ETH ZÜRICH

Integrated laser amplitude stabilization for mixed species trapped-ion experiments

by

Martin Stadler

A thesis submitted in partial fulfillment for the degree of Master of Science

supervised by
Dr. Vlad Negnevitsky
Prof. Jonathan Home

in the
Trapped Ion Quantum Information Group
Institute for Quantum Electronics

September 2018

"But it works fine in simulations...
Why?"

- Most FPGA developers at some point

ETH ZÜRICH

Abstract

Trapped Ion Quantum Information Group
Institute for Quantum Electronics

Master of Science

by Martin Stadler

One of the major sources of infidelity in trapped-ion quantum gates is intensity noise. A custom built control system for trapped ion quantum information experiments was extended to incorporate two separate laser amplitude stabilisation mechanisms. Firstly, a relatively slow feedback loop was implemented on the main control card to stabilise slow drifts on the timescale of seconds. A fast PID controller that is tightly integrated into the RF pulse generation for the AOMs controlling the laser amplitudes was written and tested. The ability to stabilise the laser amplitude in the presence of certain noise during an experimental sequence is demonstrated. The control system firmware and software were enhanced to be able to perform longer pulse sequences.

Acknowledgements

First of all, I want to thank Prof. Jonathan Home for awakening my interest in Quantum Information Processing by giving enthusiastic lectures on the topic. Thanks for allowing me to conduct my master thesis at the TIQI group. It was a great opportunity to work together with highly skilled, motivated and cooperative scientists.

In particular, thanks to Dr. Vlad Negnevitsky for being patient with me as I dived into the world of FPGA programming and got my C++ skills up to par. It was a great chance to be able to try out new approaches and have the freedom to make my own mistakes.

Thanks to Tanja Behrle and Xiaosi Xu for providing the implementation of randomized benchmarking to test out my improvements of the system.

Thanks as well to Matteo Marinelli and Karan Mehta for helping me out in the lab ever so often.

All in all, it has been a pleasure to work with the TIQI group.

Contents

Chapter 1

Quantum information processing

1.1 Prelude

History is full of examples of novel technologies overtaking old ones with remarkable performance gains over a certain period of time. In general, this growth in performance follows an S-curve [1], which means it starts exponentially but levels out when the process reaches its limits, sometimes given by the laws of physics. These limits can only be overcome by new approaches that tackle the problem from a different angle. There are countless examples of this principle for instance the audio distribution industry that started with phonograph records, moved on to Cassettes, which were superseded by CDs, which are about to be replaced by online music distribution services.

Computing is no exception in that regard. The first Turing-complete device already was envisioned in the first half of the 19th century. However, it took around 100 years until the first usable Turing-complete machine was built. Huge improvements were made shortly before and during the second world war, resulting in the first computers, first on purely mechanical, later on partly electrical (relay) basis. Relays were replaced by electric tubes which worked at much higher frequencies. The discovery of the transistor in 1956 heralded the replacement of tubes by semiconductor technology in the 1960s. 10 years later, the first integrated microprocessor went into mass production. Since then, immense efforts in shrinking semiconducting structures made it possible to increase the area density of computational elements. Currently used fabrication processes approach fundamental limits of physics which will put an end to the exponential evolution in the foreseeable future. The effects are already visible as exploding costs for every "shrink", as a transition to smaller structures is called, is a formidable problem for manufacturers [2].

A new approach is needed to overcome these hurdles. Richard Feynman ignited the vision of a fundamentally more capable way to solve computational problems in his legendary talk at the

First Conference on the Physics of Computation at MIT in 1981 [3]. He proposed the first model of a basic quantum computer to overcome the exponential scaling problem when simulating many-body quantum systems. Several quantum computing technologies have emerged in pursuit of overtaking classical computers. "Quantum supremacy" is a term which was shaped in this context. It can be interpreted as solving a computational task which is not feasible with existing classical supercomputers. This could be completing a certain task using a quantum computer faster than a classical one. It could also mean solving a problem which would require more memory than any classical computer can provide. It has yet to be demonstrated in practice and there are several critics questioning if quantum supremacy is a meaningful term [4].

1.2 Theoretical foundation

Basic mathematical operations can be carried out efficiently in a binary numeral system. The quantum analogue to the classical bit is called the qubit which can be described by a state vector composed of the orthogonal vectors $|0\rangle$, $|1\rangle$ or any complex superposition between the two. As all physical wavefunctions, the qubit has to be normalized.

$$\begin{gathered}\langle 0|1\rangle = 0 \\ \alpha\,|0\rangle + \beta\,|1\rangle = |\psi\rangle \in \mathbb{C}^2 \\ |\langle\psi|\psi\rangle|^2 = 1\end{gathered}$$

In the n qubit-case, there are 2^n coefficients according to the 2^n states which can be superposed. On the one hand, this gives rise to the exponential scaling problem for many-body quantum systems but on the other hand this is the foundation of the enormous computational power of well-controlled quantum systems. Clever quantum algorithms have been developed that leverage the power of superposition to provide exponential speed-up. The most notable being Shor's algorithm [5] which would break one of the most commonly used asymmetric encryption algorithm, RSA. To actually run a program on a quantum computer, gates between the qubits and on the qubits themselves need to be performed.

Operations on a qubit are usually described as rotations, which is obvious when considering the Bloch-sphere representation. Because of the normalization and the irrelevance of a global phase, only two real parameters are needed to describe the qubit state.

$$\psi = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle \tag{1.1}$$

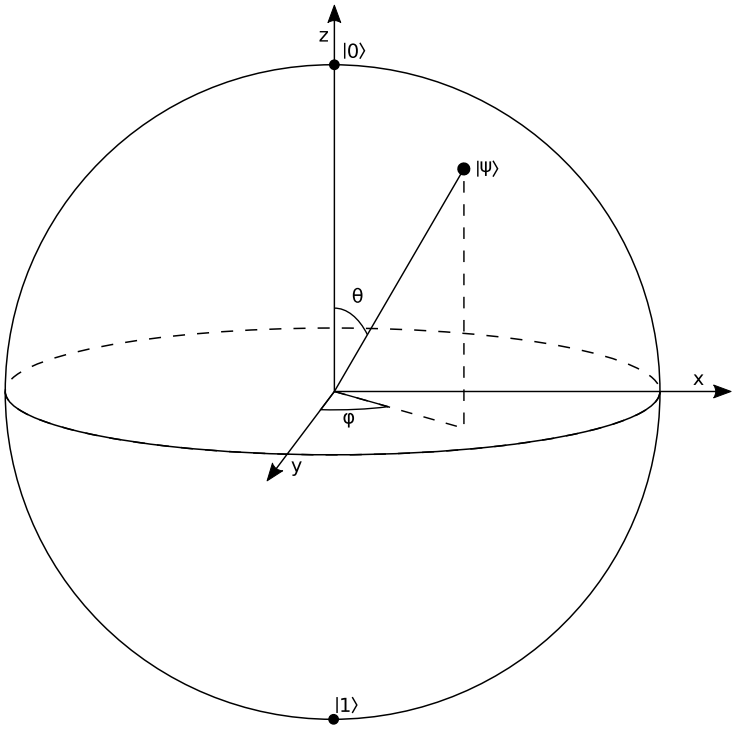


Figure 1.1: Bloch sphere representation of the state in eq. (1.1)

The angle θ and ϕ can be visualised as polar and azimuthal angle (see Fig. 1.1). Orthogonal states are on opposite sides of the sphere.

In this picture, any ideal qubit operation, i.e. unitary, acts as a rotation around a certain axis times a global phase. Fixed rotations are called gates, which are the building blocks of quantum information processing (QIP). For a better introduction and deeper insights see [6].

1.2.1 Gate implementation

The interaction between electromagnetic waves and two level systems (qubits) are described within the framework of quantum optics leading to the optical Bloch equations. A detailed description can be found in common quantum optics textbooks or Florian Leupold's thesis which is more focused on ion trapping [7]. If incoherent interaction is neglected i.e. a closed system is considered and the driving field is close to resonance with the qubit transition, the Hamiltonian of the system can be written in a simple form. Time-dependence can be removed by transforming to the rotating frame of the laser with angular frequency ω_l [7, pp. 26]:

$$H^T = -\hbar\delta \left|0\right\rangle\!\left\langle 0\right| + \frac{\hbar\Omega}{2}\left(\left|1\right\rangle\!\left\langle 0\right| e^{-i\alpha} + \left|0\right\rangle\!\left\langle 1\right| e^{i\alpha}\right)$$

The Rabi frequency, Ω, is the coupling strength between the driving field and the qubit. The described system oscillates between being in the ground or exited state which is called Rabi

oscillation. Ω is directly proportional to the electric field amplitude. The small detuning between laser and qubit is called δ. α is the phase of the driving field.

The time-evolution operator can directly be calculated from this Hamiltonian and corresponds to rotations in the Bloch sphere around the angular frequency vector $\vec{W}$. However, this is in the rotating frame. This means that z-rotations in the lab frame are simply performed by waiting with zero laser intensity and changing the phase of the driving field afterwards.

$$\vec{W} = (\Omega\cos(\alpha), \Omega\sin(\alpha), -\delta) \tag{1.2}$$

The realisation of a theoretically defined gate always has a certain error that needs to be quantified. There are several functions that can be used for this purpose. Fidelity is a useful quantity to describe how close the actually achieved state is to the target state. This is usually done in the more general density operator formalism, which combines quantum states with statistical ensembles.
It formulates states as operators which can be represented as two by two matrices in the qubit case. In the orthogonal basis $|\psi\rangle$ and $|\bar{\psi}\rangle$ this reads:

$$\rho = p_\psi |\psi\rangle\langle\psi| + p_{\bar{\psi}} |\bar{\psi}\rangle\langle\bar{\psi}|$$

Where p_ψ and $p_{\bar{\psi}}$ are the probabilities of dealing with the corresponding state. This formalism allows treating quantum systems with incomplete knowledge about it's state.
Density operators can be visualised on the Bloch sphere, where pure states (one probability is one, the other one is zero) are on the unit sphere and mixed states are inside of it. The fully mixed states where both probabilities are 0.5 is positioned at the center.

The closeness of two density operators is often characterised using the fidelity which is defined as

$$F(\rho, \sigma) \equiv \mathrm{tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}} \tag{1.3}$$

In the case of pure states this is equivalent to $F = \sqrt{|\langle\psi|\phi\rangle|^2}$. This equals the square root of the probability of measuring the state $|\psi\rangle$ in the basis state $|\phi\rangle$ [6, pp.409].

1.2.2 Randomized benchmarking

In general, it is hard to perform an arbitrary rotation with high accuracy. Therefore, a finite set of gates which can be calibrated in reasonable time is chosen, which can be used to construct a universal set of gates. This means that any arbitrary rotation can be approximated up to the required accuracy.

Table 1.1: Clifford group representation used in the TIQI group

$C_1 = I$	$C_7 = R_x(\pi)$	$C_{13} = R_x(\pi)R_y(-\frac{\pi}{2})$	$C_{19} = R_x(-\frac{\pi}{2})R_y(\pi)$
$C_2 = R_z(\frac{\pi}{2})$	$C_8 = R_x(\pi)R_z(\frac{\pi}{2})$	$C_{14} = R_x(-\frac{\pi}{2})R_z(\frac{\pi}{2})$	$C_{20} = R_x(-\frac{\pi}{2})$
$C_3 = R_z(\pi)$	$C_9 = R_x(\pi)R_y(\frac{\pi}{2})$	$C_{15} = R_y(\frac{\pi}{2})$	$C_{21} = R_x(\frac{\pi}{2})R_y(-\frac{\pi}{2})$
$C_4 = R_z(-\frac{\pi}{2})$	$C_{10} = R_y(-\frac{\pi}{2})$	$C_{16} = R_x(-\frac{\pi}{2})R_y(\pi)R_z(\frac{\pi}{2})$	$C_{22} = R_x(\frac{\pi}{2})$
$C_5 = R_y(\pi)$	$C_{11} = R_x(\frac{\pi}{2})R_z(\frac{\pi}{2})$	$C_{17} = R_x(-\frac{\pi}{2})R_y(-\frac{\pi}{2})$	$C_{23} = R_x(\frac{\pi}{2})R_y(\pi)$
$C_6 = R_y(\pi)R_z(\frac{\pi}{2})$	$C_{12} = R_x(\frac{\pi}{2})R_y(\pi)R_z(\frac{\pi}{2})$	$C_{18} = R_x(-\frac{\pi}{2})R_y(\frac{\pi}{2})$	$C_{24} = R_x(\frac{\pi}{2})R_y(\frac{\pi}{2})$

An efficient way to estimate the average gate fidelity is called Randomized Benchmarking (RB), proposed in 2008 [8]. As the name suggests, a gate sequence from a predefined set is chosen at random and applied to the initial state $|0\rangle$. A set used for this purpose is the Clifford group which corresponds to permuting the x, y and z-axis on the Bloch sphere in the single qubit-case. The z-axis can be freely put at any of the six directions. This allows for four positions of the x-axis. The y-axis is uniquely defined by the positions of z and x. Therefore, there are $4 \cdot 6 = 24$ elements in the Clifford group. This includes all π and $\pi/2$ rotations around the three axes. It is not a universal set of gates but can easily be made one by adding one $\pi/8$ rotation gate to the set.

One particular representation of the group is listed. Note that there are several possibilities to arrive at the same qubit state. The chosen one in Tab. 1.1 is the same as in the actual implementation used by the RB experiment that is used later on in this thesis. In the case of elements with multiple rotations, operations are executed from right to left. Although gates consist of up to three rotations, the minimum sequence to implement the corresponding rotation takes only one z-rotations and another x or y-rotation [9].

After a random sequence has been applied, a final gate from the Clifford group is applied such that one arrives at the randomly chosen final state, $|0\rangle$ or $R_x(\pi)|0\rangle = |1\rangle$. This way, the measurement outcome is always deterministic in the computational basis if no error occurs. The average gate fidelity is equal to the n-th square root of the error rate between expected and actually measured result if n-1 gates are performed. The final state is randomly chosen to avoid correlated errors.

1.3 Mixed-species 3D segmented trap

There are several qubit implementations with different strengths and weaknesses, most notably superconducting qubits, Nitrogen-Vacancy centers, quantum dots and trapped ions. The

Trapped-Ion-Quantum-Information (TIQI) group, where this master thesis was completed, is working with several implementations of ion traps.

In this master thesis all experiments where carried out on the segmented trap setup which was mostly constructed and built by Daniel Kienzler [10] and Hsiang-Yu Lo [11]. There are several people in the group extending and maintaining the setup including Vlad Negnevitsky [12] and Matteo Marinelli [13]. For a detailed description of the geometry and the electric and magnetic fields used in the trap please see their work.

Essentially, the segmented trap is a Paul trap with several DC electrodes. Due to the Earnshaw theorem, a dynamic field is necessary to trap a charged particle just with electric fields. 2 RF and 2 DC electrodes in one plane can be used to create a pseudo potential that keeps an ion radially confined. 2 DC electrodes in the axial direction of the trap are needed to completely trap the ion. By using segmented DC electrodes for radial confinement, the fields can be shaped in a way that allows axial transport of ions in a trap. The trap used in this thesis was built to trap both Be-9+ and Ca-40+ ions. Different species are used to reduce cross-talk between neighbours in the ion-chain if several ions are trapped with alternate species next to each other. The level diagrams including the chosen qubit transitions are depicted in Fig. 1.2.

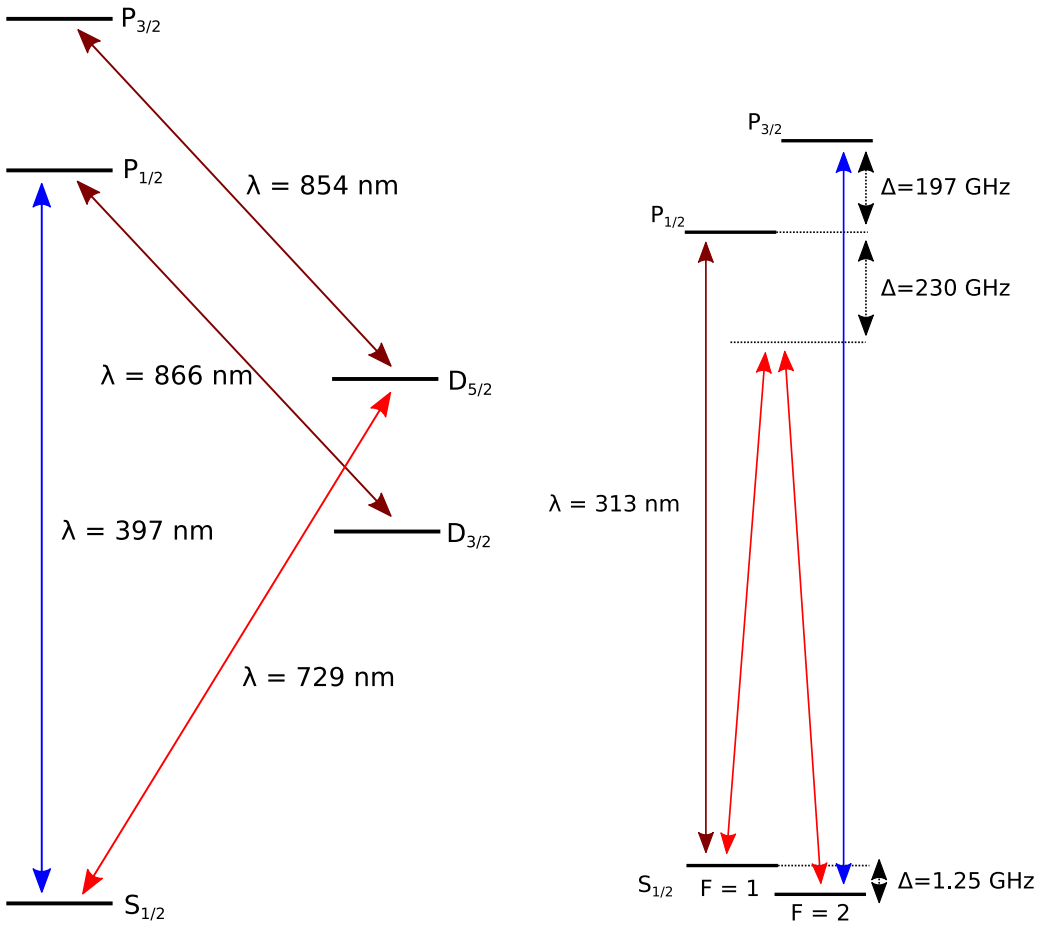
1.4 Laser control

The advanced requirements for trapped-ion experiments necessitate elaborate stabilization techniques. The narrow linewidths obligate physicists to employ several frequency stabilization techniques. Due to temperature instabilities and vibrations, amplitude fluctuations arise. It's the main goal of the thesis to improve the control systems to counteract effects that lead to amplitude instabilities.

A separate set of lasers is required to address each species of ions. A detailed description of the beam paths can be found in [10], [11] and [12]. There are several stages of frequency and amplitude stabilisation to fulfill the requirements of meaningful qubit manipulation. During each stage the feedback signal has to interact with the laser light. A versatile device in this regard is discussed in the following subsection.

1.4.1 Acousto optic modulators

The component that allows frequency, phase and amplitude control of laser beams is an electrically controlled device called an acousto optic modulator (AOM). It consists of a piezo crystal attached to a transparent cell which allows sound waves to be fed into the cell by applying RF



(a) Ca-40+ energy levels

(b) Be-9+ energy levels

Figure 1.2: Energy level structure of the used ions. Cooling and read-out transitions are coloured dark blue. Repumping transitions are dark red and the qubit transitions are red. Dashed arrows are used to draw in frequency differences. Adapted from [10, pp. 10]

voltage to the piezo crystal. The medium of the cell exhibits photo-elasticity which means the refractive index changes if the cell is deformed mechanically. In the special case of sound waves traveling through the cell, the internal strains cause the refractive index to vary periodically in space and time. Therefore, the sound wave leads to a moving grating which causes interference patterns on the other side if light is send through the medium.

There are two distinct regimes of diffraction of light by the acoustic waves, which are distinguished by the Klein-Cook parameter

$$Q = \frac{2\pi\,\lambda\,D}{\Lambda^2}$$

where λ is the optical wavelength, Λ the acoustic wavelength and D the width of the acoustic wave i.e. the length of the interaction region. Raman-Nath diffraction occurs at lower acoustic frequencies and shorter interaction lengths ($Q \ll 1$). In this regime the reflection off the acoustic waves can be neglected and incident plane waves of light appear as a corrugated wave front on the other side. Solving the diffraction integral [14] for a point on a distant screen gives

a profound explanation of the resulting intensity distribution, consisting of many interference maxima. However, the AOMs commonly used for laser control are driven in the Bragg regime ($Q \gg 1$). In this case, the occurring interference pattern is caused by the reflected part of the optical wave. This topic has been dealt with in detail in several textbooks [15], [16]. Mostly, the explicit inclusion of the phase of the acoustic wave is omitted, which is an essential part for understanding the abilities of AOMs.

A short, classical derivation based on a photonics textbook [15] shall be given:

In Fig. 1.3 the chosen coordinate system is shown. The optical and the acoustic waves are assumed to be planar. Light is incident at angle θ to the plane of the acoustic aves.

