Mapping quantum circuits to modular architectures with QUBO

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 [*]

[*] *Delft University of Technology (QuTech)*, The Netherlands
[‡] *Universitat Politècnica de Valencia*, Spain
[§] *Universitat Politècnica de Catalunya*, Spain
[¶] *LMU Munich*, Germany

[†] These authors contributed equally to this work.

***Abstract*—Modular quantum computing architectures are a promising alternative to monolithic QPU (Quantum Processing Unit) designs for scaling up quantum devices. They refer to a set of interconnected QPUs or cores consisting of tightly coupled quantum bits that can communicate via quantum-coherent and classical links. In multi-core architectures, it is crucial to minimize the amount of communication between cores when executing an algorithm. Therefore, mapping a quantum circuit onto a modular architecture involves finding an optimal assignment of logical qubits (qubits in the quantum circuit) to different cores with the aim to minimize the number of expensive inter-core operations while adhering to given hardware constraints. In this paper, we propose for the first time a Quadratic Unconstrained Binary Optimization (QUBO) technique to encode the problem and the solution for both qubit allocation and inter-core communication costs in binary decision variables. To this end, the quantum circuit is split into slices, and qubit assignment is formulated as a graph partitioning problem for each circuit slice. The costly inter-core communication is reduced by penalizing inter-core qubit communications. The final solution is obtained by minimizing the overall cost across all circuit slices. To evaluate the effectiveness of our approach, we conduct a detailed analysis using a representative set of benchmarks having a high number of qubits on two different multi-core architectures. Our method showed promising results and performed exceptionally well with very dense and highly-parallelized circuits that require on average 0.78 inter-core communications per two-qubit gate.**

***Index Terms*—quantum circuit mapping, distributed multi-core quantum computing architectures, Quadratic Unconstrained Binary Optimization (QUBO), quantum compilation, full-stack quantum computing systems**

I. Introduction

A major challenge for current NISQ (Noisy Intermediate Scale Quantum) devices is scalability. These processors suffer from high error rates and limited qubit counts and connectivity, which hinder the demonstration of the full potential of quantum computing. Well-known quantum algorithms, such as Shor's algorithm for factoring large numbers and Grover's algorithm for searching an unsorted database, are expected to provide significant speedups over classical algorithms, but these benefits will likely only be realized with quantum computers that have far more qubits than current systems, which are typically limited to hundreds of qubits. Most present-day quantum computers are implemented as single-processor devices, i.e., a single chip that contains all qubits. These designs are hard to scale mostly due to crosstalk [1] and limitations related to control electronics [2]. An alternative architectural design to scale quantum computers, similar to classical computing, lies in multi-processor (multi-core) computers, which have already been proposed by various quantum processor manufacturers. [3–6].

These new architectures will enable distributed multi-core quantum computing, that is, executing a large algorithm consisting of more qubits than there are in a single processor by distributing it over different cores. In this case, similarly to the resource-constrained NISQ devices, a quantum circuit mapping process [7] is required to ensure the efficient use of hardware resources and in turn to maximize the algorithm success rate. Considering that qubit interactions within a core are negligible when compared to those in between cores in terms of operation time and fidelity [8], quantum circuit mapping in the multi-core regime is mainly focused on minimizing the amount of inter-core communications (operations between qubits that are in different processors). This is mostly achieved by finding a good assignment or allocation of the logical qubits (qubits in the circuit) to the physical qubits in different cores and by optimally performing inter-core operations when required.

Finding an optimal solution for the mapping problem can be computationally infeasible for a large number of qubits, even for single-core devices [9]. To address this challenge, various mapping algorithms have been proposed [10]. However, due to the early-stage development of modular quantum computing architectures, just a few quantum circuit mapping techniques [8, 11] have been explored, and they are only tested on architectures with a small number of qubits and simple, unrealistic all-to-all connectivity between qubits with limited benchmark sets.

In contrast to prior approaches, we address the circuit mapping problem for multi-processor devices using a quadratic unconstrained binary optimization (QUBO) formulation. This

 DOI 10.1109/QCE57702.2023.00094

novel approach is employed for solving qubit allocation, as well as inter-core qubit communication and routing. Among the advantageous properties of this QUBO-based approach, four stand out:

i. The QUBO formulation a priori encompasses the entire solution landscape for the quantum circuit mapping problem, and therefore does not exhibit limitations stemming from a reliance on look-ahead functions. The latter or other local estimates are a common strategy for decision-making regarding qubit placement and transfer [8].
ii. Unlike previous mapping solutions, the optimization process is decoupled from the objective function, thereby enabling the selection of a tailored optimization method, depending on the size and scope of the problem at hand. More precisely, the method can be flexibly adapted to a use case by choosing a suited exact or heuristic solver without reformulating the method, for a small or large quantum circuit to be mapped, respectively.
iii. The aforementioned separation between the objective function and finding its optimal solutions allows for the latter to evolve with the development of new solving approaches. This development includes not only even more powerful software-based solvers [12] for QUBO instances, but also the construction of single-purpose quantum hardware, a so-called quantum annealer [13].
iv. The k-partitioning problem [14], which is often used in qubit mapping for multi-core systems [8], can be expressed as a quadratic objective function and has been well-defined in the literature [15]. This makes it an ideal candidate for a QUBO formulation.

The aim of our method is to find the optimal placement of qubits for all circuit slices (Fig. 1) to a multi-core system by representing the per-slice qubit allocation as a k-partitioning problem and penalizing the cost of any inter-core communication simultaneously. This method is tested by mapping an extensive set of benchmark circuits, that cover a substantial range of parameters, including the number of qubits, circuit depth, gate density and structure, to two different multi-core architecture topologies. We hope that this work represents a stepping stone for the development of new mapping techniques for modular architectures that will be essential components of future full-stack quantum systems [16].

The paper is organized as follows: Sec. II introduces the quantum circuit mapping problem, discusses the challenges of multi-core quantum computation, and presents the concept of QUBO in general. In Sec. III our QUBO-based approach for mapping quantum circuits to multi-core devices is described in detail, including the method's objectives, definition and proofs of the objective function. Sec. IV presents the experimental setup, the benchmark set used and the chosen hardware platforms. Sec. V discusses the obtained results. Finally in Sec. VI we describe possible future directions including potential solutions to overcome current limitations and conclude the paper.

II. Background and Related Work

A. Mapping of quantum circuits: from single to multiple cores

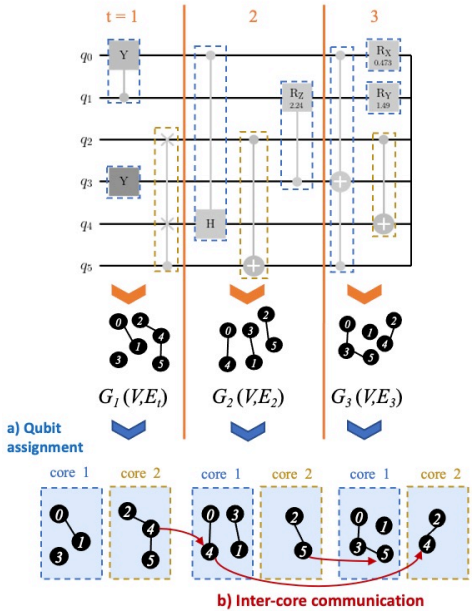

Fig. 1. Graphical outline of Ex. 1. A quantum circuit is split into three circuit slices $t = \{1, 2, 3\}$ (where the circuit slices' borders are marked by solid vertical lines in orange) represented by the respective interaction graphs $G_t(V, E_t)$. The partitioned circuit is mapped to a $k = 2$ core system (two blue shaded rectangles below each circuit slice), each with a capacity of $c_1 = c_2 = 4$ physical qubits. **a)** Logical qubits (black circles) sharing a gate interaction are assigned to the same core (dashed rectangles). The first gate $(0, 1)$ of the circuit is assigned to core 1 marked in blue dashed rectangles, while the three-qubit gate involving qubits (2,4,5) is assigned to core 2. **b)** If a qubit is assigned to a different core in subsequent slices, it must undergo a state transfer, i.e., an inter-core communication, shown by red arrows. Hence, the solution shown requires three inter-core qubit communications.

Due to the previously mentioned limitations of current NISQ devices, algorithms (in the form of quantum circuits) may need to be modified in order to be executed, through a process called quantum circuit mapping [10]. This process involves: i) assigning the logical qubits of the quantum circuit to the physical qubits of the quantum chip, an example of which is provided in Fig. 1; ii) moving (routing) the logical qubits that need to interact in form of two-qubit gates to adjacent positions on the chip. Note that this is necessary as physical qubits for most quantum technologies are not all-to-all connected; and in some cases includes iii) scheduling operations to maximally parallelize them with the aim of lowering the execution time, which is important due to the decoherence of qubits [16]. The quantum circuit mapping process can vary from device to device due to differences in technology and qubit connectivity. It is essential to perform this task to guarantee the effective utilization of scarce hardware resources and to decrease the likelihood of errors that may arise during the execution of quantum operations by minimizing the number of additional gates and the circuit latency. However, solving the quantum circuit mapping

problem for a large number of qubits can be computationally unfeasible, even for contemporary single-core devices. To tackle it, diverse mapping algorithms have been proposed, including heuristic or brute-force strategies, graph-theoretical techniques, dynamic programming algorithms, and machine learning-based solutions [2, 7, 10, 17–24].

These mapping techniques used in single-processor NISQ devices are however not applicable to multi-core quantum computing architectures, which have emerged as a promising approach to scale-up quantum computing systems. In the multi-core architectural approach, cores (QPUs) are connected with classical and quantum communication links. Quantum links allow to 'move' quantum states from one core to another or to perform inter-core quantum gates [25] (e.g., remote extended CNOT) making use of entanglement. Classical links are required to assist core coordination and job distribution [26]. The architectural complexity involving the communication channels and traffic make it more difficult to perform quantum circuit mapping when compared with mapping to single-core devices [27, 28].

This consequently led to the development of new techniques for multi-core architectures whose main aim is to reduce the number of expensive inter-core operations. To achieve this, an optimal assignment of the logical qubits to the physical qubits of different cores is crucial. The literature on this topic is limited: some proposals focused on the quantum compilation for distributed quantum computing [11, 29], whereas others focused only on the quantum circuit mapping part of compilation [8], which is closer to our work. In [8], the quantum circuit is split into smaller partitions. Interaction graphs representing operations within a circuit slice are then mapped onto cores by grouping the qubits with the highest amount of interactions together, and while taking into account different look-ahead approaches. This was one of the first proposals for mapping quantum algorithms to multi-core devices, but it was only tested on a very simple all-to-all architecture and with a limited set of benchmarks.

In order to improve their initial approach in terms of circuit and graph partitioning with an overly-restricting local look-ahead function, we rely on previous subgraph isomorphism- [21, 24, 30–32] and QUBO optimization-based single-core solutions [33, 34] where now instead of mapping logical qubits to optimal subsets of coupling graph representing the physical qubits and their connections (Fig. 1) we map qubits to different cores. In our approach, we use a k-partitioning-based QUBO formulation, which is, together with the mapping procedure explained in more detail in the following sections.

B. Quadratic unconstrained binary optimization

The introduction of Quadratic Unconstrained Binary Optimization, termed QUBO or UBQP in literature, goes back to the 1960s [35]. We will use the following definition

Definition 1 (Quadratic unconstrained binary optimization)**.** *Quadratic Unconstrained Binary Optimization (QUBO) formulation presents a NP-hard [12] mathematical problem to be optimized. The objective function to be minimized is expressed by the formula:*

$$\min_{\mathbf{x}} \mathbf{x}^T Q \mathbf{x} = \min_{\mathbf{x}} \sum_{i<j} Q_{ij} x_i x_j + \sum_i Q_{ii} x_i, \tag{1}$$

where $\mathbf{x}$ *is a vector of binary decision variables of size* N*, i.e.,* $\mathbf{x} = \{0,1\}^N$ *and* Q *is a symmetric, square matrix of* $N \times N$ *real valued constants [12].*

In recent years the QUBO model gained scientific attention which led to the development of a wide range of applications in combinatorial optimization, as well as of effective solver techniques [36–41]. Unlike many other optimization formulations, a QUBO instance can be solved with a quantum annealer, as for example the one constructed by the hardware provider D-Wave Systems Inc.[13]. Even though the computational solving ability of these devices is still restricted to small problem sizes, the rapid evolution of these systems provides hope that more realistic problems can be addressed in the near to mid-term future [42, 43]. Therefore QUBO instances are in practice still solved with either exact or approximate classical procedures on classical computers. Unlike heuristics or approximation algorithms, exact QUBO solvers guarantee that the solution found is optimal. However, the NP-hard nature of a QUBO problem implies that exact solvers result in infeasible running times for large problem sizes [44]. Therefore, approximate or heuristic methods are often used to find near-optimal solutions within reasonable time limits. Prominent examples are simulated annealing, tabu search and steepest descent, but also commercial cloud-based solvers [45]. A comprehensive literature review of QUBO applications and solving methods can be found in [12, 44, 46].

III. Solving the mapping problem for distributed multi-core quantum computing with QUBO

In this section, we will introduce our mapping approach and explain its objectives in detail including an illustrative example. We will then give the required proofs and see how solutions can differ due to a simple scalar weighting factor λ, which we use to scale the different components of the objective function.

A. Objectives of the QUBO mapping

The quantum circuit to be mapped is split into circuit slices, each consisting of a sequence of gates. This partition is generated via a recursive slicing procedure, for which we will provide more details in Sec. III-D. A circuit slice comprises a set of quantum gates that can be performed without the use of inter-core communication[1]. A circuit slice t, therefore, presents an interaction graph $G_t(V, E_t)$, in which the logical qubits of the circuit form the node set V and set specific qubit interactions[2] form edge set E_t. Each edge $e \in E_t$ denotes at least one gate operation between qubits present in circuit slice

[1]The definition of a circuit slice is specific to this study and may not match the concept of a time slice in other publications.

[2]Only qubit interactions with $n > 1$ are relevant to the mapping procedure since single-qubit gates are not core-dependent.

t. The objective function should then a) find an assignment for each time-slice graph, i.e., all qubits involved in the same gate should be assigned to the same core without exceeding the core's capacity c_j the maximum number of logical qubits a core j can hold; and b) minimize inter-core communications between assignments, where the latter refers to the relocation of a logical qubit state between two different cores of a distributed quantum system. Tab. I summarizes the notation used in this study.

TABLE I: Notation used in this study.

Symbol	Description
n	Number of logical qubits in a quantum circuit
k	Number of cores in the multi-core system
c_j	Capacity, number of qubits that a core j can hold
T	Total number of circuit slices

Let us consider the following example:

Example 1 (Qubit assignment and inter-core communication)**.** *Given a six-qubit circuit with seven multi-qubit*[3] *gates depicted in Fig. 1 split into circuit slices $t = \{1, 2, 3\}$ (details on slicing are given in III-D). The objective is to map the former to a two-core system, each core with a capacity of $c_1 = c_2 = 4$. We can write the circuit interactions as follows:*

t	*Qubits*
1	$(0, 1), (2, 4, 5), 3$
2	$(0, 4), (2, 5), (1, 3)$
3	$(0, 3, 5), (2, 4), 1$

(2)

where each row holds the six logical qubits, and tuples hold qubits involved in the same multi-qubit gate. In each circuit slice, all n qubits need to be assigned to cores, such that qubits sharing a tuple also share the same core j, and the core's capacity c_j is not exceeded. Possible assignments are

t	*Assignment*
1	$(0, 1), 3 \mapsto core_1; (2, 4, 5) \mapsto core_2$
2	$(0, 4), (1, 3) \mapsto core_1; (2, 5) \mapsto core_2$
3	$(0, 3, 5), 1 \mapsto core_1; (2, 4) \mapsto core_2$

(3)

where figures in red indicate inter-core quantum state transfers of qubits 4 and 5. Namely,

t	*Qubit* 4	*Qubit* 5
$1-2$	$core_2 \rightarrow core_1$	
$2-3$	$core_1 \rightarrow core_2$	$core_2 \rightarrow core_1$

(4)

The assignment portion of the problem can be cast into an instance of the k-partitioning problem, a well-known graph theoretical problem [14] where k denotes the number of subsets in the partition. We accomplish this as follows:

Definition 2 (Qubit Assignment)**.** *Let $G_t = (V, E_t)$ be an interaction graph of a circuit slice t with logical qubit set V and edge set E_t, where an edge $e \in E_t$ denotes at least one gate operation between qubits present in circuit slice t. For fixed integers k and c_j, the problem is to find a partition $\Phi^t = (\Phi^t_1, \ldots, \Phi^t_k)$, of the qubit set V into at most k subsets, such that the subsets $|\Phi^t_j| \leq c_j$, and the number of cut edges is minimized. A cut edge is defined as an edge whose endpoints are in different subsets Φ_j, Φ_l, where $j \neq l$.*

[3]The method is in general not restricted to how many qubits the gates of the circuit involved. Note, however, that most quantum processors only support up to two-qubit gates.

B. The objective function

In order to define the objective function, we first need to introduce binary decision variables:

Definition 3 (Binary Solution Array $\mathbf{x}$)**.** *A solution vector $\mathbf{x}$ comprises T circuit slices $\mathbf{x} = [\mathbf{x}^1, ..., \mathbf{x}^T]$, where a solution $\mathbf{x}^t$ for time slice t presents a qubit assignment. The latter is again comprised of k subsets*

$$\mathbf{x}^t = [\mathbf{x}^t_1, ...\mathbf{x}^t_k] \tag{5}$$

where a vector $\mathbf{x}^t_j$ holds $n = |V|$ solution variables. A solution variable $x^t_{ij} = 1$ indicates that qubit $i \in V$ is assigned to core j in circuit slice t and $x^t_{ij} = 0$ if it is not assigned to the core j. This yields a problem size of $N = T \cdot k \cdot n$, where T, k and n are the numbers of circuit slices, cores and logical qubits, respectively.

In order to impose core capacities c_j during the mapping of logical qubits, we will make use of so-called slack variables. Using slack variables is a well-established technique for replacing inequalities with equations in mathematical programming problems [47]. Let us consider the following example:

Example 2. *Assume we are given a circuit slice t of a three-qubit circuit and that core j has capacity $c_j = 2$. The inequality constraint*

$$\sum_{i=1}^{3} x^t_{ij} \leq 2 \tag{6}$$

states that we can assign no more than two qubits to core j. In order to write the inequality constraint as an equation, which we can then use as a quadratic objective function term, we introduce two slack variables, $y^t_{i'j} \in \{0, 1\}^2$, where $y^t_{i'j} = 1$ signifying a physical qubit i' being occupied by a logical qubit and $y^t_{i'j} = 0$ otherwise. We can then write

$$\sum_{i=1}^{3} x^t_{ij} - (y^t_{1j} + y^t_{2j}) = 0. \tag{7}$$

Eqs. (6) and (7) are equivalent. One can always ensure compliance of the solution variables to (6) by choosing suitable slack variables in (7). As an example, if $x^t_{1j} = x^t_{2j} = 0$, $x^t_{3j} = 1$, then one of the slack variables is set to one, e.g., $y^t_{1j} = 1$, $y^t_{2j} = 0$. If Eq. (6) is violated, however, i.e., $x^t_{1j} = x^t_{2j} = x^t_{3j} = 1$, Eq. (7) cannot be satisfied either, since the sum of the slack variables cannot exceed two.

With Defs. 2 and 3, we can set up the first part of the quadratic binary objective function, namely the assignment part. To do so, we adapt the k-partitioning instance given by [15] for a circuit slice t, termed $F(\mathbf{x}^t)$. The latter encodes the optimal solution(s) $\mathbf{x}^{t*}$ at its minimum.

$$\min_{\mathbf{x}^t} F(\mathbf{x}^t) = \min_{\mathbf{x}^t} S(\mathbf{x}^t) + P(\mathbf{x}^t) + R(\mathbf{x}^t) \tag{8}$$

with

$$S(\mathbf{x}^t) := \sum_{i=1}^{n} \left(\sum_{j=1}^{k} x_{ij}^t - 1 \right)^2, \tag{9}$$

$$R(\mathbf{x}^t) := \sum_{j=1}^{k} \left(\sum_{i=1}^{n} x_{ij}^t - \sum_{i'=1}^{c_j} y_{i'j}^t \right)^2, \tag{10}$$

$$P(\mathbf{x}^t) := \sum_{j=1}^{k} \mathbf{x}_j^{tT} L_t \mathbf{x}_j^t, \tag{11}$$

where $\mathbf{x}_j^t \in \{0,1\}^n$ is the binary solution vector of a circuit slice t belonging to a core j and the superscript T in $\mathbf{x}^{tT}$ denotes the transposed version of the vector $\mathbf{x}^t$. A minimum value of S ensures that a qubit i is assigned to exactly one core, R penalizes an assignment that exceeds the core's capacity with c_j additional binary slack variables $\mathbf{y}_j^t = \{0,1\}^{c_j}$ for each core j, and lastly, P serves to penalize any cut edge between cores using the graph Laplacian L_t of circuit slice t. Given an undirected interaction graph $G_t = (V, E_t)$, the graph Laplacian L_t is the $n \times n$ matrix

$$L_t = D_t - A_t \tag{12}$$

where D_t is the degree matrix and A_t is the adjacency matrix of the graph G_t [48].

Lemma 1. *If $\mathbf{x}_{opt}^t$ is a binary solution vector for which $F(\mathbf{x}_{opt}^t) = 0$ then $\mathbf{x}_{opt}^t$ represents an assignment of all logical qubits with the following properties: each qubit is mapped to exactly one core j; each adjacent pair of qubits in the graph $G_t(V, E_t)$ is assigned to the same core; and $\forall j \in \{1, \ldots, k\}$ no more than c_j qubits are assigned to core j. We refer to $\mathbf{x}_{opt}^t$ as a valid assignment.*

Proof. Since S only consists of sums comprising quadratic terms with a minimum value of zero, S is zero, iff for each $i \in V$ exactly one variable $\{x_{ij}^t | 1 \leq j \leq k\}$ has value 1. We call this property *s*.

R is zero iff for each $1 \leq j \leq k$

$$\sum_{i=0}^{n} x_{ij}^t = \sum_{i'=0}^{c_j} x_{ij}^t.$$

This can only hold, if c_j or fewer variables of $\{x_{ij}^t | i \in V\}$ have a value of 1. Otherwise $\sum_{i=0}^{n} x_{ij}^t > c_j$ and R exhibits a value greater. We call this property *r*.

P is zero iff for each $1 \leq j \leq k$

$$\mathbf{x}_j^T L \mathbf{x}_j = \sum_{i=1}^{n} d_i x_{ij}^2 - \sum_{i=1}^{n} \sum_{v \in V_i} x_{ij} x_{vj} = 0.$$

where V_i denotes the set of adjacent vertices of vertex i. The above can only hold if for each i, $\sum_{v \in V_i} x_{ij} x_{vj} = |V_i| = d_i$. Otherwise, if there is less than d_j adjacent vertices assigned to core j, $\sum_{v \in V_i} x_{ij} x_{vj} < |V_i|$ and $\mathbf{x}_j^T L \mathbf{x}_j$ exhibits a penalization value greater zero. We term this property p, where we left out the superscript t for better readability.

Together, properties *s, r* and *p* are equivalent with $\mathbf{x}^t$ encompassing a solution where each qubit $i \in V$ is mapped to exactly one core j, where less or equal to c_j qubits are assigned to a core j and where each adjacent pair of qubits in the graph $G_t(V, E_t)$ is assigned to the same core. The said properties require S, R, P to be zero implying the same for F. Thus if F is zero the properties *s, r* and *p* hold. □

The final step for the qubit assignment part is to generalize the objective function for the qubit assignment $F(\mathbf{x}^t)$ in Eq. (8) to all circuit slices T, i.e., $\mathbf{x} := [\mathbf{x}^1, \ldots, \mathbf{x}^T]$, which we term H_a

$$\min_{\mathbf{x}} H_a(\mathbf{x}) = \min_{\mathbf{x}} \sum_{t=1}^{T} F(\mathbf{x}^t) \tag{13}$$

resulting in a solution to the qubit *assignment* problem for all circuit slices.

Proof. Since H_a only consists of functions F with a minimum value of zero, H_a is zero iff F is equal to zero for all circuit slices t. This is equivalent to the property that $\mathbf{x}$ presents a valid assignment for all circuit slices. □

b) Minimizing inter-core communication: As discussed in Ex. 1, the other task of the objective function is to minimize potential inter-core communication between every pair of assignments (F_{t-1}, F_t).

Let us consider an arbitrary qubit i in two subsequent times slices: $x_{ij}^{t-1} = x_{il}^t = 1$. If $j \neq l$, then i is assigned to core j in circuit slice $t-1$ and to a different core l in circuit slice t, and thus a state transfer is necessary between cores (j, l). Hence, we can define the following penalization objective function for inter-core communication:

$$\min_{\mathbf{x}} H_t(\mathbf{x}) = \min_{\mathbf{x}} \sum_{t=2}^{T} d(\mathbf{x}^{t-1}, \mathbf{x}^t) \tag{14}$$

where

$$d(\mathbf{x}^{t-1}, \mathbf{x}^t) := \sum_{i=1}^{n} d_{jl} x_{ij}^{t-1} x_{il}^t \tag{15}$$

where d_{jl} is the hop count between cores (j, l), i.e. the number of links a state traverses in order to travel from core j to l. E.g., in the simplest case, each core is connected to all other cores (all-to-all) via a link over which quantum state teleportation can take place, then $d_{jl} = 1 \ \forall (j, l)$ where $i \neq j$ and trivially $d_{jj} = 0$. In order to count the number of inter-core communications M, we can readily use Eq. (14):

Example 3 (Count inter-core communications)**.** *Given the assignments of Ex. 1, qubit 4 is the only qubit that requires a state transfer between circuit slices 1-2. Using Eq. (15) yields*

$$\begin{aligned} d(\mathbf{x}^1, \mathbf{x}^2) = & \underbrace{d_{11}}_{0} x_{01}^1 x_{01}^2 + \underbrace{d_{11}}_{0} x_{11}^1 x_{11}^2 + \underbrace{d_{22}}_{0} x_{22}^1 x_{22}^2 \\ & + \underbrace{d_{11}}_{0} x_{31}^1 x_{31}^2 + \underbrace{d_{12}}_{1} x_{41}^1 x_{42}^2 + \underbrace{d_{22}}_{0} x_{52}^1 x_{52}^2 \\ & = 1 \end{aligned}$$

as expected.

Combining objectives (13) and (14) gives the final form of the quadratic objective function. The QUBO model for quantum circuit mapping to multi-core quantum devices thus reads

$$\min_{\mathbf{x}} H(\mathbf{x}, \lambda) = \min_{\mathbf{x}} \left[H_a(\mathbf{x}) + \lambda \cdot H_t(\mathbf{x}) \right] \tag{16}$$

where $\lambda \geq 0$ is a weighting parameter to scale H_t.

Remark. *The proof of H_a shows that if $H_a > 0$ at least one assignment is not valid and the circuit cannot be executed. Let us assume, for example, for a solution $\mathbf{x}_1$, the objective function results in $H_a = 1$ and $H_t = 1$. For another solution $\mathbf{x}_2$, the sums result in $H_a = 0$ and $H_t = 3$. In total the objective functions for $\mathbf{x}_1$ and $\mathbf{x}_2$ yield 2 and 3, respectively. Even though the solution $\mathbf{x}_1$ yields a wrong assignment, i.e., $H_a = 1$ (with one inter-core communication $H_t = 1$) and $\mathbf{x}_2$ yields a correct assignment (with 3 inter-core communications $H_t = 3$), the total objective function would decide in the favor of $\mathbf{x}_1$, which is undesirable.*

We therefore choose λ, such that $0 \leq \lambda \cdot H_t \lesssim 1$. with

$$\lambda \lesssim (Tn)^{-1} \tag{17}$$

hence

$$\max_{\mathbf{x}^t} H_t = \max_{\mathbf{x}^t} \sum_{t=2}^{T} d(\mathbf{x}^{t-1}, \mathbf{x}^t) < T \cdot n$$

in the all-to-all case, which favors correct assignments with a higher likelihood. Further, we calculate $H_a(\mathbf{x})$ in order to verify the correctness of $\mathbf{x}$, i.e., we only consider a solution to be valid, if $H_a(\mathbf{x}) = 0$.

C. Toy-model example

In order to gain an intuitive understanding of how the weighting parameter λ impacts the mapping solution, we show the outcome of the mapping of a simple circuit (Fig. 2) to a three-core quantum system, each core with a capacity of two physical qubits. We assume the cores are all-to-all connected, i.e., $d_{ij} = 1\ \forall ij$ where $i \neq j$ and trivially $d_{jj} = 0$. The overview of the given parameters is listed in Tab. II.

TABLE II: Parameter overview of toy-model example

Parameter	Value
n (# qubits)	5
T (# circuit slices)	5
λ (penalization parameter)	$0.001, 0.1$
k (# cores)	3
$c_j = c\ \forall j$ (capacity)	2

We can determine the size of the solution array $N = T \cdot k \cdot n = 5 \cdot 3 \cdot 5 = 75$, i.e., $\mathbf{x} = \{0,1\}^{75}$. Further, the problem requires $T \cdot c \cdot k = 5 \cdot 2 \cdot 3 = 30$ slack variables,

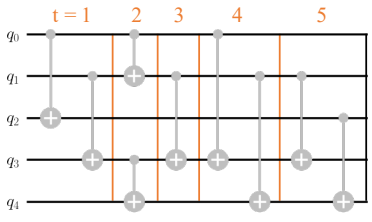

Fig. 2. Toy-model quantum circuit consisting of nine two-qubit gates in five circuit slices $t \in [1, 5]$ labeled in orange.

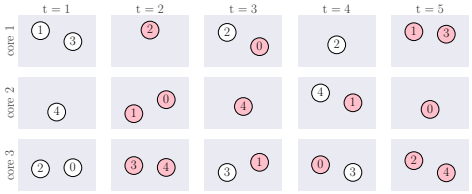

Fig. 3. $\lambda_1 = 0.001$ mapping solution of the QUBO model, where the x-axis is the timeline and the y-axis marks the different cores. Red-colored nodes signify that these qubit states have been transferred from the previous slice. This mapping hence resulted in 15 inter-core communications.

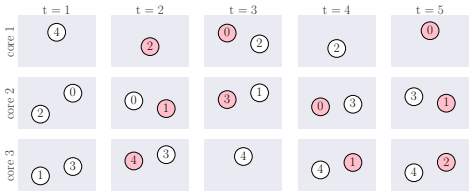

Fig. 4. $\lambda_2 = 0.1$ mapping solution of the QUBO model, where the x-axis is the timeline and the y-axis marks the different cores. Red-colored nodes signify that these qubit states have been transferred from the previous slice. This mapping hence resulted in 10 inter-core communications.

i.e., $\mathbf{y} = \{0,1\}^{30}$, yielding 105 variables in total. We can assemble the graph Laplacians. Considering the interaction graph $G_1(V, E_1)$, L^1 reads

$$L^1 = \begin{pmatrix} 1 & 0 & -1 & 0 & \\ 0 & 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

It is a simple exercise to determine the remaining Graph Laplacians, which is left to the reader. Given all the constants we can assemble the QUBO problem (see Eq. 16), which we then solve with a simulated annealing heuristic for two different λ settings, $\lambda_1 = 0.001$ and $\lambda_2 = 0.1$.

Both mapping solutions are displayed in Figs. 3 and 4 are valid mappings, since all two qubits are placed such that they can conduct their two-qubit gates (Fig. 2). The mapping with λ_1 resulted in 12 inter-core communications, while the model using λ_2 resulted in 8 inter-core communications. As expected, but still significant, a higher λ parameter yielded 33% improvement in the number of inter-core communications. Note that the weighting of λ is restricted by the validity of an assignment (Eq. 17); meaning, even though the optimal (highest) λ, which still gives a valid assignment ($H_a = 0$), is *not* a priori known, we do know that a weight set too high can jeopardize the feasibility of the solution, which is why the method must verify $H_a = 0$ to guarantee a valid assignment.

D. Time Slices of a Quantum Circuit

Solving the quantum circuit mapping problem for multi-core systems with the introduced formulation requires a quantum circuit to be partitioned into smaller blocks of gate sequences, where each so-called circuit slice t is represented by an interaction graph $G_t(V, E_t)$. The objective is to generate

graphs, where each can represent as many gates of the circuit as possible, provided that all gates in a slice can be executed without moving qubits between cores. To do so, we use a recursive procedure, outlined in Alg. 1.

Given a list of gates Ψ of a quantum circuit and a list of empty graph objects stored in the list *slices*, we iterate over all gates and apply the function ADDGATE in each step. The graph objects are initially empty and are filled over the course of the iterations, with every iteration variable t is updated accordingly, marking the number of non-empty graph objects in *slices*, i.e., the number of graph objects, which contain at least one edge. ADDGATE then decides to which graph in the list the gate is assigned. Let us consider some arbitrary gate $\Psi_l = (i_1, i_2)$ involving qubits $i_1, i_2 \in V$, one of the following four cases is about to happen:

(1) the function returns immediately if the gate is already present in the current slice t;
(2) the gate needs to be assigned to the subsequent slice $t+1$ if either or both of the qubits i_1, i_2 are involved in the current slice in least one other gate. In other words a gate (i_1, i_2) cannot be added to a slice, if another gate $(i_1, .)$ or $(i_2, .)$ is already present, as then the gates are not concurrently executable;
(3) the gate is simply added to current circuit slice t, if it is the first slice $t = 0$ and 1) and 2) are False, i.e., if the gate is not present and none of the qubits has been used;
(4) the function is called again using the previous slice $t-1$, if the current slice is not the first, i.e., $t > 0$ and 1) and 2) are False.

Algorithm 1: Slicing the quantum circuit

```
Input Ψ, slices          ▷ list of quantum circuit gates, list of empty graphs
Output slices                                  ▷ list of interaction graphs G_t

function ADDGATE(quantum circuit gate, t)              ▷ defines cases (1)-(4)
    if gate in slices[t] then
        return;
    else if any(gate_i used in slices[t]) then
        t ← t + 1
        slices[t] ← gate
        return;
    else if t is 0 then
        slices[t]← gate
        return;
    else
        return ADDGATE(gate, t − 1);
    end
end function

global slices                                  ▷ list of empty graph objects

for gate in Ψ do
                                          ▷ main iteration, conducts slicing
    t ← N.o. non-empty graphs in slices
    ADDGATE(gate, t)
    end
```

IV. Experimental setup

All tests were run on an Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz with 256 GB of RAM and 16 cores running Debian GNU/Linux 11. The method is implemented with Python 3.9.2 using Qiskit's framework [49] to generate the benchmark programs and to (pre-)process the circuits. We used dwave-neal version 0.5.9 [13] simulated annealer [50] as our dedicated solver heuristic, as it comes with the open source package of D-Wave Systems and can compete with commercial QUBO solvers [45]. It is worth noting that the selected solver can be easily replaced with any other solving method for quadratic programming. Due to the memory allocation limits of our system, we implemented a divide-and-conquer approach to divide and solve more amenable subsets of the problem before adding them back together. Considering the largest 50% of our problem sizes, up to four divisions are necessary (where one division was sufficient for 62% of them). Divide-and-conquer allows us to solve arbitrarily large QUBO instances, with the trade-off of adding some locality to the search process. Since 50,000 variables span about 80 time slices on average in our benchmarks, the added locality is still far from what is typically used for a local estimate.

A. Benchmarks

Similar to previous works [8, 26], we use as benchmarks:

- Subroutines which are building blocks of larger circuits (e.g., Shor's algorithm) such as the Quantum Fourier Transform (QFT), the Multi-Target Gate, Draper's QFT Adder [51] and Cuccaro's Ripple-Carry Adder [52] (both in their *fixed* version) utilized as Qiskit circuit implementations[4].
- Four instances of randomly generated circuits to fill the gap in structured circuits in terms of the following parameters: i) the number of gates and qubits; ii) circuit depth; and iii) circuit density, that is the average number of two-qubit gates divided by the circuit depth. In this last category, we also include Quantum Volume [53], which is a circuit having equal values for width and depth used to evaluate the overall performance of a quantum computer. Note that quantum volume circuits are by definition challenging to map due to their high gate density.[53].

We generated a total of 407 benchmark instances for these nine circuits, which parameters are shown in Fig. 5. Note that they cover qubit counts ranging from 50 to 100 qubits, circuit depths from 13 to more than 1,100, and gate densities ranging from 0.67 to 21.42. Our QUBO-based mapping framework as well as benchmark instances are available in an open-source format and can be accessed via github. We employed Qiskit-based routines in order to ensure reproducibility by the scientific community.

It is important to note that we observed rather large differences between the circuit instances we used and the ones in [8]. For instance, the QFT adder of the same number of qubits in [8] has significantly lower circuit depth, and random circuits of the same number of qubits and depth have a much higher number of gates. Therefore, our benchmark instances are similar to the benchmarks from [8] in terms of functionality and structure, but the results are not directly comparable.

[4]https://qiskit.org/documentation/apidoc/circuit_library.html#module-qiskit.circuit.library, 18.04.2023

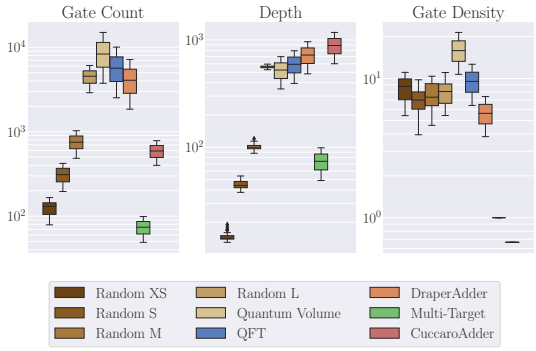

Fig. 5. Range of relevant circuit parameters of our benchmark set. Circuits between 50 to 100 qubits have been generated using Qiskit's circuit library and decomposed into CNOT gates. Random XS, S, M and L refer to the different depth intervals 13-19, 38-54, 88-120, 529-596, respectively. Note that all random circuits have an average circuit density of $\sim$ 10 two-qubit gates relative to depth, whereas Multi-Target Gate and Cuccaro Adder show a much lower density, 1 and 0.5 respectively.

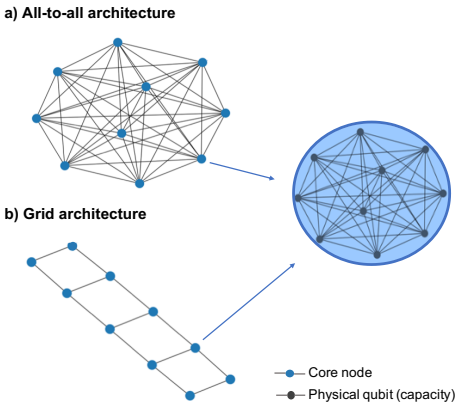

Fig. 6. Two different multi-core architectures considered in our experiments: **a)** All-to-all connected cores and **b)** 2D Grid core connectivity. Each node in the two graphs represents a core and the edges correspond to communication links between the cores. On the right, the intra-core qubit topology is shown, consisting of 10 all-to-all connected qubits.

