On Structured Design Space Exploration for Mapping of Quantum Algorithms

Bandic, Medina; Zarein, Hossein; Alarcon, Eduard; Almudever, Carmen G.

DOI
10.1109/DCIS51330.2020.9268670

Publication date
2020

Document Version
Final published version

Published in
2020 35th Conference on Design of Circuits and Integrated Systems, DCIS 2020

On Structured Design Space Exploration for Mapping of Quantum Algorithms

Medina Bandic*, Hossein Zarein†, Eduard Alarcon† and Carmen G. Almudever*
*Delft University of Technology, The Netherlands
†Technical University of Catalunya, BarcelonaTech, Spain

***Abstract*—Quantum algorithms can be expressed as quantum circuits when the circuit model of computation is adopted. Such a circuit description is usually hardware-agnostic, that is, it does not consider the limitations that the quantum hardware might have. In order to make quantum algorithms executable on quantum devices they need to comply to their constraints, which mainly affect the parallelism of quantum operations and the possible interactions between the qubits. The process of adapting a quantum circuit to meet the quantum chip restrictions is known as mapping. The resulting circuit usually has a higher number of gates and depth, decreasing the algorithm's reliability. Different mapping solutions have been already proposed. Most of them are meant for a specific quantum processor and differ in methodology, approach and features. In addition, they are usually only compared in terms of added gates, circuit depth and compilation time. No thorough comparative analysis of the different mapping solutions performance and features has been performed so far.**

In this paper, we propose to apply structured design space exploration (DSE) methodologies to the mapping procedures. This will allow not only to have a more in depth and structured analysis of their performance but also to identify what features are key and worth to implement. By using DSE we will be able to: i) determine in what regimes some mapping solutions outperform others; ii) derive optimal mapping strategies for specific quantum algorithms and quantum processors; and iii) perform an scalability analysis. In addition, DSE techniques cannot only be applied to the mapping layer that is key for bridging quantum applications to quantum devices, but also to the full-stack quantum computing system allowing for its cross-layer co-design.

***Index Terms*—Quantum circuit mapping, Design Space Exploration, quantum performance metrics, quantum benchmarks.**

I. Introduction

Quantum computing is one of the most active and promising research areas nowadays because it has the potential to solve problems which are intractable even for the most powerful classical supercomputers. Although current quantum processors, so-called Noisy Intermediate-Scale Quantum (NISQ) devices, are now capable of handling simple quantum algorithms, they are limited in size and by the presence of noise. In addition, they have several constraints that must be taken into account when executing quantum algorithms.

In most of current quantum processors qubits are arranged in a specific topology with limited connectivity, allowing only nearest-neighbour interactions among them. This is one of the main constraints of these devices, because in order for qubits to interact (perform a two-qubit gate), they should often be moved to become adjacent to each other. This means that quantum algorithms, usually described as hardware-agnostic quantum circuits, need to be adapted to respect the constraints of quantum processors. The procedure of modifying the quantum circuit to satisfy all hardware restrictions is called mapping (also known as routing, transpiling, compiling or synthesis).

978-1-7281-9132-4/20/$31.00 ©2020 European Union

The mapping problem is NP-complete and so far various solutions have been proposed to solve it [1]–[9]. They differ in approach, methodologies and metrics. What is common for most of them is that they take a bottom-up approach in which a mapper was developed for a specific quantum processor(s) and technology. In addition, the quality of the proposed mapping methods is usually assessed in terms of circuit depth and/or quantum gates overhead they result in and its compilation time. That is, the lower the added number of gates and/or the circuit depth overhead and compilation time are, the better the mapper is. However, those performance metrics do not provide any information on why or in what regime a given mapping solution is more beneficial. Therefore, we are still missing a more comprehensive comparison between those existing alternative mapping solutions.

In this paper, we propose to apply structured design space exploration (DSE) methodologies to further explore and perform a more thorough comparison of different mapping solutions and optimize them for specific quantum processors given a set of applications. More precisely, by using DSE techniques we will be able to identify what mapping features are worth it to implement, in what ranges a specific mapping method outperforms others or to optimize the mapper for a target processor and application. To this purpose, we focus on the following quantum system layers: quantum algorithms (benchmarks), mappers and quantum devices. It is worth noting that although in this work we focus on the application of DSE techniques to the mapping problem, this can be seen as the first step towards the development of a cross-layer co-design framework for full-stack quantum systems that will allow for a top-bottom, bottom-up optimization across layers. We envision that by using DSE techniques we can obtain performance trends and dimensioning design guidelines for current and next generations of quantum devices as well as for the full-stack quantum system when considering several architectural layers.

