Improved Quantum Circuits for AES: Reducing the Depth and the Number of Qubits

Qun Liu[1,3], Bart Preneel[4], Zheng Zhao[1,3], and Meiqin Wang[1,2,3](✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
{qunliu,zhaozheng}@mail.sdu.edu.cn, mqwang@sdu.edu.cn
[2] Quan Cheng Laboratory, Jinan, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[4] imec-COSIC, KU Leuven, Leuven, Belgium
Bart.Preneel@esat.kuleuven.be

Abstract. Quantum computers hold the potential to solve problems that are intractable for classical computers, thereby driving increased interest in the development of new cryptanalytic ciphers. In NIST's post-quantum standardization process, the security categories are defined by the costs of quantum key search against AES. However, the cost estimates provided by Grassl *et al.* for the search are high. NIST has acknowledged that these initial classifications should be approached cautiously, since the costs of the most advanced attacks can be significantly reduced. Therefore, accurate resource estimations are crucial for evaluating the security of ciphers against quantum adversaries.

This paper presents a set of generic techniques for implementing AES quantum oracles, which are essential for quantum attacks such as Grover's algorithms. Firstly, we introduce the mixing-XOR technique to reuse the ancilla qubits. At ASIACRYPT 2022, Huang *et al.* proposed an S-box structure with 120 ancilla qubits. We are able to reduce the number of ancilla qubits to 83 without increasing the T-depth. Secondly, we propose the combined pipeline architecture with the share technique to combine the S-box and its reverse, which achieves it with only 98 ancilla qubits, resulting in a significant reduction of 59% compared to the independent structure. Thirdly, we use a general algorithm to determine the depth of quantum circuits, searching for the in-place circuit of AES MixColumns with depth 16. Applying these improvements, we achieve the lower quantum depth of AES circuits, obtaining more precise resource estimates for Grover's algorithm. For AES-128, -192, and -256, we only require the depth of 730, 876, and 1,018, respectively. Recently, the community has also focused on the trade-off of the time and space cost of quantum circuits for AES. In this regard, we present quantum implementations of AES circuits with a lower DW-cost on the zig-zag architecture. Compared with the circuit proposed by Huang *et al.*, the DW-cost is reduced by 35%.

Keywords: Quantum Circuit · Grover's Algorithm · S-box · AES

J. Guo and R. Steinfeld (Eds.): ASIACRYPT 2023, LNCS 14440, pp. 67–98, 2023.
https://doi.org/10.1007/978-981-99-8727-6_3

1 Introduction

With the advancements in quantum computing, it has become crucial to investigate the security of cryptographic primitives against quantum attacks. For symmetric key ciphers such as AES, Grover's search algorithm [9] is the main threat, which provides a quadratic speedup, significantly reducing the time required to perform an exhaustive search for the key. The speedup reflects the asymptotic behavior of Grover's algorithm, providing a rough estimate of the security vulnerabilities introduced by quantum computers when applied to symmetric primitives [14]; moreover, estimating the detailed cost of implementing Grover's algorithm is challenging, hence most authors only use a rough approximation.

In 2016, the National Institute of Standards and Technology (NIST, US) initiated a competition[1] aimed at identifying candidate post-quantum algorithms for standardization. As part of this competition, NIST established various security categories for classifying the algorithms (refer to Table 1). These categories were defined based on the complexity of quantum attacks, which can be quantified in terms of circuit size. Specifically, security categories 1, 3, and 5 correspond to key recovery attacks against AES-128, -192, and -256, respectively.

Table 1. The three security levels for NIST's post-quantum competition defined based on the security of three AES variants.

Category	Cipher	Bound of gate counts
Level-1	AES-128	$2^{170}/\texttt{MAXDEPTH}$
Level-3	AES-192	$2^{233}/\texttt{MAXDEPTH}$
Level-5	AES-256	$2^{298}/\texttt{MAXDEPTH}$

In addition to the gate count, another important parameter known as `MAXDEPTH` has been introduced. NIST has proposed an approach that limits quantum attacks to a fixed running time or circuit depth. This limitation is motivated by the challenges associated with executing extremely lengthy serial computations. As a result, there is growing interest in improving the estimation of these levels, particularly in terms of optimizing the cost of quantum implementations for AES. This problem is of significant independent interest and has garnered considerable attention.

The introduction of security levels in NIST's post-quantum competition has spurred the search for optimal quantum circuits. With the potential advantages offered by quantum technology, global efforts are currently directed towards the design of practical large-scale quantum computer architectures, aiming to achieve a tangible quantum advantage [7]. At the highest level of abstraction, the first stage generates a quantum circuit from a reduced set of universal quantum gates.

[1] https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf

The second stage makes the computation robust to errors and thus requires the inclusion of quantum error correction methods. The industrial community has already developed tools for the synthesis and optimization of quantum circuits, including ProjectQ [11,27] and Q# [25], enabling significant advancements in this field.

1.1 Related Work

A quantum circuit can be constructed with the Clifford $+T$ gate set. Similar to [8,16,18,20,21], at ASIACRYPT 2020, Zou *et al.* [29] followed the definitions of the complexity of quantum circuits.

- *Time Complexity.* The time complexity refers to the time required to execute non-parallelizable logical T gates, also known as T-depth.
- *Space Complexity.* The space complexity corresponds to the number of logical qubits needed for the entire quantum computation, often referred to as width.
- *Circuit Complexity.* The circuit complexity is determined by the product of the *time* and *space* complexity, i.e., DW-cost. This metric has been selected as the trade-off metric by Jaques *et al.* at EUROCRYPT 2020 [14] and Huang *et al.* at ASIACRYPT 2022 [12].

Extensive research has been conducted on the design of quantum circuits for AES. The initial circuit proposal by Grassl *et al.* in [8] introduced a zig-zag structure to minimize the number of required qubits. This structure has since been adopted in various works [1,8,18,29]. Langenberg *et al.* [18] presented a new circuit for the AES S-box and key expansion, reducing the qubit count. Building upon this work, Zou *et al.* [29] further optimized the zig-zag structure and introduced new AES S-box circuits.

Early research primarily aimed to reduce qubit counts in quantum circuits. Equally important is minimizing circuit depth, given the challenges of lengthy serial computations required by Grover's algorithm. Additionally, ensuring fault tolerance in quantum circuits is crucial, making the T-depth a significant factor to consider, as emphasized in [12,13].

Nevertheless, the number of qubits can still be a bottleneck. A significant challenge in realizing large-scale quantum computers is the redundancy required for error-correction codes, which demands a large number of physical qubits [10]. Recently, the research community has taken into account both space complexity and time complexity. At EUROCRYPT 2020, Jaques *et al.* [14] proposed several methods to reduce quantum depth and the number of qubits. Building upon this work, Huang *et al.* [12] presented S-boxes of depth 3 and 4, effectively reducing the T-depth and DW-cost. Subsequently, a pipelined architecture was introduced in [13] to further decrease the circuit depth.

1.2 Our Contributions

This paper aligns with the ongoing research direction and prioritizes the depth and the number of qubits as the primary goal. By emphasizing a low-depth metric, we strike a favorable balance between time and space. To further optimize quantum circuits for AES, it becomes crucial to enhance the architecture and building blocks. The contributions put forth by this paper are outlined below.

Improved Structure of S-Box. In a recent study by Huang *et al.* [12] a structure for the AES S-box was proposed utilizing 120 ancilla qubits with a T-depth of 4. In this paper, we introduce the mixing-XOR technique, an algorithm designed to identify idle qubits within linear transformations and repurpose them to store intermediate values. This technique significantly reduces the depth, the number of qubits, and gates required for various quantum circuits. Using this technique, we notably reduce the width of the AES S-box to just 83 ancilla qubits with a T-depth of 4.

Combination of S-Box and S-Box† (Reverse of S-Box). In a study conducted by Jang *et al.* [13], the shallowed pipeline architecture was introduced to evaluate the S-box and its reverse, S-box†, simultaneously for two bytes. The S-box† operation is used to clean up[2] the ancilla qubits used in S-box. During the execution of the S-box, 120 ancilla qubits are utilized to store intermediate values. These ancilla qubits need to be cleaned up by S-box† so that they can be reused in the subsequent round. The approach proposed in [13] requires a total of $120 \times 2 = 240$ ancilla qubits to execute these two operations simultaneously. We propose the combined pipeline architecture with the share technique to combine S-box and S-box† concurrently. This technique significantly reduces the required number of ancilla qubits to just 98, resulting in a 59% reduction.

Improved Quantum Circuit for Reduced Circuit Complexity. We release various quantum circuits for AES-128/192/256 under the combined pipeline architecture, following a similar approach as in [12–14]. Compared with the results in [13], the qubit count and DW-cost are decreased by 42.4%, 41.2%, 36.5% for AES-128, -192, -256, respectively. Notably, the circuits with the AND gate and out-of-place linear layer have a lower depth. For AES-128, -192, and -256, we only require the depth of 730, 876, and 1,018, respectively. When applied to Grover's algorithm, these circuits provide more precise and efficient resource estimates than those proposed in [13,14].

Introducing AND Gates into the Zig-Zag Architecture. We propose modifications to the S-box circuit introduced by Huang *et al.* [12] at ASIACRYPT 2022. Our improved circuit achieves a remarkable reduction in qubit count with the increase of the depth (cf. Sect. 7). By applying the round-in-place zig-zag architecture, the DW-cost for AES-128 is further reduced to 132,800, while the previous best result is 204,800 in [12]. Figure 1 illustrates a comparison of the trade-off between T-depth and width, demonstrating that our circuits exhibit reduced width and DW-cost compared to [12].

[2] Cleaning up an ancilla qubit is to assign the value of the qubit to $|0\rangle$ by some reverse operations.

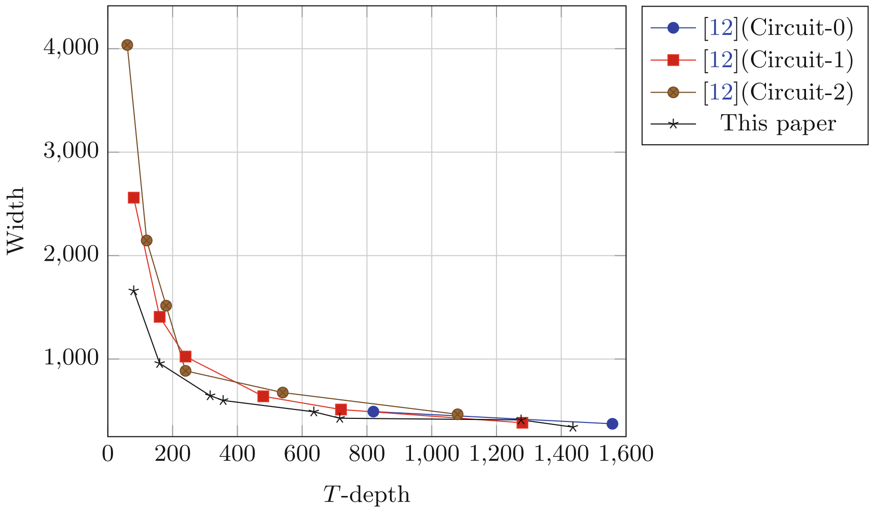


Fig. 1. Width and T-depth for implementing a quantum circuit for AES-128.

1.3 Organization

In Sect. 2, we present our notation and define our metrics. In Sect. 3, we propose an improved structure of the S-box. The combination of S-box and S-box† is described in Sect. 4. The quantum circuits for the AES components are presented in Sect. 5. The complete circuits for AES and the resource estimations based on the pipeline and zig-zag architectures are shown in Sect. 6 and Sect. 7. Finally, we conclude and propose future research directions in Sect. 8.

2 Preliminaries

2.1 Notation

Let $\mathbb{F}_2$ be the finite field with two elements 0 and 1 and let $\mathbb{F}_{2^k}$ be the finite field with 2^k elements. Consider $a, b \in \mathbb{F}_2$. Then $\overline{a} = a \oplus 1$ represents the inversion of a, and $a \oplus b$, $a \cdot b$ and $a|b$ denote the XOR, AND and OR operations of a and b. We denote with M a binary $n \times n$ matrix over $\mathbb{F}_2$; $\boldsymbol{x}$ and $\boldsymbol{y}$ are n-bit vectors over $\mathbb{F}_2$ with $\boldsymbol{x} = [x_0 x_1 \ldots x_{n-1}]$ and $\boldsymbol{y} = [y_0 y_1 \ldots y_{n-1}]$. A linear map defined by M is defined as $\boldsymbol{y}^t = M \cdot \boldsymbol{x}^t$. For a linear map M, $M^\dagger$ is defined as the adjoint of M, where $MM^\dagger = I$ and I is the identity map. The linear map is computed by a circuit as the sequential composition of the individual gates.

2.2 Quantum Circuits

A quantum state is typically written as $|u\rangle$ and $|u\rangle_n$ is used to emphasize that the state has n qubits. Quantum gates exploit quantum entanglement and the superposition states of qubits [4,6]. In general, a quantum circuit is constructed by using the widely adopted universal fault-tolerant gate set Clifford phase shift or T gate:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix},\ S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix},\ \text{CNOT} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix},\ T = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}.$$

We also employ the Pauli-X gate $X = HS^2H = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$. The Pauli-$X$ gate transforms a single qubit from $|a\rangle$ into $|a \oplus 1\rangle$, and the CNOT gate converts $|a\rangle\,|b\rangle$ into $|a\rangle\,|a \oplus b\rangle$. The Toffoli gate is a universal gate defined as the CCNOT gate and maps $|a\rangle\,|b\rangle\,|c\rangle$ to $|a\rangle\,|b\rangle\,|c \oplus a \cdot b\rangle$:

$$\begin{pmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{pmatrix}.$$

Optimization Goals. In the evaluation of quantum circuits, considering both time and memory costs is crucial for assessing their efficiency. The metrics of width, T-depth ($\#TD$), and DW-cost are the primary goals of our focus and attention. A forward-looking perspective suggests that each gate has a depth equal to one, with the T gate incurring a cost similar to other gates. This perspective defines the full depth ($\#FD$) as the time cost metric.

Implementing the AND Operation. As is shown in [24], the Clifford gates are much cheaper than the T gate. Thus, the T gate has a higher impact on the running time. In our quantum circuit, the AND operation $a \cdot b$ is the only source of T-depth. Currently, there exist multiple approaches for implementing the AND operations:

– the Toffoli gate with T-depth 1, or 4 (cf. [2,26]), achieving $|a\rangle\,|b\rangle\,|c\rangle \rightarrow |a\rangle\,|b\rangle\,|c \oplus a \cdot b\rangle$.
– the quantum AND gate using one auxiliary qubit with T-depth 1 (cf. [14]), achieving $|a\rangle\,|b\rangle\,|0\rangle \rightarrow |a\rangle\,|b\rangle\,|a \cdot b\rangle$.

The main distinction lies in the fact that the target qubit of the AND gate must be in the state $|0\rangle$, whereas the Toffoli gate does not have this requirement on its target qubit. In our paper, we consider the target qubit of the AND operation to be in the state $|0\rangle$, allowing us to convert a Toffoli gate into an AND gate. Consequently, implementing the AND operation becomes a unified process, and whether it is represented as a Toffoli gate or an AND gate depends on specific scenarios.

2.3 Description of AES Family

The Advanced Encryption Standard (AES) [5] is a block cipher with 128-bit state (16 bytes) standardized by NIST. The AES family has three members denoted as AES-128 (10 rounds), AES-192 (12 rounds), and AES-256 (14 rounds) with 128-bit, 192-bit, 256-bit keys, respectively.

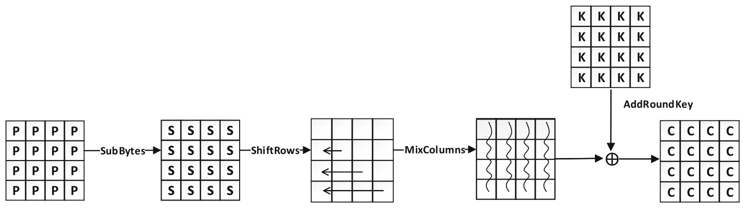


Fig. 2. Structure of an AES round.

The AES round function consists of four operations: AddRoundKey ∘ MixColumns ∘ ShiftRows ∘ SubBytes (see Fig. 2). The 128-bit state is represented in a 4×4 matrix of bytes. Next, we explain each operation.

- AddRoundKey. The operation exclusive-ors each round key to the state.
- ShiftRows. It cyclically rotates the cells of the i-th row to the left over i positions, for $i = 0, 1, 2, 3$.
- SubBytes. The operation applies an 8-bit S-box to each byte of the state in parallel. The 128-bit state is transformed with 16 S-box lookups.
- MixColumns. The MixColumns operation applies a linear transformation on each column of the state with the matrix

$$M = \begin{pmatrix} \texttt{0x02} & \texttt{0x03} & \texttt{0x01} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x03} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x01} & \texttt{0x02} & \texttt{0x03} \\ \texttt{0x03} & \texttt{0x01} & \texttt{0x01} & \texttt{0x02} \end{pmatrix}.$$

For each column in the state, we use MixColumn to represent the operation. Hence, one MixColumns operation consists of four MixColumn operations on the four columns of the state. In SubBytes, 16 S-box operations substitute the individual bytes of the state.

Denote the master key by 32-bit words $W_0, W_1, \ldots, W_{s-1}$, where $s = 4$ for AES-128, $s = 6$ for AES-192, and $s = 8$ for AES-256, respectively. Three operations are used in the algorithm: RotWord cyclically rotates the four bytes to the left by one position; Rcon exclusive-ors a constant to each byte of the word; SubWord applies four S-boxes in parallel to the four bytes of the word.

3 Improved Quantum Circuit Implementations of AES S-Box

In [14], the S-box circuit constructed by Jaques *et al.* for AES requires 120 ancilla qubits with T-depth 6. Subsequently, Huang *et al.* [12] reduced the T-depth to 4 while keeping the number of qubits at 120. In this section, our aim is to construct a new circuit from the classical circuit with a reduced number of qubits and gates, as well as less quantum depth. We build upon these papers and introduce the mixing-XOR technique (m-XOR) to accomplish this construction. The technique enables us to identify reusable qubits within the S-box, resulting in a circuit with T-depth 4, requiring only $74 + 9$ ancilla qubits, where 9 qubits with the value $|0\rangle$ are used to satisfy the parallelism of AND gates. If the S-box circuit used for improvements is updated, our approach can still be employed to reduce these three metrics.

3.1 M-XOR Technique

Two operations are used in quantum linear circuits:

- The *updating* operation is *in-place* and can be implemented by a CNOT gate $|a\rangle |b\rangle \rightarrow |a\rangle |a \oplus b\rangle$, defined as CNOT$(a, b)$.
- The *creating* operation is *out-of-place*, requiring two CNOT gates $|a\rangle |b\rangle |c\rangle \rightarrow |a\rangle |b\rangle |c \oplus a\rangle$ and $|a\rangle |b\rangle |c \oplus a\rangle \rightarrow |a\rangle |b\rangle |c \oplus a \oplus b\rangle$, defined as CNOT2$(a, b, c)$.

which correspond to s-XOR and g-XOR in classical circuits (cf. [15,28]). We observe that it is beneficial to transform a *creating* operation into an *updating* operation. Based on Observation 1, we propose the m-XOR technique to construct the quantum circuit from the classical circuit with a reduced number of qubits and gates, as well as less quantum depth.

Observation 1. *Given a quantum circuit with creating operations, some qubits can be reused by transforming creating operations into updating operations.*

Definition 1 (m-XOR technique). *The m-XOR technique mixes updating operations and creating operations to implement a linear transformation with the lowest number of qubits.*

Example 1. Given that we assign two qubits to q_a and q_b, the subsequent operation is a *creating* operation: $q_c = q_c \oplus (q_a \oplus q_b)$. If q_a is not further utilized in the subsequent circuit, this operation can be converted to an *updating* operation: $q_a = q_a \oplus q_b$. When utilizing q_c, we simply select the qubit q_a.

The detection of idle qubits forms the essence of this technique. Proposition 1 outlines the qubits that can be eliminated from the circuit.

Proposition 1. *In a sequentially written quantum circuit, the conversion from a creating operation $t_c = t_c \oplus (t_a \oplus t_b)$ to an updating operation $t_a = t_a \oplus t_b$ requires the fulfillment of the following conditions:*

- *t_a should not be utilized in the subsequent circuit.*
- *t_c does not appear in the previous circuit.*

To successfully perform the conversion, both conditions must be satisfied. Failing to meet either condition can compromise the correctness of the circuit.

Proof. For the operation $t_c = t_c \oplus (t_a \oplus t_b)$, the question is how to save the value of t_c. If t_a is involved in subsequent computations, it cannot be safely updated. If t_a is not used in the subsequent circuit, we can use $t_a = t_a \oplus t_b$ to replace t_c, which requires that t_c does not appear in the previous circuit. Otherwise, t_c is not initialized to $|0\rangle$ and the transformation ignores the original value of t_c. □

Proposition 1 implies Algorithm 1 to optimize a quantum circuit using the m-XOR technique.

Algorithm 1. Transformation from *creating* operations into *updating* operations

```
Input: A sequentially written quantum circuit $\mathcal{C}$ in sequence
Output: The optimized quantum circuit with a reduced number of qubits and gates,
    as well as less quantum depth
 1: for each gate $g \in \mathcal{C}$ do
 2:     if $g$ is creating operation $t_c = t_c \oplus (t_a \oplus t_b)$ then
 3:         if $t_c$ appears in the previous circuit then
 4:             Continue
 5:         end if
 6:         Check the subsequent circuit and count the number $n_a$ that $t_a$ is used
 7:         Check the subsequent circuit and count the number $n_b$ that $t_b$ is used
 8:         if $n_i = 0$ $(i = a$ or $b)$ then
 9:             Replace $t_c$ with $t_i$ in $g$ and in the subsequent circuit
10:         end if
11:     end if
12: end for
13: return $\mathcal{C}$
```

3.2 Applying the m-XOR Technique to AES S-Box

We apply Algorithm 1 to optimize the AES S-box from [12]. The circuit contains 34 AND operations, 120 *creating* operations, and 4 X gates. Apart from the 8-bit inputs $u_0, u_1, \ldots, u_7$ and the 8-bit outputs $s_0, s_1, \ldots, s_7$, the circuit of the S-box requires 120 ancilla qubits: $t_0, \ldots, t_{26}$, $m_0, \ldots, m_{62}$, and $l_0, \ldots, l_{29}$.

We find that 46 qubits are unnecessary. Our optimized S-box circuit requires only 74 ancilla qubits, denoted as $q_0, q_1, \ldots, q_{73}$. A detailed list of all allocated qubits is provided in Table 2. To facilitate the description, we introduce the AND operation and the X gate, which are defined as follows:

$$\begin{aligned}&\mathrm{AND}(a, b, c) \rightarrow c = a \cdot b\,,\\&\mathrm{X}(a) \rightarrow a = a \oplus 1\,.\end{aligned}$$

Table 2. New structure of S-box of AES with 74 ancilla qubits.

No.	Gate	No.	Gate	No.	Gate
0	CNOT2(u_7, u_4, q_0)	53	AND(q_{15}, q_9, q_{26})	106	CNOT(q_{25}, q_{63})
1	CNOT2(u_7, u_2, q_1)	54	AND(q_{61}, q_{21}, q_{32})	107	CNOT(q_{40}, q_{64})
2	CNOT(u_1, u_7)	55	AND(q_{16}, q_{60}, q_{35})	108	CNOT(q_{36}, q_{65})
3	CNOT2(u_4, u_2, q_2)	56	CNOT(q_{25}, q_{63})	109	CNOT(q_{21}, q_{66})
4	CNOT(u_1, u_3)	57	CNOT2(q_{16}, q_{26}, q_{27})	110	CNOT(q_{39}, q_{67})
5	CNOT2(q_0, u_3, q_3)	58	CNOT2(q_9, q_{16}, q_{28})	111	CNOT(q_{33}, q_{68})
6	CNOT2(u_6, u_5, q_4)	59	CNOT(q_{28}, q_{62})	112	CNOT(q_{16}, q_{69})
7	CNOT2(u_0, q_3, q_5)	60	CNOT2(q_{21}, q_{26}, q_{29})	113	CNOT(q_{38}, q_{70})
8	CNOT2(u_0, q_4, q_6)	61	CNOT2(q_{28}, q_{26}, q_{34})	114	CNOT(q_{41}, q_{71})
9	CNOT2(q_3, q_4, q_7)	62	AND(q_{29}, q_{28}, q_{30})	115	CNOT(q_{37}, q_{72})
10	CNOT(u_2, u_6)	63	AND(q_{27}, q_{25}, q_{31})	116	CNOT2(q_{57}, q_{58}, q_{60})
11	CNOT(u_2, u_5)	64	AND(q_{62}, q_{32}, q_{33})	117	CNOT2(q_{46}, q_{52}, q_{61})
12	CNOT2(u_7, q_2, q_8)	65	AND(q_{63}, q_{35}, q_{36})	118	CNOT2(q_{42}, q_{44}, q_{62})
13	CNOT2(q_3, u_6, q_9)	66	CNOT(q_{25}, q_{26})	119	CNOT2(q_{43}, q_{51}, q_{63})
14	CNOT(u_3, u_6)	67	CNOT(q_{30}, q_{16})	120	CNOT2(q_{50}, q_{54}, q_{64})
15	CNOT(u_5, u_3)	68	CNOT(q_{34}, q_{33})	121	CNOT2(q_{45}, q_{57}, q_{65})
16	CNOT2(q_6, u_3, q_{10})	69	CNOT(q_{31}, q_{21})	122	CNOT2(q_{58}, q_{65}, q_{66})
17	CNOT(u_0, u_4)	70	CNOT(q_{26}, q_{36})	123	CNOT2(q_{42}, q_{63}, q_{67})
18	CNOT(q_4, u_4)	71	CNOT2(q_{33}, q_{36}, q_{37})	124	CNOT2(q_{47}, q_{55}, q_{68})
19	CNOT2(q_0, u_4, q_{11})	72	CNOT2(q_{16}, q_{21}, q_{38})	125	CNOT2(q_{48}, q_{49}, q_{69})
20	CNOT(u_0, u_1)	73	CNOT2(q_{16}, q_{33}, q_{39})	126	CNOT2(q_{49}, q_{64}, q_{70})
21	CNOT(u_1, q_4)	74	CNOT2(q_{21}, q_{36}, q_{40})	127	CNOT2(q_{56}, q_{62}, q_{71})
22	CNOT2(q_1, q_4, q_{12})	75	CNOT2(q_{38}, q_{37}, q_{41})	128	CNOT2(q_{44}, q_{47}, q_{72})
23	CNOT2(q_1, q_7, q_{13})	76	CNOT(q_{40}, q_{64})	129	CNOT2(q_{66}, q_{70}, q_{73})
24	CNOT2(q_{11}, q_{10}, q_{14})	77	CNOT(q_{36}, q_{65})	130	CNOT(q_{60}, q_{46})
25	CNOT2(u_7, u_3, q_{15})	78	CNOT(q_{21}, q_{66})	131	CNOT(q_{57}, q_{48})
26	CNOT(q_0, u_5)	79	CNOT(q_{39}, q_{67})	132	CNOT(q_{61}, q_{51})
27	AND(q_8, q_3, q_{16})	80	CNOT(q_{33}, q_{68})	133	CNOT(q_{60}, q_{52})
28	AND(q_{12}, q_5, q_{17})	81	CNOT(q_{16}, q_{69})	134	CNOT(q_{61}, q_{53})
29	AND(u_4, u_0, q_{18})	82	CNOT(q_{38}, q_{70})	135	CNOT(q_{68}, q_{54})
30	AND(u_7, u_3, q_{19})	83	CNOT(q_{41}, q_{71})	136	CNOT(q_{64}, q_{59})
31	AND(q_4, q_6, q_{20})	84	CNOT(q_{37}, q_{72})	137	CNOT(q_{61}, q_{60})
32	AND(q_{11}, q_{10}, q_{21})	85	AND(q_{40}, q_3, q_{42})	138	CNOT2(q_{61}, q_{67}, s_4)
33	AND(q_0, u_6, q_{22})	86	AND(q_{36}, q_5, q_{43})	139	CNOT2(q_{63}, q_{72}, s_3)
34	AND(q_2, u_5, q_{23})	87	AND(q_{21}, u_0, q_{44})	140	CNOT2(q_{54}, q_{62}, s_0)
35	AND(q_1, q_7, q_{24})	88	AND(q_{39}, u_3, q_{45})	141	CNOT2(q_{51}, q_{69}, s_7)
36	CNOT(q_{16}, q_9)	89	AND(q_{33}, q_6, q_{46})	142	CNOT2(q_{67}, q_{69}, s_6)
37	CNOT(q_{18}, q_{16})	90	AND(q_{16}, q_{10}, q_{47})	143	CNOT2(q_{68}, q_{70}, s_1)
38	CNOT(q_{19}, q_{15})	91	AND(q_{38}, u_6, q_{48})	144	CNOT2(q_{71}, q_{48}, s_5)
39	CNOT(q_{19}, q_{21})	92	AND(q_{41}, u_5, q_{49})	145	CNOT2(q_{71}, q_{53}, s_2)
40	CNOT(q_{22}, q_{23})	93	AND(q_{37}, q_7, q_{50})	146	CNOT(q_{66}, s_7)
41	CNOT(q_{22}, q_{24})	94	AND(q_{64}, q_8, q_{51})	147	CNOT(q_{52}, s_6)
42	CNOT(q_{17}, q_9)	95	AND(q_{65}, q_{12}, q_{52})	148	CNOT(q_{59}, s_5)
43	CNOT(q_{13}, q_{16})	96	AND(q_{66}, u_4, q_{53})	149	X(s_6)
44	CNOT(q_{20}, q_{15})	97	AND(q_{67}, u_7, q_{54})	150	X(s_5)
45	CNOT(q_{24}, q_{21})	98	AND(q_{68}, q_4, q_{55})	151	CNOT(q_{66}, s_4)
46	CNOT(q_{23}, q_9)	99	AND(q_{69}, q_{11}, q_{56})	152	CNOT(q_{60}, s_3)
47	CNOT(q_{24}, q_{16})	100	AND(q_{70}, q_0, q_{57})	153	CNOT(q_{73}, s_2)
48	CNOT(q_{23}, q_{15})	101	AND(q_{71}, q_2, q_{58})	154	CNOT(q_{46}, s_1)
49	CNOT(q_{14}, q_{21})	102	AND(q_{72}, q_1, q_{59})	155	CNOT(q_{66}, s_0)
50	CNOT2(q_{15}, q_{21}, q_{25})	103	CNOT(q_{15}, q_{60})	156	X(s_1)
51	CNOT(q_{15}, q_{60})	104	CNOT(q_9, q_{61})	157	X(s_0)
52	CNOT(q_9, q_{61})	105	CNOT(q_{28}, q_{62})		

We take an example to show the process. In No. 2 of the previous circuit, the operation CNOT2(u_7, u_1, t_2) statisties Proposition 1. Therefore, we can use u_7 to save the value of t_2, reducing one ancilla qubit and one CNOT gate. The gate substitution is

$$\mathrm{CNOT2}(u_7, u_1, t_2) \rightarrow \mathrm{CNOT}(u_1, u_7)\,.$$

3.3 Comparisons of Resource Estimations of the AES S-Box

There are various implementations of S-box quantum circuits. Some circuits use Toffoli gates, while others use AND gates. We show the comparison of different Toffoli-based circuits in Table 3.

Table 3. Comparison of implementations of S-box using Toffoli gates. "$6+16$" represents 6 ancilla qubits and 16 input and output qubits of the AES S-box.

Source	Width	#Toffoli	#CNOT	#1qCliff	Toffoli depth
[18]	16+16	55	314	4	40
[29]	$6+16$	52	326	4	41
[29]	$7+16$	48	330	4	39
[29]	$8+16$	46	332	4	37
[21]	$5+16$	57	193	4	24
[21]	$6+16$	57	195	4	22
[14]	$120+16$	34	186	4	6
[12]	$120+16$	34	214	4	4
This paper	**74** $+16$	34	**168**	4	4

Next, we compare the S-box circuits utilizing AND gates with [12,14]. The authors estimated both the S-box and S-box† using Q#. To ensure a fair comparison, we adopt the same approach (cf. Table 4). As mentioned in [12], for the 4-th AND layer, the execution of 18 parallel AND gates is required, with each gate necessitating an ancilla qubit, resulting in a total of 18 qubits. We find that $q_{73}, s_0, s_1, \ldots, s_7$ remain $|0\rangle$ after the 4-th AND layer. Thus, the remaining 9 qubits need to be allocated. Thus, the width of the S-box circuit is $74 + 16 + 9 = 99$. We find that the optimized quantum circuit has a reduced number of qubits and gates, as well as less quantum depth.

4 Improved Combination of S-Box and S-Box†

In [14], Jaques *et al.* used the pipeline architecture for AES. Subsequently, Jang *et al.* [13] proposed the shallowed pipeline architecture, which necessitates $120 + 120 = 240$ ancilla qubits for executing S-box and S-box†. In this section, we

Table 4. Comparison of several implementations of S-box and S-box† based on the AND gates. Here #M counts the number of Measurements.

Source	Width	#CNOT	#1qCliff	$\#T$	#M	$\#TD$	$\#FD$
[14]	136	664	205	136	34	6	117
[12]	136	718	208	136	34	4	109
This paper	**99**	**624**	**204**	136	34	4	**101**

present the combined pipeline architecture with the share technique to combine the S-box and S-box†. The combined architecture demonstrates a significant efficiency improvement by utilizing only $74 + 24 = 98$ ancilla qubits. Note that we do not use the inverse S-box in the pipeline architecture.

4.1 Pipeline Architecture for AES

A pipeline architecture is employed to reduce the reuse of qubits and achieve lower T-depth in circuit execution. As is shown in [13], the resource estimation based on AND gates and the pipeline architecture in [14] may underestimate the depth/width cost of the circuits, partly due to the bugs in Q#[3]. Then, Jang *et al.* [13] proposed a shallowed pipeline architecture as depicted in Fig. 3. In the following, we provide a detailed description of each component.

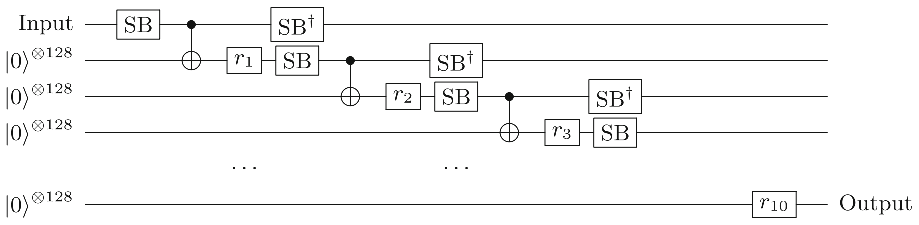

Input
$|0\rangle^{\otimes 128}$
$|0\rangle^{\otimes 128}$
$|0\rangle^{\otimes 128}$
$|0\rangle^{\otimes 128}$
SB
SB†
r_1
r_2
r_3
r_{10}
Output

Fig. 3. The shallowed pipeline architecture.

