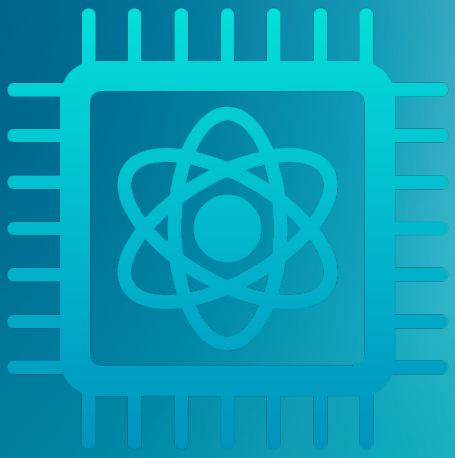
Fraunhofer FOKUS



Colin Kai-Uwe Becker | Ilie-Daniel Gheorghe-Pop | Matic Petrič | Nikolay Tcholtchev
*Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Deutschland*

# WebMarQC

## Eine Studie zu einem webbasierten Benchmarking Service für gatterbasierte Quantencomputer

# Impressum

**Kontaktadresse**
Fraunhofer Institut für Offene Kommunikationssysteme FOKUS
Kaiserin-Augusta-Allee 31, 10589 Berlin
Projektwebsite: https://webmarqc.fokus.fraunhofer.de

Prof. Dr. Nikolay Tcholtchev
nikolay.tcholtchev@fokus.fraunhofer.de

Colin Kai-Uwe Becker
colin.kai-uwe.becker@fokus.fraunhofer.de

**Autoren**

| | |
|---|---|
| Colin Kai-Uwe Becker | Fraunhofer FOKUS |
| Ilie-Daniel Gheorghe-Pop | Fraunhofer FOKUS |
| Matic Petrič | Fraunhofer FOKUS |
| Nikolay Tcholtchev | Fraunhofer FOKUS |


**Förderung**
Diese Arbeit wurde mit Mitteln des Bundesministeriums für Wirtschaft und Klimaschutz unter den Förderkennzeichen 01MQ24001 (*WebMarQC*) gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Bundesministerium
für Wirtschaft
und Klimaschutz

**Zusammenfassung**

Im Rahmen dieser Studie werden neben einer kurzen Einführung in das Benchmarken von gatterbasierten Quantencomputern auch aktuelle Herausforderungen und dazugehörige Lösungsansätze sowie der aktuelle Stand der Standardisierung in diesem Bereich beleuchtet. Mit dem im Rahmen des *WebMarQC*-Projekts entwickelten Prototyps einer über einen Webservice erreichbaren und bedienbaren Benchmarking Suite wird eine Möglichkeit zur Verwertung und Nutzbarmachung von Ergebnissen aus unterschiedlichen Benchmarking-Vorhaben vorgestellt. Durch die Möglichkeit, vorausgewählte Input-Parameter im Einklang mit Standardisierungsaktivitäten oder dem aktuellen wissenschaftlichen Konsens zu nutzen, diese in einem Benchmarking-Report zusammen mit weiteren aussagekräftigen Metadaten abzubilden und bei Bedarf vollständig auf Nutzerseite anzupassen, werden Impulse für die Vergleichbarkeit, Transparenz und Reproduzierbarkeit von Benchmarks gesetzt. Ferner werden parallelisierte beziehungsweise simultan ausgeführte Benchmarks beleuchtet und die Relevanz für NISQ-Hardware mit verfügbaren Qubits im höheren zweistelligen bis dreistilligen Bereich aufgezeigt. Schließlich werden eine Reihe von Handlungsempfehlungen formuliert, welche auf den bisherigen Entwicklungen aufbauend strategische Möglichkeiten zur Weiterverfolgung, Weiterentwicklung und Verwertung von bisherigen Ergebnissen aus unterschiedlichen Benchmarking-Vorhaben sowie der Standardisierung aufzeigen.

**Keywords:** Quantencomputing, Benchmarking, Benchmark Suite, Standardisierung

# Inhaltsverzeichnis

# 1. Warum ist die Performanzevalation von Quantencomputern relevant?

Das Benchmarken von Quantencomputern erfuhr in den letzten Jahren eine große Aufmerksamkeit, was sich in der Anzahl der mit diesem Bereich verwandten Projekten, Publikationen und daraus realisierten Benchmarking Tools widerspiegelt. Als Katalysator für diese Entwicklung sind die in den letzten Jahren realisierten Quantenvorteile zu nennen. Diese wurden durch das Benchmarken des *Sycamore*- und *Zuchongzhi*-Quantenprozessors [1–3] mittels zufallsgenerierter Quantenschaltungen, sowie zahlreicher darauf folgender Demonstrationen [4] und Fortschritte im Bereich der Quantenfehlerkorrektur gestützt [5, 6]. Diese initialen Meldungen, die nachfolgenden Demonstrationen möglicher Quantenvorteilen und das Erreichen bestimmter Meilensteine signalisieren, dass sich die experimentelle Umsetzung von Quantencomputing-Hardware fortlaufend verbessert und daher in naher Zukunft die Nutzbarmachung dieser Technologie für relevante Bereiche aus der Industrie, Wissenschaft und Wirtschaft ermöglichen könnte. Aufgrund dessen erfuhr das Quantencomputing in den letzten Jahren auch ein größeres Interesse aus Nutzersicht, Forschung und Entwicklung und ferner auch für Investoren, da erste Geschäftsmodelle rund um Quantencomputing-Hardware [1] und dazugehöriger Quantum-as-a-Service Plattformen [2], Softwareökosysteme und Dienstleistungen [3] möglich wurden. Mittlerweile befinden wir uns in einer Phase, in der die Ergebnisse der getätigten Investitionen und die Fortschritte der letzten Jahre beurteilt werden können. Häufig wird in diesem Zusammenhang der Begriff *Quantenwinter* genannt, welcher einen möglichen Einbruch von Investitionen im Quantencomputing beschreibt. Das hängt damit zusammen, dass möglicherweise der erreichte Fortschritt in der praktischen Anwendbarkeit und Skalierung von Quantencomputern hinter den anfänglichen Erwartungen liegen könnte. Vor allem in dieser Zeit kann ein faires, herstellerunabhängiges Benchmarken von Quantencomputing-Hardware zur Perfomanzevaluierung anwendungsorienterter Algorithmen und Kernmetriken ein möglichst transparentes und ehrliches Bild des aktuellen Stands zeigen und notwendige Entwicklungsfelder herausarbeiten, aber auch bisherige Erfolge und Entwicklungen in den letzten Jahren klar benennen und messbar machen.

Aktuell müssen sich interessierte Nutzer, sowie Akteure aus Industrie, Wissenschaft, Wirtschaft und Politik häufig auf von Herstellern durchgeführte Benchmarks und dazugehörige Kennzahlen verlassen, was bei zahlreichen unterschiedlichen Metriken und verschiedener Interpretationen der selben Metriken sowie unterschiedlicher Implementierungen der selben Benchmarks ein Problem der Vergleichbarkeit, Transparenz und Nachvollziehbarkeit aufwirft. Es gibt zwar durchaus eine Vielzahl an Publikationen, welche ein breiteres Feld an Benchmarkingmethoden und Ansätzen zeigen und diskutieren, allerdings sind diese in

[1] z.B. junge Hardwareanbieter wie *IQM*, *Quantinuum*, *AQT*, *SaxonQ* und *eleQtron*.
[2] z.B. *AWS Braket*, *Microsoft Azure* und *IBM Quantum*.
[3] z.B. *Munich Quantum Software Company*, *Classiq*, *Q-CTRL* und *PlanQK*.

der Regel nicht für Laien verständlich und für Endanwender aufbereitet. Potentielle Nutzer von Quantencomputern möchten wissen, welche Fähigkeiten und Vorteile diese bezogen auf unterschiedliche Anwendungsbereiche haben. Dafür möchten sie auch die verschiedenen Systeme unterschiedlicher Hersteller für diesen Anwendungsbereich sinnvoll miteinander vergleichen können und in der Lage sein, zu identifizieren, ob es ein geeignetes System für den geplanten Nutzungszweck gibt. Letzteres ist insbesondere auch von Interesse für potentielle Endanwender aus Industrie sowie Forschung und Entwicklung. Hier möchte man aktuelle Fähigkeiten verfügbarer Quantencomputing-Hardware messbar beurteilen können, um diese dann gezielt zu verbessern und zu nutzen. Investoren möchten Geschäftsentscheidungen auf Grundlage belastbarer Daten treffen und sich nicht auf Marketingaussagen verlassen.

Ein Benchmark sollte dafür unter anderem Antworten auf folgende grundlegende Fragen liefern (vgl. *DIN SPEC 91480* [7]):

**Performanzbeurteilung**

Mit welcher Güte kann eine QPU eine festgelegte Aufgabe erfüllen? Wann ist es wahrscheinlich, dass die Berechnung einer solchen Aufgabe fehlschlagen wird?

**Eignungsbeurteilung**

Welche QPU ist am besten für eine festgelegte Aufgabe geeignet? Steht eine QPU zur Verfügung, welche eine solche Aufgabe überhaupt berechnen kann?

**Entiwcklungsbeurteilung**

Wie kann man die bisherige und aktuelle Entwicklung von QPUs und Quantenalgorithmen in einem fairen und herstellerunabhängigen Rahmen quantifizieren?

**Investitionsbeurteilung**

Wie vielversprechend sind die verschiedenen QPUs? Wie hoch ist ihr entsprechendes Innovations- und Wirtschaftspotenzial?

Für die Performanzbeurteilung wurden in den letzten Jahren zahlreiche Benchmarks und dazugehörige Metriken definiert, welche sich grob in sogenannte **Mikro- und Makrobenchmarks** kategorisieren lassen. Erstere zielen dabei darauf ab, hardwarenahe Charakteristiken genau zu bestimmen und Letztere auf die Performanz des gesamten Quantencomputing-Stacks. Dabei kommen sowohl synthetische Benchmarks mit zufallsgenerierten und möglichst nicht-trivialen, höchst verschränkten Schaltungen sowie anwendungsorientierte Schaltungen mit Subroutinen bekannter Algorithmen oder vollständiger algorithmischer Implementierungen zum Einsatz. Klar ist, das Mikrobenchmarks relevante Daten für

die Weiterentwicklung der Hardware auf Qubitebene liefern. Dazu gehören unter anderem folgende Metriken:

**Kohärenzzeiten**

Kohärenzzeiten sind Zeitspannen, welche angeben, wie lange ein Qubit in einem stabilen Zustand bleibt, bevor es durch Wechselwirkungen mit seiner Umgebung gestört wird. Dabei betrachtet man zum einen die Zeit, in welcher ein Qubit vom angeregten $|1\rangle$-Zustand bis zu einem bestimmten Grenzwert in den $|0\rangle$-Zustand übergeht ($T_1$-Zeit). Zum anderen wird die Zeit betrachtet, in welcher ein Qubit bis zu einem bestimmten Grenzwert seine Phase verschiebt ($T_2$-Zeit). Längere Kohärenzzeiten bedeuten grundlegend einen größeren Zeitraum, in welchem ein Qubit zuverlässig in seinem Zustand gehalten oder manipuliert werden kann, ohne dass durch Dekohärenz erhebliche Fehlerquellen entstehen. Die Kohärenzzeiten werden in der Regel für jedes Qubit mit sogenannten *Spin-Echo-Experimenten* ermittelt und sind hardwarespezifisch und abhängig von der Kalibrierung der QPU.

