IET Quantum Communication

WILEY

The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

Superposition-Based Abstractions for Quantum Data Encoding Verification

Arun Govindankutty | Sudarshan K. Srinivasan

Department of Electrical and Computer Engineering, North Dakota State University Fargo, Fargo, North Dakota, USA

Correspondence: Sudarshan K. Srinivasan (sudarshan.srinivasan@ndsu.edu)

Received: 4 October 2024 | **Revised:** 10 February 2025 | **Accepted:** 20 February 2025

Handling Editor: Khalid Abualsaud

Funding: This paper is based upon work supported by the National Science Foundation under Grant No. 2513276.

Keywords: quantum computing | quantum computing techniques | quantum gates | quantum information

ABSTRACT

Many quantum algorithms operate on classical data, by first encoding classical data into the quantum domain using quantum data encoding circuits. To be effective for large data sets, encoding circuits that operate on large data sets are required. However, as the size of the data sets increases, the encoding circuits quickly become large, complex and error prone. Errors in the encoding circuit will provide incorrect inputs to quantum algorithms, making them ineffective. To address this problem, a formal method is proposed for verification of encoding circuits. The key idea to address scalability is the use of abstractions that reduce the verification problem to bit-vector space. The major outcome of this work is that using this approach, the authors have been able to verify encoding circuits with up to 8191 qubits with very low memory (85 MB) and time (0.29s), demonstrating that the proposed approach can easily be employed to verify even much larger encoding circuits. The results are very significant because, traditional verification approaches that rely on modelling quantum circuits in Hilbert space have only demonstrated verification scalability up to 250 qubits. Also, this is the first approach to tackle the verification of quantum encoding circuits.

1 | Introduction

Classical-quantum hybrid computing is pervasive and is able to tackle hard problems in various fields. Chemical and pharmaceutical industry, finance, satellite communication, physics, cyber-security, logistics optimisation, database search, and machine learning [1, 2] are only a few areas to quote where quantum computing and algorithms have proven their worth working in a hybrid mode with classical computing. Niche technology domains such as wireless network optimisation [3], large language model processing [4], computer graphics [5], semiconductor defect detection [6], and image processing also benefit extensively from quantum computing and algorithms. Quantum computing continues to enhance technological capabilities by solving real-world challenges through integration with classical computing systems. Quantum support vector machines (QVSM) show superior efficiency compared to traditional machine learning methods in medical field for predicting conditions such as breast cancer [7], and thyroid anomalies [8]. Data security, quantum communication, simulation, product development [9], internet of things [10], quantum internet and 6G [11], and software defined networks [12] are a few critical scenarios, where quantum computing techniques and algorithms are showing potential in hybrid mode with classical data being transformed into quantum domain for computational advantage. Verification of the encoded data to ensure the correctness of computation is thus vital in real-life scenarios and is often overlooked aspect, which is the core motivation of this work.

Cortese and Braje [13] have patented a technique that efficiently encodes classical data into entangled quantum basis states. This

method is notably versatile, capable of encoding 2^n classical bits into $n + 1$ qubits. However, the complexity of the circuit is exponential in n. Given that the accuracy of computation results is contingent upon the correctness of the input data, it is essential to verify the data encoding circuit utilised.

Verifying quantum circuits poses significant challenges, as these circuits operate within Hilbert space (a complex vector space), which is the framework used to model qubit states. Superimposed qubit states pose another challenge in verifying quantum circuits. Formal verification utilises mathematical reasoning and proofs to validate designs against their specifications. This approach is particularly effective in identifying hard-to-detect corner case bugs and can provide a high degree of assurance regarding the correctness of the design [14]. Formal verification can reduce the quantum circuit verification problem from Hilbert space to bit-vector space by using abstractions. This enables scalability even when encoding circuits become large as the input data set increases. Domain- and problem-specific verification approaches have been extensively explored to achieve scalability in quantum circuit verification. Govindankutty et al. [15] present a scalable, domain-specific method for verifying the quantum Fourier transform. Sitou Campbell and Srinivasan [16] describe a verification technique for cyclic quantum walks using a novel abstraction that decouples the verification of the circuit's functionality from the verification of its superposition behaviour (we call superposition abstraction). In this paper, we extend the concept of superposition abstraction with new abstractions and methodologies to tackle the challenge of verifying circuits used for encoding classical data into quantum states, which we refer to as quantum encoding circuits.

Contributions: Our major contributions are summarised below:

1. Application of superposition abstraction to decouple superposition behaviour of quantum data encoding circuits.
2. Quantum gate abstractions and methodology to reduce quantum data encoding circuit verification from Hilbert-space to bit-vector space.
3. Correctness properties for the quantum data encoding circuit.

The remainder of the paper is organised as follows: Section 2 explores existing formal verification techniques for quantum algorithms and circuits. Section 3 presents the background on quantum encoding circuits and motivation for this work. Section 4 outlines the key abstractions and property definitions used in our approach. Section 5 details the working principles and correctness of the abstractions. Section 6 reports the experimental verification results. Finally, Section 7 concludes the paper.

2 | Related Works

Formal verification of quantum algorithms and circuits is an active area of research, with approaches focusing on ensuring algorithm correctness. In this section, we review related works and highlight the differences with our approach. Although previous methods primarily concentrate on verifying algorithms, none directly address the critical problem of quantum data encoding. To the best of our knowledge, our work is the first to tackle this challenge, ensuring that classical data is correctly encoded into quantum states. This is a crucial step for reliable quantum computation, as the accuracy of quantum algorithms depends on the fidelity of the initial data encoding process. By focusing on this often-overlooked aspect of quantum circuit verification, our approach fills a key gap in the field. Another notable characteristic of our approach is reducing the verification problem from Hilbert space to bit-vector space, significantly enhancing scalability. This transformation enables the application of the method to large-scale quantum systems.

Algorithm-specific verification methods have been explored to increase efficiency and scalability of verification. Govindankutty et al. [15] use the idea of abstracting the rotational impact of quantum gates to verify circuits that implement the Quantum Fourier transform (QFT). The method is specific to QFT. Sitou Campbell and Srinivasan [16] introduced the idea of superposition abstraction to verify quantum walk circuits. Ours is a continuation in this line of work. We extend the superposition abstraction approach further and apply it to the verification of quantum encoding circuits.

Amy [17, 18] proposes a verification method utilising reduction techniques and rewrite rules, where quantum gates are modelled using complex path-sum models. The Haskell theorem prover is employed to perform reductions, and the rewrite rules simplify the circuit into a normal form, which is then verified. However, the method relies on dyadic arithmetic, which leads to integer overflow for larger quantum circuits with the proposed implementation. The largest circuit verified using this approach contains 96 qubits, whereas our method successfully verifies circuits with up to 8191 input qubits.

Gay et al. [19] proposed a property-checking method for quantum cryptography systems. This approach allows for the modelling and verification of quantum security protocols expressible in the quantum stabiliser formalism [20]. The properties of the protocol are expressed using Quantum Computational Tree Logic (QCTL) [21]. They demonstrate the verification of a protocol with single qubit transfer (quantum coin tossing). Neither scalability, nor verification of quantum encoding is demonstrated. Similarly, Seiter et al. [22] also propose a property-checking method utilising quantum multiple-valued decision diagrams. They demonstrate verifying Grover's search circuit with a maximum of 10 qubits. Both of these methods rely on the temporal evolution of quantum systems, requiring the evaluation of all possible quantum states and outcomes. This approach significantly limits scalability and execution time efficiency compared to our method as we abstract the verification problem to bit-vector space.