The paper is organized as follows. In section II, we will first review the basics of quantum computing. Afterwards, we will

provide an introduction and overview of the mapping problem and the different proposed solutions emphasizing what cost functions can be optimized in the mapping process. Section III will introduce the DSE methodology and explain how it can be applied to the mapping problem. Furthermore, we will give an overview of currently used performance metrics and benchmarks with focus on quantum volume. In section IV, we will show some preliminary results of our work so far related to applying DSE to the mapping problem. In Section V, the extension of DSE methodologies to other architectural layers and across the full-stack quantum system will be discussed. In Section VI, conclusions and future work are presented.

II. Running a quantum algorithm on a real quantum processor

A. Basics of Quantum Computing

In order to understand the mapping problem and what it is used for, we first need to briefly introduce the basics of quantum computing that include: qubits, quantum gates and quantum circuits. Contrary to classical computing, where we use bits to store information, the elementary information unit for quantum computing is the quantum bit or *qubit*. The difference is that rather than having only two possible states ('0' and '1' for bits and $|0\rangle$ and $|1\rangle$ for qubits), qubits can also be in superposition of both. More precisely, a quantum state (state of the qubit) can be described as: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. After measuring the qubit, the result will be a binary value 0 or 1 with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. Furthermore, its state will collapse to one of those binary outcomes, changing the state of the qubit to $|0\rangle$ or $|1\rangle$ and destroying the superposition.

To modify the state of a qubit, we can also apply other type of operation, a quantum gate. All quantum gates can be described as unitary matrices and can act on one or more qubits. The ones that are most common and supported by most of devices are single- and two-qubit gates. *Hadamard* gate, Pauli X, Y and Z are some examples of the gates that act on single qubits. A Hadamard gate for instance, takes the state of the qubit to superposition. On the other hand, *controlled NOT* (*CX, CNOT*) gate and *controlled Z* (*CZ, CPhase*) are examples of two-qubit gates. In two-qubit gates, one qubit always acts as *control* qubit and the other one acts as *target*. This means that the state of the target qubit after applying the gate depends on the state of the control qubit. For instance, in the CNOT gate and X gate is applied on the target qubit (open circle) if the control qubit (black dot) is in sate $|1\rangle$. The matrices and symbols of the aforementioned single- and two-qubit gates are shown in Figure 1 and 2, respectively.

One of the most relevant gates for the mapping problem is the so-called *SWAP* gate (Figure 2). As its name suggests, it exchanges the state of the two qubits it is applied on. This operation is used for routing the qubit states for some of the quantum technologies such as superconducting qubits as it will be described later. Note that a SWAP gate is equivalent to apply three CNOTs.

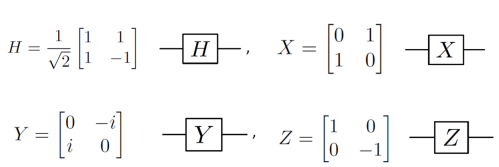


Fig. 1: Single-Qubit Gates.

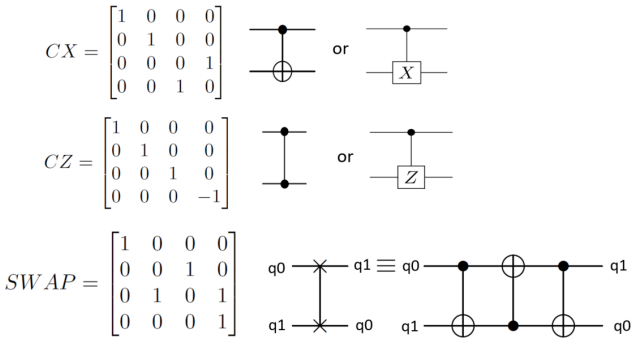


Fig. 2: Two-Qubit Gates.

