QuBEC: Boosting Equivalence Checking for Quantum Circuits With QEC Embedding

Chao Lu, *Student Member, IEEE*, Navnil Choudhury, *Student Member, IEEE*, Utsav Banerjee, *Member, IEEE*, Abdullah Ash Saki, *Member, IEEE*, and Kanad Basu, *Member, IEEE*

***Abstract*—Quantum computing has proven to be capable of accelerating many algorithms by performing tasks that classical computers cannot. As quantum algorithms and implementations grow more complex, the need for rigorous circuit verification becomes critical, ensuring correct compilation and enhancing circuit fidelity through error correction and assertions. In this article, we propose QuBEC, a decision diagram-based quantum equivalence checking approach, that requires less latency compared to existing techniques, while accounting for circuits with quantum error correction redundancy. QuBEC reduces verification time on benchmark circuits by up to 443×, while the number of decision diagram nodes required is reduced by up to 798.31×, compared to state-of-the-art strategies. The proposed QuBEC framework can contribute to the advancement of quantum computing by enabling faster and more efficient verification of quantum circuits, paving the way for the development of larger and more complex quantum algorithms.**

***Index Terms*—Quantum circuit equivalence checking, quantum error correction (QEC), quantum verification.**

I. Introduction

QUANTUM computing is poised to bring the next tectonic shift in computing. By harnessing the unique properties of quantum mechanics, such as superposition, entanglement, and interference, quantum computers can speed up a certain class of problems in fields like chemistry [1], optimization [2], and machine learning [3], beyond the abilities of most high performance classical computers. To obtain the best performance of quantum computers for solving humanity's most demanding problems, different stacks in the computing paradigm must work in unison and in an optimal fashion. It ranges from designing novel high-level algorithms to developing powerful low-level hardware. Besides algorithmic and hardware efficiencies, equally important, is a highly proficient compilation stack, and in the realities of today's NISQ hardware, the need for a performant compilation tool-chain is highly pronounced. This brings up a critical question for consideration: *"How do we verify that the compiled version of a circuit is logically equivalent to the input circuit?"*

While the compiler is expected to generate the most optimized version of the output circuit, we need it to preserve the logical integrity of the input circuit. Thus, the idea of *quantum verification* becomes critical during the compilation process. Quantum verification is a systematic approach to compare the input circuit and its compiled version to certify both circuits are logically equivalent. Numerous recent methodologies in the field of quantum verification have been developed to tackle specific challenges, and lay out a blueprint for the future [4], [5], [6], [7]. However, none of the aforementioned works delves into QEC verification methodologies. Our approaches facilitate accelerated execution speeds for equivalence checking, even in scenarios involving large quantum circuits. However, previous strategies cannot verify highly intricate benchmark circuits in less than the 60 s threshold. Therefore, in this article, we propose algorithmic optimizations to make the verification process more scalable, which not only can handle circuits with larger numbers of qubits but also accomplish the verification task faster. Moreover, we extend our verification algorithm to be applicable for quantum error correction (QEC) as well.

The addition of ancillary qubit creates a mismatch in the number of qubits and gates between the original circuit and the error-corrected circuit, and thus, makes the verification process even more challenging. To alleviate the challenge, we propose a two-step methodology for verifying quantum circuits with QEC code (QECC) encoded into them. The first step employs an intelligent circuit pruning pass to extract the functioning information of the quantum circuit. Next, the second step applies our scalable verification flow on the pruned circuit to verify the logical equivalence. To the best of our knowledge, our work is the first attempt to verify the functionality of quantum circuits with embedded QECC. To this end, we make the following contributions in this article.

1) We propose a Position Match verification strategy based on decision diagrams, that improves the state-of-the-art verification methodology.
2) We propose a methodology that enables the verification scheme to check the original functionality of a quantum circuit without QEC.
3) We evaluate our equivalence checking strategy by comparing it with existing verification approaches. Our

Manuscript received 7 August 2023; revised 18 November 2023; accepted 9 January 2024. Date of publication 2 February 2024; date of current version 20 June 2024. This work was supported by NSF under Grant 2228725. This article was recommended by Associate Editor R. Wille. *(Corresponding author: Chao Lu.)*

Chao Lu, Navnil Choudhury, and Kanad Basu are with the Department of Electrical and Computer Engineering, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: cxl200053@utdallas.edu; nxc210017@utdallas.edu; kxb190012@utdallas.edu).

