

# Technische Informatik 2 - Rechnerarchitektur

## Eigenes Skript

Studiengang Angewandte Informatik

Duale Hochschule Baden-Württemberg Karlsruhe

von

Andre Meyering

Kurs: TINF16B2  
Dozent: Prof. Dr. Jürgen Röthig  
Semester: 3. Semester (09.10.2017)  
*letzte Änderung:* 16. Oktober 2017

Dies ist das eigene Skript für „Rechnerarchitektur“ bei Herrn Prof. Dr. Jürgen Röthig für das 3. Semester im Jahr 2017. Es enthält fast alles, was im Unterricht an die Tafel geschrieben oder besprochen wurde. Die  $\LaTeX$ -Dateien sollten sich im gleichen Share befinden, in dem du diese PDF-Datei gefunden hast.

Bei Fragen, Fehlern oder Ergänzungen – oder sollten die  $\LaTeX$ -Dateien fehlen – wende dich bitte an [dhw@andremeyering.de](mailto:dhw@andremeyering.de). Ich hoffe, diese PDF hilft dir beim Lernen.

## Inhaltsverzeichnis

<b>1</b>	<b>Vorwort</b>	<b>1</b>
<b>2</b>	<b>Einstieg in die Vorlesung</b>	<b>2</b>
2.1	Digitaltechnik (Vorlesung)	2
2.2	Übersicht	2
<b>3</b>	<b>Rechner</b>	<b>3</b>
3.1	Geschichte	3
3.2	Fundamentalarchitektur	4
<b>4</b>	<b>Rechenwerk / Rechnen in Hardware</b>	<b>8</b>
4.1	Schaltnetzsynthese	8
<b>5</b>	<b>Abkürzungsverzeichnis</b>	<b>11</b>
	<b>Listingsverzeichnis</b>	<b>14</b>

## 1 Vorwort

Herr Röthig schreibt alles, was für seine Klausuren von Bedeutung ist, an die Tafel. Es ist daher nur zu empfehlen, alles mitzuschreiben, da er kein Skript besitzt und auch keinen Foliensatz. Der Unterricht ist im Vergleich zu anderen Dozenten unterscheidet sich darin, dass während der Klausur keine Hilfsmittel verwendet werden dürfen. Dafür besteht die Klausur zu 90% nur aus Abfrageaufgaben.

Dieses Skript enthält *alles*, was Herr Röthig 2017 an Wissen voraussetzt. Auf den letzten Seiten dieses Skripts findet sich zusätzlich noch eine Übungsklausur. Die Klausuren unterscheiden sich jedes Jahr nur um einige wenige Aufgaben. Ist man zwei, drei Übungsklausuren durchgegangen, so ist die Klausur einfach zu bestehen.

Zusammen mit meinem Kurs TINF16B2 haben wir dieses Skript ausgedruckt und korrigiert. Inhaltliche Fehler sollten daher (fast) keine mehr enthalten sein.

Ich wünsche dir viel Erfolg bei Herrn Röthig im Fach Rechnerarchitektur (Technische Informatik 2). Solltest du diese Skript erweitern wollen, so kannst du dich an [dhw@andremeyering.de](mailto:dhw@andremeyering.de) wenden.

## 2 Einstieg in die Vorlesung

Dozent: Prof. Dr. Jürgen Röthig

Modul: Technische Informatik II

Fach: Rechnerarchitektur

### 2.1 Digitaltechnik (Vorlesung)

- 36h/33h / 4h pro Woche
- Klausur: 21.12.2017 | 60min (ohne Hilfsmittel, Verrechnung mit Betriebssysteme)
- kein Skript, kein Foliensatz

### 2.2 Übersicht

1. Einführung, Begriffsbildung, Historie, Fundamentalarchitektur
2. Rechenwerke
3. Speicherwerk: Hauptspeicherorganisation  
Speicher: Cache, nicht-flüchtige Speichertechnologie
4. ausgewählte Kapitel aus Steuer-/Ein- und Ausgabewerke sind in 2 + 3 enthalten.

## 3 Rechner

### Rechner

Hilfsmittel zum Durchführen von „Rechnungen“.

- schneller
- fehlerfreier
- besseres Speichervermögen

### Rechenmaschine

- Abakus (mechanisch, digital)
- Rechenschieber (mechanisch, analog)

### Arbeitsweise

Man unterscheidet zwischen mechanisch vs elektrisch und digital vs analog.

