Computersysteme

2. Grundlagen Digitaler Schaltungen
   2.15 Fan-In und Fan-Out
   2.16 Standard-Schaltnetze

2.15 Fan-In und Fan-Out:

Fan-In: Die Anzahl der Eingänge in ein Gatter. Bestimmt die Anzahl der Transistoren in Reihe, die durchlaufen werden müssen, um den Ausgang zu treiben.

Fan-Out: Die Anzahl der Eingänge gleicher Größe, die vom Ausgang eines Gatters auf- oder entladen werden müssen. Bestimmt die Kapazität, die das Gatter beim Schaltvorgang umladen muss.

Der Widerstand und die Kapazität bilden einen Tiefpass, der das Schaltverhalten des Gatters beschreibt. Der Umschaltvorgang lässt sich durch die Exponentialkurve des Tiefpasses beschreiben. R*C sind die Zeit, die der Umladevorgang benötigt. Also ist diese Zeit sowohl proportional zu R wie auch zu C. Daher müssen Fan-In und Fan-Out in realen Schaltungen begrenzt werden.

Aufladevorgang (Wechsel von 0 auf 1)

$$U_a = U_e - U_e \cdot e^{-\frac{t}{RC}} = U_e \cdot (1 - e^{-\frac{t}{RC}})$$

$$\frac{U_a}{U_e} = 1 - e^{-\frac{t}{RC}}$$

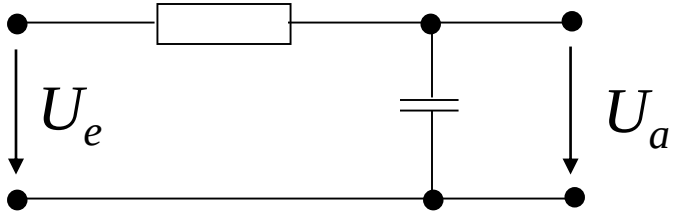
$$1 - \frac{U_a}{U_e} = e^{-\frac{t}{RC}}$$

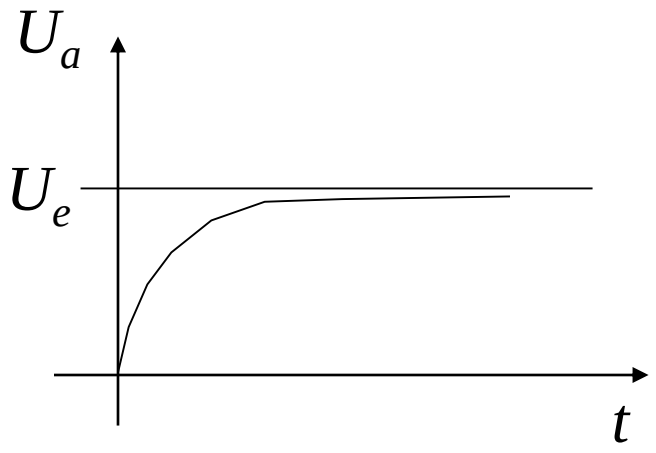
$$Annahme: \frac{U_a}{U_e} = 0{,}9$$

$$\ln(1 - 0{,}9) = -\frac{t}{RC}$$

$$-2{,}3RC \approx -t$$

$$t \approx 2{,}3RC$$





Entladevorgang (Wechsel von 1 auf 0)

$$U_a = U_e \cdot e^{-\frac{t}{RC}}$$

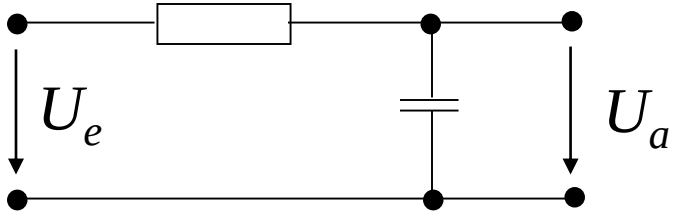
$$\frac{U_a}{U_e} = e^{-\frac{t}{RC}}$$

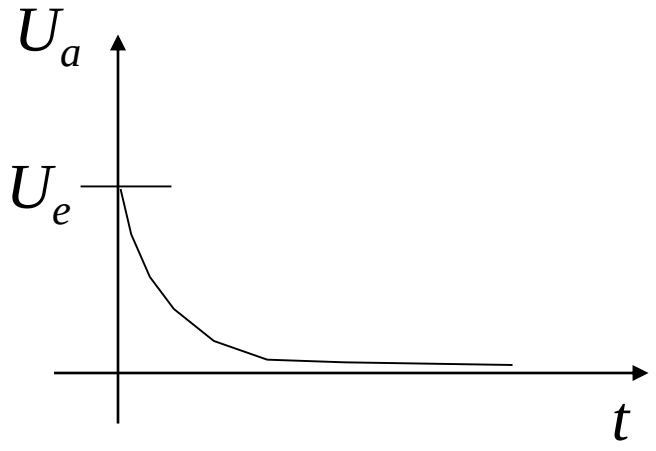
$$Annahme: \frac{U_a}{U_e} = 0{,}1$$

$$\ln 0{,}1 = -\frac{t}{RC}$$

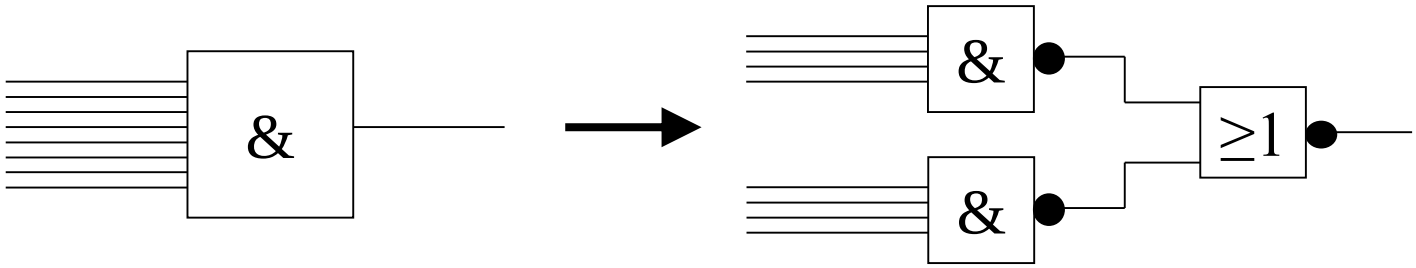
$$-2{,}3RC \approx -t$$

$$t \approx 2{,}3RC$$



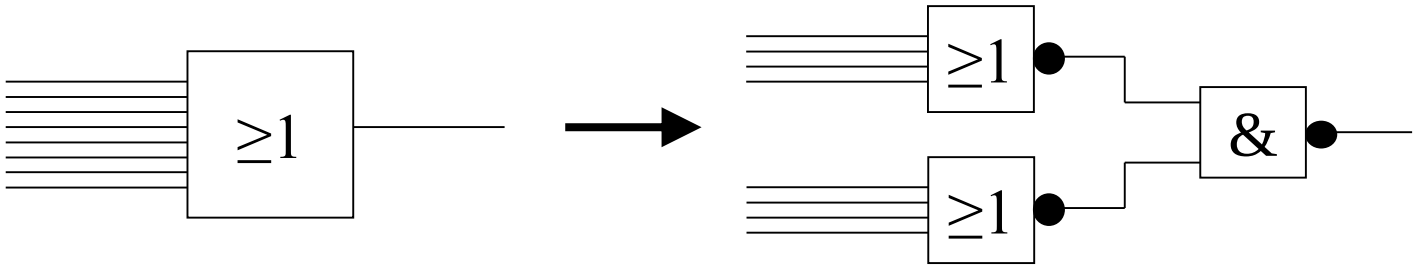


Was machen wir, wenn wir Gatter mit zu großem Fan-In haben?



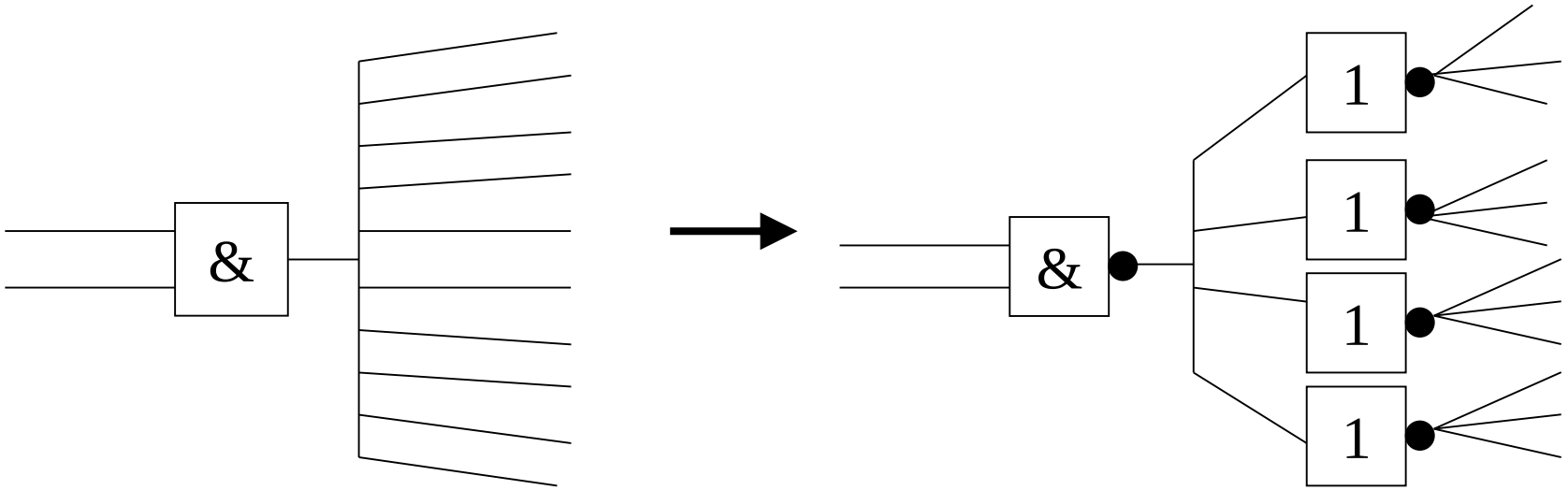
$$a \cdot b \cdot c \cdot d = \overline{\overline{a \cdot b \cdot c \cdot d}} = \overline{\overline{a \cdot b} + \overline{c \cdot d}}$$

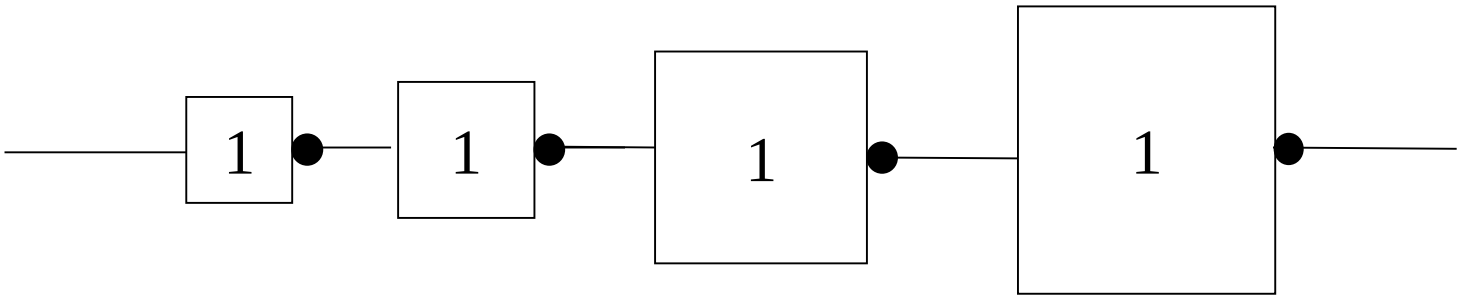
2. Beispiel: Oder Gatter mit zu großerm Fan-In



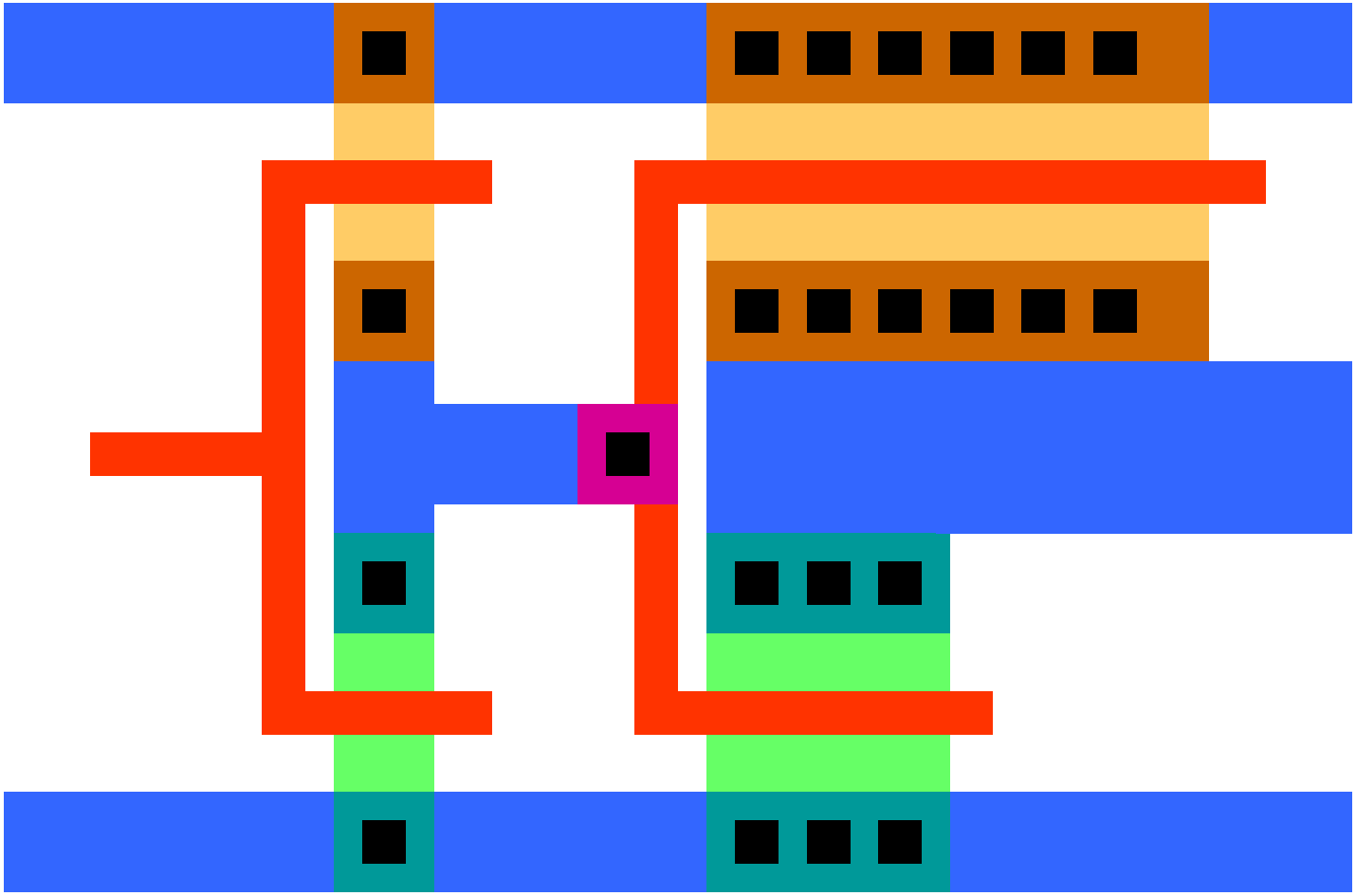
$$a+b+c+d=\overline{\overline{a+b+c+d}}=\overline{\overline{a+b}\cdot\overline{c+d}}$$

Was machen wir, wenn wir Gatter mit zu großem Fan-Out haben?





Inverterpaar als Treiber auf dem Chip von oben

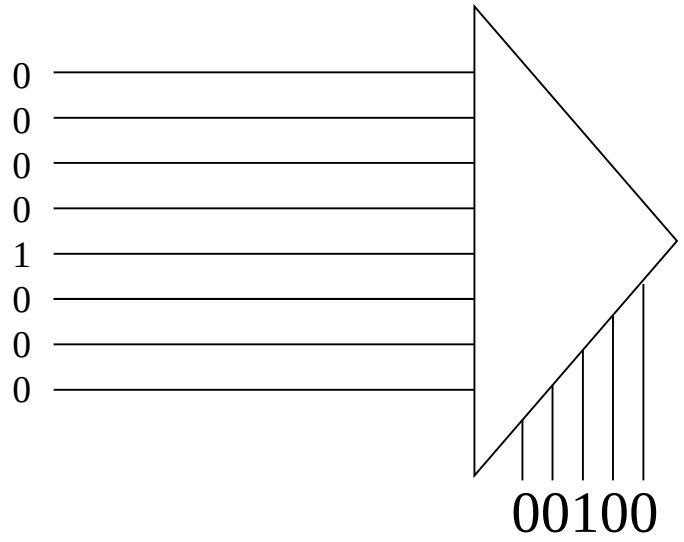


2.16 Standard-Schaltnetze

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.

Der Codierer

Ein Codierer ist ein Schaltnetz, das eine Nachricht in eine definierten Code als Input bekommt, und diese in einen anderen Code wandelt und ausgibt. Sie erinnern sich an den Binär-Aiken-Codierer für die 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 übersetzt?

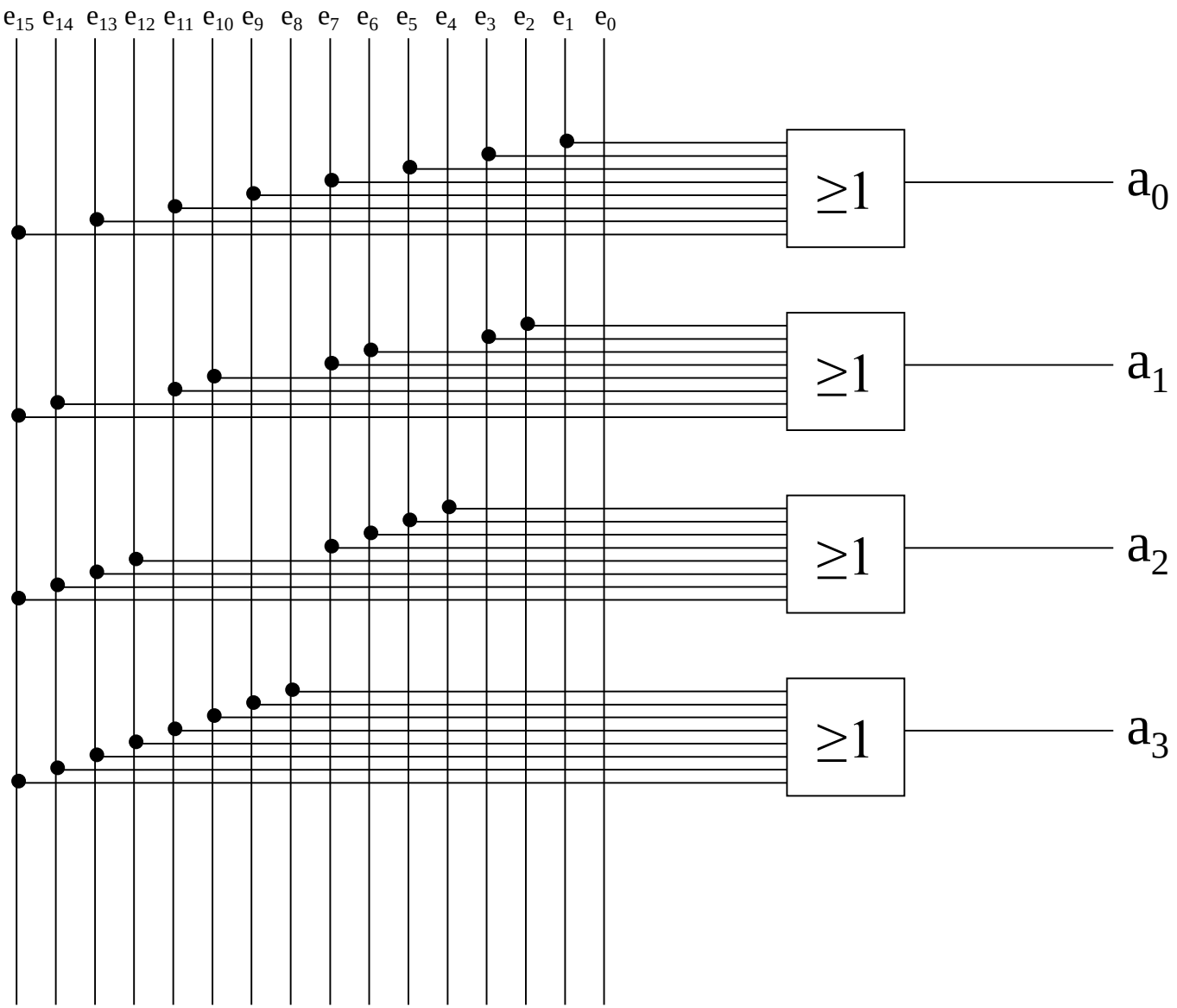


Relevanter Teil der Wertetabelle für 1-aus-16-in-Binär-Codierer

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

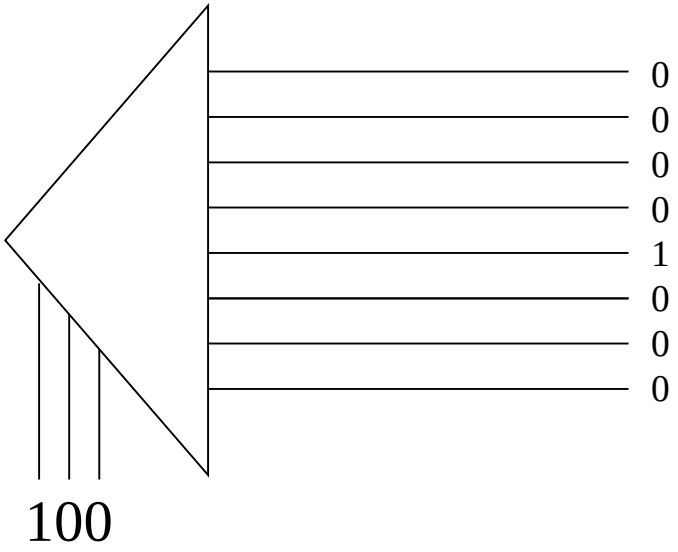
Alle anderen Werte sind X (don‘t care)

1-aus-16-in-Binär-Codierer



Der Decodierer

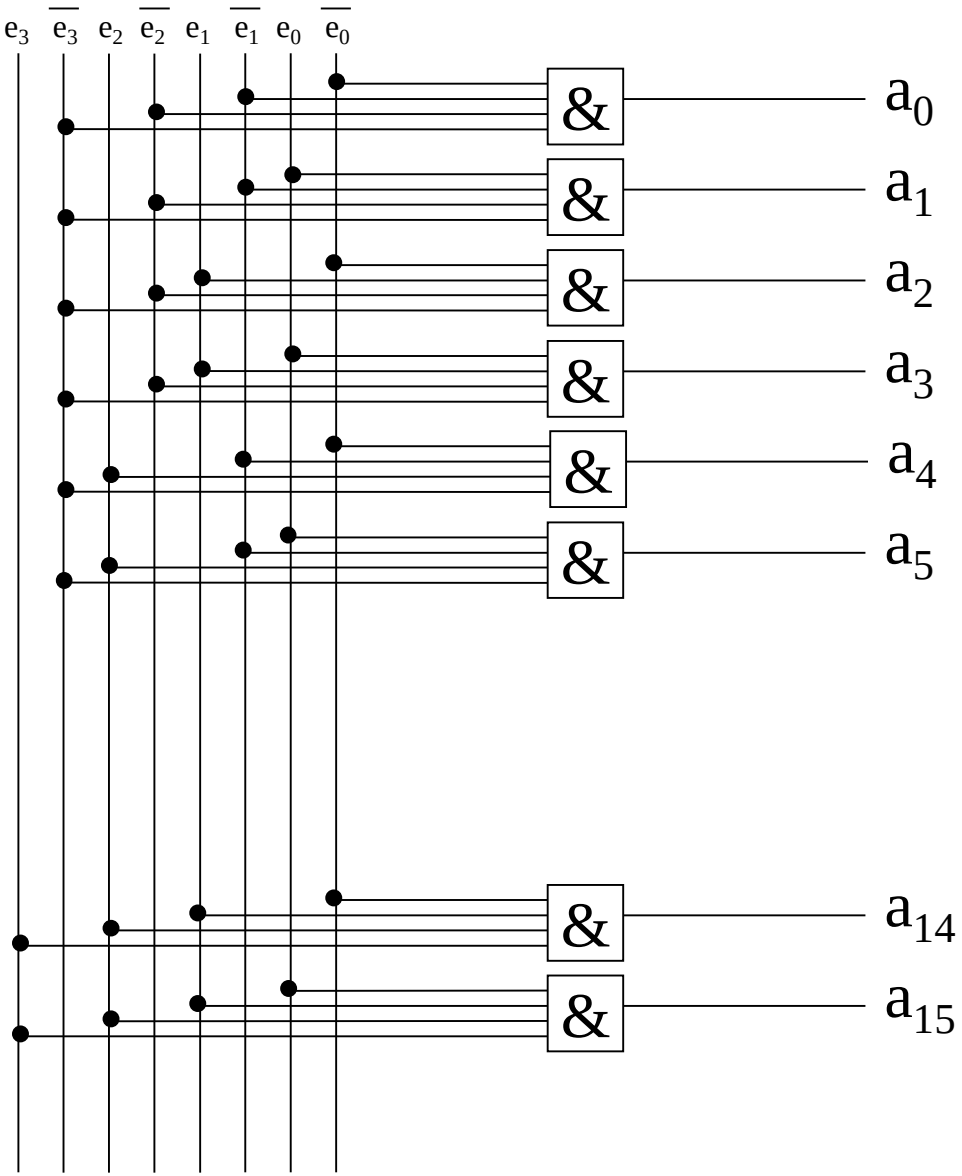
Jetzt ist umgekehrt der Eingang ein Binärcode und der Ausgang ein 1-aus-N-Code ist.



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

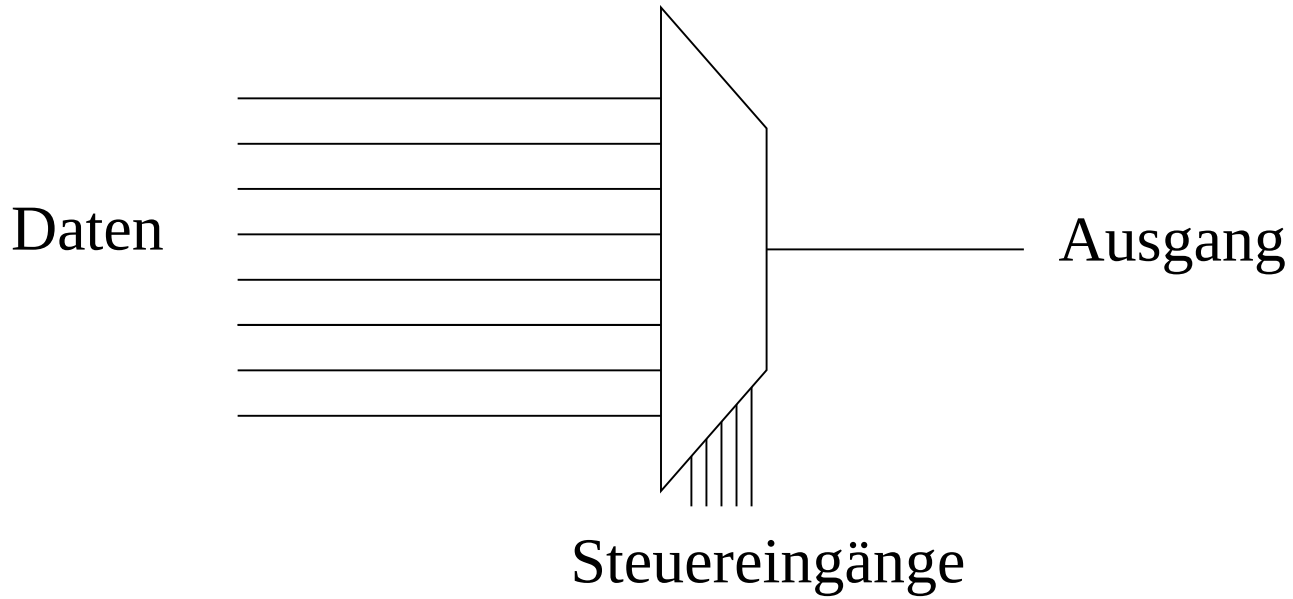
e_3	e_2	e_1	e_0	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1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



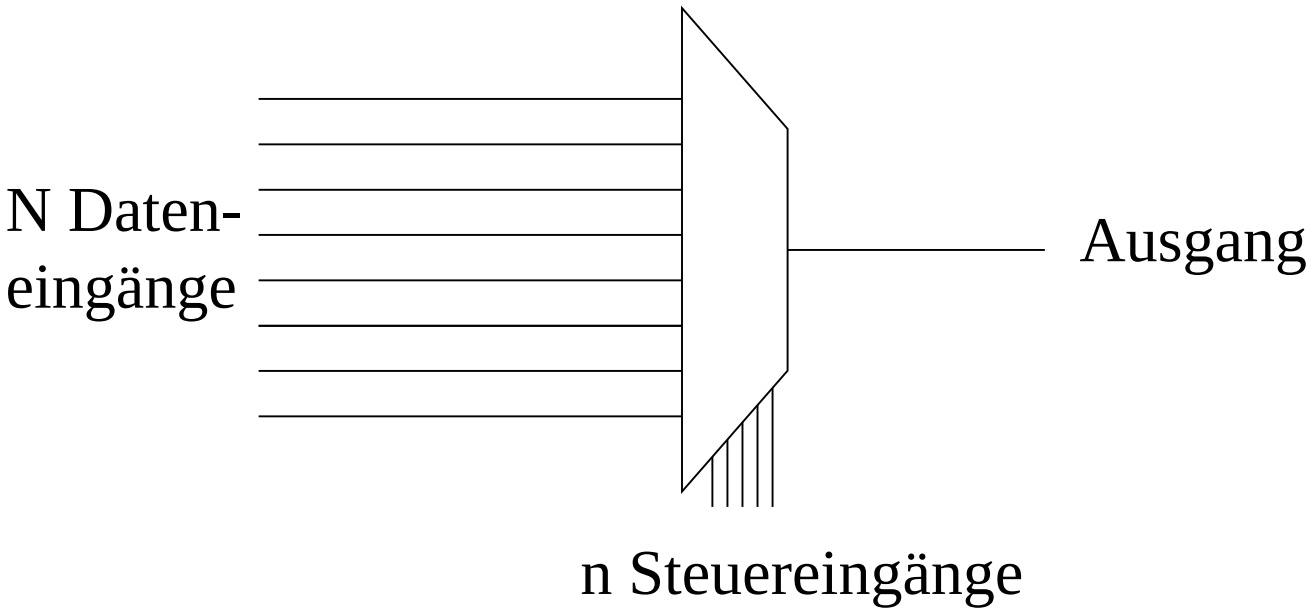
Der Multiplexer

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.

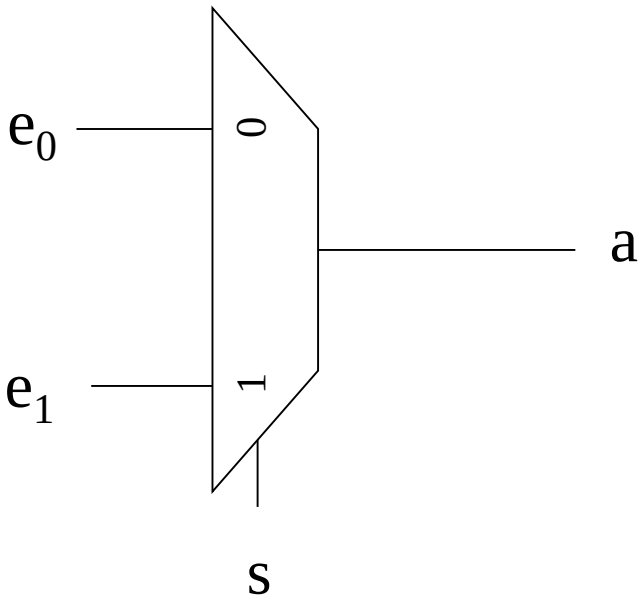


Seien die Dateneingänge mit e_i bezeichnet, i=0,...,N-1, die Steuereingänge mit s_i, i=0,..,n-1, der Ausgang mit a. Dann muss $N \leq 2^n$ sein. Die Funktion des Multiplexers wird beschrieben durch

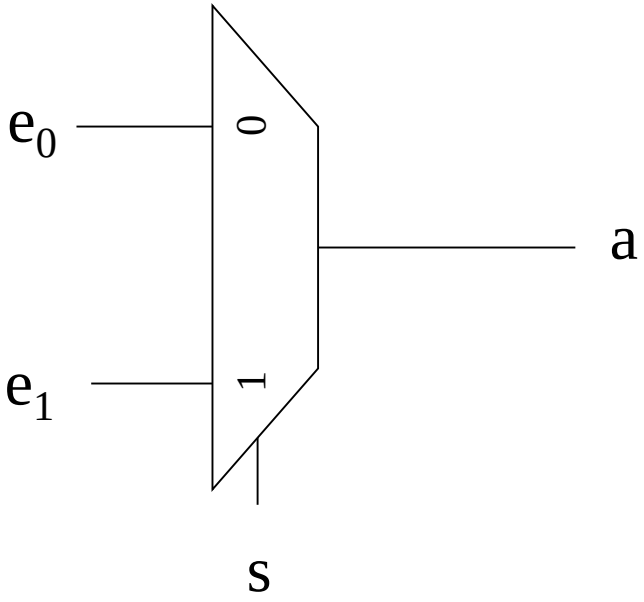
$a = e_i$, falls $(i)_{10}=(s_{n-1}s_{n-2}...s_0)_2$



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



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

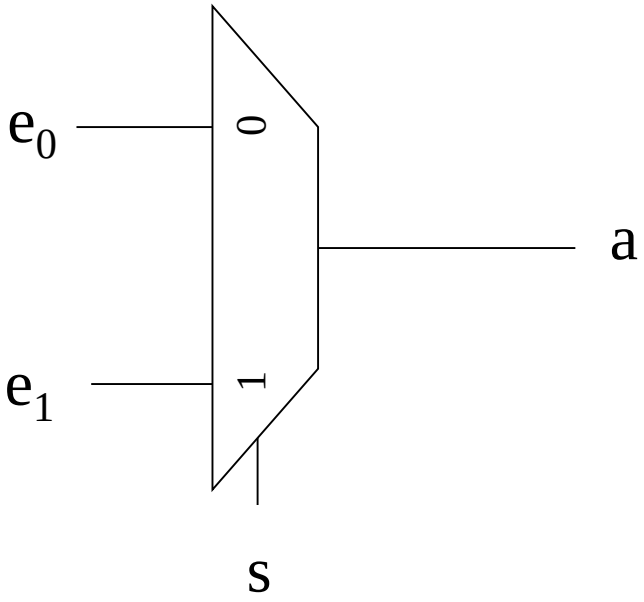
s \ e_0			
	0	1	1	0
	1	1	0	0

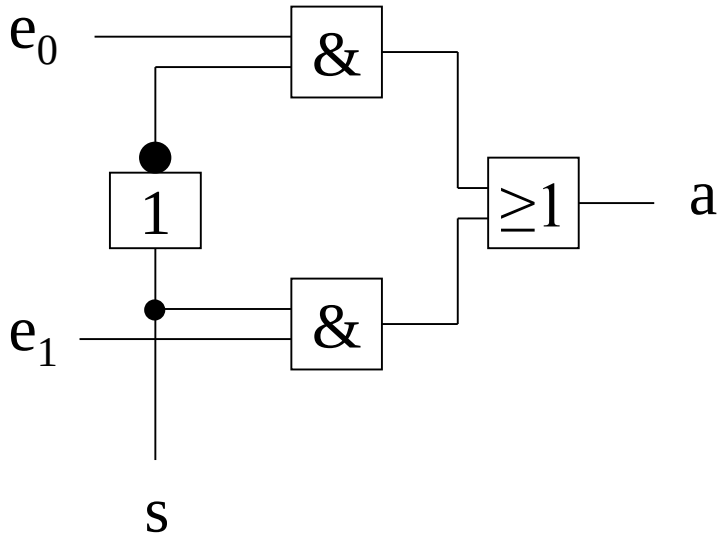
e_1

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

Realisierung:

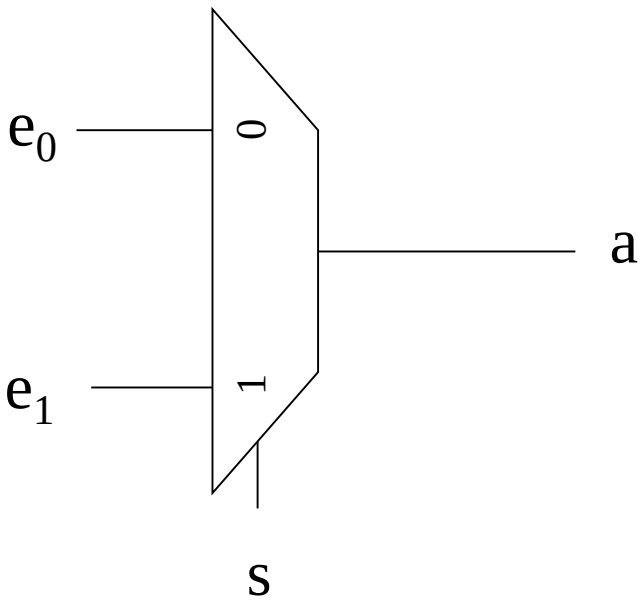
$$a = \bar{s}e_0 + se_1$$

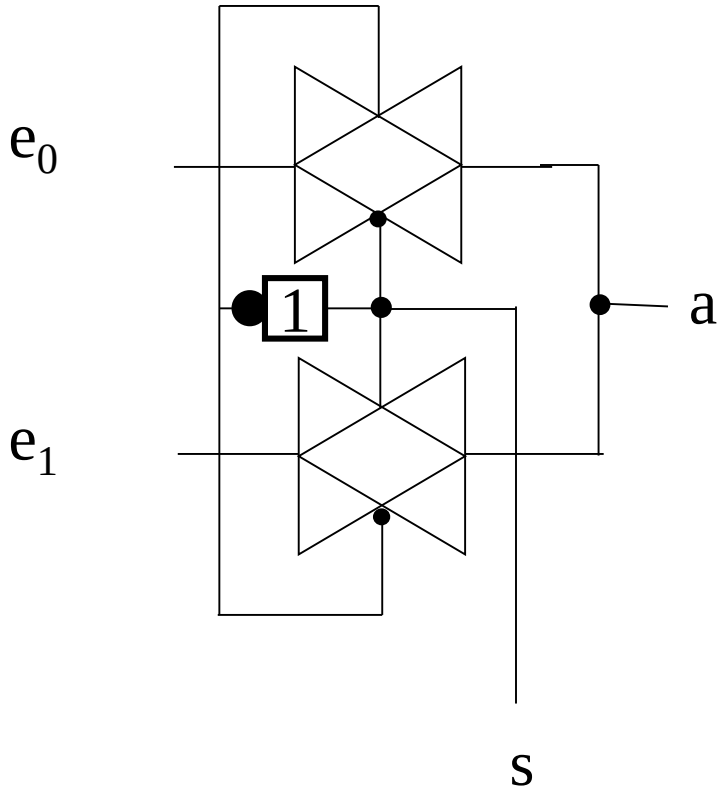




2. Möglichkeit der Realisiserung:

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



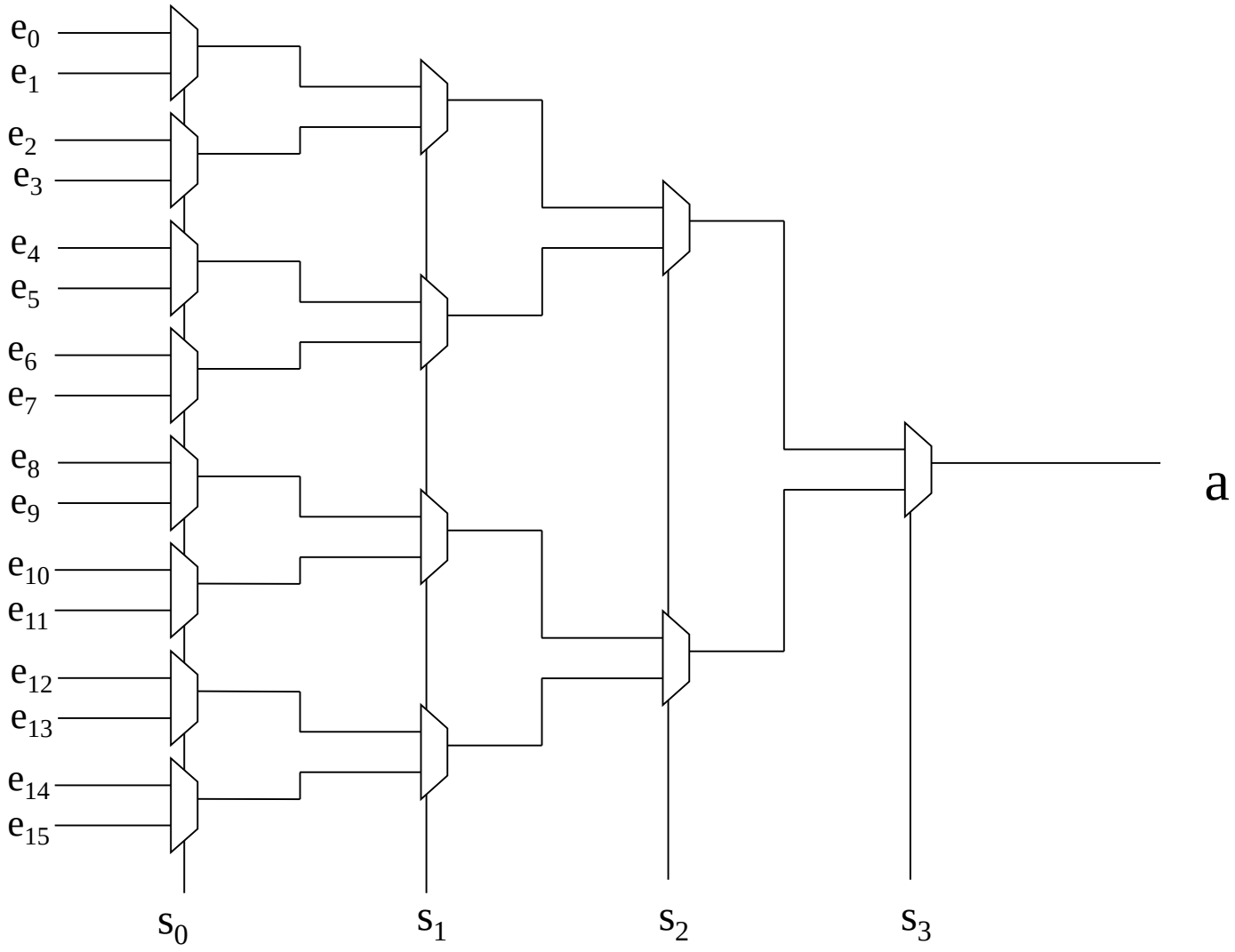


Wertetabelle für 16-auf-1-Multiplexer

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}

stark verkürzte Darstellung.
Vollständige Wertetabelle hätte $2^{20} \approx 1.000.000$ Zeilen.

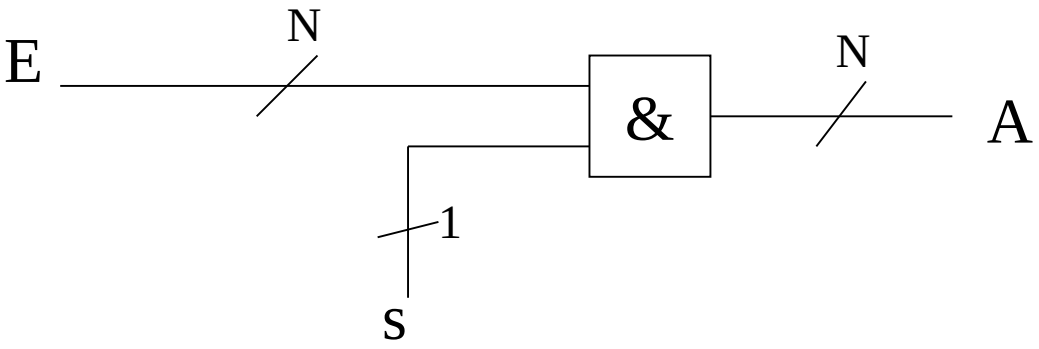
Mögliche Realisiserung:



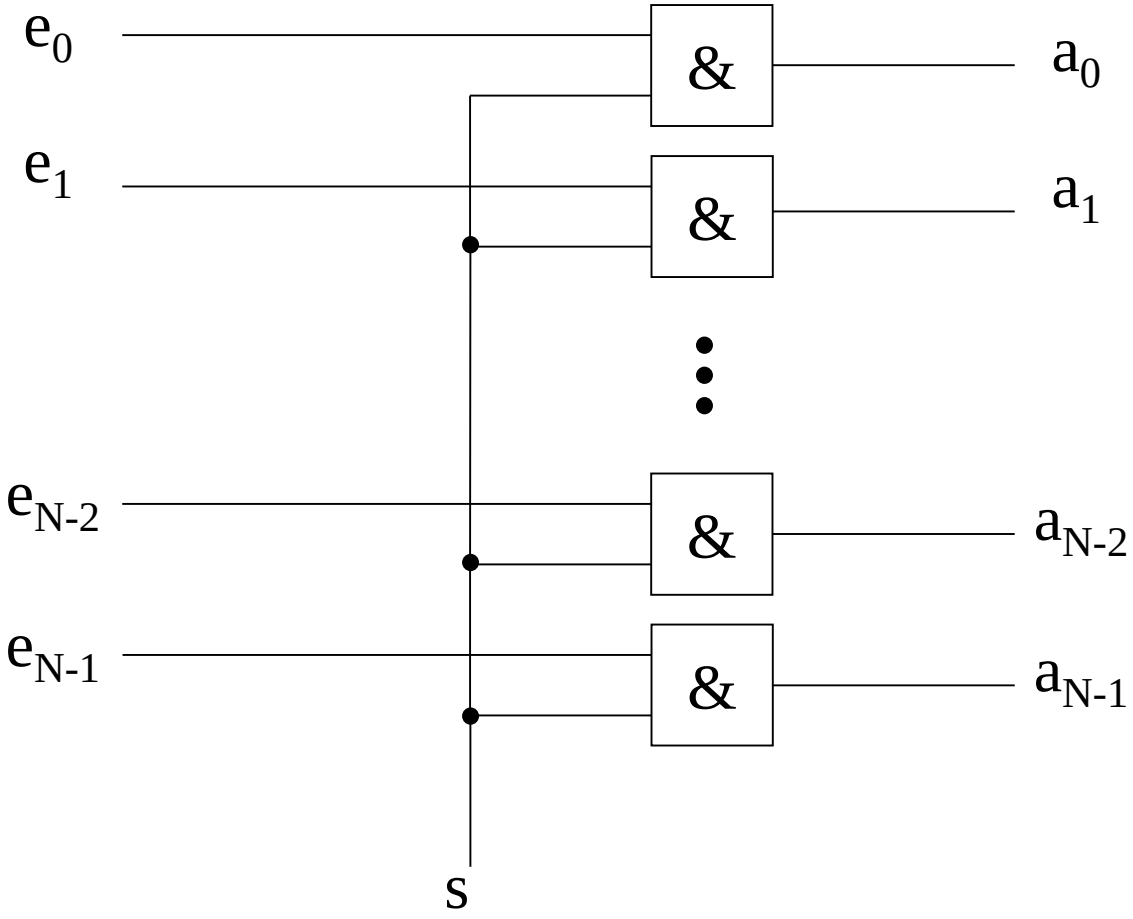
Der Datenwegschalter

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.

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

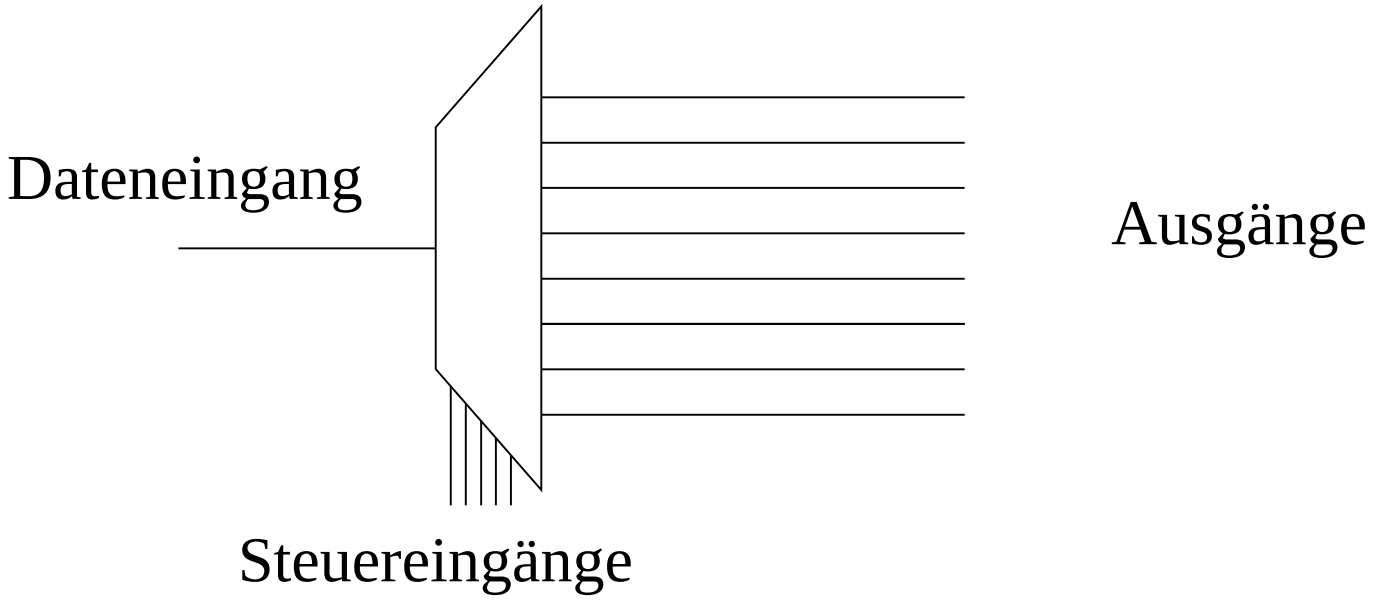


Realisierung des Datenwegschalters



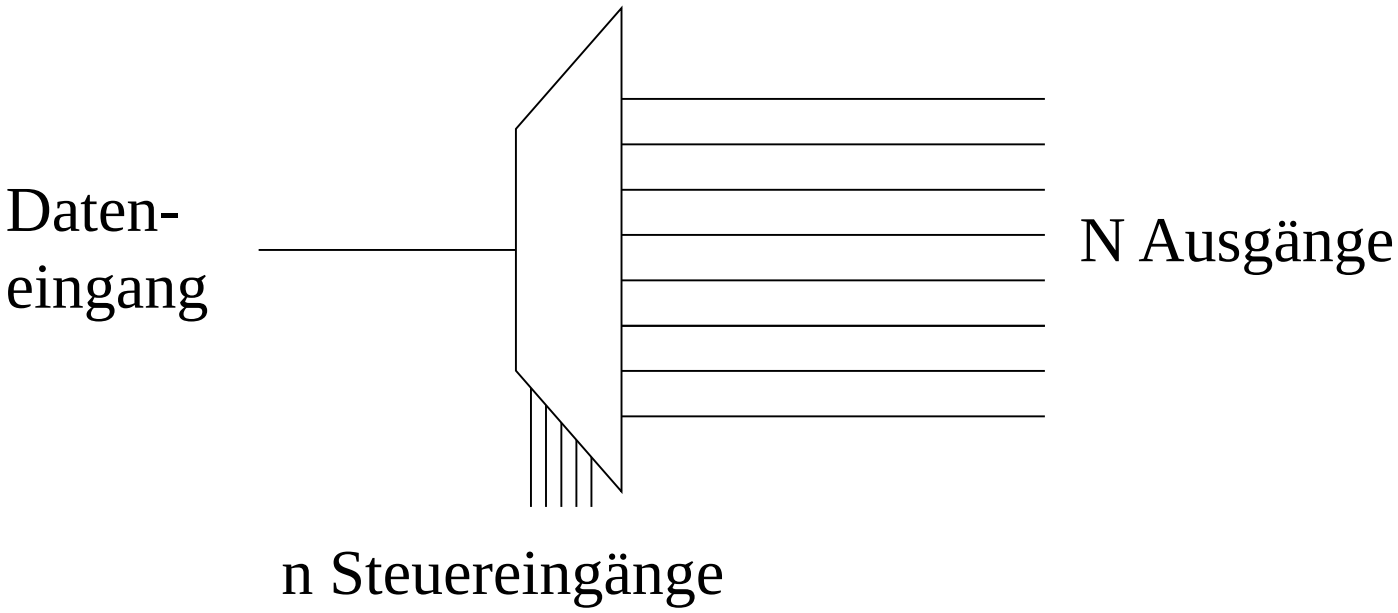
Der Demultiplexer

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.



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

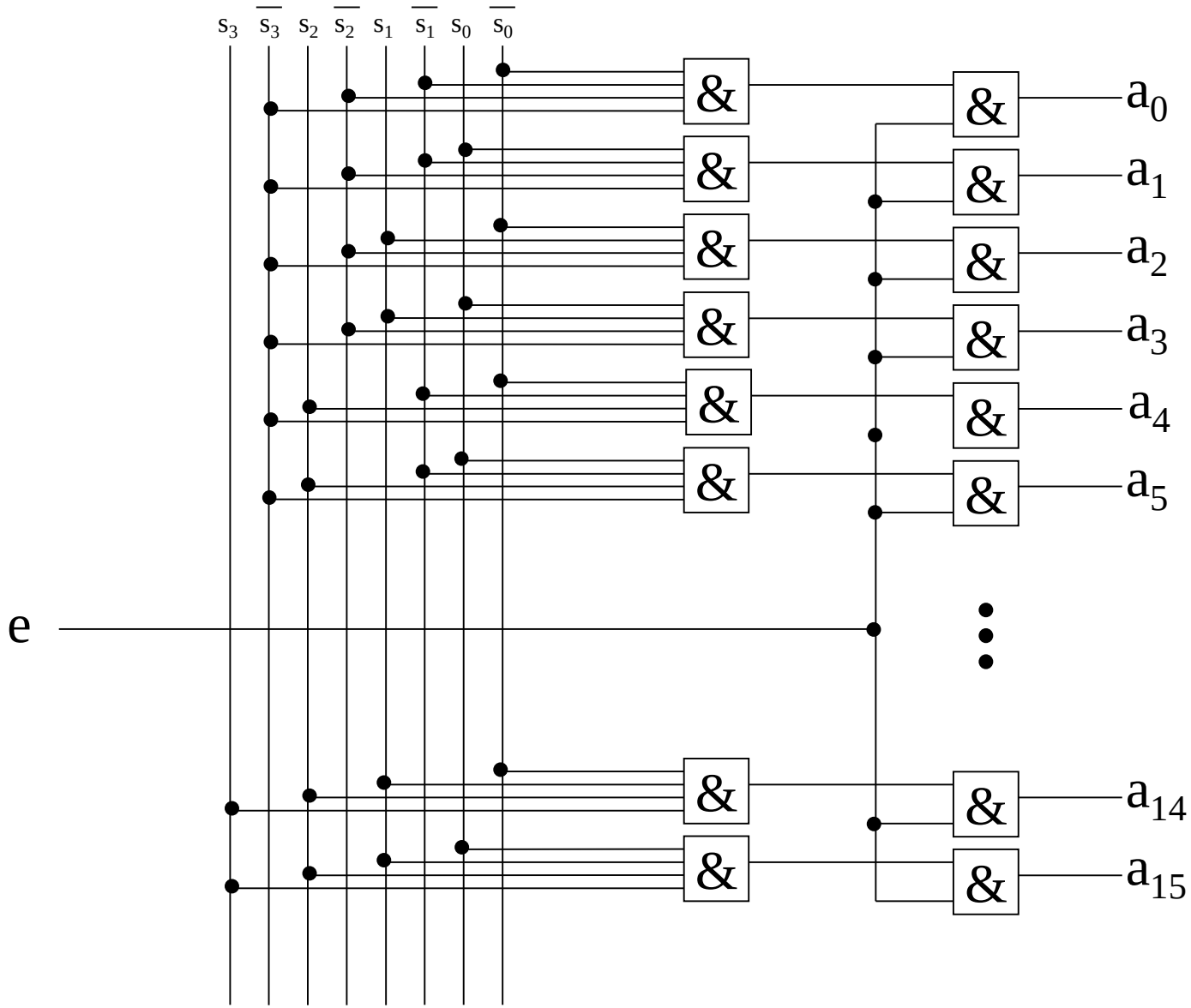
$a_i = e$, falls $(i)_{10}=(s_{n-1}s_{n-2}...s_0)_2$, sonst $a_i = 0$



Wertetabelle für 1-aus-16-Demultiplexer

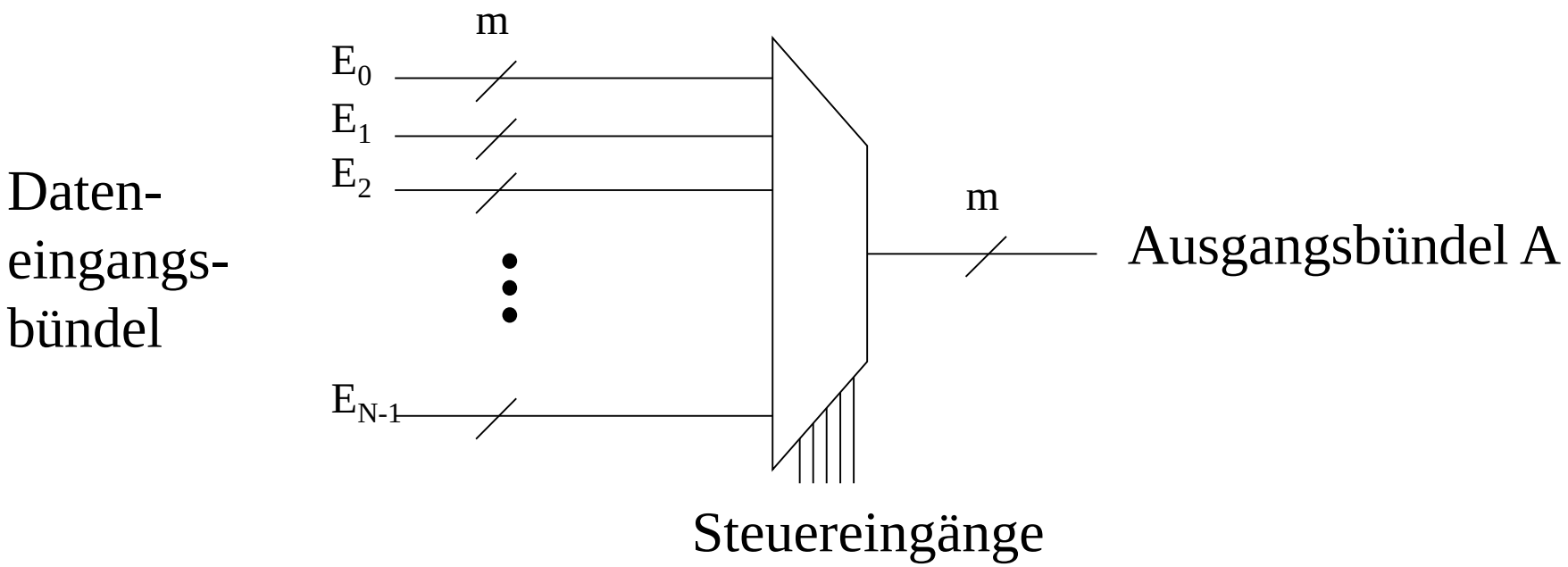
e	s_3	s_2	s_1	s_0	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
0	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0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

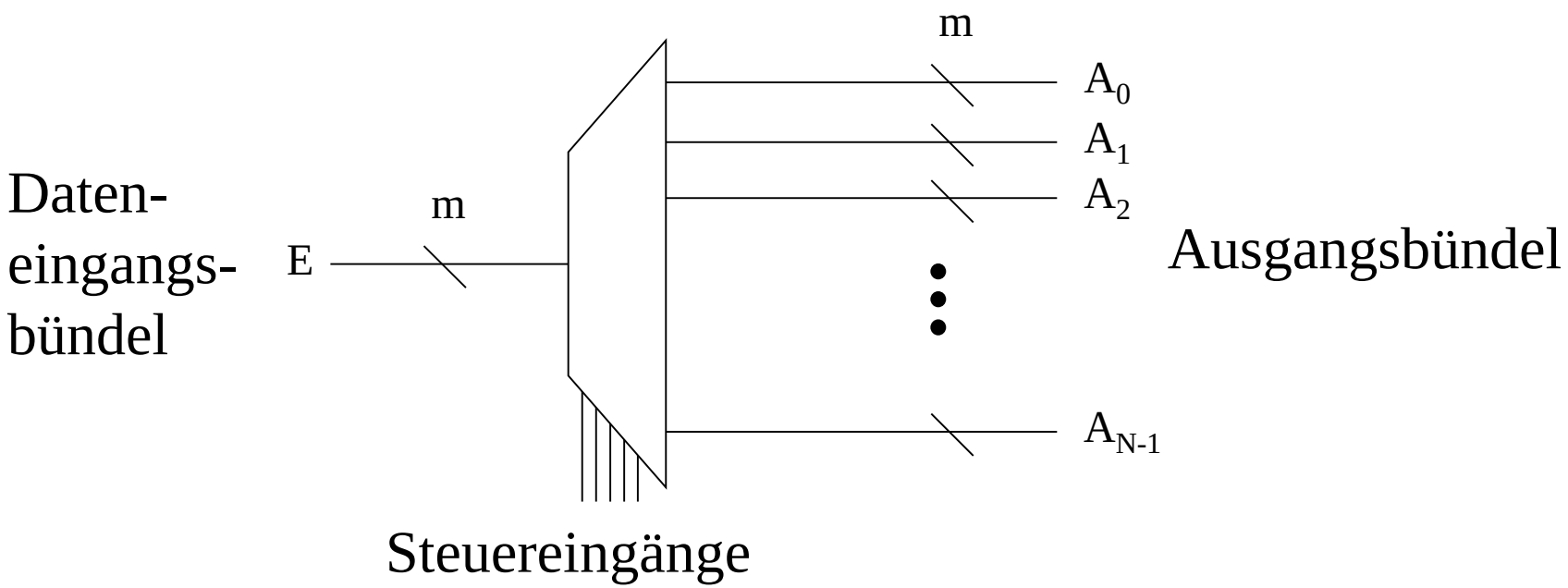
Ein Datenwegmultiplexer ist ein Schaltnetz, das aus m parallel geschalteten Multiplexern besteht. Mit ihm kann man Leitungsbündel der Stärke m schalten. Alle parallelen Leitungen werden durch die Steuereingänge auf das selbe Ausgangsbündel geschaltet.



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

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.

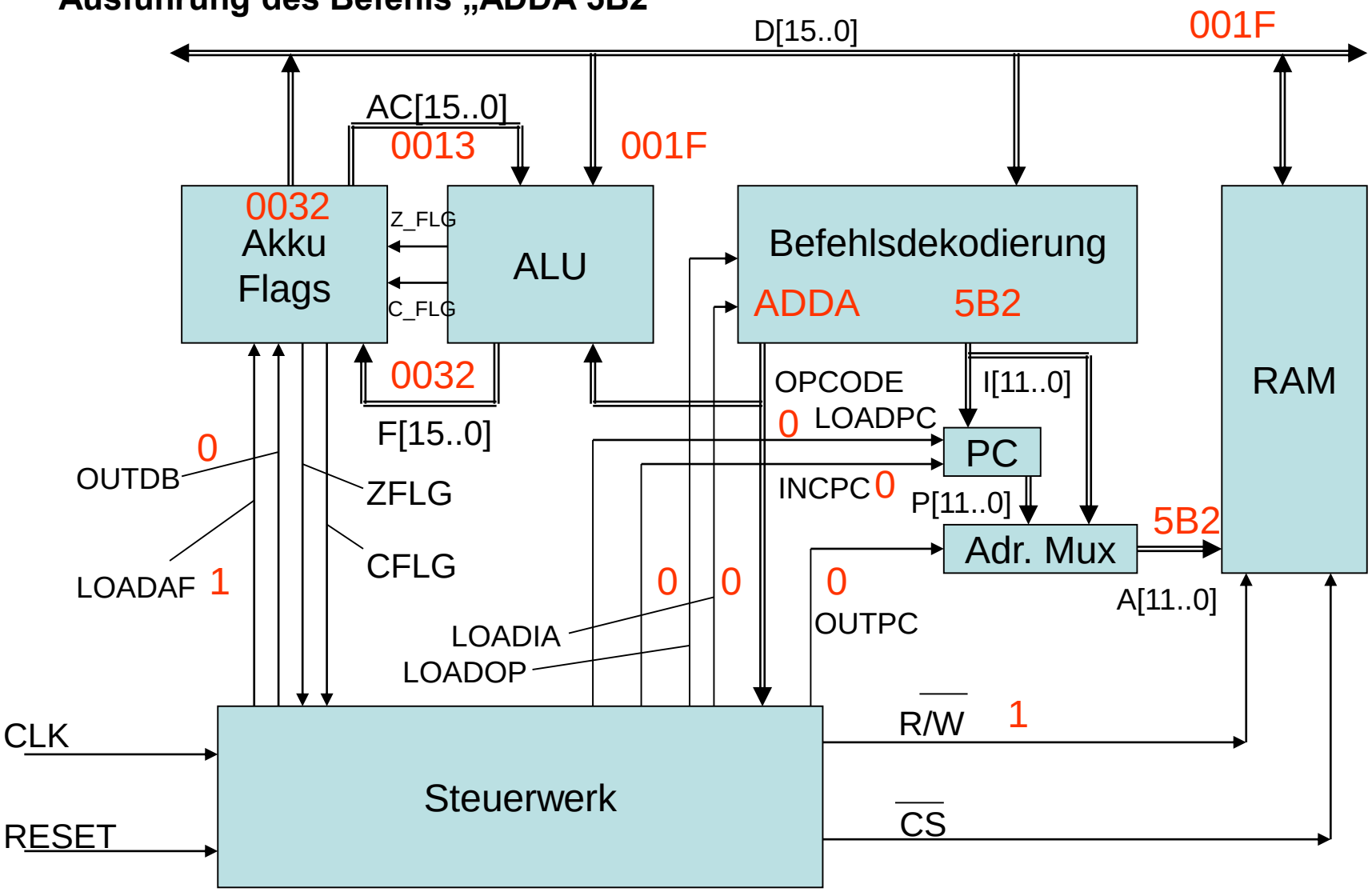


Für alle i=0,...,m-1 gilt $a_{k,i} = e_i$ genau dann wenn $(k)_{10} = (s_{n-1}...s_0)_2$
alle anderen $a_{k,i}$ sind 0.

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 $i \neq N-1$

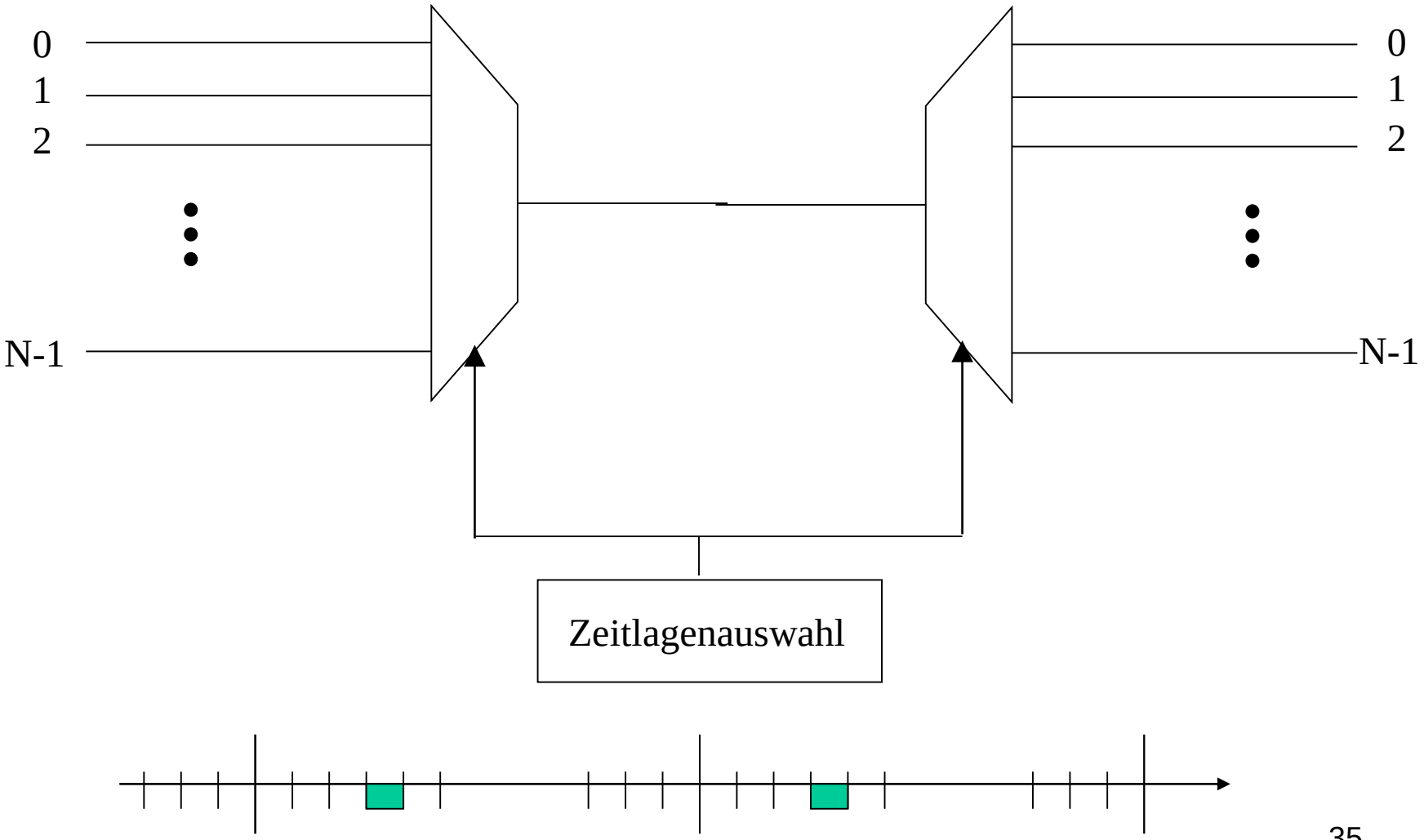
Ausführung des Befehls „ADDA 5B2“



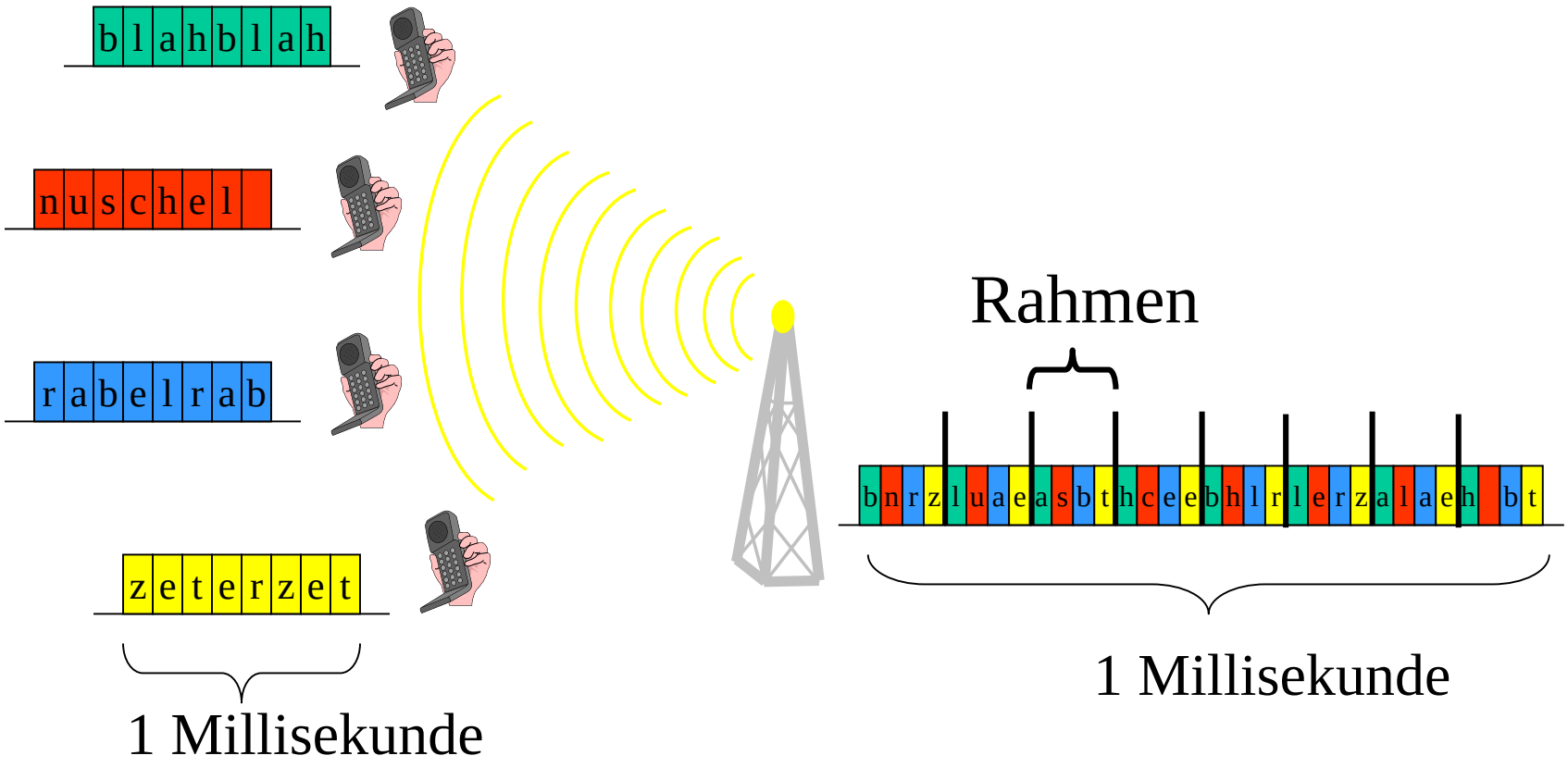
Zeitmultiplexübertragung

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.

Zeitmultiplextechnik (Time Division Multiplex Access, TDMA)

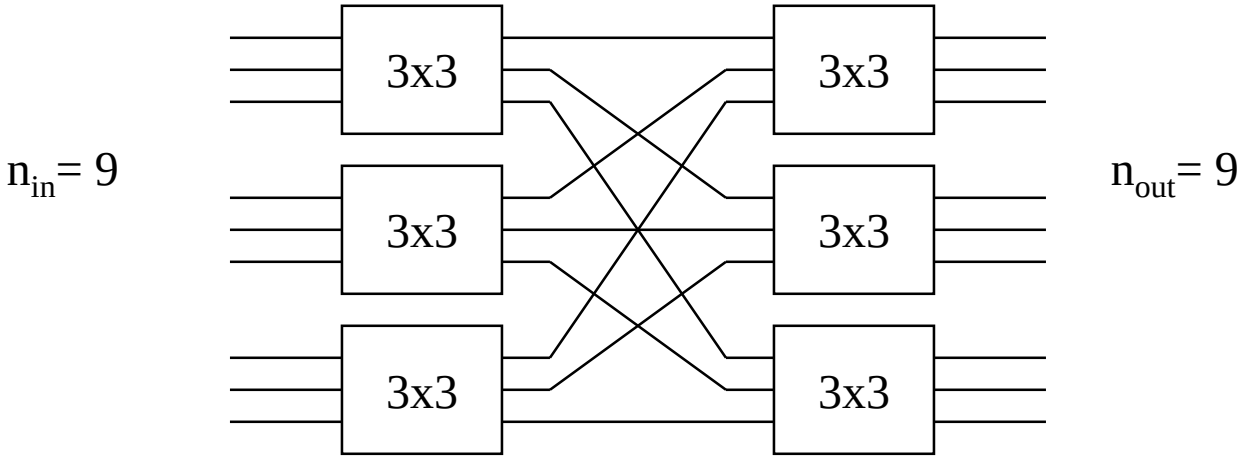


Beispiel für Zeit-Multiplex: GSM-Netz

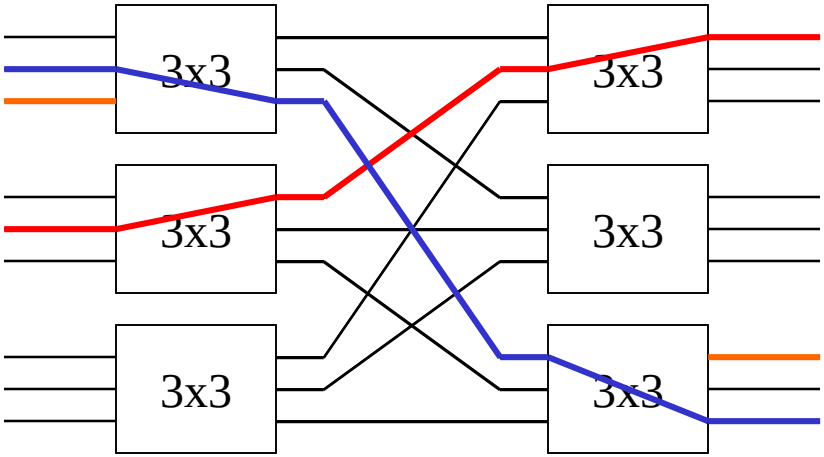


Definition:
A **p-stage switch** is a switching device where the incoming wires and the outgoing wires can be connected via p stages of switching matrices. The incoming wires are inputs to the first stage and the outgoing wires are ouputs of the last (p^{th}) stage. An output of any switching matrix of the i^{th} stage is connected by exactly one wire to the input of any switching matrix of the $(i+1)^{th}$ stage (for i = 1, 2, ..., p-1).

Example:
The following shows a 2-stage switch, where every stage consists of three (3x3-) switching matrices:

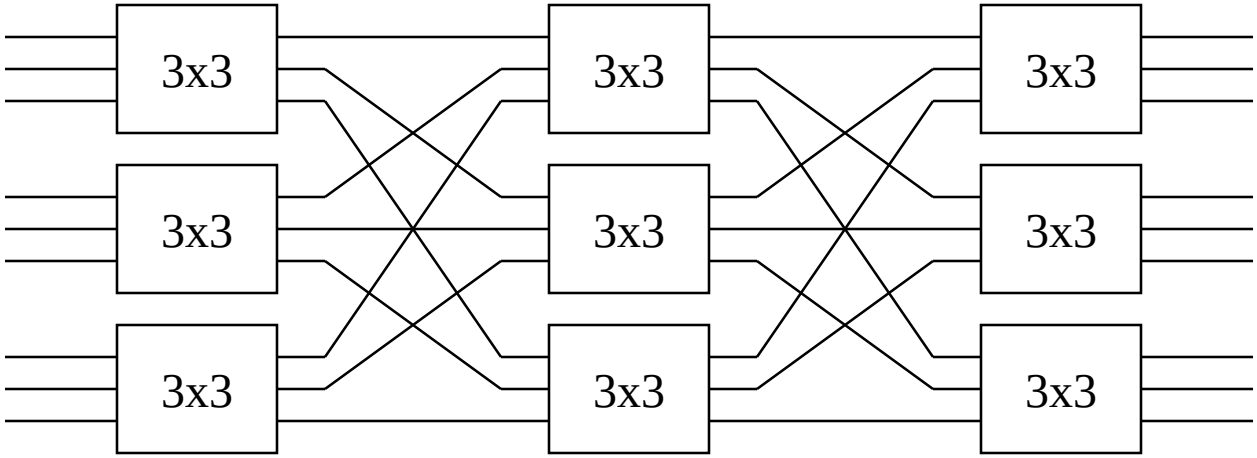


Problem: Inner Blocking



Switch a connection from to

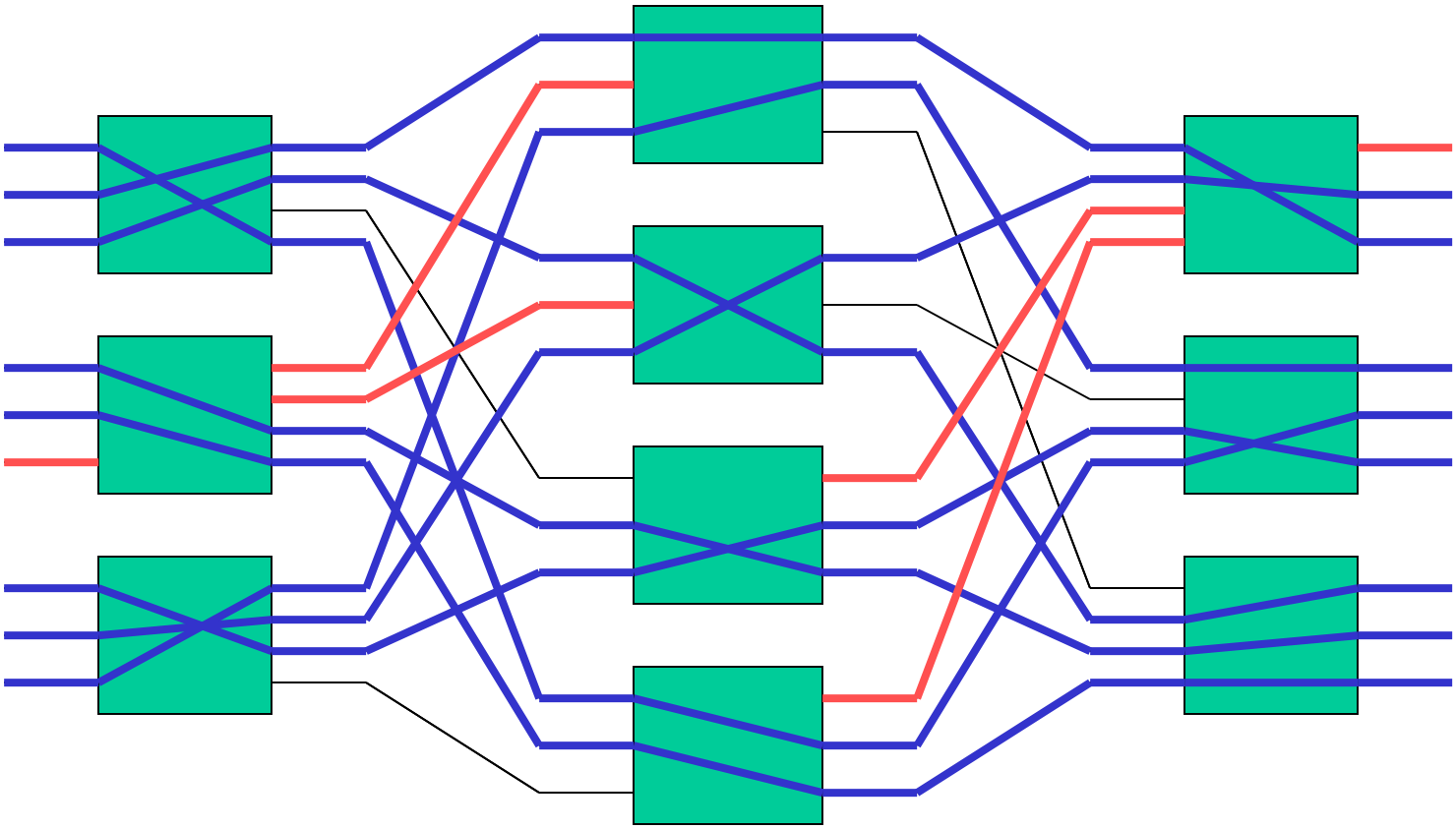
2	9
5	1
3	7



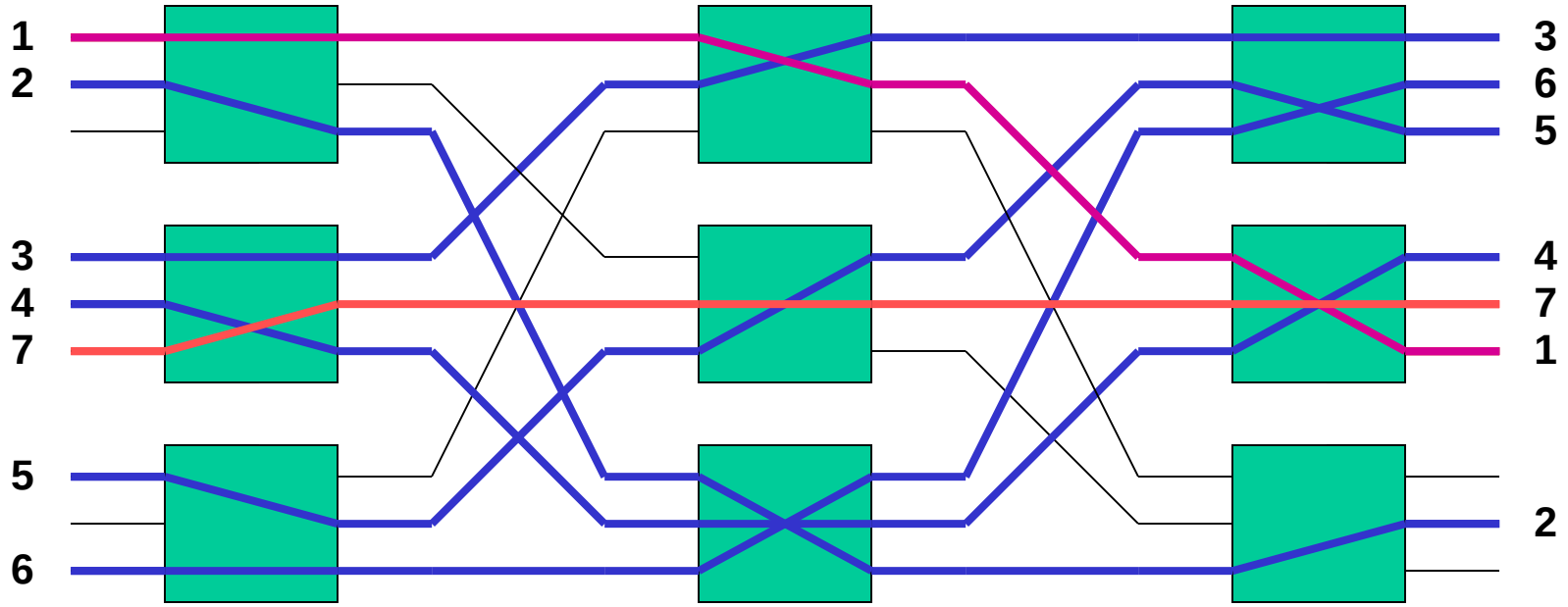
switching demand:

(1,7), (2,5), (4,1), (5,3), (6,9), (8,4)

Two further examples



Rerouting in case of inner blocking



For this purpose it can be necessary to change all previous connections in order to fulfil the extended switching demand. This process is called **reconfiguration**. Therefore, the nonblocking property (freedom of inner blockings) of a switch is also called **reconfigurably nonblocking** or **rearrangably nonblocking**.

The reconfiguration process has an influence on the **quality of service (QoS)**. For real time services this might lead to lost fragments of data, for reliable data services it might require repeated transmission of the same data blocks (ARQ) leading to a smaller effective data rate. Therefore, it is desirable, to keep the number of reconfigurations as low as possible, even if additional hardware for the switching network is required.

This is the reason, why researchers have tried to find solutions for the problem of dynamically fulfilling of switching demands without reconfiguration of existing connections. The most famous result is the theorem of Charles Clos (1953).

Definition:

A switch is called **strictly nonblocking**, if it is

(i) free of inner blockings (nonblocking, reconfigurably nonblocking)

(ii) for each fulfilled switching demand $\{(i_1,o_1), (i_2,o_2),..., (i_{r-1},o_{r-1})\}$ and every pair (i_r,o_r) of a free input channel and a free output channel there is a fulfilling $\{(i_1,o_1), (i_2,o_2), ..., (i_{r-1},o_{r-1}), (i_r,o_r)$ which is identical to the above one in the pairs $(i_1,o_1), (i_2,o_2),..., (i_{r-1},o_{r-1})$.

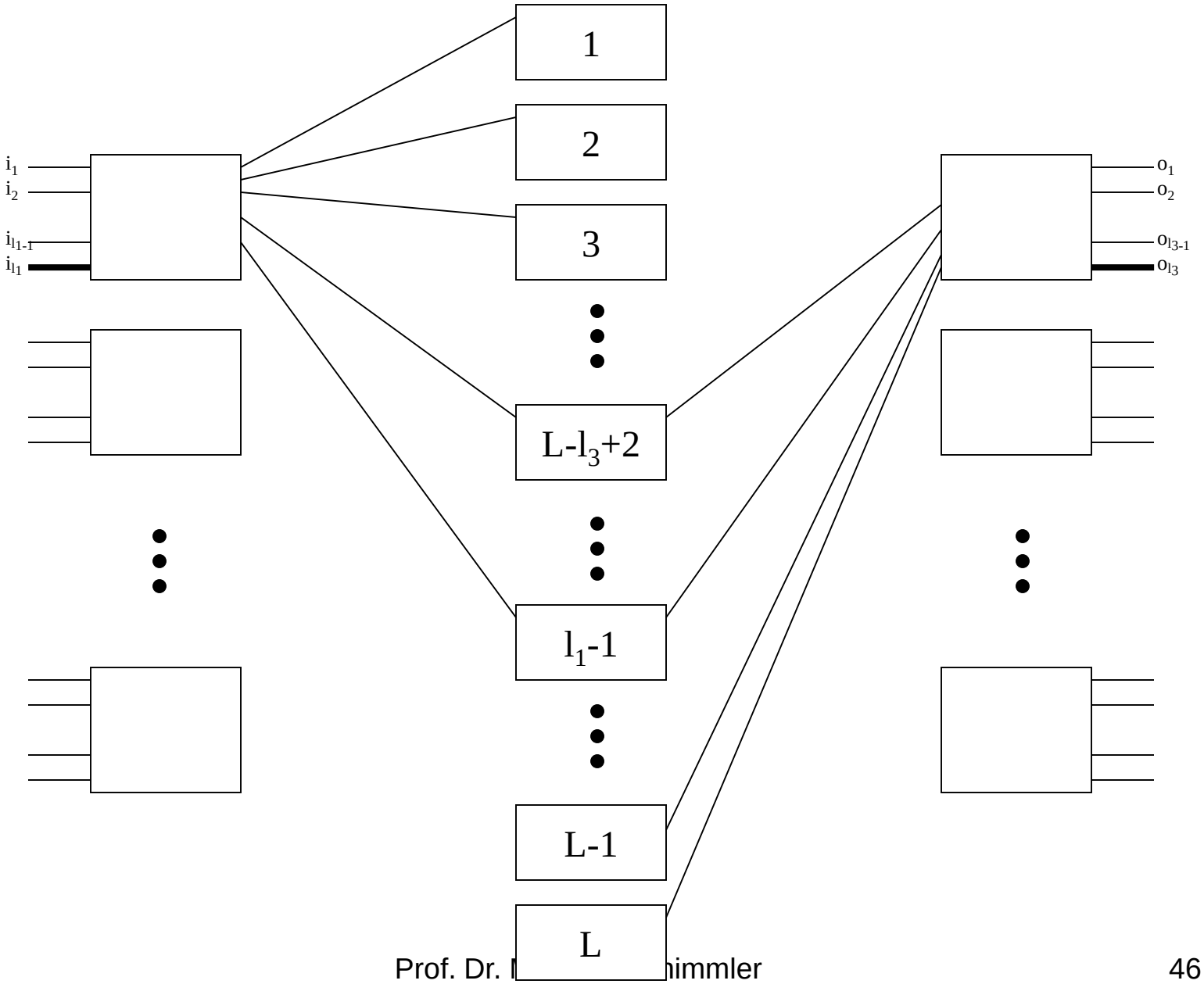
In other words: If there are switched some connections in a the switch every additional switching demand of a free incoming channel and a free outgoing channel can be switched without reconfiguration of the old connections.

Of course, the switching matrix is strictly nonblocking, but Clos could prove that there are strictly nonblocking switches of the same size with a much lower complexity:

Theorem (Charles Clos, 1953):

A three stage switch with n_i incomoing channels and n_o outgoing channels, where l_1 channels go into every switching matrix of the first stage and l_3 channels come out of every switching matrix of the last stage is strictly nonblocking if and only if the number L of switching matrices of the second stage is greater than or equal to l_1+l_3-1.

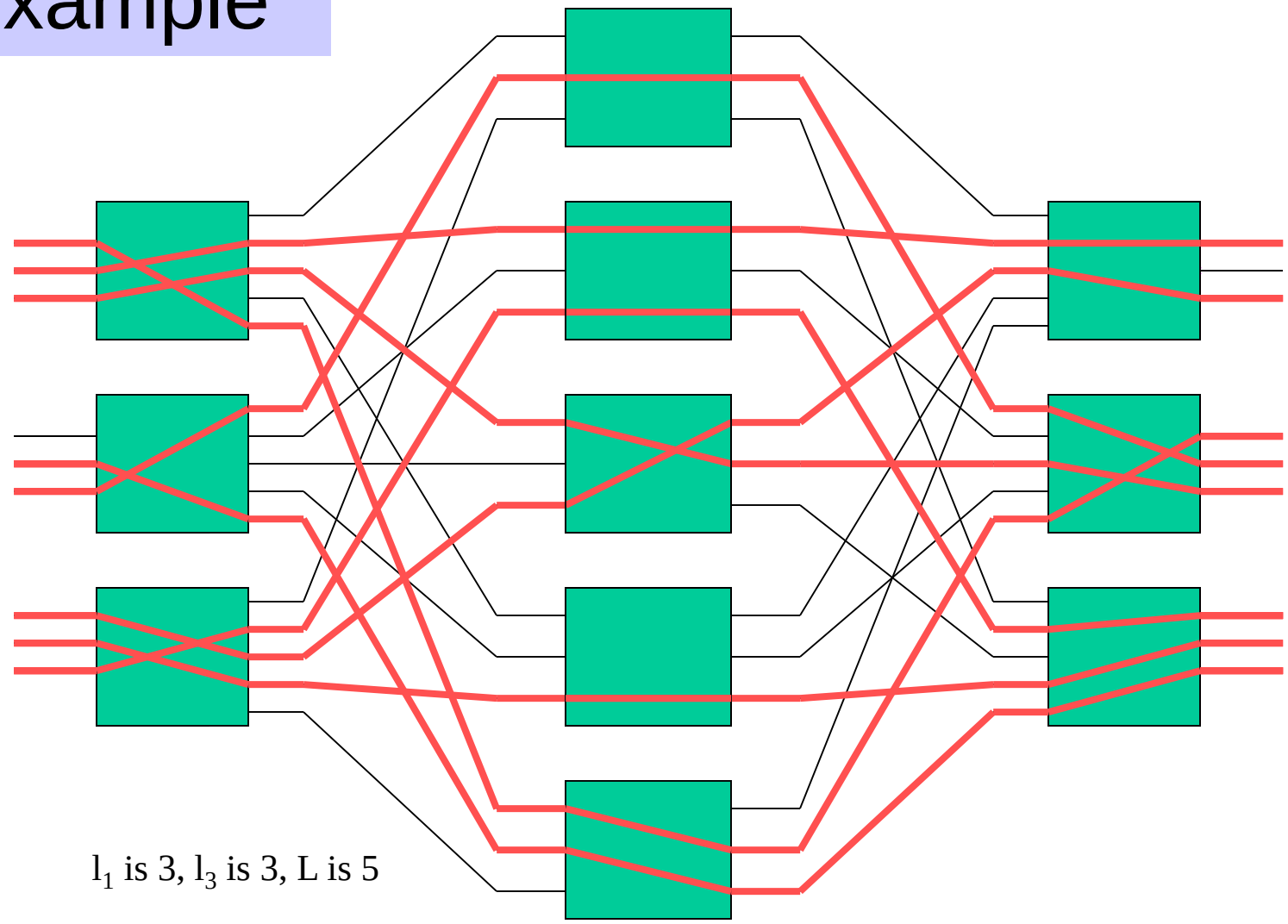
Proof:
„=>“: Proof by contradiction: Assume the number of switching matrices in the second stage is $L < l_1+l_3-1$. Then we can construct an inner blocking in the following way: The incoming channels into the switching matrix of the first stage are denoted by i_1, i_2, ...,i_{l_1}. The outgoing channels of the first switching matrix of the third stage are denoted by o_1, o_2, ..., o_{l_3}. For fulfilling a switching demand i_1 is connected via the first switching matrix of the second stage with an outgoing channel o_p, $p>l_3$. In the same way i_2 via the second switching matrix of the second stage, etc., i.e. i_k via the k^{th} switching matrix of the second stage ($k = 1, 2, ..., l_1-1$). Furthermore, the incoming channels i_p, $p>l_1$ are connected with the outgoing channels o_1, o_2, ..., $o_{l_3}-1$ via the following switching matrices of the second stage: o_1 via the $(L-l_3+2)^{th}$ switching matrix of the second stage, o_2 via the $(L-l_3+3)^{th}$ switching matrix of the second stage, etc., o_{l_3-1} via the L^{th} switching matrix of the second stage.


1
2
3
L-l3+2
l1-1
L-1
L
i1
i2
il1-1
il1
o1
o2
ol3-1
ol3

If now i_{l_1} shall be connected to o_{l_3} then it comes to an inner blocking. From the first switching matrix of the first stage there are only free connections to the switching matrices l_1, l_1+1, ..., L of the second stage. But from these there are no free wires to the first switching matrix of the third stage, since the connections of the matrices $L-l_3+2$, $L-l_3+3$, ..., L of the second stage are already in use.
There is no switching matrix in the second stage with a free wire to the first matrix of the first stage and to the first matrix of the third stage, and i_{l_1} and o_{l_3} cannot be connected.

„<= “: Let $L \geq l_1+l_3-1$. Furthermore, there is an arbitrary state of existing connections in the three-stage switch. Let the incoming channel i_k and the outgoing channel o_k be free. We show: then an additional connection from i_k to o_k can be established.
i_k comes into a switching matrix of the first stage, say K_1. o_k comes out of a switching matrix of the third stage. We all this K_3. Since K_1 has l_1-1 more inputs, only maximally l_1-1 connections from K_1 to the switching matrices of the second stage can be in use. There are L such connections, and $L \geq l_1+l_3-1$. Therefore, at least l_3 of them are still free. In other words: there are free wires from K_1 to l_3 switching matrices of the second stage.
Except from o_k another l_3-1 outgoing channels come out of K_3. These can occupy maximally l_3-1 wires from switching matrices of the second stage to K_3. Thus, there must be at least one switching matrix of the second stage with a free wire to K_1 as well as to K_3. This can be used to connect i_k and o_k.

Example



l_1 is 3, l_3 is 3, L is 5

Example