Equivalence checking methods for quantum circuit verification are proposed by Burgholzer and Wille [23] and Yamashita and Markov [24]. Equivalence checking is applicable in the context where a design already exists that is known to be verified/trusted. Optimisations/refinements of the design can then be

checked against the original. The encoding circuits we consider are designed from scratch and therefore equivalence checking is not appropriate here. A second issue is that, when the problem is not reducible to binary states, they rely on a hybrid approach, solving the problem in Hilbert space [25]. Our approach does not need a reference model, and we reduce the problem completely to binary space using abstractions. Therefore, our approach can handle circuits with a large amount of qubits compared to the maximum of 35 and 128 qubits in the above works.

Hong et al. [26] propose an approximate equivalence checking method, which focuses on analysing the impact of noise in quantum circuits, aiming to improve functional accuracy in the Noisy Intermediate-Scale Quantum (NISQ) era. By employing tensor network contraction, they compute the distance between the output states of the ideal circuit and its noisy counterpart, followed by calculating Jamiolkowski fidelity (average-case error measure). If the fidelity exceeds a predetermined threshold, the circuits are considered equivalent. This approach, while effective in evaluating quantum circuits under noisy conditions, does not address the correctness of the algorithm's implementation. Additionally, the largest circuit considered in their method contains only 16 qubits, significantly smaller than the circuits verified in our approach.

Beillahi et al. [27] and Mahmoud et al. [28] propose theorem proving approaches using higher-order logic (HOL). They formalise specific quantum gates such as controlled-phase (CZ) and controlled-not (C-NOT) by modelling optical gate primitives. Their work verifies specific individual quantum gates and Shor's algorithm for factorising integer 15 which has only 4 input qubits. Scalability of the method is not established. Liu et al. [29] formalise Quantum Hoare Logic using the Isabelle-HOL prover. They use the framework to prove the correctness of the Grover's search algorithm. Their proof needed 5 months of manual effort. In contrast, our approach is fully automatic. They have also not demonstrated the use of their framework for the verification of encoding circuits.

To the best of our knowledge, this is the first work to address the verification of quantum data encoding as previous methods focus on verifying algorithms. Table 1 highlights a comparison between our method and existing methodologies, showcasing the highest number of qubits benchmarked successfully. This elucidates the superior scalability of our approach.

3 | Background

Quantum computers are physical machines that employ and utilise the laws of quantum mechanics for computation and solving problems. Quantum circuit model is one of the widely used computation models in quantum information processing [30] and we also use the same in this work. A detailed description of quantum circuit model can be found in refs. [31–33].

3.1 | Quantum Encoding

Quantum encoding is the technique for converting classical data bits into qubits using a set of basis vectors, so that quantum algorithms can be effectively applied for computations. Several methods have been proposed to achieve this purpose, including basis encoding, amplitude encoding, and angle encoding [34]. Rath and Date [35] provide a comparison of various data encoding schemes for machine learning applications. One of their significant conclusions is that the use of Basis encoding is most often superior to classical encoding of data. Liu et al. [36] have shown that reducing the number of qubits and the depth of the quantum encoding circuit significantly improves the performance of quantum algorithms. Cortese and Braje [13] provide a state-of-the-art patented encoding method, which does logarithmic compression of qubits at the output, significantly reducing the number of qubits for processing. However, this logarithmic compression comes at the cost of circuit complexity and vulnerability to errors. Therefore, verification methods that can guarantee encoding correctness can enable the use of this method and significantly improve the performance of a large class of quantum algorithms.

The quantum encoding circuit consists of Hadamard gates (H-gates) and controlled-swap gates (C-Swap gates) Figure 1. The H-gate introduces equal superposition of states to the qubit and is defined below [31]:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The C-Swap gate takes two data qubits and one control qubit as input. The data qubits are swapped if the control qubit is in state $|1\rangle$. Otherwise, the input qubits are not swapped. The C-swap gate is defined below [31]:

TABLE 1 | Benchmark comparison.

Related work	Benchmark program verified	Qubits handled successfully
Amy [17]	GF (2^{32})-Mult	96
Gay et al. [19]	Quantum coin flipping	—
Seiter et al. [22]	Grover10	10
Burgholzer and Wille [23]	c2_182 of RevLib benchmark	35
Yamashita and Markov [24]	Modular multiplication	258
Hong et al. [26]	bv16	16
Beillahi et al. [27]	Shor	4
This work	Quantum data encoding	**8191**

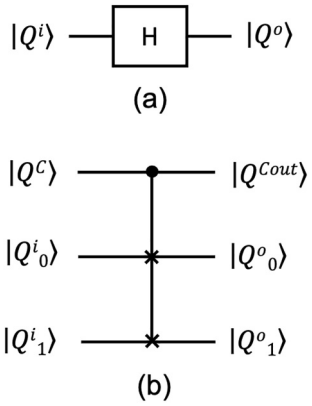

FIGURE 1 | (a) Hadamard gate (b) Controlled-Swap (C-Swap) gate.

$$\text{C-Swap} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The circuit encodes $p = 2^n$ classical bits to a superposition of one data qubit and n control qubits. The circuit for quantum encoding of 4 classical bits is shown in Figure 2a. Here, $|b_i\rangle$ represents the classical bit value in quantum state. That is, if classical bit has value 0, then $|b_i\rangle = |0\rangle$ and if the value is 1, then $|b_i\rangle = |1\rangle$. The circuit uses 4 data qubits Q_3^D, Q_2^D, Q_1^D and Q_0^D, which are mapped to classical bits b_3, b_2, b_1 and b_0, respectively. The circuit has 3 control qubits Q_2^C, Q_1^C and Q_0^C. "H" represents the H-gate. "." represents the connection of the control qubit line of the C-Swap gate and "x" represents the corresponding qubit connections that will be swapped. Initially, in the first stage of the circuit, an H-gate is applied to each of the control qubits, bringing them into an equal superposition of the basis states. These qubits are then used as control inputs to the C-Swap gates in the second stage, where consecutive data bits are swapped based on the value of the corresponding control qubit. Specifically, b_0 and b_1 are swapped based on the value of Q_1^C, b_2 and b_3 based on Q_2^C. This controlled swap process continues through subsequent stages.

