Quantum Feedback Control of Multiple Superconducting Qubits

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Yehan Liu

Dissertation Director: Michel Devoret

December , 2016

Acknowledgements

This thesis work would not have been possible without the help of many who have contributed to it during my time at Yale. First, I would like to thank my advisor Michel Devoret for giving me the opportunity to work in his group and for all of his guidance and insightful discussions over the years. I would further like to thank my committee members Dan Prober and Rob Schoelkopf for their invaluable comments and suggestions over the years, and in particular for taking the time to read through my thesis.

I am grateful to have had the opportunity to work with and learn from so many experienced researchers and post-docs: Nissim Ofek, who is always inspiring with his vast knowledge and tireless pursuit of new ideas, Shyam Shankar, who has taught me almost everything I know about experiments with his insatiable patience, and Michael Hatridge, who has always been more than willing to give me the most honest feedback.

The FPGA project also would not have succeeded without the valuable help of Ge Yang, Nick Masluk, and Kurtis Geerlings during the early stage of the development.

Finally, I would like to thank the many current and former members and visitors of the 4th floor of Becton, for all of their support, inspiration, conversation.

Contents

List of Figures

List of Abbreviations

ADC	Analog-to-digital converter
CLB	Configurable logic block
DAC	Digital-to-analog converter
DD	Driven-dissipative
DSP	Digital signal processing
FIFO	First in first out
FPGA	Field programmable gate array
FSM	Finite state machine
GB	Gigabytes
GSPS	Giga samples per second
JPC	Josephson Parametric Converter
HDL	Hardware descriptive language
IC	Integrated circuit
I/O	Input/output
LUT	Look up table
LVDS	Low voltage differential signaling
MB	Measurement-based
MSPS	Mega samples per second
PCB	Printed circuit board
QIP	Quantum information processing
QND	Quantum non-demolition

ROM Read only memory

Glossary

Actuator	The component in a feedback loop that applies an action to the plant.
block RAM	A dedicated memory element in an FPGA. Each block RAM in a Virtex 6 FPGA has 32 Kbit.
Configuring	The final step of programming an FPGA. A development software converts the fully placed and routed circuit layout into a binary file (also known as a configuration file) that is then downloaded to the FPGA and configures its logical blocks according to the design. Colloquially known as "burning the logic".
Driven-dissipative feedback	A feedback control loop in which the plant is coupled to another quantum system with specifically engineered dissipation, e.g., a quantum reservoir or bath. The reservoir acts as an effective "quantum controller". Also known as reservoir or bath engineering.

FSM	A finite state machine (FSM) models a sequential system that transits between a finite set of internal states.
Instruction memory	A look-up table that stores a list of instructions to be read by a component. The instructions in an instruction memory can either be linked as in the case of a sequencer or unlinked as in the case of the state estimator.
`internal function`	The boolean function for calculating **`internal result`**.
`internal result`	A boolean state monitored by the all sequencers in the Yngwie logic. All the sequences' branching is conditioned by its value. It is calculated by **`internal function`** which takes the states of a selection of eight 1-bit registers as inputs.
IP core	Intellectual Property (IP) cores are pre-synthesized logic blocks that provide a variety of functionalities. They are provided by Xilinx and third party FPGA development board vendors to customers so that they do not have to implement the complex logic circuits in HDL code themselves.

Master sequencer	Thee master sequencer is responsible for generating the digital sequence that determines sampling windows. However, the master sequencer does much more than just outputting a digital sequence and is, as its name suggests, the master of all the sequencers. It is in charge of deciding the global state (`internal result`) that dictates how all the sequencers transition between instructions.
Measurement-based feedback	A feedback control loop in which a controller's response is based on active measurement of the plant.
Mealy machine	A type of finite state machine in which the machine's outputs depend on both the current state and the current input.
Moore machine	A type of finite state machine in which the machine's outputs depend only on the current state. Moore machine takes more states than Mealy machine for the same task (but is more easily represented visually and understood).
Plant	The target system which a feedback loop intends to control or stabilize.

Register	A digital circuit element that takes an input signal and a clock signal and has an output. The input is sampled and the output is updated on the rising edge of the clock exclusively. The output maintains its state at all other times.
sequencer	A component that consists of at least an FSM-based control unit that orchestrates the reading of the instruction and an instruction memory storing a sequence of instructions. There are three kinds of sequencers: analog, digital and master.
Yngwie	The name of the custom logic developed on the X6-1000M board to implement the all-in-one FPGA controller

CHAPTER 1

Introduction

1.1 Thesis overview

In this thesis I will present the results of my effort to implement quantum feedback control of superconducting qubits. The first goal of the thesis work was to build a control system capable of running feedback experiments. There has been tremendous progress in both coherence and high fidelity single shot readout of superconducting qubits. Latency in measurement can now be reduced to just a small percentage of a qubit coherence time. All these improvements in system parameters paved the way for measurement-based feedback control of quantum systems and error correction protocols. But feedback control experiments were still challenging to perform because we were still missing a capable controller to complete the feedback loop. The operations of the controller need to be fast and deterministic in terms of timing and flexible. To meet those demands, we implemented an all-in-one system that contains a digitizer, a demodulator, a state estimator and an AWG on a commercial field-programmable-gate-array (FPGA) board. The FPGA system shows superior performance in terms of throughput, timing stability and on-the-fly programmability compared to traditional technology.

The FPGA controller development has been a continuing effort. As the con-

troller's capabilities became more sophisticated, the second goal of the thesis came into action: to use the control system to implement feedback experiments. As a proof of principle, we first successfully demonstrated reset of a single qubit in high entropy state to the ground state of very high purity, using an active feedback protocol implemented on the FPGA system. We then tested our feedback platform on a system consisting of two superconducting qubits coupled to a cavity. Using the same experimental setup, we stabilized entanglement of the two qubits by two nominally distinct schemes: a "passive" reservoir engineering method and an "active" correction based on conditional parity measurements. Furthermore, the flexibility of our feedback controller enabled us to implement a "nested" feedback protocol that combined both schemes to get the best of both worlds.

This thesis is organized as following:

In Chapter 1, we introduce measurement-based feedback and driven-dissipative feedback. We shed light on their connection to entropy evacuation, which lead us to study the paradox of Maxwell's demon and a desire to create a more "powerful" demon. This chapter is concluded with a highlight of key experiment results.

In Chapter 2, we start with a primer on current FPGA technology and digital design principles which are important to implementing a working control system. We then describe the architecture of the FPGA-based AWG. The chapter also provides a chronicle of the early stages of FPGA development that paved the way for the FPGA-based full controller later.

In Chapter 3, we give a detailed look of the architecture and the principles of operations of the FPGA-based all-in-one controller.

In Chapter 4, we demonstrate the proof of principle experiment to purify the ground state of a single qubit.

And lastly in Chapter 5, we present the experiment to compare and combine measurement-based feedback and driven-dissipative entanglement stabilization.

1.2 Two paradigms of feedback

There are two main kinds of feedback: measurement-based feedback and autonomous feedback.

1.2.1 Measurement-based feedback

Measurement-based (MB) feedback is more intuitively well understood for its ubiquity in our everyday life. There are many instances in which human beings want to control a system which has some its degrees of freedom coupled to an external "environment" that causes unwanted fluctuations. In many such instances, we find MB feedback come to rescue. The thermostat in a house uses MB feedback to control the temperature inside the house to our liking whether it is a scorching hot summer or a frigid winter outside. An airplane also uses MB feedback to maintain its altitude reasonably well, despite constant turbulences in the air. The central bank of a country uses MB feedback to try to keep prices and the economy stable for its people, against many unruly macroeconomic forces.

The feedback loop can be broken down in four main components (Fig. 1.1). The "system"[1] is the system of interest that we want to control. The "sensor" is the measurement device that captures an output signal from the system, which should reflect the state of the system. The sensor transmits the detected signal to the "estimator" which does some calculation to estimate the state of the system based on the captured signal. A common goal of feedback is to stabilize the system towards a predetermined target state or value. The state estimator and an "actuator", the final components in the loop, constitute the controller for the feedback loop. The controller compares the measured state or value to the desired target value. Depending on the difference, the controller applies an action on the

[1] also commonly referred to as "plant" in control theory literature, which will be used interchangeably here

target through an "actuator" to steer it.

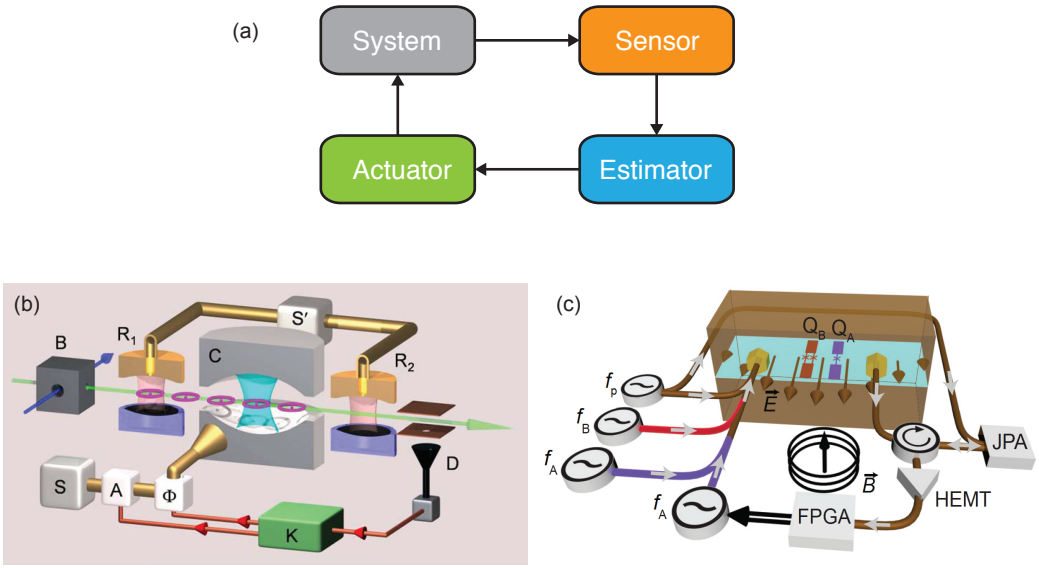

Figure 1.1: Overview of measurement-based feedback and examples. (a) The canonical flow of MB feedback loop. (b) Continuous MB feedback to stabilize a Fock state in a microwave cavity using Rydberg atoms, the first experimental demonstration real time quantum MB feedback[Sayrin et al., 2011]. (c) Deterministic entanglement generation of superconducting qubits by MB feedback[Riste et al., 2013].

Owing to the existing foundation in classical control and feedback in engineering, measurement-based feedback has been readily adopted in quantum control[Doherty et al., 2000; Wiseman and Milburn, 2009].

Two unique challenges exist for quantum feedback. Quantum systems are much more fragile than classical systems in terms of susceptibility to environmental noise. Without error correction or stabilization, a quantum state tends to have very limited lifetime. Therefore the controller in the feedback loop needs to process the measurement result and react within the relevant coherence time. This calls for very low-latency classical control electronics.

The second challenge is that measurement in quantum feedback is not a passive process. The act of measurement can change the state of the system be-

ing measured, a perturbation known as measurement back-action. The goal is to achieve quantum non-demolition (QND) measurement in which the quantum state is preserved after even repeated measurements. One strategy is have an ancilla system such that measuring the ancilla system gives us information about the primary system but does not disturb it. Another strategy, in the event that we know the target state we want to stabilize, is to devise the measurement in a way such as that the target state is effectively an eigenstate of the measurement process.

Continuous measurement-based quantum feedback was first demonstrated with Rydberg atomic qubits[Sayrin et al., 2011]. Sayrin et al stabilized on demand Fock states in a superconducting cavity by a continuous feedback loop. They used Rydberg atoms to measure the photon number in the cavity and fed the measurement result to a controller that adjusted in real time the microwave field injected into the cavity.

In recent years, superconducting qubits have been a flourishing ground for MB feedback. Several groups have applied MB feedback for experiments ranging from singe-qubit state purification, stabilization, deterministic teleportation to entanglement generation[Ristè et al., 2012a; Vijay et al., 2012; Campagne-Ibarcq et al., 2013; Riste et al., 2013; Steffen et al., 2013; Ristè et al., 2014; Córcoles et al., 2015]. Superconducting qubits have joined quantum optics as leaders in quantum feedback thanks to signifiant improvements of both the system and controller parts of the feedback loop. Common to all these MB experiments in superconducting qubits are relatively long lived qubits and parametric amplifiers for high fidelity real time readout.

1.2.2 Autonomous feedback

Autonomous feedback, despite not being as intuitive as measurement-based feedback, has been around since the industrial revolution. Autonomous feedback got its name due to the fact that there is no external controller that does active measurement and processing to reach a decision. The famous centrifugal Watt governor was based on feedback that that involves no measurement (Fig. 1.2).

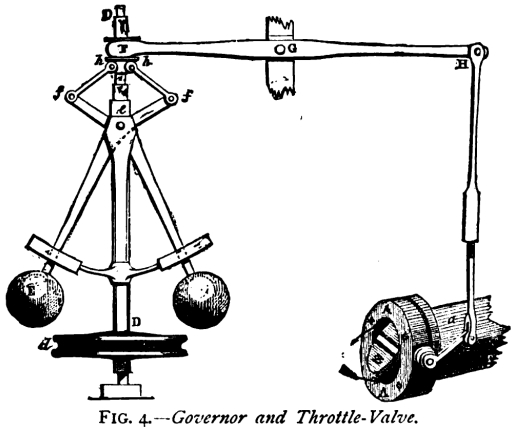

Figure 1.2: Stabilization of steam engine by autonomous feedback. The Watt governor was used to stabilize the speed of an steam engine. The steam engine is connected to the governor which functions as a regulator of the "fuel" to the engine, i.e., controlling the amount of steam flowing to the engine. The speed of the engine is positively correlated to the amount of steam input. The governor consists of a rotating main shaft with two arms, each of which has a metal ball attached at the end. When steam engine spins too fast, the shaft of the governor also rotates faster, causing the arms to rise which presses down a valve, reducing steam flow. The opposite happens and steam flow is increased when the steam engine spins too slow. The stability of feedback is actually ensured by dissipation[Maxwell, 1867]. The friction acting on the main shaft, which prevents it from moving up or down too fast, inhibits big run-away speed oscillations. Not surprisingly, dissipation plays a crucial role in autonomous feedback in general as we shall see in the rest of this thesis.

In MB feedback, the controller and the plant (see the definition in the Glossary) are typically physically separate entities; *there is no Hamiltonian that describes the entire operation of the joint system*. As illustrated by the example of the Watt governor, autonomous feedback is feedback by physical coupling, whether me-

chanical or electrical. The plant is connected with another system, the controller, which shares some of its degrees of freedom with the plant, such that *the joint system can be described by an interaction Hamiltonian*. The evolution of the joint system according to the coupled equations of motion autonomously sustains the feedback loop.

In quantum control, two seemingly distinct but closely related descriptions of autonomous feedback arise (Fig. 1.3). One description is called coherent feedback [Wiseman and Milburn, 1994; Lloyd, 2000; Nurdin et al., 2009], which parallels the structure of MB feedback more closely. Output from the plant leaks into another *quantum* system which acts as a controller. The quantum controller processes the output from the plant coherently (without any classical measurement) according to its Hamiltonian and feeds a new filtered/processed signal back to the plant, that controls the dynamics of the plant. This description is popular in the quantum optics community since signal paths from plant to controller and vice versa can be clearly delineated, through directional coupling by either optical fibers or waveguides.

The other description is called reservoir/bath engineering, or driven-dissipative (DD) feedback[Poyatos et al., 1996]. In DD feedback, the plant is coupled to another quantum system with specifically engineered dissipation, e.g., a quantum reservoir or bath. Unlike coherent feedback, DD feedback tends not to have a defined directional coupling between the plant and the controller; instead, the coupling between the two originates from a significant interactive term in the joint system Hamiltonian. Nevertheless, as we shall see with some example DD experiments, we can consider the reservoir as an effective "quantum controller" that, with the help of externally applied drives, removes entropy from the plant and steers it towards the desired state. The dissipation of the reservoir is key since it allows the "controller" to decay to its ground state, i.e., to reset, and thus the

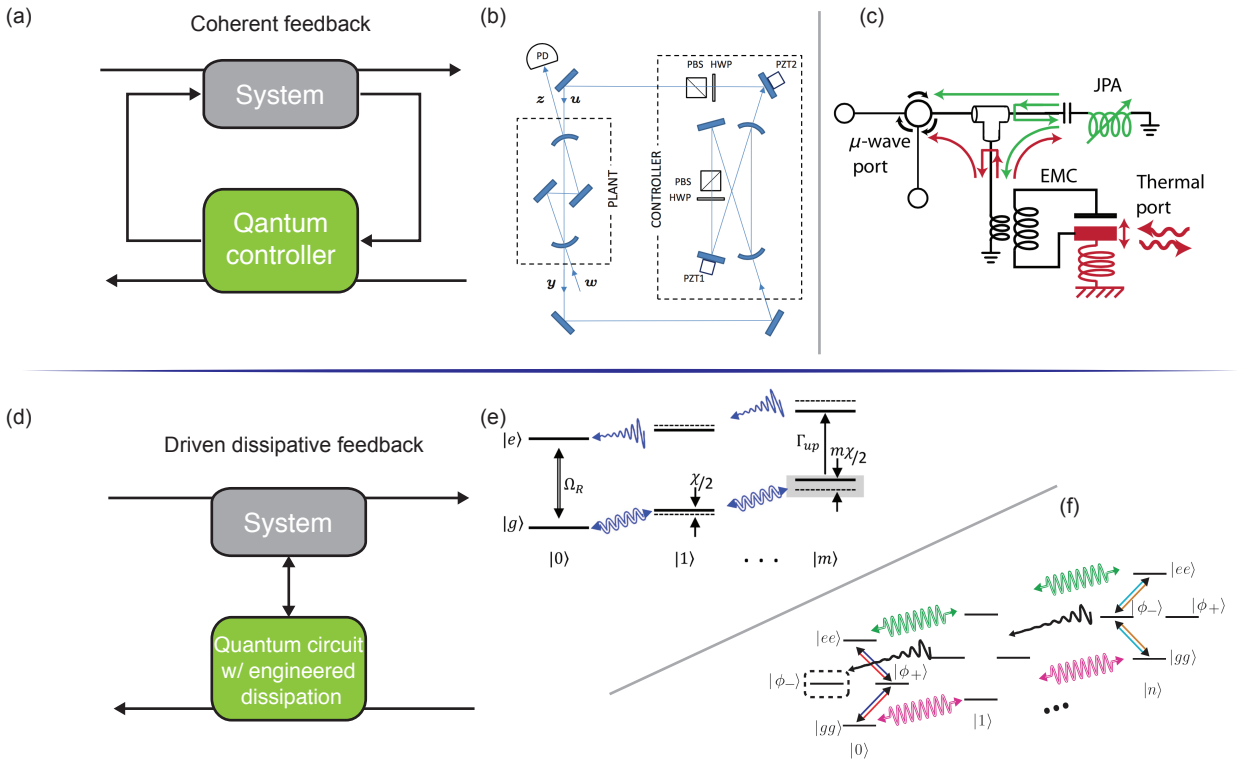

Figure 1.3: Overview of autonomous feedback and examples. (a) Control flow of coherent feedback. Lines through the components represent external drives applied to them, which provide the signal in the feedback loop. (b) An all-optical feedback loop to stabilize laser signal disturbance. The plant and controller of the feedback loop comprise two optical ring resonators; the feedback signals comprise laser beams[Mabuchi, 2008]. (c) A coherent-feedback network implemented with modular superconducting devices. A JPA (Josephson Parametric Amplifier)[Castellanos-Beltran et al., 2008] acts as the controller and a mechanical oscillator as the plant. The feedback loop achieves tunable coupling to the mechanical oscillator.[Kerckhoff et al., 2013] (d) Control of driven dissipative feedback. (e) DD stabilization of a single qubit's ground state by coupling between the qubit and a cavity with an engineered lifetime[Geerlings et al., 2013] (f) DD stabilization of an entangled state of two qubits by their interaction with a cavity with engineered dissipation [Shankar et al., 2013].

feedback loop to repeat indefinitely.

This approach has been demonstrated on a variety of physical systems including atomic ensembles[Krauter et al., 2011], trapped ions[Lin et al., 2013], mechanical resonators[Kienzler et al., 2015] and superconducting qubits[Murch et al., 2012; Geerlings et al., 2013; Leghtas et al., 2013; Shankar et al., 2013; Leghtas et al., 2015; Holland et al., 2015]. In many all of these implementations, the resonators, whether mechanical or superconducting, are designed with specific dissipation and serve as the reservoir.

DD feedback does not require any sophisticated low-latency electronics for fast real time processing. The feedback mechanism is built in the joint system Hamiltonian. As a result, feedback latency for DD is typically less than MB. Since parameters in Hamiltonian are often fixed during hardware design, adjusting the feedback by changing the "quantum controller", however, can be more challenging than re-programming a classical controller.

1.2.3 MB and DD feedback in quantum error correction

Random fluctuations in the environment inevitably cause decoherence of quantum systems that are coupled to it. In quantum computing, the limited coherence times of physical qubits pose a major obstacle to large scale quantum information processing. Protecting quantum information from decoherence-induced errors by active quantum error correction (QEC) thus is a necessary and crucial step towards building a practical quantum computer[Nielsen and Chuang, 2004; Terhal, 2015]. There is significant ongoing effort in the community to find the best strategy for this goal[Kerckhoff et al., 2010; Fowler et al., 2012; Fujii et al., 2014]. So far, many of the proposed or demonstrated strategies fall into *either* the MB *or* the DD camps.

To illustrate how the MB and DD schemes approach QEC respectively, let

us consider the quantum repetition code[Nielsen and Chuang, 2004] that corrects for a bit flip or phase flip error. Chiaverini et al[Chiaverini et al., 2004] first demonstrated the MB approach to the three-qubit repetition code on trapped ions[2]. Schindler et al[Schindler et al., 2011] and Reed et al[Reed et al., 2012] later have also shown an autonomous version of the repetition code, with trapped ions and superconducting qubits, respectively.

The repetition QEC code has three stages. In the first stage, we map the qubit we want to protect onto the logical state of three entangled qubits. In the second stage, after an error may have occurred, we reverse the operations performed in the first stage to decode the logical qubit back to three disentangled qubits. After the second stage, the primary qubit has either erroneously flipped or not. The error syndrome is now represented by the states of the two ancilla qubits. In the final stage, we apply rotations on the qubits conditioned on the error syndromes to undo the error.

The difference between the MB and DD approach is what happens in the final stage. In the MB approach, a classical controller performs projective measurements of the ancilla qubits, the results of which correspond to the eigenvalues of a set of two-qubit parity operators and give the error syndrome. *Conditioned* on the measurements, the controller applies correction gates, if necessary, to the qubits to undo the error and reset the ancilla. In the DD approach, there is no projective measurement or need for a low-latency classical controller for real time processing. Instead, a coherent multi-qubit gate, specifically, a controlled-controlled not gate (a Toffoli gate) is applied. It autonomously flips the primary qubit if and only if the two ancilla qubits are excited. Afterwards, some unconditional dissipation-based scheme is then used to reset the ancilla qubits.

During the QEC protocol, the random errors caused by noise in the environ-

[2]Cory et al were actually the first to demonstrate the quantum repetition code[Cory et al., 1998]. They implemented it in NMR. However, they did not present a method to reset the ancilla qubits.

ment induce uncertainty in the states of the ancilla qubits, corresponding to an increase of their entropy. We will look more closely at the connection between information and entropy in sec. 1.3. For now, we see that both the MB and DD strategies rely on transferring entropy out of the ancilla qubits in the final stage. In both approaches, the removal of entropy in the final stage is accomplished by a non-unitary action. In MB, this is the projective measurements of the ancilla qubits. In DD, this is done by the unconditional reset of the ancillas by a dissipative channel. It is important to note that the action applied on the ancillas must be non-unitary since an unitary operation on a system conserves its entropy (from the second law of thermodynamics, we know that entropy cannot decrease from a reversible process; nor can entropy increase in such case since inverting this reversible process would then cause a violation of the law).

1.3 Entropy and Maxwell's demon

We mentioned the concept of entropy in previous sections and suggested a connection between entropy evacuation and QEC. We shall examine it more closely in this section. In thermal and statistical physics, entropy quantifies the amount of disorder in a system. Consider the classic example, identical particles in a box with a partition: the configuration in which equal number of particles reside in each side of the partition has a higher entropy than the configuration in which all the particles reside in just one side of the partition. This is because there are many more ways to fulfill the former configuration than the latter. If we find the system in the first configuration, we do not know a priori which particles are in the right partition and which are in the left. Put in another way, without measurement, we have less certainty about the first configuration than the second configuration. This example seems to show that higher entropy implies more uncertainty about measurement outcomes.

There is indeed a quantitative definition of entropy for uncertainty in information, called Shannon entropy[Nielsen and Chuang, 2004]. It is defined as the following,

$$S = -\sum_i P_i \log_2 P_i \tag{1.1}$$

where P_i is the probability of being in state i.

For a random variable X which we wish to measure, the Shannon entropy tells us how much uncertainty we have about our answer before we measure the variable. Another equivalent interpretation is: the Shannon entropy tell us how much information we gain after we make the measurement. The unit of Shannon entropy is "bits", as the *log* in the definition is base 2. This definition of information uncertainty is also intimately connected to quantifying the optimal physical resources required (i.e., number of bits) to store information. A pure state that stays pure forever has zero entropy since we always know its state and there is no need to store the information. On the other hand, a quantum system, of which the only thing we are certain is its equal likelihood of being in the ground state and the excited state, requires 1 bit of storage[3]

Ideally, when we prepare a quantum system, say a single qubit or a set of entangled qubits in some initial state, we may wish for it to stay in that state for as long as we want. Unfortunately, errors due to decoherence occur and cause it to leave the target state and jump to other states. The increased probability for the qubits to be in non-target states correspond to an increase in the Shannon entropy, i.e., we are now more uncertain about the system before a measurement. Total entropy cannot be decreased. But as we have seen with the example of quantum

[3]We need to emphasize here that entropy is a measurement of the information *uncertainty*, not information itself. A pure state still requires many bits of information to specify its location on the Bloch sphere whereas a totally mixed state (just the center of Bloch sphere) contains no information.

repetition code, a crucial step in QEC is to transfer entropy from a subsystem about which we care about to somewhere we do not.

1.3.1 Maxwell's demon

In 19th century, James Maxwell conjectured a thought experiment. He famously wrote,

"... if we conceive of a being whose faculties are so sharpened that he can follow every molecule in its course, such a being, whose attributes are as essentially finite as our own, would be able to do what is impossible to us. For we have seen that the molecules in a vessel full of air at uniform temperature are moving with velocities by no means uniform, though the mean velocity of any great number of them, arbitrarily selected, is almost exactly uniform. Now let us suppose that such a vessel is divided into two portions, A and B, by a division in which there is a small hole, and that a being, who can see the individual molecules, opens and closes this hole, so as to allow only the swifter molecules to pass from A to B, and only the slower molecules to pass from B to A. He will thus, without expenditure of work, raise the temperature of B and lower that of A, in contradiction to the second law of thermodynamics."

This "being" became known as Maxwell's demon. The paradox raised by Maxwell's demon perplexed physicists for many decades. Instead of heat and work, we can also examine Maxwell's demon in terms of entropy. By separating the hot molecules from the cold molecules, the demon lowers the disorder of the system, thus the entropy of the system. However, this clearly violates the second law of thermodynamics, which states that entropy of the universe as a whole, or an isolated system, cannot be decreased. If the demon removes the entropy from part of a system, then entropy cannot just disappear and must be accounted for somewhere. The answer to the mystery finally came to light thanks to Rolf Landauer of IBM, who looked into irreversibility and energy consumption of compu-

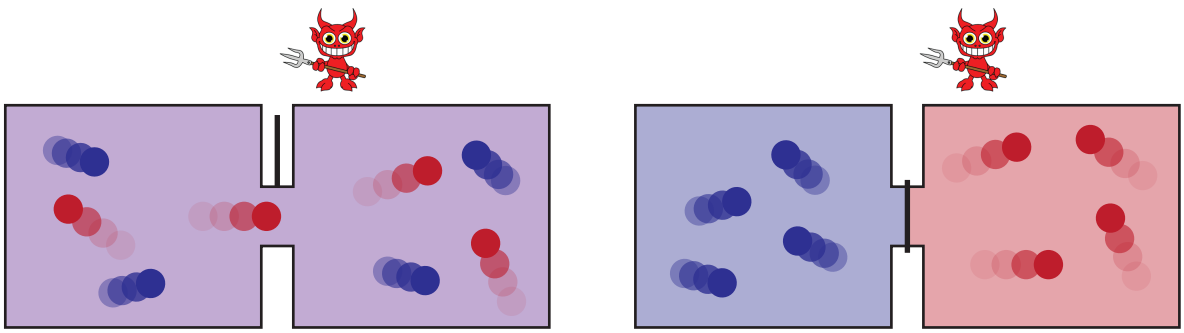

Figure 1.4: The paradox of Maxwell's demon. The demon watches air molecules in the box. It lets fast molecules travel to the right partition and slow molecules the left partition but otherwise prohibits passage between the partitions. In doing so, Maxwell's demon seemingly lowers the entropy of this isolated system and breaks the second law of thermodynamics.

tation[Landauer, 1961; Bennett, 1987].

Maxwell's demon's operations can be viewed as a cyclic process: every action applied on a single molecule is one cycle. We can further assume that almost every step in a cycle, from "seeing", to opening and closing the passage, can be designed to be reversible. For the cyclic process to be repeated however, the demon needs to be initialized in a pre-determined state at the beginning of each cycle regardless of what state it ends up in when the previous cycle concludes. It turns out that it is the resetting that is necessarily irreversible and is accompanied by an increase in entropy. Resetting, as an unconditional replacement of past record with a pre-determined value, can also be understood as erasure of information or memory.

Now we show why erasure of information is an irreversible process by following the model presented in Landauer's work (Fig. 1.5). We shall simplify the discussion by treating the demon's action as binary. There are two possible states for the demon, "opening" vs. "closing" of the partition. It registers its state as a bit on a binary storage device. Resetting or erasing this bit of information is equivalent to setting this bit to 1 regardless of its original value.

Treating the binary storage device as a double potential well, we label a particle

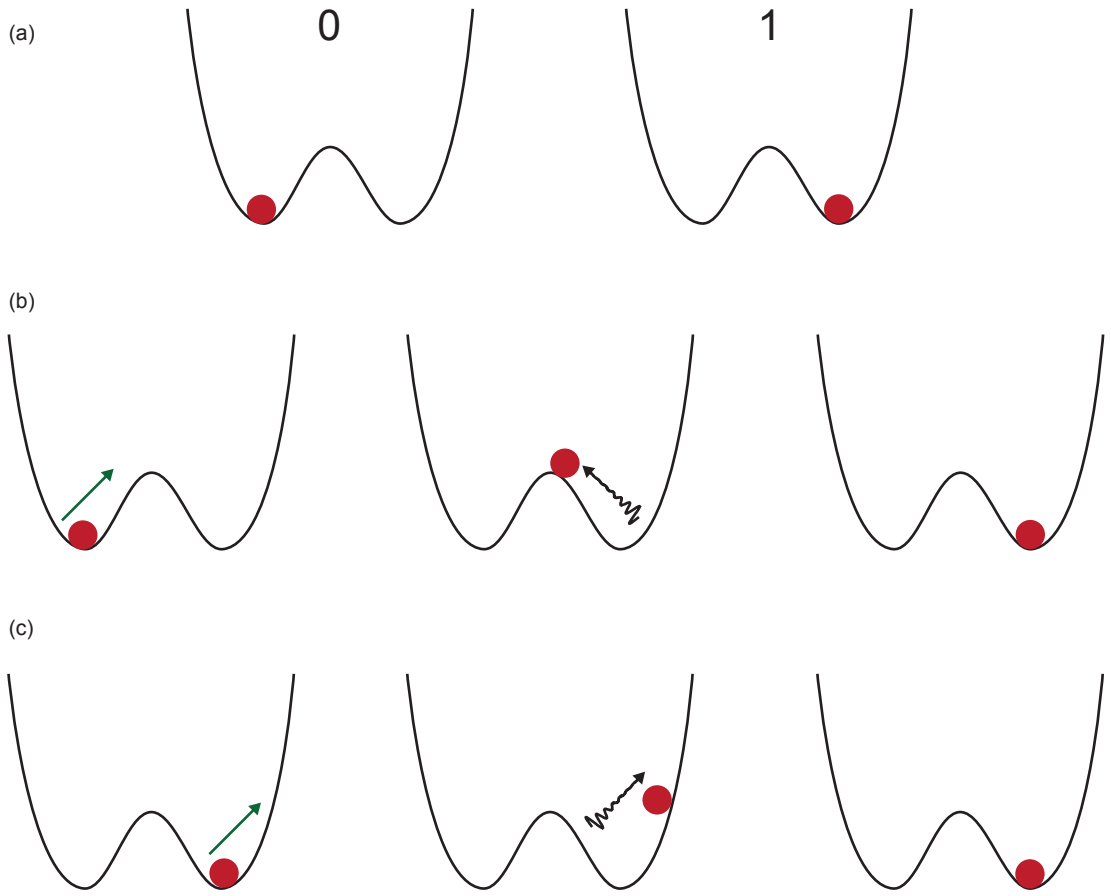

Figure 1.5: Resetting one bit of information. (a) Modeling a binary storage device as a double potential well. (b) Resetting a bit to "1" when it is initially "0". (c) Resetting a bit to "1" when it is initially "1".

sitting in the left (right) well as "0" ("1"). One naive approach to resetting is: if the particle is in the right well, we do nothing. If the particle is in the left well, we first apply a force to push it slightly past the peak of the potential barrier and then apply a retarding force in the opposite direction on the particle's way down so it would stop at the bottom of the right well without oscillating back to the left well. This approach requires two different applications of forces for setting the register value to 1 depending on its initial state. Resetting in computing, however, must be an unconditional operation: the circuitry's mechanism for erasing information must be agnostic to the content of the information.

Here is a different approach: we apply a force on the particle to the right regardless of its initial location. If the particle is in the right well, it will not only stay in this well but go further right. If the particle is in the left well, the force will push it over the barrier to the right well. Furthermore, we add a frictional force in the well to act in the opposite direction of the particle's motion. As a result, once the active force is removed, the particle's motion will be damped and eventually settle at the bottom of the right well. This approach give us a true unconditional reset operation which does not depend on the initial state. By introducing friction, we add a dissipative channel from the storage system to the environment. Dissipation is the key that makes the reset unconditional. Moreover, the frictional force allows the excess kinetic energy of the particle to be dissipated as heat to the environment, which necessarily accompanies an increase of entropy in the environment. Thus we have shown where the entropy evacuated from the system by Maxwell's demon ends up: information erasure is an irreversible process and requires entropy transfer to the environment.

1.3.2 Maxwell's demon and quantum feedback for QEC

The MB strategy for QEC removes entropy from the quantum system by projecting its ancilla system to a particular error syndrome through projective measurements and applying appropriate correction accordingly in the final stage. Since entropy cannot be destroyed, the question that remained from our previous discussion in sec. 1.2.3 is where error-induced entropy from the quantum system is removed to? After investigating the paradox of Maxwell's demon, we have the answer.

In the MB protocol, the classical controller that records the measurement result of one correction cycle needs to be reset for the next cycle. The erasure of the record and the resetting require energy dissipation through resistive elements, such as resistors. This is where the entropy removed from the quantum system ends up.

We can now make a close connection between QEC by quantum feedback and the task performed by Maxwell's demon. In quantum feedback, the controller plays the role of Maxwell's demon, responsible for evacuating entropy from the quantum system. We can view MB and DD feedback schemes as two ways which Maxwell's demon processes input information. MB feedback is like a demon operating by voluntary action: the demon (the classical controller) makes a deliberate/calculated decision based on its observation. DD feedback is like a demon operating by reflex action: the demon (the quantum controller) reacts to "stimuli", i.e., input from the target system without deliberate calculation.

Essentially, in MB feedback, a measurement outcome is recorded in a classical register. The resetting of this classical register completes the entropy transfer from the plant to the environment. In DD feedback, since there is no measurement, information is not stored in a classical device. The quantum controller, however, whether implemented as a set of ancilla qubits or as a reservoir, has its state de-

pendent on the plant through pre-engineered coupling and external drives and hence acts as a register. The "quantum register" also needs to be reset for every new feedback/correction cycle. It resets passively by relaxing back to its initial state, mediated by dissipation. We should note now that despite the different names given to the two schemes, dissipation actually plays an important role in both. In both schemes, resetting the controller enables entropy transfer from the plant to the environment and the resetting depends on dissipation.

A question that arises from making the analogy between quantum feedback and Maxwell's demon is the following: if MB and DD feedback strategies are two choices that Maxwell's demon employs to control a quantum system, can we implement a Maxwell's demon that has both capabilities, voluntary action and reflex action? We can draw inspiration from ourselves, sentient beings who are capable of both types of responses to external stimuli. There are many scenarios when we actually combine both. The simple act of walking is one example. The stabilization of our body in motion by small corrections of the steps is feedback by reflex action; the choice of walking in a particular direction is a result of voluntary action. At least our every day experiences tell us that these two types of feedback can coexist and complement each other.

We therefore came up with the idea of "nested feedback". This new strategy, consisting of multiple continuously operating feedback loops, demonstrates a more powerful version of a Maxwell's demon. It adds a new "voluntary" decision-making capability to a pre-existing Maxwell's demon and in particular, for the DD case in which the original demon acts solely on "reflex action", the enhanced demon has now two complementary decision capabilities. The nested feedback approach can be applicable to many quantum information platforms in which precise Hamiltonian control, efficient quantum measurement and fast classical control are available, such as trapped ions, Rydberg atoms, superconducting qubits,

where simpler forms of Maxwell's demon have been shown.

In the following sections, we highlight the efforts undertaken to implement both the simple and advanced forms of Maxwell's demon.

1.4 The all-in-one feedback controller

The challenges of implementing quantum feedback are manifold. The inherent requirement for low latency calls for fast measurement and processing. Furthermore, if a quantum system incurs errors too often during a feedback cycle, such that new error occurs in the middle of signal processing or pulse generation, any feedback scheme for error correction would fare poorly. Thus a successful feedback protocol also requires long coherence times.

The work in this thesis uses superconducting qubits and the circuit QED (cQED) architecture[Blais et al., 2004] as the quantum information processing platform. Over the last decade or so, the field of superconducting qubits has seen rapid progress. Many of these developments tackled the challenges of quantum control and feedback. The two key achievements have been significant improvement of qubit coherence times and fast high fidelity measurement. Superconducting qubits are now the only solid state implementation that has demonstrated the capability of QND measurement for error correction and control[Devoret and Schoelkopf, 2013].

Coherence times

One of the popular early superconducting qubit design, the Cooper Pair Box (CPB) suffered from high sensitivity to charge noise which shifts the qubit transition frequency and causes dephasing. A new design, the transmon qubit[Koch et al., 2007], benefitted from the addition of a large shunting capacitor that sup-

pressed the qubit energy level's sensitivity to charge fluctuation and as a result reduced dephasing due to charge noise.

Another breakthrough was the introduction of 3-dimensional cavity resonator[Paik et al., 2011]. In the traditional planar resonator design, electromagnetic energy dissipation through lossy materials such as surface dielectric spoils the quality factor of the resonator and limits the qubit's lifetime. The 3D cavity concentrates most of the EM field in vacuum where there is no loss and eliminates participation of the lossy surfaces. The latest generation of 3D cavities has a lifetime on the order of milliseconds and has pushed the qubit relaxation time to hundreds of microseconds[Reagor et al., 2013, 2015]. The simple geometry of the 3D cavity also provides a pristine microwave environment that has minimum spurious modes and is characterized by parameters that are easy to simulate.

Single-shot readout

Despite improvements to coherence times, quantum feedback would still not be possible without good measurement performance. Real time MB feedback requires single-shot readout. To resolve quantum state in a single feedback cycle, the detected quantum signal must dominate classical noise. To simply amplify the weak quantum signal by classical means, however, would not achieve the goal since amplification adds noise. Even the best cryogenic amplifier adds noise that is on the order of 100 times the quantum noise from the superconducting circuit. The invention of superconducting parametric amplifiers, such as the Josephson Parametric Converter (JPC) realized amplification that adds the minimum amount of noise allowed by quantum mechanics and achieves significant gain such that quantum noise accounts for approximately $\sim$90% overall system noise.

Real time control

Progress in the cQED architecture for quantum computing demands competence in both quantum engineering and classical electrical engineering. As exemplified by the developments of superconducting qubits, resonators and paramps, understanding of both quantum mechanics and microwave engineering are essential. Similarly, to close the loop of quantum feedback, especially MB feedback, requires sophisticated classical computer engineering, i.e., the development of a low-latency classical controller.

The two requirements for the controller are fast reaction and deterministic timing. The evolution of a quantum system is measured in nanoseconds. From sampling the signal from a measurement chain to sending a signal back to the quantum system, the controller also needs to respond on a time scale of nanoseconds. And to guarantee coherent control from cycle to cycle, the controller needs to ensure that the timing of each operation stays consistent and any deviation is much less than a nanosecond (on the order of picoseconds). This is not possible with a normal computer in which the execution of processes take indeterminate amount of time and is measured in milliseconds. Microcontrollers, such as Arduinos, have similar problems.