Quantum algorithms can be represented by a sequence of quantum gates that are usually described in form of quantum circuit diagrams, as shown in Figure 3. This circuit example consists of 7 qubits (horizontal lines) in which 5 CNOT gates are applied. It is worth noting that current NISQ devices are error prone. Qubits are very fragile elements that easily lose their information (decohere) and gates are not perfect, showing error rates of $10^{-2} - 10^{-3}$. This poses a limit on the size of the algorithms, in terms of number of gates and circuit depth (number of time-steps), that can be successfully run on NISQ processors.

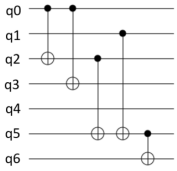


Fig. 3: Example of a quantum circuit.

B. The Mapping Problem

Quantum algorithms, described as quantum circuits, are quantum hardware-agnostic and therefore cannot be directly run on quantum devices. They need to be adapted to specific constraints of quantum chips in order to be executed. This is known as the mapping problem. Quantum hardware constraints may vary between different processor architectures, even within the same quantum technology. The constraint that has been mainly considered in previous works on mapping of quantum circuits and that is affecting this process the most is the (limited) *qubit connectivity*. This restriction is related to

the execution of two-qubit gates. For instance, for technologies like superconducting qubits and quantum dots, where qubits are arranged in a 2D topology with nearest-neighbor (NN) interactions, qubits need to be adjacent in order to interact - i.e to perform a two-qubit gate. One example of such architecture is shown in Figure 4. Another constraint that needs to be considered during the mapping process is the *primitive gate set*, which is a reduced set of gates specific for each quantum device. That means that the gates that compose a quantum circuit need to be decomposed into the ones supported by the quantum chip. In order to respect these quantum hardware constraints and transform a given quantum circuit to a version that is executable on the quantum device, a mapping procedure is required. It consists on the following steps (not necessarily in this order):

1) Decompose the gates of the circuit to the primitive gates supported by the quantum processor.
2) Map virtual qubits (qubits of circuit to be executed) to physical qubits (actual qubits on device). Also called initial placement of qubits or qubit allocation.
3) Schedule quantum operations so that all the dependencies between them (and other possible constraints coming from the use of shared classical control electronics [8]) are respected, while trying to minimize the circuit depth an therefore the execution time of the circuit.
4) Routing of qubits - moving qubits that need to interact so that they are adjacent, usually done by inserting SWAP operations. This process results in an increase of the number of operations as well as the circuit depth, decreasing the algorithm reliability or success rate. As we previously mentioned, qubits decohere and operations are faulty. Therefore, it is crucial that the overhead caused by the routing is minimal.

An illustrative example of the mapping process is shown in Figure 4 in which the circuit in Figure 3 is mapped to the Surface-7 quantum processor [10]. In this example, for sake of simplicity, we assume that all gates in the circuit are supported by the device and only the qubit connectivity restriction is considered. The first three CNOTs can be directly performed as the qubits are placed in adjacent positions. However, it is not possible to execute the fourth and fifth CNOTs because *q1* and *q5*, as well as *q5* and *q6* are not placed on near-neighbor qubits on the chip. We need then to insert a SWAP gate which exchanges the placement of virtual qubits *q3* and *q5* in the chip allowing to execute all quantum gates till the end of the circuit.

C. Prior Work

Many solutions have already been proposed in order to solve the mapping problem and some of them included in quantum compilers. The solutions differ in strategy, methodology and metric to minimize. Therefore, they can be classified based on different criteria:

- **Strategy:** We can categorize solutions in two ways. First categorization is optimal (exact) vs. heuristic solutions. Exact solutions in [5], [11], [12], use brute-force techniques that can obtain minimal results for small circuits. However, for more scalable solutions heuristics are needed [1], [13], [14]. Second, we can classify them as local (layer & permutation -based) [5] vs. global solutions (SWAP-based) [1]–[3]. Some of tools/methods used include Satisfiability Modulo Theory (SMT) solvers [2], [11], search and greedy heuristic algorithms [1], [5], [15], [16], MILP and MinLA solvers [8], [16] and machine-learning-based solutions [9], [17].
- **Metric(s) to be optimized (cost function)**: Most of works so far were focused on minimizing the number of gates [5]. That metric depends on the number of SWAPs added during the routing step. Second mostly used metric is circuit depth [8]. That metric represents the number of time-steps (layers) in the circuit. Each qubit can be engaged only in one gate per time-step. The less the number of time-steps, the shorter the circuit duration. Some works tried to combine these two in one cost-function, or check their trade-offs against each other, but no common conclusion about the matter has been made [1], [7]. Lastly, some works suggest the circuit reliability as a metric to optimize, where by choosing the most reliable path one can minimize the overall error rate [2].
- **Hardware constraint to focus on**: Most works focused on the qubit connectivity constraint and topology [11], [13]. However, other important restrictions that originate from the use of shared classical control electronics should be considered as proposed in [8], [18].
- **Additional differences**: Solutions also differ in some other aspects. For example, in using random or exact initial placement, whether they use look-ahead [1], [5], [13] or look-back schemes while scheduling operations or whether they are variation-aware [2], [3] or not (in terms of location on chip - different reliabilities of links between physical qubits and time and different coherence times of qubits).

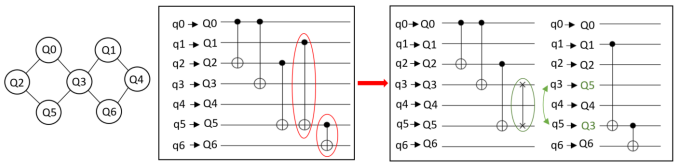


Fig. 4: Running a quantum circuit on the Surface-7 quantum processor. The most left diagram is the coupling graph of the Surface-7 chip in which each circle represents a physical qubit ($Q0$ to $Q6$) and the edges the connections (possible interactions) between them. Qi is the physical qubit label and qi represents the qubit in the circuit. In this case there is a q_i to Q_i mapping. An extra SWAP gate is required for being able to perform the last two CNOT gates.

In the majority of the mapping solutions mentioned in this section, a mapper was developed for a specific quantum processor in which different algorithms (used as a benchmarks) were mapped. In addition, all the proposed mapping methods so far are evaluated based on quantum gates overhead or/and

circuit depth they result in (raw data) and compilation time. However, those performance metrics are insufficient in case we want to have a more in depth and comprehensive comparison between solutions and determine why and in what regimes (e.g. for different number of qubits, different error rates) given mapping solutions are more favorable. As we will explain in the next section, we propose to use structured design exploration methodologies to have a more thorough comparison between different mapping solutions as well as optimize them for specific quantum processors and a class of applications.

III. Design Space Exploration Methodologies

A. What is DSE about?

Design space exploration is a structured design approach based on interdependencies of parameters, variables and metrics of some system, used for optimizing it. The goal of DSE is to improve some predefined performance metrics or a combination of them, by concurrently sweeping over a wide range of all input variables (multidimensional sweep). Therefore, for applying DSE methodologies to a given problem one needs to: i) define the design space. That is, the specific problem is described in terms of input variables and a range of values or design points that will be swept; ii) select the performance metrics that are described as functions depending on a set of different input variables; iii) choose a global cost function as a figure of merit which is composed of different performance metrics and allows to identify overall optimal points; and iv) model the interdependencies between performance metrics and input parameters. This can be done by using analytical models, computer-based simulation or experimental data that can be interpolated. Note that input variables can take continuous as well as discrete values or even to be knobs that can be turned on and off or define alternative choices. By using structured DSE methodologies one can:

- Identify design trends: through observation of the different performance metrics and figure of merit, design trends can be identified (e.g. optimal points, sweet spots, exponential or linear growth, valleys, saturation, etc.) by performing a qualitative analysis.
- Determine boundaries in the input design space: by projecting the performance metrics back to the input design space, different design areas of operations can be defined. For instance, in which parameter range a design option outperforms others. This is a quantitative analysis that provides dimensional design guidelines.
- Derive an optimal design of the system and related set of parameters.
- Perform a technology gap analysis (make predictions) by making future assumptions in the input design variables.

B. Applying DSE to the mapping problem

As mentioned already, so far most of the mapping solutions are quantum processor-specific and assessed just using single-number metrics such as quantum gate and/or circuit depth overhead and compilation time. The lower the mapping overhead and compilation time are, the better the mapper is. However, these metrics that are also used for comparing different mapping solutions do not provide any insights on performance-complexity trade-offs, or in what cases to use different mapping approaches.

