DECQA

Dictionary-based Energy-efficient Coding of Quantum Instruction Set guided by Algorithmic Information

Sibasish Mishra

Delft University of Technology

TU Delft

DECQA

Dictionary-based Energy-efficient Coding of Quantum Instruction Set guided by Algorithmic Information

by

Sibasish Mishra

in partial fulfillment of the requirements for the degree of

Master of Science
in Applied Physics

at the Delft University of Technology,
to be defended publicly on Thursday June 20, 2024 at 9:00 AM.

Student number:	5714826	
Project duration:	September 18, 2023 – May 06, 2024	
Thesis committee:	Dr. rer. nat. Sebastian Feld,	QuTech, Q&CE, TU Delft (Supervisor)
	Dr. F. Sebastiano,	QuTech, Q&CE, TU Delft
	Dr. M. Wimmer,	QuTech, TU Delft
Daily Supervisor:	Dr. ir. Aritra Sarkar,	QuTech, Q&CE, TU Delft

Cover: Branching of H-fractal mirroring the spanning of SU(2) space by gate sequences in the Solovay-Kitaev basis

Quantum & Computer Engineering

QuTech

TU Delft

Abstract

Efficiency in handling instructions within compilation and control processes is essential for scalability and fault-tolerant quantum computation. To mitigate the limited bandwidth for transmission of instructions and energy bottlenecks in cryogenic control architectures, this thesis aims to develop a compressed representation of quantum circuits. To achieve this goal, we study the concepts of algorithmic information theory and resource theory of computation. We focus on description complexity and establish compression as a useful estimate of algorithmic description complexity. With this motivation, we develop a generalized framework for the synthesis of quantum unitaries into a set of native gates and present a Huffman-encoded representation of the instruction stream that has a short code dictionary and offers a 60% compression over binary encoded representations. The developed framework offers 2 major contributions: an energy-efficient encoded representation of the quantum instruction stream and an estimate of the description complexity for quantum circuits. It qualifies as a successful algorithmic approach towards optimizing the QISA and aids the discovery of high-level quantum programming constructs.

Sibasish Mishra
Delft, February 2024

Acknowledgements

First and foremost, I extend my deepest gratitude to Dr. Sebastian Feld and Dr. Aritra Sarkar for their unwavering support, expert guidance, and boundless encouragement throughout this research journey. Your mentorship has been invaluable, shaping both my academic and personal growth. To the members of the QML group, thank you for the enriching discussions, cherished memories, and, of course, the delightful cakes that made our meetings truly enjoyable. Aritra, I am immensely grateful for your constant nurturing, collaborative spirit, and availability for our numerous brainstorming sessions. Your wise counsel and dedication have played a pivotal role in shaping this thesis, and I am truly thankful for the knowledge and experience gained from our collaboration.

I extend my sincere appreciation to Dr. Fabio Sebastiano and Dr. Michael Wimmer for graciously agreeing to serve on my committee and providing valuable insights and feedback.

To my dear friends Unik, Zulfikar, Kartikeya, and Sourav, our monthly hour-long diverse conversations have been a great source of laughter and joy. I am thankful to Sunruta for the memorable trips, outings, and shared experiences that have enriched my life during this time. Kishore, Praveen, and Héctor, thank you for being my steadfast study companions and for the countless adventures we embarked upon. Your friendship has added vibrancy to my academic journey.

Lastly, to my family members, your belief in me, steadfast encouragement, and unwavering support have been my rock through the highs and lows of this journey. I owe my success to your love and guidance.

Sibasish Mishra
Delft, The Netherlands
April 2024

List of Acronyms

AP	Solomonoff's algorithmic probability
ASIC	application specific integrated circuit
BDM	block decomposition method
BWT	Burrows-Wheeler transform
CSD	cosine-sine decomposition
FPGA	field programmable gate array
GPU	graphics processing unit
MSB	most significant bit
MTF	move-to-front transform
QAOA	quantum approximate optimization algorithm
QASM	quantum assembly language
QISA	quantum instruction Set Architecture
QML	quantum machine learning
QSD	quantum Shannon decomposition
QTM	quantum Turing machine
SKD	Solovay-Kitaev decomposition
TPU	tensor processing unit
UTM	universal Turing machine
VQE	variational quantum eigensolver

Contents

1

Introduction

A classical computation is like a solo voice—one line of pure tones succeeding each other. A quantum computation is like a symphony—many lines of tones interfering with one another.
- Seth Lloyd, MIT

Before introducing the project, we begin by setting up the premise of motivating quantum computing from an investor's and a theorist's point of view. Developing this dual motivation is essential as it also resonates with the inspiration and development of the project and its contributions.

1.1. Investor's Motivation for Quantum Computing

Starting with a brief overview of the history of computation, we follow the tide of new innovations and technology to further the scope of computation. We confront the challenges and roadblocks in the way of advancing hardware for improving computational power. We explore the possibilities of quantum computing, its advantages, and also the roadblocks and challenges on the way to achieving large-scale fault-tolerant quantum computing.

1.1.1. Evolution of Computation: A Concise Historical Overview

The history of computation is one of the most fascinating stories of human progress, marked by distinct eras and breakthroughs. Originating in the pre-mechanical era, dating back to as early as 1500 BC, manual tools such as the counting board and abacus defined this period. Then came the mechanical era of computers, which started with the development of mechanical clocks that were capable of performing astronomical calculations. The most notable innovations of this era are Blaise Pascal's arithmetic machine (1642), Charles Babbage's Difference engine (1822), and the analytical engine (1837). The invention of the vacuum tubes in the mid-20th century and the subsequent development of transistors replaced the mechanical parts with the dawn of the electronic age, resulting in the development of the first computers. The microelectronics era introduced the integrated circuit, allowing us to place several miniature transistors on silicon chips. The development of integrated circuits led to an era of rapid growth and innovation, encapsulated by Moore's Law, which predicted the doubling of transistors on a chip approximately every two years. This exponential growth spurred unprecedented advances in every sector of society, from science and medicine to business and entertainment. The genesis of theoretical computer science coincided with the evolution of computer hardware technologies, both of the domains influencing and shaping each other's trajectory. The Universal Turing Machine, a theoretical construct presented by Alan Turing in 1936 [1], is capable of simulating the behavior of any computational device by employing an algorithmic approach. The universality of the Turing machine concept played a pivotal role in the theoretical understanding of computation and the potential limitations of computation.

1.1.2. Slowdown of Moore's Law; Accelerators, ASICs, and Quantum Computing

The period of exponential increase [2] in computational power, as encapsulated by Gordon Moore, held true for several decades. However, as silicon-based technologies neared their physical limits, the pace of improvement began to slow, leading to a plateau in computational power. Conventional

computing started facing significant challenges, notably in solving complicated problems requiring massive computer resources, such as modeling molecular interactions or optimizing enormous systems [3]. To address the diminishing returns of traditional CPU scaling, the industry has turned towards specialized hardware accelerators, parallel computing, and application-specific integrated circuits (ASICs). Accelerators are specialized processors designed to handle specific kinds of workloads with greater efficiency than general-purpose processing units. Examples are graphics processing units (GPUs), tensor processing units (TPUs), and field programmable gate arrays (FPGAs). Enter quantum computing, a revolutionary approach leveraging the principles of quantum mechanics to process information. Quantum computing offers a promising choice as an accelerator due to some remarkable attributes such as parallelism and superposition, quantum entanglement, speed-up, and the potential to be implemented in hybrid approaches and machine learning. Quantum computing promises to break through the plateau, offering solutions to problems once thought intractable and paving the way for a new era of discovery and innovation.

1.1.3. NISQ era: Roadblocks and Challenges on the Path to FTQC

While the potential of quantum computing is compelling, there are several challenges that must be addressed. Quantum bits (qubits) are prone to several imperfections. The current era of quantum computing is constrained by the limited qubit count and the susceptibility of these qubits to decoherence and environmental errors, commonly referred to as quantum noise. Consequently, it has been termed the Noisy Intermediate-Scale Quantum (NISQ) era[4]. Numerous devices embodying the characteristics of this era have been developed and tested, providing an immediate platform for experimentation. This period has also spurred innovation in algorithms and compilation strategies tailored to noisy environments, such as Variational Quantum Eigensolver (VQE)[5] and Quantum Approximate Optimization Algorithm (QAOA)[6], which utilize classical optimization techniques to mitigate the constraints posed by limited quantum resources. However, the qubit counts in the few-hundreds, symbolic of the NISQ era, fall short of enabling fault-tolerant operation and demonstrating quantum advantage. Noise and scalability continue to pose significant roadblocks toward achieving Fault-Tolerant Quantum Computing (FTQC). Intensive research endeavors are underway to surmount these obstacles by exploring quantum systems with enhanced scalability potential, engineering solutions for prolonged coherence times, and devising error correction techniques through the development of diverse logical architectures. Overcoming these hurdles will pave the way for FTQC to unlock new realms of application, offering substantial computational advantages in addressing practical problems.

1.2. Theorist's Motivation for Quantum Computing

Quantum computing has revolutionized computation and information theory, challenging classical assumptions and inspiring new research. Its emergence has expanded the domains of information theory, complexity theory, and the theory of computation. In this section, we see some distinctive features of quantum computation and its impact.

1.2.1. Distinctive Features in Comparison to Classical Computing

Quantum computing is quite a distinct and novel field that fundamentally diverges from classical computing paradigms. This uniqueness can be attributed to many quantum properties such as (1) Superposition: Unlike classical bits that exist in a state of 0 or 1, qubits can exist in multiple states simultaneously; (2) Entanglement: The phenomenon where qubits become correlated, and the state of one qubit influences the state of another; (3) Reversibility: Quantum operations that are performed using quantum gates are inherently reversible that can be possibly exploited to bypass the thermodynamic costs of computation; (4) Quantum measurement: A probabilistic process introducing uncertainty in quantum systems.

1.2.2. Impact

The field of quantum information is well-established as an interdisciplinary field involving quantum mechanics, computer science, and information theory. It encompasses critical domains such as quantum algorithms and computing, communication and cryptography, and the extension of classical information theory definitions to encompass quantum information. Moreover, its impact on theoretical computer science and complexity theory is profound. For instance, the introduction of quantum Turing machine models was driven by generalizing the definitions and incorporating principles of quantum me-

chanics into the universal Turing machine model. The emergence of quantum computing algorithms for solving problems in polynomial time, with a bounded error probability, described by the complexity class BQP [7], violates the extended Church-Turing thesis [8].

1.3. Roadmaps

At present, the trajectory of quantum computing research can be broadly mapped across four distinct avenues. Listed below are the key avenues following a bottom-to-top stack approach, where substantial advancements are being pursued to drive forward the field of quantum computation:

1. At the hardware level, the research community is actively exploring new materials for the fabrication of qubits [9] and engineering of better design features [10]. These endeavors are targeted at mitigating the cause behind different noise factors that are currently responsible for plaguing the quantum processors, improving coherence times, and building space-efficient and more accurate control and measurement circuitry, leading to higher scalability.

2. The current road map in Quantum Error Correction focuses on improving both the number and fidelity of logical qubits through the application of various techniques tailored to specific platforms [11]. This pursuit involves exploring novel logical architectures [12] and error correction codes [13] designed to achieve lower error rates and increased scalability. These efforts represent a crucial pathway toward unlocking quantum advantage and facilitating the realization of practical quantum computing applications.

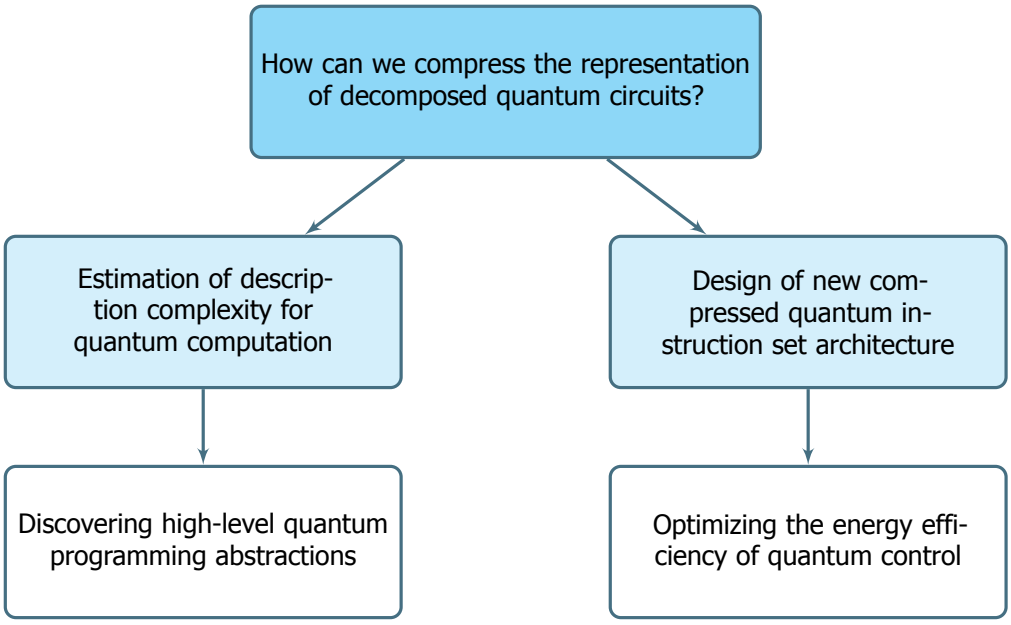
3. Conventional research in low-level control/compilation in quantum systems emphasizes the integration of software and hardware through advanced low-level control techniques. The community and industry are taking initiatives for the adoption of standardized representations such as OpenQASM3 [14] and the exploration of multi-level intermediate representations (MLIR) to streamline quantum programming and compilation processes [15]. Exploration of a cryogenic control architecture for scalable and integrated control processes [16], the quest for building more efficient QISA by handling trade-offs between different resources [17], and engineering optimal pulse shaping are some of the approaches that hold great promise.

4. Variational circuits and algorithms are finding novel applications in developing new hybrid algorithms ranging from solving optimization problems to performing molecular simulations. Quantum machine learning (QML) is a fascinating field leveraging quantum computation in different machine learning models for classification and clustering, feature selection [18], and optimization. QML techniques aim to offer advantages in areas such as generalization, accelerated learning rates, and exploration of higher-dimensional data spaces [19, 20].

Building upon the outlined roadmaps, this project is targeted at the 3rd avenue, involving improvements in compilation and more efficient QISA design. Embracing concepts from resource theory and descriptive complexity and taking cues from the efficiency gains seen in code compression within embedded systems, this endeavor seeks to harness similar principles for quantum computation.

1.4. Research Question: Compressed Representation of Quantum Computation

The central idea of this thesis revolves around the synthesis of quantum circuits into a discrete basis and finding a compressed expression of the quantum instruction stream. With the growing need for higher computational power and the functional limits of conventional circuitry, the size of the instruction stream going to the processor increases significantly, exerting a substantial and unproductive overhead in terms of system energy and processor core power. This problem is ominously present in quantum systems as well. The growing number of qubits and more sophisticated quantum algorithms and protocols, such as error-correcting regimes intensify the need for effective control processes. Against this backdrop, the research question at the heart of this study emerges:



This thesis aims to answer the main research question and, in the process, address its sub-aspects. The study starts with a review of computational resources with a particular focus on description complexity and its relation with other resources. This exploration prompts the analysis of quantum circuits from an information theory viewpoint and establishes a theoretical foundation that supports compression strategy as an estimate of the description complexity. The following two aspects are geared toward the practical application of encoding decomposed quantum circuits for generating compressed representations of quantum instruction streams, eventually empowering energy-efficient quantum control.

1.4.1. Relevance

Theoretical Justification

In the field of theoretical information theory and resource theory of computation, efforts have been made to establish a link between the thermodynamic cost of computation and the associated evolution of algorithmic complexity.

One of the most notable results in this field is the Landauer limit [21] which presents a bound on the minimum energy dissipation for performing an irreversible computation. Following this seminal result, significant attention, most notably by [22], [23], [24] and [25] has been directed towards establishing a direct correlation between the thermodynamic cost of computation in the form of change in entropy and the change in algorithmic complexity. Despite these attempts, practical methods to estimate the algorithmic description complexity remain limited. In this project, we develop and propose an estimate of the description complexity stemming from the encoded representation of decomposed quantum circuits. This method will offer a lens to evaluate quantum circuits based on their description complexity and help discover high-level quantum programming abstractions.

Practical Justification

Quantum processors operate on quantum principles, yet their control mechanisms remain classical. Control instructions, such as pulse information or Quantum Assembly Language (QASM), vary depending on the abstraction level. Therefore, the description of quantum processes has to be established at the Quantum Instruction Set Architecture (QISA) level. The thesis offers the way of estimating this description complexity by building a compressed instruction stream based on updated definitions of code-words.

The exploration of the design space of code compression in low-power embedded systems is motivated from the increasing computational load and constrained chip resources.. This thesis aims to perform this exploration of code compression in the domain of quantum computing.

Contemporary attempts at engineering cryogenic control architecture encounter challenges posed by stringent power budget as operational temperatures decline. The utilization of compressed instruction streams presents an avenue for mitigating energy consumption concerns within cryogenic

environments. Such compression not only enhances operational efficiency but also diminishes noise levels during the transmission of instructions from ambient to cryogenic temperatures.

1.4.2. Methods to be Explored

The quantum compilation process is essential for implementing quantum unitaries or circuits on quantum hardware. Central to this process is the synthesis of the quantum unitary in terms of a native gate-set. The investigation of this thesis starts with the exploration of decomposition of Harr-Random unitaries for 1q systems into a discrete basis of gates using Solovay-Kitaev decomposition (SKD). We adopt the qiskit implementation of the SKD algorithm and expand its scope from a hardcoded set of single qubit gates to operate with any general set of gates input by the user.

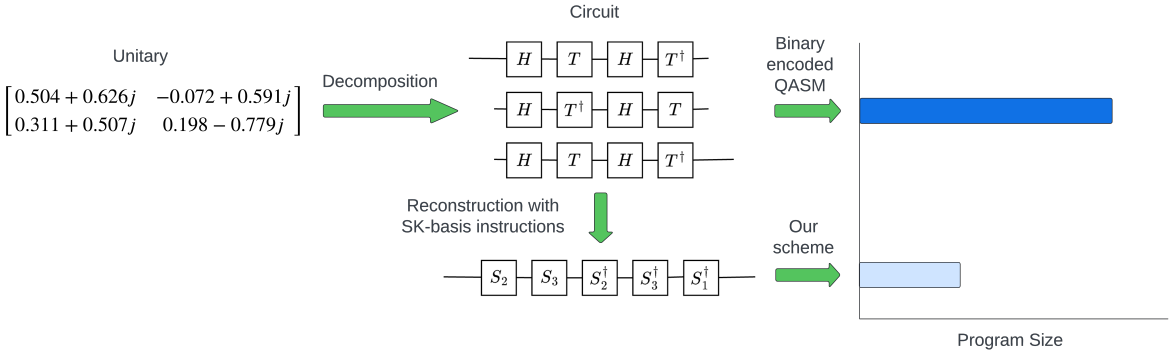


Figure 1.1: Pipeline of compression framework

Following this foundational exploration, the investigation proceeds to delve into the application of Huffman Encoding in three distinct iterations integrated with the decomposition function. The objective is to derive a compressed instruction stream. This effort results in the development of a Python tool capable of returning an encoded instruction stream

Subsequently, the scope is broadened to include multi-qubit systems, wherein the decomposition and compression techniques are scaled up to accommodate the complexities inherent in such systems. The decomposition and compression scheme are then tested on selected benchmark circuits. These circuits are real quantum algorithms that are popularly studied and used and this experiment aims to quantify the description complexity of 'useful' circuits and potentially discover high-level contexts.

1.4.3. Overview of the Thesis

The thesis is segregated into 6 chapters and follows a structured narrative that mirrors the sequential progression of the project's development.

- In Chapter 2, the story starts with a build-up of the background of algorithmic information theory and the foundation for this thesis: description complexity.
- Chapter 3 describes different ways of describing a quantum system and explains our choice of the unitary matrix as the quantum description. Methods for the decomposition of single and multi-qubit unitaries into quantum circuits and the proposed methodology of reconstructing the circuits using Solovay-Kitaev basis sequences are delineated in great detail.
- Chapter 4 lays down the framework of Huffman coding and explains the integration of Huffman coding in our decomposition method to obtain encoded quantum instruction streams for describing the unitary.
- In Chapter 5, we will have a broader look at the quantum stack and contemporary design of QISA. We evaluate the methods proposed in the thesis and elucidate the theoretical and practical contributions of the work.
- Finally, in Chapter 6, we conclude the thesis with an outlook of the project.

2

Background

This chapter contains the theoretical background for the thesis project. It outlines the literature review phase of the project and covers essential concepts in the resource theory of computation, algorithmic information theory, and the extension of these ideas in quantum computing. We focus particularly on description complexity, its formulation, related measures, and ways to estimate it. We analyze compound measures that involve multiple computational resources and eventually present a table that aggregates these measures.. This activity is significant in laying the groundwork for the project and inspires the development of experiments for the estimation of description complexity for quantum circuits and the subsequent search for practical applications.

2.1. Resources for Computation

The study of computational resources is essential for understanding the capabilities and limitations of computing systems and the algorithms that run on them. Computational resources refer to the various means and constraints under which computational tasks are executed. The expenditure of computational resources, as studied and formulated in the domain of classical computing, also holds analogous significance in quantum computing systems. Therefore, we need to build a comprehensive background on resource theory for computation in order to understand the resource costs for quantum computing and control and, in turn, improve the resource efficiency of quantum algorithms and other processes in quantum computation.

2.1.1. Different Resources for Computation

Defining something as a resource for computation involves identifying factors that influence the efficiency, scalability, and feasibility of computational tasks. While traditional resources like time and space complexity provide foundational metrics, novel resources encompassing unconventional measures and domain-specific considerations are valuable to assess the computational workload. Below are some computational resources that are foundational and find relevance to this project:

1. **Time Complexity:** The amount of time taken by an algorithm or process to complete. The *asymptotic time complexity* is the hardware agnostic measure of the number of time steps that scales as a function of the input size. The *runtime* or actual time is the time taken by the algorithm that depends on both the input size and the execution time on the machine.

2. **Space Complexity:** The amount of memory or storage space an algorithm uses during its execution. Total space complexity includes the *input space* taken by the algorithm depending on the input size and the *auxiliary space*, which is the extra/temporary space taken by the algorithm during execution.

3. **Energy:** Energy, or the thermodynamic cost of computation, is a crucial computational resource. It can be characterized by the heat incurred by running an algorithm. Studying the energy cost of algorithms can be motivated by operational power limitations of low-power or cryogenic devices or be aimed at reducing the carbon footprint and operational costs of devices.

4. **Description Complexity:** The description complexity, formally known as the Kolmogorov complexity, measures the amount of information required to describe an object (such as a string or a data structure) in the shortest way possible. This measure is theoretically well-formulated and studied in the domain of classical information theory. It sets a theoretical bound on the minimum size of programs running on a formal universal automata for performing a particular task.

5. **Approximation:** In certain situations, obtaining an exact deterministic solution is expensive in terms of specific computational resources such as time or space. Approximation algorithms aim to provide near-optimal solutions with guaranteed bounds on their performance.

2.1.2. Units and Ways of Estimation of Resources

One of the most essential characteristics of computational resources is quantifiability. A resource should be measurable or quantifiable in a meaningful way. This provides insight into the impact of resource costs on computation and helps in a comparative analysis of different algorithms or processes. Going over the units and ways of estimation and quantification of the resources listed in the above subsection:

1. **Time Complexity:** The asymptotic time complexity can be expressed in different notations and is estimated by counting the number of time steps in different situations based on the size of the input N:

   - Big-theta notation $\Theta(N)$ represents both the upper and lower bound on the number of time steps for an algorithm and is often used to analyze the average-case time complexity.
   - Big-O notation $O(N)$ represents the upper bound on the number of time steps for an algorithm and describes the worst-case time complexity.
   - Big-omega notation $\Omega(N)$ represents the lower bound on the number of time steps for an algorithm and describes the best-case time complexity.

   The actual time or runtime, on the other hand, also includes the time taken for compilation and execution of an algorithm on a hardware device and would be ideally quantified in terms of wall-clock time (Seconds) or in the number of clock cycles.

2. **Space Complexity:** Similar to asymptotic time complexity, the asymptotic space complexity can be estimated in different notations (Big-O, Big-Θ and Big-Ω). These estimates are again expressed as functions of the input size. The auxiliary space includes the program memory (instruction fetch) and the working memory (data fetch). The program memory is nonvolatile and stores the hardware-level instructions, and the data memory is used to store the values of variables and parameters during the execution of a process. Both program and working memory are quantified in units of bytes and their multiples: Megabytes and gigabytes.

3. **Energy:** The energy consumption can be measured in Joules (J) or kilowatt-hours (kWh), depending on the scale and context of computation. The energy consumption can be estimated by power monitoring tools to determine the energy consumption per cycle and clock speed of specific computing devices and account for the energy cost of the transmission of information in cables.

4. **Description Complexity:** There is no general routine to calculate the exact Kolmogorov complexity of an arbitrary string. This limitation bears a resemblance to the halting problem, undecidable problems, or Gödel's incompleteness theorem. However, the description complexity can be approximated using different techniques. Standard ways of estimating the description complexity can be the compressed bit-length of a string or program using lossless compression methods such as BZip2 [26] and LZ [27] that are aimed at exploiting statistical regularities. Alternate ways, such as BDM [28], estimate the algorithmic complexity of smaller blocks of strings, which are then summed up to reconstruct the original data. BDM offers a hybrid approach to evaluating the complexity by combining Shannon entropy in the long-range with local predictions of algorithmic complexity.

5. **Approximation:** The performance of approximation algorithms is typically measured in terms of the approximation ratio or factor, which compares the algorithm's solution to the optimal solution. Approximation algorithms can be implemented in many ways:

   - Approximate Circuits can be substituted in place of blocks of complicated circuits for performing arithmetic operations that reduce the time/space overhead. Software-level approximation techniques such as memoization, loop perforation, and task skipping are some strategies that aim to trade off the accuracy of the program with execution time.
   - Approximate storage and memory refer to devices with lower /controlled refreshed rates that lead to improvement in the working memory cost and also energy consumption.

