Ein Entwurfsablauf für Reversible Schaltkreise

Robert Wille

Arbeitsgruppe Rechnerarchitektur
Fachbereich 3 – Mathematik und Informatik
Universität Bremen, 28359 Bremen
rwille@informatik.uni-bremen.de

Abstract: Heutige Computerchips stoßen durch die stetig steigende Miniaturisierung sowie das exponentielle Wachstum der Anzahl an Komponenten zunehmend an ihre Grenzen. Daher werden in Zukunft Alternativen benötigt, welche traditionelle Schaltungen ersetzen oder zumindest ergänzen. Reversible Schaltkreise haben sich hier als sehr viel versprechend erwiesen. Sie ermöglichen unter anderem Anwendungen im Low-Power Design, im Kontext von Quantenschaltungen sowie in den Gebieten des Optical Computings, des DNA Computings und in der Nanotechnologie. Allerdings war es bisher nicht möglich, große relevante Schaltkreise automatisch zu entwerfen.

In der hier vorgestellten Arbeit wird zum ersten Mal ein Entwurfsablauf für reversible Schaltkreise präsentiert, welcher die wesentlichen Schritte im Entwurf, nämlich Synthese, Optimierung, Verifikation und Debugging, umfasst. Dazu wurde eine Vielzahl neuer Verfahren entwickelt, welche deutlich größere Funktionen und Schaltkreise als bisher unterstützen. Die Kombination der entsprechenden Methoden erlaubt erstmalig den Entwurf großer reversibler Schaltkreise und ebnet damit den Weg für die Realisierung komplexer Systeme basierend auf reversibler Logik.

1 Einführung

Die enormen Fortschritte der Halbleiterindustrie in den vergangenen Jahrzehnten haben dazu geführt, dass integrierte Schaltungen mittlerweile jeden Bereich des alltäglichen Lebens erfassen. Computer und Mikrochips sind unersetzlich geworden. Weitere Verbesserungen werden regelrecht erwartet. Dabei ist offensichtlich, dass die Entwicklung der vergangenen Jahre nicht beliebig fortgesetzt werden kann. So hat sich seit den 70er Jahren die Anzahl an Komponenten auf einem Schaltkreis ca. alle 18 Monate verdoppelt (auch als Moore's Gesetz bekannt). Mit einem Fortschreiben dieses exponentiellen Wachstums würde in naher Zukunft die atomare Ebene erreicht. Abgesehen davon führt die stetig steigende Miniaturisierung dazu, dass immer mehr Energie in Form von Wärme abgegeben wird. Da dies die Verlässlichkeit eines Schaltkreises beinträchtigen oder ihn sogar zerstören kann, stellt diese Entwicklung ein großes Hindernis beim Bau heutiger Schaltungen dar. Daher wird derzeit intensiv an Alternativen zu traditionellen Technologien (wie CMOS) geforscht. Reversible Logik bietet hier viel versprechende Möglichkeiten.

Im Gegensatz zu traditioneller Logik (wie sie in heutigen Schaltkreisen verwendet wird), realisieren Schaltkreise über reversibler Logik bijektive Funktionen. Das heißt, aus einer Belegung der Eingänge lässt sich nicht nur auf die Belegung der Ausgänge schließen, sondern umgekehrt lassen sich die Eingangsbelegungen auf Basis der Ausgangsbelegungen ermitteln. Dies steht diametral zu den bisher verwendeten Technologien. So sind

herkömmliche Gatter, wie *UND*, *ODER* und *EXOR*, nicht reversibel und können daher nicht verwendet werden. Stattdessen ist eine Reihe von speziellen reversiblen Gattern entwickelt worden. Diese ermöglichen Anwendungen in neuen Technologien, wovon zwei im Folgenden exemplarisch skizziert werden:

Reversible Logik im Low-Power Design
Die Energieabgabe und verbunden damit die Wärmeerzeugung ist für heutige Computerchips ein ernstes Problem. Dabei haben die verwendeten Materialien einen großen Einfluss auf die Wärmeabgabe. Ein viel fundamentaleres Problem wurde aber bereits in den 60ern bzw. 70ern aufgezeigt: So bewies Landauer in [Lan61], dass irreversible Operationen (unabhängig von der verwendeten Technologie) immer zu einer Energieabgabe führen. Später zeigte Bennett in [Ben73], dass Energieverluste nur komplett beseitigt werden können, wenn Berechnungen ohne Informationsverlust durchgeführt werden. Dies trifft für reversible Logik zu, da hier Eingabedaten bijektiv (und damit ohne Informationsverlust) transformiert werden. Entsprechend wird reversible Logik auch immer interessanter für Anwendungen im Low-Power Design. Im Jahr 2002 konnte bereits ein reversibler Schaltkreis produziert werden, dessen Energieverbrauch so gering ist, dass er nur durch die Energie der Eingangssignale betrieben werden konnte [DV02].

Reversible Logik als Grundlage für Quantenschaltkreis
Quantenschaltkreise [NC00] bieten eine komplett neue Möglichkeit, Berechnungen durchzuführen. Anstatt mit traditionellen Bits, wird hierbei mit so genannten Qubits gearbeitet. Diese erlauben nicht nur die Repräsentation der Zustände 0 und 1, sondern auch der Superposition von beiden. Damit können Qubits viele Zustände gleichzeitig repräsentieren und somit bestimmte Probleme deutlich schneller als bisherige Verfahren lösen. So kann z.B. das Faktorisierungsproblem mit Hilfe von Quantenalgorithmen in polynomieller Zeit gelöst werden [NC00] – für traditionelle Rechner sind bisher nur exponentielle Verfahren bekannt. Für kleine Zahlen können diese Verfahren auch schon physikalisch umgesetzt werden (siehe z.B. [VSB$^+$01]). Die jeweiligen Quantenoperationen sind dabei reversibel. Daher finden Fortschritte im Bereich der reversibler Logik zugleich auch Anwendung bei Quantenschaltkreisen.

Darüber hinaus können weitere Anwendungen von reversibler Logik im Gebiet des Optical Computings [CA87], des DNA Computings [Ben73] und in der Nanotechnologie gefunden werden [Mer93].

Den vielfältigen Anwendungsmöglichkeiten von reversibler Logik stehen allerdings auch einige Beschränkungen gegenüber. So sind aufgrund der Reversibilität Verzweigungspunkte innerhalb der Schaltung (Fanouts) wie auch Rückkopplungen (z.B. durch Flip-Flops) nicht direkt erlaubt [NC00]. Als Konsequenz lässt sich reversible Logik nur als eine Kaskade (d.h. Hintereinanderschaltung) von reversiblen Gattern realisieren. Dies führt dazu, dass sich der Entwurf reversibler Schaltkreise deutlich von denen traditioneller Chips unterscheidet. Die Forschung an entsprechenden Verfahren steht aber noch sehr am Anfang. So war es zum Beispiel bisher nicht möglich, große (reversible) Funktionen automatisch zu synthetisieren. Als Folge dieser begrenzten Synthesefähigkeiten wurden auch die weiteren Schritte im Entwurfsablauf nur ansatzweise untersucht.

In der hier vorgestellten Dissertation wurden neue Methoden vorgestellt, welche diese Stagnation durchbrechen. Dabei werden Beiträge zu den wesentlichen Schritten im Entwurf, nämlich Synthese, Optimierung, Verifikation und Debugging, geleistet und in einen integrierten Ablauf zusammengeführt. Abbildung 1 zeigt die Interaktion der jeweiligen

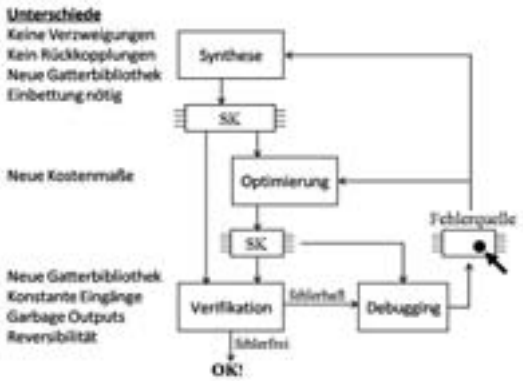


Abbildung 1: Entwurfsablauf für reversible Schaltkreise

Schritte innerhalb des entwickelten Ablaufs. Auf der linken Seite sind dabei die wesentlichen Unterschiede zum traditionellen Entwurf skizziert. Durch Kombination der vorgestellten Techniken ist es erstmals möglich, reversible Schaltkreise zu synthetisieren, welche deutlich größere Funktionen als bisher realisieren. Optimierungsansätze stellen nachfolgend sicher, dass die resultierenden Schaltungen möglichst geringe Kosten besitzen. Die Korrektheit kann anschließend durch Verfikationsmethoden (hier Äquivalenzprüfer) sichergestellt werden. Sollte ein Fehlverhalten im Schaltkreis entdeckt worden sein, helfen Debugging-Ansätze bei der Fehlersuche.

Im Folgenden wird der entwickelte Entwurfsablauf vorgestellt. Da eine ausführliche Darstellung der verschiedenen Verfahren hier nicht möglich ist, werden die jeweiligen Herausforderungen der einzelnen Schritte sowie die generellen Ideen einiger ausgewählter Ansätze kurz zusammengefasst bzw. skizziert. Der nächste Abschnitt bildet dafür die Basis, in dem er die wichtigsten Grundlagen zu reversiblen Funktionen und reversiblen Schaltkreisen einführt.

2 Reversible Schaltkreise

Reversible Schaltkreise [Lan61, Ben73, Tof80] realisieren Funktionen $f : \mathbb{B}^n \rightarrow \mathbb{B}^n$ über $X = \{x_1, \ldots, x_n\}$, wobei jede mögliche Eingangsbelegung auf eine eindeutige Ausgangsbelegung abgebildet wird (d.h. Bijektionen werden realisiert). Aufgrund der Reversibilität sind Verzweigungen (Fanouts) und Rückkopplungen (z.B. durch FlipFlops) nicht direkt erlaubt [NC00]. Daher sind alle reversible Schaltkreise Kaskaden (d.h. Hintereinanderschaltungen) von reversiblen Gattern.

Ein reversibles Gatter hat dabei die Form $g(C, T)$, wobei $C = \{x_{i_1}, \ldots, x_{i_k}\} \subset X$ eine Menge von so genannten *Control Lines* und $T = \{x_{j_1}, \ldots, x_{j_l}\} \subset X$ mit $C \cap T = \emptyset$ eine Menge von so genannten *Target Lines* darstellt. Die Menge C kann dabei auch leer sein. Wenn alle Signale der Menge C (d.h. alle *Control Lines*) mit dem Wert 1 belegt sind oder keine *Control Lines* existieren ($C = \emptyset$), wird auf den *Target Lines* die jeweilige Operation des Gatters ausgeführt. Die Werte der übrigen Signale werden ohne Veränderung weitergeleitet.

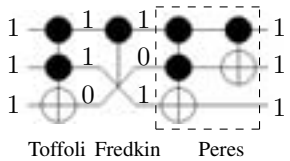


Abbildung 2: Reversible Gatter

In den vergangenen Jahren haben sich unter anderem die folgenden reversiblen Gatter durchgesetzt:

- *Multiple Control Toffoli Gatter* (zuerst vorgestellt in [Tof80]) besitzen lediglich eine *Target Line* x_j und bilden $(x_1, x_2, \ldots, x_j, \ldots, x_n)$ auf $(x_1, x_2, \ldots, x_{i_1} x_{i_2} \cdots x_{i_k} \oplus x_j, \ldots, x_n)$ ab. Das heißt, hier wird die *Target Line* invertiert, wenn alle *Control Lines* den Wert 1 haben oder keine *Control Line* existiert.
- *Multiple Control Fredkin Gatter* (zuerst vorgestellt in [FT82]) besitzen zwei *Target Lines* x_{j_1} und x_{j_2}. Die Werte dieser *Target Lines* werden vertauscht, wenn alle *Control Lines* den Wert 1 besitzen oder keine *Control Line* existiert.
- *Peres Gatter* (P) (zuerst vorgestellt in [Per85]) haben eine *Control Line* x_i und zwei *Target Lines* x_{j_1} sowie x_{j_2}. Sie bilden $(x_1, x_2, \ldots, x_{j_1}, \ldots, x_{j_2} \ldots, x_n)$ auf $(x_1, x_2, \ldots, x_i \oplus x_{j_1}, \ldots, x_i x_{j_1} \oplus x_{j_2}, \ldots, x_n)$ ab.

Abbildung 2 zeigt ein Toffoli Gatter, ein Fredkin Gatter und ein Peres Gatter in einer Kaskade. Die *Control Lines* werden dabei durch ●, die *Target Line(s)* durch $\oplus$ dargestellt (mit Ausnahme des Fredkin Gatters, bei dem stattdessen ein $\times$ verwendet wird). Die annotierten Werte demonstrieren eine mögliche Berechnung dieser Kaskade. Wie man sehen kann, ist diese Berechnung in beide Richtungen (d.h. reversibel) möglich.