Moderne „Rechner“ (PC & Co.) arbeiten elektrisch und digital. Dem gegenüber stehen elektrische Analogrechner (elektrisch und analog; um die 1920er).

## 3.1 Geschichte

### 3.1.1 Elektrischer Digitalrechner

#### ZUSE Z1, Z2 (ab ~1940)

Relais als zentrale Bauteile (elektromagnetischer Schalter mit Elektromagnet)

- ⊕ Automatismus möglich
- ⊖ langsame Geschwindigkeit
- ⊖ großer Platzverbrauch
- ⊖ Geräusche beim Schalten
- ⊖ hoher Energieverbrauch beim Schalten
- ⊖ großer Verschleiß

#### ENIAC (~1945)

Die ENIAC besitzt als zentrales Bauteil eine Elektronenröhre. Eine Elektronenröhre ist ein eigentlich analog arbeitender Verstärker, wird hier aber als digitaler Schalter genutzt. Die Funktionsweise wird in Abbildung 3.1 dargestellt, wobei die Kathode negativ und die Anode positiv geladen sind.

- ⊕ sehr hohe Geschwindigkeit
- ⊖ großer Platzverbrauch
- ⊖ ständiges Summen bei 50Hz oft möglich
- ⊖ hoher, ständiger Energieverbrauch
- ⊖ großer Verschleiß

**Moderne Rechner** Moderne transistorisierte Digitalrechner (z. B. Uniac) ab Ende der 1950er).

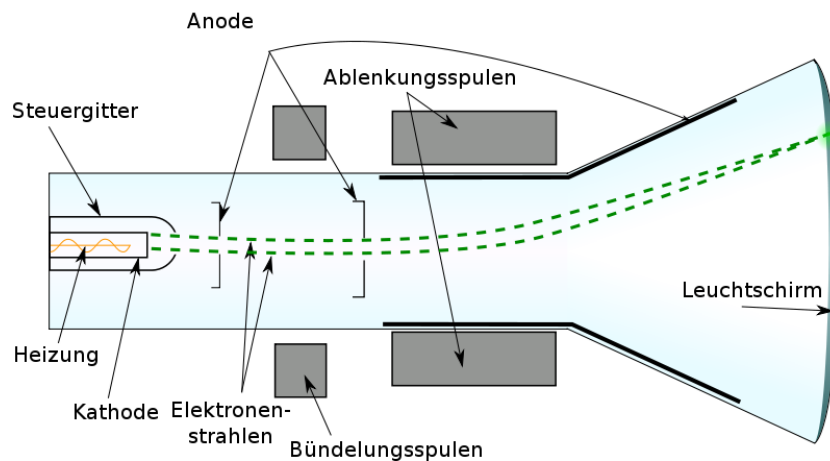


Abbildung 3.1: Funktionsweise Kathodenstrahlröhre [Quelle: Wikipedia]

- Transistor als zentrales Bauteil. Ein Transistor ist ein analog arbeitender Verstärker, wird hier aber als digital arbeitender Schalter genutzt.
- ⊕ sehr hohe Geschwindigkeit
- ⊕ sehr geringer Platzverbrauch
- ⊕ keine Geräuschentwicklung (außer Lüfter)
- ⊕ sehr niedriger Energieverbrauch
- ⊕ geringer Verschleiß

## 3.2 Fundamentalarchitektur

### 3.2.1 von-Neumann-Architektur

In Abbildung 3.2 wird die von-Neumann-Architektur vereinfacht dargestellt.

**Zentraleinheit (CPU)** Die CPU besteht aus:

**Rechenwerk** Rechnen mit Zahlen und logischen Werten

**Steuerwerk** Zuständig für das Steuern und Koordinieren aller anderen Komponenten ⇒ Interpretation und Ausführung des (Maschinensprachen-)Programms

**Speicherwerk** (Hauptspeicher, Primärspeicher)

Speichern von Informationen (sowohl Programmcode als auch Nutzdaten gleichermaßen)

**Bus** verbindet alle Komponenten und ermöglicht den Informationsaustausch/Datenfluss zwischen ihnen.

**Eingabewerk** „Schnittstelle“ für Eingabegeräte (z. B. USB-Controller, S-ATA-Controller) nicht jedoch das Peripheriegerät selbst (also nicht die Tastatur)

