INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelor Thesis

Efficient Simulation of Peaked Random Quantum Circuits using Tensornetworks

Martin Brieger

Assessor:	Prof. Dr. D. Kranzlmüller
Supervisor:	Florian Krötz
Date final version:	February 23, 2026

Efficient Simulation of Peaked Random Quantum Circuits using Tensornetworks

Martin Brieger[†1]

[1] *MNM-Team, Faculty of Mathematics, Informatics and Statistics, LMU Munich*

February 23, 2026

Abstract

Peaked Random Circuits (PRCs) are a novel class of quantum circuits promising to be classically hard, yet easily verifiable and easily computable on current NISQ-era quantum devices [1]. This makes PRCs an ideal tool to proof quantum supremacy, given no classical speedup is found. While early spoofing attempts using generic simulators have proven unfruitful [2, 3], rigorous simulation is needed to validate or refute this result. This paper provides the groundwork for this effort. First, I introduce four proposals for generating PRCs and their peculiarities. Second, I discuss the simulability of PRCs with tensornetwork formalisms, including parallelization strategies and methods for recovering the peaked bitstring. Third, I provide experimental evidence suggesting that approximate tensornetwork simulators outperform other techniques at simulating PRCs. Fourth, I demonstrate on a quantum computer that PRCs can in principle serve as a quantum-advantage protocol.

Keywords: quantum-advantage, peaked circuit, random circuit, tensornetwork, CUDA-Q, cuQuantum

[†] brieger@cip.ifi.lmu.de

Contents

1 Introduction

Building a quantum computer that outperforms the most powerful classical supercomputers in practical applications represents a major scientific frontier. Despite recent advances in logical qubits [4, 5], current noisy intermediate-scale (NISQ) era quantum devices exhibit high error rates, which necessitate substantial qubit overhead for effective correction and limit computations to small scales [6]. Short coherence times constrain algorithm depth and reliability, as environmental noise rapidly decoheres quantum states before complex operations complete [7, 8]. Foundational challenges in hardware manufacturing [9], control systems [10], and real-time decoding [11] need to be resolved before fault-tolerant, thousand-and-beyond-qubit systems become a reality.

While this ambitious technological goal is being pursued with great effort, a "side quest" has emerged – primarily among corporate actors – to demonstrate a quantum-advantage on today's near-term devices. Famously, research teams at Google and later Xanadu and USTC claimed that their quantum computers solved a random circuit sampling (RCS) task thousands of years faster than a classical supercomputer [12–16]. However, their results rely on best-known classical algorithm estimates rather than proven lower bounds [17]. Later, researchers challenged these claims by modifying the algorithms to exploit the devices' noise and lack of error correction, effectively closing the quantum-classical runtime gap again. [18–22].

Besides the intellectual and medial back-and-forth, RCS results are unsatisfying from a scientific standpoint due to their inherent difficulty in rigorous proof or refutation. Verification relies on cross-entropy benchmarks that quickly become computationally infeasible for relevant problem sizes and noise obscures the boundary between true classical hardness and simulable artifacts. Consequently, the case of achievable advantage remains empirically contested and theoretically unresolved, which undermines the confidence in said claims and underscores the need for a more robust way to demonstrate quantum-advantage.

Ideally, a quantum-advantage protocol must be (i) feasible on near-term devices, (ii) classically easy to verify and (iii) classically hard to solve. Recently, a new type of quantum circuit called a peaked random circuit (PRC) was proposed that promises to fulfill all three requirements [1]. The idea is to generate a highly complex, ideally random circuit that embeds a single output state with an overwhelming probability weight within its otherwise uniform output distribution. While a quantum computer can easily identify this peaked output bitstring by running the circuit a few times, it is hoped that finding the peaked string via classical simulation is computationally infeasible.

As of writing, four methods have been proposed for generating PRCs using variational optimization and structural obfuscation techniques [1–3, 23]. Two of these works provide early experimental evidence that simulating PRCs may indeed be computationally hard and the latter reports a successful quantum-advantage experiment demonstrating the protocol's feasibility [2, 3]. While these preprint results are encouraging, the research community must now build on them to achieve a thorough understanding of PRCs. Since the entire protocol rests on their non-simulability, it is crucial to rigorously establish that no efficient simulator or shortcut exists. Recent advances in classical simulation have refined tensornetwork methods, enabling the fast and memory-efficient simulation of complex quantum circuits [24, 25]. Applying these techniques to challenge PRC non-simulability is a logical next step. I contribute to the literature by laying the groundwork for this endeavor.

The paper is structured as follows. Section 2 introduces peaked random circuits and summarizes four proposals for their generation. Section 3 discusses simulator choice, tensornetwork approaches, parallelization, circuit pre-optimization and techniques to find the peaked bitstring. Section 4 presents experimental evidence on the classical simulation of PRCs and on running PRCs on a quantum computer. Section 5 concludes.

2 Peaked Random Circuits

In their seminal paper on peaked random circuits (PRCs), Aaronson and Zhang define a peaked circuit as an n-qubit unitary circuit C that transforms an all-zero input string $|0\rangle^n$ into an output string $s \in \{0,1\}^n$ with a probability $\delta \in (0,1]$ upon measuring [1]. Formally

$$\max_{s\in\{0,1\}^n} |\langle s| C |0^n\rangle|^2 \geq \delta$$

holds with $\delta_s \equiv |\langle s| C |0^n\rangle|^2$ being the peaked bitstring's probability weight. To proof the quantum-advantage, the idea is to run the circuit a few times on a quantum computer. If it identifies the peaked bitstring, the quantum-advantage is verified. For this to work out, finding the peaked string must be computationally hard for a classical computer, necessitating a highly non-trivial, ideally random circuit that still executes on NISQ devices.

A key hurdle is to obtain a peaked circuit with the above properties. Simply sampling from randomly generated unitary circuits and then postselecting the ones that exhibit δ-peakedness for a given δ, e.g. $\delta = 0.2$, is likely not feasible for two reasons. First, it is unknown whether a classical computer can distinguish a peaked circuit from a random circuit in polynomial time – not to be confused with the task of finding the peaked bitstring. Second, it has been observed that the measurement outcomes of truly random quantum circuits anti-concentrate, that is, as the number of gates increases, the probability mass becomes increasingly dispersed among all possible output bitstrings [26]. Indeed, Aaronson and Zhang confirm that for random circuits whose depths grow logarithmically in the number of qubits n, the probability of being δ-peaked shrinks exponentially with n.

Since the publication of the seminal paper, researchers have worked to build and study peaked circuits for quantum-advantage experiments. The following subsections summarize this work.

2.1 Constructing PRCs from Variational Search

In their foundational study, after establishing that random sampling is likely not a viable option for obtaining peaked circuits, the authors propose an artificial construction method for generating such circuits [1].

First, a brick-wall circuit C_r with n qubits and τ_r layers of random gates is generated. When run from the basis state $|0\rangle^n$, the output state of C_r is random, i.e., the output bitstrings are uniformly distributed. Second, another circuit C_p with n qubits and τ_p layers is generated that converts the random output state of C_r back to the initial basis state $|0\rangle^n$. Intuitively, this can be seen as the inverse of C_r. Composing both circuits together yields a peaked circuit $C = C_p C_r$ that holds most probability weight in the outcome 0^n. Embedding X gates into the last layer allows to specify any desired peaked bitstring.

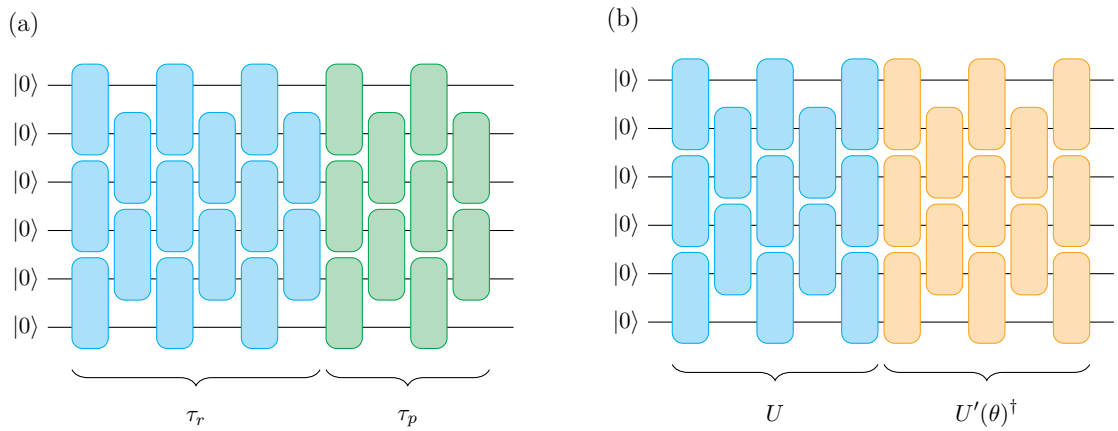


Figure 1: Illustration of PRCs built from variational search. (a) Approach of [1]. τ_r layers of random, two-qubit gates are generated, forming a 1D brick-wall structure. A smaller number of τ_p peaking layers of parameterized, two-qubit gates are added and tuned via numerical search until the final circuit is peaked on the desired output string. (b) Approach of [23]. First, a random, mirrored and trivially peaked 1D brick-wall circuit is created. The parameters of the second half of the circuit are then scrambled and tuned via numerical search until the final circuit is peaked again.

To generate the τ_p layers of C_p, the authors use stochastic gradient-descent optimization to maximize the target function

$$\max_\theta \delta_{0^n}(C(\theta)) = \max_\theta |\langle 0^n| C(\theta) |0^n\rangle|^2$$

with $C(\theta) = C_p(\theta)C_r$ being the final peaked circuit and $C_p(\theta)$ being τ_p parameterized quantum circuits with a variational parameter θ. In other words, $C_p(\theta)$ is constructed to maximize the probability weight of the final circuit's all-zero output string δ_{0^n}. Interestingly, the authors find that for the final circuit to have substantial peak weight, $\tau_p \ll \tau_r$ suffices, i.e., only a few peaking layers are needed. This method can be used not only to create one-dimensional (1D) circuits in which two-qubit gates strictly connect neighboring qubits, but also to construct higher-dimensional peaked circuits. Figure 1a illustrates the PRC design.

