A Bird's Eye View on Quantum Computing: Current and Future Trends

Beatrice Branchini, Davide Conficconi, Francesco Peverelli, Donatella Sciuto, Marco D. Santambrogio
Dipartimento di Elettronica, Informatica e Bioingegneria, Politecnico di Milano, Milan, Italy
{beatrice.branchini, davide.conficconi, francesco.peverelli, donatella.sciuto, marco.santambrogio}@polimi.it

***Abstract*—Quantum computing is a potentially highly-disruptive technology for several domains across the Computer Science field. However, many technological challenges still prevent the construction of a fault-tolerant and reliable quantum computer. These challenges sparked enormous interest in the scientific community, which led to the development of several independent strands of research. Consequently, research in the quantum computing domain is fragmented, making it highly specialized. This can lead to missed opportunities in identifying valuable research directions and contributions from neighboring research areas. For these reasons, this paper provides an overview of the current state of play in quantum computing for different quantum research areas at different layers of the development stack. Furthermore, for each of them, we highlight possible new outlooks that could be pivotal in the near future.**

***Index Terms*—quantum computing, review, future trends**

I. Introduction and Motivation

The dusk of Moore's Law and Dennard scaling contrasts with the continuously growing demand for computational power [1]. This gap between the need for computing power and achievable performance forces the search for new architectural solutions by domain-specialized systems or new technologies [2]. In this scenario, quantum computing represents a novel paradigm for which to strive. Quantum computing promises to solve classes of problems that even massively-parallel computing systems cannot handle in a reasonable amount of time, more efficiently, and with higher precision [3]. This technology relies on the laws of quantum mechanics, exploiting phenomena such as entanglement and superposition to enable higher degrees of parallelism. However, at the moment, we are far from physically implementing a reliable quantum computer and proving *quantum supremacy* [4].

No theoretical proof exists that we cannot build a large, fault-tolerant quantum computer [5]. Nevertheless, the technical challenges in building such a system are considerable and involve many fields, from Computer Science to Physics and Materials Science [5]. Therefore, many researchers started exploring quantum phenomena and their potential.

In this work, we propose an **overview** of the **quantum research landscape** from a **Computer Science and Engineering perspective**. We aim to provide an analysis of the current State of the Art of quantum hardware and the most relevant related tools used to develop quantum-accelerated applications. We also highlight which, in light of our analysis, could be **new and promising research directions** in the domain. With these evaluations, on the one hand, we want to provide a way for existing stakeholders to assess quantum research's current state of play. On the other hand, our evaluations can serve as a guide for new players toward worthy research directions.

DOI 10.1109/EUROCONXXXXX.2023.XXXXXXX

II. Electronic Design Automation for Quantum

According to the universal gate-based computation model, describing quantum algorithms translates into specifying a high-level quantum circuit with a certain number of qubits. The abstraction stack from a quantum circuit description to its execution resembles the classical one [6] (Figure 1).

Firstly, **compiling** the circuit allows for obtaining an equivalent representation in a quantum assembly language. Then, it is necessary to **synthesize** the circuit to match the underlying technology and ensure reversibility. Finally, the circuit is **mapped** onto the available hardware resources [7]. We can execute our quantum algorithms only after these three steps.

Quantum algorithms' implementation relies on a quantum circuit described through Quantum Programming Languages (QPLs): some closer to the gate representation and others more similar to high-level languages. We can divide QPLs into two classes, imperative and functional [8]. The first class comprises low–level imperative languages relying on OpenQASM [9] (by extending or adapting it) and higher–level ones built on Python. Instead, functional programming languages are relatively new and often based on lambda calculus or Haskell. In general, all the languages are agnostic on the fault tolerance of the underlying hardware and assume that both qubits and gate operations are perfect [7]. A QPL-based algorithm can be compiled through a universal set of gates.

Then, a synthesis step translates the compiled algorithm into an equivalent circuit featuring the gates belonging to the target machine's gate set. This set usually comprises a few one-qubit gates with some additional simple two-qubit gates (e.g., CNOT and SWAP gates) [7]. Therefore, given a quantum control flow, different traces might exist depending on the gate set [6]. Indeed, the synthesis procedure decomposes the compiled circuit into simpler supported operations [10] while adding some ancillary qubits to ensure the reversibility of the computation (embedding) [11]. Thus, the synthesis step can potentially apply optimizations to impact the qubits

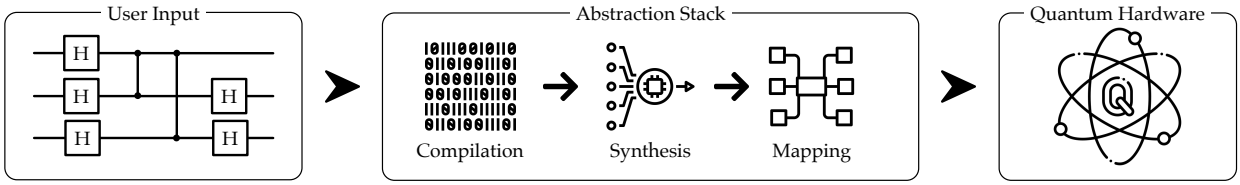

Fig. 1. Overview of the flow from the user-defined quantum circuit to its execution on quantum hardware

number and the fault tolerance of the algorithm [7]. Quantum logic synthesis algorithms are mainly split into functional and structural classes [11]. Functional synthesis algorithms require a pre-processing step to embed the non-reversible circuit into a reversible counterpart. Then, actual synthesis applies to this reversible circuit for qubit diminishing. Conversely, structural algorithms build a reversible circuit by decomposing the input structure into smaller elements, finding reversible circuits, and then bring up together again. However, they generally yield circuits with more qubits than functional synthesis [11].

Once synthesized, circuit mapping matches the components to the underlining quantum machine, such as the correspondence between logical and available physical qubits [10]. In the case of non-error-corrected quantum computers, mapping assigns a physical qubit to the logical qubit with a one-to-one correspondence. Instead, if the architecture features error correction, an ensemble of physical qubits composes a logical one. This one-to-many correspondence between logical and physical qubits is implemented through a series of SWAP gates to respect coupling constraints imposed by the device [11]. However, SWAPs are highly error-prone; therefore, minimizing the number of these gates is necessary to prevent the error from increasing [12]. Finally, we can apply more optimizations, e.g., perform gate cancellations, transformations, and commutations, and resynthesize circuit parts to make the computation even more efficient [10].

In this context, the mapping stage's optimization problems represent the main challenge within the pipeline. Finding the best resource mapping belongs to the NP-hard class of optimization problems, making them particularly challenging to solve [13]. They closely resemble classic Electronic Design Automation (EDA) optimization problems, therefore suffering from the same issues [14]. However, the expertise developed in the classical domain can help advance the quantum counterpart. We envision two paths where classical computing solutions inspire novel heuristics for quantum mapping, or knowledge in quantum will help in developing new EDA tools. Moreover, we envision more hardware-software codesign approaches to improve the overall quantum abstraction stack and push toward an increased abstraction level for quantum programming. All these techniques are unexplored in the State of the Art, but they embody exciting research directions.

III. Quantum Circuit Simulation and Emulation

Quantum Circuit Simulation (QCS) embodies a crucial component in the application development flow. It mimics the behavior of a real quantum computer on a classical device, considering the properties of the qubits and their interactions. QCS is essential for algorithms' formulation, as the number of qubits might exceed the currently-available quantum computer number. Moreover, QCS could validate new circuits, help evaluate the scalability of future larger quantum architectures [15], and overcome the issue of limited available quantum computers for the scientific community. Indeed, building and maintaining a quantum computer is costly, and companies offer limited access to their quantum computer or offer them as a pay-as-you-go service. In this context, QCS constitutes a way for researchers to access quantum computers only when necessary, considerably impacting the costs.

QCS mimics quantum mechanics properties (e.g., superposition and entanglement), making it remarkably memory- and compute–intensive. Storing all the quantum states' amplitudes and manipulating them according to gates to apply requires a significant computational power that general-purpose architecture cannot deliver, resulting in long execution times, scaling exponentially with the number of qubits [15]. For these reasons, many research efforts focused on leveraging heterogeneous architectures to speed up QCS.

The dense operations that act on qubits resemble tensor products and vector-matrix multiplications; hence, exploiting their intrinsic parallelism becomes paramount. Indeed, the research either exploits Graphics Processing Units (GPUs)' massively parallel structure or Field Programmable Gate Arrays (FPGAs) reconfigurable spatial fabric. On the one hand, many works propose to offload to GPUs the QCSs with promising results [16], [17]. Recently, NVIDIA released cuQuantum SDK [18], paving the way for the development of commercial-level libraries for QCS using GPUs. This library represents the backend for IBM's Qiskit Aer high–performance simulator [19], also providing multi-GPU support. Another valuable strategy to leverage GPUs for QCS consists of building and executing this task on a heterogeneous High-Performance Computing system to combine GPUs' computing power with the more extensive memory resources provided by Central Processing Unit (CPU)-based systems [20]. Instead, other works focus on optimizing the computation by implementing strategies to fully exploit the architecture's parallelism and increase GPU utilization [15]. However, memory requirements to represent qubits' states are still an issue in QCS on GPU.

On the other hand, FPGAs attain high performance by leveraging the reconfigurable spatial fabric combined with lower power consumption [21]–[25]. For this reason, the State

of the Art offers many simulators based on FPGAs. Existing FPGA-based quantum hardware simulators can be broadly divided into two categories: universal emulators [26], [27], and circuit-specific emulators [28]. The former category proposes implementing acceleration for generic quantum simulation operations, supporting the application of any combination of quantum gates. While this approach is more general and more suitable for fast prototyping novel quantum algorithms, it loses efficiency compared to circuit-specific hardware. The latter category relies on optimizing the target circuit to the utmost degree, leveraging algorithm-specific properties to reduce the number of operations to be performed, and optimizing the use of the memory bandwidth available on the device. While this approach may achieve the best simulation performance for a given circuit, the accelerator design effort limits this approach to a few algorithms, such as the Quantum Fourier Transform.

However, despite the promising results, these works still represent proof of concepts and not mature tools with support and a community using them. Furthermore, they feature some relevant drawbacks limiting practical usage. Resource availability represents the main constraint as it limits the maximum number of qubits that can be simulated, reaching up to 30 qubits [28] in the case of simulators tailored for a specific algorithm. In addition, data precision considerably impacts the performance in terms of execution time and accuracy of the provided solution. Based on the State of the Art presented, the next Section describes our interpretation of potential future research directions for Quantum Hardware Emulation (QHE).

A. Quantum Hardware Emulation from Ten-thousand Feet

The Quantum Strong Church-Turing Thesis states that only quantum machines can efficiently emulate quantum circuits [29]. However, without accessible and reliable quantum computers, researchers must find ways to emulate quantum circuits. Therefore, the question arises of determining which computer architecture represents the best candidate for QHE.

The problem of efficient QHE can be approached at different levels. To manage this complexity, we propose reconducting the problem of quantum circuit emulation to well-known fields of study in Computer Science, namely compiler optimization, Instruction Set Architecture (ISA) design, and emulation technology mapping[1]. The compiler optimizations targeting QHE operate on an intermediate circuit representation and could guide downstream decisions. For instance, supposing a very sparse dataflow graph circuit, the superior flexibility of an FPGA better emulates it. Conversely, if the graph is highly dense, a GPU better fits with a matrix multiplication kernel.

Differently, research at ISA-level should identify an expressive enough complete set of quantum gate operations, accounting for efficient hardware. This way, the compiler backend can focus on optimal gate selection. However, we also envision researchers investigating a middle-ground approach that offers a finite set of specialized gates as components of a QHE library. In this context, the emulated circuit gates do not necessarily correspond to the quantum gates modeled by the compiler. Instead, they may be the result of fusing several gates or decomposing them further, seeking to minimize the implementation cost of the circuit. A possible approach to realizing such a library is FPGA High-Level Synthesis technology. However, since implementation efficiency remains a primary concern, an HDL-based approach may be necessary.

Unlike implementing a more generic architecture, the main drawback of this approach is that the circuit must be synthesized from scratch whenever the circuit to emulate is changed. However, if the performance is considerably better, this might become negligible, and we propose also investigating mitigation approaches. For example, the reconfigurable fabric of FPGAs enables the implementation of quantum circuits with synthesis batches, and the time multiplexing would address resource-limited availability. Given the current limitations of quantum hardware, we envision interesting research opportunities investigating each of these problems.

[1]Emulation technology mapping maps models of quantum gates to digital circuits, fundamentally differing from quantum circuit technology mapping.

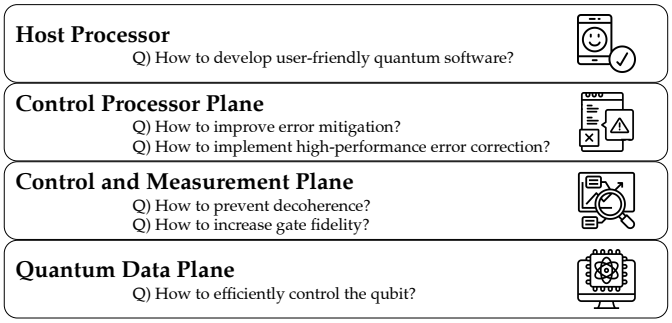

Fig. 2. Open research question for each plane in a quantum computer

IV. Quantum Hardware Stack

Assembling a reliable and usable quantum computer still represents the biggest challenge in quantum computing. Consequently, quantum hardware embodies a flourishing area of research, and noise mitigation represents a central topic due to the impact it could have on the entire field [30].

A quantum computer is a heterogeneous system, integrating classical hardware with the Quantum Processing Unit (QPU), the heart of the computation. We can highlight a similar structure in every quantum computer, regardless of the technology, and based on four layers, each with its challenges and open research questions [5] (Figure 2). The bottom layer contains the QPU, with its qubits and control logic. However, the description of this layer is out of the scope of this work as it is more related to physical science.

A. Host Processor

The Host Processor is a classical general-purpose computer that provides quantum software development tools. This component represents the interface enabling programmers to describe and execute quantum algorithms on a QPU. In this scenario, the State of the Art lacks integrated development

platform environments to ease the design of quantum applications. Furthermore, for the same reason, it is necessary to enhance libraries' usability and portability and expand the abstractions in programming languages [8]. Consequently, research efforts are directed in these directions to pave the way for future widespread usage of quantum technologies.

B. Control Processor Plane

According to the quantum algorithm to run, this level identifies the succession of quantum operations and measurements to execute the program and triggers it. This plane behaves as a bridge between the system's classical and quantum components, offloading the execution to the QPU [5], and it implements the error correction algorithm. Apart from readout decoherence and gate operations, errors may originate from different physical phenomena involving the qubit and its interactions, e.g., qubit relaxation and crosstalk [31]. There are different strategies to handle these sources of inaccuracies for Noisy Intermediate-Scale Quantum (NISQ) architectures, mainly divided into *error mitigation* [32] and *correction* techniques [33], [34].

Error mitigation strictly depends on the qubit's implementation and comprises a variety of approaches. The final correction applied is, however, minimal and performed as post-processing. In this scenario, research is currently focusing on Machine Learning (ML)- or Deep Learning-based techniques to help lower errors produced in NISQ devices, showing promising results [35], [36]. In addition, it is worth exploring the benefits of hardware acceleration for these approaches to avoid increasing overhead to quantum computation [31], [37].

Conversely, error correction targets errors at the QPU level, still involving, however, the Control Processor Plane. For example, one of the most popular strategies relies on the idea behind *repetition codes* [38]. A fabric of *physical* qubits encodes a single *logical* qubit, coupled with some auxiliary ones, called *ancilla qubits*. Ancilla qubits hold information about the state of errors, such that measuring them allows knowing which error affects which qubit. The Control Processor Plane captures this information through a decoder algorithm and adjusts the quantum computation accordingly to minimize the resulting error. However, the time required by the Control Processor for error decoding must not exceed a certain threshold (in a scale of hundreds of nanoseconds [5]) in order not to slow the next quantum operation. In this context, research directs efforts to two complementary approaches: on one side, the development of efficient decoding algorithms with lower computational complexity [39] and, on the other side, the acceleration of these algorithms through hardware architectures, e.g., GPUs, FPGAs, and others [40]–[42].

C. Control and Measurement Plane

This layer handles the analog-to-digital conversion, and vice versa, to enable communication between the classical hardware and the QPU. In one direction, it converts the digital signal indicating the operation to perform on the qubit into its analog counterpart acting on the qubit, i.e., the gate. In the other direction, it transforms the analog signal from qubit measurements into digital classical binary information (readout step), making it available for further processing.

The Control and Measurement Plane extensively uses FPGAs as they enable a tailored qubit control, thanks to their high flexibility [43]. Furthermore, FPGAs provide low latency, which is mandatory when reading the qubit's state to prevent it from decoherence and avoid inaccuracies in the results [44]. Besides readout, also gate operations open up higher chances for qubit's decoherence when modifying its state. These operations introduce errors during the computation, which can accumulate during the quantum circuit's execution [45]. Therefore, the result of the elaboration may significantly differ from the ideal desired state, as the probability of getting the correct results diminishes exponentially with the number of applied gates [5]. In this context, *gate fidelity* is a widely-used metric describing the behavior of a gate. It represents a gate's probability of modifying the qubits without introducing error and is often below 99%, unlike classical gates [46].

Therefore, performing the readout in a short time is not enough to prevent errors from occurring. High gate fidelity is also key to ensuring a higher likelihood of getting the correct result at the end of the computation. Also, it allows more effective error handling in the subsequent processing steps [46]. For these reasons, in this layer, the main research questions address the issue of how to implement gates to reduce inaccuracies. All the proposed solutions depend on the specific underlying technology of the qubit and target single- [47], [48] and multi-qubit [47], [49] gates. Further improving gate fidelity would unlock the possibility of going beyond NISQ architectures.

D. Quantum Random Access Memory

There is another research challenge that transcends the layers previously described. Memory components are fundamental elements in each computing device, and the ability to store data is essential. Therefore, also quantum architectures should comprise quantum counterparts of memory elements, such as a *quantum Random Access Memory (qRAM)*. Some preliminary attempts to build such a system exist in the State of the Art, as it is crucial to achieving a practical *quantum advantage*. Indeed, many algorithms proved the benefits of quantum computing but only supposing the existence of a qRAM [50]. Therefore, exploring such a system's feasibility is essential to fully exploit quantum computing's potential. Works in the literature model the qRAM with the same structure as a classical RAM, but the *address* and the *destination* registers are qubits instead. Even in this case, physical phenomena deteriorate the system's performance. Therefore, decoherence makes it necessary to consider implementing error-handling mechanisms to ensure correctness [51]. Also, sizing represents another issue, as large memories are not feasible under current physical assumptions [50]. There are different proposed strategies to tackle this last problem [50]. However, despite these first attempts, how to implement a qRAM efficiently remains an open question.

V. Quantum Computing Applications

Despite still being in its infancy, quantum computing has already proved to have the potential to revolutionize many big data applications, dramatically reducing processing time.

Optimization problems represent a large segment of applications already leveraging quantum technologies, relying on the annealing paradigm. Quantum annealing aims to solve classical combinatorial optimization problems, which generally rely on a cost function's minimization (or maximization). This process returns a *ground state*, which represents the solution to the optimization problem [52]. Numerous literature works illustrate the use of this computational paradigm in real-life scenarios [53], mainly reformulating NP-hard or NP-complete original algorithms following the Quadratic Unconstrained Binary Optimization (QUBO) formulation [54].

For what concerns more applicative domains, life sciences represent one of the most promising fields to prove a quantum advantage over traditional intensive computing methods [55]. The spreading of the COVID-19 pandemic again underlined the urgency to provide fast genomic analyses. However, genome analysis represents an intensive and time-consuming computation. Besides using classical hardware to speed up the process [56]–[58], quantum architectures appear to be a valid approach to pave the way for faster computations. In particular, some works in the literature have already successfully attempted to formulate the genome assembly problem as an optimization problem, leveraging the QUBO formulation and resorting to quantum annealing to perform the computation [59]. Another scenario in which quantum computing can have a considerable impact is sequence matching, where Grover's algorithm could substantially lower execution time [60]. However, despite a few attempts, the application of quantum computing in genomics still needs to be explored.

Drug design represents another exemplary use case for quantum computing [61]. Its tasks' computational complexity, which grows with the number of considered atoms [61], makes it untractable using classical architectures, slowing down the entire development process. In this context, researchers could leverage quantum computing to accelerate the most compute-intensive workhorses in drug development (e.g., drug-receptor interactions, protein folding, and binding site prediction), reaching even higher accuracy. Besides developing advanced quantum algorithms [62], solutions in the State of the Art exploit quantum ML-based techniques in heterogeneous frameworks to accelerate the drug design process [63]. However, the main issue is the available quantum hardware which prevents a practical usage of the developed solutions. Regardless of the specific application's field, more advanced hardware in memory and resources is necessary [55]. Indeed, this problem affects all disciplines more broadly related to molecular dynamics [64], which could substantially benefit from quantum acceleration for the same reasons.

VI. Conclusions

This manuscript reviews the State of the Art in quantum computing and identifies the most promising research topics. We analyzed EDA applications, QCS, and the various layers of quantum hardware, concluding with an overview of quantum computing's applications. We believe that this emerging technology has the potential to revolutionize many aspects of Computer Science. However, only cohesive research efforts can unleash the full potential of quantum computing. We provide exciting research topics on which unified efforts could result in enormous breakthroughs in quantum computing.

Acknowledgments

This work has financial support from ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU.

References

[1] J. Shalf, "The future of computing beyond moore's law," *Philosophical Transactions of the Royal Society A*, vol. 378, no. 2166, p. 20190061, 2020.

[2] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Communications of the ACM*, vol. 62, no. 2, pp. 48–60, 2019.

[3] M. Bozzo-Rey, J. Longbottom, and H. A. Müller, "Quantum computing: challenges and opportunities," in *Proceedings of the 29th annual international conference on computer science and software engineering*, 2019, pp. 393–394.

[4] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[5] N. A. of Sciences Engineering, Medicine *et al.*, "Quantum computing: progress and prospects," 2019.

[6] G. De Micheli, J.-H. R. Jiang, R. Rand, K. Smith, and M. Soeken, "Advances in quantum computation and quantum technologies: A design automation perspective," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 3, pp. 584–601, 2022.

[7] C. G. Almudever *et al.*, "The engineering challenges in quantum computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 836–845.

[8] M. A. Serrano, J. A. Cruz-Lemus, R. Pérez-Castillo, and M. Piattini, "Quantum software components and platforms: Overview and quality assessment," *ACM Computing Surveys*, 2022.

[9] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open quantum assembly language," *arXiv preprint arXiv:1707.03429*, 2017.

[10] L. Burgholzer, R. Raymond, and R. Wille, "Verifying results of the ibm qiskit quantum circuit compilation flow," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2020, pp. 356–365.

[11] A. Zulehner and R. Wille, *Introducing Design Automation for Quantum Computing*. Springer, 2020, vol. 11.

[12] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. Xie, "On the co-design of quantum software and hardware," in *Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication*, 2021, pp. 1–7.

[13] S. Raghunathan and L. Stok, "Eda and quantum computing: a symbiotic relationship?" *IEEE Design & Test*, vol. 37, no. 6, pp. 71–78, 2020.

[14] D. Paletti, F. Peverelli, and D. Conficconi, "Online learning rtl synthesis for automated design space exploration," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2022, pp. 69–76.

[15] Y. Zhao *et al.*, "Q-gpu: A recipe of optimizations for quantum circuit simulation using gpus," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 726–740.

[16] S. Efthymiou *et al.*, "Qibo: a framework for quantum simulation with hardware acceleration," *Quantum Science and Technology*, vol. 7, no. 1, p. 015018, 2021.

[17] D. Willsch, M. Willsch, F. Jin, K. Michielsen, and H. De Raedt, "Gpu-accelerated simulations of quantum annealing and the quantum approximate optimization algorithm," *Computer Physics Communications*, vol. 278, p. 108411, 2022.

[18] S. Stanwyck, H. Bayraktar, and T. Costa, "cuquantum: Accelerating quantum circuit simulation on gpus," *Bulletin of the American Physical Society*, 2022.

[19] (2021) Qiskit: An open-source framework for quantum computing.

[20] J. Doi, H. Takahashi, R. Raymond, T. Imamichi, and H. Horii, "Quantum computing simulator on a heterogenous hpc system," in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, 2019, pp. 85–93.

[21] E. D'Arnese, D. Conficconi, E. Del Sozzo, L. Fusco, D. Sciuto, and M. D. Santambrogio, "Faber: A hardware/software toolchain for image registration," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 291–303, 2022.

[22] D. Conficconi, E. Del Sozzo, F. Carloni, A. Comodi, A. Scolari, and M. D. Santambrogio, "An energy-efficient domain-specific architecture for regular expressions," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[23] R. Berzoini, E. D'Arnese, and D. Conficconi, "On how to push efficient medical semantic segmentation to the edge: the seneca approach," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2022, pp. 104–111.

[24] E. D. Sozzo, D. Conficconi, A. Zeni, M. Salaris, D. Sciuto, and M. D. Santambrogio, "Pushing the level of abstraction of digital system design: A survey on how to program fpgas," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–48, 2022.

[25] D. Parravicini, D. Conficconi, E. D. Sozzo, C. Pilato, and M. D. Santambrogio, "Cicero: A domain-specific architecture for efficient regular expression matching," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–24, 2021.

[26] M. Khalid, U. Mujahid, A. Jafri, H. Choi *et al.*, "An fpga-based hardware abstraction of quantum computing systems," *Journal of Computational Electronics*, vol. 20, no. 5, pp. 2001–2018, 2021.

[27] Y. Hong, S. Jeon, S. Park, and B.-S. Kim, "Quantum circuit simulator based on fpga," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2022, pp. 1909–1911.

[28] H. M. Waidyasooriya, H. Oshiyama, Y. Kurebayashi, M. Hariyama, and M. Ohzeki, "A scalable emulator for quantum fourier transform using multiple-fpgas with high-bandwidth-memory," *IEEE Access*, vol. 10, pp. 65 103–65 117, 2022.

[29] M. Kliesch, T. Barthel, C. Gogolin, M. Kastoryano, and J. Eisert, "Dissipative quantum church-turing theorem," *Physical review letters*, vol. 107, no. 12, p. 120501, 2011.

[30] A. D. Córcoles *et al.*, "Challenges and opportunities of near-term quantum computing systems," *arXiv preprint arXiv:1910.02894*, 2019.

[31] S. Maurya, C. N. Mude, W. D. Oliver, B. Lienhard, and S. Tannu, "Hardware efficient neural network assisted qubit readout," *arXiv preprint arXiv:2212.03895*, 2022.

[32] Z. Cai *et al.*, "Quantum error mitigation," *arXiv preprint arXiv:2210.00921*, 2022.

[33] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, 2019.

[34] R. Raussendorf, "Key ideas in quantum error correction," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, no. 1975, pp. 4541–4565, 2012.

[35] J. Kim, B. Oh, Y. Chong, E. Hwang, and D. K. Park, "Quantum readout error mitigation via deep learning," *New Journal of Physics*, vol. 24, no. 7, p. 073009, 2022.

[36] D. F. Wise, J. J. Morton, and S. Dhomkar, "Using deep learning to understand and mitigate the qubit noise environment," *PRX Quantum*, vol. 2, no. 1, p. 010316, 2021.

[37] P. Das, A. Locharla, and C. Jones, "Lilliput: a lightweight low-latency lookup-table decoder for near-term quantum error correction," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 541–553.

[38] J. M. Günther, F. Tacchino, J. R. Wootton, I. Tavernelli, and P. K. Barkoutsos, "Improving readout in quantum simulations with repetition codes," *Quantum Science and Technology*, vol. 7, no. 1, p. 015009, 2021.

[39] O. Higgott, "Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching," *ACM Transactions on Quantum Computing*, vol. 3, no. 3, pp. 1–16, 2022.

[40] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, "Qecool: On-line quantum error correction with a superconducting decoder for surface code," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 451–456.

[41] R. W. Overwater, M. Babaie, and F. Sebastiano, "Neural-network decoders for quantum error correction using surface codes: A space exploration of the hardware cost-performance tradeoffs," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–19, 2022.

[42] M. Hart, J. Mc Allister, L. Rogers, and C. Gillan, "An emulation of quantum error-correction on an fpga device," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2021, pp. 104–108.

[43] C. A. Ryan, B. R. Johnson, D. Ristè, B. Donovan, and T. A. Ohki, "Hardware for dynamic quantum computing," *Review of Scientific Instruments*, vol. 88, no. 10, p. 104703, 2017.

[44] C. Guo *et al.*, "Low-latency readout electronics for dynamic superconducting quantum computing," *AIP Advances*, vol. 12, no. 4, p. 045024, 2022.

[45] J. Wang, G. Guo, and Z. Shan, "Sok: Benchmarking the performance of a quantum computer," *Entropy*, vol. 24, no. 10, p. 1467, 2022.

[46] S. Resch and U. R. Karpuzcu, "Quantum computing: an overview across the system stack," *arXiv preprint arXiv:1905.07240*, 2019.

[47] C. Zhang, T. Chen, S. Li, X. Wang, and Z.-Y. Xue, "High-fidelity geometric gate for silicon-based spin qubits," *Physical Review A*, vol. 101, no. 5, p. 052302, 2020.

[48] Y.-C. Yang, S. Coppersmith, and M. Friesen, "High-fidelity single-qubit gates in a strongly driven quantum-dot hybrid qubit with 1/f charge noise," *Physical Review A*, vol. 100, no. 2, p. 022337, 2019.

[49] A. Kandala *et al.*, "Demonstration of a high-fidelity cnot gate for fixed-frequency transmons with engineered z z suppression," *Physical Review Letters*, vol. 127, no. 13, p. 130501, 2021.

[50] O. Di Matteo, V. Gheorghiu, and M. Mosca, "Fault-tolerant resource estimation of quantum random-access memories," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–13, 2020.

[51] C. T. Hann, G. Lee, S. Girvin, and L. Jiang, "Resilience of quantum random access memory to generic noise," *PRX Quantum*, vol. 2, no. 2, p. 020311, 2021.

[52] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: Methods and implementations," *Reports on Progress in Physics*, vol. 83, no. 5, p. 054401, 2020.

[53] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: Introduction and review," *Reports on Progress in Physics*, 2022.

[54] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics i: a tutorial on formulating and using qubo models," *Annals of Operations Research*, pp. 1–43, 2022.

[55] A. Fedorov and M. Gelfand, "Towards practical applications in quantum computational biology," *Nature Computational Science*, vol. 1, no. 2, pp. 114–119, 2021.

[56] B. Branchini, S. Breschi, A. Zeni, and M. D. Santambrogio, "Fast genome analysis leveraging exact string matching," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2022, pp. 136–139.

[57] B. Branchini, G. Gerometta, L. Cicolini, A. Zeni, E. Del Sozzo, and M. D. Santambrogio, "Surfing the wavefront of genome alignment," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 1754–1758.

[58] L. Di Tucci, D. Conficconi, A. Comodi, S. Hofmeyr, D. Donofrio, and M. D. Santambrogio, "A parallel, energy efficient hardware architecture for the meraligner on fpga using chisel hcl," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2018, pp. 214–217.

[59] K. Nałecz-Charkiewicz and R. M. Nowak, "Algorithm for dna sequence assembly by quantum annealing," *BMC bioinformatics*, vol. 23, no. 1, pp. 1–17, 2022.

[60] K. K. Soni and A. Rasool, "Pattern matching: a quantum oriented approach," *Procedia Computer Science*, vol. 167, pp. 1991–2002, 2020.

[61] M. Zinner, F. Dahlhausen, P. Boehme, J. Ehlers, L. Bieske, and L. Fehring, "Quantum computing's potential for drug discovery: Early stage industry dynamics," *Drug Discovery Today*, vol. 26, no. 7, pp. 1680–1688, 2021.

[62] B. Bauer, S. Bravyi, M. Motta, and G. K.-L. Chan, "Quantum algorithms for quantum chemistry and quantum materials science," *Chemical Reviews*, vol. 120, no. 22, pp. 12 685–12 717, 2020.

[63] B. Lau *et al.*, "Insights from incorporating quantum computing into drug design workflows," *Bioinformatics*, vol. 39, no. 1, p. btac789, 2023.

[64] F. Peverelli, D. Conficconi, D. B. Bartolini, A. Scolari, and M. D. Santambrogio, "Characterizing molecular dynamics simulation on commodity platforms," in *2022 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2022, pp. 65–78.