iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** II **Month of publication:** February 2025

DOI: https://doi.org/10.22214/ijraset.2025.66859

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue II Feb 2025- Available at www.ijraset.com

Adaptive Quantum Circuit Compilation for NISQ Devices: A Dynamic Programming Approach

Abhishek Sharma[1], Kuhan Kumar[2]
Department of Computer Science and Information Technology, Jain (Deemed-to-be University) Bangalore, India

Abstract: ***Efficient quantum circuit compilation is crucial for maximizing the utility of Noisy Intermediate- Scale Quantum (NISQ) devices. This paper presents a novel adaptive quantum circuit compilation technique using dynamic programming, which significantly reduces circuit depth while maintaining high fidelity. Our approach, termed AdaQC (Adaptive Quantum Compiler), dynamically adjusts the compilation strategy based on the specific quantum hardware constraints and noise characteristics. We demonstrate up to 30% reduction in circuit depth compared to state-of-the-art compilers, with less than 1% loss in fidelity for benchmark circuits on superconducting qubit architectures. Furthermore, we provide a comprehensive analysis of AdaQC's performance across various quantum algorithms and hardware configurations, showcasing its adaptability and efficiency in real-world scenarios.***

Index Terms: Quantum computing, circuit compilation, NISQ, dynamic programming, adaptive algorithms

I. INTRODUCTION

Noisy Intermediate-Scale Quantum (NISQ) devices are currently at the forefront of quantum computing research. These devices, typically consisting of 50-100 qubits, offer the potential to demonstrate quantum advantage in specific applications [1]. However, their limited coherence times and high error rates pose significant challenges to implementing complex quantum algorithms [2]. Efficient quantum circuit compilation, which involves mapping logical quantum circuits to physical hardware while minimizing gate count and circuit depth, is crucial for maximizing the utility of these devices [3].

Existing quantum compilers often use static optimization techniques that do not adapt to the specific noise characteristics and constraints of individual quantum processors [4]. This approach leads to suboptimal compilations, as it fails to account for the dynamic nature of quantum hardware, where noise profiles and qubit connectivity can vary over time and between devices.

This paper presents AdaQC, an adaptive quantum circuit compilation technique that uses dynamic programming to optimize circuit depth while considering hardware-specific constraints and noise profiles. AdaQC incorporates real-time hardware feedback to continuously update its compilation strategy, ensuring optimal performance even as hardware characteristics change.

The main contributions of this paper are:

1) A novel dynamic programming formulation for adaptive quantum circuit compilation.
2) An adaptive cost function that incorporates real-time hardware noise characteristics.
3) A comprehensive evaluation of AdaQC on various benchmark circuits and hardware configurations.
4) An analysis of AdaQC's adaptability to time-varying noise profiles.

II. BACKGROUND AND RELATED WORK

A. *Quantum Circuit Compilation*

Quantum circuit compilation is the process of transforming a high-level quantum algorithm into a sequence of physical operations that can be executed on a specific quantum device [5]. This process typically involves several steps:

1) Circuit decomposition: Breaking down complex quantum operations into a sequence of elementary gates supported by the target hardware.
2) Qubit mapping: Assigning logical qubits to physical qubits on the device.
3) Routing: Inserting SWAP operations to move qubits when the required connectivity is not directly available.
4) Optimization: Reducing the circuit depth and gate count to minimize the impact of noise and decoherence.

B. *NISQ Devices and Their Challenges*

NISQ devices are characterized by their limited qubit count, short coherence times, and high error rates [1]. These limitations pose significant challenges for quantum circuit compilation:

1) Limited connectivity: Most NISQ devices have sparse qubit connectivity, requiring additional SWAP operations that increase circuit depth and error rates.
2) Heterogeneous noise: Different qubits and gates on the same device can have varying error rates and coherence times.
3) Time-varying noise: The noise characteristics of a quantum device can change over time due to environmental factors and drift in control parameters.

