THE UNIVERSITY OF CHICAGO

IN-FRIDGE CLASSICAL CONTROLLERS IN QUANTUM COMPUTING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
MOHAMMAD REZA JOKAR

CHICAGO, ILLINOIS
JANUARY 2022

I dedicate this dissertation to my parents.

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES x

ACKNOWLEDGMENTS xi

ABSTRACT xii

1 INTRODUCTION 1

2 NISQ+: BOOSTING QUANTUM COMPUTING POWER BY APPROXIMATING QUANTUM ERROR CORRECTION 5
- 2.1 Introduction 5
- 2.2 Background 8
  - 2.2.1 Basics of Quantum Computation 8
  - 2.2.2 Quantum Error Correction 9
  - 2.2.3 The Surface Code 10
  - 2.2.4 Quantum Computing Systems Organization 13
  - 2.2.5 Classical Control in Quantum Computing Systems 13
- 2.3 Motivation: Decoding Must be Fast 14
- 2.4 Related Work 17
- 2.5 Decoder Overview and Design 18
  - 2.5.1 Prior Decoding Algorithms and their Practical Implications 18
  - 2.5.2 A Greedy Approach 21
  - 2.5.3 SFQ-Based Decoder 21
- 2.6 Implementation 25
  - 2.6.1 SFQ Implementation of Greedy Decoding 25
  - 2.6.2 Datapath and Subcircuit Design 26
- 2.7 Methodology 28
- 2.8 Evaluations 33
- 2.9 Conclusion 38

3 DIGIQ: A SCALABLE DIGITAL CONTROLLER FOR QUANTUM COMPUTERS USING SFQ LOGIC 39
- 3.1 Introduction 39
- 3.2 Background and Motivation 43
  - 3.2.1 Quantum computing 43
  - 3.2.2 Superconducting qubit controllers and their limitations 45
  - 3.2.3 Opportunities and challenges of SFQ quantum controllers 46
- 3.3 Related Work 48
  - 3.3.1 Cryo-CMOS based quantum controllers 48
  - 3.3.2 SFQ-based quantum gates and controllers 48
  - 3.3.3 SFQ-based accelerators 49

3.4 DigiQ quantum controller 49
3.4.1 SFQ-based universal quantum computation 49
3.4.2 Overview of *DigiQ* architecture 55
3.5 Software calibration of SIMD hardware 56
3.5.1 Calibrating single-qubit gates 58
3.5.2 Calibrating two-qubit gates 60
3.6 Methodology and Results 61
3.6.1 Hardware results of *DigiQ* 61
3.6.2 Algorithmic performance results of *DigiQ* 66
3.7 Conclusions and Future work 69

4 PRACTICAL IMPLICATIONS OF SFQ-BASED TWO-QUBIT GATES 72
4.1 Introduction 72
4.2 Background and Motivation 75
4.2.1 Physical system 75
4.2.2 Microwave optimal control 80
4.2.3 Fidelity functions 81
4.2.4 SFQ control 83
4.2.5 Prior work on SFQ-based gates and the motivation of this paper . . . 85
4.3 Detailed study of SFQ-based two-qubit gates 86
4.3.1 Methodology 87
4.3.2 Entangling SFQ-based two-qubit gates on transmon qubit devices . . 89
4.3.3 Realizing both entangling and non-entangling SFQ-based two-qubit gates on transmon devices 91
4.3.4 SFQ-based two-qubit gates on fluxonium qubit devices 94
4.3.5 Comparison with microwave-based gates 95
4.4 Conclusion 95

5 CONCLUSION 97

REFERENCES 99

LIST OF FIGURES

2.1 Boosting the quantum computation power with approximate error correction schemes. A machine with 1024 faulty physical qubits of error rate 10^{-5} has an SQV of $\approx 10^8$. By performing fast, online, approximate decoding, we can trade the number of computational qubits for gate fidelity and boost the SQV by over a factor of 3,402. Moving to a higher code distance raises this increase to a factor of 11,163. NISQ machines are severely limited by gate fidelity, and introducing error mitigation techniques can have dramatic effects on SQV. . . . 6

2.2 Fig. (a) shows a graphical illustration of a surface code mesh. Gray circles indicate data qubits, and nodes labeled X and Z indicate ancillary qubits measuring X and Z stabilizers, respectively. Ancillary qubits are joined by colored edges to the data qubits that they are responsible for measuring. In Fig. (b) a single data qubit experiences a Pauli X error indicated by red coloring, causing the neighboring Z ancillary qubits to detect an odd parity in their data qubit sets and return +1 measurement values indicated by green coloring. In Fig. (c), the data qubit in red experiences a Pauli Z error, causing the vertically adjacent X ancillary qubits to return +1 measurement values. The entire error syndrome strings for either of these two cases would include a string of 12 values, two of which would be +1 and the remaining 10 would be 0. 10

2.3 This figure shows the errors happened in one row of the surface code grid. Fig. (a) shows a data qubit error pattern spanning across ancillary qubits. Each data qubit experiencing error is indicated in red, and the ancillary qubits returning +1 measurement values are indicated in green. Each ancillary qubit that is adjacent to two erroneous data qubits does not signal the presence of any errors, as the parity of the data qubit sets are still even. This creates an *error string* that runs from the ancillary qubit on the left of the grid to the one on the right. Decoding must map these +1 values to the corresponding set of 4 data qubit errors that generated it. Fig. (b) and Fig. (c) show degeneracy in error syndrome generation by surface code data qubit error patterns. The figures depict two distinct sets of data qubit error patterns that both generate the same error syndromes. Both patterns contain the same number of physical data errors, so these patterns are equally likely assuming independence of errors. 12

2.4 Exponential latency overhead when $f = (\frac{r_{\text{gen}}}{r_{\text{proc}}}) > 1$. X-axis shows the compute time if there is no backlog and y-axis shows the actual wall clock time; if there is no backlog we expect wall clock time to be the same as the compute time (line a). Every time we encounter a T-gate we need to decode all the syndromes up until that gate before we can continue the execution [Terhal, 2015]. When we encounter the first T-gate at time T_0, we need to finish the decoding of the data generated during t_0 (not all the data is already decoded as decoding rate is slower than data generation rate) and it takes R_0 to do that. During R_0 where our quantum system is idle, more syndromes are generated and when we encounter the second T-gate at $T_1 + R_0$, we need to finish decoding those syndromes in addition to the syndromes generated during t_1 before continuing the program execution. **The syndrome data generated during the idle periods is the key reason behind data backlog creation which leads to exponential latency overhead.** 15
2.5 Running times of fault tolerant quantum algorithms with decoders of varying efficiency. The X-axis plots $\frac{r_{\text{gen}}}{r_{\text{proc}}}$. To the left of 1, data is processed as fast as it is generated, whereas rates to the right of 1 indicate that the decoder is slower than syndrome data is generated. The T-gates require synchronization with the decoder in order to execute. Prior work [Chamberland and Ronagh, 2018] claims that fast neural network inference decoders can perform inference in $\sim$ 800 ns, which places the decoder at approximately the 1.5 - 2 region for a system generating syndromes in the 400-500ns range. Our decoding results show that time to solution never exceeds 20ns, placing it below 1. Clearly computation becomes intractable quickly for slow decoders. 16
2.6 Baseline solution to find the two closest hot syndrome modules. Step1: two decoder modules have “1” hot syndrome input. Step2: the hot syndrome modules propagate grow signals. Step3: the grow signals meet at an intermediate module. Step4: the intermediate module sends pair signals in the opposite direction. Step5: pair signals arrive at the hot syndrome modules. Step6: decoding is complete. Note that the decoder modules that receive a pair signal are considered as part of the error chain that has occurred. 22
2.7 Scenarios where the SFQ decoder chooses the wrong chain where (a) no reset/boundary/equidistant mechanisms are employed, (b) no boundary/equidistant mechanisms are employed, and (c) no equidistant mechanism is employed. . . . 23
2.8 Overview of decoder module microarchitecture. 27
2.9 Pair subcircuit after SFQ specific optimizations and mapping. Triangular shapes at the bottom represent the primary inputs of the circuit and those at the top of the circuit show primary outputs. *DFF* is SFQ DRO DFF inserted for path balancing. Splitter (balanced) trees are also shown. Splitter is an asynchronous SFQ gate that receives a pulse at its input and after its intrinsic delay, it produces two almost identical output pulses. We insert splitters at the output of an SFQ gate (or a primary input) with more than one fanout. 28

2.10 Logical error rate performance of each incremental design step. The addition of resets and boundaries each contribute heavily to the realization of pseudo-thresholds, and have a dramatic effect on reducing the minimum achievable logical error rates for each code distance. 30
2.11 Results for our final design, including support for reset, boundary, and equidistant mechanisms. (a) Error rate scaling for the proposed decoder. An accuracy threshold is evident at approximately 5% physical error rate, while pseudo-thresholds span the range from $\sim 3.5\% - 5\%$. (b) Logical error rates near the 5% physical error rate value. (c) Truncated unnormalized estimated probability distributions for the execution cycles required by each code distance in simulation. Window shows up to 20 cycles for comparison across code distances. Notice that while distances $3, 5, 7$ display peaks centered at $0, 5, 9$, and 14 cycles. 31
2.12 Comparison of required code distances of different decoders to execute an algorithm consisting of 100 T-gates. Compared are the SFQ Decoder, MWPM decoder [Fowler et al., 2012a], neural network decoder [Baireuther et al., 2019], Union-Find decoder [Delfosse and Nickerson, 2017], and a theoretical MWPM decoder with no backlog. across both code distances and physical error rates. . . 35

3.1 (a) Today's controller design: controller at room temperature, (b) *DigiQ*: controller close to quantum chip. 40
3.2 (a) Bloch sphere representation of a qubit; (b) SFQ driven trajectory. The blue trajectory is driven by the periodic SFQ pulse train shown in (c), and the orange trajectory is driven by the qubit free evolution; (c) SFQ pulse train in the time domain. f is the qubit oscillation frequency. 44
3.3 The sequence of gates in one cycle of *DigiQ_opt*. 53
3.4 (a) Circuit schematic of our current generator design based on SFQ/DCs; (b) The electrical current pulse generated by our design to realize CZ gates on flux-tunable transmons. 54
3.5 Overview of our *DigiQ* architecture. 55
3.6 Calibration process in (a) today's microwave-based quantum machines; (b) *DigiQ*. 57
3.7 CZ gate error as a function of frequency drift, assuming 1, 2, or 3 U_{qq} operations and ideal single-qubit gates. 61
3.8 Power (a), area (b), and cable count (c) results of *DigiQ_min* and *DigiQ_opt* architectures. *SFQ_MIMD_naive* and *SFQ_MIMD_decomp* results are shown for comparison. 63
3.9 *DigiQ* quantum circuit execution time normalized to the Impossible MIMD system. 67
3.10 (a) Median single-qubit gate error on *DigiQ_opt* (BS=8) and *DigiQ_min* (BS=2) with 1024 qubits (well representative of other configurations); (b) CZ gate error on each qubit pair. Software can map around the outliers using the noise-adaptive mapping techniques [Murali et al., 2019a]. 69

4.1 Bit representation of SFQ pulse trains. (a) coherent pulses are applied to the qubit (1 pulse per qubit oscillation period) to perform rotations around the y axis. (b) a bitstream found by genetic algorithm to perform arbitrary unitary. Bitstreams are processed one bit at a time; if the bit is “0”, no pulse is applied to the qubit, and if the bit is “1”, one SFQ pulse is applied to the qubit. 82
4.2 Error and leakage of the best SFQ-based CZ gate with 20 ns gate time found by the genetic algorithm on transmon qubit devices with Ω_x control fields. Error is computed using Eq. (4.14) and takes into account only the n levels on which the gate was learned. Leakage is computed using Eq. (4.17) and considers higher levels. Thus, low error does not necessarily translate to low leakage. 83
4.3 Error and leakage of the best SFQ-based CZ gate found by the genetic algorithm for transmon qubit devices with Ω_x control fields (plots a and b), and transmon qubit devices with Ω_z control fields (plots c and d). Error is calculated as $1 - fidelity$, and leakage is calculated using Eq. (4.17). Two different tip angles and two fidelity functions are used in our optimal control method (see Sec. 4.2 for the details of our fidelity functions). We run the simulations with $n = 5$ energy levels, and suppress the population of higher energy levels in our optimal control method. 87
4.4 Bit representation of SFQ bitstreams applied to qubit1 (plot a) and qubit2 (plot b) on a transmon system with Ω_x control fields and 0.003 tip angle in order to realize a CZ gate with 20 ns gate time. Each SFQ chip clock cycle is 8 ps. . . . 88
4.5 Error of the best SFQ-based CZ gate (entangling two-qubit gate) and Ry90⊗I gate (non-entangling two-qubit gate) found by the genetic algorithm for transmon qubit devices with only Ω_x control fields. 90
4.6 Sensitivity analysis on qubit coupling strength in transmon system with Ω_x control fields. The results are shown for 0.003 tip angle and 10 ns (plot a), 20 ns (plot b), and 40 ns (plot c) gate times. The SFQ bitstreams are learned with *fid2*. 92
4.7 Error and leakage of the best SFQ-based CZ gate (plots a and b) and Ry90⊗I gate (plots c and d) found by the genetic algorithm for fluxonium qubit devices with Ω_x control fields. 93
4.8 Error comparison between microwave-based gates obtained using Grape code and SFQ-based gates obtained using genetic algorithm (with 0.003 tip angle). The results are reported for CZ gate (plot a) and Ry90⊗I gate (plot b). 94

LIST OF TABLES

2.1 Characteristics of the simulated benchmarks. 14
2.2 The library of ERSFQ cells and corresponding characteristics used for synthesizing the circuit into SFQ hardware. Josephson Junction count is listed in the second column. 32
2.3 Experimental synthesis results for the SFQ Decoder. Shown are all gates utilized in the synthesis, as well as submodules that comprise the main circuit. Pair_Req. and Grow subcircuits have been combined into a single subcircuit. 33
2.4 Decoder execution time in nanoseconds across each code distance studied and across all simulated error rates. 33
2.5 Empirical parameter estimation given a model of the form $P_L \approx c_1(p/p_{\text{th}})^{c_2 \cdot d}$. Shown are estimated c_2 parameter values. 34
2.6 Comparing decoding algorithms in terms of accuracy threshold. SFQ decoder sacrifices accuracy for speed in order to avoid exponential latency overhead and achieve quantum advantage. 37

3.1 Design space for SFQ-based single-qubit gate controllers. 48
3.2 Optimal parking frequencies and drift tolerance for $Rz(\phi)$ gates with $\leq 10^{-4}$ error for $N = 255$. 59
3.3 The library of RSFQ cells and corresponding characteristics used for synthesis. 61
3.4 NISQ benchmark algorithms. 66

4.1 The parameters used in the genetic algorithm. 87

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. Fred Chong, for his continuous guidance and support during my Ph.D. study. I had many insightful discussions with Prof. Chong during my study, in which I learned how to tackle challenging research problems. Thanks to the opportunity to work with him, I developed invaluable hard skills and soft skills, for which I am forever grateful.

I am thankful to Prof. Massoud Pedram and Prof. Hank Hoffman, who have been part of my Ph.D. committee, for all their constructive comments and feedback, which improved the quality of this dissertation.

I would like to thank my co-advisor during the first few years of my Ph.D. study, Prof. Yanjing Li. I am grateful for the opportunity to work on many research projects on novel optical technologies [Jokar et al., 2020, 2019b,a, 2017, Yang et al., 2021].

I would like to express my gratitude to my co-author, Rich Rines, for many insightful meetings that we had. Rich's expertise in physics was crucial in completing the quantum projects. I really enjoyed solving complex problems with Rich. I am also thankful to my co-author, Adam Holmes, for many insightful discussions that we had, and helping me learn quantum error correction and quantum optimal control.

I greatly appreciate the fruitful collaborations with my co-authors at University of Chicago, University of Southern California, and University of Illinois at Urbana-Champaign. I am also thankful to people in Prof. Chong's lab who helped me learn quantum computing.

I thank the International House (I-House) at the University of Chicago, for organizing cultural events and providing an opportunity for students and scholars from all around the world to meet and learn together. Being an I-House fellow has been a rewarding experience and has played a very important role in my growth in life.

Last but not least, I would like to express my deepest gratitude to my family and friends for their continuous love and support all through my Ph.D. study, which made this journey possible.

ABSTRACT

Today's superconducting quantum computer prototypes rely on a classical controller at room temperature that controls the qubits inside the dilution refrigerator. This approach is simple and straightforward, however, it introduces significant scalability challenges: (1) quantum error mitigation techniques that are based on room temperature error decoding face exponential latency overhead due to the data backlog caused by the slow decoding process; (2) scalability is limited due to massive costs of generating and routing the microwave control signals.

In this thesis, novel cryogenic controllers are proposed to address the aforementioned challenges. First, we develop an in-fridge classical accelerator for error decoding using ultra-fast superconducting Single Flux Quantum (SFQ) logic technology to avoid the exponential latency overhead. By enabling practical implementation of quantum error mitigation, we expand the compute volume of near-term machines by factors between 3,402 and 11,163. Second, we develop an in-fridge classical controller using SFQ logic to generate and route the control signals inside the dilution refrigerator. We use state-of-the-art SFQ synthesis tools to calculate the power and area of our in-fridge controller, and show that it can operate within the tight power and area budget of dilution refrigerators at >42,000-qubit scales. Finally, we investigate the practical implications of SFQ-based two-qubit gates and show that they can achieve similar gate fidelity and gate time to that of microwave-based gates. The results of this thesis show that cryogenic controllers play a key role in increasing the scalability and computing power of near-term quantum machines.

CHAPTER 1
INTRODUCTION

Quantum computing has the potential to revolutionize computing and have massive effects on major industries including agriculture, energy, and materials science by solving computational problems that are intractable with conventional machines [Hastings et al., 2014, Svore and Troyer, 2016]. Superconducting quantum computing, one of the most promising technologies for building a quantum computer, has been studied in industry and academia [Brink et al., 2018, Steffen et al., 2011, Fu et al., 2017, Li et al., 2020], and many prototypes have been manufactured in the recent years [Arute et al., 2019, Kelly, 2018, Steffen et al., 2011, Fu et al., 2017]. However, today's prototypes rely on room temperature controllers which present severe scalability challenges.

First, relying on room temperature controllers causes challenges for boosting the computational power of quantum machines. *Quantum error correction* is a classical control technique that decreases the rate of errors in qubits and expands the "Simple Quantum Volume" (SQV). SQV can be defined as the number of computational qubits of a machine multiplied by the number of gates we expect to be able to perform without error. Prior work has suggested and analyzed software solutions for decoding the errors, but relying on hardware-software communication can be slow, especially considering the cryogenic environment of typical quantum computing systems. If decoding occurs slower than error information is generated, the system will generate a backlog of information as it waits for decoding to complete, introducing an exponential time overhead that will kill any quantum advantage.

Second, large-scale quantum computers are essential in running many quantum algorithms and performing quantum error correction. However, today's prototypes rely on sending separate analog microwave control pulses for each qubit from a classical controller at room temperature to the quantum chip inside the dilution refrigerator, which presents severe scalability challenges due to the massive costs of generating/routing the analog microwave signals, and significant heat dissipation at millikelvin temperatures due to using a large number of high bandwidth coaxial cables [McDermott et al., 2018, Leonard et al., 2019, Li et al., 2019].

A promising approach proposed in this thesis to address the aforementioned challenges is utilizing cryogenic classical controllers. This thesis consists of three following projects:

(1) NISQ+: Boosting quantum computing power by approximating quantum error correction (Chapter 2). Quantum computers are growing in size, and design decisions are being made now that attempt to squeeze more computation out of these machines. In this spirit, we design a method to boost the computational power of near-term quantum computers by adapting protocols used in quantum error correction to implement "Approximate Quantum Error Correction (AQEC)." By approximating fully-fledged error correction mechanisms, we can increase the compute volume (qubits $\times$ gates, or "Simple Quantum Volume (SQV)") of near-term machines. The crux of our design is a fast hardware decoder that can approximately decode detected error syndromes rapidly. Specifically, we demonstrate a proof-of-concept that approximate error decoding can be accomplished online in near-term quantum systems by designing and implementing a novel algorithm in superconducting Single Flux Quantum (SFQ) logic technology. This avoids a critical decoding backlog, hidden in all offline decoding schemes, that leads to idle time exponential in the number of T gates in a program [Terhal, 2015].

Under a pure dephasing error model, the proposed accelerator and AQEC solution is able to expand SQV by factors between 3,402 and 11,163 on expected near-term machines. The decoder achieves a 5% accuracy threshold as well as pseudo-thresholds of approximately $5\%, 4.75\%, 4.5\%$, and 3.5% physical error rates for code distances $3, 5, 7$, and 9, respectively. Decoding solutions are achieved in a maximum of ~ 20 nanoseconds on the largest code distances studied. By avoiding the exponential idle time in offline decoders, we achieve a 10x reduction in required code distances to achieve the same logical performance as alternative designs.

This project was a collaboration between scholars at University of Chicago and University of Southern California. The outcome of the project was a research paper in ISCA 2020 conference [Holmes et al., 2020].

(2) DigiQ: A Scalable Digital Controller for Quantum Computers Using SFQ Logic (Chapter 3). Researchers in industry and academia have focused on designing in-fridge classical controllers in order to mitigate the scalability challenges of today's quantum computer prototypes. SFQ logic has the potential to maximize scalability thanks to its ultra-high speed and very low power consumption. However, architecture design for SFQ logic poses challenges due to its unconventional pulse-driven nature and lack of dense memory and logic. Thus, research at the architecture level is essential to guide architects to design SFQ-based classical controllers for large-scale quantum machines.

In this work, we present *DigiQ*, the first system-level design of a Noisy Intermediate Scale Quantum (NISQ)-friendly SFQ-based classical controller. We perform a design space exploration of SFQ-based controllers and co-design the quantum gate decompositions and SFQ-based implementation of those decompositions to find an optimal SFQ-friendly design point that trades area and power for latency and control while ensuring good quantum algorithmic performance. To validate and characterize *DigiQ*, we first implement it using hardware description languages and synthesize it using state-of-the-art/validated SFQ synthesis tools. Our synthesis results show that *DigiQ* can operate within the tight power and area budget of dilution refrigerators at >42,000-qubit scales. Second, we confirm the effectiveness of *DigiQ* in running quantum algorithms by modeling the execution time and fidelity of a variety of NISQ applications.

This project was a collaboration between scholars at University of Chicago and University of Southern California. The outcome of the project was a research paper that will appear in HPCA 2022 conference [Jokar et al., 2022].

(3) Practical implications of SFQ-based two-qubit gates (Chapter 4). Prior work has demonstrated high-fidelity SFQ-based single-qubit gates. However, little research has been done on SFQ-based multi-qubit gates, which are necessary to realize SFQ-based universal quantum computing. In this work, we present the first thorough analysis of SFQ-based two-qubit gates. Our observations show that SFQ-based two-qubit gates tend to have high leakage to qubit non-computational subspace, which presents severe design challenges. We show that despite these challenges, we can realize gates with high fidelity by carefully designing optimal control methods and qubit architectures. We develop optimal control methods that suppress leakage, and also investigate various qubit architectures that reduce the leakage. After carefully engineering our SFQ-friendly quantum system, we show that it can achieve similar gate fidelity and gate time to microwave-based quantum systems. The promising results of this paper show that (1) SFQ-based universal quantum computation is both feasible and effective; and (2) SFQ is a promising approach in designing classical controller for quantum machines because it can increase the scalability while preserving gate fidelity and performance.

This project was a collaboration between scholars at University of Chicago. The outcome of the project was a research paper in QCE 2021 conference [Jokar et al., 2021].

CHAPTER 2

NISQ+: BOOSTING QUANTUM COMPUTING POWER BY APPROXIMATING QUANTUM ERROR CORRECTION

2.1 Introduction

Quantum computing has the potential to revolutionize computing and have massive effects on major industries including agriculture, energy, and materials science by solving computational problems that are intractable with conventional machines [Hastings et al., 2014, Svore and Troyer, 2016]. As we begin to build quantum computing machines of between 50-100 qubits [Preskill, 2018] and larger, design decisions are being made to attempt to get the most computation out of a machine, quantified in this work by expanding the "Simple Quantum Volume" (SQV). SQV can be defined as the number of computational qubits of a machine multiplied by the number of gates we expect to be able to perform without error, as in Fig. 2.1. One limiting factor on SQV now is that physical quantum bits (qubits) are extremely error-prone, which means that computation on these machines is bottlenecked by the short lifetimes of qubits. System designers combat this by attempting to build better physical qubits, but this effort is extremely difficult and classical systems can be used to alleviate the burden. Specifically, *quantum error correction* is a classical control technique that decreases the rate of errors in qubits and expands the SQV. Error correction proceeds by encoding a set of *logical* qubits to be used for algorithms into a set of faulty physical qubits. Information about the current state of the device, called *syndromes*, is extracted by a specific quantum circuit that does not disturb the underlying computation. Decoding is the process by which an error correcting protocol maps this information to a set of *corrections* that, if chosen correctly, should return the system to the correct logical state. Fully fault tolerant machines can expand the SQV rapidly by suppressing qubit errors exponentially with the code distance.

While a fully fault-tolerant quantum computer may take many years to construct, it is

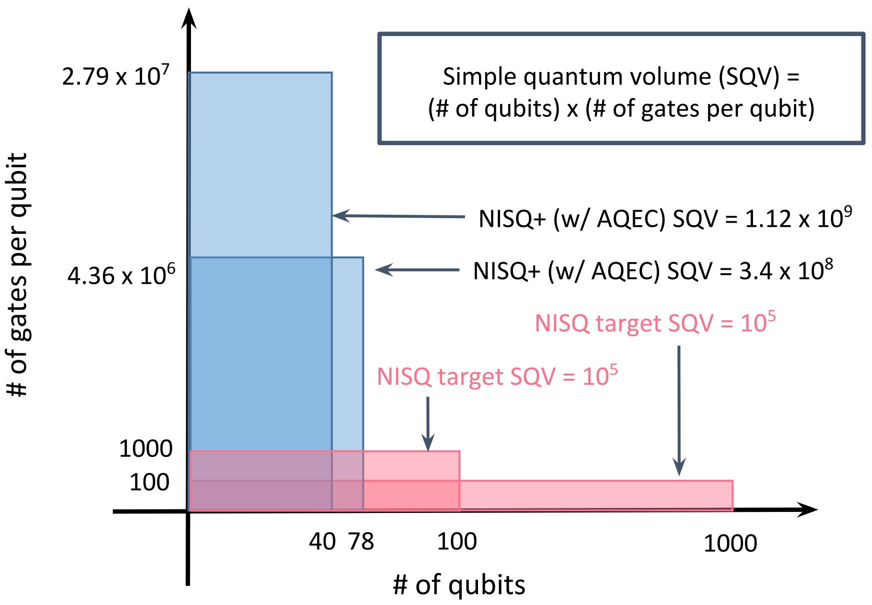


Figure 2.1: Boosting the quantum computation power with approximate error correction schemes. A machine with 1024 faulty physical qubits of error rate 10^{-5} has an SQV of $\approx 10^8$. By performing fast, online, approximate decoding, we can trade the number of computational qubits for gate fidelity and boost the SQV by over a factor of 3,402. Moving to a higher code distance raises this increase to a factor of 11,163. NISQ machines are severely limited by gate fidelity, and introducing error mitigation techniques can have dramatic effects on SQV.

possible to use the well-developed theory of error correction as inspiration for constructing *error mitigation* protocols that still provide a strong expansion in SQV. In this paper we present an approximate decoding solution specifically targeting execution time and show that we can in fact perform decoding at the speed of syndrome generation for near-term machines. Prior work has suggested and analyzed software solutions for decoding, but relying on hardware-software communication can be slow, especially considering the cryogenic environment of typical quantum computing systems. If decoding occurs slower than error information is generated, the system will generate a backlog of information as it waits for decoding to complete, introducing an exponential time overhead that will kill any quantum advantage (see Section 2.3). A hardware solution proposed here results in the ability to perform logical gates with orders of magnitude better fidelity and at the speed of syndrome generation, resulting in a major expansion in SQV as shown in Fig. 2.1. This relies on an approximate decoding algorithm implemented in superconducting Single Flux Quantum (SFQ) hardware. While the algorithmic design enables the accuracy of the hardware accelerator to be competitive at small scale with existing software implementations, the benefits of implementing the circuitry directly in SFQ hardware are numerous. Specifically, high clock speeds, low power dissipation, and unique gating style allows for our accelerator to be co-located with a quantum chip inside a dilution refrigerator, avoiding otherwise high communication costs.

This work contributes the following:

1. We design an approximate decoding algorithm for stabilizer codes based on SFQ hardware, leveraging unique capabilities that the hardware offers,

2. We show that using this new error mitigation technique, we can expand the SQV of near-term machines by factors of between 3,402 and 11,163,

3. We use Monte-Carlo simulation based benchmarking of the hardware accelerator, resulting in effective accuracy and pseudo-thresholds,

4. We perform system execution time analysis, realistically benchmarking the decoder performance in real time and showing that decoding is likely to be able to proceed at or exceeding the speed of data generation enabling the benefits of fault tolerant quantum computing.

5. We show that our online decoder requires 10x smaller code distance than offline decoders when decoding backlog accounted for.

The remainder of the paper is as follows: Section 2.2 describes the necessary background of quantum computation and details the specifications of typical quantum computing systems stacks. Section 2.2.2 describes quantum error correction and the decoding problem in detail. Section 2.4 describes relevant related work in the area ranging from optimized software implementations of matching algorithms to novel descriptions of neural network based decoders. Section 2.5 describes our decoding algorithm, and Section 2.6 describes implementation details of SFQ technology, and the circuit datapaths in detail. Section 2.7 describes our methodology for evaluation, including details of the simulation environment in which our accelerator was benchmarked, details of the metrics used to evaluate performance, and descriptions of novel synthesis tools used to generate efficient layouts of SFQ circuitry. Section 2.8 presents our accuracy results, a breakdown of the accelerator characterization including area, power, and latency footprints, a timing evaluation, and analysis of the SQV effects. Section 2.9 concludes.

2.2 Background

In this section we discuss the basics of quantum computation, quantum error correction, and a description of the fundamental components of a quantum computing system architecture.

2.2.1 Basics of Quantum Computation

Here we provide a brief overview of quantum computation necessary to discuss quantum error correction. For more detailed discussions see [Nielsen and Chuang, 2010]. A quantum computing algorithm is a series of operations on two level quantum states called *qubits*, which are quantum analogues to classical bits. A qubit state can be written mathematically as a superposition of two states as $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, where the coefficients $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. A measured qubit will yield a value of $|0\rangle$ or $|1\rangle$ with probability $|\alpha|^2$ or $|\beta|^2$, respectively, at which point the qubit state will be exactly $|0\rangle$ or $|1\rangle$. Larger quantum systems are represented simply as $|\psi\rangle = \sum_i \alpha_i |i\rangle$ where $|i\rangle$ are computational basis states of the larger quantum system.

Quantum operations (gates) transform qubit states to other qubit states. In this work we will be making use of particular quantum operations known as *Pauli gates*, denoted as $\{I, X, Y, Z\}$. These operations form a basis for all quantum operations that can occur on a single qubit, and therefore any operation can be written as a linear combination of these gates. Additionally, error correction circuits make use of the Hadamard gate H, an operation that constructs an evenly weighted superposition of basis elements when acting on a basis element. Two-qubit controlled operations will also be used, which can generate entanglement between qubits and are required to perform universal computation.

2.2.2 Quantum Error Correction

Qubits are intrinsically fragile quantum systems that require isolation from environmental interactions in order to preserve their values. *Decoherence*, for example the decay of a quantum state from a general state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ to the ground state $|\psi'\rangle = |0\rangle$ happens rapidly in many physical qubit types, often on the order of tens of nanoseconds [Tomita and Svore, 2014, Tannu et al., 2017b]. This places a major constraint on algorithms: without any modifications to the system, algorithms can only run for a small, finite time frame with high probability of success.

