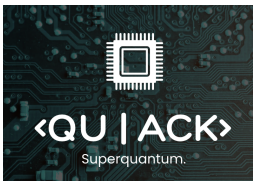


iQuHACK 2026 Write-Up: ⟨Qu|ack⟩

Marlon Jost, Tan Chun Loong, David Polzoni
Brandon Aikman, Tee Hui En

January–February 2026

1 Background

In fault-tolerant quantum computing (FTQC), the implementation of quantum gates comes at vastly different costs due to the requirements of quantum error correction. While Clifford gates [1] can be implemented transversally and thus incur minimal overhead in most error correction codes, non-Clifford gates like the T gate require significantly more expensive procedures. The T gate typically demands magic state distillation, a resource-intensive process that can consume thousands of physical qubits and multiple rounds of error correction to produce a single high-fidelity T gate. This asymmetry means that T-count [2] becomes the dominant factor in determining the overall computational cost of fault-tolerant quantum algorithms, often dwarfing the contribution of all other gates combined.

This concern is unique to the fault-tolerant regime. In the current era of noisy intermediate-scale quantum (NISQ) devices, where error correction is not yet practical, all gates suffer from similar error rates and there is no particular penalty for non-Clifford operations. However, as the field progresses toward fault-tolerant architectures capable of running large-scale algorithms like Shor's algorithm or quantum simulation of molecular systems, minimizing T-count becomes essential for feasibility. The Clifford+T gate set is universal for quantum computation, meaning any quantum algorithm can be decomposed into sequences of these gates, but achieving this decomposition efficiently requires sophisticated compilation techniques. The challenge of approximating arbitrary unitaries with optimal T-count while maintaining acceptable error bounds has become a central problem in quantum compilation, with direct implications for the practical viability of future quantum computers.

Here, we present our methods and results from tackling the iQuHACK 2026 Superquantum challenge. This challenge included eleven problems we were asked to solve as described below. For all of these problems, we were limited to the gate set 1.

[1] including Hadamard (H), CNOT, and Phase (S) gate
[2] the number of T gates in a circuit

$$\{ H, T, T^\dagger, CNOT, S, S^\dagger \} \tag{1}$$

Our solutions were evaluated according to the T-count and the circuits accuracy. The accuracy was either evaluated as fidelity or as the norm distance between our calculated and target unitaries. Details on the prompts for each problem can be found on Superquantum's Git repository [1].

2 Methods Overview

There were four main approaches used to solve the challenges. The first, and most trivial method, utilized Qiskit's default transpiler. For this approach, a given problem unitary would first be implemented as a Qiskit circuit. Often, this initial circuit would include gates not that were outside our allowed gate set 1. The subsequent transpilation process would thus be set to use only our allowed gate set. This very effectivley constrained the gate set. The transpiler would also work to minimize the number of gates by default.

Our second method utilized **manual optimization**. In this baseline method, we perform a direct exploration of the circuit space by enumerating possible gate combinations. Here's the workflow, first, we define the distance between the target matrix and the synthesized circuit using the operator norm

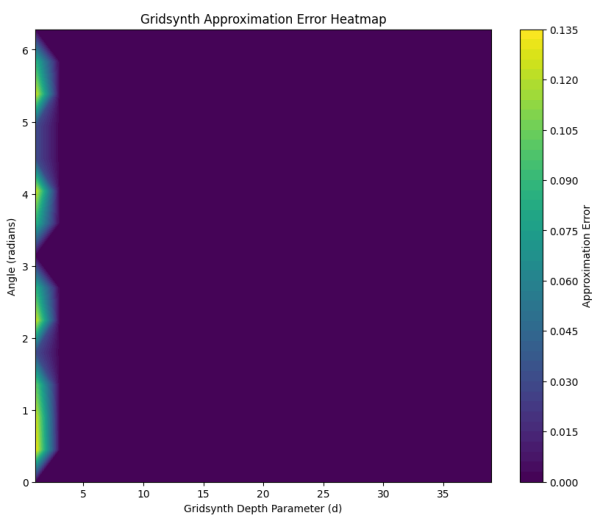
$$d(U_{\text{target}}, U_{\text{circuit}}) = \left\| U_{\text{target}} - e^{i\phi} \prod_{j=1}^{n} g_j \right\| \tag{2}$$

