

Quantum Counting for Topological Analysis of Noncoherent Systems

Tomas Sobek

Department of Informatics,
Faculty of Management Science and Informatics,
University of Žilina
Univerzitná 8215/1, 010 26 Žilina
Email: tomas.sobek@fri.uniza.sk

Miroslav Kvassay

Department of Informatics,
Faculty of Management Science and Informatics,
University of Žilina,
Univerzitná 8215/1, 010 26 Žilina
Email: miroslav.kvassay@fri.uniza.sk

Abstract—Topological analysis of binary-state systems is a fundamental part of reliability engineering. While efficient computing methods exist for coherent systems, noncoherent systems have no general subexponential method and in the worst case require enumeration of all 2^n states. Recent works have explored quantum computing for reliability analysis through Bayesian networks and general optimization frameworks, but no prior study has addressed computation of topological measures of noncoherent systems using quantum solution. This paper presents a systematic procedure for constructing a quantum oracle from a compact Boolean netlist and combines it with the quantum counting algorithm to achieve quadratic speedup. This research also investigates ancilla qubit recycling, achieving 18–46% qubit reduction, and identifies its incompatibility with controlled operations in quantum phase estimation. The approach is verified on five reference systems and LGSynth’91 benchmarks. These results establish quantum counting as a viable computational framework for topological analysis of noncoherent systems.

I. SYMBOLS AND NOTATION

Symbol	Description
ϕ	Structure function
n	Number of components
M	Number of functional states
d	Truth density ($M/2^n$)
s	Number of gates in netlist
t	QPE counting qubits
θ	Grover rotation angle
ω	Generic phase in QPE
y	QPE measurement result
G	Grover operator

II. INTRODUCTION

RELIABILITY analysis of various systems is one of the key elements of reliability engineering. For a binary-state system composed of n components, where each component can be in one of two states (functional represented by number 1 or non-functional expressed as number 0), the key task is to

This work was supported by the Ministry of Education, Research, Development and Youth of the Slovak Republic “Reliability Analysis based on Machine Learning Methods” under Grant VEGA 1/0331/25 and by the grant of Slovak Research and Development Agency “Reliable Methods for Intelligent Detection of Defects in Wind Energy Facilities” (reg. no. SK-UA-25-0088).

determine the reliability of the system as a whole. This task formally reduces to the evaluation of the structure function [1]:

$$\phi(\mathbf{x}) : \{0, 1\}^n \rightarrow \{0, 1\} \quad (1)$$

which, for each combination of component states, determines whether the system works (state 1) or not (state 0). The binary-state model is the simplest case; many real systems are more naturally described as multi-state systems, in which both the components and the system itself can occupy several levels of performance between fully functional and failed [2]. The structure-function formalism extends directly to this setting, and topological measures such as the truth density have multi-state counterparts [3]. Noncoherence is likewise meaningful for multi-state systems, where it has recently been studied in the context of importance analysis [4]; in this paper, however, we restrict ourselves to the binary-state case.

For coherent systems where the structure function is monotone (repairing a component can never worsen the state of the system), there is a rich apparatus of analytical methods, such as minimum cuts, minimum paths, inclusion-exclusion principle, and others. These methods allow computing reliability without enumerating the entire state space of size 2^n . The situation is different for noncoherent systems, where the structure function is not monotone. In such systems, repairing a component can paradoxically cause the system to fail, for example, in protection circuits, fail-safe mechanisms, or systems with redundant switching, where the switch itself can fail. For noncoherent systems, classical methods based on minimum cuts and paths fail because the notion of a minimum cut in the classical sense loses meaning in the absence of monotony. Thus, an accurate calculation of reliability in the worst case requires enumeration of all 2^n states, which implies a complexity of $\Omega(2^n)$. Quantum computing has recently been explored as a tool for complex combinatorial optimization [5] or reliability analysis [6], [7]. However, these and other works have not addressed the specific problem of truth density computation for noncoherent systems using quantum counting. In this paper, we aim to use a quantum approach to this specific problem. We use a quantum counting algorithm, which combines Grover’s search algorithm [8] with Quantum Phase Estimation (QPE), to estimate the number of functional states of a system M with

a complexity of $\mathcal{O}(\text{poly}(n) \cdot 2^{n/2})$ oracle calls. This represents a theoretical quadratic speedup over the classical approach by an exponential factor.

The key contribution of the paper is the proposal of a systematic procedure for constructing a quantum oracle from a compact representation of a noncoherent system in the form of a Boolean netlist. This procedure involves the translation of classical logic gates (AND, OR, NOT, XOR, and others) into reversible quantum operations, the management of ancilla qubits, and their uncomputation, which is necessary for preserving quantum coherence in the quantum counting algorithm. In addition, we investigate the optimization of the number of qubits by recycling ancilla qubits and present an important negative result on the incompatibility of this optimization with controlled operations in QPE.

The implementation is done in the Qiskit (framework developed by IBM) and verified on reference noncoherent systems with 3 to 6 components. In addition, a parser for the PLA format is designed, which allows the application of the algorithm to standard benchmarks from the LGSynth'91 suite.

III. RELIABILITY OF BINARY-STATE SYSTEMS

A. Structure Function

Let us consider a system S composed of n components. Each component $i \in \{1, \dots, n\}$ is in one of two states: functional ($x_i = 1$) or non-functional ($x_i = 0$). The state vector of the system is $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$. The total state space has 2^n elements.

The structure function $\phi(\mathbf{x}) : \{0, 1\}^n \rightarrow \{0, 1\}$ assigns a system state to each state vector: $\phi(\mathbf{x}) = 1$ if the system is working, $\phi(\mathbf{x}) = 0$ if the system is not working. The structure function completely characterizes the logical structure of the system, independent of the failure probabilities of individual components [1].

The truth vector $\Phi \in \{0, 1\}^{2^n}$ is defined as an ordered sequence of values of the structure function for all state vectors in lexicographic ordering. The state vectors $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{2^n-1}$ correspond to binary representations of indices $0, 1, \dots, 2^n-1$, e.g., for $n = 3$: $\mathbf{x}_0 = (0, 0, 0)$, $\mathbf{x}_1 = (0, 0, 1)$, $\dots$, $\mathbf{x}_7 = (1, 1, 1)$:

$$\Phi = (\phi(\mathbf{x}_0), \phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_{2^n-1})) \quad (2)$$

B. Coherent and Noncoherent Systems

A coherent system satisfies the monotonicity condition: if $\mathbf{x} \leq \mathbf{y}$ (in terms of component states), then $\phi(\mathbf{x}) \leq \phi(\mathbf{y})$ [1]. This means that repairing any component can never cause the system to fail. Coherent systems further require the relevance of each component and the boundary conditions $\phi(\mathbf{0}) = 0$, $\phi(\mathbf{1}) = 1$.

For coherent systems, there are efficient computational methods based on monotony. A system is uniquely characterized by its minimum cuts (the smallest set of components whose failure will cause the system to fail) [9] or minimum paths (the smallest set of components whose functionality will ensure the functionality of the system) [10]. The methods sum of disjoint products (SDP) [11], inclusion-exclusion

principle [12], and recursive decomposition (pivoting) with pruning due to monotony allow computing reliability in time significantly less than $\mathcal{O}(2^n)$.

A noncoherent system violates the monotone condition [1]. The structure function ϕ is a general Boolean function — not necessarily monotone [13]. This means that there is a situation where repairing a component causes the system to fail, or vice versa, where the system “fixes” a component’s failure. Noncoherent behavior occurs in real systems: protection circuits, logic circuits [14], fail-safe mechanisms, redundant switching systems where a switch can fail [15], network topologies with flow control logic, or systems where the failure of a single component activates a backup mode [16]. For noncoherent systems, methods based on minimum cuts and paths fail because there is no ordering on the set of failure states compatible with the structure function [10]. There is no guarantee that if a subset of components causes a failure, then their superset also causes a failure. The Boolean function ϕ can have any structure without any monotone regularity.

C. Truth Density

The truth density of a structure function is defined as:

$$d = \text{TD}(\phi) = \frac{M}{2^n} \quad (3)$$

where $M = |\{\mathbf{x} \in \{0, 1\}^n : \phi(\mathbf{x}) = 1\}|$ is the number of ones in the structure vector, i.e., the number of functional states of the system.

Truth density is a fundamental quantity for topological reliability analysis [3]. In the case where all components have the same and independent probability of functionality $p = 1/2$, the truth density is directly equal to the reliability of the system [1]. In the more general case of independent components with different probabilities p_i , the reliability is:

$$R(p_1, p_2, \dots, p_n) = \sum_{\mathbf{x}: \phi(\mathbf{x})=1} \prod_{i=1}^n p_i^{x_i} (1 - p_i)^{1-x_i} \quad (4)$$

Also for this general case, knowledge of M is a useful statistical characteristic of the system.

Formally, we define the computational problem as follows:

Remark 3.1 (Counting- ϕ): Given a Boolean function $\phi(\mathbf{x}) : \{0, 1\}^n \rightarrow \{0, 1\}$ (the structure function of a noncoherent system), specified as a Boolean circuit of size $s = \text{poly}(n)$, the task is to determine $M = |\phi^{-1}(1)|$.

Here $\phi^{-1}(1) = \{\mathbf{x} \in \{0, 1\}^n : \phi(\mathbf{x}) = 1\}$ denotes the preimage of 1 under ϕ , i.e., the set of all state vectors for which the system is functional. The notation $|\cdot|$ denotes set cardinality, so M is the number of such vectors. The assumption $s = \text{poly}(n)$ means the circuit size grows at most polynomially with the number of components, which is satisfied by all practical engineering systems.

Counting the satisfying assignments of a general Boolean function is #P-complete [17], so no worst-case polynomial-time algorithm is expected. In a black-box setting, where ϕ is accessible only through evaluations, $\Omega(2^n)$ queries are unavoidable. When ϕ is given as a compact circuit, symbolic

methods (Section III-D) can be far more efficient on structured instances; their worst-case complexity, however, remains exponential and they provide no guaranteed subexponential bound for arbitrary noncoherent functions.

It is important to note that the total classical complexity also includes the generation of the structure vector, not just its traversal. If the structure vector were available as an explicit table, simply counting the ones would cost $\mathcal{O}(2^n)$. However, generating this table itself requires $\mathcal{O}(s \cdot 2^n)$ operations, where s is the number of logic gates in the Boolean circuit representing ϕ (i.e., for each of the 2^n state vectors, evaluating ϕ requires traversing all s gates), and for large n the table cannot even be stored in memory.

D. Classical Symbolic Methods

Beyond explicit enumeration, the reliability literature offers symbolic techniques that avoid materialising the full 2^n state space. Binary Decision Diagrams (BDDs) [18] represent ϕ as a canonical directed acyclic graph and compute M (and reliability) in time proportional to the diagram size, which is often polynomial for structured systems but can grow exponentially in the worst case and is highly sensitive to variable ordering. For noncoherent systems in particular, BDD-based and logical-differential approaches have been developed, including work by the present authors and close collaborators [19], [20]. These methods are the appropriate classical baseline for compactly specified functions and frequently outperform brute-force enumeration in practice. We therefore position quantum counting as *complementary* rather than uniformly superior: it does not dominate symbolic methods on instances where the BDD stays small, but it offers a guaranteed worst-case query advantage that holds uniformly over all non-monotonous functions, including those whose every variable ordering yields an exponential-size diagram.

IV. BASICS OF QUANTUM COMPUTING

This section provides the essential basics of quantum computing for the reader familiar with the classical computational model. A more detailed introduction can be found, for example, in [21].

A. Qubit and Superposition

A qubit is a fundamental unit of quantum information, analogous to a classical bit. Unlike a classical bit, which is in one of two states, 0 or 1, a qubit can be in a superposition of both states simultaneously:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (5)$$

where $\alpha, \beta \in \mathbb{C}$ are probabilistic amplitudes satisfying $|\alpha|^2 + |\beta|^2 = 1$. When a qubit is measured, it “collapses” into the state $|0\rangle$ with probability $|\alpha|^2$ or into the state $|1\rangle$ with probability $|\beta|^2$.

A register of n qubits can be in a superposition of all 2^n states simultaneously:

$$|\psi\rangle = \sum_{\mathbf{x}} \alpha_{\mathbf{x}} |\mathbf{x}\rangle \quad (6)$$

where $\alpha_{\mathbf{x}} \in \mathbb{C}$ is the probability amplitude associated with state $|\mathbf{x}\rangle$, satisfying the normalization condition $\sum_{\mathbf{x}} |\alpha_{\mathbf{x}}|^2 = 1$. Upon measurement, the state $|\mathbf{x}\rangle$ is observed with probability $|\alpha_{\mathbf{x}}|^2$. This is the basis of quantum parallelism, meaning a quantum computer can “process” exponentially many states at once, even though only one result is obtained when a measurement is made.

B. Quantum Gates and Oracle

Quantum computation is implemented by applying quantum gates, namely unitary operators, to qubits. Basic gates include: the Hadamard gate H , the Pauli X gate, the CNOT (controlled NOT — flips the target qubit if the control is in the state $|1\rangle$), and the Toffoli gate (controlled-controlled NOT, equivalent to a reversible AND) [22].

A key property of quantum gates is reversibility, since each gate must implement a unitary operation, meaning that the computation is always invertible. This is a fundamental difference from classical logic gates (for example, AND has 2 inputs and 1 output, which loses information). This results in the need for auxiliary (ancilla) qubits in the construction of quantum analogues of classical gates.

A quantum oracle is a special unitary operation U_{ϕ} that, for a given Boolean function ϕ , performs the transformation:

$$U_{\phi}: |\mathbf{x}\rangle |q\rangle \rightarrow |\mathbf{x}\rangle |q \oplus \phi(\mathbf{x})\rangle \quad (7)$$

The input $\mathbf{x}$ remains intact and the result $\phi(\mathbf{x})$ is XORed into an auxiliary qubit. If the auxiliary qubit is prepared in the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, the XOR operation does not change the auxiliary qubit’s state but instead transfers the function value into the phase of the input register — a mechanism known as phase kickback:

$$|\mathbf{x}\rangle |-\rangle \rightarrow (-1)^{\phi(\mathbf{x})} |\mathbf{x}\rangle |-\rangle \quad (8)$$

When $\phi(\mathbf{x}) = 0$, the factor $(-1)^0 = 1$ leaves the state unchanged. When $\phi(\mathbf{x}) = 1$, the factor $(-1)^1 = -1$ flips the sign of the amplitude. This phase is not directly observable on a single state, but when the input register is in superposition, the relative phase differences between states enable interference, the foundation of both Grover’s search and quantum counting. Generally, construction of quantum oracle plays a key role for efficiency of these algorithms [23].

C. Quantum Phase Estimation

Quantum Phase Estimation (QPE) is an algorithm that, given a unitary operator U and its eigenvector $|u\rangle$ with eigenvalue $e^{2\pi i \omega}$, estimates the phase ω [24]. QPE uses a t -qubit counting register and a working register in the state $|u\rangle$. After applying controlled powers of U and the inverse Quantum Fourier Transform (QFT $^{-1}$) to the counting register, the counting register measurement is concentrated around the value $y \approx 2^t \omega$. The accuracy of the estimate depends on t ; more qubits in the counting register means finer phase resolution.

V. QUANTUM COUNTING ALGORITHM

A. Grover's Operator

The Grover operator G is a composition of two operations: the phase oracle U_ϕ , which marks “good” states (where $\phi(\mathbf{x}) = 1$) with phase -1 , and the diffuser $D = 2|\psi\rangle\langle\psi| - I$ (where I is the identity matrix), which realizes the reflection around the uniform superposition $|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n}$.

Geometrically, G is a rotation by an angle θ in a two-dimensional subspace spanned by states $|\alpha\rangle$ (a uniform superposition of “bad” states where $\phi(\mathbf{x}) = 0$) and $|\beta\rangle$ (a uniform superposition of “good” states where $\phi(\mathbf{x}) = 1$). The initial state decomposes as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|\alpha\rangle + \sin\left(\frac{\theta}{2}\right)|\beta\rangle \quad (9)$$

where the key relationship between the angle θ and the number of good states M is:

$$\sin^2\left(\frac{\theta}{2}\right) = \frac{M}{2^n} \quad (10)$$

This relationship is the core of the entire quantum counting algorithm: if we find θ , we calculate M directly from it. Tight bounds on the number of Grover iterations required have been established in [25].

B. Algorithm Description

Quantum counting applies QPE to the Grover operator G . The eigenvectors of G in the relevant subspace are $|v_\pm\rangle$ with eigenvalues $e^{\pm i\theta}$. The initial state $|\psi\rangle$ is a superposition of both eigenvectors, so QPE returns an estimate of either θ or $2\pi - \theta$, giving either M or $2^n - M$. An alternative QFT-free approach achieving the same complexity purely through Grover iterations has been proposed in [26].

The entire quantum counting procedure:

- 1) Prepare a t -qubit counting register in superposition ($H^{\otimes t}$) and an n -qubit working register in uniform superposition ($H^{\otimes n}$). Prepare the output qubit in state $|-\rangle$ for phase kickback.
- 2) For $j = 0, 1, \dots, t-1$, apply G^{2^j} controlled on the j -th qubit of the counting register.
- 3) Apply the inverse QFT $^{-1}$ to the counting register.
- 4) Measure the counting register, obtain the integer y .
- 5) Calculate the estimates. The measured integer $y \in \{0, 1, \dots, 2^t - 1\}$ from the counting register encodes the phase of the Grover operator. The angle and the number of functional states are recovered as:

$$\tilde{\theta} = \frac{2\pi y}{2^t}, \quad \tilde{M} = 2^n \cdot \sin^2\left(\frac{\tilde{\theta}}{2}\right) \quad (11)$$

The complete circuit layout is illustrated in Fig. 1.

C. Complexity

In step 2 of the algorithm, the j -th qubit of the counting register controls G^{2^j} , which requires 2^j sequential applications of the Grover operator and each application contains one oracle call. Summing over all t counting qubits, the total number of oracle calls is:

$$\sum_{j=0}^{t-1} 2^j = 2^t - 1 \quad (12)$$

To estimate the truth density with an accuracy of ε , $t \approx n/2 + \mathcal{O}(\log(1/\varepsilon))$ qubits are enough, which gives $\mathcal{O}(2^{n/2}/\varepsilon)$ oracle calls. Each oracle call costs $\mathcal{O}(s)$ elementary gates, where s is the number of classical logic gates in the Boolean netlist for ϕ (see Section VI for the detailed construction). The total gate complexity is therefore:

$$\mathcal{O}\left(s \cdot \frac{2^{n/2}}{\varepsilon}\right) \quad (13)$$

For comparison, the classical approach requires 2^n evaluations of ϕ , each in $\mathcal{O}(s)$ operations, giving a total of $\mathcal{O}(s \cdot 2^n)$. The acceleration ratio is $2^{n/2}$.

Remark 5.1 (Approximate density vs. exact count): The bound (13) concerns *approximate* estimation of the truth density d to additive accuracy ε , the quantity required for reliability analysis. Recovering the *exact* integer M is strictly stronger: consecutive admissible densities differ by $1/2^n$, so exact recovery needs $\varepsilon < 1/2^{n+1}$, hence $t \approx n$ and $\mathcal{O}(2^n)$ oracle calls. The quadratic speedup therefore applies to density (and hence reliability) estimation, *not* to exact model counting, for which we claim no speedup. The exact M reported for small n in Section VIII is a verification by-product, available only because $t \approx n$ is still affordable there.

D. Relation to Other Quantum Approaches

Several quantum methods target related counting/estimation tasks. The QFT-free approximate counting of Aaronson and Rall [26] attains the same $\mathcal{O}(2^{n/2})$ query complexity using only Grover iterations, trading the t -qubit counting register for repeated amplitude estimation – reducing width at the cost of sequential runs. Iterative QPE (IQPE) [27] similarly collapses the counting register to a single recycled qubit with adaptive feedback. Both reduce qubit width relative to QPE construction but do not change the oracle, so the netlist-to-oracle synthesis of Section VI applies to them unchanged. In the reliability domain, the quantum-Bayesian-network approach of Salfale et al. [7] uses amplitude amplification on a probabilistic graphical model rather than on a Boolean structure function; it estimates marginal probabilities directly but does not yield the truth density (functional-state count) that is our target. Our contribution is thus orthogonal: a general structure-function oracle that can be plugged into any of these estimation backends.

VI. CONSTRUCTION OF ORACLE FOR NONCOHERENT SYSTEM

The construction of an efficient quantum oracle from a compact representation of the system is a key contribution of this

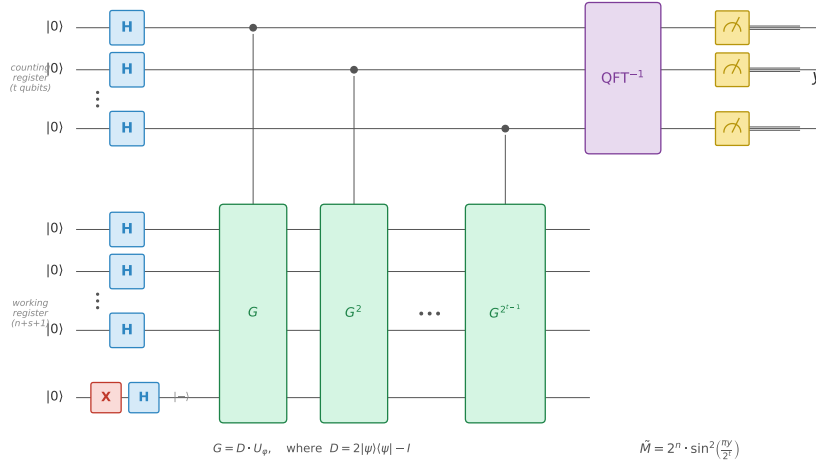


Fig. 1. Generic quantum counting circuit. The counting register (t qubits) controls powers of the Grover operator G applied to the working register. Inverse QFT and measurement yield estimate y , from which M is computed via $M = 2^n \cdot \sin^2(\pi y/2^t)$.

paper. Related work on mapping Boolean functions to quantum circuits appears in the context of quantum compilation [28], though with a different objective of circuit optimization rather than reliability analysis. The oracle must be constructed from a compact description of the system (not from an explicit truth table), otherwise its construction itself would have $\mathcal{O}(2^n)$ complexity and the quantum advantage would be negated.

A. Compact Representation

If the input was an explicit truth vector with 2^n entries, the construction of the oracle would require $\mathcal{O}(2^n)$ quantum gates, because for each entry where $\phi(\mathbf{x}) = 1$ it would be necessary to add one branch of the multi-controlled circuit. Quantum counting would then call this oracle $\mathcal{O}(2^{n/2})$ times, giving a total complexity of:

$$\mathcal{O}(2^n \cdot 2^{n/2}) = \mathcal{O}(2^{3n/2}) \quad (14)$$

This is worse than the classical for-loop with $\mathcal{O}(2^n)$. This observation is consistent with complexity-theoretic results showing that constructing quantum operators from explicit Boolean representations is #P-hard in general [29]. The cost of one oracle call is therefore decisive: from the truth table it is $\mathcal{O}(2^n)$ per call, from the compact circuit it is $\mathcal{O}(\text{poly}(n))$ per call.

B. Input Format: Netlist

The system is specified as a Boolean netlist: an ordered list of gates $G = [(\text{type}, \text{inputs}, \text{output}), \dots]$, where $\text{type} \in \{\text{AND}, \text{OR}, \text{NOT}, \text{XOR}, \text{NAND}, \text{NOR}, \text{XNOR}\}$, inputs are the names of primary inputs ($x_1, \dots, x_n$) or outputs of previous gates, and output is the unique name of the intermediate result. The last gate must have the output “out”, the value $\phi(\mathbf{x})$. We assume all gates in the netlist have at most two inputs. Gates with $k > 2$ inputs (e.g., a three-input AND) can be decomposed into $k - 1$ two-input gates, each requiring one ancilla

qubit. This decomposition is standard and does not affect the asymptotic complexity of the oracle (this is not a part of the current implementation). A netlist is essentially a Directed Acyclic Graph (DAG), which is a standard representation in circuit design.

For noncoherent systems, a netlist is a natural representation, since incoherence typically arises through negations (NOT) and non-commuting operations (XOR), which are explicitly present in the netlist. Unlike coherent systems, which consist only of AND and OR gates (corresponding to series and parallel connections), noncoherent systems require full Boolean logic.

C. Qubit Allocation

An oracle quantum circuit requires three groups of qubits:

- **Input qubits (n):** One qubit for each system component ($x_1, \dots, x_n$). These qubits must not be overwritten, Grover’s operator continues to work with them after the oracle.
- **Ancilla qubits (s):** One for each gate in the netlist. They store the intermediate results of the computation and must be returned to state $|0\rangle$ at the end.
- **Output qubit (1):** Carries the final value of $\phi(\mathbf{x})$.

The total number of oracle qubits is $n + s + 1$, where s denotes the number of gates in the Boolean netlist representing ϕ . In the standard allocation (without recycling), each gate requires exactly one ancilla qubit to store its intermediate result, so s equals both the gate count and the ancilla count.

D. Translation of Logic Gates on Quantum Gates

Each classical gate is translated into a sequence of reversible quantum gates. The output (ancilla) qubit is always initialized to $|0\rangle$ and after application contains the result of the operation. The inputs remain intact, which is a key invariant because the same wires can be read by multiple later gates.

- **AND**(a, b) $\rightarrow w$: One Toffoli gate $\text{CCX}(a, b, w)$. Toffoli flips the target qubit only if both control qubits are $|1\rangle$. Since w starts at $|0\rangle$, the result is $w = a \wedge b$. Toffoli is self-inverse ($\text{CCX} \cdot \text{CCX} = I$), which is important for uncomputation.
- **OR**(a, b) $\rightarrow w$: Uses De Morgan's law: $a \vee b = \neg(\neg a \wedge \neg b)$. Implementation: $X(a)$, $X(b)$ temporarily negate inputs; $\text{CCX}(a, b, w)$ computes $\neg a \wedge \neg b$; $X(a)$, $X(b)$ return the inputs to their original state; $X(w)$ negates the result. A total of 6 gates. The inputs are intact because $X \cdot X = I$.
- **NOT**(a) $\rightarrow w$: $X(w)$ followed by $\text{CNOT}(a, w)$. Result: $w = 1 \oplus a = \neg a$. Input a remains intact.
- **XOR**(a, b) $\rightarrow w$: $\text{CNOT}(a, w)$, $\text{CNOT}(b, w)$. Result: $w = 0 \oplus a \oplus b = a \oplus b$. Inputs intact.
- **NAND**(a, b) $\rightarrow w$: $\text{CCX}(a, b, w)$ followed by $X(w)$. Result: $w = \neg(a \wedge b)$.
- **NOR**(a, b) $\rightarrow w$: $X(a)$, $X(b)$, $\text{CCX}(a, b, w)$, $X(a)$, $X(b)$. Result: $w = \neg a \wedge \neg b = \neg(a \vee b)$. No final $X(w)$, unlike OR.
- **XNOR**(a, b) $\rightarrow w$: $\text{CNOT}(a, w)$, $\text{CNOT}(b, w)$, $X(w)$. Result: $w = \neg(a \oplus b)$.

E. Forward Pass and Uncomputation

The construction of the oracle consists of two phases, which we illustrate on the system $\phi(x_1, x_2, x_3) = (x_1 \wedge x_2) \oplus \neg x_3$ with a netlist:

- Gate 0: $\text{AND}(x_1, x_2) \rightarrow w_1$
- Gate 1: $\text{NOT}(x_3) \rightarrow w_2$
- Gate 2: $\text{XOR}(w_1, w_2) \rightarrow \text{out}$

Allocation of qubits: $q_0 = x_1$, $q_1 = x_2$, $q_2 = x_3$, $q_3 = w_1$, $q_4 = w_2$, $q_5 = \text{out}$. A total of 6 qubits. The corresponding circuit is shown in Fig. 2.

Phase 1 — Forward pass. We go through the netlist from gate 0 to gate 2. For each gate we apply its quantum equivalent:

- Gate 0 (AND): $\text{CCX}(q_0, q_1, q_3) \rightarrow q_3 = x_1 \wedge x_2$
- Gate 1 (NOT): $X(q_4)$, $\text{CNOT}(q_2, q_4) \rightarrow q_4 = \neg x_3$
- Gate 2 (XOR): $\text{CNOT}(q_3, q_5)$, $\text{CNOT}(q_4, q_5) \rightarrow q_5 = w_1 \oplus w_2 = \phi(\mathbf{x})$

After the forward pass, q_5 contains $\phi(\mathbf{x})$, but q_3, q_4 are entangled with q_0, q_1, q_2 . If we left the ancillas in this state, they would carry information about $\mathbf{x}$ and Grover's diffuser would not work correctly, since the geometric interpretation of the rotation in the 2D subspace would be disturbed.

Phase 2 — Uncomputation. We apply the same gates in reverse order (i.e., for a forward sequence U_1, U_2, U_3 , the uncomputation must be U_3, U_2, U_1), skipping gate 2 (it writes to out, which we want to preserve). Gates 1, 0 are repeated in reverse:

- Reverse gate 1 (NOT): $\text{CNOT}(q_2, q_4)$, $X(q_4) \rightarrow q_4 = 0$
- Reverse gate 0 (AND): $\text{CCX}(q_0, q_1, q_3) \rightarrow q_3 = 0$

Result: q_0, q_1, q_2 intact, $q_3 = q_4 = 0$, $q_5 = \phi(\mathbf{x})$. The transformation $|\mathbf{x}\rangle |00\rangle |0\rangle \rightarrow |\mathbf{x}\rangle |00\rangle |\phi(\mathbf{x})\rangle$ is complete.

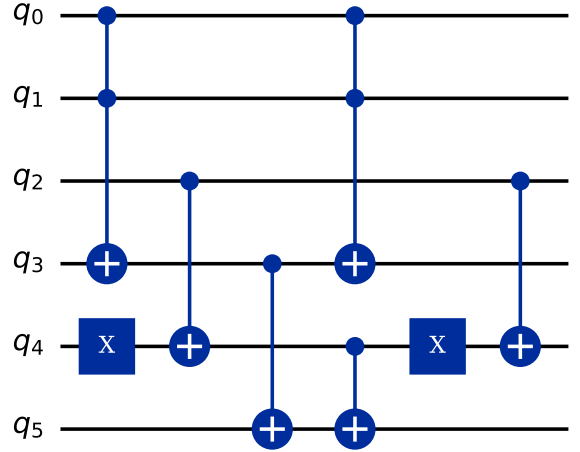


Fig. 2. Oracle circuit for the system $\phi = (x_1 \wedge x_2) \oplus \neg x_3$, showing the forward pass (left) and uncomputation (right). Input qubits q_0 – q_2 remain intact; ancilla qubits q_3 – q_4 are returned to $|0\rangle$; output qubit q_5 carries $\phi(\mathbf{x})$.

All used quantum gates (Toffoli, CNOT, X) are self-inverse: $U \cdot U = I$. Therefore, repeating the same sequence of gates inverts the computation and returns the ancilla to $|0\rangle$ independently of the value of $\mathbf{x}$. The ancilla qubit after uncomputation is not entangled with any other qubit; it is in the factorized state $|0\rangle$, which is equivalent to a fresh, unused qubit. The cost of one oracle call is $O(2s) = O(s)$, where s is the number of gates.

F. Recycling of Ancilla Qubits

Standard allocation assigns one ancilla per gate permanently. However, many intermediate wires become dead after their last use as input to a subsequent gate. Through lifetime analysis, we determine when each wire is last read, uncompute it at that point (returning its qubit to $|0\rangle$), and release the qubit into a reuse pool.

For example, in the system $\phi = ((x_1 \wedge x_2) \vee (x_3 \wedge x_4)) \wedge \neg x_5$, wires w_1 and w_2 are dead after gate 2 (OR). Their qubits are uncomputed and reused by gates 3 and 4, reducing the total from 10 to 8 qubits (20% saving). Figure 3 shows savings of 0–46% across all tested systems, with the largest reductions on SOP-derived netlists where many AND terms are consumed once by the OR tree.

Verification confirmed that the recycled oracle produces identical truth tables to the standard one. However, we observed that our recycled construction is incompatible with the *controlled-power schedule* used by QPE. In the recycled oracle, uncomputation of dead wires is interleaved with forward gates, creating sequences where a qubit is freed and immediately reused. When the entire circuit is controlled on a counting register qubit, the control condition may prevent uncomputation while still executing the subsequent gate on the same qubit, leaving it in an incorrect state. The result is that

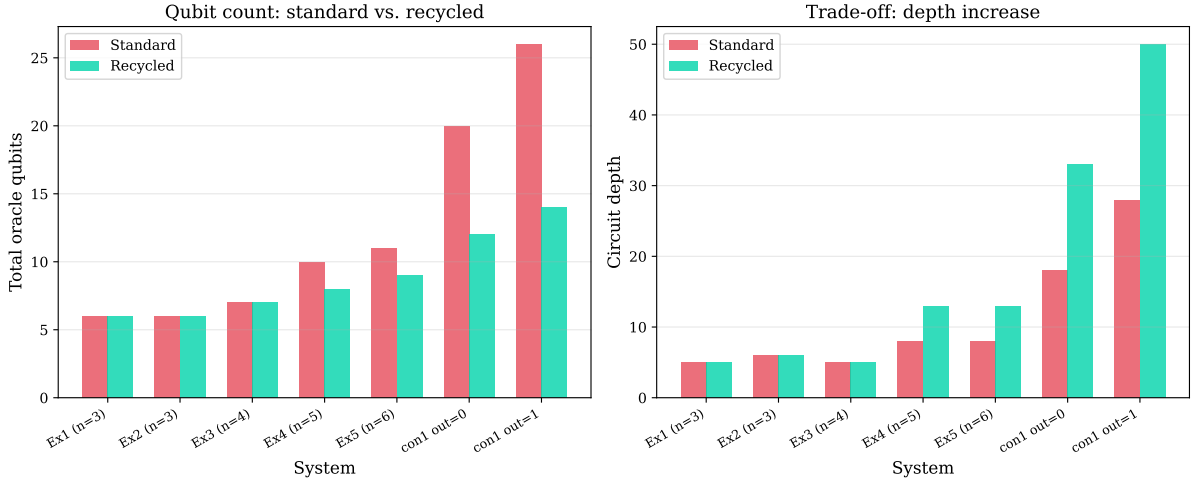


Fig. 3. Comparison of standard vs. recycled ancilla allocation. Left: qubit count reduction (18–46% for larger systems). Right: circuit depth increase — the trade-off between qubit savings and deeper circuits.

QPE with a recycled oracle produces a uniform measurement distribution instead of concentrated peaks, and the algorithm fails. We stress that this is a limitation of the specific naive schedule (controlling each interleaved free/reuse sequence as one block), not a fundamental property of quantum counting: block-wise recycling or a compute–uncompute–recompute pattern (Future Work) should restore compatibility. Therefore, recycling is applicable only for standalone oracle verification and qubit count comparison; quantum counting itself requires the standard allocation with uninterrupted forward pass and uncomputation sequences.

G. PLA and LGSynth’91

For validation on standard benchmarks, a parser for the PLA format (Espresso), used in the LGSynth’91 benchmark suite [30], was implemented. PLA defines a Boolean function as a sum of products (SOP/DNF). Each product term is translated into an AND tree of its literals (with NOT for negated inputs “0”, skipping do not-care inputs “-”), and the entire function is an OR tree of all terms.

For multi-output PLA files, each output is processed independently. The parser correctly interprets the output plane including the symbol “~” (do not care in the output plane) according to the Espresso format specification [31].

An important observation: direct SOP-to-netlist conversion generates netlists whose size is proportional to the number of product terms, not the number of inputs. The benchmark `xor5` with 16 terms produces 119 gates (and thus 119 ancilla qubits), which is unmanageable for a simulator. Logic minimization prior to oracle synthesis (e.g., algebraic factorization) would significantly reduce this size.

VII. PARAMETER AND ERROR ANALYSIS

A. Choosing the Number of QPE Qubits

The number of qubits t in the QPE counting register directly determines the accuracy of the estimate. With t qubits, the

resolution in the angle is $\Delta\theta = 2\pi/2^t$ and the error in the truth density is bounded by:

$$|\tilde{d} - d| \leq \frac{\pi}{2^t} \quad (15)$$

To accurately estimate an integer M would require $t \approx n$ qubits ($\mathcal{O}(1/2^n)$ resolution), giving $\mathcal{O}(2^n)$ oracle calls and negating the quantum advantage. However, $t \approx n/2 + \log_2(\pi/\varepsilon)$ qubits are sufficient to estimate the truth density with accuracy ε , which preserves the quadratic speedup.

In our experiments, $t \geq n/2 + 2$ was sufficient to round $\tilde{M}$ to the correct integer for the small systems tested ($n \leq 6$). This is *not* a general guarantee (cf. Remark 5.1): the rounding succeeds only because at these sizes the discretisation scale $\pi/2^t$ is already below the inter-density gap $1/2^n$. As n grows the two scales cross, exact rounding then demands $t \approx n$, and only the density estimate retains the speedup.

A special case are systems with extreme truth density, for example, a purely serial system (AND, $M = 1$) or a purely parallel system (OR, $M = 2^n - 1$). In both cases the angle is $\theta \approx 2/\sqrt{2^n}$, which for large n is extremely small, close to 0. It is difficult to distinguish from zero by QPE, and t grows to the order of n , not $n/2$.

B. Sources of Error

The QPE discretization error is an inherent error arising from the finite resolution of the counting register. For most values of M , the angle θ is an irrational number that cannot be represented exactly by t bits. The error in M propagates nonlinearly through the relationship $M = 2^n \sin^2(\theta/2)$ and the sensitivity is largest for $M \approx 2^{n-1}$ ($d \approx 0.5$) and smallest for M close to 0 or 2^n .

The statistical error from the measurement arises from the probabilistic nature of QPE. A single measurement returns the correct closest value with probability at least $4/\pi^2 \approx 0.405$ [32]. Adding m extra qubits increases the probability to $1 - 1/(2^{m+1} - 2)$.

The ambiguity of M vs. $2^n - M$ arises from the fact that $|\psi\rangle$ is a superposition of two eigenvectors of the Grover operator with phases θ and $-\theta$. QPE can return an estimate of either M or $2^n - M$. In post-processing, QPE yields the *unordered* pair $\{\tilde{M}, N - \tilde{M}\}$; the two values are indistinguishable from the phase alone. We therefore report both candidates natively rather than silently collapsing to $\min(\tilde{M}, N - \tilde{M})$, which coincides with the true count only when $M \leq N/2$ and is wrong when $M > N/2$.

Selecting the correct branch requires one bit of side information: whether $d < 1/2$ or $d > 1/2$. This bit can be obtained by classical Monte-Carlo sampling of ϕ : with $S = \mathcal{O}(\gamma^{-2} \log(1/\delta))$ random evaluations, where $\gamma = |d - 1/2|$, the side of $1/2$ is decided correctly with confidence $1 - \delta$. The procedure is cheap when d is bounded away from $1/2$ and degrades as $\gamma \rightarrow 0$; for $d = 1/2$ the branch is genuinely undetermined, which we leave as an open problem. Importantly, the absolute reliability error incurred by a wrong branch is $|1 - 2d|$: it is largest exactly where γ is largest (and the bit is easiest to obtain) and vanishes as $d \rightarrow 1/2$ (where the bit is hardest), so the *reliability* estimate is robust even though the *count* semantics remain ambiguous at $d = 1/2$. The two-eigenvector origin of this ambiguity is well known [33].

Median Amplification: To increase reliability, the entire quantum counting procedure is run R times independently and the median of the R estimates is taken. For a single run success probability $p > 1/2$ and $R = \mathcal{O}(\log(1/\delta))$ runs, the total failure probability is less than δ .

C. Diffuser Global Phase

A subtle but critical technical issue was identified during implementation. The standard implementation of the diffuser $D = 2|\psi\rangle\langle\psi| - I$ through the sequence H, X , multi-controlled Z, X, H actually realizes $-(2|\psi\rangle\langle\psi| - I)$. The extra negative sign is a global phase that has no effect on the measurement probabilities in the Grover search itself.

However, in QPE, the global phase becomes the relative phase between the control and working registers. The Grover operator G with an extra -1 has eigenvalues $-e^{\pm i\theta} = e^{i(\pi \pm \theta)}$ instead of $e^{\pm i\theta}$. QPE then estimates the phase shifted by π , giving an incorrect M .

The compensation is straightforward: a global phase π is added to the Grover operator (in Qiskit: `qc.global_phase += np.pi`), turning $-G$ into G . This is an important insight for reproducibility, as many quantum counting implementations overlook this detail.

D. Limitations of Simulation

Simulating a quantum circuit on a classical computer requires storing a state vector of size $2^{t+n+s+1}$ complex numbers (16 bytes each). For a quantum counting circuit with a total number of qubits $t + n + s + 1$, this is an exponential memory requirement. In practice, the simulation is feasible for a total number of qubits up to about 25–28 on common hardware (32 GB RAM).

This is not a limitation of the algorithm itself, but of the simulator. On a real quantum computer, quantum counting would run with $\mathcal{O}(\text{poly}(n) \cdot 2^{n/2})$ complexity without exponential memory. Current quantum hardware (e.g., IBM Eagle, 127 qubits) has a sufficient number of qubits, but gate error rates ($\sim 10^{-2}$ for two-qubit gates) and decoherence prevent meaningful computation without error correction for non-trivial systems.

VIII. IMPLEMENTATION AND EXPERIMENTS

This section presents the implementation details and experimental evaluation. All experiments were conducted using the Qiskit statevector simulator on a workstation with 32 GB RAM. The complete implementation is available on GitLab.

A. Implementation

The algorithm was implemented in Python using the Qiskit framework (version 2.3.1) and the Qiskit Aer simulator [34]. The implementation consists of five modules:

- `oracle.py` — `OracleSynthesizer` class converts a netlist to a quantum oracle. It supports two modes: standard (one ancilla qubit per gate) and optimized with ancilla qubit recycling. It implements gate translation, forward pass, and uncomputation.
- `quantum_counting.py` — the `QuantumCounting` class implements a complete quantum counting algorithm, including construction of the Grover operator (with global phase correction), QPE, inverse QFT, post-processing, and median strategy.
- `experiments.py` — extended set of experiments: accuracy vs. t for all systems, QPE histograms, oracle call comparison, ancilla recycling, verification, circuit diagram and graph generation.
- `pla_parser.py` — parser for the PLA format (Espresso) with conversion to netlist. Supports multi-output, don't-care, and custom input names.
- `pla_benchmark.py` — runs the pipeline on LGSynth'91 benchmarks with a pure classical evaluator.

B. Reference Noncoherent Systems

The algorithm was verified on five noncoherent systems with increasing complexity. In all cases, quantum counting returned the correct value of M , verified by classical enumeration of the entire state space. The results can be seen in Table I.

TABLE I
VERIFICATION RESULTS ON REFERENCE NONCOHERENT SYSTEMS

System ϕ	n	s	Qub.	t	M	d
$(x_1 \wedge x_2) \oplus \neg x_3$	3	3	11	5	4	.500
$(x_1 \vee x_2) \wedge \neg x_3$	3	3	11	5	3	.375
$(x_1 \oplus x_2) \wedge (x_3 \oplus x_4)$	4	3	12	5	4	.250
$((x_1 \wedge x_2) \vee (x_3 \wedge x_4)) \wedge \neg x_5$	5	5	16	6	7	.219
$((x_1 \wedge x_2) \vee (x_3 \wedge x_4)) \wedge \neg (x_5 \wedge x_6)$	6	5	18	7	21	.328

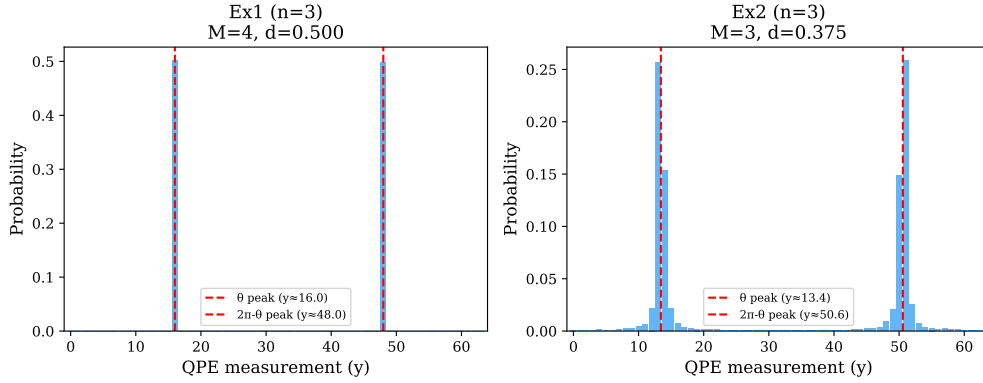


Fig. 4. QPE measurement distributions for two reference systems ($t = 6, 8192$ shots). Peaks concentrate around values of y corresponding to the correct M , confirming the algorithm's correctness.

C. Accuracy Analysis

For all reference systems, the dependence of accuracy on the number of QPE qubits t was analyzed. The results are summarized in Table II. The minimum t required for a correct estimate of M varies across systems.

TABLE II
MINIMUM QPE QUBITS FOR CORRECT M ESTIMATION

System ϕ	n	M	d	Min. t
$(x_1 \wedge x_2) \oplus \neg x_3$	3	4	.500	3
$(x_1 \vee x_2) \wedge \neg x_3$	3	3	.375	4
$(x_1 \oplus x_2) \wedge (x_3 \oplus x_4)$	4	4	.250	3
$((x_1 \wedge x_2) \vee (x_3 \wedge x_4)) \wedge \neg x_5$	5	7	.219	3

A detailed analysis for the system $(x_1 \vee x_2) \wedge \neg x_3$ ($M = 3$, $d = 0.375$) is shown in Table III:

TABLE III
ACCURACY VS. t FOR $(x_1 \vee x_2) \wedge \neg x_3$

t	Oracle calls	$\tilde{M}$	$ \Delta d $	Correct?
3	7	4	0.125	No
4	15	3	0.000	Yes
5	31	3	0.000	Yes
6	63	3	0.000	Yes
8	255	3	0.000	Yes

The results confirm that for noncoherent systems with moderate truth density, $t \approx n/2 + 2$ is sufficient for a correct estimate of M at the tested sizes ($n \leq 6$); this does not extrapolate to large n , where exact M requires $t \approx n$ (Remark 5.1). For the system with $d = 0.5$ (Ex. 1), even $t = 3$ suffices, because $\theta = \pi/2$ is exactly representable. For systems with d further from 0.5, a higher t is needed due to the irrationality of the angle θ .

The first system ($d = 0.5$) is the easiest case, since $\theta = \pi/2$ is a “nice” value exactly representable even with small t . The last system ($d = 0.219$) is interesting in that $t = 3$ suffices despite the lower d , because $M = 7$ is sufficiently far from $N/2 = 16$ for QPE to distinguish both peaks.

D. Comparison of Oracle Call Counts

The main metric of efficiency is the number of oracle calls: the classical algorithm requires 2^n calls, while quantum counting requires $\mathcal{O}(2^{n/2})$, as shown in Table IV:

TABLE IV
ORACLE CALL COMPARISON: CLASSICAL VS. QUANTUM COUNTING

n	Classical (2^n)	Quantum ($\sim 2^{n/2}$)	Speedup
10	1 024	32	$32\times$
20	1 048 576	1 024	$1\,024\times$
30	$\sim 10^9$	$\sim 32\,768$	$\sim 32\,768\times$
40	$\sim 10^{12}$	$\sim 10^6$	$\sim 10^6\times$
50	$\sim 10^{15}$	$\sim 3.4 \times 10^7$	$\sim 3.4 \times 10^7\times$
100	$\sim 10^{30}$	$\sim 10^{15}$	$\sim 10^{15}\times$

These values are theoretical, because on a simulator it is not possible to demonstrate a time speedup, since the simulation itself is exponential. However, the oracle call metric can be directly computed and compared.

E. LGSynth'91 Benchmarks

The PLA format parser was successfully applied to three benchmarks from the LGSynth'91 suite. The results are presented in Table V.

Classical verification using a pure Python evaluator (independent of the Qiskit simulator) confirmed the correctness of the parser and netlist translation on all six functions. The results are consistent with expected values — for example, `xor5` is a 5-input parity function with $d = 0.5$, and `rd53` output 0 is an “at least 4 out of 5” function with $M = \binom{5}{4} + \binom{5}{5} = 6$.

Quantum simulation on the LGSynth'91 benchmarks was constrained by the large circuit sizes generated by direct SOP-to-netlist conversion. Even the smallest benchmark suitable for quantum counting, `con1` output 0 (7 inputs, 4 terms, 13 gates), requires 20 oracle qubits in standard mode or 12 with recycling. Combined with $t = 8$ QPE qubits needed for sufficient resolution (since $M = 68$ is close to $N/2 = 64$), the total circuit comprises 28 qubits and $2^8 - 1 = 255$ controlled Grover iterations, each containing the full oracle.

TABLE V
LGSYNTH'91 BENCHMARK RESULTS

Benchmark	n	Outputs	Terms	Gates	Std. qubits	Recycl. qubits	M	d
xor5 output 0	5	1	16	119	124	—	16	0.500
rd53 output 0	5	3	5	19	24	—	6	0.188
rd53 output 1	5	3	16	119	124	—	16	0.500
rd53 output 2	5	3	11	65	70	—	20	0.625
con1 output 0	7	2	4	13	20	12	68	0.531
con1 output 1	7	2	5	19	26	—	88	0.688

While this is within the memory capacity of a 32 GB workstation, the transpilation and simulation time exceeds practical limits (estimated >30 minutes per run). For the remaining benchmarks, the SOP representation produces netlists with 19–119 gates (24–124 oracle qubits), making simulation infeasible without prior logic minimization.

This limitation is not inherent to the algorithm but to the combination of (a) the SOP representation, which generates $\mathcal{O}(k)$ gates per product term (where k is the number of terms), and (b) the exponential cost of classical simulation. On actual quantum hardware, con1 output 0 would require only 28 physical qubits — well within the capacity of current processors (e.g., IBM Eagle with 127 qubits). The practical path to quantum simulation of larger PLA benchmarks is logic minimization prior to oracle synthesis: algebraic factorization or multi-level optimization (e.g., via tools such as ABC [31]) could reduce the xor5 netlist from 119 gates to approximately 5–10, making quantum simulation feasible even on a laptop.

IX. CONCLUSION

In this paper, we proposed and implemented an application of the quantum counting algorithm to the computation of the truth density of noncoherent binary-state systems. The central contribution is a systematic procedure for synthesising a quantum oracle directly from a compact Boolean netlist, covering gate translation, ancilla management, and uncomputation, together with an explanation of why a compact representation is essential for preserving the quantum ad-

vantage. The complete pipeline was implemented in Qiskit and verified on five reference noncoherent systems with 3-6 components. Along the way we analysed ancilla recycling, which reduced the qubit count by 18–46% on the tested systems but proved incompatible with the standard controlled-power QPE schedule, an implementation-level obstruction that we document rather than dismiss. We further implemented a parser for the PLA (Espresso) format that extends the method to the LGSynth'91 benchmarks, and we identified and corrected a subtle global-phase error in the diffuser that would otherwise bias the QPE result.

Considering a system of n components, quantum counting attains a complexity of $\mathcal{O}(\text{poly}(n) \cdot 2^{n/2})$ oracle calls against the classical $\mathcal{O}(\text{poly}(n) \cdot 2^n)$, a quadratic speedup in the exponential factor. This advantage is most relevant for noncoherent systems, where no classical method guarantees a subexponential worst-case bound and where symbolic techniques such as BDDs can still degrade to exponential size. From a practical standpoint, reducing the number of oracle calls from $\mathcal{O}(2^n)$ to $\mathcal{O}(2^{n/2})$ moves the topological analysis of large noncoherent systems from the classically infeasible regime toward problems that should become tractable on future fault-tolerant quantum hardware.

Future Work

We identify the following directions for further research:

Logic minimization prior to oracle synthesis. As demonstrated by the PLA benchmarks, direct SOP conversion

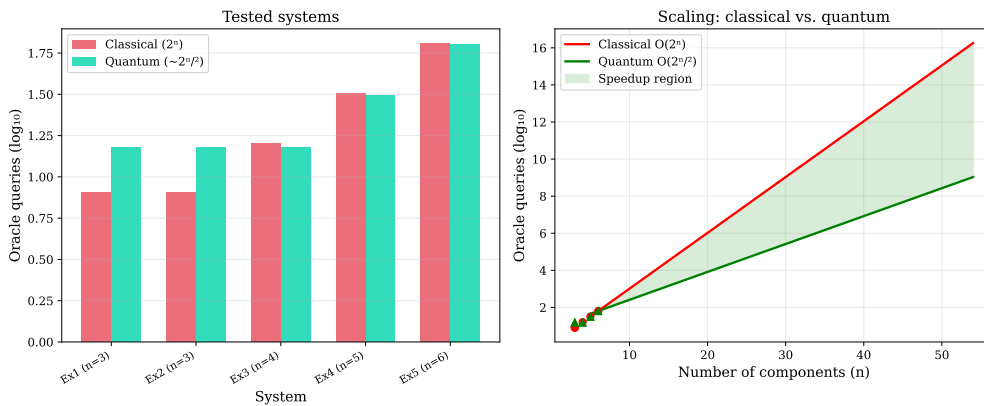


Fig. 5. Oracle call comparison on logarithmic scale: classical enumeration (2^n) vs. quantum counting ($\sim 2^{n/2}$). The gap grows exponentially with n , reaching a factor of $\sim 10^6$ at $n = 40$.

generates large netlists. Algebraic factorization, technology-independent optimization (e.g., via the ABC tool from Berkeley), or BDD-based synthesis would significantly reduce oracle size and enable simulation of larger systems.

Compatible ancilla qubit recycling. The identified incompatibility with QPE could be addressed by modifying the recycling strategy. For example, by dividing the circuit into “blocks” where recycling is performed only between blocks, not inside controlled operations. Alternatively, a compute-uncompute-recompute pattern could be used, where the entire block is uncomputed and recomputed with a new qubit assignment.

Iterative QPE (IQPE). Instead of a t -qubit counting register, a single qubit can be used and the measurement repeated t times with adaptive feedback [27]. This dramatically reduces the total qubit count at the cost of multiple sequential runs. A related QFT-free alternative that achieves the same $\mathcal{O}(2^{n/2})$ complexity using only Grover iterations without QPE has been proposed by Aaronson and Rall [26].

Testing on real quantum hardware. Current hardware (e.g., IBM Eagle, 127 qubits) has a sufficient qubit count, but gate error rates and decoherence currently prevent meaningful computation without error correction. Running quantum counting on real hardware, even with expected failure due to noise, would be a valuable experimental contribution documenting the gap between theory and practice.

GENERATIVE AI DISCLOSURE STATEMENT

During the preparation of this work the authors used Claude Opus 4.6 in order to improve the language, code review and LaTeX formatting. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

X. ACKNOWLEDGMENT

We would like to thank to our colleague Michal Mrena for proposing several interesting ideas during preparation of this paper.

REFERENCES

- [1] M. Rausand, A. Barros, and A. Høyland, *System Reliability Theory*. Bognor Regis, UK: John Wiley & Sons, Ltd., 3 ed., 2020.
- [2] E. Zaitseva and V. Levashenko, “Investigation multi-state system reliability by structure function,” in *2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX '07)*, (Szklarska Poreba, Poland), pp. 81–90, 2007.
- [3] M. Kvassay and E. Zaitseva, “Topological Analysis of Multi-state Systems Based on Direct Partial Logic Derivatives,” in *Recent Advances in Multi-state Systems Reliability: Theory and Applications* (A. Lisnianski, I. Frenkel, and A. Karagrigoriou, eds.), pp. 265–281, Cham: Springer International Publishing, 2018.
- [4] E. Zaitseva, P. Sedlacek, and V. Levashenko, “Importance analysis of non-coherent multi-state system,” *Reliability Engineering & System Safety*, vol. 266, p. 111618, 2026.
- [5] I. Gioda, D. Caputo, E. Fadda, D. Manerba, B. Silva Fernández, and R. Tadei, “Solving assignment problems via quantum computing: a case-study in train seating arrangement,” in *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pp. 217–220, 2021.
- [6] M. Yazdi, “Application of Quantum Computing in Reliability Analysis,” in *Advances in Computational Mathematics for Industrial System Reliability and Maintainability* (M. Yazdi, ed.), pp. 139–154, Cham: Springer Nature Switzerland, 2024.
- [7] S. Salfale, G. Dhakad, S. Bara, Abhinav Krishnan T. K., and I. Hazra, “Quantum Bayesian Networks for Reliability Analysis of Unmanned Systems,” in *Proceedings of the ASME International Mechanical Engineering Congress and Exposition India (IMECE-INDIA2025)*, American Society of Mechanical Engineers Digital Collection, Dec. 2025.
- [8] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, (Philadelphia, Pennsylvania, United States), pp. 212–219, ACM Press, 1996.
- [9] A. Rauzy, “Mathematical foundations of minimal cutsets,” *IEEE Transactions on Reliability*, vol. 50, pp. 389–396, Dec. 2001.
- [10] M. Kvassay, V. Levashenko, and E. Zaitseva, “Analysis of minimal cut and path sets based on direct partial Boolean derivatives,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 230, pp. 147–161, Apr. 2016.
- [11] M. O. Locks, “A Minimizing Algorithm for Sum of Disjoint Products,” *IEEE Transactions on Reliability*, vol. R-36, pp. 445–453, Oct. 1987.
- [12] M. O. Locks, “Recursive Disjoint Products, Inclusion-Exclusion, and Min-Cut Approximations,” *IEEE Transactions on Reliability*, vol. R-29, pp. 368–371, Dec. 1980.
- [13] Y. Crama and P. L. Hammer, *Boolean Functions: Theory, Algorithms, and Applications*. Cambridge University Press, May 2011. Google-Books-ID: 3KnyKpw_pbUC.
- [14] M. Kvassay, E. Zaitseva, V. Levashenko, and J. Kostolny, “Reliability Analysis of Multiple-Outputs Logic Circuits Based on Structure Function Approach,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, pp. 398–411, Mar. 2017.
- [15] W. G. Schneeweiss, “A short Boolean derivation of mean failure frequency for any (also non-coherent) system,” *Reliability Engineering & System Safety*, vol. 94, pp. 1363–1367, Aug. 2009.
- [16] S. Upadhyaya and H. Pham, “Analysis of noncoherent systems and an architecture for the computation of the system reliability,” *IEEE Transactions on Computers*, vol. 42, pp. 484–493, Apr. 1993.
- [17] L. G. Valiant, “The complexity of enumeration and reliability problems,” *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.
- [18] R. E. Bryant, “Graph-based algorithms for boolean function manipulation,” *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.
- [19] J. Kostolny, M. Kvassay, and S. Kovalik, “Reliability Analysis of Non-coherent Systems by Logical Differential Calculus and Binary Decision Diagrams,” *Communications - Scientific letters of the University of Zilina*, vol. 16, pp. 114–120, Feb. 2014.
- [20] E. Zaitseva, V. Levashenko, and J. Kostolny, “Importance analysis based on logical differential calculus and Binary Decision Diagram,” *Reliability Engineering & System Safety*, vol. 138, pp. 135–144, June 2015.
- [21] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Dec. 2010.
- [22] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, “Elementary gates for quantum computation,” *Physical Review A*, vol. 52, pp. 3457–3467, Nov. 1995.
- [23] G. G. Fogel, B. Baran, and M. Villagra, “Comparison of two types of quantum oracles based on grover’s adaptative search algorithm for multiobjective optimization problems,” in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 421–428, IEEE, 2017.
- [24] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, “Quantum algorithms revisited,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 454, pp. 339–354, Jan. 1998.
- [25] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, “Tight Bounds on Quantum Searching,” *Fortschritte der Physik*, vol. 46, no. 4-5, pp. 493–505, 1998.
- [26] S. Aaronson and P. Rall, “Quantum approximate counting, simplified,” in *2020 Symposium on Simplicity in Algorithms (SOSA)*, pp. 24–32, 2020.
- [27] C. R. Wie, “Simpler Quantum Counting,” *Quantum Information and Computation*, vol. 19, no. 11-12, 2019.

- [28] M. Soeken, G. Meuli, B. Schmitt, F. Mozafari, H. Riener, and G. De Micheli, "Boolean satisfiability in quantum compilation," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, p. 20190161, Dec. 2019.
- [29] S. Hadfield, "On the Representation of Boolean and Real Functions as Hamiltonians for Quantum Computing," *ACM Transactions on Quantum Computing*, vol. 2, pp. 18:1–18:21, Dec. 2021.
- [30] S. Yang, *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC) Research Triangle Park, NC, USA, 1991.
- [31] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic minimization algorithms for VLSI synthesis*, vol. 2. Springer Science & Business Media, 1984.
- [32] Z. Diao, C. Huang, and K. Wang, "Quantum Counting: Algorithm and Error Distribution," *Acta Applicandae Mathematicae*, vol. 118, pp. 147–159, Apr. 2012.
- [33] G. Brassard, P. Høyer, and A. Tapp, "Quantum counting," in *Automata, Languages and Programming* (K. G. Larsen, S. Skyum, and G. Winskel, eds.), (Berlin, Heidelberg), pp. 820–831, Springer, 1998.
- [34] IBM Quantum, "Introduction to Qiskit." <https://quantum.cloud.ibm.com/docs/en/guides/quick-start>, 2026. Accessed: Apr. 13, 2026.