2.1.3. Quantum Computing vs Classical Computing Based on Resources

Drawing parallels between computational resources in a classical and quantum setting involves accounting for the principles of quantum mechanics and how they impact the estimation of these resources. We'll focus on the gate-based model of quantum computing for this comparison and throughout the thesis, although other models exist, such as adiabatic quantum computing, measurement-based quantum computation, and QTM. Exploring how these resources translate into the context of quantum computing as listed in the preceding subsections:

Resources	Classical Computing	Quantum Computing
Time	Asymptotic Time Complexity: Number of time steps for the algorithm, considering unit time for each operation as a function of the input size. Total time complexity includes the runtime in addition to the asymptotic time complexity.	Quantum Circuit Depth: Total execution time of all the gates in a quantum circuit. Parallel operations can be performed in the same time step. Gate execution times and coherence time specific to hardware platform.
Space	Asymptotic space complexity: Amount of "memory units" needed for the algorithm as a function of the input size. Memory units are abstract in nature and can denote bits, words, or the number of nodes, depending on the problem and data structure used. Total space complexity also includes the runtime memory (in bytes).	Number of qubits required for algorithmic operations, with qubits representing multiple classical parameters/variables simultaneously through superposition. Offers space-efficient solutions for complicated problems, yet qubit resources are still low for a definite quantum advantage.
Energy	Energy in Joules/KWh expended by the processor while performing an algorithm or computational process.	Operational energy costs incurred by the control architecture and the refrigeration process (usually very energy expensive). Algorithmic energy cost potentially more efficient than classical computing.
Description	The shortest description of the algorithm or problem instance in a fixed computational model (UTM model).	The shortest quantum description of the circuit/algorithm that generate a specific quantum state starting from an initial state. Depends on the theoretical model for formulation of the quantum description complexity and the description of the quantum states themselves.
Approximation	Near optimal solutions by modified algorithm design, memory and time management for execution with a guaranteed performance bound.	Approximate quantum algorithms with upper bounded error probability. Exploration of trade-offs between process fidelity and depth / no. of qubits / compressed quantum circuit.

Table 2.1: Quantum Computing vs Classical Computing based on Resources

2.2. Examples of How Resources Become a Bottleneck

Example 1:

NP-Hard problems: This is the complexity class of problems that are as hard to solve as the hardest problem in the NP (non-deterministic polynomial time) class. Some examples of this class of problems

are the traveling salesman problem, the knapsack problem, and the satisfiability problem. The solutions to these problems take polynomial time in non-deterministic Turing machines; however, simulating them on deterministic Turing machines–those that model real-world computers–can cost exponential time in the worst case. As the problem size increases, it becomes infeasible to solve them with conventional computational strategies and resources.

Mitigation: Approximation Algorithms; Instead of aiming for the exact solution, approximation algorithms provide solutions that are close to optimal within a reasonable time frame.

Example 2:
The number of qubits for useful quantum computing applications such as drug discovery and molecular simulations. The quantum resource cost, such as the number of qubits, scales exponentially with molecule/problem size and often faces a bottleneck, given the contemporary platforms in the NISQ-era.

Mitigation: Incorporation of classical optimization and hybrid strategies for faster spanning of the solution space and convergence to compensate for the limited qubit count. Exploration of analog and digital-analog hybrid computation strategies specialized for specific class of problems designed to harness particular benefits of the hardware platform.

Example 3:
Theoretical limits and bounds:

- Landauer Limit: The theoretical limit of the minimum cost of energy for erasing 1 bit of information.

$$E \geq k_B T \ln 2 \tag{2.1}$$

  k_B is the Boltzmann constant and T is the operation temperature of the system. Essentially a thermodynamical limit in computation, this bound sets a constraint on the number of irreversible logical operations that can be performed while dissipating a fixed amount of energy.

- Quantum Speed Limit Theorems: Theoretical limitations that set a constraint on the minimum time it takes to evolve from one orthogonal state to another. Two popular theorems are Mandelstam-Tamm theorem [29]

$$t \geq \frac{h}{4\delta E}; \quad \text{where, } \delta E = \langle\psi|H^2|\psi\rangle - (\langle\psi|H|\psi\rangle)^2 \text{ is the variance in system energy} \tag{2.2}$$

  and the Margolus–Levitin theorem [30].

$$t \geq \frac{h}{4\langle E\rangle}; \quad \langle E\rangle = \langle\psi|H|\psi\rangle \text{ is the average energy} \tag{2.3}$$

- Beckenstein's Bound: This limitation was proposed by Jacob Beckenstein in 1981 [31] that imposes a bound on the maximum entropy that can be contained in a finite region of space and energy. In the context of computation, it can be seen as a limitation on the amount of information that can be processed or stored within a physical system.

$$S \leq \frac{2\pi k R E}{hc} \tag{2.4}$$

  S represents entropy, k is the Boltzmann constant, R is the radius of a sphere that can enclose the system, E is the total mass-energy (including rest masses), h is the reduced Planck constant, and c is the speed of light.

2.3. Description Complexity

The Description Complexity for an object is the measure of the amount of information required to describe it. It is also known as Kolmogorov Complexity, Algorithmic Complexity, or Algorithmic Entropy. In the context of information theory, the description complexity of an object can be defined as the length of the shortest program written in a set language/description that can compute the object as output. It forms an essential concept in Algorithmic Information Theory and is named after the mathematician Andrey Kolmogorov, who introduced this idea for the first time in 1963 [32]. Mathematicians Ray Solomonoff and Gregory Chaitin are also accredited for contributing independently to the study of algorithmic complexity and laying the foundations of algorithmic information theory.

An illustrative example of a binary string of length 48 is adopted. The string can possess varying levels of description complexity. The 3 cases shown below are Python codes for printing the string. It must be noted that the instruction for printing the string in any other language also qualifies for a valid description complexity, which may be even more concise.

```
print('010101010101010101010101010101010101010101010101')          # 57 characters
```

```
[print('010101010101', end="") for _ in range(4)]                   # 49 characters
```

```
[print(_ % 2, end="") for _ in range(48)]                           # 41 characters
```

2.3.1. Turing Machines

Before delving into the formal definition of description complexity, it is essential to lay the groundwork for a (Universal) Turing machine. Proposed by the British mathematician Alan Turing in 1936, the Turing machine stands as a cornerstone in the theory of computation, providing a fundamental framework for comprehending the constraints and potentials inherent in algorithmic processes.

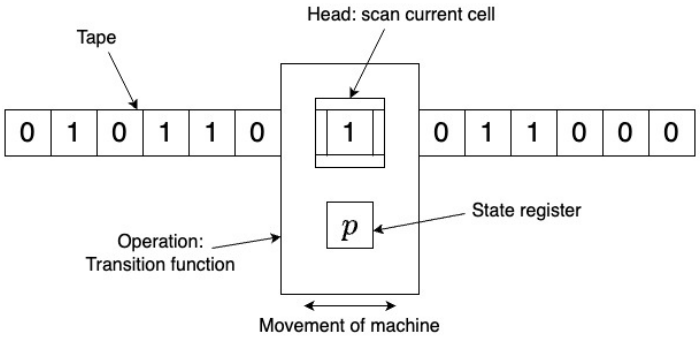


Figure 2.1: An imaginary physical Turing machine with its various components labeled

As shown in the diagram above, it consists of the following components:

1. Tape: An infinitely long tape divided into cells, where each cell contains a symbol from a finite alphabet. The tape serves as the machine's memory. There's the possibility of an output tape along an input tape as well (not shown in the figure)
2. Head: A read/write head that can move left or right along the tape.
3. State Register: The specific state of the machine at any given time that determines its behavior.
4. Transition Function: A set of rules or transitions that specify the machine's behavior. The transition function dictates what action to take (read, write, move left, move right, change state) based on the current state and the symbol read from the tape. The

The key idea behind the Universal Turing Machine (UTM) is its capability to interpret and execute the instructions of any Turing Machine, effectively making it a universal model for computation. It involves encoding the descriptions of other Turing Machines and their inputs on its tape. The UTM reads this encoded information, interprets it as the description of another Turing machine, and simulates its operations. The notion of a Universal Turing Machine is fundamental to the Church-Turing thesis, which suggests that any effectively calculable function can be computed by a Turing machine (or equivalent model). The algorithmic information of quantum computation can be described using classical instructions [33]. Therefore, the classical construct of UTM, which will form the foundation for discussing Kolmogorov complexity, can be extended to the domain of quantum computing.

Now that we have established the basis of a UTM, we can move back to description complexity. We say that a program p is a description of a string s if the program p when run on a UTM U, outputs the string s. We can write it as $U(p) = s$. The length of the shortest description for the string can be stated as:

$$K(s) = \min_{p}\{l(p) : U(p) = s\} \tag{2.5}$$

Here, $l(p)$ represents the length of the program p measured in bits. Hence, $K(s)$ is the minimal length of the program that computes the output string s. Therefore, the Kolmogorov complexity provides an estimate of the intrinsic complexity or compressibility of a string. In fact, it can be considered as the theoretical limit on the highest compression that can be achieved by a general-purpose compression technique, and it inspires researchers and developers to reach as close to this limit as possible. Some of the key properties of Description Complexity are listed below:

Key Properties

- **Invariance Theorem:** The description complexity of a string is "nearly" invariant of the choice of the description language or the UTM U. If we go from a description language L_1 to L_2, the description complexity changes at most by a constant c that is independent of the string s and only depends on the description languages L_1 and L_2. The constant factor can be interpreted as the cross-compiler length between languages. The proof for this theorem was proposed independently by the three mathematicians, Kolmogorov in [32], Solomonoff in his two-part paper [34], and Chaitin in [35]. This property shows the universal nature of description complexity and marked a pivotal moment in the birth of algorithmic information theory.
- **Semi-computability:** It is not possible to have a program that can take an object or string as input and compute the exact Kolmogorov complexity as the output. However, it can be approximated and bounded using different techniques and heuristics.

  Upper Bounds: The description complexity for a string cannot be greater than the length of the string, i.e., $K(s) \leq l(s) + c$. The constant factor c follows from the invariance theorem.

  Determining the actual value of description complexity is challenging due to the halting problem. Techniques such as lossless compression, block decomposition, prefix encoding and concept discovery help in improving estimation strategies of description complexity.
- **Symmetry of Information:** For strings s and t, $K(s,t) \approx K(s) + K(t|s)$, where $K(s,t)$ represents the description complexity of the pair (s,t), and $K(t|s)$ is the complexity of t conditional on s. The conditional Kolmogorov complexity $K(t|s)$ can be interpreted as the information content of the process that computes the string t with string s as the input.

  In essence, this property illustrates that the complexity of both the strings together roughly equals the complexity of the first string and the complexity of the second string that is not contained in the first one. This property explains the interconnectedness of information and forms the foundation for the study of thermodynamics of quantum computation in [23]

2.3.2. Related Measures and their Estimation

The extensive study of description complexity in the context of algorithmic information theory has led to the formulation of different related measures that provide meaningful insights.

Martin Löf Randomness

Martin Löf randomness, named after Swedish mathematician Per Martin-Löf is a formalism that classifies whether binary sequences (or strings in general) are "random" based on a number of tests. Strings that pass the tests are termed as being Martin-Löf random and are incompressible. These tests are heuristic in nature can be set according to the problem at hand and are aimed to recognize statistical patterns or context in sequences. These statistical patterns are exploited by compression algorithms. In their absence, the data is said to have a high description complexity and is rendered incompressible.

Algorithmic Solomonoff Probability (AP)

Ray Solomonoff presented this approach in his *Theory of Inductive Inference* [34] as an a priori probability that can be used to predict future data from a string of past data. Mathematically, the Solomonoff probability of a string s is given by

$$AP(s) = \sum_{p:U(p)=s} 2^{-l(p)} \tag{2.6}$$

where, the sum is taken over all the programs p which when run on a UTM U output the string s. $l(p)$ denotes the length of the program p, and the term $2^{-l(p)}$ is associated with the universal probability distribution of all such programs. Solomonoff probability has a close connection with description complexity from the evident major contribution to the Solomonoff probability from simple/shorter programs [36]. His theory formalizes Occam's razor by allocating higher probabilities and favoring programs with shorter descriptions. However, the key challenge with Solomonoff's probability is that it's non-computable and can only be estimated in a practical setting. This also stems from the fact (as in the case of Kolmogorov complexity) that there is no general method to enumerate all the programs that compute the output string s and halt.

2.3.3. Review of AIT in Quantum Information

Many of the above-discussed concepts of algorithmic information theory can be extended to the scope of quantum computation. This is a promising field of study aimed at understanding the resource costs of quantum computation and contributes to the design and optimization of quantum algorithms at a higher abstraction level. Before delving into this section, it is essential to select a formal model of quantum computation for discussing further ideas. Just as Kolmogorov complexity is formally defined on a UTM model of computation, the introduction of quantum Kolmogorov complexity necessitates a quantum Turing machine model (QTM). The concept was introduced by David Deutsch by presenting the QTM model as an extension of the Church-Turing thesis [37].

The notion of quantum Kolmogorov complexity for a quantum state refers to the minimum amount of either quantum [38] or classical [33] information needed to describe a program. This program, when executed on a universal quantum computer, produces the desired quantum state with high accuracy. In this thesis, we delve into techniques for estimating the description complexity, which has a classical representation, for quantum computation. One of the most fascinating articles that propels our investigation in this thesis is *'The second law of quantum complexity'* by Brown and Susskind [22].

This paper sets forth a correlation between the quantum complexity of a system with K qubits and the evolution of the entropy of a classical system with 2^K degrees of freedom. The analysis starts with the formal introduction of both the quantum and classical systems. The circuit complexity of the quantum system is explained as an evolution of Hamiltonian for the quantum system on a curvature termed as 'complexity geometry' [39]. The authors develop the premise for the auxiliary classical system by studying the progression of the total entropy of a non-relativistic particle. The total entropy evolution is influenced by two factors: the change in positional entropy, which parallels the evolution of quantum computational complexity, and the shift in kinetic entropy, which is analogous to the quantum Kolmogorov complexity. All this study eventually leads a connection between the second law of thermodynamics and its proposed "second law of quantum complexity" in quantum information theory.

Figure 2.2: Corollary from the above correspondence between thermodynamics and algorithmic complexity theory: Free energy as a resource for performing useful work translates to "uncomplexity" as a resource for performing quantum computation.

2.3.4. Compound Metrics and Trade-offs

The resource expenditure for a program can be evaluated as a cost function involving more than one computational resource. Such measures can be called composite resource costs (or compound metrics in this thesis). They serve a great importance in the study of information and resource theory, as composite measures provide a trade-off between the involved resources. The exploration and fine-tuning of these trade-offs aid in developing algorithms that are optimized for the multiple constraints present in real-world applications. This section enumerates some compound metrics involving the resources introduced in 2.1.1 and subsequently presents a table 2.3 that offers a concise and comparative study

of different compound metrics, the involved resources, and the nature of their inter-relation in the cost measure.

Levin Complexity

Leonid Levin introduced the Levin complexity [40], or K_t as a refinement of the Kolmogorov complexity by incorporating the time it takes to use that description to reconstruct the original string. It can be of greater practical interest, as it is a more nuanced view of complexity that considers both the description length and computational time.

$$K_t(s) = \min_p \left(l(p) + \log(t(p))\right) \tag{2.7}$$

where p is a program, when run on a UTM U, outputs the string s, $l(p)$ is the length of the program, and $t(p)$ is the time taken by the program to return the string s as output [41]. The logarithm balances the contribution of time complexity and the length of the program in bits towards the total Levin complexity. Levin complexity qualifies as a composite resource measure that is covered in greater detail in the following subsection. The uncomputability problem common to the above measures related to description complexity also translates to Levin complexity, and it can only be estimated in experiments. For example, imposing an additional time-bound during the estimation of Kolmogorov complexity.

Logical Depth

Logical depth [42], formulated by Charles Bennett captures the intuitive notion of "depth of information" in addition to the description complexity of an object. It can be considered as a composite resource measure, as it is the time taken by a UTM to compute a string using the minimal program that outputs the string.

$$D(s) = \min_p \{t(p) : l(p) - l(p_0) < S \wedge U(p) = s\} \tag{2.8}$$

where, $l(p)$ is the length of a program p, p_0 is the shortest program that outputs the string s and $t(p)$ is the time taken by the program to run and halt on UTM U. S is the significance parameter that is set to pick only those programs whose lengths differ within that range from that of the minimal program. It adds a temporal dimension to the measure of the information content of strings.

An approximate version of the logical depth can be defined by modifying the second requirement of the above relation 2.8. This can be done by selecting an error bound and widening the range of programs that generate an output within the error bound of the target string s. This process extends the scope of this composite resource to also include approximation as a resource.

Speed Prior

Following along the same line as Solomonoff's algorithmic probability, speed prior [43] is a prior probability distribution proposed as a computable precursor for inductive inference. It extends the scope of AP to also account for the time complexity of programs and resonates more deeply with Occam's razor by favoring not only the most simplistic programs, but also those with a lower time cost. The relation between AP and Kolmogorov complexity is equivalent to that between Speed prior and Levin complexity.

Energetic Costs of Computation

The composite resources listed so far earlier involve resources such as description, time, and sometimes space, yet a connection with energy remains absent. Many efforts have been made to establish a connection between algorithmic information theory and thermodynamics [44], particularly in developing a theoretical framework for understanding the thermodynamic costs of computation [25] [24].

One of the most influential works in this field was proposed by Zurek in the 'Thermodynamic cost of computation' [23]. It presents a lower bound on the increase in entropy caused by a computational process that takes an input string s to an output string t. This lower bound is equal to the conditional Kolmogorov complexity involving the two strings.

$$\Delta S(s \to t) \approx K(s|t) \geq K(s) - K(f) \tag{2.9}$$

Landauer's principle is invoked in the analysis, and it is explained that compression of information (from an original string s to s') can be performed reversibly without any expenditure of energy but that it leads to the expense of storing (the number of bits corresponding) to the shortest program that

performs the change ($s \rightarrow s'$) that is, $K(s|s')$. Based on these arguments, the author has refuted Maxwell's demon's challenge to the second law of thermodynamics.

As an extension to this paper, a generalized version [45] has been formulated to extend its application to any practical computation and also quantum computation. This generalization has been attained by considering an ensemble of computational processes for a given pair of input and output in place of a single process.

Metrics \ Resources	Space	Time	Description Complexity	Energy	Approximation
Quantum Volume	$V_Q = \min[N, d(N)]^2$ Maximum size of square quantum circuits that can be implimented by a computer				
Pebbling Game	$O(T,S) \rightarrow O(T^{1+\epsilon}, S \log T)$ Tradeoff between space and exponentially bounded time				
Logical Depth		$D_d(x) = \min_p\{T(p): (\|p\| - \|p^*\| < d) \wedge (U(p) = x)\}$ Measure of time subject to an upper bound on description complexity			
Approximate Logical Depth		$D'_d(x) = \min_p\{T(p): (\|p\| - \|p^*\| < d) \wedge (U(p) \approx x)\}$ Similar to logical depth, with an allowed error in string that is bounded			✓
Levin Complexity		$K_t(x) = \min_p\{l(p) + \log(t(p)): U(p) = x\}$ Additive time-bounded definition of KC.			
Speed Prior		$S(x) = \sum_{i=1}^{\infty} 2^{-i} \sum_{p \rightarrow i,x} 2^{-l(p)} \approx 2^{-K_t(x)}$ Prior distribution with exponential relation to Levin's K_t complexity			
Zurek's Energy			$\delta S(i \rightarrow o) \geq K(i) - K(o)$ Lower bound on energy for the conditional desciption complexity of replacement		
Uncomplexity	$\Delta \mathcal{C} = \mathcal{C}_{max} - \mathcal{C}$ Resource for performing directed quantum computation				

Figure 2.3: Table enumerating compound metrics involving multiple computational resources.

2.3.5. Estimation of Description Complexity

At this point, we have established the limitation in the computability of Kolmogorov complexity and its related measures. Despite these limitations, it can be approximated in practical settings, such as compression. Reiterating the definition of description complexity, it is the shortest possible description of a piece of data, and a compressed version is essentially a concise description of that data.

Lossless compression techniques attempt to find patterns and regularities in data and exploit these to generate a shorter description. The more patterns there are, the more it can be compressed, and the lower the description complexity. The fabrication of a better compression technique that can achieve a higher compression ratio (size of original data/size of compressed data) leads to an improved lower bound for description complexity. The widespread compatibility of compression techniques with diverse data types such as text, code, and images aid in extending the information-theoretic measure of description complexity to larger domains. The normalized information distance [46] uses compression as an estimate of Kolmogorov complexity for data and puts forth the information distance between two objects as a function of their description complexities. An application of this concept, the normalized Google distance [47], is implemented in the Google search engine to group similar data based on the number of hits on the search engine.

The block decomposition method (BDM) [28] is a computational technique proposed to circumvent the direct computational limitation of Kolmogorov complexity and provide an estimate of the algorithmic

complexity. It is based on the premise that, while description complexity of massive objects are non-computable, approximating it for smaller blocks of data is feasible. The technique aims to do this by breaking up the original data into smaller chunks of size $n \times n$. The algorithm constructs a database that maps all possible $n \times n$ blocks to their respective description complexities by a method known as the coding theorem method (CTM). The method hinges on using algorithmic probability 2.3.2 for defining the complexity of strings.

$$CTM_{(t,k)}(x) = -\log_2 D_{(t,k)}(x) \tag{2.10}$$

Where $D_{(t,k)}(x)$ is the probability distribution for a Turing machine U with t states and k symbols to halt for the string x.

$$D_{(t,k)}(x) = \frac{n\{U \in (t,k) : U \text{ produces } x\}}{n\{U \in (t,k) : U \text{ halts}\}} \tag{2.11}$$

$n(S)$ is the cardinality of the set S. The formalism for this finite Turing machine and the condition of its halting for small values of (t,k) are defined in the Busy Beaver problem [48]. For $t = 4,\ k = 2$, the Turing machine has a maximum runtime of $l(x) = 107$ [49], beyond which it's assumed not to halt. The total complexity by the BDM method for the original data (x) as a sum over fragments x_i is:

$$BDM(x) = \sum_i CTM(x_i) + \log(n(x_i)) \tag{2.12}$$

where, $n(x_i)$ is the number of times the fragment x_i occurs in the original data. BDM provides a good approximation for the algorithmic complexity of objects; however there can be some limitations to using it as an estimation tool of description complexity for a wider domain of data.

1. At a local level, it offers a close estimate of the description complexity. It is guaranteed by the CTM, which uses principles from Solomonoff's probability 2.3.2 and Levin complexity 2.3.4. However, at a global level, it behaves close to the Shannon entropy of the data and, therefore, fixates on the statistical regularities.

2. Extending its application to the domain of quantum computing can prove challenging as quantum states and operations might not lend themselves to decomposition into blocks in the same way as classical data structures do, considering quantum entanglement and superposition. In the way the measure is defined, it may fail to capture the description and context of quantum operations that are necessary for estimating the quantum description complexity

To conclude this section, we established the concept of description complexity and listed related measures that can be used to gather insights about the complexity of objects. We conducted a review of algorithmic information theory and looked at various compound measures that involve description complexity and other computational resources. After studying multiple ways of estimating description complexity, we choose compression as an estimation for description complexity. To reinforce this decision, the following section sheds light on some implementations of code compression for improving the efficiency of computation in related domains.

2.4. Code Compression in the Domain of Low-Power Embedded Systems

The last section of this chapter covers the application of code compression in the domain of low-power embedded systems. Low-power embedded systems are computing systems designed for specific functions with a primary focus on minimizing energy consumption. These systems are integral to a wide range of applications, from consumer electronics to industrial control systems, where power efficiency is critical. Code-compression techniques are advantageous in overcoming the energy cost, size, and memory limitations of such systems.

The study of design space of code compression [50] is an active field of research, and it involves the choice of encoding techniques: statistical vs. dictionary-based schemes, minimizing the decompression overhead, hardware cost, and power usage. The RISC-V ISA developed by UC Berkeley [51] is an open-source ISA for standard and special purpose utility and natively operates with fixed-length 32-bit instructions. It has many extensions for applications in embedded systems, personal computers, supercomputers and also quantum computation. It has extensions like *RVC* [52] for employing variable-length op-codes and bringing down the bandwidth of the instructions. Approaches like [53] compare the performance of some of these extensions and explore the design space of compressed ISA by

employing arithmetic encoding to assign shorter-length opcodes to more commonly used instructions and increase the code density.

Extending the principles of code compression and efficient encoding to the domain of quantum computation is meaningful for multiple reasons. Investigation of compression in the field of quantum computation grants a strategy for estimating the quantum description complexity of quantum circuits. Secondly, the attempts to move the quantum control architectures closer to the operating temperatures of quantum processors impose severe limitations on the energy budget. Cryogenic control of qubits offers better integration with the processor and shorter process times, eventually qualifying for a promising scalable alternative for quantum control [54]. Increasing the code density can decrease the energy cost of information transfer from room temperature to cryogenic temperatures and also permit lower cycle frequencies leading to a lower running power cost.

We conclude this chapter with the tagline *'Compression is comprehension'* [55]. We established the basics of resource theory of computation. We direct our focus on description complexity and dedicate a significant portion of this chapter to understanding the formal definition, the significance and the limitations of description complexity as a computational resource. We perform a review of algorithmic information theory and conclude the review with a table that aggregates and summarizes different compound computational resource measures. Finally, we explore the ways to estimate description complexity and motivate the adoption of compression as an estimate for description complexity. Quantum algorithms are hard to understand, with quantum mechanics being notoriously counter-intuitive; studying the compressed forms of algorithms can help us understand the algorithmic structure and eventually prove useful for establishing the quantum advantage.

3

Unitary Sequences: Description for Quantum Computation

Brevity is the soul of wit.
- William Shakespeare

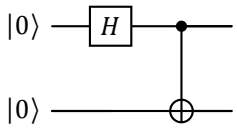
This chapter will delve into the theoretical underpinnings of quantum description, different ways of quantifying quantum description and inter-converting between them. Description complexity, in its classical form, measures the amount of information needed to describe or specify an object or sequence, typically represented by a binary string. In the quantum context, where quantum bits or qubits are considered, Quantum Description Complexity aims to quantify the information content of quantum states or processes. Following the analysis of quantum description and it's different formulations, the trade-offs between quantum circuit complexity and description complexity will be presented.

3.1. Quantum Description

A quantum description serves as a model for encapsulating information regarding quantum computing processes, encompassing quantum states, gates, and operations. It's important to tailor the representation to suit the task, hardware, or theoretical framework. A handful of different descriptions for quantum computation are listed below, and we follow a common example of the creation of the bell state $|\Phi_+\rangle$ as the process that we are trying to describe.

$$|\Phi_+\rangle = \frac{1}{\sqrt{2}}[|00\rangle + |11\rangle] \tag{3.1}$$

1. **Quantum Circuit**
   Quantum circuits are popularly adopted for expressing the quantum description. It starts with formulating a problem, such as integer factorization, search problem, or some cryptography protocol. After the problem formulation, a quantum circuit is designed by using a specific set of operations and instructions that can be performed in a step-by-step manner on a quantum computer. Quantum gates, serving as basic building blocks, are utilized for specific operations on qubits.



2. **Unitary Matrix** A unitary matrix represents the transformation of a quantum state in a closed quantum system. For an n-qubit system, they can be represented as a square matrix with size

2^n having complex entries U and obeying $UU^\dagger = I$. Unitary transformations are reversible and conserve the sum of probabilities in a quantum system. They provide mathematically elegant and computationally efficient descriptions of a quantum system. The unitary transformation can also be parameterized in terms of a Hamiltonian of evolution, which can be regarded as a separate quantum description along with the time of evolution.

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \end{bmatrix} \tag{3.2}$$

3. **QASM [56, 57]**
QASM, or Quantum Assembly Language, serves as a quantum programming language. This code serves as a textual representation of quantum circuits, allowing for a concise and human and machine-readable depiction of the quantum computation process, making it a propitious choice for quantum description.

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[2];
h q0[0];
cx q0[0],q0[1];

```

Listing 3.1: OPENQASM code for Bell-state creation circuit

4. **native-mapped QASM [58]** The designed quantum algorithm or written QASM code can seldom be directly implemented on an actual quantum backend. Factors such as connectivity constraints and unique executable gate sets contribute to the inability to directly implement quantum algorithms or QASM instructions on specific hardware platforms. The process also includes optimization steps to reduce the depth overhead to account for gate inaccuracies and limited coherence times. Therefore, Native-mapped QASM is designed to be compatible or mapped directly to the native instructions or operations of any particular quantum processor. For example, choosing the 127-qubit processor `ibm_sherbrooke` as the backend, which has the native gates `['id', 'rz', 'sx', 'x', 'ecr', 'reset']`. The same Bell-state preparation circuit can be tranpiled into native-mapped QASM instructions:

```
OPENQASM 2.0;
include "qelib1.inc";
gate rzx(param0) q0,q1 { h q1; cx q0,q1; rz(-pi/4) q1; cx q0,q1; h q1; }
gate rzx(param0) q0,q1 { h q1; cx q0,q1; rz(pi/4) q1; cx q0,q1; h q1; }
gate ecr q0,q1 { rzx(pi/4) q0,q1; x q0; rzx(-pi/4) q0,q1; }
qreg q[127];
sx q[0];
rz(-pi/2) q[1];
sx q[1];
ecr q[1],q[0];
rz(pi/2) q[0];
sx q[0];
rz(pi/2) q[0];
rz(pi/2) q[1];
sx q[1];
rz(pi/2) q[1];
```

Listing 3.2: Native-mapped QASM for Bell-state creation circuit on ibm_sherbrooke

5. **QISA microcode** The quantum instruction set architecture(QISA) microcode represents a layer of abstraction closer to the physical operation of a quantum processor. It is constructed at the architecture level, which defines how the quantum gates or operations are implemented on a processor. Going down one level brings us down to the control layer, which can contain the instructions in the form of timed microwave pulses tuned for executing desired operations on superconducting platforms. Alternatively, the instructions can be in the form of magnetic fields or laser pulses used to control trapped-ion and trapped-atom systems. Going back to our running example of Bell-state preparation, the microcode or pulse-level schedule of instructions for implementing the same circuit on the `ibm_sherbrooke` processor is:

```
Schedule(
    (0, ShiftPhase(1.57, DriveChannel(1))),
    (0, ShiftPhase(1.57, ControlChannel(0))),
    (0, ShiftPhase(1.57, ControlChannel(4))),
    (0, Play(Drag(duration=120, amp=(0.0932+0.0013j), sigma=30, beta=-0.0571, name='X90p_d0'), DriveChannel(0), name='X90p_d0')),
    (0, Play(Drag(duration=120, amp=(0.0955+0.00038j), sigma=30, beta=-0.00989, name='X90p_d1'), DriveChannel(1), name='X90p_d1')),
    (120, Play(GaussianSquare(duration=600, amp=(0.0332+0.00057j), sigma=32, width=472, name='CR90p_d0_u2'), DriveChannel(0), name='CR90p_d0_u2')),
    (120, Play(GaussianSquare(duration=600, amp=(-0.1263-0.0271j), sigma=32, width=472, name='CR90p_u2'), ControlChannel(2), name='CR90p_u2')),
    (720, Play(Drag(duration=120, amp=(0.1908+0j), sigma=30, beta=-0.0158, name='Xp_d1'), DriveChannel(1), name='Xp_d1')),
    (840, Play(GaussianSquare(duration=600, amp=(-0.0332-0.00057j), sigma=32, width=472, name='CR90m_d0_u2'), DriveChannel(0), name='CR90m_d0_u2')),
    (840, Play(GaussianSquare(duration=600, amp=(0.1263+0.0271j), sigma=32, width=472, name='CR90m_u2'), ControlChannel(2), name='CR90m_u2')),
    (1440, ShiftPhase(-1.57, DriveChannel(0))),
    (1440, ShiftPhase(-1.57, DriveChannel(1))),
    (1440, ShiftPhase(-1.57, ControlChannel(0))),
    (1440, ShiftPhase(-1.57, ControlChannel(2))),
    (1440, ShiftPhase(-1.57, ControlChannel(30))),
    (1440, ShiftPhase(-1.57, ControlChannel(4))),
    (1440, Play(Drag(duration=120, amp=(0.0932+0.0013j), sigma=30, beta=-0.0571, name='X90p_d0'), DriveChannel(0), name='X90p_d0')),
    (1440, Play(Drag(duration=120, amp=(0.0955+0.00038j), sigma=30, beta=-0.00989, name='X90p_d1'), DriveChannel(1), name='X90p_d1')),
    (1560, ShiftPhase(-1.57, DriveChannel(0))),
    (1560, ShiftPhase(-1.57, DriveChannel(1))),
    (1560, ShiftPhase(-1.57, ControlChannel(0))),
    (1560, ShiftPhase(-1.57, ControlChannel(2))),
    (1560, ShiftPhase(-1.57, ControlChannel(30))),
    (1560, ShiftPhase(-1.57, ControlChannel(4))),
    name="circuit-14"
)
```

Listing 3.3: Pulse schedule for bell-state creation circuit on ibm_sherbrooke

These varied choices underscore the flexibility in choosing a particular technique and don't affect the final description complexity as the choice of description language only leads to a constant factor difference. This constant factor equivalence is, however, subjected to no additional restrictions in defining the description languages. For example, the unitary vs the QASM representation should not be limited by the number of digits in entries of the matrix or the number and type of distinct allowed gates in the circuit. However, it is essential to note that this near independence of the choice of description is justified at the level of ensembles of systems and not at the level of an individual quantum system. The description complexity of a specific quantum system can be genuinely simplistic in one description while being highly convoluted in an alternate description.

3.1.1. Estimating Quantum Description via Circuit Compression

As described in Chapter 2 of the thesis, description or algorithmic complexity approximates an object's information content or randomness. Therefore, compression serves as a valid candidate for estimating the description complexity. Akin to simplifying classical logic circuits or compressing classical information, compression of quantum circuits offers a highly concise and efficient description of the quantum process or algorithm. The investigation into the compression of quantum circuits offers numerous valuable insights and applications, to name a few:

1. Gate Simplification and Circuit rewriting: This approach for circuit compression targets to redefine gates or sequences of gates to remove redundancy or to discover patterns of usage in quantum circuits. Such rewriting schemes contribute to the search for new and more efficient quantum instruction set architectures.

2. Approximate Compilation: Approximate representation of quantum circuits in scenarios with an admissible error rate can prove to be highly simplistic or compressible and offer a higher implementation efficiency in terms of execution time or energy cost.

3.2. Unitary Decomposition

The unitary matrix for a quantum system captures the information of the continuous evolution of the input state to the output state in a succinct and mathematically rigorous manner. The choice of the unitary as a quantum description is universal and hardware-agnostic. The process of decomposition of a unitary matrix into a sequence of quantum gates or operations is crucial in translating an abstract quantum algorithm/evolution into a practical implementation that can be executed on a quantum computer.

3.2.1. Why we target this level: The Continuous-Discrete Divide

Performing a decomposition of the unitary matrix into a discrete basis of gates and synthesizing it into a quantum circuit takes the description from a continuous ($SU(n)$) space to a discrete one. The choice of this level as the target for quantum description is motivated by this continuous-discrete divide. It facilitates the search for optimization of the circuit synthesis process by offering flexibility in selecting a discrete basis of universal gates and setting an arbitrary accuracy of decomposition.

The initial set of experiments in the thesis begins with the study of unitary matrices U for 1-qubit systems sampled at random from the Harr measure and performing the decomposition into quantum circuits. Such a set of random unitary matrices is distributed uniformly in the space of 2×2 unitary matrices, namely the $SU(2)$ space, and is widely preferred in quantum computation and information theory experiments. The unitary matrices were prepared using `qiskit.quantum_info.random_unitary` function that returns a random unitary matrix with the selected dimensions. After establishing the generalized decomposition routine in later sections of this chapter, we extend our model to multi-qubit systems.

3.3. YAQQ: Experiments with Random and Solovay-Kitaev Decomposition

One of the main visions behind this thesis project is to investigate the design and search for optimal quantum architectures. Conventional quantum architecture schemes are strongly dependent on the design of the quantum hardware platform, the connectivity constraints, and the native gate-sets. Yet Another Quantum Quantizer (YAQQ) [59] is a software tool that functions as a design space exploration of the choice of discrete basis of gates and the decomposition routines of unitary transformations. It offers functionalities for:

1. Performing the decomposition of unitary matrices into quantum circuits with user-defined gate-set and decomposition methods.

2. Search for optimal gate-set for a given data set of unitary matrices by searching for convergent solutions of a cost function with parameters such as process fidelity, circuit depth, and novelty.

3. Comparison of the performance of different gate sets and visualization of performed decompositions on the Bloch sphere and Weyl chamber.

Translating from the higher abstract level of quantum algorithms expressed in terms of unitary matrices to the level of sequences of quantum operations/gates involves the process of decomposition. Decomposition routines constitute an essential component of the quantum compiling process as they synthesize the quantum circuits in terms of the native gates and the form of quantum instruction set architecture that can be parsed by the control architecture. Decomposition techniques can be exact or approximate in nature.

Approximate decomposition offers the ability to resolve a unitary U into a quantum circuit using a finite discrete set of basis gates. The Solovay-Kitaev Theorem [7] formulated independently by Robert M. Solovay(1995) and Alexei Kitaev(1997) is one of the most celebrated results in quantum computation. In single-qubit systems, the theorem states that if we have a set of quantum gates that densely span the space of $SU(2)$, then it is possible to approximate any quantum operation by a sequence of gates from the set. The following subsection would focus more deeply on the Solovay-Kitaev Algorithm.

Another approximate decomposition technique implemented in the YAQQ package is the Random decomposition. It forms random sample sequences of arbitrary lengths from a set of gates and chooses the sequence with the highest fidelity among multiple trials.

3.3.1. Preliminary Experiments

For these experiments, we build a dataset of 10,000 random single-qubit Harr-random unitaries. We subject these unitaries to random and Solovay-Kitave decomposition with the chosen gateset `[h, t, tdg]`. We undertake these experiments with the goal of studying the general behavior in terms of decomposed circuit depth and process fidelity of these random unitaries under both types of decomposition. We visualize these sample random unitaries on the Bloch sphere with a colormap depicting the circuit depth, process fidelity, and compression ratio (described in the subsequent paragraph). These experiments serve the dual purpose of both testing the decomposition routines in the framework YAQQ and studying the data set of random unitaries and choosing an appropriate decomposition method.

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad T(\pi/8) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}; \quad T^{\dagger} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix}$$

Following the decomposition of the unitaries in the data set with the two techniques, Random and Solovay-Kitaev, the QASM code of the quantum circuit was cast into a string and compressed using the open-source `bz2` package in Python. bzip2 [26] is a lossless compression technique that uses the Burrows-Wheeler transform [60] and Huffman Coding [61]. For this phase of experiments, bzip2 serves as the estimate for the description complexity of the quantum circuits. A comparative study was undertaken between the two decomposition routines to make the best choice from an information theoretic perspective using the parameters circuit depth, fidelity, and compression ratio.

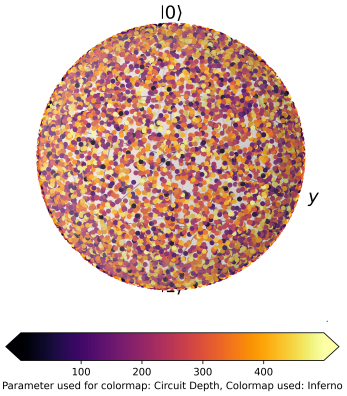


Figure 3.1: (a)

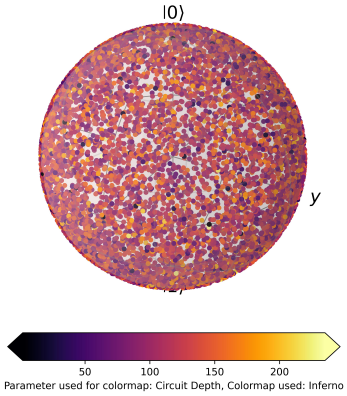


Figure 3.2: (b)

Figure 3.3: Circuit Depth after decomposition depicted using a colormap on the Bloch Sphere for the data set of Harr-random unitaries. (a) Random Decomposition, (b) Solovay-Kitaev Decomposition

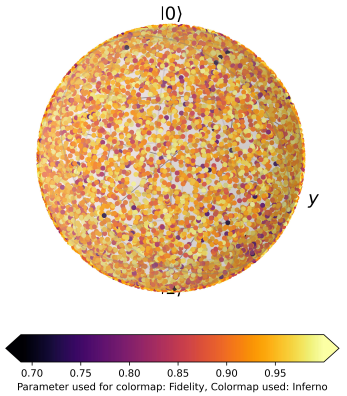


Figure 3.4: (a)

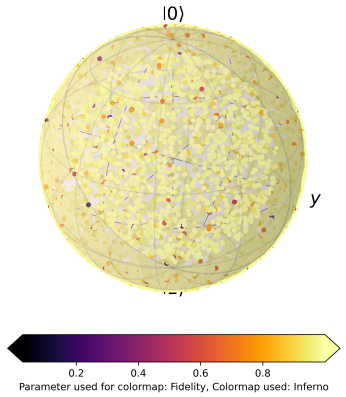


Figure 3.5: (b)

Figure 3.6: Process Fidelity for decomposition depicted using colormap on the Bloch sphere for the data set of Harr-random unitaries. (a) Random Decomposition, (b) Solovay-Kitaev Decomposition

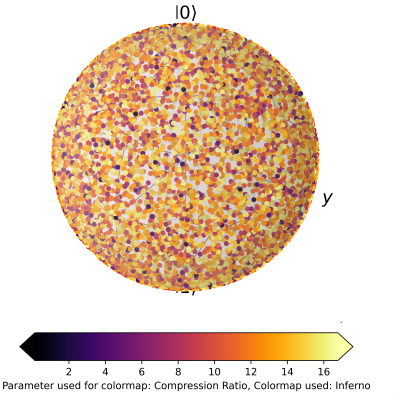


Figure 3.7: (a)

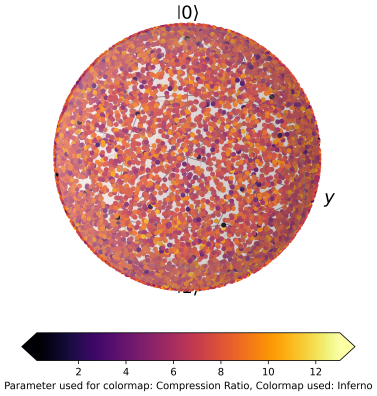
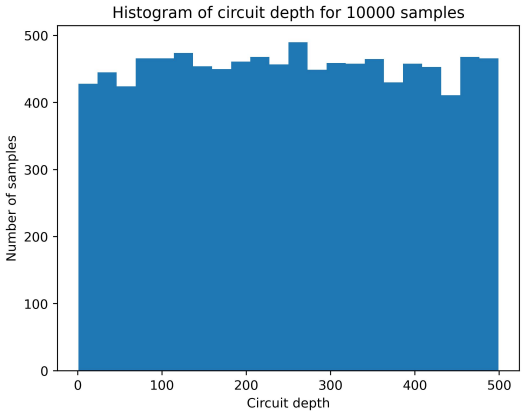


Figure 3.8: (b)

Figure 3.9: Compression Ratio = len(original QASM)/len(compressed QASM) for decomposed quantum circuits depicted using colormap on the Bloch sphere for the data set of Harr-random unitaries. Higher compression ratio value $\implies$ better compression (a) Random Decomposition, (b) Solovay-Kitaev Decomposition



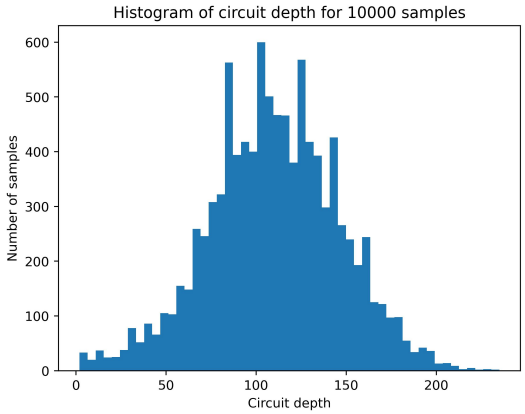
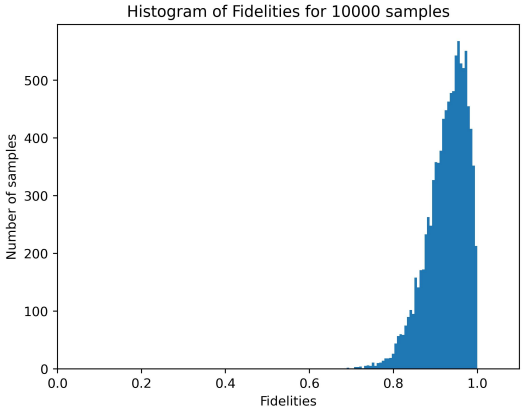


Figure 3.10: Histograms of Circuit Depth after decomposition for the data set of Harr-random unitaries. (a) Random Decomposition and (b) Solovay-Kitaev Decomposition



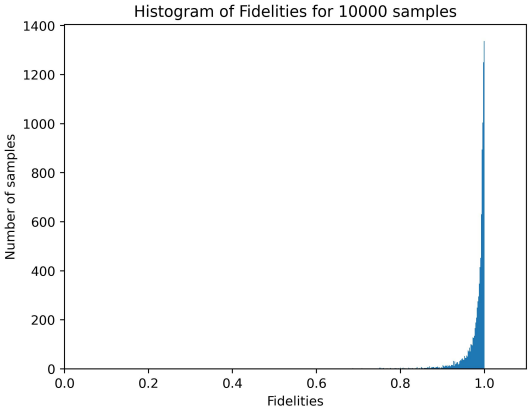


Figure 3.11: Histograms of Process Fidelity of Decomposition of samples from the data-set of Harr-random unitaries. (a) Random Decomposition and (b) Solovay-Kitaev Decomposition

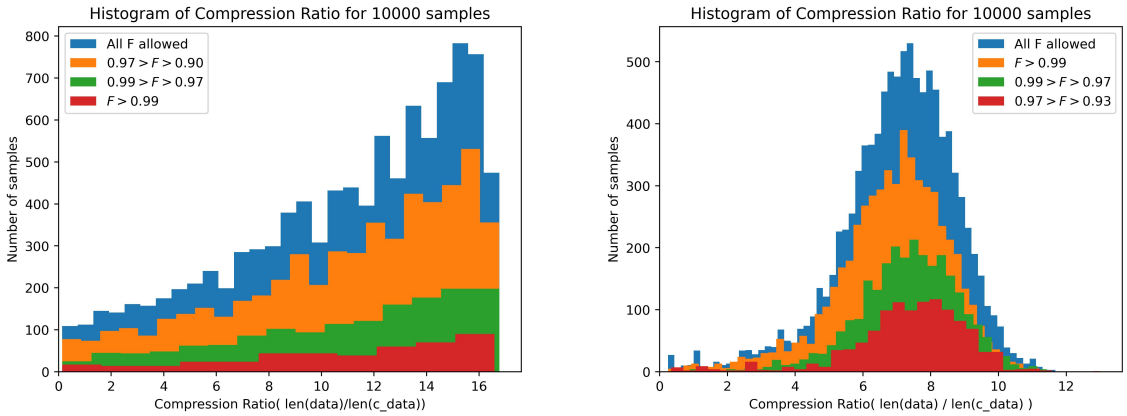


Figure 3.12: Histograms of Compression Ratio = len(original QASM)/len(compressed QASM) for decomposed quantum circuits of the data set of Harr-random unitaries with bands of allowed fidelities represented by different colors (a) Random Decomposition and (b) Solovay-Kitaev Decomposition

3.3.2. Observations and Conclusions from Above Results

- In the case of Random decomposition, the distribution of samples is generally uniform across different circuit depths of decomposed quantum circuits implying that unitaries are equally likely to have short quantum circuit expressions as extended expressions. This outcome is based on the design of the method in YAQQ. Such behavior contradicts the general insight from information theory that most objects in nature have complex descriptions. Extending this logic to the case of computing, most unitaries can be expected to possess long quantum-circuit descriptions (or a high quantum-circuit complexity).

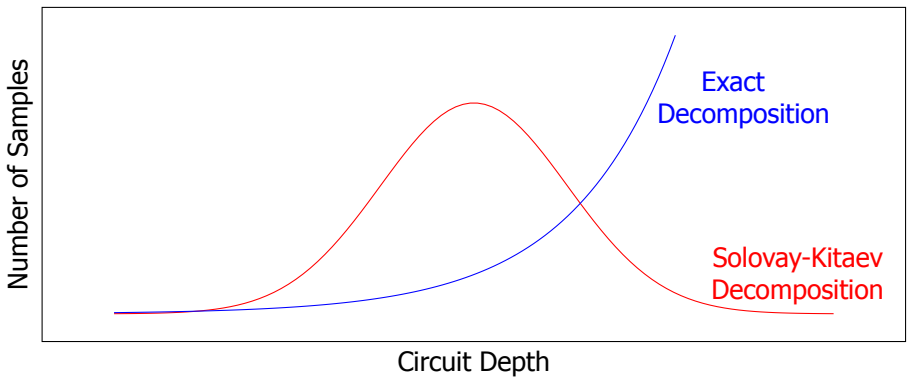


Figure 3.13: (a)Expected Curve for Circuit Depth(for Exact Decomposition) (b)Observed Curve for Circuit Depth(for Solovay-Kitaev Decomposition)

In the case of Solovay-Kitaev Decomposition, we see this rise in the number of samples with higher circuit depth of decomposition; however, the drop in the number of samples with higher circuit depths is due to the approximate nature of decomposition. The decomposition algorithm is designed to generate finite sequences of gates from a set that approximates a quantum operator with a bounded error. Therefore, the Solovay-Kitaev Algorithm is the better choice for an information-theoretic study, which conforms with the goal of this thesis. Additionally, the much higher fraction of samples with higher process fidelity for the Solovay-Kitaev Decomposition is another plus point.

- In most popular studies the "compression factor" which is defined as the ratio of the size of compressed data to the size of original data is used for compression techniques and routines. This parameter would be the inverse of the compression ratio plotted in the preceding figures and will vary in the range 0 to 1 with lower values indicating better compression. Here are the comparison plots for the compression factor corresponding to fig 3.12

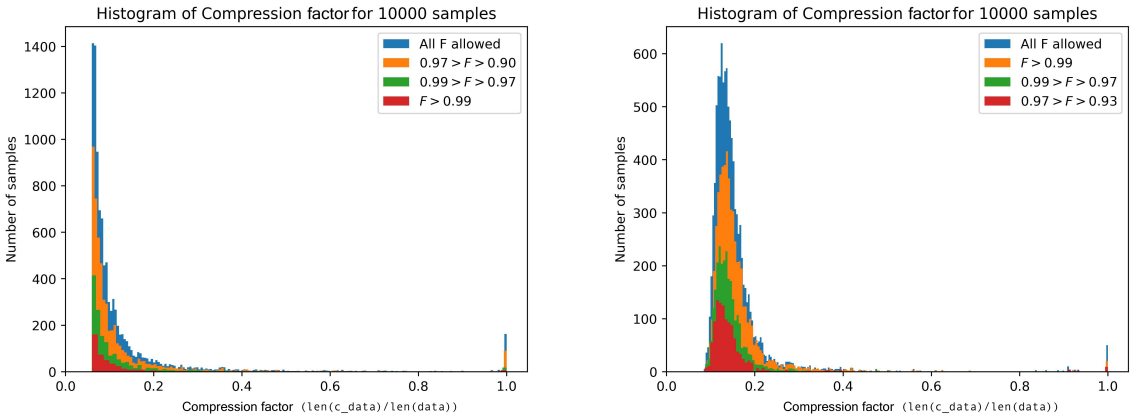


Figure 3.14: Histograms of Compression Ratio = len(original QASM)/len(compressed QASM) for decomposed quantum circuits of the data set of Harr-random unitaries with bands of allowed fidelities represented by different colors (a) Random Decomposition and (b) Solovay-Kitaev Decomposition

The rest of the thesis will consider the compression factor when assessing compression techniques. In the above plots, we can observe a small number of samples at compression factor = 1.0. This is due to the phenomenon of the pigeonhole principle, which explains the non-existence of a universal lossless compression algorithm that can compress all possible data. In the cases when the data has a low entropy or description complexity, the extra bookkeeping needed to encode the data using lossless compression methods eventually ends up increasing the file size.

Following the remark in the first observation on the expected circuit complexity trends, a majority of the samples from the data set of unitaries are predicted to have a low compression ratio or high compression factor. An object having a higher description complexity or randomness has a lower compressibility.

On this aspect, both decomposition techniques offer excellent compression for a majority of samples. It is essential, however, to note that compression with bzip2 doesn't capture the "quantumness" of the circuits and interprets the QASM strings simply as ASCII text and searches for statistical patterns. It's imperative to recognize patterns and relationships among the sequences of quantum gates to develop a meaningful estimation of quantum description complexity. This point will be taken care of, while formulating optimal coding techniques for devising an estimation of quantum description complexity in Chapter 4

The following sections in this chapter will introduce and elaborate on the decomposition techniques that will be implemented in integration with the coding techniques in the next chapter.

3.4. Solovay-Kitaev Decomposition

The Solovay-Kitaev decomposition algorithm presented by Christopher M. Dawson and Michael A. Nielsen in [62] in 2006 is one of the best reviews and applications of the Solovay-Kitaev Theorem for decomposition of quantum operators into a sequence of quantum gates. The algorithm finds approximate sequences of length $O(\log^{3.97}(1/\epsilon))$ within a time $O(\log^{2.71}(1/\epsilon))$ both characterized by the precision ϵ. In it's implementation, it includes a preprocessing step that generates a search space of composite sequences up to a length l_0 of gates that belong to a finite discrete basis. This search space is referred to as the Solovay-Kitaev (S-K) basis in the remainder of the thesis. The maximum length of sequences l_0 is referred to as the depth d of the S-K basis. The chosen set of fundamental gates and the group generated by the set, that is, the S-K basis, must fulfill the following conditions generalized for an m-qubit system:

1. All the gates in the set belong to the group of special unitary matrices $SU(m)$ and have a determinant 1
2. The set of gates is closed under inversion, implying that for every gate in the set, its hermitian conjugate must also belong to the set

3. The group that is generated by the set must densely span the space $SU(m)$. This means that, for every arbitrary unitary operation U, there must exist a product sequence of gates from the set that can approximate U with a bounded error ϵ

The Solovay-Kitaev algorithm exhibits a refined design with apparent conciseness and succinctness, yet its practical performance entails intricate complexities. The goal of devising a methodology for reconstructing the original circuit from base-case approximation drives us to probe into the algorithm's intricate operation. This goal was achieved by a rigorous examination of the algorithm across various recursion depths and keeping track of the re-arrangement of sequences at each level of progression. As a first step to this undertaking, a flowchart [3.15] has been created after careful study of the pseudocode of the algorithm presented in [62] and its implementation on `qiskit`.

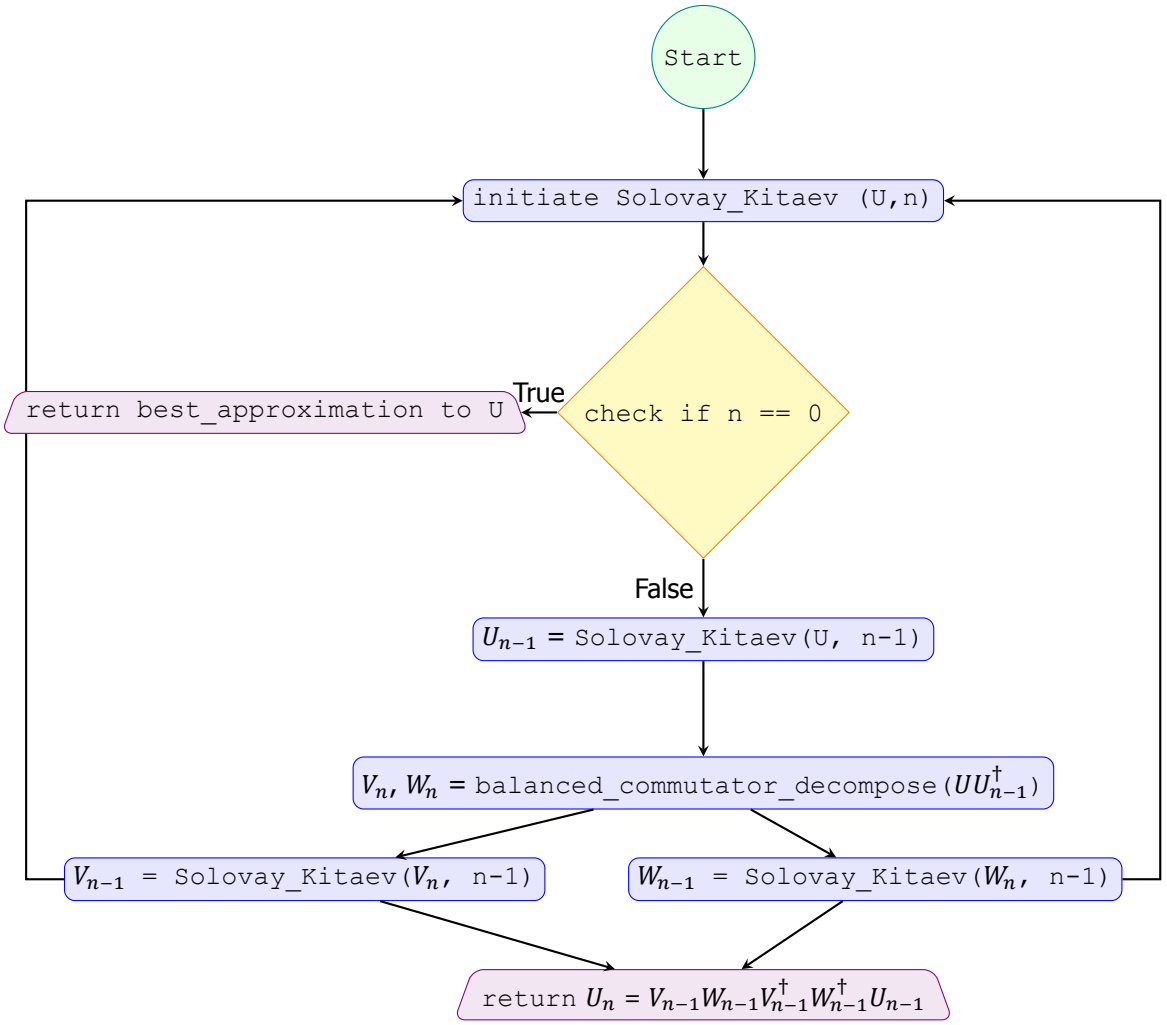


Figure 3.15: Flowchart of the Solovay-Kitaev Decomposition Algorithm for 1-qubit unitary quantum operator U and recursion depth n. Returns the ϵ_n approximation to the target unitary U computed from call of the function at the $n-1$ degree of recursion, and returns the ϵ_0 approximation in the base case

3.4.1. The Algorithm

The algorithm functions in a recursive fashion, and the degree of recursion is denoted by n in the flowchart 3.15. In the course of the process, the algorithm returns a sequence that approximates the unitary operator U up to an error ϵ_n. The approximation error at each recursion level r is related to that at the level $(r-1)$. This goes on till the base case that returns the `best_approximation` to a matrix U, that is bounded by ϵ_0.

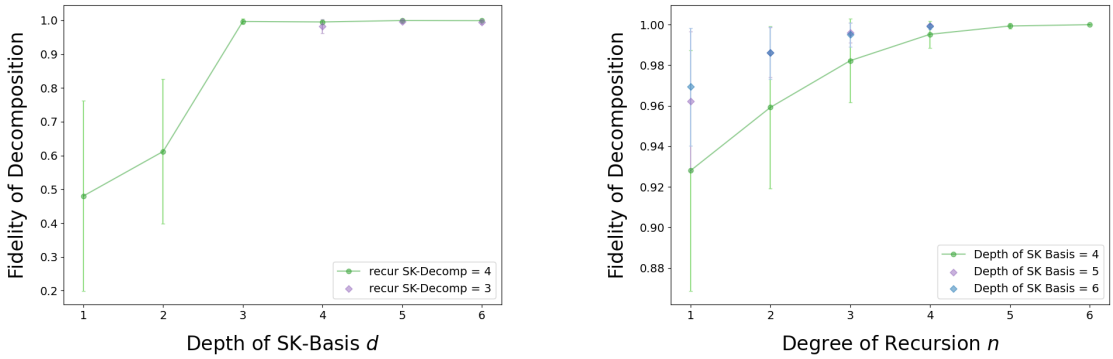
The algorithm is designed to obtain an improved approximation accuracy $\epsilon_r < \epsilon_{r-1}$. The approximation accuracy tends to reach 0 as the recursion depth n increases indefinitely. The `bal-`

`anced_commutator_decompose` method performs a balanced group commutator decomposition of the accuracy at level r defined as $\Delta = UU_{r-1}^{\dagger} = VWV^{\dagger}W^{\dagger}$ for matrices V and W. $(r-1)$ level approximation accuracies are computed by the call of the function again for matrices V and W and the r^{th} level approximate sequence $U_r = V_{r-1}W_{r-1}V_{r-1}^{\dagger}W_{r-1}^{\dagger}U_{r-1}$, consisting of all 5 terms computed form the $(r-1)^{\text{th}}$ level is returned.

3.4.2. Performance Analysis of Decomposition Process

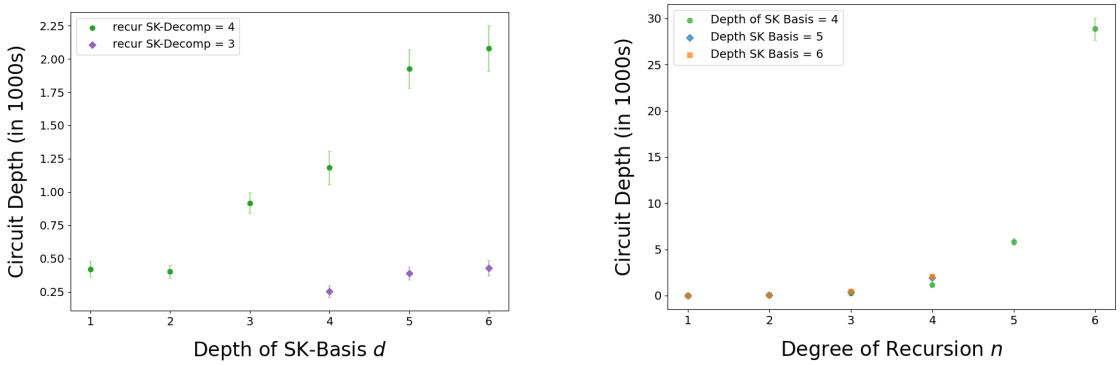
This subsection is dedicated to evaluating the performance of the Solovay-Kitaev decomposition algorithm for various configurations of depth d and degree of recursion n. This empirical analysis of performance is undertaken based on the process fidelity of decomposition and the circuit depth of the decomposed quantum circuits. The process fidelity of the decomposition process is computed using the `qiskit.quantum_info.process_fidelity` function. It computes the state fidelity between two quantum channels that are supplied as normalized Choi matrix representations of the target unitary operator and its decomposed quantum circuit.

In the subsequent plots, the average trends of process fidelity and circuit depth are depicted against varying depths of the SK basis while maintaining a fixed degree of recursion, and similarly, against varying degrees of recursion while holding the depth of the SK basis constant.



(a) (b)

Figure 3.16: Average trend of fidelity of Solovay-Kitaev decomposition with varying (a) depth of SK-basis d and (b) degree of recursion n



(a) (b)

Figure 3.17: Average trend of circuit depth of Solovay-Kitaev decomposition with varying (a) depth of SK-basis d and (b) degree of recursion n

The first two plots in Fig. 3.16 depict the average fidelity of decomposition of single qubit Harr-

random unitaries over varying depth d and degree of recursion n for SKD. Generally speaking, an increase in fidelity and circuit depth is expected for increasing d and n. The impact of change in these parameters, however is different on the overall decomposition.

Increasing the depth of SKD increases the maximum length of the gate sequences in the SK basis, and in turn increases how densely the sequences in the basis span the total space of unitaries. This improves the ϵ_0 precision, at the base layer of the algorithm leading to a more consistent approximation as we will see the upcoming subsection. An interesting inference on this note can be observed from the trends of fidelity and circuit depth of decomposition vs d. At $d = 3$, we observe a saturation of fidelity close to 1; this indicates the universality of the SK-basis of $d = 3$ in mapping the space, and this set forms a subset of basis sets of all higher depths. With increasing d, the length of gate sequences in the search space gets longer, leading to an increase in circuit depth.

The degree of recursion of SKD specifies the number of recursion steps undertaken by the algorithm for approximating the target unitaries, with the precision of the approximation increasing with each level of recursion. The circuit depth increases exponentially with increasing degree of recursion, while the fidelity follows an exponential saturation.

3.4.3. Structuring of Sequences in Decomposed Circuits and Reconstruction

An intuitive tree diagram depicting the working of the algorithm is presented in fig. 3.18. The bottom-most layer consists of 3^n blocks and corresponds to the base case of the algorithm. For each block, the `best_approximation` method finds the sequence of gates from the search space that approximates it up to ϵ_0. Each node in the level r, U_r is a composite sequence constructed from its three daughter nodes in the $r-1$ level such that $U_r = V_{r-1}W_{r-1}V_{r-1}^{\dagger}W_{r-1}^{\dagger}U_{r-1}$.

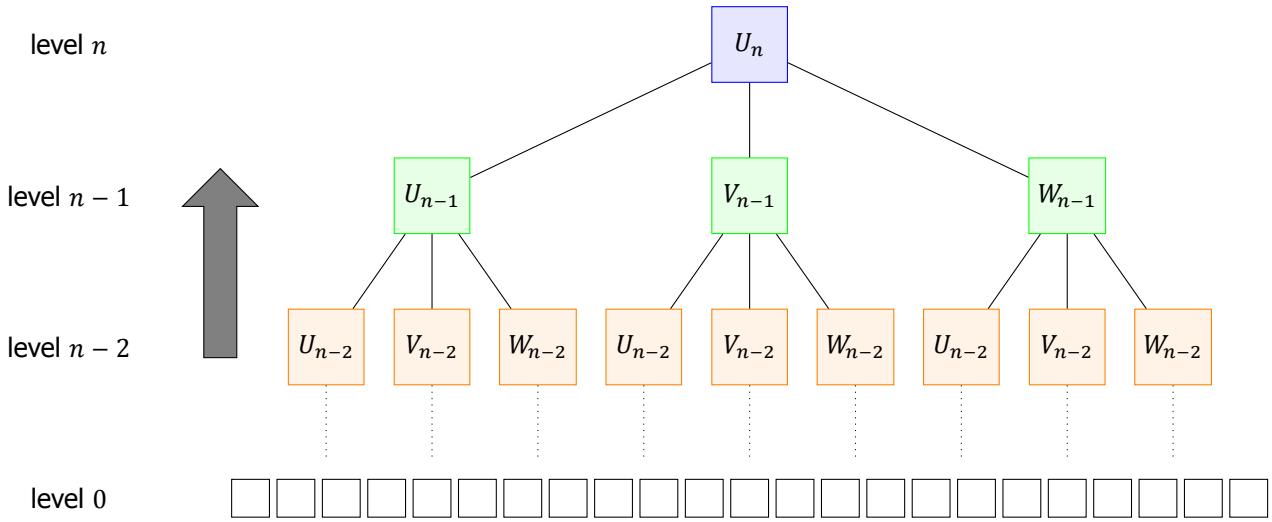


Figure 3.18: Tree of sequences depicting the working of the Solovay-Kitaev algorithm. Recursion levels are indicated on the left.

This tree diagram is a crucial contribution of this thesis as it explains the hierarchical re-arrangement of approximate gate sequences. To reconstruct the decomposed quantum circuit from its bottom-most layer of gate sequences, it is imperative to traverse the tree diagram in a bottom-up manner. As mentioned earlier, the bottom-most layer is constituted of gate sequences from the search space for the `best_approximation` function. The implementation of the decomposition algorithm on `qiskit` includes the construction of this basis for a specified depth d. In this project, qiskit's implementation of the Solovay-Kitaev algorithm has been used as the base to build up the encoded quantum instruction set model. The decomposition and subsequent reconstruction process can be elucidated through the following example. Consider a Harr-random matrix denoted as U prior to undergoing decomposition in terms of the gate set `[h, t, tdg]`:

$$U = \begin{bmatrix} 0.50359966 + 0.62609046j & -0.07233711 + 0.59090224j \\ 0.31138773 + 0.50738132j & 0.19782201 - 0.77876077j \end{bmatrix} \tag{3.3}$$

The resulting decomposed quantum circuit, achieved via Solovay-Kitaev decomposition with a recursion degree $n = 2$ and depth $d = 3$, is depicted below:

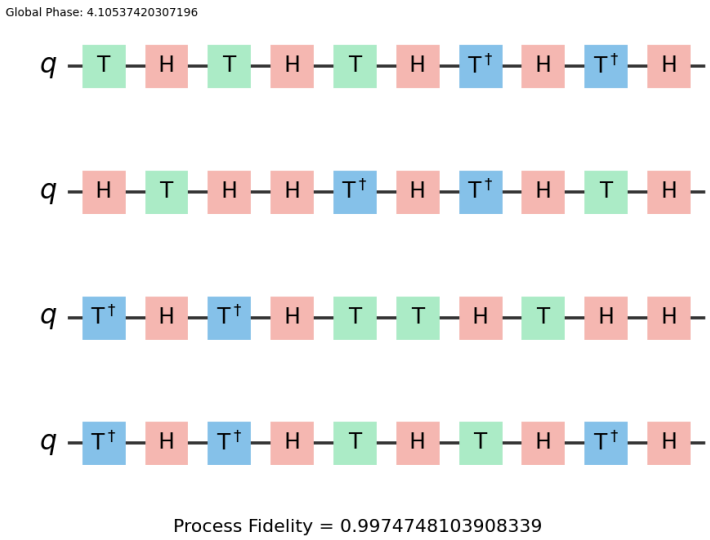


Figure 3.19: Decomposed circuit. Process fidelity is the fidelity of the decomposed circuit from the target unitary operation.

The gate sequences returned by the `best_approximation` from the SK basis at the base layer can be stored as a list. These sequences for the above case of decomposition are listed below as printed console output:

```
best_approximation gate sequence S1:  ['t', 'h']
best_approximation gate sequence S2:  ['h', 'tdg', 'h']
best_approximation gate sequence S3:  ['tdg']
best_approximation gate sequence S4:  []
best_approximation gate sequence S5:  ['h', 'tdg', 'h']
best_approximation gate sequence S6:  ['t']
best_approximation gate sequence S7:  ['t']
best_approximation gate sequence S8:  []
best_approximation gate sequence S9:  ['h', 'tdg', 'h']
```

We can observe that sometimes the empty sequence `[]` is the best approximation. These instructions are redundant and can be omitted while storing the frequencies of gate sequences. The decomposed circuit 3.19 can be rewritten in terms of above gate sequences S_i such as:

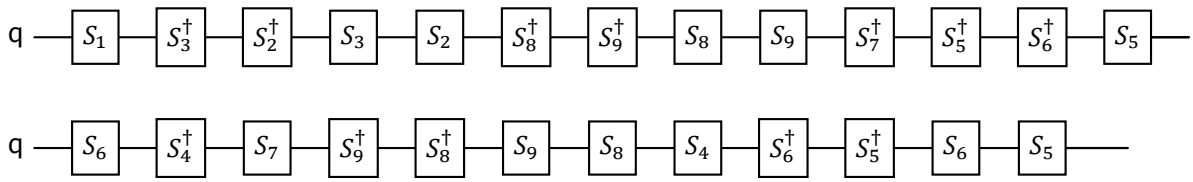


Figure 3.20: Quantum Circuit using gate sequences from the S-K basis as the building blocks.

This completes the reconstruction of the decomposed quantum circuit using the gate sequences from the base layer. The decomposed quantum circuit 3.19 has a depth of 40, is expressed as a stream of 25 instructions in 3.20. A frequency map is created based on how many times which instruction from the SK-basis gets picked in the base layer. This frequency map will be the input for the encoding process for creating the compressed instruction stream. This will be covered in detail in the following chapter.

The paper [62] also presents an extension of the algorithm for single-qubit systems to the general case of m-qubit systems. This extension is possible in theory with the modification in only one step of the algorithm – the balanced commutator group decomposition to an approximate version of itself. This change appears reasonably straightforward, albeit leading to a significant implementation complication. The depth of the S-K basis scales exponentially with the dimension of the system, causing the look-up routine to scale poorly for higher-dimensional systems. Therefore, generalizing SKD to higher dimensions doesn't pay well. This necessitates an exploration of Quantum Shannon Decomposition, an exact decomposition technique covered in the upcoming subsection.

3.5. Quantum Shannon Decomposition

The quantum Shannon decomposition(QSD) was proposed by Shende Bullock and Markov in [63] as a technique for expressing any n-qubit quantum operator as an exact decomposition into single-qubit rotations and 2-qubit controlled gates. The algorithm follows a divide-and-conquer strategy in a recursive fashion and breaks down the n-qubit unitary matrix into smaller submatrices. The algorithm starts with cosine-sine decomposition (CSD), a well-known technique in linear algebra that divides the target matrix U into smaller blocks. The algorithm recursively performs CSD and other decomposition techniques such as eigenvalue decomposition and Euler decomposition to eventually express the original complex operator as a sequence of single-qubit gates and CNOTs that can be passed as an executable stream of instructions for the hardware. This synthesis technique functions as a quantum version of the classical Shannon decomposition of boolean functions.

A quantum operation on n-qubits is represented by a unitary matrix U of dimensions $2^n \times 2^n$. According to CSD, $U = LMR^\dagger$ where L and R are block-diagonal matrices representing uniformly controlled gates and the middle matrix M that represents a controlled R_y rotation on the MSB.

$$U = \left[\begin{array}{c|c} U_{00} & U_{01} \\ \hline U_{10} & U_{11} \end{array}\right] = \left[\begin{array}{c|c} L_1 & 0 \\ \hline 0 & L_2 \end{array}\right] \left[\begin{array}{c|c} C & -S \\ \hline S & C \end{array}\right] \left[\begin{array}{c|c} R_1 & 0 \\ \hline 0 & R_2 \end{array}\right]^\dagger \tag{3.4}$$

L_1, L_2, R_1 and R_2 are unitary matrices of size 2^{n-1}. C and S are diagonal matrices such that $C^2 + S^2 = I$, thereby justifying the name of the decomposition technique. The matrices L and R are termed as quantum multiplexors and they enact L_1 (R_1) or L_2 (R_2) conditioned on the state of the MSB. The middle matrix resembles the R_y rotation matrix that is targetted on the MSB and controlled by the states of the lower-order qubits.

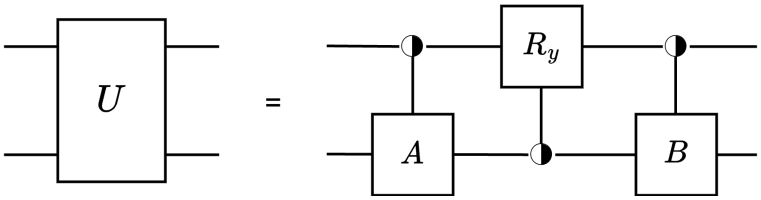


Figure 3.21: The Cosine-Sine decomposition acting on n-qubit gate U. The slash represents a bundle of wires and the box control symbol indicates multiple control wires

The left and right gates undergo a demultiplexing routine that performs an eigenvalue decomposition of the matrices.

$$\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda^\dagger \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & Q \end{bmatrix} \tag{3.5}$$

Where P and Q are unitary matrices, and Λ is a unitary diagonal matrix. The left and right matrices can be represented as quantum gates operating on the lower-order qubits and independent of the MSB. The middle matrix corresponds to a R_z operation on the MSB controlled by the lower qubits.

This process of CSD, followed by subsequent demultiplexing operation, is performed recursively until the algorithm reaches the base case. At the base level, the operator sequence consists of only single qubit gates. At this point, any single-qubit unitary operation is changed into a rotation gate following Euler decomposition. The implementation of the algorithm is presented with two optimization strategies a1 and a2. Following the first strategy, the multiplexed R_y operation in 3.21 is implemented using C-Z gates. In the second strategy, the recursion is stopped at the level of 2-qubit operations, and the resulting circuit is decomposed into CNOT gates and single-qubit rotation gates. With the

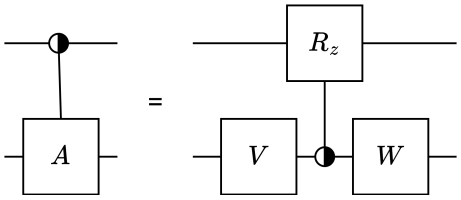


Figure 3.22: Demultiplexing of a multiplexor

optimization strategies, the algorithm yields an efficient synthesis of complicated quantum operators with a minimized number of CNOT gates.

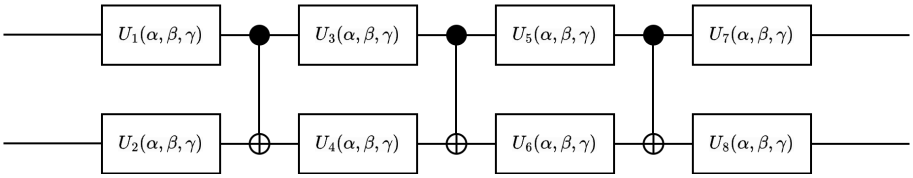


Figure 3.23: Final decomposed form of a 2-qubit arbitrary unitary operator expressed as a sequence of 1-q rotations and CNOT gates.

In this project, the `qiskit` implementation of the QSD algorithm is employed. The QSD process is fixed to run with the a2 optimization as mentioned above, and it returns a decomposed quantum circuit consisting of single qubit unitary rotations and CNOT gates as depicted in fig 3.23 for a 2-qubit system. At this point, we perform SKD on the single-qubit unitary rotations to express them in terms of the native gates(`h, t, tdg`), and leave the `cx` gates untouched. As a result, the multi-qubit random unitary operator is expressed as a quantum circuit built of gates from the discrete gate set `h, t, tdg, cx`. Adopting SKD for breaking down the intermediate gates enables the application of the modified alphabet of instructions from the S-K basis to express the decomposed quantum circuit.

3.5.1. QSD + SKD Performance for 2Q and 3Q Systems

The aim of this subsection is to perform a similar empirical performance analysis of the combined QSD + SKD technique for the decomposition of multi-qubit unitaries, as done for single-qubit systems in section 3.4.2. For the following plots, we select a dataset consisting of 200 Harr-random unitaries each for 2-qubit and 3-qubit systems. Analogous to what we did for 1q systems, the average trends for process fidelity and circuit depth are plotted against the depth of SK basis, keeping the degree of recursion fixed and vice versa.

The goal of these experiments is to observe the performance of the adopted multi-qubit decomposition routine and make a choice of optimal values of parameters d and n. The choice of these parameters derived from averaged runs over ensembles of random circuits would be adopted for employing the decomposition routine for benchmark circuits.

The trends followed by average process fidelity and circuit depth remain mostly similar to the case of 1q systems. The impact of d and n on the decomposition remain largely analogous to the case 1q systems, but they are more accentuated because of multiple instances of single qubit decompositions in 2q and 3q systems.

An example of this accentuation is a lower starting process fidelity at lower degrees of recursion resulting from a build-up of inaccurate single-qubit decompositions at lower recursion levels. The growth of circuit depth over increasing depth d is more moderate in comparison to the exponential growth observed against the degree of recursion.

Focussing on plots 3.24 (b) and 3.25 (b), we observe an improved fidelity of decomposition at depth $d = 5$ as compared to $d = 4$ and $d = 6$. This improvement in the accuracy of decomposition does not incur any significantly higher cost in terms of circuit depth as supported by plots 3.26 (b) and 3.27 (b). Akin to the circuit depth of decomposed circuits, the execution time also scales exponentially with the degree of recursion. Therefore, one has to strike a trade-off between the process fidelity and the execution time of decomposition. Taking into consideration these observations, the depth of SK-basis $d = 5$ and degree of recursion $n = 4$ is decided as the best configuration for multi-qubit decomposition.

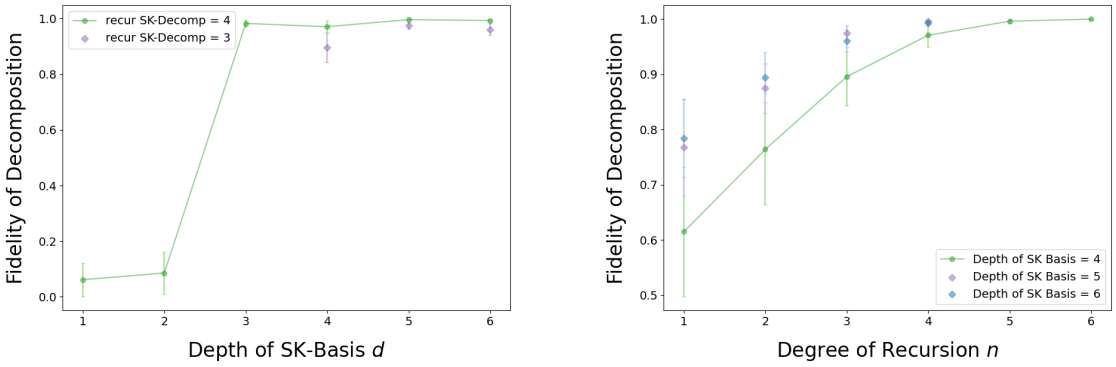


Figure 3.24: Average trends of fidelity for decomposed 2-qubit circuits with QSD+SKD with varying (a) depth of SK-basis d and (b) degree of recursion n

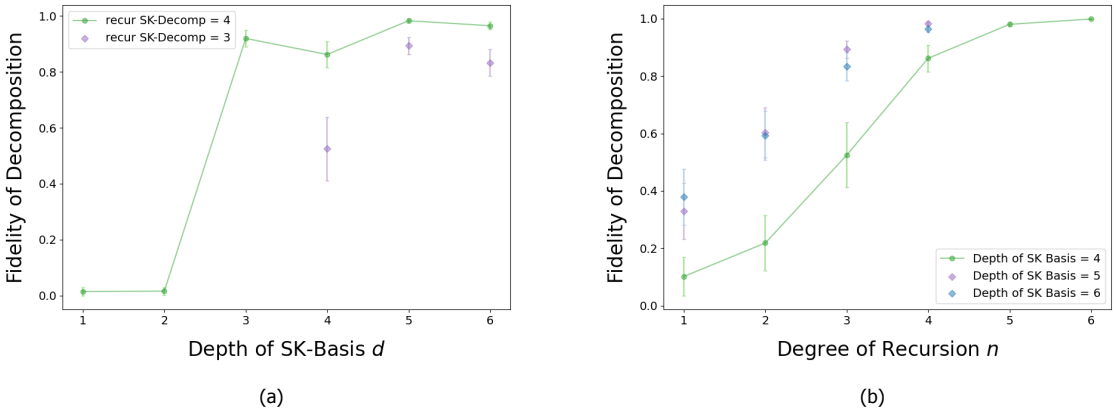


Figure 3.25: Average trends of fidelity for decomposed 3-qubit circuits with QSD + SKD with varying (a) depth of SK-basis d and (b) degree of recursion n

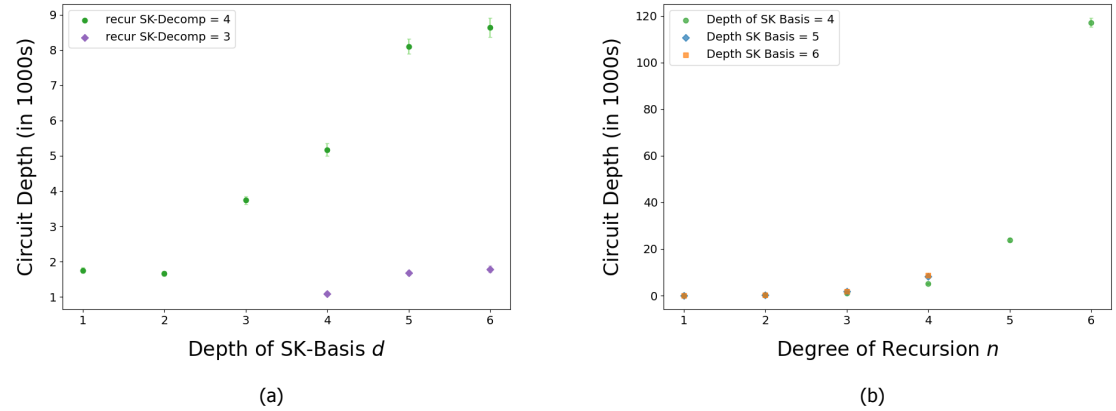


Figure 3.26: Average trends of circuit depth for decomposed 2-qubit circuits with QSD + SKD with varying (a) depth of SK-basis d and (b) degree of recursion n

In this chapter, we have explored the intricacies of quantum description through various representational forms, from algorithms and unitary matrices to assembly languages like QASM. We adopt the

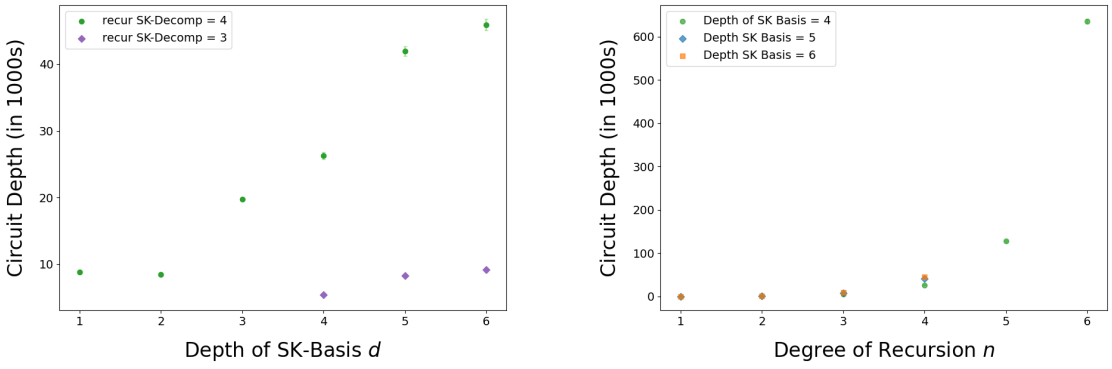


Figure 3.27: Average trends of circuit depth for decomposed 3-qubit circuits with QSD+SKD with varying (a) depth of SK-basis d and (b) degree of recursion n

unitary matrix as the description of a quantum algorithm and approach it from the design space of synthesis into a quantum circuit made up of a discrete set of gates via the experiments with YAQQ. Following this analysis, we dived in great detail into the Solovay-Kitaev decomposition algorithm and established a general decomposition routine for any single-qubit unitary into a user-defined set of native gates. Subsequently, we scaled up this routine to perform the decomposition of multi-qubit unitaries with the incorporation of Quantum Shannon decomposition. We ran experiments on these decomposition routines with varying parameters: depth of SK basis and the degree of recursion to arrive at an optimal choice for multi-qubit systems.

As we move forward, the next chapter will introduce Huffman encoding as a promising technique for compressing these quantum instructions. By applying Huffman encoding to the sequences derived from the Solovay-Kitaev basis, we aim to craft our estimate of quantum description complexity and its application for more efficient quantum control.

4

Huffman Coding for Quantum Instructions

If I had more time, I would have written a shorter letter.
- Blaise Pascal

4.1. Prefix Codes

In the field of information theory, coding is the process of changing the form of information into a different form that is more convenient for performing specific operations, storage, or communication. Reverting it back to the original form is decoding. Codes are popularly used for:

- Data compression
- Error correction
- Encryption

Throughout modern history, codes have made communication more efficient by making messages shorter or more understandable. The first electric telegraph was invented by Samual Morse in 1837, marking a notable achievement in the evolution of long-distance communication. Morse codes are an example of variable-length codes. Variable-length codes are widely used in lossless compression techniques and are essential to entropy encoding techniques. Entropy encoding methods prioritize frequently used symbols or characters with shorter codes while assigning longer codes to less commonly used symbols or characters. This attribute is significant for data compression. Encoding the stream of instructions with variable-length encoding is reasonably straightforward, but the process gets more involved when it comes to decoding it [64]. The choice of encoding must be made with special consideration to make the decoding unique, which is a prerequisite for lossless compression. This special consideration is the prefix property.

'Prefix-free' or simply 'prefix' refers to the property of a set of symbols or strings such that no element of the set is a prefix of another member of the set. Similarly, a prefix code is a coding scheme in which no code word is a prefix of another code word in the set. An example of two sets of code words is illustrated below: Set A is a prefix code while Set B is not.

Set A	Set B
"qubit"	"bit"
"algorithm"	"bitcoin"
"photon"	"algorithm"
"quantum"	"logarithm"

Prefix codes are essential in data compression as they allow efficient, unambiguous decoding, which makes them particularly suitable for compression algorithms. Huffman codes extend this concept by constructing the most efficient compression for a given set of frequencies.

4.2. Huffman Coding

David A. Huffman introduced Huffman coding [61] in 1952 as an optimal prefix code. Given a source stream of symbols, it tabulates prefix codes of variable length for each constituent symbol based on the frequency of occurrence of the symbol. It works by creating a binary tree of nodes called the Huffman tree, where each symbol gets a unique binary code. More frequently occurring symbols are placed closer to the root of the tree and are assigned shorter codes, while those that are less frequent are placed further from the root and are assigned longer codes. Consider the following application of Huffman coding for the encoding and compression of the word 'entanglement'. The first step starts with creating a dictionary of the characters used in the word arranged in descending order of the number of times they are used. The next step for building the Huffman tree involves creating leaf nodes for

character	frequency
e	3
n	3
t	2
a	1
g	1
l	1
m	1

Table 4.1: Character frequency distribution for 'entanglement'

each of these characters corresponding to their frequencies. Two less frequent nodes are combined to form a new node with their frequencies combined. This step is repeated until there is only one node left which will be the root of the Huffman tree.

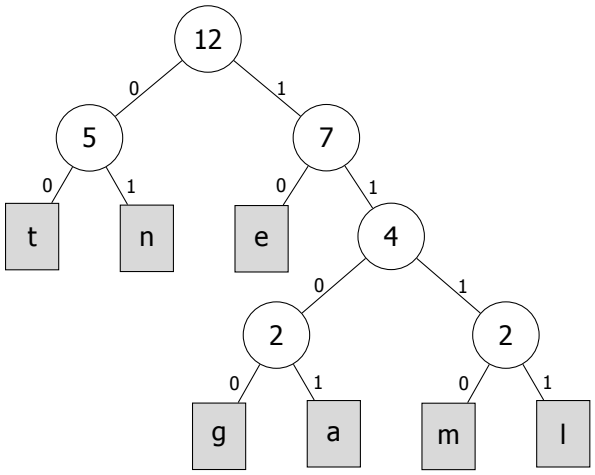


Figure 4.1: Huffman coding tree for 'entanglement' with leaf nodes as gray boxes and binary path labels for encoding.

Huffman Codes are assigned to the characters by traversing along the edges from the root and appending a 0 for a left edge and 1 for a right edge until we reach a leaf.

Character	Huffman Code
'e'	10
'n'	01
't'	00
'a'	1101
'g'	1100
'l'	1111
'm'	1110

Table 4.2: Table with Huffman codes for characters in the word 'entanglement'

Concatenating these, the Huffman encoded string becomes: `10 01 00 1101 01 1100 1111 10 1110 10 01 00` which is 32 bits long. Encoding the word using standard ASCII encoding requires $8 \times 12 = 96$ bits.

Just as lexical constructs(words) are sequentially composed of individual units(characters), so too are quantum circuits systematically constructed through the sequential application of quantum operations. These operations or fundamental units for building the quantum circuit can be quantum gates (as represented in 3.19) or composite gate sequences (as represented in 3.20). This research explores the Huffman coding of the fundamental units in these two representations, leading to different description complexities of the same quantum circuit.

The subsequent analysis is structured into different subsections highlighting the different versions of encoding for quantum examining different fundamental components (building blocks) of quantum circuits. The gate set `[h, t, tdg]` has been selected consistently for all analyses of the thesis.

4.2.1. v0: Binary Encoding

To establish a baseline for encoding efficiency, we implement a uniform binary encoding scheme for quantum circuits. Each gate within the circuit is represented by a fixed-length binary code. The length of each code, denoted as b, is calculated based on the total number of distinct gates in the dictionary:

$$b = \lceil \log_2(N) \rceil$$

where N is the number of distinct gates in the gate-set. This approach ensures each gate is uniquely representable and uses the minimal number of bits required for such representation. For the gate set `[h, t, tdg]`, we need 2 bits per gate. The total information content of the circuit, measured in bits, is the product of b and the total number of gates in the circuit. Following the example of the unitary matrix 3.3 and its decomposed circuit 3.19, which has a depth of 40, that implies that it needs $40 \times 2 = 80$ bits of information. This metric serves as the foundation for comparing the efficiency of Huffman encoding in the subsequent versions, which aims to reduce the overall bit-length of quantum circuit representations.

4.2.2. v1: Huffman Encoding the Gateset

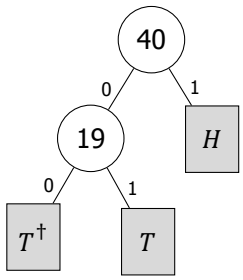
The first version of Huffman coding for the decomposed quantum circuits is based on using the gates in the chosen gate-set as the building blocks of the quantum circuit. The gates are assigned codes based on the number of times they appear in a circuit.

We return to the example of the unitary matrix 3.3 and its decomposed quantum circuit 3.19.

gate	frequency
h	21
t	10
tdg	9

Table 4.3: Frequency distribution of gates in decomposed circuit 3.19

It is straightforward to construct the Huffman tree for the frequency distribution in 4.3:



(a) Huffman Tree representation for encoding quantum gates

```
'Gate'      |     Huffman Code
--------------------------------
'h'         |                1
't'         |               01
'tdg'       |               00
```

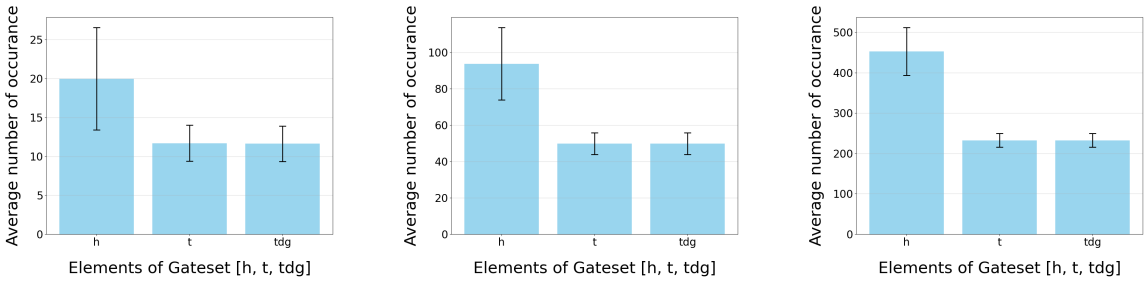
(b) Huffman code table corresponding to adjoining tree

Figure 4.2: (a) Huffman Tree Representation and (b) table of codes for Encoding of Quantum Gates

The number of bits needed for describing the quantum circuit using this version of Huffman coding is,

therefore, $21\times1+10\times2+9\times2 = 59$. This is a notable improvement over binary encoding, which needs 80 bits. The compression factor for this particular decomposition comes out to be $59/80 = 0.7375$.

In order to assign a general code for the gates, a data set of 200 Harr-random unitaries was selected. Following Solovay-Kitaev decomposition with a depth $d = 3$ and varying the degree of recursion $n = 1, 2, 3$, the gate frequencies in the decomposed circuits were stored.



(a) $n = 2$ (b) $n = 3$ (c) $n = 4$

Figure 4.3: Bar charts showing the average distribution of gate frequencies (`h`, `t`, `tdg`) in decomposed circuits obtained from Solovay-Kitaev decomposition of the data set of unitaries with $d = 3$ and $n = 2, 3, 4$

This distribution of frequencies is observed consistently over varying choices of the depth d and n. This lends generality to the adoption of this distribution of frequencies is consistently observed for varying choices of d and n, which supports the adoption of a general Huffman code for encoding the decomposed quantum circuit for any random unitary matrix.

4.2.3. v2: Encoding the SK-Basis

The second version of Huffman coding for the decomposed quantum circuits leverages gate sequences derived from the Solovay-Kitaev basis as the fundamental building blocks. The Solovay-Kitaev basis of depth d is the set of gate sequences up to length d. The Solovay-Kitaev basis for depth $d = 3$ is shown in the table. It should be noted that the null sequence `[]` is also a member of the basis, though it is not included in the table.

```
['h']
['t']
['tdg']
['h', 't']
['h', 'tdg']
['t', 'h']
['t', 't']
['tdg', 'h']
['tdg', 'tdg']
['h', 't', 'h']
['h', 't', 't']
['h', 'tdg', 'h']
['h', 'tdg', 'tdg']
['t', 'h', 't']
['t', 'h', 'tdg']
['t', 't', 'h']
['t', 't', 't']
['tdg', 'h', 't']
['tdg', 'h', 'tdg']
['tdg', 'tdg', 'h']
['tdg', 'tdg', 'tdg']
```

Revisiting our ongoing example of unitary matrix 3.3, its decomposed circuit 3.19. The subsequent step involves generating the frequency distribution of gate sequences, derived from the representation

SK basis element	frequency
['h']	0
['t']	4
['tdg']	4
['h', 't']	0
['h', 'tdg']	0
['t', 'h']	1
['t', 't']	0
['tdg', 'h']	0
['tdg', 'tdg']	0
['h', 't', 'h']	5
['h', 't', 't']	0
['h', 'tdg', 'h']	5
['h', 'tdg', 'tdg']	0
['t', 'h', 't']	0
['t', 'h', 'tdg']	0
['t', 't', 'h']	0
['t', 't', 't']	0
['tdg', 'h', 't']	0
['tdg', 'h', 'tdg']	0
['tdg', 'tdg', 'h']	0
['tdg', 'tdg', 'tdg']	0

Table 4.4: Frequency Distribution of Solovay-Kitaev Basis Elements in the Decomposed Quantum Circuit

of the quantum circuit in terms of gate sequences from the Solovay-Kitaev basis, as depicted in figure 3.20.

The Huffman tree for these gate sequences is structured to minimize the path lengths for the most frequent sequences, thereby reducing the total number of bits required for the entire circuit description.

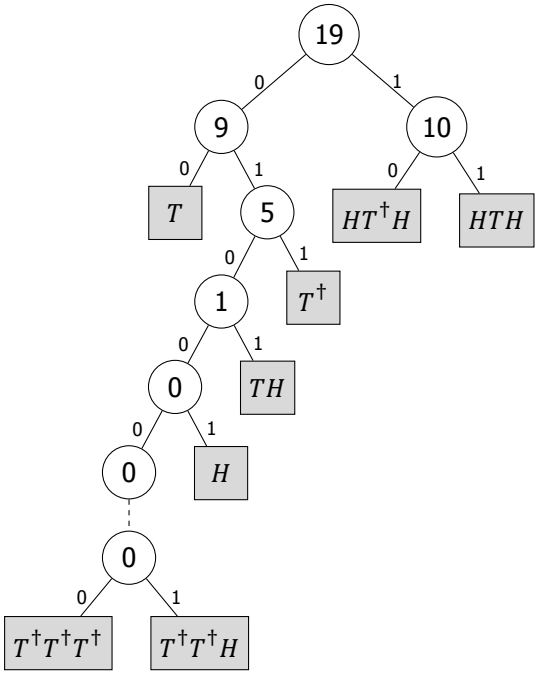


Figure 4.4: Huffman Tree for Solovay-Kitaev Basis Encoding. Leaves with a parent node value of 0 are not used in the quantum circuit. The dotted edge represents continued branching for brevity.

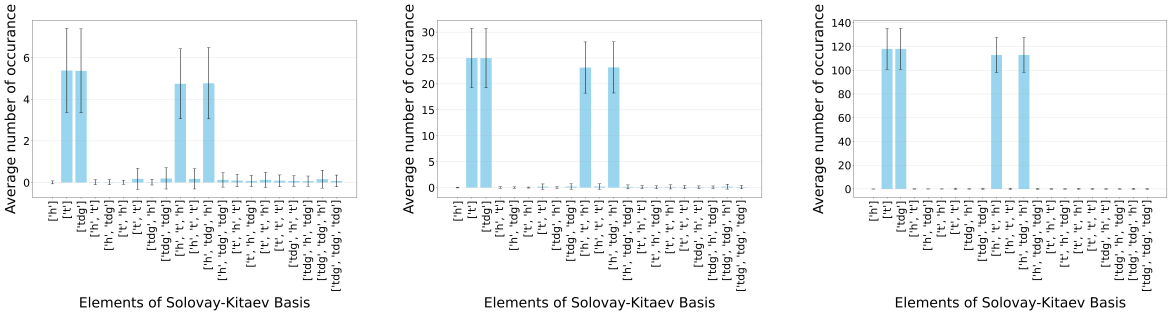
The Huffman codes for the SK basis elements are listed below after traversal through the tree 4.4
The number of bits required to describe the quantum circuit using Huffman coding of Solovay-Kitaev

'Instruction'	Huffman Code
"['h', 't', 'h']"	11
"['h', 'tdg', 'h']"	10
"['t']"	00
"['tdg']"	011
"['t', 'h']"	0101
"['h']"	01001
"['h', 't']"	010001
"['h', 'tdg']"	0100001
"['t', 't']"	01000001
"['tdg', 'h']"	010000001
"['tdg', 'tdg']"	0100000001
"['h', 't', 't']"	01000000001
"['h', 'tdg', 'tdg']"	010000000001
"['t', 'h', 't']"	0100000000001
"['t', 'h', 'tdg']"	01000000000001
"['t', 't', 'h']"	0100000000000001
"['t', 't', 't']"	0100000000000001
"['tdg', 'h', 't']"	01000000000000001
"['tdg', 'h', 'tdg']"	010000000000000001
"['tdg', 'tdg', 'h']"	0100000000000000001
"['tdg', 'tdg', 'tdg']"	0100000000000000000

Figure 4.5: Huffman codes for instructions from the Solovay-Kitaev basis

basis (v2) is 44. This is a substantial improvement over binary encoding, which needs 80 bits, and Huffman v1, which needs 59 bits. The compression factor for this particular decomposition comes out to be $44/80 = 0.55$.

Analogous to the experiments to evaluate the generality of the generated Huffman codes in v1, as illustrated in Fig. 4.3, we proceeded with a similar analysis for v2 and performed the Solovay-Kitaev decomposition of the same data set of unitaries with depth $d = 3$ and degree of recursion $n = 2, 3, 4$ and stored the frequencies of usage for the instructions from the Solovay-Kitaev basis.



(a) $n = 2$ (b) $n = 3$ (c) $n = 4$

Figure 4.6: Bar charts showing the average distribution of frequencies of instructions from the Solovay-Kitaev basis in decomposed circuits obtained from Solovay-Kitaev decomposition of the data set of unitaries with $d = 3$ and $n = 2, 3, 4$

Again, we observe a consistency in the instructions in Fig. 4.6, that are most frequently used and ones that are not. This pattern of usage holds true of a varying choice of recursion depth n; however remains specific to the depth. For different depths, we would obtain a basis set different from the one in the table, and consequently, a different distribution of usage frequencies. The significant remark on the observed average usage frequencies in Fig. 4.6 is that there's only a handful of instructions that are used much more frequently compared to the remaining instructions in the set. This trend opens the possibility of not including all the elements of the basis set in the coding process, and thereby leading

to a shorter dictionary. This is undertaken in the next version (v3)

4.2.4. v3: Encoding a Selection from SK-Basis

The third version of Huffman coding (v3) proposed in this thesis is based on the encoding of the quantum circuit in terms of the instructions in the Solovay-Kitaev basis, which is the same as in v2. However, the fundamental novelty in this version is that we make a selection of instructions form the basis and only include these selected instructions in the Huffman coding. The fundamental gates `'h'`, `'t'` and `'tdg'` are trivially included in this selection. Any other instruction appearing in the decomposed quantum circuit that doesn't belong to the selection gets broken down in terms of the fundamental gates, and their usage frequencies are updated. The selected Huffman instructions from the original basis are depicted in red color in the box 4.7.

```
['h']
['t']
['tdg']
['h', 't']
['h', 'tdg']
['t', 'h']
['t', 't']
['tdg', 'h']
['tdg', 'tdg']
['h', 't', 'h']
['h', 't', 't']
['h', 'tdg', 'h']
['h', 'tdg', 'tdg']
['t', 'h', 't']
['t', 'h', 'tdg']
['t', 't', 'h']
['t', 't', 't']
['tdg', 'h', 't']
['tdg', 'h', 'tdg']
['tdg', 'tdg', 'h']
['tdg', 'tdg', 'tdg']
```

Figure 4.7: Elements of the Solovay-Kitaev basis of depth $d = 3$, selected instructions for v3 printed in red

This selection of instructions is made from the observed trend of usage frequencies over changing degrees of recursion presented in Fig. 4.6. To clarify this argument, a raincloud plot for the usage frequencies of instructions from the Solovay-Kitaev basis is presented in Fig. 4.8. The width of the 'cloud' and the density of the scattered points represent the number of samples at a certain usage frequency. The mean and standard deviation of usage frequencies are depicted by the diamond markers and error bars, respectively. The instructions portrayed in red color have a much higher average usage frequency. `['h']` as an instruction has a low average usage, while H gate has a higher average frequency in the decomposed quantum circuits 4.3. It can then be inferred that the dominant usage of H gate comes from the `['h', 't', 'h']` and `['h', 'tdg', 'h']` instructions.

The Huffman tree and table of codes for the selected Solovay-Kitaev basis instructions are presented in Fig. 4.9. The number of bits for describing the quantum circuit using this version of Huffman coding is 45, and the compression factor comes out to be $45/80 = 0.5625$. This number is only marginally higher than the number of bits required in v2 of Huffman coding (44), but the dictionary of codes is appreciably smaller.

To answer the concern of generality, we go back to the trends in Fig. 4.6. The pattern of average frequency remains consistent over different degrees of recursion n for a set depth d. Therefore, a particular selection of instructions for the encoding also remains consistent for a particular depth d over any choice of n.

4.2.5. Handling the Qubit IDs

Scaling up the decomposition and encoding routine to multi-qubit systems requires some changes to the stream of instructions describing the quantum circuit. In addition to the stream of encoded

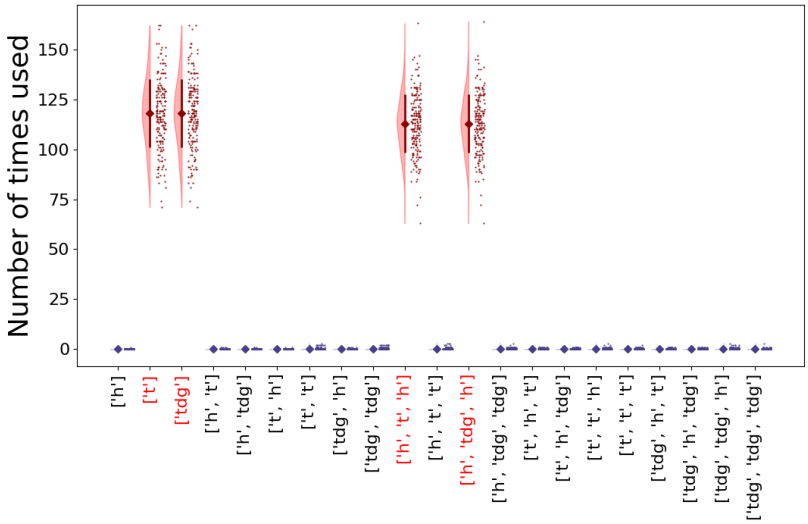
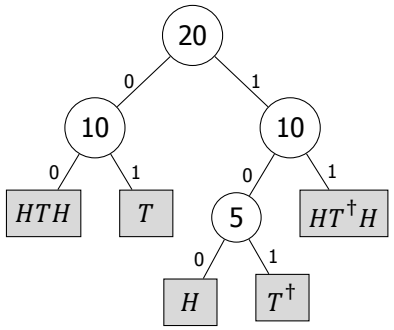


Figure 4.8: Raincloud plot of usage frequencies of SK-basis instructions in decomposed circuits performed with $n = 4$.



(a) Huffman Tree representation for encoding selected quantum instructions

'Instruction'	Huffman Code
"['t']"	01
"['h', 't', 'h']"	00
"['h', 'tdg', 'h']"	11
"['tdg']"	101
"['h']"	100

(b) Huffman code table corresponding to adjoining tree

Figure 4.9: (a) Huffman Tree Representation and (b) table of codes for Encoding of selected instructions from the Solovay-Kitaev basis(huff_v3)

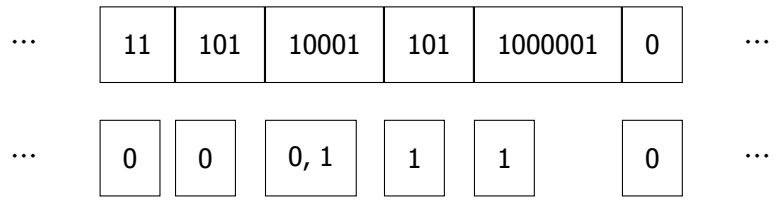


Figure 4.10: Example segment of a Huffman v3 encoded instruction stream and the corresponding strip of qubit ID stream. The code `10001` corresponds to a cx gate and, therefore, has two entries in the qubit ID specifying the control and target.

opcodes from the SK-basis, we also need to send the stream of qubit IDs to direct the operation of instructions on the designated qubit. In this project, a simple binary encoding is chosen for encoding the qubit IDs with the number of bits for encoding scaling at $\log_2(N)$, where N is the size of the system. The qubit ID stream is tailored according to the version of Huffman encoding adopted for the instruction stream, such that each opcode corresponds to one (or two in the case of `cx`) entry in the qubit ID stream. An example segment of an encoded instruction stream and the corresponding qubit ID stream for a 2 qubit system is shown in Fig. 4.10. In the event of scaling up this routine to much

larger size systems, Huffman coding can be employed to encode the qubit ID stream as well, but at the moment, that is a future outlook of this project.

4.2.6. Results and Comparison

In the prior subsections, we have outlined the three variations of Huffman coding for developing a compressed representation of decomposed quantum circuits. This section focuses on analyzing the performance of these encoding techniques. Prior to discussing the experimental setups and results, we will first establish the context and methodology used in our investigation.

The initial experiments are conducted using datasets of Haar-random unitaries for quantum systems of one qubit (1q), two qubits (2q), and three qubits (3q). These tests primarily examine the performance of random circuits. In these experiments, binary coding is utilized as the reference standard to determine the compression efficacy of the Huffman coding methods. The effectiveness of each encoding method is quantified by the compression factor, which is calculated by dividing the bit-length of the Huffman encoded representation of a decomposed quantum circuit by the bit-length of the same circuit encoded in binary. Fig 4.11 displays the average trends of compression factor for the 3 versions of Huffman coding for the dataset of single-qubit unitaries that are decomposed into quantum circuits using SKD. Similarly, the subsequent plots 4.12, 4.13 display the average compression factors for datasets of 2-qubit and 3-qubit unitaries, respectively, that are decomposed into quantum circuits using QSD + SKD.

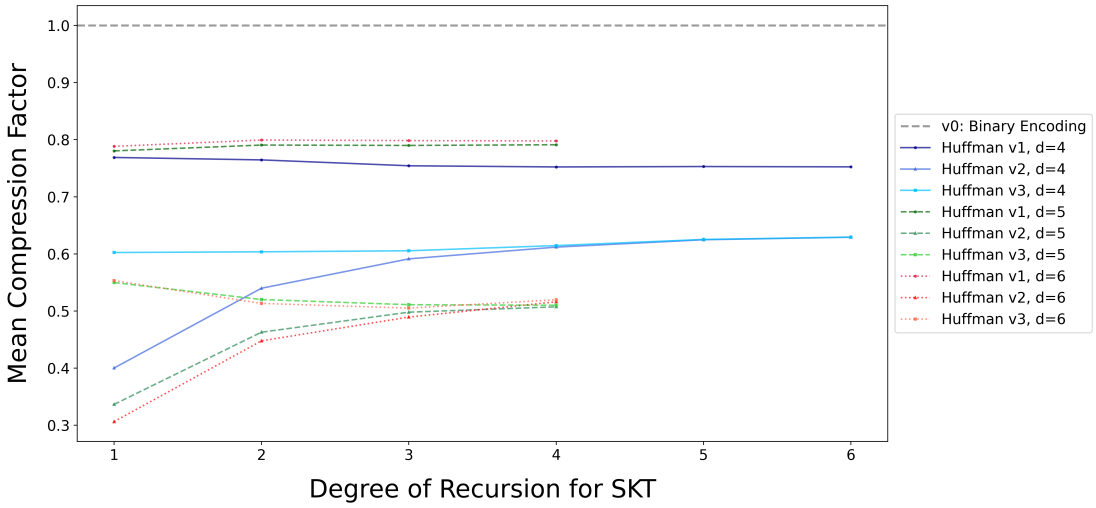


Figure 4.11: Trend of average compression factor $\frac{\texttt{len(huff encoded)}}{\texttt{len(bin encoded)}}$ vs degree of recursion n for Solovay-Kitaev decomposition for data set of 200 1q Harr-random unitaries.

Let's drive our attention to v3 of Huffman coding. As explained in subsection 4.2.4, it involves a selection of gate sequences from the S-K basis, and the selection trivially includes the fundamental gates. The gate sequences appearing in the quantum circuit that don't belong to the selection are simply expressed as a sequence of usage of the fundamental gates. Based on the updated usage frequencies of elements in the selection, a Huffman code is generated. This process can be imagined to have an intermediate extent between two limits. The lower limit would be v1, which only has the fundamental gates as the basic units for coding. And the higher limit is v2, which includes all the instructions of the S-K basis for coding. v3 is designed to screen off the instructions from coding that are used very rarely on average and lead to a much smaller Huffman tree and code dictionary.

On that note, the noteworthy inference drawn from the preceding plot is the observed performance trend of v3 Huffman coding compared to its predecessor, v2, which can be regarded as the most optimal in terms of compression. As apparent, v3 initially exhibits inferior performance to v2 at lower levels of n. However, as the degree of recursion for the S-K decomposition increases, v3 gradually achieves a compression factor comparable to that of v2. As evident from 3.16, the process fidelity of

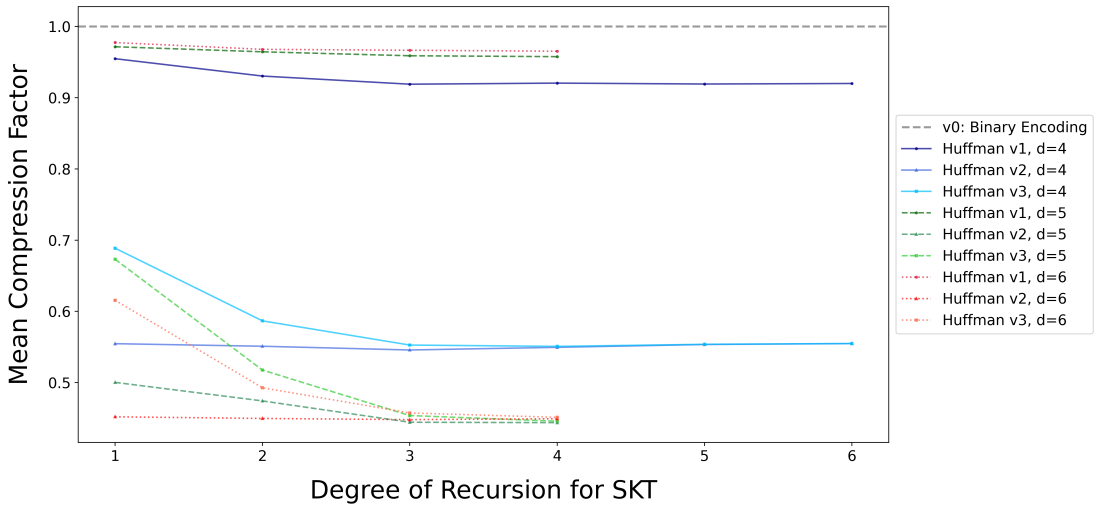


Figure 4.12: Trend of average compression factor $\frac{\texttt{len(huff\_encoded)}}{\texttt{len(bin\_encoded)}}$ vs degree of recursion n for Solovay-Kitaev decomposition for data set of 200 2q Harr-random unitaries.

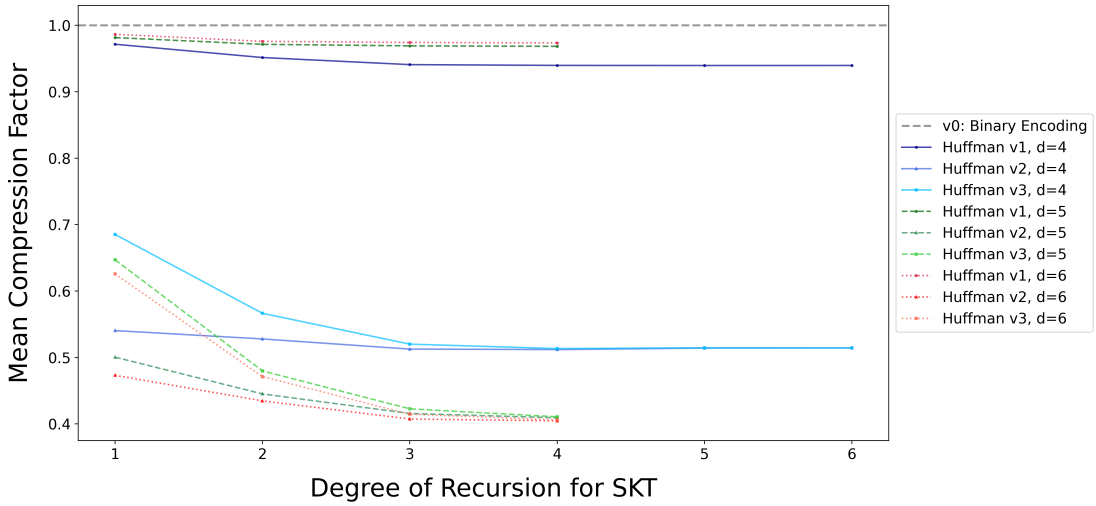


Figure 4.13: Trend of average compression factor $\frac{\texttt{len(huff\_encoded)}}{\texttt{len(bin\_encoded)}}$ vs degree of recursion n for Solovay-Kitaev decomposition for data set of 200 3q Harr-random unitaries.

decomposition increases for increasing degree of recursion.

Therefore, the principal conclusion is that at elevated levels of recursion, where the decomposed quantum circuit reliably approximates the target unitary, v3 of Huffman coding attains compression efficiency equivalent to that of v2 while concurrently maintaining a significantly reduced code dictionary. This observation underscores the assertion that v3 of Huffman coding represents the most accurate estimation of description complexity for quantum circuits, as delineated in the thesis's findings.

The observation of analogous trends for 2-qubit and 3-qubit systems presented in 4.12, 4.13 offer the same inferences in general from that of the 1-qubit system. The length of decomposed circuits for 2-qubit and 3-qubit systems is generally much longer in comparison to that of 1-qubit systems due to the nature of QSD + SKD. At the same time, the compression factors are also smaller for these larger systems. This observation reinforces the characteristic of Huffman coding, which is that its compression

efficiency becomes more pronounced when handling larger systems of data.

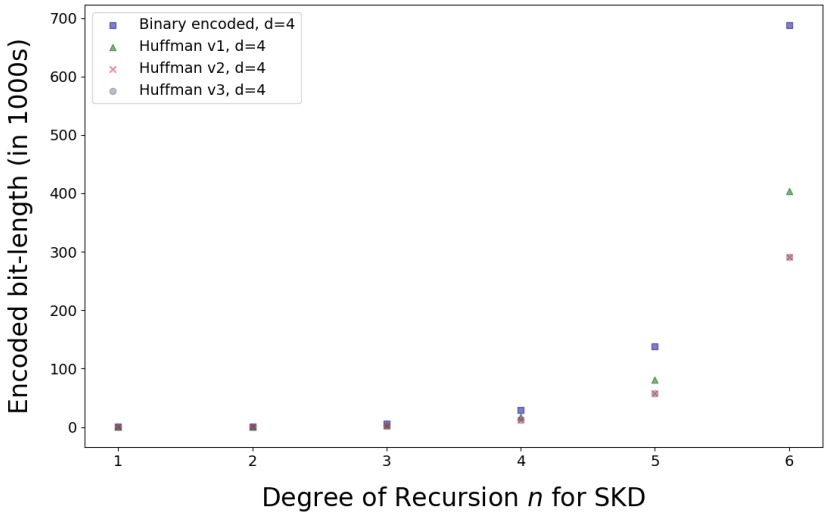


Figure 4.14: Trends of average encoded bit-length are examined for three versions of Huffman coding and binary coding, plotted against the degree of recursion n for Solovay-Kitaev decomposition. The analysis is based on a dataset of 200 2-qubit Harr-random unitaries.

Figure 4.14 illustrates the bit-length of the encoded instruction stream for decomposed quantum circuits as a function of the degree of recursion for SKD. Previous figures in this subsection have concentrated on the compression factor, which serves to determine the compressibility or provide an indirect measure of the description complexity of decomposed quantum circuits. In contrast, this figure focuses on the absolute encoded length of the instruction stream, offering a more direct quantification of the description complexity.

4.3. Benchmarks Results

In the past section 3.5.1, we evaluated the performance of the SKD+QSD decomposition scheme and selected the optimal values of parameters: a depth $d = 5$ and a degree of recursion $n = 4$. In the preceding subsections, we defined three versions of Huffman coding, along with standard binary encoding, as our baseline.

This subsection deals with the performance of the decomposition and encoding routine for benchmark circuits. We use benchmark circuits sourced from MQT Bench [65], a curated library within the Munich Quantum Toolkit. The selection of benchmarks for our experiment consists of 79 scalable benchmark circuits ranging from system sizes 2 to 6 qubits. The chosen benchmark circuits are listed in a table in the Appendix. So, far we have implemented and tested our decomposition and encoding routine on Harr-random unitaries. These circuits, though being very general, don't necessarily have practical applications in the real world. Therefore, we are going to execute our routine on these benchmark circuits which are algorithms developed for real-world use cases. We aim to study the performance of the routine on benchmarks to test whether its performance is consistent with that on sets of Harr-random unitaries. If the algorithmic insights drawn from sets of random circuits also apply to benchmark circuits of the same size, and to explore if we get any new or interesting insights.

An insight of the performance of the decomposition is depicted in 4.15 with the process fidelity as the red line plot. The circuit depth for decomposed circuits is presented as a bar plot in blue and follows a logarithmic scale on the y-axis. The benchmarks are sorted according to the increasing size of the system. The drop in the fidelity and exponential increase of circuit depth for higher system sizes is indicative of the inefficiencies of the Solovay-Kitaev decomposition process with the limited gate set `[h, t, tdg]`.

To further examine the performance of the decomposition routine with the depth of SK-basis set at $d = 5$ and degree of recursion $n = 4$ on the benchmark circuits, we prepare small datasets of 20 Harr-

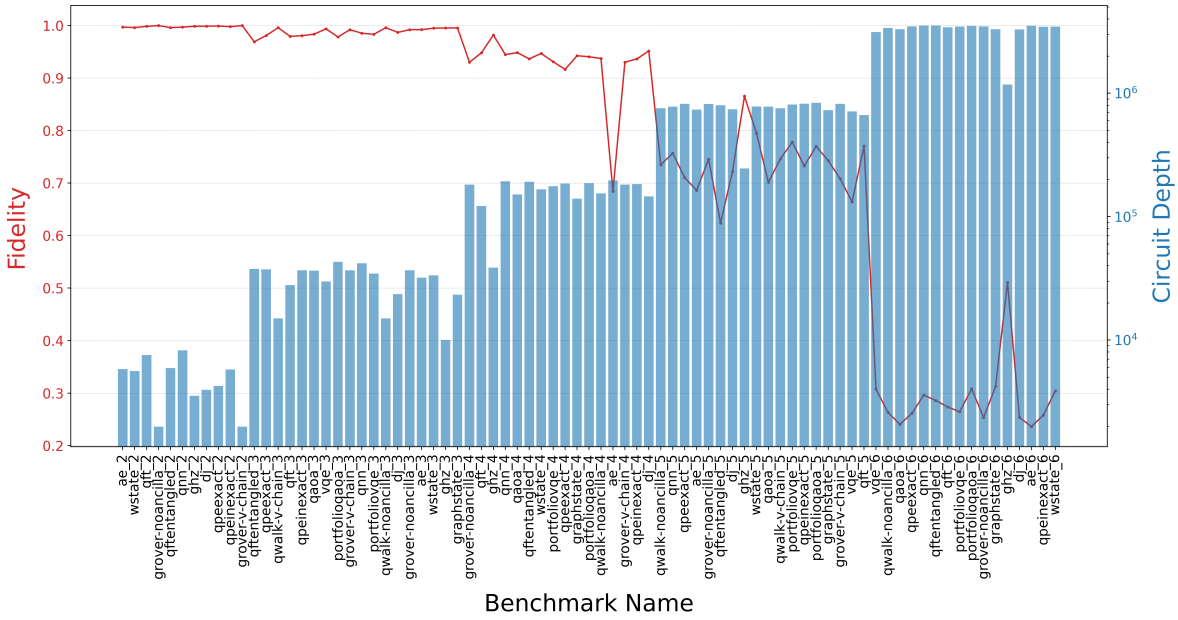


Figure 4.15: Process fidelity and circuit depth of benchmarks post decomposition into `[h, t, tdg, cx]`

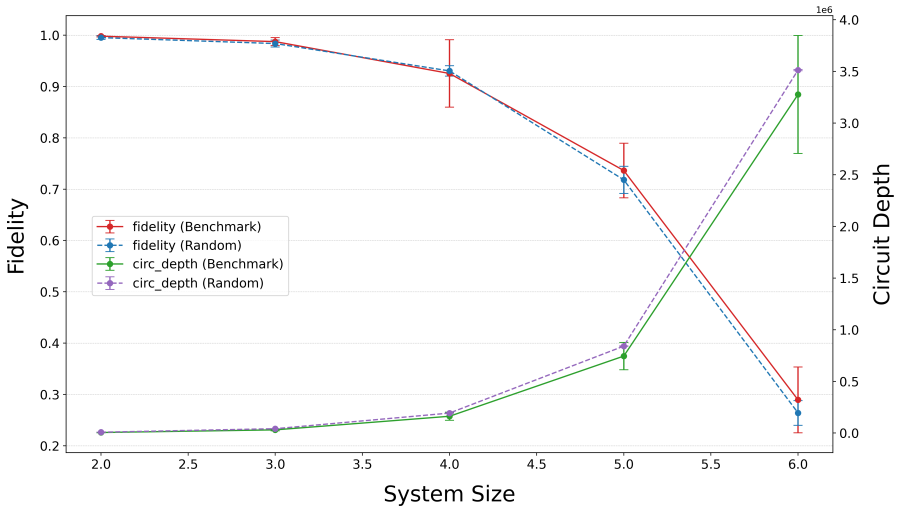


Figure 4.16: Mean and standard deviation of process fidelity and circuit depth plotted for benchmark circuits and sets of 20 random circuits for each size.

random unitaries for each size ranging from 2 to 6. The average fidelity of decomposition and circuit depth over benchmarks and random circuits grouped according to their size is presented in 4.16. We observe a closely similar performance of decomposition for benchmark and random circuits; however, the standard deviation is higher for the benchmarks, which might be an indication of the decomposition method being more sensitive to the complexity of the benchmarks.

The compression factor trends remain consistent over different benchmarks of the same size. There is a greater variation in compression factor for smaller benchmarks as compared to larger systems. The number of bits for binary encoding of the qubit IDs increases by 1 when the system size increases from 2 to 3 and at 4 to 5. At these changing points, we see a slight drop in the average compression factor for the v2 and v3 versions. This is because the v0 binary encoding suffers a larger increase in the size of the qubit ID stream, whereas the increase in the size of v2 and v3 Huffman encoded streams is not that much. The increase in the size of the total description stream is offset by the improved performance of Huffman coding for larger systems, therefore leading to a gentle drop in compression

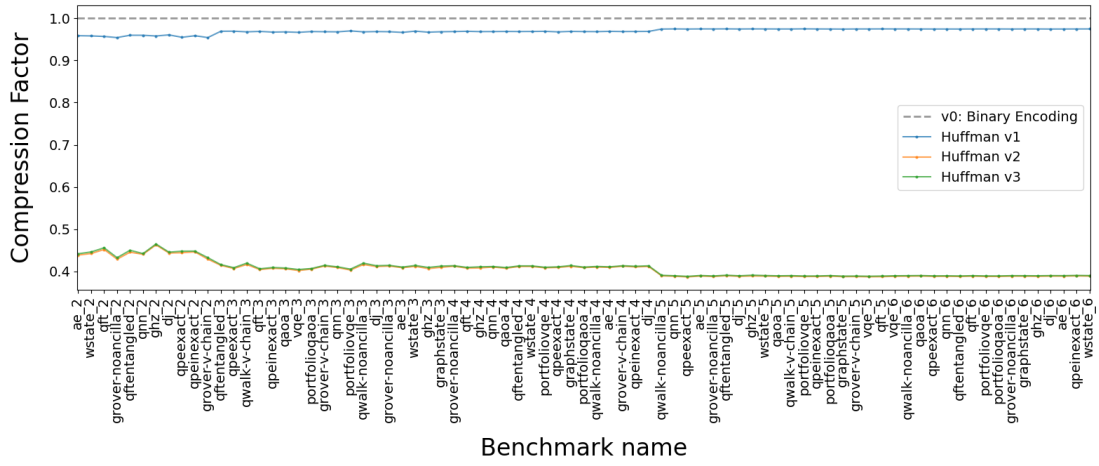


Figure 4.17: Compression factor = $\frac{\texttt{len(huff\_encoded)}}{\texttt{len(bin\_encoded)}}$ for benchmark circuits. Benchmarks sorted according to system size.

factor. The plot reveals that Huffman v2 and v3 maintain a stable and low compression factor across various benchmarks, suggesting robustness and effectiveness in different circuit configurations. The consistency in performance across diverse benchmarks indicates that these Huffman encoding schemes generalize well across different types of quantum circuits.

Another takeaway from these results stems from the consistent performance of the decomposition and encoding routine on benchmarks as compared to sets of Harr-random unitaries. The similarity in performance is evidenced in Fig. 4.16. Harr-random unitaries with larger sizes are general, easy to initialize, and run the decomposition and encoding routine on ensembles of samples. In this process, we discover high-level abstractions that also carry over to algorithms of the same size for real-world use cases. The dataset of real-world quantum algorithms is limited in number. Hence, this convenience of running experiments on analogous Harr-random unitaries to gain valuable algorithmic insights has great significance.

4.4. Action of Lossless Compression: Bzip2

Lossless compression is a method of data compression that preserves all the original data ensuring no loss of information in the compression process, allowing the exact original data to be reconstructed from the compressed data. These algorithms operate by searching for patterns and redundancies in data that can be expressed in a shorter and more concise way. bzip2 [26], developed by Julian Seward in 1996, is an efficient and popular lossless compression technique attributed to its excellent balance of compression efficiency and speed.

The pipeline depicted in Fig. 4.18 consists of a sequence of steps for transformations and encoding that rearrange and encode the data. These steps and their order are adopted to complement each other and improve overall efficiency. The distinguishing innovation of bzip2 is the Burrows-Wheeler transform (BWT) [60]. It is a transformation algorithm that rearranges the data to have runs of similar symbols and improves the effectiveness of the subsequent move-to-front transform (MTF) and run-length encoding. As a result, bzip2 is very efficient in the compression of text that contains repeated patterns and is asymmetric in nature; the decompression is faster than the compression. These qualities make it a suitable choice for the compression of QASM instructions of quantum circuits, which contain repeated instances of gate operations and qubit IDs.

4.4.1. Bzip2 over Encoded Instruction Streams

After achieving significant data compression through various Huffman encoding techniques tailored to quantum circuits, the application of further compression using bzip2 presents an interesting avenue

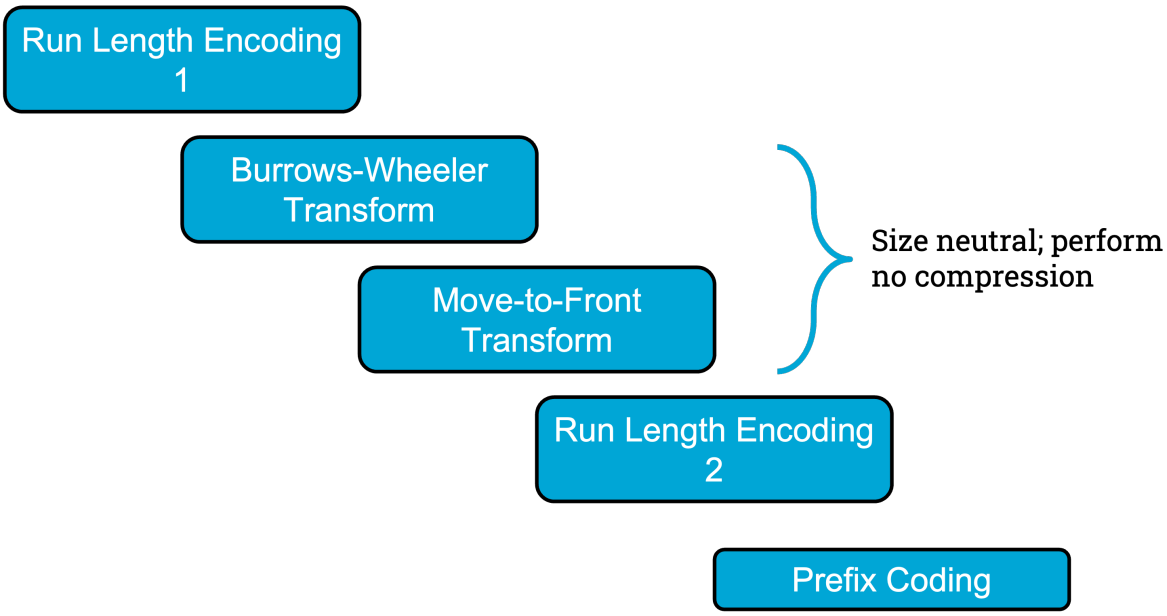


Figure 4.18: Pipeline of operations undertaken in bzip2 compression.

to explore. Given the nature of encoded quantum instruction streams, which often contain repetitive sequences of 0s and 1s, bzip2 can further optimize the transmission efficiency of these streams.

The process involves taking the binary encoding (v0) and the Huffman-encoded streams from the three variations described earlier— Huffman gate encoding (v1), Solovay-Kitaev basis encoding (v2), and selective SK-basis encoding (v3)— and subjecting them to bzip2 compression. The aim is to evaluate how much further reduction in bit-length can be achieved.

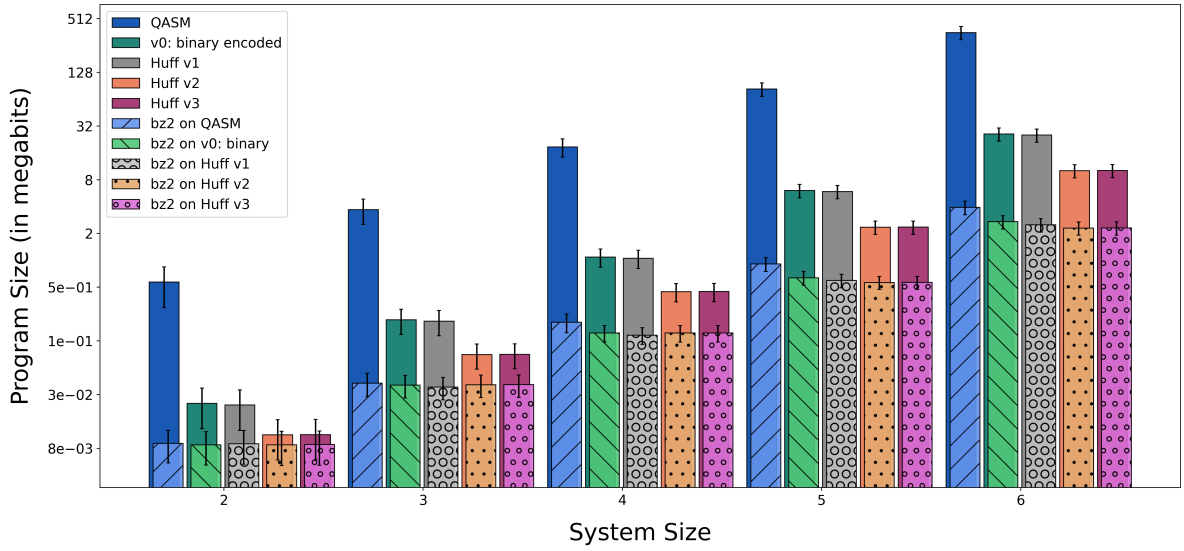


Figure 4.19: Mean program sizes plotted in log 2 scale for benchmark circuits grouped according to the size of the system. Overlaid bars depict post-bzip2 compression program sizes.

Each encoded stream is compressed using bzip2, and the resultant sizes are recorded. The average original and the bzip2-compressed program sizes are compared on a $\log_2$ scale in Fig. 4.19 to assess the effectiveness of applying a secondary compression stage. The additional compression achieved suggests that layering compression techniques, starting with Huffman encoding tailored to capture the quantum 'context' of circuits effectively, followed by bzip2, provide a robust method for minimizing quantum program size.

To conclude, this chapter offers a detailed and comprehensive exploration into the application of Huffman coding techniques for the compression of decomposed quantum circuit instruction streams. The experiments began with adopting a binary encoding scheme as the baseline and progressively incorporating more sophisticated Huffman encoding strategies. These strategies leverage the individual gate frequencies(v1) and the Solovay-Kitaev basis instructions (v2 and v3) to optimize the encoding schemes for quantum instructions. Significant improvements in compression efficiency were demonstrated through these methods, which were then further enhanced by applying bzip2 compression, illustrating the effectiveness of combining multiple compression strategies. The chapter sets the stage for evaluating these methodologies in the subsequent chapter. The next chapter will delve into the practical utility of the proposed methods in more efficient quantum control, estimation of information-theoretic measures for quantum computation, and discovery of contextual abstractions in quantum circuits.

5

Optimal QISA Design

Compression is Comprehension.
- Gregory Chaitin

This chapter involves optimal QISA design. Quantum instruction set architecture is an essential component of the quantum compiling process, and it serves as an interface between the high-level software level stack and the lower hardware components.

5.1. The Quantum Stack

As the name suggests, the quantum stack refers to the layered architecture used in the design and development of quantum computing systems. The topmost layers in the stack correspond to the application layers. They involve the computation model, the quantum algorithm for a specific application, and its programming in a quantum programming language like Qiskit [56], Cirq [66], PennyLane [67] or Q# [68]. The application layers deal with theoretical constructs of quantum computation and higher-level abstraction of the computation process.

Next comes the compilation layers. These are the layers that serve as an interface between the software layers on top and the hardware layers below. They consist of subroutines like mapping, decomposition, routing, and scheduling operations. These layers consider the specific device constraints and capabilities and compile the high-level quantum algorithms into Quantum Instruction Set Architecture (QISA) instructions. These levels play a pivotal role in bridging the gap between quantum and classical descriptions of systems. An optimized design of the QISA improves the efficiency of the subsequent control operations in terms of energy and execution times.

In the hardware layers, the control architecture manages classical operations on the qubits, including gate operations and readout, and incorporates error correction mechanisms. At the base of the stack lies the quantum processor itself, which can be implemented through various technologies, including superconducting qubits, neutral atoms or trapped ions, and semiconductor or photonic systems.

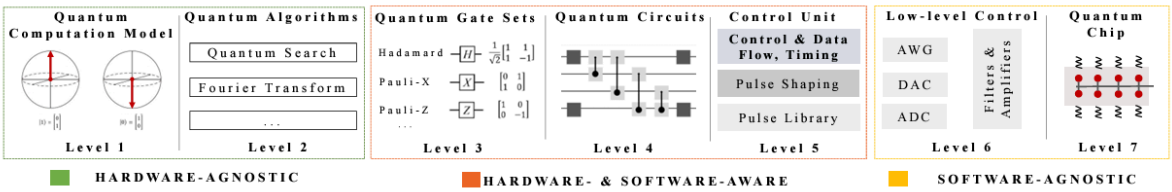


Figure 5.1: Quantum stack. Levels 3, 4, and 5 involve the compiling steps, with this project focussing specifically on level 4. Source: [17]

5.2. Energy Bottlenecks in Quantum Control Architecture

Quantum control architecture is a crucial component in the development of quantum computing technologies. The devices are essential for qubit manipulation, error correction, and feedback control.

Based on the nature of the implementation of the qubits, control is performed by various methods: superconducting qubits use microwave pulses, trapped ions are manipulated with laser pulses, photonic qubits via optical devices like beam splitters, and silicon quantum dots through electrical pulses. Most of the systems operate at extremely low temperatures close to the absolute 0. Fault-tolerant quantum computation dictates that the readout, calculation, and correction steps be performed in real-time, all occurring before the qubits decohere.

Cryogenic control architectures hold significant potential for scalable quantum control [16]. Being close to the operation temperature of the qubits, they offer a compact solution for performing localized readout and feedback-controlled qubit manipulation. These qualities provide a substantial benefit in performing error correction integrated with the qubits [16] and overcoming wiring bottlenecks in systems with a large number of qubits [69]. Field programmable gate arrays (FPGA) are a suitable candidate for these low-time and energy-consuming devices as they are programmable and can be reconfigured to implement diverse computing routines [54].

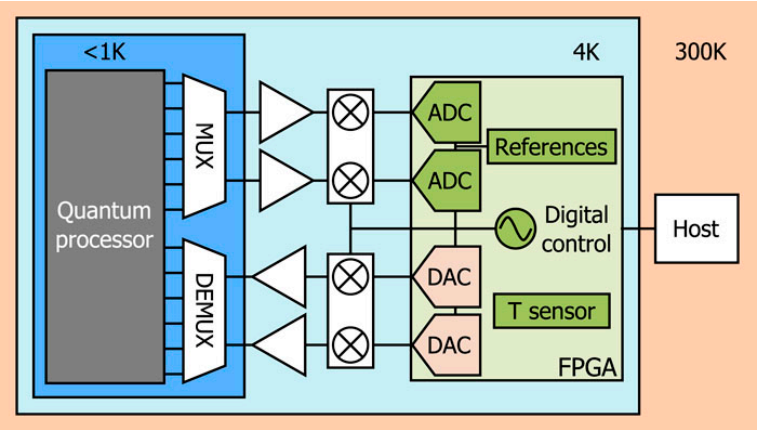


Figure 5.2: Implementation of a cryogenic control platform. Source: [54]

One of the most important limitations of these devices is the tight power consumption budget associated with their cryogenic operation temperatures. Employing these devices at temperatures far lower than their rated operating temperature range is challenging because of anomalous behaviors and non-idealities in the I-V characteristics [54]. Dilution refrigerators, which are popularly used for cooling the quantum processors to sub-Kelvin temperatures, are incredibly sensitive to the power consumption of the devices. At these temperatures, even small amounts of heat can significantly impact the performance of the system, including introducing thermal noise and reducing coherence times of quantum states. The power dissipation budget of dilution refrigerators for cooling quantum processors determines the maximum allowable heat load to maintain ultra-low temperatures. It ranges from around 1W at 4K and goes down to the order of several μW at 10 mK [70]

Huffman-encoded quantum instruction sets offer a redefined, dense (along the depth) representation of quantum circuits in terms of S-K basis instructions. The choice of an optimal gate set of decomposition along with Huffman version v3 4.2.4 proposed in the thesis aims to provide a reliable description of the quantum circuit that can be executed in fewer operations compared to the execution of native gates. The reduced throughput requirement for the encoded instructions also offers a key advantage in the transmission of the instruction stream from room temperature to the operating temperature of the control architecture. Therefore, the method of compression of quantum instructions qualifies as a high-level approach towards energy-efficient cryogenic quantum control.

5.3. Contemporary QISA Design

QISA is analogous to classical instruction set architectures such as Intel x86, ARM, and RISC-V and serves as a crucial intermediary between the higher-level quantum programming languages and the lower-level quantum hardware operations. It plays a significant role in providing a level of abstraction that is more comprehensive in describing the physical details of the micro-architecture and hardware than high-level quantum programs and algorithms [71]. eQASM [58] tackles such shortcomings of existing frameworks such as Quil[72], QuMIS[73] and openQASM[56] in offering a scalable and com-

prehensive program flow control. It features an executable instruction set architecture framework that is more scalable and flexible and allows the configuration of quantum operations with explicit timing specifications.

Alongside the efforts to enhance ISAs by incorporating specific control and hardware details, insights from overlying quantum algorithms and program synthesis can be incorporated into the design space exploration of QISAs to better bridge the gap between high-level quantum semantics and underlying control and hardware execution. This integration will not only streamline the translation from theory to practice but also expand the scalability and applicability of QISAs, as well as the resource efficiency of quantum computation for a broader array of experiments. Notable contributions to this field have been made by Butko et al. at the Lawrence Berkeley National Laboratory in their paper [17]. The authors propose a scalar quantum extension termed *QUASAR* and a vector extension *qV* based on the existing RISC-V ISA [51]. The authors highlight the efficacy of their proposed QISA frameworks by comparing them with existing frameworks(eQASM) in terms of metrics such as encoding efficiency and execution times. Assessment of ISAs by characterizing the control processor performance in terms of parameters such as circuit complexity, gate density, and diversity offers an 'algorithmic evaluation' of the QISA.

5.4. Advantages of Proposed QISA

The following two subsections respectively focus on the practical and theoretical advantages of the proposed QISA.

5.4.1. Estimation of Energy Gains

The adoption of cryogenic control architectures as motivated in Sec. 5.2 necessitates the development of efficient methods for transmitting instructions from room temperature (around 300K) to cryogenic environments (typically around 4K). This requirement stems from the need to control quantum processors operating at extremely low temperatures without introducing excessive heat that could disturb the delicate quantum states necessary for computation. Efficient transmission methods must, therefore, minimize thermal load while maintaining high data integrity and speed.

To this aid, different technologies have been proposed in the recent past for establishing high-speed, low-power transmission links between room temperature(RT) and 4K. The energy per bit of data ranges from a few hundred fJ/bit [74, 75] using optical fibers with photonic links and CMOS-based transceiver chip [76] to a few pJ/bit using CMOS DAC-based wireline transmitter [77].

The Huffman encoding routine followed by bz2 compression offers a significant reduction in the bit-size description of quantum circuits. In this subsection, we aim to perform an estimation of energy gains for the cryo-CMOS DAC-based wireline transmitter technology presented in [77] with an energy rate of 2.46 pJ/bit and data rate of 40 Gbps.

The figure 5.3 depicts the mean energy load for transmission of the quantum program (instruction stream + qubit ID stream). Adopting v3 Huffman coding proposed in this thesis brings down the heat load to approximately 2.5 % of the that incurred by QASM format (and 40% of that incurred by binary encoded format). Performing lossless compression(bzip2) on the Huffman v3 encoded instructions brings the heat load down to around 1 % of that by the QASM. Paying attention to some caveats, the subsequent bzip2 compression adds an additional overhead for decompression at the cryo-level. bzip2 is asymmetric, taking less time for decompression than compression, and is a characteristic that makes it a good choice for lossless compression. Huffman v3, offering asymptotically identical compression as v2, is a better choice for encoding because of its shorter code dictionary.

The diminished heat load during data transmission signifies a promising avenue for scalability in program size. As we envisage the era of fault-tolerant quantum computation, the complexity and size of quantum programs are poised to expand significantly, encompassing logical-level operations. The decreased energy overhead resulting from instruction transmission alleviates the stringent power constraints in cryogenic environments, particularly on control processors.

5.4.2. Estimation of Theoretical Measures

In this subsection, we evaluate the methods presented in the thesis from a theoretical perspective and assess the decomposition and encoding routine as an estimate for quantum description complexity. Enumerating some criteria or factors that can be considered as requirements for a practical estimate of description complexity:

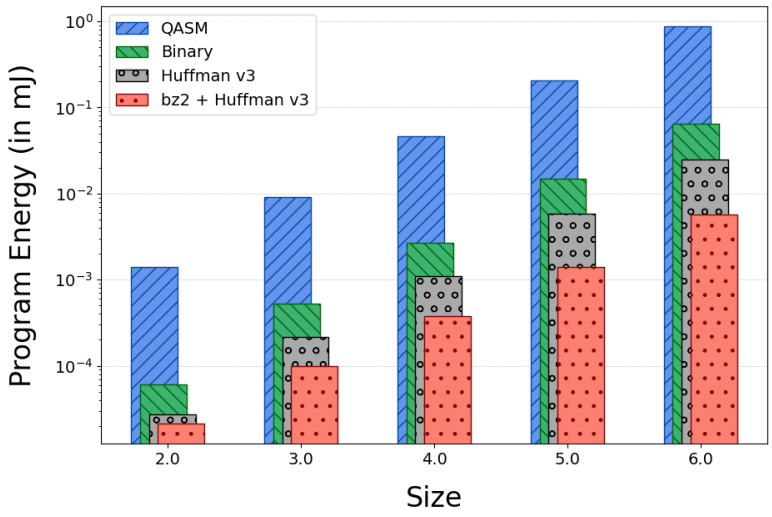


Figure 5.3: Mean Energy Consumption for Quantum Program Transmission in three formats: QASM, Binary encoded, Huffman v3 encoded, and bz2 compression on Huffman v3 encoded data. Each bar represents the mean energy consumption for a given program size, with bars for each format overlaid for direct comparison. Energy per bit = 2.46 pJ/bit [77]

- Minimality: The complexity measure should capture the shortest possible description of an object (or a quantum circuit in this context), implying the optimal compression of the information contained within the object. We observe a consistent compression factor of 0.4 for the Huffman v2 and v3 methods over binary encoded streams 4.17.
- Universality: The Solovay-Kitaev theorem essentially states that if we have a universal gate set, we can approximate any single-qubit gate to precision ϵ with a circuit length that is poly-logarithmic in $1/\epsilon$. Therefore, the SK basis made from a gate set (e.g. [h, t, tdg]) is capable of modeling the behavior of a universal quantum computer.
- Invariance: Choosing an alternate gate-set for the decomposition of the unitary matrix results in an alternate language of description of the same program. As per the Solovay-Kitaev theorem, if we change from one universal set to another, both of which are capable of densely spanning the space of SU(2), the decomposition length changes at most by a constant factor. This is because the efficiency of the algorithm fundamentally hinges on the density of the gates in SU(2) and not strictly on their individual characteristics.

Based on these criteria, the Huffman encoding of SK instructions offers a practical and effective method to approximate the description complexity of quantum circuits. Next, inspired by [22], we aim to describe the total quantum complexity as the sum of the quantum circuit complexity and quantum description complexity. The quantum circuit complexity can be defined as the size of the system (no. of qubits) multiplied by the depth of the circuit. Adopting this definition as the measure of circuit complexity and Huffman v3 as the measure for description complexity, the circuit and description complexity for benchmarks is presented in 5.4

We observe that the description complexity does not increase as rapidly as the circuit complexity with the growing system size. This suggests that the Huffman encoding of SK basis instructions is successful in capturing significant contextual or abstract information about the quantum circuits, which simplifies their representation without losing essential details necessary for accurate computation.

5.5. Usefulness in Discovering High-Level Q-Programming Abstractions

Studying high-level quantum programming abstractions through program synthesis is crucial for advancing the field of quantum computing. This approach is akin to understanding language through common substrings or patterns, which simplifies the creation and comprehension of new words or phrases.

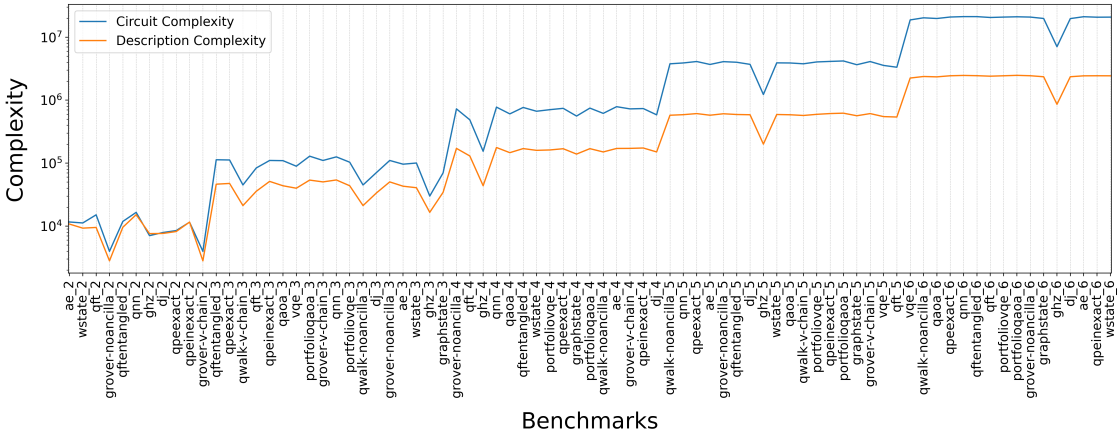


Figure 5.4: Circuit Complexity and Description complexity plotted for benchmark circuits arranged according to increasing size.

For example, in classical programming, identifying common patterns such as loops and conditional statements has led to the development of higher-level programming constructs like for-each loops and lambda expressions, which simplify coding tasks and enhance readability and maintainability. Similarly, in quantum computing, recognizing and formalizing frequently occurring sequences of quantum gates (like those from the Solovay-Kitaev (SK) basis) as high-level constructs can provide similar benefits. These sequences, once identified and abstracted, can serve as fundamental "building blocks" or macros, which encapsulate complex quantum operations into single, reusable components.

Selection for $d = 4$	Selection for $d = 5$	Selection for $d = 6$
['cx'] ['h'] ['t'] ['tdg'] ['h', 't', 'h'] ['h', 'tdg', 'h'] ['h', 't', 'h', 't'] ['h', 't', 'h', 'tdg'] ['h', 'tdg', 'h', 't'] ['h', 'tdg', 'h', 'tdg'] ['t', 'h', 't', 'h'] ['t', 'h', 'tdg', 'h'] ['tdg', 'h', 't', 'h'] ['tdg', 'h', 'tdg', 'h']	['cx'] ['h'] ['t'] ['tdg'] ['h', 't', 'h'] ['h', 'tdg', 'h'] ['h', 't', 'h', 't'] ['h', 't', 'h', 'tdg'] ['h', 'tdg', 'h', 't'] ['h', 'tdg', 'h', 'tdg'] ['t', 'h', 't', 'h'] ['t', 'h', 'tdg', 'h'] ['tdg', 'h', 't', 'h'] ['tdg', 'h', 'tdg', 'h'] ['t', 'h', 't', 'h', 'tdg'] ['t', 'h', 'tdg', 'h', 'tdg'] ['tdg', 'h', 't', 'h', 't'] ['tdg', 'h', 'tdg', 'h', 't']	['cx'] ['h'] ['t'] ['tdg'] ['h', 't', 'h'] ['h', 'tdg', 'h'] ['h', 't', 'h', 't'] ['h', 't', 'h', 'tdg'] ['h', 'tdg', 'h', 't'] ['h', 'tdg', 'h', 'tdg'] ['t', 'h', 't', 'h'] ['t', 'h', 'tdg', 'h'] ['tdg', 'h', 't', 'h'] ['tdg', 'h', 'tdg', 'h'] ['t', 'h', 't', 'h', 'tdg'] ['t', 'h', 'tdg', 'h', 'tdg'] ['tdg', 'h', 't', 'h', 't'] ['tdg', 'h', 'tdg', 'h', 't'] ['t', 'h', 't', 'h', 'tdg', 'tdg'] ['t', 'h', 'tdg', 'h', 'tdg', 'tdg'] ['t', 't', 'h', 't', 'h', 'tdg'] ['t', 't', 'h', 'tdg', 'h', 'tdg'] ['tdg', 'h', 't', 'h', 't', 't'] ['tdg', 'h', 'tdg', 'h', 't', 't'] ['tdg', 'tdg', 'h', 't', 'h', 't'] ['tdg', 'tdg', 'h', 'tdg', 'h', 't']

Analyzing average frequencies of usage of SK basis instructions leads to the identification of instructions that are used much more frequently than others, as depicted in Fig. 4.8 for the case of SK basis depth $d = 4$. This observation served as the inception for v3 of Huffman encoding, where we selected these frequently occurring SK basis instructions and included only them in the code dictionary to describe the circuit. These selected instructions for SK basis of depth $d = 4$, 5 and 6 are depicted in the table 5.5. The instructions are selected by observing average frequencies of usage over a dataset of 200 random unitaries. We observe that the gate sequences at lower depths also have a consistently

high usage at higher depths as well. The growing number of these abstractions with growing system size enables in capturing more complexity of the quantum circuits which is also observed in 5.4.

Similar approaches to discovering programming abstractions from program synthesis have been carried out in [78] by using deep reinforcement learning techniques to teach the compiler to synthesize unitaries and constantly updating the library of gates and using this library to solve similar synthesis problems.

Finding these abstractions not only streamlines the quantum programming process but also aids in developing new algorithms by reusing and combining these high-level constructs, much like forming new words in a language by understanding and applying common suffixes or prefixes.

6

Conclusion

Driven by both an investor's and a theorist's motivation, this project delivers 2 major contributions: An energy-efficient QISA and a practical estimate of the description complexity of quantum circuits. As a part of the optimal firmware design suite, this project is targeted to augment the compilation and control performance from an algorithmic point of view. To answer this, we turn to algorithmic information theory, or description complexity in particular. Based on this premise, we started with the research question:

> How can we compress the representation of decomposed quantum circuits?

The main contribution of this thesis is a generalized framework for the synthesis of quantum unitaries into a native set of gates and the subsequent representation of the decomposed circuit in terms of Huffman-encoded opcodes from the SK basis.

The background provides a detailed theoretical exploration of resource theory, algorithmic information, and complexity theory. It culminates in a concise table that aggregates various compound complexity measures, facilitating a clear comparison.

Chapter 3 deals broadly with the choice of the unitary matrix as the quantum description and methods for the decomposition of the unitary into a native set of gates. We begin with the lossless compression method bzip2 as our preliminary estimate of description complexity. The initial experiments on YAQQ serve as a sandbox for setting up a dataset of Harr-random unitaries and the performance of decomposition routines. The Solovay-Kitaev decomposition, one of the most important components of the project, was studied extensively, and the reconstruction of the decomposed circuit from the base-layer sequences from the SK basis forms the cornerstone of this project. A generalized version of SKD was developed to operate with any set of native gates input by the user. The routine was then scaled up to multi-qubit systems with the implementation of Quantum Shannon Decomposition integrated with SKD. A range of experiments were conducted to study the performance of decomposition routines for 1q, 2q, and 3q systems. These experiments also helped in the choice of optimal values of the parameters for SKD: depth $d = 5$ and degree of recursion $n = 4$, which will be used for the decomposition of benchmark circuits in the next chapter.

The foundation and implementation of Huffman coding are covered in Chapter 4. Three different versions of Huffman coding with an increasing level of sophistication are developed and presented with a comparative analysis of the average compression factor over datasets of random unitaries for different configurations of the SK decomposition. The average compression factor for Huffman v3 closely approaches Huffman v2 (the most optimal encoding in terms of compression) for higher degrees of recursion. Huffman v3, having an optimal compression performance and a much shorter code dictionary, is, therefore, the most optimal method developed in the thesis. The efficacy of the decomposition and encoding routine is then portrayed on a set of real benchmark circuits. We observe a consistent compression performance over a range of benchmarks, indicating that these encoding schemes generalize well over different quantum circuits. A subsequent lossless compression with bzip2 over

Huffman-encoded instruction streams further improves the compression and brings down the total program size.

In Chapter 5, we assess the methods and findings discussed in Chapters 3 and 4, taking a bird's-eye view of the quantum computation stack, with a particular focus on the QISA level and modern design strategies. We emphasize the crucial role of cryogenic control architecture in facilitating scalable, fully integrated quantum computation at qubit operation temperatures. We address the energy bottlenecks and constraints within these systems, proposing the use of encoded and compressed quantum instruction streams as a solution to reduce the heat load of transmission of instruction from RT to cryogenic temperature. Our discussion then shifts to the theoretical metrics demonstrated by our work. We argue that the Huffman-encoded representation of decomposed circuits provides a practical measure of quantum description complexity and observe that it is successful in capturing high-level quantum programming abstractions, offering valuable insights into the semantics of quantum algorithms and fostering the development of novel algorithms.

6.1. Outlook

Having successfully implemented the decomposition and encoding framework for quantum circuits, our focus now shifts towards exploring future avenues. This project was conceived within the context of a broader initiative aimed at optimal quantum firmware design alongside YAQQ and 'Energy Efficient Pulse Control'.

6.1.1. Implementation on Hardware

The compression experiments averaged over sets of samples 4.11, 4.12, 4.13 were conducted to obtain a fixed encoding (for a chosen depth of SKD) that performs consistently for different quantum circuits. This encoding scheme can be implemented on real hardware/ cryo-control processors to study the efficiency and processing heat load with the proposed encoded instruction streams. The SK-basis instruction stream offers a fewer number of operations as compared to the stream of gates. This can lead to a reduced heat load and a relaxation in the clock frequencies for control processors. Collaborations with research efforts on cryogenic control hardware will prove fruitful in studying these techniques with greater detail and help assess the ability to meet real-time control of quantum processors and an estimate of the processing energy efficiency of encoded quantum instruction streams.

6.1.2. Connection with YAQQ

The gateset `[h, t, tdg]` has been chosen consistently in the thesis for the Solovay-Kitaev decomposition. The decomposition and encoding framework is designed to work for any general gate set, and searching for an optimal gate set can lead to a more efficient and reliable decomposition. The YAQQ framework, which operates at a level just above DECQA in the optimal firmware suite, searches for an optimal gate set over an ensemble of random unitaries by minimizing a cost function that has the parameters fidelity, circuit depth, and novelty. The optimal gates found over 500 random unitaries are:

$$\mathrm{P1}_{\vec{a}}^{opt.500} = \begin{bmatrix} -0.99891178 + 0.j & -0.01683013 - 0.04349723j \\ -0.0406026 + 0.02294975j & +0.7722138 + 0.63364862j \end{bmatrix}$$

$$\mathrm{P1}_{\vec{b}}^{opt.500} = \begin{bmatrix} +0.54683177 + 0.j & -0.52979855 - 0.64829662j \\ +0.8231603 + 0.15291220j & +0.26287606 + +0.47950095j \end{bmatrix}$$

By adopting these gates as native gates for decomposing our target unitaries, we can achieve an improved decomposition in terms of fidelity and circuit depth.

6.1.3. Connection with Energy Efficient Pulse Control