At the output, one control qubit and three data qubits are discarded. The output state $|Q_0^{C_{out}} Q_1^{C_{out}} Q_0^{D_{out}}\rangle$ is represented by the mathematical equation given below:

$$\begin{aligned} |Q_0^{C_{out}} Q_1^{C_{out}} Q_0^{D_{out}}\rangle = \; & |00\rangle \otimes |b_0\rangle + |01\rangle \otimes |b_1\rangle \\ & + |10\rangle \otimes |b_2\rangle + |11\rangle \otimes |b_3\rangle \end{aligned} \tag{1}$$

In the above Equation, $\otimes$ denotes the tensor product. The circuit is synthesising the function of a p to 1 multiplexer. All the classical data bits are encoded into $Q_0^{D_{out}}$ in a superposition. The control qubits $Q_0^{C_{out}}$ and $Q_1^{C_{out}}$ act as the multiplexer selectors. To elaborate, when $|Q_0^{C_{out}} Q_1^{C_{out}}\rangle = |00\rangle$, then $|Q_0^{D_{out}}\rangle = |b_0\rangle$. When $|Q_0^{C_{out}} Q_1^{C_{out}}\rangle = |01\rangle$, then $|Q_0^{D_{out}}\rangle = |b_1\rangle$ and so on.

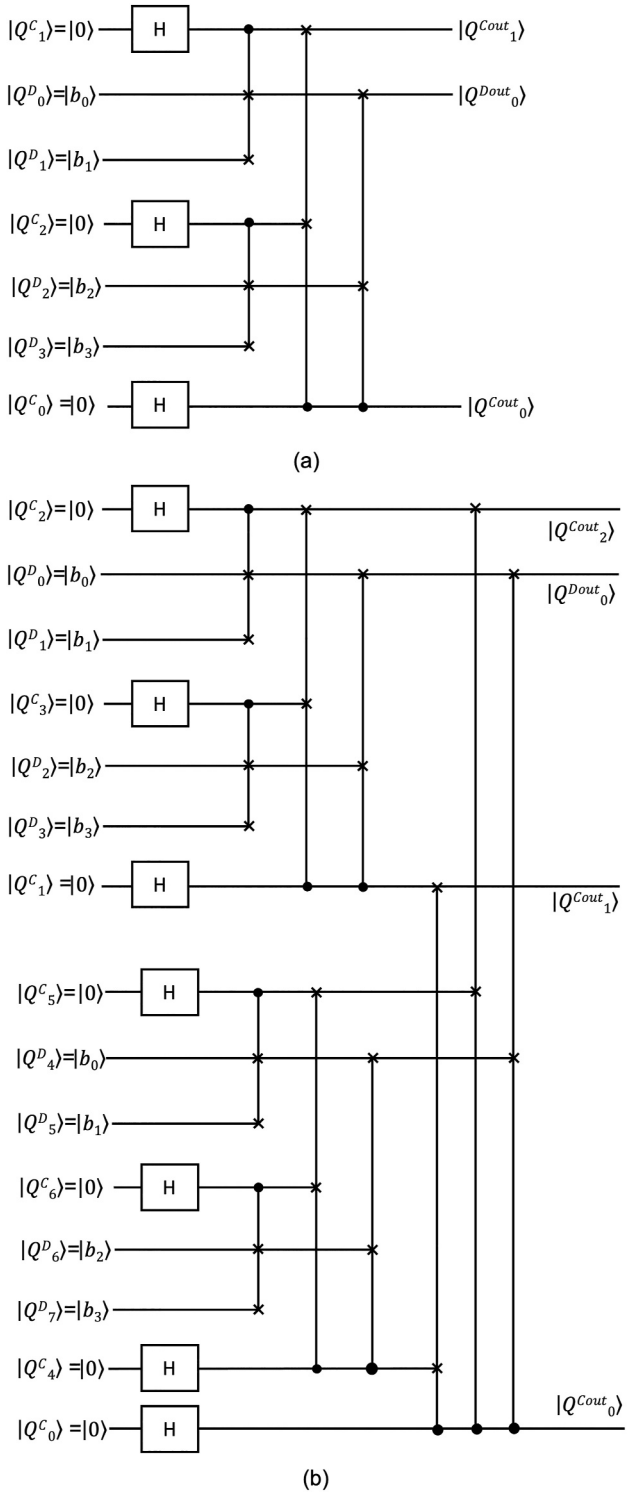

FIGURE 2 | (a) 4-bit to 3-qubit quantum encoding circuit. [13]. (b) An 8-bit to 4-qubit quantum encoding circuit.

A general p-qubit quantum encoding circuit for this technique will have p data qubits and $p - 1$ control qubits. At the output, all the data will be encoded using one data qubit and n control qubits $|Q_0^{C_{out}} Q_1^{C_{out}} \ldots Q_{n-1}^{C_{out}} Q_0^{D_{out}}\rangle$. The output equation is given below:

$$|Q_0^{C_{out}}\, Q_1^{C_{out}} ... Q_{n-1}^{C_{out}} Q_0^{D_{out}}\rangle = \; |00...0\rangle_n \otimes |b_0\rangle + |00...01\rangle_n \otimes |b_1\rangle \; + |00...10\rangle_n \otimes |b_2\rangle \; + \cdots + |11...11\rangle_n \otimes |b_{p-1}\rangle \quad (2)$$

Larger circuits are obtained by replicating the "4-bit to 3-qubit" circuit with additional control qubits. For example, the "8-bit to 4-qubit" circuit (shown in Figure 2b) will have two copies of the "4-bit to 3-qubit" circuit and an additional control qubit. The "16-bit to 5-qubit" circuit will have two copies of the "8-bit to 4-qubit" circuit with an additional control qubit and so on.

4 | Verification Methodology

This section details the proposed abstractions and properties used to verify quantum encoding circuits.

4.1 | Abstractions

A qubit state is represented as follows:

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers, and $|0\rangle$ and $|1\rangle$ are the computational basis states.

Quantum gates act linearly on their inputs. This means that the action of a gate on a qubit state is determined by how it acts on the computational basis states $|0\rangle$ and $|1\rangle$. In other words, if we know how a gate acts on a basis state, we can use the linearity of quantum mechanics to determine how it acts on any superposition of states.

Therefore, to fully characterise a gate, it is sufficient to specify its behaviour on all possible computational basis inputs. Computational basis states are those of the form $|q_A q_B q_C\rangle$, where each q_A, q_B, and q_C takes one of the values $|0\rangle$ or $|1\rangle$ for eample, consider the C-Swap gate, which operates on three qubits with inputs q_A, q_B, and q_C. The C-Swap gate swaps the states of qubits q_B and q_C if q_A is in the state $|1\rangle$, and leaves them unchanged if q_A is in the state $|0\rangle$. The computational basis states are given explicitly in Table 2.

TABLE 2 | C-Swap gate with abstract qubits.

q_A	q_B	q_C
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$

Note that it is sufficient to specify the behaviour of the gate on only the possible combinations of computational basis states (i.e., the qubit states where either $\alpha = 0$ and $\beta = 1$, or $\alpha = 1$ and $\beta = 0$) of the inputs.

The behaviour of the gate on noncomputational basis states (or superposition of computational basis states), that is, states where both α and β are nonzero, does not need to be explicitly specified. For example, consider a state such that $\alpha = \frac{\sqrt{3}}{2}$ and $\beta = \frac{1}{2}$. This is a superposition state where neither of the coefficients α or β is zero, and the state is not a simple computational basis state such as $|0\rangle$ or $|1\rangle$, but a linear combination of both.