**Ausgabewerk** „Schnittstelle“ für Ausgabegeräte (z. B. Grafikkarte)

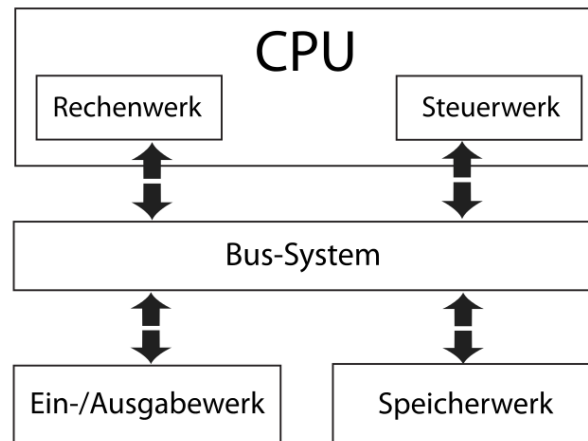


Abbildung 3.2: Vereinfachte Darstellung der von-Neumann-Architektur

### 3.2.2 Harvard-Architektur

Die Harvard-Architektur ist ähnlich der von-Neumann-Architektur, besitzt aber anstatt eines gemeinsamen, zwei getrennte Speicherwerke für Nutzdaten und Programmcode. Zusätzlich kann noch ein optionales zweites Eingabewerk existieren, welches nur für den Programmcode vorhanden ist. Das Speicher- und Eingabewerk für den Programmcode wird über einen zweiten Bus angebunden.

Dadurch ist eine klare physikalische Trennung von Programmcode und Nutzdaten möglich. Abbildung 3.3 zeigt die Harvard-Architektur und wie sich diese von der von-Neumann-Architektur unterscheidet.

### 3.2.3 Vergleich

Tabelle 3.1 auf Seite 7 vergleicht die von-Neumann-Architektur mit der Harvard-Architektur.