**SPAM-Fehler**

*State Preparation and Measurement* Fehler oder kurz *SPAM*-Fehler kennzeichnen Fehlerraten beim Präperieren eines Anfangszustands für eine Berechnung (in der Regel der $|0\rangle^{\otimes n}$ - Zustand) und beim Messen des Endzustands. Diese beiden Fehlerquellen werden gemeinsam ermittelt, da man sie experimentell nicht voneinander entkoppeln kann. Möchte man die Fehlerraten eines Zustands ermitteln, muss man diesen schließlich messen und beeinflusst somit die Ergebnisse durch entsprechende Messfehler. Die genauen Messfehler hängen dabei in der Regel auch vom zu messenden Zustand ab. Durch sogenannte *Kalibrierungsexperimente* lassen sich SPAM-Fehler charakterisieren, was als Grundlage für die einfachste Form der *Quantum Error Mitigation* benötigt wird, nämlich der *Quantum Readout Error Mitigation (QREM)*. SPAM-Fehler sind hardzwarespezifisch und abhängig von der Kalibrierung der QPU.

**Gatterfehler**

*Quantengatter* oder kurz *Gatter* sind logische Grundoperationen von Quantencomputern, mit welchen man komplexe Quantenberechnungen durchführen kann. Jede QPU hat dabei einen nativen Satz an Gattern, welcher universelle Berechnungen ermöglicht. Das heißt, dass jede zulässige Quantenoperation durch eine bestimmte Kombination dieser Gatter ausgeführt werden kann. Die individuellen Fehlerraten dieser Gatter sind ein grundlegendes Leistungsmerkmal eines Quantencomputers, weil sich diese Fehler für jede Berechnung kumulieren. Dabei unterscheidet man in der Regel zwischen Gatterfehlern von 1-Qubit-Operation und Gatterfehlern von 2-Qubit-Operationen, welche eine deutlich größere Fehlerquelle darstellen. Gatterfehler sind hardwarespezifisch und abhängig von der Kalibrierung der QPU und werden in der Regel durch Fehlerraten oder der Fidelity bzw. Güte angegeben und durch Methoden des *Randomized Benchmarking* [8] ermittelt.

**Crosstalkfehler**

Crosstalkfehler kennzeichnen Fehlerquellen, welche nicht lokal wirken, sprich sich unbeabsichtigt auf benachbarte Qubits ausweiten. Dies passiert hauptsächlich durch Gatteroperationen und lokale Messungen, welche sich ungewollt auf nicht adressierte Qubits auswirken, kann aber auch im Leerlauf von Qubits entstehen. Diese Fehler sind hardwarespezifisch und unterscheiden sich deutlich zwischen verschiedenen Implementierungen von Qubits und werden in der Regel durch Fehlerraten angegeben und mittels simultan ausgeführten Randomized Benchmarking [9] und der sogenannten *Idle Tomography* [10] ermittelt.

**Error Per Layered Gate (EPLG)**

Zur Berechnung der Error per Layered Gate Metrik werden basierend auf der Qubit-Topologie der Zielhardware disjunkte Sätze an Qubits definiert, auf denen native 2-Qubit Gatter parallel ausgeführt werden können. Mittels Methoden des *Simultaneous Randomized Benchmarking* kann eine Fidelity bzw. Güte für jede Schicht an simultan ausgeführten Gattern ermittelt werden und daraus ein durchschnittlicher Fehler pro Gatter unter Beachtung von Crosstalkeffekten errechnet werden [11].

**Verschränkungsfähigkeit**

*Verschränkung* ist eine nicht-klassische Korrelation zwischen mehreren Quantenobjekten, welche unabhängig vom räumlichen Abstand zwischen den Quantenobjekten ist und nicht durch den Austausch klassischer Information erklärt werden kann. Erst durch die Verschränkung von Qubits kann ihr gesamter Zustandsraum (Hilbertraum) zugänglich gemacht werden und durch Quantenkorrelationen der Zustand vieler Qubits gleichzeitig manipuliert werden. Zusammen mit der Eigenschaft Superpositionen von Zuständen erzeugen zu können, ist die Generierung von Verschränkung ein zentraler Aspekt, welcher das Quantencomputing gegenüber klassischem Computing für bestimmte Anwendungen vorteilhaft macht. Daher ist die Fähigkeit einer QPU höchstverschränkte Zustände zu generieren und diese möglichst lang aufrecht erhalten zu können ein zentrales Leistungskriterium. Hierbei werden in der Regel verschiedene sogenannte *Verschränkungsmaße* und *Verschränkungszeugen* für bestimmte Klassen verschränkter Zustände angewendet und darauf basierend die Fähigkeit einer QPU verschränkte Zustände zu generieren evaluiert.

Ferner sind diese Daten essentiell für die Verwendung von Software zum Transpilieren und Optimieren von Schaltungen, da so die am besten geeigneten Qubits für eine Schaltung anhand der Konnektivität und Fehlerraten ausgewählt werden können[4]. Bezogen auf die anwendungsorientierte Performanz zeigt sich jedoch, dass die Fehlerraten sich dort nicht einfach als Summe der Fehlerraten der einzelnen Bestandteile beschreiben lässt, da das zugrundeliegende Quantenrauschen abhängig von der Breite, Tiefe, Struktur und Komplexität der Schaltungen sehr kompliziert zu beschreiben ist. Es reicht daher also nicht, sich lediglich auf die hardwarenahen Metriken zu fokussieren, sondern die Performanz des Gesamtsystems muss separat evaluiert werden. **Synthetische Benchmarks** zielen dabei in der Regel darauf ab eine holistische Metrik abzuleiten. Prominente Beispiele hierfür sind:

[4] In der Regel ist die Qualität der Qubits in aktuellen NISQ-Systemen ungleich über die QPU verteilt

**Cross Entropy Benchmarking (XEB)**

Beim *Cross Entropy Benchmarking* werden zufallsgenerierte Quantenschaltungen als Benchmarking-Workload erzeugt und wiederholt auf der Zielhardware ausgeführt. Die erhaltenen Messwerte werden durch die Berechnung der sogenannten *cross-entropy fidelity* mithilfe der idealen Messwertverteilungen ausgewertet. Dabei gibt ein minimaler Wert von 0 an, dass die Ergebnisse der Zielhardware gleichwertig mit dem zufälligen Erraten der Ergebnisse sind und ein maximaler Wert von 1 entspricht einer fehlerfreien Ausführung, deren Ergebnisse mit einer idealen Simulation übereinstimmen. Dies ist ein synthetischer Benchmark, da sich die Benchmarking-Workload nicht an der Struktur und an Routinen von algorithmischen Schaltungen orientiert. Das Benchmarkergebnis ist stark von der Qubit-Topologie der Zielhardware sowie den verwendeten Methoden zur Kompilierung und Optimierung der jeweiligen Schaltungen abhängig. Das Cross Entropy Benchmarking wurde bisher mehrfach für das Demonstrieren eines Quantenvorteils verwendet.

**Quantenvolumen**

Das *Quantenvolumen* ist eine holistische Metrik, welche häufig als Indikator für die Leistungsfähigkeit einer QPU verwendet wird. In der ursprünglichen Formulierung [12] wird das Quantenvolumen durch das Ausführen quadratischer Schaltungen bestehend aus zufällig generierten Schichten ermittelt. Quadratisch bedeutet hierbei, dass die Schaltungsbreite (also die Anzahl an Qubits) der Schaltungstiefe (also der Anzahl zufallsgenerierter Schichten) entspricht. Nach Ausführung der Schaltungen auf einer Zielhardware, werden die gemessenen Ergebnisverteilungen mit idealen simulierten Verteilungen verglichen und mittels der sogenannten *heavy output probability* ausgewertet. Durch ein binäres Kriterium in Form eines Grenzwertes wird anschließend entschieden, welche Schaltungen erfolgreich und welche nicht erfolgreich auf der Zielhardware ausgeführt werden können. Die Breite bzw. Tiefe der größten noch erfolgreich ausführbaren Schaltung kennzeichnet dann das Quantenvolumen. Dies ist ein synthetischer Benchmark, da sich die Benchmarking-Workload nicht an der Struktur und an Routinen von algorithmischen Schaltungen orientiert. Das Benchmarkergebnis ist stark von der Qubit-Topologie der Zielhardware sowie den verwendeten Methoden zur Kompilierung und Optimierung der jeweiligen Schaltungen abhängig. Eine Erweiterung des Quantenvolumen-Benchmarks auf Kombinationen beliebiger Schaltungsbreiten und - tiefen wurde in [13] vorgeschlagen, wodurch die Auswertung von Schaltungen geringer Breite und großer Tiefe und großer Breite und geringer Tiefe ermöglicht werden.

**Circuit Layer Operations per Second (CLOPS)**

*Circuit Layer Operations per Second* oder kurz *CLOPS* ist eine Metrik für die Geschwindigkeit einer QPU und quantifiziert die Anzahl der zufallsgenerierten Schichten angelehnt an dem Quantenvolumen-Workload, welche die QPU pro Sekunde ausführen kann [14]. Dabei werden auch klassische Computingprozesse in die Metrik mit einbezogen, wie das Transpilieren und Optimieren der Schaltungen, sowie Verarbeitungs- und Steuermethoden für das Hintereinanderausführen mehrerer Workloads, welche in Abhängigkeit von dem Ergebnis der vorherigen Schaltung dynamisch angepasst wer-

den. Da sich die zugrundeliegende Benchmarking-Workload sehr an der des Quantenvolumen-Benchmarks orientiert, ist diese ebenfalls als synthetischer Benchmark einzuordnen.

Die durch synthetische Benchmarks erhaltene Leistungsbeurteilung lässt sich jedoch nicht eins zu eins auf die Performanz eines beliebigen Algorithmus übertragen, da hier die ausgeführten Schaltungen in der Regel anders strukturiert sind und entsprechend ein anderes Quantenrauschen erzeugen. Daher sind **anwendungsorientierte Benchmarks** von großem Interesse. Es gibt allerdings für bisher betrachtete Problemgrößen durchaus Daten, welche einen Zusammenhang zwischen dem Quantenvolumen und bestimmten anwendungsorientierten Benchmarks vermuten lassen [15]. Unter anwendungsorientierten Benchmarks versteht man dabei zum Beispiel die unmittelbare Beurteilung der Lauffähigkeit bestimmter relevanter Algorithmen, beispielsweise aus den Bereichen der Quantenchemie, Materialwissenschaften und Methoden zum Berechnen von Finanz- und Risikomodellen. Hier können einzelne Algorithmen mit einer passenden Metrik ausgewertet werden oder eine kombinierte Metrik gewählt werden, welche die Benchmarks verschiedener Algorithmen mit einer definierten Gewichtung in einer Metrik zusammenfasst.

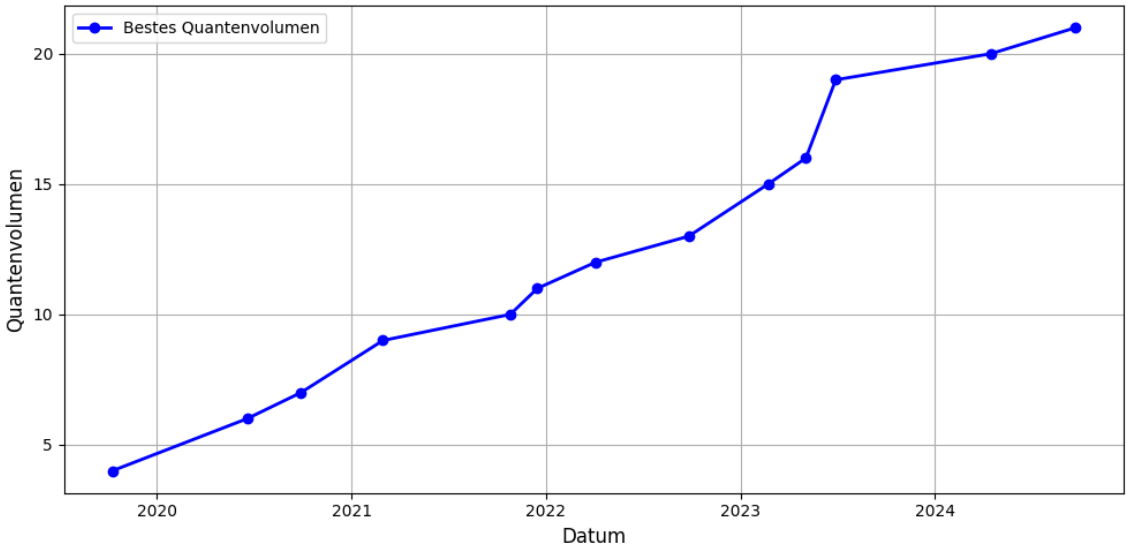



***Abbildung 1:*** *Entwicklung der jeweils besten Ergebnisse des Quantenvolumen-Benchmarks im Zeitraum von 2020 bis 2024. Die Angaben der Hersteller sind hierbei nicht abgebildet, da lediglich eine allgemeine Entwicklungstendenz gezeigt werden soll.*

Für die Entwicklungsbeurteilung ist es erforderlich, eine gewisse Datenmenge an durchgeführten Benchmarks über einen Zeitraum von mehreren Jahren zu betrachten, um verschiedene Entwicklungsindikatoren erkennen zu können. Eine erste Initiative, um solche Daten für verschiedene identifizierte Benchmarkszenarien verfügbar zu machen und zu visualisieren ist *Metriq* [5]. Die Daten stammen dabei meist aus publizierten Ergebnissen von Herstellern und Wissenschaftlern, was einerseits ein breites Spektrum an Daten ermöglicht, andererseits aber auch Fragen über die Vergleichbarkeit und die gemeinsame Interpretation der Daten aufwirft. Das Benchmarken von Quantencomputern ist bisher noch nicht weitreichend standardisiert, entsprechend gibt es keine einheitlich festgelegten Metriken

[5] https://metriq.info

mit dazugehörigen Benchmarkspezifikationen, sowie Regeln, wie die erhaltenen Ergebnisse transparent veröffentlicht und vergleichbar gemacht werden können. Erste Schritte für die Standardisierung von Quantencomputing-Benchmarks werden genauer in Abschnitt 2.2 erläutert. Ferner sind die Daten auch noch unvollständig und es bedarf einer eingehenden Kategorisierung und Sortierung. Untenstehend ist ein Entwicklungsindikator für das Quantenvolumen auf Grundlage in *Metriq* gesammelten Daten abgebildet. Allerdings sollten die Ergebnisse aufgrund der beschriebenen Unklarheiten zur Richtigkeit und Vergleichbarkeit entsprechend mit Bedacht interpretiert werden.

# 2. Aktueller Stand - Benchmarken gatterbasierter Quantencomputer

## 2.1. Aktuelle Herausforderungen

Im folgenden sind aktuelle Herausforderungen aufgelistet, die wir zum Benchmarken von Quantencomputing-Hardware in der NISQ-Ära identifiziert haben und die im Rahmen zukünftiger Projekte und Standardisierungsvorhaben angegangen werden sollten. Diese behandeln unter anderem den Zugriff auf QPUs über QaaS-Anbieter und die damit verbundenen Kosten, den Status von aktueller für das Quantencomputing relevanter Software, Hardwareunterschiede verschiedener QPUs sowie das anwendungsorientierte Benchmarking. Jede Herausforderung wird kurz beschrieben, gefolgt von möglichen Lösungsansätzen, die dazu beitragen können, diese Herausforderungen zu bewältigen.

> **Herausforderung 1**
>
> Das Entwickeln und Ausführen von Benchmarks ist teuer, wodurch es schwierig ist praktische Erfahrung sammeln zu können.

Einen praktisch nutzbaren, aussagekräftigen und möglichst skalierbaren Benchmark für aktuelle gatterbasierte Quantencomputing-Hardware zu entwickeln, erfordert das oft langwierige Experimentieren auf entsprechender Hardware. Im besten Falle werden dabei möglichst viele verschiedene Hardwareplattformen getestet, um die feinen Unterschiede zu verstehen und die Benchmarks somit möglichst fair und herstellerunabhängig zu gestalten. Allerdings ist genau dieser Hardwarezugang abseits von kleineren kostenlos angebotenen Laufzeiten-Kontigenten, in der Regel mit erheblichen Kosten verbunden. Dies erschwert es vor allem privaten Nutzern aber auch Universitäten und Institutionen genügend QPU-Laufzeit für das Experimenten mit und dem Durchführen von Benchmarks nutzen zu können. Somit fehlt abseits der Hardware-Anbieter oft an Möglichkeiten, das entsprechende Know-How und die Erfahrung zum Benchmarken von Quantencomputern aufzubauen. Dadurch werden aktuell (zum Stand der Veröffentlichung der Studie) relevante Metriken deutlich seitens der Hersteller dominiert. Hier besteht jedoch immer die Gefahr, dass eine objektive Beurteilung, gewinnbringenden Marketingzwecken weichen muss und eventuell an den Interessen der Nutzer vorbeigehen, was für faire und herstellerunabhängige Benchmarks nicht unbedingt förderlich ist.

> **Lösungsansatz 1**
>
> Es müssen praxisnahe Benchmarks mit einer möglichst geringen QPU-Laufzeit und möglichst wenigen auszuführenden Schaltungen entwickelt werden, welche auch mit einer statistisch kleineren Anzahl an Samples aussagekräftige Ergebnisse liefern.

Als ein mögliches Beispiel für diesen Lösungsansatz, möchten wir hier eine Methode zur Verschränkungscharakterisierung [16] nennen, welche mit nur zwei Schaltungen und einer moderaten Anzahl an Wiederholungen alle Qubits einer gitterähnlichen Topologie auf ihre Verschränkungsfähigkeit prüfen kann.

**Herausforderung 2**

Software-Frameworks um Benchmarks zu entwickeln und auszuführen sind in der Regel noch in frühen Entwicklungsstadien, was häufig dazu führt, dass entwickelter Code aufgrund regelmäßiger Updates mit grundlegenden Umstrukturierungen dieser Frameworks nicht mehr lauffähig ist. Dies gilt auch ganz allgemein für die Implementierung von Quantencomputing-Anwendungen.

Das frühe Stadium der verfügbaren Software-Frameworks macht es schwierig, entwickelte Benchmarks in einem lauffähigen Zustand zu halten. Häufig führen nicht mehr unterstützte Module (sogenannte *deprecations*), unterschiedliche Modulnamen (sogenannte *namespaces*) und grundlegend veränderte Workflows beim Simulieren und Ausführen von Quantenschaltungen sowie bei der Anwendung von Error Mitigation-Methoden dazu, dass oft bereits wenige Monate alter Code nicht mehr fehlerfrei ausführbar ist. Auch das Abrufen und Verarbeiten der erhaltenen Messergebnisse wird dadurch erschwert. Dies wird im allgemeinen umso schwieriger, je mehr Software-Frameworks für das Ansteuern von Hardware unterschiedlicher Hersteller verwendet werden muss und ist oft ohne speziell dafür geplantes und ausgebildetes Personal schwer zu bewerkstelligen. Ein prominentes Beispiel sind zahlreiche grundlegende Veränderungen in der *Qiskit SDK*, welche bereits in einer frühen Version im Jahr 2017 veröffentlicht wurde und seitdem kontinuierlich im Co-Design mit Verbesserungen der Quantencomputing-Hardware weiterentwickelt wird. In früheren Versionen vor dem Release von *Qiskit 1.0* waren dies zum Beispiel maßgebliche Umstrukturierungen wie der Wegfall von *Qiskit Aqua*, die Einführung der *Qiskit Runtime* und dazugehörigen *Runtime Primitives* wie dem *Sampler* und dem *Estimator* und die Einführung von nativen *Quantum Error Mitigation* Methoden. Ab Versionen nach dem Release von *Qiskit 1.0* gab es entsprechend wieder erhebliche Umstrukturierungen. Auch Frameworks, welche Herstellerneutralität erreichen wollen, wie etwa *Eclipse Qrisp* müssen die Herstellerneutralität über entsprechende Interfaces zu den jeweiligen Backends ermöglichen, wobei zum Ansteuern dieser Backends häufig ebenfalls Software in frühen Entwicklungsstadien verwendet werden muss.

**Lösungsansatz 2**

Es werden einheitliche Schnittstellen zwischen Quantencomputer-Backends und den Software-Frameworks benötigt, welche die Backends ansteuern. Dies erleichtert die Herstellerneutralität und das Ausführen von Quantenschaltungen über einen einheitlichen Prozess ohne dabei von herstellerspezifischer Software abhängig zu sein. Ein aktuelles nationales Vorhaben zur Ermöglichung einer solchen Schnittstelle ist die *DIN SPEC 91520*[a], welche sich aktuell in der Erarbeitung befindet.

[a] Geschäftsplan der DIN SPEC 91520: https://www.din.de/de/forschung-und-innovation/din-spec/alle-geschaeftsplaene/wdc-beuth:din21:381233217, zuletzt besucht am 15.10.2024

**Herausforderung 3**

Hardwareunterschiede zwischen verschiedenen Qubit-Technologien erschweren die Entwicklung von fairen und herstellerunabhängigen Benchmarks.

