Quantum Dot Circuits: Single-Electron Switch and Few-Electron Quantum Dots

A thesis presented

by

Ian Hin-Yun Chan

to

The Department of Physics

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Physics

Harvard University

Cambridge, Massachusetts

August 2003

Abstract

Quantum Dot Circuits: Single-Electron Switch and Few-Electron Quantum Dots

Advisor: Robert M. Westervelt — Author: Ian H. Chan

A strongly capacitively-coupled parallel double quantum dot was studied as a single-electron switch. The double dot was fabricated in a two-dimensional electron gas (2DEG) in a GaAs/AlGaAs heterostructure. An electrically-floating coupling gate increased capacitive-coupling between the dots, while an etched trench prevented tunnel-coupling between them. Split Coulomb blockade peaks were observed in each dot, and the Coulomb blockade conductance of the double dot formed a hexagonal pattern characteristic of coupled dots. A fractional peak splitting $f = 0.34$ was measured, which corresponds to a fractional capacitive-coupling $\alpha \equiv C_{INT}/C_{\Sigma} = 0.20$. This is an order of magnitude larger than reported for similar lateral quantum dots, and shows that the coupling gate works. The strong capacitive-coupling in our device allowed the charge state of one dot to strongly influence the conductance of the other dot and enabled it to work as a single-electron switch. By moving in a combination of gate voltages, electrons are induced in one dot (the "trigger" dot) only. In response to the change in the charge state, the conductance of the other dot (the "switched" dot) is turned on and off. The abruptness of the conductance switching in gate voltage (the switching lineshape) is determined by how well charge is quantized on the trigger dot, and was found to follow tanh and arctan forms for (respectively) good and poor charge quantization in the trigger dot.

A few-electron tunnel-coupled series double dot was studied for possible application to quantum computing. The device was fabricated in a square-well 2DEG in a GaAs/AlGaAs heterostructure. The dots were emptied of electrons in order to define the absolute number of electrons in the dot. Finite bias Coulomb blockade measurements on each dot showed that the last Coulomb blockade diamonds did not close and thus that both dots could be emptied. A three-dimensional conductance measurement of one dot in the one sidegate and the two quantum point contact voltages also showed that Coulomb blockade peaks ended, and corroborated that the dot could be emptied of electrons. The Zeeman energy of electrons in a few-electron dot, deduced from the Coulomb blockade peak spacings, was measured with an in-plane magnetic field of up to 7 T. The g-factor was found to be no different from that of bulk GaAs $|g| = 0.44$. Tunnel-coupling between the few-electron double dot was demonstrated, and a tunnel-coupling strength of $1.2e^2/h$ was estimated from the fractional peak splitting $f = 0.3$.

Table of Contents

Acknowledgements

The time I spent at Harvard working on my Ph.D. was an experience that will continue to shape me for the rest of my life, and I wish to thank everyone who contributed to it. First, I would like to thank my thesis advisor Bob Westervelt for his consistent support, creation of a wonderful working environment, and wealth of charming stories about all things physics. I am also immensely grateful to Bob for allowing me to defer my entry into graduate school by a year so that I could get married. I would also like to thank the other members of my thesis committee, Charlie Marcus and Rick Heller. Charlie Marcus is tireless in his inquisitiveness and has always been a source of frank and honest advice about physics, graduate school, and academia. Rick Heller has broadened my idea of what it means to be a Professor of Physics by being an artist and a gentleman in addition to doing great physics.

It was also my great privilege to have worked with some amazing colleagues during my time here. I thank my mentor David Duncan for showing me the ropes in the Westervelt lab and for spilling his secrets on how to operate a dilution fridge. To Lester Chen, the ever organized and patient colleague to whom I could turn to have my questions answered, thank you for much useful advice on surviving graduate school. The lab would not have been the same without Mark Topinka. Always full of creative and sometimes hare-brained schemes, Mark was the dynamo behind many project ideas. I will also remember him as a formidable opponent in our networked games of Myth. Brian LeRoy was my “buddy” with whom I have shared my Ph.D. process with most closely. His impeccable logic and intuition almost never failed to solve a problem that I encountered in the lab, and it was also great to discuss ideas with him. Chung-Sok Lee and Hak-Ho Lee brought fun and enthusiasm into the lab, and Marija Drndic added that feminine touch. To all the new students – Parisa Fallahi, Andy Vidan, Muhammed Yildirim, Ania Bleszynski, Kathy Aidala, Ben Lee, Katie Humphry, and Tom Hunt

– I regret not being able to get to know you all better, but I wish you the best in your endeavors in physics and in life.

Looking beyond the Westervelt lab to the larger physics community, I really appreciated the fellowship of other dilution fridge users from the Tinkham lab – Sarah Pohlen, Delores Bozovic, Jeannie Lau, Nina Markovic, William Neils and Sergio Valenzuela. Adding to the experimental condensed matter camaraderie in McKay were Josh Folk, Sara Cronenwett, Dominik Zumbuhl, Jeff Miller and Alex Johnson from the Marcus lab. I'd also like to express my appreciation to Leo Kouwenhoven at Delft and David Goldhaber-Gordon, now at Stanford University, for lending me their considerable intellect while they were here at Harvard. My thanks also go out to Micah Hanson and Art Gossard at U. C. Santa Barbara, who have been great research collaborators, and to the staff at Harvard – Steve Shephard, Yuan Lu, and Jim MacArthur – without whom things would quickly fall apart.

I would also like to acknowledge the huge part that friends and my smallgroup at Park Street Church have played in my life. The fellowship of other newly married couples at the "Honeymooner's" class at Park Street Church were a tremendous help in getting my fledgling marriage onto more solid footing, and the fellowship of my smallgroup has been priceless. Jon, Dawn, Nicholas ("pumpkin boy") and Rachel Demerly, Ryan and Christine Buchholz, Chris, Maria, Dillon and Noah Bradley, Kyung, Allison, Joy and Ryan Kim, Lynn and Pam Chang, Ken and Sara Pearson, Walter, Toni and Nathaniel Kim, Nate and Mee-Heh Risdon, and Tim and Joyce Dalrymple, Janice and I will miss you all sorely. Thank you for the laughter, the tears, the prayers, the babies, and the food.

To my parents, I give them my deepest thanks for all the love, support and encouragement that they have given me over the years. They always put their children first and invested in us, and one of their gifts, the *Children's Brittanica*, is probably responsible for putting me on this path of physics. I would like to remember my grandmother “Poh-Poh,” who passed away while I was at Harvard. She was one of the kindest and most-caring people I know. I would also like to remember my sister-in-law Linda Chua, who was taken violently away from us in February of 2000. She was a strong and wonderful woman, a filial daughter to her parents, a wonderful sister to her siblings, and would have been a lovely wife to her fiancé. Linda, you will always be fondly remembered.

My thanks would not be complete if I did not thank my wife of five years now, Janice Chua. I stole her away from her family and friends five years ago to come with me to the United States so that I could pursue my doctoral studies. Since then, she has expanded my horizons on life and become the touchstone of my being. I love her with all of my heart, and am so thankful that she decided to take a chance with me.

Finally, I am grateful to God for all the people that He has brought into my life, for opening my eyes to the world of relationships, and for teaching and showing me the meaning of a love that transcends all understanding. Soli Deo Gloria.

Chapter 1: Introduction

1.1 Motivation

Silicon-based computers just keep getting smaller and faster. As of this writing, Intel is mass-producing silicon chips (the Pentium IV) with feature widths of just 90 nm and which operate above 3 GHz. Eventually however, the current technology of using field-effect transistor (FET) switches will reach a limit, and an alternative technology for computation will have to take over. Extrapolating from the ever-decreasing size of circuits used for computing, it seems plausible that a new technology for computing would be to perform logic operations by shuttling *individual* electrons around in a circuit. This technology is broadly termed "single-electronics," and the creation of an electronic switch actuated by a single electron (described in Chapter 4) is an important step in this direction.

An entirely different way of increasing the speed of computation is to change the paradigm of computing itself. Rather than make smaller and faster switches, the nature of computing can be extended to take advantage of the coherence and entanglement allowed by quantum mechanics. This is the idea behind quantum computation, which has been shown to perform certain algorithms *exponentially faster* than classical computation [Shor, 1997; Grover, 1997]. One possible realization of a quantum computer (in semiconductor) is the proposal by Loss and DiVincenzo [1998], in which calculations are performed by flipping and entangling the spins of individual electrons. The first small steps toward this are described in Chapter 5, and much interesting work remains to be done in this new field of experimental quantum computation.

1.2 Overview of the Thesis

Chapter 2 introduces semiconductor quantum dots to those unfamiliar with them, and describes two-dimensional electron gases (2DEGs), quantum point contacts (QPCs), quantum dots, and the Coulomb blockade phenomenon in quantum dots. A 2DEG, which is a two-dimensional conductor fabricated in a “sandwich” (heterostructure) of different semiconductors, is the starting point for building reduced-dimensioned electronic devices. QPCs further restrict the flow of electrons in the 2DEG to one dimension, and quantum dots confine the electrons in the 2DEG to zero dimensions. The Coulomb blockade phenomenon is the dominant effect governing the electrical behavior of quantum dots, and a simple model is described.

Chapter 3 covers the experimental methods and equipment employed in the study of semiconductor quantum dots, and includes semiconductor fabrication techniques for making quantum dot devices, electronics hardware and software used in measuring the behavior of these devices, and cryogenic equipment used to cool the devices. The fabrication tips and tricks I used for building the devices described in this thesis are detailed, and several useful recipes are listed. An overview of the measurement process is given and techniques for maximizing signal-to-noise are discussed. Descriptions of how to operate the custom-built electronics boxes and custom-written software are made, and an overview of how each works is given (the details of each are found in the Appendix). The cryogenic techniques for creating the low temperatures required for observing Coulomb blockade are also described.

Chapter 4 presents experiments on strongly capacitively-coupled quantum dots, and describes how they were used to make a single-electron switch. The fabrication of the device, which increased capacitive-coupling an order of magnitude over previous dots, is detailed. A hexagonal pattern of Coulomb blockade peaks was found, which could be explained by the capacitive charging model of the Coulomb blockade. A formula connecting the hexagonal pattern and the amount of capacitive-coupling between the dots is derived, and a comparison with tunnel-coupled dots is made. The operation of the strongly capacitively-coupled dots as a single-electron switch is described, and its switching properties are explored. The abruptness of the conductance switching in one dot is found to be determined by how well charge is quantized in the other dot. Fits of different switching lineshapes were made, and good agreements were found.

Chapter 5 talks about few-electron quantum dots, and how they could be used to implement a semiconductor-based quantum computer. That the dots should contain just one electron is important to building a quantum computer, and finite-bias Coulomb blockade data and a three-dimensional scan of gate voltages are presented as evidence that this can be achieved in our few-electron dots. The Zeeman energy of electrons in a few-electron dot was measured, and no deviation of the *g*-factor from that of electrons in bulk GaAs was detected. Tunnel-coupling between dots, another essential ingredient in the construction of a semiconductor quantum computer, was also demonstrated.

Chapter 6 concludes the thesis, and possible future experiments are discussed.

Chapter 2: Background

This chapter introduces the fundamental concepts needed to understand what a quantum dot is, and describes the Coulomb blockade phenomenon, which is the dominant effect controlling the electrical properties of a quantum dot. Section 2.1 describes the two-dimensional electron gases in which our quantum dots are built, Section 2.2 describes quantum point contacts, which form the tunnel barriers to our quantum dots, and Section 2.3 describes the quantum dots. The Coulomb blockade is discussed in Section 2.4, and both the zero dc bias (Section 2.4.1) and finite dc bias (Section 2.4.2) Coulomb blockade behaviors are described.

2.1 Two-Dimensional Electron Gases

A two-dimensional electron gas (2DEG) is system of electrons spatially confined in one dimension but free to move about in the remaining two dimensions. The 2DEG we use is grown by molecular beam epitaxy (MBE) in the lab of Art Gossard at U. C. Santa Barbara from GaAs and $Al_xGa_{1-x}As$ (Figure 2.1). The bandgaps of GaAs and AlGaAs are different, with the AlGaAs being larger, so at the atomically sharp interface between the GaAs and AlGaAs, a discontinuity in the conduction bandgap is formed and a potential barrier in the conduction band is created. A layer of Si donors deposited in the AlGaAs layer provides free electrons to the material. A large portion of these electrons get trapped at the surface of the heterostructure in order to satisfy the surface states, but the rest fall into the lower lying conduction band of the GaAs layer. The electrostatic attraction between these electrons and their dopant Si atoms pulls the electrons in the GaAs layer toward the AlGaAs layer, but because of the discontinuity at the GaAs/AlGaAs interface, the electrons are unable to surmount the energy barrier and remain trapped in the GaAs layer. The conduction band discontinuity and the electrostatic attraction together form a triangular potential in the z-dimension of the heterostructure that confines the electrons in that dimension. By controlling the density (and hence Fermi level) of the conduction

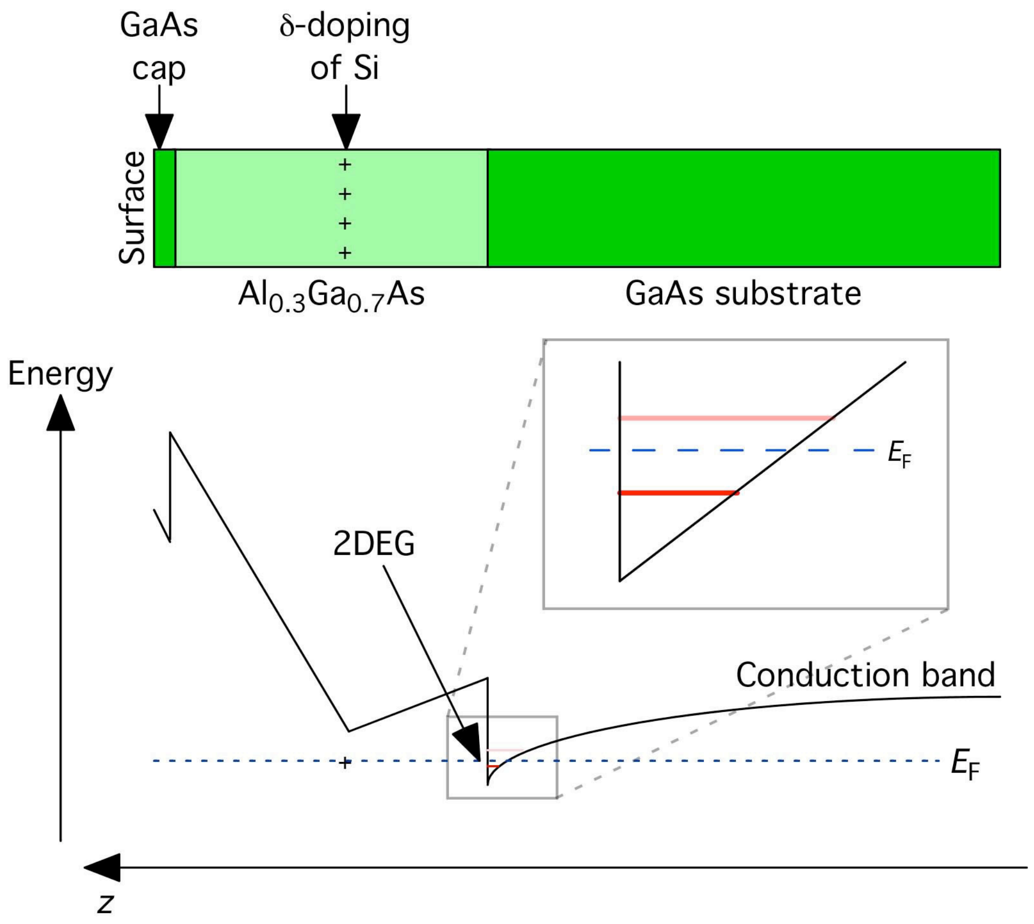


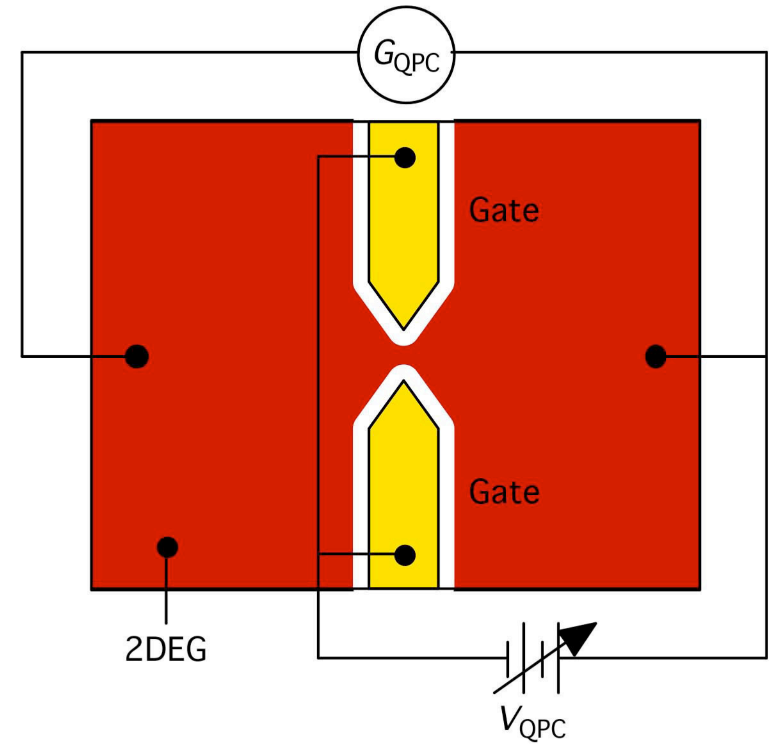
Figure 2.1 Band structure of a GaAs/Al_xGa_{1-x}As heterostructure containing a two-dimensional electron gas (2DEG). The top diagram is a vertical cross-section of the heterostructure and the lower diagram is its corresponding conduction band. The horizontal axis of both diagrams is the *z*-dimension as indicated at the bottom of the Figure. Bandgap differences between GaAs and AlGaAs cause discontinuities in the conduction band. Electrons from donors in the AlGaAs layer fall into the lower-lying GaAs conduction band (at right). These electrons are pulled electrostatically back toward the donors however, but due to the bandgap discontinuity, they become trapped in the *z*-dimension. The electrons are free to move in the other two dimensions and form the 2DEG.

electrons, only the first quantum mechanical mode in the z-dimension is populated and the conduction electrons effectively become two-dimensional.

The lattice constants of GaAs and the alloy $Al_xGa_{1-x}As$ are the same to 0.04% (for $x = 0.3$), so very little mechanical stress develops at the interface between the two semiconductors. The stress-free heterointerface makes for a defect-free interface and consequently high-mobility conduction electrons. Typical mobilities and densities for our 2DEGs are 300,000 to 500,000 cm/Vs and 3 to 4×10^{11} cm^{-2} respectively (measured without illumination, which would inject additional carriers into the 2DEG). A square-well heterostructure (as opposed to the accumulation layer heterostructure just described) can be grown by sandwiching the GaAs layer between two layers of AlGaAs. This creates two hard-wall potentials that better-define the extent of the electrons in the z-dimension, at the expense of lower electron mobility. However, for research in quantum dots where only a square micron of 2DEG is typically of interest, mobility may not be so important a factor and may be traded for better confinement of electrons by using a square-well 2DEG.

2.2 Quantum Point Contacts

The dimensionality experienced by electrons can be further reduced by constricting them with a quantum point contact (QPC). A QPC is made by applying negative voltages to metal gates just above the 2DEG on the surface of the heterostructure (Figure 2.2). The negative voltage relative to the 2DEG depletes conduction electrons in the 2DEG below and to the sides of the gates, and forces electrons to flow through the narrow constriction as shown. By varying the voltage on the metal gate, a wider or narrower constriction can be made. The conductance (1/resistance) across the QPC is measured with very small AC voltages ($eV \sim kT$) using lock-in techniques (see Chapter 3.3) so transport of electrons occurs essentially at the Fermi level. The conductance of the QPC as a function of gate voltage is also shown in Figure 2.2 and



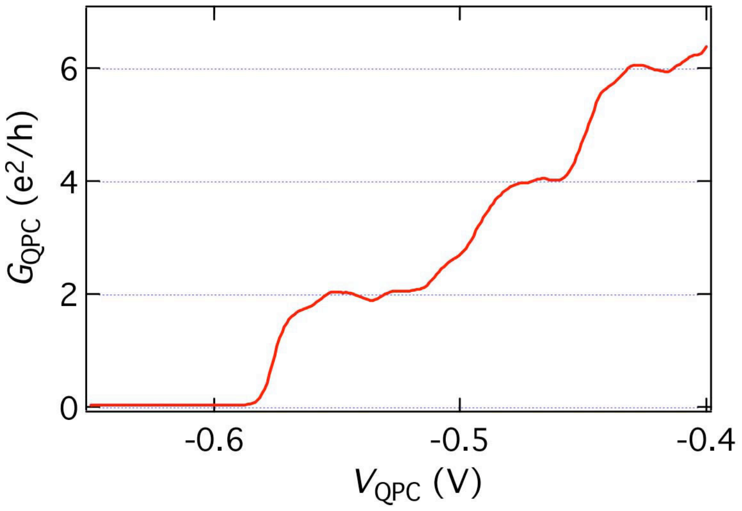


Figure 2.2 Quantum point contact (QPC) formed by surface gates. Electrons in a 2DEG just below the surface of a heterostructure are depleted by negatively-charged metal gates on the surface, forming a narrow constriction called a quantum point contact. The width of the QPC is adjustable by varying the voltage on the gates V_{QPC}, and gives a conductance trace as shown in the graph below. Conductance is quantized in units of $2e^2/h$ from each new transverse mode allowed through the QPC as V_{QPC} becomes less negative and the constriction becomes wider.

exhibits distinct plateaus in conductance in multiples of $2e^2/h$ [van Wees *et al.*, 1988; Wharam *et al.*, 1988]. This is due to electrons being confined by the QPC and forming subbands. Each transverse mode through the QPC carries a quantum of conductance $2e^2/h$ (the factor of two comes from spin degeneracy), so when a new subband is allowed through the QPC, the conductance increases by $2e^2/h$ [Landauer, 1959; Buttiker, 1986; Buttiker, 1988]. However, if enough negative voltage is applied to the QPC so that the first subband is above the Fermi level, then electrons can only tunnel across the QPC and the QPC then acts as a tunnel barrier.

2.3 Quantum Dots

Reducing the dimensionality of electrons further requires confining electrons in all three spatial dimensions. This can be achieved with the configuration of gates shown in Figure 2.3 to form a puddle or island of electrons. The puddle of electrons is known as a quantum dot, and is connected to the rest of the 2DEG via QPCs acting as tunnel barriers, which allow electrical measurements of the dot to be made. The gaps between the other gates are narrower than the openings of the QPCs, and electrons in the dot cannot leak out through those gaps.

While the energy level of electrons in the quantum dot are quantized because of the confinement, another more significant effect is that the *number* of electrons is quantized. Electrons cannot flow in and out of the dot in a continuous manner as they can in a QPC, and a price in energy must be paid for each electron that enters the dot. This energy is called the charging energy $E_C \equiv e^2/C_\Sigma$, where C_Σ is the self-capacitance of the dot, and comes from a classical capacitance effect [Meir *et al.*, 1991; Beenakker, 1991]. When electrons in the leads do not have sufficient energy to overcome the charging energy, they cannot enter the dot and current through the dot is blockaded, giving rise to the Coulomb blockade phenomenon [several good review articles include Beenakker

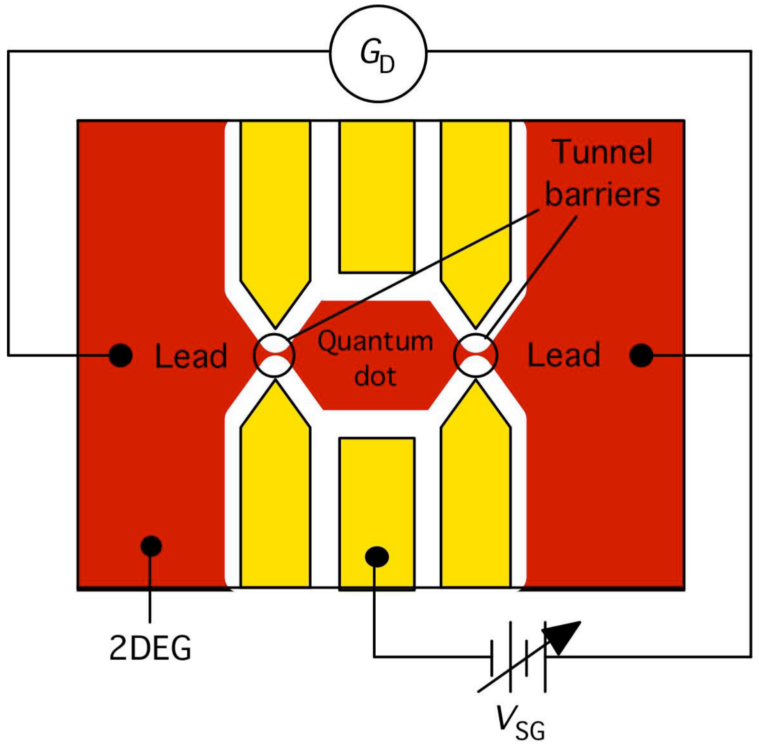


Figure 2.3 A quantum dot formed by surface gates. Electrons in the 2DEG are confined in all dimensions by the configuration of gates shown above to form a quantum dot (the gaps between gates are too narrow for electrons to be in). Two tunnel barriers to the dot are formed by QPCs so that the conductance of the dot can be measured. Because electrons in a quantum dot are confined in all three spatial dimensions, the number of electrons on the dot is quantized, leading to quantum dots being called artificial atoms.

and van Houten, 1991; Kastner, 1992, Grabert and Devoret, 1992, Sohn *et al.*, 1996; Likharev, 1999].

2.4 The Coulomb Blockade

There are two conditions that must be satisfied in order to observe Coulomb blockade. Firstly, the number of electrons on the dot must be well quantized. This means that the uncertainty in an electron's energy ΔE due to a short electron dwell-time Δt must be smaller than the charging energy [Tinkham, 1996]

$\Delta E \sim h/\Delta t << e^2/C_\Sigma$.

This implies that both tunnel barriers to the dot must have low tunneling rates, or equivalently, low conductances

$G_{QPC} << 2e^2/h$.

Secondly, the thermal distribution of electrons near the Fermi level in the leads must be smaller than the charging energy

$kT << e^2/C_\Sigma$.

2.4.1 Zero DC Bias Coulomb Blockade

A useful pictorial tool for appreciating the Coulomb blockade phenomenon is Figure 2.4. When the highest filled state of the dot plus the charging energy is higher than the Fermi level in the leads E_F, no electrons can hop on or off the dot and current is blockaded [Figure 2.4(a)]. However, when the electrostatic potential of the dot is adjusted using a gate that is capacitively coupled to the dot (conventionally called the sidegate), the energy levels in the dot are shifted up or down to bring the empty dot state into line with the Fermi level of the leads and current flows through the dot [Figure 2.4(b)]. This gives rise to a conductance versus sidegate voltage trace as shown in Figure 2.4(c) where a peak in conductance is observed everytime a dot state is brought in line with the Fermi level. The lineshape of each peak in the conductance trace is the convolution of the derivative of the Fermi function in the leads with the energy uncertainty of the electronic

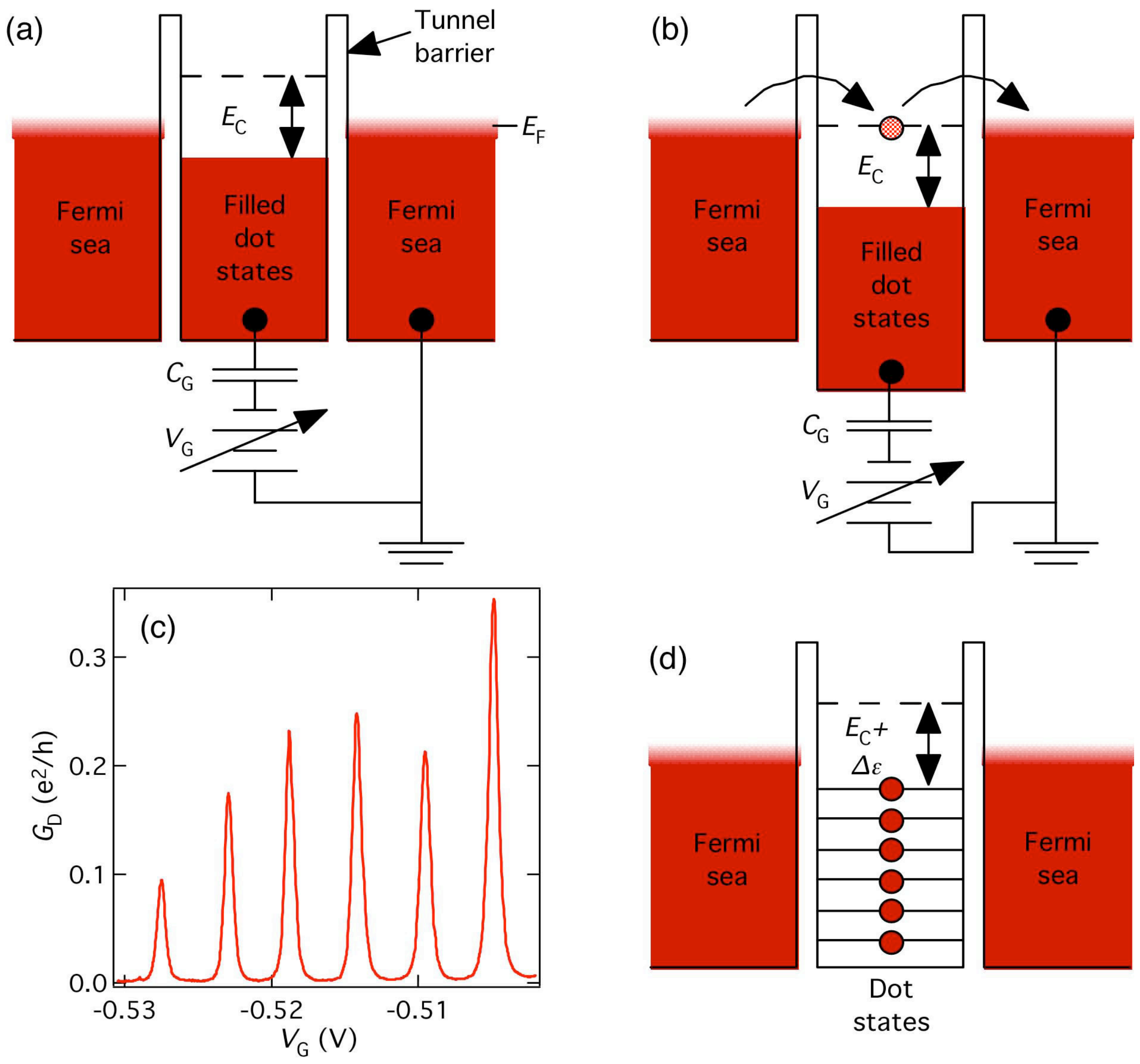


Figure 2.4 These diagrams explain the Coulomb blockade phenomenon. (a) Energy diagram showing the Fermi sea in the leads and the energy levels in the dot. The energy required to add an electron to the dot due to electrostatic repulsion is E_C, the charging energy. If this energy cannot be paid, no electron can enter the dot and current is blockaded. (b) By adjusting the energy levels in the dot with a sidegate voltage V_G, an empty state can be brought in line with the Fermi level of the leads. When this happens, electrons can hop on and off the dot to carry current. (c) Every time a new state is brought in line with the Fermi level of the leads, a peak in conductance is observed, creating the conductance trace as shown. (d) If the energy from quantum confinement $\Delta\varepsilon$ is included, the electron addition energy becomes $E_C + \Delta\varepsilon$.

states in the dot. For a dot in which kT is larger than the energy uncertainty ΔE (*i.e.* in the thermally-smeared regime), the conductance lineshape will take the form of the derivative of the Fermi function [Beenakker, 1991; Meir *et al.*, 1991]

$$G_{\mathrm{D}}(V_{\mathrm{G}}) = G_{\mathrm{D0}} \cosh^{-2}\left(\frac{e(V_{\mathrm{G}} - V_{\mathrm{G0}})C_{\mathrm{G}}/C_{\Sigma}}{2kT}\right) \tag{2.1}$$

where G_{D0} is the peak height, V_{G0} is the location of the peak, C_{G} is the sidegate–dot capacitance, and C_{Σ} is the dot self-capacitance. Note that the energy has been scaled by $C_{\mathrm{G}}/C_{\Sigma}$, which acts as a "lever-arm" converting sidegate energy eV_{G} to dot energy. For a dot in which the energy uncertainty ΔE is larger than kT (*i.e.* in the lifetime-broadened regime), the conductance lineshape will take the form of a Lorentzian [Stone and Lee, 1985]

$$G_{\mathrm{D}}(V_{\mathrm{G}}) = G_{\mathrm{D0}} \frac{\Delta E^2}{\Delta E^2 + \left(e(V_{\mathrm{G}} - V_{\mathrm{G0}})C_{\mathrm{G}}/C_{\Sigma}\right)^2} \tag{2.2}$$

where the parameters G_{D0}, V_{G0}, C_{G}, C_{Σ} and have the same definitions as above.

The picture can be modified to account for quantized energy levels $\Delta\varepsilon$ due to quantum confinement as shown in Figure 2.4(d). The energy required to add an extra electron to the dot (the addition energy) then becomes $E_{\mathrm{C}} + \Delta\varepsilon$. While $\Delta\varepsilon$ depends on the shape of the dot and its confining potential in a complicated manner, a useful estimate can be derived by assuming that the potential profile is parabolic and that the energy levels are spin degenerate. This gives an estimate for the level spacing in a dot as

$$\Delta\varepsilon \sim 2E_{\mathrm{F}}/N, \tag{2.3}$$

where N is the number of electrons in the dot.

2.4.2 Finite DC Bias Coulomb Blockade

A dc voltage V_{DS} can be applied across the quantum dot to make a finite bias Coulomb blockade measurement, as illustrated in Figure 2.5. The dc bias raises or lowers the chemical potential in one lead to gain access to states in the quantum dot.

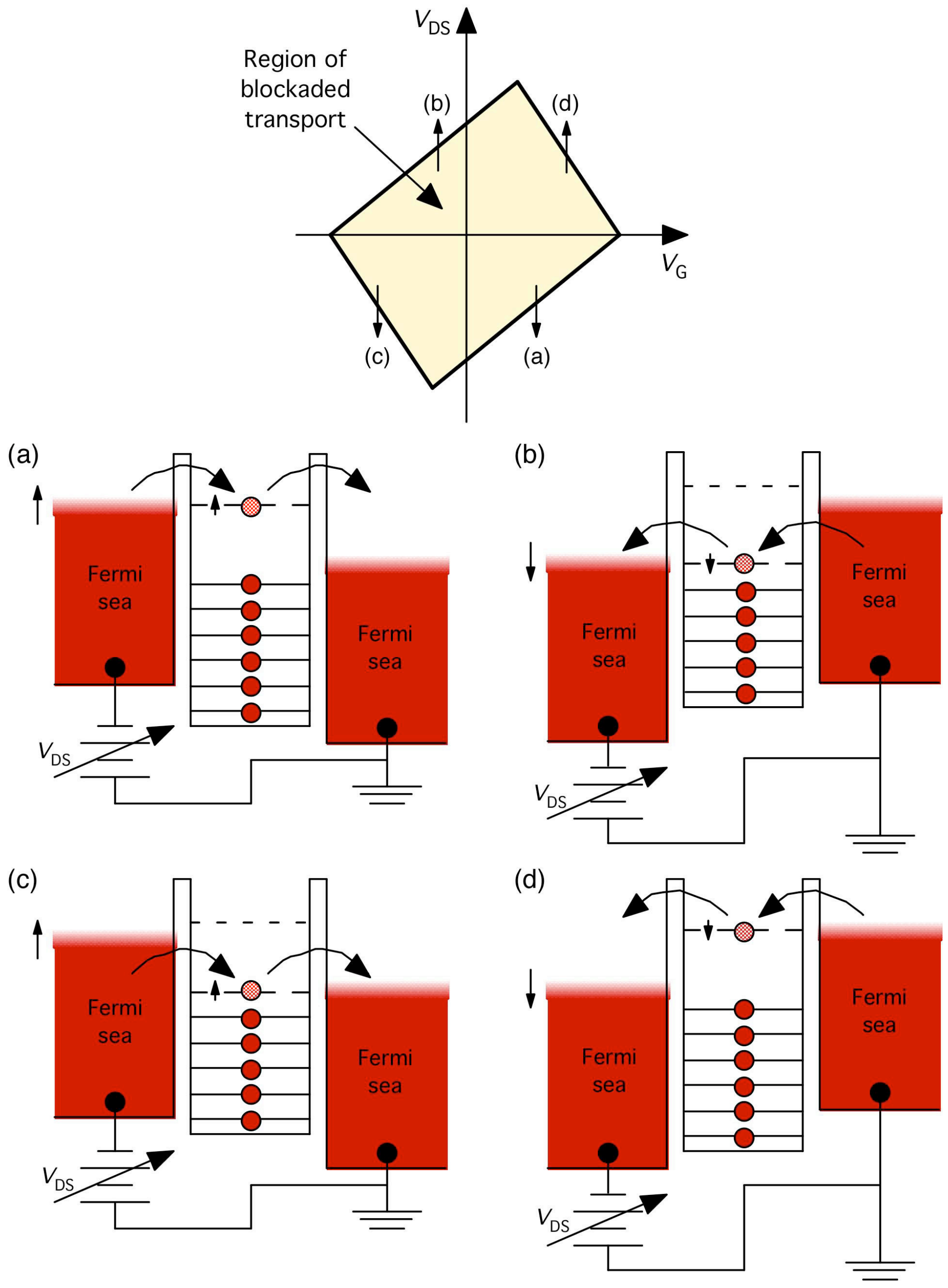
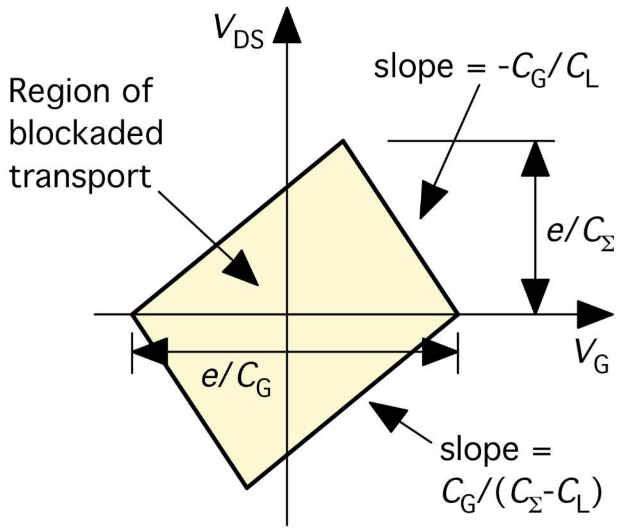


Figure 2.5 These diagrams explain the finite bias Coulomb blockade. In addition to adjusting sidegate voltage V_G, a dc bias across the dot V_{DS} can also be varied. The two voltages define a diamond-shaped region in which current through the dot is blockaded. The blockade is overcome along side (a) of the diamond by raising the left lead (lowering V_{DS}) to access the empty dot state. Along side (b), the blockade is overcome when the left lead is lowered (V_{DS} is raised) below the lowest occupied dot state. Along sides (c) and (d), capacitive coupling from the left lead to the dot raises or lowers the dot states (small vertical arrows) so that dot states are accessed by the *right* lead.

The Coulomb blockade of current through the dot can therefore be lifted by application of a dc bias. A diamond-shaped region of blockaded current is observed as a function of sidegate and bias voltages. The processes that give rise to the four sides of the Coulomb blockade diamond are illustrated in Figures 2.5(a)–(d). For the purposes of explaining the diamond shape, we assume that the dc bias across the dot is placed on the left lead of the dot. The dot states are also shifted in energy together with the left lead, albeit at a fraction C_L/C_Σ, because the left lead is capacitively coupled to the dot (just like the sidegate) with a capacitance C_L. The shifts in energy of the left lead and dots states are indicated by the vertical arrows in Figures 2.5(a)–(d). Along side (a), a new energy level in the dot is accessed by the left lead as it is raised in energy (dc bias voltage is decreased). Along side (b), a previously filled dot state is revealed as the left lead is lowered (dc bias voltage is increased). On side (c), a previously filled dot state emerges above the *right* lead due to capacitive coupling of the left lead to the dot states. Along side (d), a new dot state becomes accessible to the *right* lead as the dot states are lowered by capacitive coupling to the left lead. The maximum value that $|V_{DS}|$ can take before transport across the dot occurs is $e|V_{DS}|_{max} = E_C + \Delta\varepsilon = e^2/C_\Sigma + \Delta\varepsilon$. Since there is no scale factor between $e|V_{DS}|_{max}$ and the addition energy, a finite bias Coulomb blockade measurement allows the charging energy, and hence a dot's self-capacitance, to be measured directly (assuming $\Delta\varepsilon$ is zero). The finite bias Coulomb blockade measurement also allows other capacitances to be obtained, like the sidegate–dot capacitance C_G and the lead–dot capacitance C_L, as illustrated in Figure 2.6.

Current transport through multiple dot states can occur if several dot states lie between the chemical potentials of the left and right leads. When a new dot state becomes available for electron transport, a new peak in the finite bias conductance is observed (assuming that the level spacing is larger than the thermal distribution of electrons, $\Delta\varepsilon >> kT$). However, as long as only one electron enters the dot at a time, only one charging energy need be paid, so the difference in drain-source bias between Coulomb



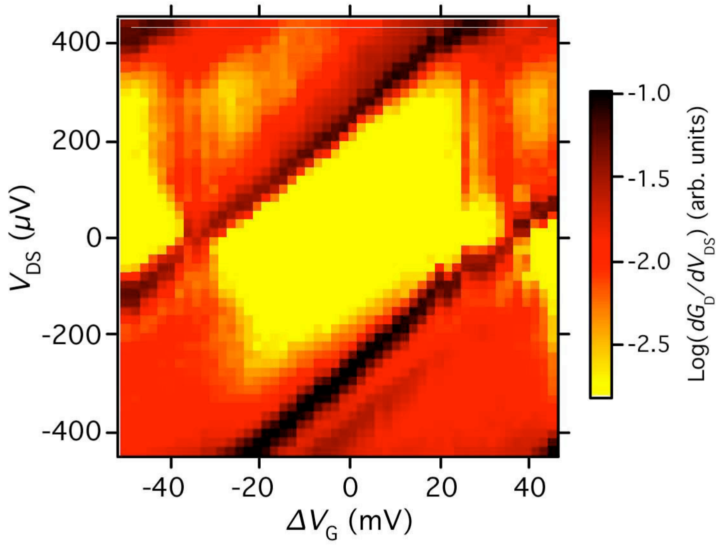


Figure 2.6 Coulomb blockade diamond dimensions and data. The dimensions of the Coulomb blockade diamond reveal many parameters about a quantum dot, the most important of which are the charging energy $E_C = e^2/C_\Sigma$, and the sidegate–dot capacitance C_G. Below is data illustrating what an actual Coulomb blockade diamond looks like.

blockade peaks is just the level spacing $e\Delta V_{\mathrm{DS}} = \Delta\varepsilon$. This technique of applying a dc bias across the dot therefore allows the level spacings to be measured directly, and is known as excited state spectroscopy.

Chapter 3: Experimental Methods

This chapter consists of four parts. The first part (Section 3.1) describes the techniques used in the fabrication of quantum dot samples. The second part (Section 3.2) delves into the measurement techniques used to probe the electronic properties of the samples, and describes some of the custom-built electronics boxes that were used. The third part (Section 3.3) covers custom-written software used to simulate the devices as well as acquire sample data. And the fourth part (Section 3.4) covers cryogenic techniques that were used to cool the samples down to the mK temperatures required for the experiments.

Within Section 3.1, steps for sample fabrication are detailed, including how to cleave chips from the 2DEG heterostructure (Section 3.1.1), clean the chips (Section 3.1.2), perform electron-beam lithography to write patterns on the chips (Section 3.1.3), fabricate metal structures (Section 3.1.4), anneal the ohmic contacts (Section 3.1.5), mill features in the sample (Section 3.1.6), make wirebonds from the sample to the chip carrier (Section 3.1.7), and keep the sample safe after fabrication (Section 3.1.8).

Section 3.2, covering measurement electronics, gives an overview of the measurement process (Section 3.2.1), describes the Ramper box (Section 3.2.2) and BiasDac box (Section 3.2.3) that supply the gate voltages for a quantum dot, discusses the analog input electronics for reading data into the computer (Section 3.2.4), and describes the differential summer box that sums two voltages without connecting their grounds together (Section 3.2.5).

In Section 3.3, which describes the custom software used our research, Section 3.3.1 discusses various Macintosh application programming interfaces and their relevance to the lab's on-going transition to the Mac OS X operating system, Section 3.3.2 covers the Poisson Solver program that performs electrostatic simulations of quantum dots, Section 3.3.3 gives some tips and tricks for using the Poisson Solver, Section 3.3.4 goes into the AFM program which I modified to perform quantum dot experiments, and Section 3.3.5 explains some tips and tricks for using the AFM program.

In the final section, Section 3.4, the cryogenic techniques for cooling the samples are described, and cover the operation of He3/4 dilution refrigerators (Section 3.4.1) and the wiring that was re-made in the Kelvinox 100 fridge (Section 3.4.2).

3.1 Sample Fabrication

When the 2DEG heterostructure is shipped to us by our collaborators at U. C. Santa Barbara, they arrive in pieces that can measure 3” along the largest sides (since they descended from 3” wafers). To turn these macroscopic chunks of heterostructure into the meso- and nano-scale devices that we experiment on, some or all of the following processing steps need to be performed: cleaving, cleaning, electron-beam lithography, metallization, annealing, ion-milling, and wirebonding. A typical simple quantum dot device will involve the following sequence of steps:

Cleaving
Cleaning
Electron-beam lithography
Metallization
Annealing
Cleaning
Electron-beam lithography
Metallization
Cleaning
Wirebonding

A picture of a completed device at various magnifications is shown in Figure 3.1.

3.1.1 Cleaving

The first step of fabrication is to cut small chips from the parent heterostructure. Chips around 2 to 3 mm a side are typically cleaved. Larger pieces are wasteful of heterostructure material since the JEOL scanning electron microscope is unable to write

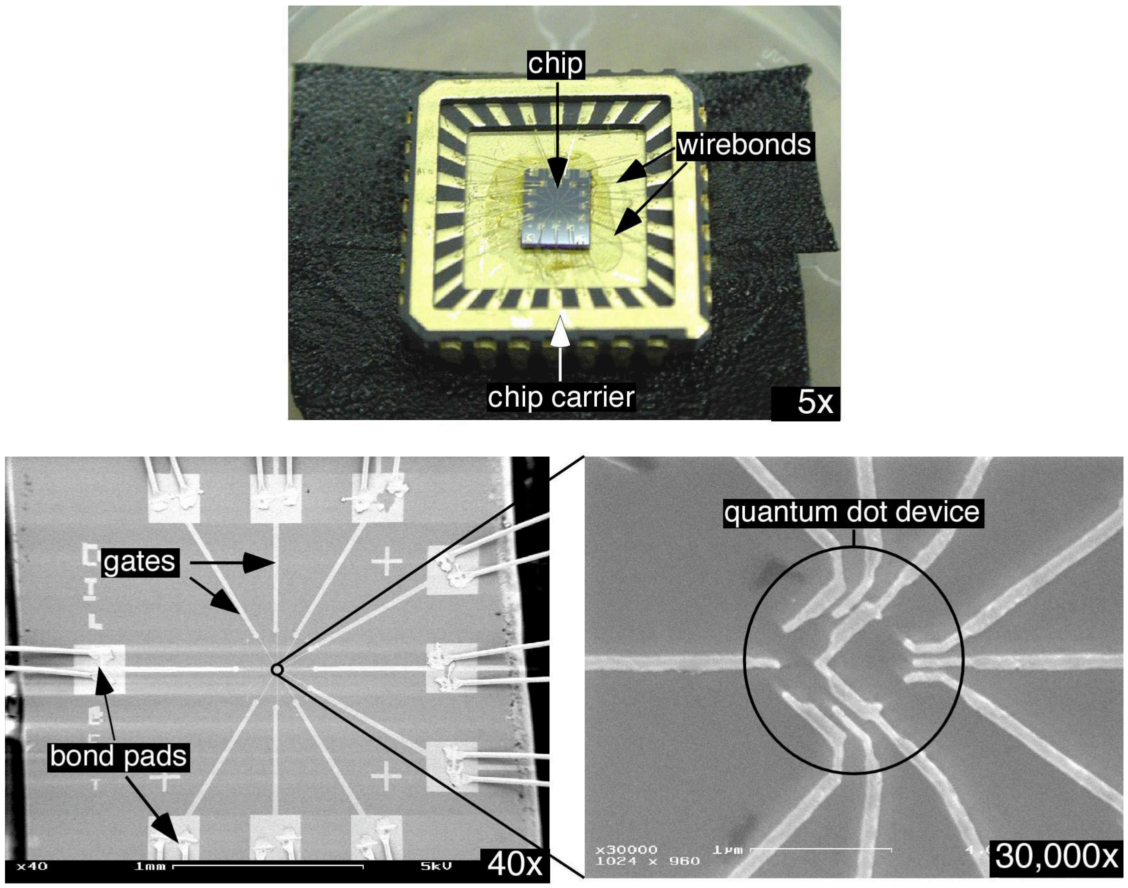


Figure 3.1 Pictures showing completed samples at 5×, 40× and 30,000× magnification. At 5×, the chip carrier, chip, and wirebonds are visible. At 40×, the bonding pads, large gates, and alignment marks become visible. At 30,000×, the quantum dot device is finally revealed.

patterns larger than 4 mm across, while smaller chips become very hard to handle. First, it is absolutely crucial to determine which side of the heterostructure is the surface containing the 2DEG layer, and which side is the substrate. Generally, the surface contains one or more semicircular clip marks on the edge left by clips holding the wafer down during material growth. The substrate side is generally dirty, possibly scratched, and contains a serial number etched into a flat of the wafer. If none of these identifying marks are visible, then either a call to the growers at U. C. Santa Barbara is required, or contacts to the 2DEG must be made to experimentally verify which side of the heterostructure contains the 2DEG.

Once the surface side is ascertained, a strip 2 to 3 mm wide is cleaved from the heterostructure [Figure 3.2(a)] using the scribe-and-snap process that will be described for cleaving a chip from the strip. After the strip is cleaved, I mark the back of the strip with a diamond scribe so that I can tell the substrate side from the surface [Figure 3.2(b)]. Several scribes about 3 mm apart are made at a 45° angle to the edge of the strip so as not to be parallel with the crystal axes of the heterostructure. This is a delicate procedure where I hold one end of the strip in the air with a pair of tweezers and rest the other end on a piece of Tekwipe on the table. Scribes are then made with the diamond scribe in the weak hand. The scribe-and-snap process for cleaving a chip from the strip is as follows. Scribe a line on the surface of the heterostructure with the diamond scribe and a ruler wrapped in Teflon tape (to prevent scratching the surface) [Figure 3.2(c)]. The scribe should be made using a firm but light pressure. The scribing process generates GaAs fragments and dust, and blowing the strip with nitrogen after each scribe is strongly advised. The strip with the scribe mark is now placed on a slightly elevated platform (a piece of filter paper folded in half is traditionally used) so that the desired chip hangs off the edge of the platform [Figure 3.2(d)]. Align the scribe mark to the edge of the platform, and press down on either side of the scribe mark with two Q-tips, being careful not to scratch the surface with the Q-tips. The location of both Q-tips should be near the

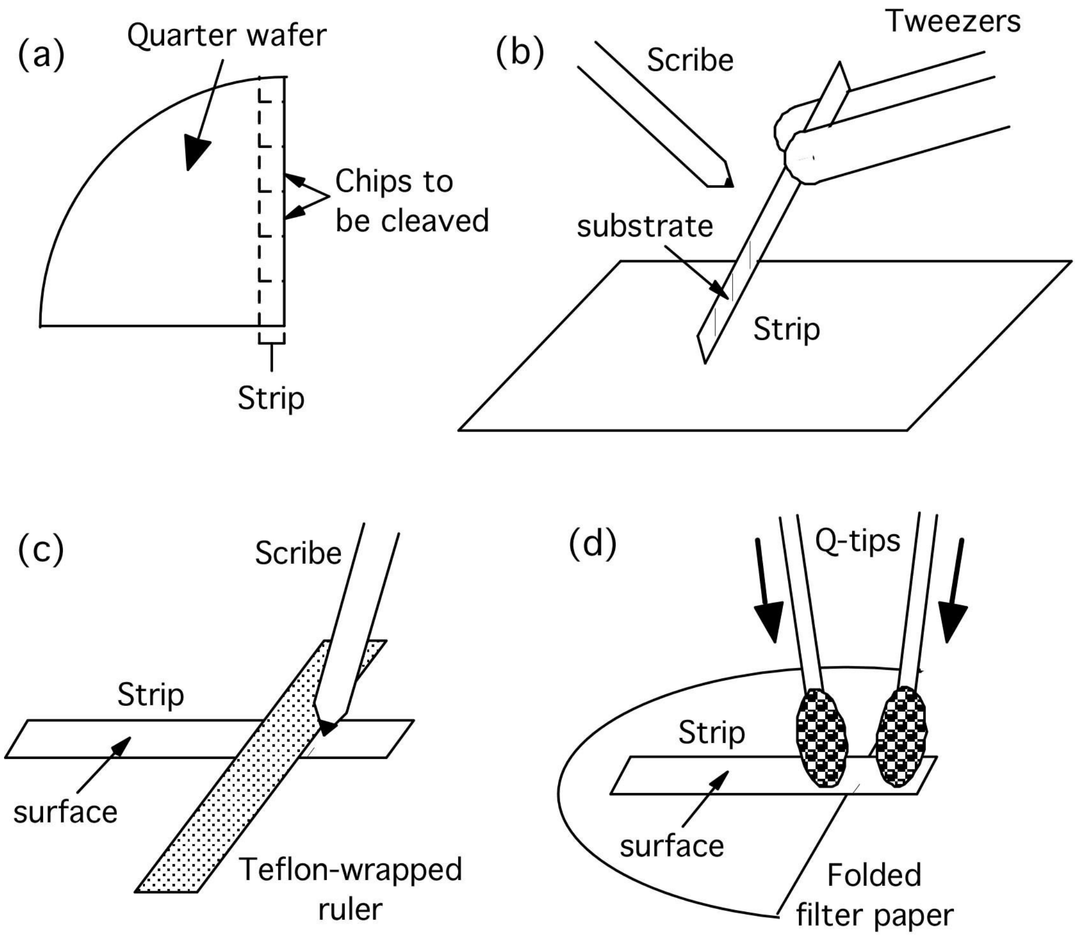


Figure 3.2 Schematic view of the cleaving process. (a) First, a strip is cleaved from the parent wafer. (b) The substrate side of the strip is marked with a diamond scribe for identification. (c) A small scribe about 0.5 mm long is made on the surface of the strip to define the size of the chip. (d) The chip is then cleaved from the strip by pressing down on either side of the scribe mark with Q-tips.

edge with the scribe mark and pressure should be increased until the chip snaps off. Blow any dust off the strip and chip.

Additional chips may be cleaved from the strip in a similar fashion. Four to six chips are normally cleaved into a batch to take advantage of some parallelism in the processing. A good pair of flat-nosed tweezers, like the Techni-Tool model 2A, is highly recommended for handling the small chips. When cleaving the chips from the strip (rather than the strip from the wafer), I generally make scribes only about 0.5 mm long that do not span the width of the strip. This is because the scribe is along a crystal axis and a clean cleave can generally be made to obtain perfectly straight edges.

Some people are fearful of the arsenic contained in the GaAs dust generated in the cleaving step and wear a dust mask while performing this step. While it is said that arsenic is very strongly bonded to gallium in GaAs and does not represent a health issue, I take the precaution of cleaving in the fume hood so that any dust is drawn into the hood. Wearing a mask while cleaving outside the fume hood does not make sense to me as the particles then remain in the room and can be inhaled by others.

3.1.2 Cleaning

The goal of the cleaning process is to remove chemical and particulate contaminants from the surface of the wafer that can interfere with the processing of the chip. Chemical contaminants include grease and oil which can cause poor adhesion of metal gates to the surface of the wafer, making it almost impossible to wirebond the sample. Particulate contaminants like dust and carbon particles can clog lithographic patterns in the resist and break up structures that should otherwise be continuous. Cleaning is a mundane part of the fabrication process, but the reward for doing it carefully is consistently successful processing. The standard three-step cleaning process is:

15 minutes in hot trichloroethylene (TCE)
15 minutes of ultrasounding in acetone

5 minutes of ultrasounding in methanol

followed by a blow-drying step. During the cleaning steps, the samples should be sitting in a Teflon basket that is then dipped into the various glass beakers containing the solvents. The Teflon basket is used because it is chemically inert, doesn't scratch the chips, and makes it easier to handle and transfer several chips at a time.

TCE is used to remove grease from the chips and it works best when heated. I normally use a temperature of about 80°C. At this temperature, convection is clearly visible in the beaker, but it is not so hot that boiling occurs. Aluminum foil should be used to cover the beaker as the plastic Parafilm wrap will be dissolved by the TCE vapor. TCE is not very flammable, so one need not worry about it catching fire on the hot plate. It is carcinogenic however, and this cleaning step should be performed inside a fume hood.

The second step of cleaning involves acetone, which is another wonderful solvent. Ultrasounding the samples during this step assists in dislodging particulate contaminants. However, it can also damage poorly adhered or very fine metal structures on the chip. For this reason, lengthy ultrasounding (> 1 s) is discouraged once fine structures have been made. The ultrasounding can also occasionally flip a chip over so that the surface rests against the bottom of the Teflon basket. Check the chips for proper orientation before further processing.

The final step is to clean the chips in methanol. Methanol is a poorer solvent than acetone, but its redeeming quality is that it leaves no residue on the chip when it dries. Even then, it is a good idea to spritz the chips with fresh methanol before drying to ensure that no residue will be left on the chips. Parafilm wraps can be used to cover the beaker for both the acetone and methanol steps.

Drying is done by blowing ultra-pure nitrogen gas at the chip. A filter on the nitrogen gun traps dust particles from the gas cylinder so that particles are not blown onto the chip. The chip itself should be held securely in a pair of tweezers and resting on a layer or two of lint-free Tekwipes while being blown dry to prevent the blast of nitrogen from flipping the chip on the tweezers, or worse, being blown away to land face down on some unclean surface.

3.1.3 Electron-Beam Lithography

Electron-beam lithography is a process by which a pattern is created in a resist on the surface of the chip with an electron-beam. The patterned resist is the mask or template by which structures in or on the chip are made. The resist used is poly(methyl methacrylate), or PMMA, which is also a transparent plastic known as Plexiglass. The many links within the PMMA macromolecule are severed by exposure to the electron-beam (scission) to form simpler compounds that are dissolved by a chemical developer. PMMA is a positive resist, so the areas exposed to the electron beam are also the areas that will form the structure on the chip.

The PMMA resist we use is dissolved in anisole and the liquid solution is applied to the surface of the chip by spinning it on. The resist should be applied as soon as possible after the chips have been cleaned so that minimal contamination occurs. Two different solutions of PMMA are used: 950K PMMA (2% PMMA in anisole by weight), and 495K PMMA (2% PMMA in anisole by weight). The 950K solution contains larger and heavier PMMA molecules (950,000 daltons vs. 495,000 daltons) and has a higher contrast than the 495K PMMA. By applying the 495K solution followed by the 950K solution, a bi-layered resist is formed as shown in Figure 3.3(a). Due to the lower contrast of the 495K PMMA and the increased backscattering of electrons at the surface of the chip, the two-layer resist will exhibit wider feature widths in the bottom layer than in the top layer, creating an undercut [Figure 3.3(b)]. This undercut is important when evaporating metals because it increases the likelihood that the metal deposited on the chip is not connected to the unwanted metal deposited on top of the resist [Figure 3.3(c)], and a clean lift-off of the unwanted metal can be achieved [Figure 3.3(d)].

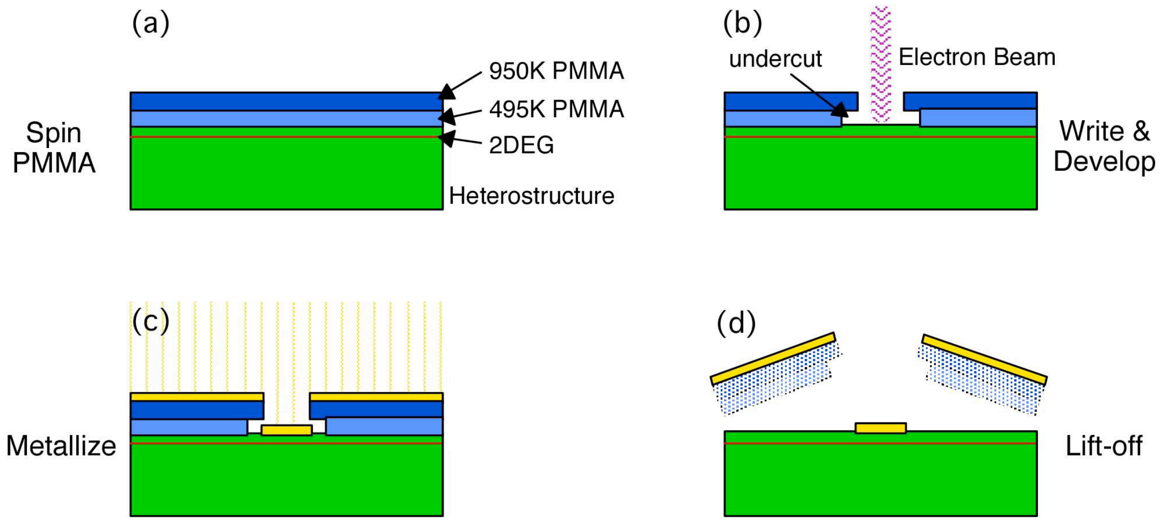


Figure 3.3 Schematic view of the electron-beam lithography process. (a) First, PMMA resist is spun on the chip. (b) Writing a pattern with an electron beam followed by development leaves a patterned structure in the resist. An undercut is formed because of the different weights of PMMA used, and (c) metal can be evaporated on to the chip without connecting to the unwanted metal on top of the resist. (d) After dissolving the remaining resist in a solvent, a metal pattern remains on the chip.

The chips are held to the spinner by a vacuum system and the chuck with the smallest hole should be used since our chips are very small. Be sure to test the vacuum system before committing a chip to it as PMMA solution gets sucked into the vacuum and over weeks or months, the vacuum tubing can get clogged. A glass pipette is used to deposit the PMMA onto the chip and it should be cleaned with methanol or at least blown with compressed air to remove particulate contaminants. A single drop of PMMA solution is deposited onto the chip with the spinner set at 500 rpm. Make sure the drop of PMMA does not contain a bubble, which can create a large defect. The spinner is then spun up to 4000 rpm for 40 seconds to evenly coat the chip. After spinning a PMMA layer on, the chip is baked at 180°C on a hot plate for at least 2 minutes to drive the remaining solvent away. Once the layer is baked, another layer of PMMA can be spun on and baked. Each layer of 495K PMMA will provide about 600 Å of resist under the above conditions (2% PMMA in anisole by weight, spun at 4000 rpm for 40 seconds). If more than 600 Å of 495K PMMA resist is desired (*e.g.* for evaporating > 600 Å of metal), another layer of 495K PMMA can be spun on before the final 950K PMMA layer. The tri-layer resist also gives added protection against pinhole defects in the resist, and is especially important in preventing ohmic contacts from being formed randomly across the sample and shorting gates to the 2DEG [Eriksson, 1997].

The shelf life of PMMA resist is specified to be 13 months, so spun chips may be safely stored for a while until needed. Be careful to keep coated chips away from organic solvents and their vapors lest the resist be damaged or dissolved. A simple check by eye as to whether a chip has a bi-layer or tri-layer resist is this: the bi-layer resist appears dark blue, while the tri-layer resist appears light green (at least for resists made with the above recipe). Note the health hazards associated with the solvent anisole. While less hazardous than chlorobenzene, anisole is nevertheless to be considered unhealthy. Some people (former students especially) recommended spinning while wearing an organic chemical respirator. I think the small drops of anisole that we work with do not present such a grave danger, so I personally do not take such precautions. On the other hand, the reader should be aware that the hoods in which the spinners reside are not fume hoods but laminar flow hoods, so the air and anisole vapors are recirculated rather than vented to the atmosphere.

The next step is to write the desired pattern using an electron-beam writer. As of this writing, there are two options – the older, tried and true JEOL/Nabity system, and the new, fancier Raith system. I have tried both, and it is my opinion that for quantum dot devices, the JEOL is more convenient, less prone to failure, and just as good as the Raith system. The JEOL/Nabity system consists of the JEOL 6400 scanning electron microscope (SEM) controlled by the Nabity Pattern Generation System (NPGS). NPGS is a software and hardware addition to the JEOL SEM that turns it into an electron-beam writer. The advantage of the JEOL/Nabity system is that the filament can produce very large currents (> 200,000 pA) useful for writing large, coarse patterns (like bonding pads) very quickly. This is in contrast to the limited current output of the Raith (~200 pA), which can be a thousand times slower for writing large features than the JEOL! The advantages of the Raith are that it can stitch a complex pattern over a large area together, and that writing is largely automated. The resolution of electron-beam writing is determined in part by the magnification at which one is writing – the higher the magnification, the higher the resolution (up to a point). However, the higher the magnification, the smaller the field-of-view becomes. If one wants to write very fine features over more than one field-of-view, then the Raith is the only option because it can stitch different fields-of-view together, thanks to its laser-interferometric stage. The JEOL, in contrast, can only write features in one field-of-view; the leads connecting the fine features to the bonding pads at the edge of the chip must be written at progressively coarser resolutions and wider fields-of-view. Nevertheless, the JEOL is recommended because there is less setup time required, has a much larger maximum writing current, and because quantum dot devices have very small active areas where fine detail is required. For detailed operating instructions for the JEOL/Nabity system, I refer the reader to Duncan [2000] or Chen [2001].

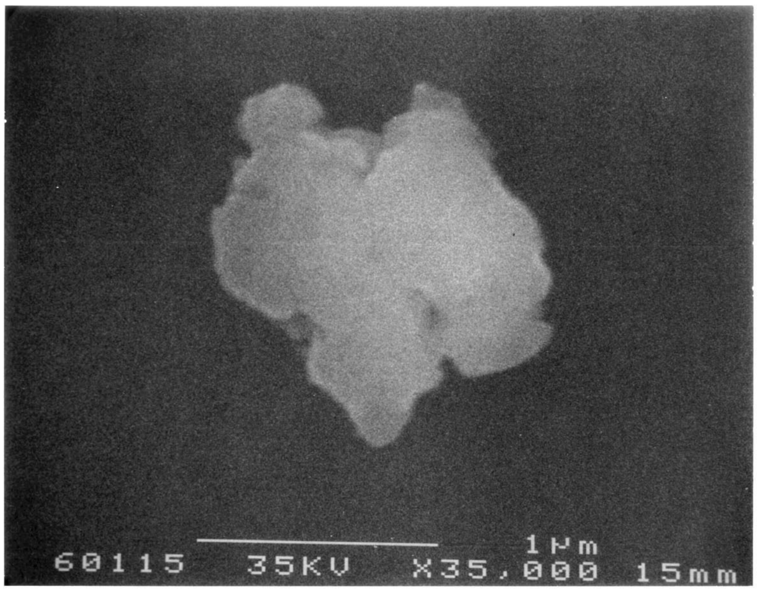
The realization that the active areas of quantum dot devices are very small allows one to optimize the highest magnification for high resolution, while optimizing the rest of the magnifications for speed. Electron-beam lithography is a sequential process, unlike optical lithography where the entire pattern from small to large is written at once. When a pattern is written by an electron-beam, the pattern is broken up into pixels and each pixel is exposed individually. In the NPGS software, the user gets to choose the pixel size, or point spacing. The smaller the point spacing, the higher the resolution but the slower the writing because of the overhead from moving from pixel to pixel. Thus I have chosen small pixels for writing the all important active area at the highest magnification, and have chosen bigger pixels for the less-critical leads. Another optimization I have made is with the current used for writing. Low currents are used to write fine features because of the smaller spot sizes, while the highest currents are used for writing leads and bonding pads because of the quicker exposure. The following then is the highly optimized recipe I developed for electron-beam lithography:

	Fine Features	**Medium Features**	**Coarse Features**
Magnification	2,000×	200×	20×
Current	4.4 pA	3,300 pA	220,000 pA
Point Spacing	10 nm	200 nm	2,000 nm
Exposure	330 nC/cm^2	615 nC/cm^2	980 nC/cm^2
Field of View	44 μm	440 μm	4,400 μm

With these settings, the 20× magnification features (large leads and bonding pads) are typically completed in about 2 minutes compared with about 15 minutes using older recipes.

The secret to great fabrication however, is in optimizing the focus and astigmatism of the electron-beam. Just like regular optics, a beam of electrons suffers distortion, the lowest order effects being focus and astigmatism. In order to obtain the tightest and roundest beam, focus and astigmatism must be adjusted before writing every chip. The PMMA resist presents a featureless and low contrast surface for adjusting focus and astigmatism however, so silver flecks have to be deposited onto the surface of the resist to focus on. This is done by gently dabbing a wooden applicator coated with silver paint onto the chip under a low-power optical microscope (~20×). The silver paint should be semi-dry so that only solid flecks get onto the chip, and the wooden applicator should be as sharp as possible at the tip. Conservatively, the silver flecks are applied to the four corners of the chip to avoid the danger of silver paint blobs forming at the center of the chip where the fine features will be written. The focus at the crucial center part of the chip is then obtained by averaging the foci at the four corners. This practice has fallen out of favor with me as the PMMA resist tends to be higher at the edges of the chip than at the center (due to surface tension at the edges of the chip) and also because it takes longer to do. I choose to dab the silver flecks close to the actual center of the chip and focus close to where I will write the fine features (typically 50 to 100 μm away). Focusing exactly over where the fine features will be written must be avoided because focusing will expose the PMMA underneath. The focus thus obtained I believe to be more accurate than by focusing at the corners of the chip ~1 mm away.

Focus and astigmatism adjustments should be performed at the same current setting used for writing the fine features, and magnifications of 100,000× and higher should be employed. The best part of a silver fleck for focusing on is the boundary between the fleck and the bare PMMA since it will be closest to the actual PMMA surface. Look also for angular features along the edge that provide boundaries at right angles so that blurring in any axis can be detected (Figure 3.4). This is especially important for adjusting astigmatism, in which blurring occurs along one axis but not the



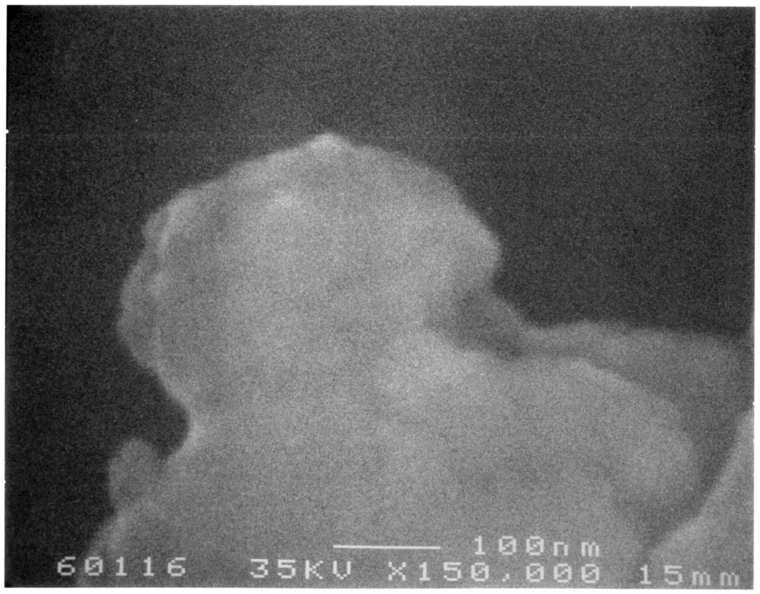


Figure 3.4 Scanning electron micrographs of a silver paint fleck on the surface of a PMMA coated chip. Silver paint flecks are used to aid adjustment of focus and astigmatism of the electron-beam. Angular features on the paint fleck, as shown in the bottom, are ideal spots for adjusting focus and astigmatism.

other. Focus and X and Y astigmatisms should be iterated to obtain the sharpest picture possible.

The current of the electron beam should be measured just prior to writing using either the PCD cup or the AEM and the hole in the sample stage [Chen, 2001]. Pick one method and always only use that method for consistency. When measuring the smallest currents, blank the beam and subtract off any offset that may be present to get the most accurate reading. Note that beam currents are negative on the JEOL while the offset is typically positive.

There are shifts in the origin of the electron-beam when magnification and current settings are changed. They do not appear to change appreciably over time, and the following settings have been found to work for the magnification and current settings listed above:

Magnification	**Offset**
2,000×	0.0, 0.0 μm
200×	+1.3, +2.5 μm
20×	-1.4, -0.4 μm

Smaller, random shifts are superimposed on the above systematic shifts, and connecting pads should be written to ensure proper connection between electrical leads written at different magnifications. Circles of 1.5 μm radius are used for connecting leads between the 2,000× and 200× layers, and circles of 15 μm radius for leads between the 200× and 20× layers.

After the device has been written, it is time to develop the PMMA in preparation for further processing. As suggested by Duncan [2000], a cleaning step before actual development is used to remove carbon paint particles that hold the chip to the SEM sample stage. In this step, the chip is briefly ultrasounded in isopropyl alcohol (IPA), spritzed with fresh IPA, and then blown dry. The carbon particles released during this cleaning step are unable to clog the PMMA resist pattern because the resist has not yet been developed. The developer used is a cocktail of IPA, methyl-isobutyl ketone (MIBK) and methyl-ethyl ketone (MEK) in the following proportions:

375 ml IPA
125 ml MIBK
6.5 ml MEK

The chip is dipped in the developer for one minute while being moved around in order to bring fresh developer into contact with the resist. Development is quenched by dipping the chip in clean IPA, and then the chip is spritzed with fresh IPA and blown dry. The patterned PMMA may then be examined under an optical microscope to check for gross problems like incorrect current or magnification settings, inter-layer connectivity, and particulate contamination. The finest features are not readily visible even at the highest optical magnification (1,000×), but larger features of about 200 nm should be detectable with a little concentration and careful focusing. The actual quantum dot structure may appear to be a single blob in the microscope even when it has turned out correctly, and it is a matter of experience and faith as to whether to proceed on to the next steps of processing or to clean the chip and re-spin the PMMA.

3.1.4 Metallization

Once the resist has been patterned, metal can be evaporated onto the chip in a thermal evaporator. Metallic structures on the chip are typically one of two classes: ohmic contacts or gates. Ohmic contacts are metal structures that, when annealed, diffuse into the wafer to make electrical contact with the 2DEG under the surface of the wafer. Gates are metal structures that define devices in the 2DEG by depleting electrons in the 2DEG. Before metal is evaporated onto the chip, the surface of the chip must be prepared to enhance adhesion of metal to the chip. This procedure involves dipping the chip for a few seconds (3 to 5 seconds) in an ammonium hydroxide solution consisting of 1:5 by volume of 34% ammonium hydroxide and water. This removes the oxide layer on the surface of the chip to expose GaAs for metal to adhere to. Failure to perform this step is almost a guarantee of future frustration as bonding pads will rip off when wirebonds are attempted. The chips should be placed in the evaporator and pumped down as soon as possible to prevent reoxidation of the surface.

For detailed instructions on operating the thermal evaporators, see Chen [2001]. An abbreviated account is given here. The sample is first mounted with carbon paint on the evaporation stage that provides thermal contact for cooling the sample. If the vacuum space in the evaporator is being pumped on, stop the pumping and vent it. The bell jar to the vacuum space should then be raised allowing access to the evaporator. Gloves should be worn while working inside the evaporator to prevent finger grease from contaminating the chamber. The sample stage is mounted upside down so the chips face the sources. Check that the shutter works properly and clears the sample. Mount the evaporation boats to the desired electrical leads and fill or refill the source materials. Lower the bell jar and rough pump the vacuum space to 100–200 mtorr, and then open the gate valve to the high vacuum pump. The roughing pump should not be used below 100–200 mtorr to prevent pump oil from backstreaming into the vacuum chamber and contaminating it. The evaporation boats should be heated briefly after switching to the high vacuum pump. This serves to test the electrical connection to the boats, drive off adsorbed water molecules from the boats, and melt the source material so that it adheres to the boat and will not fall out (this last concern was mostly one in the old Tinkham evaporator which had severe vibration problems but is now decommissioned). Once the pressure gets down to the low 10^{-7} torr, the pressure is low enough to start evaporation. Sample cooling should be turned on if it is not already on. Be mindful to enter the correct settings into the film thickness monitor each time a new material is evaporated. The current through the boats should be ramped up slowly as thermal stresses experienced by the boats can cause them to break. After allowing some of the material to evaporate off to clean the material (say 50 Å), the shutter can be opened to start deposition onto the sample. When the desired thickness is reached, the shutter should be closed and the current ramped down slowly. Once all the requisite materials have been deposited, a few (~5) minutes should be allowed for the boats and electrical leads to cool down before the high vacuum pump is shut off and the chamber vented.

The recipe for ohmic contacts is as follows:

50 Å Ni
200 Å Au
250 Å Ge
100 Å Au
50 Å Ni
400 Å Au

where the Ni layer is the first and the Au layer is the last to be deposited. The first nickel layer acts as a wetting layer and provides adhesion to the GaAs surface of the chip. The gold-germanium-gold sandwich forms a eutectic mixture with a lower meting point than the germanium or gold alone [Crouch, 1996], and provides for the germanium to diffuse into the wafer to make contact with the 2DEG in the later annealing process. The final gold layer provides adhesion for wirebonds to be made, and the nickel layer just before it forms a barrier that prevents the gold from diffusing down. Because nickel is a magnetic metal, deposition of the ohmic contacts is performed in a thermal evaporator reserved for evaporating magnetic materials. This precaution is made to keep magnetic impurities away from "clean" evaporators for devices that may be sensitive to these impurities. A special ohmic contact evaporator stage with the words "ohmics" scribed into it is also used for this purpose.

The recipe for gate deposition is as follows:

50 Å Cr
200 to 400 Å Au

The chrome layer wets the GaAs surface and provides adhesion for the gold layer. The gold layer provides adhesion for making wirebonds to the gate, and a thicker layer of gold makes it easier to wirebond to. However, a thicker layer of gold presents greater difficulty when lifting-off the excess metal, so I usually evaporate 200 Å of gold when making fine structures with small gaps (<100 nm) while going for 400 Å for less demanding structures.

The final step of metallization is to remove the excess metal by dissolving the PMMA resist. This is generally a simple process due to the undercut in a bi- or tri-layer PMMA resist. The chips are left to soak for several hours or overnight in a disposable plastic beaker of acetone. Short bursts of ultrasound may be applied (<1 s each) after an hour or two to remove large pieces of excess metal to facilitate solvent access to the fine features of the device. When it looks like all the excess metal has lifted-off, the chip should be transferred to a shallow petri dish of acetone for examination under a low-power optical microscope. During the transfer, it is critical that the chip remain wetted with acetone, otherwise the excess metal will bond to the surface of the chip and be impossible to remove. Under the optical microscope, check that all visible excess metal has indeed been removed. If not, soak the chip for longer or expose it to a couple more short bursts of ultrasound, and check again. Once all excess metal has been removed, the chip is spritzed with methanol and then blown dry with nitrogen or air.

3.1.5 Annealing Ohmic Contacts

Ohmic contacts require an annealing step to make electrical contact to the 2DEG within the GaAs/AlGaAs heterostructure. This is performed in a strip heater built by Alex Rimberg [Rimberg, 1992]. One chip at a time is placed in the middle of the strip heater, and forming gas is flowed through the chamber at a pressure of 3 psi. The forming gas, which consists of 20% hydrogen and 80% helium, provides a reducing atmosphere so that no oxidation occurs in the chamber. The heating element is a strip of nichrome alloy heated by ac currents supplied via a variable transformer, and the temperature is read with a thermocouple soldered to the nichrome strip. The sample is heated first to 110ºC for one minute to drive off moisture, and then primed at 260ºC for 10 seconds. The temperature is then turned up to 410ºC for 20 seconds and then the heat is turned off. After cooling down to about 40ºC, the sample is rotated 180º and the annealing cycle is repeated.

The repeated annealing is insurance against poor strip-to-sample contact and skipping it is not recommended. Also not recommended is putting more than one sample on the strip heater at a time. The larger thermal mass lowers the actual temperature of the samples and bad ohmic contacts have been made this way. When done correctly, the ohmic contacts should look somewhat cratered, like melted cheese on a pizza (Figure 3.5). This cratering need not be over the whole contact but should at least cover part of it. When in doubt, annealing again will not hurt the sample unless you have a backgate that you do not want to make contact with. A room temperature check can be made by measuring the resistance between ohmic contacts with micromanipulator arms belonging to the Tinkham group. Resistance between ohmic contacts should show a resistance of ~3 kΩ for properly annealed contacts.

3.1.6 Ion-Milling

In addition to metal gates, structures in the 2DEG can also be defined by milling the heterostructure with the ion-miller located in the upstairs cleanroom, although such structures are defined permanently and cannot be adjusted. In this process, argon ions are accelerated toward the sample to physically blast or mill away the exposed regions of the sample. Ion-milling is a non-selective process and will also mill away the PMMA resist, so the PMMA has to be thicker than the depth to be milled. The GaAs/AlGaAs heterostructure is milled at a rate of about 10 Å/s and the 2DEG below is depleted when all the conduction electrons are trapped at the surface of the wafer, which has been brought much closer to the donor atoms.

The sample to be milled is mounted on an ion-milling stage with carbon paint, and the stage mounted on one face of the rotator in the ion-miller. The milling action is turned on by rotating the sample into the path of the argon ions and turned off by turning off the argon beam. Cooling water is turned on after the sample is mounted to enable the large mass of the rotator to be cooled down. The sample chamber is then pumped down, first with the roughing pump and then switching over to the cryopump at about

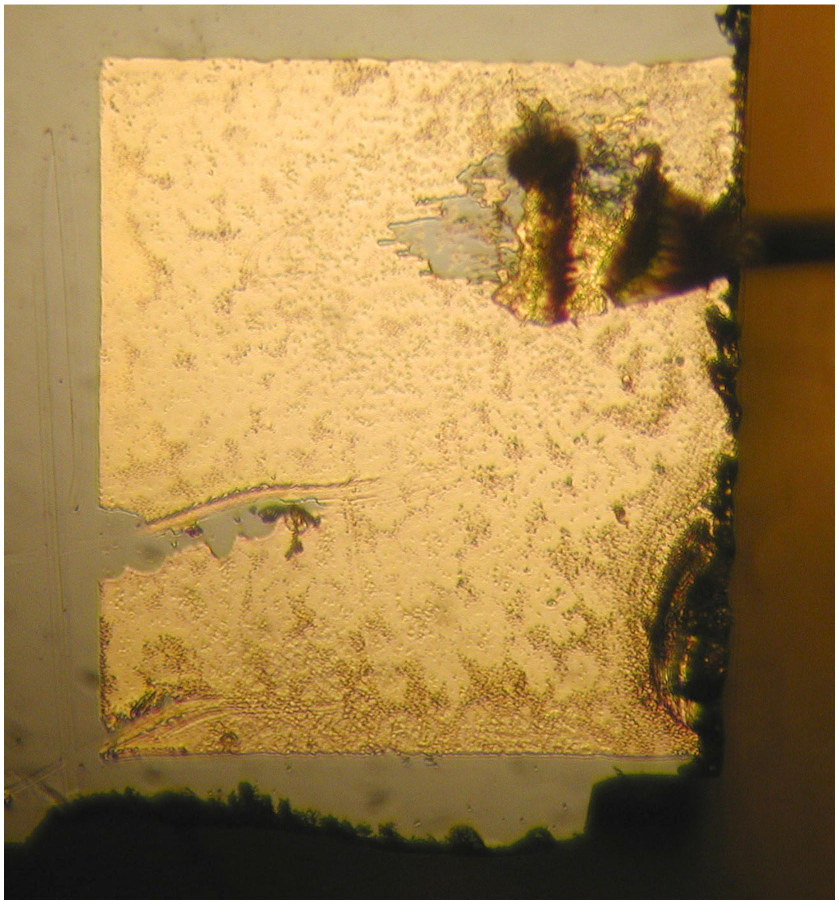


Figure 3.5 Optical microscope picture of a properly annealed ohmic contact. Note the rough, cratered texture of the contact (gold square). The feature in the upper right corner of the contact is a wirebond.

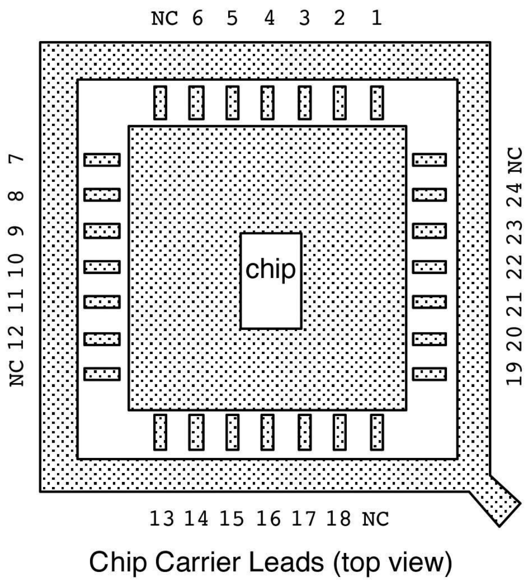
100–200 mtorr. When the pressure is reasonably low (~10^{-6} torr), turn on the mass flow controller for the argon gas to bring the sample chamber up to 2×10^{-4} torr. This is a dynamic equilibrium pressure, with the mass flow controller adding argon gas while the cryopump pumps it out. Make sure the sample is facing away from the argon gun before turning on the argon gun to the following settings:

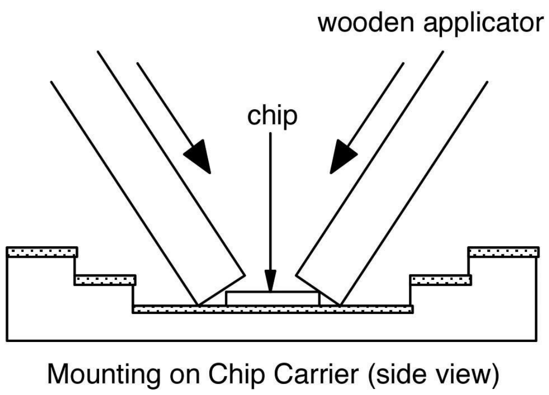
Neutralizer Filament	Discharge Voltage	Cathode Filament	Beam Voltage	Accelerator Voltage
7.25 A	40 V	~5.5 A	500 V	100 V

A pale purple glow can be seen though the glass when the argon gun is on. Recheck the chamber pressure, discharge voltage, and adjust the cathode filament until the beam current is 23 mA. When you are ready to start milling, rotate the sample to face the argon beam and start timing. When the desired time is up, turn off the argon beam and wait for the gun to cool down. Cool down can be speeded up by turning off the cryopump and filling the chamber with argon gas. After a few minutes, the chamber can be vented and opened up, and the sample removed. For more detailed instructions, refer to Chen [2001].

3.1.7 Wirebonding

Wirebonding is the process of making electrical contact from the chip carrier to the sample. The sample is first mounted on a 28-pin old-style JEDEC-standard PLCC chip carrier with a thin layer of GE varnish (Figure 3.6). Using a wooden applicator, a dab of GE varnish is spread into a thin film over the bottom of the chip carrier. The sample is placed in the center of the chip carrier and gently pressed down at the edges with two wooden applicators. Make sure the sample is facing up, of course. The GE varnish is left to dry overnight, preferably in an oven at about 80ºC. The oven is placed in a fume hood and the door left slightly ajar to allow the solvents to evaporate away. The thin layer of GE varnish provides good thermal contact between the sample and the bottom of the chip carrier, allowing the lattice of the sample to be cooled down. The 28-





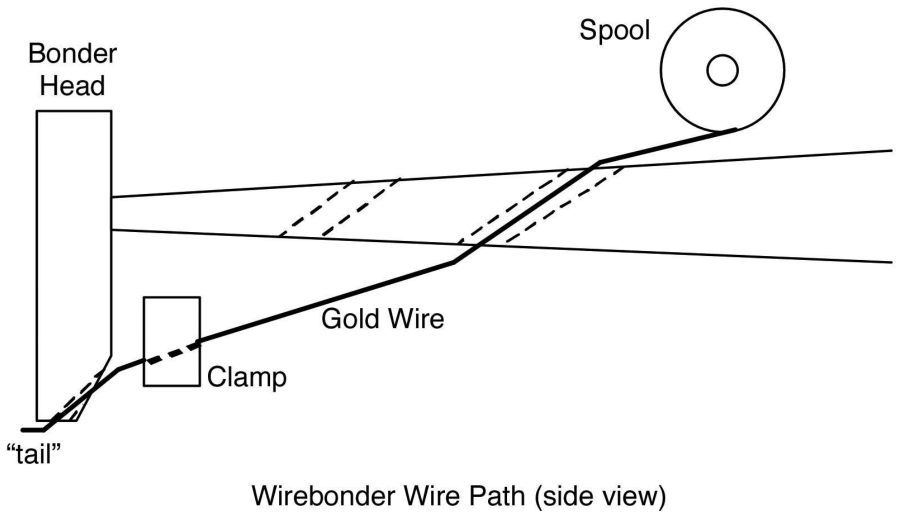


Figure 3.6 Schematic showing tools used for wirebonding. The 28-pin PLCC chip carrier is shown with the numbering scheme used in the Westervelt lab; "NC" indicates that the lead is not connected. The diagram with the wooden applicators shows how to mount the chip to the chip carrier with GE varnish as the glue. The bottom diagram shows the wire path of the gold wire in the Kulicke and Soffa wirebonder.

pin PLCC chip carrier is used to match the PLCC sockets employed in the He4 dunker stick, He3 system (“Shubi dewar”), and the dilution fridge tailpieces. Unfortunately, this version of the chip carrier and socket is obsolete, having been superseded by a newer JEDEC standard that is about 50 mils wider, which is itself considered a legacy product as of this writing. Consequently, chip carriers are hard to come by and are generally recycled. But even worse, the sockets are impossible to come by, and switching over to the newer standard will not work as they are too wide to fit the extremely tight clearances in the He3 and He4 sample probes. Let the reader be very cautious and careful not to break the chip carrier and especially the chip sockets.

Once the sample is mounted on the chip carrier, it is time to wirebond the sample and complete the final step of sample fabrication. The wirebonder we use is from Kulicke & Soffa, and it works by rubbing gold wire into the bonding pad with ultrasonic energy. The gold wire we use is 99.99% pure gold, has a diameter of 1 mil with 1–3% elongation. First, the gold wire has to be threaded properly through the machine to emerge at the bonder head (Figure 3.6). Threading fine gold wire through such a complicated wire path can cause anyone to go insane, but with patience, a little concentration, and a lot of practice, it almost becomes second nature. A pair of high quality needle-nosed tweezers helps too (*e.g*. Techni-Tool type 3C).

Once the wire is threaded, wirebonding can begin. Feeding of the wire can occur only in one direction – directly away from the user. Thus, orientate the chip carrier so that the wirebond to be made goes from the chip carrier pad (closest to you) to the sample bonding pad (further from you) in a straight line away from you. The settings for the two bonds are set separately. Typical values are:

	Search	**Force**	**Time**	**Power**
1st bond	~5.0	3	7	1.8
2nd bond	~5.0	3	7	2.0

The search and loop height settings are measured in mm above an arbitrary reference plane. The first and second search heights should be similar, while the loop height should be 0.5–1.0 mm higher, depending on the span of the wirebond. Make the first bond, move to the second bonding pad and then make the second bond. If all is well, the wire should be cut after the second bond and more wire fed to the bonder head to make a small "tail" in preparation for the next wirebond.

Unfortunately, all is usually not well. If the first bond did not connect, reset the bonder to skip the second bond and try again. If the second bond did not take or the wire was not cut, remove the first bond and try again. If the wire did not feed properly for the next wirebond, it may be possible to release the wire clamp, raise it up, reclamp the wire, and then feed the wire manually by lowering the clamp. If the wire escaped from the bonder head, it must be rethreaded.

Note that not all of the chip carrier's bonding pads are used. In all the sample probes used in the Westervelt lab, only 24 out of the 28 leads to the chip carrier are wired, or 6 out of every 7 leads on each side of the chip carrier (Figure 3.6). Looking down at the top at the chip carrier, the last lead on every side going counter-clockwise is unused. Another point to note is that the metal plane at bottom of the chip carrier is not electrically connected to any lead of the chip carrier, so I normally wirebond it to one of the 24 used leads which I then ground to provide a ground plane for the sample as well as a thermal path for the sample to be cooled. As a precaution against wirebond failure due to thermal cycling and against bad leads in the sample probe, critical gates or ohmics are wirebonded multiple times or wirebonded to multiple pads, or both.

3.1.8 Sample Safekeeping

Once a sample is completed, the last thing one wants to happen is for the sample to be destroyed by static electricity. A safe way to store completed samples is in a 1" Fluoroware (now Entegris) Single Wafer Shipper lined with carbon tape (Figure 3.7). The carbon tape shorts out the leads of the chip carrier, thus preventing voltage buildup

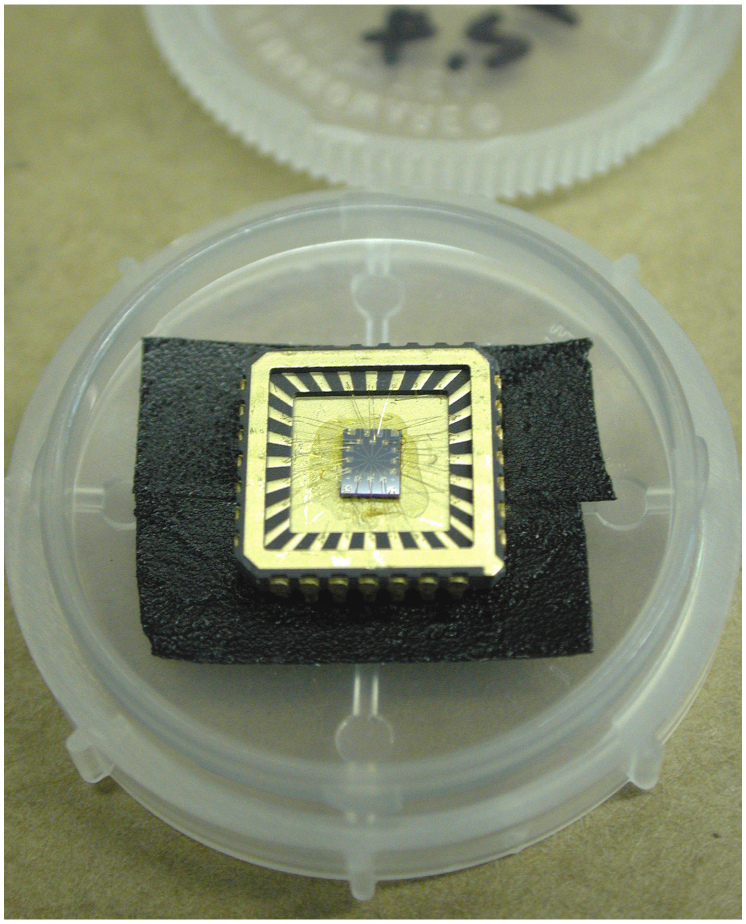


Figure 3.7 Photograph of a completed sample being stored in a 1” Single Wafer Shipper container. Carbon tape lines the bottom of the container to short the leads of the chip carrier together to protect against static electricity. With the cover on, the sample is also protected against dust and other contaminants.

between the leads, and the sealed container protects the sample from dust and other contaminants.

3.2 Measurement Electronics

3.2.1 Overview

A typical measurement setup used in quantum dot experiments is shown in Figure 3.8. A lock-in amplifier and current preamplifier are used to measure the conductance of a quantum dot as various parameters of the dot are varied. These dot parameters are controlled by dc voltages supplied by the Ramper box (Section 3.2.2) or the BiasDac box (Section 3.2.3). In the case of the BiasDac box, the voltages are controlled by a Power Macintosh computer outside the shielded room via fiber-optic cables. A differential summer box (Section 3.2.5) is sometimes used to add two different voltages together for certain measurements (*e.g.* finite bias Coulomb blockade). The output of the lock-in amplifier, which constitutes the data from the experiment, is digitized by a National Instruments (NI) data-acquisition (DAQ) card (Section 3.2.4) and read into the same Power Macintosh computer.

Conductance measurements of quantum dots are measurements of moderately high resistance devices ($R \sim h/e^2 = 25.9$ kΩ) with very small voltages ($eV \sim kT \approx 10$ μeV). Consequently, currents of a few nA and smaller are expected and low noise electronics and techniques are important for obtaining good signal-to-noise ratios. The lock-in technique achieves high signal-to-noise ratios by making measurements over a narrow frequency band about a fixed non-zero frequency. By narrowing the frequency band of the measurement, noise at other frequencies is rejected. By measuring conductance at a non-zero frequency, $1/f$ noise is circumvented, resulting is excellent signal-to-noise. A lock-in frequency of a few hundred hertz is used so the fastest resolvable conductance change is a few tens of hertz. A few hundred hertz is the highest usable lock-in frequency before RC effects from the resistance of the dot ($R \sim 1$ MΩ) and the capacitance in the leads ($C \sim 100$ pF) start to change the phase between the current and the excitation voltage. For a more detailed description of lock-in amplifiers, see Chapter 15 of *The Art of Electronic*s [Horowitz and Hill, 1989].

