Entwurfsautomatisierung für Quantencomputer[1]

Alwin Zulehner[2]

Abstract: Quantencomputer erlauben es, bestimmte Probleme exponentiell schneller als konventionelle Rechner zu lösen. Obwohl ihre Entwicklung aktuell noch in den Kinderschuhen steckt, scheinen Quantencomputer in greifbarer Nähe. Um deren Potential voll auszuschöpfen, müssen jedoch einige Entwurfsaufgaben effizient bewältigt werden. Die Dissertation stellt hierfür Ansätze vor, welche auf geschickte Weise Expertise aus dem Gebiet der Entwurfsautomatisierung nutzt. Dadurch konnten zahlreiche Methoden für den Entwurf von Quantenalgorithmen realisiert werden, die den derzeitigen Stand der Technik (sowohl hinsichtlich der Laufzeit als auch der Qualität der Ergebnisse) weit übertreffen. Die entwickelten Lösungen erzielten auch über den akademischen Bereich hinaus eine hohe Resonanz, was Auszeichnungen von Google und IBM, sowie die Integration in die Werkzeuge von IBM und Atos belegen.

1 Einführung

In den 1970er Jahren begannen Forscher, Fragen der Informatik und der Informationstheorie mit Hilfe von Quantenmechanik zu adressieren. In den daraus resultierenden neuen Forschungsrichtungen (beispielsweise Quantencomputing oder Quantensicherheit) dienen sogenannte Quantenbits (Qubits) als elementare Informationseinheit, welche im Gegensatz zu konventionellen Bits nicht nur einen der Basiszustände annehmen (in Dirac Notation $|0\rangle$ oder $|1\rangle$), sondern sich auch in einer beliebigen Überlagerung $\alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$ miteinander befinden können (wobei $\alpha_0, \alpha_1 \in \mathbb{C}$ und $\alpha_0 \alpha_0^* + \alpha_1 \alpha_1^* = 1$). Gemeinsam mit dem weiteren Phänomen der Quantenverschränkung (die beschreibt, wie sich die Zustände von Qubits gegenseitig beeinflussen), ergibt sich eine enorme Steigerung der Informationsdichte und potentiellen Rechenleistung für Quantensysteme [NC02].

Quantencomputing beschäftigt sich mit der Entwicklung von Algorithmen, die Quanteneffekte ausnutzen, um gewisse Problemstellungen deutlich schneller (oftmals exponentiell schneller) zu lösen als mit konventionellen Computern. Erste Anwendungen wie Grover's Suche [Gr96] oder Shor's Algorithmus, der es erlaubt Semiprimzahlen in polynomialer Zeit zu faktorisieren [Sh97], wurden bereits im letzten Jahrhundert publiziert. Seither hat sich das Anwendungsgebiet stark erweitert und beinhaltet nun auch Problemstellungen im Bereich der Chemie, der linearen Algebra, der Simulation von Quantensystemen und dem Machine Learning [Mo16]. Speziell im Bereich der Chemie (der Simulation von Molekülen) erwarten sich Forscher mit Hilfe von Quantencomputern starke Fortschritte; beispielsweise um bessere Katalysatoren für den Haber-Bosch Prozess zu finden, der 1-2 % des weltweiten Energieverbrauchs verursacht.

[1] Englischer Titel der Dissertation: "Design Automation for Quantum Computing"

[2] Institut für Integrierte Schaltungen, Johannes Kepler Universität Linz, alwin.zulehner@gmx.at

In letzter Zeit ist auch eine rasante Entwicklung in der Realisierung der Quantencomputer selbst beobachtbar. Während Prototypen von Quantencomputern ursprünglich an Universitäten entwickelt wurden, haben mittlerweile auch namenhafte Firmen wie *IBM*, *Google*, *Microsoft* und *Intel* als auch Startups wie *Rigetti* und *IonQ* damit begonnen, entsprechende Rechner zu realisieren [Go18]. Im Jahr 2017 startete IBM eine Initiative, um Quantencomputer über einen Cloud Service erstmals der Öffentlichkeit zugänglich zu machen. Seither wurden auf diesen Quantencomputern von mehr als 100 000 Benutzern über 6,5 Millionen Experimente durchgeführt. Im Januar 2019 präsentierte IBM außerdem *IBM Q System One*, den ersten Prototypen eines Quantencomputers, der außerhalb des Labors betrieben werden kann [IBM19].

Prognostiziert man einen gleichbleibenden Fortschritt in der Entwicklung von Quantencomputern (hinsichtlich der Anzahl der verfügbaren Qubits sowie der Genauigkeit der Operationen), werden Quantencomputer mehr und mehr für praktische Anwendungen interessant [Ar19]. Doch trotz dieser Fortschritte im Bereich der Quantenhardware werden effiziente Werkzeuge und Methoden benötigt, die den Entwurfsprozess für Anwendungen und auch für den Hardwareentwurf selbst automatisieren. Ansonsten droht eine Situation, in der Quantencomputer verfügbar sind, ihre volle Rechenkraft aber nicht genutzt werden kann. Genauer gesagt müssen insbesondere die folgenden Aufgaben effizient umgesetzt werden:

- **Simulation von Quantenschaltungen** auf konventionellen Rechnern sind wichtig, da reale Quantencomputer oft noch in ihrer Anzahl an Qubits und durch ihre Fehleranfälligkeit limitiert sind. Außerdem werden Simulatoren benötigt, um neue Quantenalgorithmen zu entwickeln, fehlerkorrigierende Codes zu evaluieren oder um die Korrektheit von Quantenschaltungen zu validieren. Darüber hinaus erlauben Simulatoren einen stärkeren Einblick in die jeweilige Anwendung, da einzelne Amplituden des aktuellen Zustands ausgelesen werden können (was bei realen Quantencomputern nicht möglich ist). Da aber sowohl Quantenzustände als auch Quantenoperationen in der Regel über (exponentiell große) Vektoren bzw. Matrizen beschrieben werden, ist die effiziente Umsetzung solcher Simulationen hochgradig komplex.
- Der **Entwurf Boolescher Komponenten** ist nötig, da sie einen wesentlichen Teil vieler Quantenalgorithmen bilden. In der Tat ist der "Quantenteil" eines zu kompilierenden Algorithmus üblicherweise bereits direkt in Form elementarer Gatter gegeben und bei vielen Quantenalgorithmen ähnlich (weshalb oft vordefinierte Bausteine zur Verfügung stehen). Bei Booleschen Komponenten (wie z.B. den häufig verwendeten Orakel-Funktionen [NC02]) ist dies jedoch nicht der Fall. Diese hängen oft von der jeweiligen Anwendung und der konkret betrachteten Instanz ab. Da Quantenschaltungen inhärent reversibel sind [NC02], benötigt man für deren Entwurf Ansätze zur Synthese von reversiblen Schaltungen. Diese Schaltungen unterscheiden sich aber fundamental von bisherigen (elektronischen) Schaltungen, weshalb hierfür komplett neue Syntheseverfahren benötigt werden.

- Schließlich sind Methoden zur **Abbildung auf reale Quantencomputer** nötig, um die jeweiligen Quantenalgorithmen (die, wie klassische Algorithmen, in der Regel in einer Hochsprache beschrieben werden) auf die elementaren Operationen der Zielarchitektur (beispielsweise den Quantencomputern von IBM oder Google) zu übersetzen. Als Zielbeschreibung dienen dabei so genannte Quantenschaltungen bestehend aus (elementaren) Quantengattern, die jeweils nur ein Qubit beeinflussen, aber von mehreren anderen Qubits kontrolliert werden können. Eine große Herausforderung besteht insbesondere darin, die logischen Qubits der Anwendung auf die physikalischen Qubits der Zielarchitektur abzubilden. Dies ist oft nicht trivial, da die möglichen Interaktionen zwischen Qubits auf den Architekturen stark limitiert sind. Aus diesem Grund muss deren Position stets angepasst werden, was zusätzliche Gatter (so genannte SWAP-Gatter zum Tauschen zweier Qubit-Werte) benötigt. Dies wiederum erhöht die Kosten und insbesondere die Fehleranfälligkeit der Berechnung. Hier effiziente Lösungen zu finden ist ein NP-hartes Problem [BKM18].

Im Bereich des Entwurfs konventioneller Schaltungen und Systeme sind ähnliche Aufgaben wohldefiniert und werden seit Jahrzehnten von Forscherinnen und Forschern bearbeitet. Dies führte zu hochentwickelten Entwurfsmethoden und Werkzeugen für Software und Hardware, die heutzutage für selbstverständlich gehalten werden und mit denen Aufgaben enormer Komplexität bearbeitet werden können (immerhin ließen sich damit heutige Schaltungen und Systeme entwerfen, die zum Teil aus über einer Milliarde Komponenten bestehen und astronomische Zustandsräume besitzen). Diese Werkzeuge und Methoden sind ein Hauptgrund dafür, dass elektronische Geräte aus praktisch allen Bereichen unseres Alltags nicht mehr wegzudenken sind.