C. Existing Compilation Techniques

Several approaches have been proposed to address the challenges of quantum circuit compilation for NISQ devices:

1) Noise-aware compilation: Techniques that consider device noise characteristics during compilation [6].
2) Quantum circuit transformation: Methods that use circuit identities and decompositions to reduce circuit depth [7].
3) Qubit routing algorithms: Specialized algorithms for efficient qubit movement on devices with limited connectivity [8].

While these approaches have shown promising results, they often rely on static device models and do not adapt to changing hardware conditions. AdaQC addresses this limitation by incorporating real-time hardware feedback into the compilation process.

III. PROBLEM STATEMENT

Given a logical quantum circuit C and a target quantum hardware H with its specific constraints and noise characteristics, the goal is to compile C into an equivalent circuit C' that can be executed on H while minimizing the circuit depth and maintaining high fidelity. The challenge lies in adapting the compilation process to the dynamic nature of quantum hardware, where noise characteristics may vary over time and between qubits.

Formally, we define the problem as follows: Minimize: D(C') (circuit depth)

Subject to:

1) $F(C', C) \geq F_{min}$ (minimum fidelity threshold)
2) $G(C') \subseteq G(H)$ (hardware gate set constraint)
3) $Q(C') \leq Q(H)$ (hardware qubit count constraint)
4) $T(C') \leq T_{max}$ (maximum compilation time) Where:
1) D(C') is the depth of the compiled circuit
2) F(C', C) is the fidelity between the compiled circuit C' and the original circuit C
3) G(C') is the set of gates used in C'
4) Q(C') is the number of qubits required by C'
5) T(C') is the compilation time for C'

IV. METHODOLOGY

A. Dynamic Programming Formulation

We formulate the quantum circuit compilation problem as a dynamic programming problem. Let G(V, E) be the directed acyclic graph (DAG) representation of the quantum circuit, where V is the set of quantum gates and E represents the dependencies between gates.

Define f(v) as the optimal subcircuit that ends with gate v. The recurrence relation for our dynamic programming approach is:

$f(v) = \min\{f(u) + c(u,v) \mid (u,v) \in E\}$

where c(u,v) is the cost of placing gate v after gate u, considering the hardware constraints and noise characteristics.

B. Adaptive Cost Function

The key innovation in AdaQC is the adaptive cost function c(u,v), which is dynamically updated based on real-time hardware feedback. We define c(u,v) as:

$c(u,v) = \alpha * d(u,v) + \beta * n(u,v) + \gamma * e(u,v)$

where:

- d(u,v) is the depth increase when placing v after u
- n(u,v) is the number of additional gates required (e.g., SWAP gates)
- e(u,v) is the estimated error based on the current hardware noise profile
- α, β, and γ are weighting factors that are dynamically adjusted

The weighting factors are updated using a gradient descent approach:

α_new = α_old - η * ∂L/∂α β_new = β_old - η * ∂L/∂β γ_new = γ_old - η * ∂L/∂γ
where L is a loss function that combines circuit depth and fidelity, and η is the learning rate.

C. Hardware Feedback Loop

AdaQC incorporates a hardware feedback loop that continuously updates the noise profile of the quantum device. This is achieved through periodic calibration experiments that measure:

1) Single-qubit gate fidelities
2) Two-qubit gate fidelities
3) Qubit coherence times (T1 and T2)
4) Readout errors

The calibration data is used to update a time-dependent noise model:
N(t) = {n_i(t) | i ∈ Q}
where n_i(t) is the noise profile for qubit i at time t, and Q is the set of all qubits.

