

Quantum Implementation and Analysis of SHA-2 and SHA-3

Kyungbae Jang¹, Sejin Lim¹, Yujin Oh¹, Hyunjun Kim¹,
Anubhab Baksi^{2*}, Sumanta Chakraborty³, and Hwajeong Seo¹

¹ Hansung University, Seoul, South Korea

² Nanyang Technological University, Singapore

³ Techno International New Town, West Bengal, India

starj1023@gmail.com, dlatpwl834@gmail.com, oyj0922@gmail.com, khj930704@gmail.com,
anubhab.baksi@ntu.edu.sg, sumanta.phd21@iiitkalyani.ac.in, hwajeong84@gmail.com

Abstract. Quantum computers have the potential to solve hard problems that are nearly impossible to solve by classical computers, this has sparked a surge of research to apply quantum technology and algorithm against the cryptographic systems to evaluate for its quantum resistance. In the process of selecting post-quantum standards, NIST categorizes security levels based on the complexity that quantum computers would require to crack AES encryption (levels 1, 3, and 5) and SHA-2 or SHA-3 (levels 2 and 4). In assessing the security strength of cryptographic algorithms against quantum threats, accurate predictions of quantum resources are crucial. Following the work of Jaques et al. in Eurocrypt 2020, NIST estimated security levels 1, 3, and 5, corresponding to quantum circuit size for finding the key for AES-128, AES-192, and AES-256, respectively. This work has been recently followed-up by Huang et al. (Asiacrypt'22) and Liu et al. (Asiacrypt'23). However, for levels 2 and 4, which relate to the collision finding for the SHA-2 and SHA-3 hash functions, quantum attack complexities are probably not well-studied.

In this paper, we present novel techniques for optimizing the quantum circuit implementations for SHA-2 and SHA-3 algorithms in all the categories specified by NIST. After that, we evaluate the quantum circuits of target cryptographic hash functions for quantum collision search. Finally, we define the optimal quantum attack complexity for levels 2 and 4, and comment on the security strength of the extended level. We present new concepts to optimize the quantum circuits at the component level and the architecture level.

Keywords: Quantum Circuit · Quantum Collision Search · SHA-2 · SHA-3 · NIST Post Quantum Cryptography · Quantum Security Levels

1 Introduction

With the progress in quantum computing, it has become essential to evaluate the vulnerability of cryptographic algorithms against it. Since quantum computers can efficiently solve certain hard problems in commonly used cryptographic algorithms, concerns arise about the compromised security of traditional cryptography. Shor's algorithm [1] is particularly noteworthy for its ability to compromise traditional public-key cryptography, such as RSA and ECC. In case of symmetric key, like AES, SHA-2, and SHA-3, the primary threat is Grover's search algorithm [2], which offers a quadratic speedup, significantly reducing the time needed for an exhaustive search.

This changing landscape requires a thorough reassessment of cryptographic protocols and exploration of quantum-resistant alternatives to ensure enough quantum security strength in the future. As widely known, the National Institute of Standards and Technology (NIST) is conducting a competition in Post-Quantum Cryptography (PQC) to standardize algorithms that are secure against potential quantum attacks. As part of this contest, NIST introduced security strength categories to classify the post-quantum cryptography candidates [3], as shown in Table 1¹. These classifications were determined by considering the intricacies of quantum attacks, measured in terms of quantum circuit size. In addition to the gate count, MAXDEPTH is another parameter proposed by NIST. It involves the concept of limiting quantum attacks by setting a maximum runtime or quantum circuit depth. This approach is motivated by the challenges associated with executing excessively long sequential computations.

* This project is partially supported by the Wallenberg – NTU Presidential Post-doctoral Fellowship.

¹ The definition of the quantum security levels as per [3] is somewhat ambiguous, as [3, Page 16] defines up to level 5, but [3, Page 17] defines up to the extended level.

Table 1: Security levels defined by NIST (in context of post-quantum cryptography competition) along with our extension.

Category	Cipher	Quantum gate count
Level 1	AES-128	$2^{157}/\text{MAXDEPTH}$
Level 2	SHA-2-256/SHA-3-256	Unspecified
Level 3	AES-192	$2^{221}/\text{MAXDEPTH}$
Level 4	SHA-2-384/SHA-3-384	Unspecified
Level 5	AES-256	$2^{285}/\text{MAXDEPTH}$
Level 6 (Extension)	SHA-2-512/SHA-3-512	Not applicable

NIST defines post-quantum security levels [3, 4] ranging from 1 to 5. Levels 1, 3, and 5 correspond to the quantum circuit size of finding a solution key for AES-128, AES-192, and AES-256 using the Grover algorithm, respectively. However, for levels 2 and 4, which relate to the collision finding (i.e., same output with different inputs) for the SHA-2 and SHA-3 hash functions, quantum circuit sizes are yet to be defined (only classical circuit sizes are reported). Inspired by this, we optimize quantum circuits for representative two cryptographic hash functions, including SHA-2 and SHA-3. The anticipated outcome from our work is to provide more efficient quantum circuits for cryptographic hash functions compared to previous works [5, 6, 7, 8, 9], while simultaneously offering the lowest complexity that can be designated as the quantum post-quantum security level by NIST (discussed in detail in Section 4).

1.1 Contributions

To the best of our knowledge, we integrated all the best techniques for optimizing the quantum circuit depth of quantum circuit implementations for SHA-2 and SHA-3 algorithms in all categories. And then, we benchmarked against the state-of-art quantum circuit implementations for two representative cryptographic hash functions (i.e. SHA-2 and SHA-3). For SHA-2 and SHA-3 algorithms, our implementations include novel tricks not yet described in the literature. In all quantum circuit implementations, we achieve the lowest circuit depth and the best performance for Grover’s algorithm. The detailed contributions are as follows:

- *Quantum Carry-Save Adder based quantum circuit implementations of SHA-2:* We present optimized quantum circuits by using the *Quantum Carry-Save Adder* (QCSA) [10] for multi-operand addition, a resource-critical step of SHA-2. Concretely, we achieve a critical path of around 1 for quantum additions in a round. This represents a significant improvement compared to the previous best result, achieved in Lee et al.’s work [5], which was 3. We also introduce efficient implementation techniques; reusing output qubits, and optimization for a fixed input length, based on the out-of-place approach.
- *Improved out-of-place and interval architecture based quantum circuit implementations of SHA-3:* We introduce novel out-of-place round architecture. Based on the improved architecture, novel techniques are incorporated; all-in-one, parallel design with copying, and trick of the X gate operation. We also develop a novel architecture that can reuse/reduce many of these ancilla qubits without increasing circuit depth, called the *interval* architecture. With the interval architecture, our quantum circuits of SHA-3 offer the lowest depths with the best trade-off performance.
- *Defining the NIST quantum attack complexity for levels 2 and 4* In a realistic approach to Grover’s search, the full circuit should be split into reduced sizes and operated in parallel to address the issue of extreme circuit depth. In this parallelization process, our depth-optimized implementations of SHA-2 and SHA-3 offer a benefit in achieving the best trade-off performance compared to qubit-optimized implementations (related discussion is given in Appendix A). Based on the estimated cost for quantum collision search using the presented SHA-2 and SHA-3 quantum circuits, we define the quantum complexity for levels 2 and 4, which have not yet been defined.

Table 2: Security levels defined in this work.

Strength	Category	Hash function	Quantum gate count
Level 2	A	SHA-2-256	$2^{188}/\text{MAXDEPTH}$
	B	SHA-3-256	$2^{183}/\text{MAXDEPTH}$
Level 4	A	SHA-2-384	$2^{266}/\text{MAXDEPTH}$
	B	SHA-3-384	$2^{260}/\text{MAXDEPTH}$
Level 6 (Extension)	A	SHA-2-512	$2^{343}/\text{MAXDEPTH}$
	B	SHA-3-512	$2^{337}/\text{MAXDEPTH}$

- *Extending the NIST quantum attack complexity for the extended level:* We also optimized the SHA-2-512 and SHA-3-512 in quantum circuits. This specified the quantum complexity for the extended level.

The security levels depending on the hash functions are mentioned in [3]. Following this, we using the naming convention for the security levels as shown in Table 2.

1.2 Related Works

In the quantum implementations, the evaluation metrics are similar to the hardware based cryptography implementations but the point of view is a bit different. In order to evaluate the performance of quantum circuits properly, the following metrics are commonly defined in related research [6, 11, 12, 13]

- *Time Complexity:* Quantum circuits can be designed with Clifford + T gates (Section 2). The time complexity can be measured by the full depth of the quantum circuit, representing the critical path of target quantum circuits. For fault-tolerant quantum computing, the T -depth, counted by non-parallelizable T gates, also signifies the time complexity of quantum circuits.
- *Space Complexity* The required number of qubits of quantum circuits (often denoted as width) corresponds to the space complexity.
- *Time-Space Complexity:* The time-space complexity is the product of the depth and width of the quantum circuit. This metric is often employed to evaluate the trade-off performance of the quantum circuit.
- *Quantum Attack Complexity:* The quantum attack complexity is determined by the product of the total number of gates and the depth of the quantum circuit. This metric was introduced by NIST [3] to establish boundaries for post-quantum levels 1 to 5 of post-quantum cryptography candidates (refer to Section 4 and Appendix A for details).

Quantum circuits for SHA-2 and SHA-3. Quantum circuit implementations of SHA-2 were initially introduced by Amy et al. in [7]. SHA-2 quantum circuits in [6, 7] have a high circuit depth due to the sequential execution of major functions, such as Ch , Maj , and addition operations. In [5], Lee et al. subsequently presented enhanced quantum circuits that achieve the low Toffoli depth with a reasonable number of qubits.

In [7], the authors introduced a round in-place architecture for the SHA-3 quantum circuit to reduce the number of qubits. In this approach, each function should be designed to be reversible (an intuitive view is in Figure 6). Song et al. [14] presented a new circuit for SHA-3 to reduce T -depth by excluding reversible functions and allocating extra qubits.

Häner et al. [9] and Meuli et al. [8] presented algorithms for optimizing Toffoli-related metrics such as T gates, Toffoli gates, and T -depth, in quantum implementation. They did not provide specific quantum circuits for SHA-2 and SHA-3; instead, they estimated the required quantum resources using their algorithms.

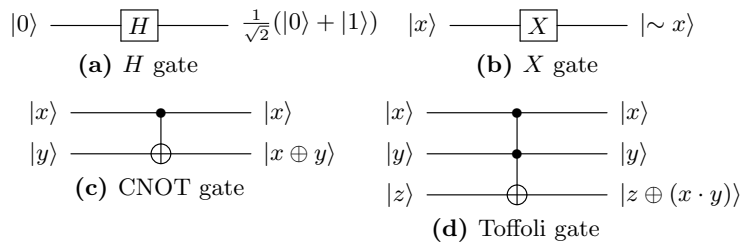


Figure 1: Quantum gates employed in our quantum circuit implementations.

2 Quantum Gates

Before describing our quantum circuits for cryptographic hash functions, the quantum gates (i.e., at the bottom level) employed for our implementation are briefly outlined in this section. Figure 1 shows the diagrams of the quantum gates utilized in this paper to design the quantum circuits (notations of \sim , \oplus , and \cdot represent NOT, XOR, and AND operations, respectively):

The H (Hadamard) gate is a fundamental quantum gate in quantum computing that creates superposition by equally weighting the $|0\rangle$ and $|1\rangle$ states. The X gate (Figure 1(b)) inverts the state of the qubit from 0 to 1 or from 1 to 0 (similar to the classical NOT operation (\sim)). The CNOT gate (Figure 1(c)) inverts the state of the target qubit (y) if the control qubit (x) is 1. Thus, this quantum gate can replace the classical XOR operation (\oplus). Figure 1(d), the Toffoli gate, inverts the state of the target qubit (z) if both control qubits (x and y) are 1. This is the most important quantum gate in this paper (and the same holds true for other papers [11, 12, 15, 16]) for optimizing quantum circuits. There are two reasons why this quantum gate is major for optimization.

Actually, the Toffoli gate is implemented by combining multiple quantum gates, such as X , CNOT, T , and H gates. Figure 1(d) is an intuitive diagram for understanding. There are many proposals to efficiently design the Toffoli gate (which consumes more quantum resources) [17, 18, 19, 20]. In this work, we decompose a single Toffoli gate using 6 CNOT gates + 2 H gates + 7 T gates, with a total depth of 8 (T -depth is 4), following one of the methods described in [17].

3 Grover's Algorithm

The steps of Grover's search can be divided into three stages: *Setting Input*, *Oracle*, and *Diffusion Operator*. We describe Grover's algorithm by applying it to the preimage search for hash functions.

1. *Setting Input*: Hadamard gates are applied to an n -qubit input to prepare a superposition state $|\psi\rangle$, resulting in equal probabilities for all 2^n inputs:

$$H^{\otimes n} |0\rangle^{\otimes n} = |\psi\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \quad (1)$$

2. In the *Oracle*, the hash function is implemented as a quantum circuit that generates the hash output using the previous input in a superposition state. The generated hash output (in a superposition state) is then compared with the known hash output (this part is usually ignored in resource estimation [11, 12, 21]), and if a match is found (i.e., if $f(x) = 1$) and the sign of the solution input (i.e., preimage) is negated (i.e., if $f(x) = -1$):

$$f(x) = \begin{cases} 1 & \text{if Hash}(x) = \text{target output} \\ 0 & \text{if Hash}(x) \neq \text{target output} \end{cases} \quad (2)$$

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle|-\rangle \quad (3)$$

3. The *Diffusion Operator* is designed to amplify the amplitude of the preimage marked by the Oracle. Due to its minimal complexity in comparison to the Oracle, the diffusion operator is usually disregarded in estimating Grover's algorithm [11, 21].

Application to Grover's Algorithm to Collision Search

For applying the Grover algorithm to the collision finding of hash functions, we investigate three approaches in this section:

Second preimage attack. In the second preimage attack, a collision search is performed by finding a second preimage for a given first preimage. Grover's algorithm searches for the second preimage that satisfies the output of the first preimage by excluding the first preimage from the input set. This method is similar to a quantum key search with the same complexity of $O(2^{n/2})$.

The second preimage attack has the advantage of not requiring quantum memory, but it may not align with NIST's considerations. NIST defines the post-quantum strength of SHA-2/3-256 as level 2, assuming that a quantum collision attack on SHA-2/3-256 is more feasible than a quantum key search on AES-192 (i.e., level 3). If we estimate the complexity of the collision search on SHA-2/3-256 using the second preimage attack it is higher than the key search on AES-192.

BHT algorithm. The Brassard, Hoyer, and Tapp (BHT) algorithm [22] combines the classical speedup from the birthday paradox with the quantum speedup from Grover's algorithm, as described in Algorithm 1.

Algorithm 1: BHT algorithm for collision search.

Input: Input set 2^n

Output: *Collision*

- 1: Select a subset K (size of $2^{n/3}$) $\in 2^n$ at random and query the hash function
 - 2: **if** there is a *Collision* in K **then**
 - 3: **return** the *Collision*
 - 4: **else**
 - 5: Construct a subset L (size of $2^{2n/3}$) $\in 2^n$ that does not include K
 - 6: **end if**
 - 7: Grover's algorithm finds $x_1 \in L$ that collides with $x_0 \in K$
 - 8: **return** (x_0, x_1)
-

In the BHT algorithm, a subset K of size $O(2^{n/3})$ is randomly selected from the set of all possible inputs (2^n represents the input size). According to the birthday paradox, the likelihood of collisions within this subset increases, allowing for a faster classical search for collisions. After selecting subset K and checking for collisions classically, a subset L of size $O(2^{2n/3})$ is constructed, excluding the elements in K . Grover's algorithm is then applied to find a collision between an element $x_0 \in K$ and an element $x_1 \in L$. Classical search could find a collision with $O(2^{n/3})$ queries, and Grover's algorithm can find a collision with $O(\sqrt{\frac{2^n}{2^{n/3}}}) = O(2^{n/3})$ extra queries to the hash function.

However, this algorithm requires a significantly large amount of quantum memory, $O(2^{n/3})$. Additionally, in [23], Bernstein discussed the impracticality of the BHT algorithm for collision search.

CNS algorithm. In Asiacrypt’17, Chailloux, Naya-Plasencia, and Schrottenloher proposed a new quantum algorithm for collision search [24], known as the CNS algorithm, with a query complexity of $O(2^{2n/5})$ using $O(2^{n/5})$ classical memory. Although the complexity of the CNS algorithm is higher than that of the BHT algorithm, it does not require quantum memory, making it practical. Thus, we adopt the CNS algorithm for estimating the cost of quantum attacks on SHA-2 and SHA-3.

The CNS algorithm employs the *Quantum Amplitude Amplification* (QAA) algorithm [25], which is a generalized version of Grover’s algorithm and consists of two phases: constructing the list and finding collisions using the QAA algorithm. We provide a brief description related to the complexity analysis as follows (for further details, refer to [24]).

S_H^d denotes the set comprising pairs of input/output $(x, H(x))$, where $H(x)$ starts with d zeros. In the first phase, a list L of size 2^{t-d} is constructed from S_H^d using Grover’s algorithm with complexity $2^{d/2}$ (i.e., square root). Thus, the entire list L can be constructed with complexity $2^{t-d/2}$. The optimized parameters for t and d are $t = \frac{3n}{5}$ and $d = \frac{2n}{5}$ according to [24]. In the next phase, the QAA algorithm performs $2^{d/2}$ Grover iterations and 2^{t-d} operations to access the list L . By executing the QAA algorithm $2^{(n-t-1)/2}$ times, the total complexity is given by $2^{(n-t-1)/2}(2^{d/2} + 2^{t-d}) + 2^{t-2/d}$. Finally, with optimized parameters, a collision can be found in $O(2^{2n/5})$ with $O(2^{n/5})$ classical memory. The authors of the CNS algorithm also presented a parallelization method that can reduce the search complexity. For parallelization with 2^s quantum instances, the search complexity exponent of the CNS algorithm for collision search is reduced to $\frac{2n}{5} - \frac{3s}{5}$ ($s \leq \frac{n}{4}$).

In this paper, to establish appropriate boundaries for NIST post-quantum security levels, we assume $s = n/6$ and a large size of classical memory. For SHA-2/3- n , the search complexities for $n = 256, 384,$ and 512 are (approximately) calculated as 2^{76} (level 2), 2^{115} (level 4), and 2^{153} (the extended level), respectively, using the formula $\frac{2n}{5} - \frac{3s}{5}$. These search complexities are appropriate, considering that the required search complexities for levels 1, 3, and 5 (Grover’s key search on AES-128, 192, and 256) are approximately $2^{64}, 2^{96},$ and 2^{128} , respectively. More discussion can be found in Section 9.1.

It is important to note that operating $2^{n/6}$ instances in parallel requires a significant number of qubits. The parallelization with $2^{(n/6)}$ instances is intended to provide suitable boundaries for levels 2, 4, and the extended level, and therefore can be adjusted.

4 NIST Post-Quantum Security Strength

NIST proposed the following approach in [3, 4] to address uncertainties in estimating the post-quantum security strengths of post-quantum cryptography candidates:

- *Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on AES and collision search on SHA-2/3.*

Each category is defined by a relatively easy-to-analyze reference primitive, serving as a baseline for various metrics that NIST considers potentially relevant to practical security. Specifically, NIST defined a separate category for each of the following security requirements as shown in Table 1. Estimated quantum resources for each category are the result of considering the quantum gate count, full depth, and MAXDEPTH. We present the related discussion in Appendix A.

Levels 1, 3, and 5 correspond to key search on AES, while levels 2, 4, and the extended level correspond to collision search on SHA-2/3. In a previous NIST document [4], the estimated resources for Grover’s key search on AES by Grassl et al. [21] were defined for levels 1, 3, and 5. Recently, NIST adjusted levels 1, 3, and 5 as Jaques et al. [11], which reduced the required quantum resources by improving the quantum circuits of AES.

It is important to note that quantum resources for levels 2, 4, and the extended level have not been defined yet. In our view, this is due to the lack of a clear estimation for collision search on SHA-2/3, unlike what the authors did in [11, 12, 21, 26]. Furthermore, there is not an optimized quantum circuit that satisfies the estimation method for quantum attacks (adopted by NIST). Generally, previous

works have focused on reducing the qubit count, which is not a recommended approach for Grover’s algorithm.

Inspired by this, we present optimized quantum circuits for cryptographic hash functions SHA-2 and SHA-3 and the clear estimation of collision search (refer to Section 3). Based on the quantum circuits of SHA-2 and SHA-3, new boundaries for levels 2, 4, and the extended level are suggested in this work.

5 Depth Optimization for Quantum Circuit Implementation

The design philosophy for our quantum circuit implementation focuses on minimizing quantum circuit depth rather than reducing the qubit count. However, it is essential to note that qubit count remains one of the most critical resources in quantum computers. In this context, we prioritize circuit depth reduction but carefully also consider the trade-off between qubit count and quantum circuit depth. We adhere to the following principles for quantum circuit implementations:

1. *If the quantum circuit depth can be effectively reduced by allocating additional ancilla qubits, it is permitted.*
2. *We reverse the quantum operations to reuse the ancilla qubits, if it is possible and there is no depth overhead for the reverse process.*

Keeping these two principles in mind, we design depth-optimized quantum implementations for cryptographic hash functions with a reasonable number of qubits. Our quantum circuits of SHA-2 and SHA-3 offer the least quantum circuit depth compared to the previous work [5, 6, 7].

In the application of Grover’s algorithm to cryptographic hash functions, it is more effective to reduce depth rather than the number of qubits (related discussion is in Appendix A). Demonstrating this, our results showcase the best performance in all trade-off metrics for SHA-2 and SHA-3 under Grover’s algorithm (strictly speaking, for parallel search). Finally, we prove that the quantum circuits for cryptographic hash functions outlined in this paper are effective under Grover’s algorithm, offering the lowest attack complexity, minimal depth, and the best trade-off metrics.

Applying AND gate. In recent quantum implementations, the use of AND gates (AND + AND[†]) is recommended to reduce T -depth, full depth, and gate count. We also provide the AND gate version of quantum circuits for three cryptographic hash functions. The AND gate operates the same as Toffoli gate, except that the target qubit must be in a clean state (i.e., $|0\rangle$). We adopt the recently updated AND gate implementation introduced in Jaques et al. [27]. The AND gate requires an ancilla qubit and consists of 11 Clifford + 4 T gates, with a T -depth of 1 and full depth of 8. The AND[†] gate is the reverse of the AND gate based on the Measurement gate. It consists of 5 Clifford + 1 Measurement gates, with a full depth of 4 (T -depth is 0). Due to the nature of the AND gate, where the target qubit must be in a clean state and the requirement of one ancilla qubit, additional considerations are included in our implementation.

6 Quantum Circuit Implementation of SHA-2

We describe the depth-optimized quantum circuits for the SHA-2 components here. For conciseness, we focus on describing our optimization method for SHA-2-256 algorithm. SHA-2-384 and SHA-2-512 share a similar inner structure with SHA-2-256, allowing our method to be equally applicable.

6.1 Implementation of Σ_0 , Σ_1 , σ_0 and σ_1

Four operations of linear layer are used in SHA-2 as follows (the notations ‘ \ggg ’ and ‘ \gg ’ respectively represent the right rotation and right shift operations):

$$\begin{aligned}
\Sigma_0(a) &= (a \ggg 2) \oplus (a \ggg 13) \oplus (a \ggg 22). \\
\Sigma_1(e) &= (e \ggg 6) \oplus (e \ggg 11) \oplus (e \ggg 25). \\
\sigma_0(W_{t-15}) &= (W_{t-15} \ggg 7) \oplus (W_{t-15} \ggg 18) \oplus (W_{t-15} \gg 3). \\
\sigma_1(W_{t-2}) &= (W_{t-2} \ggg 17) \oplus (W_{t-2} \ggg 19) \oplus (W_{t-2} \gg 10).
\end{aligned} \tag{4}$$

These operations consist of XOR operations, and each linear layer can be represented as a binary matrix. By applying PLU factorization to the binary matrix, we can obtain three factor matrices (namely, permutation, lower triangular and upper triangular). With those matrices, an in-place implementation of the linear layer can be achieved, as described in [28]. The 32×32 binary matrices associated with the SHA-2 linear layer are shown in Figure 2.

In [5], the authors presented in-place implementations of Σ_0 , Σ_1 , σ_0 , and σ_1 (i.e., without using any ancilla/output qubit), by employing PLU factorization². However, this approach leads to an increase in quantum circuit depth due to the operations of CNOT gates in limited space (i.e., within the input qubits). In [5], a maximum of 193 CNOT gates are operated within 32 qubits, resulting in a sequential flow between them (with a maximum depth of 55).

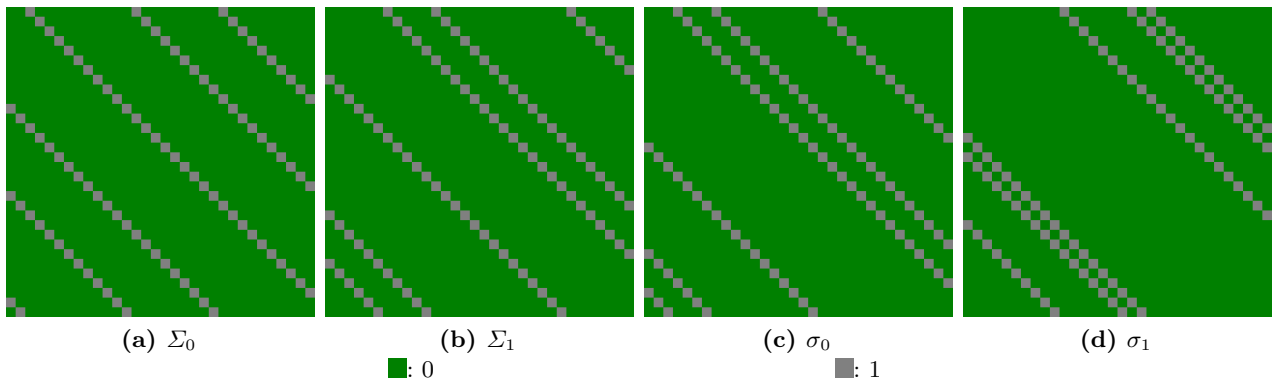


Figure 2: 32×32 binary matrices used in SHA-2 linear layer.

Out-of-place implementation. In contrast, we present out-of-place implementations of Σ_0 , Σ_1 , σ_0 , and σ_1 . We allocate 32 qubits for the *output* (our first principle in Section 5) and compute the result by XOR-ing the input qubits to the output qubits using CNOT gates. For example, for Σ_0 , we perform the following operations: CNOT ($a \ggg 2$, *output*), CNOT ($a \ggg 13$, *output*), CNOT ($a \ggg 22$, *output*). As a result, our out-of-place implementations of Σ_0 , Σ_1 , σ_0 , and σ_1 , only have a depth of 3, respectively, and require a maximum of 96 CNOT gates³. Since shift operations are included ($\gg 3$ and $\gg 10$), the required number of CNOT gates for σ_0 and σ_1 is reduced by 3 and 10, respectively.

Reusing output qubits. Later, the results of operations Σ_0 , Σ_1 , σ_0 , and σ_1 will be used as operands for additions (refer to Equations (6) and (7)). Once the additions are completed, these results are no longer needed. To reduce the number of qubits, we initialize the results to the clean state (i.e., $|0\rangle$) by performing the previous operations in reverse and then we reuse them (our second design methodology). Thanks to this approach, by incurring the initial allocation burden, subsequent linear

² This method has been used in other works too, see, e.g., [29].

³ In general, operations involving the swapping of qubits, such as shift, rotation, and rearrangement, are implemented logically without using quantum Swap gates.

Table 3: Quantum resources required for implementations of Σ_0 , Σ_1 , σ_0 and σ_1 .

Linear layer	Source	#CNOT	#Qubit (reuse)	Depth
Σ_0	PLU [5]	166	32	55
	Ours (Out-of-place)	96	64 (32)	3
Σ_1	Lee et al. (PLU) [5]	166	32	44
	Ours (Out-of-place)	96	64 (32)	3
σ_0	Lee et al. (PLU) [5]	193	32	50
	Ours (Out-of-place)	93	64 (32)	3
σ_1	Lee et al. (PLU) [5]	142	32	40
	Ours (Out-of-place)	86	64 (32)	3

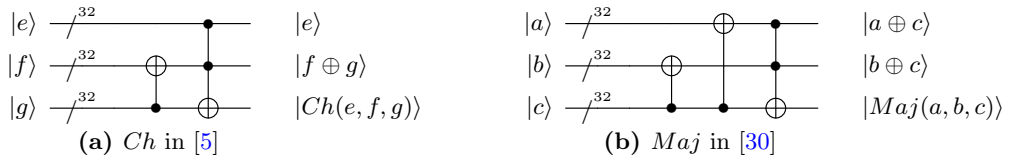
layers are implemented with low depth by reusing the initialized output qubits. Quantum resources required for implementations of Σ_0 , Σ_1 , σ_0 and σ_1 are shown in Table 3⁴.

6.2 Implementation of Ch and Maj

The operations of *Choose* (Ch) and *Majority* (Maj) are used in SHA-2 as follows:

$$\begin{aligned} Ch(e, f, g) &= (e \cdot f) \oplus (\sim e \cdot g). \\ Maj(a, b, c) &= (a \cdot b) \oplus (a \cdot c) \oplus (b \cdot c). \end{aligned} \quad (5)$$

We adopt the optimized circuits from [30] and [5], which utilize only one Toffoli gate. Similar to the operations in Section 6.1, the results of Ch and Maj operations are no longer needed after being used as operands in addition. Since the qubit values provided as inputs (i.e., a, b, c, e, f , and g) are used in the next round, we apply reverse operations (i.e., Ch^\dagger and Maj^\dagger) to initialize the results. The circuits for this and the quantum resources required for implementation are shown in Figure 3 and Table 4, respectively.

**Figure 3:** Quantum circuits for the operations Ch and Maj .**Table 4:** Quantum resources required for implementations of Ch and Maj .

Operation	#CNOT	#1qCliff	#T	Toffoli depth	#Qubit	Full depth
Ch	224	64	224	1	96	9
Maj	256	64	224	1	96	10

⁴ The required quantum resources are not provided in [7] and [6]. In [7], the description of the implementation technique is not specific, and in [6], single output qubits are shared for all Linear layer operations, which increases the depth and gate count.

6.3 Implementation of Multi-operands Addition

In the SHA-2 quantum circuit, the most resource-intensive operation is the repeated addition operations. The addition operations are carried out in both the round function and the message scheduling algorithm (Expressions (6) and (7), respectively):

$$\begin{aligned} h &= \Sigma_0(a) + Maj(a, b, c) + \Sigma_1(e) + Ch(e, f, g) + h + K_i + W_i. \\ d &= d + \Sigma_1(e) + Ch(e, f, g) + h + K_i + W_i. \end{aligned} \quad (6)$$

$$\begin{aligned} W_t &= M_t \quad (0 \leq t \leq 15) \\ &= \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} \quad (16 \leq t \leq 63). \end{aligned} \quad (7)$$

In [5], the authors introduced a quantum circuit featuring a critical path composed of three consecutive quantum adders. In their approach, they employed Draper's adder (in-place version) [31] and Takahashi's adder [32]. However, generic quantum adders designed to operate on two operands cannot handle operations with overlapping operands (more than two) in parallel. This limitation led the authors to conclude that a critical path of three was the best achievable.

Utilizing a Multi-Operand Adder Circuit. Our approach involves designing a multi-operand adder circuit to simultaneously handle all additions, thus achieving a critical path of around 1, instead of using the adder multiple times. We utilize [10]'s Quantum Carry-Save Adder (QCSA), which can handle multi-operand and provides low circuit depth. While QCSA requires a relatively large number of ancilla qubits, we can efficiently reuse most of these qubits by applying the reverse operation. Figure 4 illustrates how the QCSA from [10] is utilized and optimized for the SHA-2 algorithm.

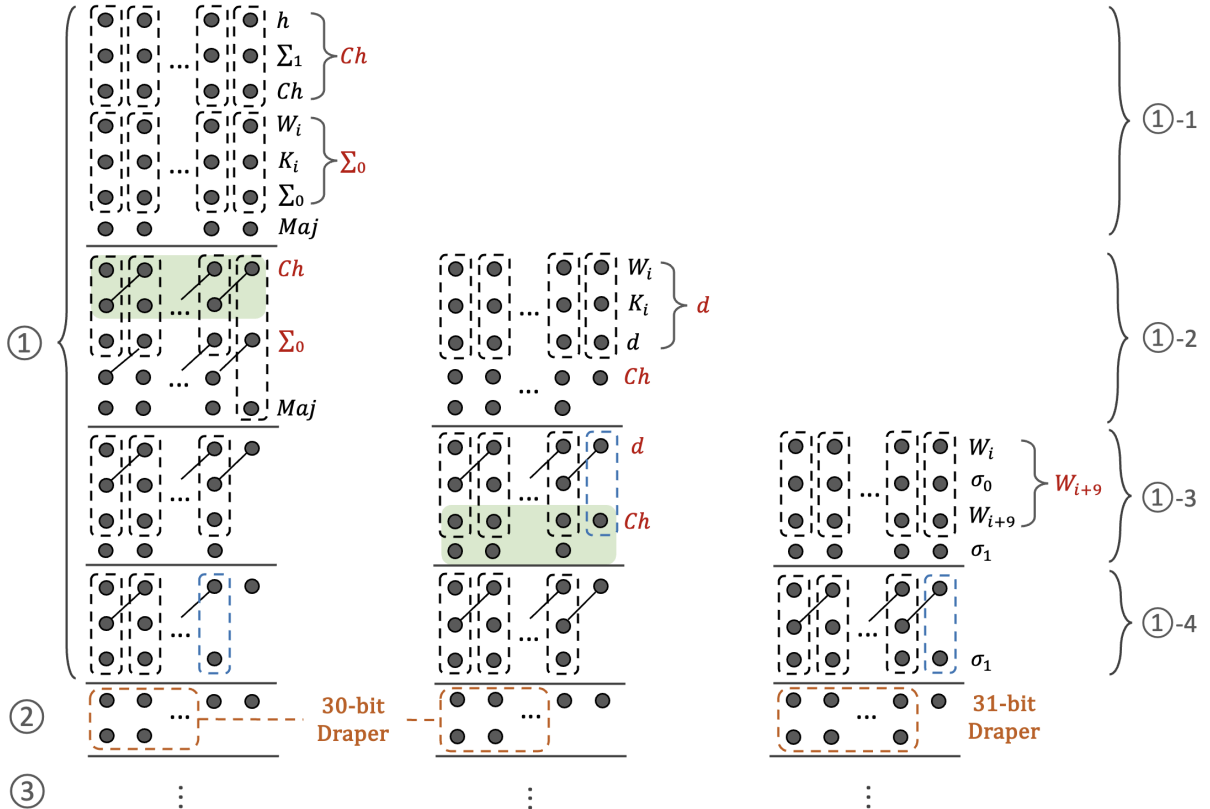


Figure 4: Optimized and modified structure of the QCSA with a critical path of around 1, ①: Partial sums; ②: Total sum; ③: Reverse.

The three QCSAs (aligned vertically) in Figure 4 are performed in parallel, with a critical path of around 1. From left to right, the operations corresponding to Expressions (6) (h , d) and (7) are performed by receiving 7, 6, and 4 operands of 32 bits, respectively. The following provides a step-by-step description of Figure 4:

- ① Partial sums: QCSA reduces the number of operands by using Quantum Full Adder (QFA, the black dashed rectangles for three operands, refer to Figure 4) and Quantum Half Adder (QHA, the blue dashed rectangles for two operands) to compute the partial sums of the operands until only two operands remain (the yellow dashed rectangles). In our implementation, four operations of partial sum are performed sequentially from ①-1 to ①-4. The operations included in each step are performed in parallel. The diagonally connected dots placed on the higher bits represent the carry bits generated from the QFA/QHA additions.
 - ①-1 : The addition operations for updating h in the round function are performed initially. In this step, five operands ($\Sigma_1, Ch, h, K_i, W_i$) overlap with those used for updating d (refer to Expression (6)). In order to optimize the architecture for the additions, we adjust the positions of the operands as follows.
Among the five overlapping operands, W_i and K_i are required for updating d in ①-2. Thus, the result of $W_i + K_i + \Sigma_0$ in ①-1 is stored in Σ_0 (marked in red), since the operands of W_i and K_i must retain their original value for the subsequent operation in ①-2. The result of $h + \Sigma_1 + Ch$ in ①-1 is stored in Ch . The updated Ch will be efficiently employed for the subsequent operation in ①-3.
 - ①-2 : As illustrated in the green part of Figure 4, the updated Ch and Σ_0 are used to calculate the result of $Ch + \Sigma_0 + Maj$. And the result of $W_i + K_i + d$ is stored in d . These two operations in ①-2 are performed in parallel.
 - ①-3 : The addition of $d + Ch$ is performed by using the updated Ch in ①-1. In this case, QHA (the blue dashed rectangle) is applied for the least significant bit as only two operands remain. At the same time, the addition of $W_i + \sigma_0 + W_{i+9}$ and the remaining part of $Ch + \Sigma_0 + Maj$ are performed in parallel.
 - ①-4 : The partial sum operations are performed until only two operands remain.
- ② Total sum: Finally, the two operands, which contain the cumulative sum of multiple operands, are added using Draper’s out-of-place adder.
- ③ Reverse: The operations performed in ① are reversed. This process restores the updated operands for subsequent rounds and initializes ancilla qubits for future use.

Table 5⁵ represents the quantum resources required for addition operations for the typical round. In our implementation, the Toffoli-depth is determined by the first QCSA in Figure 4. As a result, we achieve the lowest Toffoli-depth and Full-depth.

Table 5: Quantum resources required for the additions in SHA-2 (one round).

Source	#CNOT	#1qCliff	#T	Toffoli depth	#Qubit (reuse)	Full depth
Kim et al. [6]	18,367	3,362	19,124	224	501 (85)	1777
Lee et al. [5]	-	-	-	66	546 (162)	-
Ours	9,220	1,487	6,546	19	819 (371)	180

⁵ We could not extrapolate the result from [7, 8, 9] since the implementation technique is not specific.

Optimizing for a fixed input length. In the SHA-2-256 quantum circuit, the length of the input is fixed at 256 bits ($W_0 \sim W_7$). The words of W_8 and W_{15} are padded with constant values, and the words of $W_9 \sim W_{14}$ are set to 0. We can initialize the words W_8 and W_{15} after their use (using X gates) and efficiently utilize them, in a total of 256 qubits (32×8), instead of allocating new qubits (specifically, it is used for the output of W_i). Also, we skip the addition operations involving $W_9 \sim W_{14}$ as they are zero. This method is applied from the 10th to the 15th rounds, reducing the Toffoli depth by 2 for each round, resulting in a total reduction of 12.

(Conditional) Borrowing technique. We search idle qubits from the round function (Expressions (6)) and then borrow it for the message scheduling (Expressions (7)). We provide Algorithm 2 to decide the qubits to borrow from the round function. Even after generating all the W_i , the round function continues to proceed. We conditionally borrow the idle output qubits from the last round (i.e., in reverse order) and return them before the round function begins, resulting in a total reduction of 896 ($= 32 \times 28$) qubits.

Algorithm 2: Conditional borrowing in SHA-2-256

Input: Output qubit set $output_{round}$ in round function

Output: The output qubit set $output_{round}$ borrowed during message scheduling

```

1: Check for idle output qubits in  $output_{round}$ 
2: for the round  $i$  from 0 to 39 do
3:   if  $i < 28$  then
4:     if  $i < 19$  then
5:       Put  $output_{round}^{(63-i)}$  into  $ancilla_{borrow}^{[2i:2i+1]}$ 
6:     end if
7:     Use  $ancilla_{borrow}^i$  for the  $i$ -th message scheduling
8:   else Allocate  $output_{new}^i$  for the  $i$ -th message scheduling
9:   end if
10: end for
11: for each  $i \in ancilla_{borrow}^{[0:27]}$  in reverse order do
12:   Reverse  $i$ -th message scheduling
13: end for
14: return  $ancilla_{borrow}$ 

```

$\triangleright ancilla_{borrow}^{[28:37]}$ are employed in other operations
 \triangleright Initialize $ancilla_{borrow}^i$

Consideration of AND gate. The target qubit of the AND gate must be in a clean state due to the nature of the AND gate. Thus, we modify the quantum circuit of Ch in Figure 3 as shown in Figure 5. The same modification is applied to the circuit of Maj and similarly to the Draper adder. Additionally, to construct quantum circuits for Ch , Maj , and partial sums (① in the QCSA) using AND gates, additional ancilla qubits are required. However, we borrow idle qubits (in a clean state, $|0\rangle$) from the Draper adders in the QCSA (i.e., ③). Thus, additional ancilla qubits are allocated only for the AND gates in the Draper adders. The borrowed qubits are initialized after the operation of AND gates.

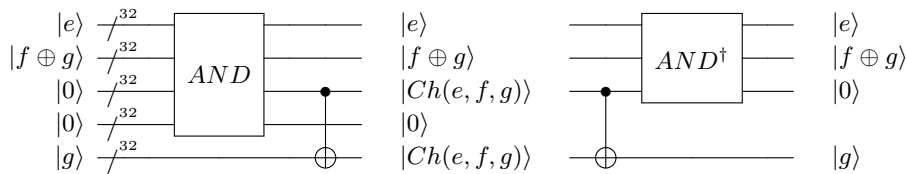


Figure 5: Quantum circuit for the AND-based Ch .

7 Results

Tables 6 and 7 present the required quantum resources for the quantum circuits and Grover’s oracle of SHA-2 in comparison to previous works [5, 6, 7, 8, 9]⁶. Our quantum circuits for SHA-2 provide the best results in terms of Toffoli and full depths. While the full depth was not reported in [5], the T -depth optimization technique presented therein increases the full depth as a trade-off. Although the trade-off performance of our TD - M is slightly lower than that reported in [5], our quantum circuit achieves improvement in the key metrics for Grover’s algorithm (TD^2 - M and FD^2 - M) through depth optimization. Furthermore, the previous work implemented only the quantum circuit for SHA-2-256 or SHA-2-512, but we extended our quantum circuits to include SHA-2-384 and SHA-2-512.

Table 6: Quantum resources required for implementations of SHA-2.

Hash function	Source	#CNOT	#1qCliff	#T	Toffoli depth (TD)	#Qubit (M)	Full depth (FD)	TD - M	FD - M	TD^2 - M	FD^2 - M
SHA-2-256	Amy et al. [7]	534,272	515,952	401,584	57,184	2,402	528,768	$1.02 \cdot 2^{27}$	$1.18 \cdot 2^{30}$	$1.79 \cdot 2^{42}$	$1.19 \cdot 2^{49}$
	Amy et al. [7] (Opt.)	4,209,072	173,264	228,992	57,184	2,402	830,720	$1.02 \cdot 2^{27}$	$1.86 \cdot 2^{30}$	$1.79 \cdot 2^{42}$	$1.47 \cdot 2^{50}$
	Kim et al. [6]	-	-	-	10,112	938	-	$1.13 \cdot 2^{23}$	-	$1.40 \cdot 2^{36}$	-
	Lee et al. [5]	-	-	-	4,418	962	-	$1.01 \cdot 2^{22}$	-	$1.09 \cdot 2^{34}$	-
	Häner et al. [9]	-	-	90,292	1,607	23,684	-	$1.13 \cdot 2^{25}$	-	$1.78 \cdot 2^{35}$	-
	Meuli et al. [8]	-	-	90,292	1,607	23,957	-	$1.15 \cdot 2^{25}$	-	$1.80 \cdot 2^{35}$	-
	Ours	693,832	84,086	495,089	1,332	5,715	12,791	$1.81 \cdot 2^{22}$	$1.09 \cdot 2^{26}$	$1.18 \cdot 2^{33}$	$1.70 \cdot 2^{39}$
SHA-2-384	Ours	1,847,124	225,008	1,335,511	1,824	13,773	17,257	$1.50 \cdot 2^{24}$	$1.77 \cdot 2^{27}$	$1.33 \cdot 2^{35}$	$1.87 \cdot 2^{41}$
SHA-2-512	Häner et al. [9]	-	-	231,788	3,303	60,448	-	$1.49 \cdot 2^{27}$	-	$1.20 \cdot 2^{39}$	-
	Meuli et al. [8]	-	-	231,788	3,304	59,995	-	$1.48 \cdot 2^{27}$	-	$1.19 \cdot 2^{39}$	-
	Ours	1,864,872	226,533	1,346,011	1,828	13,901	17,303	$1.51 \cdot 2^{24}$	$1.79 \cdot 2^{27}$	$1.35 \cdot 2^{35}$	$1.89 \cdot 2^{41}$

Table 7: Required decomposed quantum resources for Grover’s oracle on SHA-2.

Hash function	Source	#CNOT	#1qCliff	#T	#Measure	T -depth (Td)	#Qubit (M)	Full depth (FD)	Td - M	FD - M	Td^2 - M	FD^2 - M
SHA-2-256	Amy et al. [7]	1,068,544	1,031,904	803,168	0	343,104	2,403	1,057,536	$1.54 \cdot 2^{29}$	$1.18 \cdot 2^{31}$	$1.00 \cdot 2^{48}$	$1.19 \cdot 2^{51}$
	Amy et al. [7] (Opt.)	8,418,144	346,528	457,984	0	140,800	2,403	1,661,440	$1.26 \cdot 2^{28}$	$1.86 \cdot 2^{31}$	$1.35 \cdot 2^{45}$	$1.47 \cdot 2^{52}$
	Kim et al. [6]	-	-	-	0	80,896*	939	-	$1.13 \cdot 2^{26}$	-	$1.40 \cdot 2^{42}$	-
	Lee et al. [5]	-	-	-	0	9,872	963	-	$1.13 \cdot 2^{23}$	-	$1.37 \cdot 2^{36}$	-
	Häner et al. [9]	-	-	180,584	0	3,214	23,685	-	$1.13 \cdot 2^{26}$	-	$1.78 \cdot 2^{37}$	-
	Meuli et al. [8]	-	-	180,584	0	3,214	23,958	-	$1.15 \cdot 2^{26}$	-	$1.80 \cdot 2^{37}$	-
	Ours	1,387,664	168,172	990,178	0	10,656	5,716	25,582	$1.82 \cdot 2^{25}$	$1.09 \cdot 2^{27}$	$1.18 \cdot 2^{39}$	$1.70 \cdot 2^{41}$
	Ours-AND	1,334,920	535,632	340,976	76,620	1,780	5,880	19,034	$1.25 \cdot 2^{23}$	$1.67 \cdot 2^{26}$	$1.08 \cdot 2^{34}$	$1.94 \cdot 2^{40}$
SHA-2-384	Ours	3,694,248	450,016	2,671,022	0	14,592	13,774	34,514	$1.50 \cdot 2^{27}$	$1.77 \cdot 2^{28}$	$1.33 \cdot 2^{41}$	$1.87 \cdot 2^{43}$
	Ours-AND	3,566,072	1,449,800	926,208	208,808	2,544	14,127	26,338	$1.07 \cdot 2^{25}$	$1.39 \cdot 2^{28}$	$1.33 \cdot 2^{36}$	$1.11 \cdot 2^{43}$
SHA-2-512	Häner et al. [9]	-	-	463,576	0	6,606	60,449	-	$1.49 \cdot 2^{28}$	-	$1.20 \cdot 2^{41}$	-
	Meuli et al. [8]	-	-	463,576	0	6,608	59,996	-	$1.48 \cdot 2^{28}$	-	$1.19 \cdot 2^{41}$	-
	Ours	3,729,744	453,066	2,692,022	0	14,624	13,902	34,606	$1.51 \cdot 2^{27}$	$1.79 \cdot 2^{28}$	$1.35 \cdot 2^{41}$	$1.89 \cdot 2^{43}$
	Ours-AND	3,597,016	1,460,202	932,192	210,304	2,548	14,255	26,394	$1.08 \cdot 2^{25}$	$1.40 \cdot 2^{28}$	$1.35 \cdot 2^{36}$	$1.13 \cdot 2^{43}$

8 Quantum Circuit Implementation of SHA-3

SHA-3 performs 24 round operations, and each round consists of θ , ρ , π , χ and ι . The input of the SHA-3 quantum circuit consists of a 1,600-qubit state, denoted as state $S[x][y][z]$. The state S is the three-dimensional array and the sizes of the x , y , and z are 5, 5, and 64, respectively. With a novel architecture, our SHA-3 quantum circuit achieves the lowest Toffoli depth and requires fewer qubits, specifically 24 and 22400, respectively. In comparison, the previous best results reported in [8] were 24 and 44,798, respectively.