Für Quantencomputer stellt sich diese Situation jedoch komplett anders dar: Obwohl hier in den letzten Jahren viel in die Entwicklung erster Tools investiert wurde (was z.B. zu SDKs wie IBMs *Qiskit*, Googles *Cirq*, Microsofts *Quantum Development Kit* oder Rigettis *Forest* führte), liefert keine dieser Entwicklungsumgebungen Lösungen mit einer Performanz und Qualität, wie man es aus dem konventionellen Bereich gewohnt ist. Dies liegt unter anderem daran, dass Expertise aus dem Bereich der Entwurfsautomatisierung für konventionelle Schaltungen und Systeme bisher noch nicht durchgehend in die "Quantenwelt" übertragen werden konnte.

Die Dissertation [Zu19] setzt das ambitionierte Ziel, dies zu ändern und Expertise aus jahrzehntelanger Forschung im Bereich der Entwurfsautomatisierung für den Quantencomputer nutzbar zu machen. Dies führte zur Entwicklung zahlreicher Methoden, welche von Kerntechniken der Entwurfsautomatisierung inspiriert wurden und damit die oben angeführten Aufgaben deutlich effizienter und besser bewältigen. Umfangreiche Evaluationen bestätigen, dass dadurch Ergebnisse erzielt werden können, welche die Ergebnisse der bisherigen Ansätze um mehrere Größenordnungen (in Bezug auf Laufzeit und Qualität) verbessern. In den folgenden Abschnitten werden die im Rahmen der Dissertation entwickelten Ansätze kurz vorgestellt und ihr Potential diskutiert. Sie werden zeigen, dass die Nutzung von Methoden der Entwurfsautomatisierung im Bereich des Quantencomputing letztlich wesentlich für die Etablierung dieser neuen Technologie in praxisrelevanten Anwendungen ist.

2 Simulation von Quantenschaltungen

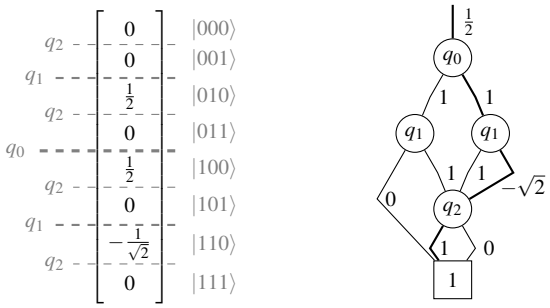
Die Simulation von Quantenschaltungen ist ein rechenintensiver Prozess, da der Zustand eines Quantensystems üblicherweise mit Hilfe eines komplexen Vektors mit exponentieller Größe beschrieben wird. Unitäre Matrizen, die ebenfalls exponentiell mit der Anzahl der involvierten Qubits wachsen, beschreiben Operationen der einzelnen Quantengatter. Die Simulation erfolgt durch schrittweise Multiplikation aller Matrizen der Gatter mit dem initialen Zustandsvektor.[2] Da der Zustandsvektor exponentiell mit der Anzahl der Qubits anwächst, sind selbst Supercomputer mit Petabytes an Speicher und tausenden Kernen auf Simulationen von Quantenschaltungen mit weniger als 50 Qubits limitiert [SSA16].

Um dieses Problem (zumindest in einigen Fällen) zu umgehen, liefert die Dissertation einen komplementären Ansatz, der auf Entscheidungsdiagrammen (eng.: decision diagrams [Br86]) basiert. Entscheidungsdiagramme haben sich bereits beim Entwurf von konventionellen Schaltungen und Systemen als überaus mächtig erwiesen und erlauben die kompakte Repräsentation zahlreicher Schaltungsbeschreibungen (z.B. in Form von Booleschen Funktionen). Die generelle Idee ist dabei, Redundanzen in der Beschreibung der Zustände und Operationen durch geteilte Strukturen zu beschreiben.