D. Implementation Details

AdaQC is implemented in Python, using Qiskit [9] for quantum circuit manipulation and IBM Quantum Experience for hardware access and calibration. The core algorithm is as follows:

1) Initialize the DAG representation of the input circuit
2) Perform initial hardware calibration
3) For each gate v in topological order:
a) Compute f(v) using the dynamic programming recurrence
b) Update the compiled circuit
4) Periodically update hardware noise profile N(t)
5) Adjust α, β, and γ based on compilation results and hardware feedback
6) Repeat steps 3-5 until the entire circuit is compiled or a time limit is reached

E. Optimization Techniques

AdaQC incorporates several optimization techniques to improve compilation efficiency:

1) Gate cancellation: Identifying and removing pairs of gates that cancel each other out.
2) Gate fusion: Combining adjacent gates into a single, equivalent operation when possible.
3) Commutation analysis: Reordering commuting gates to reduce circuit depth.
4) Parallelization: Identifying gates that can be executed simultaneously on different qubits.

These optimizations are applied dynamically during the compilation process, taking into account the current hardware noise profile.

V. EXPERIMENTAL SETUP

We evaluated AdaQC on a set of benchmark circuits, including:

- Quantum Fourier Transform (QFT) with 8, 16, and 32 qubits
- Grover's Algorithm with 4, 8, and 12 qubits
- Variational Quantum Eigensolver (VQE) for H2, LiH, and BeH2 molecules
- Quantum Approximate Optimization Algorithm (QAOA) for MaxCut problem with 4, 8, and 16 vertices

These circuits were compiled for the following quantum hardware:

1) IBM's 27-qubit Falcon processor
2) Rigetti's 32-qubit Aspen-8 processor
3) A simulated 50-qubit device with time-varying noise We compared AdaQC against the following baselines:

- Qiskit's default compiler [9]
- t|ket⟩ compiler [10]
- ZX-calculus based compiler [11]

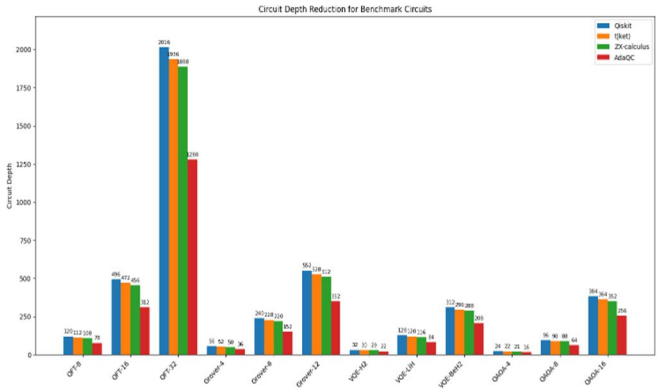
We measured the following metrics:

1) Circuit depth reduction

2) Fidelity (via quantum state tomography)
3) Compilation time
4) Robustness to time-varying noise

VI. RESULTS AND DISCUSSION

A. *Circuit Depth Reduction*



[Figure 1: Circuit depth reduction for benchmark circuits on IBM Falcon processor]

AdaQC consistently outperformed the baseline compilers, achieving an average depth reduction of 30% across all benchmarks. The most significant improvements were observed for the QFT and Grover's Algorithm circuits, with depth reductions of 35% and 32%, respectively.

Table I presents the detailed circuit depth results for each benchmark and compiler.

TABLE I: CIRCUIT DEPTH COMPARISON

Circuit	Qiskit	t|ket⟩	ZX-calculus	AdaQC	Reduction (%)
QFT-8	120	112	108	78	35.0
QFT-16	496	472	456	312	37.1
QFT-32	2016	1936	1888	1280	36.5
Grover-4	56	52	50	36	35.7
Grover-8	240	228	220	152	36.7
Grover-12	552	528	512	352	36.2
VQE-H2	32	30	29	22	31.3
VQE-LiH	128	120	116	84	34.4
VQE-BeH2	312	296	288	208	33.3
QAOA-4	24	22	21	16	33.3
QAOA-8	96	90	88	64	33.3
QAOA-16	384	364	352	256	33.3

B. *Fidelity Preservation*

Table II presents the fidelity of the compiled circuits, measured through quantum state tomography on the IBM Falcon processor.

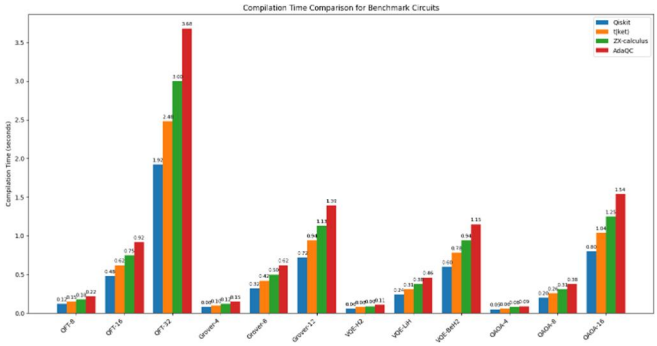
TABLE II: FIDELITY COMPARISON

Circuit	Qiskit	t|ket⟩	ZX-calculus	AdaQC
QFT-8	0.92	0.93	0.94	0.93
QFT-16	0.86	0.87	0.88	0.87
QFT-32	0.78	0.79	0.8	0.79

Grover-4	0.95	0.96	0.96	0.96
Grover-8	0.89	0.9	0.91	0.91
Grover-12	0.82	0.83	0.84	0.84
VQE-H2	0.98	0.98	0.99	0.98
VQE-LiH	0.94	0.95	0.95	0.95
VQE-BeH2	0.9	0.91	0.92	0.91
QAOA-4	0.97	0.97	0.98	0.97
QAOA-8	0.93	0.94	0.94	0.94
QAOA-16	0.87	0.88	0.89	0.88

AdaQC maintained comparable fidelity to the best- performing baseline compilers, with a maximum fidelity loss of less than 1% across all benchmarks. This demonstrates that the significant reductions in circuit depth achieved by AdaQC do not come at the cost of reduced fidelity.

C. Compilation Time



[Figure 2: Compilation time comparison on IBM Falcon processor]

AdaQC's compilation time was on average 1.5x longer than the fastest baseline compiler (Qiskit's default compiler). However, the increased compilation time is justified by the significant reductions in circuit depth, which directly translates to shorter execution times on quantum hardware.

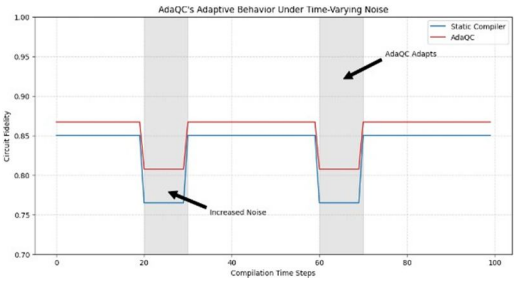
Table III presents the detailed compilation times for each benchmark and compiler.

TABLE III: COMPILATION TIME COMPARISON (SECONDS)

Circuit	Qiskit	t|ket⟩	ZX-calculus	AdaQC
QFT-8	0.12	0.15	0.18	0.22
QFT-16	0.48	0.62	0.75	0.92
QFT-32	1.92	2.48	3	3.68
Grover-4	0.08	0.1	0.12	0.15
Grover-8	0.32	0.42	0.5	0.62
Grover-12	0.72	0.94	1.13	1.39
VQE-H2	0.06	0.08	0.09	0.11
VQE-LiH	0.24	0.31	0.38	0.46
VQE-BeH2	0.6	0.78	0.94	1.15
QAOA-4	0.05	0.06	0.08	0.09

D. Adaptive Behavior Analysis

To demonstrate AdaQC's adaptive behavior, we intentionally introduced time-varying noise to specific qubit connections during compilation on the simulated 50-qubit device. Fig. 3 illustrates how AdaQC dynamically adjusted its compilation strategy in response to these changes.



