Quantum Computer Fault Injection Attacks

Chuanqi Xu
Dept. of Electrical Engineering
Yale University
New Haven, CT, USA
chuanqi.xu@yale.edu

Ferhat Erata
Dept. of Computer Science
Yale University
New Haven, CT, USA
ferhat.erata@yale.edu

Jakub Szefer
Dept. of Electrical Engineering
Yale University
New Haven, CT, USA
jakub.szefer@yale.edu

***Abstract*—The rapid growth of interest in quantum computing has brought about the need to secure these emerging computers against a range of security attacks. Among the potential security threats are physical attacks, including those orchestrated by malicious insiders within data centers where the quantum computers are located, which could compromise the integrity of computations and resulting data. To help in the understanding of emerging fault injection attacks on quantum computers, this paper presents an in-depth exploration of quantum computer fault injection attacks. This work introduces a classification of fault injection attacks and strategies, including an analysis of the domain of fault injection attacks, the fault targets, and the fault manifestations in quantum computers. The resulting classification highlights the landscape of the potential threats, and presents a road map for researchers and industry for developing security protection mechanisms against fault injection attacks for the emerging quantum computing systems.**

***Index Terms*—quantum computing, security, fault-injection**

I. Introduction

Quantum computing has accelerated in development in recent years. Many companies and universities are racing to build bigger and better machines. Among others, IBM unveiled an 1121-qubit quantum computer in late 2023, and 200-qubit IBM quantum computers with the ability to run 100 million gates are anticipated for 2029 [1].

Presently, quantum computers are in the Nosy Intermediate Scale Quantum (NISQ) regime [2], with less than 1000 qubits and no support for quantum error correction [3]. Nevertheless, these machines have the potential to help accelerate many fields such as drug discovery or finding new materials [4]–[6]. With the increase of qubits and improvement in fidelity, it will be possible to gradually move into the fault-tolerant quantum computing regime with techniques like quantum error correction. Optimistically, quantum computers and quantum algorithms promise to be applied to revolutionize many fields, such be enabling execution of Grover's [7] and Shor's algorithms that can be used to break some nowadays widely-used classical cryptographic algorithms like RSA [8].

As quantum computers grow in size, the data and information in the computing process may be sensitive and private. Further, the quantum programs themselves executed on quantum computers are also valuable intellectual properties. Integrity and confidentiality of the data or quantum programs can be compromised if there is a fault injection attack.

A. Comparison of Quantum and Classical Computer Fault Injection Attacks

In classical computer fault injection, the faults mainly target the instructions executing on the processor or the data in registers. It is also possible to inject or cause faults in DRAM memory or on the memory bus or other parts of the system. The classical processor is typically encased in a single package, and in fault injection attacks, the package is exposed to voltage glitching, clock glitching, EM, lasers, or other sources of disturbance, see Section VI for more details.

One main difference in quantum computers is that they are not, yet, self-contained within a tiny chip. Today, there is extensive classical infrastructure outside of the quantum computer that controls the qubits located in the quantum computers. This infrastructure significantly extends the possible attack surface. Given room or server-rack sized quantum computers, easy physical access also gives the opportunity to manipulate the equipment much more easily than today's nanometer-sized transistors in classical computers. Further, there is an opportunity for attackers to either manipulate the qubits, or classical registers into which the qubit measurements are read, or the control signals (either digital signals going into the controller equipment, or analog signals going between controller equipment and the quantum computer itself). This extends the attack surface even more compared to classical computers.

B. Contributions

The contributions of this work are:

- We identify the *domain of quantum computer fault injection attacks*; this domain represents the attack surface that is distinct from classical computers, and at the same time it identifies the hardware and system components that may be subject to the fault injection attacks.
- We pinpoint 3 *fault targets* specific to quantum computers: quantum processing units, quantum computer controller, and classical co-processors; within the three targets, we present further 6 specific components that can be targeted for fault attacks.
- We present *fault model*, *fault bound*, and *fault lifespan* for the different fault targets.

