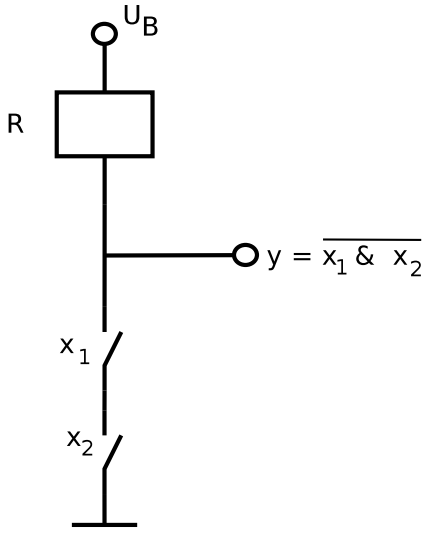
1 NAND-Schaltfunktion

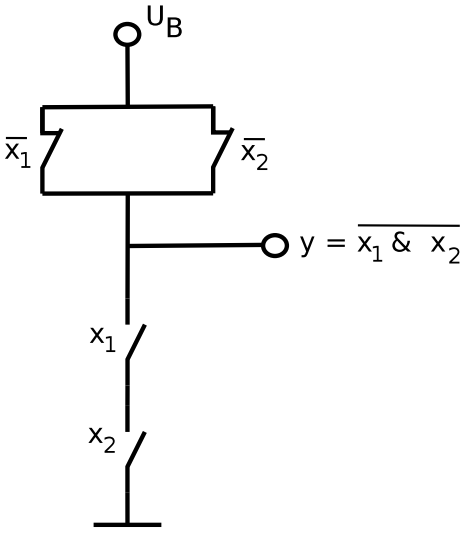
1.1 Einschalterprinzip

Die Idee des Einschalterprinzips liegt darin, dass, sobald der Schalter geschlossen ist, ein Strom von U_B zur Masse fließt.
Nachteil dieser Methode ist genau dies - es fließt Strom, was sowohl die Leistungsaufnahme als auch Abwärme einer Schaltung erhöht wird.

Diesen Nachteil versucht man durch das Zweischalterprinzip auszugleichen. Hier wird, sobald ein Schalter geschlossen wird, ein entsprechender Schalter im anderen Teil der Schaltung geöffnet.



(a) NAND im Einschalterprinzip



(b) NAND im Zweischalterprinzip

x_1	x_2	y
L	L	H
L	H	H
H	L	H
H	H	L

Tabelle 1: Spannungen

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

Tabelle 2: pos. Logik (NAND)

x_1	x_2	y
1	1	0
1	0	0
0	1	0
0	0	1

Tabelle 3: neg. Logik (NOR)

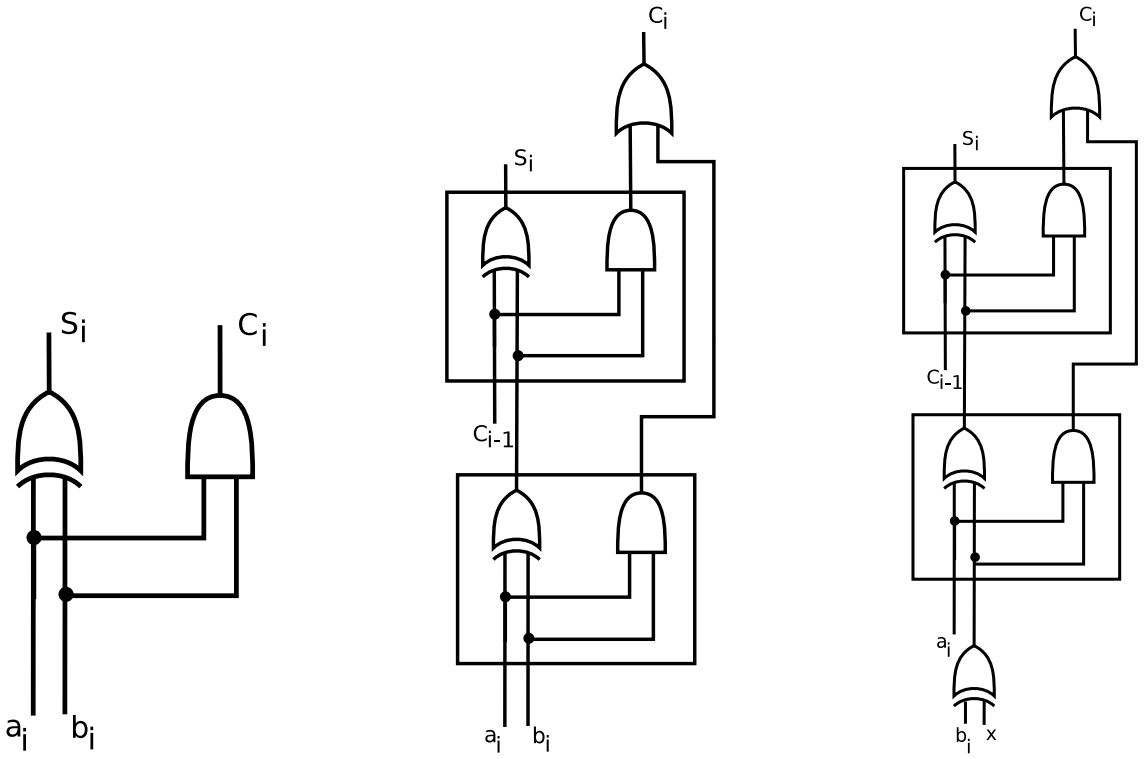
2 Entwurf eines 3-Bit-Addierers

2.1 Funktionstabelle

a_i	b_i	c_{i-1}	s_i	c_i
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Tabelle 4: Funktionstabelle: Volladdierer

2.2 Entwicklung des erweiterten Volladdierers



(c) Halbaddierer (d) Volladdierer (e) erweiterter Volladdierer

Abbildung 1: Der Weg vom Halbaddierer zum erweiterten Volladdierer

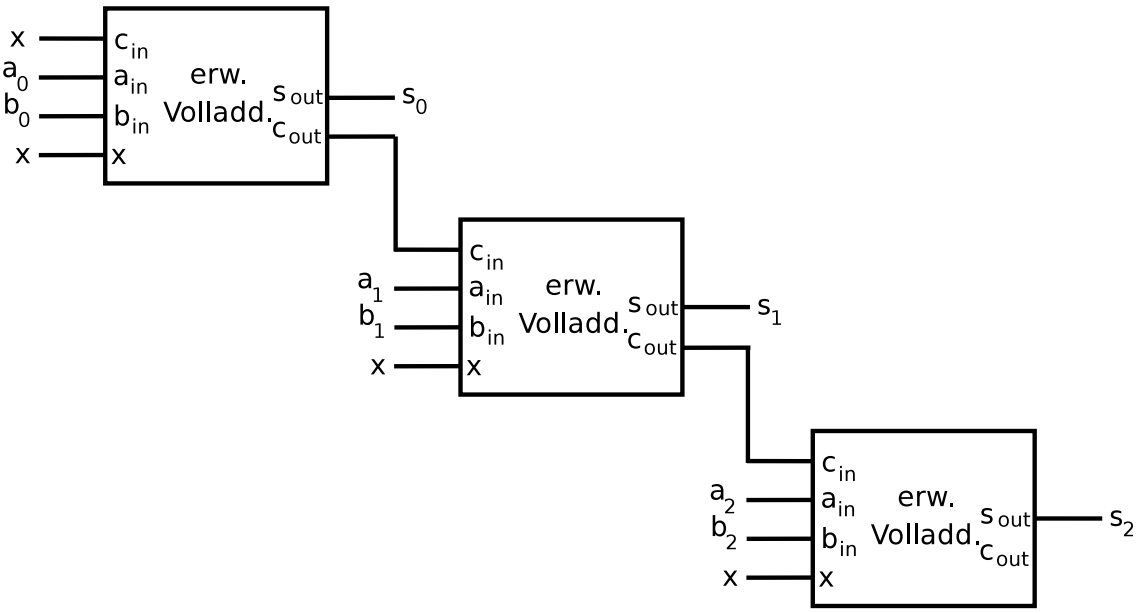
2.3 Verzögerung und maximaler Takt des Volladdierers

