Rechnerstrukturen

3. Elementare Bausteine

Inhalt

- Latches und Register
- Decoder
- Multiplexer
- Speicher
- Arithmetische Einheiten
- Endliche Automaten

3.2

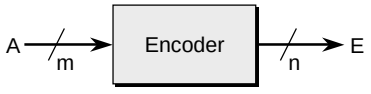
Elementare Bausteine

- Häufig verwendete Grundfunktionen
  - Umwandeln (Decoder)
  - Verteilen (Multiplexer) und Zusammenfassen (Demultiplexer)
  - Speicherung
  - Steuerungen (Endliche Automaten, State Machine)
  - Arithmetisch-logische Funktionen
    - Zählen (Zähler, Counter)
    - Addieren, Subtrahieren, ...
    - Vergleichen (Komparator)
    - Shiften, Rotieren
- Funktionsweise, Realisierungsvarianten, Zeit- und Platzeffizienz
- Spezifikation
  - Ein- und Ausgänge, Steuerungsanschlüsse
  - Zeitverhalten, elektrische Eigenschaften

3.3

Decoder

- Umwandlung
- Inverse Funktion häufig auch interessant
  - Decoder
  - Encoder
- Beispiele
  - 7-Segment
  - 1 aus n
  - Priorität
  - Parität
- Klassische Schaltnetze
  - Minimierung



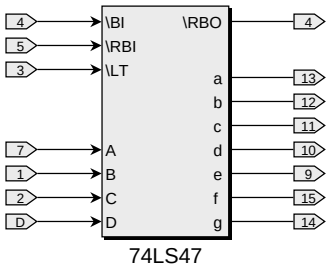
3.4

7-Segment-Decoder

3.5

Realisierung: 7-Segment-Decoder

- Zusätzliche Steuereingänge
  - LT = Lamp Test
  - BI = Blank Input
  - RBI = Ripple Blank Input
  - RBO = Ripple Blank Output



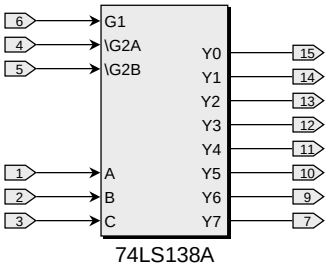
74LS47

\LT	\RBI	D	C	B	A	\BI/\RBO	a	b	c	d	e	f	g
H	H	L	L	L	L	H	L	L	L	L	L	L	H
H	X	L	L	L	H	H							
.	.	.	.	.	.	.	.	.	.	.	.	.	.
H	X	H	H	H	H	H							
X	X	X	X	X	X	L	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
L	X	X	X	X	X	H	L	L	L	L	L	L	L

3.6

“1 aus n”-Dekoder

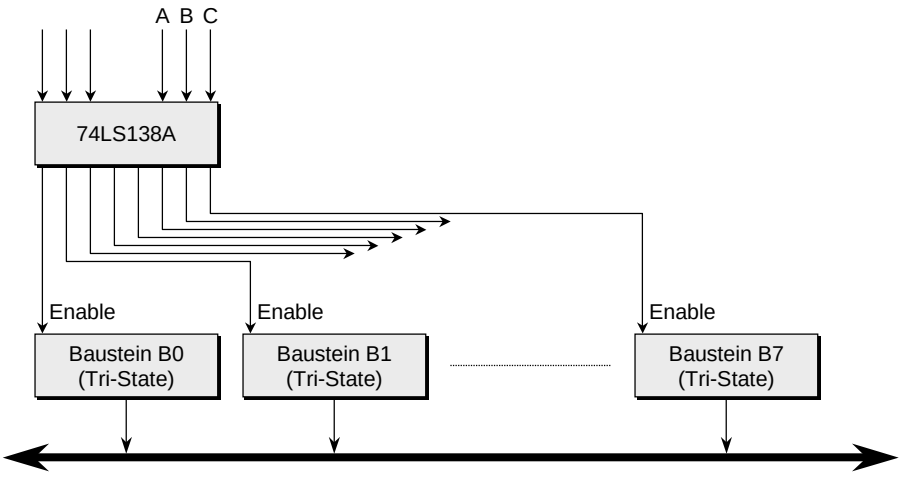
- Maximal 1 aus n Ausgängen aktiv
  - 1 aus 4
  - 1 aus 8
  - 1 aus 16
- Kaskadierbarkeit



G1	\G2A	\G2B	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	X	H	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

3.7

Primäre Anwendung eines “1 aus n”-Decoders

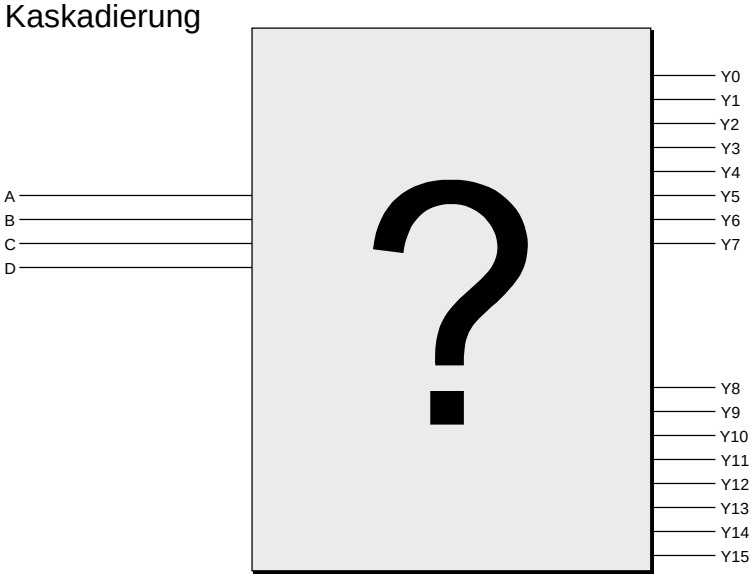


3.8

Realisierung: “1 aus n”-Decoder

3.9

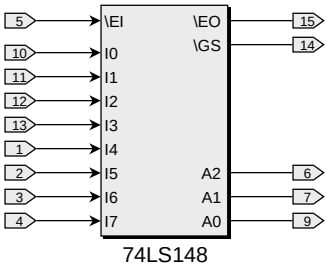
Kaskadierung



3.11

“1 aus n”-Encoder (Prioritätsdecoder)

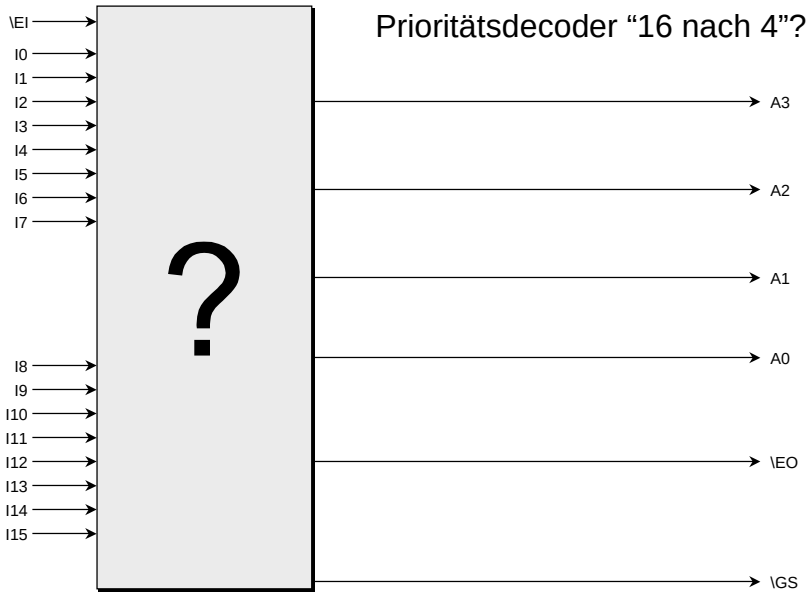
- Umkehrung des “1 aus n”-Decoders
  - 8 nach 3
  - 16 nach 4



\EI	I0	I1	I2	I3	I4	I5	I6	I7	A2	A1	A0	\GS	\EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

3.14

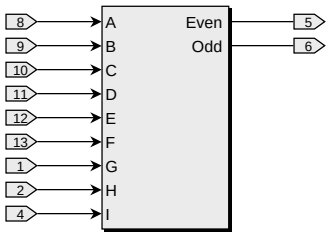
Prioritätsdecoder “16 nach 4”?



3.15

Paritätsdecoder

- Einfaches Prüfsummenverfahren
  - Gerade Parität: n Bits enthalten gerade Anzahl 1
  - Ungerade Parität: n Bits enthalten ungerade Anzahl 1
- Wahrheitstabelle etwas umständlich?
- Baustein einsetzbar
  - Erzeugung des Paritätsbit
  - Überprüfung des Paritätsbit



74LS280

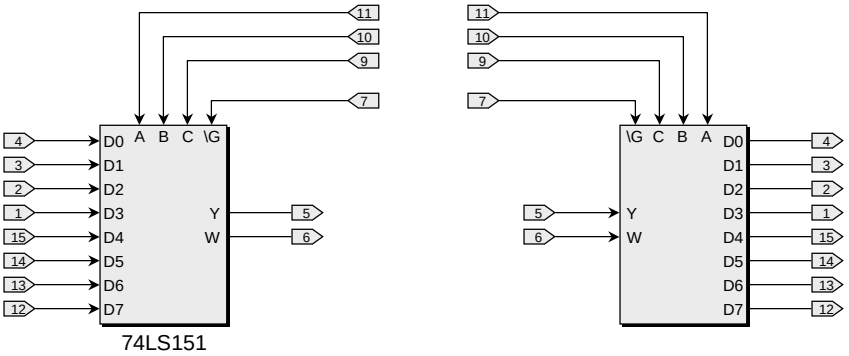
3.17

Realisierung: Paritätsdecoder

3.18

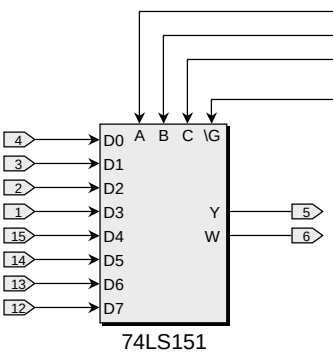
Multiplexer und Demultiplexer

- Erweiterung des einfachen Wechselschalters auf n Eingänge
- Demultiplexer: Umkehrung des Multiplexers



3.20

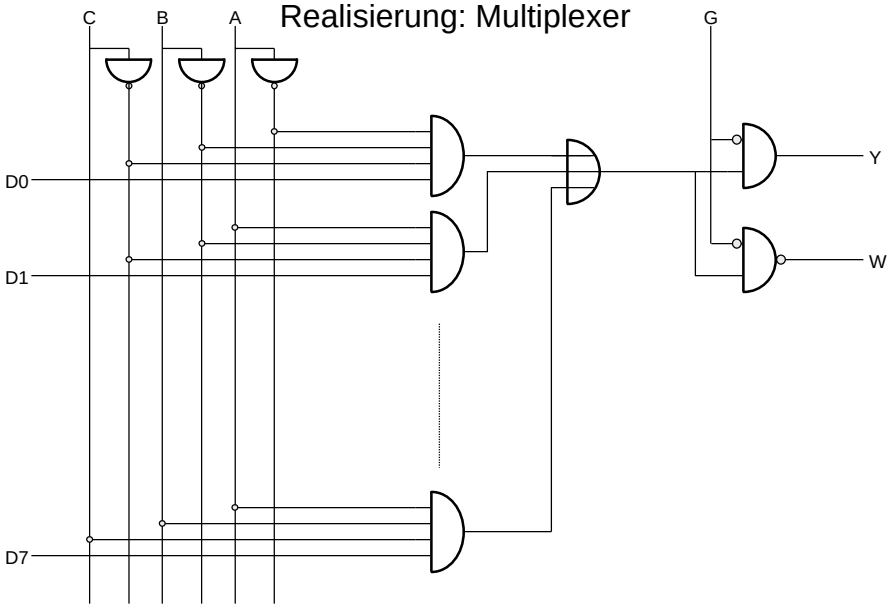
Funktionsweise Multiplexer



C	B	A	\G	Y	W
X	X	X	H	L	H
L	L	L	L	D0	\D0
L	L	H	L	D1	\D1
L	H	L	L	D2	\D2
L	H	H	L	D3	\D3
H	L	L	L	D4	\D4
H	L	H	L	D5	\D5
H	H	L	L	D6	\D6
H	H	H	L	D7	\D7

