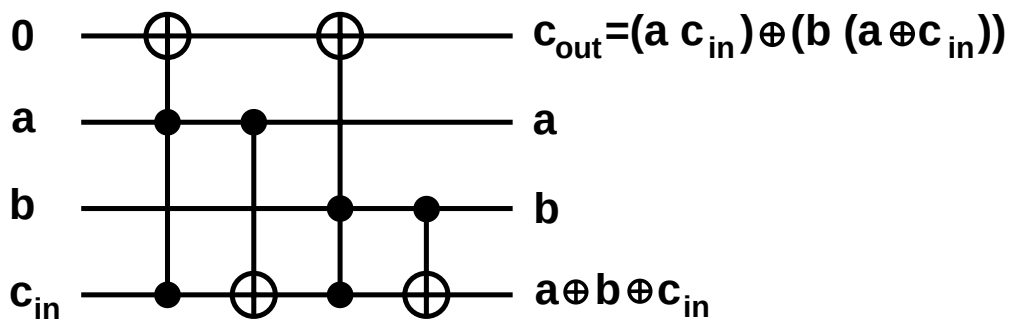
Quanten-Assembler

Erik Koch

Institut für Festkörperforschung, FZ-Jülich



Überblick

- Hardware: Logische Gatter
- Boolesche Logik: elementare Operationen
- Reversible Logik: Toffoli Gatter (3-Bit Operationen!)
- Verallgemeinerung auf Quanten Logik
- elementare Operationen: 1- und 2-Qbit Gatter

Logische Gatter

Gatter stellen elementare Operationen zur Berechnung logischer Funktionen zur Verfügung (“Maschinensprache”)

Gatter müssen physikalisch realisiert werden

— Der Rest ist Software . . .

Boolesche Logik

logische Funktionen: $f : \{0,1\}^n \mapsto \{0,1\}^m$
o.B.d.A. $m = 1$ (betrachte m verschiedene Funktionen)

2^n verschiedene Wertekombinationen in $\{0,1\}^n$:

"Wahrheits-Tafeln"

Beispiele

		NOT		
x	$f(x) = 0$	$f(x) = x$	$f(x) = \bar{x}$	$f(x) = 1$
0	0	0	1	1
1	0	1	0	1

		AND	OR	XOR	NAND
x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$	$x_1 \oplus x_2$	$\overline{x_1 \cdot x_2}$
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	1	0	0

Realisierung beliebiger logischer Funktionen

Beispiel: full-adder: binäre Addition mit Übertrag

a	*b*	*c*	*sum*	*carry*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Normalformen

Disjunktive Normalform:
Für jede Zeile der Wertetabelle, die 1 liefert,
verknüpfe input (ggf. entsprechend negiert) mit AND
$\Rightarrow$ Term ist für genau diese Zeile gleich 1, sonst 0

Verknüpfe Terme für verschiedene Zeilen mit OR

Beispiel: $sum = (\bar{a}\bar{b}c) + (\bar{a}b\bar{c}) + (a\bar{b}\bar{c}) + (abc)$

Konjunktive Normalform:
Für jede Zeile der Wertetabelle, die 0 liefert,
verknüpfe input (ggf. entsprechend negiert) mit OR
$\Rightarrow$ Term ist für genau diese Zeile gleich 0, sonst 1

Verknüpfe Terme für verschiedene Zeilen mit AND

Normalformen

Jede logische Funktion läßt sich mittels
NOT, AND und OR ausdrücken.
Außerdem müssen input-Variablen kopiert(!) werden
(no cloning!?)

COPY, NOT, AND und OR bilden eine **Basis**,
um logische Funktionen auszudrücken.
Insbes.: 1- und 2-Bit Gatter reichen aus,
um beliebige logische Funktionen auszudrücken.

Reversible Funktionen

reversible logische Funktionen: $f : \{0,1\}^n \mapsto \{0,1\}^n$,
wobei Funktionswerte Permutationen der input Wertetabelle sind.
Also 2^n verschiedene Funktionen

Beispiel: cNOT

x_1	x_2	y_1	y_2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Realisierung nicht-reversibler Funktionen mit cNOT

Beispiele:

$COPY:$	$cNOT(x,0) = (x,x)$	benötigt Steuerbit$(x_2 = 0)$
$XOR:$	$cNOT(x_1,x_2) = (x_1, x_1 \oplus x_2)$	Produziert Abfall$(y_1 = x_1)$
$NOT:$	$cNOT(1,x) = (1,\bar{x})$	Recycling:$x_1 = 1, y_1 = 1)$

cNOT:

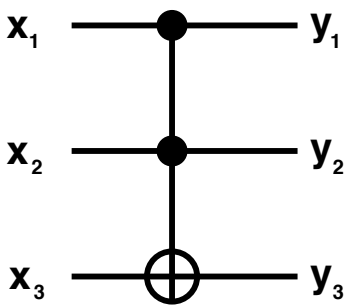
x_1	x_2	y_1	y_2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

AND in reversibler Logik

x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

AND benötigt *zwei* zusätzliche output Bits, um reversibel zu werden

Toffoli-Gatter: $y_1 = x_1$, $y_2 = x_2$ und $y_3 = (x_1 \cdot x_2) \oplus x_3$



$$AND: \quad ccNOT(x_1, x_2, 0) = (x_1, x_2, x_1 \cdot x_2)$$
$$cNOT: \quad ccNOT(1, x_2, x_3) = (1, x_2, x_2 \oplus x_3)$$

Basis

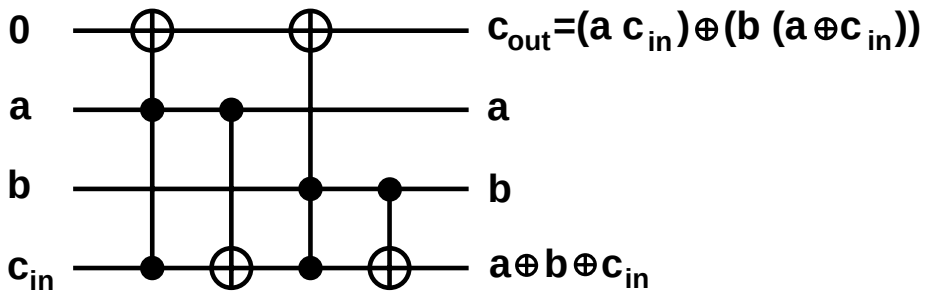
Alle Boolschen Funktionen lassen sich also mittels reversibler Logik realisieren, indem man die nicht-reversiblen Gatter durch reversible ersetzt und die dabei benötigten Steuer- bzw. Abfall-Bits zur Verfügung stellt.

Basis:
Da sich COPY, NOT und AND durch Toffoli-Gatter realisieren lassen, bildet das Toffoli-Gatter einen Basis für die reversible Logik

Da sich AND reversibel nur mittels mindestens 3-Bit Operationen realisieren lässt, muss reversible Basis mindestend ein 3-Bit Gatter enthalten . . . (Quanten-Gatter!?)

Volladdierer in reversibler Logik

a	*b*	*c*	*sum*	*carry*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Erweiterung auf Qbits

Reversible Gatter definieren Operationen auf Basiszuständen der Qbits.

Natürliche Erweiterung auf Quanten-Operationen mittels Linearität:

Abbildung zwischen Basiszuständen: $f : \{0,1\}^n \mapsto \{0,1\}^n$

linearer Operator durch Operationen auf Basis eindeutig bestimmt:
$U_f \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle = \sum \alpha_x U_f |x\rangle = \sum \alpha_x |f(x)\rangle$

reversible Gatter liefern unitäre Operationen: **Quanten-Gatter**

cNOT als Quanten-Kopierer?

cNOT

x_1	x_2	y_1	y_2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

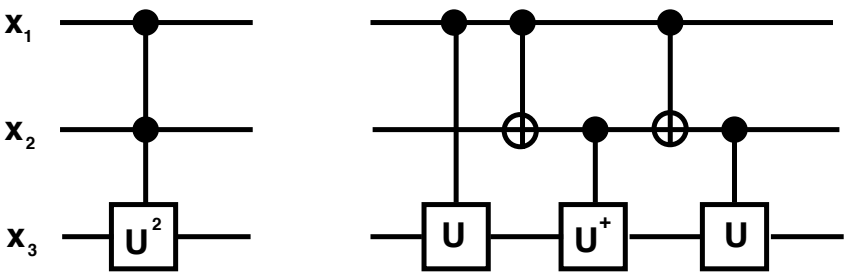
$$U_{cNOT}|\Psi\rangle|0\rangle = \alpha U_{cNOT}|0\rangle|0\rangle + \beta U_{cNOT}|1\rangle|0\rangle = \alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle$$

U_{cNOT} kopiert nur Basiszustände,
ansonsten werden *verschränkte* Zustände produziert.

Sleator-Weinfurter Konstruktion

reversible Logik benötigt 3-Bit Gatter (z.B. Toffoli)

Quanten-Gatter: 3-Bit ccU^2-Gatter mittels 2-Qbit Gatter:



x_1	x_2	x_2'	x_2''	Trafo auf x_3
0	0	0	0	$I\ I\ I = I$
0	1	1	1	$UU^\dagger I = I$
1	0	1	0	$IU^\dagger U = I$
1	1	0	1	$U\ I\ U = U^2$

Toffoli-Gatter mittels 2-Qbit Gattern realisierbar,
wenn wir $U_{\sqrt{NOT}}$ finden; d.h. $U^2_{\sqrt{NOT}} = U_{NOT}$

$\sqrt{NOT}$

$$U_{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Idee:
U_{NOT} diagonalisieren,
Eigenwerte durch Wurzeln ersetzen,
und rücktransformieren

$$U_{\sqrt{NOT}} = \frac{1}{1-i} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$$

Toffoli-Gatter durch 2-Qbit Gatter realisierbar!

Quanten-Computer kommt mit 1- und 2-Qbit Gattern aus

Nicht-klassische Quanten-Gatter

Quanten-Gatter ohne reversibles Äquivalent:
$U_{\sqrt{NOT}}$

Hadamard-Gatter:

$$U_H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

erzeugt Linearkombinationen:

$$U_H|0\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$$
$$U_H|1\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

Zusammenfassung

- Hardware: Logische Gatter
- Boolesche Logik: Disjunktive-/Konjunktive Normalform
- Reversible Logik: Toffoli Gatter (3-Bit Operation!)
- Verallgemeinerung auf Quanten Logik
- elementare Operationen: 1- und 2-Qubit Gatter
  Sleator-Weinfurter Konstruktion