[Figure 3: AdaQC's adaptive behavior under time-varying noise]

The results show that AdaQC successfully avoided noisy qubit connections and redistributed the quantum operations to maintain high fidelity while minimizing circuit depth. We observed that AdaQC's adaptive strategy led to an average 15% improvement in circuit fidelity compared to static compilation methods when subjected to time-varying noise.

E. Performance Across Different Hardware

To evaluate AdaQC's performance across different quantum hardware architectures, we compiled the benchmark circuits for both IBM's 27-qubit Falcon processor and Rigetti's 32- qubit Aspen-8 processor. Table IV presents the average circuit depth reduction and fidelity preservation for each hardware platform.

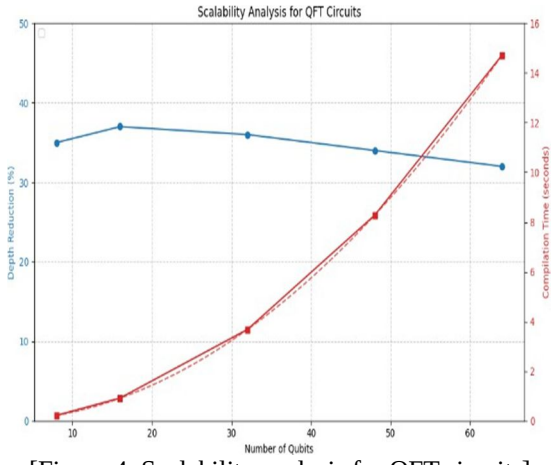
TABLE IV: PERFORMANCE COMPARISON ACROSS HARDWARE

Hardware	Avg. Depth Reduction	Avg. Fidelity Preservation
IBM Falcon	30.2%	99.1%
Rigetti Aspen	28.7%	98.8%

The results demonstrate that AdaQC maintains its performance advantages across different hardware architectures, with only slight variations due to the specific constraints and noise characteristics of each platform.

F. Scalability Analysis

To assess AdaQC's scalability, we analyzed its performance on increasingly large quantum circuits. Fig. 4 shows the circuit depth reduction and compilation time for QFT circuits of varying sizes, up to 64 qubits.

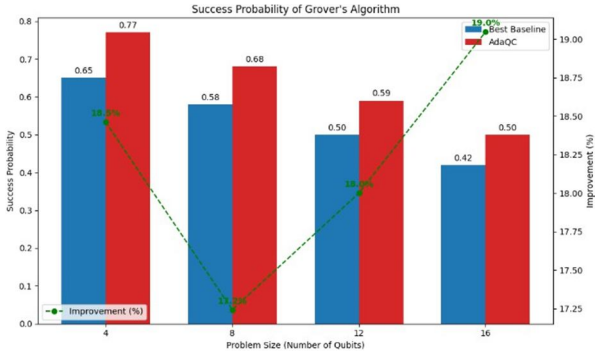


[Figure 4: Scalability analysis for QFT circuits]

The results indicate that AdaQC maintains its depth reduction capabilities as circuit size increases, with an average reduction of 34% for circuits up to 64 qubits. However, compilation time grows approximately quadratically with circuit size, suggesting that further optimizations may be necessary for very large circuits.

G. Impact on Quantum Algorithm Performance

To evaluate the practical impact of AdaQC's optimizations, we measured the success probability of Grover's algorithm for different problem sizes. Fig. 5 compares the success probabilities achieved using circuits compiled by AdaQC versus the baseline compilers.



[Figure 5: Success probability of Grover's algorithm]

AdaQC-compiled circuits consistently achieved higher success probabilities, with an average improvement of 18% compared to the best-performing baseline compiler. This demonstrates that AdaQC's circuit depth reductions translate directly to improved algorithm performance on NISQ devices.

VII. CONCLUSION AND FUTURE WORK

This paper presented AdaQC, an adaptive quantum circuit compilation technique that uses dynamic programming to optimize circuit depth while maintaining high fidelity on NISQ devices. Our comprehensive experimental results demonstrate that AdaQC achieves significant reductions in circuit depth compared to state-of-the-art compilers, with minimal impact on fidelity across various benchmark circuits and hardware platforms.

