HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

Technische Grundlagen der Informatik

Schaltnetze

Prof. Dr.-Ing. Benjamin Weinert

Gliederung

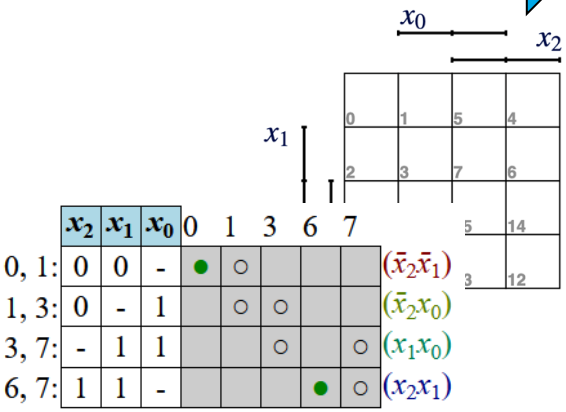
- Multiplexer / Demultiplexer
- Arithmetische Schaltungen
  - Addition
  - Inkrement
  - Subtraktion
  - Multiplikation

Motivation

Modellieren

Index i	a	b	c	y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Minimieren



$\overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_2}\,x_3 \vee \overline{x_1}\,\overline{x_2}\,x_3 \vee \overline{x_1}\,x_2\,\overline{x_3} \vee x_1\,x_2\,\overline{x_3}$

$= \overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_2}\,x_3 \vee \overline{x_1}\,\overline{x_2}\,x_3 \vee x_2\,\overline{x_3}$

$= \overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_2}\,\overline{x_3} \vee \overline{x_2}\,x_3 \vee x_2\,\overline{x_3}$

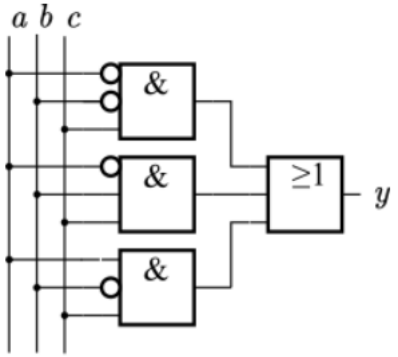
$= \overline{x_2}\,\overline{x_3} \vee \overline{x_2}\,x_3 \vee x_2\,\overline{x_3}$

$= \overline{x_2}\,\overline{x_3} \vee \overline{x_2}\,x_3 \vee \overline{x_2}\,\overline{x_3} \vee x_2\,\overline{x_3}$

$= \overline{x_2} \vee \overline{x_2}\,\overline{x_3} \vee x_2\,\overline{x_3}$

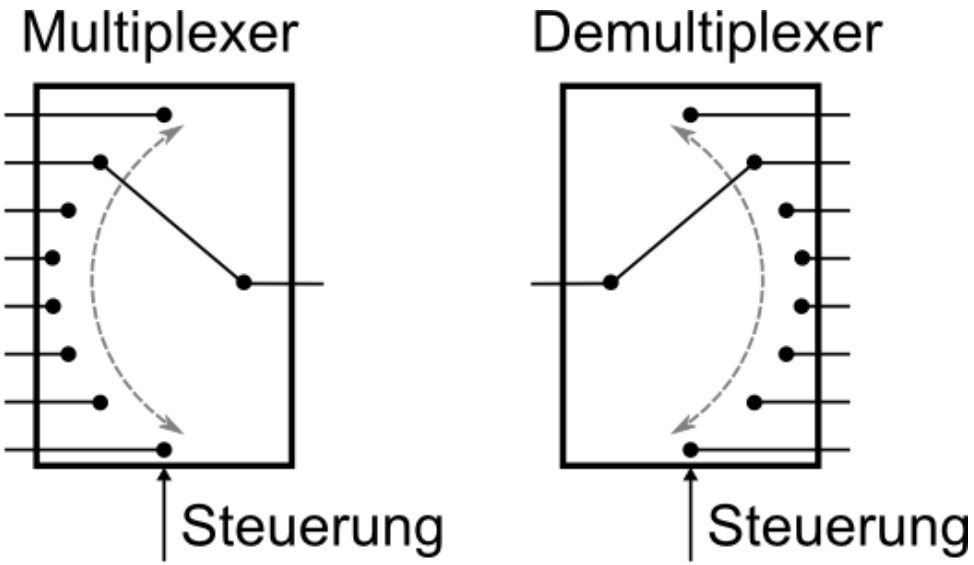
$= \overline{x_2} \vee \overline{x_3}$

Synthetisieren



Multiplexer/Demultiplexer

- Ein Multiplexer (auch "Mux") schaltet von vielen Eingängen auf einen Ausgang
- Ein Demultiplexer (auch "Demux") schaltet von einem Eingang auf viele Ausgänge

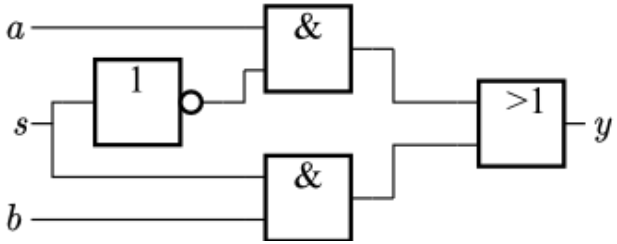


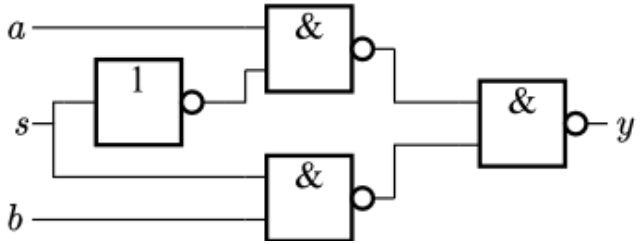
Multiplexer

- Für einen 2-zu-1 Multiplexer (auch "2:1 Mux") gilt:
  - Wenn das Steuersignal **s=1**,
    - dann ist y=b,
    - ansonsten y=a
- Diese entspricht einer if-then-else-Anweisung

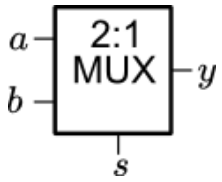
```
if (s) then y = b; else y = a;
```

- Ein 2-zu-1 Multiplexer kann leicht aus mehreren Logikgattern aufgebaut werden:



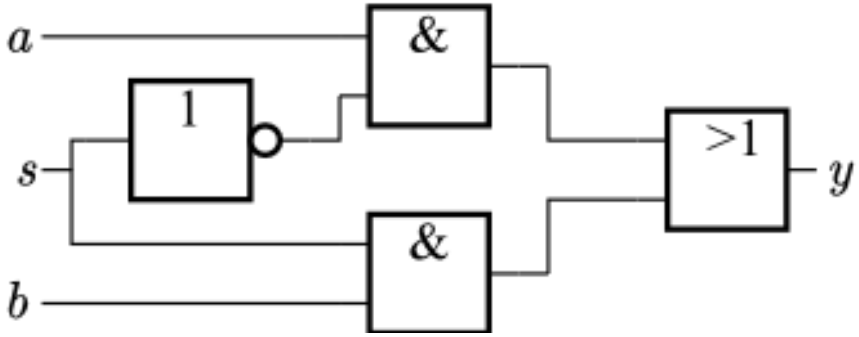


- Da Multiplexer häufig Verwendung finden, gibt es ein spezielles Symbol



Wahrheitstabelle 2-zu-1 Mux

Index i	a	b	s	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1



Multiplexer

- 2-zu-1-Multiplexer mit n=2 Dateneingängen

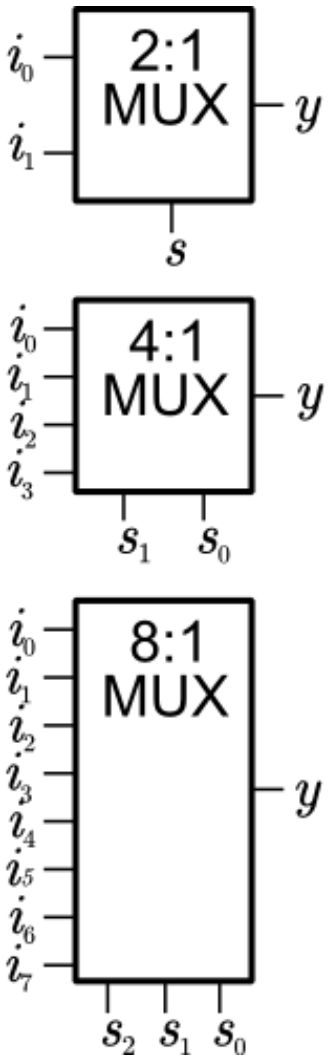
$$y = \bar{s}_0 i_0 \vee s_0 i_1$$

- 4-zu-1-Multiplexer mit n=4 Dateneingängen

$$y = \bar{s}_1 \bar{s}_0 i_0 \vee \bar{s}_1 s_0 i_1 \vee s_1 \bar{s}_0 i_2 \vee s_1 s_0 i_3$$

