# A Novel Approach to Quantum Circuit Partitioning

Joseph Clark[1], Himanshu Thapliyal[1] and Travis S. Humble[2]
[1] *University of Tennessee, Knoxville, TN, USA*; [2] *Oak Ridge National Laboratory, Oak Ridge, TN, USA*

***Abstract*—Quantum synthesis presents an effective method of circuit optimization, but scales exponentially with the number of qubits in the circuit. This problem can be addressed by partitioning the circuit into blocks with a limited number of qubits. Existing partitioning algorithms make large trade-offs to achieve either high speed or quality. We propose a method of circuit partitioning which is competitive with existing algorithms for both metrics. The proposed method is compared with two existing methods across common circuit architectures, matching an exhaustive solution in performance and a fast solution on time.**

## I. Introduction

The unitary matrix representation of a quantum operation is the basis of many quantum synthesis techniques [1] [2] [3]. These techniques may be used to optimize a quantum circuit by performing synthesis on the circuit unitary. However, because matrix size is exponential in the number of qubits on which the operation acts, the time and space required by these quantum synthesis techniques also scales exponentially with the number of qubits [4].

An alternative technique for circuit optimization has been proposed [5] which splits a circuit with $n$ qubits into partitions containing $k$ qubits, then uses a conventional synthesis tool on the partitions. This reduces the complexity from exponential in $n$ to exponential in $k$. A similar technique has also been used to synthesize approximate versions of a given circuit [6].

The optimality of the circuit partitioning algorithm used in these processes significantly affects run time and the quality of the final result. The ideal algorithm should partition the circuit into blocks containing no more than $k$ qubits, with each block maximizing some scoring function. Since multi-qubit gates are noisier than single-qubit gates on Noisy Intermediate-Scale Quantum (NISQ) systems, the chosen scoring function will maximize the number of multi-qubit gates per block. The circuit partitioning algorithm proposed in QGo [5] uses a greedy solution with exhaustive search to partition a circuit into blocks of at most $k$ qubits. This method was later incorporated into the Berkeley Quantum Synthesis Toolkit (BQSKit) [7] as ScanPartitioner, and is also used by QEst [6]. QuickPartitioner, also included in BQSKit, partitions a circuit by iterating over the gates in topological order and greedily grouping them into partitions. QuickPartitioner is faster than ScanPartitioner for most circuits, but produces partitions with fewer multi-qubit gates.

In this work we propose a method for partitioning quantum circuits which has a run time comparable to the faster partitioning algorithm while producing results comparable in quality to the exhaustive solution. The method is also compared with the other two partitioning algorithms by applying all three algorithms to a series of common test circuits, such as Quantum Fourier Transform or quantum adders and multipliers.

This paper is organized as follows: Section II describes the proposed partitioning method. Section III discusses how the methods were evaluated and results. Section IV concludes the article.

## II. Proposed Method

The proposed method is based on the ScanPartitioner method included in BQSKit, with several major modifications to improve time complexity. ScanPartitioner splits a circuit with $g$ gates and $n$ qubits into blocks with at most $k$ qubits, and is composed primarily of two steps. The first step calculates all possible sets of $k$ qubits which can interact on the target hardware. For a system in which all qubits can interact, this step results in approximately $n^k$ groups of qubits. The second step creates the largest possible group of gates acting on each group of qubits from the first step. Given that approximately $n^k$ groups are found in the first step, this step has a time complexity of $O(gn^k)$. The region with the largest number of multi-qubit gates is transformed into a partition, and removed from the active portion of the circuit. Step 2 is repeated until the active portion is empty. ScanPartitioner is therefore inefficient due to the method used to enumerate qubit interactions. The proposed method improves the time complexity of this step by generating qubit groups based on the topology of the circuit, rather than that of the target hardware.

The proposed qubit group enumeration algorithm takes the circuit to be partitioned and a map from each gate to the qubit dependencies of each gate as input, and returns the set of possible qubit groups. The map of qubit dependencies is calculated by traversing the circuit in topological order and accumulating the dependencies of each gate when the gate is encountered. The enumeration algorithm works by finding all possible combinations of connections along each individual qubit, making recursive calls to enumerate the available connections for the "other" qubit in each two-qubit gate. The recursive calls begin with the first gate for each qubit as the target and a set containing that qubit as the input set. The dependencies for the target gate are then added to each set of qubits in the input. The new set of qubit sets are passed to

