The Journal of Supercomputing (2026) 82:164
https://doi.org/10.1007/s11227-026-08255-5

Low-qubit quantum circuits for efficient integer squaring

Laura M. Donaire[1] · Gloria Ortega[1] · Ester M. Garzón[1] · Ernestas Filatovas[2] · Francisco Orts[1]

Received: 12 September 2025 / Accepted: 17 January 2026

Abstract

Quantum squaring circuits play a critical role in many quantum algorithms; however, most existing designs incur a significant qubit overhead due to the loss of input states and excessive use of ancillary qubits. In this work, we introduce a qubit-efficient quantum circuit for integer squaring that achieves a linear qubit cost of only $3N$ qubits for an N-bit input, significantly outperforming state-of-the-art designs that scale quadratically in terms of qubits. Our approach reintegrates the input operand after computation, enabling the uncomputation of intermediate results and efficient recycling of ancilla qubits. This reversible strategy prevents the retention of redundant information, which is a common limitation of prior works. The comparative analysis confirms the scalability and practicality of our design for qubit-constrained quantum hardware, offering a promising solution for arithmetic operations in resource-limited quantum environments.

Keywords Quantum circuits · Integer squaring · Qubit optimization · Reversible computation · Low-resource quantum computing

Laura M. Donaire, Gloria Ortega, Ester M. Garzón, Ernestas Filatovas and Francisco Orts have contributed equally to this work.

✉ Francisco Orts
francisco.orts@ual.es

Laura M. Donaire
laura.donaire@ual.es

Gloria Ortega
gloriaortega@ual.es

Ester M. Garzón
gmartin@ual.es

Ernestas Filatovas
ernestas.filatovas@mif.vu.lt

[1] Supercomputing-Algorithms Group, University of Almería, Almería, Spain

[2] Institute of Data Science and Digital Technologies, Faculty of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania

Published online: 13 February 2026

1 Introduction

Quantum computing is a computational paradigm that leverages principles of quantum mechanics, such as superposition, interference, and entanglement, to solve certain problems more efficiently than classical approaches [1]. Among its most celebrated milestones are Shor's algorithm for prime factorization [2] and Grover's algorithm for unstructured search [3], both illustrating how quantum algorithms can offer exponential or quadratic speedups under specific conditions.

One of the most widely used formal frameworks for expressing quantum algorithms is the quantum circuit model [4]. In this model, quantum computations are expressed as sequences of quantum gates acting on a set of qubits. Its analogy to classical digital circuits makes it particularly appealing for the design and analysis of quantum arithmetic operations, which are key building blocks across a broad range of quantum applications [5].

Arithmetic operations such as addition [6], multiplication [7], modular reduction [8], and squaring [9] form the core of many quantum algorithms, especially those involving number-theoretic or algebraic computations. Modular exponentiation of integers, as used for example in Shor's algorithm, is a central subroutine in many quantum cryptanalytic proposals. While it is typically implemented in terms of modular multiplications and may be realized using techniques that avoid explicit squaring [10], full squaring remains a natural arithmetic primitive and can serve as a building block within more general modular arithmetic constructions [11]. In this context, examining the resource cost of full integer squaring in isolation provides meaningful insights into the design of space-efficient quantum arithmetic circuits.

Despite the conceptual simplicity of squaring, existing quantum implementations tend to be resource-intensive [11]. Many designs require a large number of ancilla qubits and discard intermediate results without uncomputation, leading to substantial qubit overhead. This poses a major limitation for execution on current quantum hardware, where qubit availability remains severely constrained [12, 13]. Within the NISQ (Noisy Intermediate-Scale Quantum) paradigm [14], reducing the qubit count and optimizing the circuit structure are critical to achieving practically implementable quantum computations.

In recent years, various approaches to optimize arithmetic circuits have been proposed [15–17]. Among these, qubit efficiency, measured as the number of physical qubits required relative to the input size, remains one of the most pressing challenges [18]. Techniques such as operand reintroduction [19] and reversible computation (or "uncomputation") [20] have proven to be effective strategies for reducing qubit usage. These methods leverage the reversibility of quantum logic to recycle intermediate qubits and eliminate unnecessary garbage states, thus minimizing ancillary overhead.

In this work, we propose a novel, low-qubit quantum circuit for computing the square of an N-bit integer, designed with the explicit goal of minimizing qubit consumption. Our circuit requires only $3N$ qubits, significantly improving upon several state-of-the-art designs that scale quadratically in space complexity (e.g., up to $1.5N^2$ qubits). This gain is achieved by reintroducing the input operand after computation, allowing the circuit to explicitly uncompute intermediate values and reuse ancilla qubits throughout execution.

The main contributions of this work are:

- The design of a reversible, qubit-efficient circuit for integer squaring, suitable for integration into larger quantum algorithms such as modular exponentiation.
- A detailed analysis comparing the qubit cost of our design with existing proposals, highlighting the linear scaling in contrast to quadratic approaches.
- The demonstration of a scalable and space-efficient solution for squaring that is well suited for implementation on current qubit-limited quantum architectures.

The remainder of the paper is structured as follows. Section 2 provides a brief overview of quantum circuit metrics and reviews relevant previous work on squaring circuits. Section 3 details the architecture of our proposed circuit. Section 4 analyzes the resource usage and performance of the design through comparative tables. Finally, Sect. 5 summarizes our findings and outlines directions for future research.

2 Background

Quantum computing is fundamentally distinct from classical computing, not only in underlying physical principles but also in computational models and resource requirements [21]. In the quantum circuit model, algorithms are expressed as sequences of quantum gates applied to qubits, which can exist in complex superpositions of classical bit states. Accordingly, designing efficient quantum circuits relies heavily on understanding the structure of quantum operations and the management of limited computational resources [5].

Quantum circuits are built from a universal set of quantum gates that enable the implementation of arbitrary unitary operations. Among the most commonly used gates are the Clifford gates, which include the Hadamard (H), Phase (S), and CNOT gates, as well as non-Clifford gates such as the T gate. Together, these gates form the widely adopted Clifford+T gate set, which is compatible with fault-tolerant architectures and various quantum error-correcting codes [22].

Although Clifford+T is a convenient universal gate set and a natural choice in the context of fault-tolerant quantum computation, the resource analysis in this work follows the reversible-gate model based on {Toffoli, CNOT, X}, which is standard in quantum arithmetic. This choice reflects the structure of the circuit itself, which does not rely on explicit single-qubit phase rotations and is naturally expressed using controlled logic primitives. The circuit remains compatible with fault-tolerant architectures, since Toffoli gates can be decomposed into Clifford+T operations with known constant and asymptotic cost constructions [23]; depth or T-count estimates can therefore be derived when targeting specific logical hardware models. As the primary focus of this manuscript is the reduction of qubit width rather than gate synthesis over a universal set, we adopt the reversible-gate model for clarity while noting its direct translatability to Clifford+T.

While the gate set itself provides universality, the practical implementation of quantum circuits is constrained primarily by the number of qubits required. In this context, a crucial metric is the qubit count, which includes both the logical input/output qubits and any ancillary qubits introduced to support intermediate computations [24]. Ancilla qubits are generally initialized to the $|0\rangle$ state and used for storing temporary values or enabling reversibility in logic operations [1].

On current physical devices, in the NISQ era, the total number of qubits available is severely limited. Therefore, minimizing the qubit count is a primary design goal, especially for quantum arithmetic circuits. While several classical references and reversible implementations achieve linear-space complexity even in general multiplication [25], many recent squaring circuits adopt generic multiplication-based architectures without operand reuse and thus exhibit quadratic scaling in practice. This motivates specialized designs that exploit the structural symmetry of squaring to reduce qubit footprint.

One fundamental requirement in quantum computation is that all circuits must be reversible, owing to the unitary nature of quantum evolution [26]. This means that every transformation must be bijective, preserving a one-to-one correspondence between inputs and outputs. In practice, this often results in garbage outputs, i.e., qubits that carry no useful final information but are necessary for preserving reversibility. Managing these garbage outputs efficiently is key to maintaining low-qubit overhead [27].

A common approach to addressing this issue is to use Bennett's uncomputation technique [28]. In this strategy, the computation is performed, the result is copied to a separate register, and the computation is then reversed to return all ancilla and garbage qubits to their initial $|0\rangle$ states. This allows for *ancilla recycling* and prevents residual information from propagating, which is essential when the circuit is embedded as a subroutine in a larger computation.

These principles are especially relevant in the context of quantum arithmetic. Operations such as addition, multiplication, and squaring frequently require ancillary space to handle carries or partial results [13]. Designing these circuits to be garbage-free or to support early uncomputation is therefore crucial for achieving qubit-efficient implementations [29]. Furthermore, some arithmetic tasks, such as squaring, naturally offer opportunities for optimization due to their structural properties. In squaring, there is a single input operand, enabling operand reuse and the simplification of control logic, which can substantially reduce the number of active qubits required.

In this work, we leverage reversibility, ancilla reuse, and structural symmetry to design a squaring circuit that achieves significant savings in qubit resources without sacrificing generality. Reducing the qubit footprint is advantageous not only for execution on near-term quantum hardware (where the number of available qubits, routing overhead, and accumulated noise pose fundamental constraints) but also for classical simulation, where memory requirements scale exponentially with the number of qubits and tensor-network contraction widths are strongly size-dependent. By addressing both fronts simultaneously, the proposed circuit provides a space-efficient primitive suitable for integration into larger arithmetic routines, as described in the next subsection.

2.1 Architecture adaptation for squaring

A straightforward strategy to implement squaring is to copy the input into a second N qubit register using N CNOT gates, apply a standard reversible multiplication circuit, and finally uncompute the copy. In terms of additional workspace, this naïve approach introduces N extra qubits for operand duplication, but the total circuit width is at least $4N$ qubits (including the $2N$ qubit output register), plus any ancillary workspace $a(N)$ required by the chosen reversible multiplier. In contrast,

the aim of the present work is to avoid operand duplication altogether and derive a specialized construction that performs squaring directly in a more space-efficient manner.

The squaring circuit proposed in this work builds upon the integer multiplication architecture introduced by Muñoz-Coreas and Thapliyal [30], which we treat as a representative linear-space reversible shift-and-add multiplier. The controlled addition stage in that architecture ultimately derives from the Takahashi–Tani–Kunihiro (TTK) adder [31], with the implementation in [30] instantiating this construction. In our setting, we use this controlled TTK adder as a known primitive and adapt it to the specific structure of squaring, where operand symmetry allows a simplified control pattern and, as a minor circuit-level refinement, the use of lightweight Temporary Logical-AND constructs to streamline specific control patterns and reduce transient overhead. Other multiplier families, such as schoolbook-style reversible designs or QFT-based constructions, can also achieve low or zero ancillary cost, and our choice of [30] is made for its structural compatibility with shift-and-add squaring rather than for global optimality across all known multiplier architectures.

However, unlike general multiplication, integer squaring involves computing $A \cdot A$, where both operands are identical. This structural symmetry allows for specific simplifications not applicable to general-purpose multiplier. In our design, we take advantage of this by adapting the architecture to accept a single input operand A, eliminating the need for a second register. As a result, the complexity of input data loading and control logic is reduced, and the ancilla requirements associated with handling two distinct inputs are significantly lowered.

A central component of our construction is a conditional adder block, which we adapt to the squaring setting. Rather than treating the addition stage as a black-box component, we explicitly tailor its use to the repetitive and predictable control pattern induced by squaring. This adaptation simplifies the control flow and supports the systematic reuse of workspace qubits through uncomputation. By exploiting known operand symmetry and fixed control behavior, the modified adder eliminates unnecessary logic branches and simplifies internal gate arrangements, ultimately supporting a more compact and qubit-efficient implementation.

Additionally, we make use of Temporary Logical-AND constructs [32] as a minor implementation-level optimization for intermediate control logic. This choice does not affect the asymptotic scaling of the proposed construction, but slightly simplifies the local gate structure and facilitates early uncomputation in parts of the circuit. These lightweight subcircuits are especially effective in scenarios where the output of a control operation does not need to be preserved, enabling the early erasure of temporary values through uncomputation.

The proposed circuit achieves a substantial reduction in space complexity by combining operand reuse, reversible logic, and early garbage removal. The architecture remains general and scalable, and its structure allows for seamless integration into broader arithmetic frameworks, such as modular exponentiation, where repeated squaring operations are central. Most importantly, these improvements are achieved without compromising functional correctness, while the construction prioritizes qubit width through systematic ancilla reuse and uncomputation. As is typical in space-optimized reversible designs, this prioritization may increase depth compared to approaches that allocate more workspace.

While this work focuses primarily on minimizing qubit count through operand reuse and reversible computation, a detailed analysis of circuit depth and execution feasibility on specific quantum hardware platforms remains an important direction for future work.

3 Proposed design

Algorithm 1 Adapted conditional adder circuit

Let $A = a_0, a_1, \ldots, a_{N-1}$ and $B = b_0, b_1, \ldots, b_{N-1}$ be two N-bit registers.
Let `Ctrl` be a control qubit.
Prepare two qubits (aux_0 and aux_1) in the state $|T\rangle$ by applying H followed by T.
The result $S = A + B$ will be stored in B and aux_0, conditioned on `Ctrl`.
for $i = 1$ to $N - 1$ **do**
    Apply CNOT from a_i to b_i
end for
Apply Temporary logical-AND with controls $Ctrl$ and a_{N-1}, target aux_0
for $i = N - 2$ to 1 **do**
    Apply CNOT from a_i to a_{i+1}
end for
for $i = 0$ to $N - 2$ **do**
    Apply Toffoli with controls a_i and b_i, target a_{i+1}
end for
Apply Temporary logical-AND with controls a_{N-1} and b_{N-1}, target aux_1.
Apply Toffoli with controls $Ctrl$ and aux_1, target aux_0.
Uncompute previous Temporary logical-AND over aux_1 and release the qubit so that it can be used in future circuits.
for $i = N - 1$ down to 2 **do**
    Apply Toffoli with controls $Ctrl$ and a_i, target b_i
    Apply Toffoli with controls a_{i-1} and b_{i-1}, target a_{i+1}
end for
Apply Toffoli with controls a_0 and b_0, target a_1
Apply Toffoli with controls $Ctrl$ and a_0, target b_0
for $i = 1$ to $N - 2$ **do**
    Apply CNOT from a_i to a_{i+1}
end for
for $i = 1$ to $N - 1$ **do**
    Apply CNOT from a_i to b_i
end for

The architecture introduced by Muñoz-Coreas and Thapliyal [30] involves controlled additions to iteratively compute the product of two unsigned binary integer operands. Therefore, a central building block in our design is the reversible *conditional adder* (Algorithm 1). This subcircuit performs an in-place addition

of two N-bit numbers A and B under the control of a qubit labeled `Ctrl`. The circuit uses a minimal number of ancilla qubits and is fully reversible, ensuring that all temporary qubits can be uncomputed and recycled. It employs a combination of CNOT, Toffoli, and Temporary Logical-AND operations to compute the sum without generating garbage outputs. This efficient sum module serves as the foundation for our squaring algorithm, being instantiated repeatedly with increasing operand sizes.

The full squaring procedure is described in Algorithm 2. The circuit begins by preparing $3N$ qubits: a control qubit `Ctrl` initialized to $|0\rangle$, a register of N qubits for the input A, and ancillae needed for intermediate results and logic operations. The structure of the multiplication is simplified because that only one input is needed (i.e., $A = B$), which allows us to reduce the number of initial logical-AND operations. Specifically, the only relevant terms in the initial partial product are those of the form $a_0 \cdot a_i$, since a_0 is the least significant bit of both operands. These products are computed using Temporary Logical-AND gates and stored in ancillae denoted b_i (Fig. 1).

Algorithm 2 Qubit-efficient quantum squaring circuit

Let $A = a_{N-1}a_{N-2}\ldots a_0$ be an N-bit binary number.
Prepare $3N$ qubits.
Initialize the first qubit to $|0\rangle$ and label it `Ctrl`.
Load the N input bits of A into qubits $a_0, a_1, \ldots, a_{N-1}$.
Select $N-1$ ancilla qubits. For each $i = 1$ to $N-1$:

- Initialize the i-th qubit in the state $|T\rangle$ by applying H followed by T.
- Apply a Temporary Logical-AND with controls a_0 and a_i.
- Label the target qubit of this operation as b_i (Figure 1).

Apply a CNOT gate with control a_1 and target `Ctrl`.
Apply the `Ctrl-Add` circuit (size N) with operands a and b. Since B has one digit less than A, b_{N-1} must be set to 0 (Figure 2(a)).
The circuit outputs updated values of `Ctrl`, a, and s (sum).
Apply a CNOT gate with control a_1 and target `Ctrl`. This part is illustrated in Figure 2(b).
for $i = 1$ to $N-2$ **do**
Apply a CNOT with control a_{i+1} and target `Ctrl`.
Apply a `Ctrl-Add` circuit of size N (Figure 3(a)) using:
- operands: current a and previous s;
- shared ancilla: reused from the previous sum.
Apply a CNOT with control a_{i+1} and target `Ctrl`. This part is illustrated in Figure 3(b).
end for
After the final addition:
- a_0 remains as P_0 (P will be the result).
- For $i = 0$ to $N-2$, each s_0 is relabeled as P_{i+1}.
- The remaining bits of s from the last addition complete the result P.

The main computation proceeds by cascading a series of conditional adder blocks. The first addition is carried out using a `Ctrl-Add` circuit of size N, taking as inputs the original register a and the partial products b. Since b has size $N-1$, the most significant bit b_{N-1} is set to $|0\rangle$ to match the register size. This addition is activated under the control of qubit a_1, which is temporarily copied into `Ctrl` using a CNOT gate (Fig. 2).

Subsequent additions are performed in a similar way: for each bit a_{i+1}, its value is copied into `Ctrl` and used to trigger a conditional addition between the current accumulator and the previously computed sum. Each new addition involves registers of size N. Nevertheless, the ancilla structure is reused across additions, minimizing the total number of qubits. The integration of these additions into the main structure is illustrated in Fig. 3, with the overall construction shown in Fig. 4 for an example with $N = 6$.

Importantly, at each step, intermediate results are carefully uncomputed or recycled to ensure that no garbage accumulates. After the final addition, the result of the squaring operation is distributed across the output lines $P_0, P_1, \ldots, P_{2N-2}$, comprising bits from the original a_0 and the various s_0 and s_i produced by the conditional additions.

4 Results

This section analyzes the qubit efficiency of the proposed squaring circuit compared with several representative designs from the literature. Table 1 summarizes the asymptotic qubit requirements for five different squaring circuits, including our own. The expressions reflect how each circuit scales with input size N, highlighting significant differences in growth patterns.

As seen in Table 1, most previously reported dedicated squaring circuits exhibit a qubit cost that grows quadratically with N. For completeness, we also include the naïve copy–multiply–uncompute baseline [25, 28], which attains linear scaling but requires a larger constant factor in qubit width due to explicit operand duplication and the presence of a separate $2N$-qubit output register (and possibly additional workspace $a(N)$ from the chosen reversible multiplier).

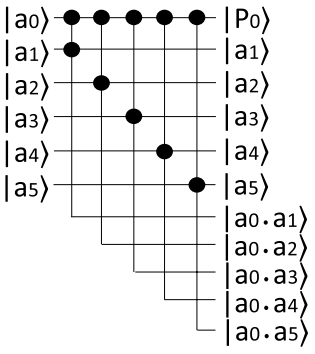
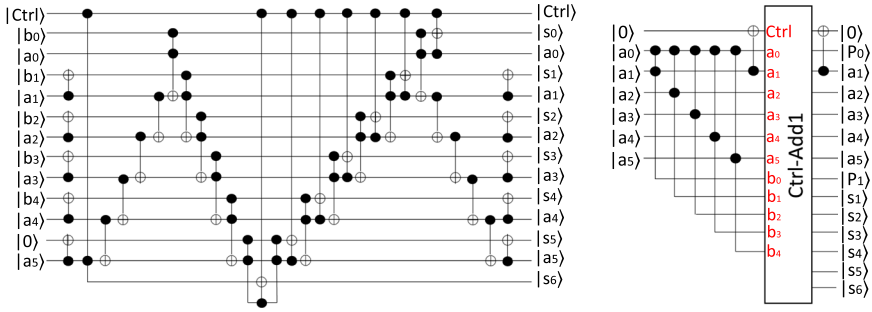


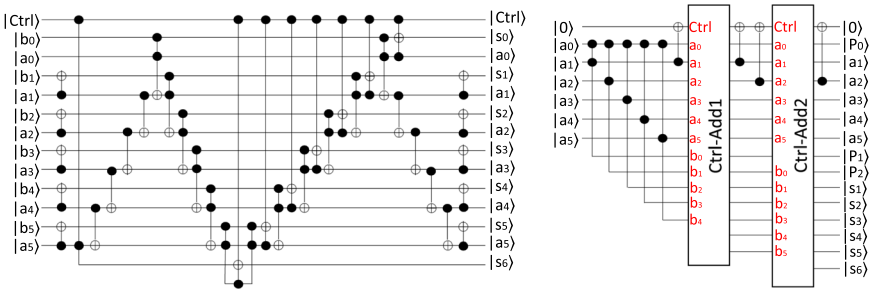
Fig. 1 Simplification of the original product between numbers, made possible by the fact that the operand is unique. An example is shown for the case $N = 6$



(a) Example of the conditional adder circuit (labeled as Ctrl-Add1 in the final circuit) used in the first step of the squaring circuit.

(b) Integration of the first adder in the squaring circuit.

Fig. 2 First use of the adder, for an example of digit size $N = 6$



(a) Example of the conditional adder circuit (labeled as Ctrl-Add2 in the final circuit) used in the second and subsequent steps of the squaring circuit.

(b) Integration of the second adder into the circuit. Subsequent adders are incorporated in a similar manner

Fig. 3 Second use of the adder, for an example of digit size $N = 6$

In contrast, the proposed circuit achieves strictly linear growth in qubit usage, with a total cost of only $3N$ qubits. This improvement is made possible by exploiting the inherent symmetry in the squaring operation, which eliminates the need for separate input registers and reduces overall ancilla usage. Moreover, the design incorporates techniques such as operand reintroduction and ancilla qubit recycling through uncomputation, further enhancing its spatial efficiency. We emphasize that the proposed circuit is not the only construction with linear qubit scaling; its contribution is achieving a reduced width of $3N$ without operand duplication, through a specialized squaring architecture with systematic reuse and uncomputation.

To further illustrate the asymptotic gap, note that quadratic-width designs grow rapidly with N, whereas the proposed circuit grows linearly as $3N$. For example, at $N = 20$ our circuit requires only 60 qubits, while the circuits in [9, 11] require 598 and 591 qubits, respectively.

This linear behavior is particularly advantageous in practice, especially for quantum systems operating under the NISQ paradigm, where qubit availability is a major

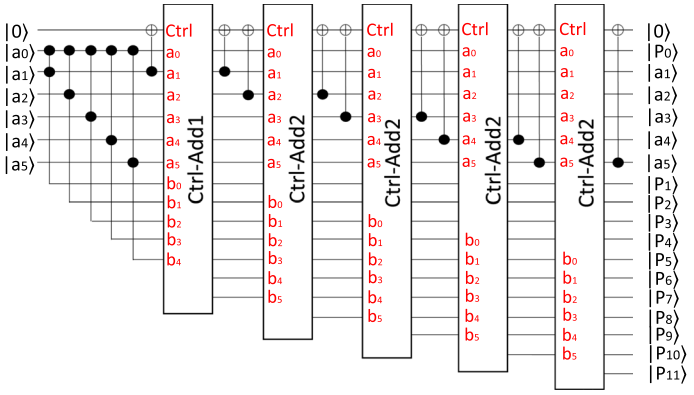


Fig. 4 Complete circuit for calculating squares. The construction for digits of size $N = 6$ is shown

constraint. The predictability and reduced footprint of our design not only make it easier to implement on physical hardware but also simplify circuit compilation, routing, and scheduling, which are known to become increasingly complex as the number of qubits grows.

Additionally, this qubit efficiency is achieved without relying on any special assumptions about the input or the computation model. The circuit is general-purpose, fully reversible, and suitable for integration into larger arithmetic routines, such as modular exponentiation and number-theoretic subroutines commonly used in quantum cryptographic applications. It therefore offers a viable solution for space-constrained quantum computations where squaring is a recurring operation.

Overall, the results clearly demonstrate that the proposed circuit substantially outperforms previous dedicated squaring designs reported in the literature in terms of qubit usage, especially when the input size increases. It also improves over the naïve baseline in qubit width by avoiding operand duplication. Its compactness, scalability, and compatibility with current hardware limitations position it as a compelling alternative for quantum arithmetic in practical applications. However, it is important to note that the comparative analysis

Table 1 Comparison, in terms of number of qubits, between the proposed circuits and the most relevant squaring circuits available in the literature

Work	Required qubits
Jayashree et al. [33]	$0.5N^2 + 3.5N + 1$
Nagamani et al. [34]	$0.5N^2 + 2.5N + 2$
Sultana and Muñoz-Coreas [9]	$1.5N^2 - 1.5$
Cho et al. [11]	$1.5N^2 - 0.5N + 1$
Naïve (copy + multiply + uncompute) [25, 28]	$\geq 4N + a(N)$
Proposed circuit	$3N$

For the naïve baseline, $a(N)$ denotes the ancillary workspace required by the chosen reversible multiplier

in this section focuses on qubit count as the primary resource metric. The proposed circuit is specifically engineered to minimize width via ancilla reuse and systematic uncomputation, and should therefore be interpreted as a space-efficient alternative rather than a depth-optimal construction. In general, this strategy tends to increase circuit depth relative to designs that allocate additional workspace. A detailed multi-metric comparison, including gate counts and depth under a unified Clifford+T cost model, is left for future work.

Table 2 complements the qubit comparison by summarizing coarse-grained Clifford+T costs under a consistent decomposition model. The results make the space–time trade-off explicit: our construction minimizes circuit width via operand reuse and systematic uncomputation, which increases sequentiality and thus leads to larger T-count and T-depth than some higher-width designs. In particular, while several existing squaring circuits achieve smaller constants in T-metrics, they do so at the expense of a substantially larger qubit footprint (Table 1).

5 Conclusions

This work introduces a qubit-efficient quantum circuit for computing the square of an integer, specifically tailored for space-constrained quantum architectures. By leveraging the structural redundancy inherent in squaring operations, our design avoids the overhead of duplicating operands and minimizes ancilla usage through operand reintroduction and systematic uncomputation. The resulting circuit achieves a linear qubit cost of $3N$ for an N-bit input, offering a substantial improvement over existing designs, which exhibit quadratic qubit growth. A complementary Clifford+T comparison is provided to make the space–depth trade-off explicit, showing that the reduced width is obtained at the cost of increased T-count and T-depth due to sequential ancilla reuse.

The reduction in qubit count has implications in both classical and quantum settings. On the one hand, fewer qubits reduce memory and computational requirements for classical simulation, where time and space complexity scale exponentially with circuit width, enabling simulations that remain tractable in regimes where quadratic-width designs become infeasible. On the other hand, smaller qubit footprints also benefit physical implementations on near-term quantum hardware by reducing routing overhead, lowering accumulated noise, and simplifying calibration and control. The proposed circuit therefore serves as a practical primitive for resource-limited quantum architectures while remaining amenable to large-scale classical analysis and verification.

Furthermore, the general-purpose and reversible nature of the circuit allows it to be seamlessly integrated into more complex quantum arithmetic routines, such as modular exponentiation. Overall, this work contributes a scalable and hardware-aware solution to a fundamental quantum arithmetic task, and it opens avenues for further space-optimized circuit design in future quantum applications.

Table 2 Coarse-grained resource comparison beyond qubit count, using Clifford+T metrics (T-count and T-depth)

Construction	T-count	T-depth
Jayashree et al. [33]	$20N^2 - 8N - 9$	$8N^2 - 8N - 11$
Nagamani et al. [34]	$22N^2 - 24N - 12$	$8N^2 - 6N - 8$
Sultana and Muñoz-Coreas [9]	$5N^2 - 4N - 4$ (N even) $5N^2 - 6N - 3$ (N odd)	$2.5N^2 - 2N - 2$ (N even) $2.5N^2 - 3N - 1.5$ (N odd)
Cho et al. [11]	$4N^2 - 4N$	$N^2 - N + 1$
Naïve [25, 28]	$\Theta(g_{\text{mult}}(N))$	$\Theta(d_{\text{mult}}(N))$
Proposed circuit	$21N^2 + 5N - 4$	$9N^2 + N - 2$

For Jayashree et al., Nagamani et al., Sultana and Muñoz-Coreas, and Cho et al., the expressions are taken from Table 3 in [11], which reports closed-form T-count and T-depth under a consistent decomposition model. For the naïve baseline, $g_{\text{mult}}(N)$ and $d_{\text{mult}}(N)$ denote the gate count and depth of the chosen reversible multiplier, and the overall cost scales as $\Theta(g_{\text{mult}}(N))$ and $\Theta(d_{\text{mult}}(N))$. These formulas are intended to provide a consistent, high-level comparison rather than hardware-specific compiled costs. All constructions are reversible

Acknowledgements This work has received financial support from projects PID2021-123278OB-I00 and PID2024-155521OB-I00 (funded by MCIN/AEI/10.13039/501100011033/FEDER "A way to make Europe"), from the assistance with reference POST_2024_00998 funded by the Regional Government of Andalusia /CUII and the ESF+, from the 2024-2025 Internal Research and Knowledge Transfer Program of the University of Almería, and from the Research Council of Lithuania under Grant Agreement No. S-ITP-25-5, and under the Program "University Excellence Initiatives" of the Ministry of Education, Science and Sports of the Republic of Lithuania (Measure No. 12-001-01-01-01 "Improving the Research and Study Environment"), Project No. S-A-UEI-23-11.

Author Contributions LMD contributed to writing—review and editing, validation, and visualization; GO contributed to writing—review and editing, validation, and visualization; EMG contributed to orchestration, idea source, and writing—original draft; EF contributed to idea source, formal analysis, and writing—original draft; FO contributed to writing—original draft, software, and execution of experiments.

Funding Funding for open access publishing: Universidad de Almería/CBUA.

Data Availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare no Conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Nielsen MA, Chuang I (2002) Quantum computation and quantum information. American Association of Physics Teachers
2. Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. IEEE, pp 124–134
3. Grover LK (1997) Quantum mechanics helps in searching for a needle in a haystack. Phys Rev Lett 79(2):325
4. Katabarwa A, Gratsea K, Caesura A, Johnson PD (2024) Early fault-tolerant quantum computing PRX. Quantum 5(2):020101
5. Fisher MP, Khemani V, Nahum A, Vijay S (2023) Random quantum circuits. Annu Rev Condens Matter Phys 14(1):335–379
6. Orts F, Ortega G, Combarro EF, Garzón EM (2020) A review on reversible quantum adders. J Netw Comput Appl 170:102810
7. Orts F, Filatovas E, Ortega G, SanJuan-Estrada J, Garzón E (2023) Improving the number of T gates and their spread in integer multipliers on quantum computing. Phys Rev A 107(4):042621
8. Yuan Y, Wang C, Wang B, Chen Z-Y, Dou M-H, Wu Y-C, Guo G-P (2023) An improved QFT-based quantum comparator and extended modular arithmetic using one ancilla qubit. New J Phys 25(10):103011
9. Sultana A, Muñoz-Coreas E (2025) Resource optimized quantum squaring circuit. J Supercomput 81(2):391
10. Iqbal SS, Zafar A (2024) Enhanced Shor's algorithm with quantum circuit optimization. Int J Inf Technol 16(4):2725–2731
11. Cho S-M, Lee C, Seo S-H (2025) Quantum circuit designs of efficient squaring. Quantum Inf Process 24(2):35
12. AbuGhanem M, Eleuch H (2024) NISQ computers: a path to quantum supremacy. IEEE Access
13. Wang Y, Liu J (2024) A comprehensive review of quantum machine learning: from NISQ to fault tolerance. Rep Prog Phys
14. Preskill J (2018) Quantum computing in the NISQ era and beyond. Quantum 2:79
15. Donaire LM, Ortega G, Garzón EM, Orts F (2024) Lowering the cost of quantum comparator circuits. J Supercomput 80(10):13900–13917
16. Remaud M, Vandaele V (2025) Ancilla-free quantum adder with sublinear depth. In: International Conference on Reversible Computation. Springer, pp 137–154
17. Wang S, Baksi A, Chattopadhyay A (2023) A higher radix architecture for quantum carry-lookahead adder. Sci Rep 13(1):16338
18. Sciorilli M, Borges L, Patti TL, García-Martín D, Camilo G, Anandkumar A, Aolita L (2025) Towards large-scale quantum optimization solvers with few qubits. Nat Commun 16(1):476
19. Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E, Latorre JI (2020) Data re-uploading for a universal quantum classifier. Quantum 4:226
20. Swathi M, RudraB (2021) Implementation of reversible logic gates with quantum gates. In: 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, pp 1557–1563
21. Combarro EF, González-Castillo ADMS, Di Meglio A (2023) A practical guide to quantum machine learning and quantum optimization. Packt Publishing, Birmingham
22. Gottesman D (1997) Stabilizer codes and quantum error correction. California Institute of Technology, Pasadena
23. Orts F, Ortega G, Garzón EM (2022) Studying the cost of n-qubit toffoli gates. In: International Conference on Computational Science. Springer, pp 122–128
24. Klimov PV, Bengtsson A, Quintana C, Bourassa A, Hong S, Dunsworth A, Satzinger KJ, Livingston WP, Sivak V, Niu MY (2024) Optimizing quantum gates towards the scale of logical qubits. Nat Commun 15(1):2442

25. Vedral V, Barenco A, Ekert A (1996) Quantum networks for elementary arithmetic operations. Phys Rev A 54(1):147
26. Regula B, Lami L (2024) Reversibility of quantum resources through probabilistic protocols. Nat Commun 15(1):3096
27. Mohammadi M, Eshghi M (2009) On figures of merit in reversible and quantum logic designs. Quantum Inf Process 8(4):297–318
28. Bennett CH (1973) Logical reversibility of computation. IBM J Res Dev 17(6):525–532
29. McArdle S, Gilyén A, Berta M (2022) Quantum state preparation without coherent arithmetic. arXiv:2210.14892
30. Muñoz-Coreas E, Thapliyal H (2018) Quantum circuit design of a T-count optimized integer multiplier. IEEE Trans Comput 68(5):729–739
31. Takahashi Y, Tani S, Kunihiro N (2009) Quantum addition circuits and unbounded fan-out. arXiv:0910.2530
32. Gidney C (2018) Halving the cost of quantum addition. Quantum 2:74
33. Jayashree H, Thapliyal H, Agrawal VK (2017) Efficient circuit design of reversible square. Trans Comput Sci 29:33–46
34. Nagamani A, Ramesh C, Agrawal VK (2018) Design of optimized reversible squaring and sum-of-squares units. Circuits Syst Signal Process 37:1753–1776