To combat this, *quantum error correction* protocols have been developed. These consist of encoding a small number of *logical* qubits used for computation in algorithms into a larger number of physical qubits, resulting in a higher degree of reliability [Lidar and Brun, 2013, Dennis et al., 2002, Fowler et al., 2012a, Terhal, 2015]. In general, developing quantum error correction protocols is difficult as directly measuring the qubits that comprise a system will result in destruction of the data. To avoid this, protocols rely upon indirectly gathering error information via the introduction of extra qubits that interact with the primary set of qubits and are measured. This measurement data is then used to infer the locations of erroneous data qubits.

While many different types of protocols have been developed, this work focuses primarily on the *surface code*, a topological stabilizer code [Gottesman, 1997] that is widely considered to be the best performing code for the medium-term as it relies purely on geometrically local interactions between physical qubits greatly facilitating its fabrication in hardware, and has

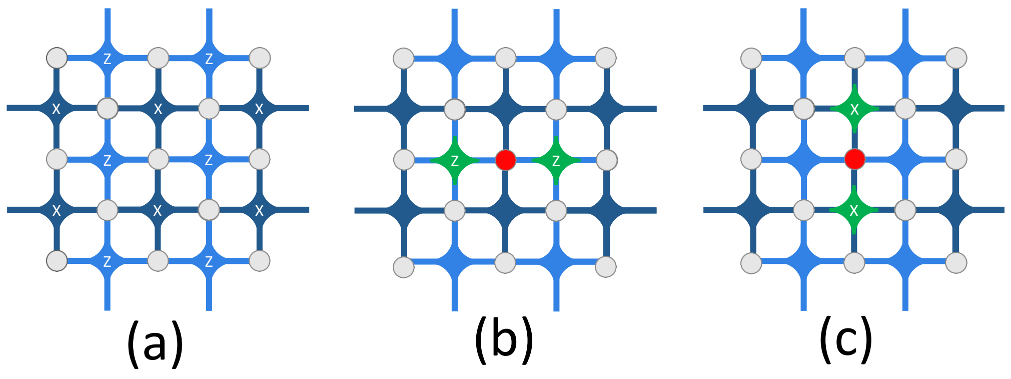


Figure 2.2: Fig. (a) shows a graphical illustration of a surface code mesh. Gray circles indicate data qubits, and nodes labeled X and Z indicate ancillary qubits measuring X and Z stabilizers, respectively. Ancillary qubits are joined by colored edges to the data qubits that they are responsible for measuring. In Fig. (b) a single data qubit experiences a Pauli X error indicated by red coloring, causing the neighboring Z ancillary qubits to detect an odd parity in their data qubit sets and return $+1$ measurement values indicated by green coloring. In Fig. (c), the data qubit in red experiences a Pauli Z error, causing the vertically adjacent X ancillary qubits to return $+1$ measurement values. The entire error syndrome strings for either of these two cases would include a string of 12 values, two of which would be $+1$ and the remaining 10 would be 0.

been shown to have very high reliability overall [Fowler et al., 2012a].

2.2.3 The Surface Code

Errors can occur on physical qubits in a continuous fashion, as each physical qubit is represented mathematically by two complex coefficients that can change values in a continuous range. However, a characteristic of the quantum mechanics leveraged by the surface code is that these continuous errors can be *discretized* into a small set of distinct errors. In particular, the action of the surface code maps these continuous errors into Pauli error operators of the form $\{I, X, Y, Z\}$ occurring on the data. This is one of the main features of the code that allows error detection and correction to proceed.

The surface code procedure that accomplishes error discretization, detection, and correction is an error correcting code that operates upon a two-dimensional lattice of physical qubits. The code designates a subset of the qubits as data qubits responsible for forming the logical qubit, and others as ancillary qubits responsible for detecting the presence of errors in the data. This is shown graphically in Fig. 2.2. Ancillary qubits interact with all of their neighboring data qubits and are then measured, and the measurement outcomes form the *error syndrome*. This set of operations forms the *stabilizer circuit*, where each ancillary qubit measures a four-qubit operator called a *stabilizer*.

Error Detection

The ancillary qubits are partitioned into those denoted as X and Z ancilla qubits. These ancilla qubit sets are sufficient for capturing any Pauli error on the data qubits, as Y operators can be treated as a simultaneous X and Z error. The action of the X stabilizer is two-fold: the four neighboring data qubits are forced into a particular state that discretizes any errors that may have occurred on them. Second, the measurement of the X ancilla qubit signals the parity of the number of errors that have occurred on its four neighbors. For example, it yields a $+1$ value if the state of the four neighboring qubits has an even number of Z errors. The same is true of the Z stabilizers – these track the parity of X errors occurring in the neighboring qubits. If an odd number of errors have occurred in either case, the ancilla qubit measurement will yield a $+1$ value, an event known as a *detection event* [Fowler et al., 2012d], otherwise these will return values of 0 or -1 depending on convention. We will refer to the ancillary qubits returning $+1$ values as *hot syndromes*. The *error syndrome* of the code is a bit string of length equal to the total number of ancilla qubits, and is composed of all of these measurement values.

Decoding is the process of mapping a particular error syndrome string to a set of corrections to be applied on the device. An example of this process is shown graphically in Fig. 2.2. In this example, the hot syndromes generated by a single data qubit error are marked in red. Each single data qubit error causes the adjacent ancillary qubits to return $+1$ values.

A different situation occurs when strings of data qubit errors cross ancillary qubits, as shown in Fig. 2.3. Here, four consecutive data qubits experience errors which generates hot syndrome measurements on the far left and right of the grid. This is because each ancillary

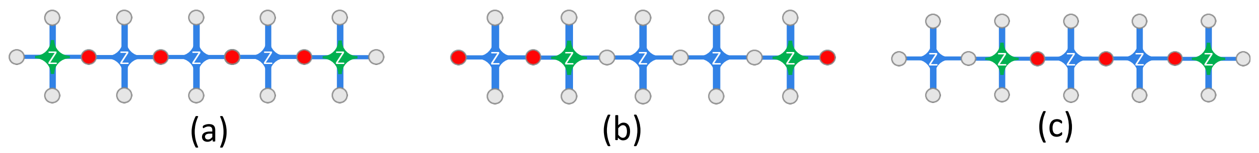


Figure 2.3: This figure shows the errors happened in one row of the surface code grid. Fig. (a) shows a data qubit error pattern spanning across ancillary qubits. Each data qubit experiencing error is indicated in red, and the ancillary qubits returning +1 measurement values are indicated in green. Each ancillary qubit that is adjacent to two erroneous data qubits does not signal the presence of any errors, as the parity of the data qubit sets are still even. This creates an *error string* that runs from the ancillary qubit on the left of the grid to the one on the right. Decoding must map these +1 values to the corresponding set of 4 data qubit errors that generated it. Fig. (b) and Fig. (c) show degeneracy in error syndrome generation by surface code data qubit error patterns. The figures depict two distinct sets of data qubit error patterns that both generate the same error syndromes. Both patterns contain the same number of physical data errors, so these patterns are equally likely assuming independence of errors.

qubit along this chain detects even error parity, so they do not signal the presence of errors. Decoding must be able to pair the two hot syndromes, applying corrections along the chain that connects them.

Error Detection Can Fail

Notice that in Fig. 2.3 (a), if the data qubits on the left and right endpoints of the chain had also experienced errors, none of the ancillary qubits would have detected the chain. This represents a class of undetectable error chains in the code, and specifically occurs when chains cross from one side of the lattice to the other. The result of these chains are physical errors present in the code that cannot be corrected, and are known as *logical errors*, as they have changed the state of the logical qubit. One important characteristic of the surface code is the minimal number of qubits required to form a logical error. This number is referred to as the *code distance, d* of a particular lattice.

2.2.4 Quantum Computing Systems Organization

While qubits are the foundation of a device, a quantum computer must contain many layers of controlling devices in order to interact with qubits. Qubits themselves can be constructed using many different technologies, some of which include superconducting circuits [Linke et al., 2017, Fu et al., 2018, 2017, Barends et al., 2014, Kelly et al., 2015], trapped ions [Maslov, 2017, Linke et al., 2017, Figgatt et al., 2019, Häffner et al., 2005, Lekitsch et al., 2017], and quantum dots [Zajac et al., 2016]. Controlling these devices is often performed by application of electrical signals at microwave frequencies [Chow et al., 2012, Yang et al., 2003, Paraoanu, 2006, Plantenberg et al., 2007].

This work focuses on systems built around qubits that require cryogenic cooling to milliKelvin temperatures [Hornibrook et al., 2015]. These systems require the use of dilution refrigerators, and typical architectures involve classical controllers located in various temperature stages of the system. Such a system is described schematically in [Tannu et al., 2017b, Hornibrook et al., 2015], and presents many design constraints. Controllers inside the refrigerator are subject to area and power dissipation constraints [Patra et al., 2018, Sebastiano et al., 2017]. Communication between stages can be costly as well. Many systems are constructed today using control wiring that scales linearly with the number of qubits, which will prohibit the construction of scalable machines [Franke et al., 2018].

2.2.5 Classical Control in Quantum Computing Systems

Error correction classical processing requires high bandwidth communication of the measurement values of many qubits on the quantum substrate repeatedly throughout the operation of the device, encouraging studies of engineering solutions [Ware et al., 2017], feasibility [Tannu et al., 2017a] and controller design [Tannu et al., 2017b]. Not only are instruction streams primarily dominated by quantum error correction operations [Levy et al., 2009, 2011], but also the classical controller responsible for error correction processing must be tightly coupled to the quantum substrate. If communicating between the quantum substrate and error cor-

	# qubits	# total gates	# T gates
takahashi_adder	40	740	266
barenco_half_dirty_toffoli	39	1224	504
cnu_half_borrowed	37	1156	476
cnx_log_depth	39	629	259
cuccaro_adder	42	821	280

Table 2.1: Characteristics of the simulated benchmarks.

recting controller is subject to excessive latencies, the execution of fault tolerant algorithms will be completely prohibited.

2.3 Motivation: Decoding Must be Fast

Decoding must be done quickly for the surface code to perform well. During actual computation on a surface code error corrected device, there exist gates called T-gates that require knowledge of the current state of errors on the device before they can execute. [1] If decoding is slower than the rate at which syndromes are generated, an algorithm will create a *data backlog*. While the machine is waiting for decoder to process the backlog, more syndrome data is accumulating on the device, which must be processed before executing the subsequent T-gate. Over time, this results in latency overhead that is exponentially dependent upon the number of such gates. Specifically, the overhead scales as $(\frac{r_{\text{gen}}}{r_{\text{proc}}})^k = f^k$, where r_{gen} is the rate of data generation, r_{proc} is the rate of decoder processing, each in bauds, f is the decoding ratio, and k is the number of T gates in the quantum algorithm. An exponentially slow quantum computer eliminates all of its usefulness.

Fig. 2.4 shows the exponential latency overhead due to data backlog. The proof of this is summarized as follows (for more details see [Terhal, 2015]): suppose $f > 1$. This implies that there will be a time t_0 in the application where we encounter a T gate and must wait for syndrome data to be decoded before continuing. Let Δ_{gen} be the amount of time that the machine must stall for processing this data. During this time an additional $D_1 = r_{\text{gen}} \times \Delta_{\text{gen}}$ bits of syndrome data is generated, which can be processed in time $\Delta_{\text{proc}} = r_{\text{gen}}\Delta_{\text{gen}}/r_{\text{proc}} =$

1. Errors commute and can be post-corrected for other gates, but not T-gates.

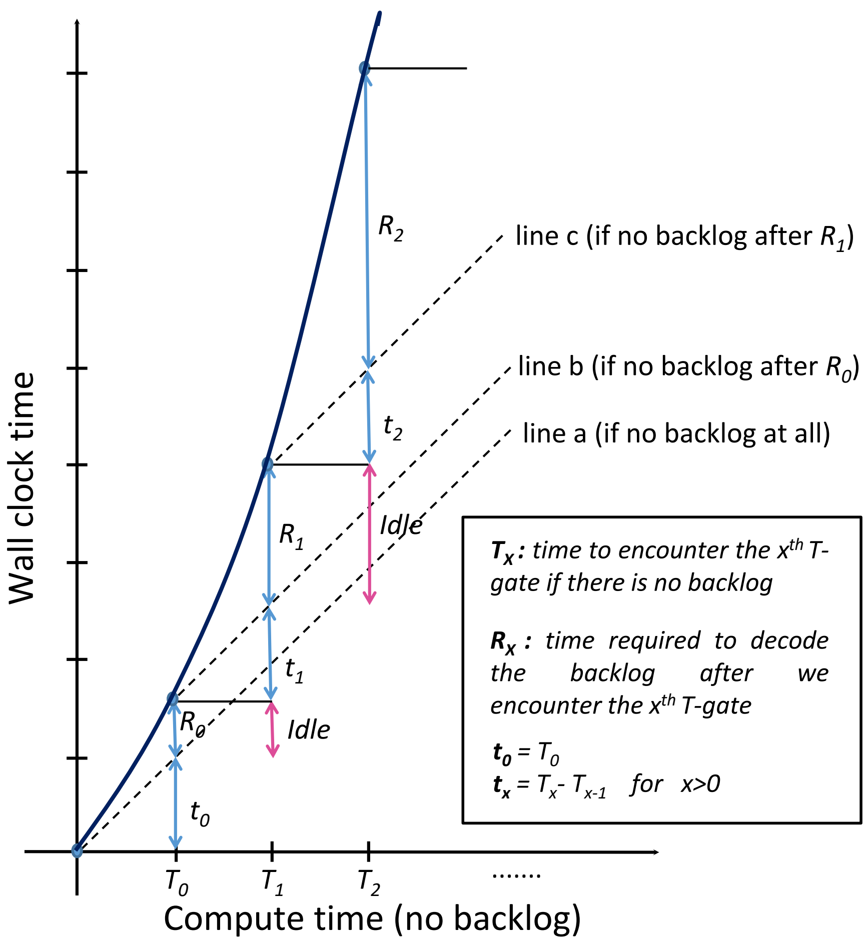

Wall clock time
Compute time (no backlog)
line c (if no backlog after R_1)
line b (if no backlog after R_0)
line a (if no backlog at all)
R_2
t_2
R_1
Idle
t_1
R_0
Idle
t_0
T_0
T_1
T_2
....
T_x: time to encounter the x^{th} T-gate if there is no backlog
R_x : time required to decode the backlog after we encounter the x^{th} T-gate
$t_0 = T_0$
$t_x = T_x - T_{x-1}$ for $x>0$

Figure 2.4: Exponential latency overhead when $f = (\frac{r_{\text{gen}}}{r_{\text{proc}}}) > 1$. X-axis shows the compute time if there is no backlog and y-axis shows the actual wall clock time; if there is no backlog we expect wall clock time to be the same as the compute time (line a). Every time we encounter a T-gate we need to decode all the syndromes up until that gate before we can continue the execution [Terhal, 2015]. When we encounter the first T-gate at time T_0, we need to finish the decoding of the data generated during t_0 (not all the data is already decoded as decoding rate is slower than data generation rate) and it takes R_0 to do that. During R_0 where our quantum system is idle, more syndromes are generated and when we encounter the second T-gate at $T_1 + R_0$, we need to finish decoding those syndromes in addition to the syndromes generated during t_1 before continuing the program execution. **The syndrome data generated during the idle periods is the key reason behind data backlog creation which leads to exponential latency overhead.**

$f\Delta_{\text{gen}}$. The backlog problem begins to be noticeable at this point, where during processing of the first block D_1, we generate a *new block* $D_2 = r_{\text{gen}} \times \Delta_{\text{proc}} = fD_1 > D1$ in size. Then, at the next T gate this process repeats, and we again generate a block of data of size $D_3 = fD_2 = f^2D_1$ bits. Hence, by the k'th T gate, we generate an overhead of f^kD_1 bits

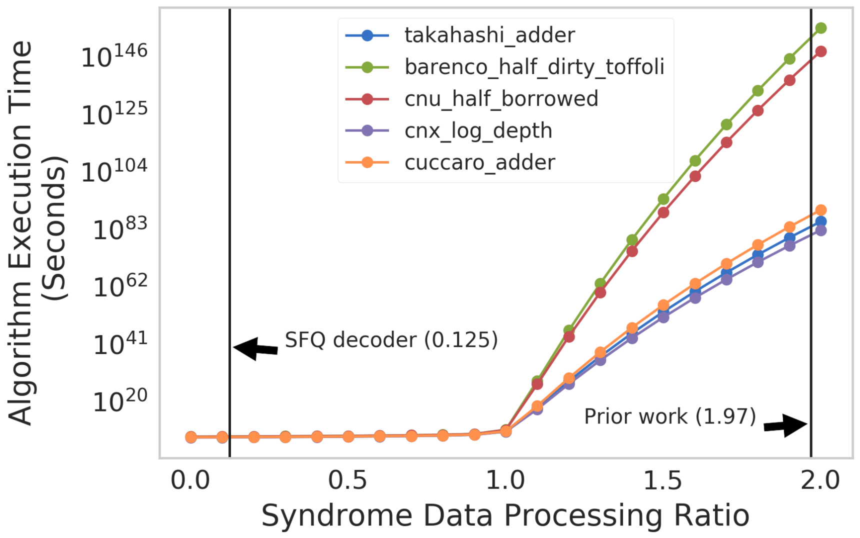


Figure 2.5: Running times of fault tolerant quantum algorithms with decoders of varying efficiency. The X-axis plots $\frac{r_{\text{gen}}}{r_{\text{proc}}}$. To the left of 1, data is processed as fast as it is generated, whereas rates to the right of 1 indicate that the decoder is slower than syndrome data is generated. The T-gates require synchronization with the decoder in order to execute. Prior work [Chamberland and Ronagh, 2018] claims that fast neural network inference decoders can perform inference in $\sim$ 800 ns, which places the decoder at approximately the 1.5 - 2 region for a system generating syndromes in the 400-500ns range. Our decoding results show that time to solution never exceeds 20ns, placing it below 1. Clearly computation becomes intractable quickly for slow decoders.

to process, exponential in *the decoder's performance ratio.*

As a specific example, consider a multiply-controlled NOT operation on 100 logical qubits from [Holmes et al., 2018]. This algorithm contains $\sim$ 2356 gates, of which 686 are T-gates after decomposition. Assuming that a syndrome generation cycle time is approximately 400 ns [Ghosh et al., 2012], and the best prior decoder requires 800 ns to execute [Chamberland and Ronagh, 2018], the ratio $(r_{\text{gen}}/r_{\text{proc}}) = 2$, and the execution time is intractable.

Fig. 2.5 shows a simulation of real quantum subroutines each composed of a different number of T gates as denoted in Table 2.1. The exponential overhead scaling shows that as decoders become slower than the rate at which data is being generated (which occurs for "syndrome data processing ratios" over 1), the overheads quickly become intractable. Regardless of the effectiveness of the decoder, if it operates at a processing ratio higher than 1 then it will impose exponentially high latency overheads on algorithm execution. The algorithms all draw inspiration from [Barenco et al., 1995]. Barenco-half-dirty-Toffoli is a logarithmic depth multi-control Toffoli gate using O(n) ancilla bits. It performs the same computation as the "cnx-log-depth" gate with a different circuit. The "cnu-half-borrowed" gives an implementation of a multi-control Toffoli using O(n) dirty ancilla, meaning the initial states of these bits does not need to be known. The Cuccaro adder is a linear depth implementation of a reversible A + B adder, i.e. two registers of the specified length added together. It has a carry in and a carry out bit as well. The Takahashi adder is an optimized version of the Cuccaro adder [Takahashi et al., 2009].

This is the primary motivation for this work – the hardware decoder must be able to execute faster than syndrome data are generated as a prerequisite for tractable fault tolerant computation.

2.4 Related Work

Early work focused on the development of and modifications to the minimum weight perfect matching algorithm (MWPM) [Edmonds, 1965b,a] to adapt it to surface code decoding [Fowler et al., 2012c,b]. This resulted in a claimed constant time algorithm after parallelization [Fowler, 2013].

Other work has constructed maximum likelihood decoders (MLD) based on tensor network contraction [Bravyi et al., 2014]. This work is computationally more expensive than minimum-weight perfect matching, but is more accurate.

Neural networks have been explored as possible solutions to the decoding problem as well [Varsamopoulos et al., 2019a, 2017, 2019b, 2018, Chamberland and Ronagh, 2018, Varsamopoulos et al., 2017, 2018, Baireuther et al., 2019, Torlai and Melko, 2017]. Feed-forward neural networks and recurrent neural networks have been explored in combination with lookup tables to form decoders. The primary distinguishing factor in these systems is that the networks function as *high level decoders* in that they predict both a sequence of error corrections on data qubits along with the existence of a logical error. In this sense, they operate at a higher level than both the MWPM and MLD decoders, seemingly at the cost of execution time with respect to training complexity.

Lastly, more customized algorithms have been developed specifically targeting the surface code decoding problem, including renormalization group decoders [Duclos-Cianci and Poulin, 2010b], Union-Find decoding [Delfosse and Zémor, 2017, Delfosse and Nickerson, 2017], and others [Wootton, 2015, Duclos-Cianci and Poulin, 2010a].

The primary distinguishing factor of our work is that the decoder design is guided by practical system performance. Accuracy has been sacrificed in order to achieve quantum advantage. While the proposed decoder design may not achieve logical error suppression at the same order as some other algorithms, the ability to perform the algorithm in SFQ hardware at or exceeding the speed of syndrome generation is achieved, as is satisfaction of system design constraints.

2.5 Decoder Overview and Design

In this section we describe prior decoding algorithms, followed by details of our approximate decoding algorithm, and demonstrate how we make efficient use of unique features of SFQ gates to implement the algorithm in hardware.

2.5.1 Prior Decoding Algorithms and their Practical Implications

Different error chains can cause the same error syndrome. The decoding problem requires that the maximally likely set of error chains be reported as a solution, given a particular error syndrome. This can be formulated as a matching problem. Specifically, given an error syndrome, we can construct a complete graph on vertices associated with each ancillary qubit that reported an error, and calculate the weight of each edge in the graph based on the Manhattan distance between its vertices on the original surface code grid, which corresponds to the shortest path between its vertices on the grid. The goal is to find the

Algorithm 1: Minimum Weight Matching Decoder Algorithm

```
Input: List of hot syndromes and their $(x, y)$ coordinates on the surface code grid.
Output: Set of error chains.
1 Construct a complete graph $G = (V, E)$ on vertices associated with hot syndromes.
2 Calculate the weight $w_{e_i}$ for each $e_i \subset E$ that connects $v_{i_1}$ and $v_{i_2}$ vertices based on
    their Manhattan distance; $w_{e_i} = |x_{i_1} - x_{i_2}| + |y_{i_1} - y_{i_2}|$, where $(x_{i_1}, y_{i_1})$ and
    $(x_{i_2}, y_{i_2})$ are the coordinates of $v_{i_1}$ and $v_{i_2}$ vertices, respectively.
3 Find a perfect matching $M \subset E$ such that $\Sigma_{e_i \in M} w_{e_i}$ is minimum.
4 Return $M$ as the set of error chains.
```

Algorithm 2: Union-Find Decoder Algorithm

```
Input: List of hot syndromes and their $(x, y)$ coordinates on the surface code grid.
Output: Set of error chains.
1 Create a cluster for each hot syndrome. These clusters are odd as each includes an
    odd number of hot syndromes (i.e., 1).
2 Initialize the coordinates of each cluster's up, down, left, and right borders on the
    grid to the coordinate of the cluster's hot syndrome.
3 While there exist an odd cluster:
    (I) Grow all the odd clusters in all directions on the grid by half edge, and update
    the coordinates of the borders.
    (II) If odd clusters meet, merge them and update the number of hot syndromes in
    the new cluster.
    (III) Remove the clusters that have an even number of hot syndromes.
4 Construct a spanning forrest, and apply the peeling decoder to find the set of error
    chains [Delfosse and Nickerson, 2017].
```

maximally likely pairing of the syndromes using these weights, which can be done by solving the MWPM problem; given our error model, the likelihood of a specific path between two vertices being the correct set of errors decreases exponentially with the length of the path, thus the shortest path has the maximum likelihood. See Algorithm 1 for the pseudocode of the MWPM decoder. This decoder is effective and provides high accuracy threshold, but it has a worst case time complexity of $O(n^3)$ [Delfosse and Nickerson, 2017], which is too slow for large code distances [Das et al., 2020]. Thus, prior work proposed faster decoders.

Union-Find is an example of such decoders that uses the Union-Find data structure to find a set of error chains. Union-Find decoder has a worst case time complexity of $O(n\alpha(n))$ where $\alpha(n) \leq 3$ for all practical purposes [Delfosse and Nickerson, 2017], while it decreases the accuracy threshold by 0.4% compared to MWPM decoder. See Algorithm 2 for the pseudocode of the Union-Find decoder. Although Union-Find decoder achieves almost-linear time complexity, the decoding time is still longer than the syndrome generation time ($> 2X$ longer), which leads to exponential latency overhead (see Sec. 2.3). Thus, we need to further speed-up the decoding algorithm.

So far, we discussed that latency is one of the main criteria for the decoding algorithms. However, for the room temperature decoders, bandwidth between the quantum chip and the decoder is another main criteria. Such decoders may need to *continuously* (every syndrome generation cycle) receive the syndrome information from the quantum chip and perform the decoding, which leads to significant scalability challenges. Note that today's quantum computers (which are not fault tolerant) already have scalability challenges as they rely on a classical controller to receive control pulses (see Chapter 3), and adding a room temperature decoder only worsen the situation. Thus, we need to develop in-fridge decoders.

SFQ is a promising logic technology to implement in-fridge decoders due to its unique characteristics such as very low power consumption and ultra-high speed. An SFQ-based decoder not only reduces the aforementioned bandwidth requirements significantly but also can perform the decoding with low latency thanks to high SFQ clock speed. The next question is: can we implement either of MWPM or Union-Find decoders using SFQ logic? For a quantum system with 1000 logical qubits and code distance of 11, MWPM and Union-Find decoders requires $>$ 1.5 MB and $>$ 2.5 MB memory capacity, respectively [Das et al., 2020], and the required memory capacity increases with the code distance and the number of logical qubits. However, one limitation of SFQ technology is the lack of a dense memory (memory density is $\sim$ 400 Kb/cm^2 based on optimistic estimations [Tannu et al., 2017b]), which makes the implementation of MWPM and Union-Find decoders infeasible at large scales. This suggests that we need to modify the decoding algorithms based on the characteristics of SFQ technology.

2.5.2 A Greedy Approach

Our decoding algorithm is based upon a greedy approximation to the minimum weight matching problem. The algorithm calculates all Manhattan distances $d(v_i, v_j)$ between vertices and sorts them in ascending order $d_1, d_2, ..., d_{k'}$ where $k' = \binom{k}{2}$. All of the corresponding probability weights are calculated, transforming this ordering to a descending order of likelihood. Then, for each edge e in descending order, add e to the solution M if it forms a matching. This means that it adds another two distinct vertices into M that were not already present. To account for boundary conditions, we introduce a set of external nodes connected to the appropriate sides of the lattice, and connected to one another with weight 0. Under this formulation, the algorithm is a 2-approximation of the optimal solution [Drake and Hougardy, 2003].

2.5.3 SFQ-Based Decoder

In this section, we introduce the functional design of our SFQ-based decoder and give some rational for each aspect of its design. As a reminder, Single Flux Quantum is classical logic implemented in superconducting hardware that does not perform any quantum computation. It is a medium used to express our classical algorithm. The decoder is placed above the quantum chip layer; it receives measurement results from ancillary qubits as input, and returns a set of corrections as output. For scalability, our decoder design is built out of a two dimensional array of modules implemented in SFQ logic circuits that we refer to as *decoder modules*. These are connected in a rectilinear mesh topology. Modules are identical and there is one module per each data and ancillary qubit, denoted as *data qubit modules* and *ancilla qubit modules*, respectively. Each decoder module has one input called the *hot syndrome input* that comes from the measurement outcome of the physical quantum bits and determines if the module corresponds to a hot syndrome (note that this input can be "1" only for ancilla qubit modules). Each module contains one output called the *error output* that determines if the module is contained in the error chain (this output can be "1" for

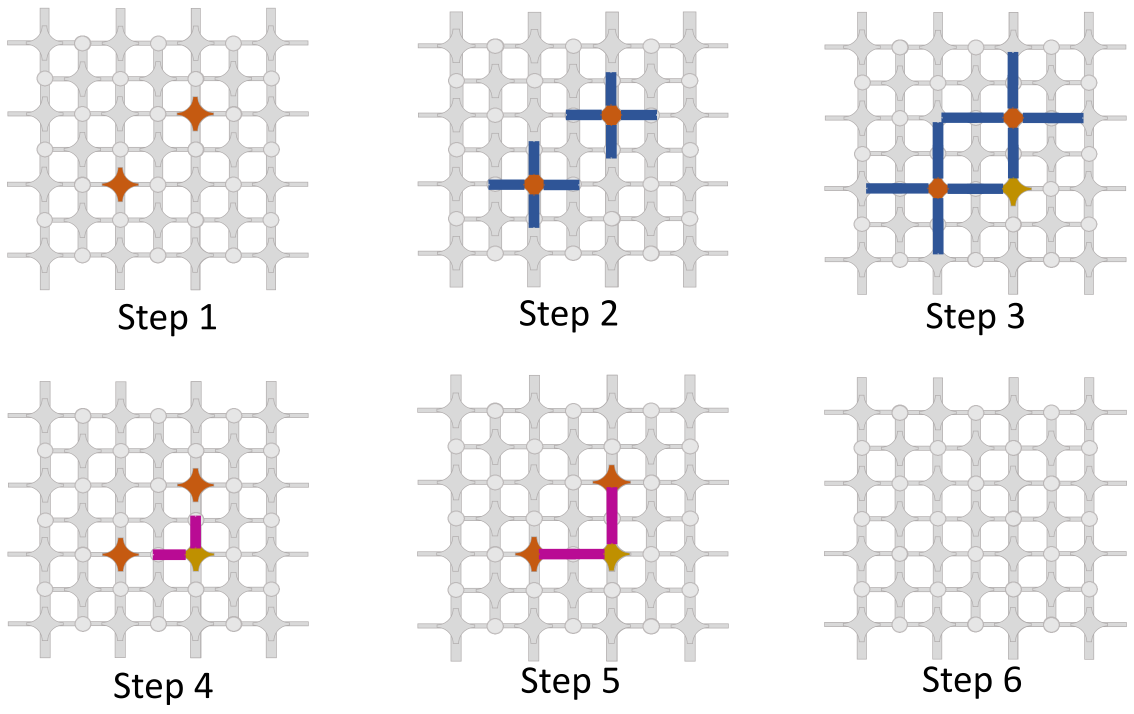


Figure 2.6: Baseline solution to find the two closest hot syndrome modules. Step1: two decoder modules have "1" hot syndrome input. Step2: the hot syndrome modules propagate grow signals. Step3: the grow signals meet at an intermediate module. Step4: the intermediate module sends pair signals in the opposite direction. Step5: pair signals arrive at the hot syndrome modules. Step6: decoding is complete. Note that the decoder modules that receive a pair signal are considered as part of the error chain that has occurred.

all of the decoder modules). In addition, each module has connections to adjacent modules (left, right, up and down).

Our approximate decoder algorithm proceeds as follows. First, the algorithm finds the two modules with "1" hot syndrome input, called *hot syndrome modules*, that are closest together. Next, the algorithm reports the chain of modules connecting them as the correction chain. Finally, it resets the hot syndrome input of the two modules and searches for the next two closest hot syndrome modules. The decoder continues this process until no module with "1" hot syndrome input exists. This is graphically displayed in Fig. 2.6.