Nach 3 Gatterlaufzeiten (= 3τ ns) sind die Ausgänge stabil.
Für den maximalen Takt gilt dann:

$$f_{max} = \frac{1}{3\tau \text{ ns}}$$

2.4 Aufbau des 3-Bit-Addierers/Subtrahierers

Zum Aufbau einer Addier-/Subtrahierschaltung verwenden wir nun die erweiterte Form des Volladdierers. Der Carry-In des 1. Volladdierers wird auf x gesetzt.
x sei hier das Bit, das angibt ob addiert oder subtrahiert wird.
Der Carry-Out des xten Volladdierers wird an den Carry-In des x+1 ten Volladdierers angeschlossen.



3 Volladdierer mit Multiplexern

3.1 Volladdierer als Funktion

$$s_i = (a_i \oplus b_i) \oplus c_{i-1}$$
$$c_i = (a_i \cdot b_i) + (a_i \cdot c_{i-1}) + (b_i \cdot c_{i-1})$$

3.2 Volladdierer aus zwei 4:1-Multiplexern

$$\begin{aligned}
s_i &= (a_i \oplus b_i) \oplus c_{i-1} \\
&= a_i \cdot ((1 \oplus b_i) \oplus c_{i-1}) + \overline{a_i} \cdot ((0 \oplus b_i) \oplus c_{i-1}) \\
&= a_i \cdot (\overline{b_i} \oplus c_{i-1}) + \overline{a_i} \cdot (b_i \oplus c_{i-1}) \\
&= b_i \cdot (a_i \cdot (0 \oplus c_{i-1}) + \overline{a_i} \cdot (1 \oplus c_{i-1})) + \overline{b_i} \cdot (a_i \cdot (1 \oplus c_{i-1}) + \overline{a_i} \cdot (0 \oplus c_{i-1})) \\
&= b_i \cdot (a_i \cdot c_{i-1} + \overline{a_i} \cdot \overline{c_{i-1}}) + \overline{b_i} \cdot (a_i \cdot \overline{c_{i-1}} + \overline{a_i} \cdot c_{i-1})
\end{aligned}$$

$$\begin{aligned}
c_i &= (a_i \cdot b_i) + (a_i \cdot c_{i-1}) + (b_i \cdot c_{i-1}) \\
&= a_i \cdot ((1 \cdot b_i) + (1 \cdot c_{i-1}) + (b_i \cdot c_{i-1})) + \overline{a_i} \cdot ((0 \cdot b_i) + (0 \cdot c_{i-1}) + (b_i \cdot c_{i-1})) \\
&= a_i \cdot (b_i + c_{i-1}) + \overline{a_i} \cdot (b_i \cdot c_{i-1}) \\
&= b_i \cdot (a_i \cdot (1 + c_{i-1}) + \overline{a_i} \cdot (1 \cdot c_{i-1})) + \overline{b_i} \cdot (a_i \cdot (0 + c_{i-1}) + \overline{a_i} \cdot (0 \cdot c_{i-1})) \\
&= b_i \cdot (a_i + \overline{a_i} \cdot c_{i-1}) + \overline{b_i} \cdot (a_i \cdot c_{i-1})
\end{aligned}$$

Wie der Volladdierer, bestehend aus Multiplexern, nun aussieht ist in der Abbildung rechts zu sehen.
Die Beschriftung der Eingänge ist ist aus der Formel abzulesen.

