UNIVERSITY of CALIFORNIA
Santa Barbara

Computing prime factors using a Josephson phase-qubit architecture: $15 = 3 \times 5$

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Physics

by

Erik Anthony Lucero

Committee in charge:

Professor John M. Martinis, Chair
Professor David D. Awschalom
Professor Wim van Dam

June 2012

The dissertation of Erik Anthony Lucero is approved:

Professor David D. Awschalom

Professor Wim van Dam

Professor John M. Martinis, Chair

June 2012

Computing prime factors using a Josephson phase-qubit architecture:
$15 = 3 \times 5$

To my parents: Mary and Anthony. My sisters: Michele and Sheila, and mi familia: Offs and Luceros.

Acknowledgements

First and foremost I would like to thank Professor John Martinis for providing me a creative environment complete with outstanding people, project, location, and what felt like an infinite supply of resources. I feel fortunate to have come through the lab during a time when the group was small -just beginning to grow- and John spent a good portion of his time in the laboratory. This overlap with "in-the-lab-John" was key to learning when to be an engineer, when to be a physicist, and when it is time to lead a climb. In John's words, "you just do it" and "it will be like an afternoon's worth of work" is a good sign that you have encapsulated an experiment with "nice data" right around the corner. It would be a laundry list of items if I were to list all of the skills, knowledge, soldering and climbing techniques that John has imparted to me so instead I'll say that after six years I still love doing physics, I have climbed some epic routes, I know how to negotiate a free oscilloscope, and most importantly, I know "how to make things". Thank you John.

Next, of course, I would like to thank my parents, Anthony and Mary Lucero. Dad, it was your dedication to finishing projects (saying that you will do something and then doing it, both with class and modesty) and always including me in their execution that I have carried with me throughout my life and applied in my career. Against my delicate constitution my dad convinced me to get up early on Saturday mornings to get to the lumber yard so we would have a full weekend to crank on our latest project. I don't know how many weekends (and it is probably best not to reveal so as not to scare any of the new graduate students) I drew on those experiences to get into the lab and push through my experiments. Mom, it was your endless supply of supportive letters and good, no excuse me, great-will towards everyone you have ever met that has helped teach me to treat every person with respect and approach all of my situations in life with a positive attitude.

And my sisters, Sheila and Michele. Sheila, thank you for your insistence that I come out to Santa Barbara. I will never forget our road-trip. I have always looked up to you and your professionalism. It was your home-cooked meals at Jax that forever changed my life and gastronomical expectations. "Spoiled", I have heard others say. And I would agree. Our connection through food, music, and our careers have helped me stay grounded, cook when I needed to, focus on the job, and excel at what I am committed to. By the way, the results are in on our ongoing competition of who can can work the longest work week...week after week after week ... you win. I am so proud of you and lucky to have you as one of my best friends and sister. If Sheila is my yang, than Michele is my yin. Michele has provided the feminine balance to my otherwise very testosterone driven life. Michele your maternal tendencies (timely care-packages) helped me

push through countless finals weeks, tests, presentations, and any other stressful career milestone. Your dedication to the family is the glue that keeps all of us close and your stress is what fuels all of our ambitions and achievements. I love you both and I would not be half-the-man or scientist without you two.

My departure from the UCSB Quantum Computing group marks the end of a paradigm. I joined the team with the first set of graduate students and I will be the last one who will have overlapped with all (except Ken Cooper) of the then Post-Docs that came to work with John and Andrew and who are now Professors or group leaders elsewhere in the world. Therefore, I would like to take this opportunity to thank all of the people that I have had the pleasure of learning from and without which I would not have been able to even propose, let alone accomplish what I have done.

Robert McDermott. Robert's professionalism in the laboratory, and ability to hack together a key experiment in an afternoon has been and will continue to be something that I strive to emulate. And I will never forget his endless supply of missions that were always in need of more warm bodies. And of course, I hope to some day be able to evoke a profound superconductivity theory mid group meeting to explain an experimental mystery, which he so often raddled of with ease.

Matthias Steffan. I have always been impressed with Matthias' cool demeanor even in the face of all kinds of challenges political and scientific. I am more than fortunate to have a mentor and friend like Matthias on my side reminding me to enjoy the good things in life, like "bits-bits-bits". I can't wait to see what we pull off together at IBM.

Eva Weig. It was your cleanroom ethos that carried through the group project after project even after you departed. We learned to never give up, but persevere even in the face of all kinds of experimental obstacles.

Nadav Katz. I admire Nadav's courage and ability to jump fields and prove himself all over again. And as much as I prefer the experiment over any simulation it was Nadav who taught me the importance of doing both.

Although success does bring new titles, like Professor, I am honored to have worked with Haohua Wang when he was known affectionately as "The Doctor". I am (along with everyone in UCSB QC-group) forever indebted to Professor Haohua Wang's cleanroom skills and all of the devices that Haohua fabricated. I am fortunate to have learned even just a small fraction of these skills from one of the cleanroom masters.

Max Hofheinz. T-Crack will remain as legendary as the time we overlapped both on the crags and in the lab. I owe Max a number of serious climbs out in the Colorado wilderness to repay him for my improved microwave engineering skills and mantle moves. Not to worry, I will be sure that the climb is "6.4 J-tree" level

crack climb.

Martin Wiedes. You are a lucky man. You managed to live in the two best places in the world, Santa Barbara, California and Boulder, Colorado. Well done. Thank you for the social outlet, volleyball on the beach, group beer moments, and for supporting a starving artist, eh graduate student.

Yi Yin. Thank you for the camaraderie, as we fabricated qubits through the many nights in the cleanroom together. Those back-to-back overnight fabribaction sessions were important to building both of our characters and devices. I imagine that we will both look back on that experience fondly (particularly when we are sharing it with those who are going in to the cleanroom to do the battle for us).

Yu Chen. Thanks to our favorite liquid helium vendor, every Thursday Yu was forced to warm up the fridge and start his experiments all over again. I think I would have derailed. However, it was your level-headedness that prevailed and what I would pull from when my 1k-pot plugged over and over in Jules. It was a pleasure sharing the lab with you and learning to keep a positive perspective despite all of the uncontrollable setbacks we suffered in the laboratory.

Matteo Mariantoni. I appreciate you carving out the time in your over booked schedule to help with the phase calibrations, a.k.a "the cows". I especially look forward to reconnecting and partaking in some of your home-cooked Italian food.

Rami Barneds. By any measure we had the best office. Whether it was the low amounts of infrared radiation, or absence of magnetic field, either way we were able to hash out some politically correct discussions in our light-tight magnetically shielded room. I really appreciated the life after PhD discussions. Here is to submitting a paper, boarding a flight, and dilutionary refrigerators in closets.

There are such wonderful and unique opportunities in one's PhD and being a RIKEN fellow in Japan marked one of those for me. I will never forget Professor Franco Nori and his "pointed" questions (literally he sat in the front row with his own LASER pointer, his green, mine red). Franco asked question after question in rapid fire succession that stretched a one hour talk into four. Great training for any physicist.

It was in Japan where I first met Tsuyoshi Yamamoto, while visiting Yasunobo Nakamura's group at NEC. Later I would learn that it was this trip and our discussion of our custom electronics that would be the catalyst for his one year visit to the Martinis group. It was quite an honor to have Tsuyoshi out in Santa Barbara and I hope that I will someday be as scientifically organized as Yamamoto-san.

Sahel Ashab. Together we proved that even a theorist can make stuff with their own bare hands: a barge in the shape of a catamaran with custom shaped coolers by Markus, Sahel, and Erik. Of course it didn't sink and Markus and I

couldn't have done it without you. Your visits were like clockwork, every summer I could count on you showing up at our doorstep (Madrid house, The Mesa House, and Broida) ready to discuss physics. And what an amazing experience to be invited out to RIKEN in Japan? Thank you for being such an accommodating host, reliable friend, novice boat-maker, and great scientist. Pieter de Groot and I still need to get back to Japan for a focused conference on qubits and Japanese culture.

Professor Martin Huber. I would not even be a physicist if it were not for Martin. Whenever I even think that I am busy I remember that Martin is somewhere, somehow juggling a larger and more complex Hilbert space than anyone I know and doing it with class. Strapped for resources and still making amazing contributions to science, Martin you are an inspiring scientist and human.

Mark Howard. There is no way I would have survived the first two years at UCSB, let alone all six if it weren't for Mark. From pasta-bakes for homework sessions that stretched through the night and on until the next day, to sleeping in the desert for Daft-Punk and Tool, to late-night pickups of the Irish boys and the know-how to properly close a pub. I can always count on Mark for the correct perspective to any problem (physics or life) and reliably avant-garde media to consume.

Markus Ansmann: The catamaran project and all our associated mis-adventures with Chuck Barfoot, Shirley-Joy-Vivian-Ms. Hall. We always improved upon our "method" related to our wild-projects, which from what I can measure culminates in earning a "wizard-degree". You enabled the entire group to produce amazing experiments with LabRAD. And I am fortunate to have such a stand-up friend and goto genius on my side, especially one with such a keen set of taste-buds and class.

Matthew Neeley: Everything Matthew touches turns into a python program, an elegant well written one at that. Every time I look at Matthew's code I learn something after unpacking the one- or two-liners. Matthews' code serves as a stand-alone guide on how to program, eh, how to think. Matthew and Markus birthed LabRAD a versatile code base that has lived beyond their times in the lab. Its impact reverberates within the group as each new student or post-doc finds themselves adding to the ever growing "Pyle" repository and reaping the benefits from all the alums. Our group's success is a testament to both of your minds and ability. Legendary. I am honored to be considered one of Matthew's peers. Thank you for all the support while you were here in California (Madrid and Mesa house) and from afar. I'll never forget the (barefoot!) runs (March Meeting, Montecito Peak, and Jesus-its-a-fire) the rides (AIDS Lifecycle), and important life discussions.

Daniel Staudigel: To an amazing friend, incredibly mature and balanced hu-

man being who is full of life, integrity, intellect, and wild-crazy physical ability. So many, alright probably all, of my boundaries have been pushed thanks to Dan especially gastronomically. I would be honored to cook with and in Dan's kitchen any day, anywhere in the world. Dan brought a balance to my graduate life via excess hedonistic endeavors in food and exercise through great coffee, countless over-indulgent meals (cave-man chicken, "light-heavy-cream sauces", home-made smoked pork-belly, and the menu goes on and on . . . Let's Get Fat!).

Radek Bialzack: Doctor, Doctor. PhD, MD! I wish your unwavering hard-work ethic could rub off a little on everyone in the world. Humankind would be in a better place because of it. Radek's devote altruistic motives and hard-work never ceased to amaze me. I look forward to your success in synthesizing physics and medicine and you saving the world.

Aaron O'Connell: I knew Aaron back when...back when we both were pushing each other to stay through the night in the cleanroom to complete our fabrication. I applaud you for going after what you want and then continuing to tack against the winds to success.

Daniel Sank: You have come a long way my friend and I am impressed to see the wizard-skills solidifying within you. I am looking forward to discussing the tasting notes of your custom ales. Seriously Daniel, it was great training you in the cleanroom, working with you in the lab, and performing electronics sorcery abroad.

Jim Wenner: I remember the first time you uttered the coveted Martinis-lab colloquialism as we ventured out and about during your first March meeting. At that point it was clear you were integral to the lab . I think it is safe to say that I have never met anyone quite like James Wenner before -the man with an endless supply of puns. The success of the lab (or any lab that is lucky enough to hire you next) hinges on your keen attention to detail and your finger on the pulse of the lab (and the plumbing that plumbs it).

Mike Lenander: Don't let Bacula (or the Physics PhD) suck the life out of you.

Julian Kelly:"Pain is temporary, but failure is forever." I imagine that a superstar like you may never even feel the pain. You are a talented individual Julian and I am thrilled (for everyone's sake) that you stayed in Santa Barbara. I am looking forward to $30\mu s$ T_1.

Ted White: Thanks for stepping up and keeping the Fastbias' up-to-date and right on time.

Peter O'Malley: Peter, although we did not overlap on projects it is good to know that you will be leading the quantum revolution along with Pedram Roushan.

Amit Vainsencher: Thanks for taking the administrative lead (along with Pe-

ter) on all the IT related projects in the lab. These are thankless jobs and will land you the honor of being the scapegoat once Skynet becomes self-aware. All I can say is good luck with that.

Anthony Megrant:"The Million-Q-Man". Enough said...or maybe three more papers worth. I'm looking forward to more MBE breakthroughs.

Jian Zhao: The man behind the bunny-suit endlessly servicing the Lesker so that all of us lowly graduate students could muck it up again. Thank you Jian. Thank you for all your hard work in the cleanroom and for being such a stand-up guy.

Professor Andrew Cleland: Thank you for recruiting and helping to convince John to come out to Santa Barbara. You two have created an amazing empire together and it has been a pleasure learning from you both. Thank you for attracting great people to work with and providing the resources necessary for our group's success. I appreciate your door literally always being open when I needed it to be -clutch moments on the fridge- like when the 1K-pot had plugged and I needed to consult on data before my experiment warmed up. Thank you for enabling my scientific contributions.

My PhD Thesis committee: Professor David Awschalom and Professor Wim van Dam. Thank you for setting aside time in your over booked schedules (even four years ago when I proposed this thesis idea) to provide your professional advisory support. I can't thank you both enough for accommodating my ever moving target end-date. And David, thank you for all of your hallway discussions, which always seemed to offer the correct perspective on my career trajectories. It was a pleasure observing how you can inspire new generations of scientists in and out of the classroom and laboratory. I look forward to the day when I am capable of doing the same.

The UCSB California Nanosystems Institute (CNSI) staff: Especially Holly Woo for always providing the desperately needed encouragement throughout my tenure. Thank you for your support of my scientific and (artistic) photographic passions. Dan Daniels, Lynne Leininger, Eva Deloa, and Bob Hanson, and of course the rest of the staff whose work behind the scenes made my graduate life that much smoother.

Doctor Fiona Goodchild: What a pleasure it was to learn from you and teach with you in the Practice of Science class. It was such an honor to be part of the Dr. Goodchild and Professor Awschalom dream team. I look forward to applying the teaching and mentoring skills I learned from both of you to continually excite new generations of scientists.

The Physics department staff: Especially Dave Prine. Thank you for your forever positive attitude and tireless negotiations with liquid Helium vendors to secure "clean" liquid Helium so that I could execute my experiments. And every-

one in the Physics machine shop. Thank you for your timely and beautiful work. Thank you Mike for having the patience to impart your knowledge to all of us "green" graduate students year after year. And Jennifer Farrar, thank you for always checking in with me about my ever-changing graduation date.

And I would like to express my appreciation to everyone on staff at the UCSB cleanroom. I feel very fortunate to have learned and worked in such a world-class facility with totally professional people that strived to make it that way. Thank you all for your endless servicing, training, and development year after year, day after day, and in most cases, night after night.

Stacie Furia: For the countless home-cooked-foodie meals that helped me (and Matthew!) push through some long and drawn out laboratory nights and weekends. And for "discovering" the photographer in me -you were the first.

Stephanie Ma: Thank you for believing in me and sticking with me through this stressful journey. I'm looking forward to "Our Rendezvouses". Here is to piñatas, succulents, epic sunsets, your relentless positive outlook on life, and you.

To all those who managed to come out and visit me in Santa Barbara through the years, Omid Masihzadeh, Paul Nied, Buddy Jerome, NaYoung Pak, Sean, Phil, and Patsy DeDycker. I know it was difficult making the time to come out and I am beginning to understand how difficult it is to leave Santa Barbara. Thank you all for the countless phone calls filled with support and great advice. I am so fortunate to have a strong network of friends that are all family.

All of you have been my giants.

Curriculum Vitæ

Erik Anthony Lucero

Education

2012 (exp.)	Doctor of Philosophy, Physics, University of California, Santa Barbara
2008	Master of Arts, Physics, University of California, Santa Barbara
2005	Bachelor of Science, Applied Physics, Magna Cum Laude, University of Colorado
2005	Bachelor of Science, Electrical Engineering, with Honors, University of Colorado

Honors and Awards

2009	Best Student Presentation, jointly awarded by The Institute for Quantum Computing, University of Waterloo and The American Physics Society Quantum Information Topical Group
2006-2009	GAANN Fellow, University of California, Santa Barbara and US Department of Education
2007	Digital Materials Laboratory Fellow, RIKEN, Wako, Japan
2005-2007	Broida Fellow, University of California, Santa Barbara
2005	Certificate of Merit Colorado Engineering Council, Finalist for the Silver Medal
2004-2005	Ben Trujillo Scholar
2003-2005	McNair Scholar
2003-2004	Phyllis Weisheit Schultz Scholar
2002-2004	William R. Simmons Scholar

2001-2004 National Institute of Standards and Technology, Professional Research Experience Program Fellow, Boulder, Colorado

Publications

Erik Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland, and John M. Martinis. Computing prime factors with a Josephson phase qubit quantum processor. *Submitted to Nature Physics*, (2012).

A. Megrant, C. Neill, R. Barends, B. Chiaro, Yu Chen, L. Feigl, J. Kelly, Erik Lucero, Matteo Mariantoni, P. J. J. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. C. White, Y. Yin, J. Zhao, C. J. Palmstrm, John M. Martinis, and A. N. Cleland. Planar Superconducting Resonators with Internal Quality Factors above One Million. *Applied Physics Letters* **100**, 113510 (2012).

Daniel Sank, R. Barends, Radoslaw C. Bialczak, Yu Chen, J. Kelly, M. Lenander, Erik Lucero, Matteo Mariantoni, M. Neeley, P. J. J. O'Malley, A. Vaisencher, H. Wang, J. Wenner, T.C. White, T. Yamamoto, Yi Yin, A. N. Cleland, and John M. Martinis. Surface spin fluctuations probed with flux noise and coherence in Josephson phase qubits. *Submitted to Physical Review Letters* (2012).

Yi Yin, H. Wang, M. Mariantoni, Radoslaw C. Bialczak, R. Barends, Y. Chen, M. Lenander, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, M. Weides, J. Wenner, T. Yamamoto, J. Zhao, A. N. Cleland, and John M. Martinis. Dynamic quantum Kerr effect in circuit quantum electrodynamics. *Physical Review A* **85**, 023826 (2012).

Matteo Mariantoni, H. Wang, T. Yamamoto, M. Neeley, Radoslaw C. Bialczak, Y. Chen, M. Lenander, Erik Lucero, A. D. OConnell, D. Sank, M. Weides, J. Wenner, Y. Yin, J. Zhao, A. N. Korotkov, A. N. Cleland, and John M. Martinis. Implementing the Quantum von Neumann Architecture with Superconducting Circuits. *Science* **334**, 61 (2011).

J. Wenner, R. Barends, Radoslaw C. Bialczak, Y. Chen, J. Kelly, Erik Lucero, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, H. Wang, T. C. White, Y. Yin, J. Zhao, A. N. Cleland, John M. Martinis. Surface loss simulations of superconducting coplanar waveguide resonators. *Applied Physics Letters* **99**, 113513 (2011).

R. Barends, J. Wenner, M. Lenander, Y. Chen, Radoslaw C. Bialczak, J. Kelly, Erik Lucero, P. O'Malley, M. Mariantoni, D. Sank, H. Wang, T. C. White, Y. Yin, J. Zhao, A. N. Cleland, John M. Martinis J. .J A. Baselmans. Minimizing quasiparticle generation from stray infrared light in superconducting quantum circuits. *Applied Physics Letters* **99**, 113507 (2011).

M. Lenander, H. Wang, Radoslaw C. Bialczak, Erik Lucero, Matteo Mariantoni, M. Neeley, A. D. O'Connell, D. Sank, M. Weides, J. Wenner, T. Yamamoto, Y. Yin, J. Zhao, A. N. Cleland, John M. Martinis. Measurement of energy decay in superconducting qubits from nonequilibrium quasiparticles. *Physical Review B* **84**, 024501 (2011).

J. Wenner, M. Neeley, Radoslaw C. Bialczak, M. Lenander, Erik Lucero, A. D. OConnell, D. Sank, H. Wang, M. Weides, A. N. Cleland, John M. Martinis. Wirebond crosstalk and cavity modes in large chip mounts for superconducting qubits. *Superconducting Science and Technology* **24**, 065001 (2011).

Matteo Mariantoni, H. Wang, Radoslaw C. Bialczak, M. Lenander, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, M. Weides, J. Wenner, T. Yamamoto, Y. Yin, J. Zhao, John M. Martinis, A. N. Cleland. Photon shell game in three-resonator circuit quantum electrodynamics *Nature Physics* **7**, 287–293 (2011).

M. Weides, R. C. Bialczak, M. Lenander, Erik Lucero, Matteo Mariantoni, M. Neeley, A. D. OConnell, D. Sank, H. Wang, J. Wenner, T. Yamamoto, Y. Yin, A. N. Cleland, J. M. Martinis. Phase qubits fabricated with trilayer junctions. *Superconducting Science and Technology* **24**, 055005 (2011).

R. C. Bialczak, M. Ansmann, M. Hofheinz, M. Lenander, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, H. Wang, M. Weides, J. Wenner, T. Yamamoto, A. N. Cleland, J. M. Martinis. Fast tunable coupler for superconducting qubits. *Physical Review Letters* **106**, 060501 (2011).

H. Wang, Matteo Mariantoni, Radoslaw C. Bialczak, M. Lenander, Erik Lucero, M. Neeley, A. O'Connell, D. Sank, M. Weides, J. Wenner, T. Yamamoto, Y. Yin, J. Zhao, John M. Martinis, A. N. Cleland. Deterministic entanglement of photons in two superconducting microwave resonators. *Physical Review Letters* **106**, 060401 (2011).

T. Yamamoto, M. Neeley, Erik Lucero, R. C. Bialczak, J. Kelly, M. Lenander, Matteo Mariantoni, A. D. O'Connell, D. Sank, H. Wang, M. Weides, J. Wenner,

Y. Yin, A. N. Cleland, John M. Martinis. Quantum process tomography of two-qubit controlled-Z and controlled-NOT gates using superconducting phase qubits. *Physical Review B* **82**, 184515 (2010).

Erik Lucero, Julian Kelly, Radoslaw C. Bilaczak, Mike Lenander, Matteo Mariantoni, Matthew Neeley, A. D. O'Connell, Daniel Sank, H. Wang, Martin Weides, James Wenner, Tsuyoshi Yamamoto, A. N. Cleland, John M. Martinis. Reduced phase error through optimized control of a superconducting qubit. *Physical Review A* **82**, 042339 (2010).

M. Neeley, R. C. Bialczak, M. Lenander, Erik Lucero, M. Mariantoni, A. D. O'Connell, D. Sank, H. Wang, M. Weides, J. Wenner, Y. Yin, T. Yamamoto, A. N. Cleland, J. M. Martinis. Generation of Three-Qubit Entangled States using Superconducting Phase Qubits. *Nature* **467**, 570–573 (2010).

B. A. Mazin, D. Sank, S. McHugh, Erik Lucero, A. Merrill, J. Gao, D. Pappas, D. Moore, J. Zmuidzinas. Thin lm dielectric microstrip kinetic inductance detector. *Applied Physics Letters* **96**, 102504 (2010).

R. C. Bialczak, M. Ansmann, M. Hofheinz, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, H. Wang, J. Wenner, M. Steffen, A. N. Cleland, J. M. Martinis. Quantum Process Tomography of a Universal Entangling Gate Implemented with Josephson Phase Qubits. *Nature Physics* **6**, 409–413 (2010).

A. D. O'Connell, M. Hofheinz, M. Ansmann, R. C. Bialczak, M. Lenander, Erik Lucero, M. Neeley, D. Sank, H. Wang, M. Weides, J. Wenner, J. M. Martinis, A. N. Cleland. Quantum ground state and single-phonon control of a mechanical resonator. *Nature* **464**, 697–703 (2010).

H. Wang, M. Hofheinz, M. Ansmann, R. C. Bialczak, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, M. Weides, J. Wenner, A. N. Cleland, J. M. Martinis Decoherence Dynamics of Complex Photon States in a Superconducting Circuit. *Physical Review Letters* **103**, 3200404 (2009).

H. Wang, M. Hofheinz, J. Wenner, M. Ansmann, R. C. Bialczak, M. Lenander, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, M. Weides, A. N. Cleland, J. M. Martinis. Improving the Coherence Time of Superconducting Coplanar Resonators. *Applied Physics Letters* **95**, 233508 (2009).

M. Ansmann, H. Wang, R. C. Bialczak, M. Hofheinz, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, M. Weides, J. Wenner, A. N. Cleland, J. M. Martinis.

Violation of Bell's inequality in Josephson phase qubits. *Nature* **461**, 504–506 (2009).

M. Neeley, M. Ansmann, R. C. Bialczak, M. Hofheinz, Erik Lucero, A. D. O'Connell, D. Sank, H. Wang, J. Wenner, A. N. Cleland, M. R. Geller, J. M. Martinis. Emulation of a Quantum Spin with a Superconducting Phase Qudit. *Science* **325**, 722 (2009).

M. Hofheinz, H. Wang, M. Ansmann, R. C. Bialczak, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, J. Wenner, J. M. Martinis, A. N. Cleland. Synthesizing arbitrary quantum states in a superconducting resonator. *Nature* **459**, 546–549 (2009).

H. Wang, M. Hofheinz, M. Ansmann, R. C. Bialczak, Erik Lucero, M. Neeley, A. D. O'Connell, D. Sank, J. Wenner, A. N. Cleland, J. M. Martinis. Measurement of the decay of Fock states in a superconducting quantum circuit. *Physical Review Letters* **101**, 240401 (2008).

M. Hofheinz, E. M. Weig, M. Ansmann, R. C. Bialczak, Erik Lucero, M. Neeley, A. D. O'Connell, H. Wang, J. M. Martinis, A. N. Cleland. Generation of Fock states in a superconducting quantum circuit. *Nature* **454**, 310–314 (2008).

Martin E. Huber, Nicholas C. Koshnick, Hendrik Bluhm, Leonard J. Archuleta, Tommy Azua, Per G. Bjrnsson, Brian W. Gardner, Sean T. Halloran, Erik Lucero, and Kathryn A. Moler. Gradiometric micro-SQUID susceptometer for scanning measurements of mesoscopic sample. *Review of Scientific Instruments* **79**, 053704 (2008).

N. Katz, M. Neeley, M. Ansmann, R. C. Bialczak, M. Hofheinz, Erik Lucero, A. D. O'Connell, H. Wang, A. N. Cleland, J. M. Martinis, A. N. Korotkov. Reversal of the Weak Measurement of a Quantum State in a Superconducting Phase Qubit. *Physical Review Letters* **101**, 200401 (2008).

M. Neeley, M. Ansmann, R. C. Bialczak, M. Hofheinz, N. Katz, Erik Lucero, A. D. O'Connell, H. Wang, A. N. Cleland, J. M. Martinis. Process tomography of quantum memory in a Josephson-phase qubit coupled to a two-level state. *Nature Physics* **4**, 523–526 (2008).

A. D. O'Connell, M. Ansmann, R. C. Bialczak, M. Hofheinz, N. Katz, Erik Lucero, C. McKenney, M. Neeley, H. Wang, E. M. Weig, A. N. Cleland, J. M. Martinis.

Microwave Dielectric Loss at Single Photon Energies and milliKelvin Temperatures. *Applied Physics Letters* **92**, 112903 (2008).

Erik Lucero, M. Hofheinz, M. Ansmann, R. C. Bialczak, N. Katz, M. Neeley, A. D. O'Connell, H. Wang, A. N. Cleland, J. M. Martinis. High-fidelity gates in a Josephson qubit. *Physical Review Letters* **100**, 247001 (2008).

M. Neeley, M. Ansmann, R. C. Bialczak, M. Hofheinz, N. Katz, Erik Lucero, A. D. O'Connell, H. Wang, A. N. Cleland, J. M. Martinis. Transformed Dissipation in Superconducting Quantum Circuits. *Physical Review B* **77**, 180508(R) (2008).

R. C. Bialczak, R. McDermott, M. Ansmann, M. Hofheinz, N. Katz, Erik Lucero, M. Neeley, A. D. O'Connell, H. Wang, A. N. Cleland, J. M. Martinis. 1/f Flux Noise in Josephson Phase Qubits. *Physical Review Letters* **99**, 187006 (2007).

M. Steffen, M. Ansmann, R. C. Bialczak, N. Katz, Erik Lucero, R. McDermott, M. Neeley, E. M. Weig, A. N. Cleland, J. M. Martinis. Measurement of the Entanglement of Two Superconducting Qubits via State Tomography. *Science* **313**, 1423–1425 (2006).

N. Katz, M. Ansmann, R. C. Bialczak, Erik Lucero, R. McDermott, M. Neeley, M. Steffen, E. M. Weig, A. N. Cleland, J. M. Martinis, A. N. Korotkov. Coherent state evolution in a superconducting qubit from partial-collapse measurement. *Science* **312**, 1498–1500 (2006).

M. Steffen, M. Ansmann, R. McDermott, N. Katz, R. C. Bialczak, Erik Lucero, M. Neeley, E. M. Weig, A. N. Cleland, J. M. Martinis. State tomography of capacitively shunted phase qubits with high fidelity. *Physical Review Letters* **97**, 050502 (2006).

Erik Lucero. Linear Current-to-Voltage Converter Utilizing Superconducting Quantum Interference Device (SQUID) Operational Amplier. *University of Colorado* Senior Thesis (2005).

Photographic Publications

Erik Lucero. Mechanical Resonator coupled to a Qubit. Quantum Ground state and single phonon control sample. *APS 2012 Calendar* (2012).

Erik Lucero. Mechanical Resonator coupled to a Qubit. Quantum Ground state

and single phonon control sample. *APS Home Page* May (2011).

Erik Lucero, Dario Mariantoni, Matteo Mariantoni. Two-Qubit Three-Resonator sample for photon shell game. Cover. *Nature Physics* April (2011).

Erik Lucero. ReZQu Architecture Four-Qubit Five-Resonator sample. *BBC* March 22 (2011).

Erik Lucero. ReZQu Architecture Four-Qubit Five-Resonator sample. *APS March Meeting* March (2011).

Erik Lucero. Mechanical Resonator coupled to a Qubit. Quantum Ground state and single phonon control sample. *Fysikaktuellt* March (2011).

Erik Lucero. Four-Qubit sample for three-qubit GHZ and W state. *New York Times* November 8 (2010).

Erik Lucero. Qubit coupled to Resonator, arbitrary state generation and Wigner tomography sample. Cover. *Quantum Measurement and Control* Cambridge University Press (2010).

Erik Lucero. Qubit coupled to Resonator, arbitrary state generation and Wigner tomography sample. *New Scientist* August 15 (2009).

Erik Lucero and Max Hofheinz. Qubit coupled to Resonator, arbitrary state generation and Wigner tomography sample. Cover *Physics Today* July (2009).

Abstract

Computing prime factors using a Josephson phase-qubit architecture:

$15 = 3 \times 5$

by

Erik Anthony Lucero

Josephson phase-quantum-bits, ("qubits"), together with superconducting resonators, comprise the essential quantum elements in a state-of-the-art quantum processor (QuP). A QuP can be used to exploit quantum mechanics to find the prime factors of composite numbers by running Shor's algorithm[57].

In this thesis, I describe the first solid-state demonstration of a compiled version of Shor's algorithm. To meet this challenge, I designed a QuP so that I could map the problem of factoring the number $N = 15$ onto a quantum circuit that is compatible with our technological capabilities. The QuP is composed of nine quantum elements: four phase qubits and five superconducting coplanar waveguide (CPW) resonators. Using this device, I ran a three-qubit complied version of Shor's algorithm and successfully found the prime factors 48 % of the time (compared to the ideal success rate of 50 %). In addition, the QuP produced coherent interactions between five quantum elements, and bi- and tripartite entanglement,

which was verified via quantum state tomography (QST).

Scaling up to nine quantum elements and performing these experiments represent key milestones to realizing a quantum computer. Continued improvements in the superconducting qubit coherence times and more complex circuits should provide the resources necessary to factor larger composite numbers and run more intricate quantum algorithms in the near future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We are at a very exciting point on the experimental road to a quantum computer. Much like the days of the first transistor, which physically occupied an entire tabletop, these coarse, or some might say crude, initial offerings of engineered quantum systems require a lot of physical space and infrastructure (e.g. Figure 1.1). It is not clear, nor will I speculate here on which architecture will ultimately persevere (it is really too early to tell). Instead, in this thesis we will delve into the current experimental challenges of building, operating, and programming a quantum processor (QuP) -a precursor to realizing a quantum computer[49, 30]- to run a compiled version of Shor's Algorithm[57]. The solid-state quantum processor described here is made up of specially engineered quantum bits herein referred to as "qubits". These qubits are of the superconducting phase qubit variety, along

with superconducting quantum memory, arranged in a quantum von Neumann architecture[39]. This *quantum* architecture is analogous to the, perhaps more familiar, *classical* von Neumann architecture, which comprises a central processing unit "CPU" (or quantum-CPU, "qu-CPU" for short) and random access memory "RAM" (or "qu-RAM"). And just as classical processors are built up from a number of simpler components (transistors or "bits"), so too is this quantum processor, which is built up from many "qubits".

Superconducting qubits belong to a family of quantum circuits that could be used as components in a quantum computer. These electrical circuits, like their classical computer counterparts of bits in silicon, share the advantage of conventional microfabrication techniques, which allows for straight-forward scaling by simply arranging more components on a chip. Achieving this scaling with quantum bits of any variety is a prodigious challenge, but the initial steps are now underway with superconducting qubits. The state of the art quantum processor described in this thesis is physically comprised of nine quantum elements: four superconducting phase qubits and five superconducting resonators all designed to be manipulated with microwave radiation.

The solid state quantum processor (QuP) that I will describe here builds on the work of many successful experiments and infrastructure. For more details on the UCSB superconducting quantum computing group's, (herein referred to

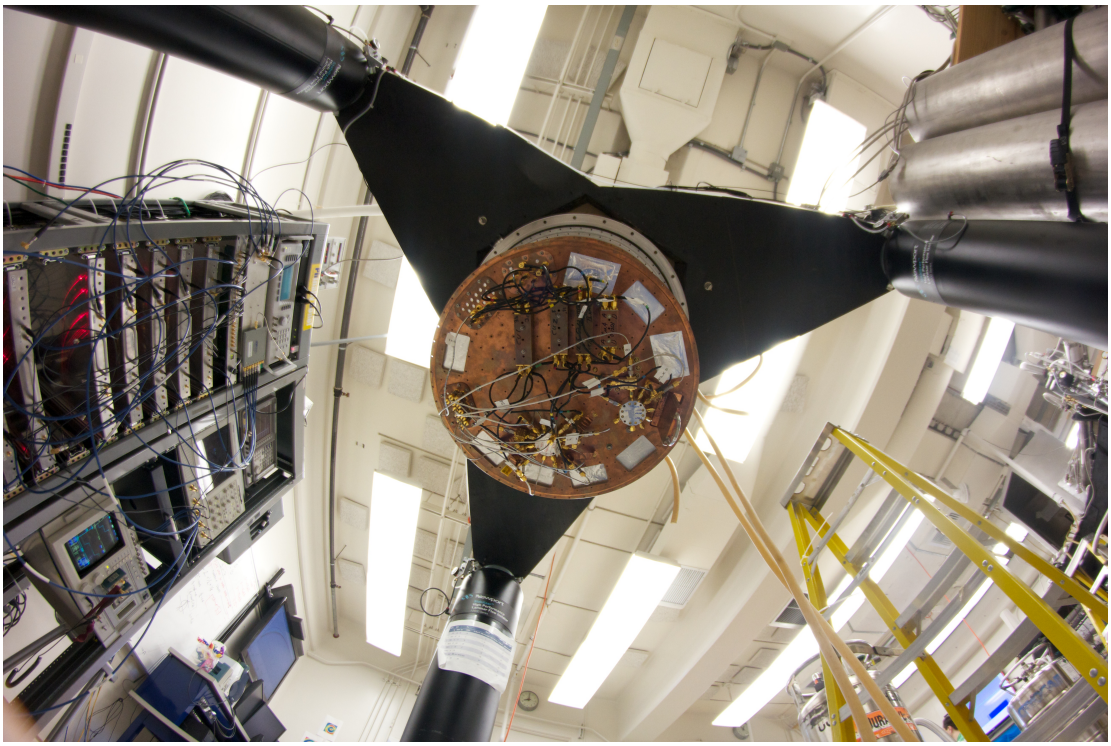

Figure 1.1: UCSB QC-Group Superconducting qubit infrastructure. Picture taken from underneath refrigerator with a wide angle lens. (Center) He^3-He^4 dilution refrigerator (DR) open for sample mounting. (Center) Cu-plate qubit sample stage with qubit devices. DR is suspended on vibration isolation pylons (edges of the black star structure). (Left) Rack of custom microwave electronics used to control qubits. (Right near yellow ladder) Dewars of cryogens: liquid Helium and liquid Nitrogen to operate refrigerator. (Bottom Left) Experimental control station.

as the "UCSB QC-group"), phase qubit fabrication, operation, and software infrastructure, I refer the interested reader to Markus Ansmann's pioneering PhD thesis[3]. Of course the complexity of the experiments have continued to evolve and I will be providing the details on how things have changed. The backbone to our control-software infrastructure relies on the LabRAD software[1] originally conceived and developed by Markus Ansmann and Matthew Neeley. Our custom qubit control-electronics were developed in-house by Professor John Martinis and myself[2]. Together these efforts have enabled a number of crucial advancements and experiments in this field, including the work I describe in this thesis.

The field of quantum computing has sparked the interest of the scientific community[3] and the popular press[4] as a number of exciting technological breakthroughs have been achieved. In an effort to stir the reader to investigate beyond this thesis, Figure 1.2 highlights a handful of key devices (by no means exhaustive) from the UCSB QC-group. These devices were chosen for their scientific relevance and because they clearly show how the hardware has evolved over the years. I make note of this evolution in hardware because each new device and experiment builds upon the success of the previous devices. I will also show in Appendix A

[1] http://sourceforge.net/projects/labrad/

[2] https://commando.physics.ucsb.edu/tw/view/Electronics/PubDocs

[3] http://www.aaas.org/news/releases/2010/1216sp_boy.shtml?sa_campaign=Internal_Ads/AAAS/AAAS_News/2010-12-16/jump_page

[4] http://www.nytimes.com/2010/11/09/science/09compute.html,http://www.bbc.co.uk/news/science-environment-12811199,http://www.nytimes.com/2012/02/28/technology/ibm-inch-closer-on-quantum-computer.html

that the software has evolved in a similar fashion. When what previously would take us months or years to calibrate and figure out, in all of the nuance and detail important to an experiment, we can now (and need to continue to be able to!) repeat in a few minutes on the freshest fabricated device.

Looking at the devices shown in Figure 1.2 one can see the increase in device complexity starting at top left and moving across the row and then down the page. The device pictured in Figure 1.2a was used to create both Fock states[24] and arbitrary quantum states[23] using a single phase qubit (white rectangle near the center) capacitively coupled to a superconducting resonator (long black line with sequential gold highlights) forming a device with a total of two quantum elements.

The next device in Figure 1.2b was used to perform the first solid-state violation of Bell's inequality[4]. It is comprised of two phase qubits (white rectangles near the green squares at either side, left and right, of the device) coupled via a resonator (serpent pattern in the center) forming a total of three quantum elements.

The purple colored device in Figure 1.2c was designed to show the first demonstration of a mechanical harmonic oscillator in its quantum ground state and in a superposition of quantum states[50]. The device paired a phase qubit with a mechanical resonator, analogous to the electromagnetic resonator in the previous two devices.

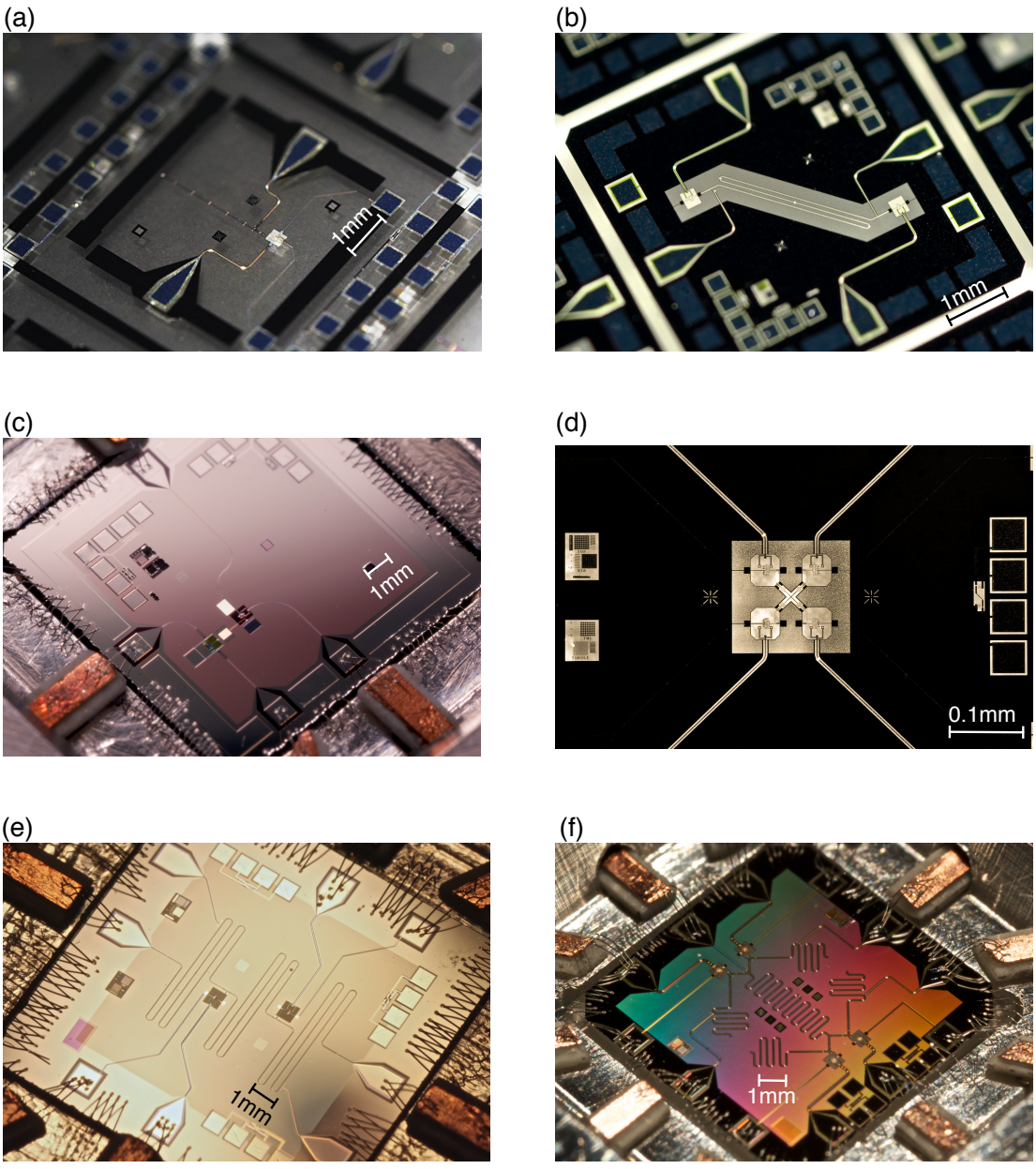

Figure 1.2: Evolution of UCSB Superconducting qubit devices.

The device pictured in the black and white photomicrograph in Figure 1.2d was used to show both classes of three qubit entanglement[47]. This device is comprised of four phase qubits (four rectangles with the white lines extending out at $\sim 45°$) coupled to one another via a capacitive island (white 'X' in the center), for a total of four quantum elements.

The device in Figure 1.2e was used to demonstrate the first prototype of the quantum von Neumann architecture[39] and it was used to create NOON states[66]. The device has two phase qubits and three superconducting resonators, where one of the resonators is shared between both qubits, for a total of five engineered quantum elements.

The final device in the photomicrograph matrix in Figure 1.2f (rainbow colored due to diffraction) was used to run the first solid sate demonstration of Shor's algorithm[34]. It is comprised of four phase qubits (rectangles positioned on the edge about the center of the chip) and five superconducting resonators (serpent patterns in the middle and splitting off from the qubit at $\sim 45°$) for a total of nine engineered quantum elements. This device is the basis for this thesis.

In this thesis we demonstrate experimental control over the nine superconducting quantum elements that form the QuP and run a quantum algorithm, namely Peter Shor's prime factorization algorithm[57] appropriately compiled for our number of qubits to factor the number $N = 15$ into its two prime factors

(wait for it). In addition, we show two-qubit entanglement and both classes of three-qubit entanglement both via fast entangling operations, and by combining single and coupled qubit gates. This state of the art solid-state QuP helps to underline the promise of superconducting qubit architectures for scaling up to a full fledged quantum computer.

1.1 Shor's Algorithm

All this talk of quantum bits, quantum processors, and quantum computers, but what are they good for? Many things[49] but let's be clear, the end-game for quantum computers is not one that merely replaces your desktop (or tablet) personal computer[5]. Instead, the vision is more revolutionary in that a full-scale quantum computer would use a completely new and different form of computation, one that relies on the physics of quantum mechanics to solve problems that would otherwise be intractable on a computer that relies merely on classical physics.

[5]Imagine a dilution refrigerator like the one pictured in Figure 1.1 and a group of graduate students to run it in every home!

1.1.1 A Practical Use of a Quantum Computer: Finding Prime Factors

The problem of finding the prime factors of some composite number[6] is considered a "hard" problem from a computer science and mathematical perspective and is why contemporary encryption schemes like RSA (named after the inventors, Ron Rivest, Adi Shamir, and Leonard Adleman) make use of this fact for secure data transmissions (e.g. sending someone your credit card information).

The problem is as follows: we are given some composite number N, like the largest published RSA number[7] shown in Table 1.1, and we seek the two constituent prime numbers p and q that were multiplied together to form N. Armed with a classical computer and the best known classical algorithm, the general number field sieve[32], it will take sub-exponential time $O(\exp[(\log(\mathrm{N}))^{1/3}(\log(\log(\mathrm{N})))^{2/3}])$ to find a solution. In other words, we can expect to wait on the order of the age of the universe, currently estimated to be 13 billion years (4×10^{17} seconds), before we obtain p and q.

However, given a quantum computer capable of running the best known quantum algorithm, Shor's algorithm[57], which exploits quantum mechanics, we would only have to wait polynomial time $O((\log(\mathrm{N}))^3)$[57]. In other words, we would

[6] A composite number is a number that is formed by the product of multiplying prime numbers together.

[7] RSA-2048 is a 2048 bit (617 decimal digit) composite number.

RSA-Number	Value
RSA-2048 =	25195908475657893494027183240048398571429282126204032027771378360436620207075955562640185258807844069182906412495150821892985591491761845028084891200728449926873928072877767359714183472702618963750149718246911650776133798590957000973304597488084284017974291006424586918171951187461215151726546322822168699875491824224336372590851418654620435767984233871847744479207399342365848238242811981638150106748104516603773060562016196762561338441436038339044149526344321901146575444541784240209246165157233507787077498171257724679629263863563732899121548314381678998850404453640235273819513786365643912120103971228221207203 57

Table 1.1: The largest published RSA number, RSA-2048, 2048 bit (617 decimal digit) composite number.

only have to wait on the order of 10's of seconds or about the time to brew a cup of coffee -a very practical amount of time, to factor even the largest RSA number. Therefore, a quantum computer can solve this problem faster than a classical computer by many orders of magnitude $\sim 10^{15}$.

This thesis was inspired by this practical application. To meet this challenge, I designed a QuP to map the problem of factoring $N = 15$ onto a quantum circuit that is compatible with our technological capabilities. $N = 15$ was chosen because it is the smallest composite number that satisfies the conditions appropriate to test Shor's algorithm[8], namely 15 is a composite number, it is not prime, and it is not even. If any of these conditions are not satisfied Shor's algorithm fails. The

[8]The next composite number being N=21, which is a significantly more complex problem.

Integer	Composite	Prime	Even	Notes
2	No	Yes	Yes	Shor Algo. Fails: prime, even
3	No	Yes	No	Shor Algo. Fails: prime
4	No	No	Yes	Shor Algo. Fails: not composite $p = q$, even
5	No	Yes	No	Shor Algo. Fails: prime
6	Yes	No	Yes	Shor Algo. Fails: even
7	No	Yes	No	Shor Algo. Fails: prime
8	No	No	Yes	Shor Algo. Fails: even
9	No	No	No	Shor Algo. Fails: not composite $p = q$
10	Yes	No	Yes	Shor Algo. Fails: even
11	No	Yes	No	Shor Algo. Fails: prime
12	Yes	No	Yes	Shor Algo. Fails: even
13	No	Yes	No	Shor Algo. Fails: prime
14	Yes	No	Yes	Shor Algo. Fails: even
15	Yes	No	No	Shor Algo. Succeeds
16	No	No	Yes	Shor Algo. Fails: not composite $p = q$, even
17	No	Yes	No	Shor Algo. Fails: prime
18	Yes	No	Yes	Shor Algo. Fails: even
19	No	Yes	No	Shor Algo. Fails: prime
20	Yes	No	Yes	Shor Algo. Fails: even
21	Yes	No	No	Shor Algo. Succeeds.