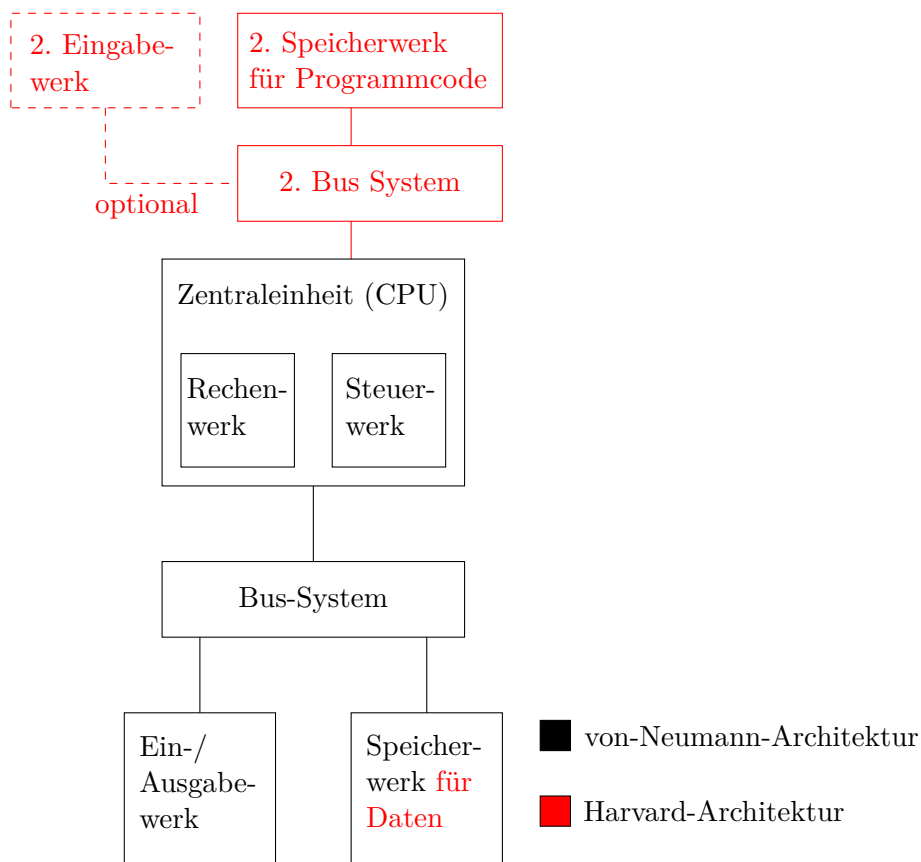


Abbildung 3.3: Vereinfachte Darstellung der Harvard-Architektur



von-Neumann-Architektur	Harvard-Architektur
<ul style="list-style-type: none"> <li>⊖ Virenanfälligkeit: Nutzdaten können als Programm ausgeführt werden</li> <li>⊕ universelle Programmierbarkeit (z. B. Compiler-Ausgabe wird als Programme ausgeführt)</li> <li>⊕ flexible Speicheraufteilung zwischen Programmcode und Daten</li> <li>⊖ möglicher Flaschenhals Bus &amp; Speicherwerk</li> <li>⊕ kostengünstig</li> </ul>	<ul style="list-style-type: none"> <li>⊕ nahezu immun gegen unabsichtlichen Virenbefall</li> <li>⊕ keine (unbeabsichtigten oder ungewollten) Änderungen an der Betriebssoftware möglich</li> <li>⊖ komplexer und teurer (durch 2 Bus und ggf. 2 Eingabewerke)</li> <li>⊖ schwer update-fähig</li> <li>⊕ bessere Performance möglich durch gleichzeitigen Zugriff auf beide Speicherwerke</li> <li>⊖ ohne zweites Eingabewerk keine Möglichkeit anderen Programmcode auszuführen.</li> <li>⊖ unflexible Aufteilung des Speichers, wenn das eine Speicherwerk voll ist, kann das andere Speicherwerk nicht genutzt werden.</li> </ul>
Einsatz	
<ul style="list-style-type: none"> <li>• übliche PC-Architektur – „Universal-PC“</li> </ul>	<ul style="list-style-type: none"> <li>• „embedded systems“ (z. B. in Waschmaschinen, KFZ-Elektronik, etc.)</li> <li>• Smartphones &amp; Co.</li> <li>• Bestandteile von PCs: BIOS, CPU-Cache in modernen CPUs (Trennung in Cache für Programmcode und Nutzdaten), NX-Flag (Non-Executable) im Hauptspeicher</li> </ul>

Tabelle 3.1: Vergleich der von-Neumann- und Harvard-Architektur

## 4 Rechenwerk / Rechnen in Hardware

### Grundlagen: Addition

4 7 1 1	Addition von mehrstelligen Zahlen wird reduziert auf die Addition von
+ 0 8 1 5	zwei (oder drei) einstelligen Zahlen (bzw. Ziffern) zu einer einstelligen
Übertrag 0 1 0 0	Zahl sowie einem einstelligen Übertrag, also einer zweistelligen Zahl als
5 5 2 6	Ergebnis. $\Rightarrow$ genauso im Binärsystem

### 4.1 Schaltnetzsynthese

#### 4.1.1 Wiederholung/Grundlegendes

**Schaltwerk** „Hat ein Gedächtnis“, da eine Rückkopplung vorliegt

**Schaltnetz** Kann nur die derzeitigen Eingangsdaten verarbeiten, da keine Rückkopplung vorliegt

**Vollkonjunktion/Minterm** UND-Verknüpfung aller vorkommenden Variablen entweder in negierter oder nicht-negierter Form

**Disjunktive Normalform (DNF)** Eine Disjunktion (ODER-Verknüpfung) von Konjunktionstermen (UND-Verknüpfungen).

**Disjunktive Minimalform (DMF)** Ist die minimale Darstellung einer Ausgabefunktion und damit eine Vereinfachung einer DNF

#### 4.1.2 Halbaddierer

Addition von zwei einstelligen Binärzahlen  $a$  und  $b$  zu einer zweistelligen Binärzahl  $c_{out}s$  (Übertrag und Summe). Schaltsymbol und Schaltnetz des Halbaddierer werden in Abbildung 4.1 dargestellt.

Die folgende Tabelle zeigt den Gedankenweg, wie ein Halbaddierer funktioniert.

Nr.	$b$	$a$	$c_{out}$	$s$	Minterm	DNF
0	0	0	0	0		
1	0	1	0	1	$\bar{b}a$	$c_{out} = ba$
2	1	0	0	1	$b\bar{a}$	$s = \bar{b}a \vee b\bar{a}$
3	1	1	1	0	$ba$	$\Rightarrow$ beides gleichzeitig auch DMF

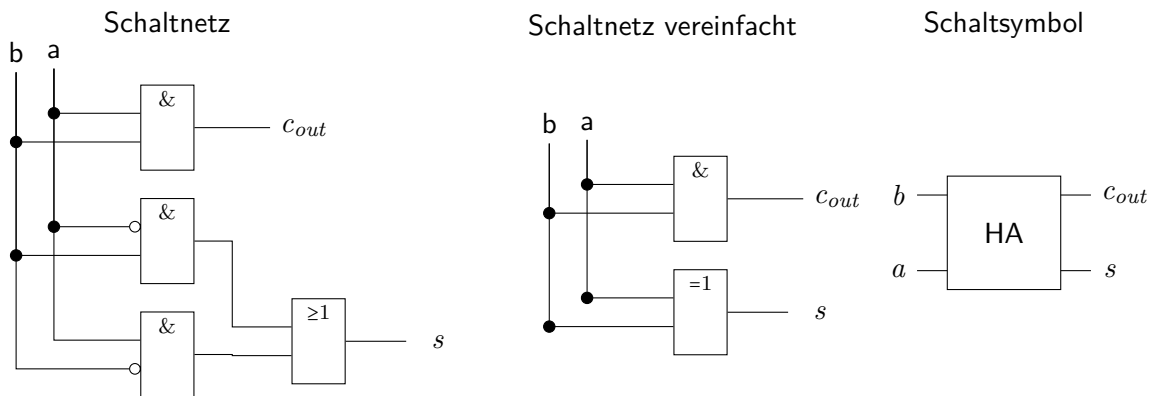


Abbildung 4.1: Halbaddierer – Schaltnetz und Schaltsymbol

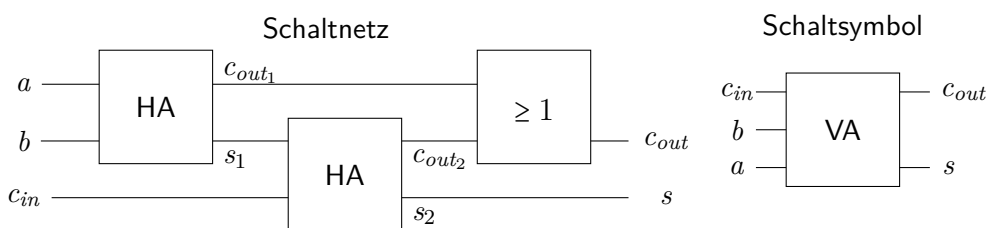


Abbildung 4.2: Volladdierer – Schaltnetz und Schaltsymbol

### 4.1.3 Volladdierer

Addition von drei einstelligen Binärzahlen  $a$ ,  $b$  und  $c_{in}$  zu einer zweistelligen Binärzahl  $c_{out}s$  (Übertrag und Summe). Schaltsymbol und Schaltnetz des Volladdierers werden in Abbildung 4.2 dargestellt.

*Hinweis:* Für die Verknüpfung von  $c_{out1}$  und  $c_{out2}$  zu  $c_{out}$  wäre eigentlich ein XOR notwendig. Da aber der Fall  $c_{out1} = c_{out2} = 1$  (also beiden Eingänge des XOR „1“) nie auftritt, reicht ein einfaches OR.

### 4.1.4 Paralleladdierer (4-Bit-Ripple-Carry-Paralleladdierer RC-PA)

Der RC-PA ist ein mehrstelliger Addierer für Binärzahlen. In den folgenden Beispielen ist er ein Addierer vierstelliger Binärzahlen  $a_3a_2a_1a_0$  und  $b_3b_2b_1b_0$ . Das Ergebnis ist  $s_4s_3s_2s_1s_0$  und somit eine 5-stellige Zahl.

Abbildung 4.3 zeigt das Schaltnetz und Schaltsymbol eines Paralleladdierers.

*Hinweis:* Ein  $n$ -Bit RC-PA ist ein Schaltnetz, kein Schaltwerk. Eine zeichnerische Anordnung mit Verbindungen nur nach unten ist nämlich möglich.

#### Schaltungsanalyse:

Bestimmung des „Aufwands“. Aufwand kann sein:

- „Hardware-Aufwand“ (in Transistoren)

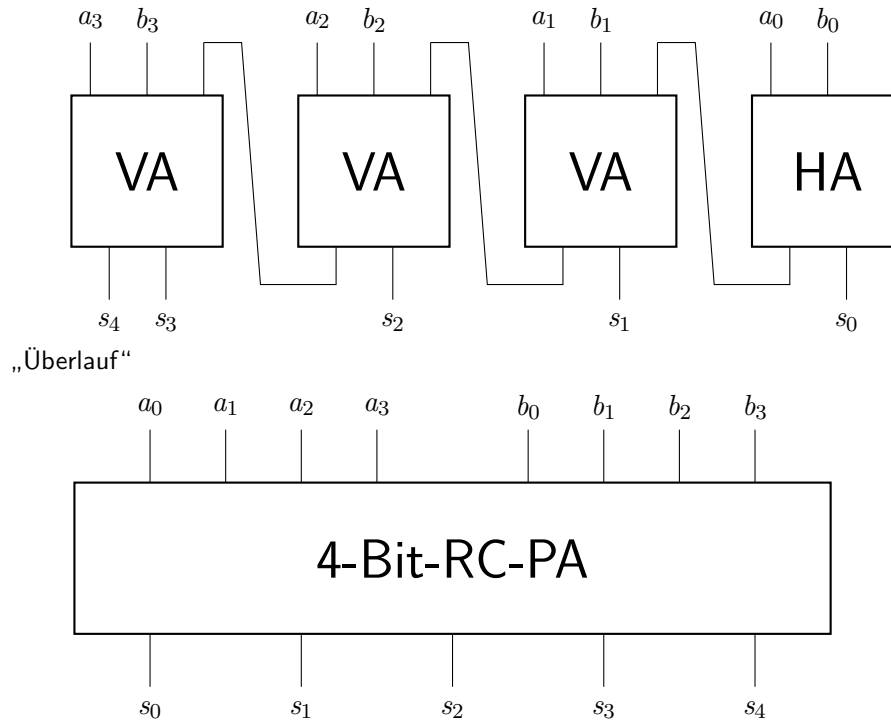


Abbildung 4.3: Paralleladdierer – Schaltnetz und Schaltsymbol

- Zeitaufwand

### Warum Zeitaufwand?

Es wird der Zeitaufwand betrachtet, da Gatter Schaltzeiten haben, typischerweise  $\sim 10$  Piko-sekunden. Insgesamt werden bei einem Signaldurchgang auf dem IC sehr viele Gatter durchlaufen. Damit sind die Schaltzeiten um Größenordnungen größer als die reine Laufzeit der Signale auf dem Leiter angegeben (letztere wird vernachlässigt, Zeitverzögerung wird in „Anzahl Gatterlaufzeiten“/GLZ).

### Hardwareaufwand des 4 Bit RC-PA

für HA: 2 Transistoren für  $c_{out}$  und 6 Transistoren für  $s \Rightarrow 8$  Transistoren

für VA: 2 HA und 2 Transistoren für  $c_{out} \Rightarrow 2 \cdot 8 + 2 = 18$  Transistoren

Benötigt werden:

1 HA = 8 Transistoren

$(n - 1)$  VA =  $(n - 1) \cdot 18$  Transistoren =  $(18n - 18)$  Transistoren

$\Rightarrow$  insgesamt:  $8 + 18n - 18 = 18n - 10$  Transistoren =  $O(n)$

Dies heißt, dass der HW-Aufwand linear mit der Breite der Summanden steigt. Dies ist gut, denn besseres (also weniger Aufwand) ist kaum zu erwarten.

## 5 Abkürzungsverzeichnis

<b>CPU</b>	central processing unit	
<b>DMF</b>	Disjunktive Minimalform .....	8
<b>DNF</b>	Disjunktive Normalform .....	8
<b>IC</b>	Integrated Circuit	
<b>GLZ</b>	Gatterlaufzeit	
<b>HA</b>	Halbaddierer	
<b>HW</b>	Hardware	
<b>RC-PA</b>	Ripple-Carry-Paralleladdierer	
<b>VA</b>	Volladdierer	

## Abbildungsverzeichnis

3.1	Funktionsweise Kathodenstrahlröhre [Quelle: Wikipedia] . . . . .	4
3.2	Vereinfachte Darstellung der von-Neumann-Architektur . . . . .	5
3.3	Vereinfachte Darstellung der Harvard-Architektur . . . . .	6
4.1	Halbaddierer – Schaltnetz und Schaltsymbol . . . . .	9
4.2	Volladdierer – Schaltnetz und Schaltsymbol . . . . .	9
4.3	Paralleladdierer – Schaltnetz und Schaltsymbol . . . . .	10

## Tabellenverzeichnis

3.1 Vergleich der von-Neumann- und Harvard-Architektur . . . . .	7
--	---

## **Listingsverzeichnis**