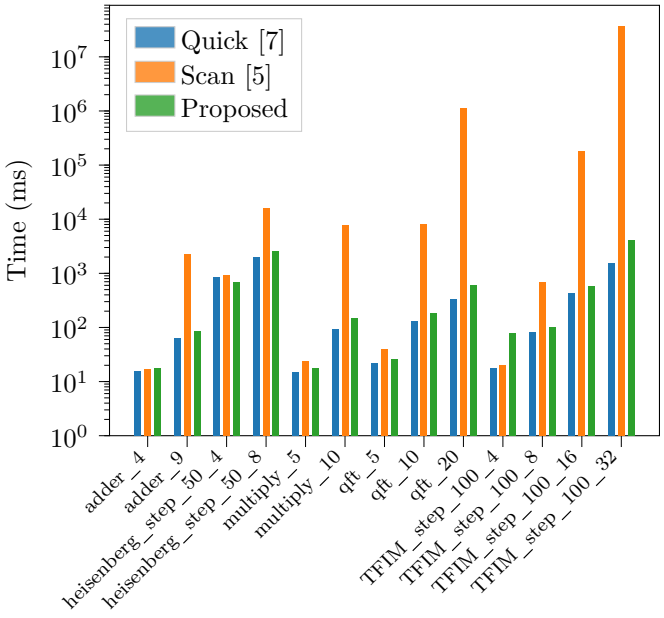


Fig. 1: Performance of Partitioning Methods Across a Variety of Circuits

another recursive call, this time with the other side of the gate as the target. In order to prevent unnecessary branching, input sets are only included in the output if the dependencies include new qubits. Values returned from the recursive call are merged into the return set, then passed as the input for a recursive call targeting the next gate which interacts with the target qubit. This process repeats until the end of the circuit is reached or the return set contains no partially complete sets. The return values from the original function call are merged and returned as the set of possible qubits. The remaining steps are identical to ScanPartitioner.

## III. Results

In order to compare the proposed method with the existing methods, a set of test circuits were partitioned using each method, and the run time and average number of gates per partition were measured for partitioner run. The test circuits include quantum adders, quantum multipliers, Quantum Fourier Transform (QFT) circuits, and transverse-field Ising Model (TFIM) circuits, among others. The circuits cover a variety of gate counts, qubit counts, and overall circuit structures. The analysis was performed with $k$ fixed at 4.

From Figure 1 it may be seen that the proposed method has a similar run time to QuickPartitioner across the tested circuits, and is faster than ScanPartitioner in all non-trivial cases. The difference in run time is especially noticeable for the 32 qubit TFIM circuit, wherein QuickPartitioner and the proposed method ran in 1.5 and 4.0 seconds, respectively, but ScanPartitioner took 10.4 hours.

In order to compare the quality of the result of each method, the average number of gates per block was measured for each partitioned circuit. The percentage difference in gates per block between the proposed method and the other methods for each circuit was calculated and averaged. Circuits with a qubit count of 4 were excluded from this part of the analysis, since all methods are guaranteed to create only one partition given $n \leq k$. The results show that circuits partitioned using the proposed method have an average of 1% fewer gates per block than those from ScanPartitioner, but 35% more than those from QuickPartitioner. Additionally, the proposed method performed the same as or better than QuickPartitioner for all circuits, and performed worse than ScanPartitioner only on the 16 and 32 qubit TFIM circuits.

## IV. Conclusion

Partition-based optimization is a promising method to address the exponential scaling of conventional synthesis-based optimization techniques. The performance of the partitioning algorithm used by this method significantly affects the quality of the resulting circuit. The proposed partitioning method addresses limitations of existing techniques, achieving speed comparable to faster partitioning methods and result quality comparable to exhaustive solutions. In a comparison of the performance of the proposed solution with one fast and one high-performing partitioning algorithm, the proposed method scales significantly better than the high-performing method on larger circuits, and shows results similar to the fast method. The quality of result produced by the proposed method are identical to the high-performing method in most cases, and show an average 34% improvement against the fast method. We also discuss the possibility of further improving the performance of the proposed method by reducing the number of repetitive circuit traversals.

## References

[1] B. Giles and P. Selinger, "Exact synthesis of multiqubit clifford+tcircuits," *Physical review. A, Atomic, molecular, and optical physics*, vol. 87, no. 3, 2013.

[2] M. G. Davis, E. Smith, A. Tudor, K. Sen, I. Siddiqi, and C. Iancu, "Towards optimal topology aware quantum circuit synthesis," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2020, pp. 223–234. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/QCE49297.2020.00036

[3] E. Smith, M. G. Davis, J. Larson, E. Younis, C. Iancu, and W. Lavrijsen, "Leap: Scaling numerical optimization based synthesis using an incremental approach," 2021. [Online]. Available: https://arxiv.org/abs/2106.11246

[4] O. Di Matteo and M. Mosca, "Parallelizing quantum circuit synthesis," *Quantum Science and Technology*, vol. 1, no. 1, p. 015003, 2016.

[5] X.-C. Wu, M. G. Davis, F. T. Chong, and C. Iancu, "Qgo: Scalable quantum circuit optimization using automated synthesis," 2020. [Online]. Available: https://arxiv.org/abs/2012.09835

[6] T. Patel, E. Younis, C. Iancu, W. de Jong, and D. Tiwari, "Robust and resource-efficient quantum circuit approximation," 2021. [Online]. Available: https://arxiv.org/abs/2108.12714

[7] B. N. Laboratory, "Bqskit," https://github.com/BQSKit/bqskit, 2022.