Then the algorithm will explores the space of all possible combination of gates up to a certain depth n. For k possible gates and a circuit depth d, the search space is:

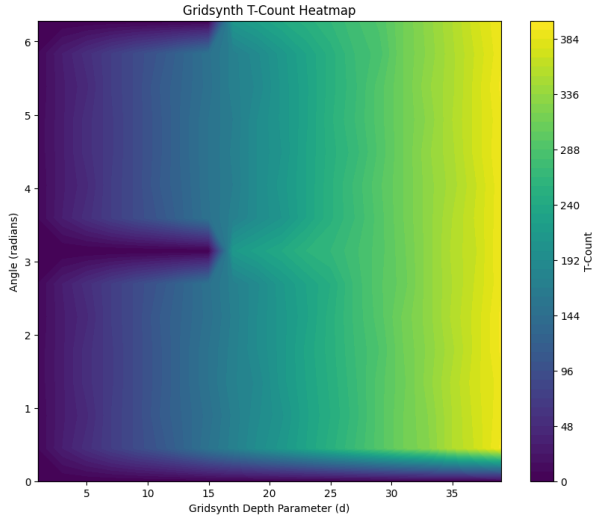
$$\text{Complexity} = O(k^d)$$

Our third method utilized the gridsynth optimizer developed by Peter Selinger [4, 3]. This software package is designed specifically to efficiently approximate arbitrary Rz gates up to some ϵ using Clifford and T gates. It achieves T-counts that are within $O(log(log(1/\epsilon)))$ of optimal, all within a runtime of $O(polylog(1/\epsilon))$. To use the method, an initial Qiskit circuit would be constructed and each instance of an Rz gate would be replaced with a call to gridsynth. The results of gridsynth would be parsed to apply the appropriate gates to our quantum circuit to replace the given Rz gate. We also analyzed the affect of the angle of rotation of the Rz gate we were approximating and the parameter d [3] on the final approximation error and T-gate count in figures 1a and 1b.

[3] $\epsilon = 10^{-d}$



(a) Approximation error of an Rz gate as a function of d and the Rz angle when using gridsynth.



(b) T-count for a single Rz gate as a function of d and the Rz angle when using gridsynth.

Our fourth method used the **gridsynth unitary synthesis method** in Qiskit (Same algorithm as the `pygridsynth` package, for a different use-case) [2]. When the unitary synthesis method is not specified, Qiskit defaults to a synthesis tool that provides a high T-count. Because one of the main drivers of the challenges was creating low-T circuits, we found that the gridsynth method provided significantly better T-count metrics.

Another method we employed is based on Reed–Muller decoding and the `rmsynth` framework for T-count optimization.

The *odd part* of a phase polynomial, namely the coefficients that are odd, can be viewed as a **binary word** of length $2^n - 1$. Each bit corresponds to a non-empty subset $S \subseteq \{1, \ldots, n\}$.

Punctured Reed–Muller codes $\mathrm{RM}(r, n)^*$ inhabit this same vector space. Decoding in such a code amounts to finding a codeword that is close to the given binary word in Hamming distance.

The tool `rmsynth` exploits this structure to reduce T-count. It takes as input a **CNOT + phase** circuit, or equivalently a $\mathbb{Z}_8$ **phase vector**. From this representation it extracts the phase-polynomial coefficients $a_S \in \mathbb{Z}_8$. The odd coefficients are interpreted as a received word and decoded in $\mathrm{RM}(n-4, n)^*$ (or a closely related code), producing a correction codeword. This correction is added to the original coefficients modulo 8, after which a new CNOT + phase circuit is synthesized.