Because the optimization step introduces bias, or structure, the resulting circuit is not perfectly random. Although the term "quasi-random peaked circuit" would be technically more accurate, I use the term "peaked random circuit" for clarity and to convey the assumption that the circuit is still random enough to be computationally hard for a classical computer.

The foundational paper studies PRCs mainly numerically and conceptually and leaves two essential problems unresolved. First, the optimization procedure to calculate the second half of the circuit, C_p, is done in one step and doesn't scale in the number of qubits and gates. For PRCs to be practical, generating sufficiently large ones must be classically feasible. Second, it remains open whether the PRCs constructed this way are actually non-trivial and hard so solve classically. It could be that the formula just creates trivially inverted circuits $C_p' = C_r$ that naturally peak on the input string.

2.2 Constructing PRCs from Postselection and Variational Search

In a follow-up paper, Yuxuan Zhang turns several of the numerically motivated conjectures of the earlier work [1] into fully rigorous complexity-theoretic statements and studies an explicit construction of PRCs [23].

Instead of the previous, random-plus-peaking-layer architecture, the author explores a random-plus-postselection approach. First, a unitary k-design circuit U is generated at random. Second, additional random circuits U' of equal architecture are generated until some $U'^{\dagger}$ is found that yields a circuit $C = U'^{\dagger}U$ being δ-peaked on a given bitstring. Here, the postselection enforces the peak rather than the variational training layers. Unlike the earlier approach, the resulting PRCs have provably high complexity: the gates appear random locally, while globally, the circuits encode a chosen bitstring. In terms of complexity theory, the author establishes that precisely estimating a single peaked output weight for such circuits is #P-hard in the worst-case scenario which can be reduced to average-case #P-hardness with the polynomial method.

While this finding answers one of the open questions and suggests that PRCs are actually classically hard to solve, the approach suffers from the same problems as those discussed in the earlier work: sampling from a random distribution is classically infeasible. In fact, the success probability of the postselection decreases exponentially with the number of qubits and the question remains whether one can even efficiently distinguish between peaked and non-peaked circuits in the first place. Ideally, a PRC construction should be easy to compute while retaining the fully random nature and hardness properties.

To address this, the author proposes a design leaning again on variational optimization. Now, a variational circuit $U'(\theta)^{\dagger}$ is optimized to peak $C(\theta) = U'(\theta)^{\dagger}U$ on a target bitstring $x_\star$ by running gradient ascent on $\log p(x_\star)$ until δ-peakedness is archived. Unlike the method from the previous paper, which appends τ_p non-random peaking layers after τ_r random layers to enforce the peak, this method starts optimization on a fully random U' to ensure local randomness everywhere except the encoded peak. Figure 1b illustrates this approach. Comparing both circuits, the author indeed finds that PRCs created this way exhibit similar properties than truly random ones created from postselection.

Still, the optimization remains a computational bottleneck as it is done in one step and doesn't scale with the circuit size. To generate practically useful, large PRCs, the author proposes to combine multiple smaller, easily computable PRCs into a single, large circuit. This circuit-stitching idea follows from the observation that such a circuit is still peaked, regardless of whether the stitching is done in the horizontal or vertical direction, or both.

Ideally, in the setting of quantum-advantage proving, the verifier can infer the resulting peaked bitstring and its approximate probability simply from the layout of the individual PRC segments. The challenger on the other hand still faces a highly non-trivial PRC with the desired classical hardness. Practically, the concern is that the peaked string can be spoofed by solving each of the sub-circuits individually. This, however, requires knowledge about the exact location of the stitches. It could be, for example, that the marginals at the sub-circuit boundaries deviate from ones found elsewhere in the circuit, which can be identified by simple pattern-matching. However, the author suggests to destroy these patterns using rewrite protocols, essentially obfuscating the sub-circuits within the brick-wall structure.

The analytically focused study concludes with three open questions. First, are larger circuits generated from the variational search still sufficiently random to be computationally hard, or

is there a critical size beyond which the variational half converges towards being the inverse of the random half? Second, does the circuit-stitching method with obfuscation yield the desired outcomes in practice? Third, can the classical hardness of this construction be proven rigorously?

2.3 Constructing PRCs from Gate-Obfuscation

A practically oriented follow-up paper aims to solve a key issue of the previous two approaches: the circuit size is limited by a numerical optimization step that doesn't scale classically [2]. The author Oleg Udalov proposes a PRC design targeting inherent scalability. Rather than optimizing the entire second half of a mirror circuit at once to generate δ-peakedness, the PRC is constructed from special two-qubit operators which act essentially as one-way functions to embed and hide X gates and consequently any desired peaked string into the circuit. Each of the circuits building blocks B_i is a unitary, two-qubit operator having a

$$(\mathrm{U}_3 \otimes \mathrm{U}_3)\ \mathrm{CZ}\ (\mathrm{U}_3 \otimes \mathrm{U}_3)\ \mathrm{CZ}\ (\mathrm{U}_3 \otimes \mathrm{U}_3)$$

structure with a parameter set S_i consisting of 18 rotational angles from the six $\mathrm{U}_3(\theta, \phi, \lambda)$ gates. Each block forms a 4×4 matrix M_i. Crucially, for every block $B(S_1)$ with matrix M_1, there is at least one other block $B(S_2)$ with matrix M_2 so that $M_1 \approx M_2$. In other words, multiple, very different sets of angles can generate blocks with nearly the same matrix. In addition, the blocks allow to commute X gates, meaning that applying an X gate from the left side and moving it to the right side changes the block's parameters S_i while the circuit's overall structure is preserved.

The PRC construction consists of three steps, which are illustrated in Figure 2. First, an n-qubit random circuit is built from $2N$ B_i-operators forming a 1D brick-wall structure. The inverse of each block is created and on the block-level, the mirrored circuit

$$B_1 B_2 \cdots B_N B_N^\dagger \cdots B_2^\dagger B_1^\dagger = \prod_{i=1}^{2N} \tilde{B}_i$$

formed, fulfilling the symmetry relation $\tilde{B}_i = \tilde{B}_{2N+1-i}^\dagger$. The resulting circuit is equal to the identity operator and trivially peaked at the input string, e.g., $|0^n\rangle \to 0^n$.

Second, a target bitstring x_{hid} is embedded in the circuit by adding X gates to the input state and moving them to random locations between any of the gates. The commutability property of the $\tilde{B}_i$ operators ensures that the brick-wall structure of the circuit is preserved. At this stage, the X gates and therefore the embedded string can still be found easily by checking whether the symmetry relation between each pair of opposing blocks is violated.

Third, the circuit is modified so that the embedded X gates are computationally difficult to detect. Each block $\tilde{B}_i$ from the second half of the circuit ($N < i \leq 2N$), with parameters S_i and matrix M_i, is replaced by a modified block $\tilde{B}_i^{\mathrm{mod}}$ with S_i^{mod} and M_i^{mod}. The replacement block is found via numerical search until a given threshold block deviation $\tilde{\delta}_i$ is reached and $M_i^{\mathrm{mod}} \approx M_i$. Starting the optimization process with a random set of angles, and considering the operators' inherent property that different parameters can produce nearly identical matrices, ensures that the resulting parameter sets differ substantially. While this measure prevents the comparison of parameters between blocks, it is still possible to compare the blocks' matrices. To eliminate this issue, neighboring blocks are merged by combining their left- and right-adjacent U_3 gates. Now, to find the X gates, all blocks need to be compared in a single step, which gets

