1.6 Schaltkreise

Zur Vorlesung Rechenanlagen

SS 2019

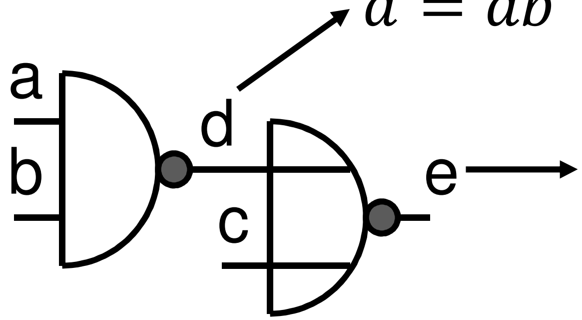
Schaltkreise

Wir können statische Aspekte des Verhaltens von digitalen Schaltkreisen mit dem nun erarbeiteten Wissen durch boolesche Gleichungen beschreiben:

- Assoziiere zu jedem Signal der Schaltung einen Namen
- Assoziiere statische Beziehungen durch Gleichungen

Beispiel:

$d = \overline{ab}$



$e = \overline{d \vee c}$

Durch Substitution von Namen durch Gleichungen erhält man nun statische Beziehungen zu anderen Leitungen:

$$e = \overline{d \vee c}\big(d = \overline{ab}\big) = \overline{\overline{ab} \vee c} = ab\bar{c}$$

Definitionen zu Schaltkreisen

Ein Schaltkreis *C* über einem Bausteinsystem *A* besteht aus einer Menge

- *G* von **Bausteininstanzen** (Gates)
- $P(C):=\{C.a_1,...,C.a_n\}$ von **äusseren Anschlüssen** (Ports)
- *N* von **Netzen**

Zu jeder Bausteininstanz *g* aus *G* sei ferner eine Menge von **Anschlüssen** (Konnektoren, Ports)

$$P(g) \coloneqq \{g.a_1, \ldots, g.a_k\}$$

und ein Baustein *comp(g)* in *A* gegeben den *g* instanziiert.

Gilt *comp(g) = comp(h)* für zwei Instanzen *g,h,* dann müssen Namen und Anzahl der Anschlüsse jeweils gleich sein: d.h. ist *g.a* in *P(g),* dann ist auch *h.a* in *P(h)*

Definitionen zu Schaltkreisen ff

Die Anschlussmengen der Bausteine bilden paarweise disjunkte Mengen, d.h. $g \neq h \Longrightarrow P(g) \cap P(h) = \{\}$

Ein Netz *s* aus der Netzmenge *N* identifizieren wir zugleich mit einer Teilmenge von Anschlüssen, die Anschlüsse die es verbindet: d.h. es ist stets $s \subseteq P(C) \cup \bigcup_{g \in G} P(g)$

Dabei liegt **jeder** Anschluss *g.x* **auf genau einem** Netz, dem Netz zum Anschluss. Wir nennen dies *net(g.x).*

Netze bilden physikalisch die Verbindungen zwischen Bausteininstanzen und dienen zur Kommunikation. Wir benutzen daher statt des Begriffs Netz auch den Begriff Signal, wenn wir Schaltkreisen ein Verhalten zuordnen.

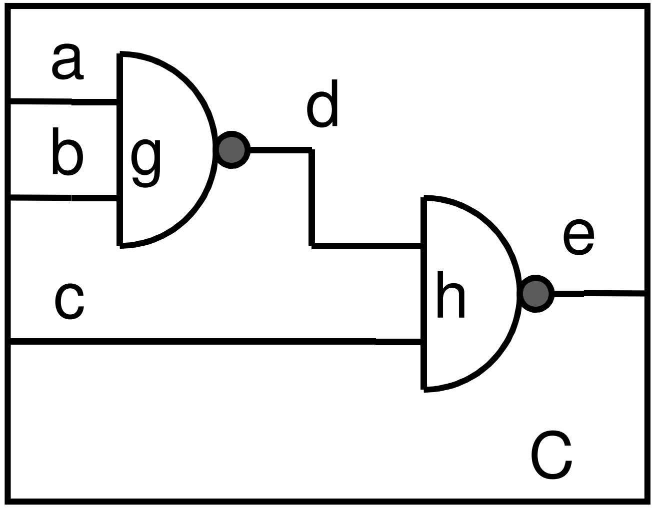
Schaltkreise in WüHDL

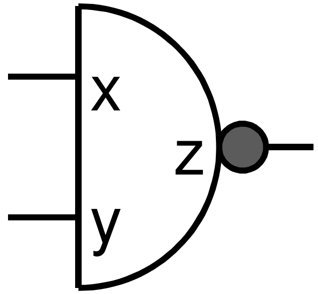
In WüHDL kann man Schaltkreise durch Instanziierung von Komponenten und das Binden ihrer Ports an Signale oder Ein-/ Ausgangsports beschreiben. Das Konzept ist etwas allgemeiner, weil man mehr Komfort bei der Beschreibung von Schaltkreisen haben möchte:

1. Das Bausteinsystem ist nicht fest, sondern wird in der COMPONENT Deklaration festgelegt. Komponenten können dabei selbst wieder durch Schaltkreise modelliert sein. Dadurch erhält man eine kurze und übersichtliche Definition eines im Grunde gigantischen finalen Schaltkreises. Man nennt diese Technik auch **hierarchischen Entwurf**.
2. Prozesse und concurrent signal assignments können im Grunde auch als Bausteininstanzen aufgefasst werden, mit dem Unterschied, dass es keine Mehrfachbenutzung gibt und das Verhalten des Bausteins direkt durch Programmcode definiert wird.

Wir beschränken uns bei der Theorie auf ein möglichst einfaches Schaltkreismodell, weil wir uns zunächst für spezielle Gattungen von Schaltkreisen interessieren.

Beispiel





Wir haben einen Schaltkreis *C* bestehend aus

- Gatter *G={g,h}, comp(g)=Nand2=comp(h)*
- Netze *N={a,b,c,d,e}*
- Ports *P(C)={C.a,C.b,C.c,C.e}*

Es ist nun etwa *P(g)={g.x,g.y,g.z}*, *P(h)={h.x,h.y,h.z}*
sowie z.B. *d={g.z,h.x}*, *e={h.z,C.e}* oder *net(g.y)=b*

Definitionen zu Schaltkreisen ff

Meist assoziiert man mit den Anschlüssen einen Wertefluss, d.h. man unterscheidet zwischen **Eingängen** und **Ausgängen.** Bausteinsysteme mit dieser Differenzierung nennt man **orientiert.**

Wir nennen also ein Bausteinsystem *A* **orientiert**, wenn zu jeder Bausteininstanz *b* die Anschlussmenge eine disjunkte Vereinigung von *In(b)* und *Out(b)* bildet.

Damit ist $\mathrm{P(g)} = \mathrm{In(g)} \cup \mathrm{Out(g)}; \mathrm{In(g)} \cap \mathrm{Out(g)} = \{\}$

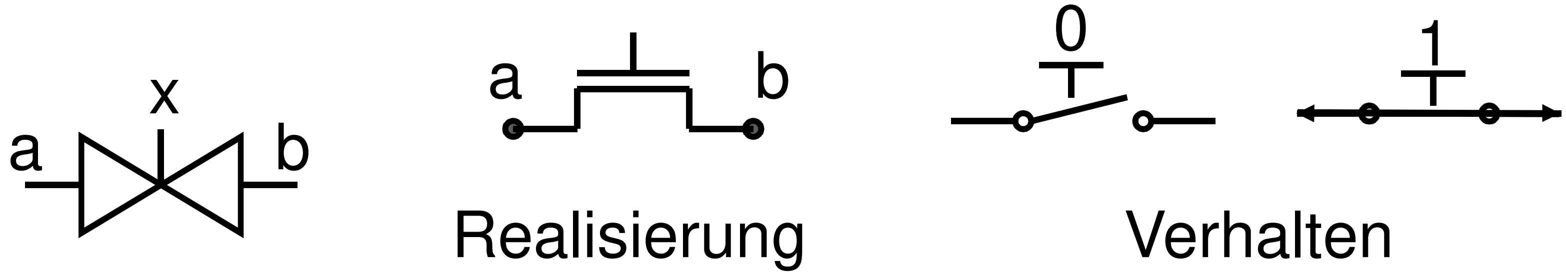
und es ist $\mathrm{In(g)} \neq \{\} \neq \mathrm{Out(g)}$

Ein Schaltkreis *C* heißt **orientiert**, wenn er über einem orientierten Bausteinsystem aufgebaut ist, und seine Ports in Eingänge *In(C)* und Ausgänge *Out(C)* klassifiziert sind, d.h. $In(C) \cup Out(C) = P(C)$ und $In(C) \cap Out(C) = \{\ \}$

Definitionen zu Schaltkreisen ff

Bemerkung:

Es gibt durchaus nützliche Bausteine, die nicht orientierbar sind, wie etwa ein Schalter:



Orientierte Schaltkreise bilden also eine Unterklasse. Allerdings sollte man weitere Bedingungen an ihre Struktur stellen, damit man ihnen ein Verhalten zuordnen kann:

Wir nennen einen orientierten Schaltkreis **wohlgeformt**, genau dann, wenn er folgende beiden Bedingungen (OC1) und (OC2) erfüllt:

Definitionen zu Schaltkreisen ff

(OC1) $\forall s \in N: \#\left(s \cap \left(In(C) \cup \bigcup_{g \in G} Out(g)\right)\right) \leq 1$

(OC1) besagt, dass höchstens ein Ausgang einer Bausteininstanz zu einem Netz *s* gehört. Da dieser Ausgang physikalisch gesehen den Pegel des Signals zu diesem Netz bestimmt, nennen wir diesen Ausgang, falls vorhanden, den **Treiber** von *s*, *treiber(s)*. Hat ein Baustein *g* nur einen Ausgang, *g.z*, so lassen wird dessen Angabe auch weg und nennen einfach *g* den Treiber von *s*.

Definitionen zu Schaltkreisen ff

(OC2) $\forall s: s \cap \bigcup_{g \in G} Out(g) = \{\} \Longrightarrow In(C) \cap s \neq \{\}$

(OC2) besagt, dass alle Netze, die nicht von Ausgangsports getrieben werden, von außen zugänglich sein müssen (zu den Eingängen des Schaltkreises gehören). Wir nennen ein Netz *s*, das einen Eingang *C.x* des Schaltkreises enthält, dann auch den Primäreingang zu *C.x*.

Wir nehmen von nun an stets an, dass unsere orientierten Schaltkreise auch wohlgeformt sind.

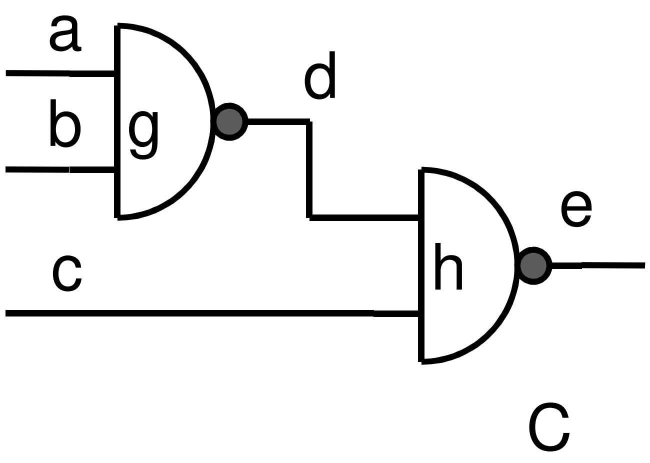
In wohlgeformten orientierten Schaltkreisen kann man nun noch weitere Begriffe betrachten:

Definitionen zu Schaltkreisen ff

Eine Folge von Netzen $s_1, \ldots s_k$ heißt **Pfad** der Länge k-1

$$\Leftrightarrow \forall 1 \leq i < k \; \exists g_i : In(g_i) \cap s_i \neq \{\} \neq Out(g_i) \cap s_{i+1}$$

Beispiel:



Pfade: *d,e*

a,d,e

b,d,e

c,e

Rückkopplungsfreie Schaltkreise

Ein orientierter Schaltkreis heißt **rückkopplungsfrei**

: $\Leftrightarrow$ Für alle Pfade $s_1 \ldots s_k$ gilt: $s_i = s_j \Rightarrow i=j$

Folgerung:

In einem rückkopplungsfreien Schaltkreis haben alle Pfade eine Länge kleiner gleich *#N.*

Beweis:

Betrachte einen Pfad $s_1 \ldots s_k$ mit $k > \#N$

Dann können nicht alle Netze verschieden sein!

(Schubfachprinzip!)

Also gibt es ein $i < j$ mit $s_i = s_j$ $\lightning$

Rückkopplungsfreie Schaltkreise ff

Rückkopplungsfreie Schaltkreise spielen eine wichtige Rolle, weil man mit ihnen alle Schaltfunktionen realisieren kann.

Beobachtung:

Für einen rückkopplungsfreien Schaltkreis ist die Funktion

$$tiefe(s) := \begin{cases} 0 & falls\ s \cap In(C) \neq \{\} \\ 1 + \max\{tiefe(r) \mid r \cap In(gate(treiber(s))) \neq \{\}\} & sonst \end{cases}$$

wohldefiniert.

Beweis *(Tiefe)*

Sei *s* ein beliebiges Netz. Wir zeigen die Beobachtung durch Induktion nach der Länge *k* eines längsten Pfades

$$s_1, \ldots, s_{k+1} = s$$

(Da alle Pfade Länge kleiner gleich #*N* haben, gibt es stets einen längsten Pfad nach *s*.)

k = 0: Dann gibt es kein Bauteil *g* mit *gate(treiber(s))=g*, sonst wäre ja *r,s* für *r* aus *sig(In(g))* ein Pfad der Länge 1. Also ist *s* ein Primäreingang und damit *tiefe(s)=0*

Beweis (*Tiefe*) ff

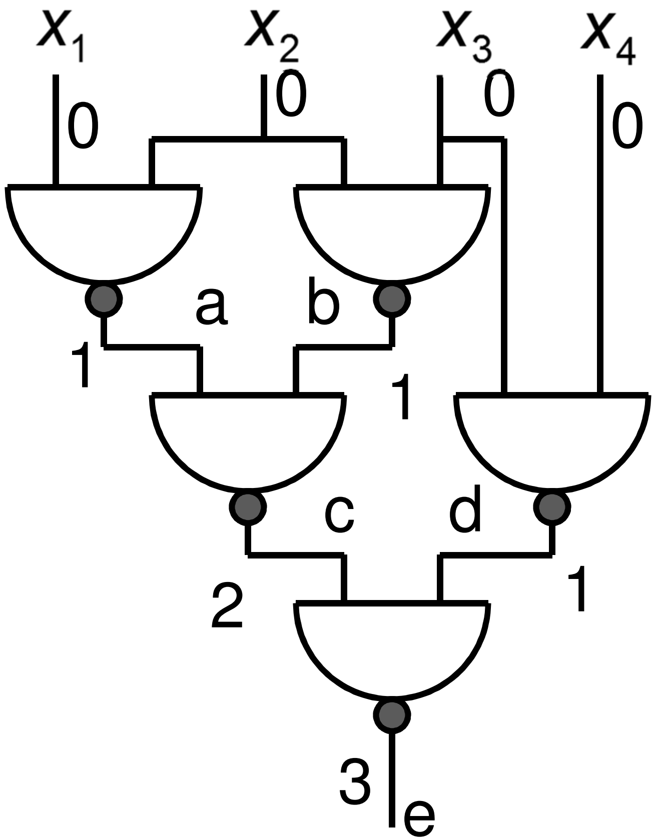
$k \to k+1$: Sei nun $s_1, \ldots, s_{k+1}, s_{k+2} = s$ längster Pfad nach s.
Dann gilt für alle r mit $r \cap In(gate(treiber(s))) \neq \{\ \}$,
dass jeder Pfad $s'_1, \ldots, s'_{l+1} = r$
Länge $l < k+1$ hat.
(sonst wäre ja $s'_1, \ldots, s'_{l+1}, s_{k+2} = s$ länger!)
Also ist $tiefe(s) = 1 + tiefe(s_{k+1})$
$= 1 + k$ -- Induktionsannahme
wohldefiniert.

Wir nennen

$$\max\{tiefe(s) \mid s \in S\} =: tiefe(C)$$

die **Tiefe eines Schaltkreises** *C*.

Beispiel: (Tiefe)



Netz	*Tiefe*
x_1	0
x_2	0
x_3	0
x_4	0
a	1
b	1
c	2
d	1
e	3

Wir wollen nun Schaltkreisen ein Verhalten zuordnen:

Statisches Verhalten

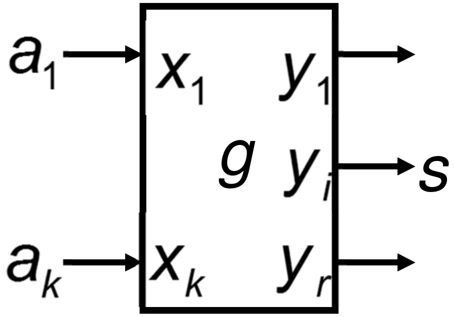
Wir nennen ein Bausteinsystem *A* **kombinatorisch,** wenn alle Bausteine orientierte Anschlüsse haben, und mit jedem Ausgang *y,* etwa durch einen booleschen Ausdruck, eine Funktion *y* über den Eingängen assoziiert ist.

Beispiel: wir assoziieren bei einem NAND2 mit Eingängen *a* und *b* und Ausgang *y* die Funktion

$$y = \overline{ab}$$

Sei *A* ein kombinatorisches Bausteinsystem, und sei *C* ein wohlgeformter, orientierter Schaltkreis über *A*. Dann können wir zu jedem Netz *s*, das von einem Gatter getrieben wird, eine Funktion über den Eingängen der Bausteininstanz *g* betrachten, dessen Ausgang *s* treibt:

Statisches Verhalten ff



Zum Bausteintyp g und Ausgang y_i sei $y_i = y_i(x_1,\ldots,x_k)$ die assoziierte Funktion.

Sei ferner für alle $i=1,\ldots,k$:

$$a_i = net(g.x_i) \text{ sowie } treiber(s) = g.y_i$$

Dann ordnen wir dem Netz s seine **lokale Funktion**

$$\boxed{C[s] = C[s](a_1, \ldots, a_k) := y_i(\ldots, x_i = net(g.x_i), \ldots)}$$

zu.

Mit einem Primäreingang u zu C.x assoziieren wir $\boxed{C[u]=x}$

Statisches Verhalten

Sei *C* ein Schaltkreis über einem kombinatorischen Bausteinsystem *A.* Wir können **Signalbelegungen** des Schaltkreises als *#N*-stellige Bitvektoren auffassen. Eine Signalbelegung nennen wir **stabil**, wenn die Belegung jedes Signals (=Netz) der Auswertung der lokalen Funktion des Signals auf den Belegungen ihrer Argumente entspricht.

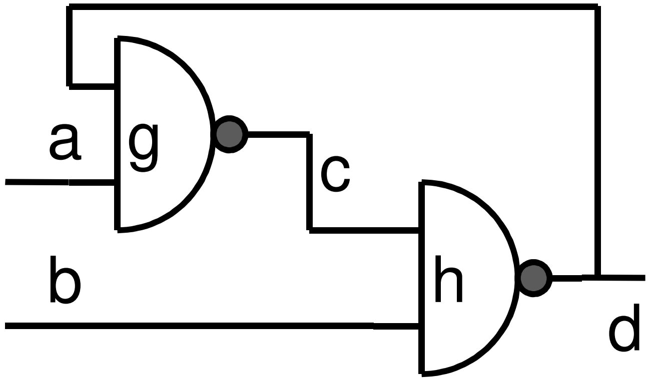
Demnach können wir jedem orientierten, wohlgeformten Schaltkreis *C* über kombinatorischen Bausteinen eine Funktion

$$\boxed{\chi_C \coloneqq \prod_{s \in N} (s \equiv C[s])}$$

zuordnen, die **Charakteristische Funktion** seines **statischen Verhaltens.**

Sie wird auf einer Signalbelegung genau dann 1, wenn der Schaltkreis mit dieser Belegung stabil ist.

Beispiel:

C = 

$$\chi_C = (c \equiv \overline{ad})(d \equiv \overline{bc})$$

(a,b,c,d) = (1,0,1,0):

Instabil, da

$$\chi_C(1,0,1,0) = (1 \equiv \overline{10})(0 \equiv \overline{01}) = (1 \equiv 1)(0 \equiv 1) = 1 \cdot 0 = 0$$

(a,b,c,d) = (0,1,1,0):

Stabil, da

$$\chi_C(0,1,1,0) = (1 \equiv \overline{00})(0 \equiv \overline{11}) = (1 \equiv 1)(0 \equiv 0) = 1 \cdot 1 = 1$$

(a,b,c,d) = (1,1,$\overline{y}$,y):

Stabil für jedes *y*, da

$$\chi_C(1,1,\overline{y},y) = (\overline{y} \equiv \overline{1y})(y \equiv \overline{1\overline{y}}) = (\overline{y} \equiv \overline{y})(y \equiv y) = 1 \cdot 1 = 1$$