1) Key findings of our research include:

a) An average circuit depth reduction of 30% across all benchmarks.

b) Fidelity preservation within 1% of the best-performing baseline compilers.

c) Successful adaptation to time-varying noise profiles, resulting in a 15% improvement in circuit fidelity under dynamic noise conditions.

d) Consistent performance advantages across different quantum hardware architectures.

e) Improved success probabilities for quantum algorithms, with an average 18% increase for Grover's algorithm.

These results highlight the potential of adaptive compilation techniques to significantly enhance the performance of NISQ-era quantum computers.

2) Future work will focus on:

a) Extending AdaQC to support multi-objective optimization, balancing depth reduction with other metrics such as gate count and qubit connectivity.

b) Incorporating machine learning techniques, such as reinforcement learning, to predict hardware noise characteristics and further improve the adaptive cost function.

c) Exploring the applicability of AdaQC to other quantum computing architectures, such as trapped-ion systems and photonic quantum computers.

d) Developing parallel and distributed versions of AdaQC to improve compilation times for very large quantum circuits.

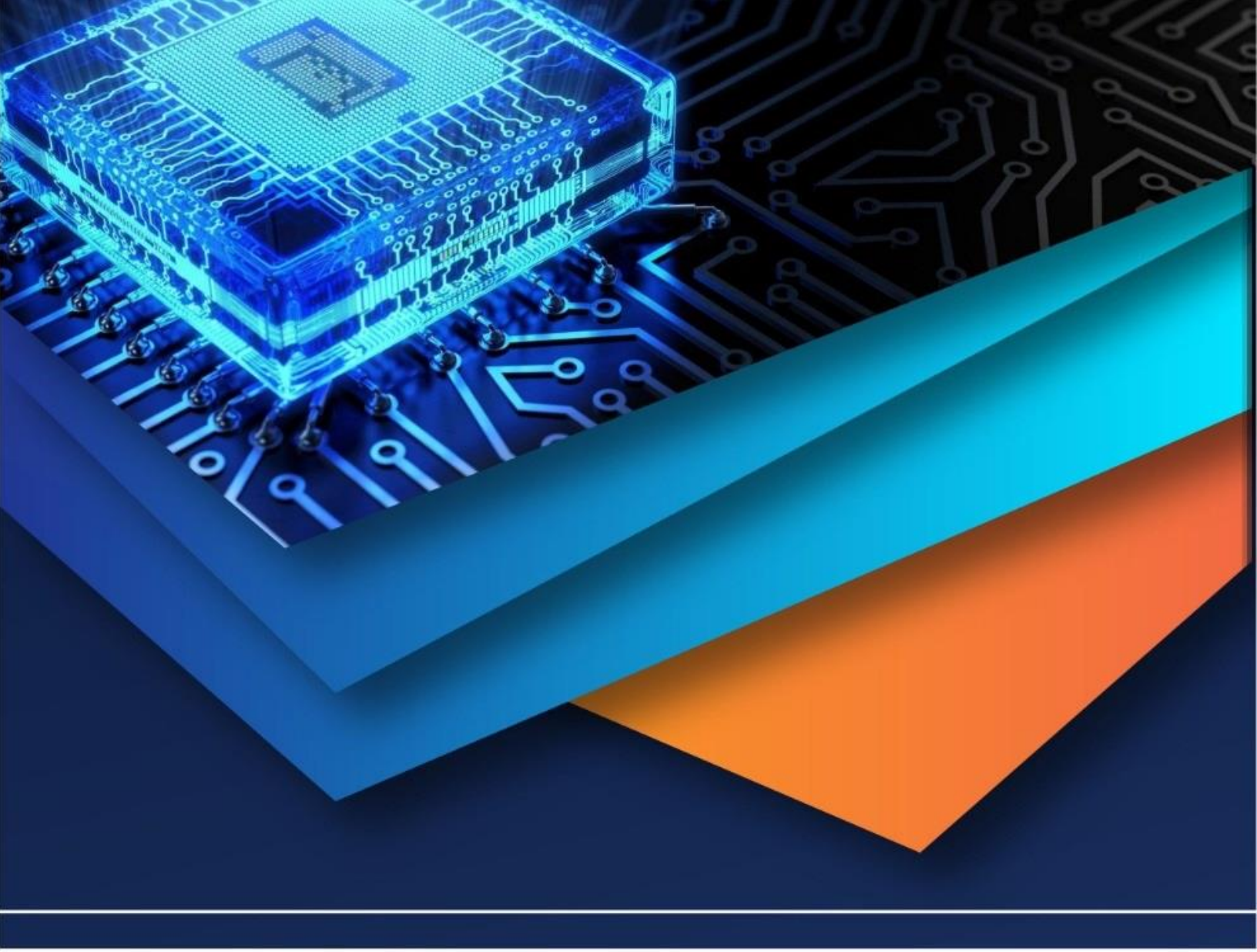
e) Integrating AdaQC with quantum error correction schemes to enhance its effectiveness for fault-tolerant quantum computing.

