EXPERIMENTAL CONTROL AND BENCHMARKING

for

SINGLE-QUBIT TRAPPED-ION TRANSPORT GATES

David P. Nadlinger

March 2016

MSc Physics Semester Project

completed in the

Trapped Ion Quantum Information Group
Institute for Quantum Electronics
ETH Zürich

This thesis was typeset using the LaTeX software originally developed by Leslie Lamport based on TeX created by Donald Knuth, and the Minion Pro typeface designed by Robert Slimbach.

The layout is inspired by Robert Bringhurst's book *The Elements of Typographic Style*, and builds on a public-domain document class by Erwin Uggedal. Plot figures have been created using the Julia programming language and the matplotlib package.

Abstract

Trapped atomic ions are currently one of the most promising candidates for the large-scale realisation of quantum information processing. This report describes design and implementation of systems for stable and precise control of $^{9}Be^{+}$ and $^{40}Ca^{+}$ ions, in particular for shuttling them along the axis of a segmented linear Paul trap, and techniques for characterising the fidelity of quantum operations.

The static single-qubit logic gates in the present experimental setup are analysed using randomised benchmarking and gate set tomography. For the $\left|4^2S_{1/2}, m = \frac{1}{2}\right\rangle \leftrightarrow \left|3^2D_{5/2}, m = \frac{3}{2}\right\rangle$ optical qubit in $^{40}Ca^{+}$, a fidelity of 0.99983(1) per computational gate is found, limited by decoherence due to magnetic field fluctuations. Laser intensity noise limits the fidelity of operations on the field-insensitive $\left|2^2S_{1/2}, F = 1, m_F = 1\right\rangle \leftrightarrow \left|2^2S_{1/2}, F = 2, m_F = 0\right\rangle$ qubit in $^{9}Be^{+}$ to 0.9951(3).

CONTENTS

INTRODUCTION 1

When Yuri Manin and Richard Feynman first discussed the potential of quantum computation in the early 1980s [1], the experimental realisation of such schemes seemed all but impossible. In the thirty years since, however, quite monumental advances in the control of quantum systems have been made – indeed, experimental quantum information science has turned from the once distant idea into a thriving and commercially relevant field. The creation and manipulation of single qubit states is part of the standard repertoire of experiments in many different systems now, ranging from linear optics over superconducting electronics and diamond lattice defects to single trapped atoms. [2] For trapped ions in particular, all the elements necessary for universal computation in the circuit model have been demonstrated, and to an accuracy exceeding what is though to be the lower bound necessary to completely suppress errors and decoherence using quantum error correction schemes. [3–5]

One big challenge for all experimental realisations of quantum computing, and for trapped-ion experiments in particular, lies in scaling up the systems. Increasing the number of qubits to values relevant for information processing applications is technically challenging, as exquisite control over each of the qubits as well as the coupling between them has to be maintained. One approach for achieving this in trapped-ion qubits has been proposed by the group around David Wineland at NIST. [4] It consists of a two-dimensional array of micro-fabricated ion traps with segmented electrodes, making up many "logic" zones between which ions can be shuttled by changing the electric potentials applied to the segments. This can be used for physically transporting quantum information between the different zones, but has also been used to split ion chains for individual addressing and other more complex operations. Velocities of several tens of meters per second have been achieved without significant heating. [6]

Following a proposal by Dietrich Leibfried [7], Ludwig de Clercq has led an effort in our group to directly utilise transport in the implementation of the quantum logic operations, which are conventionally done with the ions at rest. [8] In this scheme, which we refer to as *transport gates*, transport through a stationary laser beam is used to apply the modulated classical light field instead of directly switching the beam on and off. We have demonstrated [9] that this scheme can be used to recycle a single static laser beam to apply concurrent logic operations in several trap zones. This way, the requirements regarding laser sources

and acousto-optical control are lowered, which opens an avenue towards scalable ion trap quantum computing.[1]

1. The (classical) control of the electrode voltages is inherently local and can easily be scaled up using standard integrated-circuit technology.

The work presented in this report comprises two main parts. The initial goal for the project was to implement branching functionality for the custom digital waveform generator developed for transport experiments in the group. To provide easy access to different transport sequences as part of the usual experimental workflow, I also integrated it into the main experimental control system. This was the beginning of a series of other improvements to the monitoring and sequencing systems, which were used as part of the experiments described in [9–11]. These technical enhancements are discussed in more detail in chapter 3.

A large part of the these improvements were motivated by the desire to better characterise the transport gate operations on $^{9}Be^{+}$ ions. As a first step, two applicable techniques, randomised benchmarking and gate set tomography, were implemented for the conventional, static operations on the $^{40}Ca^{+}$ and $^{9}Be^{+}$ qubits. A theoretical discussion of their operation is provided in chapter 2 along with a short description of the experimental setup used in our group. The results of the experiments on the static qubits are presented in chapter 4.

Together with the improved transport hardware, these results demonstrate that experimental control can easily be performed to the level necessary for carrying out a quantitative analysis of the performance of the transport gates in our experiment.

BACKGROUND 2

To provide some context for the discussion of the improvements to the technical control infrastructure and the experimental results in the rest of this report, this chapter briefly reviews the most commonly used mathematical framework to describe quantum operations, outlines some techniques for their characterisation, and discusses one particular experimental setup in the Trapped Ion Quantum Information group.

2.1 QUANTUM STATES AND CHANNELS

Consider the state $|\psi\rangle$ of a generic quantum system. If the system is finite-dimensional, states can be described[1] as complex vectors of unit length in a d-dimensional complex Hilbert space,

$$|\psi\rangle \in \mathcal{H} \cong \mathbb{C}^d, \quad \|\psi\|^2 = \langle\psi|\psi\rangle = 1. \tag{2.1}$$

1. A comprehensive treatment of the following material can be found in any textbook on quantum information theory, e.g. [12].

As per the postulates of quantum mechanics, the evolution of the state under a given Hamiltonian follows the Schrödinger equation and can be expressed using a unitary propagator U:

$$|\psi(t)\rangle = U(t, t_0)\,|\psi(t_0)\rangle \tag{2.2}$$

For a finite-dimensional system, the propagator for a certain time interval is simply given by a unitary matrix $U \in \mathrm{U}(d)$.

The concept of *density operators* extends the description from pure states to (classical) mixtures of such states, as they arise in situations where there is incomplete knowledge about the system. A density operator ρ on a Hilbert space $\mathcal{H}$ is given by an endomorphism on $\mathcal{H}$ that is normalised (in the Hilbert-Schmidt norm) and positive, i.e.

$$\rho \in \mathrm{End}(\mathcal{H}), \quad \rho \geq 0, \quad \mathrm{tr}\,\rho = 1. \tag{2.3}$$

The set of density operators on $\mathcal{H}$ will be denoted as $\mathcal{S}(\mathcal{H})$.

From the identification of a pure state ψ with the orthonormal projector

$$\mathrm{P}_\psi = |\psi\rangle\langle\psi|, \tag{2.4}$$

it is clear that the transformation corresponding to the above unitary U is given by

$$\rho \mapsto U\rho U^\dagger. \tag{2.5}$$

Unitary transformations of this form, however, do not cover all operations consistent with the postulates of quantum mechanics. An important example are the consequences of imperfect control of the interactions used to manipulate an experimental system, whether due to unwanted coupling to the environment or differences in the classical control parameters between different realisations of a given operation. The most general class of physical operations is given by *completely positive trace-preserving (CPTP) maps*.

Consider two Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$. A CPTP map between the respective density operators is given by a linear operator

$$\mathcal{E} \in \mathrm{Hom}(\mathrm{End}(\mathcal{H}_A), \mathrm{End}(\mathcal{H}_B))$$

such that

$$\mathrm{tr}\,\mathcal{E}(\rho) = 1 \quad \text{for } \mathrm{tr}\,\rho = 1, \tag{2.6}$$

and for all Hilbert spaces $\mathcal{H}_R$

$$\rho' = (\mathcal{E} \otimes \mathrm{id}_{\mathcal{H}_R})(\rho_A \otimes \rho_R) \geq 0 \quad \text{for } (\rho_A \otimes \rho_R) \geq 0. \tag{2.7}$$

Equation (2.6), the *trace preservation* property, is clearly necessary to map density operators to density operators. Of course, a map between density operators must certainly also perserve their positivity. Somewhat surprisingly, however, simple positivity of two such maps $\mathcal{E}$ and $\mathcal{F}$ is not sufficent to guarantee positivity of the composed map $\mathcal{E} \otimes \mathcal{F}$. *Complete positivity*, as stated in (2.7), remedies this.

Such operations $\mathcal{E}$ are also referred to as *superoperators* or *quantum channels, processes* or *operations*.[2] In the following, we will restrict our discussion to CPTP endomorphisms, that is $\mathcal{H}_A = \mathcal{H}_B = \mathcal{H}$. Such maps naturally lend themselves to describing operations on a fixed number of qubits in the laboratory.

2. Some authors also use one or more of these terms to refer to operators that might reduce the trace.

2.1.1 *Representations of CPTP Maps*

Particularly in abstract treatments, CPTP maps are often specified in their *operator-sum* or *Kraus representation*. For $\{M_k\}_k \subset \mathrm{End}(\mathcal{H})$ with $\sum_k {M_k}^\dagger M_k = \mathbb{1}$, the map $\mathcal{E} \in \mathrm{End}(\mathcal{H})$ given by

$$\mathcal{E}(\rho) = \sum_k M_k \,\rho\, {M_k}^\dagger \tag{2.8}$$

is completely positive and trace preserving. Moreover, all CPTP maps can be written in this form, which is analogous to the post-measurement state for a set of POVM elements M_k or the weighted average of a set of unitary transformations (2.5). The operator-sum representation is not unique: Another set of Kraus operators $\{N_k\}_k$ describes the same operation $\mathcal{E}$ iff $N_k = \sum_l V_{kl} M_l$ for some unitary matrix V.

The Kraus representation has several convenient properties; for instance, it naturally guarantees positivity irrespective of the choice

for $\{M_k\}_k$. For numerical computations, however, it can be more appropriate to view the map $\mathcal{E}$ directly as an endomorphism on the d^2-dimensional vector space $\mathrm{End}(\mathcal{H})$. In this representation, density operators are simply column vectors in $\mathbb{C}^{d^2}$, and CPTP maps are described by $d^2 \times d^2$ matrices. Application of the map is then given by matrix-vector multiplication

$$\mathcal{E}(\rho) = E\,\rho, \tag{2.9}$$

where E is the matrix corresponding to $\mathcal{E}$. This is sometimes referred to as the *superoperator* or *natural* representation, and has the convenient property that the matrix for a composite operation $\mathcal{E} = \mathcal{E}_2 \circ \mathcal{E}_1$ is simply given by the product $E = E_2 \cdot E_1$.

Considering the isomorphism $\mathrm{End}(\mathcal{H}) \cong \mathcal{H}^* \otimes \mathcal{H}$, the map describing pre- and post-multiplication with two given matrices,

$$\mathcal{E}_{R,S}(\rho) = R\,\rho\,S, \tag{2.10}$$

can be expressed in the chosen coordinates as

$$E_{R,S} = S^T \otimes R. \tag{2.11}$$

In particular, the application of a unitary propagator U as in (2.5) thus corresponds to the superoperator

$$E_U = \overline{U} \otimes U. \tag{2.12}$$

2.1.2 *Example: $\pi/2$- and π-rotations*

As basic building blocks for single-qubit quantum operations (where the Bloch sphere picture is used to represent $\mathcal{H} = \mathbb{C}^2$), the gates that rotate the Bloch state vector by angles of $\pi/2$ and π radians about the coordinate axes are of particular interest for quantum information experiments (see section 2.2). Referred to as *π-pulses* in the tradition of the early nuclear magnetic resonance experiments, the π-rotations are given by the *Pauli matrices*

$$\begin{aligned}
\sigma_x &= |0\rangle\langle 1| + |1\rangle\langle 0| &&= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\
\sigma_y &= -\mathrm{i}\,|0\rangle\langle 1| + \mathrm{i}\,|1\rangle\langle 0| &&= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\
\sigma_z &= |0\rangle\langle 0| - |1\rangle\langle 1| &&= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}
\end{aligned}. \tag{2.13}$$

The Pauli matrices (scaled by 1/2) are also the generators of arbitrary rotations around the axes

$$R_i(\theta) = \mathrm{e}^{-\frac{\mathrm{i}}{2}\theta\sigma_i} \quad \text{for } i = x, y, z, \tag{2.14}$$

from which the unitaries for $\pi/2$-rotations can be derived as $R_i(\frac{\pi}{2})$.

The explicit matrix form of the corresponding superoperators depends on the choice of basis for $\mathcal{S}(\mathcal{H})$. A commonly used convention

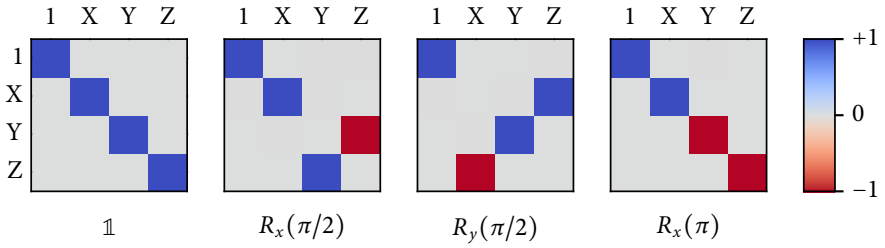

Figure 2.1: Superoperator matrices in the Pauli basis for some common single-qubit rotations.

is to regard $\mathcal{S}(\mathbb{C}^2)$ as a real vector space, endowed with the Hilbert-Schmidt norm and the orthonormal basis $\{\mathbb{1}, \sigma_x, \sigma_y, \sigma_z\}/\sqrt{2}$. In this basis, the $\pi/2$- and π-rotations take a particularly simple form. The entries of the superoperator matrices for a selection of them are shown in figure 2.1.

2.1.3 *Fidelity Measures*

To evaluate how close a given experimental operation is to an ideal target gate, it is useful to define some kind of distance measure between different quantum channels. One widely used distance (or rather closeness) measure starts from the *fidelity*[3] between two pure states $|\varphi\rangle$ and $|\psi\rangle$,

$$\mathrm{F}(\varphi, \psi) = |\langle\varphi|\psi\rangle|^2 . \tag{2.15}$$

3. Some authors understand the term to refer to the square root of this definition. Notably, Michael Nielsen uses the latter in his seminal textbook [12], but has since advocated use of (2.15) in publications such as [13].

This definition can be lifted into the density operator framework using the trace norm,

$$\mathrm{F}(\rho, \sigma) = \left\| \sqrt{\rho}\sqrt{\sigma} \right\|_1^2 , \tag{2.16}$$

where $\|S\|_1 = \mathrm{tr}\sqrt{S^\dagger S}$. Restating this as

$$\mathrm{F}(\rho, \sigma) = \left(\mathrm{tr}\sqrt{\sqrt{\sigma}\rho\sqrt{\sigma}} \right)^2 \tag{2.17}$$

one easily evaluates the fidelity to a pure state $\sigma = |\psi\rangle\langle\psi|$ as

$$\mathrm{F}(\rho, |\psi\rangle\langle\psi|) = \langle\psi|\,\rho\,|\psi\rangle . \tag{2.18}$$

By itself, F is not a metric, but can be used to define one in multiple straightforward ways (cf. [13]).

A natural interpretation of the fidelity as defined in (2.15) is the probability to observe the system $|\varphi\rangle$ in a given target state $|\psi\rangle$. The quantity

$$r_{\mathcal{E}}(|\psi\rangle) = 1 - \mathrm{F}\left(\mathcal{E}(|\psi\rangle\langle\psi|), |\psi\rangle\langle\psi|\right) = 1 - \langle\psi|\,\mathcal{E}(|\psi\rangle\langle\psi|)\,|\psi\rangle \tag{2.19}$$

can thus be interpreted as the *error probability* for an operation $\mathcal{E}$ with respect to the identity operation, given an input state $|\psi\rangle$. The *average*

error probability or *process infidelity* of $\mathcal{E}$ with respect to the identity can thus be defined analogously as

$$r_{\mathcal{E}} = 1 - F_{\mathcal{E}} = 1 - \int_{\mathcal{H}} \langle\psi| \mathcal{E}(|\psi\rangle\langle\psi|) |\psi\rangle \, \mathrm{d}\psi, \tag{2.20}$$

where the integration is performed over the unit sphere of pure states in $\mathcal{H}$ with the normalised Haar measure $\mathrm{d}\psi$. As shown in [14], the average fidelity for an operation $\mathcal{G}$ compared to an ideal unitary target gate $U_0 \in \mathrm{U}(n)$ can be calculated accordingly as

$$\begin{aligned} F(\mathcal{G}, U_0) &= F_{U_0^\dagger \mathcal{G} U_0} \\ &= \frac{1}{n(n+1)} \left(\mathrm{tr}\left(\sum_k {M_k}^\dagger M_k \right) + \sum_k |\mathrm{tr}\,(M_k)|^2 \right), \end{aligned} \tag{2.21}$$

where the $\{M_k\}_k$ are derived from a set of Kraus operators $\{G_k\}_k$ for $\mathcal{G}$ as $M_k = {U_0}^\dagger G_k$.

The (squared) fidelity, however, is not the only possible way of measuring the similarity of quantum states. Another prominent metric is that of the *trace distance* between quantum states, which is based on their distinguishability, and its extension to the entanglement-assisted (single-shot) distinguishability of CPTP maps, the *diamond norm*. Due to its relation to worst-case behaviour, it is frequently used in threshold statements about the feasibility of quantum error correction. [15] While there currently does not seem to be a single clearly superior measure, it is often possible to bound one metric by a combination of others. [13]

2.2 RANDOMISED BENCHMARKING

Once all the parameters describing a quantum operation are known, derived quantities such as its fidelity or diamond norm distance to a given target operation can easily be calculated. Techniques for estimating the coefficients for a given quantum state or process are known as *tomography*, and one such example is discussed in section 2.3. However, $O(16^n)$ real parameters are required to describe a CPTP map between density operators of a n-qubit system $(\mathbb{C}^2)^{\otimes n}$. Consequently, tomography is an involved process even for a small number of qubits and outright infeasible for larger systems in the general case.

In many experimental settings, however, the performed operations are already known to good approximation and can be calibrated to match a set of target gates by independent means. In such cases, the most interesting quantity is often the overall process infidelity or error rate, particular to provide a point of comparison to theoretical bounds on fault-tolerant quantum computing techniques. It stands to reason that it might be easier to directly determine this quantity without fully reconstructing the constituent quantum operations first.

There indeed exists a family of techniques that achieves this, *randomised benchmarking*. It relies on long strings chosen at random from a

C_0	$\mathbb{1}$	C_8	$R_y(-\frac{\pi}{2})$	C_{16}	$R_x(+\frac{\pi}{2})\ R_z(-\frac{\pi}{2})$
C_1	$R_x(\pi)$	C_9	$R_z(-\frac{\pi}{2})$	C_{17}	$R_z(\pi)\ R_y(-\frac{\pi}{2})$
C_2	$R_y(\pi)$	C_{10}	$R_z(\pi)\ R_x(+\frac{\pi}{2})$	C_{18}	$R_z(+\frac{\pi}{2})\ R_y(-\frac{\pi}{2})$
C_3	$R_z(\pi)$	C_{11}	$R_z(\pi)\ R_x(-\frac{\pi}{2})$	C_{19}	$R_z(-\frac{\pi}{2})\ R_y(+\frac{\pi}{2})$
C_4	$R_x(+\frac{\pi}{2})$	C_{12}	$R_z(+\frac{\pi}{2})\ R_x(\pi)$	C_{20}	$R_z(\pi)\ R_y(+\frac{\pi}{2})$
C_5	$R_y(+\frac{\pi}{2})$	C_{13}	$R_z(-\frac{\pi}{2})\ R_x(\pi)$	C_{21}	$R_z(-\frac{\pi}{2})\ R_x(+\frac{\pi}{2})$
C_6	$R_z(+\frac{\pi}{2})$	C_{14}	$R_z(+\frac{\pi}{2})\ R_x(+\frac{\pi}{2})$	C_{22}	$R_z(+\frac{\pi}{2})\ R_y(+\frac{\pi}{2})$
C_7	$R_x(-\frac{\pi}{2})$	C_{15}	$R_z(+\frac{\pi}{2})\ R_x(-\frac{\pi}{2})$	C_{23}	$R_z(-\frac{\pi}{2})\ R_x(-\frac{\pi}{2})$

Figure 2.2: The 24 elements of the single-qubit Clifford group $\mathcal{C}$, decomposed into π- and $\pi/2$-rotations along the coordinate axes of the Bloch sphere. [17]

certain set of gates, and has risen to wide-spread popularity in the experimental quantum information community due to its relative simplicity and robustness after having first been demonstrated in trapped ions at NIST. [16]

2.2.1 *The Pauli and Clifford Groups*

Together with the identity, the Pauli matrices form a group under multiplication, the *Pauli group*

$$\mathcal{P} = \{\pm\mathbb{1}, \pm\sigma_x, \pm\sigma_y, \pm\sigma_z\}, \tag{2.22}$$

which can be represented by the π-rotations on the Bloch sphere. $\pi/2$-rotations are not elements of the Pauli group, but can be taken to be members of its normaliser, the *Clifford group* $\mathcal{C}$:

$$\mathcal{C} = \left\{C \in \mathrm{U}(2) \;\middle|\; C\,\mathcal{P}\,C^\dagger = \mathcal{P}\right\} \,/\, \mathrm{U}(1). \tag{2.23}$$

In other words, the Clifford group permutes the elements of the Pauli group when it acts on it by conjugation – its elements permute the six states on the coordinate axes of the Bloch sphere, the eigenvalues of the Pauli matrices. $\mathcal{C}$ consists of 24 elements, which are enumerated explicitly in figure 2.2. The definition can be extended analogously to multi-qubit systems starting from the n-qubit Pauli group $\mathcal{P}_n = \mathcal{P} \otimes \ldots \otimes \mathcal{P}$.

