4. Standard-Schaltnetze

4.1 Codierer und Busse

4.2 Schaltnetzrealisierung durch Speicher und PLAs

1

4. Standard-Schaltnetze

Bisher haben wir uns einen Werkzeugkasten zum Umsetzen von Funktionsbeschreibungen in Schaltungen zugelegt. Das ist quasi unser 1x1. Wie in der Schule folgen auf das 1x1 jetzt viele weitere Operationen: Dividieren, Wurzel ziehen, Potenzieren, Logarithmieren, Integrale.... Also Operationen, deren Bearbeitung das 1x1 erfordert. In diesem Stadium sind wir jetzt.

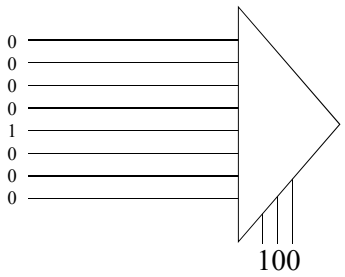
Wir lernen jetzt Grundschaltungen kennen, denen wir in Hardwaresystemen immer wieder begegnen werden, und die mit unserem Werkzeugkasten zu verstehen sind. Ziel ist, dass Sie am Ende ein Gefühl dafür haben, was ein Addierer macht, was ein Multiplexer, ein Zähler usw.

2

Der Codierer

Ein Codierer ist ein Schaltnetz, das eine Nachricht in einem definierten Code als Input bekommt, und diese in einen anderen Code wandelt und ausgibt. Sie erinnern sich an den 7-Segment-Codierer für die Anzeige der Dezimalziffern.
Häufig möchte man, dass einer der Codes ein 1-aus-N-Code ist. Dies ist eine Codierung der Zahlen von 0 bis N-1, bei der genau ein Bit 1 ist und alle anderen 0. Ein 1-aus-N-Code hat genau N verschieden Codeworte.
Wie baut man einen Codierer, der einen 1-aus-N-Code in einen Birärcode mit n Stellen übersetzt?

N Eingänge



n Ausgänge

$N=2^n$

$n = \log_2 N$

3

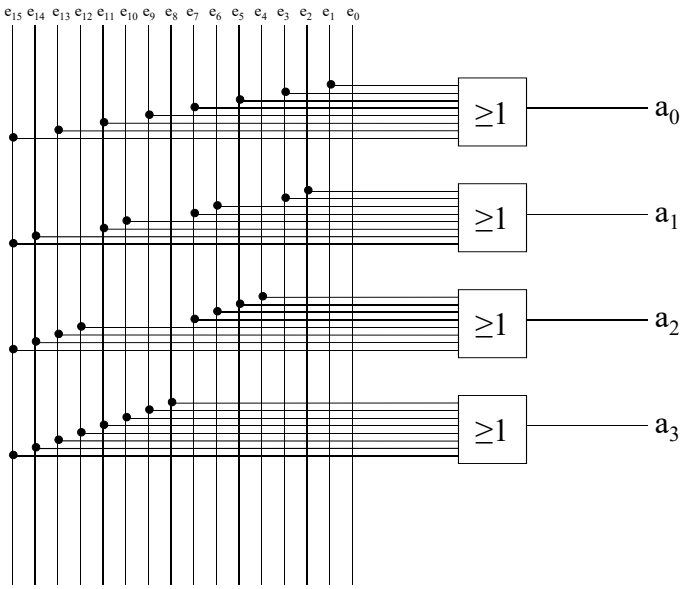
Relevanter Teil der Wertetabelle für 1-aus-16-in-Binär-Codierer

e_{15}	e_{14}	e_{13}	e_{12}	e_{11}	e_{10}	e_9	e_8	e_7	e_6	e_5	e_4	e_3	e_2	e_1	e_0	a_3	a_2	a_1	a_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Alle anderen Werte sind x (don‘t care)

4

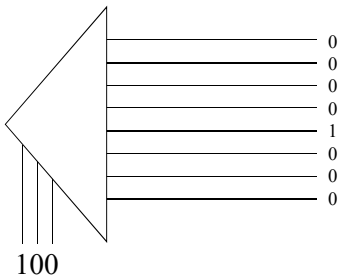
1-aus-16-in-Binär-Codierer



5

Der Decodierer

Jetzt ist umgekehrt der Eingang ein n-stelliger Binärcode und der Ausgang ein 1-aus-N-Code ist.

