A Framework for Quantum Circuit Optimization: Comparators as a Case Study

Francisco Orts and Rodrigo Gil-Merino

***Abstract*—Qubits are the minimum unit of information in quantum computing. As current quantum devices have a very limited number of available qubits, finding circuits that optimize their use is crucial for this computing paradigm. Optimizing the qubits needed to implement a quantum circuit is not trivial, as these circuits have important restrictions such as the impossibility of copying values or the requirement that the computation done in them must always be reversible. This paper presents a framework for optimizing circuits for quantum computing. The framework is able to obtain, given a quantum circuit, an equivalent one that is more efficient in terms of number of qubits. In particular, the framework has applicability in quantum circuits dedicated to arithmetic operations such as addition or multiplication. As a representative case study, it is applied to a reversible comparator circuit, obtaining a design that reduces the number of qubits and controlled gates compared to other reversible comparators reported in the recent literature. As part of the work, a public repository is included with the necessary code to simplify any other circuit.**

***Index Terms*—Quantum computing, quantum circuits, quantum optimization, reversible circuits, quantum comparator.**

I. Introduction

QUANTUM computing is able to solve certain problems more efficiently than other models of computation [1]. Shor's and Grover's algorithms are possibly the best demonstration of this superiority [2], [3]. The former is able to find the factors of a number in polynomial time, something impossible for any classical algorithm existing today. Grover's algorithm, on the other hand, can find one or more elements in an unordered database faster than all other known classical search algorithms. But quantum algorithms that achieve computational advantages are not limited to these two; there is a wide variety of them with applications in areas as diverse as chemistry [4], linear algebra [5], or machine learning [6], among others.

Received 22 May 2025; revised 30 October 2025; accepted 7 December 2025. Date of publication 12 December 2025; date of current version 15 January 2026. This work was supported in part by MCIN/AEI/10.13039/501100011033/FEDER "ERDF A way to make Europe" through Project PID2021-123278OB-I00, and PDC2022-133370-I00; and in part by the Regional Government of Andalusia /CUII and the ESF+ from the assistance with reference POST_2024_00998. Recommended for acceptance by K. Chakraborty. *(Corresponding author: Francisco Orts.)*

Francisco Orts is with the Supercomputing-Algorithms Group, Informatics Department, CEINSA, University of Almería, 04120 Almería, Spain (e-mail: francisco.orts@ual.es).

Rodrigo Gil-Merino is with Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja, 26006 Logroño, Spain.

Digital Object Identifier 10.1109/TC.2025.3642981

Although it can offer computational advantages with examples such as the aforementioned Shor or Grover algorithms, quantum computing is in full development and is currently facing certain limitations. One of these is the limited amount of resources available in current quantum devices: they have a limited number of qubits, so the problems that can be solved today are constrained by this number [7]. Fortunately, the number of qubits available to quantum computers is increasing, so this problem is becoming progressively smaller. But in the meantime, researchers are forced to look for efficient ways to represent their algorithms.

Another problem, even more serious than the previous one, is the extreme sensitivity of current quantum computers to internal and external noise [8]. Internal noise is caused by defects and inaccuracies in the devices themselves, while external noise is caused by any external disturbance. The effects of noise negatively affect the state and communication of the qubits, causing quantum algorithms to produce erroneous results. From an implementation point of view, there is intense research focused on achieving better implementations of qubits, so that the effects of noise can be reduced [9]. But there are other ways to combat the effects of noise. For instance, the design of the algorithms themselves can be done with these effects in mind and, to some extent, they can be detected and even corrected for [10]. There is also another important related factor which is the length of the algorithm: an algorithm that runs quickly will be exposed to noise for a short time and will be less prone to suffer from its effects [11].

From the two problems mentioned above (resource scarcity and noise), it can be stated that given two algorithms solving the same problem, the smaller one in terms of the number of qubits and operations needed will be more efficient than the other one [12]. Of course, assuming the operations are of the same order. The smaller algorithm is more efficient because it will consume fewer qubits, being able to solve on larger data. And in turn, it is more efficient because it will be less affected by noise by the simple fact that it is exposed to it for less time. In relation to reducing execution time, it may also be worthwhile to parallelize operations where possible, as this will also reduce the total execution time and the algorithm's exposure to noise [13].

Even with the drawbacks of the so-called noisy intermediate-scale quantum (NISQ) devices, they are already being used successfully to solve a large number of problems [14]. One of the main responsible for this is the qubit itself. Its power

is such that even with a small number of qubits it is possible to achieve surprising results if data representations are used to exploit the qubit's ability to represent infinite values [11]. A bit can contain one of two values: 0 or 1. The bit can represent any binary system, such as a switch that can be either on or off. A qubit, on the other hand, can be defined as any unit column vector in $\mathbb{C}^2$, and can therefore take infinite values. Despite the astonishing superiority of the qubit over the bit, it is common to use qubits to represent encoding binary information [15]. While this encoding is useful, wasting a qubit to represent a bit is very costly.

In this paper, we propose a framework to optimize the number of qubits needed to implement a quantum circuit. The idea behind the framework is based on the work of Pérez-Salinas et al. [11] in which they built a neural network using a single qubit. The work of Pérez-Salinas et al. is based on a simple but powerful idea: instead of inputting information directly into qubits, it can be "reintroduced" along a circuit as operations, rather than as binary values in the qubits. To implement a neural network, the authors use different spins of the qubit trying to find segments in the data. In our work, instead of trying to segment information, we focus on arithmetic circuits. This type of circuits work using binary operations, but this does not imply that the information must be encoded in binary. We have detected different operations that can be simplified and carried out by encoding the operands in the operation itself. We have therefore adapted the idea of Pérez-Salinas et al. to a different context, redefining its operations and allowing us to optimize circuits already available in the literature. The proposed framework assumes classical (basis-state) inputs, i.e., the method is designed for deterministic comparisons between non-superposed binary strings rather than quantum superpositions or entangled states.

The main contributions of this work are the following:

- A framework is defined to optimize arithmetic circuits for quantum computing.
- As a demonstration of the framework's effectiveness, it is tested with one of the most efficient reversible comparator circuits reported in the literature, obtaining a new version that reduces the number of qubits compared to previously published designs.
- The source code is made publicly available so that any interested researcher can use it to obtain optimized versions of their circuits [16].

The rest of the paper is organized as follows. Section II introduces the necessary concepts to understand the work. In particular, it explains what a quantum circuit is and how it should be measured. Section III presents the proposed framework. Subsequently, the theory behind comparator circuits is introduced together with their implementation in Section IV, which presents the main case study of this work. Section V extends the evaluation of the framework to other arithmetic circuits. Finally, the conclusions are presented.

II. Background

The most common way to programme a NISQ device is by using circuits [17], [18]. Similar to classical circuits, quantum circuits are built using gates. Quantum gates are the natural extension of logic gates to the quantum world. Every quantum gate can be represented mathematically as a complex unitary matrix [19]. Quantum gates operate on qubits, modifying their states.

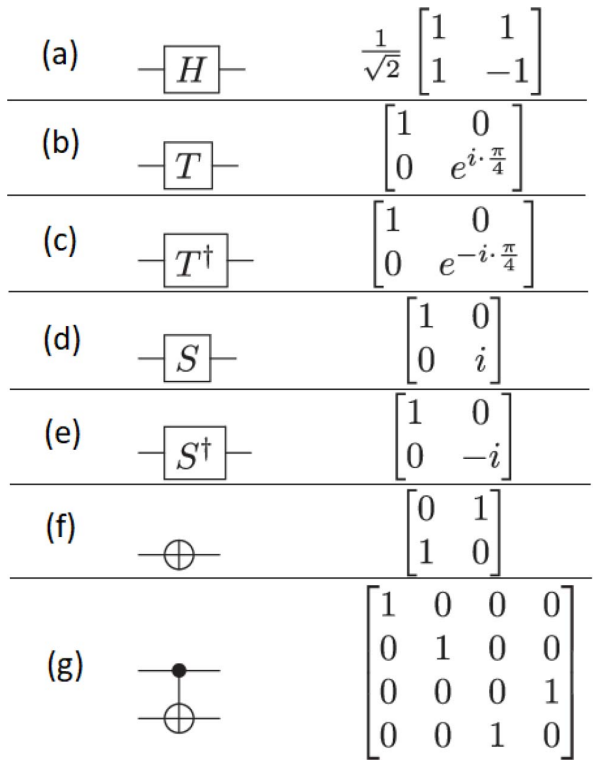

Fig. 1. Quantum gates of the Clifford+T group: (a) Hadamard gate, (b) T gate, (c) T adjoint gate, (d) S gate, (e) S adjoint gate, (f) Pauli-X gate, and (g) CNOT gate. For each gate, its symbol and matrix are shown.

In digital electronics, there are a limited number of logic gates. To work on a bit, there are only two possible logic gates: the identity gate and the NOT gate [20]. The former leaves the bit as it is, and the latter inverts it. In turn, there is a small set of gates that act on two bits: the AND gate, the OR gate, the XOR gate, and their negated versions [20]. In quantum computing, there are infinitely many possible quantum gates to work on a single qubit [1], [19]. Fortunately, there are sets of universal quantum gates that allow us to approximate all the others [19]. An example of a universal set is provided by the so-called Clifford+T group. Some of the gates belonging to the Clifford+T group are shown in Fig. 1.