In quantum computing, due to the linear nature of quantum gates, the action of a gate on such a superposition state can be determined by applying the gate to the individual basis states $|0\rangle$ and $|1\rangle$ and then using the linearity property of the gate to extend the result to the full state. Specifically, for a qubit state $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, the gate will act on the basis states $|0\rangle$ and $|1\rangle$, and the overall effect on the superposition state can be written as follows:

$$\mathcal{G}(|q\rangle) = \alpha\mathcal{G}(|0\rangle) + \beta\mathcal{G}(|1\rangle),$$

where $\mathcal{G}$ denotes the quantum gate.

Thus, the behaviour of the gate on superposition states, such as the one with $\alpha = \frac{\sqrt{3}}{2}$ and $\beta = \frac{1}{2}$, can be inferred without needing to explicitly specify the gate's action on these states. The key point is that the gate's action on the superposition state follows from the gate's action on the computational basis states $|0\rangle$ and $|1\rangle$.

Now, we extend this key idea to quantum circuits. Since quantum circuits are composed of quantum gates, the overall behaviour of a quantum circuit is also linear. This follows from the fact that quantum gates themselves are linear operators, and the composition of linear operators is also linear. In other words, if a quantum gate $\mathcal{G}$ acts linearly on the qubit state $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, then for a sequence of gates, the final state of the system can be obtained by applying the gates in sequence, preserving the linearity of the transformation. Specifically, if the circuit consists of a sequence of quantum gates $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_n$, then the overall operation of the circuit is the result of the composition of these gates, and the effect on any input state is determined by the combined action of all the gates.

As a result, understanding the circuit's action on all possible computational basis inputs is sufficient to fully characterise its behaviour. This is because, as discussed earlier, any quantum state can be expressed as a linear combination of the computational basis states $|0\rangle$ and $|1\rangle$, and the circuit's action on these basis states determines the action on any superposition of them. Thus, the behaviour of the entire quantum circuit can be deduced by examining its effect on the computational basis states $|0\rangle$ and $|1\rangle$.

We leverage this property for circuit verification. Since the overall behaviour of a quantum circuit is determined by its action on the computational basis states, verifying the behaviour

of the circuit on these basis states provides a complete method for checking the correctness of the circuit's operation. Thus, by verifying the circuit on all possible computational basis inputs, we can ensure that the quantum circuit performs the desired operations and computations are correct for the superposition states as well. The abstraction of a qubit is described next.

Definition 1. (Abstract Qubit) [16] Superposition abstraction Q of a qubit is a bit-vector tuple (s, q), where s is a 2-bit bit-vector that represents the superposition status of the qubit and q represents the qubit computational basis state.

The value of q represents the state of a qubit, which can be either $|0\rangle$ or $|1\rangle$, and are abstracted to the classical bit values 0b0 and 0b1, respectively. Specifically, $q = |0\rangle$ is abstracted to the classical state 0b0, while $q = |1\rangle$ is abstracted to classical state 0b1. These classical bit values signify that the qubit is in a definite state, either 0 or 1, without any quantum superposition. In quantum computing, however, qubits can also exist in a superposition, which is a linear combination of the computational basis states $|0\rangle$ and $|1\rangle$, and is described by the bit-vector s. The bit-vector s encodes not only the possibility of the qubit being in a classical state but also captures the quantum superposition.

The bit-vector s can take values that indicate whether the qubit is in a definite classical state or a superposition. Specifically, the values 0b00 and 0b10 indicate that the qubit is in a definite state. On the other hand, the values 0b01 and 0b11 represent superpositions of the computational basis states, meaning the qubit is not in a definite state but is instead in a probabilistic combination of $|0\rangle$ and $|1\rangle$. We will consider individual gates and their effect on qubit states next.

The C-Swap gates are used to implement the functionality of the quantum encoding circuit by performing a conditional swap operation between two qubits, typically based on the state of a control qubit. However, since C-Swap gates do not alter the quantum superposition of the qubits, they do not update the bit-vector s, which encodes the superposition information.

In contrast, the H-gates (Hadamard gates) are primarily responsible for inducing superposition in the quantum circuit. An H-gate applied to a qubit transforms its state from a definite state $|0\rangle$ or $|1\rangle$ into a superposition of these states. Therefore, in our abstraction, we model H-gates as operations that modify the s values, representing the superposition state, while leaving the q values, representing the classical state, unchanged. This abstraction allows us to separate the effects of quantum gates that induce superposition (such as the H-gate) from those that operate on classical states (such as the C-Swap gate). The abstraction of the H-gate is detailed next, where we will describe how it affects the superposition state and the underlying computational basis.

Definition 2. (Abstract H Gate) [16] Superposition abstraction of the H-gate is as follows. If $s = 0b00$ then return (0b01, q), else if $s = 0b01$ then return (0b10, q), else if $s = 0b10$ then return (0b11, q), else if $s = 0b11$ then return (0b10, q).

The abstract H-gate takes the abstracted qubit (s, q) as input. If the input qubit is in superposition, the H-gate transitions the qubit to a nonsuperposition state. If the input qubit is not in superposition, the H-gate places the qubit in superposition. The abstraction also keeps track of the number of H-gates that have been applied to the qubit. State 0b00 indicates the qubit is not in superposition and no H-gates have been applied. State 0b10 indicates the qubit is not in superposition and two or more H-gates have been applied. State 0b01 indicates the qubit is in superposition and one H-gate has been applied. State 0b11 indicates the qubit is in superposition and three or more H-gates have been applied. This count of H-gates is sufficient for verification, as no qubit in the circuit has more than one H-gate applied. The different superposition states are exploited in the correctness properties detailed in Section 4.2. Abstract C-swap gate is defined next.

Definition 3. (Abstract C-Swap Gate) Superposition abstraction of the C-Swap gate takes two abstract qubit data inputs $Q_1^i(s_1^i, q_1^i)$ and $Q_2^i(s_2^i, q_2^i)$, one abstract control qubit $Q^C(s^C, q^C)$, and has two abstract qubits as output $Q_1^o(s_1^o, q_1^o)$ and $Q_2^o(s_2^o, q_2^o)$. If $q^C = 0b1$, then $Q_1^o = Q_2^i$ and $Q_2^o = Q_1^i$. Else if $q^C = 0b0$, then $Q_1^o = Q_1^i$ and $Q_2^o = Q_2^i$.

The abstract C-Swap gate swaps the data inputs if the control qubit is in state $|1\rangle$ (indicated by $q^C = 0b1$). Otherwise, the inputs are not swapped. Note that the control qubit state is unchanged. The superposition states of the input qubits are unchanged by the abstract C-Swap gate.

The abstract encoding circuit is obtained by replacing the H-gates and C-Swap gates with their abstracted versions in the original encoding circuit. Superposition abstraction, abstract quantum gates and quantum encoding abstraction function ($QE_{Abs}()$) is elucidated in Figure 3.

4.2 | Properties

We now provide the correctness properties required to ensure the correctness of a quantum encoding circuit that encodes p binary data bits, where p is a power of 2 ($p = 2^n$). The correctness properties are defined for the abstract circuit. In

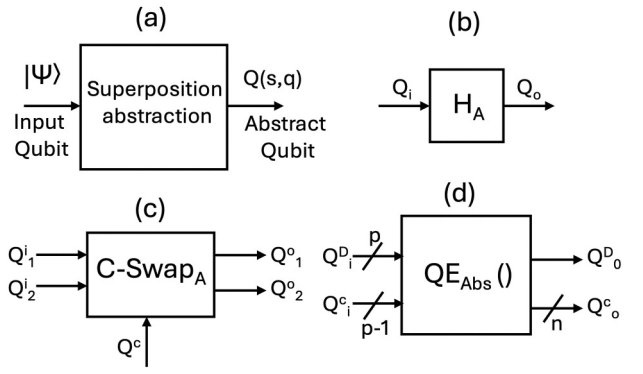

FIGURE 3 | (a) Qubit abstraction showing the abstract qubit tuple Q (s,q). (b) Abstract H-gate with abstract qubit at input and output. (c) Abstract C-Swap gate with abstract data and control qubits. (d) Quantum Encoding abstraction function with abstract qubits (p data inputs and p-1 control inputs and n+1 outputs).

Section 5, we show that if the abstract circuit satisfies the correctness properties, then the original circuit is guaranteed to be correct.

Let the function $\text{QE}_{Abs}()$ denote the abstract version of the encoding circuit. The abstract circuit will have p data qubits $Q_0^{D_{in}}$ ($s_0^{D_{in}}$, $q_0^{D_{in}}$), ..., $Q_{p-1}^{D_{in}}$ ($s_{p-1}^{D_{in}}$, $q_{p-1}^{D_{in}}$) and $p-1$ control qubits $Q_0^{C_{in}}$ ($s_0^{C_{in}}$, $q_0^{C_{in}}$), ..., $Q_{p-2}^{C_{in}}$ ($s_{p-2}^{C_{in}}$, $q_{p-2}^{C_{in}}$) at the input. The abstract circuit will reduce p data bits to $n+1$ qubits at the output, that is, $Q_0^{C_{out}}$ ($s_0^{C_{out}}$, $q_0^{C_{out}}$), ..., $Q_{n-1}^{C_{out}}$ ($s_{n-1}^{C_{out}}$, $q_{n-1}^{C_{out}}$) and $Q_0^{D_{out}}$ ($s_0^{D_{out}}$, $q_0^{D_{out}}$) at the output. The first correctness property is defined below.

Property 1. (Encoding Functional Correctness) $\langle \forall\ q_{p-1}^{D_{in}}, ..., q_0^{D_{in}}, q_{p-2}^{C_{in}}, ..., q_0^{C_{in}} \in \{0b0, 0b1\}, s_{p-1}^{D_{in}}, ..., s_0^{D_{in}}, s_{p-2}^{C_{in}}, ..., s_0^{C_{in}} = 0b00, \{(Q_0^{D_{out}}, Q_{n-1}^{C_{out}}, ..., Q_0^{C_{out}}) = \text{QE}_{Abs}\ (Q_{p-1}^{D_{in}}, ..., Q_0^{D_{in}}, Q_{p-2}^{C_{in}}, ..., Q_0^{C_{in}}) \wedge x = q_0^{C_{out}}.q_1^{C_{out}}....q_{n-2}^{C_{out}}.q_{n-1}^{C_{out}}\} \rightarrow q_0^{D_{out}} = q_x^{D_{in}} \rangle$

In the above, the symbol $\cdot$ denotes bit-vector concatenation. The notation $q_{p-2}^{C_{in}}, ..., q_0^{C_{in}}$ represents the computational basis states of the $p-1$ control qubits at the input of the circuit. Similarly, the notation $q_{p-1}^{D_{in}}, ..., q_0^{D_{in}}$ refers to the computational basis states of the p data qubits at the input.

At the output, $q_0^{D_{out}}$ represents the computational basis state of the 0^{th} data qubit. This qubit is where all the data inputs are encoded and ultimately outputted after the circuit performs its operations. The states of the control qubits at the output are denoted as $q_0^{C_{out}}, ..., q_{n-1}^{C_{out}}$, where these n control qubits are used for selection. These selection qubits determine which of the possible inputs will be selected based on the values of the control qubits at the output.

In Property 1, the value of x is obtained by concatenating the values of the selection control qubits at the output of the circuit. This concatenation process forms a bit-vector that encodes the outcome of the selection, which can be used to determine which data input to choose at the output based on the control qubits' states. Overall, the property is obtained by lifting Equation (2) to the abstract circuit and essentially captures the functional behaviour of the circuit if H-gates are not employed, which is that the circuit synthesises the functionality of a p to 1 multiplexer.

For example, for the 4-bit to 3-qubit circuit shown in Figure 2a, Property 1 will expand to:

If $x = 3$, then $q_0^{D_{out}} = q_3^{D_{in}}$, else if $x = 2$, then $q_0^{D_{out}} = q_2^{D_{in}}$, else if $x = 1$, then $q_0^{D_{out}} = q_1^{D_{in}}$, else if $x = 0$, then $q_0^{D_{out}} = q_0^{D_{in}}$.

Property 2. *(Superposition Correctness for Control Qubits) If* $s_{p-2}^{C_{in}} = 0b00 \wedge s_{p-3}^{C_{in}} = 0b00 \wedge ... \wedge s_0^{C_{in}} = 0b00$, *then the following two constraints need to be satisfied:*

a. $s_{p-2}^{C_1} = 0b01 \wedge s_{p-3}^{C_1} = 0b01 \wedge ... \wedge s_0^{C_1} = 0b01$

b. $s_{p-2}^{C_{out}} = 0b01 \wedge s_{p-3}^{C_{out}} = 0b01 \wedge ... \wedge s_0^{C_{out}} = 0b01$

The above property addresses the correct use of H-gates for the control qubits. $s_x^{C_1}$ is the superposition state of the x^{th} control qubit at stage 1, which corresponds to the state of the qubits after which H-gates have been applied. Constraint (a) states that if the control qubits are not in superposition at the input, that is, $s_{p-2}^{C_{in}} = 0b00 \wedge s_{p-3}^{C_{in}} = 0b00 \wedge ... \wedge s_0^{C_{in}} = 0b00$, then at stage 1, they should be in the superposition state of 0b01, the state obtained after the application of one H-gate, that is, $s_{p-2}^{C_1} = 0b01 \wedge s_{p-3}^{C_1} = 0b01 \wedge ... \wedge s_0^{C_1} = 0b01$. Constraint (b) states that all the control qubits should remain in the superposition state 0b01 at the output stage as well. Constraint (b) ensures that no H-gates have been applied to the control qubits after stage 1.

Property 3. *(Superposition Correctness for Data Qubits) If* $s_{p-1}^{D_{in}} = 0b00 \wedge s_{p-2}^{D_{in}} = 0b00 \wedge ... \wedge s_0^{D_{in}} = 0b00$, *then* $s_{p-1}^{D_{out}} = 0b00 \wedge s_{p-2}^{D_{out}} = 0b00 \wedge ... \wedge s_0^{D_{out}} = 0b00$.

The above property states that no H-gates should be applied to data qubits, that is, if the data qubits are in the superposition state 0b00 at the input, they should remain in that state at the output as well.

4.3 | Verification Process

The goal of this process is to verify the quantum encoding program written in a quantum programming language, for which we employ Qiskit [37]. The verification process as illustrated in Figure 4 is carried out through the following steps.