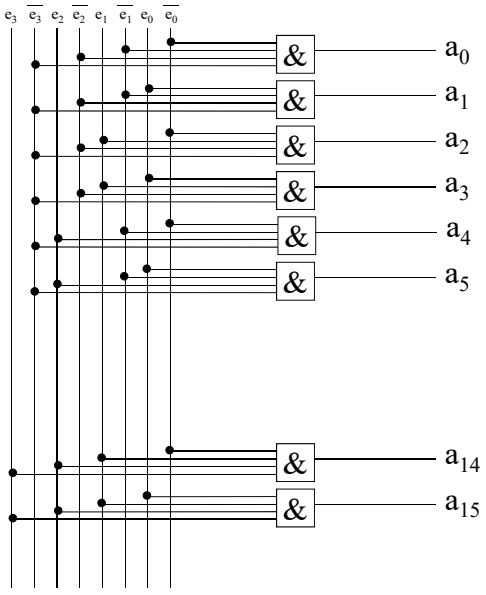


6

Wertetabelle für Binär-in-1-aus-16-Decodierer

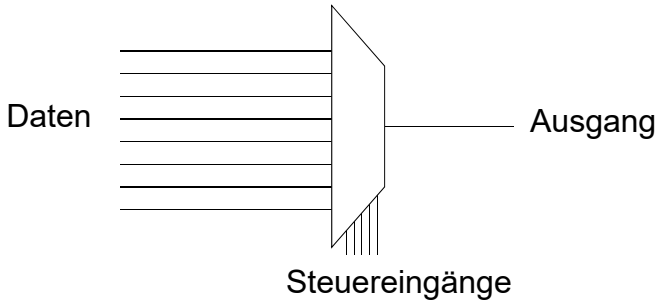
e_3	e_2	e_1	e_0	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Binär-in-1-aus-16-Decodierer



Der Multiplexer

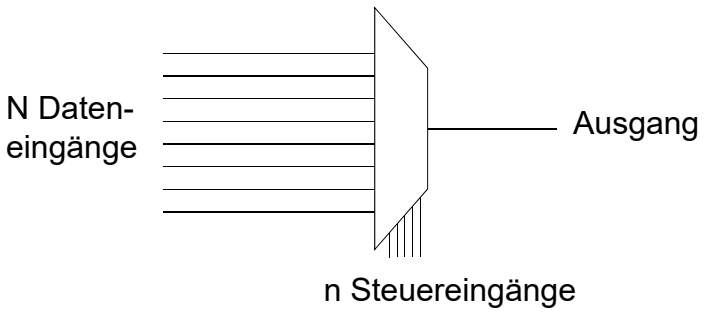
Ein Multiplexer ist ein Schaltnetz, das einen von mehreren Eingängen wählt, und diesen unverändert auf den Ausgang legt. Man kann sich einen Multiplexer wie eine Verzeigung (oder Weiche) vorstellen. Multiplexer haben Dateneingänge und Steuereingänge. Abhängig von den Signalen der Steuereingänge wird der richtige Dateneingang ausgewählt.



9

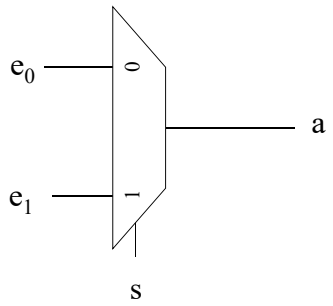
Seien die Dateneingänge mit e_i bezeichnet, i=0,...,N-1, die Steuereingänge mit s_i, i=0,..,n-1, der Ausgang mit a. Dann muss $N \leq 2^n$ sein. Die Funktion des Multiplexers wird beschrieben durch

$a = e_i$, falls $(i)_{10}=(s_{n-1}s_{n-2}...s_0)_2$



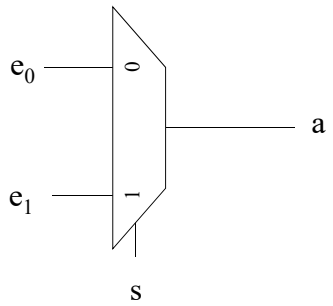
10

Beispiel: Ein 2-auf-1-Multiplexer. Dieser hat 2 Dateneingänge e_0 und e_1 und einen Steuereingang s. Wenn die Steuerleitung s=0 ist, so ist $a=e_0$ und wenn s=1 ist, so ist $a=e_1$.



s	e_1	e_0	a
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

11



s	e_1	e_0	a
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

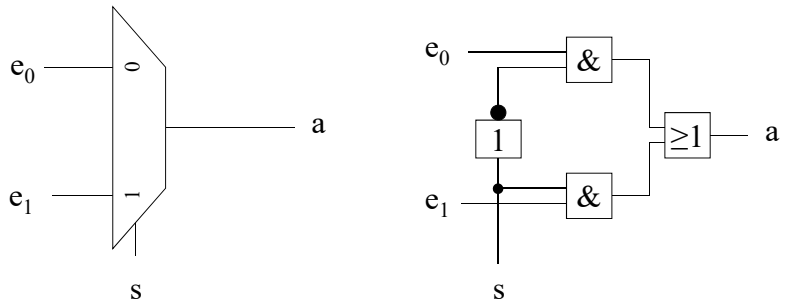
a \ s	e_0	e_1
0	1	0
1	1	0

$$a = \bar{s}e_0 + se_1$$

12

Realisierung:

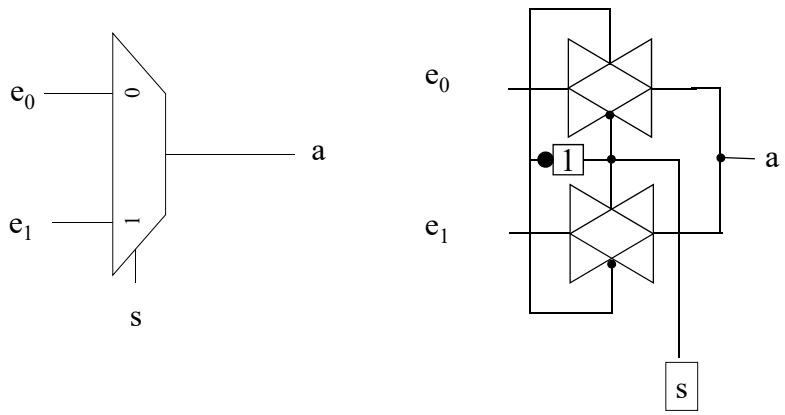
$$a = \bar{s}e_0 + se_1$$



13

2. Möglichkeit: Realisierung über Transmissionsgatter:

$$a = \bar{s}e_0 + se_1$$



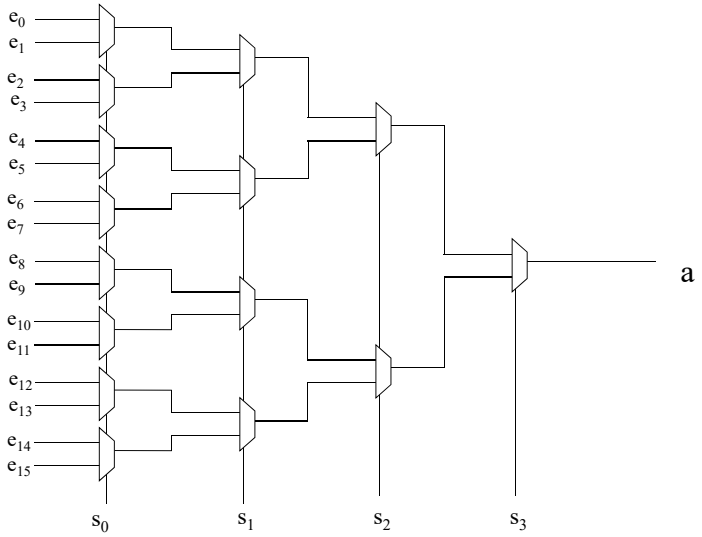
14

Wertetabelle für 16-auf-1-Multiplexer

s_3	s_2	s_1	s_0	a
0	0	0	0	e_0
0	0	0	1	e_1
0	0	1	0	e_2
0	0	1	1	e_3
0	1	0	0	e_4
0	1	0	1	e_5
0	1	1	0	e_6
0	1	1	1	e_7
1	0	0	0	e_8
1	0	0	1	e_9
1	0	1	0	e_{10}
1	0	1	1	e_{11}
1	1	0	0	e_{12}
1	1	0	1	e_{13}
1	1	1	0	e_{14}
1	1	1	1	e_{15}

15

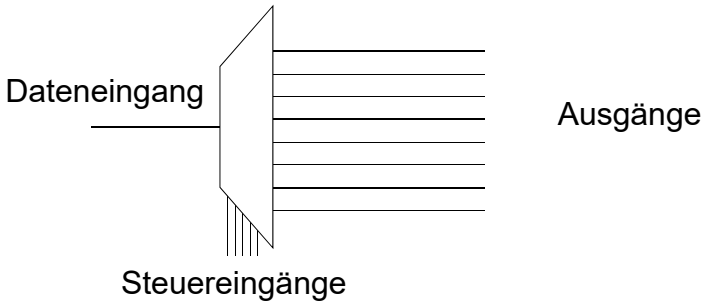
Mögliche Realisierung: ($\log_2 n$)-stufige 2-auf-1 Multiplexer



16

Der Demultiplexer

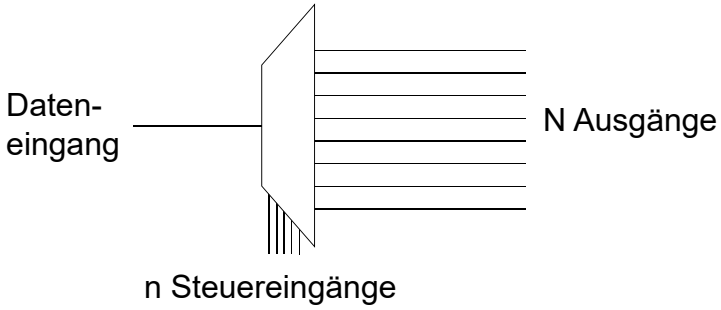
Ein Demultiplexer ist ein Schaltnetz, das seinen einzigen Dateneingang an einen von N Ausgängen übertragen kann. Die Auswahl, welcher Ausgang gewählt wird, wird durch die Steuereingänge bestimmt.



17

Seien die Datenausgänge mit a_i bezeichnet, i=0,...,N-1, die Steuereingänge mit s_i, i=0,..,n-1, der Eingang mit e. Dann muss $N \leq 2^n$ sein. Die Funktion des Demultiplexers wird beschrieben durch