The strain within the medium is proportional to the change of the refractive index as long as the amplitude is not exceedingly high. Therefore a sinusoidal voltage applied to the piezo crystal leads to a sinusoidal traveling wave in the optic medium. The refractive index can be described as its undisturbed value, n_0, minus a cosine term with a certain amplitude, Δn which is proportional to the voltage. The minus sign is due to the refractive index getting lower when the strain within the medium rises. Assuming the acoustic wave to have an angular frequency ω_a and a wave number of k_a, this yields:

$$n(x,t) = n_0 - \Delta n \cos(k_a x - \omega_a t - \phi)$$

The gradient of the refractive index causes reflections. Since the optical frequency is much higher than the acoustic frequency the refractive index function $n(x,t)$ can be treated as constant in time with a fixed phase offset $\Phi = \omega_a t + \phi$.

$$n(x) = n_0 - \Delta n_0 \cos(k_a x - \Phi) \tag{1.4}$$

The total reflectance of the cell and therefore the amplitude and phase of the reflected beam can be found by interfering the waves reflected on two planes next to each other. Assuming that the reflectance on these planes is small enough, reflected waves can be considered to have the same amplitude. Or in other words, the incoming laser beam is not significantly attenuated at the end of the interaction zone. While the path difference caused by the change of refractive index can be ignored, the geometrical difference can't be neglected. In Fig. 1.3, this is visualised.

Taking the occurring phase difference into account, the total reflectance is given by the integral in eq. (1.5), with k_o being the wave number of the optical plane wave. The obvious integral over $\mathrm{d}r$ is substituted by an integral over the size of the optical beam.

$$r_{total} = \int_{-L/2}^{L/2} e^{i2k_o \sin(\theta)x \frac{\mathrm{d}r}{\mathrm{d}x}} \,\mathrm{d}x \tag{1.5}$$

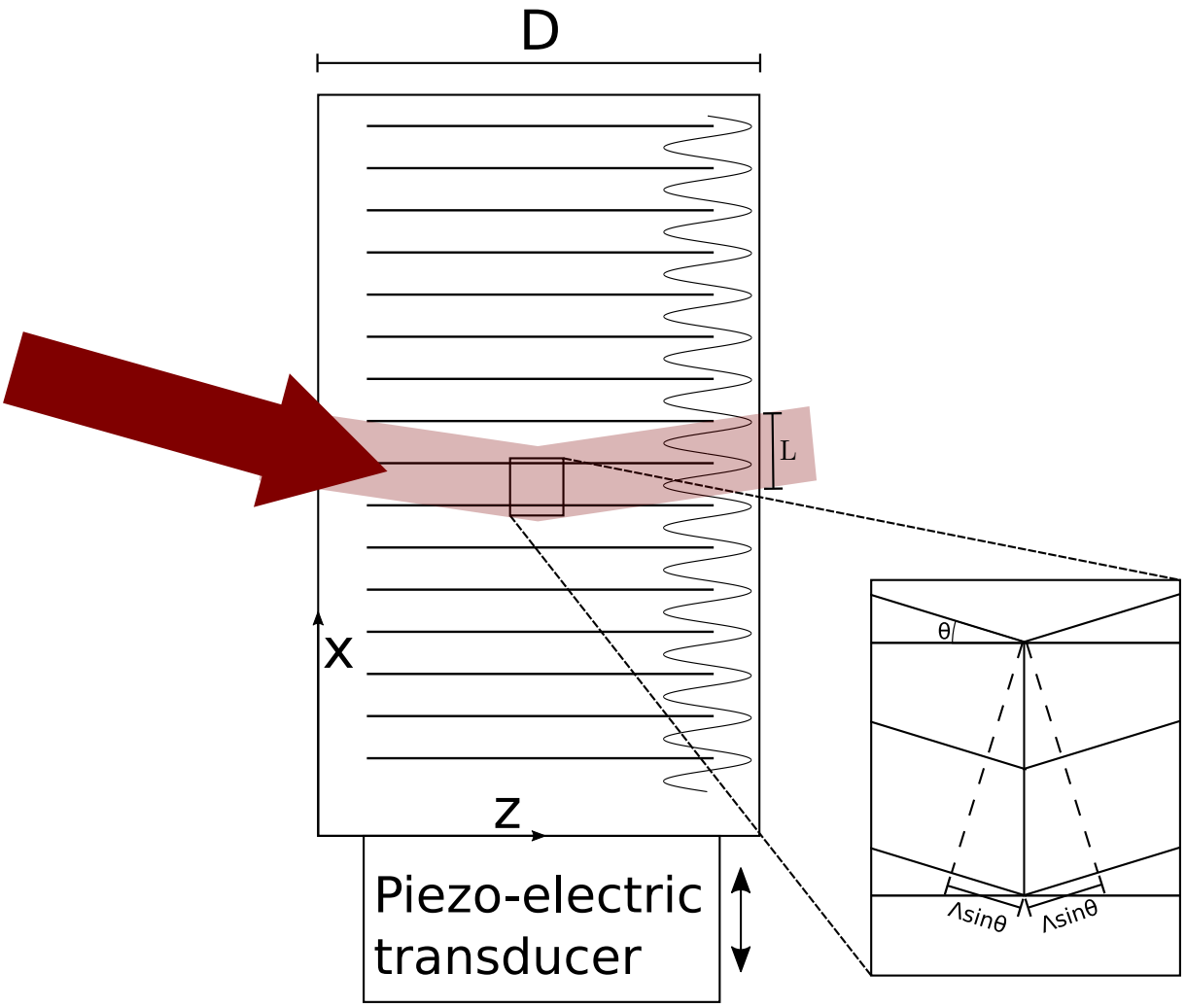


Figure 1.3: Schematic drawing of light and acoustic wave interaction in an AOM in the Bragg regime

The reflectance gradient can be derived from Fresnel equations.

If two adjacent planes orthogonal to the x-axis with a distance of Δx are considered, the refractive index changes by Δn. Extending the planes until they have a virtual boundary between them leads to the situation drawn in Fig. 1.4. The reflectance that is caused by the different refractive indices can be calculated using Fresnel equations. They are different for the polarization orthogonal and parallel to the plane of incidence (the plane defined by the wavevector of incident light and of the acoustic wave). In the orthogonal case, the reflectance would be:

$$r_s = \frac{(n-\Delta n)\cos(90-\theta) - n\sqrt{1-(\frac{n-\Delta n}{n}\sin(90-\theta))^2}}{(n-\Delta n)\cos(90-\theta) + n\sqrt{1-(\frac{n-\Delta n}{n}\sin(90-\theta))^2}} \tag{1.6}$$

The variables usually used in the Fresnel equation are drawn in Fig. 1.4. Compared to the standard variables, in eq. (1.6) the incident angle θ_i is substituted with $90-\theta$, the refraction

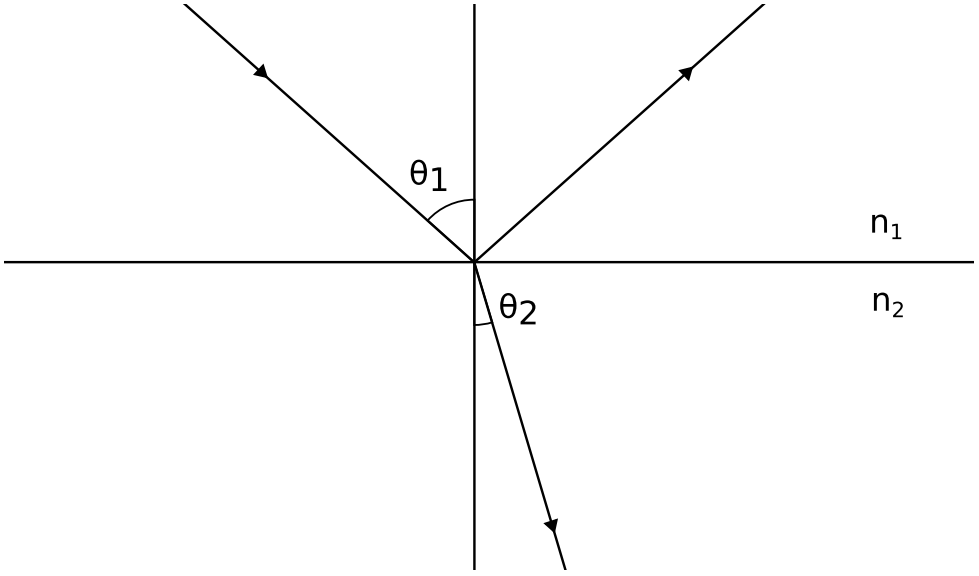


Figure 1.4: Incident, reflected and transmitted beam at a boundary of two media with different index of refraction

angle θ_t according to Snell's law $n_i \sin(\theta_i) = n_t \sin(\theta_t)$ and the refractive indices are $n_1 = n + \Delta n$ and $n_2 = n$.

The equation for parallel polarized light is the same as eq. (1.6) but n_1 and n_2 have to be exchanged. The amplitude reflectance is complex in general as it determines the phase change and the amplitude of the reflected wave. Even if the refractive index is real at all times (no absorption) the term below the square root can be negative.

It's sufficient to use the first order Taylor expansion to arrive at a useful expression of the reflectance dependent on the refractive index change because it is small.

$$\Delta r_s = \frac{-1}{2n \sin^2(\theta)} \Delta n \tag{1.7}$$

$$\Delta r_p = \frac{-\cos(2\theta)}{2n \sin^2(\theta)} \Delta n \tag{1.8}$$

Usually θ is very small and $\cos(2\theta) \approx 1$ which means that light polarized parallel and orthogonal to the plane of incidence can be treated the same (as r_s).

The derivative in eq. (1.5) can now easily be expressed:

$$\frac{\mathrm{d}r}{\mathrm{d}x} = \frac{\mathrm{d}r}{\mathrm{d}n} \frac{\mathrm{d}n}{\mathrm{d}x} = \frac{-1}{2n \sin^2(\theta)} \Delta n_0 \, k_a \sin(k_a x - \Phi) \tag{1.9}$$

Combining eqs. (1.5), (1.7) and the first derivative of eq. (1.4) leads to an integral that can be solved straightforwardly if the sine in the first derivative of $n(x)$ is replaced by exponential representation.

$$r_{total} = \tfrac{1}{2} i r' \left(\int_{-L/2}^{L/2} e^{i(2k_o \sin(\theta)+k_a)x} e^{-i\Phi} - e^{(2k_o \sin(\theta)-k_a)x} e^{i\Phi} \, \mathrm{d}x \right) \tag{1.10}$$

$$r' = \frac{-\Delta n_0 \, k_a}{2n \sin^2(\theta)}$$

There is one maximum for each term in eq. (1.10). If $2k_o \sin(\theta) = \pm k_a$ the other term will become negligible. Solving the integral for the upshifted term $2k_o \sin(\theta) = k_a$ yields:

$$r = \tfrac{1}{2} i r' L \, \mathrm{sinc}\left[(k_a - 2k_o \sin(\theta))L/2\right] e^{i\Phi} \tag{1.11}$$

In eq. (1.11) sinc is defined as $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$. It reaches its maximum when the well-known Bragg condition $n\lambda = 2\Lambda \sin(\theta)$ is true for the first order $n = 1$. The tolerance of the Bragg condition can be determined by using $\mathrm{sinc}\left[(k_a - 2k_o \sin(\theta))L/2\right] = \mathrm{sinc}\left[(\sin(\theta) - \sin(\theta_B))2L\pi/\lambda\right]$ with θ being the actual angle and θ_B the Bragg angle. The sinc function vanishes at $\sin(\theta) - \sin(\theta_B) = \lambda/(2L\pi)$. These angles are usually small so $\sin(\alpha) = \alpha$. In general, the laser beam width is much greater than the wavelength and therefore the tolerance is small.

To examine the frequency and phase behaviour of the reflected wave consider a plane wave multiplied by the complex amplitude reflectance. Considering the frequency part, $e^{i\Phi} = e^{i(\omega_a t+\phi)}$, the optical frequency will be shifted upwards by the acoustic frequency. Additionally, the phase can directly be controlled by the phase of the piezo crystal signal. Looking at r', the amplitude of the reflected beam is proportional to the amplitude of the acoustic wave which in turn is proportional to the input voltage amplitude of the piezo crystal. As the assumption that the reflected wave doesn't deplete the incoming wave significantly breaks down at higher refraction index gradients, eq. (1.11) is only true for small acoustic wave intensities. In case of saturation, the total reflectance reads

$$R = \sin^2(\sqrt{R_b}) \tag{1.12}$$

with $R_b = |r|^2$ at the Bragg angle. For small reflectances R becomes equal to R_b.

There is also a quantum interpretation of the described phenomena. Every refracted photon has to absorb a phonon. Energy and momentum conservation leads to:

$$\hbar\omega_r = \hbar\omega_o + \hbar\omega_a$$
$$\vec{k}_r = \vec{k}_o + \vec{k}_a$$

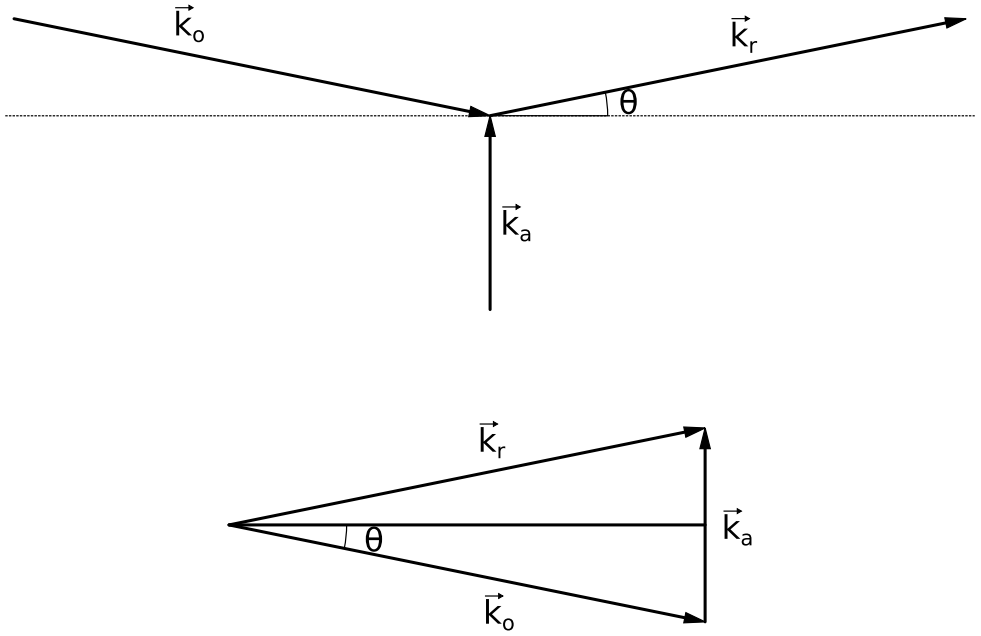


Figure 1.5: Wave vectors of incident k_o, reflected k_r and acoustic k_a wave.

The frequency increase is obvious. The Bragg condition $2k_o \sin(\theta) = nk_a$ can be derived directly from the lower schematic in Fig 1.5.

While these steps are much simpler in this picture, it is harder to explain why only one maximum can be observed in the Bragg regime. Double absorption of phonons does not produce a visible effect in this case. The phase shift of the reflected beam is not obvious either if only the particle nature of phonons and photons is taken into account.

As no additional assumptions were made in this quantum interpretation, the same holds in the Raman-Nath regime [14]. However, more maxima with a non trivial distribution of intensity can be observed. The phase shift is the same as in the Bragg regime.

The distance between the interaction zone and the piezo crystal causes a delay between the RF signal applied on the piezo crystal and the optical output as the acoustic wave first has to travel the distance to produce any measurable effect. In the segmented ion trap setup the addressed AOMs have a delay around 1 µs. There is also a rise time due to the non zero diameter of the interaction zone.

1.4.2 Amplitude instabilities

Although the intensities of the lasers driving the qubit transitions are stabilised at intermediate points in the beam path, the beam intensity after passing through the trap and illuminating the ions is not used as a feedback signal. For both ion species, an ion is used directly to calibrate the amplitude and the frequency of the final AOMs which are used to switch the qubit driving fields

on and off. This is a manual process that requires intervention of an experimenter. Although the setup is temperature stabilised, the AOMs responses drift over time and thus cause infidelities during an experimental sequence.

The effects of amplitude fluctuations are described in the following paragraphs.

As mentioned in subsection 1.2.1, the amplitude of the laser beam is directly proportional to the Rabi frequency. The achieved angle of rotation of a pulse is determined by its area. Assuming that the amplitude fluctuations are small and have a lower frequency than the linewidth of the qubit transition, an effective Rabi frequency which is the mean over the gate time may be used.

In the case of Rabi flops, a fluctuating Rabi frequency leads to decreased contrast of the Rabi oscillations. The population of the ground state $|0\rangle$ evolves as $\cos(\Omega t/2)^2$ which can be easily derived from eq. (1.1) using the angle retrieved from eq. (1.2). The amplitude of the function doesn't change but many measurements have to be conducted to deduce the final state. The percentage of measuring a bright ion at the end of an experimental sequence is equal to the actual population if the measured state is always the same. However, a changing Rabi frequency causes a different state to be measured every time and thus randomises the outcome. The deduced population approaches 0.5 as the variance of the Rabi frequency increases.

Assuming the intensity fluctuations follow a normal distribution with a relative standard deviation of σ_{Ω_0} when averaging over time t, the population evolution with time can still be solved analytically (eq. (1.13)). It equals an offset cosine attenuated by a gaussian.

$$\begin{aligned} P_0(t, \sigma_{\Omega_0}) &= 1/\sqrt{2\pi(\Omega_0\,\sigma_{\Omega_0})^2} \int_{-\infty}^{\infty} \cos(\Omega t/2)^2 \; e^{-\frac{1}{2}\left(\frac{\Omega-\Omega_0}{\Omega_0\,\sigma_{\Omega_0}}\right)^2} \mathrm{d}\Omega \\ &= \frac{1}{2}\left(1 + \cos(\Omega_0 t)\; e^{-\frac{1}{2}(\Omega_0\, t\, \sigma_{\Omega_0})^2}\right) \end{aligned} \tag{1.13}$$

This can be used to approximate the success probability of randomised benchmarking suffering from amplitude instabilities. In practice, RB is more robust than simple Rabi flopping which corresponds to applying only x rotations. Z rotations are not influenced by the laser intensity and there is a probability of $\frac{2}{3}$ of applying a rotation which would leave the state unchanged in the ideal case. A non-unity success probability can be visualised by a cycle around the target vector on the Bloch sphere. In the case of RB the target vector always coincides with one of the axes of the coordinate system. A rotation around this axis does not change the distance to the target vector and thus the success probability. However, in the case of a composite R_x and R_y rotation, only one may have no effect on the success probability.

The partial independence on the intensity noise could be approximated by a mixed distribution of a normal distribution with a probability of P_g and a delta distribution with probability $1 - P_g$

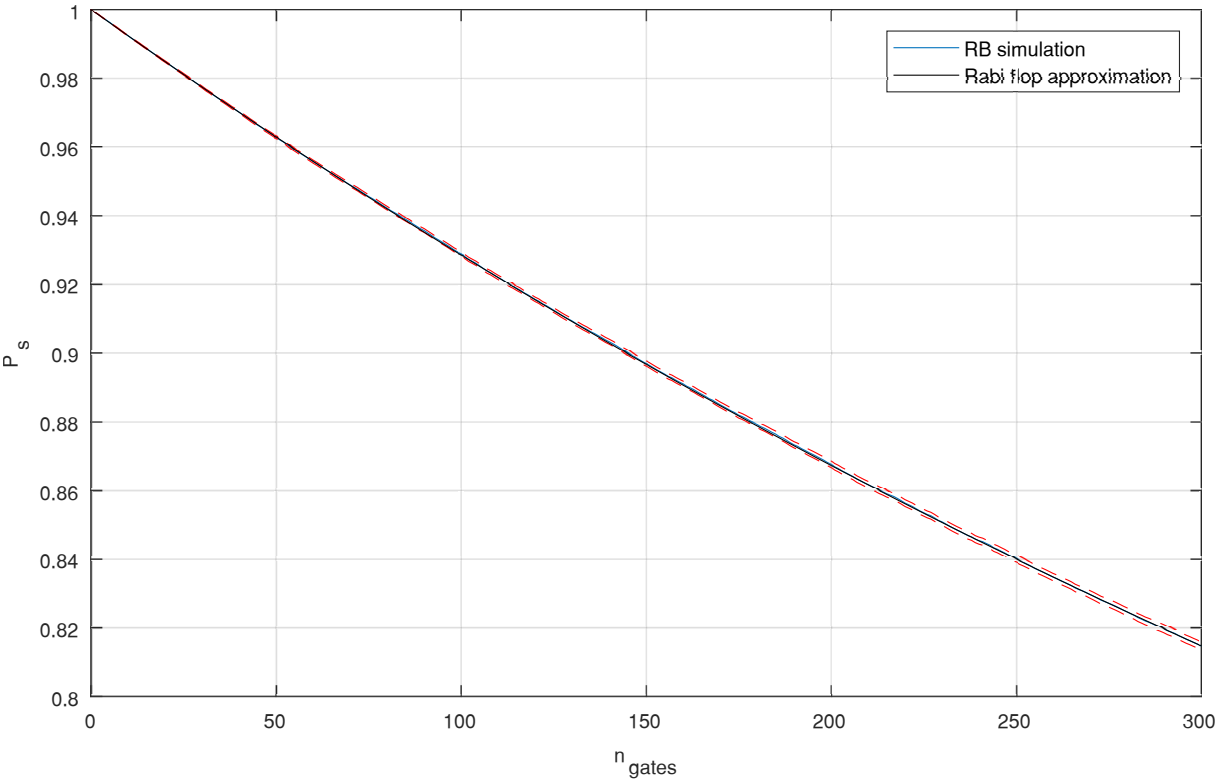


Figure 1.6: Success probability depending on the number of applied gates. The relative standard deviation σ_{Ω_0} of the Rabi frequency averaged over the time of a π-gate is 2.5 %. The Rabi flop approximation uses a reduced standard deviation of $\sqrt{\frac{1}{2}}\,\sigma_{\Omega_0}$. The 95 % confidence interval bounds of the RB simulation are indicated. The simulation uses 800 unique gate sequences.

for every RB gate. If averaged over many samples, which is the case for a high number of gates n and gate sequences, the Rabi frequency behaves normally distributed with a variance of $\sigma^2\ P_g/n$. σ^2 is the variance of the Rabi frequency averaged over the duration of a π rotation which is the average time of an RB gate. The assumptions in the previous paragraph suggest a probability of $P_g = \frac{20}{24}\frac{2}{3}$. Comparing with simulations of the RB implementation described in Sec. 1.2.1 lead to a probability of $P_g \approx \frac{18}{24}\frac{2}{3} = 0.5$ which indicates additional robustness against intensity noise. Alternatively, the same result would be achieved if RB gates were approximated by $\pi/2$ instead of π rotations.

With this result, the success probability evolution depending on the number of gates can be approximated within the 95 % confidence bounds. A demonstration can be found in Fig. 1.6.

Fitting the success probability yields a intensity noise independence probability of 0.497 ± 0.007. The uncertainty is retrieved by fitting the 95 % confidence interval bounds. The dependence of the success probability on the standard deviation using this value is plotted in Fig. 1.7.

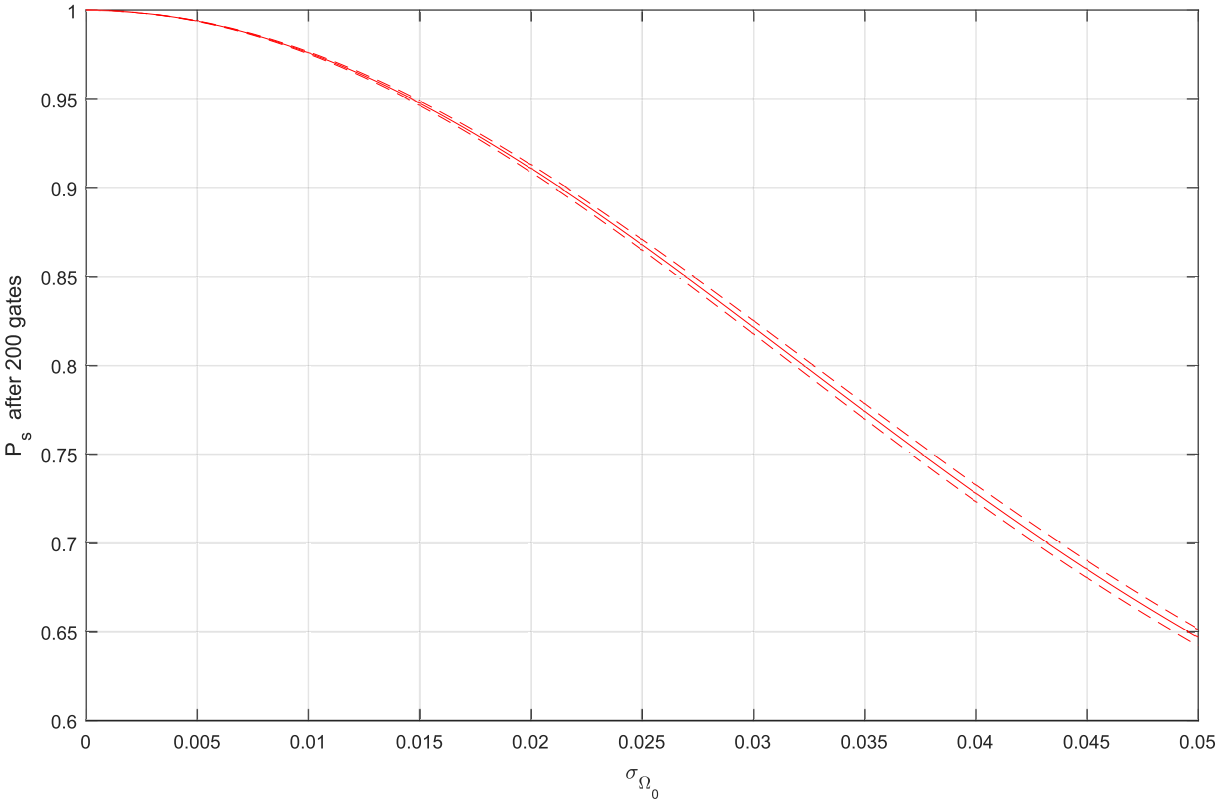


Figure 1.7: Success probability after 200 gates depending on the relative standard deviation of the Rabi frequency averaged over the time of a π-gate using the approximation discussed in the text.

1.4.2.1 Stabilisation

When stabilising the amplitude after a switching AOM, the feedback loop has to be carefully designed. The response delay of the system will be at the order of hundreds of nanoseconds mainly due to the travel times of the acoustic wave in the AOM medium. In general, this limits the bandwidth and stability of the feedback loop. A pulsed feedback signal with sharp rising and falling edges contains a considerable amount of high frequency components, which can't be tracked because of the delay. Additionally, there is no meaningful feedback signal when the RF amplitude is zero. Therefore, the feedback loop has to be paused before falling edges and continued after rising edges. Unwanted jumps in the feedback signal can be avoided by only using the feedback loop between the intensity flanks measured by the photodiode.
In general, the setpoint of the feedback loop changes when pulses with different intensities are required. Thus, the setpoint has to be adjustable similar to the required pulse intensities.
Standalone PID loops are not suitable for applications in this regime. However, if the feedback loop is integrated into the control system, it has additional information about expected changes in the signal and can adapt accordingly. Two different approaches were implemented over the course of this thesis.

Usually, the feedback signal from e.g. a photodiode is proportional to the intensity of the laser beam. The total reflectance mentioned in the previous section is proportional to the square of sine of the RF amplitude fed into the piezo crystal, which can directly be changed in the feedback loop. To get rid of the direction dependency of the first order diffracted beam on the frequency, often a double passed AOM setup is used [17]. In this case, the finally used laser beam is shifted by twice the frequency and its amplitude is dependent on the square of the single pass reflectance. This results in the intensity being approximately proportional to the fourth power of the sine of the RF amplitude. The non-linear behaviour additionally degrades the bandwidth of a feedback loop for higher dynamic ranges.

Chapter 2

Controlling the experiment

In order to conduct experiments on a larger scale reliably and reproducibly, as many experimental parameters as possible have to be controlled precisely. Especially when noise has to be averaged out by taking the mean over hundreds of runs, the control system should be capable of capturing data and tuning all relevant parameters automatically. In trapped ion quantum information processing, a multitude of different devices have to be operated coherently.

The requirements are diverse ranging from TTL (on and off) signals for e.g. shutters to high frequency (up to 400 MHz) RF signals for driving the AOMs needed to introduce frequency, phase and amplitude shifts to the laser light that interacts with the trapped ion.

Even if almost all control signals are supplied electrically, different requirements lead to different devices being used and thus it is easy to end up with an improvised patchwork of instruments which need to be controlled together.

Table 2.1: Approximate requirements for synchronously controlled devices in the mixed species ion trap setup (adapted from [12])

Type	Requirements	Timing	quantity
rf output	100 kHz - 500 MHz frequency, phase-coherent	<20 ns resolution, <1 ns jitter	16
analog output	tens of volts, >10 MHz bandwith	<100 ns jitter	32
digital output	LVTTL (0 - 3.3 V)	<100 ns jitter	32
analog input	<1 V, high stability	<100 ns jitter	4 - 8
digital input	LVTTL (0 - 3.3 V)	<20 ns resolution, <1 ns jitter, <50 µs latency	5

The TIQI group already developed a system consisting of a multitude of components that work together in harmony. It was mainly developed by Vlad Negnevitsky and is described extensively

in his PhD thesis [12]. RF outputs are primarily used to supply the inputs to amplifiers driving the AOMs. Analog outputs control the voltages on the DC electrodes in the trap to allow operations such as transport between different areas and ion crystal separation. Digital outputs are used as trigger signals or to control shutters. PMTs (Photo Multiplier Tubes) are connected to digital inputs which count rising edges which signify detected photons. Analog inputs can be used for feedback and active stabilisation of amplitudes.

The system carries the name Modular Advanced Control of Trapped IONs (M-ACTION) and was carefully designed with scalability in mind. It consists of the necessary hardware and software to perform real-time control of experiments.

There are various standalone devices which are not controlled by M-ACTION, like Pound-Drever-Hall (PDH) locks, the lasers themselves, magnetic field stabilisation and so on. However, they might have a TTL connection to the control system to raise a flag if any error occurs and the device should be checked.

2.1 FPGAs

A device frequently used in this area because of its flexibilty is the FPGA (Field Programmable Gate Array).

The average physicist, having done a fair share of scripting and programming, knows how to understand sequential code. In general just one path through the code is executed, jumping around if necessary. Parallelisation can be done explicitly but essentially results in a handful of threads being handled side by side. An FPGA is entirely different. Instead of many instructions done sequentially everything is done at the same time because the "program" described, which is actually a circuit, is implemented directly in hardware. This means that a simple addition is mapped to flip-flops and gates during compilation which is called synthesis for FPGAs. The synthesized RTL (Register Transfer Level) code is mapped and routed to physical parts on the FPGA chip in use and then written to a bit-file that needs to be flashed onto the chip to realize the designed program physically in hardware.

There are two languages specifically designed to describe such hardware, Verilog (VERIfication LOGic) and VHDL (Very High-speed-integrated-circuit Description Language). While VHDL allows more descriptive code that is considered to be self-documenting, Verilog is closer to the hardware and needs fewer lines to implement the same functionality. However, it is more prone to confusing code and definitely needs documentation. The TIQI groupexclusively uses Verilog for their projects.

A directly implemented circuit allows low latency data processing and coherent execution of complicated instructions. It also adds additional flexibility to PCBs as various peripherals can be connected to the board and be "wired" arbitrarily without having to change the fixed physical layout of the board.

A Verilog program usually consists of several modules, which are connected to each other. If this "program" is intended to be used in a larger scale project it is called an Intellectual Property (IP) core.

The "program" running on an FPGA is usually called firmware or gateware. It's helpful to keep in mind, that Verilog programs describe circuits.

2.1.1 Timing

On the FPGA, synchronous i.e. clocked signals need to travel from flip-flop to flip-flop within one clock-cycle. This limits the amount of logic that can be used between the two flip-flops and the maximum distance between them. Additionally, timing constraints can be manually set to define limits for signals exiting and entering the FPGA. During implementation the synthesis software tries to meet all constraints by placing and routing logic in a clever way. Often, significant time during FPGA development is spent writing a module which can meet all timing criteria.

2.2 Control system

The M-ACTION system could be described as a hybrid between FPGA and CPU-centric [12] architecture, which means that all time-coherent actions are controlled by one CPU with a common instruction set running the main control application. A standard PC runs the GUI which allows configuration of the control system. The main CPU does not have to take care of any visualisation and can focus on coordinating subsystems. The main application does not need an operating system and can be run bare-metal which means it runs in real-time and the application has full control over the processor. This allows low latency feedback while maintaining the advantages of not having to use specialised hardware which requires lower level design. This main application goes under the name `ionpulse SDK` and is written in modern C++, which makes it relatively easy to add new experimental sequences. It is run on a Zedboard[1].

A capable GUI called `Ionizer2` runs on a control PC and connects to the Zedboard via Ethernet and allows the user to configure experiments and plot results conveniently. `Ionizer2` offers access to several asynchronously controlled devices as well.

[1] Commercial Zynq evaluation board by AVNET using a Xilinx XC7Z020-CLG484-1 chip[18]

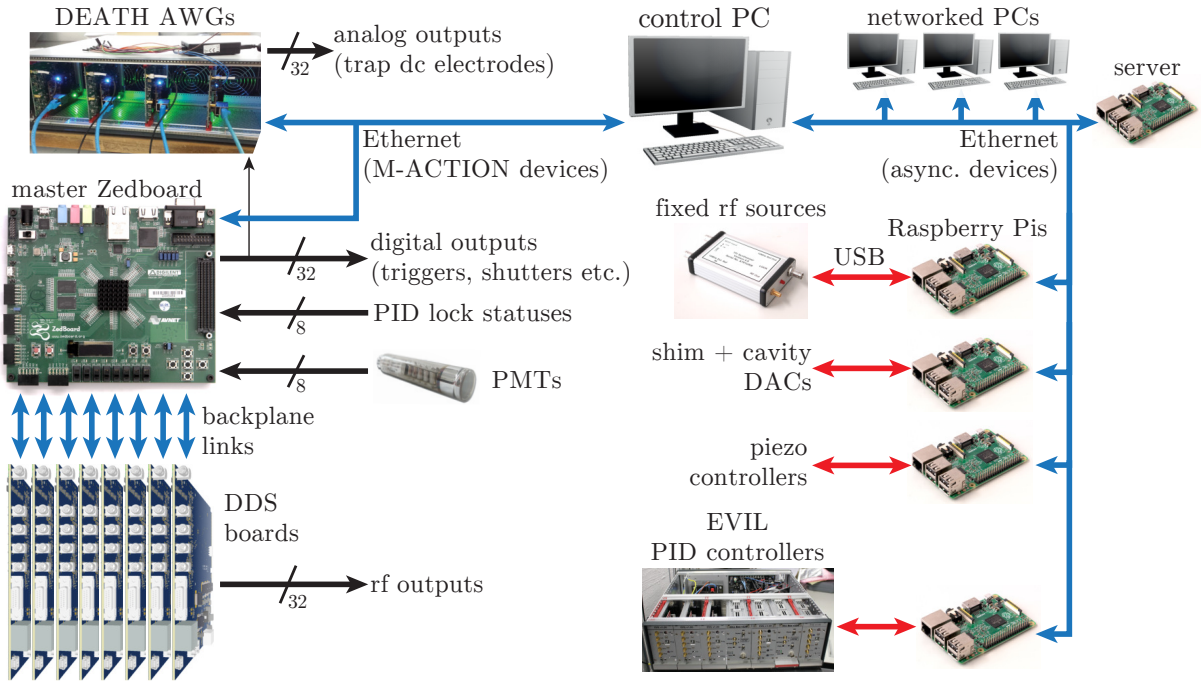


Figure 2.1: Control system used for the mixed-species ion trap setup. The network which connects the M-ACTION devices is seperate from the network for asynchronous devices, which is managed using a Raspberry Pi as the gateway server. The figure is from [12, pp.32].

The Arbitrary Waveform Generators (AWGs), called DEATHs (Direct Ethernet-Adjustable Transport Hardware), which handle the DC electrodes in the trap, receive the waveforms themselves directly from the control PC but the Zedboard decides which one they should play during an experiment and also triggers them. This allows the main application to change the currently running waveform on the DEATHs during an experiment. Each DEATH contains a smaller version of the Zedboard which controls 8 DAC channels. Direct Digital Synthesis (DDS) cards handle RF outputs. They are connected to the Zedboard via a backplane connection, which means that there are direct connections between the Zedboard and the cards that allow low latency fast communication in both ways. The DDS cards[19] have an FPGA[2] which implements the control firmware for the DDS chips themselves.

An overview of the devices used in the laboratory is shown in Fig. 2.1.

One of the main strengths of this architecture is the ability to offload low level instructions to remote system components. The main application doesn't have to take care of actual RF signals being created or waveforms being processed. It tells the relevant device what to do which in turn executes received commands. The remote devices have to be fairly complex but remain static in their functionality which makes faster experiment development possible as long as no additional new features are requested. The `ionpulse SDK` is not directly involved in actually running a time-coherent sequence. This is handled by the subsystems which receive the required

[2]Xilinx Spartan 6 XC6SLX150T

instructions ahead of time. In the case of readout feedback (e.g. in [20]) the `ionpulse SDK` sends an update dependent on the readout to the subsystems.

The `ionpulse SDK` is a powerful but in some parts quite obfuscated program as documentation is hard to come by and a big part of its functionality is hidden in the subsystems connected to the Zedboard.

2.2.1 Zedboard

The heart of the M-ACTION system, the Zedboard, carries a Zynq chip which is the most central part of the board. It consists of a dual-core ARM processor on the same chip as an FPGA. The FPGA side of the Zynq chip is called Programmable Logic (PL) while the CPU side is called Processing System (PS).

Only a single core in the PS is used to run the main application, `ionpulse SDK`, which communicates with other devices via Ethernet or with the PL via AXI (See Sec. A.1). USB connections are used to program the Zedboard and to establish a serial connection to the main application for debugging.

Several IP cores are implemented in the PL. An overview over the original functionality of the IP cores is given. Any changes are mentioned in the upcoming chapters. A schematic diagram of the cores working together is shown in Fig. 2.2.

2.2.1.1 Pulser

Digital out- and inputs on the Zedboard are handled by `pulser`. The digital outputs are handled through a First-In First-Out pipe (FIFO) that takes 64-bit instructions which get executed by a sequencer. This instruction includes a 32-bit mask which is fed to the digital outputs of the Zedboard and a 16-bit time which tells the sequencer how many clock cycles to wait until the next instruction is processed. Additionally, 8-bits are used to gate the counting modules for the PMTs. As long as the gating line is high PMT pulses are counted. The result is stored as soon as the gate line goes low. The results can then be read out via AXI. Finally, the last 8-bits of the main instruction word are used for additional flags with some of them being empty for future use.

The driver of `pulser` is simple and uses C instead of C++, like the proprietary drivers supplied by Xilinx. Multiple interface functions translate humanly readable input parameters into instruction words and allow readout of PMT data for example. Additionally, there is an elaborate selftest that checks if the firmware behaves as expected.

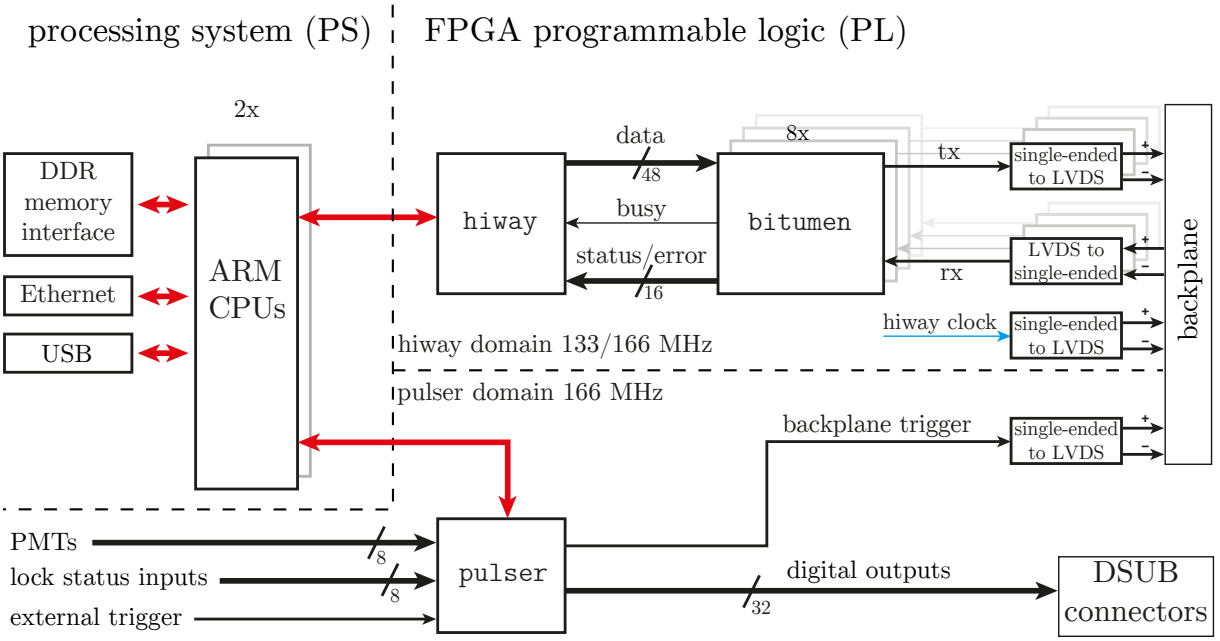


Figure 2.2: Schematic of the firmware used on the Zedboard. The red lines are AXI busses, the blue line distinguishes a clock. `hiway` handles interfacing the PS with `bitumen`, which serialises parallel data and transmits it via the backplane to the DDS cards. `pulser` takes care of remaining time-coherent actions as PMT read-in or digital outputs. The figure is adapted from [12, pp.35].

2.2.1.2 Hiway

`hiway` is used to manage the interface between the PS and up to 8 DDS cards. Often, the same instruction has to be sent to all 8 cards, which would be inefficient if every card just had its own AXI address. `hiway` enables the PS to do this. Amongst others, it has an MSB and LSB register. Writing to the LSB register will trigger a write operation to the DDS cards. The lower 8 bits of the upper 16 bits are the mask which defines the cards the lower 48 bits are sent to. `hiway` also logs all error bits that the communication IP core to the DDS card might raise.

The driver of `hiway` is much more complicated than the `pulser` driver as it is needs to generate instructions for the sophisticated DDS cards.

2.2.1.3 Bitumen

The module responsible for serialisation and deserialisation (serdes) of the in- and outgoing data is called `bitumen`.

Originally, communication was mainly transferred from the Zedboard to the DDS cards with 8 error bits being the only data response sent back to the Zedboard.

The main module on the Zedboard side, `bitumen`, takes a 48-bit instruction word and slices it into 12-bit chunks that gets fed into the FIFO of module `bit_tx`, which serialises the chunk and frames it in a bit sequence with a structure similar to Universal Asynchronous Receiver Transmitter (UART).

The counterpart on the DDS cards, `bitumen_slave` puts together the chunks deserialised by `bit_rx` and sends back data which allows the Zedboard to detect errors in transmission. Additionally, one chunk containing the aforementioned 8 error bits is sent back at the end of a transmission.

2.2.2 DDS cards

Another central piece of the M-ACTION system are the DDS cards. They consist of a Xilinx Spartan 6 FPGA and 4 Analog Devices AD9910 DDS chips. They are connected to the Zedboard via the backplane through Low-Voltage Differential Signaling (LVDS) pairs (one for sending and another one for receiving).

A DDS can be seen as a DAC where the input can only be changed in a limited way. It always puts out a sine (or cosine) with a configurable frequency, amplitude and phase. There is an accumulator that is incremented by the frequency tuning word (FTW) every cycle. When it overflows it just continues from zero. The Phase Offset Word (POW) is added after the accumulator. The resulting phase value is mapped to an amplitude through a Look-Up Table (LUT) which implements a sine (or cosine). Afterwards the amplitude is scaled by the Amplitude Scaling Factor (ASF). This results in the input value to the DAC. On the DDS cards, the DDS output is further filtered and passed on to a Variable Gain Amplifier (VGA) which allows the amplitude to be scaled as well. The gain factor is set by an additional DAC which is independent from the DDS.

The DDS chips are connected to the FPGA via Serial Peripheral Interface (SPI) which supports speeds of at most 70 MBPS but is . The FTW and the ASF are written into registers on the DDS chip via SPI and are applied when the IOUPDATE input line of the chip is pulsed. Thus, the DDS state can be changed at a precise point in time. The phase is set via a fast parallel connection. In the end, the phase defines the rotation axis of the qubit and has to be coherent during qubit manipulation. The firmware on the DDS cards keeps track of the global phase and adds a phase offset temporarily if needed. The phase of the pulse which is about to be applied is calculated and sent to the DDS chips over the parallel bus.

The Zedboard has two extra lines which are connected to all DDS cards. One is used for clock distribution and the other is used for coherent triggering. Each of these lines uses a clock buffer which takes care of the path delays between the DDS cards to makes sure they run synchronously.

The backplane clock is used for backplane communication as well which makes it synchronous. The clock frequency was recently increased from 133 MHz to 166 MHz by Chagri Önal. There is another clock domain on the DDS card which is driven by the DDS chips instead of the backplane. The core of the DDS chip runs at a high frequency of 1 GHz which is supplied from a highly stable external clock source. The DDS chips deliver a slower 125 MHz clock which is used to clock the second domain of the DDS card.

The firmware, called `minotaur`, consists of 5 major modules which handle different aspects of the DDS cards. Instruction words deserialised by `bitumen` are fed into `odecoder` which executes the outer layer of the instruction.

The other 4 modules are instantiated 4 times in the design, once for each DDS channel. Every one of the modules uses at least one Block RAM (BRAM) on the FPGA to store information needed to perform its task. The BRAMs are filled by `odecoder` according to the instructions it receives from the Zedboard.

`idecoder` is the main module controlling the flow through the experimental sequence for its channel. It tells the other modules which data to use from the BRAMs and handles branching and looping as needed (explained in Sec. 2.3). `mdds` reads in parameters for the DDS chip and sends them via SPI and parallel bus. `mvga` handles the external DAC of the VGA which is used to create shaped pulses. It has a LUT (implemented in a BRAM) which defines a shape that minimises the spectral width of the produced RF pulse. `mtimer` is responsible to coordinate all modules. Primarily, it triggers the other modules when an edge is completed and supplies `mdds` with the current time which is needed to calculate the RF phase for a pulse.

2.3 Operation

Experimental pulse sequences with varying complexity are defined before compilation of the `ionpulse SDK`. The top level sequence may contain several looped subsequences or forked paths. The driver of `hiway` takes care of translating the constructed sequence into instructions for `minotaur`. It keeps track of all BRAM addresses and detects any out of memory issues. In the case of looped subsequences, the BRAM data might not be written immediately to the BRAMs but stored in the FIFOs of `odecoder`. When the loop completes, `odecoder` is notified and writes the required number of cached data sets to the respective BRAMs.

The simple design of `pulser` is inconsistent with the programming structure of the DDS cards as it cannot perform any loops or forks but just processes the input FIFO linearly. This necessitates mirroring the behaviour of `minotaur` in the `ionpulse SDK`. Essentially, every element in the experimental sequence is wrapped with an additional class containing the information for `pulser`.

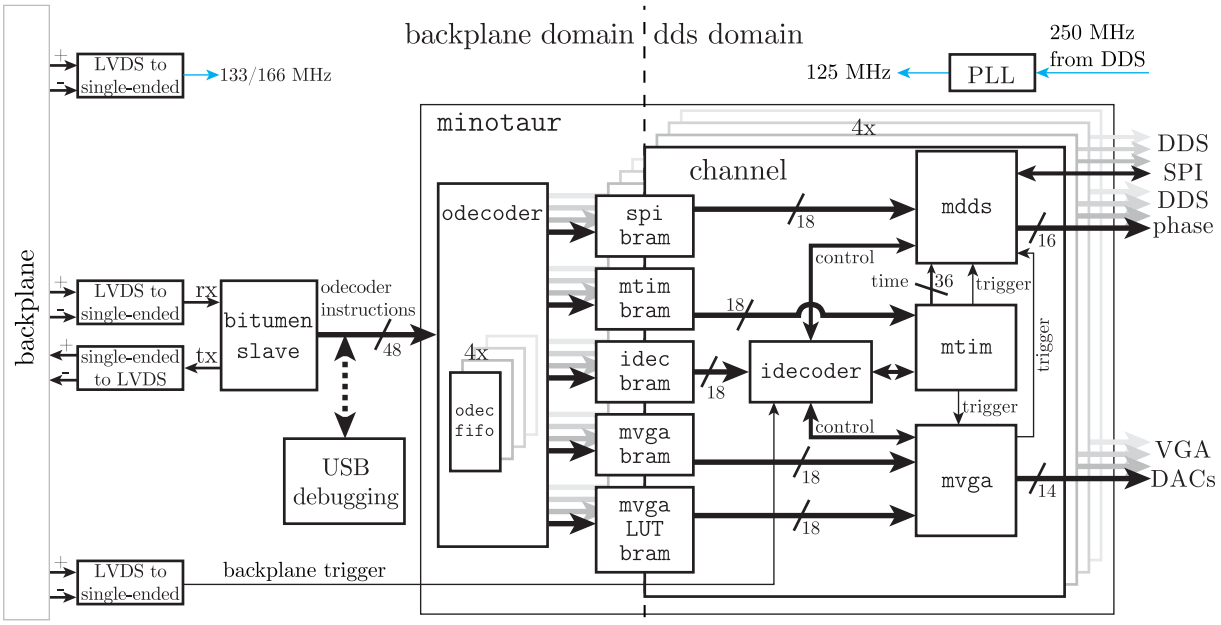


Figure 2.3: Schematic of the firmware used on the DDS cards. `bitumen_slave` deserialises the incoming data. `odecoder` distributes parts of it to the BRAMs of the submodules. `idecoder` controls `mdds`, which configures the DDS chips, `mvga`, which controls the VGAs that shape the RF output from the DDS chips, and `mtimer`, which triggers the other modules time-coherently. The figure is adapted from [12, pp.39].

The resulting construct has to be processed twice per run. Once to send the instructions to the DDS cards, not repeating looped parts, as looping a sequence is just an instruction for `minotaur`. The second time the `pulser` FIFO is fed with the right instructions and all loops have to be followed.

Different experiments appear as pages in the GUI known as `ionizer2`. They are dynamically created as they are added to `ionpulse SDK`. `ionizer2` allows scanning parameters to record a set of points. Every point is an average over multiple shots of the same experiment to improve the signal to noise ratio. Every shot consists of the defined sequence of pulses manipulating the qubit.

Chapter 3

Inter-sequence amplitude stabilisation

The M-ACTION system is certainly feature rich but cannot automatically adapt to temperature drifts. Some AOMs (especially the ones for qubit control) are switched off and on frequently. Depending on the duty cycle, their temperatures change and thus their optical properties. This influences the amplitude of the controlled beam. This chapter describes a working method which is capable of counteracting slow drifts.

3.1 XADC

The Zedboard provides a slow dual channel ADC (called XADC, the X stands for Xilinx) which works at a maximum data rate of 1 MSPS (Mega Samples Per Second) per channel with a resolution of 12 bits. It reaches its maximum accuracy if a stable voltage source at $(1.250 \pm 0.003)\,\mathrm{V}$ is used as a reference. This corresponds to an uncertainty of 9 LSBs.

The XADC can be used fully differentially which is particular useful in an environment with many digital signal lines potentially disturbing the signal. In this bipolar mode, common mode noise is rejected.

It has a range of $1\,\mathrm{V}$ starting at $0\,\mathrm{V}$ for unipolar and $-0.5\,\mathrm{V}$ for bipolar mode.

By default, the XADC is connected to internal temperature and voltage sensors but it can be connected to physical pins on the board and be used as a multipurpose monitoring tool.

The XADC comes with an internal sequencer which handles multiplexing for both ADC channels. It offers a conveniently configurable interface which amongst other parameters takes a channel mask and connects the masked channels one after the other to the ADC for measuring. This can be done internally which requires a direct connection of the desired signals to the XADC module

in the PL. Alternatively, an external multiplexer can be used. In this case, the current address of the channel under investigation is delivered to the PL and has to be forwarded to the multiplexer chip. 5 bits can be used to address up to 32 channels. In this case, the ADC is always connected to the same pins on the XADC module.

In the event-driven mode, conversions are triggered via the conversion start (CONVST) port. This enables the user to correlate measurements with other events. The XADC sends a signal on the End Of Conversion (EOC) every time a new sample is available to be read. The End Of Sequence (EOS) port announces the completion of the channel sequence defined by the channel mask and the return of the mux address to the first channel of the mask.

3.1.1 XADC signal conditioning board

Pascal Engeler developed an additional signal conditioning board during an internship in the TIQI group[21] which multiplexes 4 analog input channels to one XADC channel. The other channel is unused. It makes use of differential components to keep disturbances from electric field noise to a minimum.

Input signals are preamplified by the factor of 2.2 and sent through the multiplexer. This translates into an input range of $\pm 0.23\,\mathrm{V}$ considering the limits of the XADC. The previously mentioned 9 LSBs of uncertainty correspond to $\pm 1\,\mathrm{mV}$. The zero level of the preamplifier might be shifted by a potentiometer which offsets the inverting input level of the opamps used in the circiut. Additionally a capacitor between output and input of the opamp leads to a low pass filter with a cut-off frequency of 1 MHz. This is necessary to avoid aliasing because of the slow sample rate.

The Zedboard has a 20 pin connector dubbed "XADC header" which offers all necessary connections for the signal conditioning board. The header provides a 5 V power supply which is used for the opamps and the multiplexer.

There is no voltage inverter or negative 5 V supply on the amplifier board, which means that the common mode output level of the opamps has to be raised to $V_{ocm} = 1\,\mathrm{V}$. As the input is coupled to the output, this will raise the input device to the same level if it is DC coupled. V_{ocm} has to be used to allow one output to have a negative voltage compared to the common potential.

This causes issues if the input device is connected to ground and doesn't have a low impedance. For example, a photodiode might be directly connected to the PCB without issue. If one wants to log the signal with an oscilloscope the internal grounding of the coaxial cable shielding will connect one line to ground. This causes the input level of the inverting input of the opamp to

drop. The other input experiences a similar weaker drop as the effect is reduced by the internal resistance of the photodiode. This leads to an offset which is problematic and a power drain which is insignificant as the input resistors are 1 kΩ. The offset might be counteracted with the potentiometer but will always lead to jumps if an oscilloscope is dis- or reconnected.

3.2 Implementation

The XADC has to be instantiated in the PL and the pins connected to the main ADC channel and the multiplexer address pins wired to the right pad of the Zynq chip.

The EOC is used as an interrupt signal for the PS, which reads in the new sample. Conversions can be triggered by the PS through a general purpose IO port [21], which is accessible through a simple driver provided by Xilinx.

Communication between the XADC and the PS can be done through an implicit bus which comes with its own driver, `XAdcPs`. It is slower and contains more bugs than the AXI driver for the XADC, `XSysMon`. It acts almost like a drop-in replacement for `XAdcPs`. Only the prefixes of the functions have to be exchanged. As data is mostly read in during interrupts which hinder the linear flow of the main application, the time savings are beneficial to reduce the time spent in interrupt routines. Therefore, the XADC was connected to the PS explicitly via AXI.

To be able to trigger the XADC time coherently, I extended the `pulser` core. A previously empty flag can now be used to make `pulser` trigger CONVST during the currently running pulse. An extra register is used to specify an optional delay time between pulse start and triggering to take the rise time of the low pass filtered XADC input into account. It is only honoured if the corresponding flag is set as well. The PL uses a clock frequency of 166 MHz which means that delay times can be adjusted as precise as one clock cycle, 6 ns. The timing diagram of all relevant digital signals is displayed in Fig. 3.2.

Although the XADC is too slow to capture many samples per pulse, a pulse can be sampled in detail by repeating it many times using different trigger delays and thus measuring it at different times.

To use the XADC in `ionpulse SDK` the interrupts for EOC and EOS have to be registered in the PS. The `XScuGic` interrupt driver provides a concise set of functions to register callback functions which get called when an interrupt occurs. It stores the function handles and the respective interrupt ports in a table. In principle, the driver takes care of having a unique instance of this table. Due to conflicts with older code in the main application, which used a different approach of interrupt control, I wrote a new class to handle interrupts in the `ionpulse SDK`. It is a singleton which means that only a single instance of it can exist.

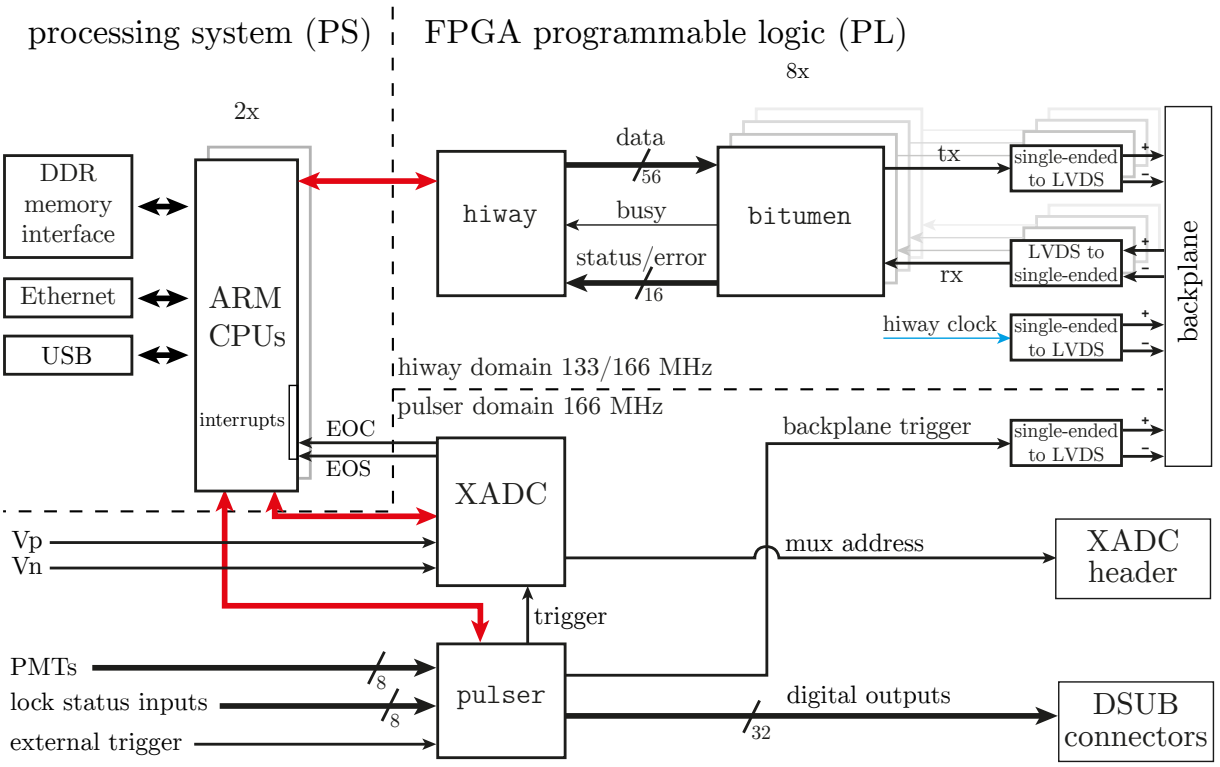


Figure 3.1: Schematic of the firmware used on the Zedboard. The red lines are AXI busses, the blue line distinguishes a clock. The XADC can be triggered by `pulser`. The figure is adapted from [12, pp.35].

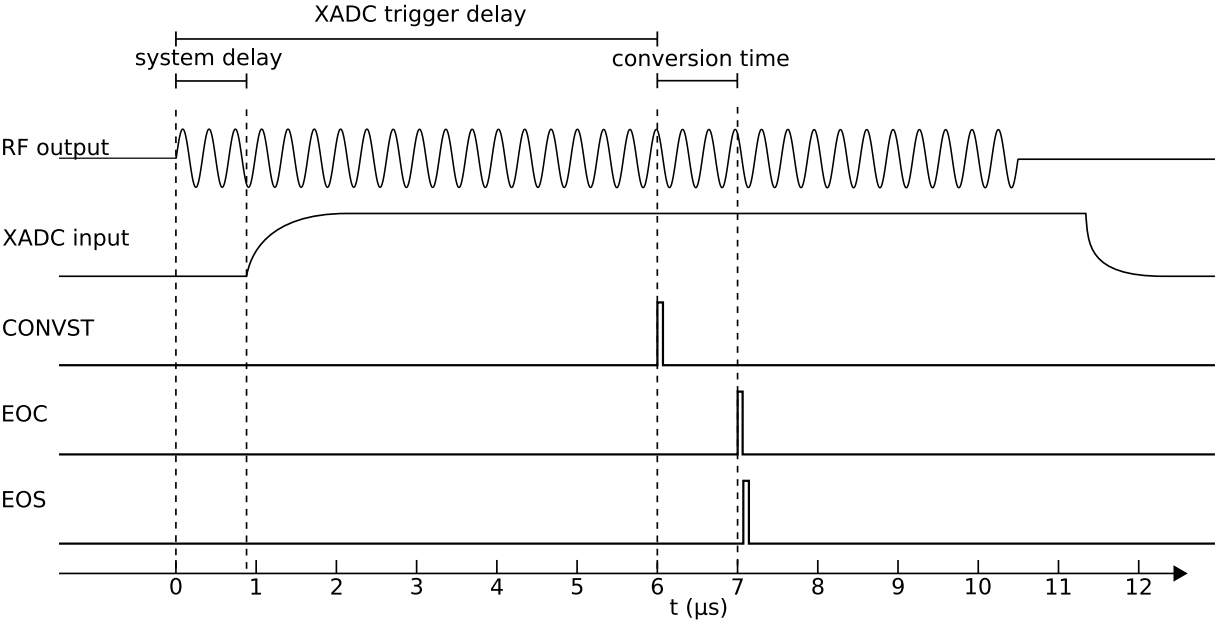


Figure 3.2: Timing diagram of XADC data taking. The `pulser` time slot starts synchronously with the DDS chips on-edge. After an inherent system delay due to the response of the AOM and cable delay, the XADC input voltage starts to climb towards the intensity measured by the photodiode. After XADC delay has passed, the conversion is triggered which finishes after a conversion time of $\approx 1\,\mu s$. EOC and EOS are pulsed sequentially. The RF signal and digital pulse lengths (6 ns) are not to scale.

The `XADCm` (m for manager) class was written to wrap the needed features of the XADC. It reads XADC values for each channel into a separate array and also contains the conversion functions to convert values into voltages. `XADCm` uses the EOS interrupt to reset the counter keeping track of which channel is being measured.

3.2.1 XADC experiments

To integrate the XADC into the experimental workflow, pages for `ionizer2` were written in the `ionpulse SDK`. One is used to measure the photodiode voltage just after a standard manual amplitude calibration. It sets the reference voltage that is used as the goal for stabilisation. The ideal pulse length and XADC delay time can also be found by scanning the respective parameter.

In order to use the XADC to actually stabilise pulse amplitudes, I wrote separate experiments for the Be and Ca lasers as the have slightly different structure. Both use a slow Proportional Integral (PI) loop which is run once per point. It uses a local amplitude which also displayed in GUI. It is limited to avoid the amplitude ever going above 100 % or below 0. For testing purposes, the local amplitude can be used without affecting any other experiments. This is useful to find the right P and I gains. To actually stabilise the globally used amplitudes, the local amplitude optionally overrides the master.

It is not possible to use this method during a running pulse sequence because the DDS cards run autonomously for the most parts once the sequence is started. In particular, values sent to a FIFO at the beginning of a sequence cannot be changed later in the sequence. Another problem is due to the current way of handling the parameters for the DDSes on the Zedboard and the DDS cards. The values for frequency, phase and amplitude have to be stored right next to each other on the BRAM. Therefore, different rotations which use the same amplitude and frequency but not phase need their own complete set of parameters. This leads to numerous objects carrying the same amplitude value and changing them after they have been sent is complicated.

The envisioned use case is doing interleaved stabilisation between actual experiments. The stabilisation experiment will be executed once in a while. `ionizer2` provides a scheduler which runs submitted experiments at certain intervals depending on their priority.

The Ca setup only uses one laser beam to address the qubit which makes the sequence straightforward as there only has to be one pulse of the qubit laser which gets measured. The Be qubit is addressed through a Raman laser which requires 2 laser beams. Therefore, two laser pulses are required, each pulsing one of the Raman lasers separately.

3.3 Results

Since the photodiode voltages exceed the input range of the XADC amplifier, attenuators have to be inserted. The 729 nm-laser needs 20 dB of attenuations while the 313 nm Raman lasers need 6 dB. The voltage level of the photodiode dependent on the amplitude set in `ionizer2` was measured in a scan to get the gradient of the voltage (see Fig. 3.3). Using the first derivative of the fit, the approximately linear voltage dependence at the operation point of around 50 % is $(6.3 \pm 0.2)\,\mathrm{mV}\,\%^{-1}$.

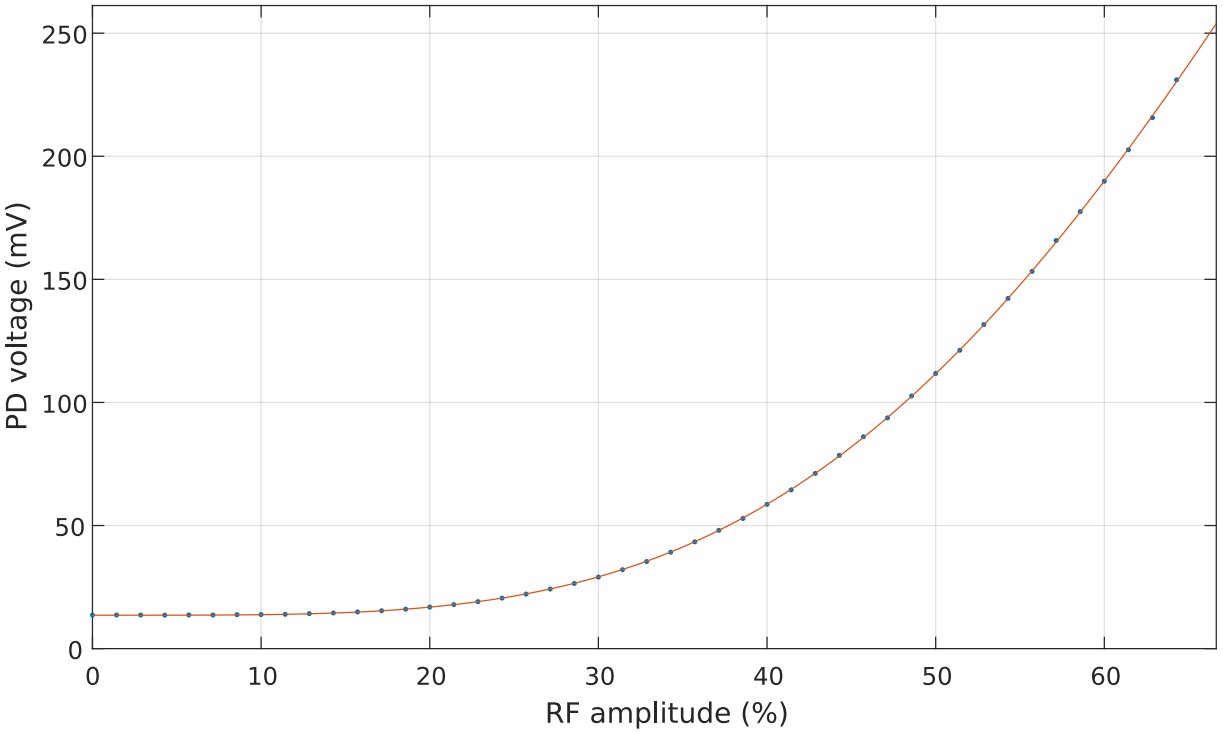


Figure 3.3: Voltage of the 729 nm laser photodiode dependent on the RF amplitude. The curve is fitted with the intensity model of a double passed AOM where the diode output voltage V is linearly dependent on the intensity. $V(x) = a + b\ \sin^4(x/c)$ with $a = (13.58 \pm 0.10)\,\mathrm{mV}$, $b = (617 \pm 9)\,\mathrm{mV}$ and $c = 73.2 \pm 0.4$.

The XADC delay was scanned to measure the pulse shape as seen be the XADC (see Fig. 3.4). A fit was made to retrieve the time constant $\tau = (1.172 \pm 0.008)\,\mu\mathrm{s}$ of the LP filter. The minimum delay time to reduce the damping effect from the filter below the 12 bit resolution of the XADC is $\tau \ln\left(2^{12}\right) = (9.75 \pm 0.07)\,\mu\mathrm{s}$. Thus a minimum settle or acquisition time of 10 µs was used to perform reference measurements. In Fig. 3.4 there is also a dead time of $(0.7 \pm 0.1)\,\mu\mathrm{s}$ before the voltage starts to rise. This is mainly caused by the AOM.

The original plan was to use the XADC to stabilise the lasers during a day or days-long experiment. The 729 nm laser amplitude is calibrated using the ion Rabi oscillations such that an

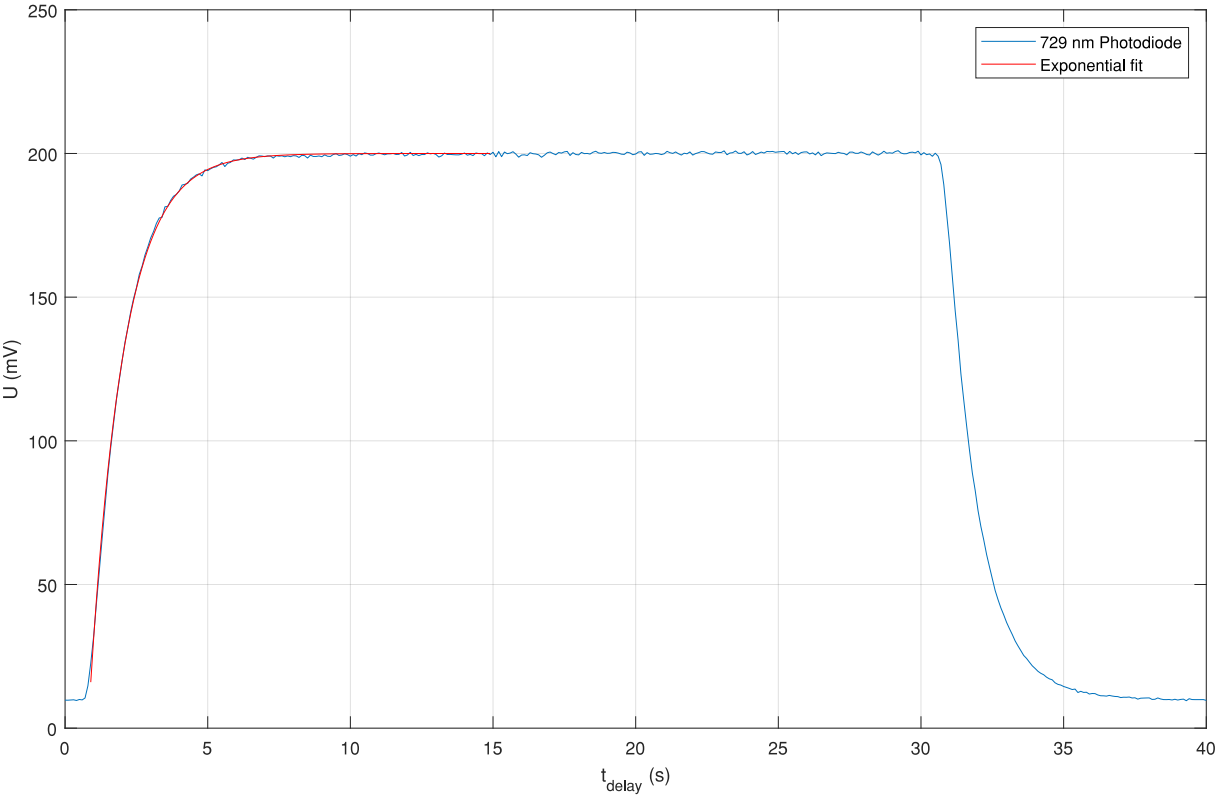


Figure 3.4: Voltage of the 729 nm laser photodiode dependent on the XADC delay. The exponential fit of the rising edge has a time constant of $\tau = (1.172 \pm 0.008)\,\mu s$.

$R_x(\frac{\pi}{2})$ gate is performed in 1.4 µs. The reference voltage can then be obtained by averaging multiple voltage measurements of a longer 10 µs pulse with the same amplitude. To allow the AOM to cool down a long waiting time is introduced between the pulses. As the same measurement has to be done for the PI loop running in between other experiments, the required time for the measurement should not significantly increase the overall time. An average of 1000 pulses takes 0.5 s and leaves a standard deviation of 0.2 mV. As the accuracy of the XADC is lower than this value the precision over time has to be investigated.

If the XADC was stable enough, the measured reference voltage should be valid over a long period of time. A frequent recalibration by probing the ion could be replaced by the automatic PI loop which stabilises the voltage relative to the reference. While testing the XADC with the segmented ion trap setup, the reference voltage seemed to drift by $(0.5 \pm 0.2)\,mV$ within one hour while the optimal amplitude changed by (0.2 ± 0.1) %. The reference voltage seems to be more stable than the amplitude as the gradient is greater than the ratio between the reference voltage and amplitude drift by the factor of 2.5 ± 1.0. The voltage drift might be explained by parameter drifts of the amplifier board, which is not in a temperature stable environment or drifts of the laser beam position in the trap. The latter cannot be detected by the photodiode because of its large surface area.

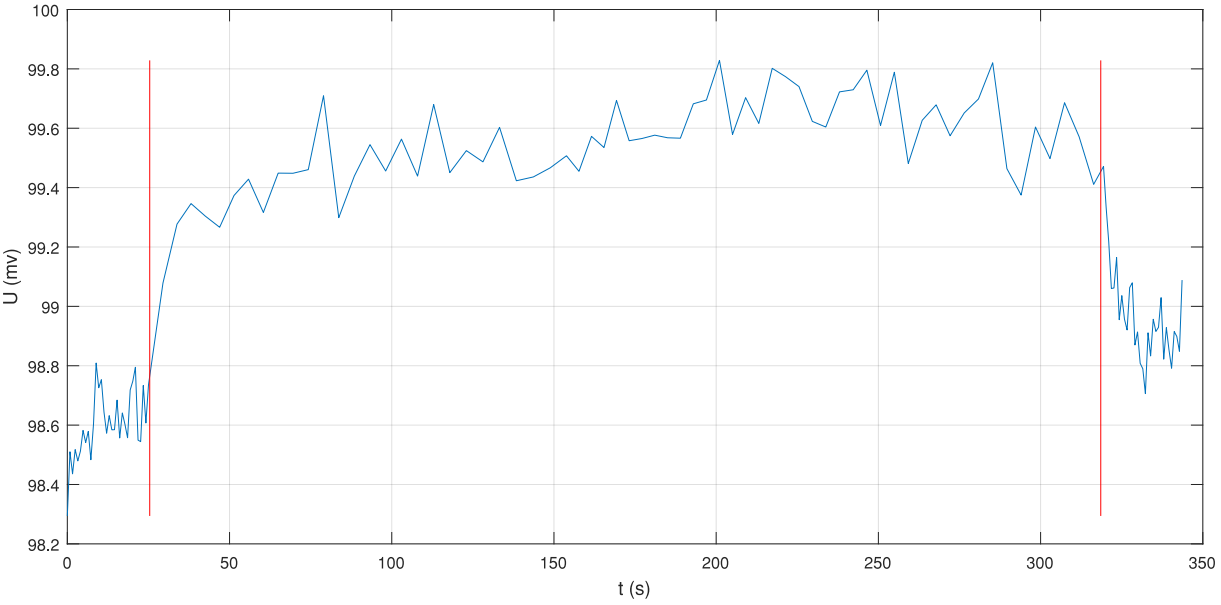


Figure 3.5: Time dependent voltage on the 729 nm photodiode. The red vertical lines mark start and stop of the randomized benchmarking experiment conducted in parallel. Data points are less dense during RB because the scheduler of `ionizer2` runs the voltage reference measurement experiment less often.

To analyse the intensity behaviour during randomized benchmarking, `ionizer2` was configured to run the XADC reference measurement experiment once after 10 RB sequences. A voltage increase of 1 mV was observed right at the beginning of the sequence. The voltage drops back to its original value after the sequence ends. This corresponds to an amplitude change of around 0.1 %. An example is given in Fig. 3.5.

The amplitude stabilisation is able to keep the measured intensity on the photodiode stable. This seems to increase the average gate fidelity as it can be seen in Fig. 3.6. The setup was not in a well aligned state and therefore the average fidelities were much lower than usual. The high error rate probably increased the visible effect of the amplitude stabilisation and was actually beneficial in this case.

Further experiments, especially with the Be lasers, have to be done to fully characterise the benefits of using a slow intensity stabilisation of laser amplitudes. With a few optimisations, for example putting the signal conditioning board into a closed box, the XADC could be used as a precise reference and improve the stability and therefore fidelity of ion trap setups.

The XADC together with the signal conditioning board could be used to monitor any slowly-varying analog signal directly through the control systems. This includes the output of standalone PID loops or temperature sensors. Together with the described software, this provides a versatile and easy to use framework which allows to integrate new components quickly into the current control system.

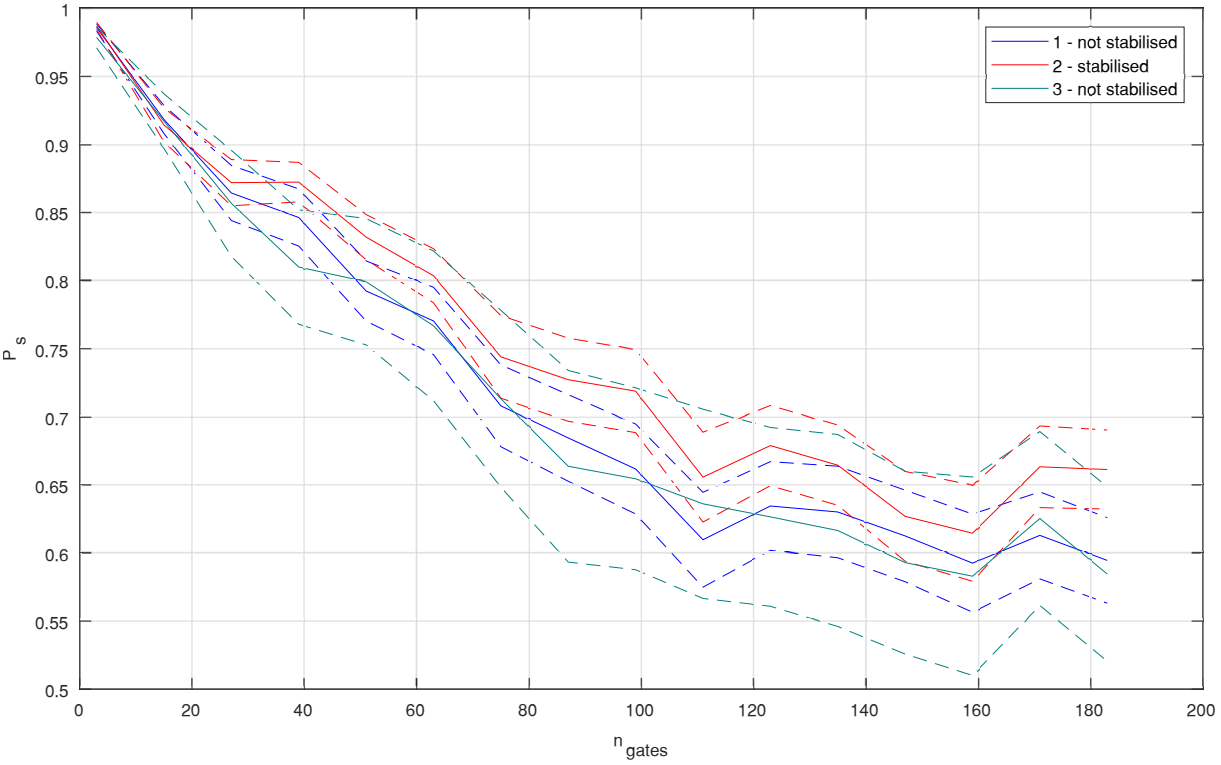


Figure 3.6: Average final state fidelity depending on the number of randomized benchmarking gates. Measurements 1, 2 and 3 were taken after each other. The fidelities are much lower than usual due to the bad state of setup when data was taken. The 68 % confidence intervals are indicated.

Since the XADC is included in all 7th-generation Xilinx chips, the acquired expertise might be helpful for other devices in the future as well.

Chapter 4

Intra-sequence amplitude stabilisation

For longer experimental sequences and pulses, the stabilisation possible with the XADC is insufficient. A scheme that could directly stabilise the pulse intensities during a sequence is needed. For example, the implementation of the Mølmer–Sørensen (MS) gate in the segmented ion trap uses pulse times of approximately 30 µs. Additionally, pulses are shaped to narrow the frequency distribution of the diffracted laser beam. As mentioned before, interaction with a running sequence from the `ionpulse SDK` is limited. Furthermore, the backplane introduces latency in the order of the delay caused by the AOMs. It is therefore desirable to integrate a higher-bandwidth feedback loop into the lower-level hardware, which can operate independently of the Zedboard in real time. It is still desirable to be able to configure the loop from the main GUI to increase usability and flexibility. As mentioned in Sec. 1.4.2.1 the loop has to be switched time coherently with the RF signal generation.

The latest hardware revision of the DDS cards includes a dual channel 62.5 MSPS[1] 14 bit ADC which can be used to perform feedback directly on the cards themselves without needing the Zedboard to intervene.

The signal conditioning for the DDS ADCs is similar to the XADC. The inputs are differential and an opamp with lifted V_{ocm} is used to amplify the signal by a factor of 2. The low pass filter has a higher cutoff frequency of around 40 MHz. The ADCs are connected to the FPGA in interleaved mode over one 14 bit bus running at 125 MHz.

Since changing the amplitude on the DDS chips adds latency through the slow SPI connection, the VGAs were chosen to change the amplitude of the RF output.

[1] Analog Devices AD9248

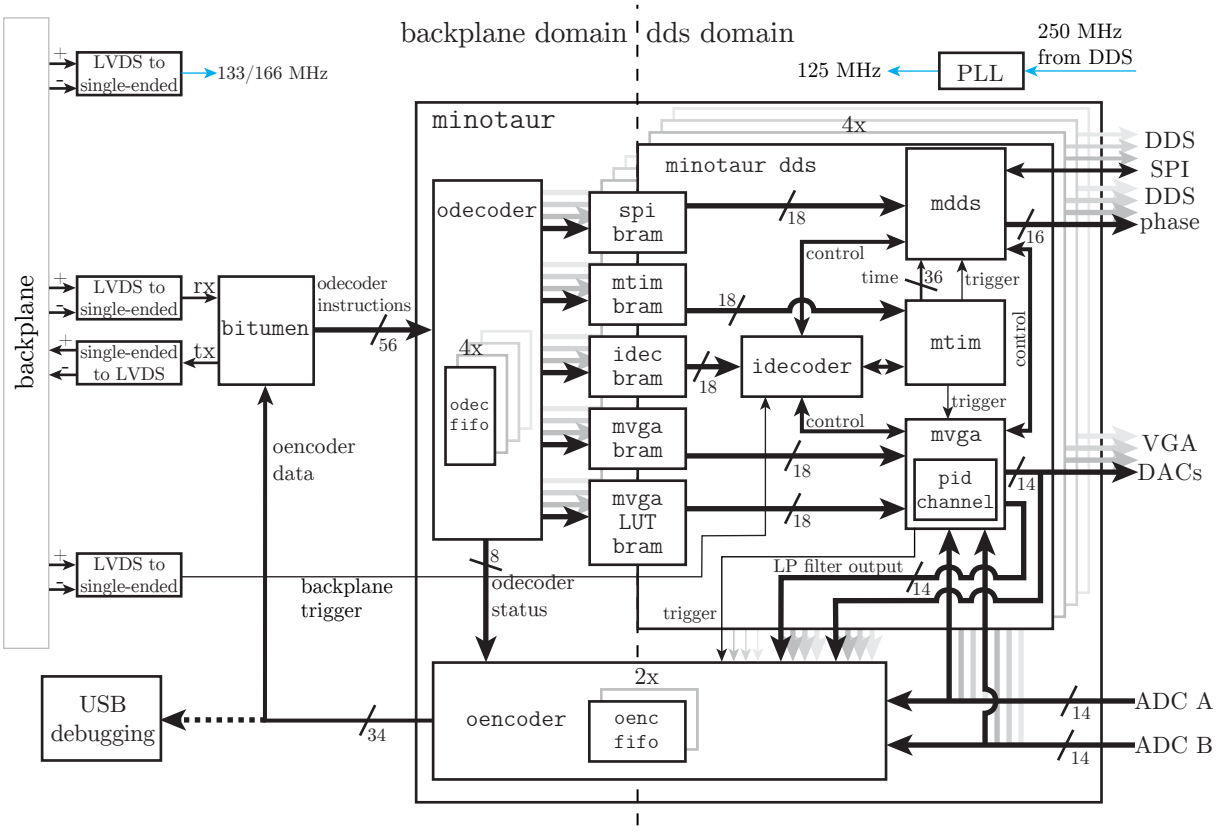


Figure 4.1: New firmware for DDS cards. Compare with Fig. 2.3. The figure is adapted from [12, pp.39].

4.1 Firm- and software changes

The next step in stabilisation involves significant changes of the firmware. However, the overall structure remains the same.

All steps were carefully planned and I made sure to have proper understanding of the old firmware to not degrade other modules by my changes. Many design steps were taken with further extensions in mind.

The architecture of the new firmware of the DDS card is visualised in Fig. 4.1. All changes are discussed in the upcoming subsections.

4.1.1 Backplane communication

From previous projects [22] it was clear that a data stream containing input and feedback is very valuable when calibrating a PID loop. In order to pass this data on from the Zedboard to `ionizer2`, it has to be sent from the DDS cards to the Zedboard first.

The old `bitumen` core does not allow two way communication and was not intended to do so. Thus a complete rewrite was necessary.

Another aspect that was not included in the old version was error correction. Errors were detected but on the DDS card did not have any effect. This means, that a faulty instruction was processed just like any other which caused problems when testing the cards with an unstable electrical connection. Once the backplane connection is physically stable almost no bit errors occur so this was not an issue in general. Two-way communication with increased traffic certainly increases the probability of encountering errors which was a reason to include error handling into `bitumen`.

The new bitumen core is symmetric, which means that both the Zedboard and the DDS cards are using the same module. The only difference is a different priority assigned to receiving and sending which can be set by a single parameter. This reduces the development effort. The master-slave architecture is dropped.

Since the backplane communication is synchronous, there is no need to divide larger messages into small chunks. For error checking, one parity bit might be inadequate to assure the integrity of a 64 bit message. A common way to use more bits to optimally identify errors are Cyclic Redundancy Checks (CRC), explained in Sec. A.3. An 8 bit CRC message is enough to detect all 4 bit errors of an up to 119 bit payload. The serialiser and deserialiser modules, `bit_tx` and `bit_rx`, were rewritten to handle a dynamic message length and calculate as well as respectively check the CRC. In theory, single bit errors could be corrected by the deserialiser however the increased complexity is not worth the effort if the bit error rate is very low like in this case.

In order to explain the new communication protocol, two `bitumen` cores communicating with each other are given names. One is called Alice, the other one called Bob, common names used in cryptography.

If Alice sends a message to Bob, the deserialiser tells him if any errors occurred. Bob responds to Alice who then repeats the message if her first message was received with faults. Alice could wait for a postive response before proceeding to send the next message. This would introduce some deadtime in the transmission line while she waits. It is more efficient to send a second message right after the first one, hoping that Bob received the first one fine and will send a positive response. Because responses are shorter than messages, Alice will know before the second message was sent completely, whether Bob saw any errors. She can then start sending the next message immediately after the second one has been serialised.

The roles can be reversed and responses need to be distinguished from new data. Two extra bits are added at the beginning of a transmission, encoding both possible message types.

Errors lead to repeated messages. To allow Bob to distinguish between a repetition and a new message a packet counter is added to every message. It increases continuously and overflows to zero. A response message includes the packet counter of the original message it is a response to and the anticipated packet counter of the next message. Alice knows that Bob acknowledged the message if the packet counter of the response is larger than the packet counter of the message by one.

The transmission lines are permanently high if inactive. The start of any transmission is marked with the line going low for one clock cycle. If a random bitflip in the transmission line from Alice to Bob occurs without a message being sent, Bob's deserialiser will treat it as a new message and assert its faultiness after it is finished. If Alice sends a new message in the meantime, the beginning will not be captured correctly. An out-of-sync issue arises because Bob will send responses reporting errors and Alice will repeat her message continuously. This can be fixed by pausing transmissions for a certain amount of time to allow the deserialiser to complete its last message and correctly identify the start of an error-free transmission after the pause. There are several mechanisms to detect out-of-sync issues.

Although `bitumen` can recover from most errors without any packet loss, sometimes the repetition limit is exceeded and `bitumen` proceeds with the next packet. In this case, Alice's packet counter will be higher than Bob's anticipated counter. A resynchronisation procedure was developed to allow Alice to override Bob's counter by repeatedly sending her new packet until Bob is sure that he should acknowledge the new message.

Even if an error did not cause any packet loss, it should still be reported to allow identification of issues with the backplane. An error register is exposed to `hiway` for example which accumulates all errors since the last transmission. It will be cleared if a new message is sent.

The new `bitumen` implementation has a stronger focus on flexibility and word structures can easily be changed by parameters. The Zedboard sends 56 bit payload messages while `minotaur` only sends 34 bits. The corresponding parameters have to be set before synthesis. The deserialisers of both sides need to know about the message length they should expect. Mismatches lead to complete communication failure.

The final core has less overhead than the old one, implements a stable error correction and is able to send messages both ways in parallel. Its flexibility should make it adaptable to future requirements. At the moment, generic differential IO pads are used to connect to the backplane. If dedicated transceivers with a much higher clockspeed than the current backplane were used only the CRC calculation would require modification to allow multiple bits to be sent per cycle. There are byte-wise calculation schemes which use LUTs to calculate CRC sums faster.

4.1.2 ADC integration

Data that needs to be sent to the Zedboard has to be prepared on the DDS card first. The `oencoder` module was written from scratch to generate messages for the Zedboard.

In general, data from the DDS clock domain needs to be sent to the Zedboard. This means it has to cross the clock domain to the backplane. Clock domain crossing is a delicate subject as the edges of clocks with no fixed frequency or phase to each other can occur arbitrarily close in time. Instabilities are the consequence, and must be considered carefully. This problem is commonly known as Clock Domain Crossing (CDC). In particular, if data crosses the domain, problematic bit flips occur on the other side. A safe way to transfer multiple bits in sync with each other to a different clocking domain is a two-port FIFO. Its input is clocked by one domain while the output is clocked by the other. On the FPGA this gets implemented directly to the dual port BRAMs which makes it fairly efficient.

Often data can not be sent over a sustained period over the backplane because it is too slow. In some cases it might be useful to have a contiguous set of data which requires a buffer. Since a FIFO has to be used anyway it can be extended to work as a buffer as well. The size of the buffer is physically limited to 1024 samples due to the size of the BRAM modules on the FPGA. The module `oencoder_buffer` wraps the FIFO and other features. A trigger was added to make it possible to trigger the acquisition of one buffer or to simply select certain samples from the incoming data stream. Averaging was also included to allow a longer timespan to be recorded in one single buffer. The sample size is freely configurable in factors of two up to $2^{15} = 32768$ samples.

One data stream of interest are the ADC values. Cagri Önal wrote the necessary module which splits the interleaved 14 bit bus into two separate buses on the FPGA. Another one is the data stream to the high-speed 14 bit DACs setting the gain of the VGAs. This stream is twice as fast as the ADC stream, because the DACs take 125 MSPS. Often two streams with a constant phase relationship are very helpful for debugging. The `oencoder` module instantiates two `oencoder_buffer` modules which can handle two stream synchronously. When sending to the Zedboard they are read out in parallel and one sample from each buffer is sent to the Zedboard in one message. There is also the possibility to send two samples of the same buffer in one message if maximum data rate for one stream is required. If more streams in parallel are required in the future, `oencoder` can be configured through parameters. The control word structure configuring the buffers would have to be changed manually.

Since both instances of `oencoder_buffer` act autonomously they are not synchronised by default. This is an issue if two streams with different data rate or averaging are recorded. Correct configuration could be ensured at a higher driver level in `hiway`. Currently, the users have to

make sure they choose the right parameters. As a failsafe mechanism, `oencoder_buffer` can be emptied and thus resynchronised by changing to the default inactive mode.

4.1.2.1 Data handling on the Zedboard

Incoming data on the Zedboard has to be buffered in the PL before it is read in to the PS. One input FIFO for every card was added to `hiway` which stores all incoming data samples. Each buffer notifies the PS if at least 16 samples are available through a separate interrupt line. To avoid interrupting the PS too often, coalescing was added which limits the interrupt frequency. In general, interrupt coalescing describes the process of accumulating work or waiting for a timeout before an interrupt is triggered. In this case, the PS is not notified for every single sample to read in but only for groups of 16. Additionally, a hard minimum wait time between interrupts is added, which allows the PS to proceed with the main application if a long stream of data is sent to the Zedboard.

The PS reads in all available samples through AXI. The `hiway` driver splits the message up into the two streams and stores them separately in a vector. The vector can be converted from raw integers to voltages and further used in the `ionpulse SDK`.

Characterisation of the ADC and its preamplifier revealed strong crossover distortion of the differential opamps. For the current application this is not an issue because the photodiode voltages are always positive and the zero voltage level is never crossed.

4.1.3 Instruction word changes

The old 48 bit instruction word structure was not versatile enough to cover the new settings for `oencoder` and ADC configuration. A new layout with many options for extensions was developed. A meaningful upper limit for word size is 56 bits because two AXI transmissions contain 64 bits and 8 bits are needed to mask the cards the instructions should be sent to. So 56 bits were chosen. The first 4 bits specify the word type which completely changes the meaning of the remaining 52 bits. So far only one word type was sufficient to cover all eventualities. The old instruction word used exclusive data and control words and avoided multiple meanings of the same bit. This structure was dropped for more flexibility. `odecoder` is now able to change the channel mask and write to BRAMs accordingly in one instruction which reduces traffic on the backplane. The empty channel mask is used to address the configuration registers of `oencoder`.

A detailed description of the new control word structure can be found in A.5.

4.1.4 Feedback loop

The module `mvga`, which is responsible for pulse shaping, was extended to incorporate a PID module which controls the VGA gain depending on the ADC readings of one channel. The module `pid_channel` includes a discrete first-order LP filter in front of the actual PID loop. It is based on the design used in the EVILs which are two-channel PID controllers used frequently in the TIQI group. It uses two identical ADCs to read in a feedback signal and is equipped with a slow and a fast DAC. They are controlled by a server software called DEVIL running on a Raspberry Pi via a serial connection over USB. The Raspberry Pis are addressable through ethernet. One single client running on a PC can be used to connect to multiple EVILs over the network (see Fig. **??**). The basic firmware design was developed by Dr. Vlad Negnevitsky, the LP filter by Alexander Hungenberg [23] and the latest version of the PID core by myself [22]. `pid_channel` could be added to `mvga` without changing the existing design too much because it had only been used for pulse shaping which does not need extensive on-the-fly configuration. The lowest addresses in the BRAM are used to store non-time-critical parameters such as LP filter responsiveness, PID gains or PID starting value. The `mvga` instruction set had enough flexibility left to add commands for reading in these new parameter values from the `mvga` BRAM.

Setting the reference voltage is a more difficult task as it can vary from edge to edge. In this case, an edge describes a parameter change of the DDS output in frequency, phase and/or amplitude. Simply adding additional instructions to an experimental sequence would work but is inefficient. Since complex experimental sequences are already taking the `idecoder` BRAM to its limit, a different approach was chosen. There has to be one voltage reference per edge amplitude. Edge amplitudes are always coupled with frequency and phase which means they can only ever take up one fourth of a BRAM[2] which are the same size for the `mdds` and `mvga` module. All edge amplitude addresses could therefore be mapped to the last fourth of the BRAM of `mvga`. When the amplitude of the edge is zero, the reference voltage is meaningless which can be marked by an extra bit as BRAMs have a width of 18 bits and only 14 bits are needed to store a reference voltage.

Now, when the `idecoder` sends the address of the parameters of the next pulse to `mdds`, `mvga` also receives it. It maps it to an address in the upper third of its BRAM and reads the corresponding reference voltage that has been stored there during the setup phase before the experimental sequence. It changes the reference voltage accordingly if it is a valid one. If not, the PID loop will stay in its current state.

The PID core has to be paused (kept at the same value) if the RF output is zero. Otherwise, it would drift uncontrollably as the feedback would have no effect on the measured voltage. Additionally, there are delays between RF output changes and voltage changes. Especially at

[2]One BRAM slot is 18 bits wide but frequency parameters are 32 bit wide and take up two slots

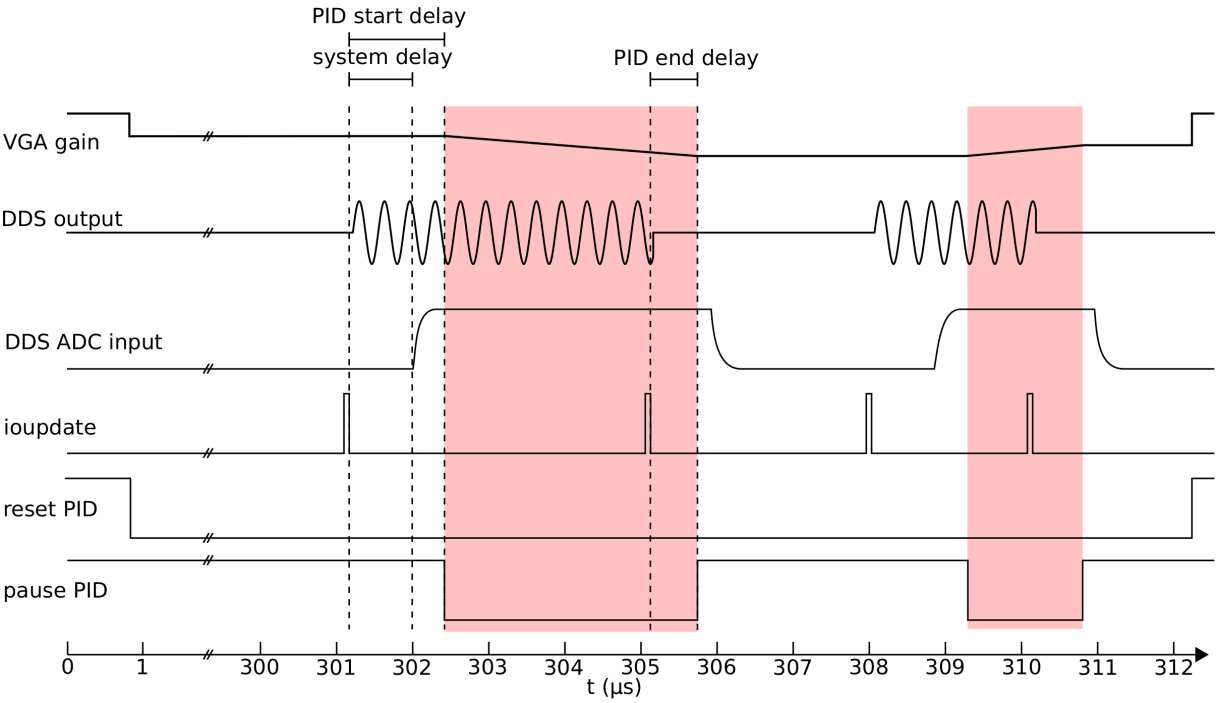


Figure 4.2: Timing diagram of a pulse that is stabilised with the DDS ADC PID loop. Red squares signify that the PID loop is active. DDS output frequency is reduced. ioupdate pulses are depicted longer than they are (8 ns). There is a short delay between the internal ioupdate signal and a change in DDS output. When `reset_pid` goes low the PID output, and therefore the VGA gain, returns to the last value before the reset. This value can be overriden by the Zedboard while the PID loop is reset.

the edges this leads to a large temporary mismatch between setpoint and signal. To avoid jumps a configurable delay was added so the PID core can be resumed after the rising edge. The delay at the end of a pulse should be shorter because the PID core has to be paused before the falling edge occurs. These delays are configured analogously to the PID gains. Rising and falling edges are discerned by the valid flag of the reference voltage of the edge.

A timing diagram of unshaped pulses is shown in Fig. 4.2. In the case of shaped pulses, the PID delays start when the shaping is completed or started (see Fig. 4.3).

4.1.4.1 PID channel

Although the basic structure for the numerical calculation of the feedback signal already existed and is straightforward, it had to be rewritten several times to add valuable features and make it efficient. Several times in the design, the FPGA synthesis tool ran into severe issues when trying to meet timing constraints. `minotaur` and `bitumen` had to be improved in multiple ways to overcome these issues.

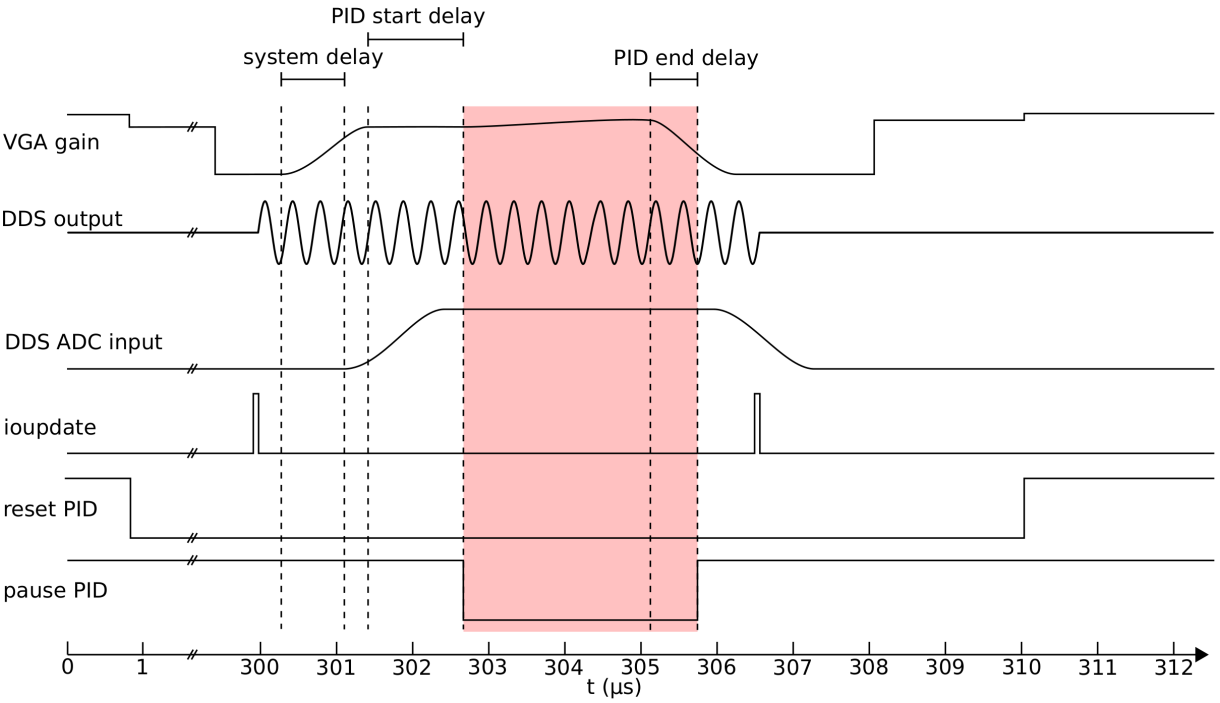


Figure 4.3: Timing diagram of a shaped pulse that is stabilised with the DDS ADC PID loop. The PID start delay counter is started when the shaping finishes. The end delay is triggered by the start of the ramp down of the VGA gain.

The FPGA contains digital signal processing (DSP) blocks which are able to perform one 18 bit pre-addition, 18 by 18 bits multiplication and 48 bit wide addition without using any additional multipurpose logic. The result can be looped back to the wide addition which makes it an accumulator if desired. The equation would be $C' = (D+B)A+C$ with D, B and A being 18 bit wide signed integers and C being a 48 bit signed integer. These DSP blocks are also faster than general purpose logic which is beneficial to avoid any timing issues. The modules `pid_lp_filter` and `pid_core` were modified to allow the synthesis tool to automatically infer DSP blocks. The LP filter fits into one DSP block and the PID core needs one block per term.

A brief description of the relevant control theory follows. For a short and superficial introduction see my semester thesis [22].

The z-transfer function of the LP filter (see Fig. 4.4) with cutoff frequency proportional to `f` reads:

$$G_{LP}(z) = \frac{\mathtt{f}\, z^{-4}}{1 - z^{-1} + \mathtt{f}\, z^{-3}}$$

The z-transfer function for the PID core (see Fig. 4.5) with gains `p`, `i`, and `d` reads:

$$G_{PID}(z) = \left(\frac{\mathtt{i}\, z^{-1}}{1 - z^{-1}} z^{-2} + \mathtt{p}\, z^{-1} z^{-1} + \mathtt{d}\, (1 - z^{-1}) z^{-1} \right) z^{-5}$$

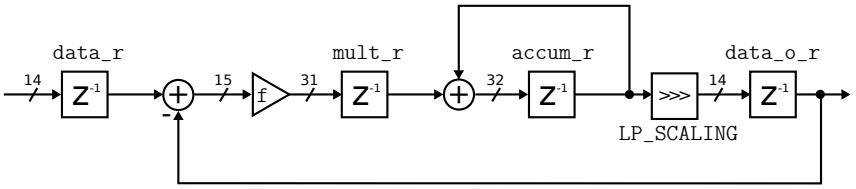


Figure 4.4: Visualised architecture of the LP filter.

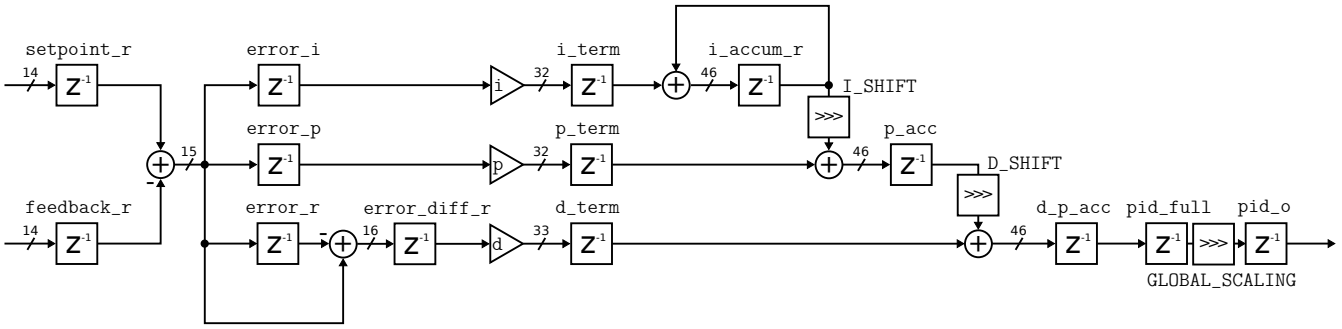


Figure 4.5: Visualised architecture of the PID controller. The final addition of the terms is performed cascaded to allow synthesis into one DSP block per term.

Putting both transfer functions together, the frequency response of the complete channel can be calculated and drawn (see Fig. 4.6). The LP filter cancels out the rising gain of the differential term for higher frequencies. To achieve a falling gain, a higher order LP filter would be necessary.

4.1.4.2 Software changes

The reference measurement experiment used for the slow stabilisation using the XADC was adapted to be used with the ADCs on the DDS cards. An interface to change the gains and delays used for the PID loop was developed. RF pulses can be recorded with `oencoder_buffer` and sent to the Zedboard. It is possible to capture several data streams. Per card one can choose between both raw DDS ADC data channels, the LP filtered ADC data per channel and the DAC values for the VGA per channel. Since `ionizer2` currently does not implement a mechanism which would allow plotting multiple values per shot, the data is sent via UART over USB as well. This is slower than the backplane by a factor of 10 but is sufficient to gather debugging data. Additionally, this also limits the maximum buffer size to the number of samples the UART buffer of the connected PC can hold. For example, the laptop used to connect to the DDS cards provided a buffer of 2048 transmissions with 8 bits each. This is enough to store 512 samples of two streams in parallel.

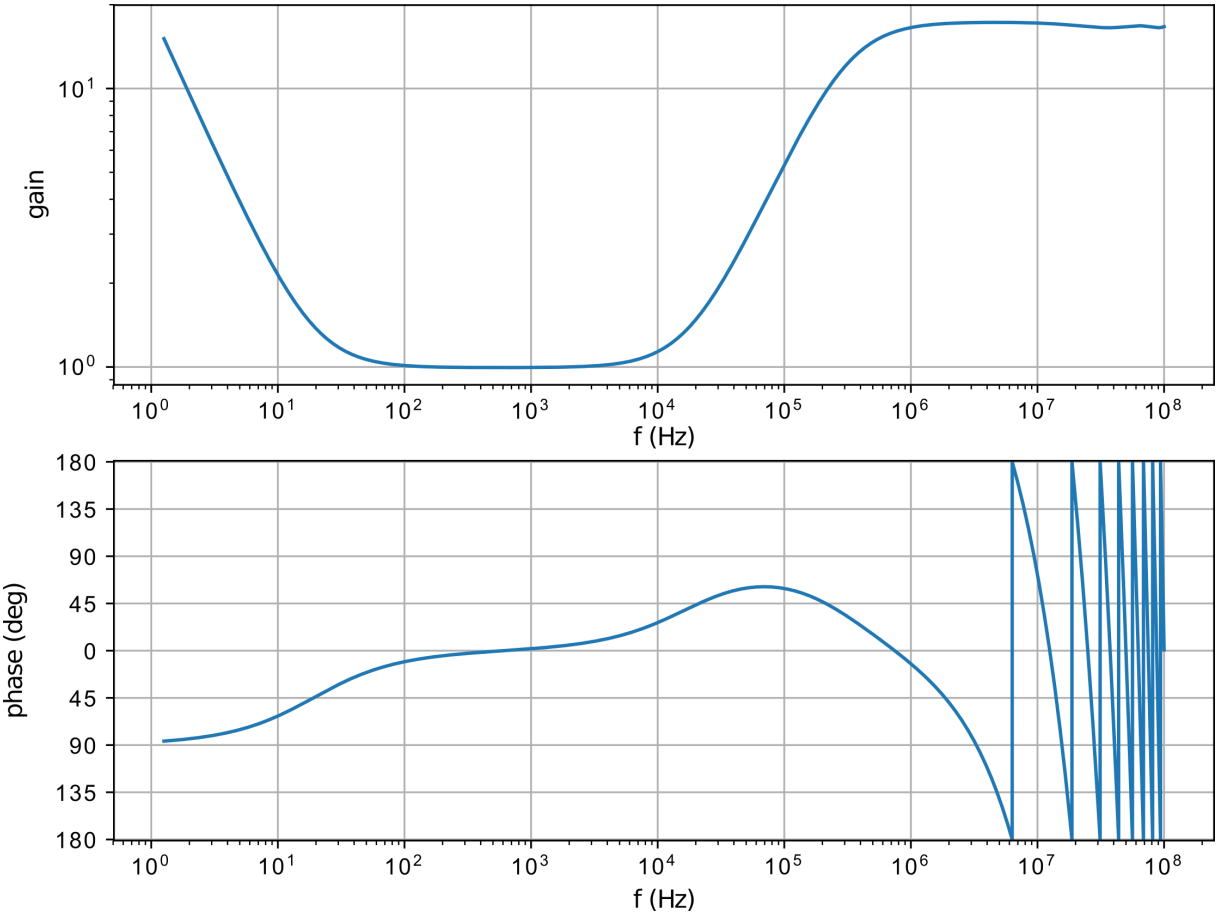


Figure 4.6: Frequency response of the PID channel including the LP filter. An additional delay of 160 ns was added to take delays in the signal path into account. `f`, `p`, `i` and `d` were chosen such that all important regimes are visible.

4.1.5 FIFO flow control

The memory available to store a pulse sequence is limited on the DDS cards. Usually, long sequences can be split into loops which use significantly less space in the BRAMs but still the new data for every loop has to be stored in the FIFOs of `odecoder`. This limits a sequence to 409 edges with an arbitrary frequency, phase, amplitude and time.

In the case of randomized benchmarking, the maximum number of gates is limited to roughly 200 gates as on average the gates are composed of two rotations. This is a limiting factor when benchmarking the segmented ion trap setup when it is operating at optimal fidelities.

When the `ionpulse SDK` processes the pulse sequence a second time and `pulser` is supplied with new instructions in time, the DDS FIFO data is also sent interleaved. The FIFO of `pulser` cannot overflow because the AXI interface is blocked when the FIFO is full. However, the Zedboard had no information about the filling level of the FIFOs on the DDS cards. As a safety feature, the maximum number of instructions for the FIFO was limited to its size, so it could never overflow.

A simple flow control mechanism, to inform the Zedboard of the FIFO space available, would allow this limit to safely be lifted.

`oencoder` was extended to send a DDS card status update when the filling levels of the `odecoder` FIFOs go below or above a certain threshold. This feature can be turned on or off from the Zedboard. `hiway` keeps track of all the current per-channel FIFO statuses and interrupts the PS if a status changes. This way, the `hiway` driver knows if the FIFOs can handle new data or not. When the DDS card FIFOs are full, `ionpulse SDK` proceeds to fill the `pulser` FIFO in the meantime.

The original mechanism which balanced `pulser` FIFO and DDS card FIFO writes couldn't handle long periods of exclusively writing to the `pulser` FIFO. Essentially, the AXI interface of `pulser` blocked too long because of too many writes and the FIFOs of the DDS cards were emptied in the meantime resulting in a pulse sequence continuing with invalid data.

This can be prevented by checking the FIFO filling levels before issuing an AXI write. However, for now this has to be done explicitly through an AXI read. This frequent polling is inefficient. A proper solution would be employing a similar status update through interrupts by `pulser` which would also replace the currently used polling of PMT FIFOs for new results. The interrupt lines have already been added to `pulser` but there was insufficient time left to accommodate the changes at the driver level.

The handling of `pulser` FIFO writes was altered to aim for a user-defined filling level. As `pulser` synchronisation issues cause problems with the workflow on the segmented ion trap, keeping above the minimum filling threshold for `pulser` was assigned a higher priority than sending to the DDS card FIFOs.

Since the lasers used to address the transitions in Ca are powerful the gate times can be as short as the minimum pulse time of 1.4 µs for a $\frac{\pi}{2}$ gate. This is not enough time for the backplane to transmit all instructions needed for a unique pulse which takes 2.5 µs. `pulser` needs to be supplied with new instructions as well, which can also take up to 2 µs per pulse. For randomised benchmarking of Ca, the data throughput currently limits the maximum number of gates.

4.2 Results

The delay of the PID loop on the card is measured by recording the step response of the loop (see Fig. 4.7).

The maximum stabilisation precision can be estimated using the signal-to-noise-ratio (SNR) of the DACs controlling the VGA and the SNR of the ADC. The DAC has an SNR of 75 dB which

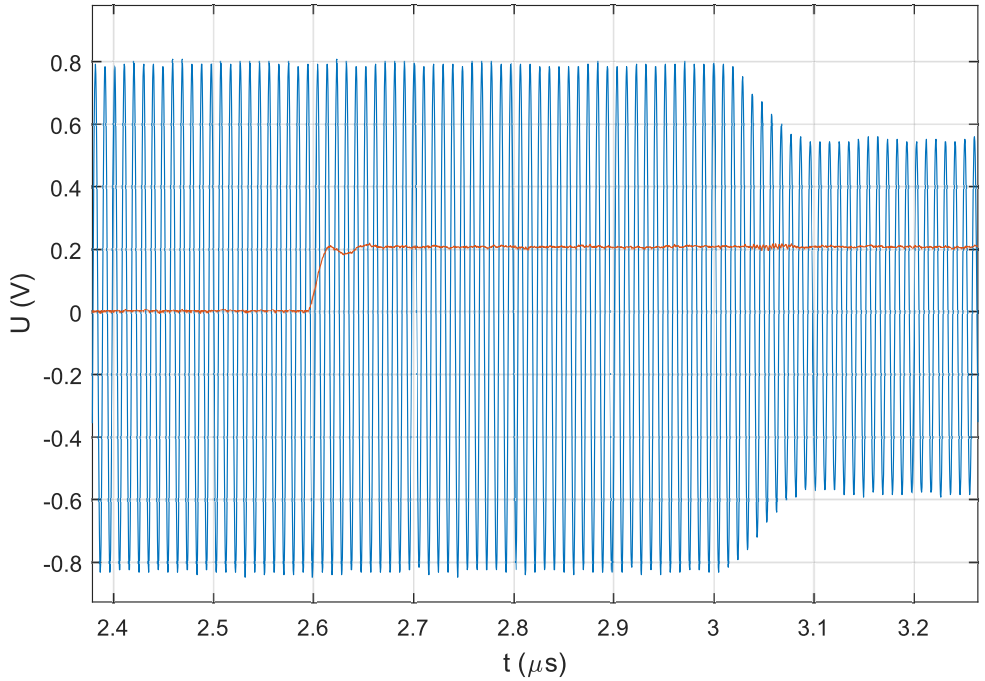


Figure 4.7: Response of the PID loop on the DDS cards to a square input signal to ADCs. The amplitude of the RF pulse starts to change after $T_{delay} = (433 \pm 8)$ ns

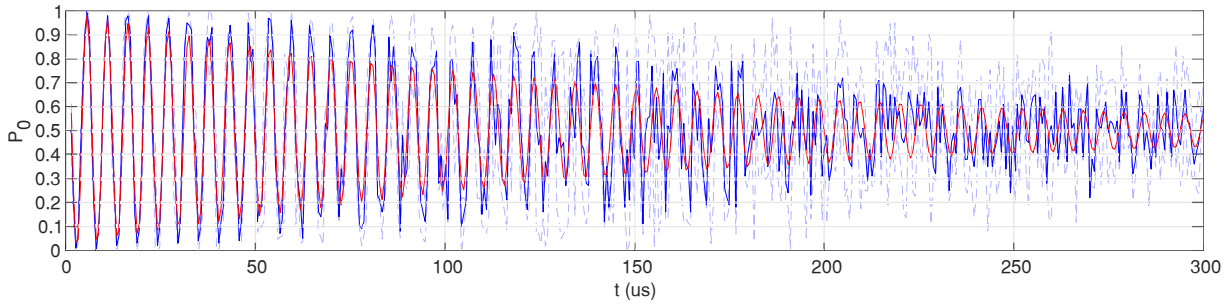
.

corresponds to voltage fluctuations of 0.22 mV or 2.7 LSBs. This causes changes of the VGA scaling of 0.13 %. Additionally the ADC fluctuates by 3.6 LSBs. This totals to an uncertainty of 4.3 LSBs RMS at the output of the PID loop which equals a maximum precision of 0.2 % of the amplified signal. This creates a lower bound of the relative standard deviation of the Rabi frequency.

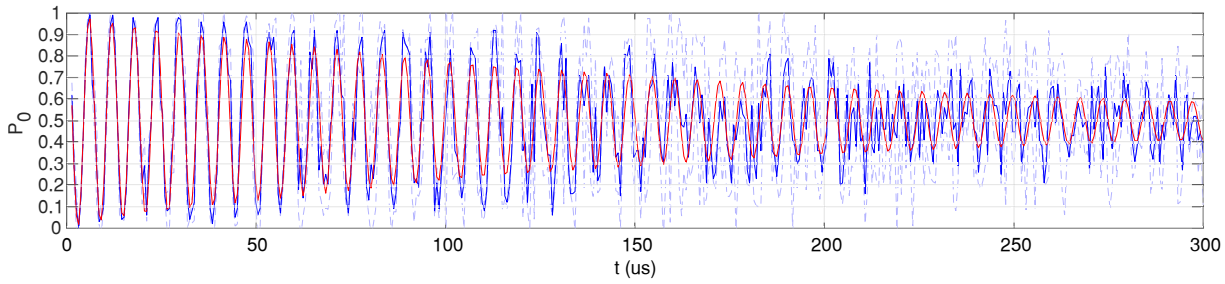
The performance of the described stabilisation algorithm was put to test by comparing Rabi flops with the PID loop inactive and active (see Fig. 4.8). The intensity noise does not increase the contrast over time significantly.

To create a scenario which allow the feedback loop to show its abilities, the intensity lock of the tapered amplifier of the 729 nm laser was released and artificially ramped at its maximum frequency of 500 Hz and a large amplitude. This results in a strong low frequency intensity noise which can easily be stabilised by the integrator term of the PID loop alone. In Fig. 4.9 a rapid decay of contrast is visible without intra-sequence stabilisation. If the feedback loop is active, the Rabi flops are unaffected by the artificially introduced noise.

Fig. 4.10 shows how the gain of the VGA is changed to counteract the varying intensity from the tapered amplifier.



(a) without stabilisation. $\sigma_{\Omega_0} = 0.060 \pm 0.003$



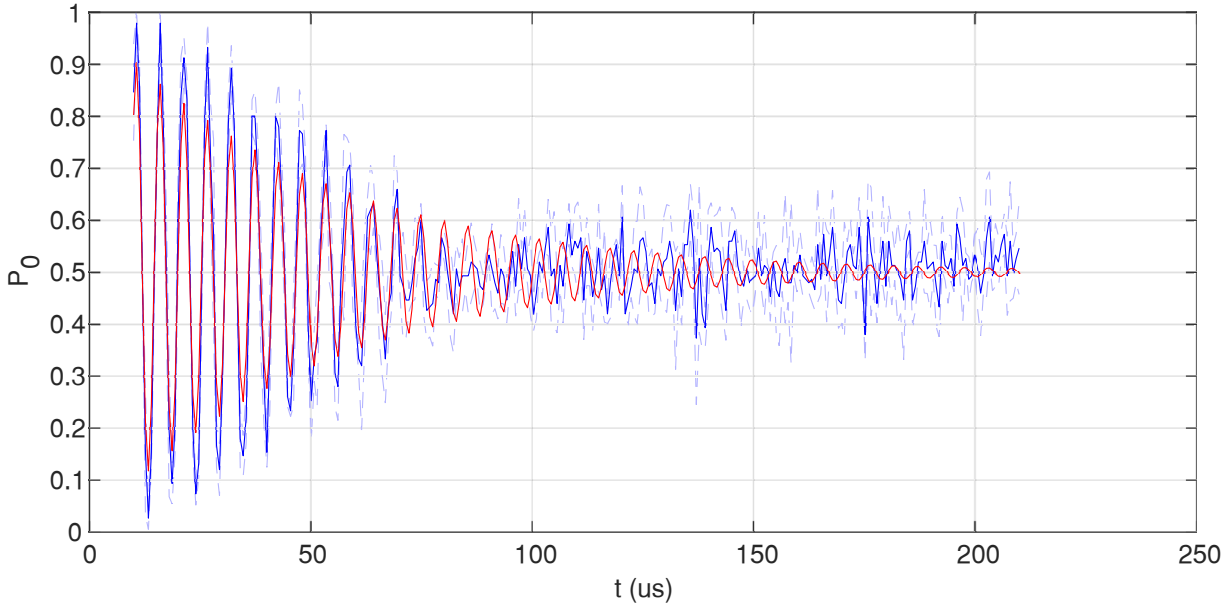
(b) with stabilisation. $\sigma_{\Omega_0} = 0.059 \pm 0.003$

Figure 4.8: Rabi flops of Ca fitted with the function described in Sec. 1.4.2.

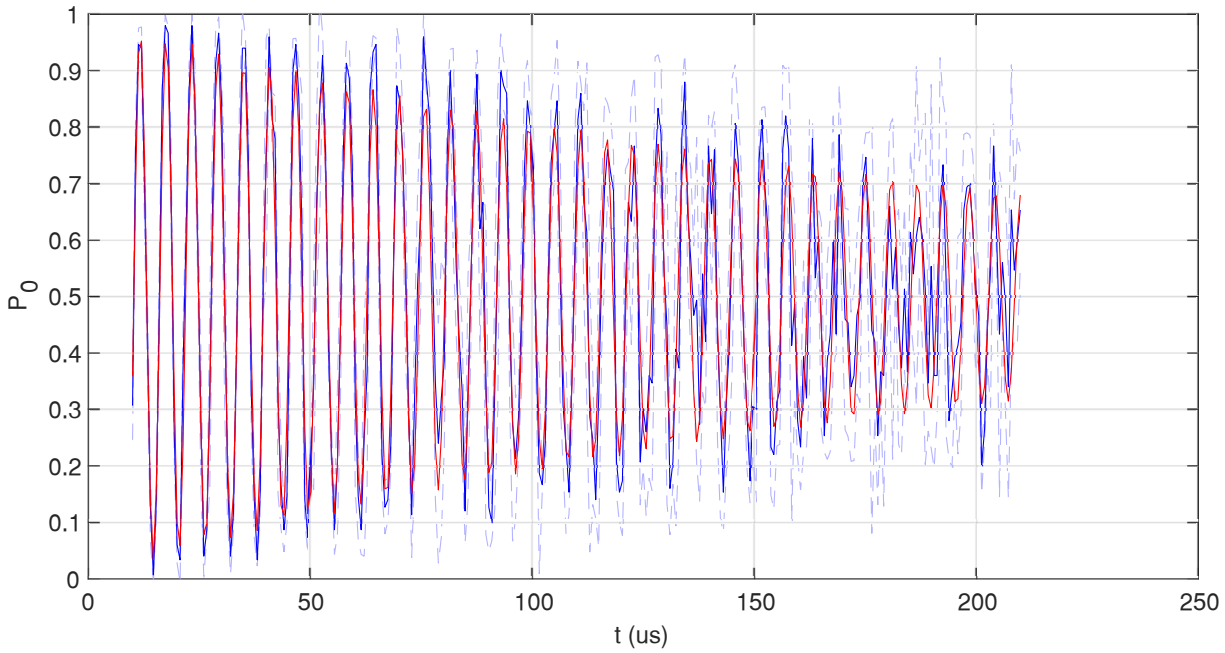
A randomised benchmarking scan over 300 randomisation indices and 3 to 183 gates in a step size of 12 was conducted. No significant difference in success probability between the stabilised and non-stabilised runs could be observed (see Fig. 4.11).

To demonstrate the capabilities of the new FIFO flow control, randomised benchmarking could be performed with up to 201 gates instead of 183. Slightly higher numbers could be achieved by using longer padding pulses in the beginning of a sequence which would allow the Zedboard to fill the FIFOs before the actual randomised benchmarking begins. In Fig. 4.12 the results of a longer sequence with 30 randomisation indices are shown. The sequence was optimised to reduce the number of backplane transmission per rotation from 6 to 5 without using stabilisation. As the setpoint has to be transmitted for every pulse as well, the maximum number of gates is lower if intra-sequence stabilisation is used.

The focus of the described experiments was to provide a proof of concept. Intra-sequence stabilisation is successfully performed. However, the intensity noise levels of the lasers were not characterised and it is unclear if the feedback loop is able to attain close to optimal noise reduction. It is important to reduce the frequency of intensity noise seen by the PID controller to a level that can be counteracted. Otherwise, the P and D terms need to operate at reduced gains to avoid amplifying noise instead of reducing it.



(a) without stabilisation. $\sigma_{\Omega_0} = 0.104 \pm 0.004$



(b) with stabilisation. $\sigma_{\Omega_0} = 0.053 \pm 0.002$

Figure 4.9: Rabi flops of Ca while artificially introducing intensity noise fitted with the function described in Sec. 1.4.2.

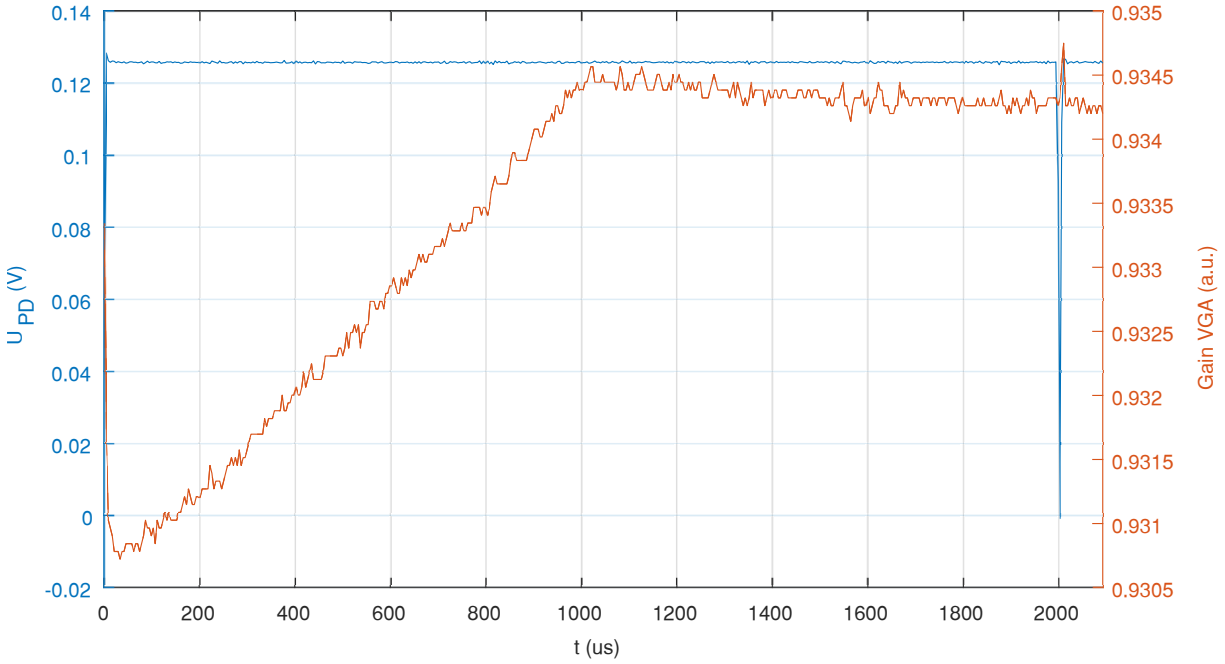


Figure 4.10: PID loop compensating the artificially added slow amplitude noise. The output of the loop is normalised with respect to its maximum achievable value.

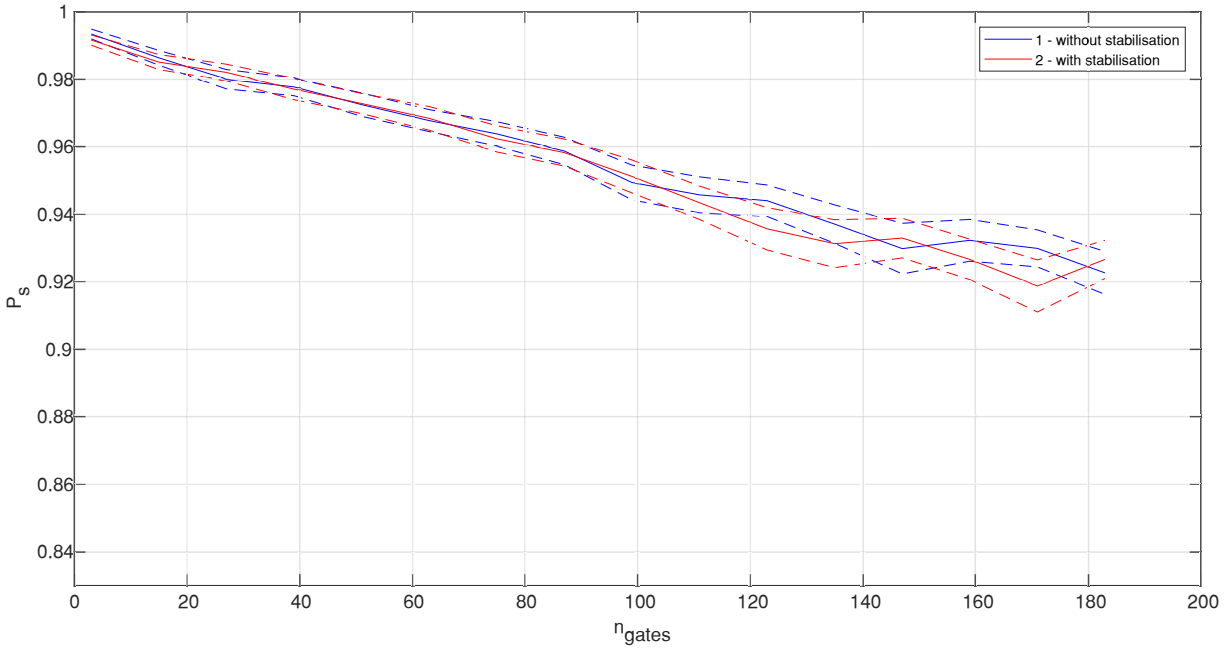


Figure 4.11: Randomised benchmarking of Ca over 300 randomised sequences. 95 % confidence intervals are indicated.

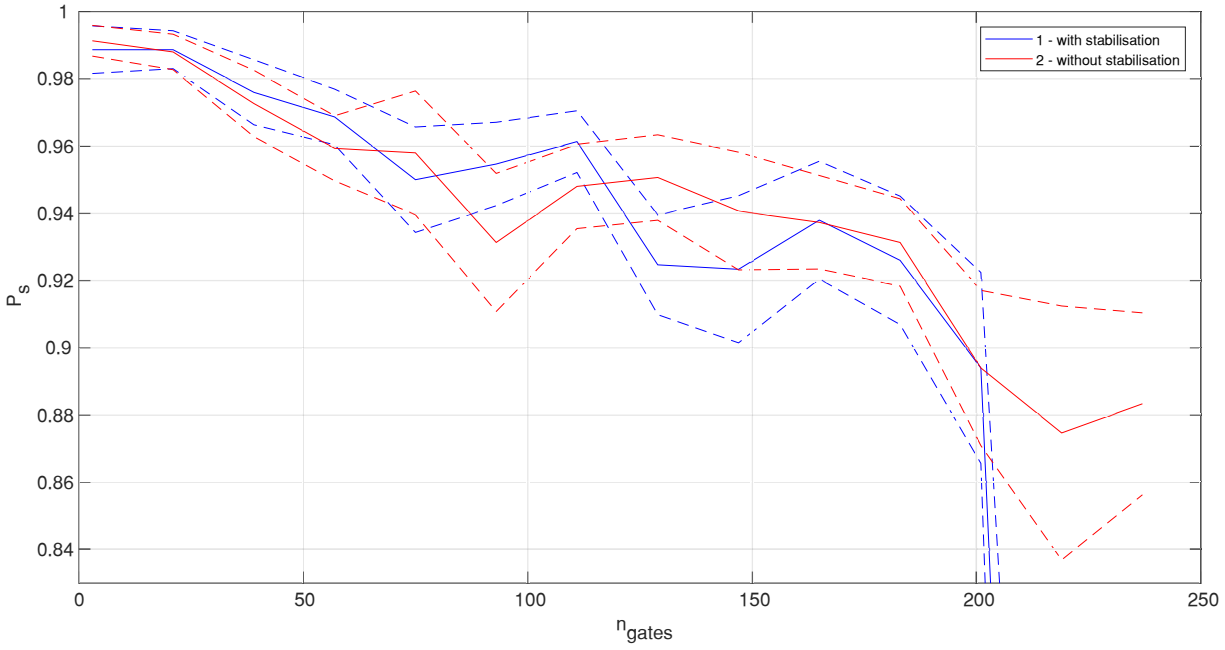


Figure 4.12: Randomised benchmarking of Ca over 30 randomised sequences making use of the FIFO flow control. 95 % confidence intervals are indicated. Due to an extra instruction that needs to be sent per loop when the amplitude is stabilised, the experimental loop starts to crash earlier. This causes the drop of the blue curve at 200 gates.

Chapter 5

Conclusion

In order to achieve very high gate fidelities in trapped-ion experiments the amplitudes of the lasers interacting with ions need to be as stable as possible. Unavoidable parameter drifts of optical elements in the beam path need to be actively counteracted. Intensity stabilisation through a feedback loop using the signal of a photodiode monitoring the laser beam after it has passed the ion trap is an effective way to do so. Stabilising the amplitude between sequences is sufficient to reduce slow drifts and thus the required frequency of manual recalibration. The duty cycle of the AOMs responsible for switching laser beamd incident on the ions cause fluctuations which are on the timescale of pulse lengths. This can be addressed by integrating a feedback loop at a lower level close to the RF generation which controls the AOMs.

The control system used in the TIQI group is a powerful tool to conduct advanced quantum information processing experiments. As it is not an off-the-shelf solution, a lot of low-level firmware design has to be done by the TIQI group. This allows the system to be tailored to the needs of trapped-ion experiments and made it possible to integrate amplitude stabilisation. However, the custom firmware that runs on each component of the system is difficult to change, as low-level design has a high learning curve and documentation is still rare. Nevertheless, the final result is flexible at a higher level and allows implementation of novel experimental sequences without much effort. Since a lot of features were added as they were needed, some parts of the source code lack a clear structure which makes it hard for newcomers to understand.

5.1 Summary

The existing control system was successfully extended to be able to perform inter-sequence stabilisation. In Ch. 3 the necessary changes and results are presented. The originally envisioned use-case of stabilising intensities during a day or week might be possible.

The written software allows additional monitoring of other slow analog signals which could be used to log a standalone PID loop or temperature sensors and notify the user when the input leaves the allowed range.
The XADC together with the developed software is a convenient tool to stabilise slow drifts.

The firmware of the DDS cards and the Zedboard was modified to make use of the ADC on the new revision DDS cards. The backplane communication was greatly improved to be more robust in the case of bit flips and to be capable of sending data from the DDS cards to the Zedboard as well.
A versatile capturing tool was created which allows monitoring of data streams on the DDS cards synchronously with a running pulse sequence.
The module which sends data to the DACs which control the VGAs was extended with a fast, feature-rich PID loop that utilises the analog data from ADCs for stabilisation.

The mentioned changes work in practice but still need a bit of improvement to be easier to use. The conducted experiments barely go beyond a proof of concept and the optimal parameters and use cases still have to be explored.

5.2 Outlook

Further tests need to be done to explore the full potential of the implemented changes. The lasers needed to address the Beryllium ion suffer from stronger intensity noise and drifts. Stabilising the intensity after the laser passed the ion might yield a measurable improvement of the fidelities.

The signal-conditioning board for the XADC would offer a higher stability if it was placed in a more temperature stable environment. Currently it is exposed to hot airflows from other electronic equipment.

To overcome bandwidth limitations from the processing system to the Programmable Logic (PL), the communication with `hiway` and `pulser` could be implemented through Direct Memory Access (DMA). In this case, the CPU cores are not directly involved in sending and receiving data from the PL but the memory controller handles direct writes to and reads from the main RAM used by the CPU.

In the future, the backplane speed will have to increase by using specialised transceiver modules. This would allow the execution of very long experimental sequences even with the short pulses used to address Calcium ions.

The filter in the DDS PID loop could be extended to a higher order to reduce the response to high frequency noise. A configurable notch filter which only cancels out certain frequencies might

also improve performance. This could be useful to remove electric signals induced by RF fields present in the setup.

The interface of the intra-sequence stabilisation needs to enhanced significantly to make it more user-friendly and allow intuitive tuning of parameters.

For several reasons, trapped-ion manipulation is currently done with pulses that have a clear separation (times of zero laser amplitude) between each other, one of them being the warming up of the AOM during intense duty cycles. The intra-sequence stabilisation can be used to reduce the effects of temperature changes within the AOM. By removing the wait times between pulses, sequences for calcium can be sped up by at least 50 to 100 % depending on the ratio between π to $\pi/2$ pulses. This would reduce the effects of spontaneous decay, heat-up of the ion during the sequence and parameter drifts.

Appendix A

Detailed FPGA documentation

A.1 AXI

AXI is short for Advanced eXtensible Interface which is a bus standard specified by ARM.

This bus follows a master-slave principle. There can be several slaves attached to one master, which are connected through an AXI interconnect. Every slave is assigned a certain address range. According to the address set by the master, the interconnect connects it to the slave. There are separate data, address and control lines for read and write operations. The most important control lines are the "ready" and "valid" signals. The receiver (which can be the master in the case of read operations) sets the "ready" flag high as soon as it can receive. The sender then changes the corresponding address or data bus to the desired value and confirms it with the "valid" flag. For a transmission to be completed, this has to be done twice. First the master sets the write or read address and than either the master (write) or the slave (read) sends data and performs the described handshake again. There are extensions to this protocol which considerably speed up the process but to keep the Verilog code simple, the AXI lite implementation described was chosen which keeps only the most essential features. This way the slave exposes several registers which don't have to be the same for read and write operations. AXI usually uses a 32-bit data bus. The address bus width can vary. Currently, 11-bits are used. It's important to mention, that AXI uses a byte aligned address. Therefore, the 2 least significant bits of the address bus are essential irrelevant. Additionally, the address used in the PS is the same as the address used in the IP cores times 4 i.e. upshifted by two bits.

A.2 Bitumen

A.2.1 Old

`bitumen` splits 48 bit instructions into 4 chunks. Each 12 bit chunk of bitumen is framed in the following bit sequence: This is the same structure used for Universal Asynchronous Receiver

bit	14	13	12:1	1	0
function	stop (1)	parity	data	special word	start (0)

Transmitter (UART), which doesn't need a synchronised clock between sender and receiver. However, the approximate data rate (called Baudrate) needs to be known, to allow successful transmissions. As 1 is sent when no transmission is taking place, the start bit (0) will mark the start of a new transmission. According to the baudrate, the middle of each bit is chosen to determine it's value, as usually the transmission line is sampled much faster than the baudrate. Alternatively, majority vote can be used as well. For UART, usually 8 to 10 data bits are used and the parity bit is optional.

`bitumen_slave` receives a transmission with the corresponding module `bit_rx` and sends an answer message containing the running XOR checksum of the received chunk. `bitumen` keeps it's own running checksum and compares the received message. If an error is detected, it will be passed on to the `hiway` module, which manages communication between the main program and the DDS cards. After 4 12 bit chunks are sent to the slave, an empty special word is sent and the slave responds with a special word containing errors that might have occurred during the transmission and 8 error bits supplied by the rest of the firmware.

Instruction words are split into chunks to reduce latency and allow higher throughput. The transmission is completed once `bitumen` receives the special word reply and the next transmission is started. If first data was sent in one piece and then a reply was sent one of lines would always be idle. This way, both lines are used at the same time.

A.2.2 New

The serialising structure was change to allow for dynamic transmission lengths. Depending on

bit width	1	8	16 - 59	2	1
function	stop (1)	CRC	data	word type	start (0)

the word type, a transmission can be a data transmission or a shorter response transmission. Data transmissions consist of the payload which is 56 bit for Zedboard to DDS card and 34 bits for DDS card to Zedboard transmissions, and the packet counter which is 3 bits long currently.

Since the data input is buffered in a small FIFO anyway, the pointer to the outgoing data of the FIFO is used as the counter.

A.3 Cyclic redundancy check

At it's core a polynom division of the data is performed. The remainder is used as the CRC sum and appended to the data when send. The check involves performing the CRC calculation again and comparing the remainder to zero. The CRC polynom has to be chosen carefully. A good choice for 8-bits is CRC-8 AUTOSAR, which is used in the automotive industry.

A.4 XADC sampling details

The XADC maximum internal clock frequency is 24 MHz which might be synthesised by dividing the supplied input clock. A single measurement then consists of setting the desired mux address, which connects the channel to the ADC. The internal track-and-hold amplifier will acquire the applied voltage level. When the conversion start (CONVST) trigger pulse is received, the input capacitor which is was charged to the input voltage level by the amplifier is detached from the input and the ADC starts converting the voltage into bits. The 12 bit result is then padded to 16 bits which increases precision in the case of internal averaging which might be used.

A.5 New instruction word for DDS cards

Tables containing the new structure can be found on the next page. The channel mask is used to target the BRAMs of the submodels associated with the corresponding channels on the DDS cards. If an empty channel mask is selected, the instruction word targets `odecoder` and `oencoder` configuration.

Figure A.1: New instruction word with channel mask not equal to zero

55:52	51	50:46		45:37							36:0						
word_page	req	dds_ch		cfg							bram_inst						
	51	50	49:46	45	44	43	42	41:39	38	37	36	35	34:32	31	30:28	27:18	17:0
0000	req_odec_status	dds_ch_mask_valid	dds_ch_mask	cfg_valid	cfg_trig	cfg_idec_rst	cfg_mtim_rst	empty	cfg_dds_ext_pwr	cfg_dds_mstr_rst	Bram_inst_valid	bram_sel_high_valid	bram_sel_high	bram_fifo_inst	bram_sel_low	bram_addr	bram_data

Figure A.2: New instruction word with channel mask equal to zero

55:52	51	50:46		45:37							36:32			31:0								
word_page	req_odec_status	dds_ch		odec and adc configuration							oenc_cfg			oenc_buffer_cfg								
55:52	51	50	49:46	45	44	43	41	40	39	38	37	36	35	34:32	31	30:27	26:23	22	21:18	17:14	13:10	9:6	5:3	2:0
0000	req_odec_status	dds_ch_mask_valid	0000	odec_cfg_valid	odec_stat_report_full	odec_stat_report_emtpy	adc_mstr_rst	duty cycle stabiliser	adc_a_pwr_off	adc_b_pwr_off	adc_output_disable	adc_cfg_valid	bc_ch_a_is_master	empty	adc_stream_avgs_valid	adc_a_avgs	adc_b_avgs	adc_trig_mask_valid	adc_a_trig_mask	adc_b_trig_mask	bc_a_src	bc_b_src	bc_a_mode	bc_b_mode

Bibliography

[1] Richard N Foster. *Innovation: The attacker's advantage*. Summit books, 1988.

[2] James Meindl. Special issue on limits of semiconductor technology. *Proceedings of the IEEE*, 89(3):223–226, 2001.

[3] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982. ISSN 1572-9575. doi: 10.1007/BF02650179. URL https://doi.org/10.1007/BF02650179.

[4] Richard Haughton. Race for quantum supremacy hits theoretical quagmire. *nature*. doi: 10.1038/nature.2017.22993. URL https://doi.org/10.1038/nature.2017.22993.

[5] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999. doi: 10.1137/S0036144598347011. URL https://doi.org/10.1137/S0036144598347011.

[6] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011. ISBN 1107002176, 9781107002173.

[7] Florian Leupold. *Bang-bang Control of a Trapped-Ion Oscillator*. PhD thesis, ETH Zürich, 2015.

[8] Emanuel Knill, D Leibfried, R Reichle, J Britton, RB Blakestad, John D Jost, C Langer, R Ozeri, Signe Seidelin, and David J Wineland. Randomized benchmarking of quantum gates. *Physical Review A*, 77(1):012307, 2008.

[9] Harrison Ball, Thomas M. Stace, Steven T. Flammia, and Michael J. Biercuk. Effect of noise correlations on randomized benchmarking. *Phys. Rev. A*, 93:022303, Feb 2016. doi: 10.1103/PhysRevA.93.022303. URL https://link.aps.org/doi/10.1103/PhysRevA.93.022303.

[10] Daniel Kienzler. *Quantum Harmonic Oscillator State Synthesis by Reservoir Engineering*. PhD thesis, ETH Zürich, 2015.

[11] Hsiang-Yu Lo. *Creation of Squeezed Schrödinger's Cat States in a Mixed-Species Ion Trap*. PhD thesis, ETH Zürich, 2015.

[12] Vlad Negnevitsky. *Feedback-stabilised quantum states in a mixed-species ion system*. PhD thesis, ETH Zürich, 6 2018.

[13] Matteo Marinelli. *Title Undecided*. PhD thesis, ETH Zürich, 2019. in preparation.

[14] C. V. Raman and N. S. Nagendra Nath. The diffraction of light by high frequency sound waves: Part iii. *Proceedings of the Indian Academy of Sciences - Section A*, 3(1):75–84, Jan 1936. ISSN 0370-0089. doi: 10.1007/BF03046238. URL https://doi.org/10.1007/BF03046238.

[15] B.E.A. Saleh and M.C. Teich. *Fundamentals of Photonics*. Wiley Series in Pure and Applied Optics. Wiley, 2007. ISBN 9780471358329. URL https://books.google.ch/books?id=Ve8eAQAAIAAJ.

[16] Amnon Yariv and Pochi Yeh. *Photonics: Optical Electronics in Modern Communications (The Oxford Series in Electrical and Computer Engineering)*. Oxford University Press, Inc., New York, NY, USA, 2006. ISBN 0195179463.

[17] Elizabeth A Donley, Thomas P Heavner, F Levi, MO Tataw, and Steven R Jefferts. Double-pass acousto-optic modulator system. *Review of Scientific Instruments*, 76(6):063112, 2005.

[18] *ZedBoard*. URL http://zedboard.org/sites/default/files/product_briefs/5066-PB-AES-Z7EV-7Z020-G-V3c%20%281%29_0.pdf.

[19] Ben Keitch, Vlad Negnevitsky, and Weida Zhang. Programmable and scalable radio-frequency pulse sequence generator for multi-qubit quantum information experiments. *arXiv:1710.04282 [quant-ph]*, 2017. URL https://arxiv.org/abs/1710.04282.

[20] Vlad Negnevitsky, Matteo Marinelli, Karan Mehta, Hsiang-Yu Lo, Christa Flühmann, and Jonathan P. Home. Repeated multi-qubit readout and feedback with a mixed-species trapped-ion register. *arXiv:1804.09703 [quant-ph]*, 2018.

[21] Pascal Engeler. Using the zynq-7000 xadc and signal pre-conditioning, 2017. ETH Zürich, Semester thesis.

[22] Martin Stadler. Adding a differential term to the evil, 2017. ETH Zürich, Semester thesis.

[23] Alexander Hungenberg. Automatic relocking of an fpga-based pid controller using a bandpass-filtering approach, 2013. ETH Zürich, Semester thesis.