



Precision-aware fixed-point emulation of Grover's algorithm: asymptotic error bounds and design guidelines

Seonghyun Choi¹ · Kyeongwon Lee¹ · Jongin Choi¹ · Chaebin Jung¹ · Woojoo Lee¹

Received: 16 June 2025 / Accepted: 26 May 2026 / Published online: 12 June 2026
© The Author(s) 2026

Abstract

We propose a rigorous precision-aware framework for fixed-point emulation of Grover's quantum search algorithm, systematically analyzing how truncation errors propagate and accumulate through quantum gate operations. Although fixed-point arithmetic significantly reduces resource requirements in quantum computing emulators, limited fractional-bit precision introduces non-negligible numerical errors, adversely impacting algorithm accuracy. To address this, we introduce a simplified two-amplitude quantum-state representation that captures Grover's algorithmic structure across iterations. Leveraging this model, we analytically derive explicit asymptotic error bounds on the emulated measurement probability distribution, demonstrating that errors scale exponentially as $O(2^{n-f})$, where n is the number of qubits and f is fractional-bit precision. Extensive numerical simulations and practical experiments on an actual fixed-point quantum emulator validate our theoretical predictions, confirming the accuracy and robustness of our error analysis. Finally, we present a closed-form expression to determine the minimal fractional-bit precision needed to meet a specified accuracy threshold, providing concrete, actionable guidelines for designing resource-efficient quantum computing emulators.

Keywords Quantum computing emulation · Grover's algorithm · Fixed-point arithmetic · Truncation error · Resource optimization · Asymptotic error analysis

✉ Woojoo Lee
space@cau.ac.kr

Seonghyun Choi
chltmdgus272@cau.ac.kr

Kyeongwon Lee
since69se@cau.ac.kr

Jongin Choi
nex1248@cau.ac.kr

Chaebin Jung
cc89124@cau.ac.kr

¹ Department of Intelligent Semiconductor Engineering, Chung-Ang University, Seoul 06974, Korea

1 Introduction

Over the past few decades, *quantum computing* (QC) has been vigorously studied in both academia [1–4] and industry [5–7]. By leveraging quantum phenomena such as superposition and entanglement, quantum computers operate on fundamentally different computational principles compared to classical machines. Owing to this unique mode of operation, quantum computers can dramatically reduce the computational overhead for certain problems and potentially solve instances that are intractable on classical supercomputers [8–10]. In fact, multiple studies have demonstrated the superiority of quantum hardware over classical counterparts in carefully chosen tasks [11, 12].

To translate these performance advantages into real-world applications, considerable effort has been dedicated to developing QC hardware via various physical implementations [13–15]. In parallel, researchers are exploring quantum algorithms that aim to efficiently tackle practical problems, such as optimization [16–18], machine learning [19–21], and complex system simulations [22–24]—all of which may benefit from quantum speedups or favorable computational scaling for specific problem classes.

Despite these advances, current quantum hardware continues to face significant challenges in *scalability* and *error correction*, and has therefore not yet reached a practical, large-scale fault-tolerant stage [4, 25]. Because of these limitations, researchers often lack direct hardware access to develop, test, and evaluate sophisticated quantum algorithms. To bridge this gap, *QC emulators*—which simulate or emulate quantum circuits on classical machines—have been actively investigated [26–33]. These emulators fill a critical role in quantum computing research, enabling rapid prototyping and testing of quantum algorithms before fully scalable quantum hardware becomes available. Moreover, recent efforts have explored strategies for integrating quantum hardware and software stacks into classical computing infrastructures, facilitating hybrid quantum-classical workflows [3, 34].

A QC emulator models quantum states and gates as vectors and matrices, respectively, on a classical computer [35]. However, numerical errors originating from the classical hardware—such as floating-point or fixed-point inaccuracies—can directly degrade the fidelity of the emulated quantum operations. Identifying and quantifying these classical errors are therefore essential for designing reliable QC emulators. Moreover, in state-vector emulation, the total memory cost scales with both the number of amplitudes and the arithmetic word length. Thus, although exponential state growth is intrinsic to classical QC emulation, precision remains a key implementation parameter because it directly affects memory footprint, data-path cost, and emulator accuracy. Although a few works have discussed precision issues in QC simulation—for example, prior studies showed that quantization errors in fixed-point emulators grow with circuit size and addressed this issue by adopting higher-precision floating-point arithmetic [28, 36]—, systematic, algorithm-specific investigations that translate arithmetic precision into explicit emulator-accuracy bounds and practical design rules remain limited. In particular, there is a lack of focused studies that determine how much fixed-point precision is minimally required for a given target algorithm and error tolerance.

In this paper, we address this gap by proposing a *quantitative framework* to analyze how classical arithmetic errors influence the accuracy of a *fixed-point-based* QC emulator. Our objective is not merely to reproduce the ideal closed-form final state of Grover's algorithm, but to analyze the iteration-by-iteration execution of a fixed-point emulator, in which truncation arises at each repeated diffusion-scaling step and accumulates over successive Grover iterations. From this analysis, we derive a closed-form formula that computes the *minimum required precision* (i.e., the number of fractional bits) required to keep errors under a specified target—thus providing practical guidelines for resource optimization.

To concretize our study, we focus on *Grover's algorithm* [37], a quantum search procedure widely regarded for its quadratic speedup in unstructured search. This speedup benefits numerous application domains demanding efficient data retrieval, and emulators specialized to Grover's algorithm have indeed been proposed [32, 38]. Grover's algorithm exploits quantum superposition to simultaneously probe multiple data items, requiring only $O(\sqrt{N})$ queries for a dataset of size N , whereas classical algorithms need $O(N)$ [39–41]. This offers a *quadratic* improvement over the classical approach, empowering quantum computers to address search problems at scales that would be infeasible for classical machines. The algorithm also underpins potential breakthroughs in cryptography [42–44] and optimization [45–47]. We emphasize that the goal of this work is not to claim immediate generality across arbitrary quantum algorithms, but to establish a rigorous algorithm-specific precision analysis in a structured and practically meaningful case where closed-form treatment is possible. Grover's algorithm is particularly well suited to this purpose because its state evolution remains in a two-amplitude subspace and the same Grover operator is repeatedly applied over many iterations, allowing arithmetic-error propagation to be analyzed formally and translated into concrete emulator design rules.

Next, we adopt a fixed-point arithmetic environment for the QC emulator, motivated not only by hardware efficiency but also by the practical need to control the per-amplitude word length in an emulator whose state storage already scales exponentially with the qubit count. This choice also reflects the common use of fixed-point arithmetic in hardware-oriented QC emulators, where fixed-point operators often offer substantially lower logic utilization, power consumption, and latency than floating-point counterparts [32, 35, 38, 48–51]. In particular, we investigate how the resulting *truncation errors*—arising from representing real numbers with a finite number of fractional bits—accumulate over multiple gate operations. Although fixed-point arithmetic mitigates the resource overheads more effectively than floating-point in many practical designs, the finite precision of fractional bits inevitably leads to truncation. Accordingly, the central design question is not merely whether truncation occurs, but how much precision is minimally required to satisfy a target error bound without unnecessary hardware over-provisioning. Throughout this work, we assume ideal oracle and diffusion operations and intentionally isolate truncation as an independent arithmetic-error source, rather than modeling device noise or imperfect quantum gates. In this work, we pinpoint the specific operations in Grover's algorithm where these truncations occur and rigorously quantify how they propagate through consecutive computational steps.

Our approach begins by proposing a simplified representation of the quantum-state within Grover's algorithm and providing a theoretical proof of its validity. Building on this representation, we derive a mathematical formula for the ℓ_2 distance between the true and emulated measurement probability distributions under varying qubit counts and fractional-bit precision. We then investigate how this ℓ_2 distance scales with these parameters, forming a rigorous theoretical foundation for understanding and predicting emulator errors. We demonstrate the soundness of our derivation by comparing the theoretical error bound with actual measurements from a working fixed-point QC emulator specialized for Grover's algorithm (henceforth referred to as the **FP-Grover** emulator). Finally, leveraging our formula, we determine the minimum fractional-bit precision required to maintain emulator errors below a specified threshold, providing practical design guidelines for resource-efficient FP-Grover emulators by explicitly addressing hardware resource usage (e.g., memory, registers, arithmetic logic) under given error constraints.

The key contributions of this paper are summarized as follows:

1. *Numerical error analysis:* We present a systematic method to analyze the truncation errors in a FP-Grover emulator, deriving how these errors manifest in the measurement probability distribution (measured by the ℓ_2 distance).
2. *Empirical validation:* We evaluate how the number of qubits and fractional bits affect the overall accuracy loss. Through comparisons of theoretical predictions and emulator outputs, we verify the validity of our approach.
3. *Design guideline:* We formulate a closed-form expression relating desired error thresholds to the necessary fixed-point precision. This formula can be used directly by designers to achieve resource-optimized emulators while guaranteeing specified accuracy levels.

This paper is structured in the following way. Section 2 briefly reviews the essential background on quantum states, gates, and the principles of Grover's algorithm. Section 3 details our simplified state representation technique for Grover's algorithm and derives the error accumulation formulas under fixed-point arithmetic. Section 4 compares the proposed theoretical formula with measured error values from a practical emulator and discusses how to determine the minimum required fractional bits for a given error target. Finally, Sect. 5 offers concluding remarks and future perspectives.

2 Fixed-point Grover emulation: preliminaries

In this preliminary section, we briefly introduce the fundamental quantum computing concepts and Grover's algorithm necessary to understand the fixed-point arithmetic emulation and associated truncation errors analyzed subsequently.

2.1 Quantum information and gate operations

In classical computing, information is stored in an array of bits, and logical gates process these bits to perform computations. In quantum computing, qubits and quantum gates, respectively, serve as counterparts to classical bits and logical gates. Specifi-

cally, the amplitude and phase of a qubit encode information, while physical operations that modify these amplitudes and phases drive the computation. Unlike classical bits, which can only assume one of two discrete values (0 or 1), a qubit can reside in either of the basis states $|0\rangle$, $|1\rangle$, or any superposition thereof. This superposition property is rooted in uniquely quantum phenomena, giving rise to the enhanced computational power that quantum computers can exhibit relative to classical devices.