When measuring a classical circuit, it is common to describe it in terms of its manufacturing cost, its execution time, its power consumption, or even the space it takes up [21]. It is easy to compare two different circuits if they have been previously measured using these metrics. If one circuit is faster than the other, it is better in these terms. However, the slower circuit could be cheaper, being more suitable than the other circuit in a cost-saving context. In fact, it is common to find designs optimized on one or several metrics that often sacrifice the rest of the metrics in favour of the priority metrics [22]. In quantum computing, there are also metrics to measure the goodness of circuits in certain aspects [23]. Mohammadi and Eshghi [24]

defined a framework for measuring quantum circuits that is widely used. This framework allows, in a simple way, to analyze and compare quantum circuits at a digital level, and is widely used in the literature [25], [26]. In particular, this paper proposes four parameters to measure and analyse the goodness of a circuit:

- Quantum cost: it gives us a rough idea of the complexity of a circuit. The quantum cost indicates the number of one or two-qubit gates that compose the circuit. There are some papers in the literature that use the number of gates (of any size) that compose a circuit as a metric [24], but this comparison can be misleading since there are quantum gates that are composed using other quantum gates. In the case of a gate of 3 or more qubits, its quantum cost will be given by the number of one and two-qubit gates that compose it.
- Delay: it is a metric related to the speed of the circuit. The speed of a quantum circuit is closely related to the technology used and the various implementations of both quantum gates and quantum qubits. Rather than measuring timing, what the delay does is to indicate the number of one and two-qubit gates that are sequentially executed on the critical path of the circuit. The delay of a multi-qubit gate will be determined by the number of one and two-qubit gates that must be executed sequentially. The delta unit ($\triangle$) is defined as the unit of delay.
- Number of qubits: it is trivial to point out that the fewer qubits a circuit needs, the more efficient it is.
- Garbage Outputs: any output of the circuit that meets the following two conditions is called a garbage output: 1) it does not contain a value that is part of the solution to the problem solved by the circuit (and is therefore useful for further operation), and 2) it does not contain the quantum state to which it was initialized at the start of the circuit. Qubits that are not restored to their original value cannot be used by subsequent circuits because their state is not known. With NISQ devices having limited resources, reverting these qubits to their original state is essential to properly utilise the available resources to build larger applications. To revert these qubits to their state, a suitable process must be performed that maintains the reversibility of the entire circuit, such as the one proposed by [27]. In short, this method consists in applying an inverse circuit. This circuit is exactly the same as the previous one, but all quantum gates are applied in the exact reverse order. The first gate will be the last, and viceversa. Although some quantum devices offer reset operations that are immediate, as in the case of IBM computers, these operations are not reversible and may cause errors if the circuits are to be embedded into other circuits.

As part of the framework and as an added contribution to this work, the repository offers a simple software that allows the automatic calculation of the Mohammadi and Eshghi metrics for any quantum circuit, as well as the ability to indicate the implementation each quantum gate uses in order to calculate metrics on any platform and in any implementation. It also identifies all the types of quantum gates a circuit uses, and the number of each type.

Although Mohammadi and Eshghi's framework does not explicitly include fault-tolerant metrics, such as the T-count and T-depth (the number of T gates and the longest sequence of T gates that must be executed sequentially in a circuit, respectively), these are implicitly reflected through the Clifford+T cost model adopted here [13]. Since the circuits considered target NISQ rather than fully error-corrected devices, practical metrics (quantum cost, delay, qubit count, and garbage outputs) remain more informative than explicit T-based figures. Nevertheless, the framework directly supports gate-level decomposition into Clifford+T primitives, allowing T-count and T-depth to be derived for comparison when required. In the extended analysis section, these metrics are computed where relevant to show consistency with recent fault-tolerant cost formulations.

Otherwise, all quantum cost and delay figures reported in this work are computed using the Clifford+T gate library. In particular, the Toffoli gate is evaluated according to the ancilla-free decomposition proposed by Amy et al. [28], which is the most commonly adopted reference implementation in the literature. Under our counting convention, this realization yields a quantum cost of 15 and a delay of 11 $\triangle$.

III. Proposed Framework

The methodology proposed is based on two main ideas. Firstly, it is suitable for quantum circuits whose operation is based on Boolean logic. This does not imply that they are not valid quantum circuits, since they support the usual features of superposition, entanglement, and the rest of quantum properties. But the algorithm they implement must be based on Boolean logic. Therefore, the main circuits that can benefit from this framework are arithmetic circuits: adders, subtractors, multipliers, dividers, and even comparators. Secondly, and as already mentioned, the idea proposed by Pérez-Salinas et al. [11] of introducing values to a circuit, not as states of the qubits, but as operations on them, is used. In other words, instead of assigning initial values to the qubits as is done in quantum arithmetic circuits currently published in the literature, the framework will provide circuits in which quantum gates are used throughout the circuit based on these values. It is usual in any quantum circuit to initialize the state of the qubits with quantum gates, but Pérez-Salinas et al. did not limit themselves to this, but reintroduce these values throughout the circuit. The main advantage is that it saves qubits. If done carefully, it is even possible to reduce the quantum cost of the circuit. In the case of Pérez-Salinas et al., it allowed them to implement a quantum classifier with a single qubit. Furthermore, Pérez-Salinas et al. justify that this reintroduction of data into quantum circuits is a natural way in quantum computing to compensate for the impossibility of copying values that exists in classical computing.

The circuit to be optimized must accept as inputs two bit strings A and B. For simplicity, let's assume that both strings have the same length N. The digits of B, that is, $b_{N-1}\cdots b_0$, are entered in a similar way as in the original circuit. That is, N qubits will be needed to represent B. However, to introduce A, the methodology of Pérez-Salinas et al. is used. Instead of using qubits initialized with the digits $a_{N-1}\cdots a_0$, such digits will

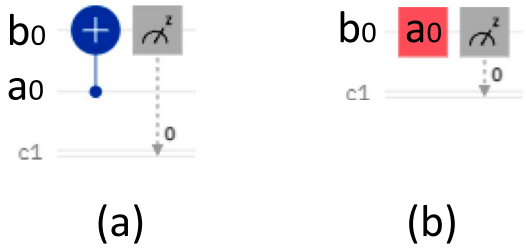

Fig. 2. (a) Operation $a_0 \oplus b_0$ using the CNOT gate. (b) The same operation, but using a single-qubit gate to represent a_0. This gate will be an identity gate if $a_0 = 0$, or a Pauli-X gate otherwise.

be applied by means of quantum gates. The simplest example of how we have adapted the idea of Pérez-Salinas et al. to an arithmetic context is shown in Fig. 2. Let us assume two bits a_0 and b_0. If we want to perform the operation $a_0 \oplus b_0$, the usual procedure in a quantum circuit is the one shown in Fig. 2(a). However, it is possible to perform this same operation using a single qubit, as shown in Fig. 2(b). Here, the red gate labelled a_0 will be a Pauli-X gate if $a_0 = 1$, or an identity gate (or simply an absence of gate) otherwise. Why these gates should be used and not others is explained just below. Otherwise, this simple example already shows the advantages and disadvantages of this methodology: the number of qubits to perform the operation has been reduced.

The target circuits must work with the bases $\{|0\rangle, |1\rangle\}$, which the user of the circuit interprets as the digits 0 and 1 respectively. Most quantum arithmetic circuits are mainly composed of CNOT and Toffoli gates. Simplistically explained, such gates assume that a rotation of the Pauli matrix (over X) is applied when one or two control qubits (for the CNOT and Toffoli gates, respectively) are in the $|1\rangle$ state, which is the state each qubit will be in when the digit it represents is set to 1. In the example shown in Fig. 2 it has been indicated that a Pauli-X gate should be used if $a_0 = 1$ precisely for this reason. Otherwise, either no gate should be used at all or, for the sake of clarity, an identity gate should be used. The example in Fig. 2 shows the simplest case, but multiple other operations can be simplified in this way. In this framework, in addition to the one mentioned in the example, these others operations are mainly used:

- Suppose the value $a_0 \oplus b_0$ from Fig. 2(b) is needed to, in turn, act as a control qubit on another qubit C. Once this operation is done, the same a_0 gate (a Pauli-X if $a_0 = 1$, or an identity otherwise) can be reapplied to revert the current state of the qubit back to contain b_0 if this value is needed to be used again.
- An auxiliary qubit can be reused several times to temporarily store different values. For instance, if at a given moment a Toffoli gate controlled by b_0 and a_0 is applied, and later a second Toffoli gate -this time controlled by b_0 and a_1- is needed, a_0 can be introduced as a gate in the auxiliary qubit, apply the Toffoli gate, revert the value of the auxiliary qubit by another a_0 gate, and finally introduce a_1 as a gate to be able to apply the second Toffoli gate.
- Two consecutive gates a_i and a_j in the same auxiliary qubit perform the operation $a_i \oplus a_j$.

Such operations can be performed on any quantum arithmetic circuit, being only necessary to adapt them to the programming language in which the circuits are coded before sending them to the quantum device. In the repository associated with this work, these operations are prepared to be applied on circuits written in Open Quantum Assembly Language (OpenQASM) [29], a programming language widely used by the quantum community. As an example, the circuit in Fig. 2(a) should be specified as follows:

```
qreg  q[2];
cx  q[0],  q[1];
```

Likewise, applying the provided software will result in the circuit in Fig. 2(b), also written in OpenQASM:

```
qreg  q[1];
id  q[0];
```

Note: In the OpenQASM code example, the `id` (identity) gate is used only as a symbolic placeholder to indicate that no operation is applied when the corresponding input bit is zero. This gate has no practical effect on the circuit state and may be safely omitted in actual implementations or simulations.

For simplicity, and to avoid possible confusion, the provided circuits should not indicate the initial value of the qubits, but this should be indicated as an argument through the software. If it is indicated that the two qubits in Fig. 2(a) are initially in the state $|0\rangle$, the circuit returned will be the one we have just shown as an example. However, if it is indicated that the first qubit (the one acting as a control) is initially in the $|1\rangle$ state, the returned circuit will be as follows:

```
qreg  q[1];
x  q[0];
```

The above-mentioned operations can always be performed taking into account that the original value of the qubit has not been modified. In the proposed software, simplification operations will not be applied when it is detected that the involved control qubit has previously participated in other operations (unless they are operations included in the framework itself). A common mistake would be to think that this limitation should not be established when such a qubit has only acted as a control qubit in other operations. However, phase kickback may be occurring [30]. It is up to the user to consider whether or not the simplification can be performed, in which case the user must manually modify the code to do so.