⁶ [7] (Opt.) refers to the T -depth-optimized version presented by the author. In [5, 6], only the qubit count, Toffoli depth, and T -depth were reported, and the T count was additionally reported in [8, 9].

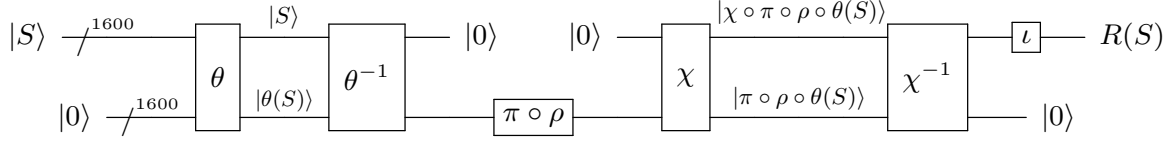


Figure 6: In-place round architecture in [7].

8.1 Improved out-of-place implementation

Amy et al. [7] proposed a round in-place architecture as depicted in Figure 6. In order to implement this, reversed components of θ and χ (i.e., θ^{-1} and χ^{-1}) were used. In contrast, Haner et al. [9] and Meuli et al. [8] presented an out-of-place architecture to optimize Toffoli depth, specifically targeting the operation of χ in SHA-3, which is the only Toffoli-demanding operation in SHA-3. Recently, Song et al. [14] also employed an out-of-place architecture to reduce Toffoli depth and full depth. However, the SHA-3 quantum circuits in [8, 33] still provide better performance (fewer qubits and lower Toffoli depth) compared to [14].

We present an improved out-of-place implementation, as depicted in Figure 7, that optimizes Toffoli and full depths, and further reduces qubit count. In the following, we offer an in-depth explanation of our components.

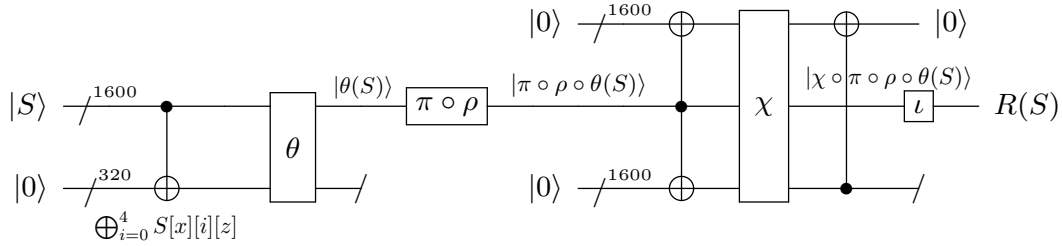


Figure 7: Out-of-place round implementation (Ours).

8.2 Implementation of θ

In the operation of θ , the XOR result of the $S(x-1, z)$ column and the $S(x+1, z-1)$ column is XORed with $S(x, y, z)$. The linear operation on state S is given by:

$$S[x][y][z] = S[x][y][z] \oplus \left(\bigoplus_{i=0}^4 S[x-1][i][z] \oplus S[x+1][i][z-1] \right) \quad (8)$$

In [7], the authors implemented θ (Equation 8) using 17,600 CNOT gates, and this count is derived as follows:

- $S[x][y][z] \oplus \rightarrow + 1$ CNOT gate.
- $\left(\bigoplus_{i=0}^4 S[x-1][i][z] \oplus S[x+1][i][z-1] \right) \rightarrow + 10$ CNOT gates.
- $[x][y][z]$ range $\rightarrow \times 5 \times 5 \times 64 = 1600$.

The quantum circuit for θ , with a total of 17,600 CNOT gates (11 CNOT gates \times 1,600), was presented in [7]. In [7], the authors also implemented the reverse operation of θ (i.e., θ^{-1}) to initialize ancilla qubits to the zero state. This approach has a benefit as it can reduce the number of qubits by continuously reusing the initialized ancilla qubits. However, the implementation of θ^{-1} requires 1,360,000 CNOT gates, which is significantly larger than what is required for θ (17,600 CNOT gates).

This is because assigning a formula for the reverse of θ is more challenging, as it depends on specific pre-computed constants. In [7], a total of 1,377,600 CNOT gates are used, including both θ and θ^{-1} , with a depth of 300.

Our implementation does not involve the reverse operation (θ^{-1}), reflecting our design philosophy. We do not aim to keep the qubit count low. Instead, our goal is to minimize depth, even if it means slightly increasing the number of qubits. We present an optimized quantum circuit for θ using only a total of 4,800 CNOT gates, with a depth of 15. Algorithm 3 describes our quantum circuit implementation of θ .

Algorithm 3: Quantum circuit implementation of θ .

Input: The state $S[x][y][z]$

Output: $\theta(S)$

```

1: Prepare the XOR results of  $S[x][z]$  columns
2: Allocate 320 qubits  $\rightarrow S_{ancilla}[x][z]$ 
3: for each column  $\in S[x][z]$  and each qubit  $\in S_{ancilla}[x][z]$  do
4:   for  $i = 0$  to 4 do
5:      $S_{ancilla}[x][z] \leftarrow \text{CNOT}(S[x][i][z], S_{ancilla}[x][z])$ 
6:   end for
7: end for
8: Update  $S[x][y][z]$  using  $S_{ancilla}[x][z]$ 
9: for each qubit  $\in S[x][y][z]$  do
10:   $S[x][y][z] \leftarrow \text{CNOT}(S_{ancilla}[x-1][z], S[x][y][z])$ 
11:   $S[x][y][z] \leftarrow \text{CNOT}(S_{ancilla}[x+1][z-1], S[x][y][z])$ 
12: end for
13: Discard  $S_{ancilla}[x][z]$ 
14: return  $S[x][y][z]$ 

```

All-in-One. In our implementation for θ , 320 ancilla qubits are allocated in advance to prepare the XOR result of the $S[x-1][z]$ column and the $S[x+1][z-1]$ column (i.e., $\bigoplus_{i=0}^4 S[x-1][i][z]$ and $\bigoplus_{i=0}^4 S[x+1][i][z-1]$). It is the same as preparing the XOR results of all columns of $S[x][z]$ (i.e., $\bigoplus_{i=0}^4 S[x][i][z]$). For this, 5 CNOT gates are employed to store the XOR result for each ancilla qubit, resulting in a total of 1,600 ($= 5 \times 320$) CNOT gates and a circuit depth of 5. As a result, the XOR results of all columns of $S[x][z]$ are stored in the 320 ancilla qubits.

Then, we can efficiently update $S[x][y][z]$ by repeatedly utilizing the pre-computed results stored in the 320 ancilla qubits, thereby avoiding the need to compute the XOR result each time. In this update, the XOR results of two columns (i.e., $S[x-1][z]$ and $S[x+1][z-1]$), stored in ancilla qubits, are XORed with each state of $S[x][y][z]$. Thus, $1,600 \times 2$ CNOT gates are used, resulting in a circuit depth of 10. Thanks to this pre-computation method, our quantum implementation of θ requires only 4,800 ($= 1,600 + 3,200$) CNOT gates and has a depth of 15 ($= 5 + 10$).

In [8, 9], optimized quantum circuits for θ were not presented as the focus was on developing algorithms for Toffoli metrics in quantum implementation, specifically targeting the χ operation, which requires Toffoli gates in SHA-3. In [14], Song et al. implemented only the quantum circuit of θ . However, they allocated 1,600 ancilla qubits for θ and did not reuse these ancilla qubits (i.e., no θ^{-1}). Although a similar approach is adopted, their implementation requires 24,000 CNOT gates, and the depth is 79, both of which are higher than the costs of our implementation.

Table 8 shows the required quantum resources for the implementation of θ in comparison to previous implementations of θ . The reported quantum resources for [7] include both θ and θ^{-1} , and 1,600 out of the 3,200 qubits are reused. Note that we also reuse 320 ancilla qubits, allocated to prepare the XOR result, with our new architecture. This will be described in detail in Section 8.4.

Table 8: Quantum resources required for implementations of θ .

Source	#CNOT	#Qubit (reuse)	Depth
Amy et al. [7]*	1,377,600	3,200 (1,600)	300
Song et al. [14]	24,000	3,200	79
Ours	4,800	1,920 (320)*	15

*: Include both θ and θ^{-1} .

*: Partially reused.

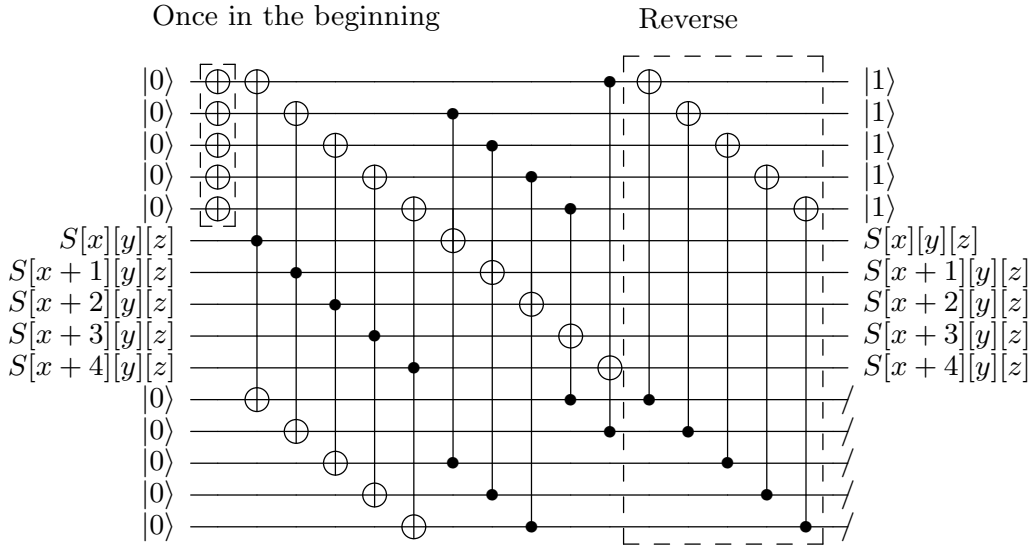
8.3 Implementation of χ

The component of the SHA-3 quantum circuit implementation that requires the most quantum resources is χ , as Toffoli gates are exclusively necessary here. The operation of χ is the sole non-linear operation in SHA-3, and its operation on the state S is defined by:

$$S[x][y][z] = S[x][y][z] \oplus (\sim S[x+1][y][z] \cdot S[x+2][y][z]). \quad (9)$$

In other words, χ represents the following 3-bit S-box: 05234167.

We present a depth-optimized quantum circuit for χ by following our two principles (allocate ancilla qubits and reverse them). Figure 8 shows our quantum circuit implementation of χ .

**Figure 8:** Quantum circuit of χ in SHA-3

Parallel design with copying. The operation of $\sim S[x+1][y][z] \cdot S[x+2][y][z]$ is implemented using Toffoli gates, and our quantum circuit for χ is designed with Toffoli depth one (i.e., all Toffoli gates are operated in parallel). According to our first principle, we allocate 3,200 ($= 1,600 \times 2$) ancilla qubits to generate two copies of the state S . The state S is copied using 3,200 CNOT gates. One of the two copies is inverted using 1,600 X gates (used as $\sim S[x+1][y][z]$). In this way, we independently prepare all operands of Equation 9, allowing us to implement χ with Toffoli depth one using 1,600 Toffoli gates.

