Received: 17 November 2021 | Revised: 7 April 2022 | Accepted: 22 April 2022

DOI: 10.4218/etrij.2021-0442

ORIGINAL ARTICLE

ETRI Journal WILEY

QPlayer: Lightweight, scalable, and fast quantum simulator

Ki-Sung Jin | **Gyu-Il Cha**

Future Computing Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

Correspondence
Ki-Sung Jin, Future Computing Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.
Email: ksjin@etri.re.kr

Funding information
Institute of Information & Communications Technology Planning & Evaluation (IITP), Grant/Award Number: 2020-0-00014

Abstract

With the rapid evolution of quantum computing, digital quantum simulations are essential for quantum algorithm verification, quantum error analysis, and new quantum applications. However, the exponential increase in memory overhead and operation time is challenging issues that have not been solved for years. We propose a novel approach that provides more qubits and faster quantum operations with smaller memory than before. Our method selectively tracks realized quantum states using a reduced quantum state representation scheme instead of loading the entire quantum states into memory. This method dramatically reduces memory space ensuring fast quantum computations without compromising the global quantum states. Furthermore, our empirical evaluation reveals that our proposed idea outperforms traditional methods for various algorithms. We verified that the Grover algorithm supports up to 55 qubits and the surface code algorithm supports up to 85 qubits in 512 GB memory on a single computational node, which is against the previous studies that support only between 35 qubits and 49 qubits.

KEYWORDS
quantum computing, quantum information processing, quantum simulator

1 | INTRODUCTION

Quantum computers are evolving at a rapid pace, shifting from theoretical ideas to practical developments [1–4]. IBM became the first company to bring quantum computing to the public cloud, allowing users to access quantum computers remotely [5], and Google demonstrated quantum supremacy using a 54 qubits quantum processor called Sycamore [6–8]. These trends are raising expectations for innovation in various fields such as quantum chemistry, cosmology, medicine, and energy physics, which are intractable with classical computing. However, because the realization of practical quantum computers is still in its infancy, a significant number of studies in this area still rely on quantum simulations using digital quantum simulators [9].

There are already numerous digital quantum simulators available for various types of quantum algorithms [10], and they have different operation methods, features, capabilities, and limitations depending on their pursuit. This diversity of quantum simulators has so far yielded extensive results in various contexts. Nevertheless, digital simulators have an explicit limit on the number of available qubits because the required physical memory grows exponentially as the number of qubits increases [11–13]. For example, the minimum memory requirement for the quantum states of 50 qubits is 16 PB [14]. Although several studies have attempted to

This is an Open Access article distributed under the term of Korea Open Government License (KOGL) Type 4: Source Indication + Commercial Use Prohibition + Change Prohibition (http://www.kogl.or.kr/info/licenseTypeEn.do).
1225-6463/$ © 2022 ETRI

overcome these problems using distributed simulations with hundreds or more servers, these efforts do not deviate from the fundamental limitations of the required exponential increase in memory space. Currently, it is only possible to simulate 30 qubit circuits on a desktop, 35 qubit circuits on a high-end server, and 49 qubit circuits on a supercomputer.

This study focuses on the underlying causes of memory problems when describing quantum states in digital computers. After careful observation of the time evolution of quantum states for various quantum algorithms, we found that all qubits do not always exist in quantum superposition states for the following reasons: (1) Some quantum circuits apply a Hadamard gate to only a few qubits across the entire circuit, (2) applying a Hadamard gate to the superposed qubit can collapse the quantum state, and (3) intermediate measurement collapses the superposed qubit state. For these reasons, the number of actual quantum states can be smaller than 2^n, depending on the number of superposed qubits.

In this study, we propose selectively tracking solid quantum states with nonzero amplitudes by utilizing this analysis instead of loading the entire quantum states into memory. This method can dramatically reduce the number of quantum states stored in memory and increase quantum operation performance. Because our approach deals with the actual quantum states that have a physical effect, it always ensures valid quantum operation results without compromising the global quantum states.

Meanwhile, our experimental evaluation results show that the proposed idea can be applied to process various quantum algorithms. We expect our method to be efficient for simulating large-scale quantum algorithms and studying new algorithms that have previously been difficult due to the limitations of digital quantum simulators. We summarize the contributions of this study as follows:

- We present a comprehensive study of digital quantum simulation, quantum states, quantum operation, and the characteristics of quantum state evolution. We also identify significant challenges required to overcome the limitations of traditional schemes.
- A novel technique to reduce the memory requirement of general quantum algorithms using the new quantum state representation scheme is also presented. By this scheme, we confirmed that reducing memory consumption in the quantum simulation can increase the number of available qubits in an algorithm.
- Further, we present a system design and detailed empirical evaluation of the proposed idea. We demonstrate that our concept can provide more qubits and faster quantum operations with less memory than before.
- We guarantee a remarkable effect in the case of a specific quantum algorithm, such as surface code. Despite supporting larger logical qubits, our scheme provides thousands of times faster operation performance than conventional simulators.
- Finally, we provide a scale-up method to maximize the capabilities of a single server instead of scaling out to multiple servers. Nevertheless, it can provide more significant benefits than a distributed simulation, which requires massive computing resources.

We organize the rest of this paper as follows. Section 2 describes various studies on digital quantum simulations and their features. Section 3 analyzes the fundamental problems of the digital quantum simulator and defines their challenges. In Section 4, we present our concept, quantum register, quantum gates, and quantum evolutions. We also explain the benefit of the proposed idea in memory and performance. Section 5 demonstrates its validity through various experiments. To this end, we classify several algorithms into four categories according to their features and discuss the detailed meanings of each test result. Finally, Section 6 presents conclusions.

2 | RELATED WORKS

Quantum computing hardware has been steadily improving since Shor discovered the prime factorization algorithm in 1994 [15]. However, until practical quantum computers are realized, it is necessary to numerically simulate quantum circuits in digital computers to predict the behavior of quantum algorithms. Generally, we can classify digital quantum simulations into two types according to the quantum state representation scheme.

2.1 | State evolution

The most general approach is to track quantum state changes. This approach prepares the full state vector of the n-qubit consisting of a complex unit vector of dimension 2^n and applies quantum gates by performing matrix–vector multiplication in chronological order. Because it guarantees the high fidelity of quantum operations, one can use them for most universal quantum algorithms and many studies have adopted this approach [16–20]. The most remarkable advantage is that this approach can represent the full information of quantum states at any point in quantum state evolution [21]. Moreover, the main interest in recent quantum research is quantum algorithms for noisy intermediate-scale quantum (NISQ) computers [22], which require frequent intermediate

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

measurements of the surface code during circuit evolution to correct quantum errors [23–25]. The state evolution approach can always preserve the whole quantum state by ensuring fidelity under these conditions. Despite these advantages, the exponential memory growth with the number of qubits in quantum simulators restricts the scale of quantum algorithms.

2.2 | Tensor contraction

This approach for quantum simulation represents quantum circuits using a geometric or topological perspective [26–28]. A quantum simulator using this approach simulates quantum circuits through tensor network contraction. The reason is that this scheme relies on the fact that quantum circuits can always be represented as tensors; 1-qubit gate is a rank-2 tensor, 2-qubit gate is a rank-4 tensor; and n-qubit gate is a rank-2^n tensor in general [29]. Recently, the advantage of representing quantum circuits mathematically has led to active research [7,13,21,30]. In this approach, computational and memory costs depend on the highest rank tensor during contraction, and time complexity increases exponentially with the width of the graph tree. Moreover, because finding the optimal contraction order is generally known as NP-complete, it must rely on heuristic methods [31,32]. Thus, this approach only simulates low-depth circuits.