IV. Comparator Circuits and Case Study

This section applies the proposed framework to a concrete case study: the optimization of quantum comparator circuits. It first reviews the main principles and existing designs of comparators, then applies the framework to the state-of-the-art circuit of Li et al., and finally analyzes and compare the resulting optimized version in detail.

A. Comparator Circuits

To demonstrate the correct functionality of the framework, its application on a comparator circuit will be shown. To facilitate the understanding of the example, the operation of this type of circuit is briefly explained here. Comparison is a basic operation

TABLE I
TRUTH TABLE OF A TWO-DIGIT BINARY COMPARATOR. TWO TWO-BIT STRINGS A AND B ARE COMPARED, AND A TWO-BIT OUTPUT S IS PRODUCED WHICH WILL BE 00 WHEN A AND B ARE EQUAL, 01 WHEN $A < B$, AND 10 WHEN $A > B$

A_1	A_0	B_1	B_0	S_1	S_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	0

of any system that works with numbers. Given two bit strings A and B each representing a binary number, a comparator circuit returns the result of the comparison across two or three bits. In the case of two-bit operation, the result is returned as a binary number. For example, 00 or 11 if $A = B$ (and they are worth 0 or 1, respectively), 10 if $A > B$, and 01 if $A < B$ (Table I). However, to avoid having to interpret the result, it is usual to provide three outputs: one for each possible case. For example, for the case $A = 1$ and $B = 0$, the output corresponding to $A > B$ will be 1, and the remaining outputs will be 0.

There is a reduced version of these comparators, called half comparator. The half-comparator performs a similar function to the previous comparator (which we will call the full comparator from now on, to clearly distinguish the two). However, the half-comparator only identifies whether $A > B$ or not. That is, it will return 1 if $A > B$, or 0 if $A <= B$. The half-comparator is useful for a large number of applications where it is necessary to check whether a given value exceeds (or does not exceed) a threshold, without the need to distinguish the case $A = B$ from the case $A < B$ [31]. In addition, the half comparator has the advantage over the full comparator in that it requires fewer resources in the form of logic gates, resulting in a more economical and space-saving physical implementation, and a single bit output. This makes half-comparators attractive for quantum computing, being used to a greater extent than full-comparators [32].

Focusing on quantum computing, the concept of a (half) comparator is exactly the same as the one described so far: given two numbers A and B, a circuit is required to determine whether $A > B$ or not. It is important to note that quantum comparators are not comparing quantum states, an operation that is undoubtedly more complex, but rather looking for a way to represent binary digits using qubits, and return an output that can be interpreted through a measurement or by another quantum circuit as a potential input. All of this complies with the requirements imposed by quantum mechanics, as all quantum circuits do. Obtaining an efficient quantum comparator for the general case of N-bit numbers is not trivial, as the large number of alternatives in the literature demonstrates [33], [34], [35], [36], [37].

TABLE II
COMPARISON IN TERMS OF QUANTUM COST, DELAY, NUMBER OF QUBITS, AND NUMBER OF GARBAGE OUTPUTS, BETWEEN THE BEST COMPARATORS PUBLISHED IN THE LITERATURE

Circuit	Quantum cost	Delay ($\triangle$)	Qubits	Garbage outputs
[33]	$38N + 1$	$26N$	$2N + 2$	0
[34]	$39N + 1$	$28N + 3$	$2N + 2$	0
[36]	$36N - 20$	$26N - 14$	$2N + 1$	0

Table II shows the most important works that have participated in the race to achieve the most efficient comparator in terms of qubits. Considering only reversible circuits, the first circuit to achieve the best result so far in terms of qubits was the circuit by Li et al. [36], with $2N + 1$ qubits. Other circuits have appeared since then, proposing different improvements for specific situations (quantum cost reduction, T-count reduction, etc.) [37], [38], [39], improvements for specific circuits [40], or alternative methodologies [41]. However, none of them (at least not reversibly) has managed to reduce the number of qubits achieved by Li et al.

The circuit of Li et al. was selected as our primary case study because it best matches the optimization objectives of this work under NISQ constraints. Moreover, Li et al.'s circuit is composed solely of Boolean-logic primitives (CNOT/Toffoli) and is specified in sufficient detail, which enables a faithful, gate-level application of our framework and a fair comparison of figures of merit. The following subsection applies the proposed optimization framework to it.

B. Optimized Comparator Design

As mentioned previously, the circuit of Li et al has been chosen as a case study. It will be demonstrated that even this circuit, one of the best comparators in terms of qubits, is susceptible to be improved by our framework, obtaining a better circuit.

Fig. 3 shows the comparator of Li et al. [36]. The circuit can be built to compare strings of any bit size N, although in the example it is shown for the specific case of $N = 4$. As discussed in Subsection IV-A, this comparator allows to determine whether a string A is larger than another string B, or not. It can be seen in Fig. 3 that 4 qubits are used to encode the bits of A, and another 4 qubits to encode the bits of B. Moreover, an auxiliary qubit is involved. In the general case, the circuit will need N qubits to encode A, N qubits to encode B, and a single qubit for auxiliary operations. The right half of the circuit is focused on reversing the garbage outputs, as explained in Subsection II. Otherwise, this is a circuit built exclusively with CNOT and Toffoli gates whose operation is based on Boolean logic. It is a candidate to be optimized by the proposed framework.

Applying the operations explained in the previous Section, the circuit shown in Fig. 4 will be obtained. This circuit requires

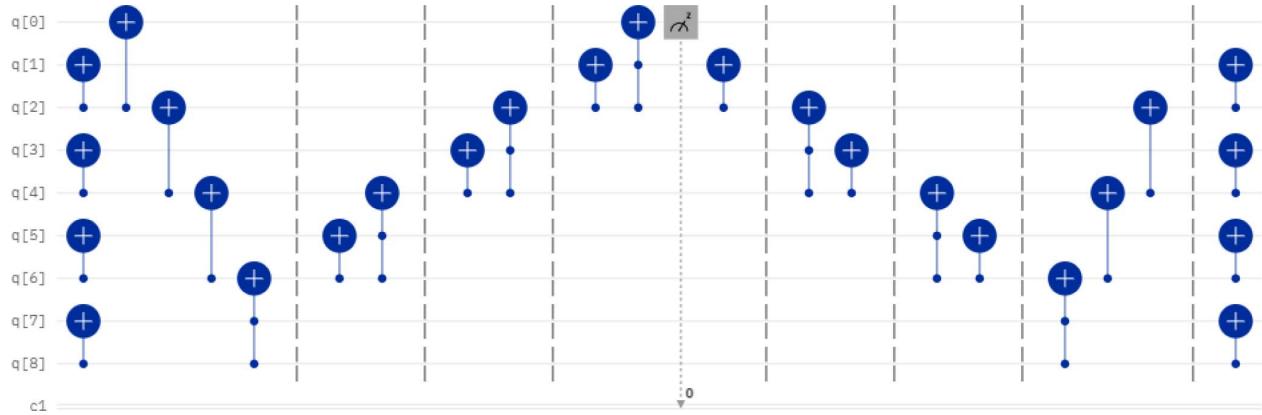

Fig. 3. Circuit proposed by Li et al. [36], for the case $N = 4$. The $q[0]$ qubit must be initialized to 0. The odd qubits will contain the digits of B (from $q[1] = B_{N-1}$ to $q[7] = B_{N-1}$). The even qubits will contain the digits of A (from $q[2] = A_{N-1}$ to $q[8] = A_{N-1}$). the circuit starts the comparison from the most significant digits instead of the least significant ones.

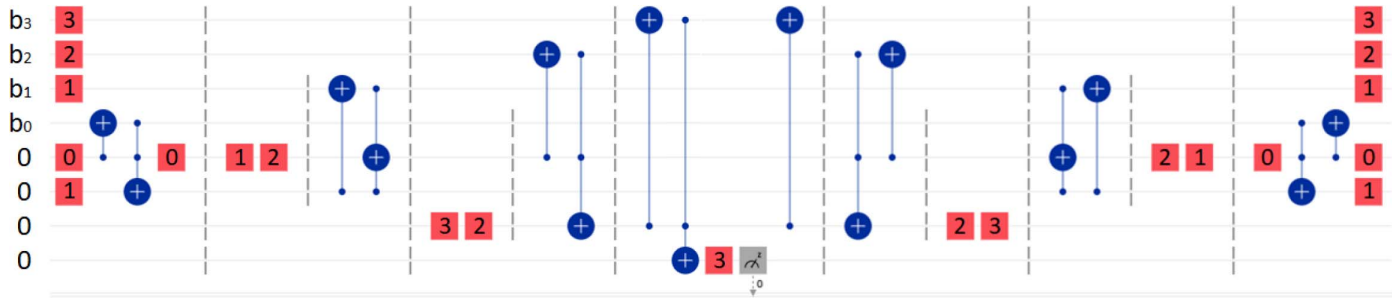

Fig. 4. Optimized comparator, for the case $N = 4$. it can be seen at a glance that it reduces the number of qubits required with respect to the original comparator. The digits of B are represented using the qubits marked as b_i. the digits of A are introduced by means of quantum gates (indicated in red) in the implementation of the circuit itself. The number of each red gate indicates the index of the digit of A. for example, a number 3 refers to a_3. This gate will be a Pauli-X gate if that digit is a 1, or an identity gate (or no gate) otherwise.

one less qubit than the previous one. In this circuit, string B is encoded in a similar way to the original circuit (Fig. 3). However, as discussed in the previous Section, the bits of A are introduced directly as operations on the qubits of B (the digits of A are indicated as red gates whose number i refers to the position a_i). The original circuit is composed of 19 CNOT gates and 7 Toffoli gates, as well as the aforementioned 9 qubits. The optimized circuit is composed of between 0 and 21 Pauli-X gates (depending on the number of 1's in A), 8 CNOT gates, and the same 7 Toffoli gates as the original, plus one less qubit than the circuit of Li et al.

For the sake of clarity, the steps to build the optimized circuit for any digit size N are given below, directly integrating the framework operations into the original algorithm:

1) To prepare $2N$ qubits. Of these, N will be used to encode the digits of B ($b_{N-1} \cdots b_0$), and the rest will be initialised to $|0\rangle$ and used as auxiliary qubits. The qubits containing the digits of B will be in the $|0\rangle$ state if the digit it represents is 0, and in the $|1\rangle$ state otherwise. No further qubits will be needed to implement the comparator.
2) To apply a Pauli-X gate on the first auxiliary qubit if $a_0 = 1$, or an identity gate if $a_0 = 0$. From now on, this type of operation, in which we apply a Pauli-X gate if $a_i = 1$ or an identity gate if $a_i = 0$, will be labelled as a_i gate. We will also apply an a_1 gate on the second auxiliary qubit, and an a_j gate on each of the qubits used to encode B except for b_0 (i.e. from $j = N - 1$ to 1). Next, a CNOT gate is applied on the first auxiliary qubit and b_0 to perform the operation $a_0 \oplus b_0$, and a Toffoli gate is applied on b_0 and the two auxiliary qubits used so far to perform the operation $(a_0 \oplus b_0)a_0 \oplus a_1$. Finally, a gate a_0 is applied on the first auxiliary qubit to reverse (uncompute) it.
3) On the auxiliary qubit that was previously uncomputed, to apply a gate a_1 and a gate a_2, so that the operation $a_1 \oplus a_2$ is implemented. To apply a CNOT gate whose control qubit is the second auxiliary qubit and whose target one is b_1, and finally a Toffoli gate whose control qubits are the qubits involved in the CNOT, and whose target qubit is the one containing $a_1 \oplus a_2$.
4) From $i = 2$ to $i = N - 2$, to repeat:
   - On a previously unused auxiliary qubit, to perform the operation $a_{i+1} \oplus a_i$ by applying two gates a_{i+1} and a_i.
   - To apply a CNOT gate, whose control qubit is the target one of the last Toffoli gate applied, and whose target qubit (of the CNOT gate) is b_i.

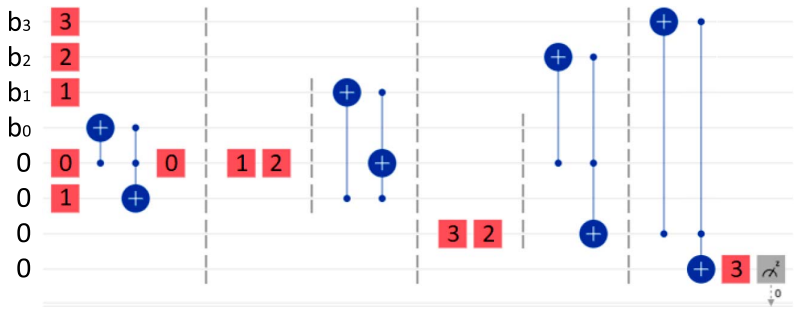

Fig. 5. Proposed circuit, with unreverted garbage outputs, for the case $N = 4$. The measured qubit contains the result, while the rest of the qubits contain undefined values that prevent their reuse without performing reset operations (thus making the circuit non-reversible).

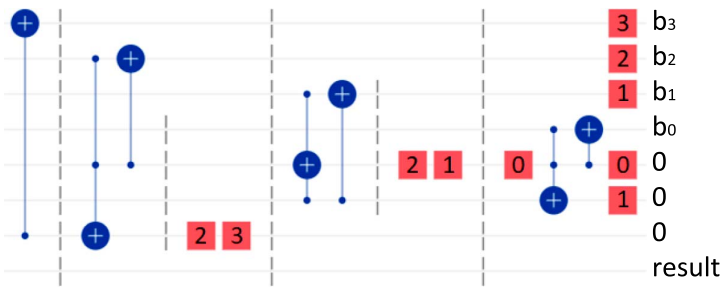

Fig. 6. Proposed circuit to reverse the circuit of Fig. 5 to avoid the presence of garbage outputs.

- Finally, to apply a Toffoli gate whose control qubits are the two qubits involved in the previous CNOT, and whose target qubit is the auxiliary qubit containing operation $a_{i+1} \oplus a_i$.

5) In the last step to build the circuit, a CNOT gate must be applied, its control qubit being the target qubit of the last Toffoli gate applied, and whose target qubit (of the CNOT gate) is b_{N-1}. Subsequently, a Toffoli gate is applied whose control qubits are the two qubits involved in the previous CNOT, and whose target qubit is the only auxiliary qubit that has not yet been used. On this qubit, the operation a_{N-1} is applied. This qubit will contain the result of the comparator.

The circuit obtained after these steps presents garbage outputs that must be reversed to uncompute such qubits (Fig. 5). To reverse the garbage outputs, it is enough to apply Bennett's scheme [27] to the circuit obtained as mentioned in Subsection II, respecting the qubit containing the result (Fig. 6).

C. Analysis of the Optimized Circuit

In order to analyse the number of qubits needed by the proposed circuit, as well as its quantum cost, delay, and presence (or not) of garbage outputs, the algorithm described in the previous section for the implementation of such a circuit will be analysed step by step. The circuit evaluation code included in the repository, mentioned in Subsection II, can also be used.

- In step 1 it is stated that $2N$ qubits are needed, of which N are used to encode B, and the other N are for auxiliary purposes. Therefore, the proposed circuit needs $2N$ qubits, of which N are auxiliary qubits.
- In step 2, two Pauli-X gates or two identity gates are used on two auxiliary qubits (the quantum cost of each of these is 1, and their delay is $1\triangle$), $N-1$ Pauli-X or identity gates on the qubits of B, $N-1$ Pauli-X or identity gates on the qubits of B, a CNOT gate (quantum cost 1, delay $1\triangle$), a Toffoli gate (quantum cost 15, delay $11\triangle$), and a Pauli-X or identity gate (quantum cost 1, delay $1\triangle$). The quantum cost of this step is $1+1+(N-1)+1+15+1 = N+18$. The delay is $1\triangle + 1\triangle + 11\triangle + 1\triangle = 14\triangle$ (all gates a_i, except the last one, can be computed in parallel).
- In step 3, two Pauli-X or identity gates, a CNOT gate, and a Toffoli gate are used. The quantum cost is $1+1+1+1+15 = 18$, and the delay is $1\triangle + 1\triangle + 1\triangle + 11\triangle = 14\triangle$.
- In step 4, $N-3$ iterations are performed. Each of these iterations involves two Pauli-X or identity gates, a CNOT gate, and a Toffoli gate. The quantum cost per iteration is $1+1+1+15 = 18$. The delay per iteration is $1\triangle + 1\triangle + 11\triangle + 1\triangle = 14\triangle$. Given $N-3$ iterations, the quantum cost will be $18N-54$, and the delay $(14N-42)\triangle$.
- In step 5, a CNOT gate, a Toffoli gate, and a PauliX or identity gate are applied. The quantum cost is $1+15+1 = 17$, and the delay is $1\triangle + 11\triangle + 1\triangle = 13\triangle$.

The quantum cost of the circuit at this point is $N+18+18+18N-54+17 = 19N-1$. The delay is $14+14+14N-42+13 = (14N-1)\triangle$. However, the garbage outputs have not yet been reversed and the circuit has $2N-1$ garbage outputs. To uncompute the garbage outputs, the inverse circuit must be run, which assumes:

- Reversing step 5 requires only one CNOT gate. It assumes a quantum cost and delay of 1 and $1\triangle$, respectively.

TABLE III
METRICS OF THE PROPOSED CIRCUIT, WITHOUT REVERSING THE GARBAGE OUTPUTS, AND WITH THE GARBAGE OUTPUTS ALREADY REMOVED. THE QUANTUM COST, THE DELAY, IN TOTAL NUMBER OF CUBITS, AND THE NUMBER OF GARBAGE OUTPUTS ARE SHOWN. REVERSING THE GARBAGE OUTPUTS HAS A HIGH COST IN TERMS OF QUANTUM COST AND DELAY, BUT ALLOWS $2N-1$ CUBITS TO BE FREED. IN BOTH CASES, THE WORST-CASE SCENARIO IS CONSIDERED: THE STRINGS TO BE COMPARED ARE ALL ONES

Circuit	Quantum cost	Delay ($\triangle$)	Qubits	Garbage
Original	$19N-1$	$14N-1$	$2N$	$2N-1$
Uncomputed	$38N-18$	$28N-14$	$2N$	0

TABLE IV
QUANTUM COST, DELAY, TOTAL NUMBER OF CNOTS AND TOTAL NUMBER OF QUBITS OF THE ORIGINAL CIRCUIT AND ITS OPTIMIZATION. THE NUMBER OF GARBAGE OUTPUTS IN BOTH CASES IS ZERO. THE QUANTUM COST AND DELAY OF THE OPTIMIZED CIRCUIT DEPENDS ON THE NUMBER OF ONES IN THE STRING A, SO RANGES ARE SHOWN WITH THE MINIMUM AND MAXIMUM POSSIBLE VALUES

Circuit	Quantum cost	Delay ($\triangle$)	Qubits	CNOT
Li et al.	$36N-20$	$26N-14$	$2N+1$	$6N-5$
Optimized	$[32N-24,$ $38N-18]$	$[22N-18,$ $28N-14]$	$2N$	$2N$

- Step 4 must be repeated entirety, but in reverse order. A quantum cost of $18N-54$ and an extra delay of $(14N-42)\triangle$ are added.
- Step 3 must also be executed in reverse order. It involves a quantum cost of 18, and a delay of $14\triangle$.
- Step 2, in reverse, adds an extra quantum cost of $N+18$, and a delay of $14\triangle$.
- No action is required to reverse step 1. Therefore, no extra costs are added in this step.

