

d-fine



Taming Quantum Computers with High-Level Software Stacks



Content

1. Executive Summary Page **3**

2. A Brief History of Quantum Computing: From Lab to Business Page **3**

3. Quantum Computing as a Service: The State of Play for Hardware and Software Page **6**

4. Programming a Quantum Computer Page **6**

5. Conclusion Page **9**

1. Executive Summary

Quantum computing promises to disrupt any industry that relies on heavy-duty computation. Use cases of this new technology range from chemistry and materials design to machine learning and optimization in finance, energy, mobility, logistics, and pharma. Over the past few years, first generation quantum computers have become available to end users. While these early-stage machines still need to overcome technical difficulties before they can outperform classical high-performance computing, they are an invaluable tool for exploring this new technology. Quantum computing requires new corporate structures and skills because classical algorithms and development methods cannot simply be reused or easily adapted. To be ready in time companies need to start this learning process today while the technology continues to mature.

Many of the largest industry players, governments and venture capitalists world-wide have ramped up heavily their investments in quantum computing. This has resulted in rapid progress of hardware and software. On the software side, development tools for quantum computer programming have emerged. On the hardware side, Google's fully programmable quantum computer has demonstrated a computational advantage over the fastest supercomputer [1]. While the task they solved has no relevance for business use cases, these developments clearly demonstrate the prowess of even relatively small quantum computers.

The accelerating pace of the field means that some industry verticals could be affected sooner rather than later. It takes time and a deep understanding of the technology to acquire the skills necessary for an assessment and the implementation of potential use cases. First steps towards adoption are joining the growing quantum computing ecosystem and exploration with available early-stage quantum machines. We support you on this journey with our expertise in assessing potential use cases, building up skills, programming prototypes, and integration. For example, d-fine is a core member of PlanQK, a consortium of industry and academic partners developing a platform, ecosystem and use cases for quantum-assisted machine learning. This report provides you with an overview of quantum computing, the prevalent hardware access model, major quantum software frameworks, and concludes with a glimpse into programming quantum computers.

2. A Brief History of Quantum Computing: From Lab to Business

What kind of computer would it take to simulate all of physics? Speculation of this kind in the early 1980s in the US—and independently in the USSR— led to the concept of a quantum computer [2], [3]. Richard Feynman and Yuri Manin noticed that simulating quantum mechanical systems on a classical computer is infeasible. As the

system size grows this simulation would rapidly consume resources beyond any conceivable classical supercomputer. They proposed to solve this issue by using controllable quantum systems to simulate other quantum systems efficiently.

This train of thought was formalised in the concept of a universal quantum computer. Such a computer enables a much richer class of operations than its classical counterpart (see Info Box 1 – Sources of Quantum Advantage). Quantum computers can perform tasks a classical computer simply cannot perform. From the abacus to modern supercomputers we have exploited the same basic principles for computation over millennia. Quantum computers are the first departure from these basic principles.

Quantum computing rose to prominence after researchers proposed specific quantum algorithms in the mid-1990s that would outperform classical counterparts. However, exploiting their advantage requires a quantum computer. Perhaps the most famous quantum algorithm is Shor's algorithm for prime number factorisation. This algorithm breaks most of today's public key cryptography systems [4]. Further experimental and theoretical breakthroughs throughout the 1990s made quantum computing a realistic possibility by the early 2000s.

In recent years, the field has seen a remarkable boost in industrial backing. This growth has led to the emergence of a supporting ecosystem. Today Google, IBM, Microsoft, and others are pushing quantum computing with substantial funding and institutional muscle. An influx of venture capital has also galvanised a budding startup scene. These startups cover a wide range of services such as quantum hardware production, systems expertise, and software development. Industrial efforts are accompanied by an increasing focus on quantum technologies by national and supranational funding bodies.¹ These joint endeavours are bearing fruits as first working devices become available to industry partners and the wider public.

Present day quantum computing devices are still limited. They cannot, yet, deliver the computational speedups associated with fully fledged universal quantum computers. Today's devices are known as noisy intermediate-scale quantum (NISQ) devices [5]. They have two main drawbacks. First, NISQ devices are limited to, say, a few hundred qubits (see Info Box 1). A low qubit count limits the amount of information that can be extracted from computations. Second, hardware implementations of qubits are subject to severe engineering constraints, such as noise and the connectivity between qubits. As a result, quantum computations on NISQ devices are much more error prone than classical computations. Taken together these limitations mean that NISQ devices can only exe-

¹ Examples are the €1bn Quantum Flagship initiative by the EU, the \$1.2bn National Quantum Initiative Act in the USA, or the \$10bn National Laboratory for Quantum Information Sciences in China.

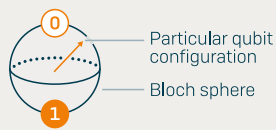
cut short algorithms. This limits the complexity and the type of problems they can solve. Once these limitations have been overcome, the upside of quantum computing is enormous. Info Box 2 exhibits some of the impacted areas. Depending on the task the upside includes a much

faster time-to-solution or better approximations leading to a competitive advantage. The enormous upside and the proliferation of NISQ devices has led to a world-wide rush among industry players for finding suitable real-world applications.

Info Box 1 - Sources of Quantum Advantage

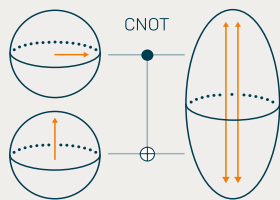
The basic principles of computation have not changed for millennia. Manual calculation with an abacus relies on the same principles as the current fastest supercomputer. Quantum computers perform their logical operations according to a different set of rules: quantum mechanics. Here are some of its key principles.

Basic Unit Information



The basic unit of classical information is a bit which is a binary state of '0' or '1'. The corresponding unit in quantum information is a qubit, which is represented by a 2-dimensional vector with values on the unit sphere. Its location on the sphere encodes information about the probability of a qubit being in state '0' or '1'. This information is called probability amplitude.

Entanglement



Entanglement is a purely quantum phenomenon. Two entangled qubits cannot be described by their individual states alone. They can only be described as a combined system. This causes strong correlations between different qubits: reading out one qubit immediately influences the outcome of the other qubit.

Interference

Similar to light, qubits can interfere with each other. Interference means that the probabilities of some outcomes can be boosted while probabilities of other outcomes can be reduced. Quantum algorithms rely on this effect to boost the probability of reading out correct solutions.

Quantum algorithms combine these concepts. Their interplay allows quantum computers to exhibit better performance for many applications. In contrast, classical algorithms cannot be reused on quantum computers.

Superposition

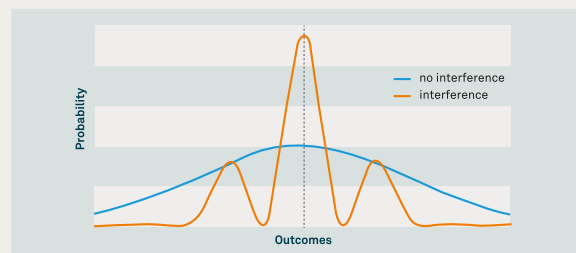


The qubit state is represented by a vector and can be rotated in any direction. This allows for configurations that have a probability of 50% for being one and 50% for being zero (or any other valid combination of probabilities). Such a superposition can be used for evaluating all possible values of a quantum function in a single execution.

Probabilistic Calculation



Quantum computation is probabilistic. Obtaining information from a quantum state requires reading out a (classical) value '0' or '1'. Reading out this classical value forces the quantum state to choose a definite state with a probability encoded in its probability amplitudes. Generally, a classical computer needs exponentially many resources to simulate such probabilities.



Info Box 2 - Potential Use Cases for Quantum Computing

In general, quantum computers excel at tasks such as the simulation of materials, optimization, search, and sampling. Given the right algorithm these tasks can be performed with a substantial speed boost on a quantum computer, leading to a faster time-to-solution or better approximations. However, most of these examples require large-scale quantum computers. The practical advantage of today's NISQ devices remains under heavy investigation by many academic and industry players.



■ **Artificial Intelligence and Machine Learning:** Artificial intelligence and machine learning has started to dominate both our daily life and our business environment. However, many of the hard problems consume enormous resources. Quantum computing could offer a more resource-effective way to tackle hard machine learning problems. It could also open up ways for learning about inherently quantum mechanical systems across several industry sectors.



■ **Chemistry & Materials:** The chemical & materials industry is expected to be an early beneficiary of quantum computing. Quantum computing offers a compelling route towards the efficient simulation of materials. This will lead to a better understanding of molecules and their chemistry and accelerate the discovery of new materials with innovative properties. Potential innovations are improved battery technology or reduced energy consumption for chemical production. Much research has been performed in this area and quantum algorithms already exist (e.g. nitrogen-fixation) that are waiting for quantum computers becoming sufficiently large for their execution.



■ **Financial Industry:** Portfolio optimization, asset allocation, risk management, or anomaly detection often depend on heavy-duty optimization and simulation algorithms. Quantum computing may provide a competitive advantage by ensuring faster time-to-solution or more accurate approximate solutions.



■ **Urban Mobility & Smart City:** The world-wide tendency towards urbanization and smart mobility will likely increase the adoption of heavy-duty optimization algorithms for complex traffic flows. Quantum computing could help improve the planning of urban mobility and smart cities. Additionally, it may help ensure the real-time operation of these systems.



■ **Supply Chain, Logistics & Energy:** Efficient logistics as well as supply chain and energy networks require solutions to complex optimization problems. One example is the cost-efficient planning and operation of energy grids with many constraints, such as caps on greenhouse gases or the availability of energy storage. Quantum computing can help reduce the time-to-solution or enable finer-grained network topologies that cannot be optimized efficiently with classical computers.



■ **Health & Pharma:** Identification, production and testing of new drugs is time-consuming and cost-intensive. Quantum computing can help discover new drugs more efficiently, e.g. by combining improved chemistry simulations and the use of machine learning tools.

3. Quantum Computing as a Service

The state of quantum hardware development bears similarity to the early days of classical computer development. While classical computers have largely settled on silicon semiconductor technology, quantum hardware manufacturers pursue several competing technologies. Currently, systems based on superconducting technology dominate the market. As of early 2020 these include the 53-qubit quantum computer “Sycamore” used by Google in their ground-breaking experiment [1], a 53-qubit computer announced by IBM, as well as a 32-qubit machine by startup Rigetti Computing. Alibaba Cloud, a subsidiary of Alibaba, has announced an 11-qubit quantum computer offering. Intel is working on a 49-qubit superconducting prototype, while also developing a second prototype using a different underlying technology, namely spin qubits. The Finnish startup IQM is developing a superconducting quantum processor. On the two issues that limit these devices, currently, superconducting qubits do well on the qubit count but are disadvantaged by large noise.

Ion trap qubit technology aims to improve on noise and connectivity of the qubits. In late 2018, IonQ has released impressive specs for their ion trap quantum computer. The Austrian startup AQT as well as Honeywell also pursue ion trap architectures. Microsoft works on topological qubits, which promise the lowest level of noise of all architectures. Unfortunately, this technology is not ready for general quantum computing, yet. The startups PsyQuantum and Xanadu use linear optics and single photons to build a quantum computer at room temperature.

An early pioneer, D-Wave, produces a 2000-qubit quantum annealer. This is a specialised computer geared towards certain optimization problems instead of universal computation.

IBM, Rigetti Computing, and D-Wave quantum machines are available to the public over the cloud.² The argument for offering quantum computing as a service are compelling:

- **Quantum computers are large and require expert maintenance.** For example, superconducting quantum computers operate at milli-Kelvin temperatures requiring liquid helium cooling and electromagnetic shielding.
- **Technological advancements are rapid.** For example, the number of qubits has steadily increased while error rates have decreased by orders of magnitude over the last few years. This means that these quantum machines would quickly need to be replaced with significantly more advanced machines.
- **Prototyping requires access to different hardware architectures.** Part of the current phase of prototyping is the comparison of different hardware architectures for a given potential application.

Thus offering a service, where quantum computers receive subtasks via the cloud and send back the results, seems to be beneficial in the near term. It also allows standardised software layers to develop on top of different hardware platforms. Such an abstract software layer can then decide on a suitable hardware platform for each sub-task. Consequently, first unified software ecosystems are starting to emerge.

4. Programming a Quantum Computer

All major hardware manufacturers have recognised that software development stacks are an essential pre-requisite for the adoption of quantum computing in non-specialist industry. Info Box 3 shows the major full-stack quantum software frameworks. “Full-stack” stands for an integrated software and hardware design providing different levels of abstraction. Many of these frameworks provide high-level libraries of ready-made quantum algorithms. They also provide ways to program quantum algorithms in a classical programming language such as Python. The lower layers provide compilers translating high-level instructions to native gate sets. Assembly-like languages describe low-level quantum operations. At the lowest level are quantum computer simulators running on classical hardware or physical quantum computers. Today’s frameworks largely follow open source licensing.³

In addition to the frameworks in Info Box 3, there are also vendor-independent software frameworks and ecosystems. Examples include Amazon Braket or the publicly-funded project PlanQK [7]. PlanQK aims at developing a platform and ecosystem for quantum-assisted artificial intelligence. Developers and experts will be able to provide quantum-assisted solutions to industry users through the platform. d-fine is a core industry member of PlanQK.

Quantum programming is a very young and rapidly evolving discipline. This means that developing quantum software is still quite different from developing classical software:

- **Hardware abstraction.** During the NISQ era, programming frameworks cannot fully abstract away idiosyncrasies of hardware implementations. For instance, the physical qubit layout often does not support the all-to-all connectivity required by general quantum algorithms.⁴





² Other manufacturers have announced plans for cloud access to selected partners. D-Wave’s system is also available for purchase.

³ For a review of open source quantum software, see [6].

⁴ This problem can be mitigated to some extent by intelligent compilers, which translate a quantum program for a given hardware layout.

Info Box 3 – Major Quantum Software Development Frameworks

All major quantum software development frameworks allow for coding quantum algorithms at different levels of abstraction. Code can be executed either on physical quantum processing units (QPUs) such as NISQ devices or on simulators of real QPUs. Libraries such as TensorFlow Quantum or PennyLane provide interfaces for seamless integration of quantum algorithms into machine learning frameworks such as TensorFlow or Pytorch.

	Rigetti Forest	IBM Qiskit	Google	Microsoft QDK	Xanadu
Quantum Libraries 	Grove	Qiskit Aqua	OpenFermion TensorFlow Quantum	Q# Chemistry Q# Numerics Q# ML	Strawberry Fields Apps PennyLane
Quantum Circuits 	pyQuil	Qiskit Terra Qiskit Ignis	Cirq	Q#	Strawberry Fields
Assembly Language 	Quil	OpenQASM			Blackbird
Hardware / Simulator 	Rigetti QPU Rigetti simulator	IBM QPU IBM simulator	Google QPU Google simulator	Microsoft simulator	Various simulators

- **Low-level programming.** Implementation of new quantum algorithms or fine-tuning of available algorithms is often at the level of individual quantum operations — equivalent to specifying individual logic gates for classical computation. However, current software frameworks often come equipped with several helper tools and functions for specifying higher-level building blocks.
- **Data encoding and type standards.** There are no standardized data encoding methods and types, yet. Quantum programmers need to choose an encoding most suitable for a given task or hardware. This is in contrast to classical computing where data types are highly standardized, e.g. the IEEE 754 formats for floating point numbers.
- **Software standards.** Often there are no formal standards between the various software frameworks. On the other hand, several frameworks have introduced plugin systems, which allow them to work with different quantum hardware at the lowest level, while providing a unified abstraction at a higher level.

- **Testing.** Simulating quantum computers is computationally demanding. Even with classical supercomputers, simulation of arbitrary quantum computers is only possible up to around 40-50 qubits (although this limit may be pushed for specific quantum algorithms).
- **Debugging.** Debugging quantum computers is hard because an observation will change the physical state. This means that the state during calculation cannot be inspected without destroying the calculation. So far there is no satisfactory way to debug quantum computations.
- **Verification.** A related issue is verifying the correctness of a quantum computation result. There is no straightforward way to compare results between quantum and classical computers. The tasks relevant for quantum computers are just not solvable on classical computers. This issue is still an active research field.

Info Box 4 gives a flavour of programming a quantum computer with the simple example of flipping two coins.

Info Box 4 - Programming a Coin Flip on a Quantum Computer

Alice and Bob have recently learned about quantum computing. Bob is intrigued by its intrinsically probabilistic nature but he is not quite sure how to exploit it. He challenges Alice to implement a coin flipping game with two coins—one coin for Alice and one coin for Bob. The result per flip should be random, either heads or tails, but both coins should always have the same outcome, either two heads or two tails. They agree on the rule that no coin is allowed to check the outcome of the other coin (this excludes, for example, “if” checks on outcomes or copying the results of one coin).



Alice fires up a browser on her phone and accesses Rigetti’s quantum computing system over the cloud. Alice remembers the superposition and entanglement principles of quantum computing. For superposition she applies the quantum operation “H” (Hadamard) on qubit 0. The first qubit is now in a state that resembles independent flips of the first coin. Then she exploits entanglement using the controlled NOT (CNOT) quantum operation on qubits 0 and 1. This step is uniquely quantum—the two qubits cannot be considered independently anymore. As a result, when Alice or Bob look at “their” coin, the two faces will always be the same—either two heads or two tails. In contrast to normal coins, it is not necessary to check or copy explicitly the state of the first coin for this outcome. In principle, Bob could even send “his” coin to the moon before they check results. The outcome would remain unchanged. Alice’s code “looks” at the coins via the `run_and_measure` method of the quantum device. This is repeated 10,000 times. The output of the coin flips are pairs of 0s or 1s (heads or tails). All 10,000 trials are perfectly correlated and each outcome occurs with approximately 50% chance. Bob is suitably impressed and vows to dive into quantum programming as well.

```
from collections import Counter
from pyquil import Program, get_qc
from pyquil.gates import H, CNOT

# Request a handle for a 2 qubit (simulated) quantum device on Rigetti's system.
device = get_qc("2q-qvm")

# Initialise a quantum program object to be executed on a quantum device.
program = Program()
# Create a superposition on qubit 0. Qubit 0 has a 50/50 chance to show 0 or 1.
program += H(0)
# Entangle qubits 0 and 1. Ensures that their outcomes are strongly correlated.
program += CNOT(0, 1)

# Run the program 10,000 times on the quantum device.
bitstrings = device.run_and_measure(program, trials=10000)
# Count the number of each outcome. (0, 0) is two heads, (1, 1) is two tails.
print(Counter(list(zip(*bitstrings.values()))))
# Output: Counter({(1, 1): 5018, (0, 0): 4982})
```


5. Conclusion

Until recently access to quantum computers was restricted to highly specialized research labs. Today several commercial manufacturers provide access to early quantum computing hardware over the cloud. An emerging software ecosystem allows interested industry players to assess this new technology by implementing concrete use cases for quantum computers.

Many discoveries in the early 20th century, such as the photoelectric effect by Einstein, paved the way for what is called the first quantum revolution. This has gifted us lasers and semiconductor hardware which revolutionized the way we process information and therefore do business. At present, we are in the middle of the second quantum revolution. Being able to program individual quantum states into usable algorithms promises to overhaul the way we do large-scale computing.

Quantum computers will not replace today's classical computers for everyday tasks. Email, word processing, and document storage are not improved by the advent of quantum computation. However, there are hard problems that quantum computers can speed up dramatically and there will be new lines of business only quantum computers can enable.

Similar to other new technologies in their early days, quantum computing still needs technical advances to fully realize its potential. However, some industries may see a meaningful quantum advantage become reality earlier than previously thought possible. While the technology matures, it is crucial for companies to assess the impact of quantum computing and plan for its adoption. d-fine can support you in building up skills, programming these devices, identifying potential use cases, and engaging with the wider quantum community.

References

- [1] F. Arute et al., 'Quantum supremacy using a programmable superconducting processor', *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.
 - [2] R. P. Feynman, 'Simulating physics with computers', *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, Jun. 1982.
 - [3] Y. I. Manin, *Computable and Uncomputable* (in Russian). Moscow: Sovetskoye Radio, 1980.
 - [4] P. W. Shor, 'Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer', *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.
 - [5] J. Preskill, 'Quantum Computing in the NISQ era and beyond', *Quantum*, vol. 2, p. 79, Aug. 2018.
 - [6] M. Fingerhuth, T. Babej, and P. Wittek, 'Open source software in quantum computing', *PLOS ONE*, vol. 13, no. 12, p. e0208561, Dec. 2018.
 - [7] PlanQK Consortium, *PlanQK - Platform and Ecosystem for Quantum-inspired Artificial Intelligence*. <https://planqk.de/en/> (accessed May 18, 2020).
-

Authors

DR MATTHIAS ROSENKRANZ
Manager, d-fine GmbH, Frankfurt
matthias.rosenkranz@d-fine.de

DR MARKUS BADEN
Manager, d-fine AG, Zürich
markus.baden@d-fine.ch

DR DANIEL HERR
Consultant, d-fine AG, Zürich
daniel.herr@d-fine.ch

DR BENJAMIN OBERT
Senior Consultant, d-fine GmbH, München
benjamin.obert@d-fine.de

d-fine

Berlin

d-fine GmbH
Friedrichstraße 68
10117 Berlin
Germany
berlin@d-fine.de

Dusseldorf

d-fine GmbH
Dreischeibenhaus 1
40211 Dusseldorf
Germany
duesseldorf@d-fine.de

Frankfurt

d-fine GmbH
An der Hauptwache 7
60313 Frankfurt
Germany
frankfurt@d-fine.de

London

d-fine Ltd
6-7 Queen Street
London, EC4N 1SP
United Kingdom
london@d-fine.co.uk

Munich

d-fine GmbH
Bavariafilmplatz 8
82031 Grünwald
Germany
muenchen@d-fine.de

Vienna

d-fine Austria GmbH
Riemergasse 14 Top 12
1010 Vienna
Austria
wien@d-fine.at

Zurich

d-fine AG
Brandschenkestrasse 150
8002 Zurich
Switzerland
zuerich@d-fine.ch