The SubBytes operation comprises 16 distinct S-boxes applied to 16 bytes, denoted as SB in Fig. 3. For the i-th round function R_i, we represent R_i as the sum of SB and r_i. Each S-box takes an 8-qubit input, utilizes a set of ancilla qubits Q_i^1 for each byte B_i $(0 \le i \le 15)$, and produces an 8-qubit output on the subsequent line. In [13], $|Q_i^1| = 120$. If we do not clean up the ancilla qubits in each Q_i^1, we would need to allocate an ancilla qubit set for the subsequent S-box operations, resulting in a total of 10 rounds $\times$ 16 bytes $\times$ 120 qubits $=$ 19,200 qubits.

Hence, the use of SB†, which represents the adjoint of SB, becomes necessary to clear the ancilla qubits. During the execution of SB, 16 sets of ancilla qubits

[3] https://github.com/microsoft/qsharp-runtime/issues/1037.

Q_i^1 $(0 \leq i \leq 15)$ are utilized. Subsequently, SB† is employed to clean up the qubits in each Q_i^1. To ensure that the T-depth does not increase, the execution of SB† in R_j and SB in R_{j+1} is performed simultaneously. To achieve this, 16 sets of ancilla qubits Q_k^2 $(0 \leq k \leq 15)$ are required. We illustrate this relationship using an example.

Example 2. We begin with an initial set of 16 Q_i^1 $(0 \leq i \leq 15)$ and 16 Q_k^2 $(0 \leq k \leq 15)$.

- In R_1, SB uses 16 Q_i^1 $(0 \leq i \leq 15)$.
- In R_2, SB uses 16 Q_k^2 $(0 \leq k \leq 15)$ and SB† cleans up the qubits in 16 Q_i^1 $(0 \leq i \leq 15)$.
- Then, in round R_3, SB uses the 16 Q_i^1 $(0 \leq i \leq 15)$ sets, and SB† clears the qubits in the 16 Q_k^2 $(0 \leq k \leq 15)$ sets.
- These two sets of ancilla qubits are alternated in the remaining rounds.
- The total count of ancilla qubits is $2 \times 16 \times 120 = 3840$.

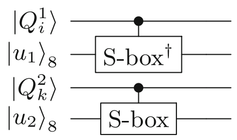


Fig. 4. Independent structure to execute S-box and S-box† simultaneously.

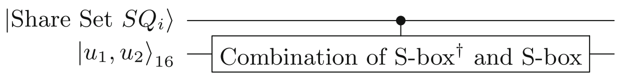
4.2 Combined Pipeline Architecture

For each byte B_i of the state, there are two ancilla qubit sets Q_i^1 and Q_i^2 used in S-box and S-box† (see Fig. 4). With the S-box circuit from [12], the independent structure in [13] requires $|Q_i^1| + |Q_i^2| = 120 + 120 = 240$ ancilla qubits. For 16 bytes of AES, the total ancilla qubit count is $240 \times 16 = 3840$.

We point out that there are unnecessary qubits for this operation, which is based on Observation 2.

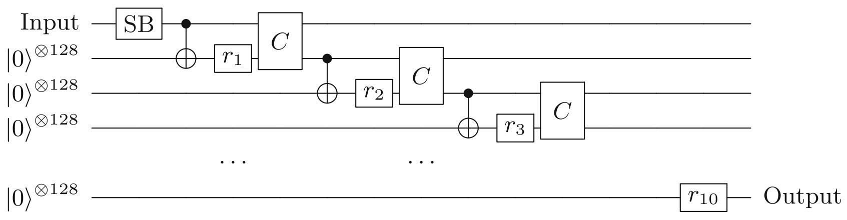
Observation 2. *In the independent structure of S-box and S-box†, during the execution of S-box†, the qubits are consistently cleaned up, and these qubits are not utilized in the S-box operation. Conversely, S-box employs a fresh qubit set to select the available qubits.*

Thus, it should be explored to reuse the qubits cleaned up by S-box† immediately. We propose the *share technique* that combines the qubit sets of S-box and S-box† (see Fig. 5). The combination only uses one set, the share set SQ_i. After the analysis in Sect. 4.3, we have $|SQ_i| = 74 + 24 = 98$, which is much smaller than $|Q_i^1| + |Q_i^2| = 240$. Using the combined S-box and S-box†, we can propose the combined pipeline architecture for AES (cf. Fig. 6).



$|\text{Share Set } SQ_i\rangle$
$|u_1, u_2\rangle_{16}$
Combination of S-box† and S-box

Fig. 5. Combined structure to execute S-box and S-box† simultaneously.



Input
$|0\rangle^{\otimes 128}$
$|0\rangle^{\otimes 128}$
$|0\rangle^{\otimes 128}$
$|0\rangle^{\otimes 128}$
SB
r_1
C
r_2
C
r_3
C
r_{10}
Output

Fig. 6. The combined pipeline architecture, where C is the combined S-box and S-box†.

Discussion on Different Pipeline Architectures. We employ a simple structure to discuss this distinction. Assume that an r-round cipher ($r \geq 1$) consists of only two components, SB and MixColumns, with corresponding depths of d_s ($d_s \geq 1$) and d_m ($d_m \geq 1$). Additionally, SB requires q_s ($q_s \geq 0$) ancilla qubits, MixColumns requires q_m ($q_m \geq 0$) ancilla qubits, and each round requires q_r ($q_r \geq 1$) qubits. Table 5 shows the comparison.

Table 5. Comparison of different pipeline architectures.

Architecture	Width	$\#FD$
Original architecture [14]	$(r+1) \cdot q_r + q_s + q_m$	$d_s + \max(d_s, d_m)$
Shallowed architecture [13]	$(r+1) \cdot q_r + \max(2q_s, q_m)$	$d_s + d_m$
Combined architecture	$(r+1) \cdot q_r + \max((1+\epsilon) \cdot q_s, q_m)$	$d_s + d_m$

In the original pipeline architecture, we first execute SB, and then run SB† and MixColumns simultaneously. Then, we use $((r+1) \cdot q_r + q_s + q_m)$ qubits with depth $(d_s + \max(d_s, d_m))$. For the shallowed pipeline architecture, we use the independent structure to execute SB and SB† simultaneously and then execute MixColumns. The circuit requires $((r+1) \cdot q_r + \max(2q_s, q_m))$ qubits with depth $(d_s + d_m)$. In the combined pipeline architecture, we use the combined structure to execute SB and SB† simultaneously and then execute MixColumns. For the share set, we use $(1+\epsilon)q_s$ qubits, where ϵ depends on the size of the share set and we have $0 \leq \epsilon \leq 1$. Then, we can obtain the following observation.

Observation 3. *If $d_s > d_m$, the shallowed and combined pipeline architectures have the lowest circuit depth. If $q_m > \epsilon \cdot q_s$, the combined pipeline architecture has the lowest width.*

4.3 Share Technique

The share technique is based on Observation 4, where we determine which qubits can be reused and preassign them accordingly.

Observation 4. *Suppose that S-box and S-box† are executed simultaneously. When S-box† clears up a qubit, the qubit can be immediately reused by S-box.*

Next, we introduce the general share technique in Definition 2. This technique utilizes five sets that satisfy Property 1, defined as follows.

Definition 2. *The share technique utilizes a set SQ_i to execute S-box and S-box† simultaneously for a single byte B_i. Four sets are employed to split SQ_i, where the set with subscript public stores the qubits in the state $|0\rangle$, and the set with subscript private stores the qubits that are being used.*

- $q_{private}^{old}$ *is the set of qubits that will be cleaned up by S-box†.*
- q_{public}^{old} *is the set of unallocated qubits for S-box†.*
- $q_{private}^{new}$ *is the set of qubits used by S-box.*
- q_{public}^{new} *is the set of qubits that are not used by S-box.*

Property 1. The five qubit sets in the public qubit technique satisfy the following equations:

$$\begin{aligned} SQ_i = q_{private}^{old} \cup q_{public}^{old} = q_{private}^{new} \cup q_{public}^{new}, \\ q_{private}^{old} \cap q_{public}^{old} = \phi\,,\ q_{private}^{new} \cap q_{public}^{new} = \phi. \end{aligned} \tag{1}$$

These equations can be explained as follows: S-box and S-box† utilize the share set SQ_i in our optimization, where we set $|SQ_i| = 74 + 24 = 98$. For each S-box operation, an ancilla qubit set stores 74 qubits, resulting in $|q_{private}^{old}| = 74$, and $|q_{public}^{old}| = |SQ_i/q_{private}^{old}| = 24$. When S-box† cleans up one qubit q, we have

$$q_{private}^{old} = q_{private}^{old}/\{q\}\,, \qquad q_{public}^{old} = q_{public}^{old} \cup \{q\}\,. \tag{2}$$

At the same time, S-box requires qubits to store intermediate values. S-box chooses these qubits from q_{public}^{old}. Assuming that q is chosen by S-box, we have

$$q_{public}^{old} = q_{public}^{old}/\{q\}\,, \qquad q_{private}^{new} = q_{private}^{new} \cup \{q\}\,. \tag{3}$$

Finally, after completing the combination, we assign the qubits not used in S-box to q_{public}^{new}, i.e., $q_{public}^{new} = SQ/q_{private}^{new}$. Since this S-box also uses 74 ancilla qubits, we have

$$|q_{private}^{new}| = |q_{private}^{old}| = 74\,, \quad |q_{public}^{new}| = |q_{public}^{old}| = 24\,. \tag{4}$$

Proposition 2 demonstrates that the sizes of the five qubit sets remain unchanged after executing the combination in the share technique.

Proposition 2. *In the share technique, after completing the combination of S-box and S-box†, the sizes of the five qubit sets remain constant.*

Proof. During the combination process, we set $|SQ_i| = a$ and $|q_{private}^{old}| = u$. By Property 1, $|q_{public}^{old}|$ is $z = a - u$.

Next, we examine the combination process in detail. S-box† cleans up all the qubits in $q_{private}^{old}$ and adds them to q_{public}^{old}. This results in a total of $u + z = a$ qubits in q_{public}^{old}. When executing S-box, u qubits are selected from q_{public}^{old}. Therefore, $|q_{public}^{old}|$ remains $a - u = z$. Additionally, the unselected qubits from $|q_{public}^{old}|$, which amount to z, are stored in $|q_{public}^{new}|$.

In conclusion, as long as q_{public}^{old} contains a sufficient number of qubits, the sizes of the qubit sets remain unchanged. This completes the proof. □

Example 3. Suppose we visualize the unallocated qubits in q_{public}^{old} as a pool of water, as shown in Fig. 7. The water in the pool represents the available qubits for allocation. The process in which S-box† cleans up the qubits can be likened to adding water to the pool, replenishing the available qubits. When S-box requires an ancilla qubit, it can be seen as drawing water (qubits) from the pool.

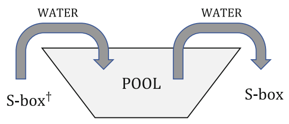


Fig. 7. The water pool in the public qubit technique.

We provide Algorithm 2 to decide the size of SQ_i. In the aforementioned process, the ability to execute the S-box depends on the availability of qubits in the pool. If the pool is empty, the S-box cannot be executed. Specifically, if we find $|q_{public}^{old}| = 0$, it means that no qubits can be selected from q_{public}^{old}, indicating that $|SQ_i|$ is too small to accommodate the S-box operation. Here, $|SQ_i|$ represents the maximum capacity of the "pool" to hold qubits. Therefore, we gradually increase the size until Algorithm 2 no longer returns "Error".

As a result, SQ_i is composed of two parts: the initially set qubits, which are assigned the value $|0\rangle$ and stored in q_{public}^{old}, and the qubits that require cleaning by S-box†, which are stored in $q_{private}^{new}$.

4.4 Applying the Share Technique to the AES S-Box

In this section, we present the complete combination of AES S-box and S-box† using the share technique. We illustrate the implementation of the S-box based on the structure provided in Table 2. To facilitate the combination, we propose a method to split the S-box circuit based on its T-depth. The S-box circuit is divided into several layers, taking into account the T-depth of each gate. The layering scheme is as follows:

Algorithm 2. Combination of S-box and S-box†

```
Input: Public qubit set $SQ_i$, used qubit set $q^{old}_{private}$, and unallocated qubit set $q^{old}_{public}$
Output: New used qubit set $q^{new}_{private}$, and new unallocated qubit set $q^{new}_{public}$
 1: The depth $d_{max}$ is the maximum of depth of S-box and S-box$^\dagger$
 2: for the current depth $d$ from 1 to $d_{max}$ do
 3:     if $|q^{old}_{public}| = 0$ then
 4:         return Error
 5:     end if
 6:     Choose $q \in q^{old}_{public}$, execute S-box under depth $d$, and put $q$ into $q^{new}_{private}$
 7:     Execute S-box$^\dagger$ under depth $d$. If one qubit $q'$ is cleaned up, put $q'$ into $q^{old}_{public}$
 8: end for
 9: $q^{new}_{public} = SQ_i / q^{new}_{private}$
10: return $q^{new}_{public}$ and $q^{new}_{private}$
```

- The first layer consists of the gates that precede the execution of T gates (No. 0–26).
- The second layer includes gates with T-depth 1 (No. 27–35).
- The third layer encompasses gates between T-depth 1 and 2 (No. 36–52).
- The fourth layer comprises gates with T-depth 2 (No. 53–55).
- The remaining layers are split according to the proposed method: No. 56–61, No. 62–65, No. 66–84, No. 85–102, No. 103–157.

The circuit is executed in order, with a total of nine layers. We denote each layer as L_i. For S-box†, the order of execution is reversed. The ancilla qubits involved in each layer are specified in Table 6.

Table 6. The layers and the corresponding ancilla qubits in S-box and S-box†.

S-box	Ancilla qubits	S-box†
L_1	$q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}$	L_9
L_2	$q_{16}, q_{17}, q_{18}, q_{19}, q_{20}, q_{21}, q_{22}, q_{23}, q_{24}$	L_8
L_3	q_{25}, q_{60}, q_{61}	L_7
L_4	q_{26}, q_{32}, q_{35}	L_6
L_5	$q_{27}, q_{28}, q_{29}, q_{34}, q_{62}, q_{63}$	L_5
L_6	$q_{30}, q_{31}, q_{33}, q_{36}$	L_4
L_7	$q_{37}, q_{38}, q_{39}, q_{40}, q_{41}, q_{64}, q_{65}, q_{66}, q_{67}, q_{68}, q_{69}, q_{70}, q_{71}, q_{72}$	L_3
L_8	$q_{42}, q_{43}, q_{44}, q_{45}, q_{46}, q_{47}, q_{48}, q_{49}, q_{50}, q_{51}, q_{52}, q_{53}, q_{54}, q_{55}, q_{56}, q_{57}, q_{58}, q_{59}$	L_2
L_9	q_{73}	L_1

In the case of S-box, in each layer L_i, every qubit q represents the qubit that is initially used by the S-box. Prior to L_i, the qubit q is initialized to $|0\rangle$. Conversely, for S-box†, in each layer L_i, each qubit q indicates the qubit that is cleaned up by S-box†. After the completion of L_i, the qubit q is reset to $|0\rangle$.

Table 7. Qubit allocation for the combination of S-box and S-box†. "Previous Pool" is the qubit count of the previous pool in L_i. "Need" is the qubit count that S-box requires in L_i. "Preset Qubits" is the number of qubits that we preset. "Cleaning" is the qubit count that S-box† cleans up. "New Pool" is the qubit count of the new pool.

Layer	Previous pool	Need	Preset qubits	Cleaning	New pool
L_1	0	16	16	1	1
L_2	1	9	8	18	18
L_3	18	3	0	14	29
L_4	29	3	0	4	30
L_5	30	6	0	6	30
L_6	30	4	0	3	29
L_7	29	14	0	3	18
L_8	18	18	0	9	9
L_9	9	1	0	16	24

To ensure that the T-depth remains unchanged, we align each layer in S-box and S-box†. In each layer L_i, when S-box requires a qubit in the $|0\rangle$ state, we check if the water pool is empty. If the pool is empty, we preset a qubit in the pool, thereby increasing the size of SQ_i. After the completion of L_i, the qubits that are cleaned up by S-box† are placed back into the pool for the next layer L_{i+1}. The size of SQ_i is the sum of $q_{private}^{old}$ and the number of preset qubits. The complete qubit allocation is illustrated in Table 7, and we provide a detailed explanation below.

- Prior to L_1, there are no qubits in the pool.
- During the execution of L_1, we preset 16 qubits in S-box. Then, one qubit is cleaned up by S-box† and added to the pool.
- In L_2, we need to preset an additional 8 qubits in the pool. A total of 18 qubits are cleaned up by S-box†, resulting in a pool size of 18 qubits.
- The subsequent steps follow a similar pattern.

Table 8. Cost comparison of the cost of the combination of S-box and S-box†. Here 32 qubits represent 16-bit inputs and 16-bit outputs of two bytes.

Source	Method	Width	#CNOT	#Toffoli(AND)	Toffoli(AND) depth
[13]	Independence	$120 + 120 + 32 = 240 + 32$	428	68	4
This paper	**Combination**	$\mathbf{74 + 24 + 32 = 98 + 32}$	**312**	68	4

In total, we preset 24 qubits in the pool. Consequently, $|SQ_i| = 74+24 = 98$. Prior to executing the combination, 74 ancilla qubits are utilized, and then we allocate 24 qubits with an initial state of $|0\rangle$ in q_{public}^{old}. After the combination, there are still 24 qubits remaining in the pool for subsequent combinations (see Proposition 2). Therefore, we have $|SQ| = 98$, $|q^{old}_{public}| = 24$, $|q^{old}_{private}| = 74$, $|q^{new}_{public}| = 24$, and $|q^{new}_{private}| = 74$. We give the complete comparison in Table 8. The number of ancilla qubits is reduced from 240 to 98.

5 The Components of Quantum Circuits for AES

This section describes the quantum circuits for the AES components MixColumns, Key Schedule, AddRoundKey, and ShiftRows. We mainly introduce the improvement of the depth of MixColumns. Other components are similar to the previous work.

5.1 Implementation of MixColumns

The implementation of MixColumns has been widely studied [12,13,29]. Usually, we can use optimized classical circuits to reduce the cost (see for example [17,19,22,23,28]).

Table 9. New circuit of MixColumns with quantum depth 16. Here every number represents a qubit. `Update`(b, a) represents the `CNOT` operation $|a\rangle|b\rangle \rightarrow |a\rangle|a \oplus b\rangle$. $|y_0\rangle, |y_1\rangle, \ldots, |y_{31}\rangle$ are represented by 24, 1, 10, 11, 12, 13, 30, 15, 8, 25, 2, 3, 4, 5, 14, 7, 0, 17, 26, 19, 20, 29, 22, 31, 16, 9, 18, 27, 28, 21, 6, 23, respectively.

`Operation`	`Operation`	`Operation`	`Operation`	`Operation`	`Operation`
Depth 1	`Update`(12, 28)	`Update`(11, 27)	`Update`(5, 4)	`Update`(7, 31)	`Update`(11, 3)
`Update`(31, 23)	**Depth 3**	`Update`(13, 21)	**Depth 8**	`Update`(4, 28)	`Update`(9, 1)
`Update`(24, 8)	`Update`(7, 15)	`Update`(30, 14)	`Update`(18, 23)	**Depth 11**	**Depth 14**
`Update`(21, 29)	`Update`(8, 23)	`Update`(18, 17)	`Update`(20, 11)	`Update`(23, 6)	`Update`(22, 30)
`Update`(26, 18)	`Update`(9, 25)	`Update`(28, 23)	`Update`(19, 26)	`Update`(1, 25)	`Update`(13, 5)
`Update`(1, 17)	`Update`(4, 12)	**Depth 6**	`Update`(3, 10)	`Update`(19, 31)	`Update`(7, 15)
`Update`(11, 3)	`Update`(6, 13)	`Update`(0, 31)	`Update`(12, 7)	`Update`(17, 7)	`Update`(16, 8)
`Update`(10, 2)	`Update`(2, 17)	`Update`(14, 5)	`Update`(24, 16)	`Update`(10, 2)	`Update`(3, 27)
`Update`(28, 20)	`Update`(5, 28)	`Update`(27, 2)	**Depth 9**	`Update`(28, 20)	`Update`(1, 17)
`Update`(14, 22)	**Depth 4**	`Update`(21, 20)	`Update`(23, 30)	`Update`(26, 18)	**Depth 15**
`Update`(27, 19)	`Update`(25, 8)	`Update`(28, 19)	`Update`(26, 1)	**Depth 12**	`Update`(14, 22)
`Update`(13, 5)	`Update`(16, 7)	`Update`(17, 24)	`Update`(18, 10)	`Update`(31, 22)	`Update`(5, 21)
Depth 2	`Update`(12, 11)	**Depth 7**	`Update`(3, 7)	`Update`(25, 16)	`Update`(15, 31)
`Update`(23, 7)	`Update`(30, 13)	`Update`(0, 24)	`Update`(20, 31)	`Update`(6, 14)	`Update`(27, 19)
`Update`(25, 1)	`Update`(15, 14)	`Update`(14, 29)	`Update`(4, 12)	`Update`(19, 11)	`Update`(8, 24)
`Update`(22, 6)	`Update`(29, 28)	`Update`(27, 18)	**Depth 10**	`Update`(1, 0)	`Update`(17, 25)
`Update`(20, 4)	`Update`(17, 23)	`Update`(20, 12)	`Update`(23, 15)	**Depth 13**	**Depth 16**
`Update`(17, 9)	**Depth 5**	`Update`(2, 26)	`Update`(18, 9)	`Update`(22, 13)	`Update`(31, 23)
`Update`(29, 13)	`Update`(8, 0)	`Update`(28, 3)	`Update`(10, 1)	`Update`(16, 7)	`Update`(21, 29)

In quantum key search, the full depth of the circuit influences the time cost for Grover's search. However, the depth in classical circuits and quantum circuits is different. The main reason is that one qubit cannot be used in two gates simultaneously. The previous work merely translated the optimal classical circuits into quantum circuits. However, we believe that relaxing the gate count constraint could lead to better depth performance. Therefore, we relaxed the gate count constraint in Xiang *et al.*'s approach in [28] and generated a series of candidate circuits. Then, we can calculate their quantum depth quickly. As a result, we obtained an in-place circuit with a depth of 16 (cf. Table 9) for comparison (cf. Table 10).

Table 10. Comparison of MixColumns implementations for each column.

Source	#CNOT	Width	$\#FD$
[3,23]	206	135	13
[19]	210	137	**11**
[14]	277	32	111
[8,29]	277	32	39
[28]	92	32	30
This Paper	98	32	**16**

5.2 Implementation of the Key Schedule

In the key schedule of AES, we use several 32-bit words to save the key. For AES-128, we use 128 qubits to represent the 128-bit master key (called W_0, W_1, W_2, W_3). After executing XOR operations for these qubits and the 128-bit state, we update 128 qubits for the next AddRoundKey operation.

The schedule is similar to [12–14]. Firstly, we use four ancilla qubit set SQ_i $(0 \leq I \leq 3)$ to run SubWord, which requires $4 \times 98 = 392$ ancilla qubits. All the S-boxes in the key schedule and round function are designed to operate in parallel. SubWord† is executed in the next round to clean up the ancilla qubits, which is introduced in Sect. 4. The 32-bit output values of SubWord are XORed to $|W_{4i+0}\rangle$. The Rcon operation is implemented with X gates for the corresponding qubits to generate W_{4i+4}. Finally, $W_{4i+5}, W_{4i+6}, W_{4i+7}$ are updated by CNOT gates. Rcon is executed by adding X gates. The schedule in our pipeline architecture is similar to [12–14].