Utsav Banerjee is with the Department of Electronic Systems Engineering, Indian Institute of Science, Bengaluru 560012, India (e-mail: utsav@iisc.ac.in).

Abdullah Ash Saki is with IBM Research, New York, NY 10598 USA (e-mail: axs1251@psu.edu).

Digital Object Identifier 10.1109/TCAD.2024.3361402

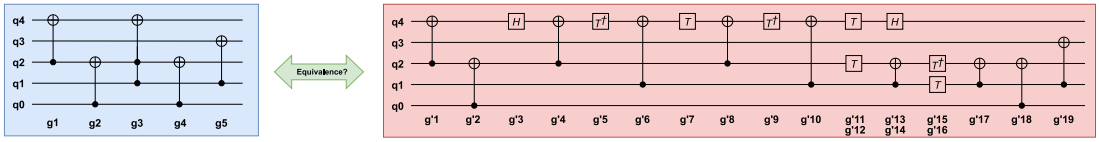

Fig. 1. Example equivalence checking of two quantum circuits.

proposed QuBEC is seen to provide enhancement up to 443× for time consumption, and up to 798.31× in Decision-Diagram nodes consumption.

II. Background

To ensure the correctness of a quantum circuit, the inverse of the quantum circuit can be appended to the original circuit. This is due to the invertible property of a quantum gate, which will generate an identity gate as each gate cancels with its inverse. However, quantum gates can cancel with each other only as long as the order of the gates is maintained. Hence, an efficient verification flow is required to verify quantum circuits so that the complex intermediate states can be avoided, by arranging the quantum circuit verification with each quantum circuit's gate order unchanged. The complexity and resource consumption of a DD-based quantum circuit simulation heavily depends on the state of qubits. Therefore, an optimized strategy for quantum circuit verification is required to address resource consumption, and improve verification efficiency on a classical computer [4].

QEC is a critical part of the design process for quantum computing, as it helps ensure that the computations are carried out with the required level of fidelity. As part of the automation goals of quantum circuit synthesis, software is used to analyze quantum circuits and add QEC circuits to the original design in order to control quantum noise. However, current quantum circuit equivalence checking techniques are unable to verify circuits that contain redundancy with ancilla qubits and duplication, as this significantly alters the qubit and circuit function. As a result, it is crucial to find verification methods that can confirm the equivalence of quantum circuits, even when one of them contains functional redundancy. To the best of our knowledge, there are no quantum circuit verification approaches with the QEC redundancy.

III. Proposed QuBEC

In this article, we propose QuBEC, a boosted quantum circuit equivalence checking scheme with QEC embedding. In this section, we initiate the discussion by examining the intricate details of our proposed Position Match strategy. This is motivated by the versatility inherent in our strategy, enabling its applicability to circuits both with and without QEC. Subsequently, we proceed to outline the implementation of our strategy, specifically taking into account the inclusion of QEC embedding.

The proposed approach stores the positions of previously verified gates, and uses a greedy method to determine the optimal path for the upcoming gates by utilizing this information. For the remainder of this section, we will use the following terminologies: the *upcoming gate* refers to the gate that is awaiting verification, while the *previous gate* is the gate that has just been verified in the last step. For example, in Fig. 1, if gates g_1 and g'_1 are already verified, they are considered *previous gates*, while gates g_2 and g'_2 are considered the *upcoming gates*. *Processing* a gate refers to the operation that the program will take the matrix of the quantum gate and append to the quantum circuit. When two gates of two different quantum circuits are deemed equivalent, we refer to them as *canceled*. The following sections explain our proposed QuBEC framework.

A. Proposed Position Match Approach

The proposed Position Match approach is applied to perform equivalence checking between two quantum circuits. To this end, it requires checking the positions of unverified quantum gates on both quantum circuits. To facilitate this, we introduce an active qubit pool to store the current active qubits, which are qubits that the previously verified quantum gates used. This increases the likelihood of cancelation between gates and ensures that the quantum gates executed on inactive qubits are not processed before the ones that are already in the active pool. The size of the active qubit pool can also affect the equivalence checking flow. If the capacity is too small, complex quantum gates such as the multiple-controlled-X (MCX) gate, which contains a high number of qubits, might exceed the pool's maximum size. As a result, the MCX gate might never be checked for equivalence, or some qubits' information of the MCX gate will be lost. Conversely, if the active qubit pool is too large, several quantum gates that are not in the same position might be processed together, reducing verification efficiency and increasing the number of intermediate nodes.

Algorithm 1 presents our proposed Position Match approach to accelerate the quantum circuit equivalence checking process. The algorithm takes as input two quantum circuits, *circ1* and *circ2*, that need to be verified, and applies several optimization methodologies, including "fuseSingleQubitGate," "reconstructSWAPs," "removeDiagonalGatesBeforeMeasure," and "reorderOperations" from the MQT/QCEC library [4]. Once the optimization is complete, the proposed Position Match algorithm is executed. The equivalence checking starts by processing the first gate of *circ1* and adding the qubits used in the gate to the active qubit pool, with the maximum size set to the size of the input gate (lines 1–3). For the while loop in the algorithm, the verification has two options to proceed: the second quantum gate in *circ1* or the first inverse gate in *circ2* (lines 4–7). The algorithm checks whether the qubits required for both gates are in the active qubit pool. If one gate is in the pool and the other is not, the algorithm

Algorithm 1 Position Match Strategy

Input: Two quantum circuits that is pending to be verified (circ1, circ2).
Output: Boolean: Whether two quantum circuits are equivalent.

```
1: pool = set with maximum number of(max (size(qubits required for the first gate in circ1), size(qubits required for the first gate in circ2)));
2: process one gate of circ1;
3: pool.add(qubits required for the first gate in circ1)
4: while remaining gates on both circuits to proceed do
5:     set1 = qubits required for next gate in circ1;
6:     set2 = qubits required for next gate in circ2;
7:     maxSize = max(length(set1), length(set2));
8:     if (set1 in pool) & not(set2 in pool) then
9:         Proceed with next gate in circ1;
10:    else if not(set1 in pool) & (set2 in pool) then
11:        proceed with next gate in circ2;
12:    else
13:        Compare the nodes consumption with circ1, circ2 and select the best path with least nodes consumption;
14:        if pool is not full then
15:            pool.add(current gate's corresponding qubits)
16:        else
17:            pool = set(max size(qubits required for current gate in circ1), size(qubits required for current gate in circ2));
18:            pool.add(current gate's corresponding qubits)
19:        end if
20:    end if
21: end while
22: while One circuit finishes the equivalence checking do
23:    Proceed the remaining gates;
24: end while
25: if the resultant DD is identity DD then
26:    return True;
27: else
28:    return False;
29: end if
```

prioritizes the gate that is in the active qubit pool (lines 8 and 9). If both gates are not in the pool, the algorithm performs a greedy node check for both paths and selects the path that requires fewer DD nodes using the required algorithm (lines 10–13). After the path is selected, the active qubit pool is updated with the new qubits from the gates that were just verified (lines 14 and 15). If the maximum size of the active qubit pool is reached, the algorithm removes the previous pool and creates a new pool with a maximum size equivalent to the highest number of qubits required among the gates that are pending verification (lines 16–19). The new qubits of the previous gate are added to the active qubit pool. The process is repeated until all the quantum gates in one quantum circuit are processed (line 20). Once the checking of one quantum circuit is concluded, the algorithm will stop using the active qubit pool and process the rest of the gates in the other circuit (lines 22–24). Once completed, the algorithm will compare the two circuits for equivalence by checking whether the resultant DD is an identity DD (lines 25–29).

At the inception, the active qubit pool is initialized with the qubits used in the first quantum gate of the circuit. The algorithm checks whether the qubits required for a specific gate are already present in the active qubit pool. Otherwise, they are added to the pool. The algorithm prioritizes verifying gates that are using qubits already present in the active qubit pool, as opposed to gates involving qubits not included in the pool. This prioritization ensures that gates with available qubits are processed efficiently. After a quantum gate has been verified, the qubits used in that gate are added to the active qubit pool. The active qubit pool has a dynamic maximum pool size, which varies with the size of quantum gates being verified, and is initially set to the size of the initial quantum gate. If the size limit is reached, the algorithm will perform an update to accommodate the qubits required for the next gate. This involves creating a new pool with a maximum size that corresponds to the highest number of qubits required among the gates pending verification. The qubits from the previous gate are also added to this new pool. This step ensures that the pool always contains the qubits relevant to the ongoing verification process. The functional equivalence of the two quantum circuits can be determined based on the inverse property of quantum gates, similar to [4].