The resulting circuit implements the **same diagonal unitary**, meaning that the phases agree modulo 2π, but typically uses **fewer T gates** when the decoder succeeds in finding a low-weight correction.

In summary, `rmsynth` minimizes the T-count for circuits that are, or can be expressed as, **CNOT + phase** circuits.

Consider a diagonal unitary of the form:

$$U\,|x\rangle = e^{i\phi(x)}\,|x\rangle\,.$$

From the phase values $\phi(0), \ldots, \phi(2^n - 1)$ one can invert the phase-polynomial relation to recover coefficients $a_S \in \mathbb{Z}_8$. These coefficients are assembled into a $\mathbb{Z}_8$ **vector** of length $2^n - 1$, with one

entry for each non-empty subset S. Running `rmsynth` on this vector produces an optimized CNOT + phase circuit. This circuit can then be emitted in OpenQASM using only `cx`, `t`, and `tdg` gates, with `h` gates appearing only when conjugation by Hadamards is required.

A general unitary is first decomposed into Hadamard and diagonal blocks, for example as:

$$U = \tilde{H} \cdot D \cdot \tilde{H}.$$

The diagonal block D is expressed as a phase polynomial and converted into a $\mathbb{Z}_8$ vector, to which `rmsynth` is applied. This yields an optimized circuit consisting solely of CNOT + phase gates. Finally, Hadamards are reinserted as needed, producing a circuit over the gate set

$$\{\texttt{h}, \texttt{t}, \texttt{tdg}, \texttt{cx}\}.$$

Consequently, in our approach `rmsynth` is applied to every diagonal component, and therefore to every challenge instance that contains a diagonal block.

In many cases, multiple approaches were attempted and the best result was used. For instance, since the basic version of Qiskit transpilation was simple to implement, the sophisticated methods we developed/implemented later on, we would then usually get better results.

Often, we would use a combination of the Qiskit gridsynth unitary synthesis method, `rmsynth`, hand derivation, or brute-forcing algorithms. Our team benefited largely from having members of different backgrounds, spanning physics, computer science/engineering, and cybersecurity. Oftentimes, our members strongest in physics would guide the theoretical derivation of a problem, and those stronger in software development would implement their algorithm or circuit.

Our best solutions are given in 1. Note that "best solution" is somewhat subjective, as we must often make choose between a more accurate solution with a higher T-count and a less accurate solution with a lower T-count.

3 Challenges

3.1 Challenge 1: Sanity Check — Controlled-Y Compilation

The target unitary for this challenge is the controlled-Y gate,

$$\mathrm{CY} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix},$$

which applies a Pauli-Y operation to the target qubit conditioned on the control qubit being in the state $|1\rangle$.

Using a standard Clifford basis-change identity, the controlled-Y gate can be expressed as

$$\mathrm{CY} = (I \otimes H)\,\mathrm{CZ}\,(I \otimes H),$$

thereby reducing the compilation task to that of implementing a controlled-Z gate. Since CZ is diagonal, with phase $\phi(11) = \pi$ and all other computational basis phases equal to zero, it can be

Qn.	Best Distance / Fidelity	Best T-Count
1	0	0
2	4×10^{-11}	208
3	0.4450	0
4	1.22×10^{-10}	607
5	2.46×10^{-16}	0
6	1.0658	1
7	Fidelity ≈ 0.9999999999998	716
8	1.51×10^{-10}	828
9	2.0543	0
10	1.06	1
11	0.765	4
12	UND	2043

Table 1: Summary of approaches and best results across Problems 1–12 according to the provided autograder.

encoded as a $\mathbb{Z}_8$ phase vector and compiled using `rmsynth`. This yields an optimized Clifford+T realization of the diagonal controlled-phase operator.