Verschiedene Qubit-Technologien, wie beispielsweise supraleitende Qubits, auf Ionenfallen basierende Qubits, Spin-Qubits und auf Photonen basierende Qubits, unterscheiden sich signifikant in ihrer physischen Umsetzung, den Kontrollmechanismen und Betriebsbedingungen. Dies hat einen Einfluss auf charakteristische Merkmale wie Kohärenzzeiten und die Fehlerraten für die Ausführung von Gattern, aber auch auf die Qubit-Topologie und die Art und Weise, wann und wie sich Qubits messen lassen. Dadurch kann sich die Art und Weise der Kompilierung und Optimierung von existierendem Code auf Ebene der Quantenschaltungen zu maschinenlesbaren Code zwischen den verschiedenen Technologien signifikant unterscheiden. Die Optimierung einer Schaltung in Abhängigkeit der nativen Gatter und Konnektivität, sowie der verfügbaren Operationen wie *mid-circuit measurements* und dynamische *if*-Abfragen [6] erfolgt dabei in der Regel für jede Hardware mit einer darauf angepassten Methodik, um die Hardware optimal zu nutzen und die Effizienz und Genauigkeit der Algorithmen zu maximieren. Eine ineffiziente Kompilierung kann zu längeren Gatter-Sequenzen und nicht optimalen Qubit-Layouts führen, die die Fehlerwahrscheinlichkeit erhöhen und die Kohärenzzeit belasten. Zusätzlich ermöglichen unterschiedliche Qubit-Technologien auch fundamental unterschiedliche Paradigmen zur Umsetzung eines Algorithmus - zum Beispiel neben der gängigen gatterbasierten Ausführung, die Ausführung mittels Methoden des *Measurement Based Quantum Computing* (MBQC) [17] oder *Quantum Annealings* [18], wodurch die zu lösende Problemstellung von Grund auf anders angegangen wird. Zukünftig können auch *Qudits* [19] mit mehr als zwei diskreten Energieniveaus, topologische Qubits [20], sowie analoge Quantenzustände, beispielsweise durch *continuous quantum variables* [21]eine größere Rolle spielen, was die Bandbreite der verfügbaren Technologien und somit auch der Umsetzungsmöglichkeiten für Quantenalgorithmen weiter vergrößert. All dies stellt eine Herausforderung dar, um einheitliche Benchmark-Metriken zu entwickeln, die objektiv und unabhängig von der spezifischen Hardware bestimmte Performanzkriterien bewerten können, was den Vergleich zwischen verschiedenen Technologien erschwert und herstellerunabhängige Bewertungen komplex macht.

**Lösungsansatz 3**

Es werden Spezifikationen für Benchmarks benötigt, welche eine gemeinsame Basis für sogenannte *Benchmark run rules* beinhalten, auf deren Basis die Benchmarks auf unterschiedlicher Hardware ausgeführt werden sollen. Diese Regeln sollten eine größtmögliche gemeinsame Basis für die Kompilierung und Optimierung der Benchmark-Workloads abbilden und dennoch Raum für technologiespezifische Optimierungen und Ausführungsmethoden lassen. Dies wird sich in der Umsetzung von Benchmark zu Benchmark wahrscheinlich stark unterscheiden.

[6] Normalerweise werden Messungen *as-late-as-possible (ALAP)* ausgeführt und somit am Ende der Schaltungen geplant. Messungen inmitten der Ausführung einer Schaltung können für dynamische *if*-Abfragen während der Quantenberechnung oder für dynamische Fehlerkorrektur verwendet werden, sind aber schwieriger umzusetzen aufgrund der Dauer der Messpulse und Crosstalk-Effekte.

**Herausforderung 4**

Viele Algorithmen können noch nicht auf aktueller NISQ-Hardware für nicht-triviale Problemgrößen ausgeführt werden, wodurch es schwierig ist, relevante und aussagekräftige anwendungsorienterte Benchmark-Workloads zu entwickeln.

Algorithmen wie etwa Shor's Algorithmus [22] zum quanten-klassichen Berechnen von Primfaktorzerlegungen oder auch der HHL Algorithmus [23] zum Lösen linearer Gleichungssysteme auf einem Quantencomputer sind auf aktueller NISQ-Hardware noch nicht oder nicht für relevante Problemgrößen ausführbar. Dennoch interessieren sich Nutzer bereits jetzt, wie nah wir aktuell an einer relativ fehlerfreien Ausführung solcher Algorithmen sind und in welchem Zeithorizont man damit rechnen kann, dass die Weiterentwicklung der Quantencomputing-Hardware eine solche Ausführung ermöglicht. Dafür benötigt man ausführbare Benchmark-Workloads, welche bestimmte Charakteristiken wie algorithmische Subroutinen, die generelle Struktur sowie spezielle Gatter eines Algorithmus enthalten.

**Lösungsansatz 4**

Algorithmische Building Blocks können identifiziert und einzeln sowie hintereinander und mit unterschiedlichen Input-Zuständen ausgeführt werden, um die Fähigkeit einer QPU entsprechende Algorithmen ausführen zu können vereinfacht evaluieren zu können. Dies ist detaillierter in der DIN SPEC 91480 [7] beschrieben.

## 2.2. Standardisierungsaktivitäten

Für einen Benchmark wird die fundierte Definition zentraler Leistungsmerkmale benötigt, welche mit diesem evaluiert werden sollen. Um diese Leistungsmerkmale berechnen zu können, müssen Messdaten aus der Ausführung von Benchmark-Workloads, welche auf Basis relevanter Anwendungsgebiete definiert werden, auf der Zielhardware gesammelt und verarbeitet werden. Dies geschieht in der Regel nach einem gewissen Protokoll, welches die Methodik zur Generierung der Workloads, zum Messen der Ergebnisse und zu deren Verarbeitung beschreibt. Dabei kann es eine Vielzahl an Freiheiten z.B. in der Wahl von Input-Parametern oder der spezifischen Ausführung der Workload auf unterschiedlicher Zielhardware geben, was die genaue Interpretation und den Vergleich von Benchmarkergebnissen erschwert.

Um vergleichbare, transparente, faire und herstellerunabhängige Benchmarks zu definieren, welche von Experten und der Community akzeptiert werden, ist die Standardisierung ein möglicher wichtiger Schritt. Als erster Schritt ist hier die *DIN SPEC 91480 - Benchmarken von Quantencomputern mit festgelegten KPIs* [7] zu nennen. Das Dokument kann weiterhin auch als Grundlage für Standardisierungsvorhaben auf internationaler Ebene, etwa in CEN/CENELEC und weitergehend ISO und IEC dienen. Ferner gibt es die Initiative des *IEEE P7131 - Standard for Quantum Computing Performance Metrics & Performance Benchmarking* [7], welche sich aktuell in der Erarbeitung befindet.

In [24] und [25] werden zu erfüllende Kriterien für die Standardisierung von Benchmarks definiert und in [26] sowie [7] speziell im Kontext des Quantencomputings eingeordnet. Nachfolgend sind diese Kriterien kurz zusammengefasst:

**Relevanz**

Die Relevanz bezeichnet das Maß inwieweit ein Benchmark sinnvolle und anwendbare Informationen für die Nutzer liefert. Ein relevanter Benchmark hat somit einen klaren Zweck und ist entsprechend für diesen konzipiert. Dies kann etwa die ganzheitliche Leistungsbeurteilung einer QPU oder eines spezifischen Anwendungsbereichs sein. Für die ganzheitliche Leistungsbeurteilung sowie für die Leistungsbeurteilung verschiedener voneinander abgegrenzter Problembereiche sind somit in der Regel verschiedene (Kombinationen von) Metriken und Benchmarks relevant.

**Reproduzierbarkeit**

Die Reproduzierbarkeit bezeichnet das Maß inwieweit ein Benchmark bei wiederholter Ausführung kohärente Ergebnisse liefert. Wichtig hierbei ist die statistische Signifikanz der Benchmark-Ergebnisse unter dem Einfluss des Quantenrauschens sowie probabilistische Merkmale der Workloads in der Definition und Auswertung der Benchmarks. Eine einheitliche Methode zur Generierung und Ausführung der Workloads ist dafür

[7] IEEE P7131-Standard for Quantum Computing Performance Metrics & Performance Benchmarking: https://sagroups.ieee.org/7131, zuletzt besucht am 31.01.2025

essentiell.

**Fairness**

Die Reproduzierbarkeit bezeichnet das Maß inwieweit ein Benchmark bei wiederholter Ausführung kohärente Ergebnisse liefert. Hierbei müssen in der Definition und Auswertung der Benchmarks statistische Signifikanz der Ergebnisse sowie der Einfluss des Quantenrauschens und probabilistische Merkmale der Workloads selbst mit einbezogen werden.

**Skalierbarkeit**

Die Skalierbarkeit bezeichnet einerseits das Maß, mit dem die Workloads eines Benchmarks im Schaltungsvolumen - also der Breite und Tiefe der auszuführenden Quantenschaltungen - mit der Problemgröße wachsen, was insbesondere für NISQ-Hardware mit stark limitierter Qubitanzahl und Kohärenzzeiten, eingeschränkter Topologie und relativ hohen Fehlerraten relevant ist. Andererseits betrifft dies auch die Komplexität und somit die Berechnungszeit des Pre- und Postprocessing der Workloads, sowie die Auswertung der Ergebnisse und Berechnung der Metriken. Hierbei sind Metriken, welche auf Grundlage klassischer Simulationen berechnet werden müssen in der Regel wenig skalierbar.

**Verifizierbarkeit**

Die Verifizierbarkeit bezeichnet den Aufwand mit dem eine auf einer QPU ausgeführte Workload auf Korrektheit überprüft werden kann. In der Regel skaliert die Größe der erhaltenen Ausgabedaten des Quantencomputers exponentiell mit der Problemgröße, weshalb Benchmarks, deren Metriken auf Grundlage klassischer Simulationen berechnet werden müssen, schwer verifizierbar sind. Dies ist eng verwandt mit der Skalierbarkeit.

**Nutzbarkeit**

Die Nutzbarkeit kennzeichnet einerseits das Maß an technischem Verständnis, um einen Benchmark ausführen und beurteilen zu können, und andererseits auch den Aufwand die Ergebnisse zu validieren und mit anderen Ergebnissen auf anderen Systemen zu vergleichen. Die Verfügbarkeit von Tools und Suiten mit denen Benchmarking-Workloads generiert, auf einer Zielhardware ausgeführt und die Ergebnisse interpretiert und validiert werden können, sind hierbei von zentraler Bedeutung.

Basierend auf diesen Kriterien wurden in [7] Handlungsempfehlungen zu allgemeinen Benchmark-Methodiken sowie für unterschiedliche Mikro-, Makro- und anwendungsorientierte Benchmarks von gatterbasierten Quantencomputern definiert. Um diese Ergebnisse im Sinne der Nutzbarkeit, Fairness und Reproduzierbarkeit praktisch anwendbar zu machen, ist die Implementierung dieser Benchmarks in einer Software ein geeigneter Schritt. Diese soll von möglichst vielen Nutzern ohne größere Hürden verwendet werden können und die Möglichkeit bieten, Benchmarkergebnisse übersichtlich und einfach verständlich darzustellen. Die Konzeption eines ersten Prototyps für eine solche Benchmarking Suite ist im nachfolgendem Abschnitt ausführlich beschrieben.

# 3. Entwickelte Benchmarks praktisch verfügbar machen - WebMarQC

Das Ziel des WebMarQC-Vorhabens ist es, einen ersten Prototypen für ein webbasiertes Benchmarking-Tool für gatterbasierte Quantencomputer zu entwickeln, welches an die Benchmarkingstrategien und -anforderungen sowie den Metriken aus [7] anknüpft und bisherige Benchmark-Implementierungen aus dem *EniQmA*-Projekt mittels der Quantenhochsprache *Eclipse Qrisp* herstellerneutral und performant nutzbar macht. Das Aufsetzen eines webbasierten Services soll häufig auftretende Probleme von Benchmarksuiten lösen, welche oft kompliziert installiert werden müssen und im Zuge der rasanten Entwicklung verschiedener Quantencomputing Software Frameworks dabei schnell inkompatibel mit verschiedenen Backends werden. Mithilfe des webbasierten Services sollen Workloads mittels *Eclipse Qrisp* für verschiedene Benchmarks generiert und auf verschiedenen Backends ausgeführt werden können. Damit soll die Nutzbarkeit von bisher etablierten Benchmarks und damit die Akzeptanz in der Community erhöht werden. Die Empfehlungen für zu verwendende Input-Parameter und Auswertungsmetriken richten sich dabei nach bisherigen Standardisierungsvorhaben, können aber von versierten Nutzern auch individuell angepasst werden. Wichtig für die Transparenz und Vergleichbarkeit ist hier, dass alle den jeweiligen Benchmark betreffende Parameter gespeichert werden und in einem finalen Report abgebildet werden sollen. Dies erhöht die Fairness und sorgt für eine klarere Interpretation der Benchmarkergebnisse.

## 3.1. Nutzung von *Eclipse Qrisp* für das Benchmarken von gatterbasierten Quantencomputern

*Eclipse Qrisp* ist ein High-Level-Programmierframework zur Erstellung und Kompilierung von Quantenalgorithmen, welche als eine eingebettete domänenspezifische Sprache (eDSL) vollständig in Python entwickelt ist. Qrisp bietet eine Programmierschnittstelle, die nahtlos mehrere Abstraktionsebenen verbindet, um eine skalierbare Entwicklung und Kompilierung von Quantenalgorithmen zu ermöglichen und im Vergleich zu vielen anderen Quantencomputing SDKs zu vereinfachen [27]. Dabei ist Qrisp Open-Source und stellt Herstellerunabhängigkeit, Lesbarkeit, Nutzerfreundlichkeit und Performanz in den Vordergrund. Qrisp ist somit insbesondere zur Implementierung standardisierter, fairer und herstelleroffener Benchmarks geeignet. Im Folgenden sind einige Aspekte aufgelistet, welche diese Eignung unterstreichen sollen:

1. Benchmark-Workloads können auf abstrakte und dem Nutzer verständliche Weise definiert und herstellerunabhängig auf unterschiedlicher Hardware ausgeführt werden. Diese Abstraktion erhöht die Akzeptanz der Benchmarks in der Community und das Verständnis, wie und warum die Workloads definiert und implementiert wurden. Es werden hierfür fortlaufend neue Schnittstellen zu den Backends verschiedener Hersteller entwickelt.

2. In Qrisp implementierte Quantenalgorithmen können oft performanter kompiliert werden, als in vergleichbaren Frameworks und bieten oft Vorteile in der resultieren Größe der erzeugten Quantenschaltungen. Dies wurde bereits anhand einer Implementierung von Shor's Algorithmus sowieso verschiedener QAOA-basierter Quantenalgorithmen [16] demonstriert. Vor allem in der aktuellen NISQ-Ära ist es von großem Vorteil, anwendungsorientierte Workloads mit so wenigen Ressourcen wie möglich erzeugen zu können, da sich diese Art von Benchmarks auf der aktuellen gatterbasierten Hardware nur für sehr kleine Problemgrößen überhaupt ausführen lassen und die Möglichkeit der Performanzevaluation somit stark begrenzt sind. Darüber hinaus wird Qrisp fortlaufend mit nativen Algorithmus-Modulen erweitert, zum Beispiel die Implementierung von Hamiltonian Simulation mittels *VQE* (Variational-time Quantum Eigensolver) oder dem *QITE*-Algorithmus (Quantum Imaginary Time Evolution), sowie dem Lösen kombinatorischer Problem mit dem *QIRO*-Algorithmus (Quantum-Informed Recursive Optimization).

3. Eclipse Qrisp unterstützt das Simulieren von Quantenschaltungen mittels zahlreicher hochperformanter externer Simulatoren, welche über ein Backend-Interface auf entsprechende Docker-Container angesteuert werden (z.B., *Stim* [28], *Qulacs* [29], *MQT DDSIM* [8]. Das vereinfacht das Entwickeln und Testen von Benchmarks bevor diese auf echter Quantencomputing-Hardware ausgeführt.

4. Die Abstraktionsebenen von Eclipse Qrisp ermöglichen Methoden zum Testen und Debuggen von Quantencomputing-Anwendungen, welche mit der herkömmlichen assemblerartigen Programmierweise dieser Anwendungen nur schwer oder gar nicht umsetzbar sind. Das vereinfacht ebenfalls das Testen und die Qualitätssicherung der entwickelten Benchmarks. Auf solche prototypischen Tools kann bereits jetzt zugegriffen werden, da diese aktuell im *EniQmA*-Projekt entwickeln und erprobt werden.

5. In die Zukunft blickend ist Qrisp insbesondere dafür geeignet, komplexere Anwendungsfälle abzubilden, welche Techniken wie automatisierte *uncomputation/garbage collection* sowie einen hohen Grad an Modularisierung benötigen. Einige dieser Techniken wurden bereits anhand eines Backtracking-Algorithmus demonstriert [30]. Ferner bietet das *Jasp*-Modul [9] durch integrierte *just-in-time* (JIT) Kompilierung immense Vorteile für die Ausführung hybrider Quantenalgorithmen, die zukünftige Unterstützung von in Echtzeit berechneten Methoden zur Fehlerkorrektur, sowie das Kompilieren von Quantenprogrammen in Größenordnungen, welche für reale Anwendungen relevant sind. Dies kommt der Implementierung entsprechender Benchmarking-Workloads zugute.

## 3.2. Die Architektur der WebMarQC-Benchmark Suite

Die WebMarQC-Benchmark Suite soll auf einer virtuellen Maschine im Backend des Webservices gehostet werden, sodass diese über eine API über den Webservice angesteuert werden kann. Die Benchmarksuite besteht dabei aus den folgenden Modulen:

[8] https://github.com/cda-tum/mqt-ddsim
[9] https://qrisp.eu/reference/Jasp/index.html

**Kernmodul:** Im Kernmodul findet sich die relevante Infrastruktur für sämtliche Prozesse der Benchmark Suite, wie die Backend-Interfaces für unterschiedliche Zielhardware, die Funktionalitäten für die Konfiguration der Standardparameter für die Benchmarks sowie der Zugangtokens für die Zielhardware, die Erfassung und Speicherung von Benchmark-Daten und Kalibrierungsdaten der Zielhardware sowie ein Command Line Interface (CLI) zum Verwenden der Benchmark Suite über das Terminal. Dieses wird auch als API für die Kommunikation zwischen der Webseite/Webservice und der Benchmark Suite verwendet.

**Benchmarkmodul**: Hier sind alle implementierten Benchmarks zu finden. Für den ersten Prototypen umfassen diese das Randomized Benchmarking und volumetrische Benchmarking, Methoden zur Verschränkungscharakterisierung sowie kleinere algorithmische Benchmarks im Sinne der sogenannten Algorithmic Building Blocks aus [7]. Zukünftig sollen weitere Benchmarks als Plugins unabhängig vom Kernmodul hinzugefügt werden können, wodurch die Benchmarksuite dynamisch neue Erkenntnisse und Weiterentwicklungen abbilden können soll.

**Datenbank:** In der Datenbank werden sämtliche Benchmarkergebnisse in verschiedenen Formaten gespeichert. Diese enthalten die Input-Parameter, bestimmte Metadaten, die Messergebnisse, Kalibrierungsdaten der Zielhardware sowie die berechneten Metriken und eventuelle Zwischenergebnisse. Zusätzlich werden dort die generierten Benchmark-Reports mit dazugehörigen Grafiken gespeichert.

Der Webservice wird über eine grafische Benutzeroberfläche auf der Webseite webmarqc.fokus.fraunhofer.de bedient, während die Generierung der Workloads, das Ausführen auf der Zielhardware sowie das Berechnen der Metriken und die Generierung der Benchmark-Reports automatisiert im Hintergrund ausgeführt werden. Die Interaktion eines Nutzers mit dem Webservice und die im Hintergrund ausgeführten Prozesse sind in Abb.2 abgebildet und werden im folgenden kurz zusammengefasst:

**Benutzerregistrierung und Anmeldung:** Beim ersten Zugriff auf die WebmarQC-Plattform können sich die Benutzer für ein Konto registrieren. Die Registrierung gewährleistet eine sichere und personalisierte Benutzererfahrung, bei der die Benutzer die erforderlichen Angaben machen und den Nutzungsbedingungen zustimmen. Nach der Registrierung melden sich die Benutzer an, um auf die Funktionen der Plattform zuzugreifen. Nach der Anmeldung wird dem Nutzer ein personalisiertes Dashboard mit einem Überblick über bisher generierte und ausgeführte Benchmarks und deren Ergebnisse, sowie der Zugang zu den Kernfunktionalitäten der Webseite angezeigt. Dazu gehören die Generierung, Ausführung und Auswertung neuer Benchmarks.

**Generieren und Ausführen von Benchmark-Workloads:** Benutzer können Benchmark-Schaltungen für verschiedene implementierte Benchmarks über eine dynamische Schnittstelle erstellen oder auswählen. Standardmäßig werden dafür empfohlene Input-Parameter vorgegeben, welche vom Benutzer jedoch angepasst werden können. Dies wird dann auch auf den Benchmark-Reports entsprechend abgebildet. An dieser Stelle ist es notwendig ein unterstütztes Ziel-Backend auszuwählen, damit die Schaltungen passend für die Spezifi-

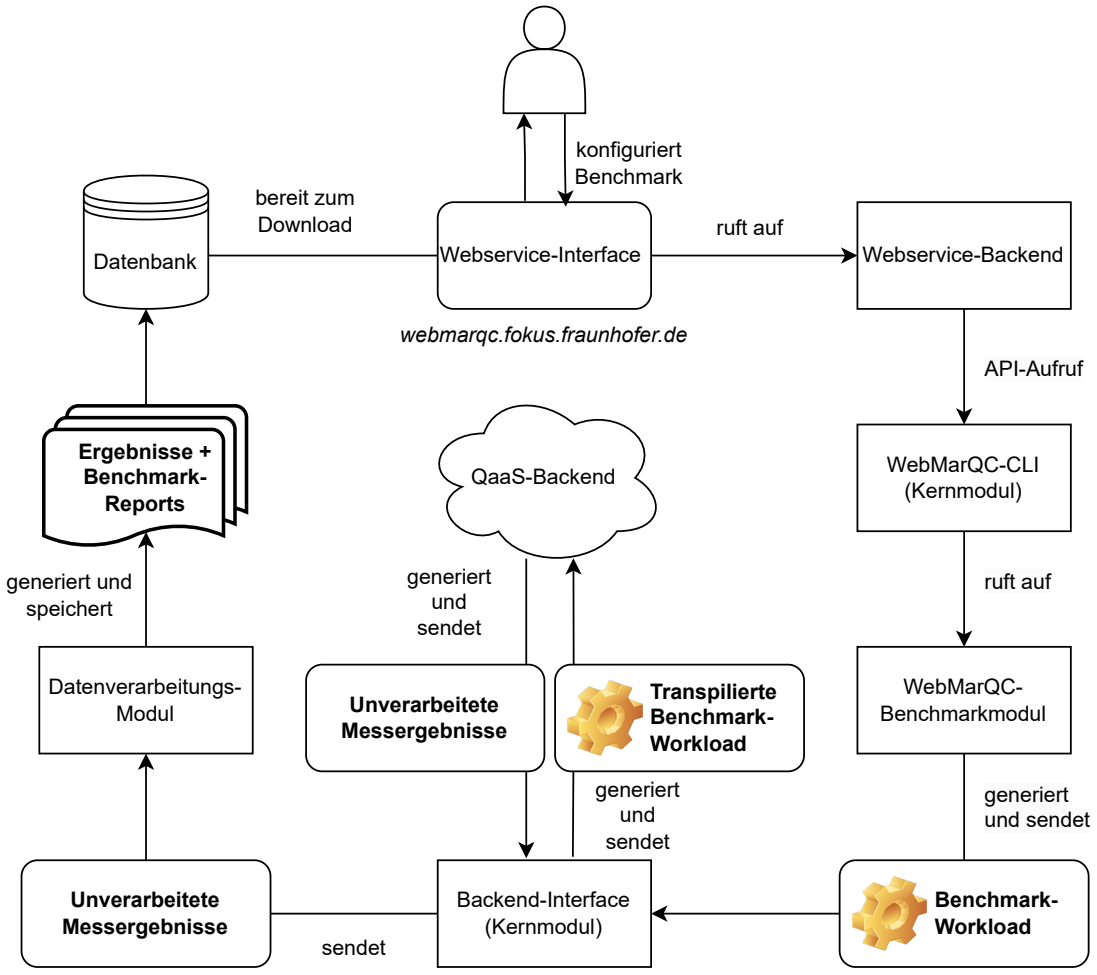



***Abbildung 2:*** *Webservice Flowdiagramm zur Nutzerinteraktion*

kationen des Backends erstellt werden können. Die generierten Schaltungen können anschließend im *OpenQASM*-Format [31] gespeichert und selber lokal ausgeführt werden [10] oder über den Webservice unter Eingabe eines gültigen Zugangstokens [11] an die Zielhardware geschickt werden. Während der Ausführung der Benchmarks erhalten die Benutzer Updates über den aktuellen Status des Jobs.

**Generieren und Verarbeiten der Benchmark-Ergebnisse:** Nach Erhalten der Messergebnisse für die entsprechende Benchmark-Workload, werden diese automatisch verarbeitet und die ausgewählten Leistungsmetriken berechnet. Anschließend werden die Benutzer benachrichtigt und die Ergebnisse stehen in der Plattform zum Download zur Verfügung. Dazu gehören die reinen Messergebnisse im JSON-Format, die generierten Benchmark-Reports im HTML- und PDF-Format, sowie generierte Grafiken im PNG-Format. Der Report bietet einen umfassenden Überblick über die Benchmark-Ergebnisse und enthält ausgewählte Metadaten zu den Input-Parametern und Softwareversionen.

[10] Hierzu wird ein entsprechender Code-Ausschnitt bereitgestellt, mit welchem die Schaltungen an das entsprechende Backend übermittelt werden können.

[11] Solche (kostenpflichtigen) Tokens werden in der Regel benötigt um Zugriff auf Cloud-basierte QaaS-Hardware zu bekommen.

Zunächst soll dieser Prototyp verschiedenen Nutzern zugänglich gemacht werden, um systematisch Nutzerfeedback zu sammeln. Dieses Feedback ist von entscheidender Bedeutung, da es ermöglicht, die Nutzeroberfläche insbesondere der Web-Oberfläche gezielt zu verbessern, sodass diese den Bedürfnissen der Anwender entspricht.

## 3.3. Paralleles Ausführen von Benchmarks

Das aktuell hauptsächlich vorherrschende Paradigma für die Ausführung von Quantenschaltungen auf einem QaaS-Backend ist die *serielle Ausführung*. Dabei werden $n$ einzelne Quantenschaltungen erstellt, in sogenannte Jobs gebündelt und an das Backend gesendet und dort der Reihe nach ausgeführt. Der Nutzer erhält dann für jeden Job die gesammelten Ergebnisse für jede Schaltung zurück. Diese Art der Ausführung ist im Allgemeinen sehr ineffizient, war aber für einige Jahre die einzige Möglichkeit, da frühe NISQ-Hardware noch über wenige Qubits verfügten und es entsprechend wenig Möglichkeit zur simultanen Ausführung mehrerer Quantenschaltungen gab. Betrachtet man jedoch, dass zahlreiche QPUs heutzutage bereits über hunderte von Qubits verfügen können, so wird ersichtlich, dass bei dieser seriellen Ausführung ein Großteil der Ressourcen ungenutzt bleiben und eine effizientere Ausführung prinzipiell möglich wäre. Orientiert man sich beispielsweise an dem Quantenvolumen (siehe Abb. 1) und nimmt dies als groben Indikator für die ungefähre Schaltungsdimensionen, welche sich erfolgreich auf einer QPU ausführen lassen, so wird deutlich, dass die Schaltungsbreite dabei nur ein Bruchteil der gesamten zur Verfügung stehenden Qubits ausmacht. Es lassen sich also prinzipiell mehrere solcher Schaltungen gleichzeitig ausführen. Dies nennt man in der Fachliteratur *Multiprogramming* [32–34].

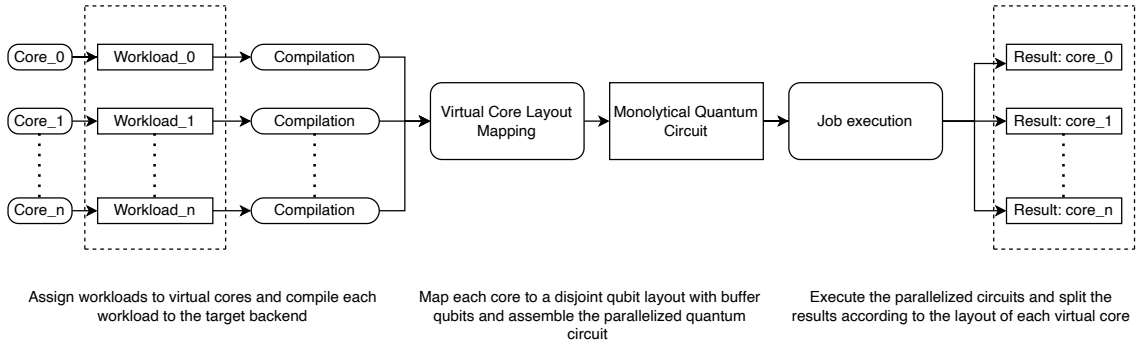



***Abbildung 3:*** *Schematischer Ablauf beim parallelisierten Ausführen von $n$ Workloads auf einer QPU mittels lokalem Multiprogramming.*

In der Regel bezieht sich dies darauf, mehrere Quantenschaltungen mittels eines speziellen Compilers zu einer einzelnen parallelisierten Schaltung mit mehreren sogenannten *virtuellen Kernen* zusammenzufügen, auf einem Backend auszuführen und die Ergebnisse danach den virtuellen Kernen zuzuordnen. Dieser Prozess ist schematisch in Abb. 3 dargestellt. Vereinfacht sorgt dies also dafür, dass bei einer Gesamtzahl von $N$ einzelnen Quantenschaltungen, nur noch $\lceil \frac{N}{k} \rceil$ Schaltungen parallel ausgeführt werden, wenn jede Schaltung in $k$ virtuelle Kerne mit simultan ausgeführten Berechnungen aufgeteilt ist,

anstatt $N$ einzelne Schaltungen seriell auszuführen. Eine andere Möglichkeit wäre das Parallelisieren von eingehenden Quantenschaltungen auf Seiten des Backend-Betreibers (dies lässt sich natürlich auch mit der Verwendung von lokalen Compilern verbinden), was zumindest nach aktuellem Stand noch nirgends angeboten wird. Daher beziehen wir uns im Folgenden nur auf das lokale Multiprogramming.

Das Multiprogramming erfährt aufgrund der dafür speziell benötigten Software, welche aktuell noch nicht in gängigen Quantencomputing-SDKs integriert ist, relativ wenig praktische Anwendung. Zusätzlich ist auch bekannt, dass das Parallelisieren von Quantenschaltungen sich negativ auf die Qualität der Ergebnisse auswirkt, was unter anderem auf den Crosstalk zwischen den Qubits der verschiedenen virtuellen Kerne zurückzuführen ist und auch damit zusammenhängt, dass gleichverteilte Fehlerraten bei aktuellen Qubit-Implementierungen schwer zu realisieren sind und durch die simultane Ausführung von Quantenschaltungen auch Qubits verwendet werden, welche höhere Fehlerraten aufweisen. Da die Qualität der Ergebnisse auf aktueller NISQ-Hardware sowieso eher niedrig ist, gibt es eventuell eine gewisse Hemmschwelle, Schaltungen zu parallelisieren, wenn vorher nicht klar ist, ob die Ergebnisse eine hinreichende Güte erreichen können. An dieser Stelle kann ein erweitertes Benchmarking zur Parallelisierungsfähigkeit einer QPU ins Spiel kommen, um genau diese Eigenschaft qualitativ und quantitativ zu Beurteilen. Bisher bietet die Literatur nur sehr wenige Benchmarks, welche mithilfe von Multiprogramming ausgeführt wurden [35, 36] und diese lassen sich nicht hinreichend verallgemeinern. Um eine weitergehende Forschung in diesem Bereich zu ermöglichen, wurde im Rahmen von WebMarQC die *Qrisp*-Infrastrukur um ein Modul erweitert, welches ein lokales Multiprogramming von mehreren Quantenberechnungen ermöglicht (siehe Abb. 3). Als erstes Fallbeispiel für einen konkreten Benchmark wurde dieses Framework auf den Quantenvolumen-Benchmark angewandt, wodurch sich beurteilen lässt, wie viele parallele Instanzen von Quantenvolumenschaltungen sich simultan auf einer QPU ausführen lassen. Zukünftig können auf dieser Grundlage weitere parallelisierte Benchmarks entwickelt werden oder verschiedene vorhandene Benchmarks parallel ausgeführt werden.

# 4. Handlungsempfehlungen

Im Rahmen dieser Studie wurden neben einer Einführung in das Benchmarken von gatterbasierten Quantencomputern auch aktuelle Herausforderungen und dazugehörige Lösungsansätze sowie der aktuelle Stand der Standardisierung in diesem Bereich beleuchtet. Zusammen mit dem vorgestellten Prototypen einer webbasierten Benchmarking Plattform, sehen wir eine Reihe von Handlungsempfehlungen, welche an die bisherigen Entwicklungen anknüpfen und Perspektiven für zukünftige Weiterentwicklungen aufzeigen. Diese sind im folgenden dargestellt.

**Bereitstellen verständlicher Benchmarking-Ressourcen**

Damit implementierte Benchmarks nicht nur effektiv genutzt werden können, sondern die Workloads, die erhaltenen Metriken und die Interpretation der Ergebnisse, insbesondere auch im Vergleich unterschiedlicher Zielhardware, von möglichst vielen Nutzern verstanden werden können, ist das Bereitstellen einfach verständlicher Ressourcen für jeden implementierten Benchmark ein geeignetes Mittel. Hierbei unmittelbar auf entsprechende wissenschaftliche Publikationen zurückzugreifen, würde ein sehr hohes mathematisches und technisches Verständnis voraussetzen, was eine Vielzahl potentieller Nutzer ausschließen könnte. Informationen, die auf nichtwissenschaftlicher Ebene erklären, wie die einzelnen Messungen durchgeführt werden und wie die Ergebnisse korrekt interpretiert werden können sollen dazu beitragen, das Verständnis für die Benchmarking-Prozesse zu fördern und den Nutzern eine fundierte Grundlage zu bieten, um die Ergebnisse in ihren eigenen Kontext zu setzen.

**Definition von Benchmarking Run Rules**

Durch einen größeren Fokus auf die Definition und Kommunikation sogenannter Benchmarking *run rules*, können weiterführende Standardisierungsaktivitäten und der Vergleich verschiedener Hardwareplattformen profitieren. Darunter kann man beispielsweise bestimmte gesetzte Kompilierungs- und Optimierungsmethoden verstehen, gewisse Zeitlimits für den gesamten Benchmark oder für gewisse Subroutinen oder auch Methoden zum Initialisieren geeigneter Input-Zustände. Es ist entscheidend, dass diese Regeln klar und verständlich in Benchmark-Reports abgebildet werden, um Transparenz und Nachvollziehbarkeit zu gewährleisten. Eine klare Darstellung dieser Regeln wird den Nutzern helfen, die Methodik hinter den Ergebnissen besser zu verstehen und deren Gültigkeit zu bewerten. Hierbei gilt es zu beachten, dass die Regeln nicht zu streng formuliert sein sollten, da dies wiederum den Vergleich unterschiedlicher Hardware sogar erschweren könnte.

**Fokussieren auf algorithmische Benchmarks**

Ein stärkerer Fokus auf die Entwicklung algorithmischer Benchmarks, die über das

synthetische *random circuit sampling* hinausgehen, ist ein wichtiger Schritt um aussagekräftigere und relevantere Benchmarkmetriken zu erhalten. Die Entwicklung algorithmischer Benchmarkergebnisse können bei ausreichender Datenlage über mehrere Jahre und auf unterschiedlicher Zielhardware ferner mit etablierten synthetischen Benchmarks verglichen werden. Dies kann zu einem besseren Verständnis synthetischer Benchmarkmetriken beitragen und diese in einen anwendungsnahen Kontext einordnen.

**Arbeiten an parallelisierten Benchmarks**

Es empfiehlt sich, verstärkt an parallelisierten Benchmarks zu arbeiten, die speziell für größere NISQ-Hardware bedeutend sind, welche zwar eine Vielzahl an Qubits besitzen, jedoch nur einen Bruchteil derer simultan mit ausreichender Güte verschränken und manipulieren können. Dadurch kann besser verstanden werden, wie sich Quantenschaltungen auf QPUs möglichst effizient ausführen lassen ohne dass die Qualität der Ergebnisse zu sehr leidet. Dies verringert die Kosten zur Ausführung von Schaltungen und reduziert effektiv die Wartezeit bei QaaS-Plattformen.

**Wissenaustausch in der Fach-Community**

Dem Schaffen von Möglichkeiten innerhalb der Community, um praktische Erfahrungen im Benchmarking von Quantencomputern zu erlangen ordnen wir eine hohe Bedeutung zu. Durch Workshops, Hackathons, Schulungen und weitere gemeinschaftliche Projekte können wir den Wissensaustausch fördern und eine engagierte Nutzerbasis aufbauen. Die *Thinq Qrisp* Community[a] kann für diesen Zweck ergänzend ein gutes Medium sein.

[a] https://qrisp.eu/general/thinq_qrisp/index.html

**Weitergehende Standardisierungsaktivitäten**

Die *DIN SPEC 91480* sollte als erster Schritt zur Standardisierung von Quantencomputing-Benchmarks verstanden werden. Es empfiehlt sich die Liste der zu bewertenden Benchmarks zu erweitern, weitere algorithmische Building Blocks zu definieren und ebenfalls das Benchmarken von Quantencomputing-Software zu betrachten. Da sich die Quantencomputing-Hardware stetig weiterentwickelt und stetig neue Methoden und Ansätze zum Benchmarken von Quantencomputern veröffentlicht werden, muss ein Standard in dieser Hinsicht auch regelmäßig aktualisiert werden. Ferner sollte ein vermehrter Austausch mit internationalen Standardisierungsgremien stattfinden, um bisherige Ergebnisse in einem größeren Konsortium zu verwerten und weiterzuentwickeln.

# Literatur


[1] F. Arute, K. Arya, and R. e. a. Babbush, "Quantum supremacy using a programmable superconducting processor," Nature 574, 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5 .

[2] S. Boixo, S. V. Isakov, *et al.*, "Characterizing quantum supremacy in near-term devices," Nature Physics **14**, 595–600 (2018).

[3] Y. Wu, W.-S. Bao, *et al.*, "Strong quantum computational advantage using a superconducting quantum processor," (2021).

[4] R. LaRose, "A brief history of quantum vs classical computational advantage," (2024), arXiv:2412.14703 [quant-ph] .

[5] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, *et al.*, "Quantum error correction below the surface code threshold," Nature (2024), 10.1038/s41586-024-08449-y.

[6] B. W. Reichardt, D. Aasen, R. Chao, *et al.*, "Demonstration of quantum computation and error correction with a tesseract code," (2024), arXiv:2409.04628 [quant-ph] .

[7] "Din spec 91480: Benchmarking quantum computers with determined kpis," (2024).

[8] J. Helsen, I. Roth, E. Onorati, A. Werner, and J. Eisert, "General framework for randomized benchmarking," PRX Quantum **3**, 020357 (2022).

[9] J. M. Gambetta, A. D. Córcoles, S. T. Merkel, B. R. Johnson, J. A. Smolin, J. M. Chow, C. A. Ryan, C. Rigetti, S. Poletto, T. A. Ohki, M. B. Ketchen, and M. Steffen, "Characterization of addressability by simultaneous randomized benchmarking," Phys. Rev. Lett. **109**, 240504 (2012).

[10] R. Blume-Kohout, E. Nielsen, K. Rudinger, K. Young, M. Sarovar, and T. Proctor, "Idle tomography: Efficient gate characterization for N-qubit processors," in *APS March Meeting Abstracts*, APS Meeting Abstracts, Vol. 2019 (2019) p. P35.006.

[11] D. C. McKay, I. Hincks, E. J. Pritchett, M. Carroll, L. C. G. Govia, and S. T. Merkel, "Benchmarking quantum processor performance at scale," (2023), arXiv:2311.05933 [quant-ph] .

[12] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," Phys. Rev. A **100**, 032328 (2019).

[13] R. Blume-Kohout and K. C. Young, "A volumetric framework for quantum computer benchmarks," Quantum **4**, 362 (2020).

[14] A. Wack, H. Paik, A. Javadi-Abhari, P. Jurcevic, I. Faro, J. M. Gambetta, and B. R. Johnson, "Quality, speed, and scale: three key attributes to measure the performance of near-term quantum computers," (2021), arXiv:2110.14108 [quant-ph] .

[15] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, and T. Proctor, "Application-oriented performance benchmarks for quantum computing," IEEE Transactions on Quantum Engineering **4**, 1–32 (2023).

[16] R. Zander, R. Seidel, M. Inajetovic, N. Steinmann, and M. Petrič, "Solving the product breakdown structure problem with constrained qaoa," (2024), arXiv:2406.15228 [quant-ph] .

[17] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest, "Measurement-based quantum computation," Nature Physics **5**, 19–26 (2009).

[18] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, "Quantum annealing: an overview," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **381**, 20210417 (2023), https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2021.0417 .

[19] Y. Chi, J. Huang, Z. Zhang, J. Mao, Z. Zhou, X. Chen, C. Zhai, J. Bao, T. Dai, H. Yuan, M. Zhang, D. Dai, B. Tang, Y. Yang, Z. Li, Y. Ding, L. K. Oxenløwe, M. G. Thompson, J. L. O'Brien, Y. Li, Q. Gong, and J. Wang, "A programmable qudit-based quantum processor," Nature Communications **13**, 1166 (2022).

[20] V. Lahtinen and J. K. Pachos, "A Short Introduction to Topological Quantum Computation," SciPost Phys. **3**, 021 (2017).

[21] V. M. Kendon, K. Nemoto, and W. J. Munro, "Quantum analogue computing," Philosophical Transactions: Mathematical, Physical and Engineering Sciences **368**, 3609–3620 (2010).

[22] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Journal on Computing **26**, 1484–1509 (1997), https://doi.org/10.1137/S0097539795293172 .

[23] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," Phys. Rev. Lett. **103**, 150502 (2009).

[24] J. von Kistowski, J. Arnold, K. Huppler, K.-D. Lange, J. Henning, and P. Cao, "How to build a benchmark," ICPE 2015 - Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (2015), 10.1145/2668930.2688819.

[25] J. v. K. Samuel Kounev, Klaus-Dieter Lange, *Systems Benchmarking*, 1st ed. (Springer Cham, 2021).

[26] C. K.-U. Becker, N. Tcholtchev, I.-D. Gheorghe-Pop, S. Bock, R. Seidel, and M. Hauswirth, "Towards a quantum benchmark suite with standardized kpis," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)* (2022) pp. 160–163.

[27] R. Seidel, S. Bock, R. Zander, M. Petrič, N. Steinmann, N. Tcholtchev, and M. Hauswirth, "Qrisp: A framework for compilable high-level programming of gate-based quantum computers," (2024), arXiv:2406.14792 [quant-ph] .

[28] C. Gidney, "Stim: a fast stabilizer circuit simulator," Quantum **5**, 497 (2021).

[29] Y. Suzuki, Y. Kawase, Y. Masumura, Y. Hiraga, M. Nakadai, J. Chen, K. M. Nakanishi, K. Mitarai, R. Imai, S. Tamiya, T. Yamamoto, T. Yan, T. Kawakubo, Y. O. Nakagawa, Y. Ibe, Y. Zhang, H. Yamashita, H. Yoshimura, A. Hayashi, and K. Fujii, "Qulacs: a fast and versatile quantum circuit simulator for research purpose," Quantum **5**, 559 (2021).

[30] R. Seidel, R. Zander, M. Petrič, N. Steinmann, D. Q. Liu, N. Tcholtchev, and M. Hauswirth, "Quantum backtracking in qrisp applied to sudoku problems," (2024), arXiv:2402.10060 [quant-ph] .

[31] A. Cross, A. Javadi-Abhari, T. Alexander, N. De Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, "Openqasm 3: A broader and deeper quantum assembly language," ACM Transactions on Quantum Computing **3** (2022), 10.1145/3505636.

[32] P. Das, S. S. Tannu, P. J. Nair, and M. Qureshi, "A case for multi-programming quantum computers," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52 (Association for Computing Machinery, New York, NY, USA, 2019) p. 291–303.

[33] S. Niu and A. Todri-Sanial, "How Parallel Circuit Execution Can Be Useful for NISQ Computing?" in *Design, Automation and Test in Europe Conference and Exhibition 2022* (2021) arXiv:2112.00387 [cs.AR] .

[34] S. Niu and A. Todri-Sanial, "Enabling Multi-programming Mechanism for Quantum Computing in the NISQ Era," Quantum **7**, 925 (2023).

[35] L. Mineh and A. Montanaro, "Accelerating the variational quantum eigensolver using parallelism," (2022), arXiv:2209.03796 [quant-ph] .

[36] S. Niu and A. Todri-Sanial, "Multi-programming Cross Platform Benchmarking for Quantum Computing Hardware," (2022), arXiv:2206.03144 [quant-ph] .

[37] F. J. Kiwit, M. Marso, P. Ross, C. A. Riofrío, J. Klepsch, and A. Luckow, "Application-oriented benchmarking of quantum generative learning using quark," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2023) p. 475–484.

[38] J. R. Finzgar, P. Ross, L. Hölscher, J. Klepsch, and A. Luckow, "Quark: A framework for quantum computing application benchmarking," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2022) pp. 226–237.

[39] T. Tomesh, P. Gokhale, V. Omole, G. Ravi, K. N. Smith, J. Viszlai, X. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, "Supermarq: A scalable quantum benchmark suite," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (IEEE Computer Society, Los Alamitos, CA, USA, 2022) pp. 587–603.

[40] F. Barbaresco, L. Rioux, C. Labreuche, M. Nowak, N. Olivier, D. Nicolazic, O. Hess, A.-L. Guilmin, R. Wang, T. Sassolas, S. Louise, K. Snizhko, G. Misguich, A. Auffèves, R. Whitney, E. Vergnaud, and F. Schopfer, "Bacq – application-oriented benchmarks for quantum computing," (2024), arXiv:2403.12205 [quant-ph] .

[41] K. Mesman, Z. Al-Ars, and M. Möller, "Qpack: Quantum approximate optimization algorithms as universal benchmark for quantum computers," (2022), arXiv:2103.17193 [cs.ET] .

[42] H. Donkers, K. Mesman, Z. Al-Ars, and M. Möller, "Qpack scores: Quantitative performance metrics for application-oriented quantum computer benchmarking," (2022), arXiv:2205.12142 [quant-ph] .

[43] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation," ACM Transactions on Quantum Computing **4** (2023), 10.1145/3550488.

# Abbildungsverzeichnis

# 5. Anhang

## 5.1. Übersicht über nationale Projekte und Vorhaben zum Benchmarken von Quantencomputern

**Munich Quantum Toolkit Benchmarking Library (*MQT Bench*)**: Diese frei zugängliche Benchmark-Suite umfasst etwa 70.000 Benchmark-Schaltungen mit 2 bis 130 Qubits. Dazu gehören algorithmische Routinen wie etwa die *Quantenfouriertransformation*, *Quantum Phase Estimation* und *Variational Quantum Eigensolver* mit verschiedenen Ansätzen, die Generierung von Zuständen wie GHZ-, W- und Graphen-Zustände und Use Cases wie das *Travelling Salesman Problem*. Diese können mittels verschiedener Abstraktionsebenen auf unterschiedliche Zielhardware angepasst werden und über *Qiskit* oder *Tket* transpiliert und optimiert werden. Die erhaltenen Schaltungen stehen schließlich im OpenQASM-Format bereit. [12] [13]

**QUARK Benchmark Suite:** Diese frei zugängliche Benchmark Suite [14] ist auf das Generieren und Ausführen von Benchmark-Workloads für verschiedener Industrieanwendungen ausgelegt. Dazu gehören das *Travelling Salesman Problem*, das *Maximum-Satisfiability-Problem* (MaxSAT), die Optimierung von Roboterpfaden, das *Maximum Independent Set Problem* (MIS), das *Set Cover Problem* (SCP) sowie das *Auto Carrier Loading* (ACL) [37, 38].

**Bench-QC:** Im Rahmen dieses Projekts wird untersucht, welche quantitativen und qualitativen Eigenschaften Quantencomputer erfüllen müssen, um Vorteile für unterschiedliche industrielle Anwendungsbereiche gegenüber klassischer HPC-Hardware erzielen zu können. Ein besonderer Fokus liegt hierbei auf Simulationsproblemen, Optimierungsfragen und Quantum Machine Learning [15]. (Projektlaufzeit: Januar 2023 – Dezember 2025)

**EniQmA Benchmark Suite:** Im Rahmen des *EniQmA*-Projekts entsteht ein Workflow-basiertes Ökoystem zur Umsetzung hybrider quanten-klassischer Algorithmen. Dazu werden verschiedene Tools entwickelt, welche unter anderem das Testen und Verifizieren von Quantencomputing-Anwendungen sowie das Benchmarken von gatterbasierten Quantencomputern umfassen [16]. Die entwickelten Benchmarks sollen in einer Benchmark-Suite und über die *PlanQK*-Plattform [17] zur Verfügung gestellt werden. (Projektlaufzeit: Juli 2022 - Juli 2025)

[12] https://www.cda.cit.tum.de/mqtbench/
[13] https://github.com/cda-tum/mqt-bench
[14] https://github.com/QUARK-framework/QUARK
[15] https://www.iks.fraunhofer.de/de/projekte/bench-qc-anwendungsgetriebenes-benchmarking-von-quantencomputern.html
[16] https://www.eniqma-quantum.de/de/eniqma/tools
[17] https://kipu-quantum.com/platform/

## 5.2. Übersicht über internationale Projekte und Vorhaben zum Benchmarken von Quantencomputern

**Metriq:** Die *Metriq* Web-Plattform [18] katalogisiert Benchmarkergebnisse zu unterschiedlichen Workloads und für unterschiedliche Hardware und stellt diese im Kontext eines zeitlichen Verlaufs dar. Die Ergebnisse können dabei direkt von den Nutzern hinzugefügt werden und benötigen als Quellen in der Regel eine dazugehörige wissenschaftliche Veröffentlichung, Studie oder Meldung. Mithilfe der Daten soll langfristig die Frage *Wie schneidet die Quantencomputing-Plattform X unter Verwendung des Software Stacks Y in der Workload Z ab und wie sieht die zeitliche Entwicklung dessen aus?* beantworten.

**SupermarQ Benchmarking Suite:** Diese frei zugängliche Benchmark Suite beinhaltet unter anderem nwendungsorientierte Benchmarks aus unterschiedlichen Bereichen, wie etwa *Hamiltonian Simulation*, *Variational Quantum Eigensolvers* und *Quantum Approximate Optimization Algorithms* [39]. Außerdem enthält sie synthetische Benchmarks zum Charakterisieren von Qubits mittels Methoden des Randomized Benchmarkings, Fidelity Estimation und Cross-entropy Benchmarkings.

**BACQ:** Das *BACQ*-Projekt (Application-oriented Benchmarks for Quantum Computing) zielt darauf ab, die praktische Leistung von Quantencomputern in objektiver und zuverlässiger Weise zu beurteilen. Dabei sollen insbesondere eine Reihe von anwendungsorientierten Benchmarks berücksichtigt werden, welche für industrielle Endnutzer relevant sind sind[19]. Dazu gehören unter anderem Anwendungen aus den Bereichen der Quantensimulation, Optimierung, dem Lösen linearer Gleichungssysteme und der Primfaktorzerlegung [40]. Die Ergebnisse sollen in einer Benchmark Suite verfügbar gemacht werden. (Projektlaufzeit: 2023 - 2026)

**NEASQC Benchmark Suite:** Diese frei zugängliche Benchmark Suite [20] ist ein Teil des europäischen *NEASQC*-Projekts, welches im November 2024 abgeschlossen wurde. Details zu den Implementierungen der Benchmarks aus den Bereichen von *Probability Loading Algorithmen*, *Quantum Phase Estimation*, *Hamiltonian Simulation* und *Quantum Amplitude Estimation*, zusammen mit den verwendeten Metriken und dem Reporting der Ergebnisse sind in einem Deliverable [21] festgehalten.

**QED-C Benchmark Suite:** Diese frei zugängliche Benchmark Suite enthält prototypische anwendungsorientierte Benchmark-Workloads, mit denen die Performanz aktueller NISQ-Hardware unter Verwendung des volumetrischen Benchmarking Frameworks charakterisieren lässt [15]. Die implementierten Workloads sind dabei auch die Grundlage für den holistischen *algorithmic qubit* Benchmark. Das Repository [22] wird von Mitgliedern des Technical Advisory Committee on Standards and Performance Metrics (Standards TAC) des

[18] https://metriq.info
[19] https://www.lne.fr/en/press-releases/bacq-delivering-application-oriented-benchmark-suite-objective-multi-criteria
[20] https://github.com/NEASQC/WP3_Benchmark
[21] https://www.neasqc.eu/wp-content/uploads/2023/10/NEASQC_D3.5_Benchmark_suite_R1.0.pdf
[22] https://github.com/SRI-International/QC-App-Oriented-Benchmarks

Quantum Economic Development Consortium (QED-C) verwaltet.

**QPACK Benchmark Suite:** Diese frei zugängliche Benchmark Suite [23] hat das Ziel hersteller- und technologieneutrale Benchmarks für NISQ-Hardware zur Verfügung zu stellen. Insbesondere werden hier verschiedene Anwendungen des Quantum Approximate Optimization Algorithm (QAOA) und Variational-Quantum-Eigensolver (VQE) betrachtet [41, 42]. Die Benchmarkergebnisse werden aggregiert und unter den Aspekten Kapzität, Skalierbarkeit, Genauigkeit und Laufzeit ausgewertet. Schließlich werden die Ergebnisse grafisch aufbereitet und in einem einzigen Diagramm dargestellt, um einen Vergleich verschiedener Zielhardware zu ermöglichen.

**QASMBench Benchmark Suite:** Diese frei zugängliche Benchmark Suite [24] enthält vorgefertigte Benchmark-Workloads aus unterschiedlichen Algorithmen im *OpenQASM*-Format [43]. Diese sind in drei verschiedene Größenordnungen von 2-10 Qubits, 11-27 Qubits und 28-433 Qubits unterteilt und können vom Nutzer selbstständig ausgeführt und ausgewertet werden.

[23] https://gitlab.com/libket/qpack
[24] https://github.com/pnnl/QASMBench