This work was supported in part by National Science Foundation grants 2312754 and 2245344.

- We propose the first classification of quantum computer fault injection attacks to help industry and researchers navigate the security of this emerging technology.

II. Background

This work focuses on superconducting quantum computers, such as those available from IBM, Rigetti, QCI, and others. The typical setup of a superconducting qubit quantum computer is shown in Figure 1. We consider today's cloud-based computers where users connect remotely to the machines. Figure 1 specifically depicts a superconducting qubit quantum computer setup. Other types of quantum computers may have different types of, for example, quantum computer controllers, but the same types of fault injection attacks can be applied.

A. Quantum Computing Basics

Analogous to the classical bit, a quantum bit, or *qubit*, is the fundamental computational unit in quantum computers. A qubit can be represented with the bra-ket representation. With $|0\rangle$ and $|1\rangle$ as the basis states, a qubit can be written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. According to Born's rule, the results of measuring $|\psi\rangle$ is either $|0\rangle$ or $|1\rangle$, with probability $|\alpha|^2$ and $|\beta|^2$ respectively. Such a phenomenon that a qubit can be measured with two results is not seen in classical computing, and it is often called *superposition*. Also, the state after the measurement will *collapse* to the resulting state, no matter what the initial state is. Similarly, an n-qubit system is spanned by 2^n basis states. Surprisingly, some multi-qubit quantum states cannot be described independently by the state of their components, which is another phenomenon that is not shown in classical computing, and this is often referred to *entanglement*. Qubits are controlled and evolved by *quantum gates*, which are the building blocks of quantum circuits, like classical logic gates are for conventional digital circuits. We refer interested readers to [9] for details.

B. Cloud-based Quantum Computers

Due to the expensive nature of quantum computing equipment, quantum computers are currently available as cloud-based systems. For example, cloud-based services such as IBM Quantum [10], Amazon Braket [11], and Azure Quantum [12] already provide access to Noisy Intermediate-Scale Quantum (NISQ) quantum computers remotely for users. In the cloud setting, the user has no control over the management server, quantum computer controllers, and the cryogenic fridge are not under the control of the user. A malicious insider or compromised cloud provider could try to perform fault injection attacks.

As in any cloud-based computing systems, there is a **management server** that is a typical classical server that sits between the users and the quantum controllers and equipment. Management servers for quantum computing commonly handle the receiving of quantum jobs, queuing, and dispatching jobs. Quantum jobs submitted by users are usually first pushed into priority queues, and based on the priority algorithms of the cloud platforms, these jobs wait in the queue, and then the information of jobs is processed and sent to quantum computer controllers after they finish waiting. Fault injection attacks in classical management servers are possible, but they are the domain of classical security, not further considered here.

Quantum programs dispatched from the management server are sent to **quantum computer controllers**. In current quantum computers, each qubit or qubit pair is typically assigned dedicated control pulses with distinct parameter settings, including the pulse waveform, pulse duration, pulse frequency, pulse amplitude, and so on. Control pulses, both microwave and baseband flux, are generated at room temperature by classical equipment such as the arbitrary waveform generator (AWG) and IQ mixers. Then these pulses will be delivered to the qubits in the cryogenic system through a series of attenuators and filters designed to suppress harmful noises when the quantum programs reach the point to run the corresponding gates. These controllers can be sources of novel fault attacks analyzed in this work.

Besides controlling the qubits, one important function of quantum computer controllers is to perform the measurement process and measurement readout results. The results from quantum computers may be stored in the controller and sent back to the management servers when jobs finish. In addition, for advanced features like dynamic circuits [13], it stores the middle-measurement results and controls future operations based on these results. Classical data in the controllers can be vulnerable to fault attacks, as AWGs, IQ mixers, and other similar equipment have not been analyzed from a security perspective before.

An **auxiliary processor** is a classical processor that is part of the quantum computer controller, or tightly coupled to the controller. It may contain user-defined code or application-specific code defining what operations to perform based on the readout data; as well as it can be used to determine what subsequent operations to execute on the quantum computer or to update the circuit executing on the quantum computer. In one example of quantum machine learning (QML) [6], based on the readout data, the co-processor can optimize the parameters of the quantum circuit and issue the next job with the updated circuit, similar to the classical machine learning. The auxiliary processor is critical to the operation of quantum computers and can be the target of fault injection attacks.

In the end, the control pulses actuate the **quantum processing unit**, also called simply the QPU, which contains the actual physical qubits. The QPU is located in the cryogenic fridge, also known as the dilution refrigerator, which is an integral part of superconducting quantum computers. These qubits are sensitive to thermal noise, which is why the frigid environment provided by the dilution refrigerator is crucial. Once the qubits are in their superconducting state, they are manipulated using microwave pulses, generated by quantum computer controllers previously introduced. The pulses are delivered through coaxial cables that are also cooled within the refrigerator to minimize thermal noise. The qubits and the fridge are other new parts not found in classical computers and can be targets of fault injection attacks.

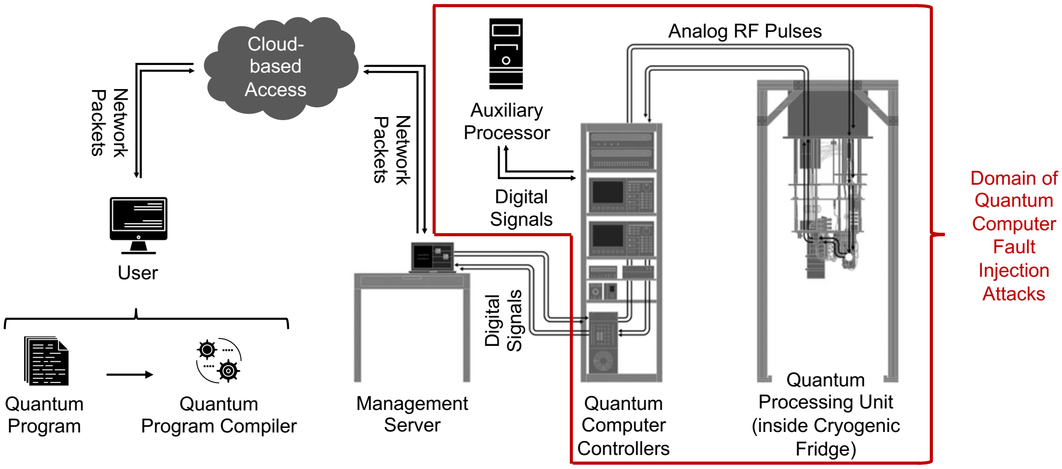

Fig. 1: Typical setup of a superconducting qubit quantum computer, figure is based on figure from IBM.

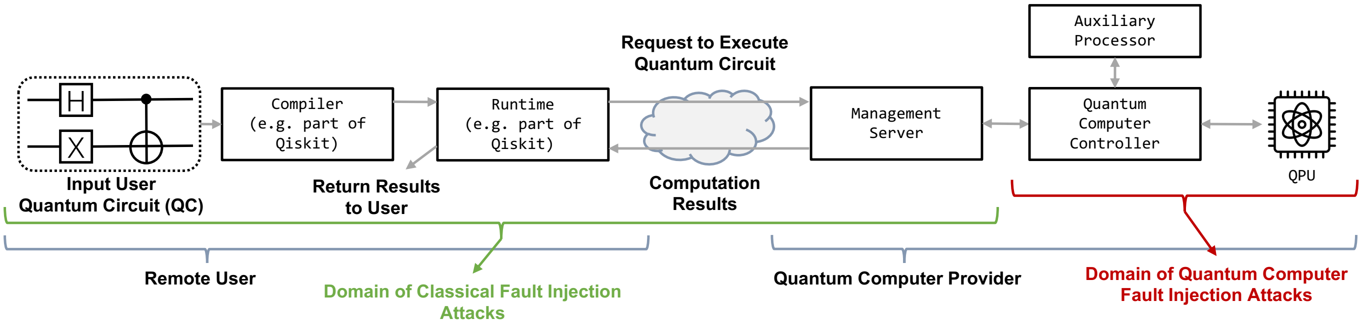
H
X
Input User Quantum Circuit (QC)
Compiler (e.g. part of Qiskit)
Runtime (e.g. part of Qiskit)
Return Results to User
Request to Execute Quantum Circuit
Computation Results
Management Server
Auxiliary Processor
Quantum Computer Controller
QPU
Remote User
Domain of Classical Fault Injection Attacks
Quantum Computer Provider
Domain of Quantum Computer Fault Injection Attacks

Fig. 2: Typical quantum computer workflow.

C. Workflow of Executing Quantum Circuits on a Quantum Computer

The typical workflow of quantum computers is shown in Figure 2. In quantum computing, users can write gate-level programs using quantum programming languages such as Qiskit [14], Amazon Braket SDK [15], or Cirq [16]. These programs consist of sequences of quantum gates that operate on qubits. The programs are then transpiled to decompose the gates into elementary quantum gates supported by the hardware. The transpiler optimizes the program by reducing gate count and improving gate ordering. It also maps logical qubits to the physical qubits available in the hardware, considering connectivity constraints. The next step is *scheduling*, where timing and control information are determined for each gate, specifying the precise microwave pulses required for their execution. When jobs are sent to quantum computer systems and start to execute, microwave electronics generate these pulses, corresponding to signals that manipulate the quantum state of the qubits. The pulses are applied to the physical qubits, implementing the desired gate operations. After execution, the resulting quantum state can be measured to obtain the computation's output. The specific details of the transpilation and scheduling process may vary depending on the programming language, hardware, and software stack used.

III. Fault Manifestation

In the context of fault injection in quantum computing, *fault manifestation* refers to the observable effect or consequence of an injected fault within the quantum system. This could include changes in the state of a qubit, alterations in the operation of a quantum gate, or eventually deviations in the outcome of a quantum algorithm. The study of fault manifestation is crucial in understanding the impact of errors on quantum computations and in developing strategies for error detection and correction. The fault manifestation can be:

A. Gate-level Program

The gate-level quantum circuit is a model used in quantum computing to describe qubit evolution and incidental operations. Computations in the gate-level quantum circuits are represented as a sequence of quantum gates acting on qubits, and other operations such as measurement, reset, and classical operations, from left to right to denote time steps. Each quantum gate, analogous to a logic gate in classical computing, performs a specific unitary operation or transformation on the quantum state of a qubit or a set of qubits. By arranging these gates in specific sequences and combinations, complex quantum algorithms can be implemented.

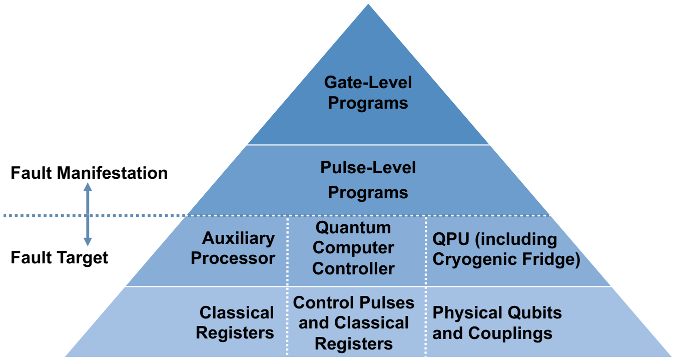
Gate-Level Programs
Pulse-Level Programs
Fault Manifestation
Fault Target
Auxiliary Processor
Quantum Computer Controller
QPU (including Cryogenic Fridge)
Classical Registers
Control Pulses and Classical Registers
Physical Qubits and Couplings

Fig. 3: The fault target and fault manifestation security pyramid for superconducting quantum computers.

B. Pulse-level Program

This is one level lower abstraction of quantum circuits. Since superconducting qubits are controlled by microwave pulses, the exact physical actions of quantum gates and other operations in gate-level circuits are correspondingly predefined microwave pulses. The pulse parameters such as frequency and amplitude are continuously changing due to the fluctuations in the environment and qubits. Therefore, the pulse parameters are frequently calibrated to reach high fidelity to the desired logic operations specified by the corresponding quantum gates. A pulse-level description provides a more granular view of quantum computation compared to the gate-level representation. It accounts for the physical implementation of quantum gates, offering insights into the precise control mechanisms and potential sources of error in quantum operations.

IV. Fault Target

Faults can occur or be injected at various locations or types of equipment within the quantum computing system. We focus here on the components within the domain of quantum computer fault injection attacks, defined in Figure 1: the Quantum Processing Unit, the Quantum Computer Controller, and the Auxiliary Processor.

A. Auxiliary Processor Faults

The auxiliary processor is a classical processor cortical in interpreting quantum computer read-out results and updating quantum program parameters. For example, in quantum machine learning (QML), there is an iterative process of running a circuit on a quantum computer, optimizing the circuit on the auxiliary processor based on results, running it again on a quantum computer with updated parameters, etc.

1) Faults in Classical Registers

Within the auxiliary processor are of course the usual components such as ALU, registers, or memory, among others.[1] Faults can be injected in these classical components to, for example, affect the computations used in QML optimization routines between executions of a circuit on a quantum computer. For program specification at the gate-level, the faults can result in gates being added, removed, or modified by changing the digital bits that specify them in the program. For program specification at the pulse-level, the faults can affect the digital specification of the amplitude, duration, or phase of the control pulses to be generated.

B. Quantum Computer Controller Faults

Quantum computer controller is typically made of equipment to generate microwave pulses to manipulate qubit states, and measurement equipment to translate quantum information into a classical format which is stored as the readout data.

1) Faults in Control Pulses

Faults can be injected into the control pulses generated by the quantum computer controller, for example, through EM radiation that affects the pulses generated by the controller, or more directly by affecting the operation of the controller itself causing it to generate wrong or modified pulses. Readout data is the classical data resulting from the measurements. Faults can also be injected into the readout control pulses through EM, for example, or the readout data can itself be directly manipulated through faults in digital registers storing the data within the controller. The control pulses control the operations or gates of the quantum computer which can be classified as unitary and non-unitary operations; both types are subject to faults:

- *Unitary Operations* – Unitary operations refer to transformations that preserve the normalization and reversibility of quantum states. Quantum gates are unitary gates, and unitary operations are the typical computational operations on the qubits, such as different X, SX, CX, or other gates.
- *Non-Unitary Operations* – Non-unitary operations are all other operations. For instance, reset or measurement are not unitary, because they collapse the state of the qubits during the execution of the operation.

2) Faults in Classical Registers

Non-unitary operations such as reset or measurement utilize classical registers. In particular, when qubits are measured, the quantum state collapses to one of the eigenstates of the measurement, and the measurement result is stored in classical registers or memories inside the control electronics. The classical registers then can be victims of fault injection that affects the classical bits:

- *Mid-Circuit Measurement* – Mid-circuit measurement allows for measuring the qubit state in the middle of the execution. The results can then be used to determine what code to execute by analyzing the classical bit measurement results. If the classical bit is modified, the circuit execution can be affected, as the classical bit at each mid-circuit measurement determines the next set of operations that will be applied.

[1] For simplicity, we specify the fault target here as “classical registers”, but the physical faults could also be in ALU, memory, or other components. Since the faults will eventually occur in or enter registers, we use the simplification of calling the target just “classical registers”.

- *Final Measurement* – The final measurement is performed at the end of each circuit. Usually, all qubits are measured, though sometimes ancilla qubits may not be measured. Injecting fault into the classical bits at this stage is effectively equivalent to manipulating the final circuit output.

C. Quantum Processing Unit Faults

The quantum processing unit implements qubits, such as the Josephson junction widely used to realize superconducting qubits. Attackers can also focus on faults in the quantum processing unit:

1) Faults in Physical Qubits or Couplings

There are many ways to influence and thus inject faults into the qubits. For instance, superconducting qubits are susceptible to decoherence, which refers to the loss of coherence and information due to interactions with the environment. External noise sources, such as thermal fluctuations or electromagnetic radiation, can cause qubits to lose their quantum states and result in errors. Faults can be injected through external means such as EM radiation or thermal changes to the fridge holding the qubits.

V. Classification

Our classification of quantum computer fault injection attacks is now presented in this section. The classification is presented in Figure 4 and detailed below.[2]

A. Fault Targets

In the classification, we separate the three targets into six specific components vulnerable to faults and list them in more detail below.

1) Quantum Processing Unit

- Target: *Qubits* are typically physical, two-level quantum-mechanical systems. A common type of qubit is built from a Josephson junction (but many others exist). As physical systems, they can be impacted by voltage changes, EM radiation, etc., that attackers can generate.
- Target: *Couplings* are typically intermediate electrical circuits used to connect qubits, they can be likewise impacted by voltage changes, EM radiation, etc.

2) Quantum Computer Controller

- Target: *Control Pules (Analog RF Signals)* are often microwave pulses sent to an antenna or transmission line coupled to the qubit with a frequency resonant with that qubit to realize an operation. The attacker can induce faults in the qubits or gate operations, e.g., by changing the frequency, phase, or envelope.
- Target: *Control Pulses (Digital Specification)* are generated by arbitrary waveform generators from digital specification, e.g. by an FPGA. The attacker can attack classical bits or classical operations that read, modify, or write the digital information, thus resulting in wrong pulses being sent.
- Target: *Classical Registers* are used, for example, to store measurement readout information during mid-circuit or final measurement. The attacker can induce faults in these classical registers, e.g., during mid-circuit measurement operations.

3) Auxiliary Processor

- Target: *Classical Registers* are also used in the auxiliary processor used to perform computations on the output. For example in quantum machine learning (QML), parts of the input circuit are optimized based on the results of computation, and the circuit is run again. The attacker can induce faults in these classical registers.

B. Fault Model

The fault model is a theoretical representation or framework that predicts or describes the types of faults that may occur in a system, their causes, and their potential effects. We have three fault models, corresponding to different targets.

1) Quantum Processing Unit

The qubits and couplings are vulnerable to three types of novel faults not found in classical computers: Faults can result in unitary type operations, which are effectively faults inducing a change in qubit state that can be reversed like any other (non-malicious) unitary gate. Faults can result in non-unitary operations, which are usually hard to reverse. Faults can result in enabling or disabling qubits or couplings, which may be similar to instruction skip faults in classical computers if a coupling is disabled, for example.

2) Quantum Computer Controller

The analog control pulses are also vulnerable to novel types of faults not found in classical computers: Faults can attenuate or amplify the analog pulses, causing different gate operations to be effectively performed. Faults can also shift the phase of the pulses, likewise resulting in different gate operations being effectively performed. The faults can also change the shape of the envelope of the pulse, again changing the gate operation performed. If the pulses are attenuated or otherwise sufficiently distorted, a gate operation may be effectively disabled. Conversely, amplifying or otherwise injecting an analog signal can create or insert a gate operation not part of the original circuit.

3) Quantum Computer Controller and Auxiliary Processor

The controller and auxiliary processor also contain digital classical information, specifying the pulses (before they are generated as analog microwave signals) and other registers. These are vulnerable to well-known stuck-at faults or bit toggling faults.

[2] The terminology used in this section focuses on superconducting qubit machines, but this classification can be equally applied to other types of quantum computers by replacing certain terms. For example, control microwave pulses can be replaced by laser pulses if ion-trap computers are considered.

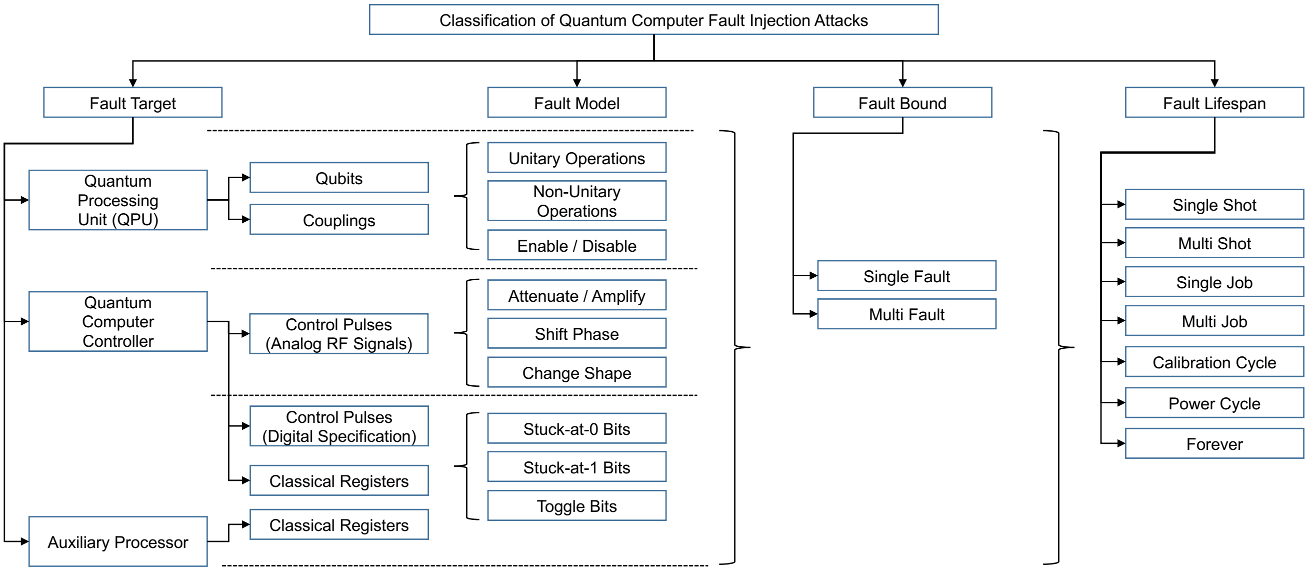

Fig. 4: Classification of quantum computer fault injection attacks.

C. Fault Bound

The fault bound is a limit or threshold that defines the maximum number of faults that a system can tolerate without significant degradation in its performance or functionality. Regardless of the fault target, there is either a single or multiple fault threat.

D. Fault Lifespan

The fault lifespan refers to the duration for which a fault persists in a system. In quantum computers, there are many more different lifespans compared to classical computers.

- Single Shot – each circuit is divided into one or more shots that are executed on a quantum computer; most short-lived faults would affect single shots. Most faults on analog pulses would fit in this category.
- Multi Shot – faults can persist through the execution of multiple shots of a circuit. Modification of the digital specification of the pulses would fit in this category.
- Single Job – multi-shot faults that last for all shots of a circuit would be Single Job faults.
- Multi Job – faults across multiple jobs of the same or different users would be multi-job faults. Faults in classical co-processor registers could fit in this category.
- Calibration Cycle – each quantum computer is calibrated frequently. Calibration can correct for changes in the environment or noise. Unitary operation-type faults in qubits could be in this category.
- Power Cycle – periodically, a quantum computer fridge has to be warmed up to replace or modify hardware, this is effectively a power cycle. Changes to control pulses which cause rapid heating and then cooling of the qubits could result in flux trapping, requiring power cycling the fridge.
- Forever – faults that permanently alter the hardware would be faults that last forever, e.g., disabling couplings.

VI. Related Work

There are only a few studies on fault injection attacks in quantum computers. Most of them are based on the hardware-induced faults in qubits [17]–[19]. Therefore, we drew inspiration from the fault injection literature in classical computing instead. Our decomposition includes Fault Target, Fault Model, Fault Bound, and Lifespan [20]–[22]. However, our classification represents the attack surface that is distinct from classical computers and, at the same time, identifies the hardware components that may be subject to fault injection attacks in quantum computers. Giraud et al. [23] classify fault injection attacks in classical computing as transient vs. permanent and invasive vs. non-invasive. However, for our study, we focused solely on non-invasive attacks and classified them as transient or permanent under the Fault Lifespan category. Furthermore, the fault target and fault manifestation security pyramid for superconducting quantum computers, shown in Figure 3, is the quantum computing counterpart of the one introduced by Verbauwhede et al. [21].

VII. Conclusion

This paper presented the first classification of fault injection attacks on quantum computers. This work first introduced the domain of quantum computer fault injection attacks. It then proceeded to present fault targets and fault manifestations for quantum computers. The resulting classification also specifies fault models unique to quantum computers, along with fault bounds and fault lifespans that should be considered. By shedding light on the vulnerabilities of quantum computers to fault injection attacks, this work contributes to the development of secure quantum computer systems.

Acknowledgements

The authors would like to thank Yao Lu for suggestions about potential flux trapping faults.

References

[1] IBM Quantum, "The hardware and software for the era of quantum utility is here," 2023. https://research.ibm.com/blog/quantum-roadmap-2033.

[2] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[3] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, p. 076001, jun 2013.

[4] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[5] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[7] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, (New York, NY, USA), p. 212–219, Association for Computing Machinery, 1996.

[8] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, p. 120–126, feb 1978.

[9] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," *Phys. Today*, vol. 54, no. 2, p. 60, 2001.

[10] IBM Quantum, 2023.

[11] Amazon Web Services, "Amazon Braket," 2023.

[12] Microsoft Azure, "Azure Quantum," 2023.

[13] IBM, "Bringing the full power of dynamic circuits to Qiskit Runtime," 2022.

[14] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," 2023.

[15] Amazon Braket SDK, 2023. https://docs.aws.amazon.com/braket/latest/developerguide/api-and-sdk-reference.html.

[16] C. Developers, "Cirq," Dec 2022.

[17] D. Oliveira, E. Giusto, E. Dri, N. Casciola, B. Baheri, Q. Guan, B. Montrucchio, and P. Rech, "Qufi: a quantum fault injector to measure the reliability of qubits and quantum circuits," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, (Los Alamitos, CA, USA), pp. 137–149, IEEE Computer Society, jun 2022.

[18] A. P. Vepsäläinen, A. H. Karamlou, J. L. Orrell, A. S. Dogra, B. Loer, F. Vasconcelos, D. K. Kim, A. J. Melville, B. M. Niedzielski, J. L. Yoder, *et al.*, "Impact of ionizing radiation on superconducting qubit coherence," *Nature*, vol. 584, no. 7822, pp. 551–556, 2020.

[19] "Exponential suppression of bit or phase errors with cyclic error correction," *Nature*, vol. 595, no. 7867, pp. 383–387, 2021.

[20] A. Baksi, S. Bhasin, J. Breier, D. Jap, and D. Saha, "A survey on fault attacks on symmetric key cryptosystems," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–34, 2022.

[21] I. Verbauwhede, D. Karaklajic, and J.-M. Schmidt, "The fault attack jungle-a classification model to guide you," in *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 3–8, IEEE, 2011.

[22] C. Shepherd, K. Markantonakis, N. van Heijningen, D. Aboulkassimi, C. Gaine, T. Heckmann, and D. Naccache, "Physical fault injection and side-channel attacks on mobile devices: A comprehensive analysis," *Computers & Security*, vol. 111, p. 102471, 2021.

[23] C. Giraud and H. Thiebeauld, "A survey on fault attacks," in *Smart Card Research and Advanced Applications VI: IFIP 18th World Computer Congress TC8/WG8. 8 & TC11/WG11. 2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS) 22–27 August 2004 Toulouse, France*, pp. 159–176, Springer, 2004.