Kapitel 5

Schaltnetze und Schaltwerke

Alle Daten, die im Rechner gespeichert sind und transportiert werden sind als binäres elektisches Signal codiert.

Typische Spannungswerte:

- 0 Volt (Masse, niedrig, 0)
- 5 Volt (hoch, 1)

Werte verändern sich in Sprüngen nach Zeittakten (Abbildung 5.1), der reale Verlauf ist anders (Abbildung 5.2). Abfragetakt ist versetzt, daher sichere Abfrage im Vergleich zum Rand des Schalttaktes.

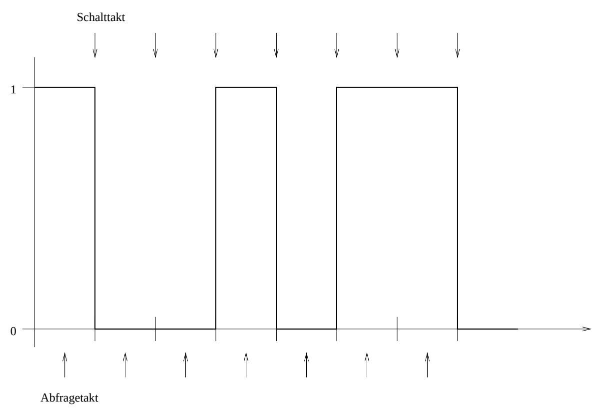


Abbildung 5.1: Digitales Signal, idealisiert

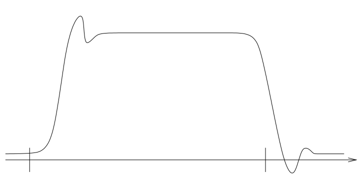
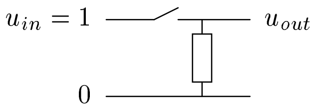


Abbildung 5.2: realer Verlauf

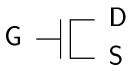
Realisierung durch Schalter.

$$u_{out} = \begin{cases} 0 & \text{falls geöffnet} \\ 1 & \text{falls geschlossen} \end{cases}$$



MOS-Transistoren. (metal-oxid-semiconductor)

G	=	Gate
S	=	Source
D	=	Drain



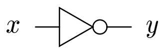
[Folgende Erklärung wörtlich—wie lautet das richtig?]

$$\text{Gate-Source-Spannung} = \begin{cases} 0 & \text{großer Widerstand S-D} \\ 1 & \text{schwacher S-D} \end{cases}$$

Gatter. Mit folgenden Gattern können beliebige Funktionen $f\colon \mathbb{B}^n \longrightarrow \mathbb{B}^m$ (n-stellige Schaltfunktionen mit m-stelligem Ergebnis) realisiert werden. (s.u.)

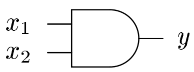
- Negation:

  $x = 0 \Longrightarrow y = 1$
  $x = 1 \Longrightarrow x = 1$

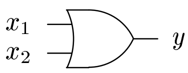
- Konjunktion:

  $x_1 = x_2 = 1 \Longrightarrow y = 1$
  sonst $\Longrightarrow x = 1$

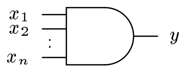
- Disjunktion:

  $x_1 = 1 \vee x_2 = 1 \Longrightarrow y = 1$
  sonst $\Longrightarrow x = 0$

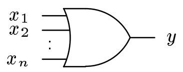
- verallgemeinerte Konjunktion:

  analoge Bedingung



- verallgemeinerte Disjunktion:

  analoge Bedingung



Die Schaltungen, mit denen die Gatter realisiert werden können, sind in Abbildungen 5.3, 5.4 und 5.5 dargestellt.

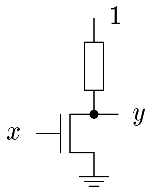


Abbildung 5.3: Realisierung der Negation

Zur Realisierung von beliebigen Funktionen $f\colon \mathbb{B}^n \longrightarrow \mathbb{B}^m$: Ist $p_j\colon \mathbb{B}^m \longrightarrow \mathbb{B}$ die Projektion auf die j-te Komponente, dann ist $f_j = p_j \circ f\colon \mathbb{B}^n \longrightarrow \mathbb{B}$ eine Boolesche Funktion, d.h. f_j hat KNF (DNF), läßt sich also durch Gatter realisieren.

f wird realisiert durch Nebeneinandersetzen der Realisierung der Funktion f_j. (Siehe Abbildung 5.6.)

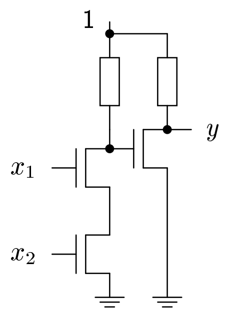


Abbildung 5.4: Realisierung der Konjunktion

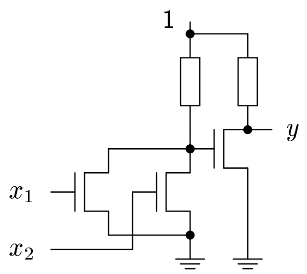


Abbildung 5.5: Realisierung der Disjunktion

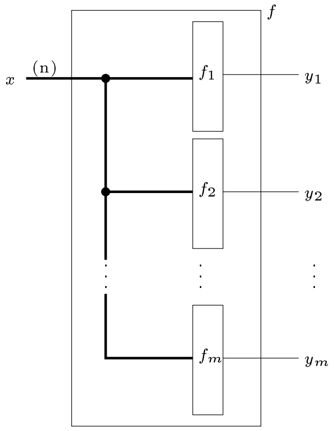


Abbildung 5.6: Realisierung einer Funktion $f\colon \mathbb{B}^n \longrightarrow \mathbb{B}^m$

Prinzipien zum Aufbau von komplexen Funktionen durch einfachere Bestandteile:

1. Tupeling.
   $f_1 : \mathbb{B}^n \longrightarrow \mathbb{B}^m$, $f_2 : \mathbb{B}^n \longrightarrow \mathbb{B}^k$:
   $(f_1, f_2) : \mathbb{B}^n \longrightarrow \mathbb{B}^{m+k}$
   Siehe Abbildung 5.7.

2. Parallele Komposition.
   $f_1 : \mathbb{B}^{n_1} \longrightarrow \mathbb{B}^{m_1}$, $f_2 : \mathbb{B}^{n_2} \longrightarrow \mathbb{B}^{m_2}$:
   $(f_1 \| f_2) : \mathbb{B}^{n_1+n_2} \longrightarrow \mathbb{B}^{m_1+m_2}$

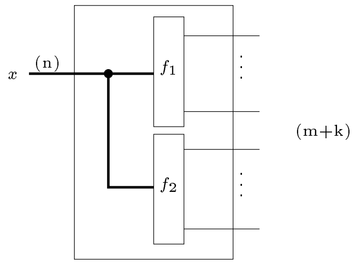


Abbildung 5.7: Tupeling

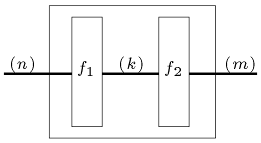


Abbildung 5.8: sequentielle Komposition

3. Sequentielle Komposition.
   $f_1 : \mathbb{B}^n \longrightarrow \mathbb{B}^k$, $f_2 : \mathbb{B}^k \longrightarrow \mathbb{B}^m$:
   Mit $f = f_2 \circ f_1$.
   Siehe Abbildung 5.8.

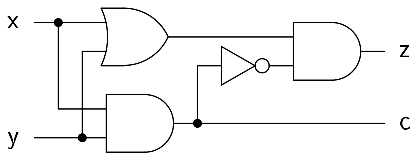
Definition 40. Ein Schaltnetz ist ein aus Gattern aufgebauter gerichteter, azyklischer Graph. (Auch: kombinatorischer Schaltkreis.)

(gerichtet: alle Kanten sind gerichtet von den Ausgängen weg und zu Eingängen zu; azyklisch: keine gerichteten Kreise)

(Ingrad: Anzahl der Eingänge in ein Gatter; festgelegt durch die Art des Gatters. Ausgrad: Anzahl der Ausgänge eines Gatters; Vervielfachung durch Tupeling möglich.)

Arithmetische Schaltnetze Halbaddierer: Addition von zwei Bits x, y; liefert Ergebnis $z = (x \vee y) \wedge \neg(x \wedge y)$ und Übertrag $c = x \wedge y$:

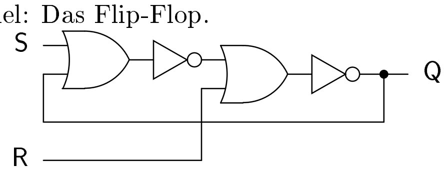
x	y	z	c
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	0



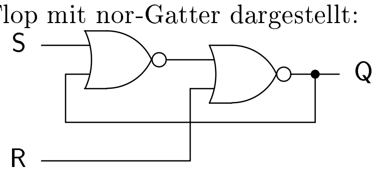
Schaltwerke

- Rückkopplung erlaubt
- Schaltwerk kann bestimmte Zustände annehmen

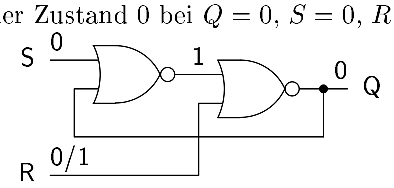
Beispiel: Das Flip-Flop.



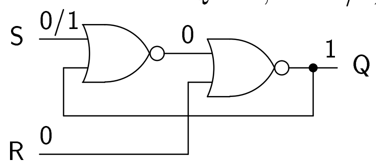
Flip-Flop mit nor-Gatter dargestellt:



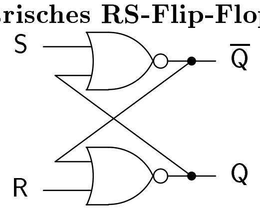
stabiler Zustand 0 bei $Q = 0$, $S = 0$, $R = 0/1$:



stabiler Zustand 1 bei $Q = 1$, $S = 0/1$, $R = 0$:



Symmetrisches RS-Flip-Flop.



Annahme: Flip-Flop ist in stabilem Zustand mit
$\overline{Q}_{alt} = \neg Q_{alt}$

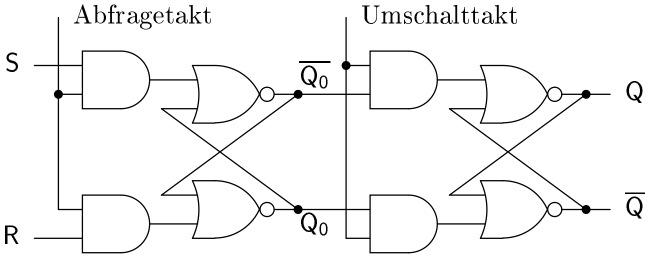
Tabelle zum Flip-Flop:

S	R	Q_{neu}	$\overline{Q}_{neu}$	
0	0	Q_{alt}	$\overline{Q}_{alt}$	
1	0	1	$0 = \overline{1}$	
0	1	0	$1 = \overline{0}$	
1	1	0	0	verboten

$R = S = 1$ verboten, weil

- $\overline{Q} \neq \neg Q$
- Schaltung gerät in instabilen Zustand (Schwingung), wenn $R = S = 0$ gesetzt wird.

Taktgesteuertes Master-Slave-RS-Flip-Flop.

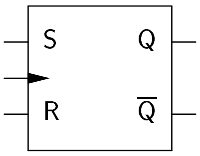


[Hier war noch ein Diagramm von Umschalttakt und Abfragetakt; es zeigt, daß die beiden entgegengesetzt getaktet sind. Zum Zeitpunkt t_0 sei die Kontrolleitung für das Abfragen gesetzt und jene zum Umschalten nicht, einen Takt später, zum Zeitpunkt t_1 gerade umgekehrt.]

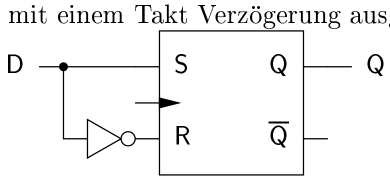
	S	R	Q_0	$\overline{\mathsf{Q}_0}$	Q	$\overline{\mathsf{Q}}$
t_0	1	0	1	0	Q_{alt}	$\overline{\mathsf{Q}}_{\text{alt}}$
t_1					1	0

S = 1 setzt das Flip-Flop also in den Zustand 1 (set). R = 1 setzt es in den Zustand 0 (reset).

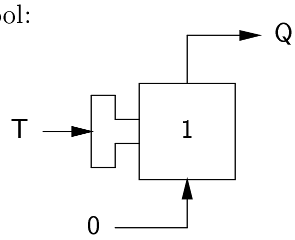
Symbol:



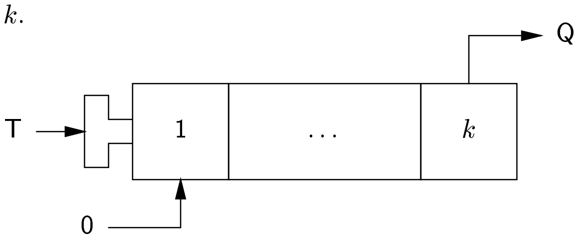
Flip-Flop als Verzögerungsglied. Signal D wird gespeichert und mit einem Takt Verzögerung ausgegeben:



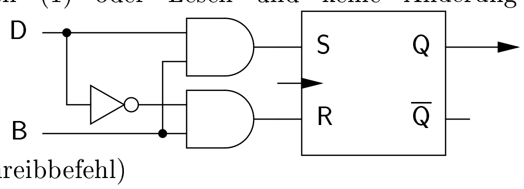
Symbol:



verallgemeinerbar: Verzögerung um k Takte; Schieberegister der Länge k.



Flip-Flop als Speicherzelle. Befehlseingang entscheidet, ob Schreiben (1) oder Lesen und keine Änderung



(0).

(D=Daten, B=Schreibbefehl)

Register (Menge logisch zusammenhängender Bits) durch parallele Anordnung von Speichzellen. [Unvollständig aussehendes Bild ausgelassen]

Schaltwerkfunktionen und endliche Automaten. Schaltnetze sind durch Boolesche Funktionen $f\colon \mathbb{B}^n \longrightarrow \mathbb{B}^m$ spezifiziert.

Erweiterung

$$f^*\colon (\mathbb{B}^n)^* \longrightarrow (\mathbb{B}^m)^*$$
$$f^*\; w_1 w_2 \ldots w_k = f(w_1) f(w_2) \ldots f(w_k)$$

d.h. f^* hat zum Zeitpunkt i die Eingabe $w_i \in \mathbb{B}^n$ und gibt $f(w_i)$ aus.

Schaltwerke sind nur durch $f^* : (\mathbb{B}^n)^* \longrightarrow (\mathbb{B}^m)^*$ spezifiziert, d.h. die Ausgabe zum Zeitpunkt i hängt auch von den Eingaben zu früheren Zeitpunkten ab, bzw. von Zustand des Schaltwerkes.

Von der Spezifikation zum Schaltwerk (Prinzip):

- Welche Zustände sind notwendig, d.h. welche Informationen müssen gespeichert werden. Codierung in $\mathbb{B}^k$.
- Beschreibung eines Taktes als endlicher Mealy-Automat.

Beispiel für ein Maschinenprogramm: Euklidischer Algorithmus. Voraussetzungen über Aufbau einer einfachen Realisierung einer Maschine:

- nur ein Register: AC (Akkumulator)
- Wortlänge von Maschinenbefehlen: 32 Bit (4 Byte)
- Hauptspeicher ist Byte-weise addressiert
- Befehlszähler erhöht seinen Inhalt jeweils um 4

Für das Beispiel siehe Abbildung 5.9.

LOAD-Befehl nutzt direkte Adressierung. Alternative: LOAD*-Befehl mit indirekter Adressierung: LOAD* xx lädt Speicherzelle, deren Adresse in xx steht.

von Neumann-Prinzipien

1. Rechner aus vier Werken

Adresse	Befehl	Kommentar
00		Speicherplatz von p
04		Speicherplatz von q
08		Speicherplatz von r
		Schritt 1
12	LOAD 00	Lade AC mit p
16	MODULo 04	Berechne den Rest von AC/q in AC
20	STORE 08	Speichere AC in r
		Schritt 2
24	IFZERO 48	Wenn AC=0, dann gehe nach 48
28	LOAD 04	Lade AC mit q
32	STORE 00	Speichere AC in p
36	LOAD 08	Lade AC mit r
40	STORE 04	Speichere AC in q
44	JUMP 12	Springe nach 12
48	END	Halte an, q enthält den ggT

Abbildung 5.9: Beispiel für Maschinenprogramm

2. Struktur des Rechners unabhängig von bearbeitetem Program (Programmsteuerung)
3. Programme und Daten in demselben Speicher–können durh Maschine verändert werden
4. Hauptspeicher in Zellen gleicher Größe geteilt, die fortlaufend numeriert sind (Adressen)
5. Programm: Folge von Befehlen, die im allgemeinen nacheinander ausgeführt werden (Sequentialität)

6. Bedingte und unbedingte Sprungbefehle setzen Sequentialitätsprinzip außer Kraft
7. Befehle und Daten werden binär dargestellt.

Abweichungen von diesen Prinzipien:

- Vervielfachung einzelner Komponenten zur Leistungssteigerung
- Vielstufige Speicherhierarchie
- in unterster ebene getrennte Speicher für Programme und Daten

Architekturen für den Prozessor.

RISC (Reduced Instruction Set Computer)

- elementare, kleine Maschinenbefehlssätze
- Operandenhol- und Ausführungsphase in je einem Grundtakt
- Adreßrechnung durch explizite Befehle (nicht Teil der Operandenholphase)
- einheitliche Befehlsformate (schnelle Decodierung)

Pipelining möglich: Parallele Verarbeitung mehrerer Befehle in den unterschiedlichen Ausführungsstufen:

```
            +----+----+----+----+----+----+
Befehl 1    | BH | DE | OP | AU | RS | AD |
            +----+----+----+----+----+----+----+
Befehl 2         | BH | DE | OP | AU | RS | AD |
                 +----+----+----+----+----+----+----+
Befehl 3              | BH | DE | OP | AU | RS | AD |
                      +----+----+----+----+----+----+----+
Befehl 4                   | BH | DE | OP | AU | RS | AD |
                           +----+----+----+----+----+----+
```

Complex Instruction Set Computer

- umfangreicher Maschinenbefehlssatz mit einigen hundert Befehlen, die zum Teil komplexe Aufgaben lösen (Komplexbefehle)
- viele Adressierungsarten und Adreßrechung in den Maschinenbefehlen
- uneinheitliches Befehlsformat
- Nutzung der Mikroprogrammierung als Implementierungstechnik
- Trennung von Befehlssatz, Implementierung und Realsierung
- Aufwärtskompatible Prozessorfamilien

Mikroprogrammierung:

- Komplexbefehel wird durch Folge einfacher Mikrobefehle realisiert
- Mikrobefehle sind in speziellen Speichern mit sehr kurzen Zugriffszeiten
- Maschinenbefehlszyklus wird in Folge von Mikrobefehlszyklen aufgeteilt

In neueren CISC-Prozessoren werden die komplexen Instruktionen in Mikrobefehle übersetzt und diese dann nach dem RISC-Prinzip verarbeitet, und Pipelining ist möglich.

Speicheraufbau und Organisation Verschiedene Speichermedien:

	Kapazität	Zugriffszeit
Magnetband	groß	lang
Optische Platte (CD)		
Magnetischplatten (HD)		
Halbleiterspeicher:	↑	↑
- DRAM (dynamischer RAM)		
- SRAM (statischer RAM)	klein	kurz

Speicherhierarchie für Zugriff:

- Prozessor mit Registern
- Cache (SRAM)
- Hauptspeicher (DRAM)

- Hintergrundspeicher (HD)
- Archivspeicher (CD, Magnetband)

SRAM (static randem access memory). Ein Flip-Flop für jedes gespeicherte Bit (Zugriffszeit: ein Prozessortakt (oder wenige)). Sechs Transistoren pro Bit.

DRAM (dynamic random access memory). Ein Transistor und einen Kondensator pro Bit; Leckströme, daher Zerstörung der Speicherinhalte; periodisches Auffrischen notwendig (refresh etwa alle 8 Nanosekunden)

Haupbestandteil des DRAMs ist die Speichermatrix (2048x2048 Bit), adressiert durch Zeilenaddresse und Spaltenaddresse. Lesen einer Zelle:

1. Auslesen der Zeile (zerstört die Zeile in der Matrix)
2. Auslesen der Bits
3. Rückschreiben der Zeile

Zugriffszeit: ca. 20-fache eines Prozessortaktes

Inhaltsverzeichnis