Step 1: **Abstraction** — The first step involves precoding the abstractions of the quantum gates in the SMT (Satisfiability Modulo Theory) language as functions.

Step 2: **Translation** — In the second step, the Qiskit program is translated into SMT using the previously precoded abstract gate functions.

Step 3: **Instantiation** — The third step is to instantiate Properties 1–3, based on the size of the circuit being verified (note that the properties are generically defined for $p = 2^n$ classical bits) and encode these properties in SMT.

Step 4: **Verification** — An SMT solver, such as z3, is then used to check the SMT file. A correct circuit will not produce a counterexample, while an erroneous circuit should produce one.

The resulting counterexample can then be used to trace the source of the error in the Qiskit/quantum program that specifies the circuit.

5 | Correctness

Equation (2) defines the specification of the quantum encoding circuit. In this section, we provide a proof demonstrating that if

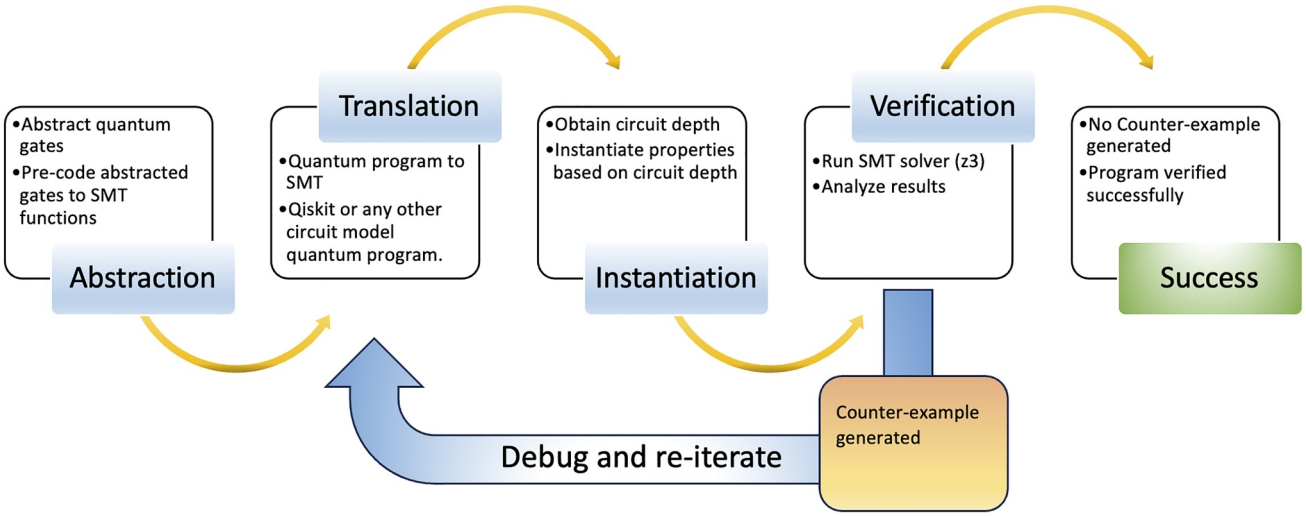

FIGURE 4 | Verification process flow.

the abstracted circuit satisfies Properties 1, 2, and 3, then the encoding circuit will be correct, meaning it will satisfy Equation (2).

Lemma 1. *If there is no H-gate in stage 1 on any control qubit line or more than one H-gate on any control qubit line, property 2 will not be satisfied.*

Proof. Property 2 verifies that if the superposition state of all control qubit lines at the input is 0b00, then the superposition state will transition to 0b01 at stage one and remain 0b01 at the output stage. If, in stage 1, no Hadamard (H) gate is applied to a control qubit line, its superposition state will remain 0b00, and consequently, Property 2 will be violated. Furthermore, if more than one H-gate is applied to a control qubit line, the superposition state of that qubit will transition to either 0b10 or 0b11 (as defined in Definition 2), which will also result in the violation of Property 2. □

Lemma 2. *If there are any H-gates applied to any data qubit line, property 3 will not be satisfied.*

Proof. Property 3 checks that if the superposition state of all data qubit lines at the input is 0b00, then their superposition state will remain in 0b00 at the output stage. If there is one or more H-gates on a data qubit line, then the superposition state of that qubit will transition to 0b01, 0b10 or 0b11 (see Definition 2) and therefore property 3 will be violated. □

Lemma 3. *If the abstract encoding circuit without H-gates satisfies Property 1, the corresponding quantum encoding circuit without H-gates will satisfy Equation (2).*

Proof. Without the H gates, the encoding circuit with only C-Swap gates synthesises the function of a p to 1 multiplexor (MUX) (Equation 2) for computational basis inputs, that is, when $q_0^{C_{out}} \ldots q_{n-1}^{C_{out}} = |0\ldots00\rangle$, then $q_0^{D_{out}} = q_0^{D_{in}}$, when $q_0^{C_{out}} \ldots q_{n-1}^{C_{out}} = |0\ldots01\rangle$, then $q_0^{D_{out}} = q_1^{D_{in}}$, and so on. Property 1 checks that the circuit with only abstract C-Swap gates synthesises the function of a p to 1 multiplexer (MUX) for Boolean inputs. The behaviour of the abstract C-Swap gate is identical to the C-Swap gate when computational basis states $|0\rangle$ and $|1\rangle$ are abstracted with Boolean values 0b0 and 0b1 (Definition 3). Therefore, if the abstracted circuit satisfies Property 1, the encoding circuit will synthesise the function of a p to 1 multiplexer (MUX) for computational basis inputs, and will therefore satisfy Equation (2). □

Theorem 1. *If the abstract encoding circuit satisfies Properties 1, 2, and 3, then the quantum encoding circuit under test will satisfy Equation (2).*

Proof. The superposition behaviour of the circuit will be correct, that is, the placement of the H gates in the circuit will be correct if Properties 2 and 3 are satisfied (Lemmas 1 and 2). If Property 1 is satisfied, the circuit with only C-Swap gates will satisfy Equation (2) without superposition (Lemma 3). Since only H-gates induce superposition in the circuit, the encoding circuit will satisfy Equation (2) if Properties 1, 2, and 3 are satisfied by the abstracted circuit. □

6 | Results

In this section, we present the verification results. The benchmarks for verification were generated by varying the number of input qubits in the quantum encoding circuit, ranging from 7 qubits (representing 4 classical data bits) to 8191 qubits (representing 4096 classical data bits). The verification experiments were conducted on an Intel(R) Core(TM) i9-12900K CPU @ 3.2 GHz with 32 GB of RAM, running the Ubuntu 64-bit operating system. Each verification experiment involved checking whether a quantum encoding circuit satisfied the conjunction of Properties 1–3. The properties were checked using the z3 SMT solver, version 4.8.12 [38].

Table 3 summarises the verification outcomes for quantum encoding circuits that are free of errors. The "Quantum Encoding Circuit" column indicates the number of classical input bits and the corresponding number of input qubits in the circuit. All benchmarks without errors satisfied the conjunction of Properties 1–3. The "Circuit With No Error" column details

TABLE 3 | Verification Results 2.

Quantum encoding circuit		Circuit with No error	
Classical bits	**Input qubits**	**Time(s)**	**Memory (MB)**
4	7	0.01	19.32
8	15	0.01	19.46
16	31	0.01	19.52
32	63	0.01	19.72
64	127	0.01	20.20
128	255	0.02	21.06
256	511	0.03	22.61
512	1023	0.06	26.05
1024	2047	0.06	28.66
2048	4095	0.12	46.32
4096	**8191**	**0.29**	**85.85**

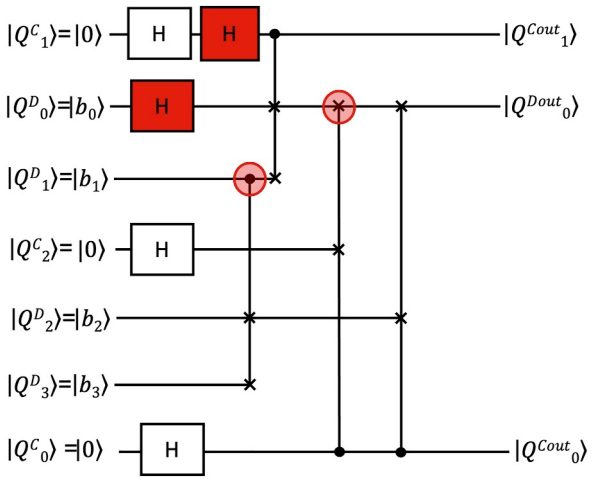

FIGURE 5 | 4-Bit to 3-qubit quantum encoding circuit with error scenarios.

the verification time (in seconds) and peak memory usage (in megabytes) required to check Properties 1–3. Remarkably, the largest circuit, consisting of 8191 qubits, was successfully verified in just ***0.29 s***, with a peak memory usage of ***85.85 MB***. This clearly demonstrates the efficiency and scalability of our proposed method in this work compared to the existing work detailed in Section 2.

Error scenarios were introduced into the quantum encoding circuit to demonstrate the effectiveness of the verification methodology in detecting faults within the circuit. Figure 5 illustrates the various error scenarios (highlighted in red). Four types of errors were introduced: multiple H-gates (or missing H-gates) in the control qubit lines, H-gates incorrectly placed in the data qubit lines, incorrect control inputs for C-Swap gates, and incorrect swap inputs for C-Swap gates.

The verification times and peak memory usage for error scenarios are detailed in Table 4. The "Data Qubit H-gate Error" column provides data for the case where an H-gate is mistakenly applied to a data qubit. The "Control Qubit H-gate Error" column presents results for incorrect H-gate placement on a control qubit line. The "Control Qubit Swap Error" column addresses the scenario where the control input to the C-swap gate is incorrect, while the "Data Qubit Swap Error" column shows statistics for cases where one of the data inputs to the C-swap gate is erroneously connected. Error locations were varied across circuit depth and qubit lines, and all benchmarks with errors failed to satisfy Properties 1-3.

The verification method successfully classified all benchmarks as either correct or erroneous. For erroneous benchmarks, the z3 tool generated a counterexample, further confirming the effectiveness of the approach. As stated earlier, other verification approaches model quantum circuits in Hilbert space (complex vector space). However, our approach employs abstractions to reduce the problem to bit-vector space. The significance of the results is that they validate our approach to be more efficient and scalable one. We have demonstrated that we

TABLE 4 | Verification Results 1.

Quantum encoding circuit		Data qubit H-gate error		Control qubit H-gate error		Control qubit swap error		Data qubit swap error	
Classical bits	**Input qubits**	**Time (s)**	**Memory (MB)**	**Time (s)**	**Memory (MB)**	**Time (s)**	**Memory (MB)**	**Time (s)**	**Memory (MB)**
4	7	0.01	19.08	0.01	19.09	0.01	19.41	0.01	19.41
8	15	0.01	19.09	0.01	19.08	0.01	19.45	0.01	19.46
16	31	0.01	19.08	0.01	19.09	0.01	19.52	0.01	19.52
32	63	0.01	19.08	0.01	19.18	0.01	19.81	0.01	19.80
64	127	0.01	19.08	0.01	19.28	0.01	20.19	0.01	20.20
128	255	0.01	19.18	0.01	19.48	0.02	21.05	0.02	21.05
256	511	0.01	19.27	0.01	19.96	0.03	22.94	0.03	22.72
512	1023	0.01	19.61	0.02	20.83	0.05	26.22	0.05	26.24
1024	2047	0.01	20.02	0.03	22.58	0.04	30.21	0.04	30.24
2048	4095	0.02	21.40	0.04	26.56	0.08	50.47	0.08	50.46
4096	**8191**	**0.03**	**24.05**	**0.08**	**42.45**	**0.15**	**87.51**	**0.15**	**87.49**

can verify quantum encoding circuits with up to 8191 qubits with very low memory and time, demonstrating that the proposed approach can easily be employed to verify even much larger encoding circuits. As can be seen from Table 1 in Section 2, other approaches have demonstrated verification of circuits with only up to 250 qubits. Also, this is the first approach that tackles the problem of quantum encoding circuit verification. Other approaches have not addressed this problem.

7 | Conclusion

We present a formal verification methodology for a quantum data encoding scheme, capable of verifying quantum circuits with over 8000 qubits in under 1 s, a significant advancement in comparison with existing methods. By leveraging superposition abstraction, we reduce quantum circuits from Hilbert space to bit-vector space, significantly improving verification efficiency and scalability. The method's completeness and robustness have been rigorously validated through extensive testing across various error scenarios. Looking ahead, we plan to extend this superposition abstraction methodology to a broader range of quantum algorithms and programs, driving efficient, reliable, and scalable verification processes that are critical for advancing quantum computing.

Author Contributions

Arun Govindankutty: conceptualization, data curation, formal analysis, investigation, methodology, software, validation, writing – original draft, writing – review and editing. **Sudarshan K. Srinivasan:** conceptualization, formal analysis, investigation, methodology, supervision, validation, writing – original draft, writing – review and editing.

Acknowledgements

This paper is based upon work supported by the National Science Foundation under Grant No. 2513276.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Data available on request from the authors.

References

1. V. Hassija, V. Chamola, A. Goyal, S. S. Kanhere, and N. Guizani, "Forthcoming Applications of Quantum Computing: Peeking into the Future," *IET Quantum Communication* 1, no. 2 (2020): 35–41, https://doi.org/10.1049/iet-qtc.2020.0026.

2. G. Arun and V. Mishra, "A Review on Quantum Computing and Communication," in *2014 2nd International Conference on Emerging Technology Trends in Electronics, Communication and Networking* (2014): 1–5.

3. M. Kim, D. Venturelli, and K. Jamieson, "Towards Hybrid Classical-Quantum Computation Structures in Wirelessly-Networked Systems," in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks* (2020): 110–116.

4. L. J. O'Riordan, M. Doyle, F. Baruffa, and V. Kannan, "A Hybrid Classical-Quantum Workflow for Natural Language Processing," *Machine Learning: Science and Technology* 2, no. 1 (2020): 015011, https://doi.org/10.1088/2632-2153/abbd2e.

5. M. Lanzagorta and J. K. Uhlmann, "Hybrid Quantum-Classical Computing With Applications to Computer Graphics," in *ACM SIGGRAPH 2005 Courses* (2005), 2.

6. Y.-F. Yang and M. Sun, "Semiconductor Defect Detection by Hybrid Classical-Quantum Deep Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 2323–2332.

7. J. P, S. Hariharan, V. Madhivanan, S. N, M. Krisnamoorthy, and A. K. Cherukuri, "Enhanced Qsvm With Elitist Non-Dominated Sorting Genetic Optimisation Algorithm for Breast Cancer Diagnosis," *IET Quantum Communication* 5, no. 4 (2024): 384–398, https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/qtc2.12113.

8. M. Sha, "Quantum Intelligence in Medicine: Empowering Thyroid Disease Prediction Through Advanced Machine Learning," *IET Quantum Communication* 5, no. 2 (2024): 123–139, https://doi.org/10.1049/qtc2.12078.

9. D. De and A. Lord, "Guest Editorial: Quantum Industry: Applications in Quantum Communication (quantum.tech europe 2022)," *IET Quantum Communication* 5, no. 3 (2024): 198–201, https://doi.org/10.1049/qtc2.12104.

10. M. S. Peelam, A. A. Rout, and V. Chamola, "Quantum Computing Applications for Internet of Things," *IET Quantum Communication* 5, no. 2 (2024): 103–112, https://doi.org/10.1049/qtc2.12079.

11. G. G. Rozenman, N. K. Kundu, R. Liu, et al., "The Quantum Internet: A Synergy of Quantum Information Technologies and 6g Networks," *IET Quantum Communication* 4, no. 4 (2023): 147–166, https://doi.org/10.1049/qtc2.12069.

12. A. Stavdas, E. Kosmatos, C. Maple, et al., "Quantum Key Distribution for V2i Communications With Software-Defined Networking," *IET Quantum Communication* 5, no. 1 (2024): 38–45, https://doi.org/10.1049/qtc2.12070.

13. J. A. Cortese and T. M. Braje, *System and Technique for Loading Classical Data into a Quantum Computer*, Vol. 9 (Patent, 2019).

14. O. Hasan and S. Tahar, "Formal Verification Methods," in *Encyclopedia of Information Science and Technology*. 3rd ed. (IGI global, 2015): 7162–7170.

15. A. Govindankutty, S. K. Srinivasan, and N. Mathure, "Rotational Abstractions for Verification of Quantum Fourier Transform Circuits," *IET Quantum Communication* 4, no. 2 (2023): 84–92, https://doi.org/10.1049/qtc2.12055.

16. B. J. Sitou Campbell and S. K. Srinivasan, "Formal Verification for Cyclic Quantum Walk Circuits," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)* (2024): 1–5.

17. M. Amy, "Towards Large-Scale Functional Verification of Universal Quantum Circuits," *Electronic Proceedings in Theoretical Computer Science* 287 (2019): 1–21, https://doi.org/10.4204/eptcs.287.1.

18. A. Matthew, *Formal Methods in Quantum Circuit Design* (University of Waterloo, 2019), http://hdl.handle.net/10012/14480.

19. S. J. Gay, R. Nagarajan, and N. Papanikolaou, "Qmc: A Model Checker for Quantum Systems," in *Computer Aided Verification*, eds. A. Gupta and S. Malik (Springer Berlin Heidelberg, 2008), 543–547.

20. Z. Tian, Quantum Stabilizer Formalism for Any Composite System (2024), https://arxiv.org/abs/2311.04255.

21. P. Baltazar, R. Chadha, P. Mateus, and A. Sernadas, "Towards Model-Checking Quantum Security Protocols," in *2007 First International Conference on Quantum, Nano, and Micro Technologies (ICQNM'07)* (2007): 14.

22. J. Seiter, M. Soeken, R. Wille, and R. Drechsler, "Property Checking of Quantum Circuits Using Quantum Multiple-Valued Decision

Diagrams," in *Reversible Computation*, eds. R. Glück and T. Yokoyama (Springer Berlin Heidelberg, 2013): 183–196.

23. L. Burgholzer and R. Wille, "Advanced Equivalence Checking for Quantum Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40 (2021): 1810–1824, https://doi.org/10.1109/tcad.2020.3032630.

24. S. Yamashita and I. L. Markov, "Fast Equivalence-Checking for Quantum Circuits," in *2010 IEEE/ACM International Symposium on Nanoscale Architectures* (2010), 23–28.

25. G. F. Viamontes, I. L. Markov, and J. P. Hayes, Checking Equivalence of Quantum Circuits and States (2007), https://arxiv.org/abs/0705.0017.

26. X. Hong, M. Ying, Y. Feng, X. Zhou, and S. Li, Approximate Equivalence Checking of Noisy Quantum Circuits (2021), https://arxiv.org/abs/2103.11595.

27. S. M. Beillahi, M. Y. Mahmoud, and S. Tahar, "Hierarchical Verification of Quantum Circuits," in *NASA Formal Methods*, eds. S. Rayadurgam and O. Tkachuk (Springer International Publishing, 2016): 344–352.

28. M. Y. Mahmoud, P. Panangaden, and S. Tahar, "On the Formal Verification of Optical Quantum Gates in Hol," in *Formal Methods for Industrial Critical Systems*, eds. M. Núñez and M. Güdemann (Springer International Publishing, 2015), 198–211.

29. J. Liu, B. Zhan, S. Wang, et al., "Formal Verification of Quantum Algorithms Using Quantum Hoare Logic," in *Computer Aided Verification*, eds. I. Dillig and S. Tasiran (Springer International Publishing, 2019), 187–207.

30. A. Barenco, C. H. Bennett, R. Cleve, et al., "Elementary Gates for Quantum Computation," *Physical Review A* 52, no. 5 (November 1995): 3457–3467, https://link.aps.org/doi/10.1103/PhysRevA.52.3457.

31. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th ed. (Cambridge University Press, 2011).

32. C. Bernhardt, *Quantum Computing for Everyone* (MIT Press, 2020).

33. R. LaPierre, *Introduction to Quantum Computing* (Springer International Publishing, 2021).

34. M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Data Encoding Patterns for Quantum Computing," in *Proceedings of the 27th Conference on Pattern Languages of Programs, Ser. PLoP '20* (Hillside Group, 2022).

35. M. Rath and H. Date, "Quantum Data Encoding: A Comparative Analysis of Classical-To-Quantum Mapping Techniques and Their Impact on Machine Learning Accuracy," *EPJ Quantum Technology* 11, no. 1 (2024): 72, https://doi.org/10.1140/epjqt/s40507-024-00285-3.

36. Q. Liu, B. Preneel, Z. Zhao, and M. Wang, "Improved Quantum Circuits for Aes: Reducing the Depth and the Number of Qubits," in *Advances in Cryptology – ASIACRYPT 2023*, eds. J. Guo and R. Steinfeld (Springer Nature Singapore, 2023), 67–98.

37. Qiskit contributors, Qiskit: An Open-Source Framework for Quantum Computing (2023).

38. L. De Moura and N. Bjørner, "Z3: An Efficient Smt Solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Ser. TACAS'08/ETAPS'08* (Springer-Verlag, 2008), 337–340.