Other studies are underway to solve the memory and computational problems of digital quantum simulators. The most promising is the distributed simulation using a supercomputer that combines multiple servers. Some are general-purpose simulators using a state evolution approach, whereas others are limited-function simulators using a tensor contraction approach. The massively parallel quantum computer simulator at the University of Groningen simulated quantum supremacy circuits up to 40 qubits on a 1000 nodes TACC supercomputer [17]. qHiPSTER simulated 45 qubits by applying a scheduling technique to reduce inter-node communication costs in 8192 nodes Cori II [18]. QuEST simulated 38 qubits random circuits using a 2048 nodes ARCUS supercomputer [19]. The Quantum Supremacy Circuit Simulation of Tsinghua University implemented a quantum circuit simulator in 16 384 nodes Sunway TaihuLight, whose results show that 49 qubits with depth 39 are reachable for current universal random circuits [33]. The common goal of these studies is to maximize distributed computing capabilities by improving parallelism and reducing communication overhead through OpenMP [34] and GPU [35]. However, these techniques only focus on utilizing substantial computing resources, not eliminating the fundamental problems of digital quantum simulation. Moreover, a few researchers have access to these supercomputing environments, and there are barriers that prevent many other quantum simulation researchers from accessing them.

3 | PROBLEM ANALYSIS

As described in Section 2, state evolution and tensor contraction are typical approaches for digital quantum simulations. This study proposes a new state evolution approach to support the full quantum state and general quantum algorithms. Here, we analyze the execution load of memory and computational resources in state evolution-based simulations.

3.1 | Qubit scalability

A quantum computer generally refers to a two-level quantum system using qubits whose state can be described by $|\Psi> = \alpha|0> + \beta|1> \left(\text{where } |\alpha|^2 + |\beta|^2 = 1\right)$. Two basis states $|0>$ and $|1>$ are orthonormal and α, β are complex numbers representing the probability $p = |\alpha|^2$ and $1-p = |\beta|^2$. Measuring qubits collapses them to either $|0>$ and $|1>$ with probability p and $1-p$. If a set of n-qubits is a quantum register of size n, the global quantum state of the n-qubit quantum register can be described as in (1).

$$|\Psi_{\text{Global}}> = \sum_{i=0}^{2^n-1} \alpha_i |i>, \text{ where } \alpha_i \in C, \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1. \quad (1)$$

That is, the n-qubit quantum register requires 2^n array vectors containing amplitudes for the full quantum state. Because each amplitude is a complex number that requires two 8-byte double data types representing a real and an imaginary part, the total memory size for the n-qubit quantum register reaches 2^{n+4} bytes. Even worse, the size of the quantum space increases exponentially with the number of qubits. We call this situation "exponential explosion." For example, a 35-qubit quantum register uses 512 GB of memory; however, increasing one qubit requires 1 TB of memory twice that of 35 qubits. The simulation limit for high-end servers with less than 1 TB of memory approximates 35–36 qubits. Therefore, if we want to simulate the same scale as Google's 72-qubit processor, Bristlecone [36], on a digital computer, it will eventually require 64 ZB of memory, which is practically impossible.

Furthermore, several studies are underway to overcome this fundamental barrier using many computing

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

resources through scaling out servers. However, this approach cannot be an underlying solution because the number of servers cannot increase indefinitely. Although many researchers have attempted quantum simulations using various supercomputers with thousands of servers over the past decade, they still have not gone beyond the 49-qubit scale limit [17–19, 33].

3.2 | Operation performance

Applying quantum gates to qubits changes the amplitude of n-qubit states, that is, 2^n complex numbers of n-qubit state vectors. Because a gate operation affects the global quantum states, applying a quantum gate is equivalent to performing a single unitary matrix of $2^n \times 2^n$ dimensions on every qubit vector, which usually involves high computational load. Fortunately, because quantum matrices have sparse features with many zero-valued entries, most digital quantum simulators speed up their computations by applying a reduced matrix–vector multiplication technique as in (2). This ensures performance improvements in many quantum operations. Nevertheless, the computation time is still a challenge, because it increases exponentially in proportion to the number of qubits.

$$\text{OpCount} = \frac{2^n}{2}, \text{ where } n = \text{the number of qubits.} \quad (2)$$

In (2), OpCount refers to the number of matrix operations according to the number of qubits. Because OpCount is related to the total arithmetic calculations, increasing the number of qubits requires exponential computation. For this reason, current technologies are trying to enhance computation performance by applying software parallelization libraries such as OpenMP or hardware accelerators such as GPGPU. These approaches can relieve some performance constraints, while the actual benefit is minimal because the effect of increasing physical cores or software parallelism is relatively insignificant to handle the computations used to track 2^n exponential quantum states.

Figure 1 shows how computation time changes as the number of qubits increases. We applied the Pauli-X gate to the QuEST simulator [19] on a 56-core server. Each core simultaneously performs matrix calculation in the divided region of the full quantum state because the QuEST simulator supports parallelism through the OpenMP library. The figure shows no significant change in performance below approximately 20 qubits, but the computational load increases exponentially above 20 qubits. The main reason for the exponential rise in the processing time is because the computational size that the simulator must process in time exceeds the computational power of 56 cores.

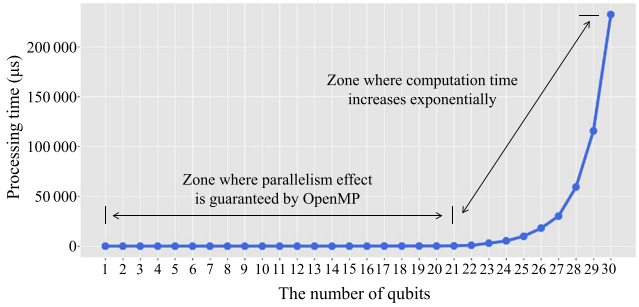

FIGURE 1 Analysis of Pauli-X gate operation time according to the number of qubits. The x- and y-axes represent the number of qubits and processing time (in microseconds), respectively

3.3 | Amplitude of quantum states

Typical quantum simulators always maintain 2^n complex vector arrays regardless of the amplitude value of each quantum state. This method makes it easy to manage changes in the entire quantum state according to quantum mechanics.

However, after carefully observing the temporal evolution of quantum states for several quantum algorithms, we focused on the fact that the number of actual, meaningful quantum states is not always 2^n. As many quantum operations progress, the number of meaningful quantum states repeatedly increase or decrease. Applying a Hadamard gate increases the number of quantum states while measuring qubits intermediately, or using a repeated Hadamard gate decreases the number of quantum states. Suppose n-qubits are initialized to $|0>$. Then, the number of the actual quantum state is only one ($\alpha|000\cdots00>$, where $\alpha = 1$), and the remaining 2^{n-1} states are unrealized quantum states having a zero-amplitude value. We refer to a quantum state with a zero-amplitude as "unrealize-state," otherwise "realized-state." Here, it is worthy to note that matrix calculations for the unrealized-state always result in the same state. Therefore, the management of matrix operations for this unrealized-state is unnecessary. Nevertheless, conventional simulators perform total matrix calculations for 2^n of entire quantum states without any consideration. If there is a way to manage only the realized-states effectively while preserving the entire quantum state, we can expect faster quantum simulation with smaller memory than traditional quantum simulators.

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

4 | QPlayer

4.1 | Design concepts

As discussed earlier, digital quantum simulators are not free from exponential explosion. So far, many studies have shown that it is theoretically impossible to completely solve the problem of exponential explosion. Therefore, we propose an alternative digital quantum simulator, the "QPlayer," to mitigate such problems instead of eliminating the exponential explosion. Our novel approach is based on the following principles:

First, we consider a digital quantum simulator having a state evolution approach that uses quantum vectors to represent quantum states. We can track quantum information at any time in circuit evolution because matrix operations are applied to quantum states at fixed locations. This can always guarantee the complete quantum states without approximating or reducing them. Moreover, it can support most general-purpose quantum gates and algorithms, just like conventional digital quantum simulators with state evolution.

Second, an effective simulator that can represent the full quantum state with only a realized-state is considered. This new simulator can support algorithms that use more qubits with less memory and computation than before. We use an algorithmic approach rather than using immense physical computing resources, so that most quantum researchers can now simulate quantum algorithms on a single server. Even for some quantum algorithms, this method can work better than using supercomputers with hundreds or more servers for certain quantum algorithms. This method also ensures the same fidelity as conventional digital quantum simulators despite using only realized-states.

Third, we consider the design of the specially optimized matrix- vector multiplication for realized-states. In traditional state evolution simulators, matrix operations are intuitive because all 2^n quantum states consist of an ordered set of array vectors. Parallelization can easily be applied by dividing the entire vector space evenly and allocating it to each core. However, because the whole quantum space consists only of realized-states, our proposed concept cannot divide all the quantum states so evenly. Thus, we specially designed an optimized mechanism to ensure parallel operations based on sophisticated control over realized-states.

Finally, we design a digital quantum simulator to ensure reversibility, a crucial concept in quantum computers. Quantum computers trace quantum state evolution through reversible operations that change the initial state of a qubit to its final form using only reversible processes [37]. All the quantum gates we provide are reversible, so every quantum circuit (qc) corresponds to a specific unit operator U_{qc} in the Hilbert space, meeting the criteria: $U_{qc}U_{qc}^{\dagger} = U_{qc}^{\dagger}U_{qc}$. There is only one irreversible element of quantum operation called measurement, which is the only way to extract useful information from qubit after the quantum computer's state acquires its final form.

4.2 | Realized-state representation

Figure 2 shows how to describe entire quantum states with only realized-states. For example, suppose that we have quantum states composed of three qubits, and all qubits are in Greenberger–Horne–Zeilinger (GHZ) states [38]. The middle refers to 2^3 theoretical quantum states that can be represented in three qubits, and eight states from $|000>$ and $|111>$ can be considered a state index with an order from 0 to 7.

Furthermore, the left is the typical amplitude array representation of state vectors in a conventional quantum state simulator. When initializing the quantum space to $|\Psi> = |000>$, 2^3 vectors of amplitude array are always prepared without considering amplitude values. Then, when the circuit evolves to the GHZ state, the quantum states change to $|\Psi> = \alpha|000> + \beta|111>$ according to the principle of quantum mechanics. As we can see from the figure, even though there are only two meaningful quantum states, quantum simulators must always maintain eight state entries. Although the remaining zero-amplitude quantum states do not affect quantum evolution or measurements, they still occupy memory space.

However, the right side of the figure is a quantum space, which is described only with realized-states. Therefore, by excluding the unrealized-states with zero-amplitude value, our simulator can support more

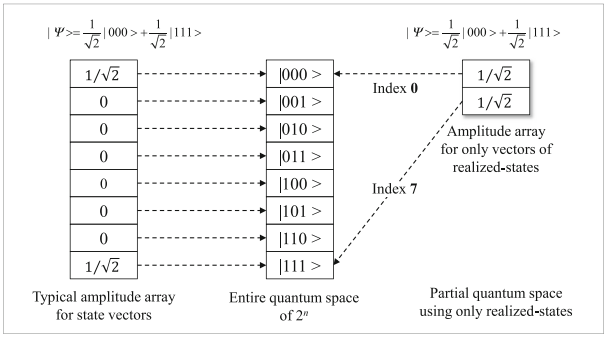

FIGURE 2 Quantum state representation using only realized-states. The center refers to a theoretical 2^n quantum space. The left refers to a typical simulator containing all amplitudes, and the right refers to our quantum representation scheme using only realized-states

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

realized-states in the same memory space. However, it is essential to understand that the global quantum states should not be distorted even if only using realized-states improves memory efficiency. Therefore, it is necessary to map the location of each realized-state to the quantum state index. This method makes it possible to accurately track all 2^3 quantum state entries. The example above can represent a quantum space with eight quantum states using only two entries.

Our idea uses a less than 2^n memory footprint because it selectively manages only realized-states. Moreover, according to the number of superposed qubits, our scheme does not always guarantee a smaller memory footprint than conventional simulations. Therefore, if all qubits are in the superposed state, the entire quantum space will eventually be 2^n. Equation (3) describes this situation.

$$M = \left(2^{\mathrm{SQ}+4} + \mathrm{QIS}\right) \left\{ \begin{array}{l} M : \text{Total memory consumption} \\ \mathrm{SQ} : \text{Number of superposed qubits} \\ \mathrm{QIS} : \text{Quantum index space} \end{array} \right\}. \tag{3}$$

The memory efficiency of our scheme is affected by the number of superposed qubits (SQ) and the quantum index space (QIS). If there is no SQ, SQ becomes zero, which means the number of all meaningful quantum states becomes one. On the contrary, if all qubits are superposed, SQ will be equal to the number of actual qubits on the typical quantum space, and the total number of quantum states will be 2^n. Moreover, QIS that is associated with the number of realized-states also increases proportionally as SQ increases. Therefore, the number of SQ mainly determines the efficiency of the memory space. The experiments in Section 5 show a significant benefit until the number of SQ is within 80% of the total number of qubits.

4.3 | Quantum register

Figure 3 shows the architecture of a quantum register for managing the global quantum states using only realized-states. The quantum register manages an index repository to store all realized-states and consists of several containers for processing quantum operations in parallel. Given an arbitrary quantum state, the quantum register block determines the appropriate container location using the equation "state index value % the number of containers." All realized-states in the quantum register are stored according to their state index values, as shown by the dotted lines in Figure 3. Entries in each container

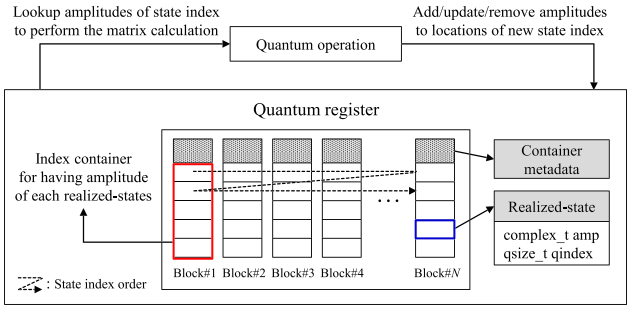

FIGURE 3 Architecture of a quantum register that stores only realized-states. It consists of several containers for fast parallel quantum operations

are stored in ascending order of their state index values, which is the same as the state order in the entire 2^n quantum space.

