Improving the number of T gates and their spread in integer multipliers on quantum computing

F. Orts* and E. Filatovas
Institute of Data Science and Digital Technologies, Vilnius University, 08663 Vilnius, Lithuania

G. Ortega, J.F. SanJuan-Estrada, and E.M. Garzón
Informatics Department, Agrifood Campus of International Excellence (ceiA3), University of Almería, 04120 Almería, Spain.
(Dated: January 29, 2024)

Quantum circuits performing arithmetic operations are critical in quantum computing because of the need for such operations in proven quantum algorithms. Although quantum computers are becoming increasingly resourceful, the number of qubits currently available is still limited. Furthermore, these qubits are heavily affected by internal and external noise. It has been proven that quantum circuits built using Clifford+T gates can be made fault-tolerant. However, the use of the T gates comes at a very high cost. If the number of T gates used in a circuit is not optimized, the cost of the circuit will be increased excessively. As a consequence, it is essential to optimize the circuits so that they are as resource-efficient as possible and also to be noise tolerant. This paper presents the design of a circuit to perform the multiplication of two integers. The circuit is built using only Clifford+T gates for compatibility with error detection and correction codes. It outperforms the circuits in the state-of-the-art in terms of T-count and T-depth.

I. INTRODUCTION

Quantum computing has demonstrated its ability to solve specific problems more efficiently than classical computers [1–3]. The most common way of physically implementing the algorithms that make it possible to exploit the properties of quantum computing is by employing circuits. In particular, arithmetic circuits are necessary to implement such algorithms. This is the case of Shor's famous algorithm, as well as other quantum algorithms applied to fields as diverse as search and quantum mechanical simulation [1, 4].

Although quantum circuits have certain similarities with classical circuits, they have several characteristics of their own [5]. One of these characteristics is that they must be reversible. This implies that given an output of the circuit, the input can always be determined. As a consequence of this reversibility, it is also true that the number of inputs and outputs is always the same. The concept is very simple: if given an output, it must be possible to recover the input, then each input must necessarily correspond to a unique output. In general, this implies that quantum circuits need more resources than classical circuits to maintain reversibility. This, coupled with the fact that current quantum computers have a limited amount of resources [6], makes it necessary for circuits to use these resources in the most optimized way possible.

The necessity of extra operations to maintain reversibility and the scarcity of resources are not the only difficulties of today's quantum computers. Current implementations of such computers are heavily affected by noise problems that affect the goodness of the results of the circuits [7]. In this sense, there is a group of quantum gates (called Clifford+T) that is used in a wide variety of research works because any circuit built exclusively using these gates is compatible with the use of error detection and correction codes [4, 8–11]. Moreover, the Clifford+T group is a universal set of gates in quantum computing since it allows any operation to be approximated [5]. However, among the gates belonging to this group, there is one that stands out for its high cost: the T gate [8, 12, 13]. Although recent results show that its cost is not as prohibitive as indicated in previous work [14], it is still more expensive than the other gates in the group. It is therefore advisable to reduce the number of T gates in order not to increase the total cost of the circuit. For this reason, it is common in the literature to find that works measure the cost of a circuit in terms of the number of T gates required for its implementation [4, 15, 16]. This number is called T-count. A second associated metric, called T-depth, is used to give an estimation of the speed of the circuit and consists of the number of T gates that are part of the critical path of the circuit.

This paper presents a circuit to perform a multiplication between binary numbers of any size (as well as superpositions of binary values). For this purpose, an analysis (and subsequent comparison) of the available multipliers has been previously carried out, including the implementation of such circuits to check their correct operation. The proposed circuit is built exclusively using Clifford+T gates, and its main goal is to minimize the T-count and the T-depth for the reasons stated above. The product of integers is a crucial operation needed for quantum algorithms used in, for example, image processing [17, 18], financial modelling [19], or renewable energy [20]. Better implementations of multiplier circuits will reduce the cost of such algorithms, extend their applicability to larger problems, and make them more resistant to noise problems [21, 22].

* francisco.gomez@mif.vu.lt

The main contributions of this work are:

- Two multiplier designs are offered: one with garbage outputs but with lower cost, and one without garbage outputs but with a corresponding increase in cost.
- The circuits are the best Wallace tree integer multipliers in terms of T-count, T-depth, and number of ancilla qubits.
- The circuits are the best multipliers for quantum computing currently available in terms of T-depth. It has a value four times lower than the second best circuit in the literature in terms of T-depth.
- The circuits are also the best in terms of T-count for working with numbers of less than 20 digits.

The rest of the paper is organized as follows. Section II presents the quantum gates and operations needed to build the circuit, and also justifies the importance of counteracting noise when performing such operations. Section III describes the methodology used to build the proposed circuit. Section IV introduces the proposed design and the necessary steps to reproduce the circuit. In Section V the proposed circuit is analysed and compared with the most important multipliers available in the state-of-the-art. Finally, conclusions are presented in Section VI.

II. BACKGROUND

A. Quantum gates

There are several paradigms of programming a quantum computer, and the most common is the use of circuits. A quantum circuit is made up of quantum gates. Since a qubit can have infinite values, there are infinitely many operations that can be performed on it and, therefore, there are infinitely many possible quantum gates [23]. However, all the necessary operations to implement the circuit proposed in this paper can be expressed by using only six basic gates: the H gate, the T gate, the conjugate transpose of the T gate, the S gate, the Z gate, and the $CNOT$ gate:

- H gate: This gate allows a qubit to be placed in superposition. It performs operation $\frac{1}{\sqrt{2}}\left[\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right]$ on a single qubit.
- T and T' gates: The T gate performs the operation $\left[\begin{smallmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{smallmatrix}\right]$ on a single-qubit. The conjugate transpose of the T gate, named as T' in this manuscript, performs the operation $\left[\begin{smallmatrix}1 & 0\\ 0 & e^{-i\frac{\pi}{4}}\end{smallmatrix}\right]$.
- S gate: It is also a single-qubit operation and it performs the operation $\left[\begin{smallmatrix}1 & 0\\ 0 & i\end{smallmatrix}\right]$. It produces the same effect as two consecutive T gates.
- Z gate: It is the equivalent to the Pauli-Z matrix, that is, $\left[\begin{smallmatrix}1 & 0\\ 0 & -1\end{smallmatrix}\right]$.
- $CNOT$ gate: This gate acts on two qubits, carrying out the operation $\left[\begin{smallmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{smallmatrix}\right]$.

Fig. 1 shows the symbols of the basic gates used in this work. Two extra gates are also considered: the Toffoli gate and the temporary logical-AND gate. These gates are built by combining the previous gates, and their description is detailed in Subsection II C.

B. Fault-tolerance

In the introduction, it was mentioned that current quantum computers are highly sensitive to internal and external noise, to the extent that this causes frequent errors in the computation performed by such computers [5]. One of the major goals of quantum computing today is to eliminate (or at least to reduce) the effects of noise. The current Noisy Intermediate-Scale Quantum (NISQ) computers try to be reliable enough to run some quantum algorithms by reducing the coherence times and gate error probabilities. Such achievements, together with methodologies adapted to the NISQ computers, allow a considerable reduction of errors [24].

There are several strategies to counteract this noise. Some techniques are related to the circuit definition itself. For instance, the faster (here, faster means shallower depth) a circuit is, the less time it is exposed to noise, and therefore fewer negative effects such noise can cause. The same can be said for the number of operations: the fewer operations a circuit has, the lower the probability of error. Other techniques are linked to the quantum machine. For example, on IBM machines, the error can be significantly reduced by choosing a measurement strategy adapted to these machines [25]. On the other hand, current quantum devices must be constantly calibrated to reduce the error in communication between qubits. This error thus depends on the current calibration and is different between each pair of adjacent qubits. Properly choosing which physical qubits to use based on the current calibration will have a positive impact on the performance of the circuit. Also, adapting the circuit to the topology of the computer will reduce the number of SWAP operations (these operations exchange the value of two qubits) required for its execution, making the circuit faster and contain fewer operations [26, 27].

A technique widely used in the literature about quantum circuits is to build such circuits using only Clifford+T gates [4, 28, 29]. A circuit composed of only this type of gates can benefit from the use of error-correcting codes [30–32]. It is not possible to implement any arbitrary function only with the Clifford gates (without the T gate), something that can be achieved (or at least improved) by including the T gate [23]. However, the problem with the T gate is its high cost. It is essential

to keep the use of the T gate to a minimum in order not to unnecessarily increase the total cost of the circuit. As a consequence, circuits are usually measured in terms of T-count and T-depth.

C. *AND* operation

In this work, the *AND* operation is of great importance. In Boolean logic, the *AND* operation accepts two or more inputs, and returns an output that will be 1 only if all inputs have the value of 1, and 0 otherwise. In quantum terms, the *AND* operation is infeasible since it is not reversible. On the other hand, neither fan-in nor fan-out is possible in quantum computing. Therefore, the number of inputs and outputs must be always the same [5]. This is not what happens in the *AND* operation since it has an input-output ratio of $N:1$, where N is the number of inputs (and 1 is always the number of outputs). Moreover, applying an *AND* operation on two quantum states does not seem trivial.

There are quantum gates that allow an *AND* operation to be performed reversibly. However, it is important to point out the differences between a classical *AND* gate and such "quantum *AND*" gates. First of all, a quantum computer does not use bits but qubits. Therefore, an *AND* operation only makes sense if we interpret two quantum states chosen as classical values 0 and 1. Secondly, its applicability (of the supposed quantum *AND* gate) must be limited to such interpretations. This is not a problem, since in this work, an arithmetic circuit is proposed, destined exclusively to work with the base states that are precisely interpreted as 0 and 1. Bearing these premises in mind, the Toffoli gate allows the operation $ABC \oplus AB$, being A, B, and C any three quantum states. By restricting the values of A and B to the standard bases, and setting $C = 0$, a quantum reproduction of the classical *AND* gate can be obtained, as shown in Table I. Limiting the usefulness of the Toffoli gate to simulating the classical *AND* operation would be a mistake, but for this work, it is not necessary to further detail its applicability.

TABLE I. Truth table of the Toffoli gate used over a target qubit (C) in the state 0. The result is similar to a classical *AND* gate.

Inputs			Outputs		
A	B	C	A	B	C
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	1

There are several designs for the Toffoli gate in the literature [8, 9, 16]. Among these designs, the most optimized in terms of quantum cost and delay is the one proposed by Amy et al. [8], shown in Fig. 2. This implementation involves 2 control qubits, and 1 target qubit, which depending on the needs of the circuit can be a qubit previously in use or a new qubit reserved for this operation. The Amy et al. design has a T-count of 7, that is, it needs 7 T gates. This implementation of the Toffoli gate is the most widely used in the quantum circuits literature [10, 33, 34].

In order to reduce the T-count of the implementation proposed by Amy et al., Jones [9] proposed a new design for the Toffoli gate. Later, Gidney [16] proposed to compute the *AND* operation using two new gates based on the work by Jones. These gates allow the T-count to be reduced to 4. These two new gates are the temporary logical-AND and its uncomputation gate (Fig. 3). The first gate is the one that performs the *AND* operation, while the second one allows reversing the operations performed, thus avoiding garbage outputs. The second gate does not perform the uncomputation of the original circuit by reversing it, but uses a measure-and-fixup approach (that was also proposed by Jones [9]). To reverse the Toffoli gate proposed by Amy et al. [8], it is necessary to apply another Toffoli gate. Therefore, the actual T-count is 14 in the case of using the Toffoli gate proposed by Amy et al., and only 4 in the case of using the temporary logical-AND and its uncomputation gate (since the latter has a T-count of 0).

The use of Gidney's implementation can allow a considerable reduction of the T-count. However, the temporary logical-AND gate cannot be applied on a qubit C already in use to perform the $C \oplus AB$ operation. It must be applied on an ancilla qubit prepared in state $\frac{1}{\sqrt{2}}(0 + e^{\frac{i\pi}{4}}1)$. Therefore, the use of the gates proposed by Gidney is not a mere replacement for the Toffoli gate, but their application must be considered in the circuit design itself.

D. Addition

Addition is another operation used recurrently in the proposed multiplier, as will be explained in next sections. Specifically, the multiplier involves two types of adders, called half and full adders, respectively. Half-adders compute the addition of two bits A and B. They return the sum of both bits, $S = A \oplus B$, as well as the generated carry, $C_{out} = AB$. Full adders add three bits, A, B, and the input carry C_{in}. Therefore, a full adder will return the sum of such bits, $S = A \oplus B \oplus C_{in}$, and the carry $C_{out} = AB + AC_{in} + BC_{in}$. There are other types of adders, but they are not used in this work.

There is a wide variety of adders for quantum computing in the literature. In 2020, Orts et al. reviewed on quantum adders [35], in which existing half-adders and full adders in the state-of-the-art (among other adders) are compared. Subsequent to this work, several adders

have emerged [4, 15]. However, none of these new circuits has improved on the existing half or full adder circuits. Therefore, the work of Orts et al. can be used to choose the circuits with the best T-count and number of ancilla qubits.

The best half-adder in terms of T-count and number of ancilla qubits available in the literature is the one described by Nielsen and Chuang [5]. This circuit is reproduced in Fig. 4(a). This half-adder has a T-count of 7, and needs only 1 auxiliary qubit. An improvement to this circuit, in terms of T-count, can be done by using a temporary logical-AND gate instead of the Toffoli gate (Fig. 4(b)). Then, the circuit will have a T-count of 4, keeping only one auxiliary qubit. If it were necessary to reverse the operation performed by the half-adders in Figs. 4(a) and (b), the total T-count would be 14 and 4, respectively.

If we focus on full adders, the best available circuit (in terms of T-count) is the one proposed by Wang et al. in 2016 [36] (Fig. 5(a)). Such a circuit has a T-count of 7, and needs only 1 auxiliary qubit. For the sake of clarity: a full adder was published in 2021 that also has this same T-count (Fig. 5(b)) [37]. However, this new circuit does not improve on the Wang et al. circuit in any way, so the circuit introduced in [36] will be used to build the multiplier.

III. THEORETICAL APPROACH

In this work, a quantum circuit using the Wallace tree technique [38] to compute the product of two integers is presented. Let $X = x_{N-1}...x_0$ and $Y = y_{N-1}...y_0$ be two integers with N digits whose product P needs to be calculated. The Wallace tree technique consists of calculating the partial products $\{p_{i,j} = x_i y_j : i \in (0, ..., N-1), j \in (0, ..., N-1)\}$. Once these partial products have been calculated, they are added to obtain P. Fig. 6 shows the correct scheme for the general N-digit case.

The proposed circuit works with the standard bases, so that 0 is represented by the 0 state and 1 by the 1 state. One qubit is used to represent each digit of X and Y, so assuming that both numbers have N digits, it is trivial to note that $2N$ qubits are required to represent them. The result P, which is also returned using the same representation, will need a total of M qubits, $P = P_{M-1}....P_1P_0$. M is the maximum number of digits needed to contain the result of the product of two binary integers of N digits, where $N \leq M \leq 2N$ [39].

Instead of directly adding the obtained N rows of partial products, as done in the original algorithm (Fig. 6), the proposed circuit performs this addition by decomposing it into smaller sums. It performs the sum of partial products using only half-adders and full adders. This sum decomposition has already been successfully used in state-of-the-art multipliers to reduce the required resources [40]. A graphical diagram of this idea, for the case $N = 4$, is reproduced in Fig. 7. In this figure, the boxes containing two values indicate sums to be performed using half-adders, while the boxes containing three values indicate sums to be performed using full adders.

IV. PROPOSED INTEGER MULTIPLIER CIRCUIT DESIGN

Our proposed circuit is based on the described methodology, but it is optimized to reduce the number of required qubits, the T-count, and the T-depth. To do so, it focuses on the following aspects:

- Finding optimized ways to implement the low-level operations, primarily the reversible AND operation.
- Defining an algorithm to build the proposed circuit for any digit size.
- Achieving a circuit free of garbage outputs.

A. Optimization of low-level operations

Regarding the optimization of low-level operations, all Toffoli gates implementing the CCNOT array (this label will be used to refer to the set of all partial products) can be replaced by temporary logical-AND gates to achieve a drastic reduction of the T-count, as explained in Subsection II C. The temporary logical-AND gate needs an extra qubit to contain the result. However, the products of the CCNOT array must be stored in an extra qubit anyway, so the replacement of gates results in a decrease in the T-count but not in an increase in the number of involved qubits. A CCNOT array built using Toffoli gates has a T-count of $7N^2$, whereas the proposed implementation has a T-count of only $4N^2$.

This substitution can also be carried out in the rest of the Toffoli gates that operate directly on an auxiliary qubit that does not contain a previous value (i.e., in those cases in which the operation $0 \oplus AB$ is performed for any pair of qubits containing the states A and B). In particular, the half-adder we have proposed in Subsection II D (Fig. 4(b)) can be used.

Toffoli gates operating on a qubit that contains a non-constant prior value are maintained so as not to concur in an increase in the number of auxiliary qubits. Therefore, to perform full additions, the most suitable circuit is the one proposed by Wang et al. (Fig. 5(a)).

B. A circuit of any size

Since the methodology described in Section 3 is perfectly valid for any data size, we have proposed an algorithm to build a circuit for any size. Once the partial products are performed, their addition is divided into $N-1$ levels. An initial level, $N-3$ middle levels, and a

final level. The first level involves 2 half adders and $N-2$ full adders. The $N-3$ middle levels and the final level need (each one) 1 half-adder and $N-1$ full adders. Therefore, a total of N half-adders and N^2-2N full adders are required. It can also be noted that all adders of the same level can be performed in parallel. The exception is the last level, whose operations must be performed sequentially. A new level could be created in which to add the last carries and thus allow parallelizing such operations, but it would involve the use of $N-1$ extra half-adders (and replacing the last $N-1$ full adders with other $N-1$ half-adders), with a consequent increase in T-count and ancilla qubits.

The algorithm to compute the quantum product of integers of any size N can be defined as follows:

1. To prepare N qubits to represent X.
2. To prepare N qubits to represent Y.
3. To prepare N^2 ancilla qubits for the CCNOT array. Moreover, it should be noted that each involved adder will require an additional extra qubit.
4. For $i = 0$ to $N-1$:
   (a) For $j = 0$ to $N-1$, a temporary logical-AND gate is applied to perform the operation $x_i y_j$. The result of the operation is temporarily stored in an ancilla qubit $p_{i,j}$.
5. A half-adder is applied to compute $p_{0,1}+p_{1,0}$. Label this adder as $a_{0,0}$.
6. For $j = 2$ to $N-1$, full adders are applied to compute $p_{0,i} + p_{1,i-1} + p_{2,i-2}$. Label these adders as $a_{0,j-1}$.
7. A half-adder is applied to compute $p_{1,N-1}+p_{2,N-2}$. Label this adder as $a_{0,N-1}$.
8. For $i = 1$ to $N-3$:
   (a) A half-adder is applied to compute $s_{i-1,1} + c_{i-1,0}$, where $s_{u,v}$ and $c_{u,v}$ are the sum and carry out of the $a_{u,v}$ operation. Label this adder as $a_{i,0}$. $s_{i-1,1}$ will be P_i.
   (b) For $j = 2$ to $N-1$, full-adders are applied to compute $s_{i-1,j} + c_{i-1,j-1} + p_{i+2,j-2}$. Label these adders as $a_{i,j-1}$.
   (c) A full adder is applied to compute $p_{i+1,N-1} + c_{i-1,N-1} + p_{i+2,i+1}$. Label this adder as $a_{i,N-1}$.
9. A half-adder is applied to compute $s_{N-3,1}+c_{N-3,0}$. Label this adder as $a_{N-2,0}$. $s_{N-2,0}$ will be added to the left of the result P.
10. For $j = 2$ to $N-1$, full-adders are applied to compute $s_{N-3,j} + c_{N-3,j-1} + c_{N-2,j-2}$. Label these adder as $a_{j,j-1}$. $s_{j,j-1}$ will be added to the left of the result P.
11. A full adder is applied to compute $p_{N-1,N-1} + c_{N-3,N-1} + p_{N-2,N-2}$. The carry and the sum of the last operation, in this order, will be the most significant digits of P.

An example for the case $N = 4$ is shown in Fig. 8. For clarity, this figure shows the levels described in the algorithm, as well as all the $p_{i,j}$, $s_{i,j}$, and $c_{i,j}$ values generated throughout the circuit. It can be seen that the operations are as described in Fig. 7: two half-adders and two full adders at level 1, one half-adder and three full adders at level 2, and one half-adder and three full adders at level 3.

C. Garbage outputs

The circuit shown in Fig. 8 contains garbage outputs. Using the metrics given by Mohammadi et al. [45, 46], we consider a garbage output to be any qubit that contains a value that is neither the target output of the circuit (in this case, the output is P) nor the same value it had at the input of the circuit. Garbage outputs are thus qubits that contain an unknown and unnecessary value, and cannot be used to perform new operations. Following this definition, in the circuits of Fig. 8 can be considered as qubits with garbage output all those marked with a.

To remove the garbage outputs, we resort to the Bennet's removal scheme [47] but considering the uncomputation gate of the temporary logical-AND as explained in Subsection II C. This means that the inverse circuit must be applied. There are two ways to perform the uncomputation of the garbage outputs:

- It is not necessary to save P: after using P, the inverse circuit is applied. No new auxiliary qubits are needed.
- It is necessary to save P: $M-2$ new auxiliary qubits are needed. The values of the result are copied to such auxiliary qubits, and the inverse circuit is then applied. It is not necessary to perform this copy for P_0 and P_1 values, as they can stay in their original qubits (which will not have to be reversed).

We would like to clarify that it is impossible to copy quantum states in quantum computing. However, it is possible to copy 0 and 1 values into prepared auxiliary qubits by means of a simple CNOT gate [5]. Also, for the sake of clarity, we mention that given a circuit Z, its inverse circuit consists of applying the same quantum gates as in Z but in reverse. The exception is the temporary logical-AND gate, which in the inverse circuit is replaced by its uncomputation gate (Fig. 3(b)).

Our proposed circuit requires N^2 temporary logical-AND gates for the CCNOT array, N half-adders, and N^2-2N full adders. In the circuit, only the temporary logical-AND and Toffoli gates have T-count, so in such terms the circuit consist of N^2 temporary-logical AND

gates (CCNOT array), N temporary logical-AND gates (half-adders), and $N^2 - 2N$ Toffoli gates (full adders). Then, the inverse circuit consists of $N^2 - 2N$ Toffoli gates and $N^2 + N$ uncomputation gates of the temporary logical-AND gate.

V. ANALYSIS AND COMPARISON

The proposed design has been reproduced in Python using ProjectQ simulator [48]. The circuit has been checked to ensure that it produces the correct output value in terms of P. In addition, the measurements shown in this section have been done using the methodology described in Orts et al. [35]. The proposed circuit has a T-count of $11N^2 - 10N$, a T-depth of $8N - 7$, it requires $2N^2 - N$ ancilla qubits, and it has $2N^2 - 3N$ garbage outputs. A second version of this circuit with 0 garbage outputs can be obtained as explained in Subsection IV C. It has been said in the previous section that the inverse circuit has $N^2 - 2N$ Toffoli gates and $N^2 + N$ uncomputation gates of the temporary logical-AND gate. Of these two gates, only the Toffoli gate has T-count, so the total T-count of the inverse circuit will be $7(N^2-2N) = 7N^2-14N$. Therefore, this second version has a T-count of $18N^2-24N$, a T-depth of $14N-14$, and $2N^2 - N + (M - 2)$ ancilla qubits (being M the number of digits of the result).

Table II shows a comparison between several integer multipliers available in the state-of-the-art and the proposed circuit. First, it is important to note that several circuits have garbage outputs. These outputs can be reversed, in which case the T-count values (and possibly the number of ancilla qubits) would be increased. On the other hand, some of the circuits have garbage outputs in their original work. However, Muñoz-Coreas et al. [33] conducted a study to obtain the garbage-free versions of these circuits. This conversion involved an increase in the rest of the metrics, but it made the circuits more useful and, above all, this comparison more accurate (see Subsection IV C). For the sake of clarity, the circuit that has been uncomputed by Muñoz-Coreas et al. are marked with an asterisk (*) in Table II. We have subjected to the same procedure those circuits subsequent to the work of Muñoz-Coreas et al. Such circuits are marked with two asterisks (**) in the table. We would also like to point out that the circuits included in this comparison have also been reproduced and tested in the ProjectQ simulator.

In terms of qubits, the best choices is the circuit of Li et al. [44], with only $N + 1$ ancilla qubits, followed by Muñoz-Coreas et al. circuit [33] and the Jayashree et al. circuit [43], with $2N + 1$ ancilla qubits. However, the Muñoz-Coreas et al. circuit improves the T-count and T-depth ($21N^2 - 14$ and $9N^2 - 6$, respectively) over the Jayashree et al. circuit ($28N^2 + 7N$ and $12N^2 + 3N$, respectively). The circuit of PourAliAkbar et al. [40] is the worst choice in terms of the number of ancilla qubits, with a total of $3N^2 - 3N - 2$.

The circuit of Li et al. is also the best in terms of T-count ($16N^2 - 14$). The second best multiplier in such terms is our proposed circuit ($18N^2 - 24$). In reality, our proposed circuit improves on that of Li et al. when working with numbers of less than 20 digits. For more than 20 digits, the circuit of Li et al. is clearly dominant. However, the circuit of Li et al. has a high depth, as shown by its T-depth of $4N^2+4N+4$. Our circuit is the best in these terms, with a T-depth of only $14N - 14$. It has a value four times lower than the second best circuit in such terms.

As a general result of the analysis and in view of the data provided by Table II, it can be observed that there are two marked trends: circuits that are balanced on all metrics, and circuits that sacrifice one metric (or even two) in order to optimize another. The decision of which multiplier to use will therefore depend on the metric whose optimization is to be prioritised. In the case where the number of qubits is the most important metric, the circuits of Li et al. is the most appropriate choice. If the priority is to minimise the T-count, the most optimized choice is the circuit proposed in this paper if working with numbers of less than 20 digits. If the numbers have more than 20 digits, the best option is again the circuit of Li et al. In cases where shallow depth is required, the best option is the proposed circuit. For more customised choices where a certain trade-off or configuration is sought, Table II should be checked to find the most suitable option.

VI. CONCLUSIONS

In this work, a quantum circuit to perform the product of two integers using the Wallace tree technique has been designed and analysed. For this purpose, the most suitable implementations of the involved reversible operations have been studied. Then, solid metrics for the description and measurement of a circuit have been searched and chosen. Subsequently, a study of the available circuits in the literature has been carried out, and we have optimized the implementation of the low-level operations employing different techniques to obtain a new circuit that improves on those available in the state-of-the-art. Moreover, the published Wallace-tree multipliers can only operate with 4-digit numbers, while ours is capable of working with numbers of any size N. An effort has also been made to generate a circuit free of garbage outputs.

A wider comparison has also been made to evaluate the proposed circuit concerning the most important quantum multipliers in order to have a global view of the cost of the multiplier circuits. The comparison shows that the proposed circuit outperforms the rest of the circuits in terms of T-count (only for numbers with less than 20 digits) and T-depth. Our proposal is also the best Wallace-tree multiplier (but not the best global circuit) in terms of required qubits.

As already mentioned, the Toffoli gates present in the

circuit have not been replaced by temporary logical-AND gates so as not to increase the number of qubits required. However, if the aim is to reduce the T-count and T-depth at any cost, the $2N^2 - 4N$ Toffoli gates can be replaced. Of these, N^2-2N Toffoli gates will be replaced by temporary logical-AND gates, and the other $N^2 - 2N$ by their uncomputation gates. Therefore, the T-count of the circuit obtained will have been reduced by $10N^2 - 20N$ but at a cost of extra $N^2 - 2N$ ancilla qubits. Nevertheless, the number of Toffoli gates to be replaced could be customized to find values in the range between this extreme and the values offered by the proposed circuit.

ACKNOWLEDGMENTS

This work has been supported by the projects: PID2021-123278OB-I00 (funded by MCIN/AEI/10.13039/501100011033/FEDER "A way to make Europe"); P20_00748 and UAL2020-TIC-A2101 (funded by Junta de Andalucía and the European Regional Development Fund, ERDF).

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM review **41**, 303 (1999).
[2] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (ACM-DL, 1996) pp. 212–219.
[3] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Physical review letters **103**, 150502 (2009).
[4] H. Thapliyal, E. Muñoz-Coreas, and V. Khalus, Quantum circuit designs of carry lookahead adder optimized for T-count, T-depth and qubits, Sustainable Computing: Informatics and Systems **29**, 100457 (2021).
[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2011).
[6] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, *et al.*, Noisy intermediate-scale quantum algorithms, Reviews of Modern Physics **94**, 015004 (2022).
[7] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).
[8] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **32**, 818 (2013).
[9] C. Jones, Low-overhead constructions for the fault-tolerant toffoli gate, Physical Review A **87**, 022328 (2013).
[10] H. Thapliyal, E. Munoz-Coreas, T. Varun, and T. S. Humble, Quantum circuit designs of integer division optimizing T-count and T-depth, IEEE Transactions on Emerging Topics in Computing **9**, 1045 (2019).
[11] H. Thapliyal and M. Srinivas, Novel reversibletsg'gate and its application for designing components of primitive reversible/quantum alu, in *2005 5th International Conference on Information Communications & Signal Processing* (IEEE, 2005) pp. 1425–1429.
[12] D. M. Miller, M. Soeken, and R. Drechsler, Mapping ncv circuits to optimized Clifford+T circuits, in *International Conference on Reversible Computation* (Springer, 2014) pp. 163–175.
[13] M. Amy, D. Maslov, and M. Mosca, Polynomial-time T-depth optimization of Clifford+ T circuits via matroid partitioning, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **33**, 1476 (2014).
[14] D. Litinski, Magic state distillation: Not as costly as you think, Quantum **3**, 205 (2019).
[15] F. Orts, G. Ortega, E. Filatovas, and E. M. Garzón, Implementation of three efficient 4-digit fault-tolerant quantum carry lookahead adders, The Journal of Supercomputing , 1 (2022).
[16] C. Gidney, Halving the cost of quantum addition, Quantum **2**, 74 (2018).
[17] H. Li, P. Fan, H. Xia, H. Peng, and G. Long, Efficient quantum arithmetic operation circuits for quantum image processing, Science China Physics, Mechanics & Astronomy **63**, 1 (2020).
[18] F. Orts, G. Ortega, A. Cucura, E. Filatovas, and E. M. Garzón, Optimal fault-tolerant quantum comparators for image binarization, The Journal of Supercomputing **77**, 8433 (2021).
[19] K. Kaneko, K. Miyamoto, N. Takeda, and K. Yoshino, Quantum pricing with a smile: Implementation of local volatility model on quantum computer, EPJ Quantum Technology **9**, 1 (2022).
[20] A. Giani and Z. Eldredge, Quantum computing opportunities in renewable energy, SN Computer Science **2**, 1 (2021).
[21] S. Kotiyal, H. Thapliyal, and N. Ranganathan, Circuit for reversible quantum multiplier based on binary tree optimizing ancilla and garbage bits, in *2014 27th international conference on VLSI design and 2014 13th international conference on embedded systems* (IEEE, 2014) pp. 545–550.
[22] A. Nagamani and V. K. Agrawal, Design of quantum cost and delay-optimized reversible wallace tree multiplier using compressors, in *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems* (Springer, 2015) pp. 323–331.
[23] C. Bernhardt, *Quantum computing for everyone* (Mit Press, 2019).
[24] T. Patel, A. Potharaju, B. Li, R. B. Roy, and D. Tiwari, Experimental evaluation of nisq quantum computers: error measurement, characterization, and implications, in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE, 2020) pp. 1–15.
[25] S. S. Tannu and M. K. Qureshi, Mitigating measure-

ment errors in quantum computers by exploiting state-dependent bias, in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (ACM, 2019) pp. 279–290.

[26] S. S. Tannu and M. K. Qureshi, Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (ACM, 2019) pp. 987–999.

[27] D. Bhattacharjee, A. A. Saki, M. Alam, A. Chattopadhyay, and S. Ghosh, MUQUT: Multi-constraint quantum circuit mapping on NISQ computers, in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (IEEE, 2019) pp. 1–7.

[28] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Error mitigation with Clifford quantum-circuit data, Quantum **5**, 592 (2021).

[29] S. M. A. Mirizadeh and P. Asghari, Fault-tolerant quantum reversible full adder/subtractor: Design and implementation, Optik **253**, 168543 (2022).

[30] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, Fault-tolerant, high-level quantum circuits: form, compilation and description, Quantum Science and Technology **2**, 025003 (2017).

[31] X. Zhou, D. W. Leung, and I. L. Chuang, Methodology for quantum logic gate construction, Physical Review A **62**, 052316 (2000).

[32] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, A new universal and fault-tolerant quantum basis, Information Processing Letters **75**, 101 (2000).

[33] E. Muñoz-Coreas and H. Thapliyal, Quantum circuit design of a T-count optimized integer multiplier, IEEE Transactions on Computers **68**, 729 (2018).

[34] E. Muñoz-Coreas and H. Thapliyal, T-count and qubit optimized quantum circuit design of the non-restoring square root algorithm, ACM Journal on Emerging Technologies in Computing Systems (JETC) **14**, 1 (2018).

[35] F. Orts, G. Ortega, E. F. Combarro, and E. M. Garzón, A review on reversible quantum adders, Journal of Network and Computer Applications **170**, 102810 (2020).

[36] F. Wang, M. Luo, H. Li, Z. Qu, and X. Wang, Improved quantum ripple-carry addition circuit, Science China Information Sciences **59**, 1 (2016).

[37] S. Gayathri, R. Kumar, S. Dhanalakshmi, B. K. Kaushik, and M. Haghparast, T-count optimized Wallace tree integer multiplier for quantum computing, International Journal of Theoretical Physics **60**, 2823 (2021).

[38] C. S. Wallace, A suggestion for a fast multiplier, IEEE Transactions on electronic Computers , 14 (1964).

[39] K. Bhardwaj, P. S. Mane, and J. Henkel, Power-and area-efficient approximate wallace tree multiplier for error-resilient systems, in *Fifteenth International Symposium on Quality Electronic Design* (IEEE, 2014) pp. 263–269.

[40] E. PourAliAkbar and M. Mosleh, An efficient design for reversible Wallace unsigned multiplier, Theoretical Computer Science **773**, 43 (2019).

[41] C.-C. Lin, A. Chakrabarti, and N. K. Jha, Qlib: Quantum module library, ACM Journal on Emerging Technologies in Computing Systems (JETC) **11**, 1 (2014).

[42] H. Babu, Cost-efficient design of a quantum multiplier–accumulator unit, Quantum Information Processing **16**, 1 (2017).

[43] H. Jayashree, H. Thapliyal, H. R. Arabnia, and V. K. Agrawal, Ancilla-input and garbage-output optimized design of a reversible quantum integer multiplier, The Journal of Supercomputing **72**, 1477 (2016).

[44] H.-S. Li, P. Fan, H. Xia, and G.-L. Long, The circuit design and optimization of quantum multiplier and divider, Science China Physics, Mechanics & Astronomy **65**, 260311 (2022).

[45] M. Mohammadi and M. Eshghi, On figures of merit in reversible and quantum logic designs, Quantum Information Processing **8**, 297 (2009).

[46] H. Thapliyal and N. Ranganathan, Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs, ACM Journal on Emerging Technologies in Computing Systems (JETC) **6**, 1 (2010).

[47] C. H. Bennett, Logical reversibility of computation, IBM journal of Research and Development **17**, 525 (1973).

[48] D. S. Steiger, T. Häner, and M. Troyer, ProjectQ: an open source software framework for quantum computing, Quantum **2**, 49 (2018).

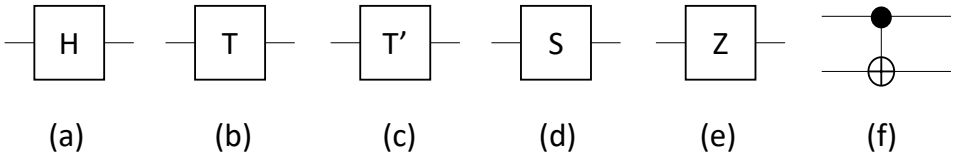

FIG. 1. Symbols of the basic gates used in this work: (a) H gate, (b) T gate, (c) conjugate transpose of the T gate, (d) S gate, (e) Z gate, and (d) $CNOT$ gate.

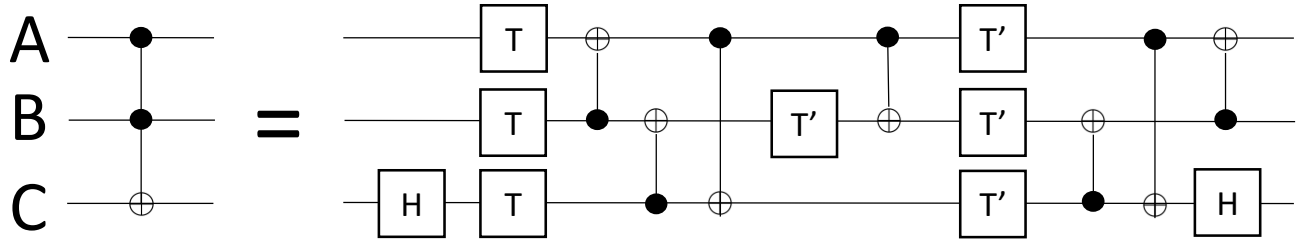

FIG. 2. Symbol and implementation of the Toffoli gate proposed by Amy et al. [8].

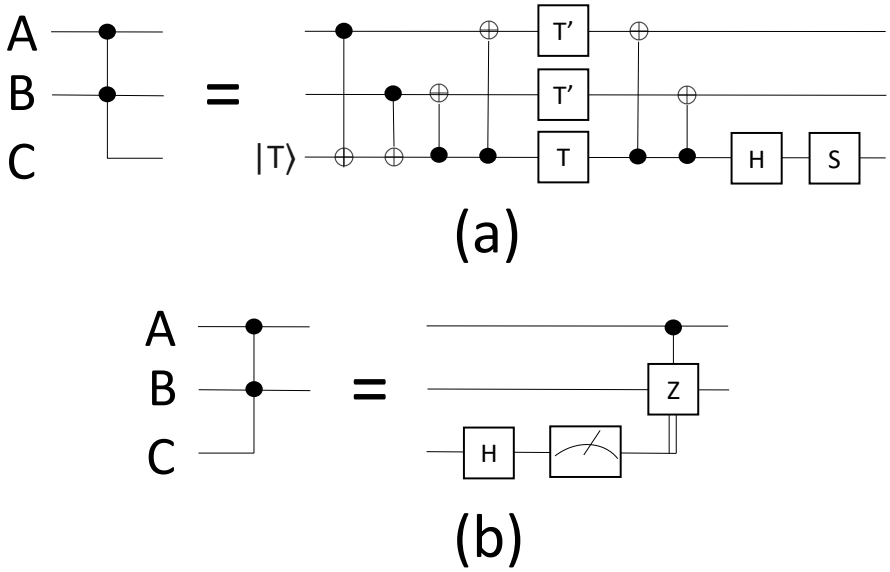

FIG. 3. (a) Symbol and implementation of the temporary logical-AND gate. The state of the target qubit (C) must be prepared using a Hadamard gate and a T gate to set it into the state $\frac{1}{\sqrt{2}}(0+e^{\frac{i\pi}{4}}1)$. Therefore, although only three T-gates are shown in the picture, the total T-count is 4. (b) Symbol and implementation of the uncomputation (gate) of the temporary logical-AND gate.

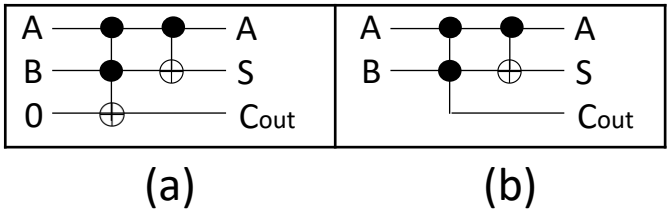

FIG. 4. (a) Half-adder proposed by Nielsen and Chuang [5]. This adder has a T-count of 7, and requires 1 ancilla qubit. (b) Improved implementation of the half-adder of Nielsen and Chuang using a temporary logical-AND operation. The T-count of this circuit is only 4, with the same number of ancilla qubits.

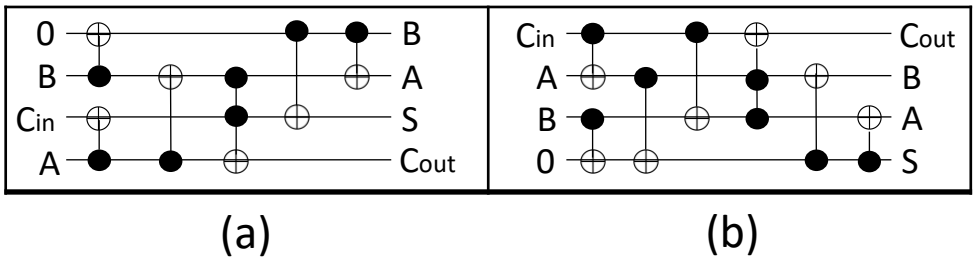

FIG. 5. (a) Full adder proposed by Wang et al. in 2016 [36]. (b) Full adder proposed by Gayathri et al. in 2021 [37]. Both adders have the same T-count (7) and only 1 ancilla qubit, but the first circuit needs one less CNOT gate than the second one.

				x_{n-1}	...	x_3	x_2	x_1	x_0
			x	y_{n-1}	...	y_3	y_2	y_1	y_0
				$x_{n-1}y_0$	...	x_3y_0	x_2y_0	x_1y_0	x_0y_0
			$x_{n-1}y_1$	$x_{n-2}y_1$	...	x_2y_1	x_1y_1	x_0y_1	0
		$x_{n-1}y_2$	$x_{n-2}y_2$	$x_{n-3}y_2$	...	x_1y_2	x_0y_2	0	0
		⋮			⋮			⋮	
$x_{n-1}y_{n-1}$	...	...	...	...	...	0	0	0	0
P_{m-1}	...	...	...	P_{n-1}	...	P_3	P_2	P_1	P_0

FIG. 6. Diagram of a $N \times N$ Wallace tree multiplier. Two distinct parts can be observed: the generation of the partial products, and the addition of such products. The size of the result P must satisfy $N \leq M \leq 2N$.

				x_3	x_2	x_1	x_0
			x	y_3	y_2	y_1	y_0
				2 x_3y_0	1 x_2y_0	0 x_1y_0	x_0y_0
			3 x_3y_1	x_2y_1	x_1y_1	x_0y_1	0
		x_3y_2	x_2y_2	x_1y_2	x_0y_2	0	0
	x_3y_3	x_2y_3	x_1y_3	x_0y_3	0	0	0
		7 x_3y_2	6 S_3	5 S_2	4 S_1	S_0	x_0y_0
		$Cout_3$	$Cout_2$	$Cout_1$	$Cout_0$		
	x_3y_3	x_2y_3	x_1y_3	x_0y_3			
	11 x_3y_3	10 x_2y_3	9 S_6	8 S_5	S_4	S_0	x_0y_0
	$Cout_7$	$Cout_6$	$Cout_5$	$Cout_4$			
	$Cout_{10}$	$Cout_9$	$Cout_8$				
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

FIG. 7. Dot diagram of a 4×4 Wallace tree multiplier. Instead of directly adding the four rows obtained after computing the partial products as it is explained in Fig. 6, the operation is decomposed into small sums of two and three bits. In the interest of clarity, such small sums are shown in boxes and with an index (in red color), so that S_i and $Cout_i$ are the result of the i-addition.

TABLE II. Comparison of $N \times N$ integer multipliers in terms of T-count, T-depth, and ancilla qubits. All the circuits are free of garbage outputs. Circuits marked with * have garbage outputs in their original designs, but Muñoz-Coreas et al.[33] have reproduced such circuits by removing them. Following the same idea, we have reproduced the circuits marked with ** by removing the garbage outputs. The metrics shown here for such circuits include the extra costs associated with such a process.

Circuit	**T-count**	**T-depth**	**Ancilla qubits**
Lin et al. [41]	$56N^2$	$24N^2$	$3N+1$
Babu* [42]	$42N^2-42N+48$	$18N^2-18N+18$	$\approx 2N$
Jayashree et al.* [43]	$28N^2+7N$	$12N^2+3N$	$2N+1$
Muñoz-Coreas et al. [33]	$21N^2-14$	$9N^2-6$	$2N+1$
PourAliAkbar et al.** [40]	$38N^2-15N-12$	$54N-54$	$3N^2-3N-2$
Gayathri et al.** [37]	$\approx 28N^2-14N$	$\approx 20N-14$	$\approx 2N^2-N+M$
Li et al.** [44]	$16N^2-14N$	$4N^2+4N+4$	$N+1$
Proposed circuit	$18N^2-24N$	$14N-14$	$2N^2-N+(M-2)$

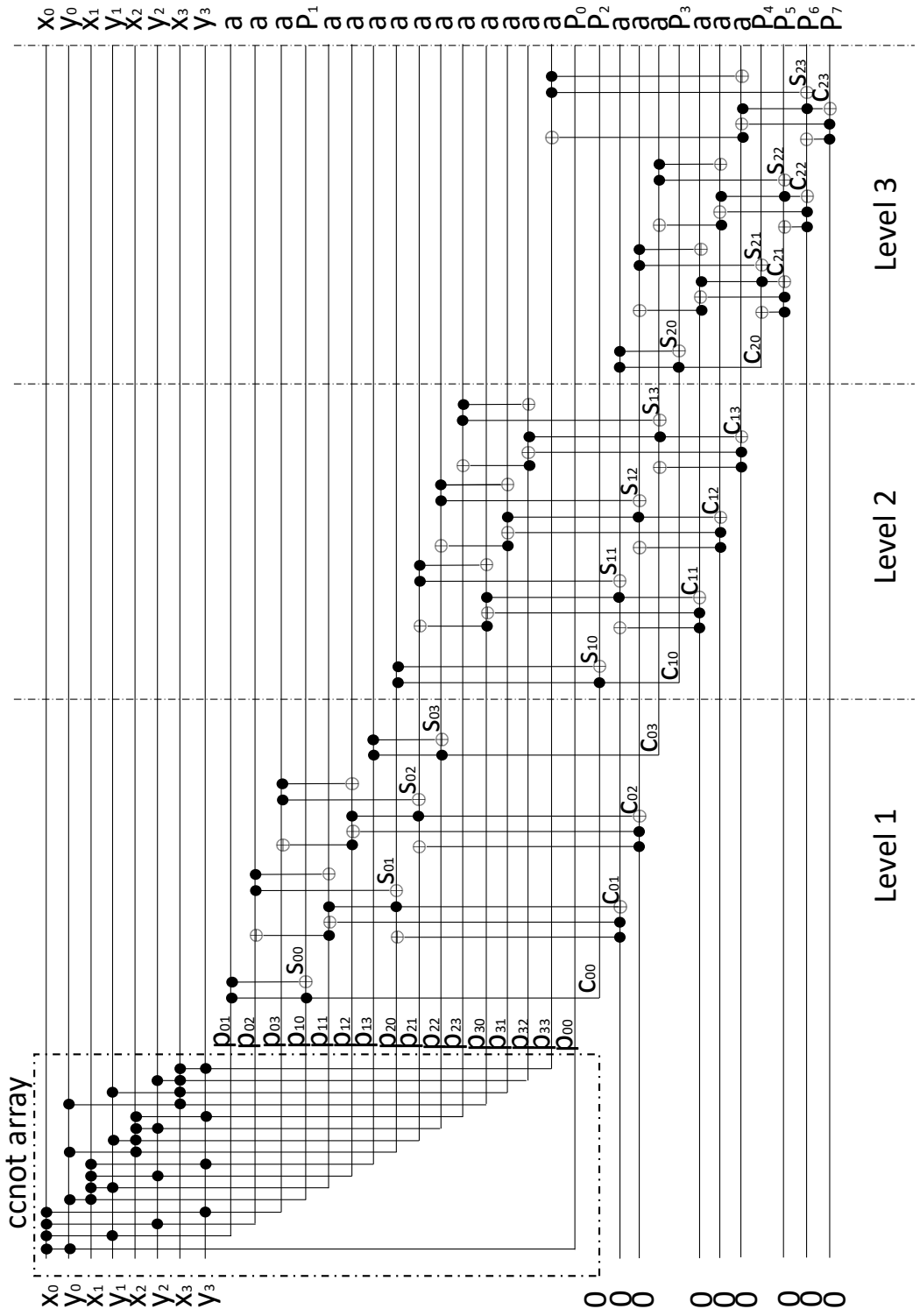

FIG. 8. Proposed implementation, for the $N = 4$ case. It involves 28 ancilla qubits. x_i and y_i (for $i = 0$ to 3) are the digits of the two numbers to be multiplied. The rest of the qubits are ancilla inputs set to 0 (or $\frac{1}{\sqrt{2}}(0 + e^{\frac{i\pi}{4}}1)$ for the temporary logical-AND). The output of the multiplication should be contained in the qubits P_j (for $j = 0$ to 7). Outputs labelled as a are garbage outputs. As it is described in Fig. 7, level 1 includes additions $0, 1, 2$ and 3, level 2 additions $4, 5, 6$ and 7, and level 3 additions $8, 9, 10$ and 11. The resulting sums and carries for such additions are labelled as S_{uv} and C_{uv}, respectively, being u and v explained in the proposed algorithm. Operations in levels 1 and 2 can be performed in parallel (when they are in different additions). Operations in level 3 must be computed sequentially.