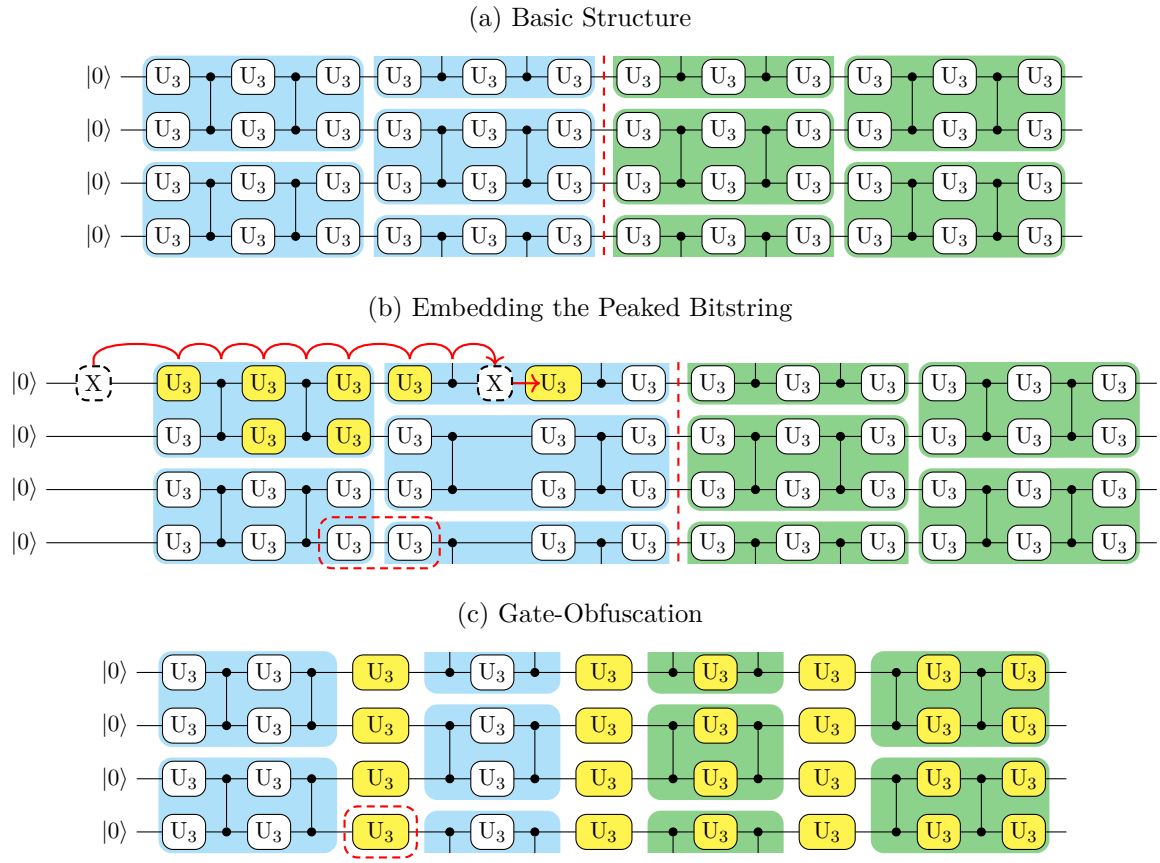


Figure 2: Illustration of the gate-obfuscation PRC design of [2]. (a) The initial circuit has a mirrored, 1D brick-wall architecture with blocks built from six U_3 and two CZ gates. (b) An X gate is inserted after the input register and commuted to a random location within the circuit. This sets the desired peaked bitstring to 0001. The X gate and the additional rotations from the commutation relations are fused with adjacent U_3 gates to preserve the circuit's structure. Gates with changed parameters are highlighted in yellow. (c) The blocks of the mirror half, highlighted in green, are replaced by an algebraically similar, yet parametrically different equivalent and each blocks' left and right adjacent U_3 gates are fused, resulting in the final, peaked circuit.

computationally more challenging with increasing circuit size.

A consequence of the matrix approximation step is that the probability weight of the peak bitstring decreases below one. The author finds that the circuit's peakedness correlates strongly with the average block deviation $\delta = \sum_{N+1}^{2N} \tilde{\delta}_i / N$, making this parameter adjustable. For a circuit of fixed size, peakedness can be increased by lowering the optimizers' target deviations $\tilde{\delta}_i$, which is a useful means of adjustment. Conversely, when scaling the size of the circuit, the required deviation to maintain a fixed peakedness decreases linearly with circuit depth and width. Optimizing a larger circuit for a lower target deviation drives the computational cost, representing a potential scalability boundary. The author investigates this behavior and finds that in practice, the target deviation required to reach a fixed peakedness decreases less sharply with the number of qubits than with the circuit depth. Consequently, the method scales best with very wide circuits. However, it remains unclear how deep the circuits can get before reaching a performance boundary and whether this approach can produce meaningfully large and sufficiently peaked circuits.

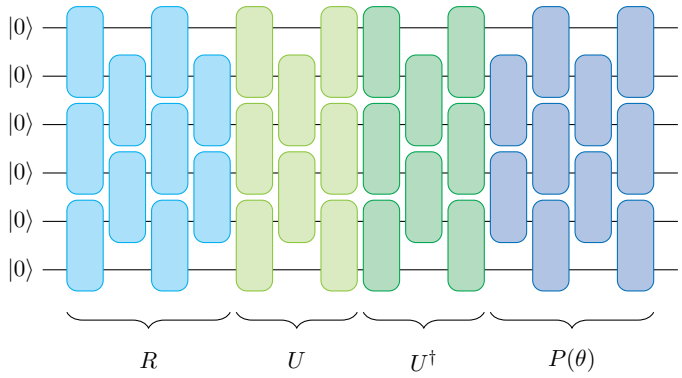


Figure 3: Illustration of the identity-obfuscation PRC design of [3]. The outer blocks $RP(\theta)$ correspond to a shallow peaked 1D brick-wall circuit generated via variational search as illustrated in Figure 1a. An identity block $UU^\dagger$ is added between the random and peaking halves to increase circuit depth. The structure is then scrambled to hide the identity and SWAP operations are added to obtain all-to-all connectivity.

Another unsolved question is whether the PRCs created this way are actually classically hard. The author does some experimentation with a generic MPS simulator, but more rigorous testing and ideally, complexity-theoretic proofing is necessary for validation.

2.4 Constructing PRCs from Identity-Obfuscation

A paper by researchers around the company Bluequbit proposes another technique for constructing scalable PRCs [3]. Unlike the previous method, which improves the optimization step, this method artificially stretches the circuit depth by embedding and obfuscating an identity block into a shallow peaked circuit.

The procedure consists of three steps. First, analogous to the foundational paper of [1], a random-plus-peaking layer circuit $C(\theta) = P(\theta)R$ is build and optimized until the output string reached a desired peak-probability δ. Unlike the other, gate-level PRC designs, this protocol relies on a tensornetwork representation of the circuit. The authors find that using tensornetwork techniques, such as contractions, allows this circuit to be constructed with an arbitrary number of qubits.

In the second step, the circuit depth is increased by inserting a random, mirrored sub-circuit $U^\dagger U$, resembling the identity, between the random and peaking halves.

Finally, the circuit is scrambled to obscure its structure and render classical simulation computationally infeasible. To achieve this, the authors leverage the tensornetwork representation of the circuit. Individual sub-circuits or gate-sets are represented as single tensors or "patches", which can then be swapped for an approximate substitute generated via numerical search to meet a target fidelity loss. This technique closely relates with the optimization introduced in [2], but is employed in different ways: an angle-sweeping step leaves the original gates in place and tunes the patches to increase fidelity while a masking step substitutes the patches' gate-sets. As a third scrambling measure, SWAP operations are implemented randomly throughout the circuit, thereby creating an all-to-all connectivity to make simulation more costly. Figure 3 illustrates this PRC design.

Notably, the paper is the first and, as of writing, the only one that demonstrates quantum advantage with a PRC. Using their identity-obfuscation design, the authors build several 56-

qubit PRCs and execute them on a Quantinuum H-2 trapped-ion device [27], then benchmark the resulting runtimes against a suite of classical simulators. By extrapolating the classical runtimes, they argue that an exponentially widening gap must exist between the two regimes. Because their PRC construction has not yet been rigorously analyzed or compared with other methods, the authors refer to their result as having archived a "heuristic" quantum advantage.

3 Classical Simulation of Peaked Random Circuits

Efficient simulation of quantum circuits and simulator choice are governed by the circuit's size, entanglement structure and gate-set. Since PRCs aim to demonstrate quantum advantage – that is, a large gap in the runtime between the classical and quantum regime – these factors are adjusted during circuit creation to maximize computational cost and simulation difficulty while leveraging the full capabilities of current quantum hardware.

The most immediate consideration for simulability is the number of qubits or circuit width. Since the state space expands to 2^n dimensions for n qubits, the memory needed to store the states' amplitudes grows exponentially. Current state-vector simulators support about 40 qubits before memory becomes a prohibitive constraint. Since PRCs designed for today's near-term devices typically have between 50 and 120 qubits, exact simulation methods are effectively ruled out.

Circuit depth compounds simulability by propagating and amplifying quantum correlations. While shallow circuits may still be efficiently simulable if entanglement is limited, deeper circuits typically generate volume-law entanglement that forces an exponential growth in simulation resources. A recent paper shows that shallow, peaked circuits can indeed be simulated efficiently [28] and protocol-grade PRCs should consequently have as many gates as reasonably possible. Limiting factors for practically achievable depth are noise and decoherence of near-term devices and the anti-concentration property of random circuits. For reference, the largest circuit used in Bluequbit's quantum advantage experiments had 56 qubits and around 4000 single- and 2000 two-qubit gates [3].

Apart from circuit size, the topology and depth of entanglement significantly impact simulation cost. For PRCs constructed with a one-dimensional brick-wall architecture using exclusively nearest-neighbor gates, the entanglement depth grows linearly with the circuit depth. The first few layers are therefore comparatively easy to simulate, but as full entanglement is achieved the cost rises sharply. By contrast, PRCs with all-to-all connectivity can rapidly entangle distant qubits, leading to a much steeper rise in entanglement and a correspondingly higher simulation cost.

Finally, the gate-set dictates simulation cost. Clifford gates are handled efficiently by the stabilizer formalism under the Gottesman–Knill theorem [29] whereas non-Clifford primitives contain non-Gaussian elements that are substantially harder to compute. PRCs are built from a universal set of U_3 and CZ gates, which extends the Clifford group. Because the U_3 rotation angles are chosen randomly and are almost never integer multiples of $\pi/2$, PRCs effectively never fall into the Clifford group. Consequently, stabilizer simulators cannot be used for PRC simulation.

Considering PRCs' relatively large size, high entanglement depth and challenging gate set, simulation requires the most performant, approximate techniques, which are discussed in the following.

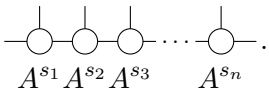
3.1 Tensornetwork Simulators

Beyond their popularity in quantum physics [30] and machine learning [31], tensornetworks (TNs) have emerged as a powerful tool for modeling complex quantum systems due to their ability to represent an exponentially large state space with reduced computational resources [24, 25]. Numerically, any state vector representing a pure, n-qubit quantum state

$$|\psi\rangle = \sum_{s_1,\ldots,s_n} c_{s_1 s_2 \cdots s_n} |s_1 s_2 \ldots s_n\rangle$$

with complex amplitudes $\{c_{s_1 s_2 \cdots s_n}\}$, each corresponding to the physical indices of the basis strings $s_1, s_2, ..., s_n \in \{0, 1\}$, can be expressed as a rank-n tensor $T^{s_1 s_2 \cdots s_n} = c_{s_1 s_2 \cdots s_n}$. This large, single tensor can be factorized into a product of lower-rank tensors $\{A\}$ with $\alpha_b = 1, ..., \chi_{\max}$ being adjacent tensors' bond indices. Often, one uses a chain of n rank-3 tensors, each corresponding to the site of a single qubit [32]:

$$\begin{aligned} T^{s_1 s_2 \ldots s_n} &= \sum_{\{\alpha_b\}} A^{s_1}_{\alpha_1} A^{s_2}_{\alpha_1,\alpha_2} A^{s_3}_{\alpha_2,\alpha_3} \cdots A^{s_n}_{\alpha_{n-1}} \\ &= \sum_{\{\alpha_b\}} \end{aligned}$$



The tensornetwork representation enables the efficient execution of numerous relevant operations such as applying gates and calculating observables, without the need to construct the full state vector of 2^n amplitudes. Instead of scaling exponentially with the number of qubits, the parameters and memory required to store them scale with the maximum bond dimension as $\mathcal{O}(n\chi^2_{\max})$. If the entanglement depth, i.e., the largest number of mutually entangled qubits, increases by one unit, maintaining an exact representation of the quantum state requires a doubling of $\chi_{\max}$. To avoid an exponential blow-up of parameters with entanglement, the maximum allowed bond dimension can be capped during tensornetwork construction by applying singular value decompositions (SVDs). Truncating small singular values reduces the memory footprint at the expense of loosing some fidelity. Popular quantum programming frameworks typically offer two types of tensornetwork-based simulators.

The first one is an exact TN formalism that does not incorporate truncation. Here, the ranks and connectivity of the tensors adjust according to the layout of the circuit's gates. Rather than forming a chain, the structure of the simulator object resembles a graph. Gates are represented as small tensors and applied by inserting them as additional nodes. Calculating, for example, an observable or an expectation value is achieved via a global contraction of the simulator object. The computational cost depends on the structure of the circuit and the quality of the contraction path. Due to the lack of truncation and memory blowup resulting from high entanglement, simulating PRCs with this approach is not feasible.

The second, common tensornetwork-based simulator is the matrix product state (MPS) formalism. Here, n rank-3 site-tensors are used and connected to form a chain. The output state of the circuit is built gate-by-gate while implementing a SVD truncation step during gate application. To apply two-qubit gates to non-adjacent site-tensors, SWAP operations must be used to reposition them next to each other. Modern MPS implementations do that automatically and efficiently whenever required. Users can fine-tune the memory-fidelity tradeoff by specifying the maximum allowed bond dimension, absolute and relative SVD cutoff values, and the SVD

algorithm. In practice, MPS simulators have proven highly useful in the approximate simulation of larger circuits as long as entanglement remains modest [33, 34].

Simulating highly entangled PRCs using the MPS formalism is fundamentally restricted by the amount of available memory. The true entanglement depth quickly exceeds the capacity of the maximum allowed bond dimension to accurately represent the quantum state, necessitating truncations and making the simulation highly approximate. Over time, compression errors compound to a point where the fine structural nuances encoding the peaked bitstring are effectively "washed out", rendering it impossible to recover. The authors of the identity-obfuscation design estimate the bond dimension and amount of memory needed to "break" one of their PRCs with 1000 two-qubit gates at $\chi_{\text{break}} \approx 10^6$, requiring 100 TB of RAM [3].

Clearly, basic TN and MPS simulators are inadequate for simulating PRCs. However, more sophisticated TN approaches may be better suited for this task. For instance, projected entangled pair states (PEPS) generalize the MPS formalism to higher spatial dimensions [30, 35]. Here, the underlying graph can form a high-connectivity topology, such as a square lattice or honeycomb. Tree tensornetworks (TTNS) arrange tensors in a tree-like structure, reducing complexity for certain circuits [36]. The multi-scale entanglement renormalization ansatz (MERA) uses a hierarchical structure that organizes entanglement layer by layer across scales, allowing for the systematic removal of short-range entanglement [37]. For the task of simulating PRCs and specifically ones with higher dimensionality, these advanced structural approaches are likely beneficial because they more accurately represent the circuits' connectivity and entanglement, which theoretically results in better performance and memory scaling.

Beyond a more faithful representation of the circuits, higher-dimensional TN architectures allow for sophisticated optimizations of the underlying contractions. The isoTNS approach [38] imposes an isometric restriction to the TN that accelerates contractions. Statistical inference techniques, such as belief propagation [39], can be used to replace or complement SVDs, allowing for context-aware truncations. Here, such a feature could be used to identify and preserve the structures of the circuit that make up the embedded bitstring.

Ultimately, none of the simulators discussed is ideally suited for PRC simulation. While this fact may support the notion of non-simulability, it might very well be the case that a simulator can be developed that achieves efficiency with this class of circuit and thereby undermines the intended quantum-advantage. Tensornetwork simulators present a strong case for this effort, and they can be tailored to handle the extreme density and entanglement characteristic of PRCs.

3.2 Finding the Peaked Bitstring

Identifying the peaked bitstring on a quantum computer involves executing the PRC and measuring the qubits in the computational basis. Repeatedly running the circuit produces a discrete probability distribution over all output states. If the quantum device has sufficient fidelity and the circuit's δ-peakedness is high enough, the hidden bitstring corresponds to the most frequently observed outcome. On classical hardware and using tensornetwork simulation, four attack strategies can be applied to find the bitstring.

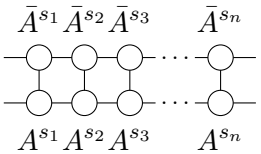
The first technique is non-deterministic sampling, which takes advantage of the fact that an n-qubit circuit in TN or MPS representation, i.e., all gates applied, encodes a probability distribution of the circuit's possible output states. Randomly drawing therefrom allows to construct a discrete distribution that reveals the peaked bitstring. A general description of the sampling

algorithm can be found in [40, 41]. To start with, the TN/MPS must be normalized so that the probability of observing a basis state with indices $s_1, s_2, ..., s_n \in \{0, 1\}$ is equal to the norm of the tensor squared:

$$p(s_1, s_2, s_3, \ldots, s_n) = |T^{s_1 s_2 s_3 \cdots s_n}|^2.$$

The tensor can be normalized efficiently by calculating the sum of this expression over the indices and dividing the TN/MPS by the square root of the resulting value:

$$\sum_{\{s\}} |T^{s_1 s_2 \cdots s_n}|^2 =$$



$$= 1.$$

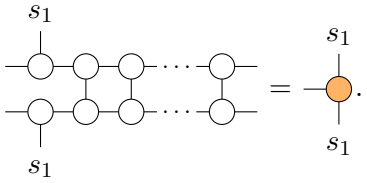
Because the entanglement structure and connectivity of the circuit induce correlations among its output bits, statistically, the event of a single bit being zero or one is dependent on the events of the other bits being zero or one. Accordingly, the chain rule dictates that the joint probability of observing a particular output string factorizes into a product of the individual bits' conditional probabilities:

$$p(s_1, s_2, \ldots, s_n) = p(s_1)p(s_2|s_1) \cdots p(s_n|s_1 s_2 \cdots s_{n-1}).$$

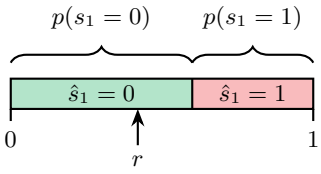
To calculate one sample, the procedure gradually builds the terms of this expression, leveraging TN/MPS summation techniques. The first and leftmost probability $p(s_1)$ is calculated by contracting the right-adjacent tensors over the remaining indices:

$$p(s_1) = \sum_{s_2, s_3, \ldots, s_n} p(s_1, s_2, \ldots, s_n) = \sum_{s_2, s_3, \ldots, s_n} T^{s_1 s_2 \cdots s_n} \bar{T}^{s_1 s_2 \ldots s_n}$$

$$=$$

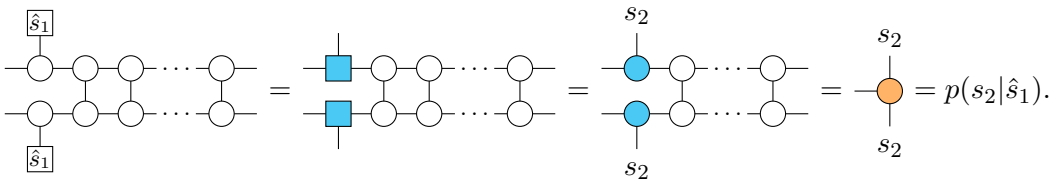


Alternatively, one can calculate the reduced density matrix (RDM), which is trivial in a right-canonical TN/MPS: $\rho^{s_1} = A_1 A_1^\dagger$. The trace values of the RDM directly correspond to $p(s_1 = 0)$ and $p(s_1 = 1)$. These probabilities are then used to set the first output bit $\hat{s}_1$ of the sample: a continuous number $r \in [0, 1]$ is chosen randomly and the bit determined to be $\hat{s}_1 = 0$ if $r < p(s_1 = 0)$ and $\hat{s}_1 = 1$ if $r \geq p(s_1 = 0)$. The following graphic illustrates the weighted random sampling:



After setting the sample's first bit, the MPS is conditioned on the value of $\hat{s}_1$. This is done by contracting the open index of the leftmost tensor with a standard basis vector $e_{\hat{s}_1}$ that contains zeros except for the value of $\hat{s}_1$ at the corresponding bit's entry. Next, the contracted tensor is normalized by division through $1/\sqrt{p(\hat{s}_1)}$. Graphically, the conditioning and calculation of the

next conditional probability can be depicted as:



The procedure then recursively calculates the rest of the bits $\hat{s}_2, \hat{s}_3, \ldots, \hat{s}_n$, using the conditional probabilities $p(s_2|\hat{s}_1), p(s_3|\hat{s}_1\hat{s}_2), \ldots, p(s_n|\hat{s}_1\hat{s}_2\cdots\hat{s}_{n-1})$. Once complete, a single sample is obtained. To build the discrete probability distribution, the procedure is repeated multiple times.

Unlike quantum-sampling, which requires executing the entire circuit from the all-zero input register, each iteration of the classical algorithm reuses the TN/MPS representation of the circuit's output state. From this starting point, it scales linearly with the number of shots or samples k as $\mathcal{O}(kn\chi^3)$. The runtime can be reduced to $\mathcal{O}(kn\chi^2)$ by bringing the input TN/MPS into right-canonical form, which itself scales as $\mathcal{O}(\chi^3)$. By right-gauging the tensornetwork, trivial RDM calculations can be used to obtain the leftmost tensors' conditional probabilities. This pre-optimization step becomes practically advantageous when a large number of samples are generated.

One such instance is precisely to draw the peaked bitstring from a PRC's exponentially large output space. With a modest probability weight δ, each sample is a highly noisy estimator of this rare event. Assuming that the non-peaking outcomes are uniformly distributed, probability theory suggests that the noise decreases with the number of samples as $\sqrt{k}$. To reliably identify the peaked string, noise has to be reduced to below δ which is archived by scaling the number of samples at least as the inverse square of the peak probability: $k = 1/\delta^2$. In practice, given that PRCs have values of δ in the range of 10^{-2} to 10^{-3}, this translates to 10^4 to 10^6 samples that need to be calculated.

The second technique recovers the peaked bitstring by evaluating single-qubit Pauli-Z expectation values. For example, to find the first bit of a four-qubit circuit, the Z_1 expectation $\langle \mathrm{Z}_1 \rangle = \langle\psi| \left(\mathrm{Z} \otimes \mathrm{I} \otimes \mathrm{I} \otimes \mathrm{I}\right) |\psi\rangle$ is calculated and used to obtain the marginal probabilities:

$$p(s_1 = 0) = \frac{1 + \langle \mathrm{Z}_1 \rangle}{2}, \qquad p(s_1 = 1) = 1 - p(s_1 = 0) = \frac{1 - \langle \mathrm{Z}_1 \rangle}{2}.$$

The first bit is determined to be $\hat{s}_1 = 0$ if $p(s_1 = 0) > p(s_1 = 1)$ and $\hat{s}_1 = 1$ otherwise. Repeating this procedure for the remaining $\langle \mathrm{Z}_2 \rangle, \ldots, \langle \mathrm{Z}_n \rangle$ yields the peaked bitstring. A straightforward implementation efficiently calculates the expectation values via contractions. Independently contracting for each site of the TN/MPS has a runtime of $\mathcal{O}(n\chi^3)$, which can be reduced to $\mathcal{O}(n\chi^2)$ by reusing left and right environment tensors along the chain.

Although this method can quickly produce an output in a single pass, it does not reliably return the true hidden bitstring. The peak is a global property of the entire PRC, whereas the algorithm constructs the output by piecing together independent marginal probabilities for each bit. Treating the bits as uncorrelated may cause the resulting string to be locally most likely yet distinct from the true peaked outcome. Nevertheless, the technique is valuable for experimental work and reliably locates the peak in PRCs that have been deliberately generated for testing and evaluation.

The third technique recovers the peaked bitstring by evaluating single-site RDMs. For a quantum state represented as a TN/MPS, the RDM at site n is denoted by ρ^{s_n} and is 2×2 in dimension. It's trace contains the probabilities of the corresponding qubit being zero or one:

$$p(s_n = 0) = \rho^{s_n}_{11}, \qquad p(s_n = 1) = 1 - p(s_n = 0) = \rho^{s_n}_{22}.$$

Again, bit n is determined to be $\hat{s}_n = 0$ if $p(s_n = 0) > p(s_n = 1)$ and $\hat{s}_n = 1$ otherwise. A naive implementation efficiently builds the RDMs by contracting the normalized TN/MPS with its conjugate and tracing out all but one site, which again scales as $\mathcal{O}(n\chi^3)$ or $\mathcal{O}(n\chi^2)$ when the TN/MPS is preprocessed and computations are reused. The probabilities obtained from the RDMs are equivalent to those obtained from the Pauli-Z expectations and the result might again differ from the true peaked bitstring.

The fourth technique is deterministic sampling, a modification of the non-deterministic sampling algorithm described earlier. Instead of treating the conditional probabilities as stochastic weights to randomly select the outcome bits, this method skips the oracle and applies the probabilities directly. Bit n is determined to be $\hat{s}_n = 0$ if $p(s_n = 0|\hat{s}_1 \cdots \hat{s}_{n-1}) > p(s_n = 1|\hat{s}_1 \cdots \hat{s}_{n-1})$ and $\hat{s}_n = 1$ otherwise. The computational cost is identical to that of generating a single sample. Unlike the previous two approaches, deterministic sampling incorporates conditional, joint information rather than merely single-marginal statistics, making it significantly more likely to recover the true peaked bitstring. Still, there is no guarantee that it will do so. Success depends on the circuits internal correlations and design choices made. In fact, the gate-obfuscation PRC design allows to embed several hidden strings. If the quantum-advantage protocol is altered to identify the second or some x-th peak, all techniques except random sampling become ineffective.

3.3 Parallelization

Parallelization can substantially reduce the runtime of complex quantum simulations by delegating parts of the simulator object and linear algebra operations to multiple units of work.

Tensornetworks are naturally structured to support distributed computations because they represent a global quantum state as a set of tensors that are mostly connected locally. The inherent locality allows independent or weakly coupled contractions to be performed in parallel over distinct sub-regions.

Optimizing the contractions can further enhance the performance. Index slicing, for example, is a technique that streamlines large, complex contractions by splitting them into a set of identically structured, independent sub-tasks that can be executed in an embarrassingly parallel way. Researchers report that simulating certain random circuits with slicing is several orders of magnitude faster than without [42]. Next to more conventional optimizations of the contraction path [43, 44], deferring contractions [45] and incorporating heuristics to reduce redundancies [46] also yield significant, practical speedups. Highly optimized and parallelized TN simulators are now widely used for large-scale quantum simulations in high-performance computing (HPC) environments [44, 47].

The ability of tensornetworks to leverage parallelization also motivates the use of GPUs, which are designed specifically for that purpose. Indeed, GPUs consistently outperform CPUs in TN-based simulations. This advantage stems from the fact that TN simulators' core operations –

high-dimensional tensor contractions and Einstein summations – can be reformulated as matrix-matrix multiplications, which thousands of specialized GPU cores excel at calculating efficiently. Studies that assess the performance impact of using GPUs for TN simulations report orders-of-magnitude speedups over optimized CPU-based simulators [48, 49].

While parallelization and the use of GPUs can drastically reduce the time to build the simulator object and apply all gates, the same techniques can also be used to speed up the task of finding the peaked bitstring via non-deterministic sampling. Although the runtime of the sampling algorithm only scales linearly with the number of samples, the time it takes to generate the hundreds of thousands of samples required for reliable identification of the peaked bitstring is significant in practice. A straightforward speedup is therefore to run the algorithm in parallel across multiple units of work, ideally GPUs. In practice, this requires either distributing the simulator object in the first place or making the simulator object portable.

3.4 Circuit Pre-Optimization

The computational complexity of TN and MPS simulations is highly sensitive to the contraction path and order in which operators are applied. By default, gate application strictly follows the sequence defined during circuit creation. State-preserving reordering operations – such as grouping gates into local interactions – can theoretically minimize the growth of the bond dimensions and delay the introduction of heavy entanglement. In practice, the potential for reordering is limited by the connectivity and entanglement structure of the circuits. PRCs with a 1D brick-wall design and U_3-CZ gate-set have their gate types arranged in an dense, alternating pattern by default. To some extent, this allows for the regrouping by single-qubit rotational gates and two-qubit entanglement gates. For example, given an application sequence of a four-qubit PRC generated via the gate-obfuscation method, regrouping yields:

1: U_3 on q_0, q_1		
2: $\mathrm{CZ}(q_0, q_1)$		1: U_3 on q_0, q_1, q_2, q_3
3: U_3 on q_0, q_1		2: $\mathrm{CZ}(q_0, q_1)$
4: $\mathrm{CZ}(q_0, q_1)$	$\Rightarrow$	3: $\mathrm{CZ}(q_2, q_3)$
5: U_3 on q_2, q_3		4: U_3 on q_0, q_1, q_2, q_3
6: $\mathrm{CZ}(q_2, q_3)$		5: $\mathrm{CZ}(q_0, q_1)$
7: U_3 on q_2, q_3		6: $\mathrm{CZ}(q_2, q_3)$
8: $\mathrm{CZ}(q_2, q_3)$		

Through experimentation, I find that while the computational cost of transpiling the circuit in this way is negligible, MPS simulation runtime improves linearly by around two percent, likely due to delayed bond dimension growth. Although this result is not relevant from a complexity-theoretic perspective, it favors parallelized gate application: an MPS simulator could be developed with distributed site-tensors that allows an entire batch of single-qubit U3 gates to be applied in parallel rather than sequentially, which might result in a significant improvement in runtime. Modern TN simulator implementations on the other hand incorporate contraction path optimization and parallelization, making the reordering step unnecessary.

While basic reordering focuses on preserving the states, more sophisticated pre-optimization, called level I optimization in the literature [50], involves simplifying the circuit. Pattern matching can be used to find common substructures and replace them with optimized equivalents.

Adjacent gates can be fused into single unitaries or complex unitary operations decomposed into smaller, optimized components. In theory, by minimizing the total number of operations and truncations, runtime and simulation cost can be drastically reduced.

On the one hand, since PRCs are designed to be highly dense and incompressible, it is unlikely that the circuits can be meaningfully simplified. Of the four current PRC designs, only the gate-obfuscation approach allows for further simplification via gate-fusing. While the left- and right-adjacent U3-gates of neighboring building blocks were merged during circuit creation, each block's inner CZ $(U_3 \otimes U_3)$ CZ sequence can still be fused into a two-qubit unitary (see Figure 2). While the reduction in circuit depth is evident, further experimentation is required to see whether it translates to a reduction in simulation runtime.