The quantum cost of the inverse circuit is $1+18N-54+18+18+N+18=19N-17$. Its delay is $1+14N-42+14+14+14=14N-13$. Such values must be added to the above, giving a total quantum cost of $38N-18$, and a total delay of $(28N-14)\triangle$. However, if instead of using identity gates when necessary (i.e. for those digits of A that are worth 0) we simply do not use any quantum gates at all, these values can be reduced (the identity gate count would have to be eliminated in such cases). On the other hand, the circuit has no garbage outputs, and $2N$ qubits were required, of which N are auxiliary. For better visualisation, the metrics of the original circuit (with garbage outputs) and the uncomputed version (with all the qubits already reverted) are shown in Table III.

As shown in Table III, the uncomputed version effectively doubles both the quantum cost and the delay while maintaining the same qubit count. This highlights the core trade-off between circuit compactness and reversibility: applying Bennett's uncomputation removes all garbage outputs and restores ancilla qubits, but at the expense of roughly doubling the circuit depth and two-qubit gate exposure. In practice, whether this trade-off is favorable depends on the target hardware. For NISQ devices, where qubit availability is the dominant limitation, the additional temporal cost is acceptable and the qubit saving outweighs the increased depth [7]. Conversely, in platforms constrained by coherence time or two-qubit gate fidelity, partial or deferred uncomputation strategies may offer a better balance between reversibility and noise resilience [42].

D. Comparison

In order to provide a useful comparison, the proposed circuit is compared to the circuit of Li et al. Since such a circuit reverses its garbage outputs (with the consequent associated cost), the proposed circuit with no garbage outputs is used in the comparison.

TABLE V
RESOURCE METRICS OF THE ORIGINAL AND OPTIMIZED COMPARATOR CIRCUITS FOR SEVERAL INPUT SIZES N. FOR EACH CASE, THE QUANTUM COST, DELAY, AND NUMBER OF CNOT GATES ARE REPORTED. THE OPTIMIZED VERSION CONSISTENTLY ACHIEVES LOWER RESOURCE USAGE WHILE PRESERVING CIRCUIT FUNCTIONALITY. THE NUMBER OF QUBITS IS NOT SHOW AS THE OPTIMIZED CIRCUIT ALWAYS REDUCES IT IN 1

N	Circuit	Quantum cost	Delay ($\triangle$)	CNOTs
2	Li et al.	52	38	7
	Optimized	[38, 56]	[26, 42]	4
4	Li et al.	124	90	19
	Optimized	[102, 132]	[70, 98]	8
8	Li et al.	268	194	43
	Optimized	[230, 284]	[158, 214]	16
12	Li et al.	412	298	67
	Optimized	[358, 436]	[246, 330]	24
16	Li et al.	556	402	91
	Optimized	[486, 588]	[334, 446]	32

In terms of the number of qubits is where the optimized circuit excels the version of Li et al.: it only needs $2N$ qubits to implement a comparison between N-bit strings. It is the only reversible comparator currently available in the quantum literature that requires this number of qubits. Achieving such a number of qubits was not easy, as using binary qubit encoding is not possible for reversibility reasons. It has been made possible by using an alternative encoding methodology.

In terms of quantum cost and delay, the result is uncertain since it will depend on the number of 1-valued bits in the A string. However, the optimized circuit reduces the number of CNOT gates to less than half the number required by the original. In a context where controlled gates introduce significantly more noise than gates acting on a single qubit [42], this is also an important achievement to note.

The result of the comparison is shown in Tables IV and V. In addition to these metrics, it should be noted that the proposed architecture not only improves the total number of qubits required, but also presents better modularity for integration into more complex quantum circuits. By not relying on qubits specifically dedicated to represent each input bit, the circuit structure allows greater flexibility to be reused as a subcomponent in broader quantum algorithms, such as classifiers, conditional functions or logic control units. This feature is especially relevant in quantum image processing applications, where multiple binary comparisons must be performed in parallel and

TABLE VI
T-COUNT, T-DEPTH, AND TOTAL NUMBER OF QUBITS OF THE ORIGINAL CIRCUIT AND ITS OPTIMIZATION. $W(N)$ REPRESENTS THE NUMBER OF ONES IN THE BINARY EXPRESSION. THE OPTIMIZED CIRCUIT MAINTAINS THE SAME T-COUNT BUT REDUCES THE NUMBER OF QUBITS BY $N-1$. HOWEVER, T-DEPTH CHANGES FROM LOGARITHMIC TO LINEAR

Circuit	T-count	T-depth	Qubits
Thapliyal et al. [43]	$40N-11W(N)-11log(N)-$ $-11W(N-1)-11log(N-1)-32$	$3log(N)+3log(N-11)+3log(N/3)+$ $+3log(N+1/3)+42$	$4N-W(N)-log(N)+1$
Optimized	$40N-11W(N)-11log(N)-$ $-11W(N-1)-11log(N-1)-32$	$3(N+1)$	$3N-W(N)-log(N)+2$

efficiently. The reduction in the number of controlled gates not only minimizes the quantum cost, but also has a positive impact on the overall fidelity rate of the circuit, improving the probability of obtaining correct results when running on real quantum hardware. These improvements make the optimized comparator an ideal candidate for practical implementation in NISQ devices, where every qubit and every operating cycle are extremely valuable resources.

E. Average-Case Analysis (Uniform and by Hamming Weight)

For the sake of clarity, this subsection examines the average case. The resources of the optimized comparator depend on the bit pattern of the fixed operand A only through the number of embedded single-qubit operations (a_i gates, realized as $\mathtt{x}$ when $a_i=1$ and omitted otherwise). Counting multiplicities over the forward circuit plus Bennett uncomputation gives $2\,\mathbb{I}[a_0=1]+3\sum_{k=1}^{N-1}\mathbb{I}[a_k=1]=3H-\mathbb{I}[a_0=1]$, where $H=\sum_{k=0}^{N-1}\mathbb{I}[a_k=1]$ is the Hamming weight of A. Adding the fixed one and two-qubit costs (CNOT/Toffoli) yields affine expressions in H for the garbage-free design

$$\mathrm{QC}(H,a_0)=32N-24+6H+4\,\mathbb{I}[a_0=1], \quad (1)$$

$$\Delta(H,a_0)=22N-18+6H+2\,\mathbb{I}[a_0=1]. \quad (2)$$

These formulas interpolate exactly the limits shown in Table IV: the best case ($H=0$, $a_0=0$) yields $32N-24$ and $22N-18$, while the worst case ($H=N$, $a_0=1$) gives $38N-18$ and $28N-14$. Under the uniform input model, where the bits of A are independent Bernoulli$(1/2)$ variables, we have $\mathbb{E}[H]=N/2$ and $\mathbb{E}[\mathbb{I}[a_0=1]]=1/2$. Therefore, the expected costs of the optimized garbage-free comparator are

$$\mathbb{E}[\mathrm{QC}]=32N-24+6\cdot\frac{N}{2}+4\cdot\frac{1}{2}=35N-22, \quad (3)$$

$$\mathbb{E}[\Delta]=22N-18+6\cdot\frac{N}{2}+2\cdot\frac{1}{2}=25N-16. \quad (4)$$

These expectations fall midway between the best and worst case bounds, consistent with the linear dependence on H. For stratified analysis by Hamming weight, the above expressions can be evaluated for $H=0,1,\ldots,N$, optionally averaging $\mathbb{I}[a_0=1]$ to $1/2$ within each class if the least significant bit is unbiased. For the additional arithmetic modules presented in Section V (adder, multiplier, and divider subcircuits), the reference metrics (T-count, T-depth, and qubit count) are invariant with respect to the bit pattern of the fixed operand in both the original and optimized versions, and therefore their worst-case and average-case values coincide.

V. EXTENDED EVALUATION: APPLICABILITY TO OTHER ARITHMETIC CIRCUITS

To demonstrate the general applicability of the proposed optimization framework beyond the comparator example, this section presents its application to several representative arithmetic circuits: an adder, a multiplier, a subtractor, and a divider. Each circuit has been evaluated using the same performance metrics adopted in its corresponding reference work, ensuring a fair and consistent comparison with previously published designs. In all cases, the proposed framework achieves reductions in qubit requirements while maintaining, or in some cases improving, the original figures of merit defined by each author.

A. Quantum Adder

The first validation case of the proposed framework corresponds to the quantum carry lookahead adder (QCLA) introduced by Thapliyal et al. [43]. This adder was originally proposed as a low-depth design for NISQ devices, optimized for the T gate count, which represents the most costly resource in Clifford+T-based quantum architectures. Following the same evaluation criteria as the original work, the circuit performance is assessed in terms of *T-count, T-depth*, and total number of qubits.

Figs. 7 and 8 show, respectively, the original and the optimized versions of the QCLA for the case $N=4$. Table VIII summarizes the comparison between both versions of the adder. The proposed optimization maintains the same T-count expression, ensuring that the number of fault-tolerant T gates remains unchanged, while reducing the qubit cost by $(N-1)$. However, this reduction in spatial resources leads to a change in the circuit depth behavior: the original logarithmic T-depth becomes linear with respect to N. This trade-off is acceptable for small and medium NISQ circuits, where qubit availability is typically the main limiting factor, and demonstrates the flexibility of the proposed framework in balancing space–time resources.

