Optimizing the depth of quantum implementations of linear layers

Chengkai Zhu[1,2][0000−0001−5250−5885] and Zhenyu Huang[1,2][0000−0002−3499−538X]

[1] SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
zhuchengkai@iie.ac.cn, huangzhenyu@iie.ac.cn

Abstract. Synthesis and optimization of quantum circuits are important and fundamental research topics in quantum computation, due to the fact that qubits are very precious and decoherence time which determines the computation time available is very limited. Specifically in cryptography, identifying the minimum quantum resources for implementing an encryption process is crucial in evaluating the quantum security of symmetric-key ciphers. In this work, we investigate the problem of optimizing the depth of quantum circuits for linear layers while utilizing a small number of qubits and quantum gates. To this end, we present a framework for the implementation and optimization of linear Boolean functions, by which we significantly reduce the depth of quantum circuits for many linear layers used in symmetric-key ciphers without increasing the gate count.

Keywords: Quantum Circuit · Reversible Circuit · Linear depth · Symmetric-key ciphers.

1 Introduction

With the rapid development of quantum technologies and quantum algorithms such as Grover's algorithm, Simon's algorithm, and Shor's algorithm, the security of modern cryptography has been challenging. It is widely known that Grover's algorithm [22] has a square root speedup over a classical algorithm in terms of the problem of database search, which can be applied to find the key for a symmetric cipher instead of a classical exhaustive key search. Moreover, quantum attacks on symmetric-key schemes are extensively studied these years, including Simon's period-finding algorithm [25,12] and other attacks derived from cryptanalytic techniques [14,13,23,30]. All these works imply that there would be a potential quantum threat to our symmetric encryption system used today.

To actually implement a quantum key search on a symmetric-key encryption scheme, e.g., a block cipher, the encryption process is supposed to be implemented as a *Grover oracle*, meaning that we should be capable of constructing a quantum circuit for the specific encryption algorithm. Meanwhile, in the call

for proposals for the standardization of post-quantum cryptography, the National Institute of Standards and Technology (NIST) makes the complexities of the quantum circuit for AES standards to categorize the security strength of post-quantum public-key schemes. All these reasons give rise to the growing appeals for studying the quantum implementation of quantum oracles of iterative symmetric-key ciphers as well as how to optimize the implementation. This has been an important and fruitful topic recently, which helps understand the quantum security of current encryption schemes and guides future post-quantum encryption designs.

Although the circuit implementations of symmetric-key ciphers differ from each other, a recurring theme can be recognized, which is to construct the quantum circuit for each building block of the cipher separately. Then we can do post-optimizations for the circuit to reduce the quantum cost, including the depth, the width (the number of qubits), and the gate count (the number of quantum gates). For the non-linear building blocks, most work has been focusing on reducing the T-depth due to its importance in fault-tolerant quantum computation [21]. The circuits that implement the linear building blocks are called *linear reversible circuits*, which only consist of CNOT gates. They have many important applications in quantum computation, e.g., stabilizer circuits.

Related work. Work in quantum implementation of symmetric ciphers mostly focuses on AES due to its popularity and importance. In 2015, Grassl et al. [21] first proposed a quantum circuit of AES and found that the number of logical qubits required to implement a Grover attack on AES is around 3000 to 7000. Followed by their work, Almazrooie et al. [4] gave a more detailed circuit of AES trying to use fewer qubits. In [33] Langenberg et al. presented an improved quantum circuit for the S-box of AES which reduced the numbers of Toffoli gates and qubits. In [49], Zou et al. constructed two quantum circuits for AES S-box and S-box^{-1}, trying to use fewer qubits as well. Except for these works primarily focusing on the number of qubits, Jaques et al. in [39] build circuits for AES and LowMC with the primary goal of reducing the circuit depth. Recently, Huang and Sun [24] proposed a general structure for implementing quantum circuits for the round functions of block ciphers. They utilized some techniques to give the state-of-the-art synthesis of AES, with respect to depth-width trade-offs. Not surprisingly, their strategy for efficient quantum circuit synthesis is also to build linear and non-linear cryptographic building blocks separately. The depth of the linear block is not considered in the first place in their work.

Apart from the efficient quantum implementation of symmetric-key ciphers, the problem of quantum circuit optimization has been studied for many years in the field of synthesis and optimization of reversible logic circuits [48,42,36,38,5], which is historically motivated by theoretical research in low-power electronics transforms in cryptography and computer graphics. The basic task is to use reversible gates to implement a reversible Boolean function, i.e., a permutation. There have been enormous algorithmic paradigms such as search-based, cycle-based, transformation-based, and BDD-based for reversible circuit synthe-

sis, both exact and heuristic. One may refer to [38] for a detailed review. Also, there are some tools developed to study the synthesis of reversible circuits [46].

Specifically for the synthesis and optimization of linear quantum circuits (the CNOT circuits), traditional methods usually yield a circuit with $O(n^2)$ gates based on standard row reduction methods such as Gaussian elimination and LU-decomposition for an $n \times n$ matrix. In [37], Patel et al. present an algorithm that uses $O\left(n^2/\log(n)\right)$ gates, which is the theoretical lower bound, to build an n-qubit linear quantum circuit. This will trivially give a bound of $O\left(n^2/\log(n)\right)$ on the circuit depth. Furthermore, Jiang et al.[28] reduce this bound by a factor of n, achieving an asymptotically optimal depth bound of $O\left(n/\log(n)\right)$. Some other efforts were also made to achieve a more compact circuit of linear layers [17,18]. The synthesis of the CNOT circuits has direct applications to the synthesis of stabilizer circuits, an important class of quantum circuits introduced by Aaronson and Gottesman [2]. However, there is still a lack of practical and efficient strategies for the optimal implementation of the linear components.

Our contribution. In this work, we first revisit the problem of optimization of a subclass of quantum circuits - CNOT circuits. We give three characterizations of the CNOT circuit depth, i.e., sequence depth, move-equivalent depth, and exchange-equivalent depth. Based on that, we focus on the problem of minimizing the circuit depth while maintaining the gate count of a gate-count-optimized CNOT circuit.

We present a practical and efficient framework in Algorithm 3 for the implementation of linear operators as well as the optimization of their circuits. Consequently, one can construct the quantum circuit for a linear Boolean function with a small number of CNOT gates and lower circuit depth. For a linear Boolean function with n-variables, our depth optimization procedure yields a complexity of $O\left(n^4/\log(n)^2\right)$.

We finally showcase the strength of our framework in quantum implementations of symmetric-key ciphers. For different linear layers used in symmetric-key ciphers, which corresponds to some invertible matrices, our method can always give a considerable reduction in the circuit depths of their implementations, and obtain the state-of-the-art quantum circuits for those linear layers. Notably, for non-invertible linear transformations that appear in the non-linear building blocks or other more complex circuit structures, our method can also make the circuits for these linear parts more compact hence reducing the depth of the whole circuit.

2 Preliminaries

2.1 Quantum circuit

Among the various alternative models used to represent a quantum computer, the circuit model is arguably the most widely used. In the circuit model of

quantum computation, a qubit (quantum bit) is a theoretically abstract mathematical object. It has two possible states $|0\rangle$ and $|1\rangle$ that are usually called basis states just as a classical bit has a state of either 0 or 1. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ and $|1\rangle$. It can be a linear combination of basic states, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

Geometrically, $|0\rangle$ and $|1\rangle$ can be represented as two-dimensional vectors $|0\rangle = [1,0]^T$ and $|1\rangle = [0,1]^T$. And $|\psi\rangle$ is described by a unit vector in a two-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^2$ of which $|0\rangle$ and $|1\rangle$ are known as computational basis states. A system of n-qubits, also called an n-qubit register, has states described by a unit vector in the Hilbert space $\mathcal{H}_n \cong \mathcal{H}^{\otimes n}$. Based on this, the evolution of quantum states is described by unitary transformations or quantum gates. A quantum gate acting on n qubits is represented by a $2^n \times 2^n$ unitary matrix. For instance, a NOT gate will invert the qubit and has a matrix form $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. There are some important single-qubit gates like the Hadamard gate H, S gate, T gate:

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

For two-qubit gates, one of the most important ones is the CNOT gate which is a two qubits (control and target) operation.

$$\text{CNOT} : |x_1\rangle|x_2\rangle \to |x_1\rangle|x_1 \oplus x_2\rangle. \tag{1}$$

It is worth mentioning that the CNOT gate, H gate, and S gate generate the Clifford group and Clifford + T forms a universal gate library. Whereas, in the field of cryptography and quantum combined, one mostly works with the Boolean functions, thus is more interested in a gate set called CNTS gate library, consisting of the CNOT, NOT, Toffoli, and SWAP gates.

From (1), we can see that a CNOT gate can be seen as an invertible linear transformation $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ over $\mathbb{F}_2^2$. Similarly, for an n-qubit system, we can characterize a CNOT gate with an $n \times n$ matrix instead of its $2^n \times 2^n$ unitary matrix form for its linearity. In detail, a CNOT gate controlled by the j-th qubit, acting on the i-th qubit ($i \neq j$) can be written as

$$E_{ij} = I + e_{ij}, \tag{2}$$

where I is the $n \times n$ identity matrix and e_{ij} the elementary matrix with all entries equal 0 but the entry (i, j) equals 1. In fact, E_{ij} belongs to the *elementary matrices* in linear algebra or matrix theory. Recall that there are three types of elementary matrices, which correspond to three types of row operations (respectively, column operations) and may be interpreted as quantum gates:

1. Row switching: interchange two rows, which will be implemented by renaming or switching the circuit wires[3].
2. Row multiplication: multiply a row with a nonzero number, which is trivial in $\mathbb{F}_2$ and not concerned in this paper.
3. Row addition: add a row to another one multiplied by a nonzero number, which in $\mathbb{F}_2$ will be interpreted as a CNOT gate.

It is a well-known theorem that any linear reversible matrix can be decomposed as a product of elementary matrices. All of these indicate that for any linear Boolean function with n variables, we can use a sequence of CNOT gates on n qubits, referred as a CNOT circuit with n qubits, to implement it.

Theorem 1. *Any invertible matrix A can be decomposed as a product of elementary matrices.*

Two metrics, named width and depth, are often used to characterize the cost of a quantum circuit. The width refers to the number of qubits that comprise the circuit and depth refers to the number of layers of gates that are not executed at the same time. Width and depth are both limiting factors in the execution of quantum algorithms. For CNOT circuits, since it can be easily implemented without auxiliary qubits, hence minimizing its width is an easy problem. Therefore, in this paper, we focus on optimizing the depth of CNOT circuits, by which we can reduce the depth of the quantum implementations of linear components of symmetric-key ciphers.

2.2 Depth of the quantum circuits

For the depth of a quantum circuit, one usually refers to the minimal number of stages the hardware needs to execute the gates when we suppose that the gates acting on different qubits are executed simultaneously. For instance, the following circuit (a) in Fig. 1 has depth 3 and the circuit (b) in Fig. 1 has depth 2 since the second and the last CNOT gates on the right circuit can be executed simultaneously. However, when the quantum circuit consists of different kinds

(a) CNOT circuit with depth 3 (b) CNOT circuit with depth 2

Fig. 1. The quantum CNOT circuits with different depth.

[3] This operation can also be implemented by 3 CNOT gates, but this will cost more quantum resources since we think the cost of rewiring is free in most cases.

of gates and we want to reduce the depth for a particular gate, e.g., the Toffoli depth that refers to the number of stages the hardware needs to execute the Toffoli gates simultaneously, the definition of depth we mentioned should be fixed.

(a) Toffoli depth 2 and circuit depth 2 (b) Toffoli depth 1 and circuit depth 3

Fig. 2. The quantum circuits with different Toffoli depth.

For instance, the two circuits illustrated in Fig. 2 contain the same gates. In circuit (b), in order to execute the two Toffoli gate in parallel, the two CNOT gates should be executed in another two different stages, which makes the circuit has depth 3.

This example shows that the depth of a circuit should be defined based on its building structure. The key issue here is that quantum circuits are written such that the horizontal axis is time, starting at the left-hand side and ending at the right. These lines define the sequence of events, and are usually not physical cables, resulting in that these two circuits in Fig. 2 are practically two different events. In this sense, the depth of the quantum circuit actually reveals its execution time. For this reason, in this paper, when we say a circuit with depth d, we mean a circuit is described by d layers of gates, and all gates in each layer can be executed simultaneously.

3 Linear depth optimization

Generally for a linear Boolean function or a linear component in symmetric-key ciphers, one may first easily obtain its matrix A from its algebraic normal form. Then implementing A with a CNOT circuit is equivalent to decomposing A into a sequence of elementary matrices and a permutation matrix. Since this permutation matrix corresponds to rewiring operations that are considered free, we only need to focus on this sequence of elementary matrices. We call it the *decomposition sequence* of A, which is defined formally as follows.

Definition 1. *A decomposition sequence* `SEQ` *of a matrix A is a finite sequence of elementary matrices with a particular order*

$$\texttt{SEQ} = \{E(c_1, t_1), E(c_2, t_2), \cdots, E(c_L, t_L)\}, \tag{3}$$

such that

$$A = P\left(\prod_{k=L}^{1} E(c_k, t_k)\right), \tag{4}$$

where P is a permutation matrix. Here, L is called the length of the sequence, and $\prod_{k=L}^{1} E(c_k, t_k)$ is called the output of the sequence.

In fact, any invertible matrix in $\mathbb{F}_2$ has a decomposition sequence followed by Theorem 1. Each $E(c_k, t_k)$ actually corresponds to a CNOT gate controlled by the line c_k acting on the line t_k in a quantum context. For convenience, we may simply use $\texttt{SEQ}$ to denote the decomposition sequence corresponding to a CNOT circuit for a given linear transformation A. For a $\texttt{SEQ}$, we say it is divided continuously into D sub-sequences if

$$\begin{aligned}
\texttt{SEQ}_1 &= \{E(c_1, t_1), E(c_2, t_2), \cdots, E(c_{k_1}, t_{k_1})\},\\
\texttt{SEQ}_2 &= \{E(c_{k_1+1}, t_{k_1+1}), E(c_{k_1+2}, t_{k_1+2}) \cdots, E(c_{k_2}, t_{k_2})\},\\
&\cdots\\
\texttt{SEQ}_D &= \{E(c_{k_{D-1}+1}, t_{k_{D-1}+1}), E(c_{k_{D-1}+2}, t_{k_{D-1}+2}) \cdots, E(c_{k_D}, t_{k_D})\},
\end{aligned} \tag{5}$$

and $\bigcup_{k=1}^{D} \texttt{SEQ}_k = \texttt{SEQ}$, and $\{\texttt{SEQ}_1, \texttt{SEQ}_2, \ldots, \texttt{SEQ}_D\}$ is called a *parallel partition* of $\texttt{SEQ}$ if $\bigcap_i \{c_i, t_i\} = \emptyset$ for any i in each $\texttt{SEQ}_k$. Then the depth of $\texttt{SEQ}$ is defined as follows.

Definition 2 (Sequence depth). *For a decomposition sequence $\texttt{SEQ}$, the depth of $\texttt{SEQ}$ is the minimum D such that there is a parallel partition of $\texttt{SEQ}$ with D sub-sequence.*

Notably, as discussed in Section 2.2, the depth of a quantum circuit is a circuit-architecture-dependent parameter, and from a parallel partition with D sub-sequences, we can easily achieve a quantum circuit with depth D. Moreover, we always cluster the CNOT gates leftward when considering the depth of a CNOT circuit particularly, making as many gates as possible run simultaneously.

It is straightforward to know that if $\{c_{i-1}, t_{i-1}\} \cap \{c_i, t_i\} = \emptyset$ (or $\{c_i, t_i\} \cap \{c_{i+1}, t_{i+1}\} = \emptyset$), $E(c_i, t_j)$ can be *moved forward (or backward)* without changing the output of the sequence. Then we have the following definition for the equivalence of two decomposition sequences.

Definition 3 (Move-equivalence). *Two decomposition sequences $\texttt{SEQ}$ and $\texttt{SEQ}'$ are move-equivalent if $\texttt{SEQ}'$ can be obtained by moving gates in $\texttt{SEQ}$ forward or backward.*

Actually, given a CNOT circuit, which corresponds to a decomposition sequence $\texttt{SEQ}$, the output of its depth from most quantum resources estimators, for example, the Q# resources estimator of Microsoft [1], is the depth of the move-equivalent sequence of $\texttt{SEQ}$ under the strategy that moves all $E(c_i, t_i)$ forward as far as possible. However, we will discuss in the next subsection that for a $\texttt{SEQ}$, after exchanging some gates that seem can not be further moved forward (or backward), we can obtain a new decomposition sequence that has lower depth than all move-equivalent sequences of $\texttt{SEQ}$.

3.1 Depth optimization for decomposition sequences

By PLU-decomposition, one can easily obtain a decomposition sequence of a matrix A. However, it is obviously not an optimal implementation when concerning two metrics - the gate count and the circuit depth.

For the gate count, it is pointed out in [47] that there are seven cases where three adjacent elementary operations can be equivalently reduced to two, implementing the same Boolean function. As a result, they built seven rules to optimize a given sequence and here we employ the same method to reduce the gate count of a decomposition sequence.

For the circuit depth, we find that the depth of a CNOT circuit can be further reduced since there are other equivalent decomposition sequences for a `SEQ` that is shallower. For example, we can swap the order of the second and the third CNOT gates of Circuit (a) in Fig. 3, maintaining the output. Then we obtain Circuit (b), which has circuit depth 2.

(a) CNOT circuit with depth 3 (b) CNOT circuit with depth 2

Fig. 3. The quantum CNOT circuits with different depths by exchanging gates.

Actually, we have the following observation for the elementary matrix (the CNOT gate) $E(c_k, j_k)$ in a decomposition sequence:

Observation 1 *For elementary matrices,* $E(c_i, t_i)E(c_j, t_j) = E(c_j, t_j)E(c_i, t_i)$ *if and only if* $t_i \neq c_j$ *and* $c_i \neq t_j$.