A single-qubit system has two basis states [52], each represented by a unit column vector:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (1)$$

Any superposition of $|0\rangle$ and $|1\rangle$ can be written as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad (2)$$

where α and β are complex amplitudes that satisfy $\|\alpha\|^2 + \|\beta\|^2 = 1$. Hence, $|\psi\rangle$ is a unit vector.

A quantum system comprising n qubits spans 2^n possible basis states. Labeling these basis states as $|0\rangle, |1\rangle, \dots, |2^n - 1\rangle$, we can model the state of the system by a 2^n -dimensional column vector:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle = [\alpha_0 \ \alpha_1 \ \dots \ \alpha_{2^n-1}]^T, \quad (3)$$

where each α_i is a complex number. As in the single-qubit case, these amplitudes must satisfy $\sqrt{\sum_{i=0}^{2^n-1} \|\alpha_i\|^2} = 1$ to ensure $|\psi\rangle$ is a unit vector.

When measuring a multi-qubit state in the computational basis, the system probabilistically collapses to one of the basis states. Specifically, the probability of obtaining $|i\rangle$ upon measurement is given by

$$P(|\psi\rangle \rightarrow |i\rangle) = \|\alpha_i\|^2, \quad (4)$$

and the sum of all measurement probabilities is 1, implying

$$\| |\psi\rangle \| = \sqrt{\sum_{i=0}^{2^n-1} \|\alpha_i\|^2} = 1. \quad (5)$$

Quantum gates: A quantum gate can be represented by a unitary matrix that acts on the state vector. Because quantum states must remain normalized (as in Eq. (5)), the operator must preserve the vector norm, making it unitary. Table 1 lists several commonly used single- and two-qubit gates, along with their matrix forms. Multi-qubit gates can be constructed from these basic operations via tensor (Kronecker)

Table 1 Representative quantum gates for single-qubit and two-qubit systems

Gate	Symbol	Unitary matrix	Gate	Symbol	Unitary matrix
Hadamard	H	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Phase-shift	P	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
Pauli	X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Controlled-NOT	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$		$CNOT$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$			

products. For instance, if $U_1, U_2, \dots, U_n$ are single-qubit gates applied in parallel to the n qubits, the overall n -qubit operation U is the Kronecker product:

$$U = U_1 \otimes U_2 \otimes \dots \otimes U_n. \quad (6)$$

2.2 Grover's algorithm

Grover's algorithm [37] is a quantum search procedure that identifies an x satisfying $f(x) = 1$, where $f(x)$ is defined as

$$f(x) = \begin{cases} 1 & \text{if } x \in S, \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

and S is the set of valid solutions (i.e., $s \in S$ implies $f(s) = 1$). Applied to an n -qubit system, Grover's algorithm first prepares the uniform superposition over all 2^n basis states, then iteratively applies k rounds of a *Grover operator* G to amplify the amplitudes of solution states. Denoting the final state by $|\psi_{\text{final}}\rangle$, one can measure it with high probability of collapsing onto a solution. Mathematically,

$$|\psi_{\text{final}}\rangle = G^k |u\rangle, \quad \text{where } G = (H^{\otimes n} Z_{\text{or}} H^{\otimes n})(Z_f). \quad (8)$$

Here, the number of iterations k is approximately

$$k = \left\lfloor \frac{\pi}{4} \sqrt{\frac{2^n}{n_s}} \right\rfloor, \quad (9)$$

where $\lfloor \cdot \rfloor$ rounds to the nearest integer and n_s is the number of solutions in S . The initial uniform superposition state $|u\rangle$ is represented as the equation below:

$$|u\rangle = H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle, \quad (10)$$

in which $|0^n\rangle$ denotes all n qubits in the $|0\rangle$ state, and $H^{\otimes n}$ is a Hadamard gate acting on each qubit.

The G operator in Eq. (8) comprises two components: an *oracle* gate, represented by the matrix Z_f , and a *diffusion* gate, given by $H^{\otimes n} Z_{\text{or}} H^{\otimes n}$. As illustrated in the middle of Fig. 1, the oracle gate identifies solution basis states $|s\rangle$ and flips their phase, effectively negating the amplitude coefficient for each solution state. Meanwhile, the diffusion gate, shown on the right in Fig. 1, reflects the amplitude of all 2^n basis states about their average. Concretely, the oracle Z_f is defined by

$$Z_f |x\rangle = (-1)^{f(x)} |x\rangle = \begin{cases} -|x\rangle & x \in S, \\ |x\rangle & x \notin S, \end{cases} \quad (11)$$

while, as written in [53], the diffusion operator $H^{\otimes n} Z_{\text{or}} H^{\otimes n}$ can be expressed as

$$\begin{aligned} H^{\otimes n} Z_{\text{or}} H^{\otimes n} &= 2|u\rangle\langle u| - I_n \\ &= \begin{bmatrix} 2^{-n+1} & \dots & 2^{-n+1} \\ \vdots & \ddots & \vdots \\ 2^{-n+1} & \dots & 2^{-n+1} \end{bmatrix} - I_n, \end{aligned} \quad (12)$$

$$\begin{aligned} \text{where } Z_{\text{or}} |x\rangle &= 2|0^n\rangle\langle 0^n|x\rangle - |x\rangle \\ &= \begin{cases} -|x\rangle & x \neq 0, \\ |x\rangle & \text{otherwise.} \end{cases} \end{aligned} \quad (13)$$

Above, $\langle x|$ is the conjugate transpose of $|x\rangle$.

Next, let us define two orthonormal vectors:

$$|\psi_S\rangle = \frac{1}{\sqrt{n_s}} \sum_{\substack{i=0 \\ i \in S}}^{2^n-1} |i\rangle, \quad (14)$$

$$|\psi_{NS}\rangle = \frac{1}{\sqrt{2^n - n_s}} \sum_{\substack{i=0 \\ i \notin S}}^{2^n-1} |i\rangle. \quad (15)$$

Using these, the initial state Eq. (10) can be expressed as a superposition of $|\psi_S\rangle$ and $|\psi_{NS}\rangle$:

$$|u\rangle = \sqrt{\frac{n_s}{2^n}} |\psi_S\rangle + \sqrt{\frac{2^n - n_s}{2^n}} |\psi_{NS}\rangle.$$

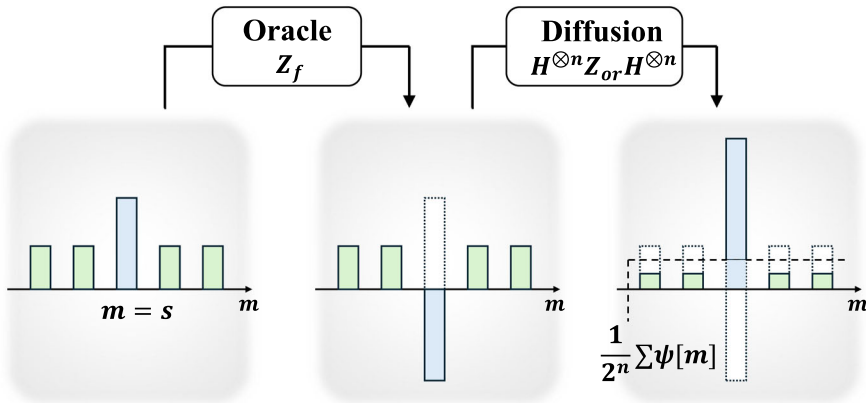


Fig. 1 Illustration of the G operation in Grover's algorithm, which reflects the amplitudes of the state about their average to amplify the solution states

From Eqs. (14) and (15), the vectors $|\psi_S\rangle$ and $|\psi_{NS}\rangle$ form an orthonormal basis in a two-dimensional subspace, as they are mutually orthogonal and each is of unit length. Consequently, one can represent the initial state of Grover's algorithm in this subspace, as shown in Fig. 2, in *angular form*:

$$|u\rangle = \sin(\theta) |\psi_S\rangle + \cos(\theta) |\psi_{NS}\rangle, \quad (16)$$

$$\text{where } \theta = \sin^{-1}\left(\sqrt{\frac{n_s}{2^n}}\right). \quad (17)$$

Here, the Grover operator G can be viewed as a rotation by 2θ on the unit circle spanned by $|\psi_{NS}\rangle$ and $|\psi_S\rangle$. After k applications of G , the state vector rotates by $2k\theta$, yielding a final state

$$\begin{aligned} |\psi_{\text{final}}\rangle &= G^k |u\rangle \\ &= \sin((2k+1)\theta) |\psi_S\rangle + \cos((2k+1)\theta) |\psi_{NS}\rangle. \end{aligned} \quad (18)$$

Choosing k according to Eq. (9) makes $(2k+1)\theta$ approach $\pi/2$, thereby maximizing the amplitude of $|\psi_S\rangle$ and enabling a high-probability measurement of a solution. Letting $\delta = \frac{\pi}{2} - (2k+1)\theta$ denote the *residual angle error*, one can show $|\delta| < \theta$, which implies

$$|\cos((2k+1)\theta)| = |\sin \delta| = \sin |\delta| < \sin \theta \quad (19)$$

Thus, the nonsolution amplitude remains strictly smaller than the initial amplitude $\sin(\theta)$, ensuring that the solution basis states are prominently amplified.

2.3 Fixed-point arithmetic and truncation error

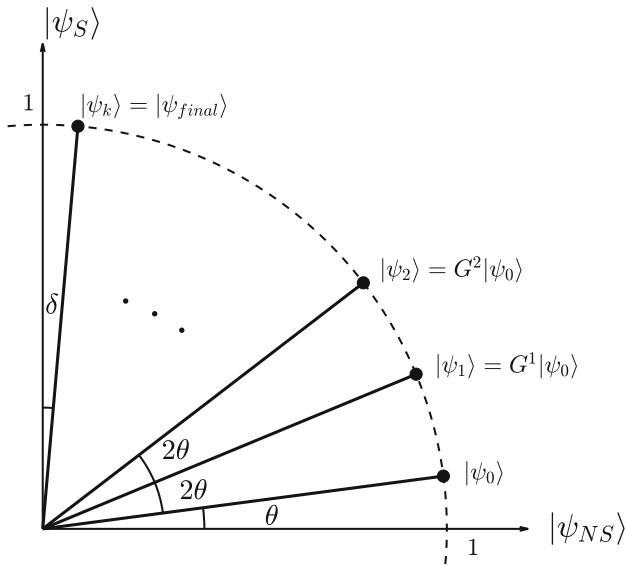


Fig. 2 Geometric interpretation of G as a rotation by 2θ in the plane spanned by $|\psi_{NS}\rangle$ and $|\psi_S\rangle$

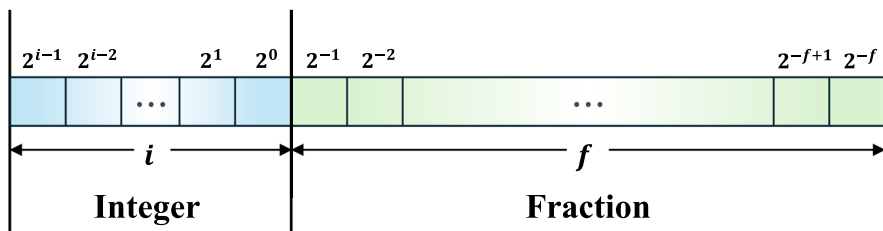


Fig. 3 A real number in fixed-point format, where fixed integer and fractional bits define its range and precision

Fixed-point arithmetic encodes real numbers by assigning fixed bit widths for the integer and fractional parts, as illustrated in Fig. 3. This approach relies fundamentally on integer operations, often making it simpler and faster than floating-point arithmetic in hardware implementations. Denoting by i and f the bit widths for the integer and fractional parts, respectively, any real number r in this system lies within

$$-(2^{i-1} - 2^{-f}) \leq r \leq 2^{i-1} - 2^{-f}, \quad (20)$$

and Δr , the resolution of r , is

$$\Delta r = 2^{-f}. \quad (21)$$

Figure 4 depicts how bit-space requirements differ for fixed-point addition and multiplication. For addition, if the integer portion exceeds the allotted bits, an overflow can occur, whereas the fractional precision remains unchanged. Consequently, no

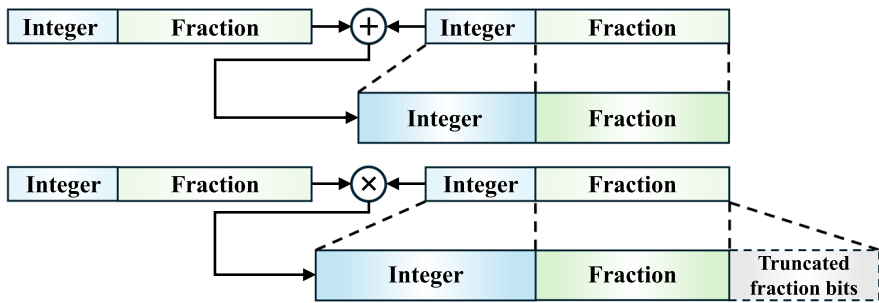


Fig. 4 Changes in bit requirements during addition and multiplication in fixed-point arithmetic. Multiplying two fixed-point numbers may require twice as many fractional bits, causing truncation in the least significant bits

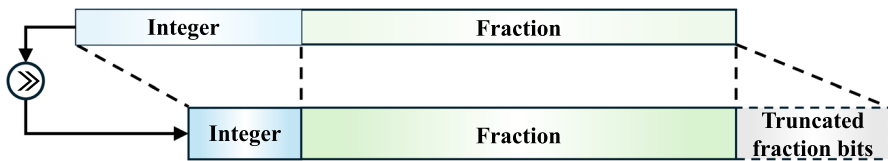


Fig. 5 Right-shifting a fixed-point bit array. The truncated fractional bits become the source of truncation error

additional truncation error is introduced during addition. In multiplication, however, both the integer and fractional parts of the product can grow beyond the original bit widths. The integer part can overflow if it exceeds the allocated bits, and the fractional part can lose bits if its precision requirement (up to 2^{-2f}) surpasses the available 2^{-f} , resulting in *truncation errors*.

In the specific case of Grover's algorithm, the diffusion operation in Eq. (12) requires accumulation of all 2^n amplitudes before scaling. In the FP-Grover setting studied in this paper, overflow is excluded by construction so that the analysis can focus on truncation due to the fractional-bit width f . The potentially largest intermediate quantity arises in this diffusion operation, where all 2^n amplitudes are summed before the 2^{-n+1} scaling. By the Cauchy–Schwarz inequality and the normalization condition in Eq. (5), this unscaled sum satisfies $\left| \sum_{m=0}^{2^n-1} \psi[m] \right| \leq \sqrt{2^n} \|\psi\|_2 = 2^{n/2}$. Accordingly, we conservatively set the integer-bit width to $i = n$ in the evaluated implementation. We do not claim that $i = n$ is the minimum possible width; rather, it is a sufficient overflow-free choice for the parameter regime considered in this work. Under the experimental settings considered in Sect. 4, the representable range in Eq. (20) is sufficiently large for the diffusion accumulator, so overflow does not occur. The oracle operation in Eq. (11) only flips signs and therefore does not increase magnitude. Hence, the analysis in Sect. 3 focuses on truncation errors induced by f .

Scaling operations (multiplying by powers of two) can often be handled by *bit-shifting* instead of a more expensive multiplication logic (i.e., *multipliers*, the high HW resource consumer). Indeed, left-shifting by a bits corresponds to multiplying by 2^a , while right-shifting by a bits corresponds to multiplying by 2^{-a} . As illustrated in

Fig. 5, right-shifting can remove low-order fractional bits, thus generating truncation errors.

Mathematically, if a real r in the arithmetic operation is represented as r_{FP} in fixed-point form with f , then the truncation error ϵ satisfies

$$r = r_{FP} + \epsilon, \quad \epsilon = r \bmod 2^{-f}. \quad (22)$$

Hence, r_{FP} is a multiple of 2^{-f} , while the residual ϵ captures the portion of r that exceeds this precision limit. In the upcoming sections, we will analyze how such truncation errors accumulate when Grover's algorithm is emulated under fixed-point arithmetic and evaluate their collective impact on the final measurement probability distribution.

3 Theoretical analysis of truncation errors in FP-Grover emulation

3.1 Preservation of quantum-state structure under Grover's iterations

From Eq. (8) in Sect. 2, Grover's operator G selectively amplifies the amplitudes of the basis states corresponding to solutions. Moreover, from Eq. (10), the initial state of Grover's algorithm assigns identical amplitudes to all basis states. Consequently, at any step in Grover's algorithm, the amplitudes of all solution states remain equal, and the amplitudes of all nonsolution states remain equal (albeit possibly at a different value). Formally, for an n -qubit system, let $|\psi\rangle$ be the state of the system at any iteration. Then the amplitude of the m^{th} basis state, $\psi[m]$, can be written as

$$\psi[m] = \begin{cases} \alpha_S, & m \in S, \\ \alpha_{NS}, & \text{otherwise,} \end{cases} \quad m = 0, 1, \dots, 2^n - 1, \quad (23)$$

where S is the set of solution indices.

In what follows, we leverage this simplified representation—the idea that any intermediate state has only two distinct amplitude values—to analyze truncation errors. We first prove, via mathematical induction, that Grover's algorithm indeed preserves the form in Eq. (23) at every iteration.

Base case: Let $|\psi_0\rangle = H^{\otimes n}|0^n\rangle$ be the initial state. From Eq. (10), all basis states share the same amplitude

$$\psi_0[m] = \frac{1}{\sqrt{2^n}}, \quad m = 0, 1, \dots, 2^n - 1. \quad (24)$$

Hence $|\psi_0\rangle$ satisfies the structure of Eq. (23), with $\alpha_S = \alpha_{NS} = 1/\sqrt{2^n}$.

Induction step: Assume that, after l iterations,

$$\psi_l[m] = \begin{cases} \alpha_{l,S}, & m \in S, \\ \alpha_{l,NS}, & \text{otherwise.} \end{cases} \quad (25)$$

We must show the same holds for iteration $l + 1$. By definition,

$$|\psi_{l+1}\rangle = G|\psi_l\rangle = (H^{\otimes n} Z_{\text{or}} H^{\otimes n})(Z_f)|\psi_l\rangle. \quad (26)$$

First, let $|\psi'_l\rangle = Z_f|\psi_l\rangle$. Since Z_f multiplies the amplitudes of solution states by -1 while leaving nonsolution states unchanged (see Eq. (11)), the resulting state $|\psi'_l\rangle$ again contains only two distinct amplitude values (with solution amplitudes negated). Applying the diffusion operator $H^{\otimes n} Z_{\text{or}} H^{\otimes n}$ defined in Eq. (12) to $|\psi'_l\rangle$ computes the sum of all amplitudes, scales it by 2^{-n+1} , and then subtracts each original amplitude individually, yielding:

$$\begin{aligned} |\psi_{l+1}\rangle &= G|\psi_l\rangle = (H^{\otimes n} Z_{\text{or}} H^{\otimes n})|\psi'_l\rangle \\ &= \left(\begin{bmatrix} 2^{-n+1} & \dots & 2^{-n+1} \\ \vdots & \ddots & \vdots \\ 2^{-n+1} & \dots & 2^{-n+1} \end{bmatrix} - I_n \right) |\psi'_l\rangle \\ &= 2^{-n+1} \begin{bmatrix} \sum_{h=0}^{2^n-1} \psi'_l[h] \\ \vdots \\ \sum_{h=0}^{2^n-1} \psi'_l[h] \end{bmatrix} - |\psi'_l\rangle \end{aligned} \quad (27)$$

One can verify by direct inspection that this preserves the two-amplitude structure:

$$\begin{aligned} \psi_{l+1}[m] &= 2^{-n+1} \sum_{h=0}^{2^n-1} \psi'_l[h] - \psi'_l[m] \\ &= \begin{cases} 2^{-n+1} \sum_{h=0}^{2^n-1} \psi'_l[h] + \alpha_{l,S}, & m \in S, \\ 2^{-n+1} \sum_{h=0}^{2^n-1} \psi'_l[h] - \alpha_{l,NS}, & \text{otherwise.} \end{cases} \end{aligned} \quad (28)$$

Thus, by mathematical induction, the state after each iteration l continues to have exactly two distinct amplitude values, as per Eq. (23).

3.2 Numerical formulation of truncation error accumulation

When using f fractional bits in fixed-point arithmetic, any portion of a real number below 2^{-f} is truncated, which ultimately leads to distortions in the measured probabilities of a QC emulator. In this subsection, we develop a formal framework to track how these *truncation errors* accumulate in FP-Grover emulation.

Setup: To avoid notation ambiguity, we first summarize the conventions used throughout Sects. 3.2 and 3.3. A single prime ($'$) denotes the quantity immediately after applying the oracle Z_f , whereas a double prime ($''$) denotes the quantity after

applying the full Grover operator G . The subscript FP denotes a fixed-point quantity, the subscript “scaled” denotes the output of the diffusion-scaling step, and the subscripts S and NS refer to solution and nonsolution amplitudes, respectively. For error notation, the Latin letter e denotes the error in the full-state notation, either as the full 2^n -dimensional error vector $|e_l\rangle$ or as its basis-indexed component form $e_l[m]$, whereas the Greek letter ϵ denotes the corresponding reduced scalar error values under the two-amplitude representation, such as $\epsilon_{l,S}$ and $\epsilon_{l,NS}$. Let $|\psi_l\rangle = G^l(H^{\otimes n}|0^n\rangle)$ denote the exact quantum state after l applications of the Grover operator G . We represent each amplitude of $|\psi_l\rangle$ in fixed-point form and gather them into a vector $|\psi_{l,FP}\rangle$. Since truncation arises during each iteration of G and propagates forward, let $|e_l\rangle$ capture the *cumulative* truncation error up to the l^{th} iteration. Formally,

$$|\psi_l\rangle = |\psi_{l,FP}\rangle + |e_l\rangle. \quad (29)$$

From Sect. 3.1, each Grover state takes the form in Eq. (23), so both $|\psi_{l,FP}\rangle$ and $|e_l\rangle$ retain two-amplitude values according to whether $m \in S$ or $m \notin S$:

$$\psi_{l,FP}[m] = \begin{cases} \alpha_{l,FP,S}, & m \in S, \\ \alpha_{l,FP,NS}, & \text{otherwise,} \end{cases} \quad (30)$$

$$e_l[m] = \begin{cases} \epsilon_{l,S}, & m \in S, \\ \epsilon_{l,NS}, & \text{otherwise.} \end{cases} \quad (31)$$

Thus, $e_l[m]$ is the basis-indexed component of the error vector $|e_l\rangle$, while $\epsilon_{l,S}$ and $\epsilon_{l,NS}$ are the two reduced scalar values taken by this component for solution and nonsolution basis states, respectively.

State update: We now expand $|\psi_{l+1}\rangle = G|\psi_l\rangle$ in fixed-point form:

$$\begin{aligned} |\psi_{l+1}\rangle &= G|\psi_{l,FP}\rangle + G|e_l\rangle \\ &= (H^{\otimes n}Z_{\text{or}}H^{\otimes n})Z_f|\psi_{l,FP}\rangle + (H^{\otimes n}Z_{\text{or}}H^{\otimes n})Z_f|e_l\rangle \\ &= (H^{\otimes n}Z_{\text{or}}H^{\otimes n})|\psi'_{l,FP}\rangle + (H^{\otimes n}Z_{\text{or}}H^{\otimes n})|e'_l\rangle \\ &= |\psi''_{l,FP}\rangle + |e''_l\rangle, \end{aligned} \quad (32)$$

where $|\psi'_{l,FP}\rangle = Z_f|\psi_{l,FP}\rangle$ and $|e'_l\rangle = Z_f|e_l\rangle$.

Detailed recurrences: Consider first the fixed-point state vector $|\psi''_{l,FP}\rangle = G|\psi_{l,FP}\rangle$. Applying the diffusion operator $(H^{\otimes n}Z_{\text{or}}H^{\otimes n})$ involves summing all 2^n amplitudes, scaling by 2^{-n+1} , and subtracting each original amplitude:

$$\begin{aligned} |\psi''_{l,FP}\rangle &= (H^{\otimes n}Z_{\text{or}}H^{\otimes n})|\psi'_{l,FP}\rangle \\ &= 2^{-n+1} \begin{bmatrix} \sum_{h=0}^{2^n-1} \psi'_{l,FP}[h] \\ \vdots \\ \sum_{h=0}^{2^n-1} \psi'_{l,FP}[h] \end{bmatrix} - |\psi'_{l,FP}\rangle \end{aligned} \quad (33)$$

Addition and subtraction do not themselves degrade fractional precision in fixed-point arithmetic, but *scaling* introduces a truncation error, denoted $\epsilon'_{l,scaled}$.

Because $|\psi'_{l,FP}\rangle$ follows Eq. (30), we can factor out the number of solutions n_s . For example,

$$\begin{aligned} 2^{-n+1} \sum_{h=0}^{2^n-1} \psi'_{l,FP}[h] &= 2^{-n+1} \left((2^n - n_s) \alpha_{l,FP,NS} - n_s \alpha_{l,FP,S} \right) \\ &= 2 \alpha_{l,FP,NS} - 2^{-n+1} n_s (\alpha_{l,FP,NS} + \alpha_{l,FP,S}), \end{aligned} \quad (34)$$

which, in a practical fixed-point implementation, corresponds to a right bit-shift—specifically, the accumulated sum of the 2^n probability amplitudes is stored in a fixed-point number with an n -bit integer part, and the scaling operation is then performed by applying an $(n - 1)$ -bit right shift to this value—that may discard fractional bits below 2^{-f} . We define

$$\epsilon'_{l,scaled} = \left(2^{-n+1} \sum_{h=0}^{2^n-1} \psi'_{l,FP}[h] \right) \bmod 2^{-f}, \quad (35)$$

reflecting the truncated remainder.

Combined fixed-point value and truncation: Hence, the scaling result can be expressed as a combination of the fixed-point amplitude $\psi'_{l,FP,scaled}$ and its truncation error $\epsilon'_{l,scaled}$:

$$2^{-n+1} \sum_{h=0}^{2^n-1} \psi'_{l,FP}[h] = \alpha'_{l,FP,scaled} + \epsilon'_{l,scaled}. \quad (36)$$

Substituting Eq. (36) into Eq. (33), where $\epsilon'_{l,scaled}$ is as defined in Eq. (35), one obtains

$$\begin{aligned} \psi''_{l,FP}[m] &= 2^{-n+1} \sum_{h=0}^{2^n-1} \psi'_{l,FP}[h] - \psi'_{l,FP}[m] \\ &= \begin{cases} \alpha'_{l,FP,scaled} + \alpha_{l,FP,S} + \epsilon'_{l,scaled}, & m \in S, \\ \alpha'_{l,FP,scaled} - \alpha_{l,FP,NS} + \epsilon'_{l,scaled}, & \text{otherwise.} \end{cases} \end{aligned} \quad (37)$$

Updated error terms: Equation (32) similarly yields the updated error vector $|e''_l\rangle = G|e_l\rangle$. Namely,

$$|e''_l\rangle = (H^{\otimes n} Z_{or} H^{\otimes n}) |e'_l\rangle$$

$$= 2^{-n+1} \begin{bmatrix} \sum_{h=0}^{2^n-1} e'_l[h] \\ \vdots \\ \sum_{h=0}^{2^n-1} e'_l[h] \end{bmatrix} - |e'_l\rangle \quad (38)$$

where

$$\begin{aligned} 2^{-n+1} \sum_{h=0}^{2^n-1} e'_l[h] &= 2^{-n+1} \left((2^n - n_s) \epsilon_{l,NS} - n_s \epsilon_{l,S} \right) \\ &= 2 \epsilon_{l,NS} - 2^{-n+1} n_s (\epsilon_{l,NS} + \epsilon_{l,S}). \end{aligned}$$

Adding the scaling-induced error $\epsilon'_{l,scaled}$ finally leads to the recurrences for the next iteration $l + 1$. In particular, we have

$$\begin{aligned} \psi_{l+1,FP}[m] &= \psi''_{l,FP}[m] - \epsilon'_{l,scaled} \\ &= \begin{cases} \alpha'_{l,FP,scaled} + \alpha_{l,FP,S} & \text{if } m \in S \\ \alpha'_{l,FP,scaled} - \alpha_{l,FP,NS} & \text{otherwise,} \end{cases} \end{aligned} \quad (39)$$

$$\begin{aligned} e_{l+1}[m] &= e''_l[m] + \epsilon'_{l,scaled} \\ &= \begin{cases} 2\epsilon_{l,NS} + \epsilon_{l,S} + \epsilon'_{l,scaled} \\ \quad - 2^{-n+1} n_s (\epsilon_{l,NS} + \epsilon_{l,S}) & \text{if } m \in S \\ \epsilon_{l,NS} + \epsilon'_{l,scaled} \\ \quad - 2^{-n+1} n_s (\epsilon_{l,NS} + \epsilon_{l,S}) & \text{otherwise,} \end{cases} \end{aligned} \quad (40)$$

with initial values

$$\psi_{0,FP}[m] = \frac{1}{\sqrt{2^n}} - e_0[m], \quad e_0[m] = \left(\frac{1}{\sqrt{2^n}} \right) \bmod 2^{-f}. \quad (41)$$

Iterating Eqs. (39) and (40) through $l = k$ yields the final state $|\psi_{k,FP}\rangle$ and the accumulated error $|e_k\rangle$ in the FP-Grover emulation.