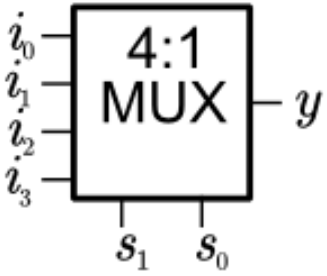
- 8-zu-1-Multiplexer mit n=8 Dateneingängen

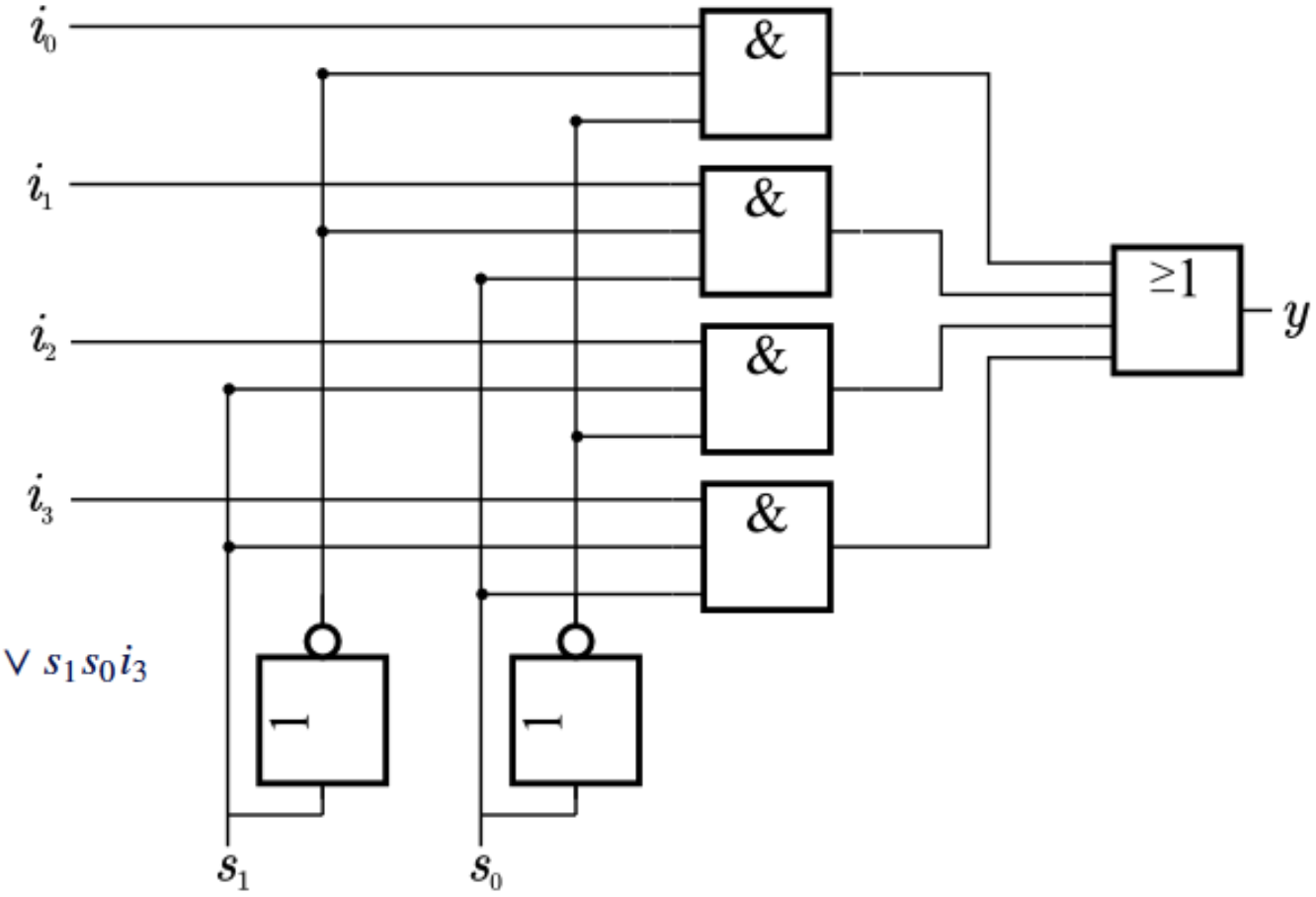
$$y = \bar{s}_2 \bar{s}_1 \bar{s}_0 i_0 \vee \bar{s}_2 \bar{s}_1 s_0 i_1 \vee \bar{s}_2 s_1 \bar{s}_0 i_2 \vee \bar{s}_2 s_1 s_0 i_3$$
$$s_2 \bar{s}_1 \bar{s}_0 i_4 \vee s_2 \bar{s}_1 s_0 i_5 \vee s_2 s_1 \bar{s}_0 i_6 \vee s_2 s_1 s_0 i_7$$



Multiplexer - Realisierung

- 4:1 Mux als zweistufige Logik

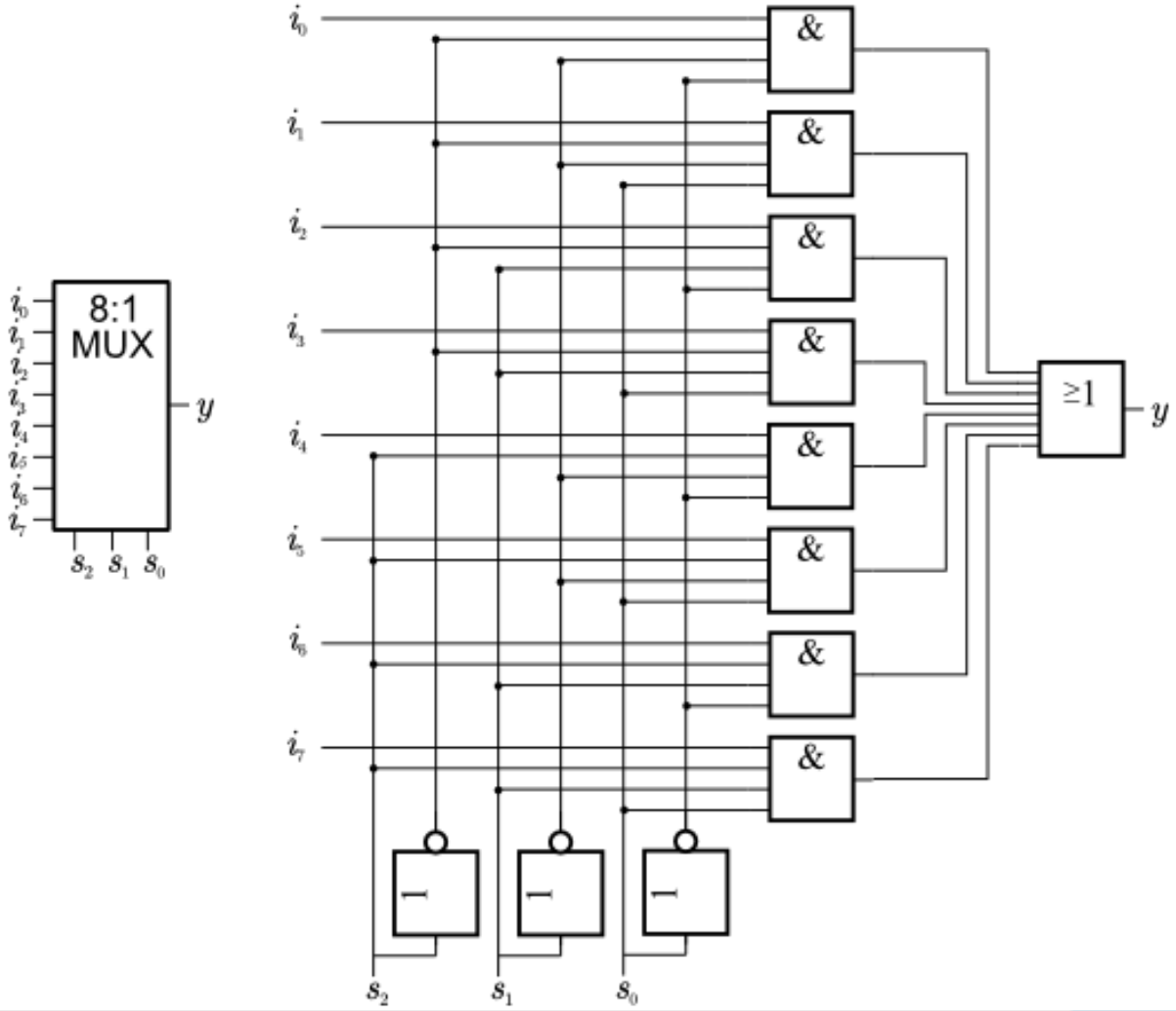




$$y = \bar{s}_1 \bar{s}_0 i_0 \vee \bar{s}_1 s_0 i_1 \vee s_1 \bar{s}_0 i_2 \vee s_1 s_0 i_3$$

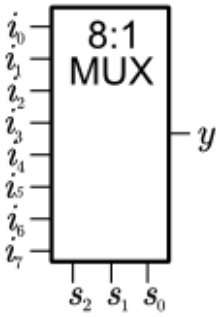
Multiplexer - Realisierung

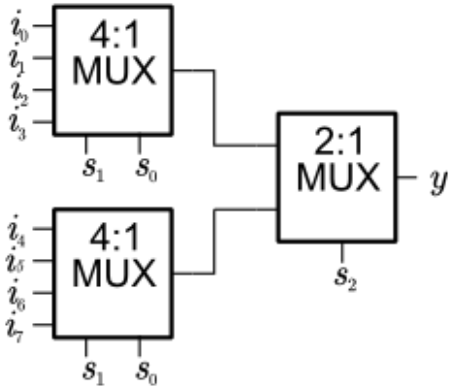
- Ein 8:1 Mux als zweistufige Logik

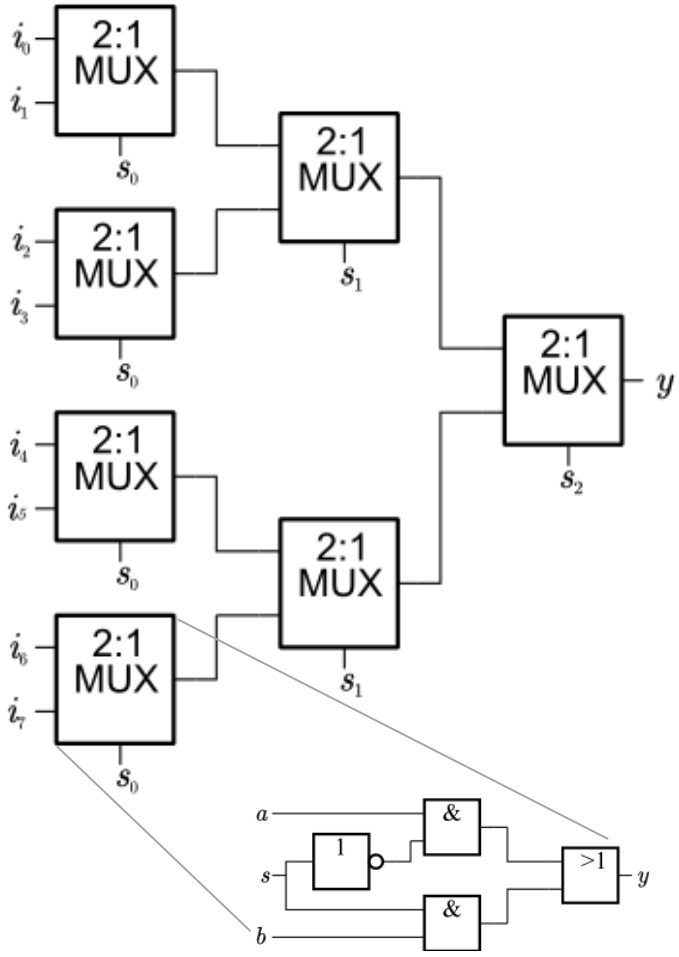


Kaskadierung von Multiplexern

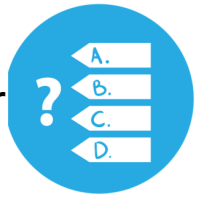
- Große Multiplexer können durch Kaskadierung von kleinen Multiplexern aufgebaut werden





