Accelerating Worst-Case Equivalence Checking of Classical Circuits via Quantum Search

Lukas Bombel, Adrian Osterwind, Domenik Helms, Nils Quetschlich and Robert Wille
German Aerospace Center (DLR), Oldenburg, Germany
{lukas.bombel, adrian.osterwind, domenik.helms}@dlr.de
Technical University of Munich, Munich, Germany
{nils.quetschlich, robert.wille}@tum.de

Abstract

Despite major advances in circuit verification, the complexity scaling of equivalence checking remains exponential in worst-case scenarios. To address this bottleneck, we present a methodological approach to achieve a quadratic speedup in these critical scenarios by leveraging an optimized quantum search method. The increasing capabilities of quantum computers and their promise to decrease computational complexity motivate this exploration of quantum alternatives for circuit verification. While prior research was limited by the requirement of knowing the exact number of counterexamples for two given circuits, our approach avoids this requirement. We analyze the complexity scaling of the method, showing that a high success probability can be ensured while maintaining the promised speedup across all stages. Although current quantum hardware constraints limit practical use, we demonstrate that quantum algorithms potentially offer a viable extension to classical equivalence checking in hard cases, paving the way for future integration into circuit verification workflows.

1 Introduction

Equivalence checking of digital circuits remains a critical task in hardware design. The rapid deployment of edge hardware has led to increasingly complex digital circuits, motivating the search for more efficient verification methods. Modern approaches like Binary Decision Diagrams (BDDs) or Boolean Satisfiability (SAT) solvers handle typical cases efficiently, but their computational complexity remains exponential in worst-case scenarios [1].

Quantum search offers a potential way to reduce this computational effort, specifically in the form of Grover's algorithm [2]. Treating the equivalence checking of combinational circuits with n bits as an unstructured database search over all 2^n possible inputs, it offers a theoretical quadratic speedup in the computational complexity compared to exhaustive search. A major limitation of this algorithm is that the fraction of counterexamples must be known for reliable execution.

In this work, we address this problem by employing a modified version of the algorithm proposed by Yoder et al. [3], avoiding the requirement of knowing the fraction of counterexamples and adding only a small overhead. Unlike previous approaches, we present a method for applying Grover's algorithm to classical circuit equivalence checking that maintains the quadratic speedup across all steps. Although quantum algorithms are not yet feasible for large circuits, our primary objective is to outline a practical pathway for integrating them into classical verification, e.g., for circuit analysis and optimization.

The remainder of this work is structured as follows: Section 2 briefly reviews classical equivalence checking and previous work on employing Grover's algorithm for this task. The proposed quantum verification method is described in Section 3, outlining the basics of Grover's algorithm and the key implementation steps. In Section 4, we compare the computational complexity scaling with classical brute-force search and previous quantum methods and discuss resource requirements. Key findings are summarized in Section 5, outlining the limitations of quantum verification and future research directions.

2 Related Work

Combinational equivalence checking primarily relies on two classical methods, Binary Decision Diagrams (BDDs) and Boolean Satisfiability (SAT) solvers. In BDD-based methods, the circuits under test are represented as canonical directed acyclic graphs. If the BDDs of two circuits are identical, the circuits are equivalent [4].

In SAT-based methods, the circuits are encoded as a satisfiability problem of a Boolean formula. If the SAT-solver finds an input assignment which satisfies the equation, that assignment corresponds to a counterexample, proving non-equivalence. Despite their efficient solving capabilities for most circuits, classical equivalence checking methods scale exponentially with input size in worst-case scenarios [5].

Quantum search algorithms offer a theoretical quadratic speedup to overcome this bottleneck, though practical implementations remain challenging. Several works have explored the use of Grover's algorithm for this purpose, but struggle to maintain the promised speedup in practice.

Quetschlich et al. [6] generate reversible quantum oracles from truth table representations and propose a method for handling the unknown number of target states in Grover's algorithm. However, the method relies on generating a probability distribution for all possible inputs from a large number of repetitions (shots) of the full algorithm, exceeding the classical worst-case complexity of 2^n.

Kole et al. [1] derive the Exclusive-Sum-of-Products (ESOP) based on the Conjunctive Normal Form (CNF) to generate reversible quantum circuits. While they optimize circuit depth and qubit count, the ESOP generation remains computationally challenging and the method still suffers from the problem of unknown target states due to the standard Grover implementation.

Stoudenmire et al. [7] implement a Grover-like search algorithm in classical computing. While this maintains a linear number of oracle calls, the simulation cost remains exponential for worst-case circuits.

These limitations highlight the need for a verification framework that can handle an unknown number of counterexamples without sacrificing the quadratic speedup, which is the gap addressed by the proposed method in this work.

3 Proposed Method

The objective of equivalence checking is to determine whether two circuits behave identically for all possible input combinations. Based on the number of bits n, this task can be framed as an unstructured search problem over a space of $N = 2^n$ inputs to find at least one counterexample. To achieve a quadratic speedup over this classical exhaustive search, we employ a modified version of Grover's algorithm.

Apart from the synthesis technique used, our method described below is based on a similar approach to that in [1] and [6], while using the modified version of Grover's algorithm proposed by Yoder et al. [3].

3.1 Quantum Search and Amplitude Amplification

Grover's algorithm [2] is a quantum computing method designed to find target elements in an unsorted database. The general procedure of the algorithm is visualized in Fig. 1 and begins with the initialization of all n input quantum bits (qubits) in a uniform superposition. The subsequent oracle and diffusion steps are then applied repeatedly, amplifying the amplitudes of the target states, while suppressing the amplitudes of the remaining states. While the oracle encodes the circuit under test and marks all target states (counterexamples) with a phase flip, the diffusion step reflects the amplitudes of the basis states around their mean value, amplifying the probability of measuring a target state [2]. As the system collapses into one of these states upon measurement, only a single observation can be made per algorithm execution (shot) [8]. Due to the probabilistic nature of quantum measurement, the observed state may vary between executions, which is why multiple shots are typically required to reliably identify target states.

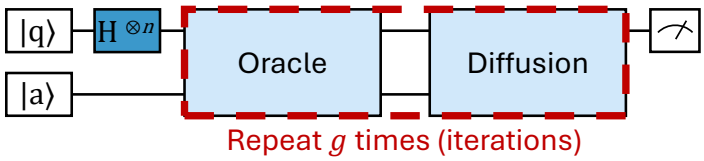

Figure 1: Circuit representation of Grover's algorithm as implemented in this work. Hadamard gates H are applied to all input qubits $|q\rangle$ to initialize a uniform superposition, while the ancillary qubits $|a\rangle$ are left in the initial state $|0\rangle$. After applying the oracle and diffusion steps for g iterations, the input qubits $|q\rangle$ are measured

To maximize the probability of measuring a counterexample, the oracle and diffusion steps are repeated g times. For $M \ll N$, the optimal number of iterations can be approximated as

$$g \approx \frac{\pi}{4}\sqrt{\frac{N}{M}} - \frac{1}{2} \tag{1}$$

with the total number of inputs $N = 2^n$ and the number of counterexamples M [8]. While this corresponds to a computational complexity scaling of $\sqrt{2^n}$ in principle, M is generally unknown in practical equivalence checking. In this case, choosing the wrong number of iterations can cause the algorithm to overshoot or undershoot, resulting in the target states no longer being correctly marked and thus significantly reducing the success probability.

To resolve this problem, we employ the modified amplitude-amplification scheme proposed by Yoder et al. [3]. This method ensures a minimum probability P of finding a target state as long as the ratio $\lambda = \frac{M}{N}$ is above a chosen threshold $\hat{\lambda}$. Instead of the fixed $180°$ phase rotations, this approach uses specific partial rotations for each iteration. The required number of iterations g is determined by

$$g \geq \frac{\ln(2/\delta)}{2\sqrt{\hat{\lambda}}} \tag{2}$$

using the desired $P = 1 - \delta^2$ and $\hat{\lambda}$. The rotation angles are calculated via Chebyshev polynomials of the first kind [9], which are chosen such that a minimum P is guaranteed for all $\lambda \in [\hat{\lambda}, 1]$.

3.2 Algorithm Construction

To implement the classical circuits under test on quantum hardware, they must first be synthesized into reversible quantum oracles. Since this is not the main focus of this paper, we use a simple method based on the miter circuit configuration proposed by Brand [10]. As shown in Fig. 2, this directly compares the outputs of two circuits for every input combination. While equivalent circuits result in a miter output of 0 for all inputs in this configuration, any counterexample drives the output to 1.

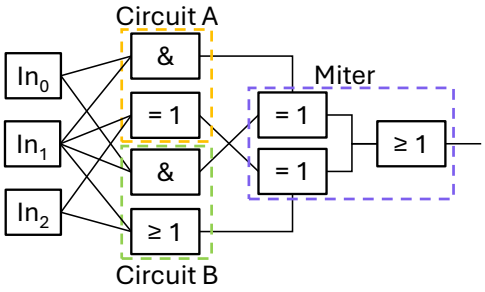

Figure 2: Miter configuration of two non-equivalent circuits. Corresponding outputs are compared via XOR gates and the outputs of all XOR gates are then combined with a final OR gate

Since quantum operations must be reversible, we adopt a gate-by-gate approach to replace the classical logic gates with reversible quantum equivalents. These are shown in Tab. 1 for all logic gate types used in this work. Note that some classical gates must be replaced with multiple quantum gates to achieve the same functionality. Also, an ancillary qubit is needed for storing the intermediate outputs for each gate. While an X gate, for example, is in itself equivalent to a classical NOT gate, the state after applying the gate needs to be written to a fresh ancillary qubit for further use, equivalent to using a new wire in classical circuits. While this increases the total qubit count, the ancillary qubits $|a\rangle$ are kept separate from the input qubits $|q\rangle$, as shown in Fig. 1. As a result, they do not expand the search space and therefore do not increase the required number of iterations g for the algorithm.

While there are synthesis methods that result in more computationally optimal circuits, these methods are often already associated with exponential complexity [11], which would negate the speedup of the quantum search. Consequently, not only the quantum algorithm itself, but also the synthesis and resulting circuit must be efficient in terms of computational complexity to achieve an overall speedup compared to classical exhaustive search. Taking into account both the synthesis and algorithm runtime,

Table 1: Reversible equivalents of classical gates for inputs *a* and *b* and output *c* in Qiskit notation. The output qubits are assumed to be initialized to $|0\rangle$

Classical Gate	**Quantum Equivalent (Qiskit)**
AND	ccx([*a*, *b*], *c*)
NAND	ccx([*a*, *b*], *c*); x(*c*)
OR	x([*a*, *b*]); ccx([*a*, *b*], *c*); x([*a*, *b*]); x(*c*)
NOR	x([*a*, *b*]); ccx([*a*, *b*], *c*); x([*a*, *b*])
XOR	cx(*a*, *c*); cx(*b*, *c*)
NOT	cx(*a*, *c*); x(*c*)

the total (time) complexity of the verification method can be formulated as

$$t_{\text{total}} = t_{\text{synthesis}} + g \times t_{\text{iteration}} \tag{3}$$

with the complexity of the synthesis $t_{\text{synthesis}}$ as well as the number of algorithm iterations g and the complexity per iteration $t_{\text{iteration}}$.

Using the synthesized oracle circuit, the full quantum circuit for realizing the quantum algorithm is constructed and executed as shown in Fig. 1. After executing the oracle and diffusion steps for g iterations according to Eq. (2), the final state is measured and can then be verified classically. If this state is not a counterexample, it may indicate either circuit equivalence or an unsuccessful search. Thus, the probability P of measuring a counterexample must be set to a sufficient level for the task. To guarantee the measurement of a counterexample with probability P, a worst case of $M = 1$ in Eq. (2) is assumed in our method.

4 Evaluation and Discussion

This section evaluates the computational behavior of the quantum equivalence checking method described in Section 3. First, the theoretical complexity scaling is analyzed in comparison to the standard Grover and classical brute-force search based on a set of benchmark circuits. Then, simulation results based on the smallest circuit of this list are presented to validate the predicted success probabilities of the algorithm. Finally, we discuss the practical limitations that determine feasibility on the quantum hardware currently available.

4.1 Theoretical scaling

The complexity scaling of the proposed method is compared to the standard Grover algorithm and classical worst-case scaling using the ISCAS-85 circuit netlists [12]. The details of the 11 circuits and scaling results of the methods are summarized in Tab. 2. As the complexity of Yoder's modified version depends on the desired probability of finding a counterexample, a target success probability of $P = 0.99999$ is chosen as an example to highlight the overhead compared to the standard Grover algorithm.

According to Eq. (2), decreasing δ^2 by a factor of ten adds a computational overhead of $\approx 0.58 \cdot \sqrt{N}$ for Yoder's algorithm. Compared to the standard Grover algorithm, this method consequently increases

Table 2: Complexity analysis of the proposed method (for $P = 0.99999$) and the standard Grover algorithm on ISCAS-85 benchmark circuits compared to classical search

Circuit	**Inputs** n	**Gates** l	**Outputs** m	**Total states** $N = 2^n$	**Yoder Iterations**	**Grover Iterations (best case)**
c17	5	6	2	$3.20 \cdot 10^1$	$1.80 \cdot 10^1$	$0.40 \cdot 10^1$
c6288	32	2416	32	$4.29 \cdot 10^9$	$2.11 \cdot 10^5$	$5.15 \cdot 10^4$
c1908	33	880	25	$8.59 \cdot 10^9$	$2.99 \cdot 10^5$	$7.28 \cdot 10^4$
c432	36	160	7	$6.87 \cdot 10^{10}$	$8.45 \cdot 10^5$	$2.05 \cdot 10^5$
c499	41	202	32	$2.20 \cdot 10^{12}$	$4.78 \cdot 10^6$	$1.16 \cdot 10^6$
c1355	41	546	32	$2.20 \cdot 10^{12}$	$4.78 \cdot 10^6$	$1.16 \cdot 10^6$
c3540	50	1669	22	$1.13 \cdot 10^{15}$	$1.08 \cdot 10^8$	$2.64 \cdot 10^7$
c880	60	383	26	$1.15 \cdot 10^{18}$	$3.46 \cdot 10^9$	$8.43 \cdot 10^8$
c5315	178	2307	123	$3.83 \cdot 10^{53}$	$2.00 \cdot 10^{27}$	$4.86 \cdot 10^{26}$
c7552	207	3513	108	$2.05 \cdot 10^{62}$	$4.62 \cdot 10^{31}$	$1.12 \cdot 10^{31}$
c2670	233	1269	140	$1.38 \cdot 10^{70}$	$3.79 \cdot 10^{35}$	$9.22 \cdot 10^{34}$

the number of iterations by a constant factor based on the desired P, which is equivalent to a fixed offset in logarithmic scaling. In the example of Tab. 2, this factor is about 4.1, with the exception of c17, due to the deviations in Eq. (1) for $N \approx M$.

For larger circuits, however, this overhead is negligible when compared to the speedup over checking through all possible input combinations classically. Also, as mentioned in Section 3.1, Grover's algorithm assumes that the number of counterexamples is known. Otherwise, the algorithm must be repeated with varying M according to Eq. (1) until a solution is found. In this case, the speedup of the algorithm compared to classical brute-force search would be lost.

4.2 Simulation-based validation

In addition to comparing the theoretical complexity scaling, the presented method was practically verified using the c17 circuit as an example. The algorithm was implemented using the quantum computing simulation framework Qiskit by IBM (see [13]). For this, 33 additional 5-bit circuits were generated, with each circuit k ($k = 0, 1, 2, ..., 32$) differing in its output from c17 for exactly k of the $N = 32$ possible input combinations. Here, $P = 0.9$ was chosen to illustrate the algorithm behavior across the different values of λ. As the true λ would not be known in real equivalence checking tasks, a value of $\hat{\lambda} = 1/32$ was assumed in all cases, which resulted in $g = 5$ algorithm iterations. Each data point was obtained from 1000 shots to get a sufficient estimate of the underlying P, since each algorithm measurement is equivalent to a random experiment with probability P of measuring a counterexample.

The results are visualized in Fig. 3, showing a good fit of the simulation data to the theoretical probabilities across all λ. This agreement confirms that the theoretical success probability accurately reflects the algorithm behavior, including the dependence on λ. Small deviations in success probability can be attributed to statistical uncertainty resulting from the finite shot count.

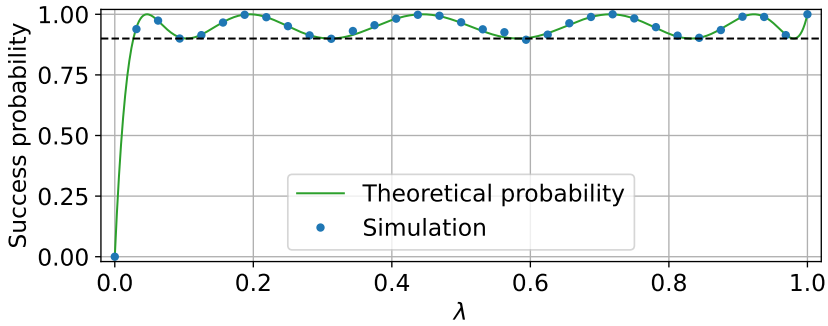

Figure 3: Theoretical probability of measuring a target state as a function of $\lambda = M/N$ for $n = 5$ qubits using Yoder's algorithm for $\hat{\lambda} = 1/32$ and $P = 0.9$ compared to simulation results

4.3 Hardware feasibility and practical constraints

The ISCAS-85 benchmark also highlights the effects of structural differences between circuits. For example, the circuits `c499` and `c1355` have identical numbers of inputs and outputs, resulting in the same search space size and therefore the same number of algorithm iterations. However, `c1355` contains almost three times as many gates. This illustrates that while the algorithm complexity scaling is determined by the number of input bits, it must also be noted that each logical gate increases the qubit count in our gate-by-gate synthesis approach.

For this reason, our method is not yet feasible on actual quantum hardware for large circuits. Instead, it serves as a proof-of-concept for a full quantum verification method of classical circuits yielding a quadratic speedup compared to classical worst-case complexity.

In practice, it must also be taken into account that a deeper quantum circuit entails a longer algorithm runtime and, in turn, an increased risk of errors on quantum hardware. Thus, tradeoffs between iteration count and circuit width and depth must be balanced. The proposed method works only if it can be guaranteed that the quantum hardware is computationally stable regarding circuit depth and that it has enough qubits available. If that is not the case, the synthesized quantum circuit must be optimized in advance, while still maintaining a sufficiently low complexity scaling in total.

Also, it has to be considered that many gates used in quantum simulation software are not directly implementable on quantum hardware. Instead, they have to be substituted by specific gate sets, such as the Clifford+T set, which leads to additional complexity and gate overhead [14].

5 Conclusion

In this paper, we presented an overview of how to leverage quantum search algorithms for equivalence checking of classical circuits. We highlighted the use of Yoder's modified version of Grover's algorithm to handle the unknown number of counterexamples in equivalence checking, which has been largely overlooked in other similar research to date. The method offers the theoretical quadratic speedup promised by Grover's algorithm for cases that exhibit the worst-case 2^n-scaling in classical equivalence checking, while only adding a small computational overhead compared to the standard algorithm. While the complexity scaling remains exponential, the speedup would enable circuits with

twice as many inputs to be verified with the same computational effort, assuming comparable costs for quantum and classical execution.

Nevertheless, retaining the quadratic speedup throughout all steps involved in the method remains an essential task, especially given the limited resources and error susceptibility of current quantum hardware. Future work could focus on optimizing reversible synthesis, both in terms of the synthesis complexity itself and the number of quantum gates and qubits in the resulting circuit. Once these demands are met even for large circuits, previously unsolvable worst-case 2^n-scenarios can be solved. Thus, quantum verification methods represent a promising potential addition to classical state-of-the-art methods.

References

[1] Abhoy Kole, Mohammed E. Djeridane, Lennart Weingarten, Kamalika Datta, and Rolf Drechsler. qSAT: Design of an Efficient Quantum Satisfiability Solver for Hardware Equivalence Checking. *ACM Journal on Emerging Technologies in Computing Systems*, 2025.

[2] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, pages 212–219, New York, New York, USA, 1996. ACM Press.

[3] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. Fixed-point quantum search with an optimal number of queries. *Physical review letters*, 113(21):210501, 2014.

[4] Paul Molitor and Janett Mohnke. *Equivalence Checking of Digital Circuits : Fundamentals, Principles, Methods*. Springer Science+Business Media, 2010.

[5] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021.

[6] Nils Quetschlich, Tobias V. Forster, Adrian Osterwind, Domenik Helms, and Robert Wille. Towards Equivalence Checking of Classical Circuits Using Quantum Computing. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 268–274. IEEE, 2024.

[7] E. M. Stoudenmire and Xavier Waintal. Opening the Black Box Inside Grover's Algorithm. *Physical Review X*, 14(4), 2024.

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2012.

[9] Theodore J. Rivlin. *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*. Wiley, New York, 2nd edition, 1990.

[10] Daniel Brand. Verification of large synthesized designs. In *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pages 534–537. IEEE Comput. Soc. Press, 1993.

[11] Jessie M. Henderson, Elena R. Henderson, Aviraj Sinha, Mitchell A. Thornton, and D. Michael Miller. Automated Quantum Oracle Synthesis with a Minimal Number of Qubits.

[12] Francis Brglez and Hiroshi Fujiwara. A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, pages 677–692, 6 1985.

[13] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.

[14] Yan Ge, Wu Wenjie, Chen Yuheng, Pan Kaisen, Lu Xudong, Zhou Zixiang, Wang Yuhan, Wang Ruocheng, and Yan Junchi. Quantum Circuit Synthesis and Compilation Optimization: Overview and Prospects, 2025.