Approximate scaling behavior: To see how $|e_k\rangle$ scales with n and f , consider sufficiently large n and small n_s , so $2^{-n+1} n_s \approx 0$. Under this approximation, the update Eq. (40) reduces to

$$e_{l+1}[m] = \begin{cases} 2\epsilon_{l,NS} + \epsilon_{l,S} + \epsilon'_{l,scaled}, & m \in S, \\ \epsilon_{l,NS} + \epsilon'_{l,scaled}, & \text{otherwise.} \end{cases}$$

Hence,

$$\epsilon_{k,NS} = \epsilon_{k-1,NS} + \epsilon'_{k-1,scaled}$$

$$\begin{aligned}
&= \epsilon_{k-2,NS} + \epsilon'_{k-2,scaled} + \epsilon'_{k-1,scaled} \\
&= \dots = \epsilon_{0,NS} + \sum_{h=0}^{k-1} \epsilon'_{h,scaled}.
\end{aligned} \tag{42}$$

Since $\epsilon_{0,NS} < 2^{-f}$ and each $\epsilon'_{h,scaled} < 2^{-f}$ (from Eqs. (41) and (35)), we have

$$\epsilon_{k,NS} = O(2^{-f}) + O(k \cdot 2^{-f}) = O(2^{\frac{n}{2}-f}), \tag{43}$$

where $k \approx O(2^{\frac{n}{2}})$ by Eq. (9). Meanwhile, the solution error $\epsilon_{k,S}$ grows even more aggressively:

$$\begin{aligned}
\epsilon_{k,S} &= \epsilon_{k-1,S} + (2\epsilon_{k-1,NS} + \epsilon'_{k-1,scaled}) \\
&= \dots = \epsilon_{0,S} + \sum_{m=0}^{k-1} (2\epsilon_{m,NS} + \epsilon'_{m,scaled}) \\
&= O(k^2 2^{-f}) = O(2^{n-f}).
\end{aligned} \tag{44}$$

Therefore, the scaling behavior of $\epsilon_{k,NS}$ and $\epsilon_{k,S}$ with respect to n and f can be summarized as follows:

$$\epsilon_{k,NS} = O(2^{\frac{n}{2}-f}), \quad \epsilon_{k,S} = O(2^{n-f}). \tag{45}$$

These results will be central to our analysis in Sect. 3.3 on the total measurement error in the final FP-Grover state.

3.3 Impact of accumulated errors on measurement probability distribution

We now use the big-O bounds derived in Sect. 3.2 for the errors in the quantum-state coefficients, $\epsilon_{k,NS}$ and $\epsilon_{k,S}$ (Eq. (45)), to determine how these errors ultimately distort the *measurement probability distribution* produced by the FP-Grover emulation. Conceptually, the severity of an emulation error corresponds to the discrepancy between its output probability distribution and the ideal distribution. Hence, we adopt the ℓ_2 distance between the true probabilities and the emulated probabilities as a representative metric, which we refer to as the ℓ_2 error.

Definition of ℓ_2 error. Let $p[m]$ be the ideal probability of measuring basis state m in the final state after k Grover iterations, and let $p_{FP}[m]$ be the corresponding probability under fixed-point emulation. We define the ℓ_2 error as

$$\ell_2 = \sqrt{\sum_{m=0}^{2^n-1} (p[m] - p_{FP}[m])^2}. \tag{46}$$

Here, $p[m]$ follows from Eq. (4), which states

$$\begin{aligned} p[m] &= |\psi_k[m]|^2 = (\psi_{k,FP}[m] + e_k[m])^2 \\ &= (\psi_{k,FP}[m])^2 + 2\psi_{k,FP}[m]e_k[m] + (e_k[m])^2. \end{aligned} \quad (47)$$

Although $(\psi_{k,FP}[m])^2$ itself may exceed the fixed-point precision 2^{-f} , we capture this mismatch by a *square* truncation error $e_{sq}[m]$:

$$p_{FP}[m] = (\psi_{k,FP}[m])^2 - e_{sq}[m], \quad (48)$$

$$e_{sq}[m] = ((\psi_{k,FP}[m])^2) \bmod 2^{-f}. \quad (49)$$

Then,

$$p[m] = p_{FP}[m] + e_{sq}[m] + 2\psi_{k,FP}[m]e_k[m] + (e_k[m])^2, \quad (50)$$

so the total truncation error in $p[m]$ is

$$e_p[m] = e_{sq}[m] + 2\psi_{k,FP}[m]e_k[m] + (e_k[m])^2. \quad (51)$$

Scaling behavior of probability errors: From Eq. (49), we know $e_{sq}[m] = O(2^{-f})$. Moreover, if $m \notin S$, we have $|\psi_{k,FP}[m]| = |\alpha_{k,FP,NS}|$, which can be expressed as:

$$\begin{aligned} |\alpha_{k,FP,NS}| &\approx |\alpha_{k,NS}| = \frac{|\cos((2k+1)\theta)|}{\sqrt{2^n - n_s}} \\ &< \frac{\sin \theta}{\sqrt{2^n - n_s}} = \frac{\sqrt{n_s}}{\sqrt{2^n} \sqrt{2^n - n_s}}, \\ \text{where } \theta &= \sin^{-1} \sqrt{\frac{n_s}{2^n}}. \end{aligned} \quad (52)$$

Thus, we conclude $|\alpha_{k,FP,NS}| = O(2^{-n})$. Conversely, for $m \in S$, $|\psi_{k,FP}[m]| = |\alpha_{k,FP,S}| \approx n_s^{-1}$, effectively a constant $O(1)$ with respect to n and f . Including the bounds for $\epsilon_{k,NS}$ and $\epsilon_{k,S}$ derived in Sect. 3.2, we obtain:

$$\begin{aligned} \epsilon_{sq,NS} &= O(2^{-f}), & \epsilon_{sq,S} &= O(2^{-f}), \\ |\alpha_{k,FP,NS}| &= O(2^{-n}), & |\alpha_{k,FP,S}| &= O(1), \\ \epsilon_{k,NS} &= O(2^{\frac{n}{2}-f}), & \epsilon_{k,S} &= O(2^{n-f}). \end{aligned}$$

Substituting these into Eq. (51) isolates the scaling behavior of the probability error $e_p[m]$. For nonsolution states:

$$\epsilon_{p,NS} = O(2^{-f}) \pm O(2^{-n}) O(2^{\frac{n}{2}-f}) + O(2^{2(\frac{n}{2}-f)})$$

$$= O(2^{-f}) \pm O(2^{-\frac{n}{2}-f}) + O(2^{n-2f}). \quad (53)$$

If $n < f$, then $O(2^{-f})$ dominates, so

$$\epsilon_{p,NS} = O(2^{-f}). \quad (54)$$

Likewise, for solution states,

$$\begin{aligned} \epsilon_{p,S} &= O(2^{-f}) + O(1) O(2^{n-f}) + O(2^{2(n-f)}) \\ &= O(2^{n-f}). \end{aligned} \quad (55)$$

ℓ_2 error of the final distribution: Because $e_p[m]$ follows the same structure as in Eq. (23), we can substitute it directly into Eq. (46):

$$\ell_2 = \sqrt{\sum_{m=0}^{2^n-1} (e_p[m])^2} = \sqrt{(2^n - n_s) \epsilon_{p,NS}^2 + n_s \epsilon_{p,S}^2}. \quad (56)$$

Assuming $n_s \ll 2^n$, we apply the bounds from Eqs. (54)–(55) to obtain

$$\begin{aligned} \ell_2 &= \sqrt{O(2^n) O(2^{-2f}) + O(2^{2(n-f)})} \\ &\approx \sqrt{O(2^{2(n-f)})} = O(2^{n-f}). \end{aligned} \quad (57)$$

Hence, each additional fractional bit (increasing f by 1) effectively halves ℓ_2 , whereas each extra qubit (increasing n by 1) doubles it, provided n_s remains relatively small compared to 2^n . This completes our theoretical analysis, giving a clear guideline on how n and f jointly drive the magnitude of truncation-induced measurement errors in FP-Grover emulation.

4 Experimental validation and design guidelines for precision optimization

4.1 Verification of the proposed truncation error formulation

To confirm that the theoretical ℓ_2 error analysis derived in previous sections accurately reflects the actual behavior of an FP-Grover emulation, we must verify that the *theoretical* errors match the *measured* errors in a practical implementation. Here, the objective is not to re-evaluate the ideal closed-form final state of Grover's algorithm, which is already available from Eq. (18), but to validate the iteration-by-iteration arithmetic behavior of a fixed-point emulator executing repeated Grover updates under ideal gate operations. Accordingly, this section isolates truncation as an independent arithmetic-error source and does not model physical gate imperfections or device-level noise. To this end, we designed Algorithm 1, which computes the final amplitudes

Algorithm 1 Arithmetic procedures for emulating Grover's algorithm using fixed-point and double-precision floating-point expressions, enabling direct evaluation of the ℓ_2 truncation error.

Input n, f, n_s n : number of qubits, f : number of fractional bits, n_s : number of solutions Output $amp_{fixed,NS}, amp_{fixed,S}$ $amp_{fixed,NS}, amp_{fixed,S}$: fixed-point expression of state amplitude <pre> 1: procedure GROVER_FIXED 2: int $amp_{fixed,NS}, amp_{fixed,S} = \lfloor 2^{-n/2+f} \rfloor$ 3: int sum 4: int $k = \lfloor \frac{\pi}{4} \sqrt{\frac{2^n}{n_s}} \rfloor$ 5: for $i = 0$ to $k - 1$ do 6: $amp_{fixed,S} = -amp_{fixed,S}$ 7: $sum = (2^n - n_s) * amp_{fixed,NS} + n_s * amp_{fixed,S}$ 8: $amp_{fixed,NS} = sum \gg (n - 1) - amp_{fixed,NS}$ 9: $amp_{fixed,S} = sum \gg (n - 1) - amp_{fixed,S}$ 10: end for 11: return $amp_{fixed,NS}, amp_{fixed,S}$ 12: end procedure </pre>	Input n, n_s n : number of qubits, n_s : number of solutions Output amp_{NS}, amp_S amp_{NS}, amp_S : double-precision reference values of the state amplitudes <pre> 1: procedure GROVER_DOUBLE 2: double $amp_{NS}, amp_S = 2^{-n/2}$ 3: double sum 4: int $k = \lfloor \frac{\pi}{4} \sqrt{\frac{2^n}{n_s}} \rfloor$ 5: for $i = 0$ to $k - 1$ do 6: $amp_S = -amp_S$ 7: $sum = (2^n - n_s) * amp_{NS} + n_s * amp_S$ 8: $amp_{NS} = 2^{-n+1} * sum - amp_{NS}$ 9: $amp_S = 2^{-n+1} * sum - amp_S$ 10: end for 11: return amp_{NS}, amp_S 12: end procedure </pre>
---	--

Input n, f, n_s
 n : number of qubits, f : number of fractional bits, n_s : number of solutions

Output ℓ_2
 ℓ_2 : distance between resulting probability distribution and the reference distribution

```

1: procedure GROVER_L2
2:   int  $amp_{fixed,NS}, amp_{fixed,S} = GROVER\_FIXED(n, f, n_s)$ 
3:   double  $amp_{NS}, amp_S = GROVER\_DOUBLE(n, n_s)$ 
4:   double  $prob_{fixed,NS}, prob_{fixed,S}, prob_{NS}, prob_S$ 
5:    $prob_{fixed,NS} = (\text{double})(amp_{fixed,NS}^2 \gg f) * 2^{-f}$ 
6:    $prob_{fixed,S} = (\text{double})(amp_{fixed,S}^2 \gg f) * 2^{-f}$ 
7:    $prob_{NS} = amp_{NS}^2, prob_S = amp_S^2$ 
8:   return  $\ell_2 = \sqrt{(2^n - n_s)(prob_{NS} - prob_{fixed,NS})^2 + n_s(prob_S - prob_{fixed,S})^2}$ 
9: end procedure

```

of Grover's algorithm and evaluates the resulting ℓ_2 error via both fixed-point and double-precision floating-point expressions. Double-precision floating-point is used as a numerical reference, since it follows the same iterative execution flow without fixed-point truncation and thereby isolates the effect of fractional-bit quantization. Overflow is not modeled as a separate error source in this section, because the evaluated FP-Grover implementation operates in the overflow-free setting described in Sect. 2.3, using a fixed integer-bit allocation chosen to exclude overflow by construction.

Algorithm 1: The procedure GROVER_FIXED computes the final quantum-state amplitudes ($amp_{fixed,NS}, amp_{fixed,S}$) in a fixed-point representation, while GROVER_DOUBLE obtains the double-precision reference amplitudes (amp_{NS}, amp_S). Both subroutines implement k iterations of Grover's operator G (see Eq. (8)). This iterative execution is essential because the truncation analyzed in this work is generated at each repeated fixed-point diffusion-scaling step, and is therefore not revealed by a single evaluation of the closed-form ideal-state expression alone. After computing these amplitudes, the procedure GROVER_L2 calculates the measured ℓ_2 error

by comparing the resulting fixed-point probabilities to the reference probabilities:

$$\ell_2 = \left\{ (2^n - n_s) (prob_{NS} - prob_{fixed,NS})^2 + n_s (prob_S - prob_{fixed,S})^2 \right\}^{\frac{1}{2}},$$

corresponding to Eq. (46).

Algorithm 2: We also implement L2_THEORETICAL, which computes the *theoretical* ℓ_2 error based on the formulas derived in Sects. 3.2 and 3.3. Concretely, this algorithm:

1. Uses Eqs. (39) and (40) to iteratively compute the final fixed-point amplitudes $(\alpha_{k,FP,NS}, \alpha_{k,FP,S})$ and the accumulated truncation errors $(\epsilon_{k,NS}, \epsilon_{k,S})$.
2. Evaluates the measurement probability error $(\epsilon_{p,NS}, \epsilon_{p,S})$ via Eq. (51).
3. Substitutes these results into Eq. (56) to produce the theoretical ℓ_2 error value.

Experimental results: We tested Algorithms 1 and 2 by varying $n = 8, 12, 16$, $f = 16, 17, \dots, 40$, and setting $n_s = 1$. The integer-bit width was set to n for all experiments, ensuring overflow-free arithmetic as described in Sect. 2.3. For each parameter set (n, f, n_s) , we computed the measured ℓ_2 error by running the actual fixed-point emulator (Algorithm 1), and separately computed the theoretical ℓ_2 error using the analytical formulation implemented in Algorithm 2. The two values are

Algorithm 2 Arithmetic procedure that implements the proposed theoretical framework for computing the ℓ_2 error in FP-Grover emulation.

```

1: procedure L2_THEORETICAL
2:   INPUT :  $(n, f, n_s)$ 
3:   double  $\alpha_{k,FP,NS}, \alpha_{k,FP,S} = 2^{-n/2} - \text{mod}(2^{-n/2}, 2^{-f})$ 
4:   double  $\epsilon_{k,NS}, \epsilon_{k,S} = \text{mod}(2^{-n/2}, 2^{-f})$ 
5:   double  $\alpha_{old,NS}, \alpha_{old,S}, \epsilon_{old,NS}, \epsilon_{old,S}$ 
6:   double  $\alpha_{scaled}, \epsilon_{scaled}$ 
7:   int  $k = \left\lfloor \frac{\pi}{4} \sqrt{\frac{2^n}{n_s}} \right\rfloor$ ;
8:   for  $i = 0$  to  $k - 1$  do
9:      $\alpha_{old,NS} = \alpha_{k,FP,NS}, \alpha_{old,S} = \alpha_{k,FP,S}$ 
10:     $\epsilon_{old,NS} = \epsilon_{k,NS}, \epsilon_{old,S} = \epsilon_{k,S}$ 
11:     $\alpha_{scaled} = 2 * \alpha_{old,NS} - 2^{-n+1} * n_s * (\alpha_{old,NS} + \alpha_{old,S})$ 
12:     $\epsilon_{scaled} = \text{mod}(\alpha_{scaled}, 2^{-f})$ 
13:     $\alpha_{scaled} = \alpha_{scaled} - \epsilon_{scaled}$ 
14:     $\alpha_{k,FP,NS} = \alpha_{scaled} - \alpha_{old,NS}$ 
15:     $\alpha_{k,FP,S} = \alpha_{scaled} + \alpha_{old,S}$ 
16:     $\epsilon_{k,NS} = \epsilon_{old,NS} + \epsilon_{scaled} - 2^{-n+1} n_s (\epsilon_{old,NS} + \epsilon_{old,S})$ 
17:     $\epsilon_{k,S} = 2\epsilon_{old,NS} + \epsilon_{old,S} + \epsilon_{scaled} - 2^{-n+1} n_s (\epsilon_{old,NS} + \epsilon_{old,S})$ 
18:  end for
19:   $\epsilon_{sq,NS} = \text{mod}(\alpha_{k,FP,NS}^2, 2^{-f})$ 
20:   $\epsilon_{sq,S} = \text{mod}(\alpha_{k,FP,S}^2, 2^{-f})$ 
21:   $\epsilon_{p,NS} = \epsilon_{sq,NS} + 2 * \alpha_{k,FP,NS} * \epsilon_{k,NS} + \epsilon_{k,NS}^2$ 
22:   $\epsilon_{p,S} = \epsilon_{sq,S} + 2 * \alpha_{k,FP,S} * \epsilon_{k,S} + \epsilon_{k,S}^2$ 
23:  OUTPUT :  $\ell_2 = \sqrt{(2^n - n_s) \epsilon_{p,NS}^2 + n_s \epsilon_{p,S}^2}$ 
24: end procedure

```

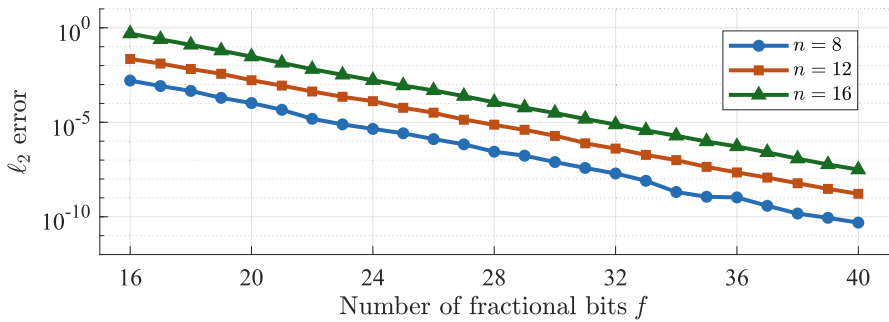


Fig. 6 ℓ_2 error versus the number of fractional bits f for $n = 8, 12, 16$. For every tested parameter combination, the measured ℓ_2 error from Algorithm 1 and the theoretical ℓ_2 error from Algorithm 2 coincide; hence, only one visible curve appears for each n . The vertical axis is logarithmic

exactly identical across all tested parameter combinations. Accordingly, the measured and theoretical results overlap completely and appear as a single visible curve for each n in Fig. 6. Figure 6 plots this common ℓ_2 value versus the fractional-bit count f on a semi-log scale.

Notably, Fig. 6 shows linear decreases of the error curves for each n . These linear trends confirm the *exponential* falloff predicted by the big-O relationship derived in Eq. (57). To numerically verify these observations, Table 2 partitions the ℓ_2 error data into intervals of width $w_{prec} = 4$ in terms of f and computes the average *scaling ratio* within each interval. For an interval ending at fractional bit $f = a$, the average scaling ratio $r_{prec}(a)$ is defined as

$$r_{prec}(a) = \left(\frac{\ell_2|_{f=a}}{\ell_2|_{f=a-w_{prec}}} \right)^{\frac{1}{w_{prec}}}, \quad (58)$$

which quantitatively characterizes how quickly the ℓ_2 error decreases as f increases. Across all intervals tested, the computed ratio consistently yields a value of $r_{prec}(a) \approx 0.5$, a result also confirmed visually in Fig. 7, which plots r_{prec} versus varying f for each n . This demonstrates that each additional fractional bit approximately halves the ℓ_2 error.

Similarly, as n increases, Fig. 6 shows that the ℓ_2 error curves shift upward by approximately a constant factor. To quantify this observation, Table 2 also presents the average scaling ratio $r_{num_q}(f)$, defined as

$$r_{num_q}(f) = \left(\frac{\ell_2|_{n=16}}{\ell_2|_{n=8}} \right)^{\frac{1}{16-8}}, \quad (59)$$

which measures how quickly the error grows as n increases from 8 to 16. Across all f 's, $r_{num_q}(f)$ consistently remains close to 2, aligning with our theoretical prediction that each additional qubit effectively doubles the ℓ_2 error. Hence, these empirical results

Table 2 ℓ_2 error for selected f with the range width $w = 4$, plus the average ratios $r_{prec}(f)$ and r_{num_q} within each range

n	f	16	20	24	28	32	36	40
8	ℓ_2	1.62×10^{-3}	1.04×10^{-4}	4.46×10^{-6}	2.78×10^{-7}	1.96×10^{-8}	1.06×10^{-9}	4.90×10^{-11}
	$r_{prec}(f)$	—	0.5034	0.4551	0.4997	0.5152	0.4820	0.4640
12	ℓ_2	2.25×10^{-2}	1.68×10^{-3}	1.31×10^{-4}	7.49×10^{-6}	4.05×10^{-7}	2.21×10^{-8}	1.64×10^{-9}
	$r_{prec}(f)$	—	0.5225	0.5287	0.4890	0.4821	0.4835	0.5221
16	ℓ_2	4.94×10^{-1}	2.98×10^{-2}	1.68×10^{-3}	1.14×10^{-4}	7.60×10^{-6}	5.24×10^{-7}	3.12×10^{-8}
	$r_{prec}(f)$	—	0.4957	0.4869	0.5112	0.5078	0.5125	0.4940
$r_{num_q}(f)$		2.0442	2.0285	2.0981	2.1220	2.1067	2.1724	2.2415

This single bold number is the midpoint data point from the experiment.

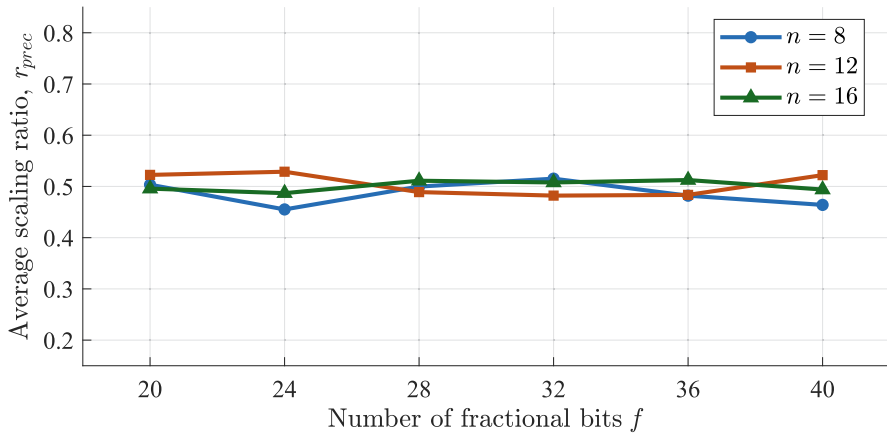


Fig. 7 Average scaling ratio of the ℓ_2 error with respect to f , evaluated for various qubit counts ($n = 8, 12, 16$)

strongly validate the scaling behavior predicted by our analytical formulation for the FP-Grover emulator.

4.2 Optimal precision selection for targeted emulation accuracy

Having experimentally validated our error model and confirmed its scaling behavior, we next leverage these insights to guide the choice of *precision* in designing the FP-Grover emulator. Specifically, our goal is to determine the minimum number of fractional bits f_{min} required to guarantee that the ℓ_2 error remains below a desired threshold $\ell_{2,max}$.

Precision formula: From Eq. (57), we assume

$$\ell_2 = a \cdot 2^{n-f}, \quad (60)$$

where a is a proportionality constant. To pin down a , we substitute specific values of (n, f, ℓ_2) from the midpoint of our empirical dataset in Sect. 4.1, namely $n = 12$, $f = 28$, $\ell_2 = 7.492 \times 10^{-6}$. Solving for a yields $a = 2^{-1.03}$, so

$$\ell_2 = 2^{n-f-1.03}, \quad (61)$$

$$\text{or equivalently, } f = n - \log_2 \ell_2 - 1.03. \quad (62)$$

Given a desired maximum error $\ell_{2,max}$, the minimal integer f_{min} becomes

$$f_{min} = \left\lceil n - \log_2 \ell_{2,max} - 1.03 \right\rceil. \quad (63)$$

Experimental validation of the formula: To confirm that Eq. (63) indeed provides a valid design rule, we tested it with $\ell_{2,max} = 10^{-3}, 10^{-5}, 10^{-7}$ for various n . As

Table 3 Minimal required fractional-bit precision (f_{min}) calculated using the proposed formula for different error thresholds ($\ell_{2,max}$), along with the corresponding verification of actual ℓ_2 errors from the FP-Grover emulator

n	$\ell_{2,max} = 10^{-3}$		$\ell_{2,max} = 10^{-5}$		$\ell_{2,max} = 10^{-7}$	
	f_{min}	ℓ_2	f_{min}	ℓ_2	f_{min}	ℓ_2
8	17	8.271×10^{-4}	24	4.459×10^{-6}	31	3.825×10^{-8}
12	21	8.742×10^{-4}	28	7.493×10^{-6}	35	4.354×10^{-8}
16	25	8.778×10^{-4}	32	7.599×10^{-6}	39	5.859×10^{-8}

These bold numbers represent the final ℓ_2 errors (a standard statistical measure used to quantify the difference between predicted and true values).

reported in Table 3, each calculated f_{min} yielded an emulator error below $\ell_{2,max}$ when plugged back into Eq. (56). Thus, the proposed formula is practically effective for determining the minimum fractional bits needed to meet an error specification.

Practical design implications: With Eq. (63), designers can quantitatively balance *accuracy* and *resource efficiency* when configuring an FP-Grover emulator. By pre-computing f_{min} , one can avoid unnecessary memory and computational overhead while ensuring the error remains below $\ell_{2,max}$. For example, in software-based FP-Grover implementations, it is crucial to select an appropriate data type (e.g., `int16` or `int32`) with the minimal memory footprint that accommodates the required fractional-bit precision. In hardware-based implementations, designers can customize register widths, arithmetic logic units (ALUs), and data-path widths based on f_{min} , minimizing hardware resource usage without compromising accuracy constraints. Thus, our method systematically guides emulator designers and system architects toward optimal precision choices, providing rigorous analysis of classical errors in quantum algorithm emulation as well as a practical resource-allocation framework for real-world FP-Grover emulator implementations.

5 Conclusion

In this work, we systematically investigated the impact of classical truncation errors on the accuracy of fixed-point arithmetic emulation for Grover's quantum search algorithm. By introducing a simplified yet rigorously justified representation of quantum states throughout Grover's iterations, we derived precise mathematical expressions to characterize how truncation errors propagate and accumulate across successive quantum gate operations. Our theoretical analysis demonstrated that truncation-induced errors in the measurement probability distributions scale exponentially as $O(2^{n-f})$, where n denotes the number of qubits and f represents the fractional-bit precision. We validated this theoretical prediction comprehensively through both numerical analysis and empirical experiments using an actual fixed-point QC emulator implementation. These empirical validations aligned precisely with our derived theoretical results, confirming the accuracy and robustness of our analytical framework. Crucially, our analysis provided a practical closed-form formula for selecting the minimal

fractional-bit precision required to meet any specified error tolerance. The reliability and applicability of this formula were verified empirically, thereby offering QC emulator designers a systematic and theoretically grounded approach to balancing computational accuracy and hardware resource constraints. Ultimately, our work bridges theoretical error analysis and practical emulator design, laying a robust foundation for precision-aware fixed-point quantum emulation of Grover's algorithm. At the same time, we note that the present study is intentionally algorithm-specific rather than universally general across arbitrary quantum circuits. The tractability of our closed-form analysis relies on the strong structural regularity of Grover's algorithm, especially its low-dimensional state evolution and repeated operator application. Extending the proposed precision-analysis methodology to other structured quantum algorithms therefore constitutes an important direction for future research.

Acknowledgements This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00345668) and in part by the Chung-Ang University Research Scholarship Grants in 2025.

Author contributions S. C.: Resources, data curation, and writing—review and editing. K. L.: Conceptualization, methodology, software, formal analysis, and writing—original draft. J. C.: Validation, visualization, and writing—review and editing. C. J.: Literature review, comparative analysis of related work, writing—revised draft, and writing—review and editing. W. L.: Conceptualization, investigation, writing—original draft and editing, supervision, and funding acquisition. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by Chung-Ang University

Data availability No datasets were generated or analyzed during the current study.