$a_i = e$, falls $(i)_{10}=(s_{n-1}s_{n-2}...s_0)_2$, sonst $a_i = 0$

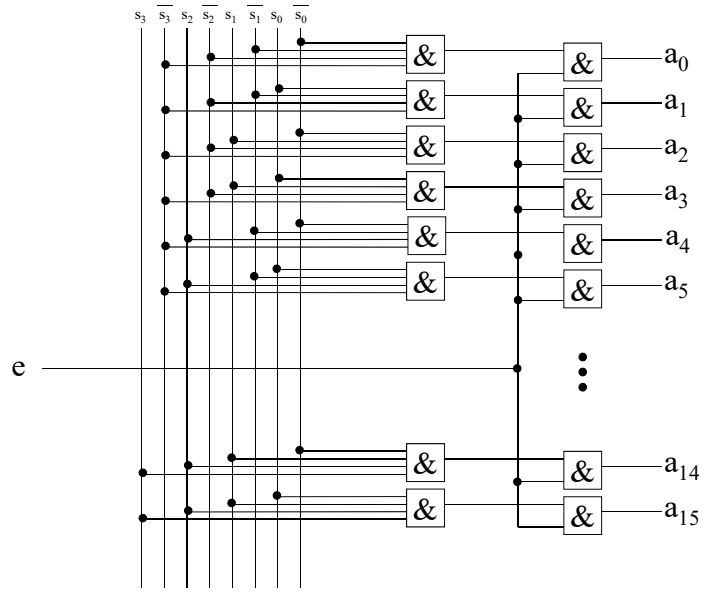


18

Wertetabelle für 1-aus-16-Demultiplexer

e	s_3	s_2	s_1	s_0	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
0	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

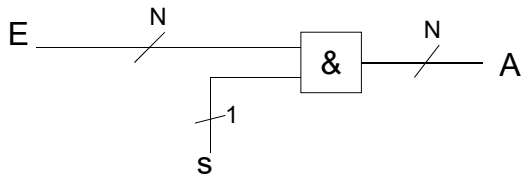
Realisierung eines 1-aus-16-Demultiplexers



Der Datenwegschalter

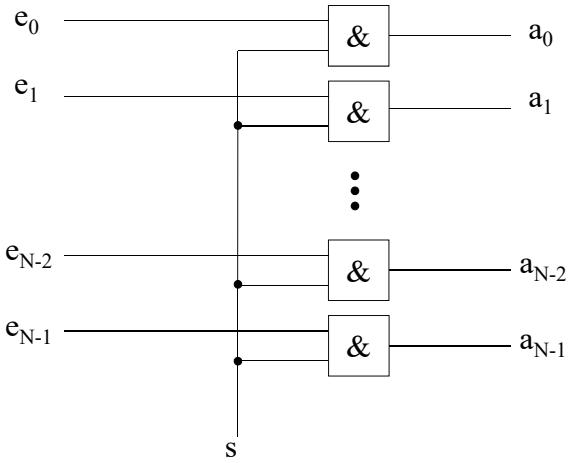
Ein Datenwegschalter ist ein Schaltnetz, das eine Reihe von Eingängen e_i unverändert an eine entsprechende Reihe von Ausgängen a_i weiterleitet, wenn ein Steuereingang auf 1 gesetzt ist (i=0,...,N-1). Wenn der Steuereingang auf 0 ist, wird an alle Ausgänge eine 0 gegeben.

$$a_i = \begin{cases} e_i, \text{ falls } s=1 \\ 0, \text{ falls } s=0 \end{cases}$$



21

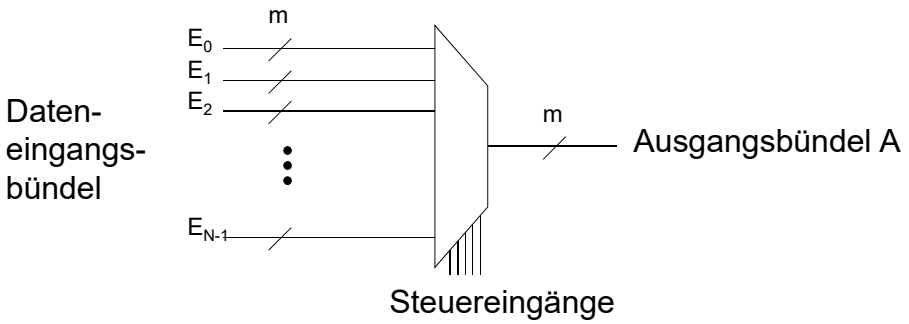
Realisierung des Datenwegschalters



22

Der Datenwegmultiplexer

Ein Datenwegmultiplexer ist ein Schaltnetz, das aus m parallel geschalteten Multiplexern besteht. Mit ihm kann man Leitungsbündel der Stärke m schalten. Alle parallelen Leitungen werden durch die Steuereingänge auf das selbe Ausgangsbündel geschaltet.

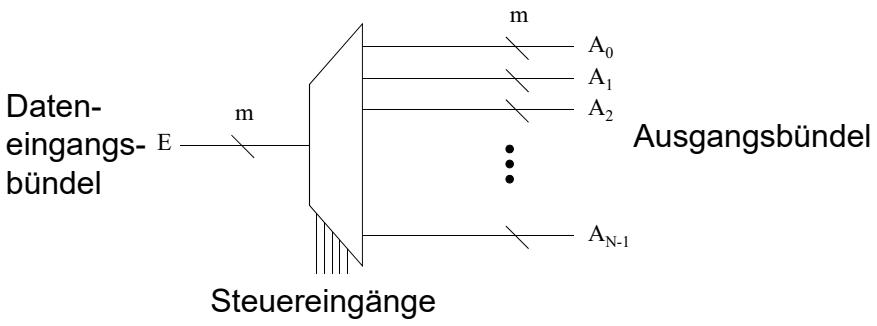


Für alle i=0,...,m-1 gilt $a_i = e_{k,i}$ genau dann wenn $(k)_{10} = (s_{n-1}...s_0)_2$

23

Der Datenwegdemultiplexer

Ein Datenwegdemultiplexer ist ein Schaltnetz, das aus m parallel geschalteten Demultiplexern besteht. Mit ihm kann man ein Leitungsbündel der Stärke m auf eins von N Ausgangsleitungsbündeln schalten.



Für alle i=0,...,m-1 gilt $a_{k,i} = e_i$ genau dann wenn $(k)_{10} = (s_{n-1}...s_0)_2$
alle anderen $a_{k,i}$ sind 0.

24

Vereinfachte Wertetabelle des Datenweg-Demultiplexers

s_{n-1}	s_{n-2}	.....	s_2	s_1	s_0	Ausgangsbündel
0	0	....	0	0	0	$A_0 = E, A_i = 0$ für $i \neq 0$
0	0	....	0	0	1	$A_1 = E, A_i = 0$ für $i \neq 1$
0	0	....	0	1	0	$A_2 = E, A_i = 0$ für $i \neq 2$
0	0	....	0	1	1	$A_3 = E, A_i = 0$ für $i \neq 3$
0	0	....	1	0	0	$A_4 = E, A_i = 0$ für $i \neq 4$
⋮						⋮
1	1	....	1	1	0	$A_{N-2} = E, A_i = 0$ für $i \neq N-2$
1	1	....	1	1	1	$A_{N-1} = E, A_i = 0$ für $i \neq N-1$

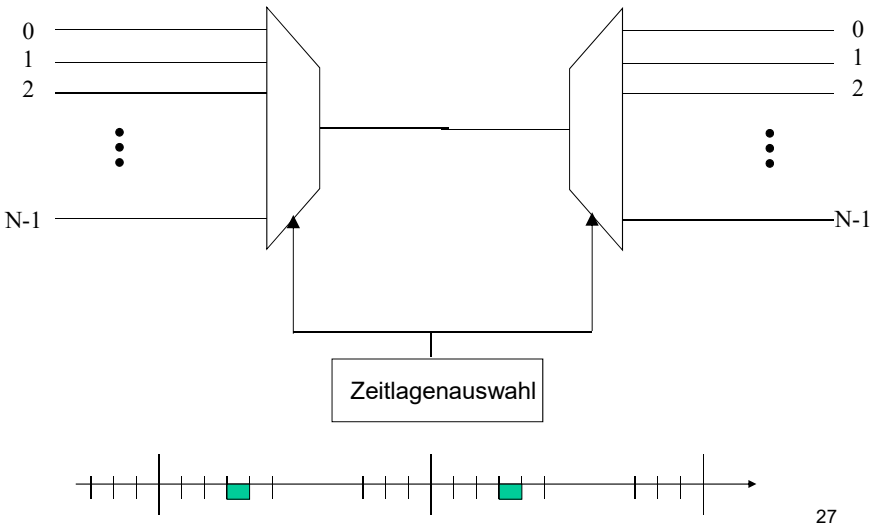
25

Zeitmultiplexübertragung

Häufig ist es notwendig, dass über dieselbe Leitung mehrere verschiedene Kommunikationsvorgänge durchgeführt werden müssen. Ein einfaches Beispiel dafür ist Ihr Telefonanschluss zu Hause, über den (z.B. bei ISDN) zwei Telefongespräche gleichzeitig und vielleicht zusätzlich (über DSL) Datenverkehr mit Ihrem Rechner abgewickelt werden. Dies gelingt, indem man die Leitung immer abwechselnd für sehr kurze Zeitintervalle jedem der beteiligten Kommunikationspaare zur Verfügung stellt. Diese Technik nennt man **Zeitmultiplex-Technik**.
Sie kann durch ein Schaltnetz aus einem Multiplexer und einem Demultiplexer realisiert werden, die beide durch eine gemeinsame Steuereinheit, die sogenannte **Zeitlagenauswahl**, gesteuert werden.

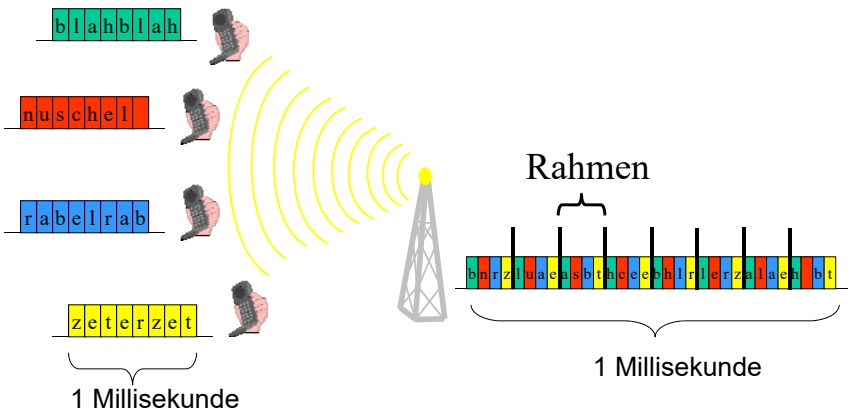
26

Zeitmultiplextechnik (Time Division Multiplex Access, TDMA)



27

Beispiel für Zeit-Multiplex: GSM-Netz



Daten werden mit N-facher Geschwindigkeit für jeden N-ten Zeitslot codiert (hier N=4)

28

Der Datenbus

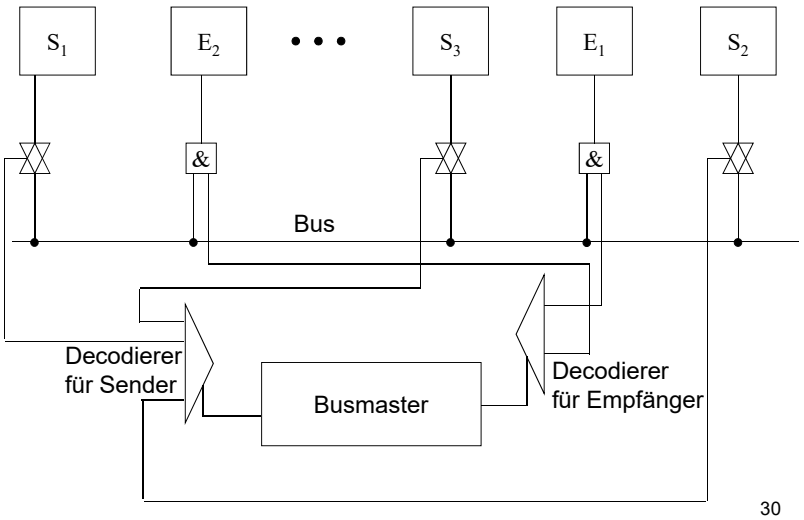
Ein Zeitmultiplexschema ist sehr statisch, denn jedes Kommunikationspaar bekommt einen festen Anteil an der Übertragungsbandbreite der Leitung. In vielen Anwendungen in der Informatik ist es allerdings wesentlich wichtiger, dass die Leitungsbandbreite dynamisch jeweils an die Kommunikationspaare zugeteilt werden kann, die aktuell den Bedarf zur Kommunikation hat. Dies führt zum Konzept des Datenbus.

Ein Datenbus ist eine Leitung oder ein Leitungsbündel, über das verschiedene angeschlossene Geräte Daten austauschen können. Diese Geräte können schreibend oder lesend auf den Bus zugreifen. Schreibende Zugreifer bezeichnet man auch als Sender, lesende als Empfänger. Natürlich kann dasselbe Gerät manchmal als Sender und manchmal als Empfänger zugreifen.

In seiner einfachsten Form ist der Datenbus eine gemeinsame Leitung (oder ein Leitungsbündel), auf das jeder Teilnehmer über ein Transmissionsgatter schreiben kann und von dem er über einen Datenwegschalter lesen kann. Die Steuerleitungen werden wieder von einer zentralen Einheit (Busmaster, Bus-Controller) bedient, die den Bus an die Kommunikationspaare nach Bedarf zuteilt.

29

Datenbus



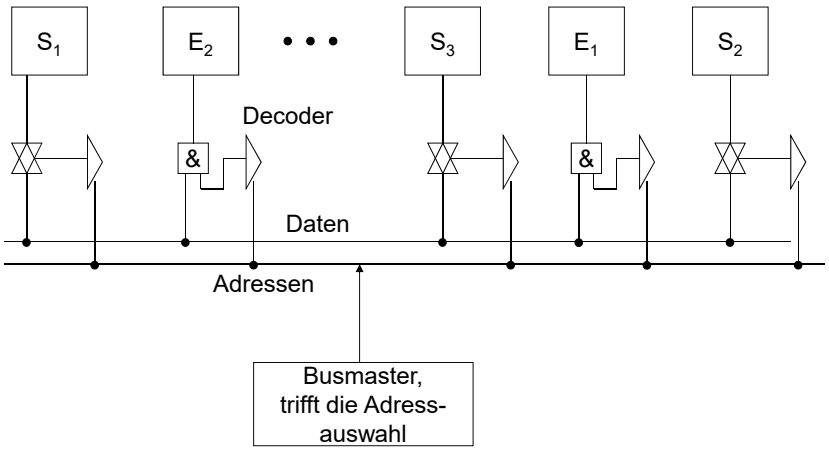
30

Der Nachteil ist offensichtlich: Der Busmaster (Bus Controller, Arbiter) muss eine direkte Leitung von seinem jeweiligen Decodierer zu jedem potentiellen Sender und Empfänger haben, die er zu geeigneter Zeit auf 1 legt, wenn ein Sender etwas für einen der mehrere Empfänger auf den Bus legen möchte.

Dieser Nachteil lässt sich vermeiden, indem man den Bus um ein Adressleitungsbündel ergänzt. Der Busmaster muss jetzt nur noch die Adressen der Kommunikationspartner auf den Adressbus legen, und jeder Sender und Empfänger hat einen Decoder, der den Adressbus liest und im Falle, dass seine Adresse auf dem Bus liegt, eine 1 für sein Transmissionsgatter bzw. seinen Datenwegschalter generiert.

31

Datenbus mit Adressbus



32

Nun stellt sich allerdings heraus, dass häufig dieselben beiden Kommunikationspartner nacheinander sehr viele Daten austauschen wollen, bevor das Kommunikationspaar wechselt. In diesem Falle bleibt die Adressauswahl lange gleich, während die Daten dauernd wechseln. Es wäre nun unwirtschaftlich, für die (über lange Zeit konstanten) Adressen ein eigenes Leitungsbündel zu realisieren.

Stattdessen verwendet man das vorhandenen Leitungsbündel für die Daten im Zeitmultiplex auch für die Adressen. Genauer: Am Anfang einer Kommunikation werden die Adressen des Senders und des/der Empfänger auf dem Bus gelegt. Diese beschalten ihre Transmissionsgatter und ihre Datenwegschalter. Danach werden kontinuierlich Daten übertragen, bis der Kommunikationsvorgang beendet ist.

Für einen solchen Bus ist ein Protokoll erforderlich, das bestimmte Absprachen über die Interpretation der Signale auf dem Bus beinhaltet. Zum Beispiel muss jeder Teilnehmer am Bus informiert werden, ob der Bus zur Zeit Adressen oder Daten überträgt. Auch das Ende einer Übertragung muss signalisiert werden.

33

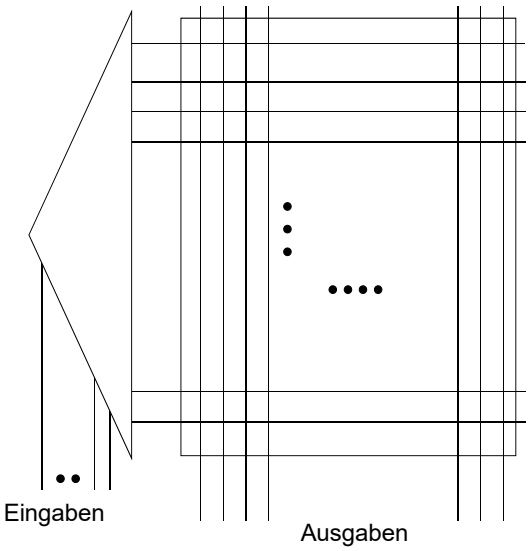
4.2 Schaltnetzrealisierung als Speicher und PLA

In der modernen Schaltungstechnik ist es häufig zu teuer, für jedes individuelle Schaltnetz eine eigene Realisierung zu bauen oder gar einen eigenen Chip. Daher gibt es viele gebräuchliche Techniken, wie man bestehende universelle Chips verwenden kann, so dass sie die Funktionalität eines gegebenen Schaltnetzes erfüllen.

Die einfachste davon ist die Realisierung von Schaltnetzen als Speicher: In einen Speicher wird die Wertetabelle der Boole‘schen Funktion geschrieben. Die Eingänge werden an die Adressleitungen des Speichers gelegt. Die Ausgänge des Speichers bilden die Ausgänge des Schaltnetzes.

34

Speicher als Schaltnetzrealisierung



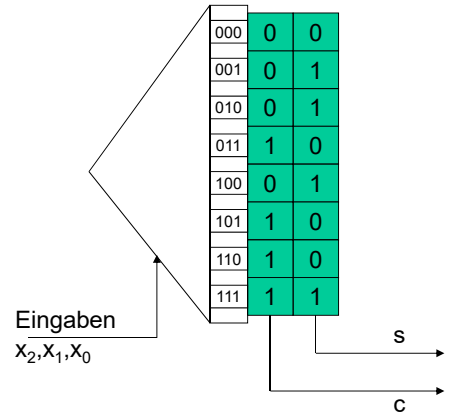
35

Beispiel: Ein Volladdierer, realisiert mit einem 8x2-Bit ROM

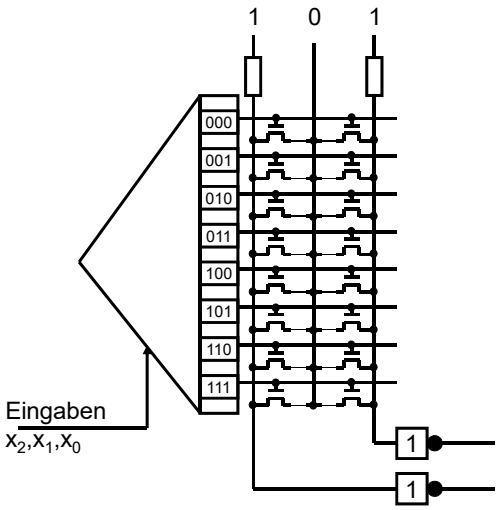
Wertetabelle

x_2	x_1	x_0	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

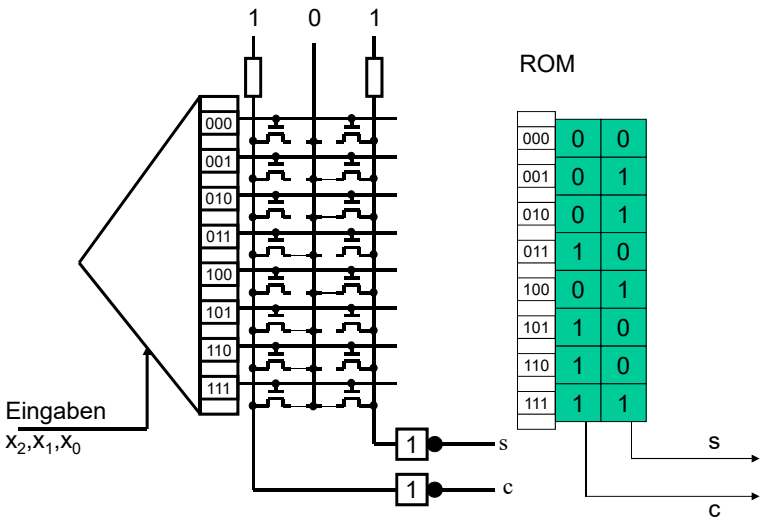
ROM



Beispiel: Ein 8x2-Bit ROM (Antifuse-Technik, d.h. nicht reprogrammierbar) vor dem Brennen zu einem Volladdierer



Beispiel: Ein 8x2-Bit ROM (Antifuse-Technik, d.h. nicht reprogrammierbar) nach dem Brennen zu einem Volladdierer



Schaltnetzrealisierung als PLA (Programmable Logic Array)

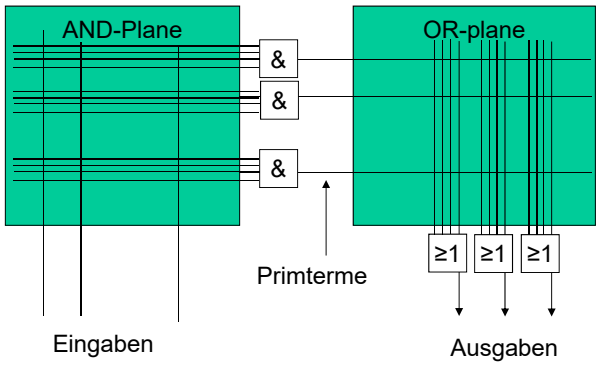
Eine andere mögliche Realisierung eines Schaltnetzes ist das PLA. Es basiert auf der Idee, ein freies Formular für eine Boole‘sche Funktion mit n Eingängen und m Ausgängen bereitzustellen, das in Abhängigkeit von der individuellen Funktion „ausgefüllt“ (gebrannt) werden kann.

Die AND-Plane generiert die Produktterme (sinnvollerweise in DMF, also Primterme), die OR-Plane fasst diese in jeweils einer Disjunktion zusammen. Natürlich werden in der tatsächlichen Realisiserung nicht AND und OR, sondern NAND-Gatter verwendet. Diese Technik kennen wir bereits.

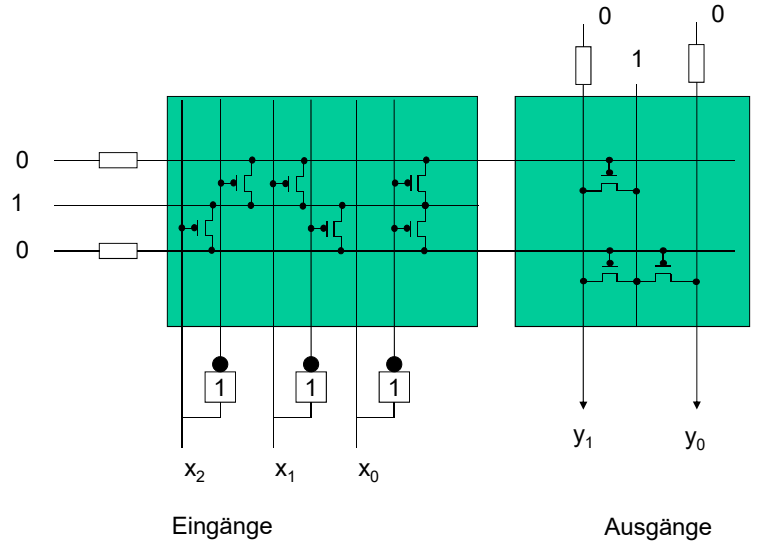
Eine Schreibweise ist bei solchen Darstellungen sinnvoll: Die AND-Verknüpfung mehrerer Leitungen, die eine Ausgabeleitung schneiden, wird durch Kreuze, die OR-Verknüpfung durch Kreise (Eselsbrücke: Buchstabe O für ODER) über dem Schnittpunkt dargestellt.

39

Realisierung eines Schaltnetzes als PLA mit AND- und OR-Plane



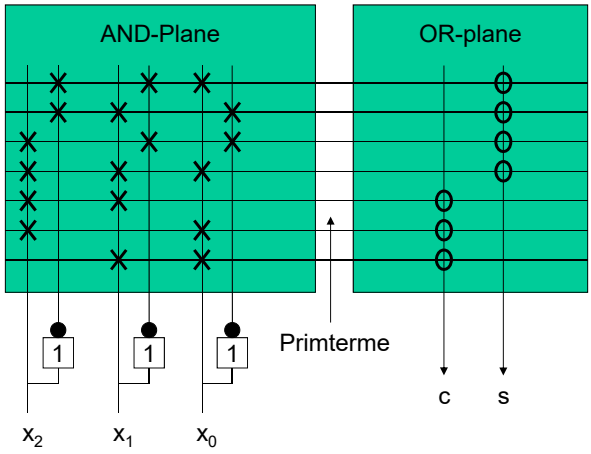
Beispiel: Funktionen $y_1 = \overline{x}_0 x_1 \overline{x}_2 + \overline{x}_0 \overline{x}_1 x_2$ und $y_0 = \overline{x}_0 \overline{x}_1 x_2$
realisiert als PLA mit AND- und OR-Plane



Beispiel: Volladdierer als PLA mit AND- und OR-Plane

Wertetabelle:

x_2	x_1	x_0	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Primterme:

$$c = x_2 x_0 + x_1 x_0 + x_2 x_1$$

$$s = \overline{x}_2 \overline{x}_1 x_0 + \overline{x}_2 x_1 \overline{x}_0 + x_2 \overline{x}_1 \overline{x}_0 + x_2 x_1 x_0$$