Baseline Solution: Our baseline design finds the two closest hot syndrome modules as shown in Fig. 2.6 as follows: 1) every hot syndrome module sends *grow* signals to all the adjacent modules in all four directions; each adjacent module propagates the grow signal

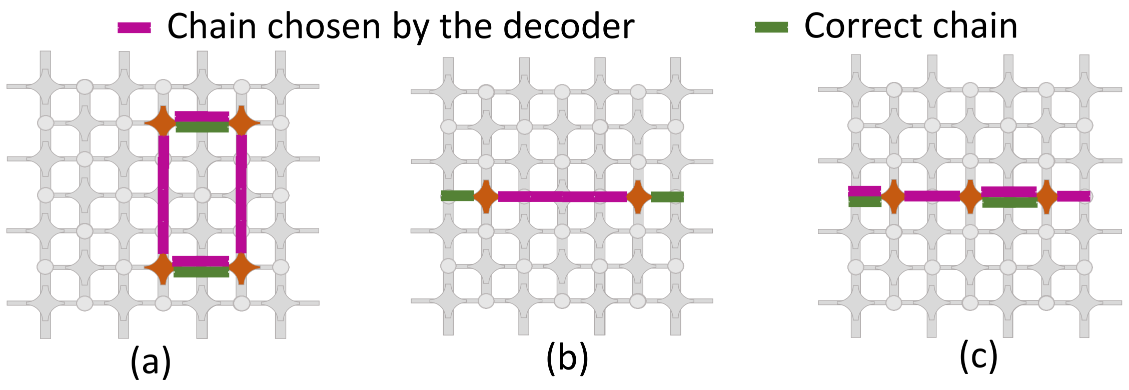


Figure 2.7: Scenarios where the SFQ decoder chooses the wrong chain where (a) no reset/boundary/equidistant mechanisms are employed, (b) no boundary/equidistant mechanisms are employed, and (c) no equidistant mechanism is employed.

in the same direction. Grow signals propagate one step at each cycle. 2) When two grow signals intersect at an *intermediate module*, we generate a set of *pair* signals and back-propagate these to their hot syndrome origins. All of the decoder modules that receive pair signals are part of the error chain. Note that more than one intermediate module might exist, however, only one of them is effective and sends the pair signals. For example, in Fig. 2.6, two intermediate modules receive the grow signals, and the decoder is hardwired to be effective (ineffective) when it receives grow signals from up and left directions (down and right directions). Intermediate module refers to the effective one. The baseline solution does not show accuracy or pseudo-threshold behavior and demonstrates poor logical error rate suppression, see the incremental results presented in Section 2.8 in Fig. 2.11.

Reset Mechanism: One flaw of the baseline system is the lack of a mechanism to reset the decoder modules after two hot syndrome modules are paired. Grow signals of the paired modules continue to propagate, potentially causing these modules to pair incorrectly with other hot syndrome modules, ultimately resulting in an incorrect error chain reported. Fig. 2.7 (a) shows an incorrect matching due to this behavior. To mitigate this, we add a reset mechanism that resets the decoder modules each time hot syndrome modules are paired and the error chain connecting them is determined. Adding the reset mechanism to the baseline system improves the performance somewhat, but does not yet achieve tolerable accuracy.

Boundary Mechanism: Another explanation for the low performance of the baseline solution is that it never pairs hot syndrome modules with boundaries. For example, if two hot syndrome modules are far from each other but are close to boundaries, the error chain with the maximum likelihood is the one that connects the hot syndrome modules to the boundaries. Fig. 2.7 (b) shows this behavior occurring on a machine. We implement a mechanism that enables pairing the hot syndrome modules with boundaries. To do this, we add decoder modules that surround the surface boundaries called *boundary module* (one per each quantum bit located at a boundary). Our solution treats boundary modules as hot syndrome modules but they do not grow and can pair only with non-boundary modules. Note that when two modules are paired, the hot syndrome input of only the non-boundary modules is reset; boundary modules are always treated as hot syndrome modules. Adding the boundary mechanism to the baseline solution augmented with the reset mechanism further increases the accuracy of the decoder.

Equidistant Mechanism: Finally, the last major reason for inefficiency of the baseline is that it does not properly handle the scenarios in which multiple hot syndrome modules are spaced within equal distances of one another, resulting in a set of pairs that are all equally likely. The baseline solution augmented with reset and boundary mechanisms works properly only if no non-boundary hot syndrome module has an equal distance to more than one other hot syndrome module; otherwise the solution pairs it with all the hot syndrome modules with equal distance. However, this is not the desired output. We need a more intelligent solution to break the tie in the aforementioned scenario, and pair the hot syndrome module to only one other module. This is shown in Fig. 2.7 (c).

To resolve these equidistant degenerate solution sets, we introduce a request – grant policy that allows for the hardware to choose specific subsets of these pairs to proceed. 1) Similar to the baseline solution, the non-boundary hot syndromes first propagate grow signals. 2) An intermediate module receives two grow signals from two different directions, and it sends *pair_request* signals in the opposite directions. Pair_request signals continue to propagate until they arrive at a module with "1" hot syndrome input. 3) The modules with "1" hot syndrome input send *pair_grant* signals in the opposite direction of the received pair_request signals. Note that multiple pair_request signals might arrive at a module with "1" hot syndrome at the same time, but it gives grant to only one of them. 4) An intermediate module receives pair_grant signals from two different directions and sends pair signals in the opposite directions. 5) Pair signals continue to propagate until they arrive at a module with "1" hot syndrome input. Boundary modules do not send grow signals but they send pair_request signals when they receive grow signals; they also send pair signals when they receive pair_grant signals.

2.6 Implementation

2.6.1 SFQ Implementation of Greedy Decoding

SFQ is a magnetic pulse-based fabric with switching delay of *1ps* and energy consumption of 10^{-19}J per switching. In addition, availability of superconducting microstrip transmission lines in this technology makes it possible to transmit picosecond waves with half of speed of light and without dispersion or attenuation. The combination of these properties together with fast two-terminal Josephson junctions, makes this technology suitable for high speed processing of digital information [Volkmann et al., 2013, Kirichenko et al., 2011, Herr et al., 2011, Takeuchi et al., 2013, Likharev and Semenov, 1991]. SFQ logic families are divided into two groups: ac-biased and dc-biased; Reciprocal Quantum Logic (RQL) [Herr et al., 2011], and Adiabatic Quantum Flux Parametron (AQFP) [Takeuchi et al., 2013] are in the first group, and Rapid Single Flux Quantum (RSFQ) [Likharev and Semenov, 1991], Energy-efficient RSFQ (ERSFQ) [Kirichenko et al., 2011], and energy-efficient SFQ (eSFQ) [Volkmann et al., 2013] are examples of the second group. The dc-biased logic family with higher operation speed (as high as 770GHz for a T-Flip Flop (TFF) [Chen et al., 1999]) and less bias supply issues are more popular than ac-biased logic family.

Our algorithm requires modules to propagate signals one step at each cycle. One approach to implement our algorithm is to use synchronous elements such as flip-flops in decoder modules. However, standard CMOS style flip-flops are very expensive in SFQ logic (e.g., one D-Flip-Flop occupies 72.4× more area and consumes 117× more power compared to a 2-input AND gate). On the other hand, SFQ gates have a unique feature that we utilize to implement our algorithm without flip-flops. Unlike CMOS gates, most of the SFQ gates (expect for mergers, splitters, TFFs, and I/Os) require a clock signal to operate [Pasandi et al., 2018]. Thus, we do not need to have flip-flops and signals can propagate one SFQ gate at each cycle.

As described earlier, our decoder requires resetting the decoder modules each time two hot syndrome modules are paired. We have a global wire that passes through all the modules and is connected to each module using splitter gates. Thus, if we set the value of the global wire, all of the decoder modules receive the reset signal at the same time, as the splitter gates do not require clock signals to operate. If a module receives a reset signal, it blocks the module inputs using 2-input AND gates (one input is *module_input* and the other input is $\bar{Reset}$). In order to reset a decoder module completely, we need to block the module inputs for as many cycles as the depth of our SFQ-based decoder because the SFQ gates work with clock cycles and one level of gates is reset at each cycle. Thus, we use a simple circuit to keep the reset signal "1" for as many cycles as the circuit depth. In each module, we pass the reset signal that comes from the global wire to a set of m cascaded buffer gates where m is the circuit depth, and the module inputs are blocked if the reset signal that comes from the global wire is "1" or at least one of the buffers has "1" output.

2.6.2 Datapath and Subcircuit Design

Fig. 2.8 shows an overview of our decoder module microarchitecture. Our decoder consists of five main subcircuits.

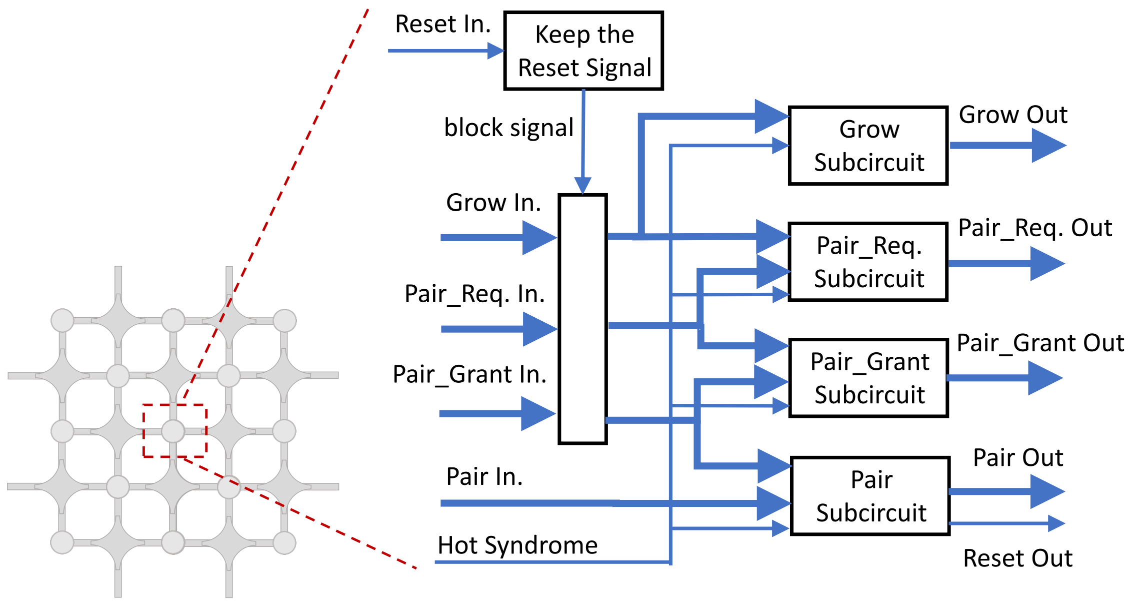


Figure 2.8: Overview of decoder module microarchitecture.

Grow Subcircuit: this subcircuit receives hot syndrome input and 4 grow inputs (from 4 different directions), and produces 4 grow output signals. Grow outputs are "1" if the hot syndrome input is "1" or if the module is passing a grow signal generated by another module.

Pair_Req Subcircuit: this subcircuit is responsible for setting the value of pair_request outputs which are "1" if two grow signals meet at an intermediate module or if the module is passing a pair_request signal that arrived at one of its input ports. The module does not pass the pair_request input signal if the hot syndrome input is "1"; in that case, the module generates a pair_grant signal instead.

Pair_Grant Subcircuit: this module determines the value of pair_grant outputs which are "1" if the module is a hot syndrome module and gives grant to a pair_request signal, or if the module is passing a pair_grant input signal to the adjacent module.

Pair Subcircuit: this subcircuit sets the value of pair outputs which are "1" if two pair_grant signals meet at an intermediate module or if a pair input signal is "1" and the hot syndrome input is not "1". If both the pair input and hot syndrome input are "1", the

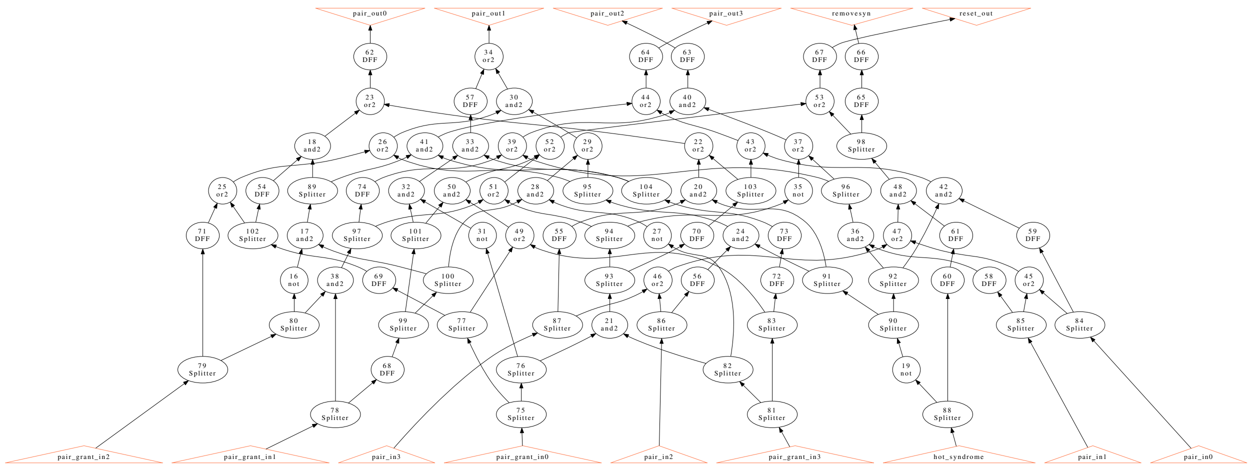


Figure 2.9: Pair subcircuit after SFQ specific optimizations and mapping. Triangular shapes at the bottom represent the primary inputs of the circuit and those at the top of the circuit show primary outputs. *DFF* is SFQ DRO DFF inserted for path balancing. Splitter (balanced) trees are also shown. Splitter is an asynchronous SFQ gate that receives a pulse at its input and after its intrinsic delay, it produces two almost identical output pulses. We insert splitters at the output of an SFQ gate (or a primary input) with more than one fanout.

module does not pass the pair signal and instead generates a global reset signal that reset all of the decoder modules and also resets the hot syndrome input. Note that the reset signal resets everything except the subcircuit responsible for passing the pair signals because it is possible that the intermediate module does not have equal distance from the paired hot syndrome modules and we do not want to stop the propagation of all the pair signals in the system when the closer module receives a pair signal (while the farther module has not received a pair signal yet). The SFQ implementation of this subcircuit is shown in Fig. 2.9.

Reset Subcircuit: this subcircuit is responsible to keep the reset signal “1” for as many cycles as the depth of our circuits. The depth is 5 in our circuits, thus reset subcircuit blocks grow, pair_req and pair_grant inputs for 5 cycles in order to reset the module.

2.7 Methodology

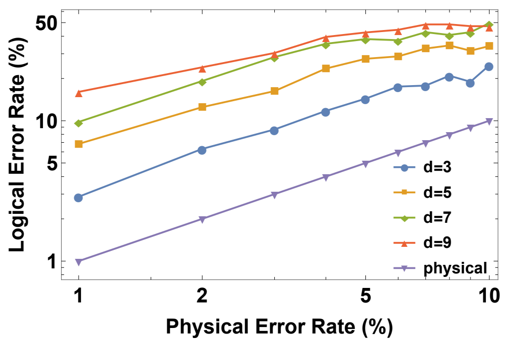
Simulation Techniques: In order to effectively benchmark the performance of a stabilizer quantum error correcting code, techniques must be used to simulate the action of the code over many cycles. This is referred to elsewhere in literature as *lifetime simulation* [Varsamopoulos et al., 2018], or simply Monte Carlo benchmarking. We constructed a simulation environment that simulates the action of the stabilizer circuits. A cycle refers to one full iteration of the stabilizer circuit. At each step within the cycle, errors are stochastically injected into the qubits and propagated through the circuits. Ancillary qubits are measured, and the outcomes are reported in the error syndrome. This syndrome is then communicated directly to the decoder simulator, which returns the corresponding correction. The correction is applied and the surface is checked for a logical error. The ratio of the number of logical errors to the number of cycles run in simulation is used as the primary performance metric.

Evaluation Performance Metrics: In our evaluations, we use the stabilizer circuits as the primary benchmark. These circuits are replicated for every ancillary qubit present in a surface code lattice. Many different lattices are also analyzed, ranging in size from code distances 3 to 9.

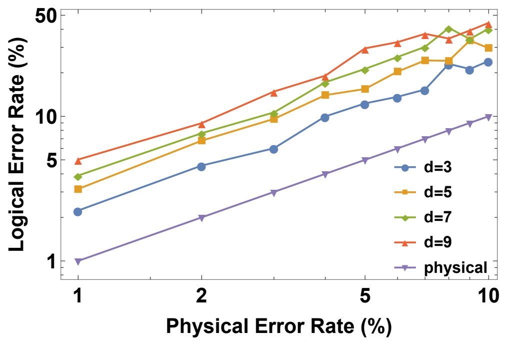
As performance metrics, we focus on *accuracy thresholds* and *pseudo-thresholds*. The former is the physical error rate at which the code begins to suppress errors effectively across multiple code distances. Below this threshold, the logical error rate P_L decreases as the code distance d increases. Above threshold these relationships invert, and P_L grows with d due to decoder performance: the presence of many errors causes the decoding problem to become too complex. In many cases, this leads to corrections that complete what would have otherwise been short error chains, forming logical errors, a process that amplifies as code distances increase.

Pseudo-threshold refers to the performance of a single code distance, and is the physical error rate at which the logical error rate is equal to the physical rate, i.e. $P_L = p$. This can be (and often is) different across different code distances. Better error correcting codes will have higher pseudo-threshold values, as well as higher accuracy thresholds.

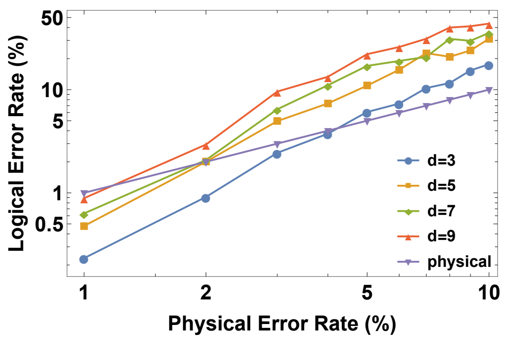
Error Models: The Monte Carlo simulation environment requires a model of the errors



Baseline design



Adding resets

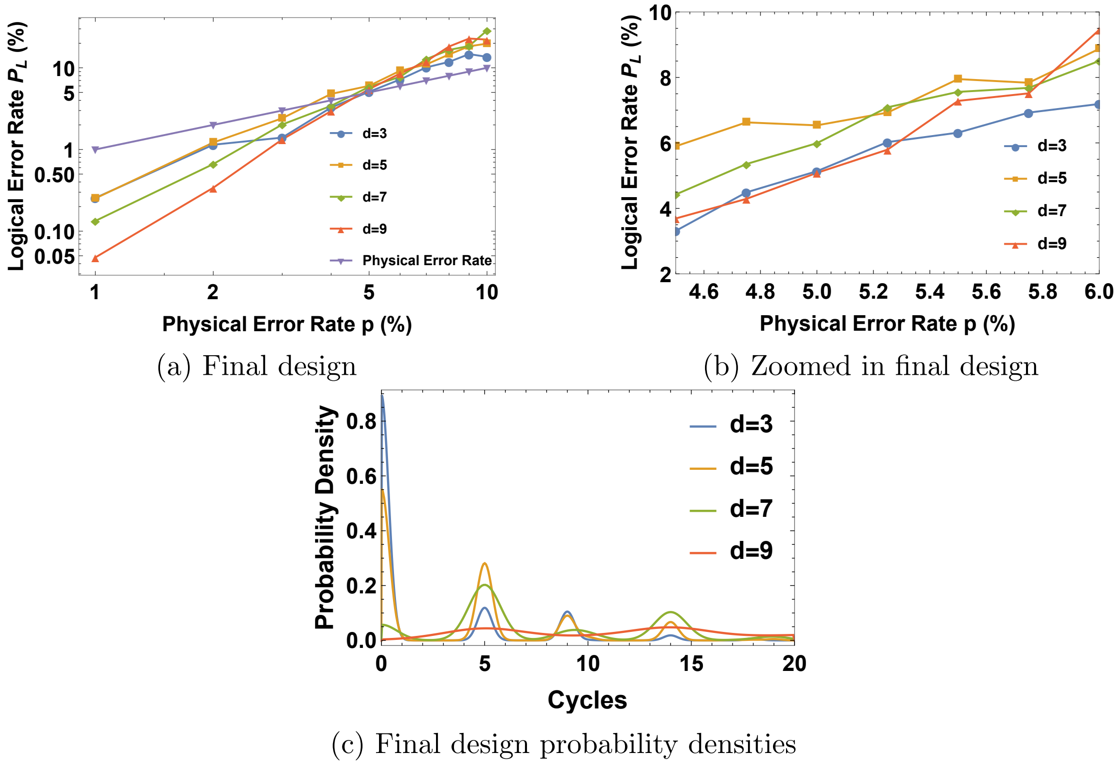


Adding resets and boundaries

Figure 2.10: Logical error rate performance of each incremental design step. The addition of resets and boundaries each contribute heavily to the realization of pseudo-thresholds, and have a dramatic effect on reducing the minimum achievable logical error rates for each code distance.

on the quantum system. We choose to focus on the *depolarizing channel* model [Nielsen and Chuang, 2010, Fowler et al., 2012a, Lidar and Brun, 2013, Terhal, 2015], parameterized by a single value p: Pauli X, Y, and Z errors occur on qubits with probability $p/3$. During simulation Pauli errors are sampled i.i.d for injection on each data qubit. We present analysis of a variation of the model, the *pure dephasing channel* [Delfosse and Nickerson, 2017, Delfosse and Zémor, 2017] comprised solely of Pauli Z errors occurring on qubits with probability p. The decoder will be operated symmetrically for both X and Z errors, allowing for simple extrapolation from these results.

Single Flux Quantum Circuit Synthesis: An ERSFQ library of cells is used in this paper to reduce the total power consumption (including the static and dynamic) of the surface code decoder as much as possible. Table 2.2 lists characteristics of this library. As



(a) Final design

(b) Zoomed in final design

(c) Final design probability densities

Figure 2.11: Results for our final design, including support for reset, boundary, and equidistant mechanisms. (a) Error rate scaling for the proposed decoder. An accuracy threshold is evident at approximately 5% physical error rate, while pseudo-thresholds span the range from $\sim 3.5\% - 5\%$. (b) Logical error rates near the 5% physical error rate value. (c) Truncated unnormalized estimated probability distributions for the execution cycles required by each code distance in simulation. Window shows up to 20 cycles for comparison across code distances. Notice that while distances $3, 5, 7$ display peaks centered at $0, 5, 9$, and 14 cycles.

seen, this library contains four logic gates including AND2, OR2, XOR2, and NOT, and it has a Destructive Read-Out D-Flip-Flop (DRO DFF) cell. Area of all logic cells are the same and it is equal to 3500 μm^2. However, area of the DRO DFF is less than the area of these gates (3000 μm^2). DRO DFFs are different from standard CMOS style flip-flips: they are specially designed for SFQ circuits and are usually used for path balancing. In Table 2.2, the total number of Josephson junctions (as a measure of complexity and cost) used in designing each gate together with the intrinsic delay of each cell is reported.

The decoder circuit and its sub-circuits are synthesized by employing ERSFQ specific logic synthesis algorithms and tools [Pasandi and Pedram, 2019b, Pasandi et al., 2018, Pasandi and Pedram, 2019d,a]. These algorithms are designed to reduce the complexity

Cell	Area (μm^2)	JJ Count	Delay (ps)
AND2	3500	16	8.7
OR2	3500	14	6.0
XOR2	3500	18	6.3
NOT	3500	12	13.0
DRO DFF	3000	11	6.8

Table 2.2: The library of ERSFQ cells and corresponding characteristics used for synthesizing the circuit into SFQ hardware. Josephson Junction count is listed in the second column.

of the final synthesized and mapped circuits in terms of total area and Josephson junction count. This is achieved by reducing the required *path balancing* DFF count for realizing these circuits. Please note that for correct operation of dc-biased SFQ (including ERSFQ) circuits, these circuits should be fully path balanced; this means that in a Directed Acyclic Graph (DAG) that represents an SFQ circuit, length of any path from any primary input to any primary output in terms of the gate count should be the same. In most of the SFQ circuits this property does not hold in the beginning. Therefore, some path balancing DFFs should be inserted into shorter paths to maintain the full path balancing property. In the algorithms we employed for mapping these circuits, a dynamic programming approach is used to ensure minimization of the total number of DFFs to maintain the balancing property [Pasandi and Pedram, 2019b,d]. In addition, a depth minimization algorithm together with path balancing is employed [Pasandi et al., 2018] to reduce the logical depth (length of the longest path from any primary input to any primary output in terms of the gate count) of the final mapped circuit. This helps to reduce the latency of the mapped SFQ circuit. As mentioned before, SFQ logic gates are pulsed-based, meaning that the presence of a pulse represents a logic-“1” and the absence of a pulse represents a logic-“0”. Each gate is clocked, and as an example, the SFQ NOT gate behaves as follows. After the clock pulse arrives, when there are no input pulses, a pulse is generated at the output of the gate representing a “1”. On the other hand, when there is an input pulse, no pulses are generated at the output, meaning a “0”. Each pulse is a single quantum of magnetic flux ($\phi_0 = \frac{h}{2e} = 2.07\text{mV}\times\text{ps}$) [Likharev and Semenov, 1991]. To simulate the SFQ circuits for

Circuit	Logical Depth	Latency (ps)	Total Area (μm^2)	Power Consumption (μW)
AND_GATE	1	8.7	3500	0.026
OR_GATE	1	6.0	3500	0.026
OR_GATE_7_INPUTS	3	18.0	33000	0.338
NOT_GATE	1	13.0	3500	0.026
Pair_Grant Subcircuit	5	115.0	293500	3.38
Pair Subcircuit	5	115.0	303500	3.53
Pair_Req./Grow Subcircuit	5	115.0	406500	4.75
Full_Circuit	6	168.0	1143000	13.44

Table 2.3: Experimental synthesis results for the SFQ Decoder. Shown are all gates utilized in the synthesis, as well as submodules that comprise the main circuit. Pair_Req. and Grow subcircuits have been combined into a single subcircuit.

Code Distance	Max	Average	Standard Deviation
3	3.86	0.29	0.59
5	9.58	0.74	1.13
7	14.7	2.06	2.05
9	19.8	3.93	3.21

Table 2.4: Decoder execution time in nanoseconds across each code distance studied and across all simulated error rates.

verifying their correct functionality, we use the Josephson simulator (JSIM) [Delport et al., 2019].

2.8 Evaluations

In this section we evaluate the performance of our proposed decoder design, both in terms of circuit characteristics including power, area, and latency, as well as error correction performance metrics of accuracy and pseudo-thresholds. We also analyze the execution time of our system, relying upon described operating assumptions and circuit synthesis results.

Threshold Evaluations: To gauge the performance of our design, we use the threshold metrics described in Section 2.7. Fig. 2.11 (a) shows the central performance result, while the top row of Fig. 2.11 shows the effect of all of the incremental design decisions on the overall performance. This evaluation simulates the performance of the decoder across a range of physical error rates. A pseudo-threshold range of between 3.5% and 5% is observed, and an accuracy threshold appears at approximately the 5% error rate. For code distance 5, the pseudo-threshold is below the accuracy threshold. This highlights the difference between

Code Distance	3	5	7	9
c_2	0.650	0.429	0.306	0.323

Table 2.5: Empirical parameter estimation given a model of the form $P_L \approx c_1(p/p_{\text{th}})^{c_2 \cdot d}$. Shown are estimated c_2 parameter values.

these metrics – an error correcting protocol like the surface code can perform well even though particular code distances may still be amplifying the physical error rates (i.e. $P_L > p$). It is important to consider both types of thresholds when evaluating decoder performance.

An interesting behavior is observed for code distance $d = 3$. This lattice performs at or surpassing the performance of all other lattices from the 3% physical error rate and above. Below this point, the lattice begins to taper off, and ultimately it converges with the distance 5 lattice. Boundary conditions were highly prioritized in our design, causing this effect. In particular, the decoder is designed such that error chains that terminate at the boundaries are more likely to be correctly identified than other patterns. This choice was made as smaller lattices are more dominated by these edge effects than larger lattices. The smallest lattice in our simulations shows this anomalous behavior, as it contains a disproportionate amount of boundary patterns. In larger lattices, syndromes are less likely to terminate in boundaries, reducing this effect.

Fig. 2.11 (b) highlights the desired threshold behavior. Examining the 6% error rate, code distance 9 is outperformed by code distance 7. Moving to the lowest physical error rate in the window, we find that the lattices perform in the order $d = 9$, $d = 3$, $d = 7$, and $d = 5$, ordered from lowest to highest logical error rate. Barring the anomalous $d = 3$ behavior described above, this is accuracy threshold behavior indicative of successful error correction performance.

Performance Analysis: To quantify the approximation factor of our design we compare the performance to that of an ideal decoder by fitting to an exponential analytical model. The achievable error rates by the surface code ideally can be described by $P_L \approx 0.03(p/p_{\text{th}})^d$ [Fowler et al., 2012a] when a minimum weight matching decoder is used in software. This model is valid for the error models we consider, as [Fowler et al., 2012a] uses and fits "class-0"

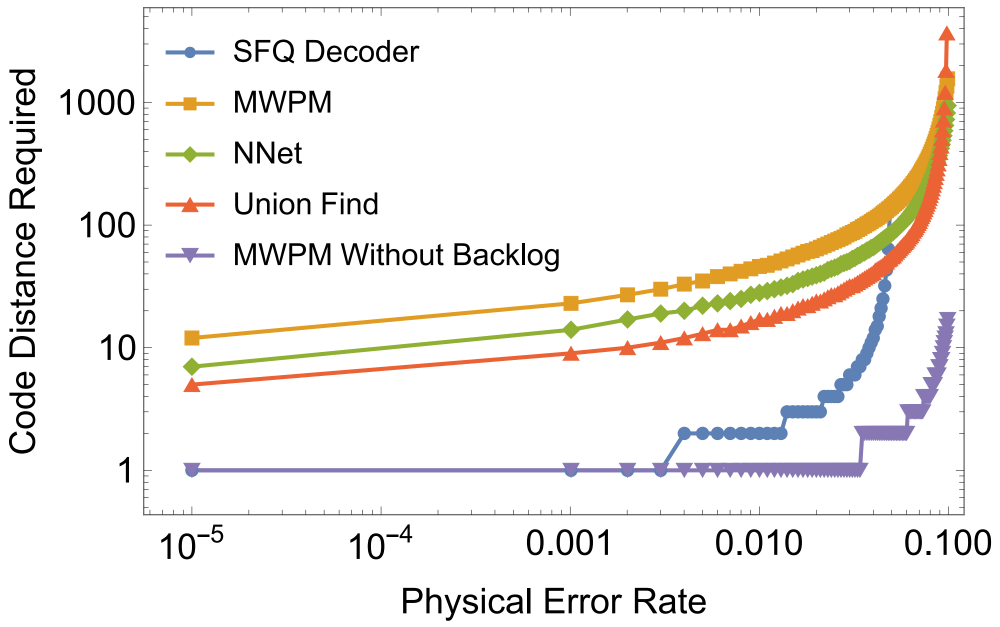


Figure 2.12: Comparison of required code distances of different decoders to execute an algorithm consisting of 100 T-gates. Compared are the SFQ Decoder, MWPM decoder [Fowler et al., 2012a], neural network decoder [Baireuther et al., 2019], Union-Find decoder [Delfosse and Nickerson, 2017], and a theoretical MWPM decoder with no backlog. across both code distances and physical error rates.

Pauli errors in the same fashion. Using a model of the form $P_L \approx c_1(p/p_{\text{th}})^{c_2 \cdot d}$, we fit values of c_1, c_2 for each code distance at physical error rates below accuracy threshold, and collect c_2 values in Table 2.5. C_2 coefficients describe the *effective code distance* for our system, and capture the approximation factor we introduce. For code distances 3 and 5, we find that the approximate decoder is roughly 65% and 43% of the optimal distance respectively. This is the trade-off made by our system in order to fit the timing and physical footprint requirements of the system.

Notice that this accuracy tradeoff results a net resource reduction for our design over other proposed designs as shown in Fig. 2.12. The data backlog imposes delays into the system that decrease the logical accuracy of any decoder that incurs this backlog. As the backlog builds up, the number of required syndrome detection cycles builds up as well, resulting in a new effective logical error rate as one logical gate now requires many more syndrome detection cycles to occur. The SFQ decoder pays an accuracy price for speed, but when the backlog is taken into consideration this tradeoff results in a significant performance gain over alternative designs.

Synthesis Results and Circuit Characterization: Table 2.3 shows experimental results for the surface code decoder circuit presented in this paper using the aforementioned ERSFQ library of cells described in Section 2.7. The full circuit demonstrates a cycle latency of 168 ps, and an area and power footprint of 1.143 mm^2 and 13.44 μW, respectively. The full decoder is comprised of a mesh of these circuit modules, requiring a single module per individual qubit. This means that for systems of code distance 9 comprised of 289 qubits, the decoder required will be of size 330.33 mm^2 and will dissipate 3.88 mW of power. Typical dilution refrigerators are capable of cooling up to $1-2$ Watts of power in the 4-Kelvin temperature region [Hornibrook et al., 2015], enabling the co-location of a decoder mesh of size 87×87, which would protect a single qubit of code distance $d = 44$, or 100 qubits of code distance $d = 5$. These values are estimations given modern day SFQ and cryogenic dilution refrigerator technology, much of which is subject to change in the future.

Execution Time Evaluation: The most important characteristic that the SFQ decoder aims to optimize is real-time execution speed. Previous works have described the syndrome generation time to be between $160-800$ ns for superconducting devices that we are focusing on in this study. [Ghosh et al., 2012, Tannu et al., 2017b].

In practice the time to solution is much lower than the upper bound of O(n) on the greedy algorithm. Table 2.4 contains the empirically observed statistics of our decoder operation. The maximum cycles to solution is well approximated by a linear scaling with a leading coefficient of ~ 15.75. Estimated probability distributions describing the required cycles to solution for each code distance are shown in Fig. 2.11 (c).

Comparison to existing approximation techniques: Trading the accuracy for decoding speed has been utilized in prior work. Union-Find [Delfosse and Nickerson, 2017] achieves a significant speed-up over the minimum weight perfect matching algorithm, while

	MWPM	Union-Find	SFQ decoder
Accuracy Threshold	10.3%	9.9%	5%

Table 2.6: Comparing decoding algorithms in terms of accuracy threshold. SFQ decoder sacrifices accuracy for speed in order to avoid exponential latency overhead and achieve quantum advantage.

the accuracy threshold decreases by only 0.4%. Despite this, the Union-Find decoding time is still longer than the syndrome generation time ($> 2X$ longer) thus exposing it to the exponential latency overhead caused by the data backlog. In contrast to prior approximation techniques, decoding time in our design is faster than syndrome generation time and thus it does not incur exponential latency overhead. However, the accuracy threshold is lower in our design (see Table 2.6), which is the price we pay in order to enable practical implementation of error-correcting quantum machines.

Effect on SQV: The net effect of our design is to expand the SQV achievable by near-term machines. An example of this is a small 100 physical qubit system in which 10^3 gates are performed per qubit, a machine that is conjectured will exist in the near future [Bishop et al., 2017] and shown in red in Fig. 2.1. By utilizing the scaling equation described in Section 2.8, we see that a homogeneous system of 78 logical qubits each of code distance 3 is capable of performing $\sim 4.36 \times 10^6$ gates per qubit. This expands SQV from $100 \times 10^3 \rightarrow 78 \times (4.36 \times 10^6) \approx 3.4 \times 10^8$, increasing by a factor of 3402. This can be pushed farther by going to the small qubit count limit, constructing a machine of 40 logical qubits each of code distance 5, yielding SQV of $\approx 1.12 \times 10^9$, an increase of $11,163$. These effects are captured in Fig. 2.1. Not all applications benefit from these expansions in the same fashion, but our techniques allow for machines to be used in ways that are tailored to individual applications, and enable much more computation to be performed on the same machine.

2.9 Conclusion

In the design of near-term quantum computers, it is vital to enable the machines to perform as much computation as possible. By taking inspiration from quantum error correction, we have designed an “Approximate Quantum Error Correction” error mitigation technique that expands the “Simple Quantum Volume” of near-term machines by factors between 3,402 and 11,163. Our design focuses on the construction of an approximate surface code decoder that can rapidly decode error syndrome, at the cost of accuracy.

Using SFQ synthesis tools, we show that the area and power are within the typical cryogenic cooling system budget. In addition, our accelerator is based on a modular, scalable architecture that uses one decoder module per each qubit. Most importantly, our decoder constructs solutions in real-time, requiring a maximum of $\sim$20 ns to compute the solution in simulation. This allows our decoding accelerator to achieve 10x smaller code distance when compared to offline decoders when accounting for decoding backlog. Thus, it is a technique that can effectively boost the Simple Quantum Volume of near-term machines.

CHAPTER 3

DIGIQ: A SCALABLE DIGITAL CONTROLLER FOR QUANTUM COMPUTERS USING SFQ LOGIC

3.1 Introduction

Superconducting quantum computing is one of the promising technologies for building a quantum computer [Brink et al., 2018, Li et al., 2020], with many prototypes having been manufactured in the recent years [Arute et al., 2019, Kelly, 2018, Steffen et al., 2011, Fu et al., 2017]. However, today's prototypes rely on sending separate analog microwave control pulses for each qubit through coaxial cables from a classical controller at room temperature to the quantum chip inside the dilution refrigerator (see Fig. 3.1(a)). This design is simple and straightforward, however presents severe scalability challenges due to the massive costs of generating/routing the analog microwave signals, and significant heat dissipation at millikelvin temperatures due to using a large number of high bandwidth coaxial cables [McDermott et al., 2018, Leonard et al., 2019, Li et al., 2019]. Thus, it is essential to build compact controllers as close as possible to quantum chips in order to generate and route the control signals locally and address the scalability problem of today's systems (see Fig. 3.1(b) for an example of such a controller).

Cryo-CMOS is an excellent near-term solution to increase the scalability of today's quantum machines, and controller prototypes based on Cryo-CMOS have been manufactured [Van Dijk et al., 2020] which can scale to >800 qubits (see Sec. 3.3). Meanwhile, Superconducting Single Flux Quantum (SFQ) logic [Kirichenko et al., 2011, Likharev and Semenov, 1991] is an emerging classical logic family and is recognized as a promising solution to maximize the scalability of in-fridge controllers due to its unique characteristics such as ultra-high speed and very low power [Leonard et al., 2019, McDermott et al., 2018, Li et al., 2019, Liebermann and Wilhelm, 2016]. However, a key remaining step is designing an SFQ-friendly controller architecture that operates within the tight power and area budget of

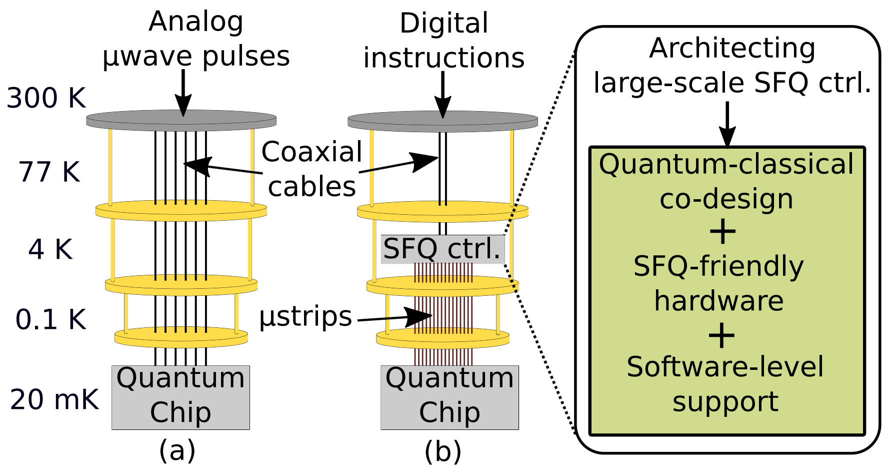


Figure 3.1: (a) Today's controller design: controller at room temperature, (b) *DigiQ*: controller close to quantum chip.

dilution refrigerators at large scales, while ensuring good quantum algorithmic performance.

Prior work has demonstrated quantum gates based on SFQ logic [Leonard et al., 2019, Li et al., 2019, Liebermann and Wilhelm, 2016], and has outlined a vision for an SFQ-based controller for fault-tolerant quantum computing that relies on repeated streaming of quantum instructions that are stored in SFQ registers [McDermott et al., 2018]. These studies done by physicists are essential in order to show the feasibility of performing SFQ-based quantum gates, and envision the potentials of SFQ logic in controlling large-scale quantum chips. However, the following architectural shortcomings remain to be addressed: (1) prior work does not consider scenarios where we are no longer primarily repeating the same quantum instructions. Thus, they are especially not suitable for running Noisy Intermediate Scale Quantum (NISQ) algorithms; (2) there has not been a detailed synthesis of a complete SFQ-based controller architecture in order to get an accurate estimate of power and area, and determine the scale we can achieve with such controllers given the power and area budget of dilution refrigerators; (3) there has not been an analysis of the implications of SFQ-based quantum controllers on quantum algorithmic performance. This analysis is necessary to assess the cost and effectiveness of SFQ-based controller designs; (4) there has not been an analysis on the robustness of SFQ-based controllers to the qubit imperfections in NISQ systems.

In this paper, we address the above architecture-level shortcomings and present *DigiQ*, the first system-level design of a scalable NISQ-friendly SFQ-based classical controller. Inspired by the SuperNPU paper [Ishida et al., 2020], which demonstrates architecture design for SFQ-based neural processing units, our paper guides architects to design SFQ-based controller architectures for large-scale quantum computers.

Architecture design for SFQ logic is different from that of CMOS logic, due to its unconventional pulse-driven nature and lack of dense memory/logic. In addition, implementation of quantum gates using SFQ pulses is different from that of microwave pulses. Thus, novel SFQ-friendly controller architecture designs are essential. We perform a design space exploration of SFQ-based controllers and co-design the quantum gate decompositions and SFQ-pulse implementation of those decompositions to ensure that our design both works within the tight power and area budget of dilution refrigerators at large scales and provides good algorithmic performance.

Quantum gate parallelism is essential to preserve good quantum algorithmic performance in many NISQ applications [Heckey et al., 2015]. Our quantum-classical co-design demonstrates that due to the lack of dense memory/logic in SFQ and tight power and area budget of dilution refrigerators, we cannot afford to simultaneously send tailored quantum gates to many qubits at large scales. The implementation of quantum algorithms with significant gate parallelism therefore requires sharing SFQ-based quantum instructions among multiple qubits (i.e., single instruction, multiple data (SIMD)). Getting good algorithmic performance on a SIMD NISQ architecture is especially challenging because of qubit variations and frequency drift, which typically require gates to be uniquely calibrated for each qubit. We propose novel software-level solutions to address these challenges and preserve SIMD parallelism.

To validate and characterize *DigiQ*, we first work out the details of quantum program execution flow, starting from our compiler at room temperature and ending with sending the control signals to qubits. Then, we implement our complete design in hardware description languages, and synthesize it using state-of-the-art/validated SFQ synthesis tools [Fourie et al., 2019, Pasandi and Pedram, 2019b,c, Shahsavani et al., 2017] to measure power, area, and delay values. Our synthesis results are obtained post-layout based on accurate extraction of the gate layouts and passive transmission lines and subsequently simulated using well-established tools for superconductive electronic applications [Fourie et al., 2019, Whiteley Research Inc., Fang, 1989], and are thus highly accurate and reliable. Finally, we show the effectiveness of *DigiQ* in running quantum algorithms by compiling a variety of NISQ algorithms for our system, and modeling the resulting execution time and fidelity.

We position ourselves as addressing the physical challenges in scaling up the NISQ machines. Our work is complementary to Perfect Intermediate Scale Quantum computing (PISQ) [Bertels et al., 2021] approach, which suggests developing new quantum algorithms assuming perfect qubits and evaluating them with classical simulators. PISQ is a great approach since it separates the development of quantum algorithms and applications from hardware and architectural research and allows researchers to focus on the development of new quantum algorithms in various scientific fields.

To summarize, our key contributions are as follows:

- **Architecting SFQ-based controllers:** We guide architects to design large-scale SFQ-based controllers by taking into consideration the opportunities (efficient on-chip broadcast and ultra-fast clock) and challenges (lack of dense memory/logic) of SFQ.

- **Quantum-classical co-design:** By co-designing quantum gate decompositions and SFQ-based implementation of those decompositions, we present *DigiQ*, an SFQ-friendly SIMD controller architecture.

- **Addressing the SIMD calibration challenges:** We present software solutions to address the quantum gate calibration challenges of SIMD hardware to make it robust to imperfections in qubit hardware.

- **Characterizing controller hardware:** We implement *DigiQ* in hardware description languages and show its feasibility in large scales in terms of power, area, and delay using state-of-the-art/validated SFQ synthesis tools.
- **Characterizing algorithmic performance:** We confirm the effectiveness of *DigiQ* in running quantum algorithms by compiling a variety of NISQ applications for *DigiQ* and modeling their execution time and fidelity.

The rest of the paper is organized as follows. Sec. 3.2 provides a background on quantum controllers, SFQ logic, and discusses the opportunities and challenges of SFQ-based controllers. We present related work in Sec. 3.3. Sec. 3.4 demonstrates the details of *DigiQ*, our scalable digital SFQ-based quantum controller architecture. Sec. 3.5 discusses the quantum gate calibration challenges of *DigiQ*, and presents software-level solutions to address them. Sec. 3.6 shows our methodology and thorough evaluation of *DigiQ*. Finally, Sec. 3.7 concludes the paper and discusses the future work.

3.2 Background and Motivation

Here we provide a background on quantum computing followed by a discussion of quantum gates/controllers in existing systems and their limitations. We then present a background on SFQ logic and discuss the opportunities and challenges of SFQ-based controllers.

3.2.1 Quantum computing

A quantum algorithm specifies a series of transformations on quantum systems called qubits, which are analogous information carriers to classical bits. The state of a single qubit can be represented as a linear combination of two states:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

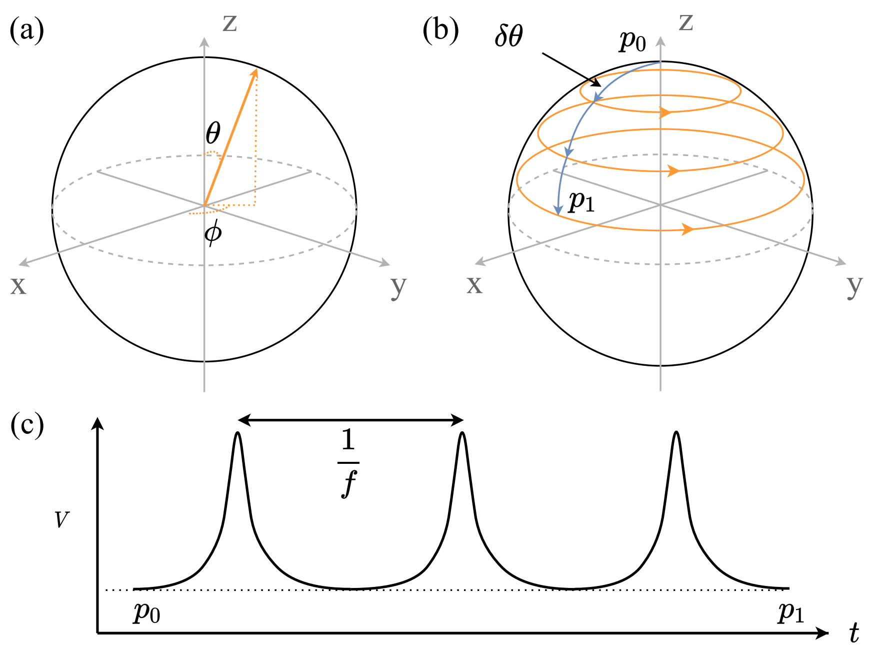


Figure 3.2: (a) Bloch sphere representation of a qubit; (b) SFQ driven trajectory. The blue trajectory is driven by the periodic SFQ pulse train shown in (c), and the orange trajectory is driven by the qubit free evolution; (c) SFQ pulse train in the time domain. f is the qubit oscillation frequency.

where the amplitudes $\alpha, \beta \in \mathbb{C}$ satisfy $|\alpha|^2 + |\beta|^2 = 1$. It is useful to visualize the state of a single qubit as a vector on the unit Bloch sphere as shown in Fig. 3.2(a), where $\alpha = \cos\theta/2$ and $\beta = e^{i\phi}\sin\theta/2$. A multi-qubit state may be written as $|\psi\rangle = \sum_i \alpha_i |i\rangle$, where $\sum_i |\alpha_i|^2 = 1$ and $|i\rangle = |i_{n-1}\rangle \dots |i_1\rangle |i_0\rangle$ are the computational basis states of the n-qubit quantum system.

Quantum gates are unitary operators which modify the state of the qubit system. Any single-qubit gate can be represented as a rotation on the unit Bloch sphere (see Fig. 3.2(a)). Rotations around any two axes can be combined to perform arbitrary single-qubit gates. Combined with a two-qubit *entangling* gate (i.e., a gate which cannot be decomposed into one-qubit gates), this is sufficient for universal quantum computation [Barenco et al., 1995].

3.2.2 Superconducting qubit controllers and their limitations

Superconducting qubits are nonlinear LC circuits built with Josephson junctions (JJ) that operate at near absolute zero temperature and oscillate at microwave frequencies. Qubits are defined using the quantized ground ($|0\rangle$) and first excited ($|1\rangle$) states of the oscillator, where the qubit's oscillation frequency is defined as the energy difference between the two levels. *Transmons* are a simple form of superconducting qubit comprising a JJ and a shunt capacitor, designed to reduce the qubit's sensitivity to electrical charge noise [Koch et al., 2007], and is the qubit technology in many state-of-the-art systems [Arute et al., 2019]. *Flux-tunable* transmons allow tuning the qubit oscillation frequency, and are implemented by replacing the transmon's JJ with a pair of parallel junctions separated by a small distance. The qubit frequency can then be shifted by driving an external magnetic flux (using a small electrical current, $\sim$1 mA) through the enclosed loop [Krantz et al., 2019]. The relationship between frequency and flux can be fine tuned by varying the parameters of each JJ individually, creating what's known as an *asymmetric* transmon [Hutchings et al., 2017].

Superconducting quantum computer prototypes perform single-qubit gates by driving transitions between the oscillator's $|0\rangle$ and $|1\rangle$ states using microwave pulses. Controllable multi-qubit operations require a means of selectively interacting qubits through some coupling architecture. Two-qubit gates such as CZ can be implemented using flux-tunable transmons by changing the frequency of the qubits temporarily such that their quantum states are on resonance [Krantz et al., 2019].

Superconducting quantum computers have received significant attention due to their convenient qubit design and configurability [Krantz et al., 2019], leading to rapid advances in their size, coherence time, and gate fidelity [Arute et al., 2019, Kelly, 2018, Steffen et al., 2011]. However, they also pose significant challenges which must be addressed for superconducting quantum computing to be scalable. Superconducting qubits need cooling to very low temperatures ($\sim$20 mK) which is expensive and complicates control hardware. Further, unlike technologies such as trapped ion (in which qubit uniformity is guaranteed by nature

[Brown et al., 2016]), the characteristics of superconducting qubits are subject to variation and drifts over time.

The controller design in today's prototypes relies on separate cables from room temperature to control individual qubits [Arute et al., 2019]. This approach has severe scalability issues. First, there is a massive electronics cost for generating and routing the analog control pulses from room temperature including the cost of arbitrary waveform generators (AWGs), microwave sources, IQ mixers (which modulate the in-phase and out-of-phase components of the drive signal [Krantz et al., 2019]), and amplifiers and attenuators that are used for thermal property matching at each stage of the fridge [Leonard et al., 2019, Krantz et al., 2019, McDermott et al., 2018]. Second, the cooling power at millikelvin stage of the fridge is very limited (<10 μW [Hornibrook et al., 2015]) and a large number of high bandwidth coaxial cables create a big heat load problem at this stage. Thus, alternative quantum controller architectures are needed to realize scalable and robust quantum machines.

3.2.3 Opportunities and challenges of SFQ quantum controllers

Superconducting Single Flux Quantum (SFQ) is a magnetic-pulse based device and a single quantum of magnetic flux, $\Phi_0 = h/2e = 2.07$ mV·ps, is used for logic bit representation. In this representation, presence of a pulse has the meaning of "logic-1", while absence of a pulse is considered as a "logic-0". Operation of SFQ logic is based on overdamped JJs [Herr et al., 2011, Takeuchi et al., 2013, Likharev and Semenov, 1991, Kirichenko et al., 2011, Volkmann et al., 2013]. There are two types of D-Flip-Flops (DFFs) in this technology: Destructive Read Out (DRO) DFF and Non-Destructive Read Out (NDRO) DFFs. In the first type the stored data will be erased after one read operation and the DFF will be reset back to its zero state. However, in the second type, multiple read operations can be done without erasing the content of DFF. SFQ devices with switching delay of 1 ps and switching energy of 10^{-19} J are considered great candidates to provide high speed solutions post-silicon and post-CMOS [Ishida et al., 2020]. Moreover, these Niobium-based devices are extremely low power and despite their cryo-cooling overhead they still consume significantly less power compared to the state-of-the-art silicon-based devices [Mukhanov, 2011]. These unique properties make SFQ an attractive logic technology to implement classical controllers with maximized scalability for quantum computers.

SFQ can be utilized not only to implement classical controller logic, but also to perform quantum gates locally. Prior work demonstrated that SFQ pulse trains can be utilized to perform single-qubit gates [McDermott and Vavilov, 2014, Liebermann and Wilhelm, 2016, Leonard et al., 2019]. For example, an intuitive approach to do rotations along the y-axis, $Ry(\theta)$, is to apply one SFQ pulse every qubit oscillation period as shown in Fig. 3.2. The time integral of an SFQ pulse is equal to the superconducting flux quantum and determines the energy deposited in the qubit. The result of this energy deposition is a small rotation of $\delta\theta$ around the y-axis. We can perform a rotation along the y-axis by keep applying one SFQ pulse every qubit period. An arbitrary single-qubit gate can be represented by a bitstream (denoted as SFQ bitstream); the gate is applied by applying the SFQ bitstream to the qubit, one bit at a time: if the bit is 1 (0), we apply (don't apply) an SFQ pulse to the qubit at the corresponding SFQ chip cycle.

Despite its high potentials, SFQ imposes unique constraints on the controller design. First, limited JJ density in existing technology (100X-10,000X lower density than CMOS [Tannu et al., 2017b]) leads to lack of dense memory/logic in SFQ. Second, SFQ design is different from that of CMOS due to unconventional pulse-driven nature of SFQ logic. SFQ logic gates receive clock signals and they are evaluated by arrival and consumption of clock pulses. Therefore, balancing the circuit path by inserting extra DFFs is essential to ensure that inputs are consumed at correct clock cycles [Pasandi and Pedram, 2019b]. The extra DFFs further increase the logic area (and power), and might incur significant costs in complex logic designs. Thus, we need to take these constraints into consideration and (1) use limited SFQ storage; (2) keep the logic simple.

3.3 Related Work

Table 3.1: Design space for SFQ-based single-qubit gate controllers.

	SFQ_MIMD_naive	SFQ_MIMD_decomp	SFQ_SIMD_decomp (DigiQ)	
			DigiQ_min	DigiQ_opt
Scalability	Limited by power, area, and bandwidth	Limited by power and area	High scalability	
Quantum program execution	No gate serialization	No gate serialization	Long decompositions	Potential serialization
Pulse calibration	Hardware	Hardware	Software	

In this section we discuss prior research on in-fridge quantum gates and controllers based on various technologies, and also SFQ-based accelerators.

3.3.1 Cryo-CMOS based quantum controllers

Due to maturity of CMOS technology, Cryo-CMOS is an attractive technology to do computation at the 4 K stage of the fridge. In [Van Dijk et al., 2020], single-qubit gate operation using a Cryo-CMOS controller prototype is demonstrated, which is done by generating microwave control signals inside the fridge using the pulse information that is stored in on-chip SRAMs. The prototype presented in [Van Dijk et al., 2020] can scale to >800 qubits given 12 mW/qubit power consumption reported in the paper and 10 W power budget [McDermott et al., 2018]. In comparison, SFQ logic can potentially maximize the scalability of quantum controllers (>42,000 qubits in our SIMD design) due to its very low power consumption. Note that SIMD can potentially increase the scalability of today's Cryo-CMOS prototypes as well, which we see as important future work (see Sec. 3.7).

3.3.2 SFQ-based quantum gates and controllers

In [Leonard et al., 2019], coherent control of a qubit using SFQ pulse trains is demonstrated using a DC/SFQ converter that is fabricated on the qubit chip; the experimental results in the paper show the feasibility of performing quantum gates using SFQ. In [Li et al., 2019], the authors propose a method to find SFQ bitstreams for qubits with different frequencies using one single SFQ clock. A genetic algorithm is used in [Liebermann and Wilhelm, 2016] as an approach to find efficient SFQ bitstreams to reduce leakage errors (i.e., unwanted energy transfer to states other than $|0\rangle$ and $|1\rangle$) and gate time. Reinforcement learning is another approach that has been utilized to find efficient SFQ bitstreams to perform quantum gates [Dalgaard et al., 2020]. In [McDermott et al., 2018], the authors outline a vision to perform fault tolerant quantum computing that relies on repeated streaming of stored SFQ bitstreams, and present a simple estimation of power by adding the power of SFQ registers. In contrast to these works, *DigiQ* is the first system-level design of a scalable NISQ-friendly SFQ-based controller.

3.3.3 SFQ-based accelerators

SFQ has been utilized to design hardware accelerators thanks to its ultra-high speed. In [Holmes et al., 2020], the authors propose an SFQ-based error decoder to accelerate quantum error correction. In [Ishida et al., 2020], the authors present an ultra-fast SFQ-based neural processing unit. In contrast to these works, our focus is on designing a scalable SFQ-based controller rather than accelerating a task.

3.4 DigiQ quantum controller

In this section we provide guidelines for designing scalable SFQ-based controller architectures for universal quantum computation, and present *DigiQ*, our novel SFQ-based controller architecture.

3.4.1 SFQ-based universal quantum computation

Design space for single-qubit gate controllers

A naive design to do arbitrary single-qubit gates is to allocate separate SFQ registers for SFQ bitstreams per qubit (similar to [McDermott et al., 2018]) and update them as needed. This design, denoted as *SFQ_MIMD_naive*, is similar in spirit to today's microwave-based designs, but relies on digital communication from room temperature rather than analog communication. *SFQ_MIMD_naive* is straightforward and provides quantum gate parallelism (i.e., multiple instruction, multiple data (MIMD) paradigm). However, it requires high communication bandwidth from room temperature in scenarios where we are not primarily repeating the same quantum gates, and thus need to update a large number of SFQ registers on-the-fly during the quantum program execution. In addition, *SFQ_MIMD_naive* is expensive in terms of power and area (5.01 mW/qubit and 13.9 mm^2/qubit just for 300-bit SFQ registers based on our results obtained using detailed SFQ synthesis tools and cell libraries). Thus, the scalability of *SFQ_MIMD_naive*, is limited by power/area/bandwidth.

In order to reduce the required bandwidth from room temperature, we can store a universal single-qubit gate set per qubit on the SFQ chip, and send the quantum gate decomposition information from the room temperature. In the simplest case where the single-qubit gate set includes two gates, we need to send only 1 bit per qubit at any given time from room temperature to select/apply one of the two gates. This design, denoted as *SFQ_MIMD_decomp*, reduces the bandwidth requirement significantly compared to *SFQ_MIMD_naive*. However, it further increases the power and area as it allocates more than one SFQ register per qubit. Thus, the scalability of *SFQ_MIMD_decomp* is still limited by power/area.

Finally, we demonstrate our scalable design. Tight power, area, and bandwidth budget of dilution refrigerators, and lack of dense memory in SFQ (see Sec. 3.2) leads us to a design where we share SFQ bitstreams among a group of qubits (i.e., SIMD). Grouping is static and done at the design time, such that qubits in a group have the same nominal oscillation frequency; we show that we can compensate for frequency drift in software in Section 3.5. Note that sharing the bitstreams can be done efficiently in SFQ by broadcasting the bitstreams (one bit per SFQ cycle) using splitter gates. This design, denoted as *SFQ_SIMD_decomp*, is a potential candidate to realize a controller with high scalability thanks to its reduced power, area, and bandwidth requirements compared to the other two designs. We therefore base *DigiQ* on this design. Table 3.1 summarizes the investigated design space.

Single-qubit gate decomposition

Out of the plethora of proposed quantum gate decomposition protocols for single-qubit gates, we must choose the ones that can be implemented efficiently using SFQ. We prefer a decomposition protocol that relies on storing/processing a limited number of SFQ bitstreams (see Sec. 3.2). This is important because (1) there is a lack of dense memory in SFQ, thus we can afford to store only a limited number of SFQ bitstreams; (2) we need to keep the SFQ logic simple, thus we can afford to process only a limited number of SFQ bitstreams at any given time. We consider two SFQ-friendly gate decompositions and present their corresponding SFQ-based architecture designs.

Our first architecture, denoted as *DigiQ_min*, is based on a minimal universal single-qubit gate set that includes only 2-4 gates (e.g., $[Ry(\frac{\pi}{2}), T]$ is sufficient for universal single-qubit computation [Kitaev, 1997, Dawson and Nielsen, 2005]) with the goal of minimizing the hardware cost. Our digital controller works with a clock (denoted as *quantum controller clock*), and similar to classical computers, each quantum program is decomposed into a number of quantum gates and each quantum gate is finished in a number of controller cycles. In *DigiQ_min*, single-qubit gates are decomposed into a sequence of the gates in the gate set, and each gate of the sequence is executed in one controller cycle. At the beginning of each controller cycle, the SFQ bitstreams corresponding to all the gate are broadcasted to a group of qubit controllers. Each qubit controller uses a simple SFQ-based multiplexer to select/apply one of the possible bitstreams using the control bits that are sent from the room temperature in each controller cycle.

A concern is that *DigiQ_min* needs long gate sequences to perform arbitrary single-qubit gates. Thus, we also present *DigiQ_opt*, in which we implement the continuous gate set $\{Ry(\frac{\pi}{2}), Rz(\phi); \phi \in [0, 2\pi)\}$, enabling constant-depth single-qubit gate decomposition [McKay et al., 2017]. $Ry(\frac{\pi}{2})$ can be implemented by storing an SFQ bitstream on the SFQ chip. The next question is: how to implement $Rz(\phi)$ gates for arbitrary ϕ efficiently?

In microwave-based systems, $Rz(\phi)$ gates can be done virtually (in software) by changing the phase of the microwave pulses of the consecutive gates [Krantz et al., 2019, McKay et al., 2017]. But, in the SFQ case, we do not have control over the phase of the SFQ bitstream. Thus, we need to perform the $Rz(\phi)$ gates in hardware. Thanks to the ultra-fast, precise clock on the SFQ chip (in the order of ps), we can perform arbitrary $Rz(\phi)$ by letting the qubits evolve freely for a precise number of SFQ chip clock cycles, which is equivalent to applying an SFQ bitstream consisting of only "0" bits. Every qubit oscillation period, a qubit performs a full rotation around the z-axis with a trajectory in the form of a unit circle (see Fig. 3.2). Now, for any ϕ, there is a point on the unit circle corresponding to $Rz(\phi)$ and we need to get as close as possible to that point in order to perform $Rz(\phi)$ with high fidelity. By applying a "0" bit every SFQ chip cycle, we get to multiple points on the unit circle in one qubit oscillation period, and we cover more points if we let the qubit evolve freely for more than one qubit period. The longer we let the qubit evolve freely, the more points we can get to on the unit circle and the more precisely we can approximate $Rz(\phi)$ for arbitrary ϕ. The fidelity of these gates is analyzed in Sec. 3.5.1.