B. Multi-core quantum computing architecture

We consider two multi-core architecture topologies comprising 10 cores, each of them consisting of 10 all-to-all connected qubits. We assume all-to-all qubit connectivity within the cores since intra-core communication is negligible compared to inter-core communication. The first topology (Fig. 6a) is similar to the one considered in [8], with all-to-all core connectivity. The second topology (Fig. 6b) showcases more realistic inter-core connectivity for future developments in modular quantum computing as shown in [4]. In order to have the same number of cores in both topologies, we choose a 2×5 grid layout. Our method is, however, easily adaptable to include more complex architectures as well, which would include intra-core qubit routing like already existing modular architectures shown in [4]. In addition, note that our method is not restricted to any particular size of the architecture.

C. Performance metrics

Our main performance metric is the number of inter-core communications M between cores, also termed non-local communications, required for executing a circuit on the quantum multi-core system. They are calculated using H_t as previously shown in Ex. 3. In addition, we provide the execution time for all mapping attempts.

V. Results and Discussion

A. Number of inter-core communications

In this section, we analyze and discuss the results of our experiments, in which we tested our method with the circuits as described above on the two different multi-core architectures depicted in Fig. 6. As we just mentioned, we use as performance metrics the number of inter-core communications and execution time and analyze how they relate to the structure of the circuits, in particular circuit density, and how they scale with a larger number of qubits, gates and depth.

1) Layout all-to-all: Fig. 7 depicts the results of all benchmarks in this study. We ran 50-100 qubit-sized circuits for each benchmark circuit (except DraperAdder and CuccaroAdder, which exhibit only even or odd numbers of qubits in their fixed version). We obtained a success rate of 87%, which means a valid assignment could be found in the first attempt for 354 out of 407 circuits.

Fig. 7.**a)** shows the number of inter-core communications for all benchmark circuits. The best performers are clearly Random XS, Random S, Random M and Multi-Target, which stayed below 5,000 inter-core communications. This is not surprising, as they cover the lower range of gate count and depth (Fig. 5), i.e., smaller circuits require smaller numbers of inter-core communications. On the higher end of the performance measure lie the adders, QFT, Random L and Quantum Volume, which yielded inter-core communications between 2,000 and 17,000. The trend of circuits with more gates and higher depth resulting in higher inter-core qubit communications is quite clear. However, the QFT and Quantum Volume benchmarks displayed an opposite tendency, as the Quantum Volume benchmark resulted in 2,800 less inter-core communication on average compared to QFT, even though Quantum Volume has about $2,000$ more two-qubit gates, 1.7 times higher density and similar depth.

When we look at the inter-core qubit communications relative to the number of two-qubit gates in a circuit 7.**b)**, the ranking of benchmarks changes significantly. In this picture, the benchmark Multi-Target, though presenting one of the smaller absolute outcomes, yields up to five inter-core communications per two-qubit gate, similar to our largest benchmark, the CuccaroAdder. The structure of these circuits can serve as an explanation, as both of them are hardly parallelizable: a circuit layer comprises only at most one two-qubit gate. This circuit structure implies circuit slices where $|E_t|$ is small and qubit assignments between circuit slices are almost identical. This is a disadvantage for the proposed method, as it performs better when a significant number of changes is

required between circuit slices(which is typically the case for dense, i.e., highly parallelized circuits): considering the 50% of the circuits with the highest density, the relative inter-core communications stay below 0.8 on average (less than one communication per two-qubit gate), while the other half of the circuits exhibit 19 inter-core communications per gate on average. The Quantum Volume benchmark accentuates these findings further: with 8,000 two-qubit gates it has by far the highest gate count and highest parallelization, and it is the second-best performer in the relative picture 7.**b)** with one inter-core communication every second two-qubit gate. 7.**c)**. depicts the weighting factors λ (17), chosen between 0.7 and 0.006. As the term H_t increases proportionally to the circuit size, we choose λ accordingly to achieve $\lambda H_t \lesssim 1$, we can see this trend clearly. It can be observed, that on the selected value, the number of inter-core communications increases, which is the expected behavior. For the sake of a high success rate and efficiency in producing the first mapping results, we determined a conservative selection of these weighting parameters.

2) 2D-Grid layout: In this experiment we tested the method for qubit sizes $n \in [50, 75, 100]$ on a 2×5 grid quantum architecture layout as depicted in Fig. 6**b)**. Tab. III lists the results, showing a similar dependency on the problem size in terms of inter-core communications of the circuit to the all-to-all layout experiment. The number of inter-core communications is however higher than the results of the first experiment due to longer hop distances between cores; for all 50-qubit circuits, for example, the number of inter-core communications is about twice as high compared to the all-to-all layout. With both layouts, we can detect better performance values for random circuits. Our selected quantum circuits were characterized in terms of standard parameters, such as the number of qubits and depth. However, there are other parameters, such as the percentage of two-qubit gates, and interaction graph characteristics, which were shown to have correlations to what can be expected from a mapping outcome [54]. Since random and real circuits can differ significantly in their inherent circuit structure, the latter is expected to have an impact on the performance as well.

B. Execution Time

Since execution time depends on the problem size $N = T \cdot n \cdot k$ (i.e., the circuit size and multi-core system size), but not on the factors in the core-distance matrix d, distinguishing between the two architecture-layout experiments is not required. Fig. 8 shows the execution runtimes which resulted in values between 0.1 and 138.36 minutes, where 3/4 of the circuits were successfully compiled in 30 minutes or less. Although these execution times are high for a compiler method, we must highlight that this method is classical by nature. One can therefore justify usage of our compilation strategy, as savings of even a few milliseconds are consequential in the success probability of a quantum algorithm [55], especially in the case of inter-core communication [56]. Furthermore, for ease of reproducibility and comparability, there were no HPC

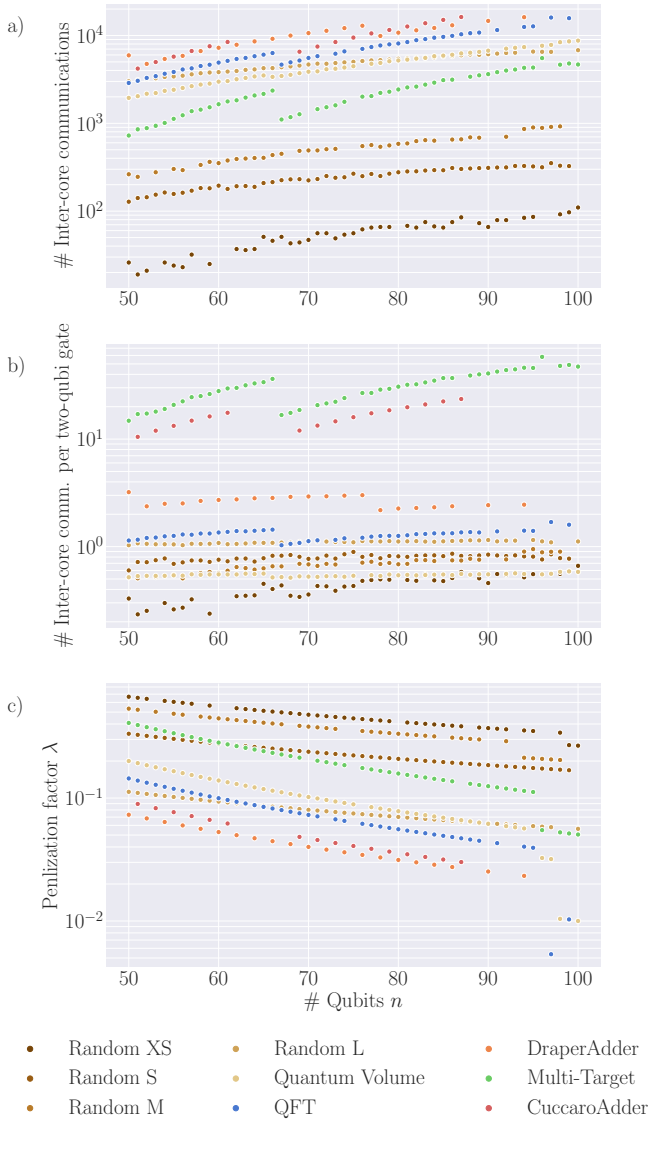

Fig. 7. Results for all benchmarks run on an all-to-all multi-core layout. **a)** and **b)** summarize the inter-core communications in absolute and relative values to two-qubit gates, respectively, for successful mappings between 50 and 100 qubits. The tendency for higher amounts of inter-core communication for larger circuits is clearly detectable. However, from a relative to two-qubit gate point of view, the circuits with high density yield lower outcomes, i.e., random circuits and QFT. **c)** depicts the penalization factor λ chosen for each run.

resources or similar performance-improving tools employed, which are typically accessible for commercial use.

VI. Conclusion and Future Work

Multi-core quantum computing architectures are one of the most promising approaches towards large-scale quantum computers, for which it is required to develop new quantum circuit mapping techniques that consider the inter-core communication requirements.

In this study, we proposed a novel approach for mapping quantum circuits to multi-core quantum architectures based

TABLE III: Results for selection of benchmarks on a 2×5-grid architecture.

Benchmark	# Qubits	Depth	Two-qubit gate count	λ	# Number of inter-core comms.
Quantum Volume	50	350	3750	0.1020	3036
	75	150	8325	0.0450	7646
	100	700	15000	0.0100	17940
Multi-Target	50	49	49	0.2041	1261
	75	74	74	0.0541	5097
	100	99	99	0.0101	13218
Random S	50	28	171	0.2333	182
	75	26	253	0.2222	273
	100	30	344	0.1000	540
Random M	50	65	412	0.1333	723
	75	61	647	0.1778	923
	100	65	867	0.0667	1888
Random L	50	350	2474	0.0225	7308
	75	357	3730	0.0150	11789
	100	344	5004	0.0112	15778
CuccaroAdder	49	577	384	0.0389	7450
	75	889	592	0.0163	19827
	99	1177	784	0.0046	45275
QFT	50	394	2525	0.0816	3947
	75	594	5661	0.0450	7443
	100	794	10050	0.0025	36800
DraperAdder	50	484	1850	0.0610	7769
	76	744	4294	0.0287	16675
	100	984	7450	0.0017	54384

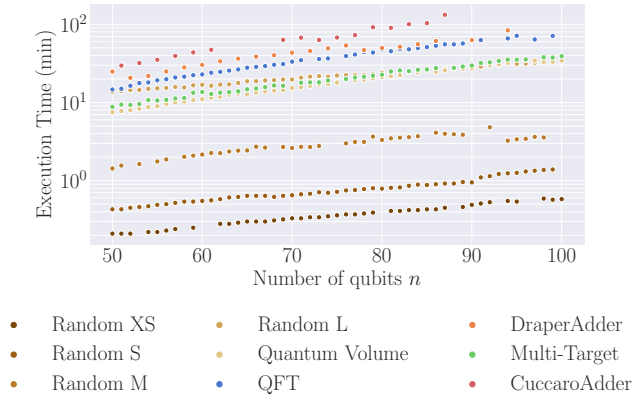

Fig. 8. Execution times of the QUBO method, which resulted in a valid mapping solution. The results span a range from the smallest circuit (50-qubits Random XS) of about 12 seconds up to over 138 min for the largest instance (100-qubits CuccaroAdder).

on QUBO. The main strengths of our method lie in the formulation of the QUBO itself, as the structure permits i) the decoupling of the problem definition from the solver, as well as ii) superseding limitations of look-ahead approaches used in previous solutions. We tested the method's functionality for a wide range of benchmarks on two different multi-core architecture layouts composed of 10 cores with a capacity of 10 qubits each. Taking stock of the analysis of our benchmark experiments, we expect a success rate of 87% to find a solution, where the most promising results could be achieved with circuits exhibiting high density and shallow circuits [57]. Note that highly-parallelized circuits are usually the most challenging to run on quantum devices and are therefore often used as benchmarks to test them, which makes them most relevant to quantum computing of the near- to mid-term future [58]. Furthermore, our method is easily adjustable to different architecture layouts as these changes require only altering the factors in the objective function. As a direct and positive consequence, these adjustments do not affect the execution time of the method.

Our method, on the other hand, faces some challenges with the scalability of quantum circuits in terms of circuit depth. In order to improve our approach, including its scalability capabilities, in our future work we will be focused on:

- Finding a suited weighting parameter λ, which is, in general, a non-trivial task and is a matter of ongoing research [59]. Based on these methods we can employ an existing technique that fits our problem instances, which will be our first solution for improving the circuit success rate.
- Differentiating between remote two-qubit gates (operations between two separate qubits in different cores) and qubit state transfers. This will help to better optimize the amount of inter-core communication, by doing the qubit transfers only when necessary, and also make more realistic multi-core architecture when combined with investigating other inter-core communication links [25, 27].
- Solving the scalability limitations related to circuit depth by decreasing the number of decision variables via an alternative problem formulation to a gate assignment version instead of the qubit assignment. This alternative problem approach in its prototype state is already in the testing phase.
- Improving the runtime performance by parallelizing the algorithm execution (the algorithm is carried out sequentially at the moment) and by using the commercial QUBO problem solvers with superior computational power. The latter will also help to handle larger problem sizes (up to 10^6 variables) [45].

Besides the described areas of improvement, an interesting direction for further experiments will be different layouts in both multi-core architecture and intra-core couplings, which will include topologies that are already proposed in [4]. In addition to that we plan to perform an in-depth scalability analysis regarding the number of cores, physical and logical qubits as well as circuit depth, to find the actual limits of our algorithm as in [28].

In summary, we expect that the introduced method exhibits potential for future qubit mapping tasks. It already demonstrates promising results for especially challenging mapping tasks to multi-core quantum architectures and exhibits the capability to be easily adjusted to both problem size and multi-core layout.

Acknowledgments

MB and SF would like to acknowledge funding from Intel Corporation. EA and CGA acknowledge support from the EU, grant HORIZON-EIC-2022-PATHFINDEROPEN-01-101099697 (QUADRATURE). SA acknowledges support from the EU, grant HORIZON-ERC-2021-101042080 (WINC).

References

[1] M. Sarovar, T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Detecting crosstalk errors in quantum information processors," *Quantum*, vol. 4, p. 321, 2020.

[2] L. Lao, H. van Someren, I. Ashraf, and C. G. Almudever, "Timing and resource-aware mapping of quantum circuits to superconducting processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[3] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, "Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects," *Physical Review A*, vol. 89, no. 2, feb 2014.

[4] N. LaRacuente, K. Smith, P. Imany, K. Silverman, and F. Chong, "Modeling short-range microwave networks to scale superconducting quantum computation," *Preprint at arXiv https://arxiv. org/abs/2201.08825 v2*, 2023.

[5] H. Jnane, B. Undseth, Z. Cai, S. C. Benjamin, and B. Koczor, "Multicore quantum computing," *arXiv preprint arXiv:2201.08861*, 2022.

[6] K. N. Smith, G. S. Ravi, J. M. Baker, and F. T. Chong, "Scaling superconducting quantum computers with chiplet architectures," 2022.

[7] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," in *Proceedings of the ASPLOS'19*, 2019, pp. 1001–1014.

[8] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Time-sliced quantum circuit partitioning for modular architectures," in *Proceedings of Computing Frontiers '20*, 2020, pp. 98–107.

[9] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. Pereira, "Qubit allocation," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, ser. CGO 2018, 2018, p. 113–125.

[10] M. Bandic, H. Zarein, E. Alarcon, and C. G. Almudever, "On structured design space exploration for mapping of quantum algorithms," in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, 2020.

[11] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi, "Compiler design for distributed quantum computing," *arXiv preprint arXiv:2012.09680*, 2020.

[12] A. P. Punnen, *The Quadratic Unconstrained Binary Optimization Problem*. Springer, 2022.

[13] "D-Wave Systems | The Practical Quantum Computing Company." [Online]. Available: https://www.dwavesys.com/

[14] S. Chopra and M. R. Rao, "The partition problem," *Mathematical programming*, vol. 59, no. 1-3, pp. 87–115, 1993.

[15] H. Ushijima-Mwesigwa, C. F. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the d-wave system," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, 2017, pp. 22–29.

[16] M. Bandic, S. Feld, and C. G. Almudever, "Full-stack quantum computing systems in the nisq era: algorithm-driven and hardware-aware compilation techniques," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1–6.

[17] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[18] R. Wille, O. Keszocze, M. Walter, P. Rohrs, A. Chattopadhyay, and R. Drechsler, "Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits," in *Asia and South Pacific Design Automation Conference*, 2016, pp. 292–297.

[19] T. Itoko, R. Raymond, T. Imamichi, and A. Matsuo, "Optimization of quantum circuit mapping using gate transformation and commutation," *Integration*, vol. 70, pp. 43–50, 2020.

[20] M. G. Pozzi, S. J. Herbert, A. Sengupta, and R. D. Mullins, "Using reinforcement learning to perform qubit routing in quantum compilers," *arXiv preprint arXiv:2007.15957*, 2020.

[21] H. Jiang, Y. Deng, and M. Xu, "Quantum circuit transformation based on subgraph isomorphism and tabu search," *arXiv preprint arXiv:2104.05214*, 2021.

[22] S. Li, X. Zhou, and Y. Feng, "Qubit mapping based on subgraph isomorphism and filtered depth-limited search," *IEEE Transactions on Computers*, 2020.

[23] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 987–999.

[24] M. A. Steinberg, S. Feld, C. G. Almudever, M. Marthaler, and J.-M. Reiner, "Topological-Graph Dependencies and Scaling Properties of a Heuristic Qubit-Assignment Algorithm," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–14, 2022, conference Name: IEEE Transactions on Quantum Engineering.

[25] D. Cuomo, M. Caleffi, K. Krsulich, F. Tramonto, G. Agliardi, E. Prati, and A. S. Cacciapuoti, "Optimized compiler for distributed quantum computing," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, 2023.

[26] S. Rodrigo, M. Bandic, S. Abadal, H. van Someren, E. Alarcón, and C. G. Almudéver, "Scaling of multicore quantum architectures: A communications-aware structured gap analysis," in *Proceedings of Computing Frontiers '21*, 2021, p. 144–151.

[27] S. Rodrigo, D. Spanò, M. Bandic, S. Abadal, H. Van Someren, A. Ovide, S. Feld, C. G. Almudéver, and E. Alarcón, "Characterizing the spatio-temporal qubit traffic of a quantum intranet aiming at modular quantum computer architectures," in *Proceedings of the 9th ACM International Conference on Nanoscale Com-*

puting and Communication, 2022, pp. 1–7.

[28] A. Ovide, S. Rodrigo, M. Bandic, H. Van Someren, S. Feld, S. Abadal, E. Alarcon, and C. G. Almudever, "Mapping quantum algorithms to multi-core quantum computing architectures," *Proceedings of the ISCAS '23*, 2023.

[29] D. Cuomo, M. Caleffi, K. Krsulich, F. Tramonto, G. Agliardi, E. Prati, and A. S. Cacciapuoti, "Optimized compiler for distributed quantum computing," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, 2023.

[30] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t$|$ket$\rangle$: A Retargetable Compiler for NISQ Devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, Jan. 2021.

[31] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. a. Pereira, "Qubit allocation as a combination of subgraph isomorphism and token swapping," *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, oct 2019.

[32] S. Li, X. Zhou, and Y. Feng, "Qubit Mapping Based on Subgraph Isomorphism and Filtered Depth-Limited Search," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1777–1788, Nov. 2021.

[33] B. Dury and O. D. Matteo, "A qubo formulation for qubit allocation," *arXiv: Quantum Physics*, 2020.

[34] L. Prielinger, "A quadratic unconstrained binary optimization approach for qubit mapping," March 2023, unpublished, Master Thesis.

[35] P. L. Hammer and S. Rudeanu, "Pseudo-boolean programming," *Operations Research*, vol. 17, no. 2, pp. 233–261, 1969.

[36] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," *arXiv preprint arXiv:1811.11538*, 2018.

[37] A. Lucas, "Ising formulations of many np problems," *Front. Phys.*, vol. 2, p. 5, 2014.

[38] C. S. Calude, M. J. Dinneen, and R. Hua, "QUBO formulations for the graph isomorphism problem and related problems," *Theoretical Computer Science*, vol. 701, pp. 54–69, 2017.

[39] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, "Traffic flow optimization using a quantum annealer," *Frontiers in ICT*, vol. 4, 2017.

[40] P. Date, D. Arthur, and L. Pusey-Nazzaro, "QUBO formulations for training machine learning models," *Scientific Reports*, vol. 11, no. 1, p. 10029, May 2021.

[41] J. Nüßlein, C. Roch, T. Gabor, C. Linnhoff-Popien, and S. Feld, "Black box optimization using qubo and the cross entropy method," *arXiv preprint arXiv:2206.12510*, 2022.

[42] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: introduction and review," *Reports on Progress in Physics*, vol. 85, no. 10, p. 104001, Oct. 2022.

[43] W. van Dam, M. Mosca, and U. Vazirani, "How powerful is adiabatic quantum computation?" in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 279–287.

[44] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," *Journal of combinatorial optimization*, vol. 28, pp. 58–81, 2014.

[45] H. Oshiyama and M. Ohzeki, "Benchmark of quantum-inspired heuristic solvers for quadratic unconstrained binary optimization," *Scientific reports*, vol. 12, no. 1, pp. 1–10, 2022.

[46] I. Dunning, S. Gupta, and J. Silberholz, "What works best when? a systematic evaluation of heuristics for max-cut and qubo," *INFORMS Journal on Computing*, vol. 30, no. 3, pp. 608–624, 2018.

[47] B. A. Murtagh and M. A. Saunders, "Large-scale linearly constrained optimization," *Mathematical programming*, vol. 14, no. 1, pp. 41–72, 1978.

[48] E. W. Weisstein, "Laplacian Matrix," publisher: Wolfram Research, Inc.

[49] M. S. Anis *et al.*, "Qiskit: An open-source framework for quantum computing," 2021.

[50] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[51] T. G. Draper, "Addition on a quantum computer," *arXiv preprint quant-ph/0008033*, 2000.

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

[53] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Physical Review A*, vol. 100, no. 3, p. 032328, 2019.

[54] M. Bandić, C. G. Almudever, and S. Feld, "Interaction graph-based profiling of quantum benchmarks for improving quantum circuit mapping techniques," *arXiv preprint arXiv:2212.06640*, 2022.

[55] M. Y. Siraichi, V. F. d. Santos, S. Collange, and F. M. Q. Pereira, "Qubit allocation," in *International Symposium on Code Generation and Optimization*, 2018, pp. 113–125.

[56] P. C. Humphreys, N. Kalb, J. P. Morits, R. N. Schouten, R. F. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, no. 7709, pp. 268–273, 2018.

[57] R. Blume-Kohout and K. C. Young, "A volumetric framework for quantum computer benchmarks," *Quantum*, vol. 4, p. 362, 2020.

[58] A. Broadbent and E. Kashefi, "Parallelizing quantum circuits," *Theoretical computer science*, vol. 410, no. 26, pp. 2489–2510, 2009.

[59] A. Verma and M. Lewis, "Penalty and partitioning techniques to improve performance of qubo solvers," *Discrete Optimization*, vol. 44, p. 100594, 2022.