# Kipu Quantum's core technology

## Our path toward useful quantum computing

*White paper #1*
*Version: 1.0 – RELEASE*

By Kipu Quantum GmbH, 13 May 2023

# Kipu Quantum's core technology – our path toward useful quantum computing

Narendra N. Hegade, Eric Michon, Daniel Volz, and Enrique Solano
*Kipu Quantum GmbH, Roonstraße 23a, 76131 Karlsruhe, Ge*rmany, *info@kipu-quantum.com*

## 1. KIPU QUANTUM AND THE ZEITGEIST OF QUANTUM COMPUTING

Kipu Quantum is driven by the idea of realizing practical quantum computing applications in the near future. To this end, we develop application-specific quantum computing solutions with original and proprietary digital, analog, and digital-analog quantum computing paradigms. In this manner, we will bring quantum usefulness to the present so that a broad class of end users may find early quantum advantage in various industry use cases.

Kipu Quantum holds the deep conviction that, even with incremental improvement, contemporary quantum processors have a realistic perspective to reach the threshold for usefulness in the next few years. This slightly improved hardware must be matched by our advanced algorithmic compression and compact problem encoding as a prerequisite.

We are aware that this is a bold claim. In our view, algorithmic innovation via smart compression and encoding plays a crucial role in each industry use case. It essentially helps to

1. reduce the required circuit width, i.e., the number of qubits needed to encode/map industrially relevant problems.
2. reduce the required circuit depth, i.e., the number of algorithmic layers, each including simultaneously applied quantum gates.
3. replace logical/abstract one- and two-qubit gate operations, which are difficult to execute on real hardware, by native one-qubit, two-qubit, and multiqubit quantum operations, which are easier to execute in the quantum processor.

Kipu Quantum's core technologies play directly into these criteria. They have been invented and developed by our quantum engineers and experts with a pragmatic vision, accepting on our shoulders the creativity challenges and requesting from nothing to minimal hardware adaptations. We believe this is the pragmatic way to systematically approach a real possibility of useful quantum computing for industry use cases, such as combinatorial optimization for logistics and supply chain optimization, modeling of molecular and material properties, folding of proteins, and optimization of financial portfolios, among others.

## 2. KIPU'S CORE TECHNOLOGICAL PARADIGMS

The notion of creating novel, compressed algorithms adapted to current quantum processors lies at the bottom of everything we do at Kipu Quantum. With this, we avoid the considerable overhead in the number of physical qubits needed to engineer each logical qubit for the so-called fault-tolerant quantum computing (FTQC) paradigm. Actual quantum hardware cannot maintain an entangled state indefinitely; the limit is the coherence time of the qubits, which typically ranges between a few microseconds to the order of one second. Therefore, the longer each gate and corresponding circuit depth take, the fewer steps the algorithm can execute.

Kipu Quantum has developed two fundamental sets of compression and encoding to tackle the current hardware limitations directly. Along these lines, by being application and hardware specific instead of hardware-agnostic[1], we can improve the required circuit depth to solve problems by factors of ten to hundred and beyond. The core elements of our toolbox are given in **Table 1**. The following section briefly describes how our digital and non-digital compression works. For more details, please refer to further dedicated whitepapers.

Our **digital compression,** digitized-counterdiabatic quantum computing (DCQC), relies on an automatic workflow of three steps and starts with a mathematical problem based on an underlying industry use case:

- **Step 1**: encoding of the computational problem in the ground state of a predefined adiabatic quantum computing process[2]

1 Hardware-agnostic digital quantum algorithms
2 Adiabatic quantum computing is based on the concept of adiabaticity. To get the solution of a problem encoded as the ground

**Table 1: Breakdown of Kipu Quantum's current Toolbox**[3]

| Type of compression | Exemplary technique | Underlying idea | Exemplary improvement |
|---|---|---|---|
| Digital | Digitized-counterdiabatic quantum computing (DCQC) | Systematically reduce the number of required digital gates[4] | 10-100x reduction in the number of digital gates and circuit depths |
| Non-digital | Digital-analog quantum computing (DAQC) | Replace large sets of one-qubit and two-qubit gates by analog blocks, native in the quantum processors | N digital gates may be replaced by a single N-qubit pulsed gate, with N in the order of 10 to 100 |

- **Step 2**: apply the maximal acceleration of such adiabatic process by adding a minimal number of counterdiabatic terms[5]
- **Step 3**: the digitization of the accelerated analog quantum dynamics in a sequence of one-qubit and two-qubit gates to be executed on commercially available gate-based quantum computers

The workflow above relies on the computational equivalence between adiabatic QC and gate-based QC, so a maximally accelerated adiabatic dynamics can be mapped onto fewer gate operations. Therefore, DCQC provides an applications-specific, highly compressed digital quantum algorithm implemented in available gate-based quantum processors[6] .

Our **non-digital compression** follows multiple techniques. The underlying idea is to use so-called native **analog blocks**[7] to replace large numbers of otherwise needed digital one-qubit and two-qubit gates. Unlike digital gates, these analog blocks can encode tremendous complexity [8] while being reasonably easy for native execution in real quantum processors. In our non-digital compression, digital-analog quantum computing (DAQC), analog blocks and digital gates are used to build versatile algorithms that require a strongly reduced number of quantum steps and resources. DAQC algorithms are hardware-specific because the proposed and used analog blocks are unique and native to their respective hardware types.

We provide here some examples for the sake of clarity. Every quantum processor can implement multiqubit entangling operations that can encode a significant degree of complexity. For instance, we may use multiqubit gates with fixed or pulsed interaction times, instead of a large number of two-qubit gates to connect the qubits, applicable to trapped-ion, superconducting, or neutral-atom quantum processors. From an intuitive estimation, a N-qubit entangling operation may replace about N two-qubit gates. Applying DAQC on a quantum processor with hundreds of qubits, we may replace hundreds of two-qubit gates by a single analog block.

state of a complex problem Hamiltonian, one starts with an initial state which is the trivial ground state of an initial Hamiltonian and evolve the system towards the problem Hamiltonian as slowly as possible. If the evolution is slow enough, it can be considered as adiabatic and the system will stay in the instantaneous ground state, ending in the ground state of the targeted Hamiltonian.

3 List is not exhaustive.

4 Digital gates are one-qubit and two-qubit gates, such as single qubit rotations, CNOT, Hadamard, CZ, among others.

5 See [https://journals.aps.org/prapplied/abstract/10.1103/PhysRevApplied.15.024038]. Counterdiabatic terms are added terms which correspond to approximate adiabatic gauge potential. This approximate adiabatic gauge potential reduces the adiabatic computation time, minimizing the probability of ending in the wrong state.

6 We can follow purely-quantum or hybrid classical-quantum approaches. We can also design a strategy where the final digital quantum algorithm uses native one-qubit and two-qubit gates, considering the available connectivity of given QC architectures.

7 In analog quantum computation, neither time nor quantum operations are discretized. The system evolves continuously under a many-body Hamiltonian with entanglement being induced by the native interactions between qubits in given quantum processors.

8 This relies on tapping into the complexity encoded in multipartite entanglement. Multipartite entanglement is the simultaneous entanglement between multiple subsystems of a many-body quantum system. Multipartite entanglement in the system allows one to tap into the full potential of a quantum computer as the information that can be encoded becomes dramatically larger.

## 3. HOW KIPU'S TECHNOLOGY TOOLBOX CAN IMPROVE THE RESULTS OF TODAY'S HARDWARE

As claimed in our introductory section, we work under the conviction that our toolbox, combined with incremental hardware improvement, will bring about the singularity of practical and useful quantum computing much sooner, possibly in a few years, than fault-tolerant quantum computing (FTQC).

While this moment may not be as early as 2023, the benefits can already be tested. Some selected results are summarized in Table 2. While our tools certainly improve the outcomes by compressing circuit-depth requirements, there may be better metrics to demonstrate improvement. For example, the time to reach the solution may be prohibitive for real applications, or the result's quality may differ from the targeted one. Therefore, we use a more comprehensive set of application-relevant metrics. The figures of merit cited in Table 2 are defined in Table 3.

Scalability is also something to consider when looking at those figures of merit. Indeed, with the increase in the system size, the success probability will decrease exponentially. This means that the number of shots required will also dramatically increase, which will turn into a time-to-solution that may be unreasonable. The increase of several orders of magnitude in the success probability for smaller systems is a clear path to reduce the time-to-solution from a few years to possibly a few days in larger systems. In the latter, we may be ready to claim quantum advantage for the obtained result.

## 4. NEXT STEPS TOWARDS AND BEYOND USEFUL QUANTUM COMPUTING

At Kipu Quantum, we believe that algorithmic compression and smart problem encoding are necessary to approach the moment a quantum computer can demonstrate industrial usefulness for the first time in a given industrial problem.

Using the Kipu Quantum computing paradigms described above, DCQC and DAQC, to design application and hardware-specific algorithms and the incremental performance of today's quantum processors, will bring forward the era of industrial usefulness of quantum computers.

We foresee several instances following the first demonstration of QC usefulness, including a gradual expansion toward a consolidated technology for other industry use cases. In such a future, multiple hardware setups, like neutral atoms, superconducting circuits, and ion traps are meant to co-exist.

**Table 2: Select examples of tangible improvement using the Kipu Quantum toolbox**

| Problem | Figure of Merit | Enhancement | Reference |
|---|---|---|---|
| Many-body entangled state preparation | Success probability | ~100 fold increase against benchmark (digitized quantum annealing) | Many-body entangled state preparation |
| Combinatorial optimization applied to prime factorization | Success probability | ~10 fold increase of a pure-quantum algorithm against a hybrid-QC benchmark (QAOA) | Prime factorization 1<br>Prime factorization 2 |
| Optimization for protein folding | Number of iterations | ~2x improvement over hardware efficient ansatz | Protein folding |
| Portfolio optimization | Success probability | ~2x improvement over QAOA | Portfolio optimisation 1 |

**Table 3: Definition of the figures of merit used in Table 2**

| Figures of Merit | Definition | What is a good value? |
|---|---|---|
| Success probability | Probability of the system to be in the ground state of the problem Hamiltonian by the end of computation in one shot. The highest the success probability, the lower the time to solution. | 100% / 1 |
| Number of iterations | Minimal number of shots necessary to reach the solution. The lower the number of iterations, the lower the time to solution. | As low as possible |
| Time to solution | Minimal time required to reach the solution, calculated by multiplying the number of iterations times the refreshing time of the device, i.e., time to run the computation once. | As low as possible |