Im Kontext der Dissertation wurde eine für den Bereich Quantencomputing spezialisierte Form von Entscheidungsdiagrammen entwickelt, welche den Zustandsvektor sowie die unitären Matrizen in vielen Fällen kompakter (d.h. in nicht-exponentieller Größe) zu repräsentieren vermag. Dazu werden die jeweiligen Vektoren und Matrizen zur Beschreibung der Zustände und Operationen eines Quantensystems zerlegt und daraus resultierende Redundanzen ausgenutzt. Das soll im Folgenden anhand von Vektoren, welche Zustände eines Quantensystems beschreiben, erläutert werden.

Konkret wird o.B.d.A. ein Quantensystem bestehend aus den Qubits $q_0, q_1, \ldots q_{n-1}$ (wobei q_0 das höchstwertige Qubit darstellt) mit Hilfe eines Vektors der Größe 2^n beschrieben, welche für jede der 2^n möglichen Basiszustände die jeweiligen Amplituden $\alpha_{00\ldots0}$, $\alpha_{00\ldots1}$, $\ldots$, $\alpha_{11\ldots1}$ enthält. Die oberen 2^{n-1} Einträge beschreiben damit die Amplituden der Basiszustände mit $q_0 = |0\rangle$, während die unteren 2^{n-1} Einträge die Amplituden für die Basiszustände mit $q_0 = |1\rangle$ beschreiben. In einem Entscheidungsdiagramm wird dieser Zusammenhang nun durch einen Knoten mit der Markierung q_0 und zwei ausgehenden Kanten (einer linken Kante für den oberen Teilvektor und einer rechten für den unteren Teilvektor) dargestellt. Führt man diese Zerlegung rekursiv fort, so erhält man letztendlich Knoten, die einzelne Einträge enthalten (d.h. Vektoren der Größe 1). Eine kompaktere Darstellung des Entscheidungsdiagramms erhält man dann, indem identische Teilvektoren durch mehrfach genutze einzelne Knoten repräsentiert werden. Weitere Konzepte wie Kantengewichte ermöglichen zusätzliche Reduktionen in der Darstellungsgröße.

[2] Alternative Simulationsverfahren existieren, welche nicht alle Amplituden des finalen Zustandsvektor berechnen (sofern dies der Anwendungsfall erlaubt) [Sh17]. Dies ändert jedoch nichts an der exponentiellen Komplexität der Simulation.



(a) Zustandsvektor (b) Entscheidungsdiagramm

Abb. 1: Repräsentationen des Zustands eines Quantensystems

Beispiel 1. *Abbildung 1a zeigt einen Zustandsvektor eines Quantensystems bestehend aus drei Qubits, d.h. die Amplituden der* $2^3 = 8$ *möglichen Basiszustände (die rechts vom Vektor annotiert sind). Abbildung 1b zeigt das dazu gehörige Entscheidungsdiagramm. Um nun z.B. den Wert der Amplitude für* $|q_0 q_1 q_2\rangle = |110\rangle$ *zu erhalten, folgt man im Entscheidungsdiagramm zuerst zweimal dem rechten Nachfolger (für* $q_0 = 1$ *und* $q_1 = 1$*) und danach dem linken Nachfolger (für* $q_2 = 0$*). Dabei werden alle Gewichte entlang des Pfades multipliziert, d.h.* $|110\rangle = \frac{1}{2} \cdot 1 \cdot -\sqrt{2} \cdot 1 = -\frac{1}{\sqrt{2}}$*. Auf diese Weise lassen sich alle Amplituden kompakter darstellen als mit der bisher üblichen Vektor-Notation (konkret reichen vier Knoten im Vergleich zu acht Vektoreinträgen).*

Die oftmals viel kompaktere Darstellung von Vektoren und Matrizen erlaubt es, den für die Repräsentation von Quantenzuständen und Quantenoperationen nötigen Speicherbedarf enorm zu reduzieren. Da zudem die nötigen Matrix-Vektor-Multiplikationen direkt in den Entscheidungsdiagrammen ausgeführt werden können und deren Komplexität hier nur polynomial mit der Anzahl der Knoten wächst, kann dadurch die Simulationszeit enorm verringert werden. Dies ermöglicht es, in vielen relevanten Situationen den aktuellen Stand der Technik deutlich zu übertreffen. Beispielsweise konnte die Simulation einer Instanz des Shors Algorithmus (zur Faktorisierung von Semiprimzahlen) in weniger als zwei Minuten durchführt werden, während z.B. Microsofts Simulator LIQ*Ui*$|\rangle$ [WS14] mehr als 30 Tage für diese Aufgabe benötigte.