Observation 1 is quite obvious from the circuit perspective meaning that two CNOT gates can swap order with each other if and only if the control qubit of the first gate is not the target qubit of the other, and vice versa. In Fig. 3 the first CNOT gate and the second CNOT gate in Circuit (a) can not be exchanged since the target qubit of the first gate is the controlled qubit of the second gate. In this way, we have the following definition for *exchange-equivalence* of two decomposition sequences.

Definition 4 (Exchange-equivalence). *For two adjacent gates* $E(c_k, t_k)$ *and* $E(c_{k+1}, t_{k+1})$ *in* `SEQ`*, we can exchange the order of the two if and only if* $t_k \neq c_{k+1}$ *and* $c_k \neq t_{k+1}$*. We say* `SEQ'` *and* `SEQ` *are exchange-equivalent if* `SEQ'` *can be obtained by exchanging the order of the gates in* `SEQ`*.*

It is worth noting that the move-equivalence is a special case of the exchange-equivalence. Obviously, now we can try to find a shallower circuit for SEQ among all exchange-equivalent decomposition sequences. To this end, we present Algorithm 1 to find the exchange-equivalent SEQ′ that has nearly optimal circuit depth for a given decomposition sequence. Intuitively, we try to apply as many quantum gates as possible in a single sub-sequence. To accomplish this, we search for possible swapping between different gates forward and backward as detailed in the function One-way-opt. Algorithm 1 uses One-way-opt twice and has the following property.

Property 1. Given a decomposition sequence SEQ with L gates, the Algorithm 1 has $O(L^2)$ steps to achieve a stable depth, meaning the depth will not be further reduced by using One-way-opt more.

Proof. Suppose the output of the Algorithm 1 is S_1 which is exchange-equivalent to SEQ. If the number of sub-sequences can still be reduced, meaning there is a redundant sub-sequence S_r all of whose gates can be moved equivalently into other sub-sequences in S_1. For any gate E_r in S_r, if the sub-sequence in which E_r can be moved lies before S_r, E_r should have been moved there in Step 1 of the Algorithm 1 by the definition of the One-way-opt procedure. Else if the sub-sequence in which E_r can be moved lies after S_r, E_r should have been moved there in Step 3 of the Algorithm 1. All of these claim that there is no such E_r that can be moved equivalently into other sub-sequences. Thus by applying One-way-opt twice, we achieve a stable depth for implementing SEQ with Algorithm 1. For each gate in SEQ, in the worst case, we may iterate through all the gates that lie after A in SEQ twice to check whether they can be executed in parallel and whether they are exchangeable. This will give us a query complexity $O(L^2)$.

Remark 1. For a linear Boolean function $\mathcal{F}$ with n variables, the result in [37] shows that it can be implemented with $O(n^2/\log(n))$ CNOT gates. Hence, by Algorithm 1, we can obtain a low-depth CNOT circuit of $\mathcal{F}$ with complexity $O(n^4/\log(n)^2)$.

Algorithm 1: CNOT depth optimization of SEQ

Input: A decomposition sequence SEQ.
Output: A low-depth decomposition sequence $\mathtt{SEQ}_{\mathrm{opt}}$ which is exchange-equivalent to SEQ.

1 $\mathtt{SEQ}_{\mathrm{left}} \leftarrow$One-way-opt(SEQ);
2 Reverse sort the gates in $\mathtt{SEQ}_{\mathrm{left}}$ to get $\mathtt{SEQ}_{\mathrm{left}}^{\mathrm{rev}}$;
3 $\mathtt{SEQ}_{\mathrm{opt}}^{rev} \leftarrow$ One-way-opt($\mathtt{SEQ}_{\mathrm{left}}^{\mathrm{rev}}$);
4 Reverse sort the gates in $\mathtt{SEQ}_{\mathrm{opt}}^{rev}$ to get $\mathtt{SEQ}_{\mathrm{opt}}$;
5 **return** $\mathtt{SEQ}_{\mathrm{opt}}$;

Example 1. Here we show a toy example to demonstrate the effectiveness of our algorithm. For a decomposition sequence

$$\mathtt{SEQ} = \{E(0,1), E(1,3), E(2,3), E(2,0), E(0,3), E(0,2), E(2,1)\}, \tag{6}$$

One-way-opt: One-way-opt CNOT depth optimization of SEQ(A)

Input: A decomposition sequence $\texttt{SEQ}(A) = \{E(c_1, t_1), E(c_2, t_2), \cdots, E(c_K, t_K)\}$ of an invertible matrix A.

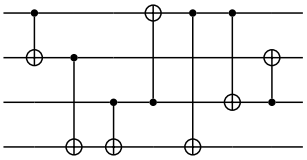
Output: A decomposition sequence $\texttt{SEQ}_{\text{out}}(A)$ of matrix A and the sequence depth d.

```
SEQ_out(A) ← {·};
d = 0;
while length(SEQ(A)) > 1 do
    Layer ← {·};
    d = d + 1;
    Move the first gate in SEQ to Layer;
    i = 1;
    while i <= length(K) do
        if E(c_i, t_i) can be executed simultaneously with all gates in Layer then
            CHANGE ← TRUE;
            for j = 1 : i do
                if E(c_j, t_j) can not swap with E(c_i, t_i) then
                    CHANGE ← FALSE;
                end
            end
            if CHANGE = TRUE then
                Layer ← E(c_i, t_i);
                Remove E(c_i, t_i) from SEQ
            end
            i = i + 1;
        else
            i = i + 1;
        end
    end
    Add all gates in Layer to SEQ_out(A);
end
Add all gates left in SEQ to SEQ_out(A);
return SEQ_out(A), d;
```

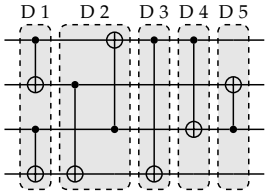
its circuit is shown as (a) in Fig. 4 whose sequence depth is obviously 7. After the first step of Algorithm 1, $E(2,3)$ is exchanged with $E(1,3)$ since it can be executed with $E(0,1)$ in parallel and is exchangeable with $E(1,3)$. Then we get an exchanging-equivalent sequence whose circuit is shown as (b) in Fig. 4 with depth 5. Furthermore, after Step 3 and Step 4, the order of $E(0,3)$ and $E(0,2)$ are swapped since they are exchangeable and $E(0,3)$ can be executed in parallel with $E(2,1)$, reducing the circuit depth from 5 to 4. Then Algorithm 1 output the sequence

$$\texttt{SEQ}_{\text{opt}} = \{E(0,1), E(2,3), E(2,0), E(1,3), E(0,2), E(0,3), E(2,1)\}, \quad (7)$$

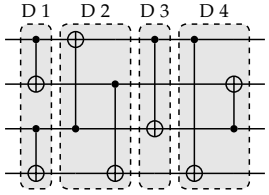
whose circuit is shown as (c) in Fig. 4.