Die jeweiligen Gattertypen sind dabei universell, das heißt mit den einzelnen Typen lassen sich alle möglichen reversiblen Funktionen realisieren. Allerdings unterscheiden sich die einzelnen Gatter in den dafür nötigen Kosten, welche je nach Anwendung bzw. Technologie anfallen. Im Gebiet der Quantenschaltkreise wurde bereits ein konkretes Maß eingeführt: die *Quantenkosten* [BBC$^+$95]. Diese geben an, wie viele Quantenoperationen nötig sind, um einen reversiblen Schaltkreis als Quantenschaltkreis zu realisieren. Analog stellen *Transistorkosten* [TG08] ein Maß für die Umsetzung reversibler Logik in CMOS Schaltungen dar. Außerdem kommen mit den Fortschritten in der physikalischen Realisierung reversibler Schaltungen stets neue bzw. konkretere Bedingungen hinzu, welche zu weiteren (technologiespezifischen) Kostenmaßen führen. Ziel beim Entwurf der jeweiligen Schaltungen ist es, die gewünschte Funktion korrekt mit möglichst geringen Kosten (abhängig von der Zieltechnologie) zu realisieren.

3 Synthese Reversibler Schaltkreise

Die Synthese ist ein zentraler Schritt im Schaltkreisentwurf. Im traditionellen Ablauf unterteilt sich die Synthese in weitere Teilschritte wie z.B. High-Level Synthese, Logiksyn-

Tabelle 1: Einbettung reversibler Funktionen

(a) Wahrheitstabelle des 1-Bit Addierers

c_{in}	x	y	c_{out}	sum	
0	0	0	0	0	0
0	0	1	**0**	**1**	**0**
0	1	0	**0**	**1**	**1**
0	1	1	*1*	*0*	*0*
1	0	0	**0**	**1**	**?**
1	0	1	*1*	*0*	*1*
1	1	0	*1*	*0*	*?*
1	1	1	1	1	1

(b) Wahrheitstabelle des eingebetteten Addierers

0	$\mathbf{c}_{in}$	**x**	**y**	$\mathbf{c}_{out}$	**sum**	g_1	g_2
0	**0**	**0**	**0**	**0**	**0**	0	0
0	**0**	**0**	**1**	**0**	**1**	1	1
0	**0**	**1**	**0**	**0**	**1**	1	0
0	**0**	**1**	**1**	**1**	**0**	0	1
0	**1**	**0**	**0**	**0**	**1**	0	0
0	**1**	**0**	**1**	**1**	**0**	1	1
0	**1**	**1**	**0**	**1**	**0**	1	0
0	**1**	**1**	**1**	**1**	**1**	0	1
1	0	0	0	1	0	0	0
1	0	0	1	1	1	1	1
1	0	1	0	1	1	1	0
1	0	1	1	0	0	0	1
1	1	0	0	1	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	1	0	1

these, Technologieabbildung und Routing. Sollen reversible Schaltungen realisiert werden, müssen weitere Aspekte beachtet werden. So lassen sich zum Beispiel irreversible Funktionen nur nach vorheriger *Einbettung* in reversibler Logik realisieren. Anschließend müssen bei der eigentlichen Synthese die herrschenden Einschränkungen (zum Beispiel das Fehlen von Verzweigungen) sowie die neuen Gattertypen berücksichtigt werden. Hierzu wurde eine Vielzahl an Ansätzen entwickelt. Im Folgenden werden die jeweiligen Kernprobleme bzw. -ideen dazu beschrieben.

3.1 Einbetten Irreversibler Funktionen

Tabelle 1(a) zeigt die Wahrheitstabelle eines 1-Bit Addierers. Dieser besitzt drei Eingänge (ein Übertragsbit c_{in} sowie die beiden Summanden x and y) und zwei Ausgänge (das Übertragsbit c_{out} und die Summe *sum*). Damit ist der Addierer klar erkennbar irreversibel, da (1) die Anzahl an Eingängen nicht mit der Anzahl an Ausgängen übereinstimmt und (2) die Funktion keine bijektive Abbildung darstellt. Selbst das Hinzufügen eines weiteren Ausgangs (was zu einer identischen Anzahl an Eingängen und Ausgängen führt) würde diese Funktion nicht reversibel machen. Damit erreicht man lediglich, dass o.B.d.A. die ersten vier Zeilen der Wahrheitstabelle eindeutig abgebildet werden (siehe rechte Spalte in Tabelle 1(a)). Spätestens für die fünfte Zeile ist dies aber nicht mehr möglich, da $c_{out} = 0$ and $sum = 1$ bereits zweimal verwendet wurden und nicht mehr länger eindeutig eingebettet werden können.

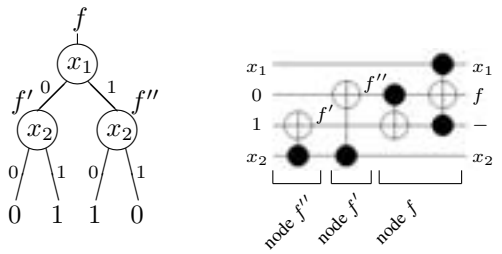
Dies führt dazu, dass – um irreversible Funktionen einzubetten – unter Umständen weitere Ausgänge nötig werden. Konkret müssen mindestens $\lceil log(m) \rceil$ Ausgänge hinzugefügt werden, wobei m die maximale Anzahl an gleichen Ausgangsmustern auf der rechten Seite der Wahrheitstabelle angibt. Angewandt auf den Addierer, bei welchem sich bis zu 3 Ausgangsmuster wiederholen, sind also $\lceil log(3) \rceil = 2$ zusätzliche Ausgänge (und verbunden damit eine weitere Leitung) nötig, um die Funktion reversibel einzubetten. Dies führt dabei zu konstanten Eingängen (die beliebig vom Entwerfer belegt werden können) sowie zu so genannten *Garbage Outputs* und damit zu unvollständig spezifizierten rever-

siblen Funktionen. Da viele Syntheseansätze aber eine komplett spezifizierte Funktion als Eingabe erwarten, müssen die nicht-spezifizierten Ausgaben schließlich noch mit konkreten Werten belegt werden. Eine mögliche (wenngleich auch naive) Einbettung des 1-Bit Addierers ist in Tabelle 1(b) dargestellt. Darüber hinaus sind aber zahlreiche weitere Einbettungen möglich. So existieren jeweils 4, 4, 3, 4, 2, 3, 2 und 4 Möglichkeiten die unspezifizierten Werte der ersten acht Zeilen zu belegen (insgesamt also 9216 Möglichkeiten). Der untere Teil der Wahrheitstabelle kann wiederum in $8!$ verschiedenen Weisen belegt werden. Schließlich besteht noch die Möglichkeit die Reihenfolge der Ausgänge der zu synthetisierenden Schaltung auf $4! = 24$ verschiedene Arten anzuordnen. Fasst man dies zusammen erhält man insgesamt $9216 \cdot 40320 \cdot 24 = 8.918.138.880$ Möglichkeiten einen 1-Bit Addierer als reversible Funktion einzubetten. Jede davon kann einen Einfluss auf das jeweilige Syntheseergebnis haben (d.h. je nach Einbettung kann ein größerer oder kleinerer Schaltkreis resultieren). Da eine erschöpfende Suche nach der besten Einbettung nicht machbar ist, wurden verschiedene Heuristiken und angepasste Syntheseverfahren dazu vorgestellt [MWD09, WGDD09].

3.2 Exakte Synthese reversibler Schaltkreise

Syntheseverfahren erzeugen für eine gegebene Funktion einen Schaltkreis, welcher diese repräsentiert. *Exakte* Synthesealgorithmen stellen dabei zusätzlich die Minimalität sicher, d.h. sie generieren Schaltkreise mit einer minimalen Anzahl von Gattern bzw. Kosten. Die Sicherstellung der Minimalität erfordert allerdings einen enormen Rechenaufwand, weshalb sich entsprechende Methoden nur auf sehr kleine Funktionen anwenden lassen. Trotzdem ist es sinnvoll, exakte Verfahren zu betrachten, da sie (1) die Synthese kleinerer Realisierungen als die derzeit bekannten erlauben, (2) die Evaluation von heuristischen Methoden (zumindest für kleine Funktionen) ermöglichen bzw. (3) für die Generierung von Basisbausteinen verwendet werden können.

Im Rahmen der Arbeit wurden verschiedene Ansätze zur exakten Synthese reversibler Schaltungen vorgestellt. Dabei wurde das Syntheseproblem als eine Sequenz von Entscheidungsproblemen (genauer: von Erfüllbarkeitsproblemen; kurz: SAT-Problemen) beschrieben. Für eine gegebene Funktion wird geprüft, ob sie sich mit $d = 1$ Toffoli Gattern realisieren lässt. Ist dies nicht möglich, wird d iterativ erhöht bis schließlich eine Realisierung ermittelt werden kann. Die jeweiligen Prüfungen („Existiert ein Schaltkreis mit d Gattern, welcher die gegebene Funktion f realisiert?“) wurden dabei als eine Instanz des Erfüllbarkeitsproblem (SAT-Instanz) kodiert und anschließend mit einem SAT-Beweiser gelöst. Da dieses Vorgehen aber nach wie vor zu hohen Rechenzeiten führt, wurden weitere Verbesserungen vorgestellt. So lässt sich die exakte Synthese durch die Kodierung auf höhere Abstraktionsebenen [GWDD09], die Ausnutzung problemspezifischen Wissens während des Suchprozesses [WG07] oder die Verwendung von Kodierungen mit Quantoren [WLDG08] weiter verbessern. In der Summe gelingt es mit Hilfe dieser Verfahren, die weltweit kompaktesten Schaltkreise für bestimmte Funktionen zu generieren. Insbesondere für die oben genannten Anwendungen (z.B Generierung von Basisbausteinen) ist dies von entscheidener Bedeutung.



(a) BDD (b) Resultierender Schaltkreis

Abbildung 3: BDD and Toffoli Schaltkreis für $f = x_1 \oplus x_2$

3.3 Heuristische Synthese für große Funktionen

Trotz der wichtigen Anwendungsmöglichkeiten von exakter Synthese ist man generell natürlich an der Realisierung größerer bzw. komplexerer Funktionen interessiert. Allerdings leiden auch die derzeitigen (heuristischen) Syntheseverfahren an einer sehr geringen Skalierbarkeit. So lassen sich damit nur Funktionen mit max. 30 Variablen realisieren. Dies liegt unter anderem in den verwendeten Datenstrukturen begründet: So basieren die meisten Verfahren auf einer Beschreibung der Funktion in Form einer Wahrheitstabelle oder ähnlicher Beschreibungen wie Permutationen (siehe z.B. [SPMH03, MMD03]) bzw. nutzen zur Synthese ein Prinzip, welches das komplette Durchlaufen einer Wahrheitstabelle erfordert.

Im Gegensatz dazu wurde ein neuer Ansatz vorgestellt, welcher erstmals die Synthese von Funktionen mit über 100 Variablen ermöglicht [WD09]. Dabei werden Schaltkreise mit Hilfe von *binären Entscheidungsdiagrammen* [Bry86] (engl.: binary decision diagrams; kurz: BDDs) erstellt. Ein BDD ist ein direkter, azyklischer Graph $G = (V, E)$ bestehend aus terminalen und nicht-terminalen Knoten, welcher eine Boolesche Funktion f repräsentiert. Jeder nicht-terminale Knoten $v \in V$ ist dabei einer Variable x_i aus der Definitionsmenge von f zugeordnet. Zudem wird in jedem nicht-terminalen Knoten $v \in V$ die Funktion f mit Hilfe der Shannon Dekomposition ($f = \overline{x}_i f_{x_i=0} + x_i f_{x_i=1}$) in zwei *Kofaktoren* $f_{x_i=0}$ und $f_{x_i=1}$ zerlegt. Besteht ein Kofaktor $f_{x_i=0}$ ($f_{x_i=1}$) nur noch aus einer Booleschen Konstante wird diese mit einem terminalen Knoten repräsentiert. Auf diese Art und Weise repräsentiert zum Beispiel der BDD aus Abbildung 3(a) die Funktion $f = x_1 \oplus x_2$. Durch Anwendung weiterer Techniken (wie Reduktionsregeln, die Verwendung spezieller Variablenordnungen oder Komplementkanten) können mit Hilfe von BDDs auch Funktionen mit mehr als hundert Variablen sehr kompakt dargestellt werden.

Hat man schließlich die Beschreibung einer Funktion in Form eines BDDs zur Verfügung, lässt sich daraus in linearer Zeit ein reversibler Schaltkreis synthetisieren, indem man die jeweiligen Knoten $v \in V$ durch eine passende Kaskade von reversiblen Gattern ersetzt. So führt zum Beispiel das erste Gatter aus der Schaltung in Abbildung 3(b) dazu, dass die Funktion des Knotens f' aus dem BDD von Abbildung 3(a) realisiert wird. Analog gilt dies für das zweite Gatter und den Knoten f''. Damit steht ein Schaltkreis zur Verfügung welcher bereits die Werte f' und f'' realisiert. Durch das Hinzufügen weiterer Gatter ist es damit schließlich möglich den letzten Knoten f und damit die komplette Funktion zu rea-

lisieren (siehe Abbildung 3(b)). Durch Anwendung dieses Prinzips auf alle weiteren Fälle (einschließlich der oben bereits genannten weiteren Techniken) lässt sich daraus schließlich ein effizienter Syntheseansatz für Funktionen mit mehr als 100 Variablen ableiten. Dies stellt einen wichtigen Meilenstein dar, da so erstmals größere Funktionen automatisch in reversibler Logik realisiert werden können.

4 Optimierung Reversibler Schaltkreise

Das primäre Ziel der Synthese ist es, Schaltungen zu erzeugen, welche die gewünschte Funktionalität realisieren. Darüber hinaus ist man daran interessiert, möglichst kompakte (d.h. kostengünstige) Schaltkreise zu erhalten. Allerdings lassen sich diese Ziele aber nicht gleichzeitig erreichen (unter anderem, da es je nach adressierter Technologie unterschiedliche Auffassungen einer „kostengünstigen“ Schaltung gibt). Daher werden in der Regel nach der Synthese die resultierenden Schaltkreise bzgl. bestimmter Kostenkriterien weiter optimiert. Dabei haben sich insbesondere die oben bereits erwähnten Quanten- oder Transistorkosten etabliert. Darüber hinaus kommen – motiviert durch aktuelle physikalische Realisierungen – stetig weitere Kriterien und damit Kostenmaße hinzu. Für beide Fälle wurden im entwickelten Entwurfsablauf entsprechende Optimierungsverfahren entwickelt [MWD10, WSD09].

5 Verifikation und Debugging Reversibler Schaltkreise

Schließlich ist neben der Realisierung der jeweiligen Schaltkreise auch die Überprüfung der Korrektheit ein essentieller Bestandteil eines Entwurfsablaufs. Durch die stetig steigende Komplexität ist für traditionelle Schaltungen die Verifikation mittlerweile sogar der dominierende Teil des Entwurfes. In der reversiblen Logik ist die Forschung hier noch sehr am Anfang. So müssen im Vergleich zur traditionellen Schaltkreisverifikation neben den neuen Gattertypen je nach Anwendung auch mehrwertige Signale, konstante Eingaben oder Garbage Outputs unterstützt werden. In der Arbeit wurden hierfür zwei verschiedene Ansätze präsentiert [WGMD09]. Durch Verwendung jener ist es möglich die größten derzeit verfügbaren reversiblen Schaltungen innerhalb weniger Sekunden automatisch auf Äquivalenz zu verifizieren.

Sollten dabei Fehler entdeckt werden, müssen diese gefunden und anschließend behoben (*debugged*) werden. Dabei handelt es sich um einen manuellen und dadurch sehr zeitintensiven Prozess. Dieser kann allerdings durch Diagnose- bzw. Debuggingwerkzeuge beschleunigt werden. Diese ermitteln Fehlerkandidaten, welche das falsche Verhalten einer Schaltung erklären und somit bei der Suche nach dem Fehler helfen. Während entsprechende Verfahren für den traditionellen Schaltkreisentwurf mittlerweile existieren (z.B. [SVAV05]), haben Untersuchen gezeigt, dass sich diese nicht unverändert für reversible Schaltkreise anwenden lassen. Dies liegt unter anderem darin begründet, dass durch die Reversibilität Fehler an beliebig vielen Stellen im Schaltkreis korrigiert und somit die tatsächlichen Fehlerstellen nur schwer identifiziert werden können. Daher wurde ein neuer Debugging-Ansatz [WGF+09] vorgestellt, welcher die besonderen Eigenschaften reversibler Schaltungen berücksichtigt und damit kleine Mengen von Fehlerkandidaten liefert.

6 Zusammenfassung

Reversible Schaltkreise stellen eine viel versprechende Alternative zu traditionellen Technologien dar. Erste Anwendungen im Bereich des Low-Power Designs und der Quantenschaltkreise konnten bereits erfolgreich umgesetzt werden. Trotzdem existiert bis heute noch kein durchgängiger Ablauf zum Entwurf entsprechender Schaltungen. Die Verfahren, welche hier kurz zusammengefasst wurden, lassen sich zu einem ersten durchgängigem Entwurfsablauf zusammen fassen. Mit ihnen ist es erstmals möglich, große Funktionen effizient zu synthetisieren, sie anschließend gemäß unterschiedlicher Kostenkriterien zu optimieren und schließlich auf ihre Korrektheit hin zu verifizieren. Für den Fehlerfall stehen darüber hinaus Methoden zum Debugging zur Verfügung.

Literatur

[BBC+95] A. Barenco, C. H. Bennett, R. Cleve, D.P. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin und H. Weinfurter. Elementary Gates for Quantum Computation. *The American Physical Society*, 52:3457–3467, 1995.

[Ben73] C. H. Bennett. Logical Reversibility of Computation. *IBM J. Res. Dev*, 17(6):525–532, 1973.

[Bry86] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[CA87] R. Cuykendall und D. R. Andersen. Reversible optical computing circuits. *Optics Letters*, 12(7):542–544, 1987.

[DV02] B. Desoete und A. De Vos. A reversible carry-look-ahead adder using control gates. *INTEGRATION, the VLSI Jour.*, 33(1-2):89–104, 2002.

[FT82] E. F. Fredkin und T. Toffoli. Conservative Logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.

[GWDD09] D. Große, R. Wille, G. W. Dueck und R. Drechsler. Exact Multiple Control Toffoli Network Synthesis with SAT Techniques. *IEEE Trans. on CAD*, 28(5):703–715, 2009.

[Lan61] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.*, 5:183, 1961.

[Mer93] R. C. Merkle. Reversible electronic logic using switches. *Nanotechnology*, 4:21–40, 1993.

[MMD03] D. M. Miller, D. Maslov und G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In *Design Automation Conf.*, Seiten 318–323, 2003.

[MWD09] D. M. Miller, R. Wille und G.W. Dueck. Synthesizing Reversible Circuits for Irreversible Functions. In *EUROMICRO Symp. on Digital System Design*, Seiten 749–756, 2009.

[MWD10] D. M. Miller, R. Wille und R. Drechsler. Reducing Reversible Circuit Cost by Adding Lines. In *Int'l Symp. on Multi-Valued Logic*, 2010.

[NC00] M. Nielsen und I. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[Per85] A. Peres. Reversible logic and quantum computers. *Phys. Rev. A*, (32):3266–3276, 1985.

[SPMH03] V. V. Shende, A. K. Prasad, I. L. Markov und J. P. Hayes. Synthesis of reversible logic circuits. *IEEE Trans. on CAD*, 22(6):710–722, 2003.

[SVAV05] A. Smith, A. G. Veneris, M. F. Ali und A. Viglas. Fault diagnosis and logic debugging using Boolean satisfiability. *IEEE Trans. on CAD*, 24(10):1606–1621, 2005.

[TG08] M. K. Thomson und R. Glück. Optimized Reversible Binary-coded Decimal Adders. *J. of Systems Architecture*, 54:697–706, 2008.

[Tof80] T. Toffoli. Reversible Computing. In W. de Bakker und J. van Leeuwen, Hrsg., *Automata, Languages and Programming*, Seite 632. Springer, 1980. Technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.

[VSB+01] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood und I. L. Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414:883, 2001.

[WD09] R. Wille und R. Drechsler. BDD-based Synthesis of Reversible Logic for Large Functions. In *Design Automation Conf.*, Seiten 270–275, 2009.

[WG07] R. Wille und D. Große. Fast Exact Toffoli Network Synthesis of Reversible Logic. In *Int'l Conf. on CAD*, Seiten 60–64, 2007.

[WGDD09] R. Wille, D. Große, G.W. Dueck und R. Drechsler. Reversible Logic Synthesis with Output Permutation. In *VLSI Design*, Seiten 189–194, 2009.

[WGF+09] R. Wille, D. Große, S. Frehse, G. W. Dueck und R. Drechsler. Debugging of Toffoli Networks. In *Design, Automation and Test in Europe*, Seiten 1284–1289, 2009.

[WGMD09] R. Wille, D. Große, D. M. Miller und R. Drechsler. Equivalence Checking of Reversible Circuits. In *Int'l Symp. on Multi-Valued Logic*, Seiten 324–330, 2009.

[WLDG08] R. Wille, H. M. Le, G. W. Dueck und D. Große. Quantified Synthesis of Reversible Logic. In *Design, Automation and Test in Europe*, Seiten 1015–1020, 2008.

[WSD09] R. Wille, M. Saeedi und R. Drechsler. Synthesis of Reversible Functions Beyond Gate Count and Quantum Cost. In *Int'l Workshop on Logic Synth.*, Seiten 43–49, 2009.

Robert Wille studierte Informatik an der Universität Bremen, wo er 2006 sein Diplom erhielt. Seit November 2006 ist er dort als wissenschaftlicher Mitarbeiter der Arbeitsgruppe Rechnerarchitektur in Forschung und Lehre tätig. Seine Forschungsinteressen konzentrieren sich dabei auf die Entwicklung eines Schaltkreisentwurfes für reversible Logik. Dies war auch Gegenstand seiner Promotion, welche er 2009 mit dem Prädikat "summa cum laude" beendete. Für seine Arbeit über die exakte Synthese wurde er außerdem 2008 mit dem „Young Researchers Award" des *IEEE International Symposium on Multiple-Valued Logic* ausgezeichnet. Darüber hinaus beschäftigt sich Herr Wille mit der Entwicklung von Algorithmen zur Lösung Boolescher Erfüllbarkeit sowie der formale Verifikation von Schaltkreisen und Systemen.