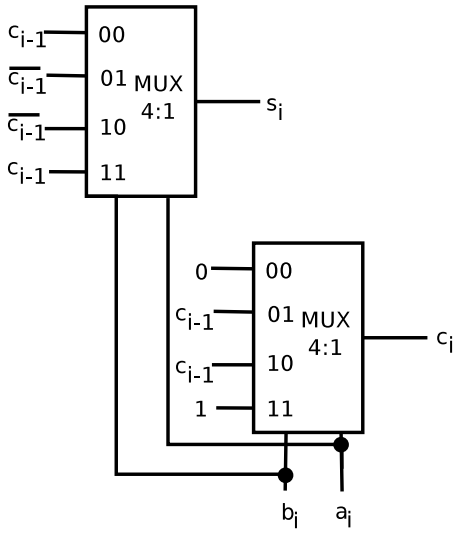


Abbildung 2: Volladdierer aus 4:1 Multiplexern

4 Flipflops

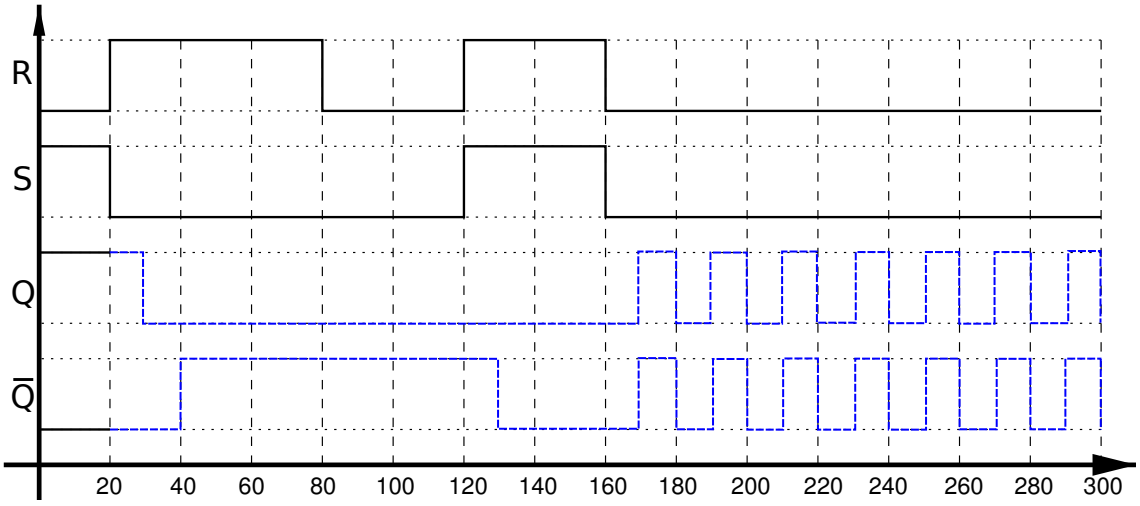


Abbildung 3: NOR-FlipFlop, beide Gatter 10ns Verzögerungszeit

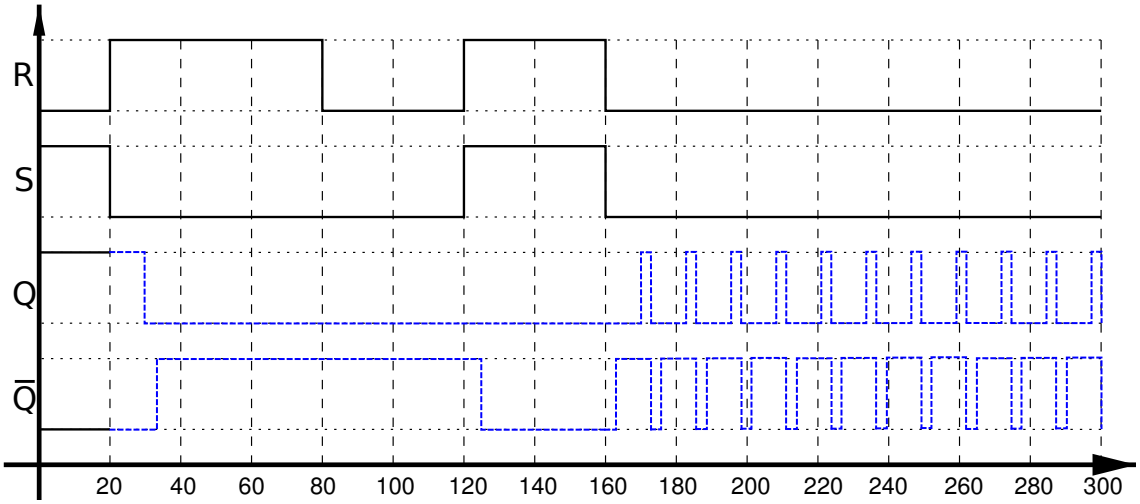


Abbildung 4: NOR-FlipFlop, Gatter 1 10ns, Gatter 2 3ns Verzögerungszeit

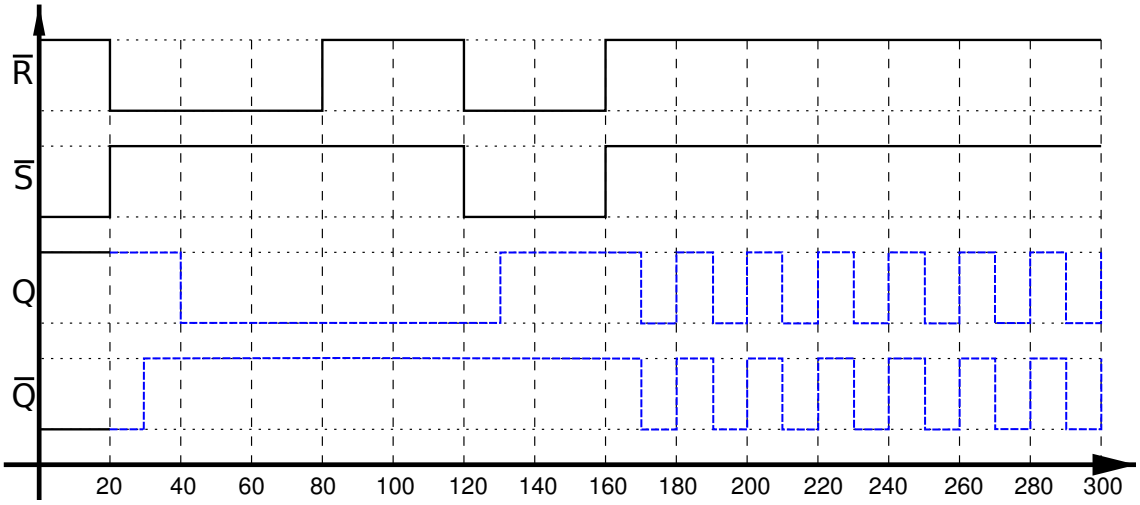


Abbildung 5: NAND-FlipFlop, beide Gatter 10ns Verzögerungszeit

5 Endliche Automaten

5.1 minimale Anzahl Zustände

Da von 0 bis 9 gezählt werden soll werden 10 Zustände benötigt.
$\rightarrow \lceil \log_2(10) \rceil = 4$ Bit benötigt.

5.2 Medwedew-Automat

Beim Medwedew-Automaten werden die Zustände direkt als Ausgabe verwendet.
Da die Anzeige im BCD - Format angesteuert werden soll, müssen die Zustände hier also als BCD-Code verwaltet werden.

5.3 Zustandsüberführungsfunktion

$\mathbf{D_iC_iB_iA_i}$	$\mathbf{T_3T_2T_1T_0}$	$\mathbf{D_{i+1}C_{i+1}B_{i+1}A_{i+1}}$
0000	0001	0001
0001	0011	0010
0010	0001	0011
0011	0101	0100
0100	0001	0101
0101	0011	0110
0110	0001	0111
0111	1111	1000
1000	0001	1001
1001	1001	0000

5.4 D-Flipflop

Der D-Flipflop übernimmt den an seinen D-Anschluss angelegten Wert 1:1 als seinen Zustand.
Andere Flipflop-Typen tuen dies nicht. T-Flipflops beispielsweise wechseln den Zustand bei angelegtem logischen 1.

