Scaling of Multi-Core Quantum Architectures: A Communications-Aware Structured Gap Analysis

Santiago Rodrigo
srodrigo@ac.upc.edu
NaNoNetworking Center in Catalonia
(Universitat Politècnica de Catalunya)
Barcelona, Spain

Medina Bandic
M.Bandic@tudelft.nl
QuTech (Delft University of Technology)
Delft, Netherlands

Sergi Abadal
abadal@ac.upc.edu
NaNoNetworking Center in Catalonia
(Universitat Politècnica de Catalunya)
Barcelona, Spain

Hans van Someren
J.vanSomeren-1@tudelft.nl
QuTech (Delft University of Technology)
Delft, Netherlands

Eduard Alarcón
eduard.alarcon@upc.edu
NaNoNetworking Center in Catalonia
(Universitat Politècnica de Catalunya)
Barcelona, Spain

Carmen G. Almudéver
cargara2@disca.upv.es
Universitat Politècnica de Valencia
Valencia, Spain
QuTech (Delft University of Technology)
Delft, Netherlands

ABSTRACT

In the quest of large-scale quantum computers, multi-core distributed architectures are considered a compelling alternative to be explored. A crucial aspect in such approach is the stringent demand on communication among cores when qubits need to interact, which conditions the scalability potential of these architectures. In this work, we address the question of how the cost of the communication among cores impacts on the viability of the quantum multi-core approach. Methodologically, we consider a design space in which architectural variables (number of cores, number of qubits per core), application variables for several quantum benchmarks (number of qubits, number of gates, percentage of two-qubit gates) and inter-core communication latency are swept along with the definition of a figure of merit. This approach yields both a qualitative understanding of trends in the design space and companion dimensioning guidelines for the architecture, including optimal points, as well as quantitative answers to the question of beyond which communication performance levels the multi-core architecture pays off. Our results allow to determine the thresholds for inter-core communication latency in order for multi-core architectures to outperform single-core quantum processors.

CCS CONCEPTS

• **Computer systems organization → Quantum computing**; **Distributed architectures**; • **General and reference → Design**; • **Networks → Network on chip**.

KEYWORDS

Quantum Computing, Many-core Quantum Computers, Quantum Communications, Design Space Exploration, Quantum Computers Scalability

1 INTRODUCTION

Quantum computing is expected to open the door to a revolution in the broad fields of computing and communications. This is possible by virtue of unconventional properties of quantum mechanics such as superposition or entanglement, which could potentially allow solving certain algorithms exponentially faster than on classical computers and provide unconditional security in communications [1, 2, 3, 4, 5].

The key challenges preventing the scaling and wide adoption of quantum computers are manifold. They could be summarized in that qubits, i.e. the *alter ego* of classical bits in the quantum world, are very sensitive to noise and any operations applied to them. That makes it very difficult to preserve their coherence and not introduce errors in the computation. To minimize this, quantum processors are nowadays kept isolated at cryogenic temperatures and controlled externally. For all this, the addressing and control of large qubit arrays becomes extremely challenging.

Despite impressive recent advances, existing realizations of quantum computers are still too small and noisy to allow for the experimental demonstration of their full potential. They are referred to as Noisy Intermediate-Scale Quantum (NISQ) computers. For instance, the largest functional NISQ computers presented thus far have less than a hundred qubits[6, 7], which is far from the number of qubits needed for showing the true advantage of these computers and their applicability to solve real problems [8, 5].

The final publication is available at ACM via http://dx.doi.org/10.1145/3457388.3458674

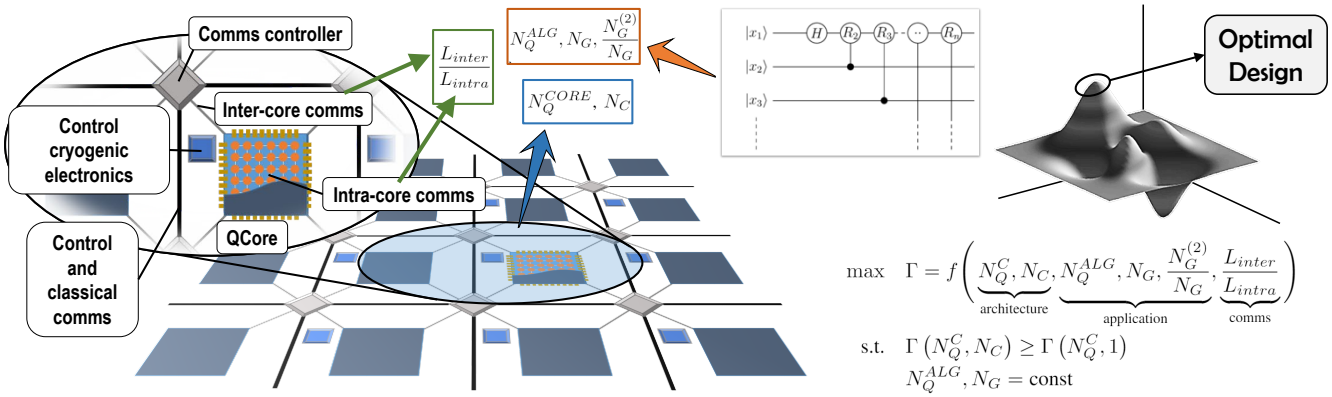

Figure 1: Design Space Exploration for multi-core quantum computing architectures

To increase these numbers, tremendous efforts are being put into allowing a dense integration of many qubits within one chip [9, 10]. However, this approach is conjectured to have its limit in a few thousand qubits, due to the impracticality of integrating the required control circuits and per-qubit wiring and still maintaining a low quantum error rate [11].

An alternative or complementary approach to scale existing quantum computers consists in the creation of quantum multi-core architectures with dozens of NISQ cores (see Fig. 1). Each of these cores would contain a relatively low number of qubits, from tens to hundreds, and be connected through a quantum interconnect for core-to-core coherent qubit transport (via, for instance, quantum teleportation or photonic shuttling [12, 13]) and a control classical network for core coordination, job distribution or quantum teleportation assistance. This disintegration approach alleviates the requirements for control circuits and improves qubit isolation, while still maintaining all the advantages of quantum parallelism as long as cores are kept coherent through the quantum interconnect.

Various proposals [14, 15, 16, 17, 18, 19, 20] in the literature agree on using this approach at multiple scales and studied the use of multiple qubit technologies (e.g. ion trap, quantum dots or impurities in solids) and quantum interconnect alternatives (e.g. ion shuttling, quantum teleportation) to realize the vision. However, to the best of our knowledge, a thorough architectural top-down analysis of the scaling potential, resource overheads and computational costs of multi-core architectures is missing. A more detailed study was conducted in [21], where technology-agnostic multi-core quantum computer performance is explored for a wide range of configurations in terms of number of qubits, number of cores and some technological parameters. However, as the authors already mention, the analysis is only a preliminary approach, as it is limited to an analytical Figure of Merit (FoM), not related to any previously existing well-known quantum metric or simulations based upon experiment-supported models. Although the conclusions on the effectiveness of the multi-core approach are promising, stronger foundations need to be laid in order to give more concrete directions for quantum computer designers and quantum technology developers.

Specifically, the trade-off between communications overhead present in multi-core architectures and the gain in size of the algorithms that can be executed on them, needs to be profiled. This will help not only to determine the viability of multi-core computing but also to characterize the *decision threshold* where the communications cost of distributing quantum computation among several cores pays off.

Studies on this trade-off are available [22, 23, 24, 25, 26], although none of them allows us to answer any of these key questions, namely: how fast should inter-core communications be in order to allow multi-core architectures to supersede traditional single-core quantum processors? For a given interconnect technology, which is the *optimal* architectural configuration that the communication costs pay off?

This is the aim of the present work. We attempt to do so by performing a Design Space Exploration (DSE) of the architectural and technological variables that configure multi-core quantum computers. We have used the OpenQL [27] compiler and QMap mapper [28] with several random and real application benchmarks to compute the actual performance for different multi-core configurations, comparing it with the traditional single-core results.

2 EXECUTING QUANTUM ALGORITHMS ON MULTI-CORE QUANTUM ARCHITECTURES

Quantum algorithms are usually expressed as quantum circuits that are agnostic of quantum hardware, i.e. it is assumed that all qubits can interact with each other or that quantum gates can be performed in parallel as long as their dependencies are respected. However, quantum processors suffer from several constraints that must be satisfied when executing a quantum algorithm on them [28]. Qubits are arranged on a specific topology, and although all-to-all qubit connectivity is possible for trapped-ion processors [29], in most of the quantum devices one of the most stringent constraints is the reduced connectivity between qubits, limiting their possible interactions to, for instance, only nearest-neighbour.

Accordingly, different mapping approaches have been proposed for realising quantum algorithms in connectivity-constrained, and

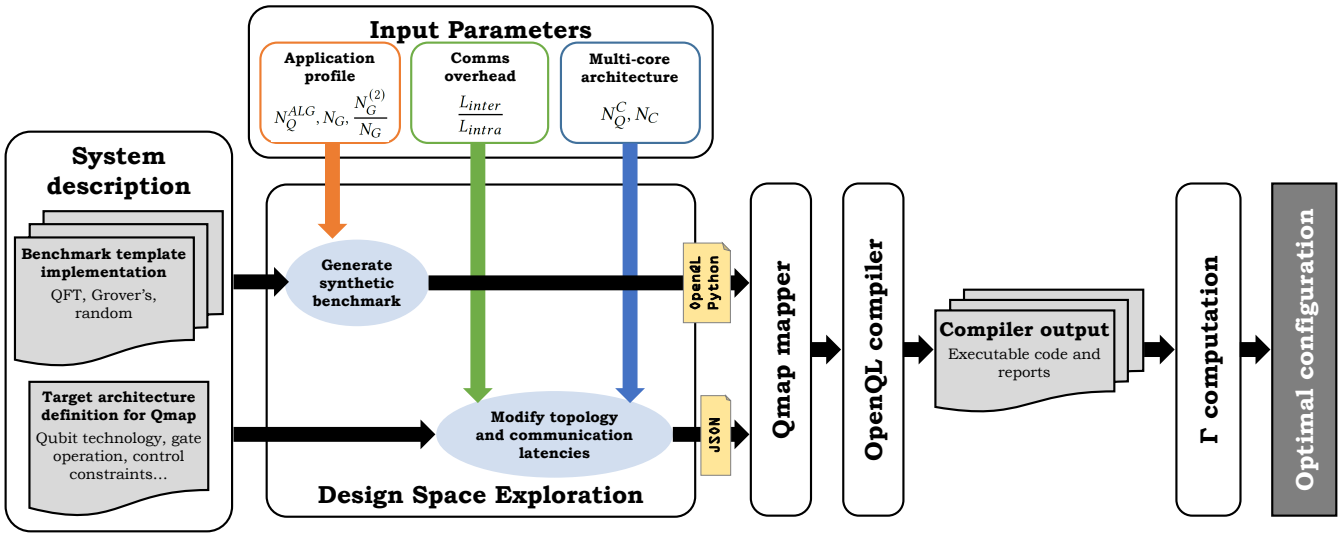

Figure 2: Flow diagram of the evaluation framework used for the Design Space Exploration

in more general resource-constrained, single-core quantum architectures [30]. The main steps in this process, which are also relevant for the multi-core case, are: 1) initial placement of qubits in which the virtual qubits (qubits in the circuit) are assigned to the physical ones (qubits in the quantum chip); 2) routing of the qubits to adjacent positions whenever they need to interact (using specific quantum operations such as SWAP gates); and 3) scheduling of quantum operations to leverage their parallelism and reduce the overall circuit latency (i.e. the time it takes to complete the execution). This process results in an increase in the number of gates and circuit depth (number of steps in the circuit) that in turn decreases the success rate of the algorithm [31].

These mapping techniques can be applied and extended when scaling the architecture to multiple cores as the connectivity among cores is also limited. In this case, virtual qubits can be mapped to a physical qubit in any core. Ideally, qubits with a high-degree of interactions should be placed in the same core. When two qubits that are in different cores need to interact, they will have to be moved to the same core. Therefore, multi-core architectures (see Fig. 1) require not only intra-core qubit movement operations (e.g. SWAP gates), but also inter-core communication operations such as quantum teleportation or qubit shuttling [22].

Depending on the interconnect topology of the multi-core architecture, the restrictions on inter-core communications will vary. As in the single-core case, this limitation will result in an increase of quantum operations. In addition, inter-core communication is slower and more error-prone than intra-core communication operations: latencies are from 5× to 100× longer, and the error rates are on average 10× to 100× worse for quantum teleportation than for two-qubit gates [25, 14, 32]. All of these overhead sources need to be analysed as they will substantially affect the algorithm execution and reliability of the final results.

3 EXPLORING THE MULTI-CORE ARCHITECTURES PERFORMANCE

In this section, we present the framework for the analysis: architectural assumptions and performance metrics, chosen benchmarks, compilation framework and problem formulation. See in Fig. 2 a flow diagram of the design exploration process.

3.1 Modeling Assumptions

Multi-core quantum topology: We assume the quantum computer to be composed of one or several interconnected cores, which could be compared to any current NISQ quantum computer; that is, a computing core consisting of some tens or hundreds of qubits. In order to keep the focus on the effect of inter-core communication, we have assumed all-to-all intra-core connectivity, i.e. every physical qubit can interact with any other as long as they placed in the same core. As in [14] and [16], full connectivity is also assumed among the cores, meaning that qubits placed in different cores are at 'one hop' distance, as there is an entangled pairs generator which is shared among all cores.

Inter-core communication: We have chosen quantum teleportation as the inter-core communication operation, as it is a firm candidate for this role, and has been already demonstrated with several technologies [33, 34, 35, 36]. Quantum teleportation is based on the properties of entangled pairs of qubits, that are shared among two entities and enable them to communicate a quantum state, through a few simple operations and some classical communication, without the need of sending the physical qubit, and hence increasing the success rate.

Decoherence and other quantum noise sources are not directly considered in the analysis. Though these error sources considerably affect the accuracy of the computation results, as mentioned before, when studying the effect of communications it is of interest to focus

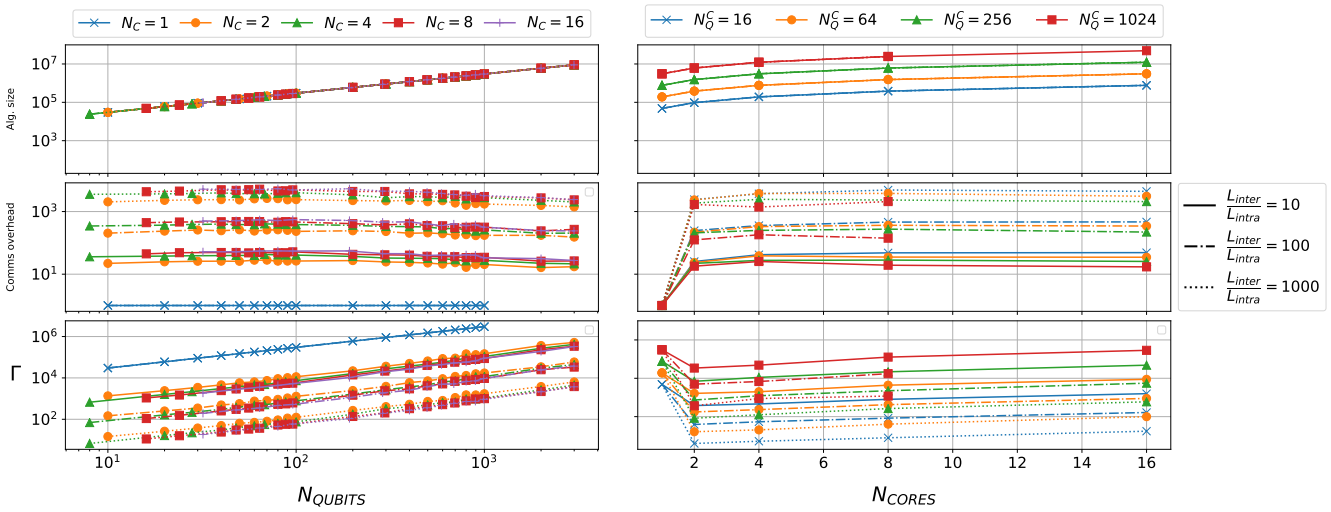

Figure 3: Full-blown variable exploration of multi-core quantum architectures for the random benchmark (80% of gates are two-qubit operations). From top to bottom, size of the algorithm (qubits × gates), communications overhead and Figure of Merit Γ are plotted against the number of qubits of the algorithm and the number of cores of the architecture, varying the number of qubits per core (and the total number of physical qubits accordingly). In this benchmark, the number of gates has been fixed, while the number of qubits of the algorithm increases along the size of the architecture.

on the latency overhead. It is the main hurdle of inter-core communications when compared to intra-core operations and deserves an isolated study. Certainly, considering the effects on the overall error rates is an important part of our future work, requiring accurate error models and specific tools.

In any case, it is key to note that in quantum communications latency and error rate are tightly related: the more time is consumed in the data transfer, the higher will be the decoherence effect on the result error (assuming no error correction or entanglement distillation techniques). Also, correcting errors and dealing with them incurs in higher execution latency. Accordingly, the present study, although focused on latency overhead, gives guidelines that are applicable as well for real-world error-prone systems.

3.2 Selected Benchmarks

With the aim of evaluating the communication overhead in multi-core quantum architectures and its potential for solving the scalability issue, we have used the Quantum Fourier Transform (QFT) and Grover's search algorithm as benchmarks [37, 25]. These benchmarks, however, have a specific structure and scale in a predefined steady manner, which limits our control over their parameters. For instance, the number of gates for Grover's algorithm scales in a linear manner with number of qubits, whereas its percentage of two-qubit gates (after decomposition) is around 30%.

In order to overcome this limitation, additionally we decided to use synthetically generated benchmarks in the form of random circuits. These can provide the necessary freedom to explore algorithm parameters like number of qubits, gates and the percentage of two-qubit gates mentioned previously. The importance of using the family of random circuit benchmarks was pointed out by some previous works as well [38, 39, 40, 25, 41, 42, 43], which also introduced various ways of generating them. They differ in type of randomness, shape (width vs. depth), gate density per layer, etc. In our case, we opted for random circuits that are generated by uniformly selecting gates from a predefined set and uniformly selecting qubits (one or two qubits for single- or two-qubit gates, respectively) to apply those gates on. Beforehand, we had to decide on the percentage of two-qubit gates, as this parameter is the one that defines the amount of qubit interactions. We chose the two extreme values of 20% and 80% to showcase the impact of communication.

3.3 Extending the Qmap mapper

In order to evaluate the communication costs, we have mapped the previously presented benchmarks into different multi-core quantum architectures. To this purpose, we used the OpenQL quantum programming framework [27] and the Qmap mapper [28] embedded in it. In this work, we have modified the Qmap mapper, which is meant for single-core resource-constrained quantum processors, and extended it to the multi-core case.

The main modifications are the following: 1) together with the definition of each core's description (gate latencies, qubit connectivity constraints, supported operations, etc.), the configuration file includes the topology specification of the multi-core architecture (how many cores, inter-core connectivity, and number of qubits/core) as well as the inter-core communication latency; 2) in the initial placement, a single core can be assigned multiple qubits; 3) an inter-core qubit transfer operation has been defined and is inserted during the routing process when necessary.