B. Quantum Subtractor

Quantum subtraction circuits can be derived directly from addition circuits. In general, a subtractor can be implemented either by adding the two's complement of one operand to the other or by modifying the carry logic of an adder to propagate borrow bits instead of carries. In this work, subtraction is treated as a natural extension of the adder optimization: the same embedding strategy is applied to fix one operand as gate parameters, thus reducing the number of required qubits

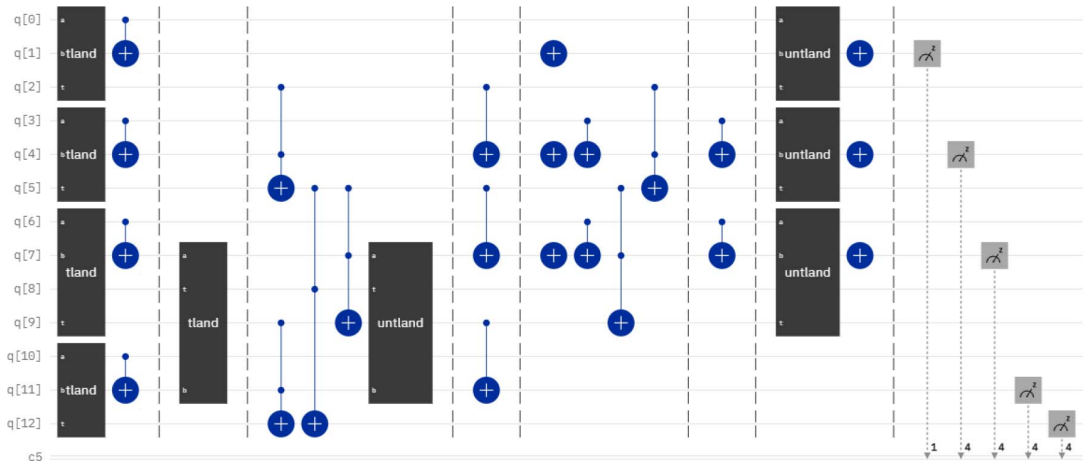

Fig. 7. Adder proposed by Thapliyal et al. [43], for the case $N = 4$.

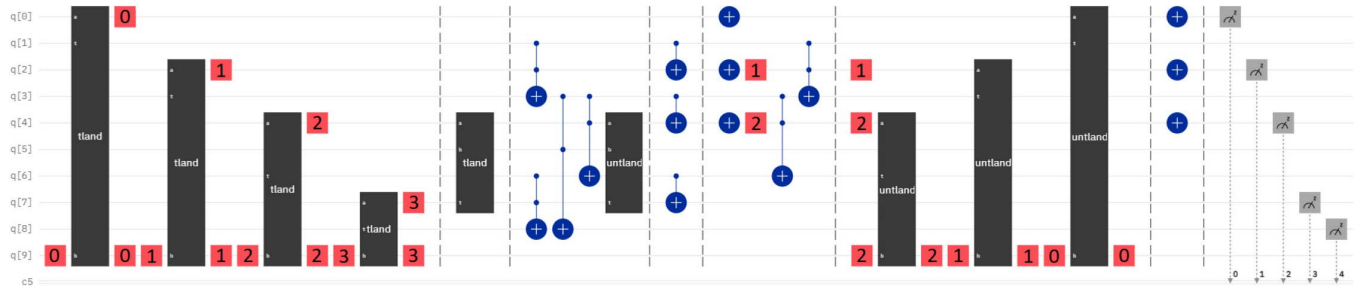

Fig. 8. Proposed adder, for the case $N = 4$.

without altering the arithmetic functionality. This demonstrates that the proposed framework is not limited to addition, but can be seamlessly adapted to subtraction-based quantum operations using the same optimization principles.

C. Quantum Multiplier

The next validation case corresponds to the multiplier circuit proposed by Orts et al. [44], which defines a Wallace-tree integer multiplier built entirely using Clifford+T gates. The original design aims to minimize the T-count and T-depth while maintaining compatibility with fault-tolerant quantum error correction codes. Following the same evaluation criteria as the original work, both the original and optimized circuits are compared in terms of *T-count, T-depth*, and total number of qubits.

Figs. 9 and 10 illustrate the Toffoli array used in the first stage of the multiplier, where the partial products are generated. The proposed optimization focuses on this part of the circuit, embedding one of the multiplicand registers as gate parameters and thus eliminating the need for $(N-1)$ qubits within the CCNOT array. Further qubit savings are achieved by applying different optimization techniques to the preceding initialization layers of the circuit.

Table VII summarizes the results obtained for both versions. The optimized circuit preserves the same *T-count and T-depth* as the original circuit ($18N^2 - 24N$ and $14N - 14$, respectively), while reducing the number of required qubits by $(N-4)$.

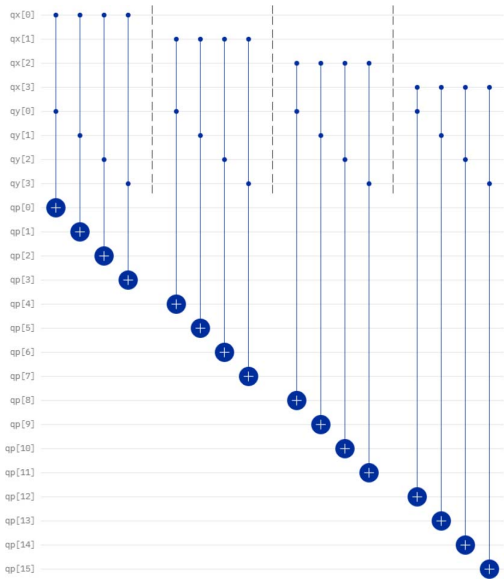

Fig. 9. Toffoli array as it is used by Orts et al. [44], for the case $N = 4$.

D. Quantum Divider

The last validation case corresponds to the fault-tolerant quantum divider proposed by Yuan et al. [45], which implements a binary long-division algorithm using the Clifford+T gate set. The divider operates iteratively, and each iteration requires three subcircuits: the comparator (*UC*), the

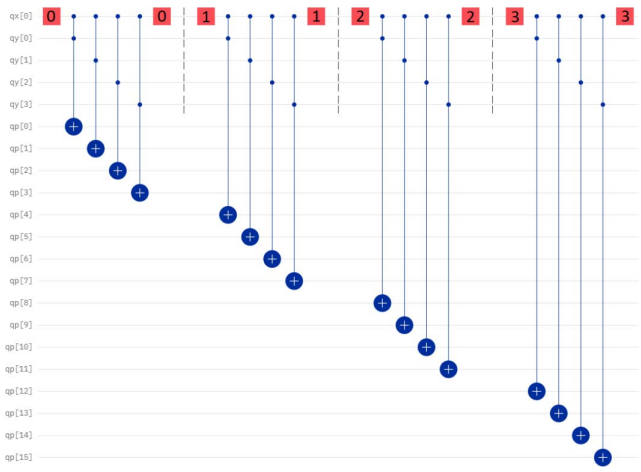

Fig. 10. Proposed Toffoli array, for the case $N = 4$.

TABLE VII
T-COUNT, T-DEPTH, AND TOTAL NUMBER OF QUBITS OF THE ORIGINAL CIRCUIT AND ITS OPTIMIZATION. M IS THE NUMBER OF QUBITS REQUIRED TO SAVE THE RESULT. THE OPTIMIZED CIRCUIT MAINTAINS THE SAME T-COUNT AND T-DEPTH METRICS, BUT REDUCES THE NUMBER OF QUBITS BY $N - 4$

Circuit	T-count	T-depth	Qubits
Orts et al. [44]	$18N^2 - 24N$	$14N - 14$	$2N^2 + N + (M - 2)$
Optimized	$18N^2 - 24N$	$14N - 14$	$2N^2 + (M - 6)$

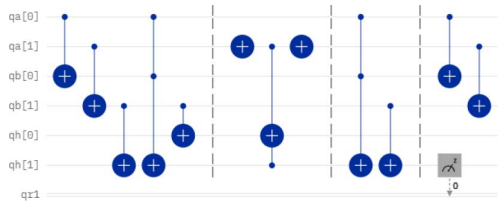

Fig. 11. UC proposed by Yuan et al. [45], for the case $N = 2$.

equal-bit subtractor (*SUB*), and the unequal-bit subtractor (*NSUB*). In every loop, the comparator determines whether the current partial dividend is greater than the divisor, and, depending on the result, either the SUB or NSUB circuit is executed to update the remainder and the quotient bit. Figs. 11–16 show the original and optimized versions of these subcircuits, for an example of $N = 2$.

The optimization of this divider follows the same principles established in previous cases. Each module (UC, SUB, and NSUB) is modified by embedding one of the operands into the gate structure, which allows the reuse of qubits across consecutive iterations. As shown in Table VIII, this approach leads to a reduction in the number of qubits in all submodules while preserving the original arithmetic functionality and reversibility. Specifically, the UC, SUB, and NSUB modules reduce their qubit requirements from $2N + 2$, $2N + 4$, and $2N + 5$ to $N + 3$, $N + 5$, and $N + 6$, respectively, with no variation in T-count or T-depth.

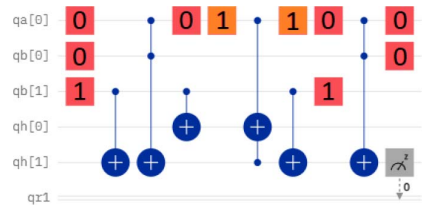

Fig. 12. Proposed UC circuit, for the case $N = 2$. The orange gates show inverse logic: They should be applied when $a[1]$ is 0.