However, our initial position match algorithm exhibits shortcomings in specific edge cases, resulting in increased consumption of intermediate DD nodes throughout the verification process. This is illustrated in Fig. 3, where the QuBEC will not process the CX gate at (2, 1) until all quantum gates in the oracle U finished processing, which eliminates the possibility of canceling gates from both circuits. This issue is addressed by incorporating a preprocessing technique known as "reconstructing bridge gates," that converts the four gates bridge gate into a single CX gate prior to commencement of the verification process.

A bridge gate is functionally similar to a SWAP gate where it is used to perform a CX gate where there is no physical connection between them. A typical pattern of a bridge gate is demonstrated in Fig. 3. By applying this preprocessing technique, the bridge pattern will be replaced with a single CX gates only on the original gate and target qubits. Such corner cases can be avoided by eliminating quantum gates that are on the ancilla qubits, and the active qubits pool will function correctly, thus, potentially preventing a slowdown.

B. Quantum Verification With Quantum Error Correction

The challenge of fully solving quantum circuit verification for QEC circuits remains an open problem. In this article, we, for the first time, introduce a solution for quantum circuit verification, which include circuits embedded with QEC. Our approach identifies and eliminates QEC redundancies from a given circuit, to facilitate the equivalence checking process.

Currently, there are various QEC design principles proposed for different architectures [8]. However, QECC is still heavily dependent on manual design for specific quantum circuits

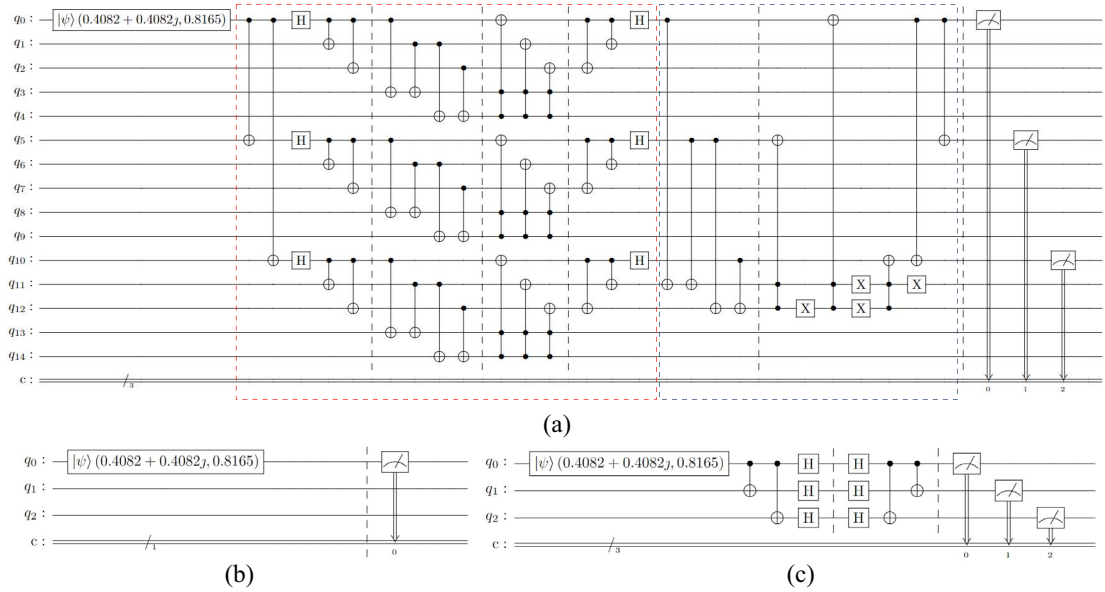

Fig. 2. Proposed verification strategy for QEC-based circuits. (a) Quantum circuit with QEC redundancy. (b) Original circuit. (c) Pruned quantum QEC circuit.

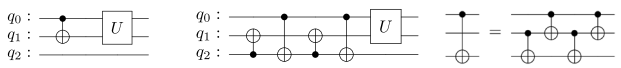

Fig. 3. Example of a corner case. Preprocessing operation converts a bridge CX gates into a single CX gate, in order to avoid failing quantum gate cancelation.

and hardware architectures. As a result, it is challenging to automatically verify whether redundancy and additional circuitry are encoded for QEC. In this article, we introduce a verification flow that integrates the QEC algorithm with QuBEC. This allows us to verify whether the original quantum circuit is functionally identical to the final circuit, which includes the error correction code. Therefore, once the automation of the QEC is implemented, QuBEC can validate the QEC circuit without knowing its functionality.

Fig. 2(a) shows the original circuit for Shor's error correction circuit [9]. Most QECC approaches involve duplicating and measuring the original qubits, alongside the introduction of ancilla bits to provide redundancy for correcting bit-flip and phase-shift errors. The measured bits are intended to serve the same purpose as the original qubits, enabling the algorithm to determine the correct result based on the majority of the measured outcomes. Based on the specification of the QECC, the error correction redundancy does not interfere or change the original computation output. Hence, we can assume that the QEC redundancy on the original circuit will not contain any gates to change the information of the qubits, if all quantum gates on the other qubits are pruned. Therefore, the pruned circuit will have the same functionality as the original without any redundancy for quantum noise. For separate qubit states, the duplication method is relatively simpler, compared to entangled states. Fig. 2(b) demonstrates the qubit state duplication circuit for three qubits, which can be accomplished easily by inserting CX gates. Please note that this pruning technique has to be performed after the quantum circuit is reshaped by reconstructing SWAP gates, since the information of qubits can be changed by using SWAP gates, which may prune out important information and keep the redundancy in the resultant circuit.

To illustrate the pruning approach, let us use a circuit with a single qubit q_0 in a superposition state and phase initialization $|\psi\rangle$ as an example. The error-corrected code, shown in Fig. 2, duplicates qubits q_0, q_5, and q_{10}. Bit-flip error correction is marked in red, while phase-shifting error correction is marked in blue. After computation, these qubits are measured for error correction, and the duplicated qubits vote for the majority output. To verify equivalence with the original 15-qubit QEC circuit, we prune out the duplicated qubits and gates, leaving only the function codes on both sides of the QEC circuit. This pruned circuit can be verified using our Position Match approach mentioned in Section III-A. Even if the two circuits have a different number of qubits, the original circuit will automatically append two empty qubits to match the pruned 3-qubit circuits for equivalence checking.

IV. Experimental Results

In this section, we demonstrate the results of our experiments for evaluating our proposed QuBEC. We evaluate the results furnished by our approach by conducting a comparative study against the state-of-the-art approaches [10].

A. Experimental Setup

In this article, QuBEC is based on the open-source code MQT/QCEC, which is available in GitHub. We conducted the experiments on Windows 11 Enterprise 22H2 version with an Intel Xeon W-2265 CPU processor. The experiments were implemented using g++ version 11.3.0 compiler and CMake version 3.22.1 on a VirtualBox virtual machine with Ubuntu 22.04.1 LTS, with 6 CPU cores assigned to the

TABLE I
COMPARISON OF OUR PROPOSED POSITION MATCH STRATEGY AGAINST EXISTING EQUIVALENCE CHECKING ALGORITHMS

Benchmark Circuits	Time Consumption (ms)						Nodes Overhead					
	Lookahead	Proportional	Position Match	Best Strategy	Acceleration over Lookahead	Acceleration over Proportional	Lookahead	Proportional	Position Match	Best Strategy	Acceleration over Lookahead	Acceleration over Proportional
dk27_225,	97	45	47	Proportional	2.06 ×	0.96 ×	225	565	240	Lookahead	0.94 ×	2.35 ×
pcler8_248,	109	38	54	Proportional	2.02 ×	0.70 ×	681	497	220	**PositionMatch**	3.10 ×	2.26 ×
5xp1_194,	173	183	138	**PositionMatch**	1.25 ×	1.33 ×	971	1482	728	**PositionMatch**	1.33 ×	2.04 ×
alu1_198,	113	69	39	**PositionMatch**	2.90 ×	1.77 ×	457	1330	636	Lookahead	0.72 ×	2.09 ×
mlp4_245,	485	393	314	**PositionMatch**	1.54 ×	1.25 ×	2243	1131	381	**PositionMatch**	5.89 ×	2.97 ×
dk17_224,	188	166	122	**PositionMatch**	1.54 ×	1.36 ×	777	636	311	**PositionMatch**	2.50 ×	2.05 ×
add6_196,	1873	1674	794	**PositionMatch**	2.36 ×	2.11 ×	4388	2838	1434	**PositionMatch**	3.06 ×	1.98 ×
C7552_205,	213	445	131	**PositionMatch**	1.63 ×	3.40 ×	458	422	344	**PositionMatch**	1.33 ×	1.23 ×
cu_219,	134	118	84	**PositionMatch**	1.60 ×	1.40 ×	763	654	384	**PositionMatch**	1.99 ×	1.70 ×
example2_231,	711	594	379	**PositionMatch**	1.88 ×	1.57 ×	1550	982	528	**PositionMatch**	2.94 ×	1.86 ×
c2_181,	460	175	89	**PositionMatch**	5.17 ×	1.97 ×	2787	1036	357	**PositionMatch**	7.81 ×	2.90 ×
rd73_312,	389	251	57	**PositionMatch**	6.82 ×	4.40 ×	2283	8679	869	**PositionMatch**	2.63 ×	9.99 ×
cm150a_210,	8340	433	90	**PositionMatch**	92.67 ×	4.81 ×	426,300	8270	534	**PositionMatch**	798.31 ×	15.49 ×
cm163a_213,	189	140	74	**PositionMatch**	2.55 ×	1.89 ×	816	1121	387	**PositionMatch**	2.11 ×	2.90 ×
sym9_317,	217	828	54	**PositionMatch**	4.02 ×	15.33 ×	1355	43,448	460	**PositionMatch**	2.95 ×	94.45 ×
mod5adder_306,	2654	60,000	221	**PositionMatch**	12.01 ×	271.49 ×	95,318	Did not Finish	3724	**PositionMatch**	25.60 ×	N/A
rd84_313,	23,019	11,967	235	**PositionMatch**	97.95 ×	50.92 ×	248,935	275,939	5147	**PositionMatch**	48.37 ×	53.61 ×
cm151a_211,	345	199	133	**PositionMatch**	2.59 ×	1.50 ×	1882	2994	700	**PositionMatch**	2.69 ×	4.28 ×
apla_203	415	319	265	**PositionMatch**	1.57 ×	1.20 ×	911	828	410	**PositionMatch**	2.22 ×	2.02 ×
adder_n64	113	46101	115	Lookahead	0.98 ×	400.88 ×	2380	1054342	2380	**PositionMatch**	1.00 ×	443.00 ×
qram_n20	415	319	265	**PositionMatch**	1.57 ×	1.20 ×	911	828	410	**PositionMatch**	2.22 ×	2.02 ×

virtual machine. To evaluate the performance of QuBEC, we conducted experiments on 19 benchmark circuits utilized in existing state-of-the-art research [10]. The results obtained were then compared with those of the previous equivalence checking strategies.

B. Result of Position Match Verification Strategy

In this section, we conduct an evaluation of the results furnished utilizing QuBEC on the benchmarks used in [4]. Additionally, we included several commonly utilized quantum circuits, to underscore the scalability of QuBEC. Table I presents the results pertaining to the 21 benchmark circuits examined in this study, encompassing both the current state-of-the-art approaches and QuBEC. The best technique for each benchmark is determined based on the DD nodes consumption, and time consumption. Column 1 consists of the benchmarks we used to evaluate QuBEC. Columns 2–7 demonstrate the time consumption. Columns 2–4 correspond to runtime of Lookahead, Proportional, and Position Match techniques, respectively. Column 5 indicates the best verification strategy for each benchmark used. Columns 6 and 7 demonstrate the improvement in *acceleration* furnished by our Position Match strategy over Lookahead and Proportional strategies, respectively. Columns 8–13 describe results obtained corresponding to node overhead and have identical outline as columns 2–7.