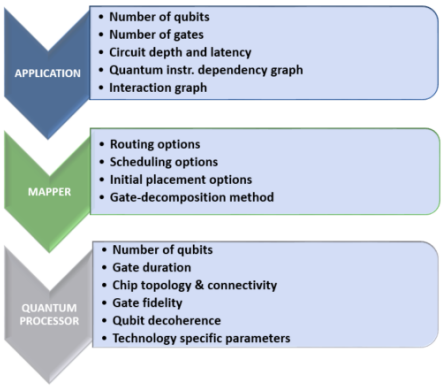


Fig. 5: DSE layers and Variables for mapping of quantum circuits.

The application of DSE methodologies will allow to perform a comprehensive and structured analysis of the different mapping procedures and help us to determine which features and optimization techniques are worth to implement, to identify in what input variable regimes a mapping approach outperforms others, to derive optimal mapping strategies for certain kind of applications and quantum processors and to perform a scalability analysis of them.

In order to achieve the outcomes stated above, we propose the layered approach shown in Figure 5 where we highlight the importance of defining proper open variables and performance metrics, and how they relate to each other. Possible input variables for each of the layers are the following.

Quantum Processor Layer: variables for this layer include the hardware constraints described in Section II such as qubit topology and connectivity and the primitive gate set. Other additional parameters are the number of qubits, gate duration (how long a gate takes to be executed), gate fidelity (how reliable the gate is), qubit decoherence (how long a qubit can hold its state), and technology-specific parameters (e.g. number of frequencies used for superconducting qubits, the trap size for trapped ion systems). These variables can therefore be used to generate different quantum processor designs whose properties (constraints) are provided to the mapper layer.

Mapper Layer: in this layer, the variables to be considered correspond to the different options and features that the several steps of the mapping process, which includes gate decomposition, placement of qubits in the physical quantum device, scheduling of operations and routing of qubits, have as described in Section II.C. For instance, routing choices include brute-force vs. heuristics, number of calculated routing paths, metric to optimise during the routing process, etc.

Application Layer: it consists of a set of quantum algorithms which can be described as hardware-agnostic quantum circuits. Currently, the quantum algorithms used as a bench-

marks for analysing the performance of the mappers are not representative as they are in most cases reversible circuits that will not provide any computational advantage compared to their classical counterparts. In addition, they are usually only profiled in terms of number of gates, circuit depth, percentage of two-qubit gates and number of interactions between qubits pairs (this latter used for deriving an optimal placement of qubits). By having a more in depth profiling of the quantum algorithms in which characteristics of the interaction graphs (i.e. how many times each pair of qubits interact and how those interactions are distributed among qubits and in time) and of the quantum instruction dependency graph (i.e. identifying clusters of operations) can be beneficial for obtaining optimal mapping solutions. These variables derived from the algorithm profiling will also be essential for developing application-specific quantum systems.

Choosing the set of performance metrics is one of the crucial parts during the DSE process. Performance metrics depend upon multidimensional functions that consider different input variables as the ones previously described in this section. Performance metrics are defined per DSE layer and serve to asses it (e.g circuit reliability for the mapper layer). However, we should also define global performance metric(s), that aggregates all the layer metrics and serve as figure of merit that should be optimized. Such a global metric should be architecture-neutral. One possible figure of merit can be the one proposed by IBM, the so-called *Quantum Volume* [19].

Quantum volume (QV) is an architecture-agnostic, single-number metric, which can be used for fair comparison between quantum technologies and devices. It is perhaps still early to form and be certain of a metric that will be able to be effective even after the NISQ era, on more scalable devices. Nevertheless, QV managed to catch the attention of the quantum research community, because it aggregates most of the elements from different layers that can affect the performance of a quantum systems. These elements include: number of physical qubits used for executing a given quantum circuit, number of gates that can be applied before gate errors and decoherence mask the result, qubit connectivity, available parallelization of operations, hardware-provided gate set, fidelity of operations and possibilities for circuit rewriting and optimization. In summary, QV tries to provide an answer to a question: can the device execute the given algorithm?. Note that devices with high-fidelity gates, good qubit connectivity, diverse gate set, and optimal circuit-rewriting tools will consequently have higher QV. In order to improve the QV value with higher number of qubits, improving gate fidelity and mapping optimization methods is a must.