The Clifford group elements are of particular interest for fault-tolerant quantum computation because they are sufficient to implement state preparation and decoding for a widely-studied class of quantum error correction codes, the so-called stabiliser codes. Additionally, augmenting it with just one other single-qubit rotation (e.g. the $\pi/8$-gate) is enough to obtain a universal set of gates. At the same time, however, the stabiliser formalism still allows efficient simulation of such circuits as per the Gottesman-Knill theorem. [18]

2.2.2 *The Randomised Benchmarking Protocol*

A randomised benchmarking experiment on a single qubit executes a number of individual sequences that consist of state preparation in $|0\rangle$,

qubit manipulation and readout in the computational (i.e. σ_z) basis. For each such sequence, a number of gates $(G_1, \ldots, G_{L-1}) \subset \mathcal{C}$ is first chosen at random from the Clifford group. The last gate G_L is then selected such that the final state $G_L \cdot \ldots \cdot G_1 \, |0\rangle$ again is an eigenvalue of σ_z.[4] If the gates G_i were indeed implemented perfectly, the subsequent measurement would thus by construction take one fixed value.

4. The expected outcome ($|0\rangle$ or $|1\rangle$) can be randomised to avoid systematic errors due to asymmetric measurement imperfections. Equivalently, the process could also be described as choosing G_L such that the gate sequence is strictly equal to the identity, followed by a random measurement.

The probability of obtaining that expected outcome (estimated from a certain number of repetitions) is then averaged over a number N of different gate sequences of the same length. If the quantum circuit is not implemented perfectly, the outcome will sometimes deviate from the expected result, either because a different state is produced due to a stationary error (e.g. due to miscalibration of the gate rotation angles), or because of fluctuations between individual realisations of the given experiment. Intuitively, the random selection of gates ensures that any such errors are sampled fairly. This can be formalised starting from the observation that the Clifford group is a unitary two-design, and thus twirling a given error map Λ over it,

$$\frac{1}{|\mathcal{C}|} \sum_k {C_k}^\dagger \circ \Lambda \circ C_k = \Lambda_{\text{dep}}, \tag{2.24}$$

yields the depolarising channel Λ_{dep} with the same average fidelity F as Λ. Randomised benchmarking can be viewed (in the limit of large N) as the L-fold concatenation of such twirls. [19]

If the procedure is repeated for a number of different lengths L, the average gate fidelity can thus be estimated independent of state preparation and measurement errors by fitting an exponential model to the observed mean probabilities (discussed more carefully in [19]). Since the estimate is obtained from the decay between different sequence lengths and not the absolute observed error, it is notably insensitive to errors in state preparation and measurements. The protocol can be implemented efficiently for large number of qubits. [20]

When attempting to make rigorous statements about the achieved gate fidelities and their uncertainties, though, the effects of finite sampling ($N < \infty$) cannot be ignored. This is particularly important in the presence of non-Markovian noise, as discussed in [17] and [21].

2.3 GATE SET TOMOGRAPHY

For operations on one or two qubits, full estimation of all parameters in the CPTP description of a gate is computationally feasible and offers additional insights[5] into the imperfections at play, for example allowing to distinguish between coherent and incoherent errors. Conceptually, a quantum state can be reconstructed to arbitrary precision by repeated preparation followed by measurement in a complete set of bases. In the simplest example for *quantum state tomography*, a density operator ρ is obtained by linear inversion from the measured probabilities

5. That are, of course, limited to the linear quantum operation model, which can notably not describe strong non-Markovian noise.

$p_k \approx \mathrm{tr}(M_k \rho)$.[6] State tomography methods can be adapted for *quantum process tomography* of an operation $\mathcal{E}$ in a straightforward manner by analysing the resulting states $\mathcal{E}(\rho_i)$ for a complete linearly independent set of input states ρ_i. [22]

Reliable and accurate tomography is surprisingly difficult to realise experimentally, however. While full state tomography of up to eight qubits has been demonstrated (e.g. in [23]), the measurements in the different required bases were assumed to be ideal in most early experiments. Similarly, it is assumed that the different input states can be prepared perfectly in standard program tomography.

To ignore state preparation and measurement errors is not a good approximation in most experimental systems, however. Usually, there is only one preferred initial state and one measurement basis, and other states and bases are realised by applying the very same rotation that are to be characterised in the case of single-qubit process tomography. This self-referential approach has been shown to severely limit the quality of tomography estimates, particularly for high-fidelity operations. [24]

Gate set tomography (GST) is a relatively recent technique that addresses these systematic issues. It was initially developed and demonstrated in trapped ions at Sandia National Laboratories [25] and has been used to characterise a variety of systems since. [26, 27]

Let, for clarity of notation, $|\rho\rangle\!\rangle$ denote a density operator ρ interpreted as a column vector like in (2.9), $\langle\!\langle E|$ the same for a POVM element E (so that the associated Born rule probability reads $\langle\!\langle E|\rho\rangle\!\rangle = \mathrm{tr}(E\rho)$) and let CPTP maps be represented by their superoperator matrices G in the same basis. GST estimates the parameters of a *gate set*

$$\mathbf{G} = \left(|\rho\rangle\!\rangle, \langle\!\langle E|, \{G_1, \ldots, G_K\}\right) \tag{2.25}$$

on some Hilbert space $\mathcal{H}$ consisting of an initial state $|\rho\rangle\!\rangle$, a POVM measurement $\{\langle\!\langle E|, \langle\!\langle \mathbb{1} - E|\}$ and a number (K) of gates $\{G_k\}_k$, from a series of experiments of the form $\langle\!\langle E| G_{k_1} \ldots G_{k_m} |\rho\rangle\!\rangle$.

In conventional process tomography, the parameters for a gate G are estimated from measured probabilities $p_{k|l} \approx \langle\!\langle E| G_k\, G\, G_l |\rho\rangle\!\rangle$, where the various $\langle\!\langle E\, G_k|$ and $|G_l\, \rho\rangle\!\rangle$ are assumed to be known. A seemingly straightforward way to avoid the latter would be to use maximum-likelihood estimation (MLE) while allowing all the parameters for $\langle\!\langle E|$, G_k and $|\rho\rangle\!\rangle$ to float. To match a model gate set to a number of gate sequences $S_i = G_{s_{iL_i}} \ldots G_{s_{i1}}$ of some length L_i for which a given outcome was observed in n_{S_i} of N_{S_i} experiments, the likelihood function

$$l(\mathbf{G}) = \prod_i p_{S_i}(\mathbf{G})^{n_{S_i}} \left(1 - p_{S_i}(\mathbf{G})\right)^{N_{S_i} - n_{S_i}} \tag{2.26}$$

would be maximised, where $p_S(\mathbf{G})$ denotes the outcome for S predicted by a given candidate gate set $\mathbf{G}$, i.e.

$$p_S(\mathbf{G}) = \langle\!\langle E| G_{s_L} \ldots G_{s_1} |\rho\rangle\!\rangle\,. \tag{2.27}$$

6. In this and the following, "≈" expresses the fact that the probability estimates are obtained from a finite number of repeated measurements.

However, the level sets of the likelihood function are highly non-convex even for short single-qubit gate sequences, which makes the numerical optimisation very computationally intensive. [24]

One key aspect of gate set tomography is that it is able to provide a closed-form estimate for $\mathbf{G}$ with only a minimal set of prior assumptions (namely, that $d = \dim \mathcal{H}$ is known, and that the result of state preparation is indeed described by a density operator, the measurement by a POVM, and the gates by CPTP maps). This estimate can in turn be used as a starting point for the full MLE optimisation problem (2.26).

2.3.1 *Linear GST*

To perform gate set tomography on a given gate set $\mathbf{G}$, first $n = d^2$ short fiducial sequences $F_1, \ldots, F_n \subset \{G_k\}_k$ are chosen. Their purpose is to expand $|\rho\rangle\!\rangle$ and $\langle\!\langle E|$ into a full set of initial states and measurements that spans all of $\mathcal{S}(\mathcal{H})$. Consequently, the set should be chosen in a linearly independent fashion, which is easy if $\mathbf{G}$ is known to approximate an ideal target gate set, but can also be achieved without any a priori knowledge by trial and error.

Then, $(n^2+1)(K+1)$ experiments (with a certain number of repetitions each) are performed to obtain an estimate for the matrices

$$\begin{aligned} (\widetilde{G}_k)_{ij} &\approx \langle\!\langle E| F_i G_k F_j |\rho\rangle\!\rangle , \\ \tilde{g}_{ij} &\approx \langle\!\langle E| F_i F_j |\rho\rangle\!\rangle , \text{ and} \\ \tilde{r}_i &\approx \langle\!\langle E| F_i |\rho\rangle\!\rangle . \end{aligned} \tag{2.28}$$

As shown in [28] and [29], an approximation for $\mathbf{G}$, the linear GST estimate

$$\hat{\mathbf{G}} = \left(|\hat{\rho}\rangle\!\rangle , \langle\!\langle \hat{E}| , \{\hat{G}_1, \ldots, \hat{G}_K\}\right),$$

can be obtained from the results as follows:

$$\begin{aligned} \hat{G}_k &= \tilde{g}^{-1}\, \widetilde{G}_k \\ |\hat{\rho}\rangle\!\rangle &= \tilde{g}^{-1} \sum_i \tilde{r}_i \; |i\rangle\!\rangle \\ \langle\!\langle \hat{E}| &= \sum_i \langle\!\langle i| \; \tilde{r}_i . \end{aligned} \tag{2.29}$$

Note, however, that $\hat{\mathbf{G}}$ is not necessarily close to the ideal target gate set $\mathbf{G}$.[7] This is due to a residual "gauge" degree of freedom in the definition of a given gate set: Let $T \in \mathrm{GL}(d^2)$ and set

$$\mathbf{G}' = \left(|\rho'\rangle\!\rangle , \langle\!\langle E'| , \{G'_k\}_k\right) = \left(T\,|\rho\rangle\!\rangle , \langle\!\langle E|\, T^{-1}, \{T G_k T^{-1}\}_k\right).$$

$\mathbf{G}'$ is physically equivalent to $\mathbf{G}$, as the only experimentally accessible quantities are the measurement outcomes for a particular gate sequence S, but

$$\begin{aligned} p_S(\mathbf{G}') &= \langle\!\langle E'| G'_{s_L} \ldots G'_{s_1} |\rho\rangle\!\rangle \\ &= \langle\!\langle E|\, T^{-1} T G_{s_L} T^{-1} \ldots T G_{s_1} T^{-1} T\, |\rho\rangle\!\rangle \\ &= \langle\!\langle E|\, G_{s_L} \ldots G_{s_1} |\rho\rangle\!\rangle = p_S(\mathbf{G}). \end{aligned} \tag{2.30}$$

7. This can be seen immediately by, for example, directly substituting (2.28) into (2.29).

In particular, since T is an arbitrary invertible matrix, these gauge transformations do not even conserve positivity of the gate set elements. To obtain a meaningful gate set, $\hat{\mathbf{G}}$ thus has to be subjected to a gauge optimisation process first (choosing T numerically to minimise for instance the amount of violation of complete positivity, or the distance to a given target gate set).

2.3.2 *Extended GST*

As mentioned earlier, the linear GST estimate (2.29) is mainly useful as a starting point for other techniques such as full maximum-likelihood estimate. In particular, it is not necessarily physical (i.e. might violate positivity) due to the error inherent to the probability estimates (2.28), or in the presence of non-Markovian noise.

However, the simple maximum-likelihood approach is not the limit of gate-set tomography. Several extensions have been developed, notably *extended (linear) GST (e(L)GST)*, where repeated strings of short "germ" sequences are used between the fiducial sequences instead of single gates. This allows the technique to reach a level of precision scaling with $1/L$ (where L is still the overall sequence length), greatly improving on the $1/\sqrt{N}$ shot noise scaling of traditional process tomography. eGST and similar techniques are discussed in [28]. The improved precision also makes it easier to quote confidence intervals for the obtained tomographic results. Providing reliable error bars is difficult in standard maximum-likelihood tomography, as enforcing positivity makes it a biased estimator and thus renders resampling techniques unsuitable. [30, 31]

An Open Source implementation of basic linear gate set tomography as well as some of these advanced techniques in the Python programming language is available in the *pyGSTi* package [32], which at the time of writing still underwent active development.

2.4 $^{9}Be^{+}$ and $^{40}Ca^{+}$ in the TIQI segmented trap

In trapped-ion quantum information experiments, qubit systems are implemented using the electronic energy levels of atomic ions. The atoms are confined spatially using a combination of static (DC) and time-dependent (radio-frequency, RF) electrical potentials. The coupling of the internal states of the atoms as well as their motion to resonant laser and microwave fields is well understood and can be used to perform the required incoherent operations like cooling and state preparation as well as coherent operations on the qubit states, while the Coloumb interaction between multiple ions trapped in the same potential provides a natural foundation for multi-qubit operations. [33]

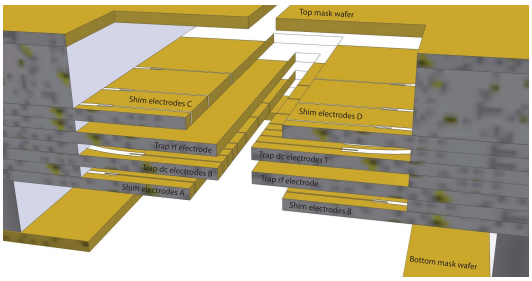

Figure 2.3: Transversal cut through the center of the stack of wafers making up the segmented-electrode trap built by Daniel Kienzler. (figure from [34])

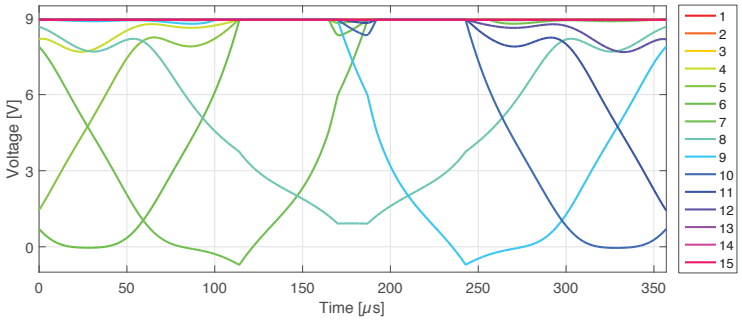

Figure 2.4: Time varying trap electrode voltages for transporting an ion from −500 μm to 500 μm over the course of 350 μs. Opposing pairs are assigned the same potential and numbered along the trap axis. (figure from [35])

The Trapped Ion Quantum Information group operates several ion trap setups. The "segmented trap" used as part of this project is a linear Paul trap, equipped to confine and manipulate Beryllium-9 and Calcium-40 ions. It consist of a stack of gold-plated, laser-machined aluminium wafers, as shown in figure 2.3. Notably, the electrodes that form the RF ground for the rotating quadrupole potential have been subdivided into 15 segments. Each segment is shunted to ground via a capacitor, forming a low-impedance path for the trap RF drive, while their DC potential can be controlled individually between ±9 V using a custom digital waveform generator system (see chapter 3.2). This makes it possible to shape the axial confining potential over time in a controlled fashion, for example to split a chain of several ions into multiple harmonic potentials, or to transport ions along the trap axis. Figure 2.4 shows an example for the voltages applied to adiabatically transport an ion from −500 μm to 500 μm, keeping it approximately at the minimum of a harmonic potential at all times.

A diagram of the experimentally relevant energy levels in $^{40}Ca^{+}$ is shown in figure 2.6. After a two-stage photoionisation process, the short-lived dipole transitions between the $4^2S_{1/2}$ and $4^2P_{1/2}$ levels at 397 nm are used for Doppler cooling, followed by electromagnetically-induced-transparency (EIT) cooling of the motional modes near their ground state using the $\left|4^2S_{1/2}, m = \pm\frac{1}{2}\right\rangle$ and $\left|4^2P_{1/2}, m = -\frac{1}{2}\right\rangle$ sublevels. Following the cooling process at the beginning of each experimental sequence, the population is pumped into the $|0\rangle = \left|4^2S_{1/2}, m = +\frac{1}{2}\right\rangle$ state, which forms the optical qubit used in this project together with the $|1\rangle = \left|3^2D_{5/2}, m = +\frac{3}{2}\right\rangle$ state. The states are connected by a 729 nm quadrupole transition, which is driven using a sub-kHz linewidth laser system consisting of a external cavity diode laser locked to a high-finesse cavity (the transmission of which is used to inject a two-stage diode tapered amplifier [36]). The qubit is read out using state-dependent fluorescence at 397 nm, after which the qubit is reset using a 854 nm laser connecting the $|1\rangle$ state to the $4^2P_{3/2}$ manifold. A repump laser at 866 nm

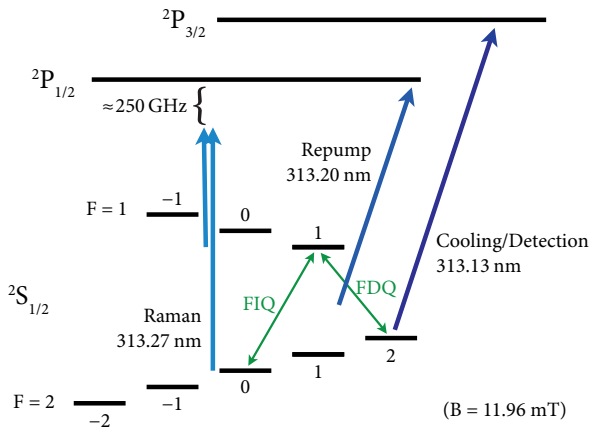

Figure 2.5: Level structure of $^9Be^+$ with the different lasers around 313 nm used in the experiment. The qubit states in the hyperfine ground state manifold are driven using Raman beam pairs detuned by ≈ 250 GHz from the $^2P_{1/2}$ manifold.

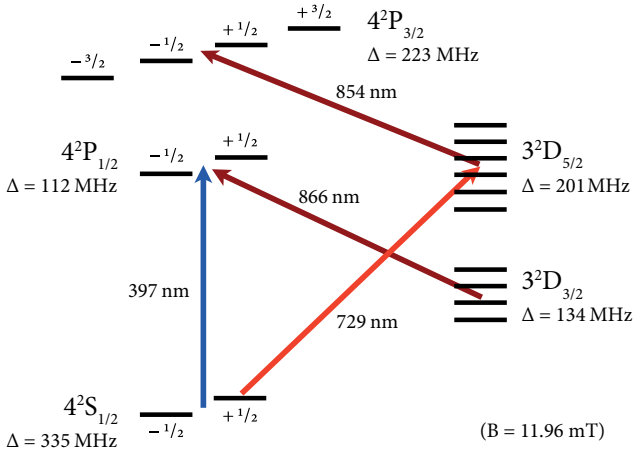

Figure 2.6: Experimentally relevant levels of $^{40}Ca^+$ and the wavelengths of the different laser beams used to couple them.

avoids trapping population in the dark $3^2D_{3/2}$ states during cooling and detection. The different operations, as well as the experimental setup for implementing them, are described in more detail in Daniel Kienzler's PhD thesis. [34]

The level scheme for $^9Be^+$ is shown in figure 2.5. Here, all the relevant operations are performed using UV lasers near 313 nm. For Doppler cooling, the $\left|2^2S_{1/2}, F = 2, m_F = 2\right\rangle \leftrightarrow \left|P_{3/2}, F = 3, m_F = 3\right\rangle$ cycling transition is is used, followed by state initialisation in $|0_{\mathrm{FDQ}}\rangle = \left|2^2S_{1/2}, F = 2, m_F = 2\right\rangle$ by optical pumping via the $2^2P_{1/2}$ manifold. $|0_{\mathrm{FDQ}}\rangle$ and $|1_{\mathrm{FDQ}}\rangle = \left|2^2S_{1/2}, F = 1, m_F = 1\right\rangle$ make up the *field-dependent qubit (FDQ)*, which can be manipulated coherently using a Raman process stimulated by a pair of laser beams detuned ≈ 250 GHz to the red of the transitions to the $2^2P_{1/2}$ manifold and read out using state-dependent fluorescence on the transition also used for Doppler cooling. At B = 11.9 mT, the energy splitting of the optical qubit in $^{40}Ca^+$ depends on the magnetic field by ≈ 11.2 kHz/μT and the $^9Be^+$ FDQ slightly stronger than that, ≈ 17.6 kHz/μT. However, the *field-independent qubit (FIQ)* between $|0_{\mathrm{FIQ}}\rangle = |1_{\mathrm{FDQ}}\rangle = \left|2^2S_{1/2}, F = 1, m_F = 1\right\rangle$ and $|1_{\mathrm{FIQ}}\rangle = \left|2^2S_{1/2}, F = 2, m_F = 0\right\rangle$ is insensitive to it to first order, which is the motivation behind that particular choice of flux density.[8] For initialisation and readout of the FIQ, $|0_{\mathrm{FIQ}}\rangle$ is mapped to $|0_{\mathrm{FDQ}}\rangle$ by applying a FDQ π-pulse. The readout fidelity can be improved by shelving the population in $|1_{\mathrm{FIQ}}\rangle$ to the $\left|2^2S_{1/2}, F = 1, m_F = -1\right\rangle$ state. Theory and experimental setup for these operations are discussed further in Hsiang-Yu Lo's PhD thesis. [37]

For both ion species, all the laser beams required in the experimental sequences are controlled using acousto-optic modulators (AOMs) driven by a set of RF synthesisers, in turn managed by the main experi-

8. In many other trapped-ion experiments, the magnetic field serves only to lift the degeneracy between Zeeman sublevels and define a quantisation axis, for which much weaker fields suffice.

mental control system, which is (partly) described in the next chapter. The photons scattered by the ions are collected by a high-numerical-aperture imaging system and detected by photomultiplier tubes and (for diagnostic purposes) two CCD cameras.

The operations in both $^{9}Be^{+}$ and $^{40}Ca^{+}$ can be driven by transporting the ions through stationary laser beams instead of the traditional scheme of using pulsed beams. However, in the current experimental setup, high-fidelity coherent operations are only within reach for the $^{9}Be^{+}$ FIQ. For $^{40}Ca^{+}$, the 729 nm qubit manipulation beam is aligned at an angle of approximately $\pi/2$ to the transport axis, making the resulting dynamics extremely sensitive to small velocity deviations (an effect further discussed and utilised in [9]). Furthermore, both the $^{40}Ca^{+}$ qubit and the $^{9}Be^{+}$ FDQ are sensitive to variations of the magnetic field along the transport path, which have been estimated to cause qubit frequency shifts of several kHz along distances typical for implementing transport gates. While this is purely a technical and not a fundamental physical restriction, it makes the $^{9}Be^{+}$ FIQ a much better candidate for demonstrating high-fidelity transport gates.

TECHNICAL IMPROVEMENTS 3

As the scope and complexity of schemes accessible in ion trap experiments grows, so does their demand on the techniques used in their implementation. To briefly put this statement into perspective and motivate the work described in this chapter, let us briefly consider two facets of the experimental setup used in and developed by our group:

Firstly, due to the high susceptibility of quantum-mechanical effects to noise, it is necessary to actively stabilise many parameters of the environment and control devices. The demand for ever higher levels of stability continues to grow especially since individual realisations of most important quantum operations have already been demonstrated in trapped ions with remarkable fidelity. [5] To this end, upwards of twenty closed-loop controllers need to be active and stable if both $^{9}Be^{+}$ and $^{40}Ca^{+}$ ions are to be used in one of our experiments. Without adequate tools, even just tracking down degradation and faults quickly as they occur becomes a challenge.

Secondly, we are always interested in realising the quantum operations of interest as quickly as possible, both to combat decoherence as well as to decrease the duration of experiments[1]. There are direct consequences of those timing granularity and stability requirements for the necessary electronics on many levels, but of particular importance for this work is the generation of trap electrode voltages for ion transport. There are currently 16 independent channels for this, the voltage levels for each of which are specified at a resolution of 16 bit and an update rate of 100 MHz.[2] Thus, the total rate of data flowing through the DAC system is 25.6 Gbit s^{-1}. While not exceedingly high by the standards of today's global communication, it is certainly more than a single desktop computer can stream over a commodity network link. Among other factors, such as stringent latency and timing stability requirements, this is one fundamental reason why custom, "intelligent" hardware projects are routinely pursued in the group.

As the experiments performed as part of and/or concurrently to this work pushed the boundary of what has been done in our group in terms of sequence complexity, a considerable amount of effort was spent on improving the technical infrastructure. Along with various general improvements to the control system soft- and hardware, the two main efforts concern the digital PID controller platform developed in our group and the aforementioned DAC system.

1. And in the long run, to increase the performance of a quantum information processor – the constant factors involved will have a direct impact on the feasibility and performance of quantum error correction.

2. For unrelated reasons, the channel data is currently only clocked out at 50 MHz, using the built-in 8× interpolation offered by the DAC chip to oversample it. This restriction is only applied very late in the FPGA pipeline, though, and could easily be lifted.

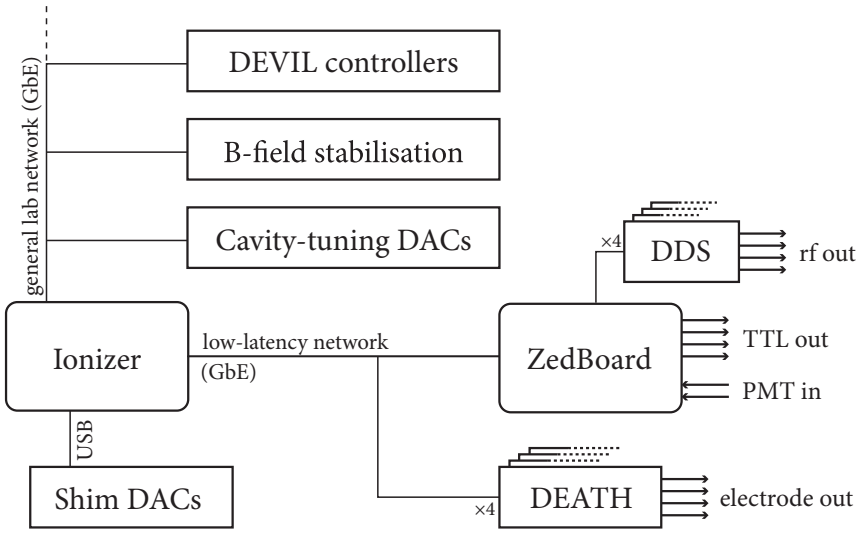

Figure 3.1: Schematic overview of the experimental control system used for the segmented trap experiment.

3.1 MAIN REAL-TIME CONTROL SYSTEM

The system used for real-time control of the experiments in the Trapped Ion Quantum information group is centred around a combined ARM CPU and FPGA platform, currently the *Xilinx Zynq-7010* chip on an *Avnet ZedBoard*. The main experimental sequencing core runs in its FPGA fabric, which controls all the logic-level signals (many of which toggle RF switches to drive AOMs) and photo-multiplier acquisition windows. This core also triggers a number of direct-digital synthesis (DDS) RF generation cards that are programmed by the main CPU over a custom backplane. They provide phase-coherent RF signals, which are mostly used to control AOMs for phase-sensitive qubit operations or where frequency agility is required. The ZedBoard also programs and triggers custom DAC hardware for generating trap electrode voltages, as will be discussed in section 3.2.

To define a particular experimental sequence, the experimenter writes a piece of C++ code to run on the ARM CPU as part of a standalone program that is altogether referred to as *IonPulse*. It uses standard programming interfaces to control real-time peripherals such as the DDS cards and the electrode waveform generators, handles any custom result processing or in-sequence branching, and also contains a list of parameters for the user to modify. All these experiments with their associated controls are aggregated by a custom PC software, *Ionizer*. It runs on a desktop computer that is connected to the ZedBoard via Ethernet, and provides a graphical user interface to control the various parameters and additional, "slow" peripherals like auxiliary control electrodes and piezoelectric elements for tuning reference cavities. It also allows the user to launch various experiments, schedules them for concurrent execution

on the hardware, and displays and archives their results.

Over the course of this project, a number of technical improvements were made to Ionizer, IonPulse as well as their network communication layer. The version control system shows more than 600 individual commits and several tens of thousands of modified lines, so the changes will not be described in much detail here. Instead, I will discuss a few major design decisions in the following subsections in the hope of informing future control system work.

3.1.1 Unified Network Communication Protocol

When initially considering how to integrate the DEATH custom DAC hardware with the other network-enabled components of the control system (see section 3.2), it became apparent that the choice of network protocols at that time was a major impediment for future expansion of the system, both in terms of new hardware and in terms of additional functionality.

The communication between Ionizer and IonPulse was based on messages with a fixed size of 888 bytes, consisting of a header made up of several flags and a completely opaque collection of bytes in the body. There were several problems with this scheme: First, it was inflexible – from a user's point of view, the only supported payloads were either a single 32 bit integer, a pair of them, or an arbitrary string of bytes –, making it hard to encode more complex data structures. For the latter reason, it was also error-prone, since client code was expected to directly read and write binary data from/to arbitrary memory offsets to express such structures as a byte string. Furthermore, the fixed size meant that the protocol was highly inefficient in that small requests would be padded into huge messages, while at the same time limiting the maximum amount of data being transmitted at once, as each message was expected to arrive as a single Ethernet frame[3].

Other hardware used various ad-hoc protocols, some of them textual, and some using a similarly rigid binary encoding. This made implementing defect-free network interfaces hard, as evidenced by the fact that new hardware was often not properly integrated with the system at all, or using serial/USB interfaces instead[4]. All these protocols were not particularly well-suited for the event-driven programming model offered by *lwIP*, the IP stack used on those applications running directly on hardware without an operating system layer, as done for IonPulse on the main ZedBoard.

Thus, a network protocol was carefully chosen for the DEATH project with the idea that it would also be used as the future default choice when developing other pieces of hardware for the laboratory control system. Perhaps the most important goal was to come up with a solution that would be easy to use from many programming environments, in particular C++ and Python. It also had to support rich data types (such as structures of arrays of numbers, lists of strings, etc.) and error conditions

3. With TCP being a stream-based protocol, no such guarantees can be made in the first place. Below a certain message size, there just happens to be little chance for packet fragmentation on a local network.

4. At the time, the fact that such consumer-grade serial port links and USB connections suffer from issues with issues with signal integrity over longer distances and differences in ground potential was growing into a major problem for the reliability of the experiment, motivating a shift towards Ethernet wherever possible.

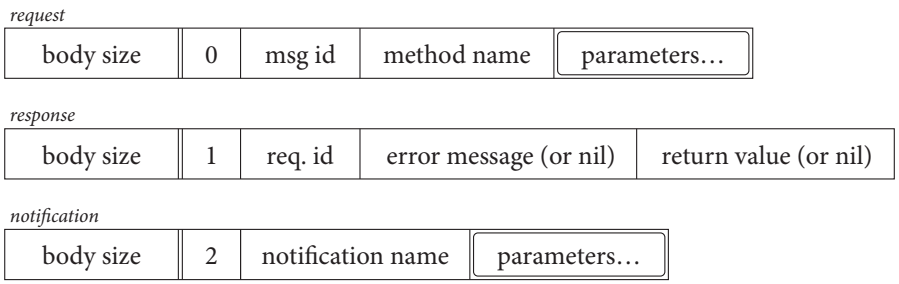

Figure 3.2: Schematic representation of the TCP-based *msgpack-rpc* networking protocol. An initial header containing the message body size as a four byte little-endian integer is followed by a variable-sized MessagePack array. There are three message-types, each signified by a fixed number, followed by the respective contents. Requests and responses are always exchanged in matching pairs, and usually only initated by one of the parties (the client). Notifications require no response and can be sent by either party.

in a straightforward manner, while offering enough efficiency so that all network communication (including high-frequency experimental result data) could be handled over this one protocol. As possible target platforms, modern 32/64 bit CPUs (or at least reasonably powerful soft-cores) were assumed, with the caveat that the implementations would have to work well in purely event-driven environments with small buffers sizes, as would be the case when not running an operating system.

Libraries developed for high-performance remote-procedure calls (RPC) in, for example, the distributed backing infrastructure of modern web applications are a good match for these requirements. The RPC programming model, in which network communication is conceptually equivalent with invoking a function with a set of parameters and return values on a remote peer, is easy to grasp for non-experts, and when extended with "one-way" messages to allow occasional communication outside the strict request/response dialog is certainly powerful enough for our purposes.[5]

5. An example for a more involved model could be that of a generalized message passing system with central queues, brokers, broadcast functionality, etc.

Some of these libraries, like Google *Protocol Buffers* or Apache/-Facebook *Thrift*, rely on code generation to provide the user with an interface to a certain data structure or network service that has been previously described in an extra file using a special interface definition language. This might be nice to start quickly, especially as e.g. Thrift can automatically generate client and server code for a given interface, and would also guarantee a certain level of robustness, as error checking would automatically be inserted where there is potential for mismatching data formats, missing parameters and so on. However, having to use an external tool to generate source code would complicate the build process for new users. Additionally, some of the features such as the automated generation of server code would only be of limited use in our situation, as they are mostly geared towards high-throughput server applications running on an operating system (in contrast to constrained environments like our "bare-metal" ARM devices).

Thus, a solution was chosen that does not rely on code generation. In particular, the new network protocol is based on *MessagePack* as the serialization layer. MessagePack, or *msgpack* in short, is a serialisation format that comes with libraries for many programming languages, among them also C++ and Python. In contrast to Protocol Buffers and Thrift, its C++ implementation makes extensive use of modern C++11/C++14 features to provide a convenient high-level interface for (de)serialising user-defined data structures and offers built-in support for many C++ standard library types. It has been shown to compile on the embedded ARM platform without any major adaptions, and can be tweaked in its handling of memory allocations and object lifetimes where necessary for performance.

MessagePack is only a data serialization format, so additional specification on top of it is needed to use it as an RPC protocol. For this, the *msgpack-rpc* protocol has been chosen. As shown in figure 3.2, its messages consist of a simple msgpack array on the top level that identifies the message type and, depending on the that type, also carries extra metadata such as the name of the method to invoke or a slot for an error message. To make the protocol easier to handle in event-driven environments such as an lwIP-based server implementation, the messages are prefixed with an explicit packet length field. The latter design decision is somewhat debatable, as it makes existing msgpack-rpc libraries less straightforward to use in the laboratory network. But to cater to the resource constraints and simplicity goals, it was necessary to implement the protocol several times for different environments anyway.[6] Fortunately, this is made easy by the MessagePack libraries; for example, a complete implementation of a generic msgpack-rpc client requires only about 50 lines of Python code.

The protocol ended up being successfully deployed in a number of networking nodes in the laboratory. It is of course used in the DEATH DAC system (see section 3.2) and the new control system scripting implementation (see next subsection). Together with Matteo Marinelli I also converted the main Ionizer/IonPulse interface to it, which led to improvements in network performance by allowing us to easily coalesce a number of smaller commands into bigger messages (e.g. when updating a list of parameters), and made it simpler to implement propagation of run-time errors in the experiments back to Ionizer[7]. The DEVIL platform, discussed in section 3.3, also makes extensive use of the protocol (albeit wrapped into ZeroMQ messages), and so do two other control system peripherials.

For ease of (re)use in new projects, various implementations have been collected in a single *tiqi-rpc* repository. Among the currently supported platforms are C++/Qt, C++/Boost.Asio (including a version for ZeroMQ), Python/PyQt, Python/ZeroMQ and Python/raw sockets. Each of the implementations comes with documentation and short source code examples. This way, it should be easy to add networking capabilities to new hardware projects developed in the group, whether

6. For example, many of the existing msgpack-rpc implementations mandate the use of some kind of asynchronous networking framework.

7. The ability to send "out-of-band" update notifications from IonPulse back to Ionizer was also (re)implemented as part of this transition, although this is more an improvement on the lower lwIP networking layer than an advantage of the protocol.

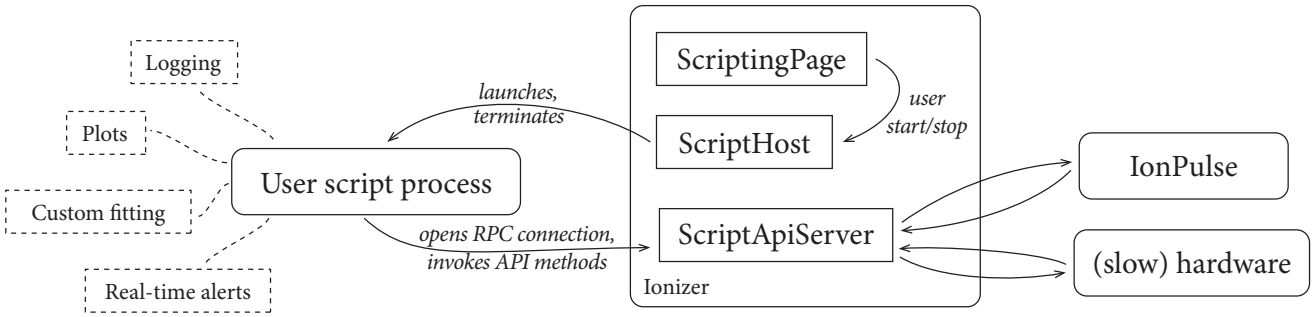

Figure 3.3: The Ionizer scripting model. User scripts are started as separate processes that communicate with Ionizer over a TPC socket. They can access all parameters set in Ionizer, but also display a custom user interface if needed.

they use an embedded CPU or a more fully-featured system like the Raspberry Pi.

3.1.2 Control System Scripting Support

In our current control system architecture, the user-specified experiments executed on the IonPulse CPU only ever acquire "a single data point" consisting of a number of repetitions of a given experiment. All higher-level scheduling decisions – for instance to execute a certain experiment a number of times with varying parameters, or to interleave several experiments for purposes of periodic recalibration – are made by Ionizer, which contains no experiment-specific code and is usually not modified by experimenters. This was problematic in so far as the only possible mode of data acquisition was to execute a scan in one or two parameters that would be iterated according to a predefined, linearly spaced grid. As the complexity of the experiments performed in the group rose, these restrictions resulted in a number of contortions where experiments with a higher-dimensional or irregularly shaped parameter space were forced to fit 1D- or 2D-scan model, making it often prohibitively involved to automate a given data acquisition or calibration process. The rigid calibration routines also started to show their limits where non-trivial calculations or adaptive calibration schemes would have been useful.

It was thus decided to add scripting support to Ionizer, with the goal of allowing user-supplied programs to control experiment execution to circumvent these restrictions (see figure 3.3). Given its popularity in the scientific computing domain and its mature ecosystem with tools for numerical calculations, fitting data and displaying plots, Python was an obvious choice for the main programming language to support for user scripts. In order not to let faulty user scripts compromise the system stability as well as to make it easy for scripts to present their own user interface in a blocking fashion, each script runs in its own process. The communication between Ionizer and the scripts is based on local TCP connections, following the unified networking protocol described in the previous section. The overhead incurred by this is negligible, and it

```
import ionizer

def main(host):
  while True:
    host.run_exp("detect Ca")

ionizer.run_script(main)
```

Figure 3.4: A minimal example for an Ionizer Python script, which in this case simply executes a certain experiment in a loop.

provides a generic interface that could be used from other programming languages and remote computers in the future, while also being able to reuse functionality developed for the main IonPulse/Ionizer connection.

The elementary operations available to scripts mirror the actions possible in the user interface: Setting certain parameters or reading their values, triggering custom actions defined remotely by the IonPulse experiment code, or running one- or two-dimensional scans. The server interface in Ionizer runs in a fully non-blocking fashion on the main user interface thread, so that there is no possibility for conflicts between scripted and manual user interaction on a programmatic level[8]. Ionizer launches each script process with a command line argument describing the TCP/IP address/port of its scripting server interface. As shown in figure 3.4, a small library is provided for user scripts that reads this argument, establishes the connection to Ionizer, and exposes the functionality as simple method calls.

8. Although, of course, it would still be confusing for the user if they tried to manually change a parameter as it was being updated by a script.

As an extension to what is available via the user interface, scripts can also directly launch an experiment on the hardware, bypassing all the usual parameter/scan setup and result archival. This can be used to implement entirely custom sequences of experiments, for example for adaptive calibration search routines, or for acquiring tomography data as discussed in chapter 2.3. The throughput of this "raw" API, which is limited by its end-to-end latency (a script invoking the appropriate RPC call, Ionizer forwarding the command to IonPulse, receiving the results and retuning them back to the script), reaches on the order of thousand experiments per second on the particular configuration used in our laboratory. Unless only very few repetitions of a particular experimental sequence are being run, this indirect interface is thus unlikely to become a bottleneck. In situations where this is the case, it would likely be beneficial to implement the algorithm in question directly on the ZedBoard so that the latency over the Ethernet connection is entirely avoided, since the latter will typically dominate any processes local to the PC.

At the time of writing, the scripting system is running reliably and has for example been used to acquire the data presented in [11]. There are still a number of clear-cut opportunities for improvement, though. From a very practical point of view, it is desirable to have commonly

used auxiliary functionality beyond the low-level Ionizer programming interface discussed here shared between scripts as well. For example, other group members have developed helper functions for setting up scan parameters in certain ways and reading back the results, etc. One particular aspect that is still missing at this point is a common abstraction for creating log files: While Ionizer saves the usual result archives also when scans are triggered from scripts (to avoid losing large quantities of data due to mistakes in scripts, and to help with reconstructing the course of action after the fact), it is often desirable to save extra information about fits, decisions made, and so on. Another related aspect, particularly concerning long-running experiments, is that of sending real-time notifications to the experimenter. During the course of this project, I implemented a small script that can be used to send out alerts (e.g. when the script detects a laser went out of lock, etc.) via email, the *Slack* chat platform, and SMS. Integrating this more tightly with the logging system would likely be useful.

There currently is another big limitation regarding the flexibility of scripting, which in some sense is a fundamental property of the current control system architecture: Each experiment (as exposed by the ZedBoard) can only be submitted to the Ionizer time-slice scheduler once; there cannot be several instances of a given experiment running in parallel. This is a major hassle for automatic recalibration of experimental parameters – for example, if a certain experiment page is used as part of a procedure for periodically calibrating a pulse time, this experiment cannot be as part of the actual data collection. In other words, it is not possible to treat a script as a building block that realizes a certain experimental procedure and that can then be seamlessly composed with other such building blocks. All those scripts implicitly share a large amount of global state, of which the run/stop states and parameters of the experiments are one part.

The most attractive solution to this seems to be to rework the experiment scheduling system such that instead of just sending a launch command to *the* experimental page, an *instance* or copy of the experiment is created and submitted to the scheduler, with its own immutable set of parameters and scan settings.[9] In an elementary programming context, this would be analogous to changing a `runexp()` function that reads parameters from global values to a function `runexp(params)` that takes all this state as an explicit argument. Of course, the hardware might still only have a fixed global list of pulse sequences with certain parameters each for performance reasons. But multiplexing several "logical" experiments onto that would automatically be handled by the higher layers of the control system (e.g. Ionizer).

Another (quite possibly easier) option to work around this problem for scripts only would be to implement a scheduler that grants each script exclusive access to all the resources in Ionizer. If a long-running experimental script is then interrupted by a periodic calibration script, the former would need to explicitly yield control by stopping all the

9. Of course, this is never quite true in a control system for a physical experiment; at least as long as there parameters not managed by it. For example, if one experiment loses the ion from the trap, other experiments will also be affected unless the control system knows to detect this and re-load the trap.

running experiments and resetting parameters as appropriate before the latter could start to execute. This manual approach seems to be rather brittle and limiting in the long run, though, and might again make composing more complex experiment structures out of smaller units hard. Similar to how virtually all computer operating systems implement pre-emptive context switching to hide from user code the fact that many processes are using the same CPU, it seems like it should be the role of a quantum optics operating system to hide the fact that certain pulse sequence templates and so on exist only once in the FPGA hardware from the experimenter.

3.2 DIRECT ETHERNET-ADJUSTABLE TRANSPORT HARDWARE (DEATH)

As discussed in chapter 2.4, in order to transport ions along the trap axis fast control over the electrical potential at a number of electrodes is required. Currently, they are arranged into 16 pairs which are each supplied with a voltage between ±9 V. In order to avoid heating the motional degrees of freedom of the trapped ions there are rather stringent requirements on the noise performance of the voltage sources. In addition to that, realising more complex experiments (such as combinations of transport gates and their evaluation) requires flexible sequencing and branching capabilities with low and deterministic latency. In our group, Ludwig de Clercq has developed a custom hardware solution to meet these requirements, which is discussed in more detail in [8]. Named *DEATH* (short for "Direct Ethernet-Adjustable Transport Hardware"), the project consists of a custom backplane-mounted PCB that integrates four analogue output channels with a commercial Zynq-based CPU/FPGA daughter-board, the *Avnet MicroZed* (a similar, but smaller, version of the ZedBoard used for the main control system).

For initial hardware testing and the very first transport experiments, a basic set of firmware and software tools had been developed by Ludwig de Clercq; it is described in some detail in the aforementioned thesis. Crucially, though, no support for having more than one sequence available at a given time and branching between them was present. Furthermore, the system was not at all integrated with the rest of the experimental control system. Waveforms and sequences were defined using a separate graphical user interface and the user code on IonPulse would only output simple binary TTL trigger signals for them. Since setting up the sequences had to be done manually every time, this made switching between different experiments involving transport tedious and error-prone. The lack of integration with the control system also made it unrealistic to realise experiments where the sequence would change on the fly (such as with randomised benchmarking).

In order to hide internal details like memory locations from the

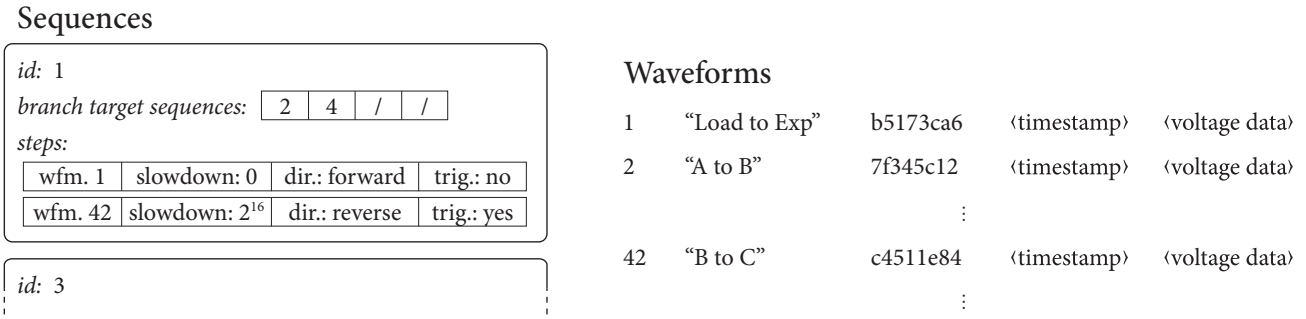

Figure 3.5: The DEATH sequencing model. The user uploads a number of waveforms, which consist of a list of samples (output voltages) and informational metadata. They can then be referenced from sequences that contain one or more waveform playback steps together with triggering and branching information.

network client and address several of the network implementation issues discussed in section 3.1.1 and some additional subtle bugs, a completely new firmware was developed both for the FPGA and the CPU part (with the exception of the DAC line interface driver).[10] Its design is discussed in more detail in the following two sections. Supporting measurements regarding the Ethernet connection latency are presented in section 3.2.3 and section 3.2.4 discusses the output stage calibration, which is now applied by the FPGA during waveform playback.

10. Note: This makes most of the discussion of RAM contents and data structures in [8] obsolete. Detailed information on the new implementation can be found in the firmware project documentation.

3.2.1 High-Level Design Considerations

The primary goal behind the new firmware project was to offer more flexible sequencing and triggering capabilities. To that end, the user can configure *sequence* and *waveform* objects, as shown in figure 3.5. Waveforms are sets of output voltage data for all the channels (electrodes) in the system. They can also have additional associated metadata, such as a human-readable description or a generation time stamp. This data is convenient for the user when manually triggering waveforms or developing new experiments, but otherwise ignored by the control system.

These waveforms can be referenced by an arbitrary number of sequences. A sequence is conceptually a unit of linear execution during which no branches occur. It consists of up to 2^{16} steps (limited by the available hardware memory), each of which references a certain waveform. Along with flags indicating whether the waveform should be played in reverse and/or that an external trigger signal should be awaited before starting playback, the user specifies a system clock divider ("slow-down factor") for each step. The latter allows quick coarse adjustments to the transport speed. At the end of each sequence, a set of other sequences can be specified (currently: 4) that act as fixed branching targets. Whereas branching to arbitrary sequences – for example, when switching to a different experiment, or entering a general recovery mode – might require network communication, the fixed targets can be selected by the main control system using dedicated hardware logic lines.

Note that branches between sequences are only followed at the end of

Figure 3.6: Screenshot of the manual waveform playback interface in Ionizer, which queries the DEATH hardware for metadata on the currently loaded waveforms and displays a summary of them in a list. Two buttons allow triggering any waveform manually (both in forward and reverse direction) with a configurable slow-down factor.

sequences. For instance, even if a branch override command is sent while a multi-step sequence is running, execution of the new sequence will only start after the current one has reached its end. This is a deliberate design choice: If network communication is required to trigger a branch that is not pre-programmed, it is advantageous to do so as soon as the branch target is known (to hide the associated delays). This requires the control system to be able to determine when the branch command will be honoured, even though the network latency is not deterministic. In the absence of another reference, such as a clock globally distributed within the laboratory, the TTL trigger lines constitute the only means of synchronisation between the DEATHs and the main control system, making sequence boundaries a natural choice with useful granularity. To verify that the hardware followed the intended control flow, a checksum of all executed sequences is computed on the FPGA, which can later be verified by the control system.[11]

In order to make it straightforward to implement the sequencing mechanism as a hardware state machine in the memory-constrained FPGA environment, both waveforms and sequences are assigned to a fixed hardware "slot" and referenced throughout the system by the respective index of that slot. Currently, 8-bit numbers are used for indexing waveforms as well as sequences. This limits the number of objects that can be concurrently active on the hardware to $2^8 = 256$ each. By simply increasing the bit width of the indices and the size of the corresponding address tables, more slots could easily be added. However, since the hardware implementation currently uses only FPGA-internal RAM (see next section), the size of the available sequence/waveform memory is going to be more of a restriction for most applications than the number of slots to index it.

Given such a system of numerically indexed slots, one important design decision is how to associate the actual waveform data with the experimental sequences that use it. One initial idea would have been to integrate the waveform data into the user code running on the ZedBoard and have the system automatically upload it to the hardware. This way, there would be no potential for mismatch; e.g. that another waveform is loaded into a given slot than the IonPulse code defining a given sequence expects. However, this scheme would be cumbersome for iteratively tweaking transport waveforms, as a new ZedBoard software would have to be compiled and uploaded for every small change. Thus, the deci-

11. After the timing-critical part (i.e. a sequence of coherent operations on the ion) has been finished, the control system can simply wait for some time such that the DEATHs are certainly idle and then fetch and compare the checksum in a blocking network request.

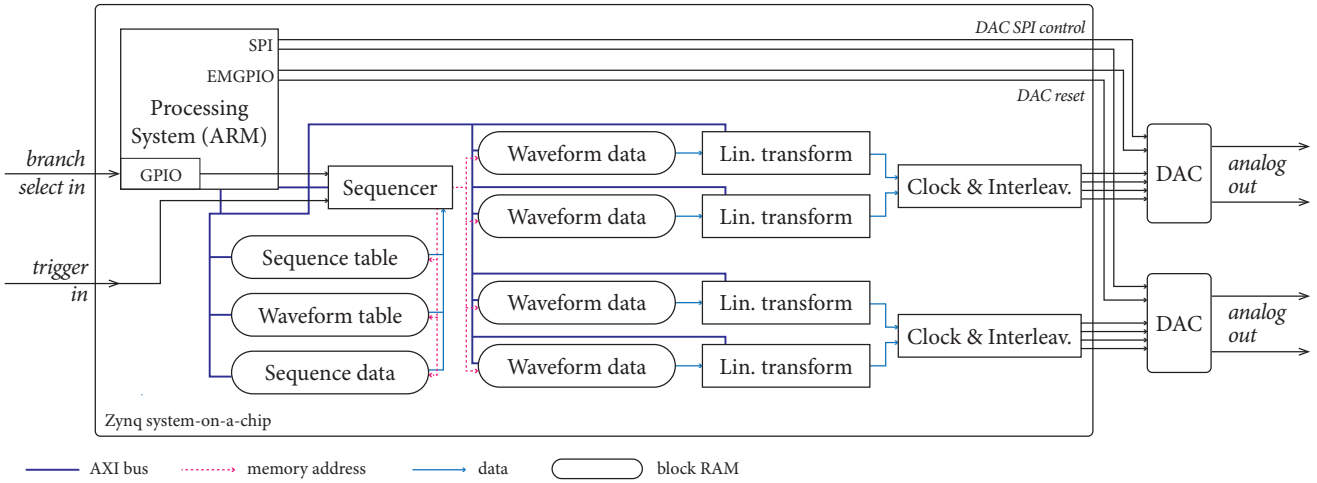

Figure 3.7: A high-level block diagram of the firmware for the DEATH Zynq ARM/FPGA chip. The sequencer core reads data from the sequence table/data and the waveform table memories to generate a stream of waveform data addresses. Four identical channel pipelines then derive the output values from this, which are then combined to drive the parallel buses to the two dual-channel DAC chips. AXI buses connect the FPGA cores to the CPU, which provides the network interface for management and uploading data.

sion was made to separate waveform management entirely from the experimental sequence code.

Instead, the waveform data is now uploaded manually by the user from within Ionizer. This it is convenient from a user's point of view, as it can be integrated with manual triggering functionality[12] (see figure 3.6) and information about the loaded waveforms can be archived as part of the usual experiment result data folders. To help avoiding situations where the set of loaded waveforms does not match the experimental code, a "unique id" field has been added to the waveform metadata. The id is set when the waveform is first generated (MATLAB, etc.), and IonPulse experiments can query it for each hardware slot to compare it to the expected value.

12. For hardware testing, sequences can also be triggered from software. However, a system with multiple DEATHs will be affected by network jitter this way. Thus, hardware trigger lines driven by the ZedBoard are exclusively used in the laboratory.

3.2.2 *Firmware and Software Design*

Figure 3.7 illustrates the architecture of the DEATH firmware, as loaded onto the Zynq chip on the MicroZed board. To provide deterministic timing, most functionality is implemented on the FPGA level. The central piece of the design is the custom sequencer core which is shared between all four channels.

The sequencer core derives its configuration from two sets of waveform and sequence memories, both of which are organised in a two-tier scheme. The 8-bit slot id that represents the waveform/sequence is used to index a *table* memory, which contains the address range in the corresponding *data* memory. When a given sequence slot is triggered by the user, the sequencer core first accesses the sequence table memory to obtain the corresponding start address in the sequence data mem-

ory. From the latter, it then reads a fixed-length header describing the branching information for the sequence, and a variable number of step entries. For each of the steps, it queries the waveform table memory for the waveform data address range corresponding to the desired waveform slot, and iterates over this range in the given direction and with the desired speed.

The output of the sequencer core is this continuous stream of addresses that represent a location in each of the four channel waveform data memories. For each of the channels, the 16-bit value stored at that address is converted into a DAC word by passing it through a stage which applies an affine transformation with configurable coefficients. Since the DEATH hardware uses two dual-channel DACs that are updated in an interleaved fashion, the respective pairs of output streams are finally merged into one and sent to the DAC chips along with the required clocks and channel selection lines over differential links.

Currently, all the memories involved are realised using the on-chip FPGA block RAM cells, which makes it trivial to implement the sequencer with low latency and deterministic timing. The waveform data memories are 4×32 kiB in size, which corresponds to 65 536 16-bit samples of output voltage data per channel. The memory storing the sequence definition data is also 32 kiB in size. As a sequence header is 8 byte in size and a single step occupies 4 byte (waveform slot, direction/trigger flags, slowdown factor), this corresponds to a total maximum sequence length of approximately 8100 transport events.[13]

13. In practice, this is unlikely to be a significant limitation, since a transport experiment of that length would be composed of several shorter subsequences. Using hardware branches, these could then be concatenated on the fly by the main control system to form the longer sequence.

The ARM core on the chip is mostly responsible for providing a high-level network interface for setting the hardware parameters. It manages the assignments of address blocks to waveform and sequence slots, and stores additional metadata about the current waveforms that is not required by the hardware in its associated DDR RAM. The configuration interface, which is based on the protocol discussed in section 3.1.1, also allows the user to configure the output transformation stage (see section 3.2.4) and read back the data currently sent to the DAC outputs. During startup, it also resets the DAC chips and configures them over an SPI interface.

Unfortunately, no additional unused FPGA pins are accessible on the first-generation DEATH circuit boards beyond the main trigger signal. Because of this, the "PMOD" header on the MicroZed board was the only viable way to add hardware branch selection lines to the existing system. These pins, however, are routed to a part of the fixed GPIO pins on the Zynq chip which are not directly accessible from the FPGA fabric. To circumvent this problem for the current board revision, an edge-triggered interrupt handler on the CPU mirrors the signals to the EMIO-accessible part of the GPIO controller. This of course introduces a hard-to-predict amount of delay and jitter to the signal, which is particularly undesirable for fast branching. This GPIO-forwarding latency has been measured to be approximately 80 ns for an otherwise unloaded CPU with relatively small jitter, but other hardware interrupts (e.g. triggered

by the network adapter) might introduce additional latency on a non-idle CPU. In preliminary tests, a delay of 1 μs between changing the branch selection lines and asserting the trigger signal has been used, and no timing issues have been observed. The functionality has yet to be used as part of a more complex experiment, though. In either case, dedicated FPGA inputs are available on the second hardware revision.

Arbitrary branches, beyond those pre-programmed and indexed by the hardware lines, can be triggered using a separate UDP-based network interface, the performance of which is discussed in more detail in the next section. To implement this branch override and the checksum read-back, the sequencer core is mapped into the ARM CPU address space over an AXI4 bus along with all the block RAMs and the output transformation stages. The CPU is the only AXI bus master, the sequencer core accesses the memories using a second port for cycle-accurate timing.

To encapsulate all the required hardware details and network communication, a C++ client library has been written. The full functionality is available both on top of the lwIP and Qt networking layer, allowing the same code to be used in IonPulse experimental code and the user interface implementation in Ionizer. However, the Ionizer client only modifies waveform data and reads status information, whereas all the sequencing manipulations are performed on the ZedBoard. Keeping all the sequence manipulation in one place is actually required in the current design, as the list of slots is managed by the client. The system configuration (number of boards, IP addresses) is set within IonPulse, and exported as a set of parameters to Ionizer. This proved especially useful for switching between different configurations during testing.

3.2.3 Network-Based Branching

As described in the above, the newly developed DEATH firmware supports pre-configured branches between different sequences. If the branches were to be selected over a standard serial link, its bandwidth would have to be relatively high to achieve low latency. For example, transmitting an 8 bit sequence id in 1 μs requires a raw transmission rate of at least 8 Mbit s^{-1}, even if extra latency incurred by buffering is ignored. Considering this and ease of integration with the main control system (which currently has no provisions for integrating high-speed data links), the current system simply uses two logic lines to select between up to four pre-programmed "fast" branches.

The Gigabit Ethernet connection between IonPulse and DEATHs is used for all other branches, including that to initially start a given sequence. As the requirements are quite different from the main configuration interface (since timing is critical, the reliability features of TCP are counter-productive), a simple UDP-based protocol is used. Its two-byte messages consist of a byte of flags and a byte with the id of the target sequence, and are designed to be broadcast to all DEATH boards on the local subnet[14].

14. To support multiple independent DEATH installations on the same network, the range of ports to use can be individually configured.

Scenario	Latency
Direct connection	15.2(2) μs
Level One *GSW-0807*	17.8(2) μs
Netgear *GS108*	33.6(2) μs
Raw NIC receive	8.8(2) μs

Figure 3.8: Mean latencies for 1-byte UDP packets sent from a ZedBoard to a MicroZed over a Gigabit Ethernet connection.

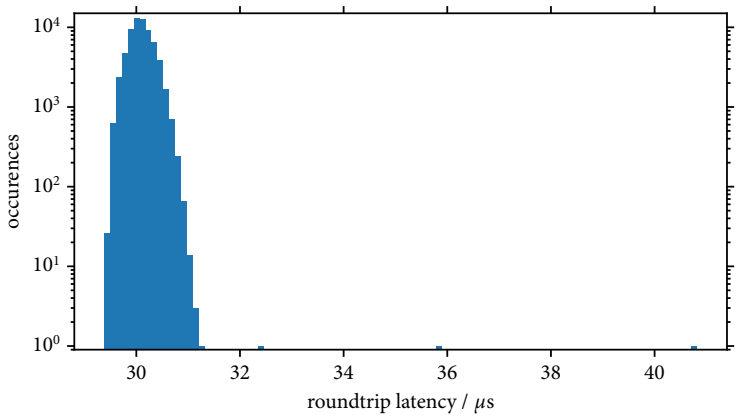

Figure 3.9: Distribution of round-trip latencies over the Gigabit Ethernet connection between two MicroZed and ZedBoards across a Level One GWS-0807 switch, measured for a 2-byte UDP broadcast packet (65 536 iterations total). Number of trials in each bin is shown on a log-scale.

To evaluate the timing properties, two sets of experiments were performed on one ZedBoard and one MicroZed. In the first, one program would assert a digital output and immediately send a 1-byte UDP packet, whereas the other would receive the packet and then toggle its output in turn. The difference in time between those two events was measured with an oscilloscope to give an estimate for the latency of the networked trigger method. The results for several different configurations are shown in table 3.8. Note that the performance heavily depends on the network configuration: While the *Level One GSW-0807* switch used in one test adds only 2.6 μs of latency to the connection, a *Netgear GS108* switch produced a significant extra delay of 18.4 μs. To estimate the overhead due to the lwIP Ethernet/IP stack, the low-level Ethernet driver code that reads the received packet from the network interface controller (NIC) ring buffer was instrumented in a similar fashion. The extra latency from this point to the receive callback function in user code is approximately 6.4 μs, which suggests that there is significant potential for optimisation in the network stack for applications that require even lower latencies.

Both of the above switches are consumer-level commodity products and no latency properties are indicated in their data sheets. From testing several more similar products, there does not seem to be a reliable indicator for the latency behaviour of a given model – all tested consumer switches operate on the store-and-forward principle (as opposed to cut-through), where complete packets are buffered by the switch before routing them to the target port. However, empirically products that advertise extra features such as quality-of-service (QoS) support are more likely to add significant processing latency (even if those features are not in use).

To obtain more detailed information on the statistical distribution of latencies, a second experiment was set up where the MicroZed board simply echoed back any received UDP packets to their sender. The firmware for the ZedBoard would send out 2^{16} 2-byte UDP packets

containing a sequence number and measure the time until the reply arrives. The resulting histogram for the boards connected using a *Level One GSW-0807* switch is shown in figure 3.9. While UDP does not offer any guarantees regarding packet loss and ordering, all test packets arrived in order. The median and mean latencies were 30.08 μs and 30.10 μs, the 99.99-th percentile 31.09 μs, and the longest observed time was 40.82 μs.

Note that the data for this was acquired sending out IP broadcast packets from the ZedBoards, matching the situation for DEATH branching. When using packets addressed directly to the MicroZed board, the used switch would sporadically reorder packets, holding single ones of them in an internal buffer for tens of milliseconds. This might explain the discrepancy between the two presented measurements, where the round-trip latency appears to be smaller than twice the one-way latency by a non-negligible amount, as regular packets were used for the first measurement.

3.2.4 Output Calibration

The DEATH hardware was designed to provide an output voltage swing of at least ±9 V, with a linear mapping between digital output words and analogue voltages. [8] Nevertheless, there are small residual differences in the response of the channels, which are corrected to first (linear) order by an additional calibration step. In the first firmware revision, the calibration consisting of gain and offset adjustments used to be applied directly to the waveform data files as part of the generation process. This of course necessitated regenerating all used files every time the hardware calibration changed. Since the capability for making on-the-fly adjustments to electrode voltages was going to be built into the new firmware anyway, the output calibration was also folded into it.

As depicted in figure 3.7, each output data word is subjected to an affine linear transformation with configurable coefficients before being sent to the DACs. All calculations are performed using 16-bit fixed-point arithmetic, with the gain factor being in the interval $[0, 2)$. In other words, the transformation is described by

$$y(x) = \left\lfloor \frac{k}{32\,768} x + d \right\rfloor \tag{3.1}$$

for gain and offset coefficients k and d. Negative offsets can be applied by using the two's-complement representation.

In order to obtain the calibration coefficients, it is first necessary to determine what the maximum range supported should be. The absolute range limits slightly varied from channel to channel as to be expected, but were near −9.2 V and +9.1 V. Keeping the range symmetrical for convenience and budgeting for some further compensation due to drifts or ageing, a target range of $[-9\text{ V}, 9\text{ V}]$ was chosen for the full waveform data range of $[0, 2^{16} - 1]$. Note that this particular range is a user-level

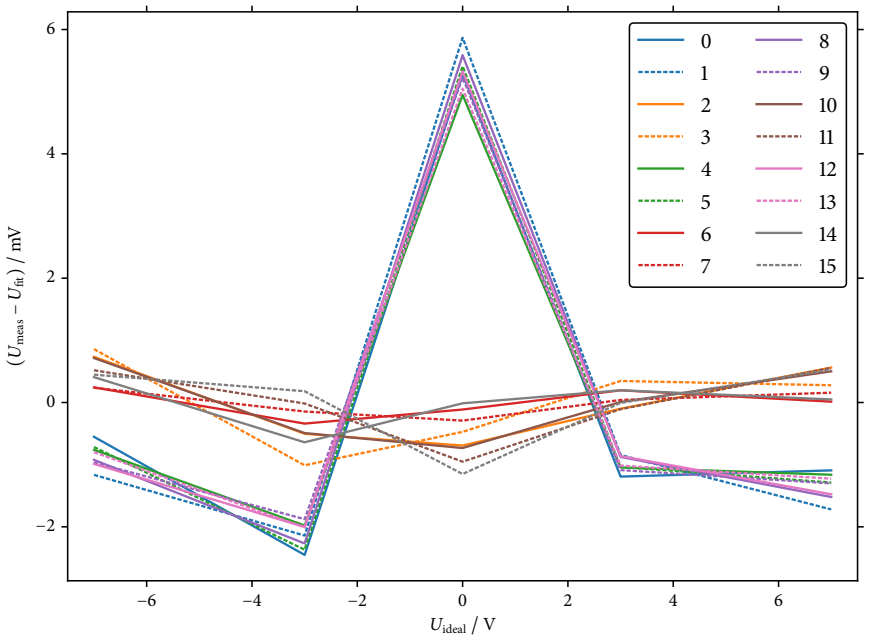

Figure 3.10: Linear fit residuals for the output DAC response curves of each DEATH channel (measurement resolution ≈ 0.5 mV due to noise). Data points at $U_{\text{ideal}} = \{-7, -3, 0, 3, 7\}$ V, lines to guide the eye. Two lines of the same colour correspond to the two channels driven by the same DAC chip.

choice that is not known by any part of the firmware or control libraries (with the exception of the Ionizer waveform loading code, which needs to convert the physical voltages from the waveform file to memory contents).

The actual response of each hardware channel then needs to be determined. A straightforward way to do this is to null out the calibration for all the channels (gain 32 768, offset 0) and load a waveform file with a set of fixed voltages across all the channels. For each channel, the output for a few different voltages is recorded. For the transport experiments performed thus far, this was done by manually measuring the outputs for waveform voltages of $U_{\text{ideal}} = \{-7, -3, 0, 3, 7\}$ V using a *Keithley 2100* $6\frac{1}{2}$-digit digital multimeter, although this could easily be automated. A linear polynomial is then fitted to the data to obtain the calibration coefficients according to (3.1).

The calibration constants are entered into Ionizer, which sends them to IonPulse on startup where they are in turn forwarded to the respective DEATH boards. Subsequently, the IonPulse user code can also alter this calibration to add a static offset to the voltage on a certain channel (modify d, leaving k constant) or to keep a certain electrode at a fixed voltage (set $k = 0$ and d to the desired value), which can be useful for calibrating some transport routines (e.g. splitting of ion strings).

The residuals from one such calibration measurement are shown in figure 3.10. Interestingly, two different classes of behaviour are visible: While there seems to be no clear trend for one half of the channels, the

other half exhibits a bump in the response around 0 V. Note that two channels sharing a DAC chip always show the same behaviour, while there is no such correlation across the two chips per physical PCB (0–3, 4–7, 8–11, 12–15). This suggests that the reason for this might be related to manufacturing differences between batches of DAC chips. No attempts were made to better characterise and compensate this non-linearity, since in practice other deficiencies in the model used for waveform generation due to imperfect geometry, stray fields and so on lead to corrections exceeding it in magnitude (cf. [35]).

No systematic investigations of the long-term drifts in calibration have been performed yet. Anecdotally, however, the output voltages appear to be stable to within ≈ 1 mV over the course of one month or so. Some measurements for shorter-term temperature-related drifts (on the scale of several hours) can be found in [8].

3.2.5 *Future Improvements*

The combination of new DEATH firmware and software has been in continuous use in our group for approximately a year with no major issues remaining. Transport is being used as part of loading the trap, and the system has also been used for the dedicated transport gate experiments described in [9] and [10]. Nevertheless, there are a few remaining improvements to be made.

Firstly, two particular adjacent channels on one circuit board (i.e. from the same DAC chip) experience an issue where for large jumps in voltages, the output will sometimes jump to a seemingly arbitrary value instead of to the target voltage. The values have been verified to be correct up until the differential buffer at the very end of the firmware pipeline. This suggests that the issue might be due to borderline timing on the parallel outputs to the DACs, maybe together with some ringing or crosstalk introduced by changes in the most significant bits of the output word. A first step to start investigating this would be to set up rigorous constraints for the timing relationship between data and clock outputs during FPGA design synthesis, which has not been done yet.

In terms of functionality, it would be useful for some experiments[15] to increase the amount of memory available to the system. A factor of two in capacity would be trivial to gain by synchronising the sequencing core to the interleaving done in the DAC output core to avoid discarding half of the data unused when executing a waveform without slowdown. It would also be fairly straightforward to extend the system to read waveform data from the on-board DDR3 RAM, which would make the waveform storage virtually unlimited (hundreds of megabytes).[16]

Another improvement would be to go back to using spline interpolation for generating the output data on-the-fly, as done previously for EVIL pulse shaping (cf. [8]). Since the waveform shapes are greatly oversampled – the low-pass filters before the trap have a corner frequency of 250 kHz, compared to the 50 MHz DAC update rate –, splines would

15. For example, to automatically explore a larger space of waveform parameters, although for this a script could also be used to upload new waveforms as needed.

16. This will require a somewhat careful approach in order to keep the output timing deterministic even though the latency through the memory subsystem is not. Since timing stability is more important than the absolute latency from trigger to start of playback, a fairly simple strategy of just buffering, say, 0.5 μs of waveform data would probably suffice, though.

considerably reduce the amount of memory needed for faithful reproduction. Perhaps more importantly, spline interpolation could also be implemented in such a way as to allow much more fine-grained control about the playback speed than is currently possible with the integer clock divider.

From a software architecture point of view, it would be desirable to increase the encapsulation of the interface exposed by the hardware. For example, if the flash memory present on the hardware is integrated into the firmware, it could be used to store the output calibration constants, and the waveform upload interface could subsequently be changed to accept voltages instead of raw 16-bit integers. In a similar fashion, now that it is clear that the networking protocol can support complex data structures with relatively little overhead, it might be desirable to manage all state (for example, the assignment of the numeric ids to sequences) on the DEATH ARM CPU and consequently also confine all the encoding of hardware tables to it. Such changes would make it easier to test and perform incremental improvements to individual devices in future, more complex quantum information experiments, by making it less likely that a certain change will require synchronised modifications to more than one system.

Finally, there currently is no convenient way of managing overrides to the output calibration constants from user code (as discussed in the previous chapter). It would be useful to add a thin layer of abstraction that saves the actual hardware calibration and then allows the user to specify voltages to pin a channel to or to add as an offset in physical units. Work on this has been started by Vlad Negnevitsky.

3.3 DASHBOARD FOR ELECTRONICALLY VARIABLE INTERACTIVE LOCK-BOXES (DEVIL)

As alluded to in the introduction to this chapter, an increasingly cumbersome aspect of the experimentalist's day in our lab was to keep track of the status of the ever-growing number of feedback control loops. In particular, this became apparent when what ultimately turned out to be an intermittent source of mechanical noise degraded the performance of the frequency-doubling cavities for addressing the $^{9}Be^{+}$ transitions. Similarly, a readily available overview of the state of the system would have been helpful when searching for the reason for slow drifts in the $^{40}Ca^{+}$ laser systems at one point.

Unless there are particularly stringent requirements in terms of loop bandwidth or noise performance,[17] the default PI controller solution in our group is the *EVIL* hardware originally started by Ludwig de Clercq and later co-developed by Vlad Negnevitsky. [8] Short for *Electronically Variable Interactive Lock-Box*, an EVIL device consists of a

17. These exceptions are mostly fast control loops locking laser currents to ultra-high finesse cavities, for which analogue servo circuits with a bandwidth on the order of 10 MHz are used.

backplane-mounted printed circuit board with two 10 bit analogue-to-digital converters and two DAC counterparts; one with 14 bit resolution and a parallel interface, the other with 12 bit resolution and a (slower) SPI interface. A commercially available daughter board houses a *Xilinx Spartan 3* FPGA that uses the two independent PI controller channels. Since the initial release, the firmware has undergone several revisions by Vlad Negnevitsky, Alexander Hungenberg and myself. [38, 39]

The FPGA offers a serial interface that can be used to configure the controller parameters and monitor several digital signals. It is exposed to the user via an *FTDI* USB-to-serial converter chip on the daughter board. Previously, a client program written in Python using the PyQt library was used to provide a graphical user interface for devices connected locally to the USB port of a PC, a single device at a time. There are several downsides to this approach. Firstly, the fact that the device needed to be connected directly to the PC controlling it led to a tangled nest of USB cables and repeaters throughout the lab, which caused various issues regarding ground loops and signal integrity. Secondly, the software supported no way of identifying the channels apart from the serial port numbers assigned by the operating systems, which are not persistent across reboots. Thirdly, this meant that naturally only a single PC could be used to monitor and manipulate a certain lock channel. Lifting the latter restriction was of particular interest in our case, as the $^{40}Ca^{+}$ laser systems are shared between three different trap projects. It would also allow easy addition of additional logging and monitoring systems, such as described later in the section.

A natural solution for all of these issues is to run a server executable on a computer close to (subsets of) the controllers, which identifies the connected devices and provides access to the rest of the laboratory via a network connection. This idea of quickly and locally converting control interfaces to Ethernet has become especially appealing with the advent of cheap off-the-shelf mini-computers, such as the *Raspberry Pi* series of ARM-based devices.

3.3.1 Design and Implementation

One possible way to add network support to the existing EVIL user interface would have been to use an emulated serial port driver on the control PC, which would forward the serial connection data over a TCP/IP network to another computer to which the device is actually connected. Such software packages exist and were experimented with in our group, but mostly seem to be designed for low data rates, lower than effective use of the data streaming feature requires. Another similar approach would be to simply break up the existing Python software – which already happened to use a background process for serial communication – into two parts that communicate over the network. The latter was attempted by Christoph Fischer, but he found that his software was two orders of magnitude slow to support streaming of even a single channel

from a *Raspberry Pi*. [40] Of course, both of these options still would not enable multiple clients to concurrently access the same hardware.

Thus, the decision was made to start from scratch and develop a system that cleanly abstracts away all the synchronous serial connection handling into the server part and allows several separate clients to connect. Following the group's acronyming tradition, the project was named *DEVIL*, where the first latter stands for the dashboard the new client software would provide. The design goal for the server software was to be able to control at least four EVILs from a single Raspberry Pi-style device, transparently handling hot-plugging (which is is important as some part of the laser system is being worked on most of the time), and providing sufficient streaming performance to allow a real-time view of all thirty-odd channels in our laboratory. It was also desirable that the new software require no modifications to the FPGA bitstream, so that the networked system could be rolled out incrementally and the old software would still be available as an emergency fall-back on the laptop computers in case of issues with the network control.

Because of the previous sobering performance results and the author's familiarity with developing performance-critical server applications in native programming languages, it was decided to write the server part in C++. To handle data input/output, the *Boost.Asio* library was selected since it supports non-blocking serial port input/output operations in addition to networking, contrary to alternatives such as *libevent* or *libev*. As a network messaging layer[18], *ZeroMQ* was used in the form of the *azmq* library.

The final server program consists of just under 3000 lines of C++ code (including extensive documentation comments), and uses *libudev* to continuously monitoring the system for hot-plug events. When a EVIL is detected, the serial number (read from the FTDI serial converter chip) is used to look up the channel names from a configuration file[19] and the serial connection is opened. For each logical channel (i.e. twice for dual-channel firmware versions) a *msgpack-rpc*-based control interface is opened on an automatically selected port and announced on the local subnet via a custom UDP-based resource discovery protocol (*Fliquer*). The single-threaded (event-based) server application continuously polls the hardware for status register updates and streaming packets (if requested), interleaving control commands as they are received from the clients.

In previous iterations of the client software iteration, the combination between the Python programming language and the Qt/pyqtgraph user interface libraries proved to work well for rapid and easily accessible GUI development. Cursory tests revealed that as long as updates were handled somewhat carefully, pyqtgraph would easily be fast enough to show a real-time overview of 50+ channels. Thus, the new client application was also developed on top of this technology stack – in fact, small portions of the control panel widget code could be directly reused. The client is now an entirely single-threaded[20] and single-process

18. The networking model is simple enough such that using raw sockets would not have required too much extra effort. However, ZeroMQ was already being used in a different project concurrently developed in the group anyway.

19. This is the only server-side piece of state/configuration.

20. As far as the user code is concerned, that is – the ZeroMQ library might internally use threads.

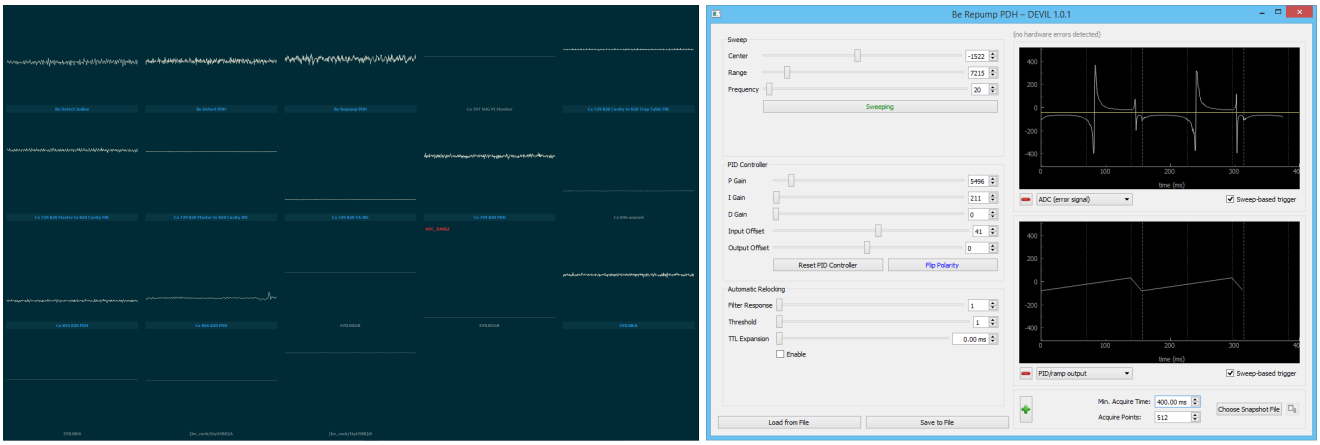

Figure 3.11: The PyQt-based DEVIL user interface, here running on Microsoft Windows 8.1. *Left:* The dashboard, giving an overview of status and error signals for the locks in the laboratory. *Right:* The interface for modifying the lock parameters of a single channel. A variable number of live plots can be displayed in the right half for error input, controller output and internal filter signals.

application, made possible by integrating the *pyzmq* library with the Qt event loop.

Altogether, the client program is made up of about 1900 lines of Python code and offers two main user interface components, as shown in figure 3.11. One is a dashboard that shows the lock status and error signal for a configurable subset of channels. The other is made up by the actual control panels for the parameters of each lock channel, the layout of which is based on the previous client software. The latter also allows the user to view additional streaming channels (such as the controller output) and to align the view of the streamed data to the hardware sweep generator cycle.

Although several hardware communication flaws were uncovered during the implementation of the new high-performance server software (see appendix A), the system has been tested to run stably with five client PCs concurrently displaying 30 channels. The precise data rates achieved by the system depend heavily on the acquisition settings and other factors. For a rough benchmark, one of the servers uses a 1.5 GHz ARMv7 core to about 17% to handle five connected EVILs. The client software uses about 8% of a single CPU core on a typical desktop computer[21] to display 3 MB s^{-1} of streaming data.

21. Intel i7-4790, 8 GiB RAM, Windows 8.1, Python 3.5.

3.3.2 *Long-Term Logging*

In addition to the real-time control software, a data logging client was also implemented. While based on the same Python code base as the graphical user interface, it is a basic command line application that subscribes to changes for all the channels on the local subnet. It monitors the error signal and controller output streams along with all the register values and writes the acquired data to an *InfluxDB* time series database on a remote server.

Figure 3.12: The *Grafana*-based long term logging web interface showing 1 h worth of data recorded for some of the locks in the $^{40}Ca^{+}$ laser system. The orange plots in the top half show the error signal input streams, the green plots in the bottom half the corresponding PI controller output. In the latter, slow drifts in the laser controllers and/or reference cavities are apparent.

For the long-term recording of data, the amount of streaming data received needs to be reduced considerably. As a compromise between time resolution (which is useful to detect correlations between drift/out-of-lock events) and bandwidth/storage requirements, a target stream update rate of one point each several seconds was chosen. In order to be less affected by changes to the stream acquisition settings in the regular client software, the logging client waits until chunks of 2^{16} samples have been accumulated for a given channel, then computes a statistical summary (mean, 20th/80th percentile, minimum/maximum) and pushes it to the database. The recorded data can then be accessed using a web interface based on the open-source *Grafana* software (see figure 3.12), or by directly using the InfluxDB API from a programming language like Python.

Averaging that many samples also helps with the 8 bit resolution limit of the hardware streaming implementation. To match the real-time user interface, the stream data is scaled by a factor of 4 such that each unit in the recorded data corresponds to a code point of the hardware input converter. The percentile measures thus appear quantized to 4 y-axis units. The analogue outputs have 14 bit resolution, which corresponds to 0.06 units.

3.3.3 *Future Improvements*

One possible direction for the future would be to design new hardware that integrates more higher-resolution channels onto a single board, maybe also directly integrating the networking capabilities by using a Zynq-like combined CPU/FPGA chip. However, there remains a number of improvements to be done to the current soft- and firmware.

Firstly, for certain applications, the 10 bit input resolution afforded by the analogue-to-digital converters is not sufficient. At the same time, the system operates at a higher clock speed than useful for the achievable loop bandwidth anyway (96 MHz vs. ≈ 500 kHz). Thus, one possible improvement would be to include a switchable averaging filter to increase the effective resolution by oversampling.

Secondly, the current controller firmware does not gracefully handle integrator overflow – the output just wraps around without any indication to the user. This behaviour should at least be detected and reported using the system status register, as done in [39]. Perhaps a feature that automatically clips the integrator values or just pauses the controller in this case would also be useful.

For both this as well as the automatic relocking mechanism developed in [38], it would be useful to route the out-of-lock status signal to the TTL port on the EVIL device. This signal could then be correlated with the scheduling information by the main control system to eliminate experiments affected by a laser glitch after the fact (or to possibly immediately retaking the affected data sets).

As for the client software, development initially targeted *PyQt* version 5, which is the most current version (Qt 5 is also used for Ionizer). However, pyqtgraph suffered from several hard-to-debug memory corruption issues when used with this version, which would lead to nonsensical plotted data and program crashes[22]. To work around the issues, the client was downgraded to *PyQt 4*. At some later point – once pyqtgraph officially supports Qt 5 –, it should be ported to version 5 again, at it is currently the only piece of software still relying on version 4 in use in the laboratory.

22. One likely cause for this would be an unforeseen interaction between the Python garbage collector and the Qt/C++ manual memory management system – it should not be possible to arbitrarily corrupt memory in Python at all.

Feature-wise, one useful addition to the software would be to detect when the control loop drifts close to the limits of the controller output range, and then display an alert to the experimenter so they can attend to the loop before it negatively affects the experiment in progress. These alerts could either be implemented in the main client software or, if the functionality is only wanted for slow drifts, on top of the long-term logging database.

STATIC QUBIT RESULTS 4

In preparation for evaluating the performance of the transport gates on the field-independent qubit in $^{9}Be^{+}$, the same experiments were performed on the conventional, static gates. This not only establishes a baseline for comparing the fidelities achieved with transport gates – no closer analysis of the static gates from a quantum-information point of view had been attempted in our group before –, but also served as a good test bed for the code to track and manipulate AOM phases in order to implement qubit rotations around different axes.

Due to ongoing problems with the $^{9}Be^{+}$ laser systems, randomised benchmarking was first performed on the $^{40}Ca^{+}$ quadrupole transition, even though the limited coherence time due to magnetic field fluctuations was likely to pose a considerable restriction for the achievable gate fidelities. The randomised benchmarking experiments are discussed in section 4.1, with a cursory estimation of the relative importance of some of the imperfections. Some proof-of-principle results for gate set tomography are presented in section 4.2.

The same experiment routines were also developed for $^{9}Be^{+}$, and attempts to run it were made at various points. However, even though the laser system was ultimately functional again, issues leading to large intensity fluctuations on the lasers for the Raman beam pair driving the qubit transitions were resolved only after the end of the project. Hence, the data for randomised benchmarking on the field-independent qubit presented in section 4.3 are to be taken merely as a proof of concept.

4.1 RANDOMISED BENCHMARKING ON $^{40}CA^{+}$

After developing the randomised benchmarking code for the 729 nm carrier transition of the $\left|4^2P_{1/2}, m = 1/2\right\rangle \leftrightarrow \left|3^2D_{5/2}, m = 3/2\right\rangle$ qubit in $^{40}Ca^{+}$, experiments were performed at various points as other aspects of the apparatus were improved. In this section the data with the highest fidelity obtained to date is presented, which is also the most recent data set.

4.1.1 Apparatus Calibration

For this data set, a harmonic potential centred in the main experimental zone of the trap was used, with (measured) frequency f_z = 2.02 MHz and a nominal offset of 1340 meV. The radial modes of motion were found at f_x = 2.26 MHz and f_y = 2.62 MHz.

Since previous experiments had shown that the deviation from the simple (motion-insensitive) carrier Hamiltonian due to population of the higher motional Fock states can considerably limit the reachable fidelities (for example, after only using regular Doppler cooling), some care was taken to optimise the cooling parameters first. The laser intensities and polarisations for Doppler cooling were first adjusted to match settings previously determined to be close to ideal for these well parameters. For EIT cooling, the pump beam intensity and the shared pump/probe beam detuning were optimised for cooling the first (x) radial mode, as diagnosed by driving its first red sideband transition. Cooling the axial (z) mode to the ground state using (pulsed) resolved sideband cooling in addition to that did not improve the observed fidelities any further.

The frequency of the 729 nm carrier transition was then calibrated using Ramsey experiments. The resolution during the calibration process (as limited by feature width and statistical uncertainties) was significantly better than 100 Hz. In addition to that, however, slow drifts on the order of 800 Hz per hour were observed, which would necessitate periodic recalibration of the frequency in longer experiments.

To stabilise the magnetic field of B = 11.96 mT the external current stabilisation and active feed-forward described in [41] were used. The latter was used to inject fields to cancel the first and third AC mains harmonics (50 and 150 Hz) at the position of the ion, with the phases and amplitudes chosen to visually cancel out the respective components in a line-triggered Bayesian-optimized Ramsey spectroscopy experiment with a wait time of 400 μs.

To calibrate the rotation angles for the gates, progressively longer sequences of the respective π- and $\pi/2$-rotations were used. For most of the experiments, the time for a $\pi/2$-rotation was fixed close to the limit of the current DDS hardware (which is already rather close to the limits imposed by the switching times of the used AOMs anyway). Figure 4.1 shows an experiment with 50 $\pi/2$-rotations about the x axis, where the gate duration is fixed to $t_{\pi/2}$ = 1.42 μs and the RF amplitude sent to the driver for the main double-pass switching AOM for the beam line is varied.[1] The laser intensity at the input of the AOM setup was actively stabilised using a photo-diode to feed back on the 729 nm tapered amplifier current.

1. In all experiments discussed here, the single-pass AOM used for creating multichromatic beams is switched on once before the beginning of the gate sequence and left at constant settings.

The sequence was executed using a special experiment definition which allows the experimenter to execute sequences of gates entered in string form (e.g. "50x" for a series of 50 $\pi/2$ rotations around the x-axis, or "xYx" corresponding to a π-rotation around the y-axis interleaved between two x $\pi/2$-rotations) using exactly the same routines as

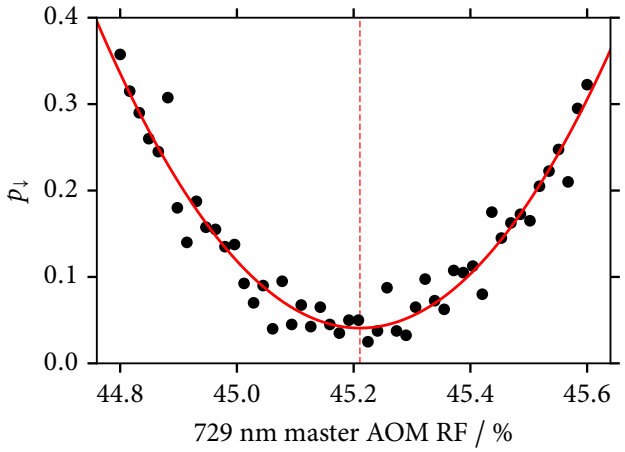

Figure 4.1: Bright state population after 50 $\frac{\pi}{2}$ pulses versus RF amplitude for the main switching double-pass AOM in the 729 nm beam path, to calibrate the laser intensity for $t_{\pi/2} = 1.42\ \mu$s. A parabolic fit is shown, centred around $a_{729} = 45.211(7)\%$ (arb. units).

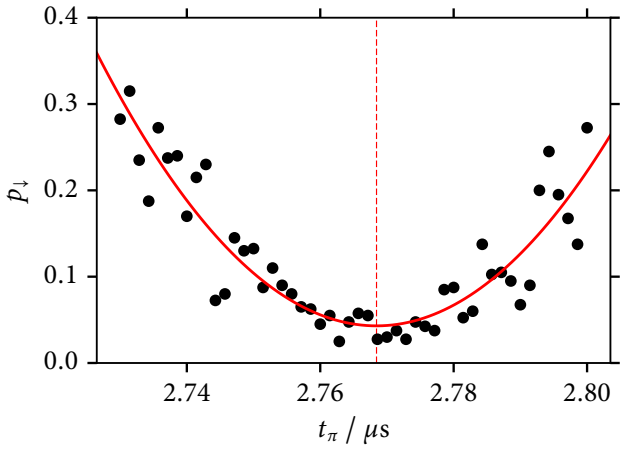

Figure 4.2: Bright state population after 25 π pulses versus RF duration per pulse. A parabolic fit is shown, with the calibrated pulse time being $t_\pi = 2.768(1)\ \mu$s.

later used for executing the automatically generated randomised benchmarking sequences. To absorb shot-to-shot intensity noise into reduced contrast for later analysis, $20 \cdot 50$ repetitions were performed per point. A parabolic fit was used to extract the minimum of $a_{729} = 45.211(7)\%$.[2] After calibrating the laser intensity in this way, the duration of the π gates was calibrated in a similar fashion using a string of 25 gates (see fig. 4.2). Again, a parabolic fit was used to obtain the gate time $t_\pi = 2.768(1)\ \mu$s (which is less than $2 \cdot t_{\pi/2}$ due to the finite AOM bandwidth).

4.1.2 *Results*

After calibrating the qubit rotations in this fashion, they were analysed using a randomised benchmarking experiment. The data shown here has been acquired using a scheme similar to the original publication [16], where in lieu of choosing a series of minimal realisations of Clifford group elements, the random sequences consist of a series of random "computational" $\pi/2$-pulses interleaved with random Pauli group elements (including $\pm\mathbb{1}$ and $\pm Z$, which are just implemented by altering the phase of the RF pulses sent to the AOMs for further gates, i.e. rotating the reference frame instead of modifying the physical state).

For the results presented here, a set of 16 different random $\pi/2$ rotation sequences was executed with 8 different random sequences of interleaved Pauli gates each, for a total of 128 random gate sequences. Due to issues with the communication between the ZedBoard and the DDS cards[3], the maximum sequence length was restricted to 102 computational gates, equivalent to a total number of 203 pulses per sequence. Each such sequence was also truncated to several shorter lengths, leading to a total of 1 408 sequences ranging between 2 and 102 computational

2. Since the precise shape of the curve depends on many uncharacterised response curves (DDS output, RF amplifiers, AOM diffraction efficiency), a parabola is used as an approximation under the assumption that the feature is symmetric. Due to the fact that a double-pass AOM is used, this is quite a good approximation as long as the scan range is small enough.

3. While not tracked down at the time of writing, the problem is likely related to the capacity of the control FIFO used to update pulse parameters. What triggers the issue is not the number of pulses, but the total number of pulse parameters sent through the FIFO during a sequence.

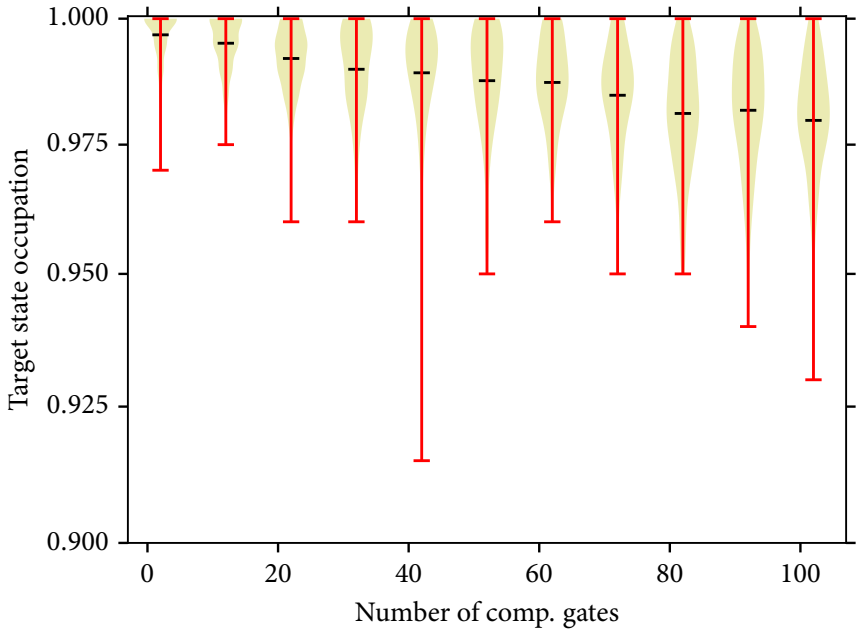

Figure 4.3: Mean $^{40}Ca^{+}$ target state populations for $16 \times 8 = 128$ random gate sequences truncated to different lengths (log-scale). Whiskers show minima/maxima, shaded areas a kernel density estimate. A least-squares fit yields an estimated average fidelity of 0.99983(1) per computational gate (an error rate of $1.69(1) \cdot 10^{-4}$) with the axis intercept being 0.9964(8).

gates in length. Due to constraints in the current DDS drivers, a pause of 1.4 μs between each gate pulse was inserted, which also helps absorb any transient behaviour in the main switching AOM[4]. Hence, the total duration per computational gate of $t_{cg} = 1.42\ \mu s + 1.40\ \mu s + 2.77\ \mu s + 1.40\ \mu s = 6.99\ \mu s$.

4. If the pulses were applied back-to-back, the gate times would have to be chosen based on the preceding pulses to reflect the AOM switching dynamics.

The gate sequences were generated on-the-fly on the ZedBoard CPU using a pseudo random number generator such that the expected target state would be $|0\rangle$ or $|1\rangle$ with equal probability (with the initial seed being set from and recorded by Ionizer for reproducibility). For each data point, the experiment was repeated 200 times. Shots where an unusually low number of scattered photons was observed during the Doppler cooling process (e.g. due to 397 nm laser dropouts or heating caused by background gas collisions) were detected on-the-fly and acquired again.

The results are summarised in figure 4.3. An exponential least-squares fit to the probabilities of finding the ion in the expected state yields an average infidelity of $1.69(1) \cdot 10^{-4}$ per computational gate. Note that the quoted uncertainty is merely the statistical error estimated from the fitting process – as also observed in other experiments and theoretically predicted in the presence of non-Markovian noise, the spread in observed outcomes for a given sequence length dominates it by far.

The full data set is shown in figure 4.4. As in the previous figure, the different sequence lengths are laid out horizontally, whereas the vertical axis now enumerates the different randomly selected gate sequences. The data points were acquired in two randomised scans over each lines,

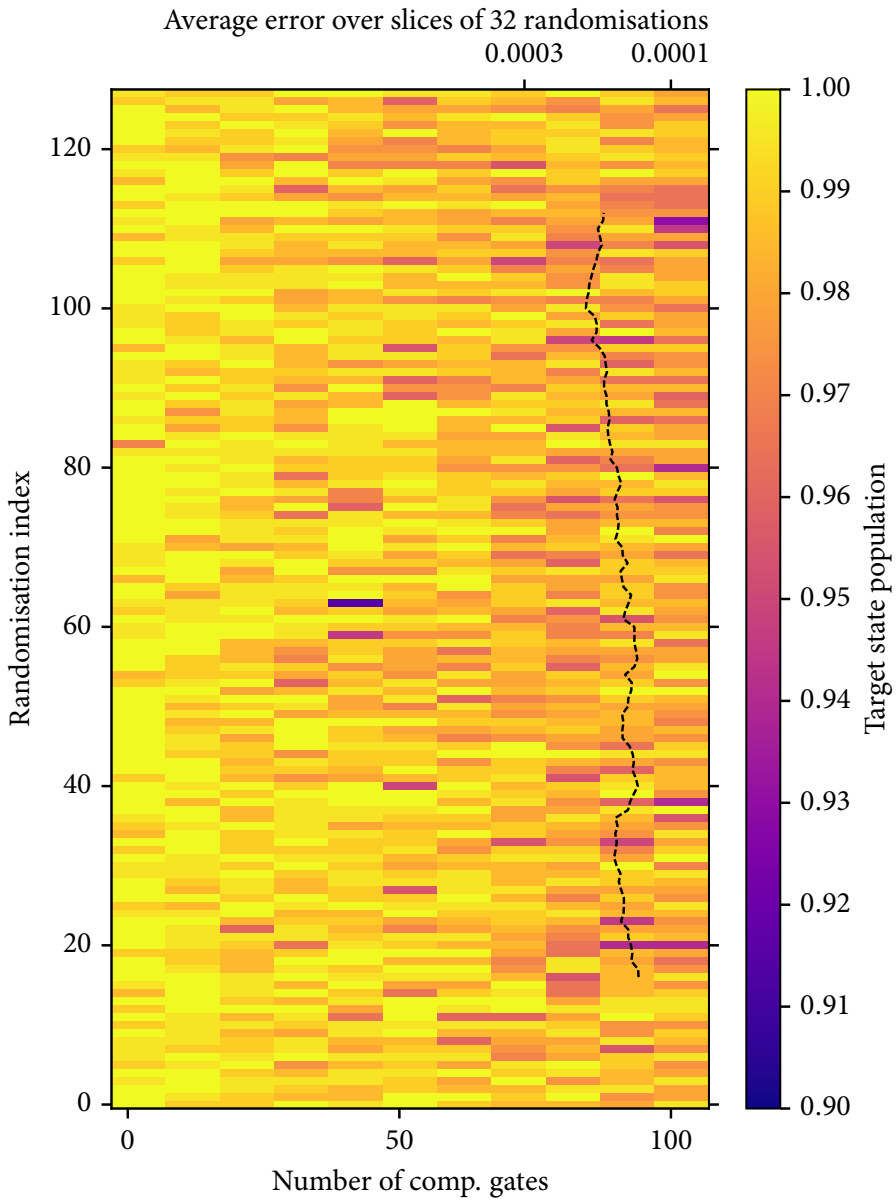

Figure 4.4: Results for individual $^{40}Ca^{+}$ randomised benchmarking sequences, with 200 repetitions per point. On the horizontal axis, the number of computational gates per sequences is shown. The vertical axis arbitrarily indexes different random sequences, with each block of 8 corresponding to a different set of Pauli randomisations for the same sequence of $\pi/2$ pulses. The signal has already been inverted where the dark state was the expected outcome. Overlaid is a dashed line showing the error as obtained from an exponential fit to a moving vertical slice of 32 randomisations each, where a slight drift over the total acquisition time of 650 s is visible.

but sequentially in the randomisation index. No drastic difference in fidelity between the data points taken at the beginning and the end of the total 11 minutes is visible, although there might be a slight systematic drift towards lower fidelities that warrants closer investigation for future high-fidelity experiments.

In the following sections, I will briefly consider the impact of a selection of known imperfections on the overall fidelity. This should not be taken as a full, rigorous characterisation of the system, however – none such was attempted, since the experiments with $^{40}Ca^{+}$ were only intended as a brief tangent on the way to evaluating $^{9}Be^{+}$ FIQ transport gates.

4.1.3 Error Sources

One source of errors that immediately comes to mind is imperfect control over the rotation angle of the gates. There are two possible issues here: Static miscalibration of the rotation angle, and fast intensity noise of the 729 nm laser light. As for the former, evaluating (2.21) for a relative miscalibration ε_Ω of the total rotation angle per computational gate (on average, π) yields

$$\mathrm{F}_{\text{int.miscal.}}(\varepsilon_\Omega) = \frac{1}{3}\left(2 + \cos(\pi\varepsilon_\Omega)\right) \tag{4.1}$$

Comparing the 95%-confidence intervals from the fits to the $\pi/2$ and π calibration scans shown in 4.1 and 4.2 to the respective widths of the features translates to a relative error in rotation angles of smaller than $\approx 1 \cdot 10^{-4}$. In addition to that, even with the intensity feedback loop engaged, relative long-term drifts of the Rabi frequency on the order of $10^{-3}\ \mathrm{h}^{-1}$ were observed, likely due to beam pointing instabilities. Even a conservative estimate of $\varepsilon_\Omega = 5 \cdot 10^{-4}$ yields only an infidelity of $4 \cdot 10^{-7}$ per computational gate – the contribution of static calibration errors to the total infidelity is negligible.

The amount of intensity noise can be estimated from the contrast in the calibration scans: Assuming that the actual shot-to-shot carrier Rabi frequency follows a Gaussian distribution around the ideal value with relative standard deviation σ_Ω, the signal of a Rabi flopping experiment is given by

$$\mathrm{p}_{|0\rangle}(\theta) = \frac{1}{2}\left(1 + \mathrm{e}^{-\frac{1}{2}\theta^2\sigma_\Omega^2}\cos(\theta)\right). \tag{4.2}$$

The data from figure 4.2 leads to an estimate of $\sigma_\Omega = 5 \cdot 10^{-3}$. Taking the ensemble average for an ideal rotation with angle π leads to the dephasing channel given by the Kraus operators $\left\{\sqrt{1-p}\mathbb{1}, \sqrt{p}\sigma_z\right\}$ with $p = \frac{1}{2}(1 - \mathrm{e}^{-\frac{1}{2}\pi^2\sigma_\Omega^2})$, resulting in a fidelity of

$$\mathrm{F}_{\text{int. noise}}(\sigma_\Omega) = \frac{1}{3}\left(2 + \mathrm{e}^{-\frac{1}{2}\pi^2\sigma_\Omega^2}\right), \tag{4.3}$$

or for the above value, an error per computational gate of $4 \cdot 10^{-6}$.

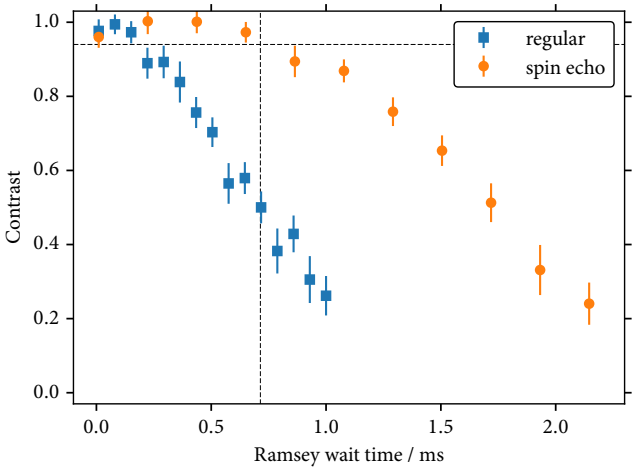

Figure 4.5: Contrast in a Ramsey experiment (amplitude of oscillations when scanning the phase of the second $\pi/2$ pulse) on the $^{40}Ca^{+}$ 729 nm carrier transition as a function of wait time, with and without an interleaved spin-echo π-pulse. Dashed lines show duration of a 102-computational-gate randomised benchmarking sequence and the level of contrast equivalent to the result from fig. 4.3.

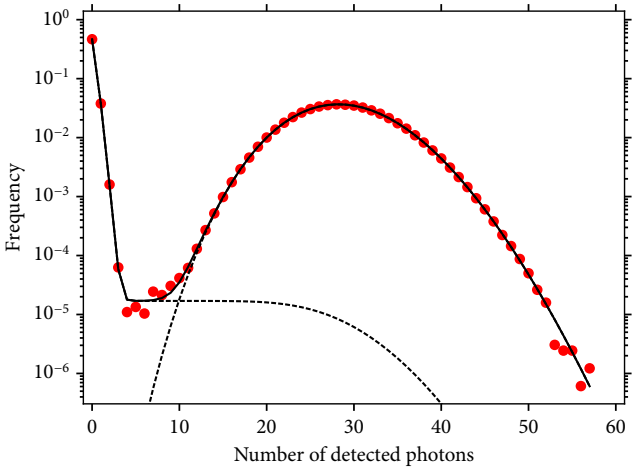

Figure 4.6: Distribution of photon counts observed during $^{40}Ca^{+}$ readout (t_r = 180 μs window, $1.63 \cdot 10^6$ shots total). Lines show a double-Poissonian model including decay from the dark state and Gaussian intensity noise on the detection laser, with parameters obtained using maximum-likelihood estimation: $\lambda_{|1\rangle} = 0.082$, $\lambda_{|0\rangle} = 28.7$, $\sigma_{397} = 1.7\%$, $\tau = 0.18$ s.

In summary, effects from imperfect laser intensity control are currently negligible. Similarly, a cursory estimate shows that even if the static qubit frequency error were to reach 100 Hz due to the observed slow drifts, the error per gate would still be below 10^{-5}. However, there is a significant amount of frequency noise affecting the qubit on shorter time scales, as determined from the loss of contrast in Ramsey experiments with varying wait times (figure 4.5). Comparing the data to the sequence duration of 713 μs for 102 computational gates, it is clear that this decoherence process is currently the main source of errors. In fact, the target state populations observed in randomised benchmarking almost reach the results for the spin echo sequence, matching the insensitivity of randomised benchmarking sequences to slow noise predicted in [17]. The hypothesis that this is the dominant error source is also corroborated by additional experiments with an additional wait time inserted in between gates, where an accordingly lower fidelity was observed.

From comparison to the coherence times observed in other traps using the same laser source, these fluctuations seems to be mainly due to magnetic field fluctuations rather than phase noise on the 729 nm laser. Giving a precise expression for the loss of fidelity due to frequency noise is hard, however, as it depends heavily on its power spectrum (which is known to be very much non-white due to residual pick-up at the harmonics of 50 Hz, the AC mains frequency). [42]

Even though randomised benchmarking is insensitive to errors in state preparation and measurement[5], the characteristics of the readout process are still relevant for quantum information experiments mak-

5. As long as they are uncorrelated to the sequence length, they do not affect the observed decay used to derive the error per gate in the ideal case at all.

ing use of those operations. The overall distribution of the number of photons detected during the state-dependent fluorescence of a number of experiments can be modelled as a weighted mixture of two Poissonian processes with different mean values for the dark state, $\lambda_{|1\rangle}$, and the bright state, $\lambda_{|0\rangle}$. In addition, there is a finite probability that $|1\rangle$ spontaneously decays into the bright state mid-way during the fluorescence process and then scatters photons for the remainder of its duration. This is a function of the ratio of readout duration to dark state lifetime, $\gamma = t_r/\tau$, and extends the count distribution for $|1\rangle$ by a low "shelf". If the 397 nm fluorescence laser exhibits an appreciable amount of intensity noise[6] – most easily assumed to be Gaussian with some relative standard deviation σ_{397} –, the distributions are accordingly broadened.

6. In our experiment, the light at 397 nm is generated using frequency doubling in a resonant cavity, which regularly results in a moderate amount of intensity noise without further stabilisation.

The overall distribution observed during a sequence of randomised benchmarking experiments (including the main result presented here) is shown in figure 4.6, and can be described well using such a model. No special care was taken to optimise the readout process parameters, and with a threshold of 10 counts, the probability for misclassifying $|0\rangle$ as $|1\rangle$ is $\varepsilon_0 = 6 \cdot 10^{-5}$, and its reverse $\varepsilon_1 = 5 \cdot 10^{-4}$, yielding an optimal average error of $\varepsilon_{\text{avg}} = 3 \cdot 10^{-4}$. This could be improved by optimising the experimental parameters (laser intensities, etc.) and then choosing the optimal detection time accordingly, and even further by using using time-resolved methods. From the intercept of the fit to figure 4.3, a total state preparation and measurement error of $3.6(8) \cdot 10^{-3}$ can be extracted, which suggests that the bulk of the state preparation and measurement errors is due to the pumping process used for initialisation.

Note that the $|1\rangle$ lifetime of $\tau = 0.18$ s derived from the maximum-likelihood fit of the model to the data is significantly less than the expected lifetime of the state in vacuum, $\tau_0 = 1.1$ s. While τ should only be regarded as a lower bound for the actual lifetime since there might be additional non-Poissonian sources of dark counts, this suggests that there might be a significant leakage of the 854 nm laser light used to reset the qubit. As this bound for the lifetime is still two orders of magnitude longer than the coherence time due to dephasing, it is currently not the dominating factor. Since the fidelity for the corresponding decay channel described by the Kraus operators

$$\left\{ |0\rangle\langle 0| + \mathrm{e}^{-\frac{\gamma}{2}} |1\rangle\langle 1| \,,\, \sqrt{1-\mathrm{e}^{-\gamma}}\, |0\rangle\langle 1| \right\}$$

evaluates as

$$\mathrm{F}_{\text{decay}}(\gamma) = \frac{1}{6}\left(3 + \mathrm{e}^{-\gamma} + 2\,\mathrm{e}^{-\frac{\gamma}{2}}\right), \tag{4.4}$$

which leads to an error per gate of $1.3 \cdot 10^{-5}$, this might be interesting to investigate in the future as other sources of errors are reduced.

One source of errors not considered here is the residual influence of the ion's thermal motion. The carrier Hamiltonian is independent of the motional state only in the Lamb-Dicke approximation, and since the residual motion of $\bar{n} \approx 10$ after Doppler cooling (i.e. without EIT

cooling) caused carrier flops to loose contrast faster than the other mechanisms discussed here, the magnitude of this coupling should be evaluated more carefully. In addition, unwanted excitation of the sidebands due to auxiliary peaks in the laser spectrum[7] would entangle the qubit state with the ion's motion and hence lead to decoherence. In future investigations, the effect of the AC stark shift induced by the 729 nm laser during gates should also be considered. While it is possible to correct for experimentally, it is currently neglected (the unperturbed frequency is used for the laser drive as well as the phase calculations).

7. For example, "servo bumps" due to the finite bandwidth of the laser frequency control loop.

4.2 GATE SET TOMOGRAPHY ON $^{40}CA^{+}$

The same method of calibrating and executing gate sequences as described in the last section was also used to perform gate set tomography on the optical qubit in $^{40}Ca^{+}$. The target gate set consisted of the two $\pi/2$ rotations $R_x = R_x(\pi/2)$ and $R_y = R_y(\pi/2)$, together with the identity $R_i = \mathbb{1}$. For the numerical representation, the Pauli basis was chosen for $\mathcal{S}(\mathbb{C}^2)$. Hence, the ideal state preparation and measurement operators are given by

$$\begin{aligned} \rho_0 = |0\rangle\langle 0| \quad &\leftrightarrow \quad |\rho_0\rangle\!\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}^{\mathrm{T}} \\ E_0 = |0\rangle\langle 0| \quad &\leftrightarrow \quad \langle\!\langle E_0| = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix} \end{aligned} \tag{4.5}$$

and the matrices for the target CPTP gate maps G_x, G_y are illustrated in figure 2.1.

As suggested in [28], an over-complete set of fiducial sequences was chosen, $F = \{(), G_x, G_y, {G_x}^2, {G_x}^3, {G_y}^3\}$. The main gate strings used consisted of the germs[8]

$$\begin{aligned} G = \{&G_x,\, G_y,\, G_i,\, G_xG_y,\, G_xG_yG_i,\, G_xG_iG_y,\, G_x,\, G_xG_iG_i, \\ &G_yG_iG_i,\, G_xG_xG_iG_y,\, G_xG_yG_yG_i,\, G_xG_xG_yG_xG_yG_y\}. \end{aligned} \tag{4.6}$$

8. This choice is not particularly interesting if only a single repetition of the germs sequences is used; they are designed for longer eGST experiments.

338 distinct gate sequences were executed using the Ionizer scripting interface, for an overall total of 396 000 detections. After applying the detection threshold, the resulting data was imported into the pyGSTi software package for further analysis. [32]

A linear GST estimate was first computed, contracted to the nearest CPTP gate set, and then further refined by using a weighted least-squares approximation (as discussed in [28]) to the full maximum-likelihood problem for the observed data. In the final step, the gauge transformation was chosen to minimise the Frobenius distance to the target gate set and the amount of CPTP violation. The final estimates for state preparation and readout and the gate maps are given in figure 4.7.

Note that the calculated 95% confidence intervals are frequently larger than the distance to the corresponding ideal operators. This

$$|\rho_0\rangle\!\rangle = \begin{pmatrix} 0.7071 \\ 0.0081 \\ 0.0003 \\ 0.7130 \end{pmatrix} \pm \begin{pmatrix} 0.0002 \\ 0.0023 \\ 0.0026 \\ 0.0002 \end{pmatrix} \qquad \langle\!\langle E_0| = \begin{pmatrix} 0.7140 \\ 0.0059 \\ -0.0004 \\ 0.7002 \end{pmatrix} \pm \begin{pmatrix} 0.0002 \\ 0.0025 \\ 0.0024 \\ 0.0002 \end{pmatrix}$$

$$G_i = \begin{pmatrix} 0.9998 & -0.0003 & 0.0013 & 0.0003 \\ 0.0002 & 0.9997 & -0.0022 & 0.0101 \\ 0.0020 & -0.0056 & 0.9985 & 0.0054 \\ 0.0003 & 0.0015 & -0.0005 & 0.9992 \end{pmatrix} \pm \begin{pmatrix} 0.0025 & 0.0029 & 0.0030 & 0.0026 \\ 0.0029 & 0.0026 & 0.0061 & 0.0068 \\ 0.0030 & 0.0064 & 0.0026 & 0.0063 \\ 0.0026 & 0.0064 & 0.0063 & 0.0026 \end{pmatrix}$$

$$G_x = \begin{pmatrix} 0.9994 & -0.0002 & -0.0002 & 0.0002 \\ -0.0008 & 0.9979 & -0.0311 & -0.0472 \\ -0.0002 & -0.0357 & 0.0059 & -0.9978 \\ -0.0003 & 0.028 & 0.9992 & 0.0067 \end{pmatrix} \pm \begin{pmatrix} 0.0007 & 0.0016 & 0.0012 & 0.0012 \\ 0.0017 & 0.001 & 0.0038 & 0.0034 \\ 0.0013 & 0.0026 & 0.0035 & 0.0008 \\ 0.0013 & 0.0027 & 0.0008 & 0.0021 \end{pmatrix}$$

$$G_y = \begin{pmatrix} 0.9999 & -0.0002 & 0.0003 & -0.0002 \\ -0.0003 & 0.0105 & 0.0314 & 0.9990 \\ 0.0004 & 0.0424 & 0.9992 & -0.0352 \\ 0.0003 & -0.9986 & 0.0411 & 0.0082 \end{pmatrix} \pm \begin{pmatrix} 0.0009 & 0.0012 & 0.0016 & 0.0013 \\ 0.0012 & 0.0038 & 0.0041 & 0.0009 \\ 0.0017 & 0.0029 & 0.0010 & 0.0027 \\ 0.0012 & 0.0009 & 0.0030 & 0.0022 \end{pmatrix}$$

Figure 4.7: GST estimate for state preparation, readout and gates in the Calcium-40 optical qubit, given in the Pauli basis. Uncertainties are half-widths of the respective 95% confidence intervals(see text).

is even more apparent when looking at the fidelities calculated from these results: The infidelities for G_i, G_x and G_y are 0.0007 ± 0.0025, 0.0014 ± 0.0008 and 0.0008 ± 0.0009, respectively. In other words, the gates are too close to the target unitaries to reliably distinguish them with the short sequences used and $\approx 10^3$ runs each. Thus, the given confidence intervals are not to be taken to be exact; they extend into unphysical (non-CPTP) gates and, consequently, fidelities greater than 1. No attempt to extract further information from the data is thus made, as any closer analysis e.g. of the error symptoms would suffer from similar caveats.

To combat this, one could adopt a more rigorous method that derives confidence intervals that represent some operational meaning, for example the algorithm presented in [30]. However, the much more straightforward solution would simply be to simply take data for extended long-sequence GST, avoiding the issues entirely by making sure the confidence region is well contained within the subset of physical gates. All the provisions for this are already in place in the current control infrastructure; the extension should be as simple as modifying a few lines in the data acquisition script.

4.3 RANDOMISED BENCHMARKING ON $^9BE^+$

The experiments analogous to those presented for $^{40}Ca^+$ were also implemented for the different $^9Be^+$ qubit transitions (FDQ/FIQ) and Raman beam configurations (collinear/90°, with the option to use the micromotion sidebands for the latter). In theory, the FIQ should offer the best performance of all the options in the current experimental setup, as it mostly eliminates decoherence due to magnetic field fluctuations. With the default Raman detuning of ≈ 250 GHz, the achievable fidelities would be limited to $\approx 1.5 \cdot 10^{-4}$ due to off-resonant scattering [37] similar to those achieved in $^{40}Ca^+$, but this could be further reduced by going to larger detunings at the expense of longer gate times or higher laser intensities.

Indeed, during calibration the qubit frequency could easily be determined to better than 5 Hz using Ramsey experiments, and even without carefully re-calibrating the magnetic field to match the field-insensitive point, the loss of contrast due to decoherence was negligible for wait times of 100 ms, which is much longer than any of the sequences that would be used for benchmarking or tomography. While operations on the field-dependent qubit – which is even more sensitive to field fluctuations than the $^{40}Ca^+$ optical qubit – are necessary for state preparation and readout, both randomised benchmarking and gate set tomography would be insensitive to errors caused by them.

However, at the time there were issues with keeping the second-harmonic generation cavity used for generating the 313 nm Raman beams on resonance with the input lasers, leading to large intensity fluctuations. Their relative amplitude has been measured to reach ≈ 7%, which according to (4.3) limits the error per pulse area of π to at least $8 \cdot 10^{-3}$.

To perform randomised benchmarking, the gate times were calibrated using the same approach as described in section 4.1.1, yielding $t_{\pi/2} = 9.57\ \mu$s and $t_\pi = 18.69\ \mu$s. The average duration of coherent FIQ manipulations during the longest used benchmarking sequences (150 pulses) was thus 1.49 ms. During calibration, slow intensity drifts on the order of 1% over some minutes were noticeable in addition to the fast noise which will become significant after the fast noise has been fixed, but should be easy to counteract by an AOM feedback loop.

One example set of results is shown in figure 4.8. This particular dataset has been acquired in a slightly different fashion from the protocol used for $^{40}Ca^+$; namely, by choosing between $\pi/2$ and π pulses at random[9], which has been shown to lead to slightly lower error estimates than the original scheme, as the $\pm R_z(\pi/2)$ gates are included in the set of gates and $\pm\mathbb{1}$ and $\pm\sigma_z$ occur with a higher frequency (all of which are simply implemented by modifying the phase of subsequent gate pulses). The exponential fit yields a fidelity of 99.51(3) % per gate pair, which

9. This was done with the idea of directly randomising over the full Clifford group to be direct correspondence with the theoretical treatment of randomised benchmarking as twirling the error channel over it. Unfortunately, a comparison with the table from figure 2.2 shows it to be insufficient.

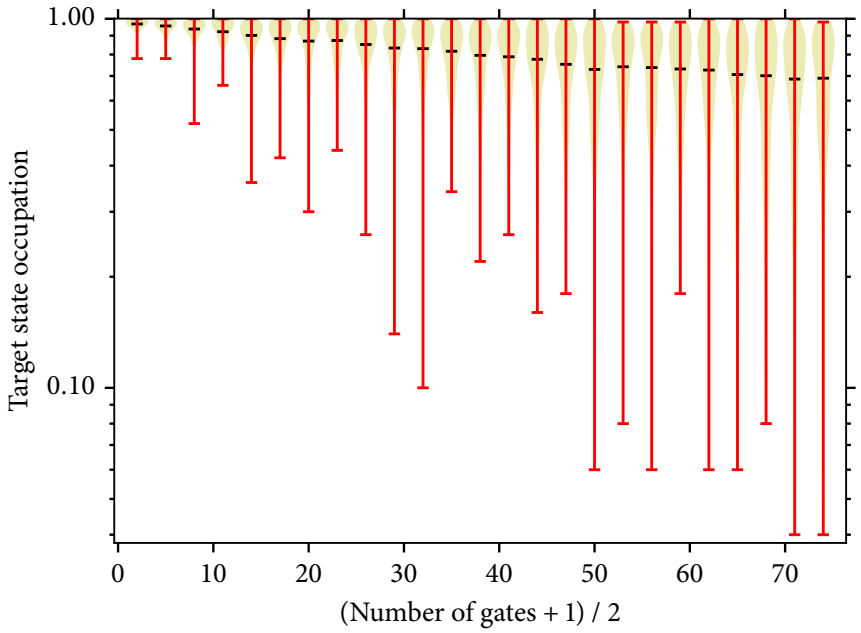

Figure 4.8: Mean $^9Be^+$ FIQ target state populations for 125 random gate sequences truncated to different lengths (log-scale). Whiskers show minima/maxima, shaded areas a kernel density estimate. A least-squares fit yields an estimated average fidelity of 0.9951(3) per gate pair, or an error rate of $4.9(2) \cdot 10^{-3}$, with the axis intercept being 0.97(1).

suggests that the the large intensity fluctuations were indeed the factor limiting the fidelity in the experiments presented here.

The data was acquired with Doppler cooling only, but the coupling to the Raman beams in collinear configuration is to first order insensitive to the ion motion. In fact, the ions exhibited a large amount of excess micro-motion, the modulation reducing the amount of light scattered during fluorescence detection by as much as 50%. If the 90° Raman beam configuration – as required for coupling to the motional modes – were to be used for high-fidelity gates, the effect of this would need closer study (along with the relative phase stability of the two beams).

The full set of data is shown in figure 4.9. This time, considerable drifts in the average fidelities over the total acquisition duration of 16 min for the $125 \cdot 25 \cdot 50 = 156\,250$ points are visible, likely due to the aforementioned drifts in the Raman laser intensity.

Note that there are some outliers where target state populations of below 0.5 are reached for rather short sequences, only for the average to then tend to 0.5 again as expected as the sequences are extended. It might be that this is due to sequences which are particularly sensitive to over- or under-rotations, but it is also possible that that there might be an issue in the program code used for translating the gates to laser pulse phases that is only triggered for certain combinations of gates. This will be easy to investigate once the intensity stability has been improved by just executing and analysing one affected sequence in isolation, tracking the evolution of the state.