We now discuss implementation details of *DigiQ_opt* single-qubit gates. Any one-qubit gate can be decomposed as $U(\phi_3, \phi_2, \phi_1) = Rz(\phi_3)Ry(\frac{\pi}{2})Rz(\phi_2)Ry(\frac{\pi}{2})Rz(\phi_1)$. In *DigiQ_opt*, we break the $U(\phi_3, \phi_2, \phi_1)$ into three parts, $Ry(\frac{\pi}{2})Rz(\phi_1)$, $Ry(\frac{\pi}{2})Rz(\phi_2)$, and $Rz(\phi_3)$. We perform each of the first two parts in one controller cycle, and absorb the $Rz(\phi_3)$ to the next controller cycles. The controller cycle length is equal to the $Ry(\frac{\pi}{2})$ gate time plus $N * SFQ\_chip\_cycle\_length$, where N is the maximum number of SFQ chip cycles that we let the qubits evolve freely in order to perform $Rz(\phi)$ gates. We perform $Ry(\frac{\pi}{2})Rz(\phi)$ by applying a "0" bitstream of length $0 \leq d \leq N$ followed by the $Ry(\frac{\pi}{2})$ bitstream, which is equivalent to delaying the $Ry(\frac{\pi}{2})$ bitstream by d SFQ chip cycles as shown in Fig. 3.3. The value of d depends on the angle ϕ and is sent by the compiler at room temperature every controller cycle; the SFQ logic in *DigiQ_opt* delays the stored $Ry(\frac{\pi}{2})$ bitstream by d SFQ

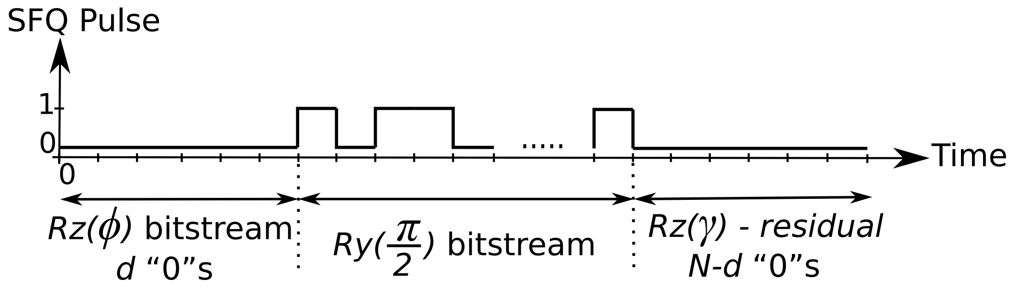


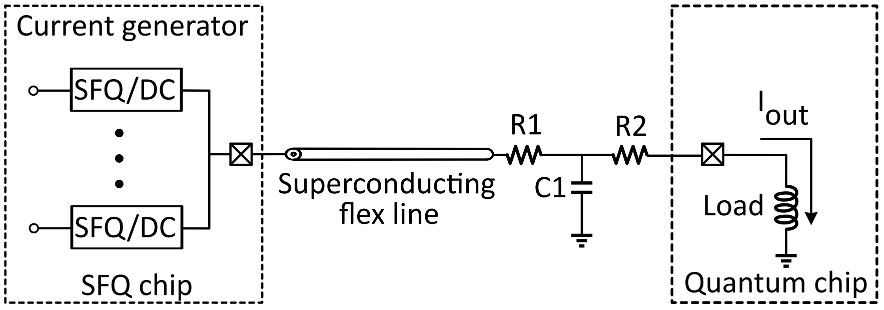
Figure 3.3: The sequence of gates in one cycle of *DigiQ_opt*.

chip cycles and broadcasts it to a group of qubit controllers. The residual $Rz(\gamma)$ rotation at the end of the controller cycle (see Fig. 3.3) can be absorbed into the next gate on that qubit.

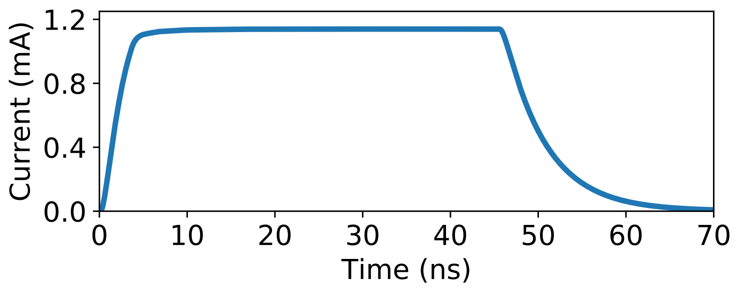
The next question is: how many distinct $Ry(\frac{\pi}{2})Rz(\phi)$ gates out of a total of $N+1$ possible gates (denoted as BS) are needed every controller cycle? The answer depends on (1) the available power and area budget inside the fridge: providing more distinct gates requires more complex logic to delay the stored $Ry(\frac{\pi}{2})$ bitstream by more distinct values at the same time, and also broadcast and process more SFQ bitstreams; (2) the similarity between the gates that qubits inside a group perform in a given instant: there will be serialization inside a group if not enough distinct gates are available in a controller cycle. The lower the BS value, the lower the hardware cost but the higher the chance of serialization (see Sec. 3.6).

Two-qubit gate design

In this paper, we develop SFQ circuits to generate electrical current waveforms necessary to realize CZ gates on flux-tunable transons (see Sec. 3.2) inside the fridge. The electrical currents are transmitted to the quantum chip using superconducting microstrip flex lines (see Fig. 3.4(a)). We deploy an array of SFQ/DCs, which are commonly used SFQ blocks to convert the SFQ pulsed representation to DC voltage levels [Kaplunenko et al., 1989]. When a *start* signal is received by the SFQ/DC blocks, they start outputting electrical current, and they keep doing so until they receive a *stop* signal (we need to turn the SFQ/DCs on (off)



(a)



(b)

Figure 3.4: (a) Circuit schematic of our current generator design based on SFQ/DCs; (b) The electrical current pulse generated by our design to realize CZ gates on flux-tunable transmons.

in the beginning (end) of the CZ gate). In order to target specific pairs of qubits on a multi-qubit system we require current waveforms to be applied to both qubits simultaneously (we need one current generator per qubit). The only difference between this approach and prior flux-tunable implementation of CZ gates is that the electrical current is generated inside the fridge. Note that we use the same two-qubit gate design for both *DigiQ_min* and *DigiQ_opt*.

We use JSIM which is a detailed circuit simulator for superconductive electronic applications [Fang, 1989] to simulate the Fig. 3.4(a) current generator circuit. Fig. 3.4(b) shows the simulated current waveform, which is generated by enabling 25 SFQ/DC blocks (the values of $R1$, $R2$, and $C1$ are 0.05 ohm, 0.05 ohm, and 10 nF, respectively). In Sec. 3.5.2, we use physical simulations to show that this waveform can be combined with software calibration techniques to realize low-error CZ gates even when qubits exhibit independent frequency variation.

Recent studies have also suggested realizing two-qubit cross-resonance gates using only

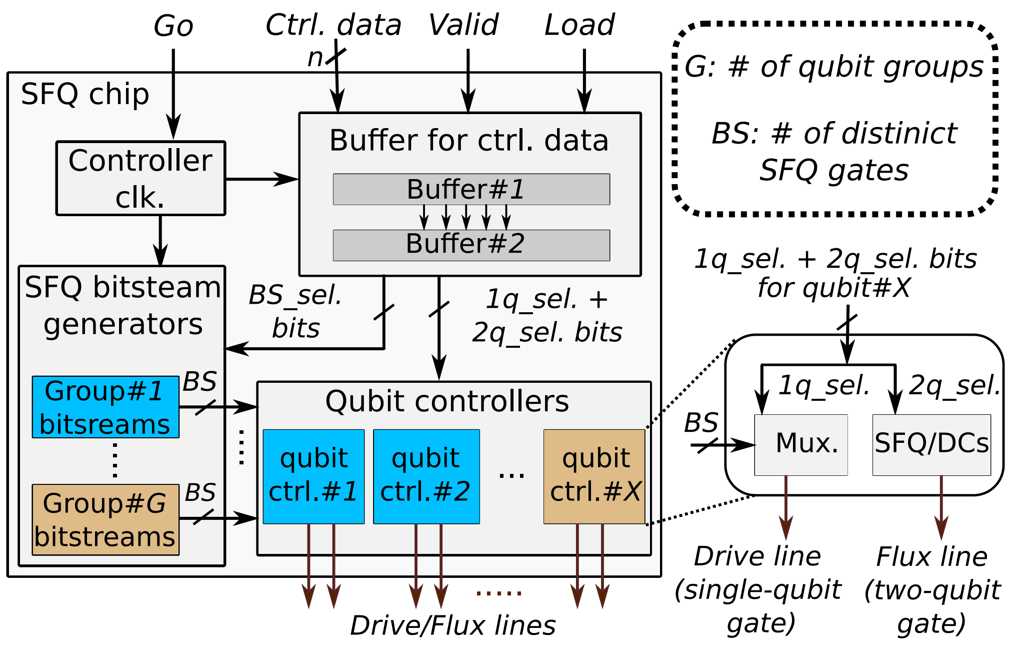


Figure 3.5: Overview of our *DigiQ* architecture.

SFQ pulses [Jokar et al., 2021, Dalgaard et al., 2020]. The design of an SFQ controller architecture based on such gates would present a different set of challenges and opportunities, the study of which we save for future work.

3.4.2 Overview of DigiQ *architecture*

Now we put it all together and demonstrate an overview of our *DigiQ* architecture that is shown in Fig. 3.5. There are *G* groups of qubit controllers. At the beginning of each controller cycle, *BS* distinct single-qubit SFQ gates per each group are broadcasted to all the qubit controllers in the group. *BS_sel* bits are used to select the *BS* distinct single-qubit gates in *DigiQ_opt* (*DigiQ_min* does not need *BS_sel* bits since its universal single-qubit gate set includes only a few gates, which are all broadcasted). Each qubit controller uses an SFQ-based multiplexer and the *1q_sel* bits to either select/apply one of the *BS* delayed bitstreams, or apply none of them (e.g., in the two-qubit gate case). For two-qubit gates, the qubit controllers of the two qubits involved use the *2q_sel* bits to determine whether to start/stop performing CZ gate by sending *start/stop* signals to the SFQ/DCs.

BS_sel, *1q_sel*, and *2q_sel* control bits are sent from the room temperature every controller cycle using *Ctrl. data* cables; a *Valid* cable is used to determine the validity of control data on data cables. A *Load* cable is also used to load the SFQ bitstreams through the data cables, which is done offline (i.e., not during the program execution); each SFQ bitstream has $\leq$300 bits (the actual bitstream length depends on the target gate and system Hamiltonian). After the transmission of the control bits of the first controller cycle is finished, a *Go* signal is sent from room temperature to start the controller clock (which is implemented using a counter that counts up every SFQ chip cycle and resets every controller cycle). At the beginning of every controller cycle, the control bits that are already buffered in a buffer (*Buffer#1* in Fig. 3.5) are transferred to a second buffer (*Buffer#2* in Fig. 3.5) to be used by qubit controllers and SFQ bitstream generators. While executing the current controller cycle, the control bits of the next controller cycle are buffered in the first buffer.

3.5 Software calibration of SIMD hardware

Real superconducting qubits exhibit variations in their oscillation frequency which can drift from day to day in experimental settings [Gokhale et al., 2020, Tannu and Qureshi, 2019]. Quantum gates are implemented using precise control signals designed using careful physical models of qubit evolution (that is, their Hamiltonian) [Krantz et al., 2019], which are extremely sensitive to small deviations in qubit frequency. The accuracy of quantum gates optimized for one set of qubit frequencies will therefore degrade rapidly if the same control signals are used on qubits exhibiting slight deviations from those frequencies. State-of-the-art quantum gates therefore require regular recalibration using experimental measurements of each qubit [Gokhale et al., 2020]. For small systems with MIMD control, qubit-specific gate calibration can be performed at the hardware level by precisely shaping control pulses for each qubit. This is not possible for a SIMD architecture such as *DigiQ*, in which control signals are shared among many qubits and so cannot be calibrated independently for each qubit.

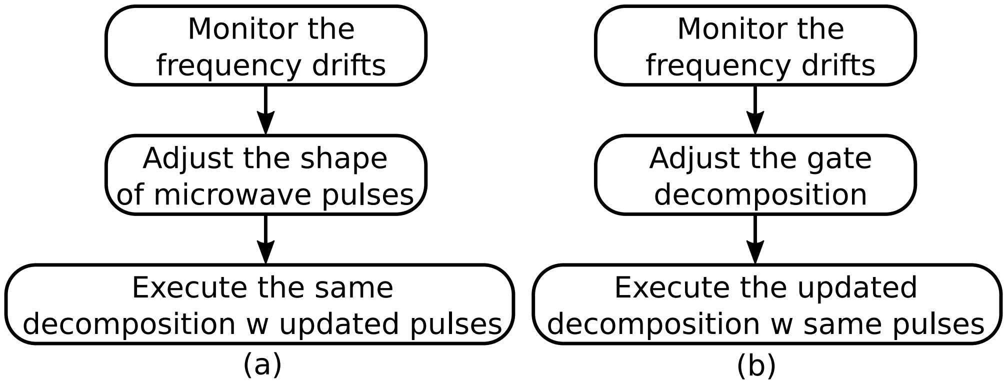

Monitor the frequency drifts
Adjust the shape of microwave pulses
Execute the same decomposition w updated pulses
(a)
Monitor the frequency drifts
Adjust the gate decomposition
Execute the updated decomposition w same pulses
(b)

Figure 3.6: Calibration process in (a) today's microwave-based quantum machines; (b) *DigiQ*.

Here, we show that this challenge can be overcome by moving gate calibration to the software level, eliding unscalable qubit-specific hardware configurability. Our approach is summarized in Fig. 3.6. In short, basis SFQ bitstreams that are stored on SFQ chip are fixed, and calibration is performed by adjusting the decomposition into the (frequency-dependent) set of basis quantum operations resulting from those fixed bitstreams; note that applying the same SFQ bitstream to multiple qubits with different qubit frequencies results in different quantum operations. Though the focus of this section is compensation for frequency drift, similar techniques could be employed to mitigate other sources of systematic error.

The gate errors reported through the remainder of this paper are calculated as $\epsilon = 1 - \overline{F}$, where $\overline{F}$ is the average gate fidelity [Nielsen, 2002, Ghosh, 2011]. We calculate gate fidelities by modeling and simulating the Hamiltonian of the quantum hardware to provide an upper bound for the gate fidelity. This approach is also used in studies on both SFQ-based and microwave-based systems as a precursor to experimental work [Li et al., 2019, Leung et al., 2017]; *DigiQ* shows comparable gate fidelity to these studies as shown in Sec. 3.6.2. Although we model the key dominant sources of systematic error, experimental evaluations on real hardware (currently available only at a small scale) are subject to various sources of noise which are not captured in the models, resulting in lower gate fidelities [Leonard et al., 2019, Sheldon et al., 2016].

3.5.1 Calibrating single-qubit gates

The key idea behind gate calibration on *DigiQ* is that these frequency-dependent operations still constitute universal gate sets for single-qubit computation [Kitaev, 1997]. We can therefore account for frequency drifts and hardware variation on each qubit by decomposing single-qubit gates on each qubit using the unique set of basis operations resulting from shared SFQ bitstreams. For both *DigiQ_opt* and *DigiQ_min*, the single-qubit calibration procedure then consists of four steps: (1) find SFQ bitstreams implementing a desired set of basis gates with high fidelity on qubits with no frequency variation, (2) characterize each qubit's actual oscillation frequency using experimental measurements [McDermott et al., 2018], (3) use the learned bitstreams and measured frequencies to determine the actual basis operations implemented on each qubit by the shared bitstreams, and (4) compile quantum circuits using the actual single-qubit basis operations determined for each qubit. The (classical) complexity of these steps scales linearly with number of qubits and circuit size.

For steps (1) and (3) we employ single-qubit physical simulations of the expected single-qubit unitary evolution U_{bs} produced by an SFQ bitstream on transmon qubits (as done in [Li et al., 2019]). To ensure we are fully accounting for state leakage, we model transmons using six energy levels, and then compute single-qubit gate fidelities by projecting the resulting evolution back into the two-level subspace (causing any leakage to be captured as additional error [Ghosh, 2011]).

For *DigiQ_opt*, calibrating gate decomposition means finding a new set of delay intervals such that the target operation is approximated by a sequence $Rz(\phi_L)U_{bs}...Rz(\phi_1)U_{bs}Rz(\phi_0)$ (where the final $Rz(\phi_L)$ is absorbed into a subsequent gate). We choose sets of delays holistically for each gate, numerically searching for the best combination of the available delays to implement that gate. We find that $L \leq 2$ is sufficient for most gates, but a subset of gates nearing π rotations around the x or y axis (e.g., Pauli X and Y operations) need $L = 3$ to obtain high fidelity on qubits with the most significant drift (whereas in the ideal case (i.e., $U_{bs} = Ry(\frac{\pi}{2})$), $L \leq 2$ is enough for all single-qubit gates).

Table 3.2: Optimal parking frequencies and drift tolerance for $Rz(\phi)$ gates with $\leq 10^{-4}$ error for $N = 255$.

Parking frequency (GHz)	Drift (GHz) for error $\leq 10^{-4}$
6.21286	± 0.01282
5.02978	± 0.01049
4.14238	± 0.00820

Two factors affect performance of the *DigiQ_opt* decomposition. First, the set of available $Rz(\phi)$ rotations is highly dependent on qubit frequency, which determines how well the $N+1$ possible delay values cover the unit circle. In the ideal case, the $N + 1$ phases are equally spaced around the unit circle, and any $Rz(\phi)$ can be approximated to within $\pi/(N + 1)$ radians; in this case we find that $N = 255$ is sufficient for error $\epsilon \leq 0.25 \cdot 10^{-4}$. We choose target frequencies with the highest tolerance for variation, as measured by the width of the interval in which any ϕ can be approximated with $< 10^{-4}$ error using one of the available delays. These optimal parking frequencies and their drift tolerance are shown for $N = 255$ in Table 3.2. Second, frequency variations can limit SIMD scheduling. For example, if a circuit calls for the same gate to be applied to many qubits, these may still require unique delay values when the decomposition of that gate differs for each qubit. However, we can increase parallelism by allowing a small error margin when choosing delay values: often, multiple sets of delays will approximate the same operation with nearly equal error, so we can choose the one with lowest cost in terms of serialization.

For *DigiQ_min*, we decompose arbitrary single-qubit gates into sequences of discrete, qubit-specific basis operations. We use a brute-force search algorithm to find the optimal decomposition of single-qubit gates for each qubit, working with the full six-level representation of the unitary basis operations so that the decomposition accounts for and mitigates leakage resulting from each basis operation.

3.5.2 Calibrating two-qubit gates

The implementation of CZ gates using flux-tunable transmons requires the shape and duration of each current pulse to be carefully calibrated to the precise qubit frequencies and hardware parameters. On a small system with MIMD control, these pulses can be individually calibrated for each pair of coupled qubits to account for variation and drift. Without this fine-grained control, we are instead left with a unique 2-qubit operation U_{qq} for each coupled pair of qubits. Here, we argue that we can instead compose CZ gates for each pair of qubits in *DigiQ* using pair-specific sequences of U_{qq} operations and single-qubit gates, again relegating calibration to software.

We can compute the unitary evolution U_{qq} for a pair of qubits using physical simulations of the empirical current waveform described in Section 3.4.1. Starting with the nominal pre-drift frequencies of 6.21286 and 4.14238 GHz from Table 3.2, we vary each qubit's frequency and compare the resulting U_{qq} to the target CZ operation. We compute unitary evolution by numerically integrating the Schrödinger equation using well-understood Hamiltonian models of pairs of capacitively-coupled flux-tunable asymmetric transmons [Krantz et al., 2019]. We assume that each transmon has an anharmonicity of 250 MHz, and the capacitive coupling strength is 10 MHz.

In Fig. 3.7(a), we show the expected minimum error when implementing a CZ gate using a single U_{qq} pulse as a function of each qubit's drift, allowing for arbitrary single-qubit gates before and after the pulse. At the ideal qubit frequencies, we find an expected error of $\epsilon = 3 \cdot 10^{-4}$. This error grows rapidly as the frequency difference between qubits drifts. In Fig. 3.7(b) and (c), we show the gate error after compiling CZ gates into sequences of 2 or 3 U_{qq} operations and intermediary single-qubit gates, similar to the "echo" sequences described in [Córcoles et al., 2013, Sheldon et al., 2016] but with single-qubit gates obtained via numerical optimization. Assuming ideal single-qubit gates, we find that 3 U_{qq} operations are sufficient to achieve $\epsilon \leq 10^{-4}$ over a wide range of frequencies. In Section 3.6.2, we report empirical gate errors after single-qubit gates are decomposed for either *DigiQ_opt* or

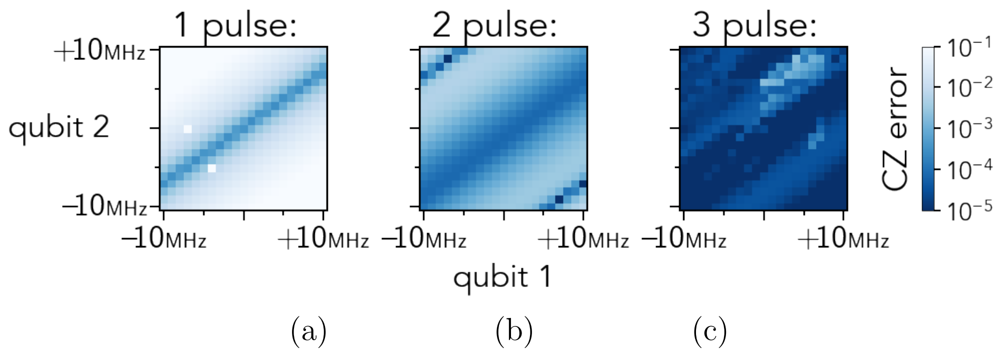


Figure 3.7: CZ gate error as a function of frequency drift, assuming 1, 2, or 3 U_{qq} operations and ideal single-qubit gates.

Table 3.3: The library of RSFQ cells and corresponding characteristics used for synthesis.

Cell	Area (μm^2)	JJ Count	Delay (ps)
AND2	3500	16	8.4
OR2	3500	14	6.1
XOR2	3500	18	5.8
NOT	3500	12	13.2
DRO DFF	3000	11	6.2
NDRO DFF	4500	18	9.3
Splitter	2000	6	7.1

DigiQ_min.

3.6 Methodology and Results

In this section, we present hardware synthesis results of *DigiQ*, followed by an analysis of its algorithmic performance.

3.6.1 Hardware results of DigiQ

We use detailed SFQ synthesis tools [Pasandi and Pedram, 2019a, Pasandi et al., 2018, Pasandi and Pedram, 2019d,b,c] to synthesize, map, and finally calculate the power, area, and delay values. The employed synthesis and technology mapping flow is as follows: first, technology-independent optimizations including balanced factorization and rewriting [Pasandi and Pedram, 2019a] are performed on the input circuit, then the circuit is mapped using a path balancing technology mapping algorithm [Pasandi and Pedram, 2019b] and fully path balanced [Pasandi and Pedram, 2019d]. Next, a standard retiming algorithm similar to [Leiserson and Saxe, 1991] is employed to further reduce the total memory element count. Finally, the memory elements (e.g., latches) are replaced with SFQ DRO DFFs, and splitters are inserted at the output of gates with more than one fanout.

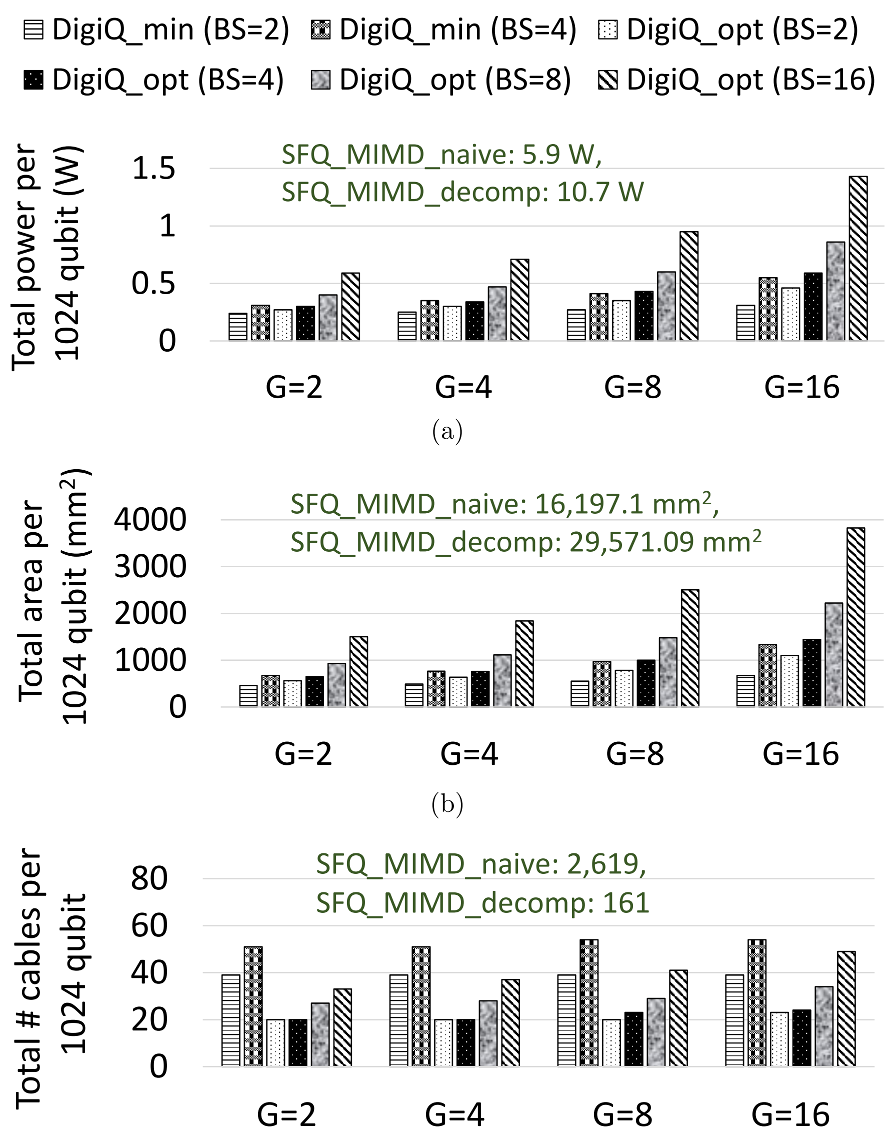
Rapid Single Flux Quantum (RSFQ) logic family [Likharev and Semenov, 1991] is used in this paper, and the library of cells is listed in Table 3.3. For wiring, we use Josephson Transmission Line (JTL) and Passive Transmission Line (PTL). JTL is used for short connection for cells next to each other and its delay is ∼1.5-2 ps. PTL is used for transmitting SFQ pulses from one logic gate to another.

Validity of our synthesis results

We have the physical layouts for all cells, including wires and logic cells. Our RSFQ power, area and delay numbers are obtained post-layout based on accurate extraction of the gate layouts and passive transmission lines and subsequently simulated using WRSpice [Whiteley Research Inc.] and JSim [Fang, 1989]. The simulation results are thus highly accurate and reliable. The SFQ library cells have been validated and their power/timing values calibrated against manufactured test chips done in the MIT Lincoln lab's SFQ5ee process node [Tolpygo et al., 2016]. The synthesis, place&route, modeling/simulation, and formal verification tools have been used to design and formally/simulatively verify tens of SFQ circuits ranging from individual data path modules and register files to major building blocks of the RISC-V Sodor core [Cong et al., 2021, Fourie et al., 2019, UC Berkeley Architecture Research]. In addition, prior work validated comparable SFQ tools and simulators with SFQ chip fabrication [Ishida et al., 2020].

Delay results

Our synthesis results show that the worst stage delay in *DigiQ* is 34.5 ps, which determines the maximum clock frequency that our SFQ chip can work with. We choose 40 ps as our



(a)

(b)

(c)

Figure 3.8: Power (a), area (b), and cable count (c) results of *DigiQ_min* and *DigiQ_opt* architectures. *SFQ_MIMD_naive* and *SFQ_MIMD_decomp* results are shown for comparison.

SFQ chip clock period (similar to [Li et al., 2019]), thus the bit period in our SFQ bitstreams is 40 ps.

Power and area results

Fig. 3.8(a) and Fig. 3.8(b) show the total power and area of *DigiQ*, respectively. We present the results for different *BS* and *G* values.

As mentioned in Sec. 3.4.1, in *DigiQ_opt*, only *BS* distinct delays are available every controller cycle for single-qubit gates, and that can potentially lead to quantum gate serialization inside a group. One solution to mitigate this serialization is to increase the value of BS, which would increase the hardware cost of quantum controller modules as they would need more complex SFQ-based multiplexers to select one out of a larger number of delayed bitstreams (see Fig. 3.5). Another solution to mitigate the serialization is to increase the G value. This solution decreases the number of qubits per each group which means less congestion and less serialization, while not increasing the hardware cost of gate controller modules. Thus, at the design time, we expected that among all the designs with the same $BS * G$, the ones with higher G value have lower hardware cost as they need less complex quantum controller modules. However, our synthesis results surprised us. As shown in Fig. 3.8, the hardware cost of the designs with the same $BS * G$ value does not differ significantly. The reason is that increasing the G value also increases the overall hardware cost due to an increase in the number of SFQ bitstream generators (see Fig. 3.5). Given our synthesis results, we conclude that keeping the G value small and increasing the BS value is preferred because it provides more flexibility in allocating delayed bitstreams to qubits. Smaller G values are preferred for *DigiQ_min* as well, as increasing the G value increases the hardware cost with no significant algorithmic advantage. The smallest G value or the maximum number of qubits in a group is determined by the area of one SFQ chip; if we cannot fit a group on one chip, we need to use multiple chips and replicate the bitstreams on each chip to avoid long distance communications, which is equivalent to dividing the qubits into multiple groups. Our results show that for 1024-qubit scale, we can fit all the designs with $G = 2$ on at most two SFQ chips (one per group), thus we use $G = 2$ for 1024-qubit benchmarks in Sec. 3.6.2.

Our results show that even our largest designs can operate within the power budget of typical dilution refrigerators at 4 K stage (i.e., a few watts [McDermott et al., 2018, Hornibrook et al., 2015, Van Dijk et al., 2020]), and can be implemented on only a few SFQ chips at 1,024-qubit scale; clock synchronization between the SFQ chips is done using

SFQ phase locked loops (PLLs) [Brock and Pambianchi, 2000]. We replicate the 1,024-qubit design in order to scale to larger number of qubits (note that replicating the 1,024-qubit design naturally increases the number of groups). *DigiQ_min*(BS=2) has the lowest hardware cost and highest scalability (>42,000 qubits given 10 W power budget [McDermott et al., 2018]). Increasing the BS value in *DigiQ_min* increases the hardware cost, but also can potentially increase the algorithmic performance (we see diminishing returns after $BS = 4$). The scalability of *DigiQ_opt* is also high, allowing >25,000 qubits (>17,000 qubits) for $BS = 8$ ($BS = 16$).

SFQ storage and Cable count results

DigiQ_min stores BS bitstreams per group and each bitstream has $\leq$300 bits. *DigiQ_opt* stores one bitstream with $\leq$300 bits per group which is delayed by BS different values at each controller cycle (see Sec. 3.4). With both designs, for each qubit at each controller cycle, we need enough control bits from room temperature to determine whether to apply one of the BS distinct gates, start/stop performing two-qubit gates, or perform no operation. For *DigiQ_opt*, an additional BS delay values for each group at each cycle are needed; since we have 256 possible delay values, each delay value requires $\log_2 256 = 8$ bits.

