

Unified Pauli-Rotation Synthesis for Relieving CX-Count Overhead in Tableau-Based Quantum Circuit Optimization Flow

Yi-Hsiang Kuo, Hsiang-Chun Yang, Hsin-Yu Chen, Chung-Yang (Ric) Huang
Graduate Institute of Electronics Engineering, National Taiwan University
 yhkuo1521@ntu.edu.tw, cyhuang@ntu.edu.tw

Abstract—Tableau representation offers an efficient framework for describing quantum circuits and has been widely adopted in tableau-based quantum circuit optimization (QCO) flows. While these flows can substantially reduce the T -count, which is critical for fault-tolerant implementations, resynthesizing the optimized tableaux back into circuits often introduces excessive two-qubit gates (2Q-gates), leading to significant 2Q-count overhead. To address this issue, we propose a unified synthesis strategy that departs from the conventional tableau-by-tableau approach. Instead of resynthesizing each tableau in isolation, our method consolidates Clifford and Pauli rotation tableaux and applies a holistic resynthesis algorithm. This unified treatment contrasts with prior approaches and enables systematic reduction of the overall 2Q-count. Experimental results on standard Clifford+ T benchmarks show that our method achieves a Geomean 2Q-count ratio of 1.28, compared to 4.24 for TKET and 1.93 for LazySynth—the state-of-the-art tableau synthesis approach—, demonstrating that unified synthesis effectively mitigates the 2Q-gate overhead in tableau-based QCO.

Index Terms—Quantum Computing, Quantum Circuit Optimization, Tableau Synthesis

I. INTRODUCTION

Quantum computing has emerged as a powerful paradigm for addressing problems that are intractable for classical computers, with a wide range of potential applications across science and engineering [1]–[4]. Realizing such applications typically demands circuits with substantial resources, making quantum circuit optimization (QCO) an essential component of the design flow. At the logical level, QCO transforms a given circuit into a functionally equivalent one with reduced resource cost, commonly measured by metrics such as circuit depth, two-qubit gate count (2Q-count), or T -count. While the T -count is a dominant metric in fault-tolerant quantum computing, the 2Q-count also plays a critical role in practice due to its higher implementation overhead and strong interaction with routing and hardware constraints [5], [6]. Consequently, it is important to consider the 2Q-count as a complementary optimization objective alongside the T -count.

Among the diverse techniques for QCO, tableau-based optimization flows constitute a widely adopted strategy. By representing Clifford and non-Clifford operations in compact tableau forms, they enable systematic Clifford circuit transformations [7] and have been shown to be highly effective in reducing the costly T -count [8]–[10]. However, despite these advantages, tableau-based optimization flows often incur substantial 2Q-count overhead during resynthesis [11]. This

overhead stems from the fact that existing synthesis methods operate in a tableau-by-tableau manner, where each sub-tableau is resynthesized in isolation, preventing cross-tableau cancellation. This introduces redundant 2Q-gates and limits the overall effectiveness of tableau-based optimization flows. The limitations of existing approaches are discussed in the next section.

In this paper, we propose a unified synthesis framework that consolidates Clifford and Pauli rotation tableaux into a holistic resynthesis process. Instead of synthesizing each tableau in isolation, we defer all Clifford operations to the end of the circuit, resulting in a unified Pauli rotation tableau followed by a single Clifford tableau. Within this framework, we introduce the Pauli-MST algorithm, abbreviated as $pMST$, which extends the minimum spanning tree paradigm introduced in [12] from diagonal parity tables to general Pauli rotations. By jointly considering the costs of both Pauli rotations and the residual Clifford tableau, $pMST$ enables globally cost-aware synthesis and systematically reduces the overall 2Q-count.

We benchmark our method against three representative synthesis strategies: TKET [13], LazySynth [11], and a naive unified baseline referred to as Basic. Experimental results on standard Clifford+ T benchmarks indicate that our method achieves the lowest 2Q-count, with a geometric mean ratio of 1.28, i.e., the 2Q-count after synthesis is on average 1.28 \times of the original circuit, compared to 1.93 for LazySynth, 2.14 for Basic, and 4.24 for TKET. These results demonstrate that unified synthesis with $pMST$ algorithm achieves substantial improvements over prior approaches.

We further examine whether this advantage persists when additional local optimization passes are applied. While all synthesis methods benefit from these optimizations, our approach consistently delivers the lowest 2Q-count. These results demonstrate that our method fundamentally improves the resynthesis of Clifford and Pauli rotation tableaux, thereby advancing the state-of-the-art in tableau-based quantum circuit optimization.

II. PRELIMINARIES AND RELATED WORK

This section provides the preliminaries for understanding tableau-based QCO, covering Pauli strings, Pauli rotations, tableau representations, and the overall flow structure.