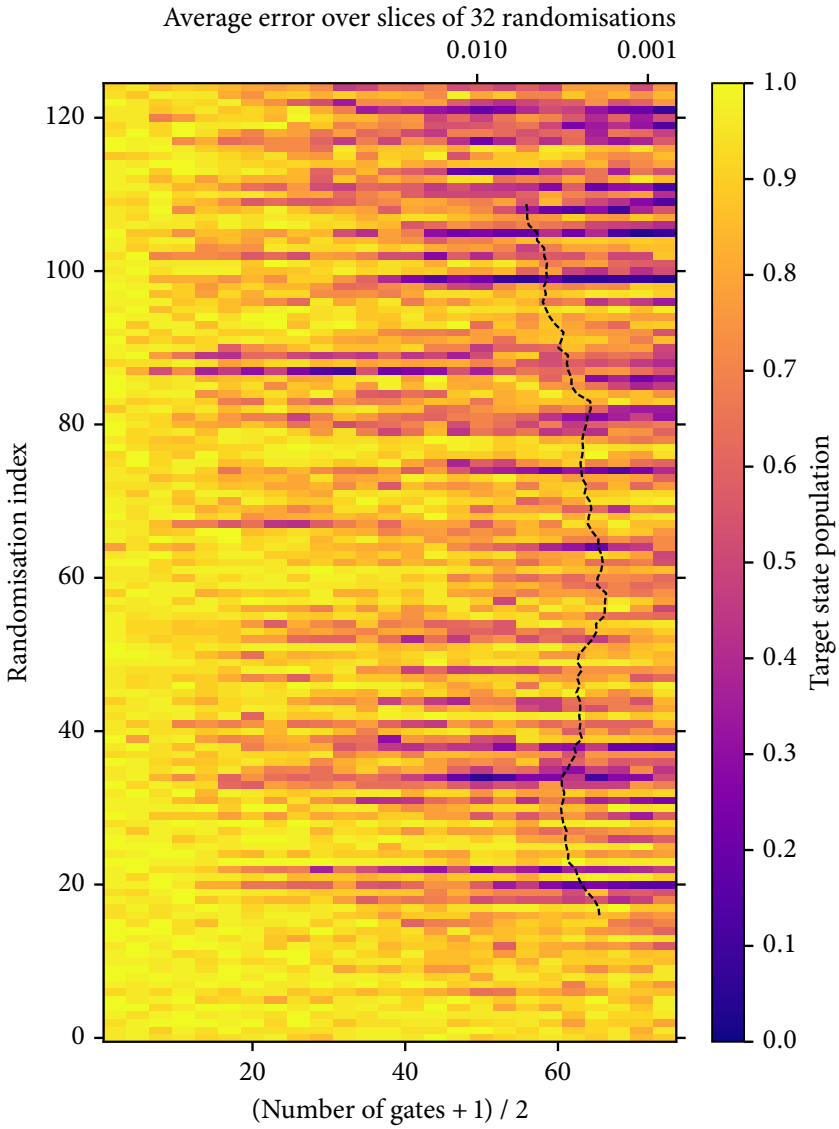

Figure 4.9: Results for individual $^{9}Be^{+}$ FIQ randomised benchmarking sequences, with 50 repetitions per point. The main (lower) horizontal axis shows the number of pairs of gates randomly chosen from all possible $\pi/2$- and π-pulses, including $\pm\mathbb{1}$ and $\pm\sigma_z$. The signal has already been inverted where the dark state was the expected outcome. Overlaid is a dashed line (top axis) showing the error obtained from an exponential fit to a moving slice of 32 randomisations each, showing considerable drifts over the total acquisition time of 974 s.

Off-resonant excitations of other transitions in the $S_{1/2}$ manifold due to imperfect Raman-beam polarisation will also need to be considered, as will – similar to $^{40}Ca^{+}$ – the AC stark shift of the qubit transition during gates due to the presence of the Raman beams (although the latter can again be suppressed almost perfectly by taking the two different frequencies into account for the AOM drive calculations).

OUTLOOK 5

The ultimate goal behind most of the work completed as part of this project was to demonstrate transport quantum logic gates with a comparable fidelity to that reachable in the usual way, where the ions remain static. All the components necessary for this have been realised:

A completely new firmware was developed for the DEATH waveform generator system in order to make complex experiments feasible, by integrating it with the main experimental control system and adding the capability to synchronise complex control flows between them. The general laboratory control system itself has been extended in multiple ways to make it possible to automate involved experimental sequences and to offer better diagnostic capabilities for performing longer experiments that require high-fidelity operations.

In addition to that, two schemes for evaluating the performance of quantum gates have been discussed in the abstract, and demonstrated in the two different ion species available in the experimental setup. The developed control system components and analysis programs are sufficiently general to be reusable for other gate implementations.

What unfortunately remains yet to be done is the last step, combining the above to demonstrate and characterise a complete set of high-fidelity single-qubit transport gates in the field-independent $^{9}Be^{+}$ qubit. This is merely a question of fixing the issues with stability and performance of the laser system used to drive the qubit transition. Once stable operation and satisfactory fidelities in the field-independent qubit are reached (e.g. $1 - F = 10^{-4}$), it should be straightforward to demonstrate transport gates at a comparable level. The level of control over the ion velocity already demonstrated in this apparatus [10, 35] is sufficient not to introduce any significant intensity errors. The gate times will increase slightly, but due to the long observed coherence times, the effect of this should still be minimal. The considerable insensitivity of the transition frequency to changes in the magnetic field [43] should also be sufficient to protect against the observed spatial variations on the 0.1 μT-level along the transport axis.

For static $^{40}Ca^{+}$ single qubit gates, it is clear that the limiting factor is currently the stability of the magnetic field environment, necessitating further work on field stabilisation and cancellation, and/or additional shielding. To inform such measures, a limited amount of information about the frequency spectrum of the fluctuations could be gained by performing randomised benchmarking experiments for a number of

different gate times, where the different influences of low-frequency coherent fluctuations and fast noise would be visible. It seems prudent, however, to augment this by a full characterisation of the noise spectrum at the position of the ion, for example using methods like that demonstrated in [44]. With that knowledge, it might be possible to further increase the fidelity by designing composite gate pulse sequences that are insensitive to the dominant noise frequencies. [45]

Finally, I expect that the control infrastructure will continue to receive significant amounts of work as the trend towards more complex experiments continues, in the short term in particular regarding the scheduling of more complex, concurrent sequences and the automated tracking and optimisation of calibration parameters. In the years to come, the quest for a scalable quantum information processing platform composed of individual modules will also provide formidable challenges in experimental control. It is my hope that the perhaps uncharacteristically extensive coverage of some of the design decisions behind the current system in this report will provide useful background for such discussions.

A EVIL HARDWARE COMMUNICATION GLITCHES

While working on the new networked client software for the EVIL loop controller (see chapter 3.3), several previously unknown issues in the hardware communication were uncovered, particularly in combination with the ARM-based mini-computers tested.

The first concerns the hardware start-up process: Whereas previously a connection would only be opened after the user selected a serial port and pressed a button in the server program, the new client software continuously listens for hot-plug events from the operating system device manager. This led to the discovery that if data is sent on the serial connection within ≈ 2 s after hardware power-on, the connection is likely to become completely inoperable. The reason for this is unclear, but it might be related to the FPGA re-configuration process. As a workaround, the software now waits 3 s before establishing the connection to a newly enumerated device.[1]

1. This is also the cause for the delay between connecting a controller and the respective channels being displayed in the UI; if the initial timeout is removed, the process is virtually instant.

Secondly, it was observed that when using a *Raspberry Pi* for a server, using a USB hub would render the communication non-functional, even with just one connected device. Several models were tested, both powered and non-powered, and inevitably the USB-to-serial converters either would not be enumerated by the operating system in the first place, or the communication would time out as soon as the server software tried to access the hardware. This was observed both with a *Raspberry Pi 1 Model B+*, featuring a single-core *Broadcom BCM2835* chipset and running the official *Ubuntu*-based Linux distribution, and the newer *Raspberry Pi 2 Model B* with its quad-core *BCM2836* CPU, this time running *Arch Linux ARM* with kernel version 3.XXX. No workaround for this was found; it seems plausible that this is due to some interaction between the USB stack in those systems-on-a-chip (which are intended for applications in mobile phones and similar devices), some packet handling peculiarities of common UBS hub chipsets, and/or the FTDI serial converter drivers.

As connecting just three EVILs to each Raspberry Pi would leave a lot of CPU power and network bandwidth unused, alternative hardware in the form of the *Hardkernel ODROID-C1* board was investigated. Its specifications are quite similar to the Raspberry Pi 2, but it notably uses a different USB chipset. And indeed, the hardware did not seem to show any issues regarding USB hubs; up to eight hardware channels

connected via a 13-port USB 3 hub were tested successfully. However, at a later time it was found that upgrading the Arch Linux ARM kernel package to version 3.18.XX lead to the serial links stopping to transmit or receive data if continuously used for some ten seconds. This occurred independently of whether a USB hub was used, and downgrading the kernel to version 3.XX solved the issue.

It was also discovered that for requests for streaming packets longer than approximately 800 bytes, sometimes too little data would arrive back over the serial connection when using an acquisition interval that would correspond to a data rate near the nominal connection speed (3 Mbit s^{-1}). The detailed characteristics of this failure mode (for example, which pieces of the packet are missing) were not investigated any further, but the fact that the packet length triggering it did not seem to depend on the precise data rate suggest the problem to somehow involve a constant buffer of some hundred bytes of data. Given the lack of handshaking on the UART connection between FPGA and serial converter chip, one possible explanation is that some of the data sent is ignored by the latter because it is e.g. internally busy exchanging some buffers with its USB protocol handling section. The idea that the USB bus part is involved in the issue is corroborated by the fact that it it would occur with the ODROID-C1 board for parameters that were stable when used with a Raspberry Pi 2.

In the end, these unresolved issues were worked around by slightly limiting the sample acquisition rates for long packets as well as implementing heuristics for returning the connection to a valid state after data corruption has taken place. This is non-trivial because the EVIL serial protocol uses variable-length commands with no explicit message framing. There is some anecdotal evidence that similar issues also occurred with the old client software, i.e. when driving the connection from a regular USB chipset as found in an x86-based PC or laptop. Because only individual controller channels were observed for short periods of time back then, such issues would have likely happened infrequently enough to just be ignored.

In spite of all those caveats, the resulting system runs stably over extended periods of time. The server software has been observed to be running continuously for several months without any interventions being required, although the communication issues are triggered from time to time when changing the streaming settings, causing the data not to be updated momentarily as the recovery heuristics are run.

LIST OF SOFTWARE REPOSITORIES

B

As mentioned in chapter 3, additional documentation for most of the software and firmware projects developed or extended as part of this work can be found along with their source code. The following list provides an overview of the respective group-internal Git repositories to make locating it easier.

bram_ctrl_16b_adapter A small FPGA core to convert a 32-bit wide memory to a 16-bit wide interface. (Verilog)

ca-ionizer The main graphical user interface for the experimental control system. Extended as described in section 3.1, along with a number of general bug fixes. (C++)

cavity-pi-server Provides a network interface to control reference cavity piezo actuators from Ionizer. Its implementation was partially rewritten to use *tiqi-rpc*. (Python)

death_control The client libraries for the DEATH waveform generators, used from Ionizer and IonPulse. (C++)

death_firmware_arm The program running on the DEATH ARM CPU. Notably, the file *network-interface.md* describes the various functions of its tiqi-rpc interface in considerable detail. (C++)

death_firmware_fpga Ties together the various other FPGA cores to form the main DEATH bitstream. (Xilinx Vivado/Verilog)

devil The client software for the networked PI controller described in chapter 3.3, providing the graphical user interface and long-term logging. (Python)

devil_server Runs on a Linux system and exposes several connected EVIL controllers to an IP network, as discussed in chapter 3.3. (C++)

ionpulse_sdk The program executed on the main control system ZedBoard, significantly extended during this project (network communication layer, ability to detect and react to transitory errors). Contains the sequence definitions developed for the transport, randomised benchmarking and tomography experiments. (C++)

linear_transform A small FPGA core to apply an affine linear transformation with coefficients set via an AXI4-Lite interface. (Verilog)

scythe The main sequencer core of the DEATH FPGA firmware. Contains a description of the various internal memory layouts. (Verilog)

tiqi-notifier Small helper script to dispatch real-time alerts from the control system via e-mail, Slack and SMS. (Python)

tiqi-rpc Implementations of the network protocol discussed in section 3.1.1 for several platforms, along with a detailed description of the protocol. (C++/Python)

tiqi-rpc-dissector Provides support for decoding the tiqi-rpc protocol in the Wireshark network traffic analyser. (C++/Lua)

tiqi-tomography Data analysis scripts for randomised benchmarking and gate set tomography experiments. (Julia/Python)

TIQI.jl Functions for importing, analysing and plotting results saved from Ionizer. (Julia)

xilinx_standalone_emu An adapted version of lwIP and other Xilinx-specific libraries to execute and test Zynq-based software on a personal computer, including its networking stack. (C)

BIBLIOGRAPHY

[1] R. P. Feynman, "Simulating physics with computers", International Journal of Theoretical Physics **21**, 467 (1982).

[2] T. D. Ladd, F. Jelezko, R. Laflamme, Y. N. Y, C. Monroe, and J. L. O'Brien, "Quantum computers", Nature **464**, 45 (2010).

[3] J. Benhelm, G. Kirchmair, C. F. Roos, and R. Blatt, "Towards fault-tolerant quantum computing with trapped ions", Nature Physics **4**, 463 (2008).

[4] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computer.", Nature **417**, 709 (2002).

[5] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, "Laser-driven quantum logic gates with precision beyond the fault-tolerant threshold", arxiv:1512.04600 (2015).

[6] R. B. Blakestad, C. Ospelkaus, A. P. Vandevender, J. H. Wesenberg, M. J. Biercuk, D. Leibfried, and D. J. Wineland, "Near-ground-state transport of trapped-ion qubits through a multidimensional array", Physical Review A **84**, 1 (2011).

[7] D. Leibfried, E. Knill, C. Ospelkaus, and D. J. Wineland, "Transport quantum logic gates for trapped ions", Physical Review A **76**, 32324 (2007).

[8] L. de Clercq, "Transport quantum logic gates for trapped ions", PhD thesis (ETH Zürich, 2015).

[9] L. E. de Clercq, H.-Y. Lo, M. Marinelli, D. Nadlinger, R. Oswald, V. Negnevitsky, D. Kienzler, B. Keitch, and J. P. Home, "Parallel Transport Quantum Logic Gates with Trapped Ions", Physical Review Letters **116**, 080502 (2016).

[10] L. E. de Clercq, R. Oswald, C. Flühmann, B. Keitch, D. Kienzler, H.-Y. Lo, M. Marinelli, D. Nadlinger, V. Negnevitsky, and J. P. Home, "Estimation of a general time-dependent Hamiltonian for a single qubit", Nature Communications **7**, 11218 (2016).

[11] D. Kienzler, C. Flühmann, V. Negnevitsky, H.-Y. Lo, M. Marinelli, D. Nadlinger, and J. P. Home, "Observation of Quantum Interference between Separated Mechanical Oscillator Wave Packets", Physical Review Letters **116**, 140402 (2016).

[12] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (2011).

[13] A. Gilchrist, N. K. Langford, and M. A. Nielsen, "Distance measures to compare real and ideal quantum processes", Physical Review A **71**, 1 (2005).

[14] L. H. Pedersen, N. M. Møller, and K. Mølmer, "Fidelity of quantum operations", Physics Letters, Section A **367**, 47 (2007).

[15] A. Y. Kitaev, "Quantum computations: algorithms and error correction", Russian Math. Surveys **52**, 53 (1997).

[16] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Randomized benchmarking of quantum gates", Physical Review A **77**, 012307 (2008).

[17] H. Ball, T. M. Stace, S. T. Flammia, and M. J. Biercuk, "The effect of noise correlations on randomized benchmarking", Physical Review A **93**, 022303 (2016).

[18] S. Anders and H. J. Briegel, "Fast simulation of stabilizer circuits using a graph-state representation", Physical Review A **73**, 1 (2006).

[19] E. Magesan, J. M. Gambetta, and J. Emerson, "Scalable and robust randomized benchmarking of quantum processes", Physical Review Letters **106**, 8 (2011).

[20] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking", Physical Review A **85**, 042311 (2012).

[21] J. M. Epstein, A. W. Cross, E. Magesan, and J. M. Gambetta, "Investigating the limits of randomized benchmarking protocols", Physical Review A **89**, 1 (2014).

[22] I. L. Chuang and M. a. Nielsen, "Prescription for experimental determination of the dynamics of a quantum black box", Journal of Modern Optics **44**, 2455 (1997).

[23] H. Häffner, W. Hänsel, C. F. Roos, J. Benhelm, D. Chek-al-Kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür, and R. Blatt, "Scalable multiparticle entanglement of trapped ions.", Nature **438**, 643 (2005).

[24] S. T. Merkel, J. M. Gambetta, J. a. Smolin, S. Poletto, A. D. Córcoles, B. R. Johnson, C. a. Ryan, and M. Steffen, "Self-consistent quantum process tomography", Physical Review A **87**, 062199 (2013).

[25] R. Blume-Kohout, J. K. Gamble, E. Nielsen, J. Mizrahi, J. D. Sterk, and P. Maunz, "Robust, self-consistent, closed-form tomography of quantum logic gates on a trapped ion qubit", arxiv:1310.4492 (2013).

[26] D. Kim, D. Ward, and C. Simmons, "Microwave-driven coherent operations of a semiconductor quantum dot charge qubit", Nature Nanotechnology **10**, 243 (2014).

[27] G. Feng, B. Buonacorsi, J. J. Wallman, F. H. Cho, D. Park, T. Xin, D. Lu, J. Baugh, and R. Laflamme, "Estimating the coherence of noise in quantum control of a solid-state qubit", arxiv:1603.03761 (2016).

[28] R. Blume-Kohout, J. K. Gamble, E. Nielsen, P. Maunz, T. Scholten, and K. Rudinger, *Turbocharging Quantum Tomography*, tech. rep. (Sandia National Laboratories, Albuquerque, New Mexico, 2015).

[29] D. Greenbaum, "Introduction to Quantum Gate Set Tomography", arXiv:1509.02921 (2015).

[30] P. Faist and R. Renner, "Practical, Reliable Error Bars in Quantum Tomography", arxiv:1509.06763 (2015).

[31] M. Christandl and R. Renner, "Reliable quantum state tomography", Physical Review Letters **109**, 1 (2012).

[32] E. Nielsen and K. Rudinger, *pyGSTi: Version 0.9.1 Alpha* (doi:10.5281/zenodo.46200, Feb. 2016).

[33] D. J. Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, and D. M. Meekhof, "Experimental issues in coherent quantum-state manipulation of trapped atomic ions", Journal of Research of the National Institute of Standards and Technology **103**, 259 (1998).

[34] D. Kienzler, "Quantum harmonic oscillator state synthesis by reservoir engineering", PhD thesis (ETH Zürich, 2015).

[35] R. Oswald, "Velocity Control of Trapped Ions for Transport Quantum Logic Gates", MSc thesis (ETH Zürich, 2015).

[36] L. Gerster, "Spectral filtering and laser diode injection for multi-qubit trapped ion", MSc thesis (ETH Zürich, 2015).

[37] H.-Y. Lo, "Creation of Squeezed Schrödinger's Cat States in a Mixed-Species Ion Trap", PhD thesis (ETH Zürich, 2015).

[38] A. Hungenberg, "Automatic relocking of an FPGA-based PID controller using a bandpass-filtering approach", Semester project report (ETH Zürich, 2013).

[39] D. Nadlinger, "Laser Intensity Stabilization and Pulse Shaping for Trapped-Ion Experiments using Acousto-Optic Modulators", Semester project report (ETH Zürich, 2013).

[40] C. Fischer, "Implementation of a Digital Lock-in Amplifier on a Field Programmable Gate Array and its Remote Control in a Local Area Network", Semester project report (ETH Zürich, 2015).

[41] C. Flühmann, "Stabilizing lasers and magnetic fields for quantum information experiments", MSc thesis (ETH Zürich, 2014).

[42] Z. Chen, J. G. Bohnet, J. M. Weiner, and J. K. Thompson, "General formalism for evaluating the impact of phase noise on Bloch vector rotations", Physical Review A **86**, 1 (2012).

[43] C. Langer, R. Ozeri, J. D. Jost, J. Chiaverini, B. Demarco, A. Ben-Kish, R. B. Blakestad, J. Britton, D. B. Hume, W. M. Itano, D. Leibfried, R. Reichle, T. Rosenband, T. Schaetz, P. O. Schmidt, and D. J. Wineland, "Long-lived qubit memory using atomic ions", Physical Review Letters **95**, 2 (2005).

[44] S. Kotler, N. Akerman, Y. Glickman, and R. Ozeri, "Nonlinear single-spin spectrum analyzer", Physical Review Letters **110**, 1 (2013).

[45] T. Green, H. Uys, and M. J. Biercuk, "High-order noise filtering in nontrivial quantum logic gates", Physical Review Letters **109**, 1 (2012).