1.2.2 Zahlen

1.2.2.1 Dezimalziffern

Um die zehn Ziffern des Dezimalsystems binär zu codieren, braucht man (mindestens) 4 Bit; im *BCD-Code* oder *8-4-2-1-Code* wird nun jede Dezimalziffer durch die Bitkombination dargestellt, die im Stellenwertsystem der Basis 2 den entsprechenden Wert hat:

$$\sum_{i=0}^{3} m_i \cdot 2^i, \quad m_i \in \{0,1\}$$

In einem Byte kann man dann zwei Dezimalstellen, in 4 Byte (32 Bit) 8 Dezi-

Wert	m_3	m_2	m_1	m_0
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 1.22: Zifferndarstellung im BCD-Code

malstellen bzw. in 8 Byte (64 Bit) 16 Dezimalstellen unterbringen. Abgesehen davon, daß in 4 Byte im BCD-Code die Zahlen

$$00.000.000 \ldots 99.999.999,$$

im Dualsystem jedoch die (im Binärstellenwertsystem zu interpretierenden Zahlen)

$$0 \ldots \sum_{i=0}^{31} 1 \cdot 2^i,$$

also

$$0 \ldots 2^{32} - 1 = 0 \ldots 4.294.967.295$$

codiert werden können, ist auch das Rechnen mit BCD-codierten Zahlen aufwendiger: es benötigt mehr Schaltfunktionen / Gatter:

$$\begin{array}{rr r l}
 & 9_{10} & & 1001_2 \\
+ & 3_{10} & + & 0011_2 \\
\hline
 & 12_{10} & & 1100 \quad \leftarrow \text{kein BCD-Code, deshalb Korrektur um 6} \\
 & & + & 0110 \\
\hline
 & & 0001 & 0010
\end{array}$$

Ziffernaddition: 1 Addition, 1 Vergleich > 9 und ggf. 1 weitere Addition

Schließlich gibt es Binärmuster, die nicht als BCD-codierte Zahl auftreten können, die Abbildung ist also nicht bijektiv. Aus Gründen der besseren Ökonomie benutzen moderne Computer deshalb fast ausschließlich binär codierte Zahlen.

1.2.2.2 Ein Ausflug in die technische Informatik

Zu Informationen der Halbleiter-Physik vergleiche etwa:

`http://jas.eng.buffalo.edu/applets/education/fab/NMOS/nmos.html`,

`http://www.gel.ulaval.ca/ chabot01/MOSFET/English.html`,

`http://jas.eng.buffalo.edu/applets/education/index2.html`.

Seien $x_i, y_i, z_i, c_i, c_{i+1} \in \{0,1\}$. Für vorgegebene c_i, x_i, y_i ist dann

$$z_i = x_i \oplus y_i \oplus c_i$$

das Ergebnisbit und

$$c_{i+1} = x_i \cdot y_i \vee x_i \cdot c_i \vee y_i \cdot c_i$$

der Übertrag.

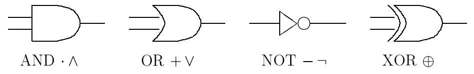


Abbildung 1.13: Gattersymbole

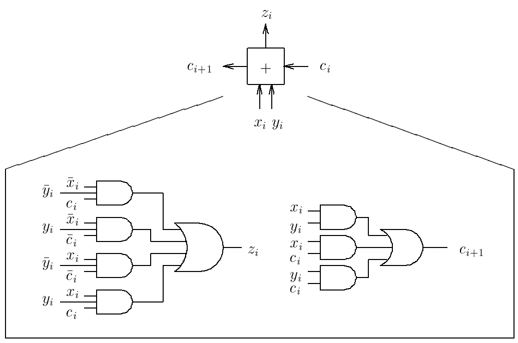


Abbildung 1.14: Der Volladdierer für 1 Bit (binär)

Allgemein gilt:

1. Jede Wertetabelle ist durch *vollständige* Beschaltung (ODER-Normalform) aller Wahrheitskombinationen realisierbar.
2. Es können durch Minimierung Gatter eingespart werden: KV-Diagramme (vgl. Übungen) oder Regeln der Schaltalgebra

Einen (experimentellen) Überblick zur Kombination von NOT-, AND- und OR-Gattern erhält man etwa mittels

`http://math.hws.edu/TMCM/java/xLogicCircuits/index.html`.

Bauen Sie etwa die obige Schaltung zur Übertragsberechung beim Einbit-Addierer mit Übertrageingang nach, und testen Sie.

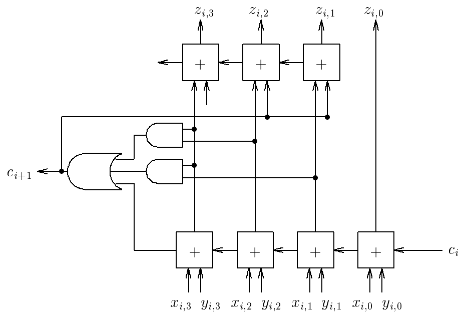
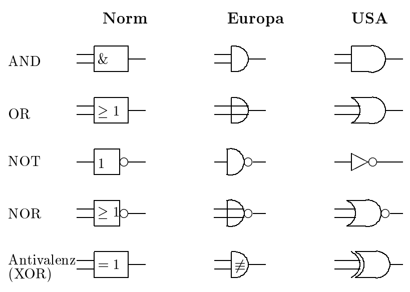


Abbildung 1.15: Der 4-Bit BCD-Addierer: es werden 7 statt 4 Volladdierern und 3 Gatter benötigt.

Der BCD-Addierer ist aufwendiger:

Gatter



und ihre Realisierungen: