The Pennsylvania State University

The Graduate School

LOGIC OBFUSCATION OF QUANTUM CIRCUITS FOR SECURITY

A Thesis in

Electrical Engineering

by

Aakarshitha Suresh

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

December 2021

The thesis of Aakarshitha Suresh was reviewed and approved by the following:

Swaroop Ghosh

Associate Professor, Electrical Engineering

Thesis Advisor

Morteza Kayyalha

Assistant Professor, Electrical Engineering

Kultegin Aydin

Department Head and Professor, Electrical Engineering

Abstract

An efficient compilation of quantum circuits is paramount to obtain reliable results in Noisy Intermediate-Scale Quantum (NISQ) regime. Efficient Compilation would involve lesser addition of extra gates that are noisy that would otherwise degrade the circuit performance. Several 3rd party compilers are evolving to offer faster compilation times and smaller depths/gate counts. They allow for more optimal compilation, making them more useful for complex circuits where addition of extra noise affects the functionality itself. These 3rd party compilers could just be a certain version of an otherwise trustworthy compiler, hence are untrusted which could allow an adversary to clone and/or Reverse Engineer (RE) the quantum circuit. These malicious attacks of any type can prove to be a bigger threat for the quantum IP vendor or the circuit designer, putting years of critical research and highly confidential information at risk.

As a solution, obfuscation of quantum circuits by intentionally corrupting the functionality by targeted insertion of dummy CNOT gates is proposed. If the untrusted compiler clones the design, they will get incorrect functionality. The designer will remove the dummy gates post compilation to restore the correct functionality. Since the quantum chip does not reveal the circuit functionality, the adversary cannot guess the dummy gate and its location or validate his guess using an oracle model as there is a lack of a golden reference model. A metric that is used to perform targeted dummy gate insertion is developed using various features of the quantum circuit. This ensures maximum corruption of functionality(or obfuscation), measured using Total Variation Distance (TVD). These results are also validated using IBM's default noisy simulation model. Our metric guided dummy gate insertion process achieves TVD of up to 28.83% and performs 10.14% better than the average TVD and performs within 12.45% of the best obtainable TVD for the benchmarks.

Table of Contents

List of Figures

List of Tables

List of Symbols

ψ Used to represent a qubit state, 1

$|.\rangle$ Notation to denote a qubit, 1

θ The angle between z-axis and the vector or the colatitude, 1

ϕ The angle between the vector and the x-axis or the longitude, 1

Acknowledgments

I would first like to thank my thesis advisor Prof. Swaroop Ghosh, Associate professor, Electrical Engineering, Pennsylvania State University. His constant guidance helped me face any type of obstacle during my thesis work without any doubts. He always motivated me to push myself even beyond my capabilities to do great progress in my research at Pennsylvania State University. I would also like to thank my committee member, Prof. Morteza Kayyalha, at the department of Electrical Engineering, Pennsylvania State University, who guided me during the presentation of this thesis, and discussed several future work ideas.

I thank Dr. Rasit Onur Topaloglu at IBM, for his constant support and useful advice. His effective ideas, helped me breakthrough some challenges I faced in my work. I also acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

I want to thank NSF, DARPA, SRC, Penn State ICDS, and the Huck Institute of Life Sciences for their financial support. The dissertation would not have been possible without their financial backing.

This material is based upon work supported by the NSF (Award CNS- 1722557, CCF-1718474, DGE-1723687, DGE-1821766, DGE-2113839, and OIA-2040667), DARPA Young Faculty Award (Award D15AP00089 and D16AP00109), SRC (Award 2657.001 and 2847.001), and seed grants from Penn State ICDS and Huck Institute of Life Sciences. Any opinions, findings, and conclusions, or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the awarding agencies.

I would also like to thank my colleagues, Abdullah Ash Saki and Mahabubul Alam, who have helped me in my work. Finally, I would like to thank my parents, my sister, and my friends, Kruthika, Srujan, Aishwarya, Srikanth, Balachandran, Ranjitha and my other family who have been my constant support system, and encouraged me whenever the goal seemed difficult to attain. A special mention to Sherrydawn Jackson and the department of Electrical Engineering. This accomplishment would not have been possible without the people mentioned above and many more who were not mentioned; therefore, I thank you all sincerely for all the support and help.

Chapter 1

Introduction

Quantum Computing has brought a phenomenal change in the world of computation. One of the earliest works in this field that generated interest was by Peter Shor in 1994, who developed an efficient algorithm known as the Shor's algorithm that factors numbers into primes faster than any other existing classical algorithm. Tremendous progress was later achieved when several other quantum algorithms were developed to solve various problems which classically were computationally complex. While the superiority of quantum computers over classical computers in terms of computation ability is arguable, there is certainly an advantage of using quantum computers over classical computers for certain problem instances. The concept that quantum computers can solve problems faster than the fastest classical computers can is known as Quantum Supremacy.

Quantum Computing can provide exponential speed-up over classical counterparts to solve various combinatorial problems e.g., data analytics and material discovery. The computing power of quantum computers is growing due to rapidly evolving noise mitigation techniques [2], ever-increasing number of qubits, and improved error rates

and decoherence times [3]. Hybrid classical-quantum computing using shallow depth variational algorithms e.g., Quantum Approximate Optimization Algorithm (QAOA) [4] has been explored to compute approximate solutions in the presence of noise. The problem-specific parametric quantum circuits designed using variational algorithms e.g., QAOA to solve certain problems embed the topology of the problem which is an asset. For applications like power grid (or other critical infrastructure) optimization, the client would like to keep the problem information confidential. Non-parametric circuits, e.g., Quantum Fourier Transform (QFT) that are optimized for higher success probability with lower gate count and depth also contain Intellectual Property (IP).

1.1 Introduction to Quantum Circuits

Each quantum circuit comprises of fundamental bits of computation called Qubits. Analogous to how a classical bit is used to represent an output state of binary 0 or 1, the field of quantum computing uses **QUBIT** or "**QU**antum **BIT**" to represent the output states. Qubit state is expressed with a *ket* ($|.\rangle$) notation. Each qubit state represented by, $|\psi\rangle$, is described as $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$. For a single qubit, there are two orthogonal basis states $|0\rangle$ and $|1\rangle$. For example, in a multi-qubit system with N qubits, there will be 2^N computational z-basis states. Associated with each computational basis state is a probability amplitude α_i, which is a complex number. Here, these probability amplitudes are a and b which are complex numbers such that their probabilities add up to one, i.e., $|a|^2 + |b|^2 = 1$. Fig. 1.1 shows a Bloch sphere which is a graphical way to represent a qubit

state $|\psi\rangle$. A bloch sphere has two orthogonal basis states on opposite sides of the sphere for each axis basis. Eqns. 1.1, 1.2 show the two orthogonal x-basis and y-basis states respectively. When a qubit is in $|+\rangle$ or $|-\rangle$ state, a measurement in x-basis will give 0(1), and if it is in $|R\rangle$ or $|L\rangle$ state, which is used to represent $|+i\rangle$ and $|-i\rangle$ respectively, then, a measurement in y-basis will give 0(1). By default, all qubits are initialized in the $|0\rangle$ state in the z-basis. Measurement can be done in any other basis as well.

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \, |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{1.1}$$

$$|R\rangle = \frac{|0\rangle + i\,|1\rangle}{\sqrt{2}} \, |L\rangle = \frac{|0\rangle - i\,|1\rangle}{\sqrt{2}} \tag{1.2}$$

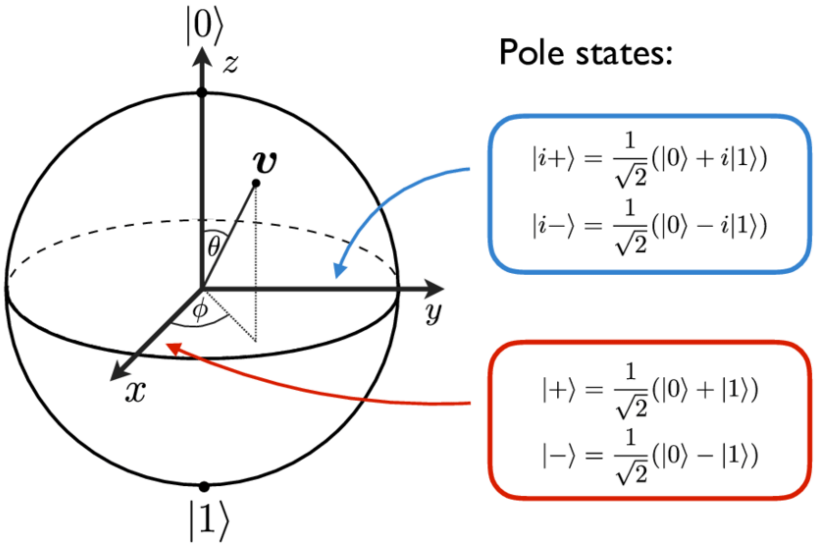

Figure 1.1. Bloch sphere representation of v.

It is known that the total probability of the system has to be 1 by the laws of quantum mechanics therefore $\| \, |\psi\rangle \, \|^2 = 1$. Eqn. 1.3 shows how a vector qubit state $|\psi\rangle$ can be written in terms of θ and ϕ. Here, θ is the angle between z-axis and the vector or the

colatitude and ϕ is the angle between the vector and the x-axis or the longitude.

$$|\psi\rangle \, or \; \upsilon = \cos\frac{\theta}{2}|0\rangle + e^{(i\phi)}\sin\frac{\theta}{2}|1\rangle \tag{1.3}$$

Each qubit can present three special properties such as Superposition, Entanglement, and Interference which can be described as follows:

- *Superposition* - Superposition is described as the ability of a qubit in quantum computing to be a combination of more than a single state at the same time. This means, for example, unlike the classical bits, which either stay in 0 or 1 for a single bit output, the qubits are a combination of both these output states at the same time. This means that for a qubit, the probability of measuring a single qubit is neither completely 100% nor 0% as 0 or 1 . This also means that multiple measurements made on identical qubits under same conditions also might not give identical results always.

- *Entanglement* - Consider some form of interaction between two qubits. If these two qubits are physically separated from each other, and only one of them is measured, it still gives some amount of information about the other qubit as if they are "entangled" with each other. This means when one qubit's state is changed, the other qubit in the pair also gets affected, instantaneously, as if they are connected. This means that their physical properties are correlated to each other, and their states are described with respect to each other.

- *Interference* - Interference is when the wave functions of the qubits can interfere

with each other either to add or to subtract and cancel out (reinforce or diminish). This interference effect can be further used to bias the measurement of a qubit towards a particular state or states.

1.1.1 Quantum Gates

Similar to logic gates in classical circuits, the operations that modulate the state of qubits and perform computations are described as Quantum gates that are represented by $2^n \times 2^n$ unitary matrices (n = number of qubits) and can work on a single qubit (e.g., X (NOT) gate) or on multiple qubits (e.g., 2-qubit CNOT/CX gates). Some of the basic quantum gates are described as follows:

- Hadamard gate- A Hadamard gate is a simple and most commonly used quantum gate in computing. It is a single qubit operation that converts $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ that is $|+\rangle$ and $|-\rangle$ respectively.

- Pauli Gates- The Pauli X gate performs the operation of negation. It acts like a logic not gate, thus turning $|0\rangle$ into $|1\rangle$ state and vice versa. It is a π radians rotation about the x-axis of the bloch sphere, also known as bit-flip. The Pauli-Y gate is a single qubit operation. It maps $|0\rangle$ to $-i * |1\rangle$ and $|1\rangle$ to $i * |0\rangle$. It equates to a rotation around the y-axis of the bloch sphere by pi radians. The Pauli-Z gate is a single qubit operation. It maps $|1\rangle$ to $-|1\rangle$ and it leaves $|0\rangle$ unchanged. It equates to a rotation around the z-axis of the bloch sphere by π radians.

- Controlled Gates- These are the type of quantum gates that can have one or

more control qubits that modulate the operation of the other target qubit. Some examples could be the CX, CY and CZ gates.

1.1.2 Basis gates and coupling constraints

A practical quantum computer normally supports a limited number of single and multi-qubit gates known as *basis gates* or native gates of the hardware. For instance, IBM quantum computers have the following native gates: I, RZ, SX, X (single-qubit), and CX-CNOT gate(two-qubit). However, the quantum circuit may contain high-level gates that are *decomposed* into the basis gates before execution. Besides, the two-qubit operation (CNOT) is only permitted between directly connected qubits. These limitations in two-qubit operations in any target hardware are also known as *coupling constraints*.

1.1.3 Quantum Circuits

Fig. 1.2 represents the quantum circuit which is read from left to right. Inputs are generally reset to zero state, and so if a qubit must be given an input of one, then a not gate is used to flip the zero back to one state. Qubits are listed in the top to bottom order, as Q0, Q1, etc. The outputs are taken at the end of the lines in the form of M or measured qubits, which means that the output if observed on one, more or all the qubits, then these qubits have the M symbol, as the output states are measured. Further, the different gates that are used to implement the given function in the circuit are placed on the required qubit lines, and these can be one or multiple-qubit gates.

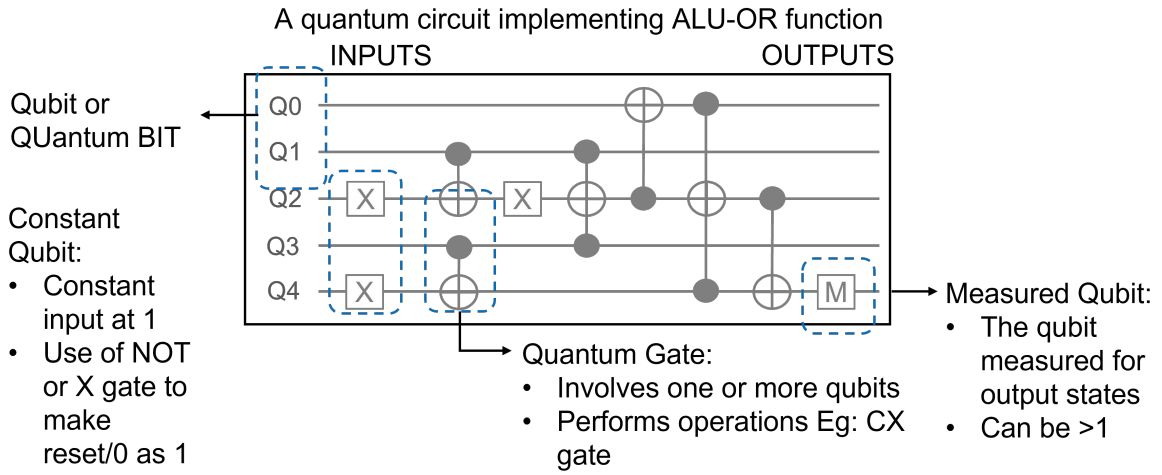

Figure 1.2. Quantum circuit fundamentals.

1.1.4 Compilation

Quantum circuit compilers e.g., Qiskit [5] perform necessary modifications (e.g., insert SWAP gates) to the input circuits to meet coupling constraints of the hardware. Besides, compilers offer higher-level circuit optimization through single/multi-qubit gate cancellation, rotation merging, and gate-reordering [6].

1.1.5 Quantum Hardware

Quantum computing field has improved a lot over these years, in the sense of implementation techniques, including the fact that quantum circuits can be run on actual quantum hardware through the cloud, and not just be simulated. For example, IBM Cloud offers access to real quantum hardware for any researchers and students [7]. Around 1 trillion circuits have been run until now on these hardware with over 65 powerful qubits across all the IBM machines openly available on the cloud.

All quantum systems deployed by IBM Quantum are based on superconducting

IBMQ Systems		IBMQ Simulators		IBMQ retired systems	
System	Qubit Count	Simulator	Qubit Count	System	Qubit Count
ibmq_santiago	5	ibmq_qasm_simulator	32	ibmq_manhattan	65
ibmq_jakarta	7	simulator_statevector	32	ibmq_almaden	20
ibmq_toronto	27	simulator_mps	100	ibmq_melbourne	15
ibmq_mumbai	27	simulator_stabilizer	5000	ibmq_vigo	5
ibmq_montreal	27	Simulator_extended_stabilizer	63	ibmq_valencia	5
ibmq_dublin	27			ibmq_athens	5

Table 1.1. IBM Quantum Systems and Simulators

qubit technology. The control and scalability of this technology pave a clear path to achieving quantum advantage with these systems. There are different Quantum behaviour simulators and IBM Quantum systems that are accessible publicly through the cloud. These systems along with their qubit count are listed in Table 1.1.

1.2 Classical versus Quantum Circuits - Differences and Challenges

Classical computing has advanced a lot over the years with the advent of fast and powerful domain specific hardware chips like Graphic Processing Units(GPUs) which are used to perform complex simultaneous calculations. Traditionally, classical computers manipulate 1's and 0's to perform complex operations on several bytes of data. However, today our best estimate suggests that at least 2.5 quintillion bytes of data is produced every day with 1.7 megabyte of data created every second [8]. Processing of this data and extraction of insights requires compute power beyond the capabilities of classical computers. The discovery of quantum computing that harnesses the power of quantum mechanics or qubits, in order to solve the most complex algebraic and combinatorial problems within a feasible amount of time, proves its significance and need in today's

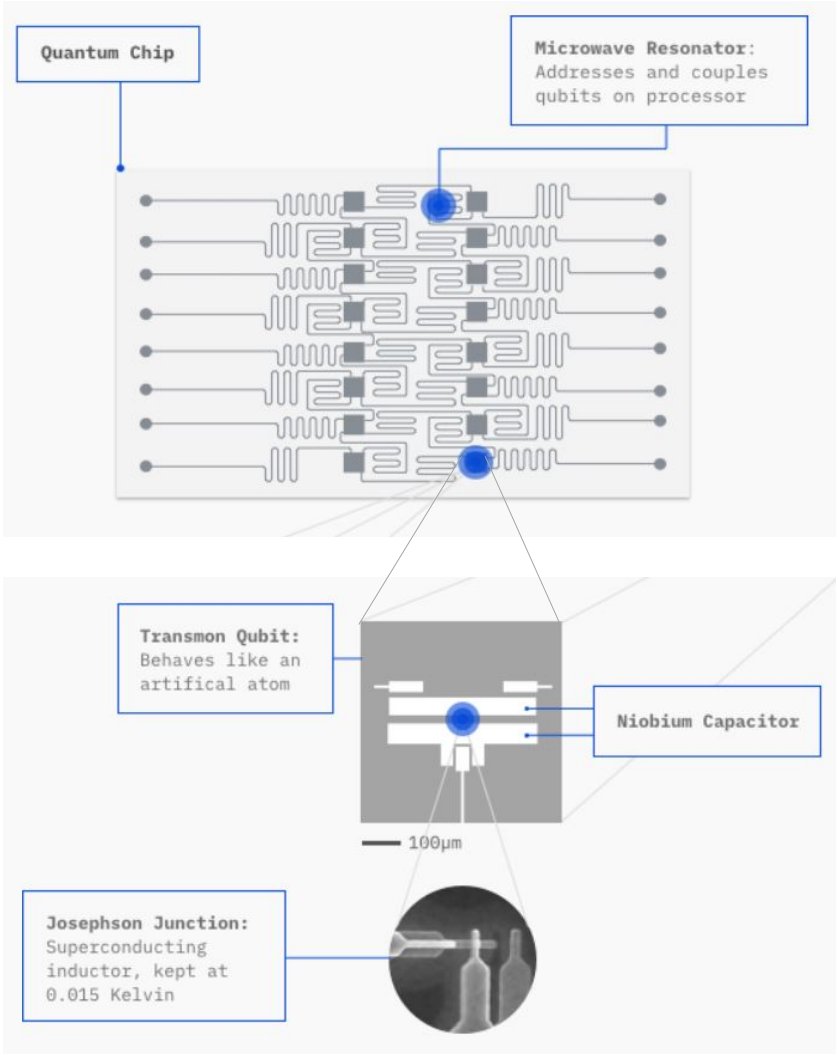

Figure 1.3. Fundamental components of hardware in a quantum computer [1].

world.

For example, a single cup of coffee contains about a billion or trillion molecules of caffeine. If a single caffeine molecule is considered, it has about 10^{48} bits worth of information. In comparison, the number of atoms in the Earth is estimated to be 10^{49}, or 10^{50}. Therefore, for one caffeine molecule, we would need a number of bits comparable to as much as 10 percent of the number of atoms in the entire planet [9]. From a classical computing perspective, it is impossible to store that much information

on such a computer. However, since qubits can have multiple states at the same time (*superposition*) or they can be correlated with each other(entangled), quantum computing is well equipped to handle such complex data in large volumes. Currently, a leading company like IBM has 50-65 qubit quantum hardware. Emergence of hardware with larger number of qubits will enable storage and processing of information at scale. A caffeine molecule's energy can be stored using about 160 qubits. From a scaling point of view, such information related to more complex molecular subjects and drugs in healthcare can be stored, studied and processed using 1000 or 10,000 qubits systems to provide a huge leap in research and technology.

Another excellent example to demonstrate the speed-up provided by quantum computing is Grover's algorithm that finds one item among N items [10]. Normally this would require a classical computer to search for at least $N/2$ items or in the worse case, all the N items. However, quantum computer would need to search just $\sqrt{N}$ items to find the one item. This an exponential speed-up for large N values. Therefore, compute-time intensive complex algorithms or problems can be processed easily in tractable time using quantum computer compared to a super computer.

Although exciting, there can be some drawbacks as well in using Noisy Intermediate Scale Quantum (NISQ) computers. For example, they can have high error rates and can have lower fidelity rates with respect to the output states. The qubits inside the quantum computer for example, IBM Chandelier need to be maintained at about 0.015K (for the Josephson junction shown in 1.3) to control noise.

Chapter 2

Proposed Threat Model and Countermeasure

This chapter describes the significance of hardware security and emphasizes on its need in today's world. Various threats involved in the design cycle of an Intellectual Property (IP) are discussed. Logic obfuscation method in hardware security is defined and explained. Threat model and its feasibility and assumed adversarial capabilities are discussed. Finally the idea proposed to tackle this threat model, is described in detail.

2.1 Need for Hardware Security

The advent of the Integrated Circuits (ICs) revolutionized the electronics industry and paved the way for smarter devices nowadays. Right from the smallest intelligent sensors to the high performance supercomputers, ICs are an integral part of today's technology. Hardware is the fundamental building block of any system and the root of trust for

an application. Therefore, ensuring hardware security becomes essential along with keeping software secure. Securing hardware guarantees safer communication, storage and processing of information, and in turn, the best use of technology.

There are several hardware security primitives that have been developed over the years now that are efficient in not only securing the critical IPs from attacks but also making sure there are no unwanted leakages, thus ensuring privacy too. In the age where high performance supercomputing is used to process critical financial data, or efficient quantum computers are used to solve complex unfathomable problems of the universe, hardware security remains a boon that safeguards crucial data used in cutting-edge technology.

2.2 Logic Obfuscation

Logic obfuscation is defined as the modification of the existing logic in the circuit to obscure the true functionality of the circuit from adversaries [11]. This method Obfuscation involves insertion of additional logic into the original design and ensures that the true functionality is revealed only when the correct set of secret keys are supplied to the design. So the obfuscated design is safe, and once it comes out of the manufacturing process, the IP vendor uses the secret key that is stored in the tamper-resistant memory (inaccessible by the attacker) on the same chip and deobfuscates the design. Obfuscation can hide the structural design or even the functional parts of the circuits at RTL, gate level or the layout level for different stages of the IP design cycle [11], [12].

2.2.1 Adversarial Attacks

An IC goes through several steps during design and manufacturing cycle. Due to the globalization of its design flow, these sensitive ICs can be sent through several organizations and entities across nations that contribute in the design cycle. During this process, any kind of rogue elements in the supply chain can attack the IP for profit. This can include IP piracy, overbuilding of ICs, to name a few [13] making designers and users reconsider their trust in hardware [14]. Some of these attacks by the rogue elements can include:

- IP piracy, counterfeiting, cloning, recycling: Many IPs during manufacturing are prone to piracy, counterfeiting and reselling as well as illegal cloning of the chips for misuse. Some of old, non-functional discarded chips can be repainted and remarked falsely and sold to the users. This way, many fake/counterfeit IPs can be circulated in the supply chain by the rogue vendors to siphon profit. Several copies of the same design implementation are made and sold off illegally to gain some extra profits.

- Tampering and Trojan insertion: ICs can be tampered during any step of the design cycle due to the lack of control on the process itself and outsourcing to the untrusted third-party entities. Trojans can be a functional or a parametric type (such as thinning of wires, weakening transistors, logic changing threshold voltages etc). This can lead to several unwanted consequences that may corrupt the functionality directly or indirectly or even damage the component permanently [15].

They can also be activated when desired making them more dangerous and difficult to catch in timely fashion.

- Reverse Engineering: If the adversary has enough computational resources, they can reverse engineer the design to understand its functionality and extract sensitive data.

The above hardware attacks can lead to huge loss in semiconductor industry [16], [13]. Thus, there is a need for securing the application critical IPs using hardware security techniques like logic obfuscation.

2.2.2 IP core obfuscation

This type of obfuscation involves a high-level transformation at the RTL design, logic/gate level or even layout level to convert the core into a more obscure architecture. This obfuscation can be of two major types: (a) Structural obfuscation-that involves changing the actual structure of the design that can involve RTL or gates, (b) Functional obfuscation-that involves modification of the function or output of the circuit. This can include use of AES and IP core locking blocks to prevent any type of malicious access to a particular IP core unless the the correct key bit sequence is given.

2.2.3 Combinational/sequential circuit obfuscation

As the name suggests, this type of obfuscation targets combinational/sequential circuits at the gate level. There can be two main types: (a) Passive obfuscation- This does

not affect the functionality of the circuit directly, but is more of a soft modification (example, syntactical) in the design, such that it is difficult for the adversary to read the functionality and understand it. (b) Active obfuscation- This actively alters the main structure/function of the circuit. It is mainly based on application of the correct key to open up hidden parts of the circuit. Embedding Finite State Machines (FSMs) can control the functional modes of the circuit and the correct states are activated only on the application of particular keys.

2.3 Proposed Threat Model

Quantum circuits can be lucrative targets of stealth for making profit if they are reused in multiple applications e.g., quantum Machine Learning (ML) circuits such as, classifiers. Problem-specific circuits optimized to solve the problem at scale can also be considered as IP. Furthermore, leaking of high-level information e.g., type of algorithm used in the circuit and the problem that is being solved can provide undue financial/political advantage to the adversary and compromise national security (depending on the problem being solved). Threat model used is as shown in Fig. 2.1, where the untrusted third-party compilers being used are the rogue body or the adversary.

2.3.1 Feasibility of threat model

One of the important aspects of quantum circuit compilation is to optimize the circuit for improved circuit depth and reduced gate count. Several 3^{rd} party compilers are

evolving that offer optimization at faster compilation time even for large quantum circuits [17, 18]. Following factors will motivate the quantum circuit designers to avail the possibly untrusted 3^{rd} party compilation services, (a) success of quantum circuit, since optimized circuit is essential to obtain meaningful results from NISQ computers. Poorly optimized circuit even though functionally identical to optimized circuit will produce inferior outputs; (b) possible lack of trusted compilers that have caught up with the latest advancements in optimization; (c) availability of efficient but untrusted 3^{rd} party compilers [17–19] that are being developed to optimize depth and gate count compared to trusted compilers. These compilers can be hosted on either the local machines by the 3^{rd} party or on the cloud service providers (e.g., AWS) to launch, (i) cloning/counterfeiting, where quantum circuit can be stolen or reproduced; and (ii) Reverse Engineering (RE), where the sensitive aspects of the quantum circuit could be extracted.

2.3.2 Adversarial Capabilities

We assume that adversary:

- Has access to the unoptimized quantum circuit (obfuscated or unobfuscated).
- Does not have access to an oracle model that can be used as a golden reference to cross-validate all the reverse engineering results to confirm the adversary's guess.
- May assume that the circuit is obfuscated. We also consider a strong adversary model by assuming that the adversary will be aware of a single dummy gate in the circuit (via collusion with the design house or leaked information) but will not

know its location. Hence, will have to have multiple reverse engineering trials in order to guess the dummy gate location correctly (there are some measures taken by designer to create confusion in the adversary).

- Has computational resources to analyze the circuit to identify and exploit clues e.g., barriers inserted in the circuit to identify the dummy gates. We assume that adversary has many computation resources in order to reverse engineer or deobfuscate the circuit, to make the attack model stronger.

2.4 Proposed Obfuscation Technique

Quantum computers can potentially be revolutionary since quantum algorithms can do specific computational jobs exponentially faster than their classical counterparts. Quantum circuits are a representation of quantum algorithms that can run on a quantum computer. However, a high-level quantum circuit needs to undergo a *compilation* process to translate any high-level gates in the circuit to machine-supported instructions and resolve any connectivity constraint. These steps increase the number of gates in the compiled circuit. Due to noise and short lifetime of qubits in NISQ computers, the extra gates degrade the circuit performance. Therefore, it is critical to have an efficient compiler to address the device restrictions with the fewest possible extra gates. In this regard, several third-party compilers [20] are emerging with better performance than established compilers like Qiskit [21] and QuilC [22]. As the quantum computing ecosystem evolves, with 1000-qubit devices projected in the next few years [23], we should expect more

third-party compilers, some of which can be untrustworthy.

Developing a novel quantum algorithm may require years of R&D effort and financial investment. Moreover, certain algorithms can contain sensitive information about the problem structure. For example, quantum approximate optimization algorithm (QAOA) embeds the topology of the problem, which is an asset. For applications like power grid (or other critical infrastructure) optimization, the client would like to keep the information confidential. Therefore, quantum algorithms and its circuit representations can be treated as intellectual properties (IPs). Submitting quantum circuits to untrusted, but high-performance, compilers can lead to piracy issues like IP theft and sensitive information leakage. Thus, it is paramount to develop compilation techniques to take advantage of efficient third-party compilers while preventing piracy issues.

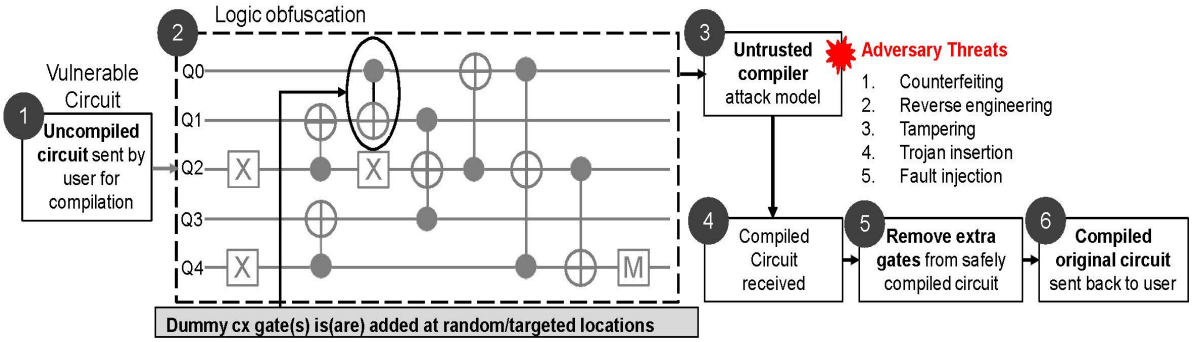

Figure 2.1. Attack model and solution proposed. The quantum circuit (Alu-OR in this example) is sent by the user to the untrusted compiler, where the adversary can steal the IP or RE the circuit. Logic obfuscation is proposed as countermeasure.

The proposed obfuscation technique inserts additional gate operations (e.g., CNOT or CX gates) referred as *dummy gates* which are not part of the original circuit. The circuit designer inserts these dummy gates in certain locations of the original circuit using our proposed metric-guided heuristic to overcome the inherent margin between correct and

incorrect functionality before sending it to the 3^{rd} party compiler. The objective is to hide the true functionality of the circuit from the untrusted compiler. The adversary needs to identify and remove the dummy gates from an obfuscated circuit to extract the original circuit. This is a computationally hard problem since any gate in the circuit can be a potential dummy gate. Furthermore, there is a lack of oracle model (as the same quantum chip implements different functionalities using microwave/laser pulses) to validate the adversarial guess. Therefore, adversarial effort to reverse engineer(RE) the obfuscated design will be extremely high. Any attempt to reuse the circuit without removing the dummy gates will result in corrupted or severely degraded performance. We choose CX as dummy gates since it is a part of the pre-compiled circuit which eliminates any potential clue for the adversary. Note, choice of CX gate is agnostic to the goal here even if, the quantum hardware on which the circuit is further executed has different set of target gates. Future studies can also consider other choices of dummy gates.

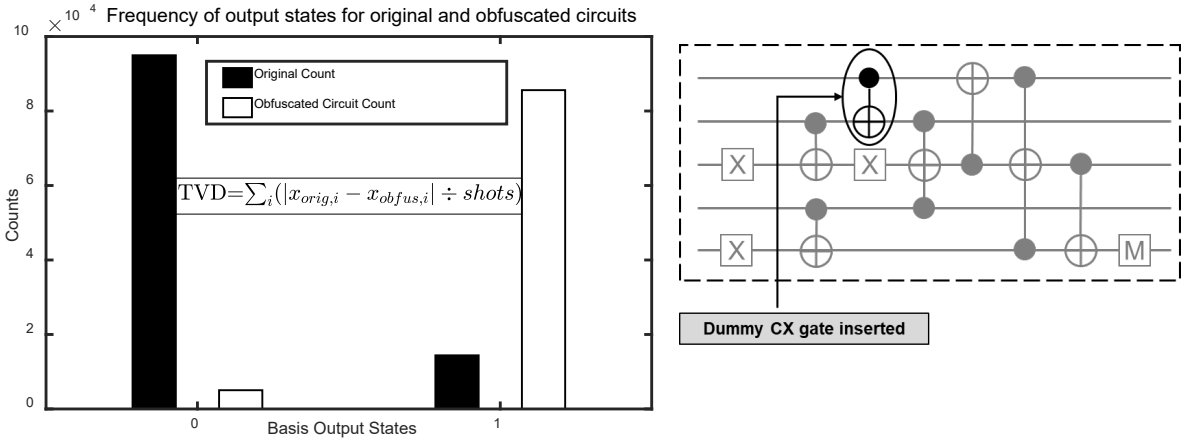

Figure 2.2. Frequency of output states for the original and obfuscated circuits. The correct output state changes after obfuscation.

To further illustrate the idea, we have executed the circuit shown in Fig. 2.1 on default

noise model of IBMQ Valencia with and without the dummy CX gate. The outputs are obtained for 4 possible inputs. For example, we observe outcome of Alu-OR circuit (Fig. 2.1) for one input over 100000 shots. The histogram in (Fig. 2.2) shows the measured output bits and observation frequency, respectively before and after inserting dummy gate at position 1. The obfuscated output distribution is quite different from the original circuit output distribution which indicates obfuscation of functionality. We quantify obfuscation quality by Total Variation Distance (TVD) [24] that measures degradation or the difference between obfuscated output and original output. TVD is defined as $\sum_i(|x_{orig,i} - x_{obfus,i}| \div shots)$. Here, x_i is the count of i^{th} element of a distribution. TVD for position 1 among the 28 possible dummy gate locations for Alu-OR circuit is 17.225%. Note, any other metric to calculate the statistical difference between distributions can also be used instead of TVD.

One of the challenges with above technique is that the designer cannot evaluate the effectiveness of the obfuscation using simulations since the correct output state (for a realistic optimization problem solved using the quantum circuit) is not known. We propose several characteristics to identify the right location where the dummy gate should be inserted to maximally corrupt the output. Such a metric is identified from the insights developed from an exhaustive analysis of quantum circuit benchmarks for various dummy gate insertion points evaluated using simulations.

2.5 Current Trends

Recently, there has been a lot of work that considers different type of attack models on quantum circuits. X.Cui et al. and N. Limaye et al. [25, 26], assume that adversary will insert Trojans in the reversible circuit before fabrication. However, this attack model is not applicable to quantum circuits since basis gates are realized using microwave/laser pulses. The quantum circuit is never physically fabricated in gate-based quantum computing even though a quantum circuit is reversible. Test pattern-based detection technique from this work does not apply to quantum circuits.

S. M. Saeed et al. [27] assume untrusted foundry that can locate ancillary and garbage lines in reversible circuits and can extract the circuit functionality. Dummy ancillary and garbage lines are added to the circuit which increases the ancillary and garbage lines post-synthesis. The attacker can identify only ancillary and garbage lines added post-synthesis, not pre-synthesis. This approach is only applicable for oracle-type or pure boolean logic-based quantum circuits and not for quantum computing. It critically depend on the presence of the ancillary and garbage lines and assumes the foundry and fabrication process of the reversible circuit is untrusted.

N. Acharya et al. [28] assume that malicious adversary in quantum cloud will report incorrect qubit quality to force erroneous computation. Adversarial model used by N. Acharya et al. [28] consider quantum cloud to be malicious instead of the compilers used. Test points are inserted in the circuit to detect any malicious change of calibration, as any change in calibration can affect the output of the quantum circuit, changing its

fidelity.

A. Saki et al. [29] assume third-party compilers as an adversarial threat and propose split compilation technique for defense. Uncompiled circuit is split into two or more splits and sent separately to the untrusted compiler for compilation. Even if the adversary tries to access the data, it gains only part of the entire circuit, which yields wrong outputs. This work also implements a robust swap routing layer that maps the physical qubits of the splits with each other, eliminating any coupling map problems that might arise due to the splitting.

The proposed work is more generally applicable for any quantum circuit. It does not depend on presence of any ancillary lines and assumes the compiler or certain version/release of the compiler is untrusted. This work addresses vulnerability of a quantum circuit at a compiler level in the quantum computing stack. A novel idea of logic obfuscation is introduced to ensure security of critical quantum circuits.

Chapter 3

Logic Obfuscation in Quantum Circuits

This chapter addresses obfuscation idea, procedure, and the experimental results. Reason for choice of the dummy gate inserted are discussed here. Experimental results are reported and analysed. Different features of a circuits are discussed, and the way they are used to form a heuristic metric to achieve targeted dummy gate insertion to get a higher TVD that means better obfuscation and in turn, a more secure circuit before compilation.

3.1 Cx/Cnot gate addition

This section explains the idea of using Cx gate as the dummy gate being inserted for obfuscation of the circuit.

3.1.1 Obfuscation Procedure

Several reversible benchmarks are taken from Revlib [30] and used for this experiment. Our goal is to introduce dummy gates in a quantum circuit strategically to obtain a good TVD before it is sent to the untrusted compiler. To achieve this, we exhaustively analyze all possible dummy gate insertion points in benchmark circuits, the corresponding TVD values, and identify their features. Once the features for a particular dummy gate location and corresponding TVD is understood, a guided heuristic is developed to insert dummy gates for an unseen circuit to maximize the functional corruption or obfuscation of the circuit.

For the exhaustive search based analysis, a quantum circuit is chosen (here circuits with only two and three qubits controlled CNOT gates and double controlled CNOT gates (CX, CCX) are considered), and two qubit CX gates are inserted at various locations without altering the circuit depth. To do this, the circuit is first divided into *slices*. Inside each slice, any two quantum gate can operate on different set of qubits, i.e., the quantum gates can operate in parallel. The impact of long distance CX gates (between non-neighboring qubits) also needs to be included in the exhaustive search-based analysis. Hence, assuming n free qubits in a slice, there are $\binom{n}{2}$ possible ways to insert the dummy 2-qubit CX into the slice if $n \geq 2$ (even on non-neighboring qubits). Since each qubit in CX gate can be either target or control, the number of possible dummy gates is $2 * \binom{n}{2}$.

To study appropriate positions for the dummy gates, we insert one dummy CX gate at each of the possible positions of the benchmark circuits from Revlib [30] repository

(Fig. 2.1) and analyze its impact on the output in terms of TVD to create a location prediction metric. These metrics are then studied to validate their effectiveness. **Average, Worst, and Best TVD:** The highest and the lowest TVD observed for each benchmark during exhaustive simulation are the best, and the worst TVDs, respectively. We term the average value of all the TVDs during the exhaustive analysis as average TVD.

3.2 Proposed Heuristic

Step – 1:

First step is to identify various features that can distinguish the dummy gate locations from each other in a given circuit in terms of TVD. Some examples of these features include number of control gates on the qubits and depth of the dummy CX gate from output (explained next). These features are used individually or in combination to determine the best metric for guided selection of CX gate location for new circuits. The features are explained below. (Fig. 3.1) illustrates the corresponding values for an example benchmark.

- **Depth from primary output feature:** We use this feature to calculate the number of layers present in between the considered position and the output. It quantifies the influence of the CX gate on the output of the circuit. For example, the depths of CX positions 1, 2, and 4 are 7, 7, and 6 respectively (Fig. 3.1).

- **Measured qubit feature:** This feature checks if either of the qubits associated

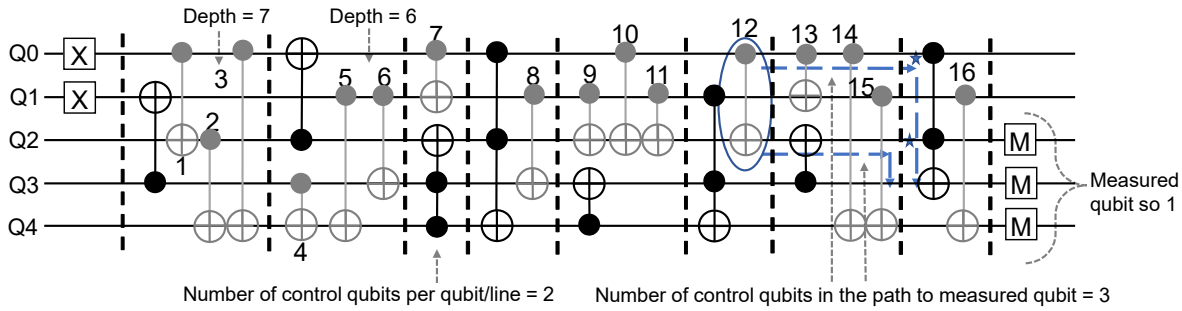

Figure 3.1. Circuit diagram of 123 counter with annotated features. Dark gates belong to original circuit; gray gates are candidate dummy locations. Vertical dashed lines indicate layers. Q0 and Q1 has constant 1 as input. For example, CX gate position 12 has depth = 2 (number of dashed regions or layers to measured qubits), number of control qubits = 5 (dark circles in the qubit lines Q0 and Q2), measured or not = 1 (since the target qubit is on a measured qubit line), constant qubits involved or not = 1 (since Q0 is a constant input), and number of control qubits in output paths = 3, as shown. The paths shown with dashed blue lines explains the latter; they start from each input qubit of CX gate 12, and continue through control qubits of following CX gates, and end in a target qubit that is measured. Doing this for each qubit of CX gate 12, we get 3 control qubits in these paths.

with a dummy CX gate gets measured eventually or not. If target qubit is measured, then it is likely that the CX gate will impact the output significantly e.g., CX–1 in Fig. 3.1 involves 1 measured qubit (i.e., Q2), and CX–7 involves none.

- **Control qubits in qubit line feature:** This feature counts the number of times a qubit involved in the dummy CX gate is used as control qubit in the circuit. This intuitively states that if this qubit is impacted, the other directly/indirectly targeted qubits will be affected as well.

- **Control qubits in output paths feature:** This feature counts the number of control qubits that can be traced in the output paths from each qubit involved in the dummy gate (source) until one of the measured output qubits is reached (destination). Multiple paths can be traced from the dummy gate to one of the measured qubits. The number of these control gates are added together for each

dummy gate with double-counting allowed. Consider CX–12 (Fig. 3.1) where the paths stemming from target qubit in the CX gate are shown as blue dashed lines. Thus, for the purposes of this metric, CX–12 has 3 control qubits including one in the top path for qubit Q0 and the Q2 qubit counted once for each path.

- **Constant input qubit feature** Some circuits have constant values as inputs for some qubits e.g., 1 or 0. These constants affect the output especially, (i) for some input combinations, or (ii) when the dummy gate involves a constant input qubit. They get applied through a CX gate, and as a result affect the other gates, eventually corrupting the output. Hence interaction of a constant input with the target qubit of a dummy gate or any other gate on that qubit line is used as a feature.

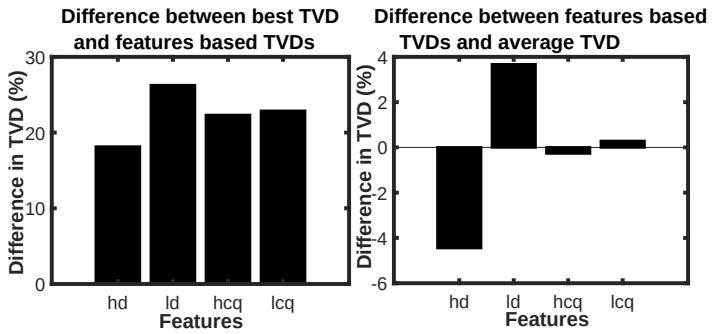

Figure 3.2. Difference (%) between feature-based TVD and, (a) the best TVD, and (b) the average TVD. hd: highest depth, ld: lowest depth, hcq: highest control qubits and lcq: lowest control qubits.

Step – 2:

We use the features described above to develop an effective metric. Initially, the dummy gate depth (or number of control qubit) feature is used individually to select the dummy gate location. However, these simple metrics produce multiple positions with the same feature values. The TVD of these positions are averaged and compared with the average and the best TVD of each of the benchmark circuit. Fig. 3.2 shows the average for 2 features namely, depth and number of control qubits. It can be noted that each of these features exhibit strong correlation towards output TVD. Furthermore, some features might maximize the TVD more than others. Since these features produce more than one positions, we use their combination to distill a single best CX location for each circuit. Fig. 3.2 shows, if we consider a single feature, the difference from best TVD is relatively high ($\approx 18\% - 26\%$, Fig. 3.2a) and it lags behind the average TVD in most cases (Fig. 3.2b). Therefore, we consider multiple features together to refine our choice(Fig. 3.4). This approach eliminates some positions and safely reduces the search space for each of the benchmarks. We aim to develop a metric that provides at least 5% better TVD than average TVD and not worse than 15% from the best TVD for most of the circuits. Usage of metric-based prediction(where metric is a combination of the features), proves to give at least 5% improvement in difference between best TVD and the metric-based-TVD, and improvement by approximately 8% in difference between the metric-based-TVD and the average TVD, when compared to the features-only based dummy gate position prediction.

- **Metrics x1, x2 and x3:** As an initial approach, in the first pass, we remove CX positions with control qubits in output paths to the measured qubits since these positions do not in any way affect the final output and hence, the TVD. This constitutes metric x1 where the remaining positions' TVDs are averaged and considered as result for metric x1. For metric x2, the positions without any measured qubits are further eliminated since they cannot directly affect the TVD. Finally, the top three score (= depth + number of control qubits in the output path) positions are taken and the corresponding TVDs are averaged to form metric x3. Removing such positions from consideration reduces the number of CX locations.

- **Metric-1:** Here, after applying x1, x2 and x3, dummy gate positions with same score but zero constant qubits are eliminated since they have less impact on TVD.

- **Metric-2:** Here, we apply metrics x1, x2, x3, and metric-1. Next, the positions with lowest two/three depth values, at least one measured qubit, and the highest number of control qubits are chosen. The lowest depth selects the positions closest to the measured qubits and if the corresponding qubits are being measured with high number of control operation, the impact on TVD will be high.

3.3 Choice of gate addition

Choice of dummy gate added is dependent on several factors, ranging from the type of circuit, function that it implements, type of gates used in the circuit etc. This section accounts for this choice made in the work.

As an initial thought, a two-qubit Swap was used, which is essentially exchange or swapping the value stored by one qubit with another. The idea was to create sufficient obfuscation or corruption measured by the TVD. Although this can be good for corrupting the output for a circuit, there are some limitations to using it. Swap gates are easily recognizable by the adversary when used in any type of circuits, especially as a part of an arithmetic circuit. This means that during compilation, if the adversary is aware that there is a dummy gate presence in the circuit, they can easily get rid of this dummy gate if it is a swap, because a swap gate is obvious as it stands out. Usually swap gates are added during the routing step in compilation process to resolve any connectivity issues between the required qubits for circuit functionality. Another scenario could be that the added dummy swap gate can/will eventually get optimized and together with other gates becomes difficult for the designer himself to deobfuscate the design. This also means, it cannot be assured that the compiled circuit can be deobfuscated correctly and precisely. Adding of extra swap gate/gates can also affect the optimization of the circuit. This problem could be further aggravated if barriers are not used.

A dummy gate should be chosen such that it provides enough security(through enough corruption of output or TVD) but also ensures confusion for the adversary and makes the deobfuscation easier for the designer. More often than not, arithmetic circuits have Cx/Cnot gate, with one qubit as the control qubit and another qubit as the target qubit, is used in the circuit. Adding Cx gate as the dummy gate for obfuscation thus ensures a sure way to create more confusion for the adversary. Along with the added dummy gate, there are also other Cx gates that were already a part of the existing original

circuit. This means that the adversary is now confused as to which could be the added dummy gate, even if they are aware that it must be a Cx gate, among other possible gates.The Cx gate not only creates confusion (since there are already two-qubit Cx gates and three-qubit CCX gates, in such arithmetic circuits under consideration), but also eases deobfuscation procedure by designer. Therefore, with use of barriers to prevent any unwanted optimizations and/or gate replacements or cancellation during the compilation process, a dummy gate Cx gate is a better choice.

3.4 Case Study- 123 Counter circuit

The 123 counter circuit (Fig. 3.1) is divided into 8 time slices between dashed barriers. For each experiment, one CX gate at a time is inserted into one of the 16 possible locations without overlapping with the other gates in the slices. After compilation and simulation for individual CX gate positions, the worst, best, and average TVD are noted to be 5.84%, 51.16%, and 24.54%, respectively.

As shown in Fig. 3.2, the best performing feature based dummy gate selection is the number of control qubits in qubit line feature which is only 18.14% worse than the best TVD and 4.44% better than the average TVD. However, this feature cannot be used in isolation since it provides multiple positions with same number of control qubits. To solve this issue various combinations of the features like depth, number of control qubits, constant qubits etc. are used to select from the positions that can potentially provide best performance.

Our x1 metric employs only *non-zero control qubits in output path to measured qubits* and performed 15.21% worse than best TVD and 7.38% worse than the average TVD. Clearly, this simple metric did not outperform average TVD. Therefore, we combine few features and next prune the list of CX gate positions. Finally, we use a score (score = depth + number of control operations in the qubits) to downselect the best position from the list. In this example, metric 1 outperformed average TVD by 8.24% and underperformed only 18.38% than the best TVD. Metric 2 overperforms and underperforms by 9.08% and 17.54% than the average and best TVD, respectively. Hence, metric 2 is a better choice for this example. The final metric should be generic i.e., it should outperform the average TVD for most of the circuits and should be as close to best TVD as possible. Our analysis indicates that metric 2 meets this requirement.

3.5 Experimental result and analysis

3.5.1 Evaluation Framework and Benchmarks

We use IBM Qiskit [5] (with realistic noise values from ibmq_valencia [5]) executed locally on an Intel Core i5-9300H CPU clocked at 2.40GHz for simulations. The default compiler backend in Qiskit is used for compilation. A Python-based wrapper is built around Qiskit to accommodate the proposed obfuscation. We use 10 reversible circuits from RevLib [30] which are commonly used in quantum circuit compilation research [2, 19, 31].

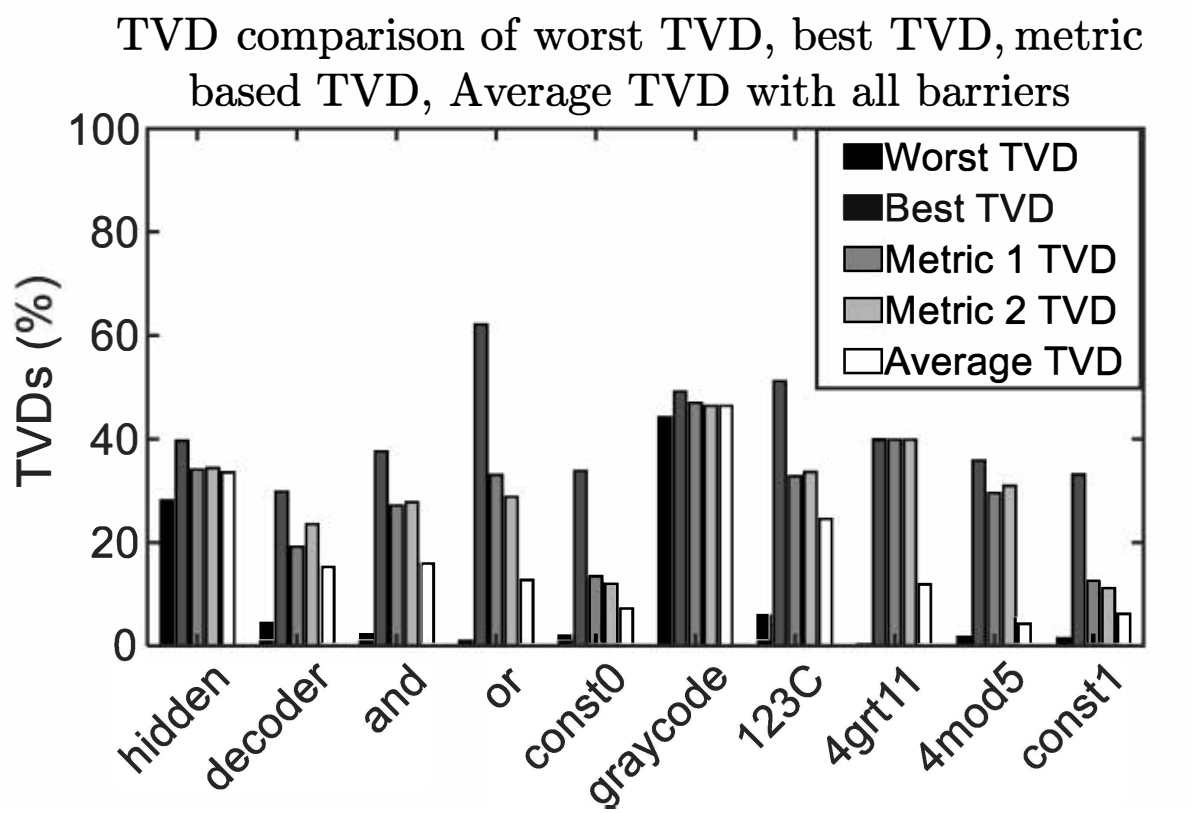

Figure 3.3. Plot for the worst, best, average and metric-based TVDs.

3.5.2 Comparative Analysis

Fig. 3.3 shows the worst, best, and average TVDs of the benchmarks alongside the TVDs observed with the TVD values obtained using Metric 1 and Metric 2. Metric 1 and 2 show 10.16% and 10.13% higher TVD respectively, than the average TVD over all the benchmarks. They underperformed by 12.42% and 12.45% on average, respectively from the best TVDs.

We show the average difference between the best TVD and the metric-based TVDs across all 10 benchmarks in Fig. 3.4(a). Note that, a higher difference indicates poor obfuscation. Metrics x1, x2 and x3 performed worse than the metrics 1 and 2 with difference of 15.21%, 13.54% and 15.01%, respectively from the best TVD. In Fig. 3.4(b), we show the mean difference between the average and the metric-based TVDs. Note that, a higher difference value indicates a better obfuscation in this case. Again, metrics x1, x2 and x3 performed worse than the metrics 1 and 2 with average difference of 7.38%, 9.04% and 6.07% each. The results indicate that metric 1 and 2 are better choices for strong obfuscation.

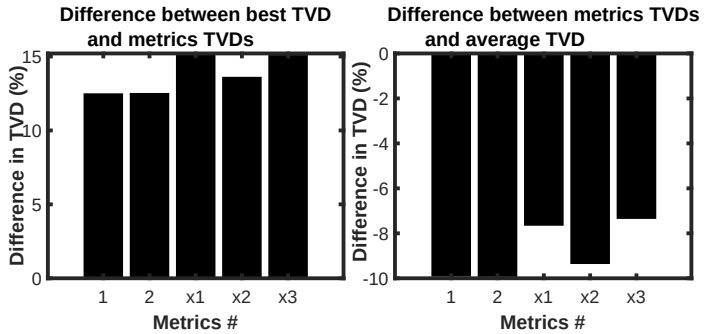

Figure 3.4. Difference (%) between metrics-based TVDs and (a) the best TVD, and (b) the average TVD.

3.5.3 Usage of noisy quantum simulator

Real quantum hardware inherently provides probabilistic outputs due to various known/unknown errors like crosstalk. This presents a challenge for analysis and derivation of clear intuition for the development of a metric for guided CX insertion e.g., crosstalk can overshadow the benefits of a certain metric. Therefore, using noisy quantum simulator with well-characterized noise models e.g., gate error and decoherence for our analysis.

3.6 Discussion

Deobfuscation of the circuit post safe compilation is a necessary step to preserve the authenticity of the circuit. Addition of barriers to the circuit gives a security versus optimization trade off. These important aspects are explained below.

3.6.1 Effect of barrier addition

A variable number of barriers are added around the existing and dummy cx gates. A minute TVD change of 2.9% for metric 1, and 3.11% metric 2 is observed on average. Therefore, insertion of barriers has minor impact on obfuscation quality.

The obfuscated circuit goes though significant optimization during compilation making it challenging for the designer to identify and remove the dummy gate post-compilation. Barriers can be introduced before and after the dummy CX gate to prevent optimization across the barriers and make the added dummy gate recognizable post-compilation. However, this may leave a clue for the adversary i.e., they may guess that the dummy

gate may be confined within the barriers. This is true if the adversary is aware that the circuit has been obfuscated.

To resolve this issue, more barriers can be introduced around the other CX gates making it hard for the adversary to locate the dummy gate. Therefore, even the CX gates that are part of the circuit are also surrounded by barriers. This will degrade the optimization quality since compiler cannot optimize gates across barriers anymore. This situation can present a security vs optimization trade-off.

Our results show that barrier inserted circuits experience on average $\approx 10\%$, $\approx 9\%$ and $\approx 17\%$ increase in depth, gate counts and transpilation time respectively compared to circuits without any barriers. Thus, the trade-off is modest.

3.6.2 Deobfuscation Process by designer

The designer can locate the dummy gate since it has been placed within a certain barrier in the netlist. One challenge to this approach is insertion of Swap gates by the compiler within the barriers to meet the hardware coupling constraints. Nevertheless, the designer exactly knows the qubits between which the dummy gate was inserted pre-compilation and the target coupling map. Therefore, the designer can easily RE the dummy gate within the barrier. The same task will be challenging for the adversary who does not know the type of dummy gate, its location within the barriers and the qubits. These advantages the designer gets over the adversary are explained as follows:

- Type and location of the dummy gate used- Designer knows dummy gate type and number of qubits it operates on because it is predetermined by him based on

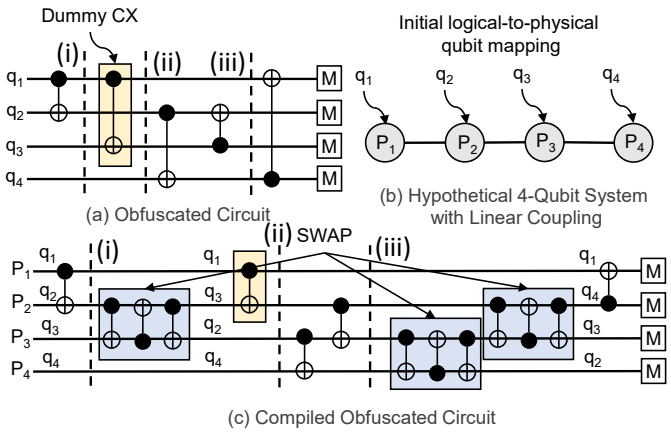

Figure 3.5. Recognition of dummy cx gate post compilation by the designer. (a) Obfuscated circuit. The designer records that a dummy gate is added between barriers "i" and "ii" pre-compilation, (b) logical-to-physical mapping of the example obfuscated circuit on the hypothetical 4-qubit hardware with linear connectivity, (c) compiled obfuscated circuit. The designer can track barriers "i" and "ii" and remove the dummy gate between them.

the type of circuits under consideration. The designer knows exactly where the target dummy gate is located is known. This is because the designer uses only the targeted insertion- extracts specific features of the circuit and follows the developed metric described above. The designer can recognise the two barriers that contain the gate. Therefore, he can easily remove both the dummy gate as well as any extra gates that could be added during compilation of the obfuscated circuit at the dummy gate location.

- Is aware of original circuit- The designer is aware of the circuit that was initially sent by the user pre-compilation, which is the original unobfuscated circuit.

To further illustrate the above, consider a scenario (Fig. 3.5)where a designer uses 3 barriers in the circuit to create 4 circuit partitions. One of those 4 partitions (say, partition 4) contains the dummy CX gate. The barriers are preserved during compilation (post-compilation circuit also contains the barriers), and the partition containing the

dummy CX gate does not have any other functional gates. Note that, the compiler may add Swap operations inside partition 4 to meet the coupling constraints of the dummy CX gate. However, these Swap gates are easily recognizable. For example, in IBM machines, these Swap gates are implemented with 3 consecutive CX gates. The designer can isolate these Swap gates from the dummy CX by matching this pattern. Next, the circuit can be deobfuscated by removing the dummy CX gate.

Chapter 4

Adversarial Effort Analysis

Adversary can be any rogue entity, that tries to get hold of the sensitive information carried in the IP and/or steal or tamper with it. As mentioned in 2 chapter, there are many evident threats such as counterfeiting, tampering, stealing, reverse engineering, cloning, side channel attacks, trojan insertion and fault injection. In this work, we assume the adversary as any rogue third party compilers used for more efficient compilation. This chapter also analyses the amount of adversarial effort needed to overcome the proposed obfucsation.

4.1 Reverse Engineering

To reverse engineer the circuit, the adversary will need to know the following:

- Knowledge about circuit being obfuscated and number of dummy gates used.
- The nature and type of dummy gate e.g., CX or SWAP or rotation.

- The location of the dummy gate.

Following scenarios summarize the adversarial reverse engineering of the obfuscation circuit.

- **Scenario 1**- If the adversary is aware of the type of dummy gate but not the location: This means that the rogue party has access to partial information and now is ready to put effort to locate the dummy gate added. For a circuit with "n" cx gates (including the dummy gate), adversary can try to remove one cx gate at a time. This will yield n copies of the circuit each of which could be a potentially correct, i.e, the original unobfuscated/pure circuit.

  The deobfuscation problem can become even more complex if multiple dummy gates are inserted by the designer since adversary will get ${}^{n}C_{2}$ copies of potentially correct circuit (for two dummy gates for example).

- **Scenario 2**- If the adversary is aware of the dummy gate location but not the type of the gate: There could be a case where the adversary knows the barriers between which the dummy gate is inserted in the circuit. Though the location of the dummy gate is known, there can be several CX gates or Swap gates or other basis gates added which can increase the effort of adversary to understand the type of gates present at that location. Adversary has to assume each gate as dummy gate giving rise to many possible circuit options one of which can be correct.

Each of the gates are implemented using specific microwave pulses, that cannot be accessed. Therefore, the lack of an oracle model or a golden design of the circuit/IP (that

Difference in TVD(%) between OG and:	Average TVD(%)	Std Dev TVD(%)
Circuit1	**6.323**	**1.182**
Circuit2	27.787	12.920
Circuit3	24.228	10.195
Circuit4	25.299	12.441

Table 4.1. Difference in TVD between the $n!$ pairs of circuit combinations that are under guess by the adversary, for a 123 counter circuit.

can be used as a reference) increases the adversarial effort. There is no way to validate the assumed pure/ obfuscated circuit assumption. Without validation, the adversary does not get a definite clue about our obfuscated circuit/IP, thus making the circuit secure.

4.2 Experimental result analysis

An experiment to analyse the adversarial effort has been implemented on a noisy model-IBMQ FakeValencia backend from IBM Quantum. Initially, the circuits chosen are obfuscated with the dummy gate chosen according to the developed metric 3. First step is to create "n" copies of the circuit, with each of the "n" CX gates removed, at a time. These circuits were then run to get the corrupted or pure output states (one of them will be the pure/unobfuscated circuit). In the second step, the TVDs of each of these circuits are compared, and the difference between these selected TVDs are taken between each "pair" of circuits. This means that there will $n!$ such differences that are listed below. As seen in Table 4.1, the average TVD difference between the original unobfuscated circuit and the circuits 1, 2 and 3 are around $24\% - 28\%$ which means that they do not provide any specific information. If there is a drastic change in the difference in TVD between any two such circuit pairs, that gives some clue to the adversary, that might act as a signature. Referring back to Table 4.1, this difference in TVD of just $\approx 6\%$ is as average and just about 2% standard deviation gives a distinguishing feature by which other circuits can be eliminated, reducing the search space to just about two circuits one of which is the original circuit.

Chapter 5

Future Work

This chapter presents the ideas that can be explored further in quantum logic obfuscation.

5.1 Larger sized circuits

Obfuscation of larger circuits i.e., with more number of qubits, or gates can be explored for scalability analysis. On one hand, the proposed idea can be easily applied to any size of the circuit. However, various factors can affect the obfuscation as the circuit size increases, such as noise in the quantum gates that degrades performance. Therefore, a better optimization with a more efficient compiler is needed (that can in turn have stronger adversarial capabilities as well). Due to higher error rates due to larger size, lower fidelity rates, the TVD measure can become a little unreliable giving rise to the need for a better quantitative measure. This can be translated into a a more detailed work, with several results to back up the said assumptions.

5.2 Multiple dummy gates

Instead of just one dummy gate as used in the work, multiple dummy gate addition can be studied. This can be done to understand the impact and the amount of obscurity can be attained for the circuit through obfuscation. If more dummy gates are added, amount of impact on the deobfuscation process, and the optimization versus security trade off can be analysed.

5.3 Dummy gate used

The size of the dummy gate used can be more than a two-qubit gate. It can also be a different type of gate beyond CX gates. This kind of work would provide a more comprehensive approach to the idea in order to understand its effectiveness. Different types of circuits like non-arithmetic circuits (like circuits that implement the QAOA or VQE algorithms), parametric/non-parametric circuits can also be tested for this idea.

5.4 Quantitative measure

TVD is the statistical distance is being used as a measure to understand the amount of corruption of output states that occurred when dummy gate was added. It quantified the obfuscation process that helped us understand the working of the idea. In future work, researchers are encouraged to explore other different measures that can further quantify this method, with even more precision for more balanced output distributions in circuits.

5.5 Other security methods

Different other hardware security techniques can be explored (example, split compilation [29]) as part of future work. Adversarial effort can be studied in detail, optimization versus extra barrier addition for security and amount of overhead on the TVD and functionality of the circuit when extra gates and barriers are added, can be analysed.

Chapter 6

Conclusion

In future, quantum circuits can implement complex algorithms with multiple high-level gates. Such circuits need to be compiled to translate high-level gates into native gates and resolve any connectivity constraints. This compilation might add extra gates that can further degrade the performance due to noise, consequently reducing the optimization chances. Therefore, designers will rely on untrusted yet efficient 3^{rd} party compilers for optimization of large scale quantum circuits. This can expose the ICs to counterfeiting, reverse engineering and other threats. We presented quantum circuit obfuscation by inserting two-qubit CX gates and measuring effectiveness using TVD as a metric. We finally presented an automated procedure to select the best position for dummy CX gate insertion to maximize the TVD. The proposed technique achieves $6-10\%$ higher TVD than average and reaches within $12-15\%$ of the best known TVD. Deobfuscation and impact of barrier addition were also discussed in detail. Some future ideas that can be used are also introduced.

Bibliography

[1] URL https://www.ibm.com/quantum-computing/quantum-computing-at-ibm/

[2] Tannu, S. S. and M. K. Qureshi. (2018) "A Case for Variability-Aware Policies for NISQ-Era Quantum Computers." *arXiv preprint arXiv:1805.10224*.

[3] URL https://en.wikipedia.org/wiki/List_of_quantum_processors

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

[5] Cross, A. (2018) "The IBM Q experience and QISKit open-source quantum computing software," in *APS Meeting Abstracts*.

[6] Nam, Y., N. J. Ross, Y. Su, A. M. Childs, and D. Maslov (2018) "Automated optimization of large quantum circuits with continuous parameters," *npj Quantum Information*, **4**(1), pp. 1–12.

[7] URL https://www.ibm.com/quantum-computing/services/

[8] URL https://www.linkedin.com/pulse/how-much-data-created-every-day-2020-kesha-shah/.

[9] URL https://www.compassdatacenters.com/compass-u/podcasts/robert-sutor/.

[10] URL https://qiskit.org/textbook/ch-algorithms/grover.html

[11] Rajendran, J., Y. Pino, O. Sinanoglu, and R. Karri (2012) "Security analysis of logic obfuscation," in *DAC Design Automation Conference 2012*, pp. 83–89.

[12] Zhang, J. (2016) "A Practical Logic Obfuscation Technique for Hardware Security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **24**(3), pp. 1193–1197.

[13] Bhunia, S. and M. Tehranipoor (2018) *Hardware Security: A Hands-on Learning Approach*, Elsevier Science.
URL https://books.google.com/books?id=wIp1DwAAQBAJ

[14] URL https://dsb.cto.mil/reports/2000s/ADA435563.pdf

[15] Karri, R., J. Rajendran, K. Rosenfeld, and M. Tehranipoor (2010) "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *Computer*, **43**(10), pp. 39–46.

[16] URL https://www.edn.com/semi_equipment-industry-stands-to-lose-up-to-4b-annually-due-to-ip-infringement/

[17] URL https://github.com/iic-jku/ibm$_$qx$_$mapping

[18] URL https://github.com/debjyoti0891/quantum-chain

[19] Das, P., S. S. Tannu, P. J. Nair, and M. Qureshi (2019) "A Case for Multi-Programming Quantum Computers." in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE/ACM, pp. 291–303.

[20] URL https://cqcl.github.io/pytket/build/html/index.html

[21] IBM, "Open Source Quantum Development-Qiskit," .
URL https://github.com/Qiskit

[22] Smith, R. S., E. C. Peterson, M. G. Skilbeck, and E. J. Davis (2020), "An Open-Source, Industrial-Strength Optimizing Compiler for Quantum Programs," .

[23] Martonosi, M. and M. Roetteler (2019) "Next Steps in Quantum Computing: Computer Science's Role," *arXiv preprint arXiv:1903.10541*.

[24] Sarovar, M., T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout (2020) "Detecting crosstalk errors in quantum information processors," *Quantum*, **4**, p. 321.
URL https://doi.org/10.22331/q-2020-09-11-321

[25] Cui, X., S. M. Saeed, A. Zulehner, R. Wille, K. Wu, R. Drechsler, and R. Karri. (2018) "On the difficulty of inserting trojans in reversible computing architectures," *IEEE Transactions on Emerging Topics in Computing*.

[26] Limaye, N., M. Yasin, and O. Sinanoglu (2019) "Revisiting logic locking for reversible computing," *IEEE European Test Symposium (ETS)*.

[27] Saeed, S. M., A. Zulehner, R. Wille, R. Drechsler, and R. Karri (2019) "Reversible Circuits: IC/IP Piracy Attacks and Countermeasures," *IEEE Transactions on Very Large Scale Integration (VLSI)*.

[28] Acharya, N. and S. M. Saeed (2020) "A lightweight approach to detect malicious/unexpected changes in the error rates of NISQ computers," *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*.

[29] Saki, A. A., A. Suresh, R. O. Topaloglu, and S. Ghosh (2021) "Split Compilation for Security of Quantum Circuits," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–7.

[30] Wille, R., D. Grosse, L. Teuber, G. W. Dueck, and R. Drechsler (2008) "RevLib: An online resource for reversible functions and reversible circuits," in *38th International Symposium on Multiple Valued Logic (ismvl 2008)*, IEEE, pp. 220–225.

[31] Ash-Saki, A., M. Alam, and S. Ghosh (2019) "QURE: Qubit Re-allocation in Noisy Intermediate-Scale Quantum Computers," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ACM, p. 141.