



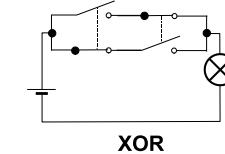
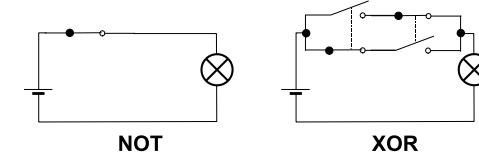
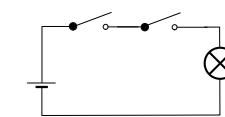
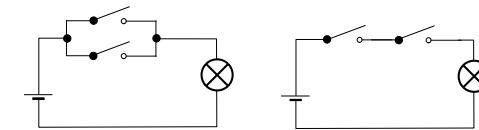
## 2. Kapitel

# Von der Schaltungslogik zur Informationsverarbeitung

Prof. Matthias Werner

Professur Betriebssysteme

## Schaltungslogik

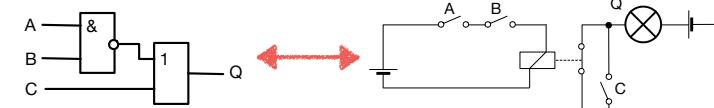


## Schaltnetze

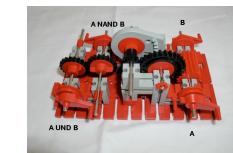
- ▶ Gatter implementieren boolesche Funktionen
- ▶ Signale werden durch elektr. Spannungen dargestellt
- ▶ Zusammensetzung von Gattern → Logikschatungen
- ▶ Schaltnetz ist eine Zusammensetzung von Verknüpfungsschaltungen **ohne Speicherverhalten**

	amerikanisch	alte DIN	DIN / IEC
AND			
OR			
NAND			
NOR			
XOR			
NOT			

## Schaltnetze (Forts.)



- ▶ Jedes Schaltnetzwerk kann auf eine elektrische Schaltung zurückgeführt werden
- ▶ Beliebige andere Technologien sind für die Implementierung möglich
- ▶ Ab jetzt ignorieren wir die Implementierung und konzentrieren uns auf die Schaltfunktion



Mechanisches AND/NAND mit  
Fischertechnik-Bausteinen  
*Bildquelle: Thomas Püttmann*

## Schaltnetze (Forts.)

- ▶ Durch logisch vollständige Funktionen können beliebige andere Funktionen „gebaut“ werden
- ▶ Die xor-Funktion ( $f(x, y) = x \oplus y$ ) soll mit Nicht-Und-Gattern realisiert werden

$$\begin{aligned}
 x \oplus y &= (x \wedge \bar{y}) \vee (\bar{x} \wedge y) \\
 &= (x \wedge \bar{y}) \vee (\bar{x} \wedge y) \vee (x \wedge \bar{x}) \vee (y \wedge \bar{y}) \\
 &= (x \wedge (\bar{x} \vee \bar{y})) \vee (y \wedge (\bar{x} \vee \bar{y})) \\
 &= (x \wedge (x \wedge y)) \vee (y \wedge (\bar{x} \wedge \bar{y})) \\
 &= (\overline{(x \wedge (x \wedge y))} \wedge \overline{(y \wedge (\bar{x} \wedge \bar{y}))})
 \end{aligned}$$

## Optimierung

- ▶ Eine Schaltung ist nicht immer optimal → zuviel Gatter
- ▶ Methoden zu Optimierung:
  - ▶ Verfahren nach McCluskey
  - ▶ Verfahren nach Karnaugh und Veitch
- ▶ Betrachten letzteres → **Karnaugh-Diagramm/KV-Diagramm**
- ▶ Anordnen der Funktionswerte in einer Matrix, so dass benachbarte Felder sich um einen Funktionsparameter unterscheiden
- ▶ Matrix ist als Oberfläche eines **Torus** aufzufassen
  - ▶ Ränder (oben/unten sowie rechts/links) sind verbunden
- ▶ Werte in Matrix eintragen, unbekannte (don't care) werden mit  $\phi$  bezeichnet
- ▶ Blöcke gleichen Werts zu Rechtecken mit  $2^n$  Kantenlänge zusammenfassen
- ▶ Block entspricht Ausdruck, der von weniger Variablen abhängig ist

## KV-Diagramm: Beispiel

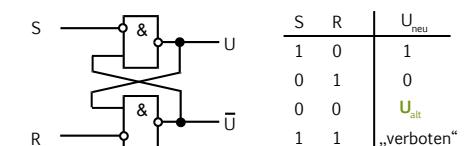
$$y = f(x_1, x_2, x_3) = x_1 \bar{x}_2 \bar{x}_3 \vee x_1 x_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 x_3$$

		$\bar{x}_1 \bar{x}_2$	$\bar{x}_1 x_2$	$x_1 \bar{x}_2$	$x_1 x_2$
$\bar{x}_3$	0	0	1	1	
$x_3$	1	0	1	1	

$y = x_1 \vee (\bar{x}_2 \wedge x_3)$

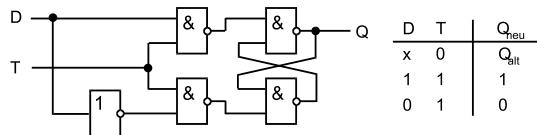
## Schaltwerke

- ▶ Ein Schaltwerk zeigt **Speicherverhalten**
  - es besitzt einen Zustand
- ▶ Durch Eingangssignale ändert sich der Zustand der Schaltung
- ▶ Einfachstes Speicherelement speichert die Zustände 0/1:
  - Flip-Flop

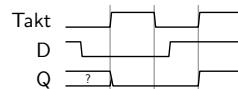


## Binäre Signale/Taktsteuerung

- Abstraktion von Signalübergängen: Takt
- Signale können dann als Rechtecksignale betrachtet werden
- synchrone Schaltungen: Hier D-Flip-Flop (auch Latch)
- auf die fallende Flanke an T wird der an D anliegende Datenwert (0 oder 1) „eingefroren“.



- Beispiel für Zeitverhalten:



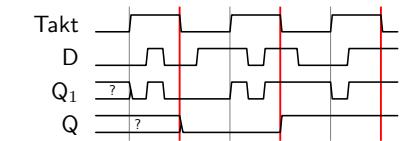
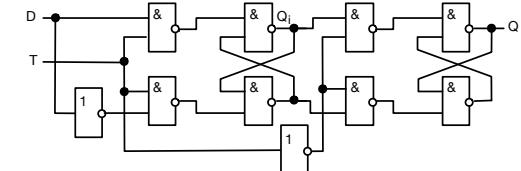
Wintersemester • Prof. Matthias Werner

55

<https://osg.informatik.tu-chemnitz.de>

## Beispiel: Master-Slave-Flip-Flop

- Mit der steigenden Flanke des Taktes wird der gesetzte Wert in die erste Stufe übernommen
- Derweil liegt am Ausgang noch der alte Wert an
- Mit der fallenden Flanke des wird die Eingangsstufe „versiegelt“ und der gesetzte Wert in die zweite Stufe (und den Ausgang) geschrieben



Wintersemester • Prof. Matthias Werner

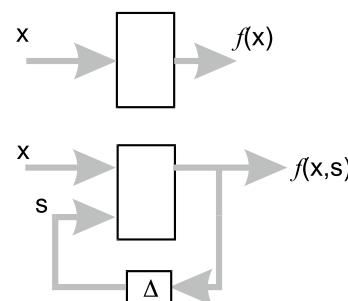
56

<https://osg.informatik.tu-chemnitz.de>

## Zustandbehaftete vs. zustandslose Schaltungen

- In zustandslosen Schaltungen hängt das Ausgangssignal ausschließlich vom Eingangssignal ab.
- Dagegen geht in das Ausgangssignal von zustands-behafteten Schaltungen die **Vorgeschichte** ein, die sich im **Zustand** manifestiert.

► Automatenmodell



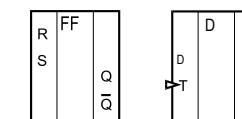
Wintersemester • Prof. Matthias Werner

57

<https://osg.informatik.tu-chemnitz.de>

## Blockschaltbilder

- Häufig stört eine zu detaillierte Darstellung das Verständnis der Funktion ➔ Blockschaltbilder



- Obwohl auch Blockschaltbilder genormt sind, werden sie sehr unterschiedlich in der Literatur dargestellt

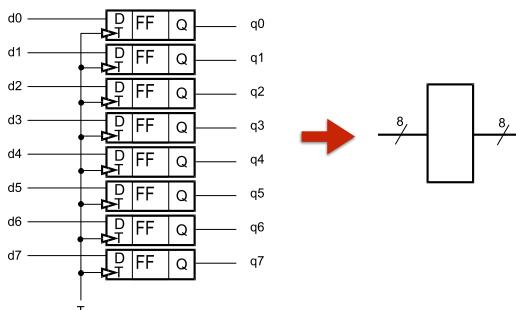
Wintersemester • Prof. Matthias Werner

58

<https://osg.informatik.tu-chemnitz.de>

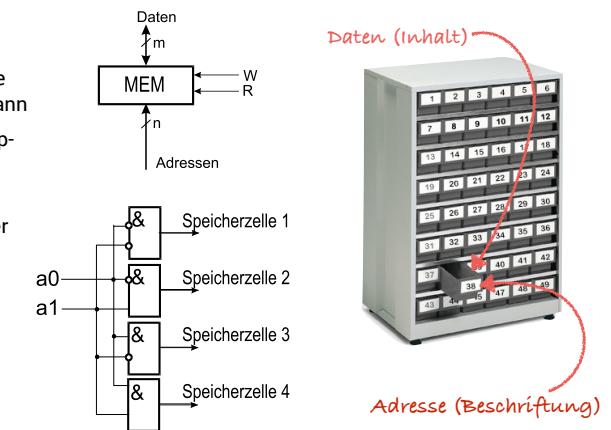
## Register

- ▶ Speicherzellen für logisch zusammengehörende Einzelbits werden zu **Registern** zusammengefasst
- ▶ Register bestehen aus unverbundenen Flip-Flops, die gemeinsam getaktet werden



## Arbeitsspeicher

- ▶ Abstraktes Modell: lineare Liste von Speicherzellen, auf die unter Angabe einer **Adresse** zugegriffen werden kann
- ▶ Halbleiterspeicher: Millionen von Flip-Flops auf einem Chip
- ▶ Das Zugangssignal für jede Speicherzelle wird aus dem Wert, der auf den Adressleitungen liegt, **decodiert**.
- ▶ Typisches Beispiel:
  - ▶  $m = 32$  (Datenwortbreite ist 32 Bit)
  - ▶  $n = 24$  (Es können  $2^{24} = 16\text{M}$  Worte angesprochen werden)



## Speicher (Forts.)

- ▶ Neben dem hier beschriebenen Prinzip des Speicherns durch Flip-Flops gibt es auch andere Speichertechnologien
- ▶ Die Struktur ...
- ▶ Speicherzellen
- ▶ Zusammenfassung zu einem Speicherwort
- ▶ Adressierung über Adressdecodierer
- ▶ ...ist aber **immer gleich**



Speicherzellen eines Mikrocontrollers mit einem Elektronenmikroskop betrachtet



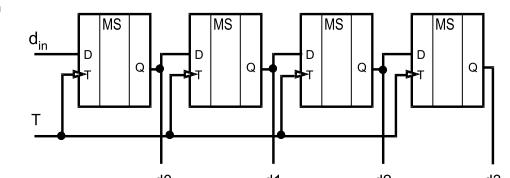
Arbeitsspeicher in Form von integrierten Schaltungen

(Bildquelle: Wikipedia)

## Schieberegister

- ▶ Mit jedem Takt wird das Datenwort um eine Stelle verschoben und das in das „freigewordene“ Bit der Wert des Eingangsbits  $d_{in}$  geschrieben:

- ▶  $t = t_0, \quad d_{in}=1: \quad D=0000$
- ▶  $t = t_0+1, \quad d_{in}=1: \quad D=0001$
- ▶  $t = t_0+2, \quad d_{in}=0: \quad D=0011$
- ▶  $t = t_0+3, \quad d_{in}=1: \quad D=0110$
- ▶  $t = t_0+4, \quad d_{in}=1: \quad D=1101$

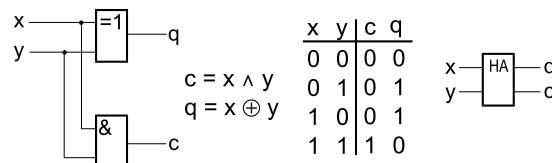


- ▶ Ist  $d_{in} = 0$ , so entspricht das Schieben in Richtung MSB (Most Significant Bit) einer **Multiplikation** mit 2, das Schieben in Richtung LSB (Least Significant Bit) einer **Division** durch 2.

## Addition

- Die Addition zweier Binärzahlen kann durch logische Verknüpfungen realisiert werden.

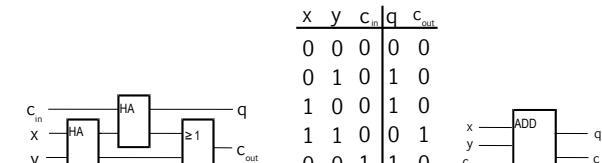
### → Halbaddierer (Half-Adder)



- Berechnung von  $x + y = z + \text{Übertrag}$  (carry)
- Übertrag zur nächsthöheren Stelle

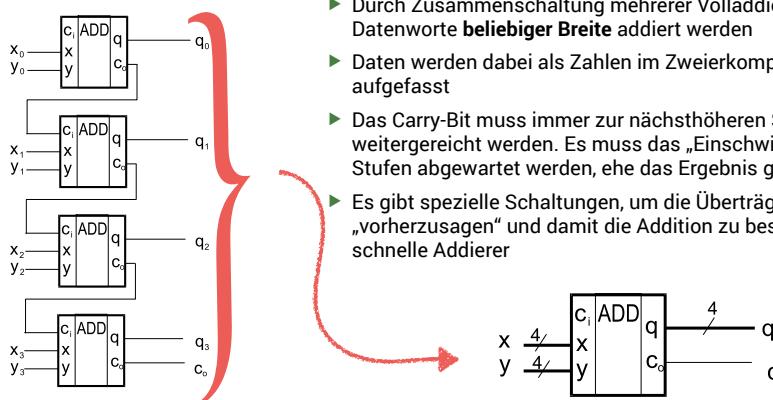
## Volladdierer

- Zur vollständigen Addition einer Stelle zweier mehrstelliger Dualzahlen muss noch der Übertrag  $c_{in}$  von vorheriger Stelle berücksichtigt werden.

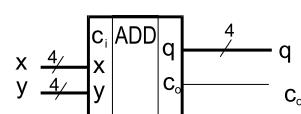


$x$	$y$	$c_{in}$	$q$	$c_{out}$
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

## n-Bit-Addierer

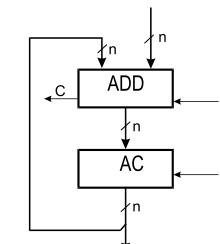


- Durch Zusammenschaltung mehrerer Volladdierer können zwei Datenworte **beliebiger Breite** addiert werden
- Daten werden dabei als Zahlen im Zweierkomplement aufgefasst
- Das Carry-Bit muss immer zur nächsthöheren Stufe weitergereicht werden. Es muss das „Einschwingen“ aller  $n$  Stufen abgewartet werden, ehe das Ergebnis gültig ist.
- Es gibt spezielle Schaltungen, um die Überträge „vorherzusagen“ und damit die Addition zu beschleunigen → schnelle Addierer



## Akkumulation

- Addition mehrerer Zahlen zusammengeschalteter Addierkette und Register (**Akkumulator**)
- fortgesetzte Addition beliebig vieler Zahlen möglich
- Akkumulator sammelt Ergebnisse vorheriger Additionen



Vorgehen:

- 1)  $AC$  löschen ( $AC := 0$ )
- 2) Addiere  $x$  zum Inhalt von  $AC$  ( $AC := AC + x$ )
- 3) Wenn nicht fertig, gehe zu 2)

Ein Programm!

## Multiplikation

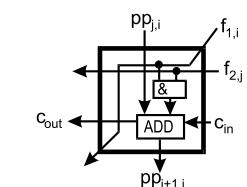
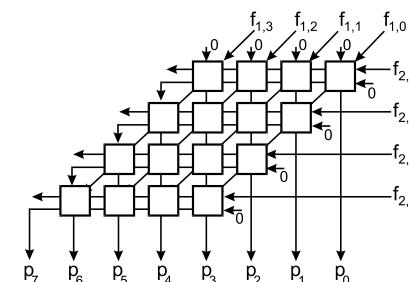
- „Papier und Bleistift“ Methode

$$\begin{array}{r} 11 \cdot 13 = 143 \\ 11 \\ \times 33 \\ \hline 143 \end{array} \quad \begin{array}{r} 11 \cdot 13 = 143 \\ 11 \\ \times 33 \\ \hline 11 \end{array}$$

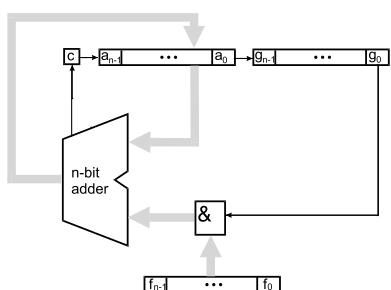
$$\begin{array}{r} 1011 \cdot 1101 \\ 1011 \quad 1 \\ 0000 \quad 0 \\ 1011 \quad 1 \\ \hline 1011 \quad 1 \\ 10001111 = 128 + 8 + 4 + 2 + 1 = 143 \end{array}$$

## Multiplikationsarray

- Das „Papier-und-Bleistift“-Verfahren wird nachempfunden:
  - Produkt ist die Summe von Partialprodukten
- Nachteil: sehr hoher Aufwand, für n Bit Multiplikation werden  $n^2$  Zellen gebraucht



## Multiplikation mit Schieberegister



- Faktoren in  $f$  und  $g$
- $a$  und  $c = 0$
- Wenn  $g_0$  gleich 1 ist wird  $f$  nach  $a$  geladen
- Das Doppelregister  $ag$  wird um eine Stelle nach rechts geschoben, das LSB des Ergebnis steht damit in  $g_{n-1}$ .
- Wenn (das neue)  $g_0$  gleich 1 ist wird  $f$  zu dem Wert in  $a$  addiert und das Ergebnis nach  $a$  geladen
- Das Doppelregister  $ag$  wird um eine Stelle nach rechts geschoben, etc.
- Das Gesamtergebnis steht am Ende in  $ag$ .