(a) CNOT circuit of SEQ



(b) CNOT circuit of $\texttt{SEQ}_{\text{left}}$



(c) CNOT circuit of $\texttt{SEQ}_{\text{opt}}$

Fig. 4. The quantum CNOT circuit of SEQ and its exchange-equivalent circuits obtained in Algorithm 1.

3.2 Finding better gate sequences

Besides the optimized implementation of a given decomposition sequence as well as all its equivalent sequences, we point out that there are actually other different decomposition sequences that can implement the linear transformation A. As an example shown in Fig. 5, circuit (a) and circuit (b) realize the same Boolean function after we rename (or swap) wire i and wire k, hence $\texttt{SEQ}_a$ for Circuit (a) and $\texttt{SEQ}_b$ for Circuit (b) are two different decomposition sequences for the same matrix. Whereas, circuit (a) has depth 4, and circuit (b) has depth 3 since the second and the third CNOT gates in (b) can be executed simultaneously. Hence, we are inspired to find a shallower implementation, by firstly constructing different decomposition sequences, then applying Algorithm 1 to find a better

decomposition sequence whose minimum depth after our optimization method in Section 3.1 is lower. We present our framework for searching a depth-optimized linear quantum circuit for a linear building block in Algorithm 3.

(a) CNOT circuit with depth 4 (b) CNOT circuit with depth 3

Fig. 5. The quantum CNOT circuits with different depths after swapping wires.

Algorithm 3: Search a low-depth implementation of SEQ(A)

Input: An invertible matrix $A \in \mathrm{GL}(n, \mathbb{F}_2)$.
Output: A low-depth decomposition sequence $\mathtt{SEQ}_{\mathrm{opt}}(A)$ of matrix A.

Translate A into a binary matrix and decompose it into $\{E_t, E_{t-1} \cdots, E_1\}$;
$\mathtt{SEQ} \leftarrow \{E_t, E_{t-1} \cdots, E_1\}$;
$\mathtt{SEQ}_{\mathrm{opt}}$ is the output of Algorithm 1 (SEQ), and $d(\mathtt{SEQ}_{\mathrm{opt}})$ is the depth of $\mathtt{SEQ}_{\mathrm{opt}}$; // Initialize the depth
$g \leftarrow t + 1$;
while $g \geq 2$ **do**
  $g = g - 1$;
  for $i = 0$ *to* $t - g$ **do**
    $\mathtt{SEQ}_1 \leftarrow \{E_t, \cdots E_{i+g-1}\}$;
    $\mathtt{SEQ}_2 \leftarrow \{E_{i+g}, \cdots E_{i+1}\}$;
    $\mathtt{SEQ}_3 \leftarrow \{E_i, \cdots E_1\}$;
    $A' \leftarrow \{E_{i+g}, \cdots E_{i+1}\}$;
    Decompose A' into $\{E'_u, E'_{u-1} \cdots, E_1\}$;
    $\mathtt{SEQ}'_2 \leftarrow \{E'_u, \cdots, E_1\}$;
    $\mathtt{SEQ}' = \mathtt{SEQ}_1 + \mathtt{SEQ}'_2 + \mathtt{SEQ}_3$; // New decomposition sequence
    $\mathtt{SEQ}'_{opt}$ is the output of Algorithm 1 ($\mathtt{SEQ}'$), and $d(\mathtt{SEQ}')$ is the depth of $\mathtt{SEQ}'_{opt}$; // Minimize depth for SEQ′
    if $d(\mathit{SEQ}') < d(\mathit{SEQ}_{\mathrm{opt}})$ *and* $|\mathit{SEQ}'| < |\mathit{SEQ}_{\mathrm{opt}}|$ **then**
      $\mathtt{SEQ}_{\mathrm{opt}} = \mathtt{SEQ}'$;
      $d(\mathtt{SEQ}_{\mathrm{opt}}) = d(\mathtt{SEQ}')$;
      Break;
    end
  end
end
return $\mathtt{SEQ}_{\mathrm{opt}}$;

4 Applications

In this section, we showcase the applications of our algorithm in different linear building blocks. From these experimental results, one can see that our algorithm can not only be applied to optimize invertible linear transformations, but also be extended to optimize non-invertible linear transformations. Consequently, it is helpful for the optimization of the whole circuit depth for some block ciphers such as AES by optimizing some linear sub-structures of the whole circuit.

Optimization for invertible linear transformations. Firstly, we apply our framework to minimize the quantum circuit depth of a large set of invertible cipher matrices. Notice that [47] gave state-of-the-art classical implementations of some cipher matrices to our knowledge in terms of the gate count, which outperform Paar's and Boyar-Peralta's heuristics [32] in most cases. And their implementation can be typically employed to produce a compact CNOT circuit implementing the linear transformation. We therefore compare our result with theirs in Table 1, concerning the circuit depth while maintaining the gate count same.

The depths of the quantum circuits for different cipher matrices optimized by our method are listed in the last column. We can see there is a significant improvement in our synthesis compared with the circuit implementation in [47]. When the matrix size is large such as an 8×8 matrix for KHAZAD [9] (the size is 8×8 in GF(8, $\mathbb{F}_2$) and 64×64 in GF(2, $\mathbb{F}_2$)), our method can give a nearly 75% improvement compared with the naive implementation with sequence depth, and a 45% improvement compared with the usual move-equivalent optimization. Even for 4×4 small matrices, our method still can reduce the depth of the circuits in some cases.

Next, we apply our framework to optimize the depth of the quantum circuit implementation for various invertible matrices, presented in different works [11,27,35,44,41,40]. All matrices compared can be found in those papers sorted by their size and the finite field they belong to. The experiment results are summarized in Table 2. Notice that for different matrices with different sizes, our method always reduces the circuit depth compared with the previous quantum circuit construction derived from the in-place implementations of classical linear circuits. Generally, the larger the size of the matrix, the greater the advantage of our method.

Optimization for non-invertible linear transformations. Besides, our framework can also be used to optimize the circuit depth of some non-invertible linear transformations, since any non-invertible linear transformation can always be expressed by a sequence as well. For example, we optimized the quantum circuit depth for AES, by reducing the depth of some linear sub-structures in its nonlinear blocks. As shown in [16], when using the tower filed structure, there are two linear components in the implementation of AES S-box, called the *top linear layer* and *bottom linear layer*, respectively. The top linear layer, which is defined as a 22×8 binary matrix, is used to generate the desired number of middle variables used for a tower field construction, while the bottom linear

layer, which is defined as an 8×18 binary matrix, is used to generate the output of the S-box from the output of the tower field construction. After applying our framework, we can reduce the depth of the top linear layer of the AES S-box circuit from 14 [39,24] to 8, compared with the previous implementations. For the bottom linear layer, we can reduce the depth from 11 to 7. Thus we can reduce the depth of linear layers of an AES S-box by 10. Even though this reduction is not very large, our new implementation will reduce the depth of the whole AES circuit significantly, since the S-box and its inverse are used iteratively in the whole circuit.

Table 1. Quantum circuit depth of cipher matrices under different optimization heuristic

Cipher	Size[1]	# CNOT[2]	Seq D[3]	Move-eq D[4]	Exchange-eq D[5]
KHAZAD [9]	64	366	112	54	30
AES [20]	32	92	41	30	28
ANUBIS [45]	32	98	40	26	20
CLEFIA M0 [43]	32	98	41	30	27
CLEFIA M1 [43]	32	103	41	21	16
FOX MU4 [29]	32	136	75	55	48
QARMA128 [6]	32	48	12	6	5
TWOFISH [31]	32	111	53	37	29
WHIRLWIND M0 [8]	32	183	93	65	51
WHIRLWIND M1 [8]	32	190	90	69	54
JOLTIK [26]	16	44	23	20	17
MIDORI [7]	16	24	9	3	3
SmallScale AES [19]	16	43	26	20	19
PRIDE L0 [3]	16	24	9	3	3
PRIDE L1 [3]	16	24	15	5	5
PRIDE L2 [3]	16	24	12	5	5
PRIDE L3 [3]	16	24	11	5	5
PRINCE M0 [15]	16	24	10	6	6
PRINCE M1 [15]	16	24	10	6	6
QARMA64 [6]	16	24	9	6	5
SKINNY [10]	16	12	3	3	3

[1] The size refers to the degree of the corresponding binary matrix.
[2] The number of CNOT gates of quantum implementation in [47].
[3] Quantum circuit depth with sequence depth.
[4] Quantum circuit depth with minimum move-equivalent depth.
[5] Quantum circuit depth with minimum exchange-equivalent depth.

Table 2. Quantum circuit depth of some invertible matrices with different optimization methods.

Matrices	Size[1]	#CNOT[2]	Seq D[3]	Move-eq D[4]	Exchange-eq D[5]
4×4 matrices in GF(4, $\mathbb{F}_2$)					
[11]	16	41	27	23	21
[27]	16	41	28	24	18
[35]	16	44	29	27	26
[44]	16	44	30	25	22
[34]	16	44	29	29	27
[27](Involutory)	16	41	25	15	14
[44](Involutory)	16	44	24	19	16
[34](Involutory)	16	44	33	27	25
[40](Involutory)	16	38	19	12	11
4×4 matrices in GF(8, $\mathbb{F}_2$)					
[11]	32	114	72	56	47
[27]	32	82	43	26	22
[35]	32	121	79	67	54
[34]	32	104	69	55	42
[44]	32	90	42	23	20
[40]	32	114	58	47	40
[27](Involutory)	32	83	34	18	14
[44](Involutory)	32	91	39	18	16
[34](Involutory)	32	87	39	19	19
[40](Involutory)	32	93	42	19	18
8×8 matrices in GF(4, $\mathbb{F}_2$)					
[41]	32	183	83	54	44
[44]	32	170	89	59	49
[44](Involutory)	32	185	85	47	37
8×8 matrices in GF(8, $\mathbb{F}_2$)					
[44](Involutory)	64	348	117	50	37

[1] The size refers to the degree of the corresponding binary matrix.
[2] The number of CNOT gates of quantum implementation in [47].
[3] Quantum circuit depth with sequence depth.
[4] Quantum circuit depth with minimum move-equivalent depth.
[5] Quantum circuit depth with minimum exchange-equivalent depth.

5 Conclusion

In this work, we focus on minimizing the depth of a subclass of quantum circuits - the CNOT circuits, especially those for the linear building blocks of symmetric-key ciphers. We are motivated by the quantum security analysis of current symmetric-key encryption systems and end up with a framework for constructing low-depth quantum circuits for linear Boolean functions. We fully characterize the CNOT circuits with decomposition sequence and its two equivalent classes, called move-equivalent sequence and exchange-equivalent sequence. Based on these two classes, we can give a clearer definition for the depth of

the CNOT circuits and achieve shallower quantum circuit implementations for a large set of cipher matrices compared with previous results.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Grant No. 61977060).

References

1. Microsoftt q#. quantum development. `https://devblogs.microsoft.com/qsharp/`
2. Aaronson, S., Gottesman, D.: Improved simulation of stabilizer circuits. Physical Review A **70**(5) (nov 2004)
3. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yaln, T.: Block ciphers – focus on the linear layer (feat. pride). Annual Cryptology Conference (2014)
4. Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of aes-128. Quantum Information Processing **17**(5), 1–30 (2018)
5. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **32**(6), 818–830 (2013)
6. Avanzi, R.: The qarma block cipher family. almost mds matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Transactions on Symmetric Cryptology **2017**(1), 4–44 (2017)
7. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Regazzoni, F.: Midori: A block cipher for low energy. Springer Berlin Heidelberg (2015)
8. Barreto, P., Nikov, V., Nikova, S., Rijmen, V., Tischhauser, E.: Whirlwind: A new cryptographic hash function. Designs Codes and Cryptography **56**(2-3), 141–162 (2010)
9. Barreto, P., Rijmen, V.: The khazad legacy-level block cipher. submission to the nessie project (2000)
10. Beierle, C., Jean, J., Kölbl, S., Leander, G., Sim, S.M.: The skinny family of block ciphers and its low-latency variant mantis. In: Annual Cryptology Conference (2016)
11. Beierle, C., Kranz, T., Leander, G.: Lightweight multiplication in $gf(2^n)$ with applications to mds matrices. In: Annual International Cryptology Conference. pp. 625–653. Springer (2016)
12. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. Cryptology ePrint Archive, Paper 2021/1239 (2021)
13. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of aes. IACR Transactions on Symmetric Cryptology **2019**(2), 55–93 (2019)
14. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: On quantum slide attacks. Cryptology ePrint Archive, Paper 2018/1067 (2018)
15. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C.: Prince - a low-latency block cipher for pervasive computing applications: Extended abstract. In: Springer Berlin Heidelberg (2012)
16. Boyar, J., Peralta, R.: A small depth-16 circuit for the aes s-box. In: IFIP International Information Security Conference. pp. 287–298. Springer (2012)

17. de Brugiere, T.G., Baboulin, M., Valiron, B., Martiel, S., Allouche, C.: Reducing the depth of linear reversible quantum circuits. IEEE Transactions on Quantum Engineering **2**, 1–22 (2021)
18. Brugière, T.G.D., Baboulin, M., Valiron, B., Martiel, S., Allouche, C.: Gaussian elimination versus greedy methods for the synthesis of linear reversible circuits. ACM Transactions on Quantum Computing **2**(3), 1–26 (sep 2021)
19. Cid, C., Murphy, S., Robshaw, M.: Small scale variants of the aes. In: International Conference on Fast Software Encryption (2005)
20. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. The Design of Rijndael: AES - The Advanced Encryption Standard (2002)
21. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to aes: quantum resource estimates. In: Springer International Publishing (2015)
22. Grover, L.K.: A fast quantum mechanical algorithm for database search (1996)
23. Hosoyamada, A., Sasaki, Y.: Quantum demiric-selçuk meet-in-the-middle attacks: applications to 6-round generic feistel constructions. In: International Conference on Security and Cryptography for Networks. pp. 386–403. Springer (2018)
24. Huang, Z., Sun, S.: Synthesizing quantum circuits of aes with lower t-depth and less qubits. Cryptology ePrint Archive, Paper 2022/620 (2022)
25. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on aes and lowmc. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 280–310. Springer (2020)
26. Jean, J., Nikolić, I., Peyrin, T.: Joltik. submission to the caesar competition (2014)
27. Jean, J., Peyrin, T., Sim, S.M., Tourteaux, J.: Optimizing implementations of lightweight building blocks. Cryptology ePrint Archive (2017)
28. Jiang, J., Sun, X., Teng, S.H., Wu, B., Wu, K., Zhang, J.: Optimal space-depth trade-off of cnot circuits in quantum logic synthesis. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 213–229. SIAM (2020)
29. Junod, P., Vaudenay, S.: Fox : A new family of block ciphers. In: Handschuh, H., Hasan, M.A. (eds.) Selected Areas in Cryptography. pp. 114–129. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
30. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. arXiv preprint arXiv:1510.05836 (2015)
31. Kelsey, B., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128bit block cipher (1998)
32. Kranz, T., Leander, G., Stoffelen, K., Wiemer, F.: Shorter linear straight-line programs for mds matrices (2017)
33. Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing the advanced encryption standard as a quantum circuit. IEEE Transactions on Quantum Engineering **1**, 1–12 (2020)
34. Li, Y., Wang, M.: On the construction of lightweight circulant involutory mds matrices. Springer, Berlin, Heidelberg (2016)
35. Liu, M., Sim, S.M.: Lightweight mds generalized circulant matrices. In: International Conference on Fast Software Encryption. pp. 101–120. Springer (2016)
36. Miller, D., Maslov, D., Dueck, G.: A transformation based algorithm for reversible logic synthesis (July 2003), 318–323 (2004)
37. Patel, K.N., Markov, I.L., Hayes, J.P.: Optimal synthesis of linear reversible circuits. Rinton Press, Incorporated (3) (2008)

38. Saeedi, M., Markov, I.L.: Synthesis and optimization of reversible circuits-a survey. ACM Computing Surveys **45**(2) (2013)
39. SamuelJaques, MichaelNaehrig, MartinRoetteler, FernandoVirdia: Implementing grover oracles for quantum key search on aes and lowmc. Springer, Cham (2020)
40. Sarkar, S., Syed, H.: Lightweight diffusion layer: Importance of toeplitz matrices (2016)
41. Sarkar, S., Syed, H.: Analysis of toeplitz mds matrices. In: Australasian Conference on Information Security and Privacy. pp. 3–18. Springer (2017)
42. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Reversible logic circuit synthesis. IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers pp. 353–360 (2002)
43. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher clefia (extended abstract). In: International Workshop on Fast Software Encryption (2007)
44. Sim, S.M., Khoo, K., Oggier, F., Peyrin, T.: Lightweight mds involution matrices. In: International Workshop on Fast Software Encryption. pp. 471–493. Springer (2015)
45. S.L.M., P., Barreto, Rijmen, V.: The anubis block cipher (2000)
46. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: Revlib: An online resource for reversible functions and reversible circuits. In: 38th International Symposium on Multiple Valued Logic (ismvl 2008). pp. 220–225. IEEE (2008)
47. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. IACR Transactions on Symmetric Cryptology pp. 120–145 (2020)
48. Zakablukov, D.V.: Application of Permutation Group Theory in Reversible Logic Synthesis pp. 1–15 (2015)
49. Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of aes with fewer qubits. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 697–726. Springer (2020)

Appendix

In the following, we present the CNOT circuit for AES MixColumns using 92 CNOT gates, which keeps the gate count the same as the implementation with classical XOR gates in [47]. After our optimization, the circuit depth is reduced from 41 to 28, compared with direct sequence depth; from 30 to 28, compared with move-equivalent circuit depth.

Table 3. A quantum circuit for AES MixColumns with depth 28, where each XOR operation is corresponding to a CNOT gate.

No.	Operation	No.	Operation	No.	Operation	No.	Operation
Layer 1		Layer 11		44	$x_8 = x_0 \oplus x_8$	68	$x_6 = x_5 \oplus x_6$
0	$x_{14} = x_6 \oplus x_{14}$	21	$x_3 = x_{11} \oplus x_3$	45	$x_{28} = x_{12} \oplus x_{28}$	69	$x_1 = x_{25} \oplus x_1$
Layer 2		22	$x_{20} = x_{19} \oplus x_{20}$	46	$x_4 = x_{31} \oplus x_4$	Layer 26	
1	$x_6 = x_{22} \oplus x_6$	23	$x_{26} = x_{10} \oplus x_{26}$	47	$x_{15} = x_7 \oplus x_{15}$	70	$x_{28} = x_{20} \oplus x_{28}$
Layer 3		Layer 12		Layer 19		71	$x_{14} = x_{30} \oplus x_{14}$
2	$x_{22} = x_{30} \oplus x_{22}$	24	$x_{11} = x_{10} \oplus x_{11}$	48	$x_{16} = x_0 \oplus x_{16}$	72	$x_{15} = x_7 \oplus x_{15}$
3	$x_{13} = x_{21} \oplus x_{13}$	25	$x_{19} = x_{18} \oplus x_{19}$	49	$x_{12} = x_{15} \oplus x_{12}$	73	$x_5 = x_{29} \oplus x_5$
Layer 4		Layer 13		50	$x_{27} = x_{31} \oplus x_{27}$	74	$x_0 = x_{24} \oplus x_0$
4	$x_{30} = x_{13} \oplus x_{30}$	26	$x_{10} = x_{18} \oplus x_{10}$	Layer 20		75	$x_{25} = x_9 \oplus x_{25}$
Layer 5		27	$x_{17} = x_9 \oplus x_{17}$	51	$x_0 = x_{31} \oplus x_0$	76	$x_3 = x_{19} \oplus x_3$
5	$x_{13} = x_{29} \oplus x_{13}$	Layer 14		52	$x_{25} = x_{24} \oplus x_{25}$	Layer 27	
6	$x_{21} = x_5 \oplus x_{21}$	28	$x_{18} = x_2 \oplus x_{18}$	53	$x_{11} = x_{15} \oplus x_{11}$	77	$x_{20} = x_4 \oplus x_{20}$
7	$x_{12} = x_4 \oplus x_{12}$	29	$x_9 = x_1 \oplus x_9$	Layer 21		78	$x_{29} = x_{21} \oplus x_{29}$
Layer 6		30	$x_0 = x_{24} \oplus x_0$	54	$x_{31} = x_7 \oplus x_{31}$	79	$x_6 = x_{14} \oplus x_6$
8	$x_5 = x_{13} \oplus x_5$	Layer 15		55	$x_{24} = x_{15} \oplus x_{24}$	80	$x_7 = x_{23} \oplus x_7$
9	$x_4 = x_{28} \oplus x_4$	31	$x_{18} = x_{17} \oplus x_{18}$	Layer 22		81	$x_1 = x_0 \oplus x_1$
Layer 7		32	$x_{10} = x_9 \oplus x_{10}$	56	$x_7 = x_{14} \oplus x_7$	82	$x_9 = x_{17} \oplus x_9$
10	$x_{13} = x_{12} \oplus x_{13}$	33	$x_{11} = x_2 \oplus x_{11}$	57	$x_{15} = x_{23} \oplus x_{15}$	83	$x_2 = x_{26} \oplus x_2$
11	$x_{29} = x_4 \oplus x_{29}$	34	$x_{24} = x_8 \oplus x_{24}$	58	$x_{12} = x_{27} \oplus x_{12}$	84	$x_{19} = x_{27} \oplus x_{19}$
12	$x_{11} = x_{27} \oplus x_{11}$	Layer 16		Layer 23		Layer 28	
Layer 8		35	$x_{17} = x_{25} \oplus x_{17}$	59	$x_{14} = x_{21} \oplus x_{14}$	85	$x_4 = x_{12} \oplus x_4$
13	$x_{12} = x_{20} \oplus x_{12}$	36	$x_2 = x_9 \oplus x_2$	60	$x_{31} = x_{22} \oplus x_{31}$	86	$x_{21} = x_{13} \oplus x_{21}$
14	$x_4 = x_{11} \oplus x_4$	37	$x_8 = x_{23} \oplus x_8$	61	$x_{16} = x_{23} \oplus x_{16}$	87	$x_{22} = x_6 \oplus x_{22}$
Layer 9		38	$x_{24} = x_{16} \oplus x_{24}$	62	$x_{27} = x_{26} \oplus x_{27}$	88	$x_{23} = x_{31} \oplus x_{23}$
15	$x_{20} = x_{27} \oplus x_{20}$	39	$x_{31} = x_{15} \oplus x_{31}$	63	$x_{30} = x_6 \oplus x_{30}$	89	$x_{17} = x_0 \oplus x_{17}$
16	$x_{11} = x_{19} \oplus x_{11}$	Layer 17		Layer 24		90	$x_{26} = x_{18} \oplus x_{26}$
17	$x_{23} = x_{31} \oplus x_{23}$	40	$x_1 = x_{17} \oplus x_1$	64	$x_{22} = x_{21} \oplus x_{22}$	91	$x_{27} = x_{11} \oplus x_{27}$
Layer 10		41	$x_9 = x_8 \oplus x_9$	65	$x_{23} = x_6 \oplus x_{23}$		
18	$x_{27} = x_3 \oplus x_{27}$	42	$x_{16} = x_{31} \oplus x_{16}$	66	$x_{26} = x_1 \oplus x_{26}$		
19	$x_{19} = x_{23} \oplus x_{19}$	Layer 18		Layer 25			
20	$x_{18} = x_{26} \oplus x_{18}$	43	$x_{17} = x_{16} \oplus x_{17}$	67	$x_{21} = x_{28} \oplus x_{21}$		