The final circuit is obtained by wrapping the synthesized diagonal block with Hadamard gates on the target qubit,

$$H_{\text{target}} \longrightarrow \text{CZ}_{\text{synthesized}} \longrightarrow H_{\text{target}}.$$

Alternatively, one may use the known minimal decomposition

$$\text{CY} = S^{\dagger}_{\text{target}}\,\text{CX}\,S_{\text{target}}, \qquad S = T^2,$$

which is already Clifford+T. In our implementation, however, the diagonal CZ block can still be replaced by an `rmsynth`-optimized synthesis in order to maintain a consistent compilation workflow.

3.2 Challenge 2: Qiskit Synthesis of a Controlled-R_y Gate

We employ the standard basis-change identity

$$\text{CR}_y(\theta) = (I \otimes H)\,\text{CR}_z(\theta)\,(I \otimes H),$$

which reduces the synthesis of $\text{CR}_y(\pi/7)$ to that of the diagonal controlled-Z rotation $\text{CR}_z(\pi/7)$. In the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, with control qubit q_0 and target qubit q_1 as in Qiskit, the diagonal phases of $\text{CR}_z(\pi/7)$ are

$$[\,0,\ 0,\ -\tfrac{\pi}{14},\ +\tfrac{\pi}{14}\,].$$

In practice, the resulting Clifford+T circuit is synthesized using Qiskit's transpiler when available. When an explicit diagonal representation is required, the phase angles are rounded into a $\mathbb{Z}_8$ phase vector and passed to the `rmsynth` compiler to obtain an optimized CNOT+phase decomposition. This diagonal block is then wrapped with Hadamard gates on the target qubit to recover $\text{CR}_y(\pi/7)$.

3.3 Challenge 3: Exact Diagonal Unitary Compilation with `rmsynth`

This challenge involves the fully diagonal operator $Z \otimes Z$, which acts as $+1$ on $|00\rangle$ and $|11\rangle$ and as -1 on $|01\rangle$ and $|10\rangle$. The corresponding phases, ordered as $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, are therefore

$$[\, \tfrac{\pi}{7},\ -\tfrac{\pi}{7},\ -\tfrac{\pi}{7},\ \tfrac{\pi}{7}\,].$$

These phases are converted into a $\mathbb{Z}_8$ phase vector, after which `rmsynth` (or Qiskit's unitary synthesis) is applied. The resulting circuit is emitted in OpenQASM using only `cx`, `t`, and `tdg` gates.

3.4 Challenge 4: Manual Decomposition of $XX + YY$

The operator $e^{i\frac{\pi}{7}(XX+YY)}$ admits the decomposition

$$e^{i\frac{\pi}{7}(XX+YY)} = (I \otimes H)\, e^{i\frac{\pi}{7}Z\otimes Z}\, (I \otimes H).$$

The diagonal unitary $e^{i\frac{\pi}{7}Z\otimes Z}$ has the same phase structure as in Challenge 3,

$$[\, \tfrac{\pi}{7},\ -\tfrac{\pi}{7},\ -\tfrac{\pi}{7},\ \tfrac{\pi}{7}\,].$$

This diagonal block is optimized using `rmsynth`, after which Hadamard gates are applied before and after on the target qubit to recover the original operator.

3.5 Challenge 5: Recognition as a SWAP Operation

For two qubits, the identity $XX + YY + ZZ = 2\,\mathrm{SWAP} - I$ implies

$$e^{i\frac{\pi}{4}(XX+YY+ZZ)} = e^{i\pi/4}\,\mathrm{SWAP},$$

which differs from the SWAP gate only by a global phase. As a result, no T gates are required: the circuit consists solely of a SWAP operation, implemented using three CNOT gates. This circuit is Clifford-only, and `rmsynth` is not needed.