Each container consists of its metadata and data entries. The area of container metadata includes the number of data entries and lock information used to ensure concurrency in parallel operations. The area of data entries stores realized-states in ascending order of state index value, and each entry has an amplitude. For example, assuming the quantum state is $|\Psi> = \alpha|00010>$, the corresponding data entry of the container stores the amplitude value "α" and the quantum state index value "2." To effectively express the quantum state index, our scheme designs a "qsize_t" data type that can describe a large number, ideally by providing quantum space with 2^{1024} states.

4.4 | Quantum gates

As described in Section 4.1, the digital quantum simulators should support a wide variety of general quantum algorithms. Our simulator also provides quantum gates, as shown in Table 1.

4.5 | State evolution

In quantum computing, quantum gates are applied in matrix form to global quantum states. Applying a single qubit gate U to the kth qubit is described as a $2^n \times 2^n$ unitary transformation $I^{\otimes n-k-1} \otimes U \otimes I^{\otimes k}$. However, most typical simulators generally use the reduced matrix–vector multiplication technique because it is costly to apply matrix operations to the full quantum state of 2^n. But a matrix operation for a 2×2 matrix still must performed 2^{n-1} times.

Figure 4 shows the difference between the proposed method and other conventional methods in performing a

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 Quantum gates list

Gates	Meaning	Gates	Meaning
I	Idle or identity	RX	*X*-axis rotation
X	Pauli-*X* (bit flip)	RY	*Y*-axis rotation
Y	Pauli-*Y* (bit + phase flip)	RZ	*Z*-axis rotation
Z	Pauli-*Z* (phase flip)	*S*	SQRT(*Z*) phase
H	Hadamard	*T*	SQRT(*S*) phase
CX	Controlled-NOT	*T*+	Conjugate of *T*
CZ	Controlled-Phase	*S*+	Conjugate of *S*
CY	Controlled-Y	SWAP	Swap
CCX	Toffoli	*M*	Measurement

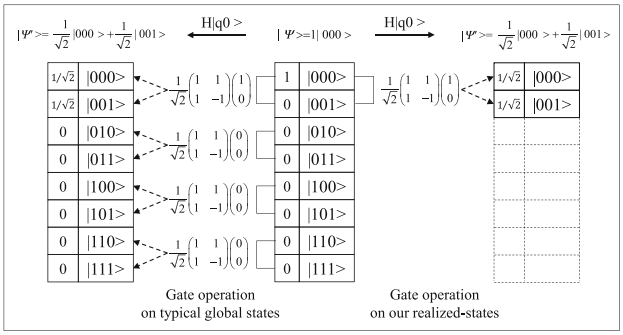

FIGURE 4 Gate operation comparison. The left side is gate operations on the typical entire state vector, and the right side is gate operations on our realized-states

one-qubit gate operation. For example, suppose we apply the Hadamard gate to the first qubit (q0) on the global quantum state $|\Psi>=|000>$.

The left side illustrates when a Hadamard gate is applied to the first qubit using the state evolution technique. This intuitively performs a 2×2 matrix computation on all pairs of target states regardless of the amplitude value. On the contrary, our method (the right side) performs a selective matrix operation on the realized-states, which is more efficient. Matrix operations applied to unrealized-states always produce zero vectors; therefore, excluding these calculations can increase the efficiency of the simulator. Our simulator computes only the quantum state where at least one of the state pairs is in the realized-state. Because each matrix operation requires multiplying four times and adding two times, the total number of calculations can be described as follows:

$$\text{Calculations} = \frac{2^{SQ}}{2} \times (4 \times \text{Multiply} + 2 \times \text{Add}), \qquad (4)$$

where SQ is the number of superposed qubits. In contrast to typical simulators requiring 2^{n-1} operations, our scheme is only affected by the number of SQ. That means, the proposed scheme can guarantee a smaller number of SQ, thus, a more significant performance improvement.

4.6 | Operation algorithm

In the previous section, we explained quantum operations using only realized-states with an intuitive example. The digital quantum simulator should always perform all quantum operations assuming a logical 2^n quantum space. However, applying a gate operation to an arbitrary qubit over the quantum algorithm is not straightforward because unrealized-states with a zero-amplitude value cause a state index hole in the entire 2^n quantum space. Therefore, an algorithm must be designed to guarantee the accuracy of quantum operations using only realized-states.

Algorithm 1 shows the operational procedure implementing a one-qubit gate using only the realized-states. Inputs of this algorithm are a target qubit number and a 2×2 matrix.

ALGORITHM 1 One-qubit gate operation

```
Function    ApplyOneQubitGate
Input       QubitNum: the number of target qubit
            M: 2 × 2 matrix of { m00, m01, m10, m11 }
1   def lowerRS : low order RS(realized-state) in the 2 × 2 matrix pair
2   def upperRS : high order RS in the 2 × 2 matrix pair
3   def ampL(U) : current RS's amplitude value
4   def newAmpL(U) : calculated RS's new amplitude value
5   for each RS in quantum register
6       determine lowerRS and upperRS
7       if already applied states pair
8           Continue
9       end if
10      set amplitudes of lowerRS and upperRS
11          if state index exists
12              set ampL(U) ← lower(upper)->amplitude()
13          else
14              set ampL(U) ← zero
15          end if
16      perform matrix calculation
17          newAmpL = m00*ampL+m01*ampU
18          newAmpU = m10*ampL+m11*ampU
19      update new amplitudes of lowerRS and upperRS
20          if ampL(ampU) is zero
21              remove old states or just skip if not exist
22          else ampL(ampU) is greater then zero
23              update old states or add new realized-states
24          end if
25  end for
```

To explain further, we refer to abbreviate a realized-state as RS. The algorithm sequentially performs all RSs in the quantum register, such as in line 5. Because all RSs are stored in the state index order, the logical execution order guarantees the index order of the logical 2^n

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

quantum space. In line 6, we first determine two state pairs for matrix calculation by shifting the current RS index and the given qubit number. In lines 7 and 9, we inspect the evolution of the selected states pair. Further, the completion of the matrix computation for the target pair goes back to the beginning of the loop statement. In lines 10–15, we look up the amplitude of each RS pair, which is related to matrix calculations. If the RS does not exist, the amplitude is set to zero. The 2×2 matrix is applied to two amplitude pairs at lines 16–18. Finally, the newly computed amplitudes are updated to the global quantum states, as shown in lines 19–24. At this point, the calculated amplitude value and the presence of the target RS determine the update method. According to the newly calculated amplitude, RS entries are added (zero → no zero), modified (no zero → no zero), or deleted (no zero → zero).

ALGORITHM 2 Two-qubit controlled gate operation

```
Function   ApplyTwoQubitGate
Input      Control: the number of control qubit
           Target: the number of target qubit
           CM: 2 × 2 matrix of { m00, m01, m10, m11 }
1   for each realized-state(RS) in quantum register
2       if state bit of control qubit is |0 >
3           continue
4       else
5           apply CM to Target as like 4–21 of ALGORITHM1
6       end if
7   end for
```

Algorithm 2 shows the operational procedure of the 2-qubit controlled gate. Input parameters include two qubits: a control qubit and a target qubit. In the 2-qubit controlled gate, matrix operations are applied to the target qubit only when the state bit of the control qubit is $|1>$. Therefore, we perform matrix operations on the states pair of the target qubit only if the state bit of the control qubit is $|1>$ in lines 2–6.

4.7 | Parallelism

Even if quantum states are described only with realized-states, avoiding the increasing computational cost is challenging, because the number of matrix calculations increases in proportion to the number of SQ; therefore, performance improvement through parallelization is essential. Consequently, we apply the OpenMP library to our simulator to support parallel calculations through multiple CPU cores. However, we assign index containers to the OpenMP thread because quantum space cannot be statically divided like a typical state vector simulator. To maximize computing resource utilization, we define the number of index containers as twice the number of CPU cores. Furthermore, because the state pairs for matrix operations sometimes span two containers, it guarantees concurrency through locks defined in each container.

5 | EXPERIMENTS AND EVALUATION

5.1 | Experimental setup

In this section, we simulate various quantum algorithms using QuEST and our newly developed state evolution simulator, QPlayer, and analyze the results from many viewpoints.

5.1.1 | Hardware environment

Experiments were performed on a Dell PowerEdge T640 single server with two Intel Xeon Gold 6132 CPUs (56 cores total) and 512 GB of memory. We did not use hardware acceleration devices, such as GPGPU, but we used only the OpenMP library provided by Linux for the simulation parallelism.

5.1.2 | Benchmark environment

Digital quantum simulators play a key role by supporting many general quantum algorithms; thus, we define four algorithm categories and analyze their simulation results to verify that our simulator works properly for general algorithms: (1) QASMBench [38] provides many quantum algorithms with various quantum circuit sizes. The benchmark of QASMBench is summarized in Table 2, and it is divided into three categories (small, medium, and large scales) according to the number of quantum qubits. This was used to evaluate the performance of digital quantum simulators compatible with general-purpose quantum gates. (2) Other generic algorithms commonly used in quantum computing, including Grover search [39] and quantum Fourier transform (QFT) [40]. (3) Surface code [23–25] is a quantum error correction algorithm that generates highly reliable logical qubits using physical quantum qubits with errors, as shown in Figure 5A. For the beyond-NISQ era, research in this field has been rapidly growing in recent years. (4) Based on Arute et al. [8], random circuit sampling (RCS) is an algorithm for randomized quantum circuit testing, as shown in Figure 5B.

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 2 QASMBench quantum circuits

Small scale			Medium scale			Large scale		
Benchmark	**Description**	**Qubits (gates)**	**Benchmark**	**Description**	**Qubits (gates)**	**Benchmark**	**Description**	**Qubits (gates)**
deutsch	Deutsch algorithm	2 (5)	bb84	Key distribution	8 (27)	cat_state	Quant. arithmetic	22 (22)
qaoa	QAOA algorithm	3 (15)	ising	Ising model	10 (480)	mulipler	Quant. multiplier	25 (3723)
qec	Error correction	5 (25)	seca	Error correction	11 (216)	bwt	Quant. arithmetic	21 (112806)
teleport	Teleportation	3 (8)	bv	Bernstein	14 (56)	ghz_state	GHZ prep.	23 (23)
lpn	Learning parity	5 (11)	simon	Simon algorithm	9 (123)	sqare	Calc. Square root	26 (280)

Note: According to the number of qubits, algorithms are divided into three categories.

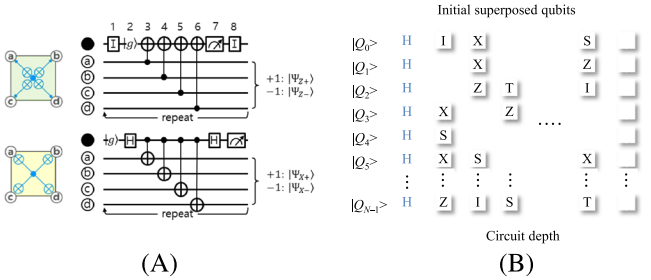

FIGURE 5 (A) The surface code algorithm, (B) example circuit for the RCS algorithm

5.2 | Experimental analysis

5.2.1 | QASMBench

Figure 6A shows the memory consumption when executing QASMBench algorithms. Their ideal memory requirement is only a few MB according to the 2^n rule because small- and medium-scale algorithms use less than 20 qubits. In this case, QuEST's memory consumption is similar to or slightly less than that of QPlayer. Even if the number of qubits is small or medium, QPlayer requires additional memory to manage state index space for realized-states. However, in large-scale algorithms, the memory consumption of QPlayer becomes much smaller than that of the QuEST. The effect of tracking only realized-states outweighs the additional memory cost required to manage the state index.

Figure 6B shows the execution time of each algorithm. The execution times of QPlayer and QuEST are similar to memory consumption patterns on a small to medium scale. But, in the large-scale algorithms, QPlayer's computational performance is much better than that of the QuEST. As illustrated in Figure 1, exponential explosions in more than 20 qubits significantly reduce the parallel execution effect in QuEST. However, QPlayer shows relatively higher performance than QuEST, even on a large scale. Numerically, QPlayer is 1.4 times faster in cat, 37 times faster in multiplier, five times faster in bwt, 1.3 times faster in ghz, and 1.8 times faster in sqaure_root compared with QuEST.

5.2.2 | Generic algorithms

The Grover algorithm, designed to apply superposed states to only a subset of all qubits, searches for a specific quantum state among many quantum states. Figure 7 shows the change in the simulation time as the number of qubits gradually increases. As a result,

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

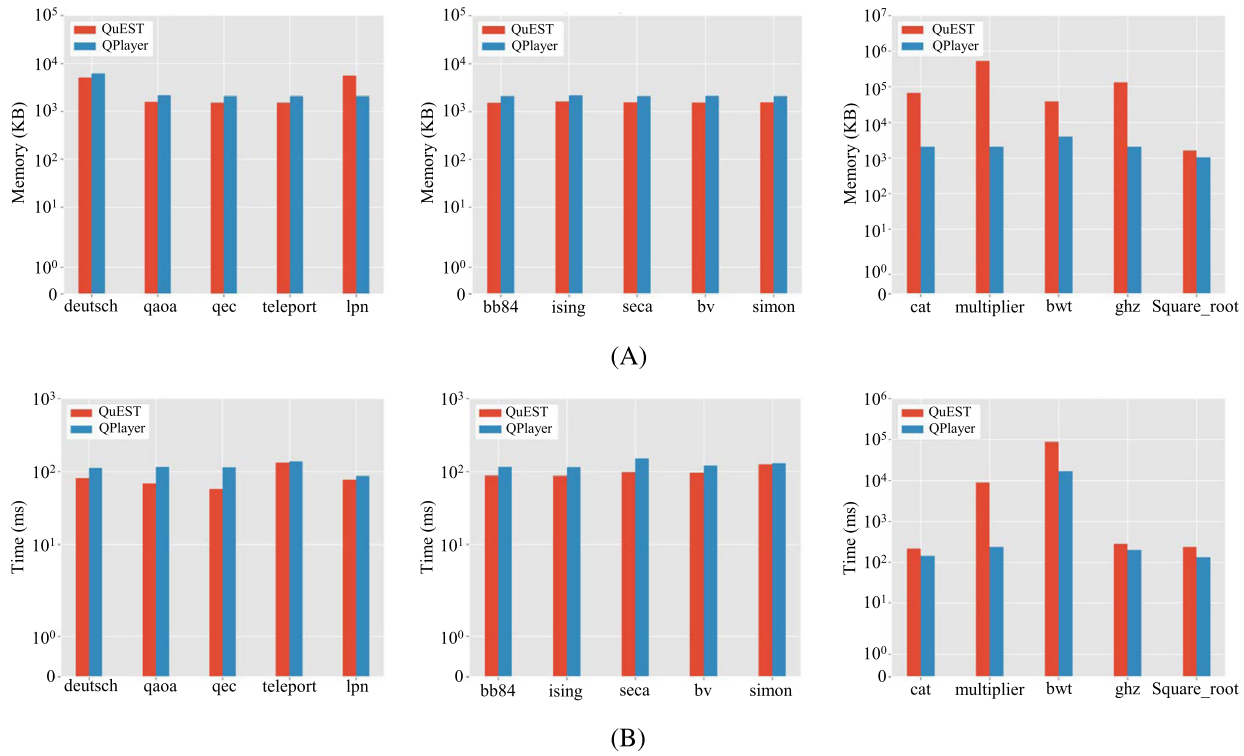

FIGURE 6 Experimental result of QASMBench according to the quantum algorithm and the number of qubits: (A) shows memory consumption for each quantum circuit, and (B) shows quantum circuit operation time

two significant consequences were obtained. First, QuEST cannot support more than 35 qubits as it always requires a 2^n quantum space regardless of the state amplitude. However, because QPlayer minimally keeps track of superposed states, it can even support Grover's algorithm with a scale of 55 qubits. Second, at the same qubit scale, QPlayer is significantly faster than QuEST. For example, the execution of Grover with 35 qubits takes 1.2 s for QPlayer and 6471 s for QuEST indicating that QPlayer is approximately 5000 times faster than QuEST.

As shown in Figure 8, the trend with which the simulation time of the QFT algorithm changes is slightly different from that of the Grover algorithm. All qubits used in this simulation gradually change to superposed states with circuit evolution in the QFT algorithm. As a result, QPlayer supports roughly 33 qubits under similar conditions, while QuEST supports 35 qubits. The difference is because QPlayer requires an additional index space for the realized-states. In other words, QPlayer also uses 2^n realized-states, just like all quantum spaces, as all qubits eventually evolve into a superposition state. Comparing the performance at the identical 33 qubits takes 841 s for QuEST and 3256 s for QPlayer, indicating that QuEST is about four times faster than QPlayer.

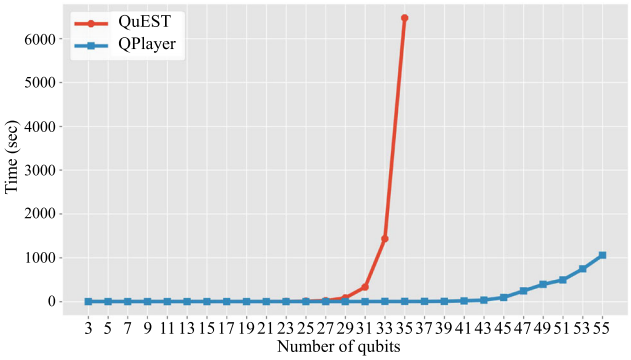

FIGURE 7 Comparison in the execution time of the Grover algorithm

5.2.3 | Surface code

The surface code is an algorithm for quantum error correction where several qubits are combined and used as one logical qubit. To generate one logical qubit, SC13 requires 13 qubits, and SC17 requires 17 qubits [24]. As the number of logical qubits increases, the required physical qubits multiply by (the number of logical qubits × 9) + 4 for SC13 and (the number of logical qubits × 17) for SC17. The important thing about the surface code is that the number of realized-states of the logical qubit

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

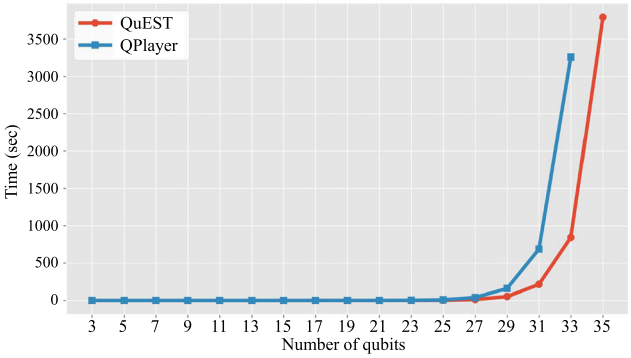

FIGURE 8 Comparison in the execution time of the QFT algorithm

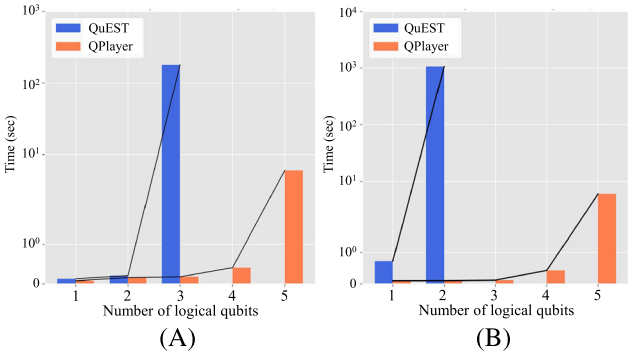

FIGURE 9 Logical qubit scalability in (A) SC13 and (B) SC17

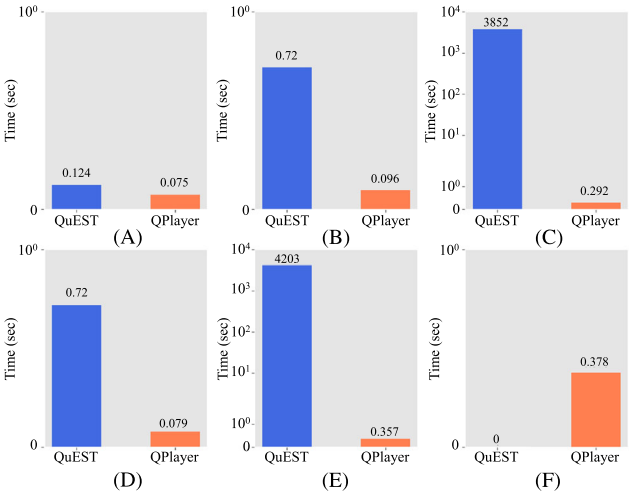

FIGURE 10 Comparison in logical operation performance: (A) SC13 ESM, (B) SC13 teleport (C) SC13 cnot, (D) SC17 ESM, (E) SC17 teleport, and (F) SC17 cnot

is not 2^n. The surface code fixes all quantum states into stabilized states, where the number of stabilized states required for a single logical qubit is only 2^4 or 2^5. To build an SC17-based logical qubit, QuEST requires quantum space with 2^{17} quantum states, while QPlayer uses only quantum space with 2^5 quantum states.