The only viable candidate that can fulfill these requirements is a real time electronic system. The defining characteristic of a real time system is that every operation is synchronized with a digital clock cycle. For example, a real time system's operations can be timed in exact integral number of 4 ns periods if it is synchronized with a 250 MHz digital clock. The most popular choice of real time electronic system is the field programmable gate array (FPGA).

An FPGA is an integrated circuit consisting of millions of digital logic gates, which are organized into many logic blocks. The interconnects between the blocks are configurable to serve different applications. All of the logic blocks have access to an extensive distribution net of clock signals that allow the logic circuitry

to operate synchronously. An FPGA chip is often integrated with other peripheral components on a circuit board, such as a phase-locked loop (PLL), analog-to-digital converters (ADC) or digital-to-analog converters (DAC), which provides an interface for the FPGA to the outside world.

Before this thesis work, the use of an FPGA in superconducting qubit quantum feedback experiments had been relatively simple. The FPGA-based controllers in these experiments (Fig. 1.6) have an analog input interface implemented by ADCs which receive the signals from a measurement apparatus[Ristè et al., 2012a; Campagne-Ibarcq et al., 2013]. The FPGA samples digitized signal, processes the input and conditionally outputs a digital marker whose level depends on the real time processing result. The digital marker controls a separate RF instrument, such as an arbitrary waveform generator (AWG). For example, when the marker level is high, the AWG plays a pre-determined pulse to manipulate a qubit in some way, otherwise it does not play anything. Essentially, the controllers act as a digital switch. The complexity of many quantum feedback experiments goes much beyond the requirement of switching on and off a pulse. Unfortunately, the implementations of the FPGA platforms demonstrated by the various groups are not flexible enough to meet more advanced demands. In particular, solely using a binary trigger to control an external generator as a way to exert the controller's influence severely handicaps the complexity of pulse sequences and consequently of possible manipulations of the quantum system.

Based on experiences of building FPGA-based AWGs in the early stage of this thesis work, we ultimately implemented a controller on a platform that has an FPGA, analog input/ouput and digital input/output. The platform we chose is the x6-1000M board (or also commonly referred to as the "card") from Innovative Integration. It includes two 1GS/s ADCs, two 1GS/s DACs, multiple digital I/O ports, all of which are controlled by a Xilinx Virtex-6 FPGA. We developed custom

Group	Berkeley	Delft	ENS	Yale	
Feedback type	Analog	Digital	Digital	Digital	
Paramp	Phase-sensitive	Phase-sensitive	Phase-preserving	Phase-preserving	
Implementation	Analog PLL	ADwin real time processor + AWGs	FPGA w/ ADCs + AWGs	FPGA w/ ADCs and DACs	
Conditional type	Analog error	Binary switch	Binary switch	Multiple choice	
Latency	~160 ns	~2.8 μs	~500 ns	~1 μs	~250 ns

Figure 1.6: The quantum feedback landscape. Care needs to be taken when comparing the latency among the different implementations. Latency quoted by Delft is total latency defined as the time between when cavity tone arrives to read the cavity and when the qubit pulse reaches the cavity after real time analysis. The first latency figure for Yale is using this definition. Another definition for latency is the time between the last sample to arrive at the FPGA board and the first sample to leave. The second latency figure for Yale and ENS's figure use this definition.

logic on the FPGA to realize the functionality of a digitizer, a digital signal processor and an arbitrary waveform generator on a single unit. Having all the components implemented on a single FPGA board gives flexibility, real time adaptability that is not achievable with previous FPGA setups. The X6-1000M controller can adjust all three components deterministically on a time scale of nanoseconds. This affords us to be generous with the complexity of our feedback experiments, such as required by the nested feedback protocol.

What gives our controller much greater flexibility compared to previous FPGA-based controllers in cQED experiments is that the logic design shares a similar architecture with a general-purpose computer. Modern computers can perform many different tasks without requiring us to reconfigure the circuitry of the CPU. We do not have to completely rebuild a computer just so that we can switch from browsing the internet to using a word processor. Different tasks are specified by different sets of instructions, e.g., "software", that tell the hardware what oper-

ations to execute and what data to operate on. We simply load different sets of instructions into memory to carry out different tasks.

In FPGA programing speak, reconfiguring the interconnects of the logic blocks is also called "loading a logic", or "burning a logic". Every "burning" defines a new hardware configuration. In principle, we can design a simple logic for every different experiment. When we want to run a particular experiment, we can burn the corresponding logic. This is highly inefficient since each "version" of the hardware can then handle only a very limited range of tasks, restricting the complexity of possible experiments. Not to mention that having to compile a new logic for a new experiment every time is not sustainable in terms of time investment. Alternatively, we design the logic in a computer architecture, with its instruction memory, data memory and registers such that the FPGA operates like a real time computer. We write a set of instructions (on a normal computer) to prescribe the operations we want to perform in an experiment, just like computer programmers write softwares for computers. After compiling the instructions to machine code, e.g., many series of 0's and 1's, we stream them into the FPGA's instruction memory, which orchestrate the on-board components to interact with the quantum system in the way we want. Through many iterations of logic development, the X6-1000M controller has now played an important part in many projects at Yale.

1.5 Purifying a single qubit

One of Divincenzo's criteria for building a quantum computer is qubit initialization[DiVincenzo, 2000], also known as qubit resetting. Initialization can mean preparing a qubit in any known initial state but a common choice is the ground state $|g\rangle$. As we have discussed in sec. 1.2.3, resetting a qubit is also necessary for QEC.

Unless the qubit is in perfect contact with a zero temperature bath, there will be population in the excited states. Assuming a Boltzmann distribution, the ratio between the populations in $|e\rangle$ ad $|g\rangle$ is given by the following equation, where f_{ge} is the qubit transition frequency between $|e\rangle$ ad $|g\rangle$.

$$\frac{P_e}{P_g} = \exp\left(-\frac{hf_{ge}}{k_B T}\right) \tag{1.2}$$

We can typically measure the population ratio from experiment and use the above equation to find out T, the effective qubit temperature.

For superconducting qubits which have achieved long coherence times in excess of 100 μs, passive resetting by thermal equilibration with a cold bath takes too long which limits experiment repetition rate. Furthermore, the "cold bath" the qubit is contact with in cQED experiments is often not cold enough, resulting in finite thermal population in the excited state. Thus, active reset is necessary. Active reset can be considered a form of Maxwell's demon in action. Both the DD[Geerlings et al., 2013] and MB schemes[Ristè et al., 2012a] have been demonstrated previously. To show X6-1000M's efficiency in a quantum feedback experiment, we also implemented a MB ground state purification protocol.

We first put the qubit in a maximally entropic state, an equal mixture of $|g\rangle$ and $|e\rangle$ for the purpose of testing the prowess of Maxwell's demon in purifying the qubit to $|g\rangle$. A projective measurement is then applied to the qubit. If the qubit is in $|g\rangle$, we do nothing. Otherwise we apply a π pulse to flip the qubit. Following this conditional step, we apply another measurement to check the result. We dubbed this protocol the "baby Maxwell's demon". It lowers the entropy of the qubit by purifying its ground state probability from 46.2% to 88.7% . Repeating this feedback step one more time can purify the ground state to a higher percentage.

We found that the ground and excited populations did not add up to 100% be-

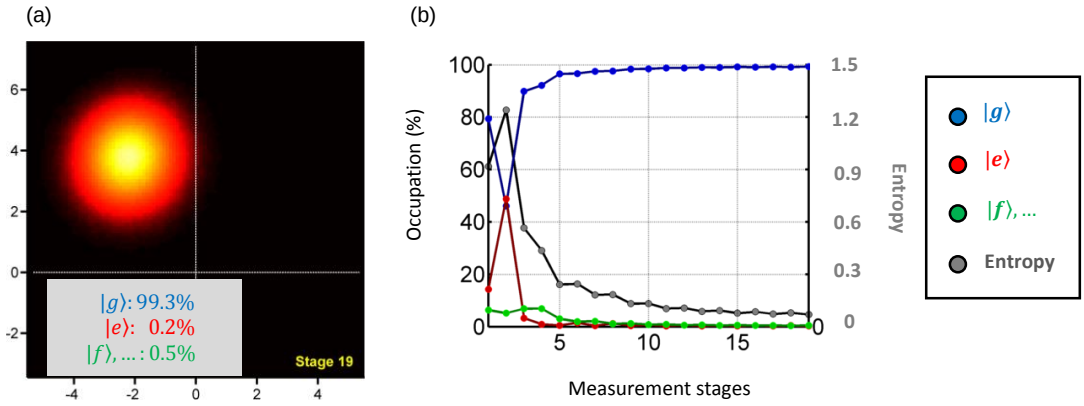

Figure 1.7: Mature Maxwell's demon purifying a single qubit. (a). Histogram at the final stage. Only ground state population visible. (b) Occupation of each state and the entropy of the system as Maxwell's demon progressively purifies the system to ground state. .

cause there was also population remaining in higher states, such as $|f\rangle$ and $|h\rangle$. It appears that repeating this simple feedback stage once does not affect the higher states' population. In principle, we can deplete the higher states by repeating the step many more times; each time we move the population in the first excited state which are then mostly contributed by the natural relaxation from the higher states. Taking advantage of the flexible FPGA controller, we, however, chose to actively deplete the higher excited states by applying pulses addressing the higher transitions. This more advanced protocol was dubbed the "mature Maxwell demon". In one run of the protocol, it progressively purified the ground state to 99.3% and lowered the entropy from a maximum value of 1.2 to 0.007.

1.6 Stabilizing two-qubit entanglement

Entanglement of multiple qubits, such as the Bell state of two qubits, is a valuable resource in many quantum algorithms and communication protocols, such as quantum teleportation. We expanded the single qubit feedback platform by

adding an identical FPGA controller to control two qubits. The task of the platform was to stabilize a Bell state of two transmons.

A scheme to autonomously stabilize a Bell state in superconducting qubits has recently been proposed by Leghtas[Leghtas et al., 2013] and experimentally implemented by Shankar[Shankar et al., 2013] at Yale. This particular task of stabilizing a single state is a proxy for more general QEC experiments where a manifold of states is protected. We realize, for the first time, an MB *stabilization* of a Bell state by repeated active correction through conditional parity measurements[Lalumière et al., 2010; Tornberg and Johansson, 2010; Riste et al., 2013]. We compare this scheme to the DD entanglement stabilization scheme in which the conditional parity switch is autonomous. By performing both schemes on the same hardware setup and cQED system, we shed light on their close connection and compare them on a level playing field.

Previous theoretical works have compared DD and MB for linear quantum control problems[Yamamoto, 2014], such as for minimizing the time required for qubit state purification[Jacobs et al., 2014] or for cooling a quantum oscillator[Hamerly and Mabuchi, 2012]. These comparisons showed DD to be significantly superior. Here we experimentally compare DD and MB on identical hardware and study two performance metrics, the state fidelity and success probability. In our particular setup, we find that distinguishing the superior approach among DD and MB is a more subtle task. The subtlety is two-fold. First, the performance difference depends on which process can be better optimized: the design of the cQED Hamiltonian or the efficiency of quantum measurement and classical control. In the current experiment, we show that DD has better steady-state performance as the cQED Hamiltonian parameters are engineered such that DD has a shorter feedback latency (Fig. 1.8). But DD's advantage over MB is not immutable. As certain experimental parameters are improved, such as coherence times and mea-

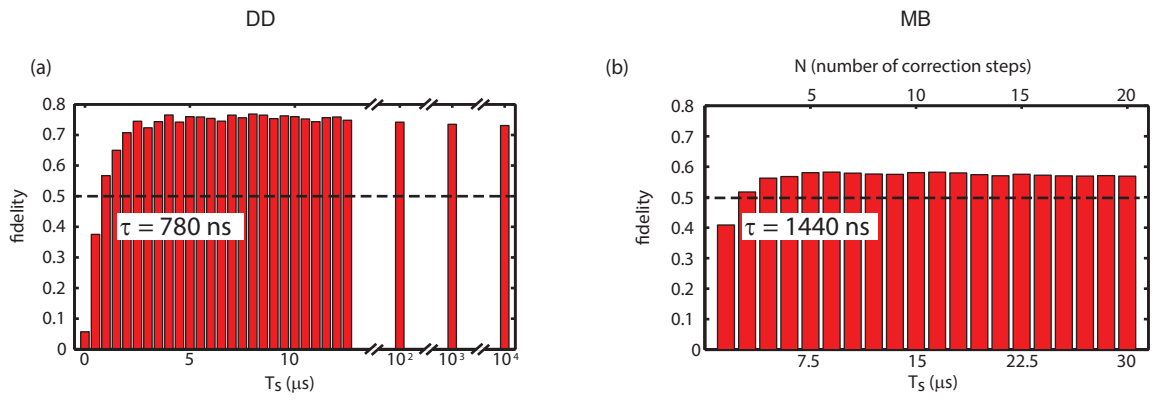

Figure 1.8: Fidelity of Bell state stabilization as a function of stabilization duration. (a) DD stabilization. (2) MB stabilization.

surement efficiency, MB's performance can catch up with DD.

Secondly, in the current situation in which neither the cQED Hamiltonian parameters nor the measurement and control parameters are ideal, we can obtain a boosted performance by combining DD and MB to get the best of both worlds. We explored this by devising a heralding method to improve the performance of both stabilization approaches. This protocol exploits the high-fidelity measurement capability and the programmability of the feedback platform. The protocol is termed "nested feedback" since it has an inner feedback loop based on either the DD or MB scheme, and an outer loop that heralds the presence of a high-fidelity entangled state in real-time. As alluded to earlier in the discussion of Maxwell's demon, this new protocol brings the more powerful version of Maxwell's demon. It adds a new "voluntary" decision-making capability to a pre-existing Maxwell's demon and in particular, for the DD case in which the original demon acts solely on "reflex action", the enhanced demon has now two complementary decision capabilities. The nested feedback protocol significantly improves entanglement fidelity for both the DD and MB schemes without sacrificing success rate (Fig. 1.9).

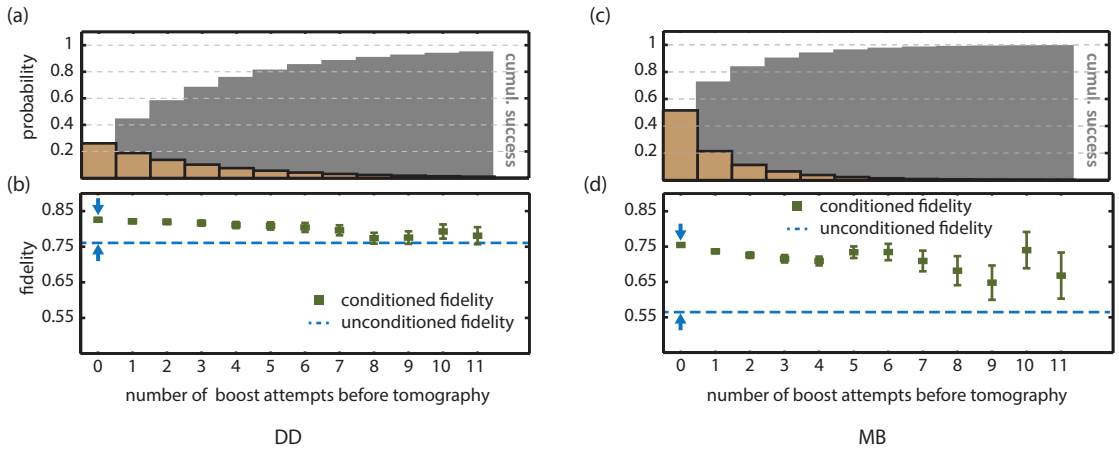

Figure 1.9: Results of the nested feedback protocol for DD and MB schemes. (a),(c) success and cumulative success probability of heralding high-fidelity entanglement for DD and MB. (b), (d) Fidelity to target Bell state as a function of number of "fidelity boosting attempts".

CHAPTER 2

Building the controller

Controlling a quantum system requires the controller to operate with precise timing and low latency. One possible answer to this challenge might be to employ commercial micro-controllers, such as Raspberry Pi's, Arduinos which are popular choices for many classical feedback experiments and are relatively easy to use. However, these controllers do not meet the requirements of quantum control, especially of superconducting qubits. The timing of processes running on them is indeterministic and long compared to the quantum system's coherence time. Another practical problem is that micro-controllers have limited number of available I/O (input/out) ports, which prevent them from interfacing with many other essential devices such as ADCs, DACs, high speed data interface with a computer and etc. On the other hand, a field programmable gate array, or FPGA, is excellent at all those areas.

2.1 A primer on FPGA

2.1.1 Architecture of an FPGA

The fundamental work horses inside a modern FPGA are LUTs (look-up tables[1], they are very small ROMs, typically 32 bit or 64 bit), multiplexers and flip-flops. They are organized into identical blocks and each is called a *configurable logic block* (CLB), the basic unit of an FPGA (Fig. 2.1). A high-end FPGA, such as Virtex 6 from Xilinx (the biggest FPGA maker) has around 40,000 CLBs. Each CLB can be configured by specifying the contents[2] of the LUTs and the connects between the LUTs, multiplexers and flip-flops. One CLB can already implement an arbitrary boolean function equivalent to a dozen logic gates. The programmability of an FPGA is achieved by configuring these CLBs and the interconnects between them.

[1]The word, "look-up tables" is used to mean slightly different things in different contexts in the FPGA community, which can be confusing. It is used later to denote a memory table implemented by block RAMs that store data used for custom processing on an FPGA. Here, however, it specifically refers to these small ROMs in CLBs

[2]The contents are truth tables for boolean functions. An example and explanation of a truth table is shown in Chapter 3.2.2.

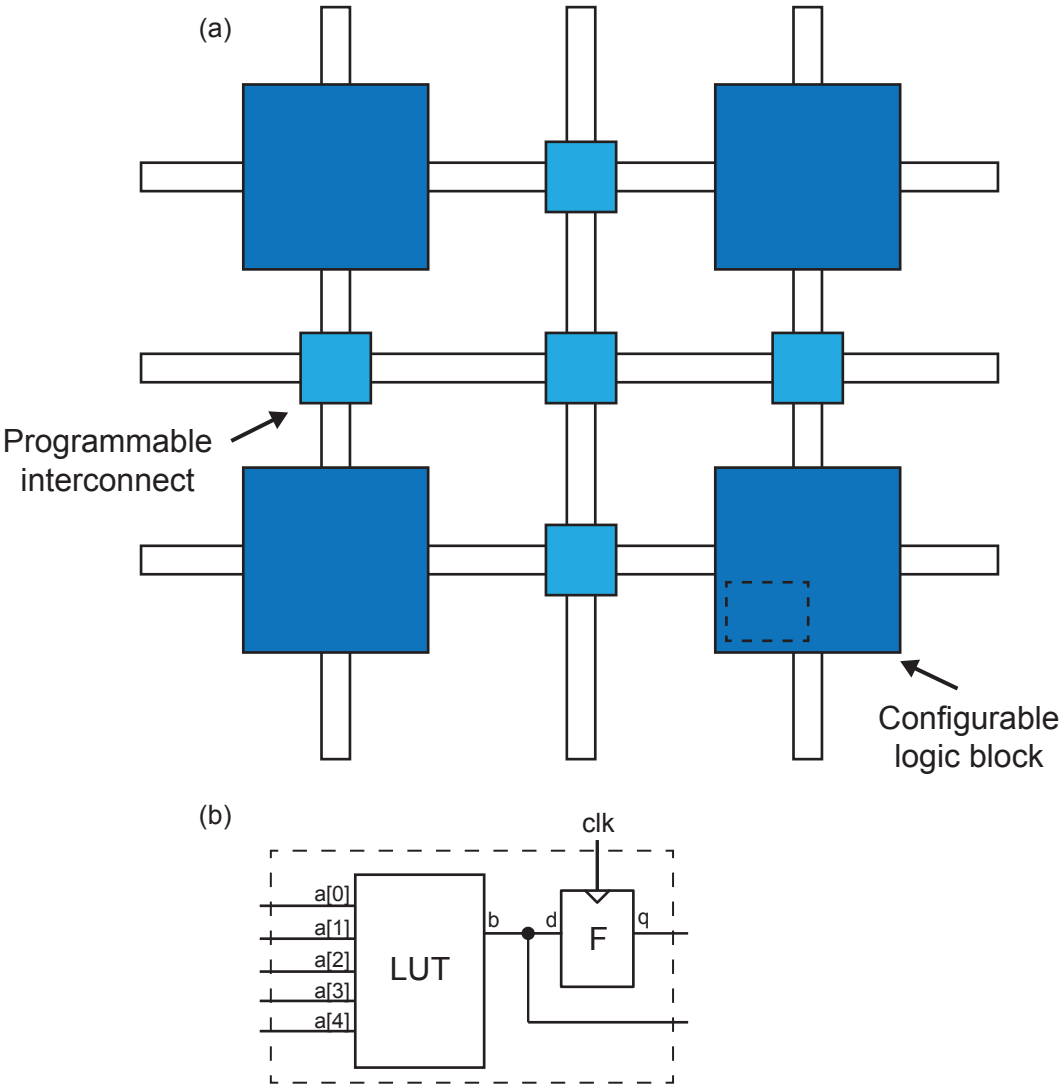

Figure 2.1: Schematic of a FPGA. (a) An FPGA is made up of a fabric of programmable logic blocks and switches between them (b) Inside each configurable logic block is several logic cells comprised of a look-up table and a flip-flop (register).

Besides the CLBs, modern Xilinx FPGAs also have many special hardware features[Xilinx]. Each Xilinx FPGA has clock management titles which filter the jitter of incoming clock signals and can generate new internal clocks by digital frequency synthesis. There is a built-in network of clock lines that distribute clock signals to every logic block with very short propagation delay and extremely low skew[3]. As we will discuss in Chapter 3.2.1, there are many clocks running on an FPGA. Different regions may use different clocks for different purposes. Nonetheless, there is usually one clock, called the system clock that synchronizes the majority of the logic processes on an FPGA; its frequency is typically limited to ∼300 MHz.

Xilinx Virtex FPGAs have dedicated memory components, called block RAMs. A Virtex-6 FPGA contains up to a thousand block RAMs, each with a capacity of 36 Kbits. The block RAMs can be initialized with memory contents during configuration or written during live operation. They can be grouped into memory tables of various sizes. All read/write operations on the block RAMs are synchronized to a clock. Since these block RAMs are embedded in the logic fabric as opposed to existing as external RAMs outside the FPGA IC chip, the read/write access can be completed within just a few clock cycles (on a clock with frequency ∼300 MHz). They are utilized exhaustively by all the custom look-up tables we implement in our logic design.

Virtex 6 FPGAs also have digital signal processing components, known as DSP slices, which are dedicated multipliers and accumulators. Instead of implementing these arithmetic operations on the CLBs ourselves, we use these embedded components which are designed and optimized especially for binary multiplication and addition in signal processing applications, such as demodulation.

In the next two sections, we talk about two important paradigms in digital

[3]Clock skew is the difference in timing that different components connected to the same clock signal see a clock transition

circuit design that are adopted extensively in our FPGA logic development.

2.1.2 Synchronization

There are two types of logic circuits, namely combinational logic and sequential logic[Harris and Harris, 2012]. An FPGA can implement both of them. The output of a combinational logic circuit depends only on the current input. It is *memoryless*. The output of a sequential logic depends on both the current input and the previous input. It has memory. The most common type of sequential logic is called synchronous sequential logic. A flip-flop, also known as an 1-bit register, is the simplest synchronous sequential logic circuit(Fig. 2.2) and is the building block for all synchronous circuits. The output of a register only gets updated on the rising edge of an input clock and all the registers in one synchronous circuit receive the same clock. As a result, the *state* of a synchronous circuit are stored by the registers and the transition into a different state can only occur at specific times marked by clock transitions. Almost all the logic designs covered in this thesis are based on synchronous logic circuits.

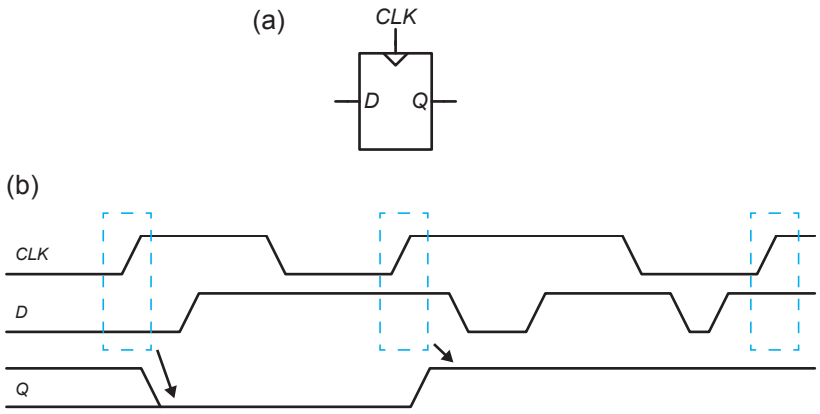

Figure 2.2: Register is the basic element of a synchronous FPGA logic circuit. (a) Schematic symbol of a register. (b) Timing diagram of a register. The input, D, is sampled and the output, Q, is updated on the rising edge of the clock exclusively. The output maintains its state at all other times.

By using synchronous logic, we guarantee that parallel processes can propagate and finish at the same time, by virtue of accessing the same clock. The timing is also deterministic since all operations are measured in exact number of clock cycles.

2.1.3 Parallelization

We already recognized the FPGA for its high speed processing capability. Now we will be more precise by what we mean by speed and how it is achieved. Speed of a processor is measured by latency and throughput[Harris and Harris, 2012]. First, we will make the following definitions to streamline the discussion:

Token: a token is a batch of data that propagates through a circuit from the input to the output.

Latency: the time it takes for the token to travel from the input to the output is called latency.

Throughput: the number of tokens the circuit produces per unit time is called throughput.

There are two ways an FPGA achieves superior processing speed: they are spatial parallelism which is usually simply called parallelization, and temporal parallelism which is called pipelining.

Parallelization boosts throughput and this is very intuitive. We duplicate the hardwares such that multiple tasks can be done simultaneously. The architecture of an FPGA, which consists of large array of identical resources, from CLBs, block RAMs, to DSP slices, naturally supports parallelization. Multiple instances of a component can be implemented on an FPGA. This will become very clear later in this chapter and the next chapter where we demonstrate specific functionalities implemented on FPGAs.

In pipelining, a task is broken up into multiple stages. The *same* hardware can

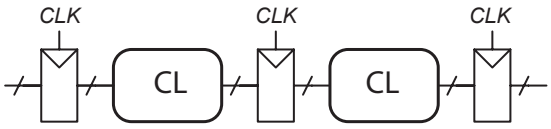

Figure 2.3: Circuit diagram of a pipeline. A pipeline is a series of combinational circuits punctured by registers. It increases data throughput.

process multiple tokens at once through a pipeline. The tokens go through the pipeline one stage at a time like an assembly line. In digital logic, pipelining is done by inserting registers in the combinational logic chain that does the processing (Fig. 2.3). Here is an example: in a Xilinx Virtex 6 FPGA, with a system clock of frequency 250 MHz, multiplication of two 18-bit numbers by a DSP slice can take at least three clock cycles, i.e., naively, one multiplication process has a latency of 12 ns and a throughput of $\frac{1}{12}$ per 1 ns. Fortunately, the multiplier has a built-in pipeline where three registers break up the multiplication process in four stages. As a token moves from the first stage to the second stage, a new token enters the first stage. The pipelined multiplier's latency increases to 16 ns (4 clock cycles) but the throughput improves by three fold to $\frac{1}{4}$ per 1 ns! Pipelining improves throughput using the same hardware resources at the expense of latency.

For signal processing applications, a particular throughput is demanded, which is determined by the sampling rate of the input, such as the rate at which ADCs transmit data to the FPGA. Typically a combination of both parallelization and pipelining of the processing logic circuits is required to maintain the throughput.

There is another important benefit of using pipelines, that makes them ubiquitous in the FPGA projects we developed. For a large logic design that uses extensive amount of resources, a large physical area of the logic array is used. Significant routing delay can occur for a signal traveling from one part of the array to another part. However, there is the requirement that a signal's logic level transition must occur a certain time before the rising edge of the clock.[4] This can be challenging for large and high speed logic designs where clock frequencies are more than 250 MHz. With the help of pipelining, we break up the route into smaller stages by inserting one or more registers. Instead of needing to cover a significant journey within a clock cycle, a signal only has to travel between two relatively closely positioned registers within a clock cycle. That said, each additional stage of pipeline introduces one extra clock cycle of latency and indeed a significant amount of latency for the feedback controllers we implemented came from the necessary pipelining required to meet timing constraints.

2.1.4 Logic development process

[4] In a synchronous circuit, there are what are called "setup" and "hold" time constraints. A signal to be sampled must remain stable for at least setup time, t_{setup} before the rising edge of the clock and must remain stable for at least hold time, t_{hold} after the rising edge of the clock. Otherwise, the sampled result might be *"metastable"* and can resolve to either logic state

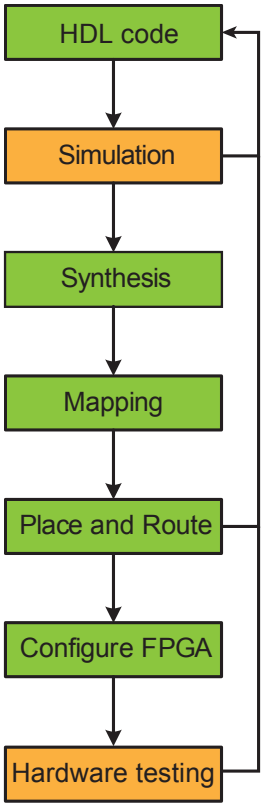

Figure 2.4: FPGA logic development flow. The flow chart describes the typical process of developing a FPGA logic circuit. Green blocks are actual design steps and yellow blocks are verification steps that are often necessary as well.

Before introducing the process for FPGA logic development in this section, we are overdue for a recommendation of some helpful literature for becoming acquainted with the subject. Two excellent texts are [Harris and Harris, 2012] and [Chu, 2011]. The former teaches fundamental concepts and principles in digital circuit design, which form the basis of hardware programming. The latter provides a practical approach to FPGA programming by serving as a "recipe" book for implementing many common applications. Both provide many illustrative examples in HDL.

Writing HDL code

Developing a functional logic on an FPGA for eventual hardware deployment follows a standard design flow (Fig. 2.4). The entire design process typically occurs inside an integrated development software provided by the FPGA maker, i.e., Xilinx. The first step is designing the hardware model using hardware descriptive language (HDL). HDL is a high-level language that describes the behavior of logic circuits using conditional statements, such as if-then-else statements, and arithmetic operators. It provides a textual description of a device in a hierarchical fashion. The main file, also called the top level, describes the entire device with all its external-interfacing ports, its constituent modules and the internal signal connections between them. The constituent modules form the sub-level and are each specified by their own HDL files. This hierarchical layering can continue for many levels for a complex logic design. There are several choices for HDL. The language used in all of our projects is VHDL. The unique part of hardware programming with HDL is in the first two letters of the acronym. Unlike a software programming language which can describe a very abstract object with custom-defined structures, the HDL code must describe a *synthesizable* device using standard digital circuit building blocks, such as multiplexers, registers, finite state machines and etc. Laying out the architecture of the target logic circuit and

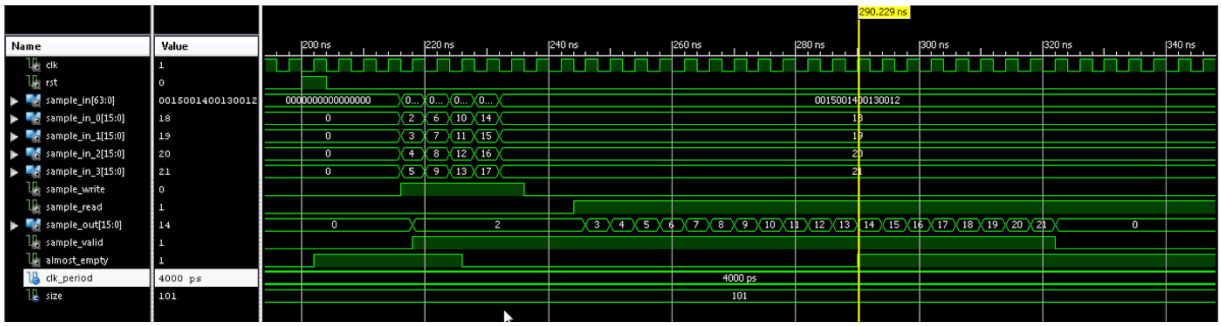

Figure 2.5: An example of a behavioral simulation of FPGA design in action. The digital waveforms are the responses of a FPGA circuit to test stimuli. Logic verification is done by examining the states and timing of the waveforms.

applying principles of digital design, e.g., synchronous discipline, pipelining, all within the hardware limitations of a particular FPGA device, are the most important processes in HDL programming.

Simulation

The next step is to simulate the HDL model. This is especially necessary after writing extensive HDL codes for a new complex design since logic errors, such as boolean algebra mistakes or signal misconnections, can easily occur. For simulation, we use HDL to construct another module, referred to as a test bench that generates test signals (clock, reset, trigger, and etc...) and applies them to the target HDL model[5]. We obtain the target model's responses comprised of both internal signals and external outputs. A simulation software executes the simulation code and displays the responses as digital waveforms (Fig. 2.5), which are monitored to verify that the HDL model behaves as expected.

[5]This is also known as behavioral simulation

Logic synthesis, mapping, place and route

These next several steps are largely automated by the development software. *Synthesis*: the software translates the HDL code to a *netlist* of generic FPGA circuit components, e.g., gates, flip-flops, memory blocks....while applying simplification and optimization of hardware resources usage when possible. *Mapping, placing and routing*: The mapping process maps the synthesized netlist to CLBs and dedicated resources of a specific FPGA. The placing and routing process determines final layout of the logic circuit – it picks the physical locations of CLBs to be used and routes the signals. These two processes form what is called the *implementation* stage. This is also the stage that timing constraints are verified. The software checks that maximum propagation delay between a source and the destination does not exceed the clock cycle that the signal is synchronized to.

Configuring and testing

In the final step of programming an FPGA, the software converts the fully placed and routed circuit layout into a binary file (also known as a configuration file) that is then downloaded to the FPGA and configures its logical blocks according to the design.

The most important test is the hardware testing of the final configured FPGA. Unfortunately, this is the least transparent and most frustrating part of developing a working FPGA design. In traditional software programming, a good debugging tool helps locate the source of the problem by providing access to the states of all the variables we want to check. Hypothetically speaking, in hardware programming, if we can probe any signal we want inside an FPGA circuit, we can also in principle diagnose any problem in a similar manner. Unfortunately in reality, only a very limited number of signals can be ported to outside the FPGA for external probing. This lack of transparency makes diagnosis of hardware bugs very

difficult. This also explains why behavioral simulation before the implementation stage is extremely important since it can rule out obvious logic errors.

However, the complication to hardware troubleshooting is that the hardware realization of a logic design could still fail to work without showing any behavioral simulation error. Failing to meet timing constraints or abide by hardware specifications of an FPGA device is usually the cause. To manually place and route critical modules to mitigate signal propagation delays is one solution (Fig. 2.6)[6]

[6]In practice, the development software also lets us generate different circuit layouts using different placing and routing algorithm options. Tweaking these options can often make a difference between a working FPGA and an useless one. Each layout results in a different binary file. Finding the working configurations is largely done by trial and error: we test each of the files by loading it to the FPGA and verify that all the features are performing as intended.

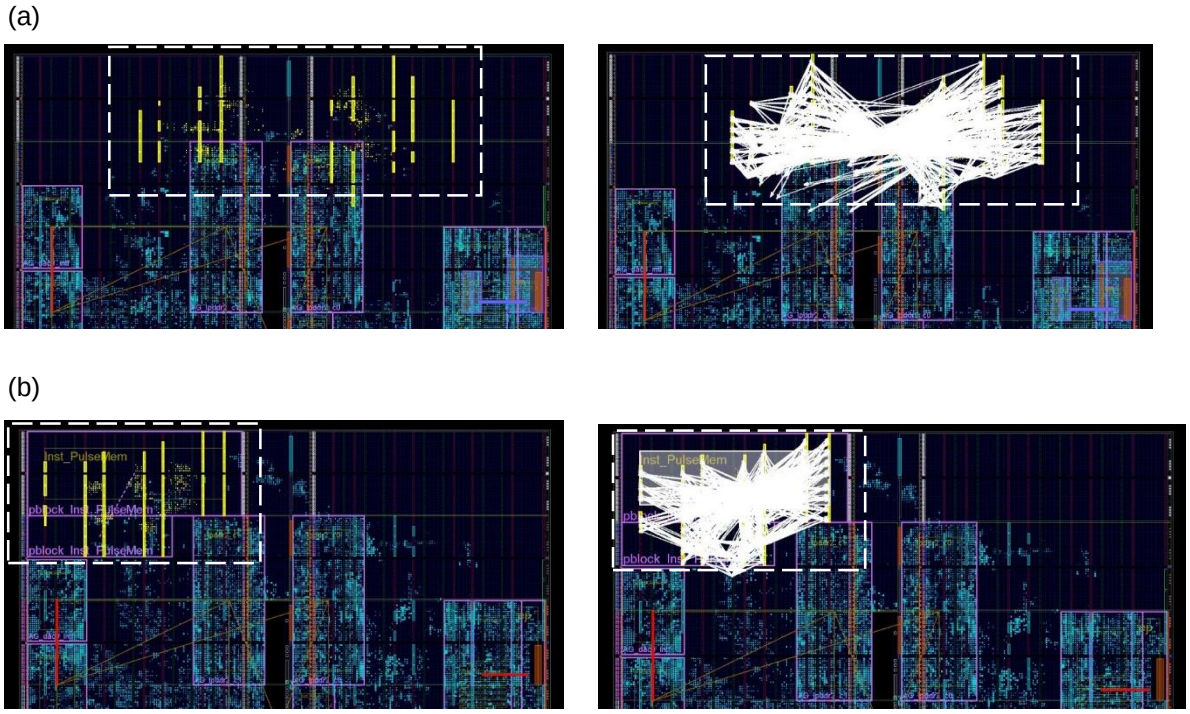

Figure 2.6: Constraining a FPGA circuit layout. Xilinx provides a design tool called PlanAhead which gives guidance to manual placement and routing of logic components. The tool shows the circuit layout of a logic design in an array of logic cells. Each figure here shows a portion of the FPGA circuit (approximately half of the FPGA chip). The cells that are ″lit up″ in the tool's visualization (blue and yellow here) are cells that are used in the design. (a) The highlighted elements in yellow are the memory blocks used. The white traces shown in the right panel are signal connections between the memory blocks. Without constraining, they are sprawling over a large area that causes timing difficulties. (b) Constraining them to a smaller area reduces routing delay.

2.2 FPGA-based AWG

To complete a feedback loop, an essential component is the actuator. For quantum feedback with superconducting qubits, the manifestation of the actuator has typically been the arbitrary waveform generator (AWG). AWGs are used in both open-loop and close-loop experiments. In the usual open-loop qubit experiments, an AWG outputs two analog pulses, on the two quadratures, I and Q, which modulate a LO (local oscillator) tone generated by a microwave generator through an IQ mixer. The RF output of the IQ mixer is the signal that controls a qubit. In addition to the analog outputs, an AWG also generates several digital pulses, for either controlling switches on the output of the IQ mixer to prevent signal leakage or for modulating cavity drives.

2.2.1 The traditional way

The commercial solutions for AWGs are convenient but are expensive and more importantly not very flexible[7]. To generate analog and digital sequences on one of these AWGs, the experimenter needs to hardcode the entire waveform of each sequence on a computer and transfer the waveform data from the computer to the AWG.

There are several problems with this approach. First, for hardcoding a pulse sequence, one needs to generate the value (2 bytes per point) at every waveform point, even if it is a pulse of constant amplitude or simple delay of zero amplitude. This requires transferring a lot of data to the AWG. Data transfer through GPIB communication is slow and usually takes about a minute. Second, AWG memory is limited. The common Tektronix AWG in the lab can play approximately 100 sequences of 8000 ns long waveforms (2×8000 wave points for the two analog

[7] the latter drawback has been improved in the latest models of AWGs

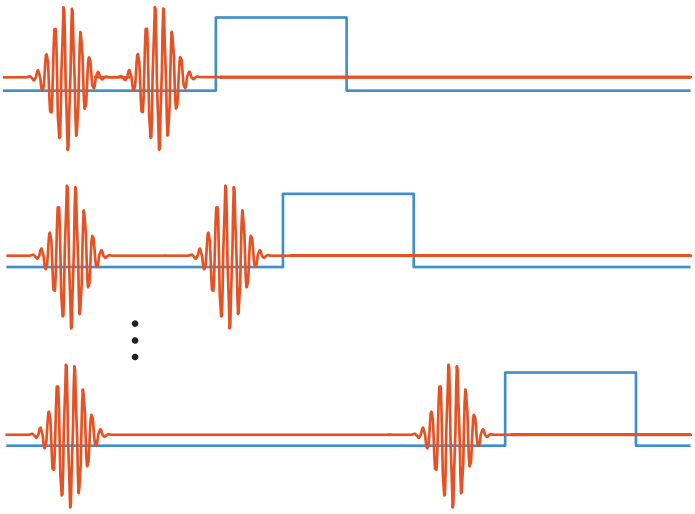

Figure 2.7: Pulse sequences for a T2 experiment, as required by a commercial AWG. The red waveforms are the qubit pulses and the blue waveforms are the digital marker pulses for modulating measurement drives. Every waveform point, including the parts with constant amplitudes, needs to be stored. This applies to both the analog and digital channels

channels of a 1GSPS AWG). A significant amount of memory is wasted on storing the values of constant pulses, such as the delays. Playing longer waveforms then requires reducing the number of sequences.