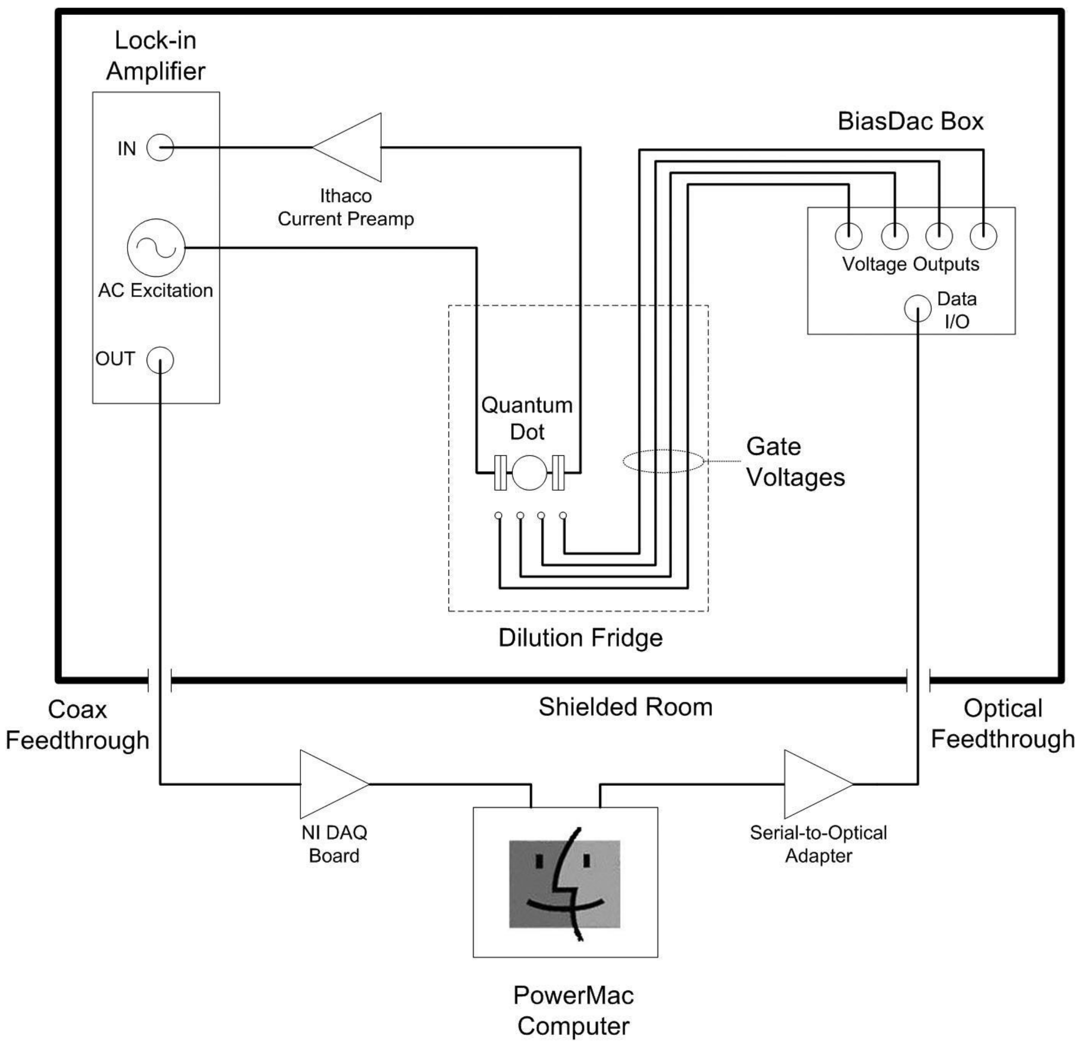


Figure 3.8 Schematic showing the experimental setup of a typical quantum dot measurement. A lock-in amplifier and current preamplifier measure the conductance of a quantum dot as a function of the gate voltages supplied by the BiasDac box. The BiasDac box is controlled by a computer outside the shielded room, which also records the output of the lock-in amplifier.

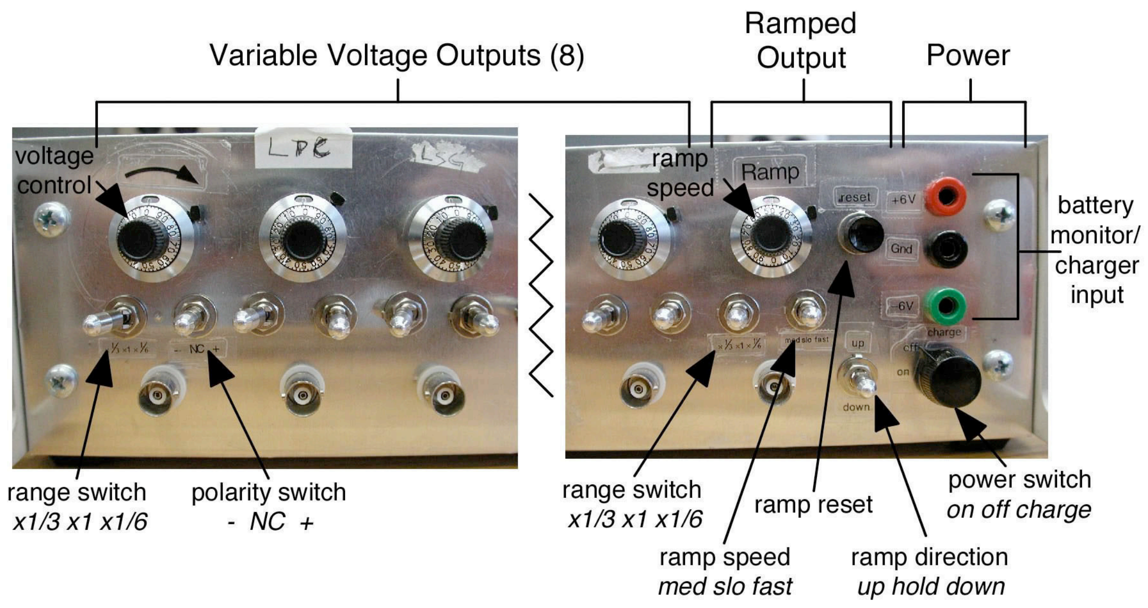
In spite of the excellent signal-to-noise capabilities of the lock-in technique, everything is done to reduce noise that could obscure the desired signal. These include filtering power and signal lines entering the experimental area and avoiding ground loops. Power entering the shielded room is heavily filtered from 100 kHz to 3 GHz. Additional power filtering is performed on sensitive equipment like the lock-in amplifiers as well as potentially noisy devices like the BiasDac box. Signal lines going in or out of the shielded room are filtered with electromagnetic interference (EMI) filters as well. Common mode filtering is performed, *i.e.* both the signal and the signal's ground lines are filtered with respect to the shielded room.

Ground loops occur when there is more than one path for current to travel between *A* and *B*, thus forming a loop that can pick up noise. This usually happens because we are in the habit of thinking that one ground is the same as another ground. This is not so. Grounds are connected via wires just like signal wires, and care should be placed on having one and only one (power) ground defined for an electrical circuit. All other equipment should then be connected to this one power ground via a "ground mecca," which is usually the electrical breakout box (the switcher box) on the fridge that breaks out the 24 electrical leads to the sample into 24 BNC jacks. Defining one and only one power ground can be tricky when dealing with electronics powered from the wall as signal ground could be connected internally to power ground. One way to overcome this is to bypass power ground with a special adapter plug. Be sure that the equipment is grounded from another source when you do this. Another way to break ground loops, especially appropriate for custom-built boxes, is to connect signal ground to power ground through a 100 to 1000 Ω resistor. This resistance is large compared to the mΩ resistance of wire, and noise from ground loops will be proportionately reduced while preventing the signal ground from floating away from power ground. Also be aware that sometimes chassis ground is also signal ground, and in rack-mounting a piece of equipment, you may inadvertently be connecting signal ground to power ground via the chassis of the equipment. Of the common wall-powered equipment used in the lab, only the lock-in amplifier (PAR 124A) has signal ground connected to chassis ground (via the lock-in excitation output; its inputs are not directly connected to chassis ground). The Ithaco (now DL Instruments) model 1211 current preamplifier, PAR 113 voltage preamplifier, and several custom electronics boxes are battery-powered boxes recharged from the wall and do not present any ground loop problems.

3.2.2 Ramper Box

The gates defining a quantum dot are energized by fixed dc voltages from either the all-analog Ramper box, originally designed by Jordan Katine [Katine, 1997], or the all-digital BiasDac box designed by the Harvard electronics guru Jim MacArthur and modified by myself and Parisa Fallahi. The Ramper box is a voltage-output box in which each output is controlled by a 10-turn potentiometer. A ramped voltage output (hence the name Ramper box) is available for sweeping voltages, but the need for manually starting and stopping ramps gets to be distracting. The Ramper box is entirely analog and thus not a source of high frequency digital noise. However, the lack of computer control represents a drawback in terms of flexibility and convenience, so the Ramper box is typically relegated to backup or supplementary duty.

Figure 3.9 shows the front panel and inside construction of the Ramper box that I built. The front panel consists of (from left to right) eight variable voltage outputs, one ramped voltage output, and one set of power controls. Each variable voltage output consists of a 10-turn potentiometer that sets the output voltage, a switch that sets the output range, and a switch that sets the output polarity. Each voltage output is unipolar with a maximum output of about ±6 V, limited by the 6 V lead-acid batteries that power the Ramper box. The ramped output consists of a 10-turn potentiometer that sets the fine speed of the ramp, a switch that sets the output range of the ramp, a switch that sets the coarse speed of the ramp, a switch that controls the direction of the ramp, and a reset button that sets the output voltage to zero. The ramped output is bipolar and the


Variable Voltage Outputs (8)
Ramped Output
Power
voltage control
ramp speed
Ramp
battery monitor/ charger input
range switch *x1/3 x1 x1/6*
polarity switch *- NC +*
range switch *x1/3 x1 x1/6*
ramp reset
power switch *on off charge*
ramp speed *med slo fast*
ramp direction *up hold down*

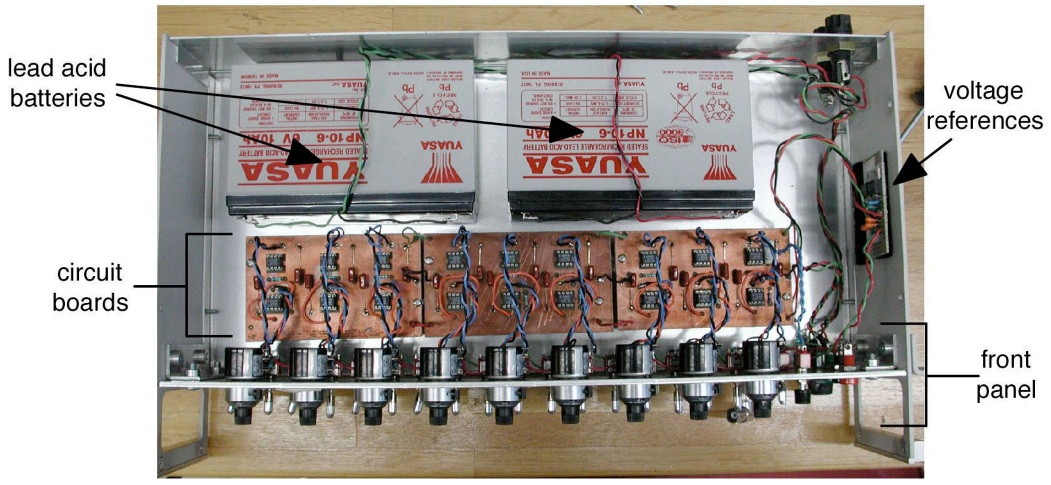


Figure 3.9 Front panel and internal construction of the Ramper box. The front panel shows the controls for the eight variable voltage outputs, one ramped voltage output, and power controls. The internal photograph shows the lead-acid batteries, voltage references, and circuit boards that constitute the Ramper box.

voltage limits of the ramp are set by the range switch. Setting the limits of the ramp to user-defined voltages is not currently implemented, so constant human intervention is necessary to stop the ramp before the output exceeds safe limits. The power controls consist of three banana plugs for access to the batteries, and a power switch. The power switch turns the Ramper box on or off, and in “charge” mode, it provides battery access via the banana plugs so that the batteries can be monitored or charged with external chargers. The circuit diagram of the variable and ramped outputs can be found in Appendix A.1.1.

Some improvements I have incorporated into the Ramper box include a more stable voltage reference than a battery (see Appendix A.1.2), built-in batteries (previous boxes had external batteries which could be disconnected accidentally), and custom-made circuit boards for tidiness. The two 6 V, 10 Ah batteries in the Ramper box should be able to power all outputs of the Ramper box for about a month (~1,000 hrs) continuously because the 18 AD797 op-amps draw only 0.5 mA of current each. Note that the outputs of the Ramper box are not electrically isolated from each other but share a common circuit ground different from chassis and power grounds. The circuit ground is referenced to power ground only when the Ramper box is connected to the sample (via the lock-in amplifier’s excitation output). Note also that the AD797 op-amps *do not* like capacitive loads, and the output will oscillate wildly if placed directly on a large capacitive load. In particular, do not connect the output of the Ramper box directly to the electrical feedthrough of the shielded room because the feedthroughs possess ~1 μF capacitance to ground. Placing a resistor (*e.g.* 1 kΩ) between the Ramper box output and a capacitive load should solve the problem.

3.2.3 BiasDac Box

Having computer-controlled voltage outputs is an incredibly useful tool for research in quantum dots. This effectively means computer control of gate voltages and therefore computer control of the dot state. This affords the experimenter great flexibility and control compared to the manually-controlled Ramper box, and is an evolution not to be underestimated. The front and rear panels of the BiasDac box are shown in Figure 3.10. The front panel is very straightforward and consists of eight voltage outputs, an on/off switch, and two LED power indicators. The rear panel consists of a mains power switch, a fuse for digital power, and two pairs of optical connectors for receiving and transmitting data from the computer. The internal construction of the BiasDac box is shown in Figure 3.11 and reveals the two power supplies (digital and analog) and the two BiasDac boards that make up the box. Each BiasDac board has four outputs, and up to 61 boards can be daisy-chained together for a maximum of 244 outputs (each BiasDac board will need a distinct board ID however). Each output can be configured to be unipolar or bipolar, and output up to ±10.0 V (Appendix A.2.2). For our boards, we have configured them to be bipolar with a 10.0 V range (*i.e.* the output ranges from -5.0 V to +5.0 V). Due to the high power demand of the BiasDac (each board consumes ~8 W), battery power is out of the question and the box is powered from the wall.

The original design of the BiasDac is shown in schematic in Figure 3.12 (the complete circuit diagram is found in Appendix A.2.1). Going from input to output, data is sent to the BiasDac board via fiber optic cable. The data is processed by a microprocessor that translates the incoming commands into the actual bits of data that will operate the DAC (the microprocessor also corrects for temperature and systematic errors in each output channel as well). The translated data is sent to the respective DAC via a capacitive coupler (ISO150) for electrical isolation of each output channel. Power to that DAC channel is also isolated through the use of dc-dc converters that transfer currents between circuits without connecting them electrically. The DAC responds to the command it receives and sets the output voltage to the desired value.

There were problems encountered with the original BiasDac design however. While differential mode noise was low (comparable to the Ramper box noise), common mode noise was very large (almost 1 V_{pp} in a 100 MHz bandwidth). This is to say

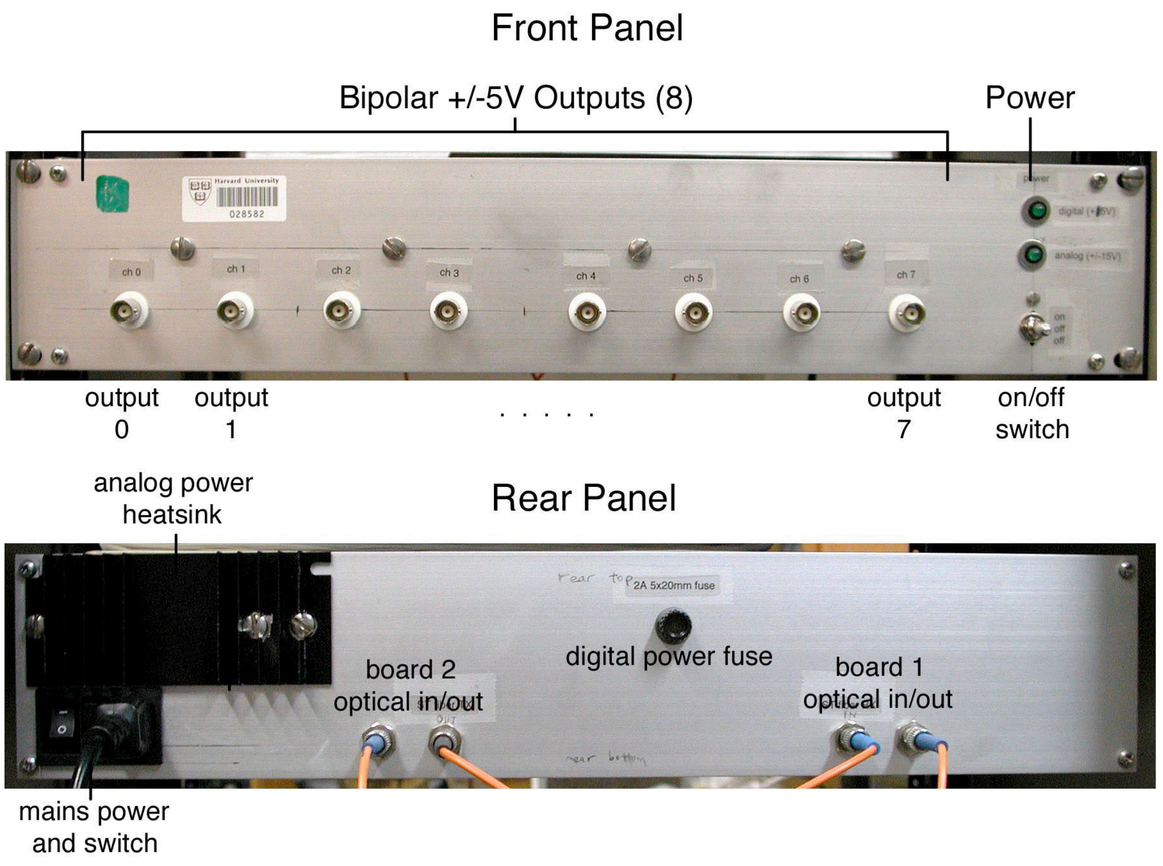


Figure 3.10 Front and rear panel photographs of the BiasDac box. The front panel shows the eight voltage outputs of the BiasDac and the on/off switch. The rear panel shows the main power entry, analog power heatsink, and optical fiber inputs and outputs.

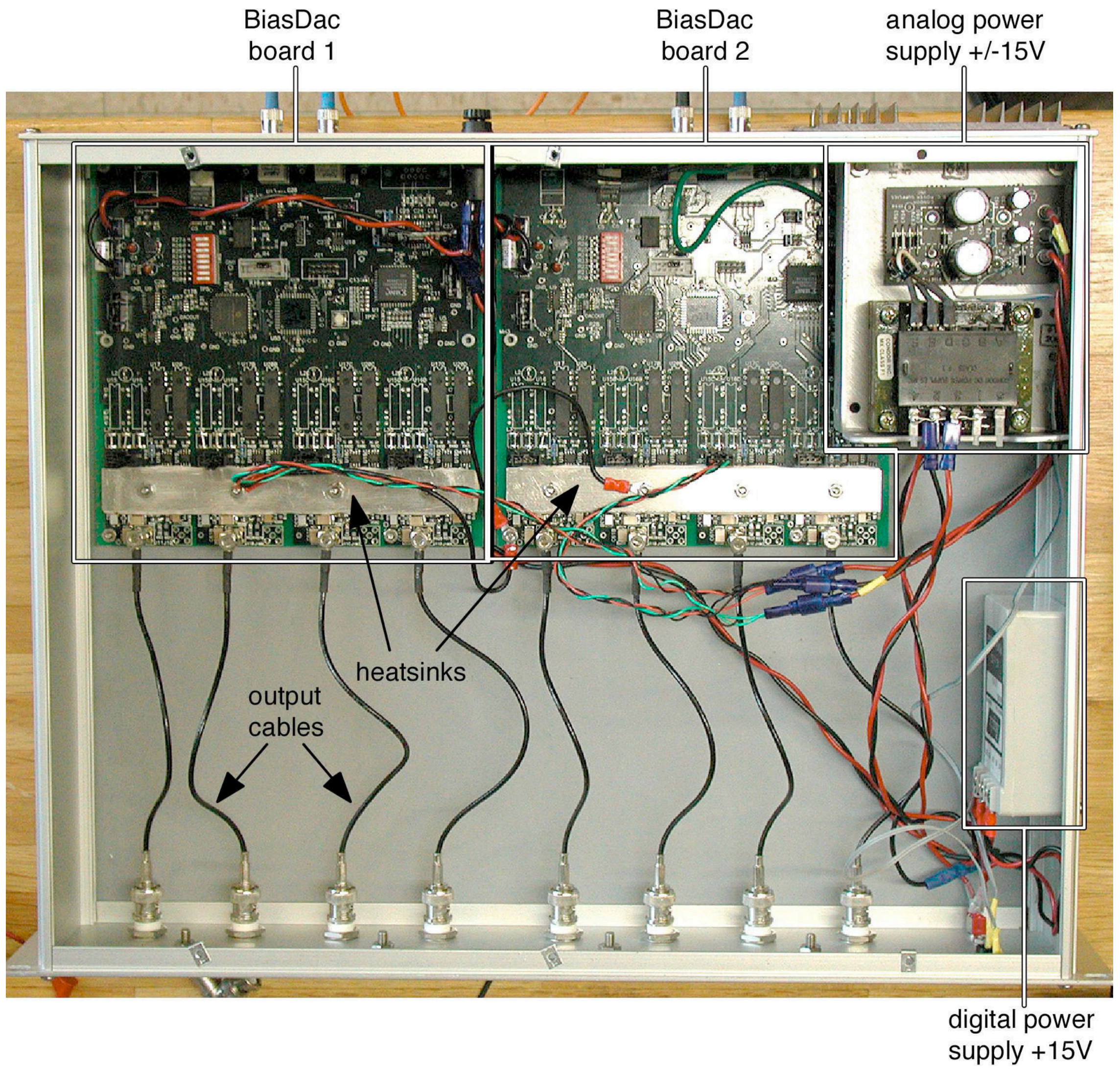


Figure 3.11 Internal construction of the BiasDac box showing the two power supplies, two BiasDac boards, and output cables. The DAC1220 DACs and related op-amps are located under the aluminum heatsink on the BiasDac boards.

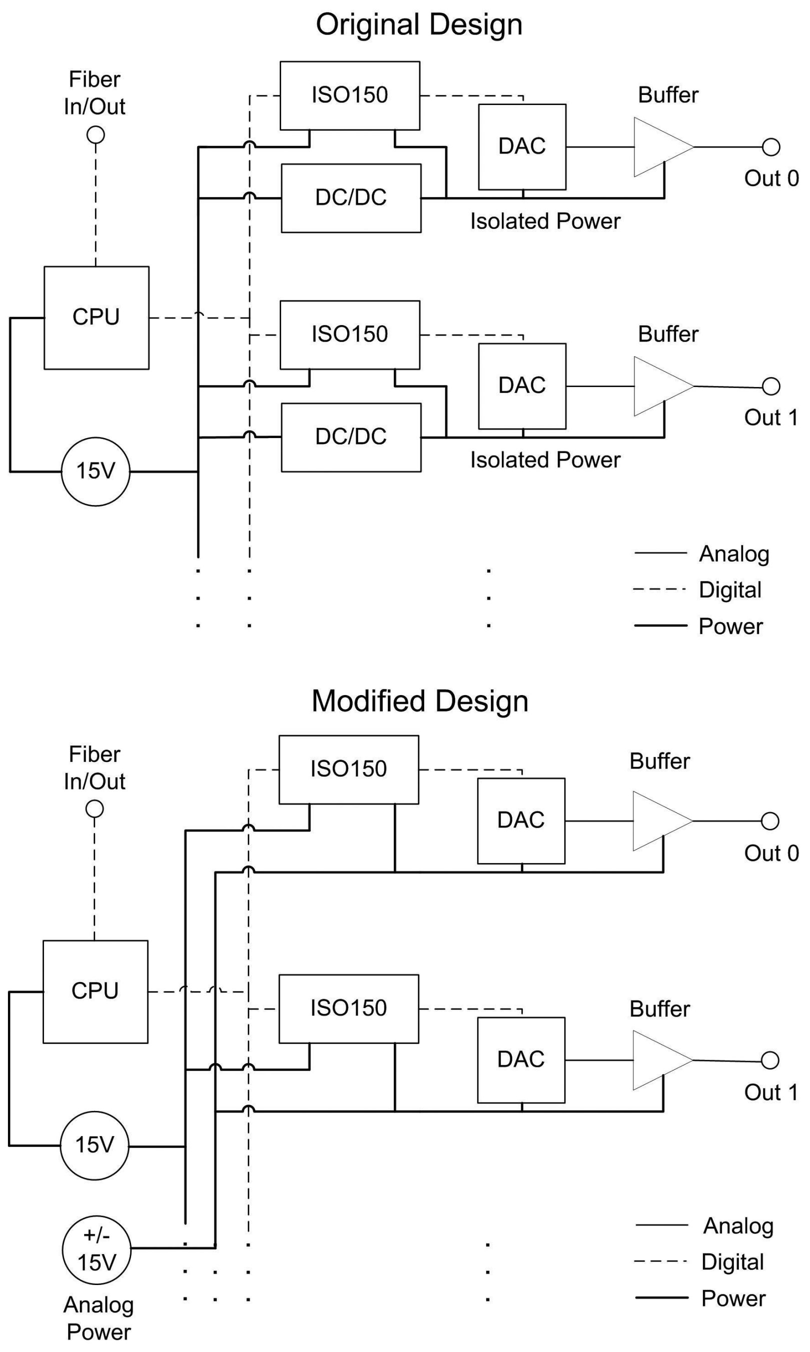
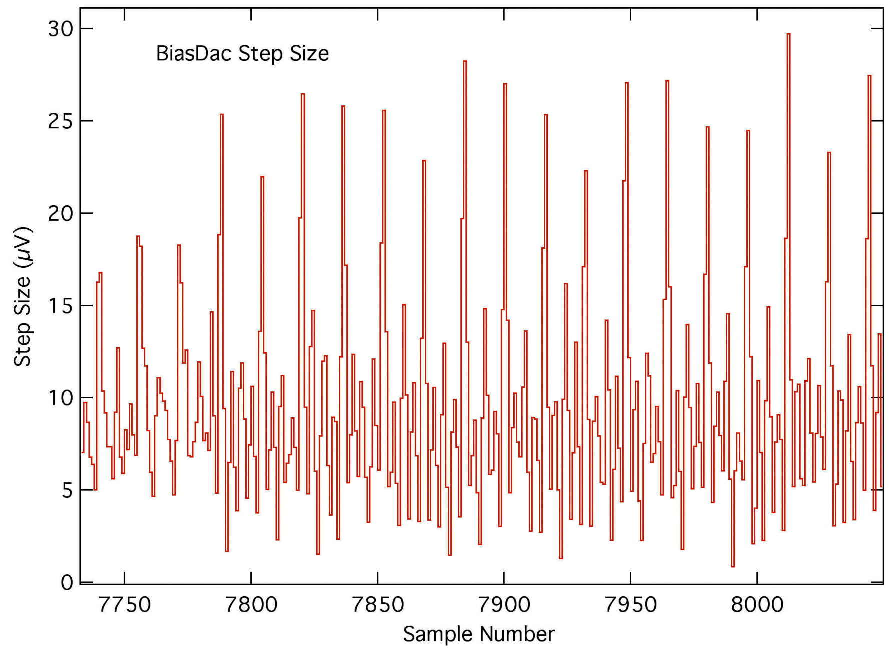


Figure 3.12 Simplified schematic diagrams of the original and modified BiasDac circuits. The dc-dc converters used to provide isolated power to each BiasDac channel were found to generate a lot of noise (Appendix A.2.4), and were removed in favor of a linear power supply without isolation between channels.

that both positive (signal) and negative (ground) outputs of a BiasDac channel were fluctuating compared to earth ground but not to each other. After some investigation, I found that the dc-dc converters and an interconnect cable between the two boards were to blame (Appendix A.2.4). Removing the dc-dc converters and powering the BiasDac channels with a separate power supply (Figure 3.12) reduced common mode noise by an order of magnitude, and replacing the interconnect cable with a fiber optic cable further reduced common mode noise by another order of magnitude. Common mode noise is finally comparable to differential mode noise (several mV_{pp} in a 100 MHz bandwidth), and should no longer be a problem. Simple *RC* filters with a 1 ms time constant are added to the BiasDac outputs before connection to the quantum dot, and should remove any residual noise. A consequence of the removal of the dc-dc converters is that the BiasDac channels now share a common circuit ground and are no longer electrically isolated from each other (though they are still not referenced to power ground except when connected in an experiment).

Another issue encountered with the BiasDac was the fact that the DAC chips used have an internal flaw that reduced the resolution of the DAC from 20 bits to 18.5 bits. According to Jim MacArthur, there are leakage paths between some input bits of the DAC and its output so that some output steps are systematically larger than others. Independent testing confirms that this is the case (Figure 3.13), and there is little we can do about it as it is a problem only Burr-Brown (now Texas Instruments) can fix.

Although there seems to be a litany of problems with the BiasDac, it is still a very useable computer-controlled voltage output box with many advantages over commercial products by National Instruments. Some of the advantages include fiber-optic input/output, electrically isolated outputs, a DAC for every output channel, and high resolution. Fiber optic I/O is very convenient as just two optical feedthroughs are needed to control eight voltages compared with the eight coaxial feedthroughs needed for a card installed in a computer outside the shielded room. The outputs of the BiasDac box are also not power



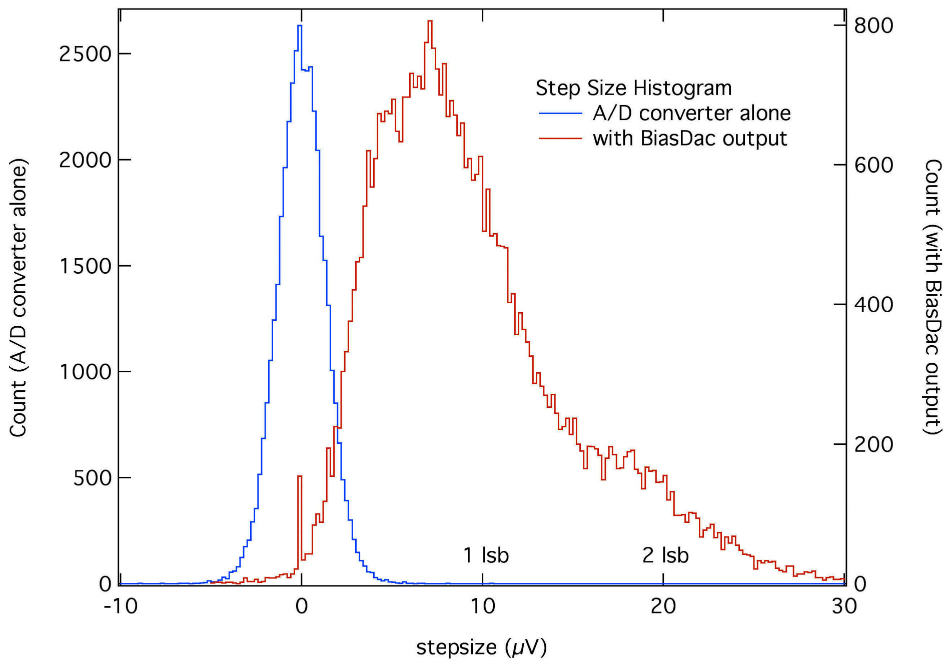


Figure 3.13 Data showing systematic errors in the BiasDac output step size. Step size for a 20 bit DAC spanning a 10 V range should be 10 μV, but due to an internal problem with the DAC chip, some steps are systematically larger than others. In the graph below, the histogram of step sizes with the A/D converter alone is much narrower than the histogram when measuring the BiasDac output, showing that the measured error is real and not an artifact of the A/D converter. The histogram of the BiasDac output goes out to about 3 lsb (least significant bits), so the effective resolution is about 18.5 bits rather than 20 bits.

ground-referenced while those on the National Instruments PCI cards are, so ground loop issues are minimized. And although it may not seem like a significant feature, the BiasDac board has one DAC per output channel. This is in contrast to NI products where one DAC is actually responsible for four or more outputs! In their scheme, the DAC is switched quickly between different outputs while the output is kept “constant” with some kind of circuit. The BiasDac DACs are also capable of 18.5 bits of real resolution and are ~6× better than the 16-bit outputs of NI boards.

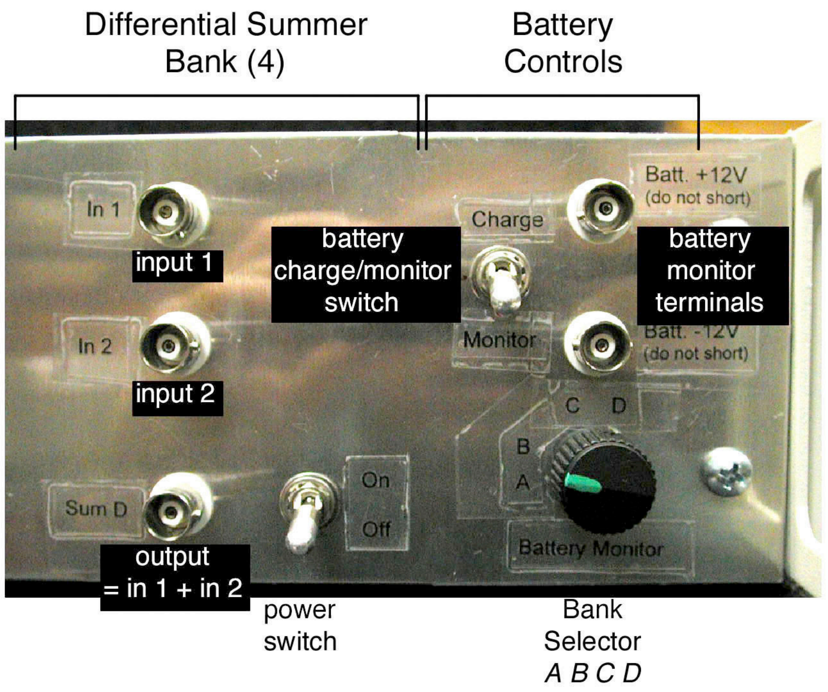
3.2.4 Analog Input

Input voltages are measured using a National Instruments PCI-6032E analog input board plugged into a PowerMacintosh computer (“Big Blue”). The board has eight differential inputs that can acquire data at 100 kSamples/s at 16-bit resolution. Input voltage range is ±10 V. The board is connected to the BNC-2110 terminal block that provides BNC jacks for the eight inputs, among other things. The ground setting for each input on the terminal block is important to setup correctly. The ground switch should be set to Floating Source (FS) when measuring voltages that are floating with respect to power ground. In this configuration, the signal ground is tied to computer ground so that the input does not float beyond the ±10 V range of the analog input board. The ground switch should be flipped to Grounded Source (GS) when measuring a voltage that is already referenced to power ground. In this configuration, the input is *not* connected to computer ground in order to avoid making a ground loop. When measuring the output of the PAR 124A lock-in amplifier, the switch should be set to GS.

3.2.5 Differential Summer Box

One final piece of equipment used in measurement is the differential summer box, which sums two voltages together without connecting their grounds. The box is typically used when performing finite bias Coulomb blockade measurements to add a dc bias voltage to the lock-in ac excitation, or when making electrometer measurements to add a compensating voltage to the electrometer sidegate voltage. The front panel and internal construction of the differential summer box is shown in Figure 3.14. The front panel consists of four banks of summers (only bank D is shown) and a power control section. Each summer bank consists of two inputs, “In 1” and “In 2,” and one output “Sum” = In 1 + In 2. The differential summer circuit is relatively straightforward and consists of three chips per bank, as shown in Figure 3.15. Each input is fed to its own differential amplifier (AD620) which provides separation of input and output grounds. The resistors on the inputs of the AD620 supply input bias currents and prevent the output of the AD620 from going to the voltage rails when there is no input. The outputs from the differential amplifiers are summed in a conventional op-amp summing circuit to produce the output.

The power control section is used to monitor and charge the different banks of batteries in the box. Each bank of summers is powered by its own pair of 12 V lead-acid batteries, and the rotary selector enables a selected pair of batteries to be monitored on the BNC jacks. A switch allows the selected pair of batteries to be charged with a pair of internal battery chargers for convenience. The circuit diagram of the power supplies is also shown in Figure 3.15. Each differential summer bank should last two weeks of continuous use before their batteries need to be recharged.



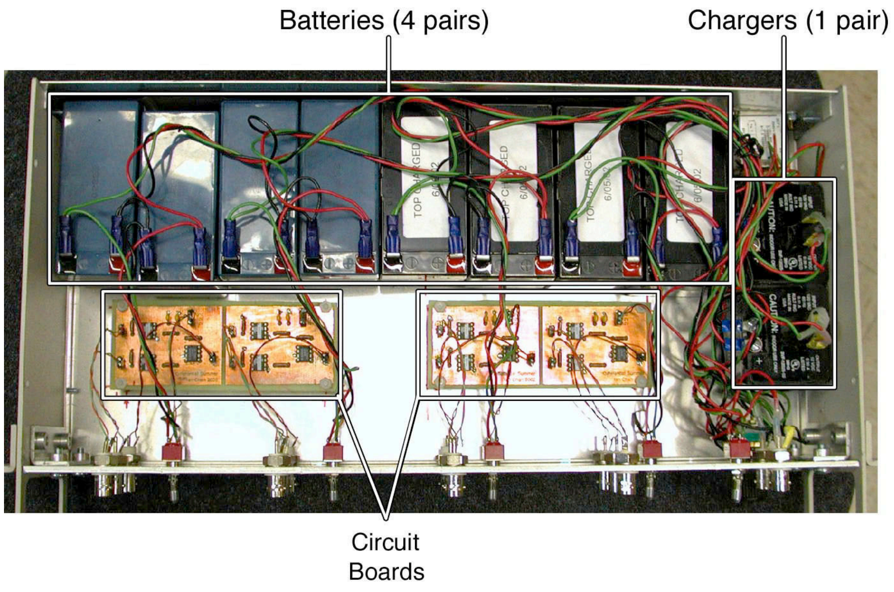
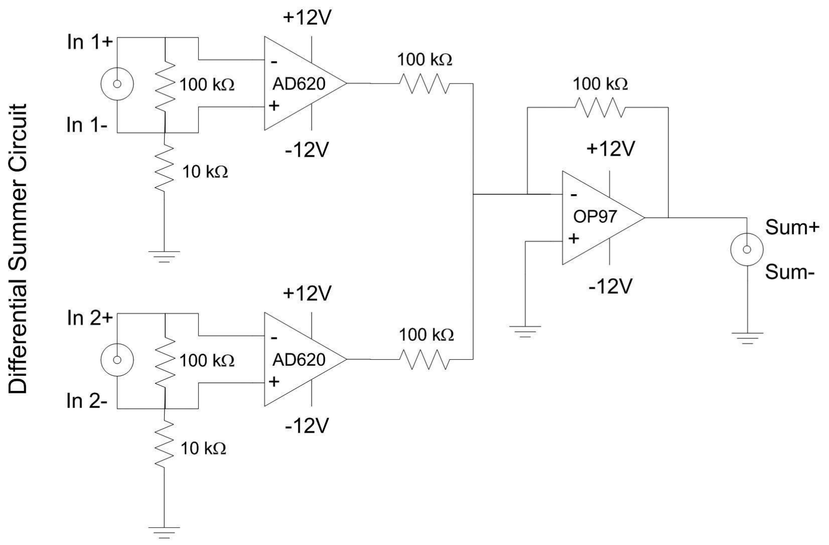


Figure 3.14 Front panel and internal photographs of the differential summer box. The front panel shows the controls for one bank of the differential summer, and the battery controls for the box. The internal picture shows the four pairs of batteries used to power the four banks of summers, the internal chargers, and the circuit boards containing the differential summers.



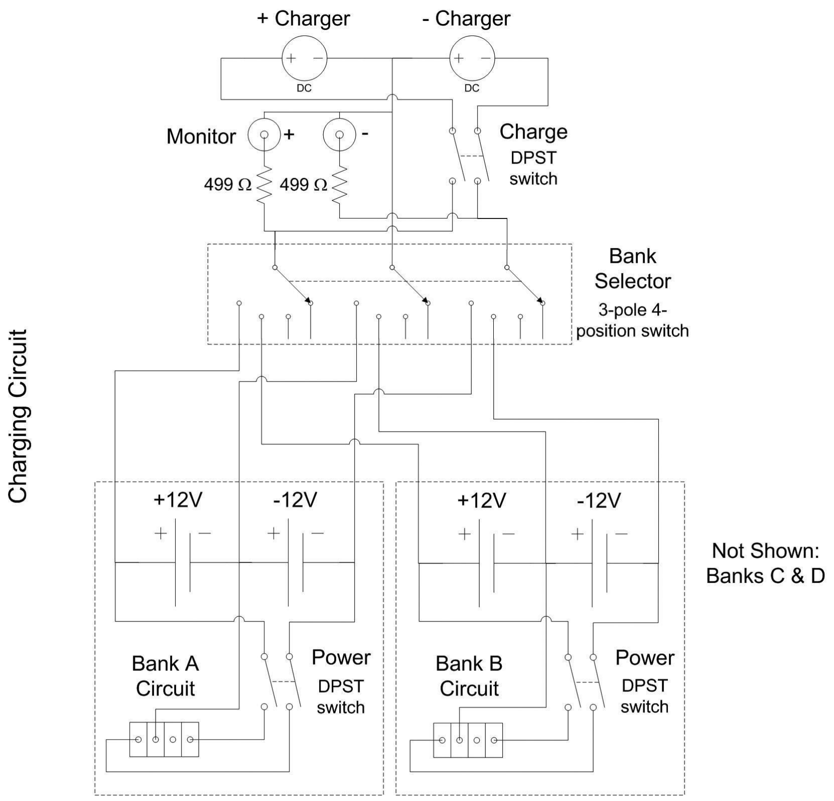


Figure 3.15 Circuit diagrams for the differential summer box. The top circuit is the diagram for one differential summer bank that adds two voltages without connecting their grounds together. The bottom circuit shows the charging and power wiring that allows one pair of BNCs and chargers to monitor or charge any of the four pairs of batteries.

3.3 Custom Software

3.3.1 The Mac OS X Application Programming Interface

Mac OS X, introduced in 2001, is a very reliable, stable, and useful operating system that provides a much needed new "brain" to complement the new "brawn" provided by the introduction of PowerPC processors in 1994. However, the radical reengineering of the Mac OS has necessitated changes in the application programming interface (API) that allows programs interact with the Macintosh (*e.g.* draw on the screen, open files, and print), with important impact for custom-written software used in the lab. The set of functions that is accessible in Mac OS 9 and earlier is known as "Classic." The subset of Classic fucntions that work in Mac OS X is known as "Carbon." The new, object-oriented API introduced with Mac OS X is "Cocoa."

The Classic API is to be avoided, since programs written for it will only run slowly if at all in Mac OS X. Brian LeRoy and myself have converted our custom software (*e.g.* Poisson and AFM) to use the Carbon API, which is compatible with both Mac OS 9 and X, and that is what we recommend new software to be written for. The Carbon API is mostly the same as the Classic API, so it will not be difficult for someone knowing the Classic API to switch over to Carbon. One need only be aware of which Classic functions are Carbon-compliant. A very good and continually updated resource is the Carbon Documentation page in the Developer section of the Apple Support website:

http://developer.apple.com/techpubs/macosx/Carbon/carbon.html/.

It gives an exhaustive list of what functions are supported, and how to use them.

The Cocoa API is supported on Mac OS X only and provides a native, object-oriented API to the Macintosh using the Objective C language. It would be recommended since we are transitioning to Mac OS X but for one problem – the *National Instruments* API for accessing their hardware was obtainable only for Mac OS 9 (the NI API for the Macintosh is not distributed or even sold by National Instruments; Brian LeRoy had to harass and cajole an NI engineer for the in-house software, and recent attempts for the

Mac OS X version have failed). Until such time that a Mac OS X version of the National Instruments API is available, the AFM program can only run in Mac OS 9, so Cocoa is out. However, the use of the Carbon API still allows the AFM program to be Mac OS X-ready while allowing it to continue running in Mac OS 9.

3.3.2 Poisson Solver

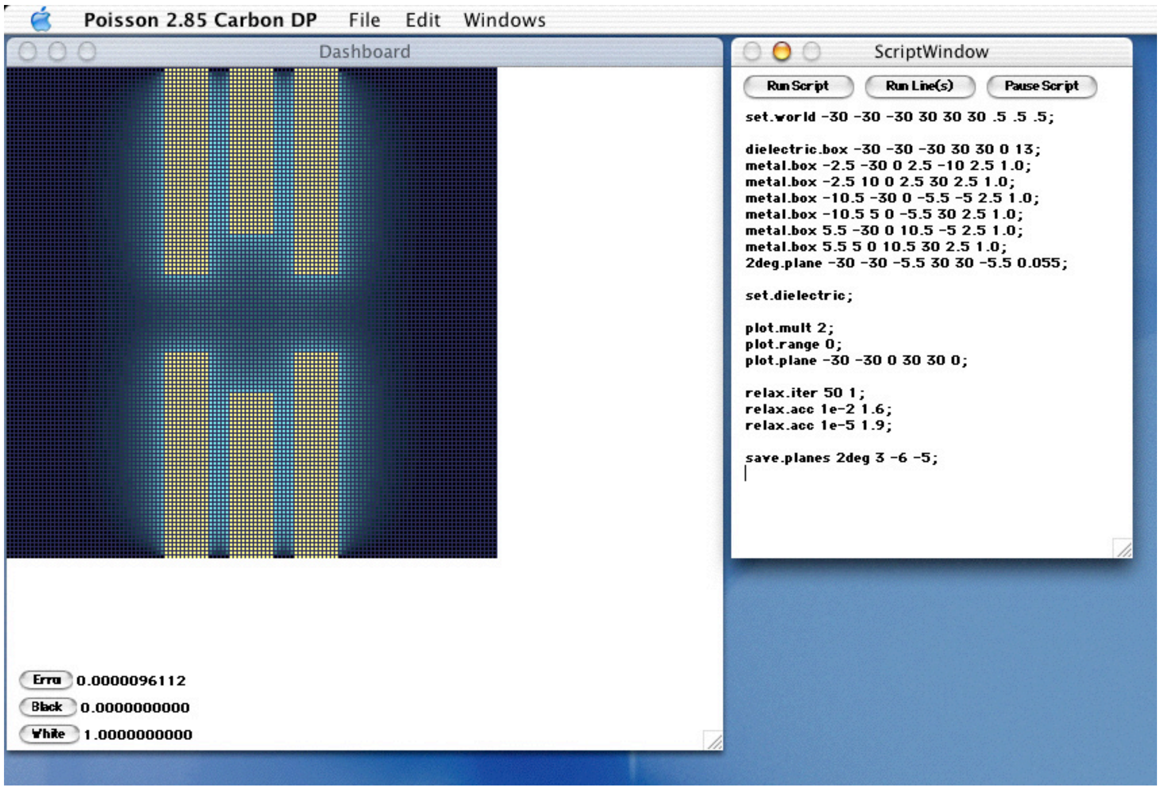
Poisson is a program that solves Poisson's equation arising from electrostatic problems using successive over-relaxation (SOR). It is a useful tool for simulating the electrostatic behavior of quantum dots and it was used to estimate the capacitive-coupling between quantum dots as well as for estimating the size of few-electron quantum dots. An introduction to the computational technique used (SOR) can be found in Chapter 19 of *Numerical Recipes in C* [Press *et al.*, 1992].

The core computational code in Poisson 2.8.5 was written by myself in C++, and mated to the user interface of a previous version (1.0) written by Mark Topinka in C. The primary improvements in my code include the correction of a boundary problem at the interface between dielectrics (see Appendix A.4.3), the use of floating point coordinates instead of integer coordinates, the ability to build structures using pictures (PICTs) drawn in a drawing program, and multiprocessing.

The user interface to Poisson 2.8.5 is shown in Figure 3.16. The left hand window is the Dashboard window that displays the color-coded potential in a slice of the simulation (gray = dielectric, cyan = dielectric boundary, yellow = metal, purple = electric-field constraint, and green = non-depleted 2DEG). The right hand window is the Script window in which the user enters commands to the program to build the simulation world and run the program. An example script is listed here:

```
set.world -30 -30 -30 30 30 30 .5 .5 .5;

dielectric.box -30 -30 -30 30 30 0 13;
metal.box -2.5 -30 0 2.5 -10 2.5 1.0;
metal.box -2.5 10 0 2.5 30 2.5 1.0;
metal.box -10.5 -30 0 -5.5 -5 2.5 1.0;
metal.box -10.5 5 0 -5.5 30 2.5 1.0;
```



```
set.world -30 -30 -30 30 30 30 .5 .5 .5;

dielectric.box -30 -30 -30 30 30 0 13;
metal.box -2.5 -30 0 2.5 -10 2.5 1.0;
metal.box -2.5 10 0 2.5 30 2.5 1.0;
metal.box -10.5 -30 0 -5.5 -5 2.5 1.0;
metal.box -10.5 5 0 -5.5 30 2.5 1.0;
metal.box 5.5 -30 0 10.5 -5 2.5 1.0;
metal.box 5.5 5 0 10.5 30 2.5 1.0;
2deg.plane -30 -30 -5.5 30 30 -5.5 0.055;

set.dielectric;

plot.mult 2;
plot.range 0;
plot.plane -30 -30 0 30 30 0;

relax.iter 50 1;
relax.acc 1e-2 1.6;
relax.acc 1e-5 1.9;

save.planes 2deg 3 -6 -5;
```

```
Error 0.0000096112
Black 0.0000000000
White 1.0000000000
```

Figure 3.16 User interface of the Poisson program. At left is the Dashboard window that displays a slice of the electrostatic problem being simulated. At right is the Script window in which commands for creating the problem and running the program are entered. For the full list of Poisson commands, see Appendix A.4.2.

```
metal.box 5.5 -30 0 10.5 -5 2.5 1.0;
metal.box 5.5 5 0 10.5 30 2.5 1.0;
2deg.plane -30 -30 -5.5 30 30 -5.5 0.055;

set.dielectric;

plot.mult 2;
plot.range 0;
plot.plane -30 -30 0 30 30 0;

relax.iter 50 1;
relax.acc 1e-2 1.6;
relax.acc 1e-5 1.9;

save.planes 2deg 3 -6 -5;
```

This example script first creates a simulation world that spans the spatial (x, y, z) coordinates (-30, -30, -30) to (30, 30, 30) in steps of (0.5, 0.5, 0.5). Then it fills the world from (-30, -30, -30) to (30, 30, 0) with dielectric of dielectric constant 13 (this represents GaAs or AlGaAs). Then it creates metal gates (`metal.box`) of various dimensions set to a voltage of 1.0 "volt." Then it creates a plane of 2DEG at z = -5.5 that depletes at an electric field strength of 0.055 "volts/unit," where "unit" is whatever unit of length you are using. This value of 0.055 "volts/unit" comes from the desire for the 2DEG that is 5.5 "units" below the gates to deplete at a gate voltage of 0.3 "volts" (0.3/5.5 = 0.054545…). Note that the 2DEG depletes at positive "volts!" The next command (`set.dielectric`) tells the program to figure out the appropriate dielectric constants to use at dielectric boundaries (see Appendix A.4.3). The next three commands tell the program what to plot in the Dashboard window [draw a 2×2 pixel for every point to be plotted, use a color scale that covers the range of potentials in the area to be plotted rather than the global range, and plot the coordinates (-30, -30, 0) to (30, 30, 0)]. The next three commands tell the program to relax the potentials, first relaxing 50 times with a SOR factor of 1, then relaxing to an accuracy of 10^{-2} with a SOR factor of 1.6, and finally relaxing to an accuracy of 10^{-5} with a SOR factor of 1.9. The last command saves planes of potentials in the z-axis (1 = x, 2 = y, 3 = z) in the range z = -6 to -5 to disk under the name "2deg." In summary, the script simulates a small quantum dot and saves a few layers of the results to disk. For the complete list of Poisson commands, see Appendix A.4.2. For a description of the C++ classes used in the Poisson solver, please refer to Appendix A.4.1.

3.3.3 Poisson Solver Tips and Tricks

Sometimes, it is necessary to simulate electrostatic problems that include a floating gate, such as the coupling gate of the strongly capacitively-coupled double dots of Chapter 4, in which the constraint on the gate is that the total charge be fixed (at zero usually) rather than the voltage be fixed. The Poisson Solver does not directly allow this, but fortunately, the linearity of the solutions to Poisson's equations provides a way to accomplish this. Linearity of the solutions means that, if A is the solution to a problem with two metal gates in which one gate is held at $V_1 = 1$ V and the other gate held at $V_2 = 0$ V, and if B is the solution to the same problem but with $V_1 = 0$ V and $V_2 = 1$ V, then the solution to the problem with $V_1 = \alpha$ V and $V_2 = \beta$ V is just the linear combination $C = \alpha A + \beta B$. In the same vein, a problem incorporating a floating gate is solved in the following manner: First, find the solution S_0 for the problem in which the floating gate is set to 0 V while the other gates are set to their desired fixed voltages. Also calculate the charge on the floating gate Q_0 for solution S_0 using the `charge.shell` and/or `charge.plane` commands. Next, find solution S_1 to the same problem, but with the floating gate set to 1 V and the other gates set to 0 V. Also calculate the charge on the floating gate Q_1 for solution S_1. The solution to the problem incorporating a floating gate is then the linear combination that fixes the charge on the floating gate to zero, $S_F = S_0 - Q_0/Q_1\, S_1$.

It is also often necessary to construct electrostatic problems in which metal features are more complex than can be made using simple geometric shapes, like the gates of the few-electron quantum dots of Chapter 5. The way to do this is to use the command `metal.readPict` to create a metal extrusion of an arbitrary two-dimensional pattern (PICT) designed in a drawing program. Figure 3.17(b) shows a PICT representing the gate pattern of a few-electron double dot created in Photoshop from an actual SEM

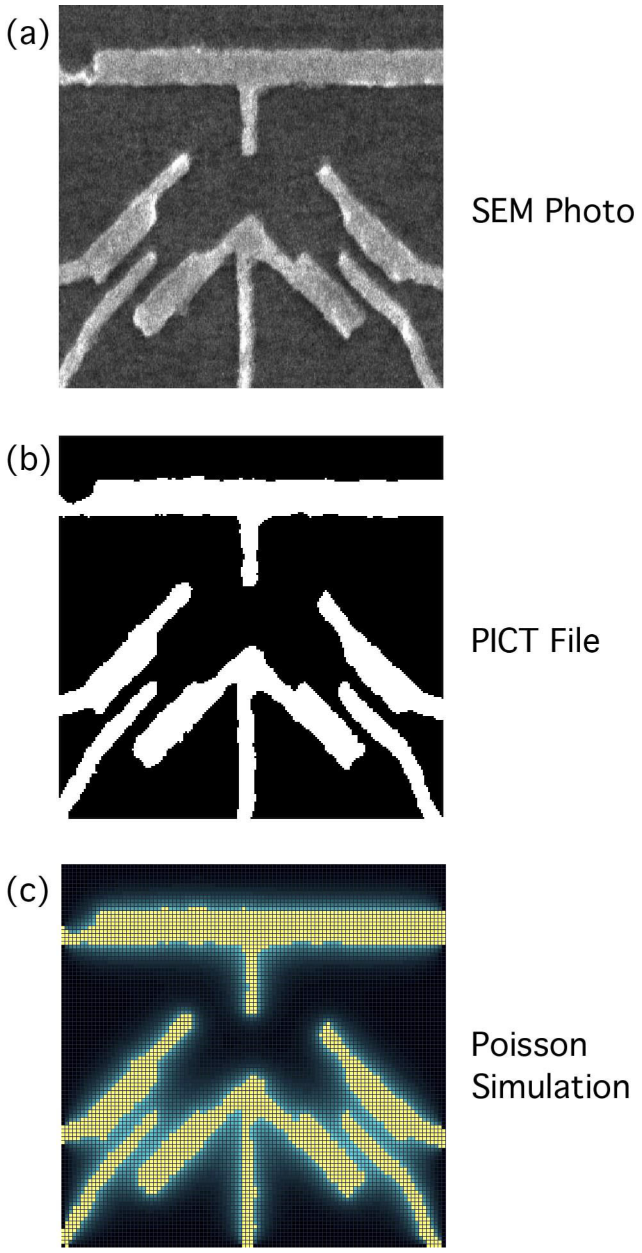


Figure 3.17 Electrostatic simulation of a gate pattern derived from an SEM photograph. (a) SEM photograph of a quantum dot device. (b) Grayscale picture of (a) after applying a “threshold” filter in Photoshop. Non-black areas will become gates, while pure black areas are “voids” that will not contain gates. (c) Electrostatic simulation of (b) in the Poisson Solver.

image [Figure 3.17(a)]. The non-black areas of the PICT define metal (with a voltage proportional to its brightness) while the black areas define "voids" where no metal is created. This pattern is read into the Poisson Solver with the command

```
metal.readPict axis start height volt_scale;
```

where *axis* defines the extrusion axis (1 = x, 2 = y, and 3 = z), *start* the coordinate at which extrusion of the pattern begins, and *height* the height of the extrusion. The brightness of the pattern is scaled by *volt_scale* so that pure white has a voltage *volt_scale* and pure black is zero, and the PICT file containing the pattern to be read in is selected using dialog boxes when the command in executed. Figure 3.17(c) shows the result of an electrostatic simulation using the PICT file of Figure 3.17(b). A similar command for making extruded dielectric patterns, `dielectric.readPict`, is also available, and almost any conceivable device can be constructed with these commands.

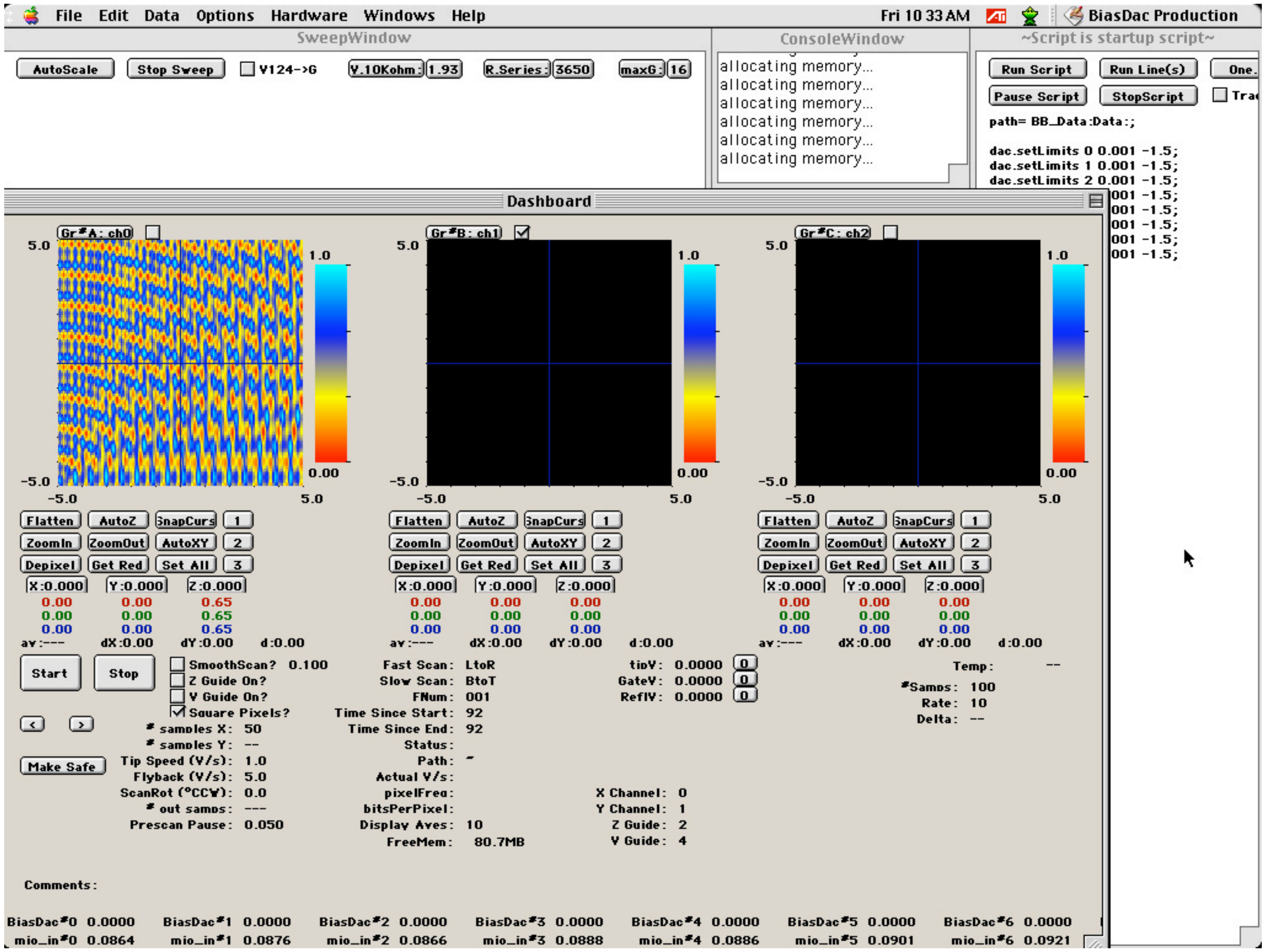
3.3.4 The AFM Program

The software used to control the BiasDac box and acquire data for quantum dot experiments is a modified version of the AFM program written by M. A. Topinka and B. J. LeRoy for their atomic force microscopy (AFM) scanning experiments. Given the massive size and complexity of the program, the reader must refer to Topinka [2002], LeRoy [2003], and this thesis to get a complete picture of how the program works, but I provide a brief description here. The current AFM program is written in C/C++ and runs in Mac OS 9 on any Power Macintosh computer (and in Mac OS X without hardware control). There are several versions of the program that will compile depending on the `#define` switch – `BIT16` compiles for use with the AFM, `NOHARDWARE` complies for data analysis purposes with no control of data acquisition hardware, and `BIASDAC` compiles for use with the BiasDac box used in quantum dot experiments. The current version of the program can perform one- or two-dimensional scans of arbitrary gate voltage channels while acquiring data using a National Instruments DAQ board on the computer. Furthermore, the program is scriptable and allows the data acquisition process to be automated. My modifications to the AFM program consist primarily of rewriting all the code for controlling output voltages to use the BiasDac, and adding features specific to quantum dot experiments like gate voltage compensation. The result is an incredibly powerful and flexible program, albeit with a rather steep learning curve for the uninitiated. A quick description of how to use the AFM program follows, but for a list of BiasDac commands and a fuller account of the modifications made to the program, please refer to Appendix A.5. The communications protocol for the BiasDac (Appendix A.2.3) describes the BiasDac command format and may also be useful in understanding the program.

Figure 3.18 shows the user interface of the AFM program. It consists of four windows: the Sweep window which shows the result of one-dimensional sweeps, the Console window which lists error messages, the Dashboard window which deals with two-dimensional scans, and the Script window in which commands can be entered. A one-dimensional sweep of voltage is started by running either of two commands in the Script window:

```
dac.sweep outCh startV endV stepSize inCh numAvg;
dac.sweepTo outCh endV stepSize inCh numAvg;
```

where *outCh* is the output channel (currently 0 to 7), *startV* is the starting voltage, *endV* is the ending voltage, *stepSize* is the spacing between points in the graph, *inCh* is the input channel to read (0 to 7), and *numAvg* is the number of readings to average over per point in the graph. In the first command, the selected output channel is ramped to the starting voltage before the sweep commences. In the second command, the sweep commences from the current output voltage. Zooming in and out of the data is done by selecting an area with the mouse in the Sweep window or by pressing the "AutoScale" button. A convenient feature exists in one-dimensional sweeps for turning lock-in output data into conductance data. This is activated by the "V124->G" checkbox in the Sweep window, which computes conductance via the formula $G = (A/V_{124} - R)^{-1}$, where G is the conductance, V_{124} is the lock-in output voltage, and A and R are fit parameters. A is



```
allocating memory...
allocating memory...
allocating memory...
allocating memory...
allocating memory...
allocating memory...
```

```
path= BB_Data:Data:;

dac.setLimits 0 0.001 -1.5;
dac.setLimits 1 0.001 -1.5;
dac.setLimits 2 0.001 -1.5;
001 -1.5;
001 -1.5;
001 -1.5;
001 -1.5;
001 -1.5;
```

```
X:0.000  Y:0.000  Z:0.000
0.00     0.00     0.65
0.00     0.00     0.65
0.00     0.00     0.65
av:---   dX:0.00  dY:0.00  d:0.00

SmoothScan?  0.100
Z Guide On?
V Guide On?
✓ Square Pixels?
# samples X:  50
# samples Y:  --
Tip Speed (V/s):  1.0
Flyback (V/s):  5.0
ScanRot (°CCW):  0.0
# out samps:  ---
Prescan Pause:  0.050

Fast Scan:  LtoR
Slow Scan:  BtoT
FNum:  001
Time Since Start:  92
Time Since End:  92
Status:
Path:  ~
Actual V/s:
pixelFreq:
bitsPerPixel:
Display Aves:  10
FreeMem:  80.7MB

tipV:  0.0000  0
GateV:  0.0000  0
RefIV:  0.0000  0

X Channel:  0
Y Channel:  1
Z Guide:  2
V Guide:  4

Temp:  --
#Samps:  100
Rate:  10
Delta:  --

Comments:

BiasDac#0 0.0000  BiasDac#1 0.0000  BiasDac#2 0.0000  BiasDac#3 0.0000  BiasDac#4 0.0000  BiasDac#5 0.0000  BiasDac#6 0.0000
mio_in#0 0.0864   mio_in#1 0.0876   mio_in#2 0.0866   mio_in#3 0.0888   mio_in#4 0.0886   mio_in#5 0.0901   mio_in#6 0.0921
```

Figure 3.18 User interface of the AFM program, rotated 90°. In the top left is the Sweep window that displays the results of one-dimensional voltage sweeps. In the top middle is the Console window that displays messages and errors. At right is the Script window in which commands for automating data acquisition are entered. The large front window is the Dashboard window that is the primary user interface for performing two-dimensional scans in voltage.

determined in the "V.10Kohm" button, where one enters the lock-in output voltage when measuring a 10 kΩ resistor. *R*, entered in the "R.Series" button, is the residual series resistance (typically ~2 kΩ from resistors in the leads) in our two-terminal measurements. The resulting conductance is displayed in units of e^2/h.

Two-dimensional scans are controlled and displayed in the Dashboard window. The Dashboard window can be broken into the four scan windows (of which three are shown) and their associated buttons (top), the universal scan controls and displays (middle), and the output and input voltage displays (bottom). A two-dimensional scan is first setup with the universal scan controls as listed below:

X Channel	Output channel that is taken to be *X*.
Y Channel	Output channel that is taken to be *Y*.
ScanRot	Counter-clockwise rotation of scan coordinates [i.e. $X' = X\cos(\theta) + Y\sin(\theta)$, $Y' = Y\cos(\theta) - X\sin(\theta)$].
Tip Speed	Scanning speed when acquiring data; also applies to one-dimensional sweeps.
Flyback	Scanning speed when *not* acquiring data (*e.g.* when moving to the start of a scan); also applies to one-dimensional sweeps.
Square Pixels?	Assumes the same number of *X* and *Y* pixels for a scan if checked.
Samples X	Sets the number of *X* pixels for the scan.
Samples Y	Sets the number of *Y* pixels for the scan (only if "Square Pixels?" is *not* checked).
Fast Scan	Direction to scan first: left to right (L to R) or vice versa, or top to bottom (T to B) or vice versa.
Slow Scan	Direction to scan second.
Prescan Pause	Amount of time to pause before scanning each line.
SmoothScan?	Smooth scan accelerates to the scanning speed instead of starting at the scanning speed; more relevant for AFM scanning.

The next steps are to set the input channel and define a scan area. These are done in any of the four scan windows (A to D). Enter the input channel with the "Gr#" button, and check the checkbox beside it. The scan area is defined in the scan window by the red and green cursors, which are moved by dragging with the mouse. The blue cursor displays the current *X* and *Y* voltages, and can also be moved with the mouse. A scan is started by pressing the “Start Scan” button, and interrupted with the “Stop Scan” button. Several input channels can be read simultaneously by checking multiple “Gr#” checkboxes. The two-dimensional scans are performed in a raster pattern with data acquired only while scanning in one direction (the “Fast Scan” direction). This is done intentionally to avoid problematic offsets that occur when scanning in two directions. In case anything goes wrong, the “Make Safe” button immediately sets all output voltages to zero.

3.3.5 AFM Program Tips and Tricks

Two-dimensional scans offer a great way to “tune” quantum dots. By scanning the voltages controlling both tunnel barriers to a quantum dot and reading the conductance through the dot, we obtain a conductance plot similar to Figure 3.19. In the top right corner of the Figure, both tunnel barriers are too open and no Coulomb blockade peaks are seen. Along the left and bottom edges of the Figure, one tunnel barrier is too closed and no conductance signal is observed. In the middle of the Figure, clear Coulomb blockade peaks are observed, indicating that both tunnel barriers are just right. Moving the blue cursors to the area of the Coulomb blockade peaks completes the tuning procedure.

When a quantum dot variable (*e.g.* electrostatic potential or tunnel barrier height) is affected by unintentional capacitive coupling to the scanned voltages, voltage compensation may be employed to keep that variable constant. The compensating voltage V_C is typically a linear sum of the scanned voltages *X* and *Y*: $\Delta V_C = \alpha\, \Delta X + \beta\, \Delta Y$ and can be implemented in the AFM program using the *Z* or *V* guides:

First, tilt a scan window by entering α with the “X” button and β with the “Y” button of the scan window.
Second, create a flat plane in the tilted scan window with the command

```
    constantscan scanWin val;
```

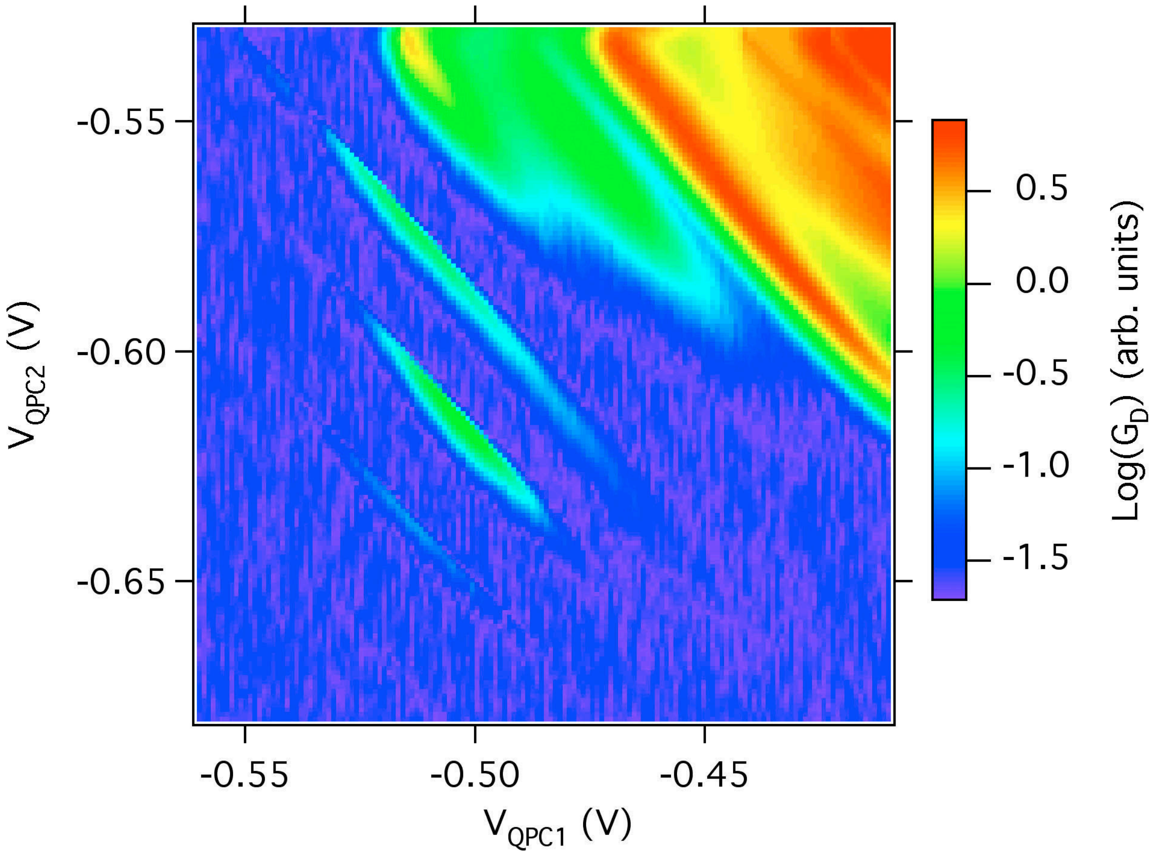


Figure 3.19 Plot of quantum dot conductance as tunnel barriers leading to the dot are "tuned." The voltages V_{QPC1} and V_{QPC2} control the heights of the tunnel barriers. In the upper right corner, both tunnel barriers are low and the dot is well connected to the leads. Conductance through the dot is high in this corner and no Coulomb blockade peaks are seen. Along the left and bottom edges of the graph, conductance is very low because one of the tunnel barriers is too high. In the middle of the graph, both tunnel barriers are "just right," and clear Coulomb blockade peaks are observed. Moving the blue cursors in the AFM program to this spot completes the tuning of the dot.

where *scanWin* is the scan window you want to create the plane in (a, b, c or d), and *val* is the height of the plane (0 should be used).
Third, un-tilt the scan window by setting both “X” and “Y” to zero.
Fourth, turn on the *Z* or *V* guide. This outputs the tilted plane on the *Z* or *V* guide channel as a function of the *X* and *Y* voltages.
Finally, the compensating voltage (the *Z* or *V* guide channel) is summed in hardware with the voltage to be compensated.

This technique is especially useful for electrometer-type measurements as the conductance of an electrometer dot is very sensitive to any changes in its electrostatic environment. Note a few caveats however. First, the compensating voltage for the *Z* guide *must* be placed in scan window A, and the compensating voltage for the *V* guide in scan window C. Second, the scan window used for guiding cannot be used for acquiring data. And third, the *Z* guide must be turned on to use the *V* guide.

3.4 Cryogenic Techniques

Cryogenic techniques are needed to cool the devices described in this thesis as the temperature kT needs to be much smaller than the charging energy E_C of these devices for Coulomb blockade to be observed (Chapter 2.4). For a device with a charging energy $E_C = 0.5$ meV ($\Rightarrow T = 5$ K), typical of the larger quantum dots of Chapter 4, a temperature of about $T = 0.5$ K is preferred, and improved resolution is achieved with even lower temperatures. All samples in this thesis were measured at the base temperature of a dilution refrigerator (T ~50 mK), unless otherwise noted.

3.4.1 He3/4 Dilution Refrigerators

A good reference that describes the physics of how a dilution fridge works is Lounasmaa [1974], but a brief overview follows. A dilution refrigerator is a closed-loop refrigeration system that uses a mixture of He3 and He4 as its working fluid. Below a certain temperature (depending on the proportion of He3 to He4), the mixture of He3 and He4 phase-separates into a He3-rich and a He3-poor phase. The He3 in the mixture has a higher vapor pressure than the He4, and is preferentially pumped out of the He3-rich phase into the He3-poor phase. The dilution of the He3 provides cooling in a process akin to evaporation, and allows our samples to be cooled down to several tens of mK.

The Westervelt and Tinkham groups share a Kelvinox 100 and a Kelvinox 400 dilution refrigerator. The operational differences between the two fridges are:

1. The Kelvinox 400 has an N_2 bath in addition to a He4 bath, so its hold time is longer (5 days versus 1.5 days).
2. The Kelvinox 400 has an indium-sealed inner vacuum can (IVC) rather than a cone-sealed one, and takes more effort to make and maintain.
3. The Kelvinox 400 has electro-mechanical valves in the gas handling system, and cooling may be placed under computer control (not always a good idea).
4. The Kelvinox 400 has an 11 T magnet [Duncan, 2000] while the Kelvinox 100 has a 7 T one.

Aside from these differences, the operation (and reliability) of both fridges is very similar. A condensed description of the operation of a dilution fridge follows; for more specific information, see Adourian [1996] regarding the Kelvinox 100, and Pohlen [1999] regarding the Kelvinox 400.

Both the Kelvinox 100 and 400 fridges are comprised of two major parts – the insert, and the dewar. The insert is the part of the fridge that gets below 4 K and contains the “innards” of the dilution fridge. The dewar is just a large, insulated container of He4 in which the insert and magnet reside. Most of the complexity of the dilution fridge lies within the insert, and that is what I will describe in more detail.

One confusing aspect to a novice is surely the number of spaces that exist in a dilution fridge insert. There are three separate spaces within the insert: the mixture space, the He4 space, and the inner vacuum space (IVC) (Figure 3.20). These three spaces are not connected and should never be confused. They should all be pumped out to high vacuum (*e.g.* with a turbo pump) before being inserted into a cold dewar. The IVC functions as the thermal link between the innards of the fridge (the mixture and He4 paths) and the helium bath of the dewar. On cooling down the insert from room temperature, the thermal link is made by filling the IVC with helium exchange gas so that heat can be conducted from the innards of the insert to the 4 K bath of the dewar. Once the insert reaches 4 K, the thermal link to the bath is broken by pumping out the exchange gas from the IVC. The still and mixing chamber heaters in the insert should be turned on for no more than an hour during this process.

Once the thermal link to the bath has been severed, the insert can be cooled down further to about 1.5 K by pumping on He4. This is performed in the He4 space (Figure 3.20). Liquid He4 from the bath is introduced into the He4 space by opening the needle valve to fill the 1K pot. This is then pumped out with the 1K pot pump attached to the other end of the He4 space, and the evaporative cooling of the He4 cools the insert.

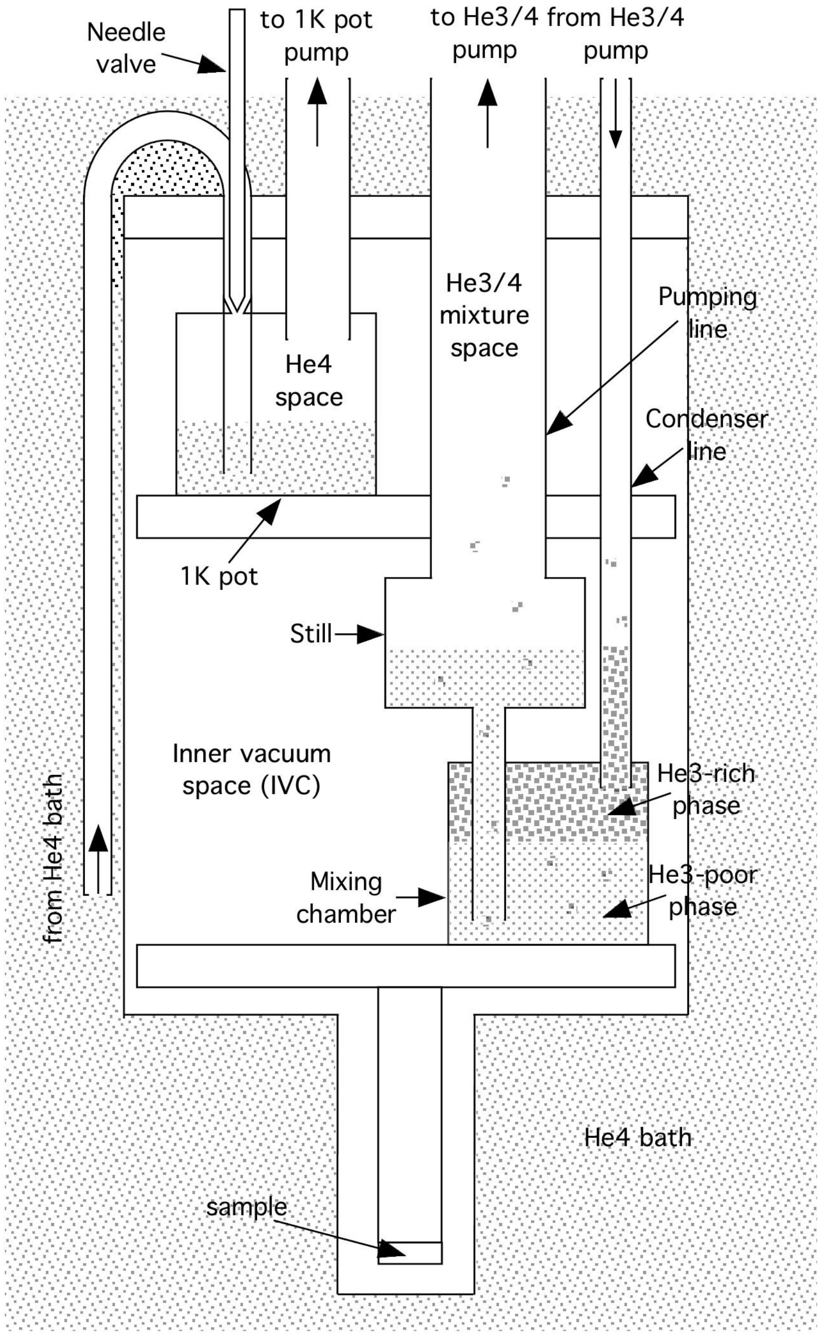


Figure 3.20 Schematic showing the insert of a He3/4 dilution refrigerator. There are three different spaces in the insert – the mixture space, the He4 space, and the inner vacuum space. The inner vacuum space is a switchable thermal link between the innards of the insert and the He4 bath of the dewar. Helium exchange gas introduced into this space makes the link, but pumping the gas out breaks it. The He4 space cools a portion of the insert to 1.5 K via evaporative cooling of liquid He4 drawn from the bath. The mixture space is where the He3/4 mixture circulates and cools the sample to ~50 mK as the He3 "evaporates" from the He3-rich phase to the He3-poor phase of the mixture.

To cool the insert further, the mixture space is used (Figure 3.20). He3/4 mixture from the dumps is introduced slowly into the mixture space, and condenses into liquid from heat exchange with the 1K pot. When all the mixture has been condensed into the He3/4 space, circulation of the mixture can begin. Evaporative cooling of the mixture initially cools the insert down further. When the temperature gets low enough, the mixture separates into the He3-rich and He3-poor phases and the dilution process takes over. The mixing chamber, where the sample is heat sunk, is the coldest part of the fridge, and gets down to several tens of mK.

There are several important rules to heed when handling the He3/4 mixture. First, the He3/4 mixture should never be mixed with "regular" He4, or vented to the atmosphere. Not only is He3 very, very expensive (~$300 per liter of gas!), the correct balance between He3 and He4 is also required so that the boundary between the He3-rich and He3-poor phases is at the correct level in the insert. Valves on the gas handling system marked with red tape require additional thought before operation as these valves contain the mixture behind them. Second, and equally important, the valves on the dumps should remain open while the mixture is "out" in the insert. This ensures that the mixture will be recovered in case the fridge warms up due to a power failure or a blockage in the circulation path. Otherwise, the mixture will be vented to atmosphere when the pressure of the mixture in the insert rises above atmospheric pressure.

There are two additional points to note. First, the needle valves of both fridges leak, so the He4 spaces cannot be evacuated completely before the inserts are put into their dewars. The needle valves can then get frozen shut from ice that could form, rendering the needle valves inoperable. The solution is to fill the He4 spaces with He4 gas to atmospheric pressure before quickly inserting the inserts into their dewars so that water vapor cannot get in. Second, the mixture in the Kelvinox 100 was lost and subsequently replenished. The resulting mixture contains slightly too much volume, and will not circulate if all the mixture from the dumps is condensed. So instead of pulling all the mixture out of the dumps with the He3/4 pump, the mixture should be condensed without using the pump, which leaves behind about 25 mbar in the dumps.

3.4.2 Kelvinox 100 Dewar Wiring

The wiring in the Kelvinox 100 dewar was re-made as the He level meter and thermometry (carbon resistor) were not working. Cryogenic cables from Lake Shore Cryotronics were used (36 awg "Quad-Lead" wires) to minimize the conducted heat load on the helium bath. A new He level meter (a 16" long sensor from American Magnetics Inc.) was wired on connector A as follows:

Connector A pins	**What**	**Description**
J	He level meter (black)	current lead (I+).
K	He level meter (red)	current lead (I–).
A	He level meter (yellow)	voltage lead (V+).
H	He level meter (blue)	voltage lead (V–).

The top-of-magnet resistor and magnet persistent switch were wired on connector B as follows:

Connector B pins	**What**	**Description**
E & J	Top-of-magnet resistor	330 Ω carbon resistor. 410 Ω at 77 K.
B & D	Magnet persistent switch	45 mA required to turn off persistent switch.

Chapter 4: Strongly Capacitively-Coupled Quantum Dots

This chapter explores using strongly capacitively-coupled quantum dots to make a single-electron switch. Conventional quantum dots fabricated side-by-side only have weak capacitive coupling between them because there is only an edge-on capacitance. By making a coupling gate that covers two adjacent dots, the coupling between them is increased by an order of magnitude because the areas of the dots are used. The strong capacitive coupling allows an electron entering one dot to alter the conductance of the other dot in a non-linear fashion and make a single-electron switch. The switching lineshape of the single-electron switch is studied, and found to depend on the charge quantization in the dot triggering the conductance switching.

Section 4.1 provides an introduction to the study of coupled quantum dots, and Section 4.2 describes the capacitive charging model of capacitively-coupled dots. The strongly capacitively-coupled quantum dot device is described in Section 4.3, and the measurement setup and basic device characterizations are described in Section 4.4. The measured Coulomb blockade behavior of the strongly capacitively-coupled dot device is detailed in Section 4.5, and comparisons with theory and with tunnel-coupled dots are made. The use of the device as a single-electron switch is explained in Section 4.6, and the factors affecting the switching lineshape are discussed in Section 4.7. And finally, the conclusions to the chapter are made in Section 4.8.

4.1 Introduction

The electrical properties of single quantum dots have been well-studied, and the Coulomb blockade phenomenon in a single dot is well-understood [Grabert and Devoret, 1992; Sohn *et al.,* 1997; Likharev, 2000]. In studying two interacting quantum dots, two issues are raised: one is the nature of the coupling between the dots (tunneling or capacitive), and the other is the electrical configuration of the dots (in series or in parallel).

Tunnel-coupled dots in series were studied in this group by Waugh [1994], Crouch [1996], and Livermore [1998]. In particular, the experiments of Livermore *et al.* [1996] showed the evolution of the Coulomb blockade peaks of a series double dot as interdot tunneling was increased, and indicated how the two dots were merging into one larger dot. Capacitively-coupled (and tunnel-coupled) dots in parallel were studied by Adourian [1996], and he found that the Coulomb blockade peaks of one dot were shifted in gate voltage when an electron entered or left the adjacent dot. This shift was due to the change in the electrostatic potential of the dot whenever an electron was added to or removed from the adjacent dot, and allowed the dot to work as an electrometer to sense nearby charges [Livermore, 1998; Duncan *et al.*, 1999; Yacoby *et al.*, 1999].

The capacitive-coupling between the dot and the charges under study is typically quite weak however, and only about 3 to 5% of field lines from an electron of interest couple to the electrometer dot [Livermore, 1998; Duncan *et al.*, 1999]. This is because the dot has an edge-on capacitance rather than an area-on capacitance to the charges of interest. The weak coupling also means that the charge acts as a perturbation to the conductance of the dot, changing it linearly.

This chapter sets out to explore stronger capacitive coupling, and to use the stronger coupling to change a dot's conductance non-linearly and make a single-electron switch. By single-electron switch, I mean a device that turns on or off from the presence or absence of a single electron. Now, quantum dots are also known as single-electron transistors (SETs), but that is somewhat of a misnomer. Although a quantum dot conducts electricity (turns on) or not depending on whether an additional electron can hop onto the dot or not, that one additional electron in the dot is induced onto the dot by many billions of electrons on the sidegate of the dot. For a true single-electron switch, the device must be turned on by the presence of a single electron in another part of the circuit, and strong capacitive coupling between that electron and the dot is needed to accomplish that. This would then open up the possibility of building computers that work by moving single electrons around, and would possibly represent the ultimate in circuit miniaturization.

4.2 Coulomb Blockade Theory of Capacitively-Coupled Dots

This section describes the capacitive charging model of Coulomb blockade for capacitively-coupled dots [see, *e.g.*, Tinkham, 1996; Livermore, 1998]. In this model, one treats quantum dots as isolated capacitors that are charged with electrons via the leads, and which potentials are altered by the sidegates.

4.2.1 Capacitive Charging Model of One Dot

Starting with the simplest case, we consider the free energy of a single quantum dot [Figure 4.1(a)]. Its free energy is computed by adding the energy stored in the capacitors of the dot and subtracting the work performed by the leads in charging them, giving:

$$E(V_{\mathrm{G1}},M) = \tfrac{1}{2C_\Sigma}(C_\mathrm{G}V_{\mathrm{G1}} - Me)^2 \tag{4.1}$$

where M is the number of electrons on the dot. This intuitively means that the free energy of the dot is proportional to the square of the difference between the induced charge on the dot $C_\mathrm{G}V_{\mathrm{G1}}$ and the actual charge on the dot Me. The quantum dot therefore has an effective charge $Me - C_\mathrm{G}V_{\mathrm{G1}}$ because, while the induced charge is continuous, the actual charge can only be in discrete units of e [Figure 4.1(b)]. The effective charge can be of either polarity, but is limited to an absolute value of $|e/2|$ before it becomes energetically unfavorable.

The parabolas of energy are shown in Figure 4.1(c) for different values of M, and at sidegate voltages where different parabolas intersect (*e.g.* the m and m+1 parabolas), two different charge states of the dot are energetically allowed and one electron at a time may hop on and then off the dot to carry electricity through the dot. These degeneracies

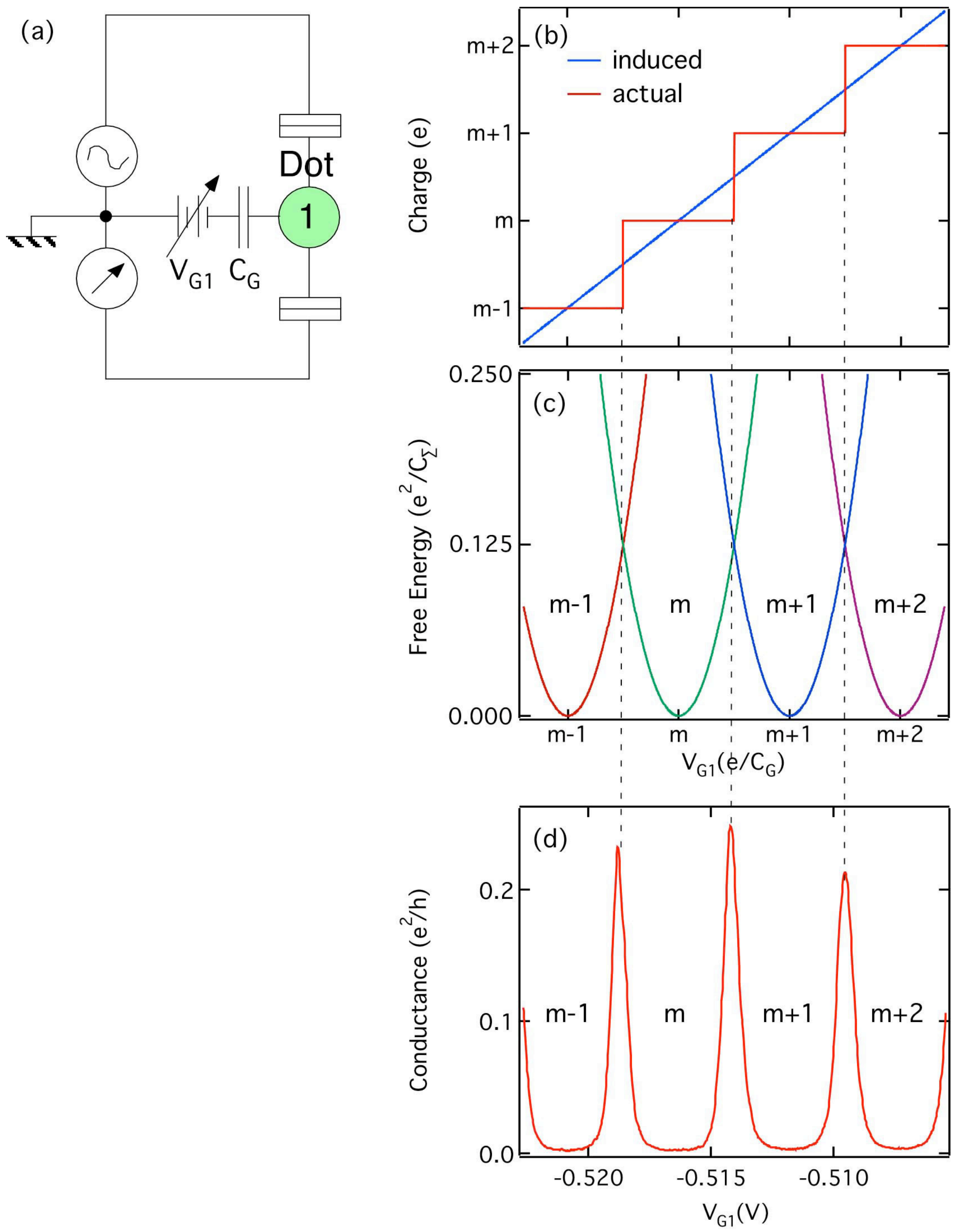


Figure 4.1 Capacitive charging model for a single quantum dot. (a) Schematic diagram of the quantum dot circuit. (b) Graph showing the difference between the charge induced on the dot by a sidegate, which is continuous, and the charge that can actually reside on the dot, which is quantized, as a function of the sidegate voltage. (c) The difference between the induced and actual charge on the dot creates an effective charge which gives rise to parabolas in energy for different numbers of electrons on the dot. Where the parabolas intersect, the number of electrons on the dot can change by one and current can flow, as electrons hop on and off the dot. (d) This current corresponds to the Coulomb blockade peaks in the conductance of the dot.

in energy therefore give rise to Coulomb blockade peaks as shown in Figure 4.1(d). At other sidegate voltages where no energy degeneracies occur, the number of electrons in the dot is energetically stable and fixed, and no current flows.

4.2.2 Capacitive Charging Model of Two Independent Dots

This model is easily extended to two *independent* dots [Figure 4.2(a)], where the total free energy is just the sum of the energies of the component dots:

$$E(V_{\mathrm{G1}},M,V_{\mathrm{G2}},N) = \tfrac{1}{2C_\Sigma}\left[(C_\mathrm{G}V_{\mathrm{G1}} - Me)^2 + (C_\mathrm{G}V_{\mathrm{G2}} - Ne)^2\right] \tag{4.2}$$

where M and N are the number of electrons in Dot 1 and Dot 2 respectively, and V_{G1} and V_{G2} are the sidegate voltages of Dot 1 and Dot 2 respectively (the capacitance C_G between sidegate and dot and the dot self-capacitance C_Σ are assumed to be identical for both dots). The free energy surface of two independent dots for different values of M and N is therefore a two-dimensional array of paraboloids in V_{G1} and V_{G2} as shown in Figure 4.2(b). The intersections between the paraboloids (where the energy degeneracy between different charge states occur) are straight vertical and horizontal lines, and give rise to straight horizontal and vertical Coulomb blockade peaks as shown in schematic in Figure 4.2(c).

The regions of blockaded conductance correspond to stable charge configurations (M, N) of the double dot as shown. The vertical Coulomb blockade peaks belong to Dot 1 (that is, current flows only through Dot 1 along the vertical peaks). This is because those peaks are formed by the intersection of charge states like (m, n) and (m+1, n), where only the charge on Dot 1 can fluctuate and carry current. In a similar way, the *horizontal* Coulomb blockade peaks belong to Dot 2, as they are formed by intersections where only the charge on Dot 2 can fluctuate and carry current [*e.g.* (m, n) and (m, n+1)]. This Coulomb blockade pattern makes intuitive sense as the dots are independent and exhibit Coulomb blockade peaks only as a function of their own sidegate voltage.

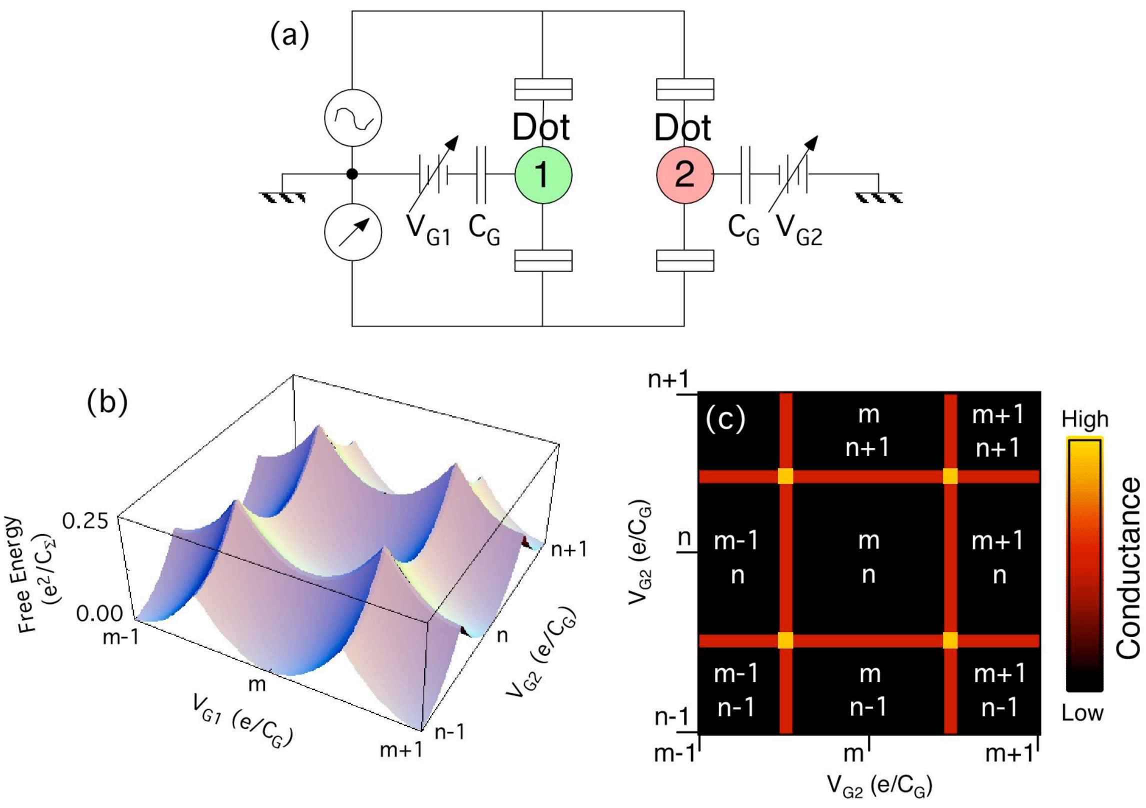


Figure 4.2 Capacitive charging model for uncoupled double quantum dots in parallel. (a) Schematic diagram of the double dot circuit. The free energy of the double dot is the sum of the free energies of the two individual dots. (b) This results in two-dimensional array of paraboloids in the two sidegate voltages, as shown in the surface plot. The intersections between the paraboloids, shown as ridges, are energy degeneracies between different charge states of the double dot. (c) These intersections form straight vertical and horizontal Coulomb blockade peaks as shown in the schematic. The charge state of the double dot that is stable for each square in the conductance is labeled for Dot 1 and Dot 2 respectively.

4.2.3 Capacitive Charging Model of Two Capacitively-Coupled Dots

When capacitive coupling is added to the model [Figure 4.3(a)], the free energy becomes [Livermore *et al.*, 1996]

$$E(V_{G1},M,V_{G2},N) = \tfrac{1}{(1-\alpha^2)2C_\Sigma}\Big[(C_GV_{G1}-Me)^2+(C_GV_{G2}-Ne)^2 + 2\alpha(C_GV_{G1}-Me)(C_GV_{G2}-Ne)\Big] \tag{4.3a}$$

where the fractional capacitive coupling $\alpha \equiv C_{INT}/C_\Sigma$ and C_{INT} is the interdot coupling capacitance. The free energy landscape now looks like Figure 4.3(b) (for $\alpha = 0.2$), where the energy paraboloids have become distorted. The additional coupling term in Equation (4.3a) expresses the electrostatic interaction between the excess charges in the two dots. When the excess charges on both dots are of the same polarity (in the direction $V_{G1} = V_{G2}$), the dots repel each other and the energy goes up. When the excess charges on both dots are of the opposite polarity (in the direction $V_{G1} = -V_{G2}$), the dots attract each other and the energy goes down. A similar understanding may be reached by rewriting Equation (4.3a) in terms of the sum ($V_{G1} + V_{G2}$) and difference ($V_{G1} - V_{G2}$) axes:

$$E(V_{G1},M,V_{G2},N) = \tfrac{1}{(1-\alpha^2)4C_\Sigma}\Big[(1+\alpha)(C_GV_{G1}+C_GV_{G2}-Me-Ne)^2 + (1-\alpha)(C_GV_{G1}-C_GV_{G2}-Me+Ne)^2\Big] \tag{4.3b}$$

Here, we find that the energy of the capacitively-coupled dots is scaled up by a factor $(1 + \alpha)$ in the sum axis (where excess charges in both dots are of the same polarity) while it is scaled down by a factor $(1 - \alpha)$ in the difference axis (where excess charges are of opposite polarities).

These distorted energy paraboloids form new intersections and give rise to a hexagonal Coulomb blockade pattern as shown in schematic in Figure 4.3(c). The formerly straight Coulomb blockade peaks (see Section 4.2.2) are now broken into line segments, and the Coulomb blockade peaks are said to be "split." The maximum difference in energy between coupled and uncoupled dots occurs when the excess charges in both dots are $|e/2|$. The resulting energy difference is (to first order in α) $\pm\alpha E_C/4$, which

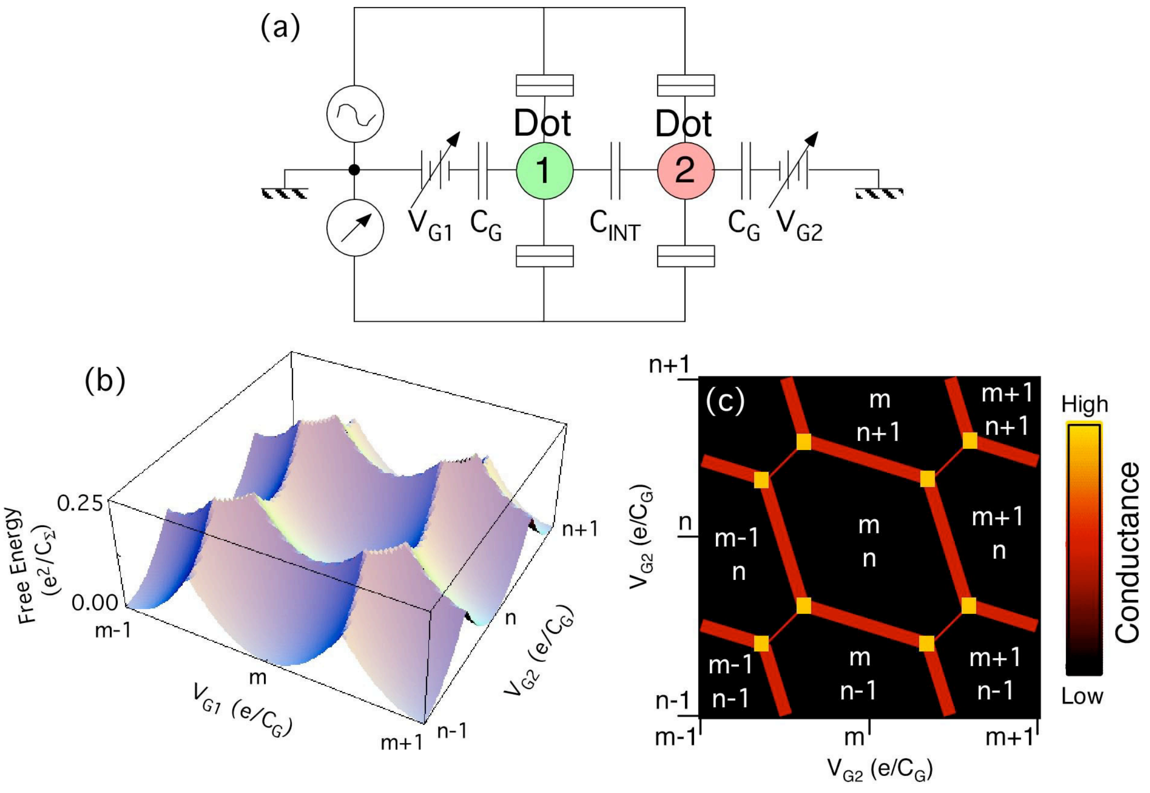


Figure 4.3 Capacitive charging model for capacitively-coupled double quantum dots in parallel. (a) Schematic diagram of the coupled double dot circuit. (b) The free energy of the coupled double dot [Eq. (4.3a)] has an additional term for the interaction between effective charges on the two dots, and distorts the paraboloids as shown in the surface plot. (c) The intersections between paraboloids are also modified and form a hexagonal Coulomb blockade pattern as shown in the schematic.

can be considered the capacitive coupling energy between the coupled dots. For the peak splitting to be observed clearly, this coupling energy must be much larger than the dominant energy uncertainty in the system. For dots in the thermally-smeared regime, this means $\alpha E_C >> kT$. This is also the same criterion for turning capacitively-coupled dots into single-electron switches.

4.3 Strongly Capacitively-Coupled Double Dot Device

The strongly capacitively-coupled parallel double quantum dot used in these experiments is shown in the colorized SEM photograph in Figure 4.4. The light gray areas are surface metal gates used to define the two quantum dots, and the dark vertical line down the middle is an etched trench that separates the two dots (and indeed, divides the 2DEG into two). The unique coupling gate, which covers both dots and serves to concentrate electric field lines between the dots, is highlighted in yellow. The electrical leads of the left-hand dot, Dot 1, are colored green while those of the right-hand dot, Dot 2, are colored red. The use of green to highlight data acquired from Dot 1 and red to highlight data acquired from Dot 2 will be used throughout this chapter. The voltages applied to the different gates of the device are defined by the black text in Figure 4.4.

The device was fabricated in an accumulation layer 2DEG labeled "KM3" by the Westervelt lab, and "960924C" by the growers at U. C. Santa Barbara. This wafer was grown by Kevin Maranowski and consists of:

50 Å GaAs (surface)
300 Å AlGaAs
8×10^{12} cm^{-2} Si δ-doping
220 Å AlGaAs
1 μm GaAs
20-period smoothing superlatice:
  20 Å AlGaAs
  20 Å GaAs
GaAs buffer
GaAs substrate

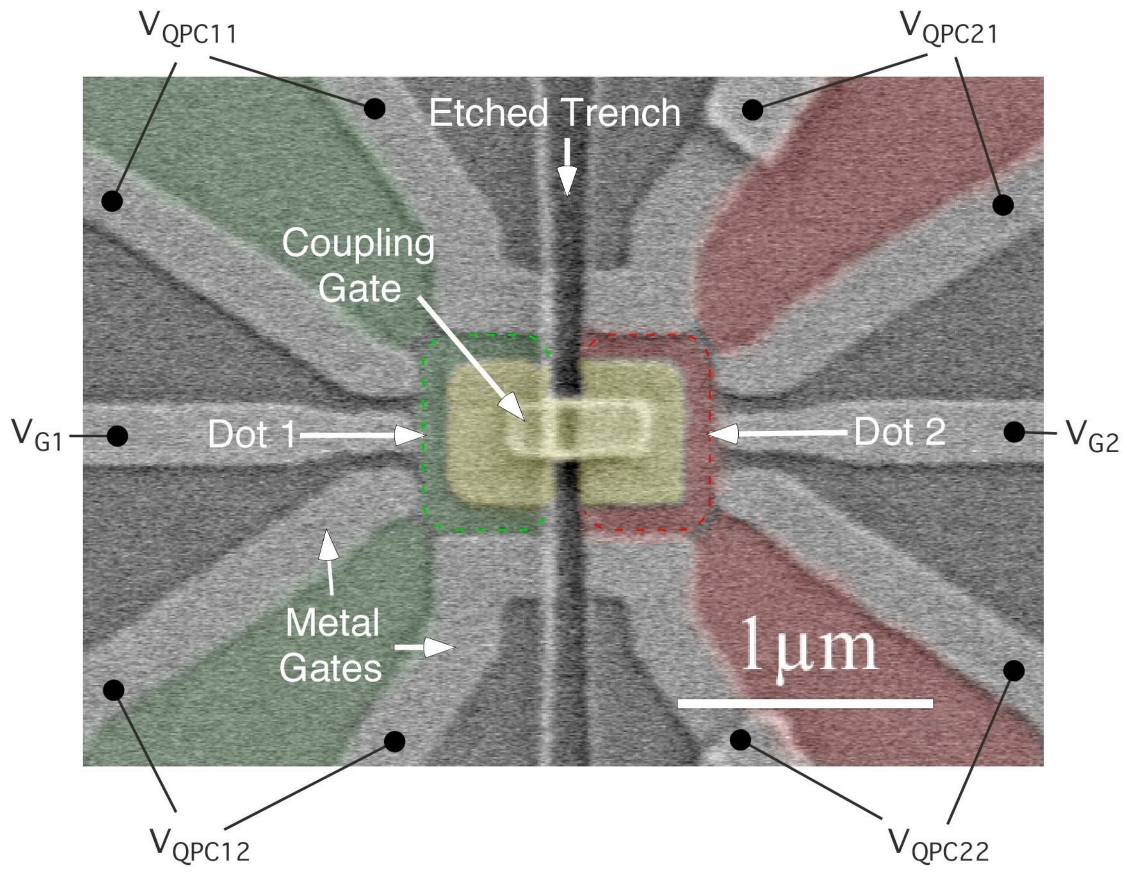


Figure 4.4 SEM photograph of the strongly capacitively-coupled double dot device. The light areas are metal surface gates that define the double dots. The dark vertical feature is an etched trench that prevents any tunneling between the double dots. The coupling gate, highlighted in yellow, provides strong capacitive-coupling between the dots. The dots are 57 nm below the surface of the semiconductor heterostructure, directly under the coupling gate. The path of current flowing through the left-hand dot, Dot 1, is colored green, while the path of current flow through the right-hand dot, Dot 2, is colored in red. The gate voltage applied to each gate is labeled by the black text.

making the 2DEG 57 nm below the surface of the wafer. A mobility μ = 450,000 cm^2/Vs and density $N = 3\times10^{11}$ cm^{-2} were measured by Kevin Maranowski at a temperature of 10 K using standard Shubnikov-deHaas measurements. All measurements of the strongly capacitively-coupled double dot device were made on sample "DDE 5.2," named after the acronym for "Double Dot Etch." It was the second device in the fifth batch of samples made. About 24 devices were attempted in total.

The steps used in fabricating this device are shown in Figure 4.5. First, a gate layer of metal was fabricated on the heterostructure. This layer also contained metallic alignment marks for aligning subsequent layers to. The capacitor plates of the coupling gate were also fabricated in this step to simplify the alignment process. Next, the etched trench was fabricated to prevent tunneling between the dots. An etch was used here instead of a more conventional metal gate so as not to short out the coupling gate. Gate metal near alignment marks would sometimes be etched (due to unintended exposure of the PMMA during alignment), and had to be patched by evaporating additional gate metal in an extra step. The final layer fabricated was another metal gate layer used to connect the two capacitor plates of the coupling gate together across the etched trench. A thick layer of metal (500 Å) had to be deposited to ensure connectivity across the trench, and required a tri-layer PMMA resist. This made alignment of the final layer more difficult because the alignment marks were obscured by the thick resist.

A difficult problem with the coupling gate was the fact that it was not connected to any voltage source and could float to an arbitrary potential. Samples were made in which the coupling gate appeared to be negatively-charged and depleted the quantum dots below. This was evidenced in measurements by the conductance of the sidegate going to zero immediately upon the sidegate depleting the 2DEG. This indicated that there was no current path below the coupling gate, and hence that no dot could be formed. No low-resistance shorts were found between the coupling gate and the sidegates, and thermally-cycling the samples did not help. High-resistance shorts were then suspected to be the

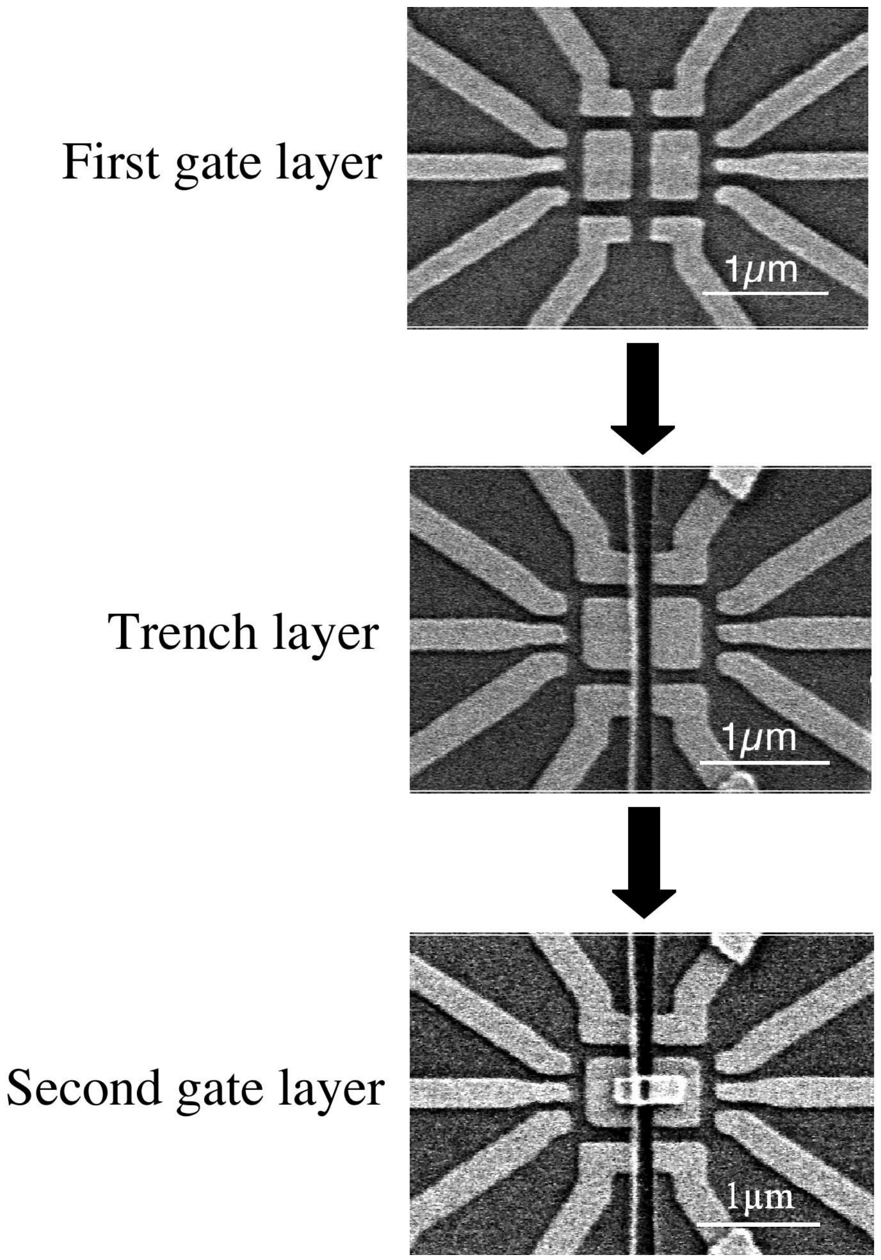


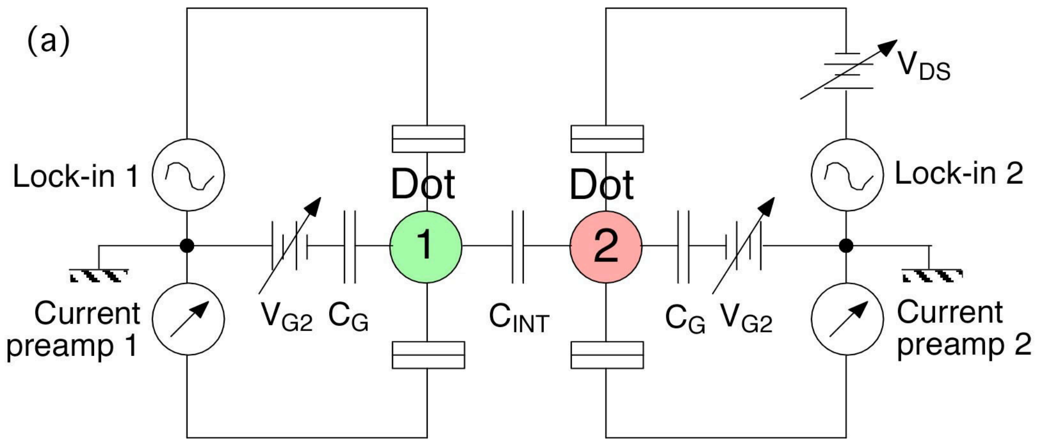
Figure 4.5 SEM photographs of the double dot device during the fabrication process. First, a layer of metal gates was patterned, including the two capacitor plates of the coupling gate. This layer also contained alignment marks (not shown) for aligning subsequent layers to. The next layer fabricated was the etched trench that separates the dots and divides the 2DEG into left and right halves. The final layer of fabrication involved depositing metal to connect the two capacitor plates across the etched trench to complete the coupling gate. A layer of PMMA resist was applied to the completed device to protect it against airborne contaminants (see text).

cause of the problem. Assuming a simple *RC* model for the charging of the coupling gate and a coupling gate self-capacitance $C = 1$ fF, an extremely high resistance $R = 10^{15}\ \Omega$ was required between the coupling gate and the other gates to prevent the coupling gate from charging up for just one *second*. Spinning a layer of PMMA resist onto newly completed devices was found to solve the problem. It is speculated that entombing the device in PMMA works because it prevents airborne contaminants from forming high-resistance shorts between the coupling gate and the other gates. Once the extra PMMA layer was spun on and the device found to work, the device was always found to work. The PMMA had to be exposed over the bonding pads and ohmics to allow wirebonds to be made to the sample of course.

The lithographic area of each dot is 500×800 nm^2. Assuming a 50 nm depletion region around each dot, a nominal dot area of 400×700 nm^2 is deduced, giving an estimated 800 electrons in each dot. Combined with the Fermi energy of 11 meV for the 2DEG, the single particle energy level $\Delta\varepsilon$ is estimated from Equation (2.3) to be 30 μeV. This is larger than the thermal energy kT of the experiments so current transport through the dot may be assumed to involve individual dot states, but is much smaller than the charging energy E_C and may be neglected otherwise.

4.4 Measurement Setup and Diagnostic Measurements

Measurements of the device were made in a Kelvinox 400 dilution fridge with the measurement setup as shown in Figure 4.6(a). Conductances of both dots were measured simultaneously with two PAR 124A lock-in amplifiers and two Ithaco 1211 current preamplifiers operating at different lock-in frequencies. Gate voltages were supplied by the Ramper box (see Chapter 3.2.2) or the Ramper box output summed with a computer-controlled output (see Chapter 3.2.5). For applying a finite dc bias across Dot 2, a computer-controlled output was summed with a lock-in ac excitation of 10 μV_{rms}.



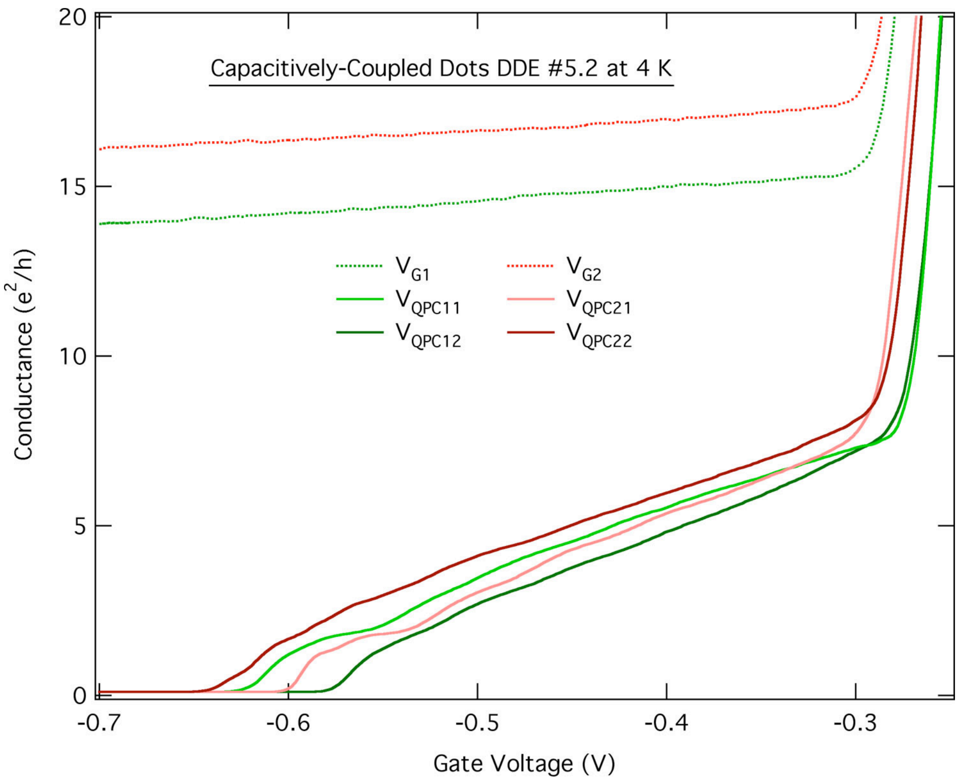


Figure 4.6 (a) Schematic showing the experimental setup. The conductances of both dots were measured simultaneously with independent sets of equipment. Additionally, a dc bias could be applied to Dot 2 for finite bias Coulomb blockade measurements. (b) Conductance data taken at 4 K verifying that each gate of the device works (see Figure 4.4 for gate voltage definitions). The sidegates of both dots did not pinch-off conductance through their respective current paths, while the QPCs did.

Figure 4.6(b) shows some diagnostic data taken at 4 K illustrating the conductance of the device as a function of the various individual gate voltages defined in Figure 4.4. It indicates that all gates for both dots work properly – point contact gates (V_{QPC11}, V_{QPC12}, V_{QPC21}, and V_{QPC22}) pinch-off when enough voltage is applied, whereas sidegates (V_{G1} and V_{G2}) do not. No conductance across the etched trench was detected ($R > 100$ MΩ), so the coupling between dots is strictly via capacitive-coupling only. Aside from Figure 4.6(b), all other data was acquired at the base temperature of the fridge. By fitting Coulomb blockade peaks to Equation (2.1), the electronic temperature of the dots was determined to be 70 mK, implying that $kT = 6$ μeV.

Figure 4.7(a) shows the finite bias Coulomb blockade data of Dot 2 from which various single-dot parameters were extracted (see Chapter 2.4.2). The Coulomb blockade diamond is visible as the yellow diamond in the center of the Figure, and the measured dot parameters are:

$E_{\mathrm{C}} = 380$ μeV
$C_{\Sigma} = 420$ aF
$C_{\mathrm{G}} = 23$ aF
$C_{\mathrm{L}} = 80$ aF,

the most important of which is the charging energy E_{C}. The parameters for Dot 1 are assumed to be the same as those for Dot 2 since both dots are nominally identical. During the measurement, Dot 1 was formed so the single-dot parameters obtained for Dot 2 are valid within the double dot system. The conductance of Dot 1 remained in the blockaded regime throughout the measurement and no change in its charge state occurred [Figure 4.7(b)].

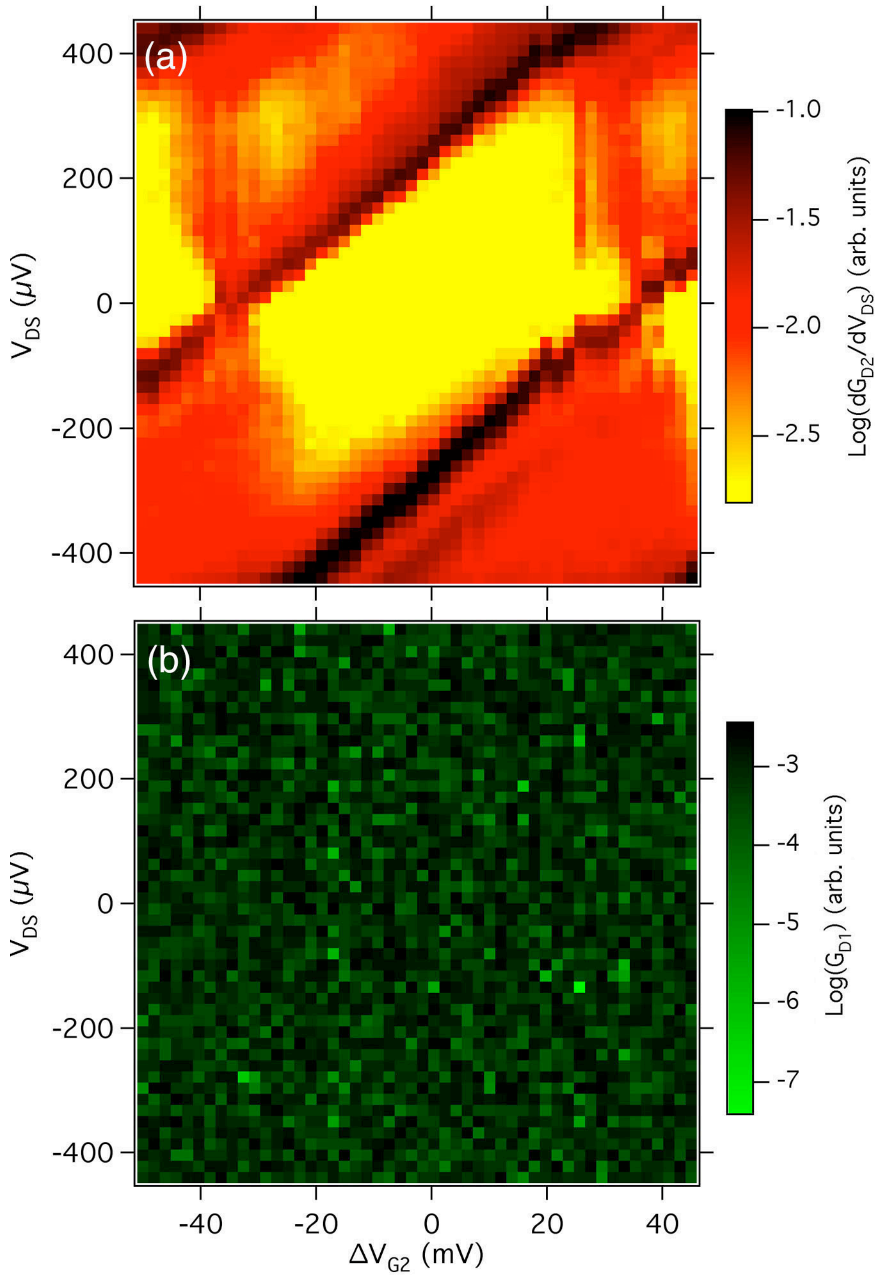


Figure 4.7 (a) Finite bias Coulomb blockade data of Dot 2. The differential conductance of Dot 2 was measured as a function of its sidegate voltage and the dc bias across Dot 2. The yellow diamond is the Coulomb blockade diamond from which various dot parameters were extracted (see text for values). (b) Conductance data of Dot 1 during the measurement of Dot 2 in (a). It shows that the current through Dot 1 was blockaded, and thus that the charge state of Dot 1 was fixed while Dot 2 was being measured.

4.5 Double Dot Coulomb Blockade Measurements

Figures 4.8(a) and (b) show the conductance G_{D1} of Dot 1 and G_{D2} of Dot 2 as a function of the two sidegate voltages ΔV_{G1} and ΔV_{G2}. Both quantum dots were adjusted so that they were weakly tunnel-coupled to their leads and charge on both dots was well-quantized. The conductances of both dots were measured simultaneously using different lock-in amplifiers operating at different frequencies. As mentioned in Section 4.2.2, if the two dots were uncoupled and independent, we would expect straight lines of Coulomb blockade peaks: vertical lines of peaks for Dot 1 (because Dot 1ferent frequencies. As mentioned in Section 4.2.2, if the two dots were uncoupled and independent, we would expect straight lines of Coulomb blockade peaks: vertical lines of peaks for Dot 1 (because Dot 1ΔV_{G1}) and horizontal lines of peaks for Dot 2 (because Dot 2's state would only be a function of ΔV_{G2}). However, what is observed in these strongly capacitively-coupled dots is that the Coulomb blockade peaks are split, as clearly shown in Figures 4.8(a) and (b). Superimposing the conductances of both dots on top of each other to obtain the conductance of the two dots in parallel, we obtain Figure 4.8(c), which shows that the peak splittings in both dots occur simultaneously, and that the Coulomb blockade peaks form a hexagonal pattern. This hexagonal pattern is exactly the expected Coulomb blockade pattern for capacitively-coupled quantum dots [Figure 4.3(c)], as explained in Section 4.2.3.

A couple of things bear noting. First, there is an interesting intersection of the (m, n) and (m+1, n-1) paraboloids, indicated in Figure 4.9(a) as side (iii). This means that the charge states of the two dots are coupled, and changing the charge state of Dot 1 requires changing the state of Dot 2 simultaneously. Current flows through both Dot 1 and Dot 2 along this Coulomb blockade peak in equal amounts, and the Coulomb blockade peak along side (iii) is yellow (= green + red). However, the probability that an electron enters Dot 1 at the same time that an electron leaves Dot 2 (or *vice versa*) is much smaller than the probability of either event alone, and the conductance of the

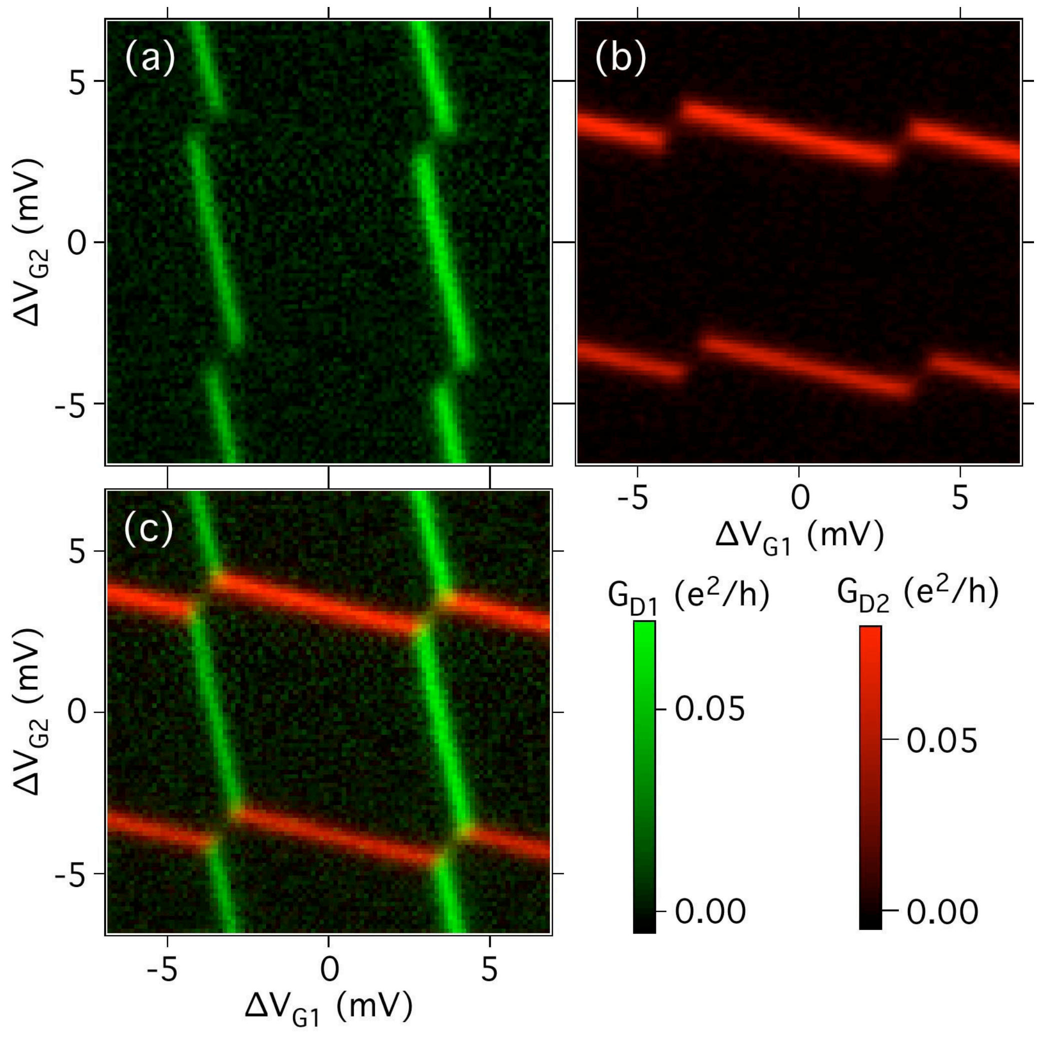
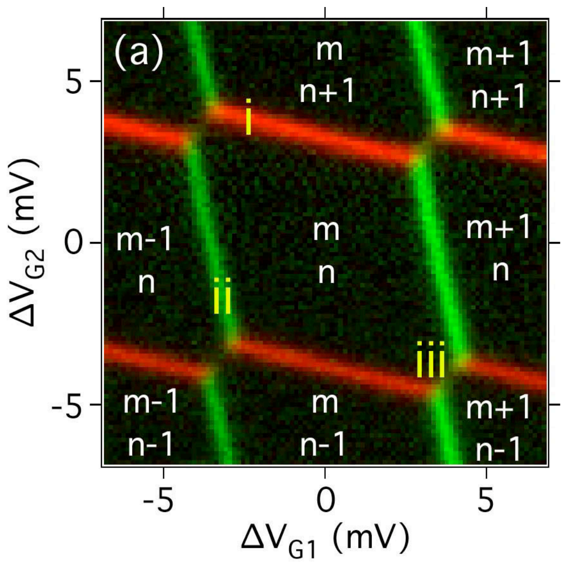


Figure 4.8 Coulomb blockade data of (a) Dot 1 and (b) Dot 2 as a function of the sidegate voltages of the two dots. Clear splitting of the Coulomb blockade peaks of both dots is evident. (c) Superposition of (a) and (c), indicating that the peak splitting in both dots occur simultaneously in sidegate voltages. The hexagonal pattern of Coulomb blockade peaks is indicative of strong capacitive-coupling between the dots.



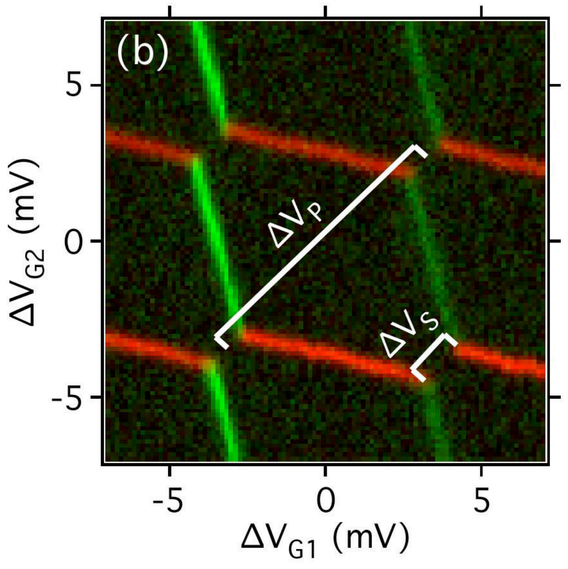


Figure 4.9 (a) Explanation of the “color coding” (green = Dot 1, red = Dot 2) of the Coulomb blockade peaks making up the unit hexagon. (*M*, *N*) labels the charge states of Dot 1 and Dot 2 respectively. Along side (i), the degeneracy is between states (*m*, *n*) and (*m*, *n*+1) so the number of electrons on Dot 2 can change between *n* and *n*+1. Current therefore flows through Dot 2 and that side is red. Along side (ii), the degeneracy is between (*m*, *n*) and (*m*–1, *n*); current flows through Dot 1 and that side is green. Along side (iii), the degeneracy is between (*m*, *n*) and (*m*+1, *n*–1). This means that that charge states of Dots 1 and 2 must change *simultaneously*. The probability of this occurring is smaller so the Coulomb blockade peak there is suppressed. (b) Peak splitting ΔV_S and period ΔV_P of Coulomb blockade data from the strongly capacitively-coupled dots. Their ratio is related to the fractional capacitive coupling α via Eq. (4.4).

double dot along side (iii) is significantly weaker than either sides (i) or (ii). Interestingly, side (iii) is a potential source of time-correlated electrons, and further investigation into its properties should be undertaken in a future work.

The second thing worth noting about the hexagonal pattern is that the fractional peak splitting $f \equiv 2\Delta V_S/\Delta V_P$, where ΔV_S is the peak splitting and ΔV_P the period of the hexagons as defined in Figure 4.9(b), is related to the fractional capacitive coupling $\alpha \equiv C_{INT}/C_\Sigma$. By solving for the locations where the distorted paraboloids in the capacitive charging model [Equation (4.3a)] intersect, we find that the fractional capacitive coupling is related to the fractional peak splitting as[1]

$$\alpha \equiv \frac{C_{INT}}{C_\Sigma} = \frac{f}{2-f} \tag{4.4}$$

From the measured fractional peak splitting $f = 0.34$ in Figure 4.9(b), a fractional capacitive coupling $\alpha = 0.20$ was obtained, meaning that one fifth of the electric field lines from one dot connect to the other dot. This is an order of magnitude larger than the 3 to 5% reported for similar lateral double dots [Livermore, 1998; Duncan *et al.*, 1999], showing that the coupling gate increases capacitive coupling, as designed. The coupling energy $\alpha E_C = 76\ \mu$eV is also correspondingly larger and significantly greater than the thermal energy $kT = 6\ \mu$eV of the experiment, as evidenced by the very clear peak splitting. This will allow the dots to be operated as a single-electron switch, as will be described in Section 4.6.

It is interesting to compare the similarities and differences between capacitive- and tunnel-coupling between quantum dots. Figure 4.10 shows the Coulomb blockade peaks of (a) the strongly capacitively-coupled parallel dots described in this chapter, and (b) the tunnel-coupled series dots of Livermore *et al.* [1996]. The two patterns are strikingly similar, and arise from similar energetics but from different mechanisms. As explained in the preceding paragraphs, the hexagonal Coulomb blockade pattern arises in capacitively-coupled dots from the change in energy of charge-polarized configurations

[1] This differs from $f = \alpha$ in Livermore *et al.* [1999], which is correct only at the endpoints $f = 0$ and $f = 1$.

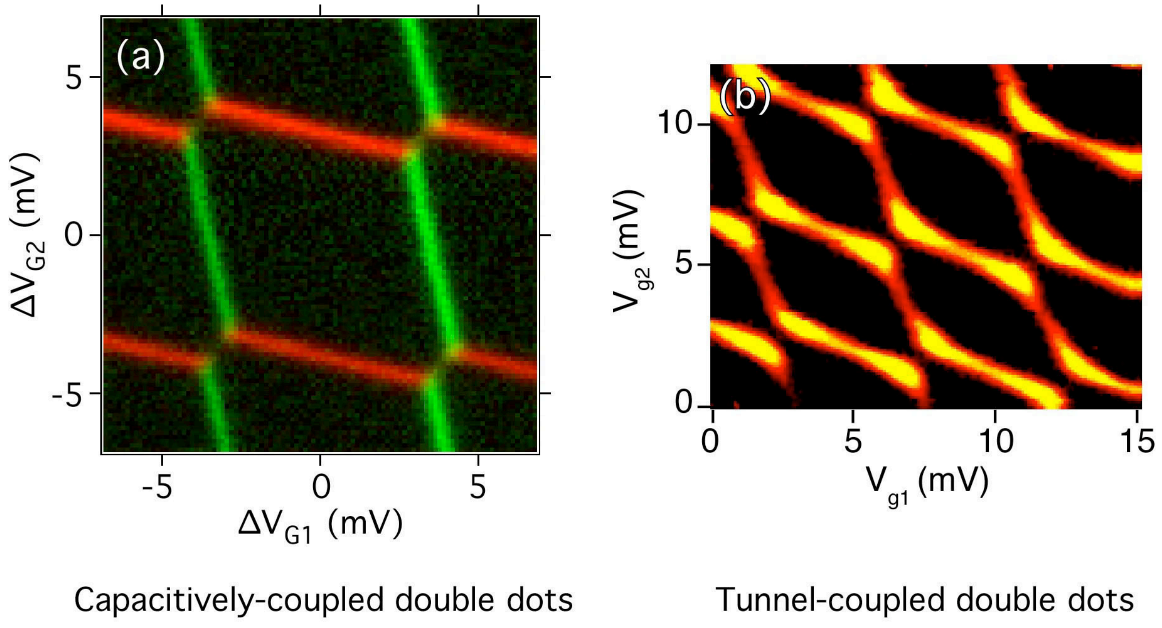


Figure 4.10 Coulomb blockade peaks from (a) capacitively-coupled double dots described in this thesis and (b) tunnel-coupled double dots [Livermore *et al.*, 1996]. The hexagonal pattern of peaks in both cases is due to the change in energy of polarized charge configurations, which can be interpreted as a bonding between the quantum dots. The double dots are bound by “ionic bonds” in the capacitively-coupled case, and “covalent bonds” in the tunnel-coupled case.

of the double dot. A similar change in the energy of charge-polarized states occurs in tunnel-coupled dots because of electron-sharing between the dots [Golden and Halperin, 1996a and 1996b]. This change in the energy of polarized states can also be interpreted as the formation of “bonding” (and “anti-bonding”) states between quantum dots to form “artificial molecules.” In the capacitively-coupled case, it is analogous to forming an ionic bond, while in the tunnel-coupled case, it is analogous to forming a covalent bond.

4.6 Operation as a Single-Electron Switch

The strong capacitive-coupling between dots, such that $\alpha E_C >> kT$, allows the change in charge state of one dot to shift the conductance of the other dot completely on or off a Coulomb blockade peak, thus changing its conductance in a non-linear way and making a switch. This is illustrated in Figure 4.11, where the charge state of Dot 1 is changed along gate voltage ΔV_{G12}, and as a result of this the conductance of Dot 2 is switched on and off. Now, gate voltage ΔV_{G12} is a special combination of sidegate voltages ΔV_{G1} and ΔV_{G2} such that ΔV_{G12} moves parallel to the Coulomb blockade peaks of Dot 2, as shown by the white line in Figure 4.11(a). This means that ΔV_{G12} does not induce charge in Dot 2, but only induces charge on Dot 1, as shown in Figure 4.11(b). It is in response to this change in the charge state of Dot 1, and only this, that the conductance of Dot 2 is switched on and off [Figure 4.11(c)]. The average conductance of Dot 2 between the “on” and “off” states differs by two orders of magnitude, and is easily detectable. Because of the symmetry between Dots 1 and 2, the roles of Dot 1 as the “trigger” dot and Dot 2 as the “switched” dot can be reversed simply by moving in a combination of sidegate voltages parallel to the Coulomb blockade peaks of Dot 1 instead of Dot 2.

The lineshapes of the trigger dot and switched dot were fit to theoretical lineshapes [shown as dashed lines in Figures 4.11(b) and (c)] and good agreements were

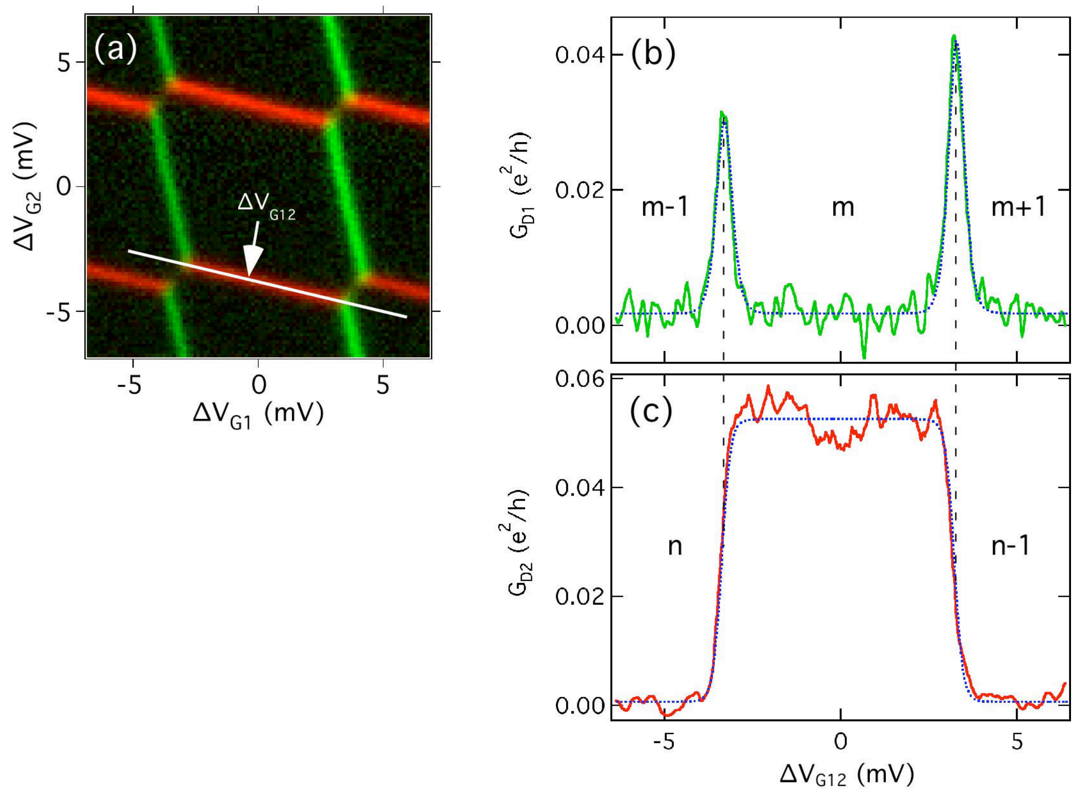


Figure 4.11 Procedure for operating the coupled double dots as a single-electron switch. (a) White line shows the location of ΔV_{G12}, which is a combination of sidegate voltages ΔV_{G1} and ΔV_{G2}. ΔV_{G12} is parallel to the Coulomb blockade peaks of Dot 2 and does not induce any charge onto Dot 2. (b) However, ΔV_{G12} does induce electrons onto Dot 1. (c) A single electron entering Dot 1 triggers Dot 2 to turn "on", and a second electron entering Dot 1 triggers Dot 2 to turn "off." Dashed blue lines are fits to the lineshapes of Dot 1 (the "trigger" dot) and Dot 2 (the "switched" dot), and good agreements were found (see text).

found. The lineshape of the trigger dot [Figure 4.11(b)] was fit to two thermally smeared Coulomb blockade peaks:

$$\begin{aligned} G_{\Theta-\mathrm{tr}}(\Delta V_{\mathrm{G12}}) = G_0 &+ \delta G_{01}\cosh^{-2}\left[\Theta_0(\Delta V_{\mathrm{G12}} - \Delta V_{\mathrm{G1201}})\right] \\ &+ \delta G_{02}\cosh^{-2}\left[\Theta_0(\Delta V_{\mathrm{G12}} - \Delta V_{\mathrm{G1202}})\right] \end{aligned} \tag{4.5}$$

where G_0 is the conductance offset, Θ_0 is the scale factor, δG_{01} and δG_{02} are the heights of the two peaks, and $\Delta V_{\mathrm{G1201}}$ and $\Delta V_{\mathrm{G1202}}$ are the locations of the two peaks. Good agreement was found as the derivative of the Fermi function is the expected Coulomb blockade lineshape for dots in the thermally smeared regime [Equation (2.1)]. The lineshape of the switched dot [Figure 4.11(c)] was empirically fit to two thermally smeared step-functions (*i.e.*, two Fermi functions):

$$\begin{aligned} G_{\Theta-\mathrm{sw}}(\Delta V_{\mathrm{G12}}) = G_1 + \tfrac{\delta G_1}{2}\Big\{&\tanh\left[\Theta_1(\Delta V_{\mathrm{G12}} - \Delta V_{\mathrm{G1211}})\right] \\ &- \tanh\left[\Theta_1(\Delta V_{\mathrm{G12}} - \Delta V_{\mathrm{G1212}})\right]\Big\} \end{aligned} \tag{4.6}$$

where G_1 is the conductance offset, Θ_1 is the scale factor, δG_1 is the height of the step-functions, and $\Delta V_{\mathrm{G1211}}$ and $\Delta V_{\mathrm{G1212}}$ are the locations of the two steps. A good fit was found for the switched dot as well.

4.7 Switching Lineshapes

Now, the abruptness of the conductance switching in gate voltage should depend on how well charge is quantized in the trigger dot. If charge on the trigger dot changed slowly with gate voltage (due to poor charge quantization), then the switched dot should be switched slowly in gate voltage as well. To verify this, Dot 1 was placed in the poorly charge-quantized regime (lifetime-broadened regime) by opening the tunnel barriers to its leads so that $h/\Delta t >> kT$, while Dot 2 was kept in the well charge-quantized regime (thermally-smeared regime). The conductances of both dots were measured simultaneously around a peak splitting and are shown in Figures 4.12(a) and (c). The Coulomb blockade peaks of Dot 1 [Figure 4.12(a)] are noticeably wider than those of Dot 2 [Figure 4.12(c)], and are evidence of the poorly quantized charge in Dot 1.

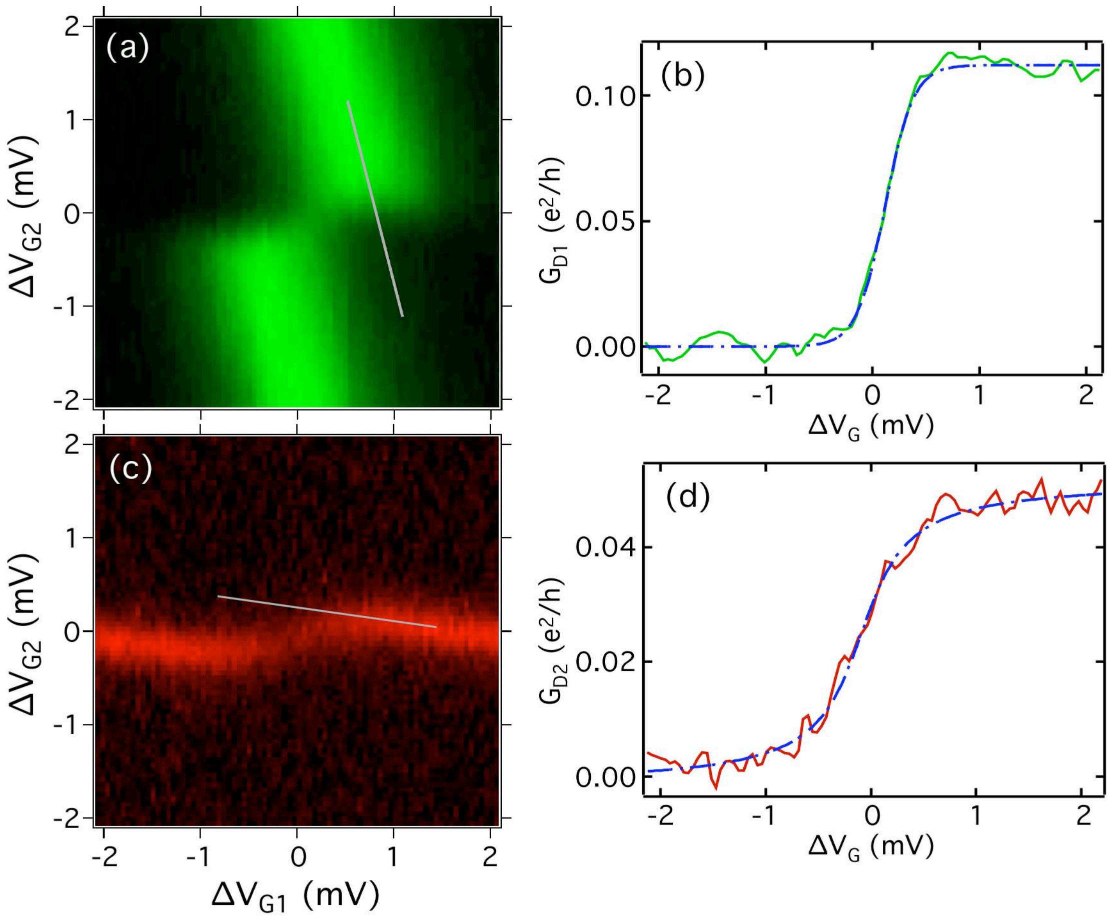


Figure 4.12 Coulomb blockade peaks of (a) Dot 1 and (c) Dot 2 at the peak splitting. For these measurements, Dot 1 was strongly tunnel-coupled to its leads (lifetime-broadened regime), while Dot 2 was poorly tunnel-coupled to its leads (thermally-smeared regime). (a) The broad peaks of Dot 1 are clearly split by the well-quantized charge states of Dot 2. (c) However, the narrow peaks of Dot 2 are poorly split by the poorly-quantized charge states of Dot 1. This shows that the peak splitting of one dot is determined by the charge quantization of the other dot. The switching lineshape of Dot 1 switched by Dot 2, indicated by the white line in (a), is shown in (b), and the switching lineshape of Dot 2 switched by Dot 1, indicated by the white line in (c), is shown in (d). Fits to the switching lineshapes in (b) and (d) were made (dashed blue lines), and good agreements were found (see text).

However, the peak splitting (and thus switching) of Dot 1 is noticeably more abrupt than that of Dot 2, and shows that it is the well-quantized charge in Dot 2 that switches the conductance of Dot 1. Conversely, the smeared peak splitting of Dot 2 confirms that it is the poorly-quantized charge in Dot 1 that switches the conductance of Dot 2.

The switching lineshape of Dot 1 switched by Dot 2 is indicated by the white line in Figure 4.12(a) and is shown in Figure 4.12(b). A single thermally-smeared step-function [similar to Equation (4.6)] was used to empirically fit the lineshape (dashed line) and good agreement was found. The switching lineshape of Dot 2 switched by Dot 1 is indicated by the white line in Figure 4.12(c) and is shown in Figure 4.12(d). This time, a single lifetime-broadened step-function was used to empirically fit the data:

$$G_{\Gamma-\mathrm{sw}}(\Delta V_{\mathrm{G12}}) = G_2 + \frac{\delta G_2}{\pi}\left\{\arctan\left[\Gamma_2(\Delta V_{\mathrm{G12}} - \Delta V_{\mathrm{G1221}})\right] + \pi/2\right\} \tag{4.7}$$

where G_2 is the conductance offset, Γ_2 is the scale factor, δG_2 is the height of the step-function, and $\Delta V_{\mathrm{G1221}}$ is the location of the step. The arctan function is the step-function counterpart to a lifetime-broadened delta-function (*i.e.* a Lorentzian), and was obtained by integrating a Lorentzian.

4.8 Conclusions

In summary, it is possible to increase capacitive-coupling between lateral quantum dots by an order of magnitude with a coupling gate. The capacitively-coupled dots have a hexagonal Coulomb blockade pattern that can be explained within the capacitive charging model as due to the electrostatic interaction between excess charges in the double dot. The strong coupling ($\alpha E_C >> kT$) achieved in our device allows the charge state of one dot to strongly influence the behavior of the other dot, and a single-electron switch can be made. The abruptness of the conductance switching in gate voltage is found to be determined by how well charge is quantized in the trigger dot.

Chapter 5: Few-Electron Quantum Dots

This chapter looks into using few-electron quantum dots for building quantum computers, as proposed by Loss and DiVincenzo [1998]. One key experimental hurdle that must be overcome is storing just one electron in each quantum dot. In this chapter, I present data that demonstrates that it is possible to store just a single electron in each of the two few-electron dots by first emptying the dot of electrons to define the absolute number of electrons in the dot. The *g*-factor of electron spins was also measured and found to be no different from that in bulk GaAs, and tunnel-coupling between two few-electron dots was demonstrated. Section 5.1 gives an introduction to quantum computing and the Loss-DiVincenzo proposal. Section 5.2 describes the few-electron double dot device and how it was fabricated. Section 5.3 covers the experimental setup for the measurements that were made, and Section 5.4 explains how the finite bias Coulomb blockade data (Section 5.4.1) and the three-dimensional gate voltage data (Section 5.4.2) signal that the dots can be emptied of electrons. The Zeeman energies of several electronic states were measured, as described in Section 5.5, and tunnel coupling between dots was demonstrated, as covered in Section 5.6. A hexagonal pattern is seen, like in Chapter 4.5, but with only a few electrons in each dot. The conclusions of this chapter are presented in Section 5.7.

5.1 Introduction

In the past few years, interest in quantum computation has really soared and is attributable to several important advances that have occurred. The discovery that it is possible to implement certain calculations exponentially faster on a quantum computer than on a classical computer (*e.g.* prime factorization) provided theoretical motivation [Shor, 1997; Grover, 1997]. Demonstrations that quantum computation is experimentally feasible [*e.g.* nuclear magnetic resonance (NMR) experiments] provided experimental motivation [Chuang *et al.*, 1998; Vandersypen *et al.*, 2001]. Unfortunately, the technology used in these NMR demonstrations is not scalable and the quantum computer roadmap seemed to hit a roadblock. However, new proposals for building scalable quantum computers were put forward, and provided several promising avenues of research in the implementation of large-scale, commercially-significant quantum computers. These include using the spins of electrons confined in quantum dots [Loss and DiVincenzo, 1998], the nuclear spins of phosphor atoms embedded in crystalline silicon [Kane, 1998], the electronic states of ions in optical traps [Cirac and Zoller, 1999], and persistent currents in superconducting rings [Mooij *et al.*, 1999]. The experimental realization of a quantum computer is extremely challenging, and the first steps are just being taken.

Of particular relevance to the Westervelt lab's expertise is the Loss-DiVincenzo scheme [Loss and DiVincenzo, 1998] shown in Figure 5.1. The quantum bit (qubit) of information would be encoded in the spin of an individual electron that is physically contained inside a small quantum dot. Quantum computation would then be performed by sequences of spin-flip and "square-root of swap" (swap$^{1/2}$) operations on the spins. These operations, which can be classified into single-qubit (spin-flip) and two-qubit (swap$^{1/2}$) operations, were shown to be the minimal set needed to construct a universal quantum computer [DiVincenzo, 1995]. The spin-flip operation would be carried out using electron-spin resonance (ESR) to flip the spin of individual electrons, while the swap$^{1/2}$ operation would entangle a pair of spins by controlling the tunneling between the quantum dots. Spin-selective filters [Folk *et al.*, 2003] may then be used to readout the results of the calculation.

An effort to realize a scalable Loss-DiVincenzo quantum computer has been taken up with funding from the Defense Advanced Research Projects Agency (DARPA) and the Nanoscale Science and Engineering Center (NSEC) by an international collaboration that includes the Westervelt group. To that end, my work has been to study small quantum dots containing just a few electrons, to explore their spin properties, and demonstrate tunnel coupling between the dots.

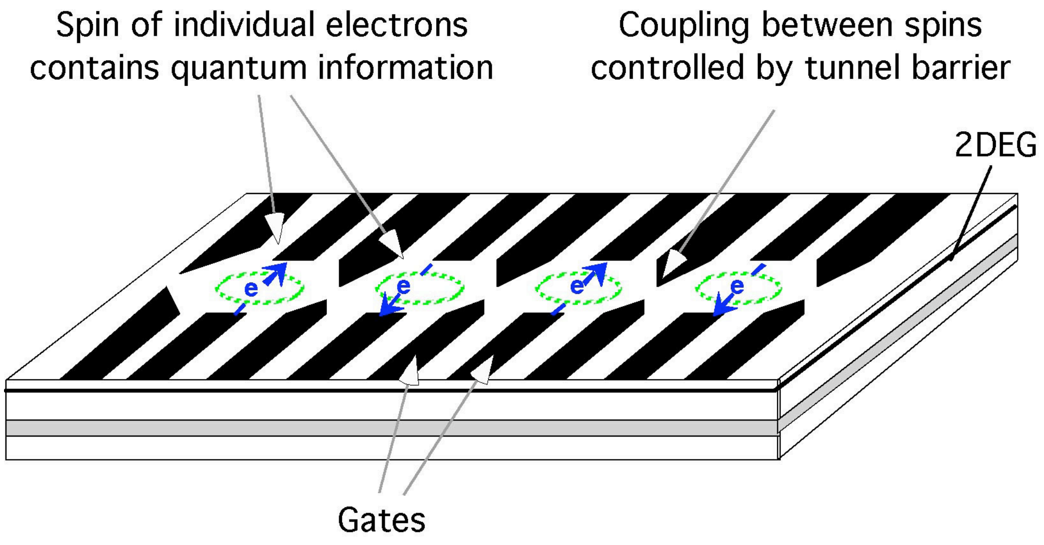

Spin of individual electrons contains quantum information
Coupling between spins controlled by tunnel barrier
2DEG
e
e
e
e
Gates

Figure 5.1 Schematic showing the Loss-DiVincenzo quantum computer proposal. Each quantum dot holds a single electron, and the spin of each electron stores a quantum bit of information. Computation is performed by a combination of flipping individual spins and entangling adjacent spins. Spin-flips are implemented by electron-spin resonance, and entanglement is performed by tunnel-coupling adjacent dots together (swap$^{1/2}$ operation).

5.2 Few-Electron Double Dot Device

The SEM photograph in Figure 5.2 shows the few-electron double dot used in these experiments. The light gray areas are surface metal gates fabricated using electron-beam lithography that define the dots. The right-hand dot is labeled Dot 1, and the left-hand dot Dot 2. The few-electron dots were designed such that the tunnel-barriers to the leads were spaced as closely as possible [after Gould *et al.*, 1999]. This was done to maximize the probability that both leads are connected to the same puddle of electrons inside the dot, as the electrostatic potential in small quantum dots is very rough. The mental picture that helps explain why this is so is that of a pond. When a pond (dot) is filled with water (electrons), the pond is a contiguous mass of water. However, when the pond is drained and close to being empty, isolated puddles of water start to form within the pond due to the unevenness of the bottom of the pond (roughness of the potential inside the quantum dot). Having electrical leads close to the same part of a dot therefore increases the chances that both are connected to the same puddle of electrons.

The device was fabricated in just one layer of gate metal on a square-well 2DEG wafer called “MH5” by the Westervelt group, or “020313B” by the Gossard group that grew it at U. C. Santa Barbara. The wafer was grown by Micah Hanson and consists of the following layers:

50 Å GaAs (surface)
250 Å AlGaAs
Si δ-doping
220 Å AlGaAs
200 Å GaAs
1000 Å AlGaAs
20-period smoothing superlatice:
  25 Å AlGaAs
  25 Å GaAs
3000 Å GaAs buffer
GaAs substrate

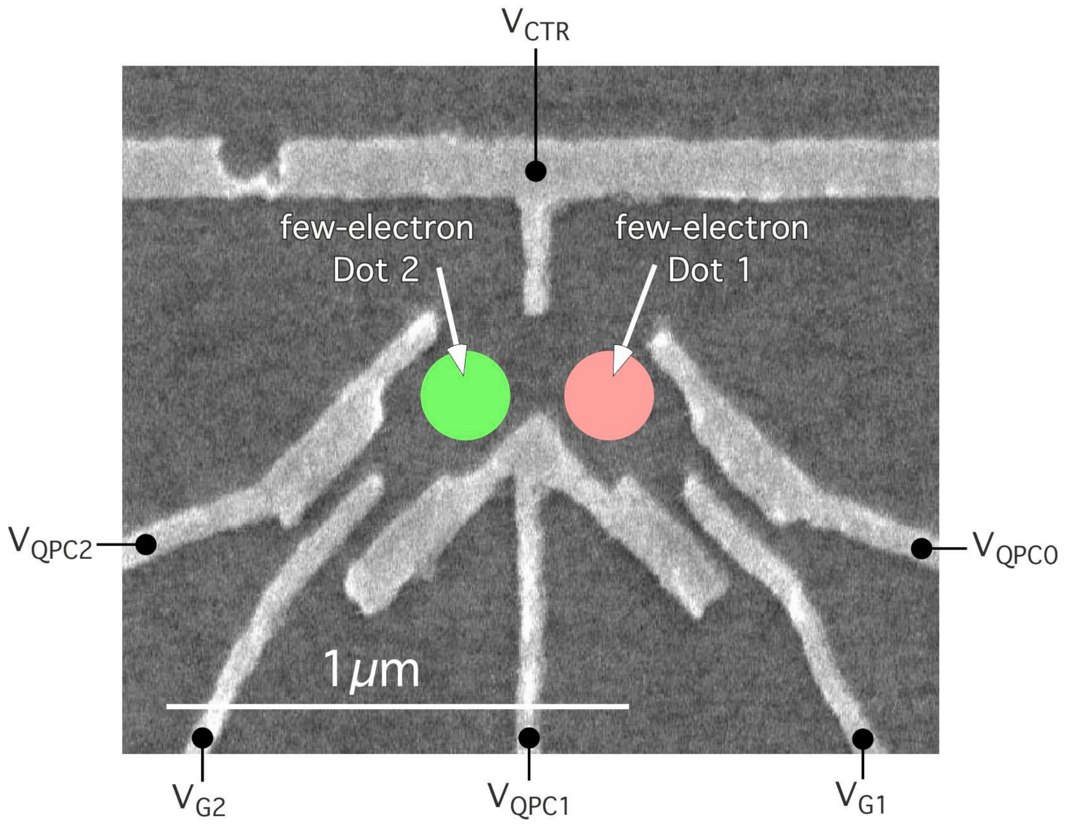


Figure 5.2 SEM photograph of the few-electron double dot device. The light areas are metal gates used to define the quantum dots. The locations of the dots are highlighted by circles – red for Dot 1 and green for Dot 2. The gate voltage applied to each gate is labeled by the black text.

The 2DEG is located 52 nm below the surface of the wafer, and was measured by Micah Hanson to have a mobility μ = 300,000 cm^2/Vs and a density $N = 4\times10^{11}$ cm^{-2} at 19 K. A square-well 2DEG was used to get better confinement of the electrons in the z-dimension. It is speculated that this makes a better few-electron quantum dot because it might be easier to electrostatically "squeeze" the electrons out of the dot using the gates that define the dot. The sample is called "EL2 MH5.2" as it was the second sample made with the MH5 wafer using the second-generation electrometer design (EL2). The lithographic size of the dot is nominally 250×260 nm^2. The number of electrons in the dot is estimated to be ~100 after accounting for a 50 nm depletion region around the dot, but there are in reality far fewer electrons in the dot.

5.3 Experimental Setup

The few-electron dot device was measured in a Kelvinox 100 dilution fridge at a nominal base temperature of 70 mK (kT = 6 μeV). The measurement setup is shown in Figure 5.3. The two few-electron dots were in series, so one lock-in amplifier and current preamplifier measured the conductance of either or both dots. Gate voltages were supplied by the BiasDac box (see Chapter 3.2.3), and finite dc bias Coulomb blockade measurements were made by summing a BiasDac output with the 10 μV_{rms} ac lock-in excitation (see Chapter 3.2.5). Magnetic field was supplied by two Kepco model BOP 20–20M bipolar power supplies (BOPs) in parallel, which energized the 7 T superconducting magnet inside the Kelvinox 100 dewar. The BOPs were placed in current-output mode and their outputs controlled by a BiasDac voltage.

5.4 Emptying Dots of Electrons

In order to know the absolute number of electrons in a quantum dot, the dot must first be emptied of electrons to define zero. The Coulomb blockade effect can then be used to fill the dot back up one electron at a time to the desired number.

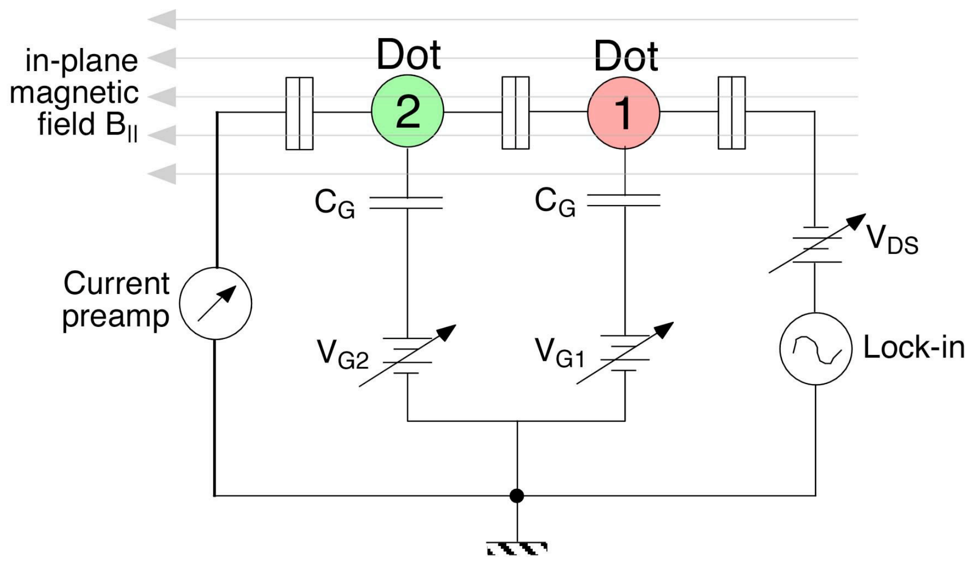


Figure 5.3 Measurement setup for the few-electron quantum dots. The conductance of Dot 1, Dot 2, or both in series was measured using conventional lock-in techniques. A drain-source DC bias voltage could be applied across the dot(s) for finite bias Coulomb blockade measurements. An in-plane magnetic field of up to 7 T could also be applied. Not shown are voltages for controlling tunnel barrier heights of the dots.

5.4.1 Single-Dot Finite Bias Coulomb Blockade Measurements

The first method to empty the dots is to make finite bias Coulomb blockade measurements on each dot. When the Coulomb blockade diamond is observed not to close, it indicates that there are no more electrons left in the dot, and this is shown in Figure 5.4 for each dot measured separately. Figure 5.4(a) shows the differential conductance dG_{D1}/dV_{DS} of Dot 1 as a function of its sidegate voltage V_{G1} and the dc source-drain bias V_{DS}, where G_{D1} is the conductance of Dot 1. Figure 5.4(b) shows the corresponding differential conductance dG_{D2}/dV_{DS} of Dot 2 as a function of its sidegate voltage V_{G2} and the dc source-drain bias V_{DS}, where G_{D2} is the conductance of Dot 2. The Coulomb blockade diamonds for both dots are visible as black diamonds, and the additional lines parallel to the sides of the diamonds are Coulomb blockade peaks from excited states of the dots. As the sidegate voltages of both dots are made more negative, the last diamond for both dots were not observed to close. Due to the low conductance of both dots at highly negative sidegate voltages, Coulomb blockade peaks were extrapolated from large drain-source biases (where they are visible) to show the boundaries of the final Coulomb blockade diamond (dashed lines). No evidence of other Coulomb blockade peaks was seen at drain-source biases out to 9 mV for Dot 1 and 7 mV for Dot 2.

With the dots known to be empty, the absolute number of electrons in each dot can be labeled as indicated by the colored numbers in Figure 5.4. This therefore enables a single electron to be stored in each dot, as required by the Loss-DiVincenzo proposal. The data also show that Dot 1 can contain up to five electrons while Dot 2 can hold up to three electrons, so these dots are very small indeed. In comparison, the 500×800 nm^2 dots of Chapter 4 were estimated to hold 800 electrons each. Each few-electron dot holds far fewer electrons than estimated (~5 versus ~100), even after taking into account a nominal 50 nm depletion boundary around the dots. This is most likely because the potential in a small dot is shallower than that of a larger dot, in addition to being smaller in the lateral

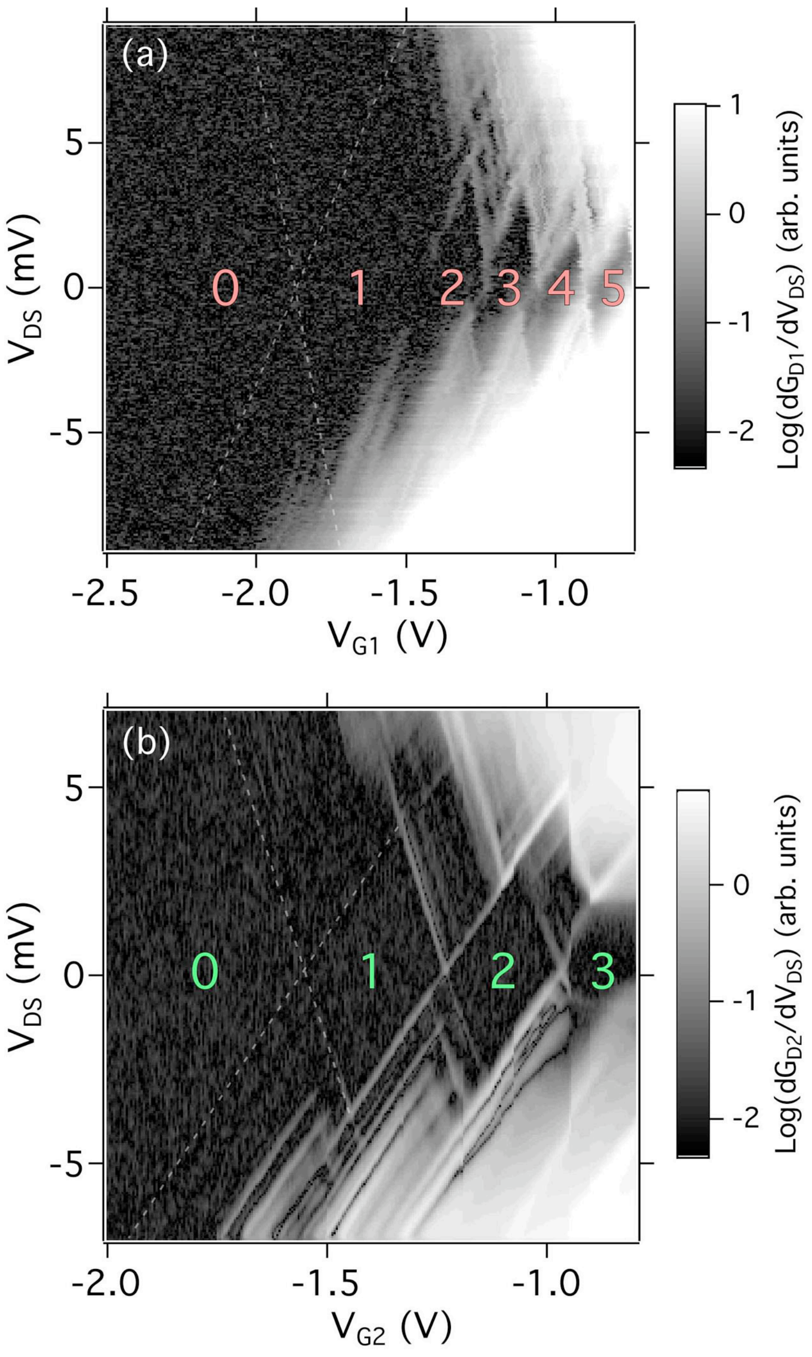


Figure 5.4 Finite bias Coulomb blockade measurements of (a) Dot 1 and (b) Dot 2, acquired separately. The left-most Coulomb blockade diamonds do not close, indicating that no more electrons are to be found inside the dots. This allows the number of electrons in each dot to be labeled as indicated. The dashed lines are extrapolations of Coulomb blockade peaks that are visible at high drain-source voltages.

dimensions. This is good news for experimentalists, as the dimensions of few-electron quantum dots are larger than anticipated and therefore easier to make. Dots smaller than these (160×200 nm^2) were initially made, but were found to be devoid of electrons.

5.4.2 Three-Dimensional Scan of a Few-Electron Dot

A drawback of using the finite bias Coulomb blockade technique for laterally-defined dots is that stray capacitance couples the sidegate voltage to the tunnel barriers of the dot and causes them to get progressively get more opaque. This problem of diminishing conductance signal can be overcome by re-tuning the QPC voltages (which control tunnel barrier heights) every time the sidegate voltage is changed, and is described here. The resulting three-dimensional scan in the sidegate and QPC voltages (which are all the gate voltages that may be adjusted) also shows that the Coulomb blockade peaks end and therefore that our dots can be emptied of electrons.

Figure 5.5 shows the three-dimensional conductance data of Dot 2 as a series of two-dimensional scans in QPC gate voltages V_{QPC1} and V_{QPC2} as sidegate voltage V_{G2} goes from (a) –0.500 V to (f) –1.625 V in steps of –0.225 V. Each two-dimensional scan is a re-tuning of the QPC gate voltages (see Chapter 3.3.5) to Dot 2, and maximizes the Coulomb blockade peak height of the Coulomb blockade peaks within the scan.

As sidegate voltage V_{G2} is made more negative in Figures 5.5 (a) to (f), the Coulomb blockade peaks are shifted toward the top right as it progressively takes less QPC voltage to induce the same number of electrons onto Dot 2. However, no new Coulomb blockade peaks are revealed in the sequence of Figures, leading to the conclusion that Dot 2 is empty past the last Coulomb blockade peak (toward the bottom left corner of each Figure). The number of electrons in Dot 2 may therefore be labeled as indicated, and shows that Dot 2 can contain up to three electrons before it merges with its leads. By choosing an appropriate combination of QPC and sidegate voltages, the number of electrons in Dot 2 may therefore be selected.

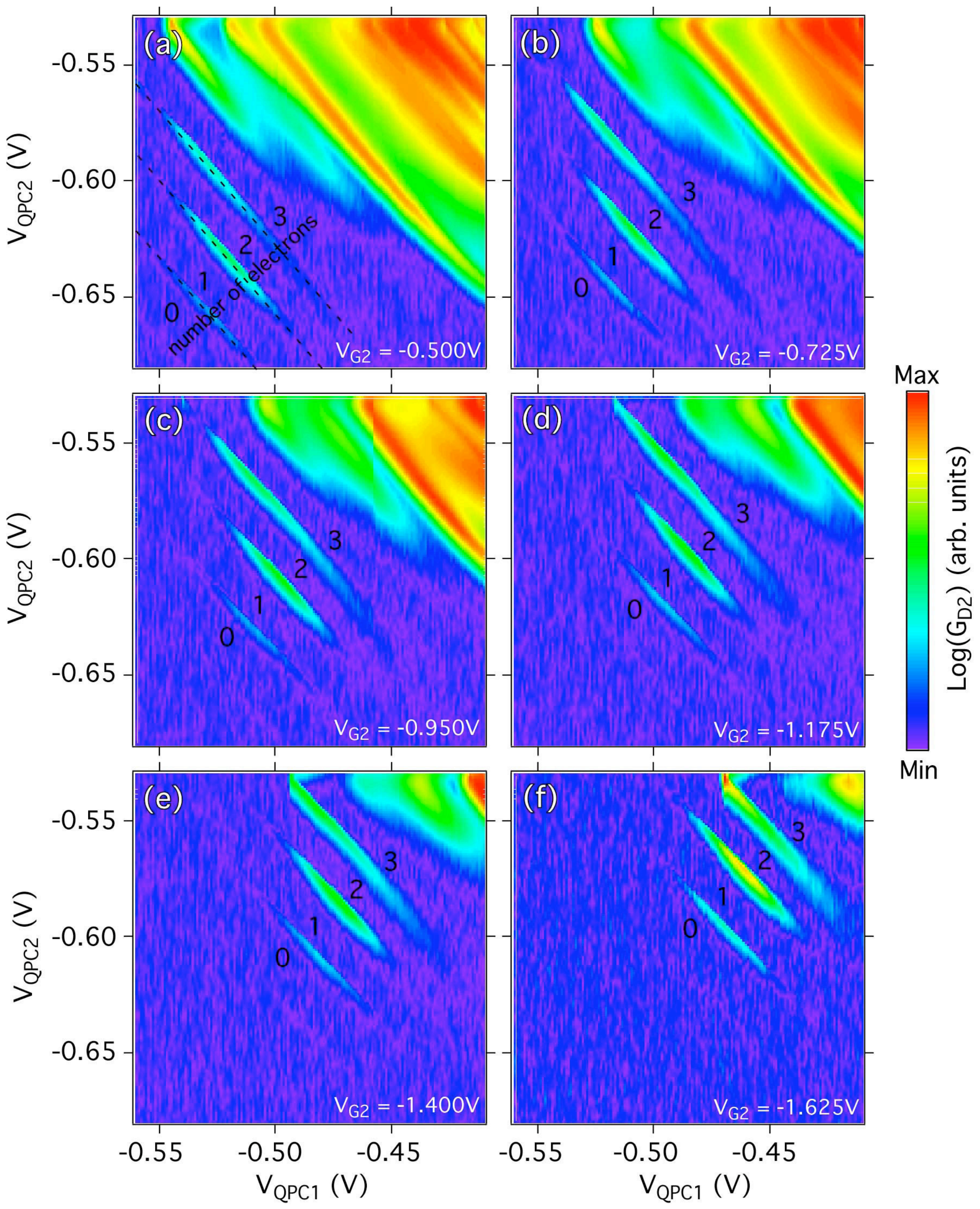


Figure 5.5 Conductance G_{D2} of Dot 2 as a function of QPC voltages V_{QPC1} and V_{QPC2}, and sidegate voltage V_{G2}. Each scan (a) to (f) in QPC voltages is a re-tuning of the tunnel barriers to Dot 2 at a different sidegate voltage. This re-tuning maximizes the Coulomb blockade peak heights for greatest peak visibility. As sidegate voltage V_{G2} is decreased from (a) –0.500 V to (f) –1.625 V, the Coulomb blockade peaks are shifted toward the top right. However, no new Coulomb blockade peaks are revealed from the bottom left, indicating that there are no more electrons left in the dot. The number of electrons in Dot 2 may thus be labeled as indicated, and the number of electrons in the dot may be set by an appropriate choice of gate voltages.

5.5 Zeeman Energy of Few-Electron States

It is also important to learn about the properties of electron spins for implementing the spin-flip operation with ESR. Given the isolated nature of electrons in few-electron quantum dots compared with electrons in bulk semiconductor, it is not obvious that the spin of these electrons will behave like those of electrons in a bulk semiconductor. In order to determine the *g*-factor of electrons inside a few-electron dot, the Zeeman energy of several electronic states was measured.

The Zeeman energy of an electron may be measured as a component of the electron's addition energy E_A. The addition energy E_A of electrons in an in-plane magnetic field is the sum of the charging energy E_C, the level spacing $\Delta\varepsilon$, and the Zeeman energy E_Z:

$$E_A = E_C + \Delta\varepsilon + E_Z, \tag{5.1}$$

where $E_Z \equiv \pm g\mu B_{\parallel}$, and g is the g-factor, μ is the Bohr magneton, and $B_{\parallel}$ is the in-plane magnetic field. The in-plane magnetic field should not affect the orbital motion of electrons in the 2DEG due to the thinness (20 nm) of the electrons in the z-dimension, so the charging energy E_C and level spacing $\Delta\varepsilon$ should not change with in-plane magnetic field. The only component of the addition energy that depends on magnetic field is therefore the Zeeman energy E_Z. This means that the sidegate voltage separating Coulomb blockade peaks ΔV_{G2} (the peak spacing) is related to the Zeeman energy as (see Chapter 2.4.1):

$$e\Delta V_{G2}\, C_G/C_\Sigma = E_A = \pm g\mu B_{\parallel} + \textit{constant in } B_{\parallel}, \tag{5.2}$$

where C_G is the sidegate–dot capacitance and C_Σ is the dot self-capacitance. One can therefore measure the conductance of a dot as a function of in-plane magnetic field and sidegate voltage to obtain the Zeeman energy, and thus the *g*-factor.

This is shown in Figure 5.6(a), where the conductance G_{D2} of Dot 2 was measured as a function of in-plane magnetic field $B_{\parallel}$ and sidegate voltage V_{G2}. Four Coulomb

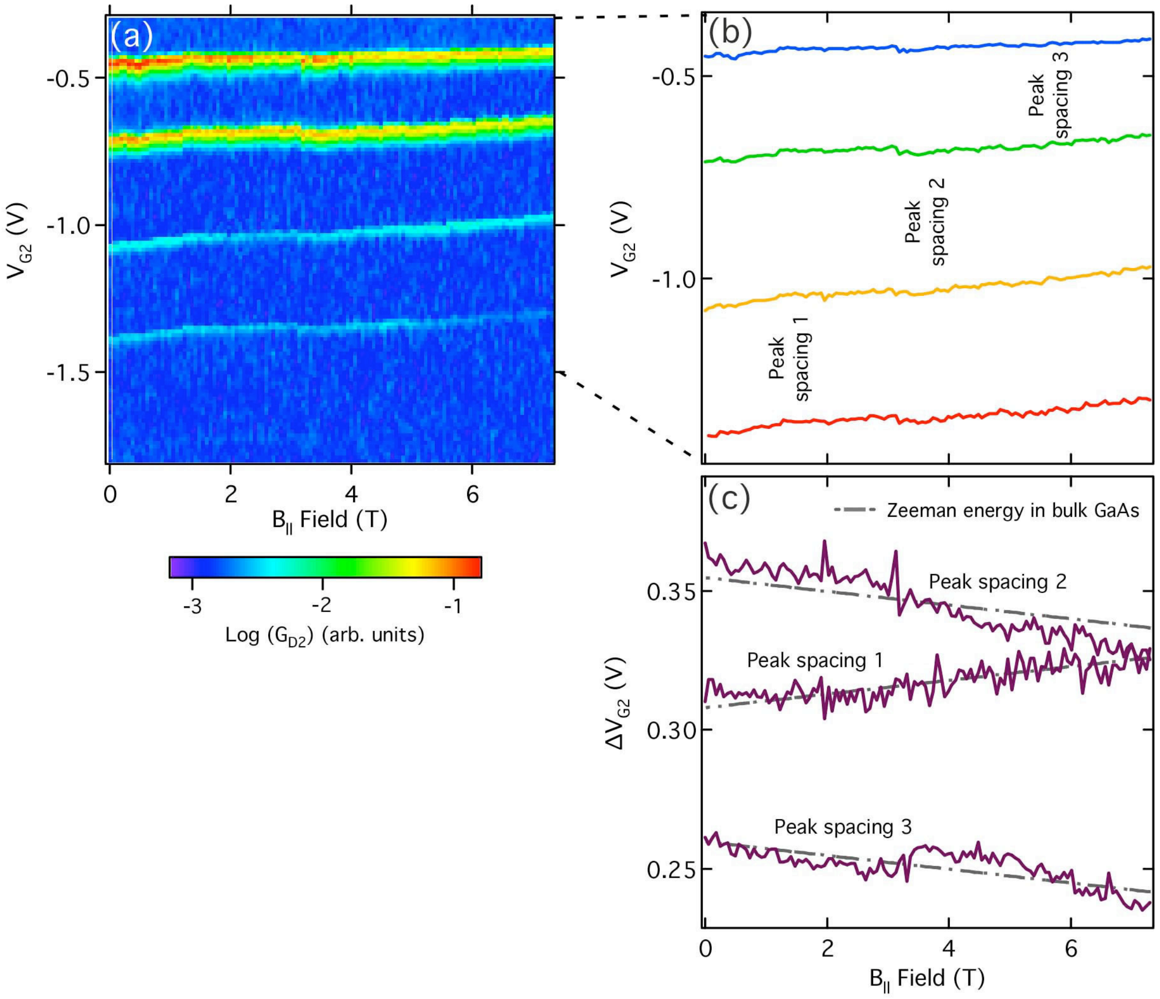


Figure 5.6 Extraction of Zeeman energy data from Coulomb blockade peak spacings. (a) Conductance of Dot 2 as a function of in-plane magnetic field and sidegate voltage. (b) Peak location of the four Coulomb blockade peaks, which was obtained by curve-fitting the data in (a). (c) Peak spacings of the Coulomb blockade peaks as a function of in-plane magnetic field. The Zeeman energy in bulk GaAs (dashed lines) assume a *g*-factor $|g| = 0.44$, and were calculated using Equation (5.2).

blockade peaks are clearly visible in the Figure; a fifth peak is very low in amplitude, and no useable data was extracted from it. The positions of these four peaks in sidegate voltage V_{G2} were computed by curve-fitting each peak to Equation (2.1) to obtain the data shown in Figure 5.6(b). The peak spacings ΔV_{G2} between the Coulomb blockade peaks are shown in Figure 5.6(c), and are observed to slope with magnetic field. The Zeeman energy traces for $|g| = 0.44$ (expected for electrons in bulk GaAs) are shown by the dashed lines, and were computed via Equation (5.2). A "lever-arm" ratio of capacitances $C_G/C_\Sigma = 1/90$ was used in the calculation, which was obtained from the ratio of the half-height to the full-width of the Coulomb blockade diamond of Dot 2 [Figure 5.4(b)] (see Chapter 2.4.2). Figure 5.6(c) shows that the measured and the expected Zeeman energies are in agreement, and therefore that the *g*-factor of electrons in a few-electron dot is consistent with the *g*-factor $|g| = 0.44$ of electrons in bulk GaAs.

Recent data from the Kouwenhoven group at Delft [Hanson *et al.*, 2003] also finds that the *g*-factor in few-electron dots is similar to that in bulk GaAs at fields below about 7 T, but indicates that there is a saturation in the Zeeman energy at higher fields (up to the measured 15 T). Further work needs to be done to clarify whether this is a genuine change in the electronic *g*-factor, as they claim, or the result of orbital effects from the strong in-plane magnetic field [Zumbuhl *et al.*, 2002]. In any case, in magnetic fields below about 7 T, there appears to be little reason to doubt that the *g*-factor of electrons, even in very small dots, is identical to that of electrons in bulk GaAs.

5.6 Tunnel-Coupled Few-Electron Dots

An important aspect of the Loss-DiVincenzo proposal [Loss and DiVincenzo, 1998] is the use of tunnel coupling between quantum dots to entangle electron spins (swap$^{1/2}$ operation). In this operation, both the strength and the duration of the coupling must be controlled. In this section, I demonstrate that tunnel-coupling between dots can be achieved, and deduce the strength of the tunneling from the Coulomb blockade peak splitting.

Figure 5.7 shows the conductance G_{DD} of Dot 1 and Dot 2 in series as a function of the two sidegate voltages V_{G1} and V_{G2}. The Coulomb blockade peaks of the series double dots are clearly observed to be split, and the hexagonal charge-stability pattern characteristic of coupled double dots is highlighted as shown. By measuring the fractional peak splitting f (defined in Chapter 4.4), the actual tunnel coupling between the dots in Figure 5.7 can be estimated. Following Livermore *et al*. [1996] and interpolating between the two theoretical formulas appropriate for weak and strong tunnel-coupling [Golden and Halperin, 1996a and 1996b], an interdot conductance of 1.2 e^2/h is estimated from the measured fractional peak splitting $f = 0.3$.

In making this estimate, the capacitive-coupling between the few-electron dots was assumed to be negligible. A finite capacitive-coupling would alter the scaling of f to give [Golden and Halperin, 1996a; Adourian, 1996]:

$$(1 - f_C) = \frac{C_\Sigma}{C_\Sigma + 2C_{INT}}(1 - f) \tag{5.3}$$

where f is the fractional peak splitting given by theory, which does not include capacitive coupling, and f_C is the fractional peak splitting that includes the effect of interdot capacitance C_{INT}. The interdot capacitance C_{INT} may be found by setting the interdot tunnel-coupling as low as possible, and calculating C_{INT} using Equation 4.4 from the peak splitting that remains.

5.7 Conclusions

The state of implementing a fully-fledged Loss-DiVincenzo quantum computer is still in its infancy, but the first experimental steps have been taken in that direction. Dots containing just a handful of electrons have been made, and the number of electrons in these dots can be determined and controlled. The Zeeman energy of electrons in these

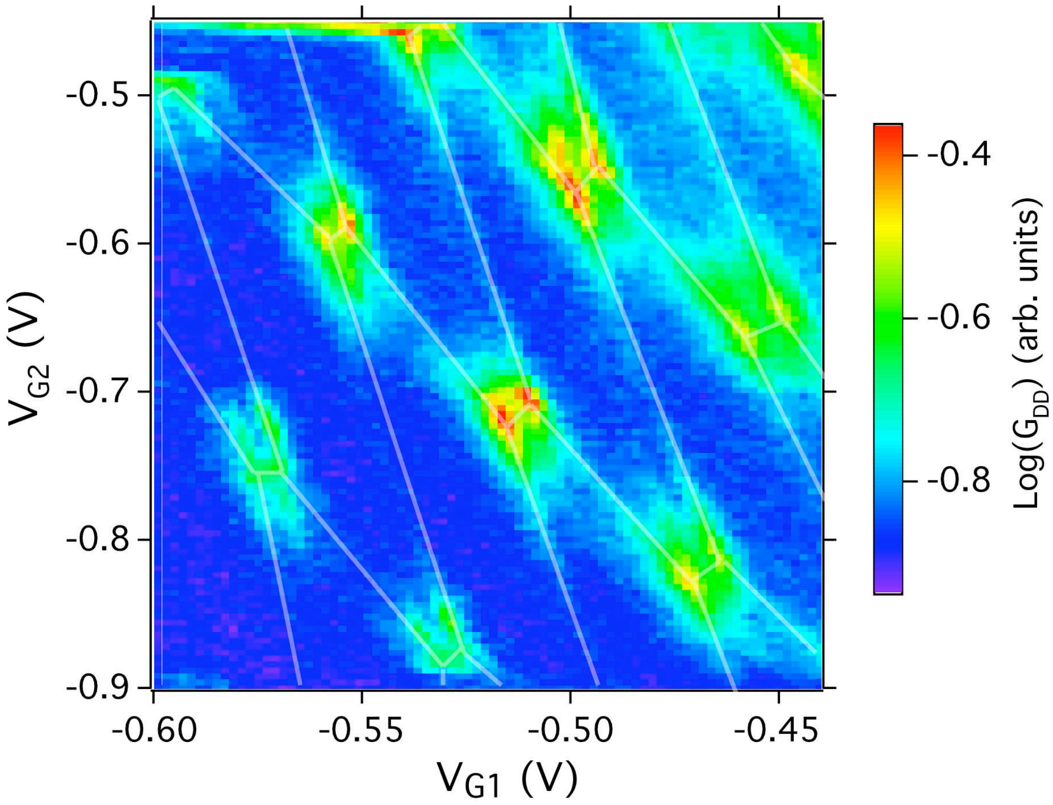


Figure 5.7 Conductance of tunnel-coupled few-electron dots in series as a function of sidegate voltages. The Coulomb blockade peaks are clearly split, and the hexagonal charge-stability pattern is highlighted. This hexagonal pattern is indicative of coupled double dots, and a fractional peak splitting $f = 0.3$ was measured.

few-electron dots was measured, and indicated that the behavior of electron spin in such small dots is no different from that in bulk GaAs (at least up to 7 T). Tunnel-coupling of few-electron dots, essential for implementing the swap$^{1/2}$ operation, was demonstrated, and the strength of the coupling was estimated from the fractional peak splitting.

The next steps in researching semiconductor quantum computers seem to be performing ESR, measuring spin coherence lifetimes, and controlling both the strength and the duration of tunnel-coupling between dots. In my opinion, controlling the tunnel-coupling is probably the most challenging and interesting task facing semiconductor quantum computers, as it goes to the heart of quantum computation, which is entanglement. The activity in quantum computing research will only intensify, and there will no doubt be many exciting results that will develop in the coming years.

Chapter 6: Conclusions and Future Directions

The application of quantum dots to single-electronic circuits and quantum computation was explored in the preceding few chapters. Chapter 4 explored using strongly capacitively-coupled quantum dots to make a single-electron switch. The strong capacitive coupling between dots was achieved using a coupling gate, and an order of magnitude increase in the coupling was achieved compared to similar devices without the coupling gate. The strong electrostatic coupling allowed single electrons entering or exiting one dot to shift the location of the Coulomb blockade peaks of the adjacent dot, which created a hexagonal pattern of split Coulomb blockade peaks in the sidegate voltages of the two dots. The capacitive charging model was found to explain the hexagonal pattern of peaks well, and the formula relating the fractional peak splitting f to the fractional capacitive coupling α was derived from the model. The strong capacitive-coupling was used to shift a dot on or off a Coulomb blockade peak to make a single-electron switch. The switch was triggered by a change in the charge state of one dot (the trigger dot), which changed the average conductance of the switched dot by two orders of magnitude. The abruptness of the switching in gate voltage was found to be determined by the charge quantization on the trigger dot. A trigger dot with well quantized charge switched the conductance of the switched dot abruptly in gate voltage, even if charge was poorly quantized in the switched dot. A trigger dot with poorly quantized charge switched the conductance of the switched dot gently in gate voltage, even if charge was well quantized in the switched dot. The abrupt switching lineshape was fit to a Fermi (tanh) function, while the gentle switching lineshape was fit to an arctan function.

Further work on capacitively-coupled dots could be done to build more complicated circuits and possibly logic gates. Indeed, there are proposals for building cellular automata computers with capacitively-coupled dots which could be tried [Tougaw and Lent, 1994; Porod, 1998]. These computers should be able to operate at very high speeds due to the very low self-capacitances of the dots and the coupling gates, and experimentally testing these speed limits could be an interesting project. The possible time-correlated currents that flow through strongly capacitively-coupled dots at the peak splitting is also intriguing, and warrants further study.

An initial effort to study quantum dots for quantum computing was detailed in Chapter 5. In the proposal put forward by Loss and DiVincenzo [1998], small quantum dots holding an electron each would store quantum bits of information (qubits) in their spins, and computation would be performed by flipping spins and entangling spins with controlled tunneling between dots. Small tunnel-coupled double dots were successfully made, and were shown to contain just a handful of electrons each. The number of electrons on each dot was controlled with the Coulomb blockade, and a novel technique of completely describing the conductance of a dot with sidegate and quantum point contact voltages was used to bolster the claim the dots could be emptied of electrons. The Zeeman energy of electrons in a few-electron dot was studied, and the resulting g-factor was found to be no different from that of electrons inside bulk GaAs, $|g| = 0.44$. Tunnel-coupling between few-electron dots was also demonstrated, and the characteristic split-peak Coulomb blockade conductance pattern was observed.

Much interesting work remains to be done in studying quantum dots as a candidate for quantum computing. An electrometer could be made near the qubit dots to use yet another method to detect the number of electrons in the qubit dots. The spin properties of the last electron in a dot also need to be scrutinized. A more accurate determination of the g-factor could be undertaken to measure any variations in the g-factor, and the effect of the confinement potential on the g-factor could be explored. Controlled tunnel-coupling between dots could also be studied. An interesting, if difficult, task could be to swap the location of a single electron inside a double dot by controlling the tunneling between the dots. One final experiment could be to make three-dimensional voltage scans of a dot at different in-plane magnetic fields (*i.e.* a four-dimensional scan) to observe how the shape of the confining potential in the dot may alter the order in which electronic states are filled.

References

Adourian, A.S. (1996). *Single Electron Transport in Parallel Coupled Quantum Dot Nanostructures*, Ph.D. Thesis, Harvard University.

Beenakker, C.W.J. (1991). *Phys. Rev. B* **44**, 1646.

Beenakker, C.W.J., and H. van Houten, (1991). "Quantum transport in semiconductor nanostructures," in *Solid State Physics* **44**, H. Ehrenreich and D. Turnbull, eds., Academic Press, San Diego.

Buttiker, M. (1986). *Phys. Rev. Lett.* **57**, 1761.

Buttiker, M. (1988). *IBM J. Res. Dev.* **32**, 63.

Chen, L.H. (2001). *Charge-Imaging Field-Effect Transistors for Scanned Probe Microscopy*, Ph.D. Thesis, Harvard University.

Chuang, I.L., L.M.K. Vandersypen, X. Zhou, D.W. Leung, and S. Lloyd (1998). *Nature* **393**, 143.

Cirac, J.I. and P. Zoller (1999). *Nature* **404**, 579.

Crouch, C.H. (1996). *Single Electron Transport and Charge Quantization in Coupled Quantum Dots*, Ph.D. Thesis, Harvard University.

DiVincenzo, D.P. (1995). *Phys. Rev. A* **51**, 1015.

Duncan, D.S. (2000). *Mesoscopic Electron Transport in Semiconductor Nanostructures*, Ph.D. Thesis, Harvard University.

Duncan, D.S., C. Livermore, R.M. Westervelt, K.D. Maranowski, and A.C. Gossard (1999). *Appl. Phys. Lett.* **74**, 1045.

Eriksson, M.A. (1997). *Cryogenic Scanning Probe Microscopy for Semiconductor Nanostructures*, Ph.D. Thesis, Harvard University.

Folk, J.A., R.M. Potok, C.M. Marcus, and V. Umansky (2003). *Science* **299**, 679.

Golden, J.M. and B.I. Halperin (1996a). *Phys. Rev. B* **53**, 3893.

Golden, J.M. and B.I. Halperin (1996b). *Phys. Rev. B* **54**, 16757.

Gould, C., A.S. Sachrajda, P. Hawrylak, P. Zawadzki, Y. Feng, and Z. Wasilewski (1999). *Proceedings of the Fifth International Symposium on Quantum Confinement: Nanostructures*, 270.

Grabert, H., and M.H. Devoret, eds. (1992). *Single Charge Tunneling*, NATO ASI Series B **294**, Plenum Press, New York.

Grover, L.K. (1997). *Phys. Rev. Lett.* **79**, 325.

Hanson, R., B. Witkamp, L.M.K. Vandersypen, L.H.W. van Beveren, J.M. Elzerman, and L.P. Kouwenhoven (2003). cond-mat/0303139.

Horowitz, P. and W. Hill (1989). *The Art of Electronics, 2nd Edition*, Cambridge Univ. Press, Cambridge, England.

Kane, B.E. (1998). *Nature* **393**, 133.

Kastner, M.A. (1992). *Rev. Mod. Phys.* **64**, 849.

Katine, J.A. (1996). *Electron Quantum Interference in Ballistic Semiconductor Nanostructures*, Ph.D. Thesis, Harvard University.

Landauer, R. (1957). *IBM J. Res. Dev.* **1**, 223.

LeRoy, B.J. (2003). *Imaging Coherent Electron Flow Through Semiconductor Nanostructures*, Ph.D. Thesis, Harvard University.

Likharev, K.K. (1999). *Proc. IEEE* **87**, 606.

Livermore, C., C.H. Crouch, R.M. Westervelt, K.L. Campman, and A.C. Gossard (1996). *Science* **274**, 1332.

Livermore, C., D.S. Duncan, R.M. Westervelt, K.D. Maranowski, and A.C. Gossard (1999). *J. Appl. Phys.* **86**, 4043.

Livermore, C.L. (1998). *Coulomb Blockade Spectroscopy in Tunnel-Coupled Quantum Dots*, Ph.D. Thesis, Harvard University.

Loss, D., and D.P. DiVincenzo (1998). *Phys. Rev. A* **57**, 120.

Lounasmaa, O.V. (1974). *Experimental Principles and Methods Below 1 K*, Academic Press, New York.

Meir, Y., N.S. Wingreen, and P.A. Lee (1991). *Phys. Rev. Lett.* **66**, 3048.

Mooij, J.E., T.P. Orlando, L. Levitov, L. Tian, C.H. van der Wal, and S. Lloyd (1999). *Science* **285**, 1036.

Porod, W. (1998). *Int. J. High Speed Electron. Syst.* **9**, 37.

Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition*, Cambridge Univ. Press, Cambridge, England.

Rimberg, A.J. (1992). *Magnetotransport in Uniform and Modulated Electron Gases in Wide Parabolic Quantum Wells*, Ph.D. Thesis, Harvard University.

Shor, P. (1997). *SIAM J. Comput*. **26**, 1484.

Sohn, L.L., L.P. Kouwenhoven, and G. Schon, eds. (1996). *Mesoscopic Electron Transport*, NATO ASI Series E **345**, Kluwer Press, Dordrecht.

Stone, A.D., and P.A. Lee (1985). *Phys. Rev. Lett*. **54**, 1196.

Tinkham, M. (1996). *Introduction to Superconductivity*, 2nd ed., McGraw-Hill, New York.

Topinka, M.A. (2002). *Imaging Coherent Electron Wave Flow Through 2-D Electron Gas Nanostructures*, Ph.D. Thesis, Harvard University.

Tougaw, P.D. and C.S. Lent (1994). *J. Appl. Phys*. **75**, 1818.

van Wees, B.J., H. van Houten, C.W.J. Beenakker, J.G. Williamson, L.P. Kouwenhoven, D. van der Marel, and C.T. Foxon (1988). *Phys. Rev. Lett*. **60**, 848.

Vandersypen, L.M.K., M. Steffan, G. Breyta, C.S. Yannoni, M.H. Sherwood, and I.L. Chuang (2001). *Nature* **414**, 883.

Waugh, F.R. (1994). Novel Architectures and Devices for Computing, Ph.D. thesis, Harvard University.

Wharam, D.A., T.J. Thornton, R. Newbury, M. Pepper, H. Ahmend, J.E.F. Frost, D.G. Hasko, D.C. Peakcock, D.A. Ritchie, and G.A.C. Jones (1988). *J. Phy. C* **21** L209.

Yacoby, A., H.F. Hess, T.A. Fulton, L.N. Pfeiffer, K.W. West (1999). *Solid State Commun*. **111**, 1.

Zumbuhl, D.M., J.B. Miller, C.M. Marcus, K. Campman, and A.C. Gossard (2002). *Phys. Rev. Lett*. **89**, 276803/1.

Appendices

A.1 Ramper Box

A.1.1 Ramper Box Circuit Diagram

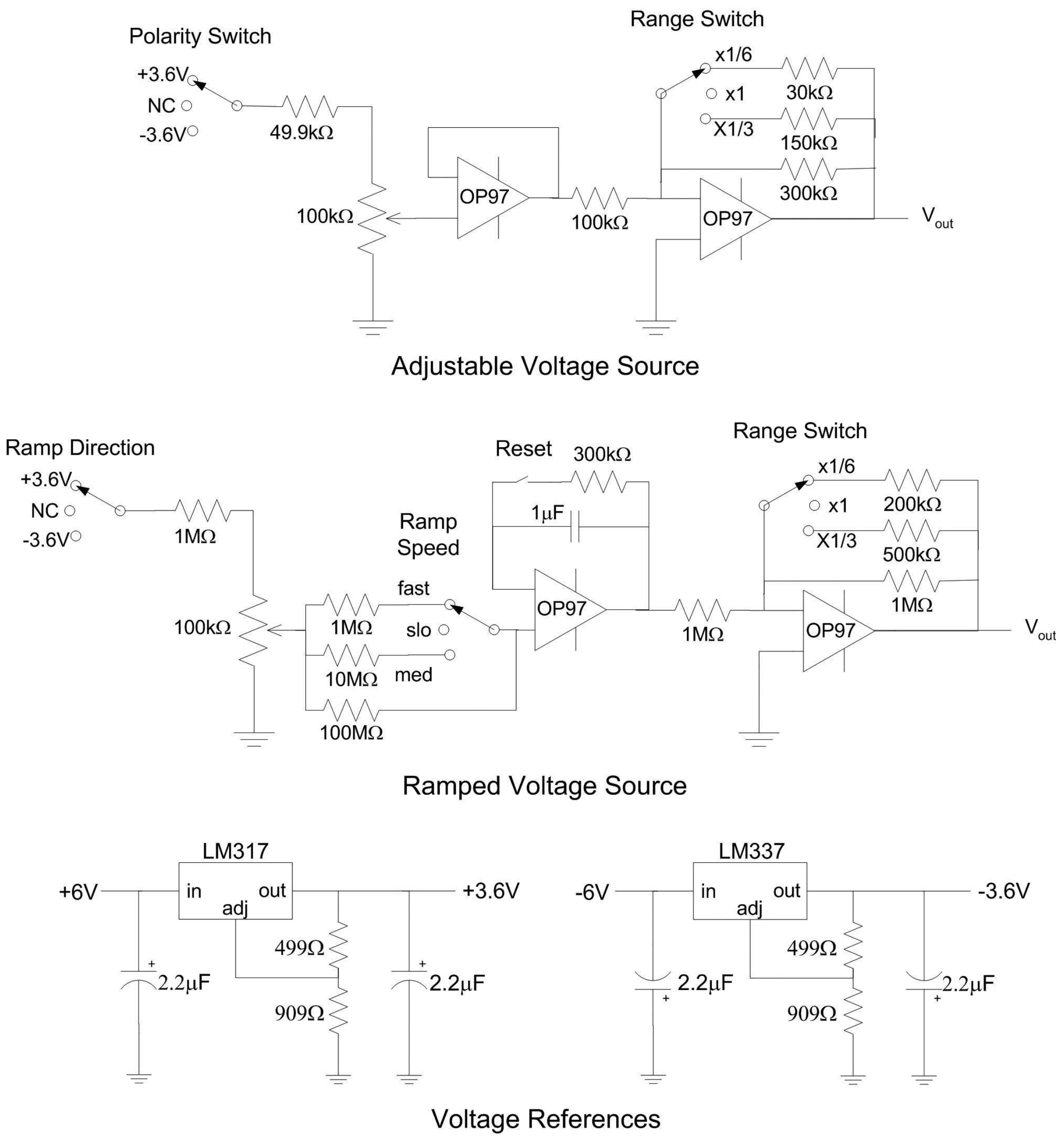


Figure A.1 Circuit diagrams for the Ramper box, showing (from top to bottom) adjustable voltage source, ramped voltage source, and positive and negative voltage references.

A.1.2 Ramper Box Reference Voltages

A pair of Energizer lithium AA batteries (L91) were used to generate a reference voltage of ~3 V (3.6 V in practice) for the Ramper Box. It was discovered however, that the output of the lithium batteries drifted by tens of mV over days, which is a large amount. Replacing the batteries with LM317 (positive) and LM337 (negative) voltage regulators improved the stability of the reference voltage tremendously, varying less than half a mV over 2.5 days.

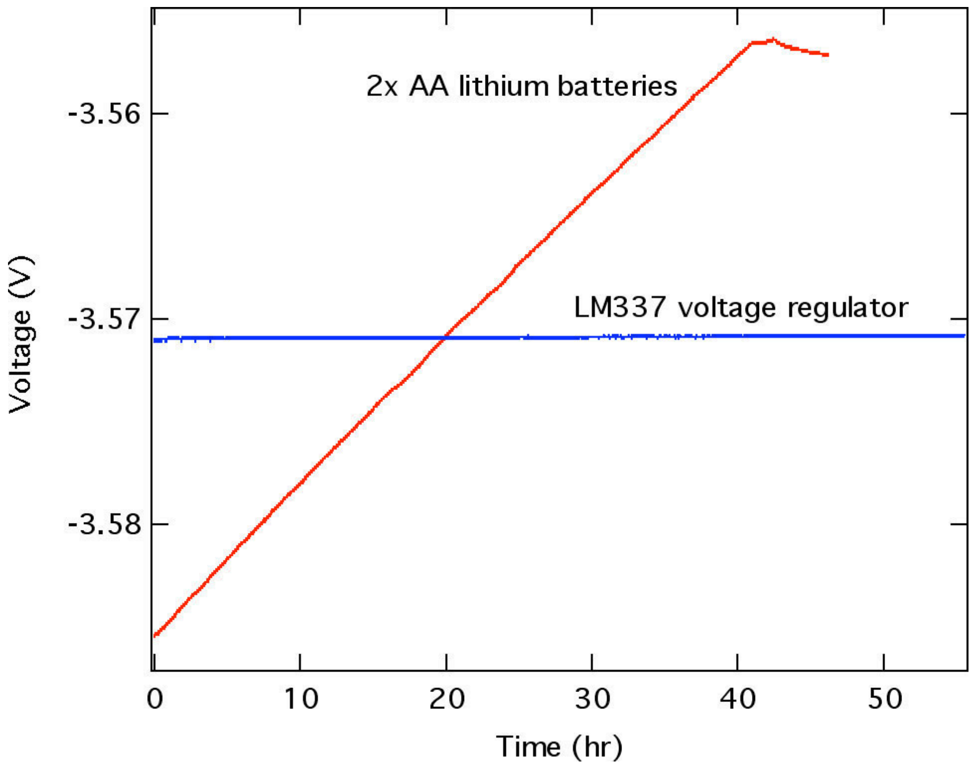


Figure A.2 Comparison of voltage drift between two Energizer lithium AA batteries and the LM337 voltage regulator. Data was acquired using the HP34401A multimeter set to slow. The LM337 voltage regulator output varied less than 0.5 mV while the lithium batteries drifted by 30 mV over two days.

A.2 BiasDac Box

A.2.1 BiasDac Board Circuit Diagram

This section was copied from BiasDac schematic diagrams made by Jim MacArthur.

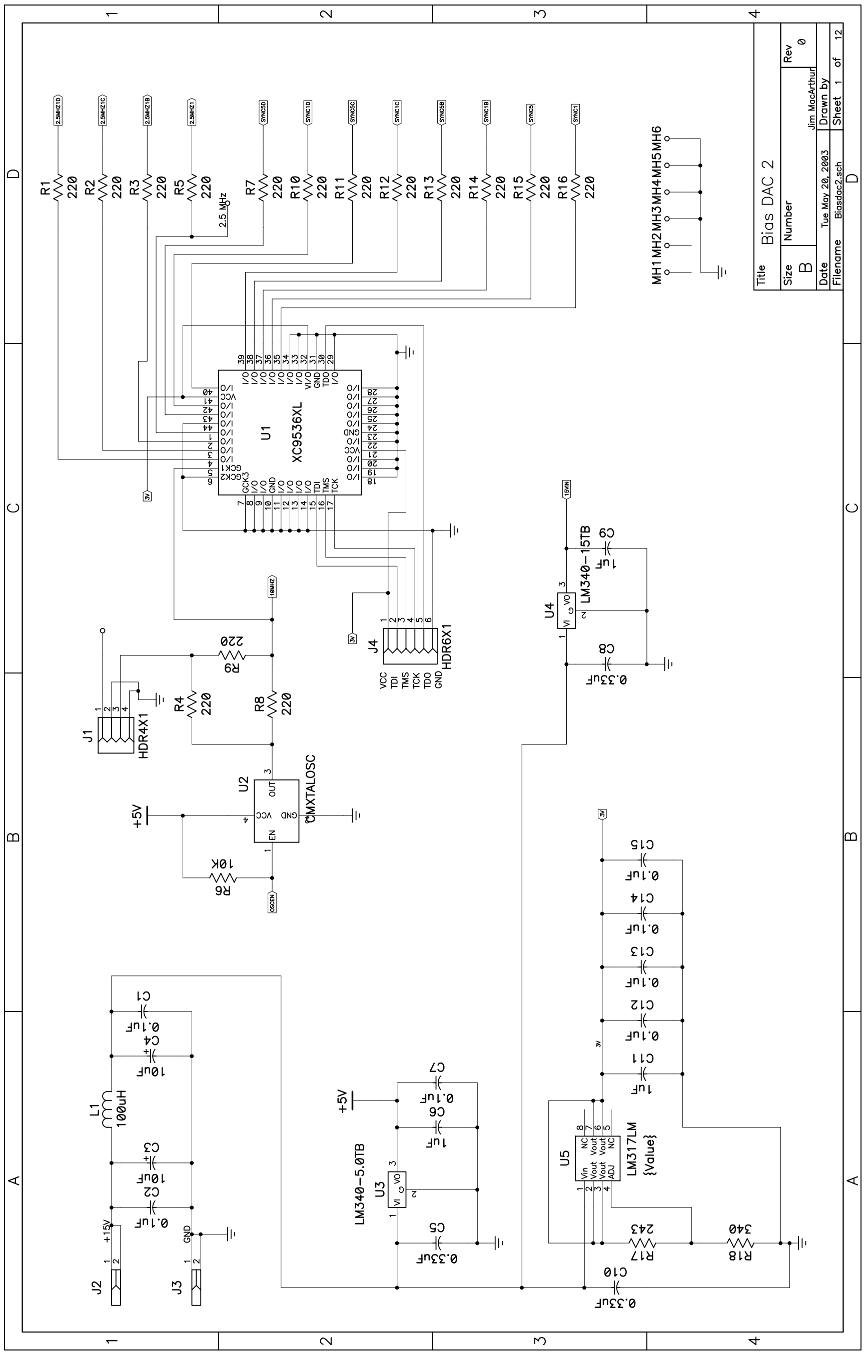


Figure A.3 Circuit diagram of the BiasDac showing the programmable CPLD, crystal oscillator, and digital power circuits.

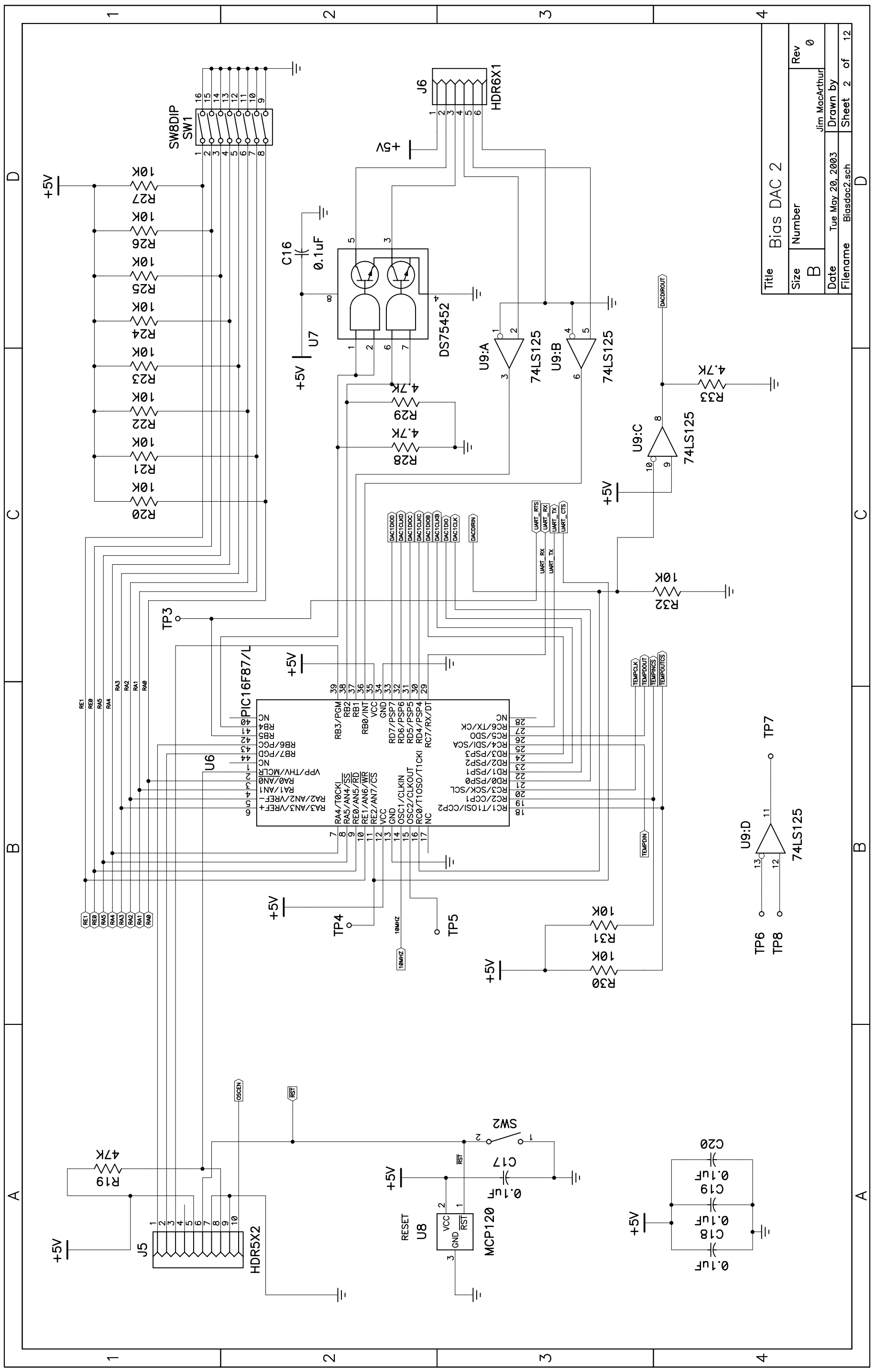


Figure A.4 Circuit diagram of the BiasDac showing the microcontroller and various digital logic.

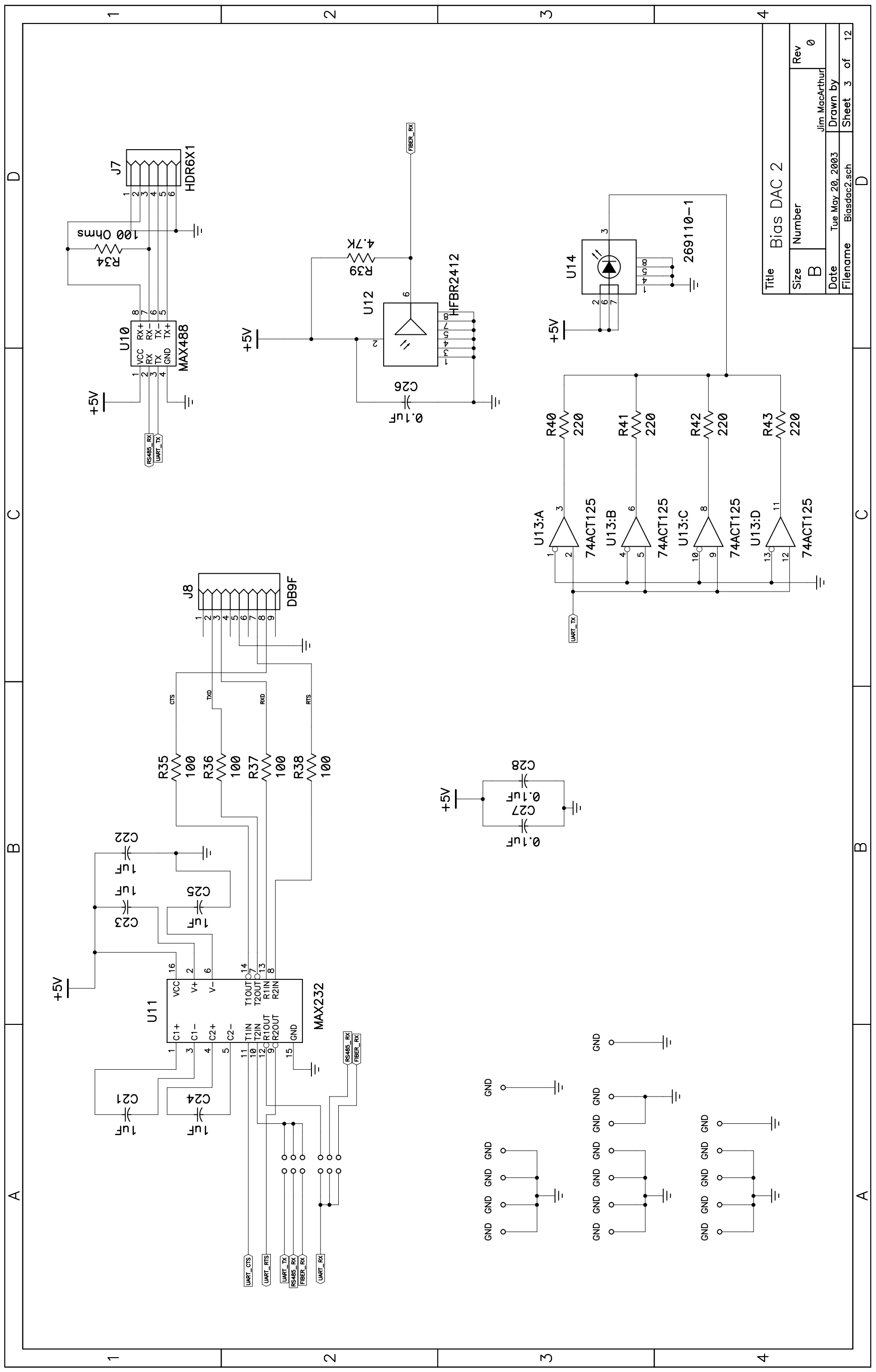


Figure A.5 Circuit diagram of the BiasDac showing the serial input/output circuits (including fiber-optic I/O).

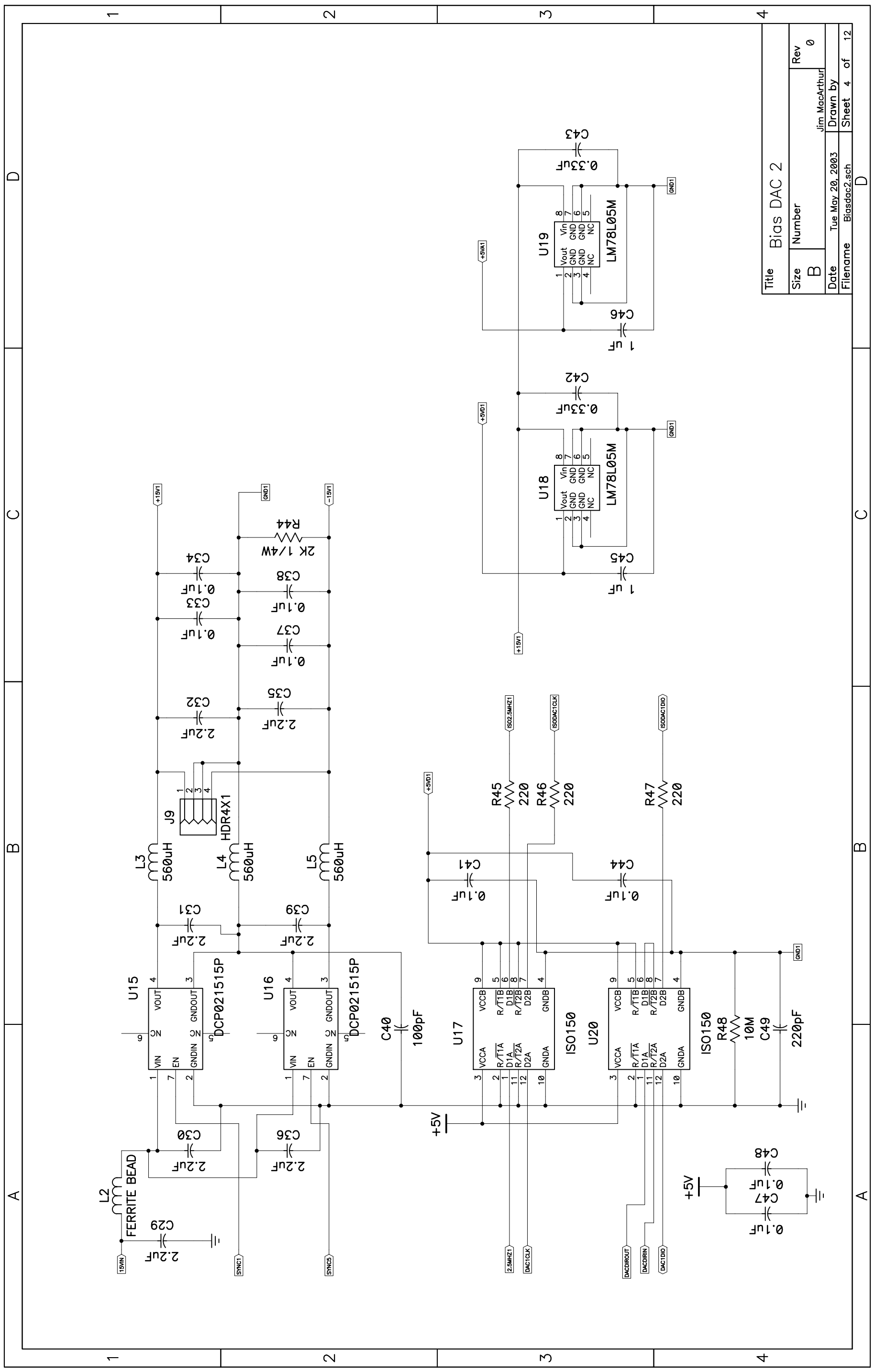


Figure A.6 Circuit diagram of the BiasDac showing the power and data isolation for each BiasDac channel. Note that the DC-DC converters were removed (see Appendix A.2.4).

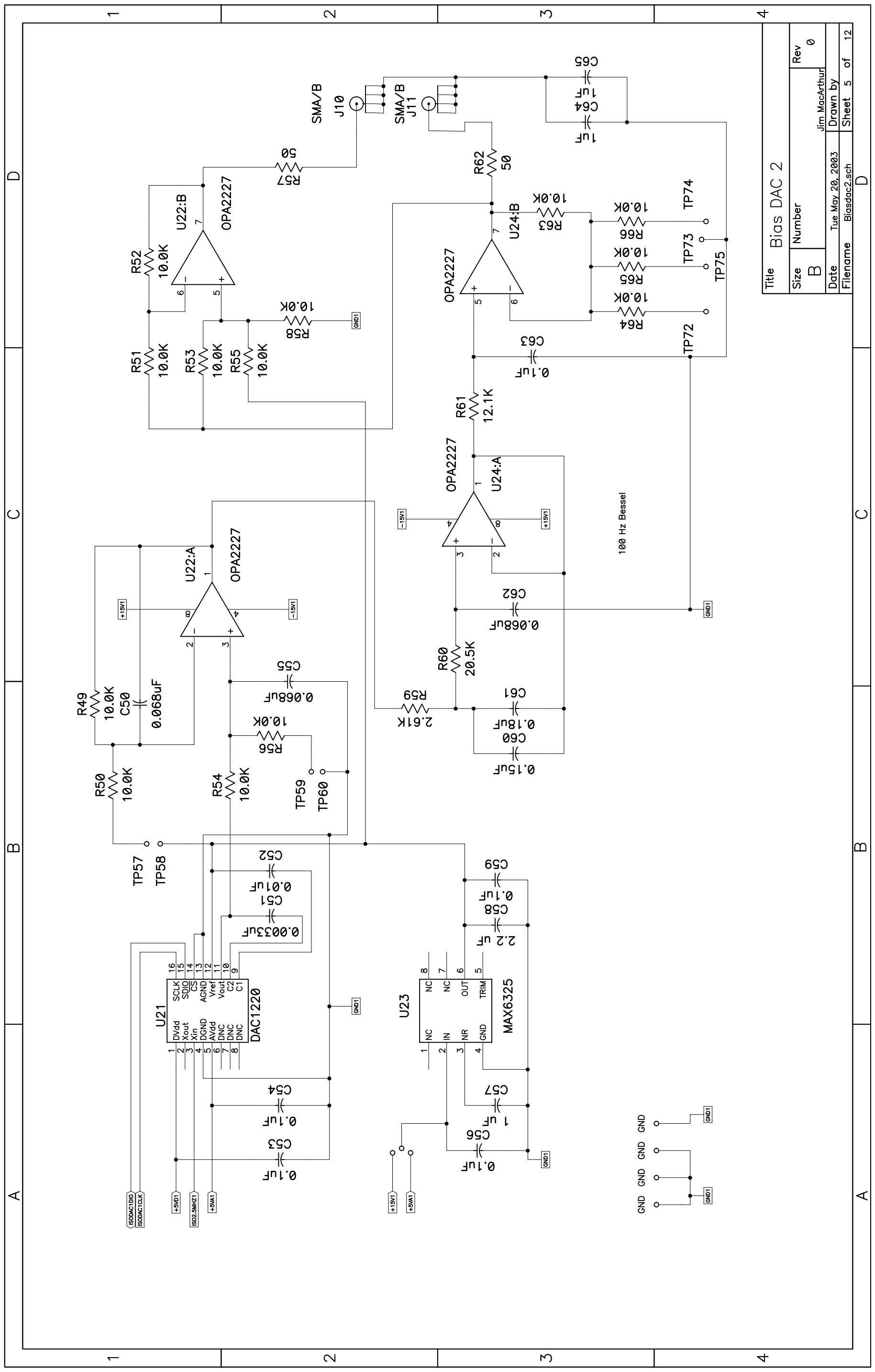


Figure A.7 Circuit diagram of the BiasDac showing the DAC, voltage reference, and supporting op-amps for each BiasDac channel. The TP*XX* jumpers are used to set the output range (see Appendix A.2.2).

A.2.2 BiasDac Board Hardware Settings

This section was largely taken from the document BiasDACusr.doc written by Jim MacArthur. The last change to that document was made on 10/02/01.

Dipswitch Settings (Device ID and Baud Rate)

There is an 8-switch dipswitch located in the upper left of the board. For the following discussion, “OFF” means the rightmost part of the switch (the side that says “open”) is depressed, and “ON” means the leftmost part (the side that has the switch numbers) is depressed.

Each device on a fiber loop must have a unique ID. The communications protocol allows up to 61 unique device IDs. These IDs are set either in EEPROM (with the Change Device ID command) or with dipswitches 1-5. If the dipswitches are all set to 0 (OFF), then the ID previously stored in EEPROM is used. Otherwise, the dipswitches determine the ID. As there are only 5 dipswitches, that means that there are only 31 unique IDs which can be set with dipswitches. As a practical matter, it is probably best to rely on dipswitches rather than EEPROM settings, as they are more user-friendly. Switch 5 is the MSB of the ID number, and switch 0 is the LSB. The default value is ID #1, meaning switch 1 is ON and switches 2-5 are OFF.

The three uppermost switches determine the baudrate of the communications interface:

8	**7**	**6**	**baudrate**
OFF	OFF	OFF	baud defined in EEPROM (see A.2.3)
OFF	OFF	ON	9600
OFF	ON	OFF	14.4K
OFF	ON	ON	19.2K
ON	OFF	OFF	38.4K
ON	OFF	ON	57.6K
ON	ON	OFF	undefined
ON	ON	ON	undefined

The default setting is ON-OFF-ON (57.6K)

Serial Port Jumper Settings

There are two sets of jumpers located in the upper right of the board. The lower set has the label “UART-RX” beneath it. It selects the serial source feeding the on-board microcontroller. When the jumper is in the leftmost setting, the source is the fiber optic receiver. The middle setting is for the RS485 differential receiver. The rightmost setting is for the RS232 receiver.

The upper jumper set has the label “RS232-TX” above it. It determines what is transmitted out the RS232 transmitter. In the leftmost setting, the fiber optic receiver is routed directly to the RS232 transmitter without going through the microcontroller. In the middle setting, the RS485 receiver is routed to the RS232 transmitter. In the rightmost setting, the microcontroller is routed to the RS232 transmitter.

Don’t panic. This seems to be a difficult topic for folks to understand, so I’ll give a few examples of how to set up these jumpers. It’s important to understand that the communications protocol is based on a loop topology: data comes from the host PC, into the first device, then from the first device to the second, and so on, then from the last device back to the host PC.

Start with the simple case of a single device hooked up to a PC through RS232. Simply plug a 9-pin RS232 cable from the PC into the device, and set the lower jumper to the rightmost setting. This connects the device’s RS232 receiver to the microcontroller. Setting the upper jumper to the rightmost setting connects the microcontroller to the RS232 transmitter. This completes the loop: PC to microcontroller to PC.

Now let’s replace the RS232 cable with a fiber link (using an RS232-fiber interface box available from the Electronics Shop). This time, the lower jumper is set to the leftmost setting, which connects the fiber optic receiver to the microcontroller. The upper jumper setting doesn’t matter, as the fiber optic transmitter is *always* connected to the microcontroller. If you wanted to add another device to this configuration, you would run a fiber from the transmitter of the first device to the receiver of the second, and a fiber from the transmitter of the second device to the receiver of the RS232-fiber interface box. You can keep this up to a total of 61 devices sharing a fiber ring.

Now let's get tricky. Suppose you don't want to pony up the $200 for an RS232-fiber interface box. You can make the first device perform the interface. Connect the first device to the PC with an RS232 cable. Set the lower jumper to the rightmost setting (RS232 to microcontroller). Set the upper jumper to the *leftmost* setting (fiber receiver to RS232 transmitter). Put a fiber optic cable between the transmitter and the receiver of the next device in the loop. Connect the last device in the loop back to the fiber optic receiver of the first device. The upper jumper will close the loop by routing the fiber optic receiver to the RS232 transmitter, and then to the PC.

One last case: you've got two boards sharing an enclosure, and you've wisely chosen to use the Master/Slave kit to connect and synchronize them. In addition to synchronizing the system clock, the kit connects the serial ports of the boards together over the RS485 interface. The correct jumper settings are:

> For the Master: lower jumper is set to leftmost position for a fiber optic input and rightmost position for RS232 input. If there are only two devices in the loop and the master is using the RS232 interface, then set the upper jumper to the middle position, which connects its RS232 transmitter to the RS485 receiver from the slave.
>
> For the Slave: lower jumper is set to middle position, which connects the RS485 receiver to the microcontroller. This means that serial data is coming directly from the master without leaving the enclosure. Upper jumper is don't-care.

To connect the Master/Slave into a fiber ring, run a fiber optic cable into the master's receiver, and another cable from the slave's transmitter to the next device in the chain.

Output Range Settings

The output range of each BiasDac channel can be configured in hardware for unipolar or bipolar output, and various output ranges (see Figure A.7 for part references):

Polarity	Jumper Settings	Op-amp Output
Unipolar	Do not short jumpers	0 to +5 V
Bipolar	Short TP57 & TP58 Short TP59 & TP60	-2.5 to +2.5 V

Op-amp Gain	Jumper Settings	Unipolar Range	Bipolar Range
1×	Do not short jumpers	0 to 5 V	-2.5 to +2.5 V
2×	Short TP74 & TP75	0 to 10 V	-5.0 to +5.0 V
3×	Short TP73, TP74 & TP75	N/A	-7.5 to +7.5 V
4×	Short TP72, TP73, TP74 & TP75	N/A	-10 to +10 V

A.2.3 BiasDac Communications Protocol

This section was largely taken from the document CommsProtocol.doc written by Jim MacArthur. The last change to that document was made on 11/07/02.

Overview

The link is set up as a ring with a single host and up to 61 target devices. Typically, each microcontroller represents a device, so if you have two 4-channel DAC boards in the same enclosure, they represent two independent devices. The serial output of the host is connected to the first device in the chain. The output of the first device goes to the input of the second device, etc. The output of the last device goes back to the host. Thus, the host can control up to 61 devices with a single serial I/O port.

Devices may support any or all of the following serial interfaces:
RS-232 (no hardware handshaking)
RS-488
Fiber-optic (840 nm, ST termination)

These interfaces may be mixed and matched throughout the ring. For example, a PC's RS232 output could go through a fiber optic adapter, then through fiber to the first device. From the first device, it could go through regular logic level signals to a second device located in the same enclosure, and from there through RS488 or fiber to another device in the same rack, and from there through fiber back to the PC. The serial protocol is standard asynchronous, 8 data bits, 1 stop bit, no parity. Supported bauds: 9600, 19200, 38400, 57600. Others possible. All devices must be running at the same baud. Eventually, the device's baud will be stored in EEPROM, but for now it will be configured with a dipswitch.

For all devices, the input data stream goes into a UART, then into a microprocessor, then is retransmitted through a UART to the next device. In other words, input serial streams are not simply electrically buffered and sent along. This has two benefits:

1) It allows very large rings because it doesn't allow timing slop to accumulate.
2) It guarantees that the data that the microprocessor read is the same data it passed along, removing a possible source of communication error.

The host controls the data flow. Each device waits to receive a byte, then retransmits either the same byte, or a different byte, depending on the protocol (see below). Regardless, every device always transmits a single byte for every byte it receives, with the exception of the No Echo byte. In order to prevent buffer overruns in target devices, the host may transmit No Echo bytes, which have a value of 0xFF, between commands. When a device receives a "No Echo" byte, it absorbs the byte and takes no further action. It does not pass it down along the serial chain. No Echo bytes allow a host to pack a series of commands into a transmit buffer and still allow an adjustable amount of space between commands to prevent the device's buffers from overrunning. Conservative programming practice suggests adding a No Echo byte at the end of every command packet, if you can afford the reduced command throughput.

The MSB of every byte is reserved for a Sync Bit. This bit, when set, indicates either the start of a command from the host, or a status byte returned from a target. All other transmitted bytes must have their MSB set to 0. When the Sync Bit is set, the C/S bit (next bit after MSB) indicates whether the byte is the start of a command (C/S = 1) or a status byte (C/S = 0).

Each Device has a 6-bit ID which is unique on a physical network. IDs 00 and 63 are reserved. Eventually, Device IDs will be stored in the Device's EEPROM, but for now it will be configured with a dipswitch.

Command Format

All commands consist of at least two bytes, followed by a checksum byte and a zero pad byte:

BYTE 1 = ID byte
11dddddd, where dddddd is the target Device ID. Note that the sync bit (bit 7) is set to "1", indicating the start of a command or status message. The C/S bit (bit 6) is also set to "1", indicating that the following message is a command to device dddddd.

BYTE 2 = command byte
00000000 = reserved
00000001 - 00111111 = Universal commands supported by all Devices
01000000 - 01111111 = Device-specific commands

BYTES 3-N = data
0-31 bytes of argument data, depending on the command. If the host is sending information to the device, it sends it here. If the host is requesting information, it sends a string of 0's, which the device replaces with the requested data. A command may both send and request data in the same string.

BYTE N+1 = parity
The host creates this byte by XORing every preceding byte in the command (and setting the MSB to 0). The target checks parity by XORing all of the command bytes, including the parity byte (and setting the MSB to 0). The result should be 0. The target replaces this byte with a parity byte generated from its outgoing data stream (which may be different from the incoming stream).

BYTE N+2 = zero-pad/status.
The host sends a byte set to 0. The selected target replaces it with a status byte (defined below).

Universal Commands

Here is the map of currently implemented Universal Commands. The commands are defined below.

00000001	Clear Error
00000010	Read From Memory
00000011	Write to Memory
00000100	Stop Program
00000101	Run Program
00000110	Change Device ID
00000111	Change Baud
00001000	Backup Settings
00001001	Set Mode Flags
00001010	Set Interrupt Period
00001011	Store Program
00001100	Store Macro
00001101	Run Macro
00001110	Block Read
00001111	Block Write
0001xxxx	Unspecified
001nnnnn	Get Device Info

<u>Clear Error</u>: 00000001

Clears "sticky" errors resulting from power-on self-tests, etc. Allows host to test device operation even if, for example, one of the device's channels is bad.

<u>Read from Memory</u>: 00000010

Diagnostic used to read one byte of device-specific data from memory The next byte is the 7 high address bits. The next byte is the 7 low address bits. The next two bytes are set to zero by the host, and filled with data by the target, as follows:

The next byte has the high nybble of data in its low nybble
The next byte has the low nybble of data in its low nybble

<u>Write to Memory</u>: 00000011

Diagnostic used to write one byte of device-specific data to memory The next byte is the 7 high address bits. The next byte is the 7 low address bits. The next byte has the high nybble of data in its low nibble. The next byte has the low nybble of data in its low nibble. In the case of the BiasDAC, reads and writes to addresses 0x0000 – 0x01FF will access the microprocessor's internal register space. Writing to this space has a high probability of causing mischief.

Reads and writes to 0x0200 – 0x027F will access the non-volatile memory used to store programs (see the Store Program command below). While the Store Program command is more convenient for storing programs, reading this space will tell a user what programs are stored there.

Reads and writes to 0x0280 – 0x02C0 will access 64 bytes of uncommitted non-volatile memory. Users may use this space to store board-specific parameters such as calibration constants, configuration information, etc. Refer to the section on Writing to Non-Volatile Memory for warnings on performing NVRAM writes.

Reads and writes to 0x0300 – 0x033F will access the internal registers of the four D/A chips. Bits 5-4 select the D/A and bits 3-0 select the register within the D/A. The D/A is a Burr-Brown DAC1220, and its registers are described in:

http://www-s.ti.com/sc/psheets/sbas082/sbas082.pdf

<u>Stop Program</u>: 00000100

Stops execution of preprogrammed commands,

<u>Run Program</u>: 00000101

Runs pre-stored program, starting at following byte location. The next byte is the starting address of the program (0-127)

<u>Change Device ID</u>: 00000110

Changes the current Device ID, and the copy stored in non-volatile memory. The next byte is the new ID. Note that this can be overridden by dipswitch settings on the device. Refer to the section on Writing to Non-Volatile Memory for warnings on using this instruction.

<u>Change Baud</u>: 00000111

Changes the copy of the baud stored in non-volatile memory, but not the current baud. Therefore, the change doesn’t take effect until a reset. The next byte is the new baud code, defined as follows:

00000000	= 9600 baud
00000001	= 14.4K
00000010	= 19.2K
00000011	= 38.4K
00000100	= 57.6K

Note that this can be overridden by dipswitch settings on the device. Refer to the section on Writing to Non-Volatile Memory for warnings on using this instruction.

<u>Backup Settings</u>: 00001000

Copies all device-specific volatile parameters into NV memory. For example, in the case of the BiasDAC, all current DAC settings get backed up. If power is then cycled, the DAC outputs will then default to the new settings. Refer to the section on Writing to Non-Volatile Memory for warnings on using this instruction. Because multiple bytes are written to NVRAM, the user must wait at least 250ms before performing another write to non-volatile memory.

<u>Set Mode Flags</u>: 00001001

Sets or clears the seven device-independent mode flags. The next byte has the new flag settings. At the moment, the only defined flag is BootToProgram, bit 0. When set to one, the target will attempt to execute a program, starting at location 0, after power-up. This command causes the changed mode flags to be stored in the device’s non-volatile memory. Refer to the section on Writing to Non-Volatile Memory for warnings on using this instruction.

<u>Set Interrupt Period</u>: 00001010.

Sets the period of the target’s internal interrupt, in microseconds. The next byte has the 7 MSBs of the period. The next byte has the 7 LSBs of the period. It is up to the host to stay within min and max period settings. At the moment, BiasDAC’s minimum period is 500 μs, and maximum is 10,000 μs. The default interrupt period is 500 μs, and some caution should be taken when changing it. As all of the program mode timing is based on interrupt periods, changing the period will change the program mode timing. Extremely long interrupt periods may also affect the temperature control loop.

Store Program: 00001011

Stores one instruction byte in the specified location, for later execution in program mode. The next byte has the location to be stored into. The next byte has the instruction to be stored. The instructions used in program mode are defined later in this spec. Refer to the section on Writing to Non-Volatile Memory for warnings on using this instruction.

Store Macro: 00001100

Identical to Store Program, except that it stores instructions into "macro memory", a volatile SRAM area 64 bytes long. Use the Store Macro and Run Macro commands when creating and running temporary programs that don't need to survive a power cycle. This preserves the life of the non-volatile memory.

Run Macro: 00001101

Identical to Run Program, except that it starts execution from macro memory. See the "Store Macro" instruction above.

Block Read: 00001110

Reads up to 127 bytes from the device. The next three bytes have 21 bits of address information, high byte first. The next byte has the number of bytes to read, called N. The next N bytes are padding, and replaced with data read from the device. The format of this data is device-specific.

Block Write: 00001111

Writes up to 127 bytes from the device. The next three bytes have 21 bits of address information, high byte first. The next byte has the number of bytes to write, called N. The next N bytes are the data to write to the device. The format of this data is device-specific.

Commands 00001110 – 00011111 are currently unspecified.

Get Device Info: 001nnnnn

Requests nnnnn bytes of device information. The command byte should be followed by nnnnn bytes of zeros, which the target replaces with device information. For all devices, the first byte should be the model number. The second byte will be the revision number. The remaining bytes will be an ASCII string with more device information. Current model numbers:

BiasDAC = 1
Frequency Counter = 2
Event Generator = 3

BiasDac/BabyDac-Specific Commands

Here is the map of currently implemented BiasDAC-Specific Commands. The commands are defined below.

010000cc	Update DAC channel
010001cc	Update High DAC channel (BabyDAC only)
010010cc	Update DAC mask
010011xx	undefined digibaby readback
010100cc	Update DAC slope
01011Sff	Set/Clear flags
011000cc	Get temperature (BiasDAC only, cc = 00)
	Get ADC value (BabyDAC only)
011001cc	DigiBaby write (BabyDAC only)
011010cc	Update DAC curve
011011cc	Recalibrate DAC channel
011100cc	Set DAC lower limit
01110100	Read AnaBaby (BabyDAC only)
01110101	undefined
0111011x	undefined
011110cc	Set DAC upper limit
011111xx	undefined

Update DAC channel: 010000cc

Followed by three bytes of DAC data, which update channel cc. The first byte of data is 00aaaaaa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. They get combined into 20-bit data of the form aaaaaabbbbbbcccccc. This is an unsigned number, meaning that 00000000000000000000 is the bottom of the DAC's range and 11111111111111111111 is the top. On power-up, the DACs are set to the values that were stored in non-volatile memory by the last Backup command.

Update High DAC channel: 010001cc (BabyDAC only)

Identical to Update DAC channel, except that it updates on of four channels on the daughterboard. Note that as of BabyDAC rev 2, the daughterboard DACs were NOT set to the values stored in NV memory.

Update DAC mask: 010010cc

Followed by two bytes of mask data. The first byte has the high nybble, and the second byte has the low nybble. DAC mask data is used in conjunction with the Update DAC Slope command to create ramped DAC waveforms. Because each interrupt only has time to update one DAC, the user programs the four DAC masks to determine which DAC gets updated when. The 8 bits of each mask correspond to 8 consecutive interrupt slots. For example, if you set:

> DAC0 mask to 01010101,
> DAC1 mask to 10000000,
> DAC2 mask to 00100000, and
> DAC3 mask to 00001000,

then DAC0 would be updated every other interrupt, and DACs 1, 2, and 3 would be updated every 8 interrupts. On power-up, all DAC masks are set to 0.

Update DAC Slope: 010100cc

Followed by 4 bytes of DAC slope data, which update channel cc. The first byte is 0aaaaaaa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. The fourth byte is 0ddddddd. They get combined into 28-bit data of the form aaaaaaabbbbbbbcccccccdd ddddd. This is a 2's comp. number which is added to the existing DAC value on every interrupt for which the selected DAC's mask bit is set. In other words, if the interrupt has been programmed with a period of 500 μs, and DAC0's mask is set to 00010001, then DAC0's value will be incremented by the slope number every 4th interrupt, or every 2ms. The slope is a left-justified value. To calculate the value for the slope, first determine the desired change in voltage as a fraction of full scale. Multiply this fraction by 4294967296 (2^{32}). Divide this result by the total number of DAC updates that will take place during the slope. As a specific example, if we have a DAC whose range is –5V to +5V, and we want to ramp from –2V to +1V, that represents a 3V rise, or 0.3 * full scale. Suppose we've set the interrupt time to 500μs, and the DAC mask to 00010001, so that it updates every 2ms, or 500Hz. If we want the slope to last for 10 seconds, that makes 5000 updates. So the DAC value = (4294967296 * 0.3) / 5000, or 257698. In hex, that's 0x0003EEA2. We drop the 4 LSBs. The final 28-bit binary number is 0000 0000 0000 0011 1110 1110 1010. Note that if the DAC update exceeds the upper or lower DAC value limit (see Set Lower Limit, below), then the slope is set to 0, and the DAC value is set to the limit. On power-up, the initial slope value is 0.

Set/Clear flags: 01011SFF

S=0 clears the flag and S=1 sets it. Sets or clears one of four hardware flags. FF selects the flag. For BiasDAC:

00 sets/clears PORTB.4, which is connected to pins 1 and 2 of driver IC U7
01 sets/clears PORTB.2, which is connected to pins 6 and 7 of driver IC U7
10 sets/clears UART_CTS

Note that U7 is an inverting driver. A high on the input turns on the output transistor, which pulls the output pin to ground.

For BabyDAC,

00 sets/clears PORTD.0 (pin 23 of DB37 connector)
01 sets/clears PORTD.1 (pin 5 of DB37 connector)
10 sets/clears PORTD.2 (pin 24 of DB37 connector)
11 sets/clears PORTD.3 (pin 6 of DB37 connector)

These bits are readable as the four LSBs of memory address 08.

Get Temperature: 01100000 (BiasDAC Only)

Followed by two bytes of 0, which are replaced with temp data from the target. The first byte has the 6 high bits of temp data in its 6 LSBs, and the second byte has the 7 low bits of temp data in its 7 LSBs. The data is two's complement, with 0 = 0 Celcius, and 0.0625 degrees C per LSB.

Get ADC Value: 011000bc (BabyDAC Only)

Followed by three bytes of 0, which are replaced with ADC data from the selected channel. The first byte of data is 0aaaaaaa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. They get combined into 21-bit data of the form aaaaaaabbbbbbbccccccc. This is an unsigned number, meaning that 000000000000000000000 is the bottom of the ADC's range and 111111111111111111111 is the top. Setting the "b" bit 0 selects the ADC on the main BabyDAC board. Setting "b" to 1 selects the ADC on the daughterboard. The "c" bit selects one of the two channels on each board.

DigiBaby Write: 011001bc (BabyDAC Only)

Followed by three bytes of digibaby data, which update register bc on the digibaby. The first byte of data is 00aaaaaa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. They get combined into 20-bit data of the form aaaaaabbbbbbbccccccc. If Digibaby is the first daughterboard, set "b" to 0. If it is the second daughterboard, set "b" to 1. The "c" bit selects between two 20-bit registers on each Digibaby.

Update DAC Curve: 011010cc

Followed by 4 bytes of DAC curve data, which update channel cc. The first byte is 0aaaaaaa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. The fourth byte is 0ddddddd. They get combined into 28-bit data of the form aaaaaaabbbbbbbcccccccddd dddd. This is a 2's comp. number which is first divided by 16, then added to the existing DAC slope on every interrupt for which the selected DAC's mask bit is set. This allows the operator to program quadratic curves into the BiasDAC.

Recalibrate DAC channel: 011011cc (BiasDAC Only)

Sends a recalibrate instruction to the selected D/A chip. While BiasDAC performs a DAC calibration on power-up, it is suggested that a recalibration be performed on all channels once the unit has reached thermal equilibrium.

NOTE: the recalibration can take as long as 500ms. During that time, no commands which update DAC values should be sent to the selected DAC channel (this includes ramps and commands sent under program control). Other DAC channels are unaffected by the command.

Set DAC Lower Limit: 011100cc

Set DAC Upper Limit: 011110cc

Followed by three bytes of DAC data, which update channel cc. These bytes are in the same format as in the Update DAC Channel command. Once the upper and lower limits are set, if the user attempts to set the DAC channel outside these limits, either with an Update DAC Channel or an Update DAC Slope command, the DAC value will be set to the nearer limit, and the slope will be set to 0. On power-up, the upper limit is set to FFFFF, and the lower limit is set to 00000.

Read Anababy: 01110100 (BabyDAC only)

Followed by 4 bytes. The first byte is the channel number. The second byte is set to 0, and is replaced by the high 2 bits of the returned ADC data. The third byte is set to 0, and replaced by bits 7-13 of the ADC data. The fourth byte is set to 0, and replaced by bits 0-6 of the ADC data.

Status Byte:

A selected target may, at any time during the transmission of a command block, replace any byte with a status byte of the format 10ssssss, indicating an error. If there is no error in the execution of the command, it replaces the last byte of the command (the zero pad) with 1000000 (0x80), indicating normal status. Other possible status bytes are:

0x81 = parity error
0x82 = non-supported command
0x83 = argument out of range

0x84 = device busy
0x85 = device recovered from reset

Writing to Non-Volatile Memory (NVRAM)

The target devices contain non-volatile memory (in the form of EEPROM) which retains data even with the power removed. The following commands perform writes to this memory:

Write to Memory (certain addresses)
Change Device ID
Change Baud
Backup Settings
Set Mode Flags
Store Program

The non-volatile memory has a specified life of 100,000 cycles, so none of the above commands should be executed in a tight loop. In addition, it takes the microprocessor 10 ms to store data into non-volatile memory, so after executing any of the above instructions, the user must wait at least 10 ms before executing another of the above instructions. Otherwise, the device will abort the command and return busy status.

A.2.4 BiasDac Box Common Mode Noise

The outputs of the BiasDac were initially thought to be impressively quiet (~2 mV_{pp} noise in a 100 MHz bandwidth, similar to the Ramper box), but if you looked at the common mode noise, the noise voltages were as large as a volt [Figure A.7(a)]! The output and ground voltages of each BiasDac channel were "whipping" around earth ground in synchrony by several volts and causing common mode currents to flow. The main culprits for this noise were the dc-dc converters supplying power to each output channel. The dc-dc converters use miniature transformers and switch current several hundred thousand times a second to and switching technology and that soak up charge from the input side and discharge the accumulated charge at the output side, repeating the cycle a million times a second. The large transient currents that flow cause noise to radiate and pollute the outputs with noise at the switching frequency of the dc-dc converter and its (sub)harmonics. Removing the dc-dc converters and powering all the output channels with a common linear power supply reduced the common mode noise on the outputs of the BiasDac by an order of magnitude to several tens of mV. The ~40 mV_{pp} common mode noise was still judge high and further tinkering revealed that the source of the remaining common mode noise (at ~60 MHz) was the interconnect cable ("master/slave" cable) between BiasDac boards. The interconnect cable slaves the clocks of the boards together and shares the input data. The slaving of the clocks was supposed to reduce low-frequency noise by avoiding small differences in their clocks, but the 30 MHz clock being transmitted across the interconnect cable was radiating and causing problems of its own instead. The solution was to keep the clocks local and transmit data between different boards by fiber optic cable. This reduced the common mode noise by another order of magnitude to <5 mV_{pp}, comparable in size to the differential mode noise, and was finally judge acceptable [Figure A.7(b)].

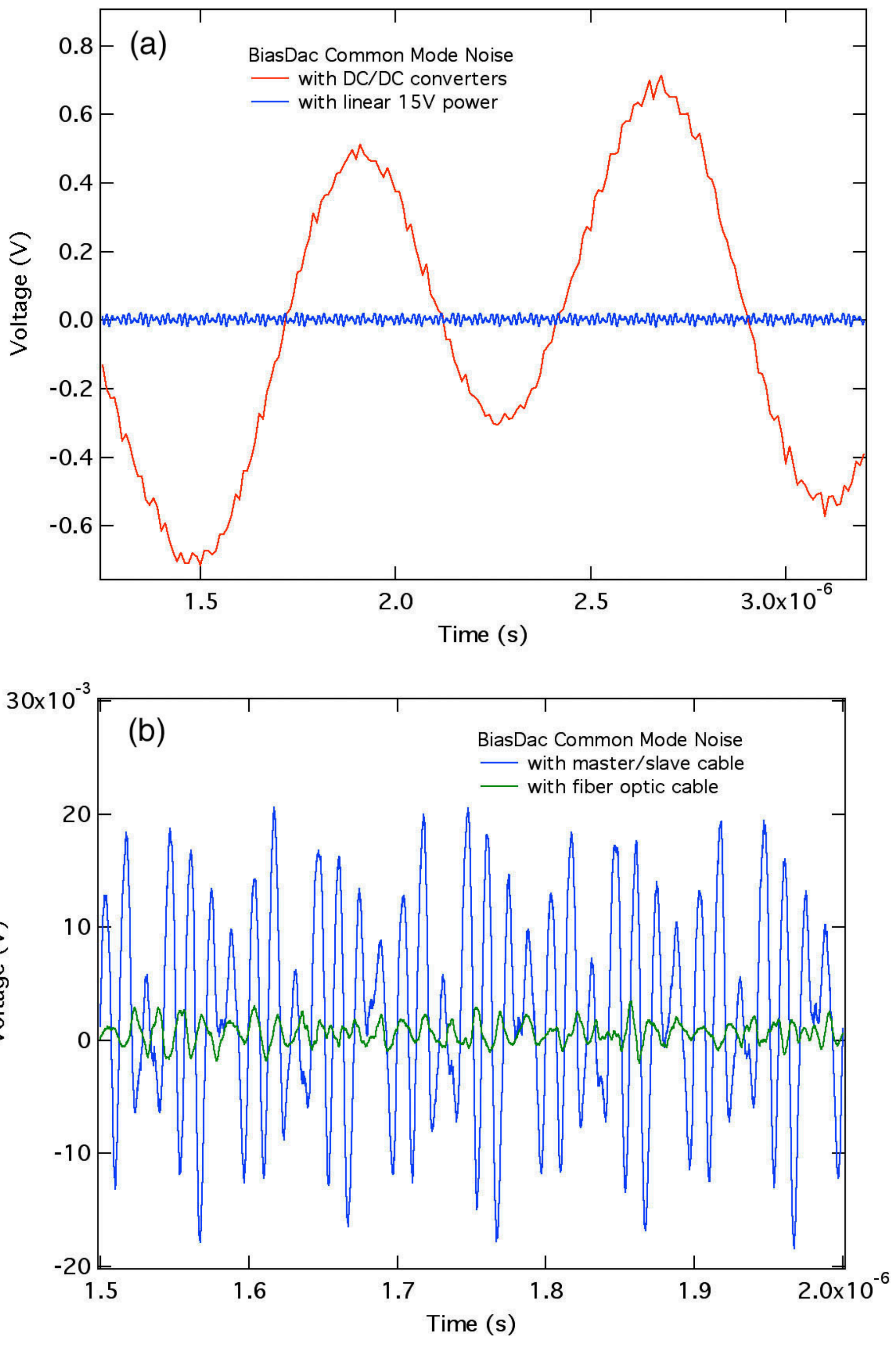
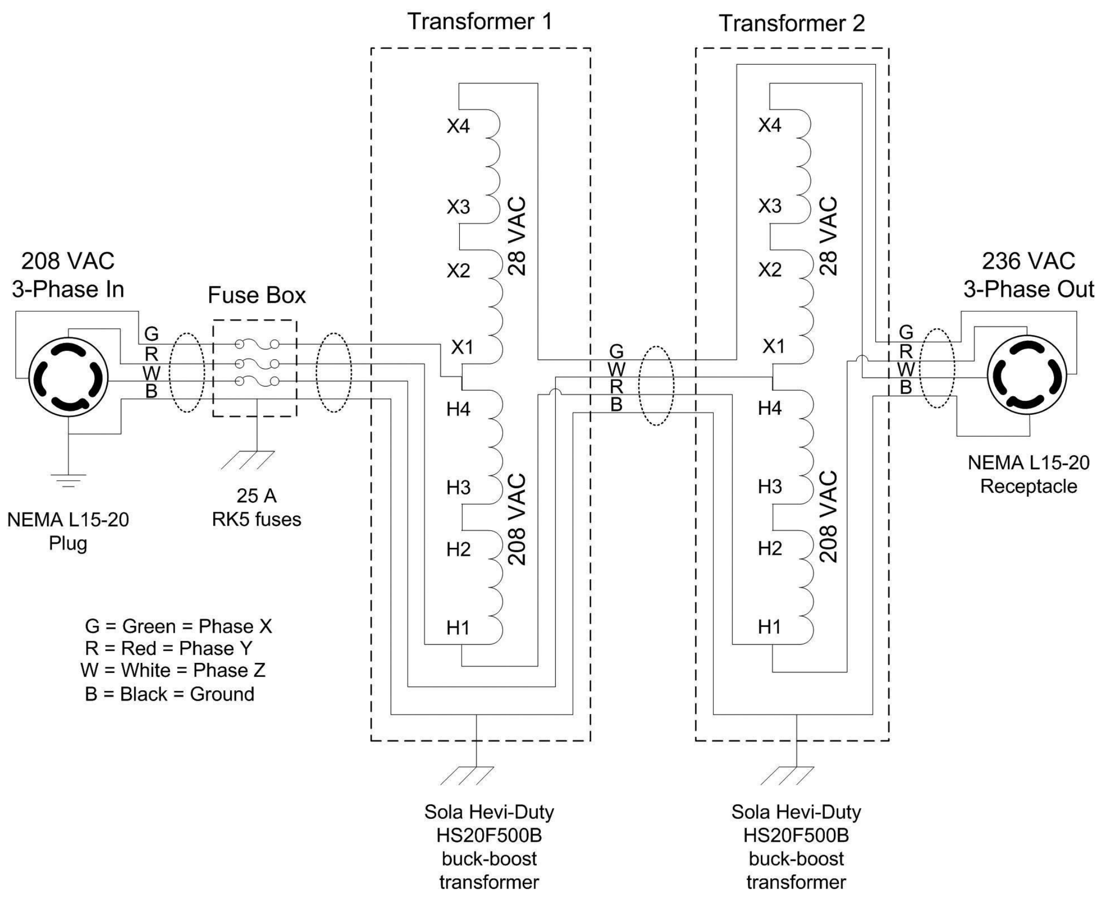


Figure A.8 Data on the common mode noise from the BiasDac. (a) The graph shows the common mode noise when the BiasDac channel is powered from DC-DC converters (red) or a linear power supply (blue). (b) The graph shows the common mode noise when two BiasDac boards are connected together with the master/slave cable (blue) or with fiber-optic cable (green).

A.3 Boost Transformer for He3/4 Rotary Pump

The Pfeiffer Duo 065DM sealed rotary-vane pump takes over from the Leybold Trivac D65B pump (after it died) as the He3/He4 circulation pump for the Kelvinox 100 dilution fridge located in McKay 210-212. Due to its unusual sealed design and magnetically-coupled motor, the only power source available for it is a 250 VAC 3-phase motor standard in Europe. In the US, the standard voltage is 208 VAC 3-phase, so the voltage needs to be stepped up to power the pump. The most efficient way to achieve this is to use a boost transformer. A boost transformer is essentially a normal transformer with the secondary winding connected in-phase with the primary winding to boost the overall voltage. This configuration is also known as an autotransformer and provides no isolation between the primary and secondary windings of a transformer. In order to boost all three phases, only *two* transformers are required; in boosting two sides of the three phases, the third side automatically becomes larger as well (Figure A.9). Due to the higher than normal line voltages in McKay 210-212 (127 VAC vs. 120 VAC), 208 VAC to 236 VAC boost transformers were chosen, making sure that they had enough output current to meet the peak current draw of the Duo 065DM (15 A). The circuit diagram is shown in Figure A.9 and a picture of the transformers is shown in Figure A.10. The measured output voltage of the boost transformer is exactly 250 VAC, 3-phase, and it has been working flawlessly for the past 3 years. The input is fused with 25 A RK5 type fuses, and standard NEMA twist-lock power plugs and receptacles are used for power mating.



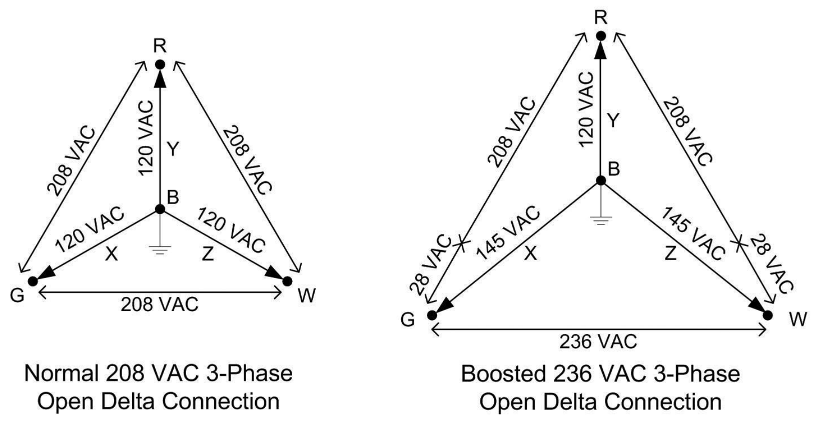


Figure A.9 The top circuit diagram shows the wiring of the two buck-boost transformers to produce 236 VAC, 3-phase output from 208 VAC, 3-phase input (the actual output is 250 VAC due to the higher line voltage in the lab). The bottom diagrams illustrate the boosting action of the two transformers, and how boosting two sides of the triangle also boosts the third side.

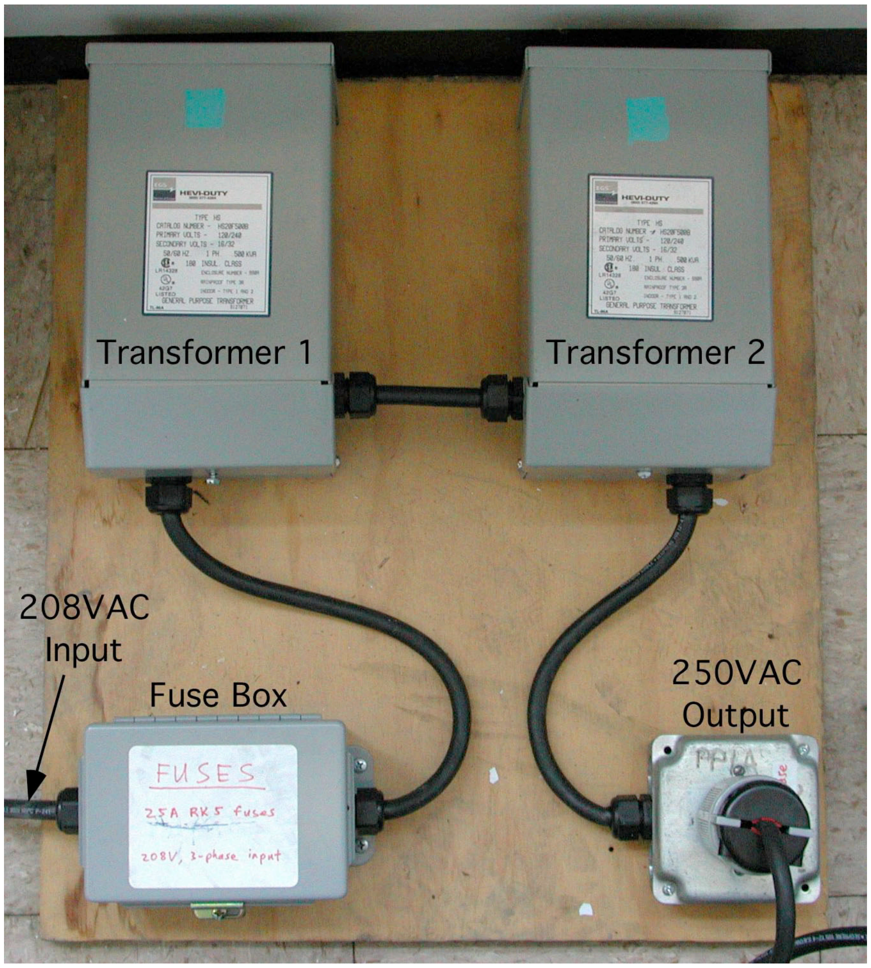


Figure A.10 Photograph of the boost transformers for producing 250 VAC 3-phase power for the Pfeiffer Duo 065DM rotary pump in McKay 210-212. The fuse box at left contains three 25 A RK5 fuses for each phase of power. The rotary pump plugs into the NEMA receptacle at right.

A.4 Poisson Solver

A.4.1 Poisson Solver Classes

The two simplest C++ objects employed in the core code are `ipoint` and `fpoint`, which are instantiations of the C++ template class `vector`. `ipoint` and `fpoint` are used to represent three-dimensional vectors using integer and floating point representations respectively. `ipoint` represents a coordinate in the simulation world while `fpoint` represents the real-world coordinate. These objects, together with `mesh3d`, allow me to accept user input in real-world coordinates while building the simulation world in integer coordinates. This is a big user interface improvement because increasing the size of the simulation world involves changing the step size only, and not the (integer) dimensions of every object in the simulation world as required before.

Another simple object is `flist`, a list of floating point numbers. This object is used to store parameters of a simulation other than potential, like the dielectric constant of a dielectric or depletion electric field of a 2DEG. This is done in the interest of saving space. Instead of storing a floating point (4 bytes) at every point in the simulation world, an index number (2 bytes) is used to reference the actual parameter in the `flist` object instead. When dealing with a million points (100×100×100) or more, this represents a large amount of memory, even on a modern computer.

Every point in the simulation world consists of a mesh point, or `mpoint`, object. This object stores the potential at that point, the type of material it is (*e.g.* dielectric, metal, 2DEG), and the index of any other parameter it needs in the `flist` object. Using a floating point for the potential, and short integers for the type and index, a total of 8 bytes is consumed for each `mpoint`. In a 200×200×200 simulation world, this consumes 64 MB of memory.

Building on the `mpoint` is `mesh3d`, the object that actually contains the three-dimensional mesh of `mpoints` constituting the simulation world. It also contains data about the corresponding real-world coordinates as well as the step size of the simulation world. A lot of the functions for building things in the simulation world (*e.g*. metal boxes or dielectric cones) are coded here. This object also performs coordinate transformations between real and simulated worlds.

The `poisson` object is a child class of `mesh3d` and is where all the logic for relaxing the mesh is coded. The crucial function `updatePoint()` contains all the logic for relaxing each `mpoint`, including types of metal, dielectric, and various boundary conditions. The `poisson` object also keeps track of the largest error on each pass, contains functions to compute charge in a plane ($q = \int \nabla\varphi \times \boldsymbol{da}$), and functions to implement multiprocessing (red-black relaxation). `adaptive` is a child class of `poisson`, and the additional feature it implements is the ability to embed a detailed mesh inside a coarser mesh. This embedding allows the central detail of the world to be computed in high resolution while extending the boundaries of the simulation world to greater distances at coarser resolution. It does this by keeping track of two `poisson` objects, one for the detailed mesh, and one for the coarser mesh. The embedding can be recursive so there is no limit to the amount of embedding allowed (save memory). The `adaptive` object contains code to transfer information between the fine and coarse `poisson` objects to allow them to relax as one. The potential at the boundary of the fine world is defined by the coarse mesh on each iteration, while the potential at the penultimate boundary of the fine world defines the corresponding potential on the coarse mesh on each iteration. In case the coordinates between fine and coarse meshes do not match exactly, a bilinear interpolation of potentials is used.

A.4.2 Poisson Solver Commands

Controlling the action of the program at the top of it all is the code written in part by Mark Topinka, which consists of a script interpreter in which the user tells the program what to do, and a display that shows the potential through a slice of the simulation world. A list of commands and their explanations is listed here:

Commands	Explanation
`set.world` *`x0 y0 z0 x1 y1 z1 dx dy dz (n)`*`;`	Creates mesh and world coordinate system, starting from (*x0, y0, z0*) and ending at (*x1, y1, z1*) in steps of (*dx, dy, dz*). (*n*) is optional maximum detail. Final mesh size is 2^n times finer than normal detail (*n*=0).
`set.boundary` *`0/1`*`;`	Sets world boundary to Dirichlet (*0*) or Neumann (*1*) conditions (default = *0*). Dirichlet ⇒ potential = 0. Neumann ⇒ perpendicular electric field = 0.
`set.dielectric;`	Properly allocates boundary dielectric constants. NOTE: MUST BE CALLED prior to relaxing.
`set.potential` *`potential`*`;`	Sets all dielectrics to potential *potential*.
`reset;`	Clears world with vacuum, and sets Dirichlet boundary conditions. DO NOT CALL unless world has been first set.
`plot.mesh` *`n`*`;`	Plots mesh *n* (default = 0)
`plot.plane` *`x0 y0 z0 x1 y1 z1`*`;`	Defines the portion of the world to plot on screen. Either *x0=x1*, *y0=y1*, or *z0=z1*.
`plot.range` *`0/1`*`;`	Plot with plotplane (*0*) or global (*1*) min & max defining the grayscale. (default = *0*)
`plot.mult` *`factor`*`;`	Size of pixels each mesh point corresponds to (default = 1). If non-uniform step size, it refers to the smallest step size.
`metal.box` *`x0 y0 z0 x1 y1 z1 potential`*`;`	Write a metal box from (*x0, y0, z0*) and to (*x1, y1, z1*) at potential *potential*.
`dielectric.box` *`x0 y0 z0 x1 y1 z1`* `ε;`	Write a dielectric box with dielectric constant ε.
`metal.shell` *`x0 y0 z0 x1 y1 z1 potential`*`;`	Write a metal shell.
`efield.shell` *`x0 y0 z0 x1 y1 z1 0/1`*`;`	Write an electric field (Neumann) constraint on the shell. Constrained points look at inside (*0*) or outside (*1*) points to determine electric field. Perpendicular field is set to 0.
`2deg.plane` *`x0 y0 z0 x1 y1 z1 deplete`*`;`	Write a plane of 2DEG that depletes at perpendicular electric field *deplete* "volts"/"meter" (where "volt" is unit of potential, and "meter" is unit of distance). The potential of the 2DEG is fixed at 0 if not depleted. Either *x0=x1*, *y0=y1*, or *z0=z1*. NOTE: a positive electric field will deplete the 2DEG, so gates with positive potentials should be used!
`metal.cone` *`x0 y0 z0 x1 y1 z1 r potential`*`;`	Write a circular metal cone with base at (*x0, y0, z0*) and tip at (*x1,y1,z1*), with base radius *r*.
`metal.ellipse` *`x0 y0 z0 x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4 x5 y5 z5 potential`*`;`	Write a metal ellipse with center (*x0, y0, z0*) and radius vectors (*x1, y1, z1*), (*x2, y2, z2*) and (*x3, y3, z3*). The 3 radius vectors must be perpendicular to each other.

`metal.readpict` *`axis start height volt_scale`*`;`	Creates an extruded metal object from a PICT file (selected at run time) in the plane defined by *axis* (1=*x*, 2=*y*, 3=*z*) beginning at *start* with extruded height *height*. Potential is scaled so that white = *volt_scale* and black = 0. NOTE: Pure black regions of the PICT file are considered "void" and are not filled. The PICT is scaled to match the proportions of the simulated world. The anti-aliasing involved in the scaling may result in unexpected potentials, especially at boundaries. The PICT is first drawn in the Dashboard window, and then read back in. This can result in unexpected behavior if the Dashboard window is obscured.
`dielectric.readpict` *`axis start height dielectric_const`*`;`	Creates an extruded dielectric object from a PICT file (selected at run time) in the plane defined by *axis* (1=*x*, 2=*y*, 3=*z*) beginning at *start* with extruded height *height*. Dielectric constant is set to *dielectric_const* regardless of color in this case. NOTE: Same as `metal.readpict`.
`relax.iter` *`iterations sor_factor`*`;`	Relax mesh *iterations* number of times with SOR factor *SOR_factor*.
`relax.acc` *`accuracy sor_factor`*`;`	Relax mesh till specified accuracy is reached. A maximum of 1000 iterations are done to avoid infinite loops. With single-precision floating points (4 bytes), an accuracy of not better than ~10^{-7} can be obtained.
`save.planes` *`filename dirn r0 r1`*`;`	Save mesh in the planes of constant *dirn* (1=*x*, 2=*y*, 3=*z*) into *filename* from range *r0* to *r1*. Filenames will be of type "*filename.xxx*" where "*xxx*" is the mesh coordinate number. NOTE: When reading files into Igor, the axes are transposed.
`charge.shell` *`x0 y0 z0 x1 y1 z1`*`;`	Compute free charge on surface of metal shell.
`charge.plane` *`x0 y0 z0 x1 y1 z1 dirn`*`;`	Compute free charge on metal surface with normal pointing in *dirn* (1=*x*, -1=-*x*, 2=*y*, *etc*...).
`charge.shell2` *`x0 y0 z0 x1 y1 z1`*`;`	Compute "free" charge on surface of metal shell using Simpson's rule rather than trapezoidal rule. NOTE: MUST MULTIPLY RESULT by correct dielectric constant to obtain final answer.
`charge.plane2` *`x0 y0 z0 x1 y1 z1 dirn`*`;`	Compute "free" charge on metal surface with normal pointing in *dirn* (1=*x*, -1=-*x*, 2=*y*, *etc*...) using Simpson's rule rather than trapezoidal rule. NOTE: MUST MULTIPLY RESULT by correct dielectric constant to obtain final answer.
`mesh.iter;`	Returns the number of iterations performed so far, in the line normally showing the error.

`mesh.error;`	Returns the max error.
`mesh.set n;`	Sets the active mesh to mesh *n* (default = 0).
`mesh.insert n;`	Inserts mesh *n* into the current mesh (for “adaptive” step size relaxation).
`mesh.detail n;`	Set mesh to detail *n* (0 is lowest resolution).
`mesh.initDetail;`	Initialize current resolution mesh from lower resolution mesh.
`mesh.average;`	Calculates average potential of all dielectrics in the world.

A typical program involving extruded parts and an embedded mesh is listed here:

```
set.mesh 0;
set.world -150 -150 -30 150 150 30 2 2 .5;

dielectric.box -150 -150 -30 150 150 0 13;
2deg.plane -150 -150 -5.5 150 150 -5.5 0.055;
dielectric.readpict 3 -3 3 1;
dielectric.readpict 3 -6 1 13;
metal.readpict 3 0 2.5 0.75;

set.dielectric;

plot.mesh 0;
plot.mult 3;
plot.range 0;
plot.plane -150 -150 0 150 150 0;


set.mesh 1;
set.world -300 -300 -60 300 300 60 4 4 1;

dielectric.box -300 -300 -60 300 300 0 13;
2deg.plane -300 -300 -5.5 300 300 -5.5 0.055;

mesh.insert 0;
set.dielectric

relax.acc 1e-3 1.6;
relax.acc 1e-6 1.9;

set.mesh 0;
save.planes dilbert 3 -6 -5;
```

A.4.3 Issues at Dielectric Boundaries

One of the hardest conceptual problems encountered in writing the Poisson Solver was trying to decide what each mesh point in the simulated world represented. It seemed like the potential value and the material property (*e.g.* dielectric constant) should be easily represented by each mesh point, but in fact the potential and material meshes are not coincident! There should in fact have been two meshes, as illustrated in Figure A.11 for the two-dimensional case. The material mesh should be formed from the crosses (×), while the potential mesh should be formed from the circles (•). This is because a potential mesh point (•) at the interface of Dielectric 1 and Dielectric 2 would not know what dielectric value it should have, as shown in the following paragraph.

Now, the new value of a potential mesh point is the weighted average of its nearest neighbors:

$$\varphi_0 = \frac{\sum \varepsilon_i \varphi_i}{\sum \varepsilon_i} \tag{A.1}$$

Consider potential mesh point φ_1 (see Figure A.11). One of its nearest neighbors is φ_0, and φ_0 should clearly be weighted by ε_1 because the space separating φ_0 and φ_1 is filled with Dielectric 1. However, consider point φ_2. φ_0 is also a nearest neighbor to φ_2, but in this case φ_0 should be weighted by ε_2 because the space between φ_2 and φ_0 is filled with Dielectric 2. Even more perplexing perhaps is what should happen between φ_3 and φ_0. In this case, the average dielectric constant $(\varepsilon_1 + \varepsilon_2)/2$ should be used. Thus you can see from this example in two-dimensions that a potential mesh point can have up to three different "dielectric constants" depending on which nearest neighbor we are considering! Therefore potential and material information cannot share the same point!

Unfortunately, since the program was already implemented as a single mesh of potential and material information, a workaround had to be improvised. Dielectrics are classified into "plane" and "edge" points in two-dimensions ("bulk", "plane", and "edge" in three). φ_1 and φ_2 are considered "plane" points while φ_0 and φ_3 are "edge" points. The dielectric constant of each point is then the average dielectric constants of the dielectrics surrounding it, so φ_1 has dielectric constant ε_1, φ_2 has dielectric constant ε_2, and φ_0 and φ_3 have dielectric constants $(\varepsilon_1 + \varepsilon_2)/2$. When calculating the appropriate weighting between two points, the dielectric constant of the *higher-dimensioned* point is used. For

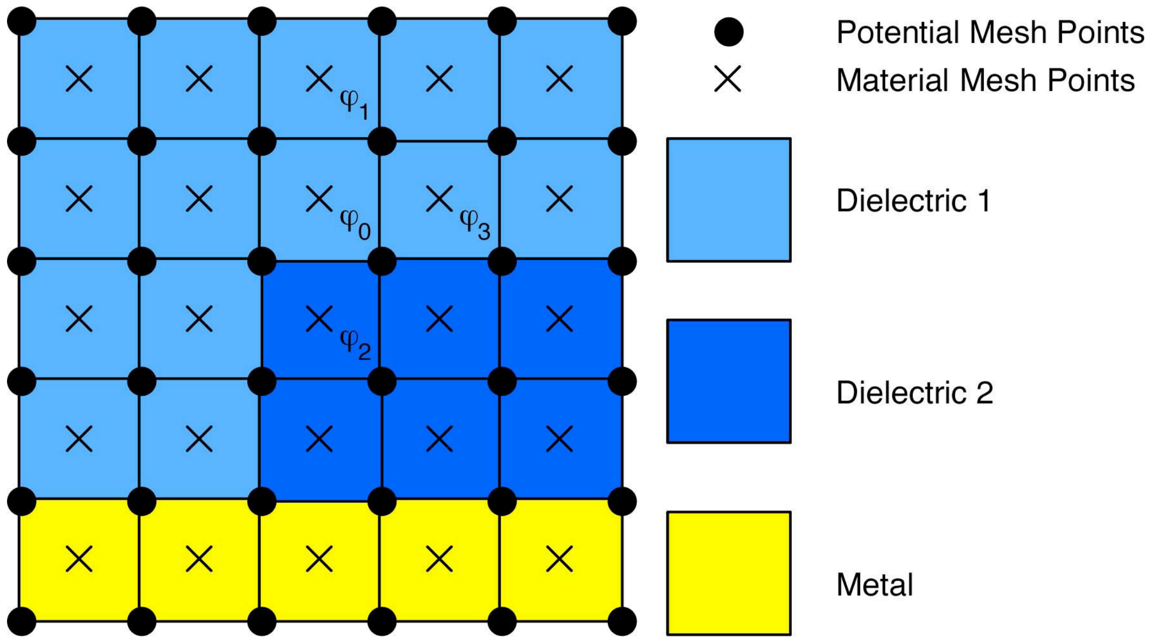


Figure A.11 Figure illustrating the need for two meshes – one for potentials and one for materials – when performing electrostatic simulations. It also illustrates the difficulties encountered at the boundaries between dielectrics when trying to determine what dielectric constant to use (*e.g.* when relaxing potential mesh points φ_0 and φ_3) (see text).

example, when figuring out the appropriate dielectric constant to use between φ_0 and φ_1, the "plane" dielectric constant of φ_1 trumps the "edge" dielectric constant of φ_0 and ε_1 is used. Between φ_0 and φ_3, the "edge" dielectric constant $(\varepsilon_1 + \varepsilon_2)/2$ is used since both are "edge" points. With this extra logic, the correct weights are used in all cases in three-dimensions and results in the correct relaxation at dielectric boundaries compared to Mark Topinka's earlier version. The script command `set.dielectric` must be called to properly calculate the dielectric constants at dielectric boundary points before the mesh is relaxed.

A.5 AFM Program

A.5.1 Main BiasDac Modifications

The AFM program was designed to output voltages to an AFM tip at several tens of kHz. In order to achieve this high speed, all the outputs for a scan line were pre-computed, stored in an array, and then the whole array was sent to the DACs via hardware. Data acquisition was synchronized with voltage output via hardware (RTSI signals between National Instruments cards). The BiasDac, however, has neither a hardware buffer for storing arrays of output data nor hardware connections for synchronization data acquisition, so a significant portion of code in the AFM program (mostly in the file NI.c) needed re-writing for use with the BiasDac. Each output to the BiasDac and each data acquisition point needed to be handled by the computer in software. Fortunately, the data rate required for quantum dot experiments is two orders of magnitude slower than for AFM experiments and software is sufficiently fast to accomplish this. An output rate of 90 Hz is used, limited by the rate at which the Macintosh flushes its serial port.

The array of output voltages was designed as an array of 16 bit integers, which was sufficient to handle the eight 12 bit DACs of the original AFM setup (4 bits of address and 12 bits of data). With the BiasDac however, 8 bits of address are required to handle the potential 244 output channels along with 20 bits of data, so two 16 bit integers are used to represent each BiasDac output (8 bits of address and 24 bits of data). This encoding is handled in the function `BD_WriteOut()` in the file NI.c, and the decoding is handle within the function `NI_DoScan()` in the file NI.c. Only the lower 20 bits of the 24 data bits are used.

The functions `NI_ScanGetSet()`, `NI_UpdateTip()`, `NI_ReturnToOrigin()`, and `NI_SetupXYZ_Back()` (all in the file NI.c) are used to compute scanning movements and were simplified in the BiasDac version to remove movements related to the *Z* motion of the AFM tip.

The program was also modified to perform conversion of voltage to bits at the channel level (as opposed to the global level) using the function `BD_WriteOut()`. This means that each BiasDac channel can theoretically have a different output range and number of bits, so the Marcus lab BiasDac box could be borrowed and chained with our own box, for example, and the mix of output ranges would not confuse the AFM program. No user interface exists for changing the settings of each output (or the number of outputs) however, but it can be implemented if required since all the "plumbing" is already in place.

Z and *V* guide operation was modified to include an offset to the guide voltage $V_G = V_{SW}(x,y) + V_O$, where V_G is the output of the guide channel, V_{SW} is the guide data stored in the scan window, (x,y) are the current *X* and *Y* outputs, and V_O is the guide offset. This offset is recalculated [in the functions `ChangezGuideFlg()` and `ChangevGuideFlg()` in the file AFM.c] whenever the guide is turned on with $V_O = V_{GON} - V_{SW}(x_{ON},y_{ON})$, where V_{GON} is the guide voltage output and (x_{ON},y_{ON}) the *X* and *Y* outputs when the guide is turned on. This is done to remove the sudden change in guide voltage output when the guide is turned on and snaps to the guiding voltage. The guide voltage output can be swept to change its value while guiding is on with the Script command `dac.sweepTo`. The necessary code for keeping track of the guide offset during the sweep is performed in the function `NI_SetVoltage()` in the file NI.c.

Another difference in the BiasDac version of AFM program is that, when launched, it does *not* initialize the outputs to any particular starting voltage. Instead, the program reads in the current BiasDac output voltages and *initializes the program* with those voltages. This is done to "pick up" the BiasDac voltages where they were "left off" in case of a computer crash.

This concludes the main modifications to the AFM program. Every modification in the AFM program code can be identified by searching for the phrase `BIASDAC`.

A.5.2 BiasDac Classes

There are four classes used to implement the interface between the AFM program and the BiasDac hardware: `GSSerialMacOS`, `Dac`, `BiasDac`, and `WABBiasDac`. The `GSSerialMacOS` class encapsulates the Macintosh serial port, and the source code for this class was downloaded from Kyle Hammond's Mac Programming Page:

http://www.cpinternet.com/~em002400/MacProgramming.html.

It shields the user from many nasty details of operating the Macintosh serial port, and was a convenient class to use.

The base class for the BiasDac is `Dac`, which is an abstract class that implements a general voltage output channel. This class stores the DAC ID, output voltage range, DAC resolution, current output value, and software output limits. The DAC ID determines which output channel a `Dac` object belongs to, the output range and resolution data allow it to convert between voltage and bits, and the software output limits allow the user to specify a safe operating range that will not be exceeded. The class, however, does *not* implement any actual communication with hardware.

That is done in the `BiasDac` class, which is a child class of `Dac`. The `BiasDac` class specifically implements a BiasDac output channel. It stores a board ID for determining which BiasDac board an output channel belongs to, and has a pointer to a `GSSerialMacOS` serial port object for communicating with BiasDac hardware. Commands to BiasDac hardware are implemented in this class according to the communications protocol found in Appendix A.2.3, and the reply from hardware is checked for validity.

The class `WABBiasDac` consists of an array of eight `BiasDac` objects and encapsulates the BiasDac box. The board and DAC IDs of each `BiasDac` object default to the following:

Array Index	0	1	2	3	4	5	6	7
Board ID	1	1	1	1	2	2	2	2
DAC ID	0	1	2	3	0	1	2	3

although they can be changed programmatically, as can the size of the array. The output settings of each `BiasDac` object default to ±5 V range, 20 bit resolution, and can also be changed programmatically. The `WABBiasDac` object also has a pointer to the serial port object so that it can write multiple outputs at a time [*e.g.* writing a vector of outputs (v_1, v_2, v_3, v_4)]. This multiple-output capability is required because the serial port on the Macintosh computer can be accessed only 90 times a second. This means that the update rate for a vector of eight voltages written individually to the serial port would be 90/8 = 11 Hz, which would be too slow. With the multiple-output capability, the vector of eight voltages is written to a buffer first before being sent to the serial port, and the update rate of the AFM program can remain at 90 Hz.

A.5.3 Selected AFM Commands

Here is the list of BiasDac-specific commands:

Commands	Explanation
`dac.setLimits` *`outCh v1 v2`*`;`	Sets software limits to the output range of BiasDac channel *outCh*.
`dac.recalibrate` *`outCh`*`;`	Recalibrates BiasDac channel *outCh*. NOTE: The output of the channel goes to random values during calibration; the software output limits may not be respected during that time.
`dac.recalibrateAll;`	Recalibrates all BiasDac channels. NOTE: See above.
`dac.set` *`outCh voltage`*`;`	Sets the output of BiasDac channel *outCh* immediately to *voltage*.
`dac.sweep` *`outCh startV endV stepSize inCh numAvg`*`;`	Starts a sweep of BiasDac channel *outCh* from *startV* to *endV* in steps of *stepSize*. Data is acquired from input channel *inCh* and the result is averaged *numAvg* times.
`dac.sweepTo` *`outCh endV stepSize inCh numAvg`*`;`	Starts a sweep of BiasDac channel *outCh* from its current voltage to *endV* in steps of *stepSize*. Data is acquired from input channel *inCh* and the result is averaged *numAvg* times.
`setPassiveSweep` *`inCh`*`;`	Makes `dac.sweep` and `dac.sweepTo` read input channel *inCh* as the *X* value instead of the BiasDac channel *outCh* (which is still swept). NOTE: Set *inCh* to –1 to disable passive sweeps.
`channel.setZGuide` *`outCh`*`;`	Sets the *Z* guide channel to be BiasDac channel *outCh*.
`channel.setVGuide` *`outCh`*`;`	Sets the *V* guide channel to be BiasDac channel *outCh*.

Here is a list of useful AFM commands:

Commands	Explanation
`setxchannel` *`outCh`*`;`	Sets the X channel to be BiasDac channel *outCh*.
`setychannel` *`outCh`*`;`	Sets the Y channel to be BiasDac channel *outCh*.
`tipspeed=` *`speed`*`;`	Sets the scanning speed while acquiring data to be *speed*.
`flyback=` *`speed`*`;`	Sets the scanning speed while not acquiring data to be *speed*.
`squarepixels` *`on/off`*`;`	Constrains the number of pixels in *X* and *Y* to be the same if *on*.
`setnpts` *`numX (numY)`*`;`	Sets the number of pixels in a scan to be *numX* square or *numX* by *numY* (see above).
`setfastdir` *`dirn`*`;`	Sets the fast scan direction to *dirn* (l2r, r2l, t2b, or b2t).
`setslowdir` *`dirn`*`;`	Sets the slow scan direction to *dirn* (see above).
`graphinchan` *`graph inCh`*`;`	Sets the input channel of scan window *graph* (a–d) to be *inCh*.
`graphonoff` *`graph on/off`*`;`	Turns the input to scan window *graph* (a–d) *on* or *off*.
`movecursors` *`cursor x y`*`;`	Moves cursor *cursor* (r, g, or b) to position (*x*, *y*).
`startscan;`	Starts a scan.
`zguide` *`on/off`*`;`	Turns the *Z* guide *on* or *off*.
`vguide` *`on/off`*`;`	Turns the *V* guide *on* or *off*.
`path=` *`pathName`*`;`	Specifies the directory in which to save files as *pathName*.
`scan.savenext` *`graph +/0 fname`*`;`	Saves the data in scan window *graph* (a–d) to filename *fname*. + increments the file counter while *0* does not.
`sweep.savenext` *`+/0 fname`*`;`	Saves the data in the sweep window to filename *fname*. + increments the file counter while *0* does not.
`constantscan` *`graph value (nx) (ny) (xWidth) (yWidth)`*`;`	Creates a flat plane in scan window *graph* (a–d) of height *value*. *nx* and *ny* optionally specify the number of pixels of the plane, and *xWidth* and *yWidth* optionally specify the range of the plane.
`copyintofrom` *`graphIn graphFrom`*`;`	Copy data from scan window *graphFrom* (a–d) to *graphIn* (a–d).
`printscanstats` *`graph`*`;`	Displays settings for scan in scan window *graph* (a–d).
`help` *`command`*`;`	Displays help (if any) for command or command fragment *command*.