On the other hand, even tough PRCs are designed to emulate the complexity of truly random circuits, they can still, in principle, contain structural patterns or redundancies introduced by the heuristics that generate them. This suggests that the same algorithms, such as those used in the variational optimization step, can be used to simplify them systematically. In the extreme case, an attack on the circuit structure could yield the peaked string directly, without the need of simulation. Of the four papers on PRCs, only the authors of the identity-obfuscation approach address the feasibility of such attacks on their circuit design and provide evidence that they are unlikely to succeed [3]. Nevertheless, developing a thorough understanding of the circuits' structures is crucial, whether for constructing structural attacks or accelerating classical simulation.

4 Experiments

After establishing a theoretical understanding of the structure and simulability of peaked random circuits, the following sections present experimental evidence on simulating PRCs using various techniques.

4.1 Simulating Peaked Random Circuits

To begin with, a diverse set of circuits with varying characteristics must be generated for testing and evaluation. Among the four PRC designs, only the gate-obfuscation and identity-obfuscation method allow to generate sufficiently large circuits. However, the authors of the identity-obfuscation PRCs do not provide a complete, reproducible description or source code for building them. Consequently, I adopted the gate-obfuscation method, which is fully disclosed on GitHub.

The gate-obfuscation protocol requires four parameters: the number of qubits, the circuit depth, the target bitstring, and the parameter $\tilde{\delta}$ controlling block-replacement fidelity. I generated 120 circuits, covering qubit counts of 6, 10, 20, 30, 40, and 50, each at depths ranging from 1 to 20. Following the author's analysis of the maximum allowed block deviation [2], I set the parameter to $\tilde{\delta} = 0.01$, which should ensure sufficient peakedness for all circuits generated. Specifically, the probability of the peaked outcome should be ten times higher than the probability of the second most common outcome. The hidden string is

$$0101111000111101001100111101000011101001100000111 01,$$

truncated to the first n bits for an n-qubit circuit. A crucial step in the protocol is to render structural attacks impossible by moving the X gates to random locations within the circuit

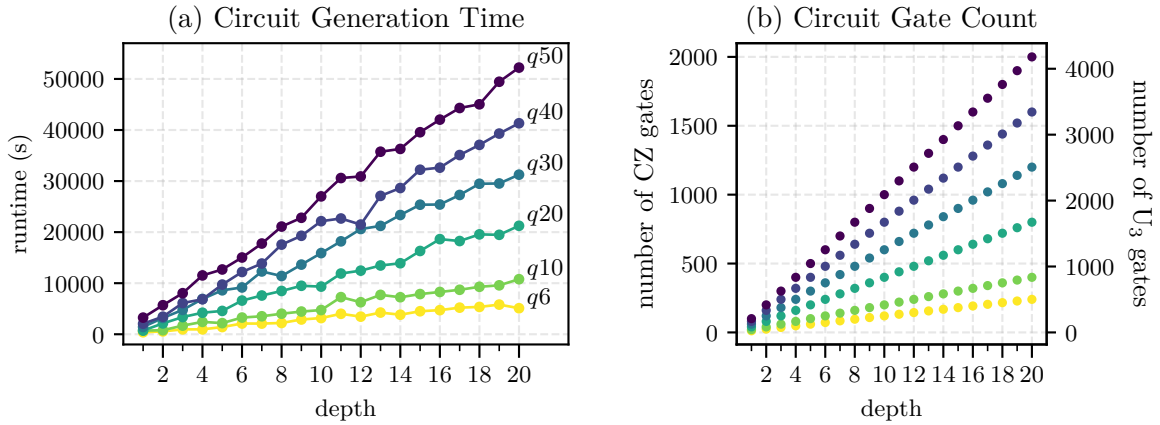


Figure 4: Characteristics of the gate-obfuscation PRCs used for testing. (a) Time it took to generate the circuits by depth and number of qubits. (b) Gate-composition of the circuits. The left y-axis shows the exact number of two-qubit CZ gates. The right y-axis shows the number of single-qubit U_3 gates. Since the two metrics do not correlate perfectly, this number is approximate and deviates by a few gates.

structure via commutation relations. However, this adds a non-deterministic component that complicates comparisons across circuits. For this reason, I generated them without commutations, which resulted in the X gates being absorbed into the first layer of U_3 gates.

Figure 4a plots the circuit generation time. For a fixed block-replacement fidelity, it scales strictly linearly, demonstrating that the gate-obfuscation protocol is inherently scalable and allows to produce arbitrarily large circuits. Figure 4b shows the gate-composition of the circuits. The largest circuit has 2000 entanglement gates and 4000 arbitrary rotation gates, matching the size of the circuit behind the Bluequbit quantum-advantage demonstration [3].

For simulating quantum circuits, several mature programming frameworks exist, with the most popular being Qiskit [51], Cirq [52], and PennyLane [53]. These frameworks offer developers and researchers a robust set of tools for areas such as circuit synthesis, error mitigation and variational algorithm design. They also implement generic state-vector, TN and MPS simulators, offering a convenient starting point for evaluating the simulability of PRCs. However, the rigorous simulation of PRCs requires more sophisticated and capable approaches, necessitating a platform that allows for simulator optimization and development. I therefore used NVIDIA's CUDA-Q [54] and cuQuantum SDK [55]. CUDA-Q supports some advanced TN simulators natively, while cuQuantum allows for low-level access to the underlying algebraic operations and hardware resources. Both frameworks support MPI-based parallelism and are engineered for NVIDIA GPUs, promising exceptional simulation performance.

I ran the benchmarks on the quantum learning machine (QLM) by Eviden, seated at the Leibniz Supercomputing Center of the Bavarian Academy of Sciences (LRZ). The system has eight terabytes of RAM, eight Intel 8450H CPUs with a total of 160 cores and eight NVIDIA A30 GPUs with 24 gigabytes of VRAM each.

I started by benchmarking the non-approximate TN simulator that ships with CUDA-Q. The implementation automatically determines the cost-optimal global contraction path. When tensors become too large to fit the GPU memory, the simulator slices them, effectively trading memory for additional runtime. In theory, the simulator should comfortably handle low-entanglement

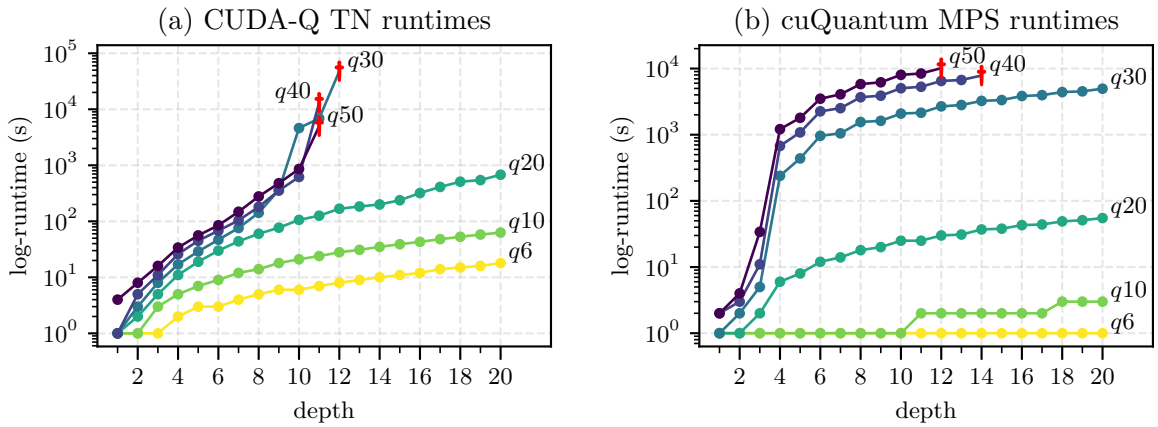


Figure 5: Comparing PRC simulation runtimes. The circuits correspond to those introduced in Figure 4. (a) Seconds until bitstring recovery using CUDA-Q's exact tensornetwork simulator. (b) Seconds until bitstring recovery using an MPS simulator implemented with the cuQuantum SDK. Red daggers indicate last successful simulation.

PRCs, but high-entanglement PRCs will inevitably encounter exponential runtime growth. I recovered the peaked bitstring by calculating the expectation values of Pauli-Z operators.

I then benchmarked an experimental MPS simulator built with the cuQuantum SDK. Unlike standard MPS implementations, I incorporated local contraction-path optimization and slicing at every contract-decompose step. Initially developed to add multi-GPU support to an MPS simulator, this feature can theoretically extend the headroom for contractions in highly memory constrained environments, such as GPUs. While the simulator can handle bond dimensions of 4096 or higher for some circuits, I constrained the maximum bond dimension to 2048 to fit the simulator object comfortably within the A30's device memory and to maintain stability at all times. Truncations used the GESVD routine with an absolute cutoff of 10^{-20} and a relative cutoff of 10^{-10}. The target bitstring was recovered from RDMs, calculated after bringing the MPS into right-canonical form. Due to its approximate nature, the simulator should enable faster and more memory-efficient simulations and allow to simulate PRCs too large for exact formalisms.

Although both simulators support execution across multiple GPUs, I chose to run the benchmarks on a single A30 and leave comprehensive multi-GPU benchmarks, including testing the distributed MPS implementation, for future work.

4.2 Classical Results

Figure 5a plots the runtime of PRCs using the exact TN simulator. For circuits containing 6, 10, and 20 qubits, the simulator handled depths up to 20. In contrast, PRCs with 30, 40, and 50 qubits entered exponential runtime growth after a depth of 10. This behavior stems from the gradual increase in entanglement with depth, which expands the state space to a point where the tensors required to hold it exceed the capacity of the available memory. At this point, the simulator slices the tensors, which dramatically slows down the computation and makes the simulation of deeper circuits intractable.