Third, using commercial AWGs to play waveforms limits the potential of feedback experiments. As several groups have demonstrated[Campagne-Ibarcq et al., 2013; Riste et al., 2013], it is possible to use a real-time controller to conditionally trigger an AWG to play a particular pulse, which otherwise plays nothing. This, however, restricts the complexity of feedback experiments that may demand more flexible waveform generation, beyond a simple decision of pulse or no pulse[8].

[8]In newer generations of Tektronix AWGs, the waveform generation method is similar to the FPGA method shown next, thus conditional branching is possible. But the latency between receiving an external marker and the time the AWG decides on a branch is still much more significant than the latency of similar process on the FPGA controller

2.2.2 The new way

Commercial AWGs are designed for many applications in vastly differently fields. Therefore it is no surprise that they are not perfectly suitable for quantum information processing experiments. Since they make no assumption about how we compose the sequences, Tektronix AWGs provides the "arbitrary-ness" in waveform generation at the expense of efficiency and flexibility. In practice, the pulses that we send to the quantum system in an experiment, such as the qubits, are far from arbitrary.

Since FPGAs became more widely known in the science research community, they have been recognized for their usefulness for quantum control and feedback[Stockton et al., 2002]. Building an FPGA-based AWG is a first step towards a full FPGA controller in quantum feedback. John Martinis from UCSB built the first FPGA-based AWG for control of superconducting qubits[Electronics]. The work started by the author in this thesis also coincided with a similar effort by BBN to build FPGA-based AWGs. The architecture of our FPGA-based AWG borrows from the organization of a basic microprocessor. This architecture, compared with the traditional AWG, no longer wastes precious memory space on storing values of constant-amplitude pulses and thus allows playing waveforms of greater number and/or length. The architecture has three key components: instruction memory, pulse memory and the control unit (Fig. 2.8).

Instruction memory

The instruction memory is a look-up table stored in block RAMs that contain a sequence of instructions specifying the pulses to be played in a particular order. Each entry or "word"[9] in the table is an instruction that provides the type of a

[9] A memory block's capacity is defined by its *width* and *depth*. A memory block is nothing but a table of bits: the *width* tells how many bits there are in a row, also called a *word*. *Depth* gives how many rows or how many words there are. All look-up tables implemented from block RAMs on

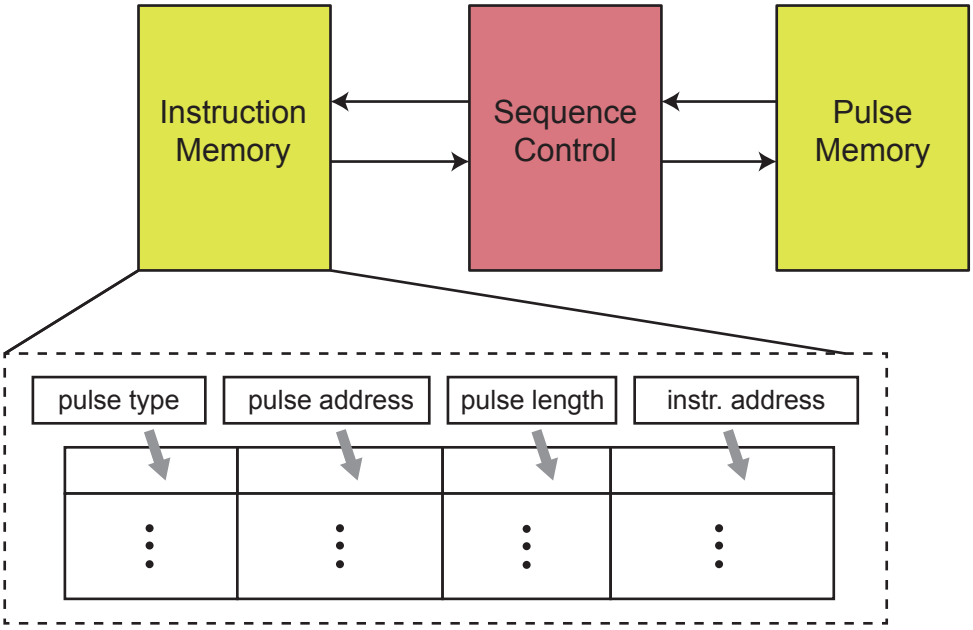

Figure 2.8: Architecture of a FPGA-based AWG. The three main components are the instruction memory, pulse memory and sequence control. Sequence control coordinates reading of both memories by specifying the locations to read from, i.e. memory addresses. The instruction memory is a table consisting of rows of instruction "words", as shown inside the dashed box. Each word has four fields: three of them store values parametrizing a pulse and the fourth stores the location of the next instruction.

pulse to be played, its length, the address of the pulse in the pulse memory, and the memory address of the next instruction. Figure 2.9b gives an example of an instruction memory. If we think of sequences of pulses to be played as a program, then the instruction memory stores the commands of this program. The capability to update the commands by rewriting the memory in vivo (without reconfiguring the FPGA) between experiments gives the programmability of the FPGA-based AWG. The use of instruction memories features extensively in our logic designs and will be seen more later.

Pulse memory

The idea of a pulse memory is simple. It is a "dictionary" or list of all the unique non-constant-amplitude pulses (Figure 2.9a). The pulse address in the instruction memory gives the location of the pulse in the pulse memory. The read port of the memory accepts an address as an input which specifics the location to read the waveform. The pulse memory is organized as a table in which each row is a 64-bit word. Since each waveform point is represented by a 16-bit DAC number, one word contains four waveform points. Playing a pulse entails reading several consecutive words by incrementing the address input at the read port.

Control unit

Sequence control is the component that orchestrates the reading of the instruction memory, the parsing and execution of the instruction and the playback of the waveforms from the pulse memory. It reads a word from the instruction memory, determines the type of pulse to be played from the instruction. If the type is a non-constant pulse, it finds the pulse in the pulse memory according to the pulse

the FPGA have two independent ports, one write port and one read port. Each port expects an address to read from or write to. In addition, the write port accepts a data word which fills the bits at the specified address.

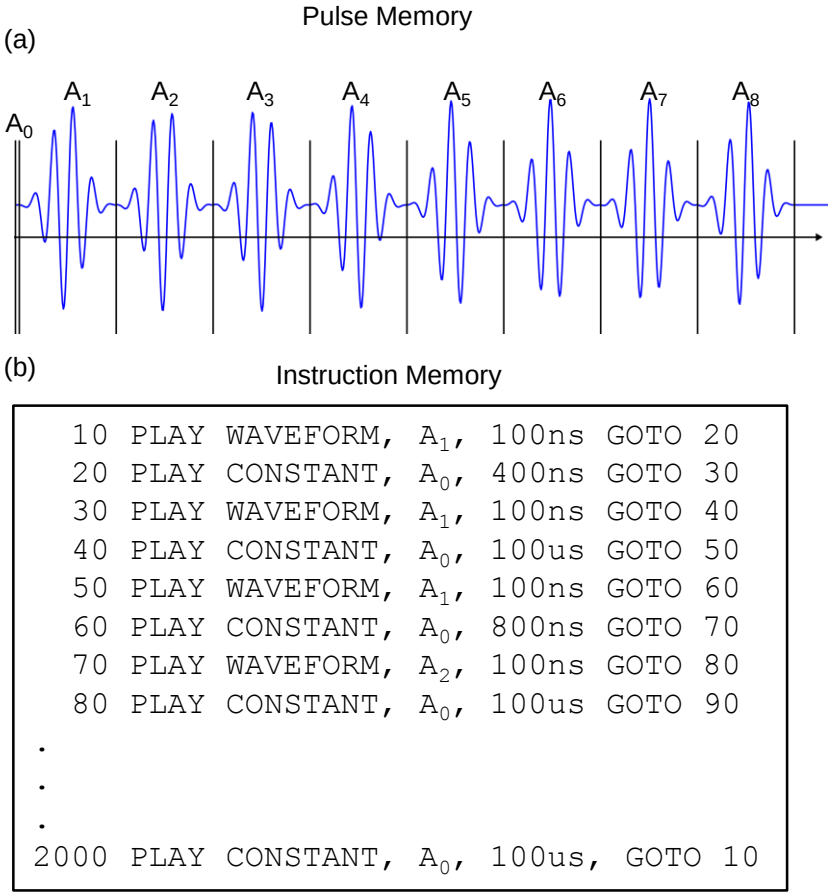

Figure 2.9: Construction of Pulse Memory and Instruction Memory. (a) Pulse memory is essentially a list of pulses. Each pulse is identified by the address of its first waveform point in the pulse memory, represented as A_n in the figure. (b) An pseudocode interpretation of an instruction memory, in the format specified in the previous figure.

address and plays it for duration set by pulse length. If the type is a constant-amplitude pulse, what is stored in the "pulse address" field of an instruction is the DAC value of the pulse and the control unit does not need to look up the pulse in the pulse memory. Once the current instruction is completed, it proceeds to the next instruction as specified by the "*next* instruction address" contained in the instruction. The control unit passes the outputs from the pulse memory to downstream logic components that interface DACs. The operation of the control unit is based on finite state machines, which we will describe in details next.

Finite state machine

A finite state machine (FSM) models a sequential system that transits between a finite set of internal states. At a given moment, the state of the FSM is called the current state and the input it sees is called the current input. State transition can depend on current state or/and current input. Applications of FSM are ubiquitous and are seen in situations that call for a predetermined sequence of responses conditioned on sequence of events presented: traffic lights, vending machines, turnstiles in subway stations.

A FSM is implemented in digital logic by two combinational logic blocks, called next state logic and output logic respectively (Fig. 2.10). The next state logic computes the next state based on the current state and the current inputs. A register stores the next state and updates the current state when the clock transitions. The output logic block computes the outputs of the FSM conditioned on the current state[10].

The operation of a FSM is represented by a state transition diagram. There

[10]For completeness, there are actually two types of FSMs. One is called Mealy FSM in which outputs depend on both the current state and the current input. The other is called Moore FSM in which outputs depend only on the current state. One can transform from one implementation to the other. For clarity of presentation, all FSMs shown in this thesis are Moore FSMs. In practice, Mealy FSMs are used more often since it involves fewer number of states and therefore has less latency

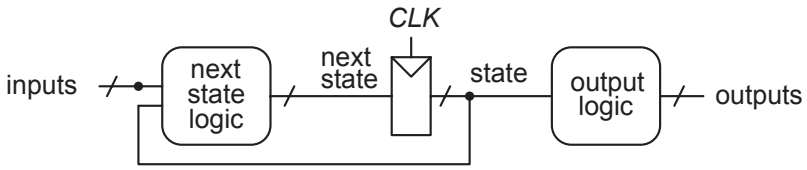

Figure 2.10: Logic circuit schematic of a FSM

are two equally valid ways of drawing such diagram. One is simply known as the state diagram (Fig. 2.11a). The states are drawn as circular nodes and transitions between states as arcs. The boolean condition for each transition is shown above each arc. The outputs or the action taken during a state are shown inside the node. There is no restriction on the total number of arcs entering or leaving a node since a state can transition to any number of states and vice versa. But the number of *labeled* arcs[11] leaving a node must be even since for every boolean condition there exists its complement. A complex FSM with many states can often be reconstructed as multiple simpler FSMs with fewer states (this is known as factored design[Harris and Harris, 2012]). A state diagram is the most popular choice for representing FSM because of its compactness and clarity. We briefly mention the other representation here, called algorithmic state machine (ASM) chart which resembles a flow chart (Fig. 2.11b)[Chu, 2011]. This method is very helpful for building FSMs on an FPGA since the ASM chart can be made more descriptive than a state diagram and the composition of the chart more closely matches how an FSM is implemented in HDL.

[11] An unlabeled arc represents unconditional transition. If there is only one arc leaving a node, it has to be unlabeled.

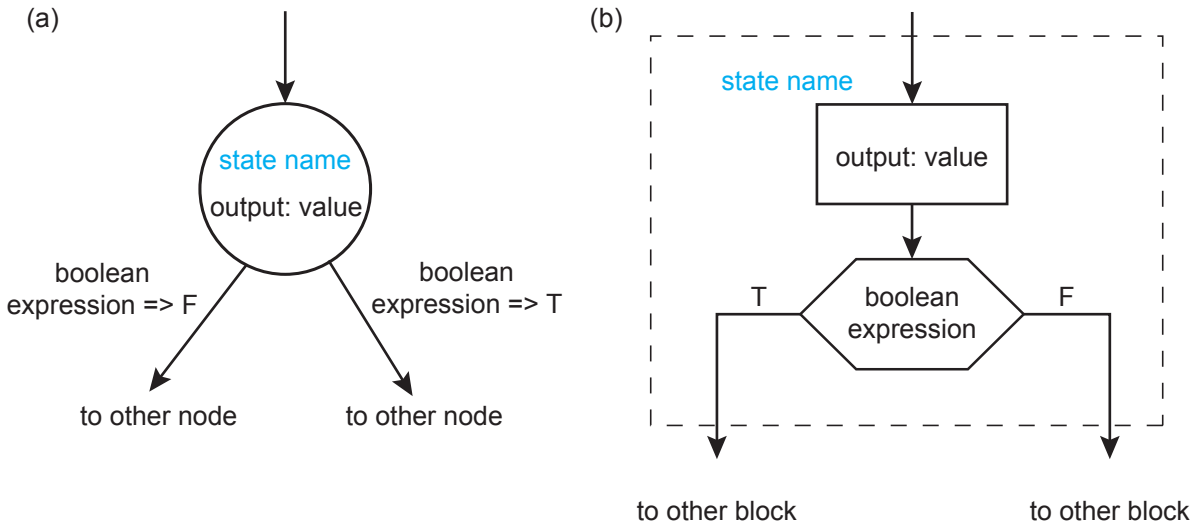

Figure 2.11: Two conventions of drawing a FSM state transition diagram. (a) a state diagram. (b) a algorithmic state machine (ASM) chart.

Sequence control by FSM

We can now demonstrate the application of FSM in implementing the control unit of the FPGA-based AWG. The sequence control consists of two FSMs (Fig. 2.12a). One FSM serves as the master which reads the instruction memory. The other one acts as the slave FSM which communicates with the pulse memory. This is an example where we have *factored* a more complex FSM into two smaller and simpler FSMs. Besides the control signals shown in Figure 2.12a, the master FSM also passes the instruction word it fetched from the instruction memory to the slave, which is not shown explicitly in the figure.

For the master (Fig. 2.12b), the initial state is "idle" in which the FSM does nothing. Right after the FPGA is started, a global reset is applied which puts the master FSM in this state. The master monitors a signal called "dac_ready" which indicates whether a user has a requested to start the AWG *and* whether the downstream components in the FPGA circuit are ready to receive the waveform

DAC data from the sequence control unit.

When "dac_ready" transitions to "1", the FSM enters into the next state, "start", in which it asserts[12] a signal called "start" to the slave FSM. This allows the slave to start executing the playback of the first pulse whose instruction it has already fetched and parsed when it was waiting in the "idle" state.

On the next state transition, there is no label on the arc connecting state "start" and state "playing". This means that the FSM transitions unconditionally from the former to the latter. In the "playing" state, the master waits for the slave FSM to step through the associated pulse memory addresses of a pulse until it receives an asserted "load_pulse_flag" from the slave FSM.

Once the master receives the asserted "load_pulse_flag", it transitions to the state "load instruction". In this state, it reads the new instruction (which it has access to already since the current instruction contains the address for the next instruction) for the next pulse and passes it to the slave. The master then transitions unconditionally back state "playing".

The slave FSM also starts in the "idle" state after the global reset (Fig. 2.12c). It monitors the "start" signal and transitions into the state "output wave" when "start" is asserted. In the state "output wave", the slave fetches the waveform data from the pulse memory by incrementing the read address of the pulse memory every clock cycle (starting from the initial pulse address specified in the instruction). A counter, "length_counter", keeps track of the number of increments and compares it to the value contained in the "pulse length" field of the instruction[13]. When the counter has incremented close to the specified pulse length value[14], it means that the playback of the current pulse is nearly complete; the slave enters

[12] "Assert" in digital design speak means switching a logic signal from "0" to "1".

[13] The "pulse length" of a pulse is not in units of real time but is the number of pulse memory words that the pulse occupies, which is the total pulse duration in terms of waveform points divided by four, since each word contains four waveform points

[14] the value specified by "pulse length" minus two

the state "load pulse" and alerts the master FSM to load the next instruction by asserting the "load_pulse_flag". The slave also resets the counter in this state and transitions unconditionally back to the state "output wave"[15].

The implementation of the sequence control of the FPGA-based AWG shows the application of FSM for controlling a data path, i.e., the extraction of information from two memories in a prescribed order. In the FPGA-based AWG and the FPGA-based controller described later, almost every major logic component has some sort of FSM-based control: the writing/updating of the look-up tables, e.g., the instruction memory and the pulse memories, so that they contain the desired instructions and pulses at the correct addresses rely on data flow mediated by FSMs; the transfer of data from the FPGA to a computer is controlled by FSMs. We will show in Sec. 3.2.2 and Chapter 5.3 that FSM is not only used extensively as a tool for developing logic circuits but also for designing experimental sequences.

2.2.3 Early prototypes

In this section and the next chapter, we will present the hardware realizations of many of the principles and designs discussed in previous sections. We give a chronicle of several generations of FPGA-based pulse or waveform generators developed during the course of this thesis work which culminated in the FPGA-based quantum feedback controller .

Early efforts centered around a commercial FPGA development board, the XEM5010 from Opal Kelly (Fig. 2.13)[Kelly], which features a Virtex 5 FPGA chip. The XEM5010 board connects to a computer through a USB port, through which

[15]The FSM shown here is a Moore machine where the action at each state depends only on what state it is. As mentioned previously, this is for the sake of clarity of presentation and consistency with more general FSMs shown in the rest of the thesis. In actual implementation, the sequence control FSMs used are Mealy machines in which the actions at each state depend not only on the state but also on the input. This reduces the number of states. For example, in the master FSM, the "start" state can be merged with the "idle" state. "Start" signal is then asserted when "dac_ready" switches to "1".

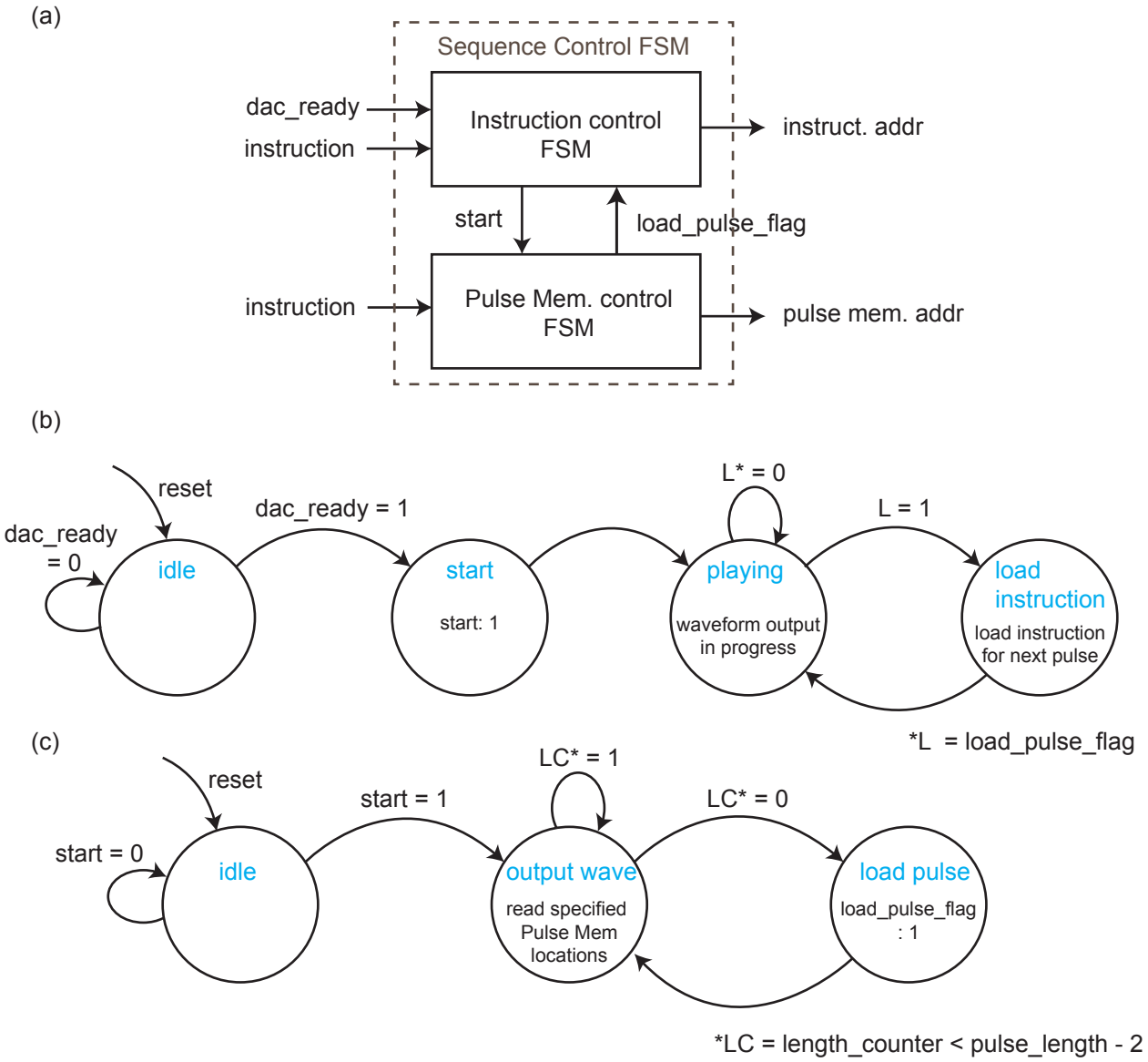

Figure 2.12: Implementation of the sequence control component of the FPGA-based AWG as a FSM. (a) A circuit schematic of the sequence control FSM implemented as two FSMs. (b) State diagram for the instruction memory control FSM which acts as the master. (c) State diagram for the pulse memory control FSM which is the slave.

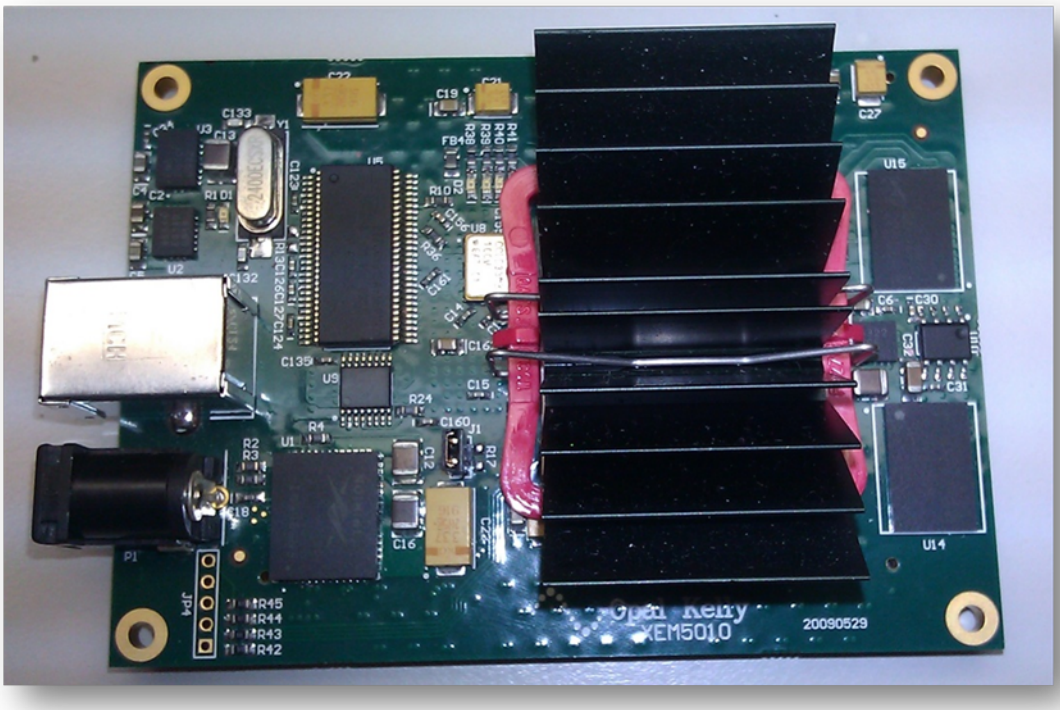

Figure 2.13: XEM5010 prototype board from Opal Kelly. The center piece of the first several generations of FPGA-enabled pulse generators developed. The large black meta looking piece is the heat sink underneath which sits the FPGA chip.

data communication for FPGA configuration and logic control take place. With the exception of the USB interface Intellectual Property (IP) core [16] provided by Opal Kelly, which is an essential component of any logic design on XEM5010, the board gives us much freedom to program the FPGA and customize the functionalities of most of its I/O pins. Access to the FPGA I/O pins are provided by two high speed board-to-board connectors. To connect the FPGA board with other clocking, digital and analog instruments in the lab, we had to build a daughter-board as an interface.

[16] Intellectual Property (IP) cores are pre-synthesized logic blocks that provide a variety of functionalities. They are provided by Xilinx and third party FPGA development board vendors to customers so that they do not have to implement the complex logic circuits in HDL code themselves.

Figure 2.14 shows the first three generations of daughterboards. The two black long connectors seen on all three boards are two high speed male Samtec board-to-board connectors. They connect to the corresponding female connectors on the XEM5010 board and provide signal access to the FPGA. The part number for each generation refers to the complete unit of the XEM5010 board connected with the daughterboard expansion.

FPGA-PS1, PS2: "PS" stands for pulse shaper. They are named so because they can only generate digital pulses which serve as square wave envelopes to modulate analog pulses generated by other sources. These early prototypes were a first attempt by the author to apply FPGA technology to built useful instruments for experiments. The logic architecture of the pulse shaper is actually quite different from what is described in Sec. 2.2.2. There is no instruction or pulse memory. Instead, a particular waveform playback program is hard coded into the logic. Once the FPGA is configured by a logic design, the pulse shaper can play exclusively one type of sequences. One logic allows the pulse shaper to play "T_2" sequences in which there are two square pulses and a varying delay between them. If an experimenter wants to play "T_2 echo" sequences, they would have to reconfigure the FPGA by a different logic that plays these sequences. Indeed, this is not a great way of utilizing the capability and flexibility of an FPGA and the lessons learned in programming the pulse shapers lead to the more powerful logic architecture. Nevertheless, FPGA-PS1 and PS2 already demonstrated features that commercial AWGs do no have. The parameters of a sequence are stored by registers in the logic whose values can be updated through the USB interface. In the example of the 'T_2" sequences, the parameters are the width of each of the square pulses and the delay between them. Through a Labview interface, one can adjust these parameters on the fly while the sequences are being played and see the change take effect immediately.

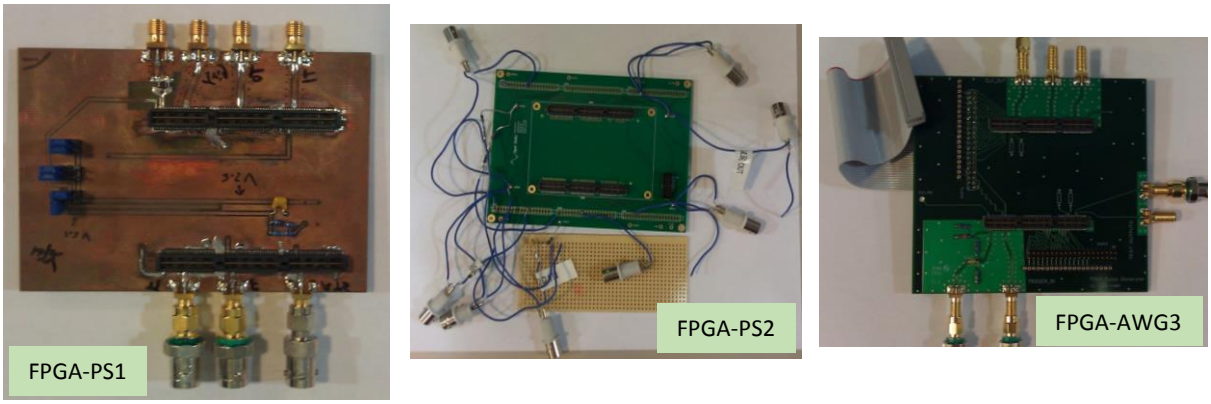

Figure 2.14: Early generations of FPGA-based pulse generators. The daughter- or expansion- boards were designed for the XEM5010 FPGA board.

FPGA-AWG3: this was the first FPGA-based AWG developed here at Yale. It supports both digital and analog output. Through two ribbon cables (one shown in Figure 2.14), the daughterboard connects to a DAC evaluation board that features a 14-bit DAC with output sampling rate up to 300 MSPS from Analog Devices (AD9755-EB). The DAC accepts two input channels (each with 14 bits). A built-in multiplexer in the DAC interleaves the data from the two channels and generates an output at a rate twice that of the input. This allows the FPGA to supply the input digital signal at just half of the DAC's output rate. Reducing the data rate between the FPGA and the DAC makes electric requirements for the daughterboard design less stringent, such that ribbon cables as a means of data transfer are tolerable. FPGA-AWG3 utilizes the logic architecture described in Sec. 2.2.2 and was successful in hardware testing. But the setup was too cumbersome because of the use of the bulky DAC evaluation board and the unwieldy ribbon cables. Since it has only one analog output channel, one would need to integrate two such setups to have two analog channels for IQ modulation required for qubit control.

500MHz-AWG

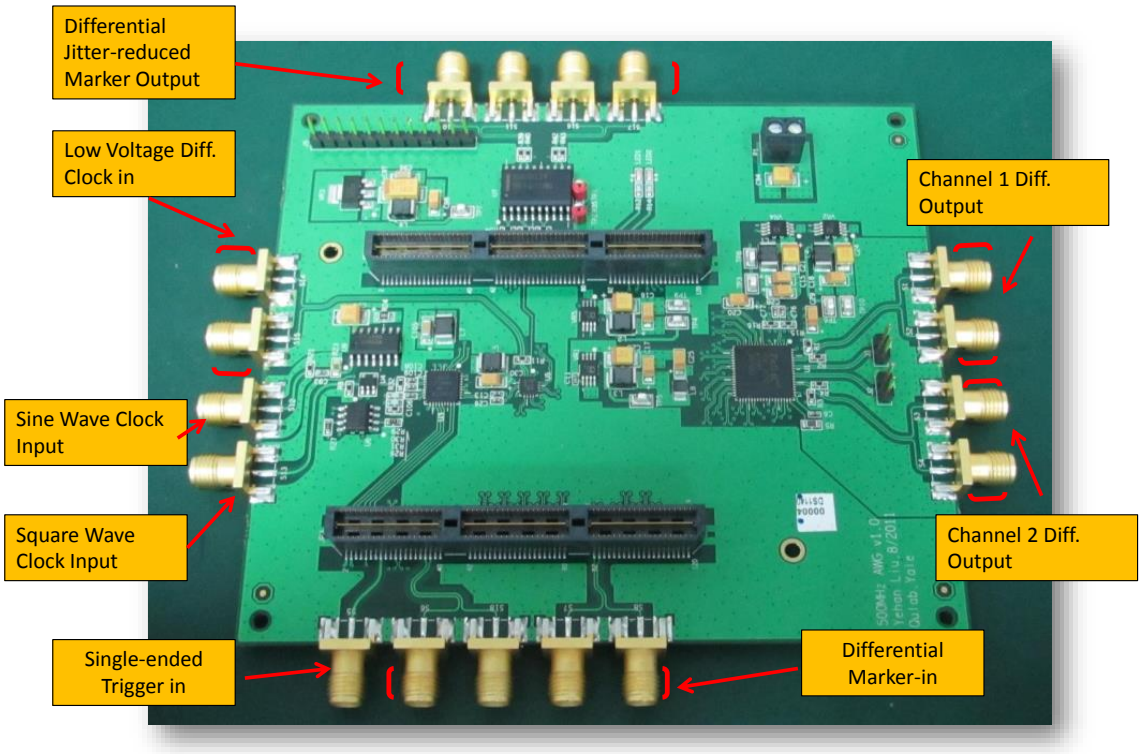

Figure 2.15: The 4th generation of FPGA-based AWG developed at Yale: 500MHz-AWG.

The last generation of FPGA-based AWG using the XEM5010 development board was unsurprisingly the most sophisticated yet. The 500MHz-AWG was developed as a significant improvement to FPGA-AWG3 in terms of capabilities and compactness (Fig. 2.15). It is named so because the AWG can output analog waveform up to 500 MHz. A DAC and several other IC components were integrated on the daughterboard. The key features include:

Dual channel DAC:

The DAC (AD9781) built-in on the 500MHz-AWG daughterboard is more advanced than the one in FPGA-AWG3. The operation of AD9781 is also different

from AD9755, consistent with the difference between very high speed DACs and mid-to-high speed DACs. The AD9781 chip actually contains two 14-bit DACs, each of which has an output sampling rate up to 500 MSPS. AD9781 samples the input digital data at *one* input port at a rate twice that of the output of each individual DAC. The input signal contains data targeted for both DACs in an interleaved fashion. A "deinterleaving" logic in the AD9781 IC splits the data and sends them to their respective destinations. Since the DAC expects input digital signal coming in at a very high rate, i.e., 1 GSPS, it uses the Low-voltage differential signaling (LVDS) standards. Differential signaling protocols are the preferred choices over single-ended transmissions for very high speed digital transfer ($>$ 500 MSPS). LVDS is a very popular choice for digital signal communication between an FPGA and peripheral digital devices. The X6-1000M board, introduced in the next chapter, also uses it. The differential signal travels over a pair of matched traces from the source to the destination. The voltage difference between the traces carries the information and has a swing of 350 mV. Compared to single-ended signaling standards, LVDS is less susceptible to noise, generates less noise itself and consumes low power[Tex]. Because of the two analog output channels of AD9755, one 500MHz-AWG can handle IQ modulation.

Clock generation and distribution:

AD9781 requires a 500 MHz clock as the input sampling clock (input data is sampled at both the rising and falling edge of this clock, thus achieving a sampling rate of 1GSPS). For synchronization, it also outputs a copy of this clock to the FPGA which uses it for streaming data into the DAC input port. The 500 MHz clock is generated by the following scheme (Fig. 2.16): a 10 MHz sine-wave clock from an atomic source is first converted to a square wave clock by a 74AC04 CMOS inverter. This square wave clock then goes through a CMOS fanout buffer (NB3L553) and is distributed to Si5325 and the FPGA board. Si5325 is a low-jitter

precision clock multiplier IC and it outputs a 500MHz clock (in LVDS standard) from the 10 MHz clock input. This 500MHz clock is then sent to a LVDS fanout buffer (CDCLVD1204). The CDCLVD1204 clock buffer distributes one of two selectable clock inputs, clock output from Si5325, or CLK_EXT directly from an external source, to 4 pairs of LVDS clock outputs. The 4 clock outputs are directed to the DAC, the FPGA and two flip-flops (MC100LVEL29, used to generate differential jitter-reduced marker outputs) respectively.

The abundance of clock sources on 500MHz-AWG, from CLK_EXT, CLK_CMOS_EXT (external CMOS standard clock, a square-wave clock input) to CLK_SINE was motivated by a desire to have a fail-safe system in case a botched circuit element on the board prevents one of them from working properly. Stable and working clocks are crucial to digital circuits' operation. Having a clocking system with redundancy gives us a peace of mind when post-production hardware/circuit debugging can be extremely challenging.

SPI control:

The FPGA controls the two major IC components on the daughterboard, AD9781 and Si5325 through Serial Peripheral Interface (SPI), a communication protocol used between a controller and its peripheral devices. This feature is essential for advanced IC components, such as the two above-mentioned, which require calibration and have many programmable options (the AD9755 DAC in FPGA-AWG3, on the hand, was simple enough that there was no control interface). Two SPI controllers were implemented as FSMs in the FPGA logic to send commands to and read status from the two components, respectively.

Multi-layered PCB layout:

Due to the high speed nature of the signal transmissions on the daughterboard, we paid careful attention to the board layout to ensure signal integrity. To

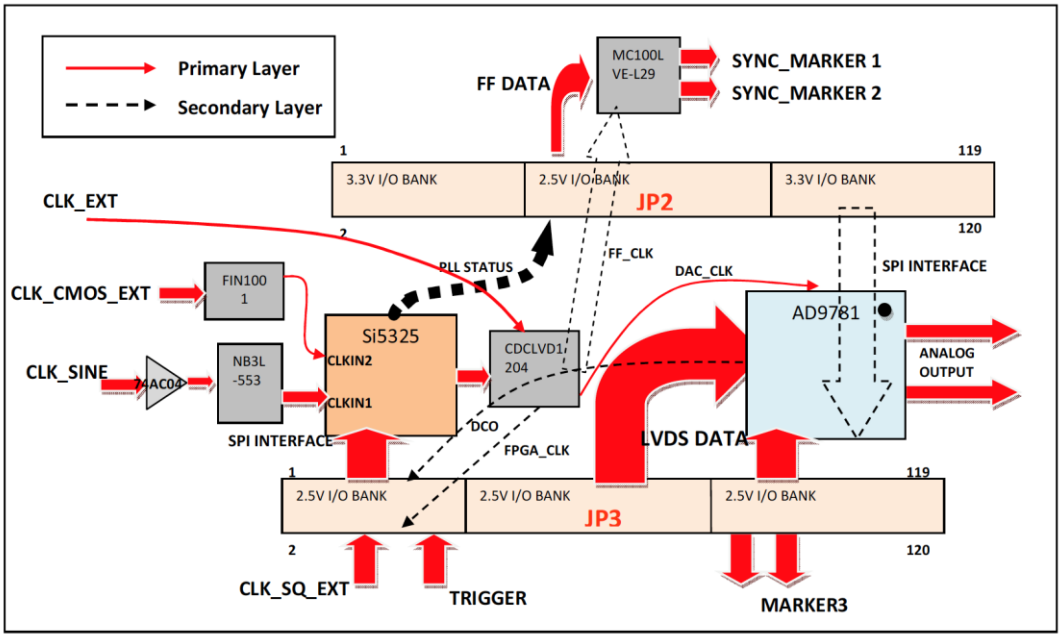

Figure 2.16: Signal routing of the 500MHz-AWG daughterboard

avoid signal reflections, all critical single-ended signal traces (clocks and data) are matched to 50 Ω impedance. Each pair of differential signal traces are laid out such that their lengths match within 0.2 mm and their separation distance stays uniform. Furthermore, they are matched to 100 Ω differential impedance. Impedance control is achieved by routing the traces as microstrip lines. This requires adding PCB layers as ground planes. The board has a total of six layers to accommodate all the components and signals. Two of them serve as primary and secondary signal routing layers (Fig. 2.16). Two of them are ground planes. Ground planes are further partitioned into analog, digital and clock ground sections. This avoids noise cross-contaminations between the different types of signals. See the appendix for more details on the schematic and layout of the 500MHz-AWG daughterboard .

The 500MHz-AWG daughterboard was fabricated and assembled by a professional PCB manufacturing company in China, chosen for its extremely competi-

tive pricing. Much to our relief, the first and only batch was a success. We were able to verify all the key functionalities, including synchronization, digital and independent analog waveform generation on both channels. During operation, contents for the instruction memory and pulse memory were generated in Mathematica. They were uploaded to the FPGA through the USB port in a Labview environment. At the time of testing 500MHz-AWG, BBN also demonstrated to us their version of an FPGA-based AWG which had comparable features and utilized a very similar logic architecture. Their AWG design has a higher analog output sampling rate of 1 GSPS while using an FPGA (made by Lattice instead of Xilinx) from an older generation with smaller block RAM capacity.

CHAPTER 3

All-in-one controller for quantum feedback

The 500MHz-AWG and the similar AWG developed at BBN improved upon the capability of a commercial AWG. But an improved AWG alone cannot execute a feedback experiment so long as a conventional computer is still used for demodulation and state estimation.

In the traditional experimental setup (Fig. 3.1), signal from a quantum system is sampled and digitized by a data acquisition card such the Acqiris or Alazar digitizer, which transfers the raw data to a computer. The Alazar card (ATS9870) has two analog input channels: each ADC converts an input sample to a 8-bit number (1 byte) every 1 ns, amounting to a total input rate of 2 GB/s. The maximum throughput between the card and a computer's RAM by the PCIE connection, however, is 1.4 GB/s, much less than the raw data input rate. A further complication arises since RAM space is limited, data transfer from the RAM to a hard drive is necessary, which takes place at a much slower rate of 140 MB/s. All this means that the duty-cycle (i.e., on-off ratio) of measurements must be much less than 100% and highly optimized software is required to process the raw data from the RAM to reduce the data flow from RAM to hard drive and to avoid overflow[Brecht, 2012]. The challenge of managing the throughput limits tells us that transferring raw data from a digitizer to a computer is not an efficient way

of transmitting information. More importantly, processing the raw data still takes time on the order of milliseconds on a computer and the timing is not deterministic. For quantum feedback, the digitization, demodulation and state estimation need to occur on the *same* dedicated hardware system such that the transfer of raw data can be avoided and timing is deterministic.

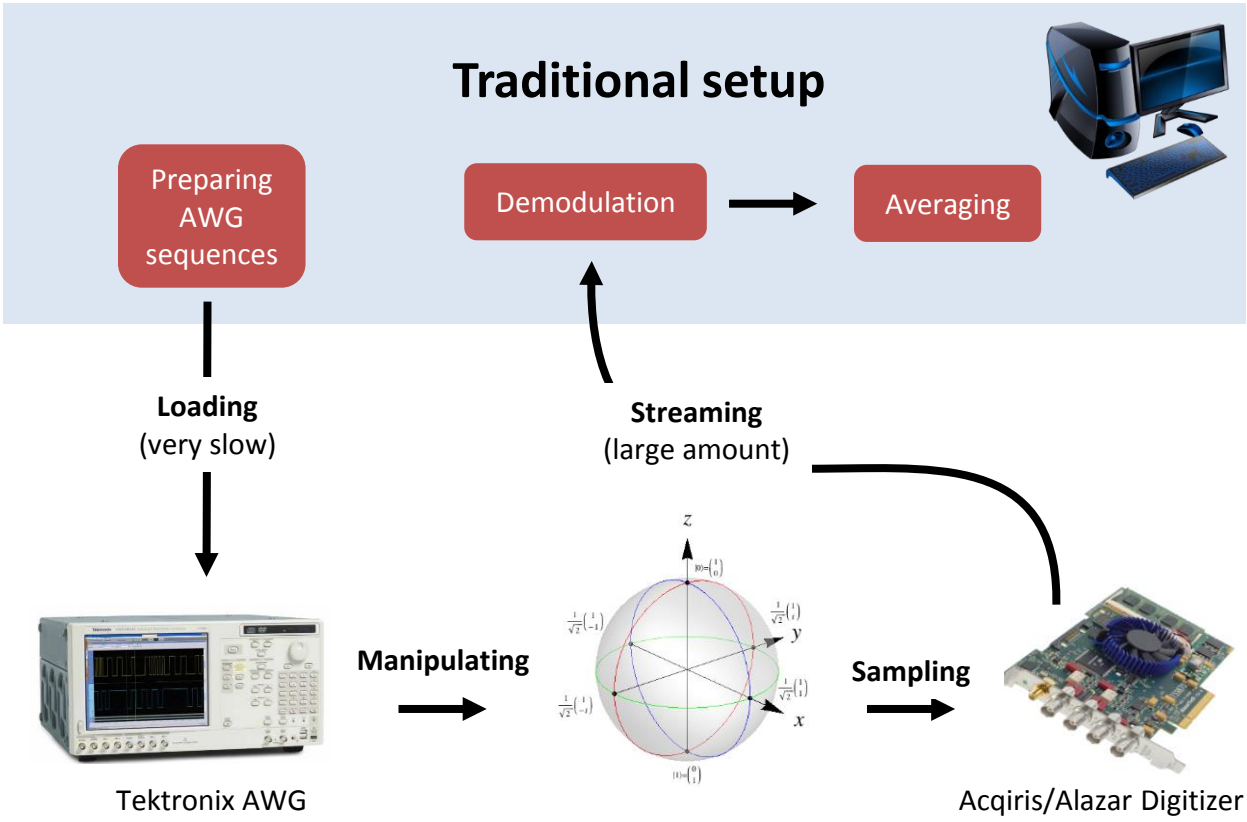

Figure 3.1: Traditional experimental setup for qubits control and measurement consists of three separate units. An AWG generates the control pulses to the quantum system. An analog-to-digital data acquisition card, such as an Alazar card, records the quantum system's responses. A computer is responsible for both generating the pulse sequences played on the AWG and for processing the record obtained by an Alazar.

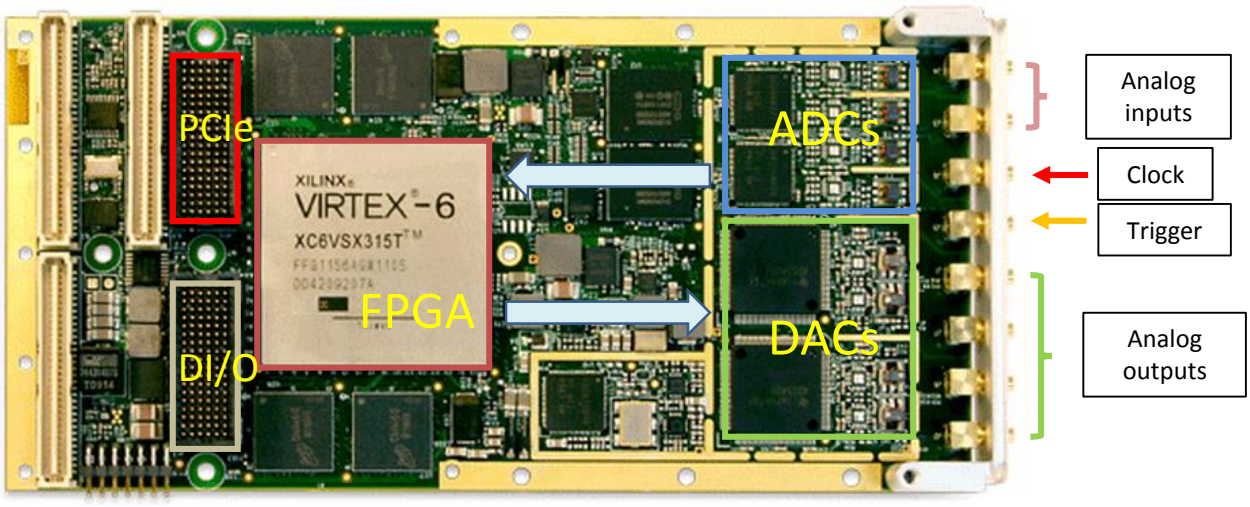

Figure 3.2: The X6-1000M FPGA board from Innovative Integration.

Despite their promising features, 500MHz-AWG and BBN's AWG were unfortunately never adopted in any experiment. The incentive to use them is low since the TEK AWGs can still handle non-feedback experiments well. The advantage of an FPGA's capability truly shines through in feedback experiments. After developing 500MHz-AWG, we were confident about meeting the new goal of uniting the "traditional setup" on one unit. In the spring of 2012, after searching the internet for the best candidate, we came across the X6-1000 board from Innovative Integration (II), an FPGA board on which we can implement both an actuator and a measurement system.

3.1 Innovative Integration's X6-1000M

The X6-1000M board is one of the most advanced and fastest FPGA boards on the market. Complete details of the board can be found in the hardware manual[Integration, 2015b]. It has a Xilinx Virtex 6 SX475T FPGA[1]. At the time of finding the X6-1000M system (early 2012), Virtex 6 was considered a mid-to-high end family in the Xilinx FPGA product line. And the SX475T FPGA especially targets high performance signal processing due to its large number of DSP cells and block RAMs in addition to high-performance CLBs. The FPGA board features two 12-bit 1 GSPS ADCs and two 1 GSPS 16-bit DACs (mechanism of operation very similar to the DAC used in 500MHz-AWG). Each of the two 1 GSPS DACs can also be configured as two 500 MSPS DACs. X6-1000M accepts a clock input for synchronization with external instruments and a trigger input that is used to synchronize all the analog channels. The board connects to a computer through Gen2 x8 PCIE interface that can provide up to 2 GB/s of transfer rate. Similar to an Alazar card[Alazar], on-board RAMs provide buffering for data transfer with a computer. However, since raw data can be processed and significantly reduced in size in real time on the FPGA, as explained in the following sections, the throughput limit rarely poses a constraint for the X6-1000M board, compared to an Alazar card. In additional to the analog channels, there are 36 digital I/O ports that are controllable by the FPGA for customized purposes.

3.1.1 Framework logic

In Chapter 2.2.3, we discussed that for the XEM5010 board, we built an almost entirely customized logic on the FPGA, save for the required USB interface. We

[1]X6-1000M can be customized with several configuration options. Early orders of X6-1000M chose the SX315 FPGA, a smaller option in the Virtex 6 family in terms of logic resources. They were all eventually replaced by the boards with SX475

implemented control interfaces in the logic ourselves for the peripheral components.

Due to the complexity of the X6-1000M board, Innovative Integration ships the FPGA with a default logic, named the Framework Logic. It is considered a skeleton logic on which an user adds custom functionalities. The logic comes with control interfaces which interact with the various hardware components, such as the ADCs, DACs, RAMs on the board.

Integrating our custom components into the Framework logic requires understanding the signal and data flow of the existing components provided by II[2].

In addition to the control interfaces for peripheral devices, two important features of the Framework logic are the "data plane" and "control plane". Both are essential for communication between the FPGA and a computer. The "data plane" is a network of logic components and connections for high throughput data transfer between the FPGA and the computer.

The "data plane" is an unsung hero. Although it is not directly involved in measurement or waveform generation, it is responsible for transferring data from the computer into all the look-up tables that make these operations for feedback possible: this task entails routing data into different *destinations*. It is also responsible for transferring data from various stages of the signal processing chain to the computer for analysis; this task entails routing data from different *sources*. To distinguish the destinations and sources, each of them has its own unique identification, an ID number. The outgoing data are assembled into packets with metadata which specifies the packet size and source IDs. For incoming data, the reverse process happens: the packets are disassembled before arriving at respective destinations. The assembly, disassembly, routing, and buffering[3] are handled by the

[2]The Framework logic manual[Integration, 2015a] provides a road map for the skeleton logic. Adding new components or modifying existing logic also entails digging through the VHDL codes of the skeleton logic

[3]Buffering refers to the queueing of data in structures like FIFOs which release data when the

"data plane".

The "control plane" maps every component in the Framework logic, whether default or custom-added, to an address, i.e., a memory address as if each component were a memory block in a shared memory. The addresses refer to specific registers associated with each component. These registers provide the computer with low-speed but on-demand control of the components. The registers that are written to, set commands or parameters for the components; the registers that are read, report statuses of the components[4]. Unlike the "data plane" which is used for large data stream sustained for at least hundreds of clock cycles, , the "control plane " is used for read and write of individual registers that occur intermittently.

Both the "data plane" and "control plane" are modified and expanded to accommodate the complex custom logic that we add.

3.2 Yngwie logic

Yngwie is the custom logic we developed on top of the Framework logic. The name Yngwie is coined by one of its creators Nissim Ofek. It stands for "**Ye**han **Ni**ssim **w**aveform **g**enerator"[5]. Since its inception, Yngwie logic has gone through many iterations, expanding its features and functionalities each time. We examine one representative version, `a006x`, which is used for the experiment in Chapter 5. All the features, operating principles and the overarching architecture shown in this chapter also apply to later versions.

The Yngwie logic architecture for quantum feedback(Fig. 3.3) consists of four main components, the sampler, the demodulator, the state estimator and the sequencer.

destination is ready to receive them

[4]The registers share an address bus and a data bus. Writing and reading of the address and data bus are done through the PCIE interface and taken care of by an IP core provided by II.

[5]The inspiration for the name comes from the Swedish heavy metal guitarist, Yngwie who is known for his superior acoustic waveform generation skills.

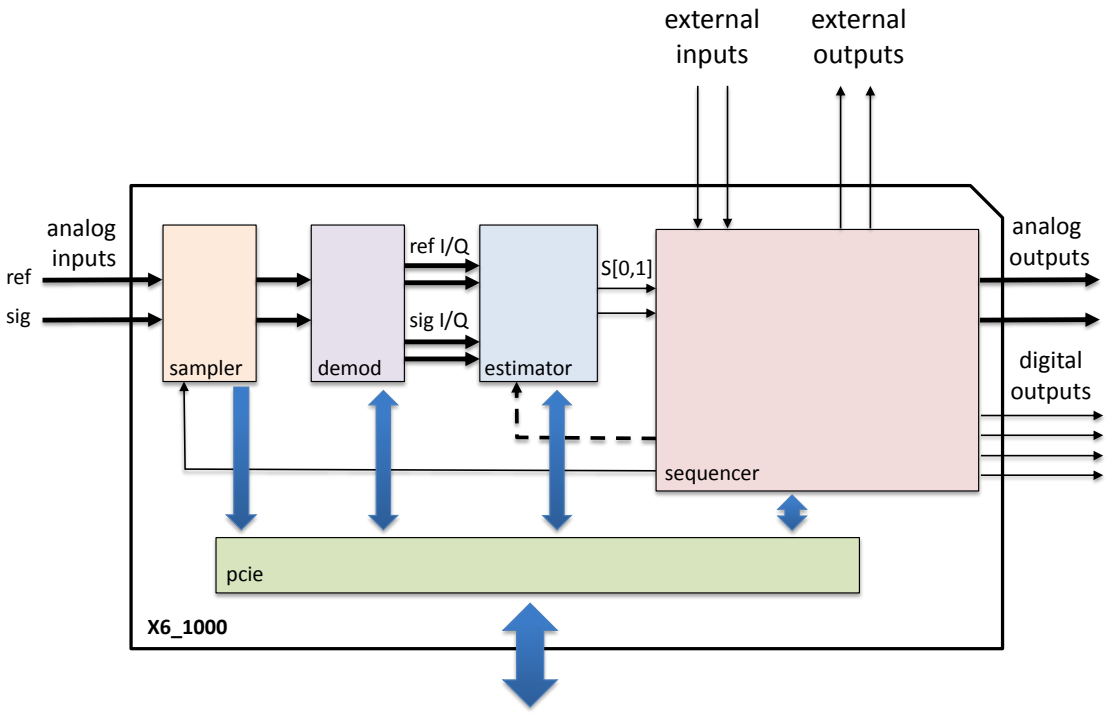

Figure 3.3: Architecture of the Yngwie logic for quantum feedback control.

3.2.1 Measurement

The sampler

The inputs to the X6-1000M board come from a heterodyne interferometric measurement setup which produces two outputs. One of them has passed through the quantum system and is called the signal; the other one, which has not interfered with the quantum system, is called the reference. The inputs from the signal and the reference channels arrive at the two ADCs, respectively and are sampled at a rate of 1 GSPS. The ADCs send the digitized data to the FPGA at the same rate. However, the 1 GHz sampling clock at the FPGA input exceeds the maximum clock frequency allowed in internal FPGA logic blocks. A serial to parallel converter right after the input comes to rescue and converts the input stream from 1 sample every 1 ns to 4 samples at once every 4 ns (synchronized to a 250 MHz clock).

An important question to answer is how sampling windows are decided on the FPGA since measurements in an experiment occur intermittently. One possibility is to turn on and off the ADCs or the FPGA inputs as dictated by the measurement sequences. However, this is not plausible because ADC calibration and synchronization with the FPGA take a long time after each restart and the time is not deterministic (typically on the order of several seconds) whereas quantum feedback experiments require the switching of sampling/measurement window to be deterministic and as often as every hundreds of nanoseconds. The solution implemented, as is often done in hardware programming in general, is to not shut down the stream source but to "label" portions of the stream as desirable or undesirable. The sampled data are paired with a 1-bit meta signal, called data_valid, which is 1 when a sample is wanted and 0 otherwise. Downstream modules, such as the state estimator processes the sample only when it is labeled as valid and

ignores it otherwise. The 1-bit meta signal received by the sampler, is a digital sequence. Programming this sequence lets us control the sampling windows. The sequence is generated by a sequencer which will be discussed in more details in Sec. 3.2.2).

The sampled raw data are directed to the demodulator. They can also be sent to the computer if this streaming option is selected. However streaming raw data to a computer causes similar problems for the X6-1000M as for the Alazar card. Therefore it is usually only done for pre-experiment calibration or debugging purposes.

The demodulator

The demodulator is the first of several custom built signal processing components that the sampled raw data go through. The readout signal and the reference signal arrive at the ADCs with 50 MHz modulation due to their mixing with a LO set at 50MHz away in the interferometer setup. To extract the in-phase (I) and quadrature (Q) components of both signals, the demodulator multiplies each of the signals with an ideal sine and cosine wave, respectively, at 50MHz. Because of the sampling rate of 1GSPS of the ADCs, one period of 50MHz contains 20 samples. As a result, the demodulator extracts the I and Q components of one period of signal by the following:

$$I_{sig,ref} = \sum_{k=1}^{20} f_k \cos\left(\frac{2\pi}{20}k\right) , \quad Q_{sig,ref} = \sum_{k=1}^{20} f_k \sin\left(\frac{2\pi}{20}k\right) \tag{3.1}$$

where f_k is the sampled raw data. On the FPGA, the values of the ideal cosine and sine waves are taken from two lookup tables that store the 20 amplitude values at those particular phase points for cosine and sine, respectively. This simple calculation may seem trivial but demonstrates the significant advantage of signal processing done on the FPGA. For traditional experiment setup that feeds raw data

directly to a computer, the demodulation is done on the computer CPU where the 20 multiplications, required for either one I or Q value, are computed sequentially. On the Yngwie logic, the demodulator utilizes multiple multipliers from DSP slices to perform as many multiplications concurrently as possible. Since 4 samples (per channel) enter the demodulator at once every 4 ns, the demodulator performs $4\times2\times2 = 16$ (number of samples $\times$ number of channels $\times$ number of quadrature components) multiplications simultaneously every 4 ns!

State estimator

In a nutshell, the state estimator further processes the sampled input and produces a measurement result that influences the sequencer. In the feedback loop, the state estimator is what connects the "detector" and the "actuator", and in the case of the X6-1000M board, the ADCs and the DACs. Before we go into the details of the state estimator, however, we need to make a small digression to address some complication.

Digression on multiple clock domains on the FPGA:

Yngwie logic can be divided into three regions (Fig. 3.4a): (1) the analog input region, consisting of the ADC interface and the demodulator. (2) the signal processing and waveforms generation region (also referred to as the middle region below), where most of the Yngwie logic is located, consisting of the state estimator and the sequencer. (3) the output region, consisting of the DAC and digital I/O interfaces. If all three regions are synchronized by exactly the same clock, this will be a perfectly functioning pipeline.

But in reality, using one clock throughout the all the regions is not possible. The input region is synchronized with the down-converted ADCs' sampling clock at 250 MHz. The output region is synchronized with the down-converted DACs' sampling clock also at 250 MHz. The ADC and DAC sampling clocks, though

nominally at the same frequency, are slightly different. The on-board PLL (phased locked loop) on the X6-1000M board generates the two clocks independently from the clock reference source to compensate for hardware and location differences between the ADC ICs and DAC ICs.

Due to the difference between the input and output clocks, the clock for the middle region needs to be faster than the other two to avoid data underflow and overflow. This clock is called the system clock and is set at 260 MHz. Underflow occurs when a read request is made when there is no more valid data to read[6]. Overflow occurs when there is valid data to read but it is ignored and consequently lost[7]. Data transfer across regions synchronized to different clocks is known as *clock domain crossing* and occurs very often in FPGA logic designs. It is typically mediated by first-in-first-out buffers (FIFOs). A sample that arrives at a FIFO *first* leaves *first*, hence the name. On the Virtex 6 FPGA, a FIFO is implemented essentially by a memory block (comprised of one or multiple block RAMs) with the write port synchronized to one clock domain and the read port synchronized to another clock domain, along with a few status outputs. In the Yngwie logic, the input region writes to a FIFO (the left FIFO in Fig. 3.4a) at a rate of 250 MHz. The signal processing and waveforms generation region reads data from the FIFO at a rate of 260 MHz. Since the read speed is higher than the write speed, overflow will never occur. Furthermore, the middle region checks the empty alert of the FIFO and stops the reading request if the FIFO becomes empty. Similarly, the middle region writes to the following FIFO (the right FIFO in Fig. 3.4a) at a rate of 260 MHz. The output region then reads data from the FIFO at rate of 250 MHz. Since now the write speed is higher than the read speed, underflow will never occur. Moreover, the middle region checks the full alert of the FIFO and stops writing if the FIFO becomes full. In actual implementation, there

[6]This happens when reading is faster than writing.
[7]This happens when reading is slower than writing.

are multiple FIFOs at each of the clock domain crossings, mediating data transfer from multiple components across two regions. For example at the clock boundary between the middle region and the output region, there are separate FIFOs responsible for data flow between the analog sequencer and the DAC interface, and between the digital sequencer and the digital I/O interface.

Realizing the proper management of data flow across the three regions was a significant milestone in the early stage of the Yngwie logic development. The effect of using a higher clock frequency for the signal processing and waveforms generation region as well as using FIFOs for clock domain crossing vs. the consequence of not doing so are demonstrated in Figure 3.4b,c.

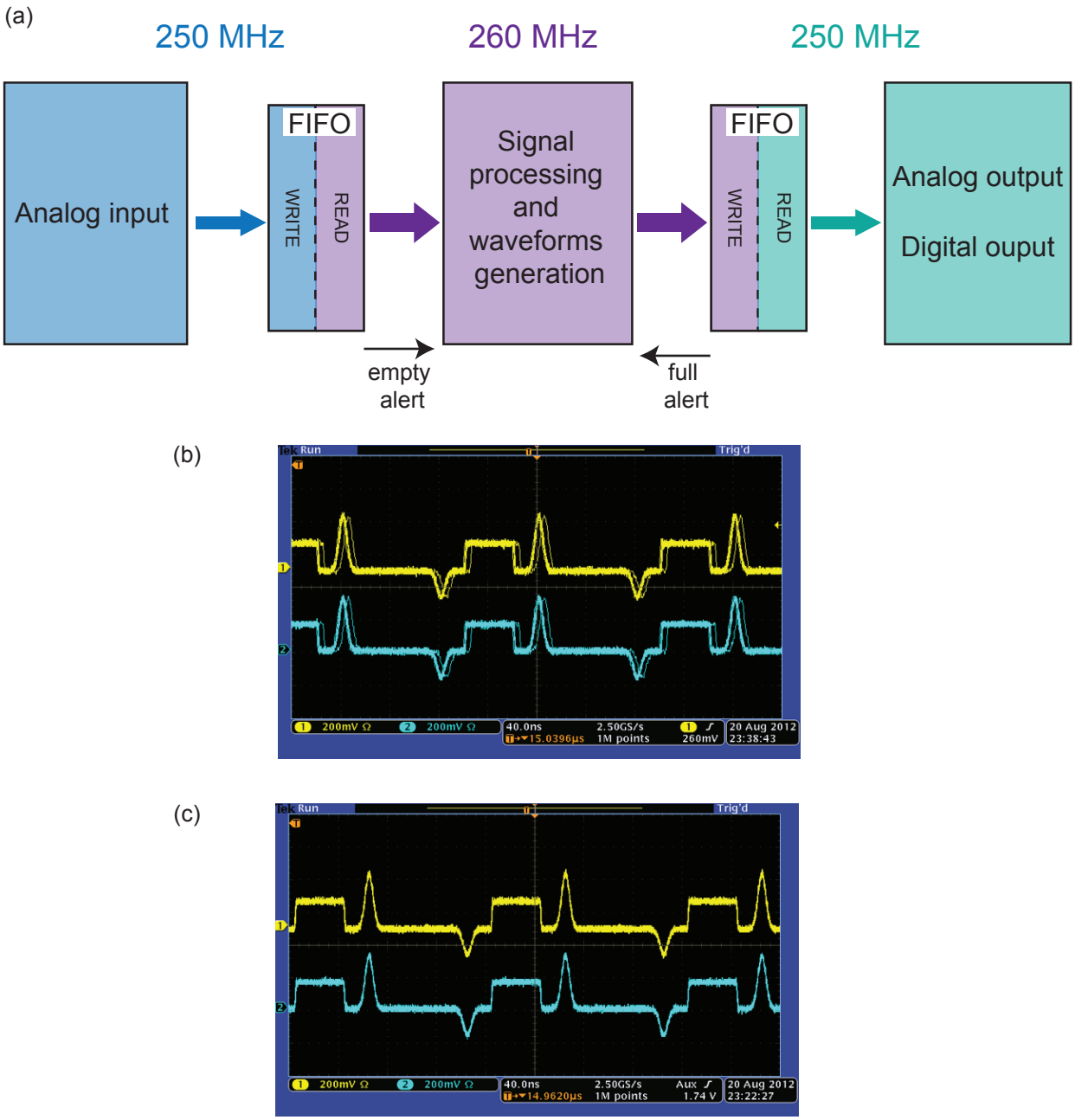

Figure 3.4: Clock domain crossing with FIFOs. (a) Yngwie logic is divided into three clock regions. (b) Both analog outputs jitter due to sample overflow/underflow when the middle region is not synchronized to a higher clock frequency and FIFOs for clock domain crossing are not used. (c) Jitter goes away when the proper changes introduced in the text are made.

The first part of the state estimator is a component that subtracts the global phase from the I and Q values. Due to phase drift of the generators, there can be a trivial phase rotation of the demodulation results in the I-Q space during an experiment over time. What we really care about are the relative I and Q values with respect to the reference input channel. The global phase subtractor produces I_{rel} and Q_{rel} by projecting the signal I-Q values to the reference I-Q values (the component is hence called **`iq_dot_product`**).

$$I_{rel} = [I_{sig}, Q_{sig}] \cdot [I_{ref}, Q_{ref}] \ , \quad Q_{rel} = [I_{sig}, Q_{sig}] \cdot [-Q_{ref}, I_{ref}] \tag{3.2}$$

I_{rel} and Q_{rel} are not normalized since division incurs significant latency on an FPGA[8]. The **`iq_dot_product`** is where the clock domain crossing occurs between the input region and the signal processing region: before entering the multipliers used for calculating the dot products shown above, the demodulated I-Q samples and the associated meta signal data_valid signals are first buffered through FIFOs. As a result, the final outputs of the dot product calculations, I_{rel} and Q_{rel}, are synchronized to the system clock of 260 MHz.

After the removal of global phase, the rest of the state estimator is responsible for summing up the stream of I_{rel} and Q_{rel} to obtain I_{sum} and Q_{sum}, which are the measurement outcomes. In a typical dispersive readout experiment, repeated measurements produce a histogram of these measurement outcomes in the I-Q space. Measurement outcomes corresponding to a particular quantum state fall into one of several Gaussian distributions, i.e., "blobs", in the histogram (Fig. 3.5).

To "estimate" the state represented by a measurement outcome in real time, thresholds are used to discriminate the measurement outcome by the region in I-Q space it belongs to. The optimal boundaries between the distributions, the ones

[8] In digital logic, there are, however, "tricks" to divide a number without actually "doing" division. Division of a number by powers of 2 can be achieved by right-shifting the number by n bits, where n is the power.

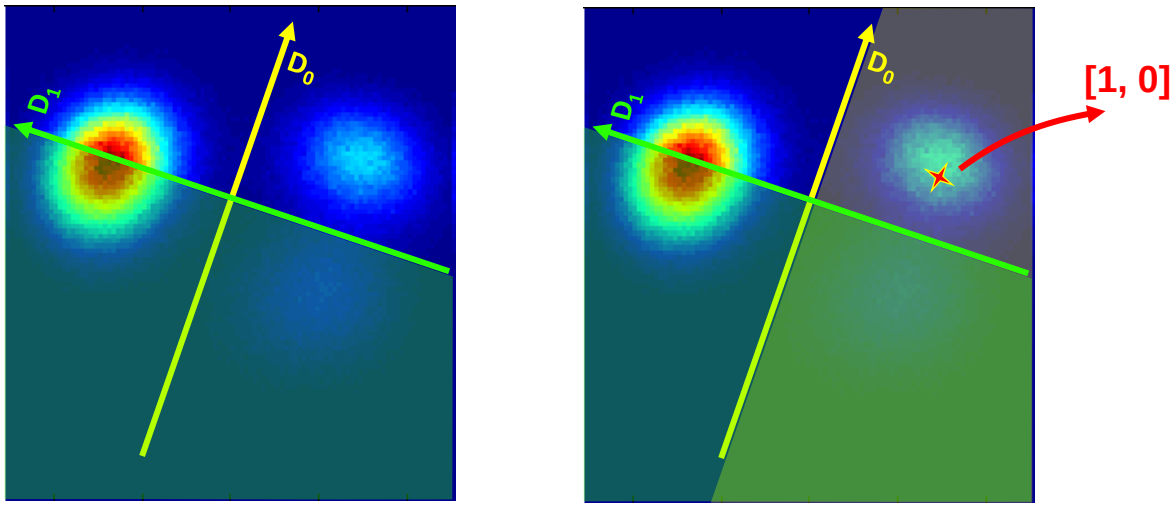

Figure 3.5: State estimation along two arbitrary directions and thresholds.

that result in minimal assignment infidelity, are pre-chosen to be the thresholds. There are two thresholds available for state estimation. They can be chosen with respect to the original I-Q axis. The result of state estimation is encoded in two bits, S_0 and S_1, based on the comparison between the measurement outcome and the thresholds. S_0 (S_1) is assigned 1 if I_{sum} (Q_{sum}) is greater than the I-threshold (Q-threshold) and 0 otherwise.

However, thresholds with respect to the original I-Q axis are not necessarily the best choice. The optimal discrimination boundaries do not have to be orthogonal to the the original I-Q axis. In such case, we can apply a linear transformation on the measurement outcomes.

$$\begin{pmatrix} \tilde{I}_{sum} \\ \tilde{Q}_{sum} \end{pmatrix} = \begin{pmatrix} \cos\alpha_0 & \sin\alpha_0 \\ \cos\alpha_1 & \sin\alpha_1 \end{pmatrix} \cdot \begin{pmatrix} I_{sum} \\ Q_{sum} \end{pmatrix} \tag{3.3}$$

Transforming the measurement outcomes is equivalent to changing the axis of the original I-Q space. Since the new axis in the transformed space do not have to orthogonal to each other, the mapping does not have to be restricted to a rotation

(if it were, then α_1 in above equation can be set equal to $\alpha_0 + \frac{\pi}{2}$). Similar to before, thresholds with respect to the axis in the new space can then be chosen for state discrimination.

The logic component for executing the state estimation procedure described above is shown in Fig. 3.6. The component consists of two parts, a main module that does the calculation and an instruction memory that supplies the parameters. The main module takes the outputs of the **`iq_dot_product`** and a signal specifying an address for the state estimation instruction memory. This signal comes from the master sequencer, which we will describe later. The main module selects the appropriate instruction word from the instruction memory by reading the content at the location selected by the instruction address.

Every measurement pulse of certain duration (i.e., every sampling window) starts a new round of calculation in the main module of the state estimator. The main module accumulates the inputs I_{rel} and Q_{rel}, respectively, whenever they are accompanied by an asserted data_valid signal. The sums are then transformed to $\tilde{I}_{sum}$ and $\tilde{Q}_{sum}$ according to equation 3.3, where the matrix elements of the transformation are contained in the instruction word (the matrix can be identity in the case of no transformation). The instruction word also provides the the values for the two thresholds for state discrimination. The field "est. length" in the instruction gives the estimation length, which dictates how many demodulation periods the main module needs to sum over. This is typically set at its maximum possible value, $(\text{measurement duration})/(20ns)$. The state estimator outputs three signals, $\tilde{I}_{sum}$, $\tilde{Q}_{sum}$, and a two-bit signal encoding S_0 and S_1. They are directed to the next component, the sequencer. They can also be streamed to the computer for post-analysis.

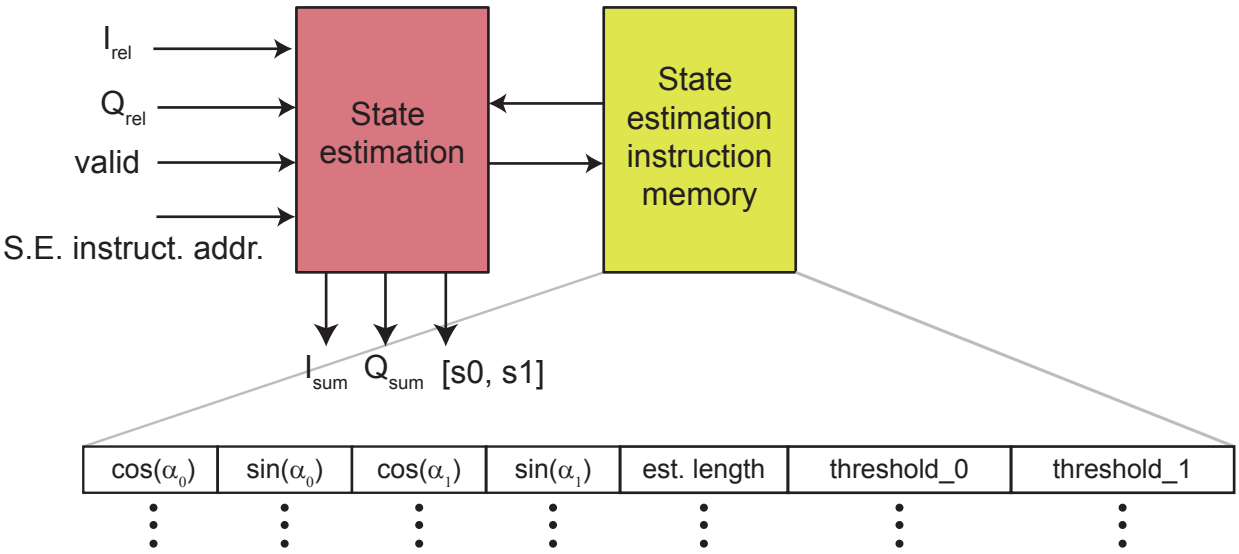

Figure 3.6: Architecture of the state estimator.

3.2.2 Sequencer

The results of the state estimator go into the Sequencer[9] (Fig. 3.7), a complex component responsible for generating analog and digital sequences conditioned on the results. Within the Sequencer, there are several constituent sequencer components: analog, digital and master. A sequencer is named so because it executes a sequence of instructions in some prescribed order. Its principle of operation is outlined in Chapter. 2.2.2: it consists of at least a FSM-based control unit that orchestrates the reading of the instruction and an instruction memory storing the sequence of instructions.

[9]From now on, to avoid confusion, we use *Sequencer* to mean the entire sequencing component and *sequencer* to mean an constituent sequencer within the *Sequencer* component.

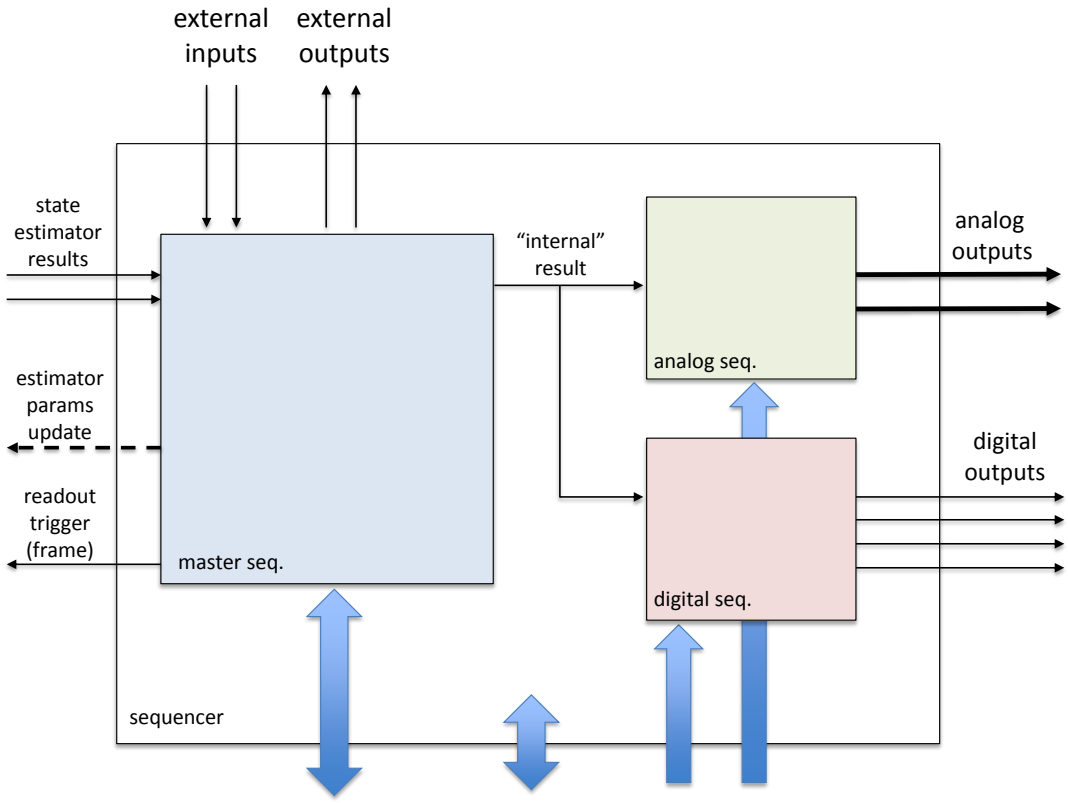

Figure 3.7: Top level architecture of the Yngwie Sequencer consisting of the master sequencer, analog sequencer and the digital sequencer.

branch type	**meaning**
(11) *Goto*	go to **addr0**
(00) *Branch*	if **internal result** = *True* go to **addr0**, else go to **addr1**
(01) *Goto subroutine*	go to **addr0** and push **addr1** onto a stack
(10) *Conditional return*	if **internal result** = *True* pop the top address from the stack, else go to **addr0**

Table 3.1:

The behavior shared by all the sequencers relates to the flow of instructions. There is a common mechanism on how a sequencer continues from one instruction to the next. Each instruction has its unique address in the instruction memory and contains at least the following fields: two instruction-address fields, **addr0** and **addr1** and a field which we call **branch type**. When we finish executing an instruction, we will move to the next depending on this **branch type**, the two addresses specified in the instruction and a *global state* of the FPGA which we call **internal result**. At any moment, **internal result** is either true or false and may change along the execution of an instruction (we will go in details about this in Sec. **??**). There are four kinds of instruction-to-instruction transitions specified by four different values of **branch type**:

The **branch type** *Goto* is the most basic one, used for non-conditional sequences and is also implicitly how instruction transitions in the FPGA-based AWG described in Chapter 2.2.2. **branch type** *Branch* is self-explanatory. **branch type** *Goto subroutine* and *Conditional return* work in tandem. Later in Sec. 3.2.2, we will discuss them in more details.

The analog sequencer generates analog waveform data to the DAC interface. Besides the more sophisticated branching options, the analog sequencer is still based on the architecture described in Chapter 2.2.2. The digital sequencer generates digital waveforms data to the digital I/O interface, which are outputted as digital pulses used for controlling external instruments or devices. Last but

not least, the master sequencer is responsible for generating the digital sequence that determines sampling windows introduced in Sec. 3.2.1. However, the master sequencer does much more than just outputting a digital sequence and is, as its name suggests, the master of all the sequencers. It is in charge of deciding the *global state* that dictates how the other sequencers transition between instructions. Moreover, the digital sequencer is a simplified version of the master sequencer. We shall therefore focus on the master sequencer in the rest of this chapter.

Master sequencer

Typical of the hierarchical nature of FPGA hardware programming, the master sequencer itself is a complex component comprised of several submodules (Fig. 3.8). The component "sequence control" coordinates the reading of the master sequence instruction memory and extracts one instruction at a time. The master sequencer updates the global state, **`internal result`**, according to this instruction.

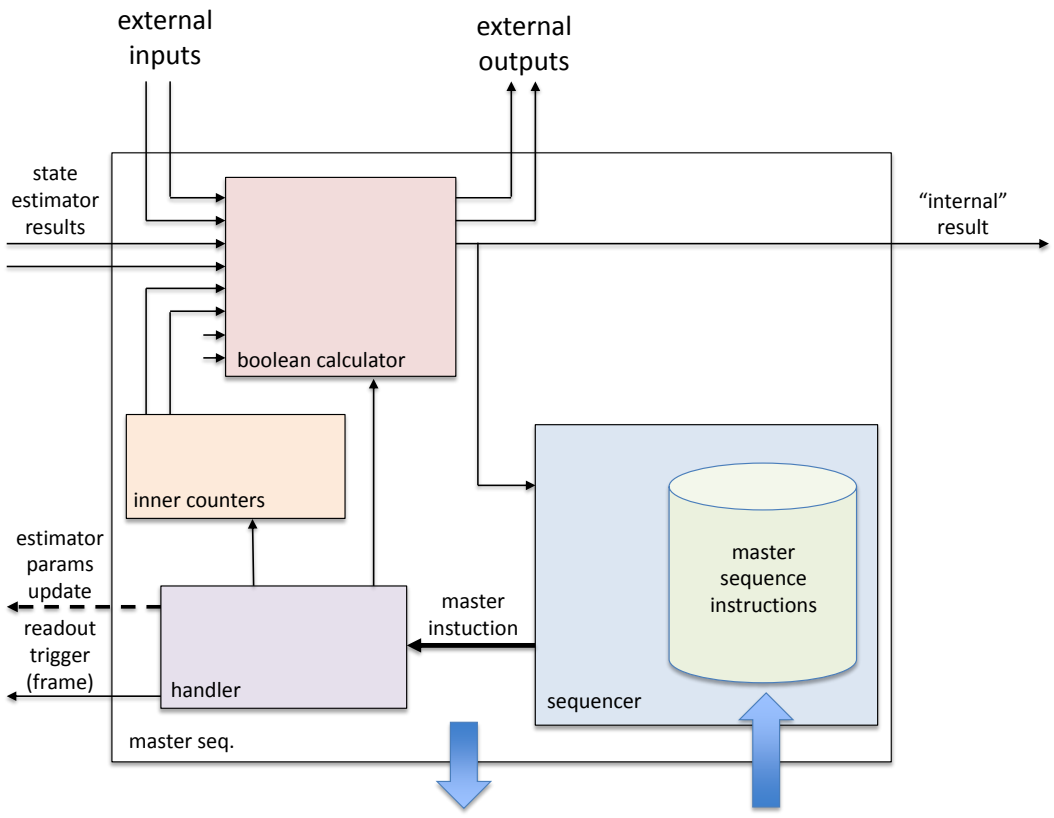

Figure 3.8: Architecture of the master sequencer.