We present a quantum circuit implementation of χ with minimal depth, achieved by allocating additional ancilla qubits. However, as previously mentioned, our design philosophy prioritizes minimizing depth while also aiming to reduce the overall number of qubits through the reuse of ancilla qubits (as per our second design principle).

Reusing ancilla qubits. We allocated ancilla qubits for two copies of the state S . One copy serves as the output, while the other serves as an operand for the χ operation. Notably, the copy designated for the operand only plays a crucial role in enabling parallel execution during the χ operation. Consequently, we can initialize these ancilla qubits after the operation by applying the reverse operation of the copy process (i.e., utilizing 1,600 CNOT gates once again, see Figure 8). This initialization ensures that the ancilla qubits are in a clean state, ready for use in the subsequent χ operation without the need for reallocation. Note that the other ancilla qubits for the other copy (the bottom lines in Figure 8) will be initialized/reused with our new architecture and will be further discussed in Section 9.

Trick of the X gate operation. We omit the X gate operation from the reverse operation. Instead of initializing the ancilla qubits to $|0\rangle$, we leave the ancilla qubits in the flipped state (i.e., $|1\rangle$) by skipping the X gate operation. This approach avoids the need for an X gate operation in the next round, resulting in reduced depth and fewer gates. As illustrated in Figure 8, the X gate operation is applied only once in the initial round, and subsequent rounds no longer require an X gate.

The quantum implementation of χ is a well-demonstrated case for reducing depth while considering the number of qubits (our design philosophy). Table 9 shows the quantum resources required for implementations of χ (including χ^{-1}), in comparison to previous implementations [7, 14]. Although specific implementation methods are not described in [8, 9], we anticipate that parallel concepts similar to our quantum circuit for χ may have been incorporated.

Table 9: Quantum resources required for implementations of χ .

Source	#CNOT	#1qCliff	#T	Toffoli depth	#Qubit (reuse)	Full depth
Amy et al. [7]	33,280	14,400	24,640	11	3,200 (1,600)	121
Song et al. [14]	10,240	6,400	11,200	5	2,240	47
Ours	14,400	6,400	11,200	1	4,800 (3,200)*	10

*: Partially reused.

8.4 Interval Architecture

We introduce a novel interval architecture that can reuse many ancilla qubits used in the round process. The interval architecture performs reverse operations with intervals while executing round operations. Note that the developed method is generic, and in our case, the optimal interval is 4 rounds. Thanks to the interval architecture, we reduce the number of qubits for the SHA-3 implementation by 26,800 without increasing the circuit depth.

Recall that the out-of-place round implementation in Figure 7. We allocated two copies for the parallel operation of χ . Among them, only one copy could be initialized, and then the other copy was discarded (i.e., garbage qubits). We denote this garbage qubits generated in the r -th round as Gbg_r .

However, if we view the out-of-place implementation over two rounds (i.e., not in a single round), we can initialize the Gbg_1 used in the 1st round using the Gbg_2 used in the 2nd round. Let us perform the reverse operation of the operations $\pi \circ \rho \circ \theta$ (the red rectangle in Figure 9(a)) from the 2nd round on the state Gbg_2 . Keep in mind that Gbg_2 is a copy of the input of χ , thus we can successfully initialize ancilla qubits for θ used in the 2nd round. Also, since the state of Gbg_2 changes to $R_1(S)$ after the reverse operation, we can perform the reverse operation (the blue rectangle in Figure 9(a)) of the 1st round using Gbg_2 .

From this 2-round interval process, we identify the following two aspects to determine the optimal interval for the SHA-3 implementation:

- In the *first* reverse operation, we can initialize only 320 ancilla qubits used in θ .

- In the *last* reverse operation, we can initialize all 3520 ($= 1600 + 1600 + 320$) ancilla qubits.

Let the interval be 3 rounds, as depicted in Figure 9(b). The first and last reverse operations are exactly the same as those in the 2-round interval process. The only difference is that we use Gbg_3 instead of Gbg_2 . For the middle round (i.e., 2nd round, the yellow rectangle in Figure 9(b)), all ancilla qubits are initialized. However, it should be noted that the initialized copy (on the top lane, not the Gbg_2 on the bottom lane) of the input of χ in the 2nd round is used for the last reverse operation. Thus, strictly speaking, the Gbg_2 and ancilla qubits for the operation of θ are initialized. In the SHA-3 implementation, the optimal interval for the reverse process is 4 rounds. Details regarding this decision will be provided in the next section.

- In the *middle* reverse operation, we can initialize 1920 ($= 1600 + 320$) ancilla qubits.

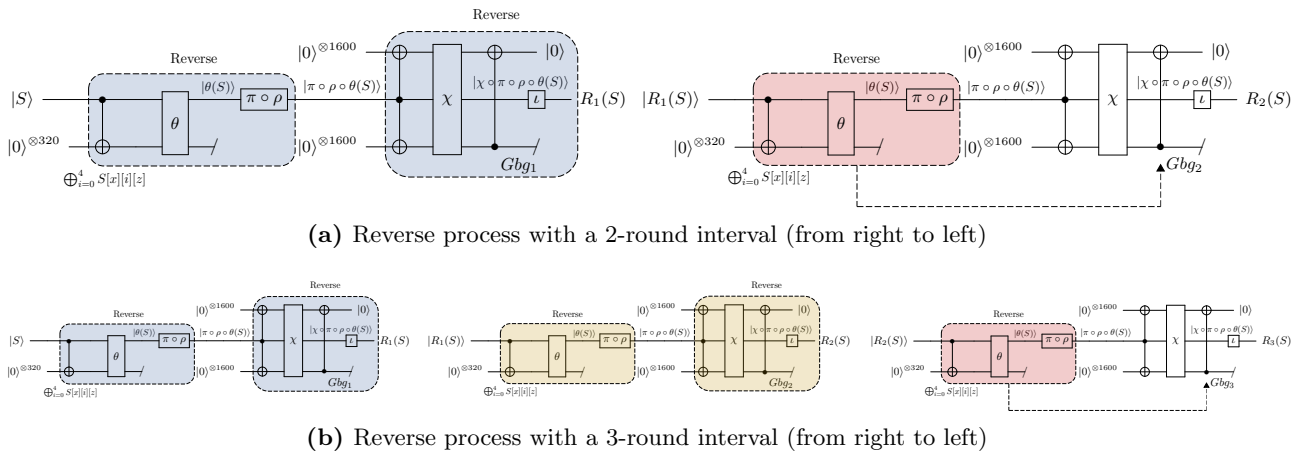


Figure 9: Interval architecture in SHA-3 quantum circuit.

Shallow technique In [7, 11, 13, 21], the presented quantum circuits wait for the reverse operation to reuse the initialized qubits in the subsequent operations. This approach can reduce the number of qubits but increases the circuit depth. In contrast, we burden the number of qubits only in the initial stage and perform the subsequent rounds and reverse operations in parallel. In other words, the subsequent rounds continue without waiting for the reverse operations. This concept was first proposed in [26] for the quantum implementation of AES, referred to as the *shallow* architecture. We incorporate this concept into our interval architecture.

Let the interval round be n . The reverse process is performed from the end of round- n , and at the same time, round- $(n + 1)$ begins. Recall that for the out-of-place round implementation (Figure 7), 320 ancilla qubits are required for the operation of θ , and 3,200 qubits are required for the operation of χ . We allocate 3,520 ($= 320 + 1,600 + 1,600$) ancilla qubits for round- $(n + 1)$ (this is the initial burden as mentioned before) because we do not have any ancilla qubits for round- $(n + 1)$ since the reverse operation of round- n has just begun. From round- $(n + 2)$, we can reuse initialized ancilla qubits in the reverse operations.

Figure 10(a) illustrates the interval architecture for $n = 3$ using the shallow technique. As we just mentioned, round-4 requires 3,520 ($= 320 + 3200$) ancilla qubits (denote this as A). On the other hand, rounds 5 and 6 can reuse initialized ancilla qubits from the reverse operations of rounds 3 and 2 (i.e., Round 3[†] and Round 2[†]), respectively (See Figure 10(a)). Recall the following two points: Rounds 5 and 6 require only 1,920 ($= 320 + 1,600$) ancilla qubits each (denote this as B), since one copy of the operation χ (1600 ancilla qubits) is initialized and reused (See Figure 8). The reverse operations of

rounds 3 and 2 (denote these as C and B for first and middle, respectively) initialize 320 and 1920 ancilla qubits, respectively (See Figure 9). Thus, round-5 requires only 1,600 ancilla qubits ($= B - C$) by reusing initialized ancilla qubits from the reverse operation of round-3 (Round 3^\dagger), and round-6 does not require any ancilla qubits ($B - B$) by reusing initialized ancilla qubits from the reverse operation of round-2 (Round 2^\dagger).

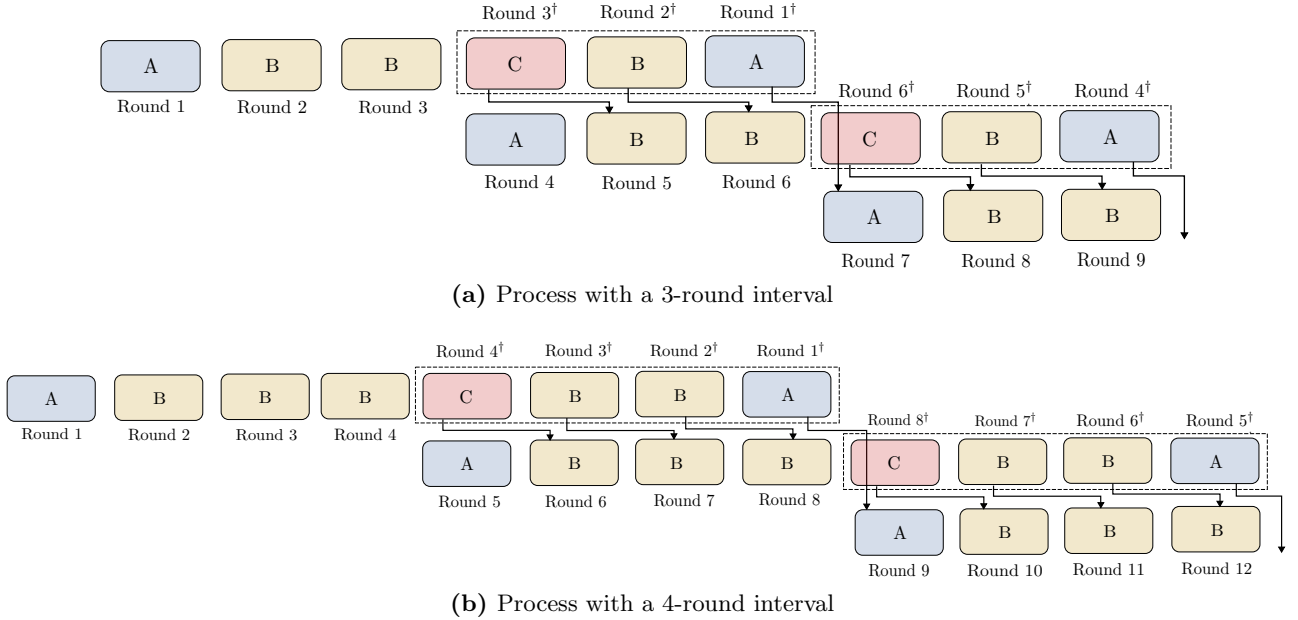


Figure 10: Shallow technique.

Like this, we iterate the reverse process and reuse technique with the interval. Note that except for the initial state (i.e., Round 4 in Figure 10(a)), the rounds which require A ancilla qubits (e.g., Round 7 in Figure 10(a)) fully reuse A ancilla qubits from the last reverse operation, without additional ancilla qubits. Depending on the interval n for 24 rounds, we can determine the required ancilla qubits as follows ($n \neq 1$).

- Ancilla qubits: $A = 3,520$, $B = 1,920$ and $C = 320$.
- Total ancilla qubits: $2A + B \cdot (n - 1) + (B - C) \cdot \lceil 24/(n - 1) \rceil$.

The optimal interval bound is $n = 4$, depicted in Figure 10(b), with the number of required ancilla qubits being 20,800.

Consideration of AND gate. For an AND gate, an additional ancilla qubit is required. Therefore, we allocate 1,600 ancilla qubits to perform 1,600 AND gates in the operation of χ . It is important to note that these ancilla qubits are initialized after the AND gates, and we reuse them subsequently.

The target qubit of the AND gate must be in a clean state (i.e., $|0\rangle$) due to the nature of the AND gate. However, as shown in Figure 8, the target qubits of the AND operations are not in a clean state. To address this, we slightly modify the circuits (with no degradation in performance) presented in Figure 8 as follows (the circuit is represented in three parts: top, middle, and bottom).

- The copying of the bottom lines is postponed.
- The target qubits of AND operations are changed from the middle to the bottom lines.
- The copying of the bottom lines (which is postponed) is performed.
- The bottom lines represent the result, and the middle lines are discarded.

8.5 Implementation of ρ , π and ι

In SHA-3, the operations of ρ and π correspond to rotation and rearrangement operations. Thus, similar to SHA-2, we implement ρ and π using the logical swap method without requiring quantum resources.

The operation of ι , which involves XORing the round constant with the state S , is categorized in classical-quantum implementation. As the round constants are predefined, X gates are applied based on the bits of the constant, where the value is 1. The implementation of ι is simplified using only X gates, and this is a conventional and generally adopted approach in quantum implementations [7, 14, 26, 34, 35].

8.6 Results

Tables 10 and 11 present the required quantum resources for the quantum circuits and Grover's Oracle of SHA-3 in comparison to previous work [7, 8, 9, 14]⁷. Our quantum circuits for SHA-3 provide the best results in terms of depths and trade-off performance compared to the previous work [7, 8, 9, 14].

Table 10: Quantum resources required for implementations of SHA-3.

Source	#CNOT	#1qCliff	#T	Toffoli depth (TD)	#Qubit (M)	Full depth (FD)	$TD-M$	$FD-M$	TD^2-M	FD^2-M
Amy et al. [7]	33,269,760	169,045	591,360	264	3,200	10,128	$1.61 \cdot 2^{19}$	$1.93 \cdot 2^{24}$	$1.66 \cdot 2^{27}$	$1.19 \cdot 2^{38}$
Amy et al. (Opt) [7]	34,260,480	215,125	499,200	264	3,200	11,040	$1.61 \cdot 2^{19}$	$1.05 \cdot 2^{25}$	$1.66 \cdot 2^{27}$	$1.42 \cdot 2^{38}$
Häner et al. [9]	.	.	153,600	24	46,400	.	$1.06 \cdot 2^{20}$.	$1.59 \cdot 2^{24}$.
Meuli et al. [8]	.	.	153,600	24	44,798	.	$1.03 \cdot 2^{20}$.	$1.54 \cdot 2^{24}$.
Song et al. [14]	821,760	153,688	268,800	120	55,360	2,860	$1.58 \cdot 2^{22}$	$1.18 \cdot 2^{27}$	$1.48 \cdot 2^{29}$	$1.65 \cdot 2^{38}$
Ours	752,000	124,937	425,600	24	22,400	578	$1.03 \cdot 2^{19}$	$1.54 \cdot 2^{23}$	$1.54 \cdot 2^{23}$	$1.74 \cdot 2^{32}$

Table 11: Required decomposed quantum resources for Grover's oracle on SHA-3.

Source	#CNOT	#1qCliff	#T	#Measure	T -depth (Td)	#Qubit (M)	Full depth (FD)	$Td-M$	$FD-M$	Td^2-M	FD^2-M
Amy et al. [7]	66,539,520	338,090	1,182,720	0	1,584	3,201	20,256	$1.21 \cdot 2^{22}$	$1.93 \cdot 2^{25}$	$1.87 \cdot 2^{32}$	$1.19 \cdot 2^{40}$
Amy et al. (Opt) [7]	68,520,960	430,250	998,400	0	864	3,201	22,080	$1.32 \cdot 2^{21}$	$1.05 \cdot 2^{26}$	$1.11 \cdot 2^{31}$	$1.42 \cdot 2^{40}$
Häner et al. [9]	.	.	307,200	0	48	46,411	.	$1.06 \cdot 2^{21}$.	$1.59 \cdot 2^{26}$.
Meuli et al. [8]	.	.	307,200	0	48	44,799	.	$1.03 \cdot 2^{21}$.	$1.54 \cdot 2^{26}$.
Song et al. [14]	1,643,520	307,376	537,600	0	960	55,361	5,720	$1.58 \cdot 2^{25}$	$1.18 \cdot 2^{28}$	$1.48 \cdot 2^{35}$	$1.65 \cdot 2^{40}$
Ours	1,504,000	249,874	851,200	0	192	22,401	1,156	$1.03 \cdot 2^{22}$	$1.54 \cdot 2^{24}$	$1.54 \cdot 2^{29}$	$1.74 \cdot 2^{34}$
Ours-AND	1,321,600	387,474	243,200	60,800	43	24,001	1,049	$1.97 \cdot 2^{19}$	$1.5 \cdot 2^{24}$	$1.32 \cdot 2^{25}$	$1.54 \cdot 2^{34}$

9 Quantum Collision Search on Hash Functions

Based on the quantum circuits for SHA-2 and SHA-3 presented in this work, we estimate the required quantum resources for quantum collision search using the CNS algorithm (refer to Section 3). To find a collision for the n -bit output of SHA-2 or SHA-3, n -bit input within the search space 2^n is explored. According to the CNS algorithm [24], Grover's circuit finds a collision with the search complexity (iteration) of $2^{2n/5 - 3s/5}$ using a parallelization method with $s = n/6$ (as we determined

⁷ The SHA-3 quantum circuit requires the same quantum resources for input lengths of 256, 384 and 512; due to the equal number of permutation.

in Section 3). As described in Section 3, the cost of the diffusion operator is ignored in resource estimations [11, 12, 21, 26, 34], and we also exclude this cost from our estimation. Finally, the complexity of quantum collision search for cryptographic hash functions SHA-2 and SHA-3 is approximately the cost of the Oracle $\times 2^{2n/5-3s/5}$ (where n is the output size and $s = n/6$). We use Toffoli and AND-based decompositions to estimate the quantum resources for quantum collision search.

In Table 12, the results use the Toffoli-based decomposition:

- 8 Clifford + 7 T gates, T -depth 4, and full depth 8.

Table 12: Toffoli-based quantum resources for quantum collision search on SHA-2 and SHA-3.

Hash	#Gate (G)	Full depth (FD)	T -depth (Td)	#Qubit (M)	G - FD	FD - M	Td - M	FD^2 - M	Td^2 - M
SHA-2-256	$1.49 \cdot 2^{97}$	$1.06 \cdot 2^{91}$	$1.77 \cdot 2^{89}$	$1.1 \cdot 2^{55}$	$1.77 \cdot 2^{188}$	$1.18 \cdot 2^{146}$	$1.97 \cdot 2^{144}$	$1.26 \cdot 2^{237}$	$1.75 \cdot 2^{234}$
SHA-2-384	$1.47 \cdot 2^{137}$	$1.9 \cdot 2^{129}$	$1.6 \cdot 2^{128}$	$1.68 \cdot 2^{77}$	$1.39 \cdot 2^{267}$	$1.59 \cdot 2^{207}$	$1.35 \cdot 2^{206}$	$1.51 \cdot 2^{337}$	$1.08 \cdot 2^{335}$
SHA-2-512	$1.95 \cdot 2^{175}$	$1.25 \cdot 2^{168}$	$1.06 \cdot 2^{167}$	$1.06 \cdot 2^{99}$	$1.22 \cdot 2^{344}$	$1.34 \cdot 2^{267}$	$1.13 \cdot 2^{266}$	$1.68 \cdot 2^{435}$	$1.20 \cdot 2^{433}$
SHA-3-256	$1.70 \cdot 2^{97}$	$1.54 \cdot 2^{86}$	$1.02 \cdot 2^{84}$	$1.08 \cdot 2^{57}$	$1.30 \cdot 2^{184}$	$1.67 \cdot 2^{143}$	$1.1 \cdot 2^{141}$	$1.29 \cdot 2^{230}$	$1.12 \cdot 2^{225}$
SHA-3-384	$1.12 \cdot 2^{136}$	$1.01 \cdot 2^{125}$	$1.34 \cdot 2^{122}$	$1.36 \cdot 2^{78}$	$1.14 \cdot 2^{261}$	$1.39 \cdot 2^{203}$	$1.84 \cdot 2^{200}$	$1.41 \cdot 2^{328}$	$1.23 \cdot 2^{323}$
SHA-3-512	$1.48 \cdot 2^{174}$	$1.34 \cdot 2^{163}$	$1.77 \cdot 2^{160}$	$1.72 \cdot 2^{99}$	$1.98 \cdot 2^{337}$	$1.15 \cdot 2^{263}$	$1.52 \cdot 2^{260}$	$1.55 \cdot 2^{426}$	$1.35 \cdot 2^{421}$

The quantum circuits, constructed using AND-based decomposition, have the lowest gate count and circuit depth. In Table 13, the results use the AND-based decomposition:

- AND gate: 11 Clifford + 4 T gates, T -depth 1, and full depth 8.
- AND[†] gate: 5 Clifford + 1 Measurement gates, T -depth 0, and full depth 4.

Table 13: AND-based quantum resources for quantum collision search on SHA-2 and SHA-3.

Hash	#Gate (G)	Full depth (FD)	T -depth (Td)	#Qubit (M)	G - FD	FD - M	Td - M	FD^2 - M	Td^2 - M
SHA-2-256	$1.49 \cdot 2^{97}$	$1.58 \cdot 2^{90}$	$1.18 \cdot 2^{87}$	$1.13 \cdot 2^{55}$	$1.18 \cdot 2^{188}$	$1.81 \cdot 2^{145}$	$1.35 \cdot 2^{142}$	$1.43 \cdot 2^{236}$	$1.60 \cdot 2^{229}$
SHA-2-384	$1.32 \cdot 2^{137}$	$1.45 \cdot 2^{129}$	$1.12 \cdot 2^{126}$	$1.72 \cdot 2^{77}$	$1.91 \cdot 2^{266}$	$1.25 \cdot 2^{207}$	$1.93 \cdot 2^{203}$	$1.81 \cdot 2^{336}$	$1.08 \cdot 2^{330}$
SHA-2-512	$1.76 \cdot 2^{175}$	$1.91 \cdot 2^{167}$	$1.48 \cdot 2^{164}$	$1.09 \cdot 2^{99}$	$1.68 \cdot 2^{343}$	$1.05 \cdot 2^{267}$	$1.62 \cdot 2^{263}$	$1.00 \cdot 2^{435}$	$1.20 \cdot 2^{428}$
SHA-3-256	$1.31 \cdot 2^{97}$	$1.39 \cdot 2^{86}$	$1.79 \cdot 2^{81}$	$1.16 \cdot 2^{57}$	$1.83 \cdot 2^{183}$	$1.62 \cdot 2^{143}$	$1.04 \cdot 2^{139}$	$1.13 \cdot 2^{230}$	$1.87 \cdot 2^{220}$
SHA-3-384	$1.73 \cdot 2^{135}$	$1.84 \cdot 2^{124}$	$1.18 \cdot 2^{120}$	$1.46 \cdot 2^{78}$	$1.59 \cdot 2^{260}$	$1.35 \cdot 2^{203}$	$1.73 \cdot 2^{198}$	$1.24 \cdot 2^{328}$	$1.02 \cdot 2^{319}$
SHA-3-512	$1.14 \cdot 2^{174}$	$1.21 \cdot 2^{163}$	$1.56 \cdot 2^{158}$	$1.84 \cdot 2^{99}$	$1.39 \cdot 2^{337}$	$1.12 \cdot 2^{263}$	$1.44 \cdot 2^{258}$	$1.36 \cdot 2^{426}$	$1.12 \cdot 2^{417}$

9.1 Update on NIST Post-Quantum Security Strength

We present updated post-quantum security levels in Table 2 by selecting the lowest quantum attack complexity for the cryptographic hash functions SHA-2 and SHA-3 and provide a brief discussion on this matter. In the initial stages (in 2016) [4], NIST designated AES-128, 192, and 256 at levels 1, 3, and 5, respectively, based on the implementation by Grassl et al. [21], and its quantum attack complexity was high. Recent research has significantly reduced the quantum attack complexity of AES [11, 12, 26] and NIST updated the quantum attack complexities accordingly primarily based on the results from Jaques et al’s work [11].

NIST designated SHA-2/3-256 at level 2 and SHA-2/3-384 at level 4 by judging that in real-world cryptanalysis, success tends to be highest when attackers can exploit highly parallel implementations. Consequently, they consider finding collisions in a hash function to be easier than searching for the key of a block cipher (does not parallelize well).

Currently, the quantum attack complexity for levels 2 and 4, corresponding to collision search on hash functions SHA-2 and SHA-3, has not been defined yet. In this work, in defining the quantum attack complexity for levels 2 and 4, we fully incorporate NIST’s judgment and considerations [36, Page 7]. We assumed the highly parallelized quantum collision search for levels 2 and 4 to be positioned between levels 1 and 3, and levels 3 and 5, respectively. Indeed, Tables 1 and 2 demonstrate that appropriate quantum attack complexities for the post-quantum security levels are defined as follows: level 1: 2^{157} , level 2: $2^{188/183}$, level 3: 2^{221} , level 4: $2^{266/260}$, and level 5: 2^{285} . Along with this, we also define the quantum complexity of collision search for SHA-2-512 and SHA-3-512 for the extended level as $2^{343/337}$, which provides the highest post-quantum security strength.

10 Conclusion

In this work, we focused on optimizing the depth of quantum circuits for the cryptographic hash functions SHA-2 and SHA-3. We detailed the implementation of target cryptographic hash algorithms in terms of core components (e.g., linear layer and additions in the round function) and architectural level. We integrated all the novel and best implementation techniques for optimizing the depth of quantum circuit implementations for SHA-2 and SHA-3 across in categories and evaluated the necessary quantum resources using architectures employing gate decomposition. After that, we obtained the lowest quantum resources for quantum collision search on SHA-2 and SHA-3. Finally, we defined the NIST quantum attack complexity for level 2 and level 4. Along with this, we suggested one more option that provides the highest post-quantum security level, the extended level.

A Time-Space Complexity Trade-off under MAXDEPTH

The quantum gate count for levels in Table 1 is derived as the product of the gate count and the full depth (e.g., level 1 for AES-128: $2^{157} = 2^{87} \times 2^{75}$, [11, Table 11]). Additionally, NIST introduced a parameter, MAXDEPTH, to account for the extreme depth of Grover’s algorithm when applied to cryptographic algorithms. If the quantum attack circuit exceeds these specified boundaries for the MAXDEPTH, it is recommended to consider parallelizing Grover’s algorithm. However, Grover’s algorithm has poor performance for parallelization, as analyzed in [6, 11]. Summarizing the analysis from [6], reducing the circuit depth by S requires increasing the number of Grover instances by S^2 (i.e., unbalanced).

If the total circuit depth D exceeds MAXDEPTH, a depth reduction using a parallel approach must be applied to satisfy MAXDEPTH. The reduction factor, S , is calculated as $\frac{D}{\text{MAXDEPTH}}$ (since $D/S = \text{MAXDEPTH}$). For the gate count, G , the count for each instance is reduced by S , and the number of instances increases by S^2 . Thus, the estimation formula for Table 1 is derived as $G \cdot \frac{D}{\text{MAXDEPTH}}$ by $\frac{G}{S} \cdot S^2$. This formulation illustrates that NIST takes into account gate count, depth, and MAXDEPTH when estimating the complexity of quantum attacks.

In terms of the trade-off metrics, $TD-M$ and $FD-M$ (where M is the qubit count, TD and FD represent Toffoli and Full depths, respectively), the qubit count M is increased by S^2 (i.e., $\frac{FD^2 \cdot M}{\text{MAXDEPTH}^2}$). Thus, in parallelization, the $FD-M$ cost changes to $\frac{FD^2 \cdot M}{\text{MAXDEPTH}}$ (with FD replaced by MAXDEPTH). In other words, the metrics of $FD-M$ and $TD-M$ transform into minimizing the FD^2-M and TD^2-M metrics under the constraint of MAXDEPTH.

References

1. P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, IEEE, 1994. **1**
2. L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996. **1**
3. NIST., “Call for additional digital signature schemes for the post-quantum cryptography standardization process,” 2022. <https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf>. **1, 2, 3, 6**
4. NIST., “Submission requirements and evaluation criteria for the post-quantum cryptography standardization process,” 2016. <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>. **2, 6, 21**
5. J. Lee, S. Lee, Y.-S. Lee, and D. Choi, “T-depth reduction method for efficient SHA-256 quantum circuit construction,” *IET Information Security*, vol. 17, no. 1, pp. 46–65, 2023. **2, 3, 7, 8, 9, 10, 11, 13**
6. P. Kim, D. Han, and K. C. Jeong, “Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2,” *Quantum Information Processing*, vol. 17, pp. 1–39, 2018. **2, 3, 7, 9, 11, 13, 22**
7. M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent, and J. Schanck, “Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3,” in *Selected Areas in Cryptography – SAC 2016* (R. Avanzi and H. Heys, eds.), (Cham), pp. 317–337, Springer International Publishing, 2017. **2, 3, 7, 9, 11, 13, 14, 15, 16, 17, 18, 20**
8. G. Meuli, M. Soeken, and G. De Micheli, “XOR-and-inverter graphs for quantum compilation,” *npj Quantum Information*, vol. 8, no. 1, p. 7, 2022. **2, 3, 11, 13, 14, 15, 17, 20**
9. T. Häner and M. Soeken, “Lowering the T-depth of quantum circuits by reducing the multiplicative depth of logic networks,” *arXiv preprint arXiv:2006.03845*, 2020. **2, 3, 11, 13, 14, 15, 17, 20**
10. P. Gossett, “Quantum carry-save arithmetic,” *arXiv preprint quant-ph/9808061*, 1998. **2, 10**
11. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, “Implementing Grover Oracles for quantum key search on AES and LowMC,” in *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II* (A. Canteaut and Y. Ishai, eds.), vol. 12106 of *Lecture Notes in Computer Science*, pp. 280–310, Springer, 2020. **3, 4, 5, 6, 18, 21, 22**
12. Q. Liu, B. Preneel, Z. Zhao, and M. Wang, “Improved quantum circuits for AES: Reducing the depth and the number of qubits.” *Cryptology ePrint Archive*, Paper 2023/1417, 2023. <https://eprint.iacr.org/2023/1417>. **3, 4, 6, 21**
13. J. Zou, Z. Wei, S. Sun, X. Liu, and W. Wu, “Quantum circuit implementations of AES with fewer qubits,” in *Advances in Cryptology – ASIACRYPT 2020* (S. Moriai and H. Wang, eds.), (Cham), pp. 697–726, Springer International Publishing, 2020. **3, 18**
14. G. Song, K. Jang, and H. Seo, “Improved low-depth SHA3 quantum circuit for fault-tolerant quantum computers,” *Applied Sciences*, vol. 13, no. 6, p. 3558, 2023. **3, 14, 15, 16, 17, 20**
15. I. Van Hoof, “Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count,” *arXiv preprint arXiv:1910.02849*, 2019. **4**
16. K. Jang, W. Kim, S. Lim, Y. Kang, Y. Yang, and H. Seo, “Optimized implementation of quantum binary field multiplication with Toffoli depth one,” in *Information Security Applications: 23rd International Conference, WISA 2022, Jeju Island, South Korea, August 24–26, 2022, Revised Selected Papers*, pp. 251–264, Springer, 2023. **4**
17. M. Amy, D. Maslov, M. Mosca, M. Roetteler, and M. Roetteler, “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*, vol. 32, p. 818–830, Jun 2013. **4**
18. Y. He, M.-X. Luo, E. Zhang, H.-K. Wang, and X.-F. Wang, “Decompositions of n -qubit Toffoli gates with linear circuit complexity,” *International Journal of Theoretical Physics*, vol. 56, no. 7, pp. 2350–2361, 2017. **4**
19. P. Selinger, “Quantum circuits of T-depth one,” *Physical Review A*, vol. 87, no. 4, p. 042302, 2013. **4**
20. P. Niemann, A. Gupta, and R. Drechsler, “T-depth optimization for fault-tolerant quantum circuits,” in *2019 IEEE 49th International Symposium on Multiple-Valued Logic (ISMVL)*, pp. 108–113, IEEE, 2019. **4**
21. M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, “Applying Grover’s algorithm to AES: Quantum resource estimates,” in *Post-Quantum Cryptography* (T. Takagi, ed.), (Cham), pp. 29–43, Springer International Publishing, 2016. **4, 5, 6, 18, 21**
22. G. Brassard, P. Hoyer, and A. Tapp, “Quantum algorithm for the collision problem,” *arXiv preprint quant-ph/9705002*, 1997. **5**
23. D. J. Bernstein, “Cost analysis of hash collisions: Will quantum computers make sharcs obsolete,” *SHARCS*, vol. 9, p. 105, 2009. **5**
24. A. Chailloux, M. Naya-Plasencia, and A. Schrottenloher, “An efficient quantum collision search algorithm and implications on symmetric cryptography,” in *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23*, pp. 211–240, Springer, 2017. **6, 20**
25. G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, “Quantum amplitude amplification and estimation,” *Contemporary Mathematics*, vol. 305, pp. 53–74, 2002. **6**
26. K. Jang, A. Baksi, H. Kim, G. Song, H. Seo, and A. Chattopadhyay, “Quantum analysis of AES.” *Cryptology ePrint Archive*, Paper 2022/683, 2022. <https://eprint.iacr.org/2022/683>. **6, 18, 20, 21**

27. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, “Implementing Grover Oracles for quantum key search on AES and LowMC.” Cryptology ePrint Archive, Report 2019/1146, 2019. <https://eprint.iacr.org/2019/1146>. 7
28. S. Roy, A. Baksi, and A. Chattopadhyay, “Quantum implementation of ascon linear layer.” Cryptology ePrint Archive, Paper 2023/617, 2023. <https://eprint.iacr.org/2023/617>. 8
29. Y. Yang, K. Jang, A. Baksi, and H. Seo, “Optimized implementation and analysis of cham in quantum computing,” *Applied Sciences*, vol. 13, no. 8, p. 5156, 2023. 8
30. S. Cuccaro, T. Draper, S. Kutin, and D. Moulton, “A new quantum ripple-carry addition circuit.” arXiv, 2008. <https://arxiv.org/pdf/quant-ph/0410184.pdf>. 9
31. T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, “A logarithmic-depth quantum carry-lookahead adder,” *arXiv preprint quant-ph/0406142*, 2004. 10
32. Y. Takahashi, S. Tani, and N. Kunihiro, “Quantum addition circuits and unbounded fan-out,” 2009. <https://arxiv.org/abs/0910.2530>. 10
33. T. Häner, S. Jaques, M. Naehrig, M. Roetteler, and M. Soeken, “Improved quantum circuits for elliptic curve discrete logarithms,” in *International Conference on Post-Quantum Cryptography*, pp. 425–444, Springer, 2020. 14
34. A. Baksi, K. Jang, G. Song, H. Seo, and Z. Xiang, “Quantum implementation and resource estimates for rectangle and knot,” *Quantum Information Processing*, vol. 20, dec 2021. 20, 21
35. J. Feng, H. Chen, S. Gao, L. Fan, and D. Feng, “Improved fault analysis on the block cipher speck by injecting faults in the same round,” in *Information Security and Cryptology – ICISC 2016: Seoul, South Korea, Revised Selected Papers* (S. Hong and J. H. Park, eds.), (Cham), pp. 317–332, Springer International Publishing, 2017. 20
36. NIST., “Post-quantum cryptography standardization historical faqs,” 2016. <https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/archive/faq-historical.pdf>. 22