Closed-loop control system and hardware-aware compilation protocols for quantum simulation with neutral atoms in optical trap arrays

by

Parth Padia

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Physics (Quantum Information)

Waterloo, Ontario, Canada, 2023

© Parth Padia 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Quantum materials offer tremendous potential for advancing electronic devices beyond traditional semiconductor-based technologies. Understanding the dynamics of these materials requires the use of quantum simulators. Quantum simulators are controlled many-body quantum systems that mimic the dynamics of a targeted quantum system. The three key features of a quantum simulator are controllability, scalability, and interactability. Controllability denotes the ability to address an individual quantum system. Scalability refers to extending this control to multiple quantum systems while maintaining their interconnectivity with a polynomial increase in resources. Interactability, on the other hand, denotes the capability to establish strong tunable interactions between a pair of quantum systems.

This thesis addresses the challenges of attaining controllability and scalability within the current Noisy Intermediate-Scale Quantum (NISQ) era, characterized by limited and error-prone qubits, for a neutral atom-based quantum simulator.

The constraints in qubit interconnectivity necessitate the use of additional swap gates for operations between non-adjacent qubits, increasing errors. To reduce these gate-based errors, we improve qubit interconnectivity by displacing atoms during simulation, thus enhancing our simulator's scalability. We compare approaches with and without atom displacement analytically and numerically, employing metrics like circuit fidelity and quantum volume. Our analysis introduces a novel metric, denoted as $\eta_{protocol}$, for comparing compilation protocols incorporating atom displacement. Additionally, we establish an inequality involving the $\eta_{platform}$ metric to compare operational protocols with and without atom displacement. We conclude from our quantum volume study that protocols assisted by atom displacement can achieve a quantum volume of 2^7, a significant improvement over the 2^6 attainable without atom displacement with the state-of-the-art two-qubit gate infidelity of $5e$-3 and atom displacement infidelity of $1.8e$-4.

Implementing a dedicated closed-loop control and acquisition system showcases our simulator's controllability. The system integrates machine learning techniques to automate experiment composition, execution, and analysis, resulting in faster and automated control parameter optimization. A practical demonstration of this optimization is conducted through imaging an atomic cloud composed of Rb-87 atoms, the first step in undertaking quantum simulations with neutral atom arrays.

The research presented in this thesis contributes to the understanding and advancement of quantum simulators, paving the way for developing new devices with quantum materials.

Acknowledgements

I would like to express my sincerest gratitude to my advisor, Alexandre Cooper-Roy, for his invaluable guidance, support, and mentorship throughout this research. His expertise and encouragement have been instrumental in shaping my academic growth as a researcher. This thesis is a result of his excitement to push the boundaries of quantum simulation with our neutral atom-based quantum simulator.

My deepest thanks go to my co-supervisor, David Cory, whose constructive feedback and encouraging words have inspired me to strive for excellence in my research.

I am grateful for my labmates Anastasiia Mashko, Artem Zhutov, and Kent Ueno, who have made the journey enjoyable and educational. Your shared knowledge and resilience in the face of challenges have been a source of constant motivation.

I extend my heartfelt appreciation to my friends Shivam, Rahul, Samridh, and Sakksham, whose jovial nature made my time outside the lab enjoyable. Additionally, I'm deeply grateful to my high-school friends Meghal, Darshil, Dhyey, and Smeet for constantly checking on me and inspiring me to improve daily. Their friendship is a gift that I will always cherish.

A special thanks to my dance team 'Waterloo Raas Warriors' for providing a much-needed escape and reminding me of the joy of life amidst the rigors of research.

To my beloved parents and sister, your unwavering faith in me and endless support have carried me through even the toughest of times. I could not have achieved this without you.

I am humbled and honored to have had the opportunity to work on this research. I am grateful for the collective efforts of my supervisor and labmates, who have contributed to its realization.

Dedication

This is dedicated to my beloved parents and dear friends.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures xi

List of Tables xiv

1 Introduction 1

1.1 Motivation 1

1.2 Thesis outline 4

2 Features of a quantum simulator: controllability, scalability, and interactability 6

2.1 Unlocking New Frontiers: Need for quantum simulators 6

2.2 Controllability: Addressing an individual quantum system 7

2.3 Scalability: Controlling multiple quantum systems with linear increase in resources 8

2.4 Interactability: Facilitating strong tunable interactions between individual quantum systems 9

2.5 Neutral Atom Arrays: Potential platform for quantum simulation 10
2.5.1 Controllability: Laser pulse driven atomic transitions 11
2.5.2 Scalability: Parallel control over an array of neutral atoms 12
2.5.3 Interactability: Engineering strong interactions between two atoms via Rydberg blockade mechanism 14

3 Experimental methods for quantum simulation with neutral atom arrays 16
3.1 Experimental Workflow: Load, rearrange, evolve, and readout neutral atom arrays 16
3.2 Experimental Architecture: Systems and hardware for neutral atom arrays 18
3.2.1 Vacuum and cooling system 20
3.2.2 Trapping system 21
3.2.3 Imaging system 23
3.2.4 Reconfiguration system 23
3.2.5 Rydberg system 25
3.2.6 Control and acquisition system 25

4 Increasing the simulation power of neutral atom arrays with dynamic architecture 27
4.1 Architecture: Topology of underlying qubits 28
4.1.1 Coupling graph: Graphical representation of physical interactions . 29
4.1.2 Connectivity graph: Graphical representation of qubit connectivity 32
4.1.3 Static architecture: Connectivity graph = truncated coupling graph 35
4.1.4 Dynamic architecture: Connectivity graph $\neq$ truncated coupling graph 36
4.2 Neutral atom array: A dynamic architecture approach 36
4.3 Five-step atom displacement protocol for dynamic architecture 38
4.4 Error sources in dynamic architecture 40
4.4.1 Coherent error 40
4.4.1.1 Gate error 40

4.4.1.2 Dephasing error . . . 41
4.4.2 Incoherent error . . . 41
4.4.2.1 Atom loss error . . . 41
4.5 Quantum circuit compilation protocol . . . 42
4.5.1 Compiling CNOT circuits for static architecture . . . 44
4.5.2 Compiling CNOT circuits for dynamic architecture . . . 45
4.5.3 Cross-benchmarking compilation protocols for static and dynamic architecture . . . 51
4.6 Performance metric: Circuit fidelity for static and dynamic architecture . . . 53
4.6.1 Cross-benchmarking operational protocols for CNOT circuit execution 55
4.7 Performance metric: Quantum volume for static and dynamic architecture 57
4.8 Conclusions and outlook . . . 65

5 Closed-loop control and acquisition system 66
5.1 Hardware architecture . . . 67
5.1.1 Slow control system . . . 68
5.1.2 Fast control system . . . 70
5.1.3 Integrating control hardware with the experimental setup . . . 71
5.1.3.1 Integrating control hardware with the cooling system . . . 74
5.2 Software architecture . . . 77
5.2.1 Labscript suite . . . 77
5.2.2 Components and technologies in labscript suite . . . 77
5.2.2.1 Labscript API: Compose and define experimental shots . . . 78
5.2.2.2 Runmanager: Compile experimental shots . . . 78
5.2.2.3 Runviewer: Visualize experimental shots . . . 79
5.2.2.4 BLACS: Execute experimental shots . . . 79
5.2.2.5 Lyse: Analyse experimental shots . . . 80
5.2.2.6 Experimental shots: Comprehensive record of the experiment 80

5.2.2.7 Camera server: Acquire images synchronously 81
5.2.2.8 ZeroMQ: Network socket for communication between labscript suite components 82
5.2.2.9 Unit conversion class: Control signals mapped to physical values 82
5.3 Software workflow 82
5.3.1 Experiment composition 83
5.3.2 Experiment execution 85
5.3.3 Experiment analysis 86
5.3.4 Experiment workflow 86
5.3.5 Closed-loop control and acquisition system 90
5.4 Conclusions and outlook 91

6 Characterizing cloud of atoms using closed-loop control and acquisition system 93
6.1 Imaging and optimizing a cloud of Rb-87 atoms 93
6.2 Optimizing control parameters with M-LOOP to expedite convergence . . . 95
6.3 Conclusions and outlook 98

References 99

APPENDICES 107

A Labscript suite codes 108
A.1 Connection table 108
A.2 Imaging an atomic cloud of Rb-87 atoms 112
A.3 Additions and modifications in the Labscript suite software package 116
A.3.1 Unit conversion class - Current source 116
A.3.2 Unit conversion class - Moglabs 117
A.3.3 Updating BLACS UI 118

A.3.4 Customized camera server 119
A.4 Summary of bug fixes 124
A.4.1 Summary of bug fixes during the installation process 124
A.4.2 Summary of bug fixes while interfacing FunctionRunner 125
A.4.3 Summary of bug fixes while interfacing Pseudo-clocks 125

List of Figures

1.1 **Quantum simulation for novel material discovery.** 3

2.1 **Relevant Rb-87 atomic levels for quantum information processing.** 12

2.2 **Generating arbitrary geometry of optical tweezers with diffractive optical elements.** 13

2.3 **Multiplexed beams for parallel execution.** 14

2.4 **Atoms exhibiting Rydberg properties when excited to a high energy state.** 15

2.5 **Rydberg blockade mechanism.** 15

3.1 **Experimental workflow for quantum simulation with neutral atom arrays.** 17

3.2 **Systems in a neutral atom-based quantum simulator experimental setup.** 19

3.3 **Modular vacuum chamber assembly on a linear translation stage.** 20

3.4 **Rb-87 D_2 transition for cooling atoms.** 21

3.5 **2D Arbitrary geometries of optical tweezers created by SLM.** . . . 22

3.6 **Dynamic optical tweezers moving an atom from one trap to another.** 24

4.1 **Truncated coupling graph of a neutral atom array platform.** . . . 32

4.2 **Connectivity graphs of IBM's superconducting qubit platforms exhibiting partial connectivity.** 34

4.3 **Connectivity graph of an ion-trap platform with five qubits.** . . . 35

4.4 Coherent transport of entangled atoms. 37
4.5 Connectivity graph of the neutral atom array platform changes after introducing atom moves. 37
4.6 Five-step atom displacement protocol for neutral atom arrays with dynamic architecture. 39
4.7 Atom loss error demonstration. 42
4.8 Quantum circuit admissibility illustration. 44
4.9 An exemplary circuit comprising eight qubits and eight CNOTs. . 45
4.10 Circuit compiled for static architecture with nearest-neighbor connectivity. 45
4.11 Atom topology for circuit optimization. 47
4.12 Compilation protocol illustration for an eight-qubit CNOT circuit. 48
4.13 Transformation matrix generation. 49
4.14 Circuit optimization algorithm for dynamic architecture. 50
4.15 Compilation protocol results. 52
4.16 Trade-off between two-qubit gate error and atom loss error to design a protocol where dynamic outperforms static architecture. 55
4.17 Efficiency of compilation protocol. 56
4.18 An exemplary square quantum volume circuit. 57
4.19 Decomposing 3-qubit quantum volume circuit to native gate set. 59
4.20 Compiling quantum volume circuit for static architecture. 60
4.21 Erroneous quantum volume circuits for both architectures. 60
4.22 A 4-qubit quantum volume circuit. 61
4.23 Output probability distribution. 61
4.24 Evidence of Haar randomness in 4-qubit quantum volume circuits. 62
4.25 Maximum allowed two-qubit gate error to achieve a certain quantum volume with static and dynamic architectures. 63
4.26 Maximum allowed two-qubit gate error and its associated heavy output probability distribution. 64

4.27 **Heavy output probability distribution for current state-of-the-art two-qubit gate and atom move fidelities on neutral atom platform.** 64

5.1 **PXIe chassis to mount digital and analog modules.** 68
5.2 **Digital, Analog and timing modules from National instrument.** . . 69
5.3 **NI PXIe 6739 analog output banks and analog output channels.** 70
5.4 **Fast control system modules.** 71
5.5 **Routing control signals to the experiment.** 72
5.6 **Spatial arrangement of the control rack (left) and the distribution rack (right).** 73
5.7 **A detailed list of control signals distributed to various hardware and their associated cable paths.** 76
5.8 **Runmanager GUI compiles experimental shots.** 85
5.9 **BLACS GUI manages and executes a queue of experimental shots.** 87
5.10 **Lyse GUI manages analysis routines and a queue of executed experimental shots.** 88
5.11 **Data flow between components of labscript suite.** 89
5.12 **Software workflow of the closed-loop control and acquisition system.** 92

6.1 **Linearly sweeping cooler frequency to find the optimal control parameter.** 94
6.2 **Automated closed-loop optimization of control parameters via M-LOOP.** 97

A.1 BLACS UI without any modifications. 118
A.2 BLACS UI after selecting relevant analog channels for display. 118
A.3 Pseudo-clock integration error when analog module (NI PXIe 6739) is introduced. 126
A.4 Pseudo-clock integration error when digital module (NI PXIe 6537) is introduced. 127

List of Tables

4.1 **Five-step atom displacement protocol to realize dynamic architecture on neutral atom platform.** 38

4.2 **Fidelity comparison of static and dynamic architecture.** 54

4.3 **Achievable quantum volume on static and dynamic architectures with the state-of-the-art two-qubit gate and atom move fidelities.** 63

5.1 **Control hardware integrated with the cooling system for amplitude modulation.** 75

5.2 **Components of the Labscript suite and their applications.** 78

5.3 **Software components and technologies used at various steps of the experiment.** 83

Chapter 1

Introduction

1.1 Motivation

Quantum materials have the potential to advance electronic devices beyond traditional semiconductor-based technologies. The emergent properties of quantum materials stem from the collective behavior of a macroscopic number of strongly interacting quantum particles. Among such materials, two-dimensional layered materials (2DLMs) have gained considerable attention following the discovery of graphene. The weak interlayer interactions enabled by van der Waals (vdW) forces in 2DLMs allow the engineering of diverse vdW hetero-structures (vdWHs) tailored to specific requirements. These vdWHs have paved the way for specialized electronic devices, including tunneling transistors, flexible electronics, and optoelectronic components such as photo-detectors and photovoltaics [Geim and Grigorieva, 2013]. The development of such devices showcases the immense potential of quantum materials and contributes to the realization of secure, environmentally friendly, and sustainable technologies [Tokura et al., 2017].

The discovery and development of new quantum materials follow a continuous cycle of five essential steps: design, fabrication, characterization, assembly, and testing. This iterative process drives the advancement of existing materials and the creation of innovative devices. During the design phase, simulating quantum materials is crucial in predicting and understanding their properties. However, simulating these properties on classical computers faces substantial challenges due to the exponential growth in parameter space and the complexities involved in capturing the emergent phenomena resulting from the interaction among strongly correlated quantum particles.

Classical simulations fall short in accurately capturing the behavior of quantum materials. The complex nature of these materials, characterized by long-range entanglement and exponential growth in variables, renders classical computational methods inadequate to study their intricate dynamics. To overcome these limitations, dedicated quantum simulators are necessary to gain insights into the properties and behavior of quantum materials in regimes inaccessible to classical simulators.

The proposal for quantum simulation has paved the way to overcome these limitations in comprehending the complexities of quantum systems. Quantum simulation entails using a controlled many-body quantum system, referred to as a quantum simulator, that mimics the dynamics of a quantum-mechanical system under investigation, as illustrated in Fig. 1.1. Through controlled studies on the dynamics of these materials using quantum simulators, we aim to better understand their unique properties using fewer computational resources than classical computers. This approach accelerates the discovery of novel materials by identifying optimal design and fabrication strategies.

Realizing a quantum simulator requires three key features: controllability, scalability, and interactability. Controllability denotes the ability to address an individual quantum system, including state initialization, manipulation, and read-out. Scalability refers to the possibility of extending this control to multiple quantum systems while maintaining their interconnectivity with a polynomial increase in resources. Interactability, on the other hand, denotes the capability to establish strong tunable interactions between a pair of quantum systems.

Quantum simulators are an invaluable tool for investigating complex quantum dynamics. However, they face significant challenges in the Noisy Intermediate-Scale Quantum (NISQ) era [Preskill, 2018]. Decoherence, the rapid loss of quantum properties due to interactions with the environment, leads to errors that compromise simulation reliability. Additionally, imperfections in physical qubits and quantum gates contribute to a high error rate. When theses error rates exceed the maximum tolerable limit set by quantum error codes, they limit the circuit depth. As a result, the complexity of feasible computations on quantum simulators is constrained.

In response to the challenges posed by quantum decoherence and high error rates inherent to the NISQ era, this thesis puts forth the concept of dynamic architecture for quantum simulators to enhance scalability. Unlike static architecture, where qubit adjacencies are fixed, dynamic architecture permits programmable on-demand adjacency. This capability mitigates the need for SWAP operations typically required for interactions among non-adjacent qubits in partially connected quantum simulators. Consequently, it leads to a reduction in circuit depth, thereby minimizing the total accumulated error.

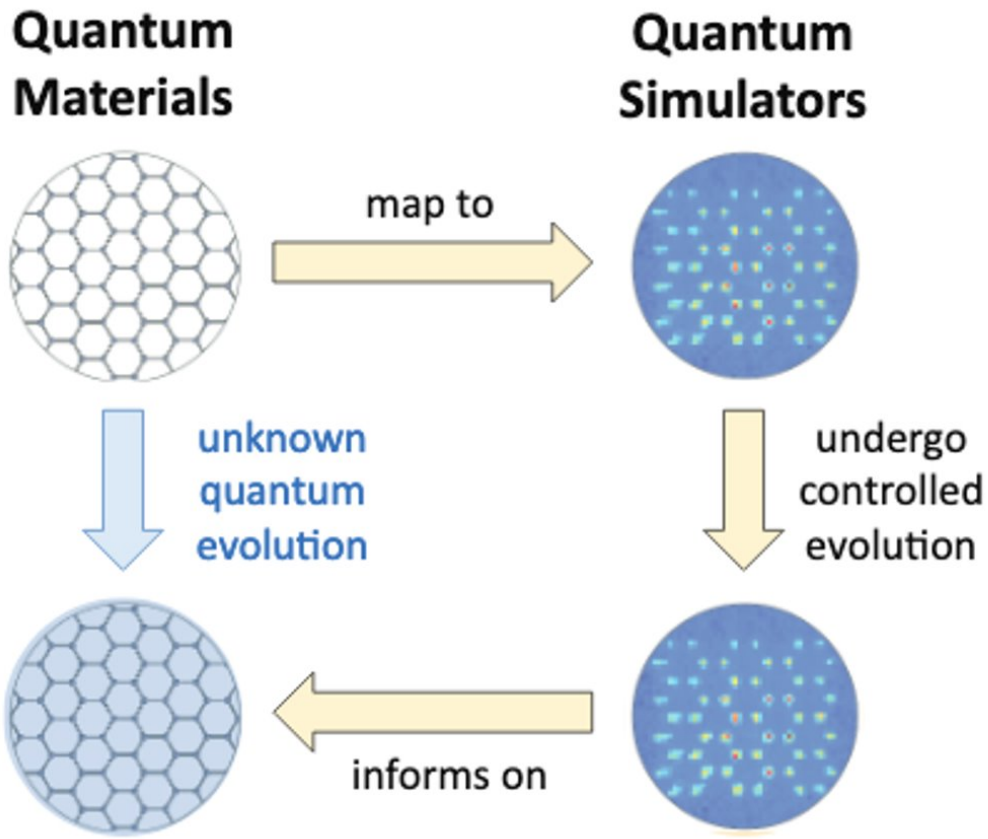


Figure 1.1: **Quantum simulation for novel material discovery.** A quantum simulator is a controlled many-body quantum system that mimics the dynamics of a quantum-mechanical system under investigation. Quantum simulators provide a powerful tool for studying the dynamics of quantum materials, expediting the discovery of novel materials. Here, a quantum simulator maps the dynamics of quantum materials to accelerate the discovery of novel materials.

This thesis further presents protocols for embodying this dynamic architecture in neutral atom arrays by shuttling atomic qubits and tailoring adjacency to match quantum circuit requirements. The superiority of the dynamic architecture over the static architecture is then numerically evaluated via cross-benchmarking under error conditions. Moreover, to enhance the simulator's controllability, the thesis details a control and acquisition system for conducting quantum simulations. A machine learning-based approach is also outlined

to automate this system to achieve faster and automated convergence. Collectively, these approaches seek to address the challenges imposed in the NISQ era and enhance the effectiveness of a quantum simulator.

1.2 Thesis outline

In Chapter 2, I first motivate the need for quantum simulators to tackle a range of issues, including the design of novel quantum materials, as discussed in Section 2.1. The chapter then dissects the key features - controllability, scalability, and interactability - necessary for building a quantum simulator. Section 2.5 outlines how a neutral atom array-based quantum simulator actualizes these features. This perspective offers a foundational understanding of how quantum simulators operate and lays the groundwork for future development.

In Chapter 3, I outline the experimental procedures for performing quantum simulation on a neutral atom array-based quantum simulator. Section 3.1 presents the sequence of operations required in such simulations, without the involved apparatus. Section 3.2 provides an overview of the systems and hardware used in these quantum simulations. This chapter, in essence, amalgamates a comprehensive literature review and experimental methodologies to conduct quantum simulations on a neutral atom array-based quantum simulator.

In Chapter 4, I describe static and dynamic architectures for quantum simulators, focusing on improving the simulator's scalability. Section 4.1 discusses these architectures, followed by the recent advancements on neutral-atom platform that enable dynamic architecture in Section 4.2. Section 4.3 introduces atom displacement protocol designed to realize dynamic architecture, while Section 4.4 considers the associated errors for the protocol. Section 4.5 outlines the circuit compilation protocol for both architectures. In Section 4.6, the circuit fidelity of both architectures is evaluated and compared, highlighting specific conditions under which the dynamic architecture surpasses the static one. In conclusion, Section 4.7 contrasts the attainable quantum volume in both architectures, indicating the maximum two-qubit gate error thresholds. This chapter offers a holistic perspective on the potential and challenges of static and dynamic architectures.

In Chapter 5, I describe a control and acquisition system designed to enhance our quantum simulator's controllability. Section 5.1 provides an overview of the control hardware and its integration with our current experimental setup. Section 5.2 details the software architecture underpinning the control and acquisition system, including a summary of Labscript suite components utilized for comprehensive experiment management. Section 5.3

further breaks down the systematic steps in managing and executing experiments. It also introduces a closed-loop control and acquisition system leveraging machine learning to automate the experimental process.

In Chapter 6, I demonstrate the capability of our control and acquisition system to create an atomic cloud of Rb-87 atoms, the first step in undertaking quantum simulations with neutral atom arrays. Section 6.1 outlines the process of creating an atomic cloud in our experimental setup and optimizes control parameters to maximize the number of atoms in the cloud. In Section 6.2, M-LOOP, a machine learning technique, is employed to optimize these control parameters automatically. This effort underscores the critical role of automation and machine learning in multi-parameter space exploration.

Chapter 2

Features of a quantum simulator: controllability, scalability, and interactability

2.1 Unlocking New Frontiers: Need for quantum simulators

The development of quantum computing marks a new era in scientific computation, pushing the limits beyond classical computing devices. At the heart of this revolution lies quantum mechanics' unique principles, such as superposition [Einstein et al., 1935] and entanglement [Dirac, 1930]. By leveraging these principles, we can realize a quantum advantage, which signifies the ability to execute specific computational tasks more efficiently than any existing classical algorithm. This advantage is no longer a theoretical promise and has been empirically validated in several cutting-edge experiments across various platforms [Arute et al., 2019] [Zhong et al., 2020] [Madsen et al., 2022]. The milestones attained suggest an expansive application scope for quantum computing. It extends from solving complex optimization problems [Shor, 1994] to enabling more efficient search algorithms [Grover, 1996] to enhanced cryptographic protocols [Ekert, 1991].

However, developing a fully functional quantum computer to solve above stated problems involves addressing technical challenges related to qubit coherence, error minimization, and scalability. Ensuring precise qubit control, cross-talk reduction, and optimizing quantum algorithms adds further complexity. Each of these challenges represents a significant

area of research in the quest to unlock the full potential of quantum computers.

While developing a universal fault-tolerant quantum computer is a long-term goal, current interest lies in evaluating the performance of quantum algorithms on near-term quantum devices. Such devices, known as Noisy Intermediate-Scale Quantum (NISQ) devices [Preskill, 2018], offer insights into the practical aspects of quantum computation. Despite their limitations in qubit number and susceptibility to errors, they allow us to assess and refine quantum algorithms, nudging us closer to the ultimate goal.

Similarly, quantum simulators are tailor-made instruments designed to tackle distinct quantum problems. Notably, they are instrumental in accelerating the process of quantum materials discovery. A quantum simulator is a controllable many-body quantum system that emulates the dynamics of a target quantum system under investigation. It equips us with the necessary tools to understand complex quantum phenomena beyond the scope of classical computations [Feynman, 1982]. Therefore, quantum simulators complement quantum computing efforts and serve as powerful tools in their own right, enabling the study of quantum systems in a controlled environment.

A practical quantum simulator should have controllability, scalability, and interactability, aligning with DiVincenzo's criteria [DiVincenzo, 2000] for a viable quantum computer. Controllability means efficiently initializing, manipulating, and reading qubit states, mirroring with DiVincenzo's guidelines for initialization and readout. Scalability involves linearly increasing resources to handle more qubits, reflecting the requirement of physical scalability in DiVincenzo's principles. Lastly, interactability and controllability also meet DiVincenzo's standards for a universal set of quantum gates and gate operations faster than qubits lose coherence (decoherence time). This convergence of features in quantum simulators represents a significant stride toward the broader objective of realizing full-scale quantum computers.

2.2 Controllability: Addressing an individual quantum system

Quantum information is generally encoded in the energy eigenstates of the quantum simulator. Controllability refers to the ability to initialize a quantum state accurately, precisely manipulate the state via defined control sequences and measure the state with efficient read-out techniques.

Initialization of desired quantum state: In the initialization step, the quantum simulator is prepared in a well-defined starting state, such as a known ground state or

a specified superposition state. Reliable and efficient initialization methods are essential to set the quantum simulator at a suitable starting point for the desired simulation or computation. Techniques like quantum state tomography [Cramer et al., 2010] and state preparation algorithms [Araujo et al., 2021] guide us to achieve the selected quantum states accurately.

Manipulation of quantum state: Quantum state manipulation is accomplished through well-defined control pulses that control the qubit's physical variables, such as energy levels, coupling strengths, and interaction times. Precisely controlling these variables allows the quantum system to transition from one desired state to another, simulating specific physical systems or implementing targeted quantum operations [Cirac and Zoller, 1995]. The ability to manipulate quantum states with high fidelity is crucial for executing quantum algorithms and conducting quantum information processing tasks.

Measurement of quantum state: After the quantum system undergoes evolution or computation, efficient and accurate read-out techniques are required to extract information about the final state. Measurement techniques in quantum simulators involve mapping the quantum state onto classical measurement outcomes that can be recorded and analyzed. Accurate measurement techniques are essential for validating simulation outcomes, assessing the effectiveness of quantum algorithms, and extracting valuable insights from the simulated system.

The controllability of quantum systems is a fundamental theoretical notion in quantum control and has practical importance because of its close connection with the universality of quantum computation [Dong and Petersen, 2010]. These aspects collectively enable efficient simulation, computation, and analysis of complex quantum systems using quantum simulators.

2.3 Scalability: Controlling multiple quantum systems with linear increase in resources

Scalability refers to a linear increase in resources to control multiple quantum systems and manage their interconnectivity to interact. Key factors contributing to scalability include the hardware's control infrastructure and connectivity of the qubits.

Scalable control infrastructure: Scalability necessitates a linear increase in resources to initialize, manipulate and measure qubits. An ideal infrastructure must be able to individually address and control many qubits simultaneously with minimal interference

and high fidelity. For instance, a study demonstrated the parallel execution of gates to create entanglement [Levine et al., 2019]. Furthermore, measurement systems should be capable of handling many qubits, mitigating any noise that could compromise accuracy.

Connectivity of the hardware: As the number of qubits grows, the simulator can model more realistic physical systems. Such systems require quantum simulators to have high connectivity between qubits. The ability to couple qubits and create entanglement between them is crucial for executing deep quantum circuits. These circuits enable the exploration of quantum dynamics with long-range quantum entanglement.

Further advancements in hardware technology and error correction techniques are essential to improve scalability. Scalability enhances the utility of quantum simulators in diverse areas such as material science, drug discovery, and optimization problems by enabling larger parameter spaces.

2.4 Interactability: Facilitating strong tunable interactions between individual quantum systems

Interactability refers to the ability to couple two qubits with strong tunable interactions, which can be switched as required. Two key requirements for tunable interactions are customizable Hamiltonians and the digital-analog programmability of simulators.

Customizable Hamiltonians: Tunable interaction plays a role in dynamically modulating the coupling strengths of qubits, turning on or off the exchange of quantum information. Such a feature can only be achieved by modulating the system's Hamiltonian, which describes the energy levels and interactions of the quantum system. The coupling parameters of the Hamiltonian can be adjusted through various techniques such as changing the distance between qubits [Bluvstein et al., 2022], modifying the strengths of the coupling fields [Kim et al., 2009], or altering the properties of the coupling medium [Majer et al., 2007]. By making these interactions tunable, the effective Hamiltonian of the system can be tailored to match the physical Hamiltonian of the system under study.

Digital-Analog programmability: Digital quantum or gate-based quantum simulation offers versatility as it can encode any Hamiltonian using one and two-qubit gates. However, achieving the high coherence, gate fidelity, and error correction required for deep quantum circuits is challenging. Conversely, a model closer to digital quantum computers incorporates analog-like elements, allowing for the activation and deactivation of multi-qubit interactions rather than decomposing them into single and two-qubit gates. This

approach can lead to shorter-depth circuits and the simulation of more complex problems, even with limited coherence and gate fidelity. Hybrid models of simulation bridge the gap between digital and analog approaches. Analog devices replicate the target Hamiltonian by mapping it to the simulator's customizable Hamiltonian. Additional capabilities, such as single-qubit gates and single-site addressability, offer increased control and expand the range of Hamiltonians that can be simulated.

Interactability in quantum simulators grants the ability to control and manipulate the interactions between qubits by customizing the system's Hamiltonian. Such customizability facilitates adaptable quantum simulations enabling the study of fundamental physics and the development of new quantum algorithms.

2.5 Neutral Atom Arrays: Potential platform for quantum simulation

Neutral atoms containing a single valence electron can be excited to high-energy states, resulting in atoms known as Rydberg atoms [Gallagher, 1988]. Due to their unique electronic configuration, Rydberg atoms possess exceptionally large electric dipole moments compared to ground state atoms, facilitating strong interactions. Importantly, these interactions can be controlled with external electromagnetic fields like lasers or microwave fields. This high controllability makes neutral atom systems ideal for constructing quantum many-body simulators with tunable parameters [Wu et al., 2021].

Quantum simulators with neutral atoms present a distinct pathway for achieving scalable quantum simulation and information processing. These systems operate within ultra-high vacuum chambers and use lasers, microwaves, and magnetic fields, to exert precise control over atom positions and their quantum state. Atoms are trapped into optical tweezers created by tightly focused lasers, generating trapping potentials at desired locations. Optical tweezers, ranging from tens to hundreds in 2D [Barredo et al., 2016] and 3D [Barredo et al., 2018], have been demonstrated in the last decade, and further scalability can be achieved by augmenting the laser power.

Once desired geometry is initialized, exciting atoms to the Rydberg state introduce strong tunable Rydberg interactions [Browaeys et al., 2016]. These interactions give rise to a diverse quantum spin model encompassing various quantum phases, each arising from the interplay between interactions and coherent driving [Bernien et al., 2017] [Ebadi et al., 2021]. Moreover, these interactions enable the implementation of various quantum information and entanglement generation protocols [Saffman et al., 2010].

Significant advancements in experimental techniques involving neutral atom-based systems have greatly contributed to the understanding and utilization of Rydberg atoms. Over the past few decades, impressive progress has been made in various areas, including the preparation of ultra-cold atomic gases [Phillips, 1998], high-resolution imaging of single atoms[Sherson et al., 2010], and the trapping of individual atoms using reconfigurable optical tweezer arrays [Endres et al., 2016] [Barredo et al., 2016]. These experimental achievements have unveiled the captivating features and potential of highly excited Rydberg states, establishing them as one of the most prominent platforms for quantum information processing based on neutral atoms. This section examines how neutral atom-based quantum simulators realize the features mentioned above.

2.5.1 Controllability: Laser pulse driven atomic transitions

Any quantum system with multiple distinguishable states can encode quantum information. Neutral atoms, like trapped ions, offer a variety of species and quantum states, presenting numerous options for physical qubits with distinct internal quantum properties. The choice of specific atoms and quantum states is primarily determined by balancing well-isolated states with longer coherence times and readily accessible quantum levels that facilitate initialization, manipulation, and detection. Recent developments in laser cooling and optical and magnetic trapping techniques have led to focused experiments exploring heavy alkali atoms, such as Rb (rubidium) and Cs (cesium).

In our lab, we plan on encoding quantum information in the hyper-fine ground state of Rb-87 atoms due to their long coherence times. Rydberg properties (discussed in Section 2.5.3) are realized by exciting the atoms to $n = 70$ with a two-photon excitation mechanism using 420 nm and 1013 nm lasers as shown in Fig. 2.1. State initialization in the ground state manifold is achieved by optically pumping the atoms to the ground state. To prepare an arbitrary known state, additional steps for coherent population transfer are required. Single qubit operations are performed by two focused laser beams using a three-level Λ-type Raman scheme, where coherent transfer between hyperfine ground state $|g_0\rangle$ and $|g_1\rangle$ is mediated by an excited intermediate state $|e\rangle$. Two-qubit entangling operations are mediated through Rydberg interactions and can be actuated in parallel with fidelity as high as 0.995[Evered et al., 2023]. State-readout is achieved by fluorescence imaging of atoms using a 780 nm laser. This destructive measurement technique can achieve a high fidelity of 0.9997 [Nelson et al., 2007].

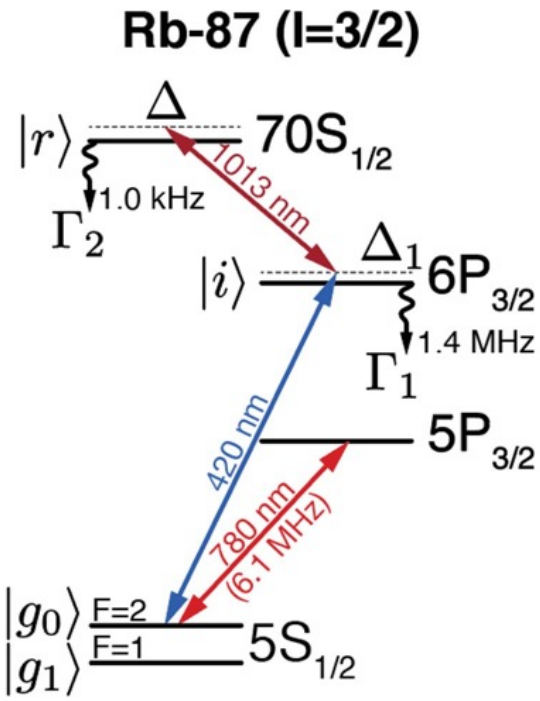


Figure 2.1: **Relevant Rb-87 atomic levels for quantum information processing.** The quantum information is encoded in the hyper-fine ground state of Rb-87 atoms due to their long coherence time. Rydberg properties are realized by exciting the atoms to $n = 70$ with a two-photon excitation mechanism using 420 nm and 1013 nm lasers. State-readout is achieved by fluorescence imaging of atoms using a 780 nm laser.

2.5.2 Scalability: Parallel control over an array of neutral atoms

Because neutral atoms are indistinguishable, the requirements for physical resources, such as laser frequencies, do not significantly increase with the scaling up of qubits. This inherent identical property simplifies the experimental setup and reduces the complexity of implementing larger-scale qubit systems. Weak magnetic dipole-dipole and Van der Waals (vdW) interactions between the ground state atoms enable precise trapping of large numbers of atoms in various configurations of optical tweezer trap arrays [Barredo et al., 2018] or magnetic trap arrays [Wang et al., 2016].

The optical tweezer platform offers advantages such as rapid experimental cycle times and relative experimental simplicity. One of the most significant challenges in neutral atom-

based quantum computing platforms arises from the stochastic nature of atom loading into individual traps. This issue is primarily caused by operating in a 'collisional blockade' regime, which imposes limitations on the probability of loading a single atom into a small-volume trap site, resulting in an efficiency of approximately 50% [Schlosser et al., 2001]. Defect-free atomic array requires atom rearrangement, using dynamic optical tweezers to fill incomplete traps by moving atoms [Endres et al., 2016][Barredo et al., 2016].

In our lab, we plan to create static optical tweezers using Spatial Light Modulator (SLM) as shown in Fig. 2.2b and dynamic optical tweezers using two crossed Acousto-optic deflectors (AODs) to rearrange Rb-87 atoms in two dimensions. Fig. 2.3 depicts dynamic trap arrays generated by two crossed AODs. We also plan to employ in-house developed reconfiguration algorithms [Cimring et al., 2022][Sabeh et al., 2022] to rearrange the atoms and create defect-free arrays. The Spatial Light Modulator (SLM) enables the creation of arbitrary optical tweezers' geometry. As depicted in Fig. 2.2c, from a single collimated Gaussian beam (Fig. 2.2a), a honeycomb lattice can be formed by applying a phase mask using SLM. This allows individual addressing of atomic qubits, enabling parallel control and manipulation.

(a) (b) (c)

Figure 2.2: **Generating arbitrary geometry of optical tweezers with diffractive optical elements.**
a. A collimated Gaussian beam which is directed onto a Spatial Light modulator (SLM). **b.** A Spatial Light Modulator (SLM) which imparts a pre-calculated phase mask on the input beam to generate the desired geometry of optical tweezers. **c.** A honeycomb lattice of optical tweezers created using SLM.

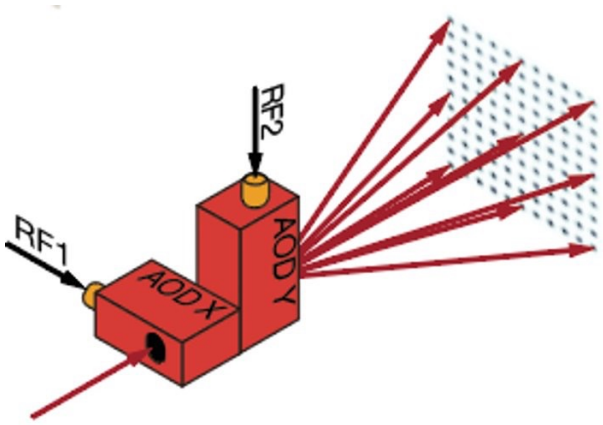
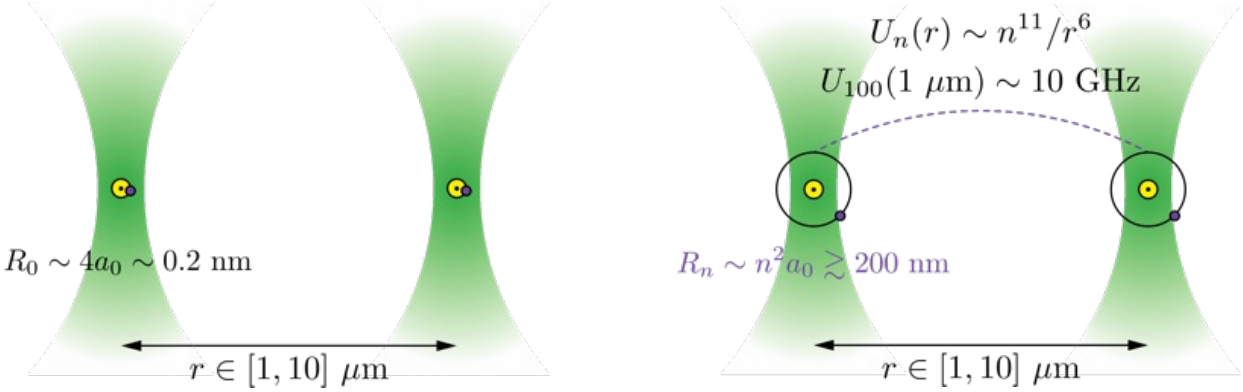


Figure 2.3: **Multiplexed beams for parallel execution.**
Multiplexed beams created by two-crossed AODs act as optical tweezers trapping single atoms. Static optical tweezers are employed for trapping atoms, while dynamic optical tweezers enable the movement of atoms in a Manhattan geometry. Image source: [Cooper et al., 2018].

2.5.3 Interactability: Engineering strong interactions between two atoms via Rydberg blockade mechanism

Neutral atoms can be excited to a high principal quantum number n leading to Rydberg interactions. The dipole moments of Rydberg atoms scale as n^2, resulting in significantly large dipole-dipole interactions that increase rapidly with increasing principal quantum number, n. By selecting appropriate Rydberg states, it is possible to control various aspects of the interaction, including its strength, sign, anisotropy, and spatial dependence. Additionally, the interaction can be switched off by transferring the atoms back to their ground state, providing control over the system as shown in Fig. 2.4 [Pritchard et al., 2010]. These strong interactions among Rydberg atoms lead to a phenomenon called Rydberg blockade. Only one atom can be excited from the ground state to a Rydberg state within a specific volume, called Rydberg blockade radius 2.5a. This is because the first excited Rydberg atom causes a shift in the Rydberg energy levels of all other nearby atoms, taking them off-resonance as shown in Fig. 2.5b. The Rydberg blockade phenomenon enables conditional dynamics that are highly desirable for quantum information processing.



(a) (b)

Figure 2.4: **Atoms exhibiting Rydberg properties when excited to a high energy state.**
a. Ground state configuration of atoms with minimal interactions. **b.** Rydberg state of atoms, where a single electron is excited to a high energy level with principal quantum number n. The strong interactions between atoms in the Rydberg state arise due to the dipole moments scaling as n^2.

(a)

(b) Energy level shift if two atoms are within blockade radius

Figure 2.5: **Rydberg blockade mechanism.**
a. A large ensemble of atoms arranged in a regular square array with spacing a and a Rydberg blockade radius R_b. **b.** Within the Rydberg blockade radius, only one atom can be excited to the Rydberg state due to energy level shift. Image source: [Browaeys and Lahaye, 2020]

Chapter 3

Experimental methods for quantum simulation with neutral atom arrays

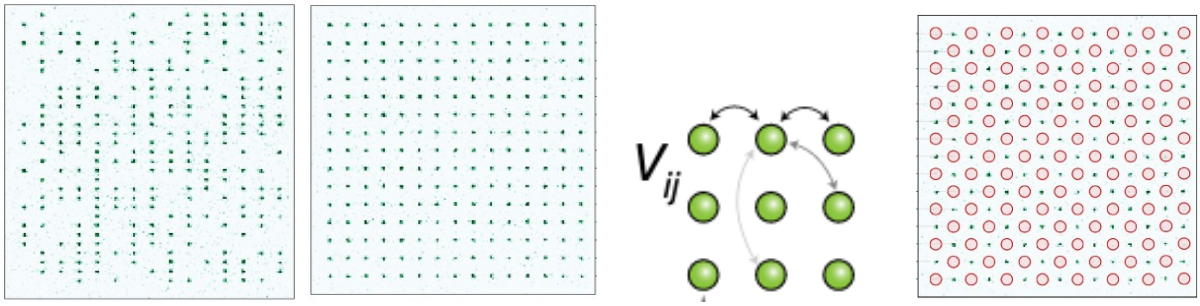
3.1 Experimental Workflow: Load, rearrange, evolve, and readout neutral atom arrays

The experimental workflow for quantum simulation with neutral atom arrays can be broken down into four steps, namely loading atoms in optical traps, rearranging atoms for defect-free geometry, letting the system evolve under targeted dynamics, and reading out the resulting state as shown in Fig. 3.1.

Loading atoms in arbitrary geometries of optical traps: The process of quantum simulation begins by creating a dilute atomic vapor within an ultra-high vacuum system operating at room temperature. Laser cooling and trapping techniques are utilized to prepare a cold ensemble of approximately 10^6 atoms within a 3D magneto-optical trap (3D MOT) [Metcalf and van der Straten, 2003]. The resulting volume of such an atomic cloud is approximately 1 mm^3.

Subsequently, a second trapping laser system isolates individual atoms within the ensemble. High numerical aperture lenses strongly focus the trapping beam into multiple spots called optical tweezers [Schlosser et al., 2001] with a diameter of around 1 μm. Each optical tweezer can hold at most one atom at a time within a trapping volume of a few μm^3. The number and arrangement of these tweezers can be customized to form arbitrary 1D, 2D, or 3D geometries using holographic methods [Nogrette et al., 2014]. Before passing through the focusing lens, the trapping beam is directed onto a spatial light modulator

(SLM), which applies an adjustable phase pattern to the light. In the focal plane of the lens, the phase modulation is converted into a desired intensity pattern. The number of optical tweezers is only constrained by the available trapping laser power and the performance of the optical system that generates the optical tweezers.



(a) (b) (c) (d)

Figure 3.1: **Experimental workflow for quantum simulation with neutral atom arrays.**
a. Atoms are loaded in arbitrary geometries of optical traps, with a 50% loading efficiency due to stochastic processes. **b.** Atom rearrangement is performed to achieve defect-free geometries. **c.** The system evolves under targeted dynamics once the desired geometry is attained. **d.** The simulation outcome is calculated by reading-out atoms via destructive measurements. Image source: [Ebadi et al., 2021]

Rearranging atoms for defect-free geometries: Although each tweezer can accommodate at most one atom, in approximately 50% of the cases, the tweezer remains empty due to stochastic loading, as shown in Fig. 3.1a. As shown in Fig. 3.1b, atoms are moved from one site to another using programmable moving optical tweezers to create compact geometries. Once atomic qubits are loaded at the desired location, control sequences are played on individual atoms to execute quantum circuits or the system is allowed to evolve under its natural Hamiltonian to simulate the dynamics of a target system as shown in Fig. 3.1c. The processing is highly efficient, occurring in less than 100 μs, while the overall sequence, including loading and readout, takes approximately 200 ms [Henriet et al., 2020].

Evolving the system under targeted dynamics: Analog and digital quantum simulations are possible with neutral atom arrays [Henriet et al., 2020]. In the case of digital quantum computing, quantum algorithms are decomposed into a series of quantum logic gates, forming a quantum circuit. These gates are implemented by applying well-defined laser pulses to individual atomic qubits. On the other hand, analog computing

involves the realization of a Hamiltonian using control pulses (here realized using laser and microwave pulses). The commonly studied natural Hamiltonian for neutral atom arrays involves atoms excited to the Rydberg state, expressed by Equation 3.1, where $\hat{\sigma_j^x}$ denotes the Pauli $\hat{\sigma^x}$ matrix on the j^{th} qubit, and $\hat{n_j}$ represents the Rydberg state occupancy. The control parameters $\Omega(t)$ and $\delta(t)$ correspond to the Rabi frequency and detuning, respectively, and can be adjusted by modifying the laser field's intensity and frequency. The third term represents the energy penalty when two qubits are simultaneously in the Rydberg states. The interaction between qubits i and j follows a van der Waals-type coupling that depends on the inverse sixth power of the distance r_{ij} between them and a coefficient C_6 associated with the Rydberg state. The qubits then evolve according to the Schrödinger equation.

$$\mathcal{H}(t) = \frac{\hbar}{2}\Omega(t)\sum_j \hat{\sigma_j^x} - \hbar\delta(t)\sum_j \hat{n_j} + \sum_{i\neq j}\frac{C_6}{r_{ij}^6}\hat{n_i}\hat{n_j} \tag{3.1}$$

Reading-out atoms via destructive measurements: After the evolving stage, the atomic qubits are read out by capturing a final fluorescence image. The readout process is designed such that each atom in the qubit state $|0\rangle$ appears bright, while atoms in the qubit state $|1\rangle$ remain dark, as depicted in Fig. 3.1d. Multiple such computation cycles are repeated to gather sufficient data for reconstructing the relevant statistical properties of the final quantum state.

3.2 Experimental Architecture: Systems and hardware for neutral atom arrays

The experimental setup of our neutral atom-based quantum simulator can be divided into six systems, namely the vacuum and cooling system, trapping system, imaging system, reconfiguration system, rydberg system, and control and acquisition system, as shown in Fig. 3.2.

The vacuum and cooling system creates a 3DMOT, making an atomic cloud of Rb-87 atoms. The trapping system generates a desired configuration of optical tweezers using SLM or AODs to trap individual atoms. The imaging system acquires images using an electron-multiplying charge-coupled-device (EMCCD) camera, which converts fluorescence photons from atoms into measurable electronic signals. As the loading efficiency of atoms in traps is non-unity, a reconfiguration system moves atoms with dynamic optical tweezers

to create a desired compact geometry of atoms. On achieving the desired geometry for quantum simulations, the rydberg system shines a laser to excite atoms to the Rydberg state, complying with a pre-defined control sequence on selected qubits, and the imaging system reads out the resultant states. The control and acquisition system coordinates with all the systems by composing, executing, and imaging defined experimental sequences.

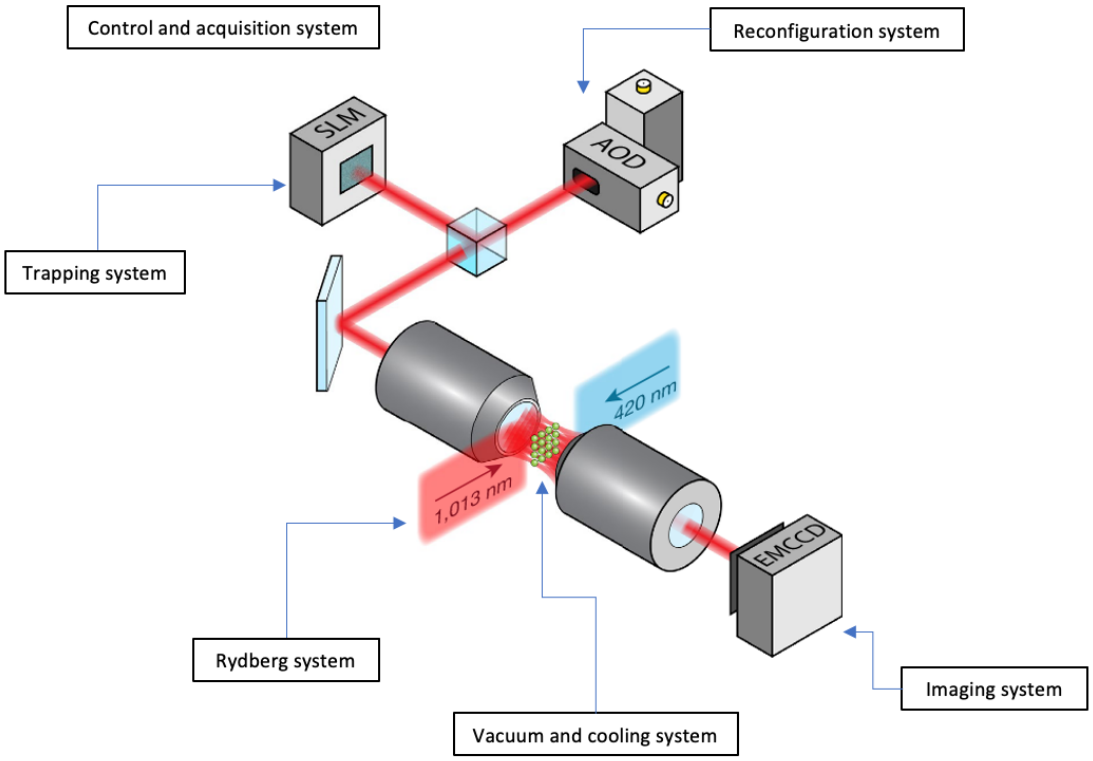
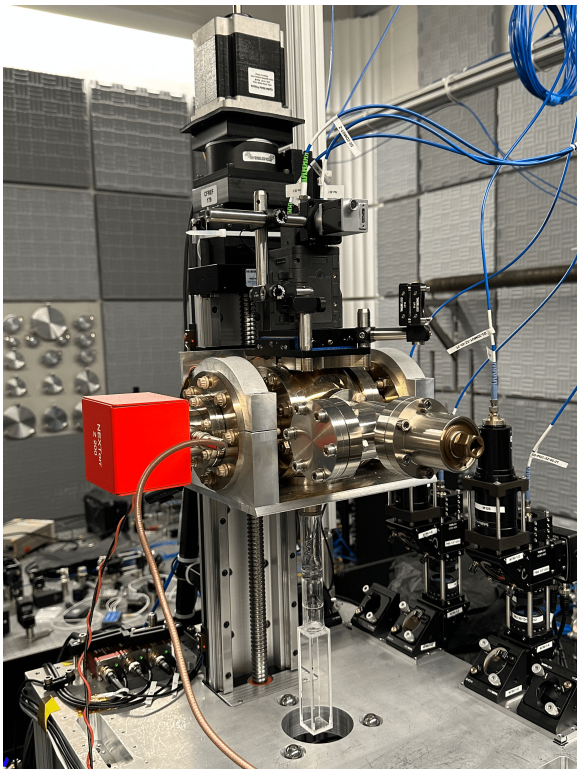


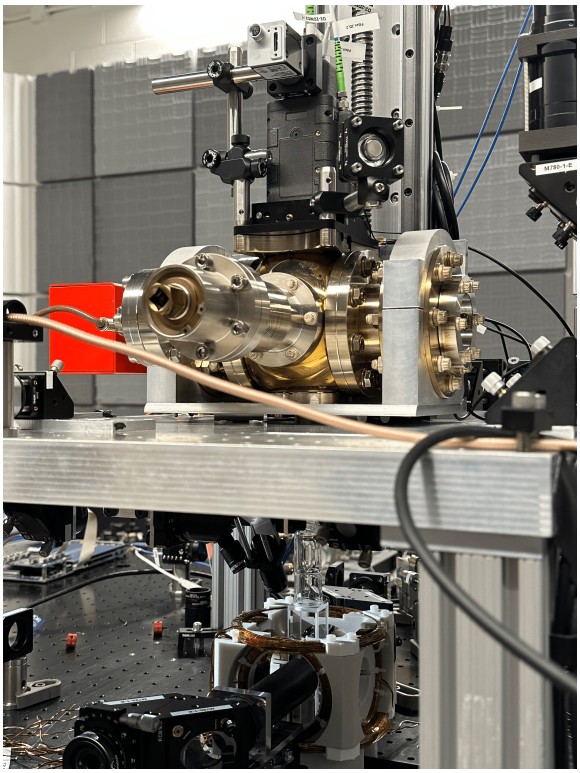
Figure 3.2: **Systems in a neutral atom-based quantum simulator experimental setup.**
The illustration showcases the six key systems: vacuum and cooling, trapping, imaging, reconfiguration, rydberg, and control and acquisition systems. Image source: [Ebadi et al., 2021]

3.2.1 Vacuum and cooling system

A vacuum chamber is at the vacuum system's core, which serves as a controlled environment with extremely low pressure. A current-modulated Rb-87 atomic source cell, which releases Rb-87 atoms into the vacuum chamber, is positioned on top of the chamber. The bottom of the chamber is connected to a glass cell, the heart of any neutral atom-based quantum simulator. Fig. 3.3 shows the complete assembly constructed in our laboratory. The assembly is mounted on a linear translation stage to make the setup modular. The glass cell is descended during experiments and moved up to set up optics and install MOT coils without damaging the glass cell.



(a) Atomic source, vacuum chamber, and glass cell assembly



(b) Assembly descended to perform experiments within glass cell

Figure 3.3: **Modular vacuum chamber assembly on a linear translation stage.** Mounting the assembly on a linear translation stage makes the experimental setup modular. The assembly is descended during experiments and moved up to set up optics and install MOT coils without damaging the glass cell.

The Rb-87 atoms dispensed from the atomic source cell are trapped within a 3D Magneto-optical trap (MOT). This trap consists of cooling and repumping beams that drive cooling transitions in the atoms, leading to energy loss and temperature reduction. The relevant cooling and repumping transitions for Rb-87 atoms are shown in Fig. 3.4. The frequencies of cooling and repumping beams are swept to fine-tune the atom-capturing process and maximize the density of trapped atoms.

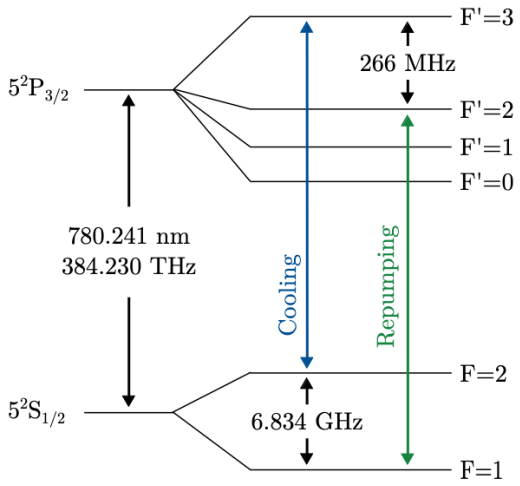


Figure 3.4: **Rb-87 D_2 transition for cooling atoms.**
Image source: [Wiegand et al., 2019]

The vacuum and cooling system creates a controlled environment where Rb-87 atoms can be cooled and trapped, setting the stage for subsequent quantum simulation.

3.2.2 Trapping system

The trapping system allows for the precise confinement of individual atoms. The system utilizes optical tweezers, created using diffractive optical elements such as Spatial Light Modulator (SLM) or Acousto-Optic Deflector (AOD) as shown in Fig. 2.2b and Fig. 3.6, respectively. These optical tweezers are capable of trapping at most one atom because the system operates in the collision blockade regime. The process of loading atoms into the optical tweezers is stochastic, resulting in a loading rate of approximately 60%. However, the loading rate can be increased up to 90% by techniques like Λ-enhanced grey molasses

cooling [Rosi et al., 2018]. Molasses involves further cooling the atoms in the 3D Magneto-optical trap (MOT), effectively increasing the efficiency of loading single atoms into the optical tweezers.

The atoms confined within the 3DMOT are transferred into optical tweezers created by trapping laser beams. The 3DMOT beams are switched off when the trapping laser beams are activated to load a single atom in each optical tweezer. The trapping system allows the creation of arbitrary geometries of optical tweezers, including 1D, 2D, and 3D configurations. This flexibility enables the arrangement of atoms in desired spatial patterns tailored to the specific requirements of the quantum simulation. As an example, Fig. 3.5 demonstrates the capability of the trapping system to create optical tweezers in arbitrary shapes. In this case, the face of my lab mates is detected using an edge-detection algorithm. A numerical technique called the Gerchberg-Saxton algorithm generates an associated phase profile to create traps in the shape of the detected face. The phase profile is fed to the SLM, which embeds the phase on the input beam.

(a) Alex (b) Anastasiia (c) Artem

(d) Kent (e) Parth

Figure 3.5: **2D Arbitrary geometries of optical tweezers created by SLM.**

The trapping system's ability to generate optical tweezers and trap single atoms in desired geometries provides the necessary control to address individual atoms and their positions.

3.2.3 Imaging system

The imaging system comprises multiple cameras that capture images of the optical tweezers and the fluorescence signals emitted by the atoms. The primary goal of imaging the optical tweezers is to achieve uniform optical tweezer intensities, leading to uniform trap depths. A closed-loop optimization algorithm ensures uniform intensity across the optical tweezers. This algorithm involves capturing images of the optical tweezers using a CMOS camera in a feedback loop. Based on these images, it calculates new phase profiles to uniformize the trap depths and updates them on the Spatial Light Modulator (SLM). By achieving uniformity, the algorithm minimizes the likelihood of atoms being attracted to deeper traps, resulting in a more uniform trapping environment.

We employ an Electron-Multiplying Charge-Coupled-Device (EMCCD) camera to capture the fluorescence emitted by the atoms. This fluorescence is induced by shining imaging light onto the atoms, specifically targeting the 780 nm transition from the ground state $|g_0\rangle$ to the $5P_{3/2}$ state, as depicted in Fig. 2.1. The EMCCD camera efficiently converts fluorescence photons into measurable electronic signals. Detection efficiencies exceeding 98.6% have been achieved in previous studies [Fuhrmanek et al., 2011]. The captured images provide valuable information regarding the number of traps filled with atoms and empty traps, enabling assessment of the loading process's success. Additionally, this data provides actionable insights to reconfigure atom positions for defect-free geometries.

The imaging system enables the imaging of atoms, which is essential for creating defect-free geometries and for read-out of the atomic qubit state. Moreover, the imaging system facilitates the imaging of optical tweezers, which is instrumental in achieving uniform trap depths.

3.2.4 Reconfiguration system

The reconfiguration system utilizes dynamic optical tweezers to create defect-free geometries of atoms. Two multiplexed Acousto-Optic Deflectors (AODs) in the X and Y directions create dynamic optical tweezers in 2D, enabling atom moves in a Manhattan geometry. These dynamic optical tweezers are designed to be deeper than the static tweezers, allowing atoms to migrate from the static traps to the dynamic traps, as shown in Fig. 3.6.

An algorithm analyzes EMCCD images in real-time and computes a series of moves for the dynamic tweezers, reconfiguring the initial atom arrangement into the desired fully assembled geometry.

The atom reconfiguration process is iterative, designed for speed and efficiency to maximize the quantum simulation time within the atoms' trap lifetime. The images captured by the EMCCD camera are transferred to a computer via a Frame Grabber Card (FGC), facilitating the reconfiguration process. A real-time analysis algorithm parses the image and determines a sequence of atom moves to create defect-free geometries. The defined atom moves are relayed to an Arbitrary Waveform Generator (AWG), which controls the Acousto-Optic Deflectors (AODs). The entire reconfiguration process forms a closed-loop optimization system: it encompasses consecutive stages of image capture, image analysis, waveform creation, and waveform streaming. This iterative mechanism ensures the generation of defect-free atom geometries.

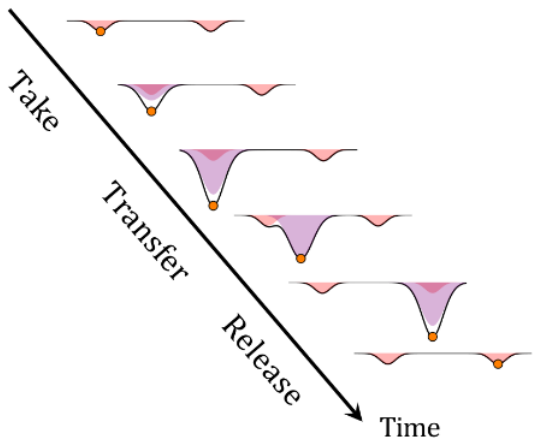


Figure 3.6: **Dynamic optical tweezers moving an atom from one trap to another.**
Dynamic optical tweezers are designed to have deeper trap depths than the static tweezers, allowing atoms to migrate from the static traps to the dynamic traps. Image source: [Henriet et al., 2020]

The reconfiguration system uses dynamic optical tweezers to move atoms and create arbitrary defect-free geometries of atoms. It employs low-latency reconfiguration algorithms [Cimring et al., 2022] [Sabeh et al., 2022] to maximize the quantum simulation time within the atoms' trap lifetime.

3.2.5 Rydberg system

The Rydberg system facilitates the excitation of atoms to the Rydberg state. A straightforward method for this excitation involves single-photon transition, offering direct laser excitation. However, dipole selection rules restrict this approach to couple ground S states and Rydberg P states. This transition requires ultraviolet range wavelengths (e.g., 297 nm for Rb-87), which present several challenges, including material degradation, limited availability of optical fibers, and low-loss optics. Moreover, Rydberg P states are characterized by a higher degree of structural complexity, anisotropy, and susceptibility to external perturbations, complicating coherent manipulation [de Léséleuc et al., 2018]. An alternative approach is the two-photon laser excitation, which can couple ground S states with Rydberg S states via an intermediate P state.

In our laboratory, we use two lasers to induce the two-photon transition required for exciting the atoms to the Rydberg state. The first laser operates at a wavelength of 420 nm, corresponding to the transition from the ground state ($5S_{1/2}$) to the intermediate state ($6P_{3/2}$). The second laser operates at a wavelength of 1013 nm, corresponding to the transition from the intermediate state ($6P_{3/2}$) to the Rydberg state ($70S_{1/2}$), as illustrated in Fig. 2.1. By designing optimal control laser pulses, the atoms are excited to the Rydberg state within a finite time, minimizing the spontaneous emission and reducing decoherence effects. Depending on the needs of the simulation, the entire atom array or only individual atoms are excited to the Rydberg state. For site-selective excitation, the addressing laser beams employ a strategy analogous to the trapping system, enabling excitation of the targeted atoms to the Rydberg state.

The Rydberg system enables the coherent excitation of atoms to the Rydberg state using a two-photon excitation mechanism.

3.2.6 Control and acquisition system

The control and acquisition system serves as the brain of the experiment enabling the coordination of heterogeneous hardware in the laboratory. It provides the ability to compose and execute experiments using two analog modules and one digital module for generating control sequences required for the experiment. Analog signals modulate parameters like current, light intensity, magnetic field strengths, and Acousto-optic modulator (AOM) frequencies. Digital signals control mechanical shutters and microwave switches and provide TTL signals for camera triggers.

The digital and analog modules are interfaced with Labscript suite, a comprehensive

software framework designed for experiment composition, execution, and analysis [P.T. Starkey, 2019]. M-LOOP, a machine learning library, is integrated with the Labscript suite automating the decision-making process to navigate the control parameter space, thus accelerating parameter optimization [Wigley et al., 2016]. By leveraging these tools, we can quickly find the optimal control settings for our experiments.

The control and acquisition system enables the creation of experimental sequences, conducts the experiments, acquires and stores images during the process, and incorporates analysis to form a closed-loop procedure for optimizing control parameters.

Chapter 4

Increasing the simulation power of neutral atom arrays with dynamic architecture

Quantum simulators can be categorized as digital or analog, each with distinct characteristics. Digital quantum simulators use quantum gates to create a series of operations that simulate any quantum phenomena. However, this versatility comes with the potential for errors and computational overhead. Conversely, analog quantum simulators exploit the natural evolution of a controlled quantum system to mimic target quantum dynamics, providing meaningful solutions to specific problems.

In digital quantum simulations, qubit connectivity is crucial. Due to physical constraints, systems with partial connectivity can't directly achieve specific qubit-to-qubit interactions required for particular computations. Consequently, auxiliary operations such as SWAP gates are necessary to facilitate gate operations between non-adjacent qubits. Advanced compilation algorithms are needed to minimize these additional gate operations, adding complexity in executing the computation on the quantum simulator. Additionally, these auxiliary operations introduce opportunities for errors to propagate in the current NISQ era.

On the contrary, all-to-all connectivity allows direct interaction between any pair of qubits, simplifying quantum circuit execution without intermediary SWAP operations. This enhanced connectivity reduces errors and increases the depth of successfully implemented circuits.

We introduce a dynamic architecture approach to tailor the connectivity between qubits

on the neutral atom array platform. Dynamic architecture provides on-demand connectivity by moving atoms during the simulation, leading to a trade-off between SWAP operations and atom movements. We evaluate the capabilities of static and dynamic architecture by computing and comparing the circuit fidelity and Quantum Volume for each approach.

In this chapter, Section 4.1 introduces static and dynamic architecture, while Section 4.2 highlights the recent advancements in neutral atom platforms enabling dynamic architecture. Section 4.3 outlines the atom displacement protocol embodying dynamic architecture, and the associated errors of the protocol are discussed in Section 4.4. Section 4.5 presents the circuit compilation protocol for both architectures. Section 4.6 compares the circuit fidelity of both architectures and analytically examines scenarios where dynamic architecture outperforms static one by evaluating error bounds. In conclusion, Section 4.7 contrasts the trend in allowed maximum error to attain particular quantum volume in both architectures.

4.1 Architecture: Topology of underlying qubits

We can use the analogy of x86 and ARM processors within classical computing to draw a parallel between different quantum simulator architectures. Both x86 and ARM architectures serve the same fundamental purpose - to decode and execute instructions based on their respective Instruction Set Architecture (ISA). Despite their shared purpose, the methodologies by which they break down tasks into executable instructions lead to varying levels of efficiency.

The x86 architecture, primarily used in personal computers, is designed around Complex Instruction Set Computer (CISC) architecture. It provides powerful processing capabilities and a wide range of features. However, these benefits come at the cost of increased power consumption. On the contrary, the ARM architecture operates based on a Reduced Instruction Set Computer (RISC) architecture, resulting in more efficient power usage. Its efficiency and straightforward design allow for better control at the fundamental level, making ARM processors ideally suited for mobile and embedded devices. Due to its efficiency, ARM processors have recently paved their way into high-performance computing and personal computing.

This analogy extends into the quantum simulator domain, contrasting static and dynamic architectures. Both architectures share a common objective: executing quantum algorithms and performing quantum simulations. However, similar to the x86 and ARM architectures in classical computing, the underlying execution mechanisms of static and

dynamic architectures differ, resulting in variations in accuracy and efficiency. Static architecture, similar to the x86, is resource-intensive and carries an operational overhead for circuit compilation, which is unsuitable for noisy quantum computations and simulations. Conversely, dynamic architecture, reminiscent of the ARM, allows on-the-fly qubit displacements to enhance simulation efficiency. However, this flexibility may introduce additional complexities in control and the potential for errors.

In this section, we formalize two types of architecture in quantum simulators: static and dynamic. We first introduce coupling and connectivity graphs for a quantum simulator to classify these architectures.

4.1.1 Coupling graph: Graphical representation of physical interactions

A quantum simulator consists of an assembly of individual quantum systems. The interactions between these systems stem from various physical forces, including Coulomb interactions and dipole-dipole interactions, or are mediated through a shared bosonic mode. These interactions form the foundation of coupling between two quantum systems, which encode qubits. The coupling graph is a representative diagram for a quantum simulator as a weighted, undirected graph. Each vertex of the coupling graph corresponds to a qubit, with edges between vertices indicating the variable coupling strengths between respective qubit pairs.

Definition 1. ***Coupling graph****: Each quantum simulator has an associated coupling graph denoted as* $\boldsymbol{Q} = (V, E, w)$*, where:*

- $\boldsymbol{V} = x_1, x_2, \ldots, x_n, n \in \mathbb{N}$*, is a finite set of graph vertices that represent the set of qubits.*
- $\boldsymbol{E} \subset V \times V$ *is a finite set of undirected edges that connect a subset of vertices, symbolizing the subset of coupled qubits.*
- $\boldsymbol{w}$: $E \to \mathbb{R}$ *is an edge weight function indicating the coupling strength between adjacent qubits.*

Implication 1. *Implications of definition 1*

- $|\boldsymbol{V}| = n$*, where* n *is the number of qubits in the simulator.*

- $|\boldsymbol{E}|$ = *total number of two-qubit couplings in the simulator.*
- *The degree of a vertex, $deg(\boldsymbol{x})$, for $\boldsymbol{x} \in \boldsymbol{V}$, implies the number of qubits to which the given qubit x is directly coupled. It can vary from 0 (completely uncoupled) to $n-1$ (maximally coupled).*
- $|\boldsymbol{E}| \leq \frac{n(n-1)}{2}$, *the total number of couplings is upper-bounded by the number of pairs of qubits.*

However, not all couplings possess the potential to execute two-qubit operations. As such, a truncated coupling graph can be considered. This graph is a sub-graph of the coupling graph and only considers couplings that are equal to or stronger than a specific minimum coupling strength.

Definition 2. ***Truncated coupling graph****: Let $\boldsymbol{Q} = (V, E, w)$ represent the coupling graph. The truncated coupling graph, $\boldsymbol{Q'} = (V, E')$, of a quantum simulator is defined as an unweighted subgraph of the original coupling graph $\boldsymbol{Q}$, where:*

- $\boldsymbol{V}$ *is the finite set of graph vertices corresponding to the set of qubits, identical to that in $\boldsymbol{Q}$.*
- $\boldsymbol{E'} = \{(x, y) \in \boldsymbol{E} | w(x, y) \geq w_{min}\}$ *is a finite set of undirected edges, signifying the subset of couplings that exceed the minimum coupling strength.*

Implication 2. *Implications of definition 2*

- $|\boldsymbol{V}| = n$, *where n is the number of qubits in the simulator. The set of vertices in $\boldsymbol{Q'}$ remains unchanged from $\boldsymbol{Q}$.*
- $|\boldsymbol{E'}|$ = *total number of two-qubit couplings in the simulator that satisfy the minimum coupling strength w_{min}.*
- $|\boldsymbol{E'}| \leq |\boldsymbol{E}| \leq \frac{n(n-1)}{2}$, *the total number of couplings is upper-bounded by the number of pairs of qubits.*
- *The degree of a vertex, $deg'(\boldsymbol{x})$, for $\boldsymbol{x} \in \boldsymbol{V}$, implies the number of qubits to which the given qubit x is coupled with coupling strength exceeding the minimum coupling strength. It can vary from 0 (completely uncoupled) to $n-1$ (maximally coupled).*
- $deg'(\boldsymbol{x}) \leq deg(\boldsymbol{x})$ *for all $\boldsymbol{x} \in \boldsymbol{V}$.*

The truncated coupling graph, **Q'**, is a filtered version of the original coupling graph, **Q**, preserving only the robust couplings. The threshold for this filtering is defined by the minimum coupling strength w_{min}. This graph eliminates the weaker couplings that might not be strong enough to execute two-qubit operations or might induce noise in the quantum simulator.

In superconducting qubits, the coupling between two qubits is achieved by connecting both qubits to an intermediate electrical coupling circuit. The strength of the coupling is related to the overlapping of their wavefunctions. Depending on the design and operation parameters, the coupling can be tuned dynamically during the operation, which makes it possible to execute two-qubit gates or to isolate qubits to prevent unwanted interactions [Wendin, 2017].

In trapped ion quantum simulators, the coupling between qubits is predominantly achieved through shared vibrational modes of the ion crystal. External laser pulses can modulate these motional modes to modify the coupling between each pair of qubits. These customizable interactions lead to a fully connected coupling graph [HAFFNER et al., 2008].

In neutral atom array quantum simulators, the interaction between two atoms can be tuned by exciting the atoms to high-energy Rydberg states. The nature of interactions between these Rydberg atoms is van der Waals interaction, characterized by Equation 4.1. The atomic species and the excited Rydberg state's principal quantum number determine the C_6 co-efficient, and R is the distance between two atoms.

$$U_{\text{vdW}} = C_6/R^6 \tag{4.1}$$

For example, consider a linear chain of 16 atoms, labeled from 1 to 16 (beginning from the left), positioned a few micrometers apart, as displayed in Fig. 4.1a. In this setup, the interaction strength between two qubits, i and j, will decay according to $1/|R_i - R_j|^6$. Here, R_i and R_j refer to the respective positions of qubits i and j. Fig. 4.1b illustrates this atom configuration's logarithmic scale coupling graph. Each vertex in this graph corresponds to an atomic qubit, and the width of the connecting edge signifies the strength of the interaction between them, thereby indicating the weight of the edge. However, not all physical interactions are potent enough to facilitate quantum information exchange, prompting us to establish a threshold. Only interactions equivalent to or stronger than the nearest neighbor interaction strength are considered, producing a truncated coupling graph depicted in Fig. 4.1c.

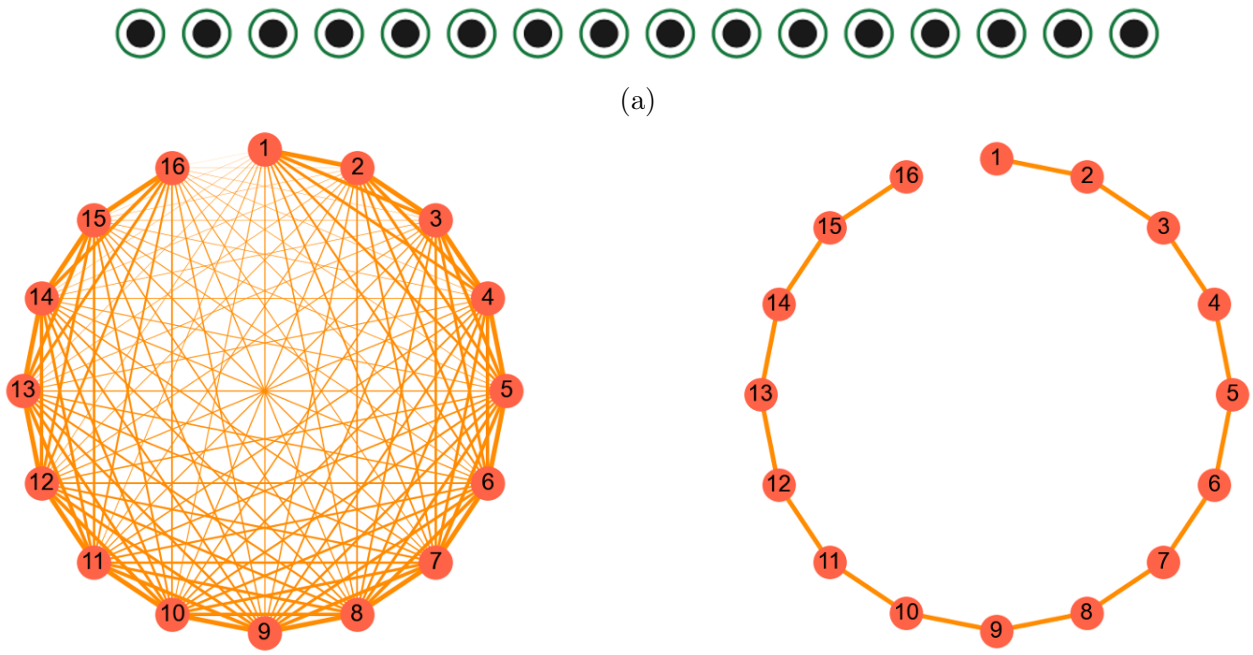


Figure 4.1: **Truncated coupling graph of a neutral atom array platform.** **a.** A 1D array of 16 atoms evenly spaced a few micrometers apart. **b.** The corresponding coupling graph for a neutral atom array simulator comprising a 1D linear chain of 16 atoms, each atom encoding a qubit. The interaction strength between the atoms decays proportional to $1/|R_i - R_j|^6$. For improved clarity, these interactions are represented on a logarithmic scale. Every vertex represents an atomic qubit, while the connecting edge's width signifies the strength of the interaction between them, indicating the edge's weight. **c.** The corresponding truncated coupling graph after thresholding the interaction strengths to the nearest neighbor interaction strength.

4.1.2 Connectivity graph: Graphical representation of qubit connectivity

A quantum simulator's scalability depends on the topology of the constituent qubits and the connectivity established amongst them. This connectivity underpins the potential interactions and facilitates the exchange of quantum information. The connectivity graph, an undirected graph, depicts the inter-qubit connectivity within a quantum simulator. Each vertex of the connectivity graph corresponds to a qubit, with edges between vertices

denoting the possibility of performing a two-qubit entangling operation on them in a single gate operation.

Definition 3. ***Connectivity Graph****: Each quantum simulator has an associated connectivity graph, denoted as* $\boldsymbol{C} = (V, E)$*, where:*

- $\boldsymbol{V} = x_1, x_2, \ldots, x_n, n \in \mathbb{N}$*, is a finite set of graph vertices that represent the set of qubits*
- $\boldsymbol{E} \subset V \times V$ *is a finite set of undirected edges that connect a subset of vertices, symbolizing that a two-qubit entangling operation can be performed on the associated qubits in a single gate operation.*

Implication 3. *Implications of definition 3*

- $|\boldsymbol{V}| = n$*, where* n *is the number of qubits in the simulator.*
- $|\boldsymbol{E}|$ *= total number of distinct two-qubit entangling operation that can be performed in a single gate operation.*
- *The degree of a vertex,* $deg(\boldsymbol{x})$*, for* $\boldsymbol{x} \in \boldsymbol{V}$*, implies the number of qubits to which the given qubit* x *can be entangled in a single gate operation. It can vary from* 0 *(isolated) to* $n - 1$ *(maximally connected).*
- $|\boldsymbol{E}| \leq \frac{n(n-1)}{2}$*, the total number of unique two-qubit entangling operations is upper-bounded by the number of pairs of qubits.*

Depending on the degree of each vertex, the connectivity of a quantum simulator can be classified as partial or full.

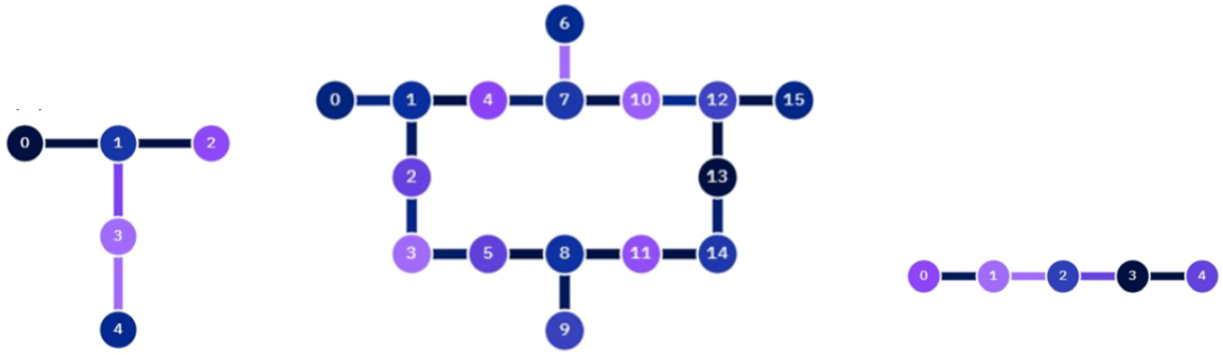
Definition 4. ***Partial Connectivity****: A quantum simulator exhibits partial connectivity if its connectivity graph,* $\boldsymbol{C} = (V, E)$*, is not a complete graph.*

Implication 4. *Implications of definition 4*

- $\exists \boldsymbol{x} \in \boldsymbol{V}$ *such that* $deg(\boldsymbol{x}) < n - 1$*, where* n *is the number of qubits in the simulator.*
- $|\boldsymbol{E}| < \frac{n(n-1)}{2}$*, the total number of unique two-qubit entangling operations is strictly less than the number of pairs of qubits.*

- *Limited direct interactions: Not every pair of qubits can interact through a single unitary operation, thereby increasing the complexity of entangling operations by introducing SWAP gates.*

Executing quantum circuits on partially connected quantum simulators presents challenges in today's Noisy Intermediate-Scale Quantum (NISQ) era. This is primarily because not all qubits are interconnected, necessitating additional SWAP operations to entangle non-adjacent qubits. Novel compilation algorithms are needed to strategically compile the circuit in a way that minimizes these SWAP operations [Mukhopadhyay et al., 2022]. However, these auxiliary operations introduce additional errors, limiting the number of layers successfully executed on the quantum simulator. For instance, Fig. 4.2 illustrates the connectivity of superconducting qubits on three different IBM platforms.



(a) (b) (c)

Figure 4.2: **Connectivity graphs of IBM's superconducting qubit platforms exhibiting partial connectivity.**
a. The five-qubit system, named Vigo, has the "T" connectivity. **b.** The sixteen-qubit system is named Guadalupe **c.** The five-qubit system, named Athens, has linear connectivity. Image source: [Zhang et al., 2021]

Definition 5. ***Full Connectivity****: A quantum simulator exhibits full connectivity if its connectivity graph,* $\boldsymbol{C} = (V, E)$*, is a complete graph.*

Implication 5. *Implications of definition 5*

- $\forall \boldsymbol{x} \in \boldsymbol{V}$*,* $deg(\boldsymbol{x}) = n - 1$*, where* n *is the number of qubits in the simulator.*
- $|\boldsymbol{E}| = \frac{n(n-1)}{2}$*, the total number of unique two-qubit entangling operations equals the number of pairs of qubits.*

In contrast to partially connected systems, fully connected systems eliminates the need for auxiliary SWAP operations to entangle any two qubits. Such direct interaction capabilities facilitate the execution of deeper quantum circuits than partially connected systems. For instance, the ion-trap platform exhibits an all-to-all coupled system, as shown in Fig. 4.3.

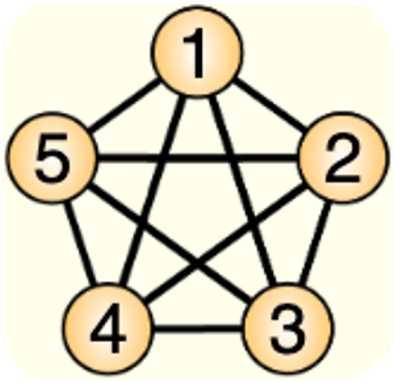


Figure 4.3: **Connectivity graph of an ion-trap platform with five qubits.** Ion-trap platform witnesses full connectivity of qubits, because qubits are all-to-all coupled via a shared vibrational mode of the ion-crystal. Image Source: [Linke et al., 2017]

4.1.3 Static architecture: Connectivity graph = truncated coupling graph

If the truncated coupling graph of the quantum simulator is the same as its connectivity graph, then the platform's architecture is static.

Definition 6. ***Static Architecture****: For a quantum simulator, let its coupling graph be represented as* $\boldsymbol{Q} = (V, E, w)$ *and its truncated coupling graph as* $\boldsymbol{Q'} = (V, E')$*, where* $E' \subset E$ *includes edges* e *such that* $w(e) \geq w_{min}$ *for all* $e \in E$*. If* $\boldsymbol{C} = (V, E'')$ *is the connectivity graph, the architecture is static if and only if* $\boldsymbol{Q'} = \boldsymbol{C}$*.*

Implication 6. *Implications of definition 6*

- *The qubit coupling is fixed and doesn't vary over time, simplifying control, but it may incur SWAP gate overheads for specific operations due to partial connectivity.*

4.1.4 Dynamic architecture: Connectivity graph ≠ truncated coupling graph

If the truncated coupling graph of the quantum simulator changes over time while the connectivity graph remains constant, and all the unique time-defined coupling graphs are subgraphs of the connectivity graph, then the platform's architecture is dynamic. We define the dynamic architecture in a way that's similar to the static architecture but with the introduction of a time variable to account for the temporal changes in the coupling graph.

Definition 7. ***Dynamic Architecture****: For a quantum simulator, let its coupling graph at time t be represented as $\boldsymbol{Q}_t = (V, E_t, w_t)$ and its truncated coupling graph as $\boldsymbol{Q'}_t = (V, E'_t)$, where $E'_t \subset E_t$ includes edges e such that $w_t(e) \geq w_{min}$ for all $e \in E_t$. If $\boldsymbol{C} = (V, E'')$ is the connectivity graph, the architecture is dynamic if and only if for all time points t, $\boldsymbol{Q'}_t \subset \boldsymbol{C}$ and $\boldsymbol{Q'}_t \neq \boldsymbol{Q'}_{t'}$ for at least one pair of distinct time points (t, t').*

Implication 7. *Implications of definition 7*

- *The qubit coupling can be programmed to vary over time and typically offers full connectivity, eliminating the need for SWAP gate overheads.*

4.2 Neutral atom array: A dynamic architecture approach

The qubit couplings on the neutral atom array platform are determined by van der Waals interactions, producing a coupling graph as illustrated in Fig. 4.1b. However, only nearest neighbor interactions, robust enough for quantum operations, are considered in the truncated coupling graph (Fig. 4.1c). The qubit connectivity graph (Fig. 4.5a) follows the truncated coupling graph because a two-qubit entangling gate can only be performed on neighboring qubits in a single operation. However, recent experimental breakthroughs have shown that entanglement can be maintained even when atoms are relocated, enabling dynamic atom coupling through distance manipulation. This yields a fully connected system, as shown in Fig. 4.5b. We refer to these programmable couplings as a dynamic architecture. Another experimental demonstration of such a system exists on ion-trap platforms. Ion crystals containing a few ions (qubits) are moved during the simulation to connect to different ion crystals, enhancing the system's connectivity [Pino et al., 2021].

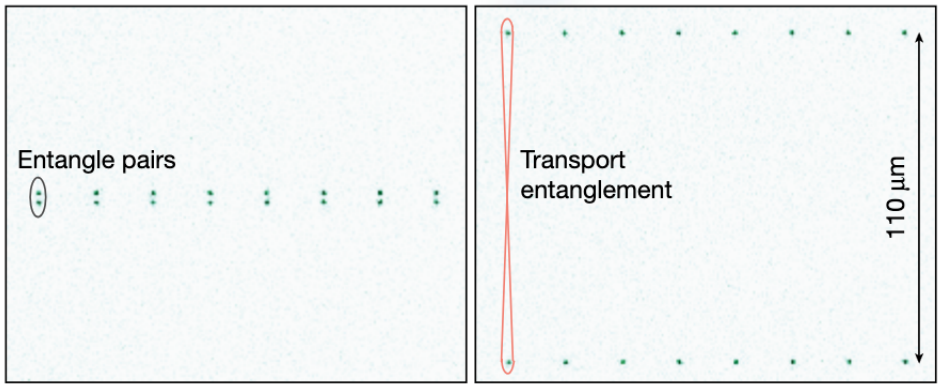
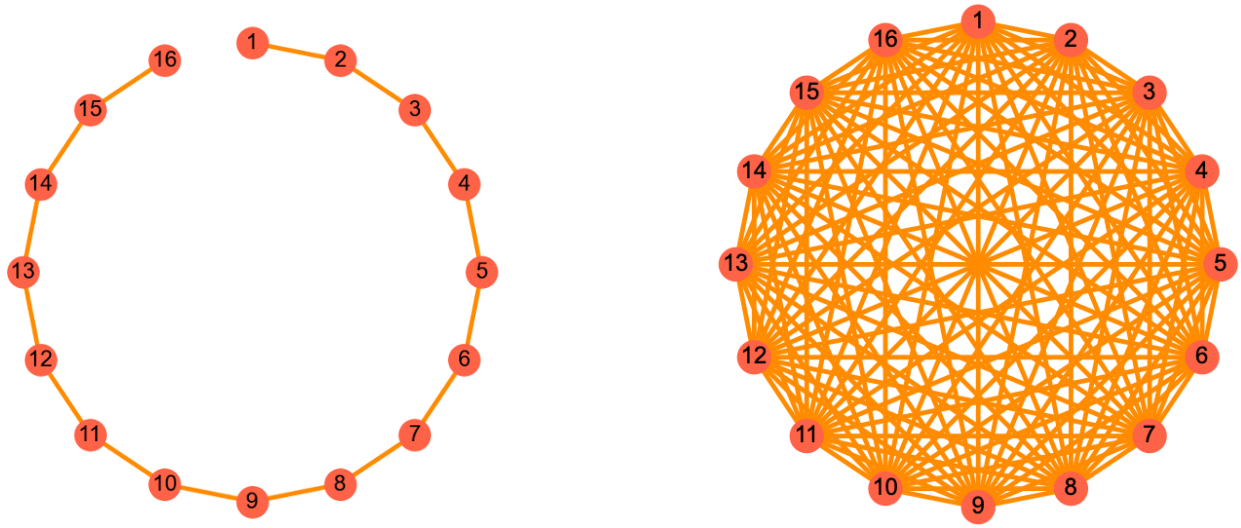


Figure 4.4: **Coherent transport of entangled atoms.**
The approach facilitates entangling operations with remotely situated qubits by transporting them, thereby establishing programmable couplings. Image source: [Bluvstein et al., 2022]



(a) (b)

Figure 4.5: **Connectivity graph of the neutral atom array platform changes after introducing atom moves.**
a. The connectivity graph of a neutral atom array simulator comprising a 1D linear chain of 16 atoms as shown in Fig. 4.1a. **b.** A fully connected graph of the neutral atom platform by relocating atoms during the simulation process, facilitating interactions across all qubits.

4.3 Five-step atom displacement protocol for dynamic architecture

The proposed five-step atom displacement protocol can be visually represented as in Fig. 4.6. The experimental region is divided into two distinct regions: the storage region and the control region. All the qubit registers are initially stored in the storage region, where control operations are restricted. To perform a unitary operation between two qubits, the respective qubits are transferred to the control region. This enables the execution of the desired unitary operation. Notably, if the control region allows for multiple control operations, gate operations can be parallelized efficiently. Parallelization reduces the total circuit execution time by enabling simultaneous execution of multiple gate operations. The ability to exploit parallelism in the control region contributes to the scalability of quantum simulation on neutral atom platforms.

Table 4.1: **Five-step atom displacement protocol to realize dynamic architecture on neutral atom platform.**

Operation	Function	Parameters
T_{sc}	Move relevant atoms to control region	number of atoms, location in storage region
T_{cc}	Move atoms within the control region to bring them next to each other	Location of atoms in control region
U	Allow atoms to interact	Interaction time
T'_{cc}	Revert the steps to move atoms to original location in control region	Previous location in control region
T_{cs}	Bring back atoms to their original position in storage region	Previous location in storage region

The five-step atom displacement protocol, designed to avoid collision of atoms, is outlined in Table 4.1. The protocol starts by moving relevant atoms from the storage region

to the control region (T_{sc}). Subsequently, atoms within the control region are rearranged to bring them into proximity (T_{cc}), facilitating their interaction. The interaction between the atoms is accomplished by allowing them to interact for a specified duration to realize a unitary operation (U). After the desired interaction time, the steps are reversed to restore the atoms to their original positions in the control region (T'_{cc}). Finally, the atoms are returned to their original locations in the storage region (T_{cs}). This protocol strategically moves atoms, providing the programmability to reconfigure the system's coupling between two qubits.

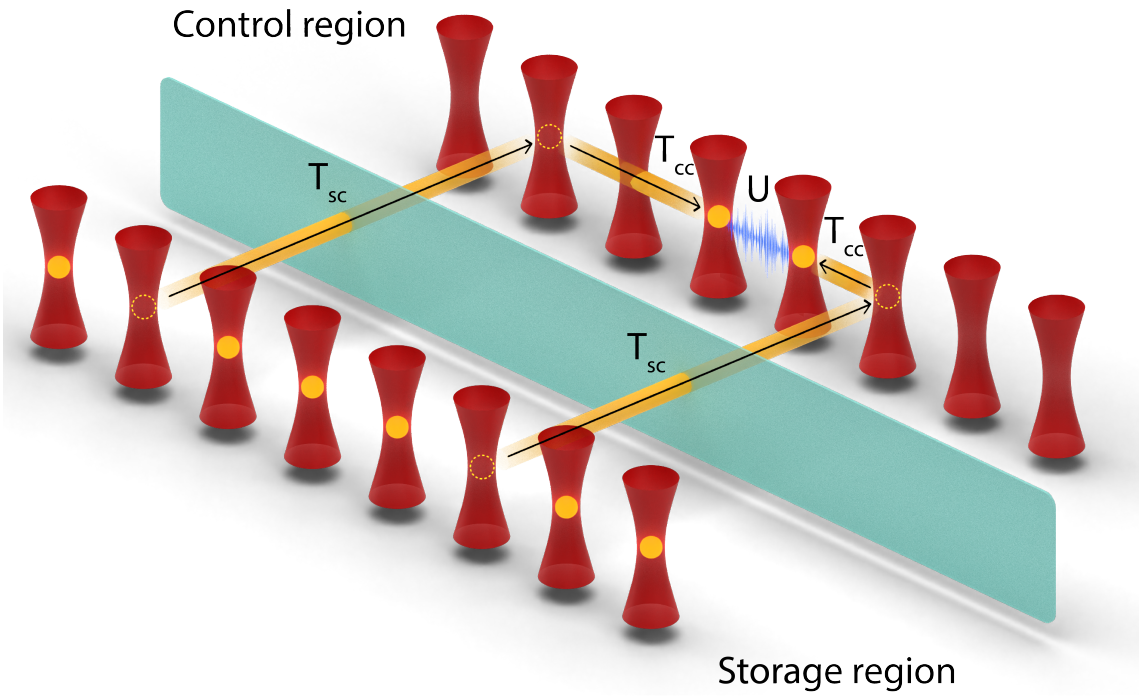


Figure 4.6: **Five-step atom displacement protocol for neutral atom arrays with dynamic architecture.**
Storage and control regions divide the experimental region. Gate operations in the storage region are prohibited. Performing an operation between two qubits necessitates bringing them close in the control region in two steps T_{sc} and T_{cc}. After executing the operation (U), the atomic qubits are brought back via same path in two steps T'_{cc} and T_{cs}.

4.4 Error sources in dynamic architecture

4.4.1 Coherent error

Coherent errors are deterministic and potentially correctable. Error-correction techniques can be employed to counteract such errors, essentially "retrieving" the original state. Within the scope of this work, we focus on two prominent forms of coherent errors - gate error and dephasing error.

4.4.1.1 Gate error

Gate error occurs due to experimental inaccuracies leading to deviations from the ideal gate operations. These errors accumulate with each subsequent gate operation. An upper limit can be estimated on these cumulative errors based on the total number of gate applications.

For instance, let's consider a probabilistic error model where the probability of an erroneous operation is p_g. Let's say the initial density matrix is ρ_0 and the final density matrix with ideal operations and imperfect operations are ρ_m and $\bar{\rho_m}$, respectively. A simple inequality using the Frobenius norm on the difference between two density matrices indicates that the quantum error should be upper bounded by a constant [Yu and Li, 2022]:

$$\|\tilde{\rho} - \rho\|_{\mathrm{F}}^2 = \|\tilde{\rho}\|_{\mathrm{F}}^2 + \|\rho\|_{\mathrm{F}}^2 - 2\,\mathrm{tr}(\tilde{\rho}\rho) \leq 2 \tag{4.2}$$

Because the product of two semi-definite matrices always has a non-negative trace, and the Frobenius norm of a density matrix is always less than 1.

If at least one error occurs in applying m gates then using the above bound $\|\bar{\rho}_m - \rho_m\|_{\mathrm{F}}^2 \leq 2$. The expected error propagation can be bounded as,

$$\begin{aligned}\mathbb{E}\,\|\bar{\rho}_m - \rho_m\|_{\mathrm{F}}^2 &\leq \|\bar{\rho}_m - \rho_m\|_{\mathrm{F}}^2 \cdot \mathbb{P}\text{ (at least one error occurs)} \;+ \|\rho_m - \rho_m\|_{\mathrm{F}}^2 \cdot \mathbb{P}\text{ (no error)} \\ &\leq 2 \cdot \mathbb{P}\text{ (at least one error occurs)} \;+ 0 \cdot \mathbb{P}\text{ (no error)} \\ &= 2\left(1 - (1-p_g)^m\right), \end{aligned} \tag{4.3}$$

where the probability of no error is $(1-p_g)^m$, and at least one error occurs is $1 - (1-p_g)^m$.

Thus, the accumulation of errors upon applying an erroneous gate m times can be approximated as:

$$\mathbb{E}\,\|\bar{\rho}_m - \rho_m\|_{\mathrm{F}}^2 \leq 2\,(1-(1-p_g)^m) \tag{4.4}$$

4.4.1.2 Dephasing error

Dephasing error, on the other hand, originates from the unwanted interaction of a quantum system with its environment. These interactions cause the quantum states to lose phase coherence over time without changing the states' populations, undermining the fidelity of quantum simulations.

4.4.2 Incoherent error

Incoherent errors, unlike coherent errors, are characterized by their irretrievability. These errors are often the result of uncontrolled interactions between the quantum system and its surrounding environment. Incoherent errors lead to a loss of quantum information by degrading the coherence of quantum states. We focus on atom loss error, an artifact of atom displacement operation that shrinks the Hilbert space by a factor of 2, decreasing the accuracy of quantum simulation.

4.4.2.1 Atom loss error

Atom loss or erasure error occurs when an atom leaks from the computational space. While dynamic optical tweezers are moving the atoms, there's a probability of losing the atom during transportation. Atom loss leads to a complete loss of a qubit, thus reducing the size of the Hilbert space by a factor of 2. Such an error can be represented as a completely positive trace-preserving (CPTP) map [Wood and Gambetta, 2018], a type of mathematical operation used to model the evolution of quantum states, given by:

$$\mathcal{E}(\rho) = (1-p_\nu)\rho + p_\nu\,|\Psi_l\rangle\,\langle\Psi_l| \tag{4.5}$$

Here, ρ is the state of the quantum system, $|\Psi_l\rangle$ is a state in the atom-lost subspace. The atom-lost subspace can be thought as a 1- dimensional system which keeps track of the lost atoms. After n applications of the channel, the state leakage $p_\nu(n)$ is given by:

$$p_\nu(n) = 1-(1-p_\nu)^n \tag{4.6}$$

Thus the resulting state after n applications of the error channel is:

$$\mathcal{E}^{\circ n}(\rho) = (1 - p_\nu)^n \rho + p_\nu(n) \left|\Psi_l\right\rangle \left\langle\Psi_l\right| \tag{4.7}$$

This shows that as n increases, the leakage probability approaches 1, meaning that all of the atoms eventually end up in the atom-lost subspace.

Figure 4.7: **Atom loss error demonstration.**

4.5 Quantum circuit compilation protocol

Quantum circuit compilation protocol is a systematic method to translate a given quantum circuit into a set of operations executable on a specific quantum hardware platform. This protocol's task is two-fold: ensuring the circuit's admissibility on the hardware and minimizing the the number of quantum gate operations and atomic movements.

In the context of quantum circuit compilation, Definition 8 introduces the quantum circuit graph $\mathbf{U} = (Q, I)$, which provides an abstract representation of the circuit's structure. The admissibility of the circuit on a given hardware, as outlined in Definition 9, determines whether $\mathbf{U}$ fits into the hardware's connectivity graph. Both these concepts are foundational to the understanding and application of the compilation protocol.

Definition 8. ***Quantum Circuit Graph****: Each quantum circuit has an associated quantum circuit graph, denoted as* $\boldsymbol{U} = (Q, I)$*, where:*

- $\boldsymbol{Q} = q_1, q_2, \ldots, q_n, n \in \mathbb{N}$*, is a finite set of graph vertices that represent the set of qubits in the circuit.*
- $\boldsymbol{I} \subseteq Q \times Q$ *is a finite set of undirected edges. An edge* (q_i, q_j)*, for* $q_i, q_j \in Q$*, signifies that a two-qubit gate operates on qubits* q_i *and* q_j *in the circuit.*

Implication 8. *Implications of definition 8*

- $|\boldsymbol{Q}| = n$, *where* n *is the number of qubits in the circuit.*
- $|\boldsymbol{I}|$ = *total number of distinct two-qubit gates in the circuit.*
- *The degree of a vertex,* $deg(\boldsymbol{q})$, *for* $\boldsymbol{q} \in \boldsymbol{Q}$, *implies the number of distinct qubits with which the qubit* q *shares a two-qubit gate within the circuit. It can range from* 0 *to* $n-1$.
- $|\boldsymbol{I}| \leq \frac{n(n-1)}{2}$, *the total number of unique two-qubit gates is upper-bounded by the number of pairs of qubits.*

Definition 9. ***Admissibility of a Quantum Circuit****: For a given circuit, let the associated quantum circuit graph be* $\boldsymbol{U} = (Q, I)$ *and let the connectivity graph of the quantum simulator be* $\boldsymbol{C} = (V, E)$. *The quantum circuit is admissible on the simulator if and only if* $\boldsymbol{U}$ *is a subgraph of* $\boldsymbol{C}$.

Implication 9. *Implications of definition 9*

- *Admissibility signifies the feasibility of executing all two-qubit gates in the circuit on the given quantum simulator, without additional swap operations.*
- *Non-admissibility of a quantum circuit requires circuit compilation, often involving extra swap gates, to align the circuit with the simulator's connectivity graph.*

Fig. 4.8 illustrates these concepts. Fig. 4.8a shows an example of a quantum circuit, and Fig. 4.8c presents its associated quantum circuit graph **U**. Figures 4.8b and 4.8d represent the connectivity graph **C** of a quantum simulator with partial and full connectivity, respectively. As depicted by the magenta arrows, the circuit is inadmissible on the simulator with partial connectivity, but admissible on the simulator with full connectivity.

Definition 10. ***Compilation Protocol****: A compilation protocol for a given quantum circuit with quantum circuit graph* $\boldsymbol{U} = (Q, I)$ *and* $\boldsymbol{g}$ *gates, generates an equivalent and admissible circuit with quantum circuit graph* $\boldsymbol{U'} = (Q, I')$ *and* $\boldsymbol{g'}$ *gates for a quantum simulator with connectivity graph* $\boldsymbol{C} = (V, E)$. *This protocol aims to minimize both the gate count* $\boldsymbol{g'}$ *and atom moves (if allowed) in the compiled circuit.*

Implication 10. *Implications of definition 10*

- *For inadmissible circuits, i.e.,* $\boldsymbol{U} \nsubseteq \boldsymbol{C}$, *the compiled equivalent* $\boldsymbol{U'} \subseteq \boldsymbol{C}$ *usually has more gates:* $\boldsymbol{g'} \geq \boldsymbol{g}$, *owing to the addition of swap gates.*
- *For admissible circuits, i.e.,* $\boldsymbol{U} \subseteq \boldsymbol{C}$, *the compiled equivalent* $\boldsymbol{U'} \subseteq \boldsymbol{C}$ *usually has fewer gates:* $\boldsymbol{g'} \leq \boldsymbol{g}$. *However, there may be additional atom move operations.*

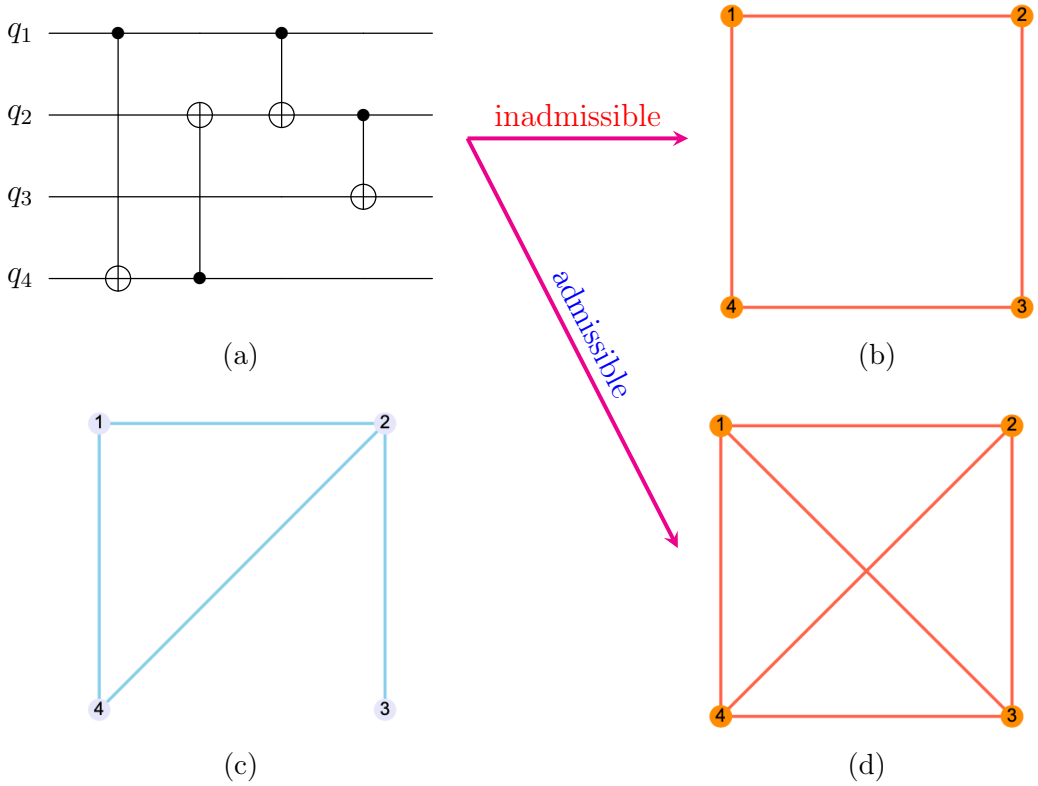


Figure 4.8: **Quantum circuit admissibility illustration.**
a. A quantum circuit with four qubits and four CNOTs. **c.** Corresponding quantum circuit graph $\mathbf{U} = (Q, I)$ based on definition 8. **b.** A quantum simulator's connectivity graph $\mathbf{C} = (V, E)$ with partial connectivity where the circuit is not admissible ($\mathbf{U} \nsubseteq \mathbf{C}$). **d.** A quantum simulator's connectivity graph $\mathbf{C} = (V, E)$ with full connectivity where the circuit is admissible ($\mathbf{U} \subseteq \mathbf{C}$).

4.5.1 Compiling CNOT circuits for static architecture

Circuit compilation for static, partially connected quantum hardware requires generating an equivalent circuit that respects the hardware's specific connectivity constraints. A notable approach to this problem, as detailed in [Mukhopadhyay et al., 2022], employs the Steiner tree method. This strategy introduces additional swap gates, enabling the transformation of the original circuit into an equivalent form compatible with the hardware's architecture. For instance, consider a quantum circuit as shown in Fig. 4.9. When this cir-

cuit is compiled for a static architecture with nearest-neighbor connectivity, the algorithm generates the equivalent circuit presented in Fig. 4.10.

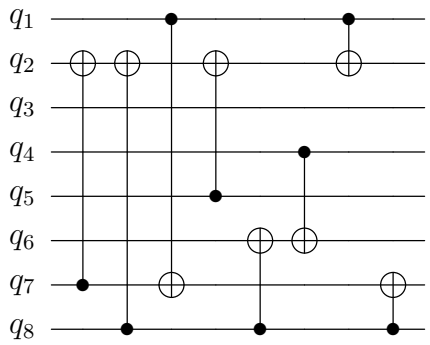


Figure 4.9: **An exemplary circuit comprising eight qubits and eight CNOTs.**

When compiling for a static architecture with nearest-neighbor connectivity, the two-qubit gates can only be applied between neighboring qubits. The stated technique generates a compiled circuit by introducing swap operations. It decomposes the swap operation into three CNOT gates and optimizes the circuit to minimize the number of CNOT gates.

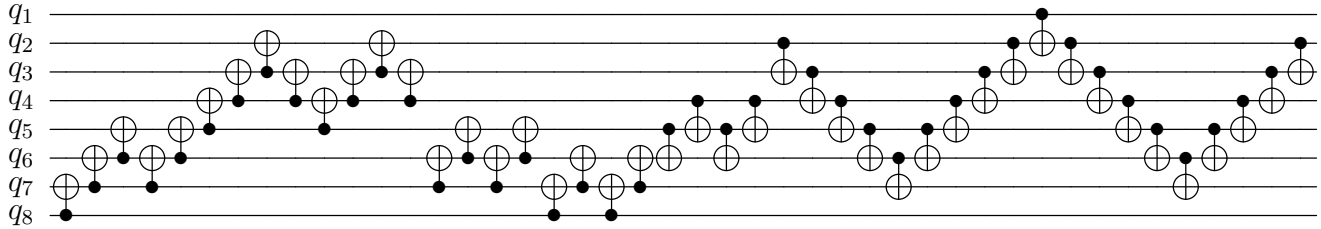


Figure 4.10: **Circuit compiled for static architecture with nearest-neighbor connectivity.**

4.5.2 Compiling CNOT circuits for dynamic architecture

Circuit compilation for a dynamic, fully connected neutral atom platform initiates by forming a transformation matrix that captures all CNOT gates within the circuit. This

approach, however, does not maintain the sequential order of the CNOTs, leading to the possibility of distinct circuits corresponding to an identical transformation matrix. Having derived the transformation matrix, a subsequent algorithm aims to reverse the circuit's operations to minimize gate count and atom movements in compliance with the hardware's constraints. This reversed execution generates an equivalent circuit, often characterized by fewer gates. Below are the transformation matrix construction algorithm and a tailored circuit optimization method that minimizes gate operations and atom movements. It's important to note that the underlying atom topology strongly influences the atom movement minimization strategy. Therefore, any changes in this topology may necessitate modifications to the optimization routine.

The transformation matrix generation algorithm, as described in Algorithm 1, creates a matrix that encapsulates the transformations performed by a sequence of CNOT gates in a quantum circuit. The algorithm begins by initializing an identity matrix of size $N_q \times N_q$, where N_q represents the total number of qubits. It then iteratively applies each CNOT gate in the provided list to the transformation matrix. Specifically, for every CNOT gate, the algorithm modifies the row corresponding to the target qubit by performing a bitwise XOR operation with the row corresponding to the control qubit.

Algorithm 1 Generate transformation matrix for N_q qubits

Input: CNOT gate sequence list
Output: $N_q \times N_q$ transformation matrix T

$T \leftarrow I_{N_q}$ ▷ Initialize T as $N_q \times N_q$ identity matrix
procedure TRANSFORMATION($T, list$)
    for each $CNOT$ in $list$ **do**
        $c \leftarrow$ CNOT[control-qubit]
        $t \leftarrow$ CNOT[target-qubit]
        $T[t, .] \leftarrow T[c, .] \oplus T[t, .]$

The key advantage of this transformation matrix approach is its capacity to simplify the quantum circuit by identifying and eliminating redundant gate operations. Notably, it can detect and cancel out pairs of consecutive CNOT gates that negate each other's effects. Furthermore, by transforming the circuit into a matrix, this method offers a concise and abstract representation for further circuit optimization.

The circuit optimization algorithm for dynamic architecture, as described in Algorithm 2, takes as input a transformation matrix that encapsulates the actions of a sequence of CNOT gates. It outputs an optimized sequence of CNOT gates. The algorithm begins by

initializing an empty list to store the resultant CNOT gate sequence. It then traverses each column in the transformation matrix in order. For each column, the algorithm examines elements below and above the current diagonal entry in the matrix. If any of these elements are one, indicating the presence of a CNOT gate, the algorithm adds this gate to the sequence, with the current column as the control qubit and the respective row as the target qubit. Again, it updates the transformation matrix via an XOR operation on the corresponding rows.

This algorithm's main strength lies in its capacity to structure the compiled CNOT gate sequence in dynamic architecture, thus reducing gate count and minimizing atom movements, all in alignment with the atom topology shown in Fig. 4.11. The presented topology features a 1D chain of eight qubits. Any CNOT operation necessitates bringing the control qubit near the target qubit. The control operation fields are stacked vertically, with the control qubit always positioned at the top and the target qubit at the bottom. Furthermore, this linear chain can be extended following the same pattern, offering a scalable solution to increase the system size when necessary.

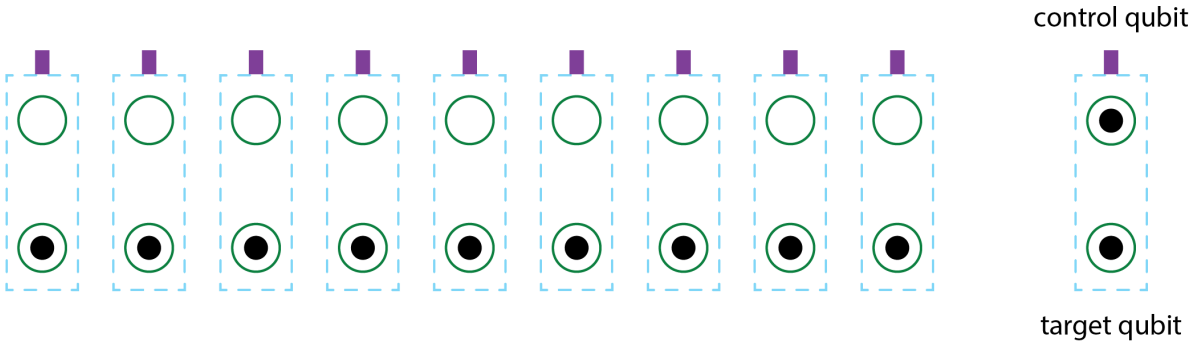


Figure 4.11: **Atom topology for circuit optimization.**

For instance, consider the quantum circuit shown in Fig. 4.12a (same as Fig. 4.9), including eight qubits and eight CNOTs. The compiled circuit contains only seven CNOTs, as shown in Fig. 4.12b.

The compilation protocol generates a systematically arranged CNOT sequence where the control qubit index strictly increases. This means that when a control qubit is selected for movement, all related operations are completed, and the qubit isn't moved again, reducing the total number of atom movements.

Algorithm 2 Circuit optimization for dynamic architecture

Input: Transformation matrix T
Output: CNOT gate sequence list C

Initialize $C \leftarrow \emptyset$ ▷ Create an empty list to store the CNOT sequence
for each *column* i from 1 to N_q **do**
    if $T_{ii} = 0$ **then**
        Find the nearest row $j (j > i)$ with $T_{ji} = 1$
        Add $CNOT_{j,i}$ to C
        $T[i, .] \leftarrow T[j, .] \oplus T[i, .]$
    for j from $i + 1$ to N_q **do**
        if $T_{ji} = 1$ **then**
            Add $CNOT_{i,j}$ to C
            $T[j, .] \leftarrow T[i, .] \oplus T[j, .]$
    for j from $i - 1$ down to 1 **do**
        if $T_{ji} = 1$ **then**
            Add $CNOT_{i,j}$ to C
            $T[j, .] \leftarrow T[i, .] \oplus T[j, .]$

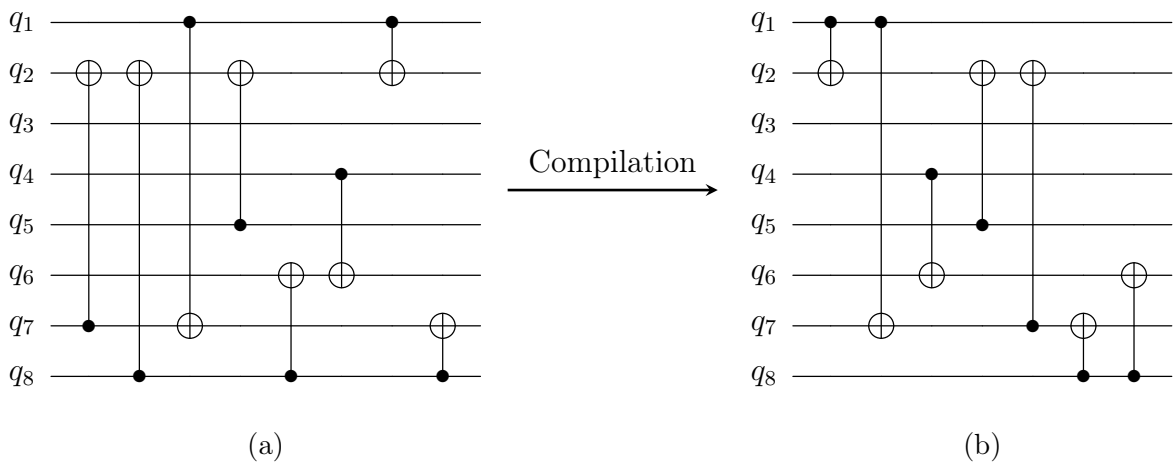


Figure 4.12: **Compilation protocol illustration for an eight-qubit CNOT circuit.**

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_7 \oplus R_2 \to R_2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_8 \oplus R_2 \to R_2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\xrightarrow{R_1 \oplus R_7 \to R_7} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_5 \oplus R_2 \to R_2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_8 \oplus R_6 \to R_6}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_4 \oplus R_6 \to R_6} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_1 \oplus R_2 \to R_2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\xrightarrow{R_8 \oplus R_7 \to R_7} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 4.13: **Transformation matrix generation.**

Figure 4.13 and Figure 4.14 demonstrates the process of compiling a quantum circuit. Initially, a transformation matrix is generated from the original circuit (Fig. 4.12a). Then, this matrix is used to produce the optimized, compiled circuit. Row-reduction-like operations are performed for each diagonal element in ascending order. This systematic procedure guarantees a consistent control qubit order in the compiled circuit, an essential factor in minimizing atom movements.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_1 \oplus R_2 \to R_2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_1 \oplus R_7 \to R_7} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\xrightarrow{R_4 \oplus R_6 \to R_6} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_5 \oplus R_2 \to R_2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_7 \oplus R_2 \to R_2}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_8 \oplus R_7 \to R_7} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{R_8 \oplus R_6 \to R_6} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 4.14: **Circuit optimization algorithm for dynamic architecture.**

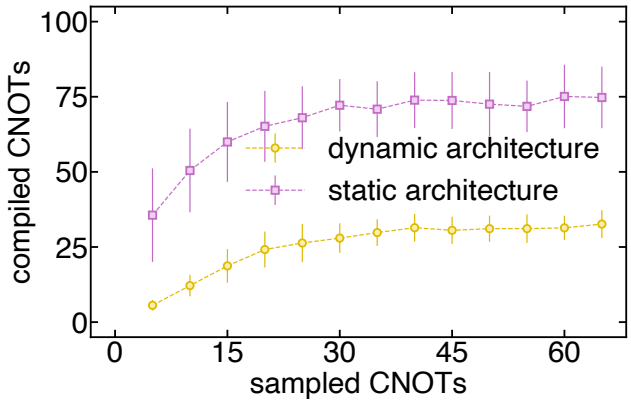
4.5.3 Cross-benchmarking compilation protocols for static and dynamic architecture

To conduct cross-benchmarking of static and dynamic architecture for randomly sampled CNOT circuits, we employ a systematic process as detailed in Algorithm 3. This process begins with the number of qubits and the count of CNOT gates as inputs and subsequently generates circuits possessing a randomized sequence of these gates. The circuits were then compiled for static and dynamic architecture, and the respective number of CNOT gates after compilation was recorded. Additionally, for the dynamic architecture, we calculate the number of atom displacements before and after the compilation. Based on these values, we calculated the $\eta_{protocol}$ metric (derived in Section 4.6), which indicates the efficiency of the compilation protocols.

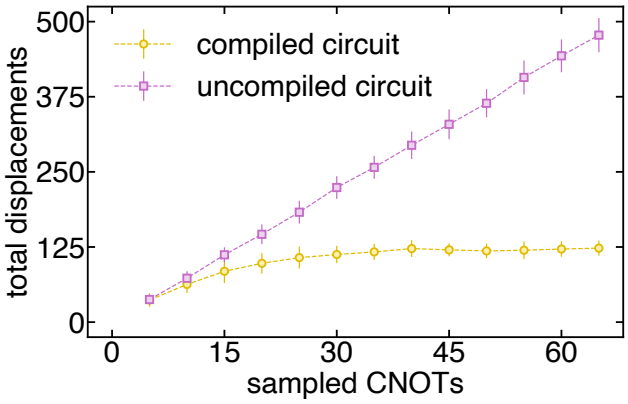
Algorithm 3 Benchmarking compilation protocols

Input: Number of qubits N_q, Number of CNOT gates M
Output: $cnotStatic$, $cnotDynamic$, $atomDispBefore$, $atomDispAfter$, η_{uncomp}, η_{comp}

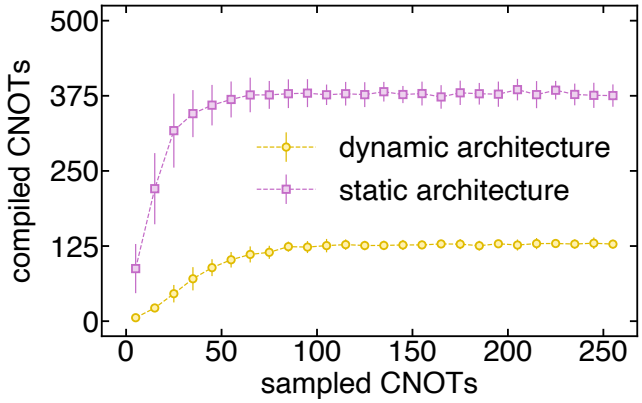
Initialize all the outputs as 0
$cnotSeq \leftarrow$ randomCNOTs(qubits:N_q, gates:M) ▷ Sample M random CNOT gates
procedure COMPILESTATIC($cnotSeq$)
    $cnotStatic \leftarrow$ compile($cnotSeq$) ▷ Compile for static architecture
    return $cnotStatic$
procedure COMPILEDYNAMIC($cnotSeq$)
    $atomDispBefore \leftarrow$ computeDisplacement($cnotSeq$) ▷ Compute disp before
    $cnotDynamic \leftarrow$ compile($cnotSeq$) ▷ Compile for dynamic architecture
    $atomDispAfter \leftarrow$ computeDisplacement($cnotSeq$) ▷ Compute disp after
    return $atomDispBefore$, $cnotDynamic$, $atomDispAfter$
procedure COMPUTEETA($cnotSeq1$,$atomDisp$, $cnotSeq2$)
    return $\frac{atomDisp}{numcnotSeq2 - numcnotSeq1}$
$cnotStatic \leftarrow$ compileStatic($cnotSeq$)
$atomDispBefore$, $cnotDynamic$, $atomDispAfter \leftarrow$ compileDynamic($cnotSeq$)
$\eta_{uncomp} \leftarrow$ computeEta($cnotSeq$, $atomDispBefore$, $cnotStatic$) ▷ Calculate $\eta_{protocol}$
$\eta_{comp} \leftarrow$ computeEta($cnotDynamic$,$atomDispAfter$, $cnotStatic$) ▷ Calculate $\eta_{protocol}$



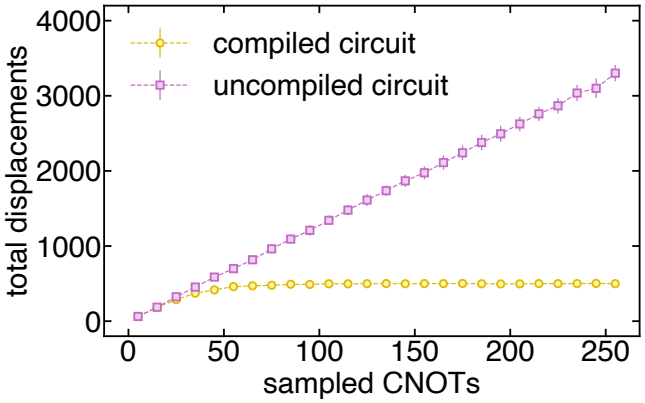
(a) $N_q = 8$



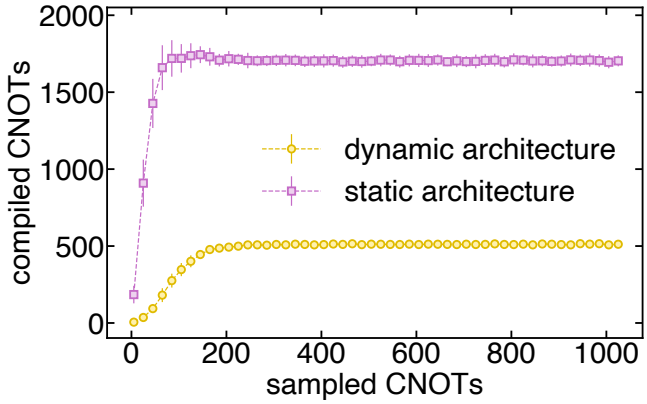
(b) $N_q = 8$



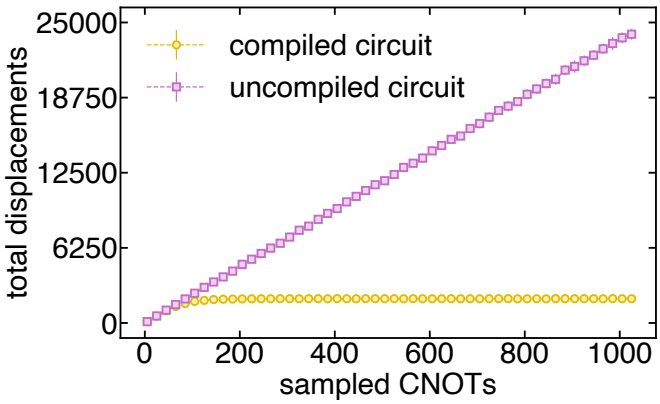
(c) $N_q = 16$



(d) $N_q = 16$



(e) $N_q = 32$



(f) $N_q = 32$

Figure 4.15: **Compilation protocol results.**

This benchmarking process was applied to systems with 8, 16, and 32 qubits, and the number of gates varied up to 2^n where n is the number of qubits. To ensure statistical significance and account for the randomness in our process, we performed this benchmarking procedure on 50 randomly sampled circuits with the same parameters, essentially implementing a Monte Carlo simulation. The results of this extensive benchmarking are presented in Fig. 4.15. Each data point in the graphs represents the outcome of the benchmarking process for one set of 50 sampled circuits.

Our results highlight the efficiency of the compilation protocol for dynamic architecture. From the three figures on left 4.15a, 4.15c, and 4.15e, we observe a notable reduction in CNOT gates compared to static architecture. On the right, the three figures 4.15b, 4.15d, and 4.15f show fewer atom moves in compiled versus uncompiled circuits because of the structure imposed by compilation.

4.6 Performance metric: Circuit fidelity for static and dynamic architecture

In this section, we analyze the performance of dynamic architecture compared to static architecture in executing a series of two-qubit gates on a quantum simulator. We consider the impact of coherent gate error (p_g) and incoherent atom loss error (p_ν) on the fidelity of circuit execution. The key parameters involved are:

- p_g: Coherent gate error probability.
- p_ν: Incoherent atom loss error probability.
- N_g^s: Number of gates in the static architecture.
- N_g^d: Number of gates in the dynamic architecture.
- N_ν: Number of atom displacements in the dynamic architecture.

We compare the fidelity of circuit execution between static and dynamic architecture, considering the number of gates and displacements involved. The table below summarizes the fidelity comparison:

Table 4.2: **Fidelity comparison of static and dynamic architecture.**

	Number of gates	**Number of displacements**	**Fidelity**
Static	N_g^s	0	$(1-p_g)^{N_g^s}$
Dynamic	N_g^d	N_ν	$(1-p_g)^{N_g^d}(1-p_\nu)^{N_\nu}$

Dynamic architecture offers an advantage when the fidelity of circuit execution is higher than that of static architecture. By analyzing the equations, we find that:

$$(1-p_g)^{N_g^d}(1-p_\nu)^{N_\nu} > (1-p_g)^{N_g^s} \tag{4.8}$$

Dividing both sides by $(1-p_g)^{N_g^d}$:

$$(1-p_\nu)^{N_\nu} > (1-p_g)^{N_g^s - N_g^d} \tag{4.9}$$

Taking log on both sides:

$$N_\nu \log(1-p_\nu) > (N_g^s - N_g^d)\log(1-p_g) \tag{4.10}$$

Given $N_g^s > N_g^d$ and $\log(x) < 0, \forall x < 1$, the inequality sign flips:

$$\frac{N_\nu}{N_g^s - N_g^d} < \frac{\log(1-p_g)}{\log(1-p_\nu)} \tag{4.11}$$

Defining $\eta_{\text{protocol}} = \frac{N_\nu}{N_g^s - N_g^d}$ as the efficiency of the compilation protocol for the platform and $\eta_{\text{platform}} = \frac{\log(1-p_g)}{\log(1-p_\nu)}$ as a metric of the experimental setup, we find that:

$$\eta_{\text{protocol}} < \eta_{\text{platform}} \tag{4.12}$$

The equation 4.12 offers two interpretations. First, it helps determine the necessary conditions for a compilation protocol on dynamic architecture to outperform the static architecture given a specific experimental setup. Such a comparison enables the strategic design of compilation protocols that align with the capabilities and limitations of the

experimental setup. Second, it identifies the error probability range where dynamic architecture would outperform static architecture for a given compilation protocol. Fig. 4.16 visually illustrates the gate versus atom displacement error relationship. For a given compilation protocol efficiency (η_{protocol}), the advantageous region for dynamic architecture is the bottom right half of η_{protocol} curve.

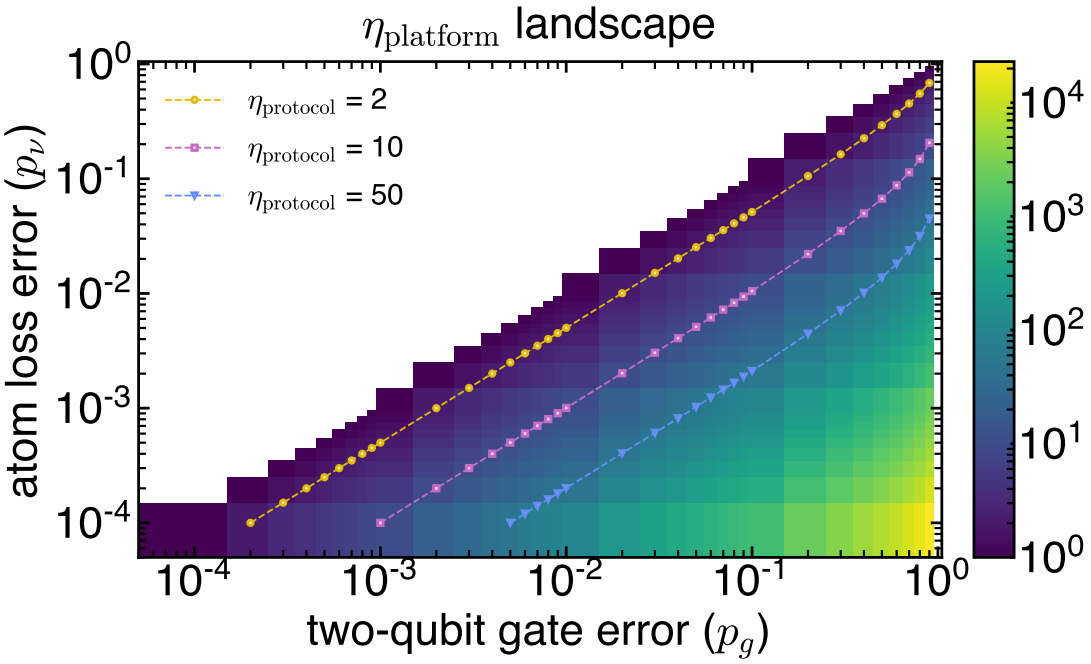
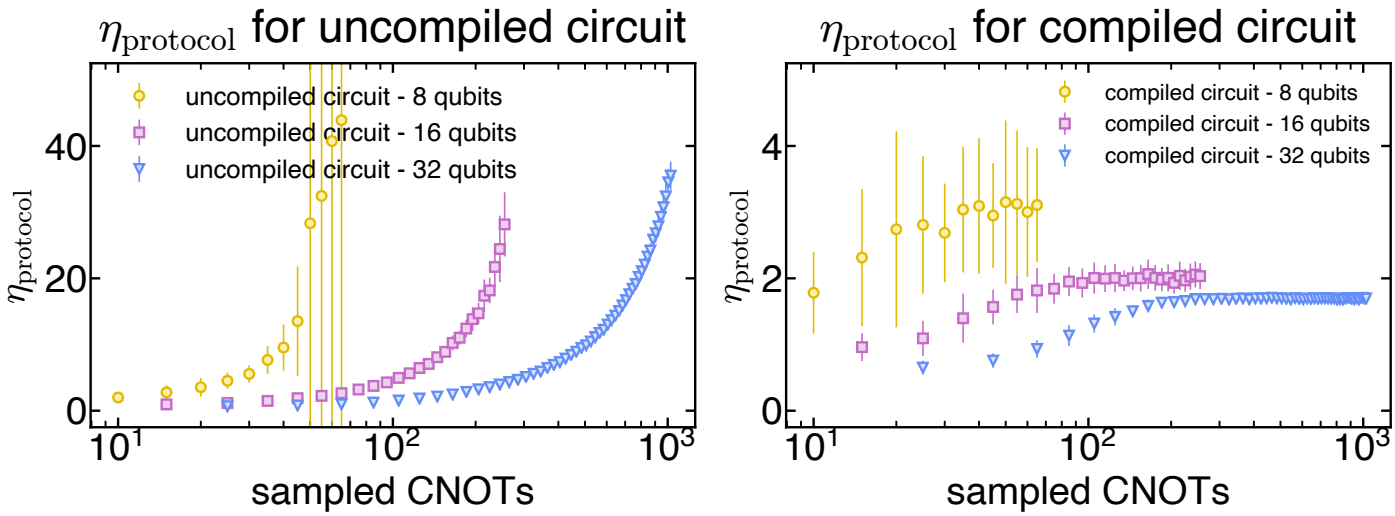


Figure 4.16: **Trade-off between two-qubit gate error and atom loss error to design a protocol where dynamic outperforms static architecture.**

4.6.1 Cross-benchmarking operational protocols for CNOT circuit execution

Using the performance metric derived in the previous section, we evaluate our compilation protocols described in Sec. 4.5.

Comparing the η_{protocol} of compiled and uncompiled circuits, we find that a smaller value, denoting superior efficiency, is consistently associated with compiled circuits. This firmly establishes the effectiveness of our compilation protocol for dynamic architecture.



(a) (b)

Figure 4.17: **Efficiency of compilation protocol.**
a. Protocol performance for uncompiled circuits. **b.** Protocol performance for compiled circuits.

In this context, we utilize the circuit fidelity metric to compare two operational protocols within dynamic architecture. The uncompiled circuit involves executing the initial circuit directly on a fully connected architecture, resulting in a rapid increase in η_{protocol} due to the larger number of required atom movements. Conversely, the compiled circuit designed for dynamic architecture offers structural advantages for our platform, resulting in fewer atom movements and, consequently, a reduction in η_{protocol}.

To estimate η_{platform}, we consider the cutting-edge two-qubit gate fidelity and atom movement fidelity within the neutral atom platform. The two-qubit gate fidelity has demonstrated exceptional performance, reaching as high as 99.5% [Evered et al., 2023], equivalent to $p_g = 0.005$. Additionally, research has shown that the infidelity related to atom displacement is approximately ∼1.8e-4 [Tan et al., 2023]. As a result, η_{platform} is estimated to be around 25. This value significantly exceeds the η_{protocol} associated with our compilation protocol, implying that compiled circuits designed for dynamic architecture are more likely to succeed than uncompiled circuits.

4.7 Performance metric: Quantum volume for static and dynamic architecture

Quantum Volume (QV) is a unified metric introduced by IBM to assess the computational capabilities and error susceptibilities of quantum processors. This metric quantifies the largest square quantum circuits a system can execute effectively. The structure of these square circuits remains consistent across different computing architectures, although architecture-specific compilers can optimize and adapt them to leverage the unique features of each architecture. Consequently, QV accounts for factors such as error rates, inter-qubit connectivity, compilation algorithm efficiency, and qubit routing strategies. This standardization allows for meaningful comparisons between diverse quantum architectures [Cross et al., 2019] [Jurcevic et al., 2020].

A prototypical square circuit C with n qubits and a depth of n can be visualized as shown in Fig. 4.18. Each layer of this circuit is defined by a randomly permuted arrangement of qubit indices and includes $\lfloor \frac{n}{2} \rfloor$ two-qubit gates U, drawn from the Haar measure on $SU(4)$.

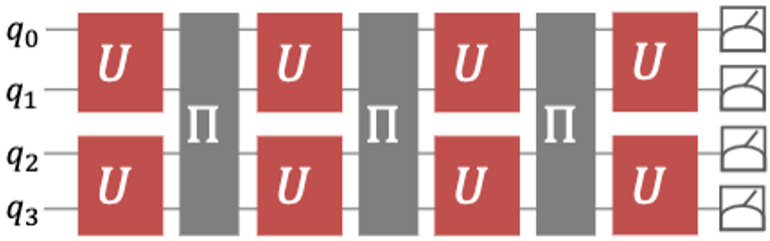


Figure 4.18: **An exemplary square quantum volume circuit.** [Baldwin et al., 2022]

The success of a model quantum circuit on a quantum architecture is determined by addressing the heavy output generation problem. The ideal output of the circuit C is $p_C(x) = |\langle x| C |0\rangle|^2$, where $x \in \{0,1\}^m$ corresponds to an observable bit string. The heavy outputs are identified by comparing individual output probabilities to the median of the output probability set, formally defined as:

$$H_C = \{x \in \{0,1\}^m \; such \; that \; p_C(x) > p_{median}\}. \tag{4.13}$$

If the probability of obtaining a heavy output from the experimental distribution exceeds $\frac{2}{3}$, it indicates that the quantum simulator has successfully passed the quantum

volume test for the circuit C [Aaronson and Chen, 2016]. Considering the observed experimental distribution as $q_C(x)$, the probability of sampling a heavy output is calculated as follows:

$$h_C = \sum_{x \in H_C} q_C(x) \tag{4.14}$$

To analyze the quantum volume for static and dynamic architectures, I introduce the simplest way to incorporate errors. We assume single-qubit gates are error-free. The native two-qubit gate of a neutral atom-based quantum simulator is the CZ gate. CZ gate is a controlled rotation by π about the z-axis and then a phase gate (S) on the first qubit, i.e., $(CR_Z(\pi))(S \otimes \mathbb{I})$. We introduce physically motivated error wherein the rotation deviates from ideal rotation by a small δ. The erroneous CZ operation thus becomes $(CR_Z(\pi - \delta))(S \otimes \mathbb{I})$. The matrix representation of the erroneous CZ gate is:

$$\tilde{U}_{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\delta}{2} + i\sin\frac{\delta}{2} & 0 \\ 0 & 0 & 0 & -\cos\frac{\delta}{2} + i\sin\frac{\delta}{2} \end{pmatrix} \tag{4.15}$$

The average gate fidelity of this erroneous operation can be evaluated using Eq. 4.16, where $\tilde{U}$ denotes the closest achievable operation on the quantum hardware to the ideal operation U [Cross et al., 2019], which results in Eq. 4.17.

$$F_{\text{avg}}\left(U, \tilde{U}\right) = \frac{\left|\text{Tr}\left(U^\dagger \tilde{U}\right)\right|^2 / 2^m + 1}{2^m + 1} \tag{4.16}$$

$$F_{\text{avg}}\left(U_{CZ}, \tilde{U}_{CZ}\right) = (\cos^2\frac{\delta}{2} + 2\cos\frac{\delta}{2} + 2)/5 \tag{4.17}$$

Subsequently, we calculate the average gate error as $\epsilon = 1 - F_{\text{avg}}$, leading to:

$$\epsilon = 1 - F_{\text{avg}} = (3 - \cos^2\frac{\delta}{2} - 2\cos\frac{\delta}{2})/5 \tag{4.18}$$

To demonstrate the effect of architecture and compilation for static and dynamic architecture, I show an exemplary 3-qubit quantum volume circuit and its decomposition to

realize on our platform. All the circuit simulations done below are performed using the Qiskit library [Qiskit contributors, 2023].

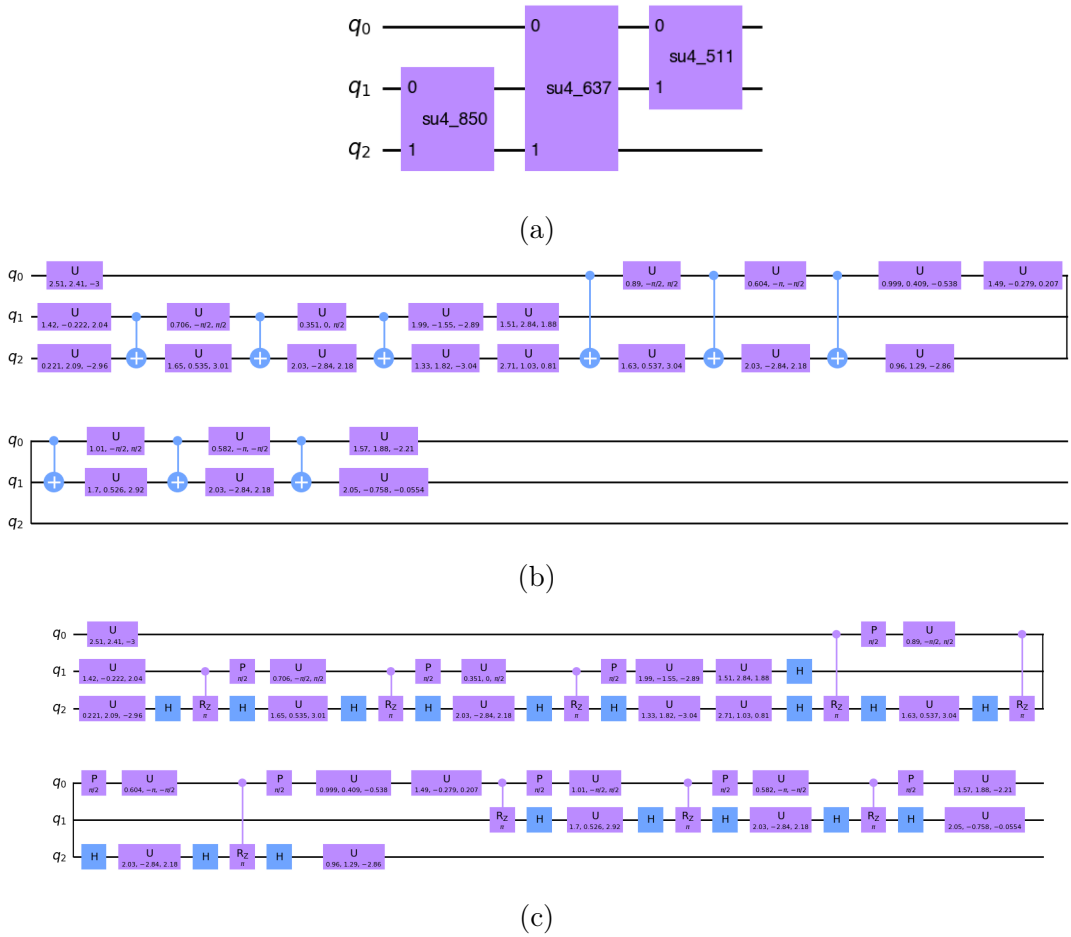


Figure 4.19: **Decomposing 3-qubit quantum volume circuit to native gate set. a.** An exemplary 3-qubit quantum volume circuit. **b.** Decomposing $SU(4)$ gates to single qubit and two-qubit gates using Qiskit. **c.** Replacing CNOT gates with native two-qubit CZ gates.

Compiling the circuit in Fig. 4.19b for static architecture using tket library [Sivarajah et al., 2020], introduces additional swap gates (which are decomposed into three CNOT gates) as shown in Fig. 4.21b. It is important to note that the compilation also routes the qubit to minimize the number of gates, thus $node_0 = q_0$, $node_1 = q_2$, and $node_2 = q_1$.

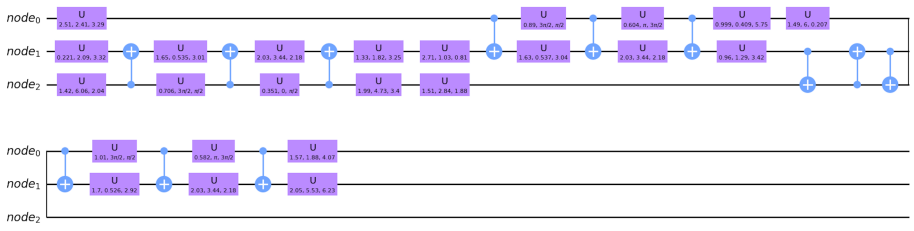
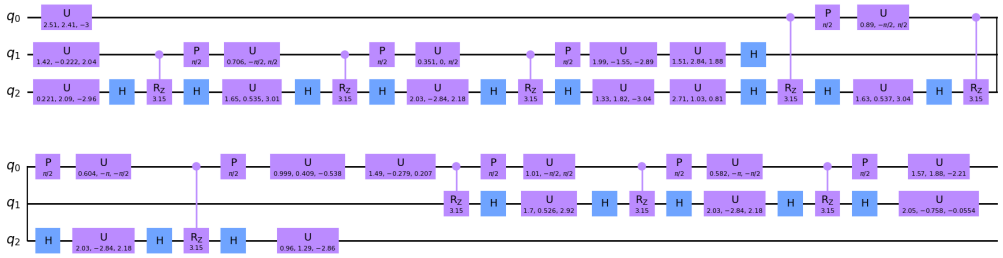
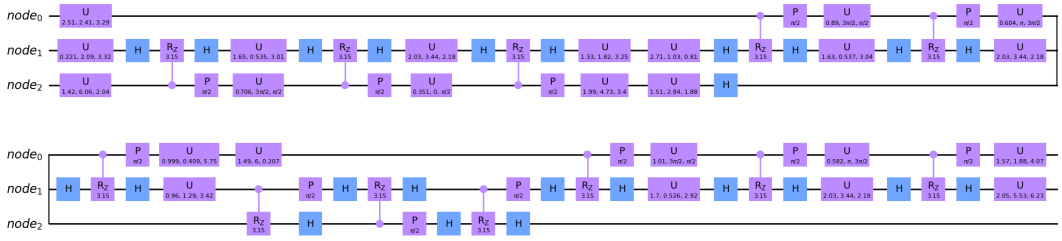


Figure 4.20: **Compiling quantum volume circuit for static architecture.** The circuit is compiled for static architecture with the nearest neighboring interaction, leading to additional SWAP operations, which are decomposed into 3 CNOT gates.

Errors are introduced in the circuit by replacing the CZ gate with an erroneous gate. For instance, the δ (as introduced in Eq. 4.17) for the circuit shown in Fig. 4.21a is 0.01 (<0.5% relative error). This is the circuit that is implemented on the platform with dynamic architecture by allowing atom moves.



(a)



(b)

Figure 4.21: **Erroneous quantum volume circuits for both architectures.** **a.** Dynamic architecture. **b.** Static architecture. Introducing error in two-qubit CZ gate with imperfect rotation about Z quantified as δ. With δ of 0.01 (<0.5% relative error), the CR_Z angle is 3.15 rad instead of π rad for CZ.

Given the erroneous circuit for static and dynamic architecture, I provide an example of checking if a circuit passes the quantum volume test. Consider the 4-qubit quantum volume circuit given in Fig. 4.22. The output probabilities of the circuit are shown in Fig. 4.23a and are shown in sorted order in Fig. 4.23b to visualize the set of states that belong to heavy outputs. The heavy outputs of the circuit add to 0.84 ($> \frac{2}{3}$). Thus, the circuit passes the quantum volume test. For an ideal device, the expected heavy output probability is asymptotically $\frac{1+ln2}{2} \sim 0.85$ [Cross et al., 2019], while for a completely depolarized device it drops to $\sim$ 0.5. An important verification is to check if the output probabilities of the sampled quantum quantum volume circuits follow Haar randomness. The evidence of Haar randomness in the output probabilities of sampled circuits is demonstrated in Fig. 4.24.

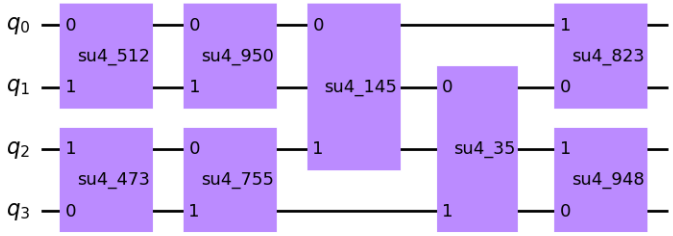
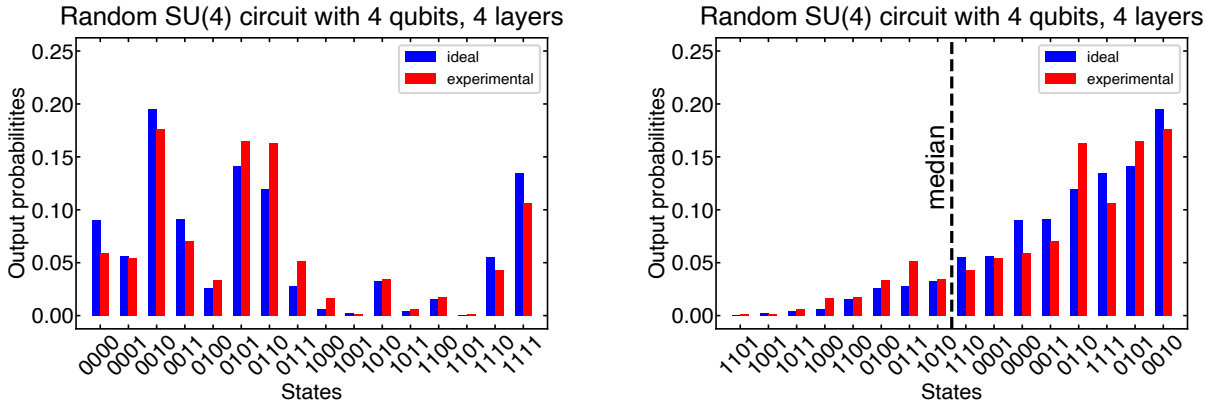


Figure 4.22: **A 4-qubit quantum volume circuit.**



(a) (b)

Figure 4.23: **Output probability distribution.**
a. Output probability distribution of ideal and experimental output probability distributions. **b.** Sorted ideal output probability distribution to visualize heavy output states.

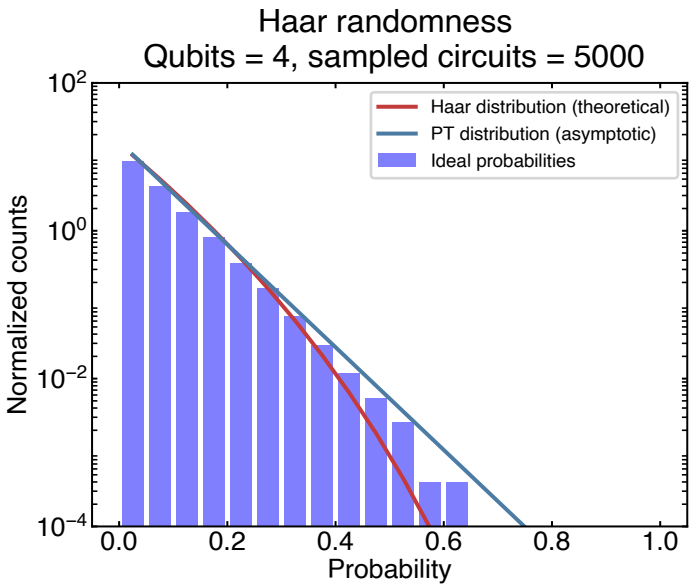


Figure 4.24: **Evidence of Haar randomness in 4-qubit quantum volume circuits.**

The primary question addressed by this analysis is the determination of the maximum allowed gate error required to achieve a specific quantum volume for both architectural approaches, as visualized in Fig. 4.25. Each data point is calculated by evaluating 500 quantum volume circuits. Here, ϵ retains its meaning from Eq. 4.18.

The maximum allowed two-qubit gate error thresholds needed to attain a particular quantum volume are notably smaller for the dynamic architecture. It is essential to note that in dynamic architecture, the number of atom moves for n-qubit circuits is upper-bounded by $2n^2$. This distinction becomes evident when observing the plot of maximum allowable two-qubit gate error thresholds for dynamic architecture, considering scenarios both with and without atom loss errors. The atom loss error value of 1.8e-4 is sourced from [Tan et al., 2023]. In our calculations, we've considered the current state-of-the-art in error rates, including a two-qubit gate fidelity of 99.5% [Evered et al., 2023], which translates to a 0.005 two-qubit gate error. Given these error rates, the attainable quantum volume on both static and dynamic architectures is summarized in Table 4.3.

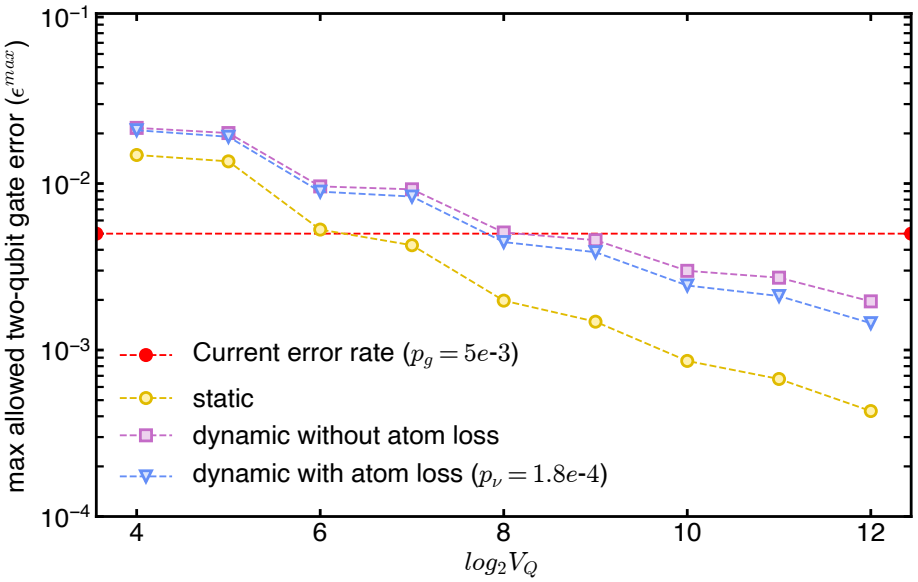
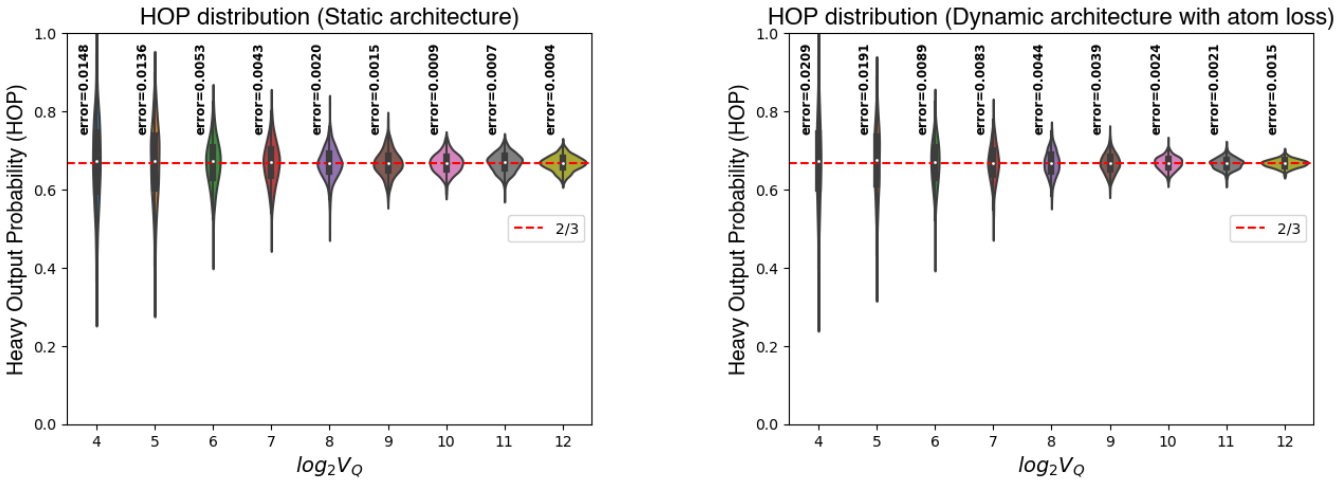


Figure 4.25: **Maximum allowed two-qubit gate error to achieve a certain quantum volume with static and dynamic architectures.**

Table 4.3: **Achievable quantum volume on static and dynamic architectures with the state-of-the-art two-qubit gate and atom move fidelities.**

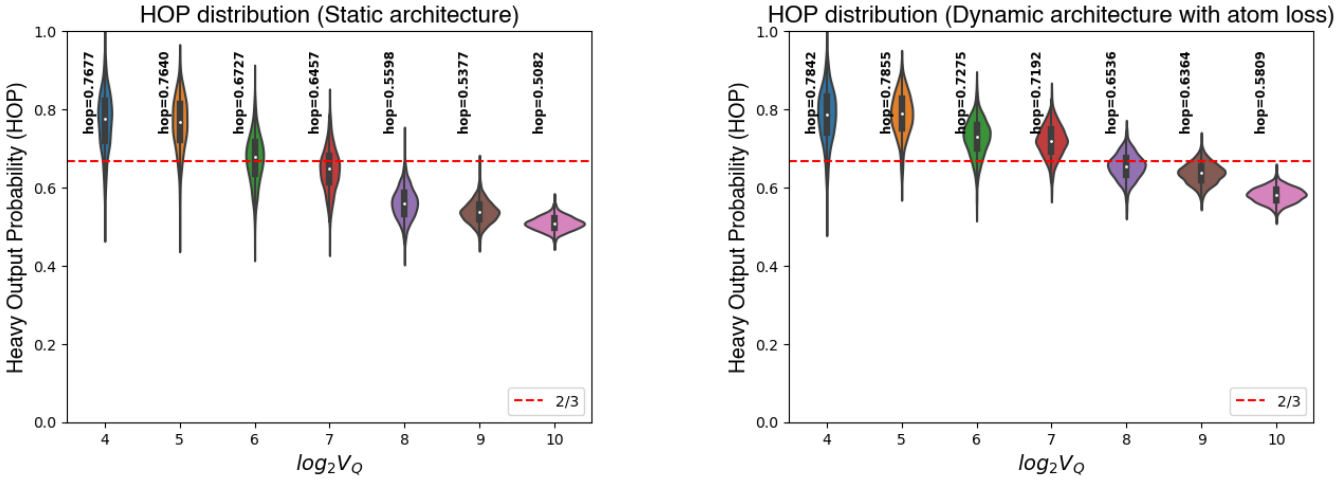
Architecture	Achievable quantum volume (V_Q)
Static architecture	2^6
Dynamic architecture w/o atom loss	2^8
Dynamic architecture w/ atom loss	2^7

To further understand the outcomes of this analysis, the distribution of heavy output probabilities for the 500 quantum volume circuits is depicted in Fig. 4.26a and Fig. 4.26b for the static and dynamic architectures (with loss), respectively. Additionally, taking into account the state-of-the-art two-qubit gate and atom move fidelities, Fig. 4.27a and Fig. 4.27b illustrate the trend in observed heavy output probabilities on a neutral atom platform.



(a) (b)

Figure 4.26: **Maximum allowed two-qubit gate error and its associated heavy output probability distribution.**
a. Static architecture. **b.** Dynamic architecture with loss.
Achieving a certain quantum volume requires passing the quantum volume test by sampling heavy outputs with a probability greater than $\frac{2}{3}$. Here, the plot shows two-qubit gate errors that are just enough to pass the test and its associated HOP distribution.



(a) (b)

Figure 4.27: **Heavy output probability distribution for current state-of-the-art two-qubit gate and atom move fidelities on neutral atom platform.**
a. Static architecture. **b.** Dynamic architecture with loss.
Considering the state-of-the-art two-qubit gate error of 0.005 and atom move infidelity of 1.8*e*-4, the static architecture's HOP drops faster than dynamic architecture.

4.8 Conclusions and outlook

In this chapter, I formalized the dynamic architecture approach for a neutral atom platform by comparing coupling and connectivity graphs of a quantum simulator. I presented a five-step protocol for implementing dynamic architecture, leveraging atom movement. Towards the end, I conducted analytical and numerical benchmarking to demonstrate that dynamic architecture outperforms static architecture with state-of-the-art two-qubit gate and atom move fidelities. I introduced a metric η_{protocol} for comparing dynamic architecture operational protocols and an inequality involving η_{platform} to compare dynamic and static architectures. Numerical results using the quantum volume metric showed that dynamic architecture achieves double the attainable quantum volume compared to static architecture. Thus, the dynamic architecture approach directly enhances the scalability of a neutral atom-based quantum simulator.

Future work will involve generalizing the five-step protocol for more complex geometries. Additionally, the compilation protocols can be extended to circuits that include single-qubit gates by recursively breaking down and merging circuits to compile for CNOT sequences within the circuit.

The error analysis for quantum volume presented in the previous section assumed a constant error rate. A more comprehensive analysis could involve sampling errors from a probability distribution, providing more realistic estimates of error thresholds. Furthermore, a more precise approximation of error thresholds can be achieved by considering additional error types, such as dephasing errors. These advancements will contribute to a more comprehensive understanding of the capabilities and limitations of dynamic architecture on a neutral atom-based quantum simulator.

Chapter 5

Closed-loop control and acquisition system

Controllability is a fundamental requirement for quantum simulators, enabling the initialization, manipulation, and measurement of quantum systems. In our experimental setup, controllability plays a central role in initializing large configurations of atoms in their ground state, manipulating atom positions to tune Rydberg-Rydberg interactions, and imaging atoms to extract the results of the quantum simulation.

The complexity of our experimental setup and the rapid timescales of the processes we study necessitate a control and acquisition system that can ensure synchronized operation across multiple hardware devices. Our control and acquisition system relies on a collection of precisely timed analog and digital modules and image acquisition hardware to meet these requirements. They interface with a broad array of hardware components distributed across multiple computing platforms. This setup also ensures the reproducibility of our measurements on the quantum simulator.

To achieve the desired level of controllability, we utilize two control systems: a slow system and a fast system. This distinction stems from the realization that different hardware components require varying levels of timing precision.

The slow system is responsible for preparing the initial state and measuring the final state of atomic qubits. It modulates the frequency and amplitude of the laser beams that are used to cool atoms in a magneto-optical trap and trap individual atoms in a multi-dimensional array of optical traps. This system ensures reliable initialization of the quantum system for subsequent experiments.

On the other hand, the fast control system enables real-time manipulation of atom positions for dynamic atom array rearrangement. This system ensures rapid movements of trapped atoms, maximizing operations within their trapping lifetimes in optical tweezers.

Furthermore, considering the importance of data traceability in our experimental studies, our control and acquisition system incorporates meticulous record-keeping. This allows us to archive all the relevant data from each experiment, including experimental parameters, raw images produced during the experiment, and any supplementary metadata in a single directory. By consolidating all the information about each experiment, the system promotes data integrity for ongoing studies and retrospective investigations.

In this chapter, I outline the control and acquisition system that enhances the controllability of our quantum simulator. Section 5.1 details the current control hardware architecture and how it integrates with the rest of the experiment's hardware. Section 5.2 focuses on the software architecture designed to conduct experiments using a software framework called Labscript suite for comprehensive experimental management. Lastly, Section 5.3 outlines the procedure undertaken to execute experiments on our neutral atom-based quantum simulator.

5.1 Hardware architecture

Our experimental setup incorporates two distinct types of control systems: a slow control system and a fast control system. These control systems serve different purposes and operate at different clock rates to meet the specific requirements of our quantum simulation experiments with neutral atoms.

The slow control system operates at a clock rate of 1 MHz. This clock rate is sufficient for the experimental time scale required for neutral atom quantum simulators. The primary role of the slow control system is to compose experiments and coordinate with all the experimental apparatus involved in our setup. It is the central hub for communication and synchronization, ensuring seamless integration of various components. The slow control system also acts as an intermediary between the fast control system and the rest of the experimental apparatus, facilitating smooth data exchange and coordination between these subsystems.

In contrast, the fast control system is integral to our reconfiguration system, designed to rearrange atoms into desired defect-free geometries for quantum simulations. Given the short lifetime of atoms in traps, it is crucial to have a low-latency reconfiguration system that can quickly and efficiently create these defect-free geometries. The fast control system

operates at a clock rate of up to 625 MHz, allowing for rapid and precise maneuvering of the atoms. By achieving compact geometries quickly, we increase the available time to perform quantum simulations.

5.1.1 Slow control system

The slow control system consists of four modules from National Instruments, all housed within the NI PXIe-1082 chassis, as illustrated in Fig. 5.1. Each of these modules is critical for the precise control and coordination of various aspects of our experimental apparatus. The ensemble includes the NI PXIe 8398 communication module, a pair of NI PXIe 6739 analog output modules, and the NI PXIe 6537 digital I/O module.

Figure 5.1: **PXIe chassis to mount digital and analog modules.**
The chassis acts as a central hub to communicate with all the mounted modules via a PCI cable connected to a desktop PC.

The first module, NI PXIe 8398 (Fig. 5.2a), serves as a communication interface between the PXI Express chassis and an external host, such as a desktop PC. It enables seamless control and interaction with the chassis from the external host, facilitating the smooth operation of the slow control system.

For analog signal generation, we use two NI PXIe 6739 analog output modules, as shown in Fig. 5.2b. Each module features 16 Analog Out (AO) banks, and every bank comprises 4 analog outputs, summing up to 64 output channels per module as shown in Fig. 5.3. For the sake of optimal performance, we utilize one channel from each AO bank, which permits an update speed of up to 1 MHz. If all channels are used simultaneously, the update speed decreases to 350 kHz. Consequently, with 16 channels utilized from each module, and considering two analog modules, we use a cumulative of 32 analog channels.

To handle digital input and output tasks, we utilize the NI PXIe 6537 digital I/O module (Fig. 5.2c). This module provides a maximum update speed of 50 MHz and offers 32 channels. It enables precise control and monitoring of digital signals, enhancing the coordination and synchronization of our experimental processes.

(a) NI PXIe 8398

(b) NI PXIe 6739

(c) NI PXIe 6537

Figure 5.2: **Digital, Analog and timing modules from National instrument.**

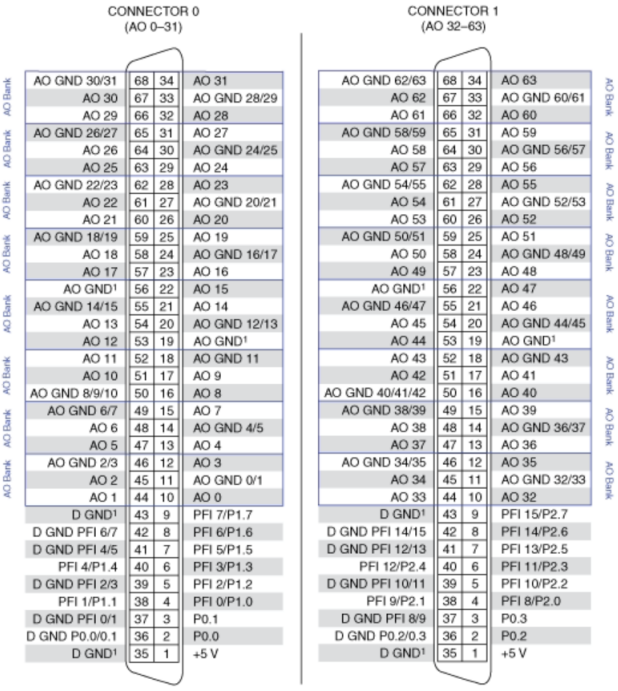


Figure 5.3: **NI PXIe 6739 analog output banks and analog output channels.** An Analog Out (AO) bank features 4 analog output channels. Utilizing all channels results in an update speed of 350 kHz while using a single channel per AO bank increases this to 1 MHz.

5.1.2 Fast control system

Our fast control system is designed to be a robust and efficient system to displace atoms and form defect-free geometries of atoms. The system encompasses five modules: Image Acquisition, Image Processing, Reconfiguration, Waveform synthesis, and Waveform Streaming. These modules work in a sequential closed loop as shown in Fig. 5.4 to arrange atoms into various geometries while keeping operational latency minimal.

After loading the atoms into optical traps, at least 50% of the optical traps are empty due to non-unity loading efficiency. A closed-loop optimization process is employed to create defect-free geometries. The closed loop starts with Image acquisition, wherein an Arbitrary Waveform Generator (AWG) triggers the EMCCD camera to capture an image at a specific timestamp. The captured image is sent to a computer via a Frame Grabber Card (FGC). The image is processed on the processor of the computer to identify

empty traps. A reconfiguration algorithm [Cimring et al., 2022] [Sabeh et al., 2022] then determines a sequence of atom moves meant to get the desired configuration. These atom moves are translated into control waveforms to be loaded and streamed on the AWG. This five-stage process repeats after all waveforms have been streamed, only ending if the desired configuration has been reached or not enough atoms are present to get the desired configuration.

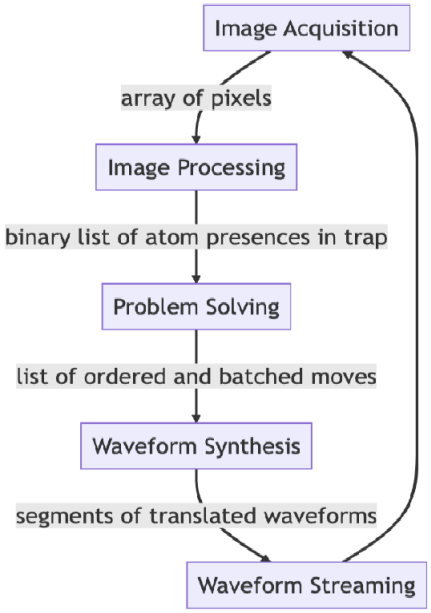


Figure 5.4: **Fast control system modules.**

5.1.3 Integrating control hardware with the experimental setup

We've designed a modular approach to integrate the control hardware with our experimental framework, ensuring efficient signal tracing. Our laboratory comprises three distinct areas: the experiment room, the control room, and the backroom (Fig. 5.5).

The experiment room houses two optical tables on which all the optical components are mounted. Underneath these tables, device-specific control hardware, including device controllers, related electronics, and BNC cables, are arranged. This organization considers the proximity of the hardware to its respective power and control inputs.

The control room is equipped with four desktop monitors, which serve as our primary interface for monitoring device states, conducting experiments, and subsequent data analysis. Additionally, a 'control room rack' is positioned in this room, enabling real-time signal measurement during experiments.

Lastly, the backroom comprises two main racks. The 'control rack' manages the generation of control signals, while the 'distribution rack' functions as the pivotal point for signal distribution throughout the setup.

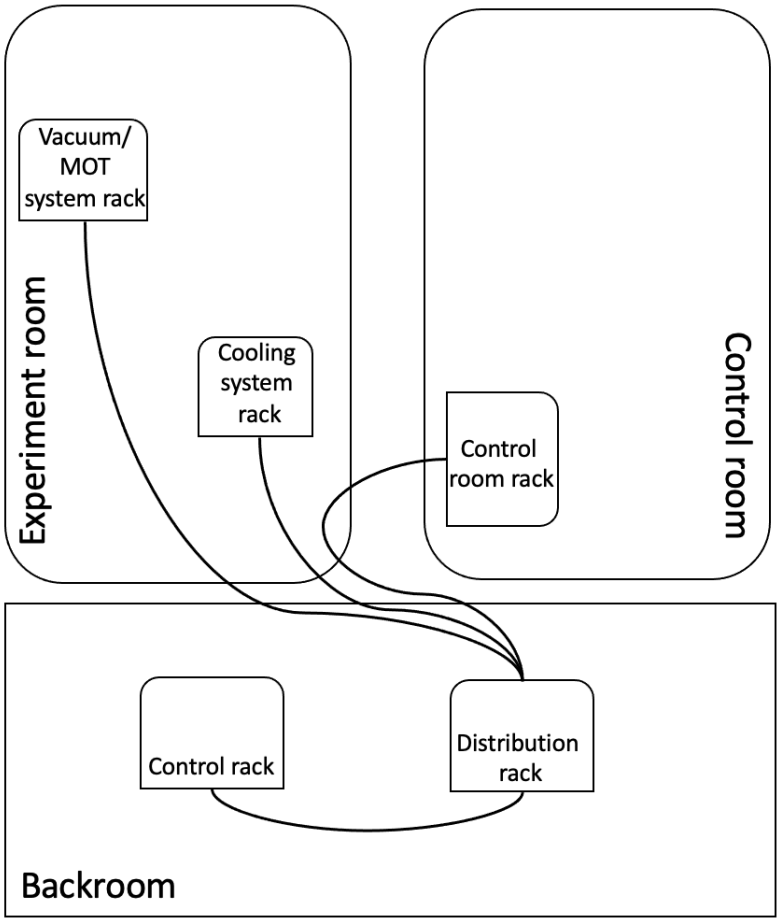


Figure 5.5: **Routing control signals to the experiment.**
The control signal distribution system is designed in a modular fashion to ensure efficient signal tracing. The 'control rack' in the backroom manages the generation of control signals, while the 'distribution rack' functions as the pivotal point for signal distribution throughout the setup.

The analog and digital signal modules are mounted on the NI PXIe-1082 chassis, as seen in Fig. 5.1. This chassis is installed at the back of the control rack in our backroom. Using BNC cables, all control signals from these modules are routed to the front patch panel of the control rack. Hence, the front patch panel of the control rack is the primary output junction for both analog and digital control signals.

The distribution rack links the control rack to different hardware components, ensuring signals are directed correctly. As shown in Fig. 5.5, the distribution rack is connected to system-specific racks like the vacuum and cooling systems that reside in the experiment room. These intermediaries then send control signals to their respective devices. This coordination is achieved through a network of BNC cables, making the distribution rack a bridge between the control system and the equipment we use in experiments.

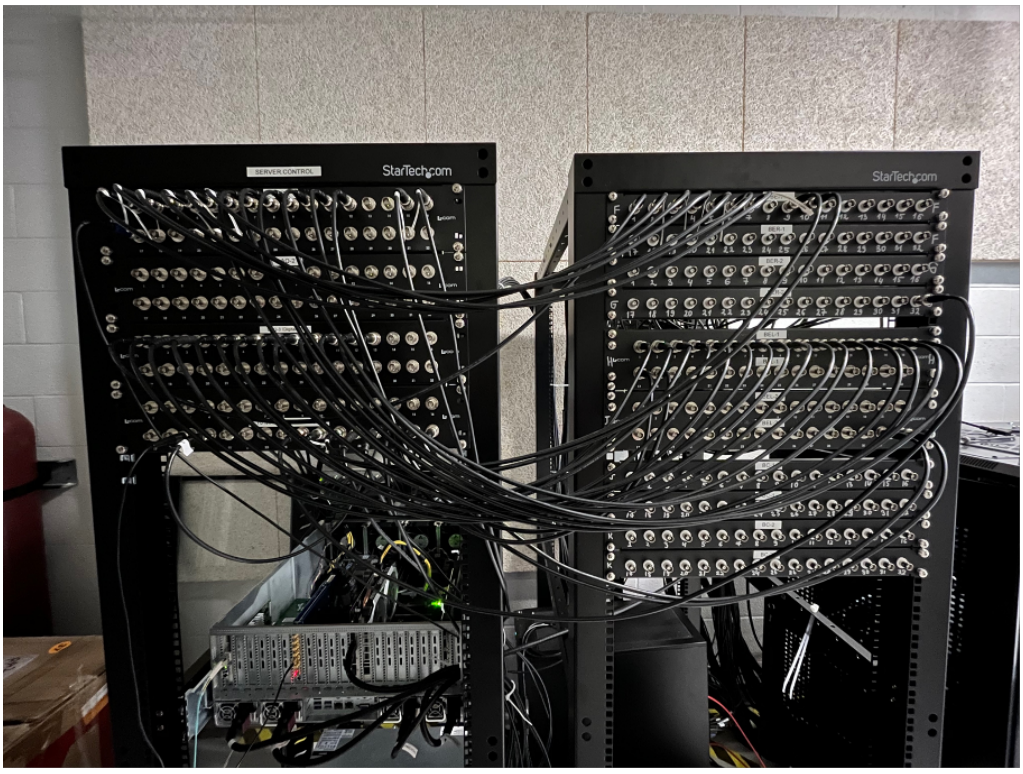


Figure 5.6: **Spatial arrangement of the control rack (left) and the distribution rack (right).**
Both racks are the core for creating and distributing control signals. Our design allows easy signal changes by swapping cables between racks, simplifying updates and troubleshooting.

The spatial configuration of these racks, with the control rack on the left and the distribution rack on the right, is shown in Fig. 5.6. Together, they form the central hub for generating and distributing the control signals. A set of BNC cables facilitates the connection between them. A testament to our modular design philosophy is the ability to add or update control signals by just switching out the short cables between the racks. This makes both updates and problem-solving easier.

The current control signal pathways are outlined in Fig. 5.7: from the control rack, through the distribution rack, to the system-specific rack, and finally to the device controller. The rows are categorized by shades, with darker rows representing digital signals and lighter ones denoting analog signals. The integration encompasses 25 digital signals, 14 analog signals, and 10 reference clocks. Providing reference clock signals to clocked devices is a good practice to synchronize all components of the experimental setup, ensuring optimal coordination. This integrated control hardware system forms the backbone of our experiment.

5.1.3.1 Integrating control hardware with the cooling system

Among the various control signals integrated with the experiment's hardware, we briefly examine the specific integration of control signals with the cooling system. This system's primary role is the cooling of Rb-87 atoms, resulting in the formation of an atomic cloud. Subsequently, atoms from this cloud are loaded into optical traps.

The two main laser components drive this system: the cooler and repumper beams. Their interaction with the atomic system is depicted in Fig. 3.4. To maximize the density of the atomic cloud, both beams' frequency and amplitude are carefully optimized.

We modulate the amplitude of cooler and repumper beams using Acousto-Optic Modulators (AOMs). The process starts with an RF synthesizer generating a constant frequency tone. This tone, at a steady power, goes through a double-balanced frequency mixer and is then amplified before reaching the AOM. To achieve amplitude modulation, we adjust the DC analog signal sent to the mixer. After the mixer, a microwave switch toggles its output, and a mechanical shutter is placed post-AOM to block the beam when necessary for the experiment. Details of the hardware components, their manufacturers, models, and control signal types are summarized in Table 5.1. Currently, our cooling system includes 7 each of AOMs, double-balance frequency mixers, microwave switches, and mechanical shutters. Each AOM is tasked with modulating the amplitude of the beams used for 2D+ MOT, 3D MOT, and PGC, each of which plays a distinct role in creating and imaging the Rb-87 atomic cloud.

Table 5.1: **Control hardware integrated with the cooling system for amplitude modulation.**

Component	Maker and model	Control signal type
Double-balanced mixer	Mini-Circuits ZFM-2-S+	Analog
Microwave switch	Mini-Circuits ZASWA-2-50DRA+	Digital
Mechanical shutter	Standford Research Systems (SRS) SR475	Digital

We modulate the frequency of cooler and repumper beams using the Moglabs ARF421 RF synthesizer. While this device supports both manual and analog modulation methods, however, we prefer the analog approach. Analog modulation ensures immediate response, which is crucial for the experiment's timing precision. Manual or software methods could introduce unpredictable delays, potentially impacting experimental outcomes.

In manual modulation, one adjusts frequency using the device's physical buttons or its software interface. Conversely, analog modulation permits continuous frequency variations by converting a -1 V to 1 V analog signal into a desired frequency range using a gain factor.

For the cooler beam, we operate within a frequency range of [70 MHz, 98.25 MHz]. With the Moglabs synthesizer's maximum permissible modulation depth of ±250MHz and a chosen gain factor of 60127707, this beam achieves a modulation depth of ±14MHz. By setting the central frequency to 84.5 MHz, we attain a sweepable range of [70.5 MHz, 98.5 MHz].

The repumper beam operates in a tighter frequency range of [82.75 MHz, 83.75 MHz]. With a selected gain of 1717934, its modulation depth is ±0.4MHz. When the central frequency is set to 83.25MHz, the achievable range is [82.85 MHz, 83.65 MHz].

System	Sub-system	Module	Device ID	Device name	Maker	Model	Device location	Control parameter	Virtual channel	NI PXIe ID	NI PXIe name	NI DAQ channel	Control rack port	Distribution rack port	Table rack port	Device controller panel
MOT	atomic source cell	current source	CS-1	atomic cell current source	Thorlabs	LDC240C	Rack R1	power	D00	3	PXIe-6537	PXI1Slot6/port0/line0	DAQ-3.1	BER-1.1	ER-5.1	Rack R1-1.1
MOT	atomic source cell	current source	CS-1	atomic cell current source	Thorlabs	LDC240C	Rack R1	current modulation	A00	1	PXIe-6739	PXI1Slot2/A00	DAQ-1.1	BER-1.2	ER-5.2	Rack R1-1.2
MOT	coils	coil driver	CC-1	compensation coil 1	Lab-made		Rack R1	Bx	A01	1	PXIe-6739	PXI1Slot2/A04	DAQ-1.2	BER-1.3	ER-5.4	Rack R1-1.4
MOT	coils	coil driver	CC-2	compensation coil 2	Lab-made		Rack R1	By	A02	1	PXIe-6739	PXI1Slot2/A08	DAQ-1.3	BER-1.4	ER-5.5	Rack R1-1.5
MOT	coils	coil driver	CC-3	compensation coil 3	Lab-made		Rack R1	Bz	A03	1	PXIe-6739	PXI1Slot2/A012	DAQ-1.4	BER-1.5	ER-5.6	Rack R1-1.6
MOT	coils	coil driver	GC-1	gradient coil 1	Lab-made		Rack R1	Gy1	A04	1	PXIe-6739	PXI1Slot2/A016	DAQ-1.5	BER-1.6	ER-5.7	Rack R1-1.7
MOT	coils	coil driver	GC-2	gradient coil 2	Lab-made		Rack R1	Gy2	A05	1	PXIe-6739	PXI1Slot2/A020	DAQ-1.6	BER-1.7	ER-5.8	Rack R1-1.8
Cooling	780 nm laser	Mixing board	MI.Switch-1	AOM driver switch 1	Mini-Circuits	ZASWA-2-50DRA+	Rack L1	cooler 3DMOT	D01	3	PXIe-6537	PXI1Slot6/port0/line1	DAQ-3.2	BEL-1.1	EL-2.1	Rack L1-2.9
Cooling	780 nm laser	Mixing board	MI.Switch-2	AOM driver switch 2	Mini-Circuits	ZASWA-2-50DRA+	Rack L1	cooler 2DMOT	D02	3	PXIe-6537	PXI1Slot6/port0/line2	DAQ-3.3	BEL-1.2	EL-2.2	Rack L1-2.10
Cooling	780 nm laser	Mixing board	MI.Switch-3	AOM driver switch 3	Mini-Circuits	ZASWA-2-50DRA+	Rack L1	cooler PGC	D03	3	PXIe-6537	PXI1Slot6/port0/line3	DAQ-3.4	BEL-1.3	EL-2.3	Rack L1-2.11
Cooling	780 nm laser	Mixing board	MI.Switch-4	AOM driver switch 4	Mini-Circuits	ZASWA-2-50DRA+	Rack L1		D04	3	PXIe-6537	PXI1Slot6/port0/line4	DAQ-3.5	BEL-1.4	EL-2.4	Rack L1-2.12
Cooling	780 nm laser	Distribution board	MI.Switch-5	AOM driver switch 5	Mini-Circuits	ZASWA-2-50DRA+	Rack L1	repumper main	D05	3	PXIe-6537	PXI1Slot6/port0/line5	DAQ-3.6	BEL-1.5	EL-2.5	Rack L1-2.13
Cooling	780 nm laser	Distribution board	MI.Switch-6	AOM driver switch 6	Mini-Circuits	ZASWA-2-50DRA+	Rack L1	repumper 3DMOT	D06	3	PXIe-6537	PXI1Slot6/port0/line6	DAQ-3.7	BEL-1.6	EL-2.6	Rack L1-2.14
Cooling	780 nm laser	Distribution board	MI.Switch-7	AOM driver switch 7	Mini-Circuits	ZASWA-2-50DRA+	Rack L1	repumper 2DMOT	D07	3	PXIe-6537	PXI1Slot6/port0/line7	DAQ-3.8	BEL-1.7	EL-2.7	Rack L1-2.15
Cooling	780 nm laser	Distribution board	MI.Switch-8	AOM driver switch 8	Mini-Circuits	ZASWA-2-50DRA+	Rack L1	repumper PGC	D08	3	PXIe-6537	PXI1Slot6/port1/line0	DAQ-3.9	BEL-1.8	EL-2.8	Rack L1-2.16
Cooling	780 nm laser	Mixing board	Me.Shutter-1	Mechanical shutter 1	SRS	SR475	Rack L1	cooler 3DMOT	D09	3	PXIe-6537	PXI1Slot6/port1/line1	DAQ-3.10	BEL-1.9	EL-3.1	Rack L1-1.1
Cooling	780 nm laser	Mixing board	Me.Shutter-2	Mechanical shutter 2	SRS	SR475	Rack L1	cooler 2DMOT	D010	3	PXIe-6537	PXI1Slot6/port1/line2	DAQ-3.11	BEL-1.10	EL-3.2	Rack L1-1.2
Cooling	780 nm laser	Mixing board	Me.Shutter-3	Mechanical shutter 3	SRS	SR475	Rack L1	cooler PGC	D011	3	PXIe-6537	PXI1Slot6/port1/line3	DAQ-3.12	BEL-1.11	EL-3.3	Rack L1-1.3
Cooling	780 nm laser	Mixing board	Me.Shutter-4	Mechanical shutter 4	SRS	SR475	Rack L1		D012	3	PXIe-6537	PXI1Slot6/port1/line4	DAQ-3.13	BEL-1.12	EL-3.4	Rack L1-1.4
Cooling	780 nm laser	Distribution board	Me.Shutter-5	Mechanical shutter 5	SRS	SR475	Rack L1	repumper 3DMOT	D013	3	PXIe-6537	PXI1Slot6/port1/line5	DAQ-3.14	BEL-1.13	EL-3.5	Rack L1-1.5
Cooling	780 nm laser	Distribution board	Me.Shutter-6	Mechanical shutter 6	SRS	SR475	Rack L1	repumper 2DMOT	D014	3	PXIe-6537	PXI1Slot6/port1/line6	DAQ-3.15	BEL-1.14	EL-3.6	Rack L1-1.6
Cooling	780 nm laser	Distribution board	Me.Shutter-7	Mechanical shutter 7	SRS	SR475	Rack L1	repumper PGC	D015	3	PXIe-6537	PXI1Slot6/port1/line7	DAQ-3.16	BEL-1.15	EL-3.7	Rack L1-1.7
Cooling	780 nm laser	Distribution board	Me.Shutter-8	Mechanical shutter 8	SRS	SR475	Rack L1	push beam	D016	3	PXIe-6537	PXI1Slot6/port2/line0	DAQ-3.17	BEL-1.16	EL-3.8	Rack L1-1.8
Cooling	780 nm laser	Single generator	AWG-1_trigger	AWG-1_trigger	Spectrum	M4i.6622-x8		sweep trigger	D017	3	PXIe-6537	PXI1Slot6/port2/line1	DAQ-3.18			
Cooling	780 nm laser	Mixing board	Mixer-1	AOM driver mixer 1	Mini-Circuits	ZFM-2-S+	Rack L1	cooler 3DMOT	A06	1	PXIe-6739	PXI1Slot2/A024	DAQ-1.7	BEL-1.17		Rack L1-1.9
Cooling	780 nm laser	Mixing board	Mixer-2	AOM driver mixer 2	Mini-Circuits	ZFM-2-S+	Rack L1	cooler 2DMOT	A07	1	PXIe-6739	PXI1Slot2/A028	DAQ-1.8	BEL-1.18		Rack L1-1.10
Cooling	780 nm laser	Mixing board	Mixer-3	AOM driver mixer 3	Mini-Circuits	ZFM-2-S+	Rack L1	cooler PGC	A08	1	PXIe-6739	PXI1Slot2/A032	DAQ-1.9	BEL-1.19		Rack L1-1.11
Cooling	780 nm laser	Mixing board	Mixer-4	AOM driver mixer 4	Mini-Circuits	ZFM-2-S+	Rack L1	master	A09	1	PXIe-6739	PXI1Slot2/A036	DAQ-1.10	BEL-1.20		Rack L1-1.12
Cooling	780 nm laser	Distribution board	Mixer-5	AOM driver mixer 5	Mini-Circuits	ZFM-2-S+	Rack L1	repumper main	A010	1	PXIe-6739	PXI1Slot2/A040	DAQ-1.11	BEL-1.21		Rack L1-1.13
Cooling	780 nm laser	Distribution board	Mixer-6	AOM driver mixer 6	Mini-Circuits	ZFM-2-S+	Rack L1	repumper 3DMOT	A011	1	PXIe-6739	PXI1Slot2/A044	DAQ-1.12	BEL-1.22		Rack L1-1.14
Cooling	780 nm laser	Distribution board	Mixer-7	AOM driver mixer 7	Mini-Circuits	ZFM-2-S+	Rack L1	repumper 2DMOT	A012	1	PXIe-6739	PXI1Slot2/A048	DAQ-1.13	BEL-1.23		Rack L1-1.15
Cooling	780 nm laser	Distribution board	Mixer-8	AOM driver mixer 8	Mini-Circuits	ZFM-2-S+	Rack L1	repumper PGC	A013	1	PXIe-6739	PXI1Slot2/A052	DAQ-1.14	BEL-1.24		Rack L1-1.16
Imaging	low-resolution imaging	CMOS camera	CMOS-1	CMOS 2DMOT	Basler	a2A3840-45umBAS		acquisition trigger	D018	3	PXIe-6537	PXI1Slot6/port2/line2	DAQ-3.19	BER-1.8	ER-4.1	
Imaging	low-resolution imaging	CMOS camera	CMOS-2	CMOS 3DMOT1	Basler	a2A3840-45umBAS		acquisition trigger	D019	3	PXIe-6537	PXI1Slot6/port2/line3	DAQ-3.20			
Imaging	low-resolution imaging	CMOS camera	CMOS-3	CMOS 3DMOT2	Basler	a2A3840-45umBAS		acquisition trigger	D020	3	PXIe-6537	PXI1Slot6/port2/line4	DAQ-3.21			
Imaging	low-resolution imaging	CMOS camera	CMOS-4	CMOS closed loop 1	Basler	a2A3840-45umBAS		acquisition trigger	D021	3	PXIe-6537	PXI1Slot6/port2/line5	DAQ-3.22			
Imaging	low-resolution imaging	CMOS camera	CMOS-5	CMOS closed loop 2	Basler	a2A3840-45umBAS		acquisition trigger	D022	3	PXIe-6537	PXI1Slot6/port2/line6	DAQ-3.23			
Imaging	high-resolution imaging	EMCCD camera	Me.Shutter-9	EMCCD external shutter	Thorlabs	SH8025T		shutter	D023	3	PXIe-6537	PXI1Slot6/port2/line7	DAQ-3.24	BER-1.12	ER-1.1	
Imaging	high-resolution imaging	EMCCD camera	Me.Shutter-10	EMCCD	Andor			acquisition trigger	D024	3	PXIe-6537	PXI1Slot6/port3/line0	DAQ-3.25	BER-1.13	ER-1.2	

System	Sub-system	Module	Device ID	Device name	Maker	Model	Device location
Control	Signal generator	NIDAQ	NI Chassis	NI Chassis	NI	PXIe 1082	lackroom control rack
Control	Signal generator	AWG	AWG-1	AWG 1	Spectrum	M4i.6622-x8	Control server
Control	Signal generator	AWG	AWG-2	AWG 2	Spectrum	M4i.6622-x8	Control server
Control	Signal generator	AWG	AWG-3	AWG 3	Spectrum	M4i.6621-x4	AOD workstation
Control	Signal generator	AWG	AWG-4	AWG 4	Spectrum	M4i.6621-x4	EMCCD workstation
Cooling	780 nm laser	RF synthesizer	RF synthesizer 1	USB RF synthesizer 1	Quonset	QM2010-4400	Rack L1
D1		RF synthesizer	RF synthesizer 2	USB RF synthesizer 2	Quonset	QM2010-4400	Right table
D1		microwave synthesizer			NI	FSW-0020	Right table
Rydberg	cavity	cavity	SLS cavity	SLS cavity	SLS		Right table
Control	Labscript suite	Pseudoclock	Pseudoclock-1				

Control rack port	Distribution rack port	Table rack port	Device controller panel
Clock-1.1			
Clock-1.2			
Clock-1.3			
Clock-1.4			
Clock-1.5	BEL-1.31	EL-1.7	
Clock-1.6	BEL-1.32	EL-1.8	
Clock-1.7	BER-1.31	ER-1.7	
Clock-1.8	BER-1.32	ER-1.8	
Clock-1.9	BER-1.30	ER-4.8	
Clock-1.10			

Figure 5.7: **A detailed list of control signals distributed to various hardware and their associated cable paths.**

5.2 Software architecture

5.2.1 Labscript suite

Labscript suite is a software package developed by Philip Starkey and his colleagues during his PhD at Monash University [P.T. Starkey, 2019]. This software was specifically designed for ultra-cold atom experiments and has since become a widely used tool. Labscript suite is an open-source code repository hosted on Github (link), where researchers from around the world contribute to its continuous development and expansion.

One of the significant advantages of the labscript suite is its flexibility, allowing researchers to integrate customized functionalities into their experiments. This capability has led to the development of additional tools and extensions that complement the core functionality of the labscript suite. Its user-friendly interface, extensive documentation, and active community support make it accessible to both novice and experienced researchers.

5.2.2 Components and technologies in labscript suite

In our lab, we rely on five key components of the labscript suite to streamline the experimental processes. These components, namely the labscript API, runmanager, runviewer, BLACS[1], and lyse work together seamlessly to facilitate the execution and analysis of experimental shots. Each component features a graphical interface, except for the labscript API. The role of each component is described briefly in Table 5.2.

The communication between components is established through a ZeroMQ (discussed in Section 5.2.2.8) network socket, ensuring error-free data exchange. The communicated data is generally the path of the HDF5 experimental shot file. We look at each technology and component in detail.

[1]Originally B.L.A.C.S. was an acronym standing for the 'BEC lab apparatus control system'. This was later updated to the 'better lab apparatus control system' when it became clear the software could be generalized to other experiments before being dropped in favor of just using 'BLACS' as a name. [P.T. Starkey, 2019]

Table 5.2: **Components of the Labscript suite and their applications.**

Component	Application
Labscript API	Composing the desired control sequence to frame an experiment
Runmanager	Compiling the control sequence into an HDF5 experimental shot
Runviewer	Visualising the control sequence in an HDF5 experimental shot
BLACS	Executing the control sequence in an HDF5 experimental shot
Lyse	Analysing the data collected during the experiment

5.2.2.1 Labscript API: Compose and define experimental shots

The labscript API serves as a tool for choreographing the experiment. A script defining the input and output hardware devices involved in an experiment and the connections between them is called a connection table. The connection table used in our laboratory is outlined in Appendix A.1. Python objects representing each hardware device are created, which are later used in experimental logic.

The experimental logic section is where users specify the desired state of each output at different times throughout the experiment. The Labscript API provides methods for manipulating these objects, depending on the type of Python object. It can be input/output of type analog or digital. The API generates the necessary instructions for the pseudoclock(s), which are responsible for managing the timing of each hardware device. An exemplary experimental sequence used in our lab to image 3DMOT is shown in Appendix A.2.

5.2.2.2 Runmanager: Compile experimental shots

The Runmanager component of the labscript suite serves as a Graphical User Interface (GUI) that primarily focuses on defining and managing global parameters for experiments. It also plays a key role in compiling the experimental logic written using the labscript

API into an HDF5 file format. This file contains a tabular representation of the hardware sequence and is further discussed in Section 5.2.2.6.

When using Runmanager, the documentation recommends grouping global parameters based on the types of experimental sequences that will be executed. For example, in our lab, we group parameters as static and dynamic parameters for creating 2DMOT and 3DMOT. This grouping enables the enable/disable functionality of parameters, providing modularity to the experiment. As our experiments progress, additional groups of parameters will be added to define control parameters for trapping and imaging atoms in tweezers.

One notable feature of Runmanager is its ability to automate the traversal of parameter space. By defining a global parameter as a list, Runmanager automatically generates multiple shots by taking the outer product of all the lists. It also allows for random shuffling of parameters when necessary. Further details on unpacking global parameters can be found in work by Starkey [P.T. Starkey, 2019].

5.2.2.3 Runviewer: Visualize experimental shots

The Runviewer component of the labscript suite serves as a tool for visualizing the hardware instructions at each channel during an experiment. By parsing the HDF5 file generated by Runmanager, Runviewer creates informative time-based plots that accurately represent the behavior of each output channel. This visualization is particularly useful when control pulses are abstracted using for-loops or when trying to understand the shape of complex ramps utilized in the experiment. It also enables comparisons between the expected output displayed in the tool and the observed output on an oscilloscope. This feature can significantly expedite the process of debugging hardware issues, as any discrepancies between the expected and observed outputs can be easily identified and investigated.

5.2.2.4 BLACS: Execute experimental shots

The BLACS component serves as an interface between the labscript suite and hardware devices. It is responsible for managing device handlers and serially executing experimental shots (as discussed in Section 5.2.2.6). These shots, stored as HDF5 files, contain hardware instructions for each channel, which BLACS parses and interprets. The two most used modes of operation on BLACS are manual and buffered modes. In manual mode, BLACS provides a graphical user interface (GUI) that enables users to control device outputs, facilitating the debugging of hardware behavior. On the contrary, in buffered mode, as the name implies, the control sequence is buffered within the hardware device. Upon the tick

of the pseudoclock, the buffered sequence is executed, resulting in changes to the device outputs. If configured accordingly, BLACS also sends software triggers to the camera servers (discussed in Section 5.2.2.7), signaling that an experimental shot is about to be executed.

5.2.2.5 Lyse: Analyse experimental shots

The Lyse component enables real-time analysis of the acquired data. Once the execution of an experimental shot is completed in BLACS, the corresponding HDF5 file is passed on to Lyse. Depending on the implementation, the HDF5 file may either contain the actual images or the path to the captured images acquired during the experiment. By leveraging the readily available data in the HDF5 file, lyse facilitates closed-loop parameter space exploration, as shown in Section 5.3.5. This is achieved by incorporating experiment-specific Python analysis files into lyse. Lyse supports both single-shot analysis and multi-shot analysis. For instance, in single-shot analysis, lyse performs calculations such as computing the integrated intensity of the captured image. The resulting processed information is then stored in shot-specific HDF5 files for future reference. On the other hand, multi-shot analysis involves extracting relevant values from each shot-specific HDF5 file and generating plots that illustrate the variation of integrated intensity with different parameters.

5.2.2.6 Experimental shots: Comprehensive record of the experiment

Following the terminology of architects of the labscript suite, an individual experimental shot refers to a specific control sequence that is executed in the experiment. An experiment is a sequence of shots with varying shot-to-shot parameters. It is important to note that it is generally advised not to vary parameters within a single shot. Instead, the documentation of the Labscript suite recommends creating a sequence of shots, each with its own set of varying parameters. This approach ensures better experimental control and facilitates systematic exploration of parameter space.

The result of compiling a control sequence script on runmanager is an HDF5 file (experimental shot file) that serves as a comprehensive record of the experimental setup. This HDF5 file contains information, including global parameters, the control sequence applied to each channel, and the underlying script itself. By encapsulating these details, the HDF5 file provides a consolidated representation of the experiment. In addition to the experimental parameters and control sequence, the HDF5 file can also be configured to

store images captured during the execution of the shot. This capability allows for a more comprehensive record of the experimental process. The HDF5 file plays a central role in facilitating communication and data exchange between the runmanager, BLACS, and lyse components. These components interact by passing the HDF5 files, allowing for seamless integration and analysis of experimental data.

The runmanager component, with its ability to compile the experimental shot file and generate the HDF5 output, serves as the initial step in the workflow. It captures all the relevant information about the experiment, providing a structured and organized format for further processing.

BLACS, as the hardware interface component, utilizes the HDF5 file to access the necessary control sequences and parameters defined in runmanager. This enables BLACS to effectively communicate with the experimental apparatus and execute the desired operations in a synchronized manner.

Finally, the analysis component, Lyse, leverages the HDF5 files to import the experimental data and associated parameters. This allows for efficient and consistent analysis of the data, as the required information is readily available within the HDF5 file structure.

5.2.2.7 Camera server: Acquire images synchronously

The Camera server is critical to synchronize control and acquisition processes, mainly because both processes run on different computers. The camera server is first initialized on the acquisition computer, which initializes the associated camera and sets its static and acquisition properties. The control computer, on the other hand, acts as a client-to-camera server. The connection between these two computers is established by providing the BIAS port number of the camera server to the control computer. Once set, every time BLACS transfers to buffered mode, a set of instructions defined in the camera server script is executed, which may include asking the camera to wait for a finite number of triggers and setting up the image-saving path following the experimental shot path on a network shared drive. It also permits adding instructions when BLACS transfers to static mode, enabling functionalities like stopping waiting for the image acquisition trigger and saving images at the defined locations. Having these functionalities makes it easier to control and organize image acquisition and saving on a shot-to-shot basis. The modified camera server script used in our laboratory is outlined in Appendix A.3.4. In addition to its primary functions, we also use the camera server's software-triggering capability to gather data from auxiliary monitoring devices, such as the Raspberry Pi. This allows us to record experimental conditions, including parameters like laser power.

5.2.2.8 ZeroMQ: Network socket for communication between labscript suite components

ZeroMQ is an open-source package link that plays an essential role in managing inter-process communication. Its primary function is to facilitate the sharing of HDF5 experimental shot file paths among the components of the labscript suite. Moreover, it ensures secure access to these HDF5 experimental shot files when multiple processes attempt to access them concurrently. Since concurrent access attempts can lead to file corruption, ensuring serialized access becomes crucial. As a solution, the architects of Labscript suite monkey-patched [2] ZeroMQ onto the h5py library. This results in the serialization of HDF5 shot access by generating a semaphore, which effectively manages the file access process.

5.2.2.9 Unit conversion class: Control signals mapped to physical values

Unit conversion classes streamline the work with hardware devices by offering customized parameter mapping for each hardware device. Labscript API functions typically expect the inputs in the form of SI units. However, these values control a different physical quantity, such as current, frequency, or magnetic field strength. Writing experimental logic directly in base units can sometimes lead to errors. Unit conversion classes tailored for each hardware device are employed to circumvent this issue. These classes automate the conversion process between the device's physical and control values, helping avoid mistakes and increase efficiency. Two examples of unit conversion classes are given in Appendix A.3.1 and Appendix A.3.2.

5.3 Software workflow

The complete software workflow established to conduct quantum simulations using a neutral atom platform can be divided into three steps: experiment composition, experiment execution, and experiment analysis. These steps, each incorporating various components and technologies, are comprehensively depicted in Table 5.3.

Our software architecture is deliberately modular and structured around these essential steps. This segmentation enhances system maintainability, encourages code re-usability, and facilitates potential future enhancements and adaptations.

[2] a technique to alter code behavior at run-time by adding or modifying object attributes or methods using external code, potentially diverging from the object's original design.

The first step, 'experiment composition', involves formulating the experimental setup and scripting the experimental procedure using components and technologies from the Labscript suite. This is the stage where the main body of the quantum simulation is defined.

During the subsequent 'experiment execution' step, we employ various technologies, including message queuing, camera servers, and device handlers, to execute the experimental sequence. This process ensures a seamless transition from theoretical design to practical execution.

The concluding step, 'experiment analysis', encompasses evaluating and interpreting the data generated from the execution phase. We extract insights from the data using various analytical software tools and produce actionable information for future experiments.

A pivotal aspect of our software workflow is incorporating M-LOOP, a machine learning technique detailed in Section 5.3.5. Integrating machine learning with our modular architecture automates the decision-making process for selecting and navigating the control parameter space, resulting in a closed-loop optimization process. This iterative refinement and high level of integration are essential to conduct experiments efficiently.

In the following sections, we delve deeper into each step, elaborating on their specific roles, the technologies employed, and the collective function in facilitating quantum simulations using a neutral atom platform.

Table 5.3: **Software components and technologies used at various steps of the experiment.**

Experimental Step	**Labscript suite components**					**Labscript suite technologies**			
	API	Run manager	Run viewer	BLACS	Lyse	Shot file	ZeroMQ	Camera Server	Unit conversion
Composition	✓	✓	✓			✓		✓	✓
Execution				✓		✓	✓	✓	
Analysis					✓	✓	✓		

5.3.1 Experiment composition

The experiment composition step consists of three key sub-steps: creating the connection table, scripting the experiment, and compiling the experiment.

Creating the connection table: The first sub-step of experiment composition involves the creation of a connection table. This table comprehensively lists and maps the physical connections between various devices used in the experiment. It delineates the communication channels of the devices, providing a clear map of the device network. Additionally, during this stage, the unit conversion class for each hardware device is scripted and incorporated into the connection table script (examples of unit conversion classes used in our lab are provided in Appendix A.3.1 and Appendix A.3.2). This inclusion ensures that any physical parameter inputs are automatically translated into corresponding control parameter values. This table serves as a central reference for the remaining stages, ensuring that each element of the experiment understands its role and interaction with other components. An instance of such a connection table, as used in our laboratory and crafted using the Labscript API, is provided in Appendix A.1 for reference.

Scripting the experiment: Once the connection table is established, the experiment's core procedure is scripted using Labscript API. This scripting involves formulating sequences of operations, specifying the time-based control of hardware devices, and defining the desired measurements. Given the Pythonic nature of the Labscript API, it allows for sophisticated control pulse scripting, facilitated by the use of common programming structures such as for and while loops.

Within this scripting stage, we also program the camera server on the acquisition system (an example of a camera server used in our lab is provided in Appendix A.3.4). The server is programmed to wait for hardware triggers during the experiment execution. This enables the synchronized capture of images precisely timed with the experiment's crucial moments. An instance of how this technique is used to image a cloud of atoms in our laboratory, crafted using the Labscript API, is illustrated in Appendix A.2.

Compiling the experiment: The final sub-step involves compiling the scripted experiment via Runmanager. This process translates the high-level language of the experiment script into a sequence of low-level instructions, which are subsequently written into an HDF5 file, also referred to as the 'experimental shot file'. As shown in Fig. 5.8, Runmanager serves a critical role in maintaining and managing the global variables of the experiment, which are essential in formulating the experimental procedure.

Runmanager extends its function to generate multiple experimental shots when a variable is a list type. Each value within the list corresponds to a unique experimental shot, allowing for an expansive control parameter space exploration. Moreover, Runmanager is designed to facilitate these lists' inner and outer products, enabling linear exploration over the parameter space. For instance, consider the global variables 'moglabs_opll_cooler_freq' and 'moglabs_opll_repumper_freq' shown in Fig. 5.8. These variables are of type list

with 10 elements, and all the dependent variables are computed as inner products of these lists. This results in a total of 100 experimental shots obtained by taking the outer product of the two lists.

Reviewing the compiled experiment is critical to ensure the control hardware transmits the intended control signals. The Runviewer component of the Labscript suite serves a vital role in debugging the control signals. It graphically represents the control signals over time, providing a virtual oscilloscope output. This allows to validate and adjust the signals before executing the experiment.

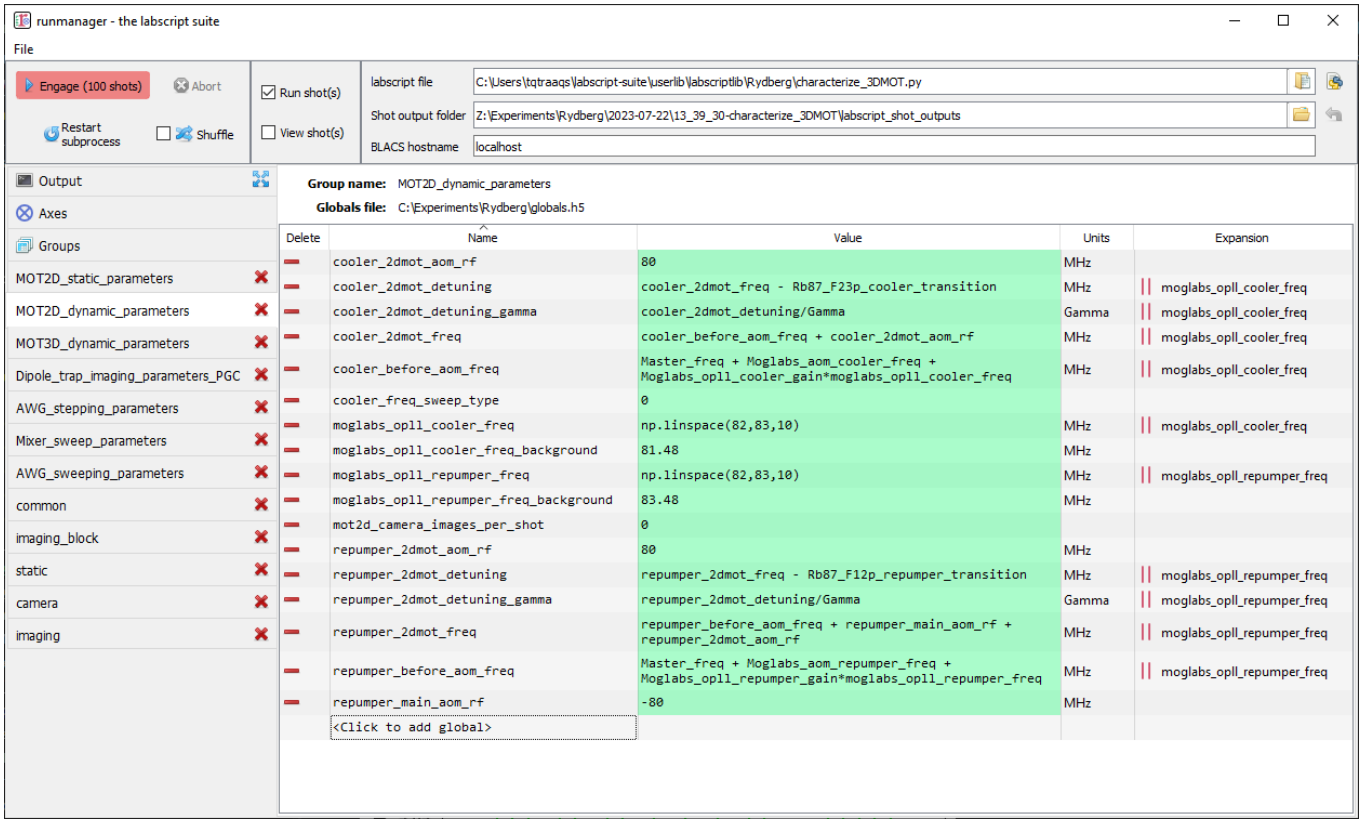


Figure 5.8: **Runmanager GUI compiles experimental shots.**

5.3.2 Experiment execution

The compiled experimental shots are stored on a shared network drive and are automatically transferred to BLACS. BLACS serves as an interface between the user and the

experimental hardware by managing all the device handlers. For instance, in Fig. 5.9, BLACS occupies the device handlers of four hardware devices as instructed in the connection table. It also manages a queue of experimental shots and executes them sequentially. When executing a shot, BLACS switches to 'buffered_mode'; after execution, it transitions back to 'manual_mode'. While in 'buffered_mode', manual modifications to the hardware channels are not allowed. The camera server saves the images captured during each shot in the same directory as the experimental shot on the shared network drive upon completion of the shot.

Furthermore, BLACS automatically places the shot in the Lyse queue if the Lyse address is configured correctly. In the example shown in Fig. 5.9, BLACS sends the shot to another computing device with the hostname 'rack-server'. If any interruption occurs during the experiment, BLACS outputs the error in the respective output of the hardware tabs. The connection table names mapped to hardware channels are imported into the BLACS GUI, facilitating easy manual control for debugging.

5.3.3 Experiment analysis

Experimental analysis in our setup can be done either on Lyse or independently by importing the HDF5 files containing control parameters and acquired raw data. Lyse offers the advantage of parsing these files into Pandas data frames, which can be accessed by analysis scripts promoting code re-usability. Lyse categorizes analysis into single-shot and multi-shot routines, as shown in Fig. 5.10. Customized analysis scripts can be added to Lyse as per experimental needs, and storing analysis results in the same HDF5 file promotes data integrity and traceability.

In our lab, we perform real-time single-shot analysis, storing the results in the experimental shot file. Multi-shot analysis, on the other hand, is done offline to extract meaningful trends from parameter sweeps. In the following section, we explain how M-LOOP integration automates decision-making for control parameter selection, minimizing human intervention in conducting experiments.

5.3.4 Experiment workflow

The execution and management of our experiments predominantly occur on a dedicated 'control' computer, which serves as the central hub for running Labscript API and Runmanager. The experimental sequences written with Labscript API are compiled to generate shot files stored on a shared network drive. Each experiment can produce multiple such

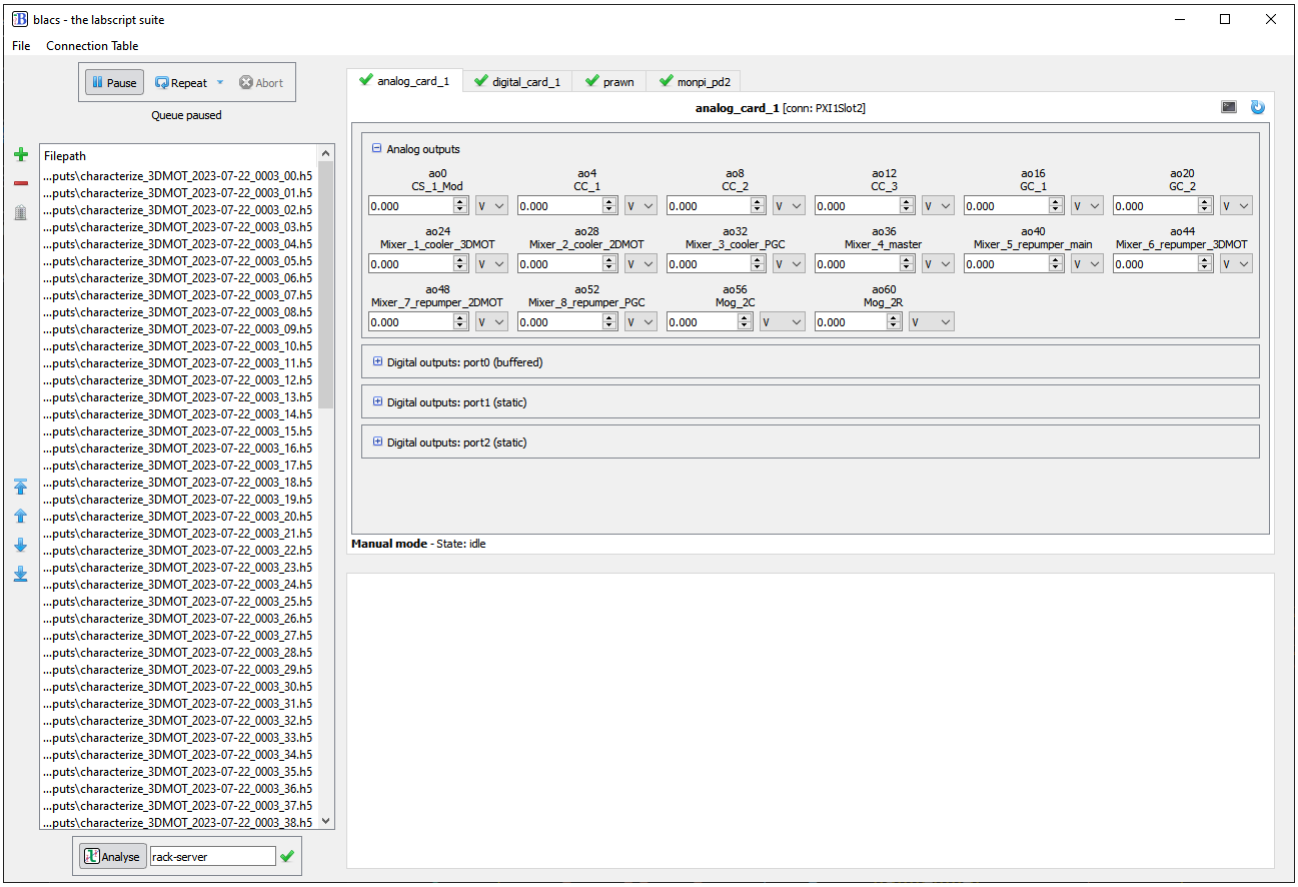


Figure 5.9: **BLACS GUI manages and executes a queue of experimental shots.**

shots, and all of them are stored on this network drive. The experiment is visualized using Runviewer to gain intuition into the sequence and debug any potential errors.

Once these experimental shot files are available on the shared network drive, the acquisition system, comprising a computer connected to a camera, comes into play. This system parses the shot files, extracting information regarding the number of images to capture for each shot. It then instructs the camera to await a specific number of hardware triggers.

The execution of the experimental sequence is initiated from the 'control' computer, which is equipped with BLACS. While running Runmanager and BLACS on different computers is possible, our current practice involves composing and executing experiments from the same 'control' computer.

Finally, as the experimental sequence concludes, the collected data and images are sent

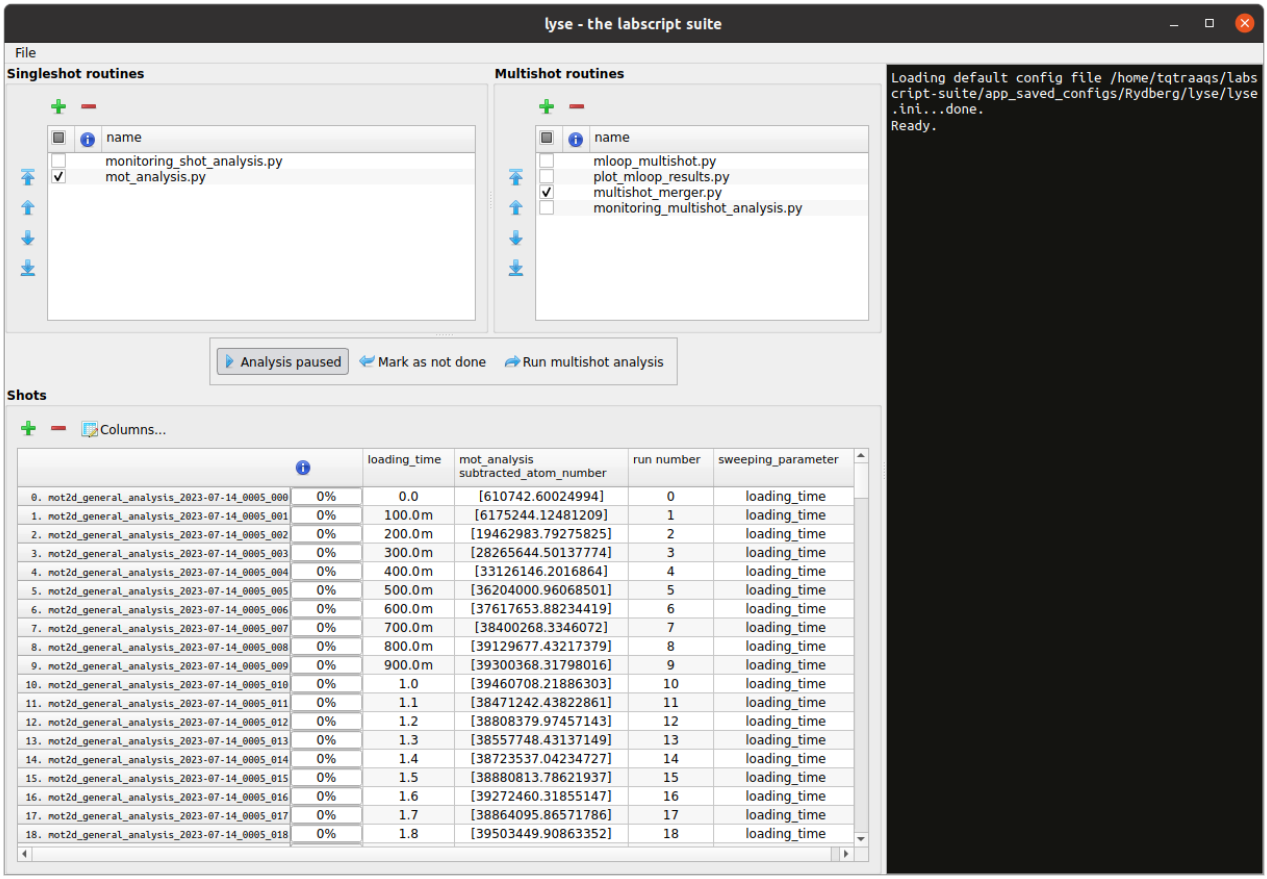


Figure 5.10: **Lyse GUI manages analysis routines and a queue of executed experimental shots.**

to Lyse, where in-depth analysis is performed. The complete workflow of our experiments can be summarized with Fig. 5.11 and Fig. 5.12.

The components of the Labscript suite — Runmanager, Runviewer, BLACS, and Lyse — have the capability to operate on separate computers. This adaptability proves advantageous when tailoring the software distribution in alignment with our experimental hardware. In our specific laboratory configuration, as illustrated in Fig. 5.12, we mainly use three computers: 'control', 'imaging', and 'analysis'. Data sharing among these systems is efficiently facilitated through a network-shared drive, employing the Labscript suite's ZeroMQ protocol for fetching and storing experimental shot files. In the current state, the additions and modifications done in implementing the Labscript suite are described in Appendix A.3. The bugs faced while integrating the Labscript suite with our experiment are documented in Appendix A.4.

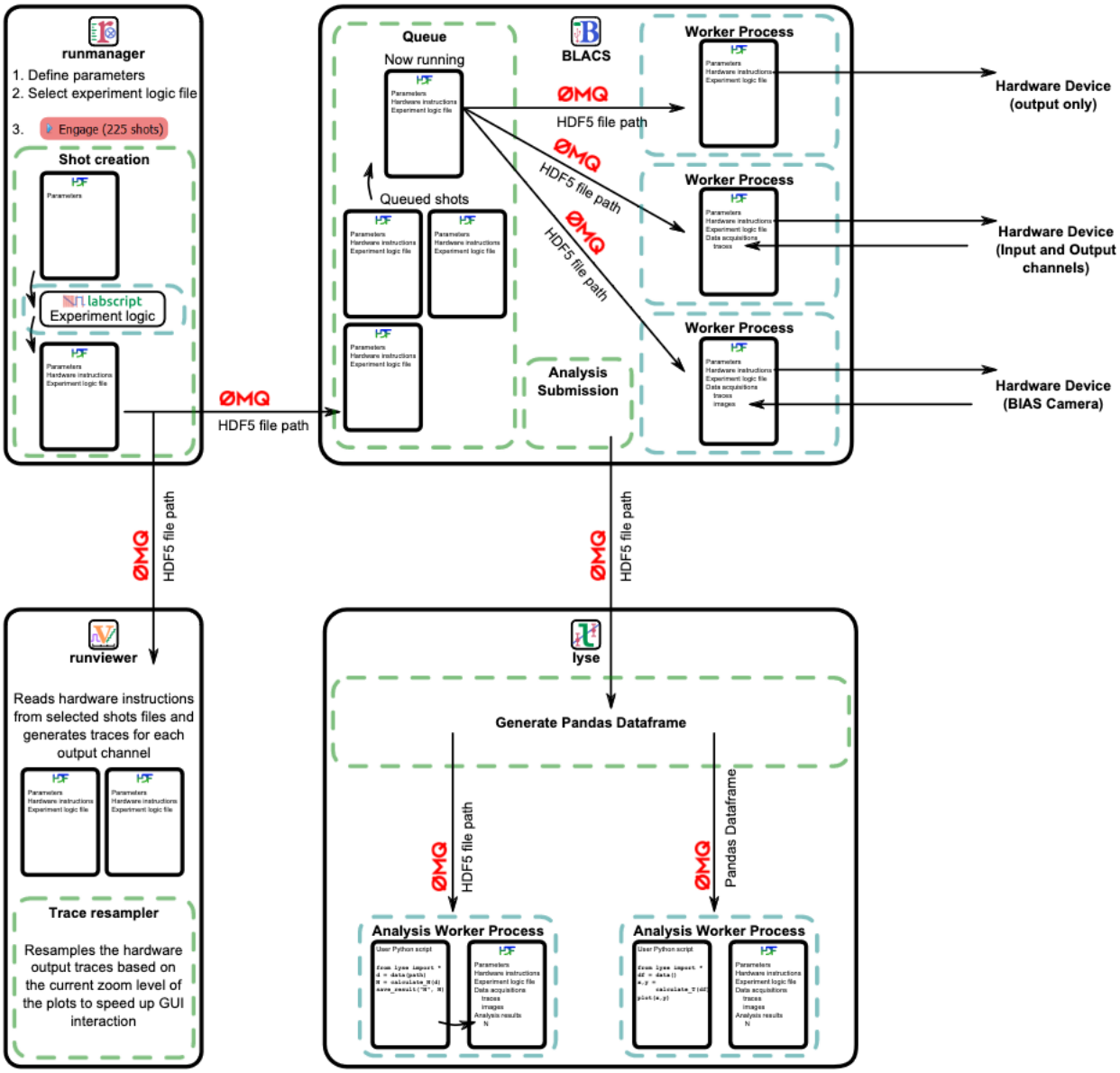


Figure 5.11: **Data flow between components of labscript suite.**
Image Source: [P.T. Starkey, 2019].
Runmanager creates experimental shots that contain all the information about the experimental sequence. These shots are passed to Runviewer for visualization and to BLACS for execution. After execution, shot files are passed to Lyse, where they are analyzed. References to a shot file are passed via ZeroMQ between components of the labscript suite.

5.3.5 Closed-loop control and acquisition system

The closed-loop control and acquisition system integrates various components and technologies from the labscript suite, automating the experiment execution through M-LOOP [Wigley et al., 2016] (as shown in Section 6.2). The central concept behind this closed-loop optimization process is identifying a specific cost function to optimize, such as total atom count, atom density, or atom temperature. Once the cost function is defined, M-LOOP is provided with the relevant parameters to be swept to maximize or minimize the cost. M-LOOP utilizes diverse machine learning techniques, including the Gaussian process, neural networks, and differential evolution, among others, to optimize the defined cost function. Crucial experiment details are stored in a configuration file, as illustrated below. The M-LOOP configuration file is utilized by a multi-shot analysis script on Lyse.

```
% [COMPILATION]
mock = false

% [ANALYSIS]
cost_key = ["characterizing_2DMOT_cooler_and_repumper", "
   gaussian_fit_amplitudes"]
maximize = true

% [MLOOP]
mloop_params = {
    "moglabs_opll_cooler_freq": {"min": 81, "max": 84, "start":
       83},
    "moglabs_opll_repumper_freq": {"min": 83.4, "max": 83.55, "
       start": 83.42}
}
num_training_runs = 40
max_num_runs_without_better_params = 50
max_num_runs = 150
trust_region = 0.1
cost_has_noise = true
controller_type = "gaussian_process"
```

Listing 5.1: An exemplary M-LOOP Configuration file with relevant parameters.

The configuration includes the choice of cost key for evaluating the experiment's performance. By setting 'mock' to 'false', we enable M-LOOP to execute real experimental runs, optimizing the system in a closed loop. In the 'MLOOP' section, crucial parameters are defined, such as 'mloop_params', which sets the range and initial values

of control parameters, and 'num_training_runs', determining the number of initial runs for training the optimization algorithm. Additionally, the parameters 'max_num_runs' and 'max_num_runs_without_better_params' set limits on the number of iterations for optimization. Finally, the 'controller_type' determines the preferred machine learning technique to optimize the cost function.

After configuring M-LOOP to optimize the control parameters based on a specified cost function, a predefined number of training runs are performed to learn the cost function iteratively. Subsequently, M-LOOP generates new experimental shots and sends them to BLACS for execution. Once executed, the shot data are sent to Lyse. Single-shot analysis routines calculate the cost, and the multi-shot analysis file, utilizing the M-LOOP configuration file, determines the next set of optimal control parameters. The iterative process continues, refining control parameters and optimizing performance in a closed loop. The seamless integration of M-LOOP, BLACS, and Lyse enables closed-loop control and acquisition, facilitating systematic parameter space exploration.

5.4 Conclusions and outlook

In this chapter, I presented a detailed overview of our lab's control hardware and software infrastructure. We reviewed the features of the labscript suite, an open-source software package that forms the core of our experiment management. It handles everything from composing experiments to their execution and analysis, serving as our control and acquisition system. Additionally, I demonstrated the integration of machine learning via M-LOOP. This has enabled us to establish a closed-loop feedback control system, automating the optimization of control parameters. This chapter emphasizes the role of infrastructure and automation in advancing our quantum simulator's controllability.

Future work will focus on expanding the control system's capabilities. We plan to integrate both single-shot and multi-shot image analysis routines into our framework to detect and reconfigure atoms during experiments. Furthermore, on the hardware side of things, we aim to unite the slow and fast control systems, which involves establishing a hardware handshake protocol. In this setup, the slow control system will trigger the fast control system to execute atom reconfiguration. Upon completion, it will signal back to the slow control system, allowing it to proceed with the subsequent steps of the experiment.

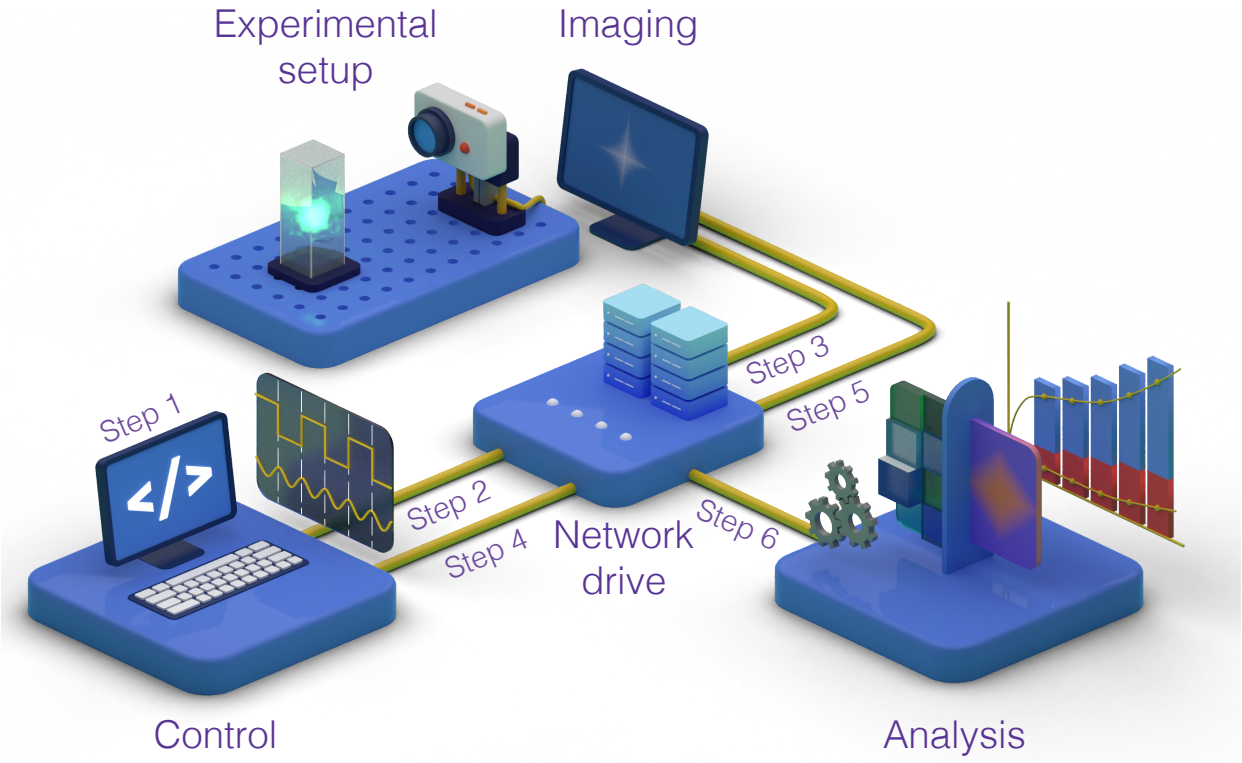


Figure 5.12: **Software workflow of the closed-loop control and acquisition system.**
The software workflow of the closed-loop control and acquisition system can be broken down into six distinct steps. **Step 1:** Use the Labscript API to define the control sequence. **Step 2:** The control sequence is compiled by the Runmanager into HDF5 experimental shots and saved on a network drive. **Step 3:** The camera server extracts the number of images to be taken from each HDF5 experimental shot and waits for that many hardware triggers. **Step 4:** The experimental shot is executed on BLACS, which occupies all the device handlers. **Step 5:** Once the shot is executed, the captured images are saved in the same directory as the experimental shot for easy data tracking. **Step 6:** The acquired data with control parameters are analyzed in real-time, creating a closed-loop control and acquisition system that automatically produces the next set of control parameters until it convergences to optimal parameters or hits an exit condition.

Chapter 6

Characterizing cloud of atoms using closed-loop control and acquisition system

The quantum simulation process with neutral atom arrays initiates by creating a dense atomic cloud. The trapping system then isolates individual atoms within the ensemble by trapping single atoms in optical traps. It is crucial to ensure the atomic cloud contains sufficient atoms for efficient optical trap loading.

This chapter underscores the capabilities of our closed-loop control and acquisition system for creating and optimizing an atomic cloud. Instead of revisiting the extensively documented physics of the 3D Magneto-Optical Trap (3D MOT) formation, our focus shifts to the technical details of our experimental setup that assist in creating and optimizing the 3D MOT. The imaging process of the Rb-87 atom cloud is detailed in Section 6.1, while Section 6.2 explains the automated parameter optimization using M-LOOP.

6.1 Imaging and optimizing a cloud of Rb-87 atoms

Creating a 3D MOT in our lab begins with generating a 2D MOT close to the atomic source cell. A push beam is employed to push the atoms from 2D MOT into a glass cell, subsequently loading them into a 3D MOT. The glass cell is positioned within a constant, calibrated magnetic field required for the loading process. The setup is shown in Fig 3.3b.

A comprehensive script in Appendix A.2 demonstrates the process of creating a 3D MOT. During the loading phase, we activate the magnetic field coils, 2D MOT, and push-beam for a specified period, allowing atoms to populate the 3D MOT within the glass cell. Post-loading, we deactivate these elements, and the 3D MOT's scattered light is captured by a Complementary Metal-Oxide-Semiconductor (CMOS) camera.

The trend in the signal captured by the CMOS camera is shown in Fig 6.1 when sweeping cooler frequency, where Γ ($\equiv 6.07$ MHz) is the decay rate of the cooling transition shown in Fig. 3.4. The total grayscale value of the CMOS camera serves as our metric. This value is proportional to the total number of atoms in the 3D MOT. It is crucial to note that while the atom cloud forms at the 'cooler frequency', the image capture always occurs at a fixed 'imaging frequency'. This approach eliminates any potential dependence of scattered light on frequency variation.

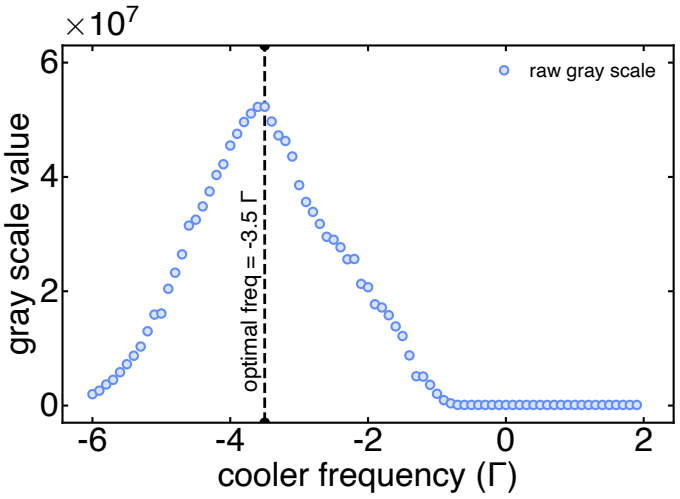


Figure 6.1: **Linearly sweeping cooler frequency to find the optimal control parameter.**

A significant drawback of linearly sweeping control parameters is that the control parameter space discretization constrains the optimal value precision. As illustrated in Fig. 6.1, the cooler frequency is discretized from -6Γ to 2Γ with equal spacing of 0.1Γ. This results in an optimal value precision extending only to one decimal place. For enhanced precision, the sweep would require repetition with a finer resolution.

Moreover, linear sweeping is not an optimal strategy when dealing with multi-parameter spaces, as the control parameter space can potentially increase exponentially. Additionally,

the process of converging to the optimal value scales with the number of discretized points, leaving no room to leverage the data collected for sub-optimal control parameters to expedite the convergence. In the following section, we illustrate how M-LOOP overcomes these issues inherent in linear control parameter sweeps.

6.2 Optimizing control parameters with M-LOOP to expedite convergence

As discussed in Section 5.3.5, M-LOOP enables closed-loop optimization of a cost function, which is generally physics-inspired. Automating the entire experimental cycle—from composition and execution to analysis M-LOOP facilitates the convergence to optimal control parameters. As in the prior section, we demonstrate this efficiency in optimizing 3D MOT cooler frequency.

The M-LOOP configuration file is fed with a cost function to maximize the number of atoms, whose value is calculated using a single shot analysis routine on lyse. As shown in the configuration file below, the 'cost_key' corresponds to the 'raw_grayscale_values' variable, which is the summation of all grayscale values acquired by the CMOS camera, which is proportional to the number of atoms in the 3D MOT. The 'maximize' variable is true to maximize the cost function. The 'mloop_params' dictionary identifies the control parameter necessitating optimization and defines a range within which to seek the optimal value. In this context, the 'cooler frequency' of the 3D MOT is optimized within the same search space as chosen above, i.e., -6Γ to 2Γ ($\equiv$ -36 MHz to 12 MHz). Here, a controller, 'gaussian_process' optimizer, learns the cost function over 'num_training_runs' and predicts the most suitable subsequent control parameter within a 'trust_region' (0.1 in this context). Upon hitting an exit condition, either 'max_num_runs' or 'max_num_runs_without_better_params'—the optimizer returns the optimal value found.

M-LOOP's avoidance of space discretization, a common feature of linear sweeps, enhances the speed of the optimal value determination due to adaptive stepping. It can be postulated that M-LOOP optimizes within the continuous domain of the control parameters.

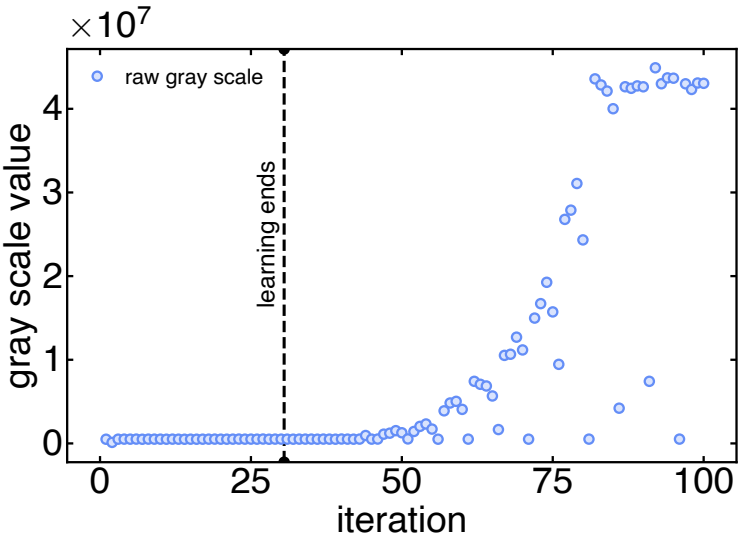
In this example, the 'gaussian_process' controller employs a gradient descent approach to maximize the cost function. As depicted in Fig.6.2a and Fig.6.2b, it is clear that the cost function (in this case, the grayscale value) is optimized following a pattern akin to gradient descent. The controller spends the initial 30 iterations learning the landscape of

the cost function. Upon completing this learning phase, each subsequent iteration sees a general increase in the grayscale value, as demonstrated in Fig. 6.2b.

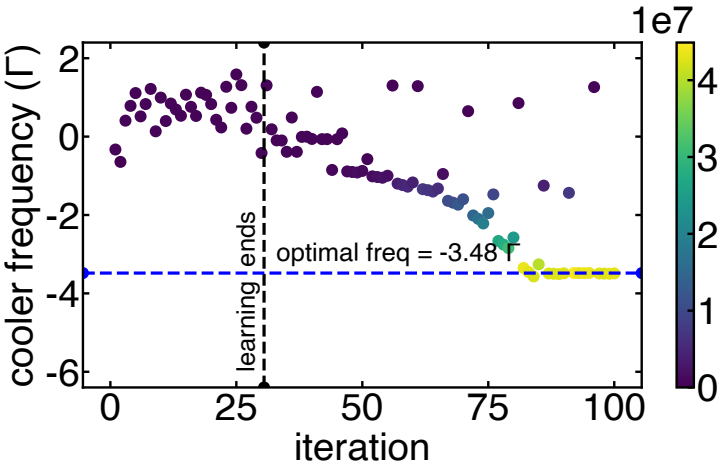
```
%[COMPILATION]
mock = false

%[ANALYSIS]
cost_key = ["mot_analysis", "raw_grayscale_values"]
maximize = true
ignore_bad = false
analysislib_console_log_level = "INFO"
analysislib_file_log_level = "DEBUG"

%[MLOOP]
mloop_params = {"nu_cooler_2DMOT": {"min": -36, "max": 12, "start":
    -2}}
num_training_runs = 30
max_num_runs_without_better_params = 40
max_num_runs = 100
trust_region = 0.1
cost_has_noise = true
no_delay = false
visualisations = true
controller_type = "gaussian_process"
console_log_level = 10
```

Listing 6.1: M-LOOP Configuration used to optimize cooler frequency.

This process of control parameter optimization can be readily extended for multi-parameter optimization in a closed-loop manner. The adjustment required is the specification of the control parameters' names (as determined under runmanager) along with the range within which the cost function is to be minimized/maximized. Moreover, it is feasible to adopt a more complex cost function. For instance, the atom density or temperature could serve as a basis for optimizing control parameters, aligning with the specific requirements of the quantum simulation experiment. This flexibility allows us to tailor the optimization process to the unique demands of the experimental procedure, potentially unlocking greater efficiency and precision.



(a)



(b)

Figure 6.2: **Automated closed-loop optimization of control parameters via M-LOOP.**

a. The 'gaussian_process' controller operates by utilizing a method similar to gradient descent, aiming to maximize the cost function (here, summed grayscale value captured on CMOS camera). **b.** The controller dedicates the first 30 iterations to learn the cost function's landscape. Upon completion of this learning phase, each subsequent iteration sees a general increase in the grayscale value, with the control parameter progressively converging toward the optimal value.

6.3 Conclusions and outlook

In this chapter, I demonstrated the use of M-LOOP to automate the optimization of control parameters. The notable benefits of adopting this methodology include adaptive stepping of parameters, reduced human oversight, and the potential for achieving faster convergence in determining optimal values. This chapter underscores the critical role that a well-developed infrastructure combined with automation plays in enhancing the controllability of our quantum simulator.

Future work will focus on extending M-LOOP's capabilities to multi-parameter optimization. We aim to incorporate M-LOOP in our daily practices to fine-tune our experiment settings. By integrating this with possible improvements in machine learning techniques, our goal is to enhance the controllability of our quantum simulator.

References

[Aaronson and Chen, 2016] Aaronson, S. and Chen, L. (2016). Complexity-theoretic foundations of quantum supremacy experiments.

[Araujo et al., 2021] Araujo, I. F., Park, D. K., Petruccione, F., and da Silva, A. J. (2021). A divide-and-conquer algorithm for quantum state preparation. *Scientific Reports*, 11(1).

[Armitage et al., 2018] Armitage, N. P., Mele, E. J., and Vishwanath, A. (2018). Weyl and Dirac semimetals in three-dimensional solids. *Rev. Mod. Phys.*, 90:015001.

[Arute et al., 2019] Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G. S. L., Buell, D. A., Burkett, B., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., Fowler, A., Gidney, C., Giustina, M., Graff, R., Guerin, K., Habegger, S., Harrigan, M. P., Hartmann, M. J., Ho, A., Hoffmann, M., Huang, T., Humble, T. S., Isakov, S. V., Jeffrey, E., Jiang, Z., Kafri, D., Kechedzhi, K., Kelly, J., Klimov, P. V., Knysh, S., Korotkov, A., Kostritsa, F., Landhuis, D., Lindmark, M., Lucero, E., Lyakh, D., Mandrà, S., McClean, J. R., McEwen, M., Megrant, A., Mi, X., Michielsen, K., Mohseni, M., Mutus, J., Naaman, O., Neeley, M., Neill, C., Niu, M. Y., Ostby, E., Petukhov, A., Platt, J. C., Quintana, C., Rieffel, E. G., Roushan, P., Rubin, N. C., Sank, D., Satzinger, K. J., Smelyanskiy, V., Sung, K. J., Trevithick, M. D., Vainsencher, A., Villalonga, B., White, T., Yao, Z. J., Yeh, P., Zalcman, A., Neven, H., and Martinis, J. M. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510.

[Baldwin et al., 2022] Baldwin, C. H., Mayer, K., Brown, N. C., Ryan-Anderson, C., and Hayes, D. (2022). Re-examining the quantum volume test: Ideal distributions, compiler optimizations, confidence intervals, and scalable resource estimations. *Quantum*, 6:707.

[Barredo et al., 2016] Barredo, D., de Lé séleuc, S., Lienhard, V., Lahaye, T., and Browaeys, A. (2016). An atom-by-atom assembler of defect-free arbitrary two-dimensional atomic arrays. *Science*, 354(6315):1021–1023.

[Barredo et al., 2018] Barredo, D., Lienhard, V., de Léséleuc, S., Lahaye, T., and Browaeys, A. (2018). Synthetic three-dimensional atomic structures assembled atom by atom. *Nature*, 561(7721):79–82.

[Bernien et al., 2017] Bernien, H., Schwartz, S., Keesling, A., Levine, H., Omran, A., Pichler, H., Choi, S., Zibrov, A. S., Endres, M., Greiner, M., Vuletić, V., and Lukin, M. D. (2017). Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551(7682):579–584.

[Bluvstein et al., 2022] Bluvstein, D., Levine, H., Semeghini, G., Wang, T. T., Ebadi, S., Kalinowski, M., Keesling, A., Maskara, N., Pichler, H., Greiner, M., Vuletić, V., and Lukin, M. D. (2022). A quantum processor based on coherent transport of entangled atom arrays. *Nature*, 604(7906):451–456.

[Browaeys et al., 2016] Browaeys, A., Barredo, D., and Lahaye, T. (2016). Experimental investigations of dipole–dipole interactions between a few Rydberg atoms. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 49(15):152001.

[Browaeys and Lahaye, 2020] Browaeys, A. and Lahaye, T. (2020). Many-body physics with individually controlled Rydberg atoms. *Nature Physics*, 16(2):132–142.

[Cimring et al., 2022] Cimring, B., Sabeh, R. E., Bacvanski, M., Maaz, S., Hajj, I. E., Nishimura, N., Mouawad, A. E., and Cooper, A. (2022). Efficient algorithms to solve atom reconfiguration problems. i. the redistribution-reconfiguration (red-rec) algorithm.

[Cirac and Zoller, 1995] Cirac, J. I. and Zoller, P. (1995). Quantum Computations with Cold Trapped Ions. *Phys. Rev. Lett.*, 74:4091–4094.

[Cooper et al., 2018] Cooper, A., Covey, J. P., Madjarov, I. S., Porsev, S. G., Safronova, M. S., and Endres, M. (2018). Alkaline-Earth Atoms in Optical Tweezers. *Physical Review X*, 8(4).

[Cramer et al., 2010] Cramer, M., Plenio, M. B., Flammia, S. T., Somma, R., Gross, D., Bartlett, S. D., Landon-Cardinal, O., Poulin, D., and Liu, Y.-K. (2010). Efficient quantum state tomography. *Nature Communications*, 1(1).

[Cross et al., 2019] Cross, A. W., Bishop, L. S., Sheldon, S., Nation, P. D., and Gambetta, J. M. (2019). Validating quantum computers using randomized model circuits. *Physical Review A*, 100(3).

[de Léséleuc et al., 2018] de Léséleuc, S., Weber, S., Lienhard, V., Barredo, D., Büchler, H. P., Lahaye, T., and Browaeys, A. (2018). Accurate Mapping of Multilevel Rydberg Atoms on Interacting Spin-1/2 Particles for the Quantum Simulation of Ising Models. *Physical Review Letters*, 120(11).

[Dirac, 1930] Dirac, P. A. M. (1930). *The Principles of Quantum Mechanics*. Clarendon Press.

[DiVincenzo, 2000] DiVincenzo, D. P. (2000). The Physical Implementation of Quantum Computation. *Fortschritte der Physik*, 48(9-11):771–783.

[Dong and Petersen, 2010] Dong, D. and Petersen, I. (2010). Quantum control theory and applications: a survey. *IET Control Theory & Applications*, 4(12):2651–2671.

[Ebadi et al., 2021] Ebadi, S., Wang, T. T., Levine, H., Keesling, A., Semeghini, G., Omran, A., Bluvstein, D., Samajdar, R., Pichler, H., Ho, W. W., Choi, S., Sachdev, S., Greiner, M., Vuletić, V., and Lukin, M. D. (2021). Quantum phases of matter on a 256-atom programmable quantum simulator. *Nature*, 595(7866):227–232.

[Einstein et al., 1935] Einstein, A., Podolsky, B., and Rosen, N. (1935). Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Phys. Rev.*, 47:777–780.

[Ekert, 1991] Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. *Phys. Rev. Lett.*, 67:661–663.

[Endres et al., 2016] Endres, M., Bernien, H., Keesling, A., Levine, H., Anschuetz, E. R., Krajenbrink, A., Senko, C., Vuletic, V., Greiner, M., and Lukin, M. D. (2016). Atom-by-atom assembly of defect-free one-dimensional cold atom arrays. *Science*, 354(6315):1024–1027.

[Evered et al., 2023] Evered, S. J., Bluvstein, D., Kalinowski, M., Ebadi, S., Manovitz, T., Zhou, H., Li, S. H., Geim, A. A., Wang, T. T., Maskara, N., Levine, H., Semeghini, G., Greiner, M., Vuletic, V., and Lukin, M. D. (2023). High-fidelity parallel entangling gates on a neutral atom quantum computer.

[Feynman, 1982] Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488.

[Fuhrmanek et al., 2011] Fuhrmanek, A., Bourgain, R., Sortais, Y. R. P., and Browaeys, A. (2011). Free-Space Lossless State Detection of a Single Trapped Atom. *Phys. Rev. Lett.*, 106:133003.

[Gallagher, 1988] Gallagher, T. F. (1988). Rydberg atoms. *Reports on Progress in Physics*, 51(2):143.

[Geim and Grigorieva, 2013] Geim, A. K. and Grigorieva, I. V. (2013). Van der Waals heterostructures. *Nature*, 499(7459):419–425.

[Grover, 1996] Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA. Association for Computing Machinery.

[HAFFNER et al., 2008] HAFFNER, H., ROOS, C., and BLATT, R. (2008). Quantum computing with trapped ions. *Physics Reports*, 469(4):155–203.

[Henriet et al., 2020] Henriet, L., Beguin, L., Signoles, A., Lahaye, T., Browaeys, A., Reymond, G.-O., and Jurczak, C. (2020). Quantum computing with neutral atoms. *Quantum*, 4:327.

[Jurcevic et al., 2020] Jurcevic, P., Javadi-Abhari, A., Bishop, L. S., Lauer, I., Bogorin, D. F., Brink, M., Capelluto, L., Günlük, O., Itoko, T., Kanazawa, N., Kandala, A., Keefe, G. A., Krsulich, K., Landers, W., Lewandowski, E. P., McClure, D. T., Nannicini, G., Narasgond, A., Nayfeh, H. M., Pritchett, E., Rothwell, M. B., Srinivasan, S., Sundaresan, N., Wang, C., Wei, K. X., Wood, C. J., Yau, J.-B., Zhang, E. J., Dial, O. E., Chow, J. M., and Gambetta, J. M. (2020). Demonstration of quantum volume 64 on a superconducting quantum computing system.

[Kim et al., 2009] Kim, K., Chang, M.-S., Islam, R., Korenblit, S., Duan, L.-M., and Monroe, C. (2009). Entanglement and Tunable Spin-Spin Couplings between Trapped Ions Using Multiple Transverse Modes. *Physical Review Letters*, 103(12).

[Levine et al., 2019] Levine, H., Keesling, A., Semeghini, G., Omran, A., Wang, T. T., Ebadi, S., Bernien, H., Greiner, M., Vuletić, V., Pichler, H., and Lukin, M. D. (2019). Parallel Implementation of High-Fidelity Multiqubit Gates with Neutral Atoms. *Phys. Rev. Lett.*, 123:170503.

[Linke et al., 2017] Linke, N. M., Maslov, D., Roetteler, M., Debnath, S., Figgatt, C., Landsman, K. A., Wright, K., and Monroe, C. (2017). Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310.

[Madsen et al., 2022] Madsen, L. S., Laudenbach, F., Askarani, M. F., Rortais, F., Vincent, T., Bulmer, J. F. F., Miatto, F. M., Neuhaus, L., Helt, L. G., Collins, M. J., Lita, A. E., Gerrits, T., Nam, S. W., Vaidya, V. D., Menotti, M., Dhand, I., Vernon, Z., Quesada, N., and Lavoie, J. (2022). Quantum computational advantage with a programmable photonic processor. *Nature*, 606(7912):75–81.

[Majer et al., 2007] Majer, J., Chow, J. M., Gambetta, J. M., Koch, J., Johnson, B. R., Schreier, J. A., Frunzio, L., Schuster, D. I., Houck, A. A., Wallraff, A., Blais, A., Devoret, M. H., Girvin, S. M., and Schoelkopf, R. J. (2007). Coupling superconducting qubits via a cavity bus. *Nature*, 449(7161):443–447.

[Metcalf and van der Straten, 2003] Metcalf, H. J. and van der Straten, P. (2003). Laser cooling and trapping of atoms. *J. Opt. Soc. Am. B*, 20(5):887–908.

[Mukhopadhyay et al., 2022] Mukhopadhyay, P., Gheorghiu, V., Huang, J., Li, S. M., and Mosca, M. (2022). Reducing the CNOT count for Clifford+T circuits on NISQ architectures.

[Nelson et al., 2007] Nelson, K. D., Li, X., and Weiss, D. S. (2007). Imaging single atoms in a three-dimensional array. *Nature Physics*, 3(8):556–560.

[Nogrette et al., 2014] Nogrette, F., Labuhn, H., Ravets, S., Barredo, D., Béguin, L., Vernier, A., Lahaye, T., and Browaeys, A. (2014). Single-Atom Trapping in Holographic 2D Arrays of Microtraps with Arbitrary Geometries. *Phys. Rev. X*, 4:021034.

[Novoselov et al., 2016] Novoselov, K. S., Mishchenko, A., Carvalho, A., and Neto, A. H. C. (2016). 2D materials and van der Waals heterostructures. *Science*, 353(6298):aac9439.

[Phillips, 1998] Phillips, W. D. (1998). Nobel Lecture: Laser cooling and trapping of neutral atoms. *Rev. Mod. Phys.*, 70:721–741.

[Pino et al., 2021] Pino, J. M., Dreiling, J. M., Figgatt, C., Gaebler, J. P., Moses, S. A., Allman, M. S., Baldwin, C. H., Foss-Feig, M., Hayes, D., Mayer, K., Ryan-Anderson, C., and Neyenhuis, B. (2021). Demonstration of the trapped-ion quantum CCD computer architecture. *Nature*, 592(7853):209–213.

[Preskill, 2018] Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79.

[Pritchard et al., 2010] Pritchard, J. D., Maxwell, D., Gauguet, A., Weatherill, K. J., Jones, M. P. A., and Adams, C. S. (2010). Cooperative Atom-Light Interaction in a Blockaded Rydberg Ensemble. *Phys. Rev. Lett.*, 105:193603.

[P.T. Starkey, 2019] P.T. Starkey (2019). A software framework for control and automation of precisely timed experiments.

[Qiskit contributors, 2023] Qiskit contributors (2023). Qiskit: An open-source framework for quantum computing.

[Rosi et al., 2018] Rosi, S., Burchianti, A., Conclave, S., Naik, D. S., Roati, G., Fort, C., and Minardi, F. (2018). Λ-enhanced grey molasses on the D2 transition of Rubidium-87 atoms. *Scientific Reports*, 8(1).

[Sabeh et al., 2022] Sabeh, R. E., Bohm, J., Ding, Z., Maaz, S., Nishimura, N., Hajj, I. E., Mouawad, A. E., and Cooper, A. (2022). Efficient algorithms to solve atom reconfiguration problems. ii. the assignment-rerouting-ordering (aro) algorithm.

[Saffman et al., 2010] Saffman, M., Walker, T. G., and Mølmer, K. (2010). Quantum information with Rydberg atoms. *Reviews of Modern Physics*, 82(3):2313–2363.

[Schlosser et al., 2001] Schlosser, N., Reymond, G., Protsenko, I., and Grangier, P. (2001). Sub-poissonian loading of single atoms in a microscopic dipole trap. *Nature*, 411(6841):1024–1027.

[Sheng et al., 2018] Sheng, C., He, X., Xu, P., Guo, R., Wang, K., Xiong, Z., Liu, M., Wang, J., and Zhan, M. (2018). High-Fidelity Single-Qubit Gates on Neutral Atoms in a Two-Dimensional Magic-Intensity Optical Dipole Trap Array. *Phys. Rev. Lett.*, 121:240501.

[Sherson et al., 2010] Sherson, J. F., Weitenberg, C., Endres, M., Cheneau, M., Bloch, I., and Kuhr, S. (2010). Single-atom-resolved fluorescence imaging of an atomic Mott insulator. *Nature*, 467(7311):68–72.

[Shor, 1994] Shor, P. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134.

[Sivarajah et al., 2020] Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., and Duncan, R. (2020). t|ket⟩: a retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6(1):014003.

[Tan et al., 2023] Tan, D. B., Bluvstein, D., Lukin, M. D., and Cong, J. (2023). Compiling quantum circuits for dynamically field-programmable neutral atoms array processors.

[Tokura et al., 2017] Tokura, Y., Kawasaki, M., and Nagaosa, N. (2017). Emergent functions of quantum materials. *Nature Physics*, 13(11):1056–1068.

[Wang et al., 2016] Wang, Y., Surendran, P., Jose, S., Tran, T., Herrera, I., Whitlock, S., McLean, R., Sidorov, A., and Hannaford, P. (2016). Magnetic lattices for ultracold atoms and degenerate quantum gases. *Science Bulletin*, 61(14):1097–1106.

[Wendin, 2017] Wendin, G. (2017). Quantum information processing with superconducting circuits: a review. *Reports on Progress in Physics*, 80(10):106001.

[Wiegand et al., 2019] Wiegand, B., Leykauf, B., Döringshoff, K., Gupta, Y. D., Peters, A., and Krutzik, M. (2019). A single-laser alternating-frequency magneto-optical trap. *Review of Scientific Instruments*, 90(10):103202.

[Wigley et al., 2016] Wigley, P. B., Everitt, P. J., van den Hengel, A., Bastian, J. W., Sooriyabandara, M. A., McDonald, G. D., Hardman, K. S., Quinlivan, C. D., Manju, P., Kuhn, C. C. N., Petersen, I. R., Luiten, A. N., Hope, J. J., Robins, N. P., and Hush, M. R. (2016). Fast machine-learning online optimization of ultra-cold-atom experiments. *Scientific Reports*, 6(1).

[Wood and Gambetta, 2018] Wood, C. J. and Gambetta, J. M. (2018). Quantification and characterization of leakage errors. *Physical Review A*, 97(3).

[Wu et al., 2021] Wu, X., Liang, X., Tian, Y., Yang, F., Chen, C., Liu, Y.-C., Tey, M. K., and You, L. (2021). A concise review of Rydberg atom based quantum computation and quantum simulation*. *Chinese Physics B*, 30(2):020305.

[Yu and Li, 2022] Yu, Z. and Li, Y. (2022). Analysis of Error Propagation in Quantum Computers.

[Zhang et al., 2021] Zhang, K., Rao, P., Yu, K., Lim, H., and Korepin, V. (2021). Implementation of efficient quantum search algorithms on NISQ computers. *Quantum Information Processing*, 20(7).

[Zhong et al., 2020] Zhong, H.-S., Wang, H., Deng, Y.-H., Chen, M.-C., Peng, L.-C., Luo, Y.-H., Qin, J., Wu, D., Ding, X., Hu, Y., Hu, P., Yang, X.-Y., Zhang, W.-J., Li, H., Li, Y., Jiang, X., Gan, L., Yang, G., You, L., Wang, Z., Li, L., Liu, N.-L., Lu, C.-Y., and Pan, J.-W. (2020). Quantum computational advantage using photons. *Science*, 370(6523):1460–1463.

APPENDICES

Appendix A

Labscript suite codes

A.1 Connection table

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from multiprocessing import connection
from labscript import *
from labscript_utils.unitconversions import *
from labscript_devices.PrawnBlaster.labscript_devices import PrawnBlaster
from labscript_devices.NI_DAQmx.models.NI_PXIe_6739 import NI_PXIe_6739
from labscript_devices.NI_DAQmx.models.NI_PXIe_6537 import NI_PXIe_6537
from labscript_devices.FunctionRunner.labscript_devices import FunctionRunner
from labscript_devices.Camera import Camera

PrawnBlaster(name='prawn', com_port='COM4', num_pseudoclocks=4,
             out_pins=[9, 11, 13, 15], in_pins=[2, 2, 2, 2])
NI_PXIe_6739(name='analog_card_1', parent_device=prawn.clocklines[0],
             clock_terminal='/PXI1Slot2/PFI0',
             clock_mirror_terminal='/PXI1Slot2/PXI_Trig0',
             MAX_name='PXI1Slot2')
NI_PXIe_6537(name='digital_card_1', parent_device=prawn.clocklines[1],
             clock_terminal='PFI4', MAX_name='PXI1Slot6')
FunctionRunner('function_runner')
```

```
AnalogOut(name='CS_1_Mod', parent_device=analog_card_1, connection='ao0'
          , unit_conversion_class=current_source,
          unit_conversion_parameters={'A_per_V': 0.4})
AnalogOut(name='CC_1', parent_device=analog_card_1, connection='ao4',
          limits=(-0.8, 0.8))
AnalogOut(name='CC_2', parent_device=analog_card_1, connection='ao8',
          limits=(-0.8, 0.8))
AnalogOut(name='CC_3', parent_device=analog_card_1, connection='ao12',
          limits=(-0.8, 0.8))
AnalogOut(name='GC_1', parent_device=analog_card_1, connection='ao16',
          limits=(-0.8, 0.8))
AnalogOut(name='GC_2', parent_device=analog_card_1, connection='ao20',
          limits=(-0.8, 0.8))
AnalogOut(name='Mixer_1_cooler_3DMOT', parent_device=analog_card_1,
          connection='ao24', limits=(0, 0.128))
AnalogOut(name='Mixer_2_cooler_2DMOT', parent_device=analog_card_1,
          connection='ao28', limits=(0, 0.128))
AnalogOut(name='Mixer_3_cooler_PGC', parent_device=analog_card_1,
          connection='ao32', limits=(0, 0.128))
AnalogOut(name='Mixer_4_master', parent_device=analog_card_1,
          connection='ao36', limits=(0, 0.128))
AnalogOut(name='Mixer_5_repumper_main', parent_device=analog_card_1,
          connection='ao40', limits=(0, 0.128))
AnalogOut(name='Mixer_6_repumper_3DMOT', parent_device=analog_card_1,
          connection='ao44', limits=(0, 0.128))
AnalogOut(name='Mixer_7_repumper_2DMOT', parent_device=analog_card_1,
          connection='ao48', limits=(0, 0.128))
AnalogOut(name='Mixer_8_repumper_PGC', parent_device=analog_card_1,
          connection='ao52', limits=(0, 0.128))
AnalogOut(
    name='Mog_2C',
    parent_device=analog_card_1,
    connection='ao56',
    limits=(-1, 1),
    unit_conversion_class=moglabs_opll,
    unit_conversion_parameters={
        'central_freq_MHz': 84.50,
```

```
        'max_depth_MHz': 250,
        'max_gain_factor': 1073709056,
        'gain': 60127707,
        },
    )

AnalogOut(
    name='Mog_2R',
    parent_device=analog_card_1,
    connection='ao60',
    limits=(-1, 1),
    unit_conversion_class=moglabs_opll,
    unit_conversion_parameters={
        'central_freq_MHz': 83.25,
        'max_depth_MHz': 250,
        'max_gain_factor': 1073709056,
        'gain': 1717934,
        },
    )

DigitalOut(name='CS_1_power', parent_device=digital_card_1,
           connection='port0/line0')
DigitalOut(name='Mi_Switch_1_cooler_3DMOT',
           parent_device=digital_card_1, connection='port0/line1')
DigitalOut(name='Mi_Switch_2_cooler_2DMOT',
           parent_device=digital_card_1, connection='port0/line2')
DigitalOut(name='Mi_Switch_3_cooler_PGC', parent_device=digital_card_1,
           connection='port0/line3')
DigitalOut(name='Mi_Switch_4', parent_device=digital_card_1,
           connection='port0/line4')
DigitalOut(name='Mi_Switch_5_repumper_main',
           parent_device=digital_card_1, connection='port0/line5')
DigitalOut(name='Mi_Switch_6_repumper_3DMOT',
           parent_device=digital_card_1, connection='port0/line6')
DigitalOut(name='Mi_Switch_7_repumper_2DMOT',
           parent_device=digital_card_1, connection='port0/line7')
DigitalOut(name='Mi_Switch_8_repumper_PGC',
           parent_device=digital_card_1, connection='port1/line0')
```

```
DigitalOut(name='Me_Shutter_1_cooler_3DMOT',
           parent_device=digital_card_1, connection='port1/line1')
DigitalOut(name='Me_Shutter_2_cooler_2DMOT',
           parent_device=digital_card_1, connection='port1/line2')
DigitalOut(name='Me_Shutter_3_cooler_PGC',
           parent_device=digital_card_1, connection='port1/line3')
DigitalOut(name='Me_Shutter_4', parent_device=digital_card_1,
           connection='port1/line4')
DigitalOut(name='Me_Shutter_5_repumper_3DMOT',
           parent_device=digital_card_1, connection='port1/line5')
DigitalOut(name='Me_Shutter_6_repumper_2DMOT',
           parent_device=digital_card_1, connection='port1/line6')
DigitalOut(name='Me_Shutter_7_repumper_PGC',
           parent_device=digital_card_1, connection='port1/line7')
DigitalOut(name='Me_Shutter_8_pushbeam', parent_device=digital_card_1,
           connection='port2/line0')
Camera(
    name='cmos_2DMOT',
    parent_device=digital_card_1,
    connection='port2/line1',
    BIAS_port=8765,
    effective_pixel_size=4e-6,
    exposure_time=1e-3,
    )
Camera(
    name='cmos_3DMOT_1',
    parent_device=digital_card_1,
    connection='port2/line2',
    BIAS_port=8765,
    effective_pixel_size=4e-6,
    exposure_time=1e-3,
    )
Camera(
    name='cmos_3DMOT_2',
    parent_device=digital_card_1,
    connection='port2/line3',
    BIAS_port=8765,
    effective_pixel_size=4e-6,
```

```
    exposure_time=1e-3,
    )
Camera(
    name='cmos_closed_loop_1',
    parent_device=digital_card_1,
    connection='port2/line4',
    BIAS_port=8765,
    effective_pixel_size=4e-6,
    exposure_time=1e-3,
    )
Camera(
    name='cmos_closed_loop_2',
    parent_device=digital_card_1,
    connection='port2/line5',
    BIAS_port=8765,
    effective_pixel_size=4e-6,
    exposure_time=1e-3,
    )
DigitalOut(name='EMCCD_shutter_external', parent_device=digital_card_1,
           connection='port2/line6')
Camera(
    name='EMCCD',
    parent_device=digital_card_1,
    connection='port2/line7',
    BIAS_port=8765,
    effective_pixel_size=4e-6,
    exposure_time=1e-3,
    )

if __name__ == '__main__':
    start()
    stop(1)
```

A.2 Imaging an atomic cloud of Rb-87 atoms

```
#!/usr/bin/python
```

```
# -*- coding: utf-8 -*-

import numpy as np
from labscript import start, stop
from labscript_utils import import_or_reload
from labscriptlib.Rydberg.Utils.ModuleControl import *

loading_time: float
loading_imaging_times: list
camera_exposure_time: float
nu_cooler_2DMOT: float
nu_cooler_imaging: float
imaging_time_gap: float
nu_cooler_infinity: float
dissipation_time_gap: float
nu_opll_repumper_optimal: float
shutter_delay_time: float
flag_get_background_image: bool
flag_2DMOT_on: bool
flag_pushbeam_on: bool
flag_3DMOT_imaging_after: bool
convert_detuning_to_opll: str
photodiode_saturation_time: float
photodiode_server_updating_period: float

import_or_reload("labscriptlib.Rydberg.connection_table")
convert_detuning_to_opll = eval(convert_detuning_to_opll)

t = 0
start()

# Measuring MOT2D cooler beam powers
turn_on_2D_MOT(t)
turn_on_pushbeam(t)
Me_Shutter_2_cooler_2DMOT.go_high(t)
Me_Shutter_6_repumper_2DMOT.go_low(t)

t += photodiode_saturation_time
```

```
t = trigger_monpi_pd2(t)

# Measuring MOT2D repumper beam powers
Me_Shutter_2_cooler_2DMOT.go_low(t)
Me_Shutter_6_repumper_2DMOT.go_high(t)

t += photodiode_saturation_time

t = trigger_monpi_pd2(t)

turn_off_2D_MOT(t)
turn_off_pushbeam(t)

t += dissipation_time_gap

# Loading stage

set_opll_repumper_freq(t, nu_opll_repumper_optimal)
set_opll_cooler_freq(t, convert_detuning_to_opll(nu_cooler_2DMOT))

turn_on_coils(t)
if flag_2DMOT_on:
    turn_on_2D_MOT(t)
if flag_pushbeam_on:
    turn_on_pushbeam(t)

turn_on_3D_MOT(t)

turn_off_PGC(t)

# Taking images from the MOT as it loads at specified times
for loading_imaging_time in loading_imaging_times:
    if loading_imaging_time + camera_exposure_time > loading_time:
        raise Exception(
            "WARNING! Taking images during loading time"
            )
        )
```

```
    cmos_3DMOT.expose(
        "{0}".format(loading_time), t + loading_imaging_time, frametype="frame_0"
    )

t += loading_time

turn_off_coils(t)

turn_off_2D_MOT(t)
turn_off_pushbeam(t)

if flag_3DMOT_imaging_after:
    set_opll_cooler_freq(t, convert_detuning_to_opll(nu_cooler_imaging))

    if imaging_time_gap > 0:
        turn_off_3D_MOT(t)

        t += imaging_time_gap
        turn_on_3D_MOT(t)

        t += shutter_delay_time
        cmos_3DMOT.expose("Main Image", t, frametype="frame_0")

        t += camera_exposure_time
        turn_off_3D_MOT(t)
    else:
        cmos_3DMOT.expose("Main Image", t, frametype="frame_0")

        t += camera_exposure_time
        turn_off_3D_MOT(t)
else:
    turn_off_3D_MOT(t)

if flag_get_background_image:
    set_opll_cooler_freq(t, convert_detuning_to_opll(nu_cooler_infinity))

    t += dissipation_time_gap
    turn_on_3D_MOT(t)
```

```
    t += shutter_delay_time
    cmos_3DMOT.expose("Background Image", t, frametype="frame_0")

    t += camera_exposure_time
    stop(t)
else:
    t += dissipation_time_gap
    stop(t)
```

A.3 Additions and modifications in the Labscript suite software package

A.3.1 Unit conversion class - Current source

```
from .UnitConversionBase import *

class current_source(UnitConversion):
    base_unit = "V"
    derived_units = ["A"]

    def __init__(self, calibration_parameters={"A_per_V": 0.4}):
        self.parameters = calibration_parameters

        UnitConversion.__init__(self, self.parameters)

    def A_to_base(self, amps):
        volts = amps / self.parameters["A_per_V"]
        return volts

    def A_from_base(self, volts):
        amps = volts * self.parameters["A_per_V"]
        return amps
```

A.3.2 Unit conversion class - Moglabs

```
from .UnitConversionBase import *

class moglabs_opll(UnitConversion):
    base_unit = "V"
    derived_units = ["MHz"]

    def __init__(
        self,
        calibration_parameters={
            "central_freq_MHz": 98,
            "max_depth_MHz": 250,
            "max_gain_factor": 1073709056,
            "gain": 1,
        },
    ):
        self.parameters = calibration_parameters

        UnitConversion.__init__(self, self.parameters)

    def MHz_to_base(self, freq):
        volts = (freq - self.parameters["central_freq_MHz"]) / (
            self.parameters["max_depth_MHz"]
            * self.parameters["gain"]
            / self.parameters["max_gain_factor"]
        )
        return volts

    def MHz_from_base(self, volts):
        freq = (
            self.parameters["central_freq_MHz"]
            + self.parameters["max_depth_MHz"]
            * (self.parameters["gain"] / self.parameters["max_gain_factor"])
            * volts
        )
        return freq
```

A.3.3 Updating BLACS UI

Earlier BLACS UI displayed all the 64 AO channels and the digital channels of NI_PXIe_6739 (analog card). Out of these 64 AO channels, only 16 channels are used to have an improved update speed of 1MS/s (otherwise 350kS/s if all channels are wired). Thus, deleting the extra channels from BLACS UI will make relevant channels more accessible.

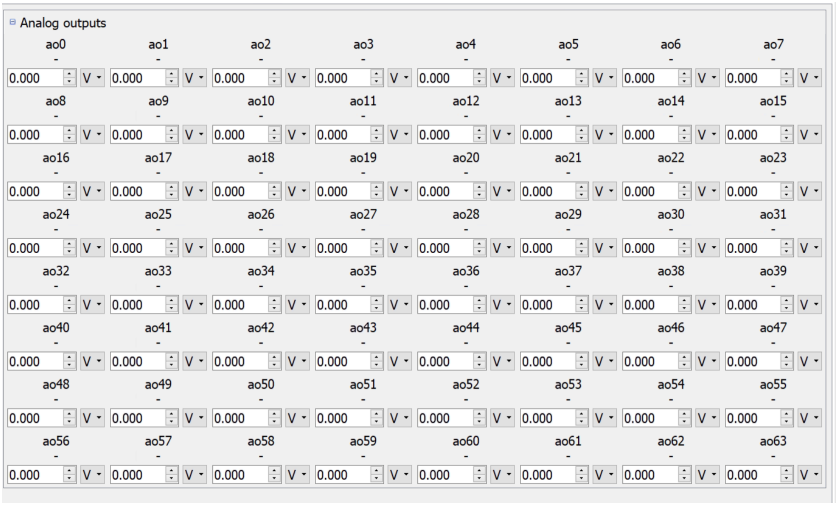


Figure A.1: BLACS UI without any modifications.

To modify the UI layout, labscript_devices/NI_DAQmx/blacs_tabs.py is changed. The script is updated by changing the AO_prop dictionary on line 67. The new dictionary checks the MAX_name of the device and selects only the relevant AO output channels to be displayed. Digital channels on the analog card aren't used, thus, they are also removed from the BLACS UI. Also, I updated the program_manual function in blacs_worker.py by selecting every 4th channel.

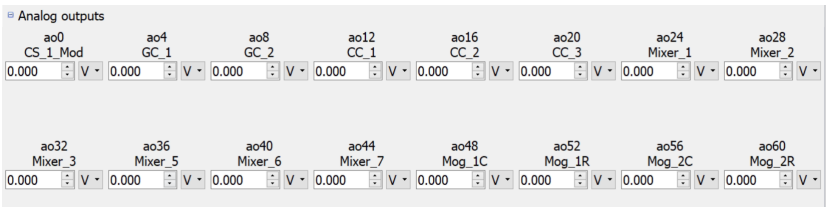


Figure A.2: BLACS UI after selecting relevant analog channels for display.

A.3.4 Customized camera server

The script from the Labscript suite is extended to make it compatible with our image acquisition and analysis software package.

```
import sys
import time
import zprocess
import labscript_utils.shared_drive
import labscript_utils.h5_lock
import h5py
import numpy as np
import os, glob
from pathlib import Path
from datetime import datetime
import numpy as np

from experiment.instruments.camera.basler_ace2
    .basler_ace2 import BaslerAce2
from experiment.toolkits
    .image_acquisition_and_processing.image_toolkit
    .image_acquisition import (
    ImageAcquisition,
)
from experiment.utils.data.data_path import DataPath
from experiment.utils.data.data_transfer import DataTransfer
from experiment.instruments.camera
    .andor_ixonultra888.andor_ixonultra888 import (
    AndoriXonUltra888,
)


class CameraServer(zprocess.ZMQServer):
    def __init__(self, port):
        zprocess.ZMQServer.__init__(self, port, type="string")
        self._h5_filepath = None

        ia1 = ImageAcquisition({"cmos": BaslerAce2(0)})
```

```
    self.cmos = ia1.get("cmos")
    with self.cmos:
        self.cmos.update_configuration_files(
            path\to\cmos_static_properties.yml,
            "static",
        )
        self.cmos.update_configuration_files(
            path\to\cmos_dynamic_properties.yml,
            "dynamic",
        )
        self.cmos.set_static_properties()
        self.cmos.set_acquisition_properties()
    self.cmos.open_connection()

def handler(self, request_data):
    try:
        print(request_data)
        if request_data == "hello":
            return "hello"
        elif request_data.endswith(".h5"):
            self._h5_filepath = labscript_utils.shared_drive.path_to_local(
                request_data
            )
            self.send("ok")
            self.recv()
            self.transition_to_buffered(self._h5_filepath)
            return "done"
        elif request_data == "done":
            self.send("ok")
            self.recv()
            self.transition_to_static(self._h5_filepath)
            self._h5_filepath = None
            return "done"
        elif request_data == "abort":
            self.abort()
            self._h5_filepath = None
            return "done"
        else:
```

```
            raise ValueError("invalid request: %s" % request_data)
    except Exception:
        if self._h5_filepath is not None and request_data != "abort":
            try:
                self.abort()
            except Exception as e:
                sys.stderr.write(
                    "Exception in self.abort() while handling another
                    exception:\n{}\n".format(str(e))
                )
        self._h5_filepath = None
        raise

def get_exp_folder_name(self, shot_path):
    exp_folder_name = "\\".join(shot_path.split("\\")[:-2])
    return exp_folder_name

def get_shot_name(self, shot_path):
    shot_name = shot_path.split("\\")[-1].split(".")[0]
    return shot_name

def get_image_folder(self, exp_folder_name, category):
    if category == "2DMOT":
        image_folder = exp_folder_name + "\\cmos_images_2DMOT\\"
    if category == "3DMOT":
        image_folder = exp_folder_name + "\\cmos_images_3DMOT\\"
    if category == "LoadingTest3DMOT":
        image_folder = exp_folder_name + "\\raw_data\\"
    return image_folder

def get_shot_globals(self, file):
    params = {}
    with h5py.File(file, "r") as f:
        for name, value in f["globals"].attrs.items():
            if isinstance(value, np.bool_):
                value = bool(value)
            if isinstance(value, np.int32):
                value = int(value)
```

```
            if isinstance(value, h5py.Reference) and not value:
                value = None
            if isinstance(value, np.str_):
                value = str(value)
            if isinstance(value, bytes):
                value = value.decode()
            params[name] = value
    return params

def transition_to_buffered(self, h5_filepath):
    """To be overridden by subclasses. Do any preparatory processing
    before a shot, eg setting exposure times, readying cameras to receive
    triggers etc."""
    print("transition to buffered")
    file_path = h5_filepath
    params = self.get_shot_globals(file_path)
    num_images = params["camera_images_per_shot"]
    exposure = params["camera_exposure_time"] * 1e6
    gain = params["camera_gain"]

    for camera in self.cmos.camera_list.values():
        camera.set_acquisition_properties({"exposure_time": exposure})
        camera.set_static_properties({"gain": gain})

    self.cmos.listen_for_hardware_trigger(image_count=num_images, line=2, delay=0)

    print("done")

def transition_to_static(self, h5_filepath):
    """To be overridden by subclasses. Do any post processing after a
    shot, eg computing optical depth, fits, displaying images, saving
    images and results to the h5 file, returning cameras to an idle
    state."""
    print("transition to static")
    file_path = h5_filepath
    image_path_3D = self.get_image_folder(
        self.get_exp_folder_name(file_path), "LoadingTest3DMOT"
    )
```

```
        project_name = self.get_shot_name(file_path)
        print("Stopping listening for hardware...")
        self.cmos.stop_listening_for_hardware_trigger(
            path=image_path_3D,
            project_name=project_name,
            filetype="png",
            asyn=True,
            save=True,
            group=file_path,
            timeout="INF",
            from_control_system=True,
        )

        print("Stopped listening for hardware trigger")

    def abort(self):
        """To be overridden by subclasses. Return cameras and any other state
        to one in which transition_to_buffered() can be called again. abort()
        will be called if there was an exception in either
        transition_to_buffered() or transtition_to_static(), and so should
        ideally be written to return things to a sensible state even if those
        methods did not complete. Like any cleanup function, abort() should
        proceed to further cleanups even if earlier cleanups fail. As such it
        should make liberal use of try: except: blocks, so that an exception
        in performing one cleanup operation does not stop it from proceeding
        to subsequent cleanup operations"""
        print("abort")


if __name__ == "__main__":

    # How to run a camera server:

    port = 8765
    print("starting camera server on port %d..." % port)
    server = CameraServer(port)
    server.shutdown_on_interrupt()
```

A.4 Summary of bug fixes

A.4.1 Summary of bug fixes during the installation process

- Installing labscript-suite:
  - Create a conda environment with Python 3.8
  - Follow the instructions: https://docs.labscriptsuite.org/en/latest/installation/regular-anaconda/ to install the necessary packages for the labscript suite and create a profile
- Important links:
  - Documentation of labscript-suite: https://docs.labscriptsuite.org/en/latest/
  - Labconfig file: https://docs.labscriptsuite.org/projects/labscript-utils/en/latest/labconfig/
- Setting up folder structure:

  https://groups.google.com/g/labscriptsuite/c/LL1j-IcsFpg/m/xtFXh5SKAgAJ. Importantly, to create the first connection table of a new experiment first run the connection_table.py on runmanager and copy-paste the resulting .h5 file into the BLACS connection_table.h5 folder after renaming it.
- To change the shot output folder on Runmanger, one needs to change the argument for output_folder_format under [runmanger] in the config .ini file. In the original version of the Labscript suite the makers have not included the script_name as an argument to generate the shot folder path. To add this argument we modify anaconda3/envs/dummy/Lib/site-packages/runmanager/__init__.py by changing line 656 with the addition of script_basename=script_basename argument.
- To prevent printing the hg (mercurial) warning on Runmanager terminal, we can disable the save_hg_info in the labscript file (anaconda3/envs/dummy/Lib/site-packages/labscript/labscript.py) by changing boolean save_hg_info in class labscript_cleanup and compiler i.e. line 2542 and 2562 respectively to False.
- Instead of repeating the connection_table in the program, we can simply import the connection_table file using the import_or_reload function. For eg:

  from labscript_utils import import_or_reload

  Import_or_reload('labscriptlib.Rydberg.connection_table')

- To change the update time of the shot output folder in Runmanager (current update time is 15 seconds) modify the script:

  anaconda3/envs/dummy/Lib/site-packages/runmanager/__main__.py by changing the value of sleep time in the function rollover_shot_output_folder on line 2461.

A.4.2 Summary of bug fixes while interfacing FunctionRunner

- The labscript suite is designed in a way such that only hardware instructions can be realized once the experimental shot has begun. However, some devices need software initialization too. To cater to this need, labscript suite developers have included a functionality called FunctionRunner which behaves as a 'device' object and can execute software instructions.
- An important note for FunctionRunner is that this can only be implemented at the beginning or/and at the end of the shot execution.
- Here's the source code for the FunctionRunner class: link
- While adding the function, I ran into the issue of the function name being encoded in binary string format and this error
- After tracing the error, I figured that in 'blacs_worker.py' file for FunctionRunner, the deserialise_function_table function extracts the name of the added function. To resolve the error, simply re-assigning the name variable as name.decode('utf=8') (on line 40) works.

A.4.3 Summary of bug fixes while interfacing Pseudo-clocks

- For a long time, we were using analog_card1 in both manual and buffered mode with a pseudo-clock connected to terminal PFI0 as shown below:

```
NI_PXIe_6739(name='analog_card_1', parent_device=prawn.clocklines[0],
     clock_terminal='/PXI1Slot2/PFI0',
     MAX_name='PXI1Slot2')
```

- Analog_card2 is wired in the same way as analog_card1 so ideally giving the appropriate clock terminal (/PXI1Slot3/PFI0) should work. However, on executing the script, the labscript suite throws an error which is related to the lack of a pseudo-clock signal (this has been confirmed by disconnecting the pseudo-clock from analog_card1, which throws the same error). At the end of the pulse labscript suite waits for all the signals to be executed, but due to lack of pseudo-clock, analog_card2 times out with WaitUntilDoneDoesNotIndicateDone error, implying that signals haven't been actuated by the device.

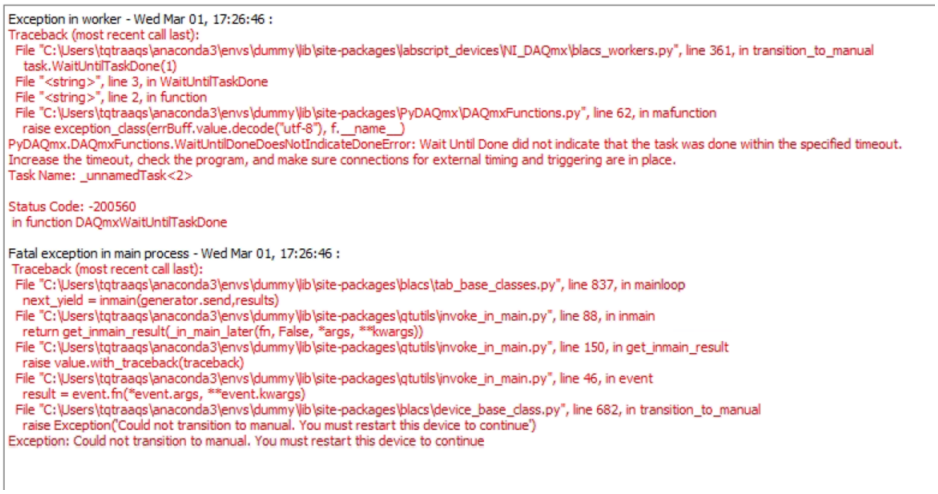


Figure A.3: Pseudo-clock integration error when analog module (NI PXIe 6739) is introduced.

- After some research, this thread and this link explain how to allow for clock mirroring between two cards. Thus, mirroring the pseudo-clock from analog_card1 to analog_card2 doesn't throw any error and works as intended.

- The links above also suggest that having independent pseudo-clocks for each card is more efficient. Thus, for now we have a working solution for two analog cards, more investigation is needed to have an independent pseudo-clock for analog_card2. I suspect that if two cards are the same, the chassis automatically assumes that the clock is shared between them and doesn't allow for an independent clock for the second card. Labscript suite's clock mirroring is simply creating that bridge using the PXI_Trig0 terminal.

- For digital_card, the first issue I faced was that the pseudo-clock can't be connected to PFI0, it needs to be connected to one of the options given below.

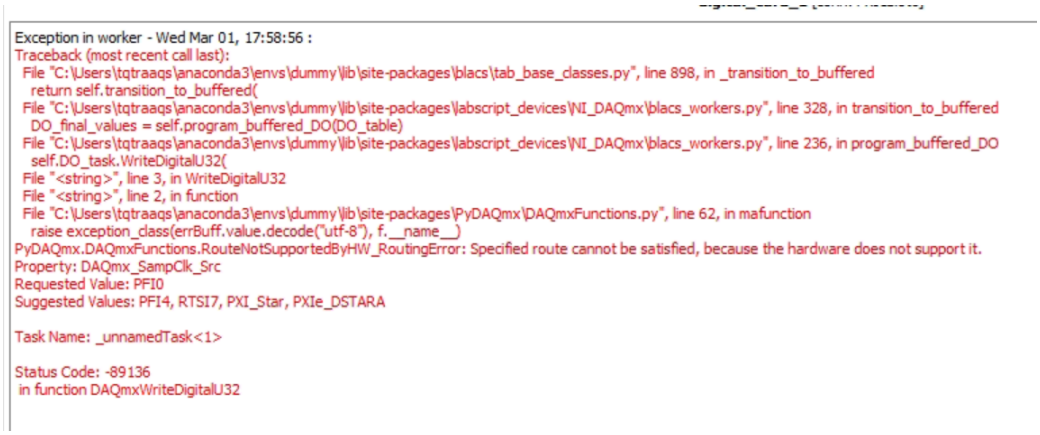


Figure A.4: Pseudo-clock integration error when digital module (NI PXIe 6537) is introduced.

- Since the PFI4 wasn't wired for the digital card, I wired the PFI4 terminal. The pseudo-clock was then accepted by the digital card.