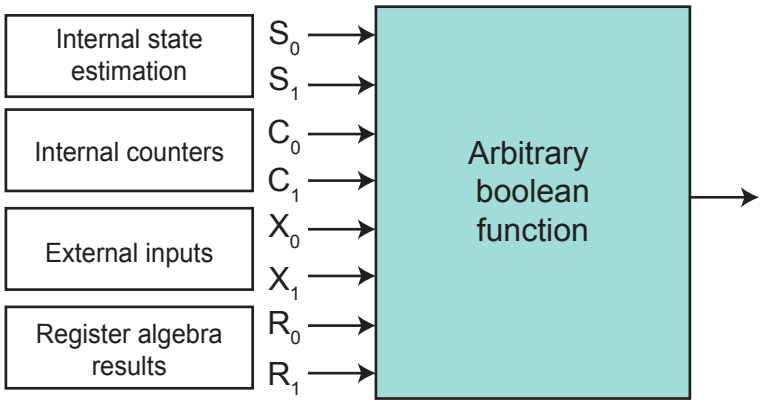

Figure 3.9: The *global state*, **`internal result`** is a boolean function of the states of eight 1-bit registers updated by the signals/processes shown in the boxes.

Recall that at any moment, **`internal result`** encodes a boolean state monitored by all the sequencers in the FPGA. At the critical moment, when the execution of an instruction is near completion, the choice of the next instruction, i.e., which instruction to *branch* to, depends on this boolean state. Before this critical moment, the master sequencer can update **`internal result`**. Manipulating the way it is updated is how we control the flow of instructions, which determines an experimental protocol.

There are several registers[10] in the master sequencer whose values affect how **`internal result`** is updated (Fig. 3.9). They are the following, identified in bold:

We have two counters which can be initialized to certain values we choose. There are two boolean registers corresponding to these two counters, $\boldsymbol{C_0}$ and $\boldsymbol{C_1}$. Each of theirs value is true when the corresponding counter has been decremented to 0, and false otherwsie. As shown later in Sec. 3.2.2, we can use the counters to implement loops.

There are two registers that hold the two boolean outputs from the state esti-

[10]Not to be confused with the registers of the "control plane" discussed in Sec. 3.1.1

mator, $\boldsymbol{S_0}$ and $\boldsymbol{S_1}$, which we have described earlier.

There are two boolean inputs that come from external sources, $\boldsymbol{X_0}$ and $\boldsymbol{X_1}$. Their origin may be another FPGA or some other instrument.

Last, we have four general-purpose 16-bit registers with which we can perform various basic arithmetic manipulations[11]. There are two boolean registers, $\boldsymbol{R_0}$ and $\boldsymbol{R_1}$, that correspond to comparison results between the values of two registers selected. We can manipulate these general-purpose registers to store values shared between instructions, like variables in a computer program.

The boolean states stored in these eight registers are called the *bare states*. These *bare states* can change during the execution of an instruction as their respective sources update them. For example, $\boldsymbol{S_0}$ and $\boldsymbol{S_1}$ change after a new measurement outcome is calculated and discriminated by the thresholds in the state estimator; $\boldsymbol{X_0}$ changes when the external source, e.g., another FPGA switches its value.

The global state,**`internal result`** is a boolean function of the *bare states*. A boolean function is often represented by a truth table (Fig. 3.10). In a truth table, the right-most column gives the result of a boolean algebraic expression as a function of the columns of inputs on the left. The size of the function, as determined by as the number of bits in the right-most column, is $2^{(\text{number of inputs})}$. The boolean function for computing **`internal result`** is called **`internal function`** in the master sequencer. Implementing a boolean function is achieved by storing all the bits of the right-most column of its truth table in a register. For a given set of inputs, we find the result of the function simply by looking up the bit at the location specified by their (the inputs) value (converted to decimal). This implementation allows the boolean function to be as simple or as complex as we desire without adding demand on the FPGA logic resources. Figure 3.10 gives an example of a truth table of an **`internal function`** that sets **`internal result`** true

[11] the complexity of the arithmetic manipulations available has been significantly expanded since the logic version described in the thesis

S_0	S_1	$S_0 \cdot \overline{S_1}$
0	0	0
0	1	0
1	0	1
1	1	0

Figure 3.10: An example of implementing a boolean function by a truth table to identify a qubit state.

when a qubit is in the excited state.

In Appendix A, we give details on how the master sequencer parses and implements an instruction from the its instruction memory.

Building sequences

The logic infrastructure that supports a range of branching options for a sequence of instructions plus the flexibility and programability of the branching conditions is what distinguishes the X6-1000M feedback controller from the commercial AWGs and controller systems used by other groups. They allow us to use control flow structures commonly seen in software programming to design sophisticated sequences that can meet the demands of complicated experiments. The implementation of control flow structures in the Yngwie sequences parallels how it is done in computer software programming at the very low level, i.e., in assembly language. This is not surprising since the Yngwie sequencers' architecture is based on microprocessor architecture which computer CPU is also based on. In this section, we attempt to put several features introduced in the last section into practice.

Loop

A very common control flow structure in programming is a loop. The **`branch type`** *Branch* together with the counters in the master sequencer allow us to implement loops in a sequence. Figure 3.11 gives a pseudocode description of the instructions for such an implementation. As an example of application, an experimental protocol may require repeating a set of tasks a certain number of times before continuing to the next step.

An initial instruction initializes one of the counters (represented by M in the Figure) to a particular custom value, which is stored in a field in the instruction (see Appendix A). This is the value that the counter counts down from. The instruction also sets **`internal function`** to be dependent solely on the zero status of the counter. The instructions that follow can be any tasks that need to be executed and repeated. After those instructions, the following instruction decrements the counter. It also monitors the output of **`internal function`**, **`internal result`** which is 1 when the counter has reached zero and 0 otherwise. Based on **`internal result`**, the sequence either reverts back to the starting instruction of the tasks (note: not the very initial instruction) and thus repeats them or continue to a new instruction.

Loop is used very often by the FPGA controller. The tasks iterated inside a loop can be as simple as a few sequential instructions or complex such that they involve conditional branching themselves. The MB stabilization experiment in Chapter 5 uses this feature to keep track of number of correction steps.

It is worth mentioning that both counters provided by the master sequencer can be used at the same time to implement a nested double loop.

Subroutine

In software programming, if a task is called very often, a function or subroutine is written for that task to avoid code duplication. Every time the task needs to be performed, the subroutine is called. The **`branch type`** *Goto subroutine* and

```
INSTRUCTION 1: M = 15
      INSTRUCTION 2: DO XXXX, GOTO INSTRUCTION 3
      INSTRUCTION 3: DO XXXX, GOTO INSTRUCTION 4
      ...
      INSTRUCTION 7: Decrement M,
                  If M = 0 GOTO INSTRUCTION 8
                           Else GOTO INSTRUCTION 2
INSTRUCTION 8: DO XXX, GOTO INSTRUCTION k
```

Figure 3.11: Implementing a loop using counters.

Conditional return achieve the same goal for the sequencer. Figure 3.12 gives a pseudocode description of the instructions for such an implementation.

In a cQED experiment, there are often sequences of pulses that need to played many times for just one iteration of the experiment. An example is the qubit state tomography pulses. For two-qubit state tomography, there are 16 sets of two-qubit pulses[Filipp et al., 2009]. Each of them needs to be repeated many times since the experimental sequences that precede them are typically alternated by some parameters. For example, in the MB stabilization experiment described in Chapter 5, we vary the number of correction steps before tomography. That number changes from 1 to 20, totaling 20 variations[12]. Taking into account both the stabilization and state tomography sequences, we have $20 \times 16 = 320$ variations of qubit pulse sequences for the complete protocol. Storing all of them in the sequence instruction memories is not efficient since many of the instructions would be duplicate.

Alternatively, we can treat each of the 16 state tomography pulse sequences and each of the 20 stabilization sequences as a unique subroutine. We only encode the instructions for the subroutine once in the sequence memory. Instead

[12]Each of the variations, i.e., a stabilization sequence with n correction steps is implemented as a loop with a counter initialized to n

of storing instructions for 320 different qubit pulse sequences, we now only have to store the instructions for $16 + 20 = 36$ subroutines. Whenever the experiment sequence needs to invoke a subroutine, the **`branch type`** *Goto subroutine* allows the sequence to jump to the address for the first instruction of that subroutine and execute its instructions. Meanwhile, it stores the address of the instruction that we should return to in a stack, which is a first-in-last-out data structure (bottom of Figure 3.12). The **`branch type`** *Conditional return* lets the sequence to return to the main thread only when certain conditions are met. We use **`internal function`** in the master sequencer to set the boolean conditions for the competition of the subroutine. When the execution of the subroutine is complete as indicated by **`internal result`** being true, we go to the instruction pointed to by the address stored on the top of the stack and remove the address from the stack (known as "pop"). The stack enables nested subroutines: we can go to another subroutine within a subroutine.

Implementing FSM on the sequencer

In MB quantum feedback, typically there are a finite number of states that we expect the target quantum system to be in. Consequently, the FPGA that controls the quantum system also needs to be in a finite number of states that correspond to the quantum states. As the FPGA monitors and steers the quantum system, it needs to be able to transition among these states, while the transitions are conditioned on input from the quantum system. This process calls for realization of a FSM. The sequencers of the Yngwie logic equip us with the capability to implement a FSM sequence. The MB feedback protocols discussed in Chapter 5 demonstrate a couple of advanced FSM sequences in action.

Here, using a common experiment, cavity spectroscopy, as an example, we give a tutorial of implementing of a FSM sequence that utilizes several features discussed so far (Fig. 3.13). During the spectroscopy experiment, a microwave

```
INSTRUCTION 1: DO XXXX, GOTOSUB INSTRUCTION 8,
RETURNTO INSTRUCTION 2
INSTRUCTION 2: DO XXXX, GOTO INSTRUCTION 5
...
INSTRUCTION 8: DO XXXX, GOTO INSTRUCTION 9
INSTRUCTION 9: DO XXXX,
               If condition
          GOTO RETURNTO
          Else
               GOTO 9
```

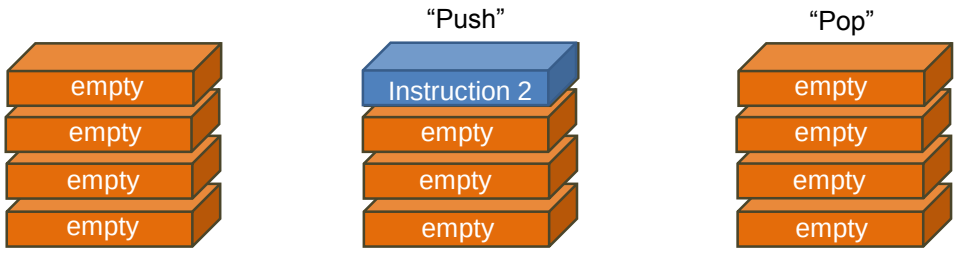

Figure 3.12: Implementing a subroutine call

generator sweeps the frequency of the tone to the cavity over a range by incremental steps. At each frequency step, the FPGA takes thousands of measurements of the cavity output and streams the results ($\tilde{I}_{sum}$ and $\tilde{Q}_{sum}$) to a computer. The FSM sequence constructed for this experiment consists of three states:

State "initialize": this state is made of just the very first instruction of the sequence, during which a counter (represented by N in the figure) is initialized to a value that is set to the desired number of averages for the measurements (i.e., number of repetitions per frequency step).

State "measure": the sequence transitions from state "initialize" unconditionally into this state (by **`branch type`** *Goto*). First, the counter is decremented by one. Then, a digital pulse to modulate the cavity drive is generated by one of the digital sequencer (A qubit π pulse may also be applied prior to the digital pulse, such as shown in Figure. 3.13, if we are interested in χ measurements) and the

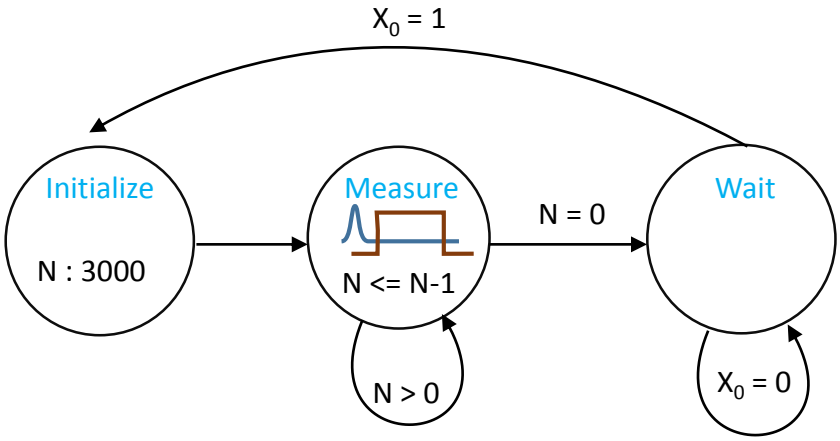

Figure 3.13: Constructing a FSM sequence for a spectroscopy experiment.

returning cavity response is demodulated and processed by the state estimator. The resulting $\tilde{I}_{sum}$ and $\tilde{Q}_{sum}$ are streamed to the computer. In the final instruction of this state, **`internal function`** is programmed to depend solely on the counter. Its output, **`internal result`**, instructs the sequencer to branch to the beginning instruction of the current state again if the counter has not reached zero. Otherwise, it branches into the final state.

State "wait": In this state, the FPGA does nothing[13]. It waits for a trigger from an external function generator that is controlled by the computer. When the computer has received all N pairs of $\tilde{I}_{sum}$ and $\tilde{Q}_{sum}$ for one frequency step, it commands the microwave generator to step up the frequency and the function generator to release the trigger. This trigger enters the logic, as $\boldsymbol{X_0}$, one of the external inputs accepted by **`internal function`**. In the final instruction of this state, **`internal result`** depends solely on $\boldsymbol{X_0}$. When it is asserted, the sequencer branches back to state "initialize" , otherwise it branches to the beginning instruction of the current state and continues to wait for the trigger.

[13]Doing nothing means that all the sequencers, analog and digital, are outputting constant zero amplitude pulse.

CHAPTER 4

Qubit cooling

4.1 Introduction

The ability to prepare a qubit in a predetermined state, also known as qubit reset or qubit initialization, is one of Divincenzo's five criteria [DiVincenzo, 2000] for building a quantum computer. Its importantance is intuitive: many algorithms, quantum (or even classical), require knowing the initiate state of inputs. Knowing the initial state, such as the initial state of the qubits, allows one to track the trajectory of a protocol. If a qubit or a set of qubits can be initialized or "cooled" to their ground states, deterministically applying arbitrary gates on them can then prepare them in an arbitrary initial state. This is also the reason why qubit reset has typically been approached as a cooling problem.

For superconducting qubits in a cQED setup, the simplest approach is just passive waiting for qubits to reach thermal equilibrium with a cold bath. This approach is problematic because waiting can take a long time as qubit coherence time now can exceed a hundred microsecond and the "cold" bath that the qubit is in is actually not cold enough.

In cQED experiments, we have often observed qubit thermal population in excess of 10%. For ensemble experiments where results of interest are averaged

over all repetition runs, the qubits that are in the "wrong" states in the beginning will report a different readout value (e.g., a different voltage reading) and bias the final result. This results in reduction of contrast. This problem may not be a serious issue for system parameter characterization experiments, such as finding $T1$ and $T2$ of the qubits. But for experiments such as state tomography, readout voltage calibration are required to account for the thermal population. For many cQED experiments, however, such as single-shot experiments, the non-negligible thermal population produce undesirable results that are harder to mitigate.

To combat this, post-selection has been the most common method to purify the initial state of a qubit to ground state. A high fidelity measurement chain enabled by a nearly quantum limited amplifier can distinguish the state of a qubit on a single measurement. An initial stage of measurement is therefore inserted before an experiment protocol to herald experiment runs in which the qubit starts in ground state, by post selection. This method can purify ground state with very high fidelity[Johnson et al., 2012; Ristè et al., 2012b]. A drawback of this method is that experimental success probability is sacrificed since a significant percentage of data are thrown away.

Using feedback to cool a qubit is the alternative approach. Both reservoir-engineering based DD feedback and MB active feedback approaches have been developed. Kurtis demonstrated the double drive reset of population (DDROP) protocol, a DD method for qubit reset[Geerlings et al., 2013]. Riste et al showed the first active purification of a qubit to ground state using MB feedback based on a FPGA[Ristè et al., 2012a]. At Yale, we implemented two similar MB feedback protocols to cool a qubit, dubbed the "baby Maxwell's Demon" and "mature Maxwell's Demon" with the all-in-one FPGA controller.

4.2 Qubit cooling by MB feedback

4.2.1 Baby Maxwell's demon

The "baby Maxwell's Demon" is the first feedback protocol implemented on the FPGA controller. It removes entropy from a qubit by purifying it to the ground state and is named such for its simplicity. It consists of several steps (Fig. 4.1). The first step is for erasing history. We applies a projective measurement (M_0) on the qubit which result in either the ground or excited state according to some thermal distribution. A $\frac{\pi}{2}$ pulse rotates the qubit to the equator of the Bloch sphere where it is in an equal superposition of the ground and excite state. At this point, information about qubit's final state in the previous round is completely erased. This step prepares the qubit in a maximum entropic state, a good test for the "baby demon". In the second step, another projective measurement (M_1) projects the qubit to either $|g\rangle$ or $|e\rangle$ with equal probability. The state estimator in the FPGA determines qubit state by applying thresholds on the measurement outcome. In the third step, a feedback decision is applied. If the qubit was determined to be $|g\rangle$, we do nothing otherwise we apply a π pulse. In the final step, a strong projective measurement (M_2) is applied again to record the state. This sequence of steps is then repeated many times. In Figure 4.2, we show the measurement outcomes of the sequence in terms of IQ histograms.

For one experiment of the "baby Maxwell's demon", at the end of M_1, $|g\rangle$ and $|e\rangle$ populations are estimated to be 46.2% and 47.2% (Fig. 4.3). The "baby Maxwell's demon" lowers the entropy by purifying the ground state from 46.2% to 88.7%. The third and fourth step (M_1 followed by the conditional π pulse) in the protocol comprise the feedback stage (Fig. 4.4). Due to imperfection in the pulse and measurement infidelity, there is still quite a significant remainder population in the excited state.

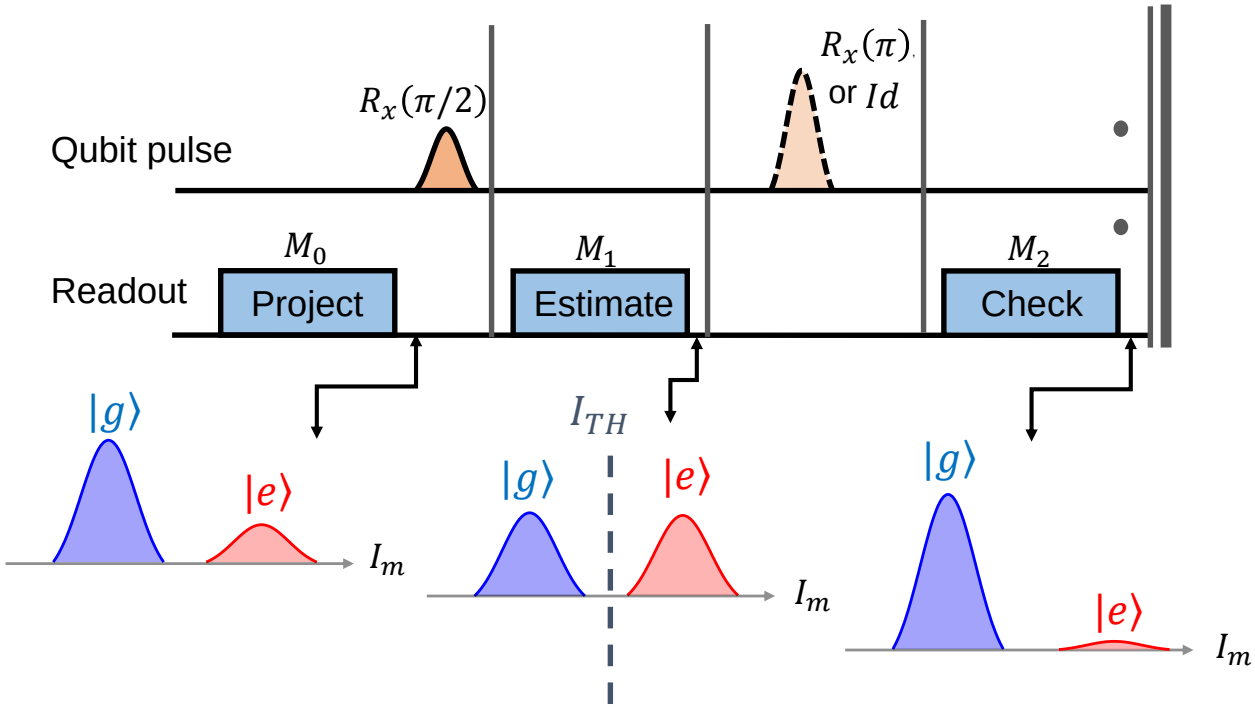

Figure 4.1: Pulse sequence of the baby Maxwell's demon.

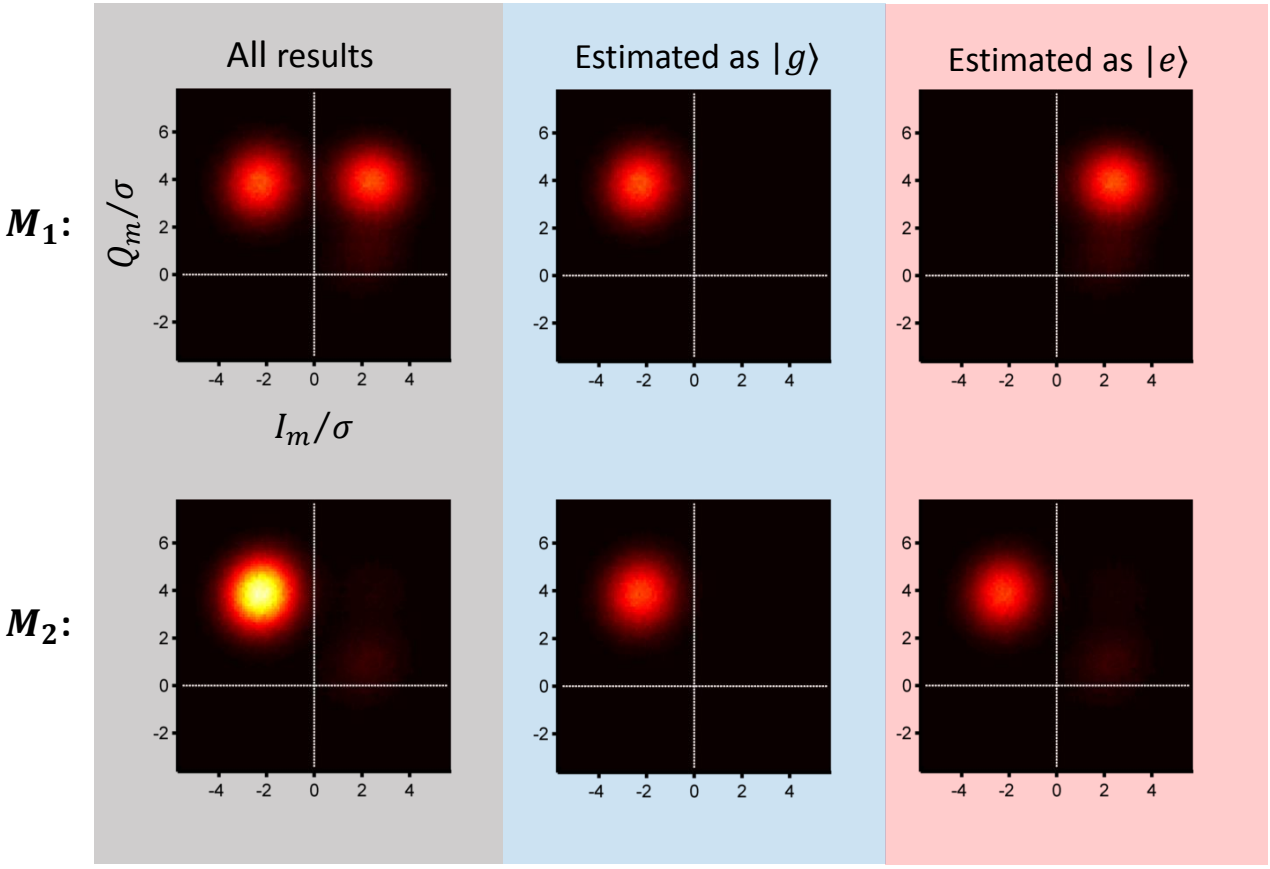

Figure 4.2: Measurement outcomes of the "baby Maxwell's demon" protocol. The top (bottom) row shows the measurement outcomes of M_1 (M_2). After M_1, as we expect, we have an even distribution of ground and excited states (top left). In addition, we separately show the histograms of measurement outcomes that were estimated to be ground states (middle) and excited states (right) in M_1. Then after M_2, which is after the conditional pulse based on the estimation decision, we have a histogram that shows most of the population in the ground state (bottom left). In the bottom middle histogram we see that all the qubits estimated to be in the ground state after M_1 are still in the ground state. And in the bottom right histogram, we see that almost all of the qubits estimated to be in the excited states get flipped to ground states.

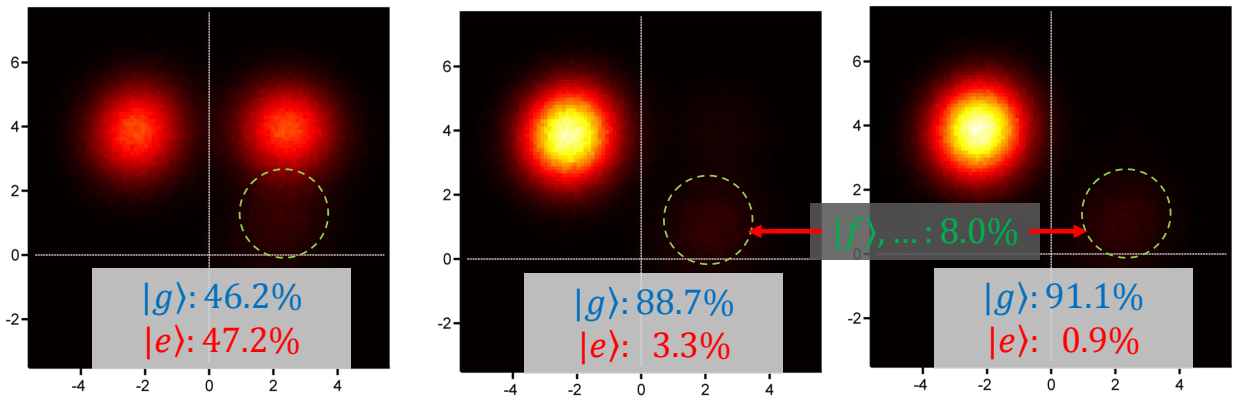

Figure 4.3: Measurement outcomes of repeating a stage of the baby Maxwell's demon.

Repeating the feedback stage one more time (Fig. 4.4) can purify the ground state further to 91.1%. However, we note that throughout the steps of the sequence, the populations estimated in $|g\rangle$ and $|e\rangle$ from the I-Q histograms do not add to 100%. This is due to the population present in the higher excited states, such as as $|f\rangle$ and $|h\rangle$, evidenced by the distribution of measurement outcomes below the $|e\rangle$ outcomes in the histogram. It appears that repeating the simple feedback stage once does not affect the higher states population. In principle, we can deplete the higher states by repeating the step many more times; in each time we move the population in the excited state which are then mostly contributed by the natural relaxation from the higher states. This is a slow process. Fortunately, taking advantage of the flexibility of the FPGA controller, we can enhance the capability of the "baby Maxwell's demon" and actively deplete population in the higher excited states.

4.2.2 Mature Maxwell's demon

We dub this more advanced protocol, which actively removes population from $|f\rangle$ and $|h\rangle$, the "mature Maxwell's demon". Figure 4.5 shows the pulse sequence for

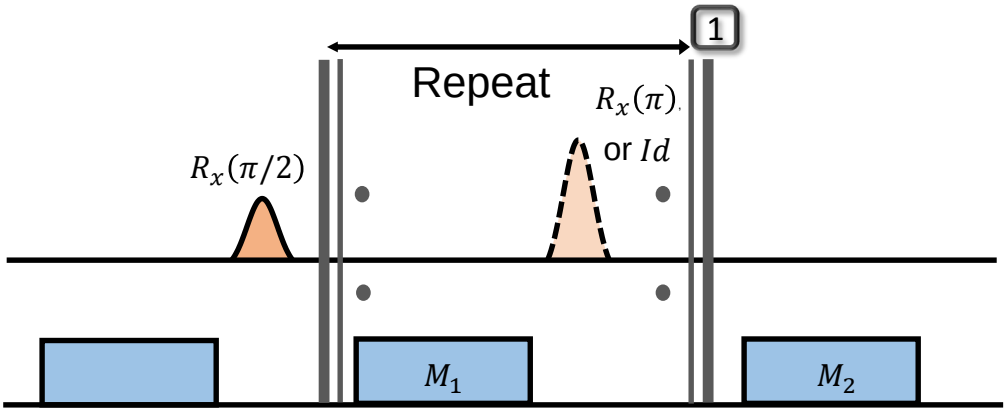

Figure 4.4: Repeating a stage of the baby Maxwell's demon

the mature Maxwell's demon and Figure 4.6 shows the corresponding effect on the qubit state population distribution for one repetition of the sequence.

The mature Maxwell's demon sequence begins with the first three steps of the baby Maxwell's demon. At the end of the baby demon sequence (with the feedback stage repeated twice), most of the population are in the ground state, very little in the excited state and a sizable remaining in the higher states, i.e., $|f\rangle$ and $|h\rangle$. Following these steps of baby Maxwell's demon, a projective measurement (M_k) is then applied to identify the qubit state. If the qubit is in $|g\rangle$, we do nothing; otherwise we apply a train of π pulses. The first π pulse is on the $|e\rangle \rightarrow |g\rangle$ transition that moves population from $|e\rangle$ to $|g\rangle$; the second π pulse is on the $|f\rangle \rightarrow |e\rangle$ transition that moves population from $|f\rangle$ to $|e\rangle$; the last π pulse is on the $|h\rangle \rightarrow |f\rangle$ transition that moves population from $|h\rangle$ to $|f\rangle$. Then we measure again (M_{k+1}); we do nothing if the qubit is in the ground state otherwise we apply a π pulse on the on the $|e\rangle \rightarrow |g\rangle$ transition to empty the population in $|e\rangle$. This sequence of steps to shuttle population from higher states to the ground state comprise a stage in the mature Maxwell's demon and can be repeated however many times as needed by the experiment.

Now we can show the mature Maxwell's demon in its full glory as the feed-

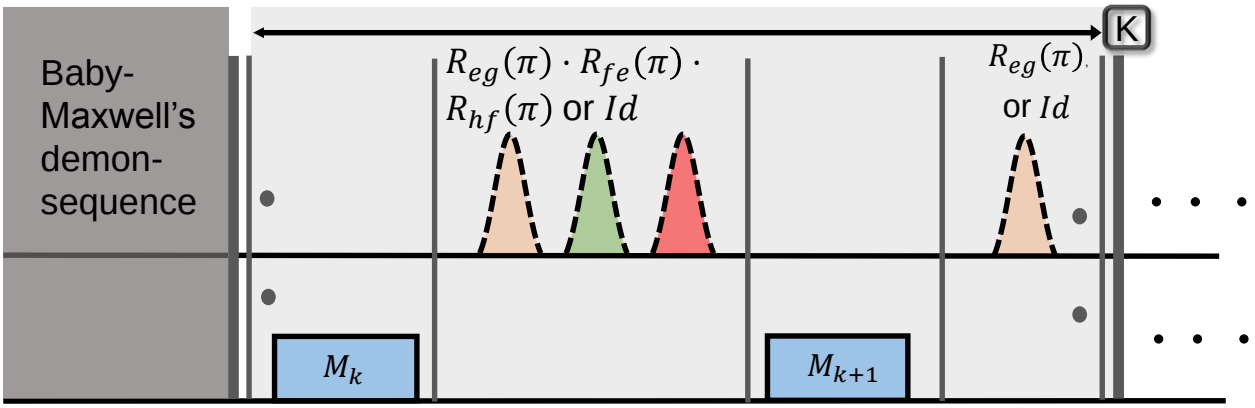

Figure 4.5: Sequence of the mature Maxwell's demon.

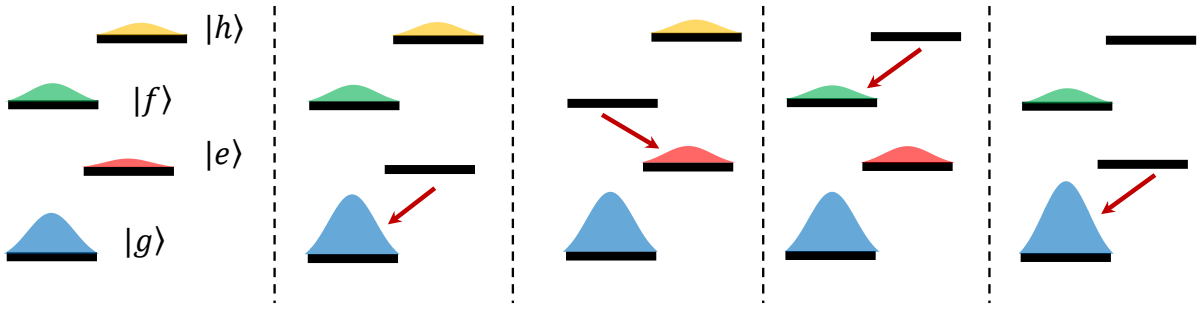

Figure 4.6: Qubit state population during mature Maxwell's demon.

back stage is repeated. In Fig 4.7, we present select histograms of measurement outcomes from M_0 through M_{18} as the purification progresses[1]

$\boldsymbol{M_0}$: the qubit is at its initial state, with the population thermally distributed.

$\boldsymbol{M_1}$: after the history erasure by the $\frac{\pi}{2}$ pulse, the qubit is in a very high-entropy/mixed state.

$\boldsymbol{M_2}$: Most of the $|e\rangle$ population is depleted.

$\boldsymbol{M_3}$: Very little $|e\rangle$ population left after repeating the conditional $e \rightarrow g\ \pi$ pulse.

$\boldsymbol{M_4}$: $|f\rangle$ state is starting to be depleted.

$\boldsymbol{M_5}$: $|h\rangle$ state population is transferred to $|e\rangle$.

$\boldsymbol{M_6}$: $|e\rangle$ state is emptied again.

$\boldsymbol{M_8 ... M_{18}}$: Repeat the last two steps over and over again.

In Figure 4.8, we plot the qubit population of each state as a function of the feedback stage. Furthermore, we also show the corresponding Shannon's informational entropy (defined in Sec. 1.3) as a function of the feedback stage. After the final stage, we have a ground state purity of 99.3% and lowered the entropy from 1.2 to 0.007. Most of the entropy depletion occurs within the first 10 stages. The rest of the stages are mainly passively depleting states even higher than $|h\rangle$. In principle, if we want to, we can also actively deplete state $|i\rangle$. and achieve similar result with few stages.

The purity of the ground state attained in Maxwell's demon cooling experiment is limited mainly by measurement errors, T_1 of the qubit and latency of the feedback. The measurement errors can be mitigated by a technique which we

[1]The actual order of π pulses applied and the resulted population shuffling between the first four energy states are slightly different from what is presented in Fig. 4.5 and Fig. 4.6. In the experiment that produced the histograms in Fig 4.7, $f \rightarrow e\ \pi$ pulse is the first qubit pulse applied after the baby demon protocol to swap the population in $|e\rangle$ (very little left) with $|f\rangle$ followed by $e \rightarrow g\ \pi$ pulse. Following these two pulses is a projective measurement. If the qubit is not in $|g\rangle$, then $h \rightarrow f$, $f \rightarrow e$ and $e \rightarrow g\ \pi$ pulses are applied in the specified order to move the remaining population from $|h\rangle$ to $|g\rangle$. The order in Fig. 4.5 and Fig. 4.6 is shown, however, for clarity and simplicity of presentation. The slight variations of the π pulse ordering work as well and achieve the same result at the end.

call "dynamic thresholding". During the Mature Maxwell's demon protocol, the threshold used to distinguish the ground state from the other states are adjusted as the feedback stages progress. Right after the qubit is put in the maximumly entropic state, where there is very similar likelihood of being $|g\rangle$ and $|\bar{g}\rangle$, the threshold is chosen such that $\epsilon_{|g\rangle||\bar{g}\rangle}$, defined as the error of measuring $|g\rangle$ when the qubit is in $|\bar{g}\rangle$, is comparable to $\epsilon_{|\bar{g}\rangle||g\rangle}$, the error of measuring $|\bar{g}\rangle$ when the qubit is in $|g\rangle$[2]. As the ground state population increases, the threshold is moved further away from the center of the $|g\rangle$ distribution and towards the center of the $|\bar{g}\rangle$ distribution[3]. Moving the threshold in such way makes it a more stringent discrimination for $|\bar{g}\rangle$. "Dynamic thresholding" ensures that $\epsilon_{|\bar{g}\rangle||g\rangle}$ becomes exponentially suppressed while the probability of $\epsilon_{|g\rangle||\bar{g}\rangle}$ occurring decreases.

Consequently, the predominant errors are caused by T_1 of the qubit and latency of the feedback. Given that the T_1 of the qubit is about 60 μs and that the thermal populations of $|g\rangle$ and $|e\rangle$ are at 79.4% and 14.3% , respectively, we can calculate T_{up} by the simple relations that $N_e\Gamma_{down} = N_g\Gamma_{up}$ and $1/T_1 = \Gamma_{up} + \Gamma_{down} = 1/T_{up} + 1/T_{down}$. Each feedback stage has a length of $T_{latency}$, 1 μs. Given these parameters, the maximum ground state purity is $\exp(-T_{up}/T_{latency}) = 99.7\%$. Note that this upper bound excludes population in higher excited states, such as $|f\rangle$, $|h\rangle$...which further reduce the ground state purity.

[2]This means that if the probability for being in $|g\rangle$ and $|\bar{g}\rangle$ is identical, then the threshold would be chosen such that the two errors are the same.

[3]This is consistent with a method in statistical learning called linear discrimination analysis or LDA[Hastie et al., 2005]. In LDA, the decision boundaries (i.e., threshold in our case) for classifying an event or outcome into different classes depend on the expected proportion (prior probability) and distributions (means and standard deviations) of each class.

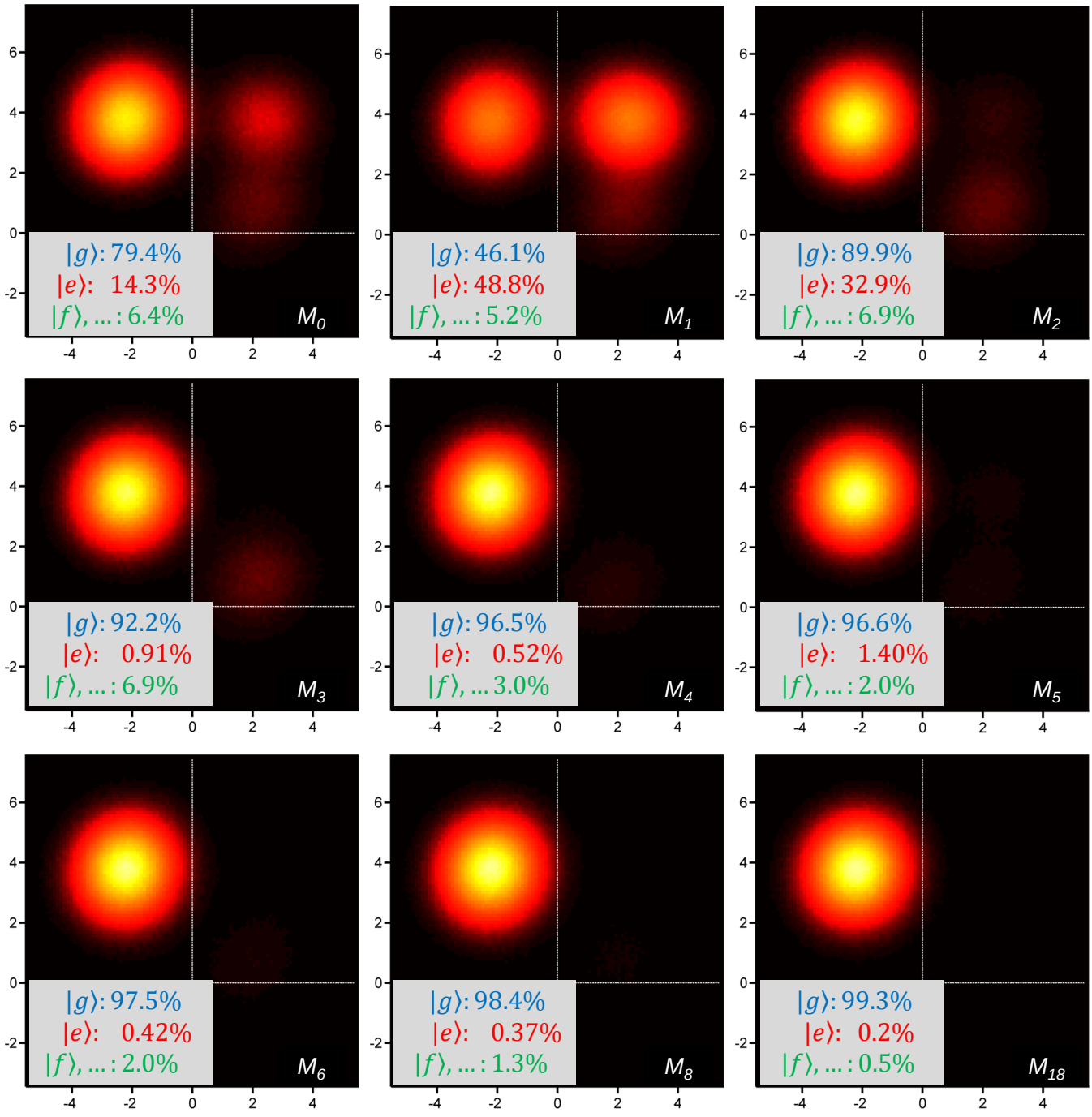

Figure 4.7: Histograms of the mature Maxwell's demon.