3.21

Realisierung: Multiplexer



3.23

Kaskadierung

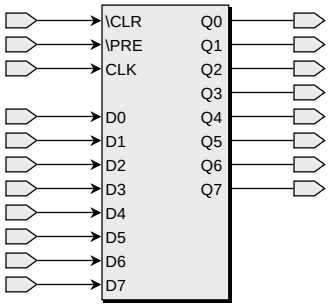
3.24

Register

- Speicherung von Bitvektoren
- Zusammenfassen mehrerer Flip-Flops
  - Manuell über externe Verdrahtung
  - Spezielle Registerbausteine
- Breites Spektrum
  - Verschiedene Flip-Flops
  - Preset
  - Clear
  - Tri-State-Ausgänge

3.26

Beispiel: 8 Bit-Register

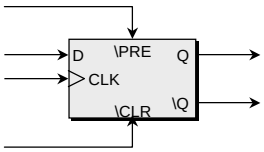


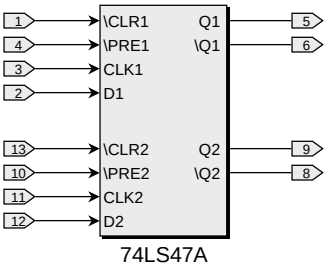
\CLR	\PRE	CLK	Dx	Qx
L	L	X	X	X
L	H	X	X	L
H	L	X	X	H
H	H	↑	L	L
H	H	↑	H	H

3.27

Realisierung: Mit 74LS47A

- 74LS47A: 2 D-Flip-Flop
  - Preset
  - Clear
- Wieviele Chips sind notwendig?
- Wieviele Ein/Ausgänge?



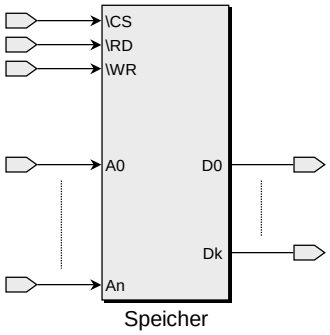


74LS47A

Realisierung: 8 Bit-Register?

Einfache Speicher

- Adressierung einzelner Speicherzellen
- Speicherbaustein
  - Kapazität (Anzahl Zellen, 2^n)
  - Größe des Datenvektors pro Adresse (k Bit)
- Tri-State-Ausgänge
- Häufig kombinierte Leitung R/$\overline{W}$
  - High = Lesen
  - Low = Schreiben
- Zwei Grundtypen
  - RAM: Lesen und Schreiben
  - ROM: Nur Lesen



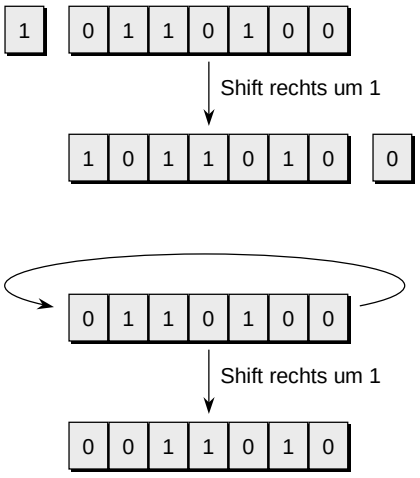
3.31

Realisierung: 16 mal 4 Bit RAM?

3.32

Shiften und Rotieren

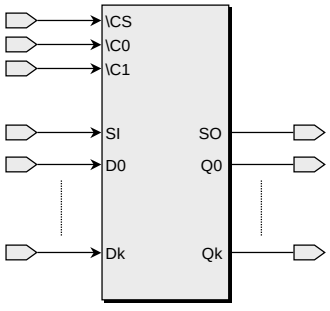
- Shift
  - Bitvektor um k Bit nach links oder rechts schiften
  - k Bit "fallen raus"
  - k Bit kommen hinzu
- Rotieren
  - Bitvektor um k Bit nach links oder rechts zyklisch shiften
- Betrachtung von Shift reicht?



3.37

Beispiel: Shift-Register

- 4 Zustände
  - Wert speichern
  - Wert laden
  - Shift links um 1
  - Shift rechts um 1
- Tri-State-Ausgang



\CS	\C0	\C1	Dx	Qx	SO	
L	X	X	Ω	Ω	Ω	
H	L	L	X	Qx	X	Hold
H	L	H	Dx	Dx	X	Load
H	H	L	X	Qx+1	Q0	Shift right, Qk=SI
H	H	H	X	Qx-1	Qk	Shift left, Q0=SI

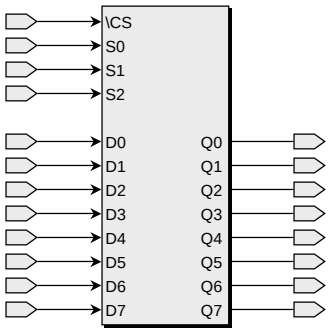
3.38

Realisierung: 2 Bit-Shift-Register?

3.39

Barrel-Shifter

- Jede beliebige Shift-Operation eines k Bit-Vektors in einem Schritt
- Beispiel k=8
- Realisierung: Multiplexer
  - Für jeden Ausgang: 2^k nach 1

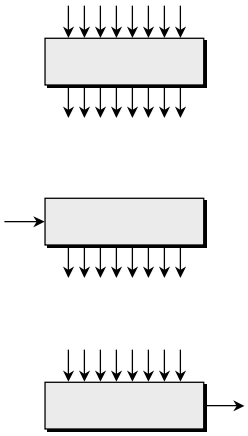


\CS	S0	S1	S2	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
L	L	L	L	D7	D6	D5	D4	D3	D2	D1	D0
L	L	L	H	D6	D5	D4	D3	D2	D1	D0	D7
L	L	H	L	D5	D4	D3	D2	D1	D0	D7	D6
L	L	H	H	D4	D3	D2	D1	D0	D7	D6	D5
L	H	L	L	D3	D2	D1	D0	D7	D6	D5	D4
L	H	L	H	D2	D1	D0	D7	D6	D5	D4	D3
L	H	H	L	D1	D0	D7	D6	D5	D4	D3	D2
L	H	H	H	D0	D7	D6	D5	D4	D3	D2	D1

3.41

Serielle/Parallele Shift-Register

- Bisher Parallel/Parallel
  - Eingabe: Bitvektor
  - Ausgabe: Bitvektor
- Umwandlung Seriell-Parallel
- Seriell nach Parallel
  - Eingabe: 1 Bit
  - Ausgabe: Bitvektor
- Parallel nach Seriell
  - Eingabe: Bitvektor
  - Ausgabe: 1 Bit



3.42

8 Bit-Seriell In/Parallel Out-Shift-Register?

- Welche Ein- und Ausgänge sind notwendig?
- Realisierung?

3.43

Logische und arithmetische Operationen

- Bitvektor
  - Logische Operationen bitweise
  - Gleichheit
- Bitvektor wird als Dualzahl aufgefaßt
  - Ganze Zahl ohne Nachkomma, Festkommazahl
  - Addition, Subtraktion, Multiplikation, Division
  - Gleichheit
  - Größer, Kleiner
- Bitvektor ist BCD-kodierte Dezimalzahl
  - Addition, Subtraktion, Multiplikation, Division
  - Gleichheit
  - Größer, Kleiner
- Bitvektor ist IEEE 754-kodierte Gleitkommazahl

Exkurs: Binäre Zahlensysteme

- Hier nur ganze Zahlen oder Festkommazahlen
- Zahlen zur Basis 2

$$01101.111_2$$
$$= 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3}$$
$$= 8 + 4 + 1 + 0.5 + 0.25 + 0.125 = 13.875_{10}$$

- Feste Anzahl Stellen
  - 8, 16, 32, 64 Bit gängig
  - Führende Nullen
- Darstellung negativer Zahlen?

2er-Komplement

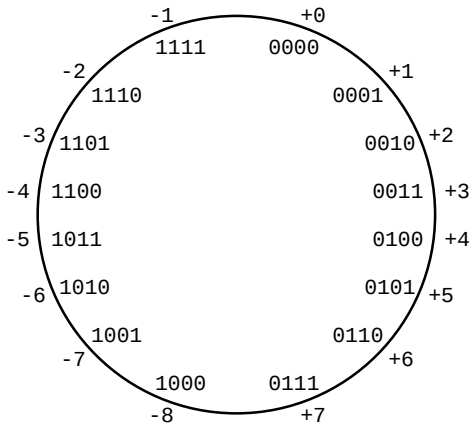
- Zahl negativ ⇔ MSB = 1

$$-N = 2^n - N$$

- Einfache Umwandlung
  1) 1er-Komplement
     - $0 \rightarrow 1$
     - $1 \rightarrow 0$
  2) 1 addieren

```
-00110101 (-53)

 11001010
+00000001
 11001011
```



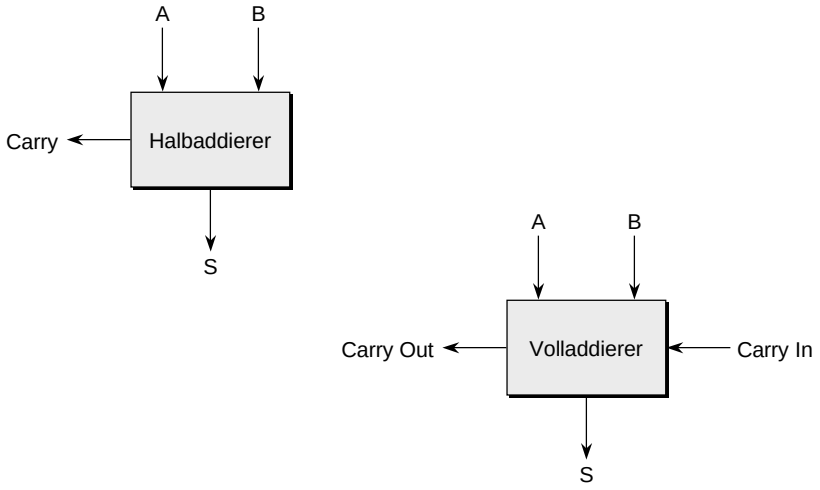
3.49

Addition

```
  00101111
+ 01111101
----------
  10101100
```


3.50

Halb- und Volladdierer?