As we discussed in 5.3, designing the QISA to be aware of the micro-architecture and control systems makes it more "comprehensive" [71] in describing the physical details of the program. An extension of this project in this direction would be to integrate the DECQA framework with a micro-architecture. This will help translate the compressed and encoded representation of decomposed quantum circuits to the pulse level.

On this aspect, connecting this framework with the 'Energy Efficient Pulse Control' framework which is envisioned to operate just below DECQA in the suite and deliver optimized pulse instructions for

our compressed circuit representations. This upgrade would complete the algorithm and information-theory-driven approach toward building an optimal quantum firmware.

6.1.4. Algorithmic Information Theory

The concept of description uncomplexity as a resource for performing computation is put forth in [22] and also proved in [79].

$$\text{Resource} = \Delta\mathcal{C} = (\mathcal{C}_{\text{max}} - \mathcal{C}) \tag{6.1}$$

In the above equation taken from [22], $\mathcal{C}$ is the complexity. Following these ideas, the algorithmic randomness for quantum circuits can be estimated from the incompressibility. The higher the compression factor, the higher the incompressibility and the higher the algorithmic randomness. The quantum circuits with low algorithmic randomness can be imagined to have a higher untapped resource for performing the target computation. This analogy can be used to prove quantum supremacy for circuits that have a high algorithmic randomness.

6.1.5. Improvements in Encoding and Decomposition

The encoding schemes proposed in this thesis are integrated with the Solovay-Kitaev decomposition. The composite instructions from the SK basis are chosen as fundamental units for defining the circuit. The scope of the encoding schemes can be expanded to other decomposition methods or reinforcement learning-based circuit compilation methods [78, 80]. Huffman coding can be employed on alternate decomposition methods by selecting relevant fundamental units that can be identified through a study of decomposed quantum circuits over sets of samples analogous to the experiments done in Chapter 4.

While scaling up the framework for multi-qubit systems, we implemented an additional qubit ID stream corresponding to the instruction stream, specifying the qubit to operate on. Currently, we are utilizing binary encoding for the qubit ID stream. As the system size increases, the number of bits per qubit ID and the total length of the qubit ID stream also scale up. We can implement Huffman coding on the qubit ID. This would help compress the qubit ID stream.

Another possible avenue would be to add the qubit ID to the gate/instruction. For example, `[h, t, h] q[0]` and `[h, t, h], q[1]` would be two different opcodes instead of counting them as the same instruction and specifying the qubit ID in a separate stream. The high-level quantum programming abstractions would be different in this case and may be useful in getting deeper insights that also describe/distinguish qubits in the register.

Bibliography

[1] A. M. Turing, *On computable numbers, with an application to the entscheidungsproblem,* Proceedings of the London Mathematical Society **s2-42**, 230 (1937), https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s2-42.1.230 .

[2] G. E. Moore, *Cramming more components onto integrated circuits,* Electronics **38**, 114 (1965).

[3] J. M. Shalf and R. Leland, *Computing beyond moore's law,* Computer **48**, 14 (2015).

[4] J. Preskill, *Quantum Computing in the NISQ era and beyond,* Quantum **2**, 79 (2018).

[5] A. Peruzzo, J. R. McClean, P. Shadbolt, M. Yung, X. Zhou, P. J. Love, A. Aspuru□Guzik, and J. L. O'Brien, *A variational eigenvalue solver on a photonic quantum processor,* Nature communications **5** (2014), 10.1038/ncomms5213.

[6] E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization algorithm,* arXiv (Cornell University) (2014).

[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[8] L. Susskind, *Horizons protect church-turing,* (2020), arXiv:2003.01807 [hep-th] .

[9] A. P. M. Place, L. V. H. Rodgers, P. Mundada, B. M. Smitham, M. Fitzpatrick, Z. Leng, A. Premkumar, J. Bryon, A. Vrajitoarea, S. Sussman, G. Cheng, T. Madhavan, H. K. Babla, X. H. Le, Y. Gang, B. Jäck, A. Gyenis, N. Yao, R. J. Cava, N. P. de Leon, and A. A. Houck, *New material platform for superconducting transmon qubits with coherence times exceeding 0.3 milliseconds,* Nature Communications **12**, 1779 (2021).

[10] F. Yan, P. Krantz, Y. Sung, M. Kjaergaard, D. L. Campbell, T. P. Orlando, S. Gustavsson, and W. D. Oliver, *Tunable coupling scheme for implementing high-fidelity two-qubit gates,* Phys. Rev. Appl. **10**, 054062 (2018).

[11] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, *Logical quantum processor based on reconfigurable atom arrays,* Nature **626**, 58 (2024).

[12] D. Litinski and N. Nickerson, *Active volume: An architecture for efficient fault-tolerant quantum computers with limited non-local connections,* (2022), arXiv:2211.15465 [quant-ph] .

[13] W. Cai, Y. Ma, W. Wang, C.-L. Zou, and L. Sun, *Bosonic quantum error correction codes in superconducting quantum circuits,* Fundamental Research **1**, 50–67 (2021).

[14] A. Cross, A. Javadi-Abhari, T. Alexander, N. De Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, *Openqasmnbsp;3: A broader and deeper quantum assembly language,* ACM Transactions on Quantum Computing **3** (2022), 10.1145/3505636.

[15] A. McCaskey and T. Nguyen, *A mlir dialect for quantum assembly languages,* in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2021) pp. 255–264.

[16] E. Charbon, F. Sebastiano, A. Vladimirescu, H. Homulle, S. Visser, L. Song, and R. M. Incandela, *Cryo-cmos for quantum computing,* in *2016 IEEE International Electron Devices Meeting (IEDM)* (2016) pp. 13.5.1–13.5.4.

[17] A. Butko, G. Michelogiannakis, S. Williams, C. Iancu, D. Donofrio, J. Shalf, J. Carter, and I. Siddiqi, *Understanding quantum control processor capabilities and limitations through circuit characterization,* in *2020 International Conference on Rebooting Computing (ICRC)* (2020) pp. 66–75.

[18] S. Mücke, R. Heese, S. Müller, M. Wolter, and N. Piatkowski, *Feature selection on quantum computers,* Quantum Machine Intelligence **5**, 11 (2023).

[19] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Quantum machine learning,* Nature **549**, 195 (2017).

[20] V. Dunjko, J. M. Taylor, and H. J. Briegel, *Quantum-enhanced machine learning,* Phys. Rev. Lett. **117**, 130501 (2016).

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

[22] A. R. Brown and L. Susskind, *Second law of quantum complexity,* Physical Review D **97**, 086015 (2018).

[23] W. H. Zurek, *Thermodynamic cost of computation, algorithmic complexity and the information metric,* **341**, 119 (1989).

[24] D. H. Wolpert and A. Kolchinsky, *Thermodynamics of computing with circuits,* New Journal of Physics **22**, 063047 (2020).

[25] A. Kolchinsky and D. H. Wolpert, *Thermodynamic costs of turing machines,* Physical Review Research **2**, 033312 (2020).

[26] J. Seward, *bzip2 and libbzip2,* avaliable at http://www.bzip.org (1996).

[27] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression,* IEEE Transactions on Information Theory **23**, 337 (1977).

[28] H. Zenil, S. Hernández-Orozco, N. A. Kiani, F. Soler-Toscano, A. Rueda-Toicen, and J. Tegnér, *A decomposition method for global evaluation of shannon entropy and local estimations of algorithmic complexity,* Entropy **20** (2018), 10.3390/e20080605.

[29] L. Mandelstam and I. Tamm, *The uncertainty relation between energy and time in non-relativistic quantum mechanics,* in *Selected Papers*, edited by B. M. Bolotovskii, V. Y. Frenkel, and R. Peierls (Springer Berlin Heidelberg, Berlin, Heidelberg, 1991) pp. 115–123.

[30] N. Margolus and L. B. Levitin, *The maximum speed of dynamical evolution,* Physica D: Nonlinear Phenomena **120**, 188 (1998).

[31] J. D. Bekenstein, *Universal upper bound on the entropy-to-energy ratio for bounded systems,* Phys. Rev. D **23**, 287 (1981).

[32] A. N. Kolmogorov, *Three approaches to the quantitative definition of information *,* International Journal of Computer Mathematics **2**, 157 (1968), https://doi.org/10.1080/00207166808803030 .

[33] P. M. Vitányi, *Quantum kolmogorov complexity based on classical descriptions,* IEEE Transactions on Information Theory **47**, 2464 (2001).

[34] R. Solomonoff, *A formal theory of inductive inference: Parts i and ii,* Information and Control **7**, 1 (1964).

[35] G. J. Chaitin, *On the length of programs for computing finite binary sequences,* J. ACM **13**, 547–569 (1966).

[36] M. Hutter, *Algorithmic information theory: a brief non-technical guide to the field,* arXiv preprint cs/0703024 (2007).

[37] D. Deutsch, *Quantum theory, the church–turing principle and the universal quantum computer,* Proc. R. Soc. Lond. A **400**, 97 (1985).

[38] A. Berthiaume, W. Van Dam, and S. Laplante, *Quantum kolmogorov complexity,* Journal of Computer and System Sciences **63**, 201 (2001).

[39] M. R. Dowling and M. A. Nielsen, *The geometry of quantum computation,* (2006), arXiv:quant-ph/0701004 [quant-ph] .

[40] L. A. Levin, *Universal sequential search problems,* Problemy peredachi informatsii **9**, 115 (1973).

[41] M. Li and P. Vitányi, *An introduction to kolmogorov complexity and its applications,* (Springer Cham, 2019) Chap. 7. Resource-Bounded Complexity.

[42] C. H. Bennett, *Logical depth and physical complexity,* in *The universal Turing machine, a half century survey*, edited by R. Herken (Oxford University Press, 1988) pp. 227–257.

[43] J. Schmidhuber, *The speed prior: A new simplicity measure yielding near-optimal computable predictions,* in *Computational Learning Theory*, edited by J. Kivinen and R. H. Sloan (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002) pp. 216–228.

[44] J. Baez and M. Stay, *Algorithmic thermodynamics,* Mathematical Structures in Computer Science **22**, 771 (2012).

[45] A. Kolchinsky, *Generalized zurek's bound on the cost of an individual classical or quantum computation,* arXiv preprint arXiv:2301.06838 (2023).

[46] P. M. B. Vitanyi, F. J. Balbach, R. L. Cilibrasi, and M. Li, *Normalized information distance,* (2008), arXiv:0809.2553 [cs.IR] .

[47] R. L. Cilibrasi and P. M. Vitanyi, *The google similarity distance,* IEEE Transactions on Knowledge and Data Engineering **19**, 370 (2007).

[48] T. Rado, *On non-computable functions,* Bell System Technical Journal **41**, 877 (1962).

[49] A. H. Brady, *The determination of the value of rado's noncomputable function σ (��) for four-state turing machines,* Mathematics of Computation **40**, 647 (1983).

[50] S. K. Menon, *Studying the code compression design space – a synthesis approach,* Journal of Systems Architecture **60**, 179 (2014).

[51] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović, *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0*, Tech. Rep. UCB/EECS-2014-54 (2014).

[52] A. Waterman, *Improving energy efficiency and reducing code size with risc-v compressed,* Master's thesis (2011).

[53] M. Perotti, P. D. Schiavone, G. Tagliavini, D. Rossi, T. Kurd, M. Hill, L. Yingying, and L. Benini, en*Hw/sw approaches for risc-v code size reduction,* (2020-05), workshop on Computer Architecture Research with RISC-V (CARRV 2020); Conference Location: online; Conference Date: May 30, 2020; Due to the Coronavirus (COVID-19) the conference was conducted virtually.

[54] H. Homulle, S. Visser, B. Patra, G. Ferrari, E. Prati, F. Sebastiano, and E. Charbon, *A reconfigurable cryogenic platform for the classical control of quantum processors,* Review of Scientific Instruments **88**, 045103 (2017), https://pubs.aip.org/aip/rsi/article-pdf/doi/10.1063/1.4979611/15975928/045103_1_online.pdf .

[55] H. Zenil, *Compression is comprehension, and the unreasonable effectiveness of digital computation in the natural world,* (2021), arXiv:1904.10258 [cs.IT] .

[56] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, *Open quantum assembly language,* (2017), arXiv:1707.03429 [quant-ph] .

[57] N. Khammassi, G. G. Guerreschi, I. Ashraf, J. W. Hogaboam, C. G. Almudever, and K. Bertels, *cqasm v1.0: Towards a common quantum assembly language,* (2018), arXiv:1805.09607 [quant-ph] .

[58] X. Fu, L. Riesebos, M. A. Rol, J. van Straten, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, V. Newsum, K. K. L. Loh, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels, *eqasm: An executable quantum instruction set architecture,* in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2019) pp. 224–237.

[59] A. Sarkar and A. Kundu, *Yaqq: Yet another quantum quantizer,* https://github.com/Advanced-Research-Centre/YAQQ (2023).

[60] M. Burrows, *A block-sorting lossless data compression algorithm,* SRC Research Report **124** (1994).

[61] D. A. Huffman, *A method for the construction of minimum-redundancy codes,* Proceedings of the IRE **40**, 1098 (1952).

[62] C. M. Dawson and M. A. Nielsen, *The solovay-kitaev algorithm,* Quantum Info. Comput. **6**, 81–95 (2006).

[63] V. Shende, S. Bullock, and I. Markov, *Synthesis of quantum-logic circuits,* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**, 1000 (2006).

[64] D. Salomon, *Variable-length Codes for Data Compression* (Springer London, 2007).

[65] N. Quetschlich, L. Burgholzer, and R. Wille, *MQT Bench: Benchmarking software and design automation tools for quantum computing,* Quantum (2023), MQT Bench is available at https://www.cda.cit.tum.de/mqtbench/.

[66] C. Developers, *Cirq,* (2023).

[67] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. D. Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran, *Pennylane: Automatic differentiation of hybrid quantum-classical computations,* (2022), arXiv:1811.04968 [quant-ph] .

[68] K. Svore, A. Geller, M. Troyer, J. Azariah, C. Granade, B. Heim, V. Kliuchnikov, M. Mykhailova, A. Paz, and M. Roetteler, *Q#: Enabling scalable quantum computing and development with a high-level dsl,* in *Proceedings of the Real World Domain Specific Languages Workshop 2018*, RWDSL2018 (ACM, 2018).

[69] X. Xue, B. Patra, J. P. G. van Dijk, N. Samkharadze, S. Subramanian, A. Corna, B. Paquelet Wuetz, C. Jeon, F. Sheikh, E. Juarez-Hernandez, B. P. Esparza, H. Rampurawala, B. Carlton, S. Ravikumar, C. Nieva, S. Kim, H.-J. Lee, A. Sammak, G. Scappucci, M. Veldhorst, F. Sebastiano, M. Babaie, S. Pellerano, E. Charbon, and L. M. K. Vandersypen, *Cmos-based cryogenic control of silicon quantum circuits,* Nature **593**, 205 (2021).

[70] P. 't Hart, English*Cryogenic CMOS Characterization for Quantum Computer Applications*, Dissertation (tu delft), Delft University of Technology (2022).

[71] F. T. Chong, D. Franklin, and M. Martonosi, *Programming languages and compiler design for realistic quantum hardware,* Nature **549**, 180 (2017).

[72] R. S. Smith, M. J. Curtis, and W. J. Zeng, *A practical quantum instruction set architecture,* (2017), arXiv:1608.03355 [quant-ph] .

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

[74] H. Wu, W. Fu, D. Wu, and M. Feng, *2.9 K VCSEL demonstrates 100 Gbps PAM-4 optical data transmission,* Applied Physics Letters **121**, 011102 (2022), https://pubs.aip.org/aip/apl/article-pdf/doi/10.1063/5.0095321/16449248/011102_1_online.pdf .

[75] P. Pintus, A. Singh, W. Xie, L. Ranzani, M. V. Gustafsson, M. A. Tran, C. Xiang, J. Peters, J. E. Bowers, and M. Soltani, *Ultralow voltage, high-speed, and energy-efficient cryogenic electro-optic modulator,* Optica **9**, 1176 (2022).

[76] J. Wang, M. I. Ibrahim, I. B. Harris, N. M. Monroe, M. I. Wasiq Khan, X. Yi, D. R. Englund, and R. Han, *34.1 thz cryo-cmos backscatter transceiver: A contactless 4 kelvin-300 kelvin data interface,* in *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023) pp. 504–506.

[77] N. Fakkel, M. Mortazavi, R. Overwater, F. Sebastiano, and M. Babaie, *A cryo-cmos dac-based 40 gb/s pam4 wireline transmitter for quantum computing applications,* in *2023 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)* (2023) pp. 257–260.

[78] L. Sarra, K. Ellis, and F. Marquardt, *Discovering quantum circuit components with program synthesis,* (2023), arXiv:2305.01707 [quant-ph] .

[79] N. Y. Halpern, N. B. Kothakonda, J. Haferkamp, A. Munson, J. Eisert, and P. Faist, *Resource theory of quantum uncomplexity,* Physical Review A **106**, 062417 (2022).

[80] L. Moro, M. G. A. Paris, M. Restelli, and E. Prati, *Quantum compiling by deep reinforcement learning,* Communications Physics **4**, 178 (2021).

Appendix

List of benchmark circuits taken from the MQTBench[65] library. The suffix `_indep_qiskit` denotes that the circuits are described at a target-independent level using qiskit as the compiler. The chosen benchmarks are all scalable algorithms with system sizes in the range 2-6.

Benchmark Codename	Circuit Name and Size
ae_indep_qiskit_2	Amplitude Estimation, 2 qubit
ae_indep_qiskit_3	Amplitude Estimation, 3 qubit
ae_indep_qiskit_4	Amplitude Estimation, 4 qubit
ae_indep_qiskit_5	Amplitude Estimation, 5 qubit
ae_indep_qiskit_6	Amplitude Estimation, 6 qubit
dj_indep_qiskit_2	Deutsch Josza, 2 qubit
dj_indep_qiskit_3	Deutsch Josza, 3 qubit
dj_indep_qiskit_4	Deutsch Josza, 4 qubit
dj_indep_qiskit_5	Deutsch Josza, 5 qubit
dj_indep_qiskit_6	Deutsch Josza, 6 qubit
ghz_indep_qiskit_2	GHZ State, 2 qubit
ghz_indep_qiskit_3	GHZ State, 3 qubit
ghz_indep_qiskit_4	GHZ State, 4 qubit
ghz_indep_qiskit_5	GHZ State, 5 qubit
ghz_indep_qiskit_6	GHZ State, 6 qubit
graphstate_indep_qiskit_3	Graph State, 3 qubit
graphstate_indep_qiskit_4	Graph State, 4 qubit
graphstate_indep_qiskit_5	Graph State, 5 qubit
graphstate_indep_qiskit_6	Graph State, 6 qubit
grover-noancilla_indep_qiskit_2	Grover's Algorithm (no ancilla), 2 qubit
grover-noancilla_indep_qiskit_3	Grover's Algorithm (no ancilla), 3 qubit
grover-noancilla_indep_qiskit_4	Grover's Algorithm (no ancilla), 4 qubit
grover-noancilla_indep_qiskit_5	Grover's Algorithm (no ancilla), 5 qubit
grover-noancilla_indep_qiskit_6	Grover's Algorithm (no ancilla), 6 qubit
grover-v-chain_indep_qiskit_2	Grover's Algorithm (v-chain), 2 qubit
grover-v-chain_indep_qiskit_3	Grover's Algorithm (v-chain), 3 qubit
grover-v-chain_indep_qiskit_4	Grover's Algorithm (v-chain), 4 qubit
grover-v-chain_indep_qiskit_5	Grover's Algorithm (v-chain), 5 qubit
portfolioqaoa_indep_qiskit_3	Portfolio Optimization with QAOA, 3 qubit
portfolioqaoa_indep_qiskit_4	Portfolio Optimization with QAOA, 4 qubit
portfolioqaoa_indep_qiskit_5	Portfolio Optimization with QAOA, 5 qubit
portfolioqaoa_indep_qiskit_6	Portfolio Optimization with QAOA, 6 qubit
portfoliovqe_indep_qiskit_3	Portfolio Optimization with VQE, 3 qubit
portfoliovqe_indep_qiskit_4	Portfolio Optimization with VQE, 4 qubit
portfoliovqe_indep_qiskit_5	Portfolio Optimization with VQE, 5 qubit
portfoliovqe_indep_qiskit_6	Portfolio Optimization with VQE, 6 qubit

qaoa_indep_qiskit_3	Quantum Approximation Optimization Algorithm (QAOA), 3 qubit
qaoa_indep_qiskit_4	Quantum Approximation Optimization Algorithm (QAOA), 4 qubit
qaoa_indep_qiskit_5	Quantum Approximation Optimization Algorithm (QAOA), 5 qubit
qaoa_indep_qiskit_6	Quantum Approximation Optimization Algorithm (QAOA), 6 qubit
qft_indep_qiskit_2	Quantum Fourier Transformation (QFT), 2 qubit
qft_indep_qiskit_3	Quantum Fourier Transformation (QFT), 3 qubit
qft_indep_qiskit_4	Quantum Fourier Transformation (QFT), 4 qubit
qft_indep_qiskit_5	Quantum Fourier Transformation (QFT), 5 qubit
qft_indep_qiskit_6	Quantum Fourier Transformation (QFT), 6 qubit
qftentangled_indep_qiskit_2	Entangled QFT, 2 qubit
qftentangled_indep_qiskit_3	Entangled QFT, 3 qubit
qftentangled_indep_qiskit_4	Entangled QFT, 4 qubit
qftentangled_indep_qiskit_5	Entangled QFT, 5 qubit
qftentangled_indep_qiskit_6	Entangled QFT, 6 qubit
qnn_indep_qiskit_2	Quantum Neural Network (QNN), 2 qubit
qnn_indep_qiskit_3	Quantum Neural Network (QNN), 3 qubit
qnn_indep_qiskit_4	Quantum Neural Network (QNN), 4 qubit
qnn_indep_qiskit_5	Quantum Neural Network (QNN), 5 qubit
qnn_indep_qiskit_6	Quantum Neural Network (QNN), 6 qubit
qpeexact_indep_qiskit_2	Quantum Phase Estimation (QPE) exact, 2 qubit
qpeexact_indep_qiskit_3	Quantum Phase Estimation (QPE) exact, 3 qubit
qpeexact_indep_qiskit_4	Quantum Phase Estimation (QPE) exact, 4 qubit
qpeexact_indep_qiskit_5	Quantum Phase Estimation (QPE) exact, 5 qubit
qpeexact_indep_qiskit_6	Quantum Phase Estimation (QPE) exact, 6 qubit
qpeinexact_indep_qiskit_2	Quantum Phase Estimation (QPE) inexact, 2 qubit
qpeinexact_indep_qiskit_3	Quantum Phase Estimation (QPE) inexact, 3 qubit
qpeinexact_indep_qiskit_4	Quantum Phase Estimation (QPE) inexact, 4 qubit
qpeinexact_indep_qiskit_5	Quantum Phase Estimation (QPE) inexact, 5 qubit
qpeinexact_indep_qiskit_6	Quantum Phase Estimation (QPE) inexact, 6 qubit
qwalk-noancilla_indep_qiskit_3	Quantum Walk (no ancilla), 3 qubit
qwalk-noancilla_indep_qiskit_4	Quantum Walk (no ancilla), 4 qubit
qwalk-noancilla_indep_qiskit_5	Quantum Walk (no ancilla), 5 qubit
qwalk-noancilla_indep_qiskit_6	Quantum Walk (no ancilla), 6 qubit
qwalk-v-chain_indep_qiskit_3	Quantum Walk (v-chain), 3 qubit
qwalk-v-chain_indep_qiskit_5	Quantum Walk (v-chain), 5 qubit
vqe_indep_qiskit_3	Variational Quantum Eigensolver (VQE), 3 qubit
vqe_indep_qiskit_5	Variational Quantum Eigensolver (VQE), 5 qubit
vqe_indep_qiskit_6	Variational Quantum Eigensolver (VQE), 6 qubit
wstate_indep_qiskit_2	W-State, 2 qubit
wstate_indep_qiskit_3	W-State, 3 qubit
wstate_indep_qiskit_4	W-State, 4 qubit
wstate_indep_qiskit_5	W-State, 5 qubit
wstate_indep_qiskit_6	W-State, 6 qubit

Table 6.1: List of benchmarks selected for the experiments in the thesis.