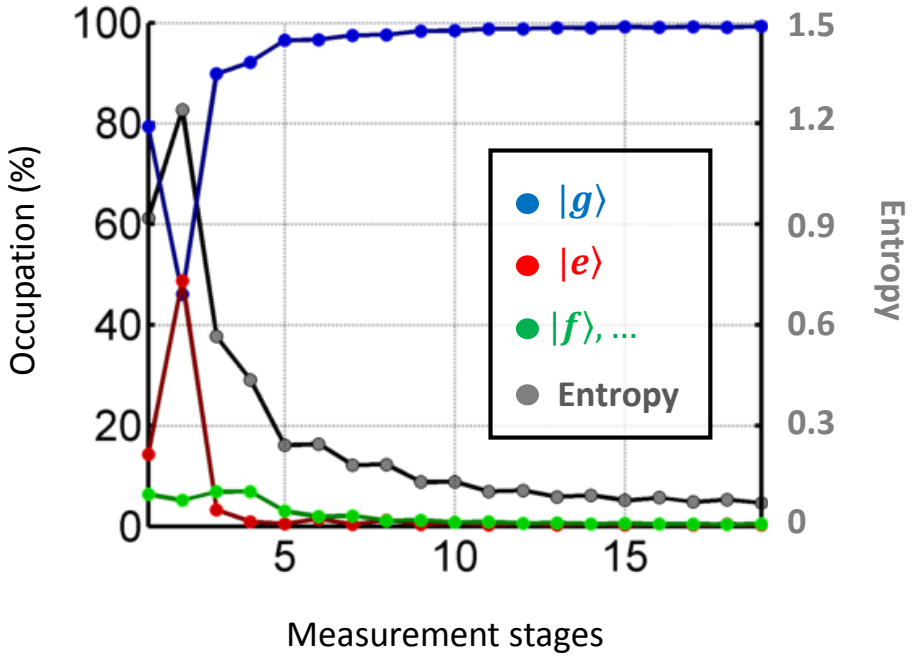

Figure 4.8: Plot of Shannon's entropy as a function of measurement stages.

CHAPTER 5

Stabilizing entanglement

5.1 Introduction

Here we report an experiment in which we built a feedback platform utilizing a nearly quantum-limited measurement chain and a customized field-programmable gate array (FPGA) system to perform MB and DD schemes within the same setup. The task of this platform was to stabilize an entangled Bell state of two superconducting transmon qubits[Schreier et al., 2008]. This particular task of stabilizing a single state is a proxy for more general QEC experiments where a manifold of states is protected. We realize, for the first time, an MB *stabilization* of a Bell state by repeated active correction through conditional parity measurements[Lalumière et al., 2010; Tornberg and Johansson, 2010; Riste et al., 2013]. We compare this scheme to a DD entanglement stabilization scheme[Shankar et al., 2013] in which the conditional parity switch is autonomous. By performing both schemes on the same hardware setup and circuit QED (cQED) system[Blais et al., 2004], we shed light on their close connection and compare them on a level playing field.

Theoretical works in the past have compared DD (under the name of "coherent feedback") and MB for linear quantum control problems[Yamamoto, 2014], such as for minimizing the time required for qubit state purification[Jacobs et al.,

2014] or for cooling a quantum oscillator[Hamerly and Mabuchi, 2012]. These comparisons showed coherent feedback to be significantly superior. Here we experimentally compare DD and MB on identical hardware and study two performance metrics, the state fidelity and success probability. In our particular setup, we find that distinguishing the superior approach among DD and MB is a more subtle task. The subtlety is two-fold. First, the performance difference depends on which process can be better optimized: the design of the cQED Hamiltonian or the efficiency of quantum measurement and classical control. In the current experiment, we show that DD has better steady-state performance as the cQED Hamiltonian parameters are engineered such that DD has a shorter feedback latency. But DD's advantage over MB is not immutable. As certain experimental parameters are improved, such as coherence times and measurement efficiency, MB's performance can catch up with DD.

Secondly, in the current experimental regime in which neither the cQED Hamiltonian parameters nor the measurement and control parameters are ideal, we can obtain a boosted performance by combining DD and MB to get the best of both worlds. We explored this by devising a heralding method to improve the performance of both stabilization approaches. This protocol exploits the high-fidelity measurement capability and the programmability of the feedback platform. The protocol is termed "nested feedback" since it has an inner feedback loop based on either the DD or MB scheme, and an outer loop that heralds the presence of a high-fidelity entangled state in real-time. Previously, heralding schemes have been demonstrated for state preparation to combat photon loss or decoherence[Moehring et al., 2007; Wagenknecht et al., 2010; Hofmann et al., 2012; Johnson et al., 2012; Ristè et al., 2012b,a; Riste et al., 2013; Bernien et al., 2013]. Extending such heralding capability to state stabilization will be a valuable addition to the QEC toolbox. Furthermore, the ability to herald in real time as opposed

to post-selection is important for on-demand and deterministic quantum information processing since only successful events lead to subsequent processing. Real-time heralding for entanglement stabilization is particularly challenging for superconducting qubits due to their shorter coherence times compared to other systems. In this article, we implement this real-time heralding capability on a time scale faster than the few microsecond coherence time of our qubit-cavity system. By extending the feedback platform developed primary for the MB approach to the DD approach, our results bring to light a new application of MB. Adding a level of MB feedback can significantly improve performance beyond what a single layer of feedback, whether DD or MB, can achieve.

We emphasize here the interest in state stabilization by DD/MB methods over more traditional state preparation by a unitary gate. In many algorithms one may simply prepare a new state when it is needed[Nielsen and Chuang, 2004]. Nevertheless, it is preferable to have a stabilized state ready to be consumed so that the algorithm can avoid going through the process of state preparation and suffer from the associated latency. There is also a more fundamental interest in the task of stabilizing a state since these feedback experiments (see previous references) can be understood as a form of Maxwell's demon in action[Bennett, 1987]. The nested feedback approach is applicable to other quantum information platforms with precise Hamiltonian control, efficient quantum measurement and fast classical control such as trapped ions and Rydberg atoms, where simpler forms of feedback have been shown. Moreover it can be applied not just to the stabilization of a single state but also to other quantum information processing tasks such as full quantum error correction of a logical qubit.

In the following, we first describe our experiment setup (Sec. 5.2) and in Sec. 5.3, explain and compare the DD and MB schemes for entanglement stabilization. Then in Sec. 5.4 we introduce experiments where a second layer of feedback is

added to form a nested feedback scheme and discuss its advantages. We end the article by a concluding summary and a short discussion of further work.

5.2 Experiment setup

The simplified schematic of our experimental setup is shown in Fig. 5.1a, while Figure. 5.2 describes the detailed wiring diagram of the system housed in an Oxford Triton 200 dilution refrigerator, at a base temperature below 20 mK. Two transmon qubits[Schreier et al., 2008], Alice and Bob, are dispersively coupled to a three-dimensional aluminum cavity [Paik et al., 2011], with frequency $f_{gg} = 7.5$ GHz when both qubits are in the ground state and linewidth $\kappa/2\pi = 2$ MHz. The photon-number resolved qubit transition frequencies[Schuster et al., 2007] with no photons in the cavity are $\omega^0_{\text{Alice}}/2\pi = 4.87$ GHz and $\omega^0_{\text{Bob}}/2\pi = 6.18$ GHz, for Alice and Bob respectively. The anharmonicity for the two qubits are $\alpha_{\text{Alice}}/2\pi = 212$ MHz, $\alpha_{\text{Bob}}/2\pi = 209$ MHz. Alice (Bob) has a T_1 of 60 μs (18 μs), $T_{2,\text{Ramsey}}$ of 9 μs (10 μs) and excited state population in $|eg\rangle$ and $|ge\rangle$ of 5% each. The dispersive shifts of each qubit to the cavity mode were designed to be nearly equal and in the strong dispersive regime ($\chi_{\text{Alice}}/2\pi = 5$ MHz, $\chi_{\text{Bob}}/2\pi = 4.5$ MHz).

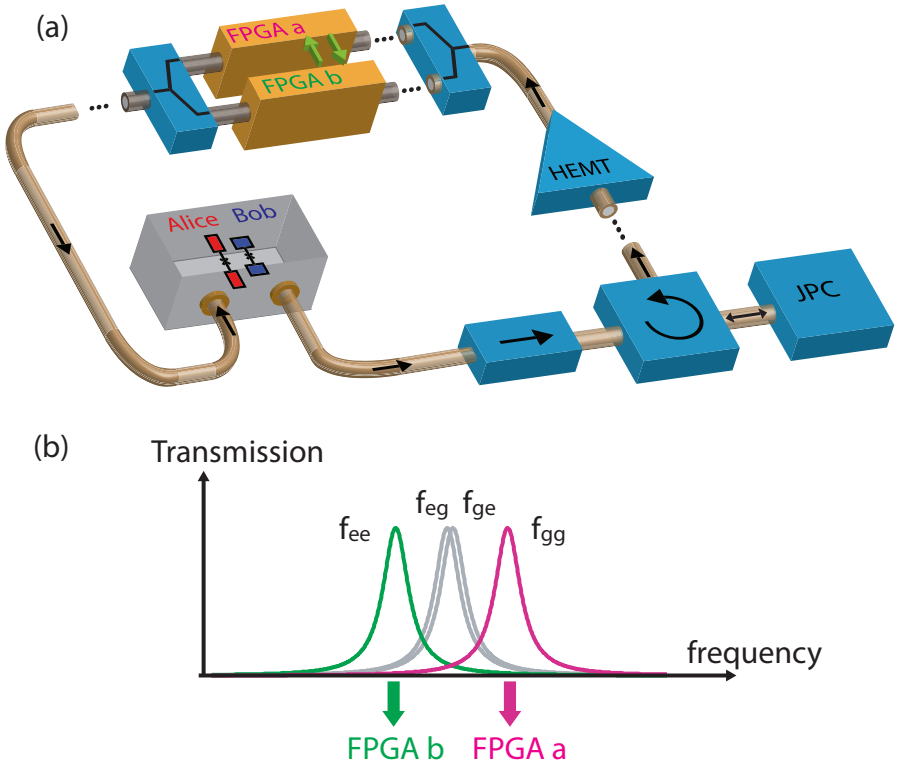

Figure 5.1: (a) Schematic of the experimental set-up. Two independently addressable transmon qubits, Alice and Bob, are dispersively coupled to a three-dimensional microwave cavity. The cavity output is directed to a nearly quantum-limited measurement chain consisting of a Josephson amplifier (JPC) followed by a semiconductor amplifier (HEMT). A pair of custom Field-Programmable Gate Array boards (FPGA a, b) monitor the amplified output and generate real-time modulated microwave drives to control the cavity-qubit system. (b) Transmission spectra of the cavity. Cavity outputs at f_{gg} and f_{ee} are fed to FPGA a and b, respectively.

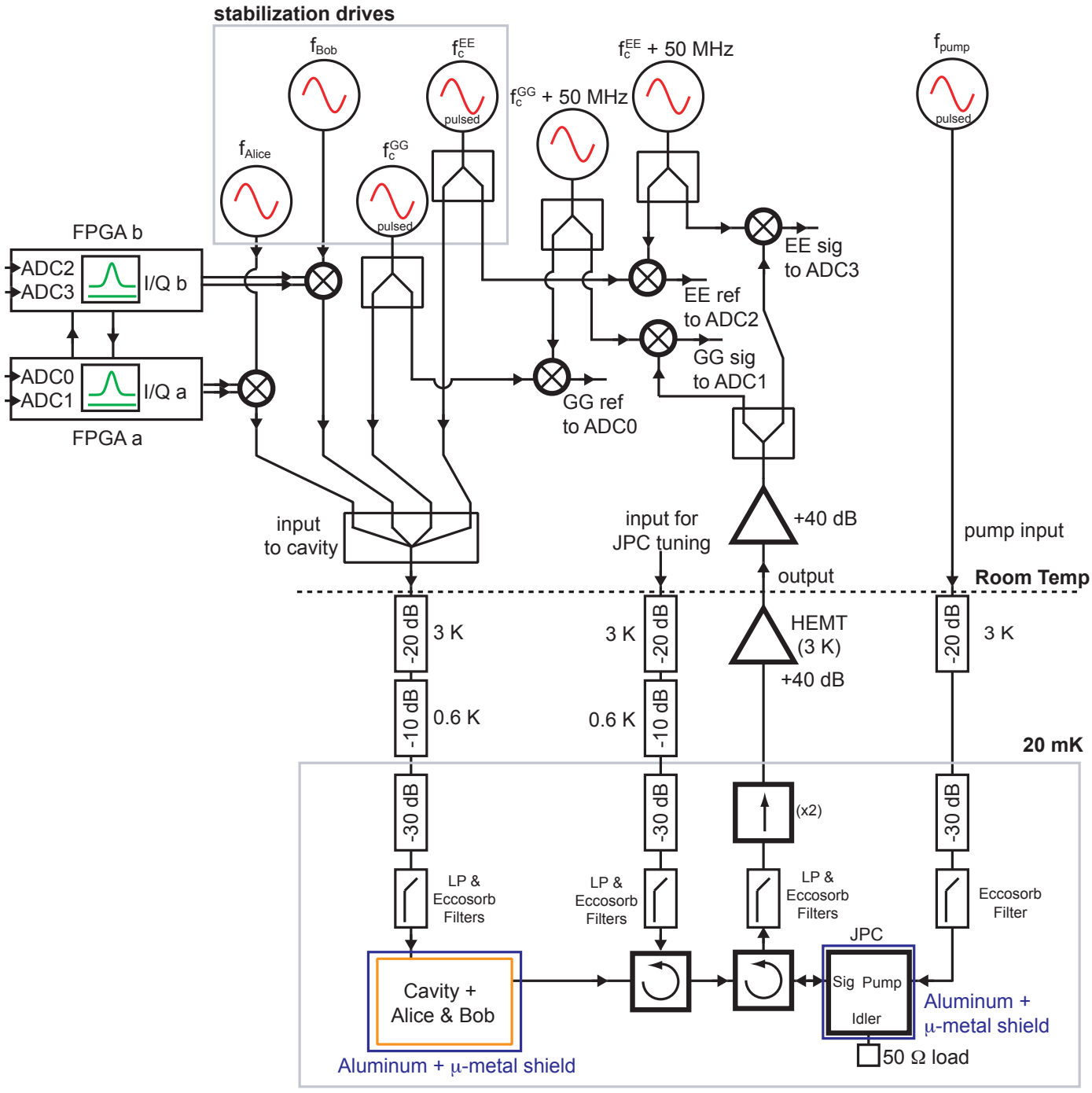

Figure 5.2: See caption on the next page.

Figure 5.2: Setup of the experiment. The qubit-cavity system was placed at the base stage of a dilution refrigerator (Oxford Triton200) below 20 mK. On the input side: Two FPGAs (Innovative Integration X6-1000M) generated the pulse envelopes in I and Q quadratures to modulate the qubit drives at f_{Alice} and f_{Bob} frequencies (Vaunix Labbrick LMS-802) for Alice and Bob, respectively. The I/Q modulations were output by the FPGAs with one and two single-sideband modulations in MB and DD (so that both zero-photon and n-photon qubit frequencies were addressed), respectively. The microwave frequency drives and I/Q modulation were mixed by IQ mixers. Two Agilent N5183 microwave generators produced the cavity drives at f_{ee} and f_{gg} respectively. The cavity drives were also pulsed by the FPGAs. On the output side: the transmitted signal through the cavity was directed by two circulators to the JPC for nearly quantum-limited amplification. It was further amplified at the 3 K stage by a cryogenic HEMT amplifier. After additional room temperature amplification, the signal was split into two interferometric setups for readouts at the f_{gg} and f_{ee} frequencies, respectively. In each of the interferometers, the signal arriving from the fridge was mixed with a local oscillator set $+50$ MHz away to produce a down-converted signal at 50 MHz. A copy of the cavity drive that did not go through the fridge was also down-converted in the same manner to produce a reference. Finally, the two signals with their respective references were sent to the analog-to-digital-converters on the FPGA boards for digitization and further demodulated inside the FPGAs. The two FPGAs jointly estimate the qubits' state by communicating their results with each other. Along the input and output lines, attenuators, low pass filters and homemade Eccosorb filters were placed at various stages to protect the qubits from thermal noise and undesired microwave and optical frequency radiation. Moreover, the cavity and JPC were shielded from stray magnetic fields by aluminum and cryogenic μ-metal (Amumetal A4K) shields.

The cavity output is amplified by a Josephson Parametric Converter (JPC) operated as a nearly quantum-limited phase-preserving amplifier [Bergeal et al., 2010] enabling rapid, single-shot readout[Hatridge et al., 2013] and thus real-time feedback. The JPC was operated with a gain of 20 dB and bandwidth of 6.2 MHz, with the frequency for maximum gain centered between the two readout frequencies, f_{gg} and f_{ee}. The gain and noise visibility ratio at both f_{gg} and f_{ee} were about 17 dB and 6 dB, respectively. We estimate the quantum efficiency η of the combined measurement chain to be 0.3. The amplified output is directed to a room-temperature classical controller realized with two FPGA boards. These FPGA boards are the all-in-one controllers configured with the Yngwie logic described in Chapter 3, responsible for data acquisition, as well as active control of the cavity-qubit system by arbitrary waveform/digital marker generation.

An essential operation for our experiment is a two-qubit joint quasi-parity measurement using the common readout cavity [Tornberg and Johansson, 2010; Lalumière et al., 2010; Riste et al., 2013]. As shown in Fig. 5.1b, the cavity is driven at f_{gg} (both qubits in ground state) and at f_{ee} (both in the excited state) at the same time. The output at f_{gg} and f_{ee} together distinguishes the even parity manifold $\{|gg\rangle, |ee\rangle\}$ from the odd parity manifold $\{|ge\rangle, |eg\rangle\}$. When the two cavity output responses *both* have an amplitude below a certain threshold, the qubits are declared to be in odd parity; when either one has amplitude above the threshold, the qubits are declared to be in even parity. We note that, unlike a true parity measurement, this readout actually distinguishes the two even parity states $|gg\rangle$ and $|ee\rangle$, hence we refer to it as a "quasi" parity measurement. However, the feedback schemes described below apply the same operation on both even states, and thus we need only record the parity of the measured state. The choice of driving at the "even" cavity resonances rather than between the "odd" resonances (f_{eg} and f_{ge}) mitigates the effect of the χ mismatch, reducing associated measurement-induced

dephasing of the odd manifold[Tornberg and Johansson, 2010]. The controller FPGA a (b) modulates the f_{gg} (f_{ee}) drive to the cavity and also demodulates the response. The two FPGAs share their measurements of the cavity response to jointly determine the parity. In addition, FPGA a and b generate the qubit pulses to Alice and Bob, respectively, which are conditioned on the joint state estimation during real-time feedback.

5.3 DD and MB stabilization – fixed time protocol

5.3.1 Principle of experiment

We first briefly outline the DD stabilization of entanglement, described in detail in Ref. [Leghtas et al., 2013] and [Shankar et al., 2013]. This stabilization targets the two-qubit Bell state $|\phi_-\rangle = \frac{1}{\sqrt{2}}(|ge\rangle - |eg\rangle)$. Figure 5.4a displays the states coupled by the autonomous feedback loop. Two Rabi drives on Alice and Bob at their zero-photon qubit frequencies (ω^0_{Alice} and ω^0_{Bob}) couple the wrong Bell state $|\phi_+\rangle$ to the even states, $|gg\rangle$, $|ee\rangle$, in the energy manifold with zero cavity photons. A second pair of Rabi drives at the n-photon qubit frequencies ($\omega^0_{\text{Alice}} - n\bar{\chi}$ and $\omega^0_{\text{Bob}} - n\bar{\chi}$, $\bar{\chi} = (\chi_{\text{Alice}} + \chi_{\text{Bob}})/2$), with their relative phase opposite to the first pair, couple $|gg, n\rangle$, $|ee, n\rangle$ to the Bell state $|\phi_-, n\rangle$. The two cavity drives, at f_{gg} and f_{ee} connect the two manifolds and hence the combined action of the six drives transfers the population from $|gg\rangle$, $|ee\rangle$ and $|\phi_+\rangle$ to $|\phi_-, n\rangle$. Finally, cavity photon decay brings $|\phi_-, n\rangle$ back to $|\phi_-, 0\rangle$. In effect, the cavity drives separate qubit states based on their parity, allowing one pair of Rabi drives to move the erroneous odd population to the even states while the other pair transfers the even states population to $|\phi_-\rangle$.

Counterparts to these elements of the DD feedback loop can be found in the corresponding MB feedback scheme. The action of our MB algorithm is shown as

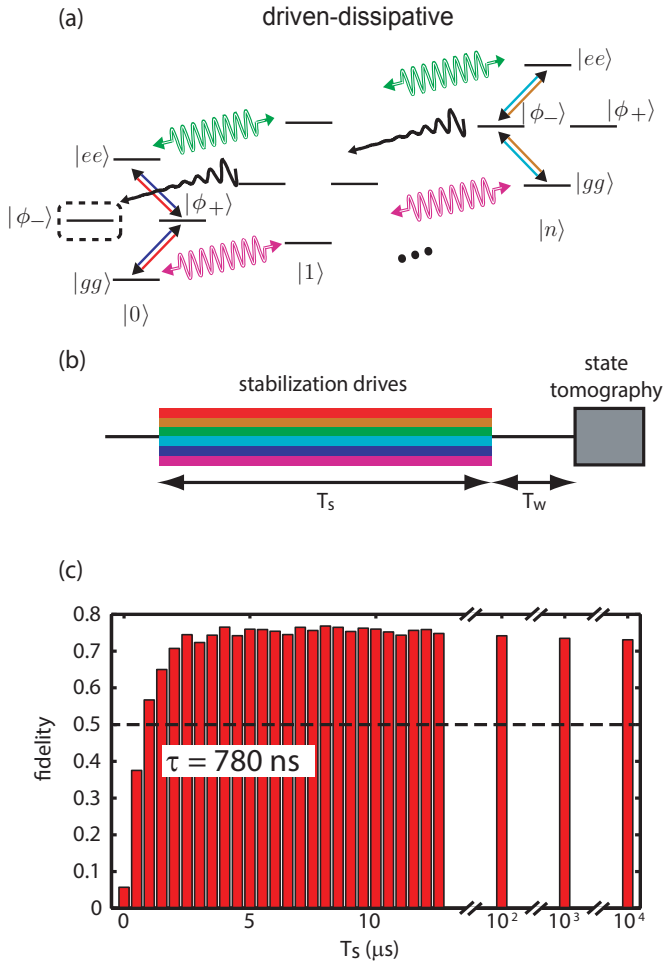

Figure 5.3: Comparison between the driven-dissipative (DD) and the measurement-based (MB) entanglement stabilization Part I: DD. (a) Diagram of qubit/cavity state evolution in the DD feedback loop. Two-qubit state manifolds are laddered for different photon numbers in the cavity (labeled by $|n\rangle$). The green and pink sinusoidal double arrows represent cavity drives, while the straight double red/blue and cyan/yellow arrows are Rabi drives on the qubits. These six drives and the cavity dissipation (black decaying arrow) couple the different states of the system such that the target state $|\phi_-\rangle$ is stabilized. (b) Functional pulse sequence for DD. The duration needed to empty the cavity of residual photons (see text) before tomography is indicated by T_w. (c) Fidelity to the target as a function of stabilization duration (T_s). Dashed line at 0.5 denotes the threshold for entanglement. The time given in the white box, τ, is the characteristic time constant of the exponential rise of fidelity.

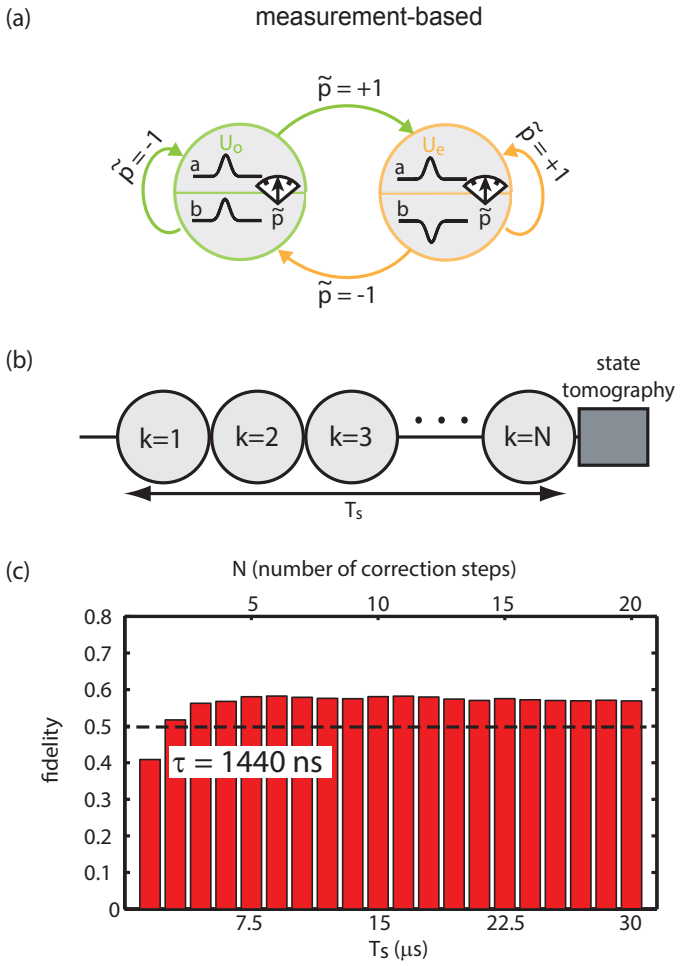

Figure 5.4: Comparison between the driven-dissipative (DD) and the measurement-based (MB) entanglement stabilization Part II: MB. (a) State machine representation of the MB feedback loop. The quasi-parity measurement reports $|gg\rangle$, $|ee\rangle$ as $\tilde{p} = +1$ (even) and $|\phi_-\rangle$, $|\phi_+\rangle$ as $\tilde{p} = -1$ (odd). For odd (even) parity, two $\frac{\pi}{2}$ pulses, with identical (opposite) phases, are applied to Alice and Bob, respectively. (b) Sequence of correction steps conditioned by the quasi-parity measurement and leading into tomography. Counter k limits the number of steps to N. (c) Fidelity to the target Bell state as a function of stabilization duration (T_S) or number of correction steps (N).

a state machine in Fig. 5.4. We describe the quasi-parity measurement $\tilde{P}$ by the projectors $P_{odd} = |ge\rangle\langle ge| + |eg\rangle\langle eg|$, $P_{gg} = |gg\rangle\langle gg|$ and $P_{ee} = |ee\rangle\langle ee|$. We assign the outcomes $\tilde{p} = +1$ to the even projectors, P_{gg} and P_{ee} and $\tilde{p} = -1$ to P_{odd}. The MB algorithm is built with a sequence of correction steps, each of which consists of a conditional unitary and a quasi-parity measurement. The two possible states of the state machine correspond to whether we apply the unitary U_E or U_O, followed by the quasi-parity measurement. Specifically, $U_E = R^{\mathrm{a}}_{\mathrm{x}}(\frac{\pi}{2}) \otimes R^{\mathrm{b}}_{-\mathrm{x}}(\frac{\pi}{2})$ where a (b) denotes Alice (Bob), and $U_O = R^{\mathrm{a}}_{\mathrm{x}}(\frac{\pi}{2}) \otimes R^{\mathrm{b}}_{\mathrm{x}}(\frac{\pi}{2})$. In a correction step k, the qubits are initially in either $|gg\rangle$, $|ee\rangle$ or in the odd manifold, due to the projective quasi-parity measurement in step $k-1$; the controller then applies U_E (U_O) if $\tilde{p}$ in previous step reported $+1$ (-1).

The effect of the state machine on the two-qubit states is shown in Tab. 5.1, where the action of the controller during one correction step is described in terms of the four basis states, $|\phi_-\rangle$, $|\phi_+\rangle$, $|gg\rangle$ and $|ee\rangle$ (the latter two are grouped in the "even" column). The quasi-parity measurement infidelity, labeled by $\epsilon_{E|O}$ ($\epsilon_{O|E}$), gives the error probability of obtaining an even (odd) parity outcome after generating an odd (even) state. Because these measurement infidelities are small, the dominant events are those that occur without measurement errors. At each step, U_E on either $|gg\rangle$ or $|ee\rangle$ followed by the quasi-parity measurement $\tilde{P}$ transfers the states to $|\phi_-\rangle$ with 50% probability. Since $|\phi_-\rangle$ is an eigenstate of U_O and $\tilde{P}$ (modulo a deterministic phase shift that can be undone, see later discussion), these operations leave it unaffected. On the other hand, U_O and $\tilde{P}$ transform $|\phi_+\rangle$ into $\{|gg\rangle, |ee\rangle\}$; more generally, they take population in any other odd state (i.e., a superposition of $|\phi_-\rangle$ and $|\phi_+\rangle$) into $|\phi_-\rangle$ and the even states.

By repeating a sufficient number of these correction steps in sequence, the controller stabilizes the target Bell state irrespective of the initial two-qubit state. The similarity between this active feedback and DD is that MB also transfers popula-

Previous state	$\|\phi_-\rangle$		$\|\phi_+\rangle$		even	
$\tilde{p}_{k-1}$	$+1$	-1	$+1$	-1	$+1$	-1
Outcome probability	$\epsilon_{E\|O}$	$1-\epsilon_{E\|O}$	$\epsilon_{E\|O}$	$1-\epsilon_{E\|O}$	$1-\epsilon_{O\|E}$	$\epsilon_{O\|E}$
Unitary	U_E	U_O	U_E	U_O	U_E	U_O
Next state	even	$\|\phi_-\rangle$	$\|\phi_+\rangle$	even	even/$\|\phi_-\rangle$	even/$\|\phi_+\rangle$

Table 5.1: Effects of the MB finite state machine of Fig. 2 on two-qubit system in the k-th step of feedback, for different starting cases (columns). Row 2 through 4 describe the result of the previous quasi-parity measurement and the corresponding unitary that will be applied in the k-th step. The symbols, $\epsilon_{E|O}$ and $\epsilon_{O|E}$, denote parity measurement errors (see text). The last row describes the possible system states attained by the applied unitary. The two alternative states for a previous "even" occur with 50% probability.

tion between different parity states by conditional Rabi drives. However, while the Rabi drives in DD are conditioned autonomously by the photon number in the cavity, the unitary Rabi pulses in MB are conditioned by real-time parity measurement performed by active monitoring of cavity outputs.

The pulse sequences for DD and MB are shown in Fig. 5.4b and e. In DD, a set of continuous-wave drives are applied for a fixed time $T_{\rm s}$ and after some delay $T_{\rm w}$ to allow remaining cavity photons to decay, a two-qubit state tomography is performed [Filipp et al., 2009; Chow et al., 2010]. The cavity and Rabi drive amplitudes and phases were tuned for maximum entanglement fidelity, following the procedure described in Ref. [Shankar et al., 2013]. In particular, the optimal cavity drive amplitudes were found to be $\bar{n} = 4.0$. For MB, the continuous drives are replaced by a pre-defined number of correction steps N, resulting in a stabilization duration of $T_s = NT_{step}$ where $T_{step} = 1.5$ μs. There is no extra delay before tomography since each correction step already contains a delay after the quasi-parity measurement due to feedback decision latency. The strength and duration of the quasi-parity measurement $\tilde{P}$ were optimized as discussed in Appendix 6.1. The optimization achieved low parity measurement infidelities $\epsilon_{E|O}$ and $\epsilon_{O|E}$ while keeping the measurement-induced dephasing arising from the χ

mismatch[Tornberg and Johansson, 2010; Lalumière et al., 2010] small compared to the natural decoherence in the same duration (the measurement induced dephasing rate was $\Gamma_m/2\pi = 22$ kHz, while the intrinsic decoherence rate of the Bell-state was $\Gamma_{Bell}/2\pi = 30$ kHz). We experimentally determined the infidelity of the quasi-parity measurement to be $\epsilon_{E|O}$ and $\epsilon_{O|E}$ of 0.04 and 0.05, respectively (see Appendix 6.3). The quasi-parity measurement also causes a deterministic qubit rotation about the respective Z axis due to an AC Stark shift [Tornberg and Johansson, 2010]; this rotation was corrected within the unitary gate U_O as discussed in Appendix 6.6.

5.3.2 Results

Fig. 5.4c,f show the fidelity to the target Bell state $|\phi_-\rangle$ as a function of stabilization time for DD and MB, respectively. The fidelity rises exponentially with a characteristic time constant of 0.78 μs (1.4 μs) and asymptotically converges to a steady-state fidelity, F_{ss}, of 76% (57%) for DD (MB). Both fidelity values agree with numerical modeling based on master equation simulation, which gives 76% and 58% for DD and MB, respectively (see the following section). The experimentally determined time constants are in reasonable agreement with their simulated values of 1.0 μs (1.4 μs) for DD (MB). In MB, this loop time is related to the step length (1.5 μs), which is given by the sum of the quasi-parity measurement duration (0.66 μs), the cable, instrument and FPGA latencies (0.69 μs), and the duration of unitary pulses (0.15 μs). On the other hand for DD, the measured loop time is close to 10 cavity lifetimes, the expected time as shown in Ref. [Leghtas et al., 2013].

It is tempting to compare the fidelities achieved by these stabilization protocols to that achieved by the application of a unitary entangling gate [DiCarlo et al., 2009; Chow et al., 2012; Barends et al., 2014]. However, these fidelities cannot be

compared on the same footing. For state preparation, the fidelity is meaningful only immediately after the gate and decays due to decoherence. On the other hand in stabilization, the state fidelity is maintained for an arbitrarily long time as shown in Fig. 5.4c,f.

5.3.3 Steady state model of DD and MB

The steady state behavior of both DD and MB is simulated by a Lindblad master equation, given by

$$\frac{d\rho(t)}{dt} = -\frac{i}{\hbar}[H(t), \rho(t)] + \kappa D[a]\rho(t) + \sum_{j=A,B} \left(\frac{1}{T_{\downarrow}^{j}} D[\sigma_{-}^{j}]\rho(t) + \frac{1}{T_{\uparrow}^{j}} D[\sigma_{+}^{j}]\rho(t) + \frac{1}{2T_{\phi}^{j}} D[\sigma_{z}^{j}]\rho(t) \right) \tag{5.1}$$

D is the Lindblad super-operator, defined for an operator O as $D[O]\rho = O\rho O^{\dagger} - (1/2)O^{\dagger}O\rho - (1/2)\rho O^{\dagger}O$. The pure dephasing rate for Alice and Bob, respectively, is given by $1/T_{\phi}^{A,B} = 1/T_{2}^{A,B} - 1/2T_{1}^{A,B}$, where $1/T_{1}^{A,B} = 1/T_{\downarrow}^{A,B} + 1/T_{\uparrow}^{A,B}$.

The Hamiltonian $H(t)$ is treated differently in DD and MB. For DD, the Hamiltonian is described in detail in the theory proposal [Leghtas et al., 2013] and parameters in the Hamiltonian, such as the cavity and qubit drive amplitudes, are swept in simulation to find the optimal values. The optimal value for the cavity drive amplitude is found to be $\kappa\sqrt{\bar{n}}/2$ with $\bar{n} = 4.0$, and $\kappa/2$ for the qubit drive amplitudes at both zero-photon and n-photon qubit frequencies. The DD simulation predicts a characteristic time constant of 1 μs and a steady state fidelity of 76% (accounting for the delay between stabilization and state tomography to allow remaining cavity photons to decay).

For MB, a correction step is broken into four segments for effectively piecewise master equation simulation. The first part contains the conditional Rabi pulses which are simulated as perfect instantaneous unitary operations on the qubits. The second part is the decay during the pulses (154 ns total). The Hamiltonian

during this part is just the the dispersive interaction between the qubits and the cavity, $H(t) = H_{disp} = \left(\chi_A \sigma_z^A/2 + \chi_B \sigma_z^B/2\right) a^\dagger a$, in the rotating frame of the two qubits (ω_A^0, ω_B^0) and the cavity mode ($(\omega_c^{gg} + \omega_c^{ee})/2$). The third part is the quasi-parity measurement during which the cavity drives at the f_{gg} and f_{ee} resonances are on and the Hamiltonian is given by $H(t) = H_{disp} + 2\epsilon_c \cos\left((\chi_A + \chi_B)t/2\right)(a + a^\dagger)$, where ϵ_c is the amplitude of the cavity drive (660 ns total). The last part is the remainder of the correction step, incurred by the latency of the feedback during which all drives are off and the qubit-cavity system is in free-decay. The dynamics in this part is again simulated by the dispersive interaction, $H(t) = H_{disp}$ (686 ns total). For the piecewise master equation simulation of a complete correction step as four segments, the density matrix at the end of a segment is used as the initial density matrix for the next segment.

Since in MB, the state at the end of a correction step depends only on the initial state at the beginning of the step, we can model the MB scheme as a Markov chain. In the Appendix 6.4, we show how we derive the transition matrix that describes this Markov chain. The model predicts a steady state fidelity of 58% to $|\phi_-\rangle$, agreeing very well with experimental results.

5.3.4 Perspectives on fixed time protocol

The superior performance of DD over MB for the steady-state fidelity is due to the difference in correction loop time, which needs to be shorter than the coherence times of the two qubits for high fidelity entanglement. For the current experimental setup, the latency of the controller and quantum efficiency of the measurement chain, which affects the fidelity of the single-shot readout, result in a longer loop time in MB. A source of the longer feedback loop time is the quasi-parity measurement duration. This measurement duration, which was optimized as discussed in the Appendix 6.1, is limited by dephasing induced by the mismatch in

χ ($\sim 10\%$) and the measurement efficiency of the output chain ($\sim 30\%$), which can both be improved in future experiments. Our simulations (Appendix 6.4) suggest that with current state-of-the-art measurement efficiency value and optimization of the FPGA/cable latency, the MB steady state fidelity can be improved to 66%. The limited measurement efficiency does not affect the performance of DD because the parity measurement and correction take place autonomously within the qubit-cavity system, indicative of its robustness against this hardware limitation. On the other hand, both DD and MB schemes benefit from longer intrinsic coherence times and reduction of the χ mismatch. For example, simulations show (Tab. 6.1 in Appendix 6.4) that if the coherence times are improved to one hundred microseconds (achieved in other state-of-the-art cQED setups), both DD and MB fidelities can increase to above 85%. For the rest of the article, however, we consider boosting the fidelity in a different manner, without making any physical changes to the qubit-cavity system.

Before we continue, a discussion on the effect of the cavity linewidth κ is due. Increasing κ while keeping χ the same would harm the performance of DD since the selective Rabi drives would not be as selective[Leghtas et al., 2013; Shankar et al., 2013]. While for some MB experiments, such as initializing a qubit to the ground state, increasing κ is beneficial as this reduces measurement time [Gambetta et al., 2007], this is not necessarily true in the MB scheme of stabilizing entanglement where the qubit T_2 is the dominant error process. In our system, the T_2's of the qubits are limited by thermal photons in the cavity resulting in $T_2 \approx 1/(n_{th}\kappa)$, where n_{th} is the average thermal photon number[Sears et al., 2012]. Thus increasing κ would reduce T_2. While the MB feedback loop would become faster by the bigger κ, the reduced coherence time eliminates any gain in fidelity.

5.3.5 Motivation for an improved protocol

The DD and MB schemes described so far are synchronous in the sense that the stabilization always ends after a pre-determined duration and the tomography follows. Decoherence and measurement errors cause the qubits to have a finite probability ($1 - F_{ss}$ =24% and 43% for DD and MB, respectively) of not being in the target state when the stabilization terminates. A more optimum protocol would rather utilize all available information to determine when to end the stabilization. For both DD and MB, information is available in the cavity output, that we can measure at the end of the stabilization period. The outcomes of these measurements, I_{gg} and I_{ee}, give real-time information on the state of the two qubits, and thus can herald a successful stabilization sequence.

In Fig. 5.5, we describe how monitoring the cavity outputs improves target state fidelity. We introduce two thresholds $\{I_{gg}^{herald}, I_{ee}^{herald}\}$ (see Appendix for details) to post-select the measurement outcomes of I_{gg} and I_{ee} respectively, and identify successful stabilization runs [Riste et al., 2013; Shankar et al., 2013]. The results of varying $\{I_{gg}^{herald}, I_{ee}^{herald}\}$ are shown in Fig. 5.5b,d for DD and MB, respectively. The color plots show fidelity improving as the thresholds become more stringent. The success probability defined as the percentage of stabilization runs kept for tomography given a set of thresholds, is also plotted as contours for both DD and MB. There is a clear trade-off between success probability and fidelity. To reach the maximum fidelity in DD of 82%, at least 75% of experiment runs need to be discarded. The trade-off is less severe in MB, where only 50% of runs need to be discarded to reach the maximum fidelity of 75%. However we aim to eliminate this trade-off all together, i.e., to improve the fidelity while maintaining a high success probability.

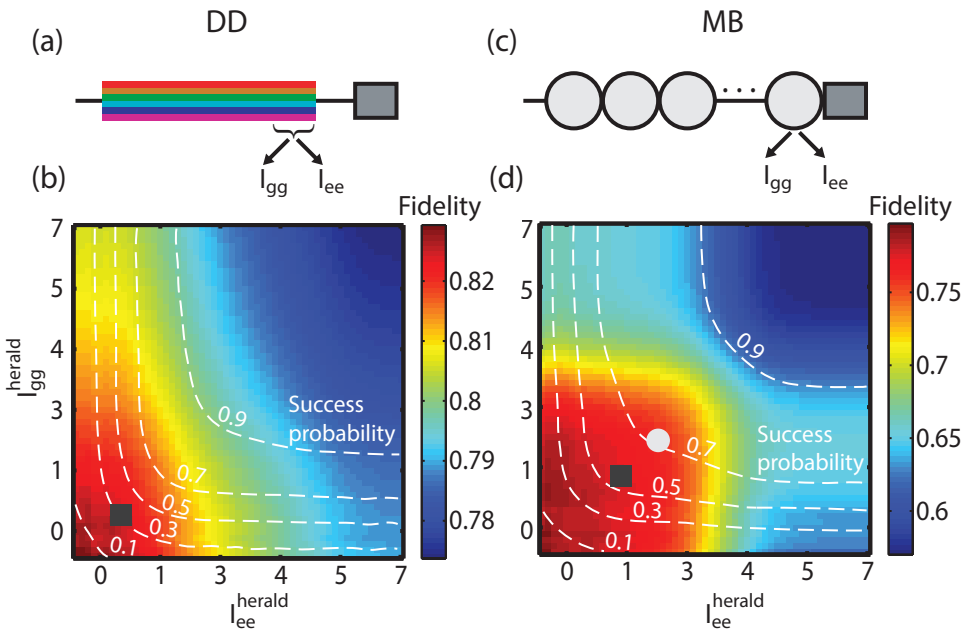

Figure 5.5: Trade-off between success probability and fidelity for both DD and MB schemes obtained by experiment. (a) Pulse sequence for heralding DD stabilization by post-selection. At the end of the stabilization period, the cavity outputs, I_{gg} and I_{ee} are measured at their respective frequencies. (b) Color plot of fidelity to the target state for DD as a function of thresholds chosen for I_{gg} and I_{ee} (see Appendix for details of the thresholds). Also plotted as white dashes are contour lines of the success probability associated with each choice. Solid black square indicates the thresholds chosen for the condition C in the nested feedback protocol described in Fig. 4. (c) and (d) same as above for MB, including the corresponding thresholds for C. Gray circle indicates the thresholds chosen for the quasi-parity measurement $\tilde{p}$ used to condition MB correction steps.

5.4 DD and MB stabilization - nested feedback protocol

This goal is achieved by introducing a nested feedback protocol (NFP), in which the stabilization feedback loop enters into a higher layer of feedback for "fidelity boosting" instead of proceeding to state tomography directly. In contrast to the "fixed time" protocol, NFP conditions the termination of stabilization on the quality of the entanglement, i.e., it heralds a successful stabilization run in real-time, as illustrated by the state machine diagram in Fig. 5.6a. The control variable C is given by $C = (I_{gg} < I_{gg}^{herald})$ **AND** $(I_{ee} < I_{ee}^{herald})$, where $\{I_{gg}^{herald}, I_{we}^{herald}\}$ are determined by the same post-selection experiment discussed previously to optimize the fidelity (black square in Fig. 3b and d). If the controller determines that the entanglement quality is not sufficient ($C = 0$), a boost phase is attempted which comprises exactly one correction step for MB or a stabilization period of similar duration for DD ($1.4\ \mu$s). During the boost phase, the cavity outputs are integrated to give $\{I_{gg}, I_{ee}\}$ which enables the next real-time assessment of C. In DD, the parity measurement and first layer of feedback is accomplished autonomously, therefore the FPGA only needs to check C. However in the MB scheme both layers of feedback are performed solely by the FPGA. It therefore checks if $C = 1$ to herald that the entanglement meets the desired quality. If not, it uses the quasi-parity thresholds (grey circles in Fig. 3d) to decide whether the qubits are in even or odd state in order to continue stabilization. This asynchronous pulse sequencing and conditioning by multiple thresholds exploit the programmable nature of the FPGA-based platform.

The asynchronous behavior of NFP is displayed in Fig. 5.6b(d) for DD (MB), which demonstrates 200 single-shot runs. The DD (MB) fidelity boosting sequence continues until either success or a maximum limit on boost attempts (set to 11 in

the experiment) is reached. For the MB protocol, the trajectory of the qubits' parity can be tracked by the conditioning outcomes of the inner-loop control variable $\tilde{p}$ and the outer-loop control variable C, which are independent. Through repeated boost attempts until success, NFP significantly improves the overall success probability. Within 11 attempts, 95% (99.8%) of DD (MB) runs satisfy the success condition compared to just 25% (50%) with simple post-selection. This is assessed by the cumulative probability, the integral of the probability of having completed a certain number of boost attempts before tomography, as plotted in Fig. 5.7c,f. Since MB requires a less stringent threshold than DD to gain fidelity improvement, the MB success probability converges to unity much faster than that of DD. Finally, we show that the high success probability does not come at the cost of reduced fidelity. The fidelity to $|\phi_-\rangle$ for DD improves from an unconditioned value of 76% to 82% (averaged over all successful attempts). For MB, the improvement is more pronounced: fidelity rises from an unconditioned value of 57% to 74%. Thus for both DD and MB, NFP attain close to the fidelity achieved via stringent post-selection. This improvement can be simulated through a Markov chain model, extended from that introduced in Sec. 5.3.3 and the results given here agree well with the simulation(see Appendix 6.5).

One will note, however, a continuous downward trend of the fidelity in both DD and MB schemes as the number of attempts increases. This is due to the non-negligible population in the $|f\rangle$ states of the two qubits in the experiment, which escape correction by the stabilization feedback loops. After each further boost attempt of stabilization, the probability of the population escaping outside the correction space thus increases, diminishing the fidelity (see Appendix 6.7). Also note that the error bars on the fidelity of MB are bigger than those in DD for large attempt numbers simply because the probability of needing many attempts is lower in MB than in DD.

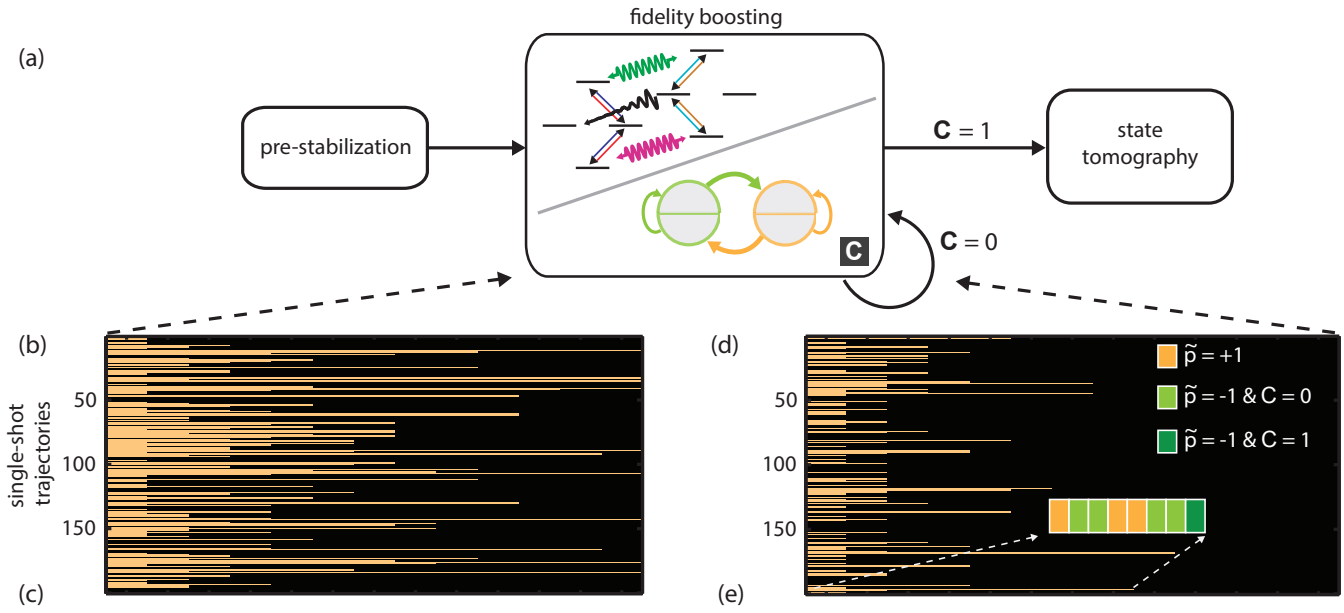

Figure 5.6: Nested feedback protocol for boosting fidelity and heralding successful stabilization run in real-time. (a) State machine representation. The control variable C (see main text and Fig. 3) determines the repetition of a boost cycle or the heralding of a successful run. The maximum number of boost attempts allowed is set to 11 in the experiment. (b), (c) 200 single-shot sequence trajectories of nested feedback for DD and MB, respectively. The trajectories are colored yellow during boost attempts and black after C is satisfied. Trajectories that are entirely black satisfied the success criterion without the need for boost. The inset in (e) shows an example of an MB trajectory consisting of both $\tilde{p}$ and C outcomes.

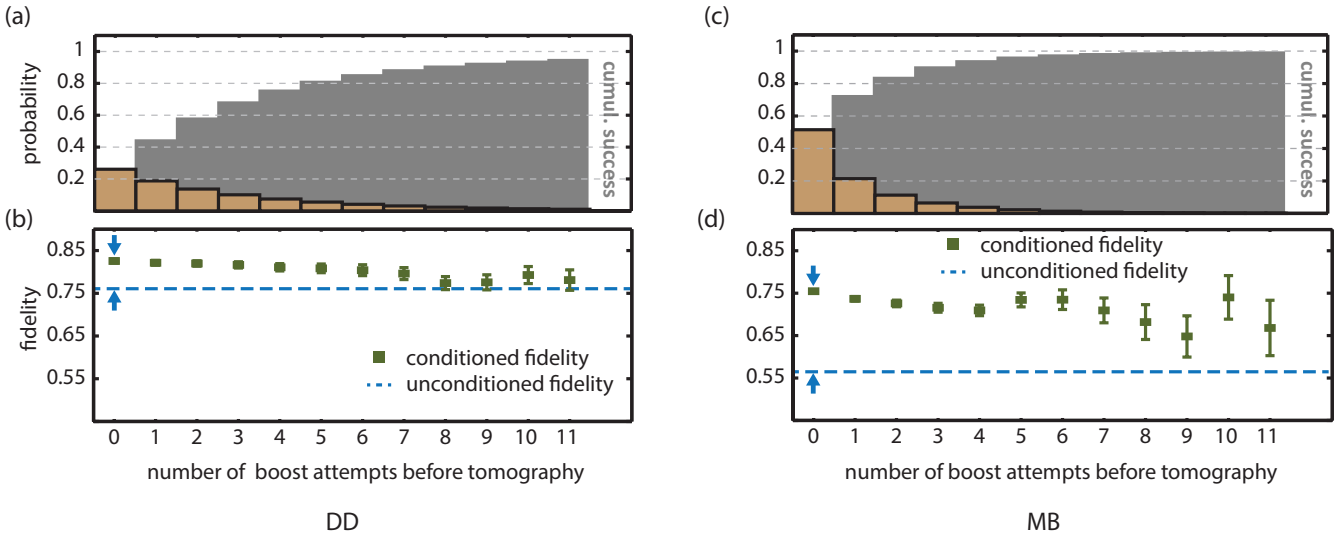

Figure 5.7: Nested feedback protocol for boosting fidelity and heralding successful stabilization run in real-time. (a),(c) Cumulative success probability (gray shade) of having completed at most a given number of boost attempts before tomography for DD and MB, respectively. Yellow bars indicate differential success probability. (b),(d) Fidelity to target Bell state for DD and MB, respectively. Green squares show the corresponding fidelity as a function of the number of boost attempts. The blue dashed line denotes fidelity without any boosting (i.e. unconditioned by C). Blue arrows denote the improvement in fidelity due to nested feedback. While the absolute fidelity of MB is worse than DD due to latency, the fidelity boost is higher.

While real-time heralding by NFP removes the trade-off between fidelity and success probability, it does so by introducing a different trade-off – high fidelity and success probability are achieved but the protocol length now varies from run to run. If NFP is a module within a larger quantum information processing (QIP) algorithm, then this asynchronous nature must be accommodated by the controller. For our FPGA-based control, NFP is easily accommodated because it is a natural extension to "fixed-time" or synchronous operation. In "fixed time" operation, the controller conditions its state by the protocol length which is predetermined and stored in an internal counter by the experimenter. On the other hand in NFP, the controller conditions its state on a pre-determined logical function of its real-time inputs.

5.5 Conclusion

In conclusion, we have implemented a new measurement-based stabilization of an entangled state of two qubits, which parallels a previous driven-dissipative stabilization scheme. Instead of coherent feedback by reservoir engineering, MB relies on actively controlled feedback by classical high-speed electronics external to the quantum system. When comparing both schemes in the "fixed-time" protocol, we observe that DD gives a higher fidelity to the target state due to lower feedback latency. Furthermore, we have improved the fidelity of both schemes by a nested feedback protocol which heralds stabilization runs with high-quality entanglement in real time. The real time heralding brings about the fidelity improvement without a common trade-off in QIP: it does not sacrifice the experiment success probability. It eliminates this trade-off by allowing asynchronicity in the experiment.

Our experiment shows some of the key advantages of MB platforms that have not been previously explored. Typically, the performance of MB feedback in terms

of state fidelity has not been at par with methods based on post-selection, due to the latency of the controller. However, post-selection methods improve state fidelity at the expense of low success probability, and hence existing digital feedback experiments have focussed on achieving deterministic state preparation, i.e. with perfect success probability[Campagne-Ibarcq et al., 2013; Ristè et al., 2012a; Riste et al., 2013; Steffen et al., 2013]. Here, we are exploring another direction of feedback which achieves high fidelity with high success probability. Our nested feedback strategy maximizes the use of the information coming out of the qubit-cavity system in order to make the correction process as efficient as possible. We find that our feedback platform, comprised of a nearly-quantum-limited measurement chain and a real-time classical controller, provides the necessary tool-set to implement such a strategy. We show that this technology can be extended to improve the performance of DD approaches as well as single-layer MB approaches themselves. This strategy could be carried out further in the future. For example, the FPGA state estimator could perform a more sophisticated quantum filter of the microwave output of the DD stabilization to herald successful events with better accuracy, significantly improving the success probability convergence rate. We also note that tools from optimal stopping, a well studied subfield of applied mathematics[Chow et al., 1971], could be used to improve our current implementation of nested feedback.

Similar ideas can be applied in the future towards other forms of stabilization, such as for stabilizing Schrödinger cat states of a cavity mode[Mirrahimi et al., 2014], a proposed logical qubit. Initial experiments on such logical qubits with high fidelity-measurement[Sun et al., 2014] or dissipation engineering[Leghtas et al., 2015] have been performed and could now be combined. Likewise, future logical qubits based on the surface code[Fowler et al., 2012] could also be stabilized by either active stabilizer measurements[Barends et al., 2014; Chow et al.,

2014; Ristè et al., 2014] or as recently proposed by dissipation engineering[Kapit et al., 2015; Fujii et al., 2014]. The idea of combining elements of measurement-based and driven-dissipative feedback into a nested protocol is also agnostic to the particular physical quantum system being controlled and thus could for example be applied to feedback experiments with trapped-ion[Lin et al., 2013; Nigg et al., 2014] and Rydberg atom systems[Sayrin et al., 2011] as well. Our experiment demonstrates that measurement-based and driven-dissipative approaches, far from being antagonistic, can be merged to perform better than either approach on its own.

CHAPTER 6

Calibration experiments and simulation for entanglement stabilization

6.1 Measurement strength and duration calibration for MB

The quasi-parity measurement strength and duration were optimized in order to maximize the fidelity of MB. This optimization was done by maximizing the fidelity of the Bell state created in a calibration experiment, similar to Ref. [Riste et al., 2013]. The qubits are prepared in ground states (by post-selection) and then two $\pi/2$ pulses, are applied to Alice and Bob, producing the state $|\psi\rangle = \frac{1}{2}(|gg\rangle + |ee\rangle + |ge\rangle + |eg\rangle)$. The quasi-parity measurement, consisting of the two cavity drives on f_{ee} and f_{gg} respectively, projects the qubits into one of the two even states or entangles the qubits into a Bell state with odd parity. We varied the duration of this parity measurement and its strength in terms of photon number (set to be identical) for each readout frequency to find the parameters that maximize the fidelity of the entangled state to the closest Bell state (Fig. 6.1). The Bell state fidelity would ideally increase and asymptotically approach one with increasing measurement time as the parity measurement better distinguishes the odd Bell state from the even states. On the other hand, at long measurement

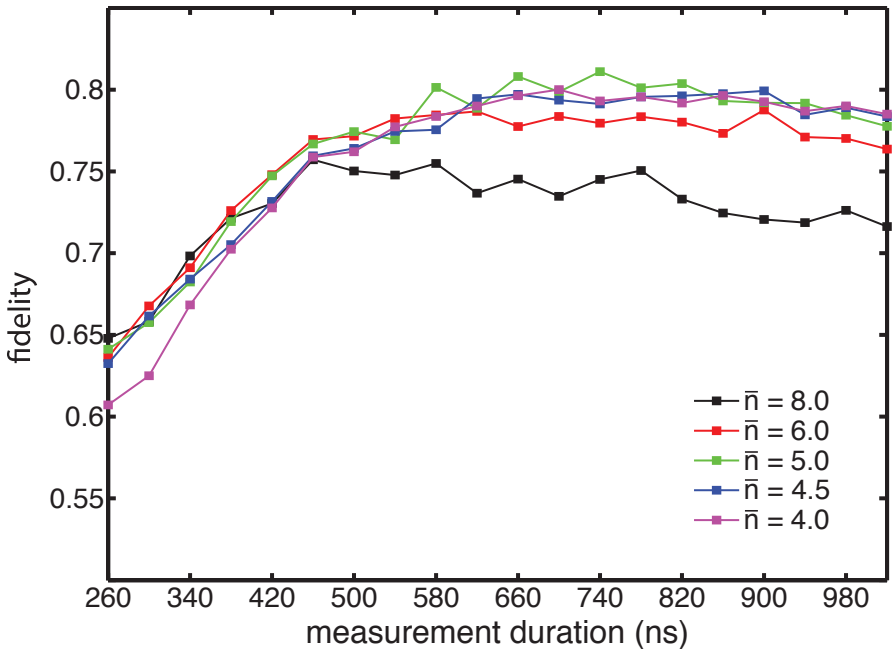

Figure 6.1: Experimental calibration of measurement strength and duration for MB. Fidelity to the state $|\phi_+\rangle = \frac{1}{\sqrt{2}}(|ge\rangle + |eg\rangle)$ after preparing the qubits in a maximally superposed state followed by a quasi-parity measurement with post-selection. The fidelity is plotted as a function of measurement duration and shown for a set of measurement strengths. The optimal values chosen for the experiment are 660 ns and $\bar{n} = 4.5$.

duration, the coherence of the entangled state decreases due to both natural and measurement-induced dephasing, the latter of which is caused by the χ mismatch between the qubits and is proportional to the average number of photons used for the measurement [Tornberg and Johansson, 2010]. Therefore, there is an optimal measurement strength and duration. For our experiment, $\bar{n} = 4.5$ for each read-out frequency and a measurement duration of 660 ns are found to be close to the optimal values and are chosen to attain a Bell state fidelity of 80%.

The value of 80% sets the upper bound on the fidelity that we should expect for heralding MB. In the actual MB experiment, an extra 310 ns delay was introduced after the quasi-parity measurement in a correction step, which does not occur in the sequence described in this section for optimizing the parity measurement parameters. This extra delay was required to accommodate the feedback latency in

MB. The conditioned fidelity we obtained for heralded MB is about 6% lower.

6.2 Measurement outcomes distribution for DD and MB

The cavity outputs at f_{gg} and f_{ee} for both DD and MB can be used to monitor the state of the qubits during stabilization (Fig. 6.2a). Histograms of the measurement outcomes I_{gg} and I_{ee}, recorded by integrating the cavity output at f_{gg} and f_{ee}, respectively, are shown in Fig. 6.2b,c,d,e for DD and MB respectively. In DD, the cavity output signals are captured while all the CW drives are still on, i.e., the qubits are being driven while their states are being monitored, whereas in MB the outputs are a result of the quasi-parity measurements which occur after the qubit pulses and thus when the qubits are not driven. We observe that the measurement outcome distribution of DD lacks the separation seen in MB which has a clear parity separatrix $\{I_{gg}^{parity}, I_{ee}^{parity}\}$. This feature also appears in numerical simulations of DD by the stochastic master equation [1]. The state estimation used in both DD and MB uses the "box car" filtering [Gambetta et al., 2007] which simply sums up the recorded cavity output signals over time to obtain the measurement outcomes. This method, while appropriate for MB, is not suited for DD since in the latter, the qubits are undergoing actively-driven dynamics when the measurement is taking place. A more advanced filter, such as a non-linear quantum filter can be designed from either first-principles or machine learning [Magesan et al., 2015] in the future to improve the state discrimination accuracy in DD.

The measurement outcomes to the left of both the I_{gg}^{herald} (shown in figure) and I_{ee}^{herald} thresholds are much less likely to come from even states than those to the right. Therefore the experiment runs with these outcomes are selected for state to-

[1] M. Silveri et al. (in preparation)

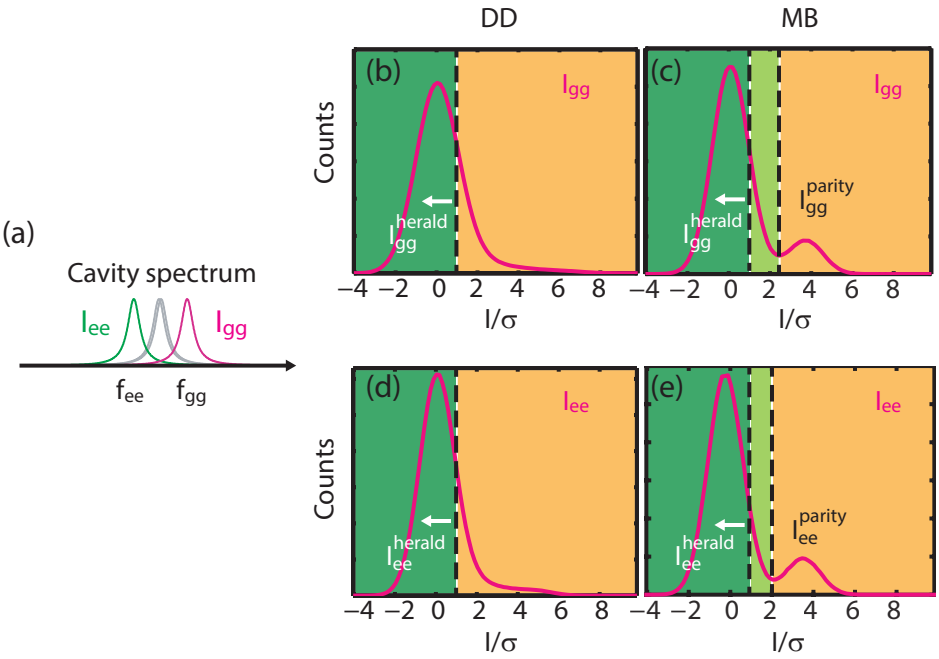

Figure 6.2: (a) The cavity outputs at f_{gg} and f_{ee} are integrated to obtain measurement outcomes I_{gg} and I_{ee}, respectively. (b),(c), (d), (e) Histograms of the measurement outcomes I_{gg} and I_{ee} for DD and MB, respectively. The leftmost dashed line (labeled I_{gg}^{herald} and I_{ee}^{herald}) indicates a particular choice of threshold for heralding the measurement outcomes. Counts to the left of both thresholds are declared success. The right dashed line (I_{gg}^{parity} and I_{ee}^{parity}) indicates the threshold for quasi-parity measurement in MB.

mography, giving the results plotted as a color map in Fig. 3 (main text). Moving the threshold further to the left increases the stringency of the threshold as fewer measurement outcomes are included. The success probability for each threshold choice (plotted as contours in Fig. 3) is calculated by the ratio of included outcomes to the total number of experiment runs.

6.3 Determining quasi-parity measurement infidelities

The determination of quasi-parity measurement infidelities consists of two steps, preparing a state of known parity followed by a quasi-parity measurement. An example histogram of the quasi-parity measurement is shown in Fig. 6.2c,e. Outcomes that are simultaneously to the left of the parity separatrix $\{I_{gg}^{parity}, I_{ee}^{parity}\}$ (right-most dashed lines in Fig. 6.2c,e) are determined to be "odd" while outcomes on the right of either separatrix are determined as "even". To determine the even parity measurement error, the qubits are first prepared in $|gg\rangle$ with a fidelity greater than 99% and the number of "odd" outcomes. Normalizing these error counts by the total number of outcomes gives $\epsilon_{O|E}$, the error probability of obtaining an odd parity outcome after generating an even state. Similarly, to determine the odd parity measurement infidelity, the qubits are first prepared in $|eg\rangle$ by preparing in $|gg\rangle$ followed by a π pulse on Alice and the number of "even" outcomes is determined. Normalizing these error count by the total number of outcomes gives $\epsilon_{E|O}$), the error probability of obtaining an even parity outcome after generating an odd state. We chose to use the states $|gg\rangle$ and $|eg\rangle$ due to the significantly longer T_1 of the Alice qubit which provides a more accurate estimate of the infidelities.

6.4 Steady state model for DD and MB

As discussed in the main text, we can describe the qubits by the density matrix $\rho = \pi_- |\phi_-\rangle \langle\phi_-| + \pi_+ |\phi_+\rangle \langle\phi_+| + \pi_{gg} |gg\rangle \langle gg| + \pi_{ee} |ee\rangle \langle ee|$. Therefore, in terms of probability distributions in the four basis states, the qubits's state, $\tilde{S}$, can be

represented by a vector,

$$\tilde{S} = \begin{pmatrix} \pi_- \\ \pi_+ \\ \pi_{gg} \\ \pi_{ee} \end{pmatrix}. \tag{6.1}$$

If the qubits are prepared in $|\phi_-\rangle$, that is $\tilde{S}^{(i)}_{\phi_-} = (1,0,0,0)^{\mathrm{T}}$, we can calculate $\tilde{S}^{(f)}_{\phi_-}$ after a correction step by applying the master equation simulation method described above. We need to consider the two possible cases where the conditional unitary applied is U_O or U_E, respectively. The $\tilde{S}^{(f)}_{\phi_-}$ is a weighted average of the two cases.

$$\tilde{S}^{(f)}_{\phi_-} = (1-\epsilon_{E|O}) * \tilde{S}^{(f)}_{\phi_-|U=U_O} + \epsilon_{E|O} * \tilde{S}^{(f)}_{\phi_-|U=U_E} \tag{6.2}$$

$\epsilon_{E|O}$ and $\epsilon_{O|E}$ are the quasi-parity measurement infidelities due to limited measurement efficiency, introduced in the main text. In a similar manner, we can obtain $\tilde{S}^{(f)}_{\phi_+}$. In the case of an even initial state, for example, $\tilde{S}^{(i)}_{gg} = (0,0,1,0)^{\mathrm{T}}$, we have

$$\tilde{S}^{(f)}_{gg} = (1-\epsilon_{O|E}) * \tilde{S}^{(f)}_{gg|U=U_E} + \epsilon_{O|E} * \tilde{S}^{(f)}_{gg|U=U_O} \tag{6.3}$$

And similarly for $\tilde{S}^{(f)}_{ee}$.

Given $\tilde{S}^{(f)}_{\phi_-}$, $\tilde{S}^{(f)}_{\phi_+}$, $\tilde{S}^{(f)}_{gg}$ and $\tilde{S}^{(f)}_{ee}$, we can construct the transition matrix $\mathcal{T}$ of a correction step,

$$\mathcal{T} = (\tilde{S}^{(f)}_{\phi_-}, \tilde{S}^{(f)}_{\phi_+}, \tilde{S}^{(f)}_{gg}, \tilde{S}^{(f)}_{ee}) \tag{6.4}$$

where the $\tilde{S}^{(f)}$'s are the columns of the 4 by 4 matrix. Now applying this transition matrix on any arbitrary initial state gives the final state after a correction step,

$$\tilde{S}^{(f)} = \mathcal{T}\tilde{S}^{(i)} \tag{6.5}$$

The transition matrix $\mathcal{T}$ is also called the stochastic matrix, with the property

that each column sums to 1. One of $\mathcal{T}$'s eigenvalues is guaranteed to be 1 and the corresponding eigenvector, $\tilde{S}_\infty$, is the steady state of the Markov chain. It can easily be shown that for any arbitrary initial state, $\tilde{S}^{(i)}$

$$\lim_{k\to\infty} \mathcal{T}^k \tilde{S}^{(i)} = \tilde{S}_\infty \tag{6.6}$$

For the given experimental parameters in MB, the $\mathcal{T}$ matrix is displayed in Fig. 6.3 and we find the steady state eigenvector to be

$$\tilde{S}_\infty = \begin{pmatrix} 0.58 \\ 0.11 \\ 0.18 \\ 0.13 \end{pmatrix} \tag{6.7}$$

Thus, the Markov model predicts a steady state fidelity of 58% to $|\phi_-\rangle$. Taking into account of the duration of a correction step (1.5 μs), we can also calculate the characteristic time constant of the MB scheme from the model, which gives 1.4 μs. Both values agree very well with experimental results.

We can calculate the expected fidelity when some of the experimental parameters are improved in the near future. If the measurement efficiency is improved from 30% to the current state-of-the-art value of 60%, the measurement duration can be reduced by half while maintaining the quasi-parity measurement infidelities[Hatridge et al., 2013]. The instrument and FPGA latencies incurred in the experiment can also be reduced by 100 ns, in the latest hardware setup and FPGA logic design in operation while this article was being prepared. The measurement duration and control latency reduction can shorten the correction step length to 1 μs, which can improve the steady state fidelity to 66%. Furthermore, if the coherence times are also improved to the state-of-the-art values in the hundred of

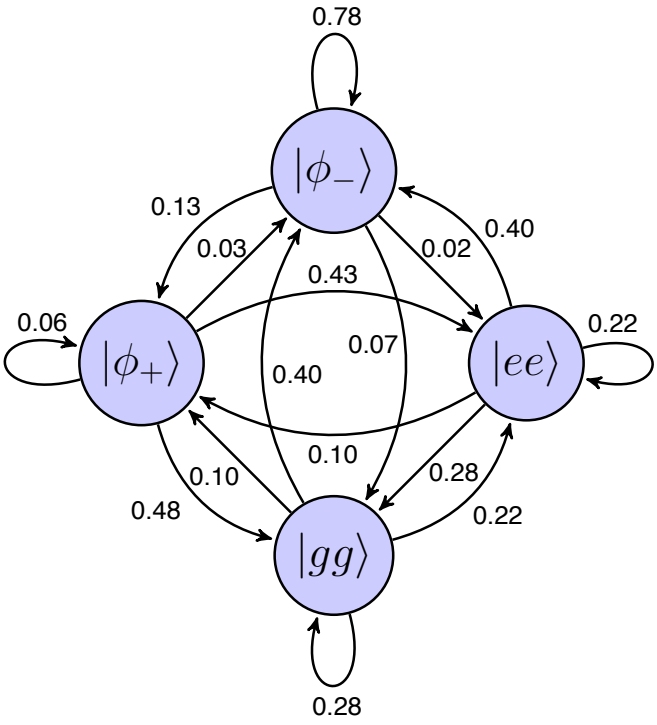

Figure 6.3: Markov model of a correction step in MB. Transition between any two nodes is possible and is represented as a directional edge. The 16 possible transitions make up the 16 matrix elements in the stochastic transition matrix, $\mathcal{T}$, corresponding to a correction step in MB. Numbers next to the edges represent the elements of $\mathcal{T}$ calculated for the current experiment parameters by a master equation simulation.

		Steady state fidelity	
		DD	MB
Experiment	current parameters	76%	57%
Simulation	$\eta = 0.6$, latency $= 586$ ns	76%	66%
	T_1,$T_2 = 100$ μs	86%	86%

Table 6.1: Listing of the steady state fidelities of the "fixed time" protocol for DD and MB schemes for the current experiment (see detailed parameters in Sec. 5.2) and simulation with improved parameters, assuming χ values as in the current experiment. The prospective values in the last row also take into account the parameter changes in the second row. Note that measurement efficiency and control latency change do not affect DD.

microseconds for superconducting qubits, both DD and MB fidelities in the "fixed time" protocol can be above 85%, limited by the χ mismatch (assumed to be 10%, as in the current experiment). The prospects of both DD and MB schemes are summarized in Tab. 6.1

6.5 Simulation of real-time heralding by nested feedback protocol for DD and MB

The Markov chain model introduced in Sec. 6.4 can be extended to simulate NFP (nested feedback protocol). We can construct a "nested" Markov chain (Fig. 6.4) for each boost attempt of NFP. At the outer-most level, there are two nodes. One node denotes the trajectories that have just been heralded as successful; the other node denotes the trajectories that require at least another boost attempt. Building on the vector description established in the previous section, we represent the heralding of trajectories before a boost attempt by a diagonal matrix $\tilde{c}$,

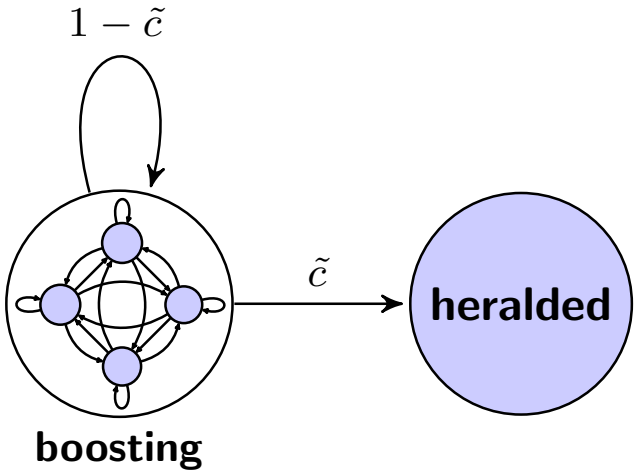

Figure 6.4: Markov model of the NFP (nested feedback protocol) for real-time heralding. This model is an extension of the model introduced for MB. During a boost attempt, transitions occur inside the "boosting" node for further stabilization. The edge leading from the "boosting" to the "heralded" node represents the real-time heralding that selects some fraction of trajectories by the threshold-dependent matrix $\tilde{c}$ (see text) at the end of a boost attempt. Those that are not selected ($\mathbf{1}-\tilde{c}$) enter into another boost attempt.

$$\tilde{c} = \begin{pmatrix} c_{\phi_-} & & & \\ & c_{\phi_+} & & \\ & & c_{gg} & \\ & & & c_{ee} \end{pmatrix} \tag{6.8}$$

After stabilization of some pre-determined duration, if the qubits are (on average) in state $\tilde{S}^{(0)}$, then the average state of the qubits that are heralded is then given by,

$$\tilde{S}^{(0)}_{herald} = \frac{\tilde{c}\tilde{S}^{(0)}}{||\tilde{c}\tilde{S}^{(0)}||_1}\,, \tag{6.9}$$

where $||\cdot||_1$ is the L_1 norm of the vector (the sum of the entries in the vector). The normalization is required since heralding selects only a subset of the trajectories, i.e., the matrix, $\tilde{c}$, does not preserve the norm of $\tilde{S}$.

Each entry on the diagonal of $\tilde{c}$ gives the fraction of trajectories that get heralded (selected) from a particular state. Their values depend on the particular heralding thresholds used, I_{gg}^{herald} and I_{ee}^{herald} (Appendix 6.2). This matrix can be determined phenomenologically by the post-selection experiment shown in Fig. 3 (main text). From the experiment, we can find the average state without any conditioning (no heralding), $\tilde{S}^{(0)} = (\pi_-^{(0)}, \pi_+^{(0)}, \pi_{gg}^{(0)}, \pi_{ee}^{(0)})^{\mathrm{T}}$ and the average state of the heralded trajectories, $\tilde{S}_{herald}^{(0)} = (\pi'_-, \pi'_+, \pi'_{gg}, \pi'_{ee})^{\mathrm{T}}$. Given the success probability P_s of using the thresholds (the white dashed contour line of Fig. 3b,d in the main text), the diagonal elements can be calculated as,

$$c_{\phi_-} = \frac{\pi'_- P_s}{\pi_-^{(0)}}, \quad c_{\phi_+} = \frac{\pi'_+ P_s}{\pi_+^{(0)}}, \quad c_{gg} = \frac{\pi'_{gg} P_s}{\pi_{gg}^{(0)}}, \quad c_{ee} = \frac{\pi'_{ee} P_s}{\pi_{ee}^{(0)}} \tag{6.10}$$

For the specific heralding thresholds used in the experiment (represented by the black squares in Fig. 3b,d in the main text), $\tilde{c}_{\mathrm{DD}}$ and $\tilde{c}_{\mathrm{MB}}$ are explicitly given by,

$$\tilde{c}_{\mathrm{DD}} = \begin{pmatrix} 0.26 & & & \\ & 0.20 & & \\ & & 0.19 & \\ & & & 0.18 \end{pmatrix}, \quad \tilde{c}_{\mathrm{MB}} = \begin{pmatrix} 0.68 & & & \\ & 0.69 & & \\ & & 0.19 & \\ & & & 0.10 \end{pmatrix} \tag{6.11}$$

Ideally for $\tilde{c}$, only c_{ϕ_-} should be non-zero. But in practice, since we cannot distinguish $|\phi_-\rangle$ and $|\phi_+\rangle$, c_{ϕ_+} is comparable to c_{ϕ_-}. Furthermore, for both DD and MB, c_{gg} and c_{ee} are also non-neglibile. In DD, this is predominantly due to the lack of separation between the even and odd measurement outcomes as discussed in Appendix 6.2. In MB, the qubits can jump during the delay between the completion of the quasi-parity measurement and the end of a correction step due to T_1 events. Thus for MB, trajectories that are heralded by very stringent thresholds still have a non-zero probability of being in the even parity states.

Given the heralding matrices $\tilde{c}_{\mathrm{DD}}$ and $\tilde{c}_{\mathrm{MB}}$, we can now calculate the average state of the trajectories that are not heralded and thus require a boost attempt as

$$\tilde{s}^{(0)}_{boost} = (\boldsymbol{I} - \tilde{c})\tilde{S}^{(0)}\,, \quad \tilde{S}^{(0)}_{boost} = \frac{\tilde{s}^{(0)}_{boost}}{||\cdot||_1} \tag{6.12}$$

where we introduce the lowercase $\tilde{s}^{(0)}_{boost}$ as the unnormalized population distribution vector. $||\cdot||_1$ denotes the L_1 norm of the numerator.

After this boost attempt, the qubits are in state $\tilde{S}^{(1)}$,

$$\tilde{S}^{(1)} = \frac{\mathcal{T}\tilde{s}^{(0)}_{boost}}{||\cdot||_1} = \frac{\mathcal{T}(\boldsymbol{I} - \tilde{c})\tilde{S}^{(0)}}{||\cdot||_1} \tag{6.13}$$

where $\mathcal{T}$ is the stochastic transition matrix that models the stabilization during a boosting attempt. In Appendix 6.4, we have already found $\mathcal{T}$ for MB. By the same method, we can also derive the effective transition matrix of a boost attempt for DD. The calculation of $\mathcal{T}$ for DD is an approximation: due to the continuous cavity drives, the state of the qubits at the beginning of a boost attempt is entangled with a qubit-state dependent cavity state, which we approximate unconditionally by the average steady-state cavity state in DD. Nonetheless, as we shall show, the model still produces a quantitative behavior that agrees very well with the experimental results. From the above equations, it is easy to show that the average qubits state of the heralded trajectories after k boost attempts is given by

$$\tilde{S}^{(k)}_{herald} = \frac{\tilde{c}\left(\mathcal{T}(\boldsymbol{I} - \tilde{c})\right)^k \tilde{S}^{(0)}}{||\cdot||_1} \tag{6.14}$$

In the case of a pre-stabilization of sufficient number of correction steps (or duration), $\tilde{S}^{(0)}$ is given by the steady-state, $\tilde{S}_\infty$, introduced in Appendix 6.4.

For each of the k boost attempts, the first entry in the vector $\tilde{S}^{(k)}_{herald}$, i.e., π'_-, gives the fidelity to $|\phi_-\rangle$. The L_1 norm of the numerator in the expression for $\tilde{S}^{(k)}_{herald}$ gives the percentage of trajectories that have completed k boost attempts.

Summing the percentages over all values of k from 0 to the maximum limit gives the overall success probability. In Fig. 6.5, we show the results of the simulation using the model described here and compare them to the experimental results presented in Fig. 4 of the main text. Furthermore, from the simulation, we find that with the realistic system parameter improvement as specified in Appendix 6.4, the fidelity of heralded trajectories can be 90% and 95% for DD and MB, respectively, with order unity success probability.