Implementierungen des Ansatzes (unter anderem mit verschiedenen weiteren Optimierungen und Anpassungen, die im Detail in der Dissertation [Zu19] beschrieben sind) wurden als Open Source-Projekte öffentlich gemacht. Neben der akademischen Community wurde dieser komplementäre Ansatz auch bereits von großen, im Bereich Quantencomputing tätigen Firmen aufgegriffen und anerkannt. So führten die Arbeiten am Simulator zu einem *Google Research Award* und die Integration der Implementierung in IBMs SDK *Qiskit* sowie der *Quantum Learning Machine* von Atos. Dies unterstreicht den Einfluss, den die Nutzung einer Kernmethode der Entwurfsautomatisierung (nämlich Entscheidungsdiagramme) und dessen Anpassung für den Bereich des Quantencomputings in diesem Feld haben kann.

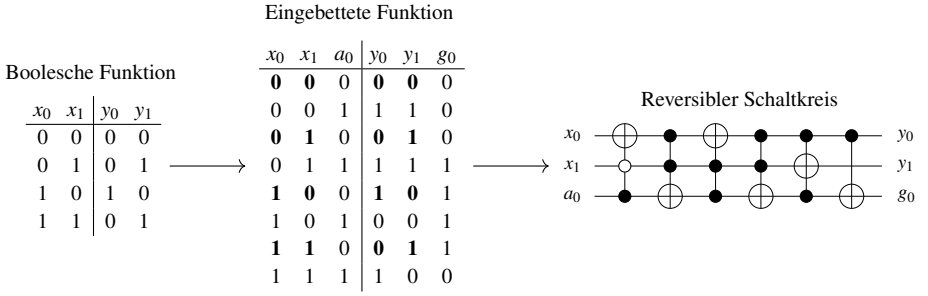


Abb. 2: Funktionale Synthese reversibler Schaltungen

3 Entwurf von Booleschen Komponenten

Die Dissertation bietet neue Ansätze zur Synthese von reversiblen Schaltungen, die für den Entwurf Boolescher Komponenten in Quantenalgorithmen benötigt werden. Obwohl diese Booleschen Komponenten (auch *Orakel* genannt) aufgrund der Reversibilität von Quantenoperationen inhärent reversibel sind, werden sie wegen ihrer Komplexität üblicherweise in nicht-reversible Komponenten zerlegt. Dies führt teils zu beträchtlichen Mehraufwänden, da diese nicht-reversiblen Teile erneut in eine reversible Funktion eingebettet werden müssen. Dies erfordert in der Regel weitere Qubits (sogenannte Hilfsqubits), welche durch die beschränkte Anzahl an physikalisch verfügbaren Qubits aber oft stark limitiert sind.

Beispiel 2. *Abbildung 2 skizziert die Synthese einer nicht-reversiblen Funktion in einen reversiblen Schaltkreis. Die in der linken Tabelle gegebene Boolesche Funktion ist nicht reversibel, da die Ausgabe* $y_0y_1 = 01$ *zweimal definiert ist (man also nicht eindeutig aus dem Ausgangsmuster das jeweilige Eingangsmuster ableiten kann). Um dies zu beheben muss eine zusätzliche Variable (ein Hilfsqubit) hinzugefügt werden, die in einem der beiden Fälle den Wert* 0 *und im anderen Fall den Wert* 1 *annimmt. Eine mögliche einbettende reversible Funktion wird in der Mitte von Abbildung 2 gezeigt. Diese Funktion wird dann mittels Syntheseansatz in einen reversiblen Schaltkreis bestehend aus Toffoli Gatter übersetzt (siehe rechte Seite von Abbildung 2).*[3]

Die große Herausforderung dieser Schritte liegt nun zum Einen in der begrenzten Skalierbarkeit der bisher verfügbaren Ansätze (die oft noch auf Wahrheitstabellen beruhen und dadurch exponentiell in der Größe der Eingangsvariablen wachsen) und zum Anderen in den zusätzlich benötigten Qubits. In der Dissertation wurden hierfür verschiedene Ideen entwickelt, die diese Probleme adressieren. Dies führte unter anderem zu einem Ansatz, der die Einbettung einer nicht-reversiblen Funktion mit der *minimalen* Anzahl an Hilfsqubits durchführt und dabei trotzdem für große Funktionen skalierbar bleibt. Ähnlich wie bei der Simulation zuvor werden dabei die einzubettenden Funktionen in Form von Matrizen beschrieben, die schließlich effizient mit Hilfe von Entscheidungsdiagrammen repräsentiert und bearbeitet werden können.