Figure 5b plots the runtime of PRCs using the approximate MPS simulator. Compared to the exact TN simulator, the 6-, 10-, and 20-qubit circuits were simulated ten times faster, which

is a significant speedup. Interestingly, at shallow depths between two and four, the 30-, 40-, and 50-qubit circuits exhibit a phase of exponential runtime growth and roughly one order of magnitude slower bitstring recovery. This bottleneck is caused by the gradual saturation of the maximum bond dimension as entanglement grows. Not only the memory requirement scales exponentially, but also the time it takes to expand the site-tensors to their maximum size. In this shallow domain, the exact simulator outperforms the approximate one: it treats entangling gates like any other operator, adding them to the global contraction that scales linear in runtime as long as memory is sufficient.

Beyond the shallow regime is where the approximate simulator shows its strengths. Once entanglement saturates – controlled by truncation and a capped bond dimension – the MPS representation and simulator object remain at a fixed size. Additional layers of entanglement merely update the existing site tensors, resulting in linear-time scaling. This capability allows the MPS to successfully simulate a 30-qubit, depth-20 PRC, a task that the exact simulator could not accomplish.

Beyond depths of 14 and 12, the bitstrings of the 40- and 50-qubit circuits could not be recovered due to overwhelming compression error. Increasing the maximum allowed bond dimension to values above 2048 would yield additional recoveries, but this strategy is ultimately limited by memory constraints and only works up to a certain circuit size. Beyond which, advanced optimizations of the simulator become mandatory. The experiments demonstrate that the approximate MPS formalism implemented with the cuQuantum SDK provides a robust foundation for further optimization and development.

4.3 Running Peaked Random Circuits on a Quantum Computer

Lastly, it is worth asking whether the PRCs that can be simulated, or rather those that cannot, are actually practical for demonstrating a quantum-advantage. For what circuit size and number of samples will current near-term devices produce an output distribution reliably peaked at the hidden bitstring?

To shed light on this question, I carried out quantum-advantage experiments on the new IQM Radiance 54-qubit superconducting device [56] seated at the LRZ compute-cube. Its qubits are arranged in a highly connected square-lattice and support arbitrary X and Y rotations as the native single-qubit gate and CZ as the native two-qubit gate, making the device ideal to run 1D brick-wall PRCs with a U_3-CZ gate-set. Apart from the Bluequbit paper [3], these experiments mark the second time PRCs have been used to demonstrate a quantum-advantage.

Using the Munich Quantum Software Stack (MQSS) [57] as an interface to schedule jobs and retrieve results, I successively executed the entire batch of 120 test circuits. Beginning with 10^4 shots per run, I logged the result only when the peaked outcome matched the true hidden bitstring. Otherwise, I reran the circuit, multiplying the sample count by ten each time, up to a maximum of 10^8 samples.

Figure 6a plots successful runs by circuit size and number of shots. Of the 120 PRCs, 37 returned an output distribution that peaked at the true hidden bitstring. Producing 10^7 and 10^8 samples did not yield additional identifications. Of the circuits that ran successfully, the two largest ones were the the 30-qubit, depth-8 circuit and the 40-qubit, depth-6 circuit. Both have 500 CZ gates and 1000 U_3 gates and are classically simulable. Larger, potentially non-simulable circuits produced output distributions resembling white noise.

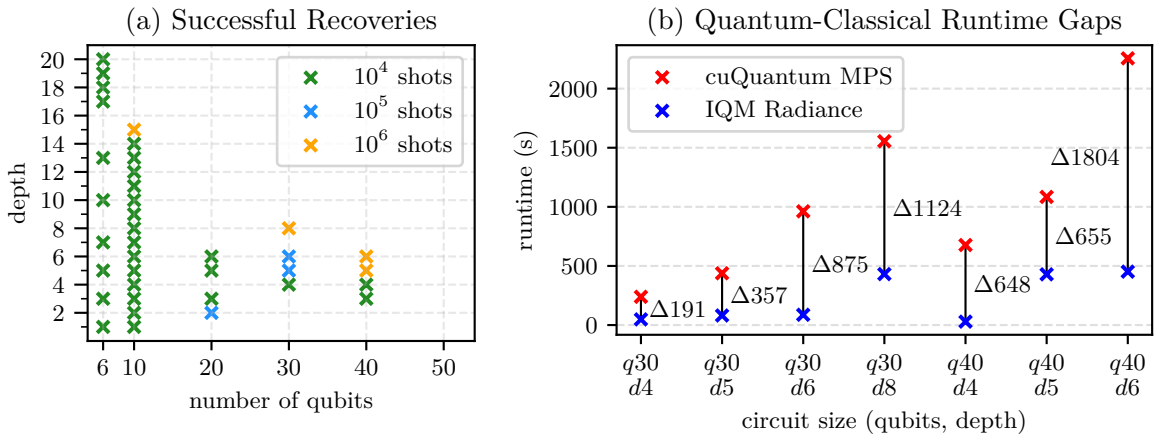


Figure 6: Quantum-advantage experiments on the 54-qubit IQM Radiance system. (a) A total of 120 gate-obfuscation PRCs with 6, 10, 20, 30, 40 and 50 qubits and depths ranging from 1 to 20 were run. Colored markers indicate successful identifications of the peaked bitstring at the specified circuit width, depth and number of shots. (b) Quantum-classical runtime gaps above 60 seconds. Markers in blue show runtimes on the quantum computer, markers in red the time it took to simulate the same circuit using an MPS simulator (corresponding to Figure 5b).

Using the MPS benchmark results, I calculated the quantum-classical runtime gaps for each circuit. While most were negligible in magnitude, the seven largest circuits exhibited a speedup: their bitstrings were recovered significantly faster by the quantum device than via classical simulation. Figure 6b shows all gaps exceeding 60 seconds. The key observation is not so much that the 40-qubit, depth-6 circuit demonstrated a 30 minute quantum-advantage, but rather that the speedup increases with circuit size – precisely the behavior expected from a functioning quantum-advantage protocol.

5 Future Directions

In this work, I introduced peaked random circuits and discussed techniques for simulating them efficiently. The classical experiments confirm that this class of circuits is difficult to simulate, yet they demonstrate that approximate tensornetwork simulators are well-suited for this challenge due to their efficiency and speed. The quantum experiments reveal a widening runtime gap between the two compute regimes and show that PRCs are viable for use as a quantum-advantage protocol.

To challenge non-simulability, the next step is to develop a tensornetwork-based simulator specifically for breaking PRCs. Ideally it would: (i) adapt a higher-dimensional tensornetwork topology that more accurately reflects the entanglement structure of the PRC, e.g., PEPS. (ii) Implement an advanced contraction scheme that leverages parallelization. (iii) Use inference techniques for informed truncation that preserves the embedded bitstring.

Lastly it is worth mentioning that besides PRCs, other NISQ-capable quantum-advantage protocols have been proposed. Hidden Code Sampling (HCS) uses quantum error-correction codes to produce output distributions with conditional peaks, enabling efficient classical verification without a full simulation [58]. What technique will eventually constitute a sound foundation for a near-term quantum-advantage proof is yet to be determined.

References

[1] S. Aaronson and Y. Zhang, "On verifiable quantum advantage with peaked circuit sampling," *arXiv preprint arXiv:2404.14493*, 2024. DOI: 10.48550/arXiv.2404.14493

[2] O. Udalov, "A method for constructing quasi-random peaked quantum circuits," *arXiv preprint arXiv:2508.07491*, 2025. DOI: 10.48550/arXiv.2508.07491

[3] H. Gharibyan, M. Z. Mullath, N. E. Sherman, V. P. Su, H. Tepanyan, and Y. Zhang, "Heuristic quantum advantage with peaked circuits," *arXiv preprint arXiv:2510.25838*, 2025. DOI: 10.48550/arXiv.2510.25838

[4] D. Bluvstein et al., "A fault-tolerant neutral-atom architecture for universal quantum computation," *Nature*, vol. 649, pp. 39–46, 2026. DOI: 10.1038/s41586-025-09848-5

[5] F. Butt et al., "Demonstration of measurement-free universal logical quantum computation," *Nature Communications*, vol. 17, no. 1, p. 995, 2026. DOI: 10.1038/s41467-026-68533-x

[6] U. Aseguinolaza, N. Sobrino, G. Sobrino, J. Jornet-Somoza, and J. Borge, "Error estimation in current noisy quantum computers," *Quantum Information Processing*, vol. 23, no. 5, p. 181, 2024. DOI: 10.1007/s11128-024-04384-z

[7] K. Bader et al., "Room temperature quantum coherence in a potential molecular qubit," *Nature Communications*, vol. 5, no. 5304, 2014. DOI: 10.1038/ncomms6304

[8] J. M. Zadrozny, J. Niklas, O. G. Poluektov, and D. E. Freedman, "Millisecond coherence time in a tunable molecular electronic spin qubit," *ACS Central Science*, vol. 1, no. 9, pp. 488–492, 2015. DOI: 10.1021/acscentsci.5b00338

[9] N. P. de Leon et al., "Materials challenges and opportunities for quantum computing hardware," *Science*, vol. 372, no. 6539, 2021. DOI: 10.1126/science.abb2823

[10] D. J. Reilly, "Challenges in scaling-up the control interface of a quantum computer," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 31.7.1–31.7.6. DOI: 10.1109/IEDM19573.2019.8993497

[11] F. Battistel et al., "Real-time decoding for fault-tolerant quantum computing: Progress, challenges and outlook," *Nano Futures*, vol. 7, no. 3, p. 32003, 2023. DOI: 10.1088/2399-1984/aceba6

[12] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, 2019. DOI: 10.1038/s41586-019-1666-5

[13] H.-S. Zhong et al., "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, 2020. DOI: 10.1126/science.abe8770