3.6 Challenge 6: Transverse-Field Ising Time-Evolution Operator

We approximate the unitary

$$e^{i\frac{\pi}{7}(XX+ZI+IZ)}$$

using first-order Trotterization,

$$e^{i\frac{\pi}{7}(XX+ZI+IZ)} \approx e^{i\frac{\pi}{7}ZI}\, e^{i\frac{\pi}{7}IZ}\, e^{i\frac{\pi}{7}XX}.$$

The term $e^{i\frac{\pi}{7}XX}$ is implemented as

$$e^{i\frac{\pi}{7}XX} = H_0\, e^{i\frac{\pi}{7}ZZ}\, H_0,$$

where H_0 denotes a Hadamard gate on qubit 0. The ZZ component uses the same diagonal phase structure as in Challenges 3 and 4,

$$[\, \tfrac{\pi}{7},\ -\tfrac{\pi}{7},\ -\tfrac{\pi}{7},\ \tfrac{\pi}{7}\,],$$

which is optimized using `rmsynth`, followed by Hadamard conjugation.

3.7 Challenge 7: High-Fidelity Two-Qubit State Preparation

The target state is generated using `qiskit.quantum_info.random_statevector(4, seed=42)`. We use Qiskit to synthesize a corresponding Clifford+T circuit, with a fallback strategy when an exact synthesis is unavailable. Any diagonal sub-blocks arising in this decomposition may be further optimized using `rmsynth`. The final output circuit is expressed exclusively in terms of `h`, `t`, `tdg`, and `cx` gates.

3.8 Challenge 8: Recognized as 2-Qubit QFT

Problem 8 provides the unitary:

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix},$$

which exhibits strong algebraic structure suitable for direct synthesis. Using `rmsynth`, we obtained an essentially exact Clifford+T implementation with operator norm distance and a best T-count.

This indicates that the unitary can be expressed entirely using Clifford operations without requiring costly non-Clifford resources.

3.9 Challenge 9: Recognized as Quantum Walk

Problem 9 presents another structured 4×4 unitary with embedded phase relationships. Similarly, we applied `rmsynth` synthesis rather than manual enumeration.

The result preserves the low non-Clifford cost, the approximation error suggests further circuit-level refinement or alternative decomposition methods may be required for improved precision.

Summary

Overall, Problems 8 and 9 demonstrate how structured unitaries can often be compiled efficiently using phase-polynomial based synthesis tools such as `rmsynth`, producing circuits with minimal or zero T-gate overhead.

3.10 Challenge 10 : (Brute-Force)

In Problem 10, the goal is to approximate a given two-qubit target unitary U_{target} using a circuit composed of a restricted gate set Rather than searching over all possible circuit structures, we begin with a fixed entangling "skeleton" circuit consisting of alternating CNOT and Hadamard gates. This provides a deterministic starting point and reduces the overall search space. The skeleton structure can be summarized as repeated layers of:

$$\text{CNOT} \rightarrow H \rightarrow \text{CNOT} \rightarrow H \cdots$$

The unitary produced by this skeleton serves as the baseline approximation.

To improve the approximation, we introduce tunable phase corrections between each skeleton gate. Specifically, we insert adjustable powers of the T gate:

$$T^k, \qquad k \in \mathbb{Z}_8,$$

applied independently to each qubit. Each "slot" corresponds to a pair of exponents (k_0, k_1), meaning that T^{k_0} is applied on qubit 0 and T^{k_1} on qubit 1.

This transforms the search problem into a discrete optimization over phase parameters instead of an unrestricted brute-force enumeration of all circuits.

To optimize the slot exponents efficiently, we employ a coordinate descent procedure:

- Initialize all phase slots to $(0, 0)$.
- Iterate through each slot and locally test small candidate updates $k \in \{0, \pm 1, \pm 2, \pm 3\}$.
- Select the exponent that yields the greatest improvement in distance.
- Repeat for multiple passes until convergence.

In addition, when two candidates achieve nearly identical distance values, we prefer the one with fewer T or $T^\dagger$ gates in order to reduce circuit cost.

The best-performing circuit is exported as a QASM file (`challenge_10.qasm`) for submission.

3.11 Challenge 11: Derived by hand

- **Convert phases to a Z_8 phase table:** All given phases are multiples of $\pi/4$ (e.g., $0, \pi, 5\pi/4, 7\pi/4, 3\pi/2$), so we map

  $$\varphi(x) = \frac{\pi}{4}\, p(x), \qquad p(x) \in \mathbb{Z}_8.$$

  This converts the specification into an integer-valued phase function $p(x)$ over 4 bits.

- **Phase-polynomial viewpoint:** We represent $p(x)$ as a *phase polynomial* over parities:

  $$p(x) \equiv \sum_{S \subseteq \{0,1,2,3\}} a_S \cdot \left(\bigoplus_{j \in S} x_j \right) \pmod 8,$$

  where $\bigoplus$ denotes XOR. Each term corresponds to applying a T-power on the parity of a subset of qubits.

- **Circuit construction (compute–phase–uncompute):** For each parity term $\left(\bigoplus_{j \in S} x_j\right)$ with coefficient a_S:

  1. Use a short CNOT network to compute the parity onto a chosen "accumulator" qubit.
  2. Apply the required phase using only $T/T^\dagger$ powers (note $S = T^2$ and $Z = T^4$).

3. Uncompute the parity by reversing the CNOT network.

Coefficients that are even modulo 8 can be absorbed into Clifford phases, while odd coefficients contribute to the non-Clifford (T) cost.

- **Optimization idea (done by hand):** We reduce the overall T-count by: (i) combining repeated parity terms, (ii) canceling inverse phases (T vs. $T^\dagger$), and (iii) choosing parities that share CNOT scaffolding so compute/uncompute networks can be reused.

3.12 Challenge 12: Bonus Question

We implement the unitary

$$U := \prod_{j=1}^{m} \exp\left(-i\frac{\pi}{8}k_j P_j\right),$$

where each P_j is a Pauli operator and $k_j \in \mathbb{Z}_8$, by first compiling the circuit into the Clifford+T gate set and isolating its diagonal phase structure. After transpilation, all non-diagonal Clifford operations are treated as basis changes, while diagonal gates are interpreted as a phase polynomial over $\mathbb{Z}_8$. In this representation, single-qubit phase gates contribute terms of the form $\exp(i\pi k x_j/4)$, where $x_j \in \{0,1\}$ is a computational basis variable, and controlled-NOT gates act by linear transformations on these variables.

The resulting phase polynomial is then optimized using `rmsynth`, which performs algebraic simplification and parity-term cancellation to minimize the total T-count required to implement the diagonal unitary. This optimization operates purely on the phase polynomial representation, independent of the surrounding Clifford structure. Finally, the optimized polynomial is synthesized back into a Clifford+T circuit. This approach exploits the fact that diagonal unitaries with $\pi/4$-quantized phases admit a compact polynomial description, allowing global T-count optimization that is not accessible through local circuit rewriting alone.

References

[1] *2026-Superquantum/challenge.pdf at main · iQuHACK/2026-Superquantum*. en. URL: https://github.com/iQuHACK/2026-Superquantum/blob/main/challenge.pdf (visited on 02/01/2026).

[2] *IBM: Qiskit 2.3 Release Summary*. URL: hhttps://www.ibm.com/quantum/blog/qiskit-2-3-release-summary (visited on 02/01/2026).

[3] *Peter Selinger: Quantum Synthesis*. URL: https://www.mathstat.dal.ca/~selinger/newsynth/#usage (visited on 02/01/2026).

[4] Neil J. Ross and Peter Selinger. *Optimal ancilla-free Clifford+T approximation of z-rotations*. arXiv:1403.2975 [quant-ph]. June 2016. DOI: 10.48550/arXiv.1403.2975. URL: http://arxiv.org/abs/1403.2975 (visited on 02/01/2026).