6.6 Measurement-induced AC stark shift and correction for MB

The quasi-parity measurement induces a deterministic phase shift between the two qubits due to measurement-induced AC Stark shift [Tornberg and Johansson, 2010]. This is evidenced by examining the phase of the Bell state created in the measurement optimization experiment described in Appendix 6.1, in which we varied the measurement duration. As the measurement duration is increased, the Bell angle of the final Bell state changes linearly (Fig. 6.6a). In order for MB to work, we need to account for the deterministic phase shift induced by the measurement. This correction is accomplished by a "Z" rotation on Bob before the unitary U_O. Fig. 6.6b gives one example of a sequence trajectory to illustrate how the correction works. With no loss of generality (and the reason will become clear soon in the discussion that follows), we construct $U_E = R_x^a(\pi/2) \otimes R_{-\phi_o}^b(\pi/2)$ and $U_O = R_x^a(\pi/2) \otimes R_{\phi_o}^b(\pi/2)$ (where ϕ_o = 0 corresponds to the X axis) such that $|\phi_o\rangle = |ge\rangle + e^{i\phi_o}|eg\rangle$ is the eigenstate of U_O and applying U_E on the even states results in $|\phi_o\rangle$ with 50% probability after the quasi-parity measurement. Suppose that the qubits are in the ground states, U_E is applied and the subsequent quasi-parity measurement gives $\tilde{p} = -1$. During the quasi-parity mea-

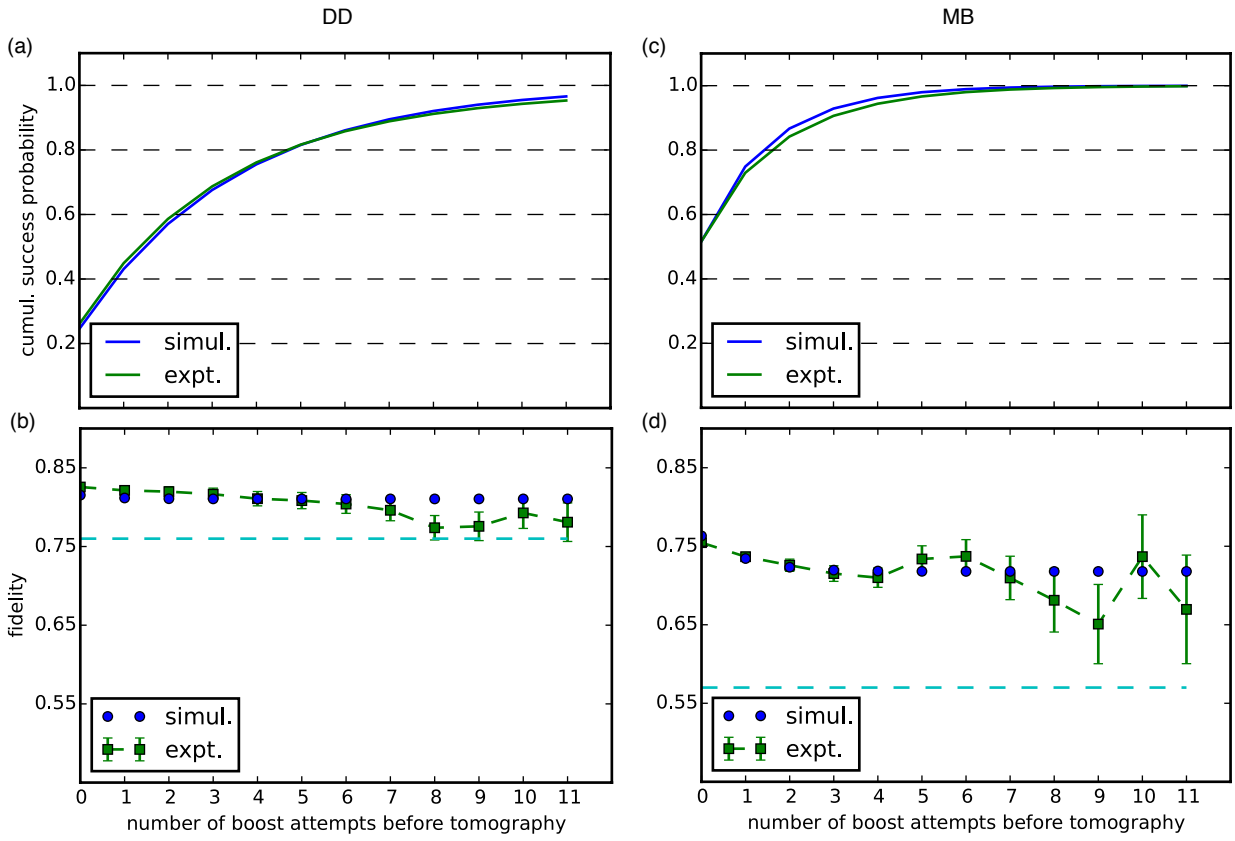

Figure 6.5: Simulation of real-time heralding by NFP. The left (right) panel presents the results for DD (MB). (a), (c) Cumulative success probability of having completed at most a given number of boost attempts before tomography for DD and MB, respectively. Green curve shows the experimental result as presented in the main text. Blue curve is the simulation result using the same experimental parameters. (b), (d) Fidelity to $|\phi_-\rangle$ for DD and MB, respectively. Cyan dashed line denotes the unconditioned steady state fidelity obtained in the experiment. Green squares (blue circles) show the corresponding fidelity as a function of the number of boost attempts during NFP obtained in the experiment (simulation).

surement, a deterministic phase shift of ϕ_D is added. Since the measurement reports odd, the next conditional unitary is U_O. The Z gate before U_O undoes the phase shift and recovers the eigenstate which U_O leaves unchanged. After another parity measurement, the qubits are in the state $|ge\rangle + e^{i(\phi_o+\phi_D)}|eg\rangle$, with the phase shift added again. Consequently, we can see that the MB sequence actually stabilizes $|ge\rangle + e^{i(\phi_o+\phi_D)}|eg\rangle$ state. In practice, for our experiment, the Z rotation for Bob is constructed from a composite of X and Y rotations, such that $R_z^b(\theta) = R_x(\frac{\pi}{2})R_y(\theta)R_x(-\frac{\pi}{2})$, and the effective correction angle θ is swept (Fig. 6.6c) to find the optimal value that cancels the deterministic phase shift and thus maximizes the fidelity. Furthermore, to make the target state of the stabilization $|\phi_-\rangle$, the rotation axis of the pulses on Bob in U_O and U_E is chosen such that $\phi_o+\phi_D = \pi$. This correction for measurement-induced AC Stark shift is also done in the simulation for MB.

6.7 *f* state measurement during NFP

While we have been treating our two qubits as purely two-level systems, in reality there are higher energy levels, in particular the second excited level is expected to play a non-negligible role in the dynamics. We find that the equilibrium qubit population not in the $|gg\rangle$ state was about 15%, and the f state was also populated. To investigate whether the decrease of the fidelity in NFP as a function of boost attempts number is due to the role played by the f-state population, we measured the populations in $|fg\rangle$, $|fe\rangle$, $|gf\rangle$, $|ef\rangle$, $|ff\rangle$ after a given number of boost attempts in DD. The population in $|fg\rangle$ was measured by applying a π pulse on Alice's *e*-*f* transition, then a π pulse on its *e*-*g* transition, followed by a measurement of the population in $|gg\rangle$. The other f-state populations are similarly obtained. The sum of these 5 populations gives the total f-state population plotted in Fig. 6.7, which indeed increases as a function of boost attempt number. It is therefore plausible

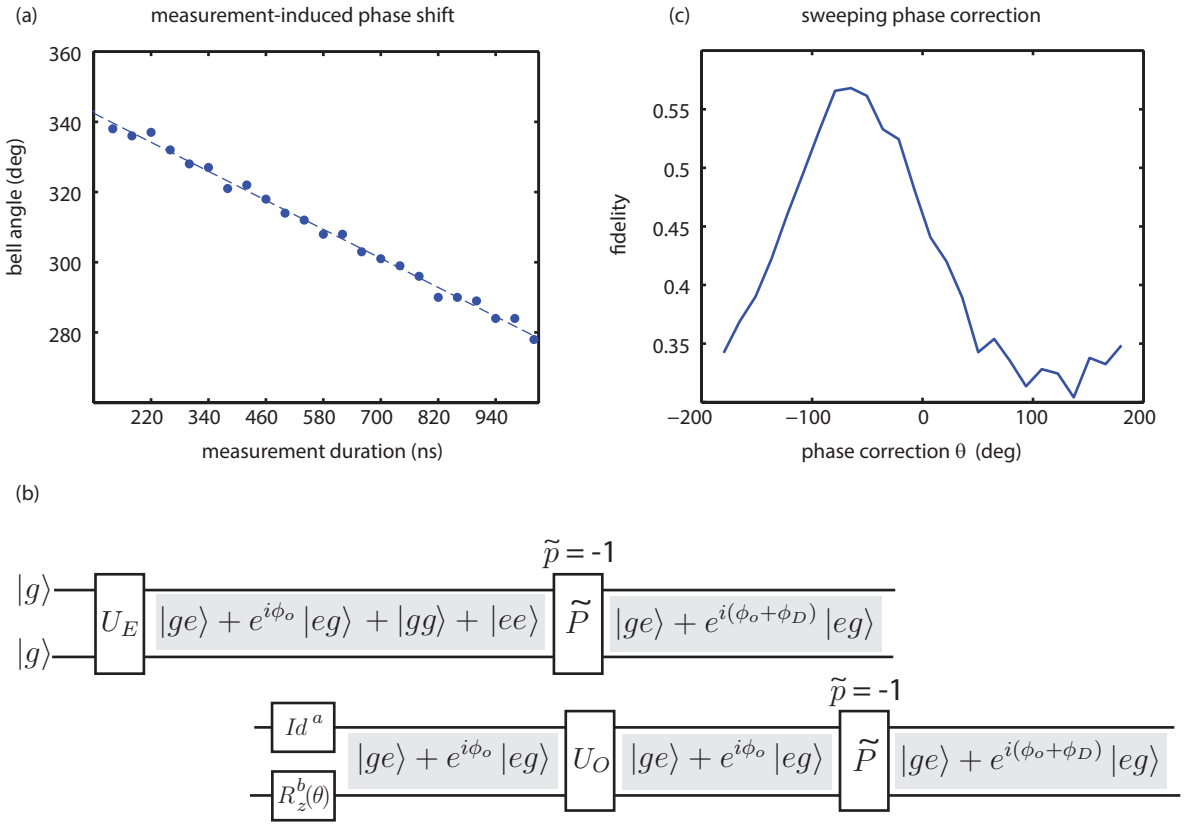

Figure 6.6: Correcting for deterministic measurement-induced AC Stark shift (a) Deterministic phase shift induced by measurement as a function of measurement duration. (b) One example of a sequence trajectory illustrating the phase correction $R_z^b(\theta)$ at work. See detailed explanation in accompanying text. (c) Fidelity of the steady state to the target Bell state as a function of the correction angle of the effective Z gate on Bob.

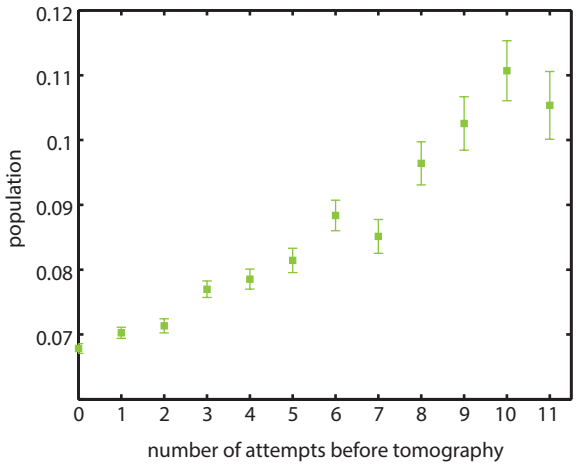

Figure 6.7: Experimental measurement of f state population plotted as a function of the boost attempts number in NFP for DD.

that the decrease in fidelity with respect to number of boost attempts is due to the system being trapped in other levels outside the correction space. This problem, which is common in similar atomic physics experiments, could be addressed by additional drives.

APPENDIX **A**

Master sequence instruction memory table

Each master instruction prescribes the updates for the arbitrary boolean function, the counters, and more. We shall go through all the major fields of an instruction, many of which we have covered already.:

`address(0), (1)`: these are the two possible master instruction addresses that we can branch to.

`branch type`: see the branch type table (Table 3.1).

`est instr add`: the master sequencer supplies this state estimation instruction address to the state estimator to access the instruction at the corresponding location.

`trigger length, trigger level`: the master sequencer is responsible for generating the digital sequence, the 1-bit meta signal data_valid that sets the desired sampling windows, mentioned in Sec. 3.2.1. This digital sequence is named "trigger" to be consistent with similar naming practice by the Alazar data acquisition card. A trigger pulse is high (low) for duration specified by **`trigger length`** if the 1-bit field **`trigger level`** is "1" ("0").

`counter value(0), (1)`: these fields store the initial values for the two counters, respectively, from which we decrement.

`load counter(0),(1), decrement(0), (1)`: whenever **`load counter`**

is "1", the corresponding counter is initialized to the value specified in **counter value**. The counter is decremented by 1 during an instruction if **decrement** is "1".

registers instruction: this field specifies operations on the four *general-purpose registers*. The formatting of this field is very similar to standard assembly language. An operation typically involves two operands. These two operands can be two of the four *general-purpose registers*. In this case, they are selected by the 2-bit fields, **target register** and **source register**, respectively, e.g., 00 selects register 0, 10 selects register 2 and etc. The result is always assigned back to the target register. One of the operands can also be a constant. In such case, the constant number is stored in the field **immediate value**. We have just given examples of two types of register operation. The type of operation is specified in the field **operation code**.

internal function, variable selection internal, load internal function: these fields affect the calculation of **internal result**. In any given instruction, **internal result** can depend on up to four of the eight *bare states*. The selection of each of the four *bare states* is determined by the 2-bit selectors **variable 0, 1, 2, 3**, respectively from the field **var sel int**, i.e., 00 selects **bare state** 0, 11 selects **bare state** 3 and so on. **internal function** is a boolean function of the four selected *bare states*. Therefore it has $2^4 = 16$ bits. The calculation for **internal result** as specified by these fields can persist for any number of instructions, i.e., several measurements. A new boolean function only takes into effect when **load int func.** is set to "1".

external function(0), (1), variable selection external, load external function: we have talked about **internal result** which is the global state used by the FPGA internally. There are also two **external results**. They are generated in exactly the same fashion as **internal result** but they

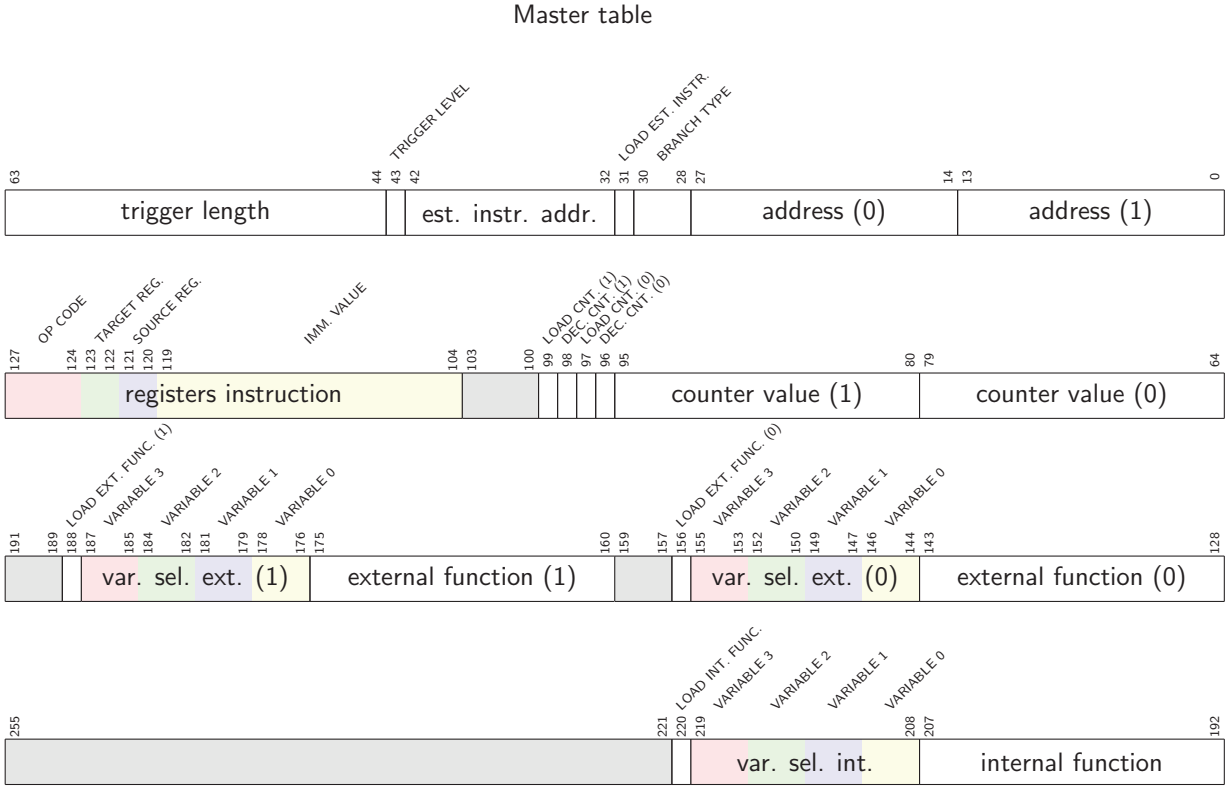

Figure A.1: Instruction memory of the master sequencer.

are exported to outside the FPGA, to be used by either another FPGA (as $\boldsymbol{X_0}$ or $\boldsymbol{X_1}$ or another instrument.

Bibliography

[Alazar] Alazar. URL http://www.alazartech.com/products/ats9870.htm.

[Barends et al., 2014] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. OMalley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500–503, 04 2014. URL http://dx.doi.org/10.1038/nature13171.

[Bennett, 1987] C. H. Bennett. Demons, engines and the second law. *Scientific American*, 257(5):108–116, 1987.

[Bergeal et al., 2010] N. Bergeal, F. Schackert, M. Metcalfe, R. Vijay, V. E. Manucharyan, L. Frunzio, D. E. Prober, R. J. Schoelkopf, S. M. Girvin, and M. H. Devoret. Phase-preserving amplification near the quantum limit with a Josephson ring modulator. *Nature*, 465(7294):64–68, 2010. ISSN 0028-0836. URL http://dx.doi.org/10.1038/nature09035.

[Bernien et al., 2013] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. Blok, L. Robledo, T. Taminiau, M. Markham, D. Twitchen, L. Childress, et al. Heralded entanglement between solid-state qubits separated by three metres. *Nature*, 497(7447): 86–90, 2013.

[Blais et al., 2004] A. Blais, R.-S. Huang, A. Wallraff, S. M. Girvin, and R. J. Schoelkopf.

Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation. *Phys. Rev. A*, 69:062320, 2004. doi: 10.1103/PhysRevA.69.062320. URL http://link.aps.org/doi/10.1103/PhysRevA.69.062320.

[Brecht, 2012] T. Brecht. Advances in data aquisition and digital processing for superconducting qubit measurements. SI talk, January 2012.

[Campagne-Ibarcq et al., 2013] P. Campagne-Ibarcq, E. Flurin, N. Roch, D. Darson, P. Morfin, M. Mirrahimi, M. H. Devoret, F. Mallet, and B. Huard. Persistent control of a superconducting qubit by stroboscopic measurement feedback. *Phys. Rev. X*, 3:021008, 2013. doi: 10.1103/PhysRevX.3.021008. URL http://link.aps.org/doi/10.1103/PhysRevX.3.021008.

[Castellanos-Beltran et al., 2008] M. A. Castellanos-Beltran, K. D. Irwin, G. C. Hilton, L. R. Vale, and K. W. Lehnert. Amplification and squeezing of quantum noise with a tunable Josephson metamaterial. *Nat. Phys.*, 4(12):929–931, Dec. 2008. ISSN 1745-2473. URL http://dx.doi.org/10.1038/nphys1090.

[Chiaverini et al., 2004] J. Chiaverini, D. Leibfried, T. Schaetz, M. Barrett, R. Blakestad, J. Britton, W. Itano, J. Jost, E. Knill, C. Langer, et al. Realization of quantum error correction. *Nature*, 432(7017):602–605, 2004.

[Chow et al., 2010] J. M. Chow, L. DiCarlo, J. M. Gambetta, A. Nunnenkamp, L. S. Bishop, L. Frunzio, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Detecting highly entangled states with a joint qubit readout. *Phys. Rev. A*, 81:062325, 2010. doi: 10.1103/PhysRevA.81.062325. URL http://link.aps.org/doi/10.1103/PhysRevA.81.062325.

[Chow et al., 2012] J. M. Chow, J. M. Gambetta, A. Córcoles, S. T. Merkel, J. A. Smolin, C. Rigetti, S. Poletto, G. A. Keefe, M. B. Rothwell, J. Rozen, et al. Universal

quantum gate set approaching fault-tolerant thresholds with superconducting qubits. *Physical review letters*, 109(6):060501, 2012.

[Chow et al., 2014] J. M. Chow, J. M. Gambetta, E. Magesan, D. W. Abraham, A. W. Cross, B. R. Johnson, N. A. Masluk, C. A. Ryan, J. A. Smolin, S. J. Srinivasan, and M. Steffen. Implementing a strand of a scalable fault-tolerant quantum computing fabric. *Nat. Commun.*, 5(4015), 06 2014. URL http://dx.doi.org/10.1038/ncomms5015.

[Chow et al., 1971] Y. S. Chow, H. Robbins, and D. Siegmund. *Great expectations: The theory of optimal stopping*. Houghton Mifflin Boston, 1971.

[Chu, 2011] P. P. Chu. *FPGA prototyping by VHDL examples: Xilinx Spartan-3 version*. John Wiley & Sons, 2011.

[Córcoles et al., 2015] A. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow. Demonstration of a quantum error detection code using a square lattice of four superconducting qubits. *Nat. Commun.*, 6, 2015.

[Cory et al., 1998] D. G. Cory, M. Price, W. Maas, E. Knill, R. Laflamme, W. H. Zurek, T. F. Havel, and S. Somaroo. Experimental quantum error correction. *Physical Review Letters*, 81(10):2152, 1998.

[Devoret and Schoelkopf, 2013] M. H. Devoret and R. J. Schoelkopf. Superconducting circuits for quantum information: An outlook. *Science*, 339(6124):1169–1174, 2013.

[DiCarlo et al., 2009] L. DiCarlo, J. M. Chow, J. M. Gambetta, L. S. Bishop, B. R. Johnson, D. I. Schuster, J. Majer, A. Blais, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. Demonstration of two-qubit algorithms with a superconducting

quantum processor. *Nature*, 460(7252):240–244, 2009. ISSN 0028-0836. URL http://dx.doi.org/10.1038/nature08121.

[DiVincenzo, 2000] D. P. DiVincenzo. The physical implementation of quantum computation. *Fortschr. Phys.*, 48:771, 2000.

[Doherty et al., 2000] A. C. Doherty, S. Habib, K. Jacobs, H. Mabuchi, and S. M. Tan. Quantum feedback control and classical control theory. *Physical Review A*, 62(1): 012105, 2000.

[Electronics] Electronics. URL http://web.physics.ucsb.edu/~martinisgroup/electronics.shtml.

[Filipp et al., 2009] S. Filipp, P. Maurer, P. J. Leek, M. Baur, R. Bianchetti, J. M. Fink, M. Göppl, L. Steffen, J. M. Gambetta, A. Blais, and A. Wallraff. Two-qubit state tomography using a joint dispersive readout. *Phys. Rev. Lett.*, 102:200402, 2009. doi: 10.1103/PhysRevLett.102.200402. URL http://link.aps.org/doi/10.1103/PhysRevLett.102.200402.

[Fowler et al., 2012] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, 2012. doi: 10.1103/PhysRevA.86.032324. URL http://link.aps.org/doi/10.1103/PhysRevA.86.032324.

[Fujii et al., 2014] K. Fujii, M. Negoro, N. Imoto, and M. Kitagawa. Measurement-free topological protection using dissipative feedback. *Phys. Rev. X*, 4:041039, 2014. doi: 10.1103/PhysRevX.4.041039. URL http://link.aps.org/doi/10.1103/PhysRevX.4.041039.

[Gambetta et al., 2007] J. Gambetta, W. A. Braff, A. Wallraff, S. M. Girvin, and R. J. Schoelkopf. Protocols for optimal readout of qubits using a continuous quantum nondemolition measurement. *Phys. Rev. A*, 76:012325, 2007. doi: 10.

1103/PhysRevA.76.012325. URL http://link.aps.org/doi/10.1103/PhysRevA.76.012325.

[Geerlings et al., 2013] K. Geerlings, Z. Leghtas, I. M. Pop, S. Shankar, L. Frunzio, R. J. Schoelkopf, M. Mirrahimi, and M. H. Devoret. Demonstrating a driven reset protocol for a superconducting qubit. *Phys. Rev. Lett.*, 110:120501, 2013. doi: 10.1103/PhysRevLett.110.120501. URL http://link.aps.org/doi/10.1103/PhysRevLett.110.120501.

[Hamerly and Mabuchi, 2012] R. Hamerly and H. Mabuchi. Advantages of coherent feedback for cooling quantum oscillators. *Physical review letters*, 109(17):173602, 2012.

[Harris and Harris, 2012] D. Harris and S. Harris. *Digital design and computer architecture*. Elsevier, 2012.

[Hastie et al., 2005] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[Hatridge et al., 2013] M. Hatridge, S. Shankar, M. Mirrahimi, F. Schackert, K. Geerlings, T. Brecht, K. M. Sliwa, B. Abdo, L. Frunzio, S. M. Girvin, R. J. Schoelkopf, and M. H. Devoret. Quantum back-action of an individual variable-strength measurement. *Science*, 339:178, 2013. URL http://dx.doi.org/10.1126/science.1226897.

[Hofmann et al., 2012] J. Hofmann, M. Krug, N. Ortegel, L. Gérard, M. Weber, W. Rosenfeld, and H. Weinfurter. Heralded entanglement between widely separated atoms. *Science*, 337(6090):72–75, 2012.

[Holland et al., 2015] E. Holland, B. Vlastakis, R. Heeres, M. Reagor, U. Vool, Z. Leghtas, L. Frunzio, G. Kirchmair, M. Devoret, M. Mirrahimi, et al. Single-photon

resolved cross-kerr interaction for autonomous stabilization of photon-number states. *arXiv:1504.03382*, 2015.

[Integration, 2015a] I. Integration. X6-1000m framework logic user guide. 2015a.

[Integration, 2015b] I. Integration. X6-1000m manual. 2015b.

[Jacobs et al., 2014] K. Jacobs, X. Wang, and H. M. Wiseman. Coherent feedback that beats all measurement-based feedback protocols. *New Journal of Physics*, 16(7): 073036, 2014.

[Johnson et al., 2012] J. E. Johnson, C. Macklin, D. H. Slichter, R. Vijay, E. B. Weingarten, J. Clarke, and I. Siddiqi. Heralded state preparation in a superconducting qubit. *Phys. Rev. Lett.*, 109:050506, 2012. doi: 10.1103/PhysRevLett.109.050506. URL http://link.aps.org/doi/10.1103/PhysRevLett.109.050506.

[Kapit et al., 2015] E. Kapit, J. T. Chalker, and S. H. Simon. Passive correction of quantum logical errors in a driven, dissipative system: A blueprint for an analog quantum code fabric. *Phys. Rev. A*, 91:062324, 2015. doi: 10.1103/PhysRevA.91.062324. URL http://link.aps.org/doi/10.1103/PhysRevA.91.062324.

[Kelly] O. Kelly. URL https://www.opalkelly.com/products/xem5010/.

[Kerckhoff et al., 2010] J. Kerckhoff, H. I. Nurdin, D. S. Pavlichin, and H. Mabuchi. Designing quantum memories with embedded control: Photonic circuits for autonomous quantum error correction. *Phys. Rev. Lett.*, 105:040502, 2010. doi: 10.1103/PhysRevLett.105.040502. URL http://link.aps.org/doi/10.1103/PhysRevLett.105.040502.

[Kerckhoff et al., 2013] J. Kerckhoff, R. W. Andrews, H. S. Ku, W. F. Kindel, K. Cicak, R. W. Simmonds, and K. W. Lehnert. Tunable coupling to a mechanical oscillator

circuit using a coherent feedback network. *Phys. Rev. X*, 3:021013, 2013. doi: 10.1103/PhysRevX.3.021013. URL http://link.aps.org/doi/10.1103/PhysRevX.3.021013.

[Kienzler et al., 2015] D. Kienzler, H.-Y. Lo, B. Keitch, L. de Clercq, F. Leupold, F. Lindenfelser, M. Marinelli, V. Negnevitsky, and J. Home. Quantum harmonic oscillator state synthesis by reservoir engineering. *Science*, 347(6217):53–56, 2015.

[Koch et al., 2007] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the Cooper pair box. *Phys. Rev. A*, 76:042319, 2007. doi: 10.1103/PhysRevA.76.042319. URL http://link.aps.org/doi/10.1103/PhysRevA.76.042319.

[Krauter et al., 2011] H. Krauter, C. A. Muschik, K. Jensen, W. Wasilewski, J. M. Petersen, J. I. Cirac, and E. S. Polzik. Entanglement generated by dissipation and steady state entanglement of two macroscopic objects. *Phys. Rev. Lett.*, 107: 080503, 2011. doi: 10.1103/PhysRevLett.107.080503. URL http://link.aps.org/doi/10.1103/PhysRevLett.107.080503.

[Lalumière et al., 2010] K. Lalumière, J. M. Gambetta, and A. Blais. Tunable joint measurements in the dispersive regime of cavity QED. *Phys. Rev. A*, 81:040301, 2010. doi: 10.1103/PhysRevA.81.040301. URL http://link.aps.org/doi/10.1103/PhysRevA.81.040301.

[Landauer, 1961] R. Landauer. Irreversibility and heat generation in the computing process. *IBM journal of research and development*, 5(3):183–191, 1961.

[Leghtas et al., 2013] Z. Leghtas, U. Vool, S. Shankar, M. Hatridge, S. M. Girvin, M. H. Devoret, and M. Mirrahimi. Stabilizing a bell state of two superconducting qubits by dissipation engineering. *Phys. Rev. A*, 88:023849, 2013. doi: 10.

1103/PhysRevA.88.023849. URL http://link.aps.org/doi/10.1103/PhysRevA.88.023849.

[Leghtas et al., 2015] Z. Leghtas, S. Touzard, I. M. Pop, A. Kou, B. Vlastakis, A. Petrenko, K. M. Sliwa, A. Narla, S. Shankar, M. J. Hatridge, M. Reagor, L. Frunzio, R. J. Schoelkopf, M. Mirrahimi, and M. H. Devoret. Confining the state of light to a quantum manifold by engineered two-photon loss. *Science*, 347(6224):853–857, 2015.

[Lin et al., 2013] Y. Lin, J. Gaebler, F. Reiter, T. R. Tan, R. Bowler, A. Sørensen, D. Leibfried, and D. Wineland. Dissipative production of a maximally entangled steady state of two quantum bits. *Nature*, 504(7480):415–418, 2013.

[Lloyd, 2000] S. Lloyd. Coherent quantum feedback. *Physical Review A*, 62(2):022108, 2000.

[Mabuchi, 2008] H. Mabuchi. Coherent-feedback quantum control with a dynamic compensator. *Physical Review A*, 78(3):032323, 2008.

[Magesan et al., 2015] E. Magesan, J. M. Gambetta, A. D. Córcoles, and J. M. Chow. Machine learning for discriminating quantum measurement trajectories and improving readout. *Phys. Rev. Lett.*, 114:200501, 2015. doi: 10.1103/PhysRevLett.114.200501. URL http://link.aps.org/doi/10.1103/PhysRevLett.114.200501.

[Maxwell, 1867] J. C. Maxwell. On governors. *Proceedings of the Royal Society of London*, 16:270–283, 1867.

[Mirrahimi et al., 2014] M. Mirrahimi, Z. Leghtas, V. V. Albert, S. Touzard, R. J. Schoelkopf, L. Jiang, and M. H. Devoret. Dynamically protected cat-qubits: a new paradigm for universal quantum computation. *New Journal of Physics*, 16

(4):045014, 2014. URL http://stacks.iop.org/1367-2630/16/i=4/a=045014.

[Moehring et al., 2007] D. Moehring, P. Maunz, S. Olmschenk, K. Younge, D. Matsukevich, L.-M. Duan, and C. Monroe. Entanglement of single-atom quantum bits at a distance. *Nature*, 449(7158):68–71, 2007.

[Murch et al., 2012] K. W. Murch, U. Vool, D. Zhou, S. J. Weber, S. M. Girvin, and I. Siddiqi. Cavity-assisted quantum bath engineering. *Phys. Rev. Lett.*, 109:183602, 2012. doi: 10.1103/PhysRevLett.109.183602. URL http://link.aps.org/doi/10.1103/PhysRevLett.109.183602.

[Nielsen and Chuang, 2004] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2004.

[Nigg et al., 2014] D. Nigg, M. Müller, E. Martinez, P. Schindler, M. Hennrich, T. Monz, M. Martin-Delgado, and R. Blatt. Quantum computations on a topologically encoded qubit. *Science*, 345(6194):302–305, 2014.

[Nurdin et al., 2009] H. I. Nurdin, M. R. James, and I. R. Petersen. Coherent quantum {LQG} control. *Automatica*, 45(8):1837 – 1846, 2009. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/j.automatica.2009.04.018. URL http://www.sciencedirect.com/science/article/pii/S0005109809002179.

[Paik et al., 2011] H. Paik, D. I. Schuster, L. S. Bishop, G. Kirchmair, G. Catelani, A. P. Sears, B. R. Johnson, M. J. Reagor, L. Frunzio, L. I. Glazman, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf. Observation of high coherence in Josephson junction qubits measured in a three-dimensional circuit QED architecture. *Phys. Rev. Lett.*, 107:240501, 2011. doi: 10.1103/PhysRevLett.107.240501. URL http://link.aps.org/doi/10.1103/PhysRevLett.107.240501.

[Poyatos et al., 1996] J. F. Poyatos, J. I. Cirac, and P. Zoller. Quantum reservoir engineering with laser cooled trapped ions. *Phys. Rev. Lett.*, 77:4728–4731, 1996. doi: 10.1103/PhysRevLett.77.4728. URL http://link.aps.org/doi/10.1103/PhysRevLett.77.4728.

[Reagor et al., 2013] M. Reagor, H. Paik, G. Catelani, L. Sun, C. Axline, E. Holland, I. M. Pop, N. A. Masluk, T. Brecht, L. Frunzio, et al. Reaching 10 ms single photon lifetimes for superconducting aluminum cavities. *Applied Physics Letters*, 102 (19):192604, 2013.

[Reagor et al., 2015] M. Reagor, W. Pfaff, C. Axline, R. W. Heeres, N. Ofek, K. Sliwa, E. Holland, C. Wang, J. Blumoff, K. Chou, et al. A quantum memory with near-millisecond coherence in circuit qed. *arXiv preprint arXiv:1508.05882*, 2015.

[Reed et al., 2012] M. D. Reed, L. DiCarlo, S. E. Nigg, L. Sun, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. Realization of three-qubit quantum error correction with superconducting circuits. *Nature*, 482(7385):382–385, 2012. ISSN 0028-0836. URL http://dx.doi.org/10.1038/nature10786.

[Ristè et al., 2012a] D. Ristè, C. C. Bultink, K. W. Lehnert, and L. DiCarlo. Feedback control of a solid-state qubit using high-fidelity projective measurement. *Phys. Rev. Lett.*, 109:240502, 2012a. doi: 10.1103/PhysRevLett.109.240502. URL http://link.aps.org/doi/10.1103/PhysRevLett.109.240502.

[Ristè et al., 2012b] D. Ristè, J. G. van Leeuwen, H.-S. Ku, K. W. Lehnert, and L. DiCarlo. Initialization by measurement of a superconducting quantum bit circuit. *Phys. Rev. Lett.*, 109:050507, 2012b. doi: 10.1103/PhysRevLett.109.050507. URL http://link.aps.org/doi/10.1103/PhysRevLett.109.050507.

[Riste et al., 2013] D. Riste, M. Dukalski, C. A. Watson, G. de Lange, M. J. Tiggelman, Y. M. Blanter, K. W. Lehnert, R. N. Schouten, and L. DiCarlo. Deterministic

entanglement of superconducting qubits by parity measurement and feedback. *Nature*, 502(7471):350–354, 2013.

[Ristè et al., 2014] D. Ristè, S. Poletto, M.-Z. Huang, A. Bruno, V. Vesterinen, O.-P. Saira, and L. DiCarlo. Detecting bit-flip errors in a logical qubit using stabilizer measurements. *Nat. Commun.*, 6(6983), 2014. URL http://arxiv.org/abs/1411.5542.

[Sayrin et al., 2011] C. Sayrin, I. Dotsenko, X. Zhou, B. Peaudecerf, T. Rybarczyk, S. Gleyzes, P. Rouchon, M. Mirrahimi, H. Amini, M. Brune, J.-M. Raimond, and S. Haroche. Real-time quantum feedback prepares and stabilizes photon number states. *Nature*, 477(7362):73–77, 09 2011. URL http://dx.doi.org/10.1038/nature10376.

[Schindler et al., 2011] P. Schindler, J. T. Barreiro, T. Monz, V. Nebendahl, D. Nigg, M. Chwalla, M. Hennrich, and R. Blatt. Experimental repetitive quantum error correction. *Science*, 332(6033):1059–1061, 2011.

[Schreier et al., 2008] J. A. Schreier, A. A. Houck, J. 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*, 77:180502, 2008. doi: 10.1103/PhysRevB.77.180502. URL http://link.aps.org/doi/10.1103/PhysRevB.77.180502.

[Schuster et al., 2007] D. I. Schuster, A. A. Houck, J. A. Schreier, A. Wallraff, J. M. Gambetta, A. Blais, L. Frunzio, J. Majer, B. Johnson, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Resolving photon number states in a superconducting circuit. *Nature*, 445(7127):515–518, 2007. ISSN 0028-0836. URL http://dx.doi.org/10.1038/nature05461.

[Sears et al., 2012] A. P. Sears, A. Petrenko, G. Catelani, L. Sun, H. Paik, G. Kirchmair,

L. Frunzio, L. I. Glazman, S. M. Girvin, and R. J. Schoelkopf. Photon shot noise dephasing in the strong-dispersive limit of circuit QED. *Phys. Rev. B*, 86:180504, 2012. doi: 10.1103/PhysRevB.86.180504. URL http://link.aps.org/doi/10.1103/PhysRevB.86.180504.

[Shankar et al., 2013] S. Shankar, M. Hatridge, Z. Leghtas, K. M. Sliwa, A. Narla, U. Vool, S. M. Girvin, L. Frunzio, M. Mirrahimi, and M. H. Devoret. Autonomously stabilized entanglement between two superconducting quantum bits. *Nature*, 504(7480):419–422, 2013. URL http://dx.doi.org/10.1038/nature12802.

[Steffen et al., 2013] L. Steffen, Y. Salathe, M. Oppliger, P. Kurpiers, M. Baur, C. Lang, C. Eichler, G. Puebla-Hellmann, A. Fedorov, and A. Wallraff. Deterministic quantum teleportation with feed-forward in a solid state system. *Nature*, 500 (7462):319–322, 2013. URL http://dx.doi.org/10.1038/nature12422.

[Stockton et al., 2002] J. Stockton, M. Armen, and H. Mabuchi. Programmable logic devices in experimental quantum optics. *JOSA B*, 19(12):3019–3027, 2002.

[Sun et al., 2014] L. Sun, A. Petrenko, Z. Leghtas, B. Vlastakis, G. Kirchmair, K. M. Sliwa, A. Narla, M. Hatridge, S. Shankar, J. Blumoff, L. Frunzio, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf. Tracking photon jumps with repeated quantum non-demolition parity measurements. *Nature*, 511(7510):444–448, 2014. URL http://dx.doi.org/10.1038/nature13436.

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

[Tex] *An Overview of LVDS Technology*. Texas Instruments, http://www.ti.com/lit/an/snla165/snla165.pdf.

[Tornberg and Johansson, 2010] L. Tornberg and G. Johansson. High-fidelity feedback-assisted parity measurement in circuit QED. *Phys. Rev. A*, 82:012329, 2010. doi: 10.1103/PhysRevA.82.012329. URL http://link.aps.org/doi/10.1103/PhysRevA.82.012329.

[Vijay et al., 2012] R. Vijay, C. Macklin, D. Slichter, S. Weber, K. Murch, R. Naik, A. N. Korotkov, and I. Siddiqi. Stabilizing Rabi oscillations in a superconducting qubit using quantum feedback. *Nature*, 490(7418):77–80, 2012. URL http://dx.doi.org/10.1038/nature11505.

[Wagenknecht et al., 2010] C. Wagenknecht, C.-M. Li, A. Reingruber, X.-H. Bao, A. Goebel, Y.-A. Chen, Q. Zhang, K. Chen, and J.-W. Pan. Experimental demonstration of a heralded entanglement source. *Nature Photonics*, 4(8):549–552, 2010.

[Wiseman and Milburn, 1994] H. M. Wiseman and G. J. Milburn. All-optical versus electro-optical quantum-limited feedback. *Physical review A*, 49(5):4110, 1994.

[Wiseman and Milburn, 2009] H. M. Wiseman and G. J. Milburn. *Quantum Measurement and Control*. Cambridge University Press, 2009.

[Xilinx] Xilinx. Virtex-6 family overview.

[Yamamoto, 2014] N. Yamamoto. Coherent versus measurement feedback: Linear systems theory for quantum information. *Physical Review X*, 4(4):041029, 2014.