Figure 9 shows the performance comparison in generating up to five logical qubits using surface code. As described above, to generate five logical qubits, SC13 requires 49 qubits, and SC17 requires 85 qubits. In Figure 9, QPlayer supports up to five logical qubits for both SC13 and SC17, whereas QuEST can only generate three logical qubits for SC13 and two logical qubits for SC17. It should be noted that QuEST only supports up to 35 qubits due to physical memory limitations on a single server. Besides, for the same number of logical qubits, QPlayer provides significantly faster performance than QuEST. For example, generating three logical qubits with SC13 takes 179 s for QuEST and 0.173 s for QPlayer, indicating that QPlayer is about 1032 times faster.

Figure 10 compares the performance of logical operations in surface code on QuEST and QPlayer. The Error Syndrome Measurement (ESM) detects errors in one logical qubit, Teleport transfers quantum information between two logical qubits, and CNOT is a logical controlled-not operation using three logical qubits [23–25]. As shown in Figure 10, QPlayer performs better in all operations than QuEST. Notably, the increase in used qubits widens the performance gap in the following order: ESM, Teleport, and CNOT operations. In ESM operation, QPlayer is 2.1 times faster in SC13 and 9 times faster in SC17 than QuEST. QPlayer has an excellent performance of 8 times faster in SC13 and 11 773 times faster in the Teleport operation of SC17. In CNOT operation, QPlayer is 13 192 times faster in SC13 than QuEST.

As confirmed by our analysis, QPlayer reveals innovative features for limited algorithms such as surface code. Most quantum error correction algorithms use only a few stabilized states and require precise tracking of quantum states with frequent intermediate measurements. Given these conditions, QPlayer can always represent the perfect logical quantum states of 2^n only using some realized-states.

5.2.4 | RCS

RCS is a technique for building a quantum circuit through random gate placement, as shown in Figure 5B. It can cover a variety of quantum circuit scenarios from the worst case to the best case in the perspective of simulation cost because RCS does not have a specific structure of circuit gates. It is even possible to analyze the behavioral pattern of quantum algorithms by setting specific parameter conditions. Table 3 shows the RCS circuit's simulation results in which the superposition ratio of the

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 3 Random circuit sampling simulation: 30 qubits, circuit depth 20

Assigned superposed qubit ratio	Execution time			Memory consumption			Number of quantum states	
	QuEST	QPlayer		QuEST	QPlayer		QuEST	QPlayer
0%	88 s	119 ms	739×	16 GB	2 MB	8000×	1 073 741 824	1
10%	89 s	130 ms	677×	16 GB	2 MB	8000×	1 073 741 824	8
20%	89 s	161 ms	547×	16 GB	2 MB	8000×	1 073 741 824	64
30%	88 s	208 ms	423×	16 GB	2 MB	8000×	1 073 741 824	512
40%	88 s	282 ms	312×	16 GB	2 MB	8000×	1 073 741 824	4096
50%	88 s	434 ms	203×	16 GB	2 MB	8000×	1 073 741 824	32 768
60%	90 s	2 s	44×	16 GB	52 MB	315×	1 073 741 824	262 144
70%	91 s	19 s	5×	16 GB	387 MB	42×	1 073 741 824	2 097 152
80%	88 s	244 s	3× (slow)	16 GB	2.8 GB	6×	1 073 741 824	16 777 216
90%	89 s	1218 s	14× (slow)	16 GB	24 GB	2× (high)	1 073 741 824	134 217 728
100%	90 s	1 h 17 min	53× (slow)	16 GB	129 GB	8× (high)	1 073 741 824	1 073 741 824

qubits is assigned as a parameter. Each simulation set the superposed qubit ratio within the range of 0%–100% and applied 30 qubits at a circuit depth of 20.

QuEST takes a similar execution time of about 90 s regardless of the superposition qubit ratio. However, QPlayer has different performance patterns depending on the superposed qubit ratio. Compared with the result of QuEST, QPlayer performs approximately hundreds of times faster at a 0%–50% superposed ratio and dozens of times faster at a 60%–70% superposed ratio.

However, when the superposed qubit ratio reaches ≥80%, the number of realized-states approaches 2^n, which increases the index management cost, so QPlayer is somewhat slower than QuEST.

Meanwhile, a similar pattern is shown in memory consumption. QuEST always uses 16 GB of memory, while QPlayer uses 2 MB at the superposed qubit ratio of less than 50%, resulting in approximately 8000 times memory savings. However, in the 90%–100% ratio, QPlayer uses two to eight times more memory. This difference is because QuEST always processes 2^n quantum states, while QPlayer has the different number of quantum states depending on the number of SQ.

6 | CONCLUSIONS

In this paper, we addressed fundamental questions of limitations of digital quantum simulators. Over the past decade, researchers are yet to resolve the exponential explosion problem of memory and computation in digital quantum simulators. Here, we proposed a novel simulator, called QPlayer. It provides more qubits and faster quantum operations with smaller memory than before. Our simulator selectively tracks limited realized-states instead of loading the full quantum state into memory. Our empirical evaluation showed that QPlayer provides more robust scalability and high performance than the state-of-the-art digital quantum simulators. We demonstrated its effectiveness with several quantum algorithms in QASMBench and verified that the simulation of Grover algorithms is possible with 55 qubits. QPlayer supports up to five logical qubits using surface code-17, corresponding to 85 physical qubits. In the RCS experiment, we found significant benefits at the superposed qubit ratio of less than 80%.

In future studies, further optimization for QPlayer and several new ideas should be explored as follows: (1) The efficient management of realized-states to further reduce memory usage and increase simulation speed, (2) support of industry-standard interface in quantum computing such as OpenQASM [41] to improve compatibility with the general quantum algorithm, (3) the study of quantum noise models with quantum errors such as bit flip, dephase, and decoherence.

You can obtain both a free copy of QPlayer test programs by (1) contacting the first author or (2) visiting https://github.com/eQuantumOS/QPlayer.

ACKNOWLEDGMENTS

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00014, A Technology Development of Quantum OS for Fault-tolerant Logical Qubit Computing Environment).

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest.