Table 1.2: Table of potential composite numbers to test Shor's algorithm.

table of integers from $N = 2$ to $N = 21$ are shown in Table 1.2 along with notes of whether or not Shor's algorithm would succeed. Note that the integer $N = 21$ also meets the conditions to test Shor's algorithm, but I leave that for a future demonstration.

To find the prime factors of (odd) composite numbers, like $N = 15$ or the RSA-2048 number, Shor's algorithm combines the power of both classical and quantum

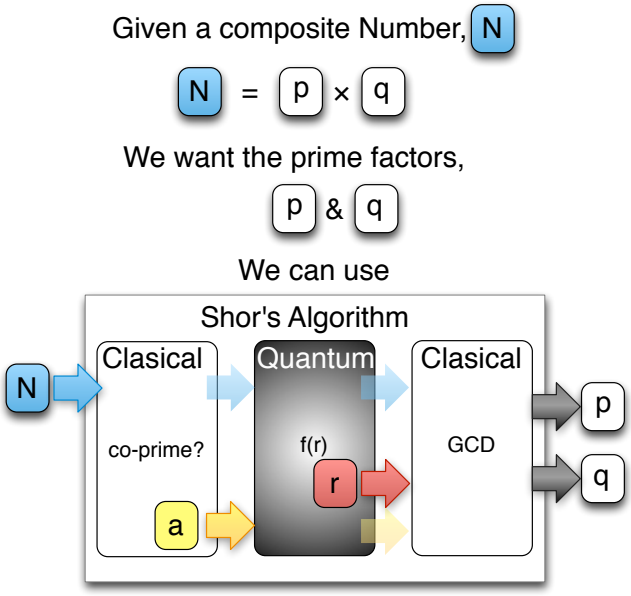

Figure 1.3: Finding prime factors using Shor's algorithm illustrated in a top-level diagram.

computation in a three subroutines, which are sketched out in Figure 1.3.

1.1.2 Classical Subroutines

The first and last step in Shor's algorithm are classical subroutines, which rely on an efficient variant of Euclid's greatest common divisor (GCD) algorithm. The first step is to select the number a. This can be done by randomly selecting a number between 1 and N and then checking that it is co-prime[9] with N via Euclid's GCD algorithm. If it is, then the next step is to continue on to the quantum routine. If it is not co-prime with N, another number is chosen and

[9] a and N being co-prime means that for $1 < a < N$ the greatest common divisor between a and N is 1.

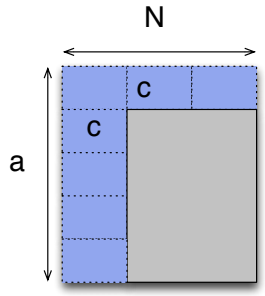

Figure 1.4: Simple illustration of Greatest Common Divisor.

checked until this condition is satisfied. A simple illustration of finding the GCD is shown in Figure 1.4, where we consider a rectangle with area $A = a \times N$ and we seek the largest value of c that divides a and N exactly, $\mathrm{GCD}(a, N) = c$. For the co-prime case between a and N, $c = 1$.

1.1.3 Quantum Subroutine

With the two inputs into the algorithm namely, the number N that we want to factor and a co-prime a, we can continue on to the quantum computation. Shor's algorithm requires a quantum computer to find the period r of the function $f(r) = a^r \mathrm{mod}(N)$. This subroutine utilizes the quantum computer's power to evaluate $f(r)$ for many values of r simultaneously. Fortunately, for the $N = 15$ case, we can check the the quantum computer's solutions by evaluating $f(r) = a^r \mathrm{mod}(N)$ for $N = 15$ and its co-primes.

r	$\mathbf{a^r}\text{mod}(N)$	**Result**
1	$7^1\text{mod}(15) = 7\text{mod}(15)$	7
2	$7^2\text{mod}(15) = 49\text{mod}(15)$	4
3	$7^3\text{mod}(15) = 343\text{mod}(15)$	13
4	$7^4\text{mod}(15) = 2401\text{mod}(15)$	1
5	$7^5\text{mod}(15) = 16{,}807\text{mod}(15)$	7
6	$7^6\text{mod}(15) = 117{,}649\text{mod}(15)$	4
7	$7^7\text{mod}(15) = 823{,}43\text{mod}(15)$	13
8	$7^8\text{mod}(15) = 5{,}764{,}801\text{mod}(15)$	1
9	$7^9\text{mod}(15) = 40{,}353{,}607\text{mod}(15)$	7
⋮	⋮	⋮
$r = 4n$ for integer n		

Table 1.3: Table of values for evaluating $a^r\text{mod}(N)$ for $a = 7$ and $N = 15$.

Consider the Example N=15

The first step is to find a co-prime a. For $N = 15$ it is straightforward to check the possible co-primes for $1 < a < N$, by simple division. Here is the set of co-primes $a \in \{2, 4, 7, 8, 11, 13, 14\}$. For pedagogical reasons I select $a = 7$ and $a = 4$ to illustrate two different periods.

The next step is to find the smallest power for $a = 7$ that satisfies the equation $a^r\text{mod}(N) = 1$. Table 1.3 tabulates the values of $a^r\text{mod}(N)$ for $a = 7$, $N = 15$, and increasing r. From the results in Table 1.3 it is clear that for co-prime $a = 7$ and $N = 15$, the function $f(r) = a^r\text{mod}(N)$ has a period, $r = 4$. In other words, the function repeats every fourth integer.

As another example, let $a = 4$. From the results in Table 1.4 $f(r)$ has a period of $r = 2$. Repeating these calculations for the remaining co-primes a, it can be

r	$\mathbf{a^r}\text{mod}(N)$	**Result**
1	$4^1\text{mod}(15) = 4\text{mod}(15)$	4
2	$4^2\text{mod}(15) = 16\text{mod}(15)$	1
3	$4^3\text{mod}(15) = 64\text{mod}(15)$	4
4	$4^4\text{mod}(15) = 256\text{mod}(15)$	1
$\vdots$	$\vdots$	$\vdots$
$r = 2n$ for integer n		

Table 1.4: Table of values for evaluating $a^r\text{mod}(N)$ for $a = 4$ and $N = 15$.

shown that for $a \in \{2, 7, 8, 13\}$ these co-primes all have a period $r = 4$ and for $a \in \{4, 11, 14\}$, $r = 2$.

Armed with period r of the function $f(r) = a^r\text{mod(N)}$ we finish the computation by employing Euclid's GCD algorithm to find the GCD between, $\text{GCD}(a^{r/2} \pm 1, N) = \{p, q\}$. For $p = \text{GCD}(\text{a}^{\text{r}/2} + 1, \text{N})$ (and $q = \text{GCD}(\text{a}^{\text{r}/2} - 1, \text{N})$), the prescription looks likes the following: $p = r_{k-1}$ for k steps, until $r_k = 0$.

$$\begin{aligned} k = 0 : \\ a^{r/2} + 1 = q_0 \times N + r_0 \\ k = 1 : \\ N = q_1 \times r_0 + r_1 \\ k = 2 : \\ r_0 = q_2 \times r_1 + r_2 \\ \vdots \\ r_{k-2} = q_k \times r_{k-1} + r_k. \end{aligned} \tag{1.1}$$

Where q_k (r_k) is the quotient (remainder) of the k^{th} step.

For $a = 7$ and $r = 4$ the calculations proceed as following:

$$
\begin{aligned}
p &= \mathrm{GCD}(50, 15) \\
k &= 0 : \\
50 &= 3 \times 15 + 5 \\
k &= 1 : \\
15 &= 3 \times 5 + 0 \\
\therefore p = r_0 &= 5, \because r_1 = 0.
\end{aligned} \tag{1.2}
$$

The algorithm finishes because $r_1 = 0$, and therefore $p = r_0$. Similarly, for q:

$$
\begin{aligned}
q &= \mathrm{GCD}(48, 15) \\
k &= 0 : \\
48 &= 3 \times 15 + 3 \\
k &= 1 : \\
15 &= 5 \times 3 + 0 \\
\therefore q = r_0 &= 3, \because r_1 = 0.
\end{aligned} \tag{1.3}
$$

Thereby completing the algorithm and recovering the prime factors, p and q that were multiplied together to give N.

1.2 A Qubit and The Bloch Sphere

In order to run a quantum algorithm, we first need quantum bits, or "qubits". Qubits are the simplest quantum system with just two states. We will call theses states $|g\rangle$ and $|e\rangle$, analogous to the classical binary digit or "bit" which can take on the values 0 or 1. To completely describe the state of the qubit $|\psi\rangle$, we can

write $|\psi\rangle = \alpha\,|g\rangle + \beta\,|e\rangle$ where the coefficients α and β are complex numbers, referred to as "amplitudes" for the system to be in these two eigenstates. In other words, the qubit state can be in some amount of $|g\rangle$ *and* $|e\rangle$ at the same time dictated by the values of α and β. These coefficients satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$ and can be interpreted as probabilities: $P_0 = |\alpha|^2$ is the probability that upon measurement the system will be found to be in state $|g\rangle$, while $P_1 = |\beta|^2$ is the probability that the system will be found in state $|e\rangle$.

Because of the normalization constraint $|\alpha|^2 + |\beta|^2 = 1$, we can write an arbitrary qubit state $|\psi\rangle = e^{i\phi_\alpha}\cos(\frac{\theta}{2})\,|g\rangle + e^{i\phi_\beta}\sin(\frac{\theta}{2})\,|e\rangle$, where the angle θ varies between 0 and π. Since global phases are unobservable in quantum mechanics, we can restrict the amplitude, α to be real (by setting $\phi_\alpha = 0$), leaving a relative phase on β ($\phi_\beta \rightarrow \phi$) and allowing us to rewrite the qubit state as $|\psi\rangle = \cos(\frac{\theta}{2})\,|g\rangle + e^{i\phi}\sin(\frac{\theta}{2})\,|e\rangle$, where the relative phase ϕ can vary from 0 to 2π.

Writing the qubit state in this spherical form, with azimuthal angle ϕ and polar angle θ, provides us with a nice geometric representation of a unit sphere, also known as the Bloch sphere, as shown in Figure 1.5.

1.2.1 Qubit Control

In the Bloch sphere picture qubit control corresponds to rotations of the qubit state $|\psi\rangle$ about some axis. These rotations are analogous to the familiar two-state

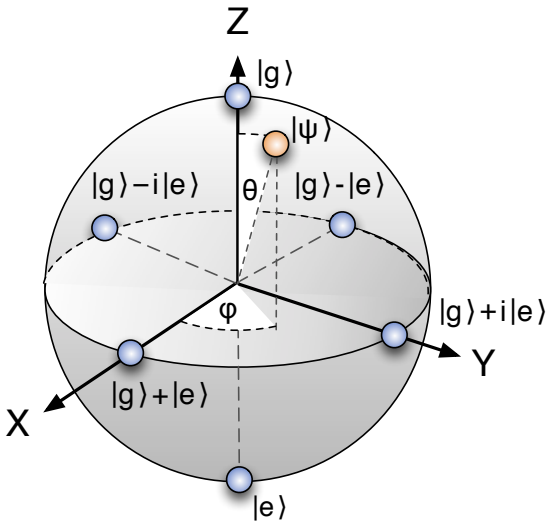

Figure 1.5: The Bloch sphere.

spin-$\frac{1}{2}$ particle subject to a magnetic field. Rotations about the x-, y- and z-axes of the Bloch sphere are generated by the Pauli matrices

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.4}$$

Where a rotation $R_z(\gamma)$ about the z-axis by some angle γ takes the initial qubit state $|\psi\rangle$ to $|\psi'\rangle$ via the unitary operation,

$$|\psi'\rangle \;=\; U\,|\psi\rangle\,, \tag{1.5}$$

where

$$\begin{aligned} U &= R_z(\gamma) \\ &= \exp(-(i\gamma/2)\sigma_z). \end{aligned} \tag{1.6}$$

Consider a uniform magnetic field applied in the z-direction to a spin-$\frac{1}{2}$ particle. The Hamiltonian that describes this interaction is given by $H = \frac{\hbar\omega}{2}\sigma_z$, where $\hbar$ is the Planck constant over 2π and ω is the Larmor frequency proportional to the applied magnetic field. The unitary time evolution operator is therefore given by

$$\begin{aligned} U(t) &= \exp(-iHt/\hbar) \\ &= \exp(-i(\omega t/2)\sigma_z). \end{aligned} \tag{1.7}$$

Comparing Equation 1.6 to Equation 1.7 we note that $U(t)$ looks just like the rotation about the z-axis $R_z(\gamma)$ with ωt replacing the angle γ. And similarly for rotations about the x-axis $R_x(\gamma) = \exp\left(-(i\gamma/2)\sigma_x\right)$ and the y-axis $R_y(\gamma) = \exp\left(-(i\gamma/2)\sigma_y\right)$.

Although three axes control is convenient, it turns out that with control over just two axes, say x and z one can build up any rotation $R_\alpha(\gamma)$, where α is an arbitrary axis. This provides complete control over the qubit state -an important requirement for quantum computation.

1.2.2 The Density Matrix Description

Pure states map to the Bloch sphere, as described above and illustrated in Figure 1.5, but how can we describe the qubit state in a real experiment, where it

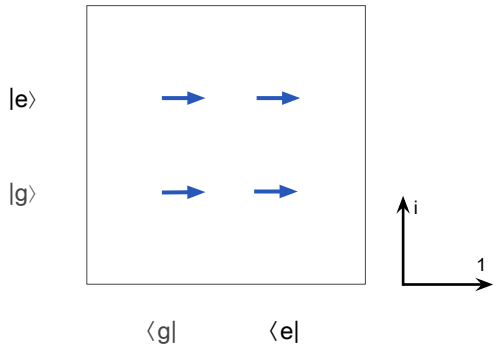

Figure 1.6: Density matrix for a single qubit. Each blue arrow represents a complex number formed by $\langle g|\rho|g\rangle$, $\langle g|\rho|e\rangle$, $\langle e|\rho|g\rangle$, and $\langle e|\rho|e\rangle$

interacts with the environment and is manipulated with imperfect controls? The qubit can no longer be described by just a pure state, but instead it can be described as a *collection* of pure states. More succinctly we can employ the density matrix formalism, which allows for a quantum system to be in a probabilistic mixture of different pure states. The density matrix ρ is defined as $\sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right|$, where p_i is the probability that the system is in the (pure) state $|\psi_i\rangle$. The density matrix formalism also allows for points inside the Bloch sphere, which represent mixed states and accounts for decoherence. Another advantage to the density matrix description is that it generalizes to more than one qubit. For the majority of this thesis I will stick with the density matrix description, but wherever possible for single qubits I will use the simpler pure state model.

To help familiarize the reader with another way the density matrix is displayed lets look at a representative density matrix for a single qubit, as shown in Fig-

ure 1.6. The axes drawn off to the right of the plot apply to each of the four blue arrows. The ordinate axis is the imaginary axis that ranges from $-i$ to i. The abscissa axis is the real axis that ranges from -1 to 1. The four blue arrows represent the complex number formed by $\langle g|\rho|g\rangle$, $\langle g|\rho|e\rangle$, $\langle e|\rho|g\rangle$, and $\langle e|\rho|e\rangle$. The data shown are for a qubit (theoretically) prepared in the equal superposition state, $|\psi\rangle = \frac{1}{\sqrt{2}}(|g\rangle + |e\rangle)$, which can be written as $\rho = |\psi\rangle\langle\psi| = \frac{1}{2}[(|g\rangle + |e\rangle)(\langle g| + \langle e|)] = \frac{1}{2}(|g\rangle\langle g| + |g\rangle\langle e| + |e\rangle\langle g| + |e\rangle\langle e|)$. So, each of the four combinations are (theoretically) represented by an arrow of length 1/2 pointing completely along the real axis.

1.3 Decoherence

Although extensive efforts are made to isolate qubits from the environment so as to protect the fragile quantum states we still have to control, measure, and connect qubits so they can interact with one another. These additional degrees of freedom (collectively referred to as "the environment") provide routes for the quantum states to leak into or "decohere". Decoherence refers to these various processes where the coupling to the environment deteriorates the quantum state.

One process is relaxation, where the excited state dissipates energy into the environment and relaxes to the ground state. Consider the Bloch sphere picture, where a qubit is prepared in the $|e\rangle$ state. As it relaxes the vector shrinks towards

the middle of the sphere and continues to relax all the way back into the ground state. Relaxation is described by an exponential decay time T_1.

Another decoherence process is dephasing. Consider a qubit state prepared in the equator of the Bloch sphere, $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\phi}|1\rangle)$ with a definite phase $\phi = 0$. Dephasing causes the qubit transition frequency to fluctuate, which is like applying randomized z-rotations, thereby randomizing ϕ and reducing the phase coherence. In the Bloch sphere picture, this process shrinks the state vector towards the center. Dephasing is commonly described by an exponential decay time T_ϕ, but because relaxation also causes loss of phase coherence the two times are often expressed as one characteristic decoherence timescale, T_2, where $\frac{1}{T_2} = \frac{1}{2T_1} + \frac{1}{T_\phi}$.

As one might guess decoherence is one of the main limiting factors for experimentalists. The trade off between protecting the qubit on the one hand (reducing decoherence) and being able to easily manipulate and encourage qubits to interact on the other (which typically increases decoherence) is a continuous engineering issue. Reducing decoherence continues to be a topic of research in all quantum information architectures, including superconducting qubits[55, 43, 8, 17].

1.4 Multiple Quantum Elements

Because our quantum processor is comprised of quantum elements that include qubits (which ideally have only two "computational states") and superconducting

resonators (which are quantum harmonic oscillators that have many levels enumerated by $0, 1, 2, 3 \ldots n$) in this thesis, I will use the notation $|g\rangle$, $|e\rangle$, to represent the "ground" and "excited" states of a qubit and I will increment the letter accordingly for the higher excited states of the qubit, i.e. $|f\rangle$, $|h\rangle$, ..., *etc.* Whereas for the resonator states, I will use the notation $|0\rangle, |1\rangle, \ldots, |n\rangle$ to represent the photon number states of the resonator.

To describe the ground state of the full nine quantum element system we simply combine the individual states for the resonators and qubits via an outer product $|00000\rangle \otimes |gggg\rangle$, where the set of five "0"s represent the ground state for the five resonators and the four "g"s represent the ground state for the respective qubits. From this notation it should be clear when describing only a subset of elements, like one resonator in the ground state and two qubits in the first excited state, e.g. $|0\rangle \otimes |ee\rangle$ (or more compactly, $|0ee\rangle$).

The current state of the art in superconducting qubits is nine quantum elements, but simply placing nine isolated quantum elements on a single chip allows for nothing more exciting than single qubit dynamics. However, with superconducting quantum elements we can make connections with superconducting wires and capacitors. The size of the capacitors set a maximum coupling strength between the elements. Although the coupling capacitors are fixed we can still turn the coupling interaction on and off by electronically adjusting the qubit $|g\rangle \rightarrow |e\rangle$

transition frequency.

In Chapter 2, I will describe the coupling scheme implemented between the phase qubits and the superconducting resonators that form the QuP. As a consequence of this scheme, the QuP is capable of various forms of *controllable* multi-quantum element interactions and the ability to build up quantum algorithms via the standard single- and coupled-qubit gate model.

1.5 Superposition and Entanglement

Superposition, as discussed above in §1.2, refers to a physical system that must be described by its amplitude to be in each of a set of possible eigenstates. Recall that these amplitudes (α and β above) are complex numbers that when squared give the probability of measuring the system in that particular state. When a system is strongly measured (as opposed to only measured weakly [29]), it will be found in one and *only one* of these eigenstates. However, all the time before and leading up to measurement the state is described by the entire superposition.

Entanglement in quantum mechanics refers to a (two or more) multi-body quantum state that can no longer be described as a product state. The system is instead in a compound state where the participating components are intertwined, or *entangled*. Entanglement plays a key role in quantum mechanics and by no surprise also in quantum computation. In fact, entanglement is the underlying

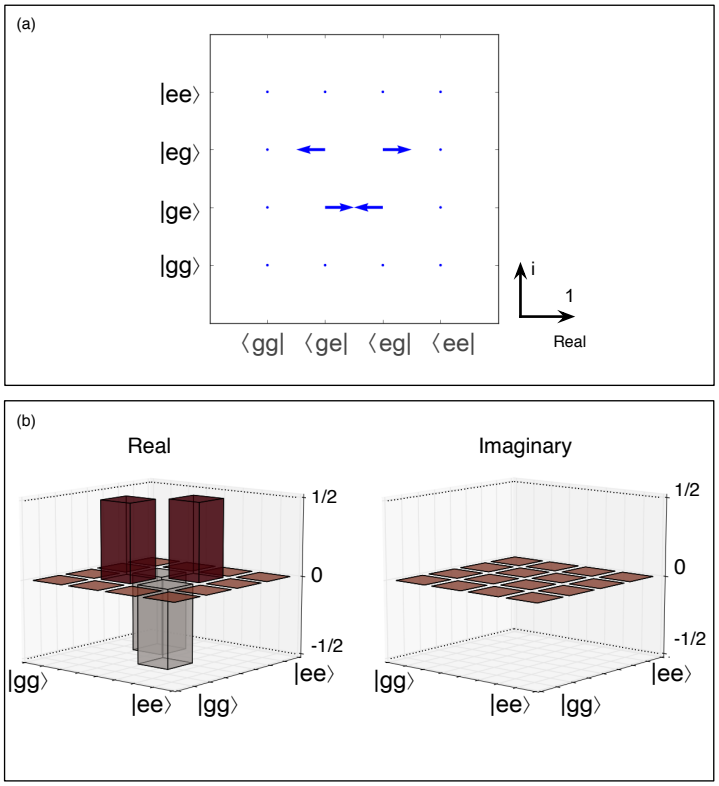

Figure 1.7: Entangled two qubit state: Bell singlet $|\psi_s\rangle = (|ge\rangle - |eg\rangle)/\sqrt{2}$. Data are described in the text and displayed in (a) arrow plot and (b) bar plot.

phenomena that gives rise to correlations between the participating qubits that violates correlations predicted by invoking classical physics. Entanglement is one of the characteristics that distinguishes quantum computers from classical computers and what gives rise to the speedup over classical computation.

A canonical example of a two qubit entangled state is a "Bell-singlet", $|\psi_s\rangle = (|ge\rangle - |eg\rangle)/\sqrt{2}$. Upon writing this state, one can glean that there is no way to extract either qubit from the entangled state to form a product state, e.g.

$|\psi_s\rangle \neq |g_1\rangle\,(|e_2\rangle - |g_2\rangle)$, where the subscripts refer to qubit 1 and 2 respectively. In an experiment, where the Bell singlet is prepared and qubit 1 is measured in the ground state $|g_1\rangle$, qubit 2 will be found in the excited state $|e_2\rangle$, and vice versa. In other words, the action of measuring qubit 1 (2) determines the outcome of qubit 2 (1). These correlations will continue to exist even when the qubits are separated in space or time or when different measurement axes are used.

The Bell singlet can be expressed as a density matrix,

$$\begin{aligned} \rho_s &= |\psi_s\rangle\langle\psi_s| \\ &= \frac{1}{2}(|ge\rangle\langle ge| - |ge\rangle\langle eg| - |eg\rangle\langle ge| + |eg\rangle\langle eg|) \end{aligned} \tag{1.8}$$

and displayed like the data shown in Figure 1.7, where I introduce another more commonly used "metropolis" or bar-graph in Figure 1.7b along with the now familiar arrow plot in Figure 1.7a. The data here are for the ideal preparation of the Bell singlet.

Starting with the arrow plot in Figure 1.7a, the axes drawn off to the right of the plot apply to all sixteen origins for arrows. The ordinate axis is the imaginary axis that ranges from $-i$ to i. The abscissa axis is the real axis that ranges from -1 to 1. The sixteen arrows represent the complex numbers formed by $\langle gg|\rho_s|gg\rangle$,$\langle gg|\rho_s|ge\rangle$,$\langle gg|\rho_s|eg\rangle$,$\langle gg|\rho_s|ee\rangle$, ..., $\langle ee|\rho_s|ee\rangle$. From Equation 1.8 there are (theoretically) four arrows of length $\frac{1}{2}$ pointing along the positive (negative), $\langle ge|\rho_s|ge\rangle$, $\langle eg|\rho_s|eg\rangle$ ($\langle ge|\rho_s|eg\rangle$, $\langle eg|\rho_s|ge\rangle$) real axis.

The metropolis plot in Figure 1.7b breaks the data into two plots (one for the real and one for the imaginary components). Although I have shown both real and imaginary components, it is common to show only the real components especially when the imaginary components are zero (or nearly vanishing experimentally). I have also left off the middle labels $|ge\rangle$, $|eg\rangle$, $\langle ge|$, and $\langle eg|$ for visual clarity. The four bars correspond to the same four numbers as above in Figure 1.7a.

Being able to prepare and measure the existence of two-qubit and three-qubit entanglement provides a benchmark that can be used across quantum architectures. Therefore, we measure both two and three qubit entanglement and in §5.7.4, we show that entanglement is necessary to successfully run a quantum algorithm.

1.6 The Road Ahead

With this introductory chapter behind us, the remainder of this thesis will proceed in the following manner. In Chapter 2, I describe our superconducting resonators and phase qubits and how they are designed, fabricated, and connected to form a quantum processor (QuP). Since we ultimately want to run a quantum algorithm on this QuP, which requires electronic-manipulation of the quantum elements, we first need to understand how to optimally control the phase qubits. Therefore, in Chapter 3, I will describe how we experimentally reduced amplitude errors, an error associated with control, down to fault tolerant levels through accurate

and precise manipulation of the qubit. Then in Chapter 4, I will discuss how we experimentally implemented a control theory to reduce phase errors, which consequently allowed us to perform even faster high fidelity single qubit gates, and thereby reduced the overall Shor algorithm time.

With a solid handle on the single qubit material, we move on to the final chapter, which describes our experiments with the nine quantum element (four phase qubits and five resonators) QuP. Wherein, I will discuss the details of: characterizing all nine-quantum elements via ("swap") spectroscopy in §5.3.1, the fast-entangling protocols and capabilities of the QuP to create two and three-qubit entangled states in §5.4, the compiling of Shor's algorithm in §5.5.1, and the final quantum circuit that we used to find the prime factors of the composite number $N = 15$ with co-prime $a = 4$. I also present the quantum runtime analysis of the algorithm in §5.6, where we show, using quantum state tomography (QST)[59] that we have quantum entanglement over the complete duration of Shor's algorithm in §5.6.1 and in §5.6.2. And finally, we present the output of the algorithm from three experimental perspectives in §5.7: full three-qubit QST, single qubit QST of the output register, and the raw output of the algorithm. We conclude with a check experiment in §5.7.4, where we run the algorithm sans-entanglement, thereby confirming that entanglement is necessary for the success of the algorithm.

This capability to build and operate a nine-quantum element device, and demonstrate a compiled version of Shor's algorithm, represents a significant step toward scaling up to larger numbers of qubits with an architecture that may eventually lead to a quantum computer.

Chapter 2

A Josephson Phase Qubit Quantum Processor

In this chapter I discuss how our superconducting resonators and phase qubits are designed, fabricated, and connected to form a quantum processor (QuP). The chapter begins with a discription of quantum integrated circuit design and a top-level description of the QuP. Next, I provide an overview of the QuP fabrication process (but leave the detailed recipe of the QuP to Appendix B). Then in the following two sections of this chapter, I discuss the individual quantum elements: namely, the linear harmonic oscillator that is physically realized with a superconducting coplanar waveguide (CPW) resonator (R) and the non-linear (anharmonic) oscillator realized with lumped inductors (L), capacitors (C), and

Josephson junctions (JJ) combined to form the phase qubit (Q). The chapter concludes with a discussion on how we physically connect these elements via superconducting wires and capacitors to create a QuP.

2.1 Quantum Integrated Circuits

Our approach to create a QuP leverages the advantages from quantum integrated circuit technology and the success of "off-the-shelf" quantum circuits that are based on superconducting resonant circuits with resonant modes in the GHz range. These quantum circuits are formed using superconducting metals patterned using the same techniques employed in the semiconductor industry to form lumped elements like Ls, Cs, JJs, and extended structures like transmission lines and CPW resonators. With this toolbox of circuit elements combined with superconductivity (a collective quantum behavior of many electrons that allows us to treat the entire circuit quantum mechanically), we can engineer our desired quantum elements rather than relying on naturally occurring quantum systems e.g. photons, spins, or atoms.

2.1.1 Large Qubits

The superconducting resonant circuits composed of lumped Ls, Cs, and JJs form qubits[1] that are relatively large $O(100\times100\,\mu\mathrm{m}^2)$ compared to atoms or ions e.g. Be^+ with a characteristic radii $O(100\,\mathrm{pm})$. At this larger scale it is easier to make physical connections to the qubits. Another advantage of the quantum integrated circuit approach is that we can engineer the impedance of the quantum elements (qubits and resonators) to be $50\,\Omega$. This simplifies the connection between individual elements to a transmission line (a.k.a. a wire) and provides the capability for long range interactions[2].

Since the spatial degrees of freedom are fixed relative to the quantum elements, and they are impedance matched to facilitate long-range interactions, we turn on and off the interactions between the elements in frequency space via electronic control and circuit design. The CPW resonators have a fixed length, and therefore fixed modes in frequency. And although the phase qubits have fixed circuit parameters they can be biased electronically to tune them in and out of resonance with the resonators. The strength of the qubit-resonator interaction is set a capacitor between the qubits and resonators as we discuss in §2.5. In this way the resonator can be thought of as bandpass filter: when the qubit is near or on

[1]The qubits are formed from Ls, and Cs with areas $O(100\times100\,\mu\mathrm{m}^2)$ and JJs with areas $O(1\,\mu\mathrm{m}^2)$.

[2]In the QuP there are interactions between elements that are separated by millimeters.

resonance with the resonator the interaction is turned on and when the qubit is tuned away from resonance the interaction is turned off. For the QuP design the interaction is off when the qubit is tuned $\sim \pm 500\,\mathrm{MHz}$ away from the resonator.

Connecting these macroscopic qubits to the (meandering) extended CPW resonator structures and arranging them in a useful circuit design, like the completed QuP shown in Figure 2.1, results in a quantum circuit occupying an area $6.25 \times 6.25\,\mathrm{mm}^2$.

The dashed-orange rectangle labeled Q_1 in Figure 2.1 highlights a representative phase qubit and its respective superconducting quantum interference device (SQUID) measurement circuit. The meandering blue traces, labeled $R_1 - R_5$ enclosed in a dashed-black rectangle are the CPW resonators. The qubits Q_i are connected to a respective memory resonator R_i and the shared central resonator R_5 with transmission lines and coupling capacitors. The resonator R_5 is a quantum bus [37, 65, 28, 56, 5, 38, 25, 58, 24, 2, 66, 39] that provides the central connection to all of the quantum elements $Q_1 - Q_4$ (and R_1 through R_4 though their respective qubits).

Around the perimeter, interrupting the blue-border in Figure 2.1, there are 12 green pads (4 squares labeled "Q_i Meas" and 8 tapered microwave launchers labeled "Q_i Control" and "R_i Control")[3]. These 12 pads provide electrical

[3]The 8 smaller green rectangles on the right are for test structures and are not connected to the QuP circuit.

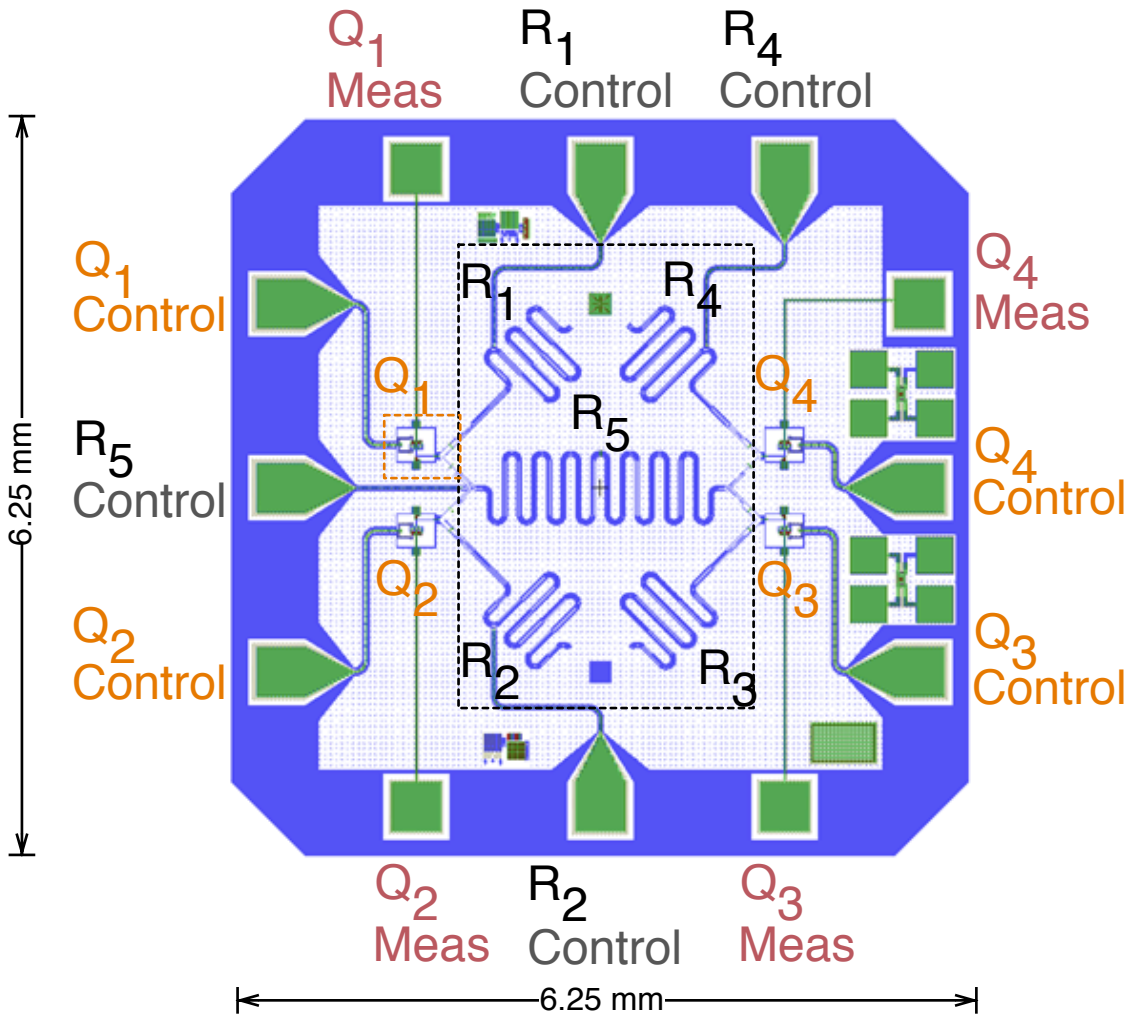

Figure 2.1: CAD layout of Quantum Processor, with phase qubits (resonators) labeled $Q_1 - Q_4$ (R_1-R_5). The orange dotted rectangle indicates a phase qubit cell. The black dotted rectangle encloses all 5 superconducting resonators. External connections are made to the 12 green pads around the perimeter of the chip consisting of 8 control and 4 measure lines.

connections to control and measure the quantum elements.

2.2 The QuP Fabrication

The entire QuP fabrication was performed using standard thin-film deposition and etching technologies all available in the UCSB Nanofabrication Cleanroom. We use Al as the superconducting metal. The Al is lithographically defined to form lumped elements, coplanar waveguide resonators, and transmission lines for electrical connections between elements. The lumped linear inductors are formed by loops of superconducting wires. Interdigitated capacitors are formed by breaks in the Al wire. The parallel-plate capacitors, important for improved qubit coherence times[41], are fabricated by sandwiching hydrogenated amorphous Silicon (a-Si:H) (a low-loss dielectric[51]) between two layers of Al. The a-Si:H dielectric is also used for wiring cross-overs. The Josephson junctions (the triangular shaped wedges in the SEM image in Figure 2.6c)[4] are formed with Al-AlO_x-Al. The entire device is fabricated on a sapphire substrate (black areas in Figure 2.6b) chosen for its low loss properties at GHz frequencies[42]. The fabrication recipe for the QuP is in Appendix B. For more details on our fabrication process please see the thesis by Ansmann[3, chap. 5].

[4]A Josephson junction is formed by interrupting a superconductor with an insulator to form the stack-up: superconductor-insulator-superconductor.

2.3 Superconducting Coplanar Waveguide (CPW) Resonator: Linear Harmonic Oscillator

Resonators comprise one type of quantum element in the QuP. A resonator, or harmonic oscillator is the simplest system that exhibits quantum behavior. Consider the parallel LC circuit shown in Figure 2.2. The voltage $V(t)$ across the circuit drives the currents $I_C(t)$ and $I_L(t)$. Using Kirchoff's current law,

$$I_C(t) = -I_L(t), \tag{2.1}$$

where the current through the capacitor is given by

$$I_C(t) = C\frac{dV}{dt} \tag{2.2}$$

and the current through the inductor is

$$I_L(t) = -\frac{1}{L}\int V(t)dt. \tag{2.3}$$

By substituting Equation 2.2 and 2.3 into 2.1 and differentiating

$$\begin{aligned} \frac{dI_c}{dt} &= -\frac{dI_L}{dt} \\ C\frac{d^2V}{dt^2} &= -\frac{1}{L}V \end{aligned} \tag{2.4}$$

which is the equation of motion for a 1-dimensional simple harmonic oscillator,

$$m\frac{d^2x}{dt^2} = -kx \tag{2.5}$$

with $m \to C$, $x \to V$, $k \to \frac{1}{L}$, and angular frequency $\omega = \sqrt{\frac{1}{LC}}$. The harmonic oscillator solution is known, which gives equally spaced energies, $\Delta E = \hbar\omega$ with energy levels given by

$$E_n = \hbar\omega(n + \frac{1}{2}) \tag{2.6}$$

and illustrated schematically in Figure 2.2. The fact that the energy spacing is degenerate means that all of the levels absorb photons of the same energy, $\Delta E = \hbar\omega$. This makes controlling any of the individual transitions of the resonator challenging with only a classical driving source, which drives the resonator into a coherent state as shown in [23]. However, with energy levels that are spaced unevenly we can drive the individual energy levels separately. In order to engineer a quantum system with energy levels spaced unevenly we need to introduce a non-linear element to perturb the harmonic potential. Fortunately, we can exploit the strong non-linearity of the Josephson effect and create a sufficiently anharmonic potential to form a qubit that, when combined with a classical drive, provides a classical to quantum transducer capable of addressing the individual resonator levels [23].

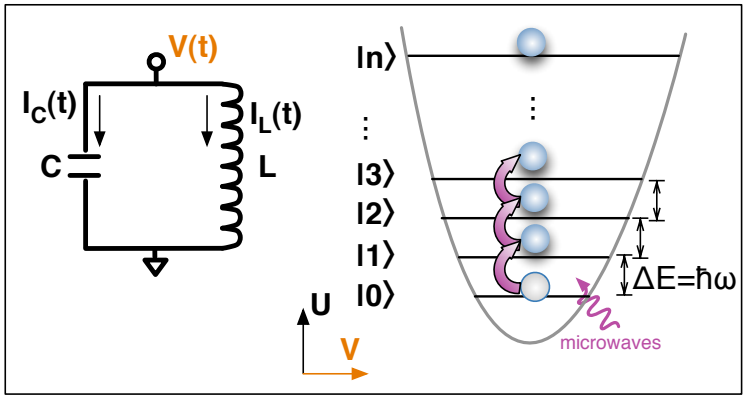

Figure 2.2: Schematic of a Linear LC harmonic oscillator. Potential energy of the harmonic oscillator with evenly spaced energy levels $|0\rangle, |1\rangle, |2\rangle, |3\rangle, \ldots, |n\rangle$.

2.3.1 Half-wavelength CPW Bus Resonator and Quarter-wavelength CPW Quantum Memory Resonators

All of the resonators in the QuP are fabricated as either a half-wavelength ($\lambda/2$) or quarter-wavelength ($\lambda/4$) CPW, which means the resonance frequency can be engineered precisely. The design values used for the five resonators are: $M_1 = 4600\,\mu$m, $M_2 = 4500\,\mu$m, $M_3 = 4550\,\mu$m, $M_4 = 4650\,\mu$m, and $B = 9910\,\mu$m, which are also annotated in Figure 2.3. The $\lambda/2$ CPW labeled "B" in the center of Figure 2.3 is the coupling bus, which mediates the interaction between all four of the phase qubits. I designed the quantum memory resonators, labeled M_1 through M_4 in Figure 2.3 to be $\lambda/4$ CPWs because of the smaller footprint. Although all of the memory resonators could have been designed the same length, I chose to stagger their lengths to simplify the frequency identification experimentally.

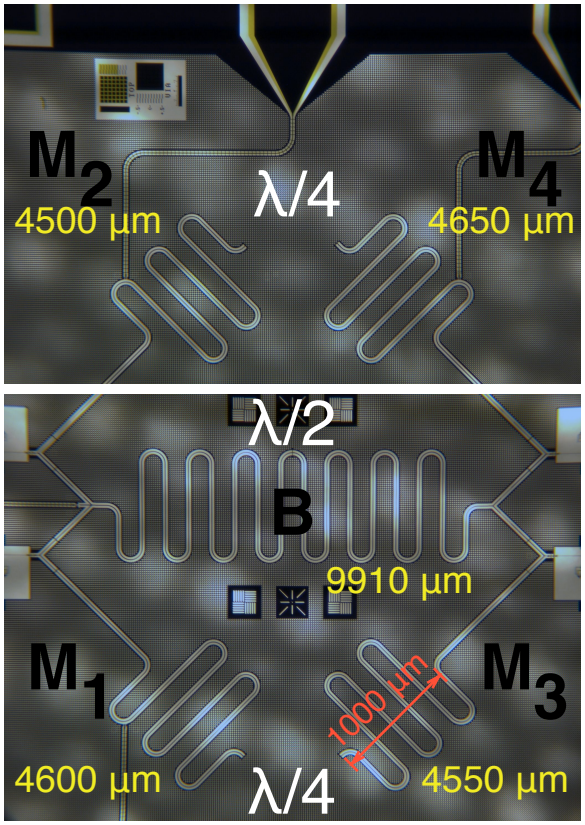

Figure 2.3: Photomicrograph of the 5 (4 quarter-wave and 1 half-wave) CPW resonators. Unwrapped lengths indicated in yellow.

Advantages of CPW Resonators

After the pioneering paper by Hofheinz[23], CPW resonators were integrated into our "off-the-shelf" quantum integrated circuit designs. The CPW resonator provides a number of benefits that we describe below.

CPW resonators have long coherence times ($T_1 \sim 5\,\mu$s and T_2 nearly $2 \cdot T_1$) compared to the phase qubit coherence times ($T_1 \sim 0.5\,\mu$s and $T_2 \sim 0.2\,\mu$s), which makes resonators great candidates for quantum memory. Because CPW resonators can be defined lithographically on the base-layer during our QuP fabrication, we can deposit very high quality superconducting materials via molecular beam epitaxial (MBE) growth methods and begin to incorporate the recent breakthrough materials research results[43] to further extend coherence times.

Another advantage of a CPW resonator is that it can be used as a drop-in coupler to connect *any* number of (phase or other frequency-tunable variety of) qubits. This "quantum bus" architecture also solves the frequency crowding problem of "always on" capacitively coupled qubits by effectively dropping in a band-pass filter so the qubits that are detuned (biased to be idling at a different frequency away from the resonators resonant frequency) will not interact with one another, thus creating a "frequency detuning" type of tunable coupler.[5] This band-pass filter created by the resonator also helps to protect the qubits from both

[5]Although, I note that the frequency detuning "on/off" ratio is inferior to the dynamically tunable coupler as pioneered by Bialczak [12, 11].

microwave and measurement crosstalk[4]. Microwave (measurement) crosstalk occurs when the microwave drive (tunneling event) on the target qubit perturbs any of the other qubits.

2.4 Phase Qubit: Nonlinear, Anharmonic Oscillator

We can engineer a nonlinear anharmonic oscillator by shunting the parallel LC-resonator with a Josephson junction to form the phase qubit as illustrated schematically in Figure 2.4c. The Josephson junction (JJ) has classical current and voltage relations,

$$I_J(t) = I_0 \sin \delta(t) \tag{2.7}$$

$$V_J(t) = \frac{\phi_0}{2\pi} \frac{d\delta}{dt} \tag{2.8}$$

where I_0 is the junction critical current, δ is the phase across the junction, and ϕ_0 is the flux quantum. By taking the canonical voltage relations for an inductor, $V = L_J(dI_J/dt)$, and using Equation 2.7 and 2.8 the Josephson junction can be interpreted as a nonlinear, tunable inductor

$$L_J = \frac{\phi_0}{2\pi I_0 \cos \delta}. \tag{2.9}$$

By using Kirchoff's current law we can analyze the phase qubit circuit in a similar fashion as we did for the parallel LC circuit.

$$I_J(t) + I_C(t) = -I_L(t). \tag{2.10}$$

Inserting the Josephson relations from Equation 2.7 and 2.8 into 2.10 and differentiating we obtain an equation of motion (similar to 2.4)

$$C\left(\frac{\phi_0}{2\pi}\right)^2 \frac{d^2\delta}{dt^2} = -\frac{\partial}{\partial\delta}\left(U(\delta)\right) \tag{2.11}$$

where $m \to C\left(\frac{\phi_0}{2\pi}\right)^2$, $x \to \delta$, and $U(\delta)$ is the potential.

$$U(\delta) = -\frac{\phi_0}{2\pi} I_0 \cos\delta + \frac{1}{2L}\left(\frac{\phi_0}{2\pi}\right)^2 \delta^2 - \frac{\phi_0}{2\pi} I_b \delta \tag{2.12}$$

where I_b is a dc-bias. The potential has three terms. The first term is the nonlinear component $\cos\delta$ from the JJ. The second term $\propto \delta^2$ is from the inductor, and the final term is a constant dc-bias linear in δ. Therefore, by inserting the JJ, we have obtained our desired non-linear, anharmonic potential $U(\delta)$ as illustrated in Figure 2.4d. The energy degeneracy has been lifted with a nonlinearity defined as $\Delta/(2\pi) = \omega_{fe}/(2\pi) - \omega_{eg}/(2\pi)$, which is typically about 200 MHz. This is an important feature in that it allows us to treat this potential as an effective two-level system that can be addressed with a classical drive (shaped microwave pulses) to excite the qubit from the ground state to the excited state (and back

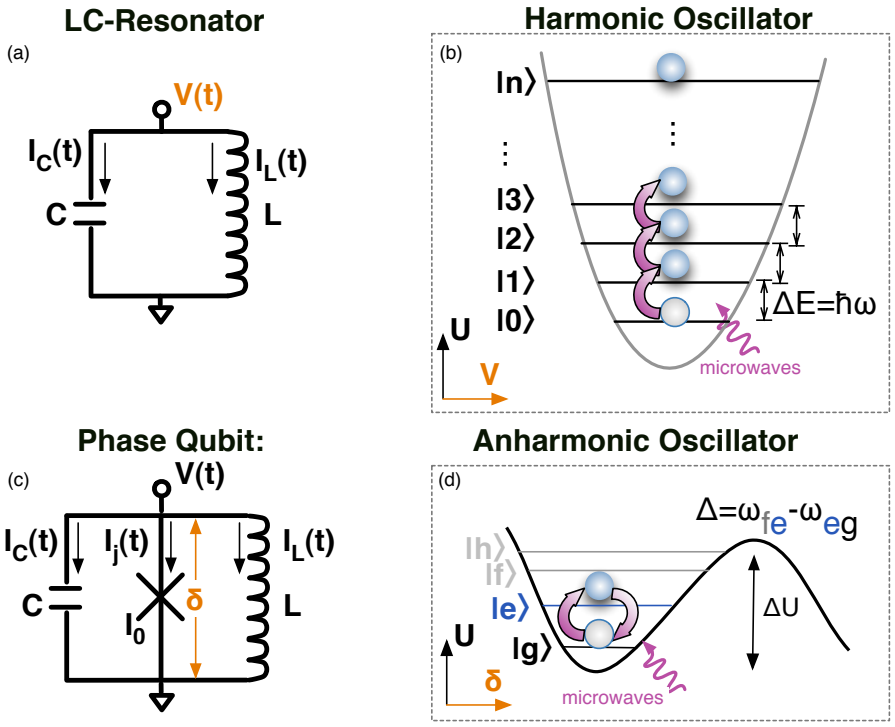

Figure 2.4: (a) Schematic of an LC-resonator. (b) Potential energy of a harmonic oscillator with evenly spaced energy levels. (c) Schematic of phase qubit. (d) Potential energy of the phase qubit with unevenly spaced energy levels.

again) without driving the higher excited states.[6]

The phase qubit circuit is designed so that the potential energy, U as a function of the phase difference δ across the junction forms a double-well potential, as illustrated in Figure 2.5b. The potential is tilted by applying flux via a current I_b through the flux bias loop coupled to the qubit loop to bias the circuit near the critical current I_0 of the junction. This leads to a potential with one very shallow

[6]This strict two-level qubit-manifold assumption is the basis of our next two chapters where we will show how we keep the qubit from "leaking" out into these higher excited states.

well on the left and a deep well on the right. The shallower left-hand well is the qubit well, where the qubit energy $\hbar\omega_{eg} \approx 6\,\mathrm{GHz}$ is set by the inductance L and the capacitance C. The wells are separated by a single flux quantum ϕ_0.

The finite barrier separating the left well from the right arises from the $I_0 \sin\delta$ Josephson relation. The height of this barrier can be adjusted by tilting the potential with the flux bias. Typically the phase qubit is biased to give a barrier height $\Delta U \sim 5\hbar\omega_{eg}$, so that on the order of 5 quantum levels exist in the left-hand well during qubit operations. In addition, the flux bias can be adjusted to tilt the potential, causing Z-rotations, and to measure the qubit state by lowering the barrier to preferentially tunnel the excited state to be readout with a SQUID. For more information about this potential energy description see [3, chap. 2].

2.4.1 Completed Qubit

Four phase qubits were used as quantum elements in the QuP (labeled $Q_1 \ldots Q_4$ in Figure 2.1). Shown in Figure 2.6b is a photomicrograph of a fabricated phase qubit along with its control and readout circuitry. The bottom panel Figure 2.6c is a scanning electron micrograph (SEM) image of the Josephson junction (the essential non-linear circuit element).

The design of the phase-qubit (and SQUID readout) circuit went through a crucial redesign and is detailed in Ref's [45, 48]. Here, I highlight a few of these

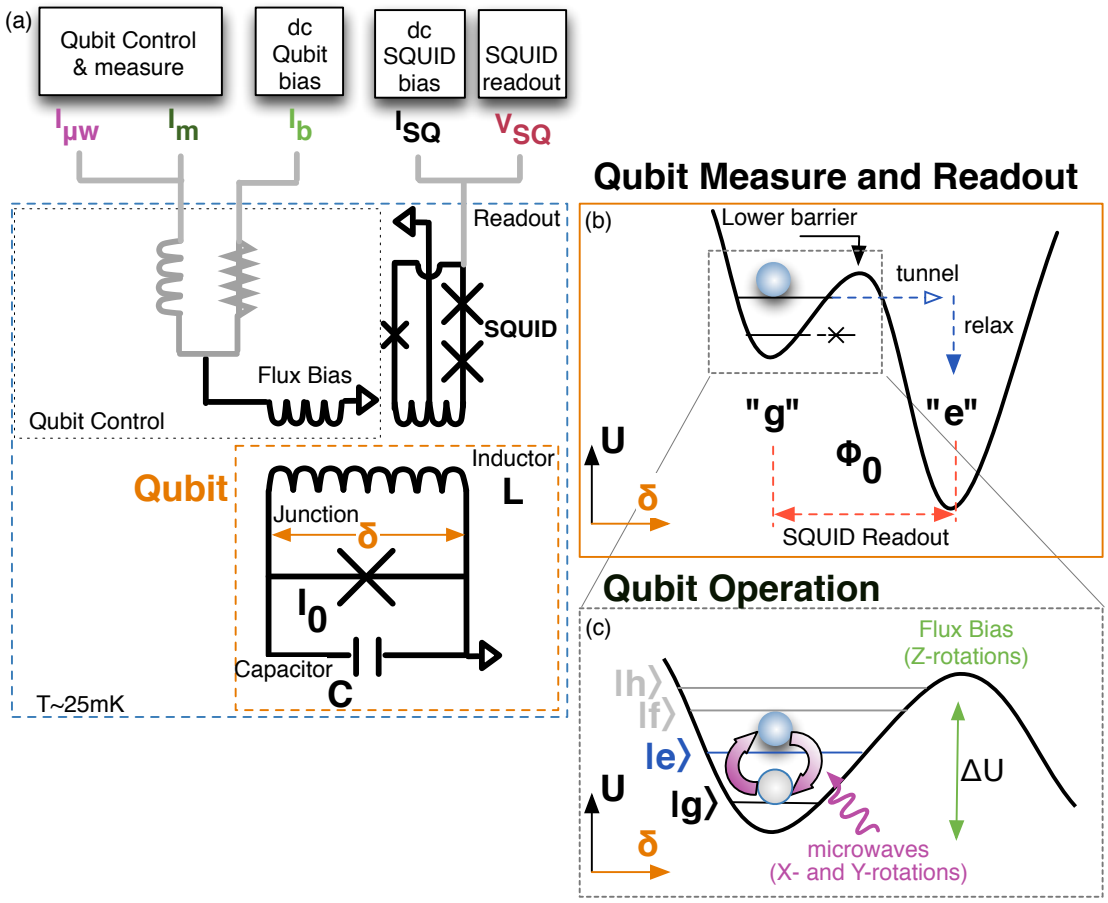

Figure 2.5: (a) Phase qubit schematic with control, measurement, and readout circuitry. (b) Double well potential energy landscape of a phase qubit, illustrating measurement and readout of $|g\rangle$ and $|e\rangle$ states. (c) Qubit operation.

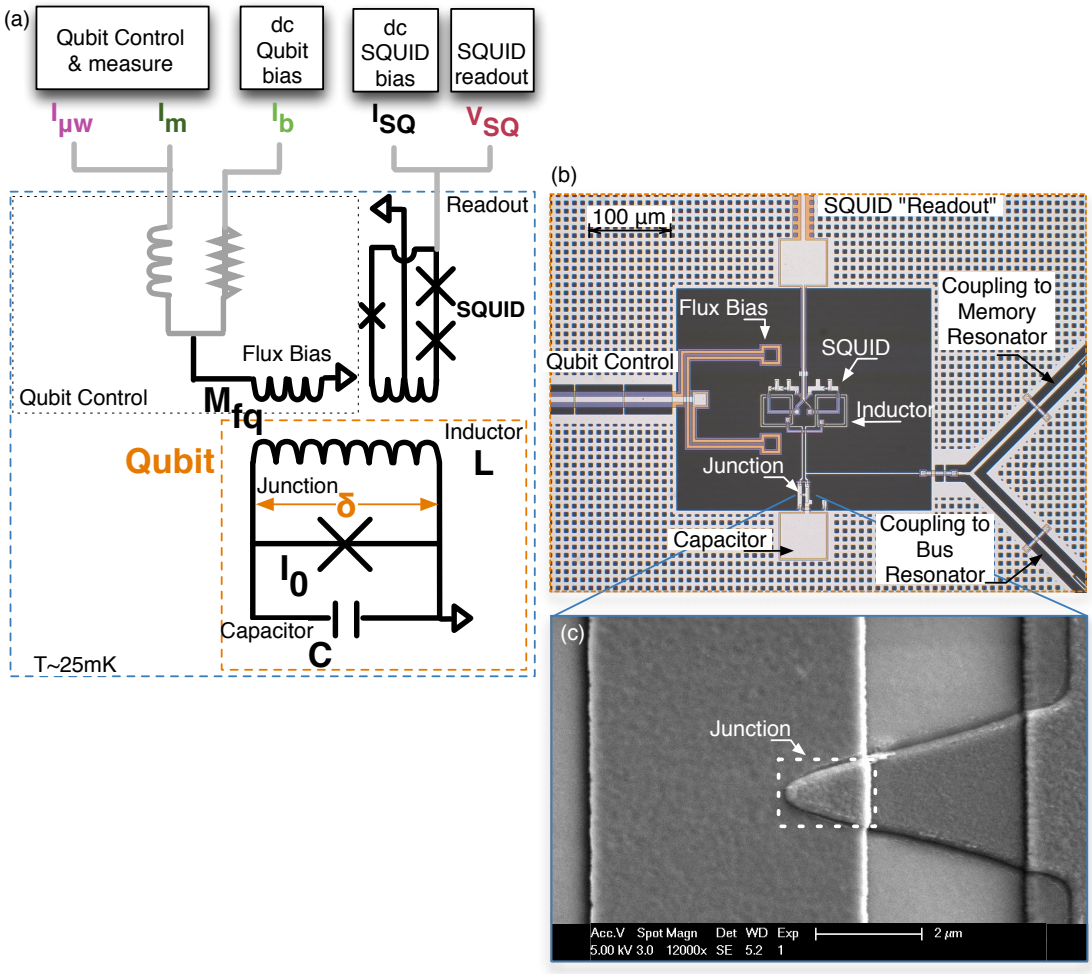

Figure 2.6: (a) The phase qubit schematic. (b) Photograph of completed phase qubit cell with control and readout circuitry annotations. (c) SEM photograph of the Josephson junction.

features and relevant circuit parameters for the device pictured and schematically represented in Figure 2.6. The flux bias coil carries all of the qubit control (microwave and measurement pulses) and bias down one line. This is an improvement over earlier phase qubit devices, which needed 2 control lines. The mutual inductance between qubit and flux bias coil is chosen to be $M_{fq} \approx 2\,\mathrm{pH}$. We aimed for phase qubit frequencies of $\sim 6\,\mathrm{GHz}$ with a junction critical current of $I_0 \approx 2\,\mu\mathrm{A}$ (with an area of $\sim 1\,\mu\mathrm{m}^2$, which is achievable with optical lithography) and lumped parallel-plate capacitance $C \approx 1\,\mathrm{pF}$. Since we want two wells in the qubit potential and having chosen I_0 and C, the inductance L is determined to be $L \approx 720\,\mathrm{pH}$.

This redesign allowed us to drop the completed phase qubit design-cell directly into the QuP design with only minor modifications, mainly in how we connected the phase qubit to the other quantum elements on chip.

2.4.2 Single-Shot SQUID-based Measurement and Readout

We obtain information about our qubits in a two step process, measurement and then readout. This process begins by applying a fast ($\sim 10\,\mathrm{ns}$) pulse to the flux bias line, which briefly lowers the potential barrier between the two wells such that the excited state sees a smaller barrier and will preferentially tunnel into the neighboring right hand well and relax, whereas the ground state will not

tunnel (instead it remains in the left hand well). This measurement process is destructive, which means that at the end of our measure pulse the quantum state is projected in either the left well "g" or right well "e". Recall from our qubit potential discussion that these two wells are separated by approximately a flux quantum and it is this feature that we use for the second part of our information extraction, readout of the state we just measured.

After encoding the left well as "g" and the right well as "e", we use the on chip superconducting quantum interference device (SQUID), as shown in Figure 2.6, to detect this large flux difference. The SQUID transduces this flux into a voltage that we amplify and record at room temperature. The measurement and readout process is illustrated schematically in Figure 2.5b.

The SQUID-based measurement and readout scheme used here is classified as single-shot because every qubit involved in the experiment is projected into a definite state ("g" or "e") upon measurement and the experiment returns one specific state at the end of every experiment. For more details on the SQUID design see [45, 48].

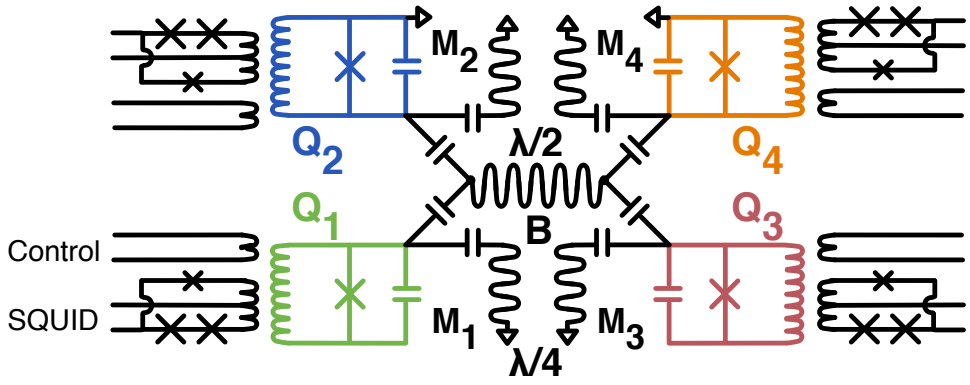

Figure 2.7: QuP schematic with 4 phase qubits, 5 resonators, and 4 SQUIDs.

2.5 Scaling Up: Connecting Multiple Quantum Elements to Form The QuP

Because of the modularity of these superconducting quantum elements and the flexibility of quantum integrated circuit design, we can arrange these quantum elements together in a QuP like in the schematic shown in Figure 2.7. The only element left to discuss is the capacitors that connect everything together.

Due to the low impedance of the phase qubit it is really straightforward to couple them to CPW resonators. In fact, we only need to connect a capacitor between the two. Of course selecting the capacitor value is critical. This qubit - capacitor - resonator circuit, illustrated in Figure 2.8 produces an interaction of the form[27], $H_{int} = (\hbar g/2)(a^{\dagger}\sigma^{-} + a\sigma+)$, where, $a^{\dagger}$ and a are respectively the photon creation and annihilation operators for the resonator, σ^{+} and σ^{-} are

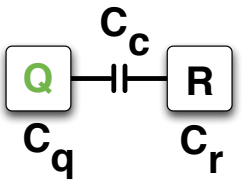

Figure 2.8: Qubit coupled to a Resonator via a capacitor.

respectively the qubit raising and lowering operators, $\hbar = h/2\pi$, and g is the coupling strength given by $g = C_c/(\sqrt{C_q C_r})$. For the qubit-to-bus resonator coupling capacitor $C_c = 4.5\,\mathrm{fF}$, combined with a qubit capacitance $C_q \sim 1.0\,\mathrm{pF}$, and a resonator capacitance of $C_r \sim 50\,\mathrm{fF}$ (at a designed frequency of 6.2 GHz), we expect a coupling strength $g = 55\,\mathrm{MHz}$.

2.6 Experimental Setup and electronics

A summary of the experimental procedure to mount our qubit chips is shown in Figure 2.9. Once the QuP fabrication is complete, we use a diamond saw to dice the 3" wafer into approximately 100 chips of size $6.25 \times 6.25\,\mathrm{mm}^2$. Next, we select the best device from the wafer by probing the test junctions and manually wire-bound it in a specially microwave engineered superconducting box (discussed in detail in [48]). This device is mounted on a Cu-plate of a He^3-He^4 dilution refrigerator and connected to all 12 of the appropriate control lines (qubit control lines Q_1-Q_4, resonator control lines R_1,R_2,R_4,R_5 and SQUID measurement lines

for Q_1-Q_4). The control lines for the resonator drive, qubit control and readout on each qubit are all carefully designed to be well filtered and impedance matched to $50\,\Omega$ to allow for precise and accurate pulse shaping of the control signals. The control lines run from the Cu-plate of the dilution refrigerator all the way to the top of the cryostat where they connect to custom-built control electronics. Finally, the dilution refrigerator is sealed up, evacuated, and cooled down to a base-operating temperature of around $20 - 30\,\mathrm{mK}$.

For more details on the cryogenics and wiring setup see the description by Ansmann[3, chap.6]. For the Shor algorithm experiment we scaled up all of the wiring and electronics to control up to ten qubits (at least two QuP chips) in one cooldown.

2.6.1 Custom Control Electronics

Shown in the bottom of Figure 2.10 is an example of our classical driving source -a Gaussian-shaped microwave pulse taken with a high-speed sampling oscilloscope. These pulses have nearly ideal spectral quality. As shown schematically in Figure 2.10 the pulses are created with a continuous microwave source controlled by an IQ mixer fed by dual 1 GHz digital to analog converters (DAC). The microwave source drives in saturation the local oscillator input of the mixer at frequency f_0. The DAC channels are generated in a custom board using AD9736 chips that

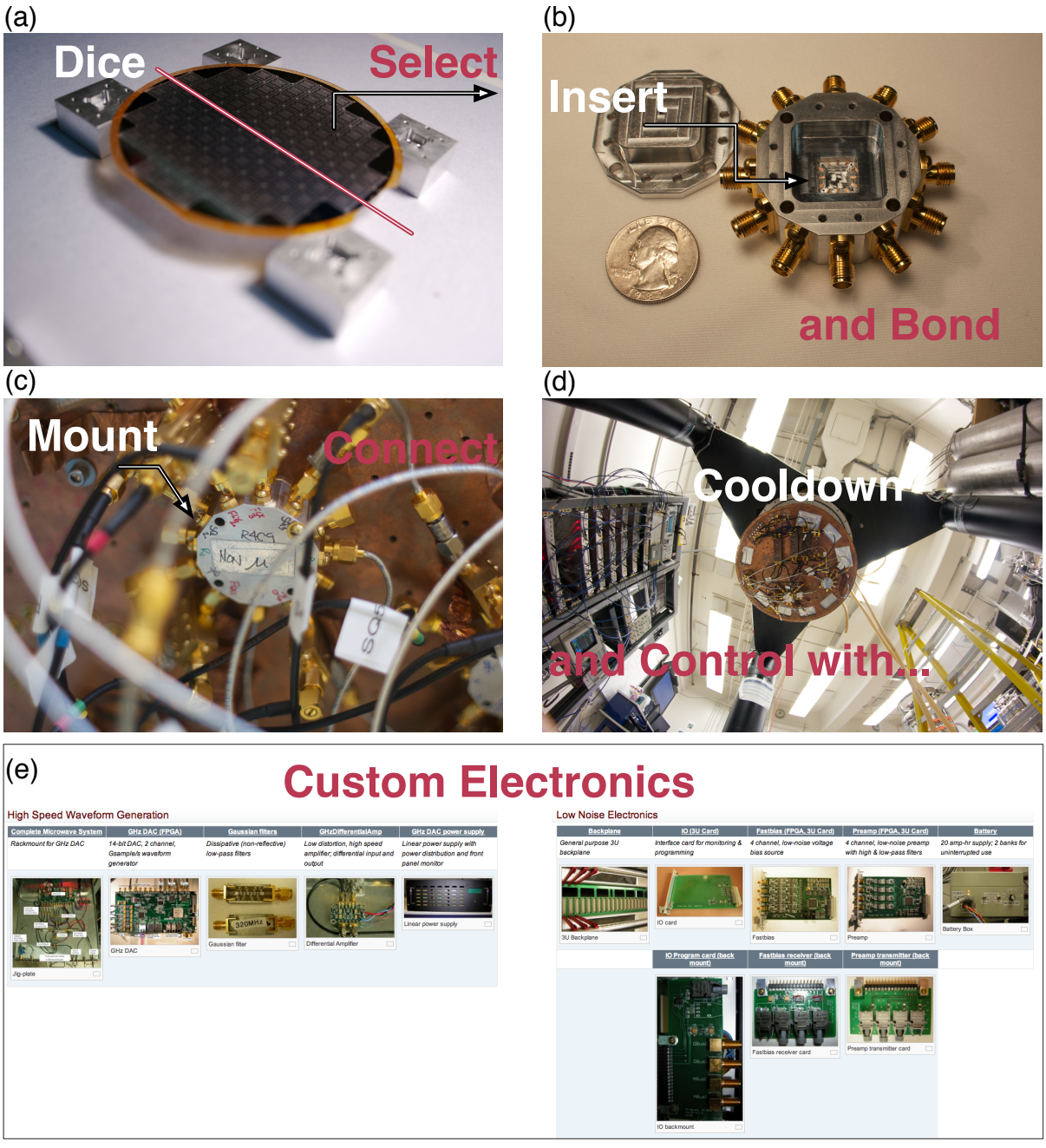

Figure 2.9: Experimental procedure summary as explained in the text.

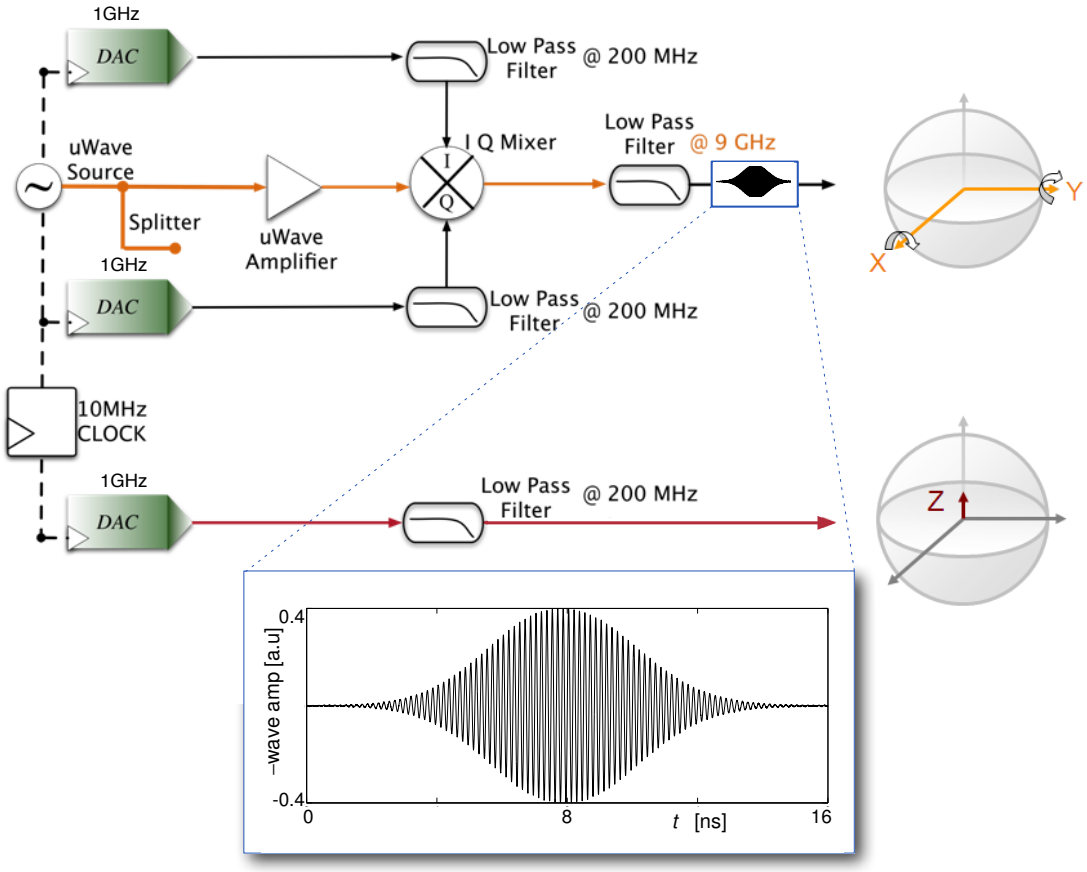

Figure 2.10: Schematic of qubit x-, y-, and z-axis control electronics and an example of an actual Gaussian-shaped microwave pulse measured with a high-speed sampling oscilloscope. Further details are explained in §2.6.1.

have 14 bit resolution. They drive the I and Q ports through 200 MHz (−3 dB frequency) dissipative Gaussian lowpass filters and low distortion differential amplifiers. The microwave output of the mixer is filtered by a 7 pole Chebyshev lowpass filter at 8.5 GHz to suppress harmonics of f_0. The large bandwidth of the control signal allows for sideband mixing. By applying sine and cosine waves at $f_{\rm sb}$ to the I and Q ports, the mixer generates an output signal at frequency $f_0 + f_{\rm sb}$. Sideband mixing allows for very high on/off ratios of qubit control since the (small) carrier leakage at f_0 is off resonance with the qubit. The digital control allows imperfections of the DAC chain and the IQ mixer to be corrected by first measuring its response function and then correcting it with deconvolution. The relative amplitudes and phases of the I and Q mixer channels are calibrated by minimizing the power at the opposite sideband $f_0 - f_{\rm sb}$. This is done at enough sideband frequencies so that all Fourier component of an arbitrary digital input signal can be corrected. In total, we obtain accurate pulse shapes with greater than 60 dB suppression of spurious frequencies and harmonics.

Qubit Control

Single qubit logic operations, corresponding to rotations about the x-, y-, and z-axes of the Bloch sphere, are generated as follows: Rotations about the z-axis are produced from current pulses on the qubit flux bias line that adiabatically

change the qubit frequency, leading to phase accumulation between the states $|g\rangle$ and $|e\rangle$ [6]. Rotations about any axis in the x-y plane are produced by microwave pulses resonant with the qubit transition frequency. The phase of the microwave pulses defines the orientation of the rotation axis in the x-y plane, and the pulse duration and amplitude control the rotation angle.

A note on our custom built electronics: We have made our custom electronics available to the public on the UCSB QC-group's TWiki[7], including our ever-growing body of knowledge related to the electronics that we have built. I am pleased that these custom electronics have been deployed all over the world in a number of laboratories and I look forward to their extended use in further research.

[7] https://commando.physics.ucsb.edu/tw/view/Electronics/PubDocs

Chapter 3

Reducing Unwanted Transitions Into The Phase-Qubit's $|f\rangle$ State: Amplitude Errors

As we learned in Chapter 2 the phase qubit has more than just two levels, but the non-linear inductance from the Josephson junction removes the energy level degeneracy, thereby allowing a classical microwave drive pulse (like the one generated in Figure 2.10) to address the separate transitions[46] including the lowest two-levels $|g\rangle$ and $|e\rangle$ used for a qubit. To illustrate the importance of this control issue, we note that many experimental systems use qubit states $|g\rangle$ and $|e\rangle$, often the ground and first excited states, chosen from a larger set of basis states

[49, 46, 15, 21, 13, 18, 70, 67, 26, 53, 52, 40]. This encoding does not preclude unwanted excitations to other available states in the basis. For example, excitations to the next higher energy state $|f\rangle$ are not necessarily small and correspond to gate errors that may not be included in standard single qubit measurements like T_1 and T_2.

Measuring only T_1 and T_2 assumes no loss in fidelity *during* a logic gate operation when the quantum state is changed, and thus it more properly corresponds to the fidelity of a memory operation. Therefore, a full measurement of gate fidelity, applicable to the fault-tolerance threshold, should include gate errors that are determined via probabilities with an absolute calibration. To that end we look to construct individual experiments that can highlight separate error cases and then build up their collective action to understand the complete fidelity of a gate operation.

Therefore, in this chapter, I will describe how we separate out measurement errors from gate errors in §3.1. Then in §3.2, I will discuss a metrology technique based on a Ramsey interference pulses sequence that enhances a particular error source, namely qubit excitations to the $|f\rangle$ state, and finally the chapter concludes with a single qubit gate fidelity measurement in §3.3.

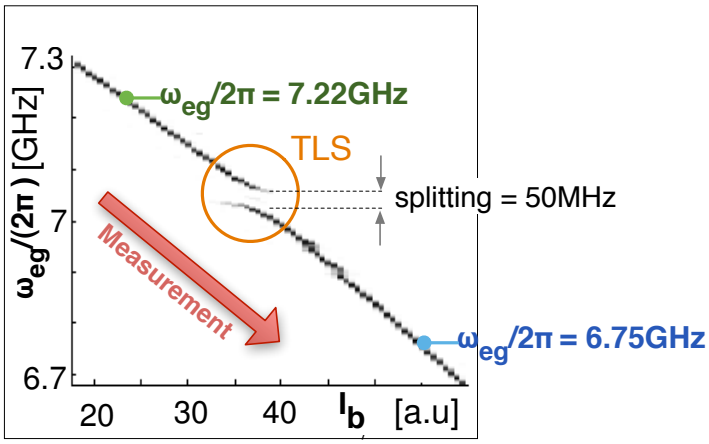

Figure 3.1: Qubit spectroscopy. Probability of tunneling is plotted in grayscale for qubit operating frequency $\omega_{eg}/(2\pi)$ versus qubit bias I_b. A two-level state (TLS) splitting shown at 7.1 GHz.

3.1 Probability Errors From Measurement

Non-ideal behavior of the qubit can arise from errors related to the qubit control or in the state measurement. First, let's focus on the physical mechanisms that lead to measurement errors. In phase qubits, measurement fidelities below unity are due to stray tunneling of the $|g\rangle$ state, the $|e\rangle$ state leaking energy to spurious two-level states (TLS)[16], and T_1 relaxation.

I first discuss errors due to stray transitions to the spurious TLS. Qubit spectroscopy is shown in Figure 3.1, where the probability of tunneling is plotted in grayscale for qubit frequency $\omega_{eg}/(2\pi)$ that is changed via the qubit bias I_b[16]. A TLS gives a resonance at 7.05 GHz that couples to the qubit with splitting size 50 MHz. To quantify the TLS effects as measurement errors, we

determined the measurement fidelity above ($\omega_{eg}/(2\pi) = 7.22\,\mathrm{GHz}$) and below ($\omega_{eg}/(2\pi) = 6.75\,\mathrm{GHz}$) it; this large TLS splitting at 7.05 GHz is highlighted in orange in Figure 3.1.

The measurement data for the qubit operated above and below the TLS are plotted in Figure 3.2. For each data set, the tunneling probability of the ground state $|g\rangle$ and the first excited state $|e\rangle$ is determined versus measurement pulse amplitude I_z. The inset in the left-panel of Figure 3.2 illustrates the pulse sequence. For the $|g\rangle$ state we apply no microwaves. For the $|e\rangle$ state experiment the X pulse is calibrated for a π-rotation to give maximum probability of the $|e\rangle$ state. The tunneling probability P_{tunnel} for the $|g\rangle$ and $|e\rangle$ state is determined versus I_z. After this calibration, I_z is chosen to give maximum visibility between the states, which is displayed in each figure by an arrow.

The difference in visibility observed between the two qubit operating frequencies is directly attributed to coupling to the TLS located at 7.05 GHz, as observed in Figure 3.1. The measurement pulse lowers the barrier for increased tunneling probabilities of the excited state, but it also reduces the qubit operating frequency (as illustrated by the"measurement" arrow in Figure 3.1) and in the case for $\omega_{eg}/(2\pi) = 7.22\,\mathrm{GHz}$ the qubit is swept through the TLS at $7.05\,\mathrm{GHz}$.

The theoretical predictions for the tunneling probabilities are given by the solid black and gray lines Figure 3.2. Theory predicts that the $|g\rangle$ state is misidentified

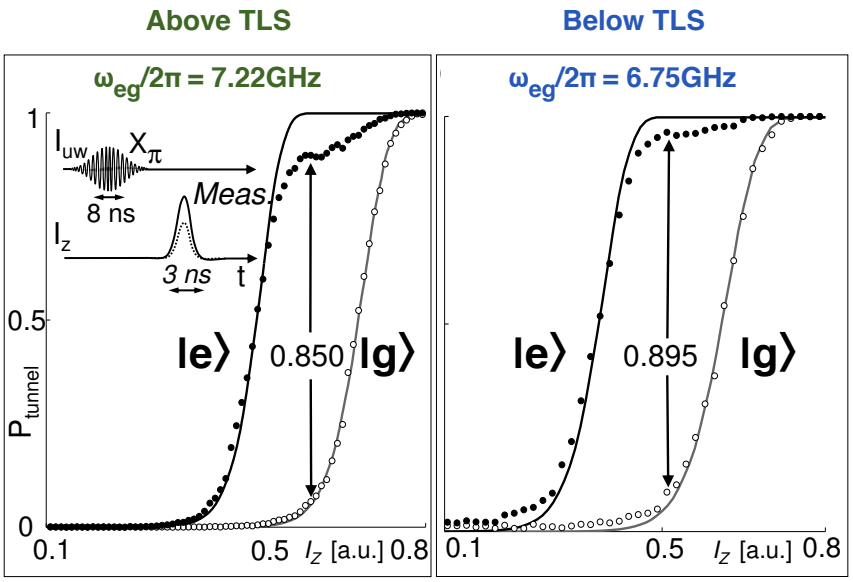

Figure 3.2: Quantifying measurement error due to TLS. Data are described in text.

as a $|e\rangle$ state with a probability of 0.034. This error is consistent with theory, and corresponds to stray tunneling events during measurement[16]. At $\omega_{eg}/2\pi =$ 6.75 GHz the $|e\rangle$ state is misidentified as the $|g\rangle$ state with a probability of 0.061, but at a higher qubit frequency, $\omega_{eg}/2\pi = 7.22$ GHz this error increases to 0.106. The increase in measurement error with qubit frequency is attributed to the TLS located between these two frequencies. With a measurement of the TLS splitting from spectroscopy of size 50 MHz, we predict a $|e\rangle$ state population decrease of 0.045, a value consistent with our data.

The remaining measurement error is accounted for with an error budget of 0.010 for T_1 decay, 0.050 for coupling to other TLS, and 0.011 for no tunneling

of the $|e\rangle$ state during measurement. With good agreement between experiment and theory, we can reliably account for measurement errors in our data.

3.2 Amplitude Errors Due to Qubit Population Leaking Into The $|f\rangle$ State

We now turn to measuring and reducing errors from the qubit leaking into the $|f\rangle$ state. There is a tradeoff between using a fast pulse for small T_1 errors due to qubit decay, or a slow pulse for small Fourier amplitude at the $|e\rangle \rightarrow |f\rangle$ transition frequency, as illustrated in the inset Figure 3.5. A short Gaussian pulse, FWHM 4 ns, produces power at the transition frequency $\omega_{fe}/2\pi$ which drives transitions outside the qubit manifold causing qubit leakage error. Therefore, we aim to find the optimal length pulse that optimizes the tradeoff between qubit relaxation and leakage.

The measurement of the qubit leaking into the higher excited state is explicitly shown in Figure 3.3, where the probability to tunnel the qubit P_{tunnel} is plotted versus the measure pulse amplitude I_z for a single X_π-pulse using $4, 5$, and 8 ns FWHM Gaussian pulses with the theory lines labeled for the three lowest states $|g\rangle, |e\rangle$, and $|f\rangle$. Just as in Figure 3.2 the data on the right in Figure 3.3 labeled "$|g\rangle$" are for the case where no microwaves are applied. To preferentially tunnel

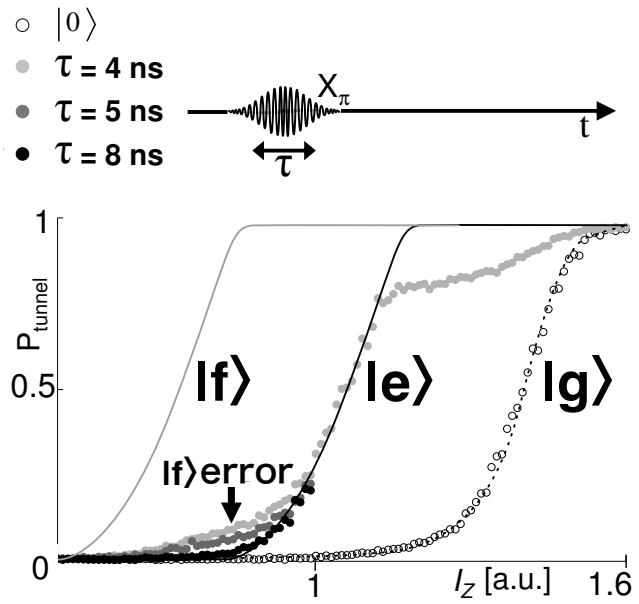

Figure 3.3: (top) Experimental pulse sequence. Data are direct measurements of the $|f\rangle$ error due to for $\tau = 4, 5, 6$ ns FWHM Gaussian X_π-pulses. Further detail in the text.

the higher (lower) excited states a smaller (larger) I_z amplitude is applied.

Focusing on the middle curve labeled "$|e\rangle$", an X_π pulse of length $4, 5$, and 8 ns is applied to prepare the excited state. The shoulder, labeled "$|f\rangle$ error", rising at smaller I_z amplitudes is the leakage error. Errors become difficult to measure below ~ 0.01 because of stray tunneling of the $|e\rangle$ state.

The $|f\rangle$ state error may be measured with much greater sensitivity by recognizing that excitation to the $|f\rangle$ state is a coherent quantum process. Using a two-pulse sequence with variable time delay as illustrated in Figure 3.4a, a Ramsey fringe may be set up between the transitions to the $|f\rangle$ state from the two

pulses. In addition most of the qubit state is in $|g\rangle$, so there is little stray tunneling from $|e\rangle$. The two X_π-pulses (of duration $\tau = 5\,\text{ns}$) are followed by a measure pulse with an amplitude calibrated to tunnel only the $|f\rangle$ state. During the first X_π-pulse both of the states $|e\rangle$ and $|f\rangle$ are excited. The second X_π-pulse causes the coherent beating of the $|f\rangle$ state.

In Figure 3.4b we plot the $|f\rangle$ state probability P_f versus pulse delay time t_{sep}. Since the periodic oscillation is due to coherent interference between the two pulses, the magnitude of this oscillation is four times the probability of exciting the $|f\rangle$ state for a single pulse[1]. More importantly, creating an oscillating signal of a constant error allows a determination of the amplitude with fewer systematic errors; this error can now be reliably measured down to 10^{-4} using this "Ramsey error filter". As a further check the oscillation frequency matches the beat frequency $(\omega_{eg} - \omega_{fe})/2\pi$ measured via spectroscopy, as shown in Figure 3.4c.

This Ramsey error filter data was repeated for 4, 5, 6, 6.5, 7, 7.5, and 8 ns FWHM Gaussian pulses. The $|f\rangle$ state errors determined in this manner are also plotted in Figure 3.5. For Gaussian pulses with width 4 and 5 ns, the data from the two methods overlap. The error drops exponentially with increasing pulse width, reaching the value 10^{-4} at 8 ns (where a magnitude of 10^{-4} is considered to be an error-threshold for fault tolerance). A simple Fourier-transform prediction[61]

[1]The first pulse populates the $|f\rangle$ with some amplitude, followed by the second pulse, which coherently doubles the population, and since we measure a probability these amplitudes are squared, hence four times the magnitude compared to the error from a single pulse.

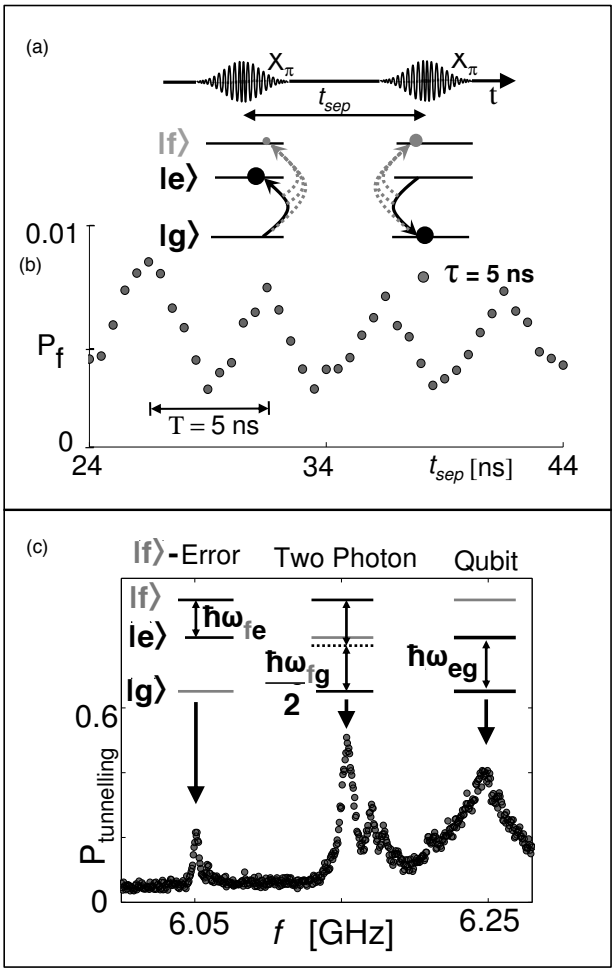

Figure 3.4: (a) Pulse sequence for the Ramsey error filter (REF) with Illustration of three-level system and transitions into the $|f\rangle$ state during X_π-pulses. (b) Probability of measuring the $|f\rangle$ state P_f versus X_π-pulse separation, t_{sep}. (c) High-power spectroscopy showing the higher transition states.

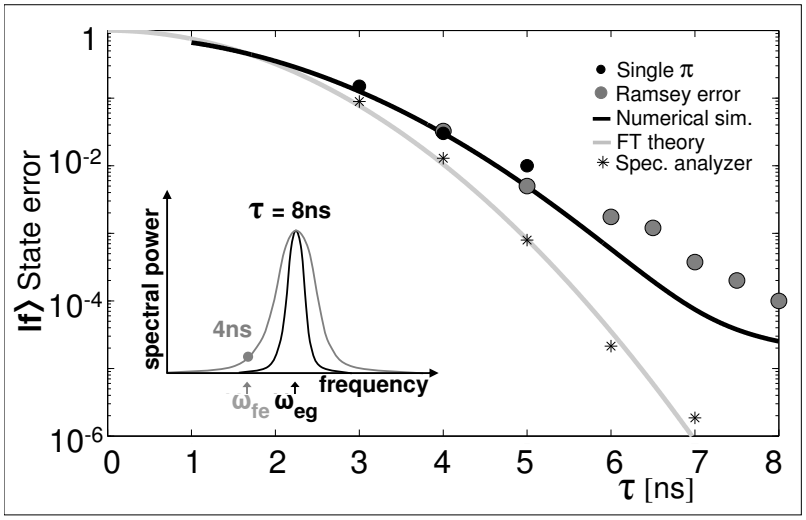

Figure 3.5: $|f\rangle$ state error versus τ FWHM pules. Inset illustrates non-zero spectral power at ω_{fe} for a 4 ns Gaussian pulse.

is plotted as a solid gray line, which is computed from the power spectrum of the Gaussian pulse at frequency $\omega_{fe}/(2\pi)$, normalized to the power at frequency $\omega_{eg}/(2\pi)$. The asterisks are a measurement of this normalized power taken from the actual control pulses. This simple comparison is an excellent check on the shaping of the microwave pulses. From this data, we can see that short pulses with a wide frequency spectrum gives large qubit error, this is illustrated in the inset of Figure 3.5 where a 4 ns pulse produces a significant amount of spectral power at $\omega_{fe}/(2\pi)$. The solid black line is a prediction of the error obtained from numerical calculations [61], which shows good agreement with the data.

3.3 High Fidelity Gates

To reduce the errors caused by unwanted $|f\rangle$ state transitions, we apply a shaped pulse significantly long enough so as to minimize the spectral components at the $|e\rangle \rightarrow |f\rangle$ transition frequency. This keeps the qubit within the two-state manifold, with residual $|f\rangle$ state population on the order of 10^{-4}. For example, a qubit frequency of $\omega_{eg}/(2\pi) = 6.5\,\text{GHz}$, and a nonlinearity $\Delta/(2\pi) = (\omega_{fe} - \omega eg)/(2\pi) = -200\,\text{MHz}$ ($\omega_{fe} = 6.3\,\text{GHz}$) requires an 8 ns length π-pulse to keep the $|f\rangle$ leakage to 10^{-4}. With this pulse length and the error budget from Section §3.1, we are poised to measure our single qubit gate fidelity.

Because the measurement error for the $|g\rangle$ state is less dependent on systematics, we choose to measure our logic gate performance by returning the qubit to the $|g\rangle$ state. And since a π-pulse is the maximum rotation of a single qubit operation conducting a pulse sequence involving π-pulses gives a measure of the maximum error for a gate. Therefore, we determine the fidelity of a gate by applying two π-pulses that produce the transitions $|g\rangle \rightarrow |e\rangle \rightarrow |g\rangle$, followed by measurement. This pulse sequence is illustrated in Figure 3.6a.

We apply 8 ns FWHM X_π-pulses similar to the one represented in Figure 2.10. We verify that we are indeed performing X_π-pulses, by testing whether the probability for the final state is independent of the phase Θ between the two microwave pulses, as indicated in Figure 3.6a. The two panels in Figure 3.6b show the ex-

perimental and theoretical probability of being in the excited state P_e (color bar) as a function of Θ and microwave detuning Δ from the qubit transition frequency $\omega_{eg}/2\pi$. The experimental data is in excellent correspondence with theoretical predictions. On resonance ($\Delta = 0$), the phase Θ has no effect, as expected, which demonstrates that the two pulses are calibrated properly as π-pulses.

Gate error is directly measured by repeating this experiment with variable time separation $t_{\rm sep}$ between the two π-pulses, as shown in Figure 3.6c. The gate error grows with increasing time $t_{\rm sep} > 9$ ns because the $|e\rangle$ state decays, and the error has a slope consistent with separate measurements of T_1. The error also increases at small times due to the overlap of the two Gaussian microwave pulses. The horizontal dashed line indicates $P_1 = 0.034$ taken without the application of microwaves; the difference between the data and the dashed line is the gate error. When the pulses are separated by a time $t_{\rm sep} = 12$ ns, we find an error $\Delta P_1 = 0.04$. Since two gate operations are used for this protocol, the fidelity for a single gate operation is 0.98.

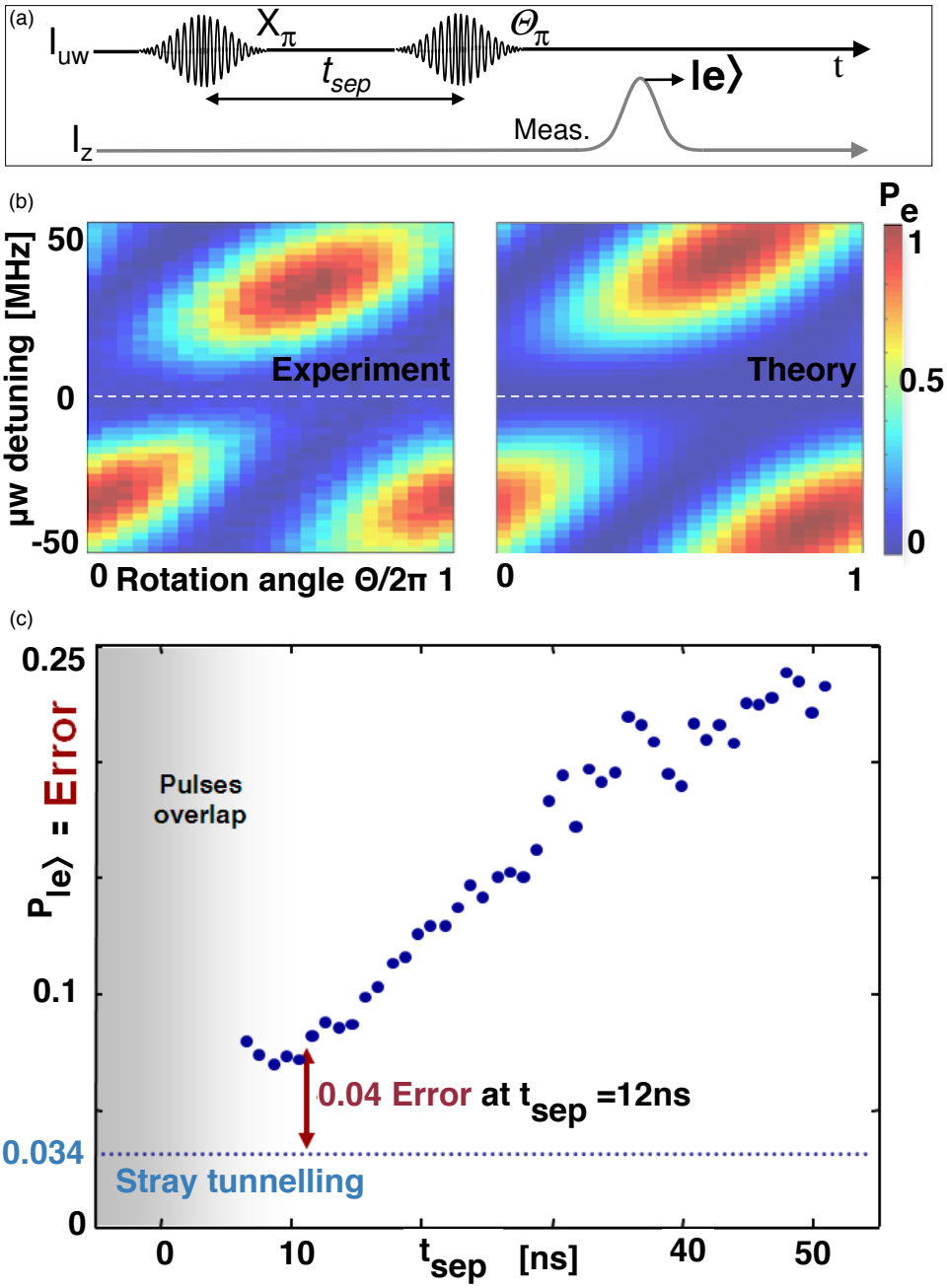

Figure 3.6: High Fidelity Gate Data as explained in text.

Chapter 4

Reducing Unwanted Virtual Transitions Into The Phase-Qubit's $|f\rangle$ State: Phase Errors

In Chapter 3, I showed how we reduced the amplitude errors associated with transitions outside of the qubit manifold by careful shaping of the control pulse (microwave envelope) and by choosing the correct gate duration, which scales inversely with the qubit nonlinearity $1/\Delta$. However, fast pulses also generate phase errors which contribute to overall gate error, but the relative contribution differs from amplitude-related errors. Consequently different measurement techniques need to be employed to quantify phase errors.

To measure possible phase error, quantum process tomography (QPT) is typically used. QPT provides a complete analysis of gate operation [45, 12, 49], but it requires $X_{\pi/2}$ and $Y_{\pi/2}$ pulses, which themselves can be error sources. This poses a circular problem: quantum process tomography relies on $\pi/2$-pulses, but we need quantum process tomography to verify that we have tuned up our $\pi/2$-pulses. To solve this dilemma, we designed a method to separately quantify the phase error generated by a gate, which we call Amplified Phase Error (APE). By using a Ramsey fringe experiment, we amplified and measured this ubiquitous source of error.[1] We chose to focus on errors related to $\pi/2$ pulses, because such pulses provide the basis for tomography and are essential in algorithms.

This chapter begins with a model, expressed in quantum circuit language, for phase errors due to virtual transitions into the higher excited states outside of the qubit manifold. Using this quantum circuit language the chapter proceeds onto a discussion of how to amplify phase errors in §4.1.1, measure in §4.1.2, and correct them in §4.1.3. To correct for phase errors, specific to virtual transitions outside of the qubit manifold, we implement "half derivative" an experimental simplification of derivative reduction by adiabatic gate (DRAG) control theory [44]. This solution uses two control pulses (X and Y simultaneously) and was born out of the theory proposed by Motzoi[44]. In §4.2 we revisit amplitude errors

[1]The APE protocol is not specific to phase qubits, it can be used in general multi-level qubit architectures [49, 46, 15, 21, 13, 18, 70, 67, 26, 53, 52, 40].

associated with HD pulses. And the chapter concludes in §4.3 with demonstrations of the improved control from HD pulses.

4.1 Phase Errors Due to Virtual Transitions

The phase error arising from virtual transitions (especially to the $|f\rangle$ state) can be modeled as effective qubit rotations about the z-axis. We restrict ourselves to simple gates comprising π and $\pi/2$ rotations. An $X_{\pi/2}$ pulse (a rotation about the x-axis by an angle $\theta = \pi/2$) ideally produces the transformation

$$X_{\pi/2} = e^{-i\sigma_x \frac{\pi}{4}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} , \tag{4.1}$$

where σ_x is one of the Pauli matrices.

To test this for an experimental system with more than two levels, we integrate the Schrödinger equation to explicitly calculate the time evolution for an arbitrary input state, which is described by a 3×3 unitary matrix U (to include the effects from the $|f\rangle$). With a Gaussian control pulse, we find the elements of U that connect the $|g\rangle$ or $|e\rangle$ state with the $|f\rangle$ state have small magnitude, consistent with the negligible $|f\rangle$ state error measured in Chapter 3. Therefore, the time evolution of the two qubit states is well described by the 2×2 submatrix of U. For more details on the qubit numerical simulatuons, see Ansmann[3, chap. 3]. From these numerical simulations we find that for small phase errors, this

submatrix can be expressed in quantum circuit language as

$$X'_{\pi/2} = e^{-i\epsilon'} Z_\epsilon X_{\pi/2} Z_\epsilon \ , \tag{4.2}$$

where Z_ϵ is the phase error of interest and $0 < \epsilon \ll 1$ [2].

$$Z_\epsilon \ = \ \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\epsilon} \end{pmatrix} . \tag{4.3}$$

Note that Equation 4.2 differs from $X^\star_{\pi/2} = Z_{-\epsilon} X_{\pi/2} Z_\epsilon$, which corresponds to a rotation about a new axis ϵ away from the x-axis in the $x - y$ plane.

4.1.1 Amplifying Phase Error

In order to best measure this error, we first sought a protocol that would amplify the error ϵ. For an arbitrary rotation θ about the x-axis, the gate operation is

$$X_\theta = \begin{pmatrix} \cos\theta/2 & -i\sin\theta/2 \\ -i\sin\theta/2 & \cos\theta/2 \end{pmatrix} \ , \tag{4.4}$$

such that $X'_{\pi/2}$ is

$$X'_{\pi/2} \ = \ \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -ie^{-i\epsilon} \\ -ie^{-i\epsilon} & e^{-i2\epsilon} \end{pmatrix} . \tag{4.5}$$

[2] The leading term in Equation 4.2 is a global phase and can be ignored.

If we consider a 2π rotation generated by concatenating four $\pi/2$ pulses, this results in

$$\begin{aligned} X'^4_{\pi/2} &\equiv (X'_{\pi/2})^4 \\ &= \frac{1}{4}\begin{pmatrix} e^{-6i\epsilon}(-1-e^{2i\epsilon}-3e^{4i\epsilon}+e^{i\epsilon}) & -ie^{-7i\epsilon}(-1+e^{2i\epsilon})^2(1+e^{2i\epsilon}) \\ -ie^{-7i\epsilon}(-1+e^{2i\epsilon})^2(1+e^{2i\epsilon}) & e^{-8i\epsilon}(-1+3e^{2i\epsilon}+e^{4i\epsilon}+e^{6i\epsilon}) \end{pmatrix} \\ &\simeq e^{-i4\epsilon} I\,, \end{aligned} \tag{4.6}$$

where I is the identity. Equation (4.6) shows that a concatenated 2π rotation does not accumulate a relative phase error it only acquires a global phase.

We next examine the pseudo-identity operation that is formed by concatenating positive and negative θ rotations. For a first-order expansion with $\epsilon \ll 1$ we find

$$\begin{aligned} I'_\theta &= (Z_\epsilon X_\theta Z_\epsilon)(Z_\epsilon X_{-\theta} Z_\epsilon) \\ &\approx \begin{pmatrix} 1+i(\cos\theta-1)\epsilon & -(\sin\theta)\epsilon \\ (\sin\theta)\epsilon & 1-i(\cos\theta+3)\epsilon \end{pmatrix}\,, \end{aligned} \tag{4.7}$$

where X_θ is an arbitrary rotation of θ about the x-axis. For $\theta = \pi$ we find that $I'_\pi = e^{-2i\epsilon'} I$, which is similar to the 2π rotation, as the phase error ϵ cancels. However, for $\theta = \pi/2$ we find,

$$\begin{aligned} I'_{\pi/2} &= X'_{-\pi/2} X'_{\pi/2} \\ &= \begin{pmatrix} e^{-i\epsilon}\cos(\epsilon) & e^{-2i\epsilon}\sin(\epsilon) \\ -e^{-2i\epsilon}\sin(\epsilon) & e^{-3i\epsilon}\cos(\epsilon) \end{pmatrix} \\ &\approx \begin{pmatrix} 1-i\epsilon & \epsilon \\ -\epsilon & 1-3i\epsilon \end{pmatrix}. \end{aligned} \tag{4.8}$$

Where X' is defined in Equation 4.5. For n applications of the pseudo-identity operation, in the limit where $0 < \epsilon \ll 1$, $\epsilon \to n\epsilon$

$$I'^{n}_{\pi/2} \approx \begin{pmatrix} 1 - in\epsilon & n\epsilon \\ -n\epsilon & 1 - 3in\epsilon \end{pmatrix}, \tag{4.9}$$

focusing on the relative phase along the diagonal elements, and by removing an overall global phase

$$I'^{n}_{\pi/2} \approx (Z_{2\epsilon})^n = Z_{2n\epsilon}\,, \tag{4.10}$$

shows a phase error accumulation. Thus by rotating back and forth with $X'_{\pi/2}$, $X'_{-\pi/2}$ operations the state accumulates phase errors, which can be measured.

4.1.2 Measuring Phase Error

To measure this error, we combine the result from Equation 4.10 with a phase measuring experiment, forming what we call an amplified phase error (APE) sequence. The APE sequence consists of inserting $n \in \{0, 1, 3, 5\}$ successive $I'_{\pi/2}$ pseudo-identity operations between the $\pi/2$ pulses that define a Ramsey fringe measurement, as illustrated in the left panel of Figure 4.1. All control pulses are separated in time by 2τ and at the end of the sequence the Z control line is pulsed to measure the probability of $|e\rangle$. The phase error is amplified by $2n$ for n

applications of the pseudo-identity operation,

$$I'^{n}_{\pi/2} \approx (Z_{2\epsilon})^n = Z_{2n\epsilon} \ . \tag{4.11}$$

By applying APE pulses to the state $|\psi\rangle = (|g\rangle - i\,|e\rangle)/\sqrt{2}$ followed by a final $\phi_{\pi/2}$ pulse, we directly probe the phase error due to the $X_{\pi/2}$ pulses. The data for the single-control (X-quadrature Gaussian shaped) APE pulse sequence are shown in Figure 4.1 along with a Bloch sphere indicating the axis of rotation and the three-level system illustrating the phase error due to virtual transitions to the $|f\rangle$. While performing on-resonance $|g\rangle \leftrightarrow |e\rangle$ gate operations at frequency f_{eg}, virtual transitions to $|f\rangle$ create a phase change in $|e\rangle$.

We plot the probability of measuring the $|e\rangle$ state versus rotation axis ϕ of the final $\phi_{\pi/2}$ pulse, for $I'^{n}_{\pi/2}$ ($n = 0, 1, 3, 5$) pseudo-identity operations. Each datapoint represents 1200 repetitions of the experiment. Fits to extract the phase shift are plotted as lines. Consistent with Eq. (4.11), the phase error scales with n as shown in Figure 4.2. For $n = 5$ the final pulse is 83° out of phase, corresponding to a 10× phase error amplification from a total of 11 pulses (10 from the APE sequence and 1 from the initial $X_{\pi/2}$), yielding 7.3° phase error per gate. The oscillation amplitude is also reduced, due to decoherence.

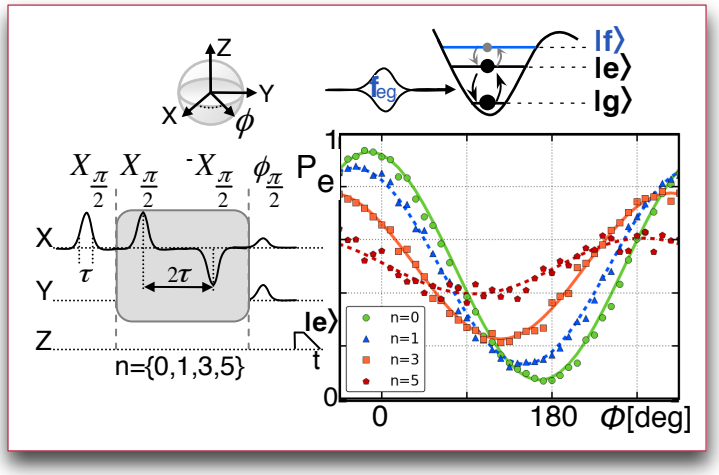

Figure 4.1: APE for X-control Gaussian pulses. (top) Bloch sphere indicating final axis of rotation. Multilevel qubit driven on resonance. (Left) APE pulse sequence. (Right) Probability of measuring the $|e\rangle$ state P_e versus final $\phi_{pi/2}$-pulse for n= $\{0, 1, 3, 5\}$ pseudo-identity operations.

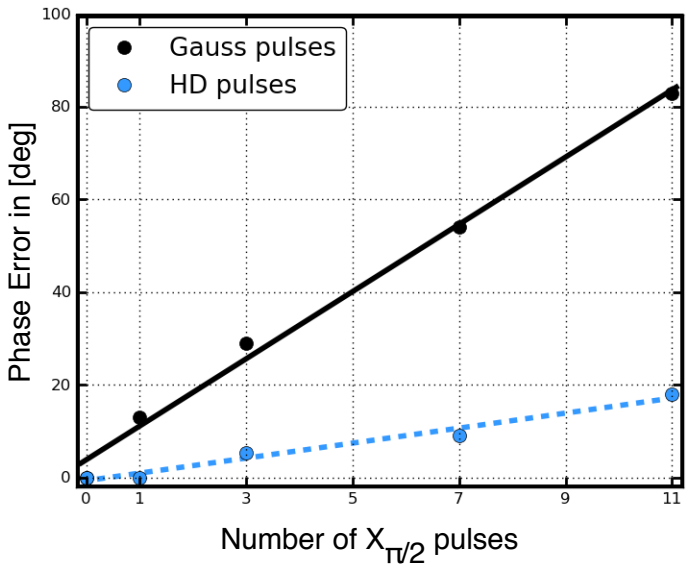

Figure 4.2: Phase error for sequential applications of $X_{\pi/2}$ pulses for Gaussian (black) and HD (blue) pulses.

4.1.3 Correcting Phase Error

To correct the phase error, we employ the derivative reduction by adiabatic gates (DRAG) protocol [44]. The original DRAG prescription uses three controls, X, Y, and Z. The X control provides the original envelope-shaping to the microwaves, which we implemented as a Gaussian in time with arbitrary amplitude A, $X = A\ \exp[-4\ln(2)(t-t_0)^2/\tau^2]$, where τ is the full-width-at-half-maximum (FWHM) and t_0 the time at the center of the pulse. The quadrature control $Y = -\dot{X}/\Delta$ is the time derivative of the X control scaled by the nonlinearity Δ. The Z control produces a dynamic detuning pulse during the gate that removes the effective z-rotations from the virtual transitions.

Half-Derivative

We find both in simulations and experiment that the Y and Z controls are not independent. From our numerical simulations[3], we plot in Figure 4.3 the gate fidelity defined as $F = \mathrm{Tr}(\chi_{\mathrm{sim}}\chi_{\mathrm{ideal}})$ in a colorscale for a range of magnitudes for the Y and Z controls. All simulated pulses are of the DRAG prescription and a fixed length of 6 ns FWHM. The circle in Fig. 4.3 indicates the values from the original DRAG prescription[44]. We find there is a ridge of maximum fidelity for the two control parameters, with peak values of fidelity having a simple linear

[3]Three level system with $\Delta/(2\pi) = -200$ MHz.

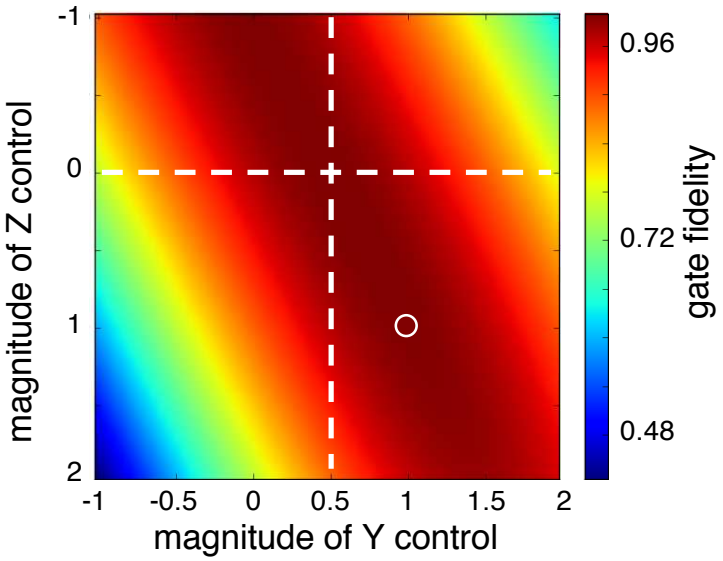

Figure 4.3: Numerical simulations of gate fidelity.

relation between the Y and Z values. Along this ridge, the maximum fidelity is insensitive to Z. Therefore, we choose to set the Z control to zero, which simplifies the experimental control procedures as it reduces the necessary control signals for optimal pulses from 3 to 2. By setting $Z = 0$ the Y control is reduced by 1/2, to give $Y = -\dot{X}/(2\Delta)$ forming the so-called "half-derivative" (HD) protocol (highlighted in Figure 4.3 with white dotted lines).

For a Gaussian envelope on the X control, the HD pulses are as illustrated in Figure 4.4 and differ from the DRAG pulses by the quadrature controls, $Y = -\dot{X}/(2\Delta)$, $Z = 0$. The Y control provides a dynamic detuning to the qubit, which keeps the microwave drive and the qubit on resonance during the gate operation

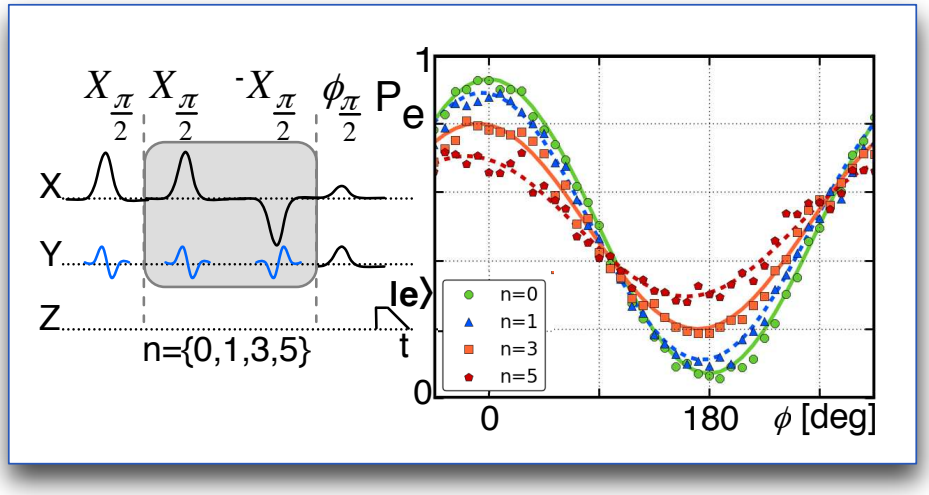

Figure 4.4: APE metrology for Half-Derivative X- and Y-control pulses.

performed by the X control, similar to the role the Z control plays in the original DRAG prescription[44]. The HD pulse sequence in Figure 4.4 is the same as Figure 4.1, but with the addition of the Y control. Data (dots) and fits (lines) are plotted for the same number of $I'_{\pi/2}$ pseudo-identity operations. We find by simply using the analytic expression for HD, $Y = -\dot{X}/(2\Delta)$ the phase error is reduced to 1.6° per gate. One can tune the phase error to zero by utilizing the APE experiment to adjust the magnitude of the Y control.

4.2 Amplitude Error: The Redux

HD pulses also reduce the amplitude errors, i.e. leakage to the $|f\rangle$ state. As shown in Figure 4.5 we plot the data from a Ramsey error filter as done in Chapter 3 for

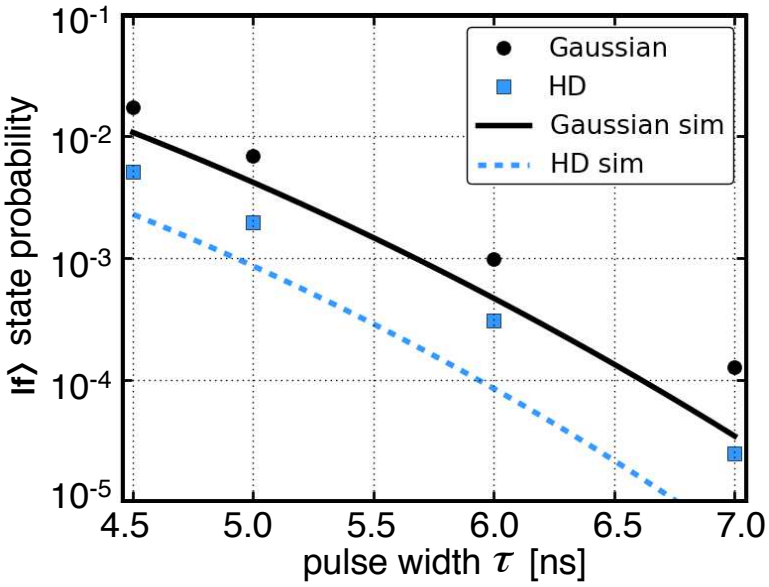

Figure 4.5: Amplitude errors due to leakage into the $|f\rangle$ state from an X_π-pulse.

both single control Gaussian (black dots) and HD pulses (blue squares). The lines are three-state simulations using Gaussian (solid black) and HD (dashed blue) pulses. A 6 ns (FWHM) HD X_π pulse gives a $|f\rangle$ state probability of 10^{-4}, almost an order of magnitude better than a non-HD pulse of the same width, which consequently provides a 20% faster gate with equivalent performance to what was shown in Chapter 3.

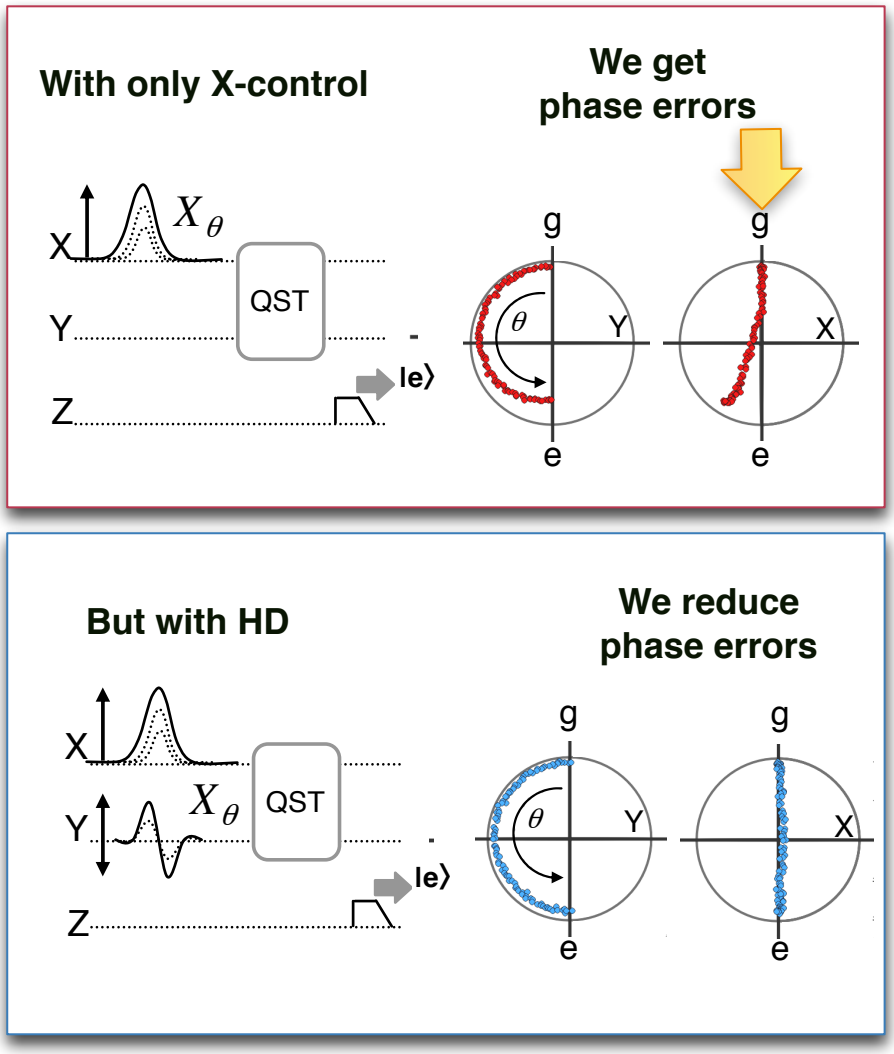

Figure 4.6: QST showing the trajectory of an X_π-pulse without HD control (top) and with (bottom).

4.3 Demonstrating Control

With calibrated $X_{\pi/2}$ and $Y_{\pi/2}$ pulses, we can now perform quantum state tomography (QST) without worry of miscalibrated measurement axes. As a practical demonstration of how HD pulses reduce phase error, we perform QST [60] with and without HD. Figure 4.6 shows the pulse sequence and data for the Gaussian pulses (HD pulses) during an X_θ rotation. The pulses are of fixed length (FWHM = 6 ns) with variable amplitude θ. QST is performed at each incremental increase of amplitude and the quantum state is recreated in the Bloch sphere from two perspectives, looking down the x and the $-y$ axes as shown to the right of each of the respective pulse sequences. In contrast with the single control Gaussian pulses, the HD pulses execute a meridian trajectory with no phase error with increasing θ.

4.3.1 Z-pulse Calibration: For Three Axis Control

For our final HD control demonstration, we calibrate our Z pulse as shown in Figure 4.7. The Ramsey-type pulse sequence consists of a static length (full-width at half-max = 6ns) with an increasing amplitude Z-pulse inserted between two HD $\pi/2$ pulses with fixed separation time $t_{\text{fixed}} = 24$ ns. The separation time is chosen to minimize overlap of the pulses. The Z_{amp} increases incrementally. We plot the probability of measuring the $|e\rangle$ state P_1 as a function of Z-pulse amplitude,

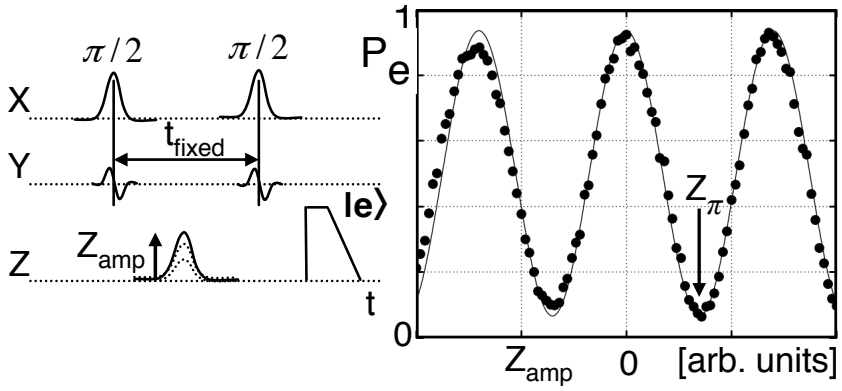

Figure 4.7: Z-pulse calibration.

$Z_{\rm amp}$. The data are plotted as points with best fit as a line. The probability of measuring the $|e\rangle$ state P_1 oscillates with increasing $Z_{\rm amp}$[60]. The arrow indicates the $Z_{\rm amp}$ that corresponds to a rotation angle of π about the z-axis.

The final demonstration of our single qubit control using the HD protocol is an (off-equator) Hadamard gate, shown in Fig. 4.8, which uses all three $x-$,$y-$,and $z-$ control axes. We incrementally increase the amplitude of all three control lines using fixed length (FWHM = 6 ns) pulses to perform rotations from 0 to $\pi/\sqrt{2}$ about both the x and z axes, which at full amplitude gives the Hadamard gate H ($|g\rangle \rightarrow (|g\rangle + |e\rangle)/\sqrt{2}$). The trajectory concludes with a second set of pulses to complete the identity operation $I = HH$, and returning to the initial state $(|g\rangle + |e\rangle)/\sqrt{2} \rightarrow |g\rangle$.

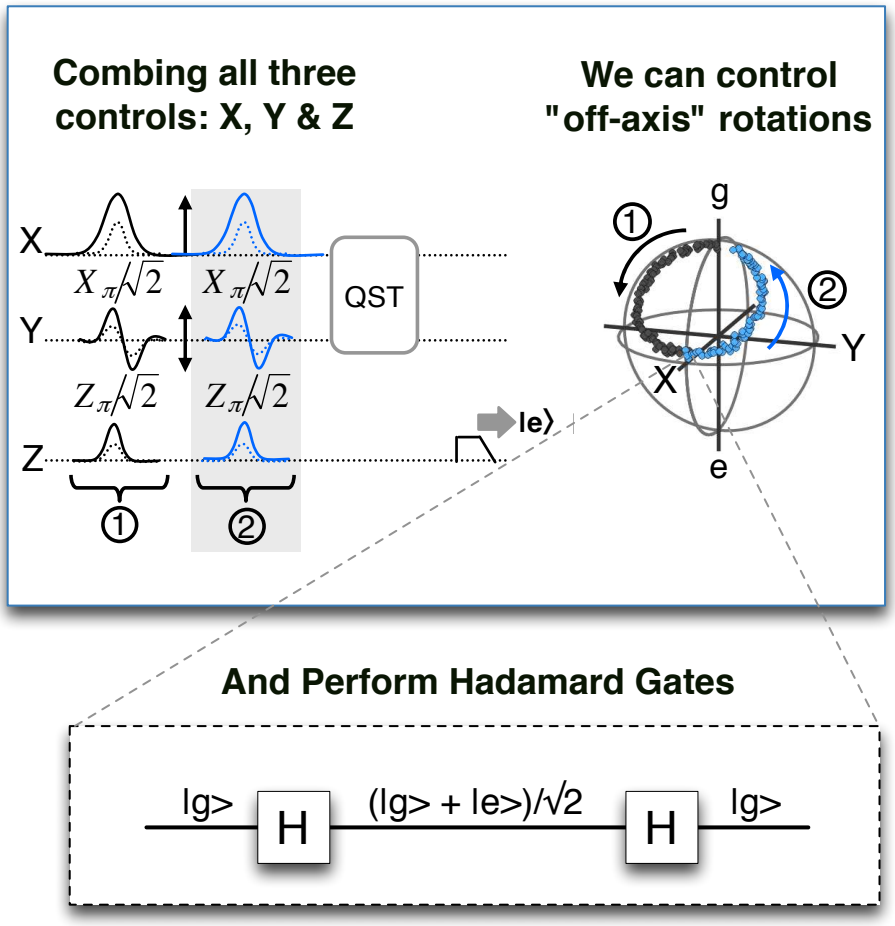

Figure 4.8: Demonstrating qubit control. Hadamard trajectory reconstructed from QST.

Chapter 5

$15 = 3 \times 5$, Some of The Time

In this chapter, we pull everything together, including the concepts covered in the introductory chapter, the design characteristics of the QuP discussed in Chapter 2, and the qubit control details covered in Chapters 3 and 4, to demonstrate the capabilities of our Josephson phase-qubit quantum processor (QuP) as shown in Figure 5.1. The chapter begins with a description of the QuP and its capabilities. In Section §5.3, I show swap spectroscopy [39], experimentally verifying the existence of all nine of the engineered quantum elements. In §5.4, I show the fast entangling operations to create Bell and $|\mathrm{W}\rangle$ states, and simultaneous coherent interactions of the four phase qubits with the bus resonator. In §5.5, I introduce the quantum circuit for the compiled version of Shor's algorithm, folllowed by the quantum runtime analysis of the algorithm in §5.6. I conclude with the results

from the Shor algorithm, and finish the computation to find the prime factors p and q of $N = 15$.

5.1 The QuP

The QuP pictured and schematically illustrated in Figure 5.1 was scaled-up to nine quantum elements from an architecture of two qubits and three resonators [39] (like the device pictured in Figure 1.2e). The QuP was fabricated with aluminum (colored regions in the photomicrograph) on a sapphire substrate (black regions) using Al/AlO_x/Al Josephson junctions.

The bottom panel of Figure 5.1 shows a complete schematic of the device. As described in Chapter 2, the device is designed with four phase qubits and five superconducting coplanar waveguide (CPW) resonators. Each qubit Q_i is individually controlled using a bias coil that carries dc, rf- and GHz-pulses to adjust the qubit frequency and to pulse microwaves for manipulating and measuring the qubit state. The GHz microwave pulses produce single qubit operations, capable of performing HD pulses as described in Chapter 4. The rf-pulses provide the control to adjust each qubit's frequency over an operating range of $\sim 2\,\mathrm{GHz}$, allowing each qubit to couple to the other quantum elements on the chip. Each qubit Q_i is connected to a $\lambda/4$ memory resonator M_i, as well as the central $\lambda/2$ bus resonator B, via interdigitated capacitors. Each Q_i is inductively coupled to

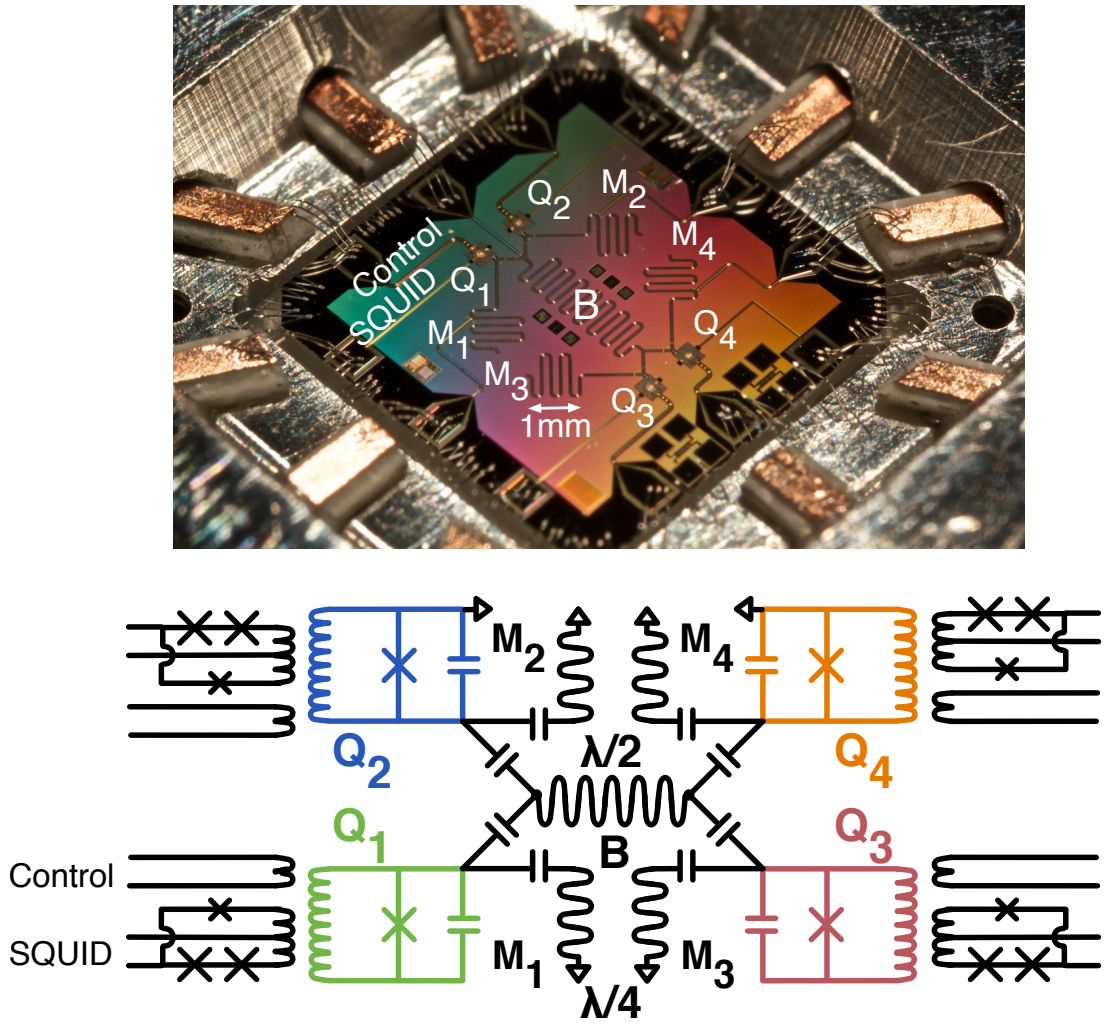

Figure 5.1: Micrograph (top) of the Josephson quantum processor and full schematic (bottom).

a superconducting quantum interference device (SQUID) for single-shot readout.

5.2 Device Description and Capabilities

To help illustrate the various operations available on the QuP, consider the "ball-and-stick" operation models in the following sections.

5.2.1 IDLE Bias

As depicted in Figure 5.2, the phase qubits can be tuned in frequency (colored ball moving up and down on the stick) to couple to other quantum elements that are static in frequency e.g. the bus and memory resonators (rectangles labeled B, $\mathrm{M}_1, \ldots, M_4$). The so called "IDLE Bias" is where the qubits ($Q_1 \ldots Q_4$) start at the beginning of an experiment and ultimately return to just prior to measurement. This state is off resonance from both the bus and memory resonators so as not to interact with them. Although the qubits are drawn as having unique IDLE biases, they can be operated at the same qubit frequency $\omega_{eg}/2\pi = f_{eg}$ provided that they are all off-resonance with the bus (memory) resonator to reduce the qubit-resonator coupling interaction by a factor of ~ 100. For a bus resonator at $f_B = 6.1\,\mathrm{GHz}$ (memory resonator at $f_{M_1} = 6.8\,\mathrm{GHz}$), and a qubit

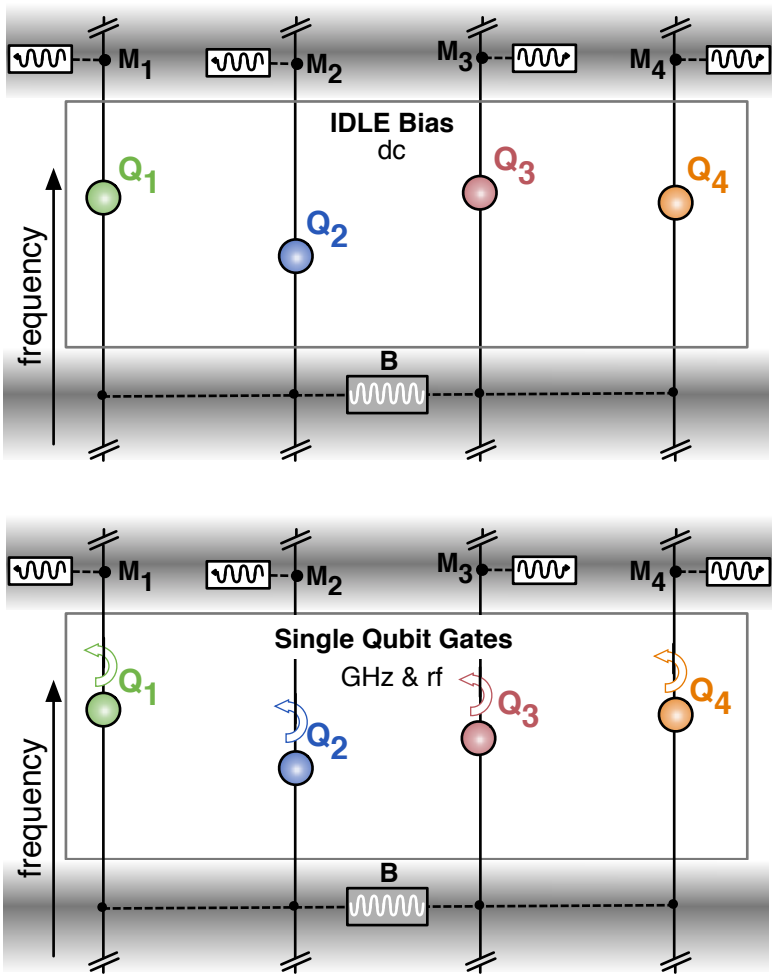

Figure 5.2: Ball-and-stick model of the Josephson quantum processor. IDLE state and performing single qubit gates.

IDLE bias frequency[1] of $f_{eg} = 6.6\,\text{GHz}$, the qubit-bus resonator coupling strength of $g = 55\,\text{MHz}$ (qubit-memory resonator $g_{M_1} = 20\,\text{MHz}$) is reduced by a factor of $\frac{(f_B - f_{eg})^2}{g^2} = 83$, $\left(\frac{(f_{M_1} - f_{eg})^2}{g_{M_1}^2} = 100\right)$.

Due to the reduced effective coupling between the qubits and resonators, the IDLE bias is where single qubit gates are performed, as illustrated in the bottom panel of Figure 5.2. The outlined arrows indicate single qubit rotations by applying GHz and (small-amplitude) rf-pulses to the respective qubit control lines, as discussed in Chapters 3 and 4.

5.2.2 Memory and Coupling Operations

As shown in the QuP circuit schematic in Figure 5.1 and recreated in the "ball-and-stick" operation model in Figure 5.3, each qubit Q_i is capacitively connected to the bus B (and respective memory M_i) resonator. The capacitive coupling is drawn as dotted lines between the quantum elements in Figure 5.3. Although the coupling capacitors are fixed, Figure 5.3 illustrates how the effective interaction can be controlled by tuning the qubits into or near resonance with the coupling bus to turn the coupling "on", or detuning Q_i to $f_B \pm 500\,\text{MHz}$ to turn the coupling "off" [23]. This tuning/detuning is controlled via fast rf-pulses (represented as dotted arrows in Figure 5.3) on the qubit control lines, with pulse rise times

[1] In practice this IDLE bias varies from day to day, but typically only on the order of a few MHz, which is automatically adjusted via the daily calibrations as discussed in Chapter A.

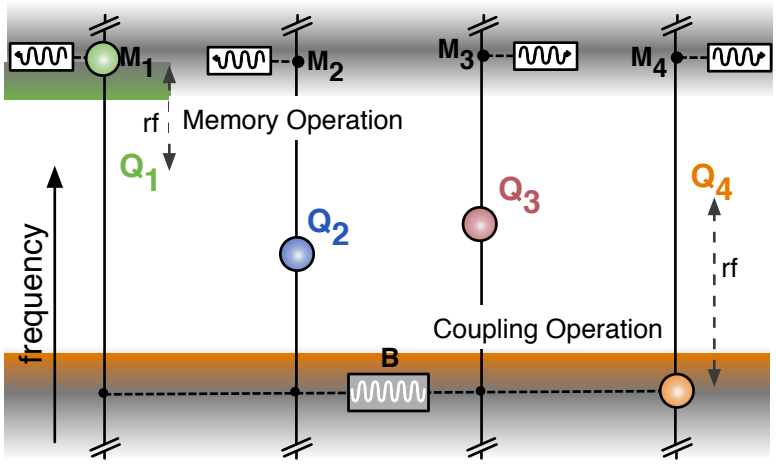

Figure 5.3: Ball-and-stick model of the Josephson quantum processor. Memory and entangling operations via rf-pulses.

$\sim$ 1 ns and pulse durations $O(10\,\text{ns})$. By applying rf-pulses on the respective control line, each qubit is tuned in and out of resonance with B (M) to perform entangling (memory) operations. In the case illustrated in Figure 5.3 an rf-pulse is applied to the control line of Q_4 (Q_1) to tune it into resonance with the bus resonator B (memory resonator M_1) for a coupling (memory) operation.

5.2.3 Simultaneous Measurement

The rf-pulses are also used for measuring the qubits. Because each phase qubit is separately coupled to its own readout SQUID we can perform simultaneous measurement of all four qubits. The capability for simultaneous measurement is an important distinction between our previous capacitively coupled devices which suffered from measurement crosstalk [3, 48]. The rf-measurement-pulses are illus-

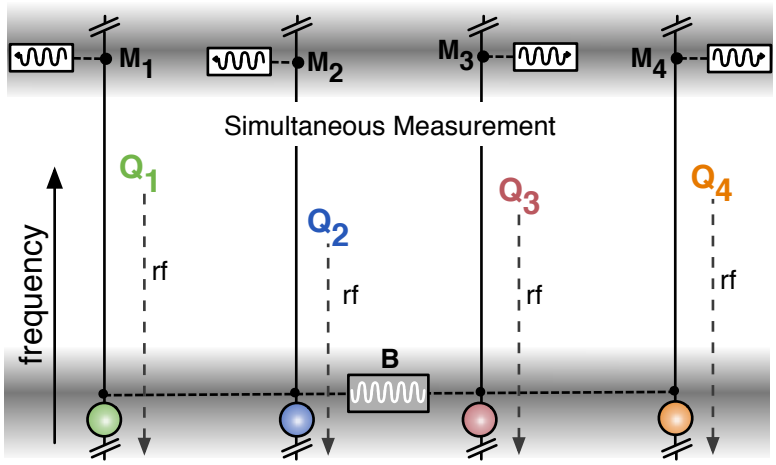

Figure 5.4: Ball-and-stick model of the Josephson quantum processor. Simultaneous measurement via rf-pulses.

trated in Figure 5.4 as dotted arrows next to each qubit indicating the application of the appropriate rf-pulse to measure the qubit as discussed in Chapter 2.

5.2.4 High-Level QuP Operations

Creating entanglement and executing quantum algorithms [49, 7] constitute high-level QuP operations built upon lower-level single and coupled qubit operations. The QuP runs quantum algorithms by a sequence of high-fidelity single-qubit gates (X, Y, Z, and H), [35, 36] (controlled by applying GHz- and rf-pulses to the respective qubit control lines at the qubit "IDLE Bias" frequency) combined with controlled-phase (C_ϕ) gates [18, 69, 39], which are composed of single qubit gates and qubit-resonator coupling operations (controlled by applying rf-pulses).

The QuP can also create entanglement by utilizing "fast-entangling logic". Fast-entanglement is realized by applying rf-pulses to the respective qubits to bring all of the participating qubits on resonance with the bus resonator at the same time [63].

5.3 Experimentally Verifying The QuP

The QuP is mounted and wirebonded into a superconducting aluminum sample holder[2] and cooled in a dilution refrigerator to $\sim 25\,\mathrm{mK}$, as outlined in §2.6. The individual qubit operation and calibrations are similar to previous works[24, 4, 47, 69, 39], with additional automated calibrations detailed in Appendix A. At this point, we have verified that each qubit can perform high fidelity single qubit gates and to that end, we are ready to characterize the remaining quantum elements and use the QuP.

5.3.1 Swap Spectroscopy: Phase Qubit as a Spectrum Analyzer

With the individual phase qubits tuned up at the nominal "IDLE Bias" we perform swap spectroscopy[39] to calibrate all nine of the engineered quantum elements on the QuP. The protocol for swap spectroscopy illustrated in the bottom panel

[2]This is a microwave engineered cavity, whose details are discussed in [48, Chapter 2]

of Figure 5.5, is as following: First we prepare the qubit in the excited state $|\mathrm{e}\rangle$, by applying a π-pulse at the IDLE Bias, we then detune the qubit away from its IDLE bias point via a fast rf-pulse and allow the quanta of energy to swap with whatever modes exists in the spectrum. Thereby demonstrating that swap spectroscopy uses the qubit as a quantum-limited spectrum-analyzer.

The top panel of Figure 5.5 shows the probability of the qubit in the exited state, P_{e} (color scale) versus frequency (vertical axis) and interaction time $\Delta\tau$ for each qubit $Q_1 \ldots Q_4$. For each individual swap spectroscopy experiment only the respective qubit is pulsed into the excited $|e\rangle$ state, all other qubits are operated at the IDLE bias so as not to participate. As a representative example, let us focus on the data for the green qubit Q_1. Initially, Q_1 is pulsed into the excited $|e\rangle$ state at its IDLE bias (dark blue on the P_e colorscale) and then detuned ($f = \Delta f \pm f_{eg}$) from the IDLE bias. At 6.1 GHz, Q_1 is on resonance with the bus resonator B. As the qubit and resonator interact for duration $\Delta\tau$, the excitation originally in Q_1 is swapped to the bus resonator B (red data). As the interaction continues, the excitation is transferred back-and-forth (red-to-blue-to-red ...) between the qubit and the bus resonator resulting in the chevron patterns centered about $f = 6.1$ GHz for the bus resonator B and $f = 6.8$ GHz for the memory resonator M_1 ($f = 7.2, 7.1, 6.9$ GHz for the memory resonators $\mathrm{M}_2 - \mathrm{M}_4$ respectively). The oscillation periods of the chevrons give the coupling strengths between $\mathrm{Q_i}$ and

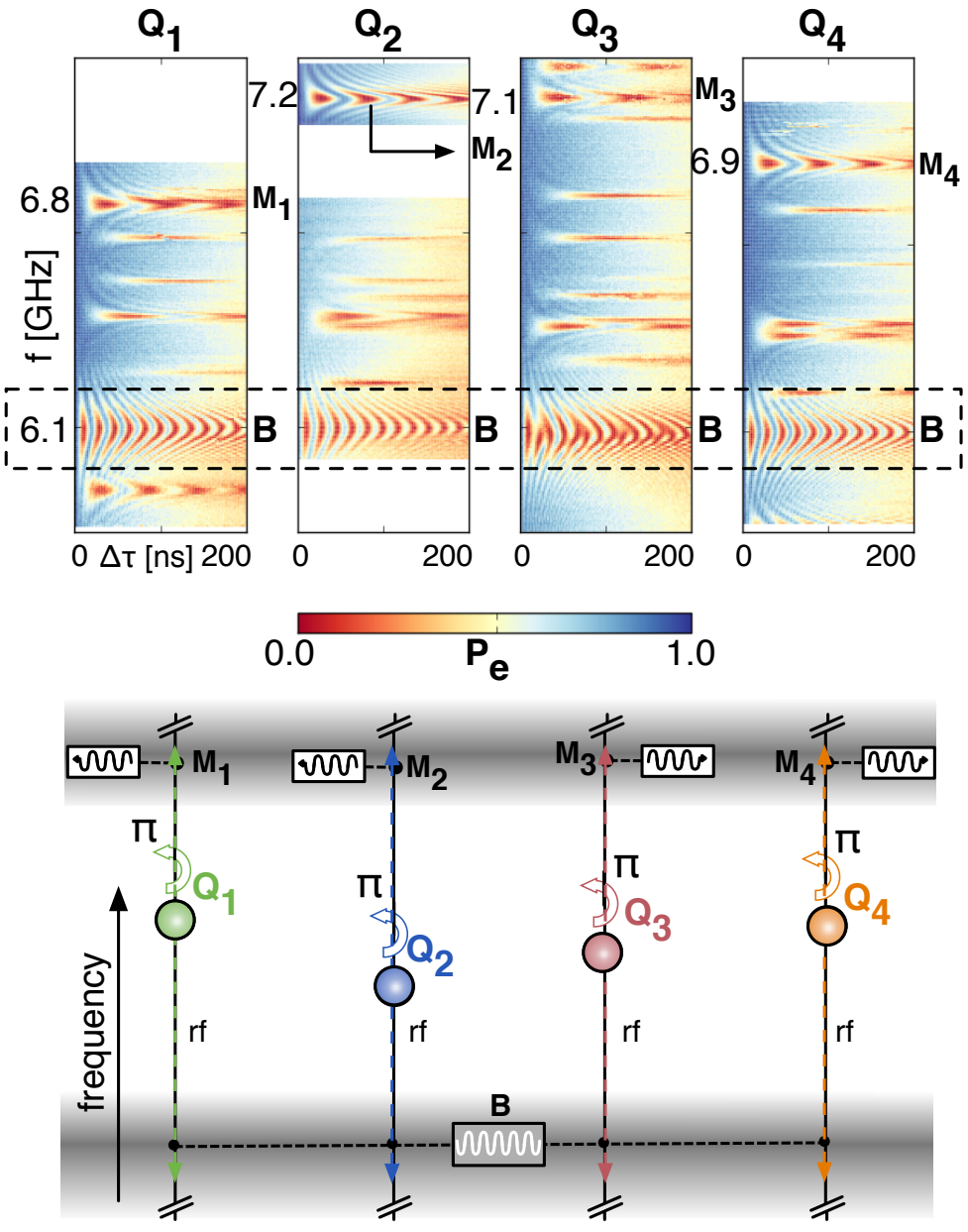

Figure 5.5: Swap spectroscopy.

B (M_i), which for the four qubits are all $\cong$ 55 MHz ($\cong$ 20 MHz). The coupling strengths between Q_i and B (M_i) were measured to be within 5 % (10 %) of the design values. Swap spectroscopy is repeated for each qubit to map out all of the intentionally engineered modes (B and M_i) and the unintentional TLS defects. With a complete qubit spectrum and all of the modes accounted for an optimal IDLE bias is chosen for each qubit to minimize the stray coupling to the various modes in the spectrum.

5.4 Fast Entangling Logic

With all of the quantum elements accounted for we can move on to more interesting demonstrations of the QuP. For the fast entangling logic demonstrations we will be using all four phase qubits and the bus resonator. The dynamics of the qubit-resonator interactions can be described by the Jaynes-Cummings model Hamiltonian[27]

$$H_{\text{interaction}} = \sum_i \frac{\hbar g_i}{2}(a^\dagger \sigma_i^- + a\sigma_i^+), \tag{5.1}$$

where g_i is the coupling strength between the bus resonator B and the qubit Q_i, $a^\dagger$ and a are respectively the photon creation and annihilation operators for the resonator, σ_i^+ and σ_i^- are respectively the qubit Q_i raising and lowering operators, and $\hbar = h/2\pi$. This Hamiltonian $H_{interaction}$ describes the swapping of excitations

between two (or more) modes when on resonance.

At the beginning of the fast entangling operations the qubits $Q_1 - Q_4$ are initialized in the ground state $|gggg\rangle$ and tuned off-resonance from the bus resonator B at an idle frequency $f \sim 6.6\,\mathrm{GHz}$. Qubit Q_1 is prepared in the excited state $|e\rangle$ via a π-pulse. The bus resonator B is then pumped into the first Fock state $n = 1$ by tuning Q_1 on resonance ($f \sim 6.1\,\mathrm{GHz}$) via a fast rf-pulse of duration $1/(2g_1) = \tau \sim 9\,\mathrm{ns}$, calibrated for an iSWAP operation between B and Q_1, $|0\rangle \otimes |eggg\rangle \rightarrow |1\rangle \otimes |gggg\rangle$ [24] as illustrated in the top panel ball-and-stick model in Figure 5.6.

The participating qubits are then tuned on resonance ($f \sim 6.1\,\mathrm{GHz}$) and left to interact with B for an interaction time $\Delta\tau$ as illustrated in the bottom-panel of Figure 5.6. The dynamics during the interaction between the $i = \{1, 2, 3, 4\}$ qubits and the bus resonator are shown in the top panel of Figure 5.5 for $N = 1$, and Figure 5.7 (left panels labeled) for $N = 2$, $N = 3$, $N = 4$.

The three panels on the left of Figure 5.7 show the probability P_{Q_i} of measuring the participating qubits in the excited state, and the probability P_B of B being in the $n = 1$ Fock state, versus $\Delta\tau$. At the beginning of the interaction the excitation is initially concentrated in B (P_B maximum) then spreads between the participating qubits (P_B minimum) and returns back to B, continuing as a coherent oscillation during the interaction time $\Delta\tau$. As shown in the $N = 2$ panel,

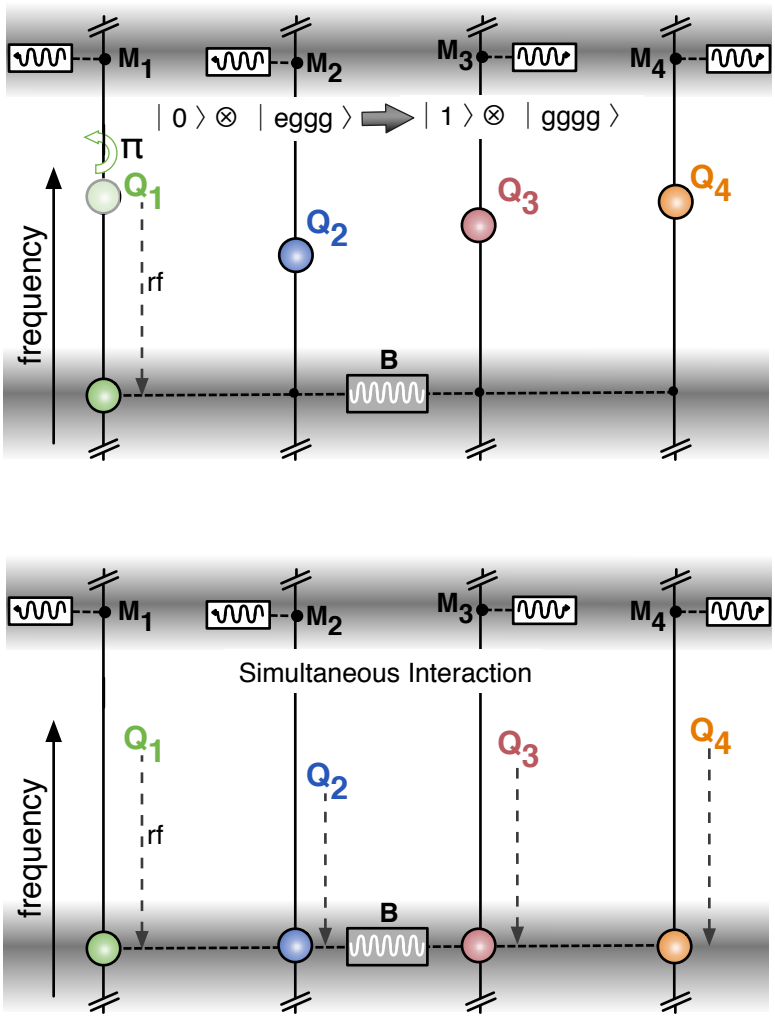

Figure 5.6: Ball and stick model for fast entangling operation.

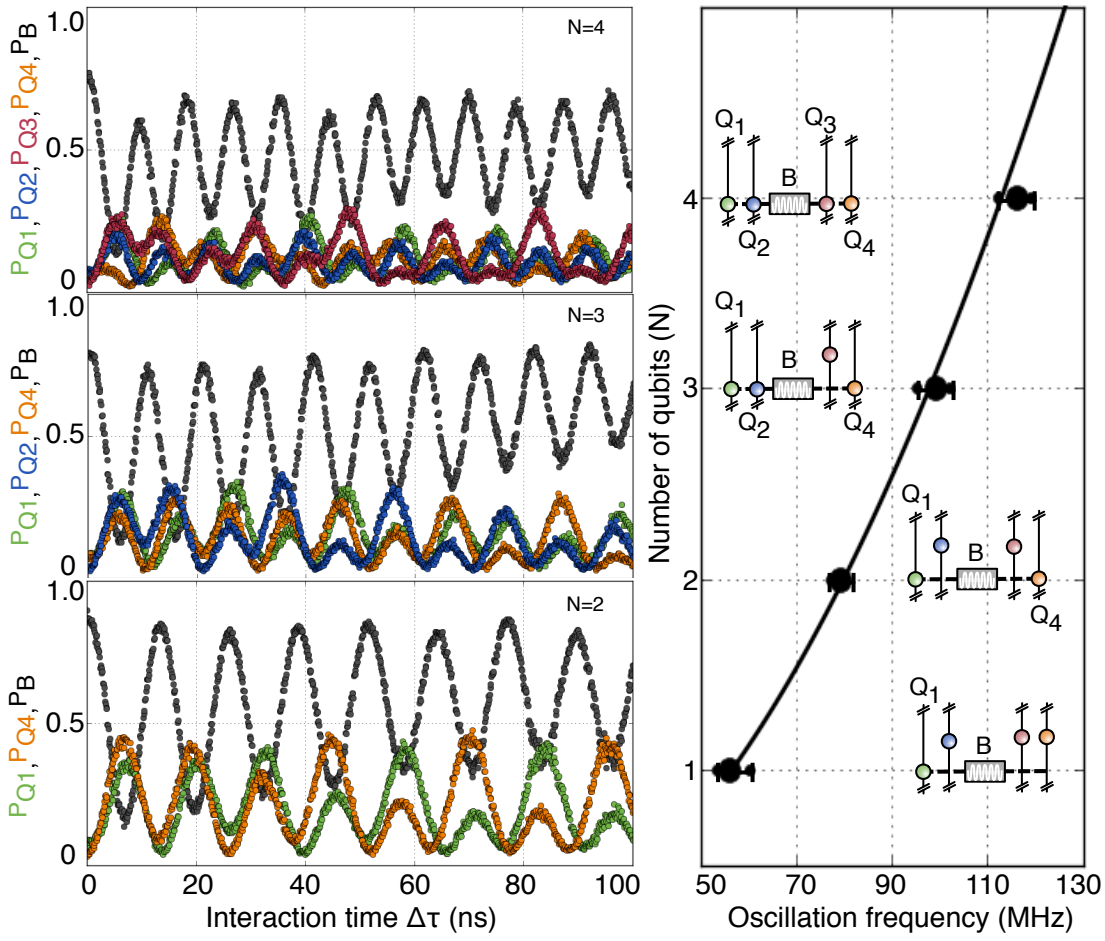

Figure 5.7: Coherent Oscillations for increasing number of qubits interacting with the bus resonator, with details explained in the text.

the bus resonator (black data) and the qubits Q_1 and Q_4 (green and orange data) coherently share the single excitation. The deviation from an equivalent sharing between the participating qubits results in a beat frequency, which is apparent after 40 ns of interaction. This is due to the difference in coupling strengths of the qubits and can be compensated for by adjusting the detuning of the participating qubits (not shown).

As the number of participating qubits increase to $N = 3$ and $N = 4$ the period of the coherent oscillation increases as shown in the left panels labeled $N = 3$ and $N = 4$ in Figure 5.7. With more qubits interacting with the bus resonator the time to swap an excitation back-and-forth is reduced. For the $N = 2$ qubits interacting with the bus resonator a single swap is (from B to Q_1 and Q_4) takes 6.5 ns, while for $N = 3$ ($N = 4$) qubits it takes 5.1 ns (4.5 ns).

5.4.1 Enhanced Coupling Strength with The Number of Qubits Interacting with The Bus Resonator

When the qubits are simultaneously tuned on resonance with B they interact with an effective coupling strength $\bar{g}_N$ that scales with the number N of qubits as $\sqrt{N}$[20], analogous to a single qubit coupled to a resonator in a n-photon Fock state[24]. These coherent oscillations continue for a time $\Delta\tau$ and increase in frequency with each additional qubit. For N qubits, $\bar{g}_N = \sqrt{N}\bar{g}$, where $\bar{g} =$

$[1/N(\sum_{i=1,N} g_i^2)]^{1/2}$. The oscillation frequency of P_B for each of the four cases $i = \{1, 2, 3, 4\}$ is shown in the right panel of Figure 5.7. The inset schematics illustrate which qubits participate. These results are similar to Ref.[20], but with a larger number N of qubits interacting with the resonator, we can confirm the $\sqrt{N}$ scaling of the coupling strength with N. From these data we find a mean value of $\bar{g} = 56.5 \pm 0.05\,\mathrm{MHz}$. The error bars on the data in the right panel of Figure 5.7 indicate the $-3\,\mathrm{dB}$ point of the Fourier transformed P_B data.

5.4.2 Rapid Entanglement: Bell and W-States

By tuning the qubits on resonance for a specific interaction time τ, corresponding to the first minimum of P_B in Figure 5.7 (for $N = 2$ and $N = 3$) we can generate Bell singlets $|\psi_S\rangle = (|ge\rangle - |eg\rangle)/\sqrt{2}$ and $|\mathrm{W}\rangle$ states $|\mathrm{W}\rangle = (|gge\rangle + |geg\rangle + |egg\rangle)/\sqrt{3}$. Stopping the interaction at this time ($\tau_{Bell} = 6.5\,\mathrm{ns}$ and $\tau_W = 5.1\,\mathrm{ns}$) leaves the single excitation evenly distributed among the participating qubits and places the qubits in the desired equal superposition state similar to the protocol in Ref.[2], but with the full quantum state tomography (QST) we are able to further analyze these states.

Figure 5.8 show the real part of reconstructed density matrices from this analysis[60]. The Bell singlet $|\psi_s\rangle = (|ge\rangle - |eg\rangle)/\sqrt{2}$ is formed with fidelity $F_{Bell} = \langle\psi_s|\,\rho_{Bell}\,|\psi_s\rangle = 0.89 \pm 0.01$ and entanglement of formation[22] EOF =

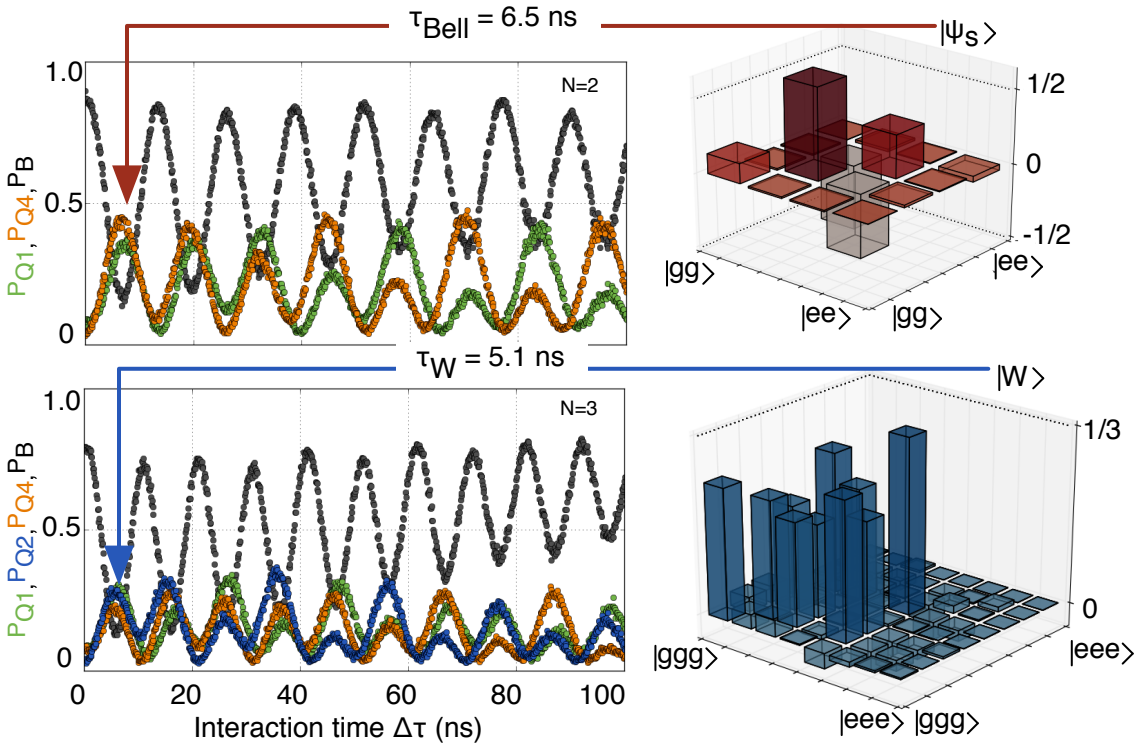

Figure 5.8: Reconstructed density matrices for Bell-state creation and three qubit W-state.

0.70. The three-qubit state $|\mathrm{W}\rangle = (|gge\rangle + |geg\rangle + |egg\rangle)/\sqrt{3}$ is formed with fidelity $F_W = \langle W| \rho_W |\mathrm{W}\rangle = 0.69 \pm 0.01$, which satisfies the entanglement witness inequality $F_W > 2/3$ for three-qubit entanglement [1]. The measured imaginary parts, which are not displayed are found to be small, with $|\mathrm{Im}\,\rho_{\psi_\mathrm{s}}| < 0.05$ and $|\mathrm{Im}\,\rho_\mathrm{W}| < 0.06$, as expected theoretically.

Generating either of these classes of entangled states (bi- and tri-partite) requires only a single entangling operation that is short relative to the characteristic time for two-qubit gates ($t_g \sim 50\,\mathrm{ns}$). This entanglement protocol has the further advantage that it can be scaled to an arbitrary number of qubits by connecting more qubits to the resonator and tuning them on resonance for the appropriate interaction time.

5.5 Compiled Version of Shor's Algorithm

The initial motivation for creating this QuP was to perform a compiled version of Shor's algorithm as proposed in [9], which was used in formulating the quantum circuit for the pioneering Shor algorithm demonstration in nuclear magnetic resonance (NMR)[64], and more recently in photonic systems[31, 33, 54]. A reformulated proposal for electrons in semiconductor nanostructures [14] also discusses the details of the quantum circuit compilation. Here, we use these previous proposals and demonstrations and map the compiled version of Shor's algorithm to

our superconducitng QuP. This makes for an interestingt demonstration of our QuP's capabilities as it combines the challenge of precise and accurate individual qubit control (as discussed in Chapter 3 and Chapter 4), with entangling operations (discussed here) to form a sequence of quantum operations that perform a meaningful quantum algorithm.

Of particular importance to the success of this experimental demonstration (and also for the success of the next generation of quantum algorithms) were the automated calibrations, which we leave for discussion in Chapter A. For now we mention that the full factoring sequence that we describe was executed after performing automatic calibration of the individual gates. We then combined them, without additional tuning, so as to factor the composite number N = 15 with co-prime a = 4, (where $1 < a < N$ and the greatest common divisor between a and N is 1).

5.5.1 Four Qubit Quantum Circuit

The quantum circuit for a compiled version of Shor's algorithm is shown in Figure 5.9 for factoring the number $N = 15$ with $a = 4$ co-prime [9, 14], which returns the period $r = 2$ ("10" in binary) with a theoretical success rate of 50 %. The three steps in the quantum algorithm are initialization, modular exponentiation,

and the quantum Fourier transform[3]. Once we have r from this routine, we can use a classical computer to calculate the prime factors, p and q (as demonstrated in Chapter 1).

In Figure 5.9, computation moves from left to right and the participating qubits labeled on the left Q_1,Q_2,Q_3, and Q_4 each have a line that represents their progression through the algorithm. Single qubit operations are represented as boxes, although in this algorithm only H-gates are used, one would represent X, Y or Z rotations of any arbitrary angle (though typically the angles are some fractions of π) with a box around the letter and a subscript for the angle. The H-gate is performed like what we saw in Chapter 4. Entangling operations between qubits are represented with a black dot for the control qubit(s), which is connected to a circle with a cross in it (just like a target) for the target qubit(s)[4]. For the entangling operations used here, we employ the C_π gate, or more commonly referred to as the C_z gate as proposed by[62] and detailed in [39]. Combined with single qubit gates, i.e. Hadamard-gates, we can form the more familiar controlled NOT (CNOT) gate. The CNOT action is to flip the the target qubit if and only if the control qubit is in the excited state. The algorithm ends with a projective measurement of the qubits of interest, which is represented as a meter

[3]Although we did not need to use the quantum Fourier transform (QFT) in this demonstration, this QuP architecture can perform the QFT as demonstrated in [39].

[4]There are controlled gates with more qubits, like a Toffoli gate which requires three qubits, two controls and one target [39].

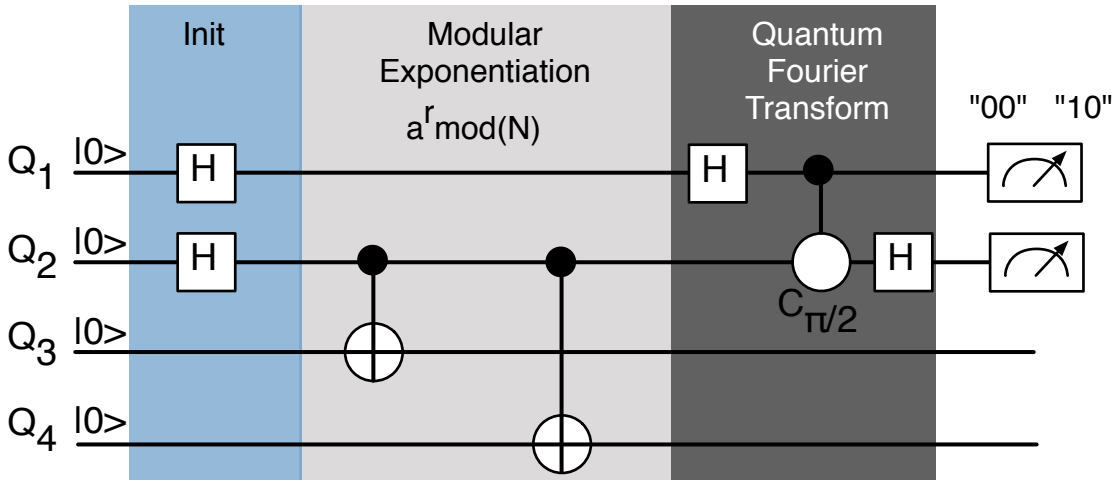

Figure 5.9: Quantum circuit of Shor's Algorithm, using four qubits to factor $N = 15$, with co-prime $a = 4$.

that returns either "0" or "1". Recall that this quantum algorithm will run for $\sim 10^5$ repetitions to build up the final probabilities.

5.5.2 Recompiling The Quantum Circuit

The algorithm can be further simplified by noticing that qubit Q_1 is initialized via a Hadamarad (H) gate and then idles until the next H-gate. This sequence is highlighted in Figure 5.10. Since the Hadamard gate is self-inverse, i.e. $H \cdot H = I$, we can replace these two gates with an Identity I gate as illustrated in Figure 5.10.

Still looking at the actions of Q_1, we notice that Q_1 is the control qubit for the controlled $C_{\pi/2}$ gate. Because Q_1 is initialized in the $|g\rangle$ and idles until the $C_{\pi/2}$ gate it therefore does not invoke the controlled action on the target qubit Q_2.

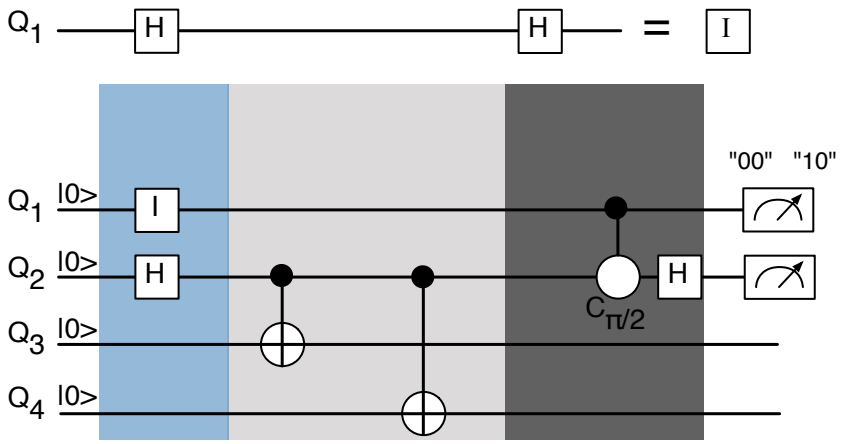

Figure 5.10: Recompiling: Hadamard, Hadamard equals Identity.

This action is highlighted in Figure 5.11. When the control qubit is $|g\rangle$, the target qubit's initial state $|\psi\rangle$ equals its final state $|\psi'\rangle$ such that $|\psi'\rangle = U|\psi\rangle \rightarrow |\psi'\rangle = I|\psi\rangle$. So, we can replace the $C_{\pi/2}$ with two I-gates as highlighted in Figure 5.11.

5.5.3 Three Qubit Quantum Circuit

The final step in the "recompiling" is to remove the redundant qubit Q_1 by noting that we always measure Q_1 in the $|g\rangle$ state. Removing Q_1 forms the three qubit version of Shor's algorithm as shown in Figure 5.12 and Figure 5.13. The quantum circuits in Figure 5.9 and Figure 5.13 are equivalent for the specific case of $N = 15$ with $a = 4$ co-prime. As discussed in Chapter 1, entanglement plays a key role in the success of a quantum algorithm therefore, we perform a quantum runtime

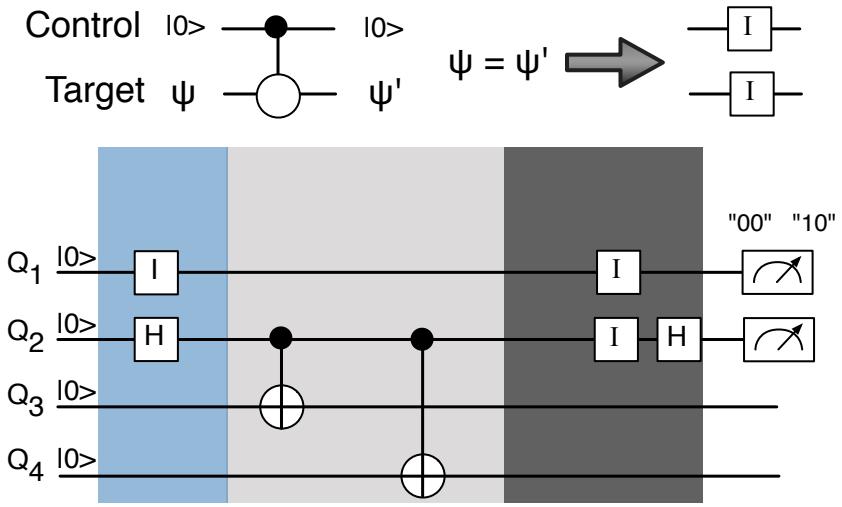

Figure 5.11: Recompiling: Controlled gate with control qubit equal zero, performs identity operation on target qubit.

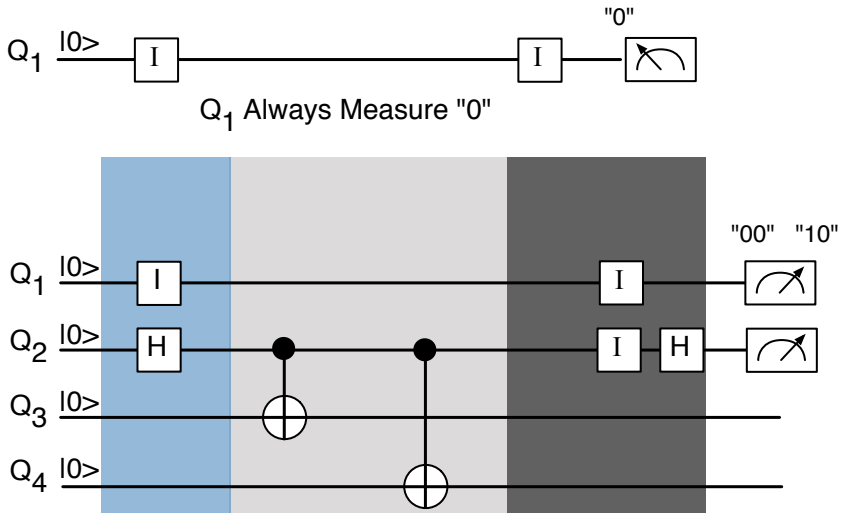

Figure 5.12: Recompiling: Q_1 is always measured in ground state, therefore it is redundant.

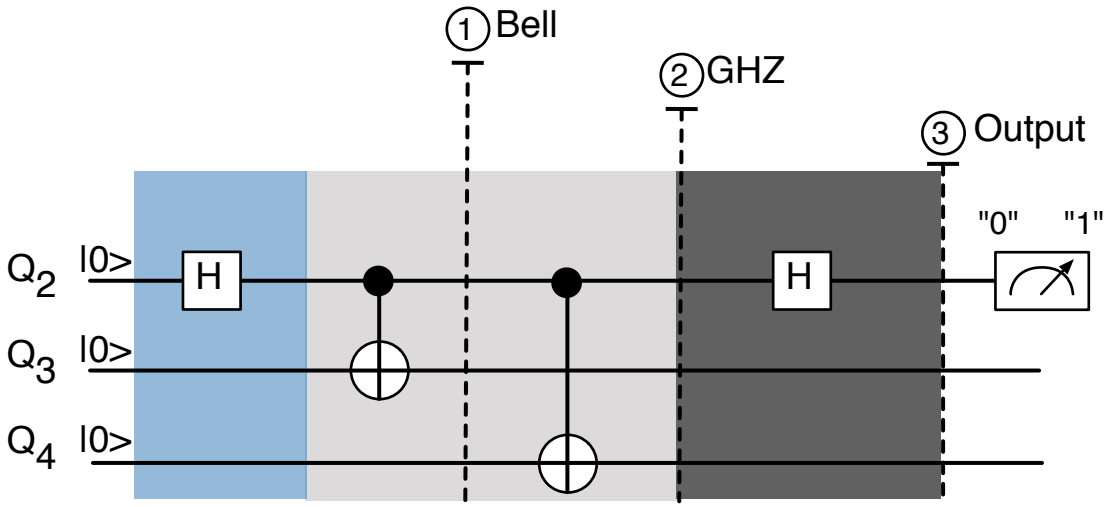

Figure 5.13: "Recompiled" quantum circuit of Shor's Algorithm, using three qubits to factor $N = 15$, with co-prime $a = 4$.

analysis to check for entanglement throughout the algorithm at the three points labeled "Bell", "GHZ" and "Output" in Figure 5.13.

5.6 Quantum Runtime Analysis

5.6.1 Step 1: Bell States via C-Phase Gate

The first breakpoint in the algorithm verifies the existence of bipartite entanglement[3]. A Bell-singlet $|\psi_s\rangle$ is formed after a H gate [36] on Q_2 and a CNOT[69, 39] between Q_2 and Q_3. Figure 5.14 shows the actual pulse sequence used. The traces are for all three control pulses (X, Y, and Z) for both qubits Q_2 and Q_3. The Hadamard gates are labeled above the X-,Y- and Z-pulses (although the Z-pulses are hard

to make out on this scale, they are indeed used). The CNOT gate is realized by sandwiching a C_z gate between two H-gates, as illustrated in the top panel of Figure 5.14 and in the actual pulse traces (captured on a high-speed oscilloscope) shown below. The gray region labeled "QST" for quantum state topography is used to reconstruct the density matrices used to analyze the quantum state.

The Z-pulses for the control qubit Q_2 do the following: The first pulse tunes Q_2 on resonance with the bus resonator (not shown) and stays on resonance long enough ($\sim 9\,\mathrm{ns}$) to execute an iSWAP operation $|Q_2B\rangle = |e\,0\rangle \rightarrow i\,|g1\rangle$, which swaps the excited state of the qubit into the resonator. Later in the sequence, the second Z-pulse applied to Q_2 returns the excitation back to the qubit via a second iSWAP operation. The third Z-pulse pulse is a small Gaussian-smoothed rectangular bump right after the second iSWAP operation that corrects for the dynamically acquired phase of Q_2. And the final Z-pulse is the measurement pulse.

The Z-pulses for the target qubit Q_3 do the following: The first pulse tunes the second excited level $|e\rangle \leftrightarrow |f\rangle$ (this is the "$|e\rangle \leftrightarrow |f\rangle$ transition", typically about $-200\,\mathrm{MHz}$ from the qubits $|g\rangle \leftrightarrow |e\rangle$ transition frequency) of the qubit on resonance with the bus resonator (not shown) and stays on resonance for the time ($\sim 15\,\mathrm{ns}$) required to do a 2π rotation i.e. $\times 2$ iSWAP operations between the states $|Q_3B\rangle : |e1\rangle \leftrightarrow |f0\rangle$. This action is conditioned on the state of Q_2. If Q_2

was excited and transferred that excitation to B (as described above) then Q_3 picks up the phase $\phi = \pi$. The second Z-pulse is the bump to correct for the dynamic phase acquired from detuning Q_3 from its IDLE bias. The final Z-pulse is the measurement pulse.

Figure 5.15, is the real part of the density matrix reconstructed from QST on $|\psi_s\rangle$. The singlet is formed with fidelity $F_{Bell} = \langle\psi_s|\,\rho_{Bell}\,|\psi_s\rangle = 0.75 \pm 0.01$ ($|\mathrm{Im}\,\rho_{\psi_s}| < 0.05$ not shown) and entanglement of formation EOF = 0.43.

5.6.2 Step 2: GHZ States After Two CNOT Gates

The algorithm is paused after the second CNOT gate between Q_2 and Q_4 to check for tripartite entanglement[48, 47, 19]. The actual pulse sequence used to generate this state is shown in Figure 5.16. The sequence builds on the previous sequence for the Bell state with the additional qubit Q_4, H-gates and second C_z.

At this breakpoint in the algorithm a three-qubit $|\mathrm{GHZ}\rangle = (|ggg\rangle + |eee\rangle)/\sqrt{2}$, with fidelity $F_{GHZ} = \langle GHZ|\,\rho_{GHZ}\,|\mathrm{GHZ}\rangle = 0.59 \pm 0.01$ ($|\mathrm{Im}\,\rho_{\mathrm{GHZ}}| < 0.06$ not shown) is formed between Q_2, Q_3, and Q_4 as shown in Figure 5.17. This state is found to satisfy the entanglement witness inequality, $F_{GHZ} > 1/2$ [1] indicating three-qubit entanglement. We note that this $|\mathrm{GHZ}\rangle$ measurement together with the $|\mathrm{W}\rangle$ measurement above in §5.4.2 is the first measurement employing *simultaneous* measurement with single-shot readout of the qubits for both classes of

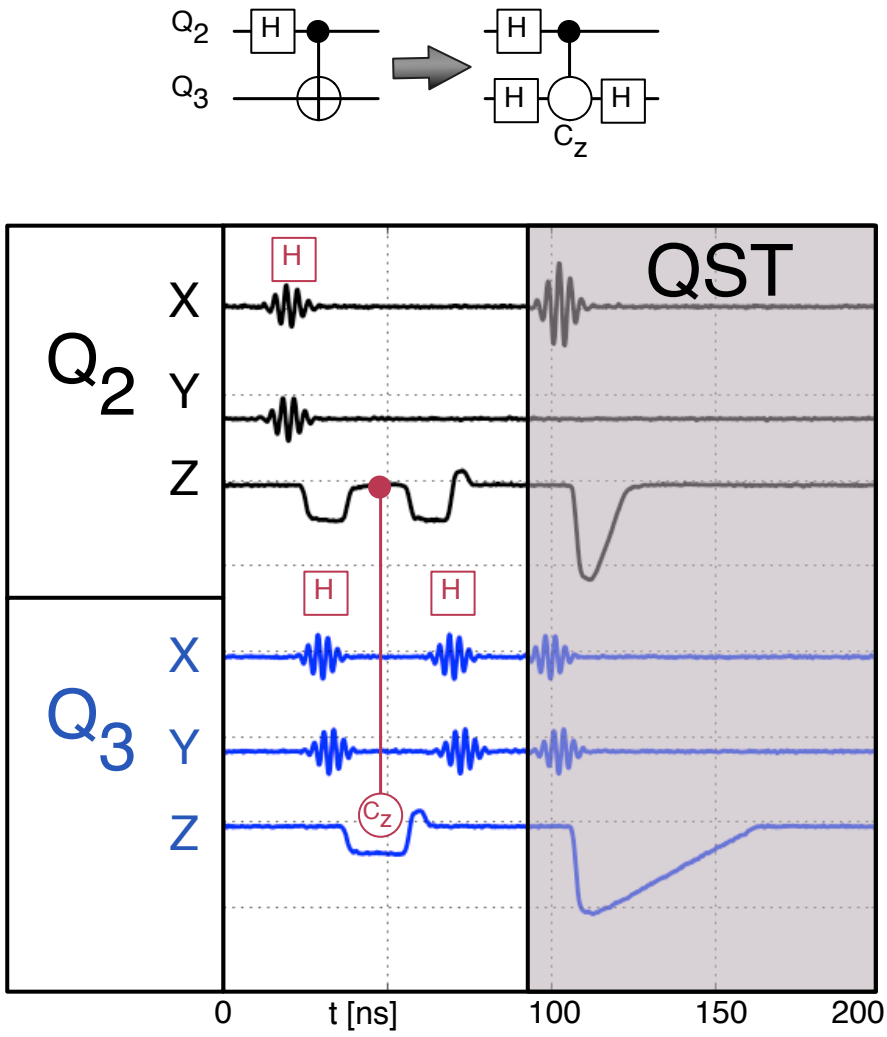

Figure 5.14: Control pulse sequence for the first breakpoint in the quantum runtime analysis. Bell state created followed by QST. Note that the CNOT gate is realized by equivalent Controlled-Z gate sandwiched between two H-gates.

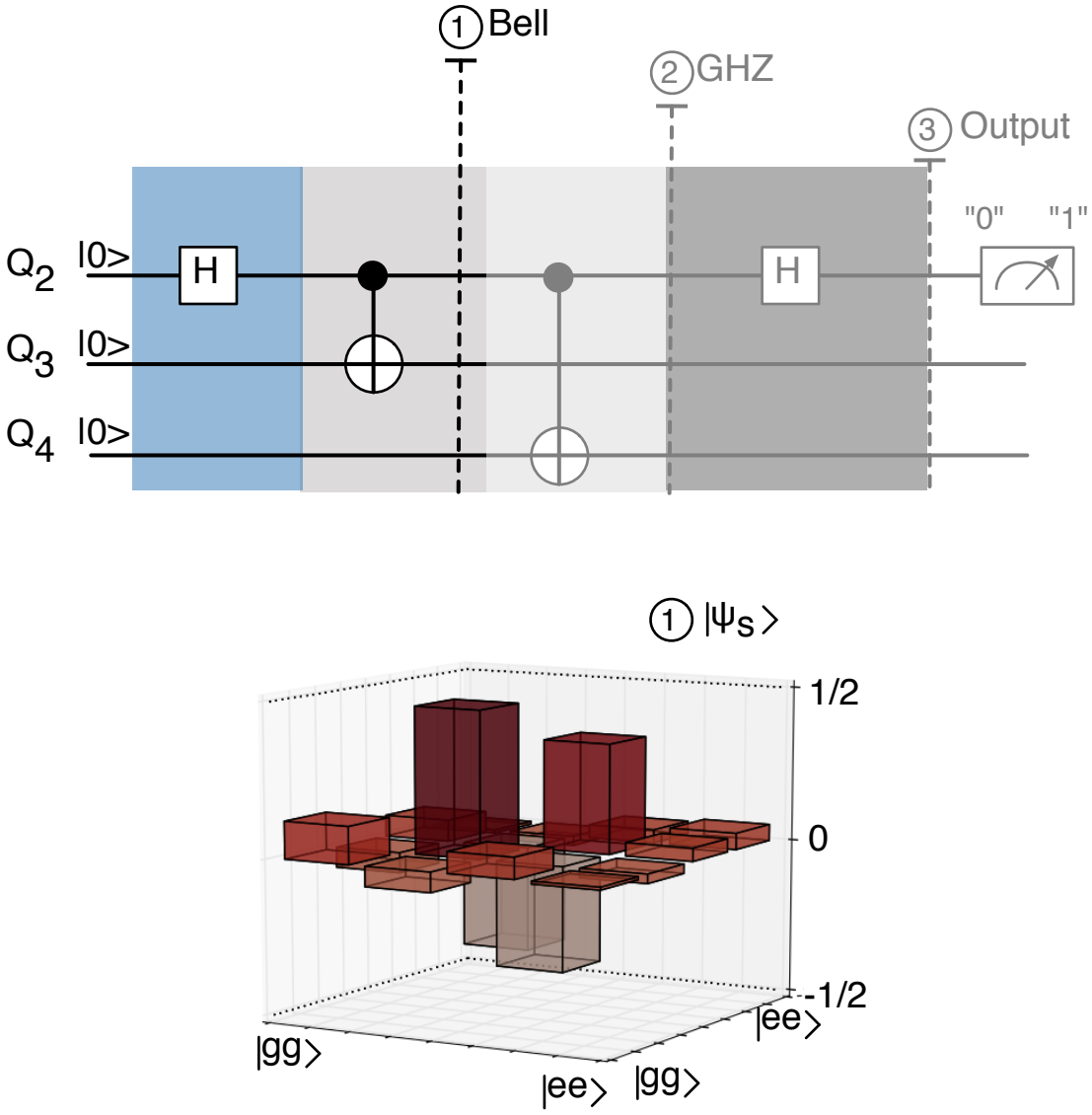

Figure 5.15: Reconstructed density matrix from QST.

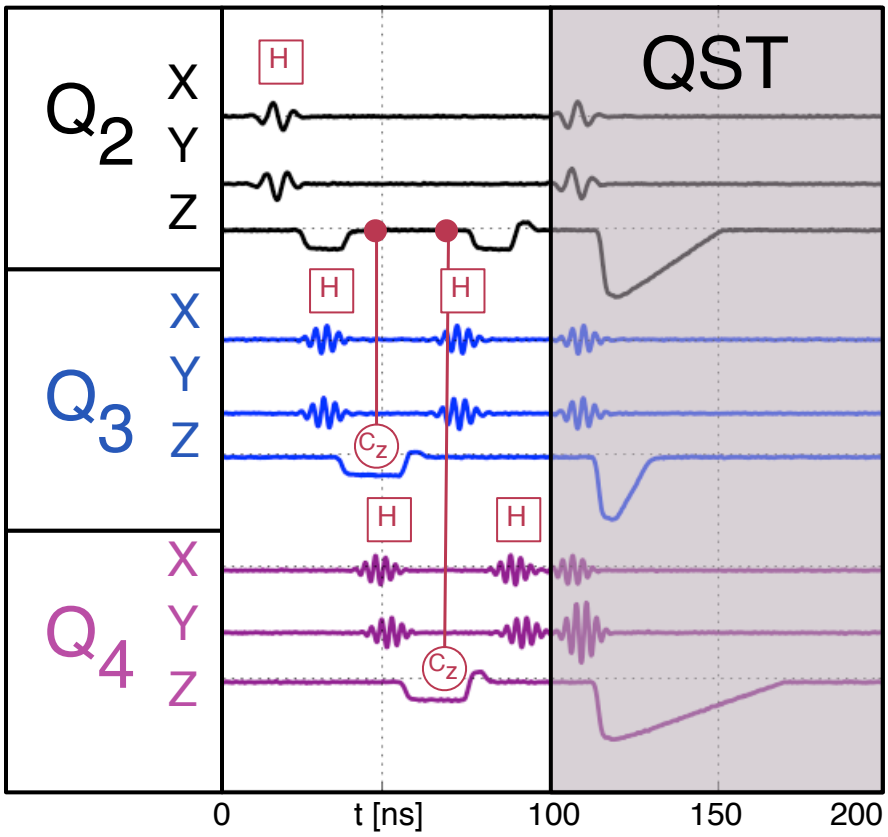

Figure 5.16: Control pulse sequence for the second breakpoint in the quantum runtime analysis. GHZ state created followed by QST.

three-qubit entanglement.

5.6.3 Step 3: Three Qubit QST

The third step in the runtime analysis captures all three qubits at the end of the algorithm, where the final H-gate on Q_2, rotates the three-qubit $|\text{GHZ}\rangle$ into $|\psi_3\rangle = H_2 |\text{GHZ}\rangle = (|ggg\rangle + |egg\rangle + |gee\rangle - |eee\rangle)/2$. The actual control pulses are shown and labeled in Figure 5.18.

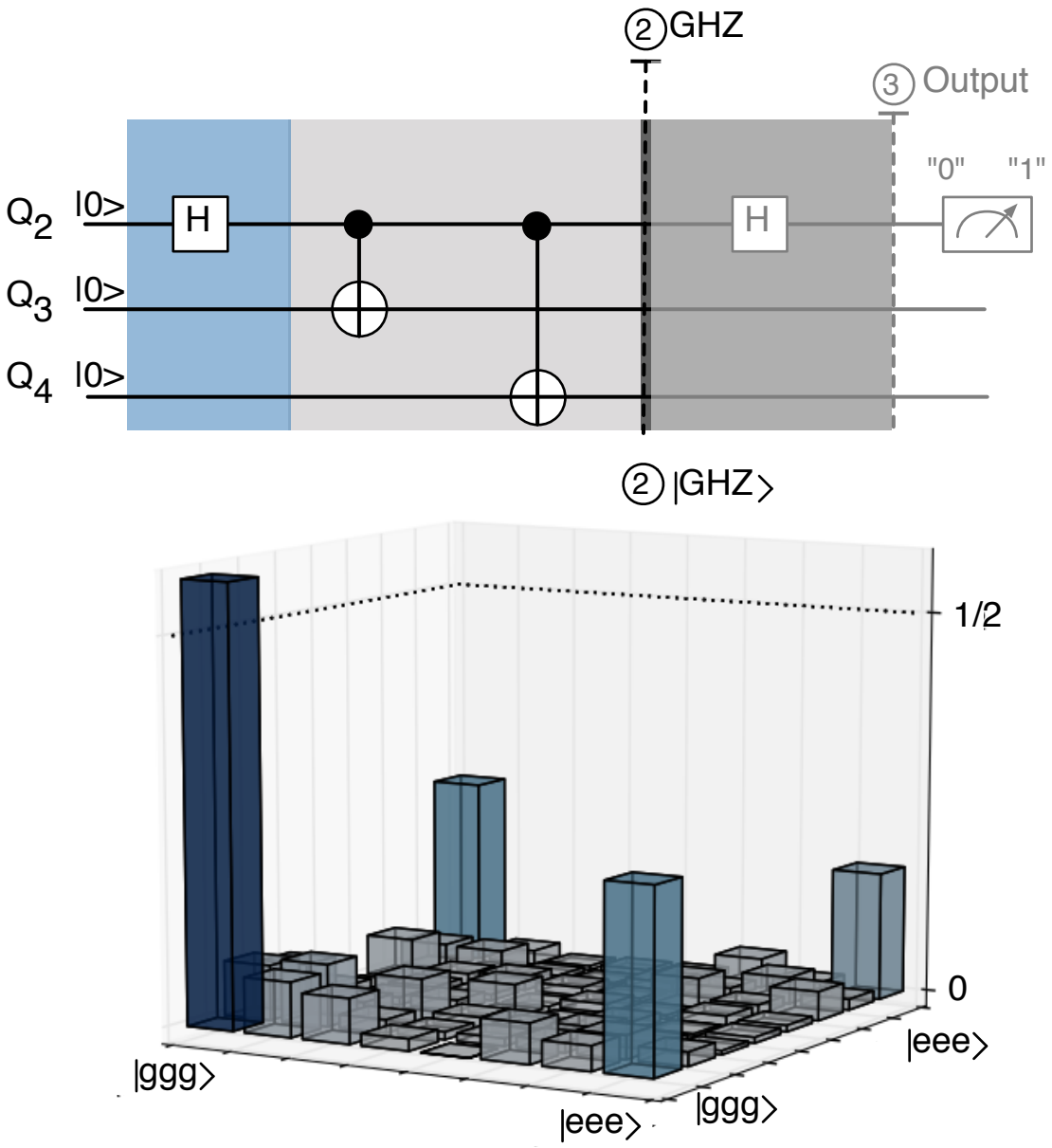

Figure 5.17: Reconstructed density matrix from QST.

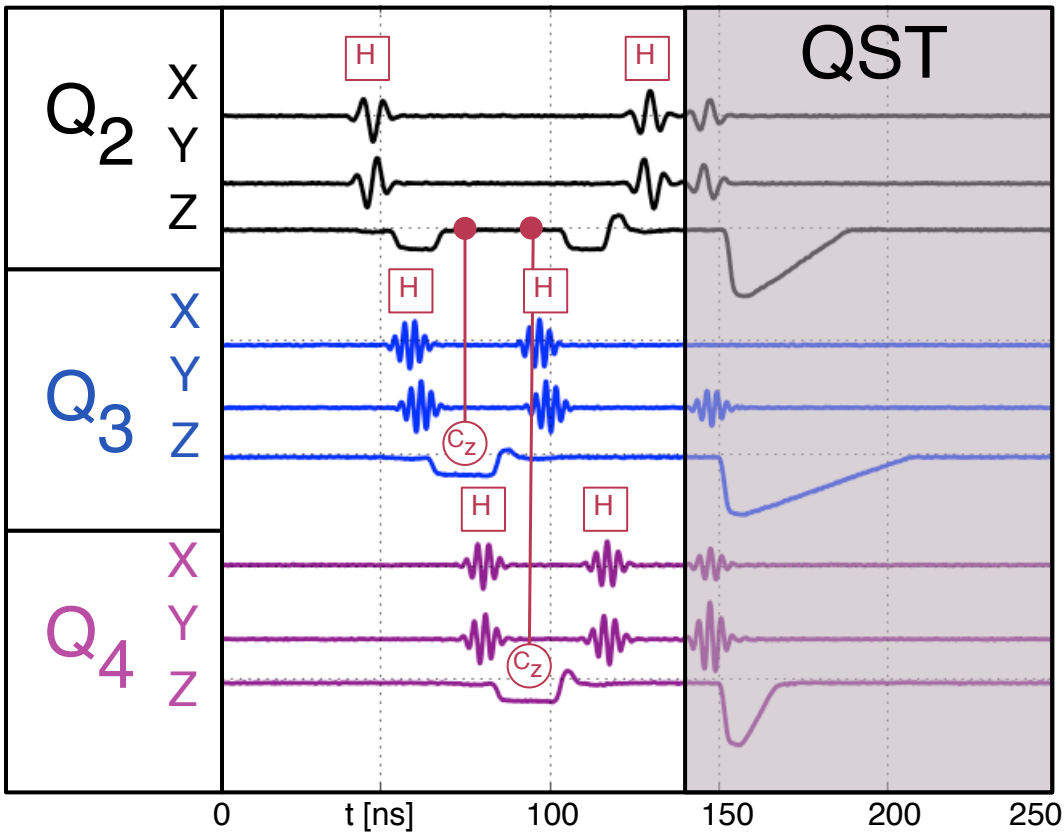

Figure 5.18: Control pulse sequence for thee-qubit Shor algorithm.

The middle panel in Figure 5.19 is the real part of the density matrix with fidelity $F = \langle\psi_3|\,\rho_3\,|\psi_3\rangle = 0.54 \pm 0.01$. Because the state is locally equivalent to a $|\mathrm{GHZ}\rangle$ state we still have violate the three qubit entanglement witness $F > 1/2$. From the three-qubit QST we can trace out the register qubit to compare with the experiment, where we measure only the single qubit register and the raw probabilities of the algorithm output, which we discuss next.

5.7 Shor's Algorithm Output

Although the success of the algorithm hinges on quantum entanglement, the final output is ideally a completely mixed state, $\sigma_m = (1/2)(|g\rangle\langle 0| + |e\rangle\langle 1|)$. Therefore, measuring only the raw probabilities of the output register does not reveal the underlying quantum entanglement necessary for the success of the computation. Thus, we perform QST at the end of the algorithm in addition to recording the raw probabilities of the output register.

Ideally, the algorithm returns the binary output "00" or "10" (including the redundant qubit) with equal probability, where the former represents a failure and the latter indicates the successful determination of $r = 2$. We use three methods to analyze the output of the algorithm: Three-qubit QST, single-qubit QST, and the raw probabilities of the output register state.

5.7.1 Three-Qubit QST and Single-Qubit QST

Single qubit QST captures only what happens to the output register qubit and disregards (does not measure) the functional qubits (Q_3 and Q_4). However, QST on the output register does provide phase information which can be useful in verifying the correct behavior of the algorithm.

The bottom two panels in Figure 5.19 are the real part of the density matrices for the single qubit output register from three-qubit QST and one-qubit QST

with fidelity $F = \sqrt{\rho}\,\sigma_m\,\sqrt{\rho} = 0.92 \pm 0.01$ for both density matrices. The density matrices were formed by: tracing-out Q_3 and Q_4 from from the data shown in the middle panel (blue bars), and directly measuring Q_2 with QST. The data are equivalent, as we expect.

5.7.2 Raw Probabilities

The raw probabilities of the output register provide the direct answer of the algorithm. From the raw probabilities calculated from 150,000 repetitions of the algorithm, we measure the output "10" with probability 0.483 ± 0.003, yielding $r = 2$, and after classical processing we compute the prime factors of $N = 15$, with a co-prime via $\mathrm{GCD}[(a^{r/2} \pm 1), N]$:

$$\begin{aligned} p &= \mathrm{GCD}[4^{2/2} + 1, 15] = 3 \\ q &= \mathrm{GCD}[4^{2/2} - 1, 15] = 5, \end{aligned} \tag{5.2}$$

and we find

$$\begin{aligned} N &= p \times q \\ 15 &= 3 \times 5 \end{aligned} \tag{5.3}$$

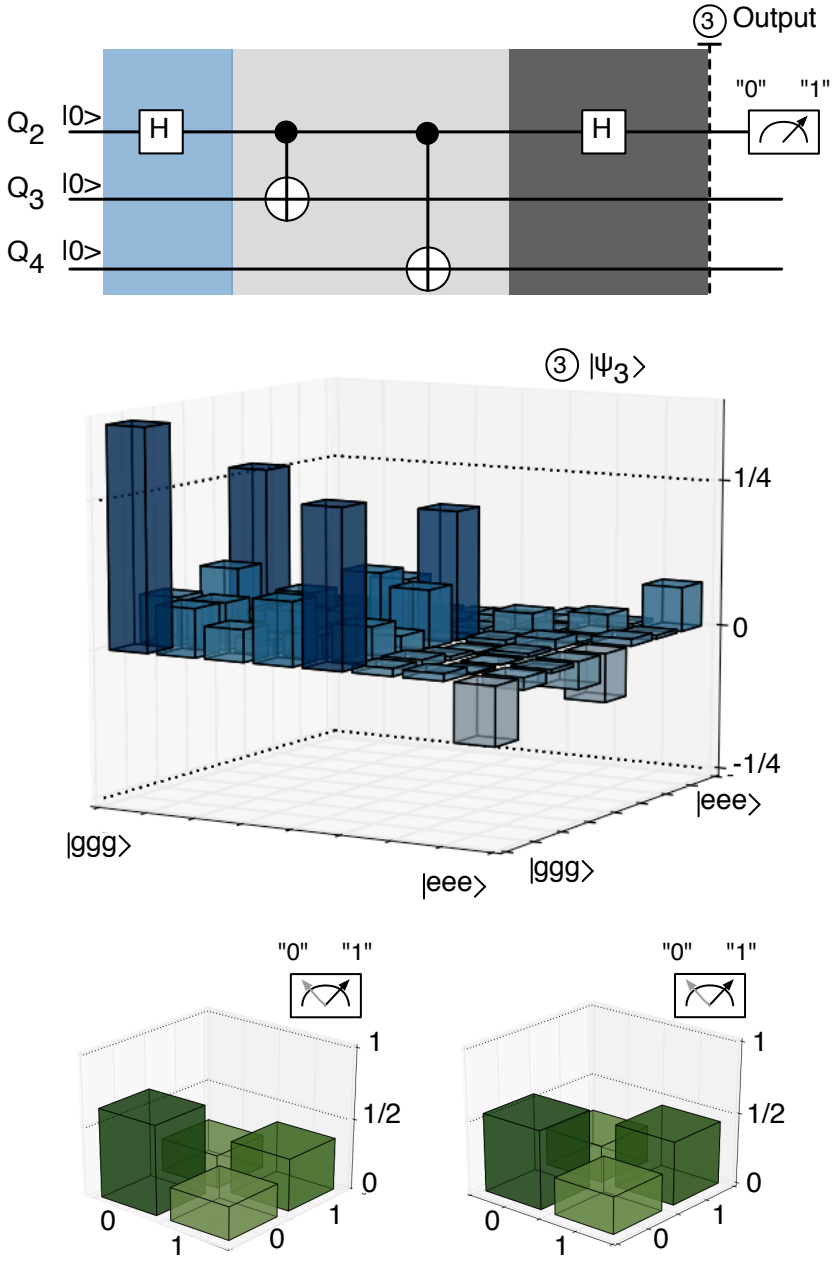

Figure 5.19: Output of the Shor Algorithm. Reconstructed density matrices: from full three-qubit QST, single-qubit density matrix via tracing out Q_2 and Q_3, and single-qubit density matrix from single-qubit QST.

5.7.3 Linear Entropy of The Output Register

The linear entropy $S_L = 4[1 - \mathrm{Tr}(\rho^2)]/3$ is another metric for comparing the observed output to the ideal mixed state, where $S_L = 1$ for a completely mixed state[68]. We find $S_L = 0.78$ for both the reduced density matrix from the third step of the runtime analysis (three-qubit QST), and from direct single-qubit QST of the register qubit.

5.7.4 Check Experiment: No Entangling Operations

As a final check, we run the algorithm without any of the entangling operations and compare the output, both single QST and raw probabilities to those of the actual algorithm. The top panel in Figure 5.20 shows the reduced quantum circuit, with entangling operations removed. The algorithm reduces to two H-gates separated by the time of the two entangling gates. Ideally Q_2 returns to the ground state and the algorithm fails (returns "0") 100 % of the time. The bottom panel in Figure 5.20 is the real part of the density matrix for the register qubit after running this check experiment. The fidelity of measuring the register qubit in $|g\rangle$ is $F_{check} = \langle g| \rho_{check} |g\rangle = 0.83 \pm 0.01$. The algorithm fails, as expected, without the entangling operations.

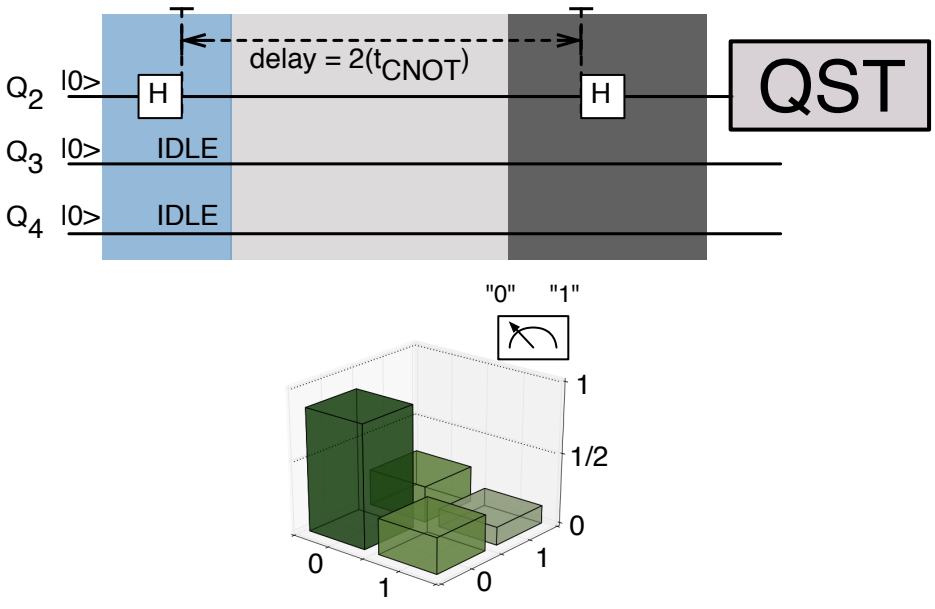

Figure 5.20: Check experiment. Run algorithm without entanglement.

5.8 Sources of Error

Short coherence times, both T_1 and T_2, are the largest sources of error, followed by the presence of two-level states (TLS) that the qubits inevitably couple to while tuning the qubits on and off resonance with the resonators. Smaller junctions were engineered to help reduce the density of TLS in the qubit spectrum, however the density of TLS in the QuP are still a source of decoherence. Perhaps even smaller junctions (with areas less than $1\,\mu\text{m}^2$) will reduce the density of TLS to the point that the errors they cause can be neglected. Another option is to deploy a control scheme that does not require the qubits to be tuned or detuned for

coupling interactions, thereby reducing the participation of the TLS.

5.9 Conclusion: $15 = 3 \times 5$

In conclusion, we have implemented a compiled version of Shor's algorithm on a modular nine quantum element QuP that correctly finds the prime factors $p = 3$, $q = 5$ of the composite number $N = 15$. We showed that the QuP can create Bell states, both classes of three-qubit entanglement $|GHZ\rangle$ and $|W\rangle$, and the requisite entanglement to execute Shor's algorithm. In addition, we produce coherent interactions between four qubits and the bus resonator with a protocol that can be scaled to rapidly create an N-qubit $|\mathrm{W}\rangle$ state. During these multi-qubit coherent interactions we also observe a $\sqrt{N}$ dependence of the effective coupling strength with the number N of participating qubits consistent with theoretical predictions. These demonstrations represent an important milestone for superconducting qubits, further proving this architecture for quantum computation and quantum simulations.

Appendix A

Daily Automated Calibrations

In this chapter, I discuss the software automation developed over the years to calibrate phase qubits. The software is general enough to accommodate a variety of quantum elements, e.g. transmon-type qubits, tunable resonators, resonator readout schemes, *etc.*, although such a discussion is left for another thesis. We have found that automating the repetitive calibration tasks is, and will continue to be, essential to the success of any quantum architecture that is serious about scaling. A characteristic for any software infrastructure is the flexibility to evolve so as to continue to meet the needs of future experiments, as opposed to recreating the framework for every subsequent evolution in hardware. To that end, I am pleased to note that just as this thesis was built on previous experiments, the code I review here is already evolving with further automation for future experiments

in the laboratory. I stress that the capability for this automation has been a UCSB QC-group effort, but is largely due to the foresight of Markus Ansmann and Matthew Neeley, who birthed LabRAD[1] to provide the infrastructure for this continuos software evolution, and Max Hofheinz for the initial automated microwave electronics calibrations.

What I name the "manual" procedure for single-qubit calibration has been described in great detail previously in [3, chap. 8]; here, l focus on the sequence of the experiments[2], show which parameters are calibrated, and provide the (commented) code used for the calibrations. This appendix begins with a brief outline of the current state of the LabRAD infrastructure (for more on the origin and philosophy of LabRAD please see[3, chap. 6]) and how it constitutes the qubit control channels. The rest of the appendix is dedicated to the automated phase qubit calibrations beginning in §A.3. In §A.3.1, using a top-level diagram, I describe the experimental software interface and how a calibration script updates qubit parameters. In §A.3.2, I detail the 41 separate parameters that need to be calibrated for *every* qubit before running a quantum algorithm. I categorize the automation into functional blocks of the phase qubit calibrations starting with the dc-bias parameters in §A.4.2, then the measurement parameters in §A.4.3, qubit

[1] http://sourceforge.net/projects/labrad/

[2] The sequence of experiments has been constructed to facilitate bootstrapping. In this context bootstrapping means the experiments are executed such that the results from the current experiment feed into the next.

X,Y pulse control calibrations in §A.4.4, the single qubit scans in§A.4.5, qubit-resonators calibrations in §A.4.6, and finally the coupled qubit gate calibrations in §A.4.7.

A.1 The Correct (Software) Tool For The Job

A high-level block diagram in Figure A.1 shows the software tools that we have chosen to facilitate the various tasks related to data-taking.[3] Within the UCSB QC-group there continues to be significant development to make as much of LabRAD rely on free-open-source software and phase-out the closed-source software (i.e. Labview and Delphi). As can be seen in Fig.A.1, Python plays a dominate role in our experiments. Here, I will be focusing solely on the Python scripts that have been created to run the qubit calibrations and experiments.

A.2 Qubit Control Channels and The Pyle

The point of this software infrastructure is to abstract-away the hardware to the point where one can still accurately, yet easily, program the experiment at a high-level, while still having access to all of the low-lying constituents for precise

[3] I'll save everyone from the long winded debate over which language is best suited for which task, and instead just comment that there were strong opinions rooted from the original creators of LabRAD. More importantly, LabRAD is an opensource project so if you think you can make any portion better with another language we all encourage you to do it and share it!

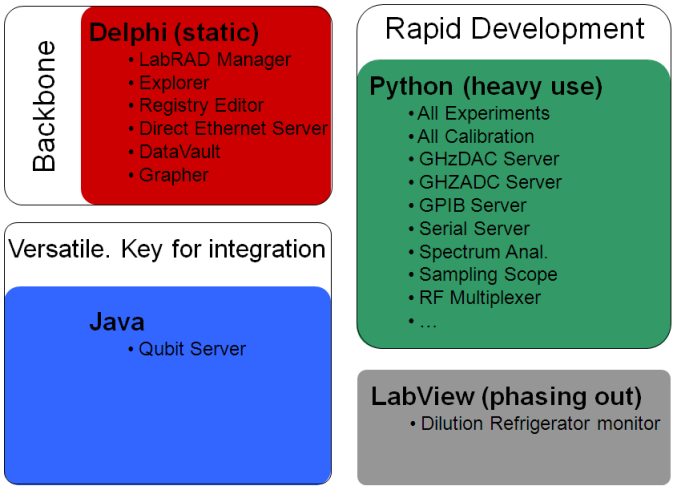

Figure A.1: The software languages and their use in experiments.

modifications to the calibrations when necessary. At a high-level this is a quantum-circuit-schematic, where the underlying qubits and control electronics have been completely calibrated and abstracted away such that all the experimenter needs to be aware of are the "quantum-resources" available to them. This abstraction is akin to programming a field programmable gate array (FPGA) using a schematic capture of boolean logic, where details of the interior gate connections are not needed. In the experimental scripts described here, the level of abstraction is to the point where we build up individual pulse sequences to form the calibrations. Therefore, the high-level coding of quantum circuit control software is well within reach.

Figure A.2 illustrates the current level of abstraction of our qubit channels. The mid-line in the figure provides a boundary between the software and the

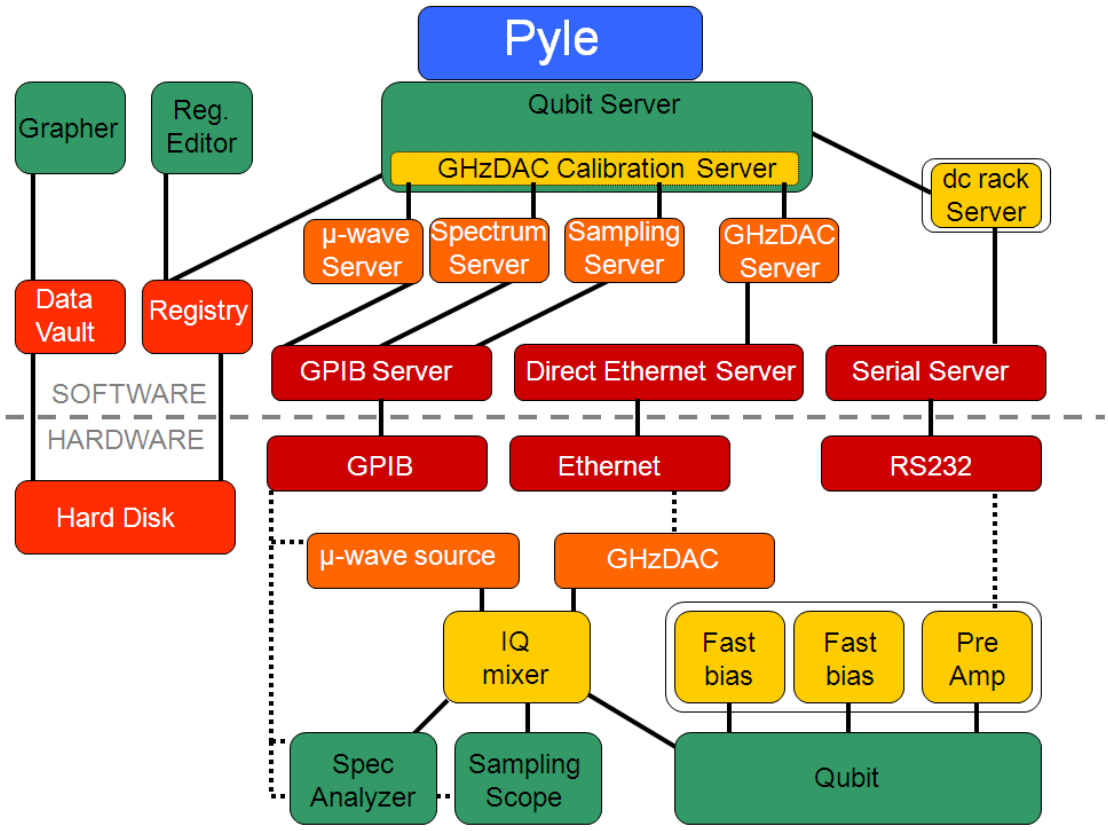

Figure A.2: Qubit Control channels in Software and Hardware.

hardware, which are nearly mirror-images of one another. The symmetry fades as we get to the top-level abstraction in the software, titled "Pyle". Pyle (as in a "pile" of code) is a repository for our experimental scripts. When composing an experiment, to later become an experimental calibration, the experimenter has access to all of the resources shown in Fig.A.2. Typically, the scripts stay within the qubit server abstraction, but one of the advantages of LabRAD is the user-defined level of granularity. One can make a call to the underlying abstraction layers, like the lower-level general purpose interface bus (GPIB) server if needed. For our discussions here we will be operating at the "Pyle" level where we are only concerned with making calls to qubit parameters handled in the Qubit Server.

A.3 Automated Qubit Calibrations

A.3.1 Experimental Interface

An experiment is run by initiating a script via a (python) command window (step 1 in Figure A.3). Upon execution of the script the existing calibrations are applied to the system (step 1.a in Figure A.3) and the qubit responds (step 2 in Figure A.3). The data are recorded in the Data Vault (step 3), plotted in the grapher (step 3.a) if desired, and the desired data values are returned to the script for calibrations and immediate analysis. If an update flag is set true, the

registry values are updated with the new calibration values (step 4). The registry contains the ever growing qubit parameters that describe the experiments. The list of qubit parameters (or "registry keys") are tabulated in Table A.1, A.2, A.3, A.4, and A.5.

A.3.2 41 Automatic Calibrations per Qubit

Figure A.4 shows a representative pulse sequence for a single repetition of a single qubit experiment. All of the arrows (and numbers) indicate a qubit parameter that must be calibrated. The relevant experimental times are indicated in each section, one repetition of an experiment takes $\sim 100\,\mu$s to complete. Typically each data point is repeated $\times 1000$. Therefore, a fast one-parameter ("1-D") sweep, consisting of ~ 1000 points, takes $O(100\,\text{sec})$ or a couple of minutes. Typically the calibration analyses consists of finding a min (max), calculating a period (via a fast Fourier transform), or fitting a well defined function -all of which are relatively quick calculations on a modest desktop computer - adding a $O(100\,\text{sec})$ to the experiment. For two-parameter ("2-D") sweeps one may need to optimize the range of values so as to help reduce the calibration time. These scans are typically are preceded by quick 1-D scans to find the appropriate range.[4]

[4]When possible, it is usually desirable and more efficient to substitute a series of 1-D scans for a a single 2-D calibration.

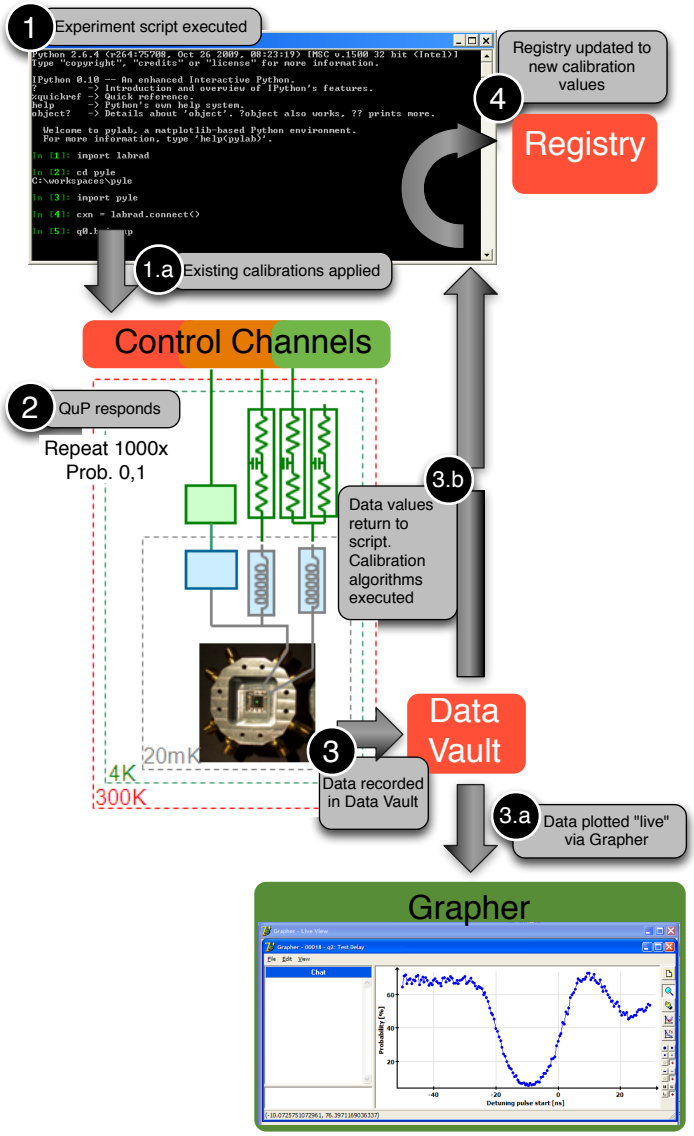

Figure A.3: Qubit Calibration Flow illustration. Steps 1 through 4 are detailed in the text. Control channels refers to the hardware and software infrastructure shown in Figure A.2.

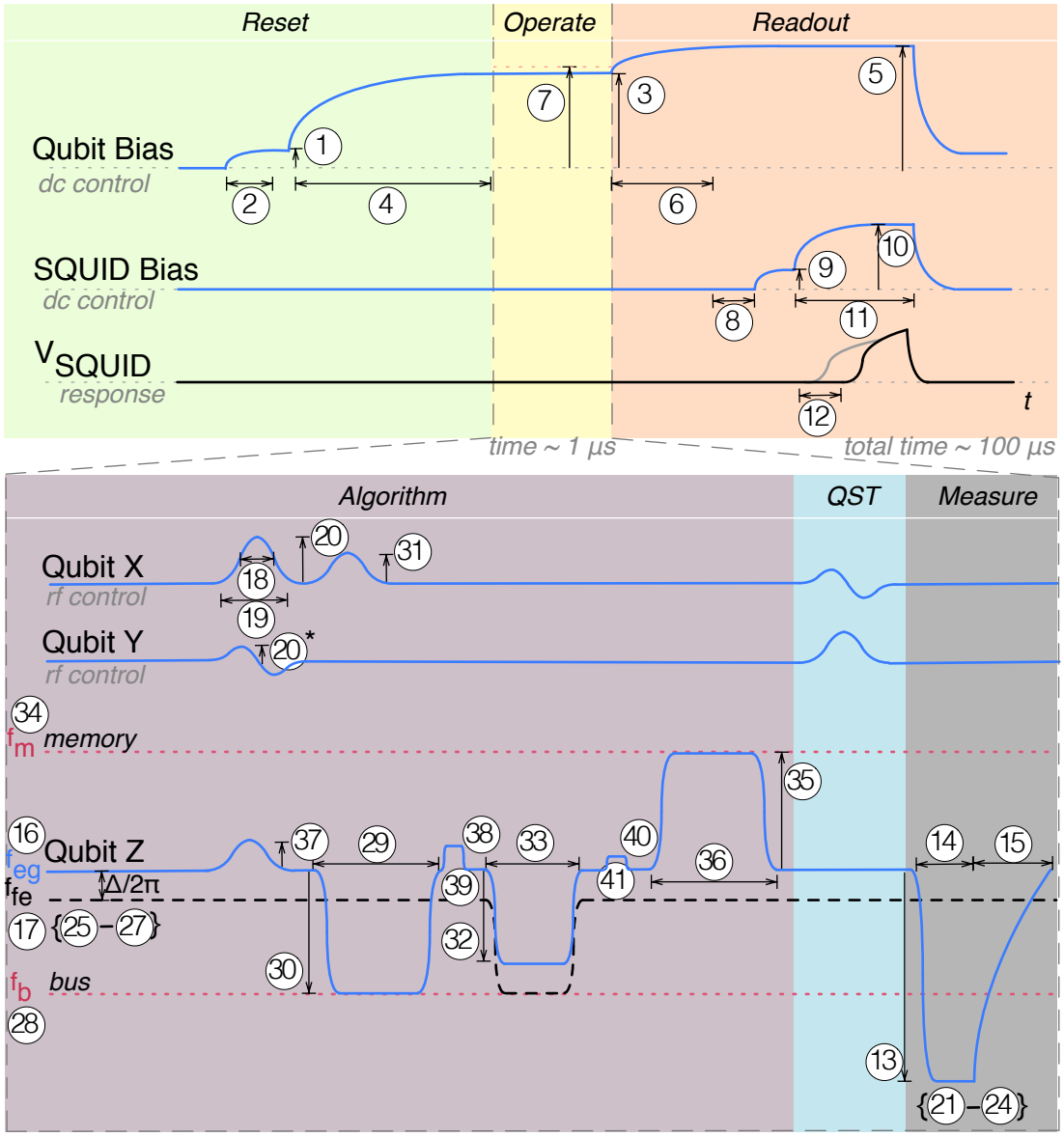

Figure A.4: Qubit parameters annotated by number, corresponding to registry keys in Table A.1, A.2, A.3, A.4, and A.5. Scales are exaggerated for clarity.

Daily Qubit Bias Calibrations				
	Parameter	Calibration	Fine Cal.	Typical Value
1	biasReset	SQUID steps		0.110 V
2	biasResetSettling	SQUID steps		8 μs
3	biasOperate	SQUID steps	Step edge	0.513 V
4	biasOperateSettling	SQUID steps		40 μs
5	biasReadout	SQUID steps		0.740 V
6	biasReadoutSettling	SQUID steps		20 μs
7	biasStepEdge	SQUID steps	Step edge	0.455 V
8*	SQUIDReadoutDelay	SQUID steps		10 μs
9	SQUIDRampBegin	SQUID steps		0.1 V
10	SQUIDRampEnd	SQUID steps		0.5 V
11	SQUIDRampLength	SQUID steps		50 μs
12	SQUIDSwitchTime	SQUID steps		37 μs

Table A.1: Table of qubit experimental bias parameters (written as they appear in the registry) for a typical qubit in the QuP. The Calibration and "Fine Cal." columns refer to the experimental calibration script detailed in Figure A.10. *Parameter 8 does not need to be calibrated day to day.

Daily Qubit Measurement Calibrations				
	Parameter	Calibration	Fine Cal.	Typical Value
13	measAmp	scurve	findMPA	−0.85 DAC Ampl.
14	measLenTop	scurve	findMPA	5 ns
15	measLenFall	scurve	findMPA	30 ns

Table A.2: Table of qubit experimental measurement parameters (written as they appear in the registry) for a typical qubit in the QuP.

Daily Qubit X,Y Pulse Calibrations				
	Parameter	Calibration	Fine Cal.	Typical Value
16	f_{eg}	Spectroscopy low power	Ramsey via freqTuner	6.54 GHz*
17	f_{fe}	Spectroscopy high power	Ramsey eror filter (REF)	6.44 GHz*
18*	piLength	*		14 ns
19*	piFWHM	*		7 ns
20	piAmp	piTunerHD		0.70
21	measEg	Visibility		0.054
22	measEe	Visibility		0.090
23	measFg	Visibility		0.946
24	measFe	Visibility		0.910
25	calRabiOvrUw	2-D Spec, Pituner		0.19 GHz
26	calZpaFunc	2-D Spec, findZpa-Func		(0.0, 0.994)
27	calDfOvrZpa	calZpaFunc, f_{eg}		−1.645 GHz

Table A.3: Table of qubit pulse parameters (written as they appear in the registry) for a typical qubit in the QuP.

Daily Qubit-Resonator Calibrations				
	Parameter	Calibration	Fine Cal.	Typical Value
28	f_B	SWAP10	SWAP10-Tuner, fockTuner	6.10 GHz*
29	cZContrlLen	SWAP10-Tuner	fockTuner	12.56 ns
30	cZContrlAmp	SWAP10-Tuner	fockTuner	−0.239 DAC Ampl.
31	piAmpfe	piTunerHDfe		0.492 DAC Ampl.
32	cZTargetAmp	SWAP21-Tuner, piTuner21	fockTuner $n = 2$	−0.160 DAC Ampl.
33	cZTargetLen	SWAP21-Tuner, piTuner21	fockTuner $n = 2$	15.40 ns
34	f_M	SWAP10	fockTuner	7.0 GHz
35	memRWAmp	SWAP10-Tuner	fockTuner	0.147 DAC Ampl.
36	memRWLen	SWAP10-Tuner	fockTuner	22.38 ns

Table A.4: Table of qubit experimental parameters for qubit-resonator calibrations.

Daily Qubit Gate Calibrations				
	Parameter	Calibration	Fine Cal.	Typical Value
37	piAmpZ	piTunerZ		0.042 DAC Ampl.
38	cZContrlPhaseCorAmp	cZCalP1	semi-auto	0.129 DAC Ampl.
39	cZContrlPhaseCorLen	*		5 ns
40	cZTargetPhaseCorAmp	cZCalP2	semi-auto	0.223 DAC Ampl.
41	cZTargetPhaseCorLen	*		5 ns

Table A.5: Table of qubit experimental parameters for gate calibrations.

A.4 Daily Automation Code

A.4.1 Top Level Function Calls

Figure A.5 shows the code for the daily automations. The header (all of the "from" and "import" statements) is for proper linking of resources. The function titled "daily bringup" defines the sequence of calibration experiments that are run at the beginning of everyday. For four qubits and five resonators the daily calibrations were completed after $\sim 4\,\text{hrs}$.[5]

A.4.2 Bias Calibrations

Figure A.6 through A.9 shows the code for the automated bias experiments. These experiments calibrate the parameters summarized in Table A.1. The biasReset parameter is the voltage to reset the qubit. The biasResetSettling parameter is the time to wait for the qubit bias to settle at its reset voltage. The biasOperate parameter is the qubit operating voltage. The biasOperateSettling parameter is the time to wait for the qubit bias to settle at its operating voltage. The biasReadout parameter is the qubit readout voltage. The biasReadoutSettling parameter is the time to wait for the qubit bias to settle at its readout voltage. The biasStepEdge

[5]Typically, my days started by waking up, logging-in remotely to initiate the "daily bringup" script. Then I would enjoy a cup of coffee, some breakfast, and commute (ride my bike) into the laboratory in time to verify the end of the calibrations and begin the higher level Shor experiments.

parameter sets the maximum bias voltage before the qubit well disappears. The SQUIDReadoutDelay parameter is the time to wait before ramping the SQUID voltage for readout of the qubit state. The SQUIDRampBegin (SQUIDRampEnd) is the beginning (ending) voltage of the SQUID ramp. The SQUIDRampLength is the length of the SQUID ramp. The SQUIDSwitchTime is the the time at which the SQUID switches into the voltage state (based on a the comparator threshold voltage).

A.4.3 Measurement Calibrations

Figure A.10 and A.11 shows the code for the automated measurement experiments. These experiments calibrate the parameters summarized in Table A.2. The measAmp parameter is the DAC amplitude set to measure the excited $|e\rangle$ state. The measLenTop parameter is the length of time that the measAmp is sustained. The measLenFall parameter is the length of time until the measAmp returns to zero.

A.4.4 Qubit X,Y Pulse Control Calibrations

Figure A.12 through A.21 shows the code for the qubit X,Y pulse calibrations for the parameters summarized in Table A.3. The parameter f_{eg} is the qubit $|g\rangle \leftrightarrow |e\rangle$ transition frequency. The parameter f_{fe} is the qubit $|e\rangle \leftrightarrow |f\rangle$ transi-

tion frequency. The piLength parameter is the full length of a π-pulse, which is set manually and is no less than twice piFWHM. The piFWHM is the FWHM length of a π-pulse, which is also set manually (see Chapter 4 for more details on how the appropriate length for these values). The piAmp parameter is the DAC amplitude caibrated for a π-pulse. The measEg parameter corresponds to the stray tunneling of the $|g\rangle$ state. The measFg parameter corresponds to the probability of measuring the $|g\rangle$ state ($1 -$ measEg $=$ measFg). The meas Fe parameter is the probability of measuring the excited $|e\rangle$ state. The measEe parameter is the amount that the $|e\rangle$ state is misidentified as the ground $|g\rangle$ state. The calRabiOvrUw parameter is the calibration that converts Rabi-amplitude to a frequency. The calZpaFunc parameter is the calibration that fits (a polynomial to) the spectroscopy curve, which is used for the calDfOverZpa calibration. The calDfOverZpa parameter is the calibration that converts z-amplitude to detuning frequency.

A.4.5 Single Qubit Scans

Figure A.22 through A.25 shows the code for the standard single qubit scans, T_1, T_2, and spin echo.

A.4.6 Qubit-Resonator Calibrations: Bus and Memory

Figure A.26 through A.33 shows the code for the qubit-resonator calibrations for the parameters summarized in Table A.4. The f_B parameter is the resonant frequency of the bus resonator. The cZContrlLen parameter is the length to perform an iSWAP between the qubit and bus resonator. The cZContrlAmp parameter is the z-amplitude to tune the qubit frequency f_{eg} on resonance with the bus resonator (to perform an iSWAP between the qubit and bus resonator). The piAmpfe parameter is the DAC amplitude to drive the a π-pulse between the $|e\rangle \rightarrow |f\rangle$ states. The cZTargetAmp parameter is the z-amplitude to tune the qubit frequency f_{fe} on resonance with the bus resonator (to perform an iSWAP^2 operation between the qubit and bus resonator). The cZTargetLen parameter is the length to perform an iSWAP^2 between the qubit and bus resonator. The f_M parameter is the resonant frequency of the memory resonator. The memRWAmp parameter is the z-amplitude to tune the qubit frequency f_{eg} on resonance with the memory resonator (to perform an iSWAP between the qubit and memory resonator for memory read and write operation). The memRWLen parameter is the length to perform an iSWAP between the qubit and memory resonator.

A.4.7 Gate Calibrations

Figure A.34 through A.36 shows the code for the gate calibrations for the parameters summarized in Table A.5. The piAmpZ parameter is the DAC amplitude to perform a π-pulse about the z-axis. The cZControlPhaseCorAmp parameter is the DAC amplitude to adjust the phase after completing a Controlled-Z gate for the control qubit. The cZControlPhaseCorLen is the length of the phase correction detuning pulse, which is set manually. The cZTargetPhaseCorAmp parameter is the DAC amplitude to adjust the phase after completing a Controlled-Z gate for the target qubit. The cZTargetPhaseCorLen is the length of the phase correction detuning pulse, which is set manually.

```
from datetime import datetime
import itertools

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from scipy.optimize import leastsq, fsolve
import time
import random

import labrad

from labrad.units import Unit
ns, us, GHz, MHz = [Unit(s) for s in ('ns', 'us', 'GHz', 'MHz')]
#from labrad.scripts.test import GHz_DAC_brinigup_all


from pyle.dataking import measurement
from pyle.dataking import multiqubit as mq
from pyle.dataking import util
from pyle.util import sweeptools as st

from pyle.dataking import noon
from pyle.dataking import ghz
from pyle.dataking import werner
from pyle.dataking import shor as shor

from pyle.dataking import hadamard as hadi

from pyle.plotting import dstools as ds
from pyle import tomo

def daily_bringup(s, pause=True):

    bringupAll(s._cxn) #brings up the GHz_DACs first
    bringup_squidsteps(s, pause=pause)
    bringup_stepedge(s, pause=pause)
    bringup_scurve(s, pause=pause)
    bringup_sample(s, pause=pause)
    single_qubit_scans(s)
    qubit_coupling_resonator_scans(s)
    qubit_memory_resonator_scans(s)
    gate_bringup(s)
    create_bell_state_iswap(s,zSweep=False)
```

Figure A.5: Automate Daily scripts.

```
###### From automateDaily.py ########
def daily_bringup(s, pause=True):

    bringupAll(s._cxn) #brings up the GHz_DACs first
    bringup_squidsteps(s, pause=pause)
    bringup_stepedge(s, pause=pause)
   # bringup_scurve(s, pause=pause)
   # bringup_sample(s, pause=pause)
   # single_qubit_scans(s)
   # qubit_coupling_resonator_scans(s)
   # qubit_memory_resonator_scans(s)
   # gate_bringup(s)
   # create_bell_state_iswap(s,zSweep=False)

def bringup_squidsteps(s, pause=True):
    N = len(s['config']) #N=4 for the four phase qubits in the QuP
    for i in range(N):
        print 'measuring squidsteps for qubit %d...' % i,
        mq.squidsteps(s, measure=i, noisy=False, update=pause)
        print 'done.'

    N = len(s['config'])
    for i in range(N):
        print 'measuring step edge, qubit %d...' % i,
        mq.stepedge(s, measure=i, noisy=False, update=pause)
        print 'done.'

    for i in range(N):
        print 'binary searching to find step edge %d...' % i
        mq.find_step_edge(s, measure=i, noisy=False)
        print 'done.'

##### From multiqubit.py ####
def squidsteps(sample, bias=st.r[-2.5:2.5:0.05, V], resets=(-2.5*V, 2.5*V), measure=0, stats=150,
               save=True, name='SquidSteps MQ', collect=False, noisy=False, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]

    if 'squidBiasLimits' in q:
        default = (-2.5*V, 2.5*V)
        bias_lim = q['squidBiasLimits']
        if bias_lim != default:
            print 'limiting bias range to (%s, %s)' % tuple(bias_lim)
        resets = max(resets[0], bias_lim[0]), min(resets[1], bias_lim[1])
        bias = st.r[bias_lim[0]:bias_lim[1]:bias.range.step, V]

    axes = [(bias, 'Flux Bias')]
    deps = [('Switching Time', 'Reset: %s' % (reset,), us) for reset in resets]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, deps, measure=measure, kw=kw)

    def func(server, fb):
        reqs = []
```

Figure A.6: Automated SQUIDsteps page 1 of 2.

```
            q['readout'] = True
            reqs.append(runQubits(server, qubits, stats, raw=True))
        data = yield FutureList(reqs)
        if noisy: print fb
        returnValue(np.vstack(data).T)
    data = sweeps.grid(func, axes, dataset=save and dataset, noisy=False)

    if update:
        squid.adjust_squid_steps(Q, data)
    if collect:
        return data
```

Figure A.7: Automated SQUIDsteps page 2 of 2.

```
###### From automateDaily.py ########
def bringup_stepedge(s, pause=True):
    N = len(s['config'])
    for i in range(N):
        print 'measuring step edge, qubit %d...' % i,
        mq.stepedge(s, measure=i, noisy=False, update=pause)
        print 'done.'

    for i in range(N):
        print 'binary searching to find step edge %d...' % i
        mq.find_step_edge(s, measure=i, noisy=False)
        print 'done.'

##### From multiqubit.py ####

def stepedge(sample, bias=None, stats=300L, measure=0,
             save=True, name='StepEdge MQ', collect=False, noisy=True, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]

    if bias is None:
        stepedge = q['biasOperate'][mV]
        stepedge = st.nearest(stepedge, 2.0)
        bias = st.r[stepedge-100:stepedge+100:2, mV]

    axes = [(bias, 'Operating bias')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

    def func(server, fb):
        q['biasOperate'] = fb
        q['readout'] = True
        return runQubits(server, qubits, stats, probs=[1])
    data = sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)

    if update:
        squid.adjust_operate_bias(Q, data)
    if collect:
        return data

def find_step_edge(sample, stats=60, target=0.5, bias_range=None,
                   measure=0, resolution=0.1, blowup=0.05,
                   falling=None, statsinc=1.25,
                   save=False, name='StepEdge Search MQ', collect=False, update=True, noisy=True
                   ):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]

    axes = [('Flux Bias', 'mV')]
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure)

    if falling is None:
        falling = q['biasOperate'][V] > q['biasStepEdge'][V]
```

Figure A.8: Step edge code page 1 of 2

```
if bias_range is None:
    stepedge = q['biasOperate'][mV]
    stepedge = st.nearest(stepedge, 2.0)
    bias_range = (stepedge-100, stepedge+100)
interval = list(bias_range)

def sweep(stats=stats):
    yield 0.5*(interval[0]+interval[1]), stats
    lower = True
    coeffs = 0.25, 0.75
    while interval[1] - interval[0] > resolution:
        stats *= statsinc
        fb = coeffs[lower]*interval[0] + coeffs[not lower]*interval[1]
        fb = st.nearest(fb, 0.2*resolution)
        yield fb, min(int((stats+29)/30)*30, 30000)
        lower = not lower

def func(server, args):
    fb, stats = args
    q['biasOperate'] = fb*mV
    q['readout'] = True
    prob = yield runQubits(server, qubits, stats, probs=[1])
    if (prob[0] > target) ^ falling:
        interval[1] = min(fb, interval[1])
    else:
        interval[0] = max(fb, interval[0])
    inc = blowup * (interval[1] - interval[0])
    interval[0] -= inc
    interval[1] += inc
    if noisy:
        print fb, prob[0]
    returnValue([fb, prob[0]])
sweeps.run(func, sweep(), save, dataset, pipesize=2, noisy=False)

fb = 0.5 * (interval[0] + interval[1])*mV
if 'biasStepEdge' in q:
    print 'Old bias_step_edge: %.3f' % Q['biasStepEdge']
print 'New biasStepEdge: %.3f' % fb
if update:
    Q['biasStepEdge'] = fb
return fb
```

Figure A.9: Step edge code page 2 of 2

```
###### From automateDaily.py ########

def bringup_scurve(s, pause=True):
    N = len(s['config'])
    for i in range(N):
        print 'measuring scurve, qubit %d...' % i
        mpa05 = mq.find_mpa(s, measure=i, target=0.05, noisy=False, update=False)
        print '5% tunneling at mpa =', mpa05
        mpa95 = mq.find_mpa(s, measure=i, target=0.95, noisy=False, update=False)
        print '95% tunneling at mpa =', mpa95
        low = st.nearest(mpa05 - (mpa95 - mpa05) * 1.0, 0.002)
        high = st.nearest(mpa95 + (mpa95 - mpa05) * 1.0, 0.002)
        step = 0.002 * np.sign(high - low)
        mpa_range = st.r[low:high:step]
        mq.scurve(s, mpa_range, measure=i, stats=1200, noisy=False, update=pause)
        print 'done.'

    for i in range(N):
        print 'binary searching to find mpa %d...' % i
        mq.find_mpa(s, measure=i, noisy=False, update=True)
        mq.find_mpa_func(s, measure=i, noisy=False, update=True)
        print 'done.'

##### From multiqubit.py ####

def scurve(sample, mpa=st.r[0:2:0.05], stats=300, measure=0,
           save=True, name='SCurve MQ', collect=True, noisy=True, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]

    axes = [(mpa, 'Measure pulse amplitude')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

    def func(server, mpa):
        q['measureAmp'] = mpa
        q.z = eh.measurePulse(q, 0)
        q['readout'] = True
        return runQubits(server, qubits, stats, probs=[1])
    data = sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)

    if update:
        squid.adjust_scurve(Q, data)
    if collect:
        return data

def find_mpa(sample, stats=60, target=0.05, mpa_range=(-2.0, 2.0), pi_pulse=False,
             measure=0, pulseFunc=None, resolution=0.005, blowup=0.05,
             falling=None, statsinc=1.25,
             save=False, name='SCurve Search MQ', collect=True, update=True, noisy=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]
```

Figure A.10: Measurement calibrations code page 1 of 2

```
axes = [('Measure Pulse Amplitude', '')]
dataset = sweeps.prepDataset(sample, name, axes, measure=measure)

if falling is None:
    falling = q['biasOperate'][V] > q['biasStepEdge'][V]
interval = [min(mpa_range), max(mpa_range)]

def sweep(stats=stats):
    mpa = 0.5 * (interval[0] + interval[1])
    yield mpa, min(int((stats+29)/30)*30, 30000)
    lower = True
    coeffs = 0.25, 0.75
    while interval[1] - interval[0] > resolution:
        stats *= statsinc
        mpa = coeffs[lower]*interval[0] + coeffs[not lower]*interval[1]
        mpa = st.nearest(mpa, 0.2*resolution)
        yield mpa, min(int((stats+29)/30)*30, 30000)
        lower = not lower

def func(server, args):
    mpa, stats = args
    q['measureAmp'] = mpa
    if pi_pulse:
        q.xy = eh.mix(q, eh.piPulse(q, 0))
        q.z = eh.measurePulse(q, q['piLen']/2.0)
    else:
        q.xy = env.NOTHING
        q.z = eh.measurePulse(q, 0)
    q['readout'] = True
    probs = yield runQubits(server, qubits, stats, probs=[1])

    prob = probs[0]
    if (prob > target) ^ falling:
        interval[1] = min(mpa, interval[1])
    else:
        interval[0] = max(mpa, interval[0])
    inc = blowup * (interval[1] - interval[0])
    interval[0] -= inc
    interval[1] += inc

    if noisy:
        print mpa, prob
    returnValue([mpa, prob])
sweeps.run(func, sweep(), save, dataset, pipesize=2, noisy=False)

mpa = 0.5 * (interval[0] + interval[1])
key = 'measureAmp' if not pi_pulse else 'measureAmp2'
if key in q:
    print 'Old %s: %.3f' % (key, Q[key])
print 'New %s: %.3f' % (key, mpa)
if update:
    Q[key] = mpa
return mpa
```

Figure A.11: Measurement calibrations code page 2 of 2

```
###### From automateDaily.py ########

def bringup_sample(s, pause=False, fine_tune=True):
    N = len(s['config'])

    bringup_pi_pulses(s, pause=pause)

    if fine_tune:
        for i in range(N):
            # choose frequency range to cover all qubits
            fmin = min(s[qubit]['f10'] for qubit in s['config']) - 0.1*GHz
            fmax = max(s[qubit]['f10'] for qubit in s['config']) + 0.1*GHz

            print 'measuring flux func, qubit %d...' % i,
            mq.find_flux_func(s, (fmin, fmax), measure=i, noisy=False)
            print 'done.'

            print 'measuring zpa func, qubit %d...' % i,
            mq.find_zpa_func(s, (fmin, fmax), measure=i, noisy=False)
            print 'done.'

            # update the calibrated ratio of DAC amplitudes to detuning and rabi freqs
            update_cal_ratios(s)

def bringup_pi_pulses(s, pause=False):
    N = len(s['config'])

    for i in range(N):
        print 'measuring spectroscopy, qubit %d...' % i,
        mq.spectroscopy(s, measure=i, noisy=False, update=pause) # zoom in on resonance peak
        mq.spectroscopy_two_state(s, measure=i, noisy=False, update=pause)
        print 'done.'

    for i in range(N):
        print 'calibrating pi pulse, qubit %d...' % i,
        mq.pitunerHD(s, measure=i, noisy=False)
        print 'done.'

        print 'fine-tuning frequency, qubit %d...' % i,
        mq.freqtuner(s, iterations=1, measure=i, save=True)
        print 'done.'

        print 'redoing pi pulse calibration, qubit %d...' % i,
        mq.pitunerHD(s, measure=i, noisy=False)
        print 'done.'

        print 'checking visibility, qubit %d...' % i
        mpa1_05 = mq.find_mpa(s, measure=i, pi_pulse=True, target=0.05, noisy=False, update=
        False)
        print '5% tunneling of 1 at mpa =', mpa1_05
        mpa0_95 = mq.find_mpa(s, measure=i, pi_pulse=False, target=0.95, noisy=False, update=
        False)
        print '95% tunneling of 0 at mpa =', mpa0_95
```

Figure A.12: Qubit X,Y pulse calibration code page 1 of 10

```
        low = max(st.nearest(mpa1_05 - (mpa0_95 - mpa1_05) * 0.5, 0.002), 0)
        high = min(st.nearest(mpa0_95 + (mpa0_95 - mpa1_05) * 0.5, 0.002), 2)
        step = 0.002 * np.sign(high - low)
        mpa_range = st.r[low:high:step]
        mq.visibility(s, mpa_range, stats=1200, measure=i, noisy=False)
        print 'done.'

        # measure e0, e1 and visibility very carefully at the correct measure-pulse amplitude
        print 'measuring visibility at calibrated mpa %d...' % i,
        Q = s[s['config'][i]]
        data = mq.visibility(s, [Q['measureAmp']]*100, stats=600, measure=i, noisy=False, name=
        'Measurement Fidelity', collect=True)
        e0, f1 = np.mean(data[:,1]), np.mean(data[:,2])
        print 'done.'
        print '  e0: %g, f0: %g' % (e0, 1-e0)
        print '  e1: %g, f1: %g' % (1-f1, f1)
        Q['measureE0'] = e0
        Q['measureF0'] = 1-e0
        Q['measureE1'] = 1-f1
        Q['measureF1'] = f1

##### From multiqubit.py ####

def spectroscopy(sample, freq=None, stats=300L, measure=0, sb_freq=0*GHz, detunings=None,
uwave_amp=None,
                 save=True, name='Spectroscopy MQ', collect=False, noisy=True, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]
    q['readout'] = True
    if freq is None:
        f = st.nearest(q['f10'][GHz], 0.001)
        freq = st.r[f-0.04:f+0.04:0.001, GHz]
    if uwave_amp is None:
        uwave_amp = q['spectroscopyAmp']
    if detunings is None:
        zpas = [0.0] * len(qubits)
    else:
        zpas = []
        for i, (q, df) in enumerate(zip(qubits, detunings)):
            print 'qubit %d will be detuned by %s' % (i, df)
            zpafunc = get_zpa_func(q)
            zpa = zpafunc(q['f10'] + df)
            zpas.append(zpa)

    axes = [(uwave_amp, 'Microwave Amplitude'), (freq, 'Frequency')]
    deps = [('Probability', '|1>', '')]
    kw = {
        'stats': stats,
        'sideband': sb_freq
    }
    dataset = sweeps.prepDataset(sample, name, axes, deps, measure=measure, kw=kw)

    def func(server, amp, f):
```

Figure A.13: Qubit X,Y pulse calibration code page 2 of 10

```
        for i, (q, zpa) in enumerate(zip(qubits, zpas)):
            q['fc'] = f - sb_freq
            if zpa:
                q.z = env.rect(-100, qubits[measure]['spectroscopyLen'] + 100, zpa)
            else:
                q.z = env.NOTHING
            if i == measure:
                q['spectroscopyAmp'] = amp
                q.xy = eh.spectroscopyPulse(q, 0, sb_freq)
                q.z += eh.measurePulse(q, q['spectroscopyLen'])
        eh.correctCrosstalkZ(qubits)
        return runQubits(server, qubits, stats, probs=[1])
    data = sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)
    if update:
        squid.adjust_frequency(Q, data)
    if collect:
        return data

def spectroscopy_two_state(sample, freq=None, stats=300L, measure=0, sb_freq=0*GHz, detunings=
None, uwave_amps=None,
                           save=True, name='Two-state finder spectroscopy MQ', collect=False,
                           noisy=True, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]

    q['readout'] = True
    if freq is None:
        f = st.nearest(q['f10'][GHz], 0.001)
        freq = st.r[f-0.20:f+0.04:0.002, GHz]
    if uwave_amps is None:
        uwave_amps = q['spectroscopyAmp'], q['spectroscopyAmp']*10, q['spectroscopyAmp']*15
    if detunings is None:
        zpas = [0.0] * len(qubits)
    else:
        zpas = []
        for i, (q, df) in enumerate(zip(qubits, detunings)):
            print 'qubit %d will be detuned by %s' % (i, df)
            zpafunc = get_zpa_func(q)
            zpa = zpafunc(q['f10'] + df)
            zpas.append(zpa)

    axes = [(freq, 'Frequency')]
    deps = [('Probability', '|1>, uwa=%g' % amp, '') for amp in uwave_amps]
    kw = {
        'stats': stats,
        'sideband': sb_freq
    }
    dataset = sweeps.prepDataset(sample, name, axes, deps, measure=measure, kw=kw)

    def func(server, f):
        reqs = []
        for amp in uwave_amps:
            for i, (q, zpa) in enumerate(zip(qubits, zpas)):
```

Figure A.14: Qubit X,Y pulse calibration code page 3 of 10

```
                q['fc'] = f - sb_freq
                if zpa:
                    q.z = env.rect(-100, qubits[measure]['spectroscopyLen'] + 100, zpa)
                else:
                    q.z = env.NOTHING
                if i == measure:
                    q['spectroscopyAmp'] = amp
                    q.xy = eh.spectroscopyPulse(q, 0, sb_freq)
                    q.z += eh.measurePulse(q, q['spectroscopyLen'])
            eh.correctCrosstalkZ(qubits)
            reqs.append(runQubits(server, qubits, stats, probs=[1]))
        probs = yield FutureList(reqs)
        returnValue([p[0] for p in probs])
    data = sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)
    if update:
        squid.adjust_frequency_02(Q, data)
    if collect:
        return data

def pitunerHD(sample, measure=0, iterations=2, npoints=21, stats=1200, save=False, update=True,
noisy=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]
    amp = q['piAmp']
    for _ in xrange(iterations):
        # optimize amplitude
        data = rabihigh_hd(sample, amplitude=np.linspace(0.6*amp, 1.4*amp, npoints),
                           measure=measure, stats=stats, collect=True, noisy=noisy)
        amp_fit = np.polyfit(data[:,0], data[:,1], 2)
        amp = -0.5 * amp_fit[1] / amp_fit[0]
        print 'Amplitude: %g' % amp
    # save updated values
    if update:
        Q['piAmp'] = amp
    return amp

def rabihigh_hd(sample, amplitude=st.r[0.0:1.5:0.05], measureDelay=None, measure=0, stats=1500L,
                name='Rabi-pulse height HD MQ', save=True, collect=False, noisy=True):
    sample, qubits = util.loadQubits(sample)
    q = qubits[measure]

    if amplitude is None: amplitude = q['piAmp']
    if measureDelay is None: measureDelay = q['piLen'] # /2.0

    axes = [(amplitude, 'pulse height'),
            (measureDelay, 'measure delay')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

    def func(server, amp, delay):
        q['piAmp'] = amp
        q.xy = eh.mix(q, eh.piPulseHD(q, 0))
        q.z = eh.measurePulse(q, delay)
```

Figure A.15: Qubit X,Y pulse calibration code page 4 of 10

```
        q['readout'] = True
        return runQubits(server, qubits, stats, probs=[1])
    return sweeps.grid(func, axes, dataset=save and dataset, collect=collect, noisy=noisy)

def freqtuner(sample, iterations=1, tEnd=100*ns, timeRes=1*ns, nfftpoints=4000, stats=1200, df=
50*MHz,
              measure=0, save=False, plot=False, noisy=True, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]
    #Automatically finds best f10, using ramsey can plot the FFT of the Ramsey fringe to
    extract true f10
    # works for sweeps with time steps that are all equivalent (i.e. not concatenated sweeps
    with diff time steps)

    # Time resolution should be at least at the Nyquist frequency, but better to oversample
    # nyfreq=float(fringeFreq)*2*10e6
    # timeRes = (1.0/float(nyfreq))*1e9
    if plot:
        fig = plt.figure()
    for i in xrange(iterations):
        data = ramsey(sample, measure=measure, delay=st.r[0:tEnd:timeRes,ns], fringeFreq = df,
           stats=stats, name='Ramsey Freq Tuner MQ', save = save, noisy=noisy,
           collect = True, randomize=False, averages = 1, tomo=False)
        ts, ps = data.T
        y = ps - np.polyval(np.polyfit(ts, ps, 1), ts) # detrend
        timestep = ts[1] - ts[0]
        freq = np.fft.fftfreq(nfftpoints, timestep)
        fourier = abs(np.fft.fft(y, nfftpoints))
        fringe = abs(freq[np.argmax(fourier)])*1e3*MHz
        delta_freq = df - fringe
        if plot:
            ax = fig.add_subplot(iterations,1,i)
            ax.plot(np.fft.fftshift(freq), np.fft.fftshift(fourier))
        print 'Desired Fringe Frequency: %s' % df
        print 'Actual Fringe Frequency: %s' % fringe
        print 'Qubit frequency adjusted by %s' % delta_freq

        q['f10'] -= delta_freq
        print 'new resonance frequency: %g' % q['f10']
    if update:
        Q['f10'] = st.nearest(q['f10'][GHz], 0.0001)*GHz

    return Q['f10']

def visibility(sample, mpa=st.r[0:2:0.05], stats=300, measure=0, level=1,
               save=True, name='Visibility MQ', collect=True, update=False, noisy=True):
    sample, qubits = util.loadQubits(sample)
    q = qubits[measure]

    axes = [(mpa, 'Measure pulse amplitude')]
    if level==1:
        deps = [('Probability', '|0>', ''),
                ('Probability', '|1>', ''),
```

Figure A.16: Qubit X,Y pulse calibration code page 5 of 10

```
                ('Visibility', '|1> - |0>', ''),
                ]
    elif level==2:
        deps = [('Probability', '|0>', ''),
                ('Probability', '|1>', ''),
                ('Visibility', '|1> - |0>', ''),
                ('Probability', '|2>', ''),
                ('Visibility', '|2> - |1>', '')
                ]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, deps, measure=measure, kw=kw)

    def func(server, mpa):
        t_pi = 0
        t_meas = q['piLen']/2.0

        # without pi-pulse
        q['readout'] = True
        q['measureAmp'] = mpa
        q.xy = env.NOTHING
        q.z = eh.measurePulse(q, t_meas)
        req0 = runQubits(server, qubits, stats, probs=[1])

        # with pi-pulse
        q['readout'] = True
        q['measureAmp'] = mpa
        q.xy = eh.mix(q, eh.piPulseHD(q, t_pi))
        q.z = eh.measurePulse(q, t_meas)
        req1 = runQubits(server, qubits, stats, probs=[1])

        if level == 2:
            # |2> with pi-pulse
            q['readout'] = True
            q['measureAmp'] = mpa
            q.xy = eh.mix(q, eh.piPulseHD(q, t_pi-q.piLen))+eh.mix(q, env.gaussian(t_pi, q.
            piFWHM, q.piAmp21, df=q.piDf21), freq = 'f21')
            q.z = eh.measurePulse(q, t_meas)
            req2 = runQubits(server, qubits, stats, probs=[1])

            probs = yield FutureList([req0, req1, req2])
            p0, p1, p2 = [p[0] for p in probs]

            returnValue([p0, p1, p1-p0, p2, p2-p1])
        elif level == 1:
            probs = yield FutureList([req0, req1])
            p0, p1 = [p[0] for p in probs]

            returnValue([p0, p1, p1-p0])

    return sweeps.grid(func, axes, dataset=save and dataset, collect=collect, noisy=noisy)


def find_flux_func(sample, freqScan=None, measAmplFunc=None, measure=0,
```

Figure A.17: Qubit X,Y pulse calibration code page 6 of 10

```
                fluxBelow=2*mV, fluxAbove=2*mV, fluxStep=0.1*mV, sb_freq=0*GHz, stats=300L,
                save=True, name='Flux func search MQ', collect=False, update=True, noisy=True
                ):
sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
qubit, Qubit = qubits[measure], Qubits[measure]

if measAmplFunc is None:
    measAmplFunc = get_mpa_func(qubit)

if freqScan is None:
    freq = st.nearest(qubit['f10'][GHz], 0.001)
    dfs = np.logspace(-3, 0, 25)
    freqScan = freq + np.hstack(([0], dfs, -dfs))
elif isinstance(freqScan, tuple) and len(freqScan) == 2:
    freq = st.nearest(qubit['f10'][GHz], 0.001)
    rng = freqScan
    dfs = np.logspace(-3, 0, 25)
    freqScan = freq + np.hstack(([0], dfs, -dfs))
    freqScan = np.array([st.nearest(f, 0.001) for f in freqScan])
    freqScan = np.unique(freqScan)
    freqScan = np.compress((rng[0][GHz] < freqScan) * (freqScan < rng[1][GHz]), freqScan)
else:
    freqScan = np.array([f[GHz] for f in freqScan])
freqScan = freqScan[np.argsort(abs(freqScan-qubit['f10'][GHz]))]

fluxBelow = fluxBelow[V]
fluxAbove = fluxAbove[V]
fluxStep = fluxStep[V]
fluxScan = np.arange(-fluxBelow, fluxAbove, fluxStep)
fluxScan = fluxScan[np.argsort(abs(fluxScan))]
fluxPoints = len(fluxScan)
step_edge = qubit['biasStepEdge'][V]

sweepData = {
    'fluxFunc': np.array([st.nearest(qubit['biasOperate'][V], fluxStep) - step_edge]),
    'fluxIndex': 0,
    'freqIndex': 0,
    'flux': 0*fluxScan,
    'prob': 0*fluxScan,
    'maxima': 0*freqScan,
}

axes = [('Flux Bias', 'V'), ('Frequency', 'GHz')]
kw = {'stats': stats}
dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

def sweep():
    for f in freqScan:
        center = np.polyval(sweepData['fluxFunc'], f**4) + step_edge
        center = st.nearest(center, fluxStep)
        for flx in center + fluxScan:
            yield flx*V, f*GHz
```

Figure A.18: Qubit X,Y pulse calibration code page 7 of 10

```
    def func(server, args):
        flux, freq = args
        for q in qubits:
            q['fc'] = freq - sb_freq # set all frequencies since they share a common microwave
            source
        qubit['biasOperate'] = flux
        qubit['measureAmp'] = measAmplFunc(flux)
        qubit.xy = eh.spectroscopyPulse(qubit, 0, sb_freq)
        qubit.z = eh.measurePulse(qubit, qubit['spectroscopyLen'] + qubit['piLen'])
        qubit['readout'] = True
        prob = yield runQubits(server, qubits, stats, probs=[1])

        flux_idx = sweepData['fluxIndex']
        sweepData['flux'][flux_idx] = flux[V]
        sweepData['prob'][flux_idx] = prob[0]
        if flux_idx + 1 == fluxPoints:
            # one row is done.  find the maximum and update the spectroscopy fit
            freq_idx = sweepData['freqIndex']
            sweepData['maxima'][freq_idx] = sweepData['flux'][np.argmax(sweepData['prob'])]
            sweepData['fluxFunc'] = np.polyfit(freqScan[:freq_idx+1]**4,
                                               sweepData['maxima'][:freq_idx+1] - step_edge,
                                               (freq_idx > 5))
            sweepData['fluxIndex'] = 0
            sweepData['freqIndex'] += 1
        else:
            # just go to the next point
            sweepData['fluxIndex'] = flux_idx + 1
        returnValue([flux, freq, prob])
    sweeps.run(func, sweep(), dataset=save and dataset, collect=collect, noisy=noisy)

    # create a flux function and return it
    p = sweepData['fluxFunc']
    if update:
        Qubit['calFluxFunc'] = p
    return get_flux_func(Qubit, p, step_edge*V)


def get_flux_func(qubit, p=None, step_edge=None):
    if p is None:
        p = qubit['calFluxFunc']
    if step_edge is None:
        step_edge = qubit['biasStepEdge']
    return lambda f: np.polyval(p, f[GHz]**4)*V + step_edge


def find_zpa_func(sample, freqScan=None, measure=0,
                  fluxBelow=0.01, fluxAbove=0.01, fluxStep=0.0005, sb_freq=0*GHz, stats=300L,
                  name='ZPA func search MQ', save=True, collect=False, noisy=True, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    qubit = qubits[measure]

    if freqScan is None:
        freq = st.nearest(qubit['f10'][GHz], 0.001)
```

Figure A.19: Qubit X,Y pulse calibration code page 8 of 10

```
    freqScan = np.arange(freq-0.1, freq+1.0, 0.005)
elif isinstance(freqScan, tuple) and len(freqScan) == 2:
    freq = st.nearest(qubit['f10'][GHz], 0.001)
    rng = freqScan
    dfs = np.logspace(-3, 0, 25)
    freqScan = freq + np.hstack(([0], dfs, -dfs))
    freqScan = np.array([st.nearest(f, 0.001) for f in freqScan])
    freqScan = np.unique(freqScan)
    freqScan = np.compress((rng[0][GHz] < freqScan) * (freqScan < rng[1][GHz]), freqScan)
else:
    freqScan = np.array([f[GHz] for f in freqScan])
freqScan = freqScan[np.argsort(abs(freqScan-qubit['f10'][GHz]))]

fluxScan = np.arange(-fluxBelow, fluxAbove, fluxStep)
fluxScan = fluxScan[np.argsort(abs(fluxScan))]
fluxPoints = len(fluxScan)

sweepData = {
    'fluxFunc': np.array([0]),
    'fluxIndex': 0,
    'freqIndex': 0,
    'flux': 0*fluxScan,
    'prob': 0*fluxScan,
    'maxima': 0*freqScan,
}

axes = [('Z-pulse amplitude', ''), ('Frequency', 'GHz')]
kw = {'stats': stats}
dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

def sweep():
    for f in freqScan:
        center = np.polyval(sweepData['fluxFunc'], f**4)
        center = st.nearest(center, fluxStep)
        for zpa in center + fluxScan:
            yield zpa, f*GHz

def func(server, args):
    zpa, freq = args
    for q in qubits:
        q['fc'] = freq - sb_freq # set all frequencies since they share a common microwave
        source
    dt = qubit['spectroscopyLen']
    qubit.xy = eh.spectroscopyPulse(qubit, 0, sb_freq)
    qubit.z = env.rect(0, dt, zpa) + eh.measurePulse(qubit, dt)
    qubit['readout'] = True
    prob = yield runQubits(server, qubits, stats, probs=[1])

    flux_idx = sweepData['fluxIndex']
    sweepData['flux'][flux_idx] = zpa
    sweepData['prob'][flux_idx] = prob[0]
    if flux_idx + 1 == fluxPoints:
        # one row is done.  find the maximum and update the spectroscopy fit
```

Figure A.20: Qubit X,Y pulse calibration code page 9 of 10

```
            freq_idx = sweepData['freqIndex']
            sweepData['maxima'][freq_idx] = sweepData['flux'][np.argmax(sweepData['prob'])]
            sweepData['fluxFunc'] = np.polyfit(freqScan[:freq_idx+1]**4,
                                               sweepData['maxima'][:freq_idx+1],
                                               freq_idx > 5)
            sweepData['fluxIndex'] = 0
            sweepData['freqIndex'] += 1
        else:
            # just go to the next point
            sweepData['fluxIndex'] = flux_idx + 1
        returnValue([zpa, freq, prob])
    sweeps.run(func, sweep(), dataset=save and dataset, collect=collect, noisy=noisy)

    # create a flux function and return it
    poly = sweepData['fluxFunc']
    if update:
        Qubits[measure]['calZpaFunc'] = poly
    return get_zpa_func(Qubits[measure], poly)

###### From automateDaily.py ########

def update_cal_ratios(s):
    s, _qubits, Qubits = util.loadQubits(s, write_access=True)

    # single-qubit bringup
    for Q in Qubits:
        # convert microwave amplitude to rabi frequency
        fwhm = Q['piFWHM'][ns]
        A = float(Q['piAmp'])
        Q['calRabiOverUwa'] = 2*np.sqrt(np.log(2)/np.pi)/(A*fwhm)*GHz # total area is 1 cycle

        # convert z amplitude to detuning frequency
        a = float(Q['calZpaFunc'][0])
        f = Q['f10'][GHz]
        Q['calDfOverZpa'] = 1/(4*a*f**3)*GHz
```

Figure A.21: Qubit X,Y pulse calibration code page 10 of 10

```
###### From automateDaily.py ########

def single_qubit_scans(s):
    N = len(s['config'])
    for i in range(N):
        print 'measuring T1, qubit %d' % i,
        mq.t1(s, stats=1800, measure=i, noisy=False)
        #TODO add T1 fits
        print 'done.'

        print 'measuring ramsey fringe, qubit %d' % i,
        #TODO bring T1 fit from above and turn on T2 fit
        mq.ramsey(s, stats=1800, measure=i, noisy=False)
        print 'done.'

        print 'measuring spin_echo, qubit %d' % i,
        mq.spinEcho(s, stats=1800, measure=i, noisy=False)
        print 'done.'

##### From multiqubit.py ####

def t1(sample, delay=st.r[-10:1000:2,ns], stats=600L, measure=0,
       name='T1 MQ', save=True, collect=True, noisy=True):
    """A single pi-pulse on one qubit, with other qubits also operated."""
    sample, qubits = util.loadQubits(sample)
    q = qubits[measure]

    axes = [(delay, 'Measure pulse delay')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

    def func(server, delay):
        q.xy = eh.mix(q, eh.piPulse(q, 0))
        q.z = eh.measurePulse(q, delay)
        q['readout'] = True
        return runQubits(server, qubits, stats, probs=[1])
    return sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)

def ramsey(sample, measure=0, delay=st.r[0:500:1,ns], fringeFreq = 50*MHz,
           stats=300L, name='Ramsey MQ', save = True, noisy=True,
           collect = False, randomize=False, averages = 1, tomo=True, fitPlot=False, t1=None,
           tRange=None):
    """Ramsey sequence on one qubit. Can be single phase or 4-axis tomo, and
    can have randomized time axis and/or averaging over the time axis multiple
    times

    PARAMETERS
    sample: object defining qubits to measure, loaded from registry
    measure - scalar: number of qubit to measure. Only one qubit allowed.
    delay - iterable: time axis
    fringeFreq - value [Mhz]: Desired frequency of Ramsey fringes
    stats - scalar: number of times a point will be measured per iteration over
            the time axis. That the actual number of times a point will be
```

Figure A.22: Code for single qubit scans page 1 of 4

```
        measured is stats*averages
name - string: Name of dataset.
save - bool: Whether or not to save data to the datavault
noisy - bool: Whether or not to print out probabilities while the scan runs
collect - bool: Whether or not to return data to the local scope.
randomize - bool: Whether or not to randomize the time axis.
averages - scalar: Number of times to iterate over the time axis.
tomo - bool: Set True if you want to measure all four tomo axes, False if
             you only want the X axis (normal Ramsey fringes).
fitPlot - plots and returns fit for T2. Requires a value for T1, so for now you need to be
around to input the value.
t1 - T1 for the qubit. You should have this from a fit and then you can just enter it e.g.
t1=400.0*ns
tRange - the time range you want to use to fit T2. Enter as a tuple (0,100).
"""
sample, qubits = util.loadQubits(sample)
q = qubits[measure]
q['readout'] = True

#Randomize time axis
if fitPlot:
    print 'will make pretty plots, just you wait and see!'
if randomize:
    delay = st.shuffle(delay)
#Generator that produces time delays. Iterates over the list of delays as many times as
specified by averages.
def delay_gen():
    for _ in range(averages):
        for d in delay:
            yield d

axes = [(delay_gen(), 'Delay')]
#If you want XY state tomography then we use all four pi/2 pulse phases
if tomo:
    deps = [('Probability', '+X', ''),('Probability', '+Y', ''),
            ('Probability', '-X', ''),('Probability', '-Y', '')]
    tomoPhases = {'+X': 0.0, '+Y': 0.25, '-X': -0.5, '-Y': -0.25} #[+X, +Y, -X, -Y] in CYCLES
#Otherwise we only do a final pi/2 pulse about the +X axis.
else:
    deps = [('Probability', '', '')]
    tomoPhases = {'+X': 0.0}

kw = {'averages': averages, 'stats': stats, 'fringeFrequency': fringeFreq}
dataset = sweeps.prepDataset(sample, name, axes, deps, measure=measure, kw=kw)

#Pump pulse is at time=0 with phase=0
pump = eh.piHalfPulse(q, 0, phase=0.0)
#Probe is at variable time with variable phase
def probe(time, tomoPhase):
    return eh.piHalfPulse(q, time, phase = 2*np.pi*(fringeFreq['GHz']*time['ns']+tomoPhases[
    tomoPhase]))

def func(server, delay):
```

Figure A.23: Code for single qubit scans page 2 of 4

```
            reqs = []
            for tomoPhase in tomoPhases.keys():
                print delay
                q.xy = eh.mix(q, pump + probe(delay, tomoPhase = tomoPhase))
                q.z = eh.measurePulse(q,delay+20*ns)
                reqs.append(runQubits(server, qubits, stats, probs=[1]))
            probs = yield FutureList(reqs)
            data = [p[0] for p in probs]
            returnValue(data)
        result = sweeps.grid(func, axes, dataset = save and dataset, collect=collect, noisy=noisy)

        if fitPlot:
            print 'Making pretty plots'
            with labrad.connect() as cxn:
                ds = cxn.data_vault
                dataset = pyle.plotting.dstools.getOneDeviceDataset(ds, datasetNumber=None, session=
                sample._dir, deviceName=None,
                           averaged=averages>1)
                pyle.fitting.dephasing.ramseyTomo(dataset, T1=t1, timeRange=tRange)
        return result

    def spinEcho(sample, measure=0, delay=st.r[0:1000:10,ns], df=50*MHz,
                       stats=300L, name='Spin Echo MQ', save=True,
                       collect=True, noisy=True, randomize=False, averages=1,
                       tomo=True):
        """Spin echo sequence on one qubit. Can be single phase or 4-axis tomo, and
        can have randomized time axis and/or averaging over the time axis multiple
        times

        PARAMETERS
        sample: object defining qubits to measure, loaded from registry
        measure - scalar: number of qubit to measure. Only one qubit allowed.
        delay - iterable: time axis
        fringeFreq - value [Mhz]: Desired frequency of Ramsey fringes
        stats - scalar: number of times a point will be measured per iteration over
                the time axis. That the actual number of times a point will be
                measured is stats*averages
        name - string: Name of dataset.
        save - bool: Whether or not to save data to the datavault
        noisy - bool: Whether or not to print out probabilities while the scan runs
        collect - bool: Whether or not to return data to the local scope.
        randomize - bool: Whether or not to randomize the time axis.
        averages - scalar: Number of times to iterate over the time axis.
        tomo - bool: Set True if you want to measure all four tomo axes, False if
                     you only want the X axis (normal Ramsey fringes).
        """

        sample, qubits = util.loadQubits(sample)
        q = qubits[measure]
        q['readout']=True

        #Randomize time axis
        if randomize:
```

Figure A.24: Code for single qubit scans page 3 of 4

```
    delay=st.shuffle(delay)
def delayGen():
    for _ in range(averages):
        for d in delay:
            yield d

axes = [(delayGen(), 'Delay')]
if tomo:
    deps = [('Probability', '+X', ''), ('Probability', '+Y', ''),
            ('Probability', '-X', ''), ('Probability', '-Y', '')]
    tomoPhases = {'+X': 0.0, '+Y':0.25, '-X': -0.5, '-Y':-0.25}
else:
    deps = [('Probability','','')]
    tomoPhases={'+X':0.0}

kw={'averages':averages, 'stats':stats, 'fringeFrequency':df}
dataset=sweeps.prepDataset(sample, name, axes, deps, measure=measure, kw=kw)

#Pump pulse is at time=0 with phase=0
pump = eh.piHalfPulse(q, 0, phase=0.0)
#Probe is at variable time with variable phase
def probe(time, tomoPhase):
    return eh.piHalfPulse(q, time, phase=2.0*np.pi*tomoPhases[tomoPhase])

def func(server, delay):
    reqs=[]
    dt=q['piLen']
    tpi = dt/2.0 + delay/2.0
    tProbe = dt/2.0 + delay + dt/2.0
    tMeas = tProbe + dt/2.0
    piPhase = 2*np.pi*df[GHz]*delay[ns]/2.0
    for tomoPhase in tomoPhases.keys():
        q.xy = eh.mix(q, pump +
                      eh.piPulse(q, tpi, phase=piPhase) +
                      probe(tProbe, tomoPhase))
        q.z = eh.measurePulse(q, tMeas)
        reqs.append(runQubits(server, qubits, stats, probs=[1]))
    probs = yield FutureList(reqs)
    data = [p[0] for p in probs]
    returnValue(data)
return sweeps.grid(func, axes, dataset=save and dataset, collect=collect, noisy=noisy)
```

Figure A.25: Code for single qubit scans page 4 of 4

```
###### From automateDaily.py ########

def qubit_coupling_resonator_scans(s):
    start = datetime.now()

    N = len(s['config'])
    for i in range(N):
        print 'measuring SWAP10 Spectroscopy, qubit %d' % i,
        mq.swap10tuner(s, measure=i, stats=1800, noisy=False, whichRes='Coupler')
        print 'measuring 2D-SWAP Spec around Coupling resonator, for qubit %d' % i,
        mq.swap10(s, swapLen=st.arangePQ(0,75,2,ns), swapAmp=None, measure=i, save=True, noisy=
        False, swapAmpBND=0.2, swapAmpSteps=0.001)
        #run focktuner level =1
        print 'fock tuner for fine calibratin of cZControlLen'
        mq.fockTuner(s, n=1, iteration=3, tuneOS=False, stats=1800, measure=i, save=True, noisy=
        False)
        print 'done. Calibrated Control qubits'

        print 'Tuning up pi-pulse for |2> of qubit %d' % i,
        mq.pituner21(s, stats = 1800, measure=i, noisy=False, findMPA=True)
        print 'done'

        print 'measuring SWAP21 Spectroscopy'
        mq.swap21tuner(s, measure=i, stats=1800, noisy=False)
        print 'measuring 2D-SWAP Spec around resonator, for qubit %d' % i,
        mq.swap21(s, swapLen=st.arangePQ(0,60,2,ns), swapAmp=None, measure=i, save=True, noisy=
        False, swapAmpBND=0.2, swapAmpSteps=0.001)
        mq.fockTuners21(s, n=1, iteration=3, tuneOS=False, stats=1800, measure=i, save=True,
        noisy=False)
        print 'done. Calibrated Target qubits'

    print 'now starting qubit-qubit timing calibrations...'
    print 'measuring qubit-qubit delay via the resonator'
    for j,k in [(0,1),(1,0), (0,2),(2,0), (1,2),(2,1), (0,3),(3,0), (1,3),(3,1), (2,3),(3,2)]:
        mq.testQubResDelayCmp(s,measureC=j, measureT=k)
    print 'now measuring resonator T1 using q0 for photon exchange'
    mq.resonatorT1(s, stats=1800, measure=0, whichRes='Coupler')
    end = datetime.now()
    print 'start:', start
    print 'end:', end
    print 'elapsed time for qubit-resonator scans:', (end-start)

def qubit_memory_resonator_scans(s, stats=1800):
    start = datetime.now()

    N = len(s['config'])
    for i in range(N):
        print 'measuring SWAP10 Spectroscopy, qubit %d' % i,
        mq.swap10tuner(s, measure=i, stats=stats, noisy=False, whichRes='Memory')

        print 'measuring 2D-SWAP Spec around Memory resonator, for qubit %d' % i,
        mq.swap10(s, swapLen=st.arangePQ(0,300,5,ns), swapAmp=None, measure=i,
                  save=True, noisy=False, swapAmpBND=0.2, swapAmpSteps=0.001, stats=stats,
```

Figure A.26: Code for qubit-resonator calibrations page 1 of 8

```
                    whichRes='Memory')
        #run focktuner level =1
        print 'fock tuner for fine calibratin of memoryReadWriteLen'
        mq.fockTuner(s, n=1, iteration=3, tuneOS=False, stats=stats, measure=i, save=True, noisy
        =False, whichRes='Memory')
        print 'done. Memory resonator tuned up'
        print 'now measuring memory resonator T1 for resonator %d' %i,
        noon.resonatorT1(s, stats=stats, measure=i, whichRes='Memory')

    end = datetime.now()
    print 'start:', start
    print 'end:', end
    print 'elapsed time for qubit-mem-resonator scans:', (end-start)


##### From multiqubit.py ####

def swap10tuner(sample, swapLen=None, swapAmp=None, swapAmpBND=0.01, iteration=3, measure=0,
stats=600L,
         name='Qeg-R10 swap tuner MQ', save=False, noisy=True, update=True, whichRes='Coupler'):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]

    if whichRes is 'Coupler':
        if swapAmp is None:
            swapAmp = q['cZControlAmp']
        if swapLen is None:
            swapLen = q['cZControlLen'][ns]
    elif whichRes is 'Memory':
        if swapAmp is None:
            swapAmp = q['memReadWriteAmp']
        if swapLen is None:
            swapLen = q['memReadWriteLen'][ns]

    for i in range(iteration):
        rf = 2**i
        swapLenOld = swapLen
        swapAmpOld = swapAmp
        print 'Tuning the swap length'
        results = swap10(sample, swapLen=st.PQlinspace(swapLen*(1-0.3/rf),swapLen*(1+0.3/rf),21,
        ns),
                         swapAmp=swapAmp, measure=measure, stats=stats,
                         name='Qeg-R10 swap MQ', save=save, collect=True, noisy=noisy)
        newLen, percent = datasetMinimum(results, swapLenOld, swapLenOld-4/rf, swapLenOld+4/rf)
        swapLen = newLen
        if whichRes is 'Coupler':
            print 'Old swap length was ',q['cZControlLen']
            if update:
                Q['cZControlLen'] = swapLen*ns
                print 'New Control swap length is ', swapLen, 'ns'
        else:
            print 'Old Memory Read/Write length was ',q['memReadWriteLen']
            if update:
```

Figure A.27: Code for qubit-resonator calibrations page 2 of 8

```
                Q['memReadWriteLen'] = swapLen*ns
                print 'New Memory Read/Write length is ', swapLen, 'ns'

        print 'Tuning the swap amplitude'
        results = swap10(sample, swapLen=swapLen,
                         swapAmp=np.linspace(max([swapAmp*(1-0.3/rf),swapAmp-swapAmpBND]),
                                 min([swapAmp*(1+0.3/rf),swapAmp+swapAmpBND]),21), measure=
                                 measure, stats=stats,
                         name='Qeg-R10 swap MQ', save=save, collect=True, noisy=noisy)
        newAmp, percent = datasetMinimum(results, swapAmpOld, swapAmpOld-4/rf, swapAmpOld+4/rf)
        swapAmp = newAmp
        if whichRes is 'Coupler':
            print 'Old Control swap amplitude was ',q['cZControlAmp']
            if update:
                Q['cZControlAmp']= swapAmp
                print 'New swap amplitude is ', swapAmp
        else:
            print 'Old Memory Read/Write amplitude was ',q['memReadWriteAmp']
            if update:
                Q['memReadWriteAmp']= swapAmp
                print 'New Read/Write amplitude is ', swapAmp

    return swapLen, swapAmp

def swap10(sample, swapLen=st.arangePQ(0,200,4,ns), swapAmp=np.arange(-0.05,0.05,0.002), measure
=0, stats=600L,
         name='Qeg-R10 swap MQ', save=True, collect=False, noisy=True, swapAmpBND=0.20,
         swapAmpSteps=0.001, whichRes='Coupler'):
    sample, qubits = util.loadQubits(sample)
    q = qubits[measure]
    if whichRes is 'Coupler':
        if swapAmp is None:
            swapAmp   = q.cZControlAmp
            coarseSet = np.arange(0,swapAmp*(1-swapAmpBND),swapAmpSteps*5)
            fineSet   = np.arange(swapAmp*(1-swapAmpBND),swapAmp*(1+swapAmpBND), swapAmpSteps)
            swapAmp   = np.hstack((coarseSet,fineSet))
            #swapAmp   = st.r[swapAmp*(1-swapAmpBND):swapAmp*(1+swapAmpBND):swapAmpSteps]
    elif whichRes is 'Memory':
        if swapAmp is None:
            swapAmp   = q.memReadWriteAmp
            fineSet   = np.arange(swapAmp*(1+swapAmpBND),swapAmp*(1-swapAmpBND), swapAmpSteps)
            swapAmp   = fineSet

    axes = [(swapAmp, 'swap pulse amplitude'), (swapLen, 'swap pulse length')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

    def func(server, currAmp, currLen):
        q.xy = eh.mix(q, eh.piPulseHD(q, 0))
        q.z = env.rect(q['piLen']/2, currLen, currAmp) + eh.measurePulse(q, q['piLen']/2 +
        currLen)
        q['readout'] = True
        return runQubits(server, qubits, stats=stats, probs=[1])
```

Figure A.28: Code for qubit-resonator calibrations page 3 of 8

```
    return sweeps.grid(func, axes, save=save, dataset=dataset, collect=collect, noisy=noisy)

def fockTuner(sample, n=1, iteration=3, tuneOS=False,stats=1500L, measure=0, delay=0*ns,
       save=False, collect=True, noisy=True, update=True, whichRes='Coupler'):
    sample, qubits, Qubits = util.loadQubits(sample,write_access=True)
    q = qubits[measure]
    Q = Qubits[measure]

    for iter in range(iteration):
        rf = 2**iter
        print 'iteration %g...' % iter
        if whichRes is 'Coupler':
            sl = q['cZControlLen']
        else:
            sl = q['memReadWriteLen']
        results = fockScan(sample, n=1, scanLen=st.PQlinspace(-max([0.3*sl['ns']/rf,1]),max([0.3
        *sl['ns']/rf,1]),21,'ns'),
                                stats=stats,measure=measure,probeFlag=False,delay=delay,
                                save=False, collect=collect, noisy=noisy, whichRes=whichRes)
        newLen, percent = datasetMinimum(results, 0, -max([0.3*sl['ns']/rf,1]), max([0.3*sl['ns'
        ]/rf,1]))
        if whichRes is 'Coupler':
            q['cZControlLen'] += newLen
        else:
            q['memReadWriteLen'] += newLen

        if save:
            fockScan(sample, n=1, scanLen=st.arangePQ(0,300,2,'ns'),
                        stats=stats,measure=measure,probeFlag=True,delay=delay,
                        save=save, collect=collect, noisy=noisy, whichRes=whichRes)
    if update:
        if whichRes is 'Coupler':
            Q['cZControlLen'] = q['cZControlLen']
        else:
            Q['memReadWriteLen'] = q['memReadWriteLen']

    return newLen

def pituner21(sample, measure=0, iterations=2, npoints=21, stats=1500L, save=False, update=True,
 noisy=True, findMPA=True):

    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]
    amp = q.piAmp21
    df = q.piDf21['MHz']
    if findMPA:
        print 'finding measure pulse amplitude for |2>'
        Q.measureAmp2 = find_mpa(sample, stats=600, target=0.05, mpa_range=(-2.0, 2.0), pi_pulse
        =True,
                 measure=measure, pulseFunc=None, resolution=0.005, blowup=0.05,
                 falling=None, statsinc=1.25,
                 save=False, name='SCurve Search for best |2> MPA MQ', collect=True, update=True
                 , noisy=True)
```

Figure A.29: Code for qubit-resonator calibrations page 4 of 8

```
    for _ in xrange(iterations):
        # optimize amplitude
        data = rabihigh21(sample, amplitude=np.linspace(0.75*amp, 1.25*amp, npoints), detuning=
        df*MHz,
                          measure=measure, stats=stats, collect=True, noisy=noisy, save=save)
        amp_fit = np.polyfit(data[:,0], data[:,1], 2)
        amp = -0.5 * amp_fit[1] / amp_fit[0]
        print 'Amplitude for 1->2 transition: %g' % amp
        # optimize detuning
        data = rabihigh21(sample, amplitude=amp, detuning=st.PQlinspace(df-20, df+20, npoints,
        MHz),
                          measure=measure, stats=stats, collect=True, noisy=noisy, save=save)
        df_fit = np.polyfit(data[:,0], data[:,1], 2)
        Delta_df = -0.5 * df_fit[1] / df_fit[0]-df
        if np.abs(Delta_df)>20:
            df += np.sign(Delta_df)*20
        else:
            df += Delta_df
        print 'Detuning frequency for 1->2 transition: %g MHz' % df
    # save updated values
    if update:
        Q['piAmp21'] = amp
        Q['piDf21'] = df*MHz
    return amp, df*MHz

def swap21tuner(sample, swapLen=None, swapAmp=None, iteration=3, measure=0, stats=600L,
         name='Qfe-R10 swap tuner MQ', save=False, noisy=True, update=True, zGate='Pi'):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q, Q = qubits[measure], Qubits[measure]

    if zGate is 'Pi':
        if swapAmp is None:
            swapAmp = q['cZTargetAmp']
        if swapLen is None:
            swapLen = q['cZTargetLen'][ns]
            swapLen = swapLen/2 #cZTargetLen is the time for a iswap^2 so we divide by two for
            this iSWAP experiment.
    elif zGate is 'HalfPi':
        if swapAmp is None:
            swapAmp = q['cPiHalfTargetAmp']
        if swapLen is None:
            swapLen = q['cPiHalfTargetLen'][ns]
            swapLen = swapLen/2 #cZTargetLen is the time for a iswap^2 so we divide by two for
            this iSWAP experiment.

    for m in range(iteration):
        rf = 2**m #not sure what m is stepping over, I think it is a dummy variable, but used
        in the linspace calc"
        swapLenOld = swapLen
        swapAmpOld = swapAmp
        print 'Tuning the swap length'
        results = swap21(sample, swapLen=st.PQlinspace(swapLen*(1-0.3/rf),swapLen*(1+0.3/rf),21,
        ns),
```

Figure A.30: Code for qubit-resonator calibrations page 5 of 8

```
                        swapAmp=swapAmp, measure=measure, stats=stats,
                        name='Qfe-R10 swap MQ', save=save, collect=True, noisy=noisy)
        newLen, percent = datasetMinimum(results, swapLenOld, swapLenOld-4/rf, swapLenOld+4/rf)
        swapLen=newLen
        if zGate is 'Pi':
            print 'Old swap length was ',q['cZTargetLen'], 'ns'
            if update:
                Q['cZTargetLen'] = (2*swapLen)*ns #cZControlLen is the time for a iswap^2.
                print 'New Target swap length is ', 2*swapLen, 'ns'

        elif zGate is 'HalfPi':
            print 'Old swap length was ',q['cPiHalfTargetLen'], 'ns'
            if update:
                Q['cPiHalfTargetLen'] = (2*swapLen)*ns #cZControlLen is the time for a iswap^2.
                print 'New Half-Pi Target swap length is ', 2*swapLen, 'ns'
        print 'Tuning the swap amplitude'
        swapLen = newLen
        results = swap21(sample, swapLen=swapLen,
                        swapAmp=np.linspace(swapAmp*(1-0.3/rf),swapAmp*(1+0.3/rf),21), measure=
                        measure, stats=stats,
                        name='Qfe-R10 swap MQ', save=save, collect=True, noisy=noisy)
        newAmp, percent = datasetMinimum(results, swapAmpOld, swapAmpOld-4/rf, swapAmpOld+4/rf)
        swapAmp = newAmp

        if update:
            if zGate is 'Pi':
                print 'Old swap amplitude was ',q['cZTargetAmp']
                Q['cZTargetAmp']= swapAmp
                print 'New Target swap amplitude is ', swapAmp
            else:
                print 'Old swap amplitude was ',q['cPiHalfTargetAmp']
                Q['cPiHalfTargetAmp']= swapAmp
                print 'New Half-Pi Target swap amplitude is ', swapAmp

    return swapLen, swapAmp

def swap21(sample, swapLen=st.arangePQ(0,100,2,ns), swapAmp=np.arange(-0.05,0.05,0.002), measure
=0, stats=600L,
         name='Qfe-R10 swap MQ', save=True, collect=False, noisy=True,  swapAmpBND=0.20,
         swapAmpSteps=0.001):
    sample, qubits = util.loadQubits(sample)
    q = qubits[measure]

    if swapAmp is None:
        swapAmp = q.cZTargetAmp
        swapAmp = st.r[swapAmp*(1-swapAmpBND):swapAmp*(1+swapAmpBND):swapAmpSteps]

    axes = [(swapAmp, 'swap pulse amplitude'), (swapLen, 'swap pulse length')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

    df = q.piDf21
    def func(server, currAmp, currLen):
```

Figure A.31: Code for qubit-resonator calibrations page 6 of 8

```
        q.xy = eh.mix(q, eh.piPulseHD(q, 0))+eh.mix(q, env.gaussian(q.piLen, q.piFWHM, q.piAmp21
        ,df =df), freq = 'f21')
        q.z = env.rect(q['piLen']*1.5, currLen, currAmp) + eh.measurePulse2(q, q['piLen']*1.5 +
        currLen)
        q['readout'] = True
        return runQubits(server, qubits, stats=stats, probs=[1])

    return sweeps.grid(func, axes, save=save, dataset=dataset, collect=collect, noisy=noisy)

def fockTuners21(sample, n=1, iteration=3, tuneOS=False, stats=1500L, measure=0,
                 save=False, collect=True, noisy=True, update=True):
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    q = qubits[measure]
    Q = Qubits[measure]

    if len(q['cZTargetLens'])<n:
        for i in np.arange(len(q['cZTargetLens']),n,1):
            q['cZTargetLens'].append(q['cZTargetLens'][0]/np.sqrt(i+1/2.0))
    for i in np.arange(1,n+1,1):
        for iter in range(iteration):
            rf = 2**iter
            print 'iteration %g...' % iter
            sl = q['cZTargetLens'][i-1]
            sl = sl/2
            results = fockScans21(sample, n=i, scanLen=st.PQlinspace(-max([0.3*sl['ns']/rf,1]),
            max([0.3*sl['ns']/rf,1]),21,'ns'),
                                  stats=stats, measure=measure, probeFlag=False,
                                  save=False, collect=collect, noisy=noisy)
            new, percent = datasetMinimum(results, 0, -max([0.3*sl['ns']/rf,1]), max([0.3*sl[
            'ns']/rf,1]))
            newLen = 2*new
            q['cZTargetLens'][i-1] += newLen

        if save:
            fockScans21(sample, n=i, scanLen=st.arangePQ(0,500,1,'ns'),
                        stats=stats, measure=measure, probeFlag=True,
                        save=save, collect=collect, noisy=noisy)
    if update:
        Q['cZTargetLens'] = q['cZTargetLens']
        Q['cZTargetLen']  = q['cZTargetLens'][0]
    return q['cZTargetLens'][0]

def resonatorT1(sample, delay=st.arangePQ(0,1,0.01,'us')+st.arangePQ(1,7,0.1,'us'),stats=600L,
measure=0,
       name='resonator T1 MQ', save=True, collect=True, noisy=True, whichRes='Coupler'):
    sample, qubits = util.loadQubits(sample)
    q = qubits[measure]

    axes = [(delay, 'Measure pulse delay')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measure, kw=kw)

    if whichRes is 'Coupler':
```

Figure A.32: Code for qubit-resonator calibrations page 7 of 8

```
        sl = q.cZControlLen
        sa = q.cZControlAmp
    else:
        sl = q.memReadWriteLen
        sa = q.memReadWriteAmp

    def func(server, delay):
        q.xy = eh.mix(q, eh.piPulseHD(q, 0))
        q.z = env.rect(q.piLen/2, sl, sa)
        q.z += env.rect(q.piLen/2+sl+delay, sl, sa)
        q.z += eh.measurePulse(q, q.piLen/2+sl+delay+sl)
        q['readout'] = True
        return runQubits(server, qubits, stats, probs=[1])
    return sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)
```

Figure A.33: Code for qubit-resonator calibrations page 8 of 8

```
###### From automateDaily.py ########

def gate_bringup(s):
    start = datetime.now()

    N = len(s['config'])
    for i in range(N):
        print 'Begin Calibrating Single Qubit Hadamard Gates'
        print 'Z-pi pulse tuner'
        mq.pitunerZ(s, measure=i, save=True, stats = 1800, update=True, noisy=False)
        print 'done tuning Z-pi amplitude for qubit %d' %i,
        hadi.hadamardTrajectory(s, measure=i, stats=1500, useHD=True, useTomo=True, tBuf=5*ns,
        save=True, noisy=False)
        print 'plotting hadamard trajectory on Bloch Sphere'
        print 'correcting for visibilities...generating pretty plots'
        hadi.plotTrajectory(path=s._dir, dataset=None, state=None) #grabs the most recent
        dataset in the current session
        hadi.plotDensityArrowPlot(path=s._dir, dataset = None) #grabs most recent dataset in
        the current session

    end = datetime.now()
    print 'start:', start
    print 'end:', end
    print 'elapsed time for single qubit gate bringups:', (end-start)

##### From multiqubit.py ####

    def cZCalP1(sample, targetAmp=st.r[-0.25:0:0.001], measureC=0, measureT=1, zGate='Pi', stats
    =1500L,
         name='Control-Z Step 1 TargetCal MQ', save=True, collect=False, noisy=True, update=
         False):
    """Generalized Ramsey. Performs the controlled-Z gate Z-pulse sequence
    with pi/2 pulse on target and no microwaves on control qubit [qt, qc]
    to calibrate the phase correction on the target qubit.
    Record any maximum and any minimum of the Ramsey.
    If Update, then this is semi-automatic with user-controlled sliders to update registry keys:
    cZCalP1Max
    cZCalP1Min

    zGate flag: ='Pi' is for a controlled Z-Pi
                ='HalfPi' is for a controlled Z-pi/2 (aka QFT)
    """
    name = '%s zGate=%s' % (name, zGate)
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    qc = qubits[measureC] #control qubit
    qt, Qt = qubits[measureT], Qubits[measureT] #target qubit

    if zGate is 'Pi':
        gateTime = qt.cZTargetLen
    elif zGate is'HalfPi':
        gateTime = qt.cPiHalfTargetLen
    else:
        gateTime = qt.cZTargetLen
```

Figure A.34: Code for qubit gate calibrations page 1 of 3

```
    axes = [(targetAmp, 'target amp phase correction')]
    kw = {'stats': stats}
    dataset = sweeps.prepDataset(sample, name, axes, measure=measureT, kw=kw)

    def func(server, targetAmp):
        start = 0
        #pad time for equal state prep time in Step 2 Cal
        start += qc['piLen']/2 + qc['cZControlLen']
        #state prep
        #Control qubit no microwaves, Target qubit pi/2
        #Tighter timing. End of piHalf pulse aligns with end of iSWAP from Control
        qt.xy = eh.mix(qt, eh.piHalfPulseHD(qt, start-qt['piLen']/2))
        #Control is IDLE

        #Target Phase swap Q21 with R21 for iswap^2 time
        qt.z = env.rect(start, gateTime, qt.cZTargetAmp)
        start += gateTime

        #Target phase correction, time is fixed sweeping amplitude
        qt.z += env.rect(start, qt.cZTargetPhaseCorrLen, targetAmp)
        start += qt.cZTargetPhaseCorrLen + qt['piLen']/2
        #Final pi/2 for Ramsey, rotate about X
        qt.xy += eh.mix(qt, eh.piHalfPulseHD(qt, start, phase=0.0*np.pi))
        start += qt['piLen']/2

        #Measure only the Target
        qt.z += eh.measurePulse(qt, start)

        qt['readout'] = True

        return runQubits(server, qubits, stats=stats, probs=[1])
#    return sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)
    data = sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)

    if update:
        squid.adjust_cZTargetPhaseCorrAmpP1(Qt, data)

def cZCalP2(sample, targetAmp=st.r[-0.25:0:0.001], measureC=0, measureT=1, zGate='Pi', stats=
1500L,
         name='Control-Z Step 2 TargetCal MQ', save=True, collect=False, noisy=True, update=
         False):
    """Generalized Ramsey. Performs the controlled-Z gate Z-pulse sequence
    with pi/2 pulse on target and a pi-pulse on the control qubit [qc, qt]
    to verify the "pi" phase shift from cZCalP1 on the target qubit.
    Look for a Min, should be really close to Max from Part 1
    zGate flag: ='Pi' is for a controlled Z-Pi
                ='HalfPi' is for a controlled Z-pi/2 (aka QFT)
    """
    name = '%s zGate=%s' % (name, zGate)
    sample, qubits, Qubits = util.loadQubits(sample, write_access=True)
    qc = qubits[measureC] #control qubit
    qt, Qt = qubits[measureT], Qubits[measureT] #target qubit
```

Figure A.35: Code for qubit gate calibrations page 2 of 3

```
if zGate is 'Pi':
    gateTime = qt.cZTargetLen
elif zGate is'HalfPi':
    gateTime = qt.cPiHalfTargetLen
else:
    gateTime = qt.cZTargetLen

axes = [(targetAmp, 'target amp phase correction')]
kw = {'stats': stats}
dataset = sweeps.prepDataset(sample, name, axes, measure=measureT, kw=kw)

def func(server, targetAmp):
    start = 0
    #state prep
    #Control g -> e
    qc.xy = eh.mix(qc, eh.piPulseHD(qc, start))
    start += qc['piLen']/2
    #Control iSWAP with Resonator
    qc.z = env.rect(start, qc.cZControlLen, qc.cZControlAmp)
    start += qc.cZControlLen
    #state prep Target
    qt.xy = eh.mix(qt, eh.piHalfPulseHD(qt, start-qt['piLen']/2))
    #Target Phase swap Q21 with R21 for iswap^2 time
    qt.z = env.rect(start, gateTime, qt.cZTargetAmp)
    start += gateTime

    #Target phase correction, time is fixed sweeping amplitude
    qt.z += env.rect(start, qt.cZTargetPhaseCorrLen, targetAmp)
    start += qt.cZTargetPhaseCorrLen + qt['piLen']/2
    #Final pi/2 for Ramsey, rotate about X
    qt.xy += eh.mix(qt, eh.piHalfPulseHD(qt, start, phase=0.0*np.pi))
    start += qt['piLen']/2

    #Measure
    qt.z += eh.measurePulse(qt, start)

    qt['readout'] = True

    return runQubits(server, qubits, stats=stats, probs=[1])
#return sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)

data = sweeps.grid(func, axes, dataset=save and dataset, noisy=noisy)

if update:
    squid.adjust_cZTargetPhaseCorrAmpP2(Qt, data)
```

Figure A.36: Code for qubit gate calibrations page 3 of 3

Appendix B

QuP Fabrication

In this Appendix I discuss the QuP fabrication[1]. I also include a schedule in §B.2 to complete the QuP recipe detailed in §B.4. In §B.3 I make note of practices that may prove to be helpful in navigating the cleanroom and successfully fabricating devices. The Appendix concludes with the QuP recipe.

B.1 Fabrication Overview

All of the fabrication was completed at the UCSB Nanofabrication facilities. The QuP fabrication process consists of three superconducting aluminum wiring layers, "base-wiring", "top-wiring" and "junction" and one dielectric layer of hydrogenated amorphous Silicon (a-Si:H) which forms the qubit shunt (parallel-plate)

[1]For more background and further detail on the UCSB QC-group's fabrication process please see [3, chap. 5].

capacitor and wiring crossovers. Seven patterning and etching steps define the circuit elements in the QuP. These etch steps are summarized with microphotographs of the QuP device after each etch step in Figure B.1. Table B.1 outlines the sequence of deposition, etching, and oxidation steps used in the QuP fabrication process.

All of the photolithography steps were completed on the GCA AutoStep 200 I-Line Wafer Stepper, using SPR955 photoresist and AZ300MIF developer. The QuP was the first device (within the UCSB QC-group) that utilized the local alignment feature on the AutoStep tool. This enabled sub-micron overlaps and reliable $1\,\mu\mathrm{m}^2$ junction areas. Dry etching was done in a Panasonic E640 inductively coupled plasma (ICP) etch system. After completing the dry etches 1165 Stripper, heated to 80° was used to remove the photoresist. All of the Al sputtering and junction oxidations were completed in a custom Kurt Lesker sputter and ion mill system. The a-Si:H dielectric material was deposited using a UNAXIS high density plasma enhanced chemical vapor deposition (HD PECVD) system.

B.2 Scheduling

The time to fabricate two 3" QuP wafers was spread over 4 days, the majority of which was done during the "graveyard-shift" 6pm-6am to avoid the competition for machines during the normal business hours. There are pros and cons to working

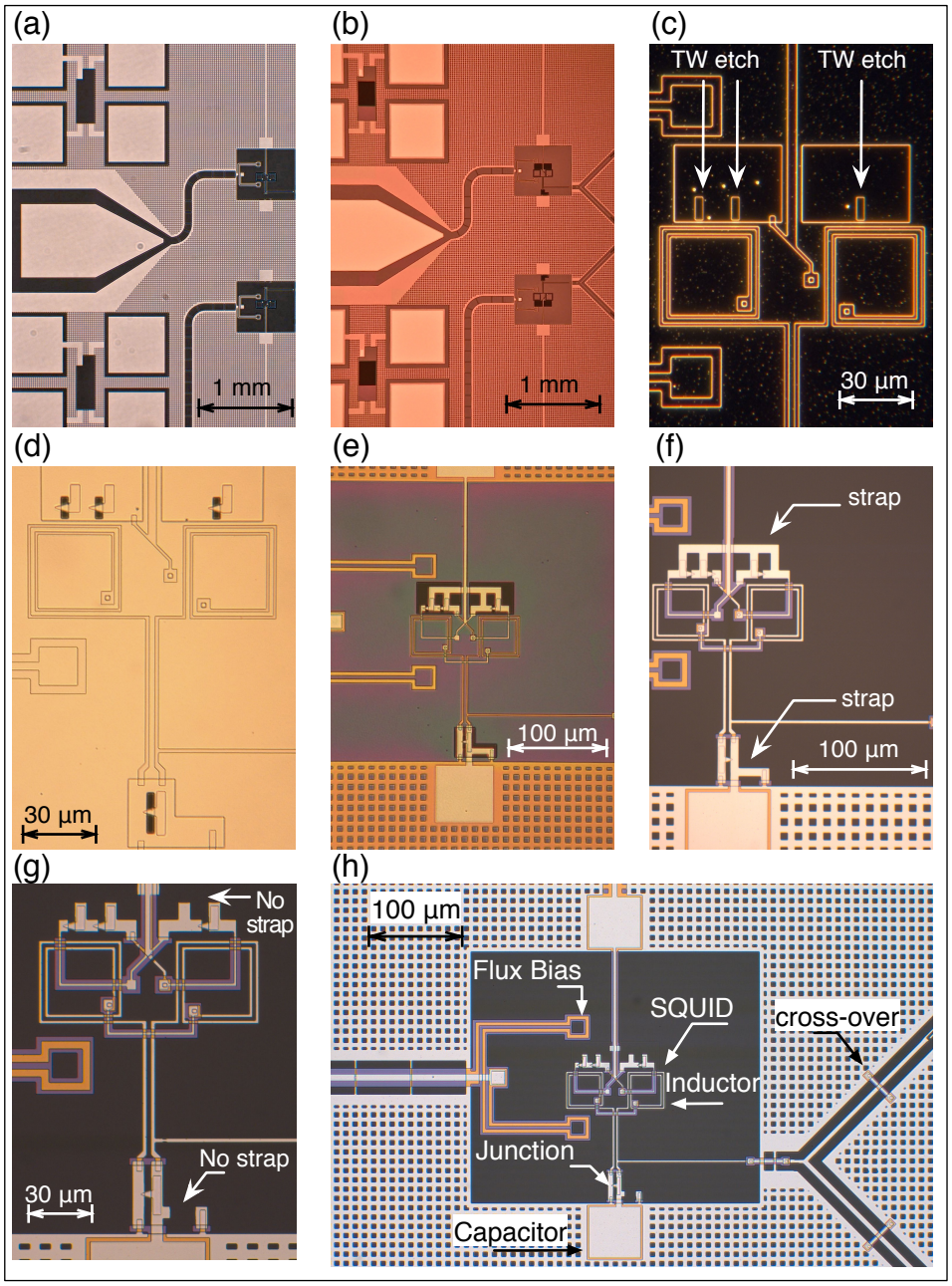

Figure B.1: (a-g) Photomicrographs of the QuP after each of the seven etch steps in fabrication. (a) Base wiring etch §B.4.4. (b) Via etch §B.4.6. (c) Top wiring etch part 1 §B.4.8. (d) Junction etch §B.4.10. (e) Top wiring etch part 2 §B.4.11. (f) Dielectric etch §B.4.12. (g) Junction protection straps etch §B.4.13. (h) Photomicrograph of a completed qubit cell with annotations.

Step	Layer	Notes
1	Deposit Base Wiring (BW), Al 200 nm	MBE grown or (Lesker) Sputtered
2	Etch BW	reactive ion etch (dry-etch)
3	Deposit Dielectric, a-Si:H 200 nm	plasma enhanced chemical vapor deposition
4	Etch Vias	dry-etch
5	Deposit Top Wiring (TW), Al 250 nm	sputtered
6	Etch TW Part 1	dry-etch
7	Oxidize	
8	Deposit Junction, Al 150 nm	sputtered
9	Etch Junction	dry-etch
10	Etch TW Part 2	dry-etch
11	Etch Dielectric	dry-etch
12	Etch Shorting Straps	transene wet-etch to remove junction protection

Table B.1: Overview of the fabrication process.

at these hours. You need to be able to handle machine errors with a level head and on your own. However, because you are sharing the entire cleanroom with only a handful of individuals it is a lot easier to define your *mise en plase*. *Mise en plase* literally translates to "everything in its place" and is used by the culinary field to describe a clean and orderly station ready to execute the recipes of the day and being prepared to deal with challenges that will inevitably come up during your shift. I like this analogy because in a lot of ways cleanroom work is like baking. You are following a recipe with ingredients that have been precisely defined for a reason and if you forget a step at any point you will be starting over. *Mise en place* in the cleanroom means defining your work flow and workspace in a manner

that facilitates the most efficient way -for you- to sequence through the recipe without missing a step and thereby completing functional devices. I found that working through the night provided the right atmosphere for me to think clearly with a relaxed confidence in the bunny suit.

In Table B.2 is a suggested, "relaxed" fabrication schedule. If you need to work at an accelerated pace than just be sure to end your day on one of the correct breaking points (before a lithographic exposure). Remember, during your fabrication the most trusted place is the place that *you* have prepared. This goes for the wet benches, the deposition chambers, the water you use, your photoresist, the photoresist spinners, your tweezers, your glassware, your reticles, your wafer holder, and the storage space for your sample when you end your day. At the end of each day, it is recommended that you place your wafer in the group's Lesker loadlock or an equivalent vacuum space that you know is not contaminated. With all that being said, I hope it is clear that a certain level of paranoia is natural to successfully working in the cleanroom.

B.3 Tips For Success In The Cleanroom

Your time and the devices that you will inevitably get out are worth more than the "consumables" (e.g. photoresist, developer, gloves, 1165, acetone, *etc*), so use what you need. Don't confuse this call to efficiency with rampant waste. I am

Day	Fabrication Steps to Complete
1	Deposit Base Wiring. Expose, Develop, Etch, Clean BW. Store in Lesker loadlock in Vaccuum (LLV)
2	Deposit aSi:H. Expose Develop, Etch, Clean Vias. Deposit Al Top Wiring (TW). Store in LLV
3	Expose, Etch, Clean, TW1. Oxidize, Dep Junctions. Store in LLV
4	Expose, Develop, Etch, Clean Junctions. Expose, Develop, Etch, Clean TW2. Expose, Develop, Etch, Clean aSiH. Store in LLV
5	Expose, Develop, Wet-Etch, Clean Straps. Store in LLV.
6	Probe Junctions, Soft-bake PR, Dice wafer. OK to store in Martinis Group desiccator (MBE lab).

Table B.2: Fabrication schedule.

not suggesting that you waste any of the resources available in the cleanroom, I am just reminding you that all of the days-worth-of-work you spent on a device can be ruined by taking a short cut. Commit to the fact that you are going to be in the cleanroom. Here are some tips that may help you in the cleanroom.

- Store in-progress device wafers in the Lesker load-lock chamber.
- Always pour a fresh bottle of resist when you start your fabrication. And don't let anyone "borrow it".
- Always install fresh napkins in the photoresist (PR) spinners before you spin your PR.
- For your lithography steps, Always use two spinners, one for Hexamethyldisilazane (HMDS) and the other for photoresist. The fumes of the HMDS linger after spinning and the last thing you want to do is blow Nitrogen on

your wafer for risk of depositing the dirt from inside the spinner on your wafer.

- Put on fresh gloves anytime you think about it. About to remove some PR with 1165? Double up on your gloves so you can peel one pair off when you are done with 1165.

- Clean out/off the wafer carrier and your tweezers with isopropanol followed by acetone after *each* lithography step. A good time to perform this task is while your wafer is soaking in 1165.

- Be sure to pour 1165 in a *dry* beaker i.e. no water inside. $1165 + H_20 =$ etching type solution.

B.4 QuP Fabrication Recipe

B.4.1 Reticle Set

The design files for the QuP reticles (a.k.a "masks") are located in the QC-group's archive directory "\Erik\Work\Ledit\ReZQArch4Q5R". The reticles were fabricated out-of-house by Digidat on quartz . The small $\sim 1\,\mu$m features, specifically the junctions, were sharp and well defined, which was critical to making smaller overlap junctions. The reticle set and corresponding lithographic steps are sum-

QuP Reticle Set			
Plate Number	Quadrant	Lithographic Step	Orientation
1	A	Base Wiring	
	B	Via Etch	Rotated 90°
	C	Top Wiring Etch 1	Rotated 180°
	D	Base Wiring Shooters	
	Middle	μ-wave shooters	
2	A	Junction Overlap Etch	
	B	Top Wiring Etch 2	Rotated 90°
	C	Dielectric Etch	Rotated 180°
	D	Junction protection strap removal	Rotated 270°

Table B.3: QuP Reticles and corresponding lithographic steps.

marized in Table B.3. The Orientation and corresponding rotations are due to the GSA Autostepper program for the lithography steps. This is not a standard, one can redefine the shutters on each step rather than rotating. However, I found that my method required fewer things to remember once I was in the cleanroom fabricating devices.

B.4.2 Base Wiring Al Deposition

The QuP was fabricated on a 3" c-plane sapphire (Al_2O_x) substrate chosen for its low loss tangent at GHz frequencies. The base wiring metal deposition was completed using the UCSB QC-group's custom Kurt Lesker superconducting metal sputtering tool. The machine is needed for $45 - 60$ min/wafer. The Al deposition step consists of two actions, an ion-mill step to clean the surface of the bare sapphire (Al_2O_x) wafer followed by the sputtering of Al. The recipe is shown in

Base Wiring Al Deposition								
Action	Experimental Controls							
	V_B	I_B	I_N	V_A	V_D	Pr_{Ar}	P_{rf}	t
	V	mA	mA	V	V	Torr	W	min
1. Mill Al_2O_x wafer	900	16	19	100	35	$2E^{-4}$	n/a	2.5
2. Sputter Al								
2.a. Clean Al target	900	16	19	100	35	$5E^{-3}$	200	3.0
2.b. Deposition	900	16	19	100	35	$5E^{-3}$	110	20.0

Table B.4: Fabrication step 1. Al base wiring deposition on UCSB QC-group's Lesker superconducting metal deposition tool. Experimental controls defined in text

Table B.4, where the experimental controls for the Kurt Lesker deposition tool are beam voltage V_B, beam current I_B, neutralizing current I_N, discharge current I_D, Argon pressure Pr_{Ar}, rf power P_{rf}, and time t. We experience a deposition rate $\sim 10\,\frac{\text{nm}}{\text{min}}$ using the settings in action 2.b. deposition.

B.4.3 Base Wiring Pattern

The AutoStep tool is needed for at least 60 min/wafer to complete the 25 unique base wiring exposures to build up the base wiring connections of the 89 potential QuPs on the wafer. Table B.5 is a wafer map for the designed qubit-resonator coupling strengths (all coupling strengths are in MHz). The map shows a range of coupling strengths from 20 MHz to 100 MHz, concentrated around 50 MHz as summarized in Table B.6. Table B.7 is the wafer map for the microwave control line coupling strengths (all coupling strength are in attoFarrads). And Table B.8

Qubit-Resonator Coupling Strengths in MHz											
	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	$C9$	$C10$	$C11$
$R1$					20	20	20				
$R2$		40	40	40	40	40	40	40			
$R3$		80	20	80	20	80	20	80	20	80	
$R4$		20	80	20	80	20	80	20	80	20	
$R5$	40	40	40	40	40	40	40	40	40	40	40
$R6$	50	50	50	50	50	50	50	50	50	50	50
$R7$	60	60	60	60	60	60	60	60	60	60	60
$R8$		100	100	50	50	100	100	50	50	50	
$R9$		50	100	100	100	20	100	100	100	100	
$R10$		50	50	60	50	60	50	60			
$R11$					60	60	60				

Table B.5: Wafer map of qubit-resonator coupling strengths

summarizes the coupling options used on the various QuP designs.

B.4.4 Base Wiring Etch

The dry Al etch that defines the base wiring is completed using the Panasonic E640 ICP etch system using a Boron trichloride (BCl_3), Chlorine (Cl_2) and Carbon tetraflouride (CF_4) recipe. The chamber is cleaned with a 10 min O_2 plasma and then conditioned for 5.5 min using a blank conditioning wafer subject to the same conditions as the actual etch as detailed in Table B.9. As soon as the wafer is removed from the Panasonic E640 ICP etch system it is soaked with deionized (DI) H_2O for 10 min to scavenge the bi-products from the Al-Cl interaction. Figure B.1a shows a photomicrograph of the device after the base wiring etch.

Summary of Q-R Coupling Strengths	
Coupling Strength MHz	# of Devices
20	12
40	18
50	22
60	17
80	9
100	11
Total Devices	**89**

Table B.6: Number of devices for the various qubit-resonator coupling strength options.

Microwave Coupling Strengths in aF											
	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	$C9$	$C10$	$C11$
$R1$					220	170	130				
$R2$		170	220	130	170	220	170	130			
$R3$		170	220	130	170	220	170	130	220	170	
$R4$		170	220	130	170	170	170	130	220	170	
$R5$	130	170	220	130	170	220	170	130	220	170	130
$R6$	130	170	220	130	170	220	170	130	220	170	130
$R7$	130	170	220	130	170	220	170	130	220	170	130
$R8$		170	220	170	130	220	170	220	170	130	
$R9$		130	170	130	170	220	220	130	220	170	
$R10$		170	220	220	130	220	170	170			
$R11$					130	170	130				

Table B.7: Wafer map of microwave coupling strengths in attoFarrads.

Summary of Coupling Strengths				
Qubit - Resonator Coupling Strength MHz	# of Devices	Microwave Coupling Strengths		
		220 aF	170 aF	130 aF
20	**12**	4	5	3
40	**18**	5	7	6
50	**22**	5	9	8
60	**17**	5	6	6
80	**9**	3	4	2
100	**11**	4	5	2
Total Devices	**89**			

Table B.8: Number of devices for the various qubit-resonator and microwave coupling strength options.

Dry Al Etch Recipe using BCl_3, Cl_2, CF_4							
Step	BCl_3 sccm	Cl_2 sccm	CF_4 sccm	P Pa	P_{rf} W	P_b W	t sec
1	20	40	0	3.0	300	0	5
2	20	40	0	0.7	300	0	5
3	20	40	0	0.7	300	70	30
4	0	0	50	2.0	700	0	5
5	0	0	50	2.0	700	20	5

Table B.9: Dry Al etch recipe using Boron trichloride (BCl_3), Chlorine (Cl_2) and Carbon tetraflouride (CF_4) with chamber pressure P, plasma rf power P_{rf}, substrate forward bias P_b, and step time t.

B.4.5 Hydrogenated Amorphous Silicon Deposition

After the base wiring has been defined the a-Si:H dielectric layer is deposited using the UNAXIS HD PECVD system. The UNAXIS needs to be reserved for 3 hours for 1 wafer (add an hour for every additional wafer) because the chamber needs to be cleaned at a temperature of 250° C with Sulfur hexafluoride (SF_6) and allowed to cool back down to the deposition temperature of 100° C. Once the chamber is clean and has cooled down to 100° C, the deposition is preceded by a seasoning step that mimics the deposition recipe. This seasoning step is used to prepare the chamber and to verify correct machine operation (e.g. to verify that the plasma ignites). The a-Si:H recipe is summarized in Table B.10 it consists of two main steps, the first is an Argon (Ar) mill that is used to remove the native Al oxide and prepare the surface for the dielectric deposition. The second is the multi-stage deposition process that results in a deposition rate of $\sim 1.3\,\text{nm/sec}$ of a-Si:H.

B.4.6 Pattern and Etch Vias in Dielectric

The vias are lithographically defined with the AutoStepper (by exposing reticle 1, quadrant B across the wafer) and finish the patterning with a dry CF_4, O_2 etch. The dry etch on Panasonic E640 ICP etch system punches holes through the dielectric and exposes the base wiring for via connections between the base wiring

a-Si:H Dielectric Deposition						
Step	SiH_4 sccm	Ar sccm	P mTorr	P_{rf} W	P_b W	t sec
1. Ar mill	0	30	1.0	600	100	15
2. Deposition						
2.a. Gas Stabilization	40	5	10.0	0	0	20
2.b. Ignition	40	5	10.0	10	20	4000
2.c. Dep. low power	30	15	2.0	400	30	15
2.d. Dep. full power	30	15	2.0	400	50	180

Table B.10: a-Si:H Dielectric deposition recipe using a HD PECVD system with chamber pressure P, plasma rf power P_{rf}, substrate forward bias P_b, and step time t.

Dry a-Si:H Etch Recipe using CF_4, O_2							
Step	CF_4 sccm	O_2 sccm	N_2 sccm	P Pa	P_{rf} W	P_b W	t sec
1	40	5	0	1.0	500	0	5
2	40	5	0	1.0	700	0	5
3	40	5	0	1.0	700	50	160
4	0	0	50	2.5	100	0	10
5	0	0	50	2.5	50	0	5

Table B.11: Dry a-SI:H etch recipe.

and top wiring Al. The chamber of the Panasonic is cleaned with a 10 min O_2 plasma and then conditioned for 5.5 min using a blank conditioning wafer subject to the same conditions as the actual etch as detailed in Table B.11. The holes etched in the dielectric can be seen in Figure B.1b, where the red colored area is the blanket of a-Si:H and the black regions are the holes.

B.4.7 Top Wiring Al Deposition

After defining the vias in the a-Si:H dielectric the wafer is covered with another layer of Al that will become the top wiring. The deposition is carried out in the Kurt Lesker tool and uses the same experimental parameters as detailed in Table B.4 with an extended time ($\sim$ 30 min) so as to deposit enough Al ($\sim 250 - 300$ nm) to fill in the voids of the a-Si:H.

B.4.8 Pattern and Etch Top Wiring Part 1

The AutoStepper exposes reticle 1, quadrant C across the wafer and finish the patterning with the same BCl_3, Cl_2 and CF_4 Al dry etch recipe detailed in Table B.9. The dry etch is also followed by a 10 min DI H_2O soak. Note that the dry Al etch is nonlinear, so the etch time in step 3 of Table B.9 only needs to be increased from 30 sec to 34 sec since most of the etch time is spent removing the Al oxide. This etch of the top wiring (part 1 of 2) cuts holes for the junctions as seen in Figure B.1c where the white arrows are pointing at the top wiring holes for the three SQUID junctions.

B.4.9 Josephson Junction Al Oxidation and Deposition

After defining the holes in the top wiring the wafer is inserted into the Kurt Lesker tool to mill, then oxidize the Al, and deposit the counter electrode of the

Junction Oxidation							
Wafer	Chamber P [mTorr] at time t [min]						
	$t = 0$	3	5	7	9	10	11
A	86	75	71	69	66	66	66
B	100	97	96	94	93		

Table B.12: Oxidations for two wafers A and B of the QuP. Wafer B oxidation ended at $t = 9$ min.

Josephson junction. The milling step uses the same experimental parameters as detailed in Table B.4 (action 1. Mill) with the exception of a reduced milling time from 2.5 min to 2.0 min. This ion mill step removes the dirty native Al oxide that will be replaced with a controlled oxide growth as detailed in Table B.12[2]. After oxidizing the Al, a fresh layer of Al is sputtered for the junction counter electrode using the same parameters as detailed in Table B.4 (action 2.b. deposition) with a sputtering time of15 min.

B.4.10 Pattern and Etch Junctions

The AutoStepper exposes reticle 2, quadrant A with the appropriate shifts in the exposures across the wafer. The shifts are detailed in Table B.13. The junctions are etched in the Panasonic E640 ICP etch system with an Ar mill combined with Cl_2. This is a slower and more controllable etch as compared to the BCl_3, Cl_2 and CF_4 Al dry etch. The junction etch recipe is detailed in Table B.14. The dry etch

[2]Ideally, both wafers have the same oxidation parameters. Since the critical current depends both on the oxide thickness and the junction area, a higher (lower) oxidation can be compensated for in the exposure shifts by increasing (decreasing) the overlap.

Junction Exposure Shifts in $[nm]$											
Wafer	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	$C9$	$C10$	$C11$
A	600	600	500	200	150	0	100	300	400	600	600
B	2400	2400	2000	800	600	0	400	1200	1600	2400	2400

Table B.13: Table summarizing the shift of Josephson junction overlaps across the two wafers A and B. The junction is designed for a $2.3\,\mu$m overlap. After calibrating an etch-back of $1.1 - 1.4\,\mu$m a shift of 0 nm resulted in $0.9 - 1.2\,\mu$m of overlap. All shifts indicated in the table are positive, resulting in additional overlap. Note, this does not follow standard shift procedures of bracketing above and below 0 nm.

Dry Junction Etch Recipe using Ar, Cl_2							
Step	Ar sccm	Cl_2 sccm	CF_4 sccm	P Pa	P_{rf} W	P_b W	t sec
1	40	3	0	3.0	400	0	5
2	40	3	0	0.2	400	0	5
3	40	3	0	0.2	400	150	160
4	0	0	50	2.0	700	0	5
5	0	0	50	2.0	700	20	10

Table B.14: Junciton etch recipe.

is followed by a 10 min DI H_2O soak. This etch defines the junctions which are the four (3 for the SQUID and 1 for the qubit) wedge shaped elements covering the black rectangles in Figure B.1d.

B.4.11 Pattern and Etch Top Wiring Part 2

To complete the second part of the top wiring pattern and etch the AutoStepper exposes reticle 2, quadrant B across the wafer and the Al is etched with the same BCl_3, Cl_2 and CF_4 recipe detailed in Table B.9. The dry etch is also followed

by a 10 min DI H_2O soak. This etch removes the majority of the Al that capped the dielectric layer as seen in Figure B.1e. The purple-green colored region is the a-Si:H dielectric that will be removed in the next etch.

B.4.12 Pattern and Etch Dielectric

After patterning the regions defined in retciel 2, quadrant C for the dielectric removal using the AutoStepper the final dry etch removes the excess a-Si:H. The same dry CF_4, O_2 etch used for the via Table B.11 clears the remaining dielectric except for the crossovers and parallel-plate capacitors. The nearly completed device is shown in Figure B.1f, where all that remains to be removed are the protection straps around the junctions.

B.4.13 Pattern and Wet Etch Junction Protection Straps

The final etch is a wet etch to remove the grounding protection straps around the junctions, which are indicated with white arrows in Figure B.1f. This etch uses 100 mL of Transene Al etch type A, warmed on a hotplate to a temperature of $T = 50^\circ$ C. At this T and with soft agitation the Al is etched at a rate $\sim$ 10 nm/sec. The shorting straps are removed (compare the voids of Figure B.1g and Figure B.1f) and a completed device is shown (with annotations of the elements) in Figure B.1h.

Bibliography

[1] A. Acín, D. Bruss, M. Lewenstein, and A. Sanpera. Classification of Mixed Three-Qubit States. *Physical Review Letters*, 87(4):040401, Jul 2001.

[2] F. Altomare, J. I. Park, K. Cicak, M. A. Sillanpää, M. S. Allman, D. Li, A. Sirois, J. A. Strong, J. D. Whittaker, and R. W. Simmonds. Tripartite interactions between two phase qubits and a resonant cavity. *Nature Physics*, 6(10):777–781, Aug. 2010.

[3] M. Ansmann. *Benchmarking the Superconducting Josephson Phase Qubit: The Violation of Bell's Inequality*. PhD thesis, University of California, Santa Barbara, 2009.

[4] M. Ansmann, H. Wang, R. C. Bialczak, M. Hofheinz, E. Lucero, M. Neeley, A. D. O'Connell, D. Sank, M. Weides, J. Wenner, A. N. Cleland, and J. M. Martinis. Violation of Bell's inequality in Josephson phase qubits. *Nature*, 461(7263):504–506, September 2009.

[5] O. Astafiev, K. Inomata, a. O. Niskanen, T. Yamamoto, Y. a. Pashkin, Y. Nakamura, and J. S. Tsai. Single artificial-atom lasing. *Nature*, 449(7162):588–90, Oct. 2007.

[6] O. Astafiev, Y. Pashkin, T. Yamamoto, Y. Nakamura, and J. Tsai. Single-shot measurement of the Josephson charge qubit. *Physical Review B*, 69(18):2–5, May 2004.

[7] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, Nov 1995.

[8] R. Barends, J. Wenner, M. Lenander, Y. Chen, R. C. Bialczak, J. Kelly, E. Lucero, P. OMalley, M. Mariantoni, D. Sank, H. Wang, T. C. White, Y. Yin, J. Zhao, A. N. Cleland, J. M. Martinis, and J. J. A. Baselmans. Minimizing quasiparticle generation from stray infrared light in superconducting quantum circuits. *Applied Physics Letters*, 99(11):113507, 2011.

[9] D. Beckman, A. Chari, S. Devabhaktuni, and J. Preskill. Efficient networks for quantum factoring. *Physical Review A*, 54(2):1034–1063, Aug. 1996.

[10] R. Bialczak, R. McDermott, M. Ansmann, M. Hofheinz, N. Katz, E. Lucero, M. Neeley, A. O'Connell, H. Wang, A. Cleland, and J. Martinis. 1/f Flux Noise in Josephson Phase Qubits. *Physical Review Letters*, 99(18):1–4, Nov. 2007.

[11] R. C. Bialczak. *Development of the Fundamental Components of A Superconducting Qubit Quantum Computer*. PhD thesis, University of California, Santa Barbara, 2011.

[12] R. C. Bialczak, M. Ansmann, M. Hofheinz, E. Lucero, M. Neeley, A. D. O'Connell, D. Sank, H. Wang, J. Wenner, M. Steffen, A. N. Cleland, and J. M. Martinis. Quantum process tomography of a universal entangling gate implemented with Josephson phase qubits. *Nature Physics*, 6(6):409–413, Apr. 2010.

[13] R. Bianchetti, S. Filipp, M. Baur, J. Fink, C. Lang, L. Steffen, M. Boissonneault, A. Blais, and A. Wallraff. Control and Tomography of a Three Level Superconducting Artificial Atom. *Physical Review Letters*, 105(22):1–4, Nov. 2010.

[14] F. Buscemi. Shors quantum algorithm using electrons in semiconductor nanostructures. *Physical Review A*, 83(1), Jan. 2011.

[15] J. Cirac and P. Zoller. Quantum Computations with Cold Trapped Ions. *Physical Review Letters*, 74(20):4091–4094, 1995.

[16] K. Cooper, M. Steffen, R. McDermott, R. Simmonds, S. Oh, D. Hite, D. Pappas, and J. Martinis. Observation of Quantum Oscillations between a Josephson Phase Qubit and a Microscopic Resonator Using Fast Readout. *Physical Review Letters*, 93(18):2–5, Oct. 2004.

[17] A. D. Corcoles, J. M. Chow, J. M. Gambetta, C. Rigetti, J. R. Rozen, G. A. Keefe, M. Beth Rothwell, M. B. Ketchen, and M. Steffen. Protecting superconducting qubits from radiation. *Applied Physics Letters*, 99(18):181906, 2011.

[18] L. DiCarlo, J. M. Chow, J. M. Gambetta, L. S. Bishop, B. R. Johnson, D. I. Schuster, J. Majer, A. Blais, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature*, 460(7252):240–244, June 2009.

[19] L. Dicarlo, M. D. Reed, L. Sun, B. R. Johnson, J. M. Chow, J. M. Gambetta, L. Frunzio, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf. Preparation and measurement of three-qubit entanglement in a superconducting circuit. *Nature*, 467(7315):574–8, Sept. 2010.

[20] J. Fink, R. Bianchetti, M. Baur, M. Göppl, L. Steffen, S. Filipp, P. Leek, A. Blais, and A. Wallraff. Dressed Collective Qubit States and the Tavis-Cummings Model in Circuit QED. *Physical Review Letters*, 103(8):1–4, Aug. 2009.

[21] N. A. Gershenfeld and I. L. Chuang. Bulk Spin-Resonance Quantum Computation. *Science*, 275(5298):350–356, Jan. 1997.

[22] S. Hill and W. Wootters. Entanglement of a Pair of Quantum Bits. *Physical Review Letters*, 78(26):5022–5025, June 1997.

[23] M. Hofheinz, H. Wang, M. Ansmann, R. C. Bialczak, E. Lucero, M. Neeley, A. D. O'Connell, D. Sank, J. Wenner, J. M. Martinis, and A. N. Cleland. Synthesizing arbitrary quantum states in a superconducting resonator. *Nature*, 459(7246):546–549, May 2009.

[24] M. Hofheinz, E. M. Weig, M. Ansmann, R. C. Bialczak, E. Lucero, M. Neeley, A. D. O'Connell, H. Wang, J. M. Martinis, and a. N. Cleland. Generation of Fock states in a superconducting quantum circuit. *Nature*, 454(7202):310–4, July 2008.

[25] A. A. Houck, D. I. Schuster, J. M. Gambetta, J. A. Schreier, B. R. Johnson, J. M. Chow, L. Frunzio, J. Majer, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Generating single microwave photons in a circuit. *Nature*, 449(7160):328–31, Sept. 2007.

[26] A. Imamoglu, D. D. Awschalom, G. Burkard, D. P. Divincenzo, D. Loss, M. Sherwin, and A. Small. Quantum Information Processing Using Quantum Dot Spins and Cavity QED. *Physical Review Letters*, 83(20):4204, 1999.

[27] E. Jaynes and F. Cummings. Comparison of quantum and semiclassical radiation theories with application to the beam maser. *Proceedings of the IEEE*, 51(1):89–109, 1963.

[28] J. Johansson, S. Saito, T. Meno, H. Nakano, M. Ueda, K. Semba, and H. Takayanagi. Vacuum Rabi Oscillations in a Macroscopic Superconducting Qubit LC Oscillator System. *Physical Review Letters*, 96(12):127006, Mar. 2006.

[29] N. Katz, M. Ansmann, R. C. Bialczak, E. Lucero, R. McDermott, M. Neeley, M. Steffen, E. M. Weig, A. N. Cleland, J. M. Martinis, and A. N. Korotkov. Coherent state evolution in a superconducting qubit from partial-collapse measurement. *Science (New York, N.Y.)*, 312(5779):1498–500, June 2006.

[30] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O'Brien. Quantum computers. *Nature*, 464(7285):45–53, March 2010.

[31] B. Lanyon, T. Weinhold, N. Langford, M. Barbieri, D. James, A. Gilchrist, and A. White. Experimental Demonstration of a Compiled Version of Shors Algorithm with Quantum Entanglement. *Physical Review Letters*, 99(25), Dec. 2007.

[32] A. Lenstra and H. W. Lenstra. *The development of the number field sieve*. Springer-Verlag Berlin Heidelberg, 1993.

[33] C.-Y. Lu, D. Browne, T. Yang, and J.-W. Pan. Demonstration of a Compiled Version of Shors Quantum Factoring Algorithm Using Photonic Qubits. *Physical Review Letters*, 99(25), Dec. 2007.

[34] E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O. Malley, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland, and J. M. Martinis. Computing prime factors with a Josephson phase qubit quantum processor. *arXiv:1202.5707v1*, pages 1–5, 2012.

[35] E. Lucero, M. Hofheinz, M. Ansmann, R. C. Bialczak, N. Katz, M. Neeley, A. D. O'Connell, H. Wang, A. N. Cleland, and J. M. Martinis. High-Fidelity Gates in a Single Josephson Qubit. *Physical Review Letters*, 100(24):247001, 2008.

[36] E. Lucero, J. Kelly, R. Bialczak, M. Lenander, M. Mariantoni, M. Neeley, A. D. O'Connell, D. Sank, H. Wang, M. Weides, J. Wenner, T. Yamamoto, A. Cleland, and J. M. Martinis. Reduced phase error through optimized control of a superconducting qubit. *Physical Review A*, 82(4):1–7, Oct. 2010.

[37] H. Mabuchi and A. C. Doherty. Cavity quantum electrodynamics: coherence in context. *Science (New York, N.Y.)*, 298(5597):1372–7, Nov. 2002.

[38] J. Majer, J. M. Chow, J. M. Gambetta, J. Koch, B. R. Johnson, J. A. Schreier, L. Frunzio, D. I. Schuster, A. A. Houck, A. Wallraff, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Coupling superconducting qubits via a cavity bus. *Nature*, 449(7161):443–7, Sept. 2007.

[39] M. Mariantoni, H. Wang, T. Yamamoto, M. Neeley, R. C. Bialczak, Y. Chen, M. Lenander, E. Lucero, A. D. O'Connell, D. Sank, M. Weides, J. Wenner, Y. Yin, J. Zhao, A. N. Korotkov, A. N. Cleland, and J. M. Martinis. Implementing the Quantum von Neumann Architecture with Superconducting Circuits. *Science*, 334(6052):61–65, Sept. 2011.

[40] J. Martinis, S. Nam, J. Aumentado, and C. Urbina. Rabi Oscillations in a Large Josephson-Junction Qubit. *Physical Review Letters*, 89(11):9–12, Aug. 2002.

[41] J. M. Martinis, K. B. Cooper, R. McDermott, M. Steffen, M. Ansmann, K. D. Osborn, K. Cicak, S. Oh, D. P. Pappas, R. W. Simmonds, and C. C. Yu. Decoherence in josephson qubits from dielectric loss. *Physical Review Letters*, 95(21):210503, 2005.

[42] R. Mcdermott. Materials Origins of Decoherence in Superconducting Qubits. *IEEE Transactions on Applied Superconductivity*, 19(1):2–13, 2009.

[43] A. Megrant, C. Neill, R. Barends, B. Chiaro, Y. Chen, L. Feigl, J. Kelly, E. Lucero, M. Mariantoni, P. J. J. O. Malley, D. Sank, A. Vainsencher, J. Wenner, T. C. White, Y. Yin, J. Zhao, C. J. Palmstrø m, and J. M. Martinis. Planar superconducting resonators with internal quality factors above one million. *Applied Physics Letters*, 100:113510, 2012.

[44] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits. *Physical Review Letters*, 103(11):110501, 2009.

[45] M. Neeley, M. Ansmann, R. C. Bialczak, M. Hofheinz, N. Katz, E. Lucero, A. O'Connell, H. Wang, A. N. Cleland, and J. M. Martinis. Process tomography of quantum memory in a josephson-phase qubit coupled to a two-level state. *Nature Physics*, 4:523–526, April 2008.

[46] M. Neeley, M. Ansmann, R. C. Bialczak, M. Hofheinz, E. Lucero, A. D. O'Connell, D. Sank, H. Wang, J. Wenner, A. N. Cleland, M. R. Geller, and J. M. Martinis. Emulation of a Quantum Spin with a Superconducting Phase Qudit. *Science*, 325(5941):722–725, August 2009.

[47] M. Neeley, R. C. Bialczak, M. Lenander, E. Lucero, M. Mariantoni, A. D. O'Connell, D. Sank, H. Wang, M. Weides, J. Wenner, Y. Yin, T. Yamamoto, A. N. Cleland, and J. M. Martinis. Generation of three-qubit entangled states using superconducting phase qubits. *Nature*, 467(7315):570–3, Sept. 2010.

[48] M. G. Neeley. *Generation of Three-Qubit Entanglement Using Josephson Phase Qubits*. PhD thesis, University of California at Santa Barbara, 2010.

[49] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, October 2000.

[50] A. D. O'Connell, M. Hofheinz, M. Ansmann, R. C. Bialczak, M. Lenander, E. Lucero, M. Neeley, D. Sank, H. Wang, M. Weides, J. Wenner, J. M. Martinis, and A. N. Cleland. Quantum ground state and single-phonon control of a mechanical resonator. *Nature*, 464(7289):697–703, 2010.

[51] A. D. OConnell, M. Ansmann, R. C. Bialczak, M. Hofheinz, N. Katz, E. Lucero, C. McKenney, M. Neeley, H. Wang, E. M. Weig, A. N. Cleland, and J. M. Martinis. Microwave dielectric loss at single photon energies and millikelvin temperatures. *Applied Physics Letters*, 92(11):112903, 2008.

[52] J. R. Petta, a. C. Johnson, J. M. Taylor, E. a. Laird, a. Yacoby, M. D. Lukin, C. M. Marcus, M. P. Hanson, and a. C. Gossard. Coherent manipulation of coupled electron spins in semiconductor quantum dots. *Science (New York, N.Y.)*, 309(5744):2180–4, Sept. 2005.

[53] P. M. Platzman and M. I. Dykman. Quantum Computing with Electrons Floating on Liquid Helium. *Science*, 284(5422):1967–1969, June 1999.

[54] A. Politi, J. C. F. Matthews, and J. L. O'Brien. Shor's quantum factoring algorithm on a photonic chip. *Science*, 325(5945):1221, Sept. 2009.

[55] C. Rigetti, S. Poletto, J. M. Gambetta, B. L. T. Plourde, J. M. Chow, A. John, S. T. Merkel, J. R. Rozen, G. A. Keefe, M. B. Rothwell, M. B. Ketchen, and M. Steffen. Superconducting qubit in waveguide cavity with coherence time approaching 0.1ms. *arXiv:1202.5533v1*, pages 1–4, 2012.

[56] D. I. Schuster, A. A. Houck, J. A. Schreier, A. Wallraff, J. M. Gambetta, A. Blais, L. Frunzio, J. Majer, B. Johnson, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Resolving photon number states in a superconducting circuit. *Nature*, 445(7127):515–8, Feb. 2007.

[57] P. Shor. Algorithms for Quantum Computation : Discrete Logarithms and Factoring. *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, pages 124–134, 1994.

[58] M. a. Sillanpää, J. I. Park, and R. W. Simmonds. Coherent quantum state storage and transfer between two phase qubits via a resonant cavity. *Nature*, 449(7161):438–42, Sept. 2007.

[59] M. Steffen, M. Ansmann, R. C. Bialczak, N. Katz, E. Lucero, R. McDermott, M. Neeley, E. M. Weig, A. N. Cleland, and J. M. Martinis. Measurement of the Entanglement of Two Superconducting Qubits via State Tomography. *Science*, 313(5792):1423–1425, 2006.

[60] M. Steffen, M. Ansmann, R. McDermott, N. Katz, R. Bialczak, E. Lucero, M. Neeley, E. Weig, A. Cleland, and J. Martinis. State Tomography of Capacitively Shunted Phase Qubits with High Fidelity. *Physical Review Letters*, 97(5):4–7, Aug. 2006.

[61] M. Steffen, J. M. Martinis, and I. L. Chuang. Accurate control of josephson phase qubits. *Phys. Rev. B*, 68(22):224518, Dec 2003.

[62] F. Strauch, P. Johnson, A. Dragt, C. Lobb, J. Anderson, and F. Wellstood. Quantum Logic Gates for Coupled Superconducting Phase Qubits. *Physical Review Letters*, 91(16):2–5, Oct. 2003.

[63] T. Tessier, I. Deutsch, a. Delgado, and I. Fuentes-Guridi. Entanglement sharing in the two-atom Tavis-Cummings model. *Physical Review A*, 68(6):1–10, Dec. 2003.

[64] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–7, Jan. 2001.

[65] A. Wallraff, D. I. Schuster, A. Blais, L. Frunzio, J. Majer, S. Kumar, S. M. Girvin, and R. J. Schoelkopf. Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics. *Nature*, 431(September):162–167, 2004.

[66] H. Wang, M. Mariantoni, R. C. Bialczak, M. Lenander, E. Lucero, M. Neeley, A. D. O. Connell, D. Sank, M. Weides, J. Wenner, T. Yamamoto, Y. Yin, J. Zhao, J. M. Martinis, and A. N. Cleland. Deterministic Entanglement of Photons in Two Superconducting Microwave Resonators. *Physical Review Letters*, 106(February):060401, 2011.

[67] J. R. Weber, W. F. Koehl, J. B. Varley, a. Janotti, B. B. Buckley, C. G. Van de Walle, and D. D. Awschalom. Quantum computing with defects. *Proceedings of the National Academy of Sciences of the United States of America*, 107(19):8513–8, May 2010.

[68] A. G. White, A. Gilchrist, G. J. Pryde, J. L. O'Brien, M. J. Bremner, and N. K. Langford. Measuring two-qubit gates. *Journal of the Optical Society of America B*, 24(2):172, 2007.

[69] T. Yamamoto, M. Neeley, E. Lucero, R. Bialczak, J. Kelly, M. Lenander, M. Mariantoni, a. OConnell, D. Sank, H. Wang, M. Weides, J. Wenner, Y. Yin, a. Cleland, and J. Martinis. Quantum process tomography of two-qubit controlled-Z and controlled-NOT gates using superconducting phase qubits. *Physical Review B*, 82(18):1–8, Nov. 2010.

[70] J. Q. You and F. Nori. Superconducting Circuits and Quantum Information. *Physics Today*, November:42–47, 2005.