5.3 Implementation of AddRoundKey and ShiftRows

In AddRoundKey, 128 CNOT gates are required and no ancilla qubits are set. For the ShiftRows operation, the swap for qubits is a logical operation that only changes the index of qubits. Therefore, the operation does not require any gates.

6 Quantum Circuit of AES Based on the Pipelined Architecture

As mentioned in Sect. 4, the pipelined architecture is suitable for implementing low-depth AES circuits. In this section, we have made improvements with different quantum circuits based on the combined pipeline architecture using ProjectQ. The code is available at https://github.com/QunLiu-sdu/Improved-Quantum-Circuits-for-AES. Then, we provide a more precise analysis of the Grover algorithm's complexity.

6.1 Resource Estimations Based on the Toffoli Gate and and Gate

In [12,13,18,20,21,29], the authors present the quantum resources needed for the circuits without decomposing the Toffoli gates. We adhere to the same circuit metrics to facilitate a fair comparison.

Grover's algorithm requires us to estimate the complexity resulting from decomposing Toffoli gates. According to [26], the Toffoli gate can be decomposed:

– the circuit with T-depth 1 and 4 ancilla qubits;
– the circuit with T-depth 4 and 0 ancilla qubits.

On the other hand, in [13,14], the authors recommend using AND gates to achieve a lower complexity for Grover's algorithm. We also utilize AND-based decomposition to construct AES quantum circuits. In fact, the circuits constructed using this method have the lowest circuit depth. We applied these circuits to Grover's algorithm, obtaining more precise resource estimates.

The structure of each AES round is shown in Fig. 6. For each round, 20 S-boxes are executed in parallel (16 S-boxes for SubBytes and 4 S-boxes for SubWord). In R_1, we execute 20 S-boxes with 20 shared qubit sets because no S-boxes† are required. In other rounds, we execute 20 combinations of S-box and S-box†. After S-box operations, the output is saved in 128 new qubits for the ShiftRows, MixColumns, and AddRoundKey operations.

Moreover, for each AES quantum circuit, we considered two implementations for the linear layer:

– in-place circuit, which utilizes the circuit found by us with depth 16.
– out-of-place circuit, which is from [19] with depth 11.

In Table 11, the results use the Toffoli gate and do not decompose it. Our circuit achieves the optimal trade-off between Toffoli depth and width. In conclusion, compared with the previous lowest results in [21], the product of our implementations achieved a reduction of 35%, 38%, 36% for AES-128, -192, and -256, respectively. Compared with the shallowed architecture in [13], the number of qubits achieves a reduction of 42%, 41%, and 36%, respectively.

In Table 12, we show the results of decomposing Toffoli gates into circuits with T-depth 4. Compared with the shallowed architecture in [13], the number of qubits, full depth, and DW-cost achieve a reduction. The minimum depth of circuits is 770, 924, and 1074 for AES-128, -192, and -256 respectively. Table 13 also shows the results of decomposing Toffoli gates into circuits with T-depth 1. To reduce the T-depth, we employed additional qubits to implement the quantum circuit.

The circuits using the AND gates are shown in Table 14, which achieves the lower circuit depth. For the in-place version, the circuit depth achieves a reduction of 13.8%, 13.6%, and 14.3%, respectively. It is worth noting that the circuits with the AND gate and out-of-place linear layer have a lower depth. For AES-128, -192, and -256, we only require the depth of 730, 876, and 1,018, respectively.

In conclusion, for the different types of AND operations, we find that the circuits decomposed using AND gates have the lowest complexity for Grover's algorithm, which we show in Sect. 6.2. We also used the AES S-box with T-depth 3 proposed in [12]. Since the number of gates was almost doubled, we were not able to obtain high-quality Grover algorithm attack complexity.

Table 11. Comparisons of quantum resources of AES without decomposing the Toffoli gates.

Cipher	Source	#CNOT	#NOT	#Toffoli	Toffoli depth	Width	Toffoli depth × Width
AES-128	[8]	166,548	1,456	151,552	12,672	984	12,469,248
	[1]	192,832	1,370	150,528	–	976	–
	[20]	53,360	1,072	16,688	12,168	264	3,212,352
	[18]	107,960	1,570	16,940	1,880	864	1,624,320
	[29]	128,517	4,528	19,788	2,016	512	1,032,192
	[12]($p = 9$)	126,016	2,528	17,888	1,558	374	582,692
	[20]	53,496	1,072	16,664	1,472	328	482,816
	[12]($p = 18$)	126,016	2,528	17,888	820	492	403,440
	[13]	81,312	800	12,240	40	6,368	254,720
	[21]($m = 16$)	77,984	2,224	19,608	476	474	225,624
	This paper(out-of-place)	75,024	800	12,920	40	**4,823**	**192,920**
	This paper(in-place)	65,736	800	12,920	40	**3,667**	**146,680**
AES-192	[8]	189,432	1,608	172,032	11,088	1,112	12,329,856
	[20]	70,736	1,160	19,328	14,496	328	4,754,688
	[18]	125,580	1,692	19,580	1,640	896	1,469,440
	[29]	152,378	5,128	22,380	2,022	640	1,294,080
	[13]	92,856	896	14,008	48	6,688	321,024
	[21]($m = 16$)	90,832	2,568	22,800	572	538	307,736
	This paper(out-of-place)	85,808	896	14,552	48	**5,356**	**257,088**
	This paper(in-place)	74,456	896	14,552	48	**3,935**	**188,880**
AES-256	[8]	233,836	1,943	215,040	14,976	1,336	20,007,936
	[20]	74,472	1,367	23,480	17,412	392	6,825,504
	[18]	151,011	1,992	23,760	2,160	1,232	2,661,120
	[29]	177,645	6,103	26,774	2,292	768	1,760,256
	[13]	113,744	1,103	17,408	56	6,976	390,656
	[21]($m = 16$)	110,688	3,069	27,816	646	502	388,892
	This paper(out-of-place)	106,704	1,119	18,360	56	**6,097**	**341,432**
	This paper(in-place)	93,288	1,119	18,360	56	**4,429**	**248,024**

Table 12. Comparisons of quantum resources of AES decomposing Toffoli gates with T-depth 4.

Cipher	Source	#CNOT	#1qCliff	$\#T$	T-depth	Width	DW-cost	$\#FD$
AES-128	[13](out-of-place)	164,256	16,832	85,680	160	7,520	1,203,200	799
	This paper(out-of-place)	152,544	19,080	90,440	160	4,844	775,040	**770**
	[13](in-place)	154,752	14,400	85,680	160	6,368	1,018,880	978
	This paper(in-place)	143,256	19,080	90,440	160	**3,688**	**590,080**	840
AES-192	[13](out-of-place)	188,520	19,440	98,056	192	8,096	1,554,432	955
	This paper(out-of-place)	173,120	21,384	101,864	192	5,356	1,028,352	**924**
	[13](in-place)	176,904	16,400	98,056	192	6,688	1,284,096	1,174
	This paper(in-place)	161,768	21,384	101,864	192	**3,944**	**755,136**	1,010
AES-256	[13](out-of-place)	231,920	23,519	121,856	224	8,640	1,935,360	1,118
	This paper(out-of-place)	216,864	26,759	128,520	224	6,124	1,371,776	**1,074**
	[13](in-place)	218,192	19,871	121,856	224	6,976	1,562,624	1,377
	This paper(in-place)	203,448	26,759	128,520	224	**4,456**	**998,144**	1,176

Table 13. Comparisons of quantum resources of AES decomposing Toffoli gates with T-depth 1.

Cipher	Source	#CNOT	#1qCliff	$\#T$	T-depth	Width	DW-cost	$\#FD$
AES-128	**This paper**(out-of-place)	281,744	19,080	90,440	40	4,844	193,760	**750**
	This paper(in-place)	272,456	19,080	90,440	40	**3,691**	**147,640**	820
AES-192	**This paper**(out-of-place)	318,640	21,384	101,864	48	5,356	257,088	**900**
	This paper(in-place)	307,288	21,384	101,864	48	**3,947**	**189,456**	986
AES-256	**This paper**(out-of-place)	400,464	26,759	128,520	56	6,124	342,944	**1,046**
	This paper(in-place)	387,048	26,759	128,520	56	**4,459**	**249,704**	1,148

Table 14. Comparisons of quantum resources with the AND gates.

Cipher	Source	#CNOT	#1qCliff	$\#T$	$\#M$	T-depth	Width	DW-cost	$\#FD$
AES-128	[13](out-of-place)	152,496	39,952	27,200	5,440	40	7,524	300,960	749
	This paper(out-of-place)	141,664	51,800	27,200	6,120	40	**4,844**	**193,760**	**730**
	[13](in-place)	142,992	37,520	27,200	5,440	40	6,372	254,880	928
	This paper(in-place)	132,376	51,800	27,200	6,120	40	**3,689**	**147,560**	**800**
AES-192	[13](out-of-place)	174,152	46,232	30,464	6,392	48	8,100	388,800	895
	This paper(out-of-place)	160,608	58,424	30,464	6,936	48	**5,356**	**257,088**	**876**
	[13](in-place)	162,536	43,192	30,464	6,392	48	6,692	321,216	1,114
	This paper(in-place)	149,256	58,424	30,464	6,936	48	**3,945**	**189,360**	**962**
AES-256	[13](out-of-place)	213,624	56,975	37,536	8,024	56	8,644	484,064	1,048
	This paper(out-of-place)	200,544	73,879	38,080	8,840	56	**6,124**	**342,944**	**1,018**
	[13](in-place)	199,896	53,327	37,536	8,024	56	6,980	390,880	1,307
	This paper(in-place)	187,128	73,879	38,080	8,840	56	**4,457**	**249,592**	**1,120**

6.2 Performance of Grover's Algorithm

In this part, based on the circuits implemented using AND gates and the out-of-place linear layer, we applied these AES circuits to Grover's algorithm, obtaining more precise and efficient resource estimates (cf. Table 15). "$r = \lceil k/n \rceil$" (plaintext, ciphertext) is the number of pairs that are required to recover a unique key. For AES-128, we just choose $r = 1$. For AES-192/-256, we choose $r = 2$. It implies that we need to use two plaintext-ciphertext pairs to determine the key. In terms of resources corresponding to Grover's algorithm, one key expansion algorithm corresponds to two round functions. We primarily focus on four metrics, $FD \times G$, $FD \times W$, $FD^2 \times G$, and $FD^2 \times W$.

Table 15. Quantum resources required for Grover's search on AES.

Cipher	r	Source	Width (W)	Gates (G)	$\#FD$	$\boldsymbol{FD \times G}$	$\boldsymbol{FD \times W}$	$\boldsymbol{FD^2 \times G}$	$\boldsymbol{FD^2 \times W}$
AES-128	1	[14]	1.92×2^{11}	1.33×2^{82}	1.08×2^{75}	1.436×2^{157}	1.038×2^{87}	1.551×2^{232}	1.120×2^{162}
		[13]	1.84×2^{12}	1.36×2^{82}	1.15×2^{74}	1.564×2^{156}	1.055×2^{87}	1.797×2^{230}	1.212×2^{161}
		This paper	1.18×2^{12}	1.37×2^{82}	1.12×2^{74}	$\mathbf{1.535 \times 2^{156}}$	$\mathbf{1.325 \times 2^{86}}$	$\mathbf{1.719 \times 2^{230}}$	$\mathbf{1.480 \times 2^{160}}$
AES-192	2	[14]	1.02×2^{13}	1.50×2^{115}	1.14×2^{107}	1.710×2^{222}	1.163×2^{120}	1.949×2^{239}	1.326×2^{227}
		[13]	1.84×2^{13}	1.45×2^{115}	1.37×2^{106}	1.988×2^{221}	1.261×2^{120}	1.365×2^{328}	1.731×2^{226}
		This paper	1.24×2^{13}	1.44×2^{115}	1.35×2^{106}	$\mathbf{1.944 \times 2^{221}}$	$\mathbf{1.679 \times 2^{119}}$	$\mathbf{1.312 \times 2^{328}}$	$\mathbf{1.130 \times 2^{226}}$
AES-256	2	[14]	1.09×2^{13}	1.84×2^{147}	1.29×2^{139}	1.187×2^{287}	1.401×2^{152}	1.531×2^{426}	1.814×2^{291}
		[13]	1.96×2^{13}	1.74×2^{147}	1.61×2^{138}	1.398×2^{286}	1.576×2^{152}	1.123×2^{425}	1.266×2^{291}
		This paper	1.43×2^{13}	1.76×2^{147}	1.56×2^{138}	$\mathbf{1.373 \times 2^{286}}$	$\mathbf{1.117 \times 2^{152}}$	$\mathbf{1.071 \times 2^{425}}$	$\mathbf{1.740 \times 2^{290}}$

7 Quantum Circuit of AES Based on the Zig-Zag Architecture

In this section, we propose AES circuits based on the zig-zag architecture utilizing AND gates. The zig-zag architecture is typically employed for low-width implementations.

At ASIACRYPT 2022, Huang *et al.* [12] aimed to reduce the DW-cost in the zig-zag architecture by introducing low-depth S-boxes based on AND gates. They proposed an improved zig-zag architecture based on the round-in-place technique. The S-box requires 120 ancilla qubits with T-depth 4, resulting in a minimum DW-cost of 204,800 (width $\times$ T-depth $= 2,560 \times 80$) for AES-128. We notice that several papers have adopted similar circuits based on the round-in-place technique to optimize the quantum circuit for AES (cf. [21]).

Our approach begins by introducing the zig-zag architecture and round-in-place technique. We then highlight the advantages of utilizing the *round-in-place zig-zag* architecture iteratively, offering an improved trade-off between width and T-depth. Subsequently, we propose a new circuit for the AES S-box that significantly reduces the required ancilla qubits to just $60 + 10$. By employing this optimized S-box in the circuits in [12], we achieve a substantial reduction in DW-cost, resulting in a final value of 132,800 (width $\times$ T-depth $= 1,660 \times 80$).

7.1 Zig-Zag Architecture and Round-in-Place Technique in [12]

The zig-zag architecture is proposed in [8], which reduces the number of qubits by performing reverse operations in each round (cf. Fig. 8). R_1, R_2, R_3, and R_4 are performed in order. Then, $R_3^\dagger$, $R_2^\dagger$, and $R_1^\dagger$ are utilized to clean up the 4-*th*, 3-*rd*, and 2-*nd* lines, which can be reused to store the outputs of R_7, R_6, and R_5, respectively. Other rounds proceed similarly. This method requires a larger T-depth. Subsequently, at ASIACRYPT 2020, Zou *et al.* [29] improved the zig-zag architecture and implemented AES-128 with 512 qubits and Toffoli depth 2016.

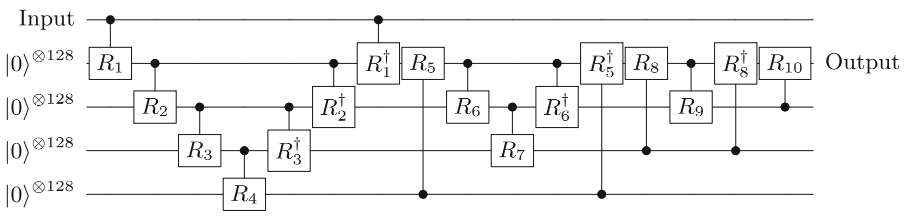


Fig. 8. Zig-zag architecture.

Huang *et al.* [12] improved the zig-zag architecture based on a round-in-place technique. We simply introduce the technique. Suppose U_f is a circuit that map $|x\rangle|0\rangle|0\rangle^{\otimes a}$ to $|x\rangle|S(x)\rangle|0\rangle^{\otimes a}$, where $|0\rangle^{\otimes a}$ denotes the ancilla qubits, then, U_f is not *in-place*. If we also have the inverse circuit $U_{f^{-1}}$ that maps $|S(x)\rangle|x\rangle|0\rangle^{\otimes a}$ to $|S(x)\rangle|0\rangle|0\rangle^{\otimes a}$, we can achieve the *in-place* circuit by swapping $|x\rangle$ and $|S(x)\rangle$. Figure 9 shows a round-in-place S-box circuit, which maps $|x\rangle|0\rangle|0\rangle^{\otimes a}$ to $|S(x)\rangle|0\rangle|0\rangle^{\otimes a}$.

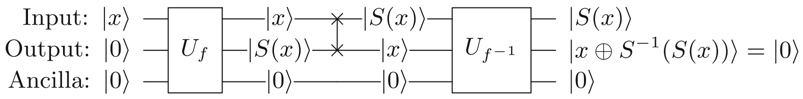


Fig. 9. Round-in-place S-box circuit.

Usually, U_f is easy to construct. Huang *et al.* [12] provide a method to convert U_f into $U_{f^{-1}}$. Suppose $x, y \in \mathbb{F}_2^8$ are the input and output of U_f. we have $y = LS_0(x) + c$, where L is a linear function and $S_0(x)$ is the inverse of x in $\mathbb{F}_2^8$. Then, $x = S_0^{-1}L^{-1}(y + c) = S_0L^{-1}(y + c) = L^{-1}(LS_0)L^{-1}(y + c)$. Let the last 4 X gates of AES S-box be U_c. $U_f = U_0 + U_c$, where U_0 implements $|x\rangle|0\rangle|0\rangle \rightarrow |x\rangle|LS_0(x)\rangle|0\rangle$. Then, the circuit in Fig. 10 is $U_{f^{-1}}$, where U_L and $U_{L^{-1}}$ are the circuits of L and L^{-1}, respectively. Huang *et al.* provide an SAT-based method and implement U_L or $U_{L^{-1}}$ by 14 CNOT gates. Finally, adding $14 \times 3 = 42$ CNOT gates and 4 X gates, U_f can be converted into $U_{f^{-1}}$.

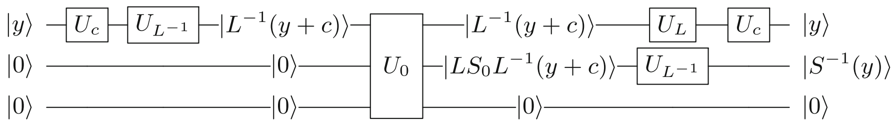


Fig. 10. The circuit of $U_{f^{-1}}$ based on U_f.

Based on the *round-in-place* S-box circuit, one can construct the *round-in-place* round function R_i easily. Figure 11 shows the round function. SubByte^1 uses S-box circuits. SubByte^{-1} uses S-box^{-1} circuits. SB, MC, and ARK represent the *in-place* ShiftRows, MixColumns, and AddRoundKey, respectively. The circuit maps $|x\rangle\,|0\rangle\,|0\rangle^{\otimes a}$ to $|R(x)\rangle\,|0\rangle\,|0\rangle^{\otimes a}$.

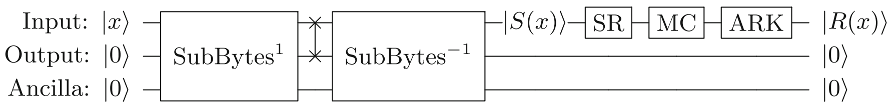


Fig. 11. Round-in-place round function of AES.

7.2 Executing the Round-in-Place Round Function Iteratively

In [12,21], the authors provide a method for executing the round-in-place round function iteratively. We simply introduce the technique. For AES, there are 16 bytes for the state. Each byte requires an ancilla qubit set to run round-in-place S-box circuits. If only m S-boxes of SubBytes are executed in parallel, then $16/m$ ancilla qubit sets are needed. We define the number of iterations as $i = 16/m$ ($i = 1, 2, 4, 8, 16$).

For $i = 1, 2$, the key schedule can be split. In the key schedule, four S-boxes are needed in each round. However, they do not require a round-in-place S-box because $|W_{4i+0}\rangle$ contains the output values of SubWord. Thus, the schedule only uses the U_f mapping $|x\rangle\,|y\rangle\,|0\rangle^{\otimes a}$ to $|x\rangle\,|y \oplus S(x)\rangle\,|0\rangle^{\otimes a}$. The S-box used in the schedule is single-depth, while the round-in-place S-box in the round function is double-depth. We can split the SubWord in the key schedule into two parts (cf. Fig. 12). SubWord$_{\frac{1}{2}}$ indicates that only half of SubWord is used in the operations. The first part and SubByte^1 are executed in parallel. The second part and SubByte^{-1} are executed in parallel. For more detailed information, refer to [12,21].

For $i = 4, 8, 16$, we execute two operations in parallel (cf. Fig. 13). We take $i = 4$ as an example. SB$_{\frac{1}{4}}$ represents a quarter of SubBytes and SW$_{\frac{1}{4}}$ represents a quarter of SubWord. It requires five ancilla qubit sets and the T-depth is four double-depth S-boxes.

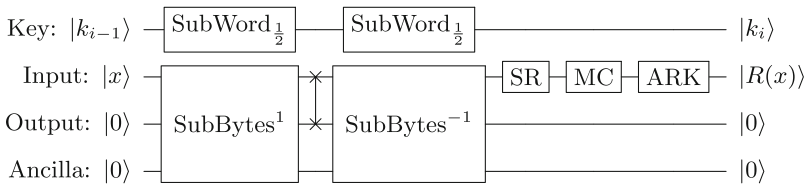


Fig. 12. Two parts in round function for $i = 1$.

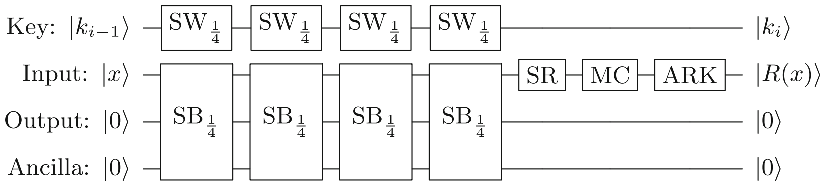


Fig. 13. Round-in-place round function executing SubBytes and SubWord in parallel for $i = 4$.

7.3 Constructing a Low-Width S-Box Circuit

To improve the circuit, we constructed a new AES S-box circuit with only 60+10 ancilla qubits. The circuit is suitable for the *round-in-place* zig-zag architecture with AND gates.

Next, we present this construction method, which always allows the use of the minimum number of qubits without increasing the T-depth. Our approach first satisfies the maximum parallel count of AND gates in different layers.

- For the structure of AES S-box with T-depth 4 in [12], 8 input qubits $u_0, \ldots, u_7$, and 8 output qubits $s_0, \ldots, s_7$.
- We consider that the target qubit of each AND gate is allocated as $|0\rangle$. $q_0, q_1, \ldots, q_{33}$ are allocated for each target qubit. There are 9, 3, 4, and 18 target qubits in T-depth 1, 2, 3, and 4, respectively.
- Because the layer in T-depth 4 contains the most AND gates, we must satisfy its parallelism first. Thus, $q_{34}, q_{35}, \ldots, q_{50}$ are allocated as the inputs of these AND gates.
- However, the AND gates in T-depth 4 still cannot be implemented in parallel. The main reason is that nine qubits are included in two AND gates at the same time.
- We have to allocate nine qubits $q_{51}, q_{52}, \ldots, q_{59}$ to restore these qubits. Thus, there are 60 ancilla qubits $q_0, q_1, \ldots, q_{59}$ in T-depth 4. Note that $q_{51}, q_{52}, \ldots, q_{59}$ can be reset as $|0\rangle$ after these AND gates and be used in other AND gates.
- In other layers, AND gates do not require any more ancilla qubits. The lower bound on the number of ancilla qubits is 60.
- For the 4-*th* AND layer, 18 ancilla qubits are required for the AND gates. Apart from 8 qubits from $s_0, s_1, \ldots, s_7$, we need to allocate 10 ancilla qubits.