Halb- und Volladdierer

- Wahrheitstabelle: Halbaddierer

A	B	S	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Lösung

$$S = A \oplus B$$

$$CO = A \cdot B$$

- Wahrheitstabelle: Volladdierer

A	B	CI	S	CO
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

- Lösung

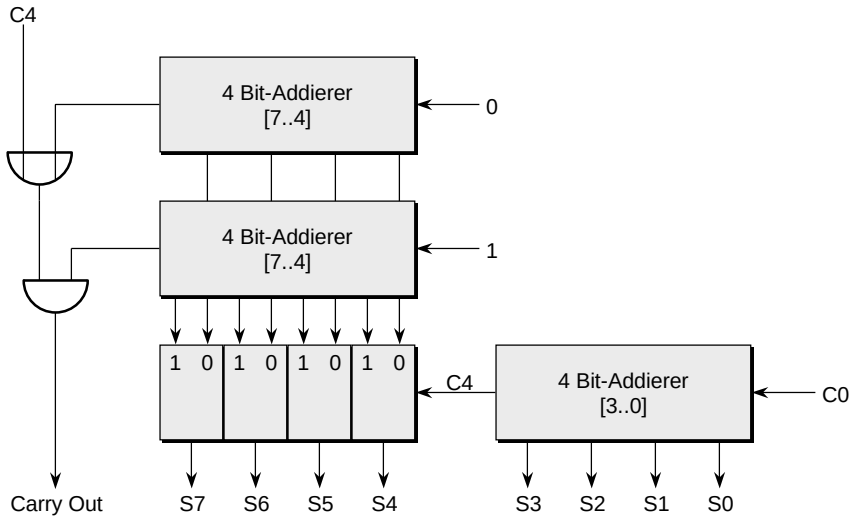
$$S = A \oplus B \oplus CI$$

$$CO = A \cdot B + A \cdot CI + B \cdot CI$$

8 Bit-Ripple-Addierer

3.53

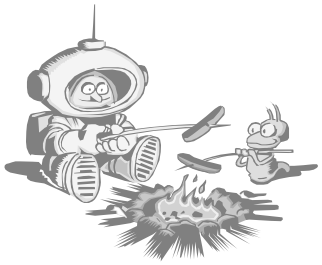
Carry-Select-Addierer

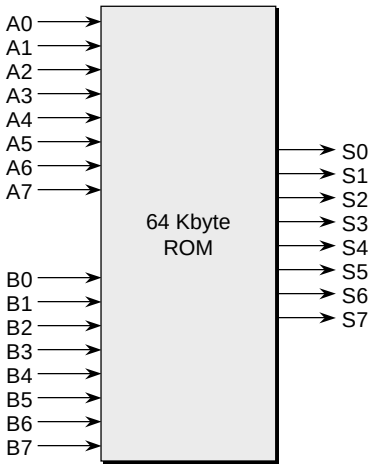


3.55

Tabellenbasierter Addierer

- Vorberechnete Summen in Nurlesespeicher
- Carry In, Carry Out?
- Zugegeben, für Addition vielleicht etwas übertrieben, aber ...





3.56

Carry-Lookahead-Addierer

- Ripple-Addierer

  $C_{i+1} = A_i B_i + C_i A_i + C_i B_i$

  $C_0 = 0$

  – Laufzeit O(n)

- Carry-Lookahead
  – Direkte Berechnung eines Carry-Bits

    $C_{i+1} = f(A_0, \ldots, A_i, B_0, \ldots, B_i)$

  – Zweistufiges Schaltnetz
  – Laufzeit O(1)

3.57

Carry-Generate und Carry-Propagate

- Wann entsteht ein Carry?
  $G_i = A_i \cdot B_i$
- Wann wird ein Carry weitergegeben?
  $P_i = A_i \oplus B_i$
- Addition durch G und P beschreiben

$$
\begin{aligned}
S_i &= A_i \oplus B_i \oplus C_i = P_i \oplus C_i \\
C_{i+1} &= A_i B_i + A_i C_i + B_i C_i \\
&= A_i B_i + C_i (A_i + B_i) \\
&= A_i B_i + C_i (A_i \oplus B_i) \\
&= G_i + C_i P_i
\end{aligned}
$$

3.58

4 Bit Carry-Lookahead-Addierer

3.59

Subtraktion

- 2er-Komplement addieren
- Erweiterung n Bit-Addierer zu n Bit-Addierer/Subtrahierer?

3.61

N Bit-Addierer/Subtrahierer

3.62

Multiplikation

- Vorzeichenlos
- Einfach erweiterbar auf vorzeichenbehaftete Multiplikation?
- Maximale Größe des Resultats?
  - n Bit Multiplikant
  - n Bit Multiplikator
- Schaltnetz
- Akkumulation der partiellen Produkte

```
   1001 (9)
 * 1011 (11)
------------
   1001
  1001
 0000
1001
------------
1100011
```


3.63

Akkumulation der partiellen Produkte?

			A_3	A_2	A_1	A_0
			B_3	B_2	B_1	B_0
S_6	S_5	S_4	S_3	S_2	S_1	S_0

3.64

Akkumulation der partiellen Produkte

			A_3	A_2	A_1	A_0	
			B_3	B_2	B_1	B_0	
			$A_3 \cdot B_0$	$A_2 \cdot B_0$	$A_1 \cdot B_0$	$A_0 \cdot B_0$	
		$A_3 \cdot B_1$	$A_2 \cdot B_1$	$A_1 \cdot B_1$	$A_0 \cdot B_1$		
	$A_3 \cdot B_2$	$A_2 \cdot B_2$	$A_1 \cdot B_2$	$A_0 \cdot B_2$			
$A_3 \cdot B_3$	$A_2 \cdot B_3$	$A_1 \cdot B_3$	$A_0 \cdot B_3$				
S_6	S_5	S_4	S_3	S_2	S_1	S_0	

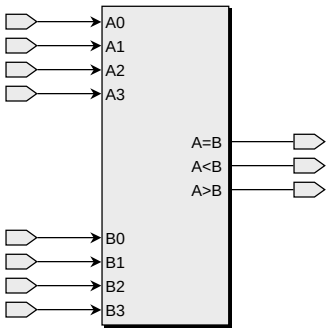
3.65

Realisierung mit Halb- und Volladdierern?

3.66

Komparatoren

- Test auf
  - Gleichheit
  - Kleiner
  - Größer
- Realisierung
  4 Bit-Komparator?



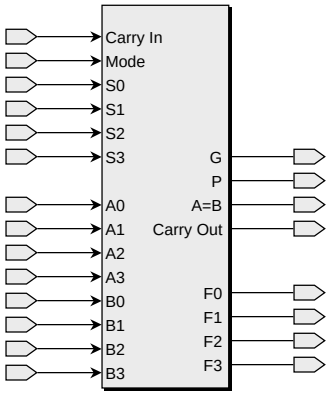
3.68

Realisierung 4 Bit-Komparator

3.69

Arithmetische-logische Einheit (ALU)

- Zusammenfassung vieler arithmetischer und logischer Operation
- Beispiel 74LS181
  - Mode = 1: Logik
    - 16 Funktionen
  - Mode = 0: Arithmetik
    - 32 Funktionen
    - z.T. außergewöhnlich
      F=AB plus (A+ not B) plus 1
  - G, P?
  - Auch als Komparator einsetzbar



74LS181

3.70

2 Bit-ALU

3.71

BCD-Arithmetik

- BCD = Binary Coded Decimal

```
0000  0
0001  1
...
1001  9
```

- Beispiel Addition
  - Realisierungsvarianten?
    - Explizite Modellierung
    - Binäre Addition und Korrektur?
- Umwandlung Binär ↔ BCD

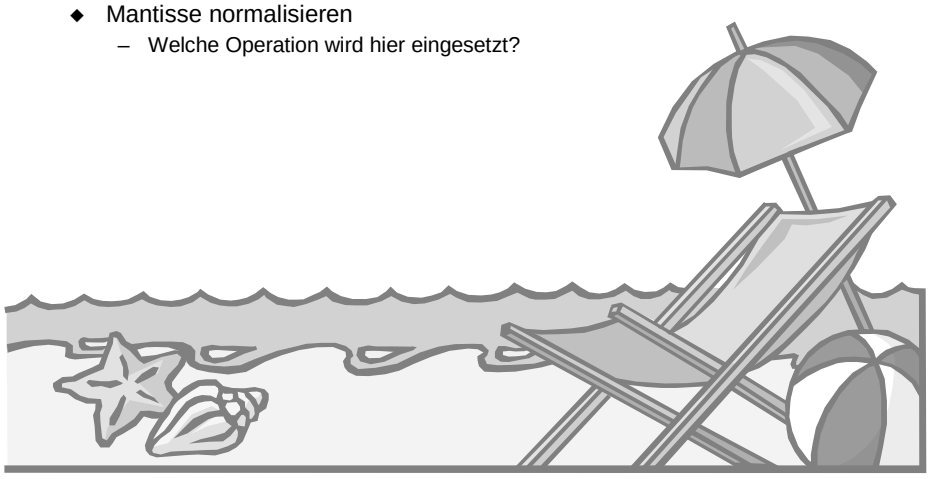
```
      0111 (7)
+     0110 (6)
      1101₂ (13)
```

```
      0111 (7)
+     0110 (6)
    1 0011_BCD (1,3)
```

3.73

Gleitkomma-Arithmetik

- Getrennte Behandlung von Mantisse und Exponent
- Mantisse normalisieren
  - Welche Operation wird hier eingesetzt?

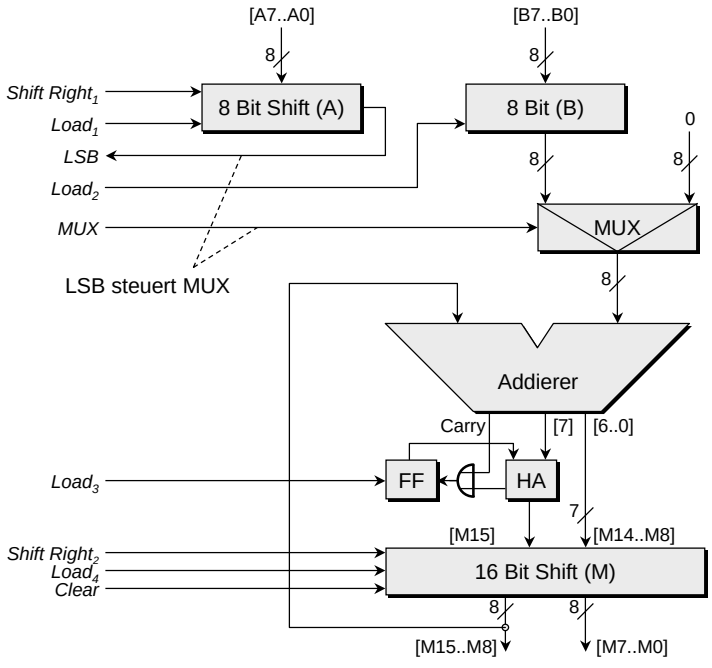


3.74

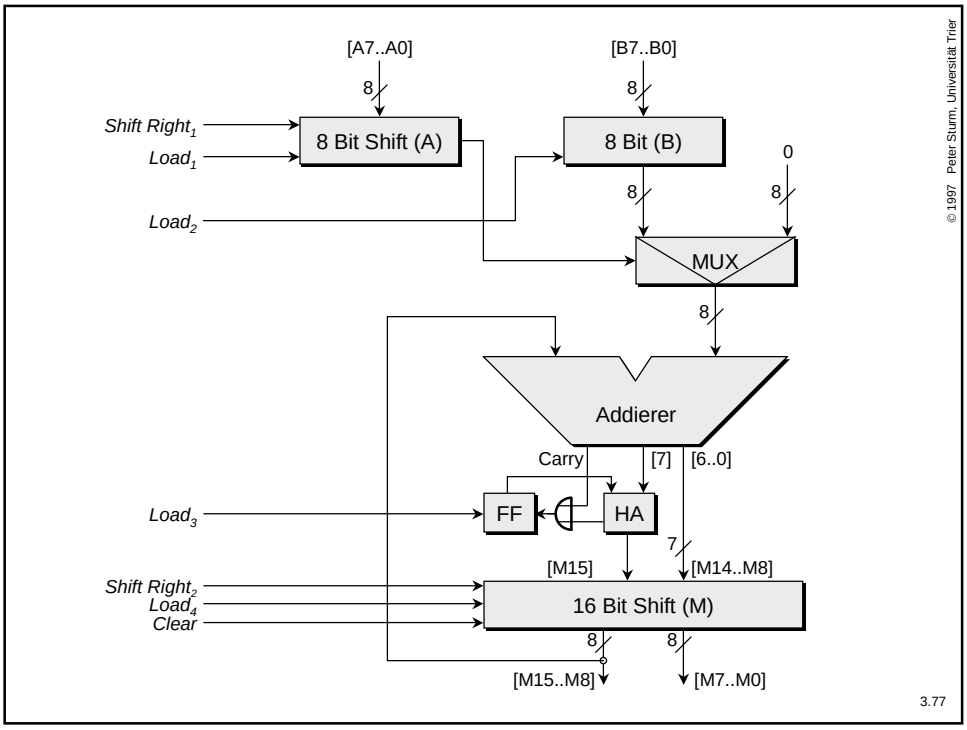
Projekt TVMUL: Iterative 8-Bit Multiplikation

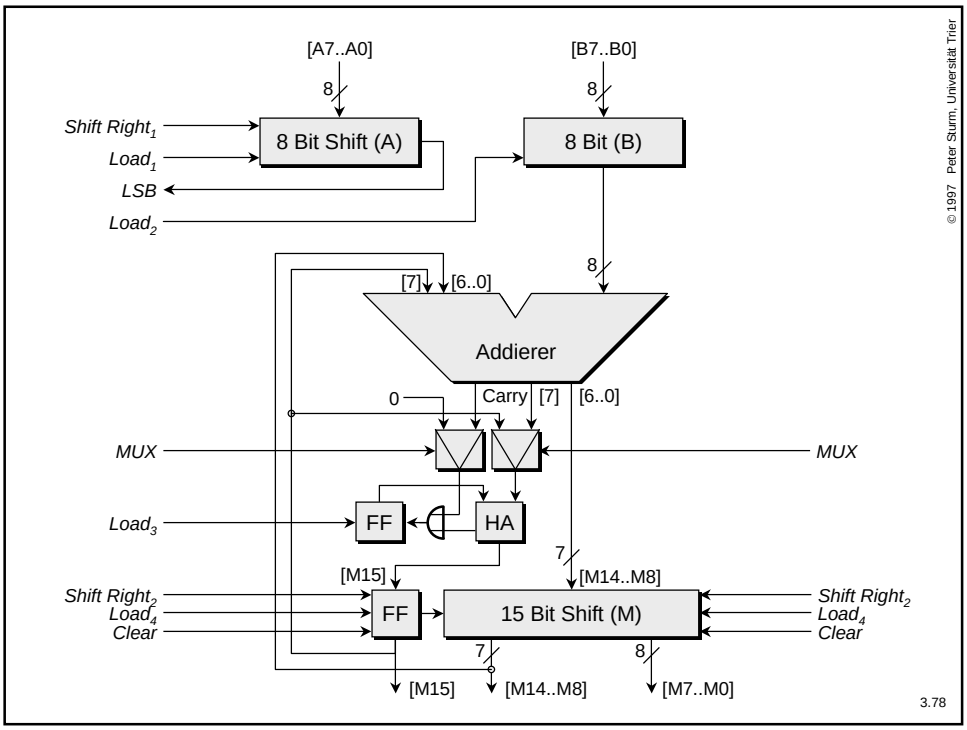
- Sequentielle Berechnung der partiellen Summen
- Steuerung der Abläufe
  - Steuerwerk
- Geringer Hardwareaufwand
- Höherer Zeitbedarf
- Bausteinvorrat:
  - 1 8-Bit Register
  - 1 8-Bit Schieberegister (parallel Ein- und Ausgabe)
  - 1 16-Bit Schieberegister (parallel Ein- und Ausgabe)
  - 1 8-Bit Addierer
  - 1 Flip-Flop
  - 1 1-Bit Halbaddierer

3.75



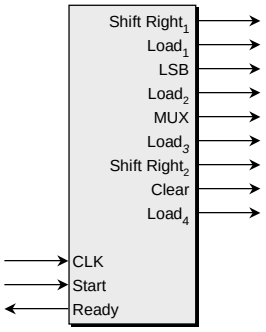
3.76


[A7..A0]
[B7..B0]
8
8
Shift Right1
Load1
8 Bit Shift (A)
8 Bit (B)
0
Load2
8
8
MUX
8
Addierer
Carry
[7]
[6..0]
Load3
FF
HA
7
[M15]
[M14..M8]
Shift Right2
Load4
Clear
16 Bit Shift (M)
8
8
[M15..M8]
[M7..M0]
© 1997 Peter Sturm, Universität Trier
3.77


[A7..A0]
[B7..B0]
8
8
Shift Right1
Load1
8 Bit Shift (A)
8 Bit (B)
LSB
Load2
8
[7]
[6..0]
Addierer
0
Carry
[7]
[6..0]
MUX
MUX
Load3
FF
HA
7
[M15]
[M14..M8]
Shift Right2
Load4
Clear
FF
15 Bit Shift (M)
Shift Right2
Load4
Clear
7
8
[M15]
[M14..M8]
[M7..M0]
© 1997 Peter Sturm, Universität Trier
3.78

TVMUL: Steuerwerk

- Übergeordnete Steuerung
  - Start (Eingang)
    - Argumente A und B liegen an
    - Multiplikation beginnen
  - Ready (Ausgang)
    - Resultat M verfügbar
  - CLK (Eingang)
    - Grundtakt
- Taktversorgung
  - Eigener Takt (intern)
  - Ein Grundtakt
    - Kleinster betrachteter Schritt
    - Kürzere Schritte nicht nutzbar
  - Mehrere Grundtakte
    - Komplizierte Versorgung
- Beschreibungs- und Realisierungstechniken?



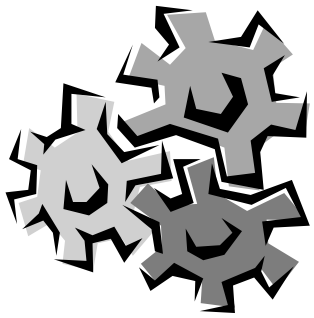
3.79

Zustandsautomaten

- Endliche Automaten

$$FSM = (\Theta, \sigma_0, f, E, A)$$
$$f : \Theta \times E \rightarrow \Theta \times A$$
$$(\sigma', a) = f(\sigma, e)$$

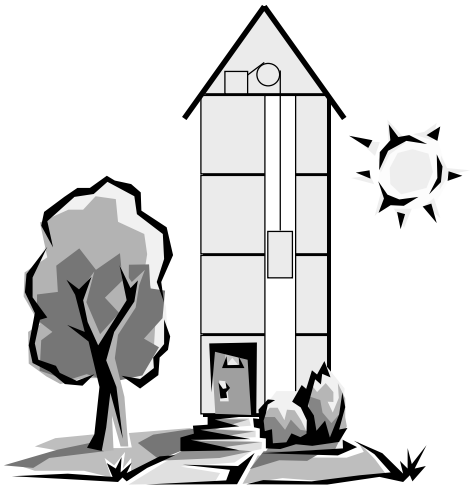
  - Endliche Zustandsmenge Θ
  - Ausgezeichneter Anfangszustand σ_0
  - Übergangsfunktion f
  - Eingabealphabet E
  - Ausgabealphabet A
- Zwei Varianten
  - Moore-Automaten
  - Mealy-Automaten



3.80

Zustandsdiagramme

- Graph
  - Knoten: Zustände
  - Kanten: Eingaben und Zustandsübergänge
- Ausgaben?
- Beispiel: Aufzugsteuerung
  - Eingaben
  - Ausgaben
  - Zustände
- Ziel: Komfortsteuerung?

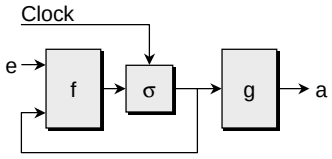


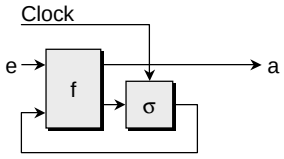
3.81

Aufzugsteuerung

3.82

Moore- und Mealy-Automaten





- Moore

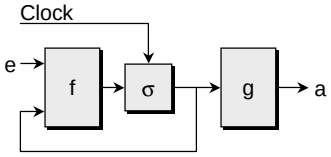
$$FSM = (\Theta, \sigma_0, f, g, E, A)$$
$$f: \Theta \times E \rightarrow \Theta, \quad \sigma' = f(\sigma, e)$$
$$g: \Theta \rightarrow A, \quad a = g(\sigma)$$

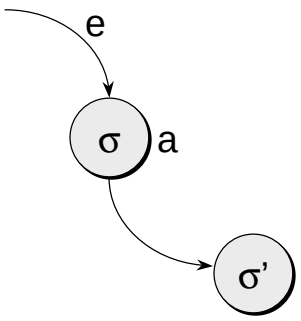
- Mealy

$$FSM = (\Theta, \sigma_0, f, E, A)$$
$$f: \Theta \times E \rightarrow \Theta \times A$$
$$(\sigma', a) = f(\sigma, e)$$

Moore-Automaten

- Synchrone Ausgabe
- Ausgaben werden nur durch den Zustand bestimmt
  - Ausgaben im Zustandsdiagramm neben den Zuständen



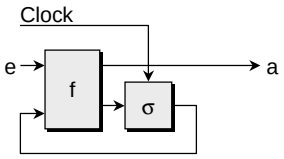


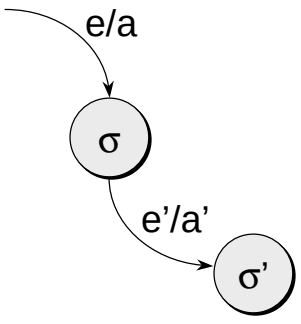
TVMUL: Moore-Zustandsdiagramm?

3.85

Mealy-Automaten

- Asynchrone Ausgabe
- Ausgaben werden durch Zustand und Eingaben bestimmt
  - Ausgaben im Zustandsdiagramm neben den Zustandsübergängen
- Benötigt häufig weniger Zustände als Moore-Automat
  - Warum?





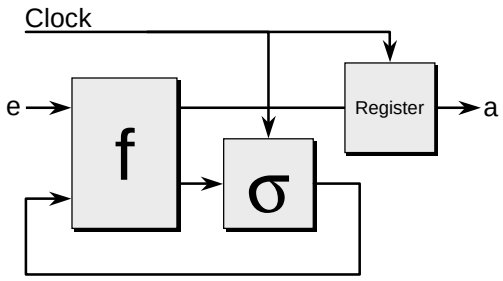
3.87

TVMUL: Mealy-Zustandsdiagramm?

3.88

Synchrone Mealy-Automaten

- Vorteil
  - Weniger Zustände im Vergleich zu Moore-Automaten
- Nachteil
  - Asynchrone Ausgaben
- Lösung
  - Taktung der Ausgabe



3.90

Übergangsfunktion

- Beschreibung durch Wahrheitstabelle
  - Mealy

$$(a, \sigma') = f(e, \sigma)$$

  - Moore

$$\sigma' = f(e, \sigma)$$
$$a = g(\sigma)$$

- Zustände minimieren
  - Hilfsautomaten
  - Systematisch
- Flip-Flop-Typ wählen
- Ansteuerungsfunktionen bestimmen und minimieren

3.91

Minimierung der Zustandsanzahl

3.92

TVMUL: Minimierung der WAIT-Zustände?

3.93

TVMUL: Zusammenfassung der Iterationszustände?

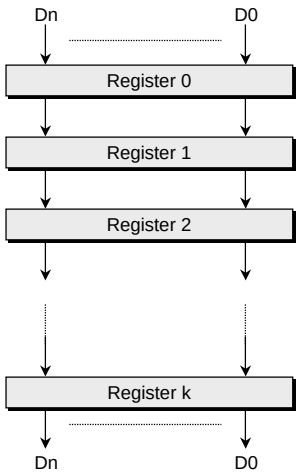
3.95

Zähler

3.97

FIFO-Speicher

- First-In First-Out
  - Nicht jede Speicherzelle direkt adressierbar
  - Kapazität = Tiefe
- Synchrone und asynchrone Variante
- Puffer zwischen unterschiedlich schnellen Komponenten
  - Sender kann schreiben, wenn Puffer nicht voll
  - Empfänger kann lesen, wenn Puffer nicht leer



3.98