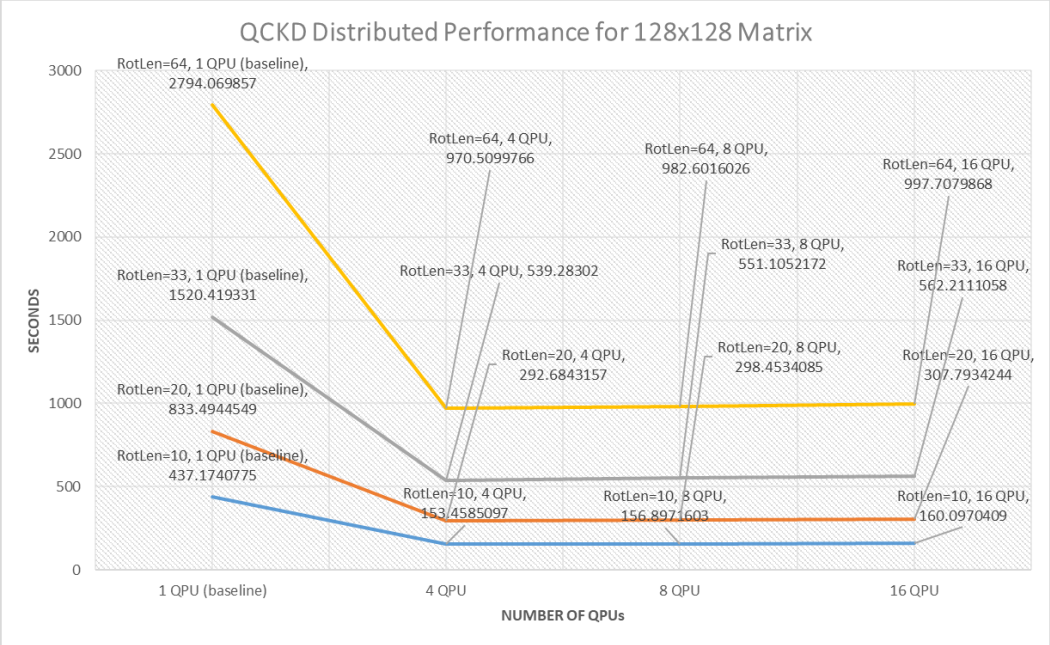
QCKD DEPENDENT VARIABLES								Dataset: 20250306.001		
DATA SIZE	RAM NUMA	TR PRI	LOAD BALANCE	ROT STR	*TAMPER DETECT %*	1 QPU BASE *KGR (KEY/SEC)*	4 QPU *KGR (KEY/SEC)*	*PERFORMANCE IMPROVE % FROM BASELINE* 4QPU	8QPU	16QPU
128x128	Enable	RT	OS 1	10	0.824015299	0	37.47706198	**%65**	64%	63%

128x128	Enable	RT	OS 1	20	0.910847982	8.68%	19.6569994	**65%**	64%	63%
128x128	Enable	RT	OS 1	33	0.958760579	13.47%	10.77597454	**65%**	64%	63%
128x128	Enable	RT	OS 1	64	0.994608561	17.06%	5.863847662	**65%**	65%	64%

Note. Noise type 20. Legend: TR=thread, PRI=priority, ROT=rotational, STR=string. Independent variables shown in *italics and blue text*. Best performance in **bold**.

The QPU configuration that yielded the best performance was 4 QPUs over all other QPU configurations. The 8 and 16 QPU configurations outperformed the 1 QPU configuration as well. The performance improvement above 4 QPU were marginal as shown in Figure 45.

Figure 45. *QCKD Performance and Key Length Plots for 128x128 Matrix.*



As expected with a larger rotation string length, the hardness increased. Increasing the length of the rotational string from 10 characters to 64 increased the tamper detection by 17.06% as shown in Figure 46.

Figure 46. *QCKD Tamper Detection vs Key Length for 128x128 Matrix.*

QCKD 256x256 Data Analysis

Table 29 shows the raw performance data collected for the QCKD 256x256 input matrix test case. The configuration that showed the best performance was the 4 QPU.

Table 29. *QCKD Performance Data Collection for 256x256 Matrix.*

QCKD INDEPENDENT VARIABLES						Dataset: 20250712.001			
NOISE TYPE	*DATA SIZE*	*RAM NUMA*	*THREAD PRIORITY*	*LOAD BALANCE*	*ROT STR*	QC PERFORMANCE TIME (SEC) *1 QPU BASE*	*4 QPU*	*8 QPU*	*16 QPU*
20	256x256	Enable	RT	OS 1	10	1752.014926	**611.1287956**	636.5551045	633.1269662
20	256x256	Enable	RT	OS 1	20	3346.220711	**1183.599152**	1231.128115	1228.240801
20	256x256	Enable	RT	OS 1	33	6096.968231	**2133.76613**	2154.006823	2200.039614
20	256x256	Enable	RT	OS 1	64	10907.82946	**3829.281751**	3880.991756	3972.441424

Note. Noise type 20. Legend: ROT=rotational, STR=string. Independent variables shown in *italics and purple text*. Best performance in **bold**.

Table 30 shows the performance improvement for 4, 8, and 16 QPUs over the 1 QPU baseline configuration. Although the performance improvement from baseline was over 63%, the scalability performance gains were marginal between 4, 8 and 16 QPUs. Tamper detection improved as the length of the rotational string used to generate the cryptographic key increased;

thus, yielding a hardened cryptographic stance. KGR dropped as expected with longer processing times required for longer rotational strings.

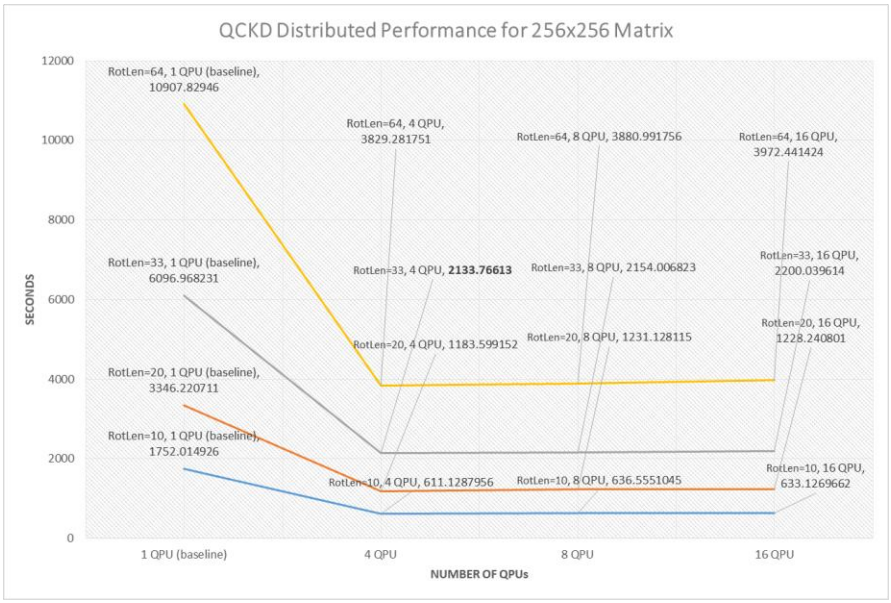
Table 30. *QCKD Tamper Detection and Performance Improvement for 256x256 Matrix.*

QCKD DEPENDENT VARIABLES								Dataset: 20250306.001		
DATA SIZE	**RAM NUMA**	**TR PRI**	**LOAD BALANCE**	**ROT STR**	***TAMPER DETECT %***	**1 QPU BASE *KGR (KEY/SEC)***	**4 QPU *KGR (KEY/SEC)***	***PERFORMANCE IMPROVE % FROM BASELINE*** **4QPU**	**8QPU**	**16QPU**
256x256	Enable	RT	OS 1	10	0.824539185	0	37.40607402	**65%**	64%	64%
256x256	Enable	RT	OS 1	20	0.908126831	8.36%	19.5850799	**65%**	63%	63%
256x256	Enable	RT	OS 1	33	0.959955851	13.54%	10.74894891	**65%**	65%	64%
256x256	Enable	RT	OS 1	64	0.994684855	17.01%	6.008161409	**65%**	64%	64%

Note. NT Noise Type. Noise type 20. Legend: ROT=rotational, STR=string. Independent variables shown in *italics and blue text*. Best performance in **bold**.

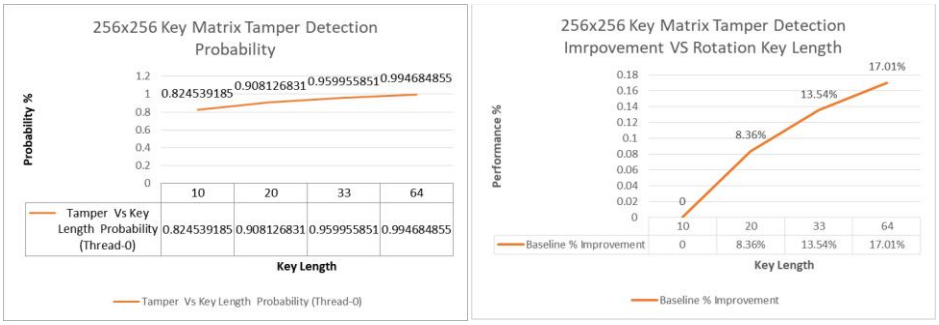
The QPU configuration that yields the best performance is 4 QPUs over all other QPU configurations. The 8 and 16 QPU configurations outperformed the 1 QPU configuration as well. The performance improvement above 4 QPU was marginal as shown in Figure 47.

Figure 47. *QCKD Performance and Key Length Plots for 256x256 Matrix.*



As expected with a larger rotation string length, the hardness of the QCKD algorithm increased. Increasing the length of the rotational string from 10 characters to 64 increased the tamper detection by 17.01% as shown in Figure 48.

Figure 48. *QCKD Tamper Detection vs Key Length Plots for 256z256 Matrix.*



QILQ Data Analysis

The following analysis covers data collection for the Interlin-q 2 QPU CNOT algorithm (Parekh et al., 2021). The purpose of this QC test case is to provide comparisons for the QPF instrument against the Interlin-q framework to evaluate the performance of both frameworks. The CNOT algorithm was used to keep the comparison between both frameworks as simple and portable. The baseline Interlin-q performance timing for 14,428 samples is the sum of the 2 QPU (baseline) column, which yields 12,063 seconds as shown in Table 31.

Table 31. *QILQ Performance Data Collection.*

QILQ DEPENDENT VARIABLES					**Dataset: 20250731.001**	
NOISE TYPE	***OUT CNOT VALUES***	***RAM NUMA***	***THREAD PRIORITY***	***LOAD BALANCE***	**SAMPLES**	***2 QPU (BASELINE) [SECS]***
20	2	Enabled 1	Real-time 2	OS 1	600	301.3747199
20	2	Enabled 1	Real-time 2	OS 1	600	515.4555676
20	2	Enabled 1	Real-time 2	OS 1	600	608.0521433
20	2	Enabled 1	Real-time 2	OS 1	600	632.0687652
20	2	Enabled 1	Real-time 2	OS 1	600	610.5633926
20	2	Enabled 1	Real-time 2	OS 1	600	398.7706721
20	2	Enabled 1	Real-time 2	OS 1	600	467.3361366
20	2	Enabled 1	Real-time 2	OS 1	600	687.3919437
20	2	Enabled 1	Real-time 2	OS 1	600	560.0409029
20	2	Enabled 1	Real-time 2	OS 1	600	528.7580044
20	2	Enabled 1	Real-time 2	OS 1	600	609.3892341
20	2	Enabled 1	Real-time 2	OS 1	600	309.7824161
20	2	Enabled 1	Real-time 2	OS 1	600	610.5456665
20	2	Enabled 1	Real-time 2	OS 1	600	405.9955735
20	2	Enabled 1	Real-time 2	OS 1	600	521.3954692
20	2	Enabled 1	Real-time 2	OS 1	600	516.6510108
20	2	Enabled 1	Real-time 2	OS 1	600	538.3430557
20	2	Enabled 1	Real-time 2	OS 1	600	365.1049728
20	2	Enabled 1	Real-time 2	OS 1	600	684.3068268
20	2	Enabled 1	Real-time 2	OS 1	600	389.1703007
20	2	Enabled 1	Real-time 2	OS 1	600	277.3111517
20	2	Enabled 1	Real-time 2	OS 1	600	326.6088393
20	2	Enabled 1	Real-time 2	OS 1	600	567.265379
20	2	Enabled 1	Real-time 2	OS 1	600	591.2084939
20	2	Enabled 1	Real-time 2	OS 1	28	40.42346501

Note. Independent variables shown in *italics and purple text*. The sum of the 2 QPU baseline column yields 12,063 seconds.

The equivalent 2 QPU performance for the QPF instrument was 52% faster with 6,029 seconds less than the Interln-q framework. QPF performance in addition to the 2 QPU configuration included 4, 8 and 16 QPUs as shown in Table 32. The assessment of performance between the Interlin-q and QPF 2 QPU configuration showed that the QPF instrument was at

least 50% faster than Interlin-q. The QPF configurations for 4, 8 and 16 QPUs also show performance improvements over the 2 QPU configuration. A factor for the QPF performance improvement is that the framework used data partitioning and CPU core allocation proximity to memory (Bai et al. 2025; Han & Wang, 2024). All QPF configurations show performance improvements over the 2 QPU Interlin-q. In summary, the configuration that showed the best performance of 76% improvement was for RAM NUMA enabled, real-time RT 2, and QC CPU affinity allocated to core 2.

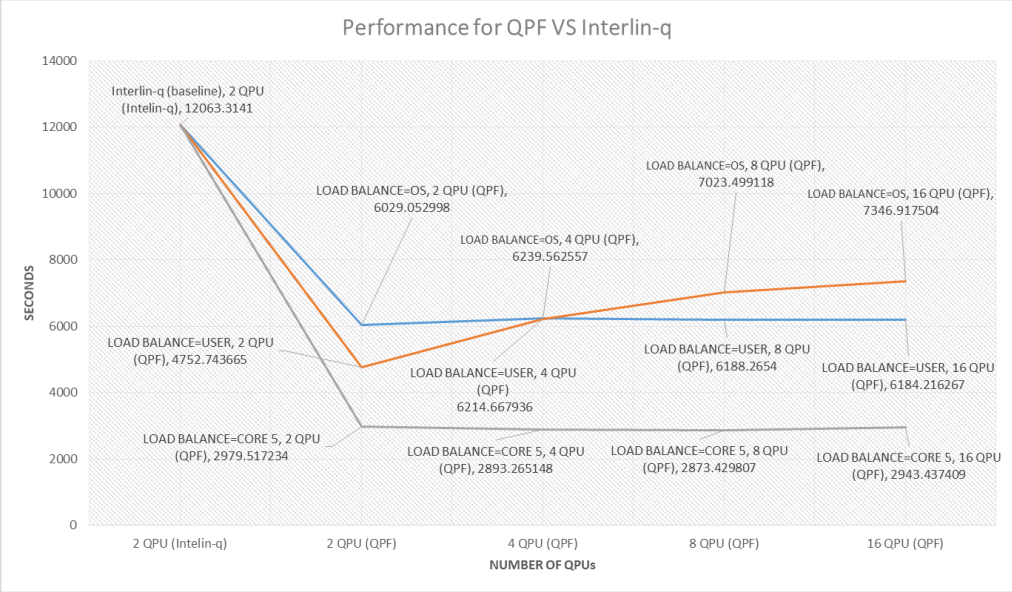
Table 32. *QILQ Dependent Variables for Data Collection.*

QILQ DEPENDENT VARIABLES								Dataset: 20250306.001			
CNOT	RAM	TH	LOAD	*QC PERFORMANCE TIME (SEC)*				*QPF PERFORMANCE IMPROVE % FROM Interlin-q BASELINE*			
GATE	NUMA	PRI	BALANCE	2 QPU	4 QPU	8 QPU	16 QPU	2QPU	4QPU	8QPU	16QPU
2	Enable	RT 2	OS 1	6029.052998	6239.562557	6188.265352	6184.216267	50%	48%	49%	49%
2	Enable	RT 2	OS 2	4752.743665	6214.667936	7023.499118	7346.917504	60%	48%	42%	39%
2	**Enable**	**RT 2**	**CORE 2**	2979.517234	2893.265148	2873.429807	2943.437409	**75%**	**76%**	**76%**	**76%**

Note. QPF Baseline is the 2 QPU performance when compared to the 2 QPU Interlin-q baseline configuration shows the baseline performance. NT Noise Type 20. Legend: TH=thread, PRI=priority. Dependent variables shown in *italics and blue text*. Best performance in **bold**.

The QPU configurations that yielded the best performance were 4 QPU or greater. Above 4 QPUs, the performance gains became marginal as shown in Figure 49. Setting a thread CPU affinity fixed to a particular CPU (core 2), benefited performance as the QILQ QC capitalized on L1-3 caching (Bai et al. 2025; Han & Wang, 2024). In summary, the QPF performance improvement gains were due to data partitioning and CPU core allocation proximity to memory banks (Bai et al. 2025; Han & Wang, 2024).

Figure 49. *QILQ Performance Plots.*



Note. Interlin-q 2 QPU QC is the baseline for this test case experiment.

Finally, increasing to 4 QPU or greater offered improved performance over the 1 QPU baseline. Figure 49 shows that scaling to more QPUs may not yield scalable performance increases.

Research Question 1 / Hypothesis

Distributed quantum framework feasibility and reliability: How can a quantum distributed framework be designed and developed to maintain system reliability and stability when executing distributed parallel QC algorithms over extended operational periods?

H1₀

Null hypothesis for distributed quantum framework shows no feasibility or reliability: Running multiple distributed parallel QC algorithms causes the quantum framework to crash at least once and not collect all samples in a single run.

$H1_a$

Distributed quantum framework feasibility and reliability: Running multiple distributed parallel QC algorithms does not affect the quantum framework reliability and ability to collect all samples in a single run.

The alternate **Hypothesis $H1_a$ – Distributed quantum framework feasibility and reliability** directly supports RQ1. The QPF instrument demonstrated robustness and the ability to run without errors during the data collection for all QC test cases including QHED, QCNN, QCCD, QCKD, and QILQ. The experiment tested and rejected the **$H1_0$ - Null hypothesis for distributed quantum framework not feasible or not reliable** by demonstrating the operation of the QPF instrument without crashes for 14,428 (sample) shots times 30 pixels, giving 432,840 samples during the instrument reliability and validity pilot experiment. In addition, a conservative estimate for the five QC test algorithms reconciled as 17 datasets times 14,428 samples per set, giving 245,275 samples without crashing or losing communications with the QPU. Note that this is a conservative estimation as most QC test cases had more than 14,428 samples per Table 14. The reliability hypothesis of the QPF was verified with a t-test for $\alpha = 5\%$. Table 22 shows the reliability of the QPF t-test p value of 0.013875, which was less than the 5% required to reject the $H1_0$ null hypothesis. Therefore, the demonstration of the QPF instrument rejects $H1_0$ and supports the alternate Hypothesis $H1_a$.

Research Question 2 / Hypothesis

Distributed quantum framework performance: What is the impact on performance response time gains by parallelizing a distributed QC algorithm in a simulation framework when compared to a single QC algorithm instance?

$H2_0$

Null hypothesis for distributed quantum framework negligible performance improvement: The performance response time from multiple distributed parallel QC algorithm instances working on a dataset shows no performance improvement when compared to the response times of a single QC algorithm instance working on the same dataset.

$H2_a$

Distributed quantum framework performance: The performance response time from multiple distributed parallel QC algorithm instances working on a dataset shows performance improvement when compared to the response times of a single QC algorithm instance working on the same dataset.

The alternate **Hypothesis $H2_a$ – Distributed framework performance** addresses RQ2. The test included all QC test cases for QHED, QCNN, QCCD, QCKD, and QILQ. The experiment tested and rejected the **$H2_0$ - Null hypothesis for distributed quantum framework negligible performance improvement** by demonstrating distributed parallel execution of all QC test case algorithms and showing performance improvements for all test cases. Table 33 shows the highest and lowest distributed performance improvements.

Table 33. *Highest and Lowest Distributed Performance for All QC Test Cases.*

ALGORITHM TEST CASE	BEST PERFORMANCE IMPROVEMENT	LOWEST PERFORMANCE IMPROVEMENT
QHED	42.05% [32x32 image, 4 QPU]	8.36% [1024x1024 image, 8 QPU]
QCNN	35.77% [256x512 image, 16 QPU]	33.80% [4x8 image, 8 QPU]
QCCD	91.55% [32x32 image, 8 QPU]	79.83% [4x8 image, 4 QPU]
QCKD	65.27% [256x256 matrix, 4 QPU, RotStr=33 chrs]	61.37% [64x64 matrix, 16 QPU, RotStr=33 chrs]
QILQ	76.18% [Affinity: CORE 8, 8-GPU]	39.10% [Affinity: OS=2 User 8-GPU]

Note. All performance metrics show improvement over the 1 QPU baseline cases.

The QPF instrument consistently demonstrated performance improvement across all QC test cases. The t-test compared two quantum results datasets comprised of at least 14,428 samples. The datasets used were a reference 1 QPU performance dataset and a test dataset with multi QPU performance. The performance improvements of the distributed process hypothesis were verified with a t-test for $\alpha = 5\%$. Table 17 shows the performance improvement when distributing processing to more than 1 QPU with a t-test p value of 5.8744E-106, which was less than the 5% required to reject the $H2_0$ null hypothesis. Therefore, the performance exhibited by quantum algorithms rejects $H2_0$ and supports the alternate Hypothesis $H2_a$.

Research Question 3 / Hypothesis

Distributed quantum framework results show quantum noise robustness: How does the integration of quantum noise models affect the robustness and accuracy of distributed parallel QC algorithms within the proposed quantum simulation framework?

$H3_0$

The null hypothesis for distributed quantum framework results shows no quantum noise modeling robustness: The distributed QC algorithm models under

quantum noise conditions will generate results that are not degraded when compared to the quantum results without noise.

$H3_a$

Distributed quantum framework results show quantum noise modeling robustness: The distributed QC algorithm under quantum noise conditions will generate results that are degraded when compared to the quantum results without noise.

The alternate **Hypothesis $H3_a$ – Distributed quantum framework results show quantum noise modeling robustness** directly addresses RQ3. The robustness of an algorithm noise model in this context is referred to as the inclusion of quantum noise to be more in family with actual results produced by QPU hardware. The robustness of the results included IBM profiles for QPU hardware. The analysis showed that the quantum noise model can be executed in a distributed fashion providing an average of 2% readout noise on the Qubits for QHED, and QCCD. For QCNN, the noise models introduced an error in the probability of correct image identification and produced a drop from 95% to 83% correct image identification. For the QCKD, the algorithms were impervious to quantum noise since the generation of the quantum rotation string depends on random Qubit draws, which noise causes by design. The QILQ test case was not considered in the evaluation of quantum noise as its objective was to establish a reference for comparing performance of the QFP instrument to the Interlin-q framework. The expectation was that QILQ CNOT results were affected by quantum noise; however, it was not used as the scope of the QILC test was performance and not results quality.

The experiment tested and rejected the $H3_0$ - Null hypothesis for quantum distributed framework results show no robustness with quantum noise modeling. Quantum noise showed

negative impact on QC results. The t-test compared two quantum results datasets comprised of at least 14,428 samples. The datasets were a reference dataset without quantum noise and a test dataset with quantum noise included. The effects of the quantum noise model hypothesis were verified with a t-test for $\alpha = 5\%$. Table 21 shows the impact of quantum noise on the results with a t-test p value of 0.013875, which was less than the 5% required to reject the $H3_0$ null hypothesis. Therefore, quantum noise adds robustness by providing more realistic results, which rejects $H3_0$ and supports the alternate Hypothesis $H3_a$.

Evaluation of the Findings

The following section evaluates the research data analysis, and provides a foundation for answering the research questions. A general finding on all algorithms test cases was that increasing the size of the input data scaled the performance of the QC test cases accordingly. Increasing the number of QPU from the baseline scaled as expected and increased performance of all the five QC test cases. An observation for QHED and QCNN is that the performance improvement peaked at 4 QPUs and degraded as the number of QPUs increased; however, the degraded performance still outperformed the 1 QPU baseline. For QCCD, the performance improvement peaked at 8 QPUs and levelled off to marginal gains with 16 QPUs. For QCKD, the performance improvement peaked at 4 QPUs and levelled off with marginal gains for 8 and 16 QPUs.

Regarding distributed quantum framework feasibility and reliability (RQ1), the QPF instrument operated in a closed loop configuration reliably without errors and demonstrated the maturity of current software, hardware, and quantum technology. The pilot experiment demonstrated the QPF instrument viability, reliability, and validity for collecting the experiment data. In addition, the data collection demonstrated the ability of the QPF instrument and the

quantum interface to reliably connect and coordinate execution of QC algorithm models. There was a total of at least 245,275 collected samples without the QPF instrument crashing for all five QC algorithms, which is more than the statistically significant minimum of 14,428 to achieve 1 in 20 probability results occurring by chance (Hazra & Gogtay, 2016).

Concerning distributed quantum framework performance (RQ2), the findings demonstrate that a 4 QPU configuration exhibits the best performance with 8 and 16 QPUs also outperforming the 1 QPU baseline. The performance of distributed processing demonstrated improvement over the 1 QPU baseline for all five QC test case algorithms. The best performance improvements within each type of QC test cases included QHED with 42.05%, QCNN with 35.77%, and QCCD with 91.55%, and QCKD with 65.27%, and QILQ with 76.18% as shown in Table 33.

Performance improvements were aligned with expected theory based on data decoupling and parallelization of smaller partitioned datasets (Bai et al., 2025; Han et al., 2024). An observation is that independent variables were optimized to obtain the best dependent variable performance for the QC algorithm tested. For example, the QCNN and QILQ algorithms benefited from proximity to cached data and allocation to a fixed core affinity for maximizing data locality instead of OS load balancing (Bai et al., 2025). On the other hand, QCs like QHED, QCCD and QCKD benefited from data decoupling and the OS load balancing for best core available (Bai et al., 2025; Han et al., 2024).

With respect to distributed quantum framework results showing quantum noise robustness (RQ3), this research experiment addressed the collection of quantum results and modeled the effects of quantum noise. Therefore, providing the capacity to model expected results to be in family with results generated by QPU hardware. The analysis showed that

quantum readout noise introduced an average 2%-pixel error on Qubits for QHED, and QCCD during image encoding (Dolciami, 2022; Mastriani, 2020). For QCNN, the noise effects also introduced a drop from 95% to 83% in correct classification as expected since the test image is affected by error in image encoding (Dolciami, 2022; Mastriani, 2020). For the QCKD, the algorithms were impervious to quantum noise since the generation of the quantum rotation string depended on random Qubit draws. Executing distributed QC algorithms with a noise model using QPU hardware profiles provided this research the ability to produce in family QPU expected results and reduce integration risks (Kop, 2024). In summary, quantum noise is the primary reason current QPU hardware cannot achieve full quantum rates since many shots are required to obtain statistically significant results (Iyengar et al., 2020; Melvin, 2022).

Limitations

A limitation is that modeling noise is based on the fidelity of the model provided by IBM, and thus like any model, the results are a prediction of the expected results with a degree of uncertainty or error.

A second limitation on the results for the multi-QPU configuration shows that there is a bottleneck provided by the computer architecture including PCI arbitration of multiple threads attempting to access the disk drive, or memory or both. Further investigation is recommended to isolate why increasing the number of QPU beyond four yields worse performance for most QC test cases except the QILQ and QCCD algorithms.

A limitation on some algorithms is that enabling quantum error modeling does add considerable processing time during simulation. For example, long processing times were observed on algorithms like the QCCD, which apply the ADC quantum model on a per pixel

basis. During initial quantum algorithm modeling, development of the initial model may be expedited by not including noise until the model matures and is integrated with the QPF.

Summary

During the initial research phases, an exploratory pilot evaluation on the reliability and validity of the QPF instrument was completed to ensure robustness of the data collection. Analysis of all quantum algorithms was completed for the multi-QPU distributed configuration showing performance improvement over the 1 QPU baseline for all QC test cases. This demonstrated the scalability of the QPF instrument. Euclidian distance and t-tests were calculated for all imagery data including QHED and QCCD to evaluate the impact of quantum noise on divergence of reference and quantum generated data. The t-tests were used to support the rejection of the null hypotheses and acceptance of the alternate hypotheses. Accuracy of classifier prediction was calculated for QCNN and shown to be impacted by quantum noise. Evaluation of cryptographic results was performed including metrics for KGR, and probability of cryptographic key tamper interdiction. However, unlike the other QC test cases, QCKD appeared to be unaffected by quantum noise due to the dependence of random Qubit draws during rotational string parsing. Limitations were considered including long processing times, and fidelity of the quantum error models downloaded from IBM QPU hardware profiles. Correlations and inferences were developed based on experiment results to answer research questions from tests on the hypotheses. A brief section on statistical analysis data distribution provided some insight on the impact of quantum noise on four of the QC test cases. For example, the QHED test case showed that around 2% of pixels were affected by quantum noise. For the QCNN test case, statistical analysis showed the impact on the performance dropped classification accuracy from 90% to 88%. For the QCCD testcase, the analysis showed that the larger the pixel value the

larger probability of the impact of quantum noise as more Qubits are required to encode the value of the pixel. Finally, for the QCKD algorithm, which relied on random Qubit flips due to quantum noise, the insight was the QCKD appears to be impervious to the effects of quantum noise. Also, for the QCKD OS scheduling of the algorithm appears to bias QPU cores 3, 9, and 13 with the QCKD algorithm on core 13 outperforming all other QCKD algorithms with larger KGR.

This chapter successfully provided empirical experiments, results, and analysis to reveal numerical patterns. The analysis provided correlations between independent and dependent variables to make assertions on hypotheses tests for answering the research questions.

Chapter 5: Implications, Recommendations, and Conclusions

This chapter synthesized a quantitative constructive experiment demonstrating that the proposed distributed QPF instrument is feasible, reliable, and capable of delivering meaningful performance improvements across multiple QC algorithms. The findings validated that distributing QC workloads across several QPUs reduces execution time, maintains system stability, and provides realistic quantum-noise behavior. The chapter also offers practical recommendations for applying the framework in real-world applications and proposes future research directions, including hardware-based validation, cross-vendor comparisons, scalability analysis, and GPU-accelerated simulation.

This quantitative, constructive experimental research provided several implications and recommendations for distributing QC algorithms. This work provided a solution that expedites processing to execute complex QCs that require more Qubits than available on a single QPU. The research experimentally tested the relationship between the independent variables for data size partitioning, NUMA memory strategy, thread CPU core affinity & priority, and load balancing. Dependent variables included algorithm performance, Euclidian distance, probability of correct image classification, KGR, and probability of cryptographic tamper detection. A deeper understanding was gleaned from the independent variables as they drove the performance of the distributed QC algorithms. In addition, this research developed the necessary artifacts and instruments to collect and analyze the experiment's results.

The problem addressed was that a single QPU cannot solve distributed quantum computing applications that require more Qubits than available on a QPU, which limits the scalability and fidelity of quantum solutions (Davis et al., 2024; van Dijk et al., 2019; Davis et al., 2023). According to Ichikawa et al. (2023), in 2022 condensed physics applications were

using more than 100 Qubits, which are approaching Amazon 105 Qubit QPU, and IBM 127 Qubit QPU (AbuGhanem, 2025). Additionally, it is known that the pervasiveness of noise effects is more pronounced as the complexity and depth of QC increases. This makes the case to partition and distribute QCs into smaller shallow algorithms spread across several QPUs (Kan et al., 2024). Therefore, algorithm decomposition into smaller components and data decoupling directly benefits distributed solutions for large scale QC algorithms (Kan et al., 2024).

The purpose of this positivistic worldview constructive research was to build and validate a novel distributed parallel framework for quantum computing. More specifically, this research developed the instrument to experimentally evaluate performance of distributed parallel QC algorithms under the impact of quantum noise modeling. This study also aimed at modeling distributed QC algorithms to expedite maturation through experimentation. The research found which combination of independent variables produced the best distributed performance and its effects on results. The independent variables traded by the study included number of QPUs, Non-Uniform Memory Allocation (NUMA) strategy, CPU core affinity & priority, and OS thread scheduling. The dependent variables analyzed for numerical patterns and trends included algorithm performance response time, Euclidian distance, probability of classification, cryptographic Key Generation Rate (KGR), and probability of tamper detection.

The selection of test case QC algorithms covered several salient computer science domains to provide a wider breath of data collection and analysis of the effects of algorithm distribution over several QPUs. The wider selection of QC algorithms allowed for a deeper understanding of the effects of QC distribution. An additional focus of the research was to demonstrate the feasibility of the QPF for distributed processing applications and framework

performance compared to a reference framework (Balamurugan et al., 2024; Shafique et al., 2024).

Implications

This dissertation benefits QPU applications that address fault tolerance solutions to remove the barrier that prohibits large scaling at quantum speeds for Shor's, Glover's, Deutsch-Jozsa's and other algorithms (Shafique et al., 2024). More specifically, the implications and recommendations provided concise practical knowledge of distributed quantum processing in various domains (Ichikawa et al., 2023). In addition, this research provided a QPF framework instrument, which can be used to integrate industrial applications for closed loop systems (Escudier, 2019). Even after QPUs reach Qubit counts that allow processing large QC algorithms, this research will be relevant to model distributed processing including applications in emergent photonic networks, which rely on distributed processing (de Forges de Parny et al., 2023; Zhang et al., 2025; Davis et al., 2023).

Research Question 1 / Hypothesis

Distributed quantum framework feasibility and reliability: How can a quantum distributed framework be designed and developed to maintain system reliability and stability when executing distributed parallel QC algorithms over extended operational periods?

$H1_0$

Null hypothesis for distributed quantum framework shows no feasibility or reliability: Running multiple distributed parallel QC algorithms causes the quantum framework to crash at least once and not collect all samples in a single run.

$H1_a$

Distributed quantum framework feasibility and reliability: Running multiple distributed parallel QC algorithms does not affect the quantum framework reliability and ability to collect all samples in a single run.

The alternate **Hypothesis $H1_a$ – Distributed quantum framework feasibility and reliability** directly supported RQ1 by answering that distributed quantum processing technology developed in this research is feasible and robust. The solution used by this research used well established frameworks like Qiskit and a hierarchical scheduler architecture that modularized and simplified distributed processing, process synchronization, and thread management.

The data statistical significance was designed for an $\alpha = 5\%$ cutoff required to reject the null hypothesis. More than 95% of performance results showed the ability of the QPF to operate without failures as well as able to collect all metrics for each frame; therefore, rejecting the null hypotheses and accepting the alternate hypothesis $H1_a$ – Distributed quantum framework feasibility and reliability.

Factors that may have affected the interpretation of results are that the QPF framework executed five distinct QC test case algorithms to collect experimental data. Each algorithm's results offered differences as some performed better under the 4 QPU configuration, and others like the QCCD benefited from scaling up to 16 QPUs. Some algorithms, like QCNN, benefited from staying persistent on a specific CPU core and not switching to another core during OS scheduling. The results of quality measures were also different and specific for each QC test case, which were not comingled or compared across test cases.

The QPF instrument findings demonstrated robustness and the ability to run without errors during the data collection for all the QC test algorithms. Therefore, the demonstration of the QPF instrument rejected $H1_0$ and supported the alternate Hypothesis $H1_a$. The experiment's findings demonstrated the feasibility of distributing and synchronizing QC across several QPUs. The experiment also demonstrated results assembly and synchronization for all QPUs (Jimenez-Navajas et al., 2024; Han et al., 2024; Lindsay et al., 2021; Solarte-Martínez et al., 2021).

Research Question 2 / Hypothesis

Distributed quantum framework performance: What is the impact on performance response time by parallelizing a distributed QC algorithm in a simulation framework when compared to a single QC algorithm instance?

$H2_0$

Null hypothesis for distributed quantum framework negligible performance improvement: The performance response time from multiple distributed parallel QC algorithm instances working on a dataset shows no performance improvement when compared to the response times of a single QC algorithm instance working on the same dataset.

$H2_a$

Distributed quantum framework performance: The performance response time from multiple distributed parallel QC algorithm instances working on a dataset shows performance improvement when compared to the response times of a single QC algorithm instance working on the same dataset.

The alternate **Hypothesis $H2_a$ – Distributed framework performance** addressed RQ2 by answering that the research findings demonstrated meaningful performance scaling as the QC algorithm was distributed across several QPUs. The performance tests included all QC test cases for QHED, QCNN, QCCD, QCKD, and QILQ. The experiment tested and rejected the **$H2_0$ - Null hypothesis for distributed quantum framework negligible performance improvement**. The experiment demonstrated distributed parallel execution for all QC test case algorithms and showed performance improvements for all test cases as expected (Kan et al., 2024; Bai et al. 2025; Han & Wang, 2024).

A factor for consideration is that a QPU is still dependent on classical CPU for process management and data storage. Factors that may have influenced results are scaling limitations of the QPU simulator. According to Iyengar et al. (2020) QPU's still relies on classical computing for mass storage and interfacing to other computing systems. This includes OS scheduling impact and data access to mass storage including RAM and disk latencies. Therefore, considerations for achieving the best QPU distribution performance needed to include factors like data size partitioning, NUMA memory strategy, QPU interface thread core affinity & priority, and thread OS load balancing (Kan et al., 2024; Bai et al. 2025; Han & Wang, 2024; Davis et al., 2023). Other factors that may also influence results are scaling limitations of the QPU simulator. For example, performance improvements peaked at 4 and 8 QPUs for some algorithms like QHED and QCCD, which may hint there is an internal limitation imposed by the QPU simulator. Scalability limitations may be caused by increased inter CPU core data transfers including PCIe arbitration during writes to disk (Dong & Peng, 2023). Factors in data locality impacted performance due to NUMA, where users drive locality of allocated memory instead of the OS interleaved CPU memory allocation. These factors could incur larger response times due

to inter GPU QPI data Bridge transfers if memory allocated is on another CPU memory banks (Bai et al. 2025). Thread CPU core affinity is important because it determines if a QPU interface thread will be moved to another core during an OS context switch, which results in the thread losing cache performance improvement benefits and proximity to a CPU interface to memory banks (Huang, 2023).

During the experiments, CPU cache played a role in performance for some algorithms like the QCNN, where test data in L1-3 cache had a considerable impact on QC algorithm response time. Data partition decoupling was also important to reduce thread synchronization interdependence and improve parallel performance (Dong et al., 2024; Davis et al., 2023). An example is the QCNN data persistence on L1-3 cache could only be capitalized if the current QPU interface thread affinity stayed on the same CPU core. Moving the QPU interface thread to another CPU core would lose all benefits of the L1-3 cache, since the data had to be fetched from main memory for every CPU core switch. A broader insight is that OS thread scheduling CPU core affinity switches occur thousands of times during the life of a thread, forcing a cache re-fetch from main memory compounding the costs of memory access (Basha et al., 2023). In addition to the processing costs of running the noise model, another factor to consider is that quantum noise also had an impact on performance as increasing the number of shots to obtain a statistical representative result negatively impacted response time (Iyengar et al., 2020).

Overall, the QPF instrument demonstrated distributed performance improvement across all QC test cases. Therefore, the performance exhibited by QC algorithms rejected $H2_0$ and supported the alternate Hypothesis $H2_a$. The best performance improvements over the 1 QPU baseline case within each type of QC test cases included QHED with 42.05%, QCNN with 35.77%, QCCD with 91.55%, QCKD with 65.27%, and QILQ with 76.18% as shown in Table

33. The reduced QPU interdependence provided performance improvements were as expected due to QC distribution, data placement locality, thread CPU core affinity & priority, and data decoupling (Kan et al., 2024; Bai et al. 2025; Han & Wang, 2024).

Research Question 3 / Hypothesis

Distributed quantum framework results show quantum noise robustness: How does the integration of quantum noise models affect the robustness and accuracy of distributed parallel QC algorithms within the proposed quantum simulation framework?

$H3_0$

The null hypothesis for distributed quantum framework results shows no quantum noise modeling robustness: The distributed QC algorithm models under quantum noise conditions will generate results that are not degraded when compared to the quantum results without noise.

$H3_a$

Distributed quantum framework results show quantum noise modeling robustness: The distributed QC algorithm under quantum noise conditions will generate results that are degraded when compared to the quantum results without noise.

The alternate **Hypothesis $H3_a$ – Distributed quantum framework results show quantum noise modeling robustness** directly addressed RQ3. The research question was answered with demonstrated concise quantum error modeling distributed across several QPUs. The quantum noise negative impact was observed and measured with dependent variables for all algorithms.

A factor to consider with the experimental results is that although quantum noise model profiles use empirical data collected by IBM cloud QPU hardware, the model offers an approximation of quantum noise phenomenology and has limitations in fidelity and performance (Qiskit Ecosystem, 2025). The noise models are statistical representations, which will vary results as the model is updated, or environmental conditions change around the dilution refrigerated QPU hardware. Quantum noise will manifest in image processing as pixel flips due to quantum errors incurred during image encoding and image processing (Dolciami 2022; Mastriani 2020). The effects of quantum noise are random; therefore, the image encoding errors will exhibit random qualities and may not be able to be reproduced exactly during sequential error collections. Further investigation is required to identify a mechanism for seeding the noise model random number generator in the QPU simulation so that a more deterministic noise performance can be evaluated and controlled for some experiments.

The findings show that the quantum noise model was executed in a distributed fashion and provided an average of 2% readout noise on the Qubits for QHED, and QCCD. For QCNN, the noise models introduced an error in image encoding, and the probability of correct image identification dropped from 95% to 83%. For the QCKD, the algorithms were impervious to quantum noise since the generation of the quantum rotation strings depended on random Qubit draws. The experiment tested and rejected the $H3_0$ - Null hypothesis for quantum distributed framework results show no robustness for quantum noise modeling. In summary, quantum noise added fidelity robustness to the QPU simulation by providing more realistic results, which rejected $H3_0$ and supported the alternate Hypothesis $H3_a$.

Recommendations for Practice

This section focuses on practical recommendations on how this research can be integrated with current systems to address real-world problems. Three areas of technology insertion are proposed, including integration of QPF with closed loop systems to address simulations in molecular medicine and high-fidelity physics (AbuGhanem, 2025). The recommendations also propose integration of distributed QCKD solutions to existing legacy cryptographic systems, and integration of distributed QPU processing to mitigate quantum noise using ML techniques as proposed by work from Melvin (2022).

Practical Recommendation 1

This research provided a solution to distribution, parallelization, and integration of QC algorithms for running complex QC that require more Qubits than available in a single QPU. The instruments included the ability of QPF to seamlessly interface with either QPU simulations and or interface with QPU cloud hardware. A note is that the QPF can concurrently interface with both QPU simulations and QPU hardware. This study was also designed to help build expertise in quantum computing integration before self-error correcting QPU arrival by 2030 (Vasconcelos 2020). A recommendation is to publish the findings and foster new research in QPU QC distribution. The QPF instrument developed for this research is a full working cross platform solution for Linux and MS Windows and can be integrated with applications for image processing, cryptographic solutions, and overall general quantum modeling for a variety of problems (AbuGhanem, 2025). Parts of this research have been published including the QHED applications at the 2025 ISADS Symposium at the University of Arizona (Salcedo & Ahmed, 2025). The feedback from the academic community was positive with ideas for collaboration across academic institutions.

Practical Recommendation 2

A finding is that the QCKD algorithm provides resilience over quantum noise due to its design. A recommendation is to integrate QCKD to mitigate risks with RSA encryption. The proposed solution would replace RSA with an enhanced distributed QCKD solution reliant on the strength of physics and not raw computing power (Peres, 2002). Publication of the findings will provide the community with a distributed processing option for QCKD problems and build credibility to facilitate deployment.

Practical Recommendation 3

This research also demonstrated that for all QC test case algorithms tested, 4 QPUs or more provide performance improvements over a 1 QPU baseline. Therefore, a recommendation is to integrate the QPF with other research, such as mitigation of quantum noise using ML to expedite processing by Melvin (2022). This recommendation would help bring self-error correction at quantum speeds before 2030's arrival of error correcting QPUs (Vasconcelos; 2020).

Recommendations for Future Research

Future Research Recommendation 1

This research relied on QPU simulation to mitigate the excessive costs of IBM cloud QPU hardware and expedite software development. IBM QPU hardware places QC jobs in queue, and jobs can idle waiting seconds or tens of minutes, which adds considerable time and risk to software development (IBM Pricing, 2025). However, the QPF instrumented is not limited to QPU simulations only. The QPF instrument was designed to communicate seamlessly with either simulated QPU or QPU cloud hardware. A recommendation for future research is to perform evaluation of each of the five QC test algorithms on QPU hardware and compare results

obtained from this research. Future studies based on QPU hardware would further strengthen these research findings and increase confidence in the QPF as a modeling and integration instrument.

Future Research Recommendation 2

Additional research could be to interface to other vendors QPU solutions, such as DWARE, Amazon, and Righetti systems, and perform a survey of how the various quantum computing solutions compare to each other in terms of performance, cost, reliability, and portability (AbuGhanem, 2025). Further research could be done in other areas such as quantum physics molecular medicine to provide expedited results to complex QC that may not fit in a single QPU.

Future Research Recommendation 3

Some of the QC test cases peaked performance at 4 QPUs and 8 QPU with marginal performance gains for 16 QPUs. Therefore, follow up research can focus on the root causes of this scalability limitation. Issues can be PCIe bus arbitration, or resource contention by the QPU simulation (Dong & Peng, 2023).

Future Research Recommendation 4

Current QPU simulation offered by IBM has limitations that include up to 30 Qubits and rely on empirically collected QPU noise models. Integration with GPU technology would also include additional QPU noise modeling capabilities that may mitigate some of the limitations on the scalability observed. A survey study using a combination of CPU versus GPU simulations may shed light into CPU limitations. This research would further expand the scalability, capabilities, and limitations of IBM QPU simulations. The findings would provide the software

architect with more information on how and when to use simulation and specific limitations, which may not be relevant depending on the problem being solved.

Future Research Recommendation 5

This research relied on CPU simulation and made no use of GPU technology to simulate QPUs. Therefore, a recommendation is to integrate GPU processing with the QPF instrument to accelerate QPU simulation processing in a similar fashion how multi-GPUs distribute image processing (Eilemann et al., 2009; Chalumuri et al., 2022). The task would develop a follow-up study for integration of GPU technology and analysis of performance and quality of results. The GPU integrated capability could be used as a comparison to this research for each of the five QC test cases, adding a validation point of reference. This future research would increase confidence in the findings of this study and provide the QPF instrument with additional capabilities to use GPU technology to expedite processing.

Conclusions

This constructive quantitative research developed a QPF instrument capable of demonstrating through empirical experimentation a solution to the problem for distributed QC algorithms that require more Qubits than available on a QPU. The experiments demonstrated effective meaningful performance improvements through scalability and distributed processing for image processing, cryptography, convolutional neural networks, and CCD physics modeling (Davis et al., 2024; van Dijk et al., 2019).

The carbon dioxide (CO_2) footprint of current HPC projections, including GPUs, are projected to account for up to 8% of the worldwide emissions if no changes are made according to the US International Trade Commission (Yang et al., 2023). Although quantum computing does not provide better performance for all computing applications; the technology does offer

power efficiencies and performance improvements that surpass HPC and multi-GPU solutions for some applications as described in this research paper.

This research extended the body of knowledge in two ways. One way is that the QPF instruments provide a solution for integrating quantum computing with closed loop simulations for industrial applications. The portability of the QPF instrument across two main OS platforms, Linux and MS Windows, will provide academia and industry with a solution that can be integrated with existing systems. In addition, the QPF instrument's robustness includes quantum noise modeling to provide results to be more in family with expected QPU hardware results.

A second way this research extended the body of knowledge is in distributed quantum computing for parallel processing. Specifically, this research supports complex QC algorithm development, integration, maturation, and optimizations using multi-QPU simulators to mitigate Qubit count limitations with current QPUs. Also, the research demonstrated how data partitioning and QC distribution translate to performance improvements (Davis et al., 2023). This research also expanded on current frameworks like Qiskit by adding a QPF instrumentation layer that distributes QC and QPU interface thread synchronization (Qiskit Ecosystem, 2025; Kan et al., 2024).

In summary, this research contributes to the literature a portable, noise-aware, multi-QPU framework that supports scalable quantum algorithm development beyond single-QPU Qubit limits. This research will benefit several applications domains by providing expedited processing of complex QC algorithms. The salient areas that will benefit from this research include molecular medicine drug discovery, artificial intelligence, machine learning, quantum networks, cryptography, image processing, ML quantum error correction, and quantum physics modeling (Balamurugan et al., 2024; Shafique et al., 2024; Melvin, 2022).

References

Abeni, L., Balsini, A., & Cucinotta, T. (2019), “Container-specific-time Scheduling in the Linux Kernel.” ACM SIGBED Review - Special Issue on Embedded Operating Systems Workshops (EWiLi’17 and EWiLi’18), 16(3), 33–38. https://doi.org/10.1145/3373400.3373405

AbuGhanem, M. (2024). IBM quantum computers: evolution, performance, and future directions. arXiv preprint arXiv:2410.00916.

AbuGhanem, M. (2025). Superconducting quantum computers: Who is leading the future? EPJ Quantum Technology, 12(1), 102.

Ahilan, A., & Jeyam, A. (2023). Breaking Barriers in Conventional Cryptography by Integrating with Quantum Key Distribution. Wireless Personal Communications: An International Journal, 129(1), 549–567. https://doi.org/10.1007/s11277-022-10110-8

Alvesson, M., & Sandberg, J. (2013). *Constructing research questions: Doing interesting research.* SAGE Publications Ltd.

Altair. (2026 27 January). Rapidminer Data Analytics. https://altair.com/data-analytics

Anand Tech. (2024 15 April). Dual sandy Bridge for Servers. https://www.anandtech.com/Show/Index/5553?cPage=5&all=False&sort=0&page=1&slug=the-xeon-e52600-dual-sandybridge-for-servers

Asaka, R., Sakai, K., & Yahagi, R. (2019). *Quantum circuit for the fast Fourier transform*. https://doi.org/10.1007/s11128-020-02776-5

Babakov, R. M., & Barkalov, A. A. (2025). An Algorithm for Solving the Problem of Algebraic Synthesis of a Finite-State Machine with Datapath of Transitions Based on a Matrix

Approach. Cybernetics and Systems Analysis, 61(3), 347–353. https://doi.org/10.1007/s10559-025-00773-z

Bai, S., Luo, H., Dong, B., Zhou, J., & Wu, F. (2025). Locality-Aware Data Placement for NUMA Architectures: Data Decoupling and Asynchronous Replication. 2025 Design, Automation & Test in Europe Conference (DATE), Design, Automation & Test in Europe Conference (DATE), 2025, 1–7. https://doi.org/10.23919/DATE64628.2025.10993285

Bajpai, M., Gupta, P., & Munshi, P. (2015). Fast multi-processor multi-GPU based algorithm for tomographic inversion for 3D image reconstruction. *International Journal of High-Performance Computing Applications*, *29*(1), 64–72. https://doi.org/10.1177/1094342013518444

Balamurugan, K. S., Sivakami A., Mathankumar M., Satya prasad, Y. J. D., & Ahmad, I. (2024). Quantum computing basics, applications, and future perspectives. *Journal of Molecular Structure*, *1308*. https://doi.org/10.1016/j.molstruc.2024.137917

Basha, S. J., Shaik, S., Nagori, N., & Shetty, V. (2023). Mobile Gaming Experience: An Approach Based on Thread Scheduler & Thread Priority Manager. 2023 IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC), High Performance Computing, Data, and Analytics (HiPC), 2023 IEEE 30th International Conference on, HIPC, 31–40. https://doi.org/10.1109/HiPC58850.2023.00018

Biswas, R., & Shlizerman, E. (2022). Statistical perspective on functional and causal neural connectomics: The Time-Aware PC algorithm. *PLoS Computational Biology*, *18*(11), e1010653. https://doi.org/10.1371/journal.pcbi.1010653

Bloomfield, J., & Fisher, M. J. (2019). Quantitative research design. Journal of the Australasian

Rehabilitation Nurses Association (JARNA), 22(2), 27–30. https://doi-org.proxy1.ncu.edu/10.33235/jarna.22.2.27-30

Bradley, O. (2023). *Quantum Monte Carlo Simulation of Electron-Phonon Models and Computational Studies of Quantum Spin Systems*.

Bravyi, S., Dial, O., Gambetta, J. M., Gil, D., & Nazario, Z. (2022). The Future of Quantum Computing with Superconducting Qubits. Journal of Applied Physics 132, 160902 (2022). https://doi.org/10.1063/5.0082975

Brunk, G., & Lübbig, H. (1981). Thermal noise generated by quantum phase relaxations in Josephson tunnel junctions. Journal of Low Temperature Physics, 42(3–4), 179–185. https://doi.org/10.1007/bf00120199

Butterfield, A., Ngondi, G., & Kerr, A. (Eds.), A Dictionary of Computer Science. Oxford University Press. Retrieved 9 Sep. 2025, from https://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-3922.

Chaduvula, K., Indira, D. N. V. V. S. L. S., Markapudi, B., & Kalyanapu, S. (2024). Quantum edge detection of medical images using novel enhanced quantum representation and hill entropy approach. *Signal, Image & Video Processing*, *18*(2), 1803–1819. https://doi.org/10.1007/s11760-023-02857-9

Chakraborty, S., Mandal, S. B., & Shaikh, S. H. (2022). Quantum image processing: challenges and future research issues. *International Journal of Information Technology: An Official Journal of Bharati Vidyapeeth's Institute of Computer Applications and Management*, *14*(1), 475–489. https://doi.org/10.1007/s41870-018-0227-8

Chalumuri, A., Kune, R., Kannan, S., & Manoj, B. S. (2022). Quantum–Classical Image Processing for Scene Classification. *IEEE Sensors Letters, Sensors Letters, IEEE, IEEE Sens. Lett*, *6*(6), 1–4. https://doi.org/10.1109/LSENS.2022.3173253

Chen, T., Ning, Y., Amritkar, A., & Qin, G. (2018). Multi-GPU solution to the lattice Boltzmann method: An application in multiscale digital rock simulation for shale formation. *Concurrency & Computation: Practice & Experience*, *30*(19), 1. https://doi.org/10.1002/cpe.4530

Creswell, J. W., & Creswell, J. D. (2017). Research design: Qualitative, quantitative, and mixed methods of approach. Sage publications.

Crnkovic, G. D. (2010). Constructive research and info-computational knowledge generation. Model-Based Reasoning in Science & Technology, 359–380. https://doi.org/10.1007/978-3-642-15223-8_20

Cong, I., Choi, S. & Lukin, M.D. (2019). Quantum convolutional neural networks. Nat. Phys. 15, 1273–1278. https://doi.org/10.1038/s41567-019-0648-8

Davis, M. G., Chung, J., Englund, D., & Kettimuthu, R. (2023). Towards Distributed Quantum Computing by Qubit and Gate Graph Partitioning Techniques. 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), Quantum Computing and Engineering (QCE), 2023 IEEE International Conference, QCE, 01, 161–167. https://doi.org/10.1109/QCE57702.2023.00026

de Forges de Parny, L., Alibart, O., Debaud, J., Gressani, S., Lagarrigue, A., Martin, A., Metrat, A., Schiavon, M., Troisi, T., Diamanti, E., Gélard, P., Kerstel, E., Tanzilli, S., & Van Den Bossche, M. (2023). Satellite-based quantum information networks: use cases,

architecture, and roadmap. *Communication Physics*, *6*(1). https://doi.org/10.1038/s42005-022-01123-7

de Ronde, C., & Cesar, M. (2018). *Against "Particle Metaphysics" and "Collapses" within the Definition of Quantum Entanglement*.

Deutsch, David. (1992). Rapid Solution of Problems by Quantum Computation. Proceedings of the Royal Society of London Series A-Mathematical Physical and Engineering Sciences 439,553.

Divya, M. B., & Prajwala, N. (2018). Facial Expression Recognition by Calculating Euclidian Distance for Eigen Faces Using PCA. 2018 International Conference on Communication and Signal Processing (ICCSP), Communication and Signal Processing (ICCSP), 2018 International Conference On, 0244–0248. https://doi.org/10.1109/ICCSP.2018.8524332

Dolciami, Chiara. (2022). A quantum circuit library for image processing. *Master's degree Thesis*, Politecnico di Torino, Corso di laurea magistrale in Ingegneria Elettronica (Electronic Engineering). http://webthesis.biblio.polito.it/id/eprint/24673

Dong, Q., Teng, R., Chen, J., Zhang, Y., & Lai, Z. (2024). Design and Implementation of Parallel Processing Algorithm for Big Data in High Performance Computing Framework. 2024 3rd International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI), Data Analytics, Computing and Artificial Intelligence (ICDACAI), 2024 3rd International Conference on, ICDACAI, 563–567. https://doi.org/10.1109/ICDACAI65086.2024.00108

Dong, Y., & Peng, C. (2023). Multi-GPU multi-display rendering of extremely large 3D environments. *Visual Computer*, *39*(12), 6473–6489. https://doi.org/10.1007/s00371-022-02740-7

Dooms, A., & Emerencia, C. (2025). Efficient quantum algorithms to break group ring cryptosystems. Journal of Information Security and Applications, 88. https://doi.org/10.1016/j.jisa.2024.103923

Dreamstime. (December 4, 2023). A sensitive matrix of the digital photo camera. https://www.dreamstime.com/stock-photo-sensitive-matrix-digital-photo-camera-olympus-image68322172

Eilemann, S., Makhinya, M., & Pajarola, R. (2009). Equalizer: A Scalable Parallel Rendering Framework. IEEE Transactions on Visualization and Computer Graphics, Visualization and Computer Graphics, IEEE Transactions on, IEEE Trans. Visual. Comput. Graphics, 15(3), 436–452. https://doi.org/10.1109/TVCG.2008.104

Escudier, M., & Atkins, T. (2019). control system. In A Dictionary of Mechanical Engineering. Oxford University Press. Retrieved 9 Sep. 2025, from https://www.oxfordreference.com/view/10.1093/acref/9780198832102.001.0001/acref-9780198832102-e-1126.

Evans, D. (2021). Quick Quantum Circuit Simulation. Journal of Computer Science Research, 3(04).

Farooq, M. S., Khan, S. A., Ahmad, F., Islam, S., & Abid, A. (2014). An evaluation framework and comparative analysis of the widely used first programming languages. Plus, One, 9(2).

Fernandez-Conde, J., Cuenca-Jimenez, P., & Toledo-Moreo, R. (2022). A multi-level AI-based scheduler to increase adaptiveness in time-constrained mobile communication environments. *Natural Computing: An International Journal*, *21*(4), 525–535. https://doi.org/10.1007/s11047-020-09813-3

Ferrari, D., & Amoretti, M. (2024, June). A design framework for the simulation of distributed quantum computing. In Proceedings of the 2024 Workshop on High Performance and Quantum Computing Integration (pp. 4-10).

Ferrari, D., Carretta S., and Amoretti, M. (2023). "A Modular Quantum Compilation Framework for Distributed Quantum Computing," in *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1-13, 2023, Art no. 2500213, doi: 10.1109/TQE.2023.3303935.

Feynman, R. P. (2018). Simulating physics with computers. In *Feynman and computation* (pp. 133-153). cRc Press.

Gargees, R. S., & Scott, G. J. (2020). Multi-Stage Distributed Computing for Big Data: Evaluating Connective Topologies. 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Computing and Communication Workshop and Conference (CCWC), 2020 10th Annual, 0626–0633. https://doi.org/10.1109/CCWC47524.2020.9031227

Gambheer, R., & Bhat, M. S. (2023). CCD Sensor Based Cameras for Sustainable Streaming IoT Applications with Compressed Sensing. *IEEE Access, Access, IEEE*, *11*, 67882–67892. https://doi.org/10.1109/ACCESS.2023.3291396

Gilleland, E. (2020). Bootstrap methods for statistical inference. Part I: Comparative forecast verification for continuous variables. *Journal of Atmospheric & Oceanic Technology*, *37*(11), 2117‑2134. https://doi-org.proxy1.ncu.edu/10.1175/JTECH-D-20-0069.1

Grurl T, Fus J, Wille R. (2023), Noise-Aware Quantum Circuit Simulation with Decision Diagrams. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*

on, *IEEE Trans Comput-Aided Des Integr Circuits Syst*. 2023;42(3):860-873. doi:10.1109/TCAD.2022.3182628

Graphics Library Utility Toolkit. (2024, September 20). OpenGL Graphics Library. https://www.opengl.org/resources/libraries/glut/

Graphics Library User Interface. (2024, September 20). GLUI Git Hub Home Page. https://github.com/libglui/glui

GoPhotonics. (2023, July 3). Experts Predict Stunning Growth for Global Image Sensor Market, Reaching USD 55.8 Billion by 2032. https://www.gophotonics.com/news/details/4008-experts-predict-stunning-growth-for-global-image-sensor-market-reaching-usd-55-8-billion-by-2032

Grover, L. K. (1988). Quantum Computers Can Search Rapidly by Using Almost Any Transformation. Phys. Rev. Lett. 80, 4329 – Published 11 May 1998. DOI: https://doi.org/10.1103/PhysRevLett.80.4329

Gultom, L. M., & Amirullah, D. (2022). Analisis Deteksi Tepi Citra Dengan Quantum Hadamard Edge Detection (QHED). *Techno.Com*, *21*(4), 968–977. https://doi.org/10.33633/tc.v21i4.6708

Han, W., Li, H., & Wang, G. (2024). Adaptive Parallel Optimization Design for Signal Processing on Multi-core and Many-core Platforms. 2024 3rd International Conference on Electronics and Information Technology (EIT), Electronics, and Information Technology (EIT), 2024 3rd International Conference On, 389–394. https://doi.org/10.1109/EIT63098.2024.10762421

Harbrecht, H., Zaspel, P. (2018). A scalable H-matrix approach for the solution of boundary integral equations on multi-GPU clusters.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, *567*(7747), 209–212. https://doi.org/10.1038/s41586-019-0980-2

Hazra, A., & Gogtay, N. (2016). Biostatistics series module 2: Overview of hypothesis testing. *Indian Journal of Dermatology*, *61*(2), 137–145. https://doi-org.proxy1.ncu.edu/10.4103/0019-5154.177775

Heshmati, M. M. K., & Emami, F. (2023). Optimized design and simulation of optical section in electro-reflective modulators based on photonic crystals integrated with multi-quantum-well structures. *Optics*, *4*(1), 227-245.

Hour, L., Go, M., & Han, Y. (2024). Improving Zero-noise Extrapolation for Quantum-gate Error Mitigation using a Noise-aware Folding Method.

Huang, T.-W. (2023). qTask: Task Parallel Quantum Circuit Simulation with Incrementality. *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Parallel and Distributed Processing Symposium (IPDPS), 2023 IEEE International*, *IPDPS*, 746–756. https://doi.org/10.1109/IPDPS54959.2023.00080

IBM Pricing. (2025 July 6). Explore access options. https://www.ibm.com/quantum/pricing

Ichikawa, T., Hakoshima, H., Inui, K., Ito, K., Matsuda, R., Mitarai, K., ... & Fujii, K. (2023). A comprehensive survey on quantum computer usage: How many Qubits are employed for what purposes. arXiv preprint arXiv:2307.16130.

Ilari, J. (1990). Experimental Results on IR Sensor Simulation. *IFAC-PapersOnLine*, *23*(3), 69–74. https://doi.org/10.1016/S1474-6670(17)52536-7

International Business Machines (2024, April 20). IBM Quantum Platform. Computer Resources – All Systems. https://quantum.ibm.com/services/resources?tab=systems

Iyengar, S. S., Kumar, L. K. J., & Mastriani, M. (2020). *Analysis of five techniques for the internal representation of a digital image inside a quantum processor*.

Jahin, M. A., Mridha, M. F., Aung, Z., Dey, N., & Sherratt, R. S. (2024). *TriQXNet: Forecasting Dst Index from Solar Wind Data Using an Interpretable Parallel Classical Quantum Framework with Uncertainty Quantification*.

Jimenez-Navajas, L. U. I. S., Bhler, F., Leymann, F., Piattini, M., & Vietz, D. (2024). Quantum Software Development: A Survey. *Quantum Information and Computation*, *24*(7&8), 0609-0642.

Johnson, O. (2019). General System Theory and the Use of Process Mining to Improve Care Pathways. *Studies in Health Technology and Informatics*, *263*, 11–22. https://doi.org/10.3233/SHTI190107

Kan, Y., He, J., & Xue, C. (2024). Applications for Large-Scale Quantum Neural Networks Based on Distributed Quantum Computing. 2024 16th International Conference on Wireless Communications and Signal Processing (WCSP), Wireless Communications and Signal Processing (WCSP), 2024 16th International Conference, 151–156. https://doi.org/10.1109/WCSP62071.2024.10827521

Kaufman, A., Sundy, D., & McGuigan, M. (2019). Quantum computation for early universe cosmology. 2019 New York Scientific Data Summit (NYSDS), Scientific Data Summit (NYSDS), 2019 New York, 1–6. https://doi.org/10.1109/NYSDS.2019.8909801

Koch, D., Wessing, L., & Alsing, P. M. (2019). *Introduction to Coding Quantum Algorithms: A Tutorial Series Using Qiskit*.

Konnik, M., & Welsh, J. (2014). *High-level numerical simulations of noise in CCD and CMOS photodetectors: review and tutorial*.

Kousiopoulos, G.-P., & Nikolaidis, S. (2024). Performance Evaluation of Different Finite State Machines Implementations. 2024 13th International Conference on Modern Circuits and Systems Technologies (MOCAST), Modern Circuits and Systems Technologies (MOCAST), 2024 13th International Conference On, 1–5. https://doi.org/10.1109/MOCAST61810.2024.10615817

Kop, M. (2025). A Principled Approach to Quantum Technologies. Available at SSRN.

Laguna, I., Ahn, D. H., de Supinski, B. R., Gamblin, T., Lee, G. L., Schulz, M., Bagchi, S., Kulkarni, M., Zhou, B., Chen, Z., & Qin, F. (2015). Debugging high-performance computing applications at massive scales. *Communications of the ACM*, *58*(9), 72–81. https://doi.org/10.1145/2667219

Lassenius, C., Soininen, T., & Vanhanen, J. (2001). Constructive Research. SobertITIT, 1–10. http://www.pm.lth.se/fileadmin/_migrated/content_uploads/3._constructive_research.pdf

Levin, M. S. (2005). Modular system synthesis: example for composite packaged software. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *IEEE Trans. Syst., Man, Cybern. C*, *35*(4), 544–553. https://doi.org/10.1109/TSMCC.2004.840065

Li, T., Zhao, P., Zhou, Y., & Zhang, Y. (2023). Quantum Image Processing Algorithm Using Line Detection Mask Based on NEQR. *Entropy*, *25*(5), 738. https://doi.org/10.3390/e25050738

Li, Y., Hao, X., Liu, G., Shang, R., & Jiao, L. (2024). QEA-QCNN: optimization of quantum convolutional neural network architecture based on quantum evolution. Memetic Computing, 16(3), 233–254. https://doi.org/10.1007/s12293-024-00417-3

Lin, S. S. (Beijing), Guo, B. (Beijing), Shum, H.-Y. (Beijing), & Gu, J. (Beijing). (2011). *Radiometric calibration from a single image*.

Lindsay, D., Gill, S. S., Smirnova, D., & Garraghan, P. (2021). The evolution of distributed computing systems: from fundamental to new frontiers. Computing, 103(8), 1859–1878. https://doi.org/10.1007/s00607-020-00900-y

Mastriani, M. (2020). *Quantum Image Processing: the truth, the whole truth, and nothing but the truth about its problems with internal image representation and outcomes of recovery*.

Meier, F., & Yamasaki, H. (2023). Energy-Consumption Advantage of Quantum Computation.

Melvin, T. (2022). High-Dimensional Signal Processing using Classical Quantum Machine Learning Pipelines with the TensorFlow Stack, Cirq-NISQ, and Vertica. 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), Quantum Computing and Engineering (QCE), 2022 IEEE International Conference on, QCE, 793–795. https://doi.org/10.1109/QCE53715.2022.00121

Miceli, R., & McGuigan, M. (2019). Effective matrix model for nuclear physics on a quantum computer. 2019 New York Scientific Data Summit (NYSDS), Scientific Data Summit (NYSDS), 2019 New York, 1–4. https://doi.org/10.1109/NYSDS.2019.8909693

Moric, Z., Milovec, M., & Petrunic, R. (2024). Application of Quantum Cryptography in Securing Network Communications. Annals of DAAAM & Proceedings, 35, 55–64. https://doi.org/10.2507/35th.daaam.proceedings.008

Muthukrishnan, H. (2022). *Improving Multi-GPU Strong Scaling Through Optimization of Fine-Grained Transfers*.

Obaidat, M. T. (2020). Cellular phone-based system for transportation engineering applications. *Alexandria Engineering Journal*, *59*(3), 1197–1204. https://doi.org/10.1016/j.aej.2020.01.040

Ortiz, A., & Oliver, G. (2004). Radiometric calibration of CCD sensors: dark current and fixed pattern noise estimation. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on Robotics and Automation*, *5*, 4730. https://doi.org/10.1109/ROBOT.2004.1302465

Paracha, U. I., Rizvi, S. M. A., Gondel, M. M. U., Park, W., & Shin, H. (2024). Adaptive Quantum Readout Error Mitigation with Transfer Learning. 2024 15th International Conference on Information and Communication Technology Convergence (ICTC), Information and Communication Technology Convergence (ICTC), 2024 15th International Conference on I, 506–511. https://doi.org/10.1109/ICTC62082.2024.10826964

Parekh, R., Ricciardi, A., Darwish, A., and DiAdamo, S. (2021). "Quantum Algorithms and Simulation for Parallel and Distributed Quantum Computing," *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, St. Louis, MO, USA, 2021, pp. 9-19, doi: 10.1109/QCS54837.2021.00005.

Peng, P. (2012). *Capturing structural distortions in digital images and videos*.

Peres, A. *Quantum Theory: Concepts and Methods*. Springer. (2002). Accessed July 11, 2024. https://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=nlebk&AN=69626&site=eds-live&scope=site

Pitowsky, I. (2020). The Argument Against Quantum Computers. Quantum, Probability, Logic: The Work and Influence of Itamar Pitowsky, 399.

Potma, E. O., Knez, D., Ettenberg, M., Wizeman, M., Nguyen, H., Sudol, T., & Fishman, D. A. (2021). High-speed 2D and 3D mid-IR imaging with an InGaAs camera. *APL Photonics*, *9*.

Qiskit Ecosystem. Qiskit Marchine Leraning 0.8.2. (2025). The Quantum Convolutional Neural Network. Accessed Jan 6, 2025. https://qiskit-community.github.io/qiskit-machine-learning/tutorials/11_quantum_convolutional_neural_networks.html

Qiskitextbook2023. Various Authors. (Accessed August 6, 2025). Qiskit Textbook. Quantum edge detection. Github. https://github.com/Qiskit/textbook/blob/main/notebooks/ch-applications/quantum-edge-detection.ipynb

QuantumXChange. Quantum Cryptography, Explained. (Accessed May 6, 2025). https://quantumxc.com/blog/quantum-cryptography-explained/

Rached, S. B., Sun, Z., Khan, J., Long, G., Rodrigo, S., Almudever, C. G., Alarcon, E., & Abadal, S. (2025). Modeling Quantum Links for the Exploration of Distributed Quantum Computing Systems. 2025 25th Anniversary International Conference on Transparent Optical Networks (ICTON), Transparent Optical Networks (ICTON), 2025 25th Anniversary International Conference On, 1–4. https://doi.org/10.1109/ICTON67126.2025.11125135

Radanliev, P. (2024). Artificial intelligence and quantum cryptography. Journal of Analytical Science & Technology, 14, 1–17. https://doi.org/10.1186/s40543-024-00416-6

Rasmusson, A.J., Barney, R. (April 19, 2025). Quantum Cryptography: Quantum Key Distribution. https://github.com/qiskit-community/qiskit-community-

tutorials/blob/master/awards/teach_me_qiskit_2018/quantum_cryptography_qkd/Quantum_Cryptography2.ipynb

Rennie, R., & Law, J. (Eds.), A Dictionary of Physics. Oxford University Press. Retrieved 9 Sep. 2025, from https://www.oxfordreference.com/view/10.1093/acref/9780198821472.001.0001/acref-9780198821472-e-3950.

Reddy, Y.R., and Y. P. K R. (2024). "A Brief Review and Challenges in Quantum Computing," *2024 IEEE Space, Aerospace and Defence Conference (SPACE)*, Bengaluru (Bangalore), India, 2024, pp. 548-551, doi: 10.1109/SPACE63117.2024.10668232.

Reubens, R. (2016). To craft, by design, for sustainability: Towards holistic sustainability design for developing country enterprises. [Doctoral dissertation, Delft University of Technology. http://resolver.tudelft.nl/uuid:0c2c14c8-9550-449d-b1ff-7e0588ccd6c2 (25.11.2017)

Roh, E. J., Shim, J. Y., Kim, J., & Park, S. (2025). Hybrid quantum-classical 3D object detection using multi-channel quantum convolutional neural network: Hybrid quantum-classical 3D object detection. The Journal of Supercomputing: An International Journal of High-Performance Computer Design, Analysis, and Use, 81(3). https://doi.org/10.1007/s11227-025-06968-7

Ruan, Y., Xue, X., & Shen, Y. (2021). Quantum Image Processing: Opportunities and Challenges. *Mathematical Problems in Engineering*, 1–8. https://doi.org/10.1155/2021/6671613

Sebastian-Lombrana, A., Ortiz, L., Brito, J. P., Faba, J., Mendez, R. B., De Buruaga, J. S., Vicente, R. J., Setien, J., Romero, J. J., Escribano, C., Salas, P., Bejarano, J. L., & Martin,

V. (2025). Advancing the Future of Quantum Communication Networks: the new MadQCI. 2025 25th Anniversary International Conference on Transparent Optical Networks (ICTON), Transparent Optical Networks (ICTON), 2025 25th Anniversary International Conference On, 1–5. https://doi.org/10.1109/ICTON67126.2025.11125393

Sager-Smith, L. M. (2023). *Exploration of Highly Correlated Systems on Classical and Quantum Devices* [The University of Chicago].

Salcedo, J. C., & Ahmed, K. (2025). Quantum Parallel Processing Framework for Image Processing Applications. 2025 IEEE 16th International Symposium on Autonomous Decentralized Systems (ISADS), Autonomous Decentralized Systems (ISADS), 2025 IEEE 16th International Symposium on, ISADS, 16–20. https://doi.org/10.1109/ISADS66912.2025.00007

Saxena, Sanjay & Sharma, Shiru & Sharma, Neeraj. (2016). Parallel Image Processing Techniques, Benefits, and Limitations. *Research Journal of Applied Sciences, Engineering and Technology.* 12. 223-238. 10.19026/rjaset.12.2324

Sentenac, T., Le Maoultt, Y., Rolland, G., & Devy, M. (2003). Temperature correction of radiometric and geometric models for an uncooled CCD camera in the near infrared. *IEEE Transactions on Instrumentation and Measurement, Instrumentation and Measurement, IEEE Transactions on, IEEE Trans. Instrum. Meas*, *52*(1), 46–60. https://doi.org/10.1109/TIM.2003.809103

Shafique, M. A., Munir, A., & Latif, I. (2024). Quantum Computing: Circuits, Algorithms, and Applications. *IEEE Access, Access, IEEE, 12*, 22296–22314. https://doi.org/10.1109/ACCESS.2024.3362955

Shor, P.W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. Proceedings 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 1994, pp. 124-134, doi: 10.1109/SFCS.1994.365700.

Shukla, A., & Vedula, P. (2023). A hybrid classical quantum algorithm for digital image processing. *Quantum Information Processing*, *22*(1). https://doi.org/10.1007/s11128-022-03755-8

Sil, A., Betkerur, J., & Das, N. (2019). P-Value demystified. *Indian Dermatology Online Journal*. *10*(6), 745.

Slide Serve. Two Issue Super Scalar CPU. (2022, March 5). https://www.slideserve.com/emerson-vinson/two-issue-super-scalar-cpu

Solar, M., Villacura, J.-P., Cisternas Alvarez, F., & Dombrovskaia, L. (2024). Detectando un Espía con Criptografía Cuántica. Memoria Investigaciones En Ingeniería, 27, 200–219. https://doi.org/10.36561/ING.27.13

Solarte-Martínez, G. R., Trejos-Buriticá, O. I., & Muñoz-Guerrero, L. E. (2021). SOA Distributed Systems Architecture. Scientia et Technica, 26(3), 328–334. https://doi.org/10.22517/23447214.22291

Stokowski, H. S., McKenna, T. P., Park, T., Hwang, A. Y., Dean, D. J., Celik, O. T., Ansari, V., Fejer, M. M., & Safavi-Naeini, A. H. (2023). Integrated quantum optical phase sensor in thin film lithium niobate. *Nature Communications*, *14*(1), 1–11. https://doi.org/10.1038/s41467-023-38246-6

Sykot, Arman, Azad, Md Shawmoon, Tanha, Wahida Rahman, Morshed, B.M. Monjur, Shubha, Syed Emad Uddin, & Mahdy, M.R.C. (2025). Multi-layered security system: Integrating quantum key distribution with classical cryptography to enhance steganographic security.

Alexandria Engineering Journal, 121(167–182), 167–182. https://doi.org/10.1016/j.aej.2025.02.056

van Dijk, J. P. G., Charbon, E., & Sebastiano, F. (2019). The electronic interface for quantum processors is important. *Microprocessors & Microsystems*, *66*, 90–101. https://doi.org/10.1016/j.micpro.2019.02.004

Vasconcelos, F. (2020). *Quantum Computing @ MIT: The Past, Present, and Future of the Second Revolution in Computing*.

Verified Market Research (2024, May 15). Global CCD Camera Market Size by Type (Area Scan Camera, Line Scan Camera), By Application (Manufacturing, Security, Surveillance), By Geographic Scope and Forecast. https://www.verifiedmarketresearch.com/product/ccd-camera-market/

Vinod, V., Lyu, D., Ruth, M., R. Schreiner, P., Kleinekathöfer, U., & Zaspel, P. (2025). Predicting Molecular Energies of Small Organic Molecules with Multi-Fidelity Methods. Journal of Computational Chemistry, 46(6), 1–13. https://doi.org/10.1002/jcc.70056

Wang, C., Chen, M.-C., Lu, C.-Y., & Pan, J.-W. (2021). Optimal readout of superconducting Qubits exploiting high-level states. Fundamental Research, 1(1), 16–21. https://doi.org/10.1016/j.fmre.2020.12.008

Wang, A., & Zhang, Z. (2024). Model and Analysis of networked finite state machines with control channels. 2024 14th Asian Control Conference (ASCC), Control Conference (ASCC), 2024 14th Asian, 631–635.

Wang, J., Ma, A., Zhong, Y., Zheng, Z., & Zhang, L. (2022). Cross-sensor domain adaptation for high spatial resolution urban land-cover mapping: From airborne to spaceborne imagery. *Remote Sensing of Environment 277*. https://doi.org/10.1016/j.rse.2022.113058

Xu, Y., Bian, D., Ju, C.-W., Zhao, F., Xie, P., Wang, Y., Hu, W., Sun, Z., Zhang, J. Z. H., & Zhu, T. (2025). Pretrained E (3)-equivariant message-passing neural networks with multi-level representations for organic molecule spectra prediction. NPJ Computational Materials, 11(1), 1–10. https://doi.org/10.1038/s41524-025-01698-z

Yan, F., Venegas-Andraca, S. E., & Hirota, K. (2023). Toward implementing efficient image processing algorithms on quantum computers. *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, *27*(18), 13115–13127. https://doi.org/10.1007/s00500-021-06669-2

Yang, Z., Meng, L., Chung, J.-W., & Chowdhury, M. (2023). Chasing Low-Carbon Electricity for Practical and Sustainable DNN Training.

Yao, Xi-Wei, et al. "Quantum image processing and its application to edge detection: theory and experiment." Physical Review X 7.3 (2017): 031041. https://arxiv.org/abs/1801.01465

Yuan-Cheng Lai, Han Lin, Yen-Hung Chen, Yu-Ling Hsiao, & Liang-Chun Chen. (2024). Selecting and Developing a Quantum Internet Simulator for the Needs of New Audience: Practices and Considerations. IEEE Access, 12, 170965–170979. https://doi.org/10.1109/ACCESS.2024.3498339

Zamora, Y. (2022). *Machine Learning for Performance Acceleration and Prediction in Scientific Computing*. https://doi.org/10.6082/uchicago.4773

Zhang, Y., Gong, Y., Fan, L., Wang, Y., Han, Z., & Guo, Y. (2025). Efficient Entanglement Routing for Satellite-Aerial-Terrestrial Quantum Networks. 2025 34th International Conference on Computer Communications and Networks (ICCCN), Computer Communications and Networks (ICCCN), 2025 34th International Conference On, 1–9. https://doi.org/10.1109/ICCCN65249.2025.11133770

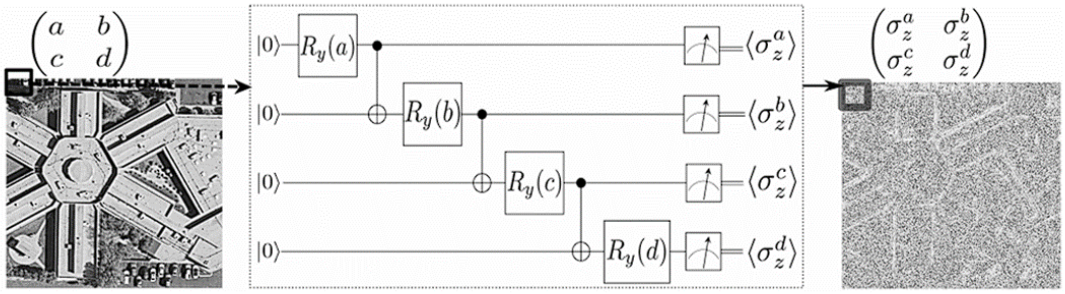
Zheng, J., Gao, Q., Ogorzalek, M., Lu, J., & Deng, Y. (2025). A Quantum Spatial Graph Convolutional Neural Network Model on Quantum Circuits. IEEE Transactions on Neural Networks and Learning Systems, Neural Networks and Learning Systems, IEEE Transactions on, IEEE Trans. Neural Net. Learning Syst, 36(3), 5706–5720. https://doi.org/10.1109/TNNLS.2024.3382174

Zhou, L., & Preindl, M. (2023). Hierarchical Software-Defined Control Architecture With MPC-Based Power Module to Interface Renewable Sources and Motor Drives. *IEEE Transactions on Sustainable Energy, Sustainable Energy, IEEE Transactions on, IEEE Trans. Sustain. Energy*, *14*(1), 83–96. https://doi.org/10.1109/TSTE.2022.3202957

Zhu, Y., Bouridane, A., Celebi, M. E., Konar, D., Angelov, P., Ni, Q., & Jiang, R. (2024). Quantum Face Recognition with Multigate Quantum Convolutional Neural Network. IEEE Transactions on Artificial Intelligence, Artificial Intelligence, IEEE Transactions on, IEEE Trans. Artif. Intell, 5(12), 6330–6341. https://doi.org/10.1109/TAI.2024.3419077

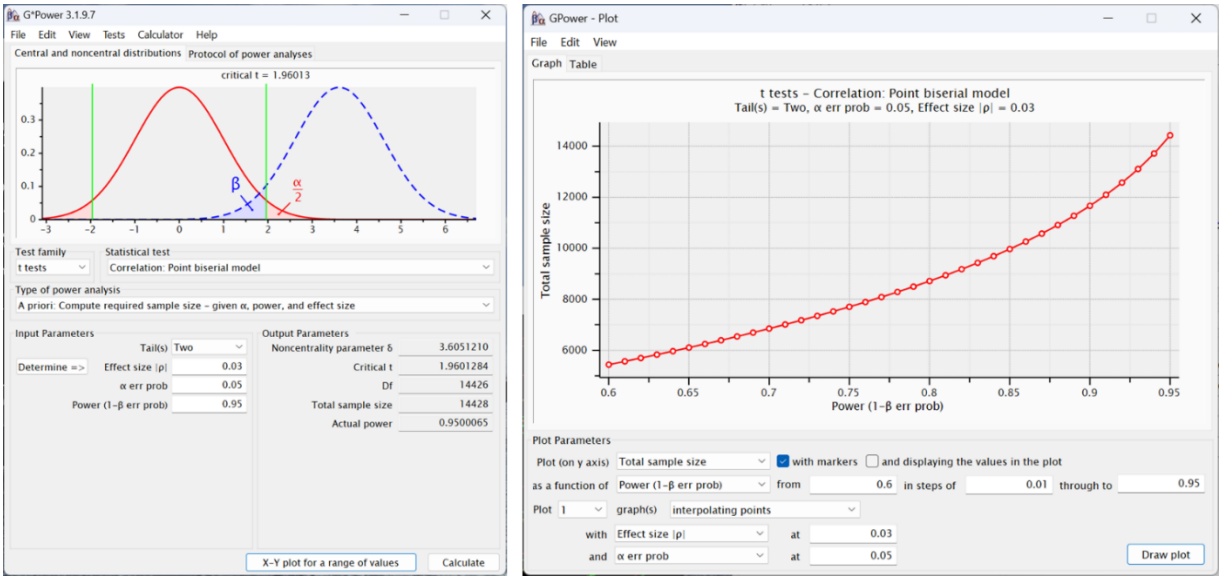
Figures

Figure 50. *Quantum Image Kernel Circuit.*



Note. Shows how a single image kernel circuit methodology is applied to an image. The figure shows how the kernel model modifies the image for emulating optical attributes and other models like fixed pattern noise, non-linearity, and other models. From Chalumuri, A., Kune, R., Kannan, S., & Manoj, B. S. (2022). Quantum–Classical Image Processing for Scene Classification. IEEE Sensors Letters, Sensors Letters, IEEE, IEEE Sens. Lett, 6(6), 1–4. https://doi.org/10.1109/LSENS.2022.3173253

Figure 51. *Sample Size for Statistical Significance.*



Note. A p-value of 0.03 demonstrates that there is less than 1 in 20 chances of results occurring by chance (Hazra & Gogtay, 2016). A sample size of 14,428 samples per dataset is required to achieve a p = 3% statistical significance cutoff. reference tool used G*Power Version 3.1.9.6.

Appendix A

Annotated Bibliography

Asaka, R., Sakai, K., & Yahagi, R. (2020). *Quantum circuit for the fast Fourier transform*. https://doi.org/10.1007/s11128-020-02776-5

The article describes the methodology for implementing FFT (Fast Fourier Transforms) to model optical blur photon dispersion with quantum circuit applications. The solution provides a proficient level of detail for the mathematical principles driving the design of a QFFT (Quantum Fast Fourier Transform) circuit. In addition, the paper addresses efficient data storage to minimize memory requirements for processing the QFFT. The paper describes in detail foundational quantum concepts such as entanglement and superposition, which drive the behavior of all quantum circuit gates including the Toffoli and Peres gates used in the model. The paper describes the arithmetic decomposition of the FFT in quantum circuits and estimation of Qubits (quantum binary digits) required for processing an image of size n pixels. A strength of the study is that it shows that the storage requirements are less than a classical FFT due to quantum superposition. A weakness of the study is that it does not provide specific details on the quantum framework used to evaluate and collect results for model validation. Future work is required to demonstrate the concepts of QFFT experimentation using quantum computing assets. Future work to demonstrate quantum timing benefits and QBIT storage achieved versus classical based FFT algorithms would benefit this study. This information would help identify if QFFT has advantages over FFT for all image sizes.

Balamurugan, K. S., Sivakami A., Mathankumar M., Satya prasad, Y. J. D., & Ahmad, I. (2024). Quantum computing basics, applications, and future perspectives. *Journal of Molecular Structure, 1308*. https://doi.org/10.1016/j.molstruc.2024.137917

The article describes the foundations of quantum computing circuits. Quantum Computing programs, unlike classical computers and fourth generation languages, are comprised of basic circuit programs akin to logical gates. The article provides a good introduction to quantum circuit programming with a satisfactory level of detail, including basic program samples that provide the reader with a quick way to evaluate the concepts. The article covers the concept of a Qubit, entanglement, superposition, wave collapse, and how a Bloch sphere can be used to represent the state of a Qubit's probabilities as a vector from origin and angle to represent the probability of 1 or 0. The article also provides a high-level design of a classical computer system to be developed to interface with a QPU. A weakness is that more examples could help the reader better understand a broader view on the different applications of the various basic quantum circuits. Further explanation of how quantum circuits could be parallelized to expedite processing could also complement this article.

Mastriani, M. (2020). *Quantum Image Processing: the truth, the whole truth, and nothing but the truth about its problems with internal image representation and outcomes of recovery*.

The article provides a detailed methodology of how images can be encoded in a QPU for image processing algorithms. The methods explained address the positional encoding of the image in terms of row and column as well as the intensity or irradiance of the image and how it can be encoded in the angle of the Qubit or in the amplitude of the waveform. Several image encoding algorithms

are explored including Flexible Representation of Quantum Images (FRQI), Novel Enhanced Quantum Representation of digital images (NEQR), and Quantum Boolean Image Processing (QBIP). Out of which FRQI and QBIP are non-exact or error prone, NEQR is exact. Each algorithm's performance is explored in processing times. A weakness in the article is that there are few details on the quantum circuits comprising each of the three images encoding algorithms. In addition, the article provides a methodology of dividing the image into smaller planes that each gets processed separately in the QPU. This technique is useful for processing larger images on systems with smaller numbers of Qubits ranging in the hundreds. As quantum technology matures and more powerful quantum resources are developed with larger number of Qubits, an incremental experiment with state-of-the-art quantum technology could be developed to determine how much technology has progressed for image encoding in terms of accuracy and processing times.

Fernandez-Conde, J., Cuenca-Jimenez, P., & Toledo-Moreo, R. (2022). A multi-level AI-based scheduler to increase adaptiveness in time-constrained mobile communication environments. *Natural Computing: An International Journal*, *21*(4), 525–535. https://doi.org/10.1007/s11047-020-09813-3

The article provides a detailed methodology of how a multi-level scheduler can be used to model an AHB (Adaptive Hybrid Broadcast) message server for variable bandwidth networks. The layered scheduler is responsible for receiving message requests, including periodic messages scheduled to be delivered at a specified frequency. The scheduler is divided into two layers, an HLS (High-Level

Scheduler), and an LLS (Low-Level Scheduler). The HLS is responsible for receiving message requests and delegation of tasks to the LLS. The LLS in turn is responsible for servicing the message request. A strength in this design is the division of responsibilities and delegation of tasks to simplify the work each scheduler performs. The message response model prioritizes groups of messages by categorizing the response type. In addition, the layered schedulers determine which messages are to be responded based on priority and network bandwidth. The solution explores responses based on message size, priority, unmodified AHB, and combines size and priority models. The article offers a detailed example of how this solution can be implemented to service vehicle traffic broadcasting messages to outline the status of the road as a vehicle drive. For example, if a traffic accident is present down the road, this system would notify users so that new traffic routes can be considered. A strength of the article is that it offers a layer of scheduler solutions that can be leveraged for multiple applications. A weakness in the article is that there is not much clarity in how the experimental results were achieved to prove the model is robust and will generate the intended results. Future work considered is the integration of real-time processing scheduling to provide a quick response for time critical applications.

Koch, D., Patel, S., Wessing, L., & Alsing, P. M. (2020). *Fundamentals In Quantum Algorithms: A Tutorial Series Using Qiskit Continued*:

The article provides foundational concepts of quantum computing using Qiskit, which is IBM's quantum computing programming language based on Python. The article explores many key elements of quantum computing including Qubit

entanglement and superposition. The article also goes in depth, explaining the main quantum gates used to build quantum circuits, or programs. In addition, the article also covers the implementation of quantum Fourier transform. The quantum Fourier transform is a key concept that will require understanding to help in the implementation of the quantum electro optical sensor model. Fourier transforms are used to apply optical Point Spread Functions (PSF) convolution to model optical effects on an image collected by a CCD camera device. Convolution is an expensive processing operation that will be explored as a model to be executed in a QPU to determine if processing can be expedited. An observation in the article is that it is a tutorial, and some concepts are not explored in depth. Overall, the article offers good foundational quantum computing concepts on which deeper knowledge can be obtained from other literature.

Konnik, M., & Welsh, J. (2014). *High-level numerical simulations of noise in CCD and CMOS photodetectors*: review and tutorial:

The article provides a detailed methodology of how the noise of a CCD (Charge Coupled Device) sensor can be emulated. The problem addressed by the article is the absence of literature for a detailed noise model to mimic CCD sensor noise in lieu of having actual CCD data. The approach divides the sensor model noise across three main sub models including photon to charge, charge to voltage, and voltage to digital image. The theory supporting the model is validated with actual sensor model noise measurements used to compare against the theoretical model established by the article. An important phenomenology modeled and discussed in detail is how temperature contributes to the noise in the CCD image. The

temperature-related noise is referred to as the dark noise source and spans across all processing from photon collection to digitization of the image. The article offers a MATLAB model with a code base describing the model, which was used in the validation of the theory. The theory provides a good foundation for developing electro-optic software models to mimic noise in support of signal processing algorithms without actual empirical imagery collected from a CCD device. The strengths of the article are the clear and concise organization of noise models from photon sampling to digitization of the image. Another strength is the foundation of experimental collected data to support the theoretical sensor noise model. A weakness per se is the lack of clarity in the unit of conversion used in some of the sub models. One final observation is that the model does not provide insight into how the noise would behave under cryogenically or actively cooled devices. The article highlights the difficulty of modeling dark noise and discusses future additional work to refine the dark noise model.

Potma, E. O., Knez, D., Ettenberg, M., Wizeman, M., Nguyen, H., Sudol, T., & Fishman, D. A. (2021). High-speed 2D and 3D mid-IR imaging with an InGaAs camera. *APL Photonics*, *9*:

The article presents an innovative approach for CCD image collection that achieves 100 frames per second for mega pixel images and 500 frames per second for kilo pixel images for 2D and 3D image processing applications. The article outlines the trend of CCD designs increasing in frame rate and pixels generated. The technique relies on two NTA (Non-degenerate photon absorption) devices. A conventional device reliant on Silicon (Si) photodetectors has problems collecting

images at low power ranges. The low power nonlinear absorption of silicon is detrimental to effective image sampling at low power conditions. The proposed technology is reliant on InGaAs (Indium Gallium Arsenide) as the photo sensor. InGaAs photo sensors have the capability to capture images with low power conditions and low image exposure integration times in the range of 1 millisecond. The technology has been evaluated on MIR (Mid Infra-Red) wavebands between 3 to 12 micrometers. A strength in the study is that the theory presented is complemented with experimental results that corroborate the paper's findings. An interesting effect of low power image collection using NTA is that high speed image collection generates gas bubbles that show up in the final collected image. The paper describes that the bubble problem has been resolved; however, there are minute details as to how the problem was solved. Additional work is required to determine if the appearance of gas bubbles has been mitigated successfully with repeatable results.

Sentenac, T., Le Maoultt, Y., Rolland, G., & Devy, M. (2003). Temperature correction of radiometric and geometric models for an uncooled CCD camera in the near infrared. *IEEE Transactions on Instrumentation and Measurement, Instrumentation and Measurement, IEEE Transactions on, IEEE Trans. Instrum. Meas*, *52*(1), 46–60. https://doi.org/10.1109/TIM.2003.809103

The article describes how photonic radiation is captured by the CCD camera and some details of the process for radiation conversion into voltages, and inclusion of noise and dark current (temperature induced noise). An important aspect to understand during the development of the refence and quantum EO CCD sensor

software models is the conversion of units from the process of photons to voltages to actual pixel counts at the digitalization phase before the output image is ready for image processing. This article primarily addresses the process of calibrating an image against a black body source, the source of the projected scene to the Focal Plane Array (FPA). The importance of this article will be relevant during the development and integration of the reference EO CCD sensor software model to ensure the output is validated. The reference EO CCD sensor model will be used to validate the output of the quantum EO CCD sensor model. Quantum computing is known for not producing reliable results, and the ability to verify the output of the quantum model is required to build trust in the data collected. As with other papers, some details are left out. For example, there is little information on some unit conversions from photons to voltages that may be required to effectively design a software model that represents the conversion process. Also there seems to be missing details about how dark current can be modeled. Additional work is required to discuss the calibration of test pattern content as the image is not described in detail. The calibration test pattern image will be needed to compare the output of the reference EO CCD sensor model to ensure its output is produced in family radiometric content as the calibration test pattern. The calibration test pattern image file containing the actual pixel values will be required to evaluate the reference model.

van Dijk, J. P. G., Charbon, E., & Sebastiano, F. (2019). The electronic interface for quantum processors is incorrect. *Microprocessors & Microsystems*, *66*, 90–101. https://doi.org/10.1016/j.micpro.2019.02.004

The article describes the physics principles and specifics of hardware design and implementation of Qubits. The article highlights the importance of scalability and how a considerable number of Qubits are required to solve difficult problems and obtain quantum supremacy. A key element of the article is that it explores various hardware design options for implementing Qubits and some of the advantages and disadvantages. The article sheds light on how a Qubit is implemented in hardware analogous to how the transistor is used to implement bits in classical computers. Limitations of scaling up are addressed as well as mitigations. Now the interface to a quantum is not a challenge as is currently implemented using common shelf electronics; however, in future quantum systems were Qubits scale to millions, the interface to the QPU could become a bottleneck. Various Qubit technologies are explored including superconducting Qubits, single-electron semiconductor Qubits, multiple-electron semiconductor Qubits, singlet-triplet Qubit, exchange-only Qubit, and hybrid Qubit. The paper also addresses QPU controller specifications that drive Qubit designs for each of the abovementioned Qubit implementations. The paper provides a good introduction to fundamentals of Qubit implementation and provides the reader with an initial roadmap of technologies and associated literature to explore to obtain deeper understanding. In summary, the article provides paths for scalability and addresses some severe challenges in Qubit design and controlling scalability impact for managing large Qubits.

Zhou, L., & Preindl, M. (2023). Hierarchical Software-Defined Control Architecture With MPC-Based Power Module to Interface Renewable Sources and Motor Drives. *IEEE Transactions on Sustainable Energy, Sustainable Energy, IEEE Transactions on, IEEE Trans. Sustain. Energy*, *14*(1), 83–96. https://doi.org/10.1109/TSTE.2022.3202957

The article addresses the application of software architecture modularization and hierarchical layers of scheduling for real-time applications. The article describes the detailed design for an MPC (Model Predictive Control) power module to interface with motor drivers. The higher layers of the software hierarchy address tasks of abstraction for controlling the motor; the middle layers of the architecture provide more specific commands for controlling the motors. Finally, the lowest layer of the design covers the actual voltage and control signals necessary to execute the commands sent from the hierarchical top and middle layers. This division of work allows the highest layers of architecture to issue abstract commands to lower layers for driving motors. In addition, the paper provides a practical application with real-time constraints on processing commands including latency from command transmittal to response to task completion. The solution provided by the architecture also addresses multi-processing devices, which allow for parallel processing to address concurrency of operation of the motor control interface. Process synchronization is also addressed as tasks spread across several processors must collaborate to complete voltage and power stabilization commands. The MPC application monitors PWM (Pulse Width Modulation) for local power switches, variable power sources including solar, battery, and three phase grid power. A strength of the study is that it provides a robust framework design for real-time applications with concurrent asynchronous events. A weakness is that the study does not define application performance well.

Appendix B

Topic Description and Supporting Literature

The design of modern CCD cameras often takes place in parallel to the development of image processing algorithms that will consume the digitized imagery produced by CCD cameras. As a result of market competition, developers of CCD cameras and algorithms cannot wait until a camera prototype is physically built to start development of algorithms. The approach often taken is to develop software models of CCD cameras that will mimic the salient phenomenology produced by noise models in the camera. This capability to model cameras in software allows for development and maturation of image processing algorithms in parallel to development of the first prototype CCD sensors.

The objective of CCD sensor modeling integration is to find and resolve integration problems of image processing algorithms before the first CCD devices are produced. The use of CCD camera models accelerates maturation of image processing algorithms and expedites delivery of the final product to the market. The objective of this proposed research is to explore the feasibility of using quantum technology to address ever increasing CCD camera complexity. In addition, the research will explore how a quantum sensor model would be integrated for industrial applications.

Literature Review

The focus of the literature review was designed to obtain a deeper understanding of physics used to model modern CCD cameras, as shown in Figure 1. Modeling camera noise is foundational to emulate the effects of noise on the image. Also, literature was reviewed to obtain a deeper understanding of the physics driving measurement of radiance source and how photons are measured by a CCD camera as shown in Figure 2 (Sentenac et al., 2003). The literature

review covers quantum computing Fourier transform principles used to model optical distortions added by CCD cameras (Asaka et al., 2019). In addition, the article also covers the processing complexity of Fourier transform algorithms in QPUs as shown in Figure 3.

The literature was reviewed to obtain a deeper understanding of the fundamentals of Qubit entanglement, superposition, basic quantum circuits, and quantum collapse conditions (de Ronde et al., 2018). Articles were also reviewed to determine the most appropriate statistical methodology to apply when validating results of the quantum sensor model versus a classical sensor model reference (Sil et al., 2019). The literature was reviewed for model fidelity stemming from the first principles of modeling, which offers the highest fidelity possible. In addition, the articles reviewed also provide a balanced approach to lower fidelity models that are driven by a good enough approach in favor of speed and less complexity (Zamora, 2022). A hierarchical scheduler architecture, as shown in Figure 7, was reviewed to help in the design of the quantum framework. The quantum framework will be used to manage the quantum CCD sensor model and to collect experiment results to support model validation, which will be based on a hierarchical scheduler architecture (Fernandez-Conde et al., 2022). Finally, a concurrent layered architecture was evaluated as shown in Figure 7 to help in the design of the quantum framework to support industrial applications (Zhou, L., & Preindl, 2023).

Explanation of Gap in Literature

The problem to be addressed in this study was that a single QPU cannot solve distributed quantum computing applications that require more Qubits than available on a QPU, which limits the scalability and fidelity of image processing, cryptography, neural networks, error correction, and physics modeling (Davis et al., 2024; van Dijk et al., 2019). According to Ichikawa et al. (2023), in 2022 condensed physics applications were using more than 100 Qubits, which are

approaching Amazon 105 Qubit QPU, and IBM 127 Qubit QPU (AbuGhanem, 2025). Additionally, the pervasiveness of noise effects increases as the complexity and depth of Quantum Circuit (QC) increases, which makes a case to partition an application into smaller shallow QC and distribute processing across several QPUs (Kan et al., 2024; Davis et al., 2023).

Distributed quantum algorithm processing is needed to develop applications to solve large scale QC algorithms, including decomposition into smaller problems for complex solutions that require more Qubits than available on a single QPU (Kan et al., 2024). This research will develop a software framework for distributed parallel processing capabilities for modeling more Qubits than available in a single QPU, including performance improvement, maturation, and integration of QC algorithms for closed-loop applications.

Appendix C

Software Instruments, Data Collection Repositories & Run Instructions

The software codes for the QPF instruments are located at github via the following links:

- Repo: https://github.com/QuantumRocker/QPF-Public
- Readme: https://github.com/QuantumRocker/QPF/blob/main/README.md

The test artifacts for all data collections are located at github via the following links:

- Repo: https://github.com/QuantumRocker/QPF-Data.git
- Readme: https://github.com/QuantumRocker/QPF-Data/blob/main/README.md

Instructions for setting up Qiskit & QPF Anaconda environment:

1. Open an anaconda console session
2. ./conda create --name QSFenv
3. source activate QSFenv
4. ./conda install pip
5. ./pip install qiskit
6. ./pip install matplotlib
7. ./pip install qiskit_ibm_runtime
8. ./pip install pylatexenc
9. ./pip install ipykernel
10. ./pip install interlin-q
11. ./pip install psutils

Instructions for compiling the codes:

12. For Linux
    a. cd QPF
    b. Make clean; make all
13. For MS Windows
    a. Batch builds clean
    b. Batch build all x64

Instructions for running the 5 QC test case algorithms:

14. Change directory to QPF/x64
15. runQPFexperimentQHED.sh and runQPFexperimentQHED_AllResolutions.sh
16. runQPFexperimentQCNN.sh
17. runQPFexperimentQCCD.sh
18. runQPFexperimentQCKD.sh
19. runQPFexperimentQILQ.sh

Appendix D

Quantum Computing Lab Resources Used for This Research

Figure 52. *Quantum Computing Lab Resources.*

Mercury1106
Lenovo
16GB
Windows 11
AMD Ryzen 7 Processor
5700U
16 cores

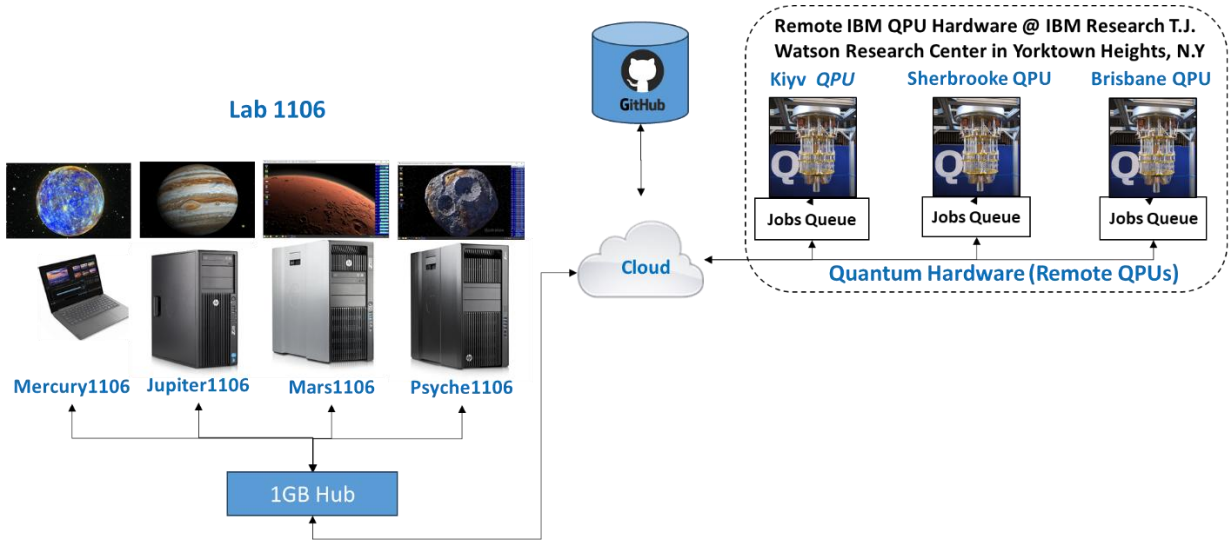
Jupiter1106
HP Z210 Workstation
8GB
Windows 10
2X Intel Xeon CPUs
E31240 @ 3.30Ghz
4 cores

Mars1106
HP Z820 Workstation
128 GB
Linux Mint 22
2X Intel Xeon CPUs
E5-2670 @ 2.60GHz
16 cores

Psyche1106
HP Z840 Workstation
125 GB
Linux Mint 22
2X Intel Xeon CPUs
E5-2697 @ 2.30GHz
36 cores

Note. Itemized computing assets.

Figure 53. *Quantum Computing Lab Resources and Remote QPUs at IBM New York Site.*

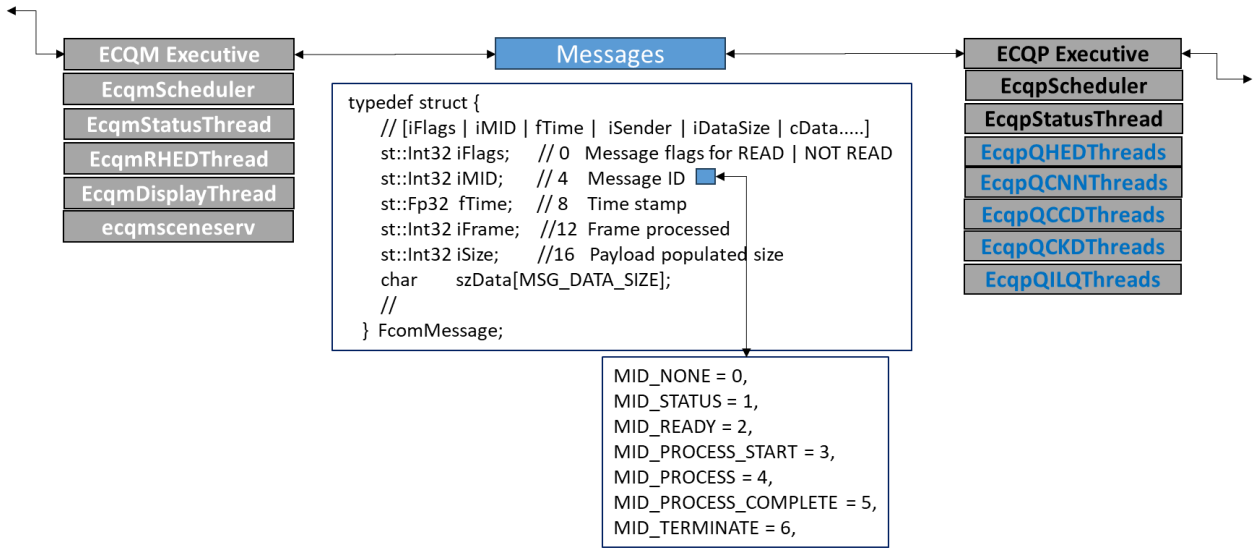


Note. Computing assts including remote QPU at IBM New York.

Appendix E

Scheduler Communications Message Architecture

Figure 54. *Hierarchical Scheduler Communications Message Structure.*



Note. Inter Scheduler Communications sends message structure with a payload of up to 4096 bytes per message. ECQM C++ main scheduler controls state machine on ECQP Python lower scheduler. Flags are used to identify when a message has been read by the scheduler.

Appendix F

VITA

JUAN CARLOS SALCEDO

EDUCATION

PhD in Computer Science: 2026, National University, San Diego, California
MBA & Technology Management: 2000, University of Phoenix, Tucson, Arizona
Bachelors of Science in Computer Engineering: 1995, University of Arizona, Tucson, Arizona

PROFESSIONAL EMPLOYMENT

Engineering Fellow, Raytheon Missiles & Defense, Jul 2001 – Present
Sr. Scientist Microcode Development, IBM, Jan 2000 - Jul 2001
Sr. Software Engineer, Environmental Systems Products, Jun 1995 - Jan 2000
Software Engineer, Westex by Milliken, Jun 1992 - May 1995

PUBLICATIONS

Salcedo, J. C., & Ahmed, K. (2025). Quantum Parallel Processing Framework for Image Processing Applications. 2025 IEEE 16th International Symposium on Autonomous Decentralized Systems (ISADS), Autonomous Decentralized Systems (ISADS), 2025 IEEE 16th International Symposium on, ISADS, 16–20. https://doi.org/10.1109/ISADS66912.2025.00007

Abstract submitted Nov 2, 2025 to CRC Press Taylor & Frances Group - Next-Generation Quantum Computing Applications. The submittal will contribute an article for Chapter 3; Quantum Cryptographic Applications. The scope of the article will expand on distributed generation of cryptographic distribution algorithms

Abstract submitted Jan 22, 2026 to 2026 Selected Areas in Cryptography (SAC) - Canada's research conference on cryptography at University of Ottawa, Canada. Quantum optimization and performance track: The scope of the article will expand on distributed performance of cryptographic distribution algorithms for real-time application

Submitted doctoral work for Poster of The Year (PoY) and was accepted on Feb 18, 2026. Presentation hosted on May 15, 2026 at National University, San Diego as part of 2026 commencement activities.

PRESENTATIONS AT PROFESSIONAL MEETINGS

Salcedo, J. C., & Ahmed, K. (2025). Quantum Parallel Processing Framework for Image Processing Applications. 2025 IEEE 16th International Symposium on Autonomous Decentralized Systems (ISADS), Autonomous Decentralized Systems (ISADS)

Many proprietary symposium presentations over the course of 30 years including HPC GPU solutions for modeling EO scene generation closed loop systems, distributed real-time architectures, distributed GPU EO sensor modeling, distributed image processing, predictive EO signature modeling, and distributed vehicle emission test systems

INTELLECTUAL PROPERTY

Four proprietary trade secrets, and one trademark

PROFESSIONAL MEMBERSHIP

Member of: IEEE society, National Society of Leadership & Success, and Omega Nu Lambda National Honor Society