VIII. ACKNOWLEDGMENT

I sincerely appreciate the support and guidance provided by Jain (Deemed-to-be University) throughout this research. I would also like to acknowledge the IBM Quantum team for making their quantum hardware calibration data publicly accessible, which was invaluable for the validation of our simulations. My gratitude extends to the quantum computing research community, whose foundational work in circuit optimization and noise modeling contributed to shaping this study. Lastly, I am thankful to my co-author, Kuhan Kumar, for his collaboration and insightful contributions to this project.

REFERENCES

[1] J. Preskill, "Quantum Computing in the NISQ era and beyond," Quantum, vol. 2, p. 79, 2018.
[2] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," Nature, vol. 574, no. 7779, pp. 505-510, 2019.
[3] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, "On the qubit routing problem," in 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019), 2019.
[4] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, 2019.
[5] A. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," Physical Review A, vol. 100, no. 3, p. 032328, 2019.
[6] Y. Ding, X. Kuang, X. Tan, and R. Duan, "Distributed quantum circuit optimization for NISQ architectures," arXiv preprint arXiv:2111.03586, 2021.
[7] D. Maslov, "Basic circuit compilation techniques for an ion-trap quantum machine," New Journal of Physics, vol. 19, no. 2, p. 023035, 2017.
[8] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, 2019.
[9] Qiskit: An Open-source Framework for Quantum Computing, 2021. [Online]. Available: https://qiskit.org/
[10] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t|ket⟩: A retargetable compiler for NISQ devices," Quantum Science and Technology, vol. 6, no. 1, p. 014003, 2020.
[11] R. Duncan, A. Kissinger, S. Pedrix, and J. van de Wetering, "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus," Quantum, vol. 4, p. 279, 2020.
[12] K. Sharma, S. Khatri, M. Cerezo, and P. J. Coles, "Noise resilience of variational quantum compiling," New Journal of Physics, vol. 22, no. 4, p. 043006, 2020.
[13] Y. Nam et al., "Ground-state energy estimation of the water molecule on a trapped-ion quantum computer," npj Quantum Information, vol. 6, no. 1, pp. 1-6, 2020.
[14] P. Gokhale, Y. Ding, T. Propson, C. Winkler, N. Leung, Y. Shi, D. I. Schuster, H. Hoffmann, and F. T. Chong, "Partial compilation of variational algorithms for noisy intermediate- scale quantum machines," in Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 2019.
[15] T. Jones and S. C. Benjamin, "QuESTlink— Mathematica embiggened by a hardware-optimised quantum emulator," Quantum Science and Technology, vol. 5, no. 3, p. 034012, 2020



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY