Revisiting the Mapping of Quantum Circuits
Entering the Multi-core Era

Escofet, Pau; Ovide, Anabel; Bandic, Medina; Prielinger, Luise; Van Someren, Hans; Feld, Sebastian; Alarcon, Eduard; Abadal, Sergi; Almudever, Carmen

DOI
10.1145/3655029

Publication date
2025

Document Version
Final published version

Published in
ACM Transactions on Quantum Computing

Revisiting the Mapping of Quantum Circuits: Entering the Multi-core Era

PAU ESCOFET, Universitat Politècnica de Catalunya, Barcelona, Spain
ANABEL OVIDE, Universitat Politècnica de València, Valencia, Spain
MEDINA BANDIC, Delft University of Technology, Delft, Netherlands
LUISE PRIELINGER, Delft University of Technology, Delft, Netherlands
HANS VAN SOMEREN, Delft University of Technology, Delft, Netherlands
SEBASTIAN FELD, Delft University of Technology, Delft, Netherlands
EDUARD ALARCON, Universitat Politècnica de Catalunya, Barcelona, Spain
SERGI ABADAL, Universitat Politècnica de Catalunya, Barcelona, Spain
CARMEN ALMUDEVER, Universitat Politècnica de València, Valencia, Spain

Quantum computing represents a paradigm shift in computation, offering the potential to solve complex problems intractable for classical computers. Although current quantum processors already consist of a few hundred qubits, their scalability remains a significant challenge. Modular quantum computing architectures have emerged as a promising approach to scale up quantum computing systems. This article delves into the critical aspects of distributed multi-core quantum computing, focusing on quantum circuit mapping, a fundamental task to successfully execute quantum algorithms across cores while minimizing inter-core communications. We derive the theoretical bounds on the number of non-local communications needed for random quantum circuits and introduce the Hungarian Qubit Assignment (HQA) algorithm, a multi-core mapping algorithm designed to optimize qubit assignments to cores with the aim of reducing inter-core communications. Our exhaustive evaluation of HQA against state-of-the-art circuit mapping algorithms for modular architectures reveals a 4.9× and 1.6× improvement in terms of execution time and non-local communications, respectively, compared to the best-performing algorithm. HQA emerges as a very promising scalable approach for mapping quantum circuits into multi-core architectures, positioning it as a valuable tool for harnessing the potential of quantum computing at scale.

CCS Concepts: • **Hardware** → **Quantum computation**; • **Theory of computation** → *Design and analysis of algorithms*; • **Networks** → *Network resources allocation;*

P. Escofet and A. Ovide contributed equally to this research.
The authors gratefully acknowledge funding from the European Commission through HORIZON-EIC-2022-PATHFINDEROPEN-01-101099697 (QUADRATURE) and grant HORIZON-ERC-2021-101042080 (WINC). This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación and European ERDF under grant PID2021-123627OB-C5.

Authors' addresses: P. Escofet, E. Alarcon, and S. Abadal, Universitat Politècnica de Catalunya, Barcelona, 08034, Spain; e-mails: pau.escofet@upc.edu, eduard.alarcon@upc.edu, abadal@ac.upc.edu; A. Ovide and C. Almudever, Universitat Politècnica de València, Valencia, 46022, Spain; e-mails: aovigon@upv.es, cargara2@disca.upv.es; M. Bandic, L. Prielinger, H. van Someren, and S. Feld, Delft University of Technology (QuTech), Delft, 2628CJ, Netherlands; e-mails: m.bandic@tudelft.nl, l.p.prielinger@tudelft.nl, j.vansomeren-1@tudelft.nl, s.feld@tudelft.nl.

ACM 2643-6817/2025/01-ART4
https://doi.org/10.1145/3655029

Additional Key Words and Phrases: Quantum computing, multi-core quantum computing architecture, quantum circuit mapping

ACM Reference Format:
Pau Escofet, Anabel Ovide, Medina Bandic, Luise Prielinger, Hans van Someren, Sebastian Feld, Eduard Alarcon, Sergi Abadal, and Carmen Almudever. 2025. Revisiting the Mapping of Quantum Circuits: Entering the Multi-core Era. *ACM Trans. Quantum Comput.* 6, 1, Article 4 (January 2025), 26 pages. https://doi.org/10.1145/3655029

1 INTRODUCTION

Quantum computing has emerged as a new computational paradigm, harnessing the unique properties of quantum mechanics, including superposition and entanglement [39], to revolutionize problem-solving. These quantum properties enable quantum computers to perform certain calculations at an unprecedented speed, addressing problems previously considered intractable for classical computers. The potential applications of quantum computing span a wide range of domains, from cryptography through algorithms like Shor's prime factorization [47] to optimized database searches using Grover's algorithm [24] and even the simulation of complex physical systems [36].

Despite the promise of quantum computing, the current landscape is characterized by a significant gap between the potential of this technology and its practical realization. Quantum computers rely on various qubit implementation technologies, including superconducting qubits [5, 37], photonic qubits [30, 51], quantum dots [26, 49], and trapped ions [9, 43]. However, irrespective of the qubit technology employed, today's quantum computers are limited to a few hundred qubits [7], far from the million-qubit scale required for tackling real-world problems [44].

Monolithic single-chip architectures face inherent limitations in scalability due to challenges related to the integration of control circuits and wiring for accessing qubits while maintaining low error rates [38]. Moreover, increasing the number of qubits within a single processor results in a higher rate of undesirable qubit interactions, leading to issues like crosstalk [14]. As a result, scaling up monolithic quantum computers to accommodate a higher number of qubits remains a major challenge, necessitating innovative approaches to overcome this bottleneck.

One promising alternative to the single-core quantum computing architecture is the concept of modular or multi-core quantum processors [6, 28, 34, 46, 50]. This approach involves interconnecting multiple moderate-size chips or **quantum cores (QCores)** through classical and quantum-coherent links [22]. By adopting a modular architecture, it becomes feasible to tackle the challenge of scalability while maintaining the benefits of quantum coherence.

However, transitioning from monolithic to multi-core quantum devices introduces multiple new difficulties, with communication between cores standing out as a critical issue. Inter-core communications in multi-core quantum processors are significantly more costly than intra-core communications, leading to complex tradeoffs and optimization challenges. This article addresses the intricate problem of mapping quantum circuits onto multi-core quantum processors, explicitly focusing on minimizing the number of non-local communications, a critical factor in maximizing performance. Few multi-core mapping algorithms have been proposed [3, 4], and their optimality has not been studied in depth. This work proposes a novel mapping technique, comparing it to existing approaches. More precisely, the contributions of this article can be summarized as:

- We perform a non-local communications characterization for Quantum Random Circuits, in which theoretical upper and lower bounds are obtained. Note that this analysis allows, for the first time, for optimality assessment of different multi-core quantum circuit mapping algorithms.

- We propose the **Hungarian Qubit Assignment (HQA)** algorithm, originally presented in [18], and conduct a design exploration improving the algorithm's performance 1.33× on structured circuits.
- We compare different state-of-the-art multi-core mapping algorithms, assessing the number of non-local communications and the execution time. We show that HQA outperforms the execution time and non-local communications of the state-of-the-art best-performing algorithm by 4.9× and 1.6×, respectively.

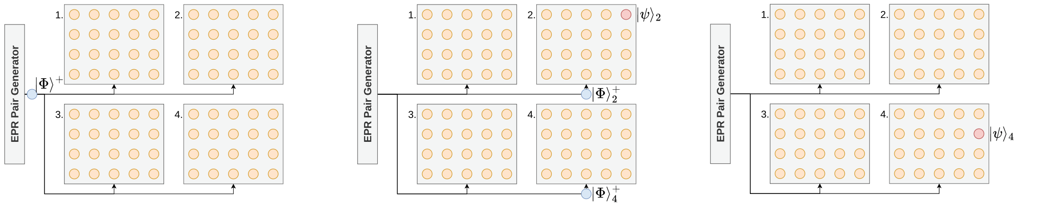
The remainder of this article is structured as follows. Section 2 provides a brief introduction to modular quantum computing architectures and discusses the challenges of these scalable systems that include the need for developing novel quantum circuit mapping techniques. In Section 3, we delve into the task of distributing quantum states from random quantum algorithms into quantum cores, establishing theoretical upper and lower bounds on the number of non-local communications required. Section 4 conducts a comprehensive review and analysis of state-of-the-art mapping algorithms designed for multi-core quantum computing architectures, comparing their performance to the previously established theoretical bounds. Section 5 introduces a novel multi-core mapping algorithm, the HQA, and evaluates its performance against the theoretical bounds. In Section 6, a series of experiments comparing various multi-core mapping algorithms are presented, assessing their scalability in terms of execution time and non-local communications. Finally, in Section 7, we discuss the results obtained and outline potential paths for future research in this critical domain.

2 ON MODULAR QUANTUM COMPUTING ARCHITECTURES

Current quantum computers have successfully integrated up to 1,000 qubits within a single processor, marking a significant milestone in the field [7, 21]. However, to address real-world problems effectively, quantum computers must scale to operate with thousands or even millions of qubits [44]. This ambitious objective highlights the need for quantum computers to be scaled, containing more qubits.

Nevertheless, scaling quantum computers to accommodate such a large number of qubits is a challenging task. A major difficulty is the intricate integration of classical control circuits and precise wiring for individual qubit addressability, all while maintaining low error rates [38]. Additionally, the quest to increase qubit counts must be accomplished without increasing crosstalk or interference among qubits [14]. Consequently, expanding monolithic quantum computers to integrate an increasing number of qubits remains a critical challenge that requires the exploration of innovative solutions.

One promising architectural alternative involves the division of the **Quantum Processing Unit (QPU)** into smaller, more manageable cells known as QCores [6, 28, 34, 50]. Different levels of modularity are envisioned at increasing system complexity, all requiring the introduction of means for communication. More precisely, for superconducting quantum processors, it was first proposed [6] to interconnect different chips through classical communication links, with the aim of executing large quantum algorithms (i.e., circuits that require more qubits than there are in a single core) using circuit cutting and knitting techniques [42, 52]. Next, in the near term, very short quantum links between adjacent chips (i.e., chip-to-chip quantum connector) will be introduced, allowing for quantum communication through two-qubit gates across processors [22]. As this technology evolves, later developments of multi-core quantum computing architectures envision the interconnection of QCores through longer quantum-coherent communication links as well as classical channels, enabling the coupling and entanglement of qubits situated in different cores [8, 22] and ultimately across different quantum computers (i.e., between different fridges). Alternatively, some

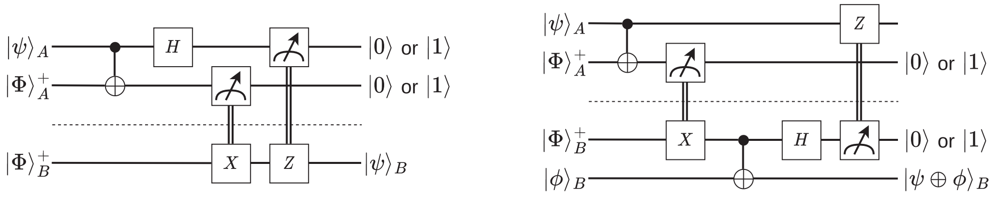


(a) Structure of the architecture, with an entangled state $|\Phi\rangle^+$ at the EPR pair generator.

(b) The entangled state is distributed to cores two and four.

(c) Movement of state $|\psi\rangle$ to core four by using the entangled state.

Fig. 1. Multi-core Quantum Computing Architecture based on the distribution of EPR Pairs. Several cores (rectangles) containing qubits (orange circles) are connected to an EPR Pair Generator, in charge of generating and distributing entangled pairs to different cores. Once the entangled states are in the core, they will be used (measured and thus consumed) for communication purposes, enabling the transfer of quantum states across cores.



(a) Teleportation Circuit.

(b) Remote Gate Circuit.

Fig. 2. EPR-based communication protocols.

designs focus on generating and distributing entangled qubit pairs to different cores, making use of quantum and classical communication channels [46].

In this work, we focus on multi-core quantum computing architectures consisting of several units that can communicate based on the generation and distribution of EPR pairs [17], as proposed by Rodrigo et al. [46] and illustrated in Figure 1. Within this architectural framework, diverse quantum cores are linked to an EPR pair generator through a quantum network, facilitating the distribution of entangled pairs. These entangled states ($|\Phi\rangle^+$) are used to transmit quantum states from one core to another, employing the principles of quantum teleportation [23], a quantum communication protocol depicted in Figure 2(a).

Beyond quantum teleportation, entangled pairs also serve as the primary resource for additional communication protocols, including the execution of remote quantum operations (illustrated in Figure 2(b)). These protocols are highly used over long-range quantum networks in **Distributed Quantum Computing (DQC)** [20]. Note that although remote two-qubit gates can also be used as communications primitives in short-range multi-core architectures, in this work, we focus on quantum circuit mapping algorithms that consider quantum teleportation as a communication means.

2.1 Challenges for Multi-core Quantum Computing Architectures

Multi-core quantum computing architectures represent a promising paradigm for overcoming the limitations associated with scaling monolithic quantum processors to accommodate larger numbers of qubits. However, this transition has its own difficulties. This section delves into the key problems that must be addressed when designing and implementing multi-core quantum computing systems. These challenges encompass the entire spectrum of quantum computing, from hardware considerations to software and performance evaluation.

2.1.1 *Rethinking the Full-stack.*

Full-stack quantum computing systems have been developed to bridge quantum algorithms with current monolithic quantum processors. However, going to modular architectures will require redesigning such a stack to extend it beyond computation, encompassing also communication. This double full-stack architecture [46] necessitates the integration of not only quantum computation elements but also support for classical and quantum communication such as the synchronization and scheduling of quantum/classical information exchange between cores.

2.1.2 *Balancing Computation and Communication Qubits.*

In an EPR-based multi-core architecture, qubits must be utilized for both communication and computation, introducing a delicate tradeoff. Achieving the right balance between qubits dedicated to computation and those reserved for communication is vital. Overallocation of qubits for communication may limit computational capabilities, while underallocation can restrict the efficient quantum state distribution and manipulation across cores.

2.1.3 *Communication Networks.*

The establishment of robust quantum/classical communication networks is central to the success of multi-core quantum computing architectures [19]. This challenge encompasses the development of technologies for the implementation of quantum-coherent links capable of transmitting quantum states with minimal decoherence [32]. Additionally, the creation of efficient quantum communication primitives and protocols is essential for orchestrating the seamless exchange of quantum information across cores.

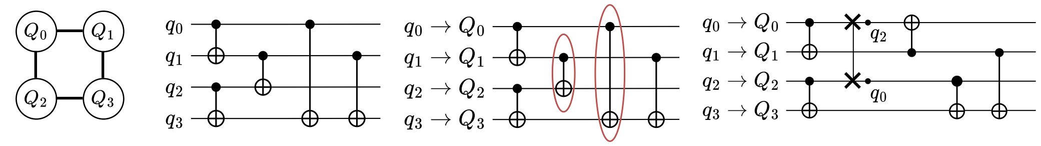
2.1.4 *Benchmarking and Performance Metrics.*

How to properly measure the performance of a quantum computer is still an open question. In the last years, there has been an effort to define a set of benchmarks and performance metrics for monolithic quantum processors [11, 53]. However, these may not capture the intricacies of modular quantum computing architectures as they are missing the communication part as well as the parallelization ability. More precisely, these metrics should include, for instance, factors such as inter-core quantum state transfer latency, communication overhead, and resource utilization efficiency. Accurate evaluation methods are critical for guiding the design and optimization of multi-core quantum computing systems.

2.1.5 *Quantum Compilers for Multi-core Quantum Computers.*

Multi-core quantum computing architectures introduce a complex compilation landscape. Quantum compilers play a pivotal role not only in translating high-level quantum programs into executable instructions but also in performing some modifications to the quantum circuit to deal with the computing hardware constraints, a process known as mapping, where the circuit is transformed to an equivalent one that complies with the restrictions of the targeted quantum processor. Compilers for multi-core architectures must consider the intricacies of inter-core communication, virtual qubit movement, and synchronization. Adapting existing compilation techniques to cater to the distributed nature of multi-core architectures is a non-trivial challenge. Developing quantum compilers capable of optimizing quantum circuits across multiple cores while minimizing non-local communications is an active area of research.

In conclusion, transitioning to multi-core quantum computing architectures is required for scaling up quantum computers, but it comes with a set of formidable challenges involving hardware, communication infrastructure, benchmarking, and software development. Addressing them is essential to unlock the full capabilities of these modular quantum systems and pave the way for the next generation of quantum computing technologies.



(a) Coupling map of a quantum processor (b) Quantum circuit with 4 (virtual) qubits and 5 two-qubit gates (c) Virtual qubits mapped to physical qubits. Unfeasible gates are highlighted (d) Final mapped circuit. SWAP gates (×) are added to satisfy the connectivity constraints

Fig. 3. Overview of the process of mapping a quantum circuit into the topology of a particular quantum computer.

2.2 Mapping of Quantum Circuits

Linked to the last challenge posed in the previous section, mapping is a critical step in the compilation process for quantum circuits. Prior to their execution, quantum circuits are modified (gate decomposition, circuit optimization, and addition of gates such as SWAPs to route the qubits, among other phases) to be adapted to the hardware's restrictions.

In monolithic quantum computers, the mapper assigns each quantum state (or virtual qubits) from the circuit to an initial physical qubit within the architecture, illustrated in Figure 3(c). Additionally, it inserts the necessary operations (mostly SWAP gates) to facilitate the movement of quantum states, ensuring that the hardware's specific coupling constraints are met so that each two-qubit gate can be performed, as depicted in Figure 3(d). An example of such coupling topology is depicted in Figure 3(a). The scarce connectivity between qubits is the major limitation of current quantum processors. This intricate mapping process is essential for the circuit to function seamlessly on the target quantum processor. Some examples of quantum mappers for monolithic quantum computers are [1, 33, 35, 45, 48].

When going from single-core to multi-core architectures, the mapping problem becomes more challenging and highly depends on how cores are connected, allowing for some communication primitives. This work assumes an EPR-distributed architectural model and focuses on distributing quantum states into cores. Along the execution of the circuit, quantum states will be moved from one core to another, ensuring that every time a two-qubit gate needs to be executed, the involved qubits will be located in the same core.

An example is depicted in Figure 4, where a circuit with three timeslices is mapped into a two-core architecture, with two qubits per core. A timeslice is defined as the set of quantum gates from the circuit that can be executed in parallel. In each one of the timeslices, the interacting qubits are located in the same core, ensuring all two-qubit gates will be feasible. The movements across cores will be performed using quantum teleportation (Figure 2(a)) between timeslices. We refer to these movements across cores as non-local communications, and how these non-local communications are performed depends on how cores are connected among them [19].

In this work's architectural model, quantum teleportation is used as the inter-core communication protocol, consisting of between four and six quantum gates (depending on the needed corrections). The movement of quantum states across cores will be performed after the mapping, adding the needed quantum gates to perform quantum teleportation. Moreover, a non-local communication requires generating and distributing an entangled pair. In the case where no ancillary qubits are present in the architecture (i.e., the number of virtual qubits in the circuit is the same as the physical qubits in the architecture), at least two qubits devoted to communication per core are required. One will be used as a buffer holding an arriving quantum state, while the other performs the teleportation of a quantum state, freeing space for the other.

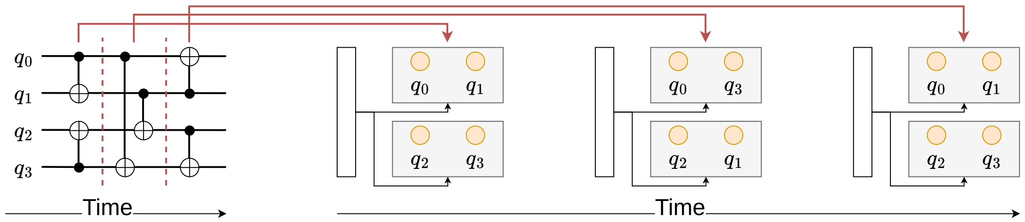


Fig. 4. Mapping a four-qubit quantum circuit (left) into a two-core Quantum Computing Architecture. For each timeslice (sections of the circuit), we assign qubits into cores, so each two-qubit interaction involves qubits located in the same core.

Since non-local communications are much more expensive than intra-core operations, in this work, we focus on the problem of moving qubits across cores, ensuring that, for each two-qubit gate, the involved qubits will be placed in the same core. Such a problem is depicted in Figure 4 and has been previously studied in [3] and [4]. Though our architectural model would support Remote Gate execution, for this work, we will only consider qubit movement across cores, making our work analogous to the one in [3] and [4].

3 NON-LOCAL COMMUNICATIONS CHARACTERIZATION

Current multi-core mapping algorithms fail to estimate how close the obtained mapping is to the optimal solution. In this section, we focus on Quantum Random Circuits, random algorithms characterized by the number of qubits q, the number of gates g, and the fraction of two-qubit gates of the circuit f. From now on, we will refer to such circuits as (q, g, f)-Quantum Random Circuit.

Such circuits are constructed by starting with an empty quantum circuit with q qubits. We then proceed to add g gates to the quantum circuit; each gate has probability f of being a two-qubit gate, and probability $(1 - f)$ of being a single-qubit gate. In both cases, the qubits involved in the operation are selected randomly.

3.1 Non-local Communication Bounds for Random Quantum Circuits

To assess the performance of multi-core mapping algorithms when mapping Quantum Random Circuits, we characterize the number of non-local communications when employing a naive strategy (Theorem 3.2), and the lower bound on the number of non-local communications when employing an optimal strategy that does not take into account future qubit interactions (Theorem 3.3), when mapping a (q, g, f)-Quantum Random Circuit into a (q, N)-quantum computing architecture (modular architecture with N cores, each of them containing $\frac{q}{N}$ qubits).

Lemma 3.1. *For a given (q, N)-quantum computing architecture and a (q, g, f)-Quantum Random Circuit, the expected number of qubits involved in unfeasible operations (q_{unf}) per timeslice is*

$$\mathbb{E}(q_{unf})_t = \frac{2(N-1)gfq}{N(q-1)t}, \tag{1}$$

where t is the number of timeslices in the circuit, and an unfeasible operation is a two-qubit gate involving qubits that are currently located in different cores.

Proof. We begin by assessing the probability that, for a specific two-qubit gate $cx(q_i, q_j)$, the qubits involved, q_i and q_j, are located within the same quantum core.

Let Q be the set of qubits in the whole architecture and $C(q_i)$ be the set of qubits in the same core as q_i. We define the probability of q_i and q_j of being in the same core as the number of qubits different than q_i, that are in the same core as q_i (i.e., $\frac{q}{N}-1$), over the total number of qubits different than q_i (i.e., $q-1$):

$$\frac{|\{q_k \in C(q_i) : q_k \neq q_i\}|}{|\{q_k \in Q : q_k \neq q_i\}|} = \frac{\frac{q}{N}-1}{q-1}. \tag{2}$$

As the circuit contains a total of g gates and the fraction of two-qubit gates is f, the expected number of two-qubit gates is given by $g \cdot f$. Let t be the number of timeslices the circuit can be sliced into. As two-qubit gates are randomly distributed across the whole circuit, the expected number of two-qubit gates in each timeslice is

$$\mathbb{E}(g_{2q})_t = \frac{g \cdot f}{t}. \tag{3}$$

By the definition of timeslice, each qubit interacts at most one time in each timeslice. Therefore, the expected number of qubits involved in a two-qubit operation per timeslice is

$$\mathbb{E}(q_{inv})_t = 2\frac{g \cdot f}{t}. \tag{4}$$

For a given timeslice, the probability of q_i being involved in a two-qubit operation is

$$P(q_i \in q_{inv})_t = \frac{2\frac{g \cdot f}{t}}{q}. \tag{5}$$

Without loss of generality, let us see the possible scenarios of q_1 when going from timeslice t_{i-1} to timeslice t_i:

- q_1 interacts in t_i with a qubit q_k in the same core it's currently in. In that scenario, no non-local communications are needed, as both interacting qubits are already located in the same core. Here $C_{t_i}(q_1)$ represents the core where q_1 is located in timeslice t_i:

$$C_{t_{i-1}}(q_1) = C_{t_{i-1}}(q_k) = C_{t_i}(q_1) = C_{t_i}(q_k). \tag{6}$$

  The probability for the first scenario is given by the probability of q_1 interacting in timeslice t_i, expressed in Equation (5), times the number of qubits in $C_{t_{i-1}}(q_1)$ different than q_1, over the number of all qubits different than q_1:

$$P(q_1 \in q_{int} \,\&\, q_1 \notin q_{unf})_t = \frac{2\frac{g \cdot f}{t}}{q} \cdot \frac{\frac{q}{N}-1}{q-1}. \tag{7}$$

- q_1 interacts in t_i with a qubit q_k in a different core than it currently is:

$$C_{t_{i-1}}(q_1) \neq C_{t_{i-1}}(q_k) \qquad\qquad C_{t_i}(q_1) = C_{t_i}(q_k). \tag{8}$$

  The probability for this second scenario is given by the probability of q_1 interacting in timeslice t_i, expressed in Equation (5), times the number of qubits not in $C_{t_{i-1}}(q_1)$, over the number of all qubits different than q_1:

$$P(q_1 \in q_{int} \,\&\, q_1 \in q_{unf})_t = \frac{2\frac{g \cdot f}{t}}{q} \cdot \frac{(N-1)\frac{q}{N}}{q-1}. \tag{9}$$

- q_1 does not interact in t_i, and therefore no non-local communications are needed. It could happen that q_1 is moved to another core to make room for an arriving qubit that needs to interact in the core, but this non-local communication has already been taken into account for the interacting qubit.

Therefore, at timeslice t_i, the probability of q_1 being involved in an unfeasible two-qubit gate is given by the second scenario, described in Equation (9).

Generalizing to all qubits, we obtain the expected number of qubits involved in unfeasible operations (q_{unf}) per timeslice:

$$\mathbb{E}(q_{unf})_t = q \cdot \frac{2\frac{g \cdot f}{t}}{q} \cdot \frac{(N-1)\frac{q}{N}}{q-1} = \frac{2(N-1)gfq}{N(q-1)t}. \tag{10}$$

□

Theorem 3.2. *For a given (q, N)-quantum computing architecture and a (q, g, f)-Quantum Random Circuit, the number of non-local communications when employing a naive assignation is upper bounded by*

$$\texttt{non-local comms} \leq \frac{2(N-1)gfq}{N(q-1)}. \tag{11}$$

Proof. From Lemma 3.1, we know at each timeslice $\frac{2(N-1)gfq}{N(q-1)t}$ qubits are expected to be involved in unfeasible two-qubit gates. Since each qubit is involved in at most one two-qubit gate, and each two-qubit gate involves two qubits, we have a total of $\frac{(N-1)gfq}{N(q-1)t}$ unfeasible two-qubit gates at each timeslice.

For each unfeasible gate $cx(q_a, q_b)$, the naive approach will use two non-local communications, one to send q_a to q_b's current core, and one to make space for q_a in the destination core, as cores have a fixed size, and all the physical qubits in the architecture hold a quantum state from the circuit.

Therefore, from timeslice t_i to timeslice t_{i+1}, the expected number of non-local communications caused by the $\frac{(N-1)gfq}{N(q-1)t}$ unfeasible two-qubit gates is upper-bounded by two non-local communications for each unfeasible two-qubit gate:

$$\texttt{non-local comms}_t \leq 2 \cdot \frac{(N-1)gfq}{N(q-1)t} = \frac{2(N-1)gfq}{N(q-1)t}. \tag{12}$$

When generalizing for all t timeslices, the number of non-local communications for a (q, N)-quantum computing architecture and a (q, g, f)-Quantum Random Circuit when using a Naive Mapping algorithm is upper bounded by

$$\texttt{non-local comms} \leq t \cdot \frac{2(N-1)gfq}{N(q-1)t} = \frac{2(N-1)gfq}{N(q-1)}. \tag{13}$$

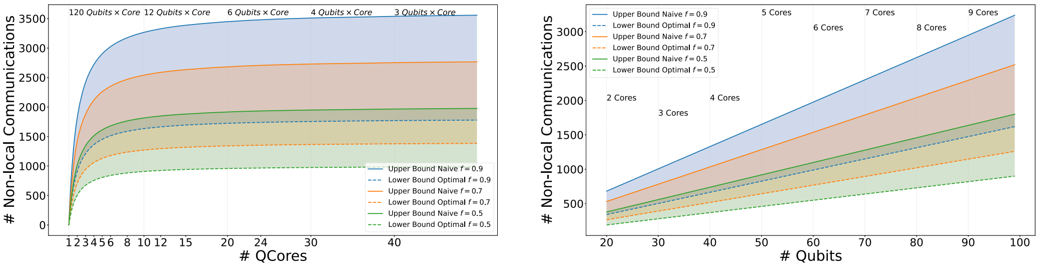
□

It may happen that when making space for q_a to arrive, the moved qubit (which is chosen randomly) was also a qubit involved in an unfeasible two-qubit gate, and it is moved to the core where the other interacting qubit was. When this happens, two unfeasible two-qubit gates are corrected using just two non-local communications (instead of four, assumed in the previous *proof*), obtaining an average of just one non-local communication per unfeasible two-qubit gate.

This is why the proposed bound in Theorem 3.2 is indeed an upper bound and a Naive approach could obtain a lower number of non-local communications. The same reasoning is applied to pose the following theorem.

Theorem 3.3. *For a given (q, N)-quantum computing architecture, and a (q, g, f)-Quantum Random Circuit, the optimal number of non-local communications without using future qubit interactions is lower bounded by*

$$\frac{(N-1)gfq}{N(q-1)} \leq \texttt{non-local comms}. \tag{14}$$



(a) $(120, 2000, f)$−Quantum Random Circuit communication bounds when increasing the number of cores in a fixed-size architecture with 120 qubits.

(b) $(q, 20q, f)$−Quantum Random Circuit communication bounds increasing the number of qubits and cores, with a fixed core size of 10 qubits per core.

Fig. 5. Non-local communications bounds for (q, g, f)−Quantum Random Circuits.

Proof. Similar to Theorem 3.2, the lower bound on the number of non-local communications is obtained from the number of unfeasible gates, proposed in Lemma 3.1. However, in this case, no extra communication to make space in the destination core will be needed, assuming the destination core contains a qubit that is also involved in an unfeasible two-qubit gate.

Therefore, from timeslice t_i to timeslice t_{i+1}, the minimum number of non-local communications caused by the $\frac{(N-1)gfq}{N(q-1)t}$ unfeasible two-qubit gates is just one non-local communication for each unfeasible two-qubit gate:

$$\texttt{non-local comms}_t \geq 1 \cdot \frac{(N-1)gfq}{N(q-1)t} = \frac{(N-1)gfq}{N(q-1)t}. \tag{15}$$

When generalizing for all t timeslices, the number of non-local communications for a (q, N)-quantum computing architecture and a (q, g, f)-Quantum Random Circuit without using future qubit interactions is lower bounded by

$$\texttt{non-local comms} \geq t \cdot \frac{(N-1)gfq}{N(q-1)t} = \frac{(N-1)gfq}{N(q-1)}. \tag{16}$$

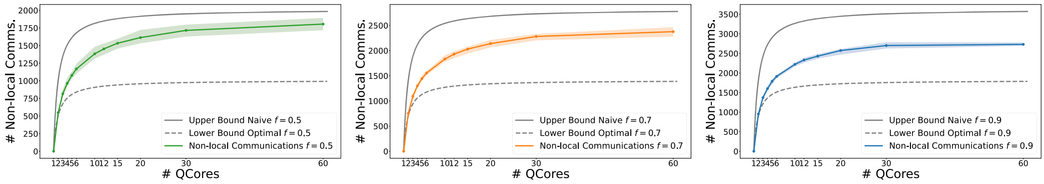
□

This last scenario is encountered when all cores have an even number of qubits involved in an unfeasible two-qubit gate. Within this scenario, only qubits involved in those unfeasible gates will be moved, needing just one teleportation per operation. However, it depends on the mapping algorithm to identify such optimal movements.

The bounds proposed in Theorems 3.2 and 3.3 are plotted in Figure 5, for different circuit sizes, and three different two-qubit gate fractions. Figure 5(a) depicts how the communication bounds vary for a fixed-size architecture with 120 qubits when partitioning it into an increasing number of cores. On the other hand, Figure 5(b) depicts the non-local communications needed when adding 10-qubit cores to the architecture, using a circuit with size depending on the number of qubits of the architecture ($g = 20q$).

It can be seen how the non-local communications when increasing the number of cores (Figure 5(a)) rapidly increase from going to a monolithic quantum computer (one core) to a multi-core with a few cores but then stabilizes, showing that, at some point, increasing the number of cores, and thus decreasing the number of qubits per core, has a minimum impact on the number of non-local communications. Figure 5(b) shows that both communication bounds have a linear growth when increasing the architecture's number of cores and the circuit's size.

These bounds provide the first two reliable metrics on the optimal number of non-local communications for Quantum Random Circuits. However, the lower bound proposed in Theorem 3.3 can



(a) $(120, 2000, 0.5)-$Random Circuit Naive Mapping.

(b) $(120, 2000, 0.7)-$Random Circuit Naive Mapping.

(c) $(120, 2000, 0.9)-$Random Circuit Naive Mapping.

Fig. 6. Non-local communications when using the Naive Mapping Algorithm for different (120, 2000, f)−Quantum Random Circuits. Communication bounds are shown in grey.

ALGORITHM 1: Naive Mapping Algorithm

```
Data: Circuit Timeslice's $Ts$
Result: Valid Qubit to Core assignments $As$
$As[0] \leftarrow$ Random Initial Assignment;
for $T \in Ts$ do
    $A_T \leftarrow$ Current Assignment;
    for $(q_A, q_B) \in T$ do
        if $A_T[q_A] \neq A_T[q_B]$ then
            $q_{aux} \leftarrow$ Random Qubit from $q_A$'s core;
            $A_T[q_{aux}] = A_T[q_B]$;
            $A_T[q_B] = A_T[q_A]$;
    $A_{T+1} \leftarrow A_T$;
```

be further improved by considering future qubit interactions. Therefore, mapping algorithms that take into account qubit interactions to optimize future movements have the potential to achieve a lower number of non-local communications than the one proposed.

3.2 Naive Mapping Approach

In order to validate the proposed bounds (Theorem 3.2 and 3.3), we propose a Naive Mapping algorithm, by which, whenever an unfeasible two-qubit gate is encountered, the involved qubits are moved together into one of both involved cores, by randomly making space in the destination core. Therefore, a qubit that was not involved in an unfeasible two-qubit gate will be moved to make space for the arriving qubit. Such an algorithm is described in Algorithm 1.

When mapping a $(q, g, f)-$Quantum Random Circuit into a modular architecture using the Naive Mapping algorithm proposed above, we expect the number of communications to be lower than the upper bound proposed in Theorem 3.2, as some movements to make space for the incoming qubits will place previously unfeasible qubits into the right core.

A comparison between the proposed bounds and the Naive Mapping approach proposed in Algorithm 1 is depicted in Figure 6, where three different Quantum Random Algorithms are mapped to different modular architectures, and the number of non-local communications obtained are compared to the communication bounds proposed in Section 3.1.

In each experiment, the needed communications are obtained after averaging the communications obtained on 20 different random circuits with the same configuration. The maximum and minimum values obtained in each scenario are shown in the shadowed area.

Figure 6 shows that the number of non-local communications needed is closer to the upper bound when the two-qubit gate frequency is lower. This is due to the movement of random qubits to make space for the arriving ones. If the selected qubit was involved in an unfeasible two-qubit gate in that same timeslice, we would have moved it anyway, and therefore, the movement to make space has no impact on the number of non-local communications. As the two-qubit gate fraction increases, it is more likely that the selected random qubit is involved in an unfeasible two-qubit gate, achieving a number of non-local communications further from the upper bound (Figure 6(c)) than when the two-qubit gate fraction is small (Figure 6(a)).

In future sections, we will use this Naive method as a baseline for the non-local communications needed to map quantum algorithms into multi-core architectures.

4 MULTI-CORE MAPPING ALGORITHMS ANALYSIS

This section reviews the inter-core mapping algorithms proposed in [3], called **Fine-grained Partitioning-relaxed Overall Extreme Exchange (FGP-rOEE)**, and the algorithm proposed in [4], based on the **Quadratic unconstrained binary optimization (QUBO)** problem. These two algorithms are some of the scarce algorithms proposed so far to solve the mapping problem for multi-core quantum computers.

Other quantum circuit mapping algorithms for distributed quantum computing have been proposed [2, 20, 54]. They make use of EPR pairs for only remote operations across cores (i.e., telegate) [13] or also for qubit teleportation (i.e., teledata). In addition, other mapping algorithms such as [55] are developed to target chiplet architectures [50], a different type of modular architectures than the ones considered in this work. In this work, we only focus on qubit distribution in modular multi-core architectures, in which processors are connected with classical and quantum links, not taking into account remote operations or a different type of processor architecture.

4.1 FGP-rOEE

Baker et al. [3] proposed the FGP-rOEE algorithm to tackle the mapping problem in multi-core quantum computers. The algorithm takes as inputs a quantum circuit with q qubits and an architecture with N cores, each core accommodating $\frac{q}{N}$ qubits. The algorithm operates under two fundamental assumptions: (1) the coupling map of each core is all-to-all—i.e., all qubits in a core have direct connections to one another, allowing the direct execution of a two-qubit gate if both qubits are located within the same core, and (2) cores are interconnected all-to-all, enabling the exchange of quantum states between any pair of cores.

The primary objective of the algorithm is to obtain a sequence of qubit-to-core assignments, with one assignment per timeslice. The quantum circuit is separated into these timeslices, and a valid assignment of qubits to cores is found for each one. A valid assignment must have every pair of interacting qubits in the timeslice assigned to the same core, ensuring that no two-qubit gate involves qubits located in different cores.

For a given timeslice t, FGP-rOEE computes the interaction graph of that timeslice, a graph with the qubits as nodes and an edge between two nodes if the qubits interact with each other in future timeslices. The edges are weighted, representing the immediacy of the interaction. These weights are called look-ahead weights and are computed using Equation (17), where $I(m, q_i, q_j) = 1$ if qubits q_i and q_j interact at timeslice m, and the exponential decay function $(2{-}x)$ is used so nearby timeslices have more impact on the look-ahead weights than latter ones:

$$w_t(q_i, q_j) = \sum_{t<m\leq T} I(m, q_i, q_j) \cdot 2^{-(m-t)}. \tag{17}$$

For qubits that interact exactly at timeslice t, a weight of infinity is set to the edges. This weight implies that, at that particular timeslice, these qubits must unequivocally reside within the same core, thus making their separation impossible.

Next, a k-partitioning algorithm is employed to partition the interaction graph into k disjoint subsets of nodes. Here, k corresponds to the number of cores N, and it is imperative that all partitions have the exact same size, as cores have a fixed size, and we can only assign $\frac{q}{N}$ qubits into each core. Due to these strict constraints, the set of suitable k-partitioning algorithms becomes notably limited.

In [3], the use of the **Overall Extreme Exchange (OEE)** algorithm [41] is proposed. The OEE algorithm builds upon the Kernighan–Lin [29] algorithm and expands its capabilities. In their work, Baker et al. introduce a variant of the OEE algorithm known as rOEE. Similar to the OEE algorithm, the rOEE also starts with an assignment and performs exchanges of nodes until a valid partition is reached (i.e., all interacting qubits are in the same partition).

This procedure is repeated for every pair of timeslices, using the previous assignment of qubits to cores as input for the rOEE algorithm. With this, a path of valid assignments is found over the whole circuit. A detailed analysis of the FGP-rOEE algorithm can be found in [40].

4.2 QUBO

Bandic et al. [4] proposed a mapped solution problem for multi-core or modular architectures based on the QUBO [25] method, relying on prior subgraph isomorphism approaches [27], and single-core solutions [16]. This approach addresses qubit allocation and inter-core communication costs through binary decision variables, being suitable for different modular architectures. QUBO introduces a mathematical problem classified as NP-hard, which is subject to optimization. The formula employed represents the objective function to be minimized as follows:

$$\min_x x^T Q x = \min_x \sum_{i<j} Q_{ij} x_i x_j + \sum_i Q_{ii} x_i, \tag{18}$$

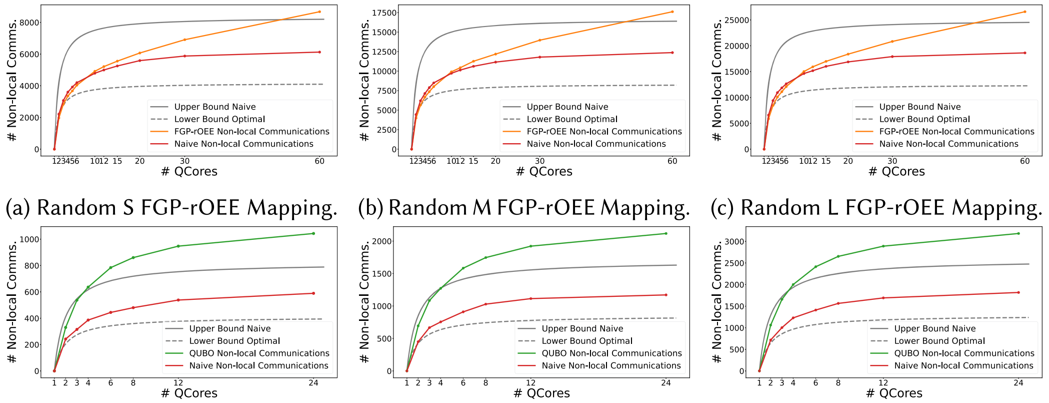
where the variable x represents a binary decision vector of dimension N, and Q designates a symmetric square matrix composed of $N \times N$ real-valued constants.

Similar to FGP-rOEE, QUBO partitions the quantum circuit into slices, each encompassing a series of gates. Each time slice can be depicted as a graph, with nodes representing distinct qubits and edges denoting interactions performed by two-qubit gates. The primary aim of the objective function is to determine an allocation for each time-slice graph, ensuring that all qubits engaged in a common gate are assigned to the same computational core without surpassing the core's capacity. Additionally, it seeks to reduce inter-core communications during these assignments. The objective function is then generalized for all time slices and modified to minimize potential inter-core communication between every pair of assignments. For more information on the mathematical process, refer to [4]. The objective function counts with a weighting factor denoted as λ, which can be employed to adjust the different components of the objective function.

4.3 Evaluation and Limitations

In this section, we compare the performance of the state-of-the-art mapping algorithms for multi-core quantum computing architectures [3, 4] described in previous sections to the non-local communication bounds proposed in Theorems 3.2 and 3.3, as well as to the naive approach proposed in Algorithm 1.

We use Qiskit's [45] Random Circuit library to match the available implementation of [4]. The performance of the FGP-rOEE algorithm [3] is assessed using three different random circuits of 120 qubits and different sizes: Random S (100 timeslices), Random M (200 timeslices), and Random



(a) Random S FGP-rOEE Mapping. (b) Random M FGP-rOEE Mapping. (c) Random L FGP-rOEE Mapping.

(d) Random S QUBO Mapping. (e) Random M QUBO Mapping. (f) Random L QUBO Mapping.

Fig. 7. Non-local communications when using the FGP-rOEE and QUBO algorithms for different Quantum Random Circuits.

L (300 timeslices). The performance of the QUBO Mapping algorithm [4] is assessed with smaller random circuits of 48 qubits and different random sizes, Random S (25 timeslices), Random M (50 timeslices), and Random L (75 timeslices). The size difference of the used circuits reflects the execution time each algorithm needs to compute the valid mapping.

Figure 7 shows the non-local communications when mapping the selected circuit into a modular architecture with a fixed number of qubits (120) distributed over an increasing number of cores. It can be seen how, though both the FGP-rOEE and QUBO mapping algorithms consider future interaction among qubits, taking into account more information than the Naive approach, their performance for Quantum Random Circuits is far from optimal, achieving in some cases a higher number of non-local communications than the Naive approach, and surpassing the Naive upper bound derived in Theorem 3.2.

Our intuition on the observed sub-optimal performance of the tested algorithms is that it can be attributed to their fundamental approach, which treats mapping as a graph partitioning problem. This approach, while suitable for certain scenarios, may not be ideal for the specific task of distributing quantum circuits into modular architectures. Quantum circuit mapping, as a problem, indeed shares similarities with graph partitioning, as it involves the allocation of computational resources. However, it introduces unique constraints, such as a fixed number of cores and a predefined number of qubits per core, and existing algorithms for graph partitioning often do not consider the specific constraints imposed by quantum computing architectures.

Furthermore, the primary objective when mapping a quantum circuit is not merely achieving a balanced allocation of qubits to cores but, more critically, minimizing the number of non-local communications needed in between time slices.

It is worth mentioning that both approaches can potentially improve the lower bound on the number of non-local communications proposed in Theorem 3.3, as they consider future qubit interaction, and, as stated in Section 3.1, mapping algorithms that take into account qubit interactions to optimize future movements have the potential to achieve a lower number of non-local communications than the one proposed.

In summary, while graph partitioning techniques offer a foundational framework for tackling the challenge of distributing quantum circuits into modular architectures, their suitability is limited by the unique constraints of mapping. The scarcity of specialized tools and the failure to

ALGORITHM 2: Hungarian Qubit Assignment

```
Data: Circuit Timeslice's $Ts$
Result: Valid Qubit to Core assignments $A$
$A_0 \leftarrow$ Initial Assignment;
$G_{unf} \leftarrow []$;
for $T \in Ts$ do
    $A_T \leftarrow$ Current Assignment;
    for $(q_A, q_B) \in T$ do
        if $A_T[q_A] \neq A_T[q_B]$ then
            $G_{unf}$.insert($(q_A, q_B)$);
    while $G_{unf}$.not_empty() do
        $C_{mat} \leftarrow$ Empty Cost Matrix;
        for $(q_A, q_B) \in G_{unf}$ do
            for $n \in Cores$ do
                if $n$ is full then
                    $C_{mat}[(q_A, q_B)][n] \leftarrow \infty$;
                else if $A_T[q_A] == n$ or $A_T[q_B] == n$ then
                    $C_{mat}[(q_A, q_B)][n] \leftarrow 1$;
                else
                    $C_{mat}[(q_A, q_B)][n] \leftarrow 2$;
        assign $\leftarrow$ Hungarian($C_{mat}$);
        $A_T \leftarrow A_T$.update(assign);
        $G_{unf} \leftarrow G_{unf}$.remove(assign);
    $A_{T+1} \leftarrow A_T$;
```

prioritize the reduction of non-local communications contribute to the underperformance of these approaches. Addressing these limitations is essential to unlocking the full potential of quantum computing systems.

5 HUNGARIAN QUBIT ASSIGNMENT

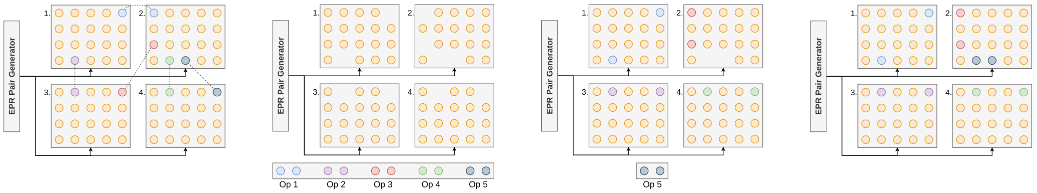
In this section, the HQA [18] is described. Similar to [3] and [4], the HQA algorithm describes how to assign qubits to cores in between timeslices and is generalized to map the whole algorithm. However, unlike the previously discussed multi-core mapping algorithms, HQA distributes two-qubit operations into cores, a much easier task than distributing qubits into cores by graph partitioning. As each two-qubit operation within a timeslice involves two distinct qubits, the algorithm's output will be a valid assignment for that particular timeslice.

5.1 General Overview

The proposed algorithm is summarized in Algorithm 2, and a simple example for a particular timeslice is depicted in Figure 8.

The algorithm starts with a valid assignment of qubits to cores for timeslice t. The following timeslice ($t + 1$) has a set of unfeasible two-qubit gates involving qubits that are currently located in different cores. This is depicted in Figure 8(a), where we can see five unfeasible two-qubit operations (color-coded) for an example architecture of four cores.

The qubits involved in the unfeasible two-qubit operations are then removed from the assignment and placed in an auxiliary vector of unassigned qubits, as depicted in Figure 8(b). Now, the



(a) Unfeasible assignment of qubits to cores.

(b) Remove the qubits involved in unfeasible operations.

(c) Each core gets assigned one unfeasible two-qubit gate.

(d) Repeat until all two-qubit gates are assigned to a core.

Fig. 8. Hungarian Qubit Assignment overview.

task is to assign each unfeasible operation to a core with enough space to take in the qubits. For this assignment, we will construct a cost matrix using the cost function in Equation (19). This cost function assigns a cost value C_t to each pair of unfeasible two-qubit operation op_i and core c_j based on how many non-local communications are needed to place both qubits involved in the operation q_A, q_B into the destination core c_j (i.e., one if a qubit involved in the unfeasible operation op_i is already placed in core c_j, two otherwise):

$$C_t(op_i, c_j) = \begin{cases} \infty & \text{if } c_j \text{ is full} \\ 1 & \text{if } q_A \in c_j \text{ or } q_B \in c_j \\ 2 & \text{otherwise.} \end{cases} \tag{19}$$

We use the Hungarian algorithm [31], a highly efficient linear assignment algorithm with polynomial time complexity ($O(n^3)$), to assign a single operation to each core using the cost matrix previously constructed. Its versatility, simple implementation, and robustness make the Hungarian algorithm a favored choice in various fields, especially when a quadratic algorithm is impractical.

The objective of the assignment problem is to find the best way to assign a set of tasks (cores) to a group of resources (unfeasible two-qubit operations) while minimizing the total cost. By doing this, at each iteration, only one operation will be assigned to each core, ensuring the core's capacity is not exceeded.

When an operation is assigned to a core, both qubits involved are placed in the free spaces of the core, decreasing by two the number of free spaces in the core every time an operation is assigned to it. Placing both qubits of each unfeasible operation into the same core ensures all two-qubit gates will be feasible in the following timeslice.

The Hungarian algorithm only assigns one operation per core. Therefore, some operations will remain unassigned in case of having more unfeasible operations than cores, as depicted in Figure 8(c). A new cost matrix will be computed for those unassigned operations, considering the new free spaces of each core and setting a weight of infinity for those cores already full, ensuring that no core exceeds its capacity and that the resulting assignment will be valid. This process is repeated until all unfeasible gates have been assigned to a core, resulting in a valid assignment for the timeslice, as shown in Figure 8(d).

It is important to note that when using the same number of virtual qubits (quantum states in the circuit) as physical qubits (qubits in the quantum computing architecture), each core must contain an even number of free spaces for this approach to work. Otherwise, when assigning operations into cores, there will be a pair of qubits left to assign and two cores with exactly one free space each, making it impossible to assign both qubits of the unfeasible operation to the same core. To ensure all cores contain an even number of free spaces and that all operations will be assigned, for each pair of cores with an odd number of space, an auxiliary two-qubit gate involving two non-interacting qubits from those cores is created, forcing all cores to have an even number of free spaces.

Regarding the initial assignment for the algorithm, a structured assignment has the potential to perform much better than a random assignment. This possibility is explored in future sections.

5.2 Considering Future Qubit Interactions

Future interactions of qubits can be added to the cost matrix to perform an assignment that further reduces the number of non-local communications. To this end, we quantify how much qubits interact in future timeslices using the same approach as in [3], described in Equation (17), and introduce the attraction force of a qubit q_i to a core c_j, which is computed as

$$\mathtt{attr}_t^q(q_i, c_j) = \sum_{i'=0}^{q} J_t(q_{i'}, c_j) \cdot w_t(q_i, q_{i'}), \tag{20}$$

where $J_t(q_{i'}, c_j) = 1$ if qubit $q_{i'}$ is in core c_j at timeslice t, and $w_t(q_i, q_{i'})$ is computed using Equation (17).

The new cost matrix is then computed using the costs given in Equation (22), where the number of non-local communications needed and the attraction forces are combined. Note that, as each operation op_i involves two qubits (q_A and q_B), the operation's attraction force to a core is the average attraction force of the involved qubits:

$$\mathtt{attr}_t^{op}(op_i, c_j) = \frac{\mathtt{attr}_t^q(q_A, c_j) + \mathtt{attr}_t^q(q_B, c_j)}{2} \tag{21}$$

$$C_t(op_i, c_j) = \begin{cases} \infty & \text{if } c_j \text{ is full} \\ 1 - \mathtt{attr}_t^{op}(op_i, c_j) & \text{if } q_A \in c_j \text{ or } q_B \in c_j \\ 2 - \mathtt{attr}_t^{op}(op_i, c_j) & \text{otherwise.} \end{cases} \tag{22}$$

The proposed approach cost function is completely tunable, allowing for more complex variations of the problem. For example, if not all cores are connected to each other, the number of non-local communications to move a qubit to a given core may be more than one. The cost function can be adapted to this case and many others, leading to a robust and widely applicable inter-core mapping algorithm.

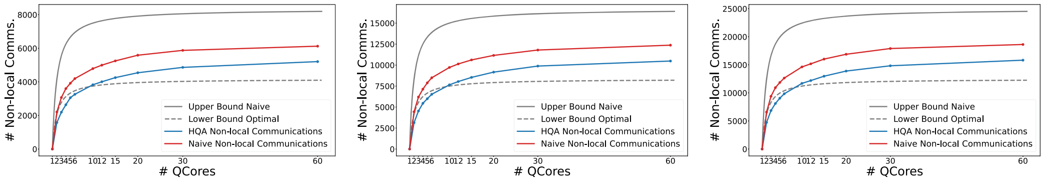
5.3 Discussion

Similar to the other two mapping algorithms discussed, we compare the HQA algorithm to the bounds on the non-local communications proposed in Theorems 3.2 and 3.3.

Figure 9 shows that, unlike the other mapping algorithms, HQA always performs better than the Naive approach, staying always below the upper bound and obtaining a lower number of communications than the lower bound for a small number of cores.

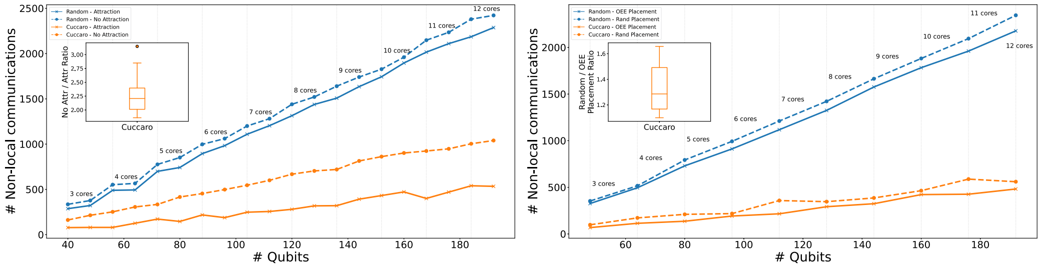
We attribute such improvement to the change of approach the proposed algorithm uses. Instead of partitioning a graph that describes the quantum circuit, the HQA assigns unfeasible two-qubit gates to cores, which directly focuses on minimizing the number of non-local communications.

Figure 10(a) depicts a comparison of the two proposed versions of the HQA algorithm, with and without considering future qubit interactions for the cost function. It can be seen that, by considering future qubit interaction and computing the attraction force as formulated in Equation (22), we manage to decrease the number of non-local communications for structured (Cuccaro) and unstructured (Random) quantum circuits.



(a) Random S HQA Mapping. (b) Random M HQA Mapping. (c) Random L HQA Mapping.

Fig. 9. Non-local communications when using the HQA algorithm for different Quantum Random Circuits. Communication bounds are shown in grey, and Naive Mapping is shown in red.



(a) HQA communications with and without using the attraction force for future qubit interactions. The use of attraction forces improves the number of non-local communications by 2.27× for the Cuccaro Adder and by 1.09× for the Random Circuit, on average.

(b) HQA communications with Random initial placement and OEE initial placement. The use of the OEE initial placement improves the number of non-local communications by 1.33× for the Cuccaro Adder and by 1.07× for the Random Circuit, on average.

Fig. 10. Different versions of the HQA, mapped into an architecture with 16 qubits per core and as many cores as needed, depending on the number of qubits used (x-axis). A structured circuit (Cuccaro Adder) and an unstructured one (Random) have been selected to highlight the importance of the attraction force. In each figure, the box plot depicts the Cuccaro Adder communications ratio with the two approaches.

Moreover, we also study the impact of the initial distribution of qubits to cores for the algorithm. Figure 10(b) shows the number of non-local communications when using a random initial partition or the graph partition obtained using the OEE [41], as proposed in [3]. We show that the mapping algorithm performs worse when starting with a random partition than when starting with a partition obtained by the OEE algorithm.

Therefore, from the results depicted in Figure 10, we will use the HQA considering future qubit interactions and starting with a partition for the experiments carried out in future sections, where we deeply analyze the performances of the three reviewed mapping algorithms, assessing them in terms of optimality (number of non-local communications) and efficiency (execution time).

6 PERFORMANCE EVALUATION OF HQA

To compare the different mapping algorithms explained before, we will focus on the scalability problem, increasing the number of cores and qubits, to analyze how the different approaches adapt to the increasing of resources. To do so, three different sets of experiments are proposed:

- For the first scalability approach, *Virtual Scaling*, a fixed-size multi-core quantum computing architecture of 10 cores and 10 qubits per core (100 physical qubits in total) has been set; it consists of mapping a quantum circuit with increasing the number of virtual qubits from 50 virtual qubits, where half of the qubits of the architecture will be used as ancillary qubits, to 100 virtual qubits, where all qubits from the architecture will be used as data qubits and no ancillary qubits will be used. With this experiment, we aim to assess the importance of ancillary qubits in multi-core mapping.

- The *Weak Scaling* approach involves mapping a quantum circuit of 200 virtual qubits into a quantum hardware with 200 physical qubits, varying the numbers of cores (from 2 to 10 cores) and qubits per core. The number of qubits per core will depend on the architecture's number of cores, ensuring that all cores will have the same number of qubits. Therefore, no ancillary qubits will be used in any hardware configuration for this approach. This experiment demonstrates how the number of communications across cores and execution time vary with the number of architecture cores and how the selected mapping algorithms perform when increasing the number of architecture cores. Due to the high number of qubits used in this experiment, we only show the performance of the proposed Naive approach (Algorithm 1), the FGP-rOEE [3], and the proposed HQA (Algorithm 2), excluding the QUBO Mapping algorithm [4] given its high runtime.
- Lastly, the last scaling approach, *Strong Scaling*, consists of increasing the number of cores (with 10 qubits each) in the architecture, from 2 cores (20 qubits) up to 20 cores (200 qubits); in each case the number of virtual qubits will be the same as the physical qubits. Again, due to the high execution time of the QUBO mapping algorithm [4], the set of experiments using this algorithm will range only up to 10 cores (100 qubits). With this experiment, we show how the algorithms behave when increasing both the number of qubits and cores, covering all the different types of scaling.

6.1 Benchmarks and Performance Metrics

For all scaling approaches, the same set of benchmarks is selected, containing high-structured quantum algorithms such as Quantum Fourier Transform [10, 39], Draper Adder [15], and Cuccaro Adder [12], as well as unstructured quantum algorithms such as Random Quantum Circuits (with a depth two times the number of virtual qubits used) and Quantum Volume [11].

We have used Qiskit's [45] implementation of the algorithms. Each algorithm is sliced into timeslices as proposed by [4], and the same set of slices is used as input for each mapping algorithm.

For the HQA experiments, we use the cost function that considers future qubit interactions (described in Equation (22)), and the initial assignments of qubits into cores will be the one obtained by the OEE [41] applied to the total interaction graph. Both decisions are supported by the exploration conducted in Section 5.3, and summarized in Figure 10.

The main performance metric used in this work is the number of non-local communications (i.e., communications across cores). The Naive algorithm proposed in Algorithm 1 will be used as a baseline for the number of non-local communications.

Moreover, we also analyze the execution time of each mapping algorithm, showing the speedup of the HQA algorithm compared to the FGP-rOEE and QUBO Mapping algorithms.

All experimental procedures were conducted on a computing system featuring an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40 GHz, equipped with 131.7 GB of RAM and 40 cores, operating on CentOS Linux 7. The simulation procedure was implemented utilizing Python 3.8, and the benchmarking tasks were facilitated through the framework provided by Qiskit [45]. The QUBO implementation was obtained from its open repository [4], and FGP-rOEE has been implemented according to its proposal in [3].

6.2 Results

This section analyzes the three different mapping algorithms based on the different scaling experiments explained in the last section: Virtual, Weak, and Strong Scaling. For each approach, we show the number of non-local communications for all the selected benchmarks, as well as an

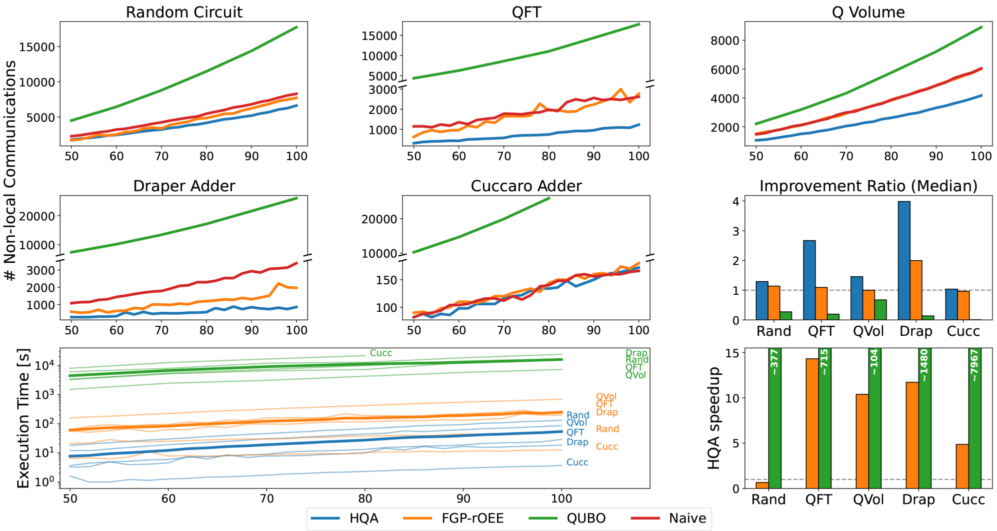


Fig. 11. Virtual Scaling. Mapping quantum algorithms into a 10-core architecture with 10 qubits per core (100 total qubits), increasing the number of qubits used in the circuit from 50 qubits to 100 qubits. The first two rows show the number of non-local communications for each one of the approaches, and the last row shows the execution time for FGP-rOEE, QUBO, and HQA. The bar plots show the improvement ratio for both the number of non-local communications and the execution time.

improvement ratio of FGP-rOEE, QUBO, or HQA over the Naive approach (median across the different data points).

Moreover, the execution time of all mapping algorithms for each benchmark is depicted in the lower row of Figures 11 through 13, along with the average execution time, over all the benchmarks. Lastly, a SpeedUp of the HQA over FGP-rOEE and QUBO is shown, highlighting the HQA algorithm's value.

6.2.1 *Virtual Scaling.* Figure 11 shows, in the first two rows, the non-local communications needed for each benchmark when increasing the number of virtual qubits of the circuit (x-axis). Results show how the number of non-local communications increases when increasing the number of virtual qubits in the circuit, independently of the quantum benchmark or the mapping algorithm. This is due to the use of ancillary qubits (physical qubits of the architecture that do not store any quantum state) for allocation purposes.

The HQA algorithm outperforms all other mapping algorithms in all benchmarks except for the Cuccaro Adder, where a similar number of non-local communications is obtained for HQA, FGP-rOEE, and the Naive approach. For most benchmarks, FGP-rOEE obtains a similar number of non-local communications than the Naive approach, highlighting that the graph partitioning approach does not perform as well as the two-qubit operations assignment approach.

Regarding the execution time, we can see that HQA outperforms QUBO in all benchmarks, and FGP-rOEE in most of them, as, for Random Circuits, the compilation time of HQA is ~ 0.67× faster (~ 1.5× slower) than FGP-rOEE. For all the other benchmarks, HQA managed to obtain the assignment with a speedup of up to 15×. The high execution time for the QUBO Mapping algorithm was expected, as the graph partition problem is posed as a quadratic optimization problem.

6.2.2 *Weak Scaling.* Figure 12 shows, in the first two rows, the non-local communications needed for each benchmark when increasing the number of cores of the architecture (x-axis), and

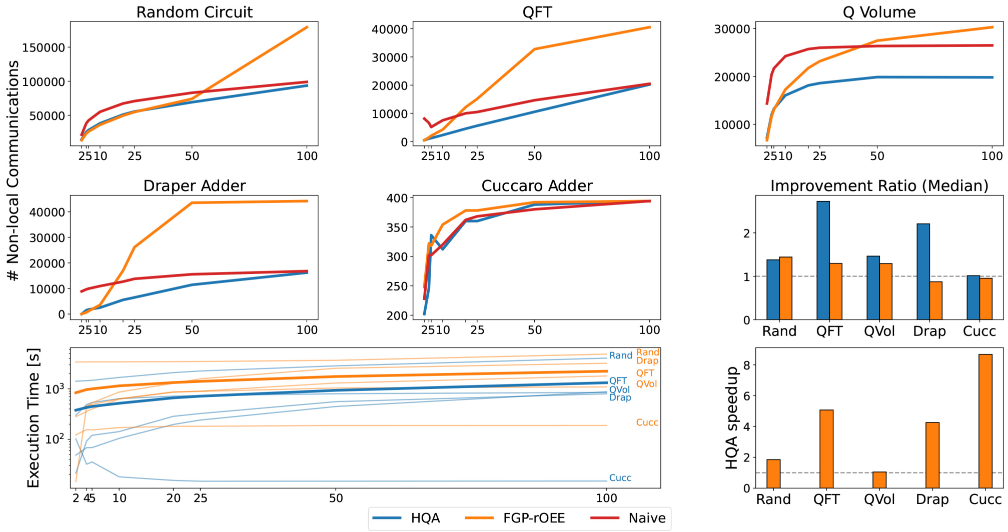


Fig. 12. Weak Scaling. Mapping quantum algorithms into a 200-qubit architecture with varying numbers of cores. The first two rows show the number of non-local communications for each one of the approaches, and the last row shows the execution time for FGP-rOEE and HQA. The bar plots show the improvement ratio for both the number of non-local communications and the execution time.

the execution time of each experiment in the bottom row. It can be seen how, for all benchmarks, for all mapping algorithms, the number of non-local communications increases as does the number of cores of the architecture. Due to the high amount of qubits used in this experiment (architecture with 200 physical qubits), the QUBO algorithm has been discarded, as it took too much time to execute.

With this scaling approach, we aim to study how mapping algorithms perform under edge cases, such as having an architecture with 100 cores and two qubits per core. Under these cases, we can see that, independently of the benchmark, FGP-rOEE performs much worse than the HQA algorithm and the Naive baseline. This is due again to the graph partitioning approach this algorithm uses, as it performs swaps of qubits among cores, until obtaining a valid assignment. When the number of cores is too high, the possible swaps of qubits increase, making the heuristics fail in choosing which pair of qubits to swap.

Thanks to the new approach of assigning unfeasible two-qubit gates instead of partitioning the graph, the HQA algorithm achieves a lower number of non-local communications than the FGP-rOEE in most benchmarks, improving the Naive baseline in all of them.

Regarding the execution time, we can see that the HQA found the solution in less time than or equal time as FGP-rOEE for all benchmarks, noting a speedup of more than 8× for the Cuccaro Adder benchmark.

6.2.3 *Strong Scaling.* Lastly, Figure 13 shows, in the first two rows, the non-local communications needed for each benchmark when increasing the number of cores and qubits of the architecture (x-axis) and the execution time of each experiment in the bottom row. In this experiment, the size of each core is fixed to 10 qubits per core, increasing the number of qubits in the circuit according to the number of qubits in the architecture. Due to the high execution cost of the QUBO algorithm, we have restricted the experiments from a 2-core architecture with 20 qubits up to an

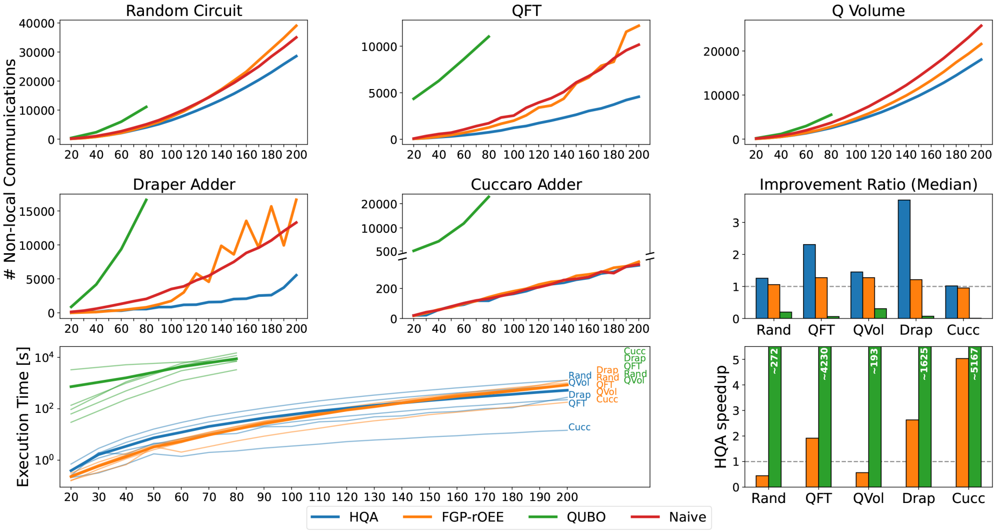


Fig. 13. Strong Scaling. Mapping quantum circuits into a k-core architecture, increasing the number of cores, with a fixed size of 10 qubits per core. The first two rows show the number of non-local communications for each one of the approaches, and the last row shows the execution time for FGP-rOEE, QUBO, and HQA. The bar plots show the improvement ratio for both the number of non-local communications and the execution time.

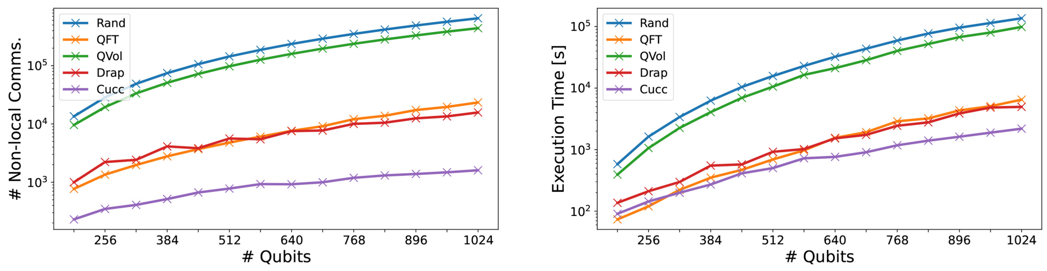
8-core architecture with 80 qubits, while for the FGP-rOEE and HQA mapping algorithms, we scale the architecture up to a 20-core architecture with 200 qubits.

As expected, it can be seen how, as we increase the amount of qubits in the circuit (and in the architecture), the number of non-local communications also increases. For all benchmarks, the HQA manages to obtain a lower number of non-local communications than the baseline, except for the Cuccaro Adder, where it achieves similar non-local communications.

The HQA also outperforms FGP-rOEE and QUBO in all benchmarks. The QUBO algorithm always performs worse than the baseline, and the FGP-rOEE algorithm performs similarly to the baseline for most benchmarks. On average, across all benchmarks and scaling approaches, the HQA improves the number of non-local communications obtained by FGP-rOEE by 1.556×.

Regarding the execution time, again, we see a high execution time for the QUBO algorithm, as it poses a graph partitioning problem using quadratic optimization. On average, for a low number of cores, the FGP-rOEE mapping algorithm obtains a faster solution than the HQA. However, as the number of cores increases, the FGP-rOEE execution time increases much faster than the HQA's, as it mostly depends on the number of cores and the number of qubits of the system, ending with a lower execution time for the HQA mapping algorithm, highlighting its scalability. For Random Circuits and Quantum Volume, which are unstructured circuits, the number of unfeasible two-qubit gates drives the HQA's computation time, leading to a higher execution time. Nevertheless, even for unstructured circuits, FGP-rOEE takes more time than HQA once the number of cores or qubits is high enough.

Without taking into account the HQA, FGP-rOEE is the best-performing multi-core mapping algorithm. The HQA algorithm managed to improve FGP-rOEE's number of non-local communications by 1.556×, reducing its execution time by 4.897×, on average, across all benchmarks and scalability approaches.



(a) HQA number of communications.

(b) HQA execution time.

Fig. 14. HQA number of communications and execution time for a system with a fixed core size of 64 qubits.

Figure 14 shows the behavior of HQA when scaling up to 1,024 qubits, also in a Strong Scaling way, with 64 qubits per core and an increasing number of cores.

7 CONCLUSIONS AND FUTURE WORK

Multi-core quantum computing architectures offer a promising path to overcoming the limitations of monolithic processors and enabling the execution of complex quantum algorithms. In this article, we have explored the intricate landscape of multi-core quantum computing, illustrating the challenges and opportunities ahead.

In this work, we have proposed theoretical bounds on the non-local communications needed to map Random Circuits into modular architectures, proving state-of-the-art multi-core mappers to be far from optimal. Moreover, we propose a novel approach, changing the paradigm of multi-core mapping from graph partitioning to two-qubit gate assignment. Throughout rigorous evaluation across different quantum algorithms and scaling approaches, we have shown the potential of HQA, obtaining better results than its analogous mapping algorithms (1.6× improvement over FGP-rOEE) while decreasing the execution time (4.9× speedup over FGP-rOEE), showcasing its potential to be scaled up.

Much more needs to be done in mapping for multi-core quantum computers. Our approach focuses on those modular architectures based on the generation and distribution of entangled states. However, other modular approaches will require different mapping algorithms that use other communication primitives than the ones used in this work. Moreover, other EPR-based communication primitives can be added to the mapping algorithm to take advantage of all types of quantum communications.

The main reason for scalability in quantum systems is to incorporate quantum error correction [39] and fault-tolerant techniques that will allow reliable and accurate computations. Therefore, it is crucial to investigate not only how to integrate quantum error correction in these new modular architectures but also what runtime and compiler support will be required considering the constraints imposed by various correction codes, leveraging them to enhance the mapping optimization process.

REFERENCES

[1] Matthew Amy and Vlad Gheorghiu. 2020. staq–A full-stack quantum processing toolkit. *Quantum Science and Technology* 5, 3 (June 2020), 034016. https://doi.org/10.1088/2058-9565/ab9359

[2] Pablo Andres-Martinez, Tim Forrer, Daniel Mills, Jun-Yi Wu, Luciana Henaut, Kentaro Yamamoto, Mio Murao, and Ross Duncan. 2023. Distributing Circuits Over Heterogeneous, Modular Quantum Computing Network Architectures. https://doi.org/10.48550/arXiv.2305.14148 arXiv:2305.14148 [quant-ph]

[3] Jonathan M. Baker, Casey Duckering, Alexander Hoover, and Frederic T. Chong. 2020. Time-sliced quantum circuit partitioning for modular architectures. In *Proceedings of the 17th ACM International Conference on Computing Frontiers (CF'20)*. Association for Computing Machinery, New York, NY, USA, 98–107. https://doi.org/10.1145/3387902.3392617

[4] Medina Bandic, Luise Prielinger, Jonas Nüsslein, Anabel Ovide, Santiago Rodrigo, Sergi Abadal, Hans van Someren, Gayane Vardoyan, Eduard Alarcon, Carmen G. Almudever, and Sebastian Feld. 2023. Mapping quantum circuits to modular architectures with QUBO. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE'23)*, Vol. 1. 790–801. https://doi.org/10.1109/QCE57702.2023.00094

[5] Alexandre Blais, Arne L. Grimsmo, S. M. Girvin, and Andreas Wallraff. 2021. Circuit quantum electrodynamics. *Reviews of Modern Physics* 93, 2 (May 2021), 025005. https://doi.org/10.1103/RevModPhys.93.025005

[6] Sergey Bravyi, Oliver Dial, Jay M. Gambetta, Darí o Gil, and Zaira Nazario. 2022. The future of quantum computing with superconducting qubits. *Journal of Applied Physics* 132, 16 (Oct. 2022). https://doi.org/10.1063/5.0082975

[7] Jerry Chow, Oliver Dial, and Jay Gambetta. 2021. IBM Quantum Breaks the 100-qubit Processor Barrier. https://research.ibm.com/blog/127-qubit-quantum-processor-eagle

[8] Jerry M. Chow. 2021. Quantum intranet. *IET Quantum Communication* 2, 1 (April 2021), 26–27. https://doi.org/10.1049/qtc2.12002

[9] Juan I. Cirac and Peter Zoller. 1995. Quantum computations with cold trapped ions. *Physical Review Letters* 74, 20 (May 1995), 4091–4094. https://doi.org/10.1103/PhysRevLett.74.4091

[10] Don Coppersmith. 2002. An Approximate Fourier Transform useful in Quantum Factoring. arXiv:quant-ph/0201067 [quant-ph]

[11] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. 2019. Validating quantum computers using randomized model circuits. *Physical Review A* 100, 3 (Sept. 2019), 032328. https://doi.org/10.1103/PhysRevA.100.032328

[12] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. 2004. A New Quantum Ripple-carry Addition Circuit. arXiv:quant-ph/0410184 [quant-ph]

[13] Daniele Cuomo, Marcello Caleffi, Kevin Krsulich, Filippo Tramonto, Gabriele Agliardi, Enrico Prati, and Angela Sara Cacciapuoti. 2023. Optimized compiler for distributed quantum computing. *ACM Transactions on Quantum Computing* 4, 2, Article 15 (Feb. 2023), 29 pages. https://doi.org/10.1145/3579367

[14] Yongshan Ding, Pranav Gokhale, Sophia Fuhui Lin, Richard Rines, Thomas Propson, and Frederic T. Chong. 2020. Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'20)*. 201–214. https://doi.org/10.1109/MICRO50266.2020.00028

[15] Thomas G. Draper. 2000. Addition on a Quantum Computer. arXiv:quant-ph/0008033 [quant-ph]

[16] Bryan Dury and Olivia Di Matteo. 2020. A QUBO Formulation for Qubit Allocation. arXiv:2009.00140 [quant-ph]

[17] A. Einstein, B. Podolsky, and N. Rosen. 1935. Can quantum-mechanical description of physical reality be considered complete? *Physical Review* 47, 10 (May 1935), 777–780. https://doi.org/10.1103/PhysRev.47.777

[18] Pau Escofet, Anabel Ovide, Carmen G. Almudever, Eduard Alarcón, and Sergi Abadal. 2023. Hungarian qubit assignment for optimized mapping of quantum circuits on multi-core architectures. *IEEE Computer Architecture Letters* 22, 2 (July 2023), 161–164. https://doi.org/10.1109/LCA.2023.3318857

[19] Pau Escofet, Sahar Ben Rached, Santiago Rodrigo, Carmen G. Almudever, Eduard Alarcón, and Sergi Abadal. 2023. Interconnect fabrics for multi-core quantum processors: A context analysis. In *Proceedings of the 16th International Workshop on Network on Chip Architectures (NoCArc'23)*. Association for Computing Machinery, New York, NY, USA, 34–39. https://doi.org/10.1145/3610396.3623267

[20] Davide Ferrari, Stefano Carretta, and Michele Amoretti. 2023. A modular quantum compilation framework for distributed quantum computing. *IEEE Transactions on Quantum Engineering* 4 (2023), 1–13. https://doi.org/10.1109/TQE.2023.3303935

[21] Jay Gambetta. 2023. The Hardware and Software for the Era of Quantum Utility Is Here. https://research.ibm.com/blog/quantum-roadmap-2033

[22] Alysson Gold, J. P. Paquette, Anna Stockklauser, Matthew J. Reagor, M. Sohaib Alam, Andrew Bestwick, Nicolas Didier, Ani Nersisyan, Feyza Oruc, Armin Razavi, Ben Scharmann, Eyob A. Sete, Biswajit Sur, Davide Venturelli, Cody James Winkleblack, Filip Wudarski, Mike Harburn, and Chad Rigetti. 2021. Entanglement across separate silicon dies in a modular superconducting qubit device. *NPJ Quantum Information* 7, 1 (Sept. 2021), 1–10. https://doi.org/10.1038/s41534-021-00484-1

[23] Daniel Gottesman and Isaac L. Chuang. 1999. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature* 402 (Nov. 1999), 390–393. https://doi.org/10.1038/46503

[24] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC'96)*. Association for Computing Machinery, New York, NY, USA, 212–219. https://doi.org/10.1145/237814.237866

[25] Peter L. Hammer and Sergiu Rudeanu. 1969. Pseudo-boolean programming. *Operations Research* 17, 2 (1969), 233–261. https://doi.org/10.1287/opre.17.2.233

[26] A. Imamoglu, D. D. Awschalom, G. Burkard, D. P. DiVincenzo, D. Loss, M. Sherwin, and A. Small. 1999. Quantum information processing using quantum dot spins and cavity QED. *Physical Review Letters* 83, 20 (Nov. 1999), 4204–4207. https://doi.org/10.1103/PhysRevLett.83.4204

[27] Hui Jiang, Yuxin Deng, and Ming Xu. 2021. Quantum Circuit Transformation Based on Subgraph Isomorphism and Tabu Search.

[28] Hamza Jnane, Brennan Undseth, Zhenyu Cai, Simon C. Benjamin, and Bálint Koczor. 2022. Multicore quantum computing. *Physical Review Applied* 18, 4 (Oct. 2022), 044064. https://doi.org/10.1103/PhysRevApplied.18.044064 Publisher: American Physical Society.

[29] B. W. Kernighan and S. Lin. 1970. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal* 49, 2 (1970), 291–307. https://doi.org/10.1002/j.1538-7305.1970.tb01770.x

[30] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. 2007. Linear optical quantum computing with photonic qubits. *Reviews of Modern Physics* 79, 1 (Jan. 2007), 135–174. https://doi.org/10.1103/RevModPhys.79.135

[31] H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1–2 (1955), 83–97. https://doi.org/10.1002/nav.3800020109

[32] P. Kurpiers, P. Magnard, T. Walter, B. Royer, M. Pechal, J. Heinsoo, Y. Salathé, A. Akin, S. Storz, J.-C. Besse, S. Gasparinetti, A. Blais, and A. Wallraff. 2018. Deterministic quantum state transfer and remote entanglement using microwave photons. *Nature* 558, 7709 (June 2018), 264–267. https://doi.org/10.1038/s41586-018-0195-y

[33] Lingling Lao, Hans van Someren, Imran Ashraf, and Carmen G. Almudever. 2022. Timing and resource-aware mapping of quantum circuits to superconducting processors. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 41, 2 (2022), 359–371. https://doi.org/10.1109/TCAD.2021.3057583

[34] Nicholas LaRacuente, Kaitlin N. Smith, Poolad Imany, Kevin L. Silverman, and Frederic T. Chong. 2023. Modeling Short-Range Microwave Networks to Scale Superconducting Quantum Computation. arXiv:2201.08825 [quant-ph]

[35] Gushu Li, Yufei Ding, and Yuan Xie. 2019. Tackling the qubit mapping problem for NISQ-Era quantum devices. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'19)*. Association for Computing Machinery, New York, NY, USA, 1001–1014. https://doi.org/10.1145/3297858.3304023

[36] Guang Hao Low and Isaac L. Chuang. 2019. Hamiltonian simulation by qubitization. *Quantum* 3 (July 2019), 163. https://doi.org/10.22331/q-2019-07-12-163

[37] Y. Nakamura, Yu. A. Pashkin, and J. S. Tsai. 1999. Coherent control of macroscopic quantum states in a single-Cooper-pair box. *Nature* 398, 6730 (April 1999), 786–788. https://doi.org/10.1038/19718

[38] National Academies of Sciences, Engineering, and Medicine. 2019. *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC. https://doi.org/10.17226/25196

[39] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press. https://doi.org/10.1017/CBO9780511976667

[40] Anabel Ovide, Santiago Rodrigo, Medina Bandic, Hans Van Someren, Sebastian Feld, Sergi Abadal, Eduard Alarcon, and Carmen G. Almudever. 2023. Mapping quantum algorithms to multi-core quantum computing architectures. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS'23)*. 1–5. https://doi.org/10.1109/ISCAS46773.2023.10181589

[41] Taehoon Park and Chae Y. Lee. 1995. Algorithms for partitioning a graph. *Computers & Industrial Engineering* 28, 4 (1995), 899–909. https://doi.org/10.1016/0360-8352(95)00003-J

[42] Christophe Piveteau and David Sutter. 2023. Circuit Knitting with Classical Communication. arXiv:2205.00016 [quant-ph]

[43] I. Pogorelov, T. Feldker, Ch. D. Marciniak, L. Postler, G. Jacob, O. Krieglsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, B. Höfer, C. Wächter, K. Lakhmanskiy, R. Blatt, P. Schindler, and T. Monz. 2021. Compact ion-trap quantum computing demonstrator. *PRX Quantum* 2, 2 (June 2021), 020343. https://doi.org/10.1103/PRXQuantum.2.020343

[44] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. https://doi.org/10.22331/q-2018-08-06-79

[45] Qiskit Contributors. 2023. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2573505

[46] Santiago Rodrigo, Sergi Abadal, Eduard Alarcón, Medina Bandic, Hans van Someren, and Carmen G. Almudéver. 2021. On double full-stack communication-enabled architectures for multicore quantum computers. *IEEE Micro* 41, 5 (2021), 48–56. https://doi.org/10.1109/MM.2021.3092706

[47] Peter W. Shor. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* 26, 5 (1997), 1484–1509. https://doi.org/10.1137/S0097539795293172

[48] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. 2020. t|ket⟩: A retargetable compiler for NISQ devices. *Quantum Science and Technology* 6, 1 (Nov. 2020), 014003. https://doi.org/10.1088/2058-9565/ab8e92

[49] Sergei Slussarenko and Geoff J. Pryde. 2019. Photonic quantum information processing: A concise review. *Applied Physics Reviews* 6, 4 (Oct. 2019), 041303. https://doi.org/10.1063/1.5115814

[50] K. N. Smith, G. Ravi, J. M. Baker, and F. T. Chong. 2022. Scaling superconducting quantum computers with chiplet architectures. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO'22)*. IEEE Computer Society, Los Alamitos, CA, USA, 1092–1109. https://doi.org/10.1109/MICRO56248.2022.00078

[51] Ajit Srivastava, Meinrad Sidler, Adrien Allain, Dominik Lembke, András Kis, and Ataç İmamoğlu. 2015. Optically active quantum dots in monolayer WSe2. *Nature Nanotechnology* 10 (May 2015), 491–496. https://doi.org/10.1038/nnano.2015.60

[52] Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. 2021. CutQC: Using small quantum computers for large quantum circuit evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'21)*. Association for Computing Machinery, New York, NY, USA, 473–486. https://doi.org/10.1145/3445814.3446758

[53] T. Tomesh, P. Gokhale, V. Omole, G. Ravi, K. N. Smith, J. Viszlai, X. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong. 2022. SupermarQ: A scalable quantum benchmark suite. In *2022 IEEE International Symposium on High-performance Computer Architecture (HPCA'22)*. IEEE Computer Society, Los Alamitos, CA, USA, 587–603. https://doi.org/10.1109/HPCA53966.2022.00050

[54] Anbang Wu, Hezi Zhang, Gushu Li, Alireza Shabani, Yuan Xie, and Yufei Ding. 2022. AutoComm: A framework for enabling efficient communication in distributed quantum programs. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO'22)*. 1027–1041. https://doi.org/10.1109/MICRO56248.2022.00074

[55] Hezi Zhang, Keyi Yin, Anbang Wu, Hassan Shapourian, Alireza Shabani, and Yufei Ding. 2024. MECH: Multi-Entry Communication Highway for Superconducting Quantum Chiplets. arXiv:2305.05149 [quant-ph]

Received 30 September 2023; revised 16 February 2024; accepted 24 March 2024