A. Pauli Strings

A Pauli string on n qubits is represented to $\{I, X, Y, Z\}^{\otimes n}$. Such a string can be encoded as a binary vector of length $2n$: the first n entries (z_1, \dots, z_n) indicate whether a Z component appears on each qubit, and the next n entries (x_1, \dots, x_n) indicate the X components. A Y operator is represented by setting both $z_i = 1$ and $x_i = 1$. This binary encoding provides an efficient algebraic representation that serves as the foundation for tableau forms. In practice, an additional entry may be appended to record auxiliary information (e.g., a phase).

B. Pauli Rotations

A Pauli rotation with respect to a Pauli string P and an angle θ is defined as

$$R_P(\theta) := e^{-i\theta P/2}. \quad (1)$$

Following the encoding introduced in Sec. II-A, such a rotation can be encoded as a column vector

$$[z_1 \cdots z_n \ x_1 \cdots x_n \ \theta]^T,$$

Each Pauli rotation can be implemented by a single-qubit phase gate together with a surrounding Clifford circuit [14]. Particularly, in the Clifford+ T setting, since the only non-Clifford gate, T gate is equivalent to $R_Z(\pi/4)$, it is common to fix $\theta = \pi/4$ and simplify the notation as $R(P)$ [15].

When conjugated by a Clifford operator C , the Pauli rotation can be rewritten as another Pauli rotation on the conjugated Pauli string, followed by the Clifford operator itself:

$$CR(P)C^\dagger = R(CPC^\dagger). \quad (2)$$

By prepending C to the Pauli rotation, we get

$$CR(P) = R(CPC^\dagger)C. \quad (3)$$

Equation (3) allows Clifford operations to be systematically commuted from one side to the other side of the circuit, forming the initial setting of the tableau conversion procedure introduced in the next section.

In addition, two Pauli rotations commute if and only if their corresponding Pauli strings commute. In other words, the commutation relation is preserved when passing from Pauli strings to Pauli rotations:

$$[P_1, P_2] = 0 \quad \Leftrightarrow \quad [R(P_1), R(P_2)] = 0. \quad (4)$$

When the Pauli string P consists only of I and Z , we refer to $R(P)$ as a *diagonal* Pauli rotation. Such diagonal Pauli rotations are particularly important in optimization flows, as they correspond to non-Clifford phase gates that can be merged or eliminated during advanced T -count reduction approach [8]–[10]. Further details will be discussed subsequently.

C. Clifford Tableau and Pauli Rotation Tableau

Tableau form provides a compact binary encoding for both Clifford operators and sequences of Pauli rotations. For an n -qubit Clifford operator, the *Clifford tableau* is represented by a $(2n+1) \times 2n$ binary matrix, where each column encodes a Pauli

string together with a phase bit.¹ Elementary Clifford gates such as Hadamard, Phase (S), and CX correspond to specific row operations on this tableau. This representation enables Clifford circuits to be systematically transformed and resynthesized [7]. Similarly, a sequence of Pauli rotations can be expressed as a *Pauli rotation tableau*, where each column corresponds to a $R_P(\theta)$.

We say that a Pauli rotation tableau is diagonal if all of its constituent Pauli rotations are diagonal, i.e., their X -part is entirely zero. In this case, the tableau is equivalent to the *phase-polynomial* representation [9], [16], [17], where each column corresponds to a phase term.

These tableau representations underlie modern tableau-based optimization flows. Clifford tableaux can be directly resynthesized via systematic row operation, while Pauli rotation tableaux—especially when diagonal—admit algebraic simplifications that enable substantial T -count reductions.

D. Tableau-based QCO

Tableau-based QCO is typically performed in three steps: converting the quantum circuit into tableaux, optimizing the tableaux to reduce the T -count, and resynthesizing the optimized tableaux back to a quantum circuit. We discuss the details of each step in the following.

1) *Circuit-to-Tableaux Conversion*: Given a Clifford+ T quantum circuit Q_C , we can represent each Clifford gate as a Clifford tableau and each non-Clifford gate (e.g., a T gate) as a Pauli rotation tableau. Any Clifford tableau adjacent to a Pauli rotation tableau can be commuted across it using (3). By repeatedly applying this rule, the circuit can be rearranged such that all Clifford gates are collected on one side, while the non-Clifford gates remain on the other. As illustrated in the first step of Fig. 1, letting \mathcal{T} and \mathcal{S} denote the Clifford and Pauli rotation tableaux of these two parts, respectively, the overall unitary of the circuit can then be expressed as $U(Q_C) = U(\mathcal{S})U(\mathcal{T})$, where $U(\cdot)$ denotes the unitary operator associated with a circuit or tableau.

2) *Tableaux Optimization*: After the Clifford and Pauli rotation parts are separated, the optimization proceeds in two phases. First, a phase-merging algorithm [15], [18] eliminates redundancies in the Pauli rotations \mathcal{S} and yields a preliminary reduction in the T -count. In the second phase, advanced phase-polynomial-like methods [8]–[10] are evoked for further T -count reduction. However, the Pauli rotation tableau is generally non-diagonal; therefore, these methods cannot be directly applied. To enable them, InternalHopt [19] partitions the circuit into alternating Clifford and diagonalized Pauli rotation (DPR) tableaux while minimizing the number of Hadamard gates (third stage in Fig. 1). Phase-polynomial-like methods are then applied on each DPR sub-tableau to further reduce the T -count. Note that all reduction related to the T -count are completed during tableau optimization, and the T -count remains unchanged afterward.

¹Unlike the classical definition of Clifford tableau in [7], where Pauli strings are encoded as rows, here we adopt a column-based encoding in this paper.

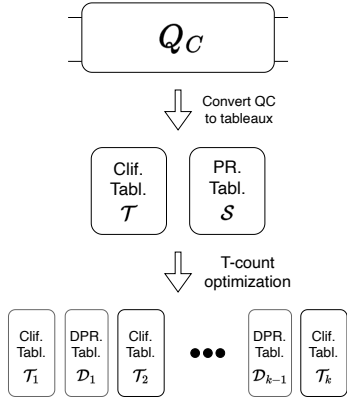


Fig. 1: Tableau-based QCO flow.

3) *Tableau Resynthesis*: In the final step of a tableau-based QCO flow, a Clifford or Pauli rotation tableau is resynthesized back to a quantum circuit. For a Clifford tableau, this is typically achieved by performing a sequence of row operations, each of which can be directly mapped to a corresponding Clifford gate and eventually reducing the tableau to an identity [7], [20], [21]. For a diagonalized Pauli rotation tableau, the resynthesis can be formulated as a parity-tableau resynthesis problem [12], [22], [23], where the Z -part of the tableau is simplified column by column into one-hot vectors through successive CX operations. Whenever a column is reduced to a single 1, the associated phase rotation (e.g., a T gate) is placed on that qubit and then removed from the tableau, resulting in a CX+ T subcircuit. More details on the resynthesis step, including its role in current tableau synthesis strategies, will be provided in the next section.

III. MOTIVATION AND PROBLEM STATEMENT

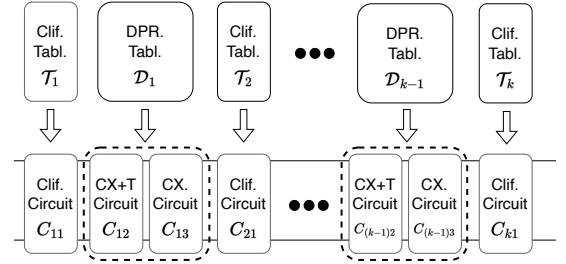
Existing tableau synthesis approaches reconstruct circuits from each tableau in isolation, which often incurs excessive 2Q-count overhead due to the lack of cross-tableau consideration. In this section, we first review the current synthesis strategy and then point out its weakness, which motivates the proposed unified synthesis framework.

A. Current Tableau Synthesis Method

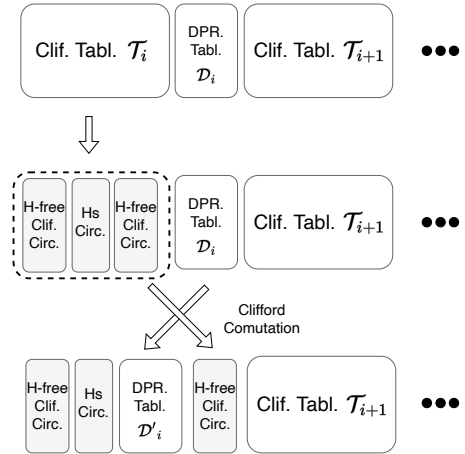
In a tableau-based QCO flow, the optimized circuit is represented as a sequence of intermediate tableaux. Each tableau can then be resynthesized into gates using the standard Clifford or Pauli rotation synthesis procedures introduced earlier, but different strategies exist for deciding when this resynthesis should take place.

The most straightforward approach is the *eager strategy*, in which every tableau is immediately resynthesized once it is produced. However, this often causes 2Q-gates to become widely scattered across different tableaux, preventing them from being grouped efficiently in the final circuit, as illustrated in Fig. 2(a).

To address this issue, LazySynth [11] was introduced. Instead of extracting every Clifford subcircuit as soon as a Clifford



(a) Eager synthesis approach.



(b) Lazy synthesis approach.

Fig. 2: Illustrative examples of tableau synthesis strategies.

tableau appears, LazySynth defers the extraction of the *H-free* Clifford tableau that follows a Pauli rotation tableau and carries the corresponding Clifford tableau forward. As illustrated in Fig. 2(b), a Clifford tableau is typically decomposed as an H-free segment, an H layer, and another H-free segment. LazySynth postpones the extraction of the last H-free segment and merges it with subsequent Clifford tableaux using commutation rules. The merged block can then be decomposed again so that its H-free part is further deferred. This reduces scattered 2Q-gates and yields more structured Clifford tableau in the final circuit.

Nevertheless, both eager and lazy synthesis ultimately treat tableaux in isolation, a characteristic that will be further examined in the next subsection.

B. Challenges in Current Methods

The tableau-by-tableau treatment prevents the synthesizer from capturing global cost interactions across tableaux. In particular, when Clifford operations are synthesized separately within each tableau, they are optimized only in a local context. This produces fragmented Clifford subcircuits that miss opportunities for cross-tableau gate cancellation, resulting in redundant 2Q-gates in the final circuit.

The underlying reason is that existing resynthesis approaches rely heavily on the diagonalized Pauli rotation tableaux, where all operators commute and can be efficiently syn-

Algorithm 1 High-level view of pMST algorithm

Input: A Pauli rotation tableau \mathcal{S} and a Clifford tableau \mathcal{T} on n qubits

Output: Clifford+T circuit C

- 1: **procedure** $pMST(\mathcal{S}, \mathcal{T})$
 - 2: $C \leftarrow$ empty circuit on n qubits
 - 3: **while** \mathcal{S} is not empty **do**
 - 4: Identify all executable rotations.
 - 5: Select the lowest-cost rotation.
 - 6: Synthesize the rotation and update the tableaux.
 - 7: **end while**
 - 8: Synthesize the residual Clifford tableau.
 - 9: **return** C
 - 10: **end procedure**
-

thesized using parity-table techniques. However, propagating Hadamards across Pauli rotation tableaux generally introduces non-diagonal operators such as X or Y , breaking the commutativity and rendering the parity-table approach inapplicable. To avoid this difficulty, prior works restrict themselves to partial deferral strategies (e.g., LazySynth), which alleviate but cannot fully eliminate fragmentation. A very recent work [23] has also considered cross-tableau interactions in a tableau-by-tableau setting, particularly across phase-polynomial blocks. While this alleviates fragmentation to some extent, the synthesis process remains fundamentally tableau-by-tableau.

To address this limitation, we adopt a more aggressive strategy: propagate all Clifford operations to the end of the circuit, as illustrated in Fig. 3. This strategy combines intermediate Clifford tableaux and enables a unified treatment of Pauli rotations. We will introduce our proposed algorithm in the next section.

IV. PROPOSED METHOD

The limitations discussed in the previous section motivate the introduction of a unified synthesis strategy that consolidates Clifford operations into a single tableau. The concept contrasts with tableau-by-tableau synthesis, which often leaves fragmented Clifford subcircuits and thereby leads to excessive 2Q-gates.

A. Overall Resynthesis Flow

Our method begins by deferring all Clifford operations to the end of the circuit, yielding a unified Pauli rotation tableau followed by one consolidated Clifford tableau. The synthesis is then performed by our Pauli Rotation Synthesis algorithm, referred to as pMST. Inspired by the minimum spanning tree (MST) approach of [12], pMST extends it from diagonal parity tables to the general setting of Pauli rotation tableaux.

The core procedure is illustrated in Algorithm 1: starting from the unified tableau, the algorithm iteratively identifies executable rotations, selects the lowest-cost candidate, synthesizes it with an MST-guided routine, and updates the tableaux afterwards. Once all Pauli rotations have been processed, the residual Clifford tableau is synthesized at the end.

This overview summarizes the high-level flow of our method. The next section elaborates on the pMST algorithm, the core process of the unified synthesis approach.

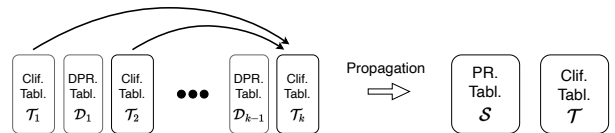


Fig. 3: Illustrative example of Clifford propagation.

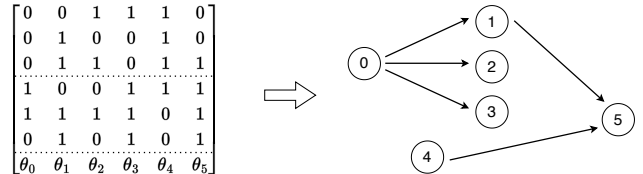


Fig. 4: Example of a dependency graph constructed from a Pauli rotation tableau.

B. The pMST Algorithm

We first explain how target rotations are iteratively selected using dependency constraints and cost estimation, then describe how each rotation is synthesized via an MST-based routine, and finally present the residual Clifford synthesis step.

1) *Target Rotation Selection:* Unlike the diagonalized Pauli rotation tableau, where all Pauli rotations commute and can be applied in an arbitrary order, a general Pauli rotation tableau may contain anticommuting pairs. These anticommutation relations impose precedence constraints, so that reordering certain rotations would change the resulting unitary operation. To capture and respect these constraints, we construct a dependency graph (DepG), where each node corresponds to a Pauli rotation and the directed edges represent the ordering constraints imposed by anticommutation.

Fig. 4 illustrates a simple example of the dependency graph. On the left is a Pauli rotation tableau with three qubits, where each column represents a Pauli rotation. On the right is the corresponding DepG, whose node indices are aligned with the tableau columns, with directed edges inserted whenever the corresponding rotations anticommute. In general, two Pauli rotations anticommute if they have different non- I operators (X , Y , or Z) on an odd number of qubits. The example shows that rotation 0 must be implemented before 1, 2, and 3, and that 4 must precede 5, highlighting how ordering constraints are made explicit.

Based on this graph, each iteration begins by identifying executable rotations (nodes with no unmet dependencies). An estimated cost, defined as the number of qubits on which the Pauli operator is X , Y , or Z , is then assigned to each candidate. Intuitively, the number of non- I qubits reflects how many qubits must be connected by 2Q-gates to realize the rotation, and thus correlates with the 2Q-gates required in its synthesis. This estimate allows the algorithm to prioritize rotations with smaller non- I qubits, which can typically be implemented with fewer 2Q-gates.

Consider the example in Fig. 4. Both rotations 0 and 4 have no incoming edges in the DepG and are therefore executable. The estimated cost of rotation 0 is 2, while that of rotation 4

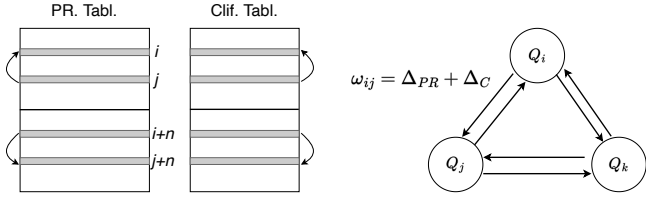


Fig. 5: Illustration of edge weight in the MST parity graph.

is 3. Following the selection strategy, the algorithm chooses rotation 0 as the next target to synthesize.

Once selected, we synthesize the rotation 0, remove it from the tableau, and then continue with the updated DepG.

2) *Synthesis of a Selected Rotation*: Once a target rotation is selected, we first *diagonalize* its Pauli string, i.e., map all X and Y operators into Z by Clifford conjugation. This is achieved by applying the appropriate single-qubit Clifford gates, for example:

$$X = HZH, \quad Y = SHZHS^\dagger.$$

We then append the Cliffords on the left of Z s to the synthesized circuit, and apply the Cliffords on the right of Z s to the Clifford tableau.

Once the selected rotation is diagonalized, we proceed to the MST synthesis routine, which constructs a cost-aware 2Q network to implement the rotation. Each qubit on which the operator is Z becomes a node in a weighted graph, and every possible 2Q-gate CX_{ij} is represented as an edge with an associated cost. From this graph, the MST selects the edges that minimize the impact on the tableaux, yielding a 2Q sequence that realizes the current rotation while lowering the cost of subsequent ones.

The edge weight is defined as

$$w(i, j) = \Delta_{PR}(i, j) + \Delta_C(i, j), \quad (5)$$

where $\Delta_{PR}(i, j)$ denotes the change in the total number of non-zero entries in the Pauli rotation tableau after applying CX_{ij} , while $\Delta_C(i, j)$ denotes the change in the distance between the Clifford tableau and the identity matrix. The resulting weight reflects whether applying CX_{ij} is beneficial overall: a negative value indicates that the operation reduces the combined cost, whereas a positive value implies an increase. This simple additive form serves as a heuristic that jointly accounts for immediate progress in the Pauli rotation tableau and the induced impact on the remaining Clifford tableau.

Figure 5 illustrates this correspondence: each edge (i, j) in the parity graph is associated with a candidate CX_{ij} , and its effect on the two tableaux yields Δ_{PR} and Δ_C , which together form the edge weight w_{ij} .

By associating each candidate CX with its joint impact on both tableaux, the MST construction considers the Pauli and Clifford parts in a unified manner.

3) *Residual Clifford Synthesis*: After all Pauli rotations have been synthesized, a residual Clifford tableau remains. We adapt the stabilizer tableau resynthesis procedure [7] to synthesize the residual Clifford tableau into a Clifford circuit. Together, these components constitute the unified synthesis framework.

TABLE I: Settings of synthesis and optimization methods.

Synthesis Approaches	
TKET	strat = Pairwise, cx_config = snake
Lazy	--rotation = MST
Basic/pMST	As described in this paper
Local Optimization Approaches	
Qiskit	CommutativeCancellation() →InverseCancellation()
TKET	RemoveRedundancies() → CliffordSimp() → RemoveRedundancies()
Qsyn	CausalFlowOpt()

V. EXPERIMENTAL RESULT

A. Experimental Settings

All experiments were conducted on a Linux server running Ubuntu 22.04.5 LTS. The hardware configuration consists of an Intel® Core™ i9-13900K processor and 128 GB of main memory.

Our unified resynthesis method was implemented in Qsyn [24], an open-source platform for quantum circuit optimization that integrates several modern QCO techniques. To provide a fair baseline in the unified setting, we also design a naive unified synthesis baseline, denoted as *Basic*. Similar to pMST, Basic defers all Clifford gates through Clifford propagation, but it synthesizes Pauli rotations only in a fixed tableau order. Each rotation is diagonalized and implemented with a locally optimized 2Q-gate sequence, without accounting for its impact on subsequent operations.

We evaluate four synthesis strategies: Lazy [11], a state-of-the-art tableau-by-tableau approach; TKET, a Pauli-graph-based method [14]; Basic, a naive unified baseline; and pMST, our proposed algorithm. Note that before resynthesis, we employ the state-of-the-art T -optimization technique, FastTODD [9], which is applied once to all synthesis strategies. Accordingly, all methods share the same T -count baseline (denoted as T-opt in Table II), and we compare the resulting circuits primarily in terms of their 2Q-counts.

We further consider three local optimization methods implemented in Qiskit [25], TKET, and Qsyn that can be applied after synthesis. The complete list of methods and their configurations is summarized in Table I.

For benchmarks, we adopt the standard Clifford+T circuits used in [11], where two large circuits in the original benchmarks [9], hwb11 and hwb12, are excluded due to the time out on FastTODD algorithm.

B. 2Q-Count Comparison

We compare four synthesis strategies: TKET, Basic, Lazy, and pMST. Table II reports the initial 2Q- T -counts, the shared post T -opt T -count, and, for each synthesis strategy, the resulting 2Q-count, its ratio to the original 2Q-count, and the runtime. This representation allows a straightforward comparison of the final 2Q-counts produced by different strategies after the full QCO flow. Overall, our proposed method pMST achieves the lowest GeoMean ratio of 1.28, compared to 1.93 for Lazy,

TABLE II: Comparison on 2Q-count.

Benchmark	Orig.		T-opt	TKET [14]			Basic			Lazy [11]			pMST		
	2Q-count	T-count	T-count	2Q-count	Ratio	Time (s)	2Q-count	Ratio	Time (s)	2Q-count	Ratio	Time (s)	2Q-count	Ratio	Time (s)
adder_8	409	399	171	1,387	3.39	0.10	564	1.38	0.03	610	1.49	0.34	302	0.74	0.06
barenco_tof_4	48	56	28	120	2.50	0.01	78	1.63	0.03	76	1.58	0.02	48	1.00	0.01
barenco_tof_5	72	84	40	205	2.85	0.01	128	1.78	0.03	131	1.82	0.01	86	1.19	0.05
barenco_tof_10	192	224	100	888	4.63	0.03	464	2.42	0.03	282	1.47	0.18	175	0.91	0.05
csla_mux_3	80	70	58	407	5.09	0.01	215	2.69	0.01	238	2.98	0.03	150	1.88	0.06
csum_mux_9	168	196	77	702	4.18	0.03	707	4.21	0.04	633	3.77	0.10	344	2.05	0.06
gf2 ⁶ _mult	221	252	113	960	4.34	0.04	688	3.11	0.06	477	2.16	0.11	442	2.00	0.17
gf2 ⁷ _mult	300	343	155	1,385	4.62	0.06	1,023	3.41	0.05	674	2.25	0.10	634	2.11	0.16
gf2 ⁸ _mult	405	448	205	2,384	5.89	0.10	1,476	3.64	0.06	1,033	2.55	0.34	907	2.24	0.27
gf2 ⁹ _mult	494	567	257	2,441	4.94	0.13	2,064	4.18	0.09	1,291	2.61	0.24	1,112	2.25	0.34
ham15-low	236	161	97	875	3.71	0.04	354	1.50	0.02	454	1.92	0.08	282	1.19	0.07
ham15-med	534	574	212	1,909	3.57	0.07	698	1.31	0.06	726	1.36	0.41	468	0.88	0.13
ham15-high	2,149	2,457	1,018	6,658	3.10	0.41	3,223	1.50	0.03	3,499	1.63	0.11	2,038	0.95	0.06
hwb6	116	105	75	349	3.01	0.01	181	1.56	0.01	152	1.31	0.04	118	1.02	0.04
hwb8	7,129	5,887	3,512	32,946	4.62	1.95	10,905	1.53	0.45	10,239	1.44	0.19	9,082	1.27	0.72
hwb10	35,170	29,939	15,876	210,853	6.00	14.77	61,236	1.74	10.44	61,501	1.75	1.70	55,943	1.59	17.20
mod_mult_55	48	49	34	169	3.52	0.01	80	1.67	0.01	117	2.44	0.01	73	1.52	0.01
mod_red_21	105	119	73	497	4.73	0.02	226	2.15	0.04	204	1.94	0.07	118	1.12	0.06
qcla_adder_10	233	238	161	1,548	6.64	0.07	668	2.87	0.07	586	2.52	0.33	279	1.20	0.14
qcla_com_7	186	203	95	544	2.92	0.03	427	2.30	0.05	324	1.74	0.07	177	0.95	0.04
qcla_mod_7	382	413	237	2,710	7.09	0.12	1,076	2.82	0.06	1,498	3.92	0.68	466	1.22	0.11
rc_adder_6	93	77	47	461	4.96	0.01	154	1.66	0.02	140	1.51	0.02	152	1.63	0.02
tof_5	42	49	31	192	4.57	0.03	94	2.24	0.01	71	1.69	0.01	51	1.21	0.02
tof_10	102	119	71	796	7.80	0.02	287	2.81	0.01	155	1.52	0.12	121	1.19	0.05
vbe_adder_3	70	70	24	178	2.54	0.01	83	1.19	0.02	97	1.39	0.01	48	0.69	0.02
Geomean					4.24			2.14			1.93			1.28	

Note: All synthesis approaches share the same T-count as post-T-opt.

2.14 for Basic, and 4.24 for TKET. The experimental results also support our claim that while tableau-based QCOs can effectively minimize the T -count, they often incur additional 2Q-count overhead.

The results also lead to three key observations. First, the performance gap between TKET and Basic highlights the advantage of tableau-based synthesis itself: the tableau representation enables efficient row operations for conjugating Clifford gates, which are not as directly supported in the Pauligraph formalism of TKET. Second, a naive unification does not necessarily yield better results. Although Basic is tableau-based and unified, it often performs worse than the tableau-by-tableau Lazy approach, indicating that naive integration without further guidance may fail to realize its potential benefits. Finally, the consistent improvement of pMST over Lazy confirms that combining unification with an effective synthesis strategy is essential to fully realize the potential of tableau-based resynthesis, thereby advancing the state of the art in reducing 2Q-count overhead.

C. Robustness under Local Optimization

To further validate the robustness of our approach, we examine whether the advantage of pMST persists after applying standard local optimization passes. This addresses the concern that our method might merely incorporate such optimizations.

We consider three representative optimizers: Qiskit, which applies commutation- and inverse-based gate cancellation; TKET, which employs redundancy removal with Clifford simplification; and Qsyn, which leverages causal-flow structure to reduce unnecessary 2Q-gates [26]. For each synthesis method, we report results under four settings: without optimization (Original) and with QiskitOpt, TKETOpt, or QsynOpt.

The results, summarized in Fig. 6, show that pMST consistently achieves the lowest Geomean 2Q-count ratio across all optimization backends. While all synthesis methods benefit from subsequent local simplifications, the relative ranking

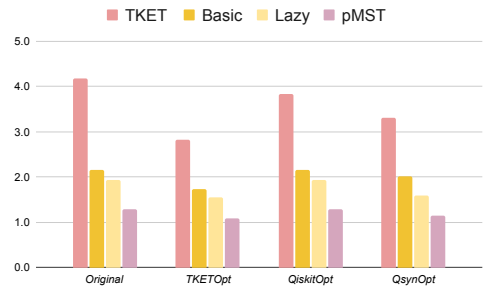


Fig. 6: 2Q-count ratios under different optimizers.

of the approaches remain unchanged: pMST yields the best performance, followed by Lazy, then Basic, and finally TKET. This outcome confirms that the superiority of pMST is not due to the exclusion of local optimization, but rather a fundamental improvement in the synthesis process itself, which continues to deliver the lowest 2Q-count even after standard post-processing.

VI. CONCLUSION

In this work, we presented a unified tableau synthesis framework that consolidates Clifford and Pauli rotation tableaux into a holistic resynthesis flow. Within this framework, we introduced the *pMST* algorithm, which extends the MST paradigm to general Pauli rotations for globally cost-aware synthesis. Experimental results on Clifford+T benchmarks show that our method achieves the lowest 2Q-count, outperforming state-of-the-art approaches even with additional local optimizations. These results confirm the effectiveness of unified synthesis and its potential for advancing quantum circuit optimization.

VII. ACKNOWLEDGMENT

This work is supported by the National Science and Technology Council, R.O.C., Project Nos.: NSTC 114-2119-M-002-020 and NSTC 114-2640-E-002-001.

REFERENCES

- [1] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *arXiv preprint quant-ph/0301141*, 2003.
- [2] C.-Y. Chen, G.-J. Zeng, F.-J. Lin, Y.-H. Chou, and H.-C. Chao, "Quantum cryptography and its applications over the internet," *IEEE Network*, vol. 29, no. 5, pp. 64–69, 2015.
- [3] D. Peral-García, J. Cruz-Benito, and F. J. García-Peñalvo, "Systematic literature review: Quantum machine learning and its applications," *Computer Science Review*, vol. 51, p. 100619, 2024.
- [4] S. Khatri and M. M. Wilde, "Principles of quantum communication theory: A modern approach," *arXiv preprint arXiv:2011.04672*, 2020.
- [5] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, "On the qubit routing problem," *arXiv preprint arXiv:1902.08091*, 2019.
- [6] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 86, no. 3, p. 032324, 2012.
- [7] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 70, no. 5, p. 052328, 2004.
- [8] L. E. Heyfron and E. T. Campbell, "An efficient quantum compiler that reduces t count," *Quantum Science and Technology*, vol. 4, no. 1, p. 015004, 2018.
- [9] V. Vandaele, "Lower t-count with faster algorithms," *arXiv preprint arXiv:2407.08695*, 2024.
- [10] F. J. Ruiz, T. Laakkonen, J. Bausch, M. Balog, M. Barekatin, F. J. Heras, A. Novikov, N. Fitzpatrick, B. Romera-Paredes, J. van de Wetering *et al.*, "Quantum circuit optimization with alphasensor," *Nature Machine Intelligence*, pp. 1–12, 2025.
- [11] M.-T. Lau, H.-C. Yang, H.-Y. Chen, and C.-Y. R. Huang, "A lazy resynthesis approach for simultaneous t gate and two-qubit gate optimization of quantum circuits," *arXiv preprint arXiv:2508.04092*, 2025.
- [12] V. Vandaele, S. Martiel, and T. G. de Brugière, "Phase polynomials synthesis algorithms for nisq architectures and beyond," *Quantum Science and Technology*, vol. 7, no. 4, p. 045027, 2022.
- [13] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, " $t|ket\rangle$: a retargetable compiler for nisq devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, 2020.
- [14] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah, "Phase gadget synthesis for shallow circuits," *arXiv preprint arXiv:1906.01734*, 2019.
- [15] F. Zhang and J. Chen, "Optimizing t gates in clifford+ t circuit as $\pi/4$ rotations around paulis," *arXiv preprint arXiv:1903.12456*, 2019.
- [16] M. Amy, "Formal methods in quantum circuit design," Ph.D. dissertation, University of Waterloo Ontario, Canada, 2019.
- [17] M. Amy, D. Maslov, and M. Mosca, "Polynomial-time t-depth optimization of clifford+ t circuits via matroid partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1476–1489, 2014.
- [18] V. Vandaele, S. Perdrix, and C. Vuillot, "Optimal number of parametrized rotations and hadamard gates in parametrized clifford circuits with non-repeated parameters," *arXiv preprint arXiv:2407.07846*, 2024.
- [19] V. Vandaele, S. Martiel, S. Perdrix, and C. Vuillot, "Optimal hadamard gate count for clifford+ t synthesis of pauli rotations sequences," *ACM Transactions on Quantum Computing*, vol. 5, no. 1, pp. 1–29, 2024.
- [20] S. Schneider, L. Burgholzer, and R. Wille, "A sat encoding for optimal clifford circuit synthesis," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 190–195.
- [21] D. Maslov and M. Roetteler, "Shorter stabilizer circuits via Bruhat decomposition and quantum circuit transformations," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 4729–4738, 2018.
- [22] M. Amy, P. Azimzadeh, and M. Mosca, "On the controlled-not complexity of controlled-not-phase circuits," *Quantum Science and Technology*, vol. 4, no. 1, p. 015002, 2018.
- [23] Z. Chen, H. Chen, Y. Jin, M. Guo, E. Jang, J. Li, C. Chan, W. W. Ro, and E. Z. Zhang, "Phasepoly: An optimization framework for phase polynomials in quantum circuits," *arXiv preprint arXiv:2506.20624*, 2025.
- [24] M.-T. Lau, C.-Y. Cheng, C.-H. Lu, C.-H. Chuang, Y.-H. Kuo, H.-C. Yang, C.-T. Kuo, H.-Y. Chen, C.-Y. Tung, C.-E. Tsai *et al.*, "Qsyn: A developer-friendly quantum circuit synthesis framework for nisq era and beyond," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2. IEEE, 2024, pp. 535–536.
- [25] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross *et al.*, "Quantum computing with qiskit," *arXiv preprint arXiv:2405.08810*, 2024.
- [26] C. Holker, "Causal flow preserving optimisation of quantum circuits in the zx-calculus," *arXiv preprint arXiv:2312.02793*, 2023.