ORCID

Ki-Sung Jin https://orcid.org/0000-0002-1997-3019

REFERENCES

1. B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, and A. G. White, *Towards quantum chemistry on a quantum computer*, Nat. Chem. **2** (2010), 106–111.
2. M. H. Devoret and R. J. Schoelkopf, *Superconducting circuits for quantum information: an outlook*, Science **339** (2013), 1169–1174.
3. B. P. Lanyon, C. Hempel, D. Nigg, M. Müller, R. Gerritsma, F. Zähringer, and C. F. Roos, *Universal digital quantum simulation with trapped ions*, Science **334** (2011), no. 6502, 57–61.
4. J. Casanova, A. Mezzacapo, L. Lamata, and E. Solano, *Quantum simulation of interacting fermion lattice models in trapped ions*, Phys. Rev. Lett. **108** (2012), 190502.
5. D. Castelvecchi, *IBM's quantum cloud computer goes commercial*, Nature **543** (2017), no. 7644, 159.
6. R. Courtland, *Google aims for quantum computing supremacy*, IEEE Spectr. **54** (2017), no. 6, 9–10.
7. L. Gomes, *Quantum computing: Both here and not here*, IEEE Spectr. **55** (2018), no. 4, 42–47.
8. F. Arute, K. Arya, R. Babbush, and J. M. Martinis, *Quantum supremacy using a programmable superconducting processor*, Nature **574** (2019), no. 7779, 505–510.
9. A. Zulehner and R. Wille, *Advanced simulation of quantum computations*, IEEE Tran. Comput.-Aided Des. Integr. Circuits Syst. **38** (2018), no. 5, 848–859.
10. *List of QC simulators grouped by programming language*, 2021, Available from: https://quantiki.org/wiki/list-qc-simulators [last accessed August, 2021].
11. I. Buluta and F. Nori, *Quantum simulators*, Science **326** (2009), no. 5949, 108–111.
12. R. P. Feynman, *Simulating physics with computers*, Theor. Phys. **21** (1982), 467–488.
13. J. Doi, H. Takahashi, R. Raymond, T. Imamichi, and H. Horii, *Quantum computing simulator on a heterogenous HPC system, (Quantum computing simulator on a heterogenous HPC system, Alghero, Italy)*, Apr. 2019, pp. 85–93.
14. J. Chen, F. Zhang, and C. Huang, *Classical simulation of intermediate-size quantum circuits*, arXiv preprint, 2018. https://doi.org/10.48550/arXiv.1805.01450
15. P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, (Proceedings 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA), Nov. 1994, pp. 124–134.
16. H. Thomas and D. S. Steiger, *0.5 petabyte simulation of a 45-qubit quantum circuit*, (Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA), 2017, pp. 1–10.
17. H. Raedt, K. Michielsen, H. B. Trieu, G. Arnold, M. Richter, and N. Ito, *Massively parallel quantum computer simulator*, Comput. Phys. Comm. **176** (2007), no. 2, 121–136.
18. M. Smelyanskiy, N. P. Sawaya, and A. Aspuru-Guzik, *qHiPSTER: The quantum high performance software testing environment*, arXiv preprint, 2016. https://doi.org/10.48550/arXiv.1601.07195
19. T. Jones, A. Brown, I. Bush, and S. C. Benjamin, *QuEST and high performance simulation of quantum computers*, Sci. Rep. **9** (2019), no. 1, 1–11.
20. H. Raedt, F. Jin, D. Willsch, M. Willsch, N. Yoshioka, N. Ito, and K. Michielsen, *Massively parallel quantum computer simulator, eleven years later*, Comput. Phys. Comm. **237** (2019), 47–61.
21. E. S. Fried, N. P. Sawaya, Y. Cao, I. D. Kivlichan, J. Romero, and A. Aspuru-Guzik, *qTorch: The quantum tensor contraction handler*, PloS one **13** (2018). https://doi.org/10.1371/journal.pone.0208510
22. J. Preskill, *Quantum computing in the NISQ era and beyond*, Quantum **2** (2018). https://doi.org/10.22331/q-2018-08-06-79
23. A. G. Fowler, A. C. Whiteside, and L. C. Hollenberg, *Towards practical classical processing for the surface code*, Phys. Rev. Lett. **108** (2012). https://doi.org/10.1103/PhysRevLett.108.180501
24. Y. Tomita and K. M. Svore, *Low-distance surface codes under realistic quantum noise*, Phys. Rev. A **90** (2014), no. 6. https://doi.org/10.1103/PhysRevA.90.062320
25. A. Erhard, H. P. Nautrup, M. Meth, L. Postler, R. Stricker, M. Stadler, and T. Monz, *Entangling logical qubits with lattice surgery*, Nature **589** (2021), no. 7841, 220–224.
26. G. F. Viamontes, I. L. Markov, and J. P. Hayes, *Graph-based simulation of quantum computation in the density matrix representation*, Quantum Inf. Comput. II **5436** (2004), 285–296.
27. I. L. Markov and Y. Shi, *Simulating quantum computation by contracting tensor networks*, SIAM J. Comput. **38** (2008), no. 3, 963–981.
28. J. Biamonte and V. Bergholm, *Tensor networks in a nutshell*, arXiv preprint, 2017. https://doi.org/10.48550/arXiv.1708.00006
29. S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven, *Simulation of low-depth quantum circuits as complex undirected graphical models*, arXiv preprint, 2017. https://doi.org/10.48550/arXiv.1712.05384
30. Z. Y. Chen, Q. Zhou, C. Xue, X. Yang, G. C. Guo, and G. P. Guo, *64-qubit quantum circuit simulation*, Sci. Bull. **63** (2018), no. 15, 964–971.
31. S. Arnborg, D. G. Corneil, and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM J. Algebraic Discrete Methods **8** (1987), no. 2, 277–284.
32. E. Amir, *Approximation algorithms for treewidth*, Algorithmica **56** (2010), no. 4, 448–479.
33. R. Li, B. Wu, M. Ying, X. Sun, and G. Yang, *Quantum supremacy circuit simulation on Sunway TaihuLight*, IEEE Trans. Parallel Distri. Syst. **31** (2019), no. 4, 805–816.
34. K. S. Jin, S. M. Lee, and Y. C. Kim, *Adaptive and optimized agent placement scheme for parallel agent-based simulation*, ETRI J. **44** (2021), 313–326. https://doi.org/10.4218/etrij.2020-0399
35. Y. W. Kim, M. H. Oh, and C. Y. Park, *Multi-communication layered HPL model and its application to GPU clusters*, ETRI J. **43** (2021), no. 3, 524–537.
36. B. M. Terhal, *Quantum supremacy, here we come*, Nat. Phys. **14** (2018), no. 6, 530–531.

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

37. M. Noorallahzadeh and M. Mosleh, *Efficient designs of reversible latches with low quantum cost*, IET Circ. Dev. Syst. **13** (2019), no. 6, 806–815.
38. A. Li and S. Krishnamoorthy, *QASMBench: A low-level QASM benchmark suite for NISQ evaluation and simulation*, arXiv preprint, 2020. https://doi.org/10.48550/arXiv.2005.13018
39. L. K. Grover, *A fast quantum mechanical algorithm for database search*, (Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA), 1996, pp. 212–219. https://doi.org/10.1145/237814.237866
40. M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2010.
41. OpenQASM 3.x Live Specification. https://qiskit.github.io/openqasm [last accessed October, 2021].

AUTHOR BIOGRAPHIES

Ki-Sung Jin received his BS and MS degrees in Computer Engineering from Jeonbuk National University, Jeonju, Republic of Korea, in 1999 and 2001, respectively. Since 2001, he has been with the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, where he has worked on developing the cluster database, distributed parallel filesystem, dual-mode big data platform, and simulation technology for the digital twin. He is currently a principal researcher. His current research interests include distributed systems, extreme storage systems, and quantum operating systems.

Gyu-Il Cha received his BS and MS degrees in Computer Science from Korea University, Seoul, Republic of Korea, in 1998 and 2000, respectively. Since 2000, he has been with the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, and is currently a principal researcher. His research interest is a quantum operating system for fault-tolerant quantum computing. He has been involved in the technology development of operating systems, memory virtualization, supercomputing, microservice architectures, and extreme storage systems.

How to cite this article: K.-S. Jin and G.-I. Cha, *QPlayer: Lightweight, scalable, and fast quantum simulator*, ETRI Journal **45** (2023), 304–317. https://doi.org/10.4218/etrij.2021-0442

22337326, 2023, 2, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2021-0442 by Readcube (Labtiva Inc.), Wiley Online Library on [25/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License