[14] Y. Wu et al., "Strong quantum computational advantage using a superconducting quantum processor," *Phys. Rev. Lett.*, vol. 127, p. 180501, 2021. DOI: 10.1103/PhysRevLett.127.180501

[15] L. S. Madsen et al., "Quantum computational advantage with a programmable photonic processor," *Nature*, vol. 606, pp. 75–81, 2022. DOI: 10.1038/s41586-022-04725-x

[16] Q. Zhu et al., "Quantum computational advantage via 60-qubit 24-cycle random circuit sampling," *Science Bulletin*, vol. 67, no. 3, pp. 240–245, 2022. DOI: 10.1016/j.scib.2021.10.017

[17] A. Zlokapa, B. Villalonga, S. Boixo, and D. A. Lidar, "Boundaries of quantum supremacy via random circuit sampling," *npj Quantum Information*, vol. 9, no. 36, 2023. DOI: 10.1038/s41534-023-00703-x

[18] B. Barak, C.-N. Chou, and X. Gao, "Spoofing linear cross-entropy benchmarking in shallow quantum circuits," *arXiv preprint arXiv:2005.02421*, 2020. DOI: 10.48550/arXiv.2005.02421

[19] C. Huang et al., "Classical simulation of quantum supremacy circuits," *arXiv preprint arXiv:2005.06787*, 2020. DOI: 10.48550/arXiv.2005.06787

[20] F. Pan, K. Chen, and P. Zhang, "Solving the sampling problem of the sycamore quantum circuits," *Phys. Rev. Lett.*, vol. 129, p. 90502, 2022. DOI: 10.1103/PhysRevLett.129.090502

[21] C. Oh, L. Jiang, and B. Fefferman, "Spoofing cross-entropy measure in boson sampling," *Phys. Rev. Lett.*, vol. 131, p. 010401, 2023. DOI: 10.1103/PhysRevLett.131.010401

[22] X. Gao, M. Kalinowski, C.-N. Chou, M. D. Lukin, B. Barak, and S. Choi, "Limitations of linear cross-entropy as a measure for quantum advantage," *PRX Quantum*, vol. 5, p. 010334, 2024. DOI: 10.1103/PRXQuantum.5.010334

[23] Y. Zhang, "Complexity and hardness of random peaked circuits," *arXiv preprint arXiv: 2510.00132*, 2025. DOI: 10.48550/arXiv.2510.00132

[24] R. Orús, "Tensor networks for complex quantum systems," *Nature Reviews Physics*, vol. 1, no. 9, pp. 538–550, 2019. DOI: 10.1038/s42254-019-0086-7

[25] A. Berezutskii et al., "Tensor networks for quantum computing," *Nature Reviews Physics*, vol. 7, no. 10, pp. 581–593, 2025. DOI: 10.1038/s42254-025-00853-1

[26] A. M. Dalzell, N. Hunter-Jones, and F. G. S. L. Brandão, "Random quantum circuits anticoncentrate in log depth," *PRX Quantum*, vol. 3, p. 010333, 2022. DOI: 10.1103/PRXQuantum.3.010333

[27] S. A. Moses et al., "A race-track trapped-ion quantum processor," *Phys. Rev. X*, vol. 13, p. 041052, 2023. DOI: 10.1103/PhysRevX.13.041052

[28] S. Bravyi, D. Gosset, and Y. Liu, "Classical simulation of peaked shallow quantum circuits," in *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, ser. STOC 2024, Vancouver, BC, Canada: Association for Computing Machinery, 2024, pp. 561–572. DOI: 10.1145/3618260.3649638

[29] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A*, vol. 70, p. 052328, 2004. DOI: 10.1103/PhysRevA.70.052328

[30] R. Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," *Annals of Physics*, vol. 349, pp. 117–158, 2014. DOI: 10.1016/j.aop.2014.06.013

[31] R. Sengupta, S. Adhikary, I. Oseledets, and J. Biamonte, "Tensor networks in machine learning," *European Mathematical Society Magazine*, no. 126, pp. 4–12, 2022. DOI: 10.4171/MAG/101

[32] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Phys. Rev. Lett.*, vol. 91, p. 147902, 2003. DOI: 10.1103/PhysRevLett.91.147902

[33] Y. Zhou, E. M. Stoudenmire, and X. Waintal, "What limits the simulation of quantum computers?" *Phys. Rev. X*, vol. 10, p. 041038, 2020. DOI: 10.1103/PhysRevX.10.041038

[34] G. Schieffer, S. Markidis, and I. Peng, "Harnessing cuda-q's mps for tensor network simulations of large-scale quantum circuits," in *2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2025, pp. 94–103. DOI: 10.1109/PDP66500.2025.00022

[35] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, "Matrix product states and projected entangled pair states: Concepts, symmetries, theorems," *Reviews of Modern Physics*, vol. 93, no. 4, 2021. DOI: 10.1103/revmodphys.93.045003

[36] V. Murg, F. Verstraete, Ö. Legeza, and R. M. Noack, "Simulating strongly correlated quantum systems with tree tensor networks," *Phys. Rev. B*, vol. 82, p. 205105, 2010. DOI: 10.1103/PhysRevB.82.205105

[37] A. V. Berezutskii, I. A. Luchnikov, and A. K. Fedorov, "Simulating quantum circuits using the multi-scale entanglement renormalization ansatz," *Phys. Rev. Res.*, vol. 7, p. 013063, 2025. DOI: 10.1103/PhysRevResearch.7.013063

[38] M. P. Zaletel and F. Pollmann, "Isometric tensor network states in two dimensions," *Phys. Rev. Lett.*, vol. 124, p. 037201, 2020. DOI: 10.1103/PhysRevLett.124.037201

[39] R. Alkabetz and I. Arad, "Tensor networks contraction and the belief propagation algorithm," *Phys. Rev. Res.*, vol. 3, p. 023073, 2021. DOI: 10.1103/PhysRevResearch.3.023073

[40] E. M. Stoudenmire and S. R. White, "Minimally entangled typical thermal state algorithms," *New Journal of Physics*, vol. 12, no. 5, p. 055026, 2010. DOI: 10.1088/1367-2630/12/5/055026

[41] A. J. Ferris and G. Vidal, "Perfect sampling with unitary tensor networks," *Phys. Rev. B*, vol. 85, p. 165146, 2012. DOI: 10.1103/PhysRevB.85.165146

[42] C. Huang et al., "Efficient parallelization of tensor network contraction for simulating quantum computation," *Nature Computational Science*, vol. 1, pp. 578–587, 2021. DOI: 10.1038/s43588-021-00119-7

[43] S.-J. Ran et al., "Tensor network contractions: Methods and applications to quantum many-body systems," *Lecture Notes in Physics*, 2020. DOI: 10.1007/978-3-030-34489-4

[44] J. Gray and S. Kourtis, "Hyper-optimized tensor network contraction," *Quantum*, vol. 5, p. 410, 2021. DOI: 10.22331/q-2021-03-15-410

[45] E. Pednault et al., "Pareto-efficient quantum circuit simulation using tensor contraction deferral," *arXiv preprint arXiv:1710.05867*, 2020. DOI: 10.48550/arXiv.1710.05867

[46] Q. Zhang, M. Saligane, H.-S. Kim, D. Blaauw, G. Tzimpragos, and D. Sylvester, "Quantum circuit simulation with fast tensor decision diagram," in *2024 25th International Symposium on Quality Electronic Design (ISQED)*, 2024, pp. 1–8. DOI: 10.1109/ISQED60706.2024.10528748

[47] T. Nguyen, D. Lyakh, E. Dumitrescu, D. Clark, J. Larkin, and A. McCaskey, "Tensor network quantum virtual machine for simulating quantum circuits at exascale," *ACM Transactions on Quantum Computing*, vol. 4, no. 1, 2022. DOI: 10.1145/3547334

[48] J. Faj, I. Peng, J. Wahlgren, and S. Markidis, "Quantum computer simulations at warp speed: Assessing the impact of gpu acceleration," *arXiv preprint arXiv:2307.14860*, 2023. DOI: 10.48550/arXiv.2307.14860

[49] M. Vallero, F. Vella, and P. Rech, "State of practice: Evaluating gpu performance of state vector and tensor network methods," *arXiv preprint arXiv:2401.06188*, 2025. DOI: 10.1016/j.future.2025.107927

[50] K. Karuppasamy, V. Puram, S. Johnson, and J. P. Thomas, "A comprehensive review of quantum circuit optimization: Current trends and future directions," *Quantum Reports*, vol. 7, no. 1, 2025. DOI: 10.3390/quantum7010002

[51] A. Javadi-Abhari et al., "Quantum computing with qiskit," *arXiv preprint arXiv:2405.08810*, 2024. DOI: 10.48550/arXiv.2405.08810

[52] Cirq Developers, "Cirq," *Zenodo*, 2025. DOI: 10.5281/ZENODO.4062499

[53] V. Bergholm et al., "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2022. DOI: 10.48550/arXiv.1811.04968

[54] The CUDA-Q development team, "Cuda-q," *Zenodo*, 2025. DOI: 10.5281/zenodo.15407754

[55] H. Bayraktar et al., "Cuquantum sdk: A high-performance library for accelerating quantum science," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 1050–1061. DOI: 10.1109/QCE57702.2023.00119

[56] L. Abdurakhimov et al., "Technology and performance benchmarks of iqm's 20-qubit quantum computer," *arXiv preprint arXiv:2408.12433*, 2024. DOI: 10.48550/arXiv.2408.12433

[57] L. Burgholzer et al., "The munich quantum software stack: Connecting end users, integrating diverse quantum technologies, accelerating hpc," *arXiv preprint arXiv:2509.02674*, 2025. DOI: 10.1145/3773656.3773669

[58] A. Deshpande, B. Fefferman, S. Ghosh, M. Gullans, and D. Hangleiter, "Peaked quantum advantage using error correction," *arXiv preprint arXiv:2510.05262*, 2025. DOI: 10.48550/arXiv.2510.05262