

Logische Schaltungen & PLAs

Benjamin Tröster

Hochschule für Technik und Wirtschaft Berlin

26. Januar 2022

Fahrplan

Recap: Darstellung Logikgatter

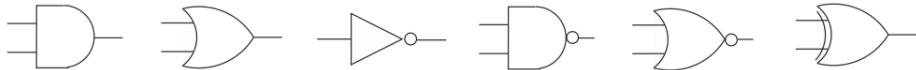
Logische Bausteine

Recap Normalformdarstellungen

Programmable Logic Array (PLA)

Recap: Darstellung Logikgatter

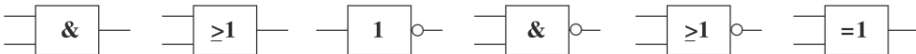
Amerikanische Symbole



Europäische Symbole



Deutsche Symbole (DIN)



AND-Gatter

OR-Gatter

NOT-Gatter
(Inverter)

NAND-Gatter

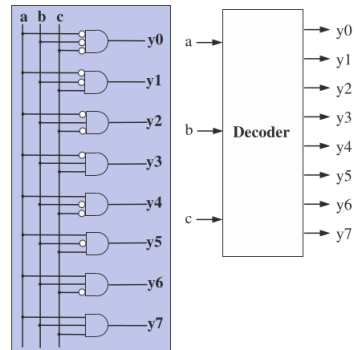
NOR-Gatter

XOR-Gatter

Abbildung: Übernommen aus: [Hof20]

Decoder

- ▶ Decoder hat n Eingänge und 2^n Ausgänge (bzw. $\leq 2^n$ Ausgänge)
- ▶ Für jede Eingabekombination genau einen Ausgang der 1 ergibt
- ▶ Alle anderen Ausgänge sind 0
 - ▶ Wir codieren „Pattern“ von Eingangsbits auf Ausgabebits
- ▶ Beispiel: 3 – to – 8-Decoder
 - ▶ Ausgang y_i auf 1, alle anderen Ausgänge 0
 - ▶ Welcher Ausgang y_i auf 1 gesetzt wird, entscheiden die Eingänge a, b, c
 - ▶ Eingänge a, b und c stellen entsprechende Dualzahl dar
- ▶ Nutzung z.B. ROMs



| Eingänge | | | Ausgänge | | | | | | | |
|----------|-----|-----|----------|-------|-------|-------|-------|-------|-------|-------|
| a | b | c | y_0 | y_1 | y_2 | y_3 | y_4 | y_5 | y_6 | y_7 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Abbildung: 3-to-8 Decoder Schaltung & Wahrheitstabelle.

3-to-8-Decoder Beispiel in C

```
1 int main(void) {  
2     int a, b, c;  
3     printf("Enter encoded binary number: ");  
4     a = getchar() - '0';  
5     b = getchar() - '0';  
6     c = getchar() - '0';  
7     if (!a && !b && !c) printf("----> y0\n");  
8     if (!a && !b && c) printf("----> y1\n");  
9     if (!a && b && !c) printf("----> y2\n");  
10    if (!a && b && c) printf("----> y3\n");  
11    if ( a && !b && !c) printf("----> y4\n");  
12    if ( a && !b && c) printf("----> y5\n");  
13    if ( a && b && !c) printf("----> y6\n");  
14    if ( a && b && c) printf("----> y7\n");  
15    return 0;  
16 }
```

Encoder

- ▶ Analog: Encoder inverse Funktion zum Decoder
- ▶ Encoder hat 2^n Eingänge, von denen genau einer wahr sein sollte
- ▶ Ausgabe von n Bits
- ▶ Beispiel: 8 – to – 3-Encoder
 - ▶ Eingänge x_0, x_1, \dots, x_7 auf Codierung in Dual an den Ausgängen d_2, d_1, d_0

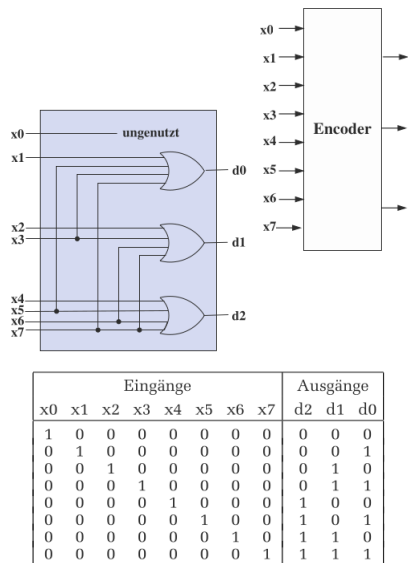


Abbildung: 8-to-3 Encoder Schaltung & Wahrheitstabelle.

