Novel SMT Encoding for Quantum Circuit Optimization

Youbo Guo[1], Fengrong Zhang[1], Lei Liao[1], Yongzhuang Wei[2], Baocang Wang[1] and Xiaogang Zhou[3]

[1] Xidian University
[2] Guilin University of Electronic Technology
[3] China Telecom Quantum Information Technology Group Co., Ltd

Abstract

In recent years, quantum circuit optimization has become an important research topic. Motivated by the fact that quantum gates act on fixed physical wires and modify only their target wires, we propose two SMT encodings: an exact-G encoding and an at-most-G encoding with null gates. Our method speeds up most tested 4-bit S-box instances, achieving up to approximately 130× speedup on the ELEPHANT S-box. Importantly, our method enables automated synthesis of practical 5-bit S-box quantum circuits, such as KECCAK and ASCON. For the KECCAK S-box, in the no-ancilla setting, our model obtains concrete implementations with 17 NCT gates and full depth 51, and with 16 NCT gates and full depth 52, improving the EUROCRYPT 2025 result of Huang et al. It further finds a 13-gate implementation with full depth 55, which is gate-count optimal in the no-ancilla setting under the NCT gate set. In addition, when one ancilla qubit is allowed, our model obtains KECCAK implementations with Toffoli count 5, matching the theoretical lower bound. Finally, our model can also be applied to small-scale linear-layer implementation; for example, it finds a 24-CNOT implementation with depth 3 for the 16×16 linear matrix of MIDORI.

Keywords: SAT/SMT, Quantum Circuit, S-box, Linear layer, NCT

1 Introduction

The rapid development of quantum information technology has profoundly affected computer and communication security. In particular, Shor's algorithm [Sho97] threatens the foundations of traditional public-key cryptography by solving integer factorization and discrete logarithm problems in polynomial time. Meanwhile, Grover's algorithm [Gro96] provides a quadratic speedup for exhaustive key search, thereby reducing the effective security level of symmetric primitives in the quantum setting. As a result, evaluating the cost of implementing symmetric ciphers on quantum computers, especially in terms of reversible and quantum circuit resources, has become an important research problem in cryptography and quantum information.

In many block ciphers, the S-box is the primary nonlinear component and plays a central role in providing confusion. Its cryptographic properties directly influence the resistance of the cipher against major attacks, including differential cryptanalysis [BS90] and linear

Emails: guoyou182@163.com (Youbo Guo), zhangfengrong@xidian.edu.cn (Fengrong Zhang), liao_lei@stu.xidian.edu.cn (Lei Liao), walker_wyz@guet.edu.cn (Yongzhuang Wei), bcwang79@aliyun.com (Baocang Wang), zhouxiaogang@chinatelecom.cn (Xiaogang Zhou)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-08-27.

cryptanalysis [Mat93]. From an implementation perspective, S-boxes often dominate the cost of nonlinear circuit realization, particularly in lightweight and resource-constrained designs. Consequently, synthesizing S-box circuits with small gate count, low depth, and limited ancillary resources is important for efficient implementations. In the quantum setting, this problem is further amplified: S-boxes must be realized as reversible circuits, and resources such as qubits, Toffoli gates, and circuit depth directly affect the cost of quantum implementations of symmetric ciphers.

Considerable effort has been devoted to optimizing S-box circuit implementations in the classical setting. Early works based on heuristic logic minimization and meet-in-the-middle search, such as the method of Boyar et al. [BMP13], LIGHTER [JPST17], and PEIGEN [BGLS19], have produced compact implementations for many lightweight S-boxes. SAT-based approaches further provide a systematic way to search for implementations that are optimal with respect to specific criteria. Stoffelen [Sto16] formulated the optimization of small S-box implementations as a SAT problem by encoding algebraic normal form relations into conjunctive normal form, and subsequent works extended this line of research to richer gate libraries, area-oriented metrics, and cryptographic design constraints [FWL+21, JCL+25, WSL+25]. However, these methods mainly target classical irreversible Boolean circuits. Their results cannot be directly translated into efficient quantum implementations, because quantum circuits must satisfy reversibility constraints.

In the quantum setting, the NCT gate set, consisting of Pauli-X, CNOT, and Toffoli gates, is commonly used as a basic reversible gate library. These gates may be viewed as reversible analogues of NOT, XOR, and AND operations, respectively. Unlike classical irreversible circuits, quantum implementations of the bijective S-boxes considered in this work must realize reversible transformations on all involved qubits. Their cost is therefore measured not only by the number of gates, but also by the number of qubits, the number of Toffoli gates, and the circuit depth. Following the full-depth metric used by Chen et al. [CLF+24], we evaluate the depth at the NCT level by assigning cost 1 to each Pauli-X and CNOT gate and cost 7 to each Toffoli gate. This weighted depth model reflects the fact that Toffoli gates are substantially more expensive than NOT and CNOT gates, while allowing the synthesis problem to be formulated directly over the NCT gate set.

Several heuristic tools have been proposed for reversible and quantum S-box implementations, including LIGHTER-R [DBSC19] and DORCIS [CBC23]. Although these approaches are effective, they do not fully address the exact decision problem under multiple simultaneous resource constraints. More recently, SAT-based models have been introduced for quantum circuit synthesis. Chen et al. [CLF+24] proposed a SAT-based decision model for low-cost quantum circuit implementations, which searches for NCT circuits under constraints on the number of gates, circuit depth, and ancillary qubits. This model can prove optimality for arbitrary 3-bit and 4-bit S-boxes under given optimization goals. Lin et al. [LYX+26] investigated NCT-based S-box implementations under different criteria, obtaining optimal implementations for several 4-bit S-boxes and providing resource bounds for 5-bit S-boxes. Another line of work focuses on optimizing specific quantum resources such as T-depth and width. Huang et al. [HZL25] studied the construction of quantum implementations with minimal T-depth or minimal width. For the 5-bit KECCAK [BDPA13] S-box, they obtained a minimal-width 5-qubit NCT implementation with 12 NOT gates and 7 Toffoli gates. Under the same weighted NCT full-depth model, this circuit has 19 NCT gates in total and weighted full depth 52.

Despite these advances, scalable exact synthesis of quantum S-box circuits remains challenging. Existing exact or SAT-based methods are highly effective for 3-bit and 4-bit S-boxes, while results for 5-bit S-boxes are still largely restricted to specific components or selected resource bounds. For general 5-bit S-boxes and larger permutations, the search space grows rapidly when gate count, full depth, and ancillary qubits are considered simultaneously. This motivates the development of more compact encodings that can make

exact quantum S-box synthesis feasible beyond the 4-bit setting.

1.1 Our Contributions

This paper makes the following two core contributions to SMT-based quantum circuit optimization.

1. SMT Encoding Scheme. We first conduct an analysis of the existing SAT-based quantum circuit decision model by Chen et al. [CLF+24]. We observe that its wiring-based encoding introduces a large number of auxiliary variables and constraints, which limits its scalability to 5-bit and larger S-boxes. To address this issue, we propose a novel SMT encoding scheme that treats quantum wires as fixed entities and lets the solver decide the gate type and the target and control wire indices at each gate layer. In the exact-G encoding, each gate layer is described by a compact set of variables $(op_k, tgt_k, c1_k, c2_k)$. This design avoids explicitly encoding the wiring relation between gates and wires, thereby reducing the number of decision variables and constraints.

Furthermore, we extend the exact-G model to an at-most-G model by introducing a null gate. This makes the decision problem monotone with respect to G. To further improve solving efficiency, we introduce redundancy-eliminating and symmetry-breaking constraints, together with prefix depth-checkpoint constraints. We also derive a Toffoli-count lower bound for NCT circuits from the homogeneous quadratic parts of the output functions. This bound helps prove the optimality of the Toffoli count in our implementations and can be used as an additional constraint.

2. Experiments and New Results. We perform comprehensive experiments to validate the effectiveness of our model. On most tested 4-bit S-boxes, our approach achieves significant speedup over Chen et al.'s method [CLF+24], up to 130× faster on the ELEPHANT S-box and 53× faster on the SKINNY S-box.

More importantly, we apply the proposed model to the automated synthesis of 5-bit S-box quantum circuits under the NCT gate set. For the KECCAK 5-bit S-box, we improve the known NCT gate-count lower bound from $G_{\min} > 9$ [LYX+26] to $G_{\min} > 12$, and obtain new implementations with better gate-depth tradeoffs. In particular, compared with Huang et al.'s [HZL25] 19-gate implementation with full depth 52, our model obtains two improved implementations: a 17-gate circuit with full depth 51, which reduces both the gate count and the full depth, and a 16-gate circuit with full depth 52, which preserves the same full depth while further reducing the gate count. Moreover, with one ancilla qubit, our implementation achieves the minimum Toffoli count of 5 for the KECCAK 5-bit S-box. For the ASCON S-box, our model also finds new NCT implementations under both width-5 and width-6 settings. Additionally, we derive the gate lower bound for 6-bit S-boxes and prove that the APN S-box proposed by Dillon and Browning [BDMW10] cannot be implemented with only 12 NCT gates. Finally, our model can also be applied to small-scale linear-layer synthesis for SKINNY and MIDORI. For the 16×16 linear matrix of SKINNY, it proves that no 11-CNOT implementation exists and finds a 12-CNOT implementation with depth 3.

1.2 Organization

Section 2 introduces the notation used in this paper, the quantum circuit decision problem, and existing methods for quantum circuit optimization. Section 3 presents the new SMT-based encoding. Section 4 evaluates the proposed model through several applications. Finally, Section 5 concludes the paper and discusses future research directions.

2 Preliminaries

2.1 Notations

The binary vector space $\mathbb{F}_2^n$ denotes the set of all n-tuples $x = (x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{F}_2$. We denote $(0, 0, \ldots, 0) \in \mathbb{F}_2^n$ by $\mathbf{0}_n$ and denote $(1, 1, \ldots, 1) \in \mathbb{F}_2^n$ by $\mathbf{1}_n$.

2.2 Quantum Circuit

To optimize the quantum circuits of cryptographic components, we first introduce the gate library and the cost metrics considered in this work. For S-boxes, we consider the circuit cost under the NCT gate set, including the gate cost, the qubit cost, and the full depth cost. For linear layers, the CNOT gate cost and the full depth cost are considered.

Quantum Gates. The NCT gate set is a basic gate library composed of Pauli-X gates, CNOT gates, and Toffoli gates. The Pauli-X gate acts on a single qubit and maps the state $|a\rangle$ to $|a \oplus 1\rangle$. The CNOT gate acts on two qubits $|a\rangle$ and $|b\rangle$, where $|a\rangle$ is the control qubit and $|b\rangle$ is the target qubit. It maps $|a\rangle|b\rangle \mapsto |a\rangle|b \oplus a\rangle$. In the circuit implementation of linear layers, the S-XOR metric [JPST17] is equivalent to the CNOT gate implementation. The Toffoli gate acts on three qubits $|a\rangle$, $|b\rangle$, and $|c\rangle$, where $|a\rangle$ and $|b\rangle$ are control qubits and $|c\rangle$ is the target qubit. It maps $|a\rangle|b\rangle|c\rangle \mapsto |a\rangle|b\rangle|c \oplus a \cdot b\rangle$. The above quantum gates can be represented by the circuit symbols shown in Figure 1.

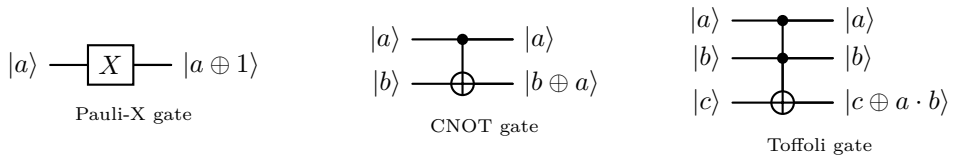


Figure 1: NCT gate set

Qubits. The width of a quantum circuit is defined as the total number of qubits used in the circuit. For an n-bit permutation, the circuit contains n data qubits corresponding to the input variables, and if M ancilla qubits initialized to $|0\rangle$ are introduced, then the circuit width is $W = n + M$.

Full Depth. The full depth of a quantum circuit refers to the maximum accumulated gate depth over all paths from input to output. In particular, since the Toffoli gate is a non-Clifford gate, it needs to be decomposed into the Clifford+T gate set. In this work, we adopt the Toffoli decomposition scheme in [Sel13], so that each Toffoli gate has depth cost 7, following the same depth-cost strategy as in [CLF+24, CBC23].

Table 1: Cost Metrics for the NCT Gate Set

Gate	Gate Count	Full Depth Cost
Pauli-X	1	1
CNOT	1	1
Toffoli	1	7

The quantum gate cost and depth cost of each gate in the NCT gate set are shown in Table 1. These metrics are used in the quantum circuit decision models reviewed in the next subsection.

2.3 Cryptographic Components

In this paper, we focus on the quantum circuit implementation of two core components in cryptographic algorithms: S-boxes and linear layers.

S-boxes. An S-box is a nonlinear mapping defined as a function $f : \{0,1\}^m \to \{0,1\}^n$, which is an important nonlinear component in cryptographic algorithms to ensure the confusion property. In this paper, we focus on bijective n-bit S-boxes: $f : \{0,1\}^n \to \{0,1\}^n$, such S-boxes can be regarded as permutations $[0, 2^n - 1] \mapsto [0, 2^n - 1]$.

For the quantum circuit implementation of S-boxes, this paper introduces two common representations: Look-Up Table (LUT) and bit-sliced representation.

Example 1. We take the S-box of the GIFT [BPP+17] as an example. The LUT and the bit-sliced representation of the GIFT S-box are shown in Table 2 and Table 3, respectively.

Table 2: Look-up Table of GIFT S-box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	1	A	4	C	6	F	3	9	2	D	B	7	5	0	8	E

Table 3: GIFT S-box bit-sliced Implementation

LUT →	**1**	**A**	**4**	**C**	**6**	**F**	**3**	**9**	**2**	**D**	**B**	**7**	**5**	**0**	**8**	**E**	**bit slice ↓**
z_3	0	1	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0x5563
z_2	0	0	1	1	1	1	0	0	0	1	0	1	1	0	0	1	0x3C59
z_1	0	1	0	0	1	1	1	0	1	0	1	1	0	0	0	1	0x4EB1
z_0	1	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	0x8778

Linear transformation. Let the input bit vector be $(x_0, x_1, \ldots, x_{n-1})$ and the output bit vector be $(y_0, y_1, \ldots, y_{m-1})$. A linear transformation over $\mathbb{F}_2$ can be represented by an $m \times n$ binary matrix A. Each output $y_i (0 \le i < m)$ can be expressed as a linear Boolean function of the input $x_j (0 \le j < n)$.

Example 2. Consider the following 4×4 binary matrix:

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

According to the rows of A, the corresponding output bits are

$$\begin{aligned} y_0 &= x_0 \oplus x_1, \\ y_1 &= x_1 \oplus x_2, \\ y_2 &= x_0 \oplus x_2 \oplus x_3, \\ y_3 &= x_1 \oplus x_3. \end{aligned}$$

Equivalently, if we denote the standard basis vectors of $\mathbb{F}_2^4$ by

$$e_0 = 1000, \quad e_1 = 0100, \quad e_2 = 0010, \quad e_3 = 0001,$$

then the four rows of A can be written as linear combinations of these basis vectors:

$$\begin{aligned} e_0 \oplus e_1 &= 1100, \\ e_1 \oplus e_2 &= 0110, \\ e_0 \oplus e_2 \oplus e_3 &= 1011, \\ e_1 \oplus e_3 &= 0101. \end{aligned}$$

2.4 SAT/SMT Problems

The Boolean Satisfiability problem (SAT) is a classic decision problem in computer science. Given a Boolean formula in Conjunctive Normal Form (CNF), SAT determines whether there exists an assignment of Boolean variables that satisfies all clauses of the formula. Based on SAT, Satisfiability Modulo Theories (SMT) extends the satisfiability problem with background theories, such as bit-vectors, integers, and arrays, thereby providing a more expressive and convenient way to represent complex constraints beyond pure Boolean logic.

Table 4: Common SMT-LIBv2 Syntax and Examples

Function	**Syntax**	**Example**
Bit-vector (n-bit)	`(_ BitVec n)`	`(_ BitVec 8)`
Declare variable	`(declare-fun NAME () TYPE)`	`(declare-fun X () (_ BitVec 8))`
Assert constraint	`(assert EXPR)`	`(assert (= X #b10000000))`
If-then-else	`(ite COND THEN ELSE)`	`(ite (= a #b00) ... ...)`
Define function	`(define-fun NAME () TYPE BODY)`	`(define-fun value () (_ BitVec 8) (ite ...))`
Bitwise AND	`(bvand ...)`	`(bvand a b)`
Bitwise XOR	`(bvxor ...)`	`(bvxor a b)`
Unsigned less-than	`(bvult ...)`	`(bvult a b)`
Distinct	`(distinct ...)`	`(distinct a b c)`
Logical OR	`(or ...)`	`(or (= a #b000) (= b #b000))`

In this work, we use the SMT-LIBv2 language to encode the quantum circuit decision problem. SMT-LIBv2 is a standardized input language for SMT solvers and is suitable for describing variables and constraints under different theories. For ease of understanding, Table 4 lists some common SMT-LIBv2 syntax formats.

2.5 Previous Works

For optimizing quantum circuits of cryptographic components, Chen et al. [CLF+24] proposed the following quantum circuit decision problem.

Definition 1 (Quantum circuit decision problem [CLF+24])**.** Given a permutation f, the quantum circuit decision problem requires us to determine whether there exists a quantum circuit implementing f that uses at most M ancilla qubits, exactly G quantum gates, and has full depth at most D.

To encode the quantum circuit decision problem of a permutation f, it is necessary to consider how quantum gates update wire states in a circuit. Each gate directly updates its target wire, and the updated wire states are then used as the inputs of the following gate layers. Therefore, the encoding must track the state of each wire after every gate layer.

Chen et al. proposed a serial encoding model. Given an n-bit permutation, M ancilla qubits, and a fixed number G of quantum gates, the circuit contains $W = n + M$ quantum wires, including n data wires and M ancilla wires. The model represents the circuit as a sequence of G serial gate-level substructures. For each layer k, Boolean state variables are used to record the value of every quantum wire after the first k gates, so that the state evolution of the circuit can be explicitly tracked.

At each gate layer, wiring variables are introduced to specify which wires are connected to the input ports of each gate, and auxiliary variables are used to encode the selected gate type. We briefly review the main components of their encoding and then analyze its variable and constraint complexity.

Encoding Quantum Gates. At each gate layer k, Chen et al.'s model needs to determine both the type of the gate and the quantum wires connected to its inputs. Let $z_s[k]$ denote the state of the s-th quantum wire before the k-th gate, where $0 \leq s < W$. The three

input variables of the k-th gate are $q_{3k}, q_{3k+1}, q_{3k+2}$. They are selected from all quantum wires by wiring variables $a_{r,s}^{(k)}$, where $r \in \{0,1,2\}$:

$$q_{3k+r} = \bigoplus_{s=0}^{W-1} \left(a_{r,s}^{(k)} \cdot z_s[k] \right), \qquad r \in \{0,1,2\}.$$

Since $a_{r,s}^{(k)}$ is combined with the bit-sliced state value $z_s[k]$, it is represented as a 2^n-bit vector for an n-bit S-box. Here $a_{r,s}^{(k)} = \mathbf{1}_{2^n}$ means that the r-th input port of the k-th gate is connected to the s-th quantum wire. Therefore, only-one constraints are needed to ensure that each input port selects exactly one wire, and mutual-exclusion constraints are required to ensure that different input ports are not connected to the same wire.

The gate type is encoded by two variables $b_{2k} \| b_{2k+1}$. Since these variables are combined with bit-sliced state values, they are represented as 2^n-bit masks, i.e., $b_{2k}, b_{2k+1} \in \{\mathbf{0}_{2^n}, \mathbf{1}_{2^n}\}$. The valid encodings are

$$\mathbf{0}_{2^n}\mathbf{0}_{2^n} : \text{CNOT}, \qquad \mathbf{0}_{2^n}\mathbf{1}_{2^n} : \text{Pauli-X}, \qquad \mathbf{1}_{2^n}\mathbf{0}_{2^n} : \text{Toffoli},$$

The invalid case $\mathbf{1}_{2^n}\mathbf{1}_{2^n}$ is excluded by $b_{2k} \cdot b_{2k+1} = \mathbf{0}_{2^n}$. Based on the selected gate type, the output value t_k of the k-th gate is encoded as

$$t_k = q_{3k} \oplus q_{3k+1} \oplus b_{2k} q_{3k+1} \oplus b_{2k} q_{3k+1} q_{3k+2} \oplus b_{2k+1} q_{3k+1} \oplus b_{2k+1}.$$

State Update. Since quantum gates act on fixed physical wires, the output of the current gate must overwrite the state of its target wire, while all other wires remain unchanged. In Chen et al.'s model, the first gate input q_{3k} is treated as the target input. Thus, if the target input is connected to the s-th wire, then the state of that wire is updated to t_k, while the states of the other wires are copied to the next layer. This update can be summarized as

$$z_s[k+1] = \begin{cases} t_k, & \text{if the target input selects wire } s, \\ z_s[k], & \text{otherwise.} \end{cases}$$

Depth Update. To constrain the full depth of the circuit, Chen et al.'s model introduces a depth variable $depth_s[k]$ for each quantum wire. When a gate acts on several wires, the depths of all participating wires are updated to the maximum accumulated depth among these wires plus the cost of the gate, while the depths of non-participating wires remain unchanged.

The model therefore first needs to identify the participating wires of the current gate. For a Pauli-X gate, only the target wire participates, and it can be identified directly by the wiring variable $a_{0,s}^{(k)}$, since the first input port q_{3k} is treated as the target input. For a CNOT gate, both the target wire and the control wire participate. Chen et al.'s model introduces auxiliary variables

$$c_s^{(k)} = a_{0,s}^{(k)} \oplus a_{1,s}^{(k)}$$

to indicate whether wire s is connected to one of the two involved input ports. For a Toffoli gate, all three input ports participate, and auxiliary variables

$$d_s^{(k)} = a_{0,s}^{(k)} \oplus a_{1,s}^{(k)} \oplus a_{2,s}^{(k)} = c_s^{(k)} \oplus a_{2,s}^{(k)}$$

are used to identify the three participating wires.

After the participating wires are identified, the model compares their current depths to compute the new depth. For a Pauli-X gate, the target depth is increased by the cost of the Pauli-X gate. For a CNOT gate with participating wires i and j, both wires are updated to

$$\max(depth_i[k], depth_j[k]) + 1.$$

For a Toffoli gate with participating wires i, j, and h, the three wires are updated to

$$\max(depth_i[k], depth_j[k], depth_h[k]) + 7.$$

Thus, Chen et al.'s model must encode not only the selection of participating wires but also the depth comparisons among the selected wires.

Analysis of Chen et al.'s Encoding. Chen et al.'s encoding can be viewed as a wiring-based encoding, which is similar in idea to classical SAT encodings for Boolean circuits [Sto16]. Although this representation is expressive, it has two main limitations: high encoding complexity and the exact-G formulation of the decision problem.

Encoding complexity. For each gate layer k, Chen et al.'s model introduces wiring variables $a_{r,s}^{(k)}$ to indicate whether the r-th input port of the current gate is connected to the s-th quantum wire. These wiring variables require only-one constraints and mutual-exclusion constraints. Moreover, the state update and the depth update depend on these wiring variables. The depth encoding further requires auxiliary variables for identifying the participating wires and constraints for comparing their accumulated depths. Together, these components generate a large number of constraints, making the model difficult to scale as the number of wires W and gate layers G increase.

Exact-G limitation. Chen et al.'s decision problem is suitable for the exact-G setting, where the circuit is required to contain exactly G quantum gates. However, such an exact-gate model does not directly answer whether a circuit can be implemented using no more than a given number of gates. In particular, if Chen et al.'s model returns UNSAT for exactly G gates, this does not imply that the instance is also UNSAT for $G-1$ gates. The following simple example illustrates this non-monotonicity.

Example 3. Consider the 3-bit permutation corresponding to a single Toffoli gate:

$$(a, b, c) \mapsto (a, b, c \oplus a \cdot b).$$

This permutation can be implemented by exactly one Toffoli gate. However, if the model requires exactly two NCT gates and does not allow an identity operation, then this one-gate implementation cannot be represented by simply padding an empty layer. As a result, the exact-two-gate instance is UNSAT even though a valid one-gate implementation exists. This shows that exact-G satisfiability is not monotone with respect to G.

To capture the more natural optimization question, we define an at-most-G decision problem.

Definition 2 (At-most-G quantum circuit decision problem)**.** Given a permutation f, the at-most-G quantum circuit decision problem requires us to determine whether there exists a quantum circuit implementing f that uses at most M ancilla qubits, at most G non-trivial quantum gates, and has full depth at most D.

Here, non-trivial quantum gates refer to actual NCT gates, including Pauli-X, CNOT, and Toffoli gates, while identity/null gates used only for padding are not counted.

The above discussion motivates the design of two SMT encodings corresponding to the two decision forms. In the next section, we present our exact-G encoding and at-most-G encoding with null gates.

3 Novel SMT-Based Encoding

In this section, we present our SMT encodings for quantum circuit optimization. The goal is to synthesize an NCT quantum circuit for a given permutation f. We first introduce an exact-G encoding, which solves the exact-G decision problem. Then, we extend this encoding by introducing null gates, so that the model can capture the at-most-G decision problem. Finally, we describe several additional constraints used to improve the search efficiency.

3.1 Exact-G Encoding

In our model, the quantum wires are regarded as fixed entities, and each gate updates the state of its target wire according to its control wires. The solver no longer determines the explicit wiring connections between gates and quantum wires. Instead, it decides the gate type and the target/control wire indices at each gate position. We summarize the basic encoding in Algorithm 1.

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be an n-bit permutation. Let M denote the number of ancilla qubits. The total number of quantum wires is $W = n + M$. All quantum wires, including data wires and ancilla wires, are indexed from 0 to $W - 1$. The circuit is represented as a sequence of G gate layers. In the basic model, each layer must contain one non-trivial NCT gate. Therefore, this basic model solves an exact-G version of the quantum circuit decision problem.

3.1.1 Bit-Sliced State Representation

We represent the S-box input and output truth tables using bit-slicing. For each $0 \le i < n$, $X_i \in \mathbb{F}_2^{2^n}$ denotes the bit-sliced truth-table vector of the i-th input bit, and $Y_i \in \mathbb{F}_2^{2^n}$ denotes the bit-sliced truth-table vector of the i-th output bit of f.

For the circuit state, $V_i[k]$ denotes the bit-sliced value stored on the i-th wire after the first k gates have been applied, where $0 \le i < W$ and $0 \le k \le G$. The initial data wires are assigned to the input bit-slices, and the ancilla wires are initialized to zero:

$$V_i[0] = X_i, 0 \le i < n. \qquad V_i[0] = \mathbf{0}_{2^n}, n \le i < W.$$

If a depth bound is considered, we also introduce a depth variable $D_i[k]$, which denotes the cumulative depth of wire i after the first k gates. Initially,

$$D_i[0] = 0, \qquad 0 \le i < W.$$

3.1.2 Gate Variables

At each layer k ($1 \le k \le G$), the model declares one operation variable op_k and three wire-index variables $tgt_k, c1_k, c2_k$. The variable tgt_k denotes the target wire, while $c1_k$ and $c2_k$ denote the possible control wires. The three wire indexes are encoded as bit-vectors of length $\ell = \lceil \log_2 W \rceil$. To ensure that the selected indexes are valid wires, the following constraints are added:

$$tgt_k < W, \qquad c1_k < W, \qquad c2_k < W.$$

For example, when $W = 5$, the valid wire indexes range from 0 to 4. Therefore, the constraint $tgt_k < W$ can be encoded in SMT as

```
(assert (bvult tgt_k #b101))
```

In the basic fixed-length model, only the three NCT gates are allowed. Thus, the 2-bit operation variable satisfies

$$op_k < 11_2.$$

In SMT, this constraint can be encoded as

```
(assert (bvult op_k #b11))
```

The three possible values of op_k are defined as follows:

op_k	Gate type	Operation on the target wire
00_2	Pauli-X	$x_{tgt} \leftarrow x_{tgt} \oplus 1$
01_2	CNOT	$x_{tgt} \leftarrow x_{tgt} \oplus x_{c1}$
10_2	Toffoli	$x_{tgt} \leftarrow x_{tgt} \oplus x_{c1}x_{c2}$

Algorithm 1: Basic Fixed-Length SMT Encoding

Input: An n-bit permutation f, gate number G, ancilla number M, and optional depth bound D

Output: SAT with a gate sequence $\{op_k, tgt_k, c1_k, c2_k\}_{k=1}^{G}$, or UNSAT

$W \leftarrow n + M$;
Construct the input bit-slices $X_0, \ldots, X_{n-1}$ and output bit-slices $Y_0, \ldots, Y_{n-1}$;
Initialize the data wires by $X_0, \ldots, X_{n-1}$ and initialize all ancilla wires to zero;
if *a depth bound D is considered* **then**
    Initialize the depth of every wire to zero;
end
for $k = 1$ **to** G **do**
    $op_k \in \{00_2, 01_2, 10_2\}$;
    $tgt_k, c1_k, c2_k \in \{0, \ldots, W-1\}$;
    assert distinct$(tgt_k, c1_k, c2_k)$;
    $v_{c1}^{(k)} \leftarrow V_{F_{\text{bit2Int}}(c1_k)}[k-1]$;
    $v_{c2}^{(k)} \leftarrow V_{F_{\text{bit2Int}}(c2_k)}[k-1]$;
    switch op_k **do**
        case 00_2 **do**
            $mask_k \leftarrow \mathbf{1}_{2^n}$;
        end
        case 01_2 **do**
            $mask_k \leftarrow v_{c1}^{(k)}$;
        end
        case 10_2 **do**
            $mask_k \leftarrow v_{c1}^{(k)} \cdot v_{c2}^{(k)}$;
        end
    end
    for $i = 0$ **to** $W-1$ **do**
        $V_i[k] \leftarrow \text{ite}(F_{\text{Int2bit}}(i) = tgt_k, V_i[k-1] \oplus mask_k, V_i[k-1])$;
    end
    if *a depth bound D is considered* **then**
        $d_{tgt}^{(k)} \leftarrow D_{F_{\text{bit2Int}}(tgt_k)}[k-1]$;
        $d_{c1}^{(k)} \leftarrow D_{F_{\text{bit2Int}}(c1_k)}[k-1]$;
        $d_{c2}^{(k)} \leftarrow D_{F_{\text{bit2Int}}(c2_k)}[k-1]$;
        Compute the new depth $newD_k$ of the current gate according to its type;
        for $i = 0$ **to** $W-1$ **do**
            $D_i[k] \leftarrow \text{ite}(Inv(i,k), newD_k, D_i[k-1])$;
        end
    end
end
for $j = 0$ **to** $n-1$ **do**
    assert $\bigvee_{w=0}^{W-1}(Y_j = V_w[G])$;
end
if *a depth bound D is considered* **then**
    for $i = 0$ **to** $W-1$ **do**
        assert $D_i[G] \leq D$;
    end
end
Invoke the SMT solver;
if *SAT* **then**
    Extract the gate sequence $\{op_k, tgt_k, c1_k, c2_k\}_{k=1}^{G}$;
end
else
    Return UNSAT;
end

In addition, it must be ensured that the target wire and control wires of any gate are different from each other, that is, tgt_k, $c1_k$, and $c2_k$ are pairwise different. In SMT, this constraint can be encoded as

```
(assert (distinct tgt_k c1_k c2_k))
```

3.1.3 State Update

For each layer k, the current values of the two control wires are selected according to $c1_k$ and $c2_k$:

$$v_{c1}^{(k)} = V_{F_{\text{bit2Int}}(c1_k)}[k-1], \qquad v_{c2}^{(k)} = V_{F_{\text{bit2Int}}(c2_k)}[k-1],$$

where F_{bit2Int} converts a bit-vector index into the corresponding integer wire index.

In SMT, the selection of $v_{c1}^{(k)}$ and $v_{c2}^{(k)}$ can be implemented by nested `ite` expressions. For example, when $W = 5$, $2^n = 16$, and $k = 1$, the value selected by $c1_1$ can be written as

```
(define-fun val_c1_1 () (_ BitVec 16)
  (ite (= c1_1 #b000) V_0_0
  (ite (= c1_1 #b001) V_1_0
  (ite (= c1_1 #b010) V_2_0
  (ite (= c1_1 #b011) V_3_0 V_4_0)))))
```

To compute the wire state after applying the gate, we define a bit-sliced mask that represents the value to be XORed into the target wire. The mask generated by the gate at layer k is defined by

$$mask_k = \begin{cases} \mathbf{1}_{2^n}, & op_k = 00_2, \\ v_{c1}^{(k)}, & op_k = 01_2, \\ v_{c1}^{(k)} \cdot v_{c2}^{(k)}, & op_k = 10_2. \end{cases}$$

In SMT, the mask can also be implemented by nested `ite` expressions. Taking $k = 1$ and $2^n = 16$ as an example, we write

```
(define-fun mask_1 () (_ BitVec 16)
  (ite (= op_1 #b00) #b1111111111111111
  (ite (= op_1 #b01) val_c1_1 (bvand val_c1_1 val_c2_1))))
```

Then the state update for each wire is given by

$$V_i[k] = \begin{cases} V_i[k-1] \oplus mask_k, & F_{\text{Int2bit}}(i) = tgt_k, \\ V_i[k-1], & F_{\text{Int2bit}}(i) \neq tgt_k, \end{cases}$$

where F_{Int2bit} converts an integer wire index into its bit-vector encoding. This update rule means that only the target wire is modified, while all other wires remain unchanged.

In SMT, this update can be expressed using an `ite` statement. For example, for the wire with index $i = 0$ at layer $k = 1$, the update is encoded as

```
(assert (= V_0_1 (ite (= tgt_1 #b000) (bvxor V_0_0 mask_1) V_0_0)))
```

3.1.4 Depth Update

For the NCT gate library used in this work, the depth costs of Pauli-X and CNOT gates are set to 1, while the Toffoli gate has depth cost 7. At each layer, the depth of all wires participating in the current gate is updated to the same new value, determined by the maximum previous depth of the participating wires plus the gate cost.

We first define a Boolean predicate $Inv(i,k)$, which indicates whether wire i participates in the gate at layer k:

$$Inv(i,k) = \begin{cases} F_{\text{Int2bit}}(i) = tgt_k, & op_k = 00_2, \\ F_{\text{Int2bit}}(i) = tgt_k \vee F_{\text{Int2bit}}(i) = c1_k, & op_k = 01_2, \\ F_{\text{Int2bit}}(i) = tgt_k \vee F_{\text{Int2bit}}(i) = c1_k \vee F_{\text{Int2bit}}(i) = c2_k, & op_k = 10_2. \end{cases}$$

In SMT, $Inv(i,k)$ can be implemented by nested `ite` expressions according to the gate type. For example, for $i = 0$ and $k = 1$, $Inv(0,1)$ can be written as

```
(define-fun inv_0_1 () Bool
  (ite (= op_1 #b00) (= tgt_1 #b000)
  (ite (= op_1 #b01) (or (= tgt_1 #b000) (= c1_1 #b000))
       (or (= tgt_1 #b000) (= c1_1 #b000) (= c2_1 #b000)))))
```

Let $d_{tgt}^{(k)} = D_{F_{\text{bit2Int}}(tgt_k)}[k-1]$, $d_{c1}^{(k)} = D_{F_{\text{bit2Int}}(c1_k)}[k-1]$, $d_{c2}^{(k)} = D_{F_{\text{bit2Int}}(c2_k)}[k-1]$. The new depth produced at layer k is

$$newD_k = \begin{cases} d_{tgt}^{(k)} + 1, & op_k = 00_2, \\ \max(d_{tgt}^{(k)}, d_{c1}^{(k)}) + 1, & op_k = 01_2, \\ \max(d_{tgt}^{(k)}, d_{c1}^{(k)}, d_{c2}^{(k)}) + 7, & op_k = 10_2. \end{cases}$$

In SMT, the maximum values and the new depth can be encoded as follows. Here, `max12_1` computes $\max(d_{tgt}^{(1)}, d_{c1}^{(1)})$, while `max123_1` computes $\max(d_{tgt}^{(1)}, d_{c1}^{(1)}, d_{c2}^{(1)})$:

```
(define-fun max12_1 () (_ BitVec 6) (ite (bvuge d_tgt_1 d_c1_1) d_tgt_1 d_c1_1))
(define-fun max123_1 () (_ BitVec 6) (ite (bvuge max12_1 d_c2_1) max12_1 d_c2_1))
(define-fun new_d_1 () (_ BitVec 6)
  (ite (= op_1 #b00) (bvadd d_tgt_1 #b000001)
  (ite (= op_1 #b01) (bvadd max12_1 #b000001) (bvadd max123_1 #b000111))))
```

Finally, the depth of each wire is updated by

$$D_i[k] = \begin{cases} newD_k, & Inv(i,k) = \text{true}, \\ D_i[k-1], & Inv(i,k) = \text{false}. \end{cases}$$

In SMT, taking $i = 0$ and $k = 1$ as an example, this update can be written as

```
(assert (= D_0_1 (ite inv_0_1 new_d_1 D_0_0)))
```

3.1.5 Final Constraints

- The output constraint requires each target output to appear on one of the wires at the final layer:

$$\forall j \in [0,n), \qquad \bigvee_{w=0}^{W-1} (Y_j = V_w[G])\,.$$

  For example, let $G = 9$ and $W = 5$. Then the constraint for the first output Y_0 can be written as

```
(assert (or (= V_0_9 Y_0) (= V_1_9 Y_0)
    (= V_2_9 Y_0) (= V_3_9 Y_0) (= V_4_9 Y_0)))
```

- If a depth bound D is considered, the final depth of every wire must not exceed D:

$$\forall i \in [0,W), \qquad D_i[G] \le D.$$

  For example, let $D = 31$. If the depth variables are encoded as 6-bit bit-vectors, then $D = 31$ is represented as 011111_2, and the final depth constraints can be written as

```
(assert (bvule D_0_9 #b011111))
(assert (bvule D_1_9 #b011111))
(assert (bvule D_2_9 #b011111))
(assert (bvule D_3_9 #b011111))
(assert (bvule D_4_9 #b011111))
```

3.2 At-Most-G Encoding with Null Gates

The basic exact-G encoding solves an exact-G decision problem: every layer must contain one non-trivial NCT gate. As discussed in Section 2.2, this exact-G formulation does not directly capture the question of whether a circuit can be implemented using no more than G gates.

To address this issue, we extend the operation encoding by introducing a fourth gate type, called the *null gate*. In the extended model, the operation variable op_k may take all four 2-bit values:

$$op_k \in \{00_2, 01_2, 10_2, 11_2\}.$$

The first three values still denote Pauli-X, CNOT, and Toffoli gates, respectively. The new value $op_k = 11_2$ denotes a null gate. A null gate is an identity operation: it does not change any wire state and does not increase the circuit depth.

With null gates, any circuit using $g \leq G$ non-trivial gates can be embedded into a G-layer instance by filling the remaining $G - g$ layers with null gates. Therefore, the satisfiability of the extended G-layer model represents the existence of a circuit with at most G non-trivial gates, rather than exactly G gates.

Compared with the basic fixed-length model, only a few parts of the encoding need to be modified.

- First, the constraint $op_k < 11_2$ is removed, so that $op_k = 11_2$ becomes a valid operation type.
- Second, the mask definition is extended to include the null gate:

$$mask_k = \begin{cases} \mathbf{1}_{2^n}, & op_k = 00_2, \\ v_{c1}^{(k)}, & op_k = 01_2, \\ v_{c1}^{(k)} \cdot v_{c2}^{(k)}, & op_k = 10_2, \\ \mathbf{0}_{2^n}, & op_k = 11_2. \end{cases}$$

  Thus, when $op_k = 11_2$, the generated mask is the all-zero vector. The target wire is XORed with $\mathbf{0}_{2^n}$, and therefore its state remains unchanged.

- Third, the predicate $Inv(i,k)$ is modified so that no wire is involved in a null gate:

$$\begin{aligned} Inv(i,k) = (&op_k = 00_2 \wedge F_{\text{Int2bit}}(i) = tgt_k) \\ &\vee (op_k = 01_2 \wedge (F_{\text{Int2bit}}(i) = tgt_k \vee F_{\text{Int2bit}}(i) = c1_k)) \\ &\vee (op_k = 10_2 \wedge (F_{\text{Int2bit}}(i) = tgt_k \vee F_{\text{Int2bit}}(i) = c1_k \vee F_{\text{Int2bit}}(i) = c2_k)). \end{aligned}$$

  For $op_k = 11_2$, none of the above clauses is satisfied. Hence, $Inv(i,k) = \text{false}$, for every wire i, and all depth values are preserved.

- Equivalently, the new depth value can be extended as

$$newD_k = \begin{cases} d_{tgt}^{(k)} + 1, & op_k = 00_2, \\ \max(d_{tgt}^{(k)}, d_{c1}^{(k)}) + 1, & op_k = 01_2, \\ \max(d_{tgt}^{(k)}, d_{c1}^{(k)}, d_{c2}^{(k)}) + 7, & op_k = 10_2, \\ 0, & op_k = 11_2. \end{cases}$$

  The last value is only a placeholder, because when $op_k = 11_2$, $Inv(i,k)$ is false for all wires and $newD_k$ is not written to any depth variable.

3.3 Redundancy-Eliminating and Symmetry-Breaking Constraints

The null-gate encoding provides a flexible at-most-G formulation, but it also increases the number of equivalent encodings. Moreover, even in the exact-G model, some addressing variables are semantically irrelevant for certain gate types. For instance, the Pauli-X gate only uses the target wire, so the control indexes $c1_k$ and $c2_k$ do not affect the gate semantics. Similarly, CNOT uses only tgt_k and $c1_k$, while $c2_k$ is irrelevant. For the Toffoli gate, the two control wires have identical semantics, so exchanging $c1_k$ and $c2_k$ gives an equivalent encoding of the same gate. Therefore, we impose the canonical ordering constraint $c1_k < c2_k$ to remove this symmetry. For the null gate, all addressing variables are fixed to zero so that the identity operation has a unique representation.

To reduce redundancy and break symmetry, we can replace the global constraint $\text{distinct}(tgt_k, c1_k, c2_k)$ with the following constraints.

$$
\begin{aligned}
&op_k = 00_2 \Rightarrow c1_k = 0 \wedge c2_k = 0,\\
&op_k = 01_2 \Rightarrow tgt_k \neq c1_k \wedge c2_k = 0,\\
&op_k = 10_2 \Rightarrow \text{distinct}(tgt_k, c1_k, c2_k) \wedge c1_k < c2_k,\\
&op_k = 11_2 \Rightarrow tgt_k = 0 \wedge c1_k = 0 \wedge c2_k = 0.
\end{aligned}
$$

For the basic exact-G model, the last rule for $op_k = 11_2$ is omitted.

These constraints do not remove any semantically distinct circuit. They only select one canonical representative from each equivalence class of syntactically different but semantically identical encodings. Therefore, they reduce the search space of the solver without changing the realizability of the target permutation.

Experimental results. To evaluate the effect of the proposed constraints, we compare the solving time of the basic exact-G model and the model with redundancy-eliminating and symmetry-breaking constraints on UNSAT instances. In this experiment, no depth bound is imposed, and the solver only checks whether there exists an implementation using exactly G gates. The tested instances are used to evaluate how efficiently the solver can prove that no such exact-G implementation exists.

Table 5: Effect of redundancy-eliminating and symmetry-breaking constraints on UNSAT instances

UNSAT instances without depth constraints						
S-box	Size	#Gate	Kissat		CaDiCaL	
			basic(s)	break(s)	basic(s)	break(s)
KECCAK	5	9	92.1	**63.1**	128.2	**99.5**
KECCAK	5	10	1480.0	**974.8**	3483.1	**1243.9**
KECCAK	5	11	115742.2	**48406.5**	132668.1	**60444.3**
ASCON	5	10	285.1	**179.4**	759.5	**489.1**
ASCON	5	11	7065.1	**6329.9**	13367.7	**8710.8**
ASCON	5	12	450960.7	**400399.9**	375110.8	**244631.6**
APN S-box [BDMW10]	6	12	50990.5	**43218.4**	83570.0	**53943.7**

basic denotes the original exact-G encoding, while *break* denotes the encoding with redundancy-eliminating and symmetry-breaking constraints.

We use Bitwuzla to solve the generated SMT-LIBv2 formulas, and test both Kissat and CaDiCaL as the underlying SAT solvers. All timings in Table 5 were measured on a machine equipped with an Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz. The experimental results show that these constraints reduce the solving time.

3.4 Toffoli-Count Lower Bound and Full Depth Constraints

3.4.1 Toffoli-Count Lower Bound

We analyze a lower bound on the number of Toffoli gates in an NCT implementation. This bound is useful for proving the optimality of the Toffoli count and for adding additional pruning constraints in the SMT model.

Before stating the bound, we recall the homogeneous quadratic part. For a Boolean function f in algebraic normal form, its homogeneous quadratic part is the sum of all degree-two monomials.

Theorem 1. *Let $F = (f_0, \ldots, f_{n-1})$ be an n-bit permutation, and let q_i be the homogeneous quadratic part of f_i. If $\dim\langle q_0, \ldots, q_{n-1}\rangle = t$, then any NCT implementation of F, even with additional ancilla qubits initialized to 0, requires at least t Toffoli gates.*

Proof. For a wire function h, let $Q(h)$ denote its homogeneous quadratic part. Initially, all data wires are input variables and all ancilla wires are initialized to 0, so the space spanned by the homogeneous quadratic parts has dimension 0.

Pauli-X and CNOT gates cannot enlarge this space, since they only add constants or take linear combinations of existing wire functions. Consider a Toffoli gate with control-wire functions u, v and target-wire function w. The target is updated as $w \leftarrow w \oplus uv$. The homogeneous quadratic part added by uv is

$$Q(uv) = u^{(0)}Q(v) \oplus v^{(0)}Q(u) \oplus u^{(1)}v^{(1)},$$

where $u^{(0)}, v^{(0)}$ are the constant parts and $u^{(1)}, v^{(1)}$ are the linear parts. The first two terms already lie in the existing quadratic space, while $u^{(1)}v^{(1)}$ is a single homogeneous quadratic polynomial. Hence, one Toffoli gate can increase the dimension of the homogeneous quadratic space by at most one.

Therefore, after g Toffoli gates, the homogeneous quadratic parts of all wire functions span a space of dimension at most g. Since the output functions of F have homogeneous quadratic parts spanning a space of dimension t, we must have $g \geq t$. □

Theorem 1 gives a lower bound on the number of Toffoli gates from the ANF representation of an S-box. If the homogeneous quadratic parts of the output functions span a t-dimensional space, then any NCT implementation must use at least t Toffoli gates. In our encodings, $op_k = 10_2$ denotes that the gate at layer k is a Toffoli gate. We define

$$N_T = |\{\, k \in \{1, \ldots, G\} : op_k = 10_2 \,\}|.$$

Then the SMT model must satisfy $N_T \geq t$. To search for an implementation with the minimum possible Toffoli count, we set $N_T = t$. If this constraint is UNSAT, we relax it to $N_T = t + 1, t + 2, \ldots$ to search for implementations with more Toffoli gates.

3.4.2 Depth-checkpoint constraints

When a depth bound is imposed, the final depth constraint only checks the depth after all G layers have been applied. However, the depth of each wire is monotonically non-decreasing during circuit construction. Once the cumulative depth of any wire exceeds the bound D at an intermediate layer, no later gate can reduce it. Therefore, such a partial gate sequence can be safely pruned before reaching the final layer.

Adding depth constraints after every layer may introduce many additional constraints. Instead, we add depth-checkpoint constraints only from the first layer at which exceeding the depth bound is theoretically possible. Let $d_{\max}$ be the maximum depth cost of a single gate. In the NCT cost model, $d_{\max} = 7$, because the Toffoli gate has depth cost 7. After the first k gates, the maximum possible serial depth is at most $kd_{\max}$. Therefore, no prefix

can violate the depth bound when $kd_{\max} \leq D$. The first layer at which a violation may occur is

$$k_0 = \left\lfloor \frac{D}{d_{\max}} \right\rfloor + 1.$$

From layer k_0 onward, we add the prefix constraints

$$\forall i \in [0, W), \qquad D_i[k] \leq D, \qquad k_0 \leq k < G.$$

The final layer G is still checked by the final depth constraint. This strategy avoids adding redundant early constraints while still pruning prefixes that have already exceeded the depth bound.

Experimental results. To evaluate the effect of depth-checkpoint constraints, we test the KECCAK S-box under the at-most-G model with a fixed depth bound 52. We compare two encodings: the basic at-most-G encoding and the encoding with depth-checkpoint constraints. The experimental results are shown in Table 6.

Table 6: Effect of depth-checkpoint constraints on KECCAK

S-box	Gate bound	Depth bound	Basic (s)	Depth-checkpoint (s)
KECCAK	19	52	20929.9	**14686.8**
KECCAK	16	52	322783.3	**264035.2**

Basic denotes the at-most-G encoding without depth-checkpoint constraints, while *Depth-check* denotes the encoding with depth-checkpoint constraints.

The experiments are performed using Bitwuzla with Kissat as the underlying SAT solver. All timings in Table 6 were measured on a machine equipped with an Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz. These results indicate that prefix depth-checkpoint constraints can improve solving efficiency.

4 Applications

In this section, we apply the proposed SMT modeling scheme to the automated search and optimization of quantum circuits for cryptographic components, including S-boxes and linear layers. We first describe two complementary search strategies used in our experiments: a gate-count-first strategy, which searches for the minimum number of gates without imposing a depth bound, and a depth-first strategy, which starts from a feasible circuit and further reduces the full depth under a given gate bound.

4.1 Search Strategies

Strategy 1: Gate-count-first search. The first strategy prioritizes the number of gates. In this mode, the depth bound is disabled, and depth variables and depth constraints are omitted. To minimize the gate count, we use the basic exact-G model. Specifically, given a search range $[G_{\min}, G_{\max}]$, we increase G from $G_{\min}$ to $G_{\max}$, and each value of G is solved as an independent decision problem. The detailed procedure is shown in Algorithm 2.

If the solver returns SAT for the first time at $G = G^*$, and all smaller gate numbers in the search range have already been checked and returned UNSAT, then G^* is the minimum gate count under the given ancilla setting. This strategy is suitable for gate-count optimization because the exact-G model has a smaller operation domain. In contrast, the at-most-G model introduces the additional null-gate value $op_k = 11_2$, which enlarges the operation search space, since the solver must also decide which layers are real gates and which layers are null gates. Therefore, when the objective is only to minimize the number of gates, the exact-G model is usually more efficient in practice.

Algorithm 2: Gate-Count-First Search

Input: An n-bit permutation f, ancilla number M, initial gate number $G_{\min}$, and maximum gate number $G_{\max}$

Output: The minimum gate number G^* and a feasible circuit C^*, or no circuit found within the searched range

```
Disable the depth bound and omit all depth variables and depth constraints;
for G = G_min to G_max do
    Construct the basic exact-G SMT model for f with M ancilla qubits;
    Invoke the SMT solver;
    if SAT then
        Extract the circuit C*;
        G* ← G;
        return (G*, C*);
    end
end
return no circuit found for G_min ≤ G ≤ G_max;
```

Strategy 2: Depth-first search. The second strategy prioritizes circuit depth. The detailed procedure is shown in Algorithm 3. In this mode, we use the at-most-G model with null gates, because depth optimization usually requires a larger gate bound to allow circuit structures with smaller depth. We first apply the gate-count-first search to obtain a feasible circuit C_0 with gate count G^*, and compute its actual full depth, denoted by D_0. This value serves as the initial depth upper bound.

Algorithm 3: Depth-First Search

Input: An n-bit permutation f, ancilla number M, a feasible circuit C_0, its gate count G^*, and a chosen gate bound G'

Output: The minimum full depth found under gate bound G', and the corresponding circuit

```
Assume G' ≥ G*;
Compute the actual full depth of C_0, denoted by D_0;
C_best ← C_0, D_best ← D_0;
while D_best > 0 do
    D_try ← D_best − 1;
    Construct the at-most-G' SMT model with null gates and depth bound D_try;
    Invoke the SMT solver;
    if SAT then
        Extract the circuit C;
        Compute the actual full depth of C, denoted by D_C;
        C_best ← C, D_best ← D_C;
    end
    else
        return (D_best, C_best);
    end
end
return (D_best, C_best);
```

Then we choose a gate bound G' satisfying $G' \geq G^*$, typically slightly larger than the minimum gate count obtained in the first strategy. The first depth bound to be tested is set to $D_0 - 1$. If the instance is satisfiable, a circuit with smaller depth is obtained, and

we compute its actual full depth D_1. The next search is then performed with depth bound $D_1 - 1$. This process is repeated until the solver returns UNSAT.

When the solver returns UNSAT for depth bound $D_{\text{best}} - 1$, the last satisfiable circuit has the minimum full depth found under the chosen gate bound G' and the fixed ancilla number M. Compared with the exact-G model, the at-most-G model is more suitable for this strategy because it allows the solver to use no more than G' non-trivial gates. Therefore, the solver can explore circuits with different effective gate counts under the same gate bound, which is useful for finding lower-depth implementations.

These two strategies are complementary. The gate-count-first strategy determines how many non-trivial gates are necessary under the fixed ancilla setting, while the depth-first strategy explores whether a larger gate bound can provide more parallelism and reduce the circuit depth. Therefore, the overall optimization process can reveal the trade-off between gate count and full depth.

4.2 Efficiency Comparison on 4-bit S-boxes

To evaluate the solving efficiency of the proposed encoding scheme, we compare our model with Chen et al.'s model on 4-bit S-boxes. For a fair comparison, the STP solver is uniformly used in the experiments, since it supports both the CVC format used by Chen et al.'s model and the SMT-LIBv2 format adopted in our work. In this comparison, we use the basic exact-G encoding and solve the instances under the same gate and depth bounds. All experiments are conducted in the same environment equipped with an Intel Core i9-14900K CPU @ 3.20 GHz.

To better explain the efficiency difference, we also compare the encoding size of the two models at the SMT representation. Chen et al.'s model explicitly encodes the wiring relation between each gate and the quantum wires. Thus, it introduces many selector variables and validity constraints for the possible gate inputs. In contrast, our model regards the quantum wires as fixed entities and only lets the solver choose the gate type and the target/control wire indices at each gate position. This reduces the number of decision variables and constraints. The comparison is summarized in Table 7.

Table 7: Encoding-size Comparison

Item	Chen et al.'s Encoding [CLF+24]	This Work
Decision variables	$(3W+2)G$ selector bit-vectors	$4G$ index variables
Legality constraints	$(3W+3)G$	$5G$
State-update constraints	$(W+4)G$	$(W+3)G$
Depth-update constraints	$\left(3W + 2\binom{W}{2} + 3\binom{W}{3}\right)G$	$(6+2W)G$

Here W denotes the circuit width and G denotes the number of gates. In Chen et al.'s encoding, the selector variables are bit-vectors over the truth-table domain, while in our encoding the index variables have bit-width $\lceil \log_2 W \rceil$.

The experimental results show that, except for the PRESENT S-box, the proposed encoding achieves a clear improvement in solving speed. For example, when solving the ELEPHANT S-box, Chen et al.'s model takes more than 7200 seconds, while our model takes less than 56 seconds, achieving a speedup of about 130 times. This result is consistent with the encoding-size comparison above, since our model uses fewer selector variables and much simpler constraints. The PRESENT instance is an exception, which suggests that the actual solving time is still affected by instance-dependent solver behavior, such as preprocessing and branching heuristics.

Table 8: Time Comparison for 4-bit S-boxes

S-box	#Gate	Full depth	Chen et al. (s) [CLF+24]	This work (s)	Speedup Multiplier
PRØST [KLL+14]	4	28	0.08	**0.03**	**2.67**
RECTANGLE [ZBL+15]	10	32	74.81	**5.74**	**13.03**
GIFT [BPP+17]	8	31	15.68	**2.85**	**5.50**
GIFT [BPP+17]	9*	30	533.83	**397.07**	**1.34**
PRESENT [SOT+21]	11	32	60.91	497.03	0.12
SKINNY [BJK+16]	10	31	359.63	**6.75**	**53.28**
MIDORI [BBI+15]	10	31	421.58	**99.12**	**4.25**
LBLOCK [WZ11]	9	31	14.51	**4.09**	**3.55**
UBLOCK [WL21]	8	31	11.18	**2.59**	**4.32**
ELEPHANT [BCDM21]	12	33	7264.87	**55.84**	**130.10**

* indicates that one ancilla qubit is introduced.

4.3 New Results for 5-bit S-box Quantum Circuits

More importantly, our model can be used to synthesize concrete quantum circuit implementations for 5-bit S-boxes. To the best of our knowledge, this is the first general SMT-based framework that can automatically obtain concrete NCT implementations for practical 5-bit S-boxes such as KECCAK and ASCON.

Table 9: Lower bounds and NCT circuit implementations for the KECCAK 5-bit S-box

Source	Width	#Gate	Full depth	#Pauli-X	#CNOT	#Toffoli	Search
Lin et al. [LYX+26]	5	> 9	–	–	–	–	–
This work	5	> 12	–	–	–	–	Strategy 1
Huang et al. [HZL25]	5	19	52	12	0	7	–
This work	5	**17**	**51**	9	1	7	Strategy 2
	5	**16**	**52**	7	2	7	Strategy 2
	5	14	54	2	5	7	Strategy 1
	5	13	55	0	6	7	Strategy 1
This work	6	> 10	–	–	–	–	Strategy 1
	6	14	37	6	3	**5**	Strategy 1
	6	13	38	4	4	**5**	Strategy 1
	6	12	39	2	5	**5**	Strategy 1
	6	11	40	0	6	**5**	Strategy 1

KECCAK S-box. For the KECCAK 5-bit S-box, the results under different width settings are summarized in Table 9. In the no-ancilla setting (Width = 5), our exact-G searches prove that no implementation exists with at most 12 NCT gates, improving the lower bound of Lin et al. [LYX+26] from Gates > 9 to Gates > 12. We also obtain several concrete implementations. Compared with Huang et al.'s [HZL25] implementation using 19 NCT gates with full depth 52, our depth-first search finds a 17-gate implementation with smaller full depth 51, and a 16-gate implementation with the same full depth 52. In addition, the gate-count-first search finds smaller-gate implementations with 14 and 13 gates, with full depths 54 and 55, respectively.

When one ancilla qubit is introduced (Width = 6), our model further improves the depth and achieves the minimum Toffoli count. In particular, we obtain implementations with $14, 13, 12$, and 11 NCT gates, all using only 5 Toffoli gates. By Theorem 1 and the ANF representation of the KECCAK S-box, its homogeneous quadratic parts span a 5-dimensional space; hence any NCT implementation requires at least 5 Toffoli gates.

Therefore, these width-6 implementations achieve the minimum possible Toffoli count. The concrete circuits are given in Appendix A.

ASCON S-box. For the ASCON 5-bit S-box, the results under two width settings are summarized in Table 10. Without ancilla qubits (Width = 5), our search proves that the gate lower bound is > 12, which matches the bound reported by Lin et al. [LYX+26].

When one ancilla qubit is introduced (Width = 6), our search proves a gate lower bound of > 11. We further obtain an implementations with Toffoli count 5. By Theorem 1 and the ANF representation of the ASCON S-box, any NCT implementation requires at least 5 Toffoli gates. Therefore, this width-6 implementations achieve the minimum Toffoli count. The concrete circuits are given in Appendix B.

Table 10: Lower bounds and NCT implementations of the ASCON 5-bit S-box

Source	Width	#Gate	Full depth	#Pauli-X	#CNOT	#Toffoli	Search
Lin et al. [LYX+26]	5	> 12	–	–	–	–	–
This work	5	> 12	–	–	–	–	Strategy 1
	5	19	51	4	8	7	Strategy 2
	5	18	52	3	8	7	Strategy 2
	5	16	53	3	6	7	Strategy 1
This work	6	> 11	–	–	–	–	Strategy 1
	6	16	37	3	8	**5**	Strategy 2

4.4 Quantum Circuit Implementation of Linear Layers

The SMT modeling framework proposed in this paper can be generalized to the quantum circuit optimization of linear layers. We can simplify the original model: the gate operation type is fixed to CNOT, so the op_k variable can be removed; meanwhile, the second control wire is no longer needed, so the $c2_k$ variable can also be discarded. The revised SMT model only retains the target wire tgt_k and the control wire $c1_k$ at each step. This can be regarded as a specialization of the proposed SMT encoding to the CNOT-only gate library.

To evaluate the effectiveness of this model, we apply it to the quantum circuit synthesis of the 16×16 linear matrices used in SKINNY [BJK+16] and MIDORI [BBI+15]. For the SKINNY linear matrix, our model obtains an implementation with 12 CNOT gates and depth 3. For the MIDORI linear matrix, our model obtains an implementation with 24 CNOT gates and depth 3. These results are consistent with the CNOT-count and depth results reported in [SF24]. The corresponding CNOT circuits are given in Appendix C. In addition, when no depth bound is imposed, our model proves that no implementation exists with exactly 11 CNOT gates for the SKINNY linear matrix.

For the SKINNY linear matrix, we further observe that several output bits are directly preserved from the input wires. Based on this observation, we introduce two pruning constraints specifically for the SKINNY instance. First, during the circuit construction, the input wires corresponding to these directly preserved outputs are forbidden from being selected as CNOT target wires. Second, in the final output matching, the directly preserved output bits are fixed to these original wires. These constraints eliminate redundant search branches in which a directly preserved wire is modified and later restored. The experimental results show that this pruning is effective for the SKINNY linear matrix: the solving time is reduced from 2429.5 seconds to 254.6 seconds.

5 Conclusion

In this paper, we designed two models for quantum circuit optimization: an exact-G model and an at-most-G model with null gates. Experimental results show that our model improves the solving efficiency for 4-bit S-boxes and enables automated synthesis of practical 5-bit S-box quantum circuits. In addition to the 5-bit KECCAK and ASCON S-boxes, we also obtain results for 6-bit S-box lower-bound analysis and linear layer implementation.

In future work, we plan to further improve the scalability of the proposed encoding for larger S-boxes. For 6-bit S-boxes, our current experiments can prove UNSAT only under small gate-count bounds, but they do not find complete implementations within a reasonable solving time. Moreover, even for the KECCAK 5-bit S-box, proving that no 12-gate NCT implementation exists took about two months. This observation motivates us to further study UNSAT instances.

References

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, Lecture Notes in Computer Science, pages 411–436. Springer, 2015. doi:10.1007/978-3-662-48800-3_17.

[BCDM21] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Multi-user security of the elephant v2 authenticated encryption mode. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, Lecture Notes in Computer Science, pages 155–178. Springer, 2021. doi:10.1007/978-3-030-99277-4_8.

[BDMW10] K. A. Browning, J. F. Dillon, M. T. McQuistan, and A. J. Wolfe. An APN permutation in dimension six. In *Finite Fields: Theory and Applications*, volume 518 of *Contemporary Mathematics*, pages 33–42, Providence, RI, 2010. American Mathematical Society.

[BDPA13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, Lecture Notes in Computer Science, pages 313–314. Springer, 2013. doi:10.1007/978-3-642-38348-9_19.

[BGLS19] Zhenzhen Bao, Jian Guo, San Ling, and Yu Sasaki. PEIGEN - a platform for evaluation, implementation, and generation of s-boxes. *IACR Trans. Symmetric Cryptol.*, 2019(1):330–394, 2019. URL: https://doi.org/10.13154/tosc.v2019.i1.330-394, doi:10.13154/TOSC.V2019.I1.330-394.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology -*

CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II, Lecture Notes in Computer Science, pages 123–153. Springer, 2016. doi:10.1007/978-3-662-53008-5_5.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *J. Cryptol.*, 26(2):280–312, 2013. URL: https://doi.org/10.1007/s00145-012-9124-7, doi:10.1007/S00145-012-9124-7.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, Lecture Notes in Computer Science, pages 321–345. Springer, 2017. doi:10.1007/978-3-319-66787-4_16.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, Lecture Notes in Computer Science, pages 2–21. Springer, 1990. doi:10.1007/3-540-38424-3_1.

[CBC23] Matthew Chun, Anubhab Baksi, and Anupam Chattopadhyay. DORCIS: depth optimized quantum implementation of substitution boxes. *IACR Cryptol. ePrint Arch.*, 2023:286, 2023. URL: https://eprint.iacr.org/2023/286.

[CLF+24] Jingwen Chen, Qun Liu, Yanhong Fan, Lixuan Wu, Boyun Li, and Meiqin Wang. New sat-based model for quantum circuit decision problem: Searching for low-cost quantum implementation. *IACR Commun. Cryptol.*, 1(1):31, 2024. URL: https://doi.org/10.62056/anmmp-4c2h, doi:10.62056/ANMMP-4C2H.

[DBSC19] Vishnu Asutosh Dasu, Anubhab Baksi, Sumanta Sarkar, and Anupam Chattopadhyay. LIGHTER-R: optimized reversible circuit implementation for sboxes. In *32nd IEEE International System-on-Chip Conference, SOCC 2019, Singapore, September 3-6, 2019*, pages 260–265. IEEE, 2019. doi:10.1109/SOCC46988.2019.1570548320.

[FWL+21] Yanhong Fan, Weijia Wang, Zhihu Li, Zhenyu Lu, Siu-Ming Yiu, and Meiqin Wang. Forced independent optimized implementation of 4-bit s-box. In Joonsang Baek and Sushmita Ruj, editors, *Information Security and Privacy - 26th Australasian Conference, ACISP 2021, Virtual Event, December 1-3, 2021, Proceedings*, Lecture Notes in Computer Science, pages 151–170. Springer, 2021. doi:10.1007/978-3-030-90567-5_8.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996. doi:10.1145/237814.237866.

[HZL25] Zhenyu Huang, Fuxin Zhang, and Dongdai Lin. Constructing quantum implementations with the minimal t-depth or minimal width and their applications. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology -*

EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part I, Lecture Notes in Computer Science, pages 155–185. Springer, 2025. doi:10.1007/978-3-031-91107-1_6.

[JCL+25] Chenhao Jia, Tingting Cui, Qing Ling, Yan He, Kai Hu, Yu Sun, and Meiqin Wang. How small can s-boxes be? *IACR Trans. Symmetric Cryptol.*, 2025(1):592–622, 2025. URL: https://doi.org/10.46586/tosc.v2025.i1.592-622, doi:10.46586/TOSC.V2025.I1.592-622.

[JPST17] Jérémy Jean, Thomas Peyrin, Siang Meng Sim, and Jade Tourteaux. Optimizing implementations of lightweight building blocks. *IACR Trans. Symmetric Cryptol.*, 2017(4):130–168, 2017. URL: https://doi.org/10.13154/tosc.v2017.i4.130-168, doi:10.13154/TOSC.V2017.I4.130-168.

[KLL+14] Erdem B. Kavun, Martin M. Lauridsen, Gregor Leander, Christian Rechberger, Peter Schwabe, and Tolga Yalçın. Prøst v1.1. CAESAR Competition Round 1 Submission, 2014. URL: http://competitions.cr.yp.to/round1/proestv11.pdf.

[LYX+26] Da Lin, Chunli Yang, Shengyuan Xu, Shizhu Tian, and Bing Sun. On the construction of quantum circuits for s-boxes with different criteria based on the SAT solver. *Quantum Inf. Process.*, 25(2):39, 2026. URL: https://doi.org/10.1007/s11128-025-05049-1, doi:10.1007/S11128-025-05049-1.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, Lecture Notes in Computer Science, pages 386–397. Springer, 1993. doi:10.1007/3-540-48285-7_33.

[Sel13] Peter Selinger. Quantum circuits of t-depth one. *Phys. Rev. A*, 87:042302, Apr 2013. URL: https://link.aps.org/doi/10.1103/PhysRevA.87.042302, doi:10.1103/PhysRevA.87.042302.

[SF24] Haotian Shi and Xiutao Feng. Quantum circuits of AES with a low-depth linear layer and a new structure. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part VIII*, Lecture Notes in Computer Science, pages 358–395. Springer, 2024. doi:10.1007/978-981-96-0944-4_12.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. doi:10.1137/S0097539795293172.

[SOT+21] Layth Sliman, Tasnime Omrani, Zahir Tari, Abed Ellatif Samhat, and Rhouma Rhouma. Towards an ultra lightweight block ciphers for internet of things. *J. Inf. Secur. Appl.*, 61:102897, 2021. URL: https://doi.org/10.1016/j.jisa.2021.102897, doi:10.1016/J.JISA.2021.102897.

[Sto16] Ko Stoffelen. Optimizing s-box implementations for several criteria using SAT solvers. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, pages 140–160. Springer, 2016. doi:10.1007/978-3-662-52993-5_8.

[WL21] Qinglin Wang and Jiqiang Lu. Fault analysis of the ARIA and ublock block ciphers. In Ian McLoughlin, editor, *2021 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), Singapore, December 11-12, 2021*, pages 1–6. IEEE, 2021. doi:10.1109/SOLI54607.2021.9672378.

[WSL+25] Zihao Wei, Siwei Sun, Fengmei Liu, Lei Hu, and Zhiyu Zhang. Technology-dependent synthesis and optimization of circuits for small s-boxes. Cryptology ePrint Archive, Paper 2025/103, 2025. URL: https://eprint.iacr.org/2025/103, doi:10.62056/akmpdkp10.

[WZ11] Wenling Wu and Lei Zhang. Lblock: A lightweight block cipher. In Javier López and Gene Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, Lecture Notes in Computer Science, pages 327–344, 2011. doi:10.1007/978-3-642-21554-4_19.

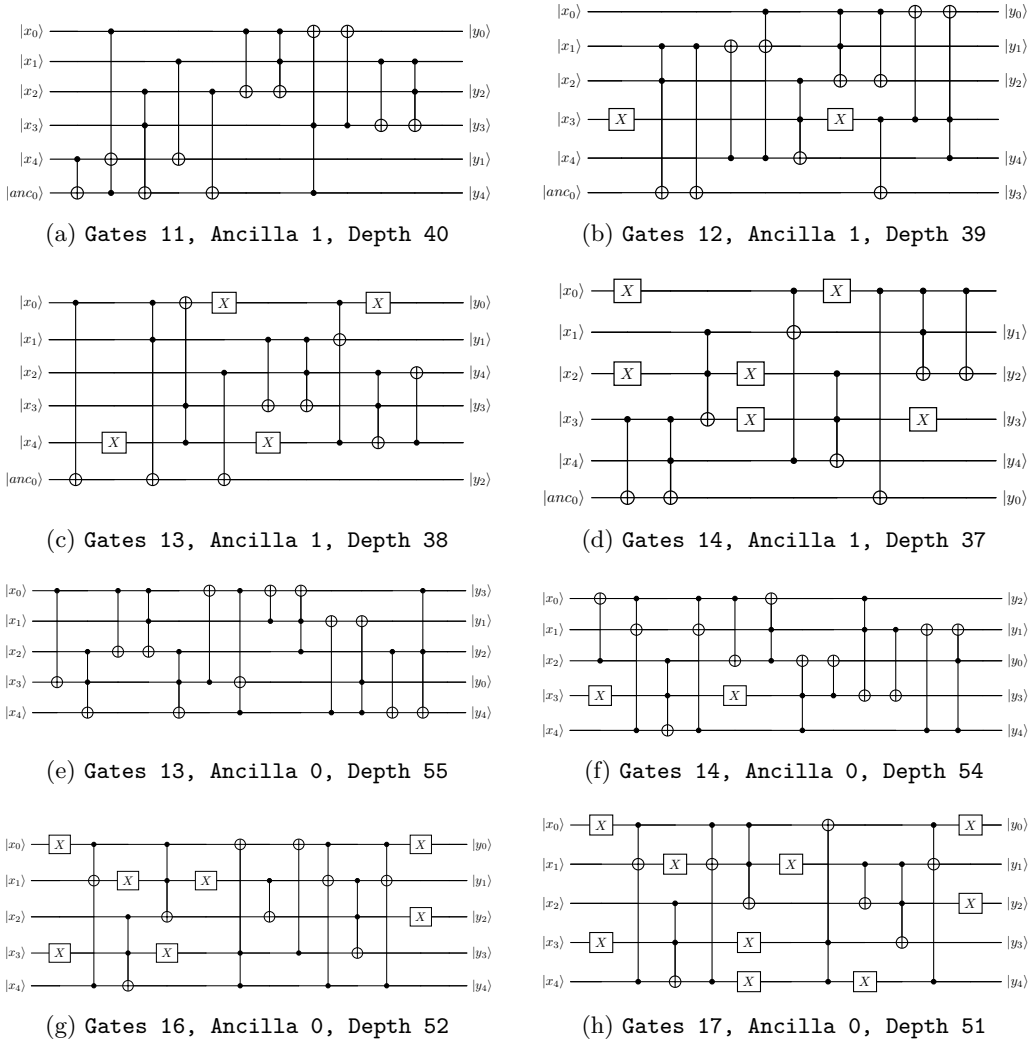
[ZBL+15] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Sci. China Inf. Sci.*, 58(12):1–15, 2015. URL: https://doi.org/10.1007/s11432-015-5459-7, doi:10.1007/S11432-015-5459-7.

A KECCAK S-box Circuits

The S-box used in KECCAK is shown as follows:

$$S_{\text{KECCAK}}(x) = [0,\ 9,\ 18,\ 11,\ 5,\ 12,\ 22,\ 15,\ 10,\ 3,\ 24,\ 1,\ 13,\ 4,\ 30,\ 7,\\ 20,\ 21,\ 6,\ 23,\ 17,\ 16,\ 2,\ 19,\ 26,\ 27,\ 8,\ 25,\ 29,\ 28,\ 14,\ 31].$$

The KECCAK S-box circuits are shown in Figure 2.



(a) `Gates 11, Ancilla 1, Depth 40`

(b) `Gates 12, Ancilla 1, Depth 39`

(c) `Gates 13, Ancilla 1, Depth 38`

(d) `Gates 14, Ancilla 1, Depth 37`

(e) `Gates 13, Ancilla 0, Depth 55`

(f) `Gates 14, Ancilla 0, Depth 54`

(g) `Gates 16, Ancilla 0, Depth 52`

(h) `Gates 17, Ancilla 0, Depth 51`

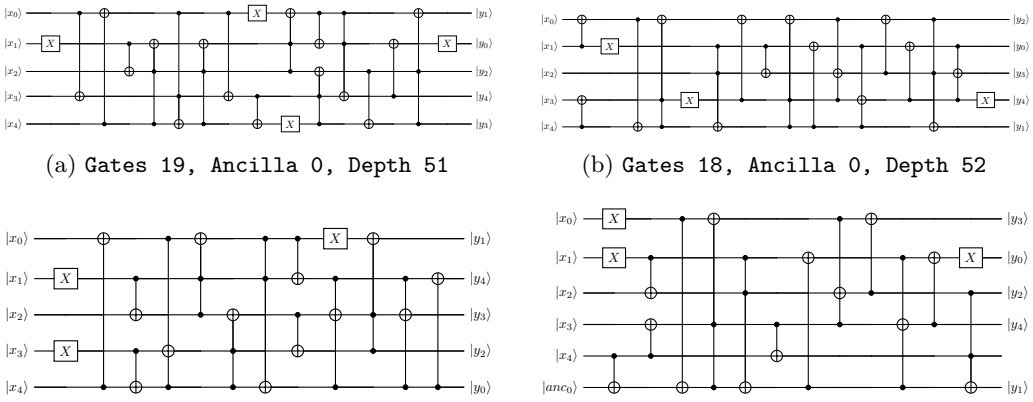
Figure 2: Quantum circuit implementations of the KECCAK 5-bit S-box.

B ASCON S-box Circuits

The S-box used in ASCON is shown as follows:

$$S_{\text{ASCON}}(x) = [4,\ 11,\ 31,\ 20,\ 26,\ 21,\ 9,\ 2,\ 27,\ 5,\ 8,\ 18,\ 29,\ 3,\ 6,\ 28,\\ 30,\ 19,\ 7,\ 14,\ 0,\ 13,\ 17,\ 24,\ 16,\ 12,\ 1,\ 25,\ 22,\ 10,\ 15,\ 23].$$

The ASCON S-box circuits are shown in Figure 3.



(a) `Gates 19, Ancilla 0, Depth 51`

(b) `Gates 18, Ancilla 0, Depth 52`

(c) `Gates 16, Ancilla 0, Depth 53`

(d) `Gates 16, Ancilla 1, Depth 37`

Figure 3: Quantum circuit implementations of the ASCON 5-bit S-box.

C CNOT Circuit for the SKINNY and MIDORI Linear Layers

Table 11: The implementation of the MIDORI linear layer with depth 3 and 24 CNOTs.

Depth	Operation	
1	$x[7] = x[7] + x[3]$	
1	$x[1] = x[1] + x[9]$	
1	$x[5] = x[5] + x[13]$	
1	$x[10] = x[10] + x[2]$	
1	$x[6] = x[6] + x[14]$	
1	$x[11] = x[11] + x[15]$	
1	$x[4] = x[4] + x[12]$	
1	$x[0] = x[0] + x[8]$	
2	$x[13] = x[13] + x[1]$	$y[5]$
2	$x[9] = x[9] + x[5]$	$y[1]$
2	$x[2] = x[2] + x[6]$	$y[10]$
2	$x[14] = x[14] + x[10]$	$y[6]$
2	$x[3] = x[3] + x[11]$	$y[7]$
2	$x[15] = x[15] + x[7]$	$y[11]$
2	$x[8] = x[8] + x[4]$	$y[0]$
2	$x[12] = x[12] + x[0]$	$y[4]$
3	$x[11] = x[11] + x[15]$	$y[15]$
3	$x[6] = x[6] + x[14]$	$y[14]$
3	$x[7] = x[7] + x[3]$	$y[3]$
3	$x[1] = x[1] + x[9]$	$y[9]$
3	$x[5] = x[5] + x[13]$	$y[13]$
3	$x[10] = x[10] + x[2]$	$y[2]$
3	$x[4] = x[4] + x[12]$	$y[12]$
3	$x[0] = x[0] + x[8]$	$y[8]$

Table 12: The implementation of the SKINNY linear layer with depth 3 and 12 CNOTs.

Depth	Operation	
1	$x[4] = x[4] + x[8]$	$y[8]$
1	$x[5] = x[5] + x[9]$	$y[9]$
1	$x[6] = x[6] + x[10]$	$y[10]$
1	$x[7] = x[7] + x[11]$	$y[11]$
2	$x[8] = x[8] + x[0]$	$y[12]$
2	$x[9] = x[9] + x[1]$	$y[13]$
2	$x[10] = x[10] + x[2]$	$y[14]$
2	$x[11] = x[11] + x[3]$	$y[15]$
3	$x[12] = x[12] + x[8]$	$y[0]$
3	$x[13] = x[13] + x[9]$	$y[1]$
3	$x[14] = x[14] + x[10]$	$y[2]$
3	$x[15] = x[15] + x[11]$	$y[3]$
$y[4] = x[0],\ y[5] = x[1],\ y[6] = x[2],\ y[7] = x[3]$ are directly preserved.		