IV. Preliminary results

As a first attempt to use DES methodologies to gain insight into the possible mapper solutions and features, we have swept in a structured way several internal parameters of the $Qmap$ mapper presented in [8]. Figures 6(b) and 6(c) show how the most common metrics used to assess the mapping procedure, the quantum gates ($G_{overhead}$) and circuit latency overhead ($L_{overhead}$), vary under different mapper design configurations (*Config. 1-4*) ordered in increasing complexity for the three routing strategies (Trivial, MinPath and MinextendRC) used in [8]. Each point represents an algorithm (benchmark) that has been mapped into the Surface-17 chip (see Table II in [8]). An aggregated figure of merit encompassing both metrics has been defined as $\frac{1}{G_{overhead} \times L_{overhead}}$. We can derive from these graphs that: i) increasing the complexity (more or improved features) of the mapper not always monotonically leads to lower gates/latency overhead; ii) the MinPath router, that optimizes for number of operations and therefore just takes one of the shortest path, and the MinextendRC, that chooses the routing path that minimally extends the circuit latency, show negligible difference; and 3) when considering both metrics (Figure 6(a)), it is observed that the overall goodness of the solution slightly improves, fluctuations apart, when increasing the complexity.

V. Discussion on the extension of the DSE methodology

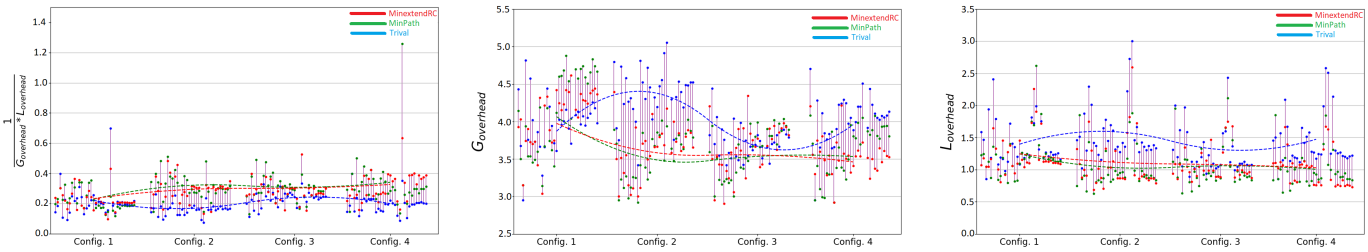
In the work presented here we have focused on the application of DSE methodologies to the mapping problem. However, more broadly, DSE methods can be applied to derive feasible and optimal quantum chip architectures as well as full-system designs for a specific kind of applications. Therefore, DSE methodologies will allow to perform a cross-layer co-design of the full-stack quantum system which is crucial in the NISQ era due to the relatively low number of qubits and the impact of noise on computation. Note that along the same lines, researchers recently started evaluating different quantum processor architecture designs based on an application-driven approach for different quantum technologies such as superconducting qubits [4] and trapped ions [20].

In addition, by extending this methodology across the several layers of the full-stack quantum system we could evaluate and compare different technologies and quantum processor choices (device level) as well as complete system designs (system level) and provide design guidelines and application-specific optimal designs for both current and future (scalability analysis) quantum hardware and full-stack quantum systems.

To this purpose, it is (again) essential the definition of a complete and classified set of benchmarks and appropriate system performance metrics, which are still missing. This issue was already identified in [21] in which they present a very large family of benchmarks, called *volumetric benchmarks*, that generalize the benchmarks used by IBM for measuring quantum volume [19]. They proposed to create such a benchmark family for probing the performance of a quantum computer and hope that a wide variety of them will be proposed to capture different performance aspects. So far, there is no consensus in the quantum computing community on what metric(s) should be used to assess quantum computers.

VI. Conclusion and future work

A key step in the compilation of quantum algorithms is the mapping process, in which the corresponding quantum circuit



(a) (b) (c)

Fig. 6: (a) Figure of merit. (b) Gate overhead. (c) Latency overhead.

is modified to comply to the quantum processor constraints. In this paper, we have proposed the application of structured DSE methodologies to perform a thorough performance comparison of different mapping solutions, allowing to analyse performance-complexity-scalability trade-offs, identify operational ranges and derive optimal designs for set of quantum applications and quantum devices. In this approach is fundamental to define the appropriate input variables to be swept and performance metrics. This idea of using DSE can also be extended to the full-stack quantum computing system to have a comprehensive understanding, and architecting and dimensioning the software and hardware layers for optimal design of current and future quantum computers. An important research topic towards the realisation of the approach presented here include the definition of accurate performance metrics for assessing the quality of the mapper as well as the quantum system and the proposition and in depth profiling of sets of quantum algorithms for benchmarking them.

ACKNOWLEDGMENTS

MB and CGA would like to acknowledge funding from Intel Corporation.

REFERENCES

[1] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for NISQ-era quantum devices. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, 2019.

[2] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1015–1029, 2019.

[3] Swamit S. Tannu and Moinuddin 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*, pages 987–999, 2019.

[4] Gushu Li, Yufei Ding, and Yuan Xie. Towards efficient superconducting quantum processor architecture design. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1031–1045, 2020.

[5] Alwin Zulehner, Alexandru Paler, and Robert 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.

[6] Davide Venturelli, Minh Do, Bryan O'Gorman, Jeremy Frank, Eleanor Rieffel, Kyle EC Booth, Thanh Nguyen, Parvathi Narayan, and Sasha Nanda. Quantum circuit compilation: An emerging application for automated reasoning. 2019.

[7] L Lao, B van Wee, I Ashraf, J van Someren, N Khammassi, K Bertels, and CG Almudever. Mapping of lattice surgery-based quantum circuits on surface code architectures. *Quantum Science and Technology*, 4:015005, 2019.

[8] Lingling Lao, Daniel M Manzano, Hans van Someren, Imran Ashraf, and Carmen G Almudever. Mapping of quantum circuits onto nisq superconducting processors. *arXiv preprint arXiv:1908.04226*, 2019.

[9] Steven Herbert and Akash Sengupta. Using reinforcement learning to find efficient qubit routing policies for deployment in near-term quantum computers. *arXiv:1812.11619*, 2018.

[10] Xiang Fu, L. Riesebos, M.A. Rol, Jeroen van Straten, J. van Someren, Nader Khammassi, Imran Ashraf, R.F.L. Vermeulen, V. Newsum, K.K.L. Loh, et al. eQASM: An executable quantum instruction set architecture. In *International Symposium on High Performance Computer Architecture*, pages 224–237. IEEE, 2019.

[11] Aaron Lye, Robert Wille, and Rolf Drechsler. Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In *Asia and South Pacific Design Automation Conference*, pages 178–183, 2015.

[12] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In *International Symposium on Code Generation and Optimization*, pages 113–125, 2018.

[13] Robert Wille, Oliver Keszocze, Marcel Walter, Patrick Rohrs, Anupam Chattopadhyay, and Rolf Drechsler. Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits. In *Asia and South Pacific Design Automation Conference*, pages 292–297, 2016.

[14] Gian Giacomo Guerreschi and Jongsoo Park. Two-step approach to scheduling quantum circuits. *Quantum Science and Technology*, 3(4):045003, 2018.

[15] Mohammad Javad Dousti and Massoud Pedram. Minimizing the latency of quantum circuits during mapping to the ion-trap circuit fabric. In *Design Automation and Test in Europe*, 2012.

[16] Tayebeh Bahreini and Naser Mohammadzadeh. An MINLP model for scheduling and placement of quantum circuits with a heuristic solution approach. *Journal on Emerhing Technologies in Computing*, 12(3):29, 2015.

[17] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology*, 3(2):025004, 2018.

[18] Gian Giacomo Guerreschi. Scheduler of quantum circuits based on dynamical pattern improvement and its application to hardware design. *arXiv:1912.00035*, 2019.

[19] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. Validating quantum computers using randomized model circuits. *arXiv:1811.12926*, 2018.

[20] Prakash Murali, Dripto M Debroy, Kenneth R Brown, and Margaret Martonosi. Architecting noisy intermediate-scale trapped ion quantum computers. *arXiv preprint arXiv:2004.04706*, 2020.

[21] Robin Blume-Kohout and Kevin C Young. A volumetric framework for quantum computer benchmarks. *arXiv preprint arXiv:1904.05546*, 2019.