Fig. 3.8(c) shows the number of required cables to send *Go*, *Valid*, *Load*, *BS_sel*, *1q_sel*, and *2q_sel* bits from the room temperature in one controller cycle given 10 Gbps return-to-zero (RZ) cables [Likharev and Semenov, 1991]. We calculate the number of data cables assuming a minimum controller clock period of 9 ns for *DigiQ_min* and 9 ns + 10.2 ns for *DigiQ_opt* (10.2 ns corresponds to 255 delay cycles) – we need enough data cables to send one set of control bits from the room temperature within one controller cycle (see Sec. 3.4.2). *DigiQ_min*(G=2,BS=2) requires only 39 cables per 1,024 qubits (52.5X less than a microwave-based quantum computer with 2 cables per qubit, 1 drive line and 1 flux line [Arute et al., 2019]). *DigiQ_opt* requires just 33 cables per 1,024 qubits in $G = 2$, $BS = 16$ design. *DigiQ_min*'s high cable count relative to most *DigiQ_opt* configurations is due to its

Table 3.4: NISQ benchmark algorithms.

QGAN	Quantum generative adversarial learning network [Lloyd and Weedbrook, 2018]
Ising	Linear Ising model spin chain simulation [Barends et al., 2016]
BV	1024-bit Bernstein-Varzirani algorithm [Bernstein and Vazirani, 1997]
Add1	256-bit ripple-carry adder [Cuccaro et al., 2004]
Add2	256-bit parallel carry-lookahead adder [Draper et al., 2006]
Sqrt10	10-bit square root via Grover search [Javadi-Abhari, 2017, Murali et al., 2019b, Grover, 1996]

shorter controller cycle.

3.6.2 *Algorithmic performance results of* DigiQ

In order to study the algorithmic impacts of *DigiQ*, we compile a common set of NISQ benchmarks for each design. The benchmarks chosen are described in Table 3.4, and together represent a diverse sample of common circuit formulations. Benchmark circuits are algorithmically generated and mapped to a 32×32 square grid via SWAP-gate insertion using the stochastic transpiler pass packaged with Qiskit Terra [Abraham et al., 2019]. Each circuit is then decomposed into CZ and single-qubit gates, and a crosstalk-aware scheduling pass [Murali et al., 2020] is used to sort and group commuting two-qubit gates which can be executed simultaneously without interference.

To model frequency variation, each qubit is modeled as an asymmetric transmon with $\sigma = 0.2\%$ variability in each of its Josephson energies (sampled from a normal distribution). At our target frequencies, this corresponds to about ± 6 MHz fluctuation in oscillation frequency, in keeping with design targets for future superconducting systems [Hertzberg et al., 2020]. Hardware variability is considered with the addition of a $\sigma = 1\%$ error to the output of each current generator. As in Section 3.5, we use these sampled hardware parameters to physically simulate each basis operation on each qubit or qubit pair (similar to prior work [Leung et al., 2017, Li et al., 2019]). We then use the resulting unitary basis operations to decompose each gate in the benchmark circuits. Gate errors in Section 3.6.2 are determined by computing the overlap between the decomposed and target gates.

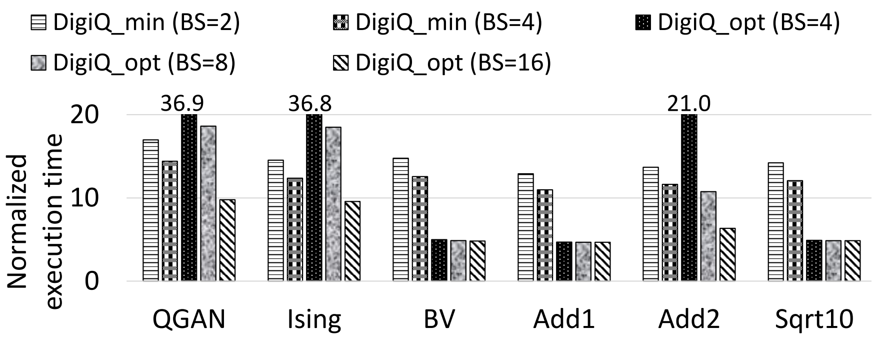


Figure 3.9: *DigiQ* quantum circuit execution time normalized to the Impossible MIMD system.

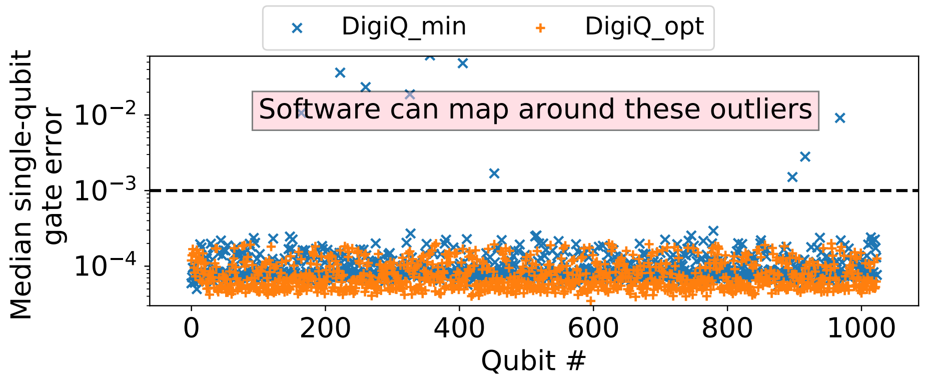
For *DigiQ_opt*, we use a 20.32 ns controller cycle time, comprising 10.12 ns for SFQ bitstreams and 255 delay cycles. The CZ gate time is 60 ns (determined from the analysis in Section 3.5.2), which expands over three controller cycles. Single-qubit gates are expanded into at most three U_{bs} pulses (see Sec. 3.5.1). For *DigiQ_min*, single-qubit gate times for the 6.21286 and 4.14238 GHz qubit frequencies are respectively 10.12 ns and 9.00 ns, again with a 60 ns CZ gate time. We decompose single-qubit gates until the approximation error falls below 10^{-4}, up to a maximum depth of 28 (at which point we find that only about 1% of gates have errors above 10^{-4}). There is no feedback loop in our benchmarks, thus we do not consider the communication latency from outside the fridge.

Circuit execution time

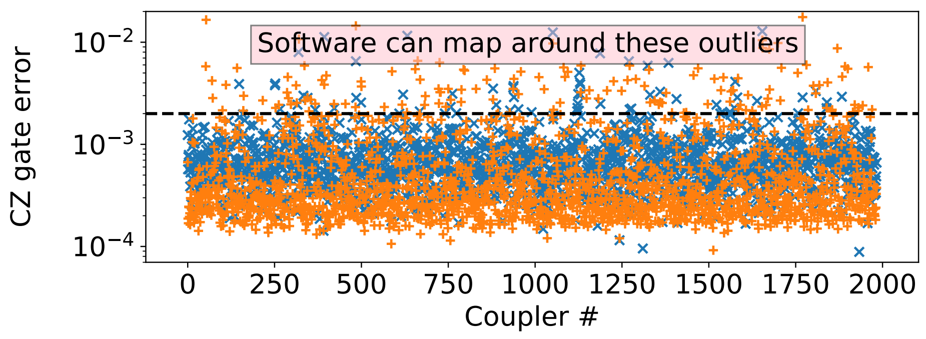
Fig. 3.9 shows the execution time results for *DigiQ* with 1024 qubits, normalized to an *Impossible MIMD* system which is assumed to have the same gate times as *DigiQ* (which are also similar to those of recent microwave-based systems [Arute et al., 2019, Gokhale et al., 2020]) but with unlimited parallelism. We emphasize that the MIMD system is impossible at large scales (see Sec. 3.2); we compare our results with the Impossible MIMD system just to quantify the serialization cost of realizing a controller design that is feasible at large scales, and to give readers a sense of how *DigiQ* would compare to today's prototypes if they did not have scalability issues. Both *DigiQ_opt* and *DigiQ_min* have some overhead in terms of execution time compared to the Impossible MIMD system (*DigiQ_opt* due to serialization and *DigiQ_min* due to longer gate decompositions). The performance of *DigiQ_opt* increases by increasing the BS value, as expected, with $BS = 16$ providing the best performance. This difference is minimal for benchmarks without significant parallelism (BV, Add1, Sqrt10). For $BS = 16$, the execution time is only $\leq 2X$ longer than what we would be able to achieve if *DigiQ_opt* could be built with zero quantum gate serialization (i.e., $BS = \infty$). In *DigiQ_min*, increasing BS from 2 to 4 reduces the depth of single-qubit gate decompositions by roughly half. However, the benefit is limited beyond $BS = 4$ due to the dominance of CZ gate decompositions which do not benefit substantially from the larger single-qubit gate set. *DigiQ_min* performs similarly to *DigiQ_opt*(BS=8) for the benchmarks with more parallelism (QGAN, Ising, Add2), in which *DigiQ_opt* experiences more gate serialization. For the remaining benchmarks, *DigiQ_min*'s long gate decomposition leads to worse performance than any *DigiQ_opt* configuration. Compared to Impossible MIMD system, *DigiQ_opt*(BS=16) is 4.7X-9.8X worse in terms of circuit execution time, while *DigiQ_min*(BS=4) is 11.0X-14.4X worse.

Gate/circuit error

We estimate the overall circuit error due to gate decomposition by taking the product of the errors of each of its gates. For both designs, typical gate error varies between qubits with different frequency drifts. Fig. 3.10(a) shows the median error of single-qubit gates (evaluated across all gates in all benchmarks) for each qubit on *DigiQ_opt*(BS=8) and *DigiQ_min*(BS=2) (the same conclusions hold for other BS values). We find that *DigiQ_opt* exhibits higher variability between qubits, whereas *DigiQ_min* is generally more stable but has a small number of especially bad outlier qubits (where the frequency drift brings the nominal T gate very close to the identity, resulting in a poor single-qubit gate set). Similar to microwave-based systems, we can map around these outlier cases in software using noise-adaptive mapping techniques [Murali et al., 2019a]. The CZ gate error (Fig. 3.10(b))



(a)



(b)

Figure 3.10: (a) Median single-qubit gate error on *DigiQ_opt* (BS=8) and *DigiQ_min* (BS=2) with 1024 qubits (well representative of other configurations); (b) CZ gate error on each qubit pair. Software can map around the outliers using the noise-adaptive mapping techniques [Murali et al., 2019a].

is driven both by the decomposition of CZ into U_{qq} and single-qubit gates, and by the error with which we can decompose those single-qubit gates on each architecture. The CZ error is >0.002 for 3% and 7% of qubit pairs in *DigiQ_min* and *DigiQ_opt*, respectively. Note that the CZ error would be >0.002 for 84% of qubit pairs if we did not use our software calibration techniques. Our results show that with software calibration *DigiQ* can achieve similar gate error to that of hardware-calibrated microwave-based gates [Leung et al., 2017].

3.7 Conclusions and Future work

Large-scale quantum computers are essential in order to perform many useful quantum algorithms. However, superconducting quantum computer prototypes have severe scalability issues due to the massive overhead of generating and routing control pulses from room temperature to quantum chip inside the dilution refrigerator. In this work, we present *DigiQ*, the first system-level design of a digital SFQ-based quantum controller architecture to maximize scalability. We provide architecture guidelines to design large-scale SFQ-based controllers by taking into consideration the opportunities (efficient on-chip broadcast and ultra-fast clock) and challenges (lack of dense memory/logic) of SFQ. Our study shows that we must deploy a SIMD architecture to operate within the tight power/area budget of dilution refrigerators, however, SIMD also introduces new challenges in terms of control pulse calibration in NISQ machines with imperfect qubits. We propose novel software-solution to address these calibration challenges efficiently, and show that software plays a key role in ensuring good quantum algorithmic performance at large scales. We fully characterize *DigiQ* controller hardware using state-of-the-art/validated SFQ synthesis tools. We also show the effectiveness of *DigiQ* in terms of quantum algorithmic performance under a variety of NISQ algorithms. We model dominant sources of systematic error in our evaluations and show that *DigiQ* has similar fidelity to microwave-based systems under the same models. Our analysis shows that *DigiQ* is a realistic path forward to realize large scale quantum computers, and we hope the promising results of this paper motivate experimentalists to further explore SFQ-based controllers.

Going forward, the fidelity of quantum gates/circuits can further be improved by performing further research at both hardware and software levels. First, the main bottleneck in increasing the gate fidelity in *DigiQ* is frequency drift of imperfect qubits, thus developing better qubits with less drift can increase the fidelity significantly. Second, decreasing the power and area consumption of SFQ circuits would allow the realization of more complex SFQ-based controllers that can potentially perform hardware calibration, which combined with our software calibration can further increase the fidelity of quantum gates/circuits. Third, further research at the software level can potentially lead to more robust decompositions with higher fidelity.

Finally, we would like to mention that the ideas proposed in this paper such as SIMD design and software calibration can potentially increase the scalability of today's Cryo-CMOS controllers as well. However, further research on such controllers based on Cryo-CMOS is needed. First, although SIMD can reduce the cost of on-chip storage in Cryo-CMOS, novel approaches are needed to share/broadcast the instructions (i.e., amplitudes/phases of microwave pulses) on the Cryo-CMOS chip with low cost to obtain overall cost reduction. Second, Cryo-CMOS has denser memory/logic, but much higher power, which would result in significantly different set of design tradeoffs. Third, *DigiQ* relies on ultra-fast clock in SFQ (i.e., in the order of ps), and Cryo-CMOS requires different architectures and gate implementations as it works with clocks in the order of ns.

CHAPTER 4

PRACTICAL IMPLICATIONS OF SFQ-BASED TWO-QUBIT GATES

4.1 Introduction

A great milestone in quantum computing is the recent development of quantum computer prototypes thanks to great efforts in industry and academia. Superconducting quantum computing is one of the most promising technologies to realize a quantum computer, having been used to realize prototypes with <100 qubits [Brink et al., 2018, Steffen et al., 2011, Fu et al., 2017, Li et al., 2020, Arute et al., 2019, Kelly, 2018]. These prototypes rely on sending analog microwave signals per qubit from a classical controller at room temperature to the quantum chip inside a dilution refrigerator in order to perform quantum operations. Unfortunately, this scheme introduces severe scalability challenges due to high costs of electronics that are used to generate the microwave signals at room temperature, as well as heat dissipation inside the dilution refrigerator caused by routing the high-bandwidth signals to the quantum chip [McDermott et al., 2018, Leonard et al., 2019, Li et al., 2019]. Thus, design decisions must be made to address the scalability challenges of today's quantum computer prototypes and realize large-scale quantum computers, which are essential in running many quantum algorithms and performing quantum error correction.

One active research area in industry and academia is designing in-fridge classical controllers, which increase the scalability of quantum machines by generating and routing the control signals locally. Due to maturity of CMOS logic, Cryo-CMOS is one attractive logic technology to build in-fridge controllers. Prior work has demonstrated Cryo-CMOS controller prototypes that generate microwave control pulses inside the dilution refrigerator, and can scale to hundreds of qubits given the power budget of dilution refrigerators [Van Dijk et al., 2020]. Meanwhile, Superconducting Single Flux Quantum (SFQ) is proposed as an alternative logic technology in the literature. SFQ logic is less mature than CMOS but can maximize the scalability of in-fridge controllers due to its very low power consumption and ultra-high speed [Leonard et al., 2019, McDermott et al., 2018, Li et al., 2019, Liebermann and Wilhelm, 2016].

SFQ-based controllers can perform quantum operations by generating a train of SFQ pulses (instead of microwave control waveforms) inside the dilution refrigerator and applying them directly to the qubits [Li et al., 2019, Liebermann and Wilhelm, 2016]. Previous work has demonstrated high-fidelity single-qubit gates with low leakage to the non-computational subspace using SFQ pulses [Leonard et al., 2019, Li et al., 2019]. Prior work also demonstrated SQF-based two-qubit gates considering a model which takes into account only the first two energy levels of the qubits (i.e., qubit computational subspace) [Dalgaard et al., 2020]. However, there is a lack of a detailed analysis in the literature on high-fidelity SFQ-based two-qubit gates which takes into account leakage to the non-computational subspace. A key unanswered question is: are SFQ-based two-qubit gates with high fidelity and low leakage feasible and effective? In this paper, we present the first thorough study on SFQ-based two-qubit gates, and demonstrate that we can realize them with high fidelity and low leakage by carefully designing our quantum optimal control method and qubit architecture.

We first demonstrate that it is essential to take higher energy levels of the qubits into consideration in our optimal control method. Similar study has been done in the literature on SFQ-based single-qubit gates [Liebermann and Wilhelm, 2016] where the authors show that taking into account the first three lowest energy levels (i.e., the qubit computational subspace and one higher energy level) in the optimal control method is sufficient to find high-fidelity gates with low leakage to higher energy levels. In this paper, we show that two-qubit gates have much higher tendency to leak to higher energy levels, thus it is challenging to find high-fidelity gates even if we take into account up to five energy levels in our optimal control method. Thus, we must take further steps by developing SFQ-based optimal control methods to suppress leakage and investigating qubit architectures and configurations that reduce leakage.

We first study transmon qubit devices with Ω_x control fields, which are widely used in both SFQ-based and microwave-based systems [Li et al., 2019, Dalgaard et al., 2020, Leung et al., 2017]. We show that we can realize high-fidelity SFQ-based two-qubit gates with low leakage to higher energy levels. We then investigate two possible extensions of this design in order to reduce the gate time while keeping the leakage low: (1) the addition of σ_z SFQ control pulses implemented via frequency-tunable split-transmon devices; and (2) the use of SFQ control pulses in combination with high-anharmonicity fluxonium qubits.

Finally, we compare our SFQ-friendly quantum system with microwave-based quantum systems, and show that we can achieve similar gate fidelity and gate time using SFQ. This shows that SFQ is a promising approach to implement classical controllers for quantum computers because it can maximize the scalability of quantum computers due to the unique characteristics of SFQ logic, while delivering similar fidelity and performance to that of state-of-the-art microwave-based systems.

To summarize, our key contributions are as follows:

- We present the first study of SFQ-based two-qubit gates that takes into consideration the leakage to higher energy levels.
- We identify and discuss the main challenge in realizing high-fidelity SFQ-based two-qubit gates, which is high leakage to non-computational qubit subspace.
- We develop optimal control methods that suppress the population of higher energy levels in two-qubit gates.
- We study various qubit architectures and configurations in an attempt to engineer a quantum system with low leakage.
- We engineer an SFQ-friendly quantum system, and show that it can achieve similar gate fidelity as that of microwave-based system – a promising result.

The rest of the paper is organized as follows. Sec. 4.2 presents a background on qubit architectures and configurations, quantum optimal control and SFQ-based gates, followed by a discussion on the motivation of our paper. Sec. 4.3 presents our methodology and the results of our detailed study on SFQ-based two-qubit gates, followed by a comparison with microwave-based two-qubit gates. Finally, Sec. 4.4 concludes the paper.

4.2 Background and Motivation

Here we provide details of the physical systems we are targeting in order to distill the basic toolbox of quantum operations available to us for implementing high-performance SFQ-based quantum gates. We motivate our analysis by describing the challenges of implementing high-fidelity gates on realistic quantum systems, the existing strategies for overcoming them on systems with analog control, and prior work on SFQ-based gates aiming to do the same.

4.2.1 Physical system

The evolution of a quantum system is governed by its Hamiltonian. For universal quantum computation, we require that the system provide (1) well-defined *qubits*, or separable two-level quantum subsystems which can be independently initialized and measured; (2) a mechanism for generating entanglement between these qubits; and (3) a method for precisely controlling the system's evolution [DiVincenzo, 2000]. For the purposes of this investigation, we consider pairs of statically-coupled superconducting qubits, with the overall system Hamiltonian,

$$\hat{H} = \sum_q \hat{H}_q + \sum_q \hat{H}_{q,d}(t) + \hat{H}_{qq}, \tag{4.1}$$

where $\hat{H}_q$ are the static Hamiltonian of each qubit, $\hat{H}_{q,d}$ are the contributions of the time-dependent control signals applied to each qubit, and $\hat{H}_{qq}$ is contributed by the inter-qubit coupling (and therefore is responsible for entanglement generation). In the following, we express each of these terms for various hardware configurations in terms of the conjugate flux and charge number quantum operators $\hat{\phi}$ and $\hat{n}$, where $[\hat{n}, \hat{\phi}] = i$.

Transmons

The superconducting transmon qubit comprises a Josephson junction (JJ) shunted to ground with a capacitor in order to minimize its sensitivity to charge noise [Schreier et al., 2008, Koch et al., 2007]. The transmon Hamiltonian can be written as [Krantz et al., 2019],

$$\hat{H}_q = 4E_C\hat{n}^2 - E_J \cos\hat{\phi}, \tag{4.2}$$

where $E_C = e^2/2C_q$ indicates the capacitive energy (with C_q including both the shunt capacitance and that of the JJ), $E_J = I_c\Phi_0/2\pi$ is the Josephson energy of a transmon with critical current I_c, and Φ_0 is the magnetic flux quantum.

The spectrum of the single-transmon system can be found by diagonalizing Eq. (4.2). For $E_C \ll E_J$ the transmon Hamiltonian can be expanded in the SHO Fock state basis,

$$\hat{H}_q \approx \omega_{01}\hat{a}^\dagger\hat{a} + \frac{\alpha}{2}\hat{a}^\dagger\hat{a}(\hat{a}^\dagger\hat{a} - 1), \tag{4.3}$$

where $\omega_{01} = \sqrt{8E_JE_C} - E_C$ is the qubit's oscillation frequency (that is, the energy gap between the ground and first excited state), $\alpha = \omega_{12} - \omega_{01} = -E_C$ is its anharmonicity, and we have made the substitution,

$$\hat{n} = \sqrt[4]{E_J/32E_C}\left(\hat{a} + \hat{a}^\dagger\right), \quad \hat{\phi} = i\sqrt[4]{2E_C/E_J}\left(\hat{a} - \hat{a}^\dagger\right), \tag{4.4}$$

using the standard creation (annihilation) operators $\hat{a}^\dagger$ ($\hat{a}$). Typical transmon qubits are configured with oscillation frequencies $\omega_{01}/2\pi$ between 3 and 6 GHz and anharmonicity $\alpha/2\pi$ between 100 and 300 MHz [Krantz et al., 2019]. The nonzero anharmonicity makes it possible to isolate and address the system's $\{|0\rangle, |1\rangle\}$ subspace, providing the required well-defined two-level qubit.

Frequency-tunable transmons

The single-JJ transmon's oscillation frequency is fixed by its hardware components. We can instead construct a *frequency-tunable* transmon by splitting its single JJ into a pair of parallel junctions (dc-SQUID) and driving an external magnetic flux φ_e through the enclosed loop. In this case the junction energy E_J in Eq. (4.2) is replaced with the flux-dependent effective energy [Krantz et al., 2019],

$$E_J' = \sqrt{(E_{J1} + E_{J2})^2 \cos^2 \varphi_e + |E_{J1} - E_{J2}|^2 \sin^2 \varphi_e}, \tag{4.5}$$

where $E_{J1,2}$ are the Josephson energies of the respective JJs and φ_e is the applied flux in units of Φ_0/π. The applied flux can then be used to tune the qubit's oscillation frequency, or equivalently implement z-axis rotations of the qubit.

For multi-qubit systems, flux control has also been employed to implement two-qubit gates by inducing resonant oscillations between multi-qubit states. For example, by bringing the qubit frequencies together, coherent oscillations between the $|01\rangle$ and $|10\rangle$ state will generate the iSWAP (or $\sqrt{\text{iSWAP}}$) gate, whereas a CZ gate can be implemented using the resonance between the $|11\rangle$ and $|02\rangle$ (or $|20\rangle$) states. The latter case takes advantage of the higher energy levels of the transmon system, allowing the quantum state to temporarily leave the two-level qubit subspace during the execution of the gate. Frequency-tunable transmons enable fast resonant two-qubit operations while decreasing crosstalk by allowing noninteracting qubits to be "parked" at well-separated oscillation frequencies. This tunability comes at the cost of added complexity and sensitivity to magnetic flux noise.

Fluxonium

Though the transmon's nonzero anharmonicity makes it possible to target the two-level (qubit) subspace for quantum computation, its weakness relative to the oscillation frequency makes it prone to leakage to higher level states. Alternative qubit technologies such as fluxonium [Manucharyan et al., 2009] have been shown to increase anharmonicity with minimal cost in terms of noise sensitivity. The fluxonium qubit is constructed similarly to the transmon, but with an additional inductive shunt to ground implemented using an array of Josephson junctions connected in series. The resulting Hamiltonian is written as [Krantz et al., 2019],

$$\hat{H}_q = 4E_C\hat{n}^2 + E_L\hat{\phi}^2 - E_J\cos\Big(\hat{\phi} + \varphi_e\Big), \tag{4.6}$$

where $E_L \ll E_J$ is the inductive energy of the junction array and φ_e is an external magnetic flux through the qubit loop.

Fluxonium's sensitivity to flux noise is minimized at $\varphi_e = 0$ and $\varphi_e = \pi$, where symmetry ensures that the energy dependence on φ_e vanishes to first order. In the latter case, the qubit's oscillation frequency ω_{01} is significantly reduced relative to that of the subsequent transition (ω_{12}), resulting in large, positive anharmonicity. It is less trivial to approximate the fluxonium spectrum analytically; instead we diagonalize Eq. (4.6) numerically to determine the computational basis states and energy spectrum of our system. With typical hardware configurations, fluxonium qubits at $\varphi_e = \pi$ have $\omega_{01} \sim 1$ GHz, while ω_{12} is 2-5 times larger. For remainder of this paper, we assume that fluxonium is operating with a $\varphi_e = \pi$ static bias flux.

Coupling

We focus on systems with static coupling between qubits, such that the interaction Hamiltonian H_{qq} is constant and uncontrollable (as opposed to, for example, tunable coupling systems [Arute et al., 2019] which allow the interaction to be switched on and off on-demand but which would complicate the implementation of an SFQ-based controller). For superconducting qubits coupled via a capacitance C_{qq}, the coupling Hamiltonian in Eq. (4.1) is,

$$\hat{H}_{qq} = g_{qq}\hat{n}_{q0}\hat{n}_{q1}, \tag{4.7}$$

where $g_{qq} = 4e^2 C_{qq}/C_{q_0}C_{q_1}$ quantifies the coupling strength. Expressing Eq. (4.9) in the energy-basis rest frame of the undriven qubit, the dominant matrix elements of the coupling Hamiltonian (after the rotating wave approximation) are,

$$\hat{H}_{qq}^{rf}(t) = J\sum_k c_{k-1,k}^{(0)} c_{l,l-1}^{(1)} e^{i(\omega_{k,k-1}^{(0)} - \omega_{l-1,l}^{(1)})t} \times \big(|k, l-1\rangle\langle k-1, l| + h.c.. \big), \tag{4.8}$$

where J is a normalized coupling constant, and $c_{k,k-1} = c_{k-1,k} \approx \sqrt{k}/2$ for transmons whereas for fluxonium can be computed numerically by diagonalizing each qubit's Hamiltonian (Eq. (4.6)). Though this interaction cannot be disabled, the *effective* coupling strength between qubits is inversely proportional to the separation between their oscillation frequencies due to destructive interference caused by time-averaging the rotating phase in Eq. (4.8). We can therefore preserve the independence of the qubits by designing the system such that the parking frequencies of coupled qubits are well separated.

Drive

The most common architecture for manipulating statically-coupled qubits is to apply microwave control signals directly to the qubits via a coupling capacitor. Given a time-dependent voltage source $V_d(t)$, the microwave drive Hamiltonian is,

$$\hat{H}_{q,d} = V_d(t)\frac{2eC_d}{C_d + C_q}\hat{n}, \tag{4.9}$$

where C_d is the capacitance of the coupling capacitor. Expressed in the rest frame of the qubit and assuming a microwave drive $V_d(t) = \Omega_x(t)V_0 \cos\omega_d t$ (where $\Omega_x(t)$ is the normalized pulse envelope and V_0 absorbs the details of the qubit and drive hardware),

$$H_d^{rf}(t) = \Omega_x(t)\sum_k c_{k+1,k} e^{i(\omega_{k,k+1} - \omega_d)t} |k+1\rangle\langle k| + h.c.. \tag{4.10}$$

The time-dependent phases in Eq. (4.10) allow us to selectively drive a given transition while others are suppressed by the time-dependent phase. For example, continuously driving with $\omega_d = \omega_{01}$ will drive Rabi oscillations in the qubit subspace while the qubit's nonzero anharmonicity $\omega_{12} - \omega_{01} = \alpha$ will suppress the $|1\rangle \leftrightarrow |2\rangle$ transition. However, in order to have a finite gate time, the envelope $\Omega_x(t)$ must itself contain Fourier components which can diminish this suppression by overlapping with higher-order transitions, especially given the transmon's relatively small anharmonicity. Analytical pulse shaping models such as the DRAG scheme [Motzoi et al., 2009] are therefore employed on microwave systems to precisely minimize the overlapping frequency components in the pulse shape.

Using the cross-resonance interaction [Paraoanu, 2006, Rigetti and Devoret, 2010] it is also possible to induce two-qubit entangling operations with precisely detuned control signals applied to one or both qubits, making fixed-frequency transmons and microwave control sufficient for universal quantum computation. Successful quantum computer prototypes have been developed using this control mechanism [Brink et al., 2018]. On these systems, the speed of cross-resonance gates is proportional to the effective coupling between the qubits, creating a tradeoff between gate time and crosstalk.

4.2.2 Microwave optimal control

In practice, the broad control schemes outlined above are insufficient for high-precision quantum gates. Analytical leakage suppression schemes are especially challenging for multi-qubit systems due to the exponentially increasing complexity of the energy spectrum and the contributions of each coupler. This complexity is especially prevalent for cross-resonance gates, in which one transition between multi-qubit states is intentionally driven while all others must be suppressed. Further, it is often desirable to allow the system to evolve outside the two-level subspace during the execution of the gate, as it provides more possible paths for realizing complicated operations within a short gate time (as in the frequency-tunable CZ implementation described above). Typical microwave systems therefore employ search-based optimal-control strategies such as the ubiquitous gradient ascent pulse engineering (GRAPE) tool [Leung et al., 2017] to generate pulse waveforms which implement quantum gates with high fidelity and low leakage.

4.2.3 Fidelity functions

Throughout this work, we quantify the performance of learned gates using two variants of average gate fidelity. In its general form, the average gate fidelity of a quantum operation $\mathcal{E}$ relative to a unitary target gate T is defined,

$$\overline{F}(\mathcal{E}, T) = \int d\psi \, \langle\psi|T^\dagger \mathcal{E}(\psi) T|\psi\rangle \,, \tag{4.11}$$

where the average is over the normalized Haar distribution of quantum states. Because we are interested only in how the gate affects *qubits*, we would like our fidelity metric to (1) be agnostic to the behavior of the gate when applied to states outside the qubit subspace, and (2) penalize gate-induced leakage from within the computational subspace. We therefore define,

$$\mathcal{E}(\psi) = \Pi U \Pi \, |\psi\rangle\!\langle\psi| \, \Pi U^\dagger \Pi, \tag{4.12}$$

where U is the simulated (unitary) evolution including higher level states, and,

$$\Pi = (|0\rangle\!\langle 0| + |1\rangle\!\langle 1|)^{\otimes n}, \tag{4.13}$$

projects it into the qubit subspace. The average in Eq. (4.11) is then taken over just states $|\psi\rangle$ in the two-level subspace $SU(2^n)$, so that a *subspace-averaged gate fidelity* can be calculated [Ghosh, 2011],

$$\overline{F_1}(U, T) = \frac{\mathrm{tr}\left(\Pi U \Pi U^\dagger \Pi\right) + \mathrm{tr}\left(T \Pi U^\dagger \Pi\right)^2}{2^{2n} + 2^n}, \tag{4.14}$$

Because of the constrained control set available with SFQ control, we would further like to broaden our search target as much as possible. In practice, single-qubit Z rotations can

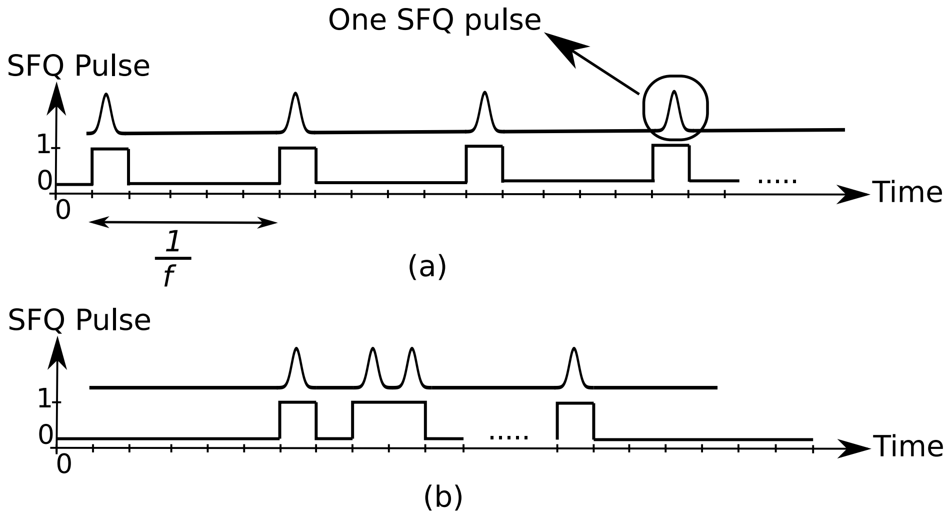


Figure 4.1: Bit representation of SFQ pulse trains. (a) coherent pulses are applied to the qubit (1 pulse per qubit oscillation period) to perform rotations around the y axis. (b) a bitstream found by genetic algorithm to perform arbitrary unitary. Bitstreams are processed one bit at a time; if the bit is “0”, no pulse is applied to the qubit, and if the bit is “1”, one SFQ pulse is applied to the qubit.

often be commuted through subsequent gates or implemented virtually. We therefore define the *Z-independent gate fidelity*, which is independent of trailing Z-rotations:

$$\overline{F_2}(U,T) = \sup_{\vec{\alpha}} \overline{F_1}\big(Z_{\vec{\alpha}}\tilde{U}, T\big), \tag{4.15}$$

$$Z_{\vec{\alpha}} = Z(\alpha_1) \otimes \cdots \otimes Z(\alpha_n). \tag{4.16}$$

Finally, we can explicitly quantify leakage by computing the probability of measuring a state outside the qubit subspace after applying the gate to a state initially within that space. Averaging over the uniform distribution of all possible two-level input states, the *average leakage* can be calculated,

$$\overline{L}(U) = 1 - \frac{\operatorname{tr}\Big(\Pi U \Pi U^{\dagger} \Pi\Big)}{2^n}. \tag{4.17}$$

From Eqs. (4.14), (4.15) and (4.17) one can show that $\overline{F_1}(U,T) \leq \overline{F_2}(U,T) \leq 1 - \overline{L}(U)$, so that as desired our fidelity metrics are upper-bound by the degree of leakage.

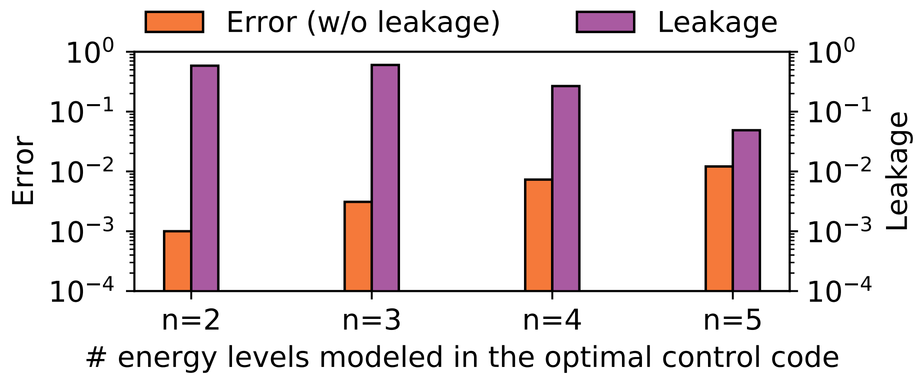


Figure 4.2: Error and leakage of the best SFQ-based CZ gate with 20 ns gate time found by the genetic algorithm on transmon qubit devices with Ω_x control fields. Error is computed using Eq. (4.14) and takes into account only the n levels on which the gate was learned. Leakage is computed using Eq. (4.17) and considers higher levels. Thus, low error does not necessarily translate to low leakage.

4.2.4 SFQ control

It has been proposed that quantum gates be implemented by applying SFQ pulses to the qubit directly in place of analog microwave control signals. The gate implementation is then described by a binary pulse train as shown in Fig. 4.1, where in each cycle of the SFQ clock a pulse is either applied or not applied to each qubit. For two-qubit gates, we can apply different pulse trains to each qubit.

A single SFQ pulse is a rapid Gaussian voltage waveform,

$$V_d(t) = \frac{\Phi_0}{\sqrt{2\pi\tau^2}} e^{-t^2/2\tau^2}, \tag{4.18}$$

with a total area of exactly $\int dt V_d(t) = \Phi_0$ and a typical pulse width of $\tau = 0.25$ ps. Approximating $V_d(t) \approx \Phi_0 \delta(t)$ and considering Eq. (4.9) in the energy-basis rest frame of the transmon, we expect a single pulse at time t_0 to implement the instantaneous gate,

$$U_x^{rf} = \exp\left\{ -i\delta_\theta \sum_k e^{i\omega_{k,k+1}t_0} \sqrt{k}\, |k\rangle\langle k-1| + h.c..\right\}, \tag{4.19}$$

where δ_θ is the *tip angle*, indicating the rotation angle induced by a single pulse in the qubit subspace. The tip angle is typically in the range of 10^{-3} to 10^{-1} radians [Li et al., 2019, Dalgaard et al., 2020], and is directly configurable via choices of qubit and coupling hardware; in our analysis we find that this configuration is extremely important for achieving high-performance SFQ gates.

In order to expand our narrow SFQ control toolset, we also consider an SFQ-based σ_z operation for frequency-tunable transmons. In this case, rather applying pulses to the qubit via a capacitive coupler, we assume that they are inductively coupled to the split transmon's dc-SQUID loop. Approximating a single pulse as a delta function, the resulting gate is then simply,

$$U_z = \sum_k e^{ik\delta_z} |k\rangle\langle k| , \tag{4.20}$$

where δ_z is the z-axis tip angle determined by the hardware configuration. In this case, achieving a non-negligible tip angle may require additional filter hardware in order to both broaden the SFQ pulse shape and mitigate distortion caused by the mutual inductance between the qubit and the control line (as discussed in [Koch et al., 2007]). With a rough calculation we find that $\delta_z \sim 0.03$ should easily be achievable using existing techniques (such as [Semenov and Averin, 2003]). Though on a single-qubit system σ_z control would not be sufficient for universal quantum control, it turns out to be remarkably effective for realizing two-qubit gates when combined with the free evolution due to the static coupler.

Unlike the microwave drive, with SFQ pulses we cannot simply select a drive frequency in order to selectively drive a given transition while off-resonant transitions are suppressed by the rotating phase in Eq. (4.19). Instead, we are limited to selecting discrete clock cycles t in which to apply $U_x(t)$. If we constrain our system to the qubit subspace ($k \in \{0, 1\}$), Eq. (4.19) is simply a unitary rotation $e^{-i\delta_\theta(\cos(\omega t)X+\sin(\omega t)Y)}$ by angle δ_θ about a time-dependent axis on the xy-plane. In this case, the problem is reduced to one of single-qubit gate composition (taking as basis gates the set of lab-frame single-clock-cycle unitaries generated by applying pulses to each possible subset of qubits), for which many analytic and search methods have been studied. Empirically, in both prior work and our own examination it appears that pulse trains implementing high-fidelity, low leakage single-qubit gates are still readily discoverable when we model the system with additional energy states [Li et al., 2019, McDermott et al., 2018, Liebermann and Wilhelm, 2016]. This is perhaps unsurprising observing Eq. (4.19); though each pulse may result in some population transfer out of the qubit subspace, the simple energy spectrum of a single qubit near its ground state makes it reasonable to expect symmetries to exist in which pairs or small groups of pulses will generate destructive interference in the non-qubit subspace (in fact, such symmetries were employed explicitly as part of the search algorithm outlined in [Li et al., 2019]).

4.2.5 Prior work on SFQ-based gates and the motivation of this paper

There has been detailed analysis of SFQ-based single-qubit gates in the literature [McDermott et al., 2018, Li et al., 2019, Liebermann and Wilhelm, 2016]. Prior work has studied the SFQ-based coherent control of qubits, and demonstrated that we can perform rotations around the X or Y axis by applying SFQ pulses every qubit oscillation period [McDermott et al., 2018, Leonard et al., 2019]. However, this approach leads to leakage to higher energy levels, thus prior work utilized a genetic algorithm to find better SFQ-based gates with low leakage and short gate time [McDermott et al., 2018, Liebermann and Wilhelm, 2016]. They show that taking into account three lowest energy levels of the qubit in their model is sufficient to realize low-leakage gates using SFQ pulses.

In [McDermott et al., 2018], the authors envision the possibility of performing SFQ-based two-qubit gates. In [Dalgaard et al., 2020], the authors implement a quantum optimal control version of the AlphaZero learning algorithm [Silver et al., 2017] to optimize the quantum dynamics, and use SFQ-based optimal control as a benchmark in their study. The authors show that they can find SFQ pulse trains to do $\sqrt{ZX}$ gate with high fidelity. However, their model of a two-qubit quantum system does not take into consideration the leakage out of the computational subspace.

Fig. 4.2 shows the importance of taking higher energy levels into consideration when learning SFQ pulses to perform two-qubit gates. In each case, we report the error of the best SFQ-based two-qubit gate we find with a genetic algorithm when modeling the quantum system using n energy levels. We then simulate the learned bitstream using a model that allows for evolution to higher energy levels, and report the leakage (Eq. (4.17)) of the resulting gate.

We can easily find SFQ-based two-qubit gates with 0.999 fidelity with $n = 2$ (consistent with prior work [Dalgaard et al., 2020]). However, we find that the learned SFQ pulse train results in a gate with high leakage when allowed to evolve out of the two-level subspace. This is an expected result—prior work has shown that we need to consider $n = 3$ to find low-leakage single-qubit gates [Liebermann and Wilhelm, 2016]. What is more surprising is that, as shown in Fig. 4.2, the genetic algorithm cannot find SFQ-based two-qubit gates with low leakage even with $n = 5$. Instead, even when we learn bitstreams using $n = 5$, if we simulate the same bitstreams on a system with more than n energy levels, the resulting quantum evolution will leak into the additional levels, resulting in a gate with both poor accuracy and high leakage. Thus, unlike the single-qubit gate case, taking the higher energy level into consideration alone is not sufficient.

In this paper, we characterize the requirements of realizing high-fidelity SFQ-based two-qubit gates. We develop quantum optimal control methods, and also investigate various qubit architectures and configurations in an attempt to engineer an SFQ-friendly quantum system that can perform high-fidelity two-qubit gates.

4.3 Detailed study of SFQ-based two-qubit gates

In this section we first discuss our methodology, followed by the results of our study on SFQ-based two-qubit gates under various qubit architectures and configurations. Then, we compare our results with that of microwave-based quantum systems.

Table 4.1: The parameters used in the genetic algorithm.

Population size	70
Selection size	60
Mutation probability	0.001
Maximum number of iterations	200,000
Target fidelity	0.999

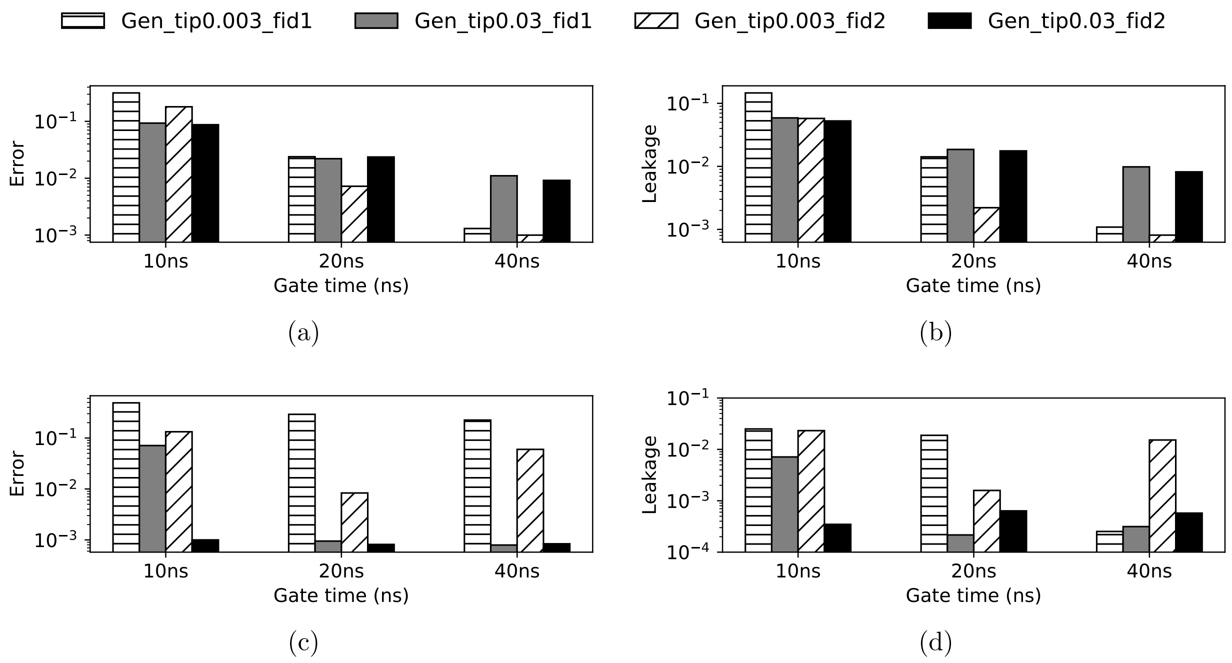
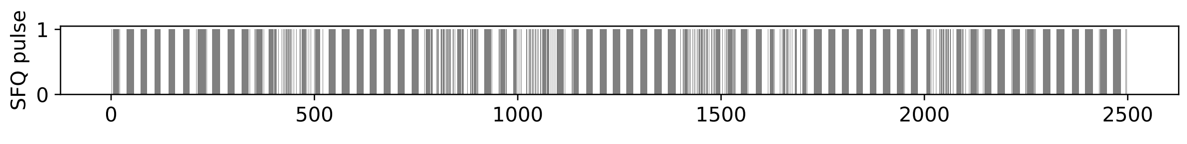


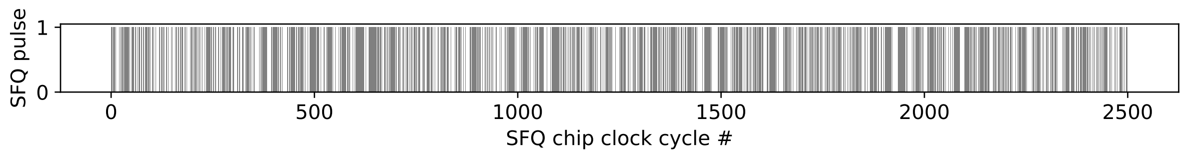
Figure 4.3: Error and leakage of the best SFQ-based CZ gate found by the genetic algorithm for transmon qubit devices with Ω_x control fields (plots a and b), and transmon qubit devices with Ω_z control fields (plots c and d). Error is calculated as $1 - fidelity$, and leakage is calculated using Eq. (4.17). Two different tip angles and two fidelity functions are used in our optimal control method (see Sec. 4.2 for the details of our fidelity functions). We run the simulations with $n = 5$ energy levels, and suppress the population of higher energy levels in our optimal control method.

4.3.1 Methodology

We model SFQ-based quantum operations by numerically integrating the relevant system Hamiltonian (Eq. (4.1)) over a single SFQ clock cycle for each possible combination of input pulses. The learning algorithm then searches for pulse streams corresponding to optimal sequences of these basis operations. In order to avoid sequences which would spill into higher levels if made available (as described in Section 4.2.5), we generate each unitary



(a)



(b)

Figure 4.4: Bit representation of SFQ bitstreams applied to qubit1 (plot a) and qubit2 (plot b) on a transmon system with Ω_x control fields and 0.003 tip angle in order to realize a CZ gate with 20 ns gate time. Each SFQ chip clock cycle is 8 ps.

evolution using extra energy levels, and then project out the extra levels after each pulse in the sequence. The resulting non-unitarity of the evolution then gets quantified by our fidelity metrics as additional leakage, forcing the algorithm to prioritize sequences which are constrained to the given number of energy levels.

We use a variant of the genetic algorithm used in prior work [Dalgaard et al., 2020] to find a train of SFQ pulses to perform quantum gates. The parameters of the genetic algorithm is summarized in Table 4.1. The genetic algorithm starts with a population of random SFQ pulse trains, and in each iteration, a number of parent pulse trains from the population are selected for generating new pulse trains based on a crossover function. Finally, if the fidelity is improved in the new SFQ pulse trains, they are replaced with the worst SFQ pulse trains in the population.

We use a variant of the GRAPE code used in [Leung et al., 2017] to find microwave pulses to perform quantum gates. We use the cost functions presented in [Leung et al., 2017] in order to suppress the occupation of forbidden states. Similar to the SFQ case, we set the target gate fidelity to 0.999.

4.3.2 Entangling SFQ-based two-qubit gates on transmon qubit devices

In this section, we present the results of our analysis on transmon qubit devices. Similar to [Leung et al., 2017], we use qubit frequencies of $\omega_{01}^{(0)}/2\pi = 3.9$ and $\omega_{01}^{(1)}/2\pi = 3.5$ GHz, anharmonicity of $\alpha/2\pi = -225$ MHz, and $n = 5$ in our study on transmons. We report the results for coupling strength of $J/2\pi = 50$ MHz in our main results and then perform a sensitivity analysis on the coupling strength. Note that we show the results for transmon with Ω_x control fields and transmon with Ω_z control fields separately in order to study the effectiveness of each control field on realizing entangling two-qubit gates.

CZ gate on transmon qubits with Ω_x control fields

Fig. 4.3(a) and 4.3(b) respectively show the error and leakage of the best SFQ pulse train found using the genetic algorithm to perform a CZ gate on transmons with Ω_x control fields. The leakage to higher energy levels is suppressed by the physical model employed in our optimal control method (as described in Section 4.3.1). The error numbers reported in this plot take leakage to higher energy levels into consideration, thus, low error translates into low leakage as shown in Fig. 4.3. We run the genetic algorithm with the two fidelity functions described in Section 4.2.3 (denoting *subspace-averaged gate fidelity* as *fid1* and *Z-independent gate fidelity* as *fid2*), two tip angles of 0.003 and 0.03 (similar to the numbers reported in the literature [Li et al., 2019, Dalgaard et al., 2020]), and three gate times of 10 ns, 20 ns, and 40 ns. Fig. 4.4 shows an example of the SFQ bitstreams that are learned using the genetic algorithm.

Our results show that it is hard to find high-fidelity CZ gates while suppressing the leakage to higher energy levels using the 0.03 tip angle with either fidelity function. By decreasing the tip angle to 0.003, we are able to realize a CZ gate with 0.999 fidelity and 40 ns gate time. Decreasing the tip angle means the amount of energy deposited into the qubit with each SFQ pulse decreases, thus, the required gate time to perform high-fidelity quantum operations increases.

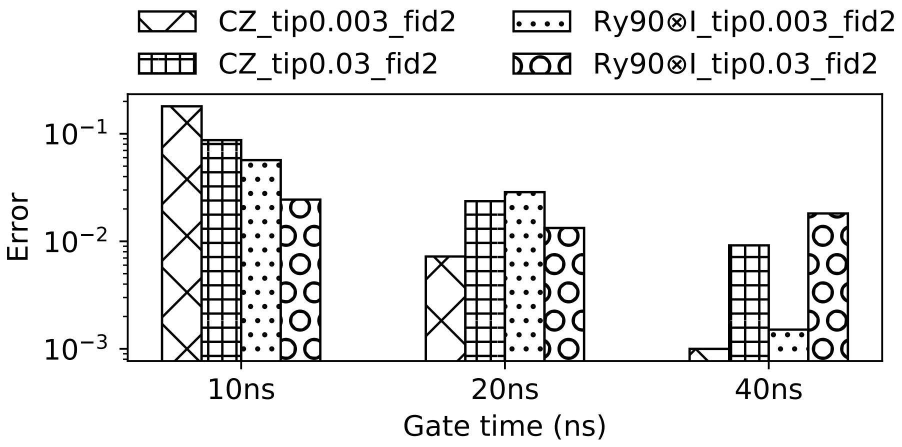


Figure 4.5: Error of the best SFQ-based CZ gate (entangling two-qubit gate) and Ry90⊗I gate (non-entangling two-qubit gate) found by the genetic algorithm for transmon qubit devices with only Ω_x control fields.

In general, *fid2* results in better SFQ-based pulse trains than *fid1*, indicating that the broader target provided by *fid2* is indeed more friendly to the highly-constrained nature of SFQ-based gate implementation.

CZ gate on transmon qubits with Ω_z control fields

Fig. 4.3(c) and 4.3(d) show the error and leakage results of transmon devices with Ω_z control fields, respectively. Here, we apply a Ω_z control field only to qubit2 (which is sufficient to realize high-fidelity CZ gates). We observe a significant reduction in the amount of leakage to higher energy levels in the case of transmon devices with Ω_z control fields compared to that of transmon devices with Ω_x control fields. Since the leakage is low in this case, we can afford to use higher tip angles in order to perform fast gates. Our results show that we can realize high-fidelity CZ gates with <0.001 error and <0.001 leakage with 0.03 tip angle and 10 ns gate time with *fid2* (longer gate time is required with *fid1*).

Our findings show that Ω_z control field with 0.003 tip angle is not sufficient to realize high-fidelity CZ gates. However, the gates that we find do have low leakage in some cases;

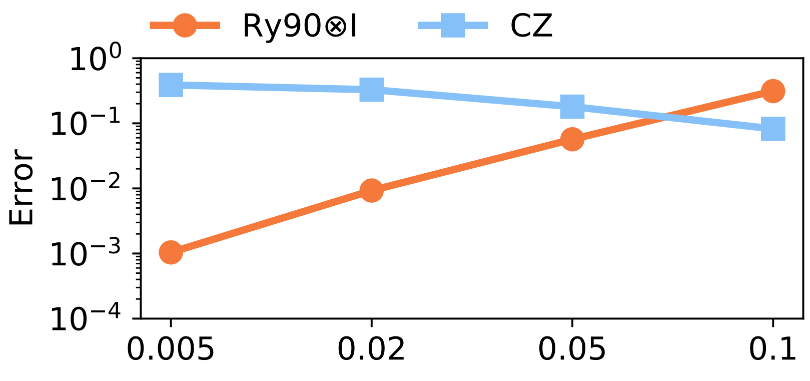
although low error translates to low leakage because we take into consideration the leakage to higher energy levels in calculating the error values, the opposite is not necessarily true (for example, the identity gate has high error if we calculate its overlap with CZ gate, but it has low leakage to higher energy levels).

4.3.3 Realizing both entangling and non-entangling SFQ-based two-qubit gates on transmon devices

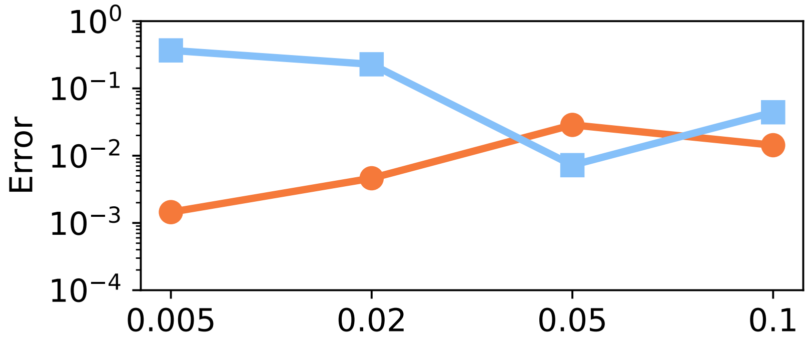
So far, we demonstrated that we can realize high-fidelity CZ gates with low leakage and short gate time using transmon qubit devices, which is a promising result. However, it is essential to ensure that we can also realize high-fidelity one-qubit gates in our two-qubit quantum system (i.e., non-entangling two-qubit gates). Next, we study the requirements of a system that can perform both entangling and non-entangling SFQ-based two-qubit gates. Transmon system with only Ω_z control fields is suitable to realize high-fidelity entangling two-qubit gates, but it does not provide enough control to perform arbitrary single-qubit gates, which as expected leads to non-entangling two-qubit gates with high error ($> 10^{-1}$ error). Thus, we need more than just Ω_z control fields to realize both entangling and non-entangling two-qubit gates. Next, we investigate two systems as possible candidates to achieve this goal.

Transmon with Ω_x control fields

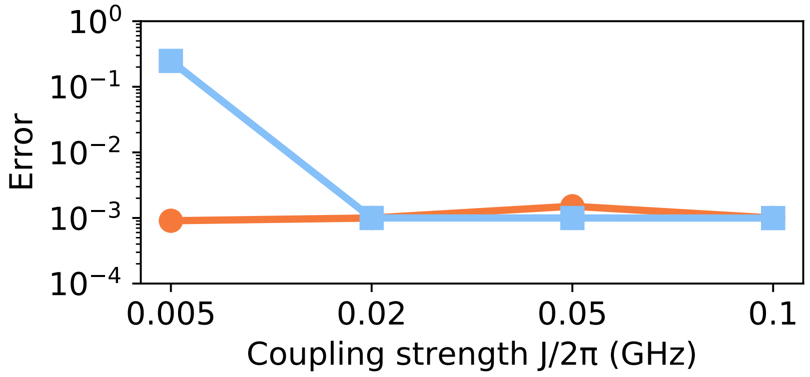
Prior work demonstrated SFQ-based single-qubit gates with <20 ns gate time on transmon devices with only Ω_x control fields [Li et al., 2019, Liebermann and Wilhelm, 2016]. In addition, we showed earlier that we can use transmon devices with Ω_x control fields to perform high-fidelity CZ gates with 40 ns gate time and low tip angle. A natural question arises: can we engineer a transmon system with Ω_x control fields that can perform both entangling and non-entangling SFQ-based two-qubit gates with high fidelity? Fig. 4.5 shows



(a)



(b)



(c)

Figure 4.6: Sensitivity analysis on qubit coupling strength in transmon system with Ω_x control fields. The results are shown for 0.003 tip angle and 10 ns (plot a), 20 ns (plot b), and 40 ns (plot c) gate times. The SFQ bitstreams are learned with *fid2*.

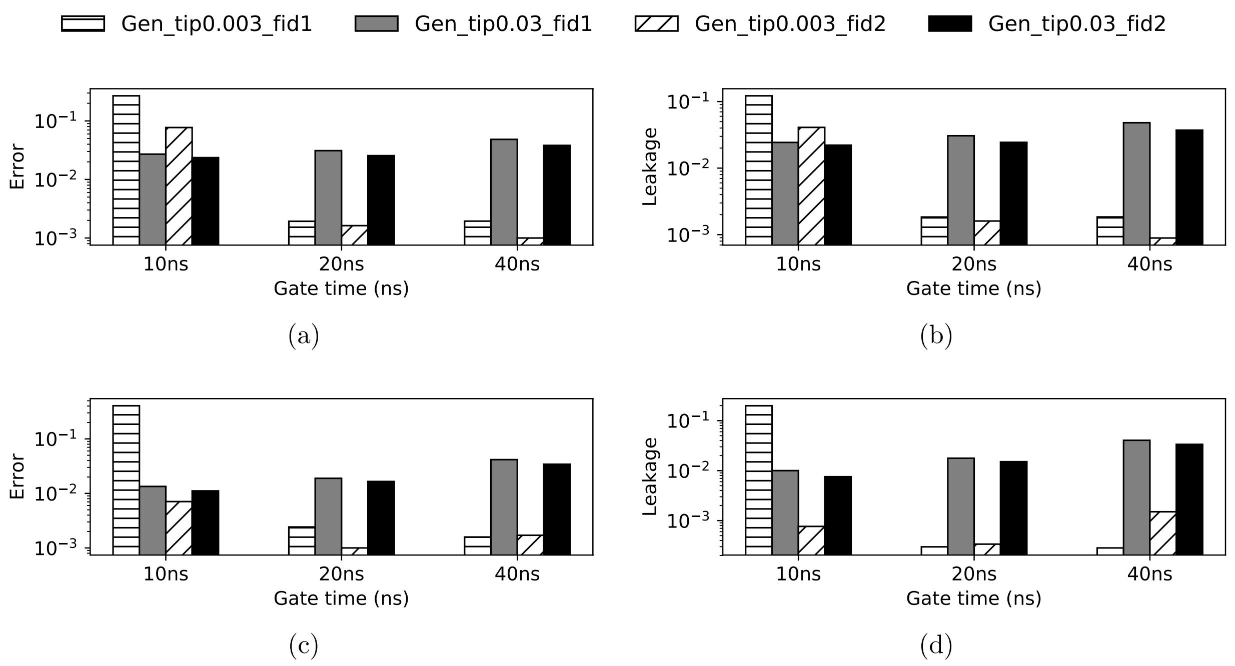
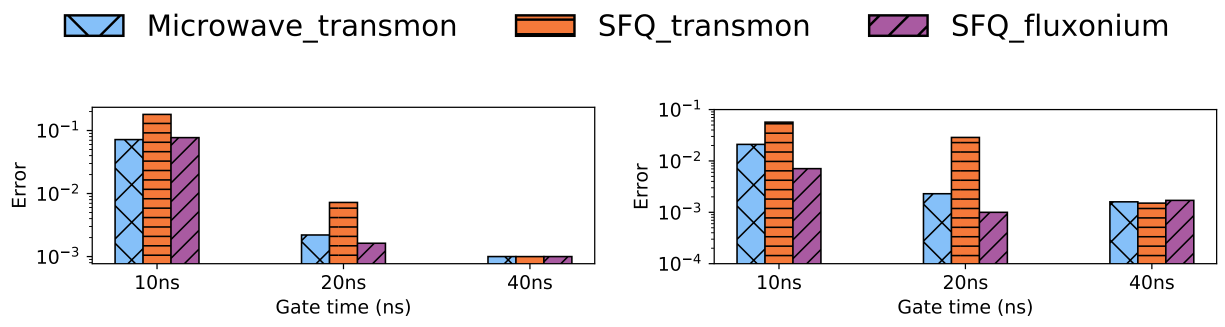


Figure 4.7: Error and leakage of the best SFQ-based CZ gate (plots a and b) and Ry90⊗I gate (plots c and d) found by the genetic algorithm for fluxonium qubit devices with Ω_x control fields.

the error results of the best SFQ-based CZ gate (entangling two-qubit gate) and Ry90⊗I gate (non-entangling two-qubit gate) found on transmon system with Ω_x control fields using the genetic algorithm. Our results show that we can realize both CZ and Ry90⊗I gates with high fidelity with 0.003 tip angle and 40 ns gate time.

One interesting observation is that the gate time of Ry90⊗I is longer than the gate time of the SFQ-based single qubit gates reported in prior work [Li et al., 2019, Liebermann and Wilhelm, 2016]. In general, it is more challenging to realize precise single-qubit gates in a two-qubit system compared to the one-qubit systems because of crosstalk with the neighbor qubit. We can reduce crosstalk and achieve faster single-qubit gate times by reducing the coupling strength (J), however this will in turn complicate the realization of two-qubit entangling gates. Fig. 4.6 shows a sensitivity analysis on the coupling strength. Our results show that realizing high-fidelity CZ gate requires higher coupling strengths and realizing high-fidelity Ry90⊗I gate requires lower coupling strengths. A coupling strength of $J/2\pi = 50$ MHz is a



(a) (b)

Figure 4.8: Error comparison between microwave-based gates obtained using Grape code and SFQ-based gates obtained using genetic algorithm (with 0.003 tip angle). The results are reported for CZ gate (plot a) and Ry90⊗I gate (plot b).

sweet spot that works well for both CZ and Ry90⊗I gates in our results.

Transmon with both Ω_x and Ω_z control fields

One possible configuration is to dedicate both Ω_x and Ω_z control fields to the transmon qubit devices, which would potentially lead to SFQ-based gates with higher fidelity and shorter gate time. However, we note that this comes at the cost of hardware complexity and heightened sensitivity to magnetic flux noise.

4.3.4 SFQ-based two-qubit gates on fluxonium qubit devices

In this section, we investigate fluxonium qubit devices as a possible candidate to realize both SFQ-based entangling and non-entangling gates with low leakage and short gate time using Ω_x control fields. Our model for the fluxonium devices assumes qubit1 (qubit2) is configured with $E_J = 5.5$ (5.7), $E_C = 1.5$ (1.2) and $E_L = 1.0$, and a static $\varphi_e = \pi$ external flux. Fig. 4.7 shows the results of our study on fluxonium devices with Ω_x control fields.

Fig. 4.7(a) and 4.7(b) show the error and leakage results of an SFQ-based CZ gate, respectively. Our results show that we can realize high-fidelity CZ gates with a gate time of 20 ns thanks to the low leakage of fluxonium devices. Similar to the case of transmons with

Ω_x control fields, better results are achieved with lower tip angle. Fig. 4.7(c) and 4.7(d) show the error and leakage results of an SFQ-based Ry90⊗I gate, respectively. Our results show that the genetic algorithm can find high-fidelity gates with 20 ns gate time.

The fluxonium results show the feasibility and effectiveness of both entangling and non-entangling two-qubit gates with short gate time and low error and leakage using only Ω_x control fields.

4.3.5 Comparison with microwave-based gates

Finally, we compare our results with that of microwave-based gates obtained from the GRAPE algorithm [Leung et al., 2017]. Fig. 4.8 shows the error results for CZ gate and Ry90⊗I gate for three designs: (1) microwave-based design with transmon devices; (2) SFQ-based design with transmon devices; (3) SFQ-based design with fluxonium devices. We learn the SFQ pulse trains with the *Z-independent gate fidelity* function. In the microwave case, *subspace-averaged gate fidelity* is sufficient to realize high-fidelity gates.

The results reported in Fig. 4.8 show that the SFQ-based design with fluxonium has similar error to that of the microwave-based design. The SFQ-based design with transmons has similar error to the other two systems for 40 ns gate time, and higher error than the other two systems for 10 ns and 20 ns gate times. The comparison results show that we can perform high-fidelity SFQ-based gates with similar gate time and gate fidelity to that of microwave-based system. Thus, SFQ is a promising approach to implement classical controllers as they can deliver quantum computers with both high scalability and high fidelity.

4.4 Conclusion

Superconducting Single Flux Quantum (SFQ) is a classical logic technology which is proposed in the literature to implement in-fridge classical controllers in order to maximize the scalability of quantum computers. In this paper, we demonstrate the first thorough analysis of SFQ-based two-qubit gates – a key remaining step in realizing SFQ-based universal quantum computing. Our results show that despite the severe challenges of realizing SFQ-based two-qubit gates, they are both feasible and effective if we carefully design our quantum optimal control method and qubit architecture. We characterize the requirements of such gates, and carefully engineer SFQ-friendly quantum systems that can perform both two-qubit gates and single-qubit gates with high fidelity on a system with fixed coupling (tunable coupling would potentially provide further isolation and less crosstalk, however, further research is required to investigate the effectiveness of such couplers on an SFQ-based system). More importantly, we demonstrate that the fidelity and gate time of these gates are comparable to that of microwave-based gates – these results show that SFQ approach can potentially not only increase the scalability of quantum machines but also maintain the fidelity and effectiveness of quantum gates, thus SFQ is a promising approach to implement classical controllers for scalable quantum machines.

CHAPTER 5
CONCLUSION

Many quantum computer prototypes have been manufactured in recent years, however, these prototypes have sever scalability challenges due to the massive costs of generating and routing the control signals from a classical controller at room temperature to the quantum chip inside the dilution refrigerator. In this thesis, we investigated the implications of pushing the classical controller to the dilution refrigerator, and demonstrated that such classical controllers can be utilized to increase the computational power and scalability of quantum machines.

First, we utilized Superconducting Single Flux Quantum (SFQ), which is a classical logic technology that can work inside the fridge with low power consumption and ultra-high speed, to implement an approximate quantum error correction. We designed an SFQ-based decoding accelerator for stabilizer codes, leveraging the unique characteristics of the SFQ technology in order to make sure that the decoder power and area are within the cryogenic cooling system budget. We demonstrated that our accelerator can boost the computational power of near-term quantum machines by over a factor of 3,402.

Second, we presented *DigiQ*, which is the first system-level design of a Noisy Intermediate Scale Quantum (NISQ)-friendly classical controller based on SFQ logic. *DigiQ* receives digital instructions from the room temperature and performs quantum gates by generating control signals inside the fridge; *DigiQ* performs single-qubit gates using SFQ pulses and two-qubit gates using electrical currents generated inside the fridge. By co-designing the quantum gate decompositions and SFQ-based implementation of those decompositions, we developed a SIMD architecture that can operate within the tight power and area budget of dilution refrigerators at large scales (>42,000-qubit), while providing good quantum algorithmic performance. We presented software solutions to address the quantum gate calibration challenges of SIMD hardware.

Third, we conducted a research on performing two-qubit operations using only SFQ pulses. We demonstrated that although SFQ-based two-qubit gates have high tendency to leak to qubit non-computational subspace, we can realize such gates with high-fidelity and low leakage by carefully designing optimal control methods and qubit architectures. We show that after engineering an SFQ-friendly quantum system, we can achieve similar gate time and gate fidelity to microwave-based quantum gates.

The results of this thesis show that in-fridge controllers based on SFQ logic can play an important role in future of quantum computing by increasing the scalability and computational power of quantum machines. The key takeaways from the results of this thesis is that 1) we need to design SFQ-based in-fridge controllers by taking into consideration the opportunities (efficient on-chip broadcast and ultra-fast clock) and challenges (lack of dense memory/logic) of SFQ in order to make sure they can operate within the tight power and area budget of dilution refrigerators; 2) quantum-classical co-design is essential to ensure that in-fridge controllers preserve quantum algorithmic performance; 3) novel solutions at the software/compiler level can mitigate the limitations of in-fridge controllers in the calibration of quantum gates.

REFERENCES

Héctor Abraham, AduOffei, Rochisha Agarwal, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, et al. Qiskit: An open-source framework for quantum computing, 2019.

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature (London)*, 574(7779), 2019.

Paul Baireuther, MD Caio, B Criger, Carlo WJ Beenakker, and Thomas E O'Brien. Neural network decoder for topological color codes with circuit level noise. *New Journal of Physics*, 21(1), 2019.

Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, 1995.

Rami Barends, Julian Kelly, Anthony Megrant, Andrzej Veitia, Daniel Sank, Evan Jeffrey, Ted C White, Josh Mutus, Austin G Fowler, Brooks Campbell, et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500, 2014.

Rami Barends, Alireza Shabani, Lucas Lamata, Julian Kelly, Antonio Mezzacapo, Urtzi Las Heras, Ryan Babbush, Austin G Fowler, Brooks Campbell, Yu Chen, et al. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, 534(7606):222–226, 2016.

Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26 (5):1411–1473, 1997.

Koen Bertels, Aritra Sarkar, and Imran Ashraf. Quantum computing–from nisq to pisq. *arXiv preprint arXiv:2106.11840*, 2021.

Lev S Bishop, Sergey Bravyi, Andrew Cross, Jay M Gambetta, and John Smolin. Quantum volume. 2017.

Sergey Bravyi, Martin Suchara, and Alexander Vargo. Efficient algorithms for maximum likelihood decoding in the surface code. *Physical Review A*, 90(3), 2014.

Markus Brink, Jerry M Chow, Jared Hertzberg, Easwar Magesan, and Sami Rosenblatt. Device challenges for near term superconducting quantum processors: frequency collisions. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 6.1.1–6.1.3. IEEE, 2018.

Darren K Brock and Michael S Pambianchi. A 50 ghz monolithic rsfq digital phase locked loop. In *2000 IEEE MTT-S International Microwave Symposium Digest (Cat. No. 00CH37017)*, volume 1, pages 353–356. IEEE, 2000.

Kenneth R Brown, Jungsang Kim, and Christopher Monroe. Co-designing a scalable quantum computer with trapped atomic ions. *npj Quantum Information*, 2(1):1–10, 2016.

Christopher Chamberland and Pooya Ronagh. Deep neural decoders for near term fault-tolerant experiments. *Quantum Science and Technology*, 3(4), 2018.

W Chen, AV Rylyakov, Vijay Patel, JE Lukens, and KK Likharev. Rapid single flux quantum t-flip flop operating up to 770 ghz. *IEEE Transactions on Applied Superconductivity*, 9 (2):3212–3215, 1999.

Jerry M Chow, Jay M Gambetta, AD Córcoles, Seth T Merkel, John A Smolin, Chad Rigetti, S Poletto, George A Keefe, Mary B Rothwell, JR Rozen, et al. Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits. *Physical review letters*, 109(6), 2012.

Haolin Cong, Mingye Li, and Massoud Pedram. An 8-bit multiplier using single-stage full adder cell in single flux quantum circuit technology. *IEEE Transactions on Applied Superconductivity*, pages 1–1, 2021.

A. D. Córcoles, Jay M. Gambetta, Jerry M. Chow, John A. Smolin, Matthew Ware, Joel Strand, B. L. T. Plourde, and M. Steffen. Process verification of two-qubit quantum gates by randomized benchmarking. *Phys. Rev. A*, 87, 2013.

Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv*, 2004.

Mogens Dalgaard, Felix Motzoi, Jens Jakob Sørensen, and Jacob Sherson. Global optimization of quantum dynamics with alphazero deep exploration. *npj Quantum Information*, 6 (1):6, 2020.

Poulami Das, Christopher A Pattison, Srilatha Manne, Douglas Carmean, Krysta Svore, Moinuddin Qureshi, and Nicolas Delfosse. A scalable decoder micro-architecture for fault-tolerant quantum computing. *arXiv preprint arXiv:2001.06598*, 2020.

Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *arXiv e-prints*, 2005.

Nicolas Delfosse and Naomi H Nickerson. Almost-linear time decoding algorithm for topological codes. *arXiv preprint arXiv:1709.06218*, 2017.

Nicolas Delfosse and Gilles Zémor. Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel. *arXiv preprint arXiv:1703.01517*, 2017.

Johannes Arnoldus Delport, Kyle Jackman, Paul Le Roux, and Coenrad Johann Fourie. Josim - superconductor spice simulator. *IEEE Transactions on Applied Superconductivity*, 29(5):1–5, 2019.

Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002.

David P. DiVincenzo. The Physical Implementation of Quantum Computation. *Fortschritte der Physik*, 48(9-11):771–783, 2000.

Doratha E Drake and Stefan Hougardy. A simple approximation algorithm for the weighted matching problem. *Information Processing Letters*, 85(4):211–213, 2003.

Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. A logarithmic-depth quantum carry-lookahead adder. *Quantum Information & Computation*, 6(4&5), 2006.

Guillaume Duclos-Cianci and David Poulin. Fast decoders for topological quantum codes. *Physical review letters*, 104(5), 2010a.

Guillaume Duclos-Cianci and David Poulin. A renormalization group decoding algorithm for topological quantum codes. In *2010 IEEE Information Theory Workshop*, pages 1–5. IEEE, 2010b.

Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965a.

Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965b.

Emerson S Fang. A josephson integrated circuit simulator (jsim) for superconductive electronics application. In *Extended Abstracts of 1989 International Superconductivity Electronics Conf.(The Japan Society of Applied Physics, Tokyo, 1989)*, 1989.

Caroline Figgatt, Aaron Ostrander, Norbert M Linke, Kevin A Landsman, Daiwei Zhu, Dmitri Maslov, and Christopher Monroe. Parallel entangling operations on a universal ion-trap quantum computer. *Nature*, 572(7769):368–372, 2019.

C. J. Fourie, K. Jackman, M. M. Botha, S. Razmkhah, P. Febvre, C. L. Ayala, Q. Xu, N. Yoshikawa, E. Patrick, M. Law, Y. Wang, M. Annavaram, P. Beerel, S. Gupta, S. Nazarian, and M. Pedram. Coldflux superconducting eda and tcad tools project: Overview and progress. *IEEE Transactions on Applied Superconductivity*, 29(5):1–7, 2019.

Austin G Fowler. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $o(1)$ parallel time. *arXiv preprint arXiv:1307.1740*, 2013.

Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), 2012a.

Austin G Fowler, Adam C Whiteside, and Lloyd CL Hollenberg. Towards practical classical processing for the surface code: timing analysis. *Physical Review A*, 86(4), 2012b.

Austin G Fowler, Adam C Whiteside, and Lloyd CL Hollenberg. Towards practical classical processing for the surface code. *Physical review letters*, 108(18), 2012c.

Austin G Fowler, Adam C Whiteside, Angus L McInnes, and Alimohammad Rabbani. Topological code autotune. *Physical Review X*, 2(4), 2012d.

David P Franke, James S Clarke, Lieven MK Vandersypen, and Menno Veldhorst. Rent's rule and extensibility in quantum computing. *arXiv preprint arXiv:1806.02145*, 2018.

X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels. An experimental microarchitecture for a superconducting quantum processor. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, page 813–825, New York, NY, USA, 2017.

Xiang Fu, MA Rol, CC Bultink, J van Someren, Nader Khammassi, Imran Ashraf, RFL Vermeulen, JC de Sterke, WJ Vlothuizen, RN Schouten, et al. A microarchitecture for a superconducting quantum processor. *IEEE Micro*, 38(3):40–47, 2018.

Joydip Ghosh. A note on the measures of process fidelity for non-unitary quantum operations. *arXiv e-prints*, 2011.

Joydip Ghosh, Austin G Fowler, and Michael R Geller. Surface code with decoherence: An analysis of three superconducting architectures. *Physical Review A*, 86(6), 2012.

Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T Chong. Optimized quantum compilation for near-term algorithms with openpulse. *arXiv preprint arXiv:2004.11205*, 2020.

Daniel Gottesman. Stabilizer codes and quantum error correction. *arXiv preprint quant-ph/9705052*, 1997.

Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996.

Hartmut Häffner, Wolfgang Hänsel, CF Roos, Jan Benhelm, Michael Chwalla, Timo Körber, UD Rapol, Mark Riebe, PO Schmidt, Christoph Becher, et al. Scalable multiparticle entanglement of trapped ions. *Nature*, 438(7068):643, 2005.

Matthew B Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. Improving quantum algorithms for quantum chemistry. *arXiv preprint arXiv:1403.1539*, 2014.

Jeff Heckey, Shruti Patil, Ali JavadiAbhari, Adam Holmes, Daniel Kudrow, Kenneth R Brown, Diana Franklin, Frederic T Chong, and Margaret Martonosi. Compiler management of communication and parallelism for quantum computation. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 445–456, 2015.

Quentin P Herr, Anna Y Herr, Oliver T Oberg, and Alexander G Ioannidis. Ultra-low-power superconductor logic. *Journal of applied physics*, 109(10), 2011.

Jared B Hertzberg, Eric J Zhang, Sami Rosenblatt, Easwar Magesan, John A Smolin, Jeng-Bang Yau, Vivek P Adiga, Martin Sandberg, Markus Brink, Jerry M Chow, and Jason S. Orcutt. Laser-annealing josephson junctions for yielding scaled-up superconducting quantum processors. *arXiv preprint arXiv:2009.00781*, 2020.

Adam Holmes, Sonika Johri, Gian Giacomo Guerreschi, James S Clarke, and AY Matsuura. Impact of qubit connectivity on quantum algorithm performance. *arXiv preprint arXiv:1811.02125*, 2018.

Adam Holmes, Mohammad Reza Jokar, Ghasem Pasandi, Yongshan Ding, Massoud Pedram, and Frederic T Chong. Nisq+: Boosting quantum computing power by approximating quantum error correction. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 556–569. IEEE, 2020.

J. M. Hornibrook, J. I. Colless, I. D. Conway Lamb, S. J. Pauka, H. Lu, A. C. Gossard, J. D. Watson, G. C. Gardner, S. Fallahi, M. J. Manfra, and D. J. Reilly. Cryogenic control architecture for large-scale quantum computing. *Phys. Rev. Applied*, 3, 2015.

M. D. Hutchings, J. B. Hertzberg, Y. Liu, N. T. Bronn, G. A. Keefe, Markus Brink, Jerry M. Chow, and B. L. T. Plourde. Tunable superconducting qubits with flux-independent coherence. *Phys. Rev. Applied*, 8, 2017.

Koki Ishida, Ilkwon Byun, Ikki Nagaoka, Kosuke Fukumitsu, Masamitsu Tanaka, Satoshi Kawakami, Teruo Tanimoto, Takatsugu Ono, Jangwoo Kim, and Koji Inoue. Supernpu: An extremely fast neural processing unit using superconducting logic devices. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 58–72. IEEE, 2020.

Ali Javadi-Abhari. *Towards a Scalable Software Stack for Resource Estimation and Optimization in General-Purpose Quantum Computers*. PhD thesis, Princeton University, 2017.

Mohammad Reza Jokar, Lunkai Zhang, Yanjing Li, and Frederic T Chong. Investigating energy-efficient technologies for next-generation optical interconnection networks. In *TECHCON*, 2017.

Mohammad Reza Jokar, Junyi Qiu, Lynford L Goddard, John M Dallesasse, Milton Feng, Yanjing Li, and Frederic T Chong. A high-performance and energy-efficient optical network using transistor laser. In *TECHCON*, 2019a.

Mohammad Reza Jokar, Lunkai Zhang, John M Dallesasse, Frederic T Chong, and Yanjing Li. Direct-modulated optical networks for interposer systems. In *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, pages 1–8, 2019b.

Mohammad Reza Jokar, Junyi Qiu, Frederic T Chong, Lynford L Goddard, John M Dallesasse, Milton Feng, and Yanjing Li. Baldur: A power-efficient and scalable network using all-optical switches. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 153–166. IEEE, 2020.

Mohammad Reza Jokar, Richard Rines, and Frederic T Chong. Practical implications of sfq-based two-qubit gates. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2021.

Mohammad Reza Jokar, Richard Rines, Ghasem Pasandi, Haolin Cong, Adam Holmes, Yunong Shi, Massoud Pedram, and Frederic T Chong. Digiq: A scalable digital controller for quantum computers using sfq logic. In *2022 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2022.

VK Kaplunenko, MI Khabipov, VP Koshelets, KK Likharev, OA Mukhanov, VK Semenov, IL Serpuchenko, and AN Vystavkin. Experimental study of the rsfq logic elements. *IEEE Transactions on Magnetics*, 25(2):861–864, 1989.

J. Kelly, Rami Barends, Austin G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, Charles Neill, P. J. J. O'Malley, C. Quintana, Pedran Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and John M. Martinis. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, 2015.

Julian Kelly. A preview of bristlecone, google's new quantum processor. *Google Research Blog*, 5, 2018.

DE Kirichenko, Saad Sarwana, and AF Kirichenko. Zero static power dissipation biasing of rsfq circuits. *IEEE Transactions on Applied Superconductivity*, 21(3):776–779, 2011.

A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. *Phys. Rev. A*, 76, 2007.

Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2), 2019.

Charles E Leiserson and James B Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1): 5–35, 1991.

Bjoern Lekitsch, Sebastian Weidt, Austin G Fowler, Klaus Mølmer, Simon J Devitt, Christof Wunderlich, and Winfried K Hensinger. Blueprint for a microwave trapped ion quantum computer. *Science Advances*, 3(2), 2017.

E. Leonard, M. A. Beck, J. Nelson, B.G. Christensen, T. Thorbeck, C. Howington, A. Opremcak, I.V. Pechenezhskiy, K. Dodge, N.P. Dupuis, M.D. Hutchings, J. Ku, F. Schlenker, J. Suttle, C. Wilen, S. Zhu, M.G. Vavilov, B.L.T. Plourde, and R. McDermott. Digital coherent control of a superconducting qubit. *Phys. Rev. Applied*, 11, 2019.

Nelson Leung, Mohamed Abdelhafez, Jens Koch, and David Schuster. Speedup for quantum optimal control from automatic differentiation based on graphics processing units. *Physical Review A*, 95(4), 2017.

James E Levy, Anand Ganti, Cynthia A Phillips, Benjamin R Hamlet, Andrew J Landahl, Thomas M Gurrieri, Robert D Carr, and Malcolm S Carroll. The impact of classical electronics constraints on a solid-state logical qubit memory. *arXiv preprint arXiv:0904.0003*, 2009.

James E Levy, Malcolm S Carroll, Anand Ganti, Cynthia A Phillips, Andrew J Landahl, Thomas M Gurrieri, Robert D Carr, Harold L Stalford, and Erik Nielsen. Implications of electronics constraints for solid-state quantum error correction and quantum circuit failure probability. *New Journal of Physics*, 13(8), 2011.

Gushu Li, Yufei Ding, and Yuan Xie. Towards efficient superconducting quantum processor architecture design. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1031–1045, 2020.

Kangbo Li, R McDermott, and Maxim G Vavilov. Scalable hardware-efficient qubit control with single flux quantum pulse sequences. *arXiv preprint arXiv:1902.02911*, 2019.

Daniel A Lidar and Todd A Brun. *Quantum error correction*. Cambridge university press, 2013.

Per J Liebermann and Frank K Wilhelm. Optimal qubit control using single-flux quantum pulses. *Physical Review Applied*, 6(2), 2016.

Konstantin K Likharev and Vasilii K Semenov. Rsfq logic/memory family: A new josephson-junction technology for sub-terahertz-clock-frequency digital systems. *IEEE Transactions on Applied Superconductivity*, 1(1):3–28, 1991.

Norbert M Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 2017.

Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Phys. Rev. Lett.*, 121, 2018.

Vladimir E. Manucharyan, Jens Koch, Leonid I. Glazman, and Michel H. Devoret. Fluxonium: Single cooper-pair circuit free of charge offsets. *Science*, 326(5949):113–116, 2009.

Dmitri Maslov. Basic circuit compilation techniques for an ion-trap quantum machine. *New Journal of Physics*, 19(2), 2017.

R. McDermott and M. G. Vavilov. Accurate qubit control with single flux quantum pulses. *Phys. Rev. Applied*, 2, 2014.

R McDermott, MG Vavilov, BLT Plourde, FK Wilhelm, PJ Liebermann, OA Mukhanov, and TA Ohki. Quantum–classical interface based on single flux quantum digital logic. *Quantum science and technology*, 3(2), 2018.

David C McKay, Christopher J Wood, Sarah Sheldon, Jerry M Chow, and Jay M Gambetta. Efficient z gates for quantum computing. *Physical Review A*, 96(2), 2017.

F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. Simple pulses for elimination of leakage in weakly nonlinear qubits. *Phys. Rev. Lett.*, 103, 2009.

Oleg A Mukhanov. Energy-efficient single flux quantum technology. *IEEE Transactions on Applied Superconductivity*, 21(3):760–769, 2011.

Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1015–1029, 2019a.

Prakash Murali, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi. Formal constraint-based compilation for noisy intermediate-scale quantum systems. *Microprocessors and Microsystems*, 66:102–112, 2019b.

Prakash Murali, David C. McKay, Margaret Martonosi, and Ali Javadi-Abhari. Software Mitigation of Crosstalk on Noisy Intermediate-Scale Quantum Computers. *arXiv e-prints*, 2020.

Michael A Nielsen. A simple formula for the average gate fidelity of a quantum dynamical operation. *Physics Letters A*, 303(4):249–252, 2002.

Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

G. S. Paraoanu. Microwave-induced coupling of superconducting qubits. *Phys. Rev. B*, 74, 2006.

Ghasem Pasandi and Massoud Pedram. Balanced factorization and rewriting algorithms for synthesizing single flux quantum logic circuits. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI)*, pages 183–188, 2019a.

Ghasem Pasandi and Massoud Pedram. A dynamic programming-based path balancing technology mapping algorithm targeting area minimization. In *Proc. IEEE/ACM Int. Conf. Comput. Aided Des. (ICCAD)*, 2019b.

Ghasem Pasandi and Massoud Pedram. An efficient pipelined architecture for superconducting single flux quantum logic circuits utilizing dual clocks. *IEEE Transactions on Applied Superconductivity*, 30(2):1–12, 2019c.

Ghasem Pasandi and Massoud Pedram. PBMap: A path balancing technology mapping algorithm for single flux quantum logic circuits. *IEEE Transactions on Applied Superconductivity*, 29(4):1–14, 2019d.

Ghasem Pasandi, Alireza Shafaei, and Massoud Pedram. SFQmap: A technology mapping tool for single flux quantum logic circuits. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

Bishnu Patra, Rosario M Incandela, Jeroen PG Van Dijk, Harald AR Homulle, Lin Song, Mina Shahmohammadi, Robert Bogdan Staszewski, Andrei Vladimirescu, Masoud Babaie, Fabio Sebastiano, et al. Cryo-cmos circuits and systems for quantum computing applications. *IEEE Journal of Solid-State Circuits*, 53(1):309–321, 2018.

JH Plantenberg, PC De Groot, CJPM Harmans, and JE Mooij. Demonstration of controlled-not quantum gates on a pair of superconducting quantum bits. *Nature*, 447(7146):836, 2007.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

Chad Rigetti and Michel Devoret. Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies. *Phys. Rev. B*, 81, 2010.

J. A. Schreier, A. A. Houck, Jens Koch, D. I. Schuster, B. R. Johnson, J. M. Chow, J. M. Gambetta, J. Majer, L. Frunzio, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Suppressing charge noise decoherence in superconducting charge qubits. *Phys. Rev. B*, 77, 2008.

Fabio Sebastiano, Harald Homulle, Bishnu Patra, Rosario Incandela, Jeroen van Dijk, Lin Song, Masoud Babaie, Andrei Vladimirescu, and Edoardo Charbon. Cryo-cmos electronic control for scalable quantum computing. In *Proceedings of the 54th Annual Design Automation Conference 2017*, page 13. ACM, 2017.

V.K. Semenov and D.V. Averin. Sfq control circuits for josephson junction qubits. *IEEE Transactions on Applied Superconductivity*, 13(2):960–965, 2003.

Soheil Nazar Shahsavani, Ting-Ru Lin, Alireza Shafaei, Coenrad J Fourie, and Massoud Pedram. An integrated row-based cell placement and interconnect synthesis tool for large sfq logic circuits. *IEEE Transactions on Applied Superconductivity*, 27(4):1–8, 2017.

Sarah Sheldon, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Phys. Rev. A*, 93, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *nature*, 550(7676):354–359, 2017.

Matthias Steffen, David P DiVincenzo, Jerry M Chow, Thomas N Theis, and Mark B Ketchen. Quantum computing: An ibm perspective. *IBM Journal of Research and Development*, 55(5), 2011.

Krysta M Svore and Matthias Troyer. The quantum future of computation. *Computer*, 49 (9):21–30, 2016.

Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. Quantum addition circuits and unbounded fan-out. *arXiv preprint arXiv:0910.2530*, 2009.

Naoki Takeuchi, Dan Ozawa, Yuki Yamanashi, and Nobuyuki Yoshikawa. An adiabatic quantum flux parametron as an ultra-low-power logic device. *Superconductor Science and Technology*, 26(3), 2013.

Swamit S Tannu and Moinuddin K Qureshi. Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 987–999, 2019.

Swamit S Tannu, Douglas M Carmean, and Moinuddin K Qureshi. Cryogenic-dram based memory system for scalable quantum computers: a feasibility study. In *Proceedings of the International Symposium on Memory Systems*, pages 189–195. ACM, 2017a.

Swamit S Tannu, Zachary A Myers, Prashant J Nair, Douglas M Carmean, and Moinuddin K Qureshi. Taming the instruction bandwidth of quantum computers via hardware-managed error correction. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 679–691, 2017b.

Barbara M Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87(2):307, 2015.

Sergey K Tolpygo, Vladimir Bolkhovsky, Terence J Weir, Alex Wynn, Daniel E Oates, Leonard M Johnson, and Mark A Gouker. Advanced fabrication processes for superconducting very large-scale integrated circuits. *IEEE Transactions on Applied Superconductivity*, 26(3):1–10, 2016.

Yu Tomita and Krysta M Svore. Low-distance surface codes under realistic quantum noise. *Physical Review A*, 90(6), 2014.

Giacomo Torlai and Roger G Melko. Neural decoder for topological codes. *Physical review letters*, 119(3), 2017.

UC Berkeley Architecture Research. The sodor processor collection. *https://github.com/ucb-bar/riscv-sodor*.

Jeroen Petrus Gerardus Van Dijk, Bishnu Patra, Sushil Subramanian, Xiao Xue, Nodar Samkharadze, Andrea Corna, Charles Jeon, Farhana Sheikh, Esdras Juarez-Hernandez, Brando Perez Esparza, et al. A scalable cryo-cmos controller for the wideband frequency-multiplexed control of spin qubits and transmons. *IEEE Journal of Solid-State Circuits*, 55(11):2930–2946, 2020.

Savvas Varsamopoulos, Ben Criger, and Koen Bertels. Decoding small surface codes with feedforward neural networks. *Quantum Science and Technology*, 3(1), 2017.

Savvas Varsamopoulos, Koen Bertels, and Carmen G Almudever. Designing neural network based decoders for surface codes. *arXiv preprint arXiv:1811.12456*, 2018.

Savvas Varsamopoulos, Koen Bertels, and Carmen G Almudever. Decoding surface code with a distributed neural network based decoder. *arXiv preprint arXiv:1901.10847*, 2019a.

Savvas Varsamopoulos, Koen Bertels, and Carmen Garcia Almudever. Comparing neural network based decoders for the surface code. *IEEE Transactions on Computers*, 2019b.

Mark H Volkmann, Anubhav Sahu, Coenrad J Fourie, and Oleg A Mukhanov. Experimental investigation of energy-efficient digital circuits based on esfq logic. *IEEE Transactions on Applied Superconductivity*, 23(3), 2013.

Fred Ware, Liji Gopalakrishnan, Eric Linstadt, Sally A McKee, Thomas Vogelsang, Kenneth L Wright, Craig Hampel, and Gary Bronner. Do superconducting processors really need cryogenic memories?: the case for cold dram. In *Proceedings of the International Symposium on Memory Systems*, pages 183–188. ACM, 2017.

Whiteley Research Inc. Wrspice circuit simulator. *http://www.wrcad.com/wrspice.html*.

James Wootton. A simple decoder for topological codes. *Entropy*, 17(4):1946–1957, 2015.

Chui-Ping Yang, Shih-I Chu, and Siyuan Han. Possible realization of entanglement, logical gates, and quantum-information transfer with superconducting-quantum-interference-device qubits in cavity qed. *Physical Review A*, 67(4), 2003.

Mingdai Yang, Mohammad Reza Jokar, Junyi Qiu, Qiuwen Lou, Yuming Liu, Aditi Udupa, Frederic T Chong, John M Dallesasse, Milton Feng, Lynford L Goddard, et al. A hybrid optical-electrical analog deep learning accelerator using incoherent optical signals. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pages 271–276, 2021.

DM Zajac, TM Hazard, Xiao Mi, E Nielsen, and JR Petta. Scalable gate architecture for a one-dimensional array of semiconductor spin qubits. *Physical Review Applied*, 6(5), 2016.