TABLE VIII
NUMBER OF CNOT GATES, T COUNT, T DEPTH, AND NUMBER OF QUBITS FOR EACH OF THE SUBCIRCUITS REQUIRED FOR THE DIVISOR OF YUAN ET AL. [45], AND THEIR RESPECTIVE IMPROVED VERSIONS. THE ROW DEDICATED TO THE DIVIDER ITSELF DESCRIBES A SINGLE ITERATION OF THE CIRCUIT, WITH A TOTAL OF (N-M+1) ITERATIONS TO BE PERFORMED, WHERE N IS THE NUMBER OF BITS IN THE DIVIDEND AND M IS THE NUMBER OF BITS IN THE DIVISOR. AN ITERATION CONSISTS OF A UC CIRCUIT, A SUB CIRCUIT, AND A NSUB CIRCUIT, EXECUTED IN SERIES AND REUSING QUBITS, IN ADDITION TO TWO ADDITIONAL CNOT GATES AND ONE TOFFOLI GATE

Circuit	CNOT gates	T-count	T-depth	Qubits
UC	$4N - 1$	$10N - 5$	$3N - 3$	$2N + 2$
UC Optimized	$2N - 1$	$10N - 5$	$3N - 3$	$N + 3$
SUB	$4N - 1$	$14N$	$6N$	$2N + 4$
SUB Optimized	$2N - 1$	$14N$	$6N$	$N + 5$
NSUB	$4N$	$14N$	$6N$	$2N + 5$
NSUB Optimized	$2N$	$14N$	$6N$	$N + 6$
Divider*	$12N$	$38N + 2$	$15N$	$2N + 5$
Divider Optimized*	$6N$	$28N$	$12N$	$N + 6$

Each divider iteration consists of one UC, one SUB, and one NSUB circuit, executed sequentially with partial qubit reuse. The cumulative cost of a single iteration, shown in the last rows of Table VIII, indicates that the optimized version achieves a reduction from $(2N + 5)$ to $(N + 6)$ qubits, together with lower CNOT count and overall quantum cost. Moreover, both T-count and T-depth are slightly reduced (from $38N + 2$ to $28N$ and from $15N$ to $12N$, respectively), which demonstrates that the framework can improve spatial and temporal efficiency simultaneously.

To end this Section, it is important to consider how hardware connectivity constraints (such as linear or grid coupling maps) may affect the physical efficiency of the proposed circuits. Although all-to-all connectivity is assumed in our analytical evaluation, nearest-neighbor architectures would require additional SWAP operations to maintain logical adjacency [1], [7]. Nevertheless, since the proposed framework reduces both the total qubit count and the number of two-qubit gates at the logical level, the number of required SWAP insertions remains proportionally lower than in the original circuits. Therefore, the relative improvements demonstrated by the framework are expected to hold across different hardware topologies.

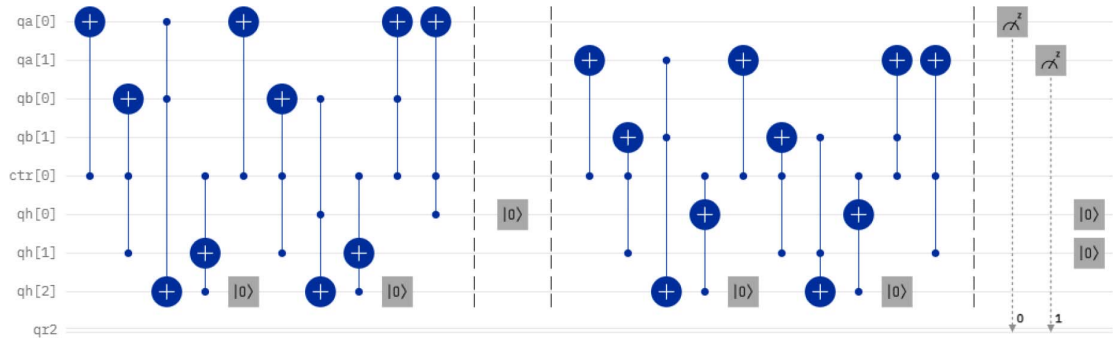

Fig. 13. SUB proposed by Yuan et al. [45], for the case $N = 2$.

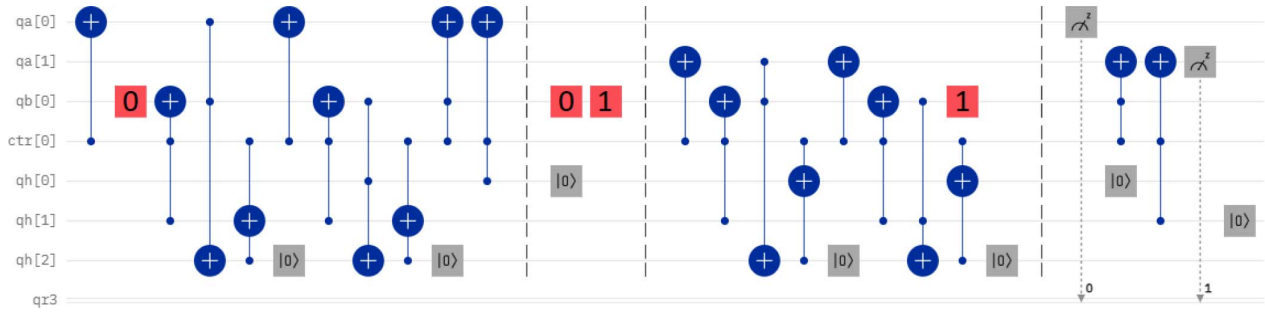

Fig. 14. Proposed SUB circuit, for the case $N = 4$, for the case $N = 2$.

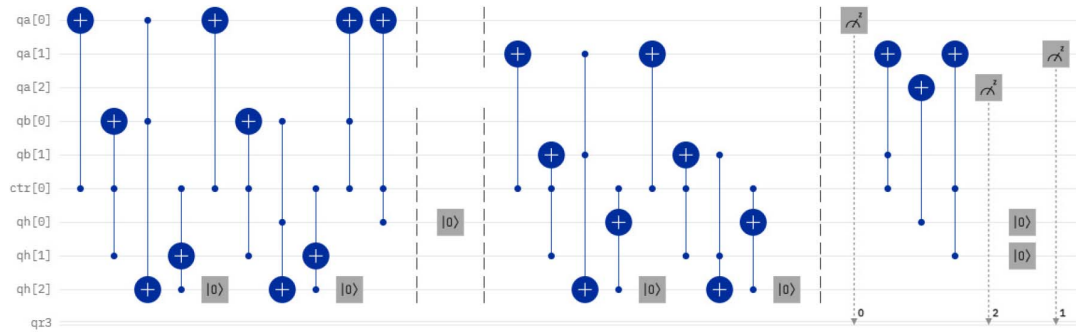

Fig. 15. NSUB proposed by Yuan et al. [45], for sizes $N = 2$ and 3.

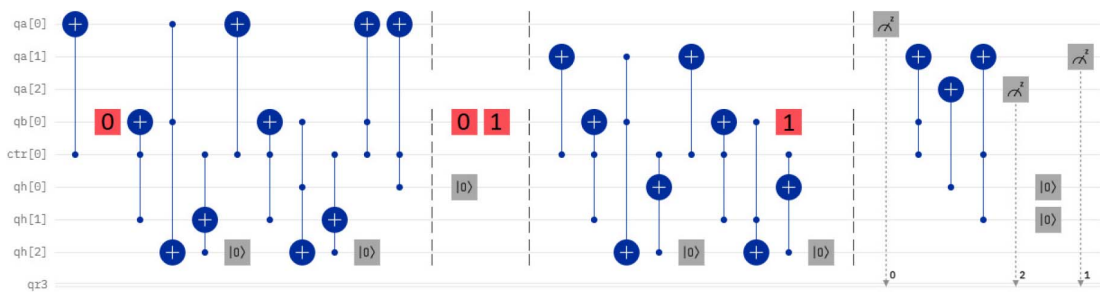

Fig. 16. Proposed NSUB circuit, for sizes $N = 2$ and 3.

VI. Conclusion

This work has introduced a novel framework aimed at optimizing quantum arithmetic circuits, focusing specifically on the reduction of qubit usage and the minimization of noise-inducing controlled operations. Unlike traditional quantum circuits, where inputs are typically encoded directly in the state of qubits, the proposed methodology employs a hybrid representation. Inspired by Pérez-Salinas et al., it reintroduces input data as quantum operations applied throughout the circuit, thereby avoiding unnecessary qubit initialization. This paradigm shift results in circuits that are not only more compact but also potentially more robust in noisy quantum environments.

The framework's main advantage lies in its capacity to reduce the number of qubits required for circuit implementation, an essential improvement considering the severe hardware limitations of current NISQ devices. By transforming part of the input encoding into logic gates, the methodology frees qubits for reuse, improves resource allocation, and aligns with the reversibility requirements inherent in quantum computing.

Additionally, the framework leads to a significant reduction in the number of CNOT and other multi-qubit gates, which are known to be major contributors to quantum noise. Since quantum coherence times are limited and error rates remain a critical bottleneck, reducing the use of noisy operations without compromising functionality is crucial. The results obtained demonstrate a noteworthy decrease in both quantum cost and delay under certain configurations, particularly in circuits where the binary inputs contain many zeros.

As a case study, the framework was applied to the comparator proposed by Li et al. The optimized version of this circuit achieves comparable or better performance across all key metrics. It reduces the number of required qubits from $2N + 1$ to exactly $2N$, while also lowering the number of CNOT gates and maintaining garbage-free output through a systematic uncomputation process based on Bennett's method. To the best of our knowledge, this is the only comparator in the literature that meets these combined criteria while maintaining a reversible circuit, making it a valuable contribution to the field of quantum circuit design.

Beyond comparators, the proposed framework has promising applicability to a wide range of Boolean logic-based quantum circuits, such as adders, subtractors, and multipliers. The extended evaluation confirmed the framework consistently achieved a reduction in qubit requirements while preserving or even improving other performance metrics defined in the original reference works. The obtained results demonstrate that the proposed optimization principles are general and can be effectively extended to complex, multi-stage arithmetic designs, confirming the practical impact and broad applicability of the approach within the context of NISQ hardware limitations.

Finally, the inclusion of an open-source repository ensures reproducibility and facilitates the adoption of this methodology by other researchers and developers in the field. The software not only applies the optimization strategy but also evaluates circuits using standardized metrics, offering a comprehensive toolset for quantum circuit analysis and enhancement.

Future research directions include the extension of this methodology to circuits beyond the arithmetic domain, deeper integration with quantum programming environments such as Qiskit or Cirq, and the exploration of machine learning techniques to automate and generalize the simplification process.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information.* 10th Anniversary ed. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219.

[4] J. Lee et al., "Even more efficient quantum computations of chemistry through tensor hypercontraction," *PRX Quantum*, vol. 2, no. 3, 2021, Art. no. 030305.

[5] L. Wossnig, Z. Zhao, and A. Prakash, "Quantum linear system algorithm for dense matrices," *Phys. Rev. Lett.*, vol. 120, no. 5, 2018, Art. no. 050502.

[6] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature Phys.*, vol. 17, no. 9, pp. 1013–1017, 2021.

[7] K. Bharti et al., "Noisy intermediate-scale quantum algorithms," *Rev. Modern Phys.*, vol. 94, no. 1, 2022, Art. no. 015004.

[8] S. Resch and U. R. Karpuzcu, "Benchmarking quantum computers and the impact of quantum noise," *ACM Comput. Surveys (CSUR)*, vol. 54, no. 7, pp. 1–35, 2021.

[9] R. Harper, S. T. Flammia, and J. J. Wallman, "Efficient learning of quantum noise," *Nature Phys.*, vol. 16, no. 12, pp. 1184–1188, 2020.

[10] C. Xue, Z.-Y. Chen, Y.-C. Wu, and G.-P. Guo, "Effects of quantum noise on quantum approximate optimization algorithm," *Chin. Phys. Lett.*, vol. 38, no. 3, 2021, Art. no. 030302.

[11] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, Feb. 2020, Art. no. 226.

[12] M.-A. Asadi, M. Mosleh, and M. Haghparast, "An efficient design of reversible ternary full-adder/full-subtractor with low quantum cost," *Quantum Inf. Process.*, vol. 19, no. 7, 2020, Art. no. 204.

[13] H. Thapliyal, E. Munoz-Coreas, T. Varun, and T. S. Humble, "Quantum circuit designs of integer division optimizing T-count and T-depth," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 2, pp. 1045–1056, Apr.-Jun. 2019.

[14] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Aug. 2018, Art. no. 79.

[15] O. A. Castro-Alvaredo, S. Negro, and I. M. Szécsényi, "On the representation of minimal form factors in integrable quantum field theory," *Nucl. Phys. B*, vol. 1000, Feb. 2024, Art. no. 116459.

[16] F. Orts and R. Gil-Merino, "QuantumMeter: Open-source framework for quantum circuit optimization." GitHub. Accessed: October 2025. [Online]. Available: https://github.com/2forts/QuantumMeter

[17] E. A. Combarro and S. Gonzalez-Castillo, *A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-On Primer to Quantum Computing: From Qubits to Quantum Machine Learning and Beyond*, vol. 1. Birmingham, U.K.: Packt Publishing, 2023.

[18] A. Sequeira, L. P. Santos, and L. S. Barbosa, "Policy gradients using variational quantum circuits," *Quantum Mach. Intell.*, vol. 5, no. 1, 2023, Art. no. 18.

[19] C. Bernhardt, *Quantum Computing for Everyone.* Cambridge, MA: MIT Press, 2019.

[20] D. A. Patterson, J. L. Hennessy, and D. Goldberg, *Computer Architecture: A Quantitative Approach*, vol. 2. San Mateo, CA, USA: Morgan Kaufmann, 1990.

[21] M. Hasan, U. K. Saha, A. Sorwar, M. A. Z. Dipto, M. S. Hossain, and H. U. Zaman, "A novel hybrid full adder based on gate diffusion input technique, transmission gate and static CMOS logic," in *Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 1–6.

[22] T. Hubregtsen, J. Pichlmeier, P. Stecher, and K. Bertels, "Evaluation of parameterized quantum circuits: On the relation between classification

accuracy, expressibility, and entangling capability," *Quantum Mach. Intell.*, vol. 3, pp. 1–19, Aug. 2021.
[23] A. Vadali, R. Kshirsagar, P. Shyamsundar, and G. N. Perdue, "Quantum circuit fidelity estimation using machine learning," *Quantum Mach. Intell.*, vol. 6, no. 1, 2024, Art. no. 1.
[24] M. Mohammadi and M. Eshghi, "On figures of merit in reversible and quantum logic designs," *Quantum Inf. Process.*, vol. 8, pp. 297–318, Aug. 2009.
[25] R. Majumdar and S. Sur-Kolay, "Approximate ternary quantum error correcting code with low circuit cost," in *Proc. IEEE 50th Int. Symp. Multiple-Valued Logic (ISMVL)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 34–39.
[26] M. Noorallahzadeh, M. Mosleh, and S.-S. Ahmadpour, "Efficient designs of reversible synchronous counters in nanoscale," *Circuits, Syst., Signal Process.*, vol. 40, no. 11, pp. 5367–5380, 2021.
[27] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Dev.*, vol. 17, no. 6, pp. 525–532, 1973.
[28] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 818–830, Jun. 2013.
[29] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open quantum assembly language," 2017, *arXiv:1707.03429.*
[30] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum algorithms revisited," *Proc. Roy. Soc. London. Ser. A, Math., Phys. Eng. Sci.*, vol. 454, no. 1969, pp. 339–354, 1998.
[31] S. L. Harris and D. Harris, *Digital Design and Computer Architecture.* San Mateo, CA, CA: Morgan Kaufmann, 2015.
[32] H. Thapliyal and N. Ranganathan, "Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs," *ACM J. Emerg. Technol. Comput. Syst. (JETC)*, vol. 6, no. 4, pp. 1–31, 2010.
[33] H.-Y. Xia, H. Li, H. Zhang, Y. Liang, and J. Xin, "An efficient design of reversible multi-bit quantum comparator via only a single ancillary bit," *Int. J. Theor. Phys.*, vol. 57, no. 12, pp. 3727–3744, 2018.
[34] H.-Y. Xia, H. Li, H. Zhang, Y. Liang, and J. Xin, "Novel multi-bit quantum comparators and their application in image binarization," *Quantum Inf. Process.*, vol. 18, no. 7, 2019, Art. no. 229.
[35] H.-Y. Xia, H. Zhang, S.-X. Song, H. Li, Y.-J. Zhou, and X. Chen, "Design and simulation of quantum image binarization using quantum comparator," *Mod. Phys. Lett. A*, vol. 35, no. 9, 2020, Art. no. 2050049.
[36] H. Li, P. Fan, H.-Y. Xia, H. Peng, and G.-L. Long, "Efficient quantum arithmetic operation circuits for quantum image processing," *Sci. China Phys., Mechanics Astron.*, vol. 63, pp. 1–13, Jun. 2020.
[37] F. Orts, G. Ortega, A. Cucura, E. Filatovas, and E. M. Garzón, "Optimal fault-tolerant quantum comparators for image binarization," *J. Supercomput.*, vol. 77, pp. 8433–8444, Aug. 2021.
[38] A. Barui, M. Pal, and P. K. Panigrahi, "A novel approach to threshold quantum images by using unsharp measurements," *Quantum Inf. Process.*, vol. 23, no. 3, 2024, Art. no. 76.
[39] T. Zhang et al., "Optimization and design for multi-valued quantum comparator circuits," *J. Phys. A, Math. Theor.*, vol. 58, no. 4, 2025, Art. no. 045303.
[40] L. M. Donaire, G. Ortega, E. M. Garzón, and F. Orts, "Lowering the cost of quantum comparator circuits," *J. Supercomput.*, vol. 80, no. 10, pp. 13900–13917, 2024.
[41] Y. Yuan et al., "An improved QFT-based quantum comparator and extended modular arithmetic using one ancilla qubit," *New J. Phys.*, vol. 25, no. 10, 2023, Art. no. 103011. doi: 10.1088/1367-2630/acfd52.
[42] G. Carrascal, A. A. Del Barrio, and G. Botella, "First experiences of teaching quantum computing," *J. Supercomput.*, vol. 77, no. 3, pp. 2770–2799, 2021.
[43] H. Thapliyal, E. Muñoz-Coreas, and V. Khalus, "Quantum circuit designs of carry lookahead adder optimized for T-count, T-depth, and qubits," *Sustain. Comput.: Inform. Syst.*, vol. 29, Mar. 2021, Art. no. 100457.
[44] F. Orts, E. Filatovas, G. Ortega, J. SanJuan-Estrada, and E. Garzón, "Improving the number of T gates and their spread in integer multipliers on quantum computing," *Phys. Rev. A*, vol. 107, no. 4, 2023, Art. no. 042621.
[45] S. Yuan, S. Gao, C. Wen, Y. Wang, H. Qu, and Y. Wang, "A novel fault-tolerant quantum divider and its simulation," *Quantum Inf. Process.*, vol. 21, no. 5, 2022, Art. no. 182.

Francisco Orts is a Senior Researcher with the University of Almería, Spain. He is a member of the Supercomputing-Algorithm Group, and also collaborates actively with the Blockchain and Quantum Technology Group with the University of Vilnius, Lithuania. He has published more than 25 scientific papers in quality journals and has been a speaker at more than 40 national and international conferences. He has worked as a Computer Engineer in construction, stock market and IT services companies, with more than 15 years of experience in the sector.

Rodrigo Gil-Merino is a Professor with the International University of La Rioja (UNIR) in the field of quantum computing. He collaborates as a Scientific Software Developer in HPC in the field of gravitation and cosmology for the University of Balearic Islands and with the University of Cantabria and the Astrophysics Institute of the Canary Islands in computer astrophysics. He also has a large experience in artificial intelligence and its applications in several fields, both in academia and private companies.

Rodrigo Gil-Merino, photograph not available at the time of publication.