5.5 Ansteuerung der Anzeige mit T-FlipFlops

$$T_0 = \overline{A}BC\overline{D} + A\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + AB\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + A\overline{B}\overline{C}\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + \overline{A}\overline{B}\overline{C}D + A\overline{B}\overline{C}D$$
$$T_1 = A\overline{B}C\overline{D} + AB\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + ABC\overline{D}$$
$$T_2 = AB\overline{C}\overline{D} + ABC\overline{D}$$
$$T_3 = ABC\overline{D} + A\overline{B}\overline{C}D$$

5.5.1 Minimieren von T_0 mittels Symmetriediagramm

In diesem Fall lässt sich die Tabelle zu einem großen Feld zusammenfassen.
$\rightarrow T_0 = 1$

B \ A				
	1_0	1_1	1_5	1_4
	1_2	1_3	1_7	1_6
	-10	-11	-15	-14
	1_8	1_9	-13	-12

C | D

Tabelle 5: Minimierung von T_0

5.5.2 Minimieren von T_1 mittels Symmetriediagramm

B \ A				
	0_0	1_1	1_5	0_4
	0_2	1_3	1_7	0_6
	-10	-11	-15	-14
	0_8	0_9	-13	-12

C | D

Tabelle 6: Minimierung von T_1

Auch in diesem Fall sieht man recht schnell was sich sinnvoll zusammenfassen lässt: Die 4 1en auf den Feldern 1 3 5 und 7 werden zu einer Insel zusammengefasst.
$\rightarrow T_1 = A\overline{D}$

5.5.3 Minimieren von T_2 mittels Verfahren von Nelson/Petrick

Für das Nelson-Verfahren benötigt man die KNF. Die don't cares werden zu 1en und fließen damit nicht in die KNF mit ein.

$$
\begin{aligned}
T_2 =& (A+B+C+D)\cdot(\overline{A}+B+C+D)\cdot(A+\overline{B}+C+D)\cdot(A+B+\overline{C}+D)\cdot \\
& (\overline{A}+B+\overline{C}+D)\cdot(A+\overline{B}+\overline{C}+D)\cdot(A+B+C+\overline{D})\cdot(\overline{A}+B+C+\overline{D})\cdot = \\
& (B+C+D)\cdot(A+\overline{B}+D)\cdot(B+\overline{C}+D)\cdot(B+C+\overline{D})\cdot = \\
& (B+C)\cdot(A+\overline{B}+D)\cdot(B+\overline{C}+D) = \\
& (AB+BD+AC+\overline{B}C+CD)(B+\overline{C}+D) = \\
& AB+BD+CD
\end{aligned}
$$

k	**PI**	**3**	**7**	p_i	c_i
0	AB	x	x	M	2
1	BD			N	3
2	CD			O	3

PA = MM = M
$\rightarrow T_2 = AB$

5.5.4 Minimieren von T_3 mittels Verfahren von Quine/McCluskey

DNF von T_3: $ABC\overline{D} + A\overline{B}C\overline{D} + \overline{A}B\overline{C}D + AB\overline{C}D + \overline{A}\overline{B}CD + A\overline{B}CD + \overline{A}BCD + ABCD$

Q_4 :

$Q_{4,4} = \{\}$

$Q_{4,3} = \{\}$

$Q_{4,2} = \{A\overline{B}\overline{C}D, \overline{A}B\overline{C}D, \overline{A}\overline{B}CD\}$

$Q_{4,1} = \{ABC\overline{D}, AB\overline{C}D, A\overline{B}CD, \overline{A}BCD\}$

$Q_{4,0} = \{ABCD\}$

Q_3 :

$Q_{3,3} = \{\}$

$Q_{3,2} = \{\}$

$Q_{3,1} = \{A\overline{C}D, A\overline{B}D, B\overline{C}D, \overline{A}BD, \overline{B}CD, \overline{A}CD\}$

$Q_{3,0} = \{\underline{ABC}, ABD, ACD, BCD\}$

Q_2 :

$Q_{2,2} = \{\}$

$Q_{2,1} = \{\}$

$Q_{2,0} = \{\underline{AD}, \underline{BD}, \underline{CD}\}$

	7	9
AD		x
BD		
CD		
ABC	x	

Da 7 und 9 gleichermaßen überdeckt sein müssen gilt für T_3:
$T_3 = AD + ABC$

DEC		**D**	**C**	**B**	**A**
0		0	0	0	0
1		0	0	0	1
2		0	0	1	0
3		0	0	1	1
4		0	1	0	0
5		0	1	0	1
6		0	1	1	0
7		0	1	1	1
8		1	0	0	0
9		1	0	0	1

Tabelle 7: Zahlen / BCD-Codierung