Note that as all-to-all intra-core connectivity is assumed, the OpenQL compiler, apart from satisfying the dependencies among different instructions, has only to take into account inter-core movement; if any pair of qubits placed in different cores need to interact,

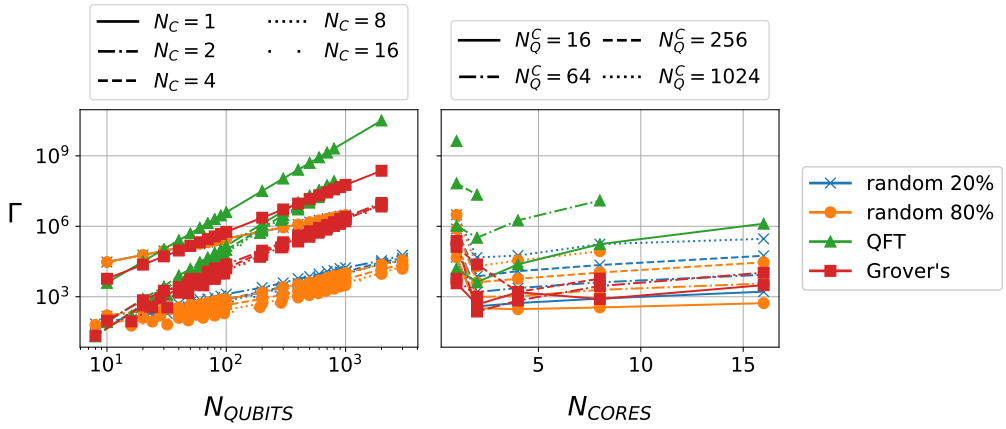

Figure 4: Several benchmarks' scalability on different multi-cores architectures: two different random benchmarks (varying the fraction of two qubit-gates (20%–80%), QFT and Grover's search

the compiler has to insert the teleportation operation and modify the execution schedule to wait for the movement to complete, after which the qubits interact inside one core. Observe however that, having any-to-any communication in both inside the core and among the cores, the routing phase of Qmap has no way to differentiate the 'one hop' distance between qubits in the same core and that of two qubits placed in different cores. In order to solve this, the compiler adds an extra hop in the latter case and avoids paths with more than one inter-core communication. Note also that the Qmap mapper tries to reduce the resulting circuit latency overhead (overall circuit execution time).

3.4 Problem Formulation

We are considering multi-core architectures as an approach that may help to unleash the quantum computing potential, which is right now limited by the number of qubits of current processors. Therefore, being focused in determining the *decision threshold* between monolithic single-core and multi-core quantum architectures, the communication cost and available computing capacity for the algorithm being executed must be the key point of the exploration.

Hence, the proposed performance metric may be defined as

$$\Gamma = \frac{\text{Algorithm size}}{\text{Comms overhead}} = \frac{\#\text{ gates} \times \#\text{ qubits required}}{\left(\frac{\text{Latency (multi-core)}}{\text{Latency (single-core)}}\right)} \tag{1}$$

In Eq. (1), the algorithm size refers to the product of its number of qubits and its number of gates. The execution latency (different from the communication latencies L_{inter} and L_{intra}) accounts for the total number of processor cycles the code takes, after performing a platform-specific compilation (i.e., mapping and instructions scheduling).

The performance metric Γ depends on three different aspects of the problem: the architecture configuration, the application-specific parameters, and the technology specifications. According to our modeling assumptions, we can narrow down these dependencies to five variables: number of qubits per core and number of cores (architecture); total number of qubits and gates required by the algorithm and 2-qubit gates fraction (application), and the ratio between the inter-core communication latency and the cost of an intra-chip communication operation (communications).

Therefore, we are looking in the design space for the points where multi-core architectures perform better than single-core processors. That is to say, to find the maximum performance Γ for any given architecture configuration (number of qubits per core N_Q^C and number of cores N_C), for every application (i.e. number of gates N_G, 2-qubit gates fraction $N_G^{(2)}/N_G$ and number of qubits required N_Q^{ALG}) and for a fixed communications overhead (L_{inter}/L_{intra}). This can be summarized in the formulation below:

$$\begin{aligned} \min \quad & \frac{L_{inter}}{L_{intra}} \\ \text{s.t.} \quad & \Gamma'\left(N_Q^C, N_C\right) \geq \Gamma'\left(N_Q^C, 1\right) \\ & N_Q^{ALG}, N_G = \text{const} \end{aligned} \tag{2}$$

Intra- and inter-core topology, qubit implementation and interconnect technology are assumed to be fixed, while the application-specific parameters vary on every benchmark.

4 RESULTS

We have performed an in-depth exploration of the design space, sweeping all the input variables in wide ranges. Standard benchmarks (QFT, Grover's) have their own profile in terms of number of gates (N_G) and 2-qubit gates fraction ($N_G^{(2)}/N_G$), both of which depend on the size of the input (number of qubits required, N_Q^{ALG}), which we have varied from 10 to 3000. In the random case, we have fixed the number of gates, while sweeping the number of qubits in the same range, as a way to keep constant the number of independent variables modifying the output performance. Aiming at the future high-scaling multi-core computers, we have explored ranging from modest sizes (2 cores and 10 qubits per core) to 16 cores and 1024 qubits per core. Finally, we have characterised costly inter-core communications with latencies ratio (L_{inter}/L_{intra}) that

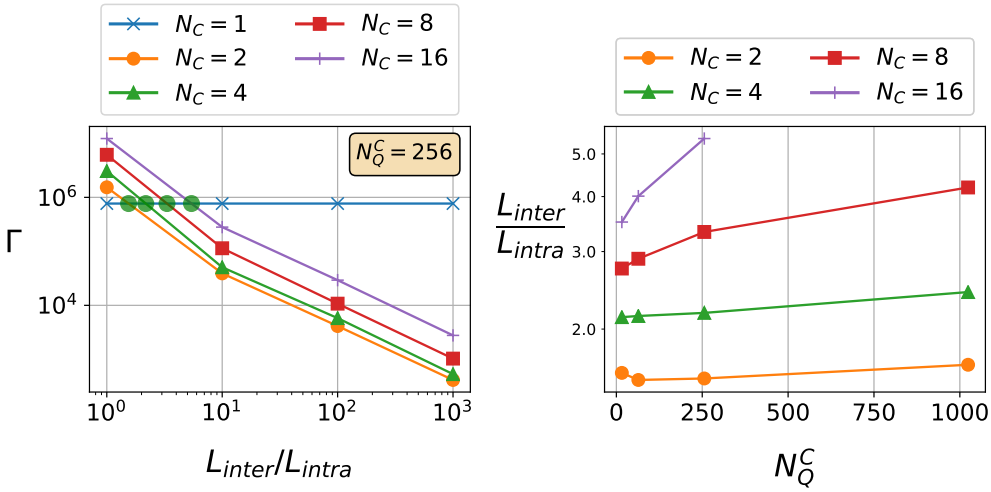

Figure 5: Design Space Exploration ultimate goal: optimal inter-core latency for every architecture (here the case for random (80%) is shown). The evaluation points lacking in the plot correspond to compilations that took an excessive amount of time to complete

Table 1: Design Space Exploration

	QFT	Grover's	Random 0.2	Random 0.8
N_G	$O(N_Q^{ALG})^2$	$O(N_Q^{ALG})$	3000	3000
$N_G^{(2)}/N_G$	~ 50%	~ 30%	20%	80%
N_Q^{ALG}	[10, 3000]			
N_C	[1, 16]			
N_Q^C	{16, 64, 256, 1024}			
$\frac{L_{inter}}{L_{intra}}$	{10, 100, 1000}			

starts from 10 and goes up to 1000, in order to stress the analysis in the most crucial point. A summary of these input parameters can be found in Table 1.

In Fig. 3, the full-blown variable space exploration for a single benchmark (the random benchmark with 80% of 2-qubit gates) is shown. The design space is projected onto the two architecture-related dimensions, i.e. total number of qubits and number of cores available in the computer. From top to bottom, the FoM Γ components are shown (the algorithm size is plotted in the first row, right below the communications overhead can be found, and the aggregated final metric is on the last row). In this way, we can see the effects of both parameters in the performance.

In the left side, we can see the different clusters of lines in the communications overhead plot (in the middle), for the three different ratios of L_{inter}/L_{intra}: the performance scales linearly with the inter-core latency. This does not benefit the multi-core approach, as the single-core processor performs always better. Very importantly, observe that the single-core curve is discontinued at 10^3 qubits, as that is the forecasted approximate upper limit (number of qubits in a single core) for current qubit integration technologies[11]. This allows multi-core architectures performance grow past that limit, achieving single-core levels of performance by increasing the number of qubits and number of cores, for the same qubit technology. For a fixed number of cores, the communications overhead decreases slowly (as more qubits are fitted inside the same core, hence less inter-core movements are needed), and thus the performance increases. Note also that the communications overhead is higher than the L_{inter}/L_{intra}, as the overhead accounts as well for delays in instructions scheduling that are a byproduct of longer qubit transportation waiting times.

The bottom-right plot contains the most valuable information in this first wide exploration: we can see from left to right the cost of going from single-core to multi-core (see the steep descent of the performance in the interval [1, 2], and most importantly, how the multi-core architecture, as the number of cores are increased (i.e. parallelism), recovers performance until almost reaching single-core performance values with 16 cores working together. That is, adding cores allows increasing the number of available physical qubits for computation, breaking the single-core qubit integration limit, and overcoming the latency overhead from inter-core communications. This sheds a light on the potential of multi-core architectures.

The same full-blown exploration for QFT, Grover's and the low-communication random benchmark are also shown in Fig. 4, and compared to the already seen high-communication random benchmark. The lowest performance is that of the random algorithms, as the two-qubit gates are distributed uniformly, without any structure, among all the qubits: even the best efforts coming from the mapper in the low-communication benchmark can decrease inter-core qubit interaction. QFT shows a steeper increase in performance as the architecture grows: even having a moderate amount of qubit communication (see Table 1), its structure allows the mapper to distribute it properly. Note that its minimum performance is the worst,

but it also shows the higher maximum, surpassing the single-core performance with as few as 4 cores. In a nutshell, multi-core architectures benefit from highly-structured pieces of quantum code, which are frequently used in many well-known algorithms.

Having explored the design space in this way, we have been able to see, for different architectures, applications and inter-core communication performance, the *balance point* where multi-core architectures supersede single-core performance. Therefore, let us now explore this the other way around, looking for the *decision threshold* where the inter-core communication adoption in multi-core architectures starts to pay off and exceed single-core performance. This *decision threshold* will depend on the application and the architecture. See in Fig. 5, left plot, the graphical representation of the *decision threshold* finding for a specific application and several architectures, as the crossing point of the performance single-core line (which is flat, as it does not depend on the inter-core communication latency) with the performance curves of different architectures. In Fig. 5, right plot, the *decision thresholds* curves for a wide range of architectures are shown. Observe that, for instance, a quantum computer with 4 cores and 256 qubits will outperform the corresponding single-core's performance if $L_{inter} < 2.1 \times L_{intra}$ (take into account that the benchmark shown is communication intensive). Note also that, even though the use of more cores implies higher inter-core qubit communication, the more cores we use, the lower the communication performance (i.e. the higher the maximum L_{inter}/L_{intra} ratio) is required.

5 CONCLUSIONS AND FUTURE WORK

In this article, we have explored the multi-core quantum architectures space in order to find the threshold where these distributed architectures outperform single-core traditional quantum processors, focusing on the inter-core communication latency. Together with these results, that indicate upper bounds for inter-core communication technology performance, we have shown that multi-core architectures, in addition to break the qubit integration limits of single-core quantum computers, exploit parallelism for widely used well-known structured quantum algorithms.

Future work will include some other important constraints, such as qubit control and operation, connectivity-constrained intra-core topology and, very importantly, intra- and inter-core operation error rates. In this way, we will be able to complement this execution latency analysis with a computation accuracy analysis. The obtained results are very promising, as they pave the way to exposing some design guidelines on these architectures using standard quantum applications.

REFERENCES

[1] Salonik Resch and Ulya R Karpuzcu. 2019. Quantum computing: an overview across the system stack. *arXiv preprint arXiv: 1905.07240*.

[2] Margaret Martonosi and Martin Roetteler. 2019. Next steps in quantum computing: computer science's role. *arXiv preprint arXiv: 1903.10541*.

[3] Stephanie Wehner, David Elkouss, and Ronald Hanson. 2018. Quantum internet: a vision for the road ahead. *Science*, 362, 6412, eaam9288.

[4] Thaddeus D Ladd, Fedor Jelezko, Raymond Laflamme, Yasunobu Nakamura, Christopher Monroe, and Jeremy Lloyd O'Brien. 2010. Quantum computers. *Nature*, 464, 7285, 45–53.

[5] Rodney Van Meter and Clare Horsman. 2013. A blueprint for building a quantum computer. *Communications of the ACM*, 56, 10, 84–93.

[6] Doug McClure and Jay Gambetta. [n. d.] Quantum computation center opens (IBM Research Blog). IBM Research Blog, editor. (). Retrieved 01/28/2020 from https://www.ibm.com/blogs/research/2019/09/quantum-computation-center/.

[7] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature*, 574, 7779, 505–510.

[8] John Preskill. 2012. Quantum computing and the entanglement frontier. *arXiv preprint arXiv: 1203.5813*.

[9] JM Hornibrook, JI Colless, ID Conway Lamb, SJ Pauka, H Lu, AC Gossard, JD Watson, GC Gardner, S Fallahi, MJ Manfra, et al. 2015. Cryogenic control architecture for large-scale quantum computing. *Physical Review Applied*, 3, 2, 024010.

[10] Harald Homulle, Stefan Visser, Bishnu Patra, Giorgio Ferrari, Enrico Prati, Fabio Sebastiano, and Edoardo Charbon. 2017. A reconfigurable cryogenic platform for the classical control of quantum processors. *Review of Scientific Instruments*, 88, 4, 045103.

[11] National Academies of Sciences. 2019. *Quantum computing: progress and prospects*. National Academies Press.

[12] Dik Bouwmeester, Jian-Wei Pan, Klaus Mattle, Manfred Eibl, Harald Weinfurter, and Anton Zeilinger. 1997. Experimental quantum teleportation. *Nature*, 390, 6660, 575–579.

[13] WK Hensinger, S Olmschenk, D Stick, D Hucul, M Yeo, M Acton, L Deslauriers, C Monroe, and J Rabchuk. 2006. T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation. *Applied Physics Letters*, 88, 3, 034101.

[14] C Monroe, R Raussendorf, A Ruthven, KR Brown, P Maunz, L-M Duan, and J Kim. 2014. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Physical Review A*, 89, 2, 022317.

[15] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi. 2019. Quantum internet: networking challenges in distributed quantum computing. *IEEE Network*, 34, 1, 137–143.

[16] Kenneth R Brown, Jungsang Kim, and Christopher Monroe. 2016. Co-designing a scalable quantum computer with trapped atomic ions. *npj Quantum Information*, 2, 1, 1–10.

[17] LMK Vandersypen, H Bluhm, JS Clarke, AS Dzurak, R Ishihara, A Morello, DJ Reilly, LR Schreiber, and M Veldhorst. 2017. Interfacing spin qubits in quantum dots and donors — hot, dense, and coherent. *npj Quantum Information*, 3, 1, 1–10.

[18] Liang Jiang, Jacob M Taylor, Anders S Sørensen, and Mikhail D Lukin. 2007. Distributed quantum computation based on small quantum registers. *Physical Review A*, 76, 6, 062323.

[19] Sahar Sargaran and Naser Mohammadzadeh. 2019. Saqip: a scalable architecture for quantum information processors.

ACM Transactions on Architecture and Code Optimization (TACO), 16, 2, 1–21.

[20] Nemanja Isailovic, Yatish Patel, Mark Whitney, and John Kubiatowicz. 2006. Interconnection networks for scalable quantum computers. In *33rd International Symposium on Computer Architecture (ISCA'06)*. IEEE, 366–377.

[21] Santiago Rodrigo, Sergi Abadal, Eduard Alarcón, and Carmen G Almudever. 2020. Exploring a double full-stack communications - enabled architecture for multi-core quantum computers. *arXiv preprint arXiv: 2009.08186*.

[22] Mark Oskin, Frederic T Chong, Isaac L Chuang, and John Kubiatowicz. 2003. Building quantum wires: the long and the short of it. In *30th Annual International Symposium on Computer Architecture, 2003. Proceedings.* IEEE, 374–385.

[23] Jeff Heckey, Shruti Patil, Ali JavadiAbhari, Adam Holmes, Daniel Kudrow, Kenneth R Brown, Diana Franklin, Frederic T Chong, and Margaret Martonosi. 2015. Compiler management of communication and parallelism for quantum computation. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, 445–456.

[24] Robert Risque and Adwait Jog. 2016. Characterization of quantum workloads on SIMD architectures. In *2016 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 1–9.

[25] Jonathan M Baker, Casey Duckering, Alexander Hoover, and Frederic T Chong. 2020. Time-sliced quantum circuit partitioning for modular architectures. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*, 98–107.

[26] Darshan D Thaker, Tzvetan S Metodi, and Frederic T Chong. 2006. A realizable distributed ion-trap quantum computer. In *International Conference on High-Performance Computing*. Springer, 111–122.

[27] Nader Khammassi, Imran Ashraf, J van Someren, Razvan Nane, AM Krol, M Adriaan Rol, L Lao, Koen Bertels, and Carmen G Almudever. 2020. OpenQL: a portable quantum programming framework for quantum accelerators. *arXiv preprint arXiv: 2005.13283*.

[28] Lingling Lao, Hans van Someren, Imran Ashraf, and Carmen G. Almudever. 2019. Timing and resource-aware mapping of quantum circuits to superconducting processors. *arXiv preprint arXiv: 1908.04226*. arXiv: 1908.04226 [quant-ph].

[29] Norbert M Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A Landsman, Kenneth Wright, and Christopher Monroe. 2017. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114, 13, 3305–3310.

[30] Carmen G. Almudever, Lingling Lao, Robert Wille, and Gian G. Guerreschi. 2020. Realizing quantum algorithms on real quantum computing devices. In *Proceedings of the 23rd Conference on Design, Automation and Test in Europe* (DATE '20). EDA Consortium, Grenoble, France, 864–872.

[31] Prakash Murali, Norbert M Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. 2020. Architecting noisy intermediate-scale quantum computers: a real-system study. *IEEE Micro*, 40, 3, 73–80.

[32] Bharath Kannan, Daniel Campbell, Francisca Vasconcelos, Roni Winik, David Kim, Morten Kjaergaard, Philip Krantz, Alexander Melville, Bethany M Niedzielski, Jonilyn Yoder, et al. 2020. Generating spatially entangled itinerant photons with waveguide quantum electrodynamics. *arXiv preprint arXiv: 2003.07300*.

[33] Stephan Welte, Bastian Hacker, Severin Daiss, Stephan Ritter, and Gerhard Rempe. 2018. Photon-mediated quantum gate between two neutral atoms in an optical cavity. *Physical Review X*, 8, 1, 011018.

[34] Juan Ignacio Cirac, Peter Zoller, H Jeff Kimble, and Hideo Mabuchi. 1997. Quantum state transfer and entanglement distribution among distant nodes in a quantum network. *Physical Review Letters*, 78, 16, 3221.

[35] Hannes Bernien, Bas Hensen, Wolfgang Pfaff, Gerwin Koolstra, Machiel S Blok, Lucio Robledo, TH Taminiau, Matthew Markham, Daniel J Twitchen, Lilian Childress, et al. 2013. Heralded entanglement between solid-state qubits separated by three metres. *Nature*, 497, 7447, 86–90.

[36] Philipp Kurpiers, Paul Magnard, Theo Walter, Baptiste Royer, Marek Pechal, Johannes Heinsoo, Yves Salathé, Abdulkadir Akin, Simon Storz, J-C Besse, et al. 2018. Deterministic quantum state transfer and remote entanglement using microwave photons. *Nature*, 558, 7709, 264–267.

[37] Michael A Nielsen and Isaac Chuang. 2002. Quantum computation and quantum information. (2002).

[38] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. 2019. Validating quantum computers using randomized model circuits. *Physical Review A*, 100, 3, 032328.

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

[40] Daniel Mills, Seyon Sivarajah, Travis L Scholten, and Ross Duncan. 2020. Application-motivated, holistic benchmarking of a full quantum computing stack. *arXiv preprint arXiv: 2006.01273*.

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

[42] Matteo G Pozzi, Steven J Herbert, Akash Sengupta, and Robert D Mullins. 2020. Using reinforcement learning to perform qubit routing in quantum compilers. *arXiv preprint arXiv: 2007.15957*.

[43] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. 2020. T|ket>: a retargetable compiler for nisq devices. *Quantum Science and Technology*.