[3] Die genaue Semantik der Gatter ist in der Dissertation [Zu19] natürlich erläutert, für die Diskussion im Folgenden aber nicht nötig.

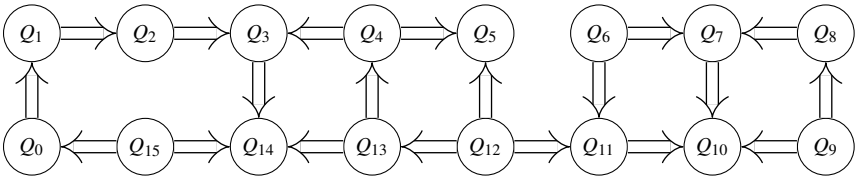


Abb. 3: IBM Q 16 Rueschlikon V1.0.0 (IBM QX3) [IB]

Weitere Verbesserungen werden erreicht, indem die beiden Schritte der Einbettung und der Synthese miteinander kombiniert werden (was zu einer sogenannten One-Pass Synthese führte). Dies verbessert die Skalierbarkeit der funktionalen Synthese weiter, da die originale (nicht-reversible) Funktion mit weniger Variablen arbeitet und dadurch das exponentielle Wachstum verringert wird. Gleichzeitig können dadurch die Kosten der resultierenden Schaltung deutlich verringert werden, da Freiheitsgrade nun gleichzeitig für Einbettung *und* Synthese ausgenutzt werden können. Und schließlich wurden noch Verfahren vorgestellt, welche mit Hilfe von Kodierungen der Ausgangsmuster erreichten, dass die Anzahl der benötigen Hilfsqubits unter die bisher bekannte minimale Anzahl reduziert werden konnte.

Insgesamt konnten damit neuartige und komplementäre Ansätze für die Synthese Boolescher Komponenten vorgestellt werden, die sowohl in Skalierbarkeit als auch Qualität dem bisherigen Stand der Technik deutlich überlegen sind.

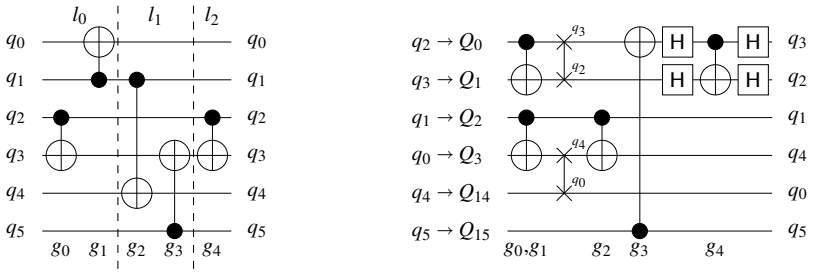
4 Abbildung von Quantenschaltkeisen auf reale Quantencomputer

Um Quantenschaltungen auf entsprechenden Quantencomputern auszuführen, müssen diese auf die Zielarchitektur abgebildet werden.[4] Genauer gesagt müssen die n logischen Qubits des Schaltkreises auf die $m \geq n$ physikalischen Qubits eines Quantencomputers abgebildet werden, während gleichzeitig sogenannte Coupling-Bedingungen erfüllt werden. Diese Bedingungen resultieren aus der physikalischen Limitation, dass nicht alle Qubits beliebig miteinander interagieren können. Tatsächlich werden die jeweils möglichen Interaktionen einer Architektur beim Entwurf eines Quantenrechners definiert und in Form einer sogenannten Coupling-Map beschrieben.

Beispiel 3. *Abbildung 3 zeigt die Coupling-Map von IBMs QX3 Quantencomputer. Die Knoten beschreiben die physikalischen Qubits, während die Kanten angeben, welche Qubits miteinander interagieren können. Die Richtung der Kante gibt zusätzlich an, welche Qubits die Operation kontrolliert und auf welches Qubit die Operation angewendet wird.*

Mit Ausnahme von trivialen Fällen existiert üblicherweise keine Abbildung, die alle Coupling-Bedingungen erfüllt. Daher werden zusätzliche Quantenoperationen (beispielsweise SWAP Operationen, die den Zustand von zwei physikalischen Qubits tauschen) in den Schaltkreis eingefügt, um die Abbildung dynamisch zu ändern.

[4] Die Dissertation geht nicht auf die Übersetzung in elementare Operationen ein, da hier bereits einige effiziente Methoden verfügbar sind [Am13].



(a) Originaler Schaltkreis (b) Abgebildeter Schaltkreis

Abb. 4: Abbildung eines Quantenschaltkreises auf IBM QX3

Beispiel 4. *Abbildung 4a zeigt einen Schaltkreis mit sechs logischen Qubits $q_0, q_1, \ldots q_5$, der auf IBM QX3 abgebildet werden soll. Abbildung 4b zeigt eine entsprechende Lösung, die alle Coupling-Bedingungen erfüllt. Die initiale Abbildung der Qubits ist dabei an der linken Seite von Abbildung 4b dargestellt. Die ersten beiden Gatter können damit problemlos ausgeführt werden, da die Interaktionen der betreffenden Qubits laut der Coupling-Map von Abbildung 3 möglich ist. Diese initiale Abbildung erlaubt aber nicht mehr die Ausführung des dritten und vierten Gatters. Entsprechend sind im Schaltkreis von Abbildung 4b SWAP Gatter hinzugefügt, welche den Zustand der Qubits tauschen, so dass die betreffenden logischen Qubits auf physikalische Qubits abgebildet werden, die miteinander interagieren können.*

Die Frage bleibt, wie diese Abbildung durchgeführt und gleichzeitig die Anzahl der zusätzlich benötigen SWAP-Gatter (die Kosten und Fehleranfälligkeit erhöhen) minimiert werden kann—ein NP-hartes Problem [BKM18]. Die Dissertation liefert mehrere Ansätze für dieses Problem.

So stellt sie ein erstes exaktes Verfahren vor, dass Lösungen mit einer minimalen Anzahl SWAP-Gatter ermittelt. Hierzu wurde eine symbolische Beschreibung des Problems entwickelt, die anschließend mit Hilfe von SAT-Beweisern bearbeitet wurde. Dadurch war erstmals die Ermittlung unterer Schranken für dieses Problem möglich. Außerdem wurde das noch freizulegende Potential aufgezeigt: So liefert IBMs eigener Ansatz zur Lösung des Problems im Schnitt um mehr als 100 % teurere Lösungen als eigentlich nötig—und das bereits bei sehr kleinen Schaltungen.

Dadurch motiviert wurden weitere Heuristiken entwickelt, die das Problem auch skalierbar lösbar machen. Erneut wurden dabei Methoden der Entwurfsautomatisierung genutzt; genauer gesagt wurden A*-Suchalgorithmen mit speziellen Optimierungen verwendet. Mit zusätzlichen Vor- und Nachbearbeitungen konnte schließlich eine Compilation-Methode entwickelt werden, die erneut derzeitigen Lösungen stark überlegen ist.

Der entwickelte Ansatz wurde zum Gewinner der *IBM Qiskit Developer Challenge* gekürt, und ebenfalls in IBMs SDK Qiskit und der Quantum Learning Machine von Atos integriert.

5 Schlussbemerkungen

Die Dissertation liefert substanzielle Beiträge für die Nutzung von Quantencomputern. Wichtige Entwurfsaufgaben wie die Simulation, der Entwurf Boolescher Komponenten und die Abbildung auf reale Quantencomputer konnten effizienter und in deutlich höherer Qualität als bisher adressiert werden. Dies wurde durch die gezielte Nutzung von Methoden und Expertise aus dem Bereich der Entwurfsautomatisierung für konventionelle Schaltungen und Systeme möglich.

Die Auswirkungen der gemachten Entwicklungen sind bereits jetzt sehr deutlich: In drei Jahren Entwicklungszeit entstanden Methoden, welche nicht nur den Gewinn der *IBM Qiskit Developer Challenge* oder die Auszeichnung des *Google Research Award* ermöglichten, sondern auch schon jetzt in Tools von IBM und Atos integriert sind. Dies zeigt deutlich das Potential der entwickelten Methoden und den Effekt, den Entwurfsautomatisierung auch im Bereich Quantencomputing hat und haben wird.

Literaturverzeichnis

[Am13] Amy, Matthew; Maslov, Dmitri; Mosca, Michele; Roetteler, Martin: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 32(6):818–830, 2013.

[Ar19] Arute, Frank; Arya, Kunal; Babbush, Ryan; Bacon, Dave; Bardin, Joseph C; Barends, Rami; Biswas, Rupak; Boixo, Sergio; Brandao, Fernando GSL; Buell, David A et al.: Quantum supremacy using a programmable superconducting processor. Nature, 574(7779):505–510, 2019.

[BKM18] Botea, Adi; Kishimoto, Akihiro; Marinescu, Radu: On the Complexity of Quantum Circuit Compilation. In: Symposium on Combinatorial Search. 2018.

[Br86] Bryant, Randal E.: Graph-Based Algorithms for Boolean Function Manipulation. IEEE Trans. on Computers, 35(8):677–691, 1986.

[BRB90] Brace, Karl S; Rudell, Richard L; Bryant, Randal E: Efficient Implementation of a BDD Package. In: Design Automation Conf. S. 40–45, 1990.

[FDJ13] Fowler, Austin G; Devitt, Simon J; Jones, Cody: Surface code implementation of block code state distillation. Scientific reports, 3:1939, 2013.

[Go18] Gomes, Lee: Quantum Computing: Both Here and Not Here. IEEE Spectrum, 55(4):42–47, 2018.

[Gr96] Grover, Lov K.: A Fast Quantum Mechanical Algorithm for Database Search. In: Symposium on the Theory of Computing. S. 212–219, 1996.

[IB] IBM Q team: , IBM Q 16 Rueschlikon backend specification V1.0.0. `https://ibm.biz/qiskit-rueschlikon`. Accessed: 2019-06-15.

[IBM19] IBM Unveils World's First Integrated Quantum Computing System for Commercial Use. `https://newsroom.ibm.com/2019-01-08-IBM-Unveils-Worlds-First-Integrated-Quantum-Computing-System-for-Commercial-Use`, 2019. Accessed: 2019-06-15.

[Mo16] Montanaro, Ashley: Quantum algorithms: an overview. npj Quantum Information, 2:15023, 2016.

[NC02] Nielsen, Michael A; Chuang, Isaac: Quantum Computation and Quantum Information. AAPT, 2002.

[Pr18] Preskill, John: Quantum Computing in the NISQ era and beyond. Quantum, 2:79, 2018.

[Sh97] Shor, Peter W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Jour. of Comp., 26(5):1484–1509, 1997.

[Sh17] Shi, Andrew: Recursive Path-Summing Simulation of Quantum Computation. arXiv preprint arXiv:1710.09364, 2017.

[SHR18] Soeken, Mathias; Haener, Thomas; Roetteler, Martin: Programming quantum computers using design automation. In: Design, Automation and Test in Europe. IEEE, S. 137–146, 2018.

[SSA16] Smelyanskiy, Mikhail; Sawaya, Nicolas P. D.; Aspuru-Guzik, Alán: qHiPSTER: The Quantum High Performance Software Testing Environment. arXiv preprint arXiv:1601.07195, 2016.

[WS14] Wecker, Dave; Svore, Krysta M: LIQUi|>: A Software Design Architecture and Domain-Specific Language for Quantum Computing. arXiv preprint arXiv:1402.4467, 2014.

[Zu19] Zulehner, Alwin: Design Automation for Quantum Computing. dissertation, Johannes Kepler University, Linz, 2019.

Alwin Zulehner erhielt seinen Master- und Doktortitel von der Johannes Kepler Universität Linz in Österreich (2015 beziehungsweise 2019). Sein Forschungsinteresse liegt im Bereich der Entwurfsautomatisierung für Quantencomputing. In diesem Bereich hat Alwin Zulehner zahlreiche Paper bei renommierten internationalen Konferenzen und in Fachzeitschriften veröffentlicht, wie beispielsweise den *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems* (TCAD), der *Asia and South Pacific Design Automation Conference* (ASP-DAC), der *Design, Automation and Test in Europe* (DATE) Konferenz, der *Design Automation Conference* (DAC), und der *International Conference on Computer-Aided Design* (ICCAD). Für seine Beiträge wurde er zudem mit mehreren *Best Student Awards* und dem *Early Research Achievement Award* der JKU ausgezeichnet. Darüber hinaus hat er die *IBM Qiskit Developer Challenge* mit seinem Compiler für Quantencomputer gewonnen und wesentlich zu einem *Google Research Award* beigetragen.