– The final number of ancilla qubits of the S-box is 70.

After executing the S-box, we need to execute S-box† to clean up the ancilla qubits, which corresponds to U_f in Fig. 9. We estimate the resource of S-box and S-box† (cf. Table 16). Compared with the S-box circuit with 83 ancilla qubits, the new circuit requires more gates and depth. Therefore, this S-box circuit does not optimize the complexity of Grover's algorithm.

Table 16. Implementation of U_f (S-box and S-box†) based on the AND gates. Here #M counts the number of Measurements.

Width	#CNOT	#1qCliff	#T	#M	T-depth	Full depth
60+10+16	688	220	136	34	4	132

Figure 10 shows how to transform U_f into $U_{f^{-1}}$, which requires 42 additional CNOT gates and 4 X gates based on U_f. Thus, $U_{f^{-1}}$ requires $688 + 42 = 730$ CNOT gates and $220 + 4 = 224$ 1qClifford gates.

In the previous work [12,29], there are two types of S-boxes. The first type is used in SubBytes, called the $\mathcal{C}^0$-circuit, which maps $|x\rangle |0\rangle |0\rangle^{\otimes a}$ to $|x\rangle |S(x)\rangle |0\rangle^{\otimes a}$. The second type is used in SubWord, called the $\mathcal{C}^*$-circuit, which maps $|x\rangle |y\rangle |0\rangle^{\otimes a}$ to $|x\rangle |y \oplus S(x)\rangle |0\rangle^{\otimes a}$. We follow a unified principle to design the circuit of the AES S-box. If the output qubits $s_0, s_1, \ldots, s_7$ are only used to save the output, the S-box is suitable for both $\mathcal{C}^0$- and $\mathcal{C}^*$-circuits. Note that for the $\mathcal{C}^*$-circuit, 18 ancilla qubits are also required for the AND gates. As 8 qubits from $s_0, s_1, \ldots, s_7$ can not be used. we need to allocate 18 ancilla qubits.

7.4 Applying New S-Box Circuit into the Architecture in [12]

With reference to Fig. 12, we can calculate the resources for each round of AES. We take AES-128 as an example. Note that there is no MixColumns operation in the last round of AES. In SubBytes^1, there are 16 U_f. In SubBytes^{-1}, there are 16 $U_{f^{-1}}$. In MixColumns, there are $92 \times 4 = 368$ CNOT gates. In AddRoundKey, there are 128 CNOT gates. In the Key Schedule, there are 4 U_f, and at most 4 X gates for Rcon. Therefore, one round of AES requires 20 U_f, 16 $U_{f^{-1}}$, $368 + 128 = 496$ CNOT gates, and at most 4 X gates for Rcon. AES-128 requires $10 \times 20 = 200$ U_f, $10 \times 16 = 160$ $U_{f^{-1}}$, $10 \times 496 - 368 = 4592$ CNOT gates, and $8 \times 1 + 2 \times 4 = 16$ X gates.

Next, based on the number of iterations i ($i = 1, 2, 4, 8, 16$), different trade-offs of width/T-depth are provided in Table 17. We describe how to calculate the number of ancilla qubits and T-depth.

For the case of $i = 1, 2$, we execute the two-part round function (cf. Fig. 12), which allows the key schedule to require only $\frac{2}{i} \times (60 + 18)$ ancilla qubits. In round function, SubBytes^1 and SubBytes^{-1} require $\frac{16}{i} \times (60 + 10)$ ancilla qubits. The number of ancilla qubits is $\frac{2}{i} \times 78 + \frac{16}{i} \times 70$. Then, 256 input qubits and

Table 17. Different trade-offs of width/T-depth for quantum circuit of AES-128.

AES-128	Width	T-depth	DW-cost
$i = 1$	$256 + 128 + 156 + 1120 = 1660$	$80 \times 1 = 80$	**132,800**
$i = 2$	$256 + 64 + 78 + 560 = 958$	$80 \times 2 = 160$	153,280
$i = 4$	$256 + 32 + 78 + 280 = 646$	$80 \times 4 = 320$	206,720
$i = 8$	$256 + 16 + 78 + 140 = 490$	$80 \times 8 = 640$	313,600
$i = 16$	$256 + 8 + 78 + 70 = 412$	$80 \times 16 = 1280$	527,360

$\frac{128}{i}$ output qubits are needed. For each round, the T-depth is $4 \times 2 \times i = 8i$. Therefore, T-depth of AES-128 is $10 \times 8i = 80i$.

For the case of $i = 4, 8, 16$, we execute the round function like Fig. 13. The key schedule requires 78 ancilla qubits. In round function, SubBytes[1] and SubBytes^{-1} require $\frac{16}{i} \times 70$ ancilla qubits. The number of ancilla qubits is $78 + \frac{16}{i} \times 70$. Then, 256 input qubits and $\frac{128}{i}$ output qubits are needed. The T-depth is $10 \times i \times 2 \times 4 = 80i$.

Furthermore, we can also make an interleaved execution of the S-boxes between key schedule and round functions, reducing the number of additional qubits by increasing the T-depth (cf. Fig. 14). However, this method does not affect the lowest DW-cost. Therefore, we compared this approach with the circuits used in [12] in Fig. 1.

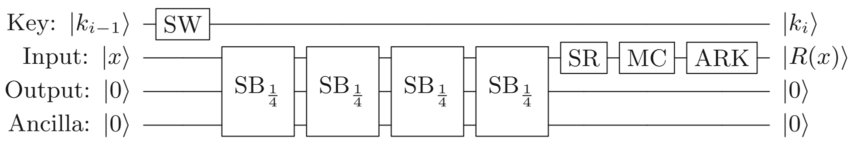


Fig. 14. Round-in-place round function executing SubBytes and SubWord serially for $i = 4$.

Finally, AES-128 can be implemented by the *round-in-place* round function with the lower DW-cost 132,800, while the previous best result is 204,800 in [12]. For AES-192, we achieve a circuit with DW-cost (width $\times$ T-depth) $1,724 \times 96 = 165,504$. For AES-256, we achieve a circuit with DW-cost (width $\times$ T-depth) $1,788 \times 112 = 200,256$.

8 Conclusion

In this paper, we investigated the optimization of quantum circuits for AES variants (-128, -192, -256). We provided an improved structure of the S-box based on the m-XOR technique and combined the S-box and S-box† based on the share technique. Then, we introduce the implementations of the AES components. Next, we estimated the required resources based on the pipelined and zig-zag architectures with Toffoli gates and AND gates. The combined pipeline architecture reduces the depth and the number of qubits required for various quantum circuits. Although our circuits perform well in the quantum case, we believe that further improvements are possible by exploiting the structure of the S-box. If a superior S-box circuit is proposed, our method can be immediately applied to reduce the complexity of the AES quantum circuit.

Acknowledgements. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant No. 62032014), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025), Quan Cheng Laboratory (Grant No. QCLZD202306).

References

1. Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of AES-128. Quantum Inf. Process. **17**(5), 1–30 (2018)
2. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(6), 818–830 (2013). https://doi.org/10.1109/TCAD.2013.2244643
3. Banik, S., Funabiki, Y., Isobe, T.: Further results on efficient implementations of block cipher linear layers. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **104-A**(1), 213–225 (2021). https://doi.org/10.1587/transfun.2020CIP0013
4. Brylinski, J.L., Brylinski, R.: Universal quantum gates. In: Mathematics of Quantum Computation, pp. 117–134. Chapman and Hall/CRC, Boca Raton (2002)
5. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), 2nd edn. Information Security and Cryptography, Springer, Cham (2020). https://doi.org/10.1007/978-3-662-60769-5
6. DiVincenzo, D.P.: Quantum gates and circuits. Proc. Roy. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **454**, 261–276 (1998)
7. Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N.: Surface codes: towards practical large-scale quantum computation. Phys. Rev. A **86**, 032324 (2012). https://doi.org/10.1103/PhysRevA.86.032324
8. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (ed.) PQCrypto 2016. LNCS, vol. 9606, pp. 29–43. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29360-8_3

9. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, 22–24 May 1996, pp. 212–219. ACM (1996). https://doi.org/10.1145/237814.237866
10. Hanks, M., Estarellas, M.P., Munro, W.J., Nemoto, K.: Effective compression of quantum braided circuits aided by ZX-calculus. Phys. Rev. X **10**(4), 041030 (2020)
11. Häner, T., Steiger, D.S., Svore, K., Troyer, M.: A software methodology for compiling quantum programs. Quantum Sci. Technol. **3**(2), 020501 (2018). https://doi.org/10.1088/2058-9565/aaa5cc
12. Huang, Z., Sun, S.: Synthesizing quantum circuits of AES with lower T-depth and less qubits. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 614–644. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22969-5_21
13. Jang, K., Baksi, A., Song, G., Kim, H., Seo, H., Chattopadhyay, A.: Quantum analysis of AES. IACR Cryptololgy ePrint Archive, p. 683 (2022)
14. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing Grover oracles for quantum key search on AES and LowMC. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 280–310. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_10
15. Jean, J., Peyrin, T., Sim, S.M., Tourteaux, J.: Optimizing implementations of lightweight building blocks. IACR Trans. Symm. Cryptol. **2017**(4), 130–168 (2017). https://doi.org/10.13154/tosc.v2017.i4.130-168
16. Kim, P., Han, D., Jeong, K.C.: Time-space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2. Quantum Inf. Process. **17**(12), 339 (2018). https://doi.org/10.1007/s11128-018-2107-3
17. Kranz, T., Leander, G., Stoffelen, K., Wiemer, F.: Shorter linear straight-line programs for MDS matrices. IACR Trans. Symm. Cryptol. **2017**(4), 188–211 (2017). https://doi.org/10.13154/tosc.v2017.i4.188-211
18. Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing the Advanced Encryption Standard as a quantum circuit. IEEE Trans. Quantum Eng. **1**, 1–12 (2020). https://doi.org/10.1109/TQE.2020.2965697
19. Li, S., Sun, S., Li, C., Wei, Z., Hu, L.: Constructing low-latency involutory MDS matrices with lightweight circuits. IACR Trans. Symm. Cryptol. **2019**(1), 84–117 (2019). https://doi.org/10.13154/tosc.v2019.i1.84-117
20. Li, Z., Gao, F., Qin, S., Wen, Q.: New record in the number of qubits for a quantum implementation of AES. Front. Phys. **11**, 1171753 (2023)
21. Lin, D., Xiang, Z., Xu, R., Zhang, S., Zeng, X.: Optimized quantum implementation of AES. Cryptology ePrint Archive (2023)
22. Lin, D., Xiang, Z., Zeng, X., Zhang, S.: A framework to optimize implementations of matrices. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 609–632. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75539-3_25
23. Liu, Q., Wang, W., Fan, Y., Wu, L., Sun, L., Wang, M.: Towards low-latency implementation of linear layers. IACR Trans. Symm. Cryptol. **2022**(1), 158–182 (2022). https://doi.org/10.46586/tosc.v2022.i1.158-182
24. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information, 10th Anniversary edn. Cambridge University Press, Cambridge (2016)
25. Q#, M.: Quantum development. https://devblogs.microsoft.com/qsharp/
26. Selinger, P.: Quantum circuits of t-depth one. CoRR abs/1210.0974 (2012). arxiv.org/abs/1210.0974

27. Steiger, D.S., Häner, T., Troyer, M.: ProjectQ: an open source software framework for quantum computing. Quantum **2**, 49 (2018). https://doi.org/10.22331/q-2018-01-31-49
28. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. IACR Trans. Symm. Cryptol. **2020**(2), 120–145 (2020). https://doi.org/10.13154/tosc.v2020.i2.120-145
29. Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of AES with fewer qubits. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 697–726. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_24