

Einführung
in
Logische Schaltungen

Inhaltsverzeichnis

1. Einführung
 1. Was sind logische Schaltungen
 2. Grundlegende Elemente
 3. Weitere Elemente
 4. Beispiel einer logischen Schaltung
2. Notation von logischen Schaltungen
 1. Motivation
 2. Operationen und Rechenregeln
 3. Disjunktive und Konjunktive Normalform
 4. Optimierung

1. Einführung

1.1 Was sind logische Schaltungen?

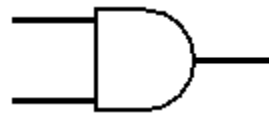
- Physikalisch realisiert durch Halbleiter (Transistoren, früher Röhren)
- Logische Schaltungen bilden die unterste Basis von Prozessoren, hier wird nur noch zwischen „Strom fließt“ und „Strom fließt nicht“ unterschieden. Darauf werden die einzelnen Logik-Ebenen bis hin zu höheren Programmiersprachen (Delphi, C, C++, ...) aufgebaut
- Anwendung in vielen digitalen Dingen, die uns im Alltag begegnen (Uhr, Fernbedienung, ...)

1.2 Grundlegende Elemente

Diese Elemente haben grundlegende Funktionalität in den logischen Schaltungen und sind zwingend notwendig, um komplexere Elemente zu konstruieren:

- AND

E1	E2	A
0	0	0
0	1	0
1	0	0
1	1	1



- OR

E1	E2	A
0	0	0
0	1	1
1	0	1
1	1	1



- NOT

E	A
0	1
1	0



1.3 Weitere Elemente

Elemente, die durch Kombination von Grundelementen erzeugt werden können (Aufbau → Logisim Beispiele):

- XOR

E1	E2	A
0	0	0
0	1	1
1	0	1
1	1	0



- NOR

E1	E2	A
0	0	1
0	1	0
1	0	0
1	1	0



- NAND

E1	E2	A
0	0	1
0	1	1
1	0	1
1	1	0



NAND kann als einziges grundlegendes Element alle anderen Elemente erzeugen! (→ Logisim Beispiele)

AND und OR können kaskadiert werden, d.h. man kann AND und OR-Gatter mit beliebig vielen Eingängen erstellen.

1.4 Beispiel einer logischen Schaltung

Eine allgegenwärtige Anwendung logischer Schaltungen befindet sich in Sieben-Segment-Anzeigen. Die meisten Sieben-Segment-Anzeigen müssen lediglich zwischen 10 verschiedenen Zeichen unterscheiden können: 0-9. Die kleinstmögliche Anzahl von Bits, die benötigt wird, um 10 Werte darzustellen, beträgt 4 ($2^3=8$, $2^4=16$), also benötigt man 4 Eingänge. Angesteuert werden muss jeder Balken der Anzeige einzeln, also 7 Ausgänge:

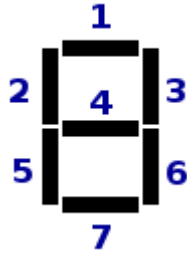


Tabelle mit Eingangs-/Ausgangswerten:

E1	E2	E3	E4	Z.	A1	A2	A3	A4	A5	A6	A7
0	0	0	0	0	1	1	1	0	1	1	1
0	0	0	1	1	0	0	1	0	0	1	0
0	0	1	0	2	1	0	1	1	1	0	1
0	0	1	1	3	1	0	1	1	0	1	1
0	1	0	0	4	0	1	1	1	0	1	0
0	1	0	1	5	1	1	0	1	0	1	1
0	1	1	0	6	1	1	0	1	1	1	1
0	1	1	1	7	1	0	1	0	0	1	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1
1	0	1	0	A	1	1	1	1	1	1	0
1	0	1	1	b	0	1	0	1	1	1	1
1	1	0	0	C	1	1	0	0	1	0	1
1	1	0	1	d	0	0	1	1	1	1	1
1	1	1	0	E	1	1	0	1	1	0	1
1	1	1	1	F	1	1	0	1	1	0	0

Balken A1 ist genau dann an, wenn $(E1=0, E2=0, E3=0, E4=0)$ oder $(E1=0, E2=0, E3=1, E4=0)$ oder $(E1=0, E2=0, E3=1, E4=1)$ oder $(E1=0, E2=1, E3=0, E4=1)$ oder $(E1=0, E2=1, E3=1, E4=0)$ oder $(E1=0, E2=1, E3=1, E4=1)$ oder $(E1=1, E2=0, E3=0, E4=0)$ oder $(E1=1, E2=0, E3=0, E4=1)$ oder $(E1=1, E2=0, E3=1, E4=0)$ oder $(E1=1, E2=1, E3=0, E4=0)$ oder $(E1=1, E2=1, E3=1, E4=0)$ oder $(E1=1, E2=1, E3=1, E4=1)$

Wie man sieht, ist diese Art der Erstellung von Schaltungen sehr umfangreich und unkomfortabel, daher stellt sich die Frage, ob man das nicht optimieren kann.

Bei A1 treten zum Beispiel folgende Terme auf: „(E1=0, E2=0, E3=1, E4=0) oder (E1=0, E2=0, E3=1, E4=1)“. Da E4 nur die Werte 0 und 1 annehmen kann, ist es hier offensichtlich egal, welchen Wert E4 hat. Man könnte also diese beiden Terme zu (E1=0, E2=0, E3=1) zusammenfassen.

Gibt es eine systematische Methode, solche Zusammenhänge zu erkennen?

2. Notation von Logischen Schaltungen

2.1 Motivation

Wie man feststellt, ist die Formulierung von logischen Zuständen durch z. B. „(E1=0, E2=0, E3=1)“ eine langwierige Schreibarbeit. Daher sucht man nach geeigneten Schreibweisen, um eindeutig Zustände auszudrücken.

Der Zustand eines Ausgangs wird durch Konjunktion und Disjunktion beschrieben. e_1 steht dabei für E1=1, \acute{e}_2 steht dabei für E2=0 (normalerweise wird ein negierter Eingang durch einen Querstrich über dem Symbol gekennzeichnet, allerdings wird das hier so vereinfacht dargestellt).

2.2 Operationen und Rechenregeln

- $e_1 + e_2 = A$ (OR)
- $e_1 * e_2 = A$ (AND)
- $0+0 = 0; 0+1 = 1+0 = 1; \underline{1+1 = 1}$
- $0*0 = 0; 0*1 = 1*0 = 0; 1*1 = 1$

2.3 Disjunktive Normalform und Konjunktive Normalform

Das Wort „Konjunktiv“ entspricht dem und (AND)

Das Wort „Disjunktiv“ entspricht dem oder (OR)

Die Disjunktive Normalform ist eine Disjunktion von Konjunktiv-Termen, d. h. man verbindet die einzelnen Terme intern mit einem AND und alle Terme zusammen mit einem OR. Beispiel eines Terms in Disjunktiver Normalform anhand des Ausgangs A1 (man sucht nach 1):

$$A1 = \acute{e}_1\acute{e}_2\acute{e}_3\acute{e}_4 + \acute{e}_1\acute{e}_2e_3\acute{e}_4 + \acute{e}_1\acute{e}_2e_3e_4 + \acute{e}_1e_2\acute{e}_3\acute{e}_4 + \acute{e}_1e_2e_3\acute{e}_4 + \acute{e}_1e_2e_3e_4 + e_1\acute{e}_2\acute{e}_3\acute{e}_4 + e_1\acute{e}_2\acute{e}_3e_4 + e_1\acute{e}_2e_3\acute{e}_4 + e_1\acute{e}_2e_3e_4 + e_1e_2\acute{e}_3\acute{e}_4$$

Beispiel eines Terms in Konjunktiver Normalform anhand des Ausgangs A1 (man sucht nach 0):

$$\acute{A}1 = (e_1+e_2+e_3+e_4) * (e_1+e_2+\acute{e}_3+e_4) * (e_1+e_2+\acute{e}_3+\acute{e}_4) * (e_1+\acute{e}_2+e_3+\acute{e}_4) * (e_1+\acute{e}_2+\acute{e}_3+e_4) * (e_1+\acute{e}_2+\acute{e}_3+\acute{e}_4) * (\acute{e}_1+e_2+e_3+e_4) * (\acute{e}_1+e_2+e_3+\acute{e}_4) * (\acute{e}_1+e_2+\acute{e}_3+e_4) * (\acute{e}_1+\acute{e}_2+e_3+e_4) * (\acute{e}_1+\acute{e}_2+\acute{e}_3+e_4) * (\acute{e}_1+\acute{e}_2+\acute{e}_3+\acute{e}_4)$$

2.4 Optimierung

Man kann Schaltungen optimieren und damit die Anzahl der benötigten Gatter reduzieren. Wir betrachten hier nur die Anzahl von vier Eingängen.

Man erstellt eine Tabelle, die jede Kombination von an/ausgeschalteten Eingängen zulässt (Beispiel für A1). Die Position ergibt sich aus dem Zustand der Eingänge, der Wert aus dem des Ausgangs:

	e_1				
	1	0	1	1	
e_2	1	1	0	1	e_4
	1	1	1	1	
	0	1	0	1	
	0	1	0	1	
	e_3				

Man sucht nun Blöcke von 2 oder 4 Einsen. Dabei versucht man, möglichst wenig Blöcke zu bilden und alle Einsen müssen abgedeckt sein. Den Wert von A1 bestimmt man nun mit Formeln für alle eingezeichneten Blöcke:

$$A1 = \acute{e}_1\acute{e}_3 + \acute{e}_2\acute{e}_4 + \acute{e}_1\acute{e}_2\acute{e}_4 + e_1e_2e_3 + e_1\acute{e}_3\acute{e}_4 + e_1\acute{e}_2e_3$$

Um eine Optimierung in der Disjunktiven Normalform vorzunehmen (dann optimaler, wenn vielen Nullen in der Tabelle vorhanden sind) sucht man Blöcke von Nullen.