8-to-3-Encoder Beispiel in C

```
1 int main(void) {
2     int x0, x1, x2, x3, x4, x5, x6, x7, d0=0, d1=0, d2=0;
3     printf("Enter 8 Bit Binary Number: ");
4     x0 = getchar() - '0';
5     x1 = getchar() - '0';
6     x2 = getchar() - '0';
7     x3 = getchar() - '0';
8     x4 = getchar() - '0';
9     x5 = getchar() - '0';
10    x6 = getchar() - '0';
11    x7 = getchar() - '0';
12    d0 = x1 || x3 || x5 || x7;
13    d1 = x2 || x3 || x6 || x7;
14    d2 = x4 || x5 || x6 || x7;
15    printf("----> %d%d%d (d2, d1, d0)\n", d2, d1, d0);
16    return 0;
```

Multiplexer (Selektor)

- ▶ Multiplexer werden oft auch als Selektoren bezeichnet, da sie unter den Eingangssignalen eines auswählen
- ▶ Multiplexer führt Datenpfade zusammen
- ▶ Multiplexer: Mehrere Eingänge und einen Ausgang
 - ▶ Wobei dieser einem der Eingänge entspricht, der durch eine Steuerung ausgewählt wird

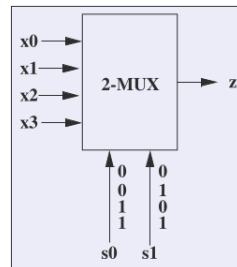
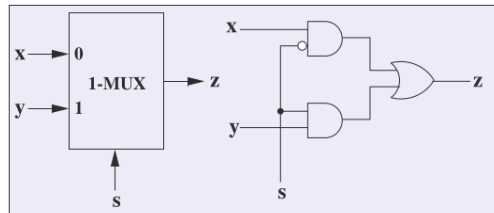


Abbildung: Multiplexer mit zugehöriger Schaltung und ein 2-Multiplexer.

1-Multiplexers

- ▶ 1-Multiplexers als boolesche Funktion: $z = \bar{s}x \vee sy$
- ▶ Was folgender Tabelle entspricht
- ▶ Multiplexer: Mehrere Eingänge und einen Ausgang

| s | $\bar{s}x$ | sy |
|-----|------------|------|
| 0 | x | 0 |
| 1 | 0 | y |

- ▶ Bei $s = 0$ wird also x weitergeleitet
- ▶ Bei $s = 1$ wird y zum Ausgang weitergeleitet.

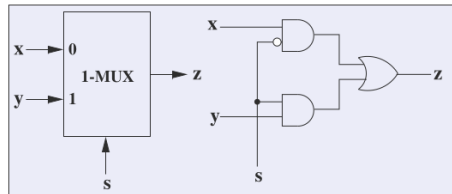


Abbildung: Multiplexer mit zugehöriger Schaltung.

2-Multiplexers

- ▶ Beim 2-Multiplexer sind zwei Steuereingänge vorhanden, also vier Wahlmöglichkeiten
- ▶ Es ergeben sich folgenden Auswahlmöglichkeiten:

| $s : 0$ | s_1 | z Ausgang |
|---------|-------|--------------------|
| 0 | 0 | x_0 |
| 0 | 1 | x_1 |
| 1 | 0 | x_2 |
| 1 | 1 | x_3 |

- ▶ Aus dieser Tabelle lässt sich dann die folgende Schaltfunktion herleiten:

$$z = s_0 s_1 x_0 \vee s_0 s_1 x_1 \vee s_0 s_1 x_2 \vee s_0 s_1 x_3$$

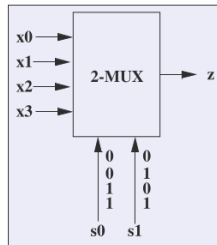


Abbildung: Multiplexer mit zugehöriger Schaltung und ein 2-Multiplexer.

2-Multiplexer-Realisierung: Bottom-Up

► Schaltfunktion:

$z = s_0 s_1 x_0 \vee s_0 s_1 x_1 \vee s_0 s_1 x_2 \vee s_0 s_1 x_3$ bezeichnet als Bottom-Up

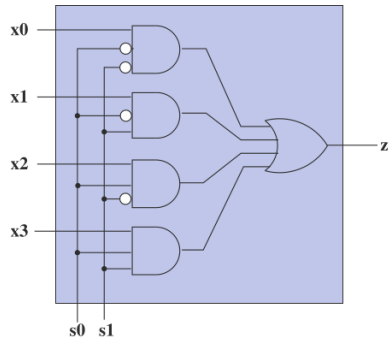


Abbildung: Realisierung eines Bottom-Up 2-Multiplexer.

2-Multiplexer-Realisierung: Top-Down

- ▶ Top-Down-Ansatz mithilfe von 1-Multiplexern
- ▶ Folgendes gilt:
 1. Ausgang von 1 – MUX_z : $z = s_0 a + s_0 b$
 2. Ausgang von 1 – MUX_a : $a = s_1 x_0 + s_1 x_1$
 3. Ausgang von 1 – MUX_b : $b = s_1 x_2 + s_1 x_3$
- ▶ Einsetzen von 2, 3 in 1 ergibt:
$$z = s_0 s_1 x_0 \vee s_0 s_1 x_1 \vee s_0 s_1 x_2 \vee s_0 s_1 x_3$$
- ▶ Anmerkung: Top-Down braucht mehr Gatter (Preis), mehr Platz und langsamer also Bottom-Up

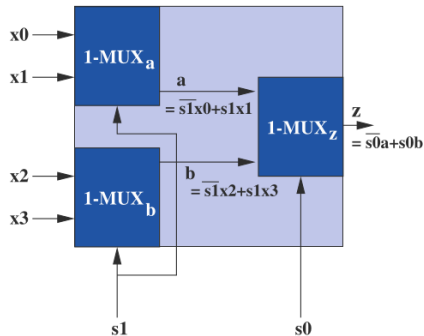


Abbildung: Realisierung eines Top-Down 2-Multiplexer.

n -Multiplexer

- ▶ Multiplexer mit beliebigen Anzahl von Eingaben realisierbar
- ▶ n Eingangssignalen werden $\log_2 n$ Selektoreingabe benötigt
- ▶ Dreiteiliger Aufbau:
 1. Decoder: aus $\log_2 n$ Selektoreingaben n Signale erzeugt, die jeweils einen anderen Eingabewert auswählen,
 2. n AND-Gattern: Kombination jeweils eines Signals des Decoder mit einem Eingangssignal
 3. OR-Gatter: n Eingängen (bzw. $n - 1$ hintereinander geschaltete OR- Gatter zwei Eingängen), das die Ausgaben der AND-Gatter verknüpft.

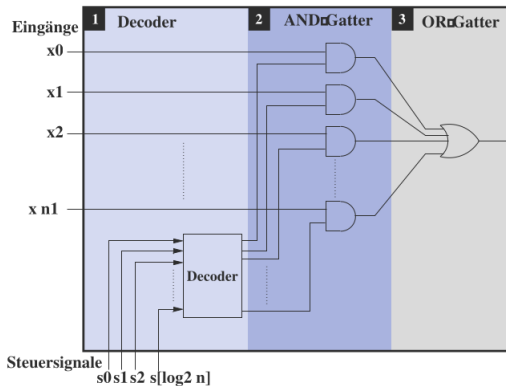


Abbildung: Beispiel für einen n -Multiplexer.

Demultiplexer

- ▶ Während für einen Multiplexer Folgendes gilt:
 - ▶ 2^d Eingänge ($x_0, x_1, \dots, x_{2^d-1}$)
 - ▶ d Steuersignale (s_0, s_1, \dots, s_{d-1}) und
 - ▶ ein Ausgang z mit $z = \sum_{i=0}^{2^d-1} x_i \cdot s_0 s_1 \dots s_{d-1}$
- ▶ gilt für einen Demultiplexer Folgendes:
 - ▶ ein Dateneingang x ,
 - ▶ d Steuersignale (s_0, s_1, \dots, s_{d-1}) und
 - ▶ 2^d Ausgänge ($z_0, z_1, \dots, z_{2^d-1}$) mit $z_i = x \cdot s_0 s_1 \dots s_{d-1}$
- ▶ Demultiplexer: Steuersignale legen fest auf welchen Ausgang das Eingangssignal gelegt wird

Demultiplexer

- ▶ Während für einen Multiplexer Folgendes gilt:
 - ▶ 2^d Eingänge ($x_0, x_1, \dots, x_{2^d-1}$)
 - ▶ d Steuersignale (s_0, s_1, \dots, s_{d-1}) und
 - ▶ ein Ausgang z mit
$$z = \sum_{i=0}^{2^d-1} x_i \cdot s_0 s_1 \dots s_{d-1}$$
- ▶ gilt für einen Demultiplexer Folgendes:
 - ▶ ein Dateneingang x ,
 - ▶ d Steuersignale (s_0, s_1, \dots, s_{d-1}) und
 - ▶ 2^d Ausgänge ($z_0, z_1, \dots, z_{2^d-1}$) mit
$$z_i = x \cdot s_0 s_1 \dots s_{d-1}$$
- ▶ Demultiplexer: Steuersignale legen fest auf welchen Ausgang das Eingangssignal gelegt wird

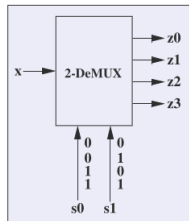
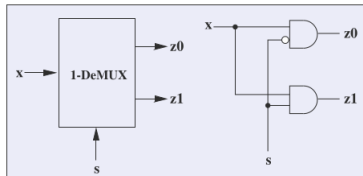


Abbildung: 1-Demultiplexer mit zugehöriger Schaltung und ein 2-Demultiplexer

1-Demultiplexer

- ▶ x steht dabei für den Eingabewert und s für einen Selektor – d.h. einen Steuerwert (control value)
- ▶ Steuerwert bestimmt, zu welchem der Ausgabewerte der Eingabewert weitergeleitet wird
- ▶ Booleschen Funktionen: $z_0 = x\bar{s}$ und $z_1 = xs$
- ▶ Entspricht folgender Wahrheitstabelle:

| s | x | Auswahl | Schaltfunktion |
|-----|-----|---------|------------------|
| 0 | x | z_0 | $z_0 = x\bar{s}$ |
| 1 | x | z_1 | $z_1 = xs$ |

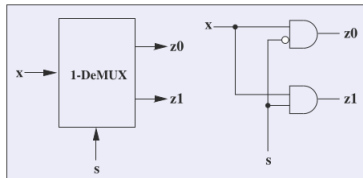


Abbildung: 1-Demultiplexer mit zugehöriger Schaltung.

2-Demultiplexer

- ▶ 2-Demultiplexer sind zwei Steuereingänge vorhanden → vier der Ausgabesignale auszuwählen
- ▶ Es ergeben sich
- ▶ Entspricht folgender Wahrheitstabelle:

| s_0 | s_1 | Auswahl | Schaltfunktion |
|-------|-------|---------|---------------------------------------|
| 0 | 0 | z_0 | $z_0 = x\overline{s_0}\overline{s_1}$ |
| 0 | 1 | z_1 | $z_1 = x\overline{s_0}s_1$ |
| 1 | 0 | z_2 | $z_2 = xs_0\overline{s_1}$ |
| 1 | 1 | z_3 | $z_3 = xs_0s_1$ |

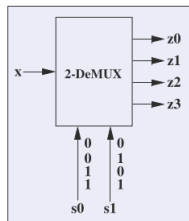


Abbildung: 2-Demultiplexer mit zugehöriger Schaltung.

2-Demultiplexer-Realisierung: Bottom-Up

- Direkte Realisierung als Schaltung
 z_0, z_1, z_2, z_3 in parallel

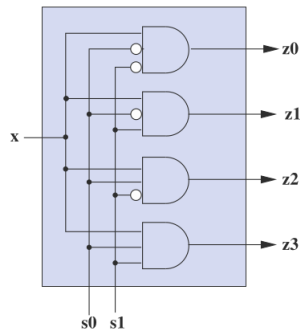


Abbildung: 1-Demultiplexer mit zugehöriger Schaltung.

2-Demultiplexer-Realisierung: Top-Down

- Realisierung in Top-Down unter Verwendung von 1-Demultiplexern
- Folgende Gleichungen gelten:

$$z_0 = a \overline{s_1} \quad \rightarrow \quad z_0 = x \overline{s_0} \overline{s_1}$$

$$z_1 = a s_1 \quad \rightarrow \quad z_1 = x \overline{s_0} s_1$$

$$z_2 = b \overline{s_1} \quad \rightarrow \quad z_2 = x s_0 \overline{s_1}$$

$$z_3 = b s_1 \quad \rightarrow \quad z_3 = x s_0 s_1$$

- Anmerkung: Top-Down braucht mehr Gatter (Preis), mehr Platz und langsamer als Bottom-Up

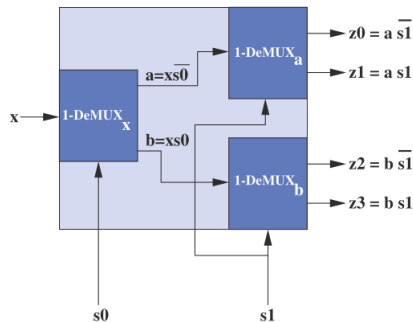


Abbildung: 1-Demultiplexer mit zugehöriger Schaltung.

Recap Normalformdarstellungen

- ▶ Normalform beschreibt eine eindeutige Darstellung
- ▶ Vollform: Ausdruck, in dem jede Variable genau einmal vorkommt
- ▶ Literal: Teilausdruck, der entweder negierte oder unnegierte Variable darstellt
- ▶ Wahrheitstafeldarstellung ist eine Art der Normalformdarstellungen
- ▶ Bool'sche Ausdrücke hingegen sind keine Normalformdarstellung
 - ▶ Jede bool'sche Funktion durch unendlich viele Ausdrücke beschrieben werden

Normalformdarstellungen

- ▶ Vollform: Ausdruck, in dem jede Variable genau einmal vorkommt
- ▶ Vollkonjunktion (**Minterm**): Ausdruck, in dem sämtliche vereinbarten Variablen (bzw. deren Negate) konjunktiv verbunden sind
 - ▶ Beispiel: $A, B, C : A \wedge \neg B \wedge C$
- ▶ Volldisjunktion (**Maxterm**): Ausdruck, in dem sämtliche vereinbarten Variablen (bzw. deren Negate) disjunktiv verbunden sind
 - ▶ Beispiel: $A, B, C : A \vee \neg B \vee \neg C$
- ▶ Negationen nur in atomarer Form
 - ▶ $\neg(A \wedge B)$: nicht atomar
 - ▶ $(\neg A \vee \neg B)$: atomar

Formale Definition

Definition (Minterm, Maxterm, Literal)

Sei $f(x_1, \dots, x_n)$ eine beliebige n -stellige boolesche Funktion. Jeder Ausdruck der Form

$$\hat{x}_1 \wedge \dots \wedge \hat{x}_n \quad \text{mit } \hat{x}_i \in \{\overline{x_i}, x_i\}$$

heißt **Minterm**, jeder Ausdruck der Form

$$\hat{x}_1 \vee \dots \vee \hat{x}_n \quad \text{mit } \hat{x}_i \in \{\overline{x_i}, x_i\}$$

wird **Maxterm** genannt.

Der Teilausdruck \hat{x}_i , der entweder aus einer negierten oder einer unnegierten Variablen besteht, heißt **Literal**.

Disjunktive Normalform

- ▶ Die disjunktive Normalform (DNF) ist jene Darstellungsart, bei der eine Reihe von Vollkonjunktionen disjunktiv verknüpft wird. Negationen treten nur in atomarer Form auf.
 - ▶ $(A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge C)$
- ▶ Andere Bezeichnungen:
 - ▶ Kanonische disjunktive/konjunktive Normalform (KDNF/KKNF)
 - ▶ Vollständige disjunktive/konjunktive Normalform

Beispiel: Disjunktive Normalform

$$f(x_1, x_2, x_3) = (x_1 \Rightarrow x_2) \wedge (\neg x_1 \Leftrightarrow x_3)$$

| | x_1 | x_2 | x_3 | $x_1 \Rightarrow x_2$ | $\neg x_1 \Leftrightarrow x_3$ | $(x_1 \Rightarrow x_2) \wedge (\neg x_1 \Leftrightarrow x_3)$ |
|---|-------|-------|-------|-----------------------|--------------------------------|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| 3 | 0 | 1 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 0 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 0 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 0 | 0 |

Vollkonjunktion/Minterm: 2: $(\neg x_1 \wedge \neg x_2 \wedge x_3)$, 4: $(\neg x_1 \wedge x_2 \wedge x_3)$, 7: $(x_1 \wedge x_2 \wedge \neg x_3)$

DNF: $(\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3)$

Konjunktive Normalform

- ▶ Die konjunktive Normalform (KNF) ist jene Darstellungsart, bei der eine Reihe von Volldisjunktionen konjunktiv verknüpft wird. Negationen treten nur in atomarer Form auf.
 - ▶ $(\neg A \vee \neg B \vee \neg C) \wedge (A \vee B \vee C) \wedge (A \vee \neg B \vee \neg C)$
- ▶ Andere Bezeichnungen:
 - ▶ Kanonische disjunktive/konjunktive Normalform (KDNF/KKNF)
 - ▶ Vollständige disjunktive/konjunktive Normalform

Beispiel: Konjunktive Normalform

$$f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee x_3$$

| | x_1 | x_2 | x_3 | $x_1 \wedge x_2$ | $(x_1 \wedge x_2) \vee x_3$ |
|---|-------|-------|-------|------------------|-----------------------------|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 6 | 1 | 0 | 1 | 0 | 1 |
| 7 | 1 | 1 | 0 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 |

Vollkonjunktion/Minterm: 1: $\neg(\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$, 3: $\neg(\neg x_1 \wedge x_2 \wedge \neg x_3)$, 5: $\neg(x_1 \wedge \neg x_2 \wedge \neg x_3)$

Volldisjunktion/Maxterm: 1: $(x_1 \vee x_2 \vee x_3)$, 3: $(x_1 \vee \neg x_2 \vee x_3)$, 5: $(\neg x_1 \vee x_2 \vee x_3)$

KNF: $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

Allgemeines Verfahren beim Erstellen einer Schaltung

- ▶ Zusammengefasst gilt folgendes Verfahren beim Erstellen einer Schaltung:
 1. Aufstellen der Wahrheitstabelle zur gesuchten Schaltung
 2. Beim Herleiten einer Normalform zwei Möglichkeiten:
 - ▶ Disjunktive Normalform: Vollkonjunktion bilden alle Zeilen denen 1 zugeordnet ist, mit 0 belegte Variablen negieren. Disjunktive Verknüpfung der Vollkonjunktionen.
 - ▶ Konjunktive Normalform: Bilden der Volldisjunktion, Wahrheitstabelle dessen Zeilen 0 zugeordnet ist, Variablen mit 1 belegt werden negiert. Diese Volldisjunktionen werden dann konjunktiv verknüpft.
 3. Minimierungsversuch mittels Äquivalenzumformungen via booleschen Algebra.

Programmable Logic Array (PLA)

- ▶ Form der programmierbaren logischen Schaltung
 - ▶ „Logisches Programmieren in Hardware“
- ▶ PLA hat eine Menge von Inputs als Eingabe und zwei Stufen von Logiken
 - ▶ ein Feld von ANDs
 - ▶ Generiert eine Menge von Produkten (Konjunktionen)
 - ▶ Auswahl der Konjunktionsterme durch Entfernen von Schaltgliedern (aus der UND-Matrix)
 - ▶ Ein Feld von ORs
 - ▶ Disjunktive Verknüpfung der Konjunktionsterme erfolgt mittels der ODER-Matrix
- ▶ Da jede Schaltfunktion kann als DNF (sum of products form) oder KNF (product of sums form) dargestellt werden kann
 - ▶ Ist eine Realisierung von Schaltungen mithilfe von DNF/KNF möglich
- ▶ PLAs verwenden üblicherweise DNFs verwendet

Schaltkreisrealisierung durch PLAs

- ▶ Üblicherweise wird die DNF verwendet
- ▶ Ausgangsbasis Wahrheitstabelle, Eingabekombinationen als Produkte mit Ausgabe 1
- ▶ Diese Herangehensweise führt zu einer Zwei-Level-Repräsentation
- ▶ PLA: Halbleiterschaltkreis, bestehend aus hintereinander geschalteten AND- und OR-Matrizen, um Schaltwerke für logische Funktionen in DNF zu erstellen

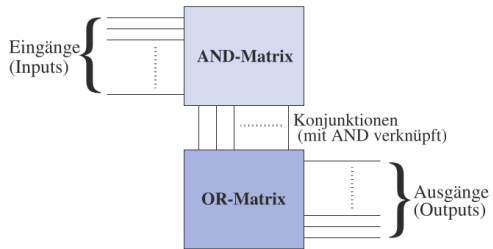
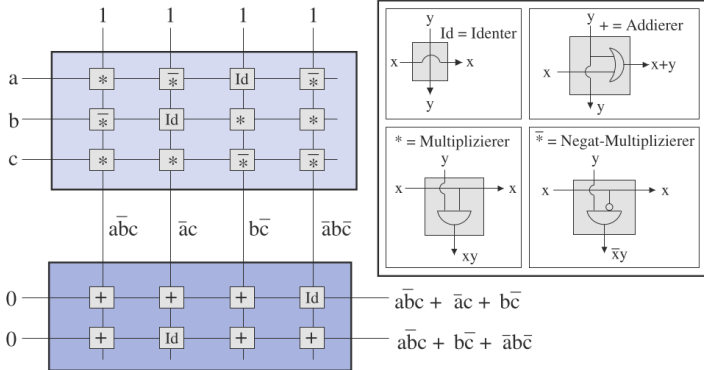


Abbildung: Programmable Logic Array (PLA)

Schaltkreisrealisierung durch PLAs

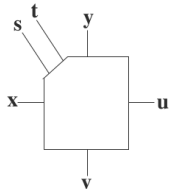
- Die AND-Matrix repräsentiert dabei die Konjunktionsterme
 - Termauswahl erfolgt bei der Programmierung mittels eines speziellen Geräts durch das Entfernen von Schaltgliedern aus der AND-Matrix
- Disjunktive Verknüpfung der Konjunktionsterme erfolgt mit der OR-Matrix



Schaltkreisrealisierung durch PLAs

- ▶ Zuordnung numerischer Wert zu Schaltungstyp: Giterpunkt via $s, t \rightarrow$ Bausteintyp (eigentliche Programmierung)
- ▶ Überführung der Logik-Gitter in Matrix der Form: $(n + m) \times k$
 - ▶ n die Anzahl der Variablen,
 - ▶ m die Anzahl der verschiedenen booleschen Funktionen und
 - ▶ k die Anzahl der Teilterme ist
- ▶ Ersten n Zeilen dieser Matrix kommen dabei nur die Werte 0, 2 und 3
- ▶ Letzten m Zeilen nur die beiden Werte 0 und 1

$$\begin{pmatrix} 2 & 3 & 0 & 3 \\ 3 & 0 & 2 & 2 \\ 2 & 2 & 3 & 3 \\ \hline 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$



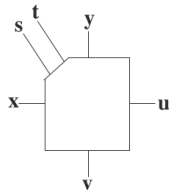
| Bausteintyp | Nummer | s | t | u | v |
|----------------------|--------|---|---|---------|------------|
| Identifier | 0 | 0 | 0 | x | y |
| Addierer | 1 | 0 | 1 | $x + y$ | y |
| Multiplizierer | 2 | 1 | 0 | x | xy |
| Negat-Multiplizierer | 3 | 1 | 1 | x | $\bar{x}y$ |

Abbildung: Über zwei Zuleitungen s und t programmierbarer Gitterbaustein.

Schaltkreisrealisierung durch PLAs

- ▶ Horizontal sind die Eingangssignale in die AND-Matrix
- ▶ Produkt-Term-Lines sind die vertikalen Eingangssignale
- ▶ Ausgaben sind u, v
- ▶ Folgenden Schaltfunktionen können hergeleitet werden:

$$u = x + \bar{s}ty, \quad v = \bar{s}y + sy(t \oplus x)$$



| Bausteintyp | Nummer | s | t | u | v |
|----------------------|--------|---|---|---------|------------|
| Identer | 0 | 0 | 0 | x | y |
| Addierer | 1 | 0 | 1 | $x + y$ | y |
| Multiplizierer | 2 | 1 | 0 | x | xy |
| Negat-Multiplizierer | 3 | 1 | 1 | x | $\bar{x}y$ |

Abbildung: Über zwei Zuleitungen s und t programmierbarer Gitterbaustein.

Schaltkreisrealisierung durch PLAs

- ▶ Ursprünglich wurde eine Matrix aus Sicherungen (Fuse Network) verwendet
 - ▶ Programmierung: Zu realisierenden logischen Funktion, einzelne Sicherungen mittels hohen Strom durchgebrannt
 - ▶ Problem: Über größere Zeiträume werden einzelne Sicherungen auf Grund von Kristallisierung wieder leitend
- ▶ Anti-Fuse-Technologie: Besteht PLA aus einer Diodenmatrix, jede Diode ein Bit repräsentiert
- ▶ Dioden so verschalten, dass sie den Strom sperren
 - ▶ Programmierung: Gezieltes zerstören bestimmter Dioden mittels eines sehr hohen Stroms
 - ▶ Hierdurch wird leitende Verbindung realisiert
 - ▶ Nach dem „Brennen“ werden die geschriebenen Daten durch Bitmuster defekter/funktionierender Dioden repräsentiert
- ▶ Daten beliebig oft auslesbar, einmal programmierbar – keine Änderungen

Nutzung PLAs



- ▶ Lösung: GAL (Generic Array Logic)
- ▶ PLAs nur für kleine Logikbausteine, größere Probleme mit ASIC, FPGA und CPLD etc.
- ▶ Programmable Array Logic (PAL, nur AND-Matrix programmierbar) und Programmable Read-Only Memory (PROM, nur OR-Matrix programmierbar)
- ▶ PLA für Kontrolle von Datenpfade – Definiert Zustände im Instruction-Set und gibt zulässige Folgezustände vor
- ▶ PLAs als Zählfunktion
- ▶ PLA als Decoder
- ▶ PLA als BUS-Schnittstelle für IO-Programmierung

Plakatives Beispiel

- ▶ Eingangssignal 1: Anschaltknopf (an/aus)
- ▶ Eingangssignal 2: Sicherheitsschalter (an/aus)
- ▶ Ausgangssignal: Motor (an/aus)
- ▶ Mögliche Programmierung:
 - ▶ Wenn Anschaltknopf = an UND Sicherheitsschalter = an, dann Motor = an.
 - ▶ Wenn Anschaltknopf = an UND Sicherheitsschalter = aus ODER
 - ▶ wenn Anschaltknopf = aus UND Sicherheitsschalter = an ODER
 - ▶ wenn Anschaltknopf = aus UND Sicherheitsschalter = aus, dann Motor = aus.

[Wik21]

Quellen I

-  Hoffmann, Dirk W (2020). *Grundlagen der technischen Informatik*. Carl Hanser Verlag GmbH Co KG.
-  Wikipedia (2021). *Programmierbare logische Anordnung*. https://de.wikipedia.org/wiki/Programmierbare_logische_Anordnung. Accessed: 2021-02-12.