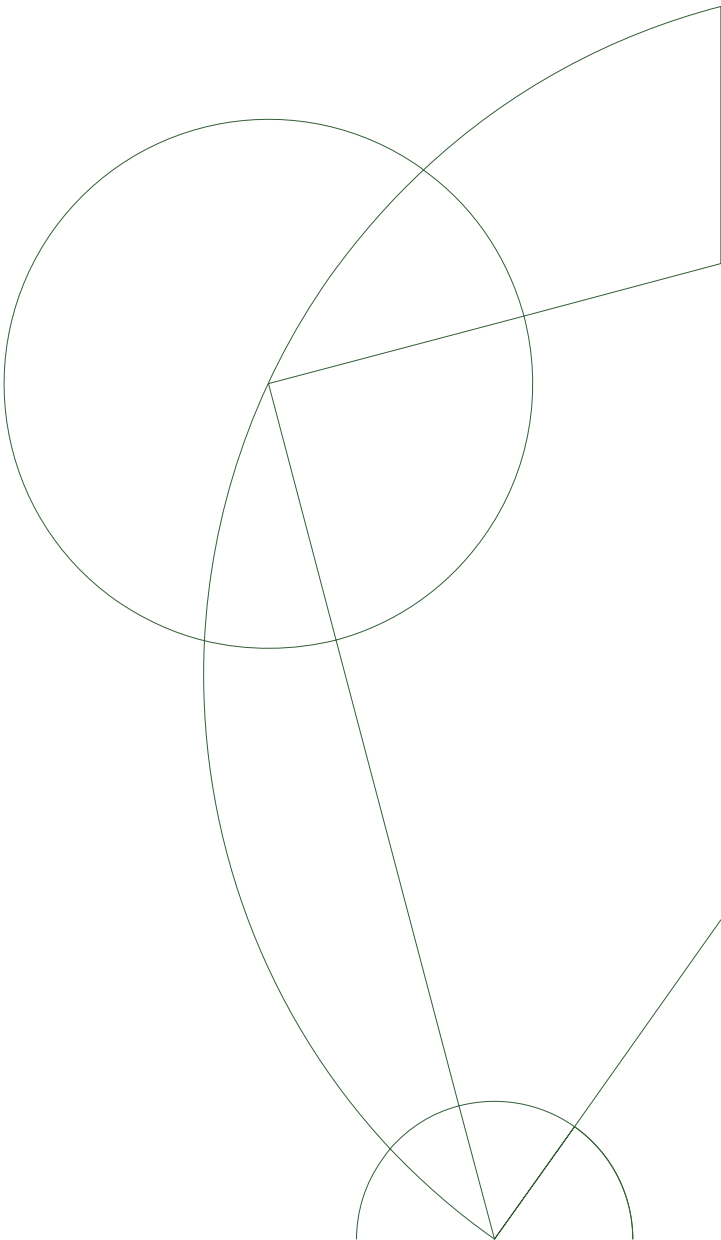
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Bachelors project

Rasmus Skytte Eriksen
zrm439@alumni.ku.dk

Improving coherence time by FPGA based feedback

Noise compensation in the resonant exchange-only qubit



Charles Marcus

June 11, 2014

Abstract

In this bachelors thesis I discussed how to use a fast feedback mechanism, implemented on an FPGA, to increase the coherence time of the resonant exchange only qubit. Using the fundamentals of the theory of feedback, I showed how a PID feedback controller could be implemented digitally using C on an FPGA and the limiting factors associated with the controller. Specifically it was found that the bit-depth of the digitization can be an important limiting factor as well as the speed the feedback controller can be operated. Despite being implemented on a fast platform, the feedback controller bandwidth was much smaller than the expected 200 kHz and found to be only on the order of 35 kHz, suggesting the need for further code optimization. I discussed how the nuclear and electrical noise in the environment effect the qubit and discussed how the feedback loop could be used to compensate the effects of the noise by feeding back on the depletion-gates defining the qubit. The conclusion was that rate the qubit could be read out is likely the limiting factor in terms of the bandwidth of the feedback loop, meaning that even a relatively slow FPGA can be used.

Contents

List of Figures

1 Introduction

Perhaps the most significant technological movement in the past century has been development of the information technology. A technology which is being integrated increasingly more intimately in our lives. Besides the social aspects, the unprecedented gathering of information and data-analysis from all aspects of society has given significant advances in our knowledge of the world. The technology is however being pushed to its limits as transistors are made smaller and smaller making it harder and harder to comply with Moore's prediction on the growth of technology. We, in the scientific community, have the opportunity to further the movement despite this stagnation with the development of the quantum computer. From a physicists view it is however the opportunity to investigate the isolated quantum system, the propagation of quantum information, the ability to manipulate nature at the level of single electrons, which is the primary driving factor for such work, but the promising consequences of Shor's or Grover's algorithm justifies the development from the societies point of view.

There are many suitable proposals for the essential building block of the quantum computer: the 'qubit'. A 'Qubit' is the shorthand term for a quantum-bit, which is the quantum computing analogous of the binary information bit used in classical computing. The advantage of the quantum bit is that the super-position property of quantum mechanics allow for a completely different concept of computing, opening the door for solving new types of problems.

The different proposals for the qubit has each its own advantages and disadvantages and no particular qubit has yet proven to be the obvious choice to form the basis of a quantum computer. At the Center for Quantum Devices of the Niels Bohr Institute where this thesis was written, we focus (among other qubits) on a qubit based on the spin of confined electrons, the so called spin-qubit, which is a promising candidate. Especially the resonant exchange-only qubit which has been the primary focus for this thesis. It is, like other qubits, not perfect and one disadvantages of this qubit is that the coherence time is currently shorter than what is required for doing fault-tolerant quantum computing, because of a de-phasing time on the order of ~ 25 ns.

The goal of this bachelors thesis is to employ a new way of combating decoherece, namely to use a fast feedback-circuit, implemented on a type of processor called an FPGA, to increase this coherence time of the resonant exchange qubit, improving its prospects for quantum computing.

In this bachelors thesis I have focused on describing how noise in the qubits environment effects the coherence time of our qubit, and on developing an FPGA based feedback system to extend the coherence time by measuring the energy splitting of the qubit-states and adjusting the manipulation of the qubit accordingly.

I begin this thesis with a description of control theory, the mathematical framework describing feedback and show how the use of feedback changes the systems dynamics. From here I consider how to choose a controller based on the system at hand and I show how a specific controller was implemented in practice. I then describe in simple terms the formation of the resonant exchange qubit and consider in detail how the noise

from different sources effect the qubits. The purpose of course is that by understanding effects of the the noise we can better correct for it using feedback. At the end of the thesis I discuss the different available choices of the feedback channel, i.e. the input of the system that we choose to let the feedback controller manipulate, along with the advantages and disadvantages of these channels.

The goal of testing the feedback-circuit on the resonant exchange qubit has turned out to be a too ambitious goal to achieve within the deadline of the thesis, since we have not been able to get a working qubit in time. I have therefore put extra effort into using what I have found to describe in principle how one can use the feedback mechanism to compensate for noise in the qubits environment and thereby extend the time in which the qubit retains phase-coherence.

2 Control theory

The concept of feedback is familiar to most physicists, but regardless I will start by going through the basics of control theory. It is, as the name implies, the mathematical framework to which the theory of feedback belongs. Books on control theory are mostly aimed towards mathematicians, but a mathematically rigorous description is not of interest here and I will instead follow the more physicist-friendly description of control theory given by Bechhoefer [1].
We start by considering the a dynamical system of the general form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.1a}$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \tag{2.1b}$$

The internal state of the system is represented by $\mathbf{x}$, the inputs to the system by $\mathbf{u}$, the measurable output of the system by $\mathbf{y}$, while $\mathbf{f}$ is used to represent non-linear system dynamics and $\mathbf{g}$ relates the output to the internal state. To further analyse this general dynamical system, it is beneficial to switch to the frequency-domain which in control theory is done by means of the Laplace transformation:

$$\mathcal{L}[f(t)] = \int f(t)e^{ist}\mathrm{d}s \equiv f(s) \tag{2.2}$$

The motivation for switching to frequency-domain, rather than using time-domain, is that in order to calculate the response of a system to an input we need to convolve the input with the system, and convolutions are, by the convolution theorem, simply multiplications in frequency-domain. In frequency-domain the analysis can be simplified further by considering the transfer function $G(s)$ which captures the systems observable dynamics in terms of the system input, but the system remains subject to the unknowable internal state $\mathbf{x}$:

$$G(s) \equiv \frac{y(s)}{u(s)} \tag{2.3}$$

Note that I write $G(s)$, $y(s)$ and $u(s)$ explicitly in the Laplace coordinate s to underline the change to Laplace domain. The transfer function is in general a complex function, and useful information about the dynamics of the control loop may be exctracted from the transfer function by plotting its magnitude $|G(i\omega)|$ and argument $\arg(G(i\omega))$ along the frequency axis ($s = i\omega$). These type of plots are in control theory called Bode plots and are used to characterize the system response. Alternatively, if the system dynamics are unknown, one can infer information about the dynamics of a system by taking Bode plots of the system and creating a model of the system to match these dynamics. Such model-building is needed if one wishes to have optimal control of the system but if the dynamics are not easily measured or are prone to changing, then a more general approach is preferable.

2.1 Feedback

Our ultimate goal is to stabilize such a dynamical system and this in done by utilizatig a feedback mechanism, that is we alter the dynamics of the system by feeding information of the output $\mathbf{y}$ back to the system via the inputs $\mathbf{u}$. To understand this process we will briefly consider the terms *open-loop*- and *closed-loop*-dynamics. Open-loop dynamics is simply the situation described in the (2.1); where the internal state of the system is allowed to evolve freely under the influence of $\mathbf{u}$. Feedback, on the other hand, is the process of *closing the loop*, whereby a situation is created in which the system inputs $\mathbf{u}$ are made dependent on the system output s$\mathbf{y}$, i.e. by letting $\mathbf{u} \to \mathbf{u}(\mathbf{y})$. To see how this alters the system dynamics we briefly consider the block diagram in figure 1. In this closed loop configuration the system output is measured by a feedback controller and is compared to some reference level $r(s)$. The difference between the reference level and the measured signal $\tilde{y}(s)$ forms the error signal $e(s)$. This error signal forms the basis of the feedback controllers response which is calculated by the control law $K(s)$ and leads to change in system input: $u(s) = K(s)e(s)$, by channg the system input it effectively completes the transformation $\mathbf{u} \to \mathbf{u}(\mathbf{y})$.

Keep in mind the principal distinction between the ideal system output $y(s)$ and the measurable system output $\tilde{y}(s) = H(s)[y(s)+d(s)]+\xi(s)$ which is subject to disturbances $d(s)$ and sensor noise $\xi(s)$. The sensor mechanics $H(s)$ are also explicitly written, since there for any measurement are some dynamics present.

In the case of our set-up, we read in the system output electronically via an analog-to-digital converter and the signal is low-pass filtered prior to sampling (with a 3 dB cut-off frequency ~ 178 kHz). This means that the sensor have the associated dynamics:

$$H_0(s) = \left.\frac{1}{1 + s/\omega_0}\right|_{\omega_0 = 2\pi \times 178 \text{ kHz}} \tag{2.4}$$

Furthermore, when outputting the a signal from the feedback controller via a digital-to-analog converter, there is inevitably, even for fast computations, added delay-dynamics to the system since the converter is a *zero-order-hold* digitizer, which leads to an effective

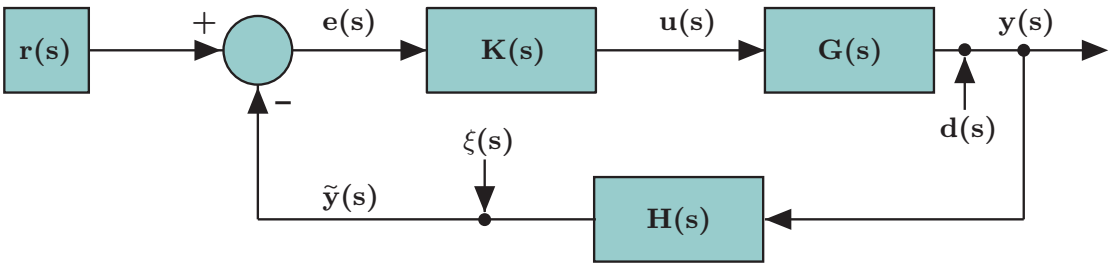


Figure 1: *Closed-loop dynamics. The output $y(s)$ of the system $G(s)$ is measured by a sensor of associated dynamics $H(s)$. This output is subtracted from a reference signal $r(s)$ to create the error signal $e(s)$. The control dynamics $K(s)$ then generate the response based on the error signal and modifies the system dynamics by changing the system inputs $u(s)$. In this model of the closed loop, disturbances $d(s)$ in the output and noise $\xi(s)$ from the sensor has been added as well.*

delay in the signal by half the operation time of the digital-to-analog converter:

$$H_1(s) = \exp\left(-\frac{sT_S}{2}\right) \tag{2.5}$$

Where T_S is the sampling time. We will see later that the bandwidth of the feedback loop is not negligible and that the sensor dynamics are an important part of the feedback loop dynamics.
The dynamics in this closed-loop configuration are found by considering the system output:

$$y(s) = K(s)G(s)e(s) + d(s) \tag{2.6}$$

and the error signal:

$$e(s) = r(s) - H(s)y(s) - \xi(s) \tag{2.7}$$

When the equations are solved with respect to $y(s)$ the final form of the system output, under the influence of the feedback loop, is found as given below:

$$y(s) = \frac{K(s)G(s)}{1 + K(s)G(s)H(s)}\left[r(s) - \xi(s)\right] + \frac{1}{1 + K(s)G(s)H(s)}d(s) \tag{2.8}$$

This equation, if studied in more detail, shows us some of the limitations of the feedback loop as well as some of the benefits of feedback control. For instance the case of a bad controller can be identified, that is if $K(s)G(s)H(s)|_{s=i\omega} \to -1$, since the system response will diverge leading to instability.

Rather than continuing such analysis, we will for now focus on how noise and disturbances enter into the feedback loop. We briefly consider the *tracking error* $e_0(s)$ (while for the moment assuming no sensor dynamics, $H(s) = 1$).

$$\begin{aligned} e_0(s) \equiv r(s) - y(s) &= \frac{K(s)G(s)}{1 + K(s)G(s)}\xi(s) + \frac{r(s) - d(s)}{1 + K(s)G(s)} \\ &= T(s)\xi(s) + \left[1 - T(s)\right]\left[r(s) - d(s)\right] \end{aligned} \tag{2.9}$$

Where the closed-loop-dynamics T was introduced in the last step:

$$T(s) = \frac{K(s)G(s)}{1 + K(s)G(s)} \tag{2.10}$$

This tracking error is the deviation of the actual system output from the reference signal, as opposed to the error signal $e(s) = r(s) - y(s) - \xi(s)$, which is the difference between the measured output signal and the reference. This equation can help in defining the property of a good controller, that is to have the tracking error be small compared to the disturbances and noise in the system. Furthermore, the trade-off between the effect of noise and disturbances are made clear: for small T, disturbances is the major contribution to the tracking error, while for large T, the sensor noise plays the dominant role.

2.2 Digital controller

With our knowledge of how the feedback mechanism alters the system dynamics the difficult part is now to find a suitable form of the feedback controller $K(s)$ such that the system is both stable and robust. If the goal is to have optimal control then the principle of loop-shaping described in detail by Bechhoefer [1] is a promising method where one tailors the feedback controller to the dynamics of the system. The downside is that the dynamics can be difficult to probe and can change over time making it a cumbersome method. I have, for the time being, limited my focus to the common choice for the controller known as the *Proportional-Integral-Derivative* controller (PID-controller). This controller is chosen because of its widespread usage, the amount of literature on the controller, and because of its prospect to do automatic tuning of the control parameters by stability tests (e.g. Relay feedback test or a Ziegler-Nichols test)[1]. Finally, the PID controller is chosen due to its simple implementation as we will see below.
Intially the PID controller is written in time-domain as the following:

$$K(t) = K_P\, e(t) + K_I \int_0^t e(t')\mathrm{d}t' + K_D \left.\frac{\mathrm{d}e(t')}{\mathrm{d}t'}\right|_{t'=t} \tag{2.11}$$

In (2.11) we identify each of the terms as corresponding to either the proportional, integral, or derivative term each with associated parameters K_j. To implement the PID controller as a digital controller we must discretize the equation and will for that purpose write the integral as a Riemann sum and adopt the approximation $\left.\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right|_{t=t_n} \approx \frac{e_n - e_{n-1}}{T_S}$, with T_S being the time between samples.

$$K_n = K_P\, e_n + K_I T_S \sum_n e_n + K_D \frac{e_n - e_{n-1}}{T_S} \tag{2.12}$$

Equation (2.12) can be written in the more common form be rewriting it in terms of a common gain K_C. When writing the PID controller in the common form it is usually

[1] The interested reader could read *Non-parametric Tuning of PID Controllers* by Boiko, I.

done with the simultaneous change to the variables $T_I = \frac{K_I T_S}{K_C}$ and $\frac{1}{T_D} = \frac{K_D}{T_S K_C}$. These variables serve as the tuning parameters for the integral part and the derivative part respectively.

$$K_n = K_C \left(e_n + T_I \sum_n e_n + \frac{1}{T_D} (e_n - e_{n-1}) \right) \tag{2.13}$$

This form of the PID controller may now be readily implemented into the digital controller (see Appendix D for implementation). I have also protected the controller against the problem of *Integral Wind-up*; in the event of the control signal saturating, due to the controllers finite output capability, the integral term of the controller diverges without reflecting the control dynamics. This is prevented by freezing the integral term while the controller is saturated.

3 Feedback equipment

The *Field-Programmable-Gate-Array*, or FPGA for short, was chosen to serve as the platform to implement our feedback controller on. I therefore start this section by introducing the principles of the FPGA and the equipment used to implement the digital controller. I then, describe a testing circuit devised to simulate the conditions of the readout-circuit used in the qubit experiment.

3.1 How an FPGA works

The FPGA consists of an *array* of configurable logic blocks, each block consisting of 4 Look-Up-Tables, 1 D Flip Flop and a multiplexer[2]. Every computational operation can be written in terms of boolean logic tables, that is for each configuration of the input, the boolean table has a corresponding output value which depends on the logical *gate* it represents. The Look-Up-Tables in the logical block implement these logical tables and the Flip Flop serves as a 1 bit storage, together with the multiplexer they form the basis needed for computation[2]. The FPGA is programmed by configuring the initial state of the logic block and this can be done by software, from where the first part of the name, *field-programmable*, hails. It is this flexible programming of FPGA which makes it such a useful device, since it in essence becomes programmable hardware having computational speeds that are comparable to an *Application-Specific-Integrated-Circuit* (ASIC)[3], these circuits are, as the name suggests, designed for a single set of operations (like a digital clock or a remote control). Of course the FPGA is slower than the ASIC, but it shows that the FPGA are capable of very high speed computation.

The configuration of an FPGA is programmed in either Verilog or VHDL, which are so called *Hardware Description Language*s (HDL) operating at a level of abstraction called the *Register Transfer Level* where one considers digital flow between hardware registers and logical operations on that flow. Alternatively a programming-language with

[2]The number of Look-Up-Tables and other contents of the configurable logic block will be different for a different model of the FPGA

a higher level abstraction can be used to program the FPGA; this code is then translated by a compiler to HDL and from there program the FPGA. This is the approach used for this thesis and as such our controller is implemented by use of C rather than use of HDL. The downside to this is that optimal performance from the FPGA cannot be expected since there is no low-level control of the HDL code[2]. Combining the fast computations of the FPGA with on-chip Analog-to-Digital and Digital-to-Analog converters allow for rapid control once the digital controller is implemented.

The FPGA based controller is implemented on an Altium Nanoboard 3000 which utilizes a Analog-to-Digital converter[3] (ADC), an Xilinx Spartan 3AN 1400K FPGA, and a Digital-to-Analog converter[4] (DAC) to allow for fast input-computation-output. By use of Altium Designer, the software handling the Nanoboard, I organized the internal routing of the ADC to FPGA and FPGA to DAC, and constructed a soft microprocessor (TSK-3000A[5]) which runs the feedback controller (See Appendix A for the FPGA design schematics). The limmiting factor on the Nanoboard is the sampling rate of the ADC which is limited to 200 kHz. When testing the effective rate at which the FPGA can respond by setting up a minimal working example (see Appendix C) it was found that not only was the ADC sampling at around 60% of the expected rate ($\sim$ 115 kHz), but also that the DAC was limited to the same writing frequency, meaning that a read-write cycle could only be carried out at a rate of $\sim$ 56 kHz in the current configuration. This rate is then an upper-bound of the rate our PID controller will operate at.

3.2 Testing the digital controller

To test the implementation of the PID controller on the Nanoboard I created small readout-circuit that simulates the readout of qubit experiments and then had the FPGA control the resonances of a test-circuit (see figure 2 a). In this scenario, the FPGA controls the bias voltage of a varactor, i.e. a tunable capacitor, of a driven LC circuit. The resonance of this LC circuit is thus controlled by the FPGA. In figure 3 the magnitude of the reflected signal as a function of frequency and varactor bias is shown. By driving the circuit at a frequency on the slope of a resonance (Indicated by the $\star$) the FPGA can change the bias on the varactor to keep the amplitude of the reflected signal constant. The reflection is being read out by means of homodyne detection: A HP8648A signal generator is driving the test-circuit at the given frequency and the reflected signal is mixed with the driving signal before being heavily low-pass filtered. The resulting DC signal is amplified and read in by the FPGA (see figure 2 b). This readout circuit is similar in function to the readout circuit of the experiments on the qubit (see Appendix F). The readout-circuit used in the qubit experiments is more advanced than this testing readout-circuit, but they utilize the same demodulation principle which makes the testing readout-circuit a useful development platform for testing the feedback controller.

[3] Texas Instruments, ADC084S021
[4] Texas Instruments, DAC084S085
[5] http://techdocs.altium.com/display/ADRR/TSK3000A

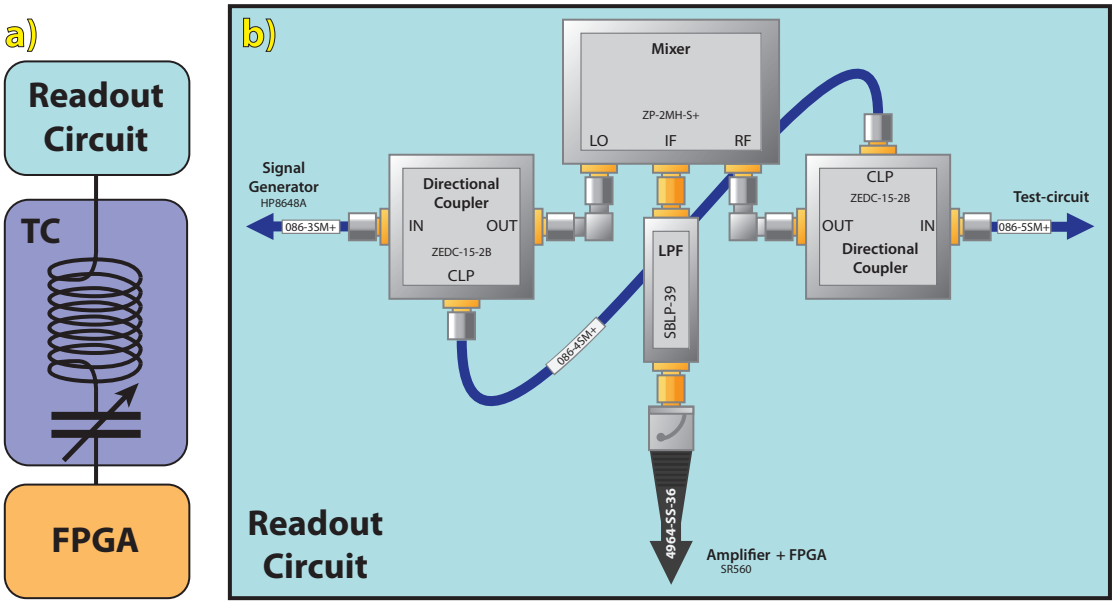


Figure 2: *Feedback test-circuit and readout.* **a)** *Overview of the test circuit (TC). The test-circuit consists of an inductor (*330 nH*) and a varactor which is controlled by the FPGA.* **b)** *Overview of the readout-circuit. The test-circuit is driven by a HP8648A signal generator and the reflected signal from the circuit is mixed down to DC by means of homedyne detection. Once the signal is demodulated it is amplified by an SR560 and sent to the FPGA. All parts are from Mini Circuit.*

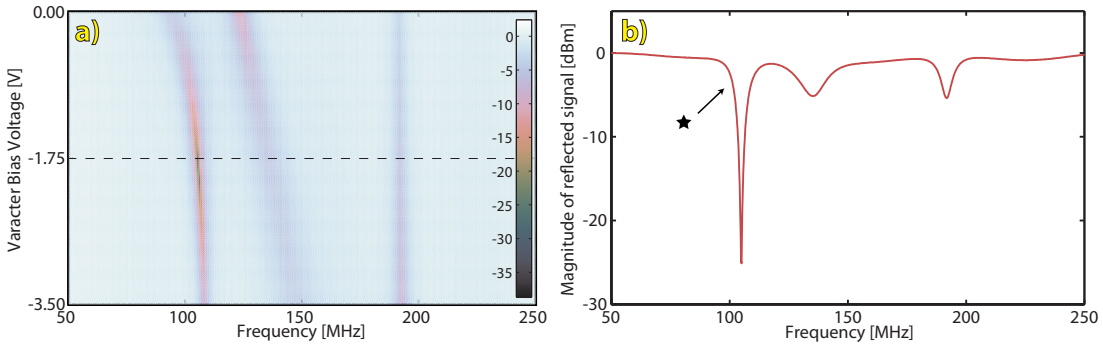


Figure 3: *Test-circuit resonances. The amplitude of the reflected signal from of the test-circuit is measured by a Network Analyser:* **a)** *A 2D image of the reflection magnitude (in* dBm*) plotted as a function of driving frequency and bias voltage on the varactor. The dashed line indicate the line cut shown in* **b)**. *The line cut for bias voltage* $V_B = 1.75$ V *showing the magnitude of the reflected signal with three resonances visible. The* $\star$ *indicates the chosen driving frequency used in testing.*

4 The Resonant Exchange Qubit

The system we want to use feedback on is the *Triple-Dot Resonant-Exchange Spin-Qubit*. To understand how noise effects this qubit, I begin by briefly describing the formation of quantum dots, i.e. an object which exhibit the three-dimensional confinement of electrons. From here we work our way to understanding the formation of the Triple-Dot and onwards to the formation of the qubit. I then describe the basics of the qubit and go on to describe in detail how noise stemming from different sources effect the qubit and how it leads to loss of coherence.

4.1 The Gallium-Arsenide Spin-Qubit

The spin-qubit is formed in a heterostructure of layered Gallium-Arsenide and Aluminium-Gallium-Arsenide (GaAs and $Al_{0.3}Ga_{0.7}As$). I will not go into detail of how this wafer is structured but rather simply sketch the most important features needed to understand the system. If we consider the energy levels along the z-axis of the GaAs wafer, the heterostructure has a single triangular well, which is formed at an interface 91 nm below the surface; this well dips below the chemical potential which is carefully adjusted to allow only one electron-band of the well to be filled and this leads to the formation of a *two-dimensional electron gas* (2DEG). A series of Titanium/Gold gates are then lithographically defined on the top of the wafer. By biasing these gates it is possible to locally deplete the 2DEG below, allowing for the confinement of electrons and thus the formation of quantum dots. The gates are distributed to create three adjacent dots which constitutes the *Triple-Dot* (see figure 4 a).

4.2 The Triple Quantum-Dot

With the electrons confined to two dimensions in the GaAs wafer, the gates are used to confine the electrons completely, allowing for full gate control of the dot size and the dot interactions. I will adopt the model shown in figure 4 b, that is I consider three spins confined on a line each feeling a Heisenberg interaction J_i with the neighbouring spins in the system. The potential is controlled by the gates for which I use the naming convention where the outer gates acts as *walls*, the three retracted gates act as left, right, and middle *plunger*, and the two middle non-retracted gates act as left and right tunnel *barrier*. The different gates have different effects on the qubit and ignorering the cross-coupling between the gates for the moment, their effects can be summarized: The walls of the triple-dot control the loading of electrons into the qubit, the barriers control the tunneling rate from the left and right dot into the middle dot, and the three plungers control the size of the three dots.

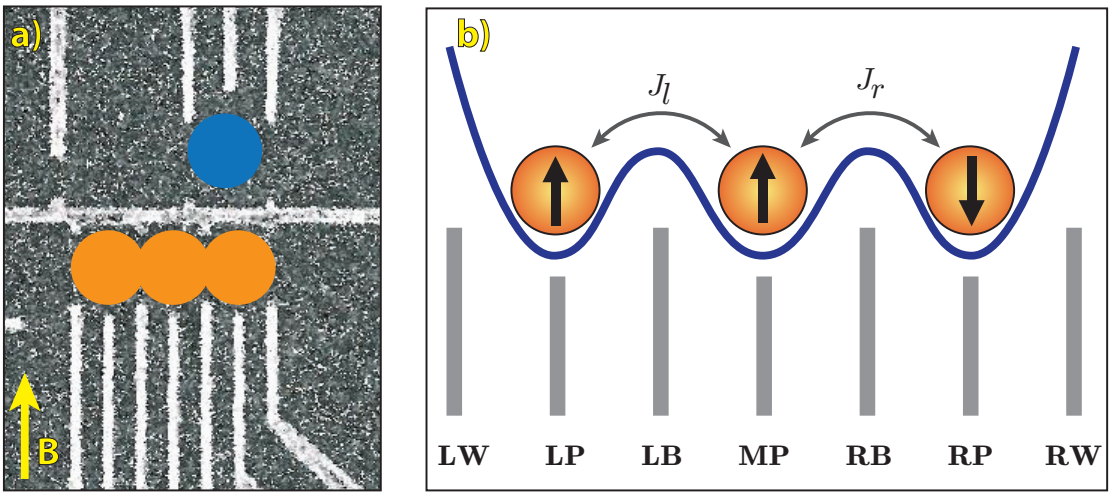


Figure 4: *Triple quantum-dot.* **a)** *A Scanning Electron Microscope picture of the device used in the experiments. The blue circle indicate the position of the sensor dot used for readout and the orange circles indicate the position of the dots. The magnetic field direction is indicated by the arrow. This device has been fabricated by Johannes Beil and Anton Kovyakh from the Center for Quantum Devices.* **b)** *The three dots of the qubit interact via the Heisenberg exchange J_i. In this ideal case, the electrons are confined in three dots based on the local electrostatic environment defined by the gates and as such the exchange interaction can be controlled by said gates. L = Left, M = Middle, R = Right, W = Wall, P = Plunger, B = Barrier.*

4.3 The Resonant-Exchange Qubit

With the triple-dot in place, it is now possible to construct the Resonant-Exchange Qubit. The description below is for the case where triple-dot is depleted to the few-electron regime such that only a single electron is occupying each dot. When the triple-dot is in the $(1,1,1)$ charge-state (where (l,m,r) is the charge occupancy in the left, middle, and right dot), the three electron spins in the triple-dot span an 8-dimensional state space, where the $(S=\frac{1}{2},\, S^z=\frac{1}{2})$ subspace is used as the logical encoding space for the qubit. In this subspace there are three states, $|Q\rangle$, $|0\rangle$, and $|1\rangle$, which are written in terms of the spin states $|S_n, S_l, S_m\rangle$ as:

$$|0\rangle = \frac{1}{\sqrt{6}}\left(|\uparrow\uparrow\downarrow\rangle + |\downarrow\uparrow\uparrow\rangle - 2\,|\uparrow\downarrow\uparrow\rangle\right) \tag{4.1a}$$

$$|1\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\uparrow\downarrow\rangle - |\downarrow\uparrow\uparrow\rangle\right) \tag{4.1b}$$

$$|Q\rangle = \frac{1}{\sqrt{3}}\left(|\uparrow\uparrow\downarrow\rangle + |\uparrow\downarrow\uparrow\rangle + |\downarrow\uparrow\uparrow\rangle\right) \tag{4.1c}$$

These three states are energetically separated from the remaining five states by a large Zeeman splitting to prevent leakage out of the $\left(S^z=\frac{1}{2}\right)$ sub-space. In this sub-space, the states $|0\rangle$ and $|1\rangle$ acts as the logical states and the $|Q\rangle$ state is only the remaining state that has not been energetically separated and this state is therefore a possible

leakage channel. Two-axis universal control around the Bloch sphere for the qubit is implemented by use of gate voltages only, i.e. we have complete electrical control of the two exchange couplings J_l and J_r which serve as the rotation axes in the qubit logical space (see figure 5 a). Rotation in the qubit space is controlled by detuning the energy levels of the left and right dot, and the moving of the qubit to the left and right on the detuning axis (see figure 5 b) shifts between rotations around J_l and J_r respectively.

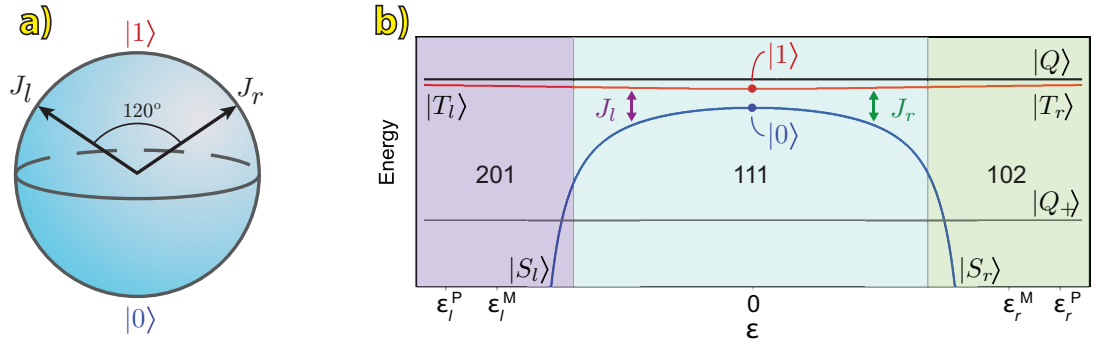


Figure 5: *The Resonant Exchange Qubit.* **a)** *A Bloch Sphere representation of the logical qubit space showing the two rotation axes J_l and J_r. Figure adapted from Medford et al. [4].* **b)** *The energy diagram of the Resonant Exchange Qubit as published by Medford et al. [4]. The figure depicts the energy splitting of the two logical states along a detuning axis ϵ, which cuts through the three charge states indicated by the coloured background. The qubit is operated around the center point of the $(1,1,1)$ region by pulsing along the detuning axis. During pulsing around the center-point the qubit rotates around J_r when it is detuned to the right and similarly rotates around J_l when detuned to the left. For operation at the center of the $(1,1,1)$ region the energy splitting is insensitive to first order noise in the detuning.*

4.4 Spin relaxation and qubit de-phasing

In order to have successful quantum computation there is at least one specific requirement that needs to be fulfilled; we require the ratio of the qubit coherence-time and the operation-time of the qubit to be large. Or to put it differently, in order to do fault-tolerant quantum computation the operations must be performed on the qubit much faster than the quantum state is destroyed by interactions with the environment.

This leads us to the core of this thesis: the extension of coherence-time for the resonant exchange qubit. To do this I will now investigate the mechanism leading to the decay of a spin quantum state.

In general there are two types of decay; first there is spin relaxation, which is the loss of a specific quantum state by the spins 'relaxing' from the excited state $|1\rangle$ to the energetically lower ground state $|0\rangle$ by emission of energy due to interactions with the environment, similarly a photon can excite the spins of the groundstate and cuase the loss of the quantum state. This type of relaxation decay is associated with the time scale T_1.

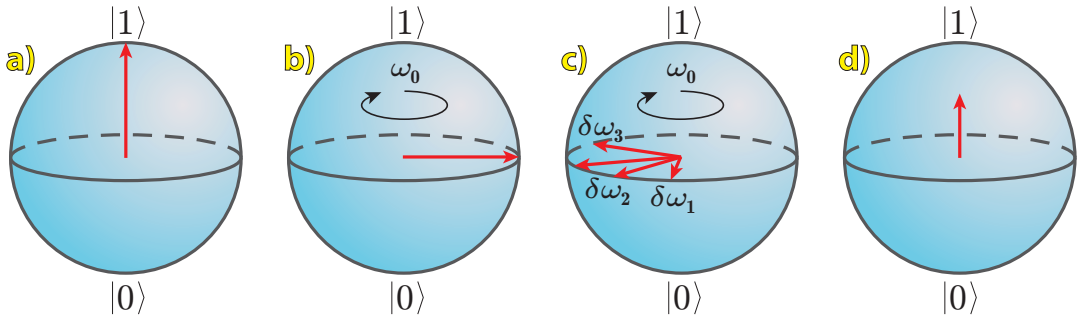


Figure 6: *Qubit de-phasing. A presentation of the quantum state on the Bloch sphere during a Ramsey measurement, starting from point* **a)** *the qubit is initially in the state* $|1\rangle$*, at point* **b)** *a* $\frac{\pi}{2}$*-pulse is applied to the qubit state and the state precesses at frequency* ω_0*.* **c)** *during precession, the noise in the system drives the state differently among the states in the ensemble.* **d)** *After a time t, when a second* $\frac{\pi}{2}$*-pulse is applied, the average return probability of the ensemble decreases.*

The second type of decay of the quantum state is known as qubit de-phasing and have two characteristic time-scales associated, T_2 and T_2^*. For T_2 style decay, we lose phase-coherence of a single spin by energy-conserving interactions with the environment and time-scale of these interactions set the upper-bound on the phase-coherence time we can achieve for the qubit. During experiments the limiting factor is the ensemble de-phasing time T_2^*; here different states in the ensemble experience a different instantaneous environment compared to the other states of the ensemble and this mechanism leads to loss of phase-coherence: The difference in local environment between states in the ensemble drive state precession at different frequencies leading to accumulation of phase differences between the states (hence the term de-phasing). When averaging over several measurements in order to do estimation of the quantum-state, e.g. by measuring return probability as function of time (Free Induction Decay), the act of averaging over the ensemble leads to loss of return probability (see figure 6).

There are two major causes of the loss of quantum coherence when it comes to exchange coupled spins, the first is magnetic noise δB coming from the fluctuating spins of the nuclei in the material and since this type of noise is attributed as the most significant cause for decoherence in the Resonant Exchange Qubit[4, 5], I will begin my investigation here.

Magnetic noise comes from fluctuations of the non-zero spin of the Gallium-Arsenide lattice; these fluctuations stem from hyperfine interactions that lead to effective magnetic fields (called Overhauser fields) in the quantum dots. Since the nuclei fluctuates randomly the mean local nuclear field can modelled as Gaussian distributed with $\langle B_{nuc}\rangle = 0$ T, and variance $\sigma_{B_{nuc}} = 2.0 \pm 0.1$ mT[4]. These fluctuations create a fluctuating magnetic gradient ΔB_z between the dots of the qubit. I take a quick detour and consider the double-dot spin-qubit, the Singlet-Triplet qubit. Here the magnetic gradient between the dots acts as one of the rotation axes needed for universal control, so magnetic noise enters directly into the operations of the qubit. Therefore the noise δB in this nuclear gradient lead to pure qubit de-phasing[6].

In the case of the Resonant Exchange Qubit, magnetic field differences between the right and the left side of the triple-dot drives oscillations between the logical states $|0\rangle$ and $|1\rangle$, and Zeeman energy differences between the left dot pair and the right dot pair lead to leakage out of the logical subspace[4]. The leakage occurs when the Zeeman energy of the left (right) side is comparable to the exchange energy in the left (right) side. This leakage can however be strongly suppressed by requiring $E_{\text{Zeeman}} \gg J \gg \delta B$[5], and this is the regime in which the qubit is operated but nuclear noise is still believed to be the dominating source for de-phasing[4].

The second kind of noise, i.e. electrical noise, is intrinsically present in semiconductor devices and enters via different mechanism, e.g. Johnson noise from the gates or piezoelectricity generated by phonon noise[7]. In our model for interaction, the exchange interaction between the spins is controlled by the gates and therefore electrical noise in the gates enters as noise δJ in the exchange-interactions J_i. The qubit is however electrically protected during operation, in the sense that noise in the detuning $\epsilon(t)$ only enters at second order, since $\frac{\mathrm{d}J}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} = 0$ (see figure 5 b). This description of charge noise from the gates can be taken further as described by Hu et al. [8]; here they divide the effects of into noise in the detuning, coming from noise in plunger gates, and into effects of noise coming from the barriers leading to noise in the tunneling rates.

Even though spin de-phasing due to electrical noise is small compared to nuclear noise, it has been proposed that electrical noise will be an important decoherence channel with the scaling of the number of gates needed in the device[8], and ultimately could make electrical noise a very important source for decoherence when the number of qubits is scaled up.

Another effect of electrical noise is proposed by Huang et al. [7]: since the dots are gate-defined, they model how electrical noise, by shifting the position of the dot, lead to an effective magnetic noise in the qubit. This effect leads to spin-relaxation via spin-orbit coupling to the nuclei, and to negligible de-phasing.

The combined effect of the electrical and magnetic noise is decay of the quantum state on the order of $T_2^* \sim 10$ ns for the Singlet-Triplet qubit[6], while the resonant exchange qubit is less sensitive to the noise and decay happens in times on the order of $T_2^* \sim 25$ ns[4].

Usually a lot of effort is put unto understanding and removing magnetic noise only, but for the Resonant Exchange Qubit, the most successful model for describing the noise contributing to qubit de-phasing under free induction decay is the model made by Medford et al. [4] and includes both nuclear **and** electrical noise. I therefore wish to focus my efforts on suppressing the qubit de-phasing regardless of the mechanism causing the decay. So I will try to implement a type of feedback which compensates for both types of noise, with the only requirement that noise in the qubit is low-frequency compared to the bandwidth of the feedback loop.

4.5 Noise decoupling

Among the efforts to eliminate the effect of the noisy environment, there is the Hann echo approach of applying a sequence of π-pulses during qubit operation to eliminate the low-frequency part of the noise. By Hann echo, one effectively high-pass filters the noise in the environment and can by this process re-phase the qubit. By use of even more advanced pulse-sequences, such as Carr-Purcell-Meiboom-Gill or Concatenated Dynamical Decoupling, this process can compensate the effects of noise even better but this is complicated to achieve and will only partially re-phase the quantum state[4, 5].

By the method of such Hann-echo style pulse sequences, it is possible to infer information of the noisy environment felt by the qubit; by varying the number of pulses in the sequence, one effectively increases the characteristic frequency of the high-pass filter (filtering frequency goes as one over the pulse-separation period), and it is therefore possible to probe the spectral density of the noise. Jim Medford has in his work investigated the power spectral density of the Single-Triplet Qubit[9] and of the Resonant Exchange Qubit[10]; he found a power-law model for the power spectral density, $S(\omega) \sim \omega^{-\beta}$, with $\beta = 2.6 \pm 0.1$ for the Singlet-Triplet Qubit and $\beta = 5 \pm 1$ for the Resonant Exchange Qubit. This is a promising result when considering using feedback to increase the coherence time since a large portion of the noisy environment is low-frequency, and because we are limited by operation time and readout fidelity we require (most of) the noise to be slow compared to the measurement time.

5 Experimental set-up

Now that we have sufficient knowledge about the Resonant Exchange Qubit and the information available of how noise effects the qubit coherence I will discuss how to use FPGA mediated feedback to elongate the coherence time. First I discuss the work that has been done to achieve exactly this for the Singlet-Triplet qubit. Then I will describe how to implement the FPGA-based feedback controller into the current qubit operation by considering the possible channels available to use for feedback in our device. Finally I consider bandwidth limitations of these channels and of the feedback loop as well as limitations of the feedback loop in general.

5.1 FPGA-based feedback on the Singlet-Triplet qubit

Recent work has shown great increase in the coherence time for the Singlet-Triplet spin-qubit by means of nuclear programming and FPGA based feedback on the exchange interaction. Shulman et al. [11] have implemented a two step process of first using dynamic nuclear polarization to set the mean nuclear gradient between the two dots, and have by this process increased T_2^* from ~ 10 ns to ~ 70 ns. The second step is to use FPGA based feedback to control the frequency of pulsing $\epsilon(t)$ during qubit operations. They use the FPGA to estimate the instantaneous nuclear gradient $\Delta B_z + \delta B_z$ and tune the frequency of the exchange interaction to match the frequency of the rotations around this gradient leading to slower de-phasing of the quantum state. By switching between

measuring the fluctuating background and running qubit operation, they have used their method of Bayesian Hamilton parameter estimation to increase T_2^* even further to ~ 2 µs[11].

5.2 Operational bandwidth

Clearly FPGA based control is an effective method, at least for the Singlet-Triplet qubit using dynamic nuclear programming, but to see how well the results translate over to the resonant exchange qubit we need to make some considerations on our set-up, starting with consideration of the bandwidth of qubit operation. In order to achieve the best possible noise compensation, we need to not be limited by the bandwidth of the feedback controller, but rather to be limited by the time it would take to measure the splitting of the qubit, which is the rate at which the qubit state can be read out. As mentioned earlier, the best achievable operation rate of the FPGA in its current configuration is ~ 56 kHz for a minimum working example (see Appendix C). In fact the situation gets even worse when I measure the speed of the FPGA with the PID controller implemented, having an operational rate of ~ 35 kHz (see Appendix E). The fastest way to read out the qubit is Single-shot measurement, and this was on the Medford Triple-Dot Resonant Exchange Qubit achieved in the time-scale of ~ 10 µs, and since several measurements are needed to estimate the quantum state, we need on the order of ~ 100 single-shot measurements meaning it would take on the order of miliseconds or hundreds of microseconds to estimate the energy splitting of the qubit. This means that we are likely not limited by the operation rate of the FPGA.

5.3 Choosing a feedback channel

The next consideration of the feedback loop is to choose the system input to feedback through and here there are a few options. One could, as Shulman et al. [11], feedback on the driving frequency of the qubit using a voltage controlled oscillator (VCO) to match the frequency to the energy splitting. Alternatively one can feedback on the gate voltages controlling the energy splitting from the exchange interaction; this method leads to two further options, to use the middle plunger (MP) or to use the two outer plungers (LP and RP) simultaneously to give the same effect as changing the middle plunger. These choices of feedback on the gates are different in two ways. To understand how they are different we must first consider the bandwidth of the gates and secondly consider what the gates control.

In order to minimize charge noise from the gates, most of the gates are heavily low-pass filtered with a bandwidth on the order of tens of kilohertz, while some of the gates, the fast-lines, are wired differently and have two lines connected by a bias tee, one is fast with a bandwidth on the order of gigahertz but this line is high-pass filtered and attenuate DC currents, while the other is one of the heavily low-pass filtered lines attenuation most high frequency signal. The left and right plunger are fast-lines and the bandwidth of the feedback loop would be the limiting factor to how fast the feedback circuit can respond. The middle plunger, is on the other hand one of the bandwidth

limited lines, with an expected bandwidth on the order of tens of kilohertz, meaning that it would be similar to the feedback bandwidth. This is not really a limitation in a general sense, since the middle plunger in principle could be wired as a fast-line as well, but it is rather a practical problem since the number of fast-lines is limited. The upshot to using the middle plunger as the feedback channel is that it directly controls the size of the $(1, 1, 1)$ region giving direct control of the operation regime. It is possible, that using the left and right plunger simultaneously will have their combined effect do the same as changing the middle plunger and therefore also change the size of the $(1, 1, 1)$ region, but if this is not the case they could not be used as a feedback channel. Medford et al. [10] measured the return probability as a function of voltage on the middle plunger and as a function of the driving frequency (see Medford et al. [10] figure 2 d) showing that both the middle plunger and the driving frequency control the de-phasing and that both channels can be used as a feedback channel.

When I measured the frequency bandwidth of the middle-plunger, I found that it was significantly less than the expected tens of kilohertz having a 3 dB cut-off around ~ 4.5 kHz (see Appendix G). This unfortunately makes it a lot less viable as a feedback channel, and in many cases leaves us with the remaining two choices of either feeding back on the DC offset on the fast-lines or on the driving frequency of the detuning.

Another limitation of the feedback controller I am using, is that the ADC and DAC only have an 8-bit resolution, meaning that they can distinguish between 256 levels in the $0\text{ V} - 3.3\text{ V}$ range. This resolution can however be improved by attenuating the output of the feedback controller. In the operating regime for qubit the required bias voltage on the gates are typically on the order of hundreds of millivolts, meaning that if the feedback controller is to deliver such biases alone the resolution voltage is on the order of a couple of millivolt, which is significantly lower than the resolution used to fine-tune the gates, which can be on the order of tenths of millivolt. In some cases this is not a problem since the required voltage for a gate can be on the order of tens of millivolt, as is the case for the current configuration of our triple-dot, where the middle plunger is operated around 40 mV, and the feedback loop can then achieve a high resolution on this gate making the middle-plunger viable as a feedback channel in terms of resolution. But in general the bias needed changes alot even within the same device so this will not always be the case.

In an attempt to devise a solution to the case where the required voltage is on the order of hundreds of millivolt, our first solution-attempt was the use of a bias tee, where the DC offset could be applied by another DAC and have the feedback controller use a much narrower range to adjust around this bias. This has so far been unsuccessful, and I suspect this is because that the feedback controller could only respond with a step impulse, and never be allowed to stabilize to any other value than set by the second DAC, since any DC offset set by the feedback controller would not pass through the bias tee. This limitation is again not a principal problem but rather a problem of the particular 8-bit feedback controller I have used for this thesis; it is however worth noting that high resolution on an FPGA based system is often very expensive and that a feedback channel where bit depth is not important can be preferable in that sense.

The overall conclusion for the choice of feedback controller is that if the feedback controller is to have direct control over the gate voltages then the DAC of the feedback controller would benefit greatly by having a higher resolution than the 8 bits of the NANOBOARD. I have not considered the effect of using the voltage controlled oscillator as feedback channel in detail, but its promise have been shown by Shulman et al. [11], and this channel does not have the apparent bit-depth problem.

6 Outlook

I will in the following section describe some of future plans for use of the FPGA based feedback that have been considered during the making of this thesis. There is of course the obvious plan for testing the current feedback controller on a working resonant exchange qubit, which was not achieved for the deadline, but I will in this section focus on another area where the feedback controller can be useful.

6.1 Faster charge sensing with feedback

The first useful implementation of the FPGA-based feedback is to use it for charge sensing. In the current state of affairs, the quantum dots are first tuned by measuring transport either directly through the dot or by measuring transport through the adjacent sensor dot. Unfortunately this method of measuring transport is very slow as we are limited by the speed of the DMM's and the increasing number of qubits that need to be tuned this is making it an increasingly time-consuming process. Alternatively there is the approach of measuring the charge state with reflectometry: here an inductor, soldered to the sample holder, is connected to the sensor which acts as an RC component, sensitive to the local electrostatic environment. In the event of an electron tunnelling into an adjacent dot the resonance frequency of the reflectivity circuit changes slightly, and the reflected signal changes amplitude, allowing us to readout the charge state. This method is very fast but is limited to sweeping only a small part of parameter-space since cross-coupling between the gates and the sensor move the sensor away from the sensitive point when the gate voltages change too much. Therefore the qubits are tuned up by first measuring transport and find the approximate regimes for the gate voltages and then we switch to reflectometry and fine-tune the parameters. This process can be optimized by using the feedback-circuit to feedback on the sensor to compensate for the cross-coupling during reflectometry measurement. This would be accomplished by using the plunger of the sensor (the middle gate in figure 4 a) to maintain the sensitive point during sweeps. This is exactly what the test demodulation circuit is a model for (see figure 2): For the test circuit, the FPGA changes the capacitance of the varactor in the circuit to keep the amplitude of the reflected signal constant, whereas for charge sensing it would control the sensor's middle-plunger to accomplish the same. Changes in the charge occupancy of the triple-dot would then show up in the feedback response which then can be used to read out the charge state. The only current problem with implementing this method, is that the sensor is currently fine-tuned by changing the

voltage on the sensor plunger on the order of tenths of milivolt, a resolution which is not achievable since the operation point of the current sensors are on the order of hundreds of millivolt as described above (see section 5.3).

7 Summary and conclusion

I have in this bachelors thesis shown the basics of control theory and how to use it to understand the concept of feedback as well as its impact on the dynamics of a general system. I then discussed the implementation of a specific feedback controller, namely the digital PID controller in form of C code running on a soft processor in an FPGA. The controller was then tested on a circuit built specifically to simulate the conditions used under experiments. The FPGA platform was tested for speed showing that the current implementation is operating at $\sim$ 35 kHz but optimization is believed achievable, but I did not further investigate how this could be improved. One solution suggested in literature is to consider in the workings of the FPGA at the register transfer level and to use this knowledge to optimize the C code, such that the compiler can fully utilize the speed of the FPGA[2].

I then discussed the formation of the resonant exchange qubit and considered how a large part of the leakage out of the logical subspace was suppressed in the operating regime. I went on to discuss in detail the mechanisms leading to electrical and magnetic noise entering into the qubit, and presented the knowledge available on the spectral density of the noise, namely that resonant exchange qubit is dominated by low frequency noise $S(\omega) \sim \omega^{-5}$, suggesting the possibility for a highly effective feedback loop.

I ended by considering the promise FPGA-based feedback have shown for other spin-qubits and I discussed how to implement the feedback in practice for the resonant exchange qubit, along with a discussion on the limitations of my feedback controller and the limitations of the feedback channels available.

I found that the limiting factor in terms of the bandwidth of the feedback loop is likely the time it takes to estimate the energy splitting of the qubit which would be on the order of miliseconds, meaning that even a relatively slow FPGA can be used.

8 Acknowledgements

I would like to end this bachelors thesis with an expression of my gratitude to the people of the Center for Quantum Devices for helping me with the process of doing the work presented in this bachelors thesis. Especially, I would like to thank Johannes Beil, Anton Kovyakh and Ferdinand Kuemmeth for their useful discussions and willingness to pass their knowledge onto me.

I would like to thank Charlie Marcus for inviting me to join the Center for Quantum Devices and for giving me the opportunity to work on the spin-qubit experiment as well as guiding me through this thesis.

References

[1] J. Bechhoefer. "Feedback for physicists: A tutorial essay on control". In: *Reviews of Modern Physics* 77 (3 2005), pp. 783–836. DOI: 10.1103/RevModPhys.77.783. URL: http://link.aps.org/doi/10.1103/RevModPhys.77.783 (cit. on pp. 2, 5).

[2] S. Hauck. *Reconfigurable computing: The theory and practice of FPGA-based computation*. Morgan Kaufmann, 2008, pp. 3–27, 155–181. ISBN: 978-0-12-370522-8 (cit. on pp. 6, 7, 18).

[3] I. Kuon et al. "Measuring the Gap Between FPGAs and ASICs". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26 (2 2007), pp. 203–215. DOI: 10.1109/TCAD.2006.884574. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4068926 (cit. on p. 6).

[4] J. Medford et al. "Self-consistent measurement and state tomography of an exchange-only spin qubit." In: *Nature nanotechnology* 8 (2013), pp. 654–9. DOI: 10.1038/nnano.2013.168 (cit. on pp. 11–14).

[5] J. Hung et al. "Decoherence of an exchange qubit by hyperfine interaction". In: *Unpublished* (2014), p. 15. URL: http://arxiv.org/abs/1404.6220 (cit. on pp. 12–14).

[6] J. R. Petta et al. "Coherent Manipulation of Coupled Electron Spins in Semiconductor Quantum Dots". In: *Science* 309 (5744 2005), pp. 2180–2184. DOI: 10.1126/science.1116955. URL: http://www.sciencemag.org/content/309/5744/2180.abstract (cit. on pp. 12, 13).

[7] P. Huang et al. "Electron spin relaxation due to charge noise". In: *Physical Review B* 89 (19 2014), p. 195302. DOI: 10.1103/PhysRevB.89.195302. URL: http://link.aps.org/doi/10.1103/PhysRevB.89.195302 (cit. on p. 13).

[8] X. Hu et al. "Charge-Fluctuation-Induced Dephasing of Exchange-Coupled Spin Qubits". In: *Physical Review Letters* 96 (10 2006), p. 100501. DOI: 10.1103/PhysRevLett.96.100501. URL: http://link.aps.org/doi/10.1103/PhysRevLett.96.100501 (cit. on p. 13).

[9] J. Medford et al. "Scaling of Dynamical Decoupling for Spin Qubits". In: *Physical Review Letters* 108 (8 2012), p. 086802. DOI: 10.1103/PhysRevLett.108.086802. URL: http://link.aps.org/doi/10.1103/PhysRevLett.108.086802 (cit. on p. 14).

[10] J. Medford et al. "Quantum-Dot-Based Resonant Exchange Qubit". In: *Physical Review Letters* 111 (5 2013), p. 050501. DOI: 10.1103/PhysRevLett.111.050501. URL: http://link.aps.org/doi/10.1103/PhysRevLett.111.050501 (cit. on pp. 14, 16).

[11] M. D. Shulman et al. "Suppressing qubit dephasing using real-time Hamiltonian estimation". In: *Unpublished* (2014), pp. 1–8. URL: http://arxiv.org/abs/1405.0485 (cit. on pp. 14, 15, 17).

Appendices

Appendix A FPGA design

The FPGA design software for the NANOBOARD 3000, Altium Designer, control the internal routing of signals within the board based on the schematics of the design. This design stage is two-fold; There is the signal routing from the ADC, DAC and clock-board to the OpenBus, and there is the routing of signals within the OpenBus. These OpenBus based schematics are Altiums way of handling traditional FPGA schematics, and are by the design software translated to match the peripherals to the pins of the FPGA. Both of these schematics are then translated into VHDL and programmed onto the FPGA.

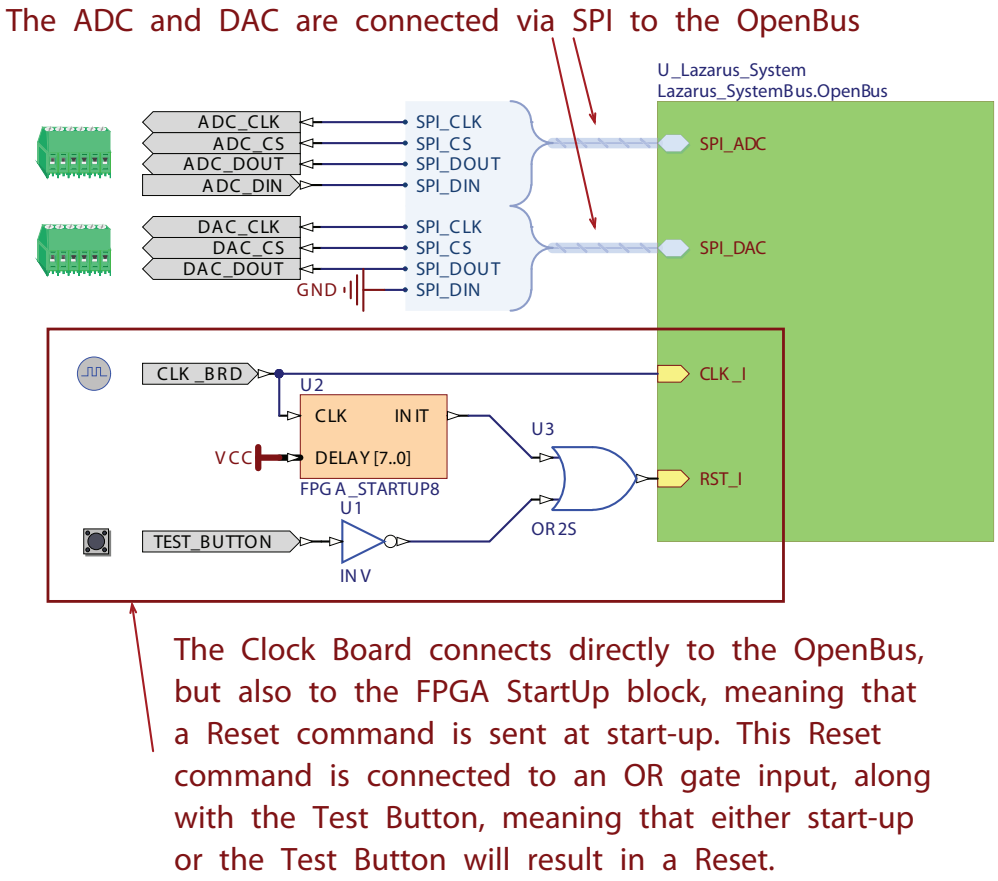


Figure 7: *Peripherals schematics. These are the schematics showing the connection of the peripherals to the OpenBus. The DAC and ADC are connected via the SPI bus protocol, and the clock board and test button are connected to fixed pins of the FPGA.*

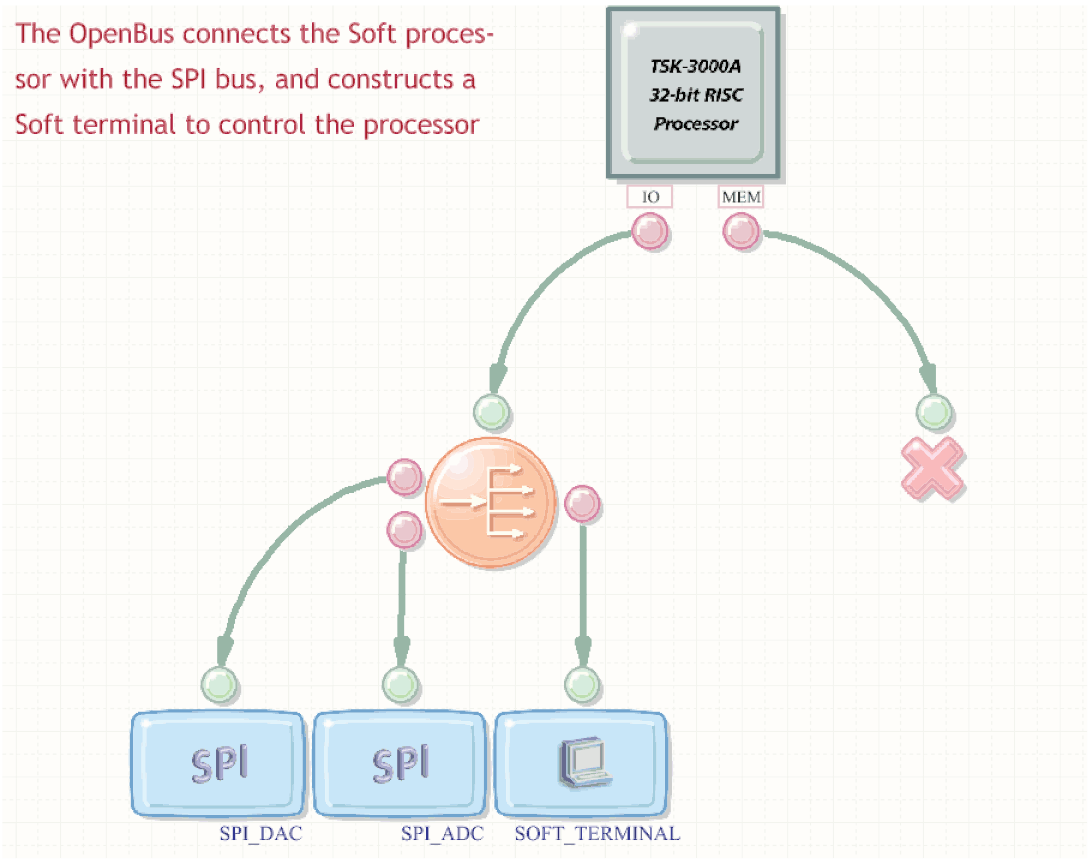


Figure 8: *Altium OpenBus. The OpenBus handles the signal connections from the SPI connected peripherals to the soft processor (TSK-3000A) handling the feedback controller. The OpenBus is further connected to a soft terminal to probe the system during testing.*

Appendix B FPGA code structure

This section show the overall structure of the C-code twhich is compiled to VHDL. This structure does not change for the different code snippets in the appendices below, only the *main program* changes between the different tests.

```
1  // Include standard C-library files
2  #include <stdlib.h>          // Defines numeric conversion functions,
3                               // pseudo-random numbers generation func-
4                               // tions, memory allocation, process con-
5                               // trol functions.
6  #include <stdio.h>           // Defines core input and output functions
7  #include <time.h>            // Defines date and time handling functions
8
9  // Include Nanoboard C-library files
10 #include <devices.h>         // Software Platform Generated Devices
11 #include <timing.h>          // Timing and timer services
12 #include <drv_adc084s021.h> // ADC084S021 ADC Driver
13 #include <drv_dac084s085.h> // DAC084S085 DAC Driver
14
15 // Declare initializing function
16 static void init( void );
17
18 // Declare rounding function
19 static int float2int( float K );
20
21 // Define DAC and ADC pointers
22 adc084s021_t    *adc;
23 dac084s085_t    *dac;
24
25
26 // Main program of the FPGA
27 void main( void )
28 {
29
30
31 }
```

```
32 // ADC and DAC initialization function
33 static void init( void )
34 {
35     // Print to terminal
36     printf( "Opening ADC:  ");
37 
38     // Initialize the ADC driver
39     adc = adc084s021_open( DRV_ADC084S021_1 );
40 
41     // Report success or failure to terminal
42     puts ( adc ? "OK!" : "Failed" );
43 
44 
45     // Same procedure for DAC
46     printf( "Opening DAC:  ");
47     dac = dac084s085_open( DRV_DAC084S085_1 );
48     puts ( dac ? "OK!" : "Failed" );
49 }
50 
51 
52 // Rounding function for float-integer conversion
53 static int float2int( float K )
54 {
55     // Float Conversion use flooring to convert to integer,
56     // by adding 0.5 to the float the conversion acts as a
57     // rounding rather than flooring function
58     return( (int)(K + 0.5) );
59 }
```

Appendix C ADC sampling rate

To test the effective sampling rate of the ADC, I set up a simple test where the FPGA continuously performs a series of two DAC writes followed by two sequences of reading the ADC and writing to the DAC. The output of the DAC was then read by fast data acquisition card (AlazarTech ATS-94440) and the internal delays of the FPGA was directly measured. The main code segment of the test is posted below.

```
26 // Main program of the FPGA
27 // ADC sample rate test
28 void main( void )
29 {
30
31     // Initialize DAC and ADC
32     init();
33
34     // Loop forever
35     for (;;)
36     {
37         // Write to DAC
38         dac084s085_write( dac, DAC084S085_OUTA, (uint8_t)0, true);
39         // Read ADC channels
40         adc084s021_read( adc, 0 );
41
42         // Write to DAC
43         dac084s085_write( dac, DAC084S085_OUTA, (uint8_t)25, true);
44         // Read ADC channels
45         adc084s021_read( adc, 0 );
46
47         // Write to DAC
48         dac084s085_write( dac, DAC084S085_OUTA, (uint8_t)50, true);
49
50
51         // Write to DAC
52         dac084s085_write( dac, DAC084S085_OUTA, (uint8_t)75, true);
53     }
54
55 }
```

From this sequence we get the writing time of the DAC, (by the last two steps) and the combined time of a ADC read and DAC write (from the first two steps). From this sequence we find that the ADC has an effective sampling rate of ($\sim$ 115 kHz), and the DAC is limited to the same operating frequency, leading to a read-write cycle at a rate of $\sim$ 56 kHz (see figure 9).

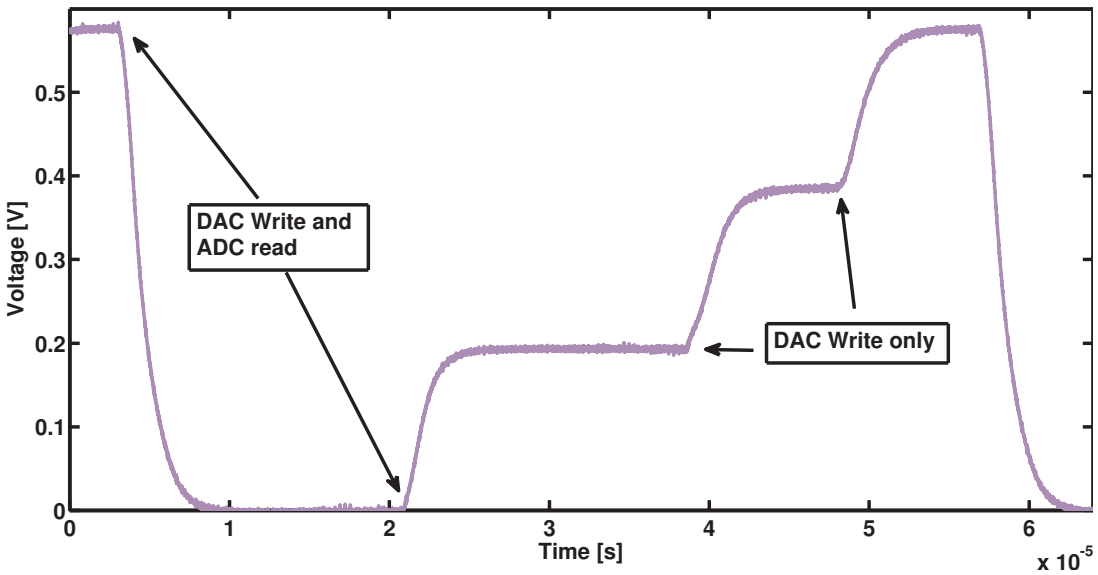


Figure 9: *ADC sampling rate. The Analog-to-Digital converter of the Nanoboard 3000 was tested to find the effective sampling rate of the board configuration; the FPGA performed a sequence of reads of the ADC and writes to the DAC (Write 0 - Read - Write 25 - Read - Write 50 - Write 75)*

Appendix D PID controller code

Here you will find the code implementing the PID controller. Note that the control parameters are tuned between runs and are set at a random value below.

```
26 // Main program of the FPGA
27 void main( void )
28 {
29     // == Define variables ==
30 
31     // Various signals
32     int y;                // ADC read value
33     int e = 0;            // Error signal
34     int R = 85;           // Reference signal (76 = 1V)
35     int K = 0;            // Feedback response
36 
37     // PID controller variables
38     float KC = 2;         // Generalized gain
39     float TI = 1;         // Integral time constant
40     float TD = 1000;      // Derivative time constant
41 
42     // PID response variables
43     int sum = 0;          // Sum needed for integral part
44     int diff = 0;         // Difference needed for derivative part
45 
46     // Previous values to be used in next iteration
47     int e_prev = 0;       // Previous Error measurement
48 
49 
50     // Initialize DAC and ADC
51     init();
52 
53     // Set DAC to median value (8 bit DAC, 256 values)
54     dac084s085_write( dac, DAC084S085_OUTA, (uint8_t)127, true);
```

```
55      // Loop forever
56      for (;;)
57      {
58          // Read ADC channel 0
59          y = adc084s021_read( adc, 0 );
60
61          // Calculate error signal
62          e = R-y;
63
64          // Calculate difference
65          diff = e - e_prev;
66
67          // Calculate PID feedback
68          K = 127 + float2int(KC*e + KC*TI*sum + KC/TD*diff);
69
70          // Set boundary on K
71          if(K>255)
72          {
73              K = 255;
74          }
75          else if(K<0)
76          {
77              K = 0;
78          }
79          else  // To prevent integral wind-up,
80          {     // We only update the sum term when K is not saturated
81              sum = sum + e;
82          }
83
84          // Write to DAC
85          dac084s085_write( dac, DAC084S085_OUTA, (uint8_t) K, true);
86
87          // Update prev values
88          e_prev = e;
89      }
90
91  }
```

Appendix E PID operation rate

After implementing the PID controller I wanted to test how this changed the rate at which the feedback circuit could respond. This was tested in a similar manor as the sampling rate of the ADC (see Appendix C). By having the FPGA change between outputting 0 and 255 after performing a feedback calculation.

```
26 // Main program of the FPGA
27 void main( void )
28 {
29     // == Define variables ==
30 
31     // Various signals
32     int y;                  // ADC read value
33     int e = 0;              // Error signal
34     int R = 85;             // Reference signal (76 = 1V)
35     int K = 0;              // Feedback response
36 
37     // PID controller variables
38     float KC = 2;           // Generalized gain
39     float TI = 1;           // Integral time constant
40     float TD = 1000;        // Derivative time constant
41 
42     // PID response variables
43     int sum = 0;            // Sum needed for integral part
44     int diff = 0;           // Difference needed for derivative part
45 
46     // Previous values to be used in next iteration
47     int e_prev = 0;         // Previous Error measurement
48 
49 
50     // Initialize DAC and ADC
51     init();
52 
53     // Set DAC to median value (8 bit DAC, 256 values)
54     dac084s085_write( dac, DAC084S085_OUTA, (uint8_t)127, true);
```

```
55      // Read ADC channel 0
56      y = adc084s021_read( adc, 0 );
57
58      for(;;)
59      {
60
61      // Read ADC channel 0
62      y = adc084s021_read( adc, 0 );
63
64      // Calculate error signal
65      e = R-y;
66
67
68      // Calculate PI feedback
69      K = 128 + float2int(KC*e + KC*TI*sum);
70
71
72      // Set boundary on K
73      if(K>255)
74      {
75          K = 255;
76      }
77      else if(K<0)
78      {
79          K = 0;
80      }
81      else  // To prevent integral windup,
82      {     // We only update the sum term when K is not saturated
83          sum = sum + e;
84      }
85
86
87      // Write 0 to DAC regardless of K
88      dac084s085_write( dac, DAC084S085_OUTB, (uint8_t)0, true);
89
90
91      // Update prev values
92      e_prev = e;
```

```
93      // Read ADC channel 0
94      y = adc084s021_read( adc, 0 );
95
96      // Calculate error signal
97      e = R-y;
98
99
100     // Calculate PI feedback
101     K = 128 + float2int(KC*e + KC*TI*sum);
102
103
104     // Set boundary on K
105     if(K>255)
106     {
107         K = 255;
108     }
109     else if(K<0)
110     {
111         K = 0;
112     }
113     else  // To prevent integral windup,
114     {     // We only update the sum term when K is not saturated
115         sum = sum + e;
116     }
117
118
119     // Write  255 to DAC regardless of K
120     dac084s085_write( dac, DAC084S085_OUTB, (uint8_t)255, true);
121
122
123     // Update prev values
124     e_prev = e;
125     }
126
127
128 }
```

This test show the effective operating frequency of the FPGA implemented feedback controller being on the order of ~ 35 kHz (see figure 10).

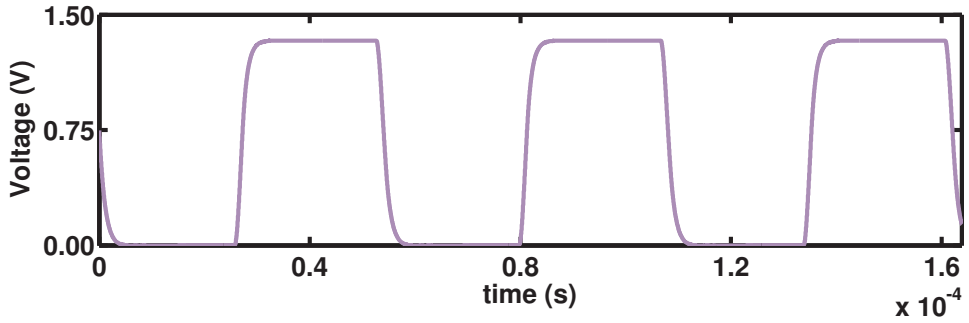


Figure 10: *Operational rate of PID controller. A similar test as in Appendix C, and allow us to calculate the effective operation rate of the PID controller to be in the order of* ~ 35 kHz

Appendix F Demodulation circuit for the qubit experiment

This appendix show the readout circuit used in the qubit experiment. The green boxes indicate the circuit elements similar to the elements in the FPGA test-circuit (see figure 2). The transmission line, TX, and receiving line, RX, are connected by an directional coupler inside the fridge.

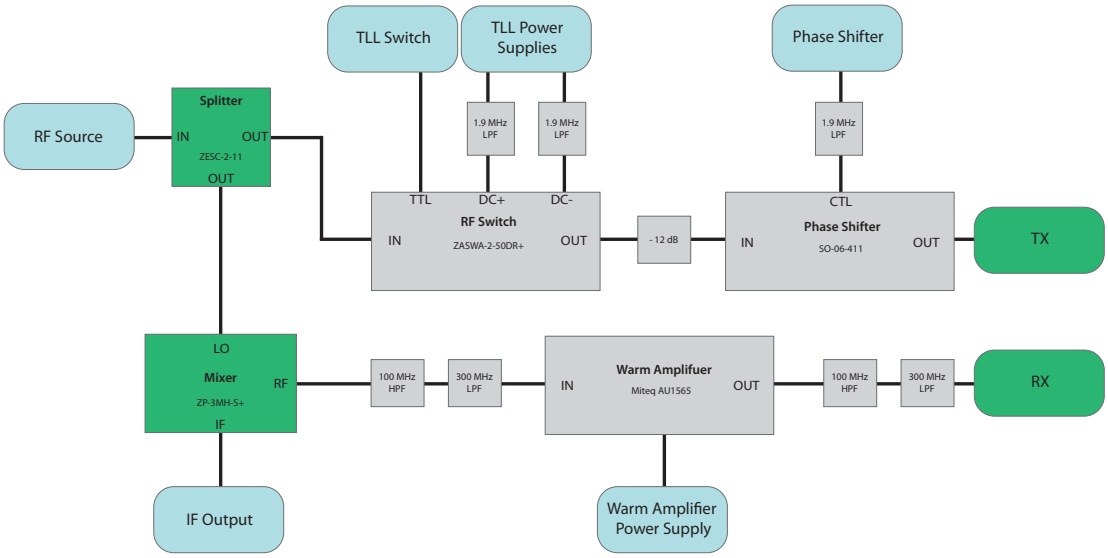


Figure 11: *Qubit Demodulation Circuit. Schematic of the demodulation circuit used in the experiment on the Qubit. The RF signal is split in two, one used fas the local oscillator for homodyne detection and one sent to readout the sensor. This signal is controlled by a RF Switch and a phase-shifter before going into the transmission line (TX). The reflected signal is sent trough a directional coupler and comes back via the receiving line (RX). The green elements (along with the directional coupler in the fridge) indicate the elements which the test demodulation circuit (see figure 2) is based on. The schematics are based on Jim Medfords original drawing from his thesis (At the time of writing it is available here: http://qdev.nbi.ku.dk/student_theses/pdf_files/Medford_2013.pdf pp.129)*

Appendix G Bandwidth of DC lines

In the process of determining a suitable channel for feedback, I needed to test the bandwidth of the DC lines all the way down the electrostatic gates defining the quantum dots. To measure the bandwidth we tuned to sensor into a regime where it was sensitive to reflectometry measurement and applied a sinusoidal signal on top of the bias voltage for the middle plunger using a bias tee. Due to the cross-coupling between the gate and the sensor the reflected signal of the reflectometry circuit changes in amplitude at the same frequency as the applied sinusoidal signal, and this enabled us to probe the signal, as seen by sensor. With the use of a Dynamical Signal Analyser (HP3561A), we could extract amplitude of the driving frequency in the reflected signal, and measure the attenuation as a function of frequency (see figure 12). We find that the 3 dB is at ~ 4.5 kHz, which is much smaller than the expected tens of kilohertz.

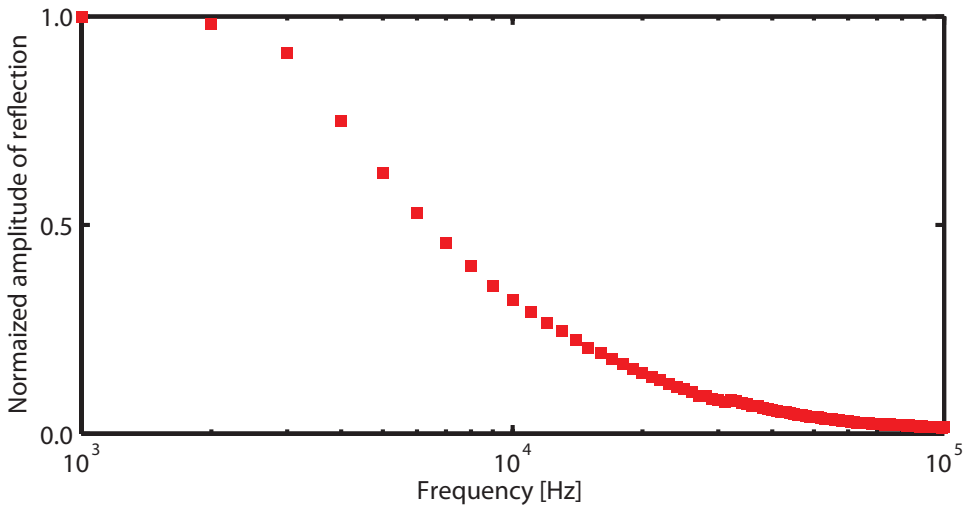


Figure 12: *DC line bandwidth. By using the cross-coupling between the sensor and the middle plunger of the triple-dot, I was able to measure bandwidth of the DC lines by driving the middle plunger at different frequencies, and extracting the amplitude of reflection from the reflectometry circuit by use of a dynamical signal analyser.*