Low-depth Construction of Grover Oracles from Fully Functional Quantum Circuits

Jiaqi Gu, Behzad Abdolmaleki
School of Computer Science, University of Sheffield, UK
jgu35@sheffield.ac.uk, behzad.abdolmaleki@sheffield.ac.uk

***Abstract*—Grover oracle is the core component of the Grover search algorithm. Instead of constructing a Grover oracle from scratch, we consider a common practice that how to construct a Grover oracle based on an existing fully functional quantum circuits (FFQCs). A FFQC typically performs computations for a primary target and includes ancillas (qubits for intermediate storage) restoration. Although these circuits can be directly integrated into an oracle, we find that this inevitably introduces circuit redundancy. To address this, we propose a low-depth transformation method that can convert an existing FFQC into a low-depth Grover oracle. Additionally, our method can further reduce the width while retaining the previously achieved low depth.**

***Index Terms*—Grover search, quantum circuit optimization, cryptography**

I. Introduction

Grover search is a quantum exhaustive search algorithm [1]. For a task with a search space of size N, it can find the target in $O(\sqrt{N})$ iterations, whereas a classical algorithm would require $O(N)$ steps. Although some works [2], [3] have shown that, with suitable parallelization, one can reduce the number of iterations, the exact performance of Grover search has been proven optimal [4]. That is, its theoretical lower bound $\Omega(\sqrt{N})$ cannot be broken through. No matter how it is parallelized, one must perform $\Omega(\sqrt{N})$ Grover iterations sequentially. A Grover iteration comprises a Grover oracle, which marks the targets, and a Grover diffusion operator, which amplifies the measurement probability of the targets. Since the diffusion operator is universal, constructing the Grover oracle has been extensively researched across many areas. Grover search is sensitive to depth growth in a Grover oracle. Any increase in depth of the oracle gets magnified by a factor of $\Omega(\sqrt{N})$. Therefore, reducing the depth of the Grover oracle is a critical task. We consider the case that the oracle is constructed using existing quantum circuits. Such circuits are often designed for other tasks, and directly using them to construct the Grover oracle can result in significant redundancy.

A. Our Results

Overall, our method to construct a Grover oracle based on existing fully functional quantum circuits (FFQCs) proceeds in two sequential steps:

1. Under basis $\{|0\rangle, |1\rangle\}$, we can remove (all) ancilla uncomputation circuits in FFQCs, trading off increased width to achieve a lower depth.
2. Reintroduce parallelizable ancilla uncomputation circuits to reduce the width while preserving the overall depth unchanged.

To justify the reasonableness of our method, we first discuss why ancilla uncomputation components become redundant in the Grover oracle, in Section III. We later discuss the outcomes of all possible scenarios in Section IV. In Section V, we show how to apply method to the quantum circuits of AES.

B. Technical Overview

We briefly summarize how uncomputation (on both main qubits and ancillas) is introduced and why it becomes redundant.

How to Locate Targets. Grover search was introduced as an algorithm for database search [1]. It follows an oracle-query pattern. The Grover oracle applies phase inversion to all target states that satisfy the condition, where we say it marks the targets. We typically formalize the condition as a Boolean function $f : \{0,1\}^* \to \{0,1\}$. Given a query string y and a function $F : \{0,1\}^{|x|} \to \{0,1\}^{|y|}$, we wish to find any x such that $F(x) = y$. We then define $f(x) = 1$ if and only if $F(x) = y$. There are two common ways to do the marking. One is to use CNOT gates with an ancilla prepared in the $|-\rangle$ state: one computes $|f(x)\rangle$ and then performs a phase kick-back via CNOT$|f(x), -\rangle$, thereby inverting the phase of the target states. The other is to apply a $(|y|-1)$-controlled CZ gate (a $(|y|)$-controlled CX gate directly on $|F(x)\rangle$, w.r.t. $|F(x), -\rangle$). Regardless of which method we use, it is necessary to compute $F(x)$. In Section III-B, we use the first method for illustration to explicitly demonstrate that the condition is satisfied.

The Necessity of Uncomputation in Grover Oracle. As noted above, computing $F(x)$ is unavoidable. One can either store $|F(x)\rangle$ in-place (i.e., overwrite the original $|x\rangle$) or out-of-place (in independent qubits). In a single iteration, the goal of Grover oracle is to mark the target $|x\rangle$ and then hand it to the diffusion operator for phase amplification. If $F(x)$ is computed and stored in-place, we must uncompute $F(x)$ to restore the original x before the next iteration (or final measurement). If $F(x)$ is stored out-of-place in other qubits, ideally, we are not required to uncompute it. However, if we never uncompute $F(x)$ or restore ancillas, we would need to allocate fresh qubits on each of the $O(\sqrt{N})$ iterations. For a search space of size N, this leads to an $O(\sqrt{N})$ growth in ancilla count (whereas for $|x\rangle$ we only require $O(\log N)$ qubits). Therefore, the Grover

oracle usually adopts the computation-mark-uncomputation pattern, as shown in Section III-B, where the uncomputation also refers to ancilla restoration.

The Way to Implement Uncomputation. A straightforward way to implement uncomputation is to perform the inverse of every gate in the computation circuit in the reverse order. When all gates in the computation circuit are Hermitian (and thus self-inverse), this produces a perfectly symmetric oracle structure. Alternatively, one can measure a qubit to obtain a classical bit c to decide the next process. For example, we can apply an X^c operation to an ancilla to reset it to $|0\rangle$. However, the measurement-based method comes with several limitations. Most critically, if the measured ancilla is entangled with the main qubits, the measurement induces unwanted collapse. There is a hybrid one which we call *measurement-and-reset hybrid method*. However, there is a controversy that: although it can reduce circuit depth, the efficiency drawbacks of the semiclassical operations it relies on may be even more severe. For this reason, we focus on implementing uncomputation using only quantum gates. Consider the computation-mark-uncomputation oracle pattern. The right-hand-side uncomputation of a Grover oracle functionally overlaps with the local uncomputation in the left-hand-side computation, because the computation part (literally, $F(x)$) is usually built directly from some $F(x)$ circuit for general use. In Section III, we discuss in detail why such redundancy can be introduced accidentally and lead to depth increase.

II. Preliminary

A. Grover search

In the following, we summarize the Grover search algorithm at a high level. First, one prepares an n-qubit register in the uniform superposition $|\mathbf{\Psi}\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$, where x are binary digits from 0 to $2^n - 1$. Second, one should form the Grover oracle O_f for a specific function f, and any x that satisfies $f(x) = 1$ is our target state. O_f will mark all target state $|x\rangle$ by inverting the phase $O_f |x\rangle = -|x\rangle$, and all phases of others remain the same. Next, one could apply the Grover diffusion operator D_n on $O_f |\mathbf{\Psi}\rangle$ to amplify the phases of the marked states. To make the target states stand out in the final measurement, we should execute Grover iteration on initial state $|\mathbf{\Psi}\rangle$ no more than $\frac{\pi}{4}\sqrt{2^n}$ times. Each Grover iteration consists of two successive steps: i) Mark the target states by O_f. ii) Amplify the phases of marked states by D_n.

III. When Should and Should Not We Restore Ancillas

Ancillas (or ancillary qubits) are additional qubits initialized $|0\rangle$ to used as temporary storage. By allowing out-of-place computation, ancillas help reduce the circuit depth (the depth-width trade-off). They ultimately face two possible outcomes: after the computation, they are either reset back to $|0\rangle$ for the next task or discarded as *garbage*. It becomes another trade-off whether to incur the extra circuit required to uncompute (restore) the ancillas or to discard these expensive resources.

A. Ancillas in General Computation

In general quantum computing tasks, we usually implement a fully functional quantum circuit (FFQC) while taking resource management into consideration. A fundamental operation is to restore ancillas for qubit reuse. Based on this, our goal is to find a balance between circuit width and depth so that we can make rational use of existing resources and complete computation efficiently. Consider the structure shown in Fig. 2: the ancillas restored after F_1 are reused in the subsequent task F_2. One can imagine that without ancilla restoration, there would be a catastrophic accumulation of garbage qubits.

B. Ancillas in Grover Search

The Grover oracle O_f occupies the vast majority of an iteration. Thus, intuitively, reducing the depth of O_f is our ultimate goal. For a general computational task represented by a mapping $F : \{0,1\}^{|x|} \to \{0,1\}^{|y|}$, suppose y^* is our target output. We aim to find the corresponding x^* such that $F(x^*) = y^*$. In Grover search, we typically model a function $f : \{0,1\}^{|x|} \to \{0,1\}$ such that $f(x^*) = 1$ if and only if $F(x^*) = y^*$. In practice, f can be constructed from F together with a comparison circuit (to check if $F(x^*) = y^*$), which is usually an n-control-bit Toffoli gate together with n CNOT gates and X gates.

However, the ancillas in Grover search find themselves in a delicate situation. Given a specific f, one typically implements the Grover iteration as shown in Fig. 1. This reflects the fact that, no matter what $f(|\mathbf{\Psi^*}\rangle, |\mathbf{0}\rangle) = |\text{main}, \text{ancs}\rangle$ are, by the end of the Grover oracle we always restore all ancillas back to $|\mathbf{0}\rangle$ and turn $|\mathbf{\Psi}^*\rangle$ into $O_f|\mathbf{\Psi^*}\rangle$.

There are two methods to implement the uncomputation phase f^{-1}. The first that we call the *unitary method* produces a structure symmetric around the central CNOT gate by taking the inverse of each gate in f (which, for Hermitian gates, is the gate itself) and placing them in reverse order. This reverses all qubits except phase information. The alternative, which we call the *measurement-and-reset hybrid method*, is to uncompute the main qubits by the *unitary method*, but to restore the ancillas by measurement and classical control bits. However, it introduces time growth caused by the semiclassical operations, involving measurement readout and reset through a classical controller. In the superconducting platform, we have a CZ gate reaching 6.5ns [5], a single-qubit gate reaching 6.25ns [6], and a readout-and-reset reaching 140ns [7]. Thus, we only consider the first method to implement f^{-1}, which is to execute the inverse of each gate in the reverse order of f.

The depth increase introduced in any uncomputation subcircuit in f (also in f^{-1} by symmetry) will be magnified by a factor of $2 \cdot \frac{\pi}{4}\sqrt{2^n}$. However, if the function f is built from some FFQC F, which is usually designed under the consideration of the width-depth balance. The oracle O_f can be far from reaching the minimum because of the functional overlaps between f's ancilla restoration part and f^{-1}.

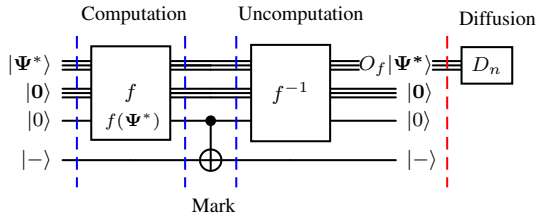


Fig. 1: The structure of a Grover iteration

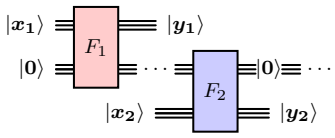


Fig. 2: Reuse ancillas in independent FFQC F_1 and F_2

IV. Removal of Ancillas Restoration

In this section f can refer to the computation part of O_f as shown in Fig. 1, or can refer to a general computation task. For convenience, in the remainder of this work, our use of "uncomputation" specifically denotes the "restoration" for the ancillas.

A. Removal Theorem

Let f be the target function that takes as input the main qubits $|\boldsymbol{x}\rangle$. In a typical implementation of f, we construct a unitary U_f that also takes as input the ancillas $|0...0\rangle$ for temporary storage, computing $U_f|\boldsymbol{x}\rangle|0...0\rangle = |f(\boldsymbol{x})\rangle|\cdot\rangle$. Without considering extreme cases, qubits $|\cdot\rangle$ consists of $|\mathbf{0}\rangle$ and $|garbage\rangle$. Such a circuit usually takes the form computation-uncomputation.

We give the formal definition of both computation components (CC) and uncomputation components (UC). A *component* can refer to a gate, a set of gates, or a subcircuit. We use superscript to indicate the time span of some component. E.g. $\mathcal{UC}^{[t_s,t_c]}$ is a UC from step t_s to t_c. E.g. $\mathcal{CC}^{t_s}$ is a CC that starts and ends at step t_s. The subscript of a component denotes the qubits to which the component acts. E.g. $\mathcal{CC}^{t_s}_{\boldsymbol{x}}$ is a CC that starts and ends at step t_s, working on qubits $|\boldsymbol{x}\rangle$. The subscript of qubits $|\boldsymbol{x}\rangle$ denotes the time to which the state corresponds. E.g. $|\boldsymbol{x}_t\rangle$ is the quantum state of $|\boldsymbol{x}\rangle$ at step t. We use the depth of a circuit, denoted by $\mathcal{T}$, as a measure of the total execution time. Given some of the main qubits $|\boldsymbol{x}\rangle$, ancilla qubits $|\boldsymbol{y}\rangle \overset{init.}{\leftarrow} |\mathbf{0}\rangle$ and some $0 < t_s < t_c < \mathcal{T}$, we have the following definitions:

Definition IV.1. (Computation component). A component $\mathcal{C}^{[t_s,t_c]}$, such that $\mathcal{C}(|\boldsymbol{x_{t_s}}\boldsymbol{y_{t_s}}\rangle) = |\boldsymbol{x_{t_c}}\boldsymbol{y_{t_c}}\rangle$, is a computation component, abbreviated as CC.

Definition IV.2. ((Ancilla) Uncomputation component). A component $\mathcal{C}^{[t_s,t_c]}$, for any intermediate state $|\boldsymbol{x_{t_s}}\boldsymbol{y_{t_s}}\rangle$, such that $\mathcal{C}(|\boldsymbol{x_{t_s}}\boldsymbol{y_{t_s}}\rangle) = |\boldsymbol{x_{t_c}}\rangle|\mathbf{0}\rangle$, is an uncomputation component, abbreviated as UC.

Later in Lemma IV.3, we will find that there are situations where UC and CC coexist and are inseparable. It can be captured by the definitions above. We naturally give the definition of computation-only components (COC):

Definition IV.3. (Computation-only component). A component $\mathcal{C}$ is a computation-only component, if it is a CC but not a UC, abbreviated as COC.

We first present our main result as follows:

Theorem IV.1. *(Removal Theorem in Computational Basis) For a target function f and a circuit U_f, where $U_f|\boldsymbol{x}\rangle|\mathbf{0}\rangle^{\otimes|\boldsymbol{y}|} = |f(\boldsymbol{x}), garbage\rangle|\mathbf{0}\rangle^{\otimes|\boldsymbol{y}-garbage|}$, if $\{|0\rangle, |1\rangle\}$ is the only adopted basis in the computation task of f, then we can remove all individual UCs and inseparable (UC, CC)s from U_f, i.e. we get a new U'_f that $U'_f|\boldsymbol{x}\rangle|\mathbf{0}\rangle = |f(\boldsymbol{x}), garbage\rangle$.*

Proof. This theorem follows by a case analysis combining Lemma IV.2, IV.3, and IV.4. □

The proof of Lemma IV.2, IV.3, and IV.4 can be found in Section IV-B. The estimation of depth decrease depends on the specific FFQC implementation. In Section V, we show how to apply our method to the FFQCs of AES (an encryption algorithm).

Let $\mathcal{UC}_y$ denote some UC for an ancilla qubit $|y\rangle$ and $\mathcal{CC}$ denote some CC. We say that $\mathcal{UC}_y$ and $\mathcal{CC}$ are *related components*, when $\mathcal{CC}$ uses $|y\rangle$ as temporary storage or uses the value stored in $|y\rangle$ for computation. It naturally applies to multi-qubit cases. A UC is always related to some CCs in a properly designed circuit.

If two components are performed *simultaneously*, it means that they are initiated and completed at the same step. For components initiated at different steps but completed at the same step or vice versa, we can always find their subcomponents that are performed *simultaneously*. For the same reason, we only consider the strictly non-overlapping case in the *before* and *after* cases.

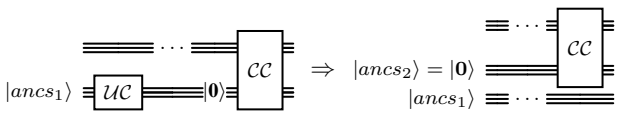
B. Proof of Removal Theorem

The complexity of a circuit can be measured by metrics such as the number of gates, the depth (parallelism), and the width (number of inputs). Counterintuitively, however, one cannot assume that removing some functionality from a circuit will always yield a simpler circuit. In order to prove Theorem IV.1, we present Lemma IV.2, Lemma IV.3 and Lemma IV.4.

1) Case: Before:

Lemma IV.2. *Immediately before a related $\mathcal{CC}$, if a $\mathcal{UC}$ is performed on ancillas whose states are $|ancs_1\rangle$, we can always remove this $\mathcal{UC}$, at the cost of an increase in ancillas by at most the number of $|ancs_1\rangle$.*

Proof. Consider the following structure on the left side; we can replace it with the structure on the right side:



The qubit number of $|ancs_2\rangle$ is equal to $|ancs_1\rangle$. The replacement severs the relatedness between $\mathcal{UC}$ and $\mathcal{CC}$. □

However, the circuit depth may not necessarily be reduced by removing $\mathcal{UC}$. If the following relationship holds between $\mathcal{UC}$, $\mathcal{CC}^*$ and $\mathcal{CC}$, we can not have a reduction in depth by removing $\mathcal{UC}$.

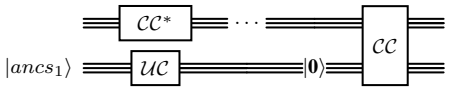


Fig. 3: The scenario in which removing UC cannot bring depth decrease

2) Case: After:

Lemma IV.3. *If some $\mathcal{UC}$ is performed after all related CCs, then the removal of $\mathcal{UC}$ yields a simpler circuit.*

Proof. For the latest $\mathcal{CC}$ ends at time step t_1 and $\mathcal{UC}$ starts at time step t_2, where $0 \leq t_1 \leq t_2 \leq \mathcal{T}$, there must be at least one gate of $\mathcal{UC}$ at step t_2. By removing $\mathcal{UC}$, we can get a simpler circuit. □

In this case, a scenario similar to the one in Fig. 3 should also be considered. However, can the related computation and uncomputation be performed simultaneously?

3) Case: Simultaneously: Now we consider the scenario where, while a CC completes its task, the related UC also happens to restore relevant ancillas. We assume all circuits are composed of gates in a universal quantum gate set $\{H, T, CNOT\}$ [8] on which the proof of Lemma IV.4 is based. According to multi-qubit SolovayKitaev theorem [9], similar conclusions will be drawn for other universal gate sets.

Lemma IV.4. *For some $\mathcal{UC}^{[t_s,t_c]}$ and $\mathcal{CC}^{[t_s,t_c]}$ related and performed simultaneously, we can find subcomponents $\mathcal{CC}_x^{t_c}$ and $\mathcal{UC}_y^{t_c}$, where $|x\rangle$ and $|y\rangle$ are two single qubits, that:*

(i) *$\mathcal{CC}_x^{t_c}$ and $\mathcal{UC}_y^{t_c}$ are not related, or*

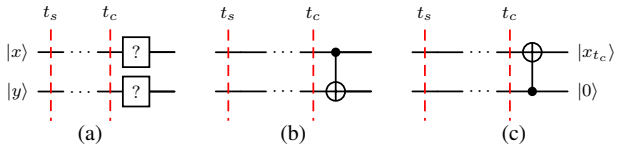
(ii) *$(\mathcal{CC}_x^{t_c}, \mathcal{UC}_y^{t_c})$ is an indivisible component, and*

(1) *$|x_{t_c-1}\rangle$ and $|x_{t_c}\rangle$ are different but statistically equivalent according to measurements in basis $\{|0\rangle, |1\rangle\}$, or*

(2) *$(\mathcal{CC}_x^{t_c}, \mathcal{UC}_y^{t_c})$ is redundant.*

Remark. We can remove $\mathcal{UC}_y^{t_c}$ in case (i) and (ii. 2), but we have to resolve a dispute in (ii. 1) before we remove any component.

Proof. For a certain main qubit $|x\rangle$ and a certain ancilla qubit $|y\rangle$, $\mathcal{UC}_y^{t_c}$ and $\mathcal{CC}_x^{t_c}$ can only appear in one of three distinct configurations:



For configuration (a), the question-mark-gate ? can refer to H, T or I. Apparently, $\mathcal{UC}_y^{t_c}$ and $\mathcal{CC}_x^{t_c}$ are not related.

For the remaining two cases, firstly, let $|x_{t_{c-1}}\rangle$ and $|y_{t_{c-1}}\rangle$ represent the states of qubits $|x\rangle$ and $|y\rangle$ at step t_{c-1} respectively. Thus, we have $|x_{t_{c-1}} y_{t_{c-1}}\rangle = p_{00}|00\rangle + p_{01}|01\rangle + p_{10}|10\rangle + p_{11}|11\rangle = (p_{00}, p_{01}, p_{10}, p_{11})$. The same applies to states at other time steps.

In configuration (b), although $\mathcal{UC}_y^{t_c}$ and $\mathcal{CC}_x^{t_c}$ are inseparable, we use $|\mathcal{CC}_x^{t_c}(x_{t_c-1})\rangle$ to emphasize that the state $|x_{t_c-1}\rangle$ is evolving via $\mathcal{CC}_x^{t_c}$. We have

$$\begin{aligned} CNOT\,|x_{t_c-1} y_{t_c-1}\rangle &= (p_{00}, p_{01}, p_{11}, p_{10}) \\ &= |\mathcal{CC}_x^{t_c}(x_{t_c-1}), 0\rangle \end{aligned}$$

which indicates $p_{01} = 0$ and $p_{10} = 0$. Thus, $|x_{t_c-1} y_{t_c-1}\rangle = (p_{00}, 0, 0, p_{11})$. Provided perfect measurements, any measurement in basis $\{|0\rangle, |1\rangle\}$ and local to $|CC_x^{t_c}(x_{t_c-1})\rangle = |x_{t_c}\rangle$ and $|x_{t_c-1}\rangle$ are statistically no different. Namely, one can think of $|x_{t_c}\rangle = |x_{t_c-1}\rangle$ classically holds. However, the $CNOT$ gate certainly changes measurement results of the control qubit in superposition bases (e.g. $\{|+\rangle, |-\rangle\}$). Therefore, we have the following assertions:

(b. i) Under computational basis, $CNOT$ in (b) is only a redundant UC. This corresponds to case (ii. 1).

(b. ii) Under any superposition basis, $CNOT$ completes both the computation task and the restoration. This corresponds to case (ii. 1).

In configuration (c), we have ($CNOT_{reverse}$ refers to the $CNOT$ gate by switching the control and controlled qubits.):

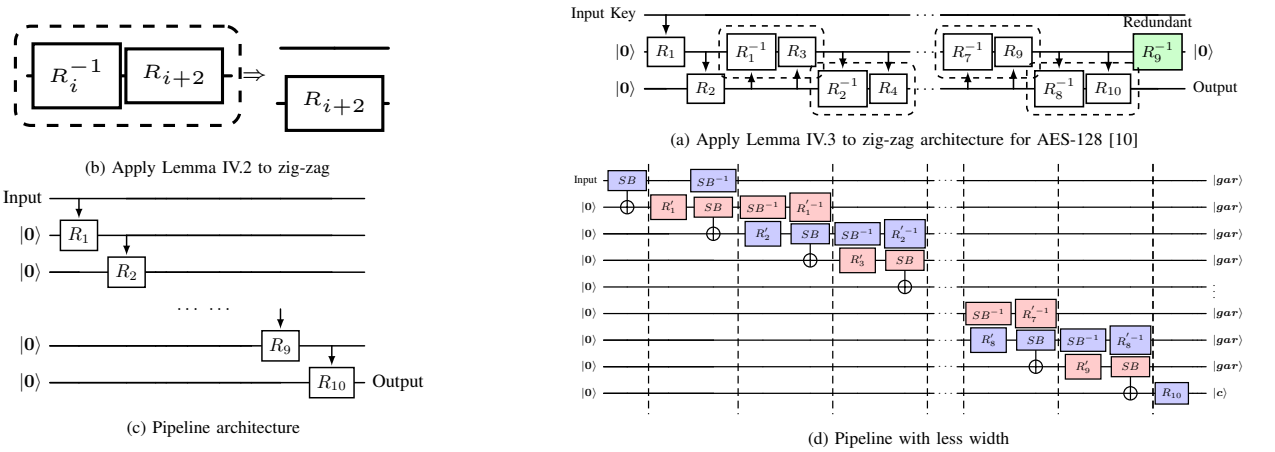
$$\begin{aligned} CNOT_{reverse}\,|x_{t_c-1} y_{t_c-1}\rangle &= (p_{00}, p_{11}, p_{10}, p_{01}) \\ &= |\mathcal{CC}_x^{t_c}(x_{t_c-1}), 0\rangle \end{aligned}$$

which indicates $|\mathcal{CC}_x^{t_c}(x_{t_c-1}), 0\rangle = (p_{00}, 0, p_{10}, 0) = |x_{t_c-1} y_{t_c-1}\rangle$. This corresponds to case (ii. 2) in the Lemma. □

V. Apply our method to quantum circuits of AES

We confine our discussion to the examination of two architectures of AES encryption: 1) zig-zag and 2) pipeline.

1) Initial FFQC: Zig-zag Architecture: The zig-zag architecture for AES was first used in [11]. It has 10 rounds of the sequentially executed Round functions. We consider the version improved by [10]. The circuits shown in Fig. 4a correspond to the computation part f of the Grover oracle.



(b) Apply Lemma IV.2 to zig-zag

(a) Apply Lemma IV.3 to zig-zag architecture for AES-128 [10]

(c) Pipeline architecture

(d) Pipeline with less width

Fig. 4: Transformation from zig-zag to pipeline architecture.

2) Step 1: Removing All Uncomputation in Yields Pipeline: In this step, we apply Theorem IV.1 to zig-zag to obtain a pipeline. Firstly, by Lemma IV.3, we can at least remove R_9^{-1}. Secondly, as shown in Fig. 4b, we can apply Lemma IV.2 to the zig-zag architecture, iteratively, to get a pipeline architecture as shown in Fig. 4c. The current lowest-depth quantum circuit for AES-128 by [12] is at a depth of 647 using Toffoli gates, which adopted a pipeline architecture.

3) Step 2: Reintroducing Depth-preserving Uncomputation for Lower Width: One can recall this scenario by referring to Fig. 3. As shown in Fig. 4d, we propose a strategy slightly different from [12] that reduces their circuit width from 7280 to 7104 while preserving its depth. We recommend that the reader consult Fig. 7(b) in [12] for comparison. All the (Hermitian) subcircuits between the slices can be conducted simultaneously, i.e., $depth(R_i^{'-1} \cdot SB^{-1}) = depth(SB \cdot R'_{i+1})$. Also, we use the same implementation of their low-depth SB (SubBytes) and R'_i (linear component consisting of ShiftRows and MixColumns), as well as the Key Schedule. However, our modification changes how restored ancillas are redistributed.

From $i = 1$, $SB \cdot R'_{i+2}$ uses the ancillas restored by $R_i^{'-1} \cdot SB^{-1}$. The number of qubits can be computed as three parts:

- Two independent sets of ancillas are shared, each consisting of 144×16 (SubBytes) plus 64×4 (MixColumns), for a total of $2560 \times 2 = 5120$ ancillas.
- From the Input stage through R'_1 to R'_9, each stage requires its own 128 ancillas, adding up to 1280. R_{10} can reuse the ancillas restored by $R_8^{'-1} \cdot SB^{-1}$ (where $144 + 64 = 208$ are restored), and therefore do not need to require more qubits.
- Their KeySchedule uses 128 main qubits plus 1444 ancillas for SubWords, totaling 704 ancillas.

All data above are taken from [12]. In total, the new width of the AES-128 quantum circuit is $5120+1280+704 = 7104$.

VI. Conclusion

We propose a method to transform an arbitrary FFQC into a low-depth Grover oracle. We justify the rationality and necessity of this transformation and present the procedure for applying this method to quantum circuits of AES. This further demonstrates that quantum circuits constructed for Grover search have different optimization objectives from those constructed for general computation.

References

[1] L. K. Grover, "A fast quantum mechanical algorithm for database search," 1996.

[2] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing Grover oracles for quantum key search on AES and LowMC," in *EUROCRYPT 2020, Part II*, ser. LNCS, A. Canteaut and Y. Ishai, Eds., vol. 12106. Springer, Cham, May 2020, pp. 280–310.

[3] S. Creemers, "Speeding up grovers algorithm," *European Journal of Operational Research*, vol. 326, no. 1, pp. 13–27, 2025.

[4] C. Zalka, "Grover's quantum searching algorithm is optimal," *Phys. Rev. A*, vol. 60, pp. 2746–2751, Oct 1999.

[5] Y. Chew, T. Tomita, T. P. Mahesh, S. Sugawa, S. de Léséleuc, and K. Ohmori, "Ultrafast energy exchange between two single rydberg atoms on a nanosecond timescale," *Nature Photonics*, vol. 16, no. 10, p. 724729, Aug. 2022.

[6] E. Hyyppä, A. Vepsäläinen, M. Papi, C. F. Chan, S. Inel, A. Landra, W. Liu, J. Luus, F. Marxer, C. Ockeloen-Korppi, S. Orbell, B. Tarasinski, and J. Heinsoo, "Reducing leakage of single-qubit gates for superconducting quantum processors using analytical control pulse envelopes," *PRX Quantum*, vol. 5, no. 3, Sep. 2024.

[7] Y. Sunada, S. Kono, J. Ilves, S. Tamate, T. Sugiyama, Y. Tabuchi, and Y. Nakamura, "Fast readout and reset of a superconducting qubit coupled to a resonator with an intrinsic purcell filter," *Physical Review Applied*, vol. 17, no. 4, Apr. 2022.

[8] P. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, "On universal and fault-tolerant quantum computing: a novel basis and a new constructive proof of universality for shor's basis," in *40th Annual Symposium on Foundations of Computer Science*, 1999, pp. 486–494.

[9] C. M. Dawson and M. A. Nielsen, "The solovay-kitaev algorithm," *Quantum Info. Comput.*, vol. 6, no. 1, p. 8195, Jan. 2006.

[10] J. Zou, Z. Wei, S. Sun, X. Liu, and W. Wu, "Quantum circuit implementations of AES with fewer qubits," in *ASIACRYPT 2020, Part II*, ser. LNCS, S. Moriai and H. Wang, Eds., vol. 12492. Springer, Cham, Dec. 2020, pp. 697–726.

[11] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: Quantum resource estimates," in *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, T. Takagi, Ed. Springer, Cham, 2016, pp. 29–43.

[12] K. Jang, A. Baksi, H. Kim, G. Song, H. Seo, and A. Chattopadhyay, "Quantum analysis of AES," *CiC*, vol. 2, no. 1, p. 25, 2025.