As seen in Table I, QuBEC performs significantly better in terms of time consumption when compared to both Lookahead and Proportional approaches. Compared with Lookahead technique, QuBEC furnishes up to 97.95× acceleration (for benchmark *rd84_313*) over the former. Similarly, our position-match strategy outperforms the Proportional technique by yielding an acceleration greater than 400.88× (for benchmark *adder_n64*). In terms of node consumption, QuBEC provides substantial improvement over both existing techniques, reducing the node overhead by up to 798.31× and 443× over Lookahead and Proportional strategies, respectively, as shown in Table I.

C. Result of QEC Verification

QuBEC is evaluated using three quantum circuits, as shown in Table II. The first column describes the circuit name, followed by the second and third columns, which present the number of gates in unpruned and pruned circuits, respectively, while the last two columns represent the verification time and the number of DD nodes used. These three circuits with 9, 13, and 46 quantum gates with QEC redundancy are pruned to 1, 3, and 11 gates, respectively, to perform the verification with the original circuit without QEC redundancy. We also obtain the time consumption and DD nodes overhead for the equivalence checking process. Table II demonstrates that the QuBEC can verify the quantum circuits within a short-time interval of 40 μs and requiring as low as three DD nodes. This is because the scale of the pruned quantum circuit is smaller than the benchmark circuits shown in Section IV-B. Please note that QuBEC requires the measured qubit to be the functional qubits only, while the ancilla qubits are not measured. This might not hold for some QEC strategies when the ancilla qubits requires measurement. We intend to address these corner cases in the future. Nevertheless, to the best of our knowledge, QuBEC is the first step toward QEC-based equivalence checking for quantum circuit verification.

TABLE II
QUANTUM VERIFICATION WITH QEC STABILIZER CODE

Quantum Circuit	QEC Gates	Pruned Cricuit's Gates	Verification Time (μs)	DD Nodes
Bit-Flip QEC Circuit	9	1	40	3
Phase-Shifting QEC Circuit	13	3	50	3
Shor's QEC Circuit	46	11	130	7

V. CONCLUSION

In this article, we proposed QuBEC, a boosted equivalence checking strategy with embedded QEC code for quantum circuit verification. Our proposed Position Match strategy is based on active qubit pools, that reduces the decision diagram nodes overhead along with time consumption. When evaluated across a set of 21 benchmark circuits, the Position Match algorithm outperforms state-of-the-art approaches for 18 out of 21 benchmarks, with up to 443× reduction in time consumption and 798.31× reduction in DD nodes consumption.

Furthermore, we propose an approach that can verify circuits with QEC redundancy. In the future, we intend to improve the scalability of the proposed approach.

REFERENCES

[1] J. P. Lowe et al., *Quantum Chemistry*. Amsterdam, The Netherlands: Elsevier, 2011.

[2] Y. Li, M. Tian, G. Liu, C. Peng, and L. Jiao, "Quantum optimization and quantum learning: A survey," *IEEE Access*, vol. 8, pp. 23568–23593, 2020.

[3] C. Lu, S. Kundu, A. Arunachalam, and K. Basu, "Survey on quantum noise-aware machine learning," in *Proc. IEEE 15th Dallas Circuit Syst. Conf. (DCAS)*, 2022, pp. 1–2.

[4] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 9, pp. 1810–1824, Sep. 2021.

[5] C.-Y. Wei, Y.-H. Tsai, C.-S. Jhang, and J.-H. R. Jiang, "Accurate BDD-based unitary operator manipulation for scalable and robust quantum circuit verification," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 523–528.

[6] T.-F. Chen, J.-H. R. Jiang, and M.-H. Hsieh, "Partial equivalence checking of quantum circuits," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, 2022, pp. 594–604.

[7] X. Hong, M. Ying, Y. Feng, X. Zhou, and S. Li, "Approximate equivalence checking of noisy quantum circuits," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 637–642.

[8] J. Roffe, "Quantum error correction: An introductory guide," *Contem. Phys.*, vol. 60, no. 3, pp. 226–245, 2019.

[9] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, no. 4, 1995, Art. no. R2493.

[10] L. Burgholzer and R. Wille, "Improved DD-based equivalence checking of quantum circuits," in *Proc. 25th Asia South Pacific Design Autom. Conf.*, 2020, pp. 127–132.