POLITECNICO DI TORINO

Master's Degree in Software Engineering

Master's Degree Thesis

Fault Injection for Quantum Circuits

Supervisors

Prof. Bartolomeo MONTRUCCHIO

Dr. Edoardo GIUSTO

Candidate

Nadir CASCIOLA

July 2022

Summary

Quantum computing is an alternative type of computation that has the potential of outperforming classical computation in some specific problems. While shown to be theoretically possible, currently there are several technical challenges that prevent the construction of a quantum computer big enough to be useful. Among these, there is the fact that qubits, the fundamental elements of quantum computation, are highly susceptible to external sources of faults, such as ionizing radiation. These faults are also called *transient faults*, due to their temporary nature.

The work done in this thesis aims to better understand how transient faults propagate and corrupt the execution of quantum circuits, through the simulation of faults impacting the quantum circuits. In order to simulate the fault, we use a fault injector built on the Qiskit framework, one of the most widely used frameworks for quantum computing. The injector simulates the fault by inserting into the tested quantum circuit a parametrized quantum gate, a *U-gate*, in a specific location in the circuit in order to arbitrarily change the state of the qubit at that point of the execution. By inserting arbitrary faults in arbitrary locations of the tested quantum circuit, we can study its different vulnerabilities.

We perform three fault injection analyses: in the *single fault injection*, we inject one fault at a time in three different circuits; we learned that indeed different circuits can have different fault vulnerabilities: the same fault can be well tolerated by some circuits while being critical for others. In the *double fault injection*, we simulate a particle hitting two neighboring qubits instead of just one, by adding a second fault; as expected, we found that adding a second fault worsens the quality of the output in all cases. Lastly, in the injection on *quantum circuit cuts*, we apply our single fault injection analysis on a novel technique in the field of quantum computing research, which is quantum circuit cutting, a process that allows splitting a large quantum circuit into multiple, smaller subcircuits, which can then be independently executed in order to reconstruct the full output of the circuit. Here, by cutting a circuit and injecting faults in one of its subcircuits, we can study how faults propagate from a subcircuit to the final reconstructed output;

we found that given a circuit, its subcircuits can have different vulnerabilities to faults, and some of them can be much more critical than the others.

The work done in this thesis is an attempt to better understand fault propagation in quantum circuits. We believe some of these results can be used as first steps towards future work regarding, for example, the physical protection of quantum devices, by helping to concentrate the protection efforts to the more critical parts of the circuits.

Acknowledgements

Vorrei ringraziare prima di tutto la mia famiglia per avermi sempre supportato e dato la possibilità di intraprendere questo percorso.

Vorrei anche ringraziare i miei amici che hanno dato un senso e riempito di bei ricordi questi anni a Torino.

Infine, un sentito grazie a tutti quelli con cui ho collaborato durante il progetto di questa tesi, in particolare Edoardo e Emanuele: grazie davvero per essere stati sempre disponibili e pronti ad aiutarmi nei momenti di dubbio e difficoltà.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Context

The 20th century brought with it a new way of understanding and modeling the world around us: quantum mechanics. Physicists gradually realized that matter, at its most fundamental level, behaves very differently from what you would expect based on our everyday, human-scale world.

This bizarre behavior is so removed from everyday life that still today, a century later, physicists limit themselves in studying *how* quantum systems behave and evolve, and the question of *why* is in the realm of speculation and philosophy. In fact, there are several different interpretations of quantum mechanics, but no consensus has been reached [1].

In the 1980s, ideas started circulating about designing systems to exploit these weird rules of nature, extracting computations from them. The theoretical feasibility of this was shown by physicist Paul Benioff [2], and soon after it was suggested that it had the potential of outperforming classical computation in some tasks [3].

Nowadays, quantum computing is in its infancy. While it stands on solid theoretical ground, there are several technical challenges standing in the way of building an actually useful quantum computer.

One of them, arguably the hardest one, is maintaining the quantum properties of the qubits long enough to perform the computation. These qubits, which are the basic elements that hold the information, are highly sensitive to noise and interference.

1.2 Thesis work

The work done in this thesis aims to perform simulations of errors occurring in quantum computation in order to better understand and model the propagation and impact of these faults.

In order to do this, a quantum fault injector is used that arbitrarily modifies a qubit during a computation and observes the result. This way, data is gathered on all possible faults and it is better understood which are the most critical, causing the wrong output, and which are the less impactful that can be ignored.

Chapter 2

Quantum computing fundamentals

2.1 A new type of computation

Let's start at the beginning. Why quantum computation? Can't we do everything with our already powerful classical computers? Well yes, technically we can. Any classical computer can simulate any quantum computer. The key point is how many resources it would use to perform the same task. As we will soon show, quantum computers can perform some calculations exponentially faster than classical ones. Note the use of the word *exponentially*: in practical terms, this implies that yes, a classical computer can simulate a quantum computer. But if in order to do that, it has to store more information then there are atoms in the universe, it would be completely useless. That is the order of magnitude we are talking about.

How can a quantum computer be so efficient then? It is built to directly exploit some fundamental properties of matter. Instead of forcing matter into discrete 1s and 0s, like classical computation, it lets subatomic particles freely interact as they do in nature, following the rules of quantum mechanics, exploiting the full range of their (almost) real-valued properties. A quantum algorithm can be thought of as a careful way of setting up this interaction such as when we observe the result at the end, the state of the particles can be interpreted as the answer to our problem.

2.2 Concepts from quantum mechanics

2.2.1 Wave-particle duality and interference

What are these quantum properties that we are exploiting through a quantum computer? Historically, the first experiment that showed some quantum properties of matter was the double-slit experiment. In it, particles (like photons or electrons) are beamed to a screen that can detect their arrival, but before hitting the screen, they have to pass through a double slit, like in figure 2.1. If you were to predict what would happen with no knowledge of quantum mechanics, you would expect the particles to hit the screen in two bands, corresponding to the two slits like in figure 2.1. What instead really happens is that on the screen, an interference

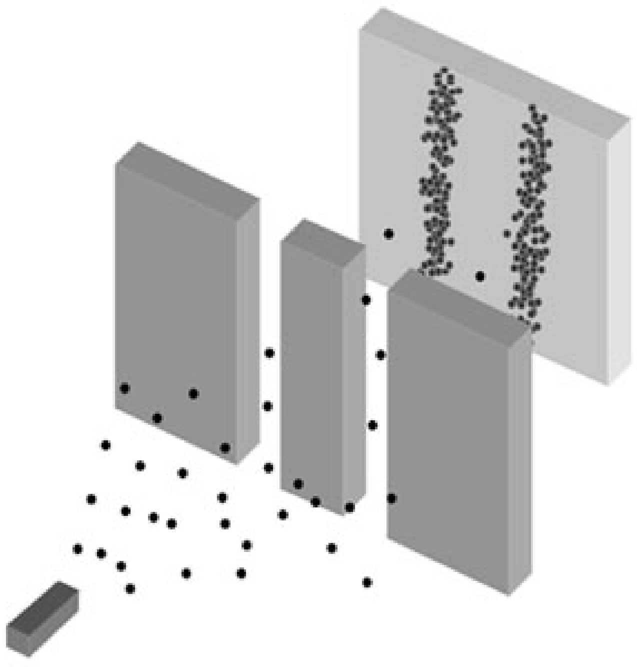


Figure 2.1: Double slit experiment as you would expect it from classical physics. This is NOT what happens! Source: *superquantumphysics* website [4]

patterns shows up, like in figure 2.2. What does this tell us about the particles that we sent then? It tells us that they are behaving like waves. More specifically, that they are exhibiting a specific property of waves: *interference.*

Imagine the whole experiment taking place on the surface of a small pond. Instead of the particle source that sends out particles, you throw a rock in its place, sending out waves. The waves pass through the slits: on the other side of the slits, there

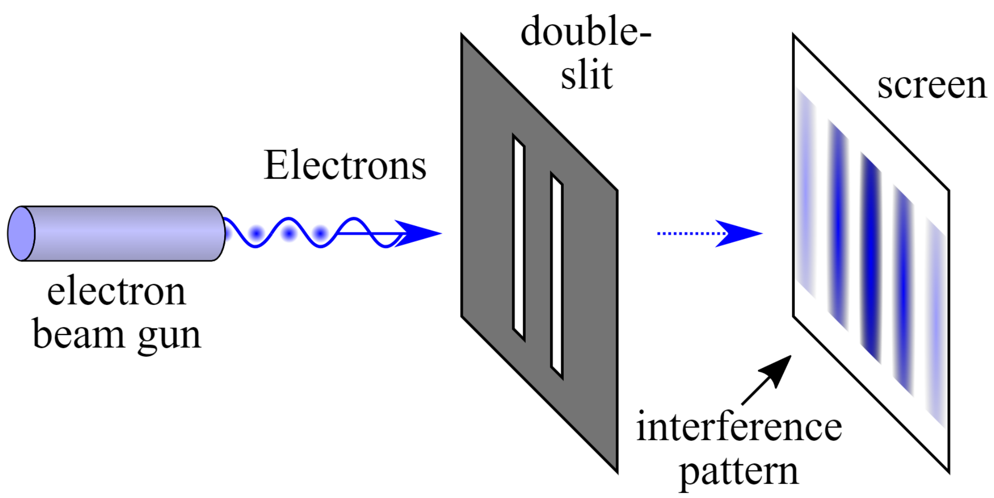


Figure 2.2: Double slit experiment. Source: Nature Noon article [5]

are now two sources of waves (the two slits). These two waves propagating from the two slits now interfere with each other: when two crests or troughs meet, you have *constructive* interference: the resulting wave is higher at the crests and lower at the troughs, corresponding to a higher amplitude wave; when a crest meets a trough, you have *destructive* interference: the two waves cancel out, and the water neither rises nor sinks, equivalent to a zero-amplitude wave.

Moreover, this interference pattern on the screen shows up even if we beam out one particle at a time. Meaning: send out a particle and wait for it to show up on the screen. Record where it landed, and then send out another particle. Repeat this to collect enough data, and slowly the same interference pattern shows up again! This seems to suggest that a single particle can interfere with *itself*.

How do we interpret this? Hard to say. What we can say is that particles appear to intrinsically posses some wave-like properties, like interference, and this is one of the properties that quantum computers will exploit.

2.2.2 Quantum superposition and quantum states

Let's first understand the classical counterpart of quantum superposition: wave superposition. When two (or more) waves are traversing the same space, the resulting wave at each point in space will be the sum of the amplitudes of the individual waves at that point.

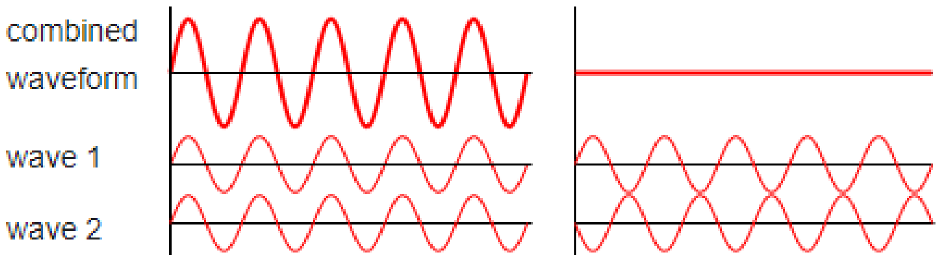


Figure 2.3: Wave superposition. Source: Wikipedia [6]

This means that any classical wave can be thought of as the resulting sum of other waves. This has a corresponding principle in the quantum world, but first: why do we even need the concept of quantum superposition? What is the phenomenon that it helps to describe?

Take the example of the double slit experiment. Like described in the previous section, even when shooting one electron at a time through the double slit, the interference pattern still appears. If you think classically, the electron should either go through the first slit, or through the second. So there are two possible states: "electron goes through slit 1" and "electron goes through slit 2". And there should be no interference pattern on the screen. In reality, after enough electrons, the interference patter does show up, and the only way to explain this is that while going through the slits, the electron is in neither one of the two classical states: it is in a *quantum state* that is the superposition of the two classical states.

This quantum state is different from classical states: after being observed, it collapses into one of the two classical states. But before being observed, it is in neither of them. The observation actually *changes* the quantum state, collapsing it into one of the classical states. Why this happens is still not exactly known: it is called the *measurement problem*. "Measurement" here is apparently anything that interacts with the quantum system to be measured: it does not involve explicitly consciousness.

This can be further experimentally confirmed by placing a detector, hence performing a measurement, just before the double slit. In this case, the quantum state of the electron is collapsed just before going through the double slit, and the electron actually goes through only one of the slits producing the 2 bands like in figure 2.1 instead of the interference pattern.

Quantum superposition then describes this quantum state that particles find themselves in before being measured.

Let's introduce the notation for quantum states that will be essential later on. We need a simple quantum state: for example, take an electron and its *spin* property; it can either be *up* or *down*. Let's define the "spin down" state as $|\downarrow\rangle$ and the "spin up" state as $|\uparrow\rangle$. Then, the most general state of the electron would be

$$|\psi\rangle = c_0|\downarrow\rangle + c_1|\uparrow\rangle$$

c_0 and c_1 being the *amplitudes*, complex numbers that encode the probability of observing their respective state. More specifically, $|c_i|^2$ gives the probability of observing state i.

2.2.3 Quantum entanglement

Entanglement is iron to the classical world's bronze age.

Michael Nielsen and Isaac Chuang

We have seen now that particles at subatomic scales possess some fuzzy behaviors, like interference and superposition. We have then described that until they are measured, they don't have definite properties: they are defined by a superposition of possible states, but before observation they aren't in any of those states.

There is another principle the we will add: the uncertainty principle. It asserts that there is an intrinsic amount of uncertainty in some properties of quantum objects. This uncertainty does not correspond to our lack of knowledge of those properties; instead, the mathematical description of quantum mechanics does not support some pairs of well defined properties for quantum objects. For example, the original formulation of the uncertainty principle states that the more precisely the *position* of some particle is defined, the less precisely its *momentum* is.

Now, with these concepts at our disposal, we can follow the same reasoning that gave rise to the famous EPR thought experiment, from which later the concept of entanglement emerged. In this experiment, a pair of particles is prepared in such a way so that the values of their positions and momenta depend on each other. For example, if after preparing them you measured the position of particle A, you would also be able to know the position of particle B, and the same thing applies to their momenta. The two particles prepared in this way will be later defined as being in an *entangled state.*

The paradox now is revealed when you realize that you can separate the two particles while not observing them, so maintaining their quantum state, and then you can choose to measure one of them, say particle A. If you choose to measure its position, then you would be able to predict also the position of particle B, and so particle B would have a definite position. If you instead chose to measure the momentum, then particle B would have a definite momentum (and not a definite position). But particle B is now light years away: what is the reality of particle B? Definite position or momentum? It can't depend on your choice of which measurement to perform on A, because the two particles are separated and that would imply an instantaneous *message* being transmitted between the two.

The EPR thought experiment has been experimentally verified several times: when you measure particle A and collapse its quantum state, the quantum state of particle B also collapses, even when the two particles are separated by a large distance.

This showed that quantum mechanics violates the *principle of locality*, and so is a *nonlocal* theory. Even though this is really counterintuitive, it does not lead to any physics-breaking conclusions: while entanglement shows events affecting one another in a faster-than-light way, it has been demonstrated that you cannot use this phenomenon to transmit *information* faster than light, because of the probabilistic nature of quantum states. Maybe locality was a wrong assumption that we made about the universe.

In conclusion, quantum entanglement is a phenomenon that occurs when a group of particles interact in such a way as to have some of their properties, like position or momentum, correlated with one another. This, together with the quantum description of reality, means that the quantum state of each particle cannot be described independently of the state of the others.

2.3 Quantum computation

2.3.1 The qubit

The quantum bit, or *qubit*, is the fundamental concept of quantum computation, analogous to the classical *bit*. The classical bit can be realized by any device that can be in one of two possible distinct states: for example, an electrical circuit that can have two voltage levels. The qubit, on the other hand, is a *two-state quantum system*, one of the simplest quantum systems that can exhibit quantum properties. For example, take the electron and its *spin* property: it can either be in the spin *up* state or spin *down* state but before being measured, and this is the main difference with a classical bit, it is in a superposition of the two.

If we define the spin up state as $|0\rangle$ and the spin down state as $|1\rangle$, the general state of the qubit is:

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

where α and β are the probability amplitudes, and are complex numbers. When this qubit is measured, it will be in state $|0\rangle$ with probability $|\alpha|^2$ and in state $|1\rangle$ with probability $|\beta|^2$. Since α and β are used to derive the probabilities, there is a further constraint:

$$|\alpha|^2 + |\beta|^2 = 1$$

It turns out that the amount of information contained in a single qubit is equivalent to the information needed to specify a point on the surface of a sphere. So a natural visual representation of a single qubit is the so called *bloch sphere*, as seen in figure 2.4.

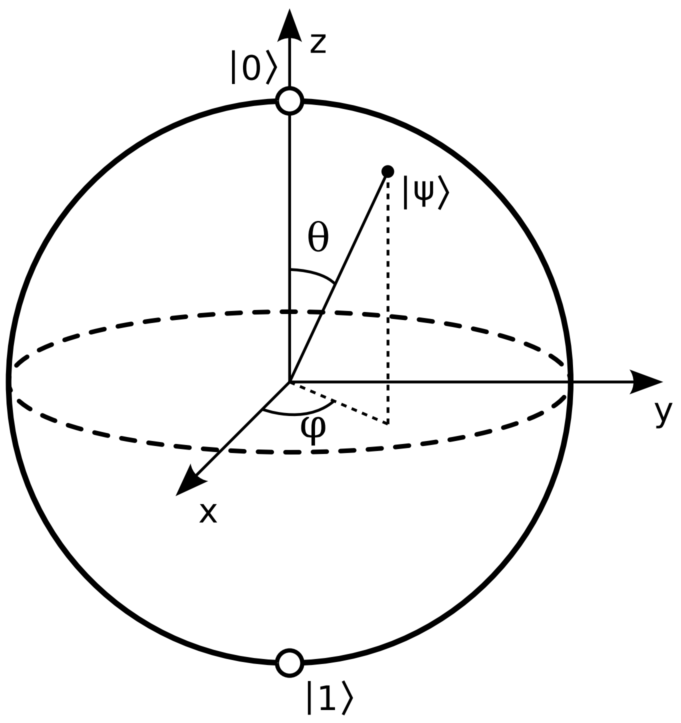


Figure 2.4: Bloch sphere representation of a single qubit. Source: Ketterer Andreas [7]

It might seem as though a qubit possesses an infinite amount of information, given that there is an infinite amount of points on the surface of a sphere. While that may be true, that information is fundamentally not accessible, because the only way to get something "out" of a qubit is to measure it, and that only gives you a 0 or a 1, so 1 bit of information; furthermore, once you measure it you collpase its quantum state and all that information is lost. In other words, given a qubit, in order to get the full decimal expansion of its α and β amplitudes, you would need to perform an infinite amount of measurements on an infinite amount of identical qubits.

What happens when you have multiple qubits together? Just like two classical bits can be in 4 possible states, 00, 01, 10 and 11, two qubits are defined by four states: $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$, and they can be in a superposition of these four states.

Multiple qubits can also be in an entangled state; for example, the *Bell state* defined like this:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

indicates that the two qubits always assume the same value: 50% of the time being in $|00\rangle$ and 50% of the time being in $|11\rangle$.

2.3.2 Quantum gates

How do we make the qubits perform calculations for us? We need to be able to change their state. We can do that by applying *quantum gates* to them. Classical bits are also manipulated by gates, more specifically *logical gates.* One of these gates is, for example, the *NOT* gate: it flips the classical bit, transforming a 0 in a 1 and vice versa. What would be the quantum equivalent of the NOT gate?

Firstly, it would need to transform a qubit in state $|0\rangle$ in a qubit in state $|1\rangle$ and vice versa. But that is not enough: what about all the other states described by the superposition $\alpha|0\rangle + \beta|1\rangle$? The quantum NOT gate, it turns out, exchanges the position of α and β, transforming the state

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

into

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

Interestingly, it corresponds to a rotation of π radians around the x axis of the Bloch sphere.

A single-qubit gate can be represented by a 2x2 matrix; for example, the quantum NOT gate just described is represented by the matrix

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

This makes it easy to calculate its effect on a given qubit. The state of the qubit can be written in vector form as $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, with the two entries corresponding to the amplitudes for $|0\rangle$ and $|1\rangle$ respectively. Then, the state of the qubit after the NOT gate has been applied to it will be

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

We have seen that single-qubit gates can be described by 2x2 matrices. What about multi-qubit gates? In general, a gate affecting n qubits is described by a 2^n square matrix.

2.3.3 Quantum circuits

To obtain a complete model of quantum computation, quantum gates alone aren't enough. A full model is represented by a *quantum circuit*: it contains quantum gates, wires, measurements, initializations of qubits and possibly other operations.

Let's see a visual representation of a quantum circuit to describe its notation. In figure 2.5 is represented a quantum circuit representing a famous quantum algorithm, the Deutsch-Jozsa algorithm.

First, we can see that there are four wires corresponding to the four qubits of our hypothetical quantum device, labeled from q_0 to q_3. There is also a notation indicating the presence of three classical wires (holding classical bits) that will store the measurements at the end (the wire at the bottom labeled c). Next, there is a series of quantum gates: H and X modify the state of the single qubit they are applied to, while the three $CNOT$ gates, the vertical lines with the + at the bottom, are two-qubit gates. Lastly, the qubits q_0 to q_2 are measured, and the outcome of each measurement (either 0 or 1) is put in one of the classical wires and will be the output of the quantum circuit.

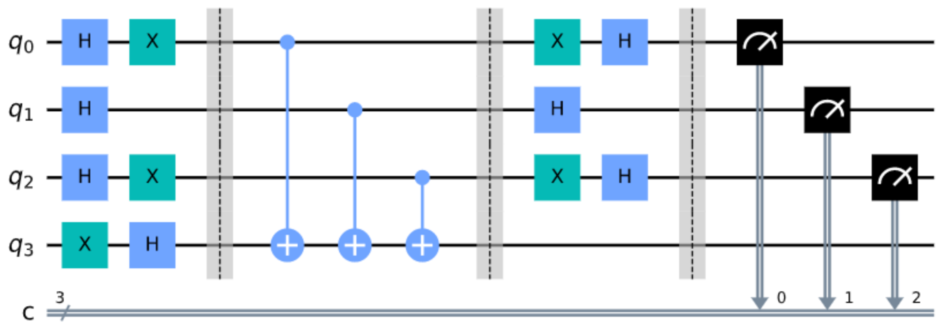


Figure 2.5: Deutsch-Jozsa algorithm. Source: Qiskit code

2.3.4 Quantum annealing

The previous description of quantum computation, the one involving quantum gates and quantum circuits, is not the only model of quantum computation available; though it is the most common, the most familiar (given its use of gates similar in some ways to classical gates) and it is also universal. *Quantum annealing* is another model of quantum computation that is applicable only to a certain subset of problems; more specifically, optimization problems that aim to find the global minimum of a given objective function. That is why we defined the gate array model as *universal* and not the quantum annealing one. Examples of such problems may be traffic flow solutions and molecular interactions.

In a quantum annealer device, quantum properties such as superposition, entanglement and quantum tunneling allow the qubits to simultaneously explore the landscape of the given objective function to optimize, making it possible to quickly discover local and global minima.

While the term *quantum annealing* exists in academia since 1988, current implementations of quantum annealing computers are being developed by D-Wave Systems, with customers such as Google and NASA.

2.4 Quantum algorithms

What problems can a quantum computer solve? Let's take a look at some important quantum algorithms that have been developed.

2.4.1 Deutsch-Jozsa algorithm

The Deutsch-Jozsa (DJ) algorithm was the forst example of a quantum algorithm being more efficient than the best classical algorithm.

The problem

The input of this algorithm is a hidden (thus, treated as a *black box*) function $\boldsymbol{f}$. This is a function that takes as input a string of bits, and returns either **0** or **1**:

$$f(\{x_0, x_1, x_2, ...\}) \rightarrow 0 \ or\ 1,\ where\ x_n\ is\ either\ 0\ or\ 1$$

This function $\boldsymbol{f}$ is guaranteed to be either *balanced* or *constant*. A constant function returns all 0's or all 1's for any input, while a balanced function returns 0's for exactly half of all inputs and 1's for the other half. The problem then is to determine whether $\boldsymbol{f}$ is balanced or constant.

The classical solution

Classically, in the worst case the problem can be solved by checking half of all possible inputs to $\boldsymbol{f}$ plus one: in that case, if we see all 0's or all 1's we know for sure that it is constant. Otherwise, we can stop at the first different output that we get. Since the total number of possible inputs is 2^n, we need $2^{n-1} + 1$ iterations to be certain of the result.

The quantum solution

A quantum computer can solve this problem after only one call to the function $\boldsymbol{f}$, provided that we also have implemented such function $\boldsymbol{f}$ as a quantum oracle (thus, in a quantum circuit). In Figure 2.6 there is a conceptualized schema of the algorithm. There are two quantum registers: the first one composed of n qubits, and the second one containing 1 qubit. To these two registers is applied a Hadamard gate (H). Then, the quantum oracle is applied, which contains the function $\boldsymbol{f}$. After the oracle, Hadamard gates are again applied and then the first register can be measured. At this point, if the qubits in the first register are all in the state $|0\rangle$ we know that the function $\boldsymbol{f}$ is constant; otherwise, it is balanced. This works because if the oracle is constant, it has no effect on the input qubits; hence, the qubits at the end are in the same state they started with, because the second Hadamard gate reverses the effect of the first one (the Hadamrd gate is its own inverse: two Hadamard gates in sequence have no effect). On the other hand, if the oracle is balanced, it affects the qubit and the second Hadamard gate doesn't reverse the first one, ending up in a state different than the all-zero one.

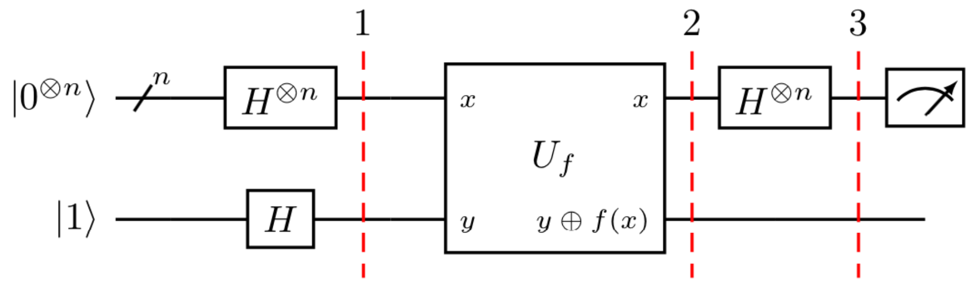


Figure 2.6: Steps of the Deutsch-Jozsa algorithm. Source: Qiskit Textbook [8]

2.4.2 Grover's algorithm

Grover's algorithm allows a quantum computer to search unstructured data quadratically faster than any classical computer. Suppose you have a list of N items, and you want to locate a specific one that has some unique property: this the *marked item*. To find it using classical computation, you would need to check, in the worst case, all N items. A quantum computer, with Grover, can find the marked item in $\sqrt{N}$ steps. In Figure 2.7 is shown a possible implementation of a 3-qubit Grover circuit. Similar to Deutsch-Jozsa, Grover's algorithm also makes use of an *oracle*. In this case, the oracle is a function (implemented in the quantum circuit) that given the full search space, is able to *mark* the item that we want to find. Marking in this case means adding a negative phase to it, while leaving all the other items unchanged. For example, in the Figure 2.7 case, the algorithms searches for a specific state in the possible 8 (2^n where $n = 3$, the number of qubits) states. The oracle marks the state $\omega = 101$, adding a negative phase to it; the matrix representation of the oracle is like this:

$$U_\omega = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{l} \\ \\ \\ \\ \\ \leftarrow \omega = 101 \\ \\ \\ \end{array}$$

The algorithm is composed of three main steps: the first is an initialization phase, where an Hadamrd gate is applied to the qubits. Then, the oracle is applied which marks the wanted item with a negative phase. Lastly, there is the *amplitude amplification* step, which makes use of the marked item's negative phase to increase its amplitude (so the probability of being measured) while reducing the amplitude of the other non-marked states. After this step there is a near-certainty to end up measuring the marked state, successfully finding the wanted item.

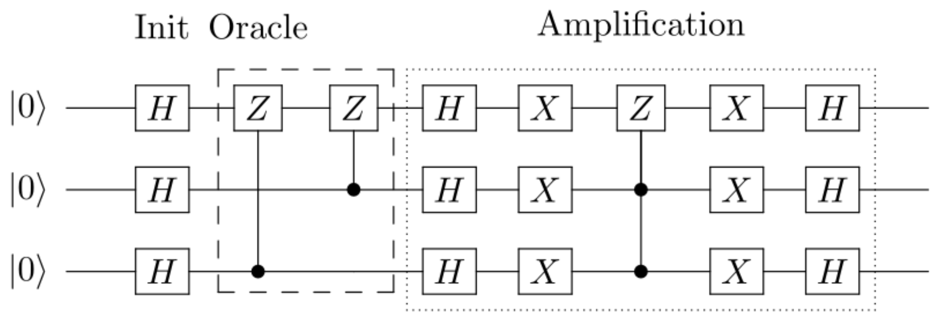


Figure 2.7: Example of a 3-qubit Grover implementation. Source: Qiskit Textbook [8]

2.4.3 Quantum Fourier transform

The Quantum Fourier transform (QFT) is the quantum analogue to the discrete Fourier transform over the amplitudes of a wavefunction. It is a very important quantum algorithm, being part of many other quantum algorithms, such as Shor's algorithm and quantum phase estimation.

The quantum Fourier transform acts on the qubits by transforming their state between two bases, the computational (Z) basis, and the Fourier basis. In fact, all multi-qubit states in the computational basis have corresponding states in the Fourier basis, and the QFT simply translates between these two bases.

$$|State\ in\ Computational\ Basis\rangle \xrightarrow{\text{QFT}} |State\ in\ Fourier\ Basis\rangle$$

The H-gate is actually a single-qubit QFT, transforming the $|0\rangle$ and $|1\rangle$ states (Z-basis) to the $|+\rangle$ and $|-\rangle$ states (X-basis). Working with qubits in the Fourier basis is useful because their value is encoded in them through fractional phases, and this makes some operations easier, like addition and multiplication. In Figure 2.8 is shown a general N-qubit implementation of a QFT algorithm, with H being

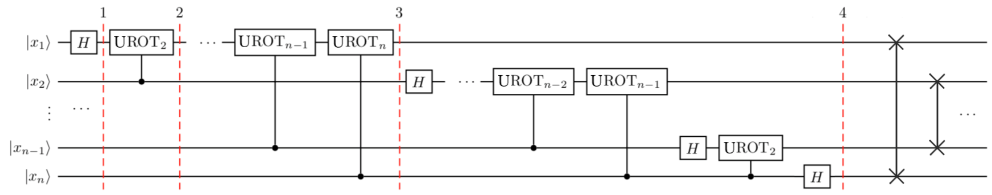


Figure 2.8: N-qubit QFT implementation. Source: Qiskit Textbook [8]

the Hadamard gate and $UROT$ being a 2-qubit rotation defined as follows:

$$UROT_k = \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(\frac{2\pi i}{2^k}\right) \end{bmatrix}$$

At the end of this circuit the final state is the QFT of the input state, but with the order of the qubits reversed.

2.4.4 Shor's algorithm

Shor's algorithm is a famous quantum algorithm for finding the prime factors of an integer. This is an extremely important problem, since current cryptographic algorithms, like RSA, are based on the assumption that factoring large integers is computationally expensive. This is true for classical algorithms, since the best known classical algorithm for factoring integers runs in *sub-exponential time*, more precisely:

$$O\Big(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}}\Big)$$

which is quite expensive. Shor's algorithm, on the other hand, is able to factor integers in *polynomial time*, more specifically it runs in:

$$O\Big((\log N)^2(\log\log N)(\log\log\log N)\Big)$$

which is almost exponentially faster than the classical one. Once we are able to construct quantum computers big enough to run Shor's algorithm for large integers, everyone will have to switch from using current cryptographic algorithms like RSA to new algorithms collectively known as *post-quantum cryptography*, which are resistant to quantum algorithms as well as classical ones.

The strength of Shor's algorithm is that it turns the integer factoring problem into a *period finding* problem. Period finding means given a periodic function, find its period: the smallest integer after which the function starts repeating itself. This problem can be efficiently solved, in polynomial time, by using the quantum phase estimation algorithm (based on the quantum Fourier transform). Thus, Shor's algorithm can use the quantum phase estimation algorithm to efficiently solve the factoring problem as well.

Chapter 3

Quantum noise

Quantum devices, like classical ones, are affected by different sources of noise. We can divide them in two main categories: *intrinsic* noise and *stochastic* noise.

3.1 Intrinsic noise

Intrinsic noise relates to noise that depends on the physical realization of the qubit in question. It is always present, even in the absence of external noise sources. Usually, two main metrics are used to quantify intrinsic noise: T_1, or relaxation time, is the amount of time that it takes for a qubit in an excited state (i.e. $|1\rangle$) to naturally decay to a lower energy state (i.e. $|0\rangle$). The other metric is T_2, or dephasing time, which is the time that it takes for a qubit in a superposition state (i.e. $|+\rangle$ or $|-\rangle$) to dephase to a state in which the phase cannot be accurately predicted; this happens because of the loss of quantum coherence. In fact, T_2 can be thought of as the loss of quantum coherence over time.

3.2 Stochastic noise: ionizing radiation

Stochastic noise is instead the noise caused by external sources. These are, for example, cosmic rays. The Earth, end everything on its surface, is constantly bombarded by high-energy particles. The sources of these particles are various astrophysical processes, and originate in different parts of space: from solar eruptions of our own Sun, to astronomical objects in our galaxy and even in distant galaxies.

When these particles hit computers, they can cause errors. In classical computers for example, a particle hit can cause data corruption on memory or a wrong execution on a CPU. These errors are called *transient faults* or *soft errors*, because they are one-time events, not permanent damage. Classical computers deal with

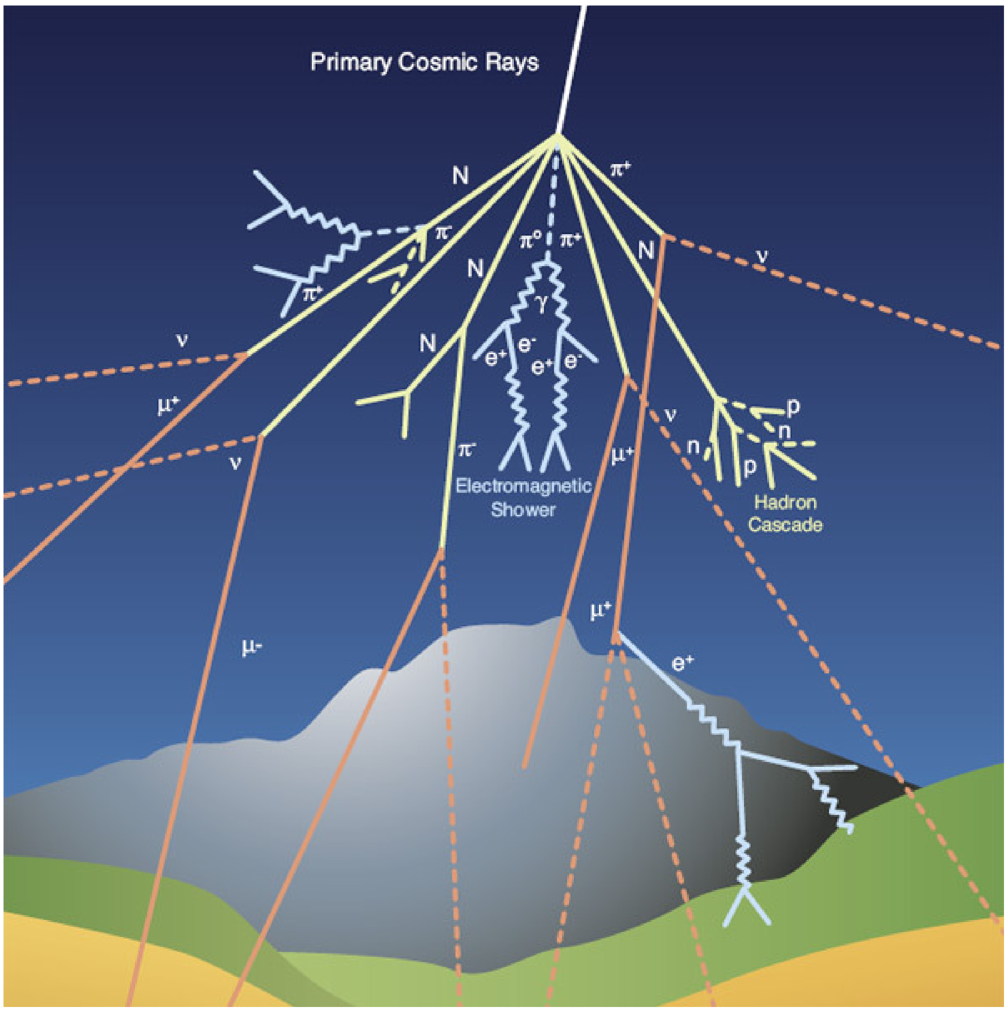


Figure 3.1: Cosmic rays visualization. Source: CERN [9]

this by using error corrected memory (ECC memory, which uses redundant bits to detect and correct errors) or repeating instructions on CPUs.

In quantum computers, it has been shown in recent studies that ionizing radiation can cause the decoherence of the computer's qubits [10, 11], leading to a wrong calculation. This suggests that mitigating the effect of ionizing radiation on quantum computers will be a critical component of achieving fault-tolerant quantum computers. Figure 3.2 shows a plot from one of these recent studies, specifically *McEwen et al, 2022* [11]. The experiment consisted in preparing the qubits in the excited state $|1\rangle$, then allowing them to idle for 1 μs and then measuring them, repeating this process multiple times. The objective was to collect data on qubit

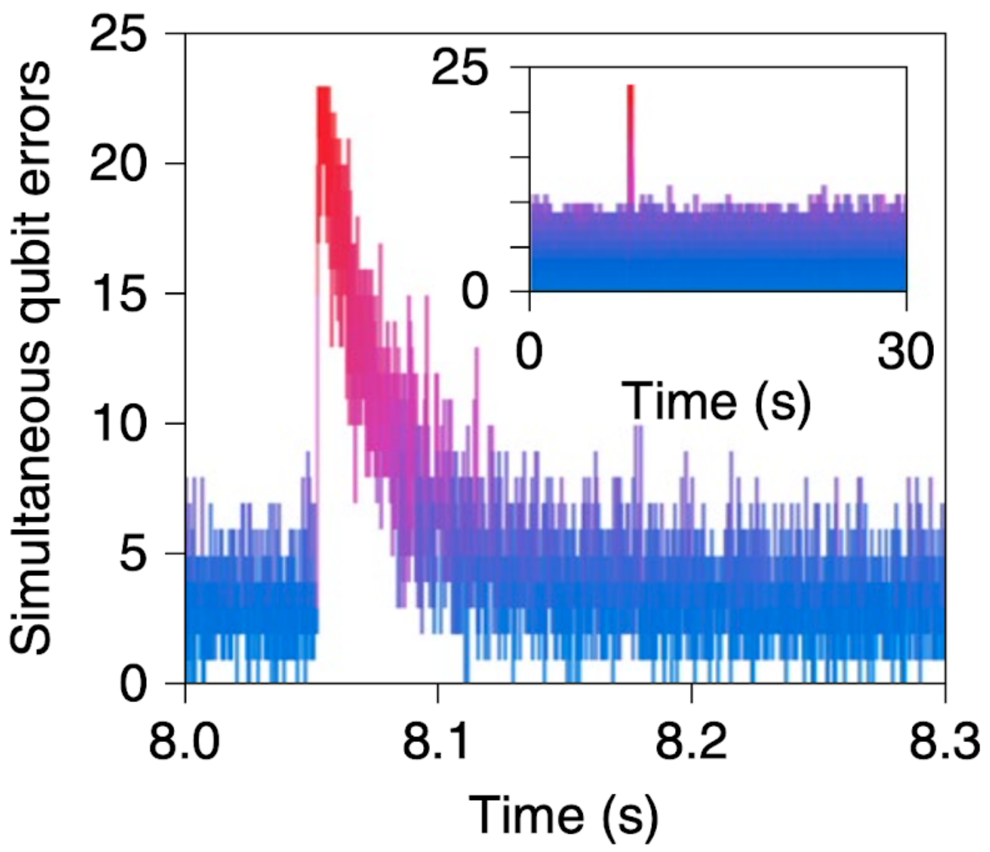


Figure 3.2: Plot showing the impact of a cosmic ray hitting a quantum device: the number of qubit errors (y-axis) suddenly spikes, before returning to normal levels. Source: McEwen et al, 2022 [11]

errors in the event of a particle strike. The Figure shows one particle strike and its effect on a 26 qubit subset of a Google Sycamore processor: we can see that the quantum processor has a baseline of around 4 errors, but when the particle hits the error rate suddenly spikes to around 23-24, thus effectively saturating the chip. Then, the error rate gradually returns to the baseline. They counted as errors any measurement where a qubit was in the $|0\rangle$ state, since that implies that it had decayed from the prepared $|1\rangle$ state.

This study clearly shows how critical transient faults, and ionizing radiation in particular, can be for a quantum computer.

Chapter 4

Fault injection in quantum computation

4.1 Modeling the fault

When a particle hits a quantum computer, what actually happens? How can we model it? In classical computers, it is relatively easy: a particle hitting a memory cell could change a 0 into a 1, or vice versa, and there are no other options. If a particle hits a qubit, on the other hand, it can change its value in any way: since the value of a qubit corresponds to a point on the surface of a sphere, we can imagine the particle changing this point to any other point, equivalent to a rotation around some axis. Mathematically, this means that the way a particle changes a qubit can be described by two real numbers, which we will call θ and ϕ, as shown in figure 4.1. The qubit state $|\psi\rangle$ (green) is *shifted* to $|\psi*\rangle$ (red), and this shift can be described by how much their θ and ϕ have changed.

This way, we can model any shift of any intensity of a single qubit.

4.2 Fault injection

Now that we know how to model an impact of a particle, how do we go about using it to learn more about faults in quantum computers? The idea is to simulate the effect of a particle hitting a given quantum circuit, and then observe how the output of the circuit is impacted.

A particle would change the state of a qubit in a random way. In order to simulate that, we need to arbitrarily change the state of a qubit in a given position

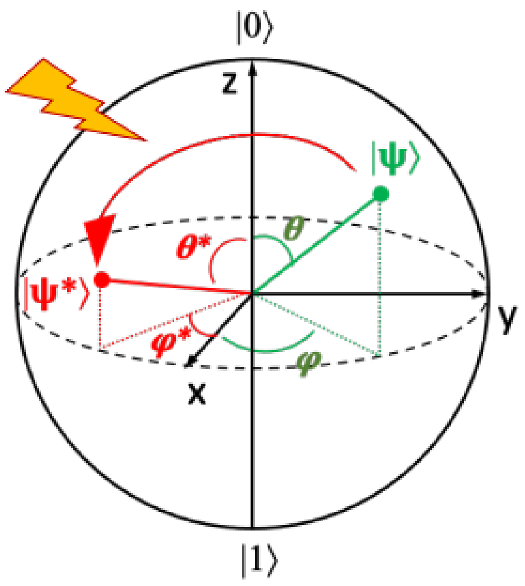


Figure 4.1: Modeling of the impact of ionizing radiation on a single qubit. The qubit state is changed from $|\psi\rangle$ to $|\psi*\rangle$. Source: Oliveira et al., 2021 [12]

in the circuit. In Qiskit, the quantum computing framework that we are using, there is a gate that allows us to do just that: the U gate. It is the most general

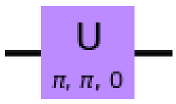


Figure 4.2: Qiskit's U-gate. Its parameters are θ, ϕ and λ

gate available in Qiskit. By defining a U gate with a specific θ and ϕ and inserting it into a circuit, we can change a qubit in any way we want.

Let's take a circuit as an example: an implementation of Grover's algorithm. This is an algorithm used to search an unordered list. A 2-qubit implementation is shown in figure 4.3. Since we want to simulate all possible faults, we will inject the U gate in each possible position in the circuit. Not only that, but for each of those positions we will inject a range of possible values of θ and ϕ. Usually, these will be from 0 to 2π with increments of $\frac{\pi}{12}$. For example, in figure 4.4 there is a possible injection in the Grover circuit: the qubit *q0* is injected after the first gate, with a ϕ shift of $\frac{\pi}{4}$.

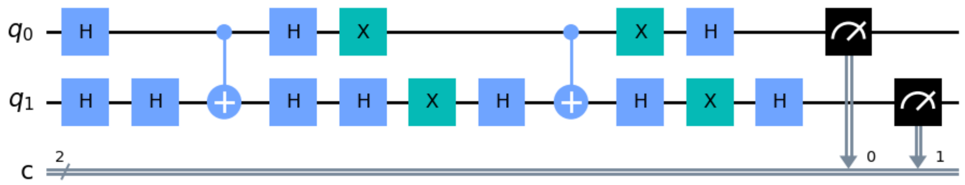


Figure 4.3: 2-qubit version of Grover's algorithm.

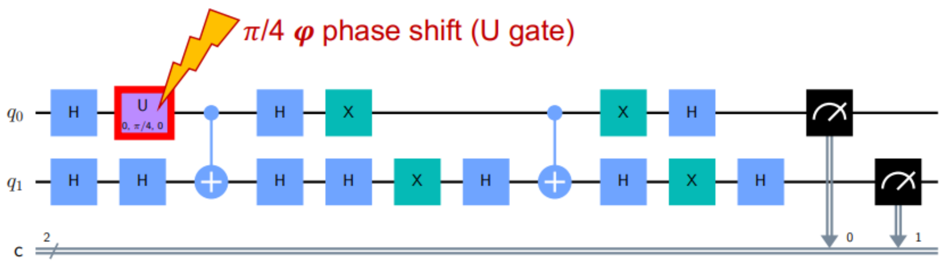


Figure 4.4: Example of injected Grover circuit. Source: Oliveira et al., 2021 [12]

4.3 QVF metric

4.3.1 The output of a quantum circuit

We now have a way of simulating a fault in a quantum circuit. How do we characterize its effect on the output of the circuit? First, let's see how the output of a quantum circuit looks like. In order to extract information from the qubits, they need to be measured. This step can be seen as the rightmost operations on the Grover circuit example, in figure 4.3. The result of each measurement operation is either $|0\rangle$ or $|1\rangle$, since the complex quantum state of the qubit is collapsed by the measurement. Actually, for the sake of precision, this is not generally true: the result of the measurement depends on which measurement basis is used. The *computational basis*, the one that results in $|0\rangle$ or $|1\rangle$, is the most common one.

If we measure N qubits then, we expect to get a string of N bits. In addition, quantum computation has an intrinsic probabilistic component, hence the same computation is performed several times, and our final result will be a distribution of these N bit strings.

For example, in figure 4.5 is the output of the Grover circuit shown in figure 4.3. Here we can see that the string *11* has a 81.6% probability of being measured, much more than all the others, so we can easily define it as *the* output of the circuit. Though we can begin to see here how it can be tricky to judge the *quality* of the output of a quantum circuit. What if there are two strings with equal probability, one correct and one incorrect? Is that better or worse than the case in which the most probable one is an incorrect one? And, most importantly, how can we define a metric that, while considering all these factors, decides on a number that represents the quality of a circuit output?

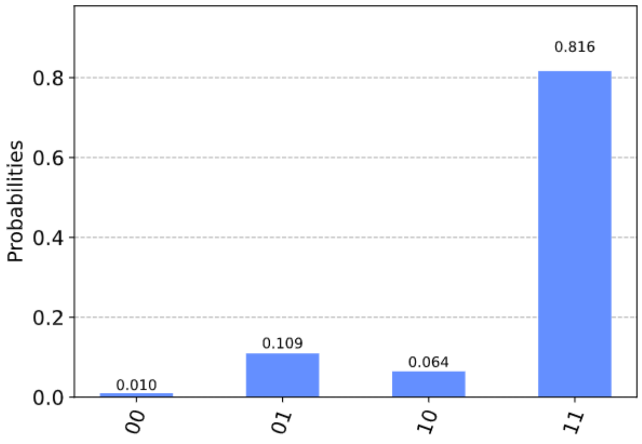


Figure 4.5: Example of the output of the Grover circuit.

4.3.2 The QVF metric

As hinted at in the previous section, determining the correctness of the output of a quantum circuit is not a binary correct/incorrect problem. There is some nuance to it: for example, the best case scenario is when the correct state is the one with the most probability of appearing by a large margin, like in figure 4.5. On the other hand, if there is ambiguity when choosing the most probable state, it is not an ideal situation, even if in that case the correct state might be the most probable one.

We want then to define a metric that, given the probability distribution of the states of a circuit, outputs a number between 0 and 1 that reflects how *wrong* the output of the circuit is. For example, if the most probable state is an incorrect one by a large margin, the metric should be close to *1*. When there is ambiguity when choosing a state, maybe the correct state is tied in probability with an incorrect one, the metric should be around 0.5. In the best case scenario, when it is easy to select the correct state, it should be close to 0.

We can define such a metric starting with the Michelson Contrast, which is used in graphical processing, and applying it to quantum computing outputs. This equation helps us to define how confidently we can select the correct state among all the states in the output:

$$Contrast = \frac{P(A) - P(B)}{P(A) + P(B)}$$

where $P(A)$ is the probability of the correct state and $P(B)$ is the highest probability among the incorrect states. The contrast calculated as such is in the range $[-1, 1]$, since it is possible to have $P(A) < P(B)$, especially when considering fault injected circuits. Also, we want a metric where a lower value means a more correct output, and this is the opposite. In order to fix this and shift the range to $[0, 1]$, we can define the metric as:

$$QVF = 1 - (Contrast + 1)/2$$

The QVF (Quantum Vulnerability Factor) defined like this possesses the desired properties discussed previously: values close to zero indicate that the correct state is the most probable and there is no ambiguity when selecting it. QVF values close to 0.5 indicate that the correct state and some other incorrect state have similar probabilities, making it hard to confidently select the correct one. Lastly, QVF values close to 1 imply that some incorrect state has even higher probability than the correct one, which is a worst case scenario. In figure 4.6 there is an example of QVF values for three different faults injected in the 2 qubit Grover circuit, showing some of these properties.

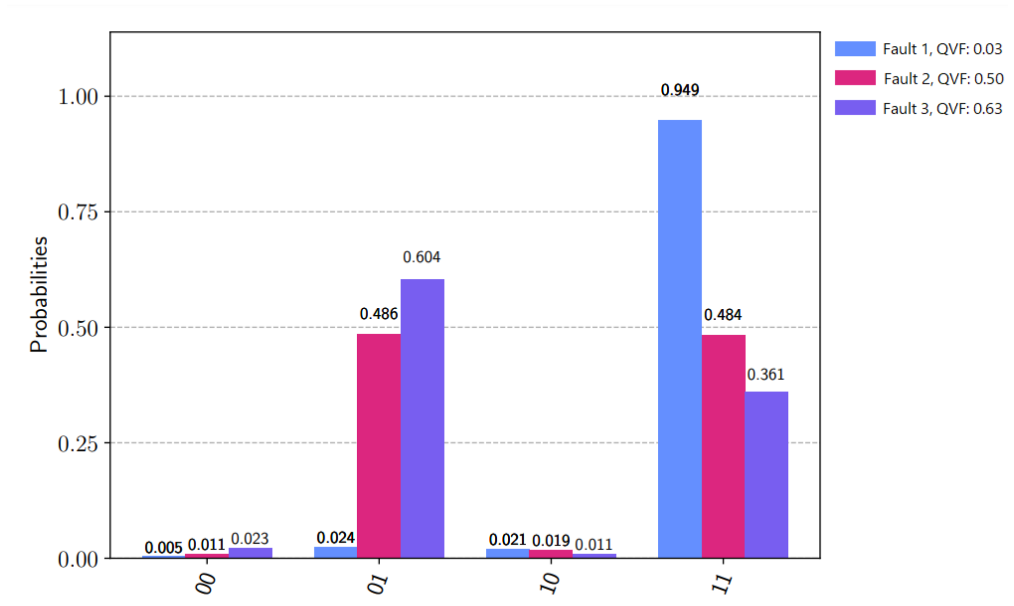


Figure 4.6: Example of QVF on 3 different outputs of the 2 qubit Grover circuit. Source: Oliveira et al., 2021 [12]

Chapter 5

Quantum circuit cutting

Part of this thesis involves studying the propagation of faults when a novel technique, *quantum circuit cutting*, is applied to quantum circuits. This chapter will serve as a brief introduction to circuit cutting.

5.1 Basic idea

Quantum circuit cutting is a novel technique that consists in splitting a quantum circuit into multiple and smaller *sub-circuits* (or *cuts*). These subcircuits can then be run independently from one another and their individual outputs can be recombined to reconstruct the final output, which is demonstrated to be equivalent to the output of the full (*uncut*) original circuit [13, 14]. This recombination is a classical computation, and is sometimes defined as the *classical postprocessing step*. The advantage that circuit cutting offers is that you can effectively execute N-qubits quantum circuits on M-qubits quantum computers, where $M < N$. For example, in Figure 5.1 is shown how a 4-qubit implementation of the Quantum Fourier Transform algorithm is cut into three 3-qubit subcircuits. We can see how each of the three subcircuits is smaller than the original circuit, both in number of operations and in number of qubits, effectively allowing the execution of the original 4-qubit circuit with a device with potentially only 3 qubits.

5.2 Circuit cutting process

The process of circuit cutting involves, first and foremost, selecting the cut locations. This can be done manually or automatically. In this work, we used a recently published framework that automatically selects the cut locations; this will be detailed in section 6.3. For now, let's take the cutting of the QFT circuit in Figure 5.1 as an example. Each qubit *wire* has been cut in one or more locations;

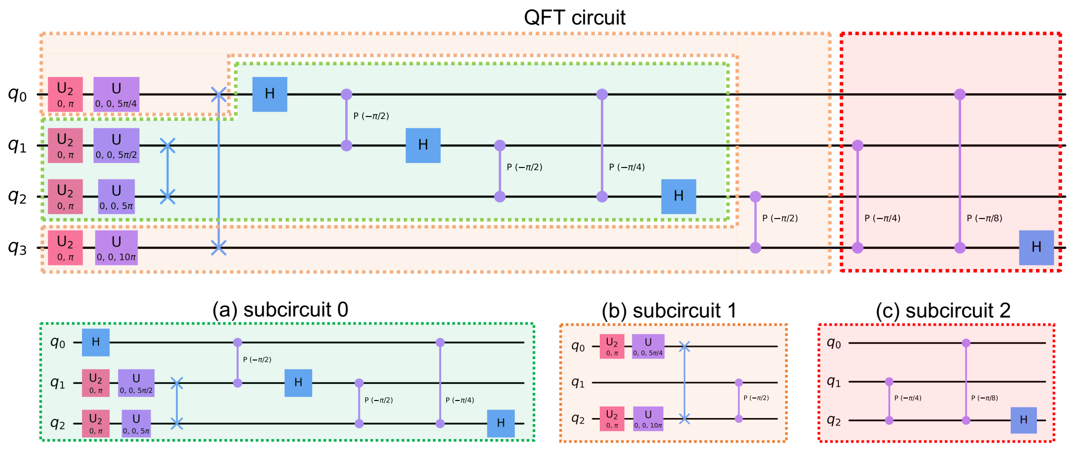


Figure 5.1: Original 4-qubit QFT circuit (top) and the three subcircuits of 3 qubits each (bottom) obtained from circuit cutting. The three cuts output combination provides the original circuit output, allowing the cuts to be executed on quantum computers with fewer qubits. Source: [15]

q0, for example, has been cut between its third and fourth operation and between its sixth and seventh. The set of all cuts on all qubit wires defines a *cut solution*, from which the smaller subcircuits to be executed can be extracted.

After obtaining a cut solution, the subcircuits are executed just as you would execute any quantum circuit. The reconstruction of the original circuit's final output is done by performing a series of Kronecker products between the relevant outputs of the subcircuits, and finally summing them together. This process relies on measuring each (relevant) qubit of each subcircuit on a set of orthonormal matrix bases, for example the set of Pauli matrices $I,\ X,\ Y,\ Z$. For this reason multiple measurements are required, and the executions of the circuits should be performed several times in order to better capture its probabilistic nature. Note that this is not a problematic overhead, since even in standard quantum computing it is expected to perform the experiments multiple times, given the general probabilistic properties of quantum computers.

Chapter 6

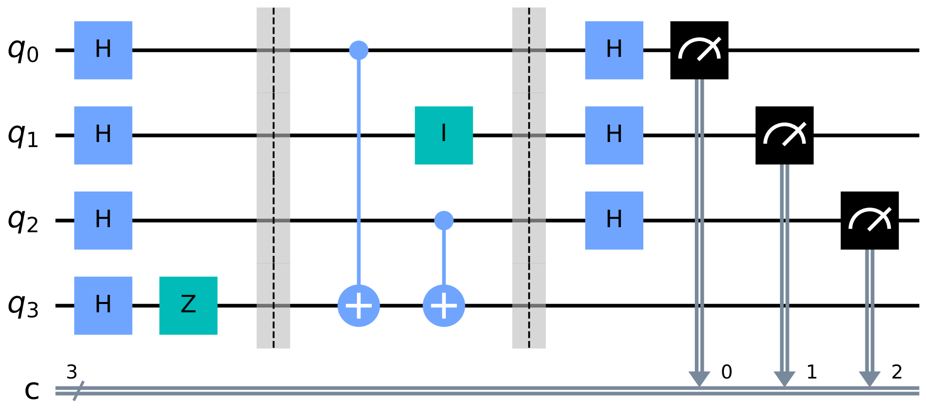
Experimental setup

In this Chapter, we'll describe more in detail the experiments performed, after having given a general introduction to the methodology in the previous Chapter 4. In the following Chapter, Chapter 7, we'll present the results. The experiments are divided in three parts: *single fault injection*, where we inject one fault at a time on a given quantum circuit; *double fault injection*, where we inject two faults at a time. And lastly, *injection on quantum circuit cuts*, where we perform (single) fault injection on a recent innovation in quantum computing, *circuit cutting*, which aims to split large quantum circuits in smaller, independent portions to be more easily executed. This will be better explained in its own section.

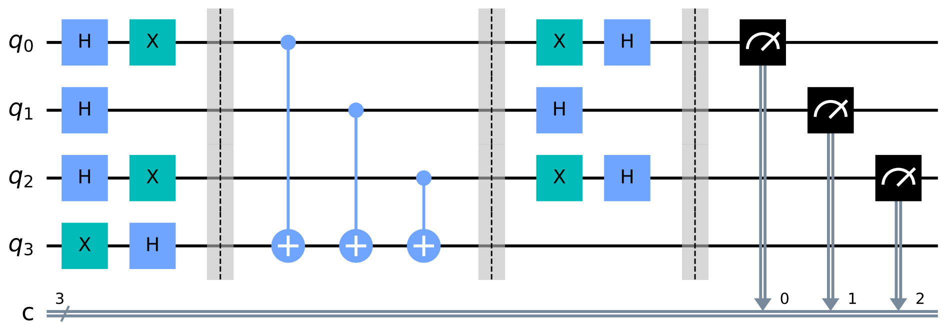
6.1 Single fault injection

This experiment involves taking a circuit and injecting faults one at a time over all possible positions in the circuit, over a defined range of (θ, ϕ) values. The tested circuits are: Bernstein–Vazirani, Deutsch–Jozsa and Quantum Fourier Transform, shown in Figure 6.1.

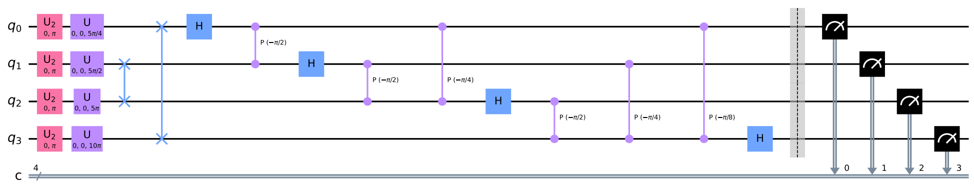
Deutsch–Jozsa (DJ, Figure 6.1b), while of limited practical use, was the first algorithm that showed that Quantum Computer could be faster than classical computers and is a circuit that, given a function executed, is able to identify if the function is constant or balanced. Bernstein-Vazirani (BV) algorithm is an extension of Deutsch-Josza that identifies a string encoded in a function. Lastly, Quantum Fourier Transform (QFT) is the quantum analogue of the discrete Fourier transform. It is particularly interesting as it is a fundamental part of many quantum algorithms, such as Quantum Phase Estimation (QFE) and Shor's factoring algorithm.



(a) Bernstein–Vazirani circuit (4 qubits) used in the injection



(b) Deutsch–Jozsa circuit (4 qubits) used in the injection



(c) Quantum Fourier Transform circuit (4 qubits) used in the injection

Figure 6.1: Quantum circuits used for single fault injection.

First, the considered circuit is executed as is, without faults, to extract the *gold output*, i.e. the output of the fault-less execution, that will be used as reference. Then, faults are injected in each possible position in the circuit. More in detail, this means that a *U-gate* that rotates the qubit by a specific (θ, ϕ) combination is inserted before each gate in the circuit, one at a time. This is done for all possible combinations of (θ, ϕ) over a range of $\theta = [0, \pi]$ and $\phi = [0, 2\pi[$, each with a step of $\frac{\pi}{12}$. This configuration results in 312 possible injections for each position in the quantum circuit. Each injected circuit generated as such is then executed just as one would execute any quantum circuit, and its output compared with the gold output to compute the specific QVF value for that injection.

6.2 Double fault injection

Since it has been shown that a single particle strike can impact multiple qubits [16], the next step is to try and inject multiple faults at the same time. Here, we inject a maximum of two faults simultaneously. We will denote the first fault as (θ_0, ϕ_0) and the second one as (θ_1, ϕ_1). It is still unknown how to exactly model a particle strike affecting multiple qubits; thus, we proceed as following: the first fault, (θ_0, ϕ_0), is injected exactly the same way as for the single fault injection, so considering all possible positions in the circuit and all (θ, ϕ) combinations in a certain range. Then, once we select a location for the first fault and a specific (θ_0, ϕ_0) shift, we inject a set of possible second faults, considering all the *neighboring* qubits and all lower phase shifts with respect to the first one, i.e. $\theta_1 \leq \theta_0$ and $\phi_1 \leq \phi_0$.

An important information to specify is what exactly we mean by *neighboring* qubits. Circuits representations like those in Figure 6.1 are *logical* circuits. They denote the sequence of operations that are specified by the programmer who designed it. This is not the exact sequence of operations that are executed in a given physical quantum computer. This may be because a quantum computer may not have those specific operations available, and so it has to execute a different sequence of operations that is mathematically equivalent to the logical one, but different in order to remain within the constraints of its hardware. This process of rewriting a logical quantum circuit to match the topology of a specific quantum device is called *transpilation.* Coming back to the location of the second fault then, when talking about neighboring qubits, we first take a physical quantum computer as reference. Then, looking at its topology, we identify the qubits that are actually physically close to each other. Lastly, when choosing the location of the second fault, we consider all those qubits physically adjacent to the qubit of the first fault. For example, in Figure 6.2 there is an example of a quantum computer's *coupling map*, more specifically IBM's Vigo. This depicts the qubit couples that permit CNOT gates between them, which are 2-qubit gates, indicating physical adjacency. In this study, a fault in qubit *1*, for example, is going to generate (smaller) faults on qubits *0, 2, 3*; while a fault in qubit *4* is only going to produce faults in qubit *3*.

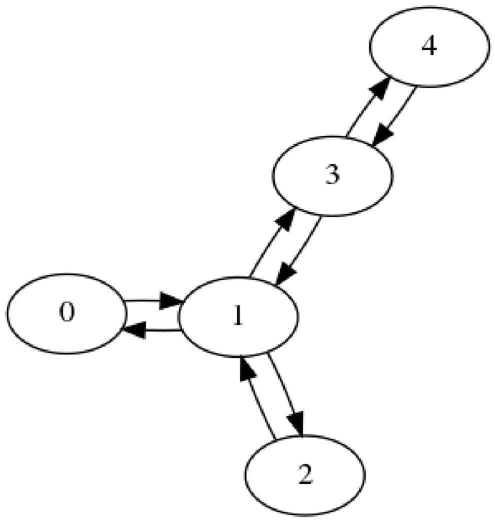


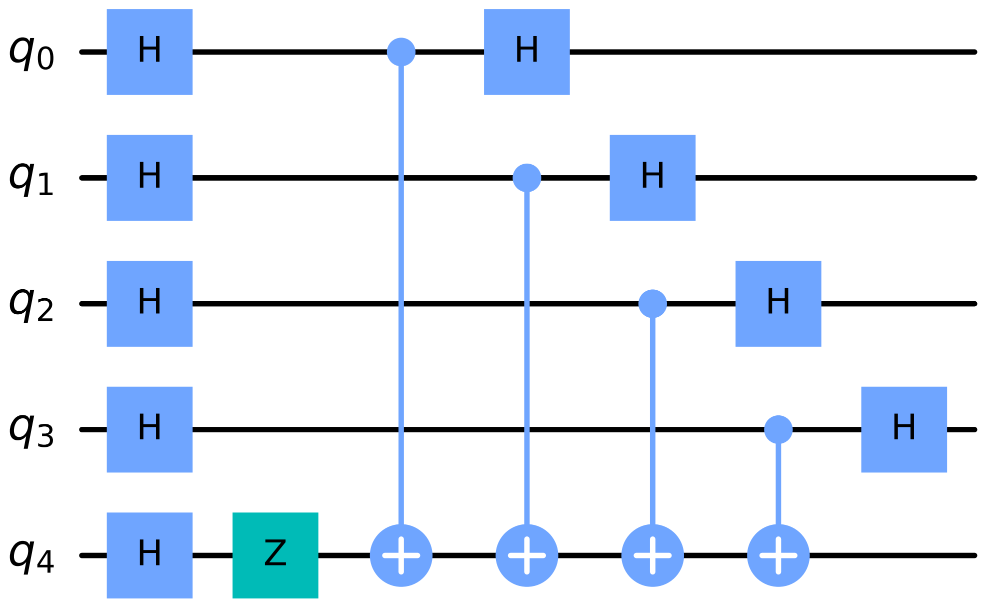
Figure 6.2: IBM's Vigo *coupling map*, indicating its topology. In this work this is used to select where the second fault propagates.

6.3 Injection on quantum circuit cuts

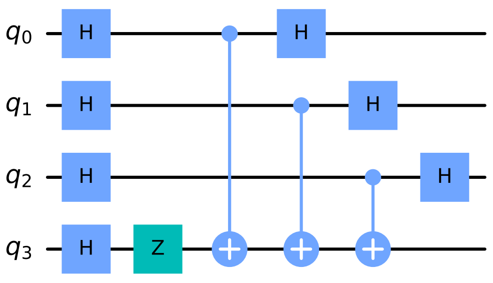
As introduced in Chapter 5, circuit cutting is a novel technique that allows splitting large quantum circuits into several smaller and independent subcircuits (or *circuit cuts*). In this thesis we want to study how transient faults impact the execution of these subcircuits, with a possible goal, for example, of selective hardening. If, in fact, we find that some subcircuits are more vulnerable to transient faults than others, an eventual future hardening technique can be used more efficiently for protecting the most vulnerable subcircuits.

6.3.1 Tested quantum circuits and their cuts

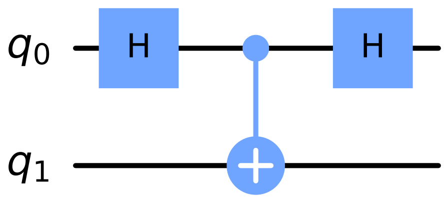
We performed this injection on three quantum circuits: Quantum Fourier Transform (4-qubits), Bernstein–Vazirani (4-qubits) and Deutsch–Jozsa (4-qubits). The QFT circuit and its cut solution are represented in Figure 5.1. The original circuit of 4 qubits is split into three subcircuits of 3 qubits each. We can describe the reduces size of the subcircuits using the number of operations and the *circuit depth*, with the latter being defined as the highest number of operations on a single qubit wire that can be found on the quantum circuit in consideration. In QFT's case, the subcircuits reduce the uncut circuit's depth of 7 to a depth of 6, 4, and 3, respectively, while the number of operations is reduced from 20 to 11, 6, and 3. The tested cut solutions of the BV and DJ circuits are shown respectively in Figures 6.3 and 6.4. They are both 4-qubit circuits with an additional *ancilla* qubit. They are split into two subcircuits of 4 (*3 + 1 ancilla*) and 2 (*1 + 1 ancilla*) qubits each. For BV, the original circuit has a circuit depth of 6, reduced to 5 and 3 respectively in the subcircuits. The number of operations, 14, is split in 11 and 3. For DJ, the number of operations, 22, is divided into 17 and 5 respectively, while the circuit depth goes from 6 of the uncut circuit to 5 for both subcircuits. Thus, we can say that for both BV and DJ the first subcircuit is considerably larger than the second.



(a) Bernstein-Vazirani full uncut circuit

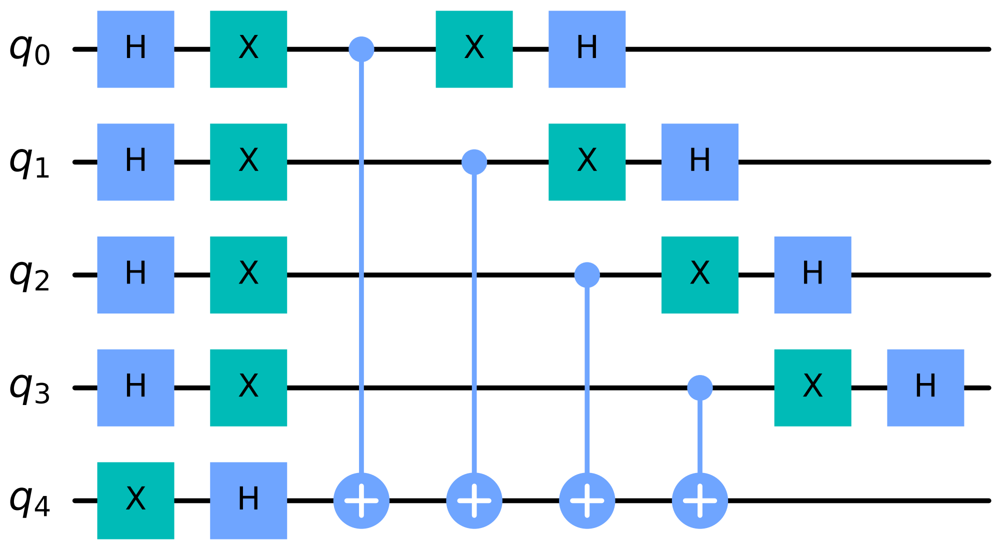


(b) Bernstein-Vazirani's subcircuit #0

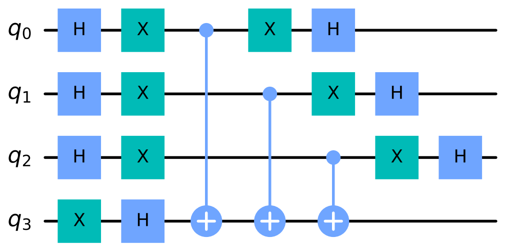


(c) Bernstein-Vazirani's subcircuit #1

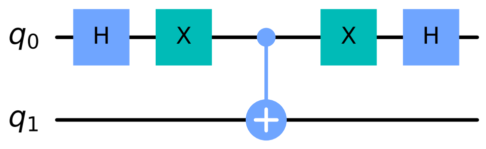
Figure 6.3: (a) The uncut BV circuit. Through the technique of circuit cutting, this circuit is split into two subcircuits, shown below it in (b) and (c).



(a) Deutsch–Jozsa full uncut circuit



(b) Deutsch–Jozsa's subcircuit #0



(c) Deutsch–Jozsa's subcircuit #1

Figure 6.4: (a) The uncut DJ circuit. As with BV, this circuit is split into two subcircuits, shown below it in (b) and (c).

6.3.2 Methodology and frameworks used

To select a cut solution and thus extract the subcircuit from the original circuit, we use a recently developed framework called CutQC [14]. CutQC is an implementation of the circuit cutting technique that given a quantum circuit to cut and some constraints, automatically finds a cut solution within those constraints. In addition, CutQC also optimizes for the cut solution with minimal classical overhead needed to reconstruct the final result in the classical postprocessing step.

Once the cut solution has been found and the subcircuits extracted, we perform our fault injection on each subcircuit. This is done exactly the same way as for the single fault injection, as described in section 4.2. Faults are simulated by inserting a parametrized U-gate in all possible positions of the subcircuit. The additional step that is required here is the recombination part: since we aim to study the impact that faults have on the final output of the circuit, each time we inject a fault on one subcircuit we then recombine its corrupt output with the output of the other, faultless, subcircuits. We can then measure the impact (i.e. compute the QVF value) of that specific fault in that specifc subcircuit on the final output of the circuit.

To recombine the subcircuits outputs we use MLFT's implementation (Maximum Likelihood Fragment Tomography) of circuit cutting [17], a technique that aims to reconstruct the "most likely" probability distribution defined by a quantum circuit, given the measurement data obtained from its subcircuits.

Chapter 7

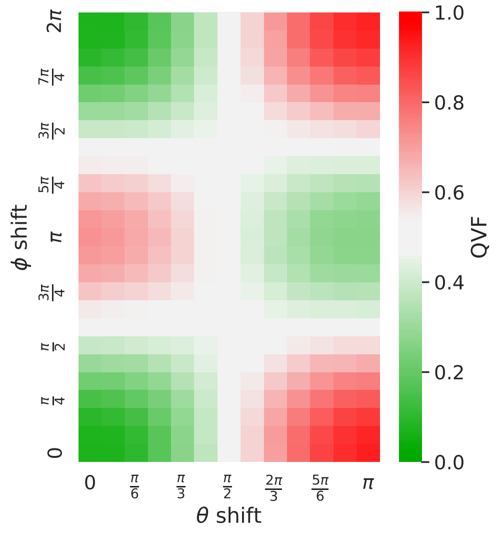
Experimental results

Having explained the process by which we inject the faults and the metric used to judge the outputs, we can in this section show the results of some injections.

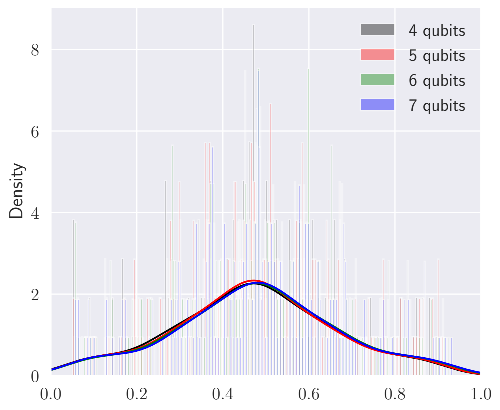
7.1 Single fault injection

First, here we discuss the single fault injection results, which means injecting one fault at a time exactly as described in section 4.2. In Figures 7.1, 7.2 and 7.3 there are the results for the three considered circuits, respectively: Bernstein-Vazirani, Deutsch-Josza and Quantum Fourier Transform. These take the form of a QVF heatmap (a) and a QVF distribution histogram; in the heatmap, as described in section 4.2, each combination of angles (θ, ϕ) is injected in each possible position of the circuit and a QVF value is calculated in each of those cases. The heatmap displays for each combination of (θ, ϕ) the QVF for that angle combination averaged over all positions in the circuit. The value of QVF is represented by color, with green indicating when it's close to zero, white when it's around 0.5 and red close to 1. We injected faults with values for $\theta = [0, \pi]$ and $\phi = [0, 2\pi]$, with steps of $\frac{\pi}{12}$. As expected, we can see in all circuits that the QVF for angles close to $(0, 0)$ is green, indicating that the output is similar to the faultless execution, which makes sense when injecting faults with very small angles. In the histograms, (b), we plot the distribution of QVF values for injections on circuits with increasing circuit sizes, from 4 to 7 qubits. This is both another way of visualizing the same information of the heatmaps (in the 4-qubit case), and also a way to compare different behaviours that may emerge when increasing the number of qubits of the circuit.

Looking at the whole heatmaps, we notice how there are green regions, red regions and white regions. We can then deduce that if a fault occurs with angles that fall in a green region ($QVF < 0.45$), on average, it will have no impact on the



(a) 4-qubit Bernstein-Vazirani single fault injection heatmap

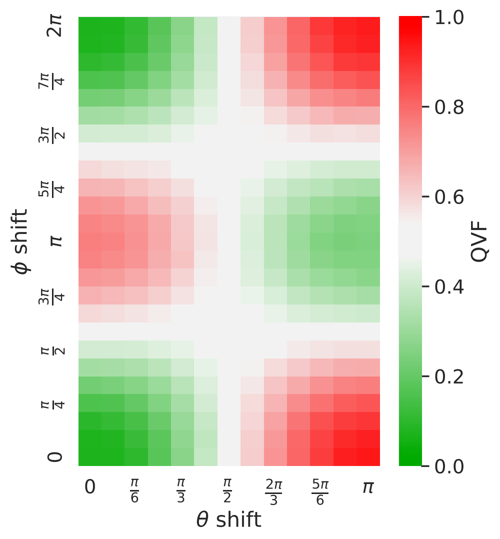


(b) Histogram of the QVF distribution for Bernstein-Vazirani considering increasing circuit size.

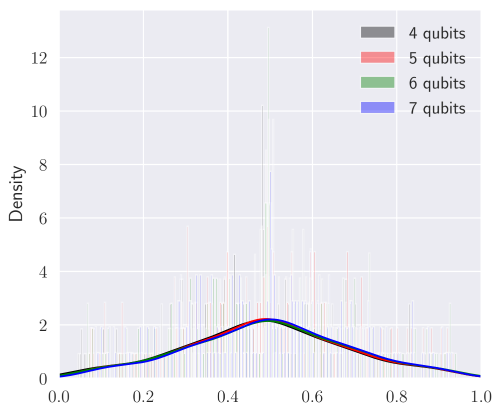
Figure 7.1: Results of single fault injection on BV circuit. (a) Shows the QVF heatmap for the 4-qubit circuit, (b) shows the QVF distribution histograms for increasing circuit sizes.

correctness of the output (the circuit will still produce the correct state as the most probable one). Otherwise if the angles fall in a white region ($0.45 < QVF < 0.55$), the output will be ambiguous and the correct state cannot be confidently selected. Lastly, if the angles fall in a red region ($QVF > 0.55$), the circuit will output a wrong state as the most probable one.

Figures 7.1 and 7.2 present the results for the single fault injection on the BV and DJ circuits respectively. The two circuits behave similarly, and the two heatmaps (a) show clearly defined red and green regions. To better analyze these results, let's consider shifting only one variable. For example, by keeping θ fixed at 0 and increasing ϕ (equivalent to going vertically up in the $\theta = 0$ column) we can learn that ϕ shifts are critical when they are around the value of π. The same can be said of θ values (going horizontally from left to right at $\phi = 0$). Interestingly though, when both θ and ϕ are around π, the QVF turns green; this seems to suggest that the combination of these angle shifts has a compensating effect resulting in a correct output. Looking instead at the histograms (b), they show that the vulnerability to faults in these two circuits does not change when you increase the size (number of qubits) of the circuit. These histograms plot the distribution of QVF values starting at 4 qubits (hence, the same data as the heatmaps) going



(a) 4-qubit DJ single fault injection heatmap



(b) Histogram of the QVF distribution for DJ considering increasing circuit size.

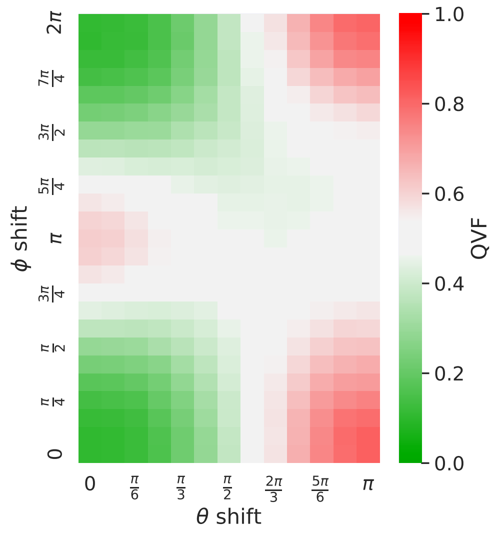
Figure 7.2: Results of single fault injection on DJ circuit. (a) Shows the QVF heatmap for the 4-qubit circuit, (b) shows the QVF distribution histograms for increasing circuit sizes.

to 7 qubits, and they all have the same distribution. In Figure 7.3 there are the results of the same analysis on the QFT circuit. On the left (7.3a) we can see the QVF heatmap for the 4-qubit QFT circuit. Unlike the heatmaps for BV and DJ, this one is not symmetric over ϕ, but presents a diagonal area with lower QVF. Lastly, considering 7.3b, this too shows a different behaviour than BV and DJ: QFT circuits of different sizes actually have different QVF distributions. In fact, when increasing the number of qubits the QVF tends to the average value (lower standard deviation, thus higher peak around 0.5). We can then say that, as QFT circuit scales up, the number of harmless faults is reducing and the probability to have a dubious output ($0.45 < QVF < 0.55$) increases.

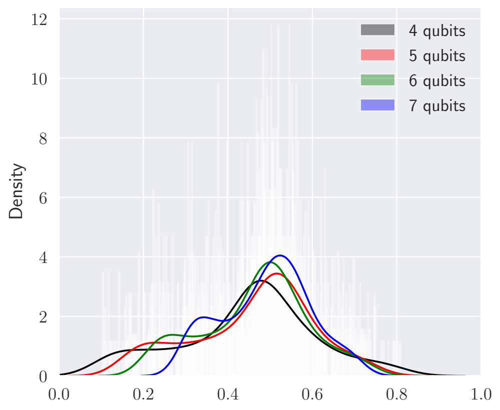
7.2 Double fault injection

In this section we will show the results of the double injection experiment. This is where, after injecting the first fault, we also inject a second fault on a neighboring qubit with a smaller magnitude. The process is explained more in detail in section 6.2.

In Figure 7.4 there are the results for the Bernstein–Vazirani circuit. In (a)



(a) 4-qubit QFT single fault injection heatmap



(b) Histogram of the QVF distribution for QFT considering increasing circuit size.

Figure 7.3: Results of single fault injection on QFT circuit. (a) Shows the QVF heatmap for the 4-qubit circuit, (b) shows the QVF distribution histograms for increasing circuit sizes.

is repeated, for reference, the QVF heatmap of the single fault injection of the BV circuit (it is a portion of Figure 7.1a). In (b) is figured the heatmap of the *double* fault injection. It is to be interpreted in a slightly different way than the heatmap of the single injection: since this time, for each (θ, ϕ) combination there isn't a single QVF value, but a series of QVF values, one for each different second fault (θ_1, ϕ_1). Thus, in order to still be able to construct the heatmap, we averaged all the QVF values of the second faults and put that average in the heatmap. This means that in Figure 7.4b, each (θ, ϕ) spot is the average QVF of all the possible configurations of first fault $((\theta, \phi)$, fixed) plus second fault $((\theta_1, \phi_1)$, ranging $\theta_1 \leq \theta$ and $\phi_1 \leq \phi)$. Since just performing an average doesn't give the full picture, we also added Figure 7.4c that depicts all the second faults (θ_1, ϕ_1) individually for a specific first fault $(\theta = \pi, \phi = \pi)$. This 3D graph has on the XY plane the angles of the second fault (θ_1, ϕ_1). The height of the points (Z axis) is the QVF for that specific second fault. As reference, it is also plotted as a grey XY plane the height of the QVF with only the first fault. As we can see, the plane is at the same height as the (0, 0) point, which correspond to a non-existing second fault.

As expected, the second injection worsens (increases) the average QVF in all cases. Specifically, the area around (π, π) was green in the single fault injection, but turns white/red in the double fault injection. These seem to be the angles most affected by the insertion of the second fault. This is better visually represented in Figure 7.5, where we show, for each (θ, ϕ) spot, the difference in QVF values between the single and the double fault injection. As we can see, there is no blue in the graph; meaning that the insertion of the second fault always increases the QVF. Additionally, we notice how the biggest increase in QVF is in the region around (π, π).

Additionally, Figure 7.4c shows the QVF for the Bernstein-Vazirani circuit obtained by fixing the phase shift in the first fault to (π, π) and injecting a series of second faults of $\theta_1 \leq \pi$ and $\phi_1 \leq \pi$. It can be interpreted as a depiction with increased granularity of the highlighted square in Figure 7.4b. It is interesting to observe a behavior that results in a lower QVF of the second injection when both θ_1 and ϕ_1 assume values closer to π, while the worst QVF values are found when only one of the two shifts is close to π and the other tends to 0.

Lastly, Figure 7.6 shows the distribution of QVF values for single (black) and double (red) fault injection on the Bernstein-Vazirani circuit. We can generally see how the QVF values for the double fault injection tend toward the right side of the graph, indicating a higher mean QVF; this is confirmed by computing the average: 0.46 for the single injection and 0.53 for the double injection. Additionally, the distribution for the double injection is also more concentrated at higher QVF values, with a standard deviation of 0.1839 against 0.1818 of the single fault injection. This data confirms the conclusion that a double fault has a worse effect on the output.

7.3 Injection on quantum circuit cuts

In this section we present the results of the injection on quantum circuit cuts. A description of circuit cutting can be found in Chapter 5, while the details of how we performed the injection are explained in 6.3. Briefly, quantum circuit cutting is a novel technique that allows to split a large quantum circuit into multiple smaller subcircuits. These subcircuits can be independently executed and their outputs recombined to reconstruct the final output of the original circuit. Here, we performed a fault injection on these subcircuits, aiming to better understand how a fault in one subcircuit propagates to the final reconstructed result of the full circuit. In addition, we also found how some subcircuits have different vulnerabilities to transient faults than other subcircuits, opening up possibilities for future selective hardening techniques.

7.3.1 Fault injection

We tested three quantum circuits of 4-qubits each: Deutsch–Jozsa (DJ), Bernstein–Vazirani (BV) and Quantum Fourier Transform (QFT). We show our results through a series of heatmaps, similar to the single and double fault injection. In the subcircuit heatmaps (for example (b), (c), (d) in Figure 7.7), each (θ, ϕ) spot represents how critical a fault of those angles occuring in that subcircuit is, when considering how much it corrupts the final reconstructed output. More specifically, to each (θ, ϕ) spot is assigned a QVF value (green-red spectrum), that is the average QVF for faults of those angles in that subcircuit (average over all position in the subcircuit).

In Figure 7.7 there are the results for the QFT circuit. In 7.7a is reported, for reference and easier comparison, the single fault injection on the full QFT circuit (with no circuit cutting). We can see here how the general trend is that the only harmless faults (green values) are when the θ shift is lower than $\frac{\pi}{2}$. This result is reasonable if you consider that a θ shift changes the qubit $|0\rangle - |1\rangle$ probabilty, and a shift of $\frac{\pi}{2}$ rotates the qubit vertically by half of a Bloch Sphere, which is enough to start inverting the $|0\rangle - |1\rangle$ probabilty. We can then say that a shift in θ is more crticial than a shift in ϕ. Instead, the criticality of a shift of both θ and ϕ simultaneously cannot be easily estimated. In Figures 7.7a, 7.7b, 7.7c are shown the heatmaps for the injection on QFT's subcircuits. At a glance, we can already see how the three subcircuits all have different vulnerabilities to different faults. For example, a phase shift of (π, π) leads to an ambiguous or incorrect output if injected in subcircuits 0 or 2, while being non-critical for subcircuit 1. Another interesting behaviour that we can observe is that subcircuit 2 seems to be more influenced by θ shifts rather than ϕ shifts, as we can see by its heatmap having a more *vertical* pattern: faults characterized by $\theta \leq \frac{\pi}{2}$ tend toward being non-critical, those with $\theta \approx \frac{\pi}{2}$ produce an ambiguous output and those with $\theta > \frac{\pi}{2}$ result in an incorrect state.

In Figures 7.8 and 7.9 are shown the heatmaps for the injection on BV and DJ circuits respectively. We can immediately see that they are similar; this is to be expected since BV is an extension of DJ and the circuits are similar. The first heatmap (a) is, as before, the single fault injection on the full circuit (no circuit cutting), added for reference. It is easy to notice that in both algorithms, subcircuit #0 (b) is very similar to the full circuit injection (a). This makes sense when you consider that for these circuits, subcircuit #0 is significantly larger than subcircuit #1, thus it has the biggest impact on the overall vulnerability to faults. This can be seen in the circuits drawings in Figures 6.3 and 6.4 respectively for BV and DJ. Another thing to notice in both cases is that subcircuit #1 is slightly less critical, in general, to transient faults. In fact, although maintaining the same general pattern in the heatmap, the red regions are less intense while the green ones are sometimes larger.

In Figure 7.10 are plotted the distribution histograms of QVF values for the injection on all three considered circuits (and their subcircuits). This is the same data as for the previous heatmaps in Figures 7.7, 7.8 and 7.9, but it is another way of displaying it. We can have a more general view of the distribution of QVF values over the full fault injection, while also providing the mean ($\bar{X}$) and the standard deviation (σ). For instance in the histogram for the QFT circuit, in 7.10a, we can see that the uncut circuit has a QVF distribution more concentrated around the middle, with a mean of 0.44 and a low standard deviation (0.17), which translates to a higher number of white values in the heatmap with respect to its subcircuits. Subcircuit #0 instead reduces the height of the peak and shifts some values to a lower QVF, meaning that its heatmap is overall more green, as can be seen in Figure 7.7b. Subcircuits #1 and #2 both further lower the height of the peak, meaning a more spread out distribution of values, with subcircuit #2 having the highest mean QVF, 0.5, indicating a higher sensitivity to faults. Figures 7.10b and 7.10c show the histograms for BV and DJ circuits respectively. Both circuits perform similarly, with subcircuit #0 being the more critical one in both cases.

7.3.2 Selective hardening

While an effective hardening technique does not yet exist for quantum circuits, we can use this data to estimate the impact of an eventual selective hardening technique. For example, here we assume that we can completely protect one subcircuit from all faults, and we estimate how that would impact the final output of the full circuit. We do this by replacing the QVF value of injections on that subcircuit with a QVF equal to its QVF in ($\theta = 0, \phi = 0$). We choose this approach instead of just assuming its QVF to be always zero in order to preserve any intrinsic noise in the computation. We show these results again through heatmaps and histograms.

In Figure 7.11 are shown the results of this selective hardening estimation for the QFT subcircuits. Figures 7.11a, 7.11b and 7.11c correspond to the QVF value heatmaps for the full circuit assuming one specific subcircuit to be completely shielded from faults, like described before. They represent subcircuit #0, #1 and #2 respectively. This result, for example, shows that if we are able to completely protect subcircuit #0 from transient faults, it would have the biggest positive impact on the vulnerability of the full circuit execution with respect to the other subcircuits. This result might seem at first counter-intuitive, since subcircuit #0 is not the most critical one: it is in fact the subcircuit with the lowest mean QVF (Figure 7.10a). But this is explained by the fact that subcircuit #0 is the biggest one, as shown in 5.1, and so has the largest impact when shielded. Figure 7.11d shows the same data as the previous three figures, but in histogram form for easier comparison, and we can clearly confirm here that protecting subcircuit #0 offers the lowest mean QVF. Figures 7.12 and 7.13 present the results of the selective hardening estimate on BV and DJ respectively. The two circuits and subcircuits behave similarly; in both cases, subcircuit #0 is overwhelmingly the most important one to protect. As before, this is due to its bigger size, as shown in Figures 6.3 and 6.4.

This selective hardening analysis helps to identify which subcircuit is the most important one to protect from radiation. Although no such protection technique exists today, recent studies [10] have shown that ionizing radiation is a problem for quantum computers, and thus some hardening technique may be necessary.

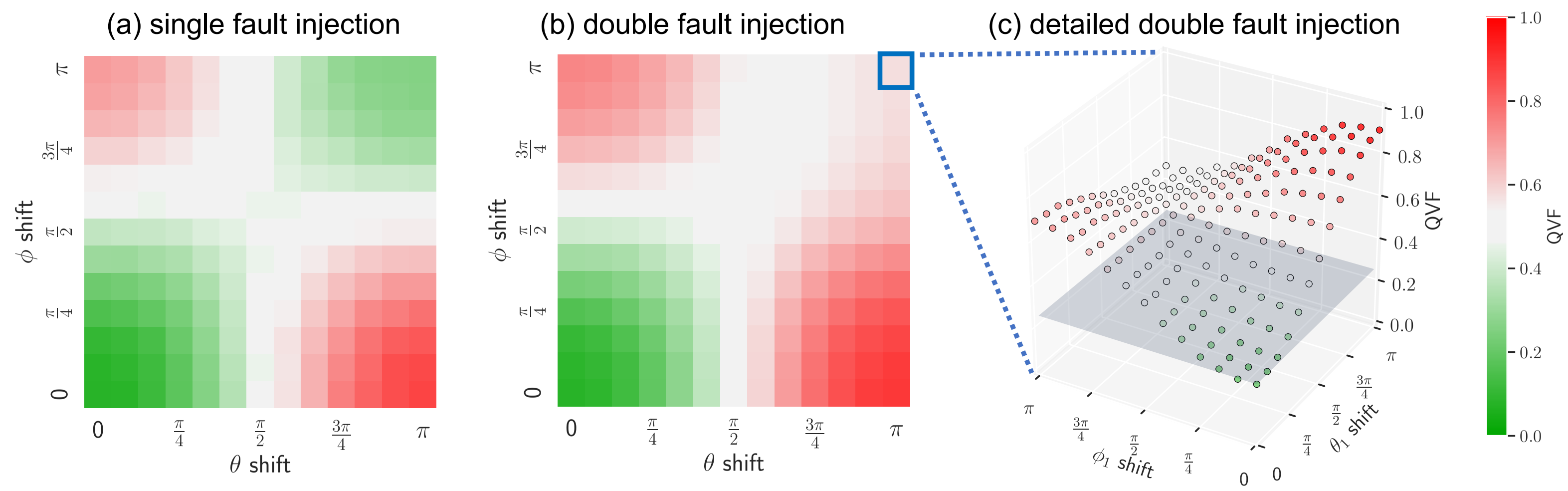


Figure 7.4: QVF for Bernstein-Vazirani for (a) Single and (b) Double fault injection. (c) Details of the QVFs for all the possible double faults injections, with the first fault injection fixed to (π, π).

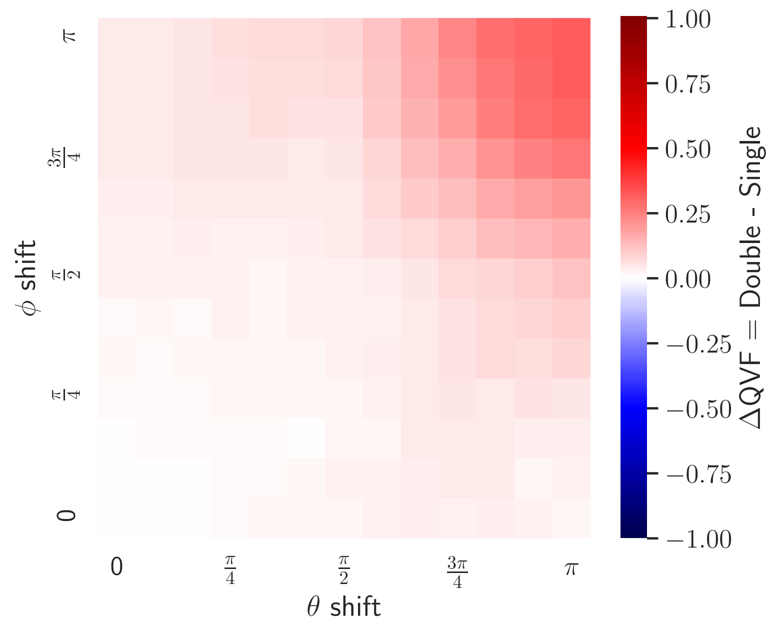


Figure 7.5: Difference of the QVFs for Bernstein-Vazirani between single and double fault injection.

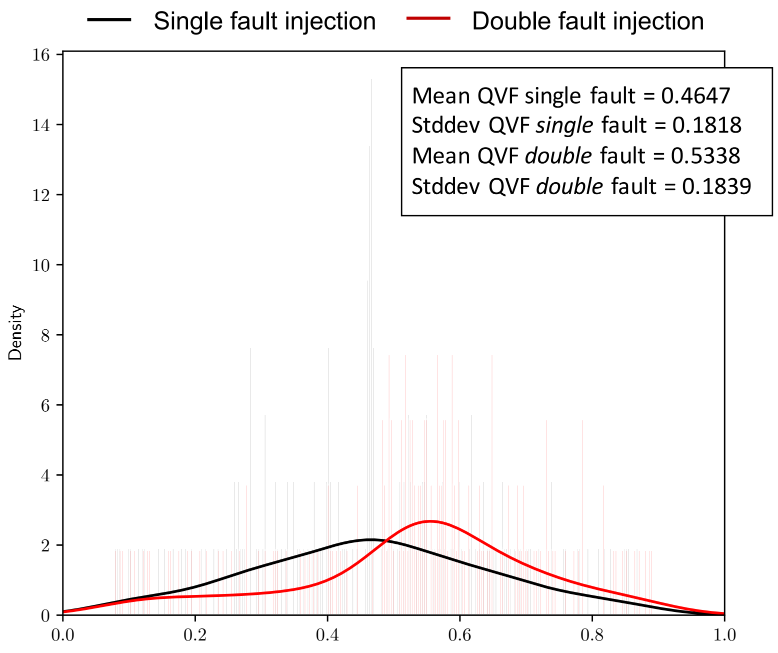
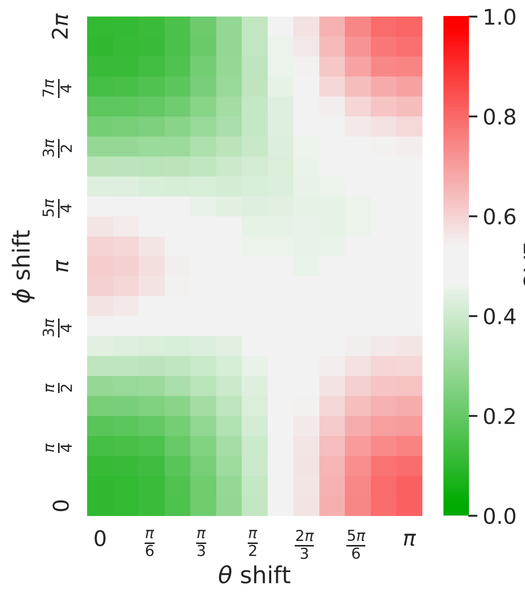
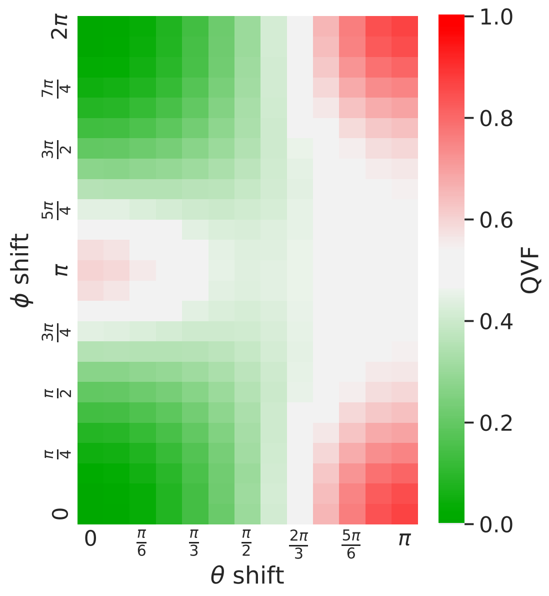


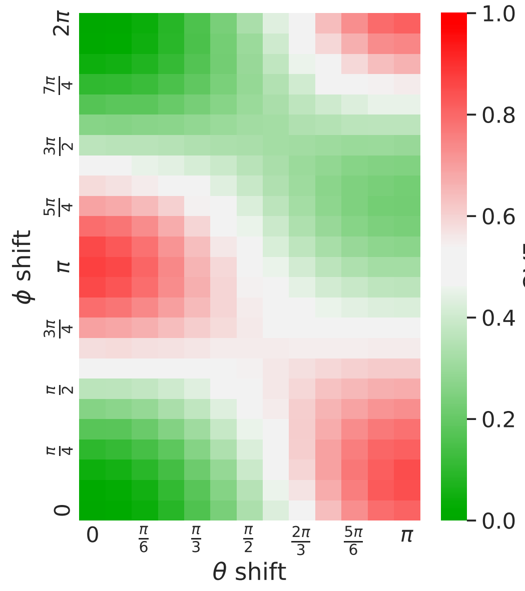
Figure 7.6: Bernstein-Vazirani distribution histograms for single and double fault injection.



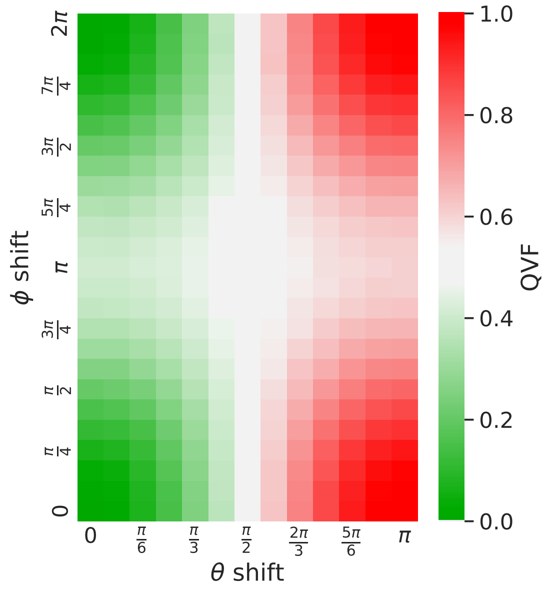
(a) Uncut QFT circuit



(b) Subcircuit 0

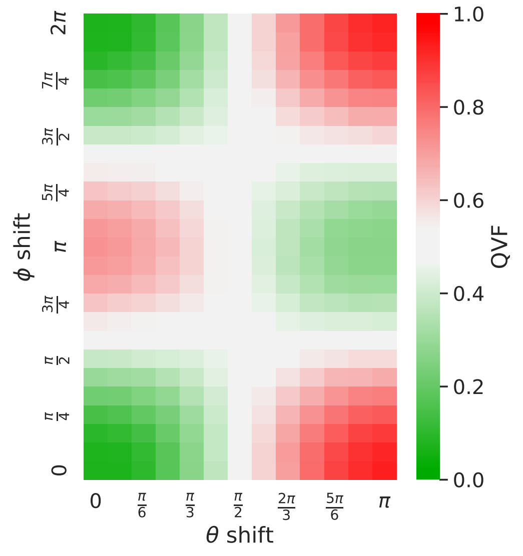


(c) Subcircuit 1

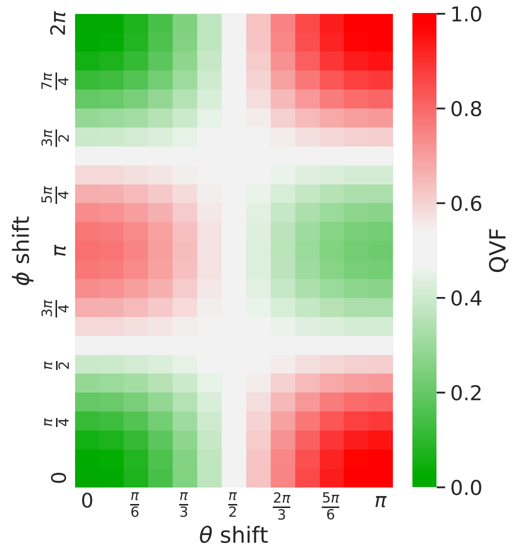


(d) Subcircuit 2

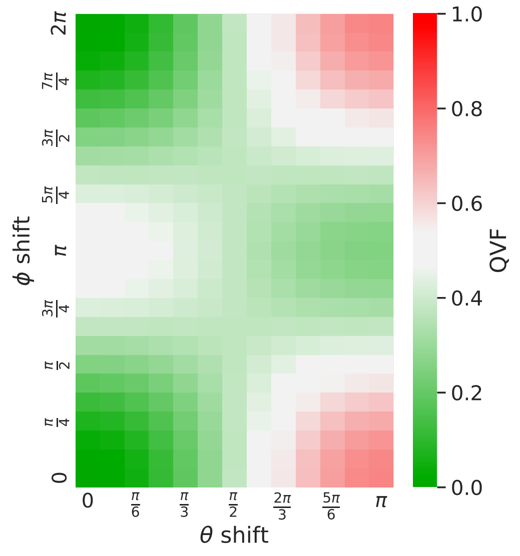
Figure 7.7: QVF heatmaps for the QFT circuit and the subcircuits. The green color indicates a low QVF (the correct state can be confidently selected), the red color indicates a higher QVF (an incorrect output is more likely to be selected), and the white color indicates a dubious output (i.e., correct and incorrect states have about the same probability).



(a) Uncut BV circuit

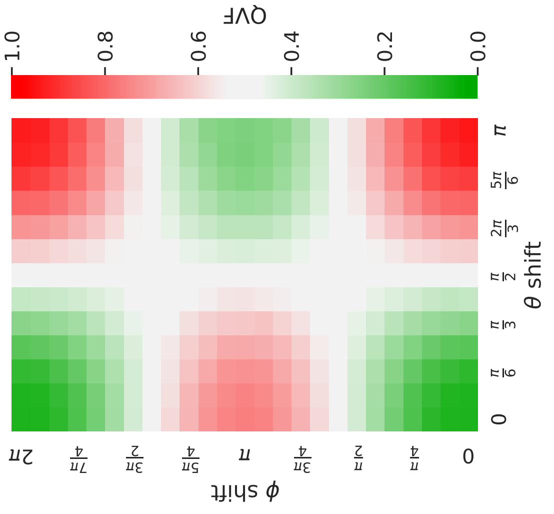


(b) Subcircuit 0

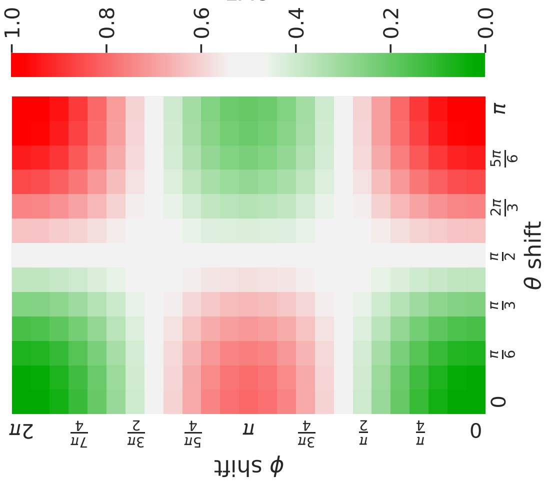


(c) Subcircuit 1

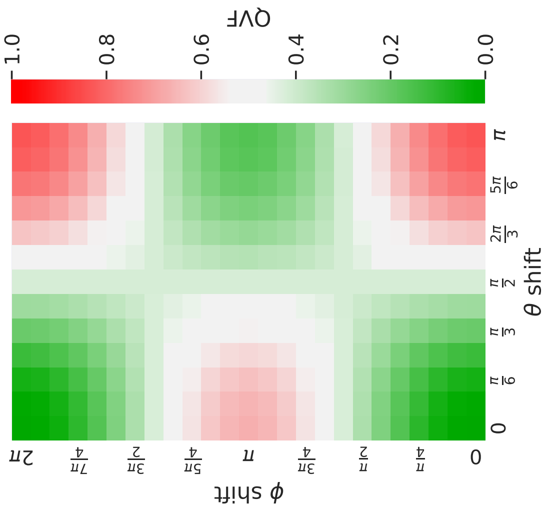
Figure 7.8: QVF heatmaps for the BV circuit and the subcircuits.



(a) Uncut DJ circuit

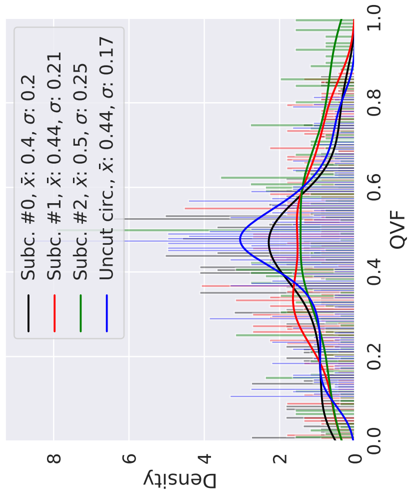


(b) Subcircuit 0

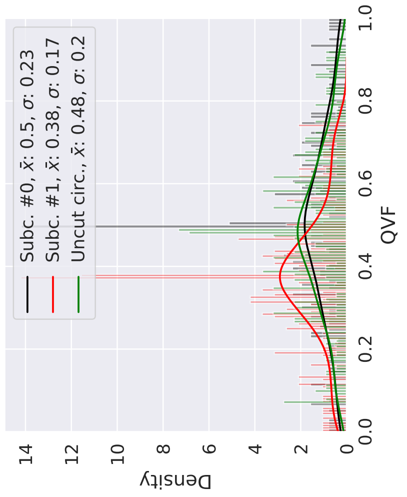


(c) Subcircuit 1

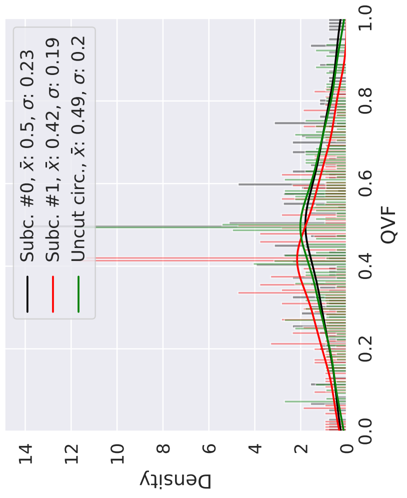
Figure 7.9: QVF heatmaps for the DJ circuit and the subcircuits.



(a) QFT

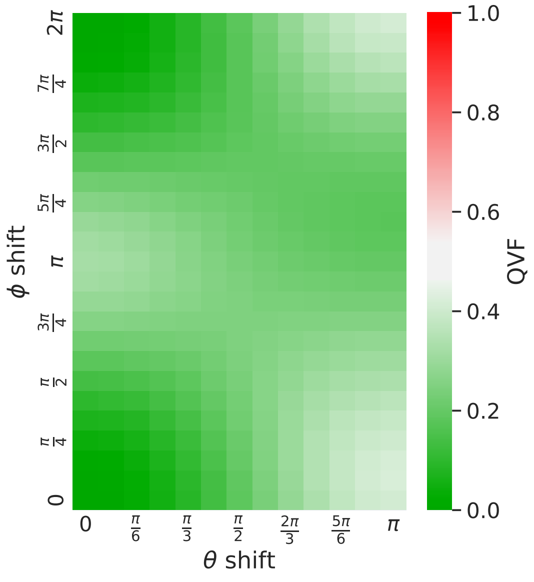


(b) BV

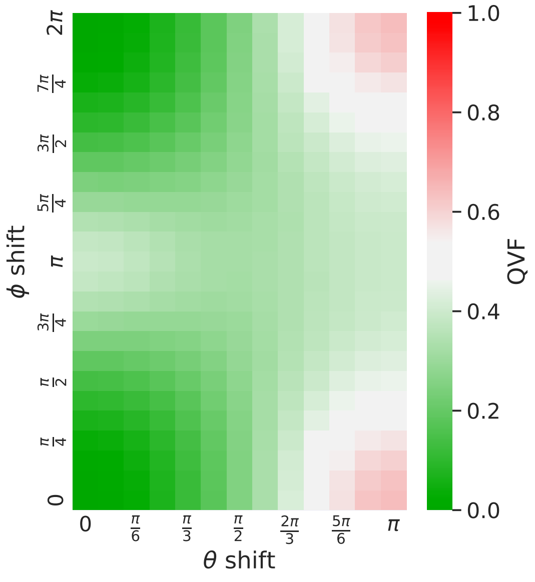


(c) DJ

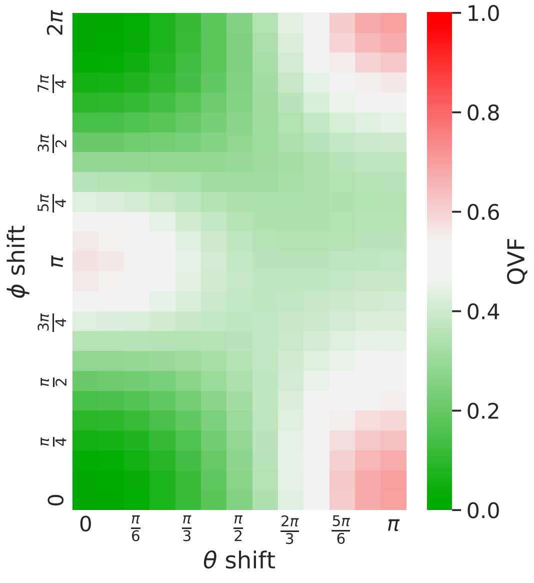
Figure 7.10: Histograms of the QVF distribution for QFT, BV and DJ circuits.



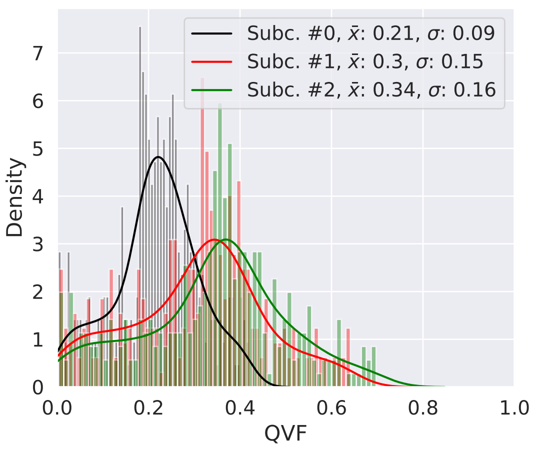
(a) Protection of QFT's subcircuit 0



(b) Protection of QFT's subcircuit 1

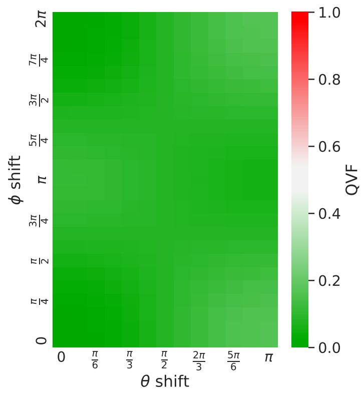


(c) Protection of QFT's subcircuit 2

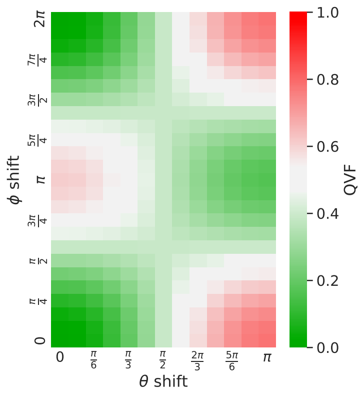


(d) Histogram of protected QFT subcircuits

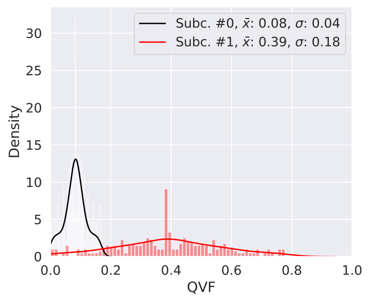
Figure 7.11: Heatmaps assuming complete protection from faults for each of the QFT subcircuit, and the histogram showing their distributions.



(a) Protection of BV's subcircuit 0

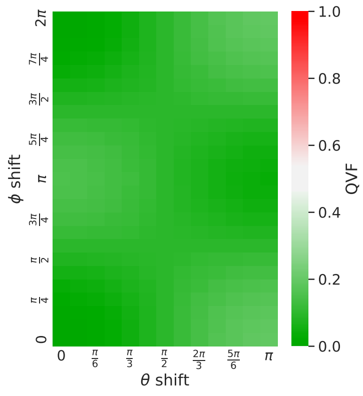


(b) Protection of BV's subcircuit 1

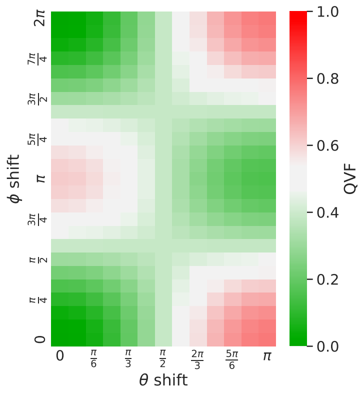


(c) Histogram of protected BV subcircuits

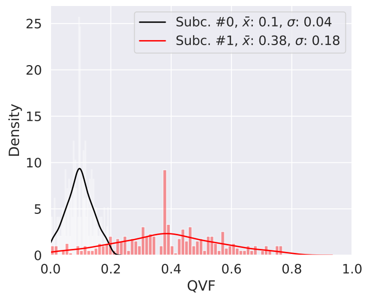
Figure 7.12: Heatmaps assuming complete protection from faults for each of the BV subcircuit, and the histogram showing their distributions.



(a) Protection of DJ's subcircuit 0



(b) Protection of DJ's subcircuit 1



(c) Histogram of protected DJ subcircuits

Figure 7.13: Heatmaps assuming complete protection from faults for each of the DJ subcircuit, and the histogram showing their distributions.

Chapter 8

Conclusions

The work done in this thesis is an attempt to better understand fault propagation in quantum circuits. Recent studies have shown how critical transient faults can be to quantum devices [11], and we believe that studying the sensitivity of quantum circuits to these faults can be a first step towards mitigating their impact. We learned that each quantum circuit can have different fault propagation characteristics, helping us to identify which faults and which qubits are more critical. We also looked at quantum circuit cutting and found that the various subcircuits into which a circuit is cut can have significantly different vulnerabilities to faults between one another.

These results lead us to suggest how this work could be used for helping future hardening techniques. Indeed, if each circuit has its own sensitivity to faults, these hardening efforts can be more efficiently spent by concentrating them on the more critical portions of the quantum device.

Bibliography

[1] Maximilian Schlosshauer, Johannes Kofler, and Anton Zeilinger. «A snapshot of foundational attitudes toward quantum mechanics». In: *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics* 44.3 (2013), pp. 222–230. ISSN: 1355-2198. DOI: `https://doi.org/10.1016/j.shpsb.2013.04.004`. URL: `https://www.sciencedirect.com/science/article/pii/S1355219813000397` (cit. on p. 1).

[2] Paul Benioff. «The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines». In: *Journal of statistical physics* 22.5 (1980), pp. 563–591 (cit. on p. 1).

[3] Richard Phillips Feynman. «Simulating physics with computers». In: *International Journal of Theoretical Physics* 21 (1982), pp. 467–488 (cit. on p. 1).

[4] Anumita Das and David Barry. *superquantumphysics*. URL: `https://sites.google.com/site/superquantumphysics/the/the-double-slit-experiment` (cit. on p. 4).

[5] *Nature Noon article about the double slit experiment*. URL: `https://naturenoon.com/double-slit-experiment-simple/` (cit. on p. 5).

[6] *Wave superposition*. URL: `https://en.wikipedia.org/wiki/File:Interference_of_two_waves.svg` (cit. on p. 6).

[7] Andreas Ketterer. «Modular variables in quantum information». In: (Oct. 2016) (cit. on p. 9).

[8] Amira Abbas et al. *Learn Quantum Computation Using Qiskit*. 2020. URL: `http://community.qiskit.org/textbook` (cit. on pp. 14–16).

[9] *Cosmic rays*. URL: `https://cds.cern.ch/images/CMS-PHO-GEN-2017-008-1` (cit. on p. 18).

[10] Antti P Vepsäläinen et al. «Impact of ionizing radiation on superconducting qubit coherence». In: *Nature* 584.7822 (2020), pp. 551–556 (cit. on pp. 18, 42).

[11] Matt McEwen et al. «Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits». In: *Nature Physics* 18.1 (2022), pp. 107–111 (cit. on pp. 18, 19, 51).

[12] Daniel Oliveira, Edoardo Giusto, Betis Baheri, Qiang Guan, Bartolomeo Montrucchio, and Paolo Rech. «A Systematic Methodology to Compute the Quantum Vulnerability Factors for Quantum Circuits». In: *arXiv preprint arXiv:2111.07085* (2021) (cit. on pp. 21, 22, 25).

[13] Tianyi Peng, Aram W. Harrow, Maris Ozols, and Xiaodi Wu. «Simulating Large Quantum Circuits on a Small Quantum Computer». In: *Physical Review Letters* 125.15 (Oct. 2020). DOI: `10.1103/physrevlett.125.150504`. URL: `https://doi.org/10.1103%5C%2Fphysrevlett.125.150504` (cit. on p. 26).

[14] Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. «CutQC: using small Quantum computers for large Quantum circuit evaluations». In: *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, Apr. 2021. DOI: `10.1145/3445814.3446758`. URL: `https://doi.org/10.1145%5C%2F3445814.3446758` (cit. on pp. 26, 33).

[15] Nadir Casciola, Edoardo Giusto, Emanuele Dri, Daniel Oliveira, Bartolomeo Montrucchio, and Paolo Rech. «Understanding the Impact of Cutting in Quantum Circuits Reliability to Transient Faults». In: (2022) (cit. on p. 27).

[16] C. D. Wilen et al. «Correlated charge noise and relaxation errors in superconducting qubits». In: *Nature* 594.7863 (2021), pp. 369–373. DOI: `10.1038/s41586-021-03557-5`. URL: `https://doi.org/10.1038/s41586-021-03557-5` (cit. on p. 30).

[17] Michael A Perlin, Zain H Saleem, Martin Suchara, and James C Osborn. «Quantum circuit cutting with maximum-likelihood tomography». In: *npj Quantum Information* 7.1 (2021), pp. 1–8 (cit. on p. 34).