Declarations

Conflicts of interest The authors have no conflicts of interest to declare.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

References

1. Kim, Y., Eddins, A., Anand, S., Wei, K.X., Van Den Berg, E., Rosenblatt, S., Nayfeh, H., Wu, Y., Zaletel, M., Temme, K., et al.: Evidence for the utility of quantum computing before fault tolerance. *Nature* **618**(7965), 500–505 (2023)
2. Kikuchi, Y., Mc Keever, C., Coopmans, L., Lubasch, M., Benedetti, M.: Realization of quantum signal processing on a noisy quantum computer. *npj Quantum Inform.* **9**(1), 93 (2023)
3. Cranganore, S.S., De Maio, V., Brandic, I., Deelman, E.: Paving the way to hybrid quantum–classical scientific workflows. *Futur. Gener. Comput. Syst.* **158**, 346–366 (2024)

4. Gheorghiu, V., Mosca, M.: Quantum resource estimation for large scale quantum algorithms. *Futur. Gener. Comput. Syst.* **162**, 107480 (2025)
5. Google AI Quantum, Collaborators*, Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., et al.: Hartree-fock on a superconducting qubit quantum computer. *Science* **369**(6507), 1084–1089 (2020)
6. Zhang, H., Boothby, K., Kamenev, A.: Cyclic quantum annealing: searching for deep low-energy states in 5000-qubit spin glass. *Sci. Rep.* **14**(1), 30784 (2024)
7. Quantum, M.A., Aghaee, M., Alcaraz Ramirez, A., Alam, Z., Ali, R., Andrzejczuk, M., Antipov, A., Astafev, M., Barzegar, A., Bauer, B., et al.: Interferometric single-shot parity measurement in inas-al hybrid devices. *Nature* **638**(8051), 651–655 (2025)
8. Ladd, T.D., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C., O'Brien, J.L.: Quantum computers. *Nature* **464**(7285), 45–53 (2010)
9. Cao, Y., Romero, J., Olson, J.P., Degroote, M., Johnson, P.D., Kieferová, M., Kivlichan, I.D., Menke, T., Peropadre, B., Sawaya, N.P., et al.: Quantum chemistry in the age of quantum computing. *Chem. Rev.* **119**(19), 10856–10915 (2019)
10. Gyongyosi, L., Imre, S.: A survey on quantum computing technology. *Comput. Sci. Rev.* **31**, 51–71 (2019)
11. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., Biswas, R., Boixo, S., Brandao, F.G., Buell, D.A., et al.: Quantum supremacy using a programmable superconducting processor. *Nature* **574**(7779), 505–510 (2019)
12. Zhong, H.-S., Wang, H., Deng, Y.-H., Chen, M.-C., Peng, L.-C., Luo, Y.-H., Qin, J., Wu, D., Ding, X., Hu, Y., et al.: Quantum computational advantage using photons. *Science* **370**(6523), 1460–1463 (2020)
13. Kelly, J., Barends, R., Fowler, A.G., Megrant, A., Jeffrey, E., White, T.C., Sank, D., Mutus, J.Y., Campbell, B., Chen, Y., et al.: State preservation by repetitive error detection in a superconducting quantum circuit. *Nature* **519**(7541), 66–69 (2015)
14. Maurand, R., Jehl, X., Kotekar-Patil, D., Corna, A., Bohuslavskyi, H., Laviéville, R., Hutin, L., Barraud, S., Vinet, M., Sanquer, M., et al.: A CMOS silicon spin qubit. *Nat. Commun.* **7**(1), 13575 (2016)
15. Friis, N., Marty, O., Maier, C., Hempel, C., Holzäpfel, M., Jurcevic, P., Plenio, M.B., Huber, M., Roos, C., Blatt, R., et al.: Observation of entangled states of a fully controlled 20-qubit system. *Phys. Rev. X* **8**(2), 021012 (2018)
16. Ajagekar, A., You, F.: Quantum computing for energy systems optimization: challenges and opportunities. *Energy* **179**, 76–89 (2019)
17. Zhou, L., Wang, S.-T., Choi, S., Pichler, H., Lukin, M.D.: Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. *Phys. Rev. X* **10**(2), 021067 (2020)
18. Wang, Y., Kim, J.E., Suresh, K.: Opportunities and challenges of quantum computing for engineering optimization. *J. Comput. Inf. Sci. Eng.* **23**(6), 060817 (2023)
19. Cai, X.-D., Wu, D., Su, Z.-E., Chen, M.-C., Wang, X.-L., Li, L., Liu, N.-L., Lu, C.-Y., Pan, J.-W.: Entanglement-based machine learning on a quantum computer. *Phys. Rev. Lett.* **114**(11), 110504 (2015)
20. Ramezani, S.B., Sommers, A., Manchukonda, H.K., Rahimi, S., Amirlatifi, A.: Machine learning algorithms in quantum computing: a survey. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
21. et al: et al.: Lloyd, S., et al: Mohseni, M., et al: Rebentrost, P.: Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411* (2013) et al:
22. Iriyama, S., Ohya, M.: Computational complexity and applications of quantum algorithm. *Appl. Math. Comput.* **218**(16), 8019–8028 (2012)
23. Ollitrault, P.J., Miessen, A., Tavernelli, I.: Molecular quantum dynamics: a quantum computing perspective. *Acc. Chem. Res.* **54**(23), 4229–4238 (2021)
24. Outeiral, C., Strahm, M., Shi, J., Morris, G.M., Benjamin, S.C., Deane, C.M.: The prospects of quantum computing in computational molecular biology. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **11**(1), 1481 (2021)
25. Rieffel, E.G., Asanjan, A.A., Alam, M.S., Anand, N., Bernal Neira, D.E., Block, S., Brady, L.T., et al.: Assessing and advancing the potential of quantum computing: a NASA case study. *Futur. Gener. Comput. Syst.* **160**, 598–618 (2024)
26. Buluta, I., Nori, F.: Quantum simulators. *Science* **326**(5949), 108–111 (2009)

27. Häner, T., Steiger, D.S., Smelyanskiy, M., Troyer, M.: High performance emulation of quantum circuits. In: SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 866–874. IEEE (2016)
28. Mahmud, N., El-Araby, E., Caliga, D.: Scaling reconfigurable emulation of quantum algorithms at high precision and high throughput. *Quantum Eng.* **1**(2), 19 (2019)
29. Li, H., Pang, Y.: FPGA-accelerated quantum computing emulation and quantum key distillation. *IEEE Micro* **41**(4), 49–57 (2021)
30. Altman, E., Brown, K.R., Carleo, G., Carr, L.D., Demler, E., Chin, C., DeMarco, B., et al.: Quantum simulators: architectures and opportunities. *PRX Quantum* **2**, 017003 (2021)
31. El-Araby, E., Mahmud, N., Jeng, M.J., MacGillivray, A., Chaudhary, M., Nobel, M.A.I., Islam, S.I.U., Levy, D., Kneidel, D., Watson, M.R., Bauer, J.G., Riachi, A.E.: Towards complete and scalable emulation of quantum algorithms on high-performance reconfigurable computers. *IEEE Trans. Comput.* **72**(8), 2350–2364 (2023)
32. Choi, S., Lee, W.: Developing a grover's quantum algorithm emulator on standalone FPGAs: optimization and implementation. *AIMS Math.* **9**(11), 30939–30971 (2024)
33. et al: et al:Choi, S., et al.Lee, K., et al:Lee, J.-J., et al:Lee, W.: Standalone FPGA-based QAOA emulator for weighted-MaxCut on embedded devices (2025). <https://arxiv.org/abs/2502.11316> et al:
34. Beck, T., Baroni, A., Bennink, R., Buchs, G., Pérez, E.A.C., Eisenbach, M., da Silva, R.F., et al.: Integrating quantum computing resources into scientific HPC ecosystems. *Futur. Gener. Comput. Syst.* **161**, 11–25 (2024)
35. Khalid, A.U., Zilic, Z., Radecka, K.: FPGA emulation of quantum circuits. In: IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings., pp. 310–315 (2004). IEEE
36. Mahmud, N., Haase-Divine, B., Kuhnke, A., Rai, A., MacGillivray, A., El-Araby, E.: Efficient computation techniques and hardware architectures for unitary transformations in support of quantum algorithm emulation. *J. Sign. Proc. Syst.* **92**(9), 1017–1037 (2020)
37. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 212–219. (1996)
38. Bag, K., Goswami, M., Kandpal, K.: FPGA based resource efficient simulation and emulation of grover's search algorithm. In: 2022 IEEE 19th India Council International Conference (INDICON), pp. 1–6. IEEE (2022)
39. Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.: Strengths and weaknesses of quantum computing. *SIAM J. Comput.* **26**(5), 1510–1523 (1997)
40. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. *Fortschritte der Physik: Prog. Phys.* **46**(4–5), 493–505 (1998)
41. Zalka, C.: Grover's quantum searching algorithm is optimal. *Phys. Rev. A* **60**(4), 2746 (1999)
42. Preston, R.H.: Applying grover's algorithm to hash functions: a software perspective. *IEEE transactions on quantum engineering* **3**, 1–10 (2022)
43. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to aes: quantum resource estimates. In: International Workshop on Post-Quantum Cryptography, pp. 29–43. Springer (2016)
44. Schwabe, P., Westerbaan, B.: Solving binary with grover's algorithm. In: International Conference on Security, Privacy, and Applied Cryptography Engineering, pp. 303–322. Springer (2016)
45. Habibi, M.R., Golestan, S., Soltanmanesh, A., Guerrero, J.M., Vasquez, J.C.: Power and energy applications based on quantum computing: the possible potentials of grover's algorithm. *Electronics* **11**(18), 2919 (2022)
46. Bogatyrev, V., Moskvina, V.: Application of grover's algorithm in route optimization. In: 2023 Intelligent Technologies and Electronic Devices in Vehicle and Road Transport Complex (TIRVED), pp. 1–5. IEEE (2023)
47. Chakrabarty, I., Khan, S., Singh, V.: Dynamic grover search: applications in recommendation systems and optimization problems. *Quantum Inf. Process.* **16**, 1–21 (2017)
48. Pilch, J., Długopolski, J.: An FPGA-based real quantum computer emulator. *J. Comput. Electron.* **18**, 329–342 (2019)
49. Liang, S., Lu, Y., Guo, C., Luk, W., Kelly, P.H.J.: PCQ: Parallel compact quantum circuit simulation. In: 2024 IEEE 32nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) (2024)

50. Vu, T.H., Le, V.T.D., Pham, H.L., Nguyen, Q.C., Nakashima, Y.: FQsun: a configurable wave function-based quantum emulator for power-efficient quantum simulations. *IEEE Access* **13**, 93271–93286 (2025)
51. Lagostina, L., Volpe, D., Zamboni, M., Turvani, G.: AEQUAM: accelerating quantum algorithm validation through fpga-based emulation. *IEEE Access* **13**, 130232–130255 (2025)
52. Kaye, P., Laflamme, R., Mosca, M.: *An Introduction to Quantum Computing*. OUP Oxford, Oxford (2006)
53. Nielsen, M.A., Chuang, I.L.: *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge (2010)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.