IEEE Transactions on Quantum Engineering

Received 4 May 2023; revised 25 July 2023; accepted 3 August 2023; date of publication 10 August 2023; date of current version 4 September 2023.

Digital Object Identifier 10.1109/TQE.2023.3303935

A Modular Quantum Compilation Framework for Distributed Quantum Computing

DAVIDE FERRARI[1] (Member, IEEE), STEFANO CARRETTA[2,3], AND MICHELE AMORETTI[1] (Senior Member, IEEE)

[1]Quantum Software Laboratory, Department of Engineering and Architecture, University of Parma, 43124 Parma, Italy
[2]Department of Mathematical, Physical and Computer Sciences, University of Parma, 43124 Parma, Italy
[3]INFN—Sezione di Milano-Bicocca, Gruppo Collegato di Parma, 43124 Parma, Italy

Corresponding author: Davide Ferrari (e-mail: davide.ferrari1@unipr.it).

This work was supported by the EU Flagship on Quantum Technologies through the project Quantum Internet Alliance (EU Horizon Europe) under Grant 101102140.

ABSTRACT For most practical applications, quantum algorithms require large resources in terms of qubit number, much larger than those available with current noisy intermediate-scale quantum processors. With the network and communication functionalities provided by the quantum Internet, distributed quantum computing (DQC) is considered as a scalable approach for increasing the number of available qubits for computational tasks. For DQC to be effective and efficient, a quantum compiler must find the best partitioning for the quantum algorithm and then perform smart remote operation scheduling to optimize Einstein–Podolsky–Rosen (EPR) pair consumption. At the same time, the quantum compiler should also find the best local transformation for each partition. In this article, we present a modular quantum compilation framework for DQC that takes into account both network and device constraints and characteristics. We implemented and tested a quantum compiler based on the proposed framework with some circuits of interest, such as the VQE and QFT ones, considering different network topologies, with quantum processors characterized by heavy-hexagon coupling maps. We also devised a strategy for remote scheduling that can exploit both TeleGate and TeleData operations and tested the impact of using either only TeleGates or both. The evaluation results show that TeleData operations can have a positive impact on the number of consumed EPR pairs, depending on the characteristic of compiled circuit. Meanwhile, choosing a more connected network topology helps reduce the number of layers dedicated to remote operations.

INDEX TERMS Distributed quantum computing (DQC), quantum compilation, quantum Internet.

I. INTRODUCTION

Noisy intermediate-scale quantum (NISQ) processors are characterized by a few hundred quantum bits (qubits) with nonuniform quality and highly constrained physical connectivity. Hence, the growing demand for large-scale quantum computers is motivating research on distributed quantum computing (DQC) architectures [1] as a scalable approach for increasing the number of available qubits for computational tasks, and experimental efforts have demonstrated some of the building blocks for such a design [2]. Indeed, with the network and communications functionalities provided by the Quantum Internet [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], remote quantum processing units (QPUs) can communicate and cooperate for executing computational tasks that each NISQ device cannot handle by itself.

In general, when moving from local to distributed quantum computing one faces two main challenges, namely, quantum algorithm partitioning and execution management [1]. To partition a monolithic quantum algorithm, a *quantum compiler* must be used to find the best breakdown, i.e., the one that minimizes the number of gates that are applied to qubits stored at different devices. Such *remote gates* can be implemented by means of three communication primitives that we denote as `Teleport` [14] (quantum state teleportation), `Cat-Ent` (cat-entanglement), and `Cat-DisEnt` (cat-disentanglement) [15], [16]. These primitives require that an entangled state is consumed and a new one must be distributed between the remote processors through the quantum link before another interprocessor operation can be executed. Through these primitives, one can perform two

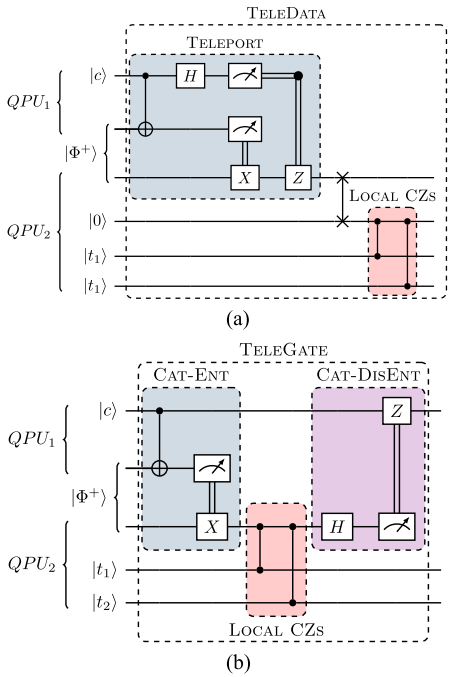


FIGURE 1. **(a) Circuit representation of `TeleData` by means of the `Teleport` primitive, which allows one to move the quantum state of a data qubit $|c\rangle$ to one qubit of an entangled pair and, then, swap it to a free data qubit. While both the state of the entangled pair and $|c\rangle$ are now lost, multiple CZ acting on the teleported qubit can then be executed. (b) Circuit representation of `TeleGate` by means of `Cat-Ent` and `Cat-DisEnt` primitives. After the `Cat-Ent` operation, the second qubit of the entangled pair participates in an entangled state with the control qubit and can act as a shared copy (not an actual copy, due to the no cloning theorem) of the original $|c\rangle$ control qubit. Multiple remote CZ with same control qubit and different target can be executed between `Cat-Ent` and `Cat-DisEnt`. It is worth noting that, between `Cat-Ent` and `Cat-DisEnt`, the control qubit is entangled with its shared copy and cannot be targeted by other gates.**

types of remote operations, namely `TeleData` and `TeleGate` [2], [17], [18], as shown in Fig. 1. The literature on quantum compilers [19], [20], [21], [22], [23], [24], [25], [26], [27], [28] focuses on qubit assignment and remote gate scheduling, while paying less attention to the integration of local routing.

Regarding execution management, in general, given a collection $\mathscr{P}$ of quantum circuit instances to be executed, this collection should be divided into nonoverlapping subsets $\mathscr{P}_i$, such that $\mathscr{P} = \cup_i \mathscr{P}_i$. One after the other, each subset must be assigned to the available QPUs. In other words, for each execution round i, there exists a schedule $S(i)$ that maps some quantum circuit instances to the quantum network. If DQC is supported, some quantum circuit instances may be split into subcircuit instances, each one to be assigned to a different QPU, as illustrated in Fig. 2.

In this work, we focus on the first challenge, i.e., quantum algorithm partitioning. We present a modular quantum compilation framework for DQC that, for the first time, takes into account both network and device constraints and characteristics. We illustrate the experimental evaluation of a quantum compiler based on the proposed framework, using

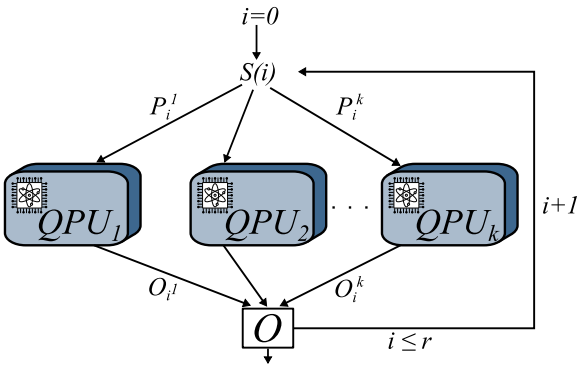


FIGURE 2. **Execution of multiple quantum circuit instances with k QPUs. For each execution round i, a schedule $S(i)$ maps some quantum circuit instances to the quantum network — each QPU receiving a quantum circuit P_i^j that is either a monolithic one or a subcircuit of a monolithic one. The classical outputs are accumulated into an output vector O.**

some circuits of interest (VQE, QFT, graph state preparation) and different network topologies, with quantum processors characterized by heavy hexagon coupling maps. The heavy-hexagon topology has been chosen by IBM [29] for its scalability and performance, offering reduced error-rates while affording the opportunity to explore error correcting codes [25], [30], [31], [32], [33]. We also devised a strategy for remote scheduling that can exploit both `TeleGate` and `TeleData` operations, and tested the impact of using either only `TeleGates` or both. The evaluation results show that `TeleData` operations can have a positive impact on the number of consumed Einstein–Podolsky–Rosen (EPR) pairs, depending on the characteristic of compiled circuit. Meanwhile, choosing a more connected network topology helps reduce the number of layers dedicated to remote operations.

The rest of this article is organized as follows. In Section II, related works on quantum compiling for DQC are discussed. In Section III, the proposed modular quantum compilation framework is illustrated in detail. In Section IV, the experimental evaluation of a Python-based implementation of the compiler is presented. Finally, Section V concludes this article.

II. RELATED WORK

Quantum compilation for DQC is characterized by two fundamental steps, *qubit assignment* and *remote gate scheduling*. In DQC, qubit assignment is generally tackled as a partitioning problem. Specifically, for a given set of virtual qubits, one needs to choose a partition that maps subsets of logical qubits to processors, while minimizing the number of required interactions among different subsets. The main goal is to minimize the number of consumed EPR pairs shared between QPUs, as it is the main bottleneck to distributed quantum computation.

Andrés-Martínez and Heunen [19] used *cat-entanglement* [16] to implement remote quantum gates. With this technique, one can implement multiple contiguous nonlocal CZ gates that act on a common qubit, while using only one

EPR pair. The chosen gate set contains every one-qubit gate and two single two-qubit gates, namely the CNOT and the CZ gate (i.e., the controlled version of the Z gate). The authors considered no restriction on the connectivity between QPUs. Then, they reduced the problem of distributing a circuit across multiple QPUs to hypergraph partitioning. The proposed approach, which was evaluated against five quantum circuits (including QFT), presents a caveat, in particular there is no way to customize the number of communication qubits of each QPU.

Sundaram et al. [20] presented a two-step solution, where the first step was qubit assignment. Circuits were represented as edge-weighted graphs with qubits as vertices. The edge weights corresponded to an estimation for the number of cat-entanglement operations. The problem was then solved as a minimum k-cut, where partitions had roughly the same size. The second step was finding the smallest set of cat-entanglement operations that would enable the execution of all `TeleGate`s. The authors started with the assumption that each remote gate could be executed, by means of a one cat-entanglement, only in the partition of one of its operands. In this setting, the problem could be reduced to a vertex-cover problem over bipartite graphs, allowing for a polynomial-time optimal solution based on integer linear programming. They also provided a $O(\log n)$-approximate solution, where n was the total number of global gates, for a generalized setting, where remote gates could be executed on an intermediary partition, by means of greedy search algorithm. In [21], the same authors extended their approach to the case of an arbitrary-topology network of heterogeneous quantum computers by means of a Tabu search algorithm.

In [22], by Daei et al., the circuit becomes an undirected graph with qubits as vertices, while edge weights correspond to the number of two-qubit gates between them. Then, the graph is partitioned using the Kernighan–Lin (K–L) algorithm for VLSI design [34], so that the number of edges between partitions is minimized. Finally, each graph partition is converted to a quantum circuit.

In [23], the authors represented circuits as bipartite graphs with two sets of vertices—one set for the qubits and one for the gates—and edges to encode dependencies of qubits and gates. Then, for the qubit assignment problem, they proposed a partitioning algorithm via dynamic programming to minimize the number of `TeleData` operations.

Nikahd et al. [24] formulated the minimum k-cut partitioning problem as an ILP optimization problem, to minimize the number of remote interactions. They employed a moving time-window and applied the partitioning algorithm to small sections of the circuit, thus the partition might change with the moving window by means of `TeleData` operations.

In [25], Cuomo et al. modeled the compilation problem with an integer linear programming formulation. The formulation is inspired to the vast theory on dynamic network problems. The authors manage to define the problem as a generalization of *quickest multicommodity flow*. This result allows to perform optimization by means of techniques coming from the literature, such as a *time-expanded* representation of the distributed architecture.

Ovide et al. [26], investigated the performance of the qubit assignment strategy proposed by Baker et al. [27] on the Cuccaro and QFT adder circuits [35], [36], under the assumption of local and network all-to-all connectivity. In [27], qubit assignment was treated as a graph partitioning problem, under the assumption that a SWAP operation primitive exists to exchange data qubits between different QPUs, i.e., it is not required to check if there are free data qubits available on the QPUs. Ovide et al. showed that, in general, the wider the circuit, the higher the number of remote operations, although it highly depends on the specific circuit to be compiled.

III. MODULAR QUANTUM COMPILATION FRAMEWORK

As mentioned in Section I, there is a lack of a modular framework for compiling quantum circuits to DQC architectures. Such a framework should be circuit agnostic, i.e., able to compile any circuit to any suitable DQC architecture. Current proposals from the literature tackle the problems of qubit assignment and remote gate scheduling, but do not take into account the local connectivity of each QPU. The framework presented in this article and illustrated in Fig. 3 fills the gap between local compilation and compilation for DQC. The proposed framework is modular, meaning that each module tackles a different aspect of the quantum compilation problem for DQC. Within the framework, modules do not depend on the specific implementation of the others, but only on their functionality and outputs, meaning that one could use a better implementation of one module without changing any other module.

The quantum compilation framework can take any quantum circuit and any network configuration as input. Fig. 4 depicts an example of a network configuration, which describes how QPUs are connected into the target DQC architecture, including quantum channels capacity, i.e., the number of communication qubits for each channel. The network configuration includes descriptions of the internal configurations of the QPUs, i.e., the coupling map and the set of available data qubits and communication qubits. More specifically, data qubits are a qubit subset dedicated to computational tasks, while communication qubits are another subset reserved for entanglement generation over the network [37]. The coupling map may have any shape. The coupling map is a directed graph where each vertex corresponds to a qubit and directed edges determine the possibility of executing two-qubit gates between the connected qubits.[1] Fig. 5 shows an example of a coupling map with 20 data qubits and 8 communication qubits, highlighted in blue.

Having these inputs, the first module of the framework regards the qubit assignment. Once a suitable qubit assignment

[1]Specifically, the source and destination vertices of a directed edge can be the control and target qubit respectively of a two-qubit gate. An edge could be undirected, meaning that both qubits can act as control or target.

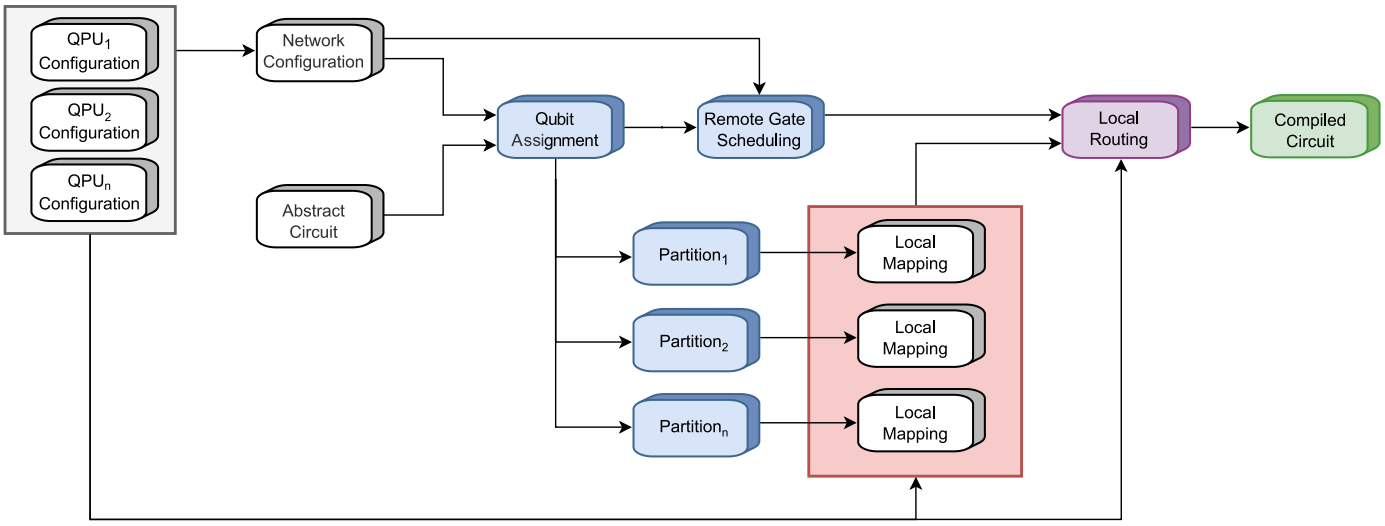


FIGURE 3. Workflow of the proposed modular quantum compilation framework for DQC architectures.

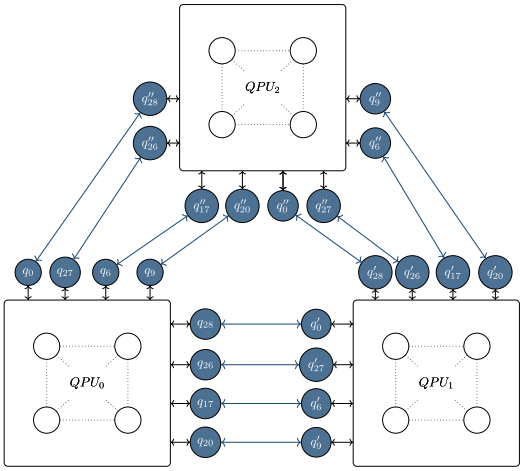


FIGURE 4. DQC architecture comprising three of the QPUs illustrated in Fig. 5. Each QPU is connected to the others and supports up to 4 communication qubits per connection.

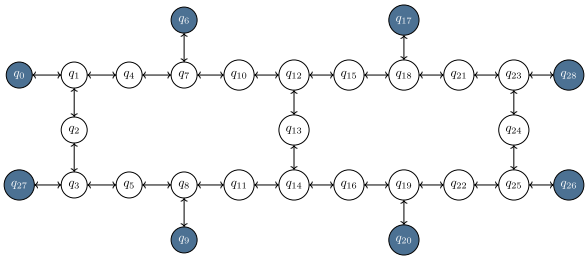


FIGURE 5. QPU configuration with 20 data qubits and 8 communication qubits, inspired by IBM's heavy hexagon devices [38].

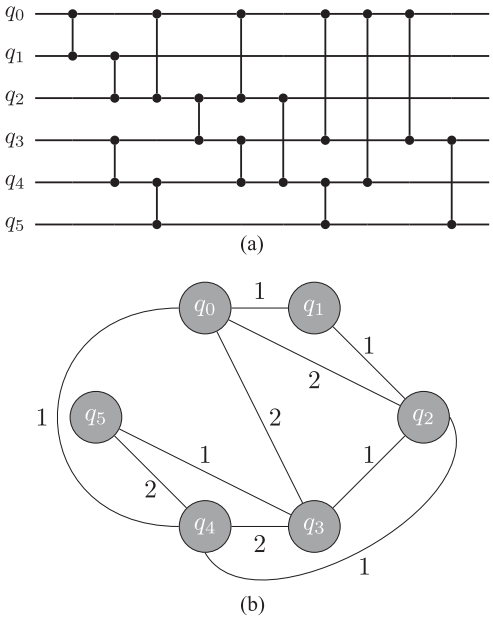


FIGURE 6. The quantum circuit in (a) can be represented as the weighted graph in (b).

has been found, the next module schedules remote gates accordingly. Meanwhile, the local mapping of qubits assigned to each QPU is performed. Finally, the last module performs local routing while taking into account the given schedule for remote gates and the local mappings.

The output of the compiler based on the proposed framework is a compiled circuit for DQC, containing all the scheduled local and remote gates.

In the following subsections, we propose some possible implementations of the qubit assignment, remote gate scheduling, and local routing modules.

A. QUBIT ASSIGNMENT

As mentioned in Section I, the goal is to partition the circuit in order to minimize the communication cost, i.e., the number of remote operations and consequently, the number of consumed EPR pairs. To this aim, a quantum circuit qc can be represented as an undirected weighted graph $G_{qc}(V, E)$, as shown in Fig. 6, where each edge $e \in E$ has weight $W(e) \in$

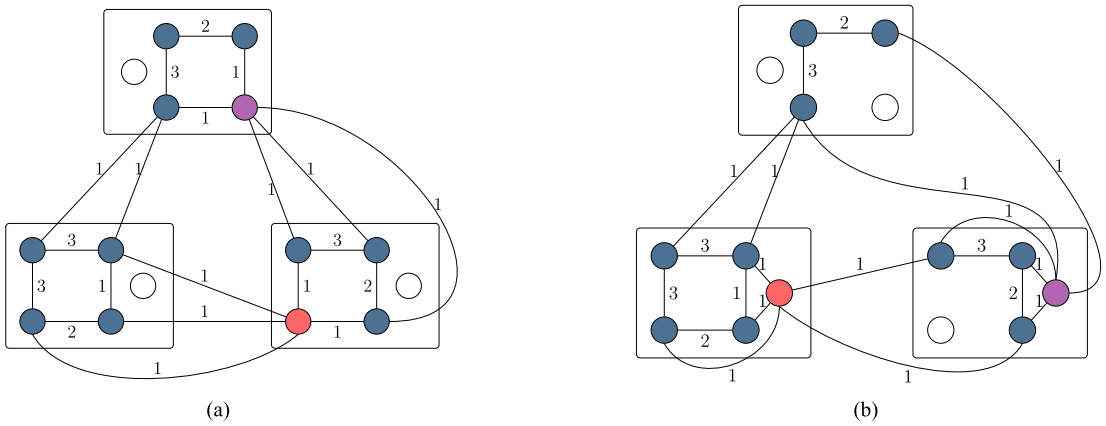


FIGURE 7. **(a) Example of initial graph partitioning. There are three partitions, each holding 4 qubits, and 8 edges between different partitions. The total communication cost is equal to 8. White nodes represent available qubits that have not been utilized. (b) Graph partitioning where qubits have been moved to achieve a better solution. The new total communication cost is now instead equal to 6.**

$\mathbb{N}$. The set of vertices V corresponds to the qubits in qc and the weight of each edge is equal to the number of two-qubit gates between the corresponding qubits.

The qubit assignment problem can then be treated as a graph partitioning problem where the objective is to compute a k-way partitioning such that the sum of inter-partition edge weights is minimized. Given k available QPUs, the result of k-way partitioning are k roughly equally sized circuit partitions. There are several algorithms available that can efficiently find a solution. In this work, METIS's multilevel k-way partitioning [39] was used. This is an initial solution and not an optimal one, as the QPUs would probably be underutilized and the circuit's qubits unnecessarily scattered through all QPUs. In fact, for each partition it is checked if changing the assignment of a qubit to another partition would benefit the overall communication cost. Between all the useful changes found, the best one is chosen and iteratively the search for possible changes continues, until either all qubits have changed partition one time or no further good improvements can be found. An example of improvement from the initial solution is depicted in Fig. 7. Regarding the computational complexity, we could not find a rigorous analysis of the METIS's algorithm complexity but the authors presented a technical report [40] showing outstanding performance when partitioning graphs with millions of nodes and edges. Focusing on the initial solution improvement, the algorithm needs to iterate over all qubits and partitions. In the worst case scenario, the complexity of this step is $O(n^3 p)$, where n is the number of qubits in the circuit and p the number of partitions.

B. REMOTE GATE SCHEDULING

We implemented an algorithm to schedule remote gates for DQC architectures in order to investigate the impact of using both `TeleData` and `TeleGate` operations. The main strategy, described in Algorithm 1, requires three inputs: 1) the quantum circuit to distribute; 2) the configuration of the network onto which such circuit will be executed; and 3) a suitable qubit assignment, as computed by in the previous module, described in Section III-A. To schedule remote gates, the algorithm described in Algorithm 1 scans the quantum circuit gate by gate and stops when it encounters a gate that, based on the current partitioning, involves qubits on different QPUs. The algorithm then searches for feasible `TeleData` operations that could cover the gate under consideration.[2] `TeleData` operations are scheduled by taking into account the memory capacity of each QPU, otherwise data qubits storing valuable information would get overwritten by a teleportation. Finally, to each possible `TeleData` is assigned the following cost function:

$$\frac{n_{EPR}}{n_{cov}} \frac{delay}{\bar{d}_t} \tag{1}$$

where n_{EPR} is the number of consumed EPR pairs, n_{cov} is the number of covered gates, which may include more gates than the one under direct analysis—as shown in Fig. 1(a)—and $delay$ is the time, measured in discrete intervals, that must be waited before actually executing the gate. The $delay$ is estimated by the cost function, based on when the quantum links for entanglement generation were last used and when the gate should be executed. It may be the case that before executing a `TeleData` operation, one needs to wait for a previously scheduled one to complete. The aim is to evaluate possible remote operations against resource consumption (n_{EPR}) and execution time ($delay$), both crucial for the performance of the system, with respect to the number of covered gates (n_{cov}). The $delay$ is scaled with the mean decoherence time $\bar{d}_t$ of the physical qubits, turning it into a dimensionless number.

The algorithm selects the `TeleData` operation with the lowest cost and then, covers a portion of the following gates[3] with `TeleGate` operations. `TeleGate` operations are chosen and scheduled in a similar manner to the `TeleData` ones. `TeleGates` exploit the `Cat-Ent` primitive, as

[2]Here, "to cover" means to make a gate executable.
[3]Dimension of the portion to cover is set by a customizable parameter.

Algorithm 1: Remote Gates Scheduler.

Input: quantum circuit QC, network configuration N and qubit assignment P
Output: quantum circuit with remote gates D

```
function SCHEDULE
D ← ∅
covered ← ∅
for all g ∈ QC do
  if g ∉ D then
    if g is local then
      put g into covered and D
    else
      TeleData ← FIND TELEDATA(g, N, P)
      TeleGate ← FIND TELEGATE(g, N, P)
      if COST(Teledata) < COST(TeleGate) then
        put TeleData into D
      else
        put TeleGate into D
      end if
      put g into covered and D
      put extra covered gates into D and covered
    end if
  end if
end for
end function
```

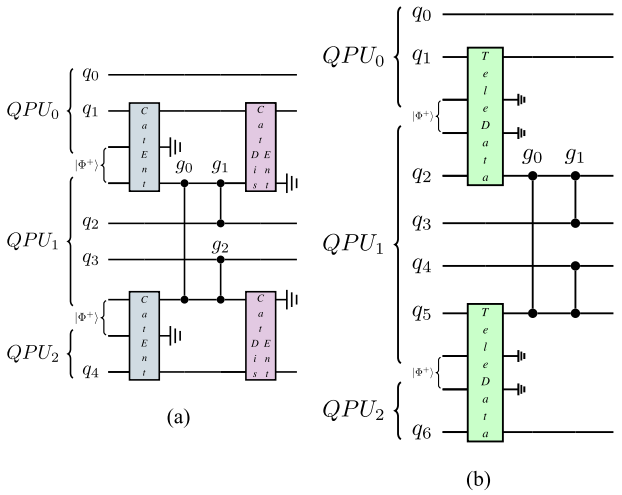


FIGURE 8. (a) Gate g_0 is covered by sharing qubit q_1 and q_4 with QPU_1 using two TeleGates. By doing this, also gates g_1 and g_2 are covered. (b) Qubits q_1 and q_6 are teleported to QPU_1 with two TeleDatas, gates g_0, g_1, and g_2 are consequently covered.

shown in Fig. 1(b) [16]. Indeed, TeleGates can be divided in three steps. First, with the Cat-Ent primitive, the one in the blue box in Fig. 1(b), the control qubit of a remote gate is entangled with a communication qubit on the QPU holding the target qubit. This qubit is called a "shared" control. Then, gates controlled by the same shared control qubit can be be executed locally. Finally, with the Cat-DisEnt primitive, the one in the violet box in Fig. 1(b), the shared control qubit is measured and a correction operation is applied to the original control qubit at the first QPU.

Both TeleGate and TeleData can either migrate one qubit to the other's QPU or both to a different one, as shown in Fig. 8, depending on the cost of such operation (computed as in (1)). Fig. 8(a) shows the first case, in which gate g_0 is covered by sharing qubit q_1 and q_4 with QPU_1 through two TeleGate operations. Fig. 8(b) depicts the second case, where gate g_0 is covered by sharing qubits q_1 and q_6 with QPU_1, using two TeleDatas. By doing this, also gates g_1 and g_2 are covered.

The scheduler also compiles the same portion of circuit by scheduling only TeleGate operations. Finally, it computes a cost for the two different strategies—one with TeleData and TeleGate, the other with just TeleGate—and selects the one with the lowest amount of consumed EPR pairs. Finally, the scheduler resumes scanning gates in search of the next gate to cover. Given that the scheduler has to search for both TeleGate and TeleData operations to cover each remote gate, and that we also take into account a portion of subsequent gates, the worst case computational complexity of this compilation module can be estimated as $O(r^3 p)$, with r being the number of remote operations and p the number of partitions.

C. LOCAL ROUTING

The designed local routing algorithm takes as input a partitioned circuit with already scheduled remote operations and handles the local routing accordingly. The algorithm requires the partitioned circuit, with scheduled remote operations, the network configuration, and all QPU configurations, specifically coupling maps including connections between data qubits and communication qubits.

The core strategy is the following: The algorithm scans the circuit and for every gate that involves qubits not directly connected on their specific QPU, computes the shortest sequence of necessary SWAP gates. When it encounters a TeleData or TeleGate operation, it first checks if the involved data qubits are in proximity of one of the available communication qubits. If not, it computes the shortest paths to the less recently used communication qubit. The less recently used communication qubit is chosen to avoid as much delay as possible in entanglement generation. At this stage of compilation, due to local SWAPs, the state of a data qubit may now reside on a communication qubit and vice versa. This is not necessarily an issue [41], but it is better to move the communication qubit back to its original position, after the remote operation is completed and before it is used again. This is crucial to not lose the state of a data qubit physically stored at a communication qubit location, due to a new remote operation. An example of such instance is shown in Fig. 9. The computational complexity of this local routing implementation depends directly on the number of two-qubit gates and remote operations, and can

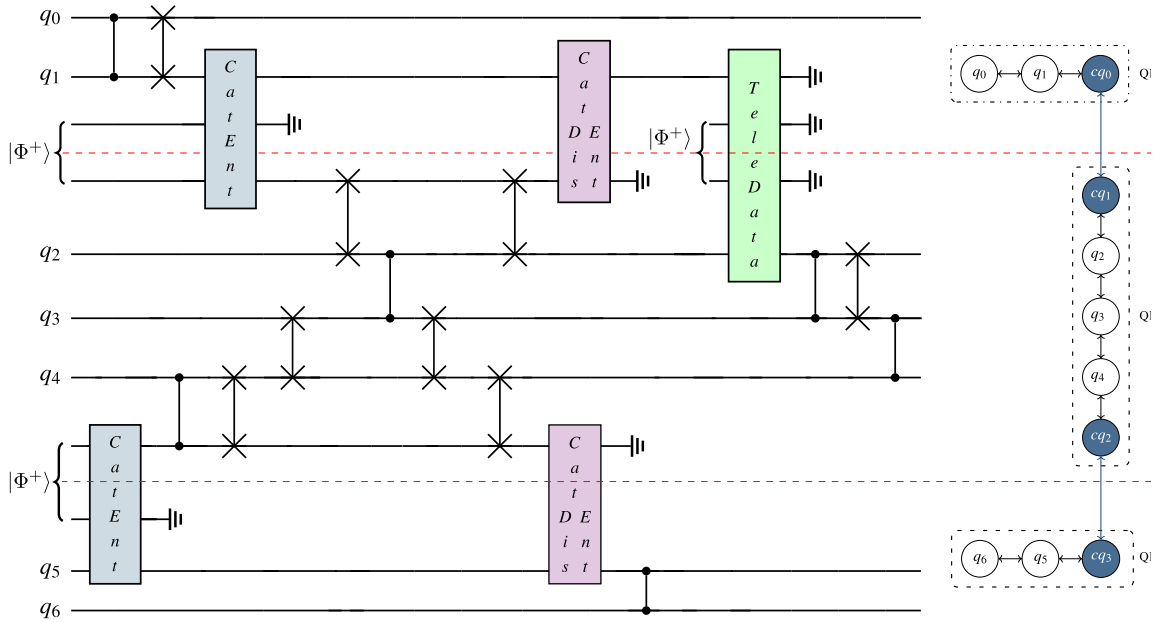


FIGURE 9. Example of remote gate scheduling and local routing. Local gates are interlaced with `Cat-Ent`, `Cat-DisEnt`, and `TeleData` operations as well as `SWAP` gates.

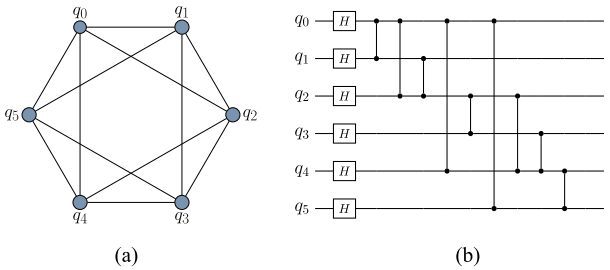


FIGURE 10. (a) Example of graph state used to create graph state circuits. Each vertex represents a qubit in the graph state, and there is an edge between every pair of interacting qubits. This graph state has 6 qubits, but can be scaled up to an arbitrary large number of qubits. (b) Quantum circuit to create the 6-qubit graph state represented in (a).

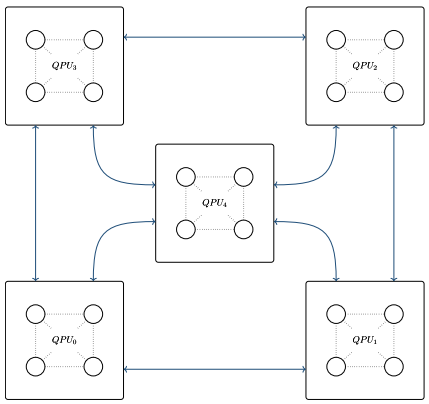


FIGURE 11. DQC architecture comprising five QPUs. Each QPU is connected to at least three other QPUs.

be estimated as $O((g+r)n\log n)$. This result stems form the fact that in the worst case scenario, for each two-qubit gate g and remote operation r, the algorithm uses Dijkstra's shortest path algorithm with a complexity of $O(n\log n)$ on coupling map with n qubits.

IV. EVALUATION

We implemented a quantum compiler based on the modular framework presented in Section III. The compiler was tested against three classes of quantum circuits, namely, VQE, QFT, and graph state circuits (an example of graph state used is shown in Fig. 10). QFT and VQE were selected because they are widely used in many different contexts. Together with the Grover Operator (GO) [42] and the Harrow/Hassidim/Lloyd (HHL) method for linear systems [43], these circuits cover most practical scenarios [44]. The graph state circuit was included as graph states are relevant for applications like clock synchronization, secure communication, distributed sensing, distributed one-way quantum computations [45].

The purpose of the proposed evaluation is twofold. On the one hand, showing that the proposed framework is modular and flexible enough to allow the user for testing different compilation strategies against multiple network topologies. On the other hand, illustrating the specific impact of different configurations on the considered quantum circuits. For example, it will be shown that QFT circuit compilation benefits from telegate+teledata, but heavily relies on remote operations, while VQE circuit compilation does not benefit from teledata, but requires only a handful of remote operations.

The circuits were compiled for the DQC architecture illustrated in Figs. 4 and 11, comprised, respectively, of 3 and 5 QPUs, denoted as *Net-3* and *Net-5*. To increase the number

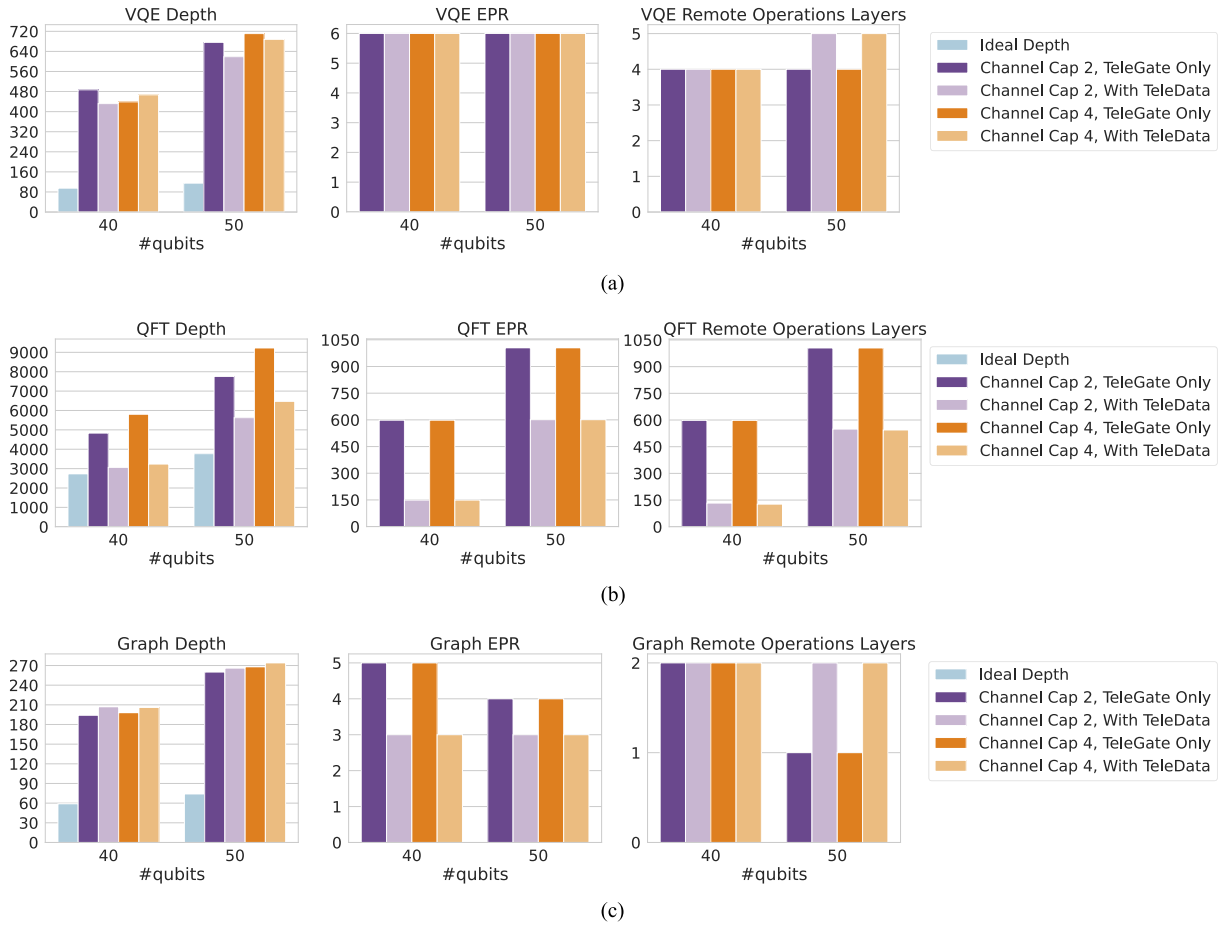


FIGURE 12. Compilation results for the Net-3 with QPU-21 of (a) VQE, (b) QFT, and (c) Graph circuits. The number of qubits of the circuits varies from 40 to 50, while the channel capacity varies from 2 to 4.

of available data qubits, the QPUs in Fig. 5 can be scaled up in a modular fashion [29]. In the following, *QPU-n* denotes a QPU with *n* data qubits.

The assumed architectures are consistent with current superconducting quantum devices (such as those produced by IBM [46] and Google [47]), but also with other physical solutions, like NV centers [48] and trapped ions [49]. For all these technologies, there are promising results concerning the possibility of realizing quantum links between separated devices [50], [51].

For each quantum circuit, the tests concerned remote gate scheduling with only the `TeleGate` operation, as well as with both the `TeleGate` and `TeleData` operations. For each compiled circuit the depth,[4] the number of EPR pairs consumed and the layers dedicated to remote operations were used to analyze the results.

Fig. 12 shows compilation results of VQE, QFT, and Graph circuits on Net-3 with QPU-21. It can be seen that exploiting `TeleData` operations alongside `TeleGates` tends to be more beneficial, in terms of total depth, especially for QFT circuits. Moreover, using `TeleData` operations can greatly reduce the number of EPR pairs consumed and layers dedicated to remote operations for QFT circuits, up to more than 50%, while being slightly detrimental for VQE circuits. This behavior can be traced back to the specific structure of these circuits. Specifically, QFT circuits are characterized by multiple sequences of two-qubit gates involving almost all qubits in each sequence and having one common control qubit that changes in each sequence, as depicted in

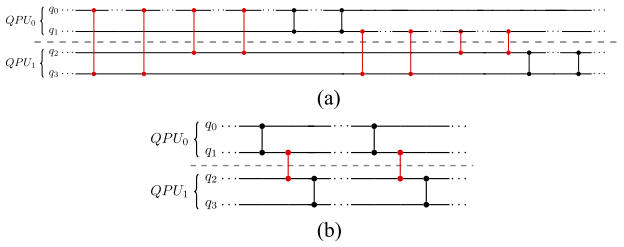


FIGURE 13. (a) Example of QFT circuit with four qubits distributed on two QPUs. (b) Example of VQE circuit with four qubits distributed on two QPUs.

[4]Depth of a quantum circuit is a measure of how many layers of quantum gates, executed in parallel, it takes to complete the computation defined by the circuit.

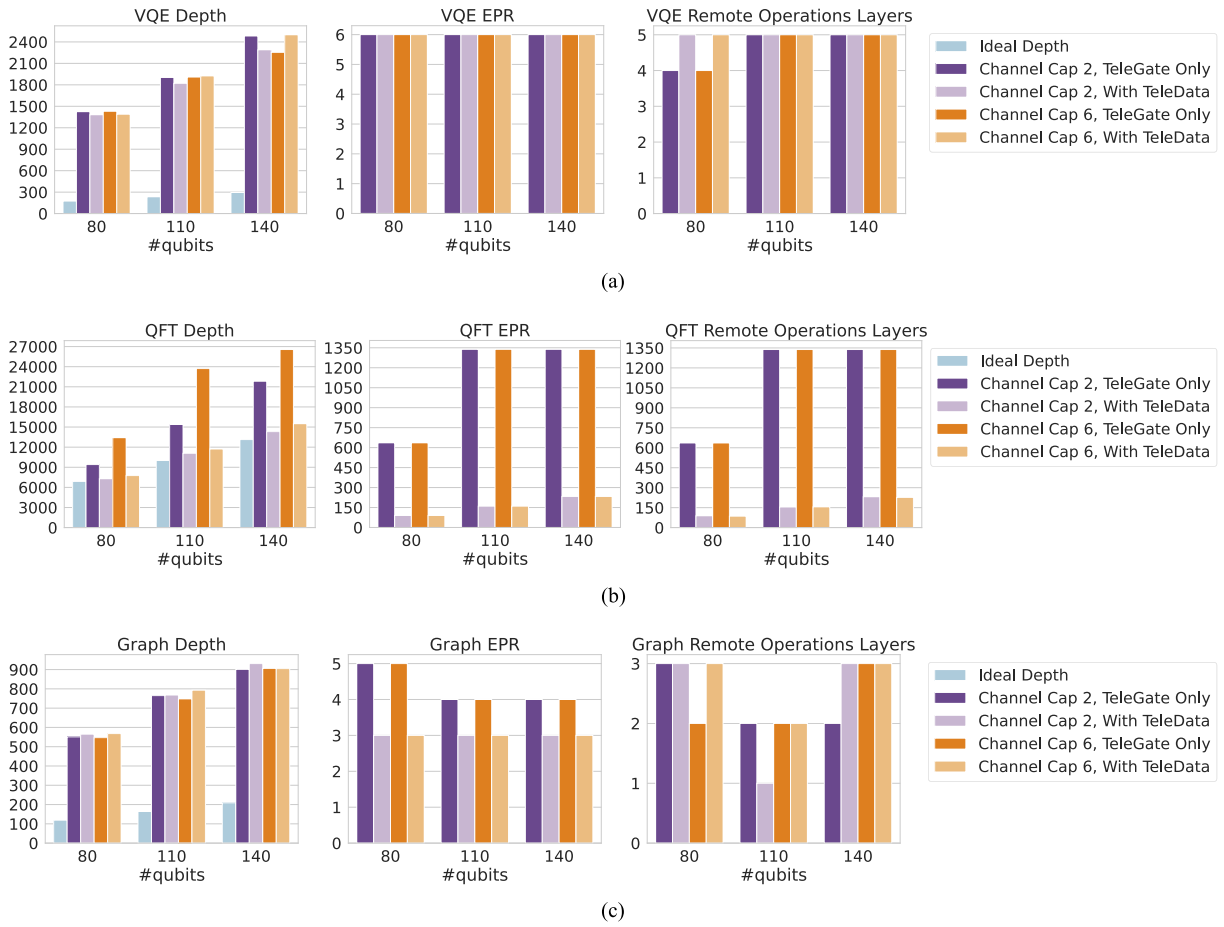


FIGURE 14. Compilation results for the Net-3 with QPU-63 of (a) VQE, (b) QFT, and (c) Graph circuits. The number of qubits of the circuits varies from 80 to 140, while the channel capacity varies from 2 to 6.

Fig. 13(a). If a sequence involves remote operations, it is better to teleport the control qubit and execute all subsequent two qubit gates in the sequence locally. On the other hand, the VQE circuits used are characterized by chains of two qubit gates, such that each qubit in the chain acts as the control of a gate and the target of the subsequent gate, as shown in Fig. 13(b). This means that there will be only a handful of remote operations and there is no need to teleport a control qubit. Regarding Graph state circuits, there is an increase of one unit in the remote operations layers, as shown in Fig. 12(c), which is opposed to a decrease of also one unit in the number of EPR consumed, when using `TeleData` operations.

All figures show a slight increase in total circuit depth when the channel capacity changes from 2 to 4. At first glance, this may seem counter intuitive. Indeed, it is worth pointing out that the adopted local routing does not change the number of consumed EPR pairs defined by the qubit assignment and remote scheduling modules, but may increase the number of remote operation layers. The reason is that the local routing tries to use all available communication qubits, regardless of their distance from data qubits in the local coupling map. Designing a different strategy is out of the scope of this article, whose purpose is to illustrate a modular framework and show the multiplicity of circuits and network configurations that can be evaluated by means of that framework. From previous work, we know that local routing could be improved by means of circuit optimization techniques [52] and noise-aware strategies [53]. Interestingly, there seems to be no difference in the number of layers dedicated to remote operations with respect to the channel capacity. We suppose that, due to the low connectivity between data qubits and communication qubits on each QPU, local routing operations create an upstream bottleneck with deleterious effects despite the increase in channel capacity.

Some tests were also made using Net-3 with QPU-63 devices, with the number of data qubits used by the circuits varying between 80, 110, and 140. The results are reported in Fig. 14. While there is still not much of a difference when changing the channel capacity, the use of `TeleData` operations is greatly beneficial when distributing QFT circuits, which, from the number of EPR consumed, appear to be the circuit class that more heavily depends on remote operations, among those tested.

By maintaining Net-3 but changing to QPU-125 devices, the compiler was tested on circuits with up to 250 qubits.

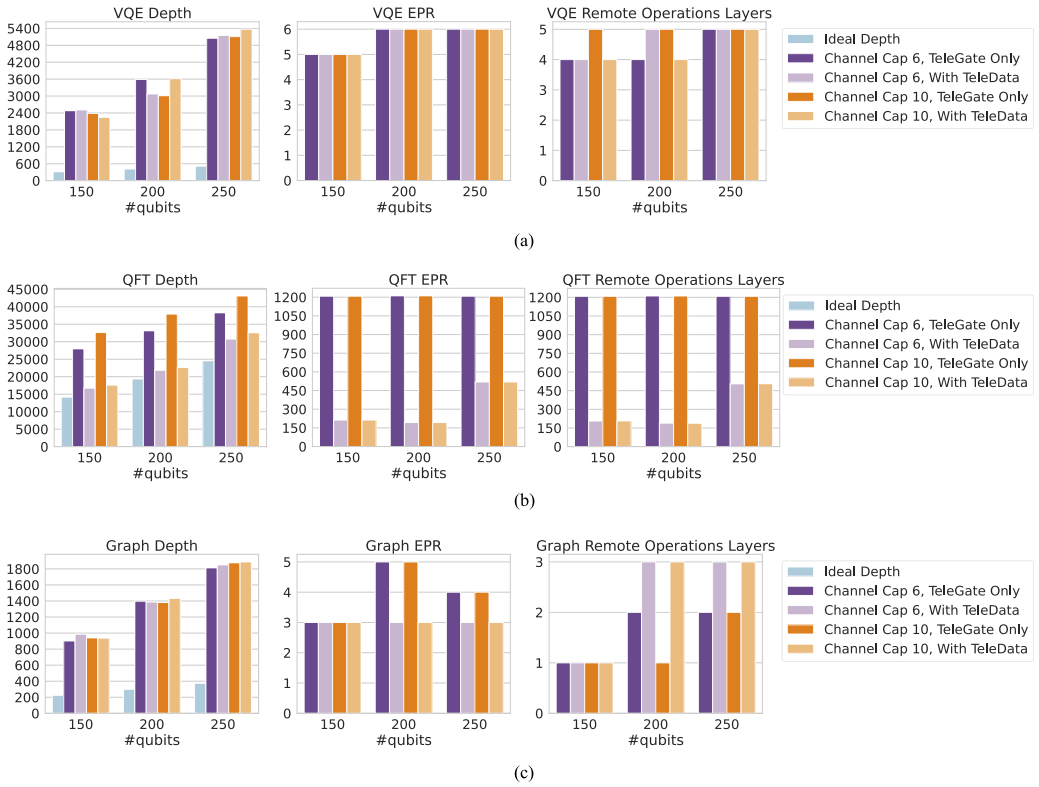


FIGURE 15. Compilation results for the Net-3 with QPU-125 of (a) VQE, (b) QFT, and (c) Graph circuits. The number of qubits of the circuits varies from 150 to 200, while the channel capacity varies from 6 to 10.

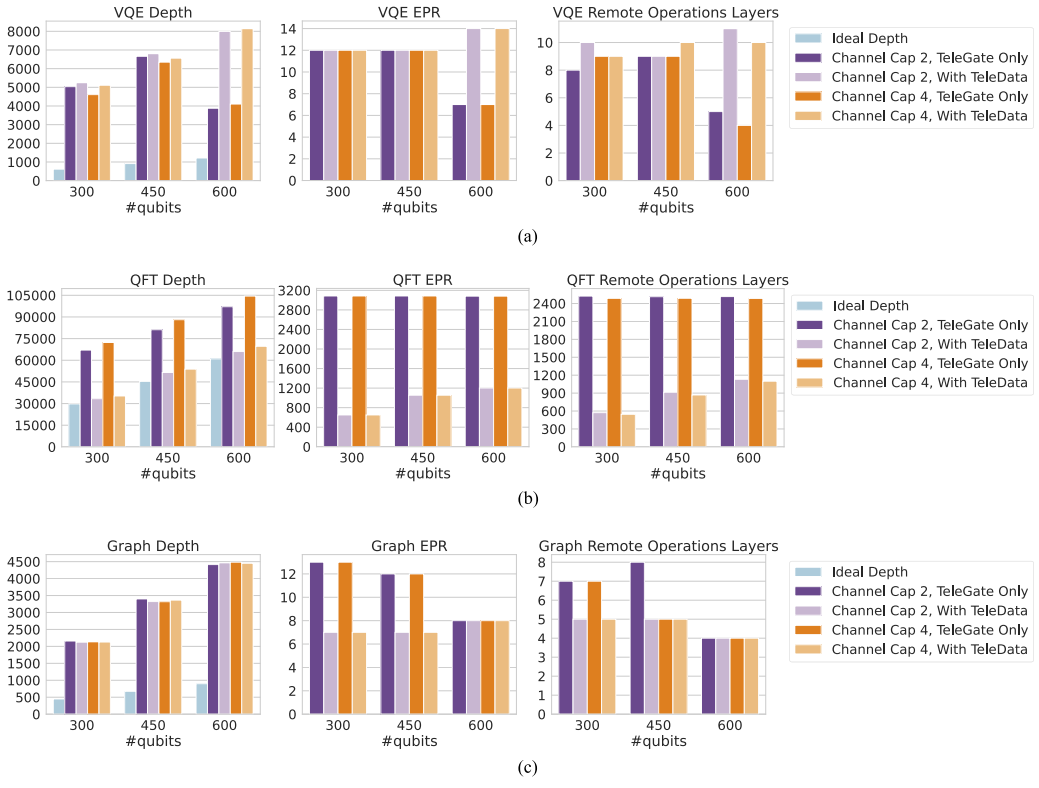


FIGURE 16. Compilation results for the Net-5 with QPU-125 of (a) VQE, (b) QFT, and (c) Graph circuits. The number of qubits of the circuits varies from 300 to 600, while the channel capacity varies from 2 to 4.

At this stage, an interesting observation can be made from Fig. 15. It seems that, for Graph states circuits, when the number of qubits grows, and the data qubits capacity of the network is topped up, while the number of EPR pairs consumed remains unchanged, there is an almost unnoticeable increase in the number of layers for remote operations.

Finally, the total number of data qubits was further increased, by exploiting Net-5 with QPU-125 devices. Therefore, it was possible to compile circuits up to 600 qubits, as depicted in Fig. 16. There are two results that stand out in these figures. Firstly, for VQE circuits, there is a slight increase in the layers of remote operations when the maximum number of qubits is reached and `TeleData` operations are employed. The opposite can be observed for Graph state circuits, where the number of remote operations layers decreases, although marginally, when the maximum number of data qubits allowed by the network is reached. This trend goes against the observation made previously for the same type of circuits albeit with the Net-3 topology, which outlines the impact of different network topologies and suggests that choosing a more connected network is in fact beneficial.

A final remark should be made regarding the general difference between the results for different classes of circuits. Although the QFT circuits greatly benefit from the use of `TeleData`, they still need a massive amount of remote operations. In contrast, the VQE and Graph circuits, which are not particularly affected by the use of `TeleData`, require very few remote operations for their distributed execution, which makes them better candidates for DQC.

V. CONCLUSION

In this work, we introduced a general-purpose modular quantum compilation framework for DQC that takes into account both network and device constraints and characteristics. We illustrated the experimental evaluation of a quantum compiler based on the proposed framework, using some circuits of interest (VQE, QFT, graph state preparation) characterized by different widths (up to 600 qubits). We considered different network topologies, with quantum processors characterized by heavy hexagon coupling maps. We also presented a strategy for remote scheduling that can exploit both `TeleGate` and `TeleData` operations, and tested the impact of using either only `TeleGates` or both operations. We observed that `TeleData` operations may have a positive impact on the number of consumed EPR pairs, depending on the specific characteristics of the circuit. In fact, we also observed that some classes of circuits are more suitable for DQC than others, i.e., they can be distributed more efficiently. Furthermore, we showed that choosing a more connected network topology helps reduce the number of layers dedicated to remote operations.

Regarding future work, we will focus on integrating noise-adaptive compilation strategies into the framework, both for local routing [53] and remote gate scheduling. We shall then evaluate the impact of different strategies on the quality of computation results, which depend also on the selection of suitable metrics. To produce such metrics we need to actually execute the compiled circuits, either by means of a quantum network simulator or on real hardware. In the first case, there are already available simulators with different levels of abstraction, depending on how realistic the simulations need to be. These simulations will be crucial to understand the impact that remote operations, and any resulting local routing overhead, have on the quality of the computation due to the effects of noise.

ACKNOWLEDGMENT

This research benefits from the High Performance Computing facility of the University of Parma, Italy.

DATA AVAILABILITY

All data and code required to reproduce all plots shown herein are available at https://doi.org/10.5281/zenodo.7896588.

REFERENCES

[1] M. Caleffi et al., "Distributed quantum computing: A survey," 2022, *arXiv:2212.10609*, doi: 10.48550/arXiv.2212.10609.

[2] R. V. Meter and S. J. Devitt, "The path to scalable distributed quantum computing," *Computer*, vol. 49, no. 9, pp. 31–42, Sep. 2016, doi: 10.1109/MC.2016.291.

[3] S. Pirandola and S. L. Braunstein, "Physics: Unite to build a quantum internet," *Nature*, vol. 532, no. 7598, pp. 169–171, Apr. 2016, doi: 10.1038/532169a.

[4] E. Gibney, "Chinese satellite is one giant step for the quantum internet," *Nature*, vol. 535, no. 7613, pp. 478–479, Jul. 2016, doi: 10.1038/535478a.

[5] W. Dür, R. Lamprecht, and S. Heusler, "Towards a quantum internet," *Eur. J. Phys.*, vol. 38, no. 4, May 2017, Art. no. 043001, doi: 10.1088/1361-6404/aa6df7.

[6] C. Simon, "Towards a global quantum network," *Nature Photon.*, vol. 11, no. 11, pp. 678–680, 2017, doi: 10.1038/s41566-017-0032-0.

[7] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, 2018, Art. no. eaam9288, doi: 10.1126/science.aam9288.

[8] M. Zomorodi-Moghadam, M. Houshmand, and M. Houshmand, "Optimizing teleportation cost in distributed quantum circuits," *Int. J. Theor. Phys.*, vol. 57, pp. 848–861, 2018, doi: 10.1007/s10773-017-3618-x.

[9] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, "Quantum internet: From communication to distributed computing!," in *Proc. 5th ACM Int. Conf. Nanoscale Comput. Commun.*, 2018, pp. 1–4, doi: 10.1145/3233188.3233224.

[10] M. Caleffi, D. Chandra, D. Cuomo, S. Hasaanpour, and A. S. Cacciapuoti, "The rise of the quantum internet," *Computer*, vol. 53, no. 6, pp. 67–72, Jun. 2020, doi: 10.1109/MC.2020.2984871.

[11] L. Gyongyosi and S. Imre, "Entanglement concentration service for the quantum internet," *Quantum Inf. Process.*, vol. 19, no. 8, 2020, Art. no. 221, doi: 10.1007/s11128-020-02716-3.

[12] L. Gyongyosi and S. Imre, "Routing space exploration for scalable routing in the quantum internet," *Sci. Rep.*, vol. 10, no. 1, 2020, Art. no. 11874, doi: 10.1038/s41598-020-68354-y.

[13] M. Amoretti and S. Carretta, "Entanglement verification in quantum networks with tampered nodes," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 3, pp. 598–604, Mar. 2020, doi: 10.1109/JSAC.2020.2967955.

[14] A. S. Cacciapuoti, M. Caleffi, R. V. Meter, and L. Hanzo, "When entanglement meets classical communications: Quantum teleportation for the quantum internet," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3808–3833, Jun. 2020, doi: 10.1109/TCOMM.2020.2978071.

[15] J. Eisert, K. Jacobs, P. Papadopoulos, and M. B. Plenio, "Optimal local implementation of nonlocal quantum gates," *Phys. Rev. A*, vol. 62, Oct. 2000, Art. no. 052317, doi: 10.1103/PhysRevA.62.052317.

[16] A. Yimsiriwattana and S. Lomonaco, "Generalized GHZ states and distributed quantum computing," *Contemp. Math.*, vol. 381, pp. 131–147, 2005, doi: 10.1090/conm/381/07096.

[17] R. V. Meter, K. Nemoto, W. J. Munro, and K. M. Itoh, "Distributed arithmetic on a quantum multicomputer," in *Proc. 33rd Int. Symp. Comput. Architecture*, 2006, pp. 354–365.

[18] R. V. Meter, W. J. Munro, and K. Nemoto, "Architecture of a quantum multicomputer implementing Shor's algorithm," in *Theory of Quantum Computation, Communication, and Cryptography*, Y. Kawano and M. Mosca, Eds. Berlin, Heidelberg: Springer, 2008, pp. 105–114, doi: 10.1007/978-3-540-89304-2_10.

[19] P. Andrés-Martínez and C. Heunen, "Automated distribution of quantum circuits via hypergraph partitioning," *Phys. Rev. A*, vol. 100, Sep. 2019, Art. no. 032308, doi: 10.1103/PhysRevA.100.032308.

[20] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan, "Efficient distribution of quantum circuits," in *Proc. 35th Int. Symp. Distrib. Comput.*, 2021, pp. 41:1–41:20, doi: 10.4230/LIPIcs.DISC.2021.41.

[21] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan, "Distribution of quantum circuits over general quantum networks," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, 2022, pp. 415–425, doi: 10.1109/QCE53715.2022.00063.

[22] O. Daei, K. Navi, and M. Zomorodi-Moghadam, "Optimized quantum circuit partitioning," *Int J. Theor. Phys.*, vol. 59, no. 12, pp. 3804–3820, Dec. 2020, doi: 10.1007/s10773-020-04633-8.

[23] Z. Davarzani, M. Zomorodi-Moghadam, M. Houshmand, and M. Nouribaygi, "A dynamic programming approach for distributing quantum circuits by bipartite graphs," *Quantum Inf. Process.*, vol. 19, 2020, Art. no. 360, doi: 10.1007/s11128-020-02871-7.

[24] E. Nikahd, N. Mohammadzadeh, M. Sedighi, and M. S. Zamani, "Automated window-based partitioning of quantum circuits," *Physica Scripta*, vol. 96, no. 3, Jan. 2021, Art. no. 035102, doi: 10.1088/1402-4896/abd57c.

[25] D. Cuomo et al., "Optimized compiler for distributed quantum computing," *ACM Trans. Quantum Comput.*, vol. 4, no. 2, pp. 1–29, Feb. 2023, doi: 10.1145/3579367.

[26] A. Ovide et al., "Mapping quantum algorithms to multi-core quantum computing architectures," 2023, *arXiv:2303.16125*, doi: 10.48550/arXiv.2303.16125.

[27] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Time-sliced quantum circuit partitioning for modular architectures," in *Proc. 17th ACM Int. Conf. Comput. Front.*, 2020, pp. 98–107, doi: 10.1145/3387902.3392617.

[28] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi, "Compiler design for distributed quantum computing," *IEEE Trans. Quantum Eng.*, vol. 2, 2021, Art. no. 4100720, doi: 10.1109/TQE.2021.3053921.

[29] IBM, "The IBM Quantum heavy hex lattice," 2021. [Online]. Available:https://research.ibm.com/blog/heavy-hex-lattice

[30] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, and J. M. Gambetta, "Experimental demonstration of fault-tolerant state preparation with superconducting qubits," *Phys. Rev. Lett.*, vol. 119, Oct. 2017, Art. no. 180501, doi: 10.1103/PhysRevLett.119.180501.

[31] N. Sundaresan, I. Lauer, E. Pritchett, E. Magesan, P. Jurcevic, and J. M. Gambetta, "Reducing unitary and spectator errors in cross resonance with optimized rotary echoes," *PRX Quantum*, vol. 1, Dec. 2020, Art. no. 020318, doi: 10.1103/PRXQuantum.1.020318.

[32] A. D. Córcoles et al., "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature Commun.*, vol. 6, no. 1, Apr. 2015, Art. no. 6979, doi: 10.1038/ncomms7979.

[33] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, "Topological and subsystem codes on low-degree graphs with flag qubits," *Phys. Rev. X*, vol. 10, Jan. 2020, Art. no. 011022, doi: 10.1103/PhysRevX.10.011022.

[34] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Tech. J.*, vol. 49, no. 2, pp. 291–307, Feb. 1970, doi: 10.1002/j.1538-7305.1970.tb01770.x.

[35] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," 2004, *arXiv:quant-ph/0410184*, doi: 10.48550/arXiv.quant-ph/0410184.

[36] L. Ruiz-Perez and J. C. Garcia-Escartin, "Quantum arithmetic with the quantum fourier transform," *Quantum Inf. Process.*, vol. 16, no. 6, Apr. 2017, Art. no. 152, doi: 10.1007/s11128-017-1603-1.

[37] W. Kozlowski et al., "Architectural principles for a quantum internet," *Internet Eng. Task Force, Internet-Draft draft-irtf-qirg-principles-10*, 2022.

[38] IBM, "IBM quantum systems," 2020. [Online]. Available: https://quantum-computing.ibm.com/services/resources

[39] "Metis github repository," 2023. [Online]. Available: https://github.com/KarypisLab/METIS

[40] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1998, doi: 10.1137/S1064827595287997.

[41] A. Dahlberg et al., "NetQASM—a low-level instruction set architecture for hybrid quantum–classical programs in a quantum internet," *Quantum Sci. Technol.*, vol. 7, no. 3, Jun. 2022, Art. no. 035023, doi: 10.1088/2058-9565/ac753f.

[42] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[43] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, Oct. 2009, Art. no. 150502, doi: 10.1103/PhysRevLett.103.150502.

[44] A. Jayakumar et al., "Quantum algorithm implementations for beginners," *ACM Trans. Quantum Comput.*, vol. 3, no. 4, pp. 1–92, Jul. 2022, doi: 10.1145/3517340.

[45] C. Meignant, D. Markham, and F. Grosshans, "Distributing graph states over arbitrary quantum networks," *Phys. Rev. A*, vol. 100, Nov. 2019, Art. no. 052333, doi: 10.1103/PhysRevA.100.052333.

[46] Y. Kim et al., "Scalable error mitigation for noisy quantum circuits produces competitive expectation values," *Nature Phys.*, vol. 19, no. 5, pp. 752–759, May 2023, doi: 10.1038/s41567-022-01914-3.

[47] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, Feb. 2023, doi: 10.1038/s41586-022-05434-1.

[48] T. V. Der et al., "Decoherence-protected quantum gates for a hybrid solid-state spin register," *Nature*, vol. 484, no. 7392, pp. 82–86, Apr. 2012, doi: 10.1038/nature10900.

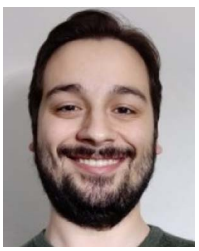
[49] P. Hrmo et al., "Native qudit entanglement in a trapped ion quantum processor," *Nature Commun.*, vol. 14, no. 1, Apr. 2023, Art. no. 2242, doi: 10.1038/s41467-023-37375-2.

[50] IBM, "Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing," 2022. [Online]. Available:https://research.ibm.com/blog/ibm-quantum-roadmap-2025

[51] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson, "Qubit teleportation between non-neighbouring nodes in a quantum network," *Nature*, vol. 605, no. 7911, pp. 663–668, May 2022, doi: 10.1038/s41586-022-04697-y.

[52] D. Ferrari, I. Tavernelli, and M. Amoretti, "Deterministic algorithms for compiling quantum circuits with recurrent patterns," *Quantum Inf. Process.*, vol. 20, no. 6, Jun. 2021, Art. no. 213, doi: 10.1007/s11128-021-03150-9.

[53] D. Ferrari and M. Amoretti, "Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks," in *Proc. 19th ACM Int. Conf. Comput. Front.*, 2022, pp. 237–243, doi: 10.1145/3528416.3530250.



Davide Ferrari (Member, IEEE) received the Ph.D. degree in information technologies from the University of Parma, Parma, Italy, in 2023.

Druring his doctoral studies, he worked on quantum compiling, quantum optimization, and distributed quantum computing. He was a Research Scholar with Future Technology Lab, University of Parma, working on the design of efficient algorithms for quantum compiling. He is currently a Research Fellow with the Department of Engineering and Architecture, University of Parma. He is involved in the Quantum Information Science research initiative with the University of Parma, where he is a member of the Quantum Software Laboratory. His research interests include quantum optimization applications and efficient quantum compiling for local and distributed quantum computing.

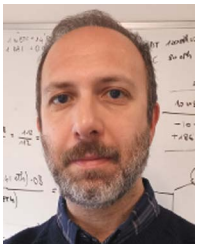
Dr. Ferrari was the recipient of the IBM Quantum Awards Circuit Optimization Developer Challenge in 2020.

IEEE Transactions on Quantum Engineering

Stefano Carretta received the Ph.D. degree in physics from the University of Parma, Parma, Italy, in 2005.

He is currently a Full Professor in physics of matter with the University of Parma. He contributed some of the first proposals for the use of magnetic molecules as qubits and the first proposal for exploiting molecular nanomagnets as quantum simulators. He has authored or coauthored more than 150 research papers published in international journals. His research interests include the theoretical modeling of the quantum behavior of magnetic molecules and quantum information processing.

Dr. Carretta was the recipient of the Le Scienze (the Italian version of Scientific American) medal and of the President of the Italian Republic medal for his research on molecular nanomagnets in 2006, and the prestigious Olivier Kahn International Award for his contribution to the theory of molecular magnetism in 2011. He was a member of the commission of experts on quantum technologies for the 2021–27 Italian National Research Program (PNR), and he is involved in several national and European projects involving quantum technologies. He is currently one of the Principal Investigators of an European ERC Synergy Project.



Michele Amoretti (Senior Member, IEEE) received the Ph.D. degree in information technologies from the University of Parma, Parma, Italy, in 2006.

In 2013, he was a Visiting Researcher with LIG Lab, Grenoble, France. He is currently an Associate Professor of computer engineering with the University of Parma. He has authored or coauthored more than 130 research papers in refereed international journals, conference proceedings, and books. He is involved in the Quantum Information Science research and teaching initiative with the University of Parma, where he leads the Quantum Software Laboratory. His current research interests include quantum computing, high-performance computing, and the Internet of Things.

Dr. Amoretti is an Associate Editor for the journals IEEE TRANSACTIONS ON QUANTUM ENGINEERING and *International Journal of Distributed Sensor Networks*. He is currently one of the Principal Investigators of the European HE project Quantum Internet Alliance. He is the CINI Consortium delegate in the UNI/CT 535 "Quantum Technologies" UNINFO Commission, which is the national mirror of ISO/IEC JTC 1 WG 14 "Quantum Information Technologies" and of CEN/CENELEC JTC 22 "Quantum Technologies."

Open Access funding provided by 'Università degli Studi di Parma' within the CRUI CARE Agreement