

Imperial College of Science, Technology and Medicine
Department of Physics

Full stack development toward a trapped ion logical qubit

David Bretau

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Physics at the Imperial College of London
May 2021

Declaration

I declare the work presented in this thesis as my own, and that the contribution of others is clearly referenced.

May 1, 2021

Copyright

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Abstract

Quantum error correction is a key step toward the construction of a large-scale quantum computer, by preventing small infidelities in quantum gates from accumulating over the course of an algorithm. Detecting and correcting errors is achieved by using multiple physical qubits to form a smaller number of robust logical qubits. The physical implementation of a logical qubit requires multiple qubits, on which high fidelity gates can be performed.

The project aims to realize a logical qubit based on ions confined on a microfabricated surface trap. Each physical qubit will be a microwave dressed state qubit based on $^{171}\text{Yb}^+$ ions. Gates are intended to be realized through RF and microwave radiation in combination with magnetic field gradients. The project vertically integrates software down to hardware compilation layers in order to deliver, in the near future, a fully functional small device demonstrator.

This thesis presents novel results on multiple layers of a full stack quantum computer model. On the hardware level a robust quantum gate is studied and ion displacement over the X-junction geometry is demonstrated. The experimental organization is optimized through automation and compressed waveform data transmission. A new quantum assembly language purely dedicated to trapped ion quantum computers is introduced. The demonstrator is aimed at testing implementation of quantum error correction codes while preparing for larger scale iterations [1].

Acknowledgements

- First and foremost I would like to thank Prof. Myungshik Kim for accepting me in the first cohort of the Quantum System Engineering CDT. This crazy journey would not have started without that fateful interview with him in company of Dr Mark Oxborrow and Prof. Bruno Clerckx.
- I would like to thank Prof. Winfried Hensinger for accepting me to do my doctoral project in the Ion Quantum Technology group. His vision has provided a great deal of inspiration throughout these years.
- I would like to thank Dr Florian Mintert for his help from London, his sharp comments and his energy deployed to overcome all the administrative challenges encountered through the years.
- I express my thanks to Dr Sebastian Weidt, whose multitasking capabilities, being an academic, father and a CEO, maintaining efficiency everywhere, are definitely what I would call scalable. His critical insight at our weekly meetings was regularly helpful and needed.
- I would like to thank Dr Sam Hile for his invaluable help and support across those three and a half years, from hardware design, software engineering, theoretical physics to thesis writing.
- I would like to thank Alex Owens, a true brother in arms and relentless worker whose help transformed our experiment into, well, a working one.
- I would like to thank Raphaël Le Brun-Ricalens, the hardest worker I know, for your friendship, your help, and also all these evenings of moral support getting shot together in PUBG.
- I would like to thank Dr Zak Romaszko for his passion and humor brought in the group, and for giving me this probably once in a lifetime opportunity to play MGS4. I would also like to thank Martin Siegele for his incredible work on making the next chip generation a reality, Dr Reuben Puddy for his explanations on microfabrication and allowing us to play his really fun card game and Dr Seokjun Hong for all of his time checking those traps. You guys of the micro-fab team designed a working X-junction surface ion trap and I will be forever grateful for it. I will thank the subcontracting company which fabricated it next week.
- I would like to thank Dr Tomas Navickas for basically keeping this lab in one piece for all those years, whose practical knowledge about ion trapping kept us on the right track on many occasions.

- I would like to thank Mitch Peaks and Iason Apostolatos, true friends, true geeks and great colleagues. They made these years so much more enjoyable.
- I would like to thank Christophe Valahu for his help in the lab and on my thesis. I hope you get that dream job of yours.
- I would like to thank Mark Webber for all the interesting discussions and collaboration on quantum compilation.
- I would like to thank Nick Johnson and Dr Harry Bostock, the old guard, for their help and all our great discussions.
- I would like to thank Pedro Taylor-Burdett for his hard work on the initial atomic oven designs. I sincerely hope that this new experiment of yours will succeed.
- I would like to thank Chris Knapp for his contribution to the group's atmosphere. We didn't have much occasions for collaborating because of covid, but I wish you best for your PhD.
- I would like to thank Dr Anton Ground, an inspiration to me as a scientist, engineer and person.
- I would like to thank Dr Anna Webb for mentoring me for those first few months in the group and teaching me everything to know about doing research with trapped ions.
- I would like to thank Dr Simon Webster for his help and support during my first year in the research group.
- I would like to thank all those who arrived this year: Falk Bonus, Daisy Smith, Sahra Kulmiya, Scott Mason and Dr Mariam Akhtar. You guys rock and this research group has bright days ahead of it because of you all. Have fun. I will miss the Saturday lab work.
- I would like to thank the first QSE cohort for their patience and humor.
- I would like to thank my family and friends back in France for their overwhelming support and love. Finally, I also would like to thank Pamela for her incredible patience and support. I wish to thank Rydberg despite the blatant ignorance of this cat about my thesis questions.

Chapter Contributions

The research presented in this thesis is undoubtedly a team effort. Although the contributions of other researchers are regularly mentioned throughout the thesis, this section provides a summary, per chapter, of the specific contributions to the presented work:

- Chapter 3: The implementation of the polychromatic gate and experimentally characterizing its robustness was conducted by Dr Anna Webb, Dr Simon Webster, Dr Adam Lawrence and myself. I participated in the search for implementing the gate itself, helped characterize the injected noise and measured with Dr Simon Webster the first robustness scan. The macroscopic blade trap experiment was developed over nearly fifteen years and involves a long list of people [2]. I developed the calibration simulation.
- Chapter 5: The contributions to the ProjectQ framework of existing compilation algorithms results from a collaborative work with Daisy Smith. I alone designed the QCCD syntax and the shuttling viewer software. The current pulse sequence structure was written by Christophe Valahu, based on the work of Dr Simon Webster and Dr Adam Lawrence, and I extended it to include shuttling sequences.
- Chapter 6: I designed, implemented and tested the complete software framework for laser control and monitoring. I designed the cyro-cooler software following a requirement list from Raphaël Le Brun-Ricalens. I implemented and tested the LabVIEW code with the assistance of Christophe Valahu. The core logic controlling the heater PID was written by Dr Sam Hile, but I added the pressure reading capability and memory efficient logging. I designed the GUI interface to the heater controller, built the backup system and wrote the driver between the heater and the backup modules. I designed, implemented and tested all of the automation tasks listed in Section 6.4.
- Chapter 7: The logical apparatus was built by Dr Sam Hile, Alex Owens and myself, based on a functional UHV structure previously assembled by Dr David Murgia and Dr Tomas Navickas. The “back PCB” was designed by Dr David Murgia. Dr Sam Hile designed the “Front PCB”, configured the Sinara firmware, the external DC filter box, the “top PCB” antenna and genially converted a copper tube into a functional heat exchanger. The cryogenic cooling system was designed by Raphaël Le Brun-Ricalens. Alex Owens built the atomic oven flexible bellows and the atomic oven mount. I designed the new atomic oven after several iterations, helped in the early stages by Pedro Taylor-Burdett. The latest designed also integrated ideas from Mitch Peaks. I aligned the atomic oven, and designed most of the ARTIQ code for experimental control. All the work on automation and monitoring is my own. I contributed to the laser optic path and the imaging optic implementation. At the exception of the EMCCD python driver written by Dr Sam Hile, all the work on instrumentation is my own.

- Chapter 8: The parametric excitation setup was installed by Alex Owens and I took and analyzed the data. The python framework for simulating shuttling protocol was initially, and is still, developed by Dr Sam Hile, with important contributions from Alex Owens, Sahra Kulmiya and Dr Mehrez Agnaou. I designed the compiling tool for shuttling via waveform concatenation. I studied the waveform compression and upload rate to the hardware. The successful X-junction shuttling protocol was designed by Alex Owens with the assistance of Dr Sam Hile and myself. The successful splitting/merging protocol was designed by Alex Owens. I designed and wrote the code compiling ProjectQ code into ARTIQ experiments.
- Appendix B: I designed the concept of a shuttling format, which was formalized under my supervision in a document written by Sahra Kulmiya. This appendix is a shortened adaptation of that document.

Contents

Abstract	3
Acknowledgements	5
Chapter contributions	6
Contents	9
List of Tables	17
List of Figures	19
1 Introduction	23
1.1 Quantum computing	23
1.1.1 Quantum physics and qubits	23
1.1.2 Quantum gates and quantum circuits	25
1.1.3 Tools for characterizing quantum computers	28
1.1.4 Qubit technology and connectivity	29
1.2 Applications of quantum computing	30
1.2.1 Quantum Fourier transform and related tools	30
1.2.2 Quantum simulation	32
1.2.3 The Shor algorithm	32
1.2.4 Oracles and the Grover algorithm	33
1.2.5 Numerical optimization	34
1.3 Building quantum computers	35
1.3.1 The full stack model	35
1.3.2 Quantum computers are not classical computers	35
1.3.3 Toward large scale computation	36
1.3.4 Thesis outline	37

2	Trapped ion quantum computing with magnetic field gradients	38
2.1	Trapped ion technology	38
2.1.1	Earnshaw theorem	38
2.1.2	Paul traps	39
2.1.3	Ion crystals	42
2.1.4	Surface traps	43
2.2	Quantum logic with trapped ions	45
2.2.1	Electronic states	45
2.2.2	Quantum harmonic oscillator	46
2.2.3	The interaction Hamiltonian	46
2.2.4	The Jaynes-Cummings Hamiltonian	47
2.2.5	Single qubit gates	48
2.2.6	The geometric gate	50
2.2.7	The Mølmer-Sørensen gate	51
2.3	Ytterbium as a qubit	52
2.3.1	The ytterbium atoms	53
2.3.2	The ytterbium ions	54
2.3.3	Dressed state configuration	55
2.3.4	Ion Doppler cooling	58
2.3.5	Preparation and detection	58
2.3.6	The effective Lamb-Dicke parameter	60
2.3.7	Sideband cooling	61
2.4	The QCCD architecture	62
2.4.1	Ion manipulations	62
2.4.2	Position dependent quantum logic	63
3	Robust quantum computing	66
3.1	Robust quantum gates	66
3.1.1	The phase space representation	67
3.1.2	Gate robustness characterization	67
3.1.3	Dynamical decoupling and gate robustness	68
3.2	The polychromatic Mølmer-Sørensen gate	69
3.2.1	Theory	69

3.2.2	Experimental apparatus	71
3.2.3	Gate implementation	73
3.2.4	Gate characterization	74
3.2.5	Outlook and further work	74
3.3	Calibrating a trapped ion based quantum computer	75
3.3.1	Definitions	76
3.3.2	Introduction to the calibration problem	76
3.3.3	Weak node dependency	77
3.3.4	Time based calibration simulation	78
3.3.5	Remarks on the calibration model	79
3.3.6	Calibration of an experimental entangling gate	80
3.3.7	Simulation results	81
3.3.8	Remarks and outlook	83
4	Quantum error correction	84
4.1	Motivations for QEC	84
4.1.1	Physical limitations of quantum computers	84
4.1.2	Depolarizing noise model	85
4.1.3	Detecting quantum errors	87
4.1.4	The operator measurement	87
4.1.5	Correcting error with syndrome measurements	88
4.2	Running error corrected quantum circuits	89
4.2.1	The stabilizer formalism	89
4.2.2	Logical circuits	91
4.2.3	The fault tolerance threshold	91
4.3	The surface code	93
4.3.1	Surface code stabilizer and the decoding problem	93
4.3.2	Logical qubits and logical gates	94
4.4	The color code	96
4.4.1	Description	96
4.4.2	Application to large scale trapped ion devices	97
4.5	Practical implementation of QEC	98
4.5.1	Efficient simulation by importance based sampling	98

4.5.2	Experimental effort toward 17 surface code	99
4.5.3	Hardware requirements for testing QEC in a QCCD architecture	99
5	Quantum software compilation	102
5.1	Choosing a framework for writing and compiling quantum code	102
5.1.1	Stack representation of a trapped ion quantum computer	102
5.1.2	Choice of quantum framework	104
5.1.3	The ProjectQ syntax	105
5.1.4	The ProjectQ syntax efficiency	106
5.2	Adapting a quantum software framework to trapped ion quantum computers	106
5.3	Quantum circuit optimization	108
5.3.1	Gate reduction	109
5.3.2	Gate decomposition	109
5.3.3	Gate commutation	110
5.4	Ion router	111
5.4.1	Existing ion trap QPU provider	112
5.4.2	The QCCD syntax	112
5.4.3	Routing in a X-junction	115
5.4.4	Visualizing the QCCD code	118
5.4.5	Routing problems	118
5.4.6	Future work	119
5.5	Hardware instructions	119
5.5.1	The OpenPulse syntax	120
5.5.2	The PulseSequence syntax	121
5.5.3	Future work	125
5.6	Compiling QEC codes	126
5.6.1	Toward large scale compilation	126
5.6.2	Shuttling requirement for running a stabilizer	128
6	Automation for quantum experiments	130
6.1	Laser controller	130
6.1.1	Introduction	130
6.1.2	Problem assessment	131
6.1.3	Organization	131

6.1.4	Future work	134
6.2	Cryogenic controller	135
6.2.1	Hardware setup and controller goals	135
6.2.2	Risk assessment	135
6.2.3	Organization	136
6.2.4	Calibration	137
6.2.5	Use case	138
6.2.6	Future work	138
6.3	Vacuum system oven controller	139
6.3.1	Problem assessment	139
6.3.2	Organization	140
6.3.3	Future work	142
6.4	Automated quantum experiment control	142
6.4.1	Ion recrystallization	142
6.4.2	Automated experiment integration	143
6.4.3	Long term automation and outlook	143
7	The logical qubit apparatus	144
7.1	Ultra high vacuum chamber and content	144
7.1.1	Vacuum technology	144
7.1.2	Vacuum chamber	146
7.1.3	Surface ion trap	147
7.1.4	Internal electronic setup	148
7.1.5	Helium cooling copper tube	150
7.2	ARTIQ setup	152
7.2.1	The ARTIQ framework	153
7.2.2	Electronic hardware	154
7.2.3	DAC configuration	155
7.2.4	Experimental operating system	156
7.3	Atomic oven setup	158
7.3.1	Trapping on a surface chip	158
7.3.2	Design of a new atomic oven	160
7.3.3	The fluorescence test	161

7.3.4	Design of a new atomic oven mount	161
7.3.5	Movable atomic oven structure	163
7.3.6	Atomic oven control	163
7.4	Laser setup	164
7.4.1	Laser hardware in IQT	164
7.4.2	Optical setup	166
7.5	Imaging setup	169
7.5.1	Photon collection setup	169
7.5.2	Qubit readout fidelity estimation	170
7.5.3	Software ion analysis	173
7.5.4	Laser alignment protocol	174
7.5.5	Software laser beam detection	175
8	Scalable shuttling architecture	178
8.1	Trapping	178
8.1.1	Ion lifetime	178
8.1.2	Secular frequency measurements	180
8.1.3	Micromotion compensation	181
8.2	Ion position control on a surface chip	183
8.2.1	Simulations	184
8.2.2	Prototype model implementation	185
8.2.3	Shuttling on a X-junction	188
8.2.4	Splitting and merging	189
8.3	Software integration toward a functional QPU	191
8.3.1	CPU limitations for shuttling instructions	191
8.3.2	Scalable shuttling via FPGA waveform control	193
8.3.3	Full stack software integration	193
8.4	Conclusion and future work	194
9	Conclusion	195
9.1	Summary	195
9.2	Future work	196
	Bibliography	198

A Z stabilizer circuit in the QCCD syntax 217

B Shuttling waveform format 219

B.1 Introduction 219

B.2 Specification 219

B.3 Example 221

List of Tables

3.1	Summary of the calibration nodes	82
4.1	Error corresponding to the syndrome measurement of a 3 qubits bit-flip error codes	89
5.1	List of the most popular quantum software frameworks available	105
7.1	Numerical estimation of the ion lifetime in different pressure regimes	145

List of Figures

1.1	Bloch sphere diagram	24
1.2	Generic quantum circuit	26
1.3	Quantum circuit for entangling qubits	27
1.4	Connectivity map of a superconducting quantum computer	30
1.5	Main step for QUBO optimization in a quantum computer	34
1.6	Scalable ion trap QPU module	36
2.1	Potential from hyperbolic RF electrodes in a Paul trap	40
2.2	Stability diagram simulation	42
2.3	Surface trap geometry and rotation electrode influence on potential	44
2.4	Sideband and carrier transitions	49
2.5	Ytterbium energy level for ionization	53
2.6	Pumping cycle for $^{174}\text{Yb}^+$ and $^{171}\text{Yb}^+$	54
2.7	Dressed state and noise representation	57
2.8	Preparation and readout cycle for $^{171}\text{Yb}^+$	59
2.9	Pulses to drive the Mølmer-Sørensen with magnetic field gradient	61
2.10	Illustration of the advantages of long-wavelength radiations with surface chips containing three X-junctions and six gate zones	65
3.1	Pulses required for different number of tones of a polychromatic Mølmer-Sørensen gate	71
3.2	Blade trap experiment	72
3.3	Instantaneous RF power evolution in two tones Mølmer-Sørensen gate	74
3.4	Experimental measurements of polychromatic Mølmer-Sørensen gate robustness against heating and frequency drifts	75
3.5	Calibration node description	76

3.6	Examples of calibration node	77
3.7	Calibration graph for configuring a magnetic field gradient entangling gate	81
3.8	Simulation results of calibration impact on available computing time	82
4.1	Classical noise channel	85
4.2	Depolarizing error channel	86
4.3	Comparison between a physical quantum circuit and its logical equivalent	91
4.4	Surface code representation	94
4.5	Surface code decoding	94
4.6	Reduced single logical qubit encoded with the surface code	95
4.7	Logical T and S gates by consuming a magic state	96
4.8	Single logical qubit encoded with the color code	97
4.9	Transversality of logical gates for color qubits	97
5.1	Generic layer structure for quantum computers	104
5.2	Adaptation of a quantum circuit into the ProjectQ quantum code syntax.	105
5.3	Layer structure of the quantum compilation architecture implemented in the research group . .	107
5.4	Quantum code decomposition of the GHZ function.	108
5.5	Cancellation and merging identities.	109
5.6	CNOT gate decomposition with the ion trap gateset.	110
5.7	H gate decomposition with the ion trap gateset.	110
5.8	QCCD code illustration of a shuttling sequence, implementing an entangling gate in a X-junction between two distant ions	116
5.9	QCCD code visualization software in a X-junction	118
5.10	Pulse and Command syntax in OpenPulse	120
5.11	Hardware compilation of quantum code into a Pulse sequence	122
5.12	Example of allocation for a single qubit quantum circuit run on a three qubits quantum computer	126
5.13	ProjectQ extension to compile QEC code	127
6.1	Diagram representation of the laser control framework	132
6.2	Laser controller front panel GUI	133
6.3	Master laser controller GUI	134
6.4	Cryogenic cooler hardware	136
6.5	Cryogenic temperature controller GUI	137

6.6	PT100 sensitivity	138
6.7	Screenshot of the cryogenic controller during trapping runs	139
6.8	Screenshot of an online monitor covering a successful bake	140
6.9	Baking software monitor GUI	141
6.10	System diagram of the baking system	141
7.1	Vacuum chamber picture and content description	146
7.2	X-junction picture and layout description	147
7.3	Internal electronic setup of the vacuum chamber	149
7.4	Patch antenna simulation for in-vacuum MW emission	149
7.5	Cooling copper tube in the vacuum chamber	150
7.6	Content of the vacuum chamber during assembly	151
7.7	Cooling sequence in the logical qubit apparatus in UHV conditions	152
7.8	Automation levels for physics experiments	156
7.9	Different atomic oven behaviors and their impact on trapping	159
7.10	Simulation of the atom ballistic behavior in the atomic oven	160
7.11	Final design of the atomic oven used in the experiment	161
7.12	Fluorescence experiment description	162
7.13	Atomic oven characterization	162
7.14	Atomic oven position adjustment via fluorescence measurement	164
7.15	Diagram of the atomic oven controller box	165
7.16	Allan deviation plot for the different laser frequencies via a two-hour continuous measurement of the wavelength	166
7.17	Optical setup of the laser beams sent to the vacuum system	167
7.18	Detection fidelity estimation for the PMT setup	173
7.19	Single trapped $^{174}\text{Yb}^+$ ion picture	174
7.20	Ion recognition via Hough circle feature detection	174
7.21	Alignment procedure diagram of a laser beam into a surface chip	175
7.22	Picture of the UV laser beam overlaid onto the chip	176
7.23	Automatic scan over the camera focus	177
8.1	Ion statistic obtained from more than three hundred registered ions	179
8.2	Secular frequency measurements against the RF voltage	180

8.3	Micromotion impact on the linewidth spectrum	181
8.4	Micromotion compensation along the z axis	182
8.5	Example of concatenation of linear shuttling sequences	185
8.6	Sequence diagram of compiling a shuttling order across an micro fabricated chip	186
8.7	Waveform corrections between the generic and the required shuttling sequence	187
8.8	Syntax representation used for parsing shuttling instructions	189
8.9	Shuttling sequence to approach the X-junction center	190
8.10	Shuttling sequence to split a ion crystal of even numbers of ions	190
8.11	Splitting protocol measured by the EMCCD camera	191
8.12	Compilation time taken by ARTIQ to upload shuttling waveforms into the FPGA	192
9.1	Surface chip with buried current wires	197
A.1	Compilation of the Z stabilizer circuit into the trapped ion gate set	217

Chapter 1

Introduction

1.1 Quantum computing

1.1.1 Quantum physics and qubits

We define qubits as a two level quantum systems whose states are defined by convention with a ground state $|0\rangle$ and an excited state $|1\rangle$, similar to the classical bit of information encoded on the binary states 0 or 1. Higher dimensional quantum systems are called qudits but will not be covered in this thesis. The qubit $|\Psi\rangle$ is formulated as:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1.1)$$

with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. A helpful illustration to represent individual qubits is by reformulating α and β such as:

$$|\Psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \quad (1.2)$$

with θ is the polar angle and ϕ the azimuthal angle of the spherical coordinates of a vector of radius 1. These coordinates cover a sphere commonly called the Bloch sphere as illustrated in Figure 1.1. A single qubit can occupy any position on this sphere, the top and bottom of the sphere being $|0\rangle$ and $|1\rangle$, and any other intermediate position representing a superposition of those states.

Measuring a qubit is equivalent to projecting the qubit to either the $|0\rangle$ state with a probability $|\alpha|^2$ or the $|1\rangle$ state with a probability $|\beta|^2$.

Measurements can be done on different basis. We associate the Z basis measurements, commonly the physical measurement, with the projection on the $|0\rangle$ and $|1\rangle$ state. Projections on the X basis, associated with the

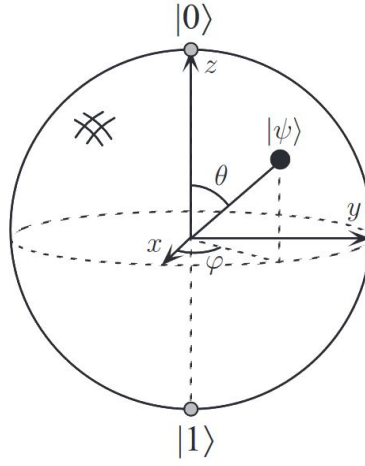


Figure 1.1: Bloch sphere representation of a two level quantum state. The quantum state can occupy any position on the sphere and is only characterized by the two angles θ and ϕ . If the state is not fully $|0\rangle$ or $|1\rangle$, then the quantum state is considered as being a superposition of those two levels. Figure taken from [3].

superposition states $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ are sometimes required in some algorithms. Let's define now two qubits $|\Psi_1\rangle$ and $|\Psi_2\rangle$. We can define the quantum register based on these two qubits as:

$$|\Psi_1, \Psi_2\rangle = \alpha|0, 0\rangle + \beta|0, 1\rangle + \gamma|1, 0\rangle + \delta|1, 1\rangle \quad (1.3)$$

Interestingly we can choose parameters such that this quantum state cannot be written as the tensor product of two individual qubits: such a situation is a physical phenomena called entanglement. One such state can be obtained for $\alpha = \delta = \frac{1}{\sqrt{2}}$ and $\beta = \gamma = 0$:

$$\Psi_{entangled} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \neq (\alpha_a|0\rangle + \beta_a|1\rangle) \otimes (\alpha_b|0\rangle + \beta_b|1\rangle) \quad (1.4)$$

An important consequence of entangled states is that measuring either of the two qubits will project both of them simultaneously.

Numerical quantum computers can encode information in qubits similar to classical information. An integer $x = \sum_{i=0}^{N-1} x_i 2^{n-1-i}$ with $x_i \in 0, 1$ can for example be represented with N qubits $|x_0\rangle, \dots, |x_i\rangle, \dots, |x_{n-1}\rangle$:

$$|x\rangle = \sum_{i=0}^{N-1} x_i |x_i\rangle \quad (1.5)$$

It is possible to form a quantum register $|\Psi\rangle$ of size N as a superposition of every possible value of x :

$$|\Psi\rangle = \frac{1}{2^{N-1}} \left(\sum_{x=0}^{2^{N-1}} |x\rangle \right) \quad (1.6)$$

The superposition of a large quantum register is a powerful tool in quantum computation: by assuming the evolution operator as a function $f(x)$ with x being the encoded integer described above, a single application of the evolution operator over a superposition state of the entire register will produce a superposition of all the results $f(x)$.

1.1.2 Quantum gates and quantum circuits

The time evolution of a quantum state $|\Psi(t)\rangle$ subject to a set of interactions represented by the Hamiltonian operator H can be formulated with the Schrödinger equation:

$$i\hbar \frac{d}{dt} |\Psi(t)\rangle = H |\Psi(t)\rangle \quad (1.7)$$

By assuming a time independent Hamiltonian and knowing the initial state $|\Psi(t_0)\rangle = |\Psi_0\rangle$ we can model the time evolution of the quantum state at time t :

$$|\Psi_t\rangle = e^{-iH.t/\hbar} |\Psi_0\rangle = U(t) |\Psi_0\rangle \quad (1.8)$$

Where $U(t)$ is the evolution operator after a time t . Using a quantum system to formulate and solve a problem means encoding an algorithm in the evolution operator U of a quantum system, using a known input quantum register. The solution of that problem is obtained by measuring the probability distribution of that quantum register after application of the evolution operator. This direct approach of re-engineering the complete Hamiltonian for each new problem is called adiabatic quantum computing but presents scalability issues and an absence of intermediate representations.

A very common approach is to decompose the total evolution operator U into a combination of primitives that are supposed to be easier to physically implement. This gate based formulation mirrors greatly classical computing and its use of logical NOT and AND gates as the foundation for building any possible algorithm. Equivalent to its classical counterpart, a quantum device is capable of performing universal quantum computation if its available gate set allows to build any U_{tot} for any qubit size register.

Gates are therefore unitary evolution operator applied to subset of qubits in the register. A gate applied to N qubits is commonly called a N qubit gate and is characterized by a Matrix $M \in SU(2^N)$. Gates are usually represented in a diagram called a quantum circuit as illustrated in Figure 1.2, where horizontal lines represent the quantum registers, usually defined in the $|0\rangle$ state on the left and measured on the right. For simplicity in this section the horizontal line displayed will represent a single qubit.

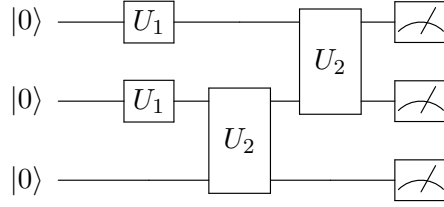


Figure 1.2: Generic representation of a quantum circuit. Qubits are usually initialized in the $|0\rangle$ state, then gates are applied to evolve the quantum states according to the user intent. Here are displayed a single qubit gate U_1 and a two qubit gate U_2 . The qubit measurement operation is represented on the right.

Important single qubit gates are the rotation gates R_x, R_y, R_z depending of a rotation angle θ :

$$[R_x(\theta)] = \begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}, [R_y(\theta)] = \begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}, [R_z(\theta)] = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \quad (1.9)$$

We define the Pauli matrices $\sigma_x, \sigma_y, \sigma_z$ the corresponding rotation gates for an angle π up to a global phase:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (1.10)$$

Finally we consider the Hadamard gate H which translates between superposition states of the X basis and the bare states of the Z basis:

$$[H] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (1.11)$$

More importantly, where single qubit gates can generate state superposition, two qubit gates can implement entanglement. The most famous two qubit gate is the CNOT gate:

$$\begin{array}{c} \bullet \\ | \\ \oplus \end{array} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (1.12)$$

A useful entangling gate for trapped ions is called the R_{xx} gate:

$$[R_{xx}(\theta)] = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos(\theta) & 0 & 0 & -i \sin(\theta) \\ 0 & \cos(\theta) & -i \sin(\theta) & 0 \\ 0 & -i \sin(\theta) & \cos(\theta) & 0 \\ -i \sin(\theta) & 0 & 0 & \cos(\theta) \end{bmatrix} \quad (1.13)$$

The entangled state defined in equation 1.4 can be produced by a circuit composed of a Hadamard gate and a CNOT gate as displayed Figure 1.3 (Left). Larger entangled states can be produced by increasing the number of applied CNOT gates (Right).

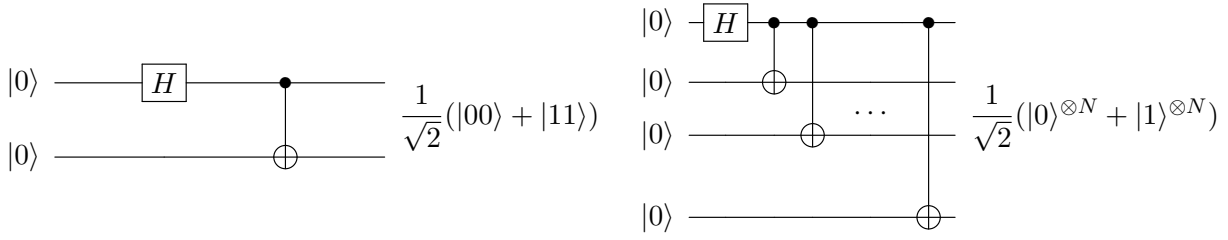


Figure 1.3: Quantum circuit generating (Left) the entangled state defined in equation 1.4 and (Right) the generalized version for N entangled qubits.

The CNOT entangling gate can be reformulated as a $\text{Control}(\sigma_x)$ gate, where Control indicates the application of a gate to a set of qubits only if a qubit selected to be the control qubit is in the $|1\rangle$ state. Furthermore any N qubit gate U can be transformed into a “Controlled” version as illustrated below in equation 1.14:

$$|\psi\rangle \text{ --- } \bullet \text{ --- } \left[\begin{array}{c} \vdots \\ \vdots \\ \vdots \end{array} \right] \left[\begin{array}{c} \vdots \\ \vdots \\ \vdots \end{array} \right] U \left[\begin{array}{c} \vdots \\ \vdots \\ \vdots \end{array} \right] = \left[\begin{array}{cc} I^{\otimes N} & 0^{\otimes N} \\ 0^{\otimes N} & U \end{array} \right] \quad (1.14)$$

with 0 the 2×2 null matrix and I the 2×2 identity matrix. Decomposing such control gate, and more generally decomposing any N qubit gates into single and two qubit gates is outside the scope of this thesis but efficient algorithms can be found in [4].

Two common gate sets capable of universal quantum computations are the superconducting gate set containing $CNOT$, H and R_z and the trapped ion gate set containing R_x, R_y and R_{xx} . Gate representation is the common but not only approach taken for modeling and implementing quantum logic. Finally the capability of building an operational quantum computer, using qubits and gate based interactions, can be reduced to a minimum list of hardware requirement that are today known as the DiVincenzo criteria [5]:

- A scalable physical system with well characterized qubit: each qubit must be physically encoded on a quantum phenomena that require proportionally less resource to build when increasing the system size.
- The ability to initialize the state of the qubits to a simple fiducial state: the qubits must be able to be reset, with its simplest application being the initialization in the $|0\rangle$ state.
- Long relevant decoherence times: the qubit should keep its information for the duration of the quantum computation. Quantum states are usually fragile and loose their properties after a defined amount of

time. Coherence is described in the next section, and its wider consequences for quantum computation in Chapter 4.

- A “universal” set of quantum gates: as described above, the universal gate set allows to build up an arbitrary evolution operator over any desired qubit register.
- A qubit-specific measurement capability: the physical system hosting the qubits needs to be able to project the state of a chosen qubit at the required time and record that measurement.

1.1.3 Tools for characterizing quantum computers

Metrics of quality are required in order to build and characterize quantum computers. When attempting to physically implement the evolution operator to replicate a quantum gate such as the ones above, imperfections will occur, from both the qubit itself or from experimental limitations or errors. Physical states can no longer be easily represented with the pure state bracket notation $|\psi\rangle$ and are better described by switching to the Heisenberg representation, where the operators are time dependent instead of the quantum states, described there with the density matrix ρ . In order to evaluate a physical qubit compared to its pure state version we use a metric called quantum Fidelity F defined [3, Section 9.2.2] as:

$$F(|\psi\rangle, \rho) = \sqrt{\langle\psi|\rho|\psi\rangle} \quad (1.15)$$

Secondly, the qubit itself deteriorates in time due to the environmental noise. T_1 is the relaxing time and is associated with the spontaneous decay of the $|1\rangle$ state to the $|0\rangle$ state. the T_2 time is the dephasing time and represents the ability of the qubit to stay in a superposition state. Both terms are the half time coefficient associated with an exponential decrease in time of the phenomena characterized.

Initialization and readout of a quantum state are also subject to errors and are usually considered together in term of noise by using the term State Preparation And Measurement (SPAM) errors.

Gates as well need to be characterized for each input and output, and are usually be characterized by a Fidelity metric for each gate type and the location where the gate occurs. Among many schemes available, the linear gate set tomography [6] factors out SPAM errors. Regardless of the architecture, entangling gates are usually harder to implement with high fidelity than single qubit gates.

Not only qubits and gates can be benchmarked but the quantum computer itself requires a metric allowing comparison of the efficiency of different devices regardless of the choice of technology used to build the device. For example, the number of qubits is not sufficient to represent a system capability, because it does not give

information about gate fidelity and the time decay associated to the qubit properties. Gate fidelities are not good enough as most architectures have different gate matrices with differing capabilities.

One of the most promising metric encompassing all of these difficulties is the Quantum Volume (QV) [7] which requires a quantity called the achievable depth and measured as follow: the device runs quantum circuit layers based on two components: the first step assigns the n qubits in pairs randomly, then each pair is required to run a $SU(4)$ unitary quantum gate, involving a decomposition into hardware compatible gates and additional quantum gates for implementing SWAP operations if the two qubits are not directly connected. For each circuits containing d layers the experimental device attempts to solve the Heavy Output problem [8]. The achievable depth is defined as the maximum number of layers for which the problem is solved by the device.

The Quantum Volume metric depends on the width or number of qubit used, n , as well as the achievable depth $d(n)$ such as:

$$\log_2 QV = \operatorname{argmax}_n [\min(n, d(n))] \quad (1.16)$$

1.1.4 Qubit technology and connectivity

There is several technological approaches in the choice of physical qubit and how to drive the quantum gates. The present thesis focuses in trapped ions but alternatives have also been actively pursued, with notably superconducting qubits and optical qubits. Most architectures have unfortunately constraints in how and which qubits can be interacted with by the experimental control. Some of the constraints are due to the device design itself: for example early quantum computing prototypes with only one or two qubits will be designed to force the quantum state readout on all qubits to simplify the software and hardware requirement.

Constraints can also be due to the nature of the physical qubit itself. One such constraint is the qubit connectivity map. Let's consider the worst case with superconducting qubits and silicon based qubits. The entangling capability of these qubits has to be micro-fabricated alongside the qubits themselves. Most devices built by IBM and Google for example have only nearest-neighbor connectivity maps: as displayed in Figure 1.4 with a map of an IBM machine, qubits can only interact with up to their four closest qubits.

Connectivity limitations can be overcome at the cost of additional quantum gates in the circuit: Quantum states can be swapped via a SWAP gate between two physical qubits to “translate” the qubits of interest where they

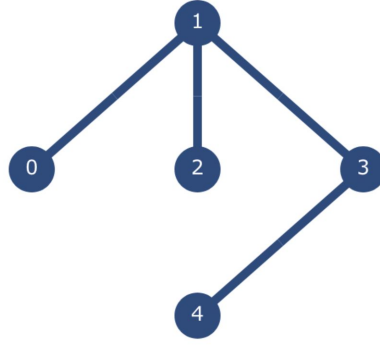


Figure 1.4: Connectivity map of the five qubits IBM quantum machine VIGO. Qubits not connected in the Figure require additional swap gates before they can be entangled. Figure taken from [9].

can interact. A SWAP gate can be build with three CNOT gates:

$$\begin{array}{c} \text{---} \times \text{---} \\ \text{---} \times \text{---} \end{array} = \begin{array}{c} \bullet \quad \oplus \quad \bullet \\ | \quad | \quad | \\ \oplus \quad \bullet \quad \oplus \end{array} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.17)$$

The low connectivity architectures described above require a significant overhead of entangling gates for running arbitrary quantum circuits. Trapped ion devices on the other hand, have by nature a fully connected map but at the additional cost of ion displacement which will be described in 2. Connectivity is also constraining the choice of quantum error correction scheme that can be implemented as described in Chapter 4.

1.2 Applications of quantum computing

This section will introduce the three main axes of application of quantum computing, by first introducing a tool commonly used by all of them: the quantum Fourier transform.

1.2.1 Quantum Fourier transform and related tools

The Fourier transform is a widely used mathematical transformation. Its discrete formulation transforms a vector $X \in \mathbb{C}^N$ into a vector $Y \in \mathbb{C}^N$ such as:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N} \quad (1.18)$$

A quantum equivalent defined as an operator QFT can be defined by replacing X and Y with the quantum register $|X\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$ and $|Y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$, with $x_i, y_j \in \mathbb{C}^2$ and $|j\rangle, |k\rangle$ quantum states each defined with n qubits such as defined in equation 1.5 :

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle \quad (1.19)$$

the Quantum Fourier transform can be represented as a unitary gate in a quantum circuit, and can be efficiently implemented with Hadamard gates and $C(R_z)$ gates [3]. By defining the primitive P_n such as:

$$\quad (1.20)$$

We can concatenate those primitive and obtain the QFT gate:

$$\quad (1.21)$$

A direct application of QFT is the Quantum Phase Estimation (QPE) algorithm: if we consider an eigenstate $|u\rangle$ of an operator U with eigenvalue $e^{2\pi i \phi}$ then we can directly retrieve the eigenvalue ϕ by encoding it in another quantum register. By considering $|u\rangle$ a N qubits eigenstate of U , with U a N qubit operator and a register of t qubits initialized at the $|0\rangle$ states, then the algorithm can be represented by the following circuit:

$$\quad (1.22)$$

The accuracy of the measurement depends on the qubit quantity t used to encode the phase. This algorithm is an important tool for designing quantum algorithms as it allows a direct and non probabilistic access to eigenvalues of interest.

1.2.2 Quantum simulation

The idea of using controllable quantum systems to simulate the evolution of other quantum systems comes from Feynman lectures [10] and represents the first formulation of quantum computers. Because the size of a quantum system exponentially increases the total information size necessary to describe it, so only small quantum systems can be modeled accurately by a classical simulation. To illustrate quantum simulation we will consider the problem of simulating a molecule using a quantum computer.

The general approach is to first map the fermionic Hamiltonian representing the molecule in the spin Hamiltonian of quantum computers using for example the Jordan-Wigner transformation, then evolve the Hamiltonian in time using the Trotter decomposition and extract information of interest from the obtained eigenstates [11, 12]. Energy eigenvalues can be extracted with the QPE tool in the full configuration interaction algorithm [13]. This decomposition is unfortunately not realistically applicable to quantum hardware in the near future due to the very large number of gates needed [14].

A more recent algorithmic tool emerged as a hybrid classical/quantum approach, designed for quantum chemistry simulation on near term quantum hardware: the Variational Quantum Eigensolver (VQE) [15]. The ground state $|\phi_0\rangle$ of the fermionic Hamiltonian $H_{tot} = \sum_i H_i$ is searched by minimizing $\frac{\langle\phi|H_{tot}|\phi\rangle}{\langle\phi|\phi\rangle}$. The variation of $|\phi\rangle$ for a parameter θ is implemented via an ansatz circuit, and the choice of θ is determined at each iteration classically. The appeal of VQE for near term devices is that each element H_i can be run on a circuit independently. It considerably reduces the circuit size at the expense of having to run many more instances of quantum circuits. VQE efficiency is strongly linked to the selected ansatz circuit, which can be improved by building the ansatz itself inside the algorithm [16].

VQE simulation of small molecules have already been implemented on superconducting and trapped ion quantum computers [17, 18] with the numerical accuracy required for practical application.

1.2.3 The Shor algorithm

Defining $N = p_1 p_2$ with p_1 and p_2 two prime numbers, we consider the problem of finding those factors by only knowing N . The best known classical algorithm runtime grows sub-exponentially when N gets larger [19]. On the other hand the Shor algorithm [20] can find a factor with a high probability $p = 3/4$ with a polynomial time growth, achieving therefore, an exponential speedup with quantum computers compared to their classical counterpart.

The Shor algorithm works as follow: We first choose a positive integer $x < N$ randomly. For most cases when

x is not a factor of N , we then search the smallest integer r for which $x^r = 1 \pmod{N}$. Provided r is found and even, then either the greatest common divisor (gcd) of $(N, x^{r/2} - 1)$ or $\gcd(N, x^{r/2} + 1)$ will be a factor of N [3].

Finding r is achieved via Quantum phase estimation, applied on the modular exponentiation operator $U_{x,N}$ with parameter x and N :

$$U_{x,N}|y\rangle \equiv |xy \pmod{N}\rangle \quad (1.23)$$

Its eigenstates $|u_s\rangle$ have for eigenvalues $\exp(2\pi i s/r)$ but as we do not know how to implement them, the algorithm is run with the input state $|1^n\rangle$ state which is straightforward to produce and is as a matter of fact the superposition of all the $|u_s\rangle$ states [3]. The measurement will provide several s/r with s a variable integer and r constant, allowing to deduce r . The circuit construction for $U_{x,N}$ is outside the scope of this thesis but several recipes can be found in [21].

A direct consequence of the Shor algorithm is that asymmetric encryption, which is at the foundation of most security protocols on the internet, becomes vulnerable: The RSA encryption [22] builds its keys based on a product of two prime numbers, and the standard 2048 bit keys would be compromised with quantum computers having at least 4096 qubits.

1.2.4 Oracles and the Grover algorithm

Quantum computers can be used to find solutions to an optimization problem with faster polynomial speed to their classical counterpart with a quantum search algorithm called the Grover algorithm. We define a set of possible solutions x where only an unknown subset satisfy a given problem. We define an Oracle O as an operator associated to a function $f(x)$, returning 1 if x is a solution and 0 otherwise. The operator has the following effects [3]:

$$O|x\rangle|q\rangle = |x\rangle|q \oplus f(x)\rangle \quad (1.24)$$

We also define the conditional phase shift gate D over a qubit register such as $D|x\rangle = -|x\rangle$ if $x > 0$. For example the one qubit gate version of D is the Pauli operator σ_z . Provided a construction of O is given, we can build a Grover iterate $G = H^{\otimes n} D H^{\otimes n} O$. The Grover algorithm extracts with high probability the state $|\phi_i\rangle$ satisfying the solution $f(x)$ from a general superposition $|\phi\rangle = \sum_i |\phi_i\rangle$ by applying a number of Grover iterates dependent on the initial amplitude of $|\phi_i\rangle$ [3].

Although remarkable in itself, this algorithm can be extended to two other tools. The first one is Quantum amplitude algorithm, which is a Quantum phase estimation algorithm applied to the Grover iterate. This algo-

rithm measures the amplitude probability of the set of states associated with the oracle. It is straightforward to recognize that Quantum amplitude estimation is a natural first step to configure the Grover algorithm itself. The second tool is the Quantum counting algorithm, which is simply the Quantum amplitude estimation output multiplied by the number of possible states, producing the number of states which are associated to the oracle.

1.2.5 Numerical optimization

The abstract concept of the oracle can be exploited for numerical optimization problems as long as it can be reliably built. A profound generalization of the optimization problem, which is of particular interest for quantum computing, is the Quadratic Unconstrained Binary Optimization problem (QUBO) which can be formulated as follows:

for $n \in \mathbb{N}$, $M \in \mathbb{Z}^{n \times n}$, find $\min(x^t M x) \forall x \in \mathbb{Z}^n$.

The above definition can also be reformulated as a maximization problem. There is a wide range of optimization problems that can be reformulated in the QUBO syntax, such as the maximum cut problem or the graph coloring problem [23].

Generic mapping of QUBO problems into quantum oracles is possible via canonical oracle construction [24], allowing a full algorithmic implementation and resolution of modern day optimization problems in a quantum computer as displayed in Figure 1.5.

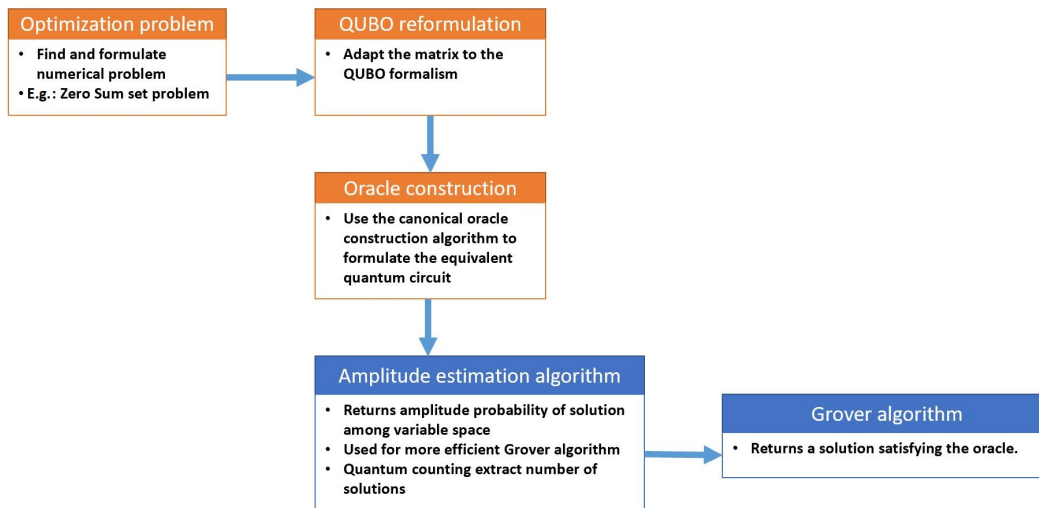


Figure 1.5: Schematic representation on solving a QUBO optimization problem with a quantum computer. Orange blocks are algorithms run by classical computer while blue blocks are run by quantum computers.

1.3 Building quantum computers

1.3.1 The full stack model

As hinted above, quantum computer operations can be represented with quantum circuits by implementing instructions on a physical device only using a restricted gate set of single and two qubit gates but it may lead, for problems unreachable by classical computers, to thousands and even millions of quantum gates. Additionally translating efficiently problems from highly abstracted representations to low level circuits to hardware systems requires a set of compilation tools operated by classical computers. The high fragility of quantum systems will require schemes detailed in Chapter 4. Correcting errors is mainly achieved through additional qubits and repeated calibration to maintain its capabilities and reducing its operational time. This thesis intends to explore how to build a fully operational quantum computer using a layer based approach from high level algorithms down to running gates, and test the tools developed by implementing the single instance of a quantum error correcting code. Similarly to classical computing where a complete approach, called full stack model, considers the entire stack of technological layers from the front end to the back end, a full stack model of the ion trap quantum computer will be considered in the thesis.

1.3.2 Quantum computers are not classical computers

The above section about possible application of quantum computers illustrates that quantum computers will probably never replace day-to-day computers Central Processing Units (CPU): being more efficient at very specific type of problems they will act as specialized hardware extension similar to Graphical Processing Units (GPU), therefore quantum computers are being increasingly called Quantum Processing Units (QPU). This thesis will use both terms equivalently.

A second remark is that there is a large variety of technological choices for building a digital quantum computers, including superconducting circuits, trapped ions, trapped atoms, photonic circuits, quantum dots and NV-center. The first four technologies have already been adopted by companies working in the field, but no devices are yet capable of providing useful calculation unreachable to classical computers. This situation can be compared to the early digital computer era, where the general technological trend moved from mechanical calculators to electromechanical calculators, to electronic calculators based on vacuum tubes. Each steps represented an improvement in computing speed and device reliability. The replacement of vacuum tubes by much more reliable and power-efficient transistors started the race toward miniaturization and the Moore law. Quantum computers are still in their infancy and the concept of a “quantum transistor”, a technological ingredient

simplifying the operation of a quantum computers, does not exist yet.

1.3.3 Toward large scale computation

The IQT group published in 2017, a blueprint [1] gathering key solutions to enable the fabrication of a large scale quantum computer. The main element concerns the microwave scheme described further in Chapter 2 2.4.2, allowing a single source to target an arbitrarily high number of qubits for conducting quantum gates in parallel. The magnetic field gradient would be generated via a high current sent into wires, micro-fabricated below the surface chip at the gate zone. The estimated current that needs to be applied is around 10 A, which would likely require cryogenic cooling in order to remove excess heat. All electronics, such as the DACs, FPGAs and control of the integrated features, will be integrated into lower layers of the chip with vertical integrated access.

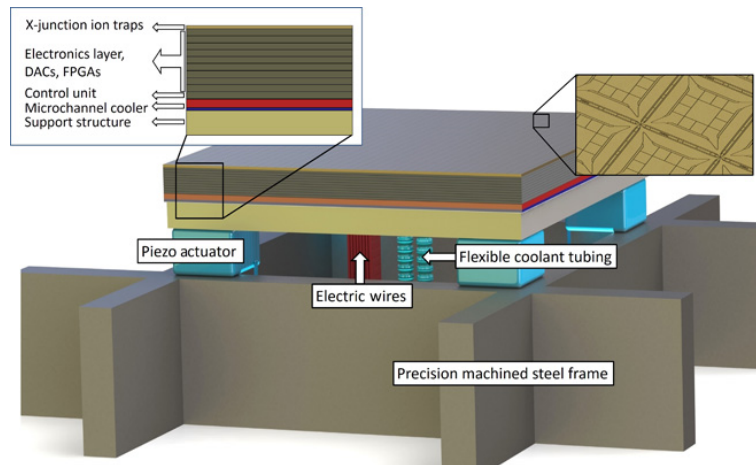


Figure 1.6: Module description of the IQT blueprint: the chip is a stack composed of the cooling structures, electronic modules and on top is the surface ion trap. The module is hosted on piezo modules for enabling position alignment to neighbour modules. Figure taken from [1]

Such integration is estimated to allow the construction of modules hosting 36×36 junctions such as displayed in Figure 1.6 which, considering a dozen ions per junction, will produce a few thousand qubits device. A single module will surpass the noisy intermediate-scale quantum (NISQ) regime [25], limited to approximately one hundred qubit. The second key feature would be to connect those modules by aligning them together via piezo modules at a micrometer scale and shuttle ions through the gap between them. This would then provide means for an arbitrarily large device to be built, allowing for large scale quantum computation.. Those key features need to be tested individually so the research group is divided into projects focusing into a subset of technology which requires experimental verification:

- A linear chip with a high magnetic field gradient to test high speed and high fidelity quantum gates
- A double surface chip system separated from each other with a piezo module to test chip-to-chip shuttling
- A cryogenic surface chip with quick chip turnover for chip benchmarking and shortcuts to adiabatic shuttling.
- An x-junction surface chip to test buried current wires for quantum gates and circuits run with position dependent quantum gates

This thesis work is centered around the X-junction surface chip experiment, but most theoretical and experimental considerations mentioned here can be applied to the others systems.

1.3.4 Thesis outline

This thesis has the following structure: Chapter 2 introduces the theoretical framework to drive quantum gates with ions using magnetic field gradients. The scalable gate architecture is presented. Chapter 3 details the implementation of a robust entangling gate and its impact on the availability of a quantum computer is modeled with a simulation of calibration scheduling. Once the theoretical foundation to build scalable and robust quantum computer is presented, Chapter 4 introduces an overview of Quantum Error Correction (QEC) codes and in particular the topological surface code, allowing the quantum computer to run the large scale quantum algorithms described in this Chapter. A review of simulation tools and early experiments is presented. Chapter 5 details the software frameworks used for implementing quantum code circuits in physical devices; basic structures and algorithm are presented for compiling arbitrary quantum circuits into hardware compatible code. Contributions to an open source framework are listed. Finally a new quantum code syntax is presented adapting the quantum circuit level of abstraction to a shuttled ion architecture and its importance in a full stack architecture is discussed. Chapter 6 describes automation projects built for improving the work efficiency of every experiment in the research group, including supervisory control and data acquisition (SCADA) projects and a list of automation tasks for calibrating the system described in Chapter 3. Chapter 7 presents the apparatus designed for testing QEC experiments. A new atomic oven structure is described and tools for software assisted measurement and calibration are considered. Chapter 8 presents early results of the apparatus. The ion trap is characterized, ion shuttling protocols are implemented and a scalable approach for a large scale ion shuttling router is detailed. Routing instructions built from Chapter 5 are discussed. Finally, the Conclusion discusses ongoing experimental preparations for implementing logic gates with a magnetic field gradient.

Chapter 2

Trapped ion quantum computing with magnetic field gradients

The type of quantum computer studied and experimentally developed in this thesis employs individual trapped ions used as single quantum bits or qubits and belong therefore to the quantum charged coupled device (QCCD) category [26]. This chapter will introduce the theory behind ion trapping and the implementation of quantum computing with trapped ions. Then the research group specialized scheme replacing laser beams by microwave (MW) and RF signals in combination with magnetic field gradients will be introduced [27], as well as the strategy adopted to build a large scale quantum computer [1].

2.1 Trapped ion technology

Single ions have electronic quantum states suitable to encode qubits. In order to control them as such, those ions need to be confined and eventually moved into specifically designed positions. This section will describe the fundamentals about trapped ion technology and its application to quantum computing.

2.1.1 Earnshaw theorem

Ions can be considered as charged particles in space. In a 1 dimensional case, a charged particle c_1 with charge q_1 will give rise to an electric potential at any position r equal to:

$$V(r) = \frac{1}{4\pi\epsilon_0} \frac{q_1}{r} \quad (2.1)$$

A particle c_2 with charge q_2 at a position r will experience a force equal to:

$$\vec{F}_{c_2}(q_2, r) = q_2 \vec{E}(r) = -q_2 \nabla V(r) \propto \frac{q_1 q_2}{r^2} \quad (2.2)$$

With $\vec{E}$ the electric field which is a vector related to the electric potential via $\vec{E}(r) = -\nabla V(r)$. Confining a charged particle to a single point in space using only static electric fields is not possible for the following reason: the point in space in which the particle would be confined would require repulsive forces from all directions in space, which implies that $\nabla \cdot \vec{E} < 0$. However by applying the first equation of Maxwell for the field generated by a density of charge ρ :

$$\nabla \cdot \vec{E} = \frac{\rho}{\epsilon_0} \quad (2.3)$$

the field confinement not being generated at the particle position, we therefore obtain at the particle position $\rho = 0$ and so $\nabla \cdot \vec{E} = 0$. This contradiction leads to Earnshaw's theorem, stipulating that electrostatic confinement of a particle in space is impossible.

2.1.2 Paul traps

There are two different approaches regarding confining ions in space. The first one is based on initial discoveries from Penning [28] and later formalized by Dehmelt [29] and consist of combining a heavy magnetic field in combination with static electric fields and is called Penning traps. This technology does not require high RF voltage fields but generating the B field has been a major obstacle over the last years toward scalability and micro chip integration. New schemes [30] have however been proposed recently to overcome those challenges but will not be studied in this thesis.

The second technology, first proposed by Paul et al. [31] back in 1953 and later on called Paul traps, uses a combination of RF electric fields combined with static fields. The effect of the RF electric field is studied when a charged particle at location $\vec{r} = (x, y, z)$ of charge e and mass m is exposed to a RF field of frequency $\Omega/2\pi$ and amplitude V_{RF} . the equation of motion is then expressed as:

$$m \frac{d^2 \vec{r}}{dt^2} = -e \nabla V(\vec{r}, t) \quad (2.4)$$

On a 2 dimensional plane (x,y) as represented in Figure 2.1, we consider 4 hyperbolic electrodes situated at a distance r_0 from the trap center. Different electrode geometries are accounted for via a geometric factor η_{RF} with each pair of opposite electrodes applying a RF field V_{RF} with a π phase with the other pair, then the field

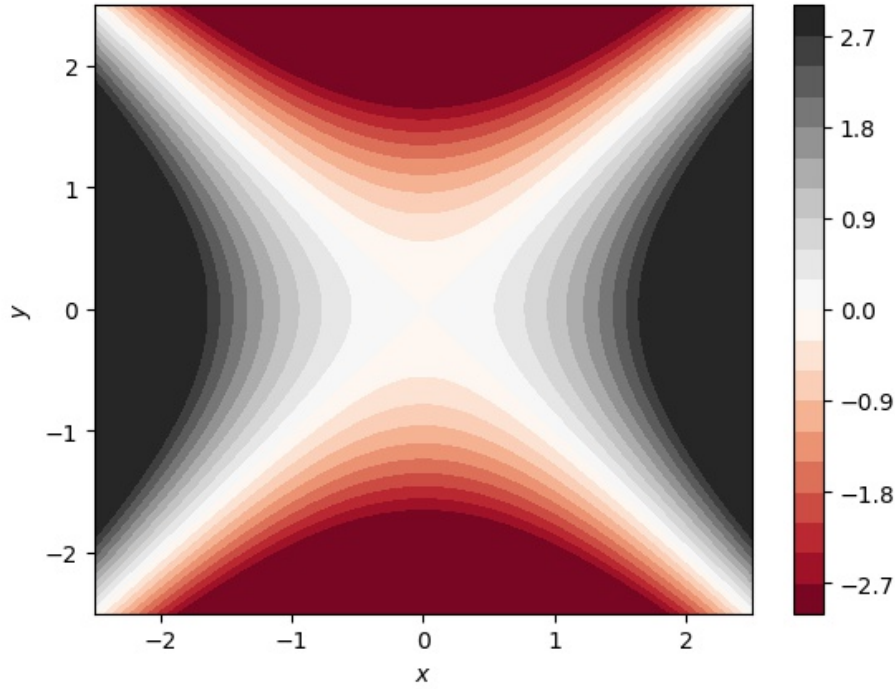


Figure 2.1: 2-D representation of the hyperbolic RF electrodes in a Paul trap and its potential in the trapping region at a certain instant in time. The neighbor electrodes have an opposite phase to each other. The endcap electrodes, allowing confinement in the third dimension, are not represented here. The displayed scale is arbitrary.

potential near the center is equal to:

$$V_{RF}(x, y, t) = \eta_{RF} \frac{x^2 - y^2}{2r_0^2} V_{RF} \cos(\Omega t) \quad (2.5)$$

We then apply a static field potential V_{DC} on the z axis with 2 endcap electrodes with hyperbolic shapes situated at $(0,0,z_0)$ and $(0,0,-z_0)$. As for the RF electrodes, different geometries are simplified by the geometric factor η_{DC} . The static potential near the center is equal to:

$$V_{DC}(x, y, z) = \eta_{DC} \frac{2z^2 - (x^2 + y^2)}{2z_0^2} V_{DC} \quad (2.6)$$

We can now solve the equation of motion for the two contributions $V(x, y, z, t) = V_{RF}(x, y, t) + V_{DC}(x, y, z)$:

$$\frac{dx^2}{dt^2} = -\frac{e}{m}x \left(\frac{\eta_{RF}V_{RF} \cos(\Omega t)}{r^2} - \frac{\eta_{DC}V_{DC}}{z_0^2} \right) \quad (2.7)$$

$$\frac{dy^2}{dt^2} = \frac{e}{m}y \left(\frac{\eta_{RF}V_{RF} \cos(\Omega t)}{r^2} + \frac{\eta_{DC}V_{DC}}{z_0^2} \right) \quad (2.8)$$

$$\frac{dz^2}{dt^2} = -\frac{e}{m} \frac{2z}{z_0^2} \eta_{DC}V_{DC} \quad (2.9)$$

The z axis can be solved as a harmonic oscillation $z(t) = z_{amp} \cos(\omega_z t)$ with ω_z being called the axial frequency and being defined as:

$$\omega_z = \sqrt{\frac{2e\eta_{DC}V_{DC}}{mz_0^2}} \quad (2.10)$$

the x and y axis equations of motion are equivalent to the Mathieu equation [32] formulated as:

$$\frac{dx^2}{dt^2} + (a_x - 2q_x \cos(2\tau))x = 0 \quad (2.11)$$

By identifying equation 2.7 with equation 2.11 we calculate:

$$a_x = \frac{4e\eta_{DC}V_{DC}}{m\Omega^2 z_0^2} \quad (2.12)$$

$$q_x = \frac{2e\eta_{RF}V_{RF}}{m\Omega^2 r_0^2} \quad (2.13)$$

$$\tau = \frac{\Omega t}{2} \quad (2.14)$$

A very useful solution of the Mathieu equations for $a_x = 0$ and q_x small is:

$$x(\tau) = x_0 \cos\left(\frac{q_x \tau}{\sqrt{2}}\right) \left(1 - \frac{q_x}{2} \cos(2\tau)\right) \quad (2.15)$$

This solution contains a main oscillation term, called secular oscillation and an additional oscillation, faster and smaller in amplitude, which is commonly called micromotion oscillation. The radial secular frequency ω_r is deduced from equation 2.13 and 2.14 and is equal to:

$$\omega_r = \frac{e\eta_{RF}V_{RF}}{m\Omega r_0^2} \quad (2.16)$$

When an ion is confined in a Paul trap, micromotion is, in general, a detrimental factor and needs to be minimized.

More generally, the solution space covered by a_x and q_x contains two types of solutions [33] as shown in Figure

2.2: there are aperiodic solutions which corresponds to unstable ion trap configurations where no ions can be trapped and confined, and periodic solutions with a behavior similar to the solution described in equation 2.15 where ions oscillate and stay confined in the trap.

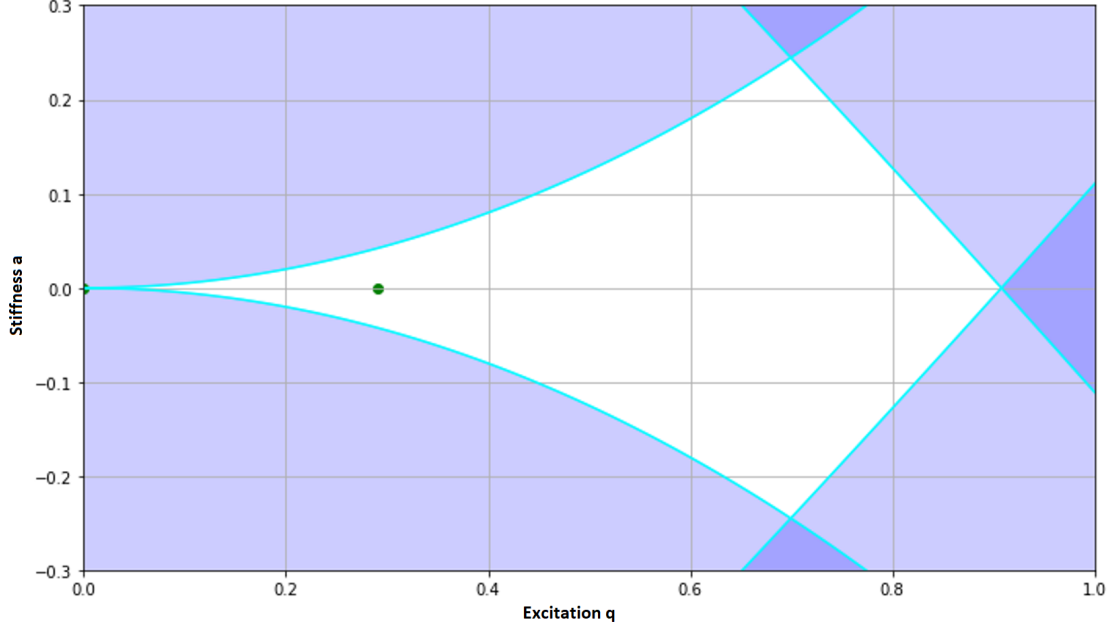


Figure 2.2: Simulation of the stability diagram in the space (a, q) for a hyperbolic linear trap: solutions of the Mathieu equations in the shaded area are aperiodic or unstable, whereas the white area is periodic/stable. The green point represents the experimental configuration used to trap ions in the experimental setup described in Chapter 7. The simulation was written by Dr Sam Hile based on [34].

2.1.3 Ion crystals

The solutions described above only considered a single ion, however a single harmonic potential can trap multiple ions simultaneously. Each ion will be affected both by the trapping potentials and by the Coulomb force of equation 2.2 from every other ions. By only considering the weak axial confinement alongside the z axis, we can calculate [35] the ion positions $x_n(t)$ by considering the potential energy of each individual oscillator $V_{ind,m} = \frac{1}{2}\omega_z^2 x_m(t)^2$, with ω_z the trap axial frequency and the potential energy for each interaction $V_{int,n,m} = \frac{e^2}{8\pi\epsilon_0} \frac{1}{|x_n(t) - x_m(t)|}$. The total potential energy over a chain of N ions is written as:

$$V_{tot} = \sum_{m=1}^N \frac{1}{2}\omega_z^2 x_m(t)^2 + \sum_{n,m=1; n \neq m}^N \frac{e^2}{8\pi\epsilon_0} \frac{1}{|x_n(t) - x_m(t)|} \quad (2.17)$$

We solve those positions at equilibrium for $x_m(t) = x_m^0$ when $\left[\frac{\partial V}{\partial x_m} \right]_{x_m=x_m^0} = 0$ with M the ion mass.

Analytic solutions give chains of static positions called ion chains or Coulomb crystals. Among the noticeable outcomes we can observe that first, the inter-ion distance is not homogeneous as ions in the center of the chain are closer together than at its extremities. Secondly, we can see that the minimum inter-ion distance d_{min} decreases with the number of ions N in the chain with the following relation [35]:

$$d_{min}(N) = \left(\frac{e^2}{4\pi\epsilon_0 M \omega_z^2} \right)^{1/3} \frac{2.018}{N^{0.559}} \quad (2.18)$$

Dynamic study of an ion chain reveals additional vibrational eigenmodes due to the Coulomb interactions. The number of additional eigenmodes scales linearly with the number of ion and their frequencies can be calculated analytically as well. In the simple case of a two ion chain, the center of mass (COM) mode with frequency ω_z is in phase with the movement of the two ions. The additional mode is called the stretch mode and corresponds to out-of-phase movements between the two ions and its frequency is calculated as $\omega_s = \sqrt{3}\omega_z$.

Using large single ion chains as a Quantum processing unit is a straightforward application of the concepts detailed above, with existing systems being able to entangle ion chains of up to 24 ions without single ion control [36], up to 20 ions with single ion control [37] and single qubit addressing in a single ion crystal of up to 79 ion [38]. Individual ions in the chain are usually interacted with dedicated laser beams [39]. Cooling and preparation time can remain constant regardless of the ion chain [40] as well as gate fidelity [41], but the gate time scales unfavorably with it. A quantum computer defined as a single ion chain has a full connectivity map although gates have fidelities and speed improving as the involved ions are closer inside the chain [42]. The choice of qubit position in the chain is optimized before the circuit is run.

Larger ion chains require larger laser setup as well as a higher level of control to maintain gate fidelity and readout. Finally collision rates, which will be detailed later in Chapter 7, constrain the ion chain lifetime, which has been assessed to scale inversely with the ion chain size [43]. Trapped ion devices with a qubit register above a few hundred cannot be trapped in a single harmonic potential, therefore an alternative trapping geometry is required.

2.1.4 Surface traps

A surface trap is an electrode geometry defined in a plane or surface, comprising all RF and DC electrodes necessary for trapping, with the trapping region being above it. First implemented in 2006 [44, 45], they use micro-fabrication technologies to achieve electrode dimensions down to the micrometer. The most common

geometry as displayed in Figure 2.3 A is the two RF electrode trap, where the quadrupole RF potential is generated via only two RF electrodes separated by grounding electrodes. The RF potential above the trap is shown in Figure 2.3 B where the trapping position is above the ground electrode. Axial confinement is achieved via external DC electrodes on both sides of the RF electrodes, whose applied voltage generates a field equivalent to the harmonic potential described above with the static endcap electrodes.

As described in [46], the RF electrode geometry is chosen to adjust the ion height. The DC electrodes still implement the harmonic potential on the axial axis, but can now be numerous enough to manipulate the ion position across the axial axis. The dimensions can be adjusted to optimize ion displacement as detailed in 2.4.1. Surface geometry has however a cost due its non hyperbolic shape. The trap depth, which is the potential barrier at the ion position preventing it to escape the trap and formed by the RF and DC fields, is for equivalent applied voltages between 30 and 200 lower than for the original hyperbolic trap [47]. The stability region of a surface trap is also smaller than for the hyperbolic equivalent [48].

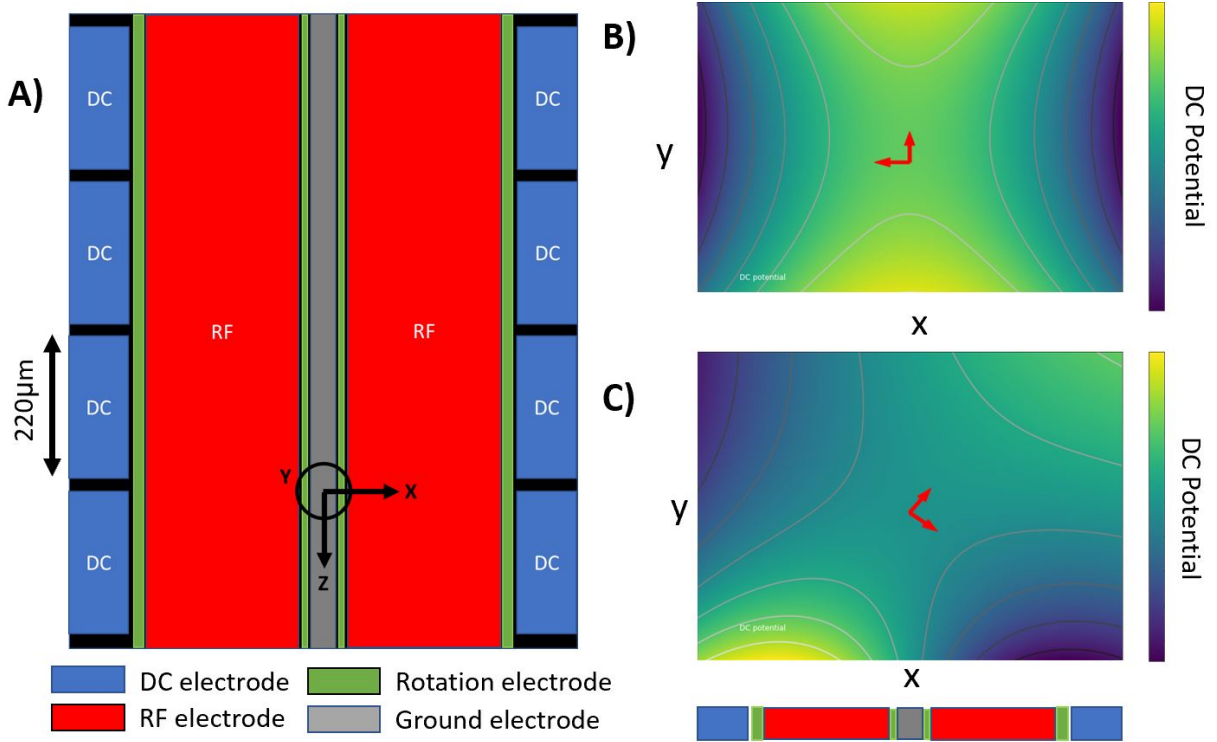


Figure 2.3: A) Top view of the 5 wires surface chip geometry identical to the linear sections of the X-junction trap used in Chapter 4. B) Simulation of the DC potential without using rotation electrodes. the eigenvectors of the DC potential at the ion height are represented in red. One axis is orthogonal to the chip so the ion motion in this direction cannot be efficiently Doppler cooled. C) By applying an asymmetric voltage on the rotation electrodes (from left to right: +3.4V/-1V/+1V/-3.4V) obtained by numerical optimization, the modified DC potential has no eigen-axis orthogonal to the chip: all of the ion axes of motion are accessible to the Doppler cooling laser beam.

It has been shown in 2.1.2 that the ion oscillates with 3 frequencies in the 3 axes of space. Laser Doppler cooling

involves cooling every of those axes. It means that the laser beam chosen for cooling cannot be orthogonal to any eigen-axis of the ion trap. The axial eigen-axis is colinear to the RF tracks, which constrains the laser beam to be at an angle of 45° with it. One of the two eigen-axis from the RF potentials, orthogonal to the RF tracks, will also be cooled by the 45° beam. However the second one, as illustrated in Figure 2.3 B, is orthogonal to the chip therefore hardly accessible to the laser beam. A solution is to implement additional DC electrodes along the RF tracks called rotation electrodes. These electrodes are meant to produce an asymmetric field such that the ion height is maintained but the trap eigen-axis are rotated enough, as shown in Figure 2.3 C to be both covered by Doppler cooling by a laser beam colinear to the chip. Asymmetric trapping field does not decrease the trapping stability region of a surface trap, and interestingly is enlarged at the angle of 45° [48].

2.2 Quantum logic with trapped ions

This section will first summarize the essential concepts of atomic physics, then, the basics of light-matter interactions, used to implement the single and two qubit gates in most ion trap devices will be covered. We will not use the operator symbol $\hat{}$ for simplicity.

2.2.1 Electronic states

We first describe a two level system representing two spin states of an ion and determine its Hamiltonian: the ground state will be represented as $|0\rangle$ and the excited state will be represented as $|1\rangle$. In anticipation of quantum computing application later in the section we consider those two states the basis to encode a qubit. We define the energy separation between the two states by $E_0 = \hbar\omega_0$. The Hamiltonian of this two level system, which we will call the spin state Hamiltonian H_s , is then defined as:

$$H_s = \frac{\hbar\omega_0}{2} (|0\rangle\langle 0| - |1\rangle\langle 1|) \quad (2.19)$$

we recognize the Pauli operator σ_z in the basis $|0\rangle, |1\rangle$ defined in Chapter 1:

$$H_s = \frac{\hbar\omega_0}{2} \sigma_z \quad (2.20)$$

The eigenstates of this Hamiltonian are simply $|0\rangle$ and $|1\rangle$. The two eigenenergies are two midpoints separated by $\hbar\omega_0$.

2.2.2 Quantum harmonic oscillator

The quantum version of the harmonic oscillator is well known in quantum physics and can be applied for ions confined in the quadratic potential of an ion trap :

We consider an ion of mass m defined by the position and moment operators x and p inside an harmonic potential of quadratic coefficient k . The motional state Hamiltonian H_m is therefore:

$$H_m = \frac{\hbar}{2} \left(\frac{p^2}{m} + kx^2 \right) \quad (2.21)$$

The energy solutions are determined by the Schrödinger equation:

$$E_n = \hbar\omega \left(\frac{1}{2} + n \right) \quad (2.22)$$

with n an integer, and $\omega = \sqrt{k/m}$. We identify ω as being the axial secular frequency ω_z when the harmonic potential is in the axial axis. The position operator is also renamed z . The different n represents the different energy levels of the ion motional state $|n\rangle$ which can be operated on by means of the destruction and creation operators a and $a^\dagger$, also known as the ladder operators and defined as:

$$a = \sqrt{\frac{m\hbar\omega_z}{2}} \left(z + \frac{ip}{m\hbar\omega_z} \right) \quad (2.23)$$

$$a|n\rangle = \sqrt{n}|n-1\rangle \quad (2.24)$$

$$a^\dagger = \sqrt{\frac{m\hbar\omega_z}{2}} \left(z - \frac{ip}{m\hbar\omega_z} \right) \quad (2.25)$$

$$a^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle \quad (2.26)$$

$$a^\dagger a|n\rangle = n|n\rangle \quad (2.27)$$

2.2.3 The interaction Hamiltonian

When using trapped ions for quantum information experiments, either hyperfine or Zeeman states are chosen as the basis of the qubit. the qubit is manipulated by interacting the trapped ion with an external coherent source. Let us consider a system where a coherent electromagnetic source $Be^{i(kz+\phi-\omega t)}$ of phase ϕ , frequency ω and magnetic field amplitude B is applied on an ion characterized by a motional state $|n\rangle$ in a harmonic potential of frequency ω_z and a 2 level spin state $|s\rangle = [|0\rangle, |1\rangle]$ of transition frequency ω_0 and can be represented by the

state ψ :

$$|\psi\rangle = |s, n\rangle \quad (2.28)$$

The ion dipole moment being $\vec{\mu} = g\mu_B\vec{S}$, with g the g -factor, μ_B the Bohr magneton and S the electron-spin angular momentum. For a two level system we have $S = \sigma_x/2$. The interaction with the magnetic field source is based on the magnetic dipole interaction $H_i = -\vec{\mu} \cdot \vec{B}$. We rewrite the position operator z of the ion by defining z_0 as the ion's spatial extent of the ion wavefunction along the axial axis of the ion trap:

$$z = z_0(a^\dagger + a) \quad (2.29)$$

A few quantities are important to define before displaying the simplified Hamiltonian. the coupling between the internal and motional states of the ion, also called spin-motion coupling, is defined by the Lamb-Dicke parameter:

$$\eta = kz_0 = \frac{\omega \cdot z_0}{c} \quad (2.30)$$

We further define $\sigma_\pm = \frac{\sigma_x \pm i\sigma_y}{2}$ with $\sigma_x, \sigma_y, \sigma_z$ the Pauli matrices. In the Lamb-Dicke approximation defined as :

$$n\eta^2 < 1 \quad (2.31)$$

and by suppressing the fast oscillating terms with the Rotating Wave Approximation, the interaction Hamiltonian H_i can be developed as [3]:

$$H_i(t) = \frac{\hbar\Omega}{2}(\sigma_+e^{i(\phi-\omega t)} + \sigma_-e^{-i(\phi-\omega t)}) + i\frac{\eta\hbar\Omega}{2}(\sigma_+a + \sigma_-a^\dagger + \sigma_+a^\dagger + \sigma_-a)(e^{i(\phi-\omega t)} - e^{-i(\phi-\omega t)}) \quad (2.32)$$

With the strength of the dipole interaction defined by the Rabi frequency Ω :

$$\Omega = \frac{g\mu_B B}{2} \quad (2.33)$$

2.2.4 The Jaynes-Cummings Hamiltonian

The Hamiltonian of this entire system, commonly called the Jaynes-Cummings Hamiltonian is therefore:

$$H_{tot} = H_s + H_m + H_i \quad (2.34)$$

This Hamiltonian contains a well known part $H_s + H_m$ whose eigenstates are known and a more complex term H_i . The Hamiltonian can therefore be transformed in the interaction picture with:

$$H_I = e^{i(H_s+H_m)t} H_i e^{-i(H_s+H_m)t} \quad (2.35)$$

This Hamiltonian, once expended to first order then simplified further by using again the Rotating Wave Approximation in the Lamb-Dicke approximation defined in equation 2.31, leads to three cases, displayed in Figure 2.4 and forming a toolbox at the basis of any quantum state manipulation with trapped ions. All three are oscillations between the $|0\rangle$ and the $|1\rangle$ state in a phenomenon called Rabi oscillation. The first one happens when the field is resonant with the transition, a.k.a $\omega = \omega_0$ and only involve the states $|0, n\rangle$ and $|1, n\rangle$. The rate of oscillation is given by the Rabi frequency Ω .

$$H_c = \frac{\hbar\Omega}{2}(\sigma_+ e^{i\phi} + \sigma_- e^{-i\phi}) = \hbar\Omega(\sigma_x \cos(\phi) - \sigma_y \sin(\phi)) \quad (2.36)$$

The two others results involve spin-motion coupling as the Rabi oscillations also involve in those cases the motional modes, with a slower rate given by $\eta\Omega$. The first one, occuring for $\omega = \omega_0 - \omega_z$, is called the red sideband and coherently oscillates between $|0, n\rangle$ and $|1, n-1\rangle$:

$$H_r = i\frac{\eta\hbar\Omega}{2}(\sigma_+ a e^{i\phi} - \sigma_- a^\dagger e^{-i\phi}) \quad (2.37)$$

The last occurs at $\omega = \omega_0 + \omega_z$, is called the blue sideband. Similar to the red sideband losing a phonon during the oscillation, it involves the gain of a phonon when reaching the upper spin state:

$$H_b = i\frac{\eta\hbar\Omega}{2}(\sigma_+ a^\dagger e^{i\phi} - \sigma_- a e^{-i\phi}) \quad (2.38)$$

2.2.5 Single qubit gates

The final goal in studying the Jaynes-Cumming Hamiltonian for quantum computing is to extract a set of evolution operator $U(t) = e^{-iH_I t}$ which can be used as quantum logic gates such as the ones previously described in Chapter 1 and allowing the construction of a ion trapped based quantum computer.

For single qubit gates, the carrier interaction H'_c is used. For the first gate studied, the phase ϕ is set to zero

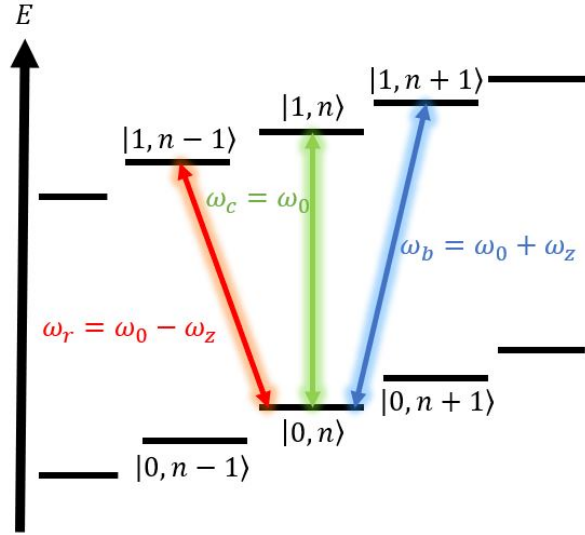


Figure 2.4: Energy levels from the spin and motional states of an ion and the transitions associated with the carrier (green), the red sideband (red) and the blue sideband (blue). Those three transitions are applicable for most n except $n=0$ which has no red sideband transition allowed.

which gives the following evolution function:

$$U_{c,\phi=0}(t) = e^{-i\hbar\Omega\sigma_x/2} \quad (2.39)$$

by replacing t with $\theta = t/\hbar\Omega$ we define a rotation gate around the X axis:

$$R_X(\theta) = U_{c,\phi=0}(\theta) = e^{-i\theta\sigma_x/2} \quad (2.40)$$

The matrix representation for R_X is:

$$R_X(\theta) = \begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (2.41)$$

A second single qubit gate orthogonal to R_X can be found for $\phi = \pi/2$, which leads to a similar rotation gate but over the Y axis:

$$R_Y(\theta) = U_{c,\phi=\pi/2}(\theta) = e^{-i\theta\sigma_y/2} \quad (2.42)$$

The rotation gate can similarly be written on a matrix form:

$$R_Y(\theta) = \begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (2.43)$$

These two rotation gates are enough to reach any points on the Bloch sphere. The physical implementation is quite straightforward: the angle θ is physically adjusted by the duration t of the pulse compared to the Rabi frequency, while the choice over the axis of rotation is controlled by the phase of the pulse sent. A particular angle of interest is $\theta = \pi$, commonly called π pulse where the gates are equal to the Pauli matrices up to a global phase:

$$R_X(\pi) = -i.\sigma_x \quad (2.44)$$

$$R_Y(\pi) = i.\sigma_y \quad (2.45)$$

2.2.6 The geometric gate

The approach toward an entangling gate with trapped ions involves multiple ions trapped in the same ion chain in a harmonic potential and therefore sharing the same motional mode. It is therefore possible, by encoding a qubit in the internal levels, to interact multiple qubits by using the common motional mode of ion as a “quantum bus” and achieve entanglement. The spin-motion coupling is available via the red and blue sidebands but a usable two qubit gate needs both maximal entanglement of the spin states and no residual spin-motion coupling at the end of the pulse so that the qubits stay isolated after the gate. The asymmetry between the red and blue sideband on the motional ground state was used to design the first entangling gate called the Cirac-Zoller gate [49]. The scheme is a controlled phase gate which has been implemented [50] but remains experimentally challenging as it requires a pure motional ground state, which is never fully reached and worsens rapidly with time due to heating.

The most important scheme used nowadays for entangling trapped ions is the geometric phase gate, which by driving a spin dependent interaction, achieves an evolution operator:

$$U(\theta) = e^{-iS_\phi^2\theta} \quad (2.46)$$

where S_ϕ is the spin operator defined as:

$$S_\phi = i \left(\sigma_+ e^{i\phi} - \sigma_- e^{-i\phi} \right) \quad (2.47)$$

where ϕ is a phase related to the pulses driving the spin operator. This phase can be used to construct different geometric gate but it will now be simplified to $\phi=0$ and referred to the ion k such as $S_{\phi=0,k} = \sigma_{y,k}$ for clarity.

The evolution operator of an ideal geometric gate is the following:

$$U(\theta) = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos(\theta) & 0 & 0 & -i \sin(\theta) \\ 0 & \cos(\theta) & -i \sin(\theta) & 0 \\ 0 & -i \sin(\theta) & \cos(\theta) & 0 \\ -i \sin(\theta) & 0 & 0 & \cos(\theta) \end{bmatrix} \quad (2.48)$$

By driving this interaction for a time such as the $\theta = \pi/4$, Bell states can be produced out of the bare states:

$$|0\rangle|0\rangle \rightarrow \frac{1}{\sqrt{2}} (|0\rangle|0\rangle - i|1\rangle|1\rangle) \quad (2.49)$$

$$|1\rangle|1\rangle \rightarrow \frac{1}{\sqrt{2}} (-i|0\rangle|0\rangle + |1\rangle|1\rangle) \quad (2.50)$$

2.2.7 The Mølmer-Sørensen gate

One interaction able to drive such geometric phase gates is called the Mølmer-Sørensen [51, 52] gate, a gate similar to an earlier theoretical scheme [53] proposing to interact with spin states independently from the initial motional state. This gate works by applying pulses which can only be driving coherently ions when combined and not individually. By applying equally intense fields of Rabi frequency Ω near the blue and red sidebands, $\omega_{field1} = \omega_r - \delta$ and $\omega_{field2} = \omega_b + \delta$, the two fields can be used for a coherent two photon interaction:

$$\omega_{field1} + \omega_{field2} = \omega_c - \omega_z - \delta + \omega_c + \omega_z + \delta = 2\omega_c \quad (2.51)$$

The Mølmer-Sørensen Hamiltonian is the sum of the red and blue sideband Hamiltonians with an additional off resonant term $e^{i\delta t}$. When simultaneously applied to two ions of index $k \in [0, 1]$, sharing the same motional mode, with the fields red and blue detuned on the center of mass mode, the resulting Hamiltonian is:

$$H_{MS} = \sum_{k=1}^{N=2} \frac{i\hbar\eta\Omega}{2} \sigma_{y,k} \left(a^\dagger e^{i\delta t} - a e^{-i\delta t} \right) \quad (2.52)$$

The evolution operator of this time varying Hamiltonian is extracted via the Magnus expansion and can be written as:

$$U_{MS}(t) = e^{\sigma_{y,[0,1]}(\alpha(t)a^\dagger - \alpha^*(t)a)} e^{-4\chi(t)\sigma_{y,[0,1]}^2} \quad (2.53)$$

with $\sigma_{y,[0,1]} = \sigma_{y,0} + \sigma_{y,1}$ the collective spin operator and:

$$\alpha(t) = \frac{i\eta\Omega}{2\delta} (e^{i\delta t} - 1) \quad (2.54)$$

$$\chi(t) = \frac{\eta^2\Omega^2}{4\delta^2} (\delta t - \sin(\delta t)) \quad (2.55)$$

The ideal geometric gate defined in equation 2.46 is obtained for $\alpha(t) = 0$ and therefore with the smallest non zero gate time duration is $t_g = 2\pi/\delta$. The gate is characterized by its effective Rabi frequency $\tilde{\Omega}$:

$$\tilde{\Omega} = \frac{\eta^2\Omega^2}{2\delta} \quad (2.56)$$

The phase gate θ is equal to $4\chi(t_g)$ and is controlled by the detuning δ . Full entanglement when $\theta = \pi/4$ is obtained for $\delta = \pm 2\eta\Omega$. Different geomtric phase θ can be applied by adjusting the detuning and the gate time:

$$\delta(\theta) = \pm\eta\Omega\sqrt{\frac{\pi}{\theta}} \quad (2.57)$$

$$t_g(\theta) = \frac{2\sqrt{\pi\theta}}{\eta\Omega} \quad (2.58)$$

We can therefore define the Mølmer-Sørensen gate as a rotation gate R_{yy} : $U(\theta) = R_{yy}(\theta)$. The R_{xx} can be similarly implemented by adjusting the relative phases of the fields driving the gate. An important remark is that contrary to the Cirac-Zoller gate which requires the motional ground state, the Mølmer-Sørensen can be applied with any motional state, although the sensitivity to errors does increase with the initial motional state eigenvalue.

2.3 Ytterbium as a qubit

As the general framework to build quantum computer with trapped ions was presented, this section will detail the specific solution used in the research group with ytterbium ions and the technological tools implemented to enable the quantum gates previously described. A recent improvement on entangling gate will be explained, then the application for large scale quantum computing will be described with its key challenges as well as the solutions conceived to overcome them.

2.3.1 The ytterbium atoms

Ions used for trapped ion quantum computers need to be easy to trap and to possess long coherent states suitable for being used as qubits. Ytterbium is a rare earth metallic element with a photo-ionization energy of 6.25 eV which is equivalent to a radiation of 198 nm. However it is possible to ionize Ytterbium ions via a 2 photon absorption process as represented in Figure 2.5: first, displayed by the blue arrow, a radiation coherently excites an isotope dependent transition from $^1S_0$ to $^1P_1$, then a non resonant photon with any wavelength below 394 nm, represented by the purple arrow, ionize the atom. Most further operations regarding cooling, preparation and detection will involve a 369 nm laser. For practicality the 369 nm source will be used for the non-resonant ionization process.

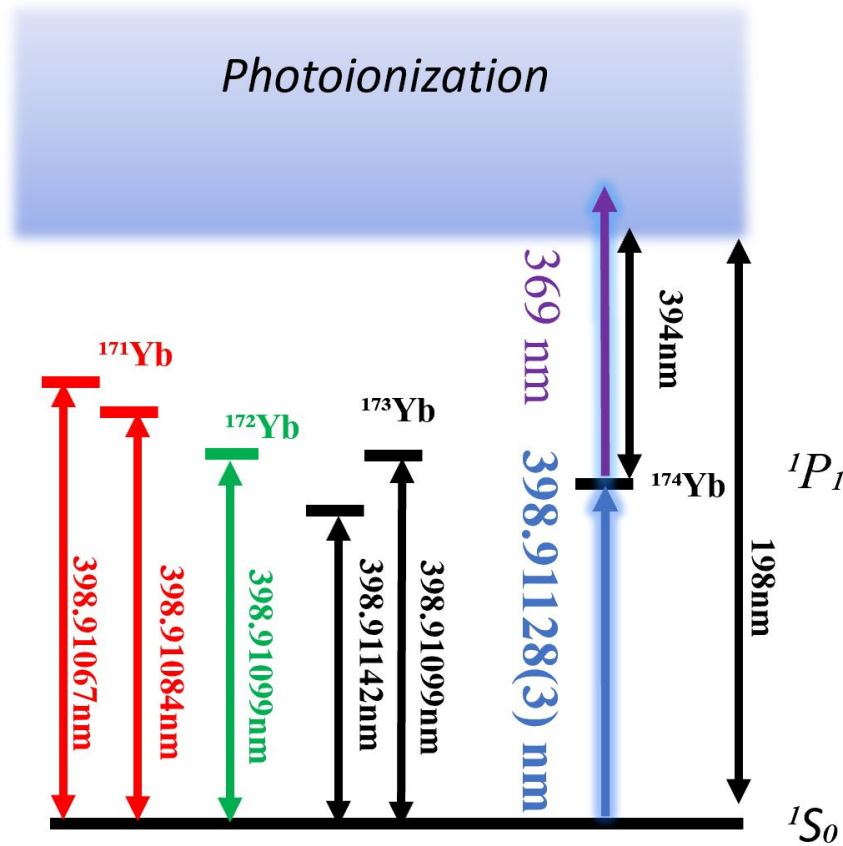


Figure 2.5: Energy level diagram for neutral Ytterbium, showing the two photons ionization process. The first transition from $^1S_0$ to $^1P_1$ is isotope dependent, therefore the first laser wavelength can be tuned to selectively excite only the isotope required for the experiment. Numerical values for the transitions are taken from [54]. Multiple transitions are available for ^{171}Yb and ^{173}Yb because they have non integer spins. The diagram is not at scale.

2.3.2 The ytterbium ions

The largest lived coherent states in atomic physics can be identified in hyperfine states, resulting from the coupling between the nuclear spin and the electron spin. Hyperfine states only appear for isotopes with a non zero nuclear spin, therefore an odd isotope number. The first isotope of interest, $^{174}\text{Yb}^+$, has no nuclear spin and by extension cannot be used for quantum information processing. However, this isotope is the most abundant in nature (around 34%), and has transitions very close to the isotope chosen for quantum logic in the research group, $^{171}\text{Yb}^+$. It is usual to start trapping experiment with the 174 isotope to reduce any experimental errors such as micromotion and identify key parameter such as the trap secular frequencies or the optimal power settings for the atomic oven.

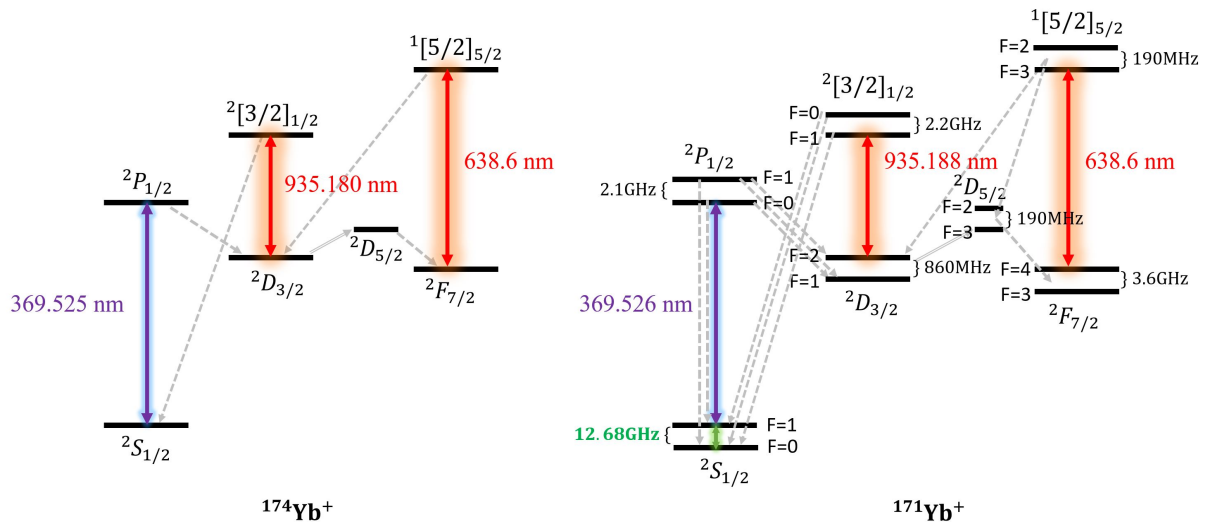


Figure 2.6: Simplified electronic energy level diagram for $^{174}\text{Yb}^+$ (Left) and $^{171}\text{Yb}^+$ (Right). $^{171}\text{Yb}^+$ has additional hyperfine state which are not represented at scale and are shown with no external magnetic field applied. The wavelengths are very similar and the same lasers can be used to trap the two different isotopes. The dashed lines represent the allowed transition from spontaneous decay. Glowing arrows represent the addressed transitions for the pumping cycle: The purple arrow represents the 369 nm laser, the red arrows the 935 nm and the 638 nm repump lasers while the green arrow represents the microwave addressing the $|^2S_{1/2}, F = 1\rangle$ hyperfine states.

As shown in Figure 2.6, both isotopes need two lasers aligned at the ion trapping position for cooling. The first one, the 369 nm laser, does the Doppler cooling by pumping the state $^2S_{1/2}$ to $^2P_{1/2}$ which will then emit 369 nm light by fluorescence. With a 0.5% probability the $^2P_{1/2}$ will instead decay to the $^2D_{3/2}$. By pumping it to the $^3[3/2]_{1/2}$ state with a 935 nm laser beam, it will decay back to the initial $^2S_{1/2}$ and close the cooling cycle. On rare occasions due to inelastic scattering due to collision, the $^2D_{3/2}$ state can transition to the $^2F_{7/2}$ state via the $^2D_{5/2}$ state. This “dark state” cannot emit 369 fluorescence, and unless repumping, will look on the imaging setup as if the ion left the trap. A third laser at 638 nm but can pump the $^2F_{7/2}$ state to the $^2D_{3/2}$ state

via the $^1D_{5/2}$ state but it can be neglected for most experiments: the phenomena was observed on another ion trap to occur in average less than daily. In order to treat dark ions and ions leaving the trap on the same level, every new attempt to trap ions is preceded by a reset of the RF applied to the chip.

For $^{171}\text{Yb}^+$, the hyperfine states generate more transitions to cover in order to close the repumping cycle. $^2S_{1/2}$ is covered by continuously emitting a microwave field at the transition between $F=0$ and $F=1$ at $f = 12.64$ GHz. The states $^2D_{3/2}$ and $^3[3/2]_{1/2}$ are covered by increasing the 935 nm laser beam power such as the transitions get power broadened enough to be addressed with it, therefore closing the pumping cycle.

2.3.3 Dressed state configuration

The state $^2S_{1/2}, F = 1$ can be further split in the presence of a magnetic field into 3 states with each having their own magnetic moment due to the Normal Zeeman effect: $m_F = -1$, $m_F = 0$ and $m_F = 1$. For weak magnetic fields, B , $m_F = 0$ is insensitive to first order, while the -1 and $+1$ Zeeman states can be approximated as linearly dependent on B . From now on the states from $^2S_{1/2}$ will be defined as:

$$\begin{aligned} |0\rangle &= |^2S_{1/2}, F = 0\rangle \\ |-1\rangle &= |^2S_{1/2}, F = 1, m_F = -1\rangle \\ |0'\rangle &= |^2S_{1/2}, F = 1, m_F = 0\rangle \\ |+1\rangle &= |^2S_{1/2}, F = 1, m_F = +1\rangle \end{aligned}$$

It is possible to encode a qubit with $|0\rangle$ and any of the Zeeman triplet states. we define the Zeeman shift δ_B defined as:

$$\delta_B = \mu_B B / \hbar \quad (2.59)$$

with μ_B being the Bohr magneton. By defining the $|0\rangle \leftrightarrow |0'\rangle$ transition as $\omega_0 = 12.64$ GHz, we can write the other transitions as:

$$\omega_{+1} = \omega_{|0\rangle \leftrightarrow |+1\rangle} = \omega_0 + \delta_B \quad (2.60)$$

$$\omega_{-1} = \omega_{|0\rangle \leftrightarrow |-1\rangle} = \omega_0 - \delta_B \quad (2.61)$$

The second order dependency on the $|0'\rangle$ state is defined as:

$$\omega_{0'} = \omega_{|0\rangle \leftrightarrow |0'\rangle} = \sqrt{\omega_0^2 + \frac{g_J^2 \mu_B^2 B^2}{\hbar^2}} \quad (2.62)$$

If we assume that a MW field generator is setup to control a qubit but its frequency cannot be adjusted, then it is possible to encode the qubit on $|0\rangle \leftrightarrow |+1\rangle$ by modifying the magnetic field B at the ion position to tune the transition in resonance. Rabi oscillations with the $|+1\rangle$ state have been demonstrated [55], however the decoherence time happens to be much smaller compared to non-magnetic field sensitive qubits: the $|0\rangle \leftrightarrow |0'\rangle$ basic qubit encoding reported a T2 time of 2.5 seconds [56] whereas a qubit encoded in $|0\rangle \leftrightarrow |+1\rangle$ in the research group had a measured T2 time of 2.2 ms [55]. This instability originates from the environmental magnetic field noise: any magnetic field noise is directly equivalent to a noise on the $|+1\rangle$ and $|-1\rangle$ energy levels. The overall effect on Rabi oscillations when driven by a fixed frequency is a reduced coherence time. T2 time in general is mainly constrained by experimental factors: a recent experiment encoding the $|0\rangle \leftrightarrow |0'\rangle$ qubit reducing external noise, applying more complex pulses and continuously cooling with other species had reported a T2 time of 4235 seconds [57], which is 3 orders of magnitude above the first reported measurement. To prevent the noise from affecting the qubit, a scheme called state dressing can be employed which will keep the magnetic field tunability while protecting the qubit from magnetic field noise. By applying a field ω_{+1} at the $|0\rangle \rightarrow |+1\rangle$ transition and ω_{-1} at $|0\rangle \rightarrow |-1\rangle$ with the same intensity Ω and phase $\phi = 0$, we obtain a sum of the Hamiltonian representing a coherently driven transition in equation 2.36:

$$H_D = \frac{\hbar\Omega}{2}(\sigma_+^{+1} + \sigma_-^{+1} + \sigma_+^{-1} + \sigma_-^{-1}) \quad (2.63)$$

where $\sigma_{\pm}^{\pm 1}$ are the operators $\sigma_{\pm}$ defined earlier but associated to transitions between $|0\rangle$ and $|\pm 1\rangle$. The dressed states are the eigenstates of this Hamiltonian [55]:

$$\begin{aligned} |D\rangle &= \frac{1}{\sqrt{2}}(|+1\rangle - |-1\rangle) \\ |u\rangle &= \frac{1}{2}(|+1\rangle + |-1\rangle) + \frac{1}{\sqrt{2}}|0\rangle \\ |d\rangle &= \frac{1}{2}(|+1\rangle + |-1\rangle) - \frac{1}{\sqrt{2}}|0\rangle \end{aligned}$$

The dressed state representation is illustrated in Figure 2.7: The hyperfine state energy levels are represented on the left while the dressed state is shown on the right. The qubit defined by $|0'\rangle \leftrightarrow |D\rangle$ is not affected by magnetic field noise to first order because its transition $\omega_{|0'\rangle \leftrightarrow |D\rangle}$ is magnetic field insensitive to first order as well. Interestingly, driving the qubit is still magnetic field sensitive because it is addressed via the $\omega_{|0'\rangle \leftrightarrow |+1\rangle}$ transition. Experimental coherence time measurement for $|D\rangle$ was measured at 650 ms, which is two orders of magnitude above the non protected states $|\pm 1\rangle$. The rest of this thesis will therefore assume the qubit as being

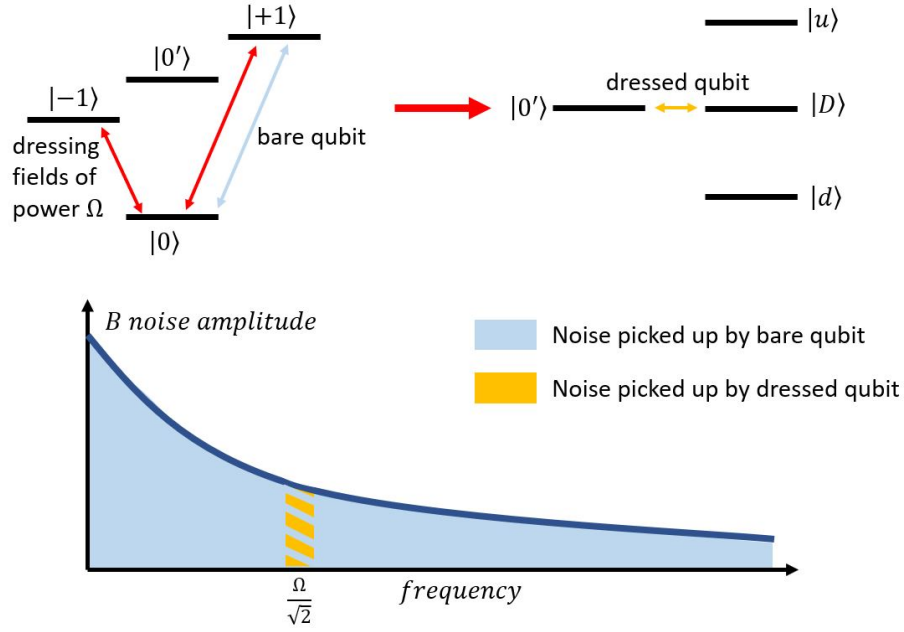


Figure 2.7: Dressed state qubits diagram compared to the bare hyperfine states of $^{171}\text{Yb}^+$. The noise spectrum, assumed to be $1/f$, is displayed with the bandwidths which influence the bare qubit and the dressed qubit representation.

defined on the $|0'\rangle$, $|D\rangle$ states.

Once the state dressing is in effect, an RF field can be applied on the $|0'\rangle$ to $|+1\rangle$ or $|-1\rangle$ transition to achieve the single qubit gate operation. It is important to remark that due to the second order component of the Zeeman splitting from equation 2.62, these two transitions do not have exactly the same frequency, guaranteeing only one “access” to the dressed state.

The $|u\rangle$ and $|d\rangle$ dressed states are however still magnetic field sensitive and their energy levels are separated from $|D\rangle$ by $\frac{\Omega}{\sqrt{2}}$ [55]. Therefore the magnetic field noise still affects the qubit coherence time, but as illustrated in the bottom diagram of Figure 2.7, the magnetic field noise bandwidth affecting the qubit is restrained around the transition frequency $\frac{\Omega}{\sqrt{2}}$. As higher frequency components of the magnetic field noise have lower amplitudes [58], a simple procedure to improve the dressed qubit coherence time would be to increase the microwave field power dressing the states.

In order to use this result with the single qubit gate scheme described in Section 2.2.5, the dressed state $|D\rangle$ needs to be coherently mapped to the bare state $|0\rangle$ when no dressing fields nor magnetic fields are applied to conserve the qubit during shuttling. A standard method is the partial stimulated Raman adiabatic transition (STIRAP)[59] which shapes the amplitude of the dressing fields to coherently transfer the state between $|0\rangle$ and $|D\rangle$. However Randall et al. [60] elaborated a more generic control toolbox to adapt two level controls to any multi-level systems, including robust gates detailed in Chapter 3, and as an application designed a robust

and coherent state mapping between $|0\rangle$ and $|D\rangle$ adapted from the two-level BB1 composite pulse [61].

2.3.4 Ion Doppler cooling

An ion is trapped in the trap center when a neutral atom is brought into the trapping potential and then photo-ionized. Neutral atoms are usually sent via a device called atomic ovens. Being thermalized by sublimation on a solid source they retain kinetic energy when being trapped. Furthermore, heating sources constantly affect trapped ions such as electric and magnetic noise as well as anomalous heating on surface traps. In order to cool down initial ions, then maintain them cold in the trap, RF and DC fields are not sufficient and a cooling technology called Doppler cooling has to be employed [62], combining the Doppler effect and conservation of momentum.

By the Doppler effect, ions in motion have an absorption spectrum shifted from an idle case. Defining a single photon source, the absorption spectrum is broadened covering both cases where the ion moves toward the source and away from it. Assuming a light source bandwidth smaller than the ion bandwidth, it is possible to adjust the photon wavelength λ so that ions absorb photons only when moving toward the source, a process called “red detuning”.

By conservation of momentum, the ion momentum is modified by h/λ away from the light source for each photon absorbed. Spontaneous emission of the absorbed photon however can occur in any direction which has less impact on the ion kinetic energy. Therefore, by red detuning the light source the average added momentum over time will always be negative or null, meaning that the ion kinetic energy will decrease until a minimum $E_{Doppler}$ determined by the natural transition bandwidth Γ of the ion:

$$E_{Doppler} = \hbar\Gamma/2 \quad (2.64)$$

2.3.5 Preparation and detection

In $^{171}\text{Yb}^+$ ions, the qubit is usually defined on $|^2S_{1/2}, F = 0\rangle$ and one or more of the Zeeman states from $|^2S_{1/2}, F = 1\rangle$. Before being able to discuss how one can manipulate a qubit defined on the hyperfine states of trapped ion systems, two Di Vincenzo criteria need to be addressed with the ytterbium ion: the ability to initialize a quantum state and the ability to measure the quantum state at the end of the experiment.

The two mechanisms are shown Figure 2.8: For quantum state preparation, the 369 nm laser beam, usually used for Doppler cooling through the $|^2S_{1/2}, F = 1\rangle \rightarrow |^2P_{1/2}, F = 0\rangle$ transition is detuned to reach the $|^2P_{1/2}, F = 1\rangle$ as shown by the purple arrow on the left diagram, while the microwave fields resonantly

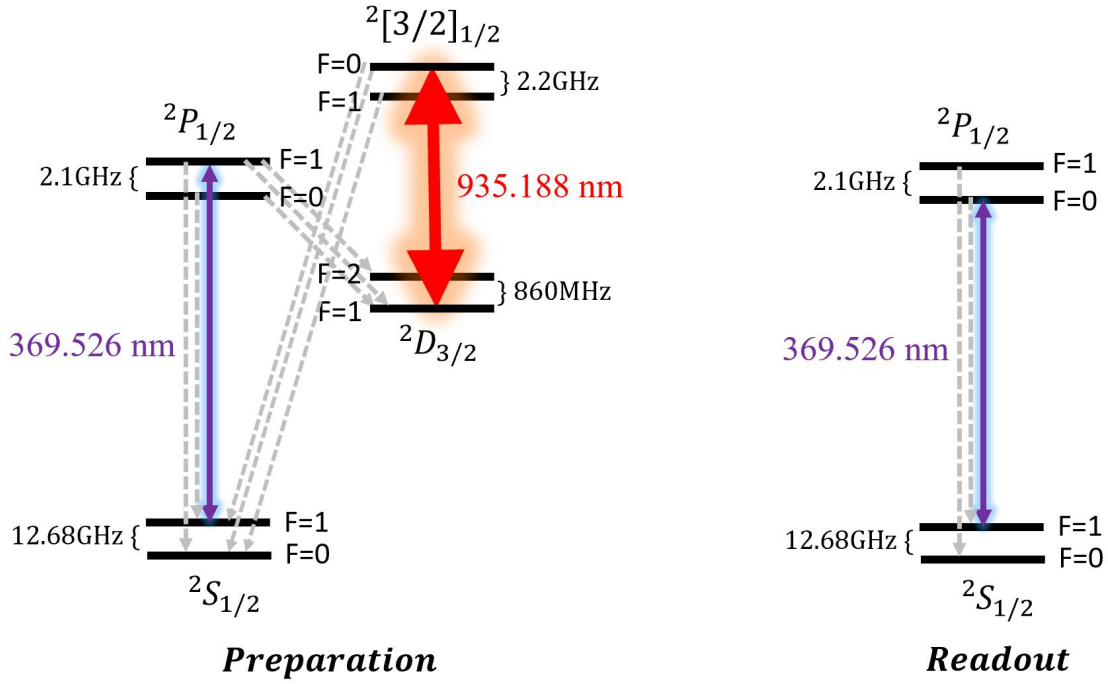


Figure 2.8: Electronic energy level diagram for $^{171}\text{Yb}^+$ for quantum state preparation (Left) and readout (Right). Both steps require the microwave fields off. The large red arrow represents the high power 935 nm addressing simultaneously the two possible transitions between the $^2D_{3/2}$ and $^3[3/2]_{1/2}$ states by power broadening.

addressing the $|^2S_{1/2}, F=0\rangle \rightarrow |^2S_{1/2}, F=1\rangle$ transitions are turned off. The $|^2P_{1/2}, F=1\rangle$ state can, with a decay rate of 19.6 MHz, either decay spontaneously to the $|^2S_{1/2}, F=0\rangle$ state which is then repumped, or to the $|^2S_{1/2}, F=1\rangle$ state, as represented by the dotted grey arrow, which cannot be re-pumped anymore by microwave fields. Therefore any initial qubit configuration in $^2S_{1/2}$ will inevitably end up in $|^2S_{1/2}, F=0\rangle$.

While running a quantum information experiment with ytterbium, the 369 nm laser beam is switched off to not interact with the qubits. For measuring the qubit and projecting it in either $F=0$ or $F=1$, the 369 nm laser beam is switched on again as displayed on the right of Figure 2.8; because it will only bring $F=1$ population to the P state, only the $F=1$ state will start emitting fluorescence. Because the targeted P state can only decay back to the $F=1$ state, the ion can continuously emit fluorescence, while a qubit projected on the $F=0$ will stay dark.

The photon distribution for a given measurement time follows a Poissonian statistics, where the mean photon number is limited by both the readout time and the quality of the experimental setup. Several experimental procedures exist to determine the projected state during measurement. The methods range in complexity from the threshold method which determines the number of photons received above which the logical state $|1\rangle$ is more likely to be measured, to neural network based decision integrating both space and time resolution of the photon emission [63].

2.3.6 The effective Lamb-Dicke parameter

The rate at which the Mølmer-Sørensen gate is driven is the Lamb-Dicke parameter defined in equation 2.30 which is inversely proportional to the wavelength of the field driving the gate. Common gate implementation uses 355 nm Raman mode locked lasers [64] to drive the MW and RF based transitions in ytterbium to provide a Lamb-Dicke parameter above 0.1. Directly driving a Mølmer-Sørensen gate with microwave fields results in a parameter using microwave wavelengths instead of optical ones and therefore is reduced by four orders of magnitude, preventing any direct use of global fields for generating entanglement.

There is however the possibility to use magnetic field gradients with magnetic field sensitive quantum states to enhance the spin-motion coupling. First proposed by Mintert et al. [27], it was implemented for the first time by Ospelkaus et al. [65] and then improved near fault tolerance in the research group by Weidt et al. [66, 55]. This new framework provides the resonant as well as the red and blue sideband Hamiltonian with a new effective Lamb-Dicke parameter η_{eff} . by considering the magnetic field influence on the Zeeman state at first order being $\omega_{\pm 1} = \omega_0 \pm \mu_B B$, and by assuming a magnetic field gradient along the axial secular frequency axis z $\frac{\delta B}{\delta z}$, then we obtain [27]:

$$\eta_{eff} = \frac{\mu_B}{\omega_z} \sqrt{\frac{1}{2m\omega_z}} \frac{\delta B}{\delta z} \quad (2.65)$$

This new Lamb-Dicke parameter has the remarkable property of not depending on the electromagnetic wavelength used to drive the interaction. Therefore, by engineering a field gradient strong enough, it becomes possible to drive a spin-motion interaction with global fields and therefore drive a Mølmer-Sørensen gate. This effective interaction can be associated with the dressed state scheme to drive an entangling gate between $|0'\rangle$ and $|D\rangle$, enabling entangling gates only based on a magnetic field gradient, microwave and radio-frequency fields.

An important difference with laser based spin-motion interaction is that each ion does not have the same transition frequencies anymore as they sit at a different position in the gradient. As illustrated in Figure 2.9, two ions of an ion chain placed inside a gradient share the same motional frequency ω_z but the $|+1\rangle$ and $|-1\rangle$ are different, constraining to duplicate both the microwave fields represented in green to drive the dressing fields for each ion. The RF fields driving the Mølmer-Sørensen interaction, represented by blue and red arrows, need also to be duplicated for each ion. In total eight fields are therefore needed to implement a Mølmer-Sørensen gate with global fields on dressed qubits.

It was introduced earlier in 2.3.3 than the $|+1\rangle$ and $|-1\rangle$ are linearly dependent on the magnetic field, but there is a small asymmetric quadratic dependence allowing to distinguish the $|0'\rangle \leftrightarrow |+1\rangle$ transition from $|0\rangle \leftrightarrow |-1\rangle$. While this dependency is not necessary to cover most of the above gate scheme consideration, it

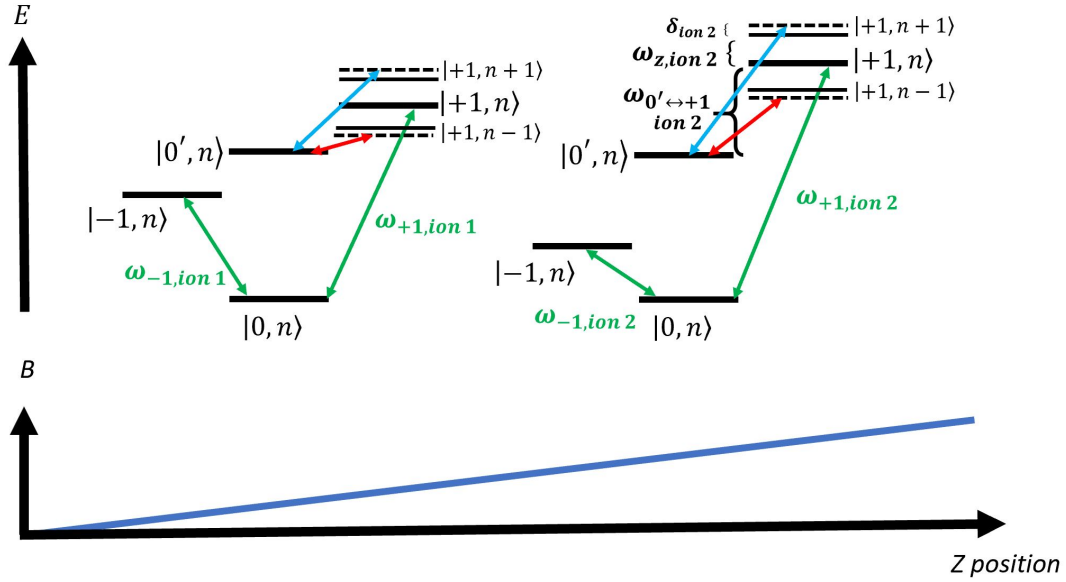


Figure 2.9: Pulses needed to drive a Mølmer-Sørensen entangling gate with dressed state qubits in a magnetic field gradients. The two ions have different state transitions for $|-1\rangle$ and $|+1\rangle$ which need custom MW (Green) for state dressing and RF for driving the spin-motion interaction near the red and blue sidebands (Red & Blue).

does however induce a small AC-Stark energy shift δ_{AC-S} when driving the Mølmer-Sørensen gate. This shift can be calculated with the quadratic magnetic field shift $\Delta\omega_{\pm}$ and the radio frequency power applied to the gate Ω_{RF} [55]:

$$\delta_{AC-S} = \frac{3\Omega_{RF}\Delta\omega_{\pm}}{4\omega_z} \quad (2.66)$$

and needs to be applied near the red and blue detuning such as:

$$\delta_r = -\delta + \delta_{AC-S} \quad (2.67)$$

$$\delta_b = \delta + \delta_{AC-S} \quad (2.68)$$

2.3.7 Sideband cooling

The Mølmer-Sørensen gate was previously mentioned to be insensitive to the initial motional state however its sensitivity to errors such as parameter mis-set or heating increases with the initial $\bar{n}$. Doppler cooling described in 2.3.4 is not sufficient for implementing high fidelity gates and an additional cooling mechanism is required, employing sideband Hamiltonian interaction to bring the motional state of the qubit to as close as possible to zero before starting the quantum circuit: sideband cooling.

Sideband cooling [67, 68] is a succession of cooling steps, applied on the $|0, n_i\rangle$ logical ground state. Each

step i is composed of two pulses: a π pulse on the red sideband bringing the qubit to $|1, n_i - 1\rangle$ state and a repump pulse, implemented similarly to the state preparation sequence mentioned in 2.3.5, conserving the motional state while bringing the internal state to $|0, n_i - 1\rangle$. The π pulse duration for a given Rabi frequency is dependent on the initial motional state related to the Rabi frequency Ω of the carrier and the Lamb-Dicke η parameter:

$$t_{\pi,n} = \frac{\pi}{\sqrt{n}\eta\Omega} \quad (2.69)$$

The complete sideband cooling sequence apply these steps in series starting with an arbitrary n_{max} chosen as a compromise between cooling power and time duration of the sequence itself. Values of $\bar{n}$ below 1 are reachable with this technology.

In the case of MW/RF implementation of sideband cooling via magnetic field gradient, cooling the dressed state instead of the bare state is necessary to maintain the qubit state coherence during the entire cooling process. Because the repump pulse can only happen on the $|0\rangle$ state of the Yb ion, cooling the dressed state adds as additional steps the transition between the $|0'\rangle$ and $|0\rangle$ states before the repump pulse and the transition back after it [69].

Interestingly measuring the effect of sideband cooling can be implemented by sideband spectroscopy: the mean phonon value $\bar{n}$ can be calculated by measuring the ratio of the red and blue sideband amplitude [55].

2.4 The QCCD architecture

The ion position control enabled by surface chip can be coupled to quantum logic operations to form a type of quantum processing unit called QCCD [26]. A short summary of ion position manipulation will be introduced, then the concept of QCCD devices with magnetic field gradients will be presented.

2.4.1 Ion manipulations

Manipulating ions above a surface chip consists of varying the voltages applied to the electrodes confining the ion to move the DC harmonic potential and therefore the ion. Any shuttling protocols need to leave the qubit state unchanged, which involves 2 conditions:

- The ion internal state defining the qubit as defined in 2.3.5 has to be maintained
- The ion motional state needs to remain near the motional ground state $|0\rangle$

The internal qubit state can be affected if it is sensitive to external fields such as a magnetic field. Shuttling a magnetic field sensitive qubit across an unknown constant magnetic field adds a phase which needs to be compensated for [70]. A motional state different from the ground state does not directly affect the gate fidelity described in 2.2.7 but increase the sensitivity to any parameter misset, causing the gate fidelity to deteriorate faster with time. All designed shuttling solutions need to maintain a motional state as small as possible to implement high fidelity quantum gates. Heating is, however, not entirely due to shuttling: increase in the ion motional mode is also a static phenomena from the electrode noise which, being not entirely explained, is commonly called “anomalous heating” and follows an empirical law of increasing with the ion-electrode distance by a $1/d^4$ factor [71].

Most shuttling protocols designed until today can be organized into two categories. The first one is called adiabatic shuttling and defines shuttling sequences where the ion movement caused by the modification of the DC voltages, represented by the DAC update rate f_V , is slower than the ion periodic motion ω_z in the axial direction:

$$f_V \ll \omega_z \quad (2.70)$$

The ion can be approximated as static along every point of the trajectory and a shuttling sequence is designed by calculating a DC static confinement potential in every position the ion will take along the path. Heating in the adiabatic regime occurs when the axial secular frequency, directly determined by the DC harmonic potential in the axial direction, changes across the shuttling sequence and adds perturbation to the ion motion. Shuttling in this regime is slow and by definition bounded by the axial secular frequency. As most ion traps have axial secular frequency ranging between a few hundred kilohertz to a few megahertz, shuttling protocol durations are between a millisecond and a few microseconds. The second category has many appellations from diabatic shuttling, non-adiabatic shuttling, shortcut to adiabaticity or superadiabatic regime, and consists of faster shuttling protocols coherently controlling the ion motional state. Some protocols such as the “Bang Bang” protocol have shuttling time in the hundreds of nanoseconds [72] but are much harder to implement because they are more susceptible to ion loss and demand much faster and less noisier hardware.

2.4.2 Position dependent quantum logic

By combining the gate scheme using magnetic field gradients to tune the ion ability to be driven by a gate and the previous description of surface ion traps capability in Section 2.4.1, we can introduce the concept of zones. Zones are localized section of the microfabricated trap dedicated to a set of functions of a quantum computer. A memory zone would be for example located away to all RF and MW fields as well as laser beams. A gate zone

is where a magnetic field gradient is present over localized sections of an ion trap. Ions are shuttled between zones when the current zone is not capable of accomplishing the requested function on the chosen ion. When it is required for the experiment to apply a gate operation; we can consider as an example an ion located in a memory zone; the ion is first shuttled to the nearest available gate zone, then the global fields are broadcasted to drive the gate interaction. The QCCD architecture is fully connected: all gates applied have similar speed and fidelities at the cost of additional dephasing and heating of the qubit during shuttling.

At first glance this organization looks like a microwave alternative to laser based ion processors, where lasers are carefully aligned to one specific trapping zone. However the real advantage lies in the fact that the radio-frequency and microwave radiation fields can target any qubits in a system that have their transitions tuned correctly via the magnetic field. In other words, regardless of the number of gate zones built with their magnetic field gradient, the set of 8 fields needed to drive one gate will actually drive all the Mølmer-Sørensen gates in the system, assuming the fields have constant power and phase across the entire chip. Early simulations from the research group shows spatial inhomogeneities from microwave and radiofrequency fields when sent into a vacuum system [73, Section 4.6.3], which has been experimentally observed in a QCCD prototype where different positions on the chip could experience a different microwave power of up to 3% [74]. Such differences is acceptable for dressed states, as it has a negligible impact on the AC-Stark shift mentioned in Section 2.3.6, and can be mitigated for coherent state transfer using robust pulses [60]. Additional fields corresponding to different positions in the gradients can also be emitted to correspond to additional quantum gates. Considering two single qubit gates in addition of the entangling gates are enough to reach universal quantum computation, a total of fourteen microwave and radiofrequency fields would be enough to achieve any quantum circuits on any number of qubits, as displayed in Figure 2.10.

While the laser complexity employed to drive gates is substantially reduced, they are still required for trapping, cooling, repumping and detection. There is already a wide variety of trap features implemented for trapping and detection [75], as larger surface chips will require more integrated features for scalability.

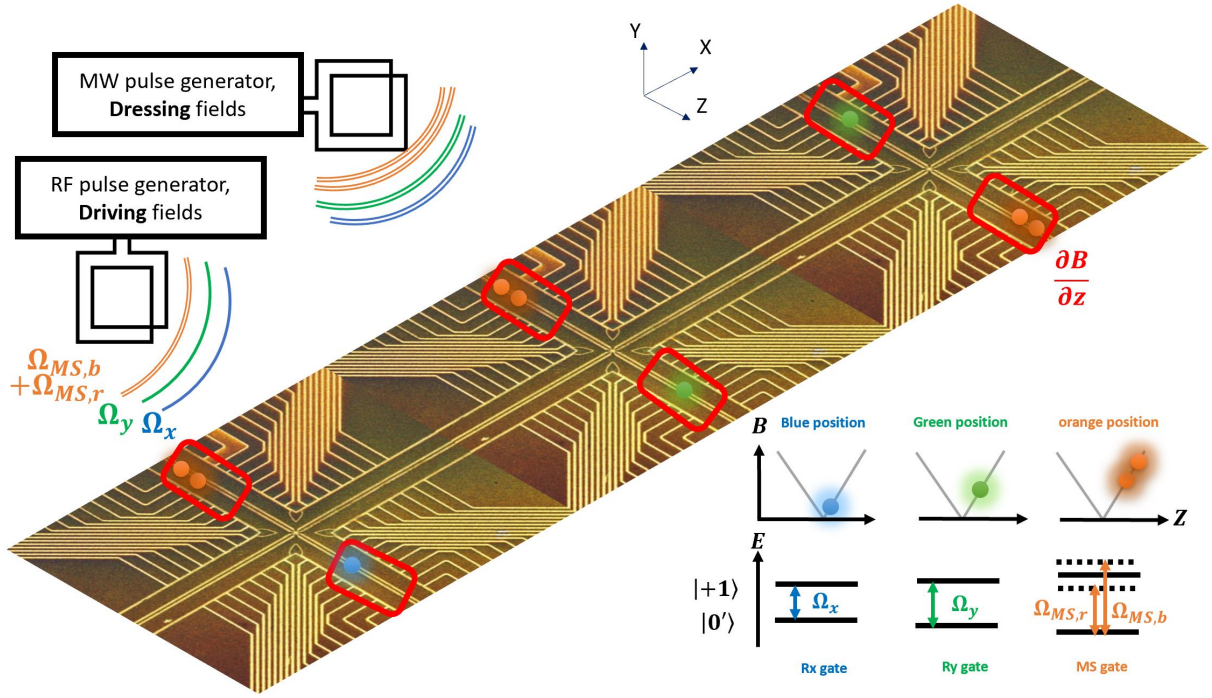


Figure 2.10: Illustration of the advantages of long-wavelength radiations with surface chips containing three X-junctions and six gate zones (delimited in red). The dressing microwave and driving radio-frequency fields emitted correspond to two different single qubit gates (blue and green) and the entangling gate (orange). Based on the position of the ions in the gate zones, its transitions will have different frequencies due to the magnetic field gradient. The selected gate is applied by bringing the ion at the physical position where the transition is equal to the frequency of the radio-frequency fields assigned to this gate. Figure inspired from [66]

Chapter 3

Robust quantum computing

The main obstacle towards building a quantum computers is the quality of the entangling gates. As described previously in Chapter 2, entangling gates with trapped ions are commonly slower and implemented with a fidelity lower than the single qubit gates. This limitation is due to the higher sensitivity to experimental noise and parameter errors. Engineering entangling gates more robust to these constraints can not only increase the gate fidelity but more importantly reduce the calibration procedures required to keep them operational.

First of all, this chapter will focus on formulating the robustness of an entangling gate, discuss the limitations of the Mølmer-Sørensen gate, and provide examples of inherently robust entangling schemes. Then the Polychromatic Mølmer-Sørensen gate will be introduced and its physical implementation described. Finally a reflection into quantifying the entangling gate robustness on the broader scope of calibration is considered, and a simulation is run to illustrate the benefits from using robust gates.

3.1 Robust quantum gates

Before introducing a new entangling gate scheme, a set of tools needs to be presented to analyze gate robustness. These tools will be presented alongside the Mølmer-Sørensen gate, and a few examples of more robust schemes will be presented to illustrate the difference in robustness.

3.1.1 The phase space representation

The general evolution operator of the Mølmer-Sørensen gate, described in equation 2.53, can be reformulated [2] as:

$$U_{MS} = e^{i\chi(t)} (|\leftarrow_{\phi}\rangle\langle\leftarrow_{\phi}|D(\alpha(t)) + |\rightarrow_{\phi}\rangle\langle\rightarrow_{\phi}|D(-\alpha(t))) \quad (3.1)$$

where $|\leftarrow_{\phi}\rangle\langle\leftarrow_{\phi}|$ and $|\rightarrow_{\phi}\rangle\langle\rightarrow_{\phi}|$ are the eigenstates of the collective spin operator $S_{\phi,[0,1]}$ and D is the displacement operator:

$$D(\alpha) = e^{\alpha a^{\dagger} - \alpha^* a} \quad (3.2)$$

The term $\alpha(t)$ can thus be represented as the motional evolution of the qubit in phase space (position x and momentum p). This evolution is conditional to the initial qubit states and a projection on the bare states will produce the matrix defined in equation 2.48.

The Mølmer-Sørensen gate's phase space trajectory can be described as starting at $(0,0)$, evolving along a circle of radius r and center c which is dependent on the initial qubit state, and coming back to $(0,0)$ at the end of the gate time t_g with [2]:

$$r = \frac{\eta\Omega}{2\delta} \quad (3.3)$$

$$c_{|\leftarrow_{\phi}\rangle\langle\leftarrow_{\phi}|} = \frac{i\eta\Omega}{2\delta} e^{-i\phi} \quad (3.4)$$

$$c_{|\rightarrow_{\phi}\rangle\langle\rightarrow_{\phi}|} = -\frac{i\eta\Omega}{2\delta} e^{-i\phi} \quad (3.5)$$

The space covered by the trajectory is the phase θ associated with the entangling gate. This geometric approach allows to reformulate the requirements of a geometric gate to implement entanglement: the state has to start and finish at the center $(0,0)$ to keep the motional state decoupled from the spin state before and after the gate, and the trajectory needs to cover a surface equal to θ .

3.1.2 Gate robustness characterization

This geometric representation being available to all geometric gates, it will now be used as the basis to discuss the sensitivity to different types of errors.

As previously discussed in Section 2.2.7, the trajectory $\alpha(t)$ needs to reach 0 at $t=t_g$, as any non zero value will result in residual spin motion entanglement. A timing error Δt can be protected for example by slowly approaching the phase space origin at the gate time such that $\alpha(t_g + \Delta t) \simeq \alpha(t_g) = 0$.

When driving a Mølmer-Sørensen gate two main sources of errors can arise from the RF applied field frequen-

cies: a symmetric detuning error $\Delta\omega_{sym}$ due to a drift of the secular frequency ω_z such that the red and blue fields are detuned equally by $\Delta\omega_{sym}$, and asymmetric detuning errors, although less common, which are an unequal detuning error in the red and blue fields applied. The impact of these errors in phase space are similar to the previous timing errors in the sense that the trajectory does not fully come back to the phase space origin, and therefore also results in residual spin-motion entanglement.

A generic toolbox for minimizing sensitivity to errors using the phase space trajectory was established over the key requirement [76]: the average or center of mass of the trajectory $\langle\alpha(t)\rangle$ should be minimized to 0, and is derived from formulating errors generated by zero mean dephasing noise. Both symmetric and asymmetric detuning errors are attenuated the closer this term approaches 0.

This condition explains the high sensitivity of the Mølmer-Sørensen gate to noise, with the center of the trajectory being half way between the origin and the largest $|\alpha(t)|$ distance.

The regularly mentioned sensitivity of the geometric gate fidelity F to the initial mean phonon occupation $\bar{n}$ is also illustrated with the trajectory $\alpha(t)$ by quantifying the error E when the phase space trajectory does not come back to the origin at $t = \tau$ [76]:

$$E_{\alpha(\tau) \neq 0} \propto |\alpha(\tau)|^2 (\bar{n} + 1/2) \quad (3.6)$$

Sideband cooling, which occurs before the quantum circuit and minimizes the mean phonon of the qubit, is effectively improving the robustness of any driven geometric gates. Sideband cooling is counteracted by heating, which progressively increases the mean phonon number and therefore the gate sensitivity. The influence of heating while the gate is being driven is considered in the next section. As a final remark, assessing the robustness of a quantum gate by evaluating only a single metric linked to the phase space trajectory is efficient but however does not provide any information nor indication on how to physically drive the qubits.

3.1.3 Dynamical decoupling and gate robustness

Before considering the robustness to experimental drift and/or parameter miset, most new gates were designed to be resilient against coherent errors. While robustness was presented above relative to the phase space trajectory, the efficiency metric of these dynamical decoupling gates to suppress coherent errors was modeled as a filter function filtering a generic $1/f$ noise spectrum [77]. A more resilient gate will have a smaller filter bandwidth. Dynamical decoupling schemes can be labeled in two broad categories: pulse dynamical decoupling (PDD) and continuous dynamical decoupling.

Pulse dynamical decoupling is an extension of the Hahn spin echo [78], in which the qubit is left to freely

evolve during the time $\tau/2$ after which a π pulse is applied to refocus noise in the second half of the evolution. Series of π pulses can be implemented on a qubit gate to minimize dephasing [79], one such scheme being the BB1 composite pulse [61] mentioned in Chapter 2. A decomposition of the Mølmer-Sørensen gate with Walsh modulation enabled dynamical decoupling to be applied to entangling gates [80]. The phase space trajectory was decomposed into multiple smaller loops such that π pulses could be applied at every origin crossing. Alternatively, a fast PDD scheme involves applying π pulses even when the spin and motion are entangled and careful timing still results in the trajectory ending at the origin [81]. PDD gates require generally a longer gate time and the capability to implement short π pulses, or in other words high powered fields.

Continuous Dynamical Decoupling (CDD) which involves continuously driving a qubit's transition to decouple it from noise has already been introduced in Chapter 2 in the context of dressed states. Continuous drivings effectively introduce a new dressed state picture in which only noise with a frequency matching the new energy gap will decohere the system. The energy gap opened by the dressing fields is usually proportional to the driving power, therefore amplitude noise may also result in a dephasing mechanism. Such amplitude fluctuations can be made insensitive by dressing the dressed states themselves, resulting in a “double-dressed states” configuration [82].

3.2 The polychromatic Mølmer-Sørensen gate

The tools outlined in the previous section indicate that numerous schemes can accomplish entangling gates, and their characterization should use new metrics on top of the gate fidelity, such as duration and robustness. In this section the work on a novel entangling gate is presented, called the polychromatic gate, starting with the theoretical construction of the gate scheme to provide robustness, and then followed by its physical implementation and characterization.

3.2.1 Theory

An extension of the Mølmer-Sørensen gate was proposed by Haddadfarshi et al. [83], adding more harmonics to drive the gate to provide degrees of freedom with which gate robustness can be engineered. With ω_0 being the resonant transition, ω_z the secular frequency and δ the frequency detuning from the sideband as defined in equation 2.57, one can formulate as displayed in Figure 3.1 the polychromatic pulses of order T as :

$$f_{b/r}(t) = \sum_{j=1}^T c_j \cos((\omega_0 \pm \omega_z \pm j\delta)t) \quad (3.7)$$

with c_j being the amplitude of the j th harmonic, a negative value corresponding to a 180 degrees phase added to the tone. The original gate is obtained with $T = 1$ and $c_1 = 1$. For $T > 1$, a similar derivation of the evolution operator written in equation 2.53 leads to additional displacement and phase terms for each tone. A global displacement and phase functions can be derived as [2]:

$$\alpha(t) = \sum_{j=1}^T \alpha_j(t) \quad (3.8)$$

$$\chi(t) = \sum_{m,k=1}^T \chi_{m,k}(t) \quad (3.9)$$

With the displacement and phase contributions of each tone being defined as [2]:

$$\alpha_j(t) = \frac{i\eta c_j \Omega}{2j\delta} e^{-i\phi} \left(e^{ik\delta t} - 1 \right) \quad (3.10)$$

$$\chi_{m,k,m \neq k}(t) = \frac{\eta^2 c_m c_k \Omega^2}{4\delta^2} \left(\frac{1}{k(m-k)} \sin(\delta(m-k)t) - \frac{1}{mk} \sin(\delta mt) \right) \quad (3.11)$$

$$\chi_{m,m}(t) = \frac{\eta^2 c_m^2 \Omega^2}{4\delta^2 m^2} (m\delta t - \sin(\delta mt)) \quad (3.12)$$

Generating a fully entangled state at the gate time t_g , equivalent to $4\chi(t_g) = \pi/2$, provides a first constraint on the parameters c_j :

$$\sum_{j=1}^T \frac{|c_j|^2}{j^2} = 1 \quad (3.13)$$

Haddadfarshi et al. [83] derived an analytic expression to measure the impact of heating on the gate fidelity based on the Master equation. Their finding anticipates the condition described in Section 3.1.2 for general gate robustness: the most efficient attenuation of heating errors occurs for $\langle \alpha(t) \rangle = 0$, but also for minimizing $\langle |\alpha(t)|^2 \rangle$. It provides two additional constraints, and combined with the equation 3.13, the recipe to calculate the parameters is:

$$\sum_{j=1}^T \frac{c_j}{j} = 0 \quad (3.14)$$

$$\text{Minimize} \sum_{j=1}^T \frac{c_j^2}{j^2} \quad (3.15)$$

In the simplest case, for two tones, the amplitude coefficient are calculated to be $c_1 = -0.576$ and $c_2 = 1.152$ [83]. The experimental interest toward this new entangling gate rose from the work of Collingbourne [84] who discovered by simulation a high robustness of the gate to symmetric detuning errors or secular frequency drifts.

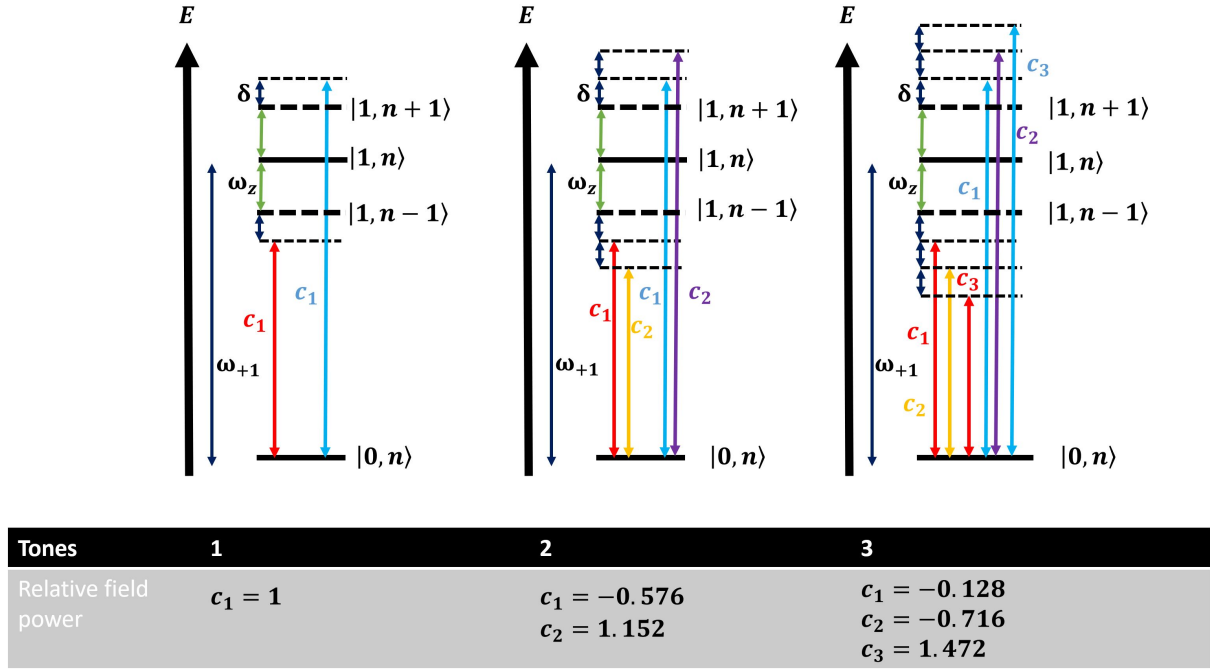


Figure 3.1: Energy level representing the pulses c_i required to drive a polychromatic Mølmer-Sørensen for one tone (Left), two tones (Center) and three tones (Right). The one tone gate is identical to the standard Mølmer-Sørensen gate. The bottom table indicates the power necessary for each pulse, in units of the effective Rabi frequency defined in equation 2.56. Negative numbers indicates that the pulses have an opposite phase.

3.2.2 Experimental apparatus

The theoretical construction of this gate proposes in theory a simple extension of the Mølmer-Sørensen gate. The predicted robustness of this new gate scheme needs to be confirmed experimentally. The experimental apparatus used to implement both the classic and polychromatic Mølmer-Sørensen gates in the research group is now described, then the steps required to experimentally implement the polychromatic gate are detailed.

The ion trap

The ion trap is a gold-plated blade trap including two blades for RF fields and two blades each divided in three sections for DC fields. The ion position is further adjusted by three additional compensation electrodes, located near the blades. The trap is located in a vacuum chamber with a ion gauge¹ to monitor the pressure and an ion pump² to maintain it. The pressure remains at 10^{-11} mbar. The RF field is controlled by an RF field generator³ and the signal is filtered with a homemade helical resonator. The DC fields are generated by a high voltage

¹Varian 9715015

²Varian StarCell 9191145

³HP8640

digital to analog converter⁴ (DAC) and sent to a 30Hz low pass filter adjacent to the vacuum system connectors linked to the trap electrodes [2].

As displayed in Figure 3.2, on each side of the blade trap are located two pairs of permanent samarium cobalt (SmCo) magnets, all coated with nickel and copper to prevent outgassing from the magnets themselves. The magnets are aligned to maximize the magnetic field gradient while minimizing the magnetic field offset at the ion position. The static magnetic field gradient was measured to be 23.6(3) T/m, but a residual magnetic field offset still exists. Furthermore, during the ion trapping procedure, the magnetic field offset was measured to drift likely due to temperature fluctuation associated with the atomic oven operation. Therefore the magnetic field offset is compensated to reach 10 G with three Helmholtz coils located outside the vacuum system [85] and is adjusted during each step of every experiments [55]. The effective Lamb-Dicke parameter when driving the stretch mode $\omega_z \sqrt{3}/2\pi = 461$ kHz is measured as $\eta = 0.004$ [2].

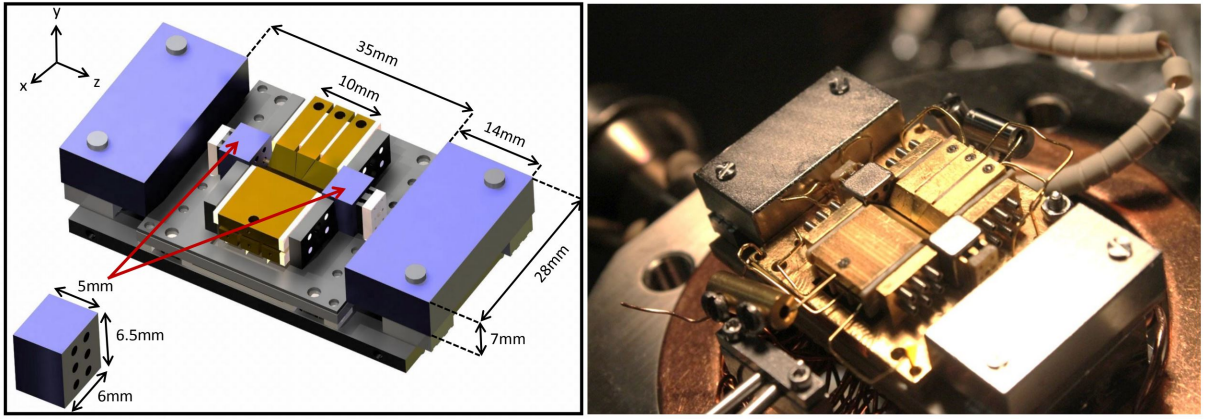


Figure 3.2: Blade trap experiment diagram (Left) and photo (Right) of the apparatus installed in the ultra high vacuum system. The magnets are represented in Purple and the electrodes in Orange. Figure taken from [55].

Optical setup

The 369 nm laser is sampled from a 1W Ti-sapphire laser⁵ doubled in frequency⁶ whereas the 399, 638 and 935 nm lasers are all homemade external cavity diode lasers (EDCL). The 369nm and 399nm are combined via a 369nm filter and the 638nm and 935nm beams are combined via a cold mirror [2]. The laser control interface is the one described in Section 6.1 and the optical setup for measuring the ion fluorescence is identical to the one later build for the logical qubit apparatus in Section 7.5.

⁴Stahl HV40-16

⁵Msquared Solstis

⁶Msquared ECD-X

Electronic setup

The MW and RF fields are emitted with antennas outside the vacuum system. Both signals are generated with a single two channels arbitrary waveform generator⁷ (AWG). Because the frequency range generated is in the MHz, the MW field is generated by combining the AWG waveform with a fixed MW signal generated by a vector signal generator⁸ (VSG). The mode of interest is filtered with a narrow band pass filter. The signal is then amplified by a MW amplifier and finally emitted by a MW horn antenna⁹. The RF signal is generated directly by the AWG, is amplified by a RF amplifier and sent by a helical antenna wrapped around the imaging tube.

The AWG waveform is generated by a dedicated computer connected to the computer running the experiment [86]. The experiment is run by an FPGA¹⁰ controlled by a LabVIEW program synchronizing the imaging setup, the AWG and the laser control via TTL pulses following the PulseSequence syntax described in Section 5.5.2.

3.2.3 Gate implementation

The gate was implemented with a carrier $\Omega/2\pi = 36$ kHz and a gate time $t_g = 3.4$ ms. Two important drawbacks from this scheme is the field power required to drive the gate. Although the tones are applied with a fixed amplitude c_j , their spectral proximity is close enough so that constructive and destructive interference affect the instantaneous field power during the gate time. This phenomena, as displayed in Figure 3.3, is unavoidable because the spectral distance is the same parameter δ characterizing the gate time t_g . This time dependent instantaneous power has two consequences: first of all, the instantaneous power was evaluated, in equation 2.66, to have a direct impact on the field detuning due to second order AC Stark shifts. The polychromatic gate therefore requires, a modulation of the gate field's frequencies in order to counteract the time dependent Stark shift.

Finally the RF supply chain containing the amplifier and antenna are required, at the middle of the gate time for two tones, to emit a higher peak power. Power ratings required to drive a polychromatic gate are therefore 70% higher. The maximum peak field power is an important “budget” or limitation when building and operating a quantum computer. In other words at equivalent power budget, the two tones gate time will be on average 70% slower than the original gate.

⁷Keysight M8190A

⁸Keysight E8267D

⁹Flann Microwave 18240-10

¹⁰National Instruments PXI-7842R

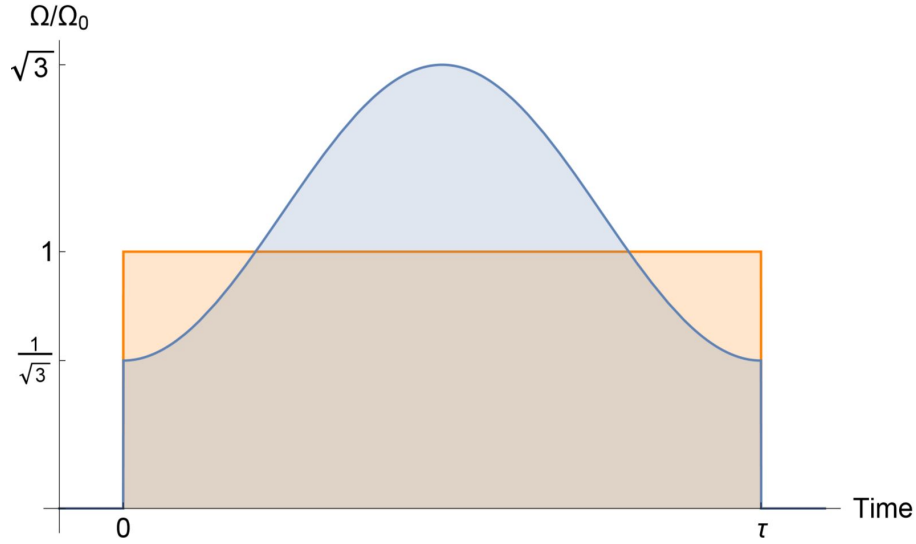


Figure 3.3: Instantaneous RF field power when driving a single tone Mølmer-Sørensen entangling gate (Orange) and a two tone gate (Blue). the instantaneous power modulation appearing from two tones is due to their spectral proximity. The peak power for two tones is $\sqrt{3}$ times the peak power for a standard entangling gate. Figure taken from [2].

3.2.4 Gate characterization

A systematic study of the impact on heating rate compared between the polychromatic gate and the original Mølmer-Sørensen gate involves a controllable heat source coupled to the ion crystal. Heating was injected via an external signal generator producing a bandpass noise signal centered at the ion's secular frequency and applied on one of the DC electrodes used to trap the ion crystal. The heat injected was calibrated by measuring sideband heating rates as mentioned in 2.3.7 for a set of different noise amplitudes.

As displayed in Figure 3.4, I contributed to Webb et al. [87] in studying the polychromatic gate's robustness against heating (Left) and secular frequency drifts (Right). A similar work was published simultaneously by Shapira et al. [88]. The power budget consideration mentioned above was tested by measuring the fidelity against the heating rate of a single tone gate with the equivalent peak power to the one used for the two tone gates. The result is plotted with the legend "1 tone, high power" and demonstrates the advantage of polychromatic gates over the standard gate for both heating and secular frequency drifts.

3.2.5 Outlook and further work

The polychromatic gate satisfies the robustness conditions described in 3.1.2 and was demonstrated in 3.2.4. The addition of amplitude varying tones at fixed phase allows enough degrees of freedom to minimize heating

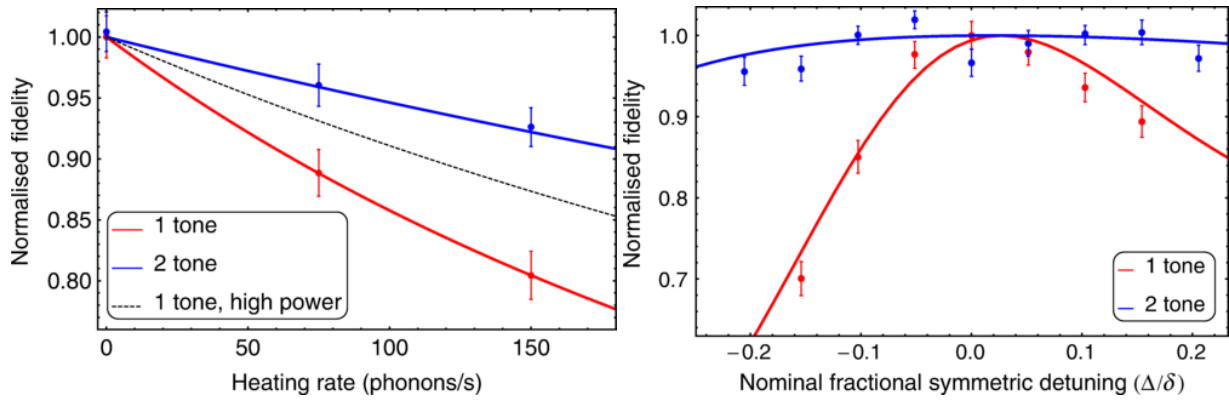


Figure 3.4: Robustness of a two tones polychromatic gate compared to the original Mølmer-Sørensen, by artificially inducing heating (Left) and shifts in the axial secular frequency commonly called symmetric detuning, formulated as the ratio between the drift Δ and the detuning δ defined in equation 2.57 (Right). The fidelities are displayed relative to their 0 induced noise equivalent, because of the additional experimental constraints on the fidelity. The 1 tone, high power on the left is a single tone gate with peak Rabi frequency equivalent to the one used to drive the two tone gate. The two tone gate is visibly more robust in gate fidelity to heating and symmetric detuning than the original gate, even at equivalent power budget. Figures taken from [87].

rate errors but other alternatives to the Mølmer Sørensen gate have since been devised. Because the initial polychromatic gate was constrained over a binary choice of phase and the amplitudes adjusted for compensating heating only, a more agnostic approach has been taken by Lishman et al. [89] by enabling every tone to have its own phase: the tone calculation is achieved by maximizing the gate fidelity including most empirical errors encountered with trapped ion qubits.

Alternative robust gates can be employed. By applying phase modulated fields in combination of spin echoes, simulations with a set of physical parameters reasonable improved compared to the system described in Section 3.2.2, predict a gate fidelity above 99.9% [90].

3.3 Calibrating a trapped ion based quantum computer

The reasons motivating the construction of robust gates is their higher fidelity and their easier implementation in hardware. However the obstacles preventing the construction of quantum computers are not only speed, fidelity and scalability: calibration is a commonly underestimated process which is time consuming and needs to be applied frequently. Robust gates, being resilient against experimental drifts, will ensure that a quantum computer remains operational for longer periods of time and reduce the required amount of calibrations.

This section will attempt to quantify the global impact of a polychromatic gate over a standard Mølmer-Sørensen gate when operating a quantum computer, instead of the impact over a single gate application. We will first describe in this section a framework to identify and correct calibration drift of an experimental quan-

tum computer. From there, a simulation of the device used in the previous section will enable to quantify how the gate choice influences the device's ability to run experiments.

3.3.1 Definitions

Before detailing calibration procedures, we need to introduce the following definitions:

1. Calibration: experiment designed to return a set of parameters
2. Parameter: physical quantity necessary for operating a quantum computer.

For a given parameter P , we only know its value after calibration but it does drift across time. As a quantum computer involves multiple sets of parameters with different sensitivities to drift, a model is therefore required to handle efficiently when and which calibration to run.

3.3.2 Introduction to the calibration problem

The most complete yet hardware agnostic approach to formalize and solve the choice of a calibration routine is called Optimus [91] and defines all required calibrations as nodes of a directed acyclic graph. A calibration C_{p1} dependent of another one C_{p2} is represented in Figure 3.5 on the left.

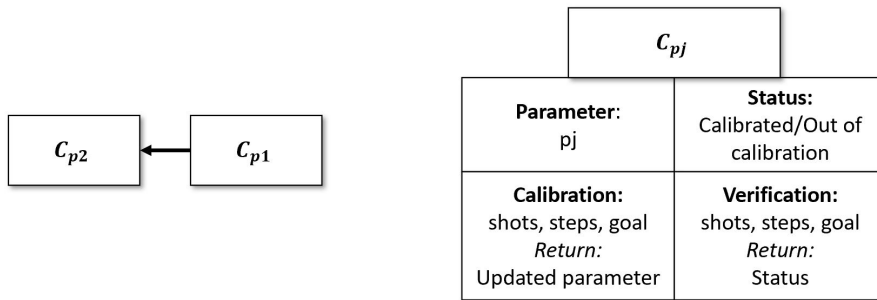


Figure 3.5: Node representation for calibration procedures.(Left) Calibration dependency is indicated by arrows, showing as an example a parent node C_{p2} over which a children node C_{p1} is dependent (Right). Each calibration node represents a calibration procedure containing information about its parameter, a boolean characterizing the calibration status and a set of physical information to run both the calibration procedure and a faster version to check if the previous calibration still is effective. Figure inspired from [91].

Each node (Right Figure) has a binary working state representing if it is calibrated or not. Interestingly each node is defined as being capable of running two types of experiments: the standard calibration which usually

consists in scanning a parameter until some condition is matched, as well as a much faster experiment only querying if the known parameter is still the correct one, which we define as a verification experiment.

This dual approach allows fast estimation of an entire system's working state. Because an uncalibrated parent node will have a negative effect on a child node, the main algorithm to verify the graph is to apply verification experiments from the root nodes to the end of the chain. When a verification fails at a node, then its calibration is applied and the verification procedure resumes. A special status is also covered when both verification and calibration for a node do not generate good enough data, indicating a wider problem. In this case, its parent node is considered next and not its child node.

The only timing consideration is addressed by adding to each calibration node, a timeout after which a verification is required.

3.3.3 Weak node dependency

This calibration strategy is both flexible and efficient with its two levels of calibrating experiments, but can be further improved by increasing the detail of the relationships between nodes: indeed while some nodes have dependencies for any variation of parameters, some other nodes only require a basic level of calibration from a parent and are not affected in case the later experiences drift. In order to illustrate this we will build on the example given in [91] but applied to trapped ions.

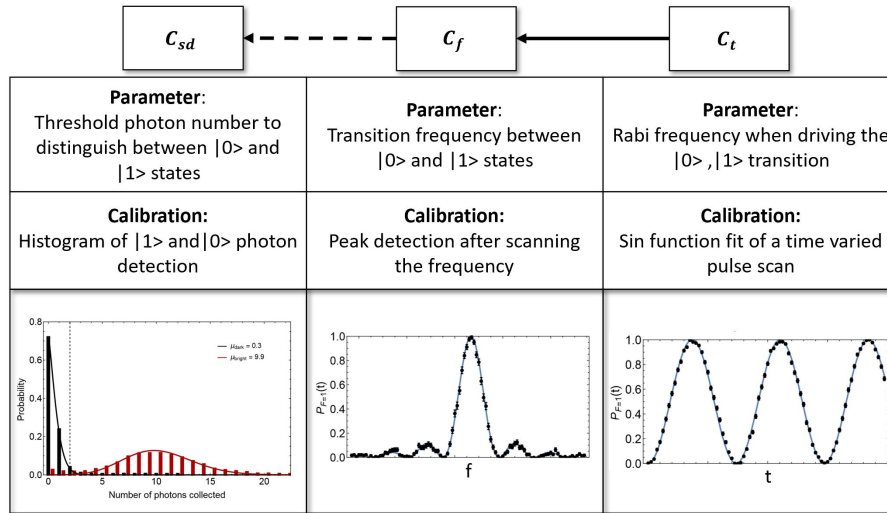


Figure 3.6: Example of three nodes of a calibration system for a single qubit gate. While the state detection calibration is necessary for subsequent experiments, drifts loosely affect further calibrations. However the time scan is directly dependent on the frequency scan and needs to be more closely monitored. Experimental scans are taken from [55].

In order to drive a transition between ground and excited states, an Hamiltonian similar to equation 2.39 is

required, implying both a knowledge of the Rabi frequency and the transition frequency. The classic approach is therefore to calibrate first the state detection mechanism C_{sd} mentioned in section 2.3.5, then to scan the transition frequency C_f and finally at this frequency, scan the pulse length C_t and measure the Rabi oscillations. The node representation defined above is displayed in Figure 3.6.

However the dependencies between those nodes are not the same: while C_t is utterly dependent on C_f and any drift in the latter will cause errors in the former, C_f is only loosely dependent on C_{sd} , as the frequency peak of the scan will not be modified by it. We define this loose dependency as a weak dependency: we display it as a dotted arrows and represent it in a node via a new status called “Loosely calibrated” and given by the verification procedure with another goal parameter. The timeout does not need to be changed. If an experiment needs to be run and only has as parent node a weak dependency, then the “Loosely calibrated” status is enough. In the case where there is a strong dependency, the calibration routine has to be called.

3.3.4 Time based calibration simulation

This model described above formulates time resources using timeouts but does not provide answers regarding the total time needed and which calibration may represent a bottleneck in the calibration chain. I designed a simulation tool to better understand the dynamic of calibration and provide an estimation of its impact on the time available for running a quantum computer.

We first define a fidelity function $F_P(t)$ representing the efficiency of the calibrated value p from the calibration node C_p after a time t . This simulation is intended to describe a trapped ion system and will be based on the following assumptions:

1. The number of qubits is small enough so that single and two qubit gate calibrations apply to all qubits
2. Each parameter to calibrate needs only one experiment to run
3. The ion loss rate is neglected
4. The fidelity function after calibration decreases linearly in time
5. When calibrated a fidelity function has a value of 100%
6. Only one calibration can be conducted at a time

We consider the functions to calibrate based loosely on the DiVincenzo criteria [5]. The fidelity function in a node C_p can be described with two properties: the time taken for calibration T_p and the fidelity decay rate

Γ_p . If we consider t_0 the time after which a calibration was just conducted, then we can write the fidelity function $F_P(t)$ as: $F_P(t) = 100 - (t - t_0)\Gamma_p$. We distinguish a node status with its fidelity function, defining $F_P(t) > 99\%$ as calibrated, $99\% > F_P(t) > 90\%$ as loosely calibrated and below as not calibrated. Knowing the fidelity function removes the constraint of chaining calibrations and applying verification functions from the directed acyclic graph model. We consider the quantum computer calibrated if all of its calibration nodes are calibrated. The time for which a quantum computer is considered calibrated is called T_A whereas the time used to run any calibration is defined as T_C .

When running a simulation of the calibration needed to maintain a quantum computer, we will consider two metrics to estimate the device robustness: the average time available before having to run calibration measurements, $\langle T_A \rangle$, and the percentage of time available to run circuits, which we define as A :

$$A = \frac{T_A}{T_A + T_C} \quad (3.16)$$

Increasing a quantum computer's productivity is now formulated as maximizing $\langle T_A \rangle$ and A . These two metrics have different applications: A is the immediate time ratio of how long a quantum computer can be used to run quantum circuits. Businesses employing Quantum as a Service (QaaS), already deployed with for example Azure Quantum and Amazon Braket¹¹, are billing each quantum circuits being run and therefore the amount of time being used. Increasing A directly increases the rentability of a quantum computer following this model.

$\langle T_A \rangle$ represents the ability to run quantum circuits with high depth but without including qubit associated errors such as decoherence. Even by assuming slow gates of 3.4 ms such as the polychromatic gate implemented in Section 3.2, a required 5 ms sideband cooling and a 1 ms quantum state readout, a $\langle T_A \rangle$ equal to a second would still be sufficient to run a quantum circuit with a depth above a hundred. While not representing much interest for NISQ devices, large scale quantum computers on the other hand have much different requirements, having to use Quantum Error Correction codes on millions of qubits on large time scales. Using the magnetic field gradient of the research group, an estimation of the time required to successfully run the Shor algorithm was calculated to be $T_A = 110$ days [1].

3.3.5 Remarks on the calibration model

We can identify two extreme regimes for those simulations. The first one is when $\frac{1}{\Gamma_p} \gg T_p$, which allows a simplified $A \approx 100\%$. The device will be mostly available and the different calibrations will not interfere with

¹¹<https://aws.amazon.com/braket/pricing/>

each other. On the opposite side of the spectrum we can study a much worse scenario for two nodes i and j defined as:

$$\frac{1}{\Gamma_i} < T_j \quad (3.17)$$

$$\frac{1}{\Gamma_j} < T_i \quad (3.18)$$

meaning that every time a node i is calibrated, the node j will be out of calibration and vice versa, so in consequence $A=0\%$. Most devices hopefully do not have such loss rate compared to the calibration time, but because they will have many processes experiencing decay, one can question the availability of a quantum computer be on a day to day basis.

This model is a poor approximation of real deterioration happening in most quantum computing experiments as the real fidelity threshold for calibration experiments may vary, the decay in fidelity is non-linear and can impact several calibration nodes simultaneously, so scenarios where a system is assessed calibrated but was not experimentally calibrated may occur and must be taken into account. Verification experiments also take time and further reduce the available time T_A . However if the most important time scales at which processes need recalibration is known, then a first order estimation on time scale available to run a quantum circuit can be deduced.

3.3.6 Calibration of an experimental entangling gate

The Mølmer-Sørensen gate in the magnetic field gradient scheme is composed of several pulses of different power, length and frequency [66] and some of those settings need to be measured up to several times a day. The recipe to prepare the experimental setup to implement a two qubit gates is greatly inspired from Randall [55] and follows the graph displayed in Figure 3.7:

the first set of parameters regards the operation and calibration of the experiment itself. Before trapping ions, lasers need to be adjusted and wavemeters need to be calibrated. These processes are gathered in the node C_{laser} . The acceptable drift is represented by the AOM range, 60MHz via the double pass setup described in Chapter 7, available to maintain the 369 nm wavelength on the ion. Recalibration involves adjusting laser locking and the AOM settings. Ion trapping and ion monitoring is represented by C_{ion} . Long term drifts in the fluorescence are due to mechanical instabilities, but the term C_{ion} also covers the rare events when the ion leaves the trap and new ions have to be trapped. Recalibration is based on the average time in the experiment to trap a new ion and optimize the fluorescence.

The next parameters correspond to calibrating the qubits. The calibrating time is for all the nodes involved

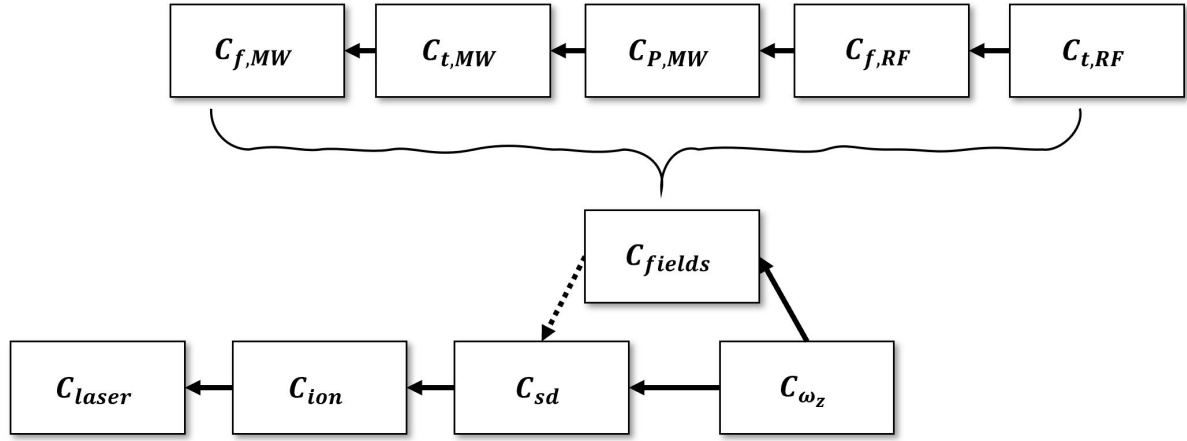


Figure 3.7: Directed acyclic graph representing the calibration nodes needed to calibrate a Mølmer-Sørensen gate.

directly linked to the number of experiments to be run and the number of shot per experiments. Assessing the quantum state detection is in the node C_{sd} . Drifts above a few percents change the photon count threshold to estimate the quantum state readout and will affect the measurements of all subsequent experiments. Adjusting the frequency and amplitude of the microwave and radio-frequency fields is gathered in one node C_{field} for simplicity but fine tuned simulation should consider the extended chain considering each transitions. Among these fields the RF frequency calibration is the most regularly required as errors directly translate in the entangling gate as asymmetric detuning errors. The secular frequency node C_{ω_z} is the final step before considering a gate calibrated, although extracting fidelity metrics from the gate itself would require two additional experiments. The difference between the Mølmer-Sørensen gate and the polychromatic gate are illustrated by their sensitivity to secular frequency drifts and are therefore associated to different nodes: $C_{\omega_z,MS}$ for the Mølmer-Sørensen gate, and $C_{\omega_z,PMS}$ for the polychromatic gate. The acceptable drifts of 10 Hz and 100 Hz correspond to the gate fidelity dropping by around 5 %. While this large range can be considered as unacceptable for most large devices, they correspond to a NISQ scale with limited qubits and quantum circuit depth. All those nodes and their numerical estimations are summarized in the table 3.1.

3.3.7 Simulation results

Based on the model summarized in Table 3.1, a python simulation is run over a full virtual week with a one minute step. As mentioned above the knowledge of the fidelity function allows for the removal of any dependencies between the nodes and to run them independently. We therefore evolve the fidelity functions at each time step. When a fidelity function goes under 99%, the system occupies a “calibrating” status during the cali-

Node parametrization				
Name	Typical values	acceptable drifts	$\frac{1}{\Gamma}$ drift timescale	Calibration time (s)
C_{laser}	369.52506nm	0.03pm	1 day	300
C_{ion}	2k counts/s	100 counts/s	1 day	600
C_{sd}	85% for two ions	1%	2 hours	300
C_{fields}	14.1MHz	20Hz	12 hours	1800
$C_{\omega_z, MS}$	272kHz	10Hz	20 mins	300
$C_{\omega_z, PMS}$	272kHz	100Hz	3 hours	300

Table 3.1: Summary of the calibration nodes considered for the simulation. Most of the quantities represented are empirical estimations. Two different versions of the secular frequency calibration node are given to represent the robustness of the polychromatic gate $C_{\omega_z, PMS}$ described in 3.2.4 compared to the Mølmer-Sørensen gate $C_{\omega_z, MS}$.

bration time associated to the node. When two fidelity functions both go under the threshold the worst fidelity function is chosen to be re-calibrated. Other fidelity functions still evolve with time during calibration. The outcome is a list of times during which the system was fully calibrated. The histograms are displayed in Figure 3.8.

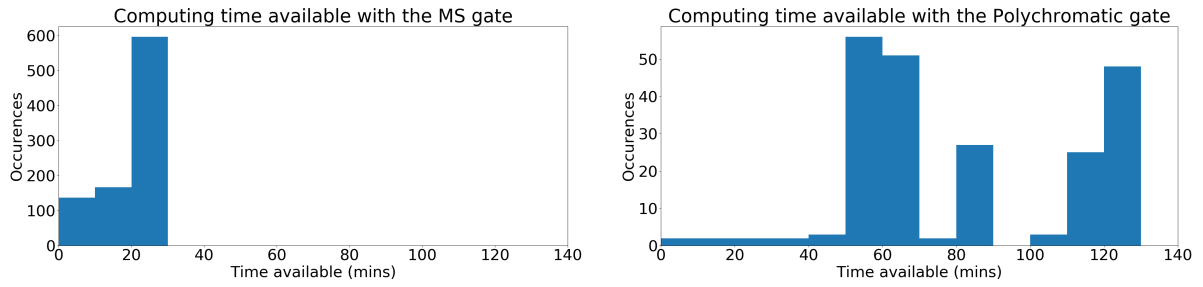


Figure 3.8: Histograms of the time available to run quantum circuits before requiring calibration, for the original Mølmer-Sørensen gate (Left) and the polychromatic gate (Right). The higher sensitivity to experimental drift for standard gates reduces the time available and increase the required number of calibration. While not affecting noticeably the gate fidelity as defined in Chapter 1, the polychromatic attenuates the effect from ω_z , the biggest parameter instability in the device.

For standard Mølmer-Sørensen gates (left), the average time available was simulated to be $\langle T_A \rangle = 16$ minutes, with the device being available for $A = 71\%$ of the time. The polychromatic gate (Right), introduced in Section 3.2.4, was not only efficient to reduce heating and therefore increasing the gate's maximum fidelity, but also towards reducing the error due to the shift in the secular frequency mentioned in the above model. The fluctuations of the secular frequency observed in this system are completely within range of the shielding capabilities of this new gate. This improves the simulation by outputting a mean available time of $\langle T_A \rangle = 80$ mins with a $A = 88\%$ availability. This new protocol alone, with a modest overhaul complexity increase in the entire system, was able to increase the capabilities of this two qubits quantum computer device by 17%.

3.3.8 Remarks and outlook

The simulation illustrates that robust gates have effectively more impact on a quantum computer than its improved fidelity. The polychromatic gate was shown to both improve $\langle T_A \rangle$ and A , demonstrating therefore an advantage for both NISQ and large scale devices. However, the simulated value of $\langle T_A \rangle$ with the polychromatic gate is still orders of magnitude lower than the 110 days value estimated with running the Shor algorithm in a large scale quantum computers. The difference is large enough to question if improving gate robustness will be sufficient for large scale quantum computing.

In addition to more robust gates such as the polychromatic gates with more tones, further physical improvements of a quantum computer can participate in increasing productivity. More stable and robust locking of the lasers is achievable by using more stable references and automating the locking mechanism. Ion lifetime can be increased by reducing the pressure via cryogenic cooling. Cooling will also reduce the anomalous heating rate and will increase as a consequence the gate fidelity. Transition stability can be improved by better stabilizing the magnetic field, for example by increasing the stability of the mechanism described by Randall [55]. Drifts in the antenna power can be reduced by actively locking the output power with feedback measurements from a probe. Drifts in the axial secular frequency can be reduced with more stable DACs and better low-pass filters. Alternately, allowing calibration procedures to run in only local sections of a device while the quantum circuit is running elsewhere, could, in some extent, bring A to a 100%. The trapped ions will regularly move between sections of the trap so that each section stays calibrated. A new calibration metric can then be defined as the largest quantum register that can be used in a quantum circuit, while maintaining calibrated quantum control at all time.

While this simulation displays the importance of error robustness and time budget necessary for calibrations, it does not yet reflect experimental realities where fidelity functions are hidden and cross-dependent, and where interplay between verification and calibration needs to be setup. More refined simulations will provide further insight into the effect of gate robustness and the impact of scalability. In conclusion, designing an operational quantum computer will require, in addition to optimizing the gate fidelities as well as the number of qubits, the identification of the calibration experiments and their timescale in order to maintain the device to its full capacity.

Chapter 4

Quantum error correction

Large scale quantum computing is a regime in which most quantum algorithms, like the Shor algorithm described in Chapter 1, can be implemented. The physical gate schemes described in Chapter 2 and their robust upgrades detailed in Chapter 3, are still limited by the physical qubits and their coherence time. For large enough quantum circuits, the physical qubits have to be replaced by more robust logical qubits which are built by Quantum Error Correction (QEC) codes. This chapter will review the fundamentals of Quantum Error Correction codes, and present the theoretical and experimental efforts to implement one of its most promising schemes, the surface code. The focus is set on answering the following problem: How do QEC codes work, and how can we implement them on a physical device?

4.1 Motivations for QEC

4.1.1 Physical limitations of quantum computers

The algorithms described in Chapter 1, which are to provide a quantum advantage over classical computers, have limitations due to the challenges of building a physical device: the number of qubits available, their coherence and the residual noise/error in running gates, state preparation and measurement. The Quantum Volume metric also covered in Chapter 1 encompasses these limitations. A quick overview of running the Shor algorithm for a 2048 bit key with a standard Quantum Fourier Transform would need at least 4096 qubit and therefore $\log_2 QV = 4096$, which is three orders of magnitude above the best systems available today with $\log_2 QV = 8$ [24]. This gap between current and expected performance is hardly expected to be crossed by using the physical qubits as logical qubits of the quantum circuit. The biggest limitation will not come from

the qubit number but rather the achievable circuit depth - the number of gates that can be performed in series before the output is indistinct from noise. Large quantum circuits like the Shor algorithm, require a circuit depth polynomial with the qubit size [92] and cannot therefore rely on the quantum volume metric, which only certifies a depth linear with the qubit size.

Error correction encompasses sets of techniques to protect classical bits of information from a noisy environment. As a simplistic illustration we will consider one classical bit of information being transmitted with a probability $p \ll 1$ of being affected by noise under the form of a bit-flip operation. This can be represented as a noise channel such as displayed in Figure 4.1.

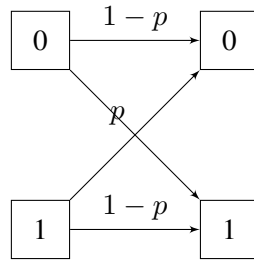


Figure 4.1: Bit-flip error channel with probability p .

We can encode a single logical bit over multiple physical qubits and determine p_L , the probability the logical qubit suffers an error. A basic strategy to encode the information is to duplicate it three times, into for example $[0_L] \rightarrow [0][0][0]$ and $[1_L] \rightarrow [1][1][1]$. Then if a single bit flip error happens it is still possible to deduce the correct bit by a majority vote strategy. The probability for the logical bit to actually experience an error is now $p_L = p^2$, assuming independent errors, as at least 2 are necessary to cause a logical error. Therefore as long as $p < 1/2$, the logical bit is more resistant to noise than the physical bit. More importantly, this logical encoding can be extended to as many bits as needed to reduce p_L : using an n (odd) bit repetition code we have $p_L = 2^{-(n-1)}$ and the code can correct up to $(n-1)/2$ errors.

Because of the non-cloning theorem [93], an arbitrary quantum state Ψ cannot be duplicated into another register. The duplicating code just described above is therefore not applicable to quantum states thus other schemes are needed. This chapter will describe the tools to similarly encode logical qubits over multiple physical qubits, to reduce error rates and improve coherence times.

4.1.2 Depolarizing noise model

While more hardware specific noise models will be described later, a very useful and generic model is the depolarizing channel. Any qubit with a state $|\Psi\rangle$ going through a depolarizing channel will have an additional

Pauli σ_i gate applied to it with probability p_i as displayed in Figure 4.2. σ_x is a bit flip error, σ_z is a phase flip error and σ_y is a combination of the first two. With a probability $p_0 = 1 - p_x - p_y - p_z$ the state will not be affected. If the noise channels are further simplified with equal probability then the depolarizing channel is considered homogeneous. This channel can be applied for each single qubit gate applied. For multi-qubit gates each qubit involved independently experiences the depolarizing channel, e.g. in the case of a 2 qubit gate 15 error combinations are possible: $\{\sigma_x\sigma_x, \sigma_x\sigma_y, \sigma_x\sigma_z, \sigma_xI, \sigma_y\sigma_x, \sigma_y\sigma_y, \sigma_y\sigma_z, \sigma_yI, \sigma_z\sigma_x, \sigma_z\sigma_y, \sigma_z\sigma_z, \sigma_zI, I\sigma_x, I\sigma_y, I\sigma_z\}$.

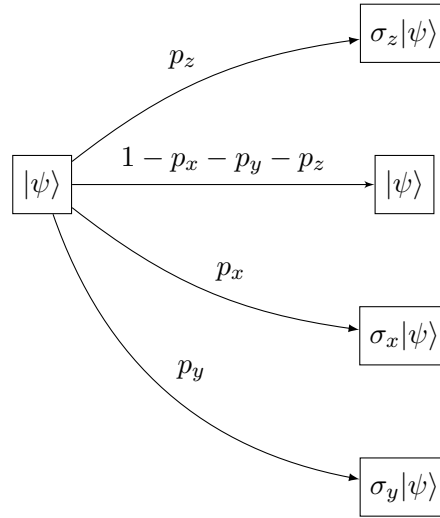


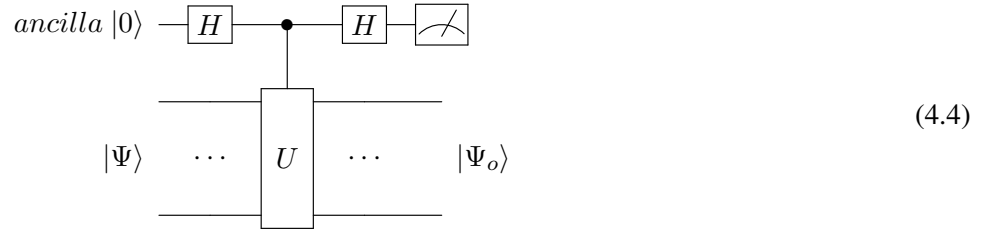
Figure 4.2: Depolarizing error channel with probabilities p_x , p_y and p_z .

One of the strengths of the depolarizing model is the ability to model even small noise effects such as the equivalent of a rotation gate over a small angle. This can be understood by considering a circuit composed of a single qubit gate $R_x(\theta)$ before measurement. If a systematic but small angle error $\delta\theta$ is applied every time at the gate, the projection over the measurement basis will only result in an error with probability $\sin^2 \delta\theta$. When sampled many times, the output distribution is equivalent to a circuit with a perfect rotation gate over which an additional σ_x is added to the circuit with probability $p_x = \sin^2 \delta\theta$. Depolarizing channel parameters are usually obtained empirically based on the experimental gate fidelity values. It is also possible to apply this channel for idle qubits when gates are being applied to other ones. One can use the coherence times T_1, T_2 of a qubit and define the depolarizing noise with probabilities [94]:

$$p_x = p_y = \frac{1 - e^{-t/T_1}}{4}, \quad p_z = \frac{1 - e^{-t/T_2}}{2} - \frac{1 - e^{-t/T_1}}{4} \quad (4.1)$$

In order to extract information from the logical qubit without changing it, a circuit called the operator measurement is used. For any unitary Hermitian matrix U of size 2^n , and any arbitrary quantum state $|\Psi\rangle$ composed of n qubits, $|\Psi\rangle$ can be projected as an eigenstate of U by the use of an additional qubit named an ancilla qubit,

prepared in $|0\rangle$ and measured for the sole purpose of this circuit:



The output state $|\Psi_o\rangle$ depends on the ancilla measurements M :

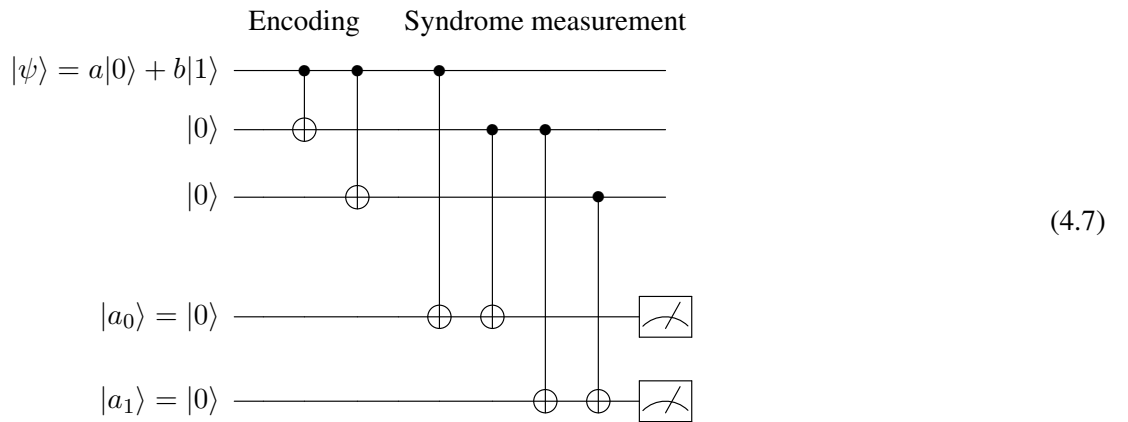
$$M = +1 \rightarrow |\Psi_o\rangle \propto |\Psi\rangle + U|\Psi\rangle \quad (4.5)$$

$$M = -1 \rightarrow |\Psi_o\rangle \propto |\Psi\rangle - U|\Psi\rangle \quad (4.6)$$

In both cases $|\Psi_o\rangle$ is an eigenstate of U because of the property $U \times U = I$. This circuit can be repeatedly applied to “maintain” $|\Psi\rangle$ as an eigenstate of U if that state is exposed to perturbation such as in the depolarizing model described in Section 4.1.2.

4.1.5 Correcting error with syndrome measurements

In the code described in Section 4.1.3, it is possible to extract information about errors with operator measurements and ancilla qubits. We consider the case where a single bit-flip error occurs on a three entangled qubit state. Two ancilla and CNOT gates are enough to measure the $Z1 \otimes Z2$ and $Z2 \otimes Z3$ operators. The eigenstates common to the operators $Z1 \otimes Z2$ and $Z2 \otimes Z3$, by using the formulation in Section 4.1.4, are the logical states $|0_L\rangle = |000\rangle$ and $|1_L\rangle = |111\rangle$. The sequence of the two measurement operators is the following circuit:



Each ancilla qubit assesses if a set of two neighboring qubits have experienced a bit flip error on one of them. Two ancilla qubits are therefore enough to deduce which qubit may have been affected as is represented in

Table 4.1. Measuring ancilla qubits to determine error patterns is called a syndrome measurement. An ancilla is called “flagged” if its measurement is different from the one expected for no error. If more than one error occurs in these 3 qubit codes, then either the wrong qubit will be flagged, which will lead to a wrong correction, or, for three errors, the code will indicate no errors. The result is, in both case, defined as a logical error.

Syndrome measurement of a 3 qubits bit flip error code		
Syndrome	Error	Qubit index
0,0	None	N.A
1,0	σ_x	1
1,1	σ_x	2
0,1	σ_x	3

Table 4.1: Error corresponding to the syndrome measurement of a 3 qubits bit-flip error codes. Two bits are enough to determine if and where an σ_x error occurred but no other errors can be distinguished.

This five qubit circuit can correct specific errors without affecting the qubit state by only measuring the ancilla prior to correction. As mentioned in the depolarizing model, both bit-flip and phase-flip, as well as a combination of both can happen. The two different error detection circuits can however be concatenated to define the Shor code, with nine data qubits for encoding the logical qubit and with eight ancilla qubits to provide the measurement syndrome. This syndrome is capable of correcting any single arbitrary error happening on the nine data qubits. This logical qubit, using the basis $|0_L\rangle = |0_{L,pf}0_{L,pf}0_{L,pf}\rangle$ and $|1_L\rangle = |1_{L,pf}1_{L,pf}1_{L,pf}\rangle$, is robust to one arbitrary error.

4.2 Running error corrected quantum circuits

The main problems are now defined: multiple types of errors are possible in qubits and schemes combining entangled data and ancilla qubits enable detection and correction of a limited occurrence of those errors. To satisfy the DiVincenzo criteria and build logical quantum computers, those newly defined logical qubits also require logical gates as well as logical preparations and measurements. We now generalize the previous example and define how to run quantum circuits based on logical qubits.

4.2.1 The stabilizer formalism

The Shor code possesses two remarkable properties: The measurement syndrome used to detect errors does not modify the information encoded in the logical qubit and logical gates manipulating this logical qubit commute with the measurement syndrome circuit. The Shor code is actually part of a class of quantum code sharing the same properties, the stabilizer codes.

A stabilizer code is described as follow [95]: For Hilbert spaces of dimension 2^n defined by n qubits we define a coding space T encoding k logical qubits on a 2^k Hilbert space generated by the stabilizer S , a subgroup of Pauli matrices of 2^{n-k} elements, such that

$$T = \{|\psi\rangle, \text{ such that } M|\psi\rangle = |\psi\rangle \forall M \in S\} \quad (4.8)$$

Any state $|\psi\rangle$ can be projected in S by sequentially applying all the stabilizers M belonging to S using the operator measurement circuit described in Section 4.1.4. As previously encountered in Section 4.1.5, each stabilizer measurement will provide an outcome of eigenvalues +1 or -1. A cycle of stabilizer measurements, in which each stabilizer of the logical qubit is measured, provides a syndrome, a list of the measurement results for all the ancilla qubits. Repeatedly measuring the stabilizers in these cycles allows for monitoring the appearance of errors, which will cause a local sign change in the syndrome. The stabilizer measurements “stabilize” the code space, projecting the system either back into the space spanned by the logical qubit basis states, or into a state having a bit-flip or phase-flip error with a corresponding measurement syndrome. If the errors are below the maximum number of correctable errors, the measurement syndrome will indicate the error location and type, and can then be corrected by applying additional gates to restore the logical qubit. Stabilizers are characterized by the concept of weight:

- Operator weight : number of non-unity Pauli matrices present in the operator.

As an example, let us consider the circuit in 4.7 protecting from at most one bit flip error. The codewords are the $|000\rangle$ and $|111\rangle$ states, and the two stabilizers are $Z1 \otimes Z2$ and $Z2 \otimes Z3$, which are both weight-2 stabilizers. If we consider a superposition between a codeword $|000\rangle$, and a non codeword state $|100\rangle$ - due to an over-rotation or a small error probability for a bit flip error to happen - the stabilizer measurement will project the state into either $|000\rangle$, with no resulting correction to apply, or the non codeword $|100\rangle$ with the syndrome measurement indicating the first qubit being the location of the error which needs to be corrected. In both cases, after measurement and correction, the state is again the codeword $|000\rangle$.

Additionally, we can find operators not in S but which commute with every element of S . They therefore share the same eigenvectors in T but can transform between any states within. This “degree of freedom” in the code space enables the selection of logical qubit eigenvectors and logical operators acting on those qubits which we call logical gates. A logical qubit is characterized by its distance:

- Code distance : Minimum weight of the logical operators defining the logical qubit gates.

A code of distance n can correct at most $(n - 1)/2$ arbitrary errors: a distance 3 logical qubit is the minimum distance, therefore, to be capable of correcting one arbitrary error.

As a final remark, in addition to representing stabilizer codes by the basis $|0_L\rangle$ and $|1_L\rangle$ encoding the logical qubit, an alternative representation consists of listing the stabilizer operators. This representation is both more concise and more practical for implementing quantum error corrections circuits.

4.2.2 Logical circuits

In order for logical qubits to replace physical ones, each operation of a quantum computers, such as single qubit gates, entangling qubits, preparation and measurements would need their own logical version. Each logical operation would be followed by an error measurement cycle using ancilla qubits to detect any occurring errors. Errors can be corrected physically after each cycle, or a recording of every error can be stored to deduce the expected data qubit state after their measurement. The second method, while increasing the complexity of the classical data processing, does not add additional sources of errors to the quantum circuits.

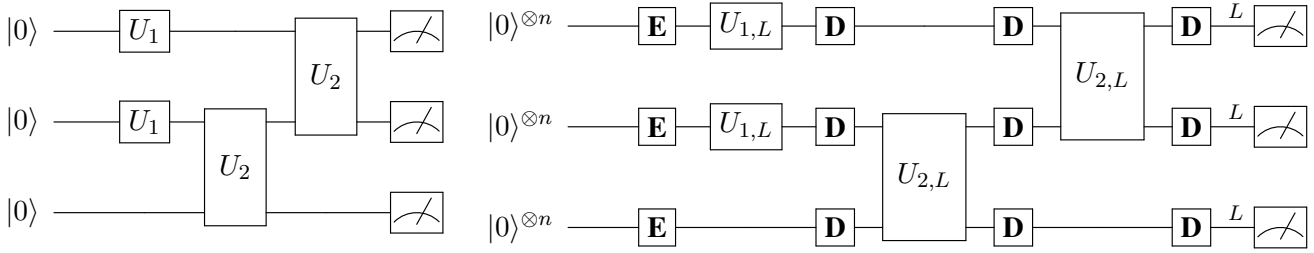


Figure 4.3: Comparison between a physical quantum circuit (Left) and its logical equivalent (Right). The logical circuit is considered here to have an identical logical gate set $U_{i,L}$ to the physical gate set U_i . Each logical qubit needs to be encoded before applying logical gates (here represented with the $\mathbf{E}$ operator). After each step in the circuit, the stabilizers $\mathbf{D}$ are applied to determine any errors which have occurred.

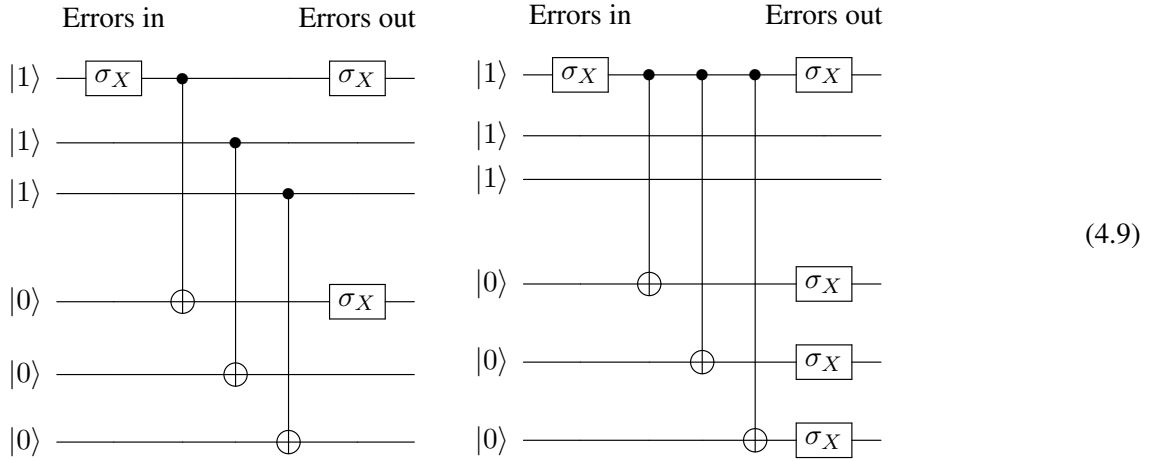
The logical gate set needs to be diverse enough to enable universal quantum computation, but does not need to represent the same gates as its physical counterparts. An abstract example is given Figure 4.3: The circuit is similar to the one described in Chapter 1 but with logical operations applied to sets of physical qubits and with additional quantum error correction cycles using ancilla qubits.

4.2.3 The fault tolerance threshold

QEC codes are aimed at running logical circuits with logical qubits with lower error rates than their physical equivalent. Similar to the concatenation of classical error code to arbitrarily increase the quality of the logical

bit, we search quantum error code whose robustness to noise increases with the number of physical qubits. Each logical operation needs to be fault tolerant: if an error occurs on a physical qubit in or before a logical operation then the error cannot propagate to more than one physical qubit per logical qubit involved. This condition prevents errors propagating faster than they can be corrected [3, 96]. A fault tolerant logical single qubit gates applied to a logical qubit with one faulty physical qubit in input should output a logical qubit with at most one faulty physical qubit. In the case of a fault tolerant logical two qubit gates between two logical qubits, one having a faulty physical qubit, the two logical qubits in output should have each, at most, one faulty physical qubit.

A two qubit gate example is illustrated [97] with two logical qubits $|1_L\rangle = |111\rangle$ and $|0_L\rangle = |000\rangle$ entangled via a logical CNOT gate to form a logical Bell state. The following circuits represent two different logical CNOT gates, and the propagation of a bit-flip error equivalent to a σ_X gate when a single bit-flip error is present on $|1_L\rangle$:



The first logical CNOT gate leads to one single additional error in the other logical qubit, and thus is fault tolerant. The second logical CNOT gate leads however, to several additional errors and does not satisfy the criteria for fault tolerance.

For each of these fault tolerant codes and for a given error model, a figure of interest is the error rate below which a logical circuit has lower error rates than their physical equivalent. We can define a value p_{th} - the fault tolerance threshold - for a given QEC scheme and error model: the physical error rate below which any desired logical error rate can be achieved, by enlisting more physical qubits per logical qubit. Intuitively this is the physical error rate the code can “deal” with, where the errors introduced by the added complexity of the QEC circuits is in balance with the rate errors are corrected.

This value is an approximation of the several error probabilities involved in the error model, but gives an efficient target for physical devices to reach. Once a device is capable of implementing beneficial quantum

error code correction, any circuit depth can be implemented by increasing the number of qubits involved in the logical qubit.

4.3 The surface code

Now that most of the tools necessary to understand fault tolerant quantum computation and quantum error correction have been introduced, the most promising scheme to implement on quantum hardware, the surface code, is considered. The surface code shows a very high error rate threshold of 1% for homogeneous depolarizing error models [98].

4.3.1 Surface code stabilizer and the decoding problem

The surface code is a topological quantum error correction scheme with only nearest neighbor interaction which can therefore be used by virtually all quantum computing architectures, even those with limited connectivity as mentioned in Section 1.1.4. It is represented as a 2-D grid containing data qubits, with an ancilla qubit in the center of each square as displayed in Figure 4.4. Each square, commonly called a plaquette, is associated with a single weight 4 stabilizer with which the ancilla qubit is used to measure the syndrome. There are two types of plaquette, measuring in the Z and X basis, and are therefore called Z and X plaquettes.

Unfortunately the initial register $|0\rangle^{\otimes n}$ is not a simultaneous eigenstate of all of the stabilizers, and thus is not a suitable choice for the basis state $|0_L\rangle, |1_L\rangle$. Initialization of a logical qubit consists of applying a cycle of syndrome measurements on this register. The initial ancilla measurement will output a sequence of random bits, but it will also project the data qubits into the stabilizer eigenspace, providing a suitable basis state $|0_L\rangle$. This logical state, is commonly called the quiescent state, and is associated with bit string from the first syndrome measurement. Any subsequent syndrome measurements will provide the same bit string unless errors occur. Similarly to Section 4.1.5, we will consider an ancilla qubit being “flagged” if it detects an error, defined as its measurement being different from the expected measurement from the initial quiescent state.

When multiple cycles are run in time, a 3-D syndrome map in space and time is obtained. Different types of errors such as σ_x , σ_z or measurement errors can then be deduced as displayed in Figure 4.5: $\sigma_x(\sigma_z)$ errors are flagged by 2 neighbor ancilla qubits of a Z(X) stabilizer near the data qubit. σ_y errors flag both X and Z stabilizers. Measurement errors flag only a single ancilla qubit over multiple cycles. However for large and noisy systems running surface codes, more complex features can appear in the syndrome map due to multiple

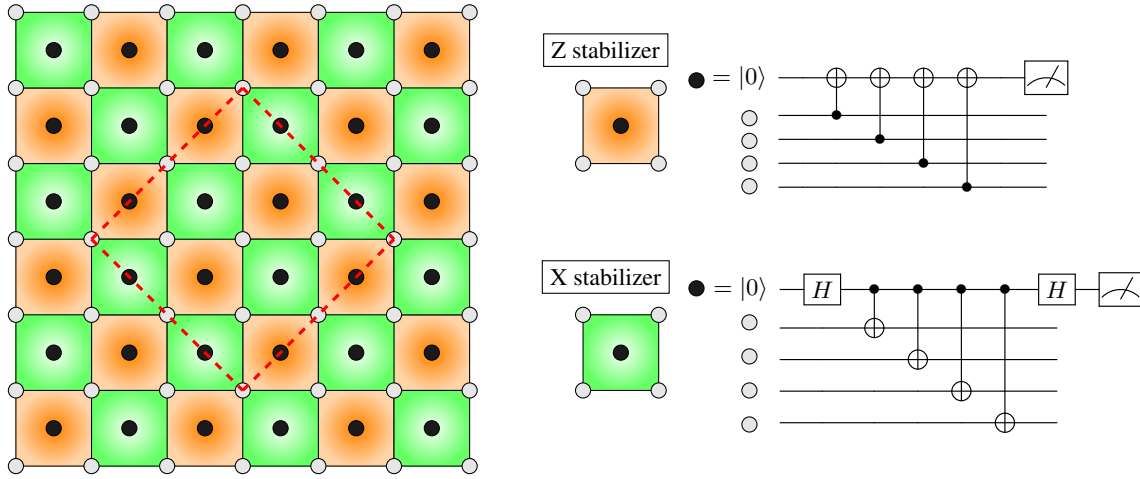


Figure 4.4: Surface code representation using data qubits (Grey circles) and ancilla qubits (Black circles). Each ancilla qubit is either in a Z stabilizer (Orange) or in an X stabilizer (Green), for which the quantum circuits are represented on the right. The red square on the left represents the surface needed to encode a logical qubit resistant to at most one error. The logical qubit is "cut" from the surface code by not applying the stabilizing operations on ancilla qubit outside the red area. These cut along the Z stabilizer and X stabilizer introduce a degree of freedom in the space of states contained in the red area, thus defining the logical qubit.

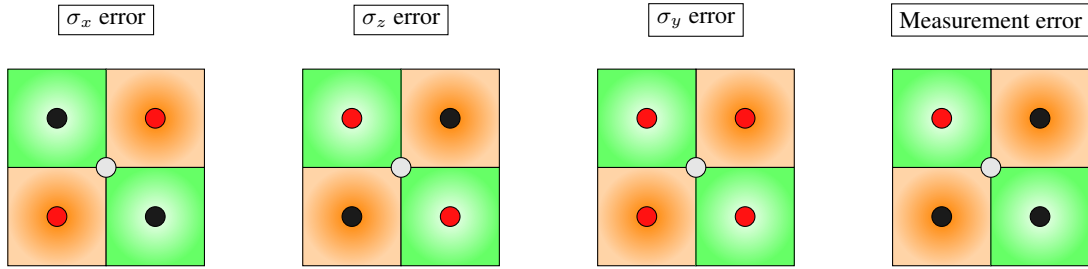


Figure 4.5: Decoding of a Surface code for a single data qubit, assuming no errors occurring on any other data qubits nor on the ancilla qubits. Standard syndrome measurements show the ancilla qubit in black, while ancilla qubits measuring a sign flip are displayed in red. In these single error scenarios, the different Pauli errors are distinguishable as well as SPAM errors.

errors occurring on neighboring qubits.

Designing a fast and accurate decoder is still an active field of research due to the large amount of data accumulated at each cycle and the high dependence of the error model. Among a large variety of schemes one can use minimum weight matching [99], maximum likelihood [100] and neural-network decoders [101].

4.3.2 Logical qubits and logical gates

We now describe a way to define single logical qubits and how to implement a universal gate set with the surface code QEC.

Individual qubits are defined by introducing degrees of freedom in the stabilizer grid. Among the different

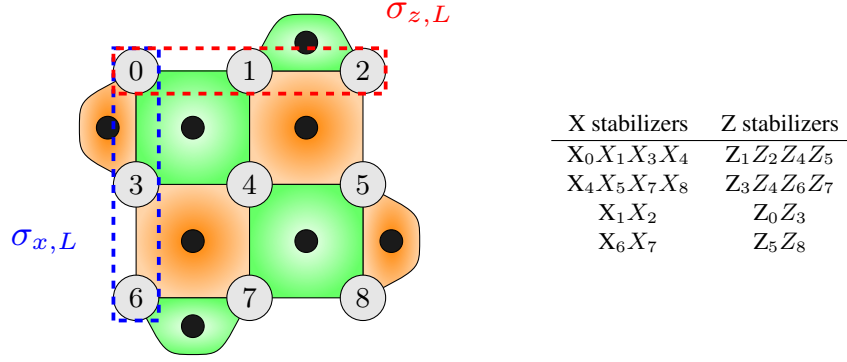


Figure 4.6: Reduced single logical qubit encoded with the surface code. This configuration only needs nine data qubits and eight ancilla qubits, by using weight-2 stabilizers in addition of the standard weight-4 stabilizers considered earlier. The stabilizers are represented on the right. The logical operators $\sigma_{x,L}$ and $\sigma_{z,L}$ can be applied by running physical $\sigma_{x,L}$ ($\sigma_{z,L}$) in the data qubits delimited by the dashed Blue (Red) box.

approaches available, a popular one consists in isolating the stabilizers inside a delimited area of the surface code. The surface delimited depends on the number of errors the logical qubit is designed to be resilient against: a distance 3 code can correct 1 physical error per QEC cycle. The minimal logical qubit, correcting at most one error, is shown in red in Figure 4.4 and uses twenty-five qubits with weight-3 and weight-4 stabilizers. Some ancilla qubits can be removed at the cost of reducing to weight-2 some stabilizers, while maintaining the same distance 3. The resulting logical qubit has its rearranged stabilizers displayed in Figure 4.6.

This logical qubit uses nine data qubits and eight ancilla qubits. Two Pauli gates $\sigma_{x,L}$ and $\sigma_{z,L}$ are implemented by running a sequence of σ_x (or σ_z) on three neighboring data qubits. Logical Hadamard gates can be encoded by deformations in the surface encoding the logical qubits [102]. CNOT gates are encoded via a third ancilla logical qubit in a surface deformation process called lattice surgery [103].

Not all gates can be performed fault tolerantly within the surface code to complete a fault tolerant universal gate set: Instead those gates are constructed first by building, with a high enough fidelity, a specific state in a process called magic state distillation [104], then, as displayed Figure 4.7, by mapping the magical state to the logical state, the desired logical quantum gate can be applied effectively. Two important gates requiring magic states are the phase gates $T=R_z(\pi/4)$ and $S=R_z(\pi/2)$. They require the magic states, also called logical auxiliary states initialized in $|S_L\rangle = \frac{1}{\sqrt{2}}(|0_L\rangle + i|1_L\rangle)$ and in $|T_L\rangle = \frac{1}{\sqrt{2}}(|0_L\rangle + e^{i\pi/4}|1_L\rangle)$. The Magic state distillation is one of the most costly resources in a fault-tolerant quantum computation with a surface code, as many physical $|S\rangle/|T\rangle$ states are required produce high enough quality $|S_L\rangle/|T_L\rangle$ states [105].

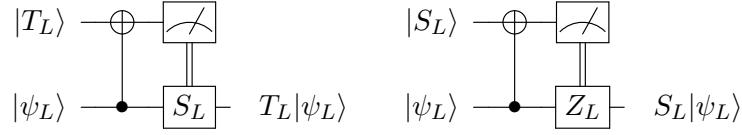


Figure 4.7: Logical T and S gates by consuming a magic state. Both schemes use logical CNOT gates and on-the-fly measurements. The T gate is more costly in space and time resources because it itself requires an S gate. Both $|S_L\rangle$ and $|T_L\rangle$ states are obtained via the costly state distillation process.

4.4 The color code

The surface code, with its nearest neighbor interaction only feature, is a promising QEC code for every architectures. But because QCCD architectures are virtually globally connected via shuttling, a similar QEC code is getting attention due its transversal gate capabilities: the color code [106].

4.4.1 Description

The color code is a topological QEC code, using data qubits as vertices. Each vertex is connected to three other vertices, allowing the plaquettes, the surfaces delimited by the vertices, to have three different and non-adjacent colors. Each plaquette is associated with two measurement operators requiring therefore two ancilla qubit per plaquette. Similarly to the surface code, the color code consists in running periodically the measuring operators to deduce and correct, via measurement of the ancilla qubits, errors occurring on the data qubit.

A derivative of the color code, called the triangular code, starts from a static and fully define color code on a sphere and removes a single vertex and its three surrounding plaquettes. The resulting code has a single degree of freedom used to encode the logical qubit. The minimal amount of qubits needed for this logical qubit is seven data qubits and six ancilla qubits, and both its representation and the associated stabilizers are displayed in Figure 4.8.

With the exception of T gates, requiring state distillation similarly to the surface code, all logical gates are transversal, which means logical operations can be implemented by simply applying the equivalent physical operation to each data qubit encoding the logical qubit, as illustrated in Figure 4.9. A logical CNOT gate is implemented similarly, in running physical CNOT gates between the corresponding data qubits of the two logical qubits.

Such considerable simplification comes at the cost of a more constraining error threshold of 0.08% [107] and, in the QCCD architecture described in Section 2.4.2, an increase in the amount of shuttling sequences needed to run. Indeed, any logical qubit on a surface trap could be directly entangled at the cost of shuttling all their

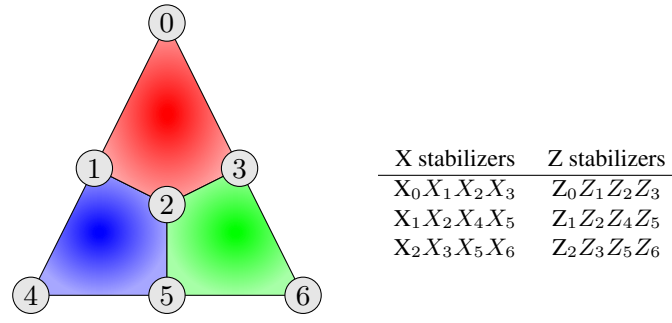


Figure 4.8: Single logical qubit encoded with the color code. This configuration only needs seven data qubits and six ancilla qubits, by using six weight-4 stabilizers as displayed on the right.

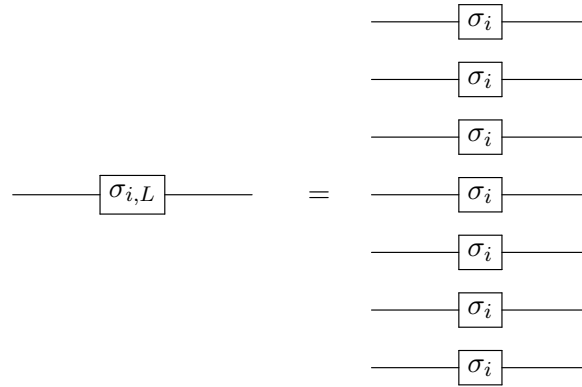


Figure 4.9: Transversality of logical gates for color qubits. A logical single qubit gate, for $\sigma_i \in [\sigma_x, \sigma_z]$, H and S, consists of applying the equivalent physical gate to each data qubit belonging to that logical qubit.

corresponding physical qubits to the same zone. As mentioned in Section 2.4.1, shuttling ions adds heating and unwanted phase rotations. The advantages of transversal gates can easily be lost due to noisy shuttling parameters [108], but recent high fidelity shuttling of trapped ion qubits [70] indicate it could be implemented in the near future. Few experiments on the color code have yet been conducted [109].

4.4.2 Application to large scale trapped ion devices

The transversal representation of color codes is conceptually easier to represent, but in the case of the gate architecture using magnetic field gradients described in Chapter 2, it is also efficient to implement in hardware. The applied RF and MW field can, in principle, drive the same gate at multiple gate zones simultaneously therefore, a device with at least one gate zone per qubit does not require modifying the antenna hardware when scaling circuits from physical qubits to logical qubits. A set of fields driving gates on a physical qubit will also drive the entire logical qubit.

For devices satisfying the condition for this efficient scaling of the driving field, the experimental control can be

further abstracted. The trap area delimiting the zones containing the logical qubit can be abstracted, similarly to how logical qubits are constructed from physical qubits, into a logical zone. Each logical zone would host one logical qubit by default, and be able to prepare, measure, and apply single logical qubit gates. These zones can apply entangling logical gates by “shuttling” a second logical qubit into the same logical zone by shuttling all the corresponding physical qubits.

4.5 Practical implementation of QEC

After describing the fundamentals of QEC and detailing the implementation of two particular schemes, we now consider two additional problems associated with QEC which are how to effectively simulate such codes and how to experimentally implement such codes into hardware.

4.5.1 Efficient simulation by importance based sampling

When simulating logical qubits with an error model such as the depolarizing noise, the circuit encoding the qubit is run with noise gates inserted with a probability associated to the noise parameters. For low noise behavior, most simulations run without error gates and therefore waste computing time without giving any information about code performance.

An efficient alternative is called importance based sampling [110]: an error rate $p_L(n)$ is found for a fixed number of errors, n , placed randomly in a circuit. By calculating this rate for multiple values of n and summing these with weights $A_n(p_e)$ corresponding to the number of ways each number of errors can happen associated with the probability for the error to happen, we can find a logical error rate $p_L(p_e)$. We can define a cut off at a certain value of n , the total probability of having this many errors or more in a cycle being very small, by assuming all weight n and above errors result in a logical error. The final error rate p_L is deduced from [110]:

$$p_L(p_e) = \sum_n^{N_{arb}} A_n(p_e) p_L(n) \quad (4.10)$$

Importance sampling has two interesting features for logical code simulations: firstly, when simulating the circuit no error probability is considered, which means that for the same error model, a different error rate can be considered just in post-processing and not in running new simulation. Secondly, this model can be extended to more error parameters, such as the distinction between single qubit gate errors and two qubit gates error, but also rotation errors, heating [42] and shuttling [108] effects.

4.5.2 Experimental effort toward 17 surface code

Experimental demonstrations are far from demonstrating beneficial quantum error correction and careful consideration of experimental implementation decomposes progress toward fully beneficial QEC into two main steps [111]. Firstly, a logical qubit undergoing error detection and correction cycles should remain coherent longer than the physical qubits of which it is comprised of. Syndrome measurement cycles in QEC adds complexity to the quantum circuit and therefore opportunities for errors to occur. Beneficial corrections cycles would indicate that the syndrome measurements correct more errors than they introduce. This step is fundamental for longer lasting quantum memory but does not involve the use of fault tolerant operation and fault tolerant circuits. The second step is to demonstrate that not only logical qubits have better lifetimes than physical qubits but that they can also be coherently controlled with logical gates.

Despite very focused consideration of running a 17 qubit surface code [112, 113], no experiments have yet been fully conducted. Most results were obtained for individual stabilizers [114, 115], but promising results on experimental lattice surgery with smaller encoding [116], combined with the increase in quantum volume of available devices indicate that such experiments are now within reach.

4.5.3 Hardware requirements for testing QEC in a QCCD architecture

A fully functional large scale QCCD quantum computer is not required to study QEC protocols. The needed functionalities can be summarized as:

- A segmented ion trap. Each electrode should be individually controlled to enable shuttling between trapping zones. The smallest trapping unit described in the blueprint from the research group [1] is the X-junction surface trap which provides at least an independent zone for each arm of the X-junction, therefore allowing enough space to run circuits from a single stabilizer up to a full logical qubit. Shuttling across the junction, shuttling in the corner of the junction, splitting and merging an ion crystal, and linear shuttling are the required waveforms to fully manipulate a chain of trapped ions in an X-junctions.
- A quantum logic control capability. At least one zone where a magnetic field gradient can be generated on demand to drive, via MW and RF fields, the desired quantum gates. The device also needs to achieve quantum state preparation and readout on selected zones while maintaining qubits in other zones to protect the data qubits while the ancilla qubits are measured. Single and two qubit gate fidelity should remain above 99% after a full stabilizer circuit involving shuttling. QEC codes do not have constraints

on the measurement time [117], but the state detection efficiency is required to be at least equivalent to the gate fidelity for the surface code [98], and therefore is also required to be above 99%.

- A compilation interface to translate the quantum circuits representing the QEC into a series of physical pulses representing shuttling and logic state control. The hardware needs to be able to sustain long and diverse shuttling sequences and gate pulses to enable high depth quantum circuits generated by the multiple cycles of stabilizer measurements. As a first objective with a static compiler, the apparatus should be capable of running a single stabilizer measurement involving five qubits.
- A feed-forward capability to adjust the quantum circuit on the fly based on in-circuit measurements. While feed-forward is not fundamentally needed for compensating errors detected with the stabilizers; only tracking the errors is required; losing ions is a perturbation event that can only be compensated by bringing new ions into the quantum circuit. Feed-forwarding involves therefore, the integration of the quantum state readout analysis and the compiler into the real-time experimental control. This final functionality will enable us to run multiple series of stabilizer measurements with a correction step after each error detection, as well as being able to inject a new ion in the circuit when detecting an ion loss.

The gate fidelity requirement enables the deduction of an additional constraint on the ion trap heating rate, in order to maintain the entangling gate error below 1%, and assuming a negligible contribution from the other sources of noise. For a two-tone polychromatic gate, the error is defined as [2]:

$$\varepsilon_{\dot{n}} \simeq 1 - \frac{1}{8} \left(3 + 4e^{-\frac{\dot{n}t_g}{6}} + e^{-\frac{2\dot{n}t_g}{3}} \right) \quad (4.11)$$

Considering the previously implemented polychromatic gate having $t_g = 3.4$ ms for $\delta B/\delta z = 23.6$ T/m, an ideal gradient of 140 T/m enables $t_g = 570$ μ s. The trap heating rate can be constrained to be:

$$\dot{n} < 11 s^{-1} \quad (4.12)$$

Additional hardware requirements can be obtained when taking shuttling into account and will be facilitated once the compiler is formulated. Heating and time constraints associated with shuttling can be deduced from the compiled circuit of a single stabilizer, in order to maintain gate fidelities, across the entire circuit, above 99%.

The timing requirements associated with the feed-forward capability are associated with, first, the time taken to estimate the measured state in real-time and secondly, the time taken to decode the error syndrome and the

choice of the subsequent quantum circuit to apply. The measurement is not a constraint due to the existence of low-resource approaches such as the threshold method as already considered in Section 2.3.5. More expensive methods, such as machine learning assisted readout, will slow down the total readout operation and therefore may impact the gate fidelity considered just above.

The error syndrome decoding will require, for large scale architecture and large error rates, dedicated time to compute the most probable syndrome following the schemes mentioned in Section 4.3.1. However in the case of an experiment only involving the logical qubit composed of seventeen physical qubit, detailed in Section 4.3.2, eight ancilla qubits are used and therefore only 256 different syndromes exist. All the resulting errors and resulting correction can be calibrated by the experiment and would then be cost-less to read in real-time [113]. Time constraints can in conclusion be neglected for experiments up to a few logical qubits. The physical implementation of feed-forward still needs to be considered, first in the compiler, then in each hardware device handling the quantum control fields and the shuttling waveforms.

Chapter 5

Quantum software compilation

This section will detail more abstract software levels of a quantum computer in the context of trapped ion Noisy intermediate scale quantum (NISQ) devices based on trapped ions. The quantum circuit representation detailed in Chapter 1 is both still too low level for general users to fully exploit quantum computers and too high level to represent the physical operation to achieve in the hardware. More intermediate representations are required so that each compilation step can be studied, implemented and updated independently. While those software layers can be avoided in most prototypes with very low qubit counts, medium size circuits such as the one needed for QEC codes detailed in Chapter 4, will have its output quality significantly increased depending on the optimization achieved in software. On the long term with larger device continuously employing topological QEC codes, such layers will become indispensable. A full stack implementation at NISQ level consisting of a gate decomposition layer, a compilation to hardware layer and a pure hardware layer is detailed and illustrated with important user cases. Implementing QEC codes in the above framework, not as the main circuit to run but as a layer itself, is also considered.

5.1 Choosing a framework for writing and compiling quantum code

5.1.1 Stack representation of a trapped ion quantum computer

For clarity we adopt the following definitions:

- **Quantum code:** Text representation of a quantum circuit

- **Syntax:** Literal representation of quantum code. Any algorithm in two different syntaxes must still have the same probability distribution if run on a perfect simulator.
- **Command:** Single instruction of quantum code
- **Layer:** Code that receives a quantum code of syntax A and emits a quantum code with syntax B; A and B can be the same.
- **Compiler:** Layer adapting the quantum code to a subset of physical constraints associated with quantum computing hardware
- **Backend:** Layer running the quantum code and producing the measurements. Backends are mostly either a simulator or quantum hardware, but alternative backends can for example draw circuits as images or perform circuit analysis.
- **Framework:** Code capable of handling a set of several layers. They can handle several quantum code syntaxes as well as a choice of several backends. They allow users to run arbitrary code on arbitrary backends and get the probability distribution outcome effortlessly from a user point of view.

We also need to distinguish the different denominations given to qubits:

- **Physical qubit:** Physical entity encoding the qubit in the quantum hardware. In this thesis the physical qubits are individual $^{171}\text{Yb}^+$ ions.
- **Computational qubit:** Qubit defined in a circuit and attached to a physical qubit via a map handled by the framework.
- **Logical qubit:** Qubit used for quantum computation. If a quantum error correction scheme is used then the framework handles a mapping of one-to-many map to decompose logical qubits into circuits made of computational qubits. If no scheme is used then the logical qubits are the same as the computational qubits.

The framework structure used in this chapter is illustrated in Figure 5.1 and is the widely adopted representation in the community for running code on quantum computer [118, 119]. Each layer is defined as an optimizer which receives the code written in a certain syntax as input, and produces the equivalent code written in a more hardware dependent syntax. Such organization allows a clear separation of tasks between layers. Defining the several syntaxes used in between for the quantum code will provide clear objectives for the development of those layers.

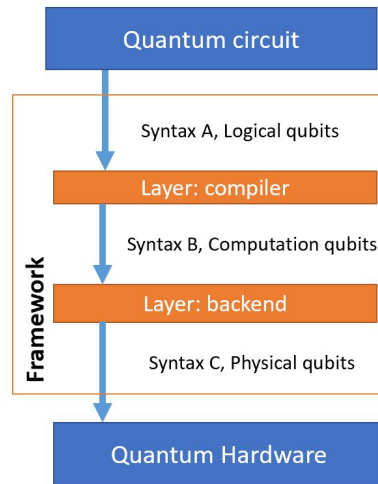


Figure 5.1: Generic framework structure for quantum computers. A generic quantum program is first translated in a quantum code (Syntax A) and progressively adapted to the hardware by successive layers of compilation (Syntax B). The outcome (Syntax C) is a quantum code that can be directly implemented in a quantum hardware. The number of layers as well as the choice of syntax can be adjusted.

5.1.2 Choice of quantum framework

Nowadays a great number of softwares for quantum computation are available to formulate quantum circuits, but still few of them are able to act as a full framework [120]: they not only need a layer structure but also at least one syntax for interacting with a backend. We can quickly summarize them in Table 5.1: First of all, most of them are open source python modules, aimed mainly for education and research purposes, and all provide both access to simulators and to a quantum hardware. Most frameworks have recently acquired the capacity to interact with different QPU architectures demonstrating the flexibility required from a quantum software framework. The last two elements of the table are more intriguing as they represent purely private efforts to monetize quantum computing via their cloud platform, and are only available yet as early access. However they aim to reach the same level as the other frameworks as they can compile the customer’s quantum code to different quantum computing architectures. The list presented in Table 5.1 is not exhaustive and alternative frameworks can be found [121, 122].

ProjectQ [123] was selected amongst the other frameworks for its easier internal mechanism and its better independence from existing quantum hardware companies. The departure of the original founders of the project occurring the same year in which we started to write updates was on one hand detrimental to its popularity but permitted a greater freedom in implementing new features.

¹<https://github.com/Qiskit/qiskit>

²<https://github.com/quantumlib/cirq>

Quantum software framework			
Name	Company	Backends available	Hardware based syntax name
Qiskit ¹ [124]	IBM	IBM, AQT	OpenQASM
Cirq ²	Google	Google, AQT, Pasqal	N.A.
Forest ³	Rigetti	Rigetti	QUIL
ProjectQ ⁴ [123]	N.A.	IBM, AQT	N.A.
Azure Quantum ⁵	Microsoft	IonQ, Honeywell, 1QBit and QCI	Q#
Amazon Braket ⁶	Amazon	IonQ, Rigetti and D-Wave	Unknown

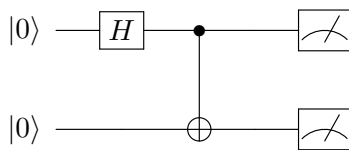
Table 5.1: List of the main quantum software frameworks available. The first four elements are open source python modules whereas the last two are cloud based frameworks in early access. No descriptions for them are currently available.

5.1.3 The ProjectQ syntax

ProjectQ works with layer objects called engines, which are the main components. They are wrapped in a special engine object, called the MainEngine, holding information about the qubits and the list of engines to handle. The MainEngine is the entry point for any commands, holds the list of engines chosen to compile the circuit and also handle the qubit register. All commands circulate through the engines down to the backend. A command has the following syntax which will be used for the rest of the thesis:

$$\{command\ name\} \mid \{qubit\ number\}$$

A Bell state formed by one Hadamard gate followed by a CNOT gate is expressed in the ProjectQ syntax in Figure 5.2.



```

1 eng=projectq.MainEngine()
2 #Qubit initialization
3 qureg = eng.allocate_qureg(2)
4 #Quantum circuit
5 H | qureg[0]
6 CNOT | (qureg[0],qureg[1])
7 All(Measure) | qureg

```

Figure 5.2: Adaptation of a quantum circuit (Left) into the ProjectQ quantum code syntax (Right). The framework main object called the MainEngine (Line 1) contains the qubit information (Line 3). The circuit itself is implemented by overriding the python OR operator “|” (Line 5-7).

Engines requiring the knowledge of multiple commands, like a circuit optimizer, can buffer the incoming

³<https://github.com/rigetti/pyquil>

⁴<https://github.com/ProjectQ-Framework/ProjectQ>

⁵<https://azure.microsoft.com/en-us/services/quantum/>

⁶<https://aws.amazon.com/braket/>

commands. Storing and releasing commands is controlled by each engine, so to avoid any potential stop in the flow of commands through the engines, a special command called “flush” forces all engines to release all buffered commands down to the backend. I acquired experience on the ProjectQ framework by writing two updates on the IBMQ backend^{7,8} to match the corresponding API updates in Qiskit, to interact with the IBM quantum machines, as well as implementing the backend for the Alpine Quantum Technology (AQT) quantum machines⁹.

5.1.4 The ProjectQ syntax efficiency

The choice of the syntax shown above is to stay coherent with the python code used in ProjectQ and was not defined nor particularly studied to be an efficient hardware based quantum code. Several languages exist, and are usually co-designed with existing frameworks: among many examples there is QUIL[125] with Forest, OpenQASM[126] with Qiskit and cQASM[127] with OpenQL. Some provide powerful functionality such as lambda functions, variable parameters for VQE circuits [128] as well as in-circuit measurement with subsequent logic decision making. This plurality of quantum code syntax was recently acknowledged by Sandia National Laboratories with the introduction of their own syntax called Just Another Quantum Assembly Language (JAQAL)[129]. The ProjectQ syntax is limited but the framework is easy to reconfigure allowing extensions.

5.2 Adapting a quantum software framework to trapped ion quantum computers

A quantum compilation architecture has been developed in the research group using ProjectQ. This explicit focus on the technologies used enables a more detailed description of the needed layers and is displayed in Figure 5.3: At the highest level, functions consists of entire quantum algorithms, such as the Shor algorithm. The Quantum code decomposer then translates these algorithms into quantum circuits, as well as the logical qubits into sets of physical qubits. NISQ devices have a low qubit count and thus being usually too small for such a feature, this layer will not be a priority in this project and will remain ignored. The second layer optimizes the circuit and adapts it to the gate operation implementable by the quantum hardware. The third layer adapts the quantum circuit to a set of gate and shuttling operations characteristic of QCCD devices. Finally,

⁷<https://github.com/ProjectQ-Framework/ProjectQ/pull/349>

⁸<https://github.com/ProjectQ-Framework/ProjectQ/pull/366>

⁹<https://github.com/ProjectQ-Framework/ProjectQ/pull/353>

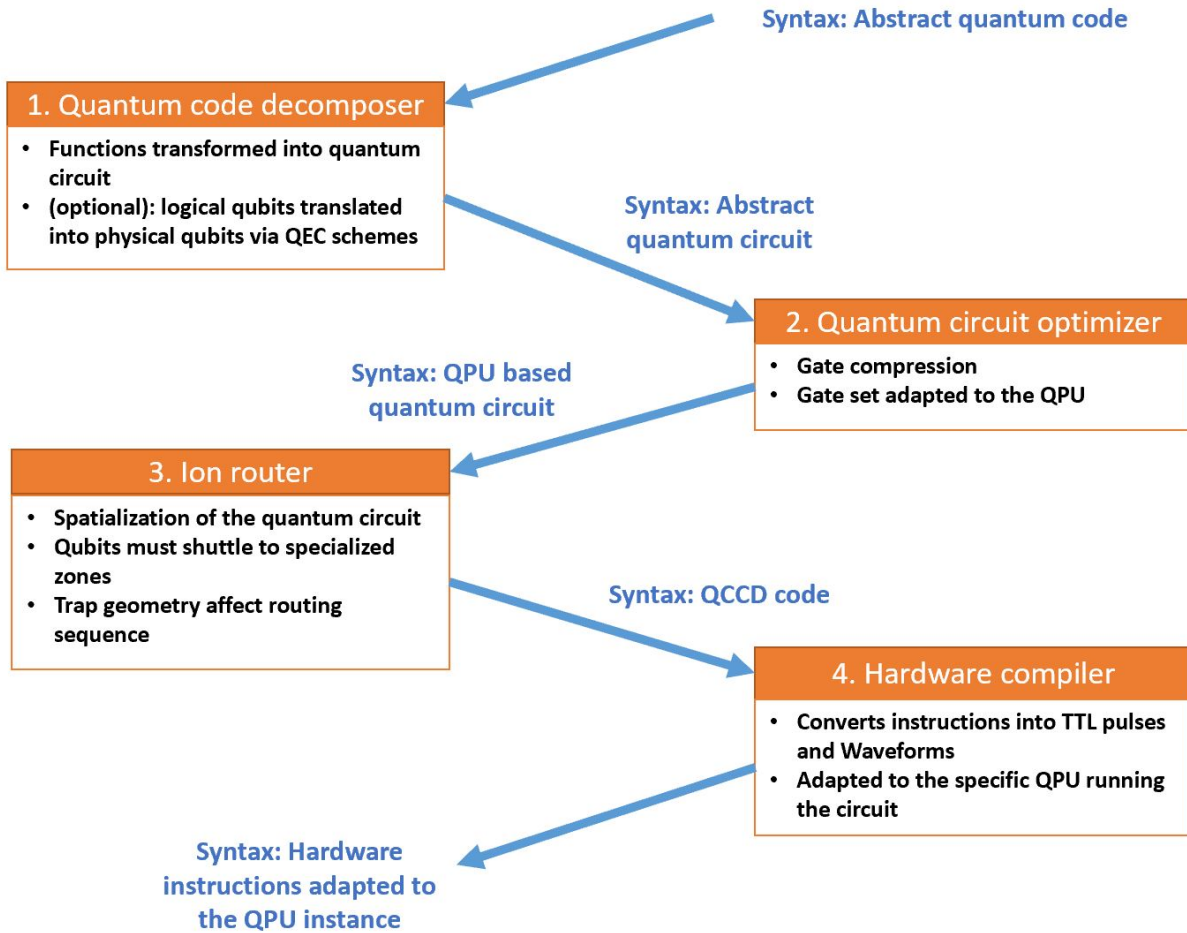


Figure 5.3: Layer structure of the quantum compilation architecture implemented in the research group. Layers are represented by the orange blocks summarizing their function while the blue arrows indicate an intermediate syntax to host the quantum code. The first layer translates high level function into computational qubits. The second layer optimizes the circuit by reducing the gate count and adapting the gate set to the hardware. The third layer translates computational qubit into physical qubits by adding shuttling commands to the quantum code. The last layer translates the commands into hardware instructions.

gates and shuttling sequences are transformed into the relevant set of hardware pulses to be implemented by the experiment. All syntaxes are aimed to be common for all experiments across the research group and detailed enough to be reused by other ion trapping groups or companies.

In order to illustrate all the layers in the following sections the simple example of a Greenberger - Horne - Zeilinger (GHZ) state with three qubits [130] will be used. A GHZ state in the abstract quantum code formalism, can be formed with a high level gate we will define simply as the GHZ gate and with three qubits

initialized in a register q . The evolution operator of this GHZ gate is:

$$GHZ = \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix} \quad (5.1)$$

The first layer, ignoring QEC code, decomposes this gate into single and two qubit gates as shown in Figure 5.4:

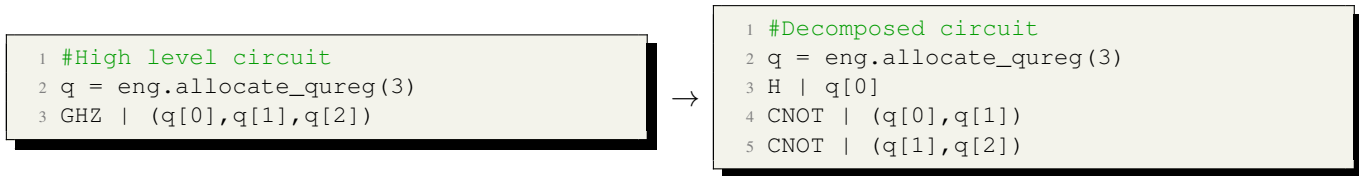


Figure 5.4: Quantum code decomposition of the GHZ function. The GHZ state (Left) is common enough to be defined as a single command in a quantum code (Center). A decomposition layer should translate this high level command into single and two qubit gates (Right) for facilitating hardware implementation.

This decomposition is classic and can be found in almost all the quantum frameworks previously mentioned.

5.3 Quantum circuit optimization

Layer 2 from Figure 5.3 is at the quantum circuit level and focuses first into reducing the number of gates applied in the circuit while keeping the total evolution operator equivalent. Simultaneously, it adapts the arbitrary gate set to one which can be physically implemented in hardware. The ProjectQ framework introduced above is able to do some optimization but independent quantum software developed in the physics community are also capable of such tasks, some being compatible with existing frameworks [131]. Exotic compilation methods, such as using neural network circuit generation [132] or circuit compilation assisted by quantum computers [133] are not considered here as they require too much computing power for the small increased gain in compilation efficiency. Some tools are architecture specific, such as the crosstalk optimizer for superconducting architectures [134]. Gate optimization is particularly useful for NISQ devices where quantum resources are

sparse and noisy. Running the same algorithm with fewer gates have considerable impact on the result quality: the run time is faster, thus reducing the impact of decoherence and by applying less gates the total error rate of the evolution operator will be reduced.

5.3.1 Gate reduction

A simple optimizing protocol in ProjectQ as displayed in Figure 5.5 is to cancel self inverse gates if they are next to each other, and merge consecutive rotation gates of the same rotation axis:

```
1 #Cancelling example 1
2 H | q[i]
3 H | q[i]
4 #Cancelling example 2
5 Rx(theta) | q[j]
6 Rx(-theta) | q[j]
```

```
1 #Merging example
2 Rx(theta) | q[i]
3 Rx(gamma) | q[i]
4 #the two above gates will
5 #merge so that they can cancel
6 #the gate below
7 Rx(-(theta+gamma)) | q[i]
```

Figure 5.5: Cancellation (Left) and merging (Right) identities. Cancelling happens when consecutive gates are equivalent to the identity gate, while merging proceeds when consecutive rotation gates of the same type are identified.

5.3.2 Gate decomposition

The gate replacer tool is a fundamental compilation operation adapting a circuit from a gate-set to another. Adapting circuits is required due to the limited gate-set available in QEC codes, and in the physical architecture of quantum computers. In the case of trapped ion quantum computers, it was already mentioned in Chapter 2 that only three gate types were physically implementable: the Rotation X gate $R_x(\theta)$, the Rotation Y gate $R_y(\theta)$ and the Mølmer-Sørensen $R_{xx}(\theta)$ gate. Arbitrary quantum circuits can be translated and optimized on this gate subset [135]. In the architecture considered at the Ion Quantum Technology research group, only Mølmer-Sørensen interactions over two qubits are used but it is possible to extend this entangling gate up to any number of ions, reducing the gate count of most circuits such as the surface code stabilizers [116].

Let us consider two examples: The CNOT gate can be decomposed in several different circuits as represented in Figure 5.6, with additional Rotation gates. The decomposition choice is represented by freedom parameters s and v being freely chosen on $\{+1, -1\}$. Similarly the Hadamard gate can be decomposed into two possible circuits as shown in Figure 5.7.

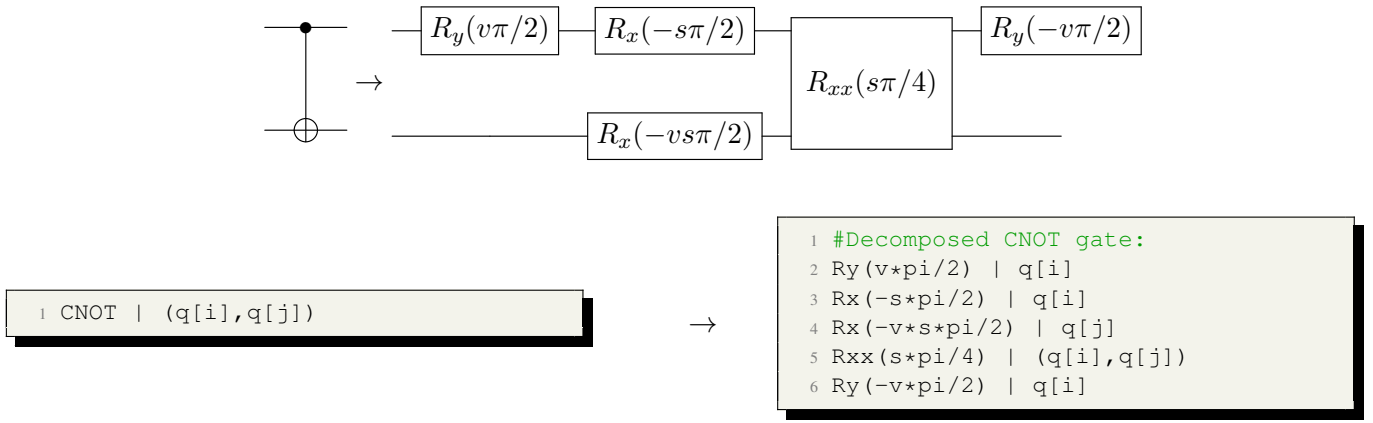


Figure 5.6: CNOT gate decomposition with the ion trap gateset with the quantum circuit representation (Top) and the quantum code representation (Bottom).

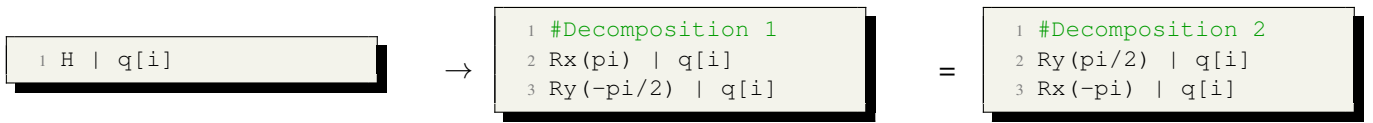


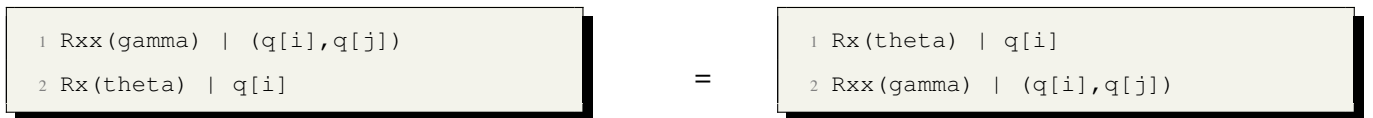
Figure 5.7: H gate decomposition with the ion trap gateset. The choice among the possible decomposition should be made to merge and/or cancel more gates.

The different possible combinations of gate decomposition can be used to cancel successive rotation gates to reduce the gate count.

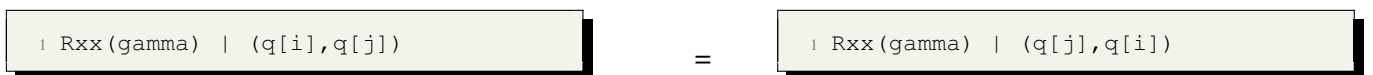
5.3.3 Gate commutation

The commutation optimizer [136] is a tool which permits additional merging and cancelling operations by exploiting commutation rules between some quantum gates. The trapped ion compatible gate set is interesting in that regard because:

1) the R_x gate commutes with the R_{xx} gate:



2) the R_{xx} gate is symmetric:



3) the R_{xx} gate commutes with itself:

```

1 Rxx(theta) | (q[i],q[j])
2 Rxx(gamma) | (q[i],q[k])

```

=

```

1 Rxx(gamma) | (q[i],q[k])
2 Rxx(theta) | (q[i],q[j])

```

Those three commutation rules allow a larger set of cases for which optimization is possible. The software implementation in ProjectQ was written by Daisy Smith¹⁰ under my supervision. It was written to also cover superconducting circuits using Rz and CNOT gates.

The three compilation tools can be tested as an example on the GHZ gate toy model described earlier. It produces the following code for trapped ion quantum computing architecture:

```

1 #High level code
2 GHZ | (q[i],q[j],q[k])

```

→

```

1 #Generic 1 & 2 qubit
   gates
2 H | q[i]
3 CNOT | (q[i],q[j])
4 CNOT | (q[i],q[k])

```

→

```

1 #Circuit optimization
2 Ry(pi/2) | q[i]
3 Rx(3*pi) | q[i]
4 Ry(pi/2) | q[i]
5 Rx(3*pi) | q[i]
6 Rx(7*pi/2) | q[j]
7 Rxx(pi/4) | (q[i],q[j])
8 Ry(7*pi/2) | q[k]
9 Rxx(pi/4) | (q[i],q[k])
10 Ry(7*pi/2) | q[i]

```

This output is not yet optimal. In particular the qubit $q[i]$ has four consecutive single qubit gates applied to it. As any unitary $SU(2)$ can be decomposed into at most 3 rotation gates, a further compilation tool that could be implemented would identify large chain of single qubit gates and replace them by a 3 single qubit gate sequence.

5.4 Ion router

Most frameworks detailed in 5.1.2 were initially conceived with superconducting architectures in mind, where qubits are fixed and connectivity is limited to nearest-neighbor maps. Artificial global connectivity is achieved by swapping quantum states between neighbor qubits via sequences of three CNOT gates, as already mentioned in Section 1.1.4. Such overhead is obviously detrimental to the algorithm fidelity. Mapping computational qubits into physical qubits by minimizing the swapping operations is still an active subject of research [137, 131, 138]. The quantum code reaching the backend layer of the chosen framework is a quantum circuit code with the additional CNOT gates due to swapping operations coupled to a map between computational and

¹⁰<https://github.com/ProjectQ-Framework/ProjectQ/pull/386>

physical qubits, to assign the measurement to the appropriate qubit.

Mapping computational qubits to physical qubits is usually one of the final layers of compilation for most known frameworks described above. The final layer, the backend, sends the quantum code to a server online controlling an existing quantum computer. Most available quantum hardware have their own Application Programming Interface (API) protocol to which they communicate data, so quantum frameworks usually contain different backends depending on the targeted hardware to run the quantum circuit. Most quantum hardware APIs only accept quantum code corresponding to layer 2 from Figure 5.3, running all the more hardware-related compilation internally. Most APIs are therefore relatively easy to understand as it stays at the quantum circuit level.

5.4.1 Existing ion trap QPU provider

The only public API for trapped ion quantum computers is AQT. Their device has global qubit connectivity and does not mention ion shuttling¹¹. Their architecture does not involve shuttling operations as they consist of static ion chains in which selected entangling gates are achieved by sending laser beams for each individual ion [139].

The company IonQ, similarly to AQT, has a quantum hardware consisting only of ion chains¹². Honeywell is the only ion trap based QPU provider which employs shuttling protocols, but their API restricts the access to quantum circuits only, staying in layer 2 of compilation¹³. It indicates that they compile their routing sequence internally.

5.4.2 The QCCD syntax

All of the above, representing the state of the art of existing quantum computers, is unfortunately not applicable to the research group because our trap designs require that the ions, representing the qubits, can only be interacted for specific operations in specific zones. Global connectivity is effectively possible however it comes at the cost of implementing a series of shuttling operations described in Chapter 2 that themselves induce noise and time latency [140].

The following section outlines the ProjectQ backend implemented to describe the circuit associated with shuttling instructions. It will require a new formalism bringing the quantum circuit code outside of the ProjectQ

¹¹<https://www.aqt.eu/aqt-json-tutorial/#json-syntax>

¹²<https://ionq.com/technology>

¹³<https://www.honeywell.com/en-us/company/quantum/quantum-computer>

framework which will be used as a gateway for hardware implementation.

This shuttling approach is fundamentally different from the mapping procedure between computational qubits and static physical qubits from superconducting architectures. Unfortunately most if not all quantum code frameworks, including ProjectQ, were initially designed for these superconducting architectures. Efficient code for QCCD devices will require an extension of the available set of commands by including shuttling related operations. Such hybrid quantum circuits with shuttling syntax has already been considered in the past. They are however mostly unbalanced, between syntax focused on quantum operations [141] and syntax focusing on shuttling operations [142]. The most recent approach mixes shuttling, quantum operations and even sympathetic cooling via ancillary species [111], although the syntax of quantum operations are still written with the index of the qubit register. Indeed using the QCCD toolbox mixes in the same syntax two opposite concepts:

- gate commands that are applied to qubits, which need to be subsequently identified as physical ions in a gate zone
- shuttling commands, although mentioning qubit number, mainly concerns ion positions on the trap

This confusion between the computational qubits and the physical ion position hosting the qubit is amplified by the fact that most operations in QCCD architectures, such as preparation and gate pulses, are physical processes targeting a physical position. The action of sending a laser or microwave field is independent of the qubit number in the quantum circuit but entirely depends on the ion position representing the qubit.

Position based commands

We now define a new syntax for representing a sequence of gates and shuttling operations applied to a trapped ion quantum processor, called “QCCD code”. It reuses the ProjectQ syntax with the “|” separator but will be applied to physical positions on the trap instead of qubit numbers. This syntax allows a direct representation of the physical tasks to be run by the hardware through the entire quantum circuit, covering both quantum logic and ion transport operations. Removing qubit indexes which is not directly understandable nor controllable by the hardware enables the QCCD code to be the first pure Quantum Assembly Language for QCCD architectures which can be run as such by the hardware via one-to-one equivalence between commands and physical pulses, without further compilation.

For clarity we will define a trapping zone as a section on the microtrap able to hold and manipulate an ion

crystal. This definition is common and has been used by all the other QCCD syntax mentioned above. Zones are connected between them by a routing map hosted by the quantum framework. Zones are attributed parameters, such as the maximum number of ions in a chain that can be held in the zone, as well as a list of quantum operations that can be applied in the zone. An ion can be identified on the trap using two pieces of information: The name of the zone in which the ion crystal is placed, and its position in the ion crystal. To distinguish the first from the last position in a crystal, a reference point has to be defined by the framework.

Static commands

Quantum gates are executed as a resonant RF/MW pulse in a magnetic field gradient. Therefore correct addressing depends on the exact position of the ions involved:

$$\{Gate\} \mid \{Position\ on\ trap\}[\{ion\ position\}/\{crystal\ size\}]$$

Quantum logic operations involving laser beams, such as Preparation and Measurement, cover an entire trapping zone and therefore cover every ion in the ion crystal of the trapping zone:

$$\begin{aligned} &Prepare \mid \{Position\ on\ trap\} \\ &Measure \mid \{Position\ on\ trap\} \{Qubits\ ids\ being\ measured\} \end{aligned}$$

To trap new ions or index them in the quantum framework, we use the Init function:

$$Init \mid \{Position\ on\ trap\}$$

A single Init function is used with a list of arm names representing the ordered list of qubits being used. If a zone is mentioned multiple times ions are piled up in the same crystal.

For simplicity we omit sympathetic cooling, but later iteration of the compilation layer could simply trigger this operation before entangling gates: either in forcing it before each two qubit gates, or by assessing the heat level induced by all previous gates and shuttling operations. In both cases new commands need to be created for handling shuttling and state preparation of the second type of ions.

The map between computational qubits from the quantum framework with the physical positions used in the QCCD code is implemented with the Init and Measurement commands: In the Preparation command the qubit index is registered in the quantum framework but does not need to be mentioned in the QCCD code itself. However the Measurement operation directly links the zone to be measured with the classical bits in which

the qubit measurements will be stored. Those readout bits are stored by the QPU and returned to the quantum framework which is aware of the map between the classical bits and the computational qubits.

Reconfiguring commands

Shuttling protocols are assumed to integrate splitting and merging operations. This integration is a direct consequence of the ratio of ions that can be trapped compared to the number of trapping zones available in a trap, which is always greater than 1. However splitting and merging operations can only be implemented successfully if there is a knowledge of the number of ions in the crystal. As illustrated in Figure 5.8, every shuttling command involves the ion position in the departing arm and the crystal length in both the departing and targeted arm:

$$Shuttle \mid \{Leaving\ arm\}[\{ion\ position\}/\{crystal\ size\}]-\{Arriving\ arm\}[\{crystal\ size\ before\ arriving\}]$$

We want to distinguish the swap operation from the splitting and merging operations which rotates the ion crystal in the arm. This separation is intended to allow the routing algorithm to use or not use this feature if it is physically implemented:

$$Swap \mid \{Position\ on\ trap\}$$

5.4.3 Routing in a X-junction

We have now defined the QCCD syntax and can now use it to implement logic based quantum circuits, where 1) the number of ions that is allowed in an arm is far less than the total number of ions in the trap, and 2) the different operations required for quantum computation ,such as preparation, measurement and gates, are not all available in the same arms. We will use for example a single X-junction incorporating a single gate zone as displayed in Figure 5.8. We will assume that the maximum number of ions per arm is two, as it relaxes the complexity on implementing splitting and merging protocols. Two arms can apply readout, since they are on the same 369 nm laser beam path, but only one arm can apply preparation and gates. We will also assume that all arms can split, merge and swap crystals. Finally we will restrict quantum circuits to initialize all qubits at the beginning as well as only doing measurements at the end of the circuit. This last constraint will considerably simplify the algorithm and initial physical implementation, but we are aiming to remove it in the near future.

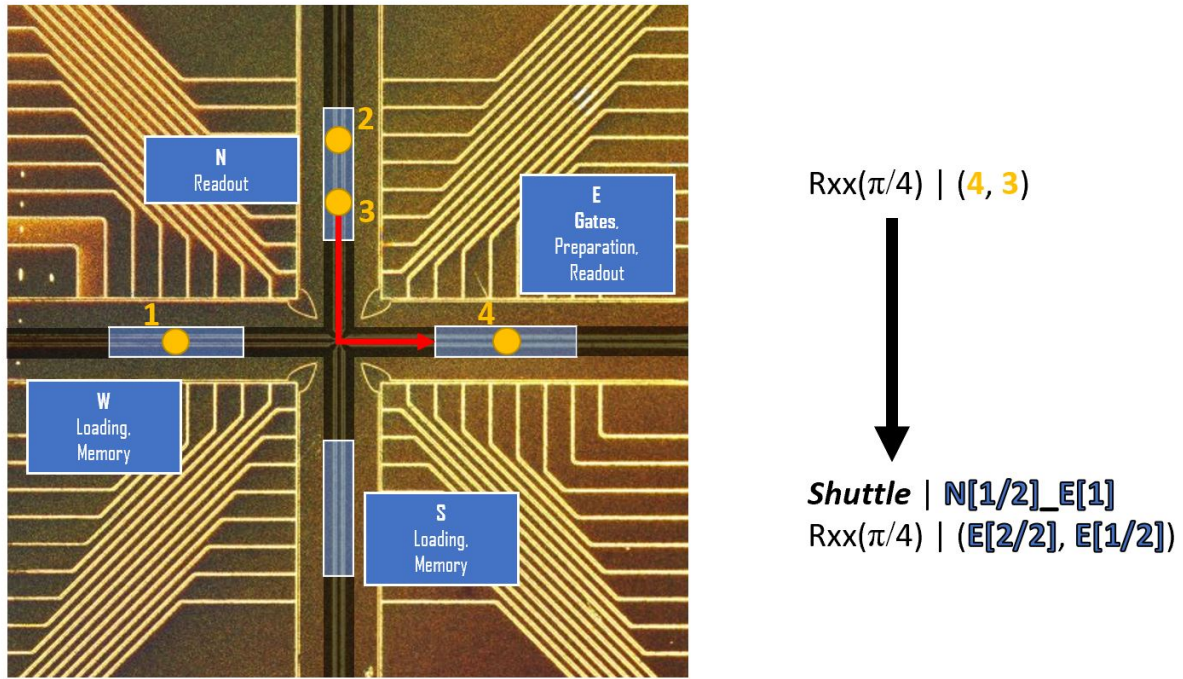


Figure 5.8: QCCD code illustration of a shuttling sequence, implementing an entangling gate in a X-junction between two distant ions. (Orange) The ions are the physical qubits, and each qubit is identified by its own ID number in the ProjectQ syntax. (Blue) Zones in which ions can be trapped and interacted with. Each zone is associated with a name (N, E, W, S) for the QCCD syntax, and a list of available controls to interact with the ions inside of each zone. (Top right) An entangling gate command, in the ProjectQ syntax, between two ions in two different zones, is decomposed into a QCCD code (Bottom right). The QCCD code contains all the required information to shuttle one ion in the zone containing the second one. The red arrow represents the ion trajectory resulting from the shuttling command.

We will now describe the algorithm involved when a qubit is required for a single qubit gate but is not located in a gate zone. Firstly, the compiler assesses the gate zone to see if there is a place for the incoming ions. If the zone is full the compiler will preemptively shuttle an ion to an available memory zone. Shuttling of the targeted ion can then proceed.

All commands from the quantum circuit involving shuttling are extended from this basic algorithm, although both preparation and measurement, named here P/M for simplification, presents an additional challenge unknown to superconducting architectures. Since there are more ions than the number of places available in the zone capable of applying the P/M command, a single “P/M — All” will involve multiple cycles of shuttling and P/M applications. Lets consider a simple numerical example with the X-junction: 5 ions in this trap containing only one preparation zone hosting at most 2 ions will have to apply three cycles of shuttling and preparations to initialize all qubits. The GHZ gate example is illustrated on the X-junction with the compiler providing the following QCCD code:

```

1 #Generic quantum
  circuit
2 q = eng.
  allocate_quireg(3)
3 H | q[i]
4 CNOT | (q[i],q[j])
5 CNOT | (q[i],q[k])

```

```

1 #Native quantum circuit
2 q = eng.allocate_quireg(3)
3 Ry(pi/2) | q[i]
4 Rx(3*pi) | q[i]
5 Ry(pi/2) | q[i]
6 Rx(3*pi) | q[i]
7 Rx(7*pi/2) | q[j]
8 Rxx(pi/4) | (q[i],q[j])
9 Ry(7*pi/2) | q[k]
10 Rxx(pi/4) | (q[i],q[k])
11 Ry(7*pi/2) | q[i]

```

```

1 #QCCD code
2 Init | N N E
3 Shuttle | E[1/1]_W[0]
4 Shuttle | N[1/2]_E[0]
5 Shuttle | N[1/1]_E[1]
6 Prepare | E[2]
7 Shuttle | E[1/2]_S[0]
8 Shuttle | W[1/1]_E[1]
9 Prepare | E[2]
10 Ry(pi/2) | E[2/2]
11 Rx(3*pi) | E[2/2]
12 Ry(pi/2) | E[2/2]
13 Rx(3*pi) | E[2/2]
14 Shuttle | E[1/2]_W[0]
15 Shuttle | S[1/1]_E[1]
16 Rx(7*pi/2) | E[1/2]
17 Rxx(pi/2) | ( E[2/2], E[1/2])
18 Shuttle | E[1/2]_S[0]
19 Shuttle | W[1/1]_E[1]
20 Rx(7*pi/2) | E[1/2]
21 Rxx(pi/2) | ( E[2/2], E[1/2])
22 Ry(7*pi/2) | E[2/2]
23 Shuttle | S[1/1]_N[0]
24 Measure | N 1
25 Measure | E 2 0

```

The noticeable increase in the number of commands, even for simple circuits, comes from the constraints in place on the chip, forcing multiple shuttling and swap operations to appropriately send ions to their desired place. This observation has already been made for Y-junctions [108] when comparing different implementations of logical qubit gates. Shuttling through the junction does not have the same time duration and noise model than splitting, swapping or simply shuttling across a zone, the compiler needs to be adapted to those physical characteristics and minimize the amount of noisy shuttling operations when possible.

In particular for zones capable of hosting large ion chains the impact of topology has an impact on the fidelity. Swapping two specific qubits located in a $n > 2$ ion crystal, equivalent to a ion crystal reordering, involves splitting, merging and ion crystal rotation operations, adding more errors for ions located further apart. For the unusual case of ion chains above 10 qubits and with a noise model including state-of-the-art shuttling protocols, swapping operations are recommended to be implemented by quantum gates [143], similar to superconducting architectures with three inverted CNOT gates rather than by position control.

5.4.4 Visualizing the QCCD code

Further improvements and debugging require a clear visualization tool to read the QCCD into a dynamic display of shuttled ions in a chip. This visualization tool was developed using the python module PyQtGraph and an example of a stabilizer measurement with five qubits on the chip is displayed in Figure 5.9, the module choice allowing a future integration inside the ARTIQ framework.

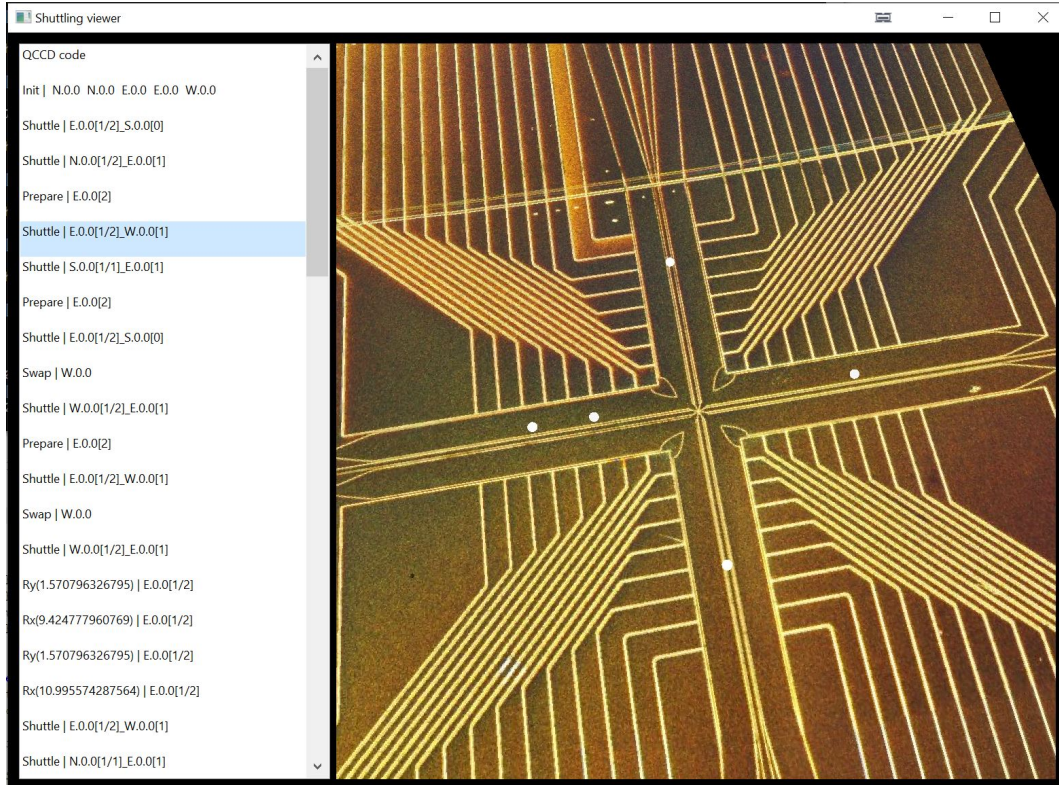


Figure 5.9: QCCD code visualization software in a X-junction: QCCD instructions (Left) are dynamically read and represented in the 3-D model (Right)

5.4.5 Routing problems

Running shuttling based quantum circuits raises problems on the initial and final ion positions when running a quantum circuit:

- The initial position assignment will probably not correspond to the current positions in the trap.
- Most circuits will be run multiple times, forcing the ions to return to their initial positions.
- Ions need to be trapped again to match the required number for the algorithm.

The position mismatch between what is required and the physical position will be resolved by applying a shuttling sequence that will be named “pre-shuttling” and “post-shuttling”, for bringing the ion in position before running the circuit and bringing the ion back to the required position between two runs. While the logic and syntax is the same as for compiling the quantum circuit itself, the instructions are handled and stored separately by the compiler.

5.4.6 Future work

The QCCD syntax was elaborated with the aim of compiling small circuits in a very confined space where a sequential execution is the only possibility. Very few tasks could be run in parallel and would not have much impact on the ability to run the circuit. However by considering traps containing more than one junction substantial speed gain could be made if the multiple gate zones are used at the same time.

Furthermore the routing algorithm to translate quantum circuits into QCCD code may be inefficient as ion positions are only considered for one command at a time: ions that are used more often than others could be more appropriately handled by storing them in or near the gate zone preferably to other ions.

Finally the QCCD syntax should handle logical “if” statements so that corrections of quantum errors and ion loss could be dynamically implemented in real-time. The most direct approach is to extend the ProjectQ framework to conditional logic, and later adapt the QCCD syntax with the new ProjectQ commands.

5.5 Hardware instructions

The QCCD syntax described above can be considered as a minimal representation of the hardware instructions having to be run on our shuttling based trapped ion architecture. However, while the overarching logical steps of a quantum circuit have been well defined, there is still a gap in mapping these onto actual hardware. Converting a circuit pulse to a hardware pulse requires encoding RF and MW waveforms, the time control of laser beams and programmable TTL signals. Every individual command of the QCCD syntax needs to be decomposed into an exact series of hardware pulses that physically implement the command.

Describing a quantum circuit at the hardware level as a pulse sequence allows for modularity. The compilation of the pulse sequence is designed such that the resulting instructions are hardware agnostic. Therefore, a given pulse sequence syntax can be employed across many different experiments that contain different hardware. This syntax is intended for use in the research group but is reproducible enough to be adopted by any quantum computing teams.

It is important to specify for clarity that the hardware syntaxes presented in this section are not the lowest hardware layer that operate a quantum computer. The waveforms shuttling ions and/or driving the gates are, fundamentally, the lowest hardware representation. The role of the presented syntax is to gather all the required information so that the AWG, the FPGA and the DACs can generate the exact waveforms needed to run the requested quantum circuit and at the right time. The AWG to generate the quantum control waveforms, the DAC to generate the ion shuttling waveforms and the FPGA for triggering and synchronizing.

5.5.1 The OpenPulse syntax

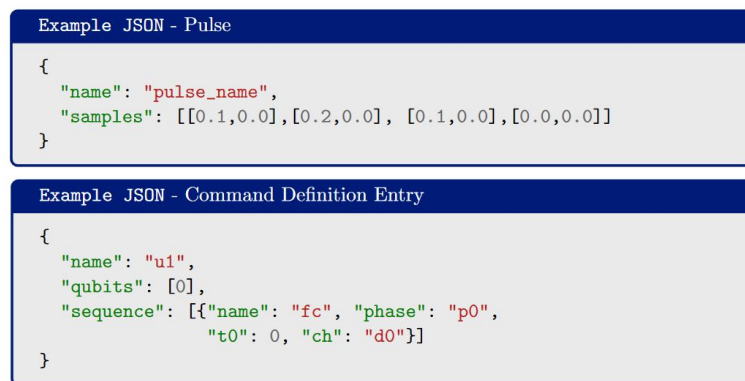


Figure 5.10: Pulse and Command syntax in OpenPulse. (Top) The pulse syntax has its amplitude, frequency and phase information encoded in the list of complex numbers representing each point of the waveform. (Bottom) The command syntax consists in building the sequence of pulses via the “sequence” keyword, and indicates which qubit is affected via the “qubits” keyword. Each pulse from the sequence provides additional information such as (“ch”) which indicates which hardware will apply the waveform and (“t0”) for when the waveform is applied. The particular pulse “fc” represented in this command shifts the phase of all subsequent waveforms by the amount “p0”, which is a practical approach for superconducting devices to implement R_z gates. Phase shifts are not practical for qubits driven by global microwave as every qubit would be affected by the shift. Figures taken from [144].

While much less common than the quantum circuit representation, pulse sequence syntaxes are gaining more attention recently for their ability to switch the hardware implementation of gates and use noise resilient schemes. The most famous pulse syntax is OpenPulse [144] from IBM and is integrated with OpenQASM in Qiskit [124]. The OpenPulse specification is an extension of the existing OpenQASM syntax which formulates quantum circuits with JSON objects. This extension, as illustrated in Figure 5.10, gives a more direct access to the hardware: there is the ability to define new gates by assembling JSON objects representing pulses. Each pulse object contains frequency, phase and time information that can be directly produced by a waveform generator. Furthermore, gates can be compiled by choosing a sequence from a predefined pulse library. The robust gates considered in Chapter 3 can be constructed from such pulses using sequences of amplitudes, frequencies

and phases. The OpenPulse syntax is flexible enough to accommodate different QPU architectures including trapped ions, although shuttling is not explicitly mentioned. Improvements in circuit fidelity exploiting the pulse sequence syntax have already been shown [145, 146].

5.5.2 The PulseSequence syntax

The syntax used in the research group, simply called PulseSequence, is a syntax designed by Dr Simon Webster and Dr Adam Lawrence. It is also a JSON object but contrary to OpenPulse, is not extended from a quantum circuit JSON representation. It is produced by a python program accepting either quantum circuit instructions or calibrations mentioned in Section 3.3, and I later adapted it to read QCCD code. As illustrated in Figure 5.11, The PulseSequence consists therefore of a frontend to configure the sequence of pulses to apply, and a backend which is the compiled representation of these pulses. The compiled instructions are split in sections to be sent to the FPGA, the AWG and the DAC's drivers.

On the frontend, each physical action is represented via pulses. Triggering TTL channels are added with the `addPulse()` function, MW and RF signals are generated with a `addRFShapedPulse()` function, and Shuttling operations are implemented with the `addDCPulse()` function. Additional information for running the experiment, automatic logging and data analysis are added with the `addParam()` function. Any sequence is initialized with a `initPulseList()` function and once the sequence is completed, a `makeSequence()` function is called to return a compiled version, the backend. The functions are built using the following parameters:

```
1 initPulseList()
2 addPulse(name='name', time=0, step=0, count = 0, opts = {})
3 addRFShapedPulse(name='name', time=0, step=0, timeps=0, IQ=0, channels=[NULL_FCHAN])
4 addDCPulse(name='name', params='')
5 makeSequence()
```

with “name” a label for debugging, “time” the pulse duration, “step” the time step for implementing a scan, the non-zero “count” is the index of the memory register in which a light measurement is recorded, `opts` is a dictionary containing TTL and analog output channels to use. “timeps” indicates the time to ramp up and down the field, “IQ” indicates if the pulse is a MW or RF pulse. “channels” is a list of channel JSON objects, each channel representing a single AC field, with additional stepping parameters for the frequency, the amplitude and the phase:

```
1 channel('freq'=0, 'freq step'=0, 'amp'=0, 'amp step'=0, 'phase'=0, 'phase step'=0, 'channel
   inv'=True)
```

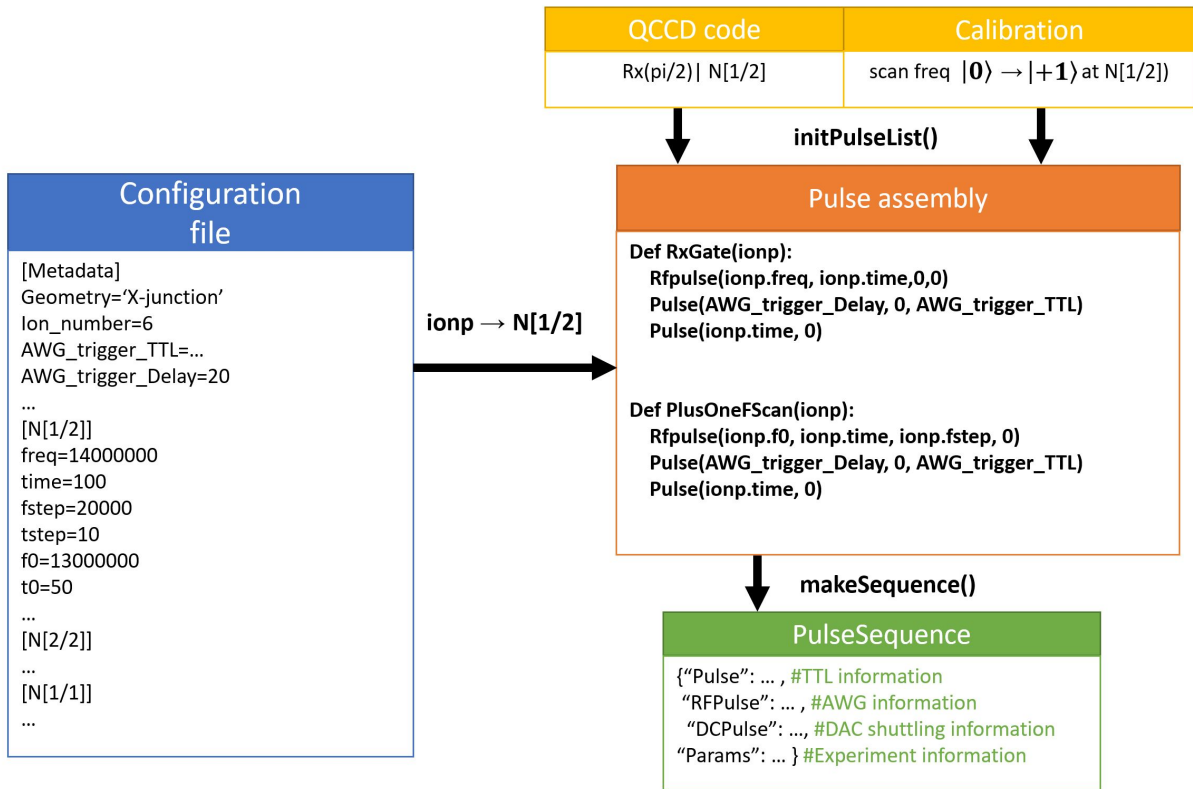


Figure 5.11: Hardware compilation of quantum code into a Pulse sequence. (Yellow) The code is flexible enough to accept both QCCD code and calibration experiments. (Blue) Due to the QCCD architecture, calibration of pulses are associated to the ion position in the gate zone (e.g. N[1/2], the first ion in a two ion crystals in the gate zone N) so the configuration file, indicating which physical values the pulse has to be built with, is categorized by the ion position. (Orange) The compiler associates the gate definition to a series of pulses, replacing the time/frequency/amplitude/phase information from the configuration file. (Green) The output is a JSON code containing all the physical information required to run the circuit/scan. Information aimed at specific hardware (e.g. RFPulse for the AWG) is gathered together and transmitted to that device.

here “channel inv” is a keyword to help reduce the time cost associated to build the pulse waveform for the AWG. A single RFPulse can generate any desired fields at the same time, by increasing the number of channels being used, which is required for state dressing considered in Section 2.3.3 and the Mølmer-Sørensen gate scheme detailed in Section 2.2.7.

The RFPulse defined above, “shapedPulse”, is one option to generate a RF or MW waveform and involves slowly ramping the signal at the start and at the end of the pulse. More waveform shapes are available, such as the “rfMappingPulse” which has a frequency oscillating in time, required for matching the oscillating AC-Stark shift of the polychromatic gate detailed in Section 3.2.3, or “blackmanPulse” to implement high-fidelity quantum state transfer as mentioned in Section 2.3.3.

Both RFPulse and DCPulse are always accompanied by a Pulse for triggering, with the duration equal to the AWG/DAC delay time if there is any, followed by an empty pulse of duration equal to the RFPulse/DCPulse

duration, to keep every devices synchronized. For example:

```
1 addRFShapedPulse('Pi pulse on ' + state1.name, state1.time, 0, 0, MWOUT, state1.fchan())
2 addPulse('AWG trig delay', AWG_TRIG_DELAY, 0, opts = OPTS_AWG_TRIG)
3 addPulse('Pi pulse ion 1', state1.time, 0, opts = OPTS_AWG_MW_PULSE)
```

where “state1”, obtained from the configuration file, indicates the relevant name, time and channels informations. The trigger delay time is indicated by “AWG_TRIG_DELAY”, and the TTL map associated with triggering the AWG is represented by “OPTS_AWG_TRIG”. “OPTS_AWG_MW_PULSE” is the idle TTL map while the AWG is running.

All these pulses can be called by higher level functions, such as state preparation, readout, quantum gate and calibration scans, enabling an easy access for the experimental user to control the calibration process, and an easy translation of QCCD circuit code into a PulseSequence object. A very common function for calibration is the frequency scan function `freqscan()`:

```
1 def freqscan(state = "", det = 0, step = 0, steps = 0, runs = 0) :
2     state = eval(state)
3     initPulseList()
4     addPrepPulse()
5     addRFShapedPulse('Freq scan ' + state.name, state.time, 0, 0, MWOUT, state.fchan(freqdet
6         = det, freqstep = step))
7     addPulse('AWG trig delay', AWG_TRIG_DELAY, 0, opts = OPTS_AWG_TRIG)
8     addPulse('Freq scan ' + state.name, state.time, 0, opts = OPTS_AWG_MW_PULSE)
9     addReadoutPulse(DET_COUNT_1)
10    addParam("steps", steps)
11    addParam("runs", runs)
12    return makeSequence()
```

with “det” the initial frequency detuning from the current calibrated value, “step” the frequency value to add to the field at each step of the scan, “steps” the maximum number of steps to apply, runs is the number of time the same step is repeated to accumulate good enough quantum state readout statistics. This function also calls another high-level function `addPrepPulse()`, which is a sequence of RFPulses and Pulses to prepare the ion in the $|0\rangle$ state. The “steps” and “runs” parameters are outside the scope of the PulseSequence, and are added via `addParam()` to be accessed directly by the software supervising the experiment.

An important feature is the configuration file which is assigned to each experiment, updated after each calibration and which defines all the physical values needed to run any commands in a quantum circuit, such as measured transition frequencies or measured Rabi frequencies. It is an easier but less organized approach than OpenPulse which uses several JSON configuration objects for each gates and qubits. As displayed in Figure

5.11, the configuration file contains meta-data about the experiment architecture for local compilation of shuttling sequences detailed in Section 5.4, but also contains the information necessary to address any ion within the crystal. That information requires continuous calibration as was discussed in Chapter 3. Each call to a gate or a calibration simply needs to mention which ion position is addressed to access the relevant physical parameters.

When the `makeSequence()` function is called, the `PulseSequence` generates a JSON object containing three main components, built from each of the `Pulses`, `RFPulses` and `DCPulses`:

- **TTL information:** List of JSON objects aimed at the FPGA controlling the experiment. Each object of the list indicates when and which TTL channel is to be triggered, or a time delay associated with the use of another device. Triggering is an essential part of a QPU as preparation, pulses and readout have to be scheduled in time as precisely as possible. Most experimental devices, such as the camera or the waveform generator, are triggered by the FPGA. A simple example is given for state preparation:

```
1 { "string": "PREP", "time": 500.0, "step": 0, "dio": 131075, "count": 1, "AO": 8 }
```

In addition of the previously mentioned parameters of the Pulse, “dio” represents the binary list of TTL that needs to be off or on to physically activate electronic switches or trigger external devices and “AO” is the analog output control from the FPGA, mainly dedicated to controlling the AOM settings.

- **Pulse information:** String representing a list of waveforms to be generated by the waveform generator. Generating the waveform from the pulse information is a slow process that has to happen before the start of the experiment. The framework to use the AWG with the OpenPulse syntax was developed by Dr Adam Lawrence [86], and consists, in the apparatus described in Section 3.2.2, in forming the String as a LabVIEW variant, transmitting the variant to a second computer directly connected to the AWG, generating the waveform in that second computer from the variant and with the assistance of a GPU, and finally uploading the waveform to the AWG. By generating the waveform in the second computer, only a small amount of information is transmitted over the network, compared to OpenPulse encoding the entire waveform in the same object. The AWG framework can be easily adapted to any other architecture by replacing the LabVIEW variant communication to a language agnostic communication based on json objects.
- **Shuttling information:** Shuttling ions across a surface microtrap is a process sharing similar difficulties with the AWG: multi-channel DC waveforms have to be generated to manipulate the ions, and then need

to be triggered later on at specific times. The information is similar to the QCCD syntax but only includes ion displacement sequences and not any quantum state manipulation.

The apparatus from the research group uses OpenPulse due to its simplicity: firstly it was elaborated years before the IBM specification and all the hardware needed to drive quantum gates with trapped ions in the laboratory was configured in it. For more details, a full presentation of the software and hardware organization and compiling a PulseSequence into a waveform generated by the AWG, is described in [86]. Secondly, while being less exhaustive in terms of controlled parameters, the PulseSequence was originally conceived for driving scans and was only later adapted by Christophe Valahu to run quantum circuits. OpenPulse needs to define new pulse objects for each step of a scan, requiring for example, for a frequency scan, to define a new pulse for each step. However, a PulseSequence object contains step information of amplitude, frequency and phase for each pulse signal. A single PulseSequence object can implement most scans, making this syntax efficient for calibrating the device as detailed in Chapter 3.

Such a syntax is therefore adaptable to any QCCD trap geometry and usable with any hardware, while simplifying calibration scans by sharing the same syntax used for quantum circuits.

5.5.3 Future work

The PulseSequence syntax, as well as the QCCD syntax, first require a white paper or document formalizing their capabilities. They will then require the ability to implement parallel instructions and conditional logic. Parallelism in the PulseSequence which is so close to the hardware presents the additional challenge that two simultaneous instructions may each take a different time to complete. Parallelism may also impact quantum operations. For shuttling instructions, a DAC may share the same update bandwidth for multiple electrodes. For gates, using too many tones can reach the power limit of the amplifier. Such situations can be mitigated in hardware by having better DACs and amplifiers, and in software by restricting the number of allowed parallel instructions for each type of command, in the backend engine of ProjectQ.

The conditional logic functionality should be added by replicating or taking inspiration from the OpenPulse syntax. It first requires the ability to read the quantum state in real-time, and OpenPulse already standardizes the measurement result as a JSON object, its encoding in a memory register and its reuse for subsequent gates. Conditional pulses are written by adding a keyword “conditional” on the qubit register of interest, and is then applied only if that register was measured to be 1. By extending this keyword to commands, entire circuits can be easily triggered by a single measurement.

As mentioned in Section 5.5.2, the current PulseSequence is static because the waveform to be applied needs

to be generated first on time scales that are much larger than the run-time of a quantum circuit. Extending the hardware to support conditional logic consists of extending the waveform generation to all possible pulses that may be applied in the circuit. The choice of the waveform to apply in real-time will also require an additional triggering sequence, sent by the hardware which determined the qubit measurement, to the device. The shuttling compilation and its extension to conditional logic is commented in Chapter 8. Extending the AWG framework to enable continuous RF/MW signal generation with conditional logic has been partially solved for situations where phase control is not required, such as sideband cooling [86]: The AWG has an internal memory that can be addressed, in real-time, with a signal generated from the FPGA. The RFPulse will be replaced by a loading signal to load the requested waveform on the AWG, and a subsequent Pulse for triggering will still be required.

5.6 Compiling QEC codes

5.6.1 Toward large scale compilation

Compiling QEC codes in a quantum framework involves major changes from a NISQ based architecture. The first required feature which is appearing in recent frameworks such as Q# is qubit allocation. The hardware compiler needs to be aware of available ions to be used as ancilla or auxiliary qubits, as displayed in Figure 5.12, and be able to reuse them as another qubit later during the computation. It also involves the hardware capability of preparing and measuring selected qubits while not affecting the rest of the quantum register. This feature is required for the ancilla qubits, the auxiliary logical qubits, and to handle ion losses in the trap. This hardware compiler can be at first written, for static circuits, as a small extension of the backend compiling the QCCD code.

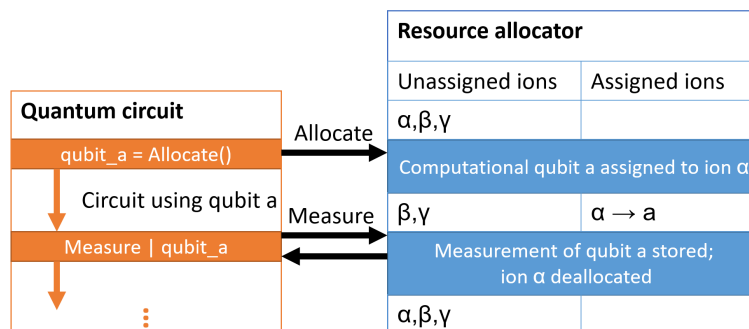


Figure 5.12: Example of allocation for a single qubit quantum circuit run on a three qubits quantum computer: the allocating command interacts with an allocation manager to decide which available qubit should be allocated for this circuit. Once the qubit is measured, the ion does not belong to the circuit anymore, joins back the list of available qubits and can be reused later in the quantum circuit.

A second feature is a mapping between logical qubits used for the algorithms and the physical qubits of the device. Instead of a one-to-one mapping of the computational qubits to the physical qubits, both the maps of the computational qubits to physical qubits and the computational circuit itself is translated into a new physical circuit, with data and ancilla qubits, in which logical gates and stabilizer measurements are encoded. The computational qubits have to adapt to the gate-set required in input for the QEC code, but the physical circuit still needs to be translated to hardware. An integration in ProjectQ could be achieved by assigning a MainEngine for each logical layer as represented in 5.13. Each of those would have its own circuit representation and its engine list adapted to compile into the next required gate set. The backend engines, hosting the encoding and decoding functions, will convert the computational qubits and circuits from one layer to a lower one based on the requested QEC code. The new circuit, and qubit register, is then sent to the new MainEngine. The final MainEngine will translate as before the circuit into the chosen hardware device. Once the circuit is run and the qubit measured, the measurements are returned to the upper backend which will decode the logical qubit and carry the logical measurement to a further upper layer if there is one. This approach minimizes the required amount of software development by taking advantage of the flexible nature of the engines objects, and would enable immediately QEC code concatenation via increasing the number of MainEngines.

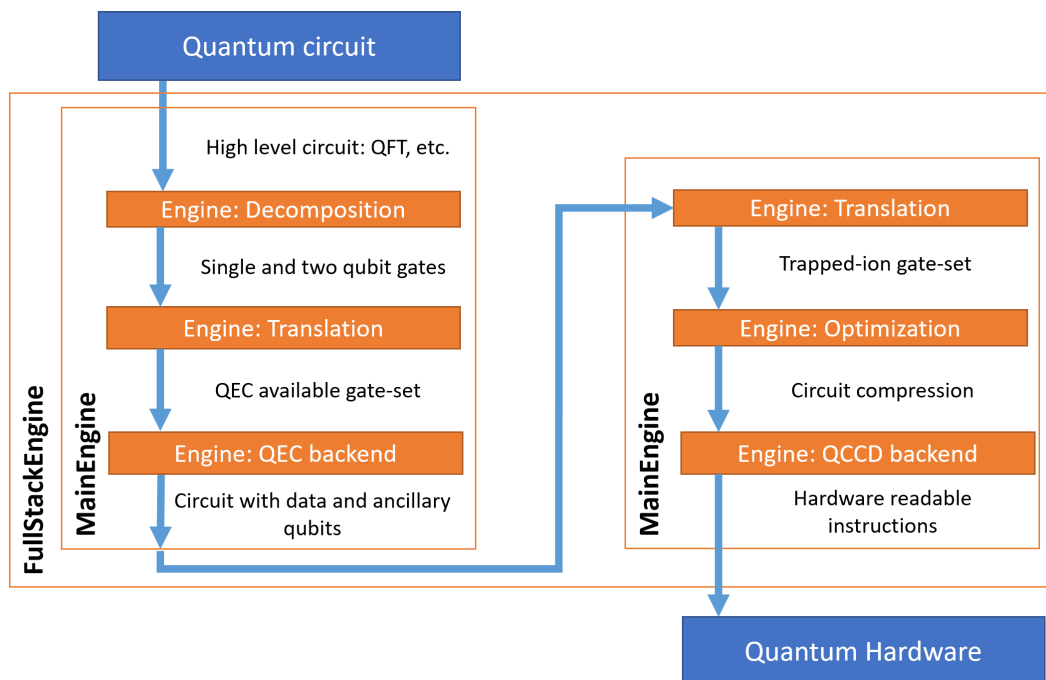


Figure 5.13: ProjectQ extension to compile QEC codes. The MainEngine structure is conserved but is extended to QEC compilation Backends. The new circuit generated by this MainEngine is sent to another MainEngine dedicated to compiling code for trapped ion devices. A FullStackEngine object would supervise the flow of quantum code between MainEngines similarly to the MainEngine supervising the flow between its Engines. This organization is adaptable to any level of concatenation of QEC as well as to any quantum hardware architecture.

Finally the allocation of logical resources, including the positioning of logical qubits and state distillation spaces, will require its own compilation tool similarly to the physical qubit allocation tool described above. Depending on the number of logical qubits involved and on the number of physical qubits available, the logical resource allocator would assign positions of logical qubits and auxiliary qubits such that the total logical circuit depth is minimized. Such strategies have already been formulated [147], and could be implemented in ProjectQ via dedicated engines.

5.6.2 Shuttling requirement for running a stabilizer

The compilation chain presented in this chapter can be used, as a short term application, to further define the hardware requirements in order to experimentally implement quantum error correction codes. Entangling gate fidelities, susceptible to the initial motional mode, will degrade in time due to both the natural heating rate of the chip, and also due to additional heating induced by each shuttling sequence. In polychromatic gates, $\bar{n}$ does not influence symmetric errors and is therefore only affecting the asymmetric detuning. A numerical simulation of the polychromatic gate fidelity indicates a resistance to most expected asymmetric detuning errors at $\bar{n} = 20$, remaining above 99% [2].

By detailing the QCCD syntax representing a single stabilizer measurement, time and heating constraints can be obtained for shuttling. We assume a maximum of two ions per zone, with only the East arm able to apply the prepare, readout and gates functions. The QCCD code is available in Appendix A.

The last and worst entangling gate has, when the fields start being applied, an initial $\bar{n}$ composed of the heating $t_{finalgate} \times \dot{\bar{n}}$ and of the additional heat from the shuttling sequences $\bar{n}_{shuttling}$. The time $t_{finalgate}$, between the initial preparation and the time at which the final entangling gate is applied, can be formulated as:

$$t_{finalgate} = 3t_g + 11t_{sq} + 7t_m + 16t_{split/merge} + 12t_{junction} + t_{swap} \quad (5.2)$$

with t_g the two qubit gate duration, t_{sq} the single qubit gate duration, t_m the mapping duration between dressed states and clock states to switch between quantum gates and shuttling sequences, $t_{split/merge}$ the splitting and merging sequence duration, t_{swap} the swapping sequence duration and $t_{junction}$ the duration to shuttle across the X-junction. By reusing the predicted t_g from Chapter 4, an average single qubit gate equal to the mapping time $t_{sq} = t_m = 21 \mu s$ [2], splitting and swapping duration approximated to a hundred microsecond, and the heating rate requirement from equation 4.12, a time limit can be imposed for shuttling across the X-junction, limited by the coherence time of the dressed state of 650 ms as mentioned in Section 2.3.3. The slowest

shuttling time is estimated to be:

$$t_{junction} < 50 \text{ ms} \quad (5.3)$$

with the contribution to $\bar{n}$ being $\bar{n}_{Time} = 6.7$. This constraint does relax the hardware requirement for the DACs update rate: Assuming fifteen electrodes need to be updated on average for each step, with a $2 \mu\text{m}$ step and with the shuttling distance, between two arms of the X-junction, being 2.2 mm , the minimum channel update rate, assuming a shared bandwidth for a single DAC module, is therefore:

$$f_{DACupdate\text{rate}} > 330 \text{ kHz} \quad (5.4)$$

The constraint on extra heating due to shuttling is now considered with the budget left on $\bar{n}$:

$$12\bar{n}_{junction} + 16\bar{n}_{split} + \bar{n}_{swap} < 13.3 \quad (5.5)$$

Assuming the different sequences contribute heat to the ion equally, the heat induced by ion position control can be simplified as:

$$\bar{n}_{shuttling} < 0.46 \quad (5.6)$$

The physical system that will test QEC codes is now constrained for both quantum control and ion shuttling capabilities. The framework for compiling quantum code was demonstrated in this chapter as suitable for initial studies of QEC codes as well as having clear solutions toward operating larger devices.

Chapter 6

Automation for quantum experiments

As the requirements are now defined to build the experiment dedicated to test QEC codes, the hardware architecture can now be detailed. As mentioned in Chapter 1, several apparatus are operating in the research group in parallel and share multiple experimental requirements, such as the control and monitoring of the lasers used to trap ytterbium ions, cooling down the vacuum system, preparing the vacuum system containing the ion trap, and finally running the experimental calibrations.

This chapter describes three supervisory control and data acquisition (SCADA) projects built with the aim of being used by every experiment in the research group, and a set of software improvements in the experiment described in Section 3.2.2. These projects mitigate complex tasks previously developed for each experiment in providing functional black boxes accessed via a simple and ergonomic interface.

6.1 Laser controller

6.1.1 Introduction

Lasers are a fundamental part of the ion trapping process. In selecting high power laser units, they can be shared by several experiments simultaneously. However for local experimental control, such sharing represents a design issue. Either concurrent experiments can all control the laser, which would be a regular source of conflict and disruption, or, only one experiment assumes control forcing the others to rely on it; giving more responsibilities and forming an inter-experiment hierarchy.

This dilemma was solved by designing and implementing a global laser controller framework, detached from every local experiment controller but reachable by them to provide monitoring and control. By abstracting

away the considerations of hardware instrumentation, disparities between laser technologies, calibration, and locking, controlling an ion trap device is simplified which increases productivity and efficiency.

6.1.2 Problem assessment

Laser control usually relies on sending an analog signal to a dedicated laser controller unit. However for resilient control of a laser both piezo-actuator control and laser current control are required, both being sensitive and fragile: large current or voltage changes could permanently damage the laser diode.

A more specific issue comes from the complexity of such a controller, which has to remotely access and control several computers, wavemeters, lasers and DAC systems. The ability to override the framework with manual access at any layer needs to be simple and practical: such an ecosystem may require regular maintenance and code upgrade impairing the controller behavior, or on-the-fly wavelength tuning of a laser may be needed for specific experiments.

6.1.3 Organization

The projects are written entirely in LabVIEW, and consists of two main layers: the hardware controllers and the controller manager.

Hardware controller

The controllers are individual LabVIEW projects dedicated to control a specific element of hardware. There are two categories of controllers. The first is the wavemeter controller, a simple interface to the High Finesse SDK. The second is the DAC card controller, which can lock diode lasers with a PID loop between the wavelength information from the wavemeter and the analog control of its piezo-actuators and current source. The number of lasers that can be locked with a DAC card only depends on the number of channels available.

Each controller GUI is designed so that the controllers can be operated individually. In the case of the DAC card, the interface as displayed in Figure 6.2 allows, among other parameters, the modification of the PID gain settings or the choice of the DAC channel being used. The DAC channel can even be modified manually without locking mechanism in case mode hops are interfering with the PID loop. The code is reusable to most DACs, with as many channels as needed with small adjustments, and has already been deployed on three different DAC cards.

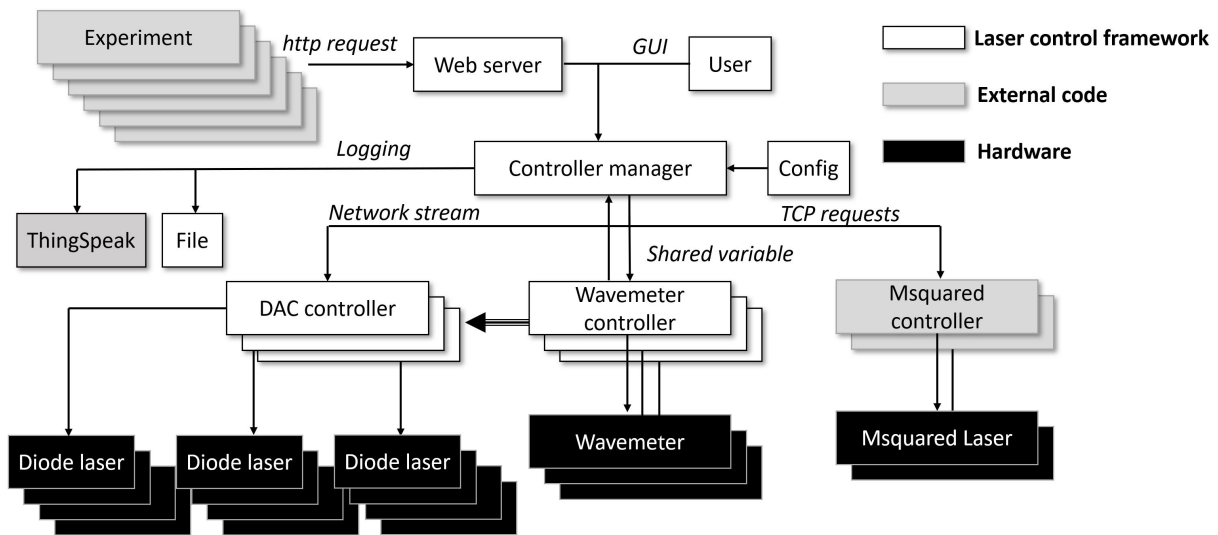


Figure 6.1: Diagram representation of the laser control framework. Experiments and users interact with the top layer software either through the built in GUI interface or remotely via a web server API which relays request to the appropriate laser controller and displays the laser wavelength.

The wavemeter controller broadcasts the wavelength information as shared variables over the research group network and are accessed by the required software such as the DAC controllers.

Controller manager

The controller manager acts as a central point for interacting with all controllers within the framework in the laboratory. It identifies each laser with a cluster of information such as its wavelength and its DAC controller channels. This information is setup in a configuration file, read when the program starts. The program stores data for long term monitoring, where the wavelength of each laser is saved in log files every two seconds. The data is also sent to a ThingSpeak server so that the laser information can be accessed in real time online. The configuration file used by the controller manager is also written as a shared variable so all DAC controllers can quickly update their map between laser and wavemeter channels. It limits the effort expected from the user to update the framework when new lasers are added.

As displayed in Figure 6.3, the GUI provides locking and monitoring capability over all lasers within the framework. By pressing the locking button, the wavelength target and the locking command will be sent to the appropriate channel and DAC controller over the local network. The controller manager is also capable of interacting with the M Squared Solstis laser SDK. The M Squared was considered to be already controlled by its own hardware controller with only an abstracted set of functions being available via TCP/IP based requests. Therefore the access to the M Squared lasers are directly integrated to the controller manager code and not via

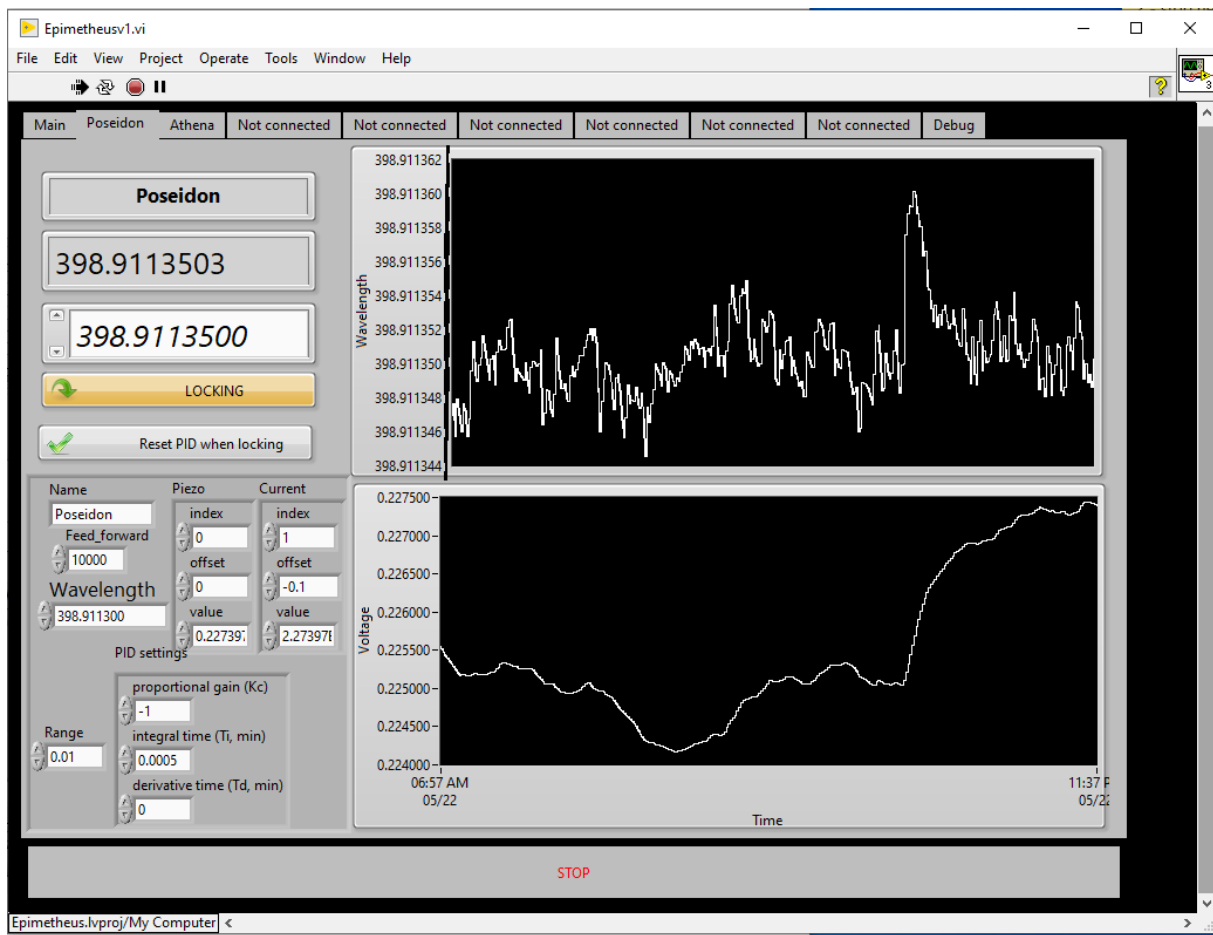


Figure 6.2: Laser controller GUI. Each tab displays all information corresponding to a single laser. On the top left the laser name as well as the target and current wavelength are shown. On the right are located accumulation scans of both the wavelength and the voltage applied to the piezo-actuator controller to lock the laser. On the bottom left is the control panel where the DAC channels can be modified as well as offset values to those channels can be manually added. the wavelength target can be adjusted and locking is achieved via the locking button. The number of laser connected is dependent of the number of DAC channels available.

another hardware controller project.

The distinctions between all categories of lasers are abstracted away in the controller manager with a state machine representation for each controlled laser. The “locked” and “unlocked” states are accessed via a multi-step sequence whose code content depends of the type of laser considered.

API

The controller manager also deploys a LabVIEW web server accessible on the internal network. A REST API, using HTTP requests as interface, was developed so that experiments can remotely access both monitoring and control features. Each HTTP request is associated with its experiment via the sender IP address. The request is then added to a custom FIFO buffer tracking “non-completed” requests. The buffer is available as a LabVIEW

global variable to be accessed asynchronously by the controller manager and protected by semaphores to avoid data corruption. Once the request is completed the request in the buffer is updated to a “completed” mode which triggers the answer to the HTTP request from the server. Finally, the request in the buffer is deleted.

The API requires the experiment to “subscribe” to specific lasers. Once subscribed to a laser the experiment can monitor its wavelength, power seen by the wavemeter and can also send new locking targets. Subscription from an experiment to a laser can be seen in the controller manager GUI in Figure 6.3 next to the laser wavelength and power: Each laser lists all the experiments subscribed to it.

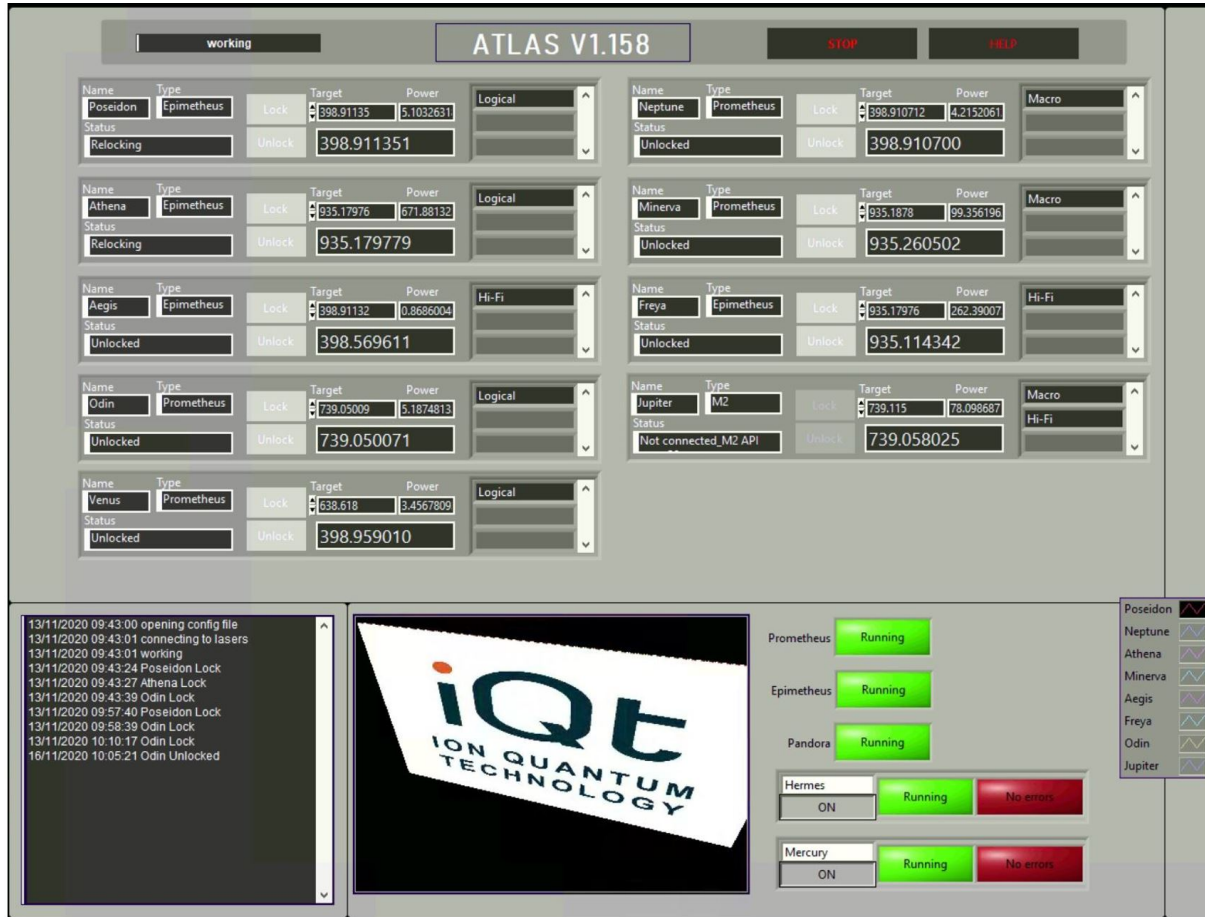


Figure 6.3: Master laser controller GUI. Laser controls seen in Figure 6.2 are also available here. Information about laser power are also displayed in addition of the wavelength. Additional displays regarding the wavemeters status and the command logging box enables quick debugging.

6.1.4 Future work

Laser control and monitoring are the objectives of this framework which is now deployed and working. However such a complex software architecture may require several improvements in the future both increasing the resilience of laser control and its scale. Two main updates are therefore recommended for future developers of

this framework:

- PID controllers would greatly benefit from an auto-calibration feature. Indeed simultaneously applying voltage and current based on a experimentally measured ratio, called feed-forward, significantly improves the wavelength range attainable for a laser without mode-jumping. While Toptica lasers are usually already calibrated, home made diode lasers have very sensitive coefficient and set points, which would need to be measured regularly. It unfortunately involves long scans, and heavy post-treatment analysis which are non trivial tasks for robust automation.
- Concurrent control of a shared laser can have serious disruptive effect on an experiment. Therefore the controller should be updated with a “policy” warning of a change, or forbidding changes of a laser wavelength as long as more than 1 local experiment controller is subscribed to it.

6.2 Cryogenic controller

This section will cover the distributed cryogenic controller providing the cold Helium distribution lines mentioned in Chapter 7, software developed with the help of Christophe Valahu.

6.2.1 Hardware setup and controller goals

In order to explain fully some of the design decision associated with this controller a brief overview of the cryogenic system itself, designed by Raphaël Le Brun-Ricalens [148], will be presented first: The system is decomposed, as displayed Figure 6.4, into two main parts being the cooling reservoir, the cryostat, and the distribution lines. The cooling element is composed of two parallel cold heads. The distribution lines connect up to four experiments across the laboratory with an output feed line providing the cold Helium from the cryostat and a return feed line bringing the warmer Helium back to the cryostat.

6.2.2 Risk assessment

The short introduction above highlights two main aspects requiring a software automation process: a passive routine monitoring the temperatures in the cryogenic cooler and in the several experiments as well as an active control of the valves connecting the laboratory distribution lines to the experiment distribution lines. This active part is rarely active as most experiments stay warm or cold for long periods of time. On the contrary,

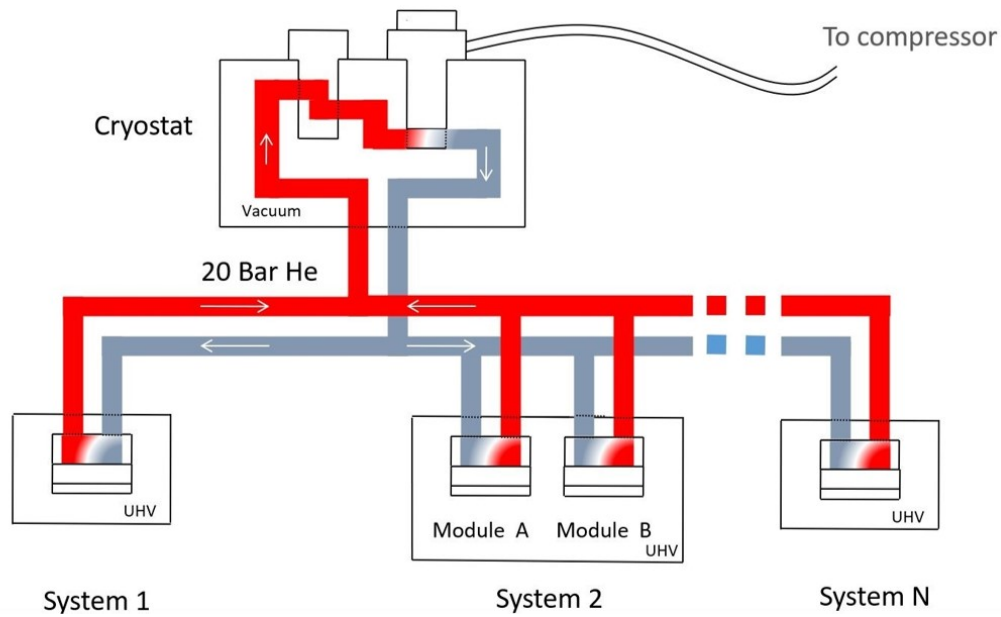


Figure 6.4: Cryogenic cooler hardware. (Top) Cooling reservoir providing cold Helium via distribution lines up to (Bottom) four experiments. The distribution lines form a closed loop: the cold helium (Gray) is sent from the reservoir to the experiment, while the warmer Helium (Red) is sent back to the reservoir. Figure taken from [149].

passive temperature monitoring is used for both the cooling procedure and stable regimes for the experiments. Continuous data acquisition over long periods of time is required.

6.2.3 Organization

The software organization is mainly centered on the passive monitoring and logging of all temperature sensors in the laboratory related to the cryogenic system. As the number of experiments simultaneously requiring cryogenic cooling is variable and that the number and placement of temperature sensors in each of those systems is left to the experiment team, the software was designed with two degrees of freedom:

- the number of experiments that can be monitored should be adjustable quickly as well as the number of temperature sensors in each experiment;
- the system should be device-agnostic, due to different hardware modules being used in different experiments, to measure temperatures.

This flexibility is integrated via a configuration file which enables a static mapping between a temperature sensor and its location in an experiment. It allows different sensors from different experiments to be connected to

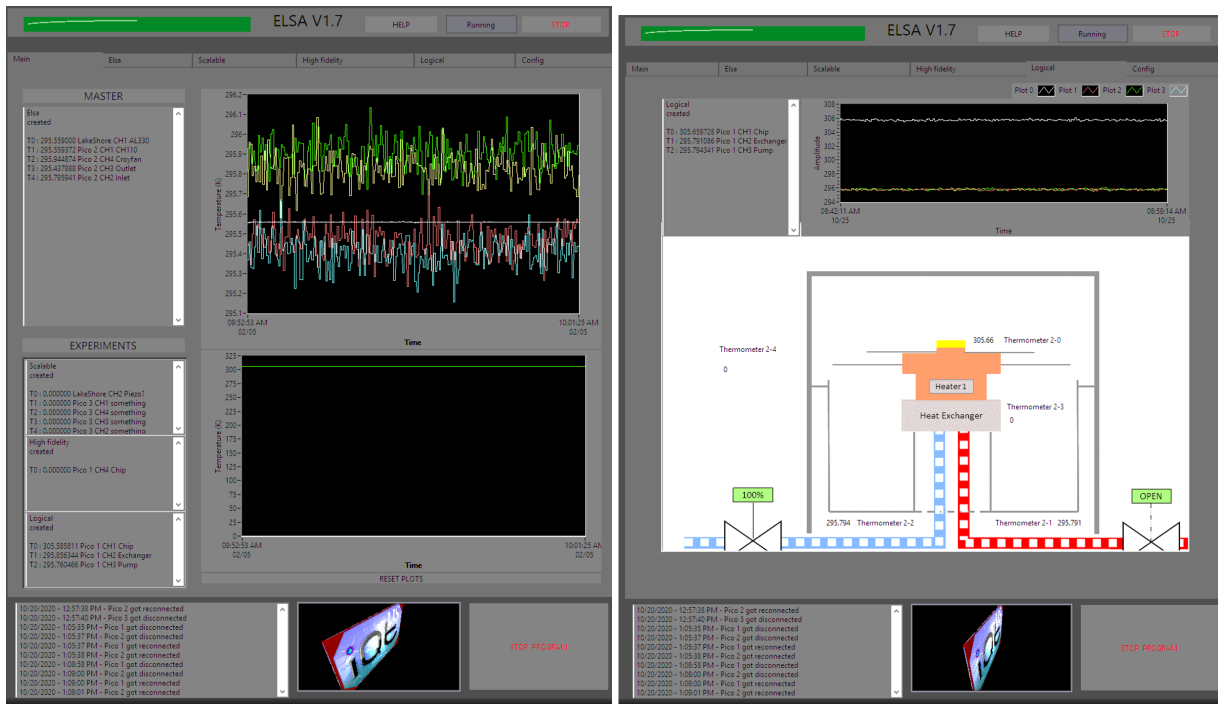


Figure 6.5: Cryogenic temperature controller GUI. (Left) Main panel displaying all temperatures in the system, with the cryogenic system on top and the experiments at the bottom. The bottom left logging display informs the user of any hardware communication issues. (Right) Individual experimental tab. Temperatures are displayed on top while the position of the sensors are represented in the experiment diagram at the bottom.

the same hardware data logger without any inconvenience, as well as saving temperature data in files organized per experiment and not per data logger. For long term operation and resilience to hardware errors, the software can attempt to reconnect to any logging device having communication issues. It enables uninterrupted monitoring for months without software crashes.

All temperatures are saved at a two second intervals and are sent to a ThingSpeak account for remote online monitoring.

6.2.4 Calibration

The standard **IEC 751** concerns platinum resistance temperature sensors, which includes the PT100 sensor, and defines tolerances and resistance-to-temperature conversion tables. This standard covers down to 70 kelvin, as at lower temperature the sensitivity of platinum sensors considerably degrades as illustrated in Figure 6.6. The common module used in this architecture¹ therefore stops providing temperature measurements outside of this standard temperature range.

It is however possible to access the physical resistance measurements instead at a faster rate and deduce the

¹Pico Technology PT-104 Data Logger

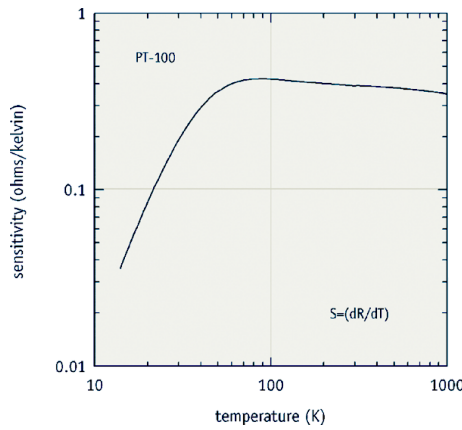


Figure 6.6: PT100 sensitivity. Below around 70 kelvin the sensitivity decreases and reduces the measurement accuracy. Individual calibration are needed at lower temperature to provide accurate measurements. Figure taken from [150].

temperature later in software. Measurement below 70 kelvin becomes possible, although not accurate up to a few degrees. This inaccuracy can be compensated by buying individually calibrated sensors which provide a dedicated resistance to temperature map per sensor: The software was extended to enable the use of such sensors: by indicating in the configuration file which sensor is calibrated and its calibration file number, the controller will be able to access the conversion table and provide accurate temperature measurement without additional post processing.

6.2.5 Use case

This cryogenic monitoring system is also useful for diagnostic situations which it was not originally designed for. The example described here occurred during the early trapping runs in the logical qubit apparatus shortly after baking. The signal sent to the RF electrodes of the chip is strong enough to affect the chip temperature and can therefore be monitored by the PT100 located under it. As shown in Figure 6.7, the software logging provided evidence that the chip stopped resonating following a trapping run.

6.2.6 Future work

The major challenge yet to be faced is to safely integrate the valve control opening the helium feed line to each experiment into the monitoring software. The remote control itself of the valves, implemented by Raphaël LeBrun-Ricalens, consists of a high torque and low speed DC motor directly connected to each valve, with the motor being controlled by an Arduino module.

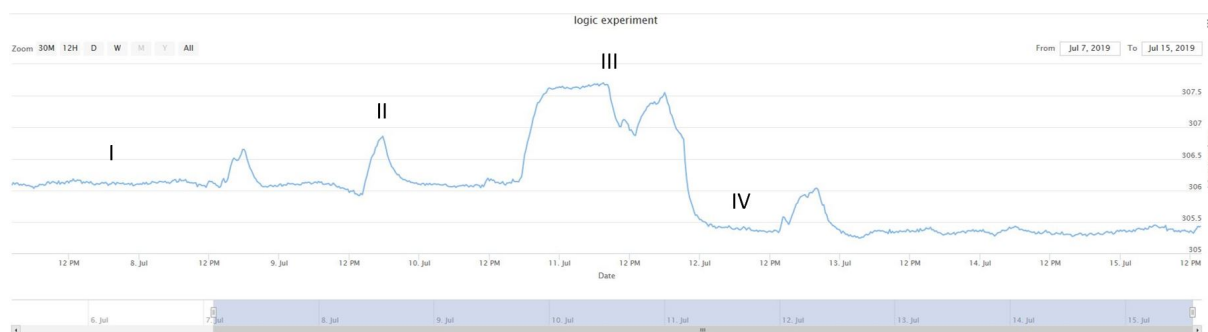


Figure 6.7: Screenshot of the cryogenic controller during trapping runs. The temperature at the chip when resonating with a 200 V amplitude RF signal is displayed in (I). When the atomic oven is used, the temperature of the chip can be seen to increase as shown in (II). In this case, a longer and more intense run of the atomic oven was attempted resulting in a higher temperature measured at the chip (III). The subsequent temperature in (IV) after the oven was turned off indicates the chip was not resonating anymore.

These has not been integrated yet because the valve control does not provide feedback over how much the valve is opened. Translating a command to the Arduino module regarding the aperture percentage of the valve, will require a very careful calibration for each valve. Any error happening during or after this calibration may result in opening cold helium lines which may have catastrophic consequences to the laboratory, the experiment and the cryogenic system itself. Therefore a protocol to guarantee the working condition of the valve controller is still being developed and verified.

6.3 Vacuum system oven controller

In order to reach UHV conditions vacuum systems are often “baked” as described in Chapter 7 by increasing their temperature to accelerate the release of adsorbed elements.

6.3.1 Problem assessment

The software control of the baking chamber is a PID loop adjusting a heater based on a temperature measurement of the system. This active control system does not present much danger for the user, even at the maximum achievable temperature of approximately 200 °C. However the vacuum system is fragile as its windows are quite sensitive to temperature fluctuations of more than 50 °C/h. A software or hardware incident leading to a sudden change on the PID loop output, such as stopping the heater entirely, or switching it to 100% of its capacity, is assumed to apply a temperature shock large enough to permanently damage the vacuum system and should be avoided at all costs.

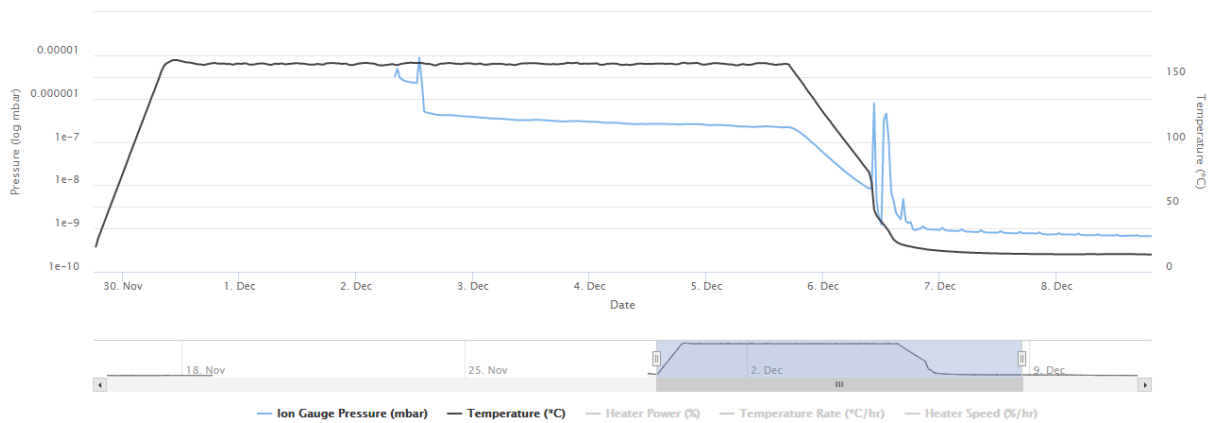


Figure 6.8: Screenshot of an online monitor covering a successful bake. The temperature (black) ramp is stabilized at 10 degrees per hour, while the high temperature target was maintained for a week. After a few days the ion pump is turned on allowing to monitor and log the pressure (blue). The pressure slowly decrease during the bake then goes down when the system is cooled down. The further pressure peaks at the end of the bake corresponds to the use of the TSP.

6.3.2 Organization

The software control itself was written by Dr Sam Hile and uses a state machine to easily distinguish the different program behavior. Logging information is saved regularly both locally and online for remote monitoring, and include information including the heater state, the applied heater power, the system pressure and its temperature. A GUI was developed in python as a wrapper around the state machine code and is displayed in Figure 6.9.

The heater is controlled by a custom made switch between the current source and the heating elements. This switch is activated via a TTL signal: the current flows to the heater as long as a logical “1” is received. The TTL signal is generated from a Raspberry pi 3b on which the software control code is hosted, and the heating power modulation is achieved via PWM signaling with a 0.1Hz clock frequency.

It was observed that incidents with the Raspberry pi itself, although extremely rare, are susceptible to happen when such a control code is run uninterrupted for more than two weeks. In order to mitigate any risk of hardware failure a backup module was designed using an Arduino and implemented in the system as shown in Figure 6.10. It controls the TTL output signal of the Raspberry pi and receives regular updates on the PWM cycle values via serial communication from the Raspberry pi. If the main control software were to malfunction in any way, the Arduino will cut the TTL connection between the Raspberry pi and the heater module and replace it by its own, and will send a constant signal modulation based on the last received heater value. Therefore the system should stay at the same temperature until a user notices the situation and manually intervenes to restart the software itself.

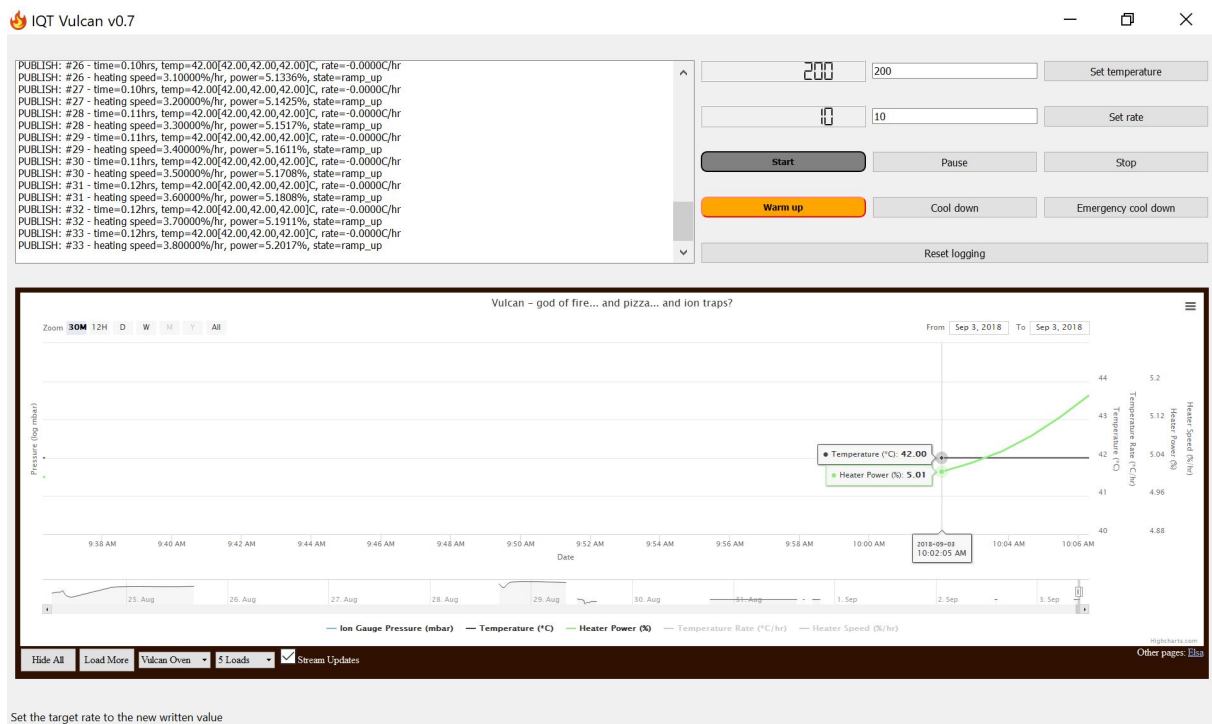


Figure 6.9: Screenshot of the baking software monitor (Vulcan). The GUI is organized in three sections: the logging on the top left allows the read the most recent information on the system, the top right hosts all the controls, and the bottom contains a web browser directly accessing the online monitoring tool.

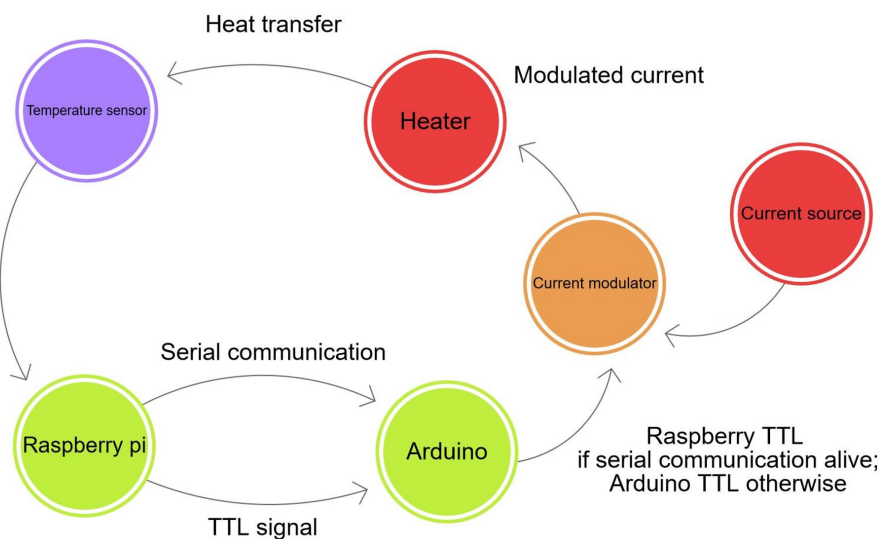


Figure 6.10: System diagram of the baking system. The feedback loop is based on the system temperature, and is controlled by a Raspberry pi. The Arduino monitors the Raspberry pi activity and remains passive. The heater is controlled by modulating the incoming current. If the Raspberry pi stops being operational for any reason, the Arduino will keep sending the last modulated current to the heater to maintain the system temperature.

An additional short code was written in python to be run on an external computer to monitor the online logging information from the software controller. Lack of updates after a certain preconfigured timeout will trigger

emails being sent to the user for quickly responding to the situation. This code is hosted by a different host than the pi so that that alert message can be sent despite an internet failure on campus.

6.3.3 Future work

The layers of protection installed are sufficient to circumvent most major issues that could occur during a bake but a clearer documentation, and more automated recognition of the different interfaces for pressure measurement devices and an updated GUI interface to assist baking procedures will greatly help future users.

6.4 Automated quantum experiment control

The experimental apparatus in which the polychromatic gate was implemented and characterized in Section 3.2.3 uses a home made LabVIEW program developed and maintained by the research group. I implemented a series of quality of life improvements to increase automation and productivity which are detailed in this section. Although written for this particular framework, the ideas presented here are generic and could be deployed in any trapped ion quantum experiment if applicable, and improved following the calibration framework formulated in Section 3.3.

6.4.1 Ion recrystallization

Ions in Quantum experiments regularly decrystallise, due to a collision or when staying too long without Doppler cooling. For high enough trap depth the ions do not leave the trap immediately and can be cooled down to recrystallise again by momentarily red detuning the Doppler cooling laser.

Ion decrystallization was the only reason experimental users were required to monitor the apparatus during experimental scans. A recrystallization module was installed to trigger, both at rest or at run time at a step N or in the preparation step, when the ion fluorescence would drop to zero. If it happens at run time the experiment would pause. The Doppler cooling laser would then be red detuned and switched to high power via the AOM until the ions appear again or a timeout is reached after which the ion is considered definitely lost. If an experiment was running, then the scan restarts at step N-1 erasing the previous datapoint.

The ability to pause and stop experiments can be further improved by additional monitoring of the wavelength or the applied RF voltage on the electrode for more precise diagnostic of the situation.

6.4.2 Automated experiment integration

The LabVIEW experimental code, as described in Chapter 5, uses a Python framework for compiling pulse sequences into hardware. The interaction between python and LabVIEW is automatic, although the data analysis, via a separate Mathematica notebook, was not. I integrated most post-experiment analysis by converting existing Mathematica notebooks into .wla Mathematica scripts that can be called remotely by LabVIEW code for automatic evaluation of the scan data. The script also returns the fit quality allowing to the main LabVIEW code running the experiment to monitor if the scan is already satisfactory, and then to automatically stop the scan. This architecture was further improved by Christophe Valahu who replaced the Mathematica code by Python scripts, and facilitated in the LabVIEW main program the exploration, visualization and analysis of previously taken data.

I separated the calibration parameters from the Python experiment scripts by defining them in a dedicated configuration file, readable by the experiment scripts, and settable by the post-experiment scripts if a fit is found for the desired scan. By combining these two features, experiments can stop and upgrade their calibration parameters by themselves.

6.4.3 Long term automation and outlook

A final improvement was made by adding a server-client python script on top of the experiment scripting interface for scheduling experiments automatically. The server runs a python script of commands, representing scans but that could be effortlessly adapted to quantum circuits, translating into asynchronous commands to the LabVIEW experiment scripting interface, as the client. Once a scan is run, its success is returned to the server. Because the server script is pure python, dynamical calibration can be programmed without requiring any user intervention. This server-client structure is similar to the scheduler which was later used in the logical qubit experiment in Chapter 7, and could be used as the foundation for the efficient calibration routine detailed in Chapter 3.

Chapter 7

The logical qubit apparatus

Conducting a seventeen qubit surface code experiment requires trapping, maintaining, shuttling and coherently manipulating seventeen ions on a trap. There are only few experimental quantum processors on this scale: the 53 qubits Rochester device by IBM [151], the 53 qubits Sycamore devices with Google [152] and the 28 qubits Aspen-7 device by Rigetti [153]. At the time of writing this chapter, the best commercial system with ions are the IonQ 32 and 11 qubit devices [38]. It is a significant challenge to build an experimental setup with the capacity of handling multiple ions as well as the robustness to calibrate and maintain all quantum operations. This chapter will describe the initial hardware built to prepare for the seventeen qubit surface code experiment, as well as the development process for some of the technologies implemented in our setup.

7.1 Ultra high vacuum chamber and content

7.1.1 Vacuum technology

It is vital to reduce the pressure in the environment of a trapped ion device. While the ion may be confined by the trap itself flying molecules and particles in the air will regularly collide with the ion and push it out of the trap, or transition the ion via inelastic scattering to a state not repumped by the available lasers. A simplistic model of ion lifetime is therefore presented.

The mean free path λ is the average distance traveled by a particle of radius r before entering into a collision with a gas of pressure p and temperature T with another particle. It is calculated via:

$$\lambda = \frac{kT}{4\sqrt{2}p\pi r^2} \quad (7.1)$$

where k is the Boltzmann constant. We will assume a worst case scenario and define this value as the average distance traveled by particles in the vacuum before a collision with the trapped ion. The next approximation is to consider the gas to be made of H_2 molecules only, because they are the main outgassing component from stainless steel [154], giving the kinetic diameter $r_{H_2} = 2.10^{-10}$ m. We approximate the vacuum system to be at room temperature $T = 300$ K. The time before a collision is then the mean free path divided by the speed at which the H_2 are traveling. The probability of different velocities is given by the Boltzman distribution. We assume that the velocity is given by the average of the Boltzman distribution $v_p = \sqrt{\frac{2kT}{m}}$. With $m_{H_2} = 3.35 \cdot 10^{-26}$ kg the Hydrogen mass we calculate $v_p = 1570$ m/s. The average lifetime τ_{ion} of the ion can finally be estimated as:

$$\tau_{ion} = \frac{\lambda}{v_p} = \frac{\sqrt{kTm_{N_2}}}{8p\pi r_{H_2}^2} \quad (7.2)$$

While much better refinement to this model can be developed, this first overview allows to clearly distinguish an inverse relationship between ion lifetime and pressure as illustrated in Table 7.1.1: There is a threshold around 10^{-9} mbar or 10^{-7} Pa, called Ultrahigh vacuum (UHV) regime, below which it becomes possible to confine ions for more than a few minutes.

Ion lifetime estimation			
Regime	Pressure P (Pa)	Mean free path of the gas particles in the vacuum chamber (m)	Lifetime of the trapped ion (s)
Air	10^5	2.79×10^{-8}	1.77×10^{-11}
Rotary pump	10^{-1}	2.79×10^{-2}	1.77×10^{-5}
UHV threshold	10^{-7}	2.79×10^4	1.77×10^1
Common ion traps	10^{-10}	2.79×10^7	1.77×10^4

Table 7.1: Numerical estimation of the ion lifetime in different pressure regimes. The passage down to the Ultra High Vacuum (UHV) regime mirrors lifetime growing above a second. The usual target of 10^{-11} mbar regularly mentioned in the literature gives hour long lifetimes.

When considering a quantum circuit running a number of ions, n , we can estimate the lifetime of the qubit register, defined as the average time before a single ion in the register is lost, to be:

$$\tau_{register}(n, \tau_{ion}) = \ln \left(\frac{n}{n-1} \right) \tau_{ion} \quad (7.3)$$

where we have assumed an average ion lifetime of τ_{ion} . Good vacuum conditions will limit the difficulties of re-trapping and/or keeps the ions in the trap longer. A seventeen qubit register - the required register size for one surface code based logical qubit - can be used as an example. If we use ions with a lifetime of one day, our register will be disturbed (by the loss of an ion) approximately every 90 minutes.

7.1.2 Vacuum chamber

Such a pressure regime involves the use of a specialized chamber called a vacuum chamber in combination with at least one initial baking around 180°C and thereafter continuous pumping. One important obstacle regarding UHV experiments is the choice of materials: they need to be able to first survive the baking condition then not outgas in the chamber when the pressure drops. Many components such as organic materials will continuously outgas in low pressure environment, considerably reducing the capacity of the chamber to stay in UHV conditions. UHV compatible materials include most metallic and ceramic components [155].

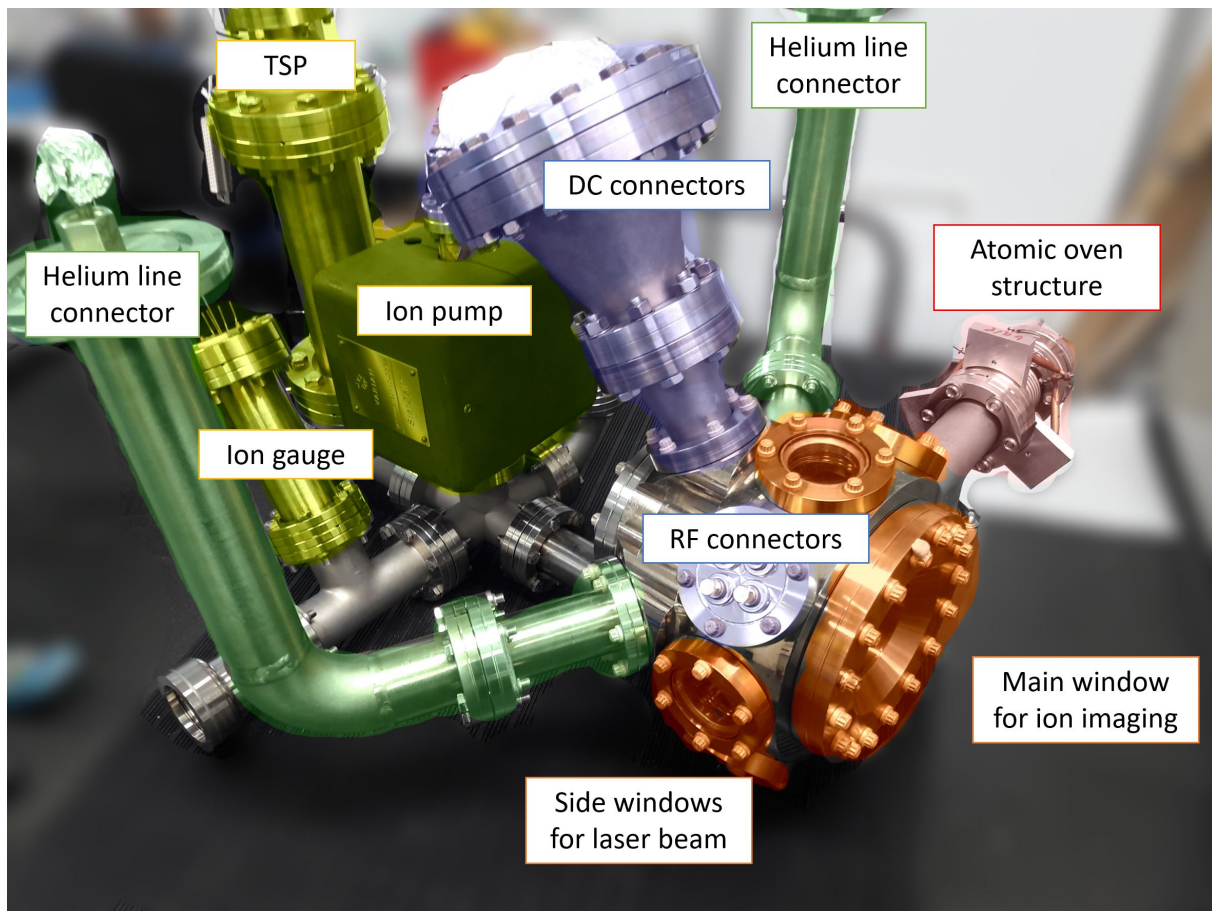


Figure 7.1: Picture with false colors of the vacuum chamber used for the logical qubit apparatus, with distinguished areas for UHV control (yellow), ion control (blue), trapping (red) and optical access (orange).

The complete system to operate the ion trap in a vacuum system is displayed in Figure 7.1: areas in yellow correspond to the components necessary to reach and maintain UHV conditions inside the vacuum chamber. The ion pump¹ actively maintains the UHV conditions and the ion gauge² gives the pressure reading. The Titanium sublimation pump (TSP) is used during the baking process and the metallic valve, not visible in the

¹Agilent 4UHV controller

²Agilent XGS-600

Figure and behind the ion pump, is used before baking for initial pumping with a turbo-molecular pump. The two elbows in green are the Connectors between the internal cooling line in the chamber and the distributed cryogenic Helium line. The sections in blue represent the RF and DC signal connectors which are used to trap and manipulate the ions. The red structure in the atomic oven setup is used for ion trapping. Finally the optics are represented in orange and surround the main chamber which contains the ion trap. The side-windows are used for laser access while the main chamber is used for ion imaging.

7.1.3 Surface ion trap

The Paul trap geometry used for the experiment is a micro-fabricated X-junction surface trap as shown in Figure 7.2. The trap has a size of $10 \times 12 \text{ mm}^2$ and contains 80 DC electrodes, 2 RF electrodes and 2 ground planes. The chip geometry was designed to trap at $125 \text{ }\mu\text{m}$ height and was build by the external contractor INEX Microtechnology Ltd using micro-fabrication processes described in [75, Chapter 3].

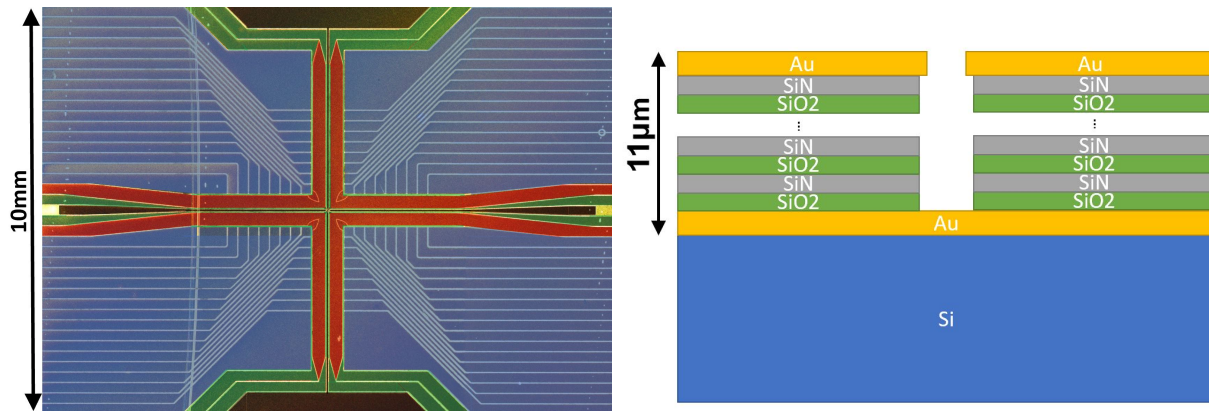


Figure 7.2: (Left) High resolution picture with false colors of the previous generation of the X-junction surface chip used in the vacuum chamber. The electrodes in blue reaching the corners of the X-junction are the DC electrodes. The red electrodes drawing the X-junction shape are the RF electrodes. The green electrodes at the top and bottom of the image reducing in size to follow the RF electrodes are the rotation electrodes. The grounded electrodes are in black and all the blue area represents the DC electrodes. (Right) Diagram of the chip layers for two neighbor electrodes. the gold layer at the top is slightly larger to shield the dielectric layer from being charged by the UV lasers. The dimensions are not at scale.

As represented on the right of Figure 7.2 the chip is built on a silicon substrate with a two level gold electrode layers separated by a dielectric structure consisting of alternating layers of silicon nitride and silicon dioxide. Every new chip is benchmarked on the maximum sustainable RF voltage applied by running a destructive test on one sample. Every chip considered for use is visually inspected and checked for shorts or disconnected electrodes. Chips with shorted electrodes but with no visible obvious damage are first cleaned with a sonic bath in acetone. It is observed that for some chips shorted electrodes happen after the integration in the vacuum

system. Such shorts are removed by applying large current, on the order of 1 A over a few seconds, on the shorted connection. The location of the short, assumed to be small compared to the dimension of the electrodes, is heated up and vaporizes.

The chip heating rate can be estimated from the previous chip generation, measured at a temperature of 8 K and ion height $h_{ion} = 98 \mu\text{m}$ by Dr Anton Grounds to be $\dot{n} = 423 \text{ s}^{-1}$ for a single ion [156]. At room temperature and ion height $h_{ion} = 125 \mu\text{m}$, the heating rate is projected to be $\dot{n}_{300K} \approx 1600 \text{ s}^{-1}$. For entangling gates being driven on the stretch mode, the effective heating rate constrained in equation 4.12 is calculated by dividing the measured heating rate, for a single ion, by a factor of at least 40 [2]. The projected heating rate on the stretch mode $\dot{n}_{st,300K} \sim 40 \text{ s}^{-1}$ is, at room temperature, above the specification calculated in equation 4.12, but will be within specification at 70 K.

7.1.4 Internal electronic setup

All electrical channels on the chip need to be brought inside the vacuum system via special flanges containing electrical connectors while maintaining vacuum. The RF interfaces were chosen to be SMA connections, and inside the vacuum chamber the signal is carried with a coaxial cable. The DC signals are sent via two DD-50 connectors, and the signal is carried via individual Kapton insulated wires.

The connection between the in-vacuum wires and the chip is a three-layer PCB setup adapted from [157]. The DC signals are connected to a PCB called the “Back PCB” which contains an RC filter at a frequency cutoff of 260 kHz to suppress the noise accumulated in the vacuum via the RF signal lines. The back PCB carries the DC signals to a “front PCB” via spring loaded pins. The front PCB also has SMP connectors to receive the RF trapping signal as well as the coherent RF and MW signals described below. Both DC and RF signals are carried in close proximity to the chip and connected to it via wirebonds. The back PCB is fixed to the stainless steel frame attached to the vacuum system, while the front PCB is fixed via both bolts to the chip holder and the spring loaded pins from the back PCB.

The two separate branches of RF electrodes on the X-junction chips are fed from a single connector on the front PCB. The RF tracks are designed to have equal signal path lengths and therefore phase matching between the two branches, which will avoid phase induced micro-motion of the ions.

The front PCB was redesigned from [157] to host the third PCB layer and the current channels in prevision for the next generation of surface chips, which will have micro-fabricated current wires below the trapping electrodes to generate the magnetic field gradient required for quantum control.

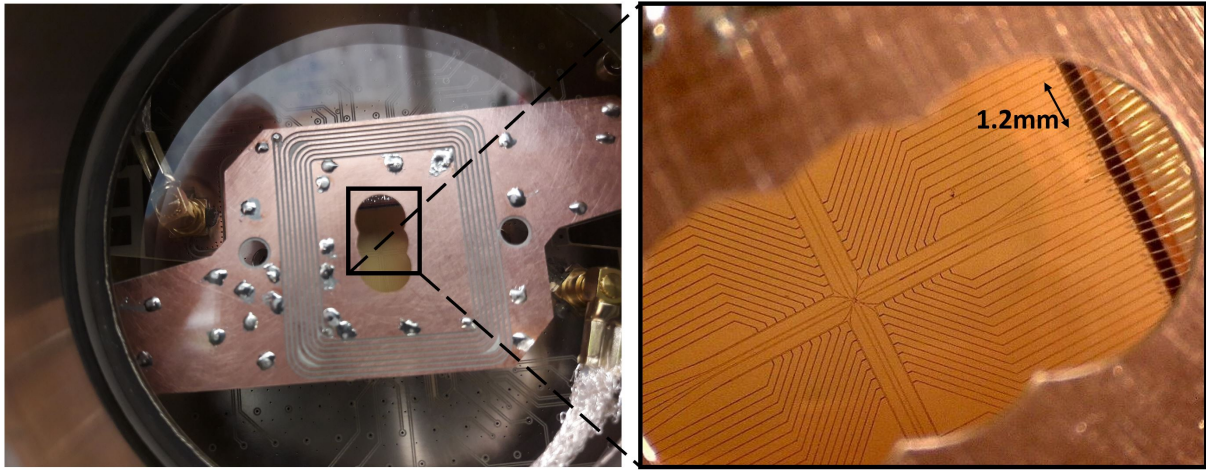


Figure 7.3: (Left) Internal electronic setup seen from the main window flange. We can recognize the surface chip in the center with the RF and MW antenna above it. The front PCB connecting MMX connectors is visible at the top and bottom of the picture. (Right) Zoomed in photo of the surface chip in the vacuum system. Wirebonds between the chip and the front PCB can be seen.

The third and top layer is a new PCB³ containing two RF and MW antenna that can emit the signals required to coherently control the ytterbium ions. The RF antenna was milled in the side facing the viewport and is visible in Figure 7.3. The MW antenna is a patch antenna, designed with simulations in EMPro displayed in Figure 7.4, to emit at 12.6 GHz and with a central aperture large enough to not block the ion fluorescence to the imaging setup. The patch antenna is milled on the side facing the ion. This top PCB is fixed on top of the front PCB and the chip, as is illustrated in Figure 7.3, with SMP connectors which carry the RF and MW signal from the front PCB to the antennas.

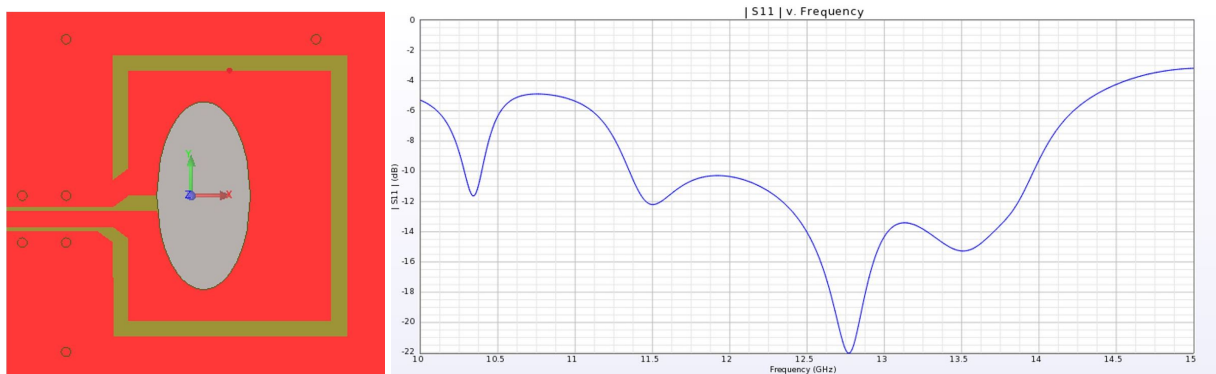


Figure 7.4: Patch antenna simulation for in-vacuum MW emission. (Left) Patch antenna model in EMPro. The design is adjusted to allow the ion fluorescence through the center, and to emit at 12.6 GHz. (Right) Simulated reflection factor S_{11} . The emission peak at 12.6 GHz is broadened due to asymmetries added to the patch antenna design.

³conceived by Dr Sam Hile

7.1.5 Helium cooling copper tube

Bringing down the temperature below 70 K will remove one order of magnitude of anomalous heating from the chip and therefore considerably increase the quantum computation fidelities [158]. Cryogenic ion traps have been demonstrated successfully below 10 K but with special apparatus including heat shields and vibration compensation [159, 156, 160]. The research group distributed cryogenic system however targets only 70 K following the blueprint requirements [1], enabling cooling of room temperature ion traps via a single helium circulation line positioned inside the vacuum system.

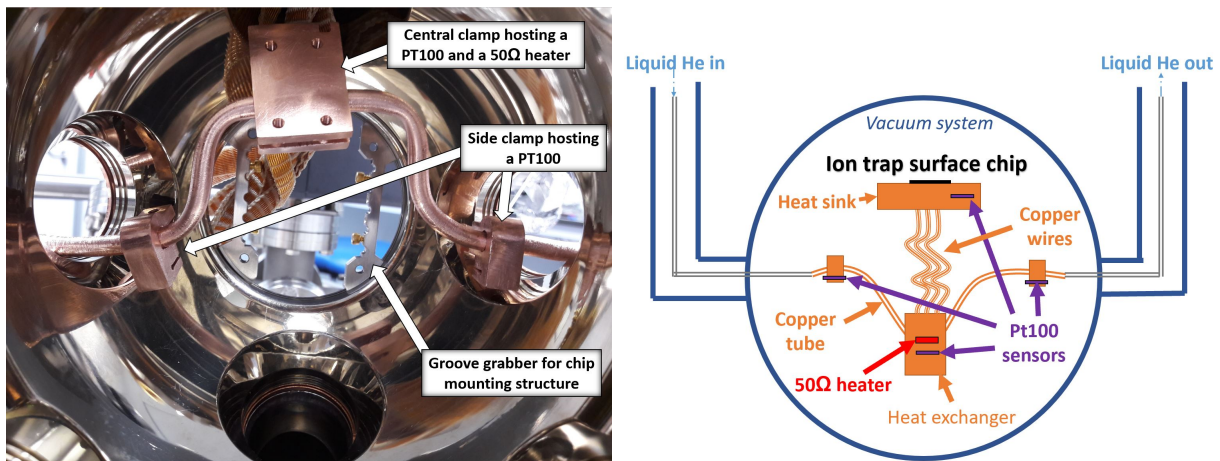


Figure 7.5: Copper tube installation inside the vacuum chamber. (Left) The chip mount, the temperature sensors, copper braids and the ion chip mount have not been installed at the time where the photo was taken. We can recognize the shape given to the copper tube to avoid the center of the chamber, as well as the three clamps installed to attach temperature sensors. (Right) diagram of the connection between the copper line and the ion trap mounting structure.

As displayed in Figure 7.5 the Helium circulation is implemented with a copper tube so that the shape can be adapted to avoid collision with the chip holder. Two copper structures are clamped on each side of the vacuum system with bolted PT100 temperature sensors for heat absorption measurement by the apparatus. A main clamping structure in the middle of the copper tube also hosts a 50 Ohm heater for local temperature stabilization as well as a welded set of thick copper wires, commonly called copper braids, for conducting the cooling power to the chip.

The connection between the chip holding structure and the cooling setup is illustrated in Figure 7.6. Those copper braids are bolted into a “bottom” copper block. A “top” copper block is bolted into the bottom one within the central aperture of a stainless steel frame fixed to the vacuum system. Therefore the copper blocks are fixed in place. The top copper block also hosts a fourth temperature sensor, the closest one to the chip, and connects to a final copper block hosting the chip via melted indium.

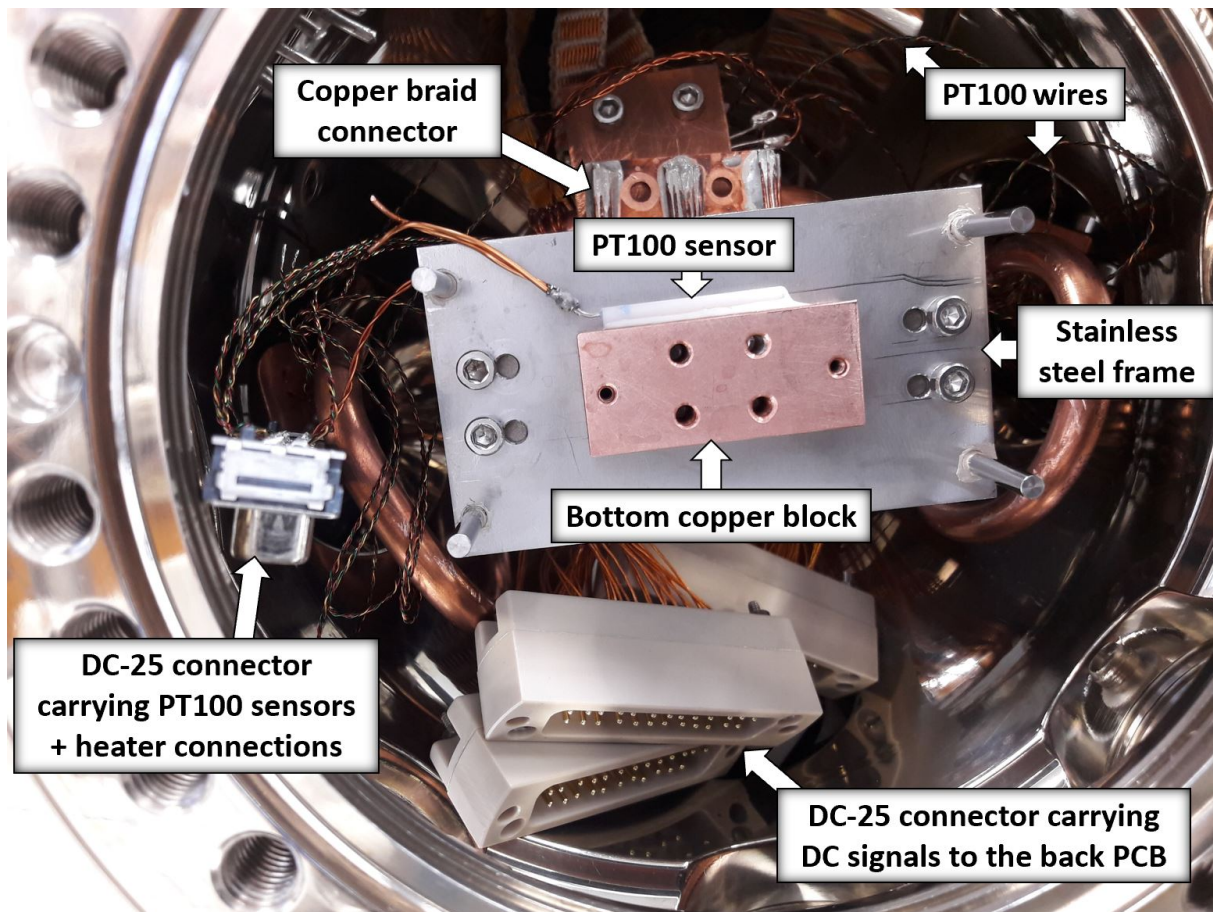


Figure 7.6: Content of the vacuum chamber while being assembled. The copper tube, its copper clamps and sensors can be seen in the back. The stainless steel frame attached to the vacuum system has been installed with the first copper block containing a PT100 sensor.

The temperature sensors are uncalibrated PT100 as they are cheap and reliable at the target temperature of 70 K, and are all thermally anchored to the cooling tube [161, Chapter 4.4]. However, as detailed in Section 6.2.4, when reaching lower temperatures every sensor follows a distinguishable resistivity curve and uncertainties up to ± 5 K can be expected. A cooling cycle is presented Figure 7.7 starting when the distributed cooling system is cold but the distribution lines are closed at the beginning. The final temperature measured near the chip is higher than the expected 70 K, but the copper tube circulating the Helium is measured to reach 50 K: it indicates that the distribution lines are working and that the high temperature at the chip may be due to a bad thermal contact with the copper tube.

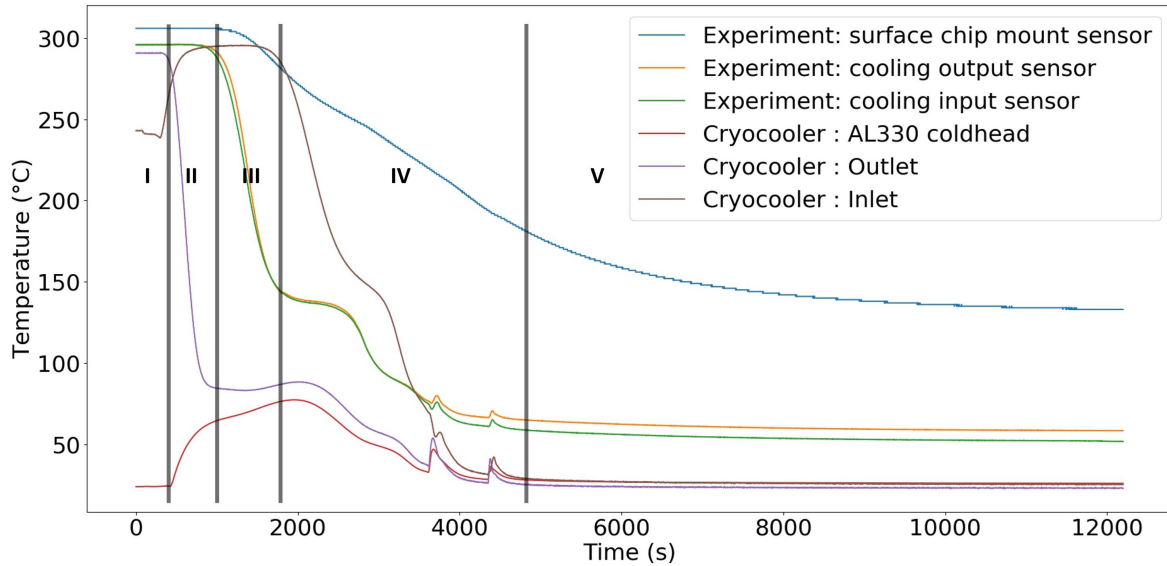


Figure 7.7: Cooling sequence in the logical qubit apparatus in UHV conditions. At the beginning in (I) the valves connecting the cryocooler to the experiment are closed and the helium is confined in the cold heads. (II) The first step is to turn on the cryofan sending the helium gas to the distribution lines in the laboratory, leading to a quick decrease in temperature of the cryocooler. (III) Then the valve between the helium lines and the experiment is opened leading to a temperature drop in the copper tube sensors. The heat absorbed by the cryocooler can be seen in the increase in temperature of the cold head as well as the very late decrease (IV) in temperature of the returned helium to the cryocooler at the inlet. (V) The two systems stabilize after more than an hour. the short increase in temperature in the cold heads corresponds to the manual injection of warm helium in the system to maintain the helium pressure constant at 20 bar.

7.2 ARTIQ setup

Quantum logic with trapped ions requires real-time control of several devices such as RF signal switches for quantum control, DAC waveforms implementation for fast shuttling and accurate triggering for high fidelity quantum state detection. The essential tool for real-time control is a specialized type of computer called a Field-Programmable Gate Array (FPGA). It is a device hosting re-programmable logical units enabling the implementation of hardware efficient code enabling fast, reliable control and monitoring.

Choosing the FPGA is not only a choice of speed, memory and available input/output channels but also involves choosing the language to interact with it. One of the most common languages is called Verilog and is a hardware description language: it involves a steep learning curve and time consuming specialization. Higher level language programming do however exist with hidden compilation tools transcribing the code into Verilog or equivalent hardware description language. Such processes allow easier development of experiments relying on both the CPU and the FPGA. The Advanced Real-Time Infrastructure for Quantum physics (ARTIQ) project was developed by a company called M-labs in collaboration with ion trapping groups and aims to provide a fully open source real-time experimental control framework [162]. It includes a software environment, ARTIQ,

and a hardware architecture called Sinara. Sinara devices are organized around an FPGA board called Kasli containing the FPGA⁴, data connectors to the CPU operating system and to other Sinara devices. This section will describe the electronic and software control from the ARTIQ framework used for operating the experiment.

7.2.1 The ARTIQ framework

The ARTIQ software is a python framework that allows us to write real-time control experiments into the Kasli FPGA with pure python code. In addition it provides a unified interface to interact with external devices so that not only Sinara modules but any piece of hardware can be accessed asynchronously through an interface such as Serial or REST API. An asynchronous dataset system is used to write and read python data objects across multiple experiments. Finally, plotting is implemented via a custom concept of applets adapted from PyQtgraph plots connected to the datasets to display in real time newly obtained information. This entire structure was designed to interact with a command line system. A GUI is built on top using the command line layer to display all the information and provide a central access point to every functionality.

Additional devices were added to the logical qubit apparatus for the purpose of increased automation. The drivers were written in python and the interface to ARTIQ is achieved via automatically building network-based asynchronous objects on the same computer, therefore using the IP address 127.0.0.1, and automatically converting any calls in the Experiment script to asyncio-based communication to the driver. In addition to writing drivers to home-built devices such as the laser control system described in Section 6.1, the cryogenic temperature monitor detailed in Section 6.2 and arduino projects such as the laser shutters, the following drivers have been written and implemented: the Andor SDK tools to control the EMCCD camera, Agilent 2000 series for the oscilloscopes, Picolog tc08 data logger for measuring the oven temperature, the IPS2010 power supply for powering the atomic oven, the NEX Torr ion pump, the Varian XGS-600 ion gauge and the Agilent 33522A RF generator for micromotion detection via parametric excitation.

Experiments are run via a Scheduler which maintains a queue of submitted experiments and decides which one to run. Multiple experiments can be run at the same time via different “pipelines” when they use different devices. Experiments in a particular pipeline can have different priority levels depending on the urgency of the situation. Finally, Experiments can in their own script submit themselves or other Experiments to the Scheduler, which will be a key point for the automation architecture. Experiments automatically save data and configuration in the hdf5 format [163].

⁴Xilinx XC7A100T-2FGG484I

7.2.2 Electronic hardware

Kasli is the master controller that interacts with the computer hosting ARTIQ. Kasli can be connected with additional real-time hardware modules with specially designed interfaces called Eurocard Extension Modules (EEM) which are controlled by the computer via Kasli.

The logical qubit apparatus needs three categories of hardware real-time control: DC signals, RF signals and TTL signals. The variety of hardware devices available in the Sinara architecture allows full customization of the experiment to meet the criteria detailed in Section 4.5.3.

- DC signals:

DC signals correspond to the voltages applied to the chip for axial confinement and shuttling. The ARTIQ module for DAC cards is called Zotino and hosts 32 channels of 16-bit precision DAC with a shared bandwidth of 1 Msample/s, which is three times above the minimum requirement calculated in equation 5.4. As the chip has 80 DC electrodes three Zotino cards were installed to enable individual electrode control for shuttling ions in separate zones. By splitting the neighboring electrode channels to the different DACs, the apparatus can shuttle with a maximum speed improved by a factor of three. An ADC card was also installed to provide DC measurements with a shared bandwidth of 1.5 Msample/s.

- RF signals:

We can split technologies of RF signal generation in two categories, firstly the Arbitrary Waveform Generator (AWG) which allows step-by-step construction of the time waveform permitting complex multi-mode signals to be generated. The AWG is a key component for driving complex pulses for quantum gates. These pulses are computationally expensive and involve heavy memory management although a homemade framework was developed in the research group to integrate them in running quantum control experiments [86, Chapter 4]. As the current chip is not capable of quantum logic experiments the AWG installation was not prioritized. On the other hand the Direct Digital Synthesiser (DDS) allows low-cost generation of single sinusoidal waveforms of arbitrary frequency and phase. While much less capable than an AWG, it can still provide the waveform needed for 1) generating the RF trapping potential via combination with an RF amplifier and a custom made RF helical cavity resonator [164] tuned to trap resonance and 2) operating external devices such as the Acoustic-Optic-Modulator (AOM). One module using the ARTIQ card for DDS signal generation was implemented providing 4 independent channels.

- TTL signals:

Transistor–transistor logic (TTL) is a widespread logic representation in electronics where a voltage

line V_{TTL} is considered to be a logical 0 for $V_{TTL} \lesssim 2V$ and a logical 1 otherwise. It enables hardware switches controlling for example laser power or long-wavelength radiation emission to be switched on and off with nanosecond accuracy which is sufficient to implement the PulseSequence framework detailed in Section 5.5.2 for running quantum experiments. It can also be used as a trigger signal for real-time control of complex devices. A device such as a camera or an oscilloscope may for example be configured on a slow-CPU channel but the device activation can be triggered by a TTL signal controlled by a real-time channel. Two ARTIQ TTL cards were integrated into the experiments, one with BNC and the other with SMA connectors. Each card has four input and four output channels.

In order to lock the laser power by modulating the AOM drive signal amplitude to the photodiode measuring the 369 nm beam leaving the vacuum system, an ADC and DDS card were installed together to form a PID locking mechanism with a fast 1 MHz update rate.

This real-time hardware configuration is enough to trap ions and implement test-shuttling experiments. Quantum logic will require the AWG and a homemade low-noise and high-stability current generator developed in the research group by Harry Godwin. This real-time hardware update will only be capable of testing static quantum circuits compiled and implemented as laid out in this thesis, but the FPGA ability to run python code in real time allows the implementation of conditional logic, feature described in Chapter 5.

7.2.3 DAC configuration

The Zotino cards employed in the logical qubit experiment, like most DAC systems employed by ion trapping experiments, do not have enough stability for reliable ion trapping: much of the noise both inherent to DAC itself and picked up by the cable when going to the chip needs to be removed by additional filtering. An easy and robust filter for DACs is the passive low pass filter which removes all voltage dynamics greater than the cut-off frequency of the filter. Such noise protection is a direct obstacle for ion shuttling which relies on the speed at which the DAC signals can be modified. A compromise needs to be found to both account for noise removal and fast shuttling operations.

All DAC channels go through a custom-built filter box⁵ containing a two-stage low pass filter with a frequency cutoff of 72kHz allowing a rise time of 10 μ s. The box receives as input the three 32 DAC channel cables, and outputs two DD-50 connector cables via easily re-configurable jumper wires. The noise accumulated between this filter and the chip is handled by the back PCB as described in 7.1.4.

⁵Designed by Dr Sam Hile

7.2.4 Experimental operating system

Software architecture and automation

The ARTIQ framework is a toolbox capable of handling both small and large experimental setups. In a basic case the user runs one experiment which calls all required devices, saves data in an array which is plotted in real time for feedback but is properly analyzed later. This method, although efficient, does not include full-time information monitoring, as the experiment, usually a scan, only covers the data specifically requested for the occasion. A classic experiment is centered on the scan to be taken and does not consider secondary values such as for example the presence of an ion in the trap or the laser wavelengths. These pieces of necessary information are manually handled by the user, who must trap ions prior to the scan as well as adjust the lasers and all other parameters necessary to run the experiment itself. Full automation of a quantum computing experiment, including choosing which experiment to set up and run, as well as analyzing the results, is possible with ARTIQ but not straightforward. Automating an experiment is time consuming but increases the experiment reliability and productivity. It also allows continuous operation in the unfortunate case of a worldwide pandemic.

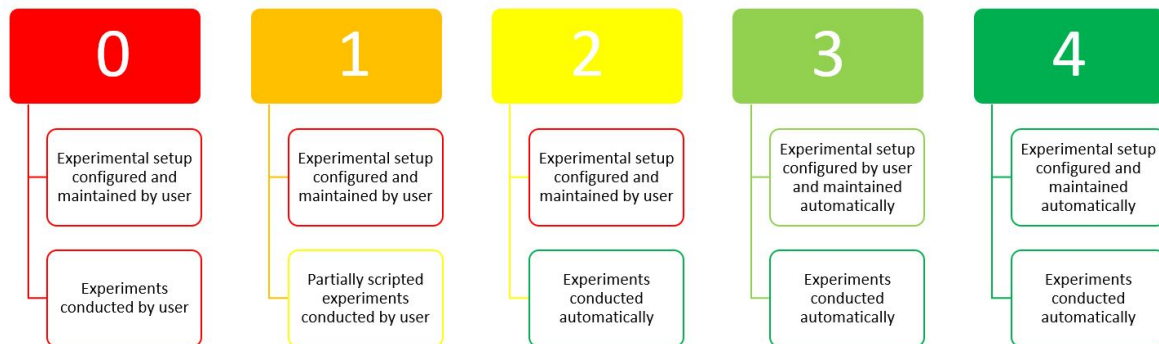


Figure 7.8: Automation levels for physics experiments. Following the syntax used for self-driving cars, those levels define mainly the level of controls given to the computer of preparing, maintaining, conducting and choosing experiments. The levels range from fully manual experiments where the user intervention is constant to a fully automated experiments, where the user input is only required once for the global experiment goal to achieve. Remote control of an experiment is only feasible from level 3 where only occasional presence is required.

The idea of having an “automatic” experimental organization can be argued to be impossible because there is always a need for the user to be manually involved in the experimental setup, especially in the case of trapped-ion technology. The answer is actually more nuanced than a dichotomy between manual and automatic experiments, as a very similar problem was studied in the industry regarding self-driving cars. A classification was created by SAE International for autonomous cars with levels called SAE levels: 0 being fully manual and 5

being fully automated. A similar table such as that shown in Figure 7.8 could be used regarding automation in an experiment, decomposing the problem into the experimental setup and conducting the experiments themselves. The priority in this apparatus was given to automating the experiments which only focus on controlling the main sensors and signal generators. Maintaining the setup in a workable state involves control and monitoring of any secondary devices necessary to trap and maintain ions such as the laser control, laser shutters, temperature monitoring, trap RF control and pressure monitoring. The final step is to start the system and calibrate it automatically, involving complex mechanical controls such as automatic laser beam alignment, camera focus control and automatic fiber coupling. Although not yet implemented, some tools toward full automation will be presented.

Automation in the logical apparatus

As mentioned above the software organization was focused on monitoring and operating the experiment in software assuming some initial preparations were realized in person, such as aligning the laser beam to the chip or coupling from the main laser heads with the right amount of power into the optical fibers.

The experimental code is decomposed into three layers:

- **passive monitoring:** multiple pipelines running in parallel individually monitoring devices. In case of a lost connection or a need for changing the monitoring parameters those experiments can easily be stopped and restarted without cascading errors.
- **active monitoring:** ARTIQ code reading existing measurements from the passive monitoring experiments via datasets and acting on them by updating its own datasets or triggering experiments. As an example the camera code and the atomic oven power supply code are active: the camera code assesses if there is a trapped ion and updates the corresponding dataset. The atomic oven code runs on a loop but does not activate the power supply if there is already a trapped ion.
- **experiment:** These experiment files represent the experimental code run by the user and can rely on active monitoring to take longer and more complex measurements. Following the previous example, assuming an ion stays in the trap for ten minutes in average, then by checking the active datasets for confirming the presence of an ion before each measurement, the code can either take the measurement and move on or wait until the atomic oven power supply activates, trap an ion and stop. Thus even with a ten-minutes lifetime, experiments taking hours can be run.

The jump from level 3 to level 4 automation as described Figure 7.8 would involve both more hardware control and the integration of quantum logic calibration as detailed in Chapter 3, with experimental calibration code capable of both fast-but-basic verification and slow-but-standard calibration. More hardware control does not require a reorganization of the three previously mentioned software layers but simply a larger number of active monitoring experiments running. Efficient quantum logic calibration however will involve the design of a fourth layer, the scheduler, which will decide which experiments to run and in which order. Each calibration experiment in the third layer would need to be normalized to include the verification and calibration functions.

7.3 Atomic oven setup

An atomic oven is a vital element required for trapping as it provides the neutral atomic flux that will be used to trap the ion with. This section describes the limitations that previous designs in the research group were having and their significance for the logical qubit apparatus in particular. The design steps for a more efficient and useful atomic oven is then presented with its calibration tests.

7.3.1 Trapping on a surface chip

As described in Chapter 2, ions can be trapped by ionizing neutral ytterbium atoms with a two-photons absorption process at the trapping position. The technology used to react neutral atoms is called an atomic oven. An atomic oven is a tube containing the block of ytterbium oriented toward the trapping position. It is activated by heating the tube to a temperature high enough for vapor to be emitted. The main issue presented by atomic ovens is deposits of ytterbium atoms on the electrodes: they react with the laser beam, become electrically charged and progressively deteriorate trapping capability. Surface chips are considerably more susceptible to such phenomena as atomic ovens require pairs of antagonistic conditions:

- The ytterbium atom flux must reach the trapping position but it must not reach the chip
- The ytterbium atom flux must be dense enough to trap quickly and reliably but sparse enough to avoid collisions with already trapped ions

While the second pair of conditions are achieved via control of the heat applied to the ovens the first pair of conditions have two different design solutions: The first one is to emit the ytterbium atom flux orthogonal to the chip via a loading hole designed in a region of the surface chip [165]. This integration of a micro-fabricated

atomic oven in the chip itself is a complex design task but would enable clear scalability features and avoids all contact between the ytterbium flux and the electrodes. An alternative approach is to position the oven parallel to the chip and engineer the flux to be ballistic, as summarized in Figure 7.9. By positioning the oven slightly below the chip level [157, Chapter 5.5.3], the flux will both reach the ion height and not make contact with the chip.

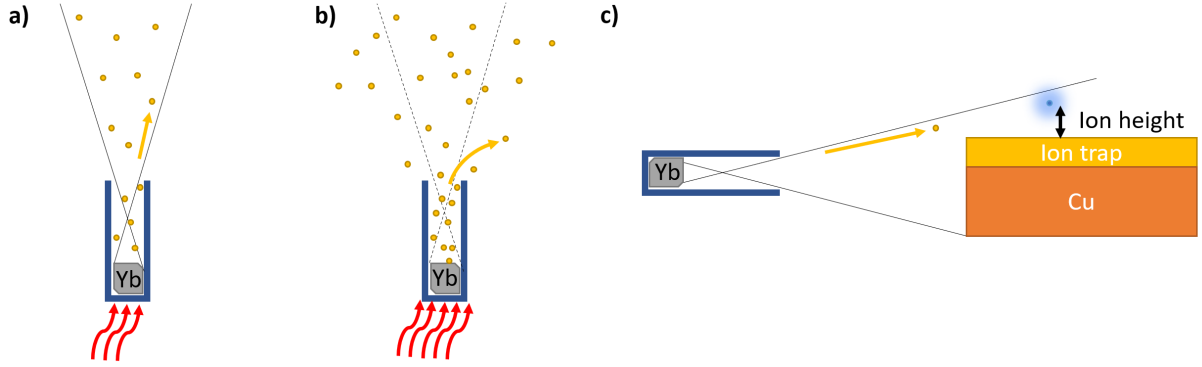


Figure 7.9: Different atomic oven behaviors and their impact on trapping. a) Molecular effusion model : each ytterbium atom can be approximated flying in straight line without interacting with other ytterbium atoms. The atomic flux dispersion is fully determined by the atomic oven dimensions. b) Hydrodynamic flow : the flux becomes broader because of collisions between the ytterbium atoms. The flux density decreases and the flight direction from the ytterbium atoms may change after leaving the oven. For the same oven geometry, transition between these two regimes is possible by increasing the vapour pressure via increasing the oven temperature. c) The oven is positioned below the chip to shield atoms from being deposited on the surface. Shielding is only possible if the oven are in the molecular effusion model.

At low enough temperatures and in a UHV environment, the emitted vapor will follow the regime of molecular effusion where no collisions are considered: the emitted atoms will have a ballistic behavior. To characterize the molecular effusion regime the mean free path λ of the emitted vapor has to be compared to the atomic oven diameter as well as its length [166]. λ is calculated by adapting equation 7.1 in using $r_{Yb} = 2.42 \times 10^{-10}$ m and by taking the pressure value to be the vapor pressure produced by the heated atomic oven and defined as [167]:

$$\log(p_{vapour}(T)) = 14.117 - \frac{8111}{T} - 1.0859 \log(T) \quad (7.4)$$

where $p_{vapour}(T)$ is in Pa and the constants are given for ytterbium atoms specifically. An estimation of both the vapor pressure and the mean free path for ytterbium ovens is displayed in Figure 7.10 with the point of interest being 10^{-2} mmHg or 1.333 Pa, which is a recognized threshold for considering the oven being activated [166]. As shown in Figure 7.10, for temperatures at which an ion can be trapped in the apparatus, the mean free path appears to be longer than the diameter of the oven, indicating that molecular effusion actually takes place. However the dimensions of the atomic oven, in the same order of magnitude as the estimated value of the mean

free path, translates into a bigger dispersion radius at the oven output.

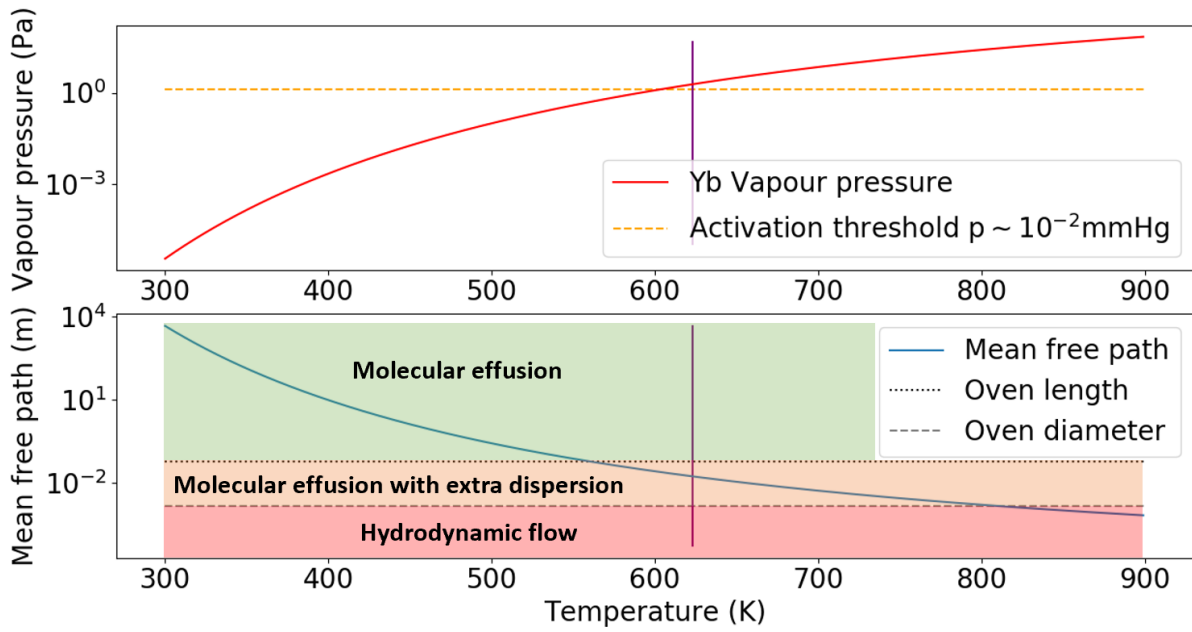


Figure 7.10: (top) Ytterbium vapour pressure against the temperature applied to the atomic oven. (bottom) Mean free path of ytterbium vapour against the oven temperature. We know that the activation threshold for trapping is $\sim 10^{-2}$ mmHg. From the top graph we this translates to a temperature of 620 K which is confirmed by the experiment. Using the bottom graph we can see that a temperature of 620 K corresponds to a mean free path smaller than the oven length and greater than the oven diameter. This size of mean free path places us in the ballistic regime but with additional dispersion.

7.3.2 Design of a new atomic oven

A new generation of atomic oven was designed with the double objective of monitoring the temperature for fast ramping cycles and increasing the ytterbium precision to maximize the chance of trapping emitted ytterbium in one arm of the X-junction surface chip and minimize the probability of collisions with previously-trapped ions in the other arms.

The new design shown Figure 7.11 uses a stainless steel tube clamped on one side of a type K thermocouple which is the most appropriate sensor from 300 K to 800 K. The heating is provided by a Nichrome wire surrounding the stainless steel tube and clamped to copper wires. The current flow in this design does not travel through the thermocouple and thus avoids offset measurements [168]. Being clamped inside the steel tube, the thermocouple is touching or in close proximity with the ytterbium material. The final ovens built into the system had a stainless steel tube of length $L = 6$ cm and tube diameter, $D = 1.5$ mm.

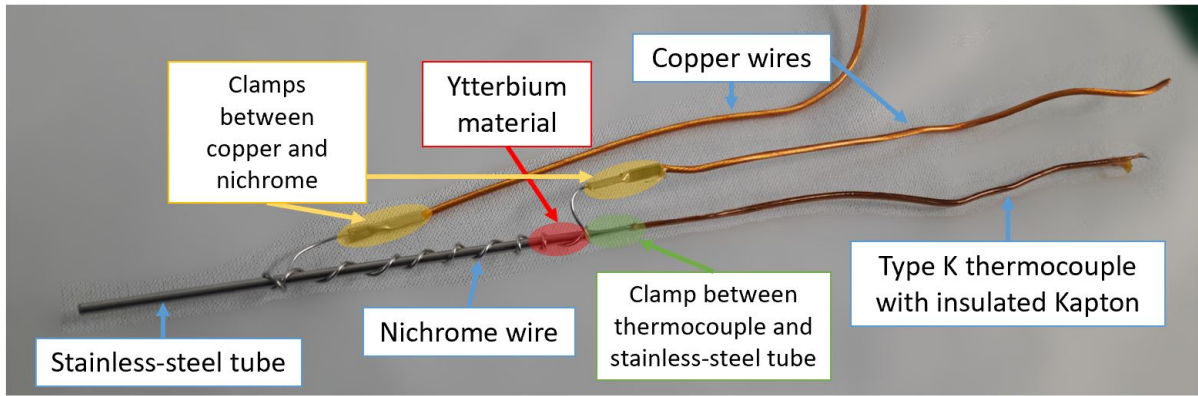


Figure 7.11: Final design of the atomic oven used in the experiment. The stainless steel tube (Left) is connected via clamping to the thermocouple wire. The tube is wrapped with chromium wire fixed in place via the thermocouple clamping. The Ytterbium material is inserted at the bottom of the oven, entering in contact with the thermocouple. All control and monitoring wires (Right) connect to a single DA-15 connector

7.3.3 The fluorescence test

The fluorescence test is a fundamental experiment to validate the integrity of an atomic oven setup. We use only the first of the two lasers required for trapping. The neutral ytterbium beam will fluoresce if illuminated at the wavelength resonant with the $^1S_0 \leftrightarrow ^1P_1$ transition.

In the logical qubit apparatus, the atomic oven design and the oven integration in its mounting structure were tested in a bell jar. The 399 nm laser beam was sent at a 0 degree angle directly into the stainless steel tube to maximize the fluorescence visibility. Such an angle shifts the absorption frequency due to the Doppler effect and needs to be compensated by red detuning the laser away from the known fluorescence frequency [54]. As seen in Figure 7.12 the tests were both successful so the complete atomic oven setup was integrated into the logical qubit apparatus.

7.3.4 Design of a new atomic oven mount

In order to reduce heat dissipation and noise coupling from the atomic oven ground to the RF and DC signals ground, the atomic oven mount hosting all the atomic ovens was decoupled electrically and thermally from the vacuum system. The material selected for this decoupling was Macor because of its excellent thermal and electrical properties and its UHV compatibility. The atomic oven is connected to the vacuum system via a DA-15 connector flange. Each oven uses two channels for the thermocouple reading and two channels for the heating circuit. Since there were four ovens to connect to the vacuum system, each with four channels, it was decided to merge together all the grounding cables for the heating circuits. This allowed us to connect four

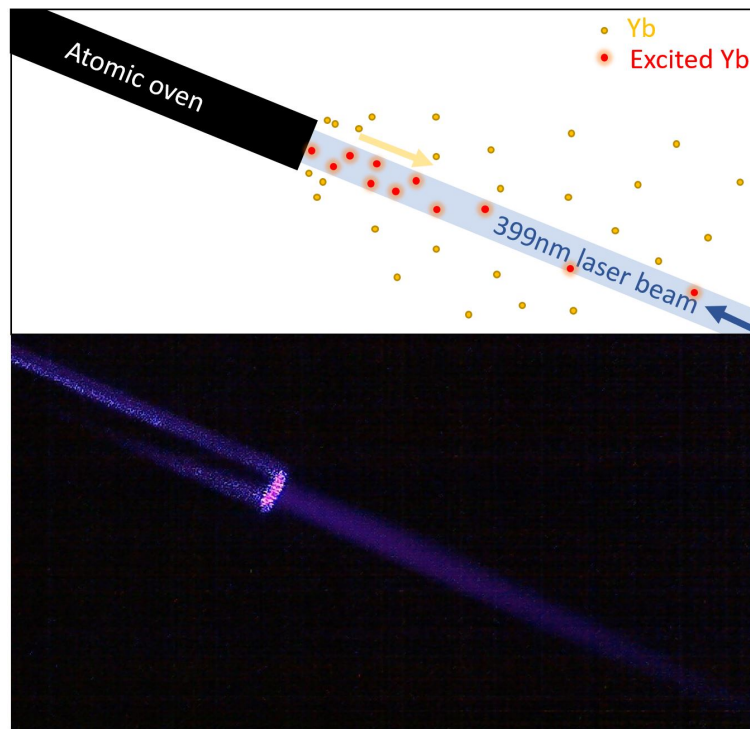


Figure 7.12: Fluorescence experiment diagram (top) and experimental observation (bottom) with a camera. The stainless steel tube can be seen partially illuminated by the 399 nm laser beam entering it. When the oven temperature goes above the fluorescence threshold, ytterbium atomic flux is emitted in enough quantities to visualize fluorescence along the shared path between its trajectory and the laser beam.

independent atomic ovens with twelve connections.

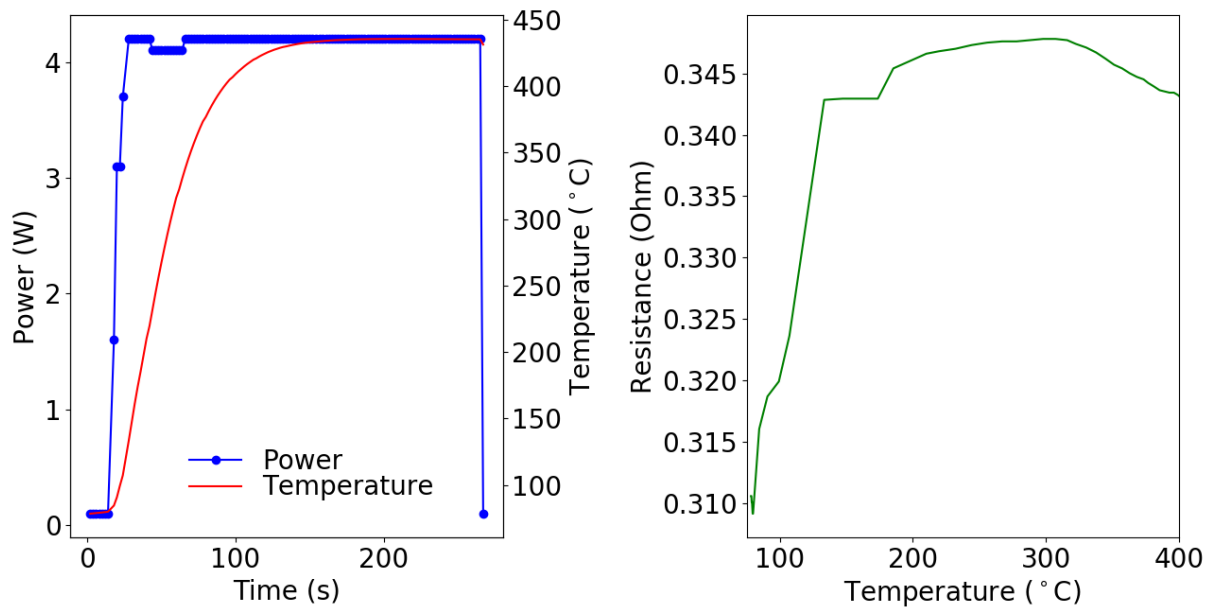


Figure 7.13: Atomic oven characterization. (Left) Temperatures above 400 °C are reached at 4 W of electrical power. (Right) The high atomic oven resistivity is essentially due to the Nichrome wire. Small changes due to the heat load are measured but stabilize for trapping operations at 400 °C.

Electronic testing of one of the atomic oven is displayed in Figure 7.13 and indicates a 4W power consumption to reach above 400 K, which is three times lower than existing ovens used in the research group but still two times higher than stainless steel ovens opened on the side [168]. Improvements on the temperature/power efficiency can be achieved by better thermal contact between the Nichrome wire and the stainless steel.

7.3.5 Movable atomic oven structure

The ability to control the position and orientation of the atomic oven would allow us to maximize the neutral atom flux sent at the selected trapping position. The current system was designed and assembled by Alex Owens. First of all the oven position flexibility is assured by connecting the Macor element holding the oven mount to a movable bellows. The control over the position was implemented via length-adjustable rods bolted across the two sides of the movable bellow in a “hexapod” configuration.

Alignment of the atomic oven is illustrated in Figure 7.14 by adjustments of the adjustable rods, across multiple fluorescence tests for keeping the pressure below a fixed level. The difference between a roughly aligned oven before baking and a fluorescence adjusted calibration was observed to be around two orders of magnitude. It is a usual rule of thumb to attempt trapping runs at temperatures below the value at which fluorescence from the neutral atom beam becomes visible. Adjusting the oven orientation brought down the target temperature for trapping runs from near 450 °C to 350 °C, value roughly estimated in 7.3.1 to be the activation temperature of the oven. The efficiency of the oven position is therefore concluded to be maximally optimized.

7.3.6 Atomic oven control

The DA-15 connector from the vacuum flange connecting to the heating elements and the power supply needs to be manually mapped to a thermocouple reader module and a power supply. The setup was organized as represented in Figure 7.15, with 2 USB connectors to the experimental computer for driving devices, a DA-15 connector to connect to the vacuum chamber connector via a single cable, and 2 input current supply connectors from the power supply. The atomic oven controller contains the temperature data logger and a relay switch connected to an arduino module, connecting the chosen oven to the power supply line. The power supply itself is controlled via a serial connection to the computer and the three devices are incorporated in ARTIQ.

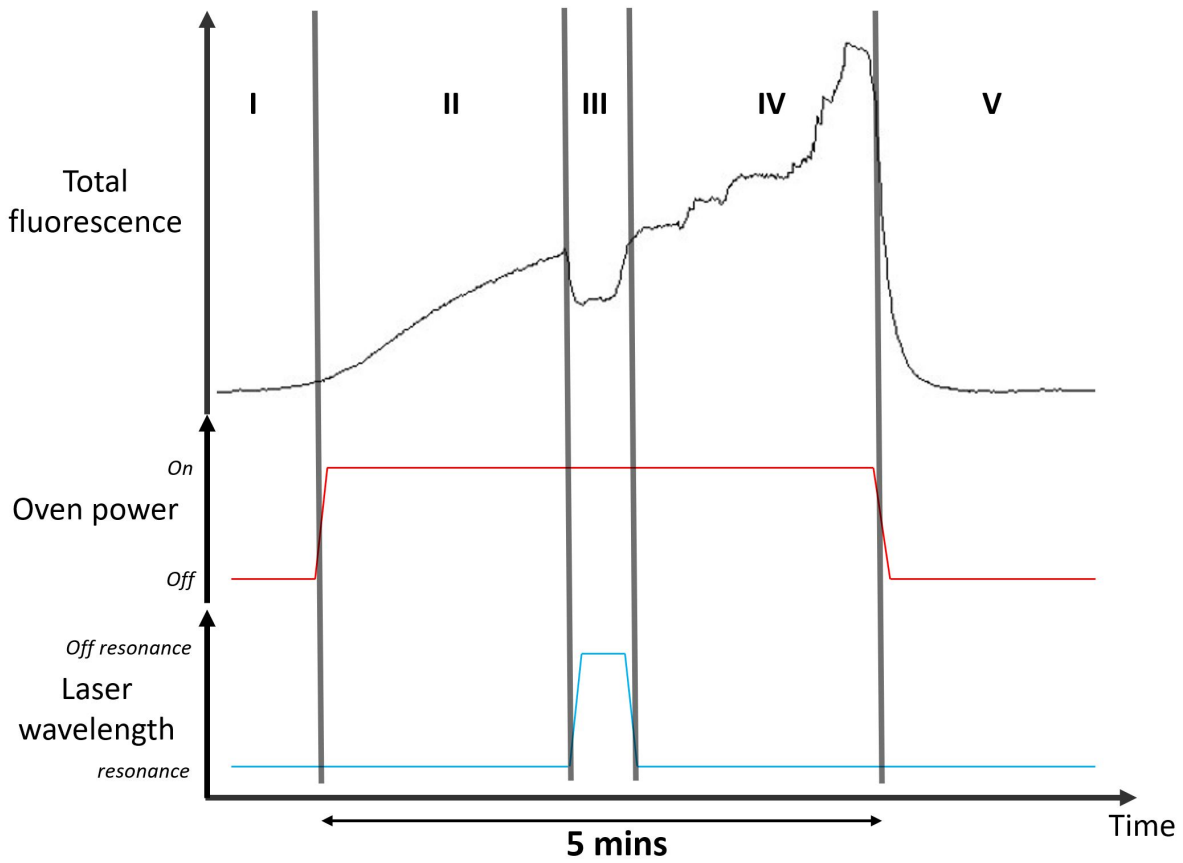


Figure 7.14: Atomic oven position adjustment via fluorescence measurement. The fluorescence was measured using a fluorescence test at the trapping position taken by the EMCCD camera. The y (top) scale is the total pixel sum on the image (arbitrary units) and the x scale is time. The middle and bottom y scales indicate the oven and laser settings respectively. The oven being set at high power causes an increase in both ytterbium fluorescence and light noise from the oven being set above 400 °C (II) compared to the noise floor at rest (I). The first gap in brightness (III) corresponds to the 399 nm laser shifted in frequency away from the fluorescence wavelength to assess the presence of ytterbium fluorescence among the light noise. The subsequent increase in brightness (IV) comes from the manual adjustment of the adjustable rods on the oven mounts. The final decay (V) corresponds to switching off the power applied to the atomic oven at the end of the fluorescence test.

7.4 Laser setup

This section will focus on the laser setup used by the logical qubit apparatus, describe the hardware, the optical layout and the strategies deployed toward more reliable trapping. The description will not focus on the ability to trap barium nor ytterbium 171 as those tasks have not been achieved at the time of writing the thesis.

7.4.1 Laser hardware in IQT

The research group shares laser resources between the different projects for cost and size efficiency. Each project either traps $^{174}\text{Yb}^+$ or $^{171}\text{Yb}^+$, and thus needs three lasers emitting at 369 nm, 399 nm and 935 nm as

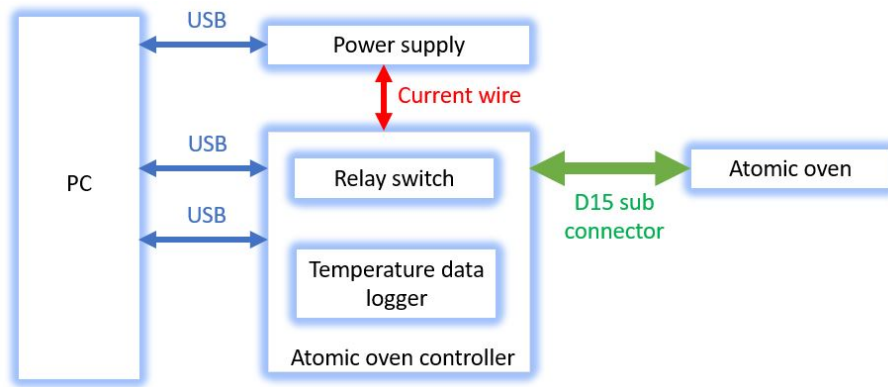


Figure 7.15: Diagram of the atomic oven controller box. The power supply was not integrated into the atomic oven controller, but interfaces the controller via the computer. A single cable is required to connect the atomic oven controller to the atomic oven flange.

described in Chapter 2. Some projects require a 638 nm laser for repumping after a collision event. Sharing lasers is possible using high powered laser units which are sampled via glass windows or beam-splitters and sent to an experiment table via fiber optic. Using fiber optics to connect the laser head and experiment allows us to treat experimental and laser issues independently. If there are issues with a laser we can recouple the fiber to a different laser without touching the experimental set-up. The experiment is at the time of writing connected by optical fiber to a source of 15mW 369 nm light. To produce the 369 nm light, we start with a Continuous Wave (CW) diode laser of 532 nm light, which is fed to a Ti-sapphire crystal to generate 739 nm light. The 739 nm light is converted by a doubling cavity into the desired 369 nm light. The other lasers are also high power CW diode lasers, with 250 μ W from the 399 nm laser and 6 mW of 935 nm laser⁶.

All lasers are locked in frequency by a central Laser controller described in Section 6.1. The 399 nm and 935 nm are locked with a PID loop between the wavemeter reading and the piezo actuator controller. The 369 nm laser is also locked, before the doubling cavity. Some of the 739 nm beam is fed through an external optical cavity which was built by Dr Thomas Navickas. This cavity uses low thermal expansion mirrors and is located in a vacuum system for maximum stability. It has a measured linewidth of 6.2 MHz [169]. The laser control setup is stable enough to trap reliably over at least 24 hours without manually interacting with the 369 nm laser. The laser wavelengths in the logic apparatus are stable enough to trap reliably for at least 24 hours, therefore this wavelength stability could be a useful benchmark when building similar experiments. Accurately measuring the wavelength however is limited by the accuracy of the commercial wavemeter⁷. The wavemeter used in the research group is rated with a 60 MHz absolute accuracy and a 2 MHz relative accuracy. This wavemeter is calibrated several times a week to a rubidium vapour cell via saturated absorption spectroscopy, although

⁶both are Toptica DL pro units

⁷High Finesse WS7-60 Series

a more regular calibration is planned and described in 6.1. A continuous measurement across two hours was conducted (Figure 7.16) with a 0.56 second interval sampling time using the HighFinesse Longterm software to identify any measurable drifts that didn't halt trapping operations. The measurement was conducted on all three lasers although stability is most important for the two UV lasers [169, Chapter 1]. The Allan deviation plot for all lasers remains below the 2 MHz accuracy threshold because of the wavemeter so no further analysis can be conducted.

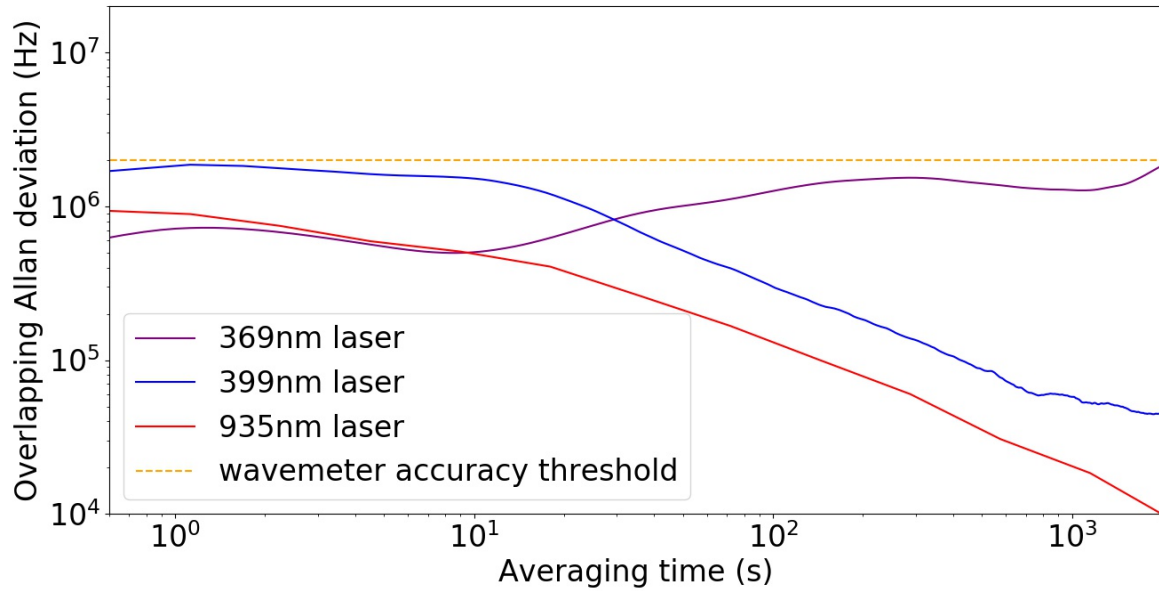


Figure 7.16: Allan Deviation plot for the lasers frequencies via a 2 hours continuous measurement of the wavelength. The measurements are taken at 1.8 Hz for the 399 and 935 lasers and 8 Hz for the 369 nm laser, although this measurement was taken using the 739 nm laser head i.e. before the laser goes through the doubling cavity. The wavelengths are converted to frequency in software. The drift in calibration is only visible on the 369 nm laser because the 399 nm and 935 nm lasers are locked to the wavemeter value itself. The relative sensitivity of the wavemeter being 2 MHz, we interpret the measured fluctuation of the wavelengths as noise.

7.4.2 Optical setup

The optical setup is represented in Figure 7.17. It starts with the couplers converting the laser beam from the optical fibers to colimated free space beams, followed by an iris for laser safety and free space optics and translation stages which send the beams to the trap.

The 369 nm first goes through an acousto-optic-modulator⁸ (AOM) double-pass setup. An AOM converts a laser beam of frequency f_0 into multiple beams of order n whose frequency is modulated by the driving signal f and is equal to $f_{\pm n} = f_0 \pm nf$. By adjusting the power and frequency of the RF signal sent to the AOM, we

⁸Isomet 1206C-833

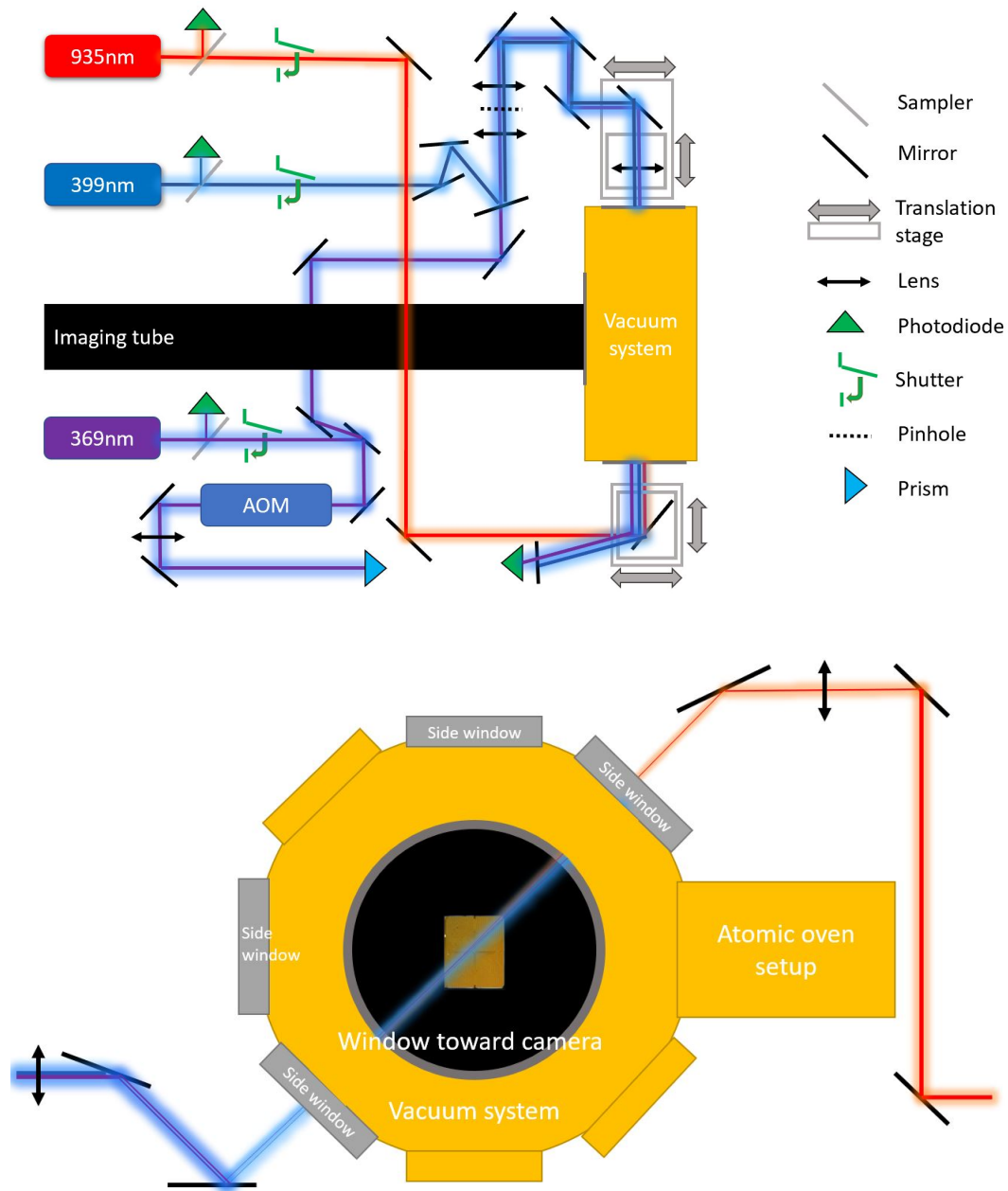


Figure 7.17: Optical setup of the laser beams sent to the vacuum system. (Top) Birds-eye view of the optical setup showing the paths each laser takes from its coupler to the vacuum chamber. All lasers are positioned vertically above the chip via translation stages and UV laser beams can also translate horizontally. (Bottom) Side-view of the vacuum system illustrating the 45° angle orientation of the incoming laser beams to the chip. For clarity, output beams are not represented.

can precisely control the amplitude and frequency of an order n beam. However each n -order beam exits the AOM with an angle which depends on the frequency modulation. This angle dependency can be removed by sending the beam back into the AOM and extracting the generated n -order beam a second time. This “double pass” setup means the angle of the second pass can be compensated by the angle of the first pass [170, 171], while maintaining frequency control. In our apparatus the double pass is configured to extract the positive first

order beam $n = 1$, therefore the 369 nm beam frequency f_{dp} after the double pass is written as $f_{dp} = f_0 + 2f$, where f_0 is the frequency of the original beam that enters the AOM, and f is the driving signal applied to the AOM.

The 369 nm and 399 nm beam are co-aligned via a narrow 369 nm filter⁹, which lets through the 369 nm beam but entirely reflects the 399 nm laser beam. The two beams then go through a telescope setup which cleans the Gaussian mode using a pinhole at the focus point of the telescope and increases the beam diameter. The diameter of the UV beams above the chip has to be smaller than the ion height to avoid light scatter off the chip, which would reduce the ion visibility and charge the dielectric layers of the chip via the photoelectric effect. The relation between the beam diameter ω_1 at the focal length f at a wavelength λ with an input colimated beam diameter ω_0 is as follow:

$$\omega_1 = \frac{f\lambda}{\omega_0\pi} \quad (7.5)$$

The telescope magnification M is the ratio of the two lenses' focal length. The two lenses we started to use had the same focal length, therefore the beams arrived at the chip with a magnification of 1. The beam waist at the chip can be changed by switching these lenses. In this setup the 369 nm and 399 nm beams was expanded by a factor of 4 to reduce the beam waist at the chip to below 100 μm .

The final step consists of a two axis translation stage with micrometer adjustment screws, upon which is mounted a 15 cm lens followed by a mirror for angle adjustment. The 369 nm and 399 nm beams are sent at an angle of 45° compared to the RF electrode axis to enable Doppler cooling of the ion as described in Chapter 2. As the chip's X junction is positioned orthogonal to the optical table it was decided to send the 369 nm beam from underneath the vacuum system via a mirror placed on the table. The 935 nm beam is not filtered nor combined with any other beams and is sent at 45° in the opposite direction to the coaligned 369 nm and 399 nm beams. The 369 nm laser beam intensity is locked: a photodiode is installed in the return path of the 369 nm beam with a 369 nm filter similar to the one used to combine the 369 nm and 399 nm beams, to factor out any non-369 nm light. This photodiode feeds a PID loop which controls the RF signal applied to the AOM. Thus the second order beam going in the vacuum system and cooling the ion can be maintained at the desired power.

⁹Semrock FF01-370/36-25

7.5 Imaging setup

7.5.1 Photon collection setup

As briefly mentioned in Chapter 2, Ytterbium ions can be detected when emitting fluorescence in the $^2S_{1/2}$ - $^2P_{1/2}$ transition. We will define now the fluorescence rate and evaluate each of its components [172]. A measured ion fluorescence rate is determined by:

$$F_{rate} = \eta\gamma P \quad (7.6)$$

where η is the photon collection efficiency of the imaging system, $\gamma = 2\pi \times 19.6$ MHz is the $^2S_{1/2}$ - $^2P_{1/2}$ decay rate and P is the population of the excited state. We define s to be the saturation parameter calculated from the laser power p via the following relation:

$$s = \frac{p}{3p_{sat}} \frac{\gamma^2}{\gamma^2 + 4\Delta^2} \quad (7.7)$$

where p_{sat} is the saturation intensity and Δ is the laser detuning from the S-P transition. We also define s_0 as the saturation parameter for $\Delta = 0$. Equation 7.7 however ignores both the effect of power broadening, by considering p small, and the effect of the magnetic field and laser beam polarization on Zeeman broadening. The fluorescence rate for $^{174}\text{Yb}^+$ is then proportional to:

$$F_{rate} \propto \frac{s}{1 + (1 + \xi/2)s} \quad (7.8)$$

where ξ is the repumping factor due to the 935 nm repumping, which is commonly chosen to be 0.1 for this isotope. The main conclusion is that for $^{174}\text{Yb}^+$ the fluorescence rate never decreases due to an increase in the laser power, and that detuning the laser frequency only slows down the rate at which fluorescence increases with laser power.

We will now consider the photon collection efficiency which depends on the optical efficiency of collecting light from the ion, the transmission coefficients of the optics at the wavelength considered and finally the quantum efficiency (QE) of the light sensor. The setup reuses most concepts from [173, Chapter 6].

Fluorescence emitted by the ion is at a random orientation. The first requirement of the imaging setup is to be able to collect light from the widest solid angle at the ion position. The main constraint is to keep the focal length longer than the ion-window length so that the imaging setup can be installed outside the vacuum. This is commonly accomplished using triplet lenses, but a more optimized setup using custom-made optics has been

able to increase collection efficiency considerably [174]. The current triplet lens¹⁰ has a numerical aperture (NA) of 0.293, which translates to a collection efficiency of:

$$\eta_{collection} = \frac{1 - \cos(\arcsin(NA))}{2} = 2.2\% \quad (7.9)$$

The collected light is focused by an additional lens onto the light sensor. The window and the doublet lens are both estimated to have a 97% transmission efficiency, along the triplet lens is rated with $T = 97\%$. The fluorescence is recorded by an EMCCD camera¹¹ using 13 μm pixels and a QE for 369 nm $\eta_{EMCCD} = 30\%$. In order to distinguish between multiple ions and to obtain enough information to align the laser beams on the chip, the image at the ion position needs to be magnified. Considering the ion-ion distance in a Coulomb crystal is typically in the order of 10 μm , a minimum magnification factor of 2 is needed.

The total magnification is the product of the triplet lens magnification, which is a fixed value, and the second lens magnification which is the main adjustable parameter. The second lens focal length and position was adjusted to match a DC electrode diameter of 200 μm to 320 pixels, corresponding to a magnification of around 22.

The camera is situated in an optical black box to filter out light noise. In close proximity to the camera a flipper mechanism¹² holding a narrow bandwidth Semrock 369 nm filter can quickly switch the imaging setup between either laser alignment or ion trapping. The 369 nm filter has 90% transmission efficiency. A second equivalent flipper mechanism can also redirect the light to a Photomultiplier tube (PMT)¹³ as a backup detection system. This PMT has a count sensitivity of $4.7 \times 10^5 \text{ s}^{-1} \cdot \text{pW}^{-1}$ which is equivalent to a QE of $\eta_{PMT} = 25\%$.

The total photon collection efficiency is therefore estimated to be $\eta = \eta_{collection} \times \eta_{optics} \times \eta_{EMCCD} = 0.54\%$ for the EMCCD setup, and $\eta = 0.44\%$ for the PMT.

7.5.2 Qubit readout fidelity estimation

The presented imaging system has not yet been used to trap $^{171}\text{Yb}^+$ but an estimation of fidelity of the quantum state readout, steps described in Section 2.3.5 of Chapter 2, can still be calculated. As indicated in Section 4.5.3, we require a minimum readout fidelity of 99%.

The model of the ion fluorescence readout by Acton et al. [175] requires the fluorescence rate of $^{171}\text{Yb}^+$ arriving on the sensor, the noise associated to the sensor and the leakage parameters associated to the ion

¹⁰Special Optics 54-17-29-369

¹¹Andor iXon Ultra 888

¹²New Focus 8892

¹³Hamamatsu Photon counting head H10682-110

species. The fluorescence rate for $^{171}\text{Yb}^+$ ions, assuming an optimal angle θ_o between the applied magnetic field and the laser beam electric field of $\theta_o = \arccos\left(\frac{1}{\sqrt{3}}\right)$, is defined as [172]:

$$F_{rate} = \frac{\gamma}{36} \frac{\Omega^2}{\Delta^2 + \left(\frac{\Gamma_{171}}{2}\right)^2} \quad (7.10)$$

$$\left(\frac{\Gamma_{171}}{2}\right)^2 = \frac{\gamma^2}{2} + \frac{1}{6} \left(\frac{\Omega^4}{36\delta_B^2} + 4\delta_B^2 \right) \quad (7.11)$$

With δ_B the Zeeman splitting already defined in equation 2.59 and Ω the Rabi frequency of the laser beam linked to the linewidth γ and the saturation parameter at zero detuning s_0 by [172]:

$$\Omega = \sqrt{\frac{s_0}{2}} \gamma \quad (7.12)$$

For $s_0 = 2$, $\Delta = 2\pi \times 2$ MHz and $\delta_B = 2\pi \times 10$ MHz, we calculate $F_{rate} = 7.57 \times 10^6$ photons/s.

In an ideal case, both the bright state and the dark state configurations are closed loops and increasing the time readout directly increases the readout fidelity. However, small leakages from the bright state to the dark state and vice versa, are characterized by the two rates per scattered photon $C_{b \rightarrow d}$ and $C_{d \rightarrow b}$ [172], dependent on δ_B and Ω via Γ_{171} :

$$C_{b \rightarrow d} = \frac{2}{3} \left(\frac{\Gamma_{171}}{2\Delta_{HFP}} \right)^2 \left(1 - \frac{F_{rate}}{\gamma} \right) = 2.40 \times 10^{-5} \quad (7.13)$$

$$C_{d \rightarrow b} = 2 \left(\frac{\Gamma_{171}}{2(\Delta_{HFP} + \Delta_{HFS})} \right)^2 = 1.56 \times 10^{-6} \quad (7.14)$$

where $\Delta_{HFP} = 2\pi \times 2.1$ GHz is the hyperfine splitting of the $^2P_{1/2}$ state and $\Delta_{HFS} = 2\pi \times 12.64$ GHz is the hyperfine splitting of the $^2S_{1/2}$ state. The Poisson statistics briefly introduced in Section 2.3.5 is, in reality, coupled to the exponential decay due to state leakage. The probability distributions for the bright and dark states can be written as [175]:

$$P_{bright}(n, t) = \frac{e^{-(1+C_{b \rightarrow d}/\eta)\lambda_0(t)} \lambda_0(t)^n}{n!} + \frac{C_{b \rightarrow d}/\eta}{(1 + C_{b \rightarrow d}/\eta)^{n+1}} P(n+1, (1 + C_{b \rightarrow d}/\eta)\lambda_0(t)) \quad (7.15)$$

$$P_{dark}(n, t) = e^{-(1+C_{d \rightarrow b}/\eta)\lambda_0(t)/\eta} \left[\delta_n + \frac{C_{d \rightarrow b}/\eta}{(1 - C_{d \rightarrow b}/\eta)^{n+1}} P(n+1, (1 - C_{d \rightarrow b}/\eta)\lambda_0(t)) \right] \quad (7.16)$$

where δ_n is the single-argument Kronecker delta, $P(a, x)$ the incomplete Gamma function and $\lambda_0(t)$ the mean photon value readable by the sensor during the detection time t and defined as $\lambda_0(t) = F_{rate} \times t$.

The noise sources on the PMT, the dark noise and background noise, are Poissonian, therefore the measured dark and bright state distributions can be simulated by applying a convolution with the distribution representing

the sensor noise rate. Noise rates for this PMT are estimated to be 640 s^{-1} . The best threshold n to distinguish dark states and bright states is calculated by finding the minimum detection error and is recalculated for each measurement time. A set of detection error values is accumulated against different detection times to find the best measurement time and the highest readout fidelity accessible.

Simulating the noise for an EMCCD camera requires a more involved model, for which a characterization is outside the scope of this thesis. The key points to understand the EMCCD behavior are summarized from Burrell [176] and Tubbs [177]:

- The pixels convert the incoming photons into electrons, which are amplified via an electron-multiplier process which has its own Poissonian distribution dependent on the gain setting.
- The electron distribution before readout is a weighted sum over the incoming photon count, between the incoming photon distribution and the electron-multiplier distribution.
- The readout noise, which is convoluted with the electron distribution to form the final readout measurement, is Gaussian and needs to be experimentally characterized.
- In addition to the dark count rate, negligible for these devices when using the electron-multiplier feature, the EMCCD cameras have a clock-induced charge noise which is independent of the measurement time, and needs to be characterized experimentally for each pixel.
- Injecting leakage errors between bright and dark states in camera readout simulations, while not practical to implement analytically, can be simplified in time-bin simulations where the leakage probability is “digitized” over the time bins.

The simulations for the PMT are displayed in Figure 7.18: The optimum readout fidelity of $F_{1qb} = 99.2\%$ is acceptable for QEC schemes. This simulation confirms the current imaging-optical setup is capable of quantum state measurements satisfying the requirement to run QEC protocols. An experimental setup, with lower transmission path efficiency but with calcium ions, has already been demonstrated, using an EMCCD camera, to reach a quantum state readout fidelity of 99.99%, while being able to spatially distinguishing multiple qubits [178]. Quantum state readout has also been demonstrated with ytterbium in the research group, with the apparatus mentioned in Chapter 3 having a transmission efficiency five times lower than the logical apparatus [179], with a 96.8% fidelity for single ion readout with a PMT [55] and 93.4% fidelity for two qubit readout with an EMCCD camera [86].

Finally, quantum state detection fidelity decreases when reading simultaneously the state of two qubits with

the PMT, as the Poissonian distribution of mean value $2\lambda_0$ has more overlap with a distribution of mean value λ_0 than the distribution of a dark state. This reduction is overcome with shuttling: by splitting the ion crystal before a measurement, two ions can be read sequentially with a fidelity of $F_{2qb} = F_{1qb}^2 = 98.4\%$ for a PMT.

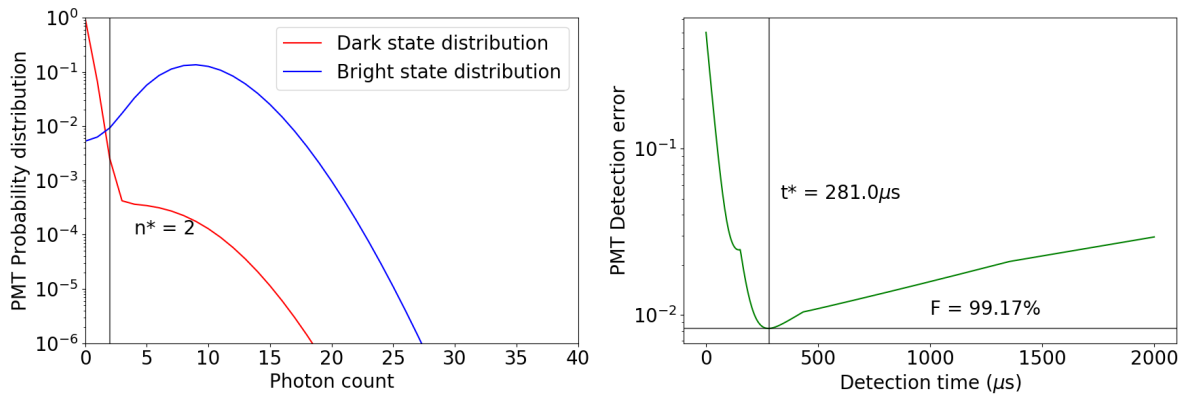


Figure 7.18: Readout error simulation for PMT setup. (Left) Histogram of the probability distributions for the dark (Red) and bright (Blue) state during the detection time. The value n^* represents the optimal photon count which minimizes the readout error when using the threshold readout method. (Right) Optimal detection error against detection time. The minimum error value e , represented by the maximum readout fidelity $F = 1 - e$ and the corresponding optimal detection time t^* .

7.5.3 Software ion analysis

The camera is mostly controlled by the Andor Solis software for initial trapping and for laser alignment. However for the purpose of automation it was more appropriate to write a driver to directly obtain and analyze the images with ARTIQ, attaining level 2 in the automation scale shown in Figure 7.8. The camera driver was implemented both in LabVIEW and with python wrappers¹⁴

Once the first ions were trapped with the full camera resolution, a compact signal for automatic monitoring was used by taking a 5×5 super-pixel image where each super-pixel accumulates the fluorescence of a 12×12 physical pixels area as shown in Figure 7.19. The binning size was chosen so that the center pixel of the image is the only one measuring the fluorescence emitted by the ion while the other pixels provide information about background noise as well as radial and axial excitation. Binning also reduces readout noise from the camera and thus increases the signal over noise (SNR) ratio. Once an ion is trapped the brightness is high enough compared to the background noise that we could implement a fluorescence threshold above which ARTIQ considers an ion to be trapped. Such a configuration is robust enough to reliably trap ions using software over a period of more than 24 hours to take long scan measurements of the secular frequencies.

¹⁴<https://pypi.org/project/andor/>

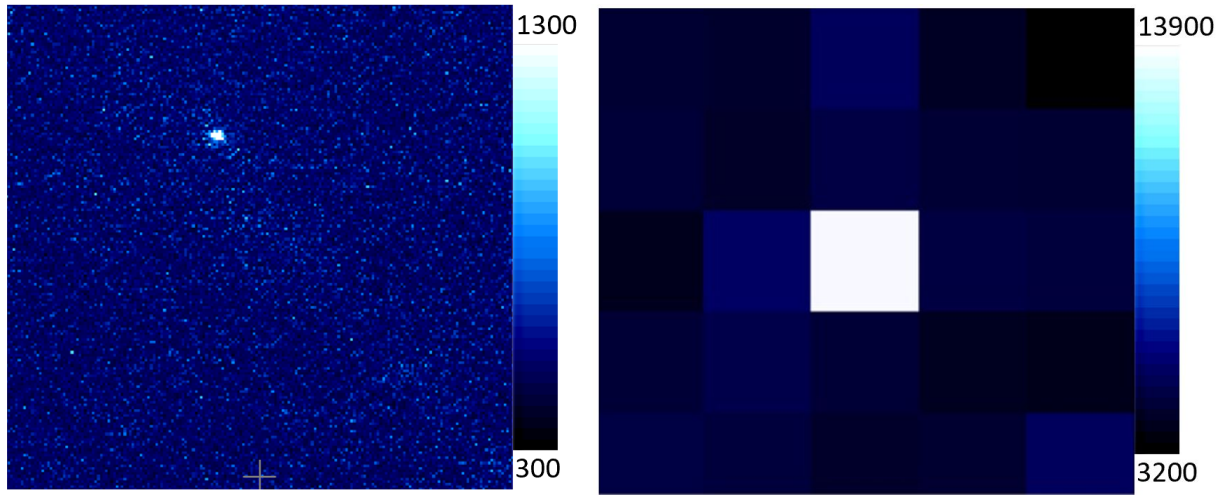


Figure 7.19: (Left) Picture at magnification of 22 ($0.6\mu\text{m}/\text{pixel}$) of a single trapped $^{174}\text{Yb}^+$ ion using the full resolution of the camera, with a SNR of 2. (Right) Picture of an ytterbium ion using a binning of 12 pixels, and an increased SNR of 4. Both pictures were taken with the same 300 ms exposure time.

A second automation module was written to distinguish between multiple ions, and, concurrent to measuring the fluorescence, obtain position information to handle imaging setup drifts and cover new experiments such as shuttling. This image recognition module, mainly based on the Hough circle detection algorithm [180], is implemented and shown in Figure 7.20.

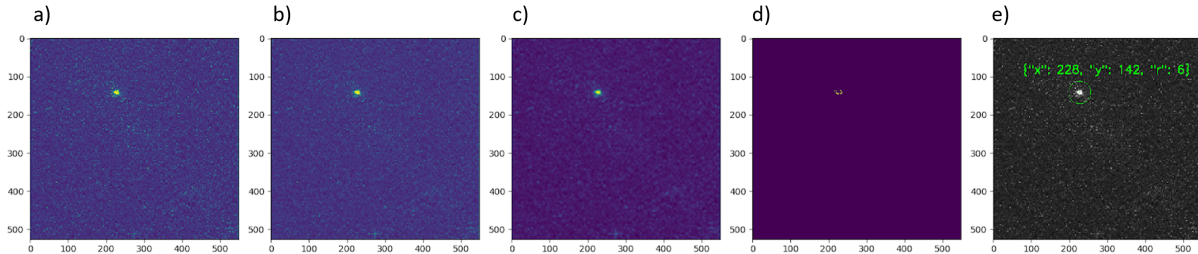


Figure 7.20: Ion recognition via Hough circle feature detection. The image taken by the camera (a) is cleaned by applying first a small blur (b) followed by a Non-local Means Denoising algorithm (c). The ion is transformed into a hollow circle by a Canny filtering (d) to increase the quality of the Hough algorithm. The recognized ion and its information are obtained (e) by adjusting initial parameters depending on the expected ion diameter.

7.5.4 Laser alignment protocol

The UV laser beams for ionization, cooling and readout must be focused and aligned to the trap location with a $\sim 10\mu\text{m}$ precision. The beam waists must also be smaller than the ion height because any scattering of UV light on the chip surface may charge the dielectrics layers via the photoelectric effect and add detrimental offsets to the trapping potentials [181]. Furthermore, if the 369 nm laser beam scatters off the chip near the trapping

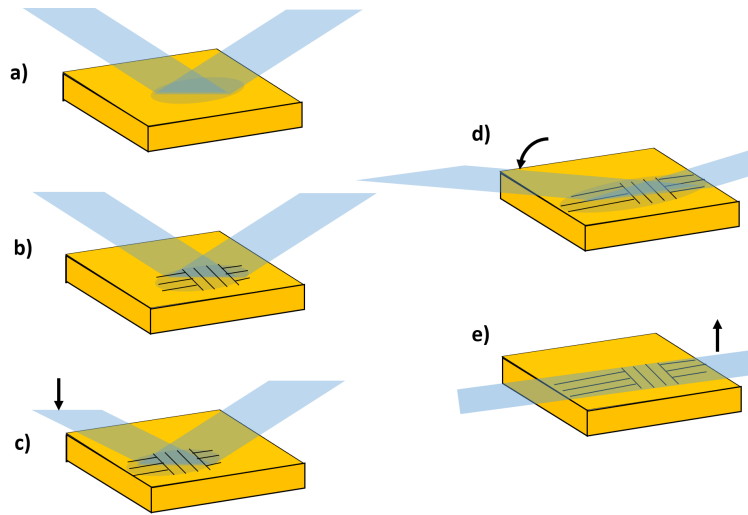


Figure 7.21: Diagram of the alignment procedure of a laser beam onto a surface chip. (a) The beam must be scattered off the chip so that light gets collected by the camera. (b) The camera is then adjusted in focus so that the chip electrodes can be seen clearly. The laser beam waist can be estimated and the laser position is known. The orientation of the beam is adjusted until it is parallel to the chip by repeatedly (c) moving the beam position toward the chip then (d) adjusting the beam orientation to get the maximum brightness on the camera. With small adjustments the operator should eventually notice a decrease in the maximum brightness that can be obtained. This means that the laser beam is clipping the chip (e) meaning that the laser beam is covering the chip as much as possible and is therefore parallel to it.

position then the scattering will appear on the camera and considerably reduce the signal over noise ratio when detecting a trapped ion. As the chip was designed with an ion height of $125\text{ }\mu\text{m}$, the 369 nm and 399 nm laser beam waist at the trapping region must be smaller than this value. The methods to reach such a laser beam waist are explained in Section 7.4.2.

The method for aligning each laser beam is illustrated in Figure 7.21 and consists of progressively clipping the laser beam on the chip by iteratively adjusting the laser beam incident angle. The aligned beam as seen by the camera is shown in Figure 7.22.

7.5.5 Software laser beam detection

Although less important than the algorithm in 7.5.3, the ability to automatically align the laser beams at the ion trapping position following the procedure shown in 7.5.4 would be a major step toward level 5 of automation as discussed in section 7.2.4.

Control can be implemented by replacing the manual travel micrometer head by a piezo-actuator controlled version¹⁵, allowing both manual and remote control of translation stages. These piezo actuators were tested on

¹⁵Thorlabs travel piezo inertia actuator

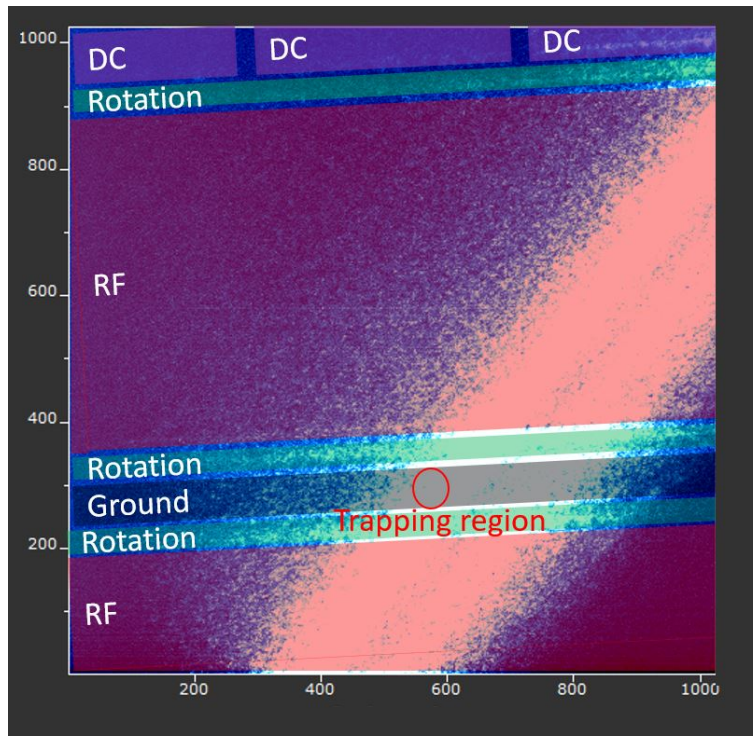


Figure 7.22: Picture of the UV laser beam overlaid onto the chip. False color and labels are added to provide better understanding of the chip position. The two beams in this photo are parallel to each other and not fully superposed. Scattering on the chip allows visualization of the beam position by identifying the electrodes of the chip. The top three DC electrodes combined with the ground and rotation electrodes surrounding the trapping position are enough to finely align the beam for trapping.

the z-axis control of the imaging tube which controls the focus: by bringing the focus onto the chip and recognizing the laser beam position, more piezo actuators would enable complete automation of the laser alignment. Therefore two image recognition algorithms are to be tested in relation to the remote control of the camera focus: one to identify the laser beam position and the other to recognize the camera focus.

The beam is identified on the camera by fitting the camera image $B(x,y)$ with a Gaussian beam of width ω , direction $y = ax + b$, amplitude C and background noise N :

$$B(x,y) = C \times \exp \left(-\frac{(a \cdot x - y + b)^2}{2\omega^2(a^2 + 1)} \right) + N \quad (7.17)$$

The focus can be deduced from a scan, such as the one shown in Figure 7.23. The focus will correspond to the imaging tube position with the smallest ω fit.

Another algorithm is still required to complete the identification of the laser beam position on the chip: recognizing the location on the chip that the camera is pointing at. A successful algorithm has not yet been implemented and is still being investigated. There are two main difficulties for implementing this feature: the

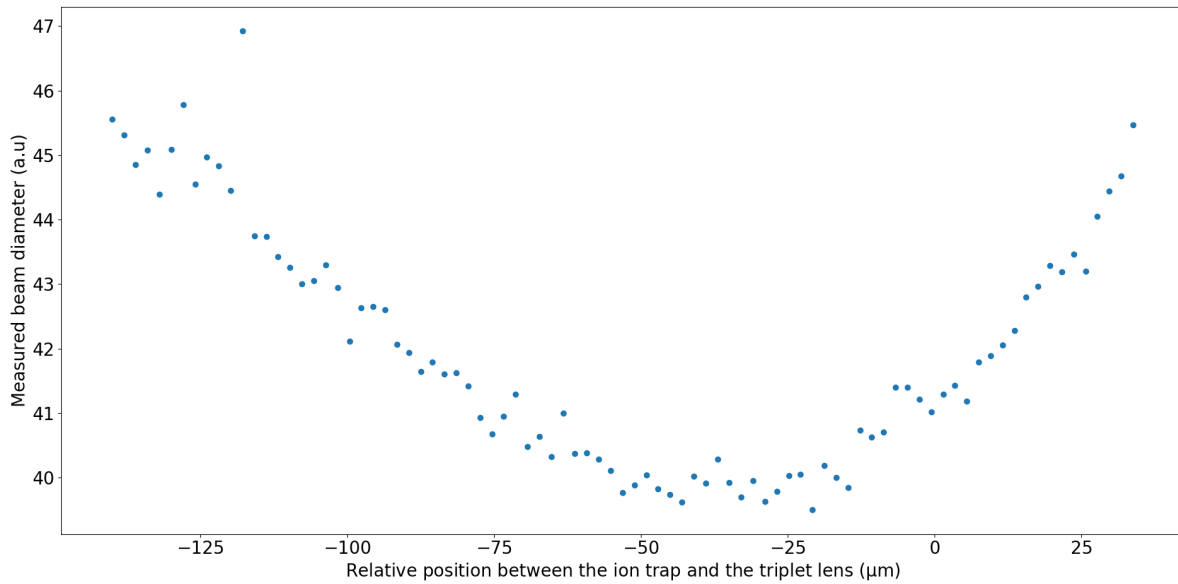


Figure 7.23: Automatic scan over the camera focus. (x-axis) The piezo actuator scans with a step size accurate to 20 nm, over the imaging tube position toward the chip. (y-axis) For each step, an image is taken by the camera and the laser beam width on the trap is fitted with equation 7.17. The focus is at the position for which the laser beam width fit is minimized.

first obstacle is identifying the chip features, which are mostly straight lines either parallel or at right angles to each other, is impacted by large amount of noise. A robust denoising procedure specifically dedicated to this issue would enable the use of the Hough line feature recognition algorithm. The second obstacle is that even if the features are recognized, this doesn't necessarily tell us which section of the chip the camera is looking at. Either a user input or a more extensive control over the camera position, to enable the detection of a unique feature such as the center of the X-junction, would allow the program to deduce the absolute position on the chip that the camera is pointing at.

Chapter 8

Scalable shuttling architecture

The sequence of experiments from initial trapping to shuttling implementations is now described. The aim of these experiments is to use the system presented in Chapter 7 to implement the QPU primitive operations from Chapter 2, incorporated with shuttling based instructions via the QCCD syntax introduced in Chapter 5. This setup can then be used to perform key building blocks of QEC codes mentioned in Chapter 4. As chips with current carrying wires have been delayed by fabrication challenges, we are focusing on the implementation of shuttling sequences useful for QCCD architectures. Each tool described in this chapter is made to be modular and reusable in order to speed up the implementation of low to medium scale quantum algorithms in the research group.

8.1 Trapping

This section is focused on initial trapping experiments and metrics measured to characterize the trap. Hardware limitations which will impact further experiments are detailed here, while new tools are presented to give the experimental user more control and feedback over the ion trap.

8.1.1 Ion lifetime

The software detection tools presented in Chapter 7 allow us to monitor the trapped ion and retrap in the eventual loss of an ion. The software tools also allow us, as illustrated in Figure 8.1, to record several metric pertaining to the performance of the ion trap. Since ion detection is based on a fluorescence threshold method, several successive ions may be falsely registered due to laser power or frequency fluctuations as well as laser

locking faults. Such duplicates can be identified with enough logging information from the apparatus.

The survival of ions in the trap is limited to less than five minutes and could be explained by the vacuum system's pressure. Ion lifetimes below a minute were omitted as they do not reflect the performance of the chip, but are rather due to other unfavorable conditions such as undetected laser power fluctuations and bad laser beam alignment. The measured pressure from the ion gauge of $1.5 \cdot 10^{-10}$ mbar gives via equation 7.2 an estimated lifetime of twelve minutes, which is slightly higher than the fitted decay time, however on the same order of magnitude. The monitoring can be improved by systematic calibration of the lasers in the laser control framework detailed in Section 6.1 which will remove the frequency fluctuations, and by also recording the laser powers.

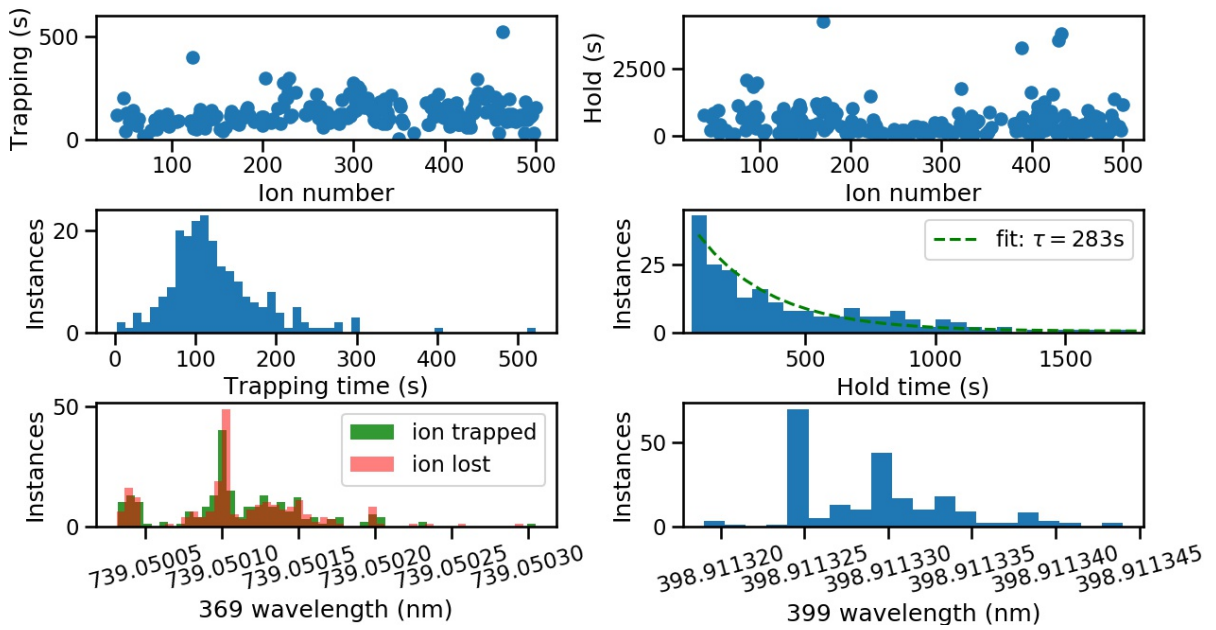


Figure 8.1: Ion statistic obtained from more than three hundred registered ions. The trapping time is mainly consistent but raises slowly due to the choice of slowly decreasing the 369 nm power. The ion lifetime has both unstable period and good trapping conditions period, due to drifts in the experimental calibration of the trapping conditions. The laser wavelengths (Bottom) are fluctuating due to drifts of the wavemeter's calibration. However the strong overlap of 369 nm wavelengths from trapped and lost ions suggest that frequency fluctuations do not account for the poor ion lifetimes.

The high pressure in the system is associated with a high collision rate, and therefore increases the likelihood of the ion falling in the $^2F_{7/2}$ state that can only be repumped with a 638 nm laser. A few short experiments with a 638 nm laser did not seem to increase the ion lifetime but would require further tests to confirm it. Increasing the RF trapping voltage, which provides a better radial confinement, did not have significant effect either. Activating the Helium cryogenic cooling is expected to both reduce heating due to the electrodes and reduce the pressure and should have a significant impact on the ion lifetime.

8.1.2 Secular frequency measurements

The radial and axial secular frequency determined in equations 2.10 and 2.16 from Chapter 2 were experimentally measured by adding an electric field oscillating near the estimated secular frequency. On resonance, the ion motion is amplified in the secular frequency's axis so that it becomes noticeable on the imaging system, and the spatial extent of the elongated ion is then recorded as illustrated on the left plot in Figure 8.2. The radial secular frequencies were stimulated with an RF signal sent via the in-vacuum PCB antenna. The axial secular frequencies were excited by driving an RF signal directly onto an on-chip DC electrode, since the in-vacuum antenna did not provide a sufficiently strong electric field component in the axis.

Radial secular frequency measurements are displayed in Figure 8.2: For every trapping RF voltage applied to the chip, a very strong signal can be measured at resonance. The linear fit matches the theory and provides an experimental validation of the ion radial secular frequency.

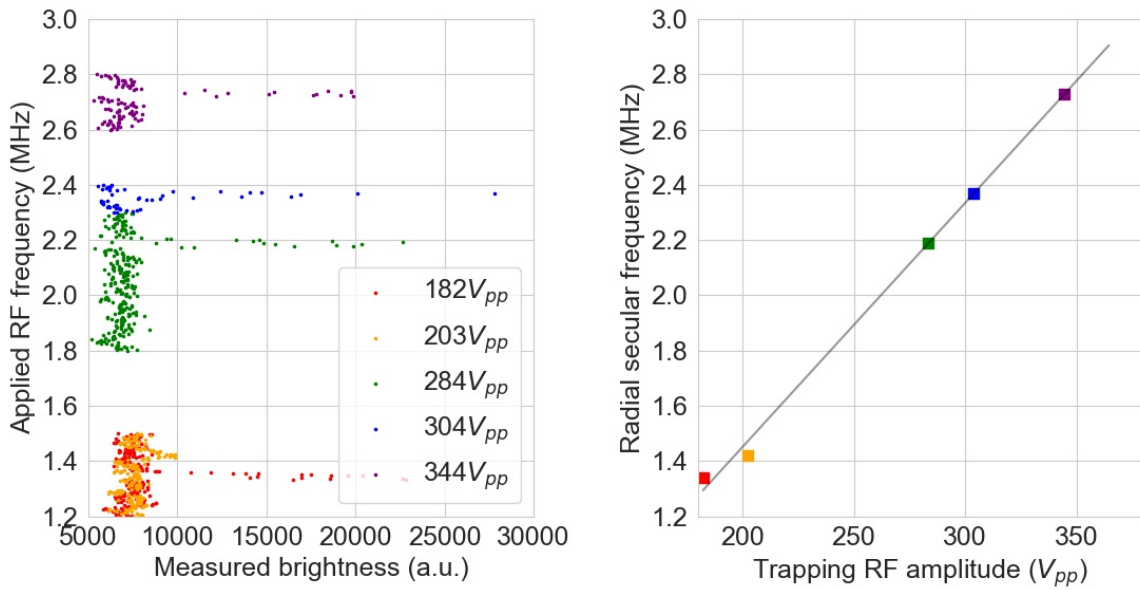


Figure 8.2: Secular frequency measurements against the RF voltage. (Left) Secular frequency measurements scans: an external RF field is applied (y axis) and the ion brightness is recorded (x axis). When the applied field matches the secular frequency, the ion motion is enhanced which translates into a brightness peak. (Colors) For each scan, a different AC voltage is applied on the RF electrodes, changing the radial secular frequency. (Right) The fits between the trapping RF amplitude applied on the RF electrodes and the resulting secular frequency confirms the expected linear relation.

8.1.3 Micromotion compensation

When the DC null is not located at the RF null the ion will be influenced by extrinsic micro-motion which can have detrimental effects on the ion line-shape, affecting the ion cooling rate, distorting the ion fluorescence and ultimately the ion lifetime. The two main causes are an straight static field E_{ext} or a phase difference between the two RF electrodes ϕ_{ac} . The phase difference is ruled out due to the geometry of the front PCB which was designed to have the same RF track length between the single input RF connector and the two wirebonding locations for the RF electrodes. The RF wavelength is more than twenty meters while the RF track length difference is estimated to be below a few millimeters.

The normalized absorption spectrum can be modeled [182] at low saturation by :

$$Pe'(\omega_{laser}) = \frac{1}{4} \sum_{n=-\infty}^{\infty} \frac{J_n^2(\beta)}{\left(\frac{\omega_{atom} - \omega_{laser} + n\Omega}{\gamma}\right)^2 + \frac{1}{4}} \quad (8.1)$$

where ω_{laser} is the laser frequency, ω_{atom} is the ion's transition frequency, J_n is the Bessel function of order n , γ the natural linewidth of the transition and β the effective micromotion parameter. It can be observed in Figure 8.3 that because Ω and γ have the same order of magnitude in this apparatus, both the frequency and amplitude of the maximum brightness peak are highly dependent on the amount of micro-motion which prevents further characterization of the system.

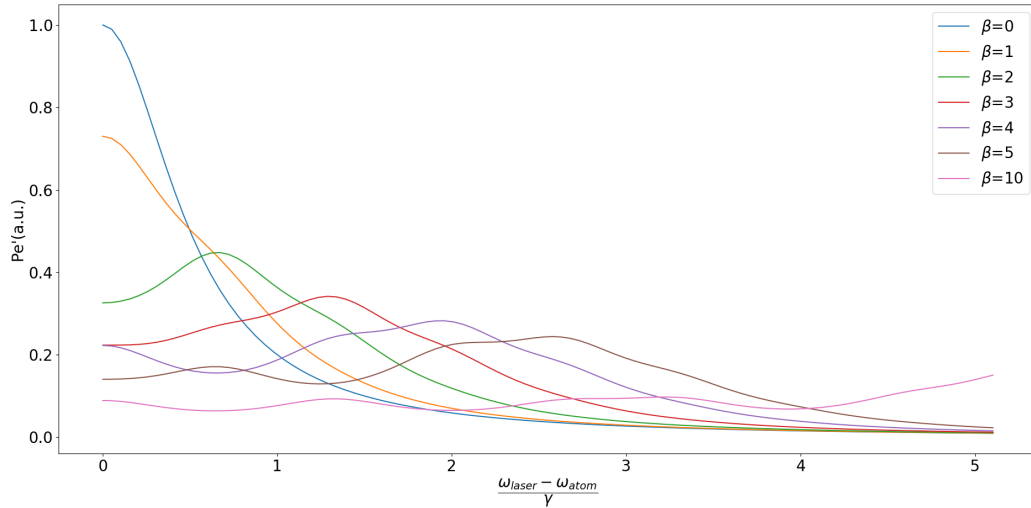


Figure 8.3: Micromotion impact on the linewidth spectrum. The simulation used the ratio from the apparatus $\frac{\Omega}{\gamma} = 0.66$ for different amounts of micromotion β .

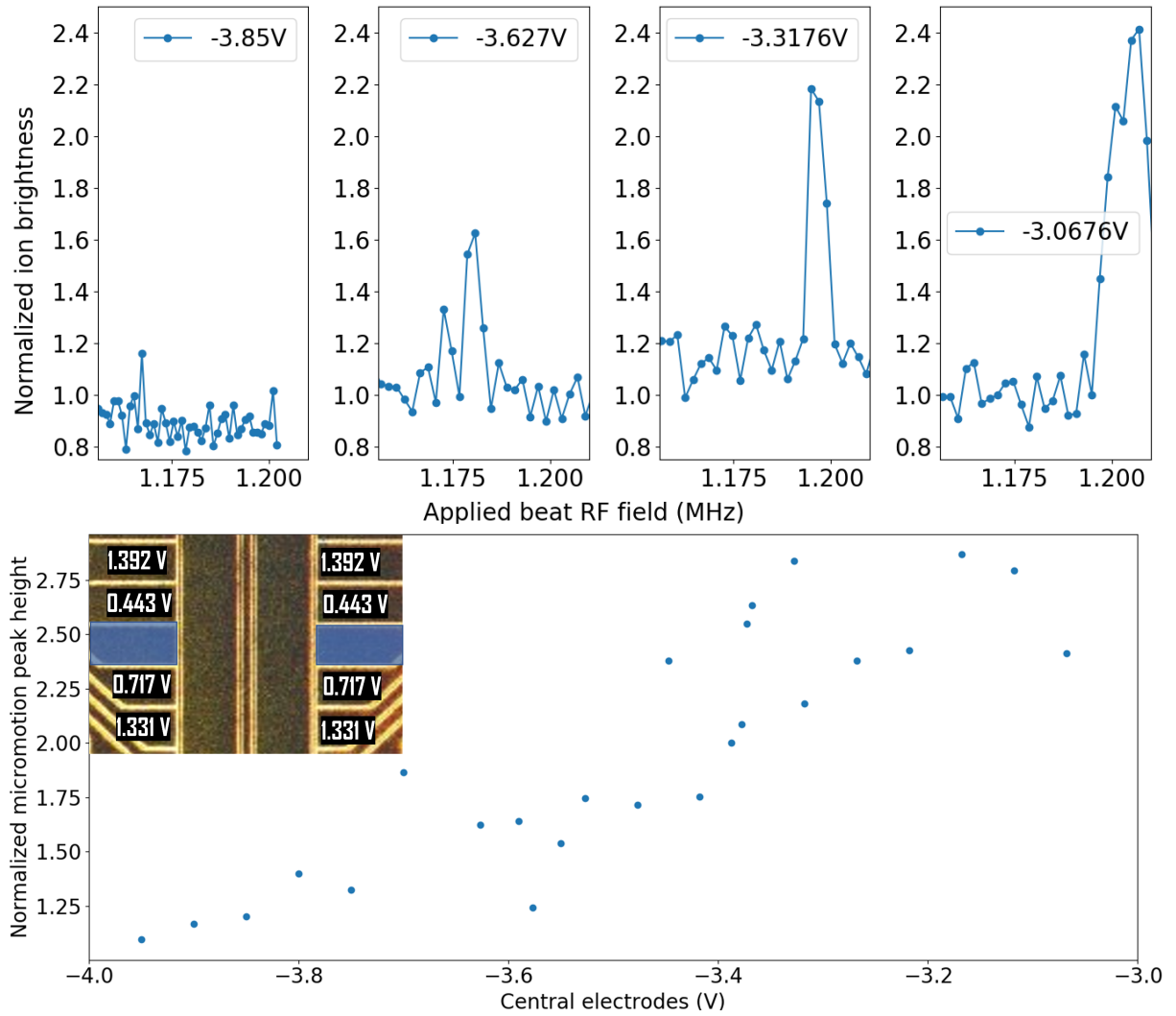


Figure 8.4: Micromotion compensation along the z axis. (Top) Scans of the external RF applied field (x axis) and resulting ion brightness (y axis). The scans are normalized to the ion brightness at rest. The center peaks of the parametric excitation are close to each other and not constantly at the same frequency, which is attributed to fluctuations of the RF trapping amplitude. (Bottom) Peak from the parametric excitation scans against the ion height, controlled by the voltage applied on the pair of central electrodes displayed in the chip representation in blue. The datapoints are normalized at each parametric excitation scan. The decrease in peak amplitude is considered as a decrease of the micromotion parameter. The optimal ion position is when the parametric excitation peaks disappear.

There are several techniques to minimize excessive micromotion in ion traps [183], including photon-correlation [182], sideband spectroscopy [184] and ultracold atoms probing [185]. Initial experiments to observe micromotion consisted in varying the amplitude of the field applied to the RF electrodes. Axial micromotion would result in visible ion displacement along the axial axis. The absence of any measured displacement indicated that the only possible micromotions was either due to phase mismatch, which was ruled out, or from micromotion in the ion height axis which would not be visible on the camera for such amplitude scans. Micromotion was then more accurately measured with parametric excitation [186], in which the RF electric field Ω applied to

the RF electrode is modulated with an external RF field $\Omega' = \Omega + \Delta$. When Δ is equal to the radial secular frequency and if the system is suffering from micromotion, then the ion's motion will be amplified, commonly called being “tickled”. Micromotion is compensated by adjusting the potentials applied to the DC electrodes, in order to minimize the observed tickle. Thus the tickle scan was run for different voltages applied to the central pair of electrodes and resulted in different ion heights.

The results of these scans are shown in Figure 8.4: by detuning the voltage applied on the pair of central electrodes which control the ion height, the tickle amplitude gradually decreases until micromotion becomes too weak for the ion tickle to be observed. The electrode voltage at which the micromotion peaks becomes indistinguishable from the ion fluorescence at rest represents the optimal ion height for which micromotion is minimized.

8.2 Ion position control on a surface chip

In parallel of the characterization work, the second category of experiments available to this chip is ion shuttling. As previously mentioned from Chapter 2, a shuttling waveform consist of applying in series a list of voltages, each step confining the ion in a given position of the trajectory. Successful shuttling consists of overcoming two challenges: simulating the waveform to be applied to the chip based on the knowledge of the chip geometry, and successfully applying the waveform with the required speed and accuracy, limited by the hardware.

Two important shuttling operations are considered in this section, as they represent key components for the QCCD syntax defined in Chapter 5 and by extension most QCCD architectures: the X-junction shuttling and the splitting/merging operations. Junction shuttling is the cornerstone technology to scale microfabricated traps on 2-D architectures, while splitting and merging enables individual ions to be brought together in a single ion crystal and execute entangling gates, and reciprocally to separate them after the gate is applied. Shuttling across a junction has been demonstrated for a T-junction [187], Y-junctions [165] and 3-D traps X-junctions [188, 189]. The surface X-junction sequence is much harder to simulate and the only working X-junction surface chip known to date has only shuttled around the corner of the junction but not through it [190].

This section will describe the developed simulation tools and hardware compilation steps regarding the implementation of shuttling on the logical qubit apparatus.

8.2.1 Simulations

DC trapping voltages for initial trapping and early shuttling were based on the House equations [46]. Solving ion position's with an arbitrary shaped ion trap for a set of DC and RF electrode voltages is a process involving the following steps [191]:

1. An electrostatic model of the DC electrodes is simulated. The potential of every electrodes n , at any point of the trap $\mathbf{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ and defined as $\phi_n(\mathbf{r})$, is recorded. The collection of these electric potential grids is named the basis potential. The total potential at a position $\mathbf{r}$ from the DC electrodes with an applied voltage V_n , is equal to $\sum_{n=1}^N \phi_n(\mathbf{r})V_n$.
2. The RF electric field is by nature not static, but the time-independent pseudopotential $\phi_{PS}(\mathbf{r})$ of the RF potential can be calculated by similarly evaluating the static potential of the RF electrode. The total potential $\phi(\mathbf{r})$ at any point $\mathbf{r}$ is defined as the sum of the RF pseudopotentials with the DC potentials:

$$\phi(\mathbf{r}) = \phi_{PS}(\mathbf{r}) + \sum_{n=1}^N \phi_n(\mathbf{r})V_n \quad (8.2)$$

3. Solving the ion position at $\mathbf{r}_0$ for a set of voltages $[V_0, \dots, V_N]$ consists in finding the minimum potential via enforcing the following equation:

$$\nabla \phi(\mathbf{r}_0) = 0 \quad (8.3)$$

The flexibility needed for simulating multiple positions for shuttling protocol, as well as the ability to quickly change electrode geometries to study multiple types of chips led to the development of a dedicated python module¹. An external python file defines multiple geometries over which the one of interest is chosen. The trapping potentials are solved at a given position by a minimizer algorithm based on configurable conditions, such as the secular frequency, residual stray fields and total sum of the squared voltages or the symmetry between electrode pairs. Those conditions can also be adjusted for each positions depending on the desired shuttling waveform. High numbers of shuttling sequences are simulated and tested across on the chip, thus requiring a formatting standard for shuttling waveform files. The standard used in the research group is a JSON based object, with mandatory fields for the voltages solutions and allows optional fields depending on the user need, and is described in Appendix B. Fast and simple turnaround between different sequences is made possible by having the python simulation module producing sequences with this syntax and the ARTIQ experiment extended to be able to read them.

¹developed by Dr Sam Hile

8.2.2 Prototype model implementation

The voltage sequence simulated in Section 8.2.1, or waveform, corresponds to a large array of data and only shuttles the ion along the requested trajectory. While these solutions allow us to minimize shuttling-induced heating, they cannot be scaled up due to the exponential size of the simulation as the number of electrodes increases. Furthermore, increasing the number of controlled electrodes increases the memory size taken by the shuttling sequence. Future chips with several X-junctions, and potentially thousands of electrodes cannot be simulated with this approach. Devising an architecture controlling the ion position by only using a local set of electrodes is required. This section considers a prototyping solution of implementing ion position control over larger distances via a concatenation of local sequences at the appropriate electrodes such as displayed in Figure 8.5: in this example local files containing only 5 pairs of electrodes are concatenated together to shuttle an ion across a trajectory that would require globally 9 pairs of electrodes. The calculation cost improvement can be quickly estimated by considering each pair of electrode as an additional dimension to minimize in the search space: the difference is therefore an exponential improvement of exponent 1/4.

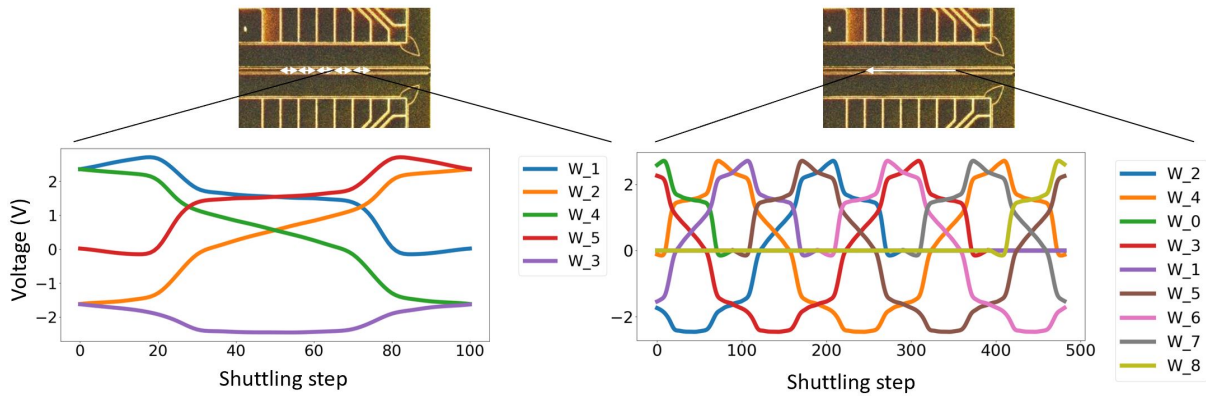


Figure 8.5: Example of concatenation of linear shuttling sequences. (Left) Local sequence determined by the simulations. Each curve represents a pair of electrode voltages when shuttling an ion across the two borders of the middle electrode. (Right) By requesting a longer linear shuttling sequence the local sequence is duplicated at the appropriate electrodes and the sequences over the different electrodes are concatenated together.

In order to enable such a scheme the files containing local sequences must be complemented by a file containing hardware information on the electrodes as illustrated Figure 8.6: Based on a description of the trap, the software decomposes a shuttling instruction into the necessary local sequences, assigns in each of them every voltage instructions to the correct electrodes in the trap, and finally concatenates it all into one ARTIQ instruction.

The concatenation requires from the local sequences that they share the same voltages over the concatenation point so that no heating or loss occurs between two regions. Local sequences however do not take into account a specific electrode shape on the chip: the deviation of the duplicated waveform with the required one may lead

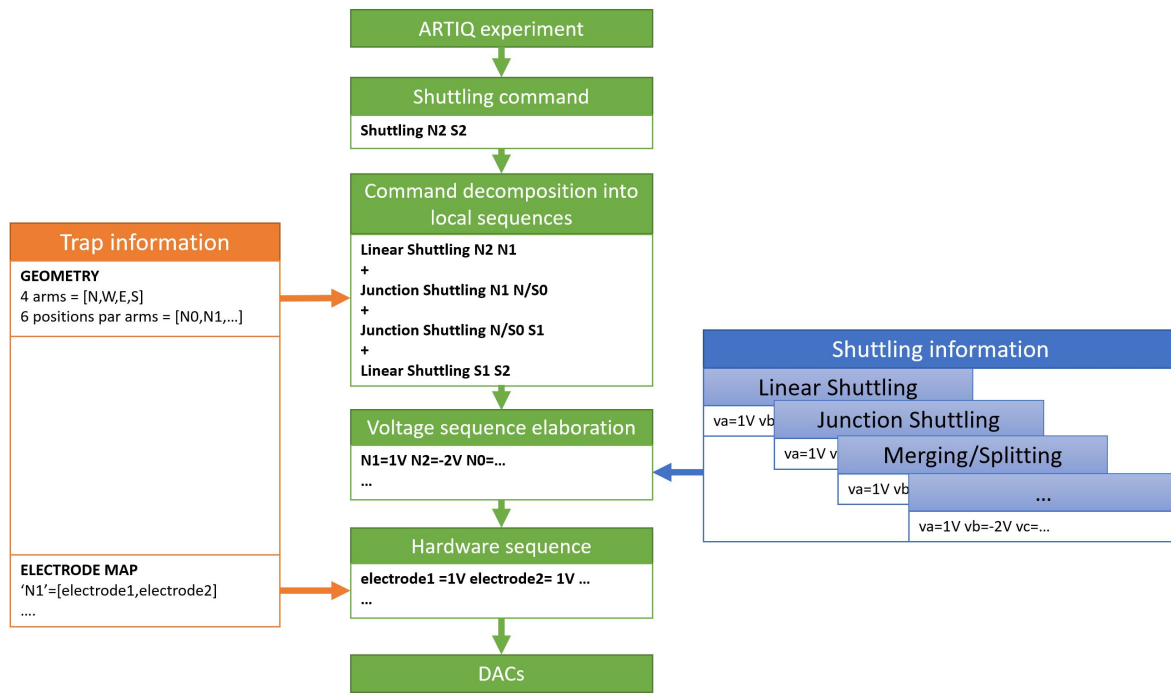


Figure 8.6: Sequence diagram of compiling a shuttling order across an micro fabricated chip, with the X-junction as an example. Sequences in green represent the steps taken by the ARTIQ software between first receiving a single shuttling command and its final hardware implementation with the DACs. The use of local sequences (Blue) combined with information of the trap (Orange) allows flexible reconfiguration and low memory usage.

to additional risks, such as ion loss and heating as well as significant drifts in the ion's position. Therefore the file containing the mapping of electrodes and their hardware address has to be extended to include correction parameters to take into account geometry variations.

In order to theoretically demonstrate the effectiveness of waveform compensation, a set of waveforms were simulated over one arm of the X-junction trap used in the project. As displayed in the concatenation example in Figure 8.5, there are five distinctive regions using five pairs of electrodes each. The simulated waveforms obtained for each specific regions are plotted in Figure 8.7 against the simulated central waveform which we will define as the standard one. Over the twenty waveforms considered nineteen can be reasonably compensated by a second order polynomial correction factor. The problematic waveform is in the closest region to the X-junction which has a stronger geometry difference. An experimental validation is still required, as implementing such corrections may require to take into account imperfections of the chip or impurities deposited on it. Higher order polynomial corrections may therefore be needed.

Waveform compensation is therefore validated by simulation over most regions on the chip, but first requires for each region an assessment of its validity. The volume of data manipulated is still however considerably reduced regardless of the addition of those factors: in the above case of linear shuttling on the arm, each region

encompasses five waveforms of a hundred points each, while a set of five second order polynomial factors only represents fifteen data points.

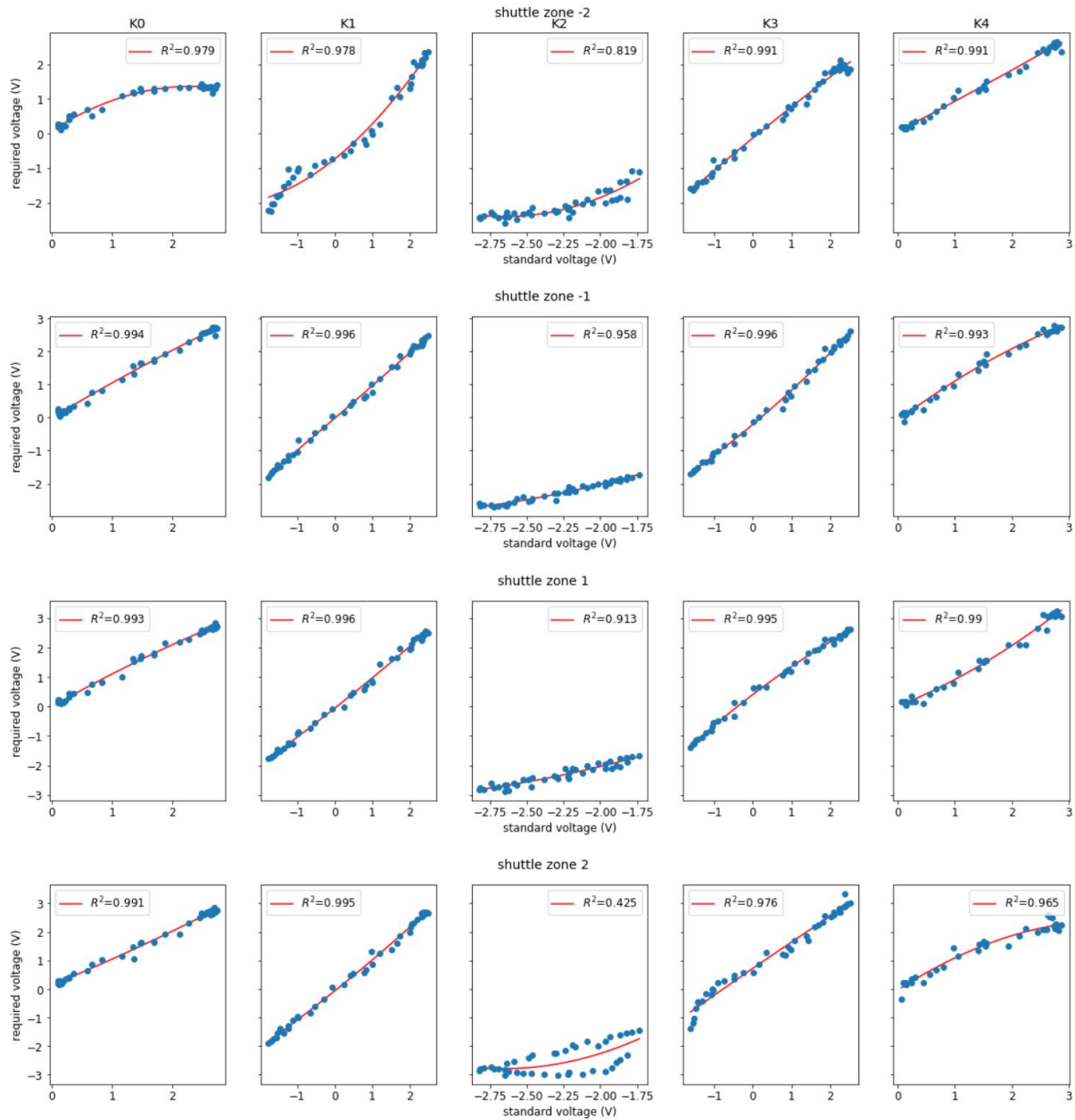


Figure 8.7: Waveform corrections between simulations of the generic and required shuttling sequence. Each row represents a shuttling region on the linear arm, with the region zero defining the standard waveform and thus not being plotted, while each column represents the shuttling waveform to apply to the local pair of electrode. Each graph displays the required simulated waveform (y axis) compared to the waveform simulated in the central zone 0 (x axis). The red curve is a second order polynomial fit. A good fit across all waveform of a region indicates that this region can be in theory implemented with the standard waveform in combination of the correction factor determined by the fit.

In conclusion a globally efficient shuttling can be implemented with small local sequences with a minimal added cost. The polynomial factors are at least an order of magnitude smaller than a local sequence. Recalling

the fact that a single X-junction includes around 80 DC electrodes, we can estimate that each added X-junction in a quantum computer processor will increase the required data size for shuttling by the equivalent of only one local sequence. This scaling of the data size is within reach of virtually every CPU for NISQ devices of less than a thousand junctions but may not prove sufficient for larger devices.

A second advantage of local sequence concatenation is the ability to run parallel shuttling sequences, an important next step toward scalability, as long as these sequences do not share a common electrode at the same time of the sequence. Implementing parallel shuttling will require an extension of the QCCD syntax to support parallel statements. Shuttling speed with parallel ion control will be limited by the DAC bandwidth, which is the hardware limit on how fast a DAC card can modify the voltage of a DAC channel. Most DAC cards, including the Zotino card used in the logical qubit apparatus, share the same bandwidth for all its channels. Increasing the bandwidth per channel is either achieved by increasing the number of DAC cards being used or by using DAC cards with a higher bandwidth.

8.2.3 Shuttling on a X-junction

Shuttling ions on an X junction can be decomposed via local shuttling solutions down to two sequences: a linear shuttling sequence requiring five pairs of electrodes and covering the position control over the central electrode pair, and the shuttling across the X junction. Shuttling sequences are simply a concatenation of trapping solutions. The distance between two points of a sequence is required to be below the scale in which the electric potential, in the axial axis, remains harmonic [192, Appendix D] and was thus chosen to be $2\ \mu\text{m}$. They are saved as JSON files with electrode labels chosen by convention to be easily adaptable by the software running the sequence concatenation. All motion control available on the chip is displayed in Figure 8.8.

By symmetry it is only required to solve the trajectory from one arm to the center. There are major obstacles toward shuttling in an X-junction : first of all the DC electrodes are further away from the center of the junction than from an arm on the chip, reducing the influence of the electrostatic field. This can be seen Figure 8.9 where the simulated waveform reaches the maximally allowed DC voltages near the center. Secondly, the secular eigen axis at the center have degenerate solutions for the axial and radial components if all fields around the junction are symmetric, therefore a change in the secular frequency and an unavoidable increase of the motional mode. [192, Section 5.6]. The sequence was implemented for shuttling between the west and the east arm by bringing down all rotation electrodes to zero volt while shuttling. The sequence was confirmed by seeing the ion leaving the area observed by the camera and coming back at the end of the sequence. While unlikely, it remains possible that the ion stays stuck near the junction and does not cross it before being brought back. Stronger experimental

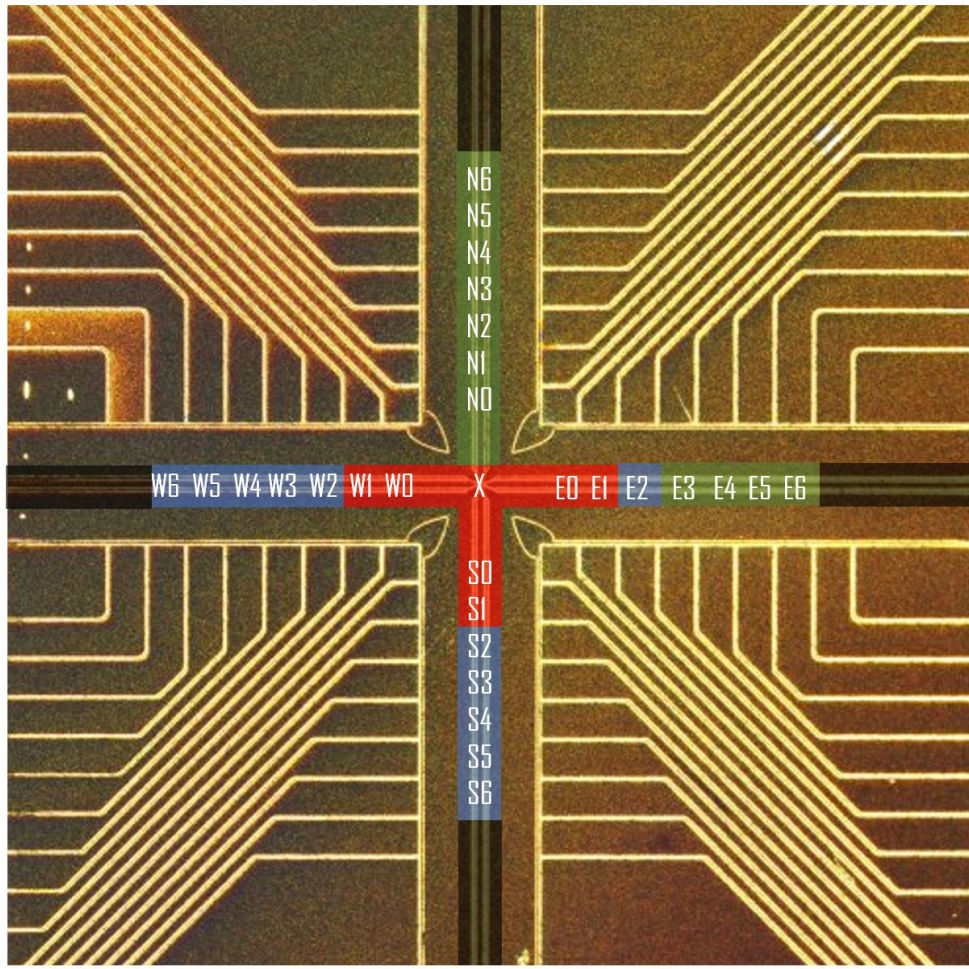


Figure 8.8: Syntax representation used for parsing shuttling instructions. (Red) area covered by local X-junction shuttling sequence. (Blue) area covered by local linear shuttling sequence. (Green) hardly accessible area due to electrodes shorted and/or wirebonded in pairs. (Black) inaccessible areas due to the lack of electrodes on the sides. The default position to trap ions was chosen to be W3.

confirmation of the shuttling sequence is needed such as observing the ion in the other arm.

8.2.4 Splitting and merging

By adding a quadratic term to the trapping potential [193], a single potential well containing an ion crystal can be progressively split into two separate wells. With fine tuning adjustment of the electrode voltages one can select which section of the ion crystal is split. Reciprocally, a merging operation consists of running the splitting operation in reverse, leading to the merging multiple ion crystals into one. Therefore the two processes follow the same voltage sequence and associated errors and are investigated as a single process.

Splitting sequences are simulated, as displayed in Figure 8.10, by extending the python simulation module mentioned above to incorporate a quadratic potential as well as solving the minimizer over multiple points of

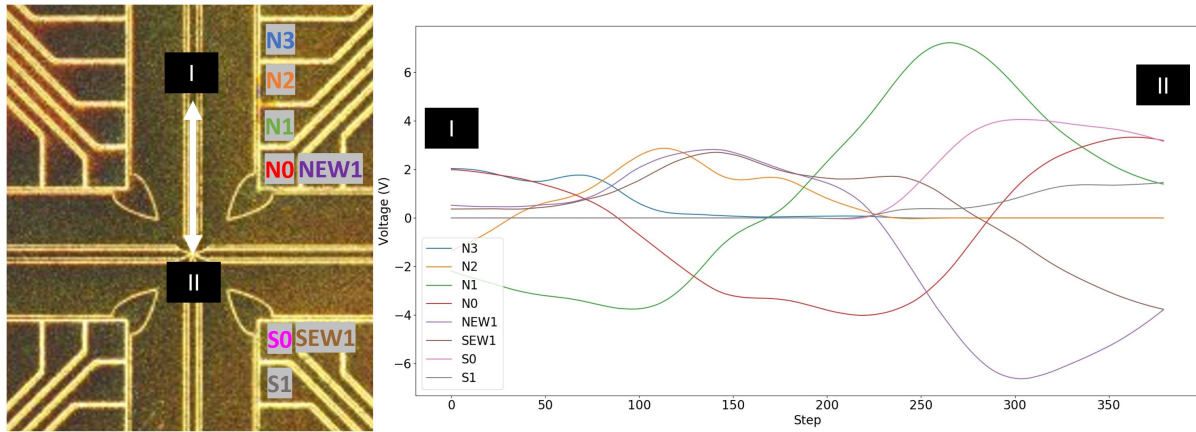


Figure 8.9: Shuttling sequence to approach the X-junction center. The trajectory mapped by this sequence is represented on the left, with the sequence itself displayed on the right. The waveform was simulated on the North-South axis due to the simulation software architecture, assuming each electrodes could be controlled individually, and was run experimentally in the West-East axis. Each waveform represents a pair of one or multiple electrodes on the chip. The increased distance between the ion and the electrodes can be seen near the center of the chip at the end of the waveform, where the required DC voltages are much higher in general and even reach for some electrodes the maximum value achievable in hardware.

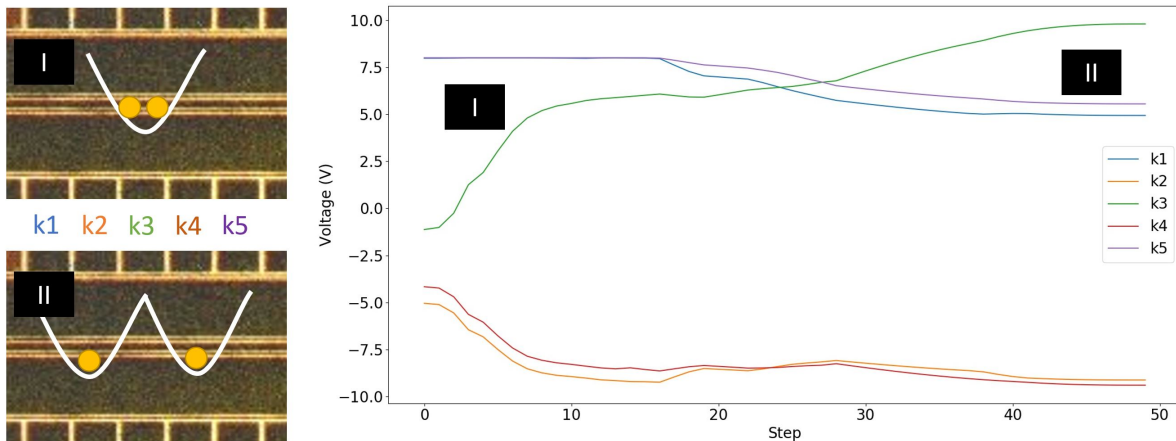


Figure 8.10: Shuttling sequence to split an ion crystal of even numbers of ions. The two different end states “merged” and “split” controlled by this sequence are represented on the left with I and II. The sequence itself is displayed on the right. Each waveform represents a pair of electrodes on the chip. The sequence is applicable on any linear section of the surface chip conditional to a single offset field applied on an adjacent pair of electrodes to compensate the small local geometry difference.

interest instead of one. A slow splitting and merging protocol is measured with the EMCCD camera in Figure 8.11, illustrating the separation of the two ions.

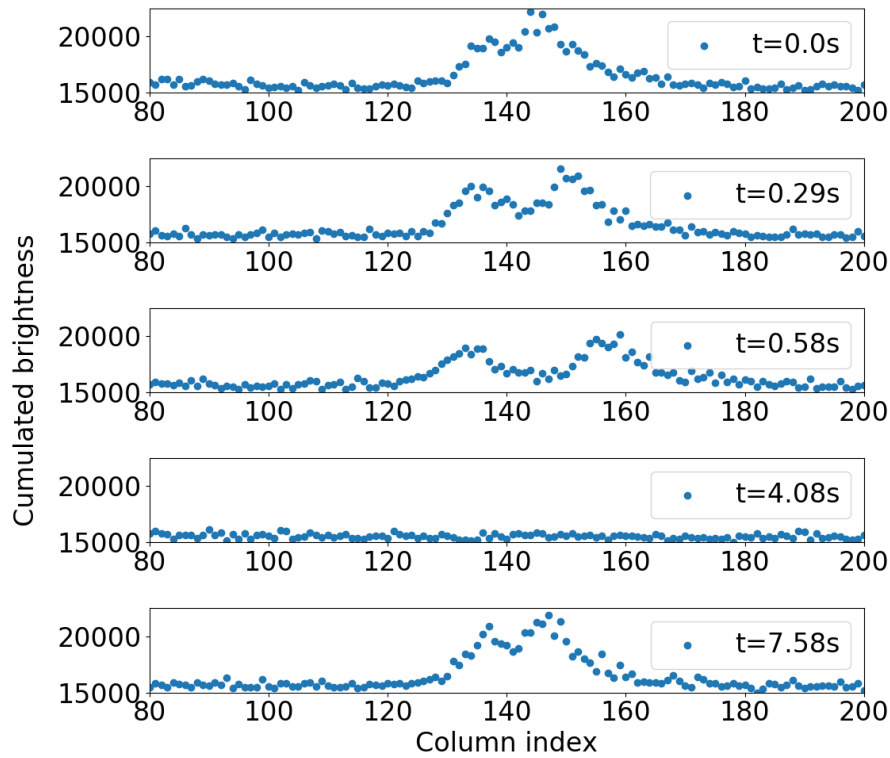


Figure 8.11: Splitting protocol measured by the EMCCD camera. The steps displayed in Figure 8.10 are implemented with a 50 ms step delay, which leads to a splitting time of 2.5 s. The camera has an integration time of 130 ms. The first three measures up to $t = 0.58$ s demonstrate the separation process of the ion crystal. Once split, at $t = 4.08$ s, the ions are too far away from the laser beam to be visible. Finally, after a 3 seconds delay the merging sequence is run, and the ions are recrystallized again, at $t = 7.58$ s, in the same ion crystal.

8.3 Software integration toward a functional QPU

The physical protocols to shuttle ions on a surface chip have been detailed and implemented. Combining those results with the organization used in other experiments of the research group to implement quantum logic operations, it becomes possible to envision with the next generation of the apparatus how to running quantum circuit algorithm. The remaining hardware challenges are considered in detail at the conclusion.

8.3.1 CPU limitations for shuttling instructions

We consider quantum algorithms as being sequences of operations such as those defined in Chapter 5. In this context we identify two types of operations: quantum control sequences, which are equivalent to a time delay, and shuttling sequences. The design and implementation of complex shuttling sequences was split into a two step process, starting with the prototyping model. As described in Section 8.2.2, the DC waveform is calculated in the CPU, and the resulting commands are sent to the FPGA as lists of arrays. This model allows the user

to quickly change any desired local sequences by adding new files and arbitrary speed ramps can be added via interpolation on the concatenated waveform in order to minimize heating. Unfortunately, the compilation time of the resulting arrays to be sent to the FPGA is very large. From Figure 8.12, we can see that the threshold after which the compilation becomes problematic is between one and two round trips across the X-junction, which is much less than most algorithmic requirements discussed earlier in Chapter 5.

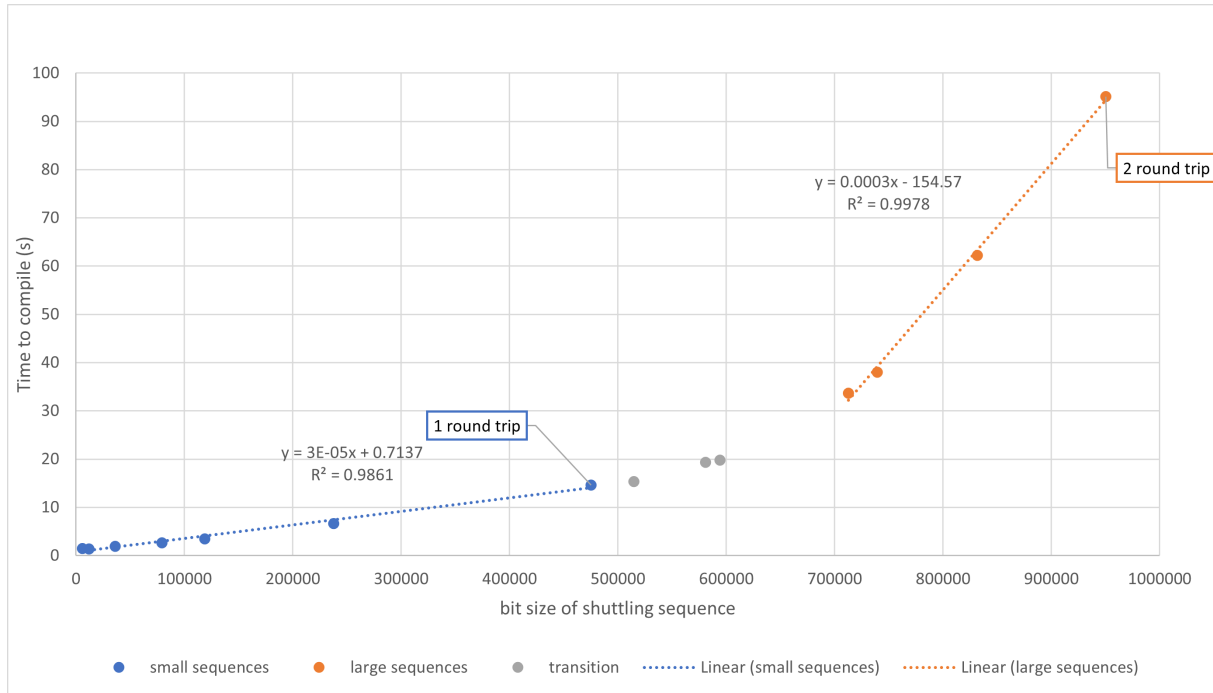


Figure 8.12: Compilation time taken by ARTIQ to upload shuttling waveform into the FPGA. For small waveforms a first linear trend is observed. When the data size to be sent reaches a certain threshold a much steeper linear trend appears. Larger data sizes crash the FPGA compiler.

Furthermore, larger shuttling files also lead to compilation error. This limit arises purely from the chosen architecture due to the choice of assembling all the shuttling waveforms in the CPU. Even the compression method detailed in Section 8.2.2, allows at best dozen of round trip across two arms of the X-junction. Such a limit is big enough to allow for a complete study of any local sequences, as well as initial testing of combination of sequences. This compiler can be used to select the best waveform maximizing shuttling speed and fidelity, while minimizing heating. It is however not applicable for running arbitrary large scale circuits with high fidelity, and another approach to run shuttling sequences is required.

8.3.2 Scalable shuttling via FPGA waveform control

A second version of the compiler would consist in integrating the DC waveforms in the FPGA. The waveforms stored locally in the FPGA would be the same building block used to build a concatenated shuttling sequence described in Section 8.2.2. With the sequences directly available in the FPGA instead of being calculated in the CPU, the compilation overhead is removed and arbitrarily long sequences can be run. For Sinara devices, the access to the FPGA memory is accessed via the DMA module². The next steps to enable such an architecture are to compress all the required waveforms for controlling the ions in QCCD code in the FPGA memory, and to demonstrate real-time concatenation of waveforms by the FPGA, by sequentially loading the waveforms, without impacting the shuttling speed.

8.3.3 Full stack software integration

The logical qubit apparatus was integrated to the software stack described in Chapter 5. The integration occurred while the sequence limited shuttling compiler was still in use, as the ARTIQ syntax is aimed to remain the same.

The latest layer of the full stack software organization consists in translating the pulse sequence JSON dictionary into the relevant experiment for the experimental controller to run. The organisation needs to handle loop calibrations as described in Chapter 3, as well as running most of the compiler stack asynchronously from the experimental computer. The concept of running experiments is already simplified in ARTIQ with python objects called `EnvExperiment` which prepare the experiment, run real time operations and analyze the obtained data; more importantly, Python files can already be transmitted and submitted to the ARTIQ schedule over a network.

The hardware layer is therefore a python script first translating the pulse sequence data into an ARTIQ experiment python file, then submitting it to the computer running the targeted quantum processing unit. This solution allows us to clearly store the physical process behind the quantum circuit for debugging and reusability, and provides little additional work since we are reusing most of the ARTIQ framework.

²[https://m-labs.hk/artiq/manual/core_drivers_reference.html# module-artiq.coredevice.dma](https://m-labs.hk/artiq/manual/core_drivers_reference.html#module-artiq.coredevice.dma)

8.4 Conclusion and future work

This chapter presented some investigation of the shuttling aspect of trapped ion technology, and further demonstrated that one of the main obstacles to build quantum computers is scalability. More than just running gates across large scale devices, multiple technological aspects, such as calibration, compilation and as seen in this chapter shuttling ions, are relatively straightforward to demonstrate but require very efficient approaches when running on a longer time scale.

The work presented in this chapter can be further developed in multiple directions. The pressure estimated in Section 7.1.1 and measured with the ion pump can be further characterized by observing ion hopping and ion crystal reordering on a double-well potential [194] similar to the one created in the splitting sequence. The surface chip still needs to be characterized for heating rates and shuttling around the corner protocols still need to be implemented. A second step would be to apply an external magnetic field so that $^{171}\text{Yb}^+$ can be trapped. Although the lack of a gradient would prevent the implementation of entangling gates, having access to single qubit quantum logic would allow us to characterize the phase accumulated from each shuttling sequence. Heating rate characterization of each type of shuttling sequence would further help estimate the capabilities of the prototype once fully operational. A prototype of scalable shuttling compilation in the FPGA is still required but a potential solution has been described.

A major update hinted at in most of the previous sections consists in replacing the existing chip in the near future with a surface chip containing micro-fabricated current coils, allowing magnetic fields and magnetic field gradients to be generated on demand. This would allow for the implementation of entangling gates on the apparatus, the last remaining element in the way of achieving a fully functional quantum computer prototype.

Chapter 9

Conclusion

9.1 Summary

This thesis represents a full stack approach on developing a quantum computer prototype, from software to hardware considerations, with the objective of studying physical implementations of quantum error correcting codes.

In Chapter 1 the main algorithms aimed to be implemented on quantum computers are quickly reviewed, outlining the circuit size involved and the struggle to define a correct metric evaluating a quantum computer's efficiency.

Chapter 2 contained the detailed theoretical framework used to build quantum computers in the Ion Quantum Technology group. The advantages in terms of scalability of this framework are explained and the hardware organization for large scale devices is illustrated.

Chapter 3 focuses on the importance of gate robustness in the control of a quantum computer. After shortly reviewing the concept of robustness, a robust gate scheme is described, implemented and characterized. A different approach on estimating gate robustness is then conducted to further illustrate advantages when calibrating and operating a quantum computer. They show that gate robustness not only increases the circuit fidelity but also increases the duty cycle between operations and calibrations and the average available time.

In Chapter 4 the fundamentals of Quantum Error Correction are presented and in particular the surface code which is today the most promising scheme to realize in the near future. Tools to evaluate and implement those schemes and reflections about the impact of ion shuttling are outlined. A first set of requirements is outlined for experimentally test QEC codes.

In Chapter 5 is considered the software stack of a quantum computer. The main elements to compile a generic

quantum circuit into hardware instructions are presented. In particular a new hardware quantum code syntax is introduced, specifically designed for quantum computers with shuttling ion architectures. The stack is merged with the existing pulse sequence layer already used in the research group with the aim of integrating older experiments under the same framework. Quantum Error Correction in this framework is also considered, and additional requirements are established regarding shuttling protocols.

In Chapter 6 a set of automation software is presented, controlling and monitoring different experimental tasks common to ion trapping experiments. The built-in interfaces are important in providing individual apparatus automated access to monitored data and complex control capabilities.

In Chapter 7 the experimental apparatus to demonstrate quantum error correction schemes on trapped ions is detailed, including in particular, the design of a new adjustable atomic oven and an overall development on automating the ion trapping device.

In Chapter 8 the early operations and measurements of the experiment, oriented toward the physical shuttling architecture, are presented. The experimental implementation of several shuttling protocols is described, notably ion shuttling through an X-junction and splitting and merging operations on two ion crystals, outlining some challenges and presenting an architecture toward larger devices and larger shuttling sequence operations required for quantum circuits.

9.2 Future work

The most essential component missing in the experiment as already described in Chapter 8 is the ability to generate a magnetic field gradient that is required to implement entangling gates. A new generation of micro-fabricated X-junction surface traps with buried copper channels that can generate the magnetic field gradients, are in development, with a first prototype being characterized in the research group and displayed in Figure 9.1. The steps toward implementing a fully working quantum computer prototype are as follow:

1. The new chip needs to be characterized. We will measure the maximum current that can be applied so that the cooling power is sufficient to prevent run-away effects from thermal dissipation. If run-away occurs below the desired current level, a similar study can be run for shortly applied current, as high gradients are only required during the gate time. Crosstalk measurements between the copper channels and the DC and RF electrodes will then be conducted to confirm the independence between the trapping layer and the gradient layer of the chip.
2. The chip will be inserted in the vacuum system along with the additional current connectors for the

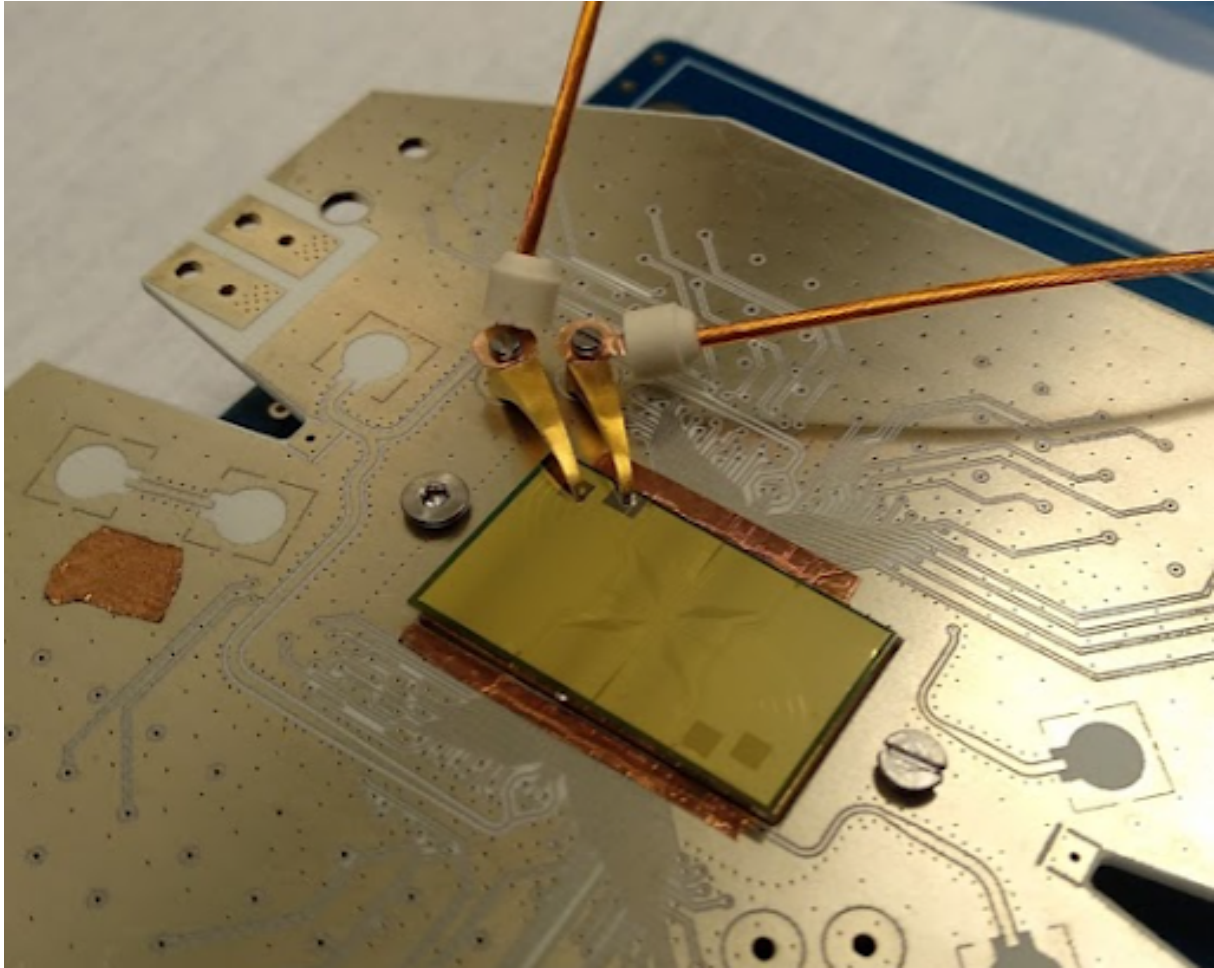


Figure 9.1: Surface chip with buried current wires. The existing PCB from Chapter 7 are reused. The picture displays initial characterizations of a buried current wire by connecting clamps (corner of the chip) as input and output of the current path.

copper channels, and baked as mentioned in 7. Initial trapping of $^{174}\text{Yb}^+$ will be conducted similarly to the current apparatus, to confirm the working state of the optic setup, the imaging setup and the shuttling architecture.

3. $^{171}\text{Yb}^+$ will be trapped following the same protocol as described in [157], using small currents applied to the buried copper wire to generate the required magnetic field. This step is followed by the optimization of the fluorescence, and the installation of the AWG for driving the external fields for quantum control.
4. Quantum control is at this stage implemented, starting by the calibration described in Chapter 3, and followed by quantum gates. The current applied to the copper channel will be gradually increased to characterize the magnetic field gradient and the qubit stability at these gradients.
5. Once the quantum control toolbox is characterized, heating and phase errors due to shuttling protocols can be finely measured and minimized. From that stage onwards, quantum circuits involving quantum

gates and shuttling operations studied in Chapter 5 can be implemented, and initial studies of QEC codes described in Chapter 4 can be conducted.

Several engineering aspects were considered in this thesis, demonstrating the shift of quantum computing from an academic curiosity towards a robust industrial technology. The heavy use of frameworks, calibrations and automation tools, as outlined on several occasions, is indispensable to facilitate and speed up common experiments in order to build up quantum control of trapped ions into a fully functional quantum computer. Among the subjects approached in this thesis, some need further investigation to bring further improvement to a quantum computing system:

- The QCCD syntax introduced in Chapter 5 needs to be tested experimentally, and extended such as to incorporate parallel instructions and conditional logic. ProjectQ and the PulseSequence should be similarly extended to reach capabilities similar to the OpenQASM 3.0 syntax [195]. The routing algorithm used to demonstrate the use of the QCCD syntax was not efficient and will require a dedicated investigation.
- The quantum code framework used in this does not yet fully integrate quantum error correction. The challenges to integrate concatenated logical operations have been shortly described in Section 5.6.1 and will require additional software engineering.

Bibliography

- [1] B. Lekitsch, S. Weidt, A. G. Fowler, K. Mølmer, S. J. Devitt, C. Wunderlich, and W. K. Hensinger, “Blueprint for a microwave trapped ion quantum computer,” *Science Advances*, vol. 3, no. 2, p. e1601540, 2017.
- [2] A. E. Webb, *Robust quantum logic for trapped ion quantum computers*. PhD thesis, University of Sussex, 2018.
- [3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. USA: Cambridge University Press, 10th ed., 2011.
- [4] M. Mttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, “Quantum circuits for general multiqubit gates,” *Physical Review Letters*, vol. 93, p. 130502, Sep 2004.
- [5] D. P. DiVincenzo, “The physical implementation of quantum computation,” *Fortschritte der Physik*, vol. 48, no. 9-11, pp. 771–783, 2000.
- [6] R. Blume-Kohout, J. Gamble, E. Nielsen, J. Mizrahi, J. Sterk, and P. Maunz, “Robust, self-consistent, closed-form tomography of quantum logic gates on a trapped ion qubit,” *arXiv:1310.4492*, 2013.
- [7] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, “Validating quantum computers using randomized model circuits,” *Physical Review A*, vol. 100, p. 032328, Sep 2019.
- [8] S. Aaronson and L. Chen, “Complexity-theoretic foundations of quantum supremacy experiments,” in *Proceedings of the 32nd Computational Complexity Conference, CCC ’17*, (Dagstuhl, DEU), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
- [9] “System configuration.” <https://quantum-computing.ibm.com/docs/manage/systems/configuration>. Accessed: 2021-03-16.

- [10] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, Jun 1982.
- [11] S. Lloyd, "Universal quantum simulators," *Science*, vol. 273, no. 5278, pp. 1073–1078, 1996.
- [12] R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, "Simulating physical phenomena by quantum networks," *Physical Review A*, vol. 65, p. 042323, Apr 2002.
- [13] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, "Simulation of electronic structure Hamiltonians using quantum computers," *Molecular Physics*, vol. 109, pp. 735–750, Mar 2011.
- [14] D. Wecker, B. Bauer, B. K. Clark, M. B. Hastings, and M. Troyer, "Gate-count estimates for performing quantum chemistry on small quantum computers," *Physical Review A*, vol. 90, p. 022305, Aug 2014.
- [15] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, p. 4213, Jul 2014.
- [16] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, "An adaptive variational algorithm for exact molecular simulations on a quantum computer," *Nature Communications*, vol. 10, p. 3007, Jul 2019.
- [17] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, pp. 242–246, Sep 2017.
- [18] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. P. Lanyon, P. Love, R. Babbush, and et al., "Quantum chemistry calculations on a trapped-ion quantum simulator," *Physical Review X*, vol. 8, p. 031022, Jul 2018.
- [19] M. E. Briggs, *An introduction to the general number field sieve*. PhD thesis, Virginia Tech, 1998.
- [20] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, pp. 1484–1509, Oct 1997.
- [21] I. L. Markov and M. Saeedi, "Constant-optimized quantum circuits for modular multiplication and exponentiation," *Quantum Info. Comput.*, vol. 12, pp. 361–394, May 2012.
- [22] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, Feb 1978.

- [23] F. Glover, G. Kochenberger, and Y. Du, “Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models,” *4OR*, vol. 17, pp. 335–371, Dec 2019.
- [24] A. Gilliam, M. Pistoia, and C. Gonciulea, “Canonical construction of quantum oracles,” *arXiv:2006.10656*, 2020.
- [25] J. Preskill, “Quantum Computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, Aug 2018.
- [26] D. Kielpinski, C. Monroe, and D. J. Wineland, “Architecture for a large-scale ion-trap quantum computer,” *Nature*, vol. 417, pp. 709–711, Jun 2002.
- [27] F. Mintert and C. Wunderlich, “Ion-trap quantum logic using long-wavelength radiation,” *Physical Review Letters*, vol. 87, p. 257904, Nov 2001.
- [28] F. M. Penning, “Die glimmentladung bei niedrigem druck zwischen koaxialen zylindern in einem axialen magnetfeld,” *Physica*, vol. 3, no. 9, pp. 873–894, 1936.
- [29] H. Dehmelt, “Radiofrequency spectroscopy of stored ions I: Storage,” vol. 3 of *Advances in Atomic and Molecular Physics*, pp. 53–72, Academic Press, 1968.
- [30] S. Jain, J. Alonso, M. Grau, and J. P. Home, “Scalable arrays of micro-Penning traps for quantum computing and simulation,” *Physical Review X*, vol. 10, p. 031027, Aug 2020.
- [31] W. Paul and H. Steinwedel, “Ein neues massenspektrometer ohne magnetfeld,” *Zeitschrift Naturforschung Teil A*, vol. 8, p. 448, Jul 1953.
- [32] E. Mathieu, “Mémoire sur le mouvement vibratoire d’une membrane de forme elliptique,” *Journal de Mathématiques Pures et Appliquées*, vol. 13, pp. 137–203, 1868.
- [33] R. F. Wuerker, H. Shelton, and R. V. Langmuir, “Electrodynamic containment of charged particles,” *Journal of Applied Physics*, vol. 30, no. 3, pp. 342–349, 1959.
- [34] I. Kovacic, R. Rand, and S. Sah, “Mathieu’s equation and its generalizations: Overview of stability charts and their features,” *Applied Mechanics Reviews*, vol. 70, 2018. 020802.
- [35] D. F. V. James, “Quantum dynamics of cold trapped ions with application to quantum computation,” *Applied Physics B*, vol. 66, pp. 181–190, Feb 1998.
- [36] I. Pogorelov, T. Feldker, C. D. Marciniak, L. Postler, G. Jacob, O. Kriegelsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, *et al.*, “Compact ion-trap quantum computing demonstrator,” *PRX Quantum*, vol. 2, no. 2, p. 020343, 2021.

- [37] N. Friis, O. Marty, C. Maier, C. Hempel, M. Holzäpfel, P. Jurcevic, M. B. Plenio, M. Huber, C. Roos, R. Blatt, and B. Lanyon, “Observation of entangled states of a fully controlled 20-qubit system,” *Physical Review X*, vol. 8, p. 021012, Apr 2018.
- [38] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pistenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, and J. Kim, “Benchmarking an 11-qubit quantum computer,” *Nature Communications*, vol. 10, no. 1, p. 5464, 2019.
- [39] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, “Demonstration of a small programmable quantum computer with atomic qubits,” *Nature*, vol. 536, pp. 63–66, Aug 2016.
- [40] L. Feng, W. L. Tan, A. De, A. Menon, A. Chu, G. Pagano, and C. Monroe, “Efficient ground-state cooling of large trapped-ion chains with an electromagnetically-induced-transparency tripod scheme,” *Physical Review Letters*, vol. 125, p. 053001, Jul 2020.
- [41] Y. Wu, *Quantum Computation in Large Ion Crystals*. PhD thesis, University of Michigan, 2019.
- [42] C. J. Trout, M. Li, M. Gutierrez, Y. Wu, S.-T. Wang, L. Duan, and K. R. Brown, “Simulating the performance of a distance-3 surface code in a linear ion trap,” *New Journal of Physics*, vol. 20, p. 043038, Apr 2018.
- [43] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, “Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator,” *Nature*, vol. 551, pp. 601–604, Nov 2017.
- [44] D. Stick, W. K. Hensinger, S. Olmschenk, M. J. Madsen, K. Schwab, and C. Monroe, “Ion trap in a semiconductor chip,” *Nature Physics*, vol. 2, p. 36, Jan 2006.
- [45] S. Seidelin, J. Chiaverini, R. Reichle, J. J. Bollinger, D. Leibfried, J. Britton, J. H. Wesenberg, R. B. Blakestad, R. J. Epstein, D. B. Hume, W. M. Itano, J. D. Jost, C. Langer, R. Ozeri, N. Shiga, and D. J. Wineland, “Microfabricated surface-electrode ion trap for scalable quantum information processing,” *Physical Review Letters*, vol. 96, p. 253003, Jun 2006.
- [46] M. G. House, “Analytic model for electrostatic fields in surface-electrode ion traps,” *Physical Review A*, vol. 78, p. 033402, Sep 2008.

- [47] J. Chiaverini, R. B. Blakestad, J. Britton, J. D. Jost, C. Langer, D. Leibfried, R. Ozeri, and D. J. Wineland, “Surface-electrode architecture for ion-trap quantum information processing,” *Quantum Info. Comput.*, vol. 5, pp. 419–439, Sep 2005.
- [48] F. A. Shaikh and A. Ozakin, “Stability analysis of ion motion in asymmetric planar ion traps,” *Journal of Applied Physics*, vol. 112, p. 074904, Oct 2012.
- [49] J. I. Cirac and P. Zoller, “Quantum computations with cold trapped ions,” *Physical Review Letters*, vol. 74, pp. 4091–4094, May 1995.
- [50] Q. A. Turchette, C. S. Wood, B. E. King, C. J. Myatt, D. Leibfried, W. M. Itano, C. Monroe, and D. J. Wineland, “Deterministic entanglement of two trapped ions,” *Physical Review Letters*, vol. 81, pp. 3631–3634, Oct 1998.
- [51] K. Mølmer and A. Sørensen, “Multiparticle entanglement of hot trapped ions,” *Physical Review Letters*, vol. 82, pp. 1835–1838, Mar 1999.
- [52] A. Sørensen and K. Mølmer, “Entanglement and quantum computation with ions in thermal motion,” *Physical Review A*, vol. 62, p. 022311, Jul 2000.
- [53] G. J. Milburn, “Simulating nonlinear spin models in an ion trap,” *arXiv:quant-ph/9908037*, 1999.
- [54] A. H. Nizamani, J. J. McLoughlin, and W. K. Hensinger, “Doppler-free yb spectroscopy with the fluorescence spot technique,” *Physical Review A*, vol. 82, p. 043408, Oct 2010.
- [55] J. Randall, *High-Fidelity Entanglement of Trapped Ions using Long-Wavelength Radiation*. PhD thesis, Imperial College London, 2016.
- [56] S. Olmschenk, K. C. Younge, D. L. Moehring, D. N. Matsukevich, P. Maunz, and C. Monroe, “Manipulation and detection of a trapped yb^+ hyperfine qubit,” *Physical Review A*, vol. 76, p. 052314, Nov 2007.
- [57] P. Wang, C.-Y. Luan, M. Qiao, M. Um, J. Zhang, Y. Wang, X. Yuan, M. Gu, J. Zhang, and K. Kim, “Single ion qubit with estimated coherence time exceeding one hour,” *Nature Communications*, vol. 12, p. 233, Jan 2021.
- [58] D. Reagor, Yan Fan, C. Mombourquette, Quanxi Jia, and L. Stolarczyk, “A high-temperature superconducting receiver for low-frequency radio waves,” *IEEE Transactions on Applied Superconductivity*, vol. 7, no. 4, pp. 3845–3849, 1997.

- [59] U. Gaubatz, P. Rudecki, S. Schieman, and K. Bergmann, "Population transfer between molecular vibrational levels by stimulated raman scattering with partially overlapping laser fields. a new concept and experimental results," *The Journal of Chemical Physics*, vol. 92, no. 9, pp. 5363–5376, 1990.
- [60] J. Randall, A. M. Lawrence, S. C. Webster, S. Weidt, N. V. Vitanov, and W. K. Hensinger, "Generation of high-fidelity quantum control methods for multilevel systems," *Physical Review A*, vol. 98, p. 043414, Oct 2018.
- [61] S. Wimperis, "Broadband, narrowband, and passband composite pulses for use in advanced nmr experiments," *Journal of Magnetic Resonance, Series A*, vol. 109, no. 2, pp. 221–231, 1994.
- [62] J. H. Wesenberg, R. J. Epstein, D. Leibfried, R. B. Blakestad, J. Britton, J. P. Home, W. M. Itano, J. D. Jost, E. Knill, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Fluorescence during Doppler cooling of a single trapped atom," *Physical Review A*, vol. 76, p. 053416, Nov 2007.
- [63] A. Seif, K. A. Landsman, N. M. Linke, C. Figgatt, C. Monroe, and M. Hafezi, "Machine learning assisted readout of trapped-ion qubits," *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 51, p. 174006, Aug 2018.
- [64] R. Islam, W. C. Campbell, T. Choi, S. M. Clark, C. W. S. Conover, S. Debnath, E. E. Edwards, B. Fields, D. Hayes, D. Hucul, I. V. Inlek, K. G. Johnson, S. Korenblit, A. Lee, K. W. Lee, T. A. Manning, D. N. Matsukevich, J. Mizrahi, Q. Quraishi, C. Senko, J. Smith, and C. Monroe, "Beat note stabilization of mode-locked lasers for quantum information processing," *Optics Letters*, vol. 39, pp. 3238–3241, Jun 2014.
- [65] C. Ospelkaus, U. Warring, Y. Colombe, K. R. Brown, J. M. Amini, D. Leibfried, and D. J. Wineland, "Microwave quantum logic gates for trapped ions," *Nature*, vol. 476, pp. 181–184, Aug 2011.
- [66] S. Weidt, J. Randall, S. C. Webster, K. Lake, A. E. Webb, I. Cohen, T. Navickas, B. Lekitsch, A. Retzker, and W. K. Hensinger, "Trapped-ion quantum logic with global radiation fields," *Physical Review Letters*, vol. 117, p. 220501, Nov 2016.
- [67] F. Diedrich, J. C. Bergquist, W. M. Itano, and D. J. Wineland, "Laser cooling to the zero-point energy of motion," *Physical Review Letters*, vol. 62, pp. 403–406, Jan 1989.
- [68] F. Schmidt-Kaler, C. Roos, H. C. Ngerl, H. Rohde, S. Gulde, A. Mundt, M. Lederbauer, G. Thalhammer, T. Zeiger, P. Barton, and et al., "Ground state cooling, quantum state engineering and study of decoherence of ions in Paul traps," *Journal of Modern Optics*, vol. 47, pp. 2573–2582, Nov 2000.

- [69] S. Weidt, J. Randall, S. Webster, E. Standing, A. Rodriguez, A. Webb, B. Lekitsch, and W. Hensinger, “Ground-state cooling of a trapped ion using long-wavelength radiation,” *Physical Review Letters*, vol. 115, Jun 2015.
- [70] Y. Wan, R. Jrdens, S. D. Erickson, J. J. Wu, R. Bowler, T. R. Tan, P.-Y. Hou, D. J. Wineland, A. C. Wilson, and D. Leibfried, “Ion transport and reordering in a 2d trap array,” *Advanced Quantum Technologies*, vol. 3, no. 11, p. 2000028, 2020.
- [71] Q. A. Turchette, Kielpinski, B. E. King, D. Leibfried, D. M. Meekhof, C. J. Myatt, M. A. Rowe, C. A. Sackett, C. S. Wood, W. M. Itano, and et al., “Heating of trapped ions from the quantum ground state,” *Physical Review A*, vol. 61, p. 063418, May 2000.
- [72] H. A. Frst, M. H. Goerz, U. G. Poschinger, M. Murphy, S. Montangero, T. Calarco, F. Schmidt-Kaler, K. Singer, and C. P. Koch, “Controlling the transport of an ion: classical and quantum mechanical solutions,” *New Journal of Physics*, vol. 16, p. 075007, Jul 2014.
- [73] E. D. Standing, *Design and fabrication of high magnetic eld gradients towards fault tolerant two-qubit gates with trapped ions using long-wavelength radiation*. PhD thesis, University of Sussex, 2016.
- [74] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, “Demonstration of the trapped-ion quantum ccd computer architecture,” *Nature*, vol. 592, pp. 209–213, Apr 2021.
- [75] Z. D. Romaszko, S. Hong, M. Siegele, R. K. Puddy, F. R. Lebrun-Gallagher, S. Weidt, and W. K. Hensinger, “Engineering of microfabricated ion traps and integration of advanced on-chip features,” *Nature Reviews Physics*, vol. 2, pp. 285–299, Jun 2020.
- [76] C. D. B. Bentley, H. Ball, M. J. Biercuk, A. R. R. Carvalho, M. R. Hush, and H. J. Slatyer, “Numeric optimization for configurable, parallel, error-robust entangling gates in large ion registers,” *Advanced Quantum Technologies*, vol. 3, no. 11, p. 2000044, 2020.
- [77] T. J. Green, J. Sastrawan, H. Uys, and M. J. Biercuk, “Arbitrary quantum control of qubits in the presence of universal noise,” *New Journal of Physics*, vol. 15, no. 9, p. 095004, 2013.
- [78] E. L. Hahn, “Spin echoes,” *Physical Review*, vol. 80, pp. 580–594, Nov 1950.
- [79] L. Viola and S. Lloyd, “Dynamical suppression of decoherence in two-state quantum systems,” *Physical Review A*, vol. 58, pp. 2733–2744, Oct 1998.

- [80] D. Hayes, S. M. Clark, S. Debnath, D. Hucul, I. V. Inlek, K. W. Lee, Q. Quraishi, and C. Monroe, “Coherent error suppression in multiqubit entangling gates,” *Physical Review Letters*, vol. 109, p. 020503, Jul 2012.
- [81] T. Manovitz, A. Rotem, R. Shaniv, I. Cohen, Y. Shapira, N. Akerman, A. Retzker, and R. Ozeri, “Fast dynamical decoupling of the mølmer-sørensen entangling gate,” *Physical Review Letters*, vol. 119, p. 220505, Nov 2017.
- [82] I. Cohen, S. Weidt, W. K. Hensinger, and A. Retzker, “Multi-qubit gate with trapped ions for microwave and laser-based implementation,” *New Journal of Physics*, vol. 17, p. 043008, Apr 2015.
- [83] F. Haddadfarshi and F. Mintert, “High fidelity quantum gates of trapped ions in the presence of motional heating,” *New Journal of Physics*, vol. 18, p. 123007, Dec 2016.
- [84] S. Collingbourne, “Entangling Gates For Quantum Computation,” Master’s thesis, Imperial College London, UK, 2017.
- [85] J. J. McLoughlin, A. H. Nizamani, J. D. Sivers, R. C. Sterling, M. D. Hughes, B. Lekitsch, B. Stein, S. Weidt, and W. K. Hensinger, “Versatile ytterbium ion trap experiment for operation of scalable ion-trap chips with motional heating and transition-frequency measurements,” *Physical Review A*, vol. 83, p. 013406, Jan 2011.
- [86] A. M. Lawrence, *High-fidelity quantum logic on trapped ions with microwave radiation*. PhD thesis, Imperial College London, 2019.
- [87] A. E. Webb, S. C. Webster, S. Collingbourne, D. Bretau, A. M. Lawrence, S. Weidt, F. Mintert, and W. K. Hensinger, “Resilient entangling gates for trapped ions,” *Physical Review Letters*, vol. 121, p. 180501, Nov 2018.
- [88] Y. Shapira, R. Shaniv, T. Manovitz, N. Akerman, and R. Ozeri, “Robust entanglement gates for trapped-ion qubits,” *Physical Review Letters*, vol. 121, p. 180502, Nov 2018.
- [89] J. Lishman and F. Mintert, “Trapped-ion entangling gates robust against qubit frequency errors,” *Physical Review Research*, vol. 2, p. 033117, Jul 2020.
- [90] I. Arrazola, M. Plenio, E. Solano, and J. Casanova, “Hybrid microwave-radiation patterns for high-fidelity quantum gates with trapped ions,” *Physical Review Applied*, vol. 13, p. 024068, Feb 2020.

- [91] J. Kelly, P. O'Malley, M. Neeley, H. Neven, and J. M. Martinis, "Physical qubit calibration on a directed acyclic graph," *arXiv:1803.03226*, 2018.
- [92] S. Beauregard, "Circuit for shor's algorithm using $2n+3$ qubits," *arXiv:quant-ph/0205095*, 2002.
- [93] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, pp. 802–803, Oct 1982.
- [94] P. K. Sarvepalli, A. Klappenecker, and M. Rötteler, "Asymmetric quantum codes: constructions, bounds and performance," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 465, no. 2105, pp. 1645–1672, 2009.
- [95] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997.
- [96] J. Preskill, "Fault-tolerant quantum computation," *Introduction to Quantum Computation and Information*, pp. 213–269, Oct 1998.
- [97] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, p. 076001, Jun 2013.
- [98] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, p. 032324, Sep 2012.
- [99] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, "Towards practical classical processing for the surface code," *Physical Review Letters*, vol. 108, p. 180501, May 2012.
- [100] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Physical Review A*, vol. 90, p. 032326, Sep 2014.
- [101] S. Varsamopoulos, K. Bertels, and C. G. Almudever, "Decoding surface code with a distributed neural networkbased decoder," *Quantum Machine Intelligence*, vol. 2, pp. 1–12, Mar 2020.
- [102] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, "Poking holes and cutting corners to achieve clifford gates with the surface code," *Physical Review X*, vol. 7, p. 021029, May 2017.
- [103] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *New Journal of Physics*, vol. 14, p. 123011, Dec 2012.

- [104] S. Bravyi and A. Kitaev, “Universal quantum computation with ideal clifford gates and noisy ancillas,” *Physical Review A*, vol. 71, p. 022316, Feb 2005.
- [105] J. O’Gorman and E. T. Campbell, “Quantum computation with realistic magic-state factories,” *Physical Review A*, vol. 95, p. 032338, Mar 2017.
- [106] H. Bombin and M. A. Martin-Delgado, “Topological quantum distillation,” *Physical Review Letters*, vol. 97, p. 180501, Oct 2006.
- [107] A. J. Landahl, J. T. Anderson, and P. R. Rice, “Fault-tolerant quantum computing with color codes,” *arXiv:1108.5738*, 2011.
- [108] M. Gutiérrez, M. Müller, and A. Bermúdez, “Transversality and lattice surgery: Exploring realistic routes toward coupled logical qubits with trapped-ion quantum processors,” *Physical Review A*, vol. 99, p. 022330, Feb 2019.
- [109] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, “Quantum computations on a topologically encoded qubit,” *Science*, vol. 345, pp. 302–305, Jun 2014.
- [110] M. Li, M. Gutiérrez, S. E. David, A. Hernandez, and K. R. Brown, “Fault tolerance with bare ancillary qubits for a $[[7,1,3]]$ code,” *Physical Review A*, vol. 96, p. 032341, Sep 2017.
- [111] A. Bermudez, X. Xu, R. Nigmatullin, J. O’Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. G. Poschinger, C. Hempel, J. Home, F. Schmidt-Kaler, M. Biercuk, R. Blatt, S. Benjamin, and M. Müller, “Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation,” *Physical Review X*, vol. 7, p. 041061, Dec 2017.
- [112] Y. Tomita and K. M. Svore, “Low-distance surface codes under realistic quantum noise,” *Physical Review A*, vol. 90, p. 062320, Dec 2014.
- [113] J. R. Wootton, A. Peter, J. R. Winkler, and D. Loss, “Proposal for a minimal surface code experiment,” *Physical Review A*, vol. 96, p. 032338, Sep 2017.
- [114] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, and et al., “State preservation by repetitive error detection in a superconducting quantum circuit,” *Nature*, vol. 519, pp. 66–69, Mar 2015.

- [115] M. Takita, A. Crcoles, E. Magesan, B. Abdo, M. Brink, A. Cross, J. M. Chow, and J. M. Gambetta, “Demonstration of weight-four parity measurements in the surface code architecture,” *Physical Review Letters*, vol. 117, p. 210505, Nov 2016.
- [116] A. Erhard, H. Poulsen Nautrup, M. Meth, L. Postler, R. Stricker, M. Stadler, V. Negnevitsky, M. Ringbauer, P. Schindler, H. J. Briegel, R. Blatt, N. Friis, and T. Monz, “Entangling logical qubits with lattice surgery,” *Nature*, vol. 589, pp. 220–224, Jan 2021.
- [117] D. P. DiVincenzo and P. Aliferis, “Effective fault-tolerant quantum computation with slow measurements,” *Physical Review Letters*, vol. 98, p. 020501, Jan 2007.
- [118] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, “A software methodology for compiling quantum programs,” *Quantum Science and Technology*, vol. 3, p. 020501, Feb 2018.
- [119] F. T. Chong, D. Franklin, and M. Martonosi, “Programming languages and compiler design for realistic quantum hardware,” *Nature*, vol. 549, pp. 180–187, Sep 2017.
- [120] R. LaRose, “Overview and Comparison of Gate Level Quantum Software Platforms,” *Quantum*, vol. 3, p. 130, Mar 2019.
- [121] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, “Scaffec: Scalable compilation and analysis of quantum programs,” *Parallel Computing*, vol. 45, p. 217, Jun 2015.
- [122] S. Liu, X. Wang, L. Zhou, J. Guan, Y. Li, Y. He, R. Duan, and M. Ying, “ $q|si\rangle$: a quantum programming environment,” *SCIENTIA SINICA Informationis*, vol. 47, pp. 1300–1315, Oct 2017.
- [123] D. S. Steiger, T. Häner, and M. Troyer, “ProjectQ: an open source software framework for quantum computing,” *Quantum*, vol. 2, p. 49, Jan 2018.
- [124] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernandez, J. Carballo-Franquis, A. Chen, C.-F. Chen, J. M. Chow, A. D. Crcoles-Gonzales, A. J. Cross, A. Cross, J. Cruz-Benito, C. Culver, S. D. L. P. Gonzalez, E. D. L. Torre, D. Ding, E. Dumitrescu, I. Duran, P. Eendebak, M. Everitt, I. F. Sertage, A. Frisch, A. Fuhrer, J. Gambetta, B. G. Gago, J. Gomez-Mosquera, D. Greenberg, I. Hamamura, V. Havlicek, J. Hellmers, ukasz Herok, H. Horii, S. Hu, T. Imamichi, T. Itoko, A. Javadi-Abhari, N. Kanazawa, A. Karazeev, K. Krsulich, P. Liu, Y. Luh, Y. Maeng, M. Marques, F. J. Martn-Fernandez, D. T. McClure, D. McKay, S. Meesala, A. Mezzacapo, N. Moll, D. M. Rodriguez, G. Nannicini, P. Nation, P. Ollitrault, L. J. O’Riordan, H. Paik, J. Prez, A. Phan, M. Pistoia, V. Prutyanov, M. Reuter, J. Rice, A. R. Davila, R. H. P. Rudy, M. Ryu,

- N. Sathaye, C. Schnabel, E. Schoute, K. Setia, Y. Shi, A. Silva, Y. Siraichi, S. Sivarajah, J. A. Smolin, M. Soeken, H. Takahashi, I. Tavernelli, C. Taylor, P. Taylour, K. Trabing, M. Treinish, W. Turner, D. Vogt-Lee, C. Vuillot, J. A. Wildstrom, J. Wilson, E. Winston, C. Wood, S. Wood, S. Wrner, I. Y. Akhalwaya, and C. Zoufal, “Qiskit: An Open-source Framework for Quantum Computing,” Jan 2019. <https://doi.org/10.5281/zenodo.2562111>.
- [125] R. S. Smith, M. J. Curtis, and W. J. Zeng, “A practical quantum instruction set architecture,” *arXiv:1608.03355*, 2016.
- [126] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, “Open quantum assembly language,” *arXiv:1707.03429*, 2017.
- [127] N. Khammassi, G. G. Guerreschi, I. Ashraf, J. W. Hogaboam, C. G. Almudever, and K. Bertels, “cqasm v1.0: Towards a common quantum assembly language,” *arXiv:1805.09607*, 2018.
- [128] P. J. Karalekas, N. A. Tezak, E. C. Peterson, C. A. Ryan, M. P. da Silva, and R. S. Smith, “A quantum-classical cloud platform optimized for variational hybrid algorithms,” *Quantum Science and Technology*, vol. 5, p. 024003, Apr 2020.
- [129] A. J. Landahl, D. Lobser, B. Morrison, K. M. Rudinger, A. Russo, J. W. Van Der Wall, and P. L. W. Maunz, “Jaql the quantum assembly language for qscout,” *arXiv:2003.09382*, 2020.
- [130] D. M. Greenberger, M. A. Horne, and A. Zeilinger, “Going beyond bells theorem,” *Bells Theorem, Quantum Theory and Conceptions of the Universe*, pp. 69–72, 1989.
- [131] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, “ $t|ket\rangle$: a retargetable compiler for nisq devices,” *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, 2020.
- [132] M. Swaddle, L. Noakes, H. Smallbone, L. Salter, and J. Wang, “Generating three-qubit quantum circuits with neural networks,” *Physics Letters A*, vol. 381, no. 39, pp. 3391–3395, 2017.
- [133] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, “Quantum-assisted quantum compiling,” *Quantum*, vol. 3, p. 140, May 2019.
- [134] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, “Software mitigation of crosstalk on noisy intermediate-scale quantum computers,” *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1001–1016, Mar 2020.

- [135] D. Maslov, “Basic circuit compilation techniques for an ion-trap quantum machine,” *New Journal of Physics*, vol. 19, p. 023035, Feb 2017.
- [136] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, “Automated optimization of large quantum circuits with continuous parameters,” *npj Quantum Information*, vol. 4, no. 1, pp. 1–12, 2018.
- [137] R. S. Smith, E. C. Peterson, M. G. Skilbeck, and E. J. Davis, “An open-source, industrial-strength optimizing compiler for quantum programs,” *Quantum Science and Technology*, vol. 5, p. 044001, Jul 2020.
- [138] G. G. Guerreschi, “Scheduler of quantum circuits based on dynamical pattern improvement and its application to hardware design,” *arXiv:1912.00035*, 2019.
- [139] P. Schindler, D. Nigg, T. Monz, J. T. Barreiro, E. Martinez, S. X. Wang, S. Quint, M. F. Brandl, V. Nebendahl, C. F. Roos, and et al., “A quantum information processor with trapped ions,” *New Journal of Physics*, vol. 15, p. 123012, Dec 2013.
- [140] M. Webber, S. Herbert, S. Weidt, and W. K. Hensinger, “Efficient qubit routing for a globally connected trapped ion quantum computer,” *Advanced Quantum Technologies*, vol. 3, p. 2000027, Jul 2020.
- [141] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, “An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures,” *SIGARCH Comput. Archit. News*, vol. 33, p. 186196, May 2005.
- [142] K. E. Stevens, J. M. Amini, S. C. Doret, G. Mohler, and A. W. Volin, Curtisand Harter, “Automating quantum experiment control,” *Quantum Information Processing*, vol. 16, p. 56, Jan 2017.
- [143] P. Murali, D. M. Debroy, K. R. Brown, and M. Martonosi, *Architecting Noisy Intermediate-Scale Trapped Ion Quantum Computers*, p. 529542. IEEE Press, 2020.
- [144] D. C. McKay, T. Alexander, L. Bello, M. J. Biercuk, L. Bishop, J. Chen, J. M. Chow, A. D. Croles, D. Egger, S. Filipp, J. Gomez, M. Hush, A. Javadi-Abhari, D. Moreda, P. Nation, B. Paulovicks, E. Winston, C. J. Wood, J. Wootton, and J. M. Gambetta, “Qiskit backend specifications for openqasm and openpulse experiments,” *arXiv:1809.03452*, 2018.
- [145] P. Gokhale, A. Javadi-Abhari, N. Earnest, Y. Shi, and F. T. Chong, “Optimized quantum compilation for near-term algorithms with openpulse,” in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, (Los Alamitos, CA, USA), pp. 186–200, IEEE Computer Society, Oct 2020.

- [146] H. Ball, M. Biercuk, A. Carvalho, J. Chen, M. R. Hush, L. A. de Castro, L. Li, P. J. Liebermann, H. Slatyer, C. Edmunds, V. Frey, C. Hempel, and A. Milne, “Software tools for quantum control: Improving quantum computer performance through noise and error suppression,” *Quantum Science and Technology*, vol. 6, no. 4, p. 044011, 2021.
- [147] D. Litinski, “A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery,” *Quantum*, vol. 3, p. 128, Mar 2019.
- [148] F. R. Lebrun-Gallagher, N. Johnson, M. Akhtar, S. Weidt, D. Breaud, S. J. Hile, A. Owens, and W. K. Hensinger, “A scalable helium gas cooling system for trapped-ion applications,” *arXiv:2106.07580*, 2021.
- [149] R. Lebrun-Ricalens, “Scalable closed cycle helium cryostat and in-vacuum electronics for fast ion transport,” Master’s thesis, University of Sussex, UK, 2016.
- [150] “Platinum.” <https://www.lakeshore.com/products/categories/overview/temperature-products/cryogenic-temperature-sensors/platinum>. Accessed: 2021-01-13.
- [151] “Quantum takes flight: Moving from laboratory demonstrations to building systems.” <https://www.ibm.com/blogs/research/2020/01/quantum-volume-32/>. Accessed: 2020-04-06.
- [152] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
- [153] “Aspen-7.” <https://www.rigetti.com/>. Accessed: 2020-04-06.
- [154] R. Calder and G. Lewin, “Reduction of stainless-steel outgassing in ultra-high vacuum,” *British Journal of Applied Physics*, vol. 18, pp. 1459–1472, Oct 1967.

- [155] D. Coyne *et al.*, “LIGO Vacuum Compatible Materials List,” tech. rep., LIGO Scientific Collaboration, Jun 2014.
- [156] A. Grounds, *Cryogenic technologies for scalable trapped ion quantum computing*. PhD thesis, University of Sussex, 2018.
- [157] D. F. Murgia, *Microchip ion traps with high magnetic field gradients for microwave quantum logic*. PhD thesis, Imperial College London, 2016.
- [158] L. Deslauriers, S. Olmschenk, D. Stick, W. K. Hensinger, J. Sterk, and C. Monroe, “Scaling and suppression of anomalous heating in ion traps,” *Physical Review Letters*, vol. 97, p. 103007, Sep 2006.
- [159] P. B. Antohi, D. Schuster, G. M. Akselrod, J. Labaziewicz, Y. Ge, Z. Lin, W. S. Bakr, and I. L. Chuang, “Cryogenic ion trapping systems with surface-electrode traps,” *Review of Scientific Instruments*, vol. 80, no. 1, p. 013103, 2009.
- [160] G. Pagano, P. W. Hess, H. B. Kaplan, W. L. Tan, P. Richerme, P. Becker, A. Kyprianidis, J. Zhang, E. Birckelbaw, M. R. Hernandez, Y. Wu, and C. Monroe, “Cryogenic trapped-ion system for large scale quantum simulation,” *Quantum Science and Technology*, vol. 4, p. 014004, Oct 2018.
- [161] J. W. Ekin, *Experimental techniques for low-temperature measurements: cryostat design, material properties, and superconductor critical-current testing*. Oxford University Press, 2015.
- [162] S. Bourdeauducq, whitequark, R. Jördens, Y. Sionneau, enjoy digital, cjbe, D. Nadlinger, hartyp, JBoulder, D. Slichter, Drew, mntng, r srinivas, apatura iris, S. Mackenzie, Z. Smith, P. K, M. Weber, kemstevens, F. Held, and D. Leibrandt, “m-labs/artiq: 4.0,” Nov 2018. <https://doi.org/10.5281/zenodo.1492176>.
- [163] T. H. group, “Hdf5.” <https://portal.hdfgroup.org/display/HDF5/HDF5>. Accessed: 2020-02-28.
- [164] J. D. Sivers, L. R. Simkins, S. Weidt, and W. K. Hensinger, “Efficient preparation and detection of microwave dressed-state qubits and qutrits with trapped ions,” *Applied Physics B*, vol. 107, p. 921, Jun 2012.
- [165] D. L. Moehring, C. Highstrete, D. Stick, K. M. Fortier, R. Haltli, C. Tigges, and M. G. Blain, “Design, fabrication and experimental demonstration of junction surface ion traps,” *New Journal of Physics*, vol. 13, p. 075018, Jul 2011.

- [166] K. J. Ross and B. Sonntag, “High temperature metal atom beam sources,” *Review of Scientific Instruments*, vol. 66, no. 9, pp. 4409–4433, 1995.
- [167] C. B. Alcock, V. P. Itkin, and M. K. Horrigan, “Vapour pressure equations for the metallic elements: 2982500k,” *Canadian Metallurgical Quarterly*, vol. 23, no. 3, pp. 309–313, 1984.
- [168] T. G. Ballance, J. F. Goodwin, B. Nichol, L. J. Stephenson, C. J. Ballance, and D. M. Lucas, “A short response time atomic source for trapped ion experiments,” *Review of Scientific Instruments*, vol. 89, no. 5, p. 053102, 2018.
- [169] T. Navickas, *Towards high-delity microwave driven multi-qubit gates on microfabricated surface ion traps*. PhD thesis, University of Sussex, 2017.
- [170] N. A. Riza, “Scanning heterodyne optical interferometers,” *Review of Scientific Instruments*, vol. 67, no. 7, pp. 2466–2476, 1996.
- [171] E. A. Donley, T. P. Heavner, F. Levi, M. O. Tataw, and S. R. Jefferts, “Double-pass acousto-optic modulator system,” *Review of Scientific Instruments*, vol. 76, no. 6, p. 063112, 2005.
- [172] S. Ejtemaee, R. Thomas, and P. C. Haljan, “Optimization of yb^+ fluorescence and hyperfine-qubit detection,” *Physical Review A*, vol. 82, p. 063419, Dec 2010.
- [173] J. D. Siverns, *Yb ion trap experimental set-up and two-dimensional ion trap surface array design towards analogue quantum simulations*. PhD thesis, University of Sussex, 2011.
- [174] R. Noek, G. Vrijsen, D. Gaultney, E. Mount, T. Kim, P. Maunz, and J. Kim, “High speed, high fidelity detection of an atomic hyperfine qubit,” *Optics Letters*, vol. 38, pp. 4735–4738, Nov 2013.
- [175] M. Acton, K.-A. Brickman, P. C. Haljan, P. J. Lee, L. Deslauriers, and C. Monroe, “Near-perfect simultaneous measurement of a qubit register,” *Quantum Info. Comput.*, vol. 6, p. 465482, Sep 2006.
- [176] A. H. Burrell, *High Fidelity Readout of Trapped Ion Qubits*. PhD thesis, Exeter College, Oxford, 2010.
- [177] R. N. Tubbs, *Lucky Exposures: Diffraction limited astronomical imaging through the atmosphere*. PhD thesis, Cambridge University, 2003.
- [178] A. H. Burrell, D. J. Szwer, S. C. Webster, and D. M. Lucas, “Scalable simultaneous multiqubit readout with 99.99% single-shot fidelity,” *Physical Review A*, vol. 81, p. 040302, Apr 2010.

- [179] S. Weidt, *Towards microwave based ion trap quantum technology*. PhD thesis, University of Sussex, 2013.
- [180] C. Kimme, D. Ballard, and J. Sklansky, “Finding circles by an array of accumulators,” *Commun. ACM*, vol. 18, pp. 120–122, Feb 1975.
- [181] D. T. C. Allcock, T. P. Harty, H. A. Janacek, N. M. Linke, C. J. Ballance, A. M. Steane, D. M. Lucas, R. L. Jarecki, S. D. Habermehl, M. G. Blain, D. Stick, and D. L. Moehring, “Heating rate and electrode charging measurements in a scalable, microfabricated, surface-electrode ion trap,” *Applied Physics B*, vol. 107, no. 4, pp. 913–919, 2012.
- [182] D. J. Berkeland, J. D. Miller, J. C. Bergquist, W. M. Itano, and D. J. Wineland, “Minimization of ion micromotion in a Paul trap,” *Journal of Applied Physics*, vol. 83, no. 10, pp. 5025–5033, 1998.
- [183] J. Keller, H. Partner, T. Burgermeister, and T. Mehlstäubler, “Precise determination of micromotion for trapped-ion optical clocks,” *Journal of Applied Physics*, vol. 118, no. 10, p. 104501, 2015.
- [184] S. C. Doret, J. M. Amini, K. Wright, C. Volin, T. Killian, A. Ozakin, D. Denison, H. Hayden, C.-S. Pai, R. E. Slusher, and A. W. Harter, “Controlling trapping potentials and stray electric fields in a microfabricated ion trap through design and compensation,” *New Journal of Physics*, vol. 14, p. 073012, Jul 2012.
- [185] A. Härter, A. Krüchow, A. Brunner, and J. Hecker Denschlag, “Minimization of ion micromotion using ultracold atomic probes,” *Applied Physics Letters*, vol. 102, no. 22, p. 221115, 2013.
- [186] U. Tanaka, K. Masuda, Y. Akimoto, K. Koda, Y. Ibaraki, and S. Urabe, “Micromotion compensation in a surface electrode trap by parametric excitation of trapped ions,” *Applied Physics B*, vol. 107, pp. 907–912, Jun 2012.
- [187] W. K. Hensinger, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, L. Deslauriers, C. Monroe, and J. Rabchuk, “T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation,” *Applied Physics Letters*, vol. 88, no. 3, p. 034101, 2006.
- [188] R. B. Blakestad, C. Ospelkaus, A. P. VanDevender, J. M. Amini, J. Britton, D. Leibfried, and D. J. Wineland, “High-fidelity transport of trapped-ion qubits through an X-junction trap array,” *Physical Review Letters*, vol. 102, p. 153002, Apr 2009.

- [189] C. Decaroli, R. Matt, R. Oswald, C. J. Axline, M. Ernzer, J. Flannery, S. Ragg, and J. P. Home, “Design, fabrication and characterisation of a micro-fabricated stacked-wafer segmented ion trap with two x-junctions.,” *Quantum Science and Technology*, vol. 6, p. 044001, Jul 2021.
- [190] K. Wright, J. M. Amini, D. L. Faircloth, C. Volin, S. C. Doret, H. Hayden, C.-S. Pai, D. W. Landgren, D. Denison, T. Killian, R. E. Slusher, and A. W. Harter, “Reliable transport through a microfabricated x-junction surface-electrode ion trap,” *New Journal of Physics*, vol. 15, no. 3, p. 033004, 2013.
- [191] R. Blakestad, C. Ospelkaus, A. VanDevender, J. Wesenberg, M. Biercuk, D. Leibfried, and D. Wineland, “Near-ground-state transport of trapped-ion qubits through a multidimensional array,” *Physical Review A*, vol. 84, no. 3, p. 032314, 2011.
- [192] R. B. Blakestad, *Transport of Trapped-Ion Qubits within a Scalable Quantum Processor*. PhD thesis, University of Colorado, 2010.
- [193] H. Kaufmann, T. Ruster, C. T. Schmiegelow, F. Schmidt-Kaler, and U. G. Poschinger, “Dynamics and control of fast ion crystal splitting in segmented Paul traps,” *New Journal of Physics*, vol. 16, p. 073012, Jul 2014.
- [194] Y. Aikyo, G. Vrijsen, T. W. Noel, A. Kato, M. K. Ivory, and J. Kim, “Vacuum characterization of a compact room-temperature trapped ion system,” *Applied Physics Letters*, vol. 117, no. 23, p. 234002, 2020.
- [195] “Openqasm 3.x live specification.” <https://qiskit.github.io/openqasm/>. Accessed: 2021-03-25.

Appendix A

Z stabilizer circuit in the QCCD syntax

The most constraining stabilizer of the surface code to implement in a QCCD architecture, when limiting to two ions per zone, is the Z stabilizer. The trapped ion compatible circuit, displayed in Figure A.1, is obtained by applying the tools presented in Section 5.3:

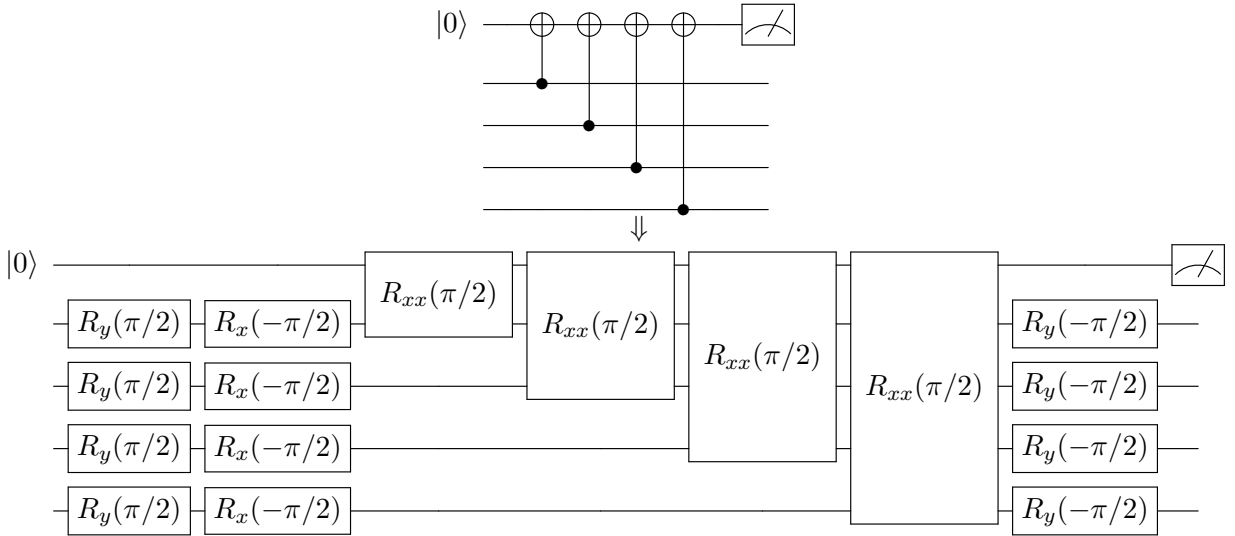


Figure A.1: Compilation of the Z stabilizer circuit, defined with CNOT gates, into the trapped ion gate set, composed of R_x , R_y and R_{xx} gates.

Initially, all the qubits are prepared. To compile the stabilizer into QCCD code efficiently, the ancilla qubit is assigned to the gate zone first. After this, each data qubit is sent to the gate zone where all the corresponding gates are applied, after which the qubit is removed to a memory zone and the next data qubit is introduced to the gate zone. This process repeats sequentially, one by one, for all of the data qubits.

```

1 #Initialization
2 Init | S N N E E #ancilla in South arm
3 Prepare | E[2]
4 Shuttle | E[1/2]_W[0]
5 Shuttle | E[1/1]_W[1]
6 Shuttle | N[1/2]_E[0]
7 Shuttle | N[1/1]_E[1]
8 Prepare | E[2]
9 Shuttle | E[1/2]_N[0]
10 Shuttle | S[1/1]_E[1]
11 Swap | E[2]
12 Prepare | E[2]
13 #Quantum circuit
14 Ry(pi/2) | E[1/2]
15 Rx(7*pi/2) | E[1/2]
16 Rxx(pi/2) | ( E[1/2], E[2/2])
17 Ry(7*pi/2) | E[1/2]
18 Shuttle | E[1/2]_S[0]
19 Shuttle | N[1/1]_E[1]
20 Ry(pi/2) | E[1/2]
21 Rx(7*pi/2) | E[1/2]
22 Rxx(pi/2) | ( E[1/2], E[2/2])
23 Ry(7*pi/2) | E[1/2]
24 Shuttle | E[1/2]_S[1]
25 Shuttle | W[1/2]_E[1]
26 Ry(pi/2) | E[1/2]
27 Rx(7*pi/2) | E[1/2]
28 Rxx(pi/2) | ( E[1/2], E[2/2])
29 Ry(7*pi/2) | E[1/2]
30 Shuttle | E[1/2]_N[1]
31 Shuttle | W[1/1]_E[1]
32 Ry(pi/2) | E[1/2]
33 Rx(7*pi/2) | E[1/2]
34 Rxx(pi/2) | ( E[1/2], E[2/2])
35 Ry(7*pi/2) | E[1/2]
36 #Measurement of the ancilla
37 Shuttle | E[1/2]_W[0]
38 Measure | E 0

```

Appendix B

Shuttling waveform format

This section transcribes part of the document formatting the waveform files for ion shuttling, generated by the simulation python framework written by Dr Sam Hile, and read by the DC concatenation module to assemble and upload the waveforms run by the DACs. That document was written by Sahra Kulmiya, under my supervision and based on my format design.

B.1 Introduction

Although each chip is very unique, which makes specifying a general format difficult, a format is very important for many reasons. There are many experimental control systems in the lab, therefore using the same code to read shuttling sequences would be advantageous. We may also want to try multiple shuttling protocols for the same chip, or concatenate them together, and we should be able to switch them easily.

The waveform consists of voltages applied to several electrodes and varying over time. Either for physical reasons, due to electrodes wirebonded together, or due to optimization, to add more constraints on finding an optimal solution, the key components for simulating the potential on the chip are not the electrodes, but a cluster of electrodes called knobs.

A general specification is outlined in the next section, and an example is given further below.

B.2 Specification

Each shuttling sequence file should be named in the following way: **ChipName_TypeofShuttleVN.json**.

```

1 {"Name_of_Chip" : *string_here*,
2  "Author" : *string_here* ,
3  "Number_of_Knobs" : *integer_here*, # 6
4  "Number_of_Steps" : *integer_here*, # 50
5  "Knob_to_Electrode_Example" : *dict_here*, # { K 1 :[e1,e4], K 2 :[e2,e5,e11] }
6  "Sequence_Description" : *string_here*,
7  "Generation_Time" : *datetime_here*,
8  "Comments" : *string_here*,
9
10 "Voltage_Values" : [{"key_1": float, # list of length *Number_of_Steps*
11                      "key_2": float, # each element a dict matching the Knobs
12                      "key_3": float, ..},
13                      {...},{...}]
14 "Info_Values" : [{"position": float, # list of length *Number_of_Steps*
15                  "sec_freq": float, # each element a dict matching the Knobs
16                  }, {...},{...},...}]

```

the entries are now detailed, with the distinction between mandatory, recommended and optional indicated by the **must**, **should** and **could** keywords.

- The name/specification of the chip being tested **must** be included in the file and in the name of the file.
- The author/creator of the shuttling protocol **must** also be included.
- The “Knobs” key is a generic description of the number of electrodes used in the shuttling sequence. The number of Knobs is arbitrary for each shuttling protocol. For example, a linear sequence can use 5 knobs, and an X junction can use between 8 to 12 Knobs. This **must** be included.
- The number of steps corresponds to the number of dictionaries generated. This **must** be included.
- The “knob to electrode example” entry **should** describe how the knobs and electrodes are grouped for this sequence.
- “Sequence description” **could** give a short description of the shuttling sequence the file represents, i.e. swapping, merging, corner shuttling etc.
- “Generation time” is where the date of creation of the shuttling file **could** be entered.
- The comments section is where the user could write further information about their chip and **should** be as detailed as possible. This can also potentially include a link to a file that displays a labelled image of the chip and the electrode and also a short description of the electrode labeling used as the keys.

- “Voltage values”: The voltages are here introduced as key:value pairs, also known as dictionaries, in order to lessen ambiguity in any shuttling file. This is vital because in particularly complex shuttling protocols, several knobs are involved at a time and indicating which knobs are involved becomes important. The size of this list should be the same as the number of steps indicated above. These **must** be included in any file.
- “Info values”: describes further parameters that could be included in the shuttling sequence such as the particular position of the shuttling path, e.g relative to the z,x,y directions, and the secular frequency of the shuttling sequence, as well as any other information. The size of this list should be the same as the number of steps indicated above. These **must** be included in any file.

B.3 Example

The following example is given for the linear shuttling waveform over the distance of one electrode, approximately 200 μm , thus involving five pairs of electrodes, or five knobs. Only the first points of the waveform are represented:

```

1 { "Name_of_Chip" : "DazzaWP2W1XA2",
2   "Author" : "D. Bretaude",
3   "Number_of_Knobs": 5,
4   "Number_of_Steps" : 101,
5   "Knob_to_Electrode_Example" : {L1:[e1,e2],L2:[e3,e4],L3:[e5,e6],L4:[e7,e8], L5:[e9,e10]}
6   "Sequence_Description" : "This sequence describes a linear shuttle, starting between L3
   and L4, and stopping between L2 and L3.",
7   "Generation_Time" : "March_2020",
8   "Comments" : "The labelled chip can be found in Directory/Filename/..",
9
10  "Voltage_Values" : [{ "L_5":3.4167459035780268,
11                        "L_4":-1.9404774268418745,
12                        "L_3":-1.9639135757523316,
13                        "L_2":3.4045037309378987,
14                        "L_1":0.09814417877801251},...],
15  "Info_Values" : [{ "zrelative": -100.0 ,
16                    "sec_freq (kHz)" : 110},...}]

```

the information e1,e2,... corresponds to the electrode label for one specific case although that waveform can be applied to any section of a linear trap. In “Info_Values”, each datapoint is associated to a “zrelative” keyword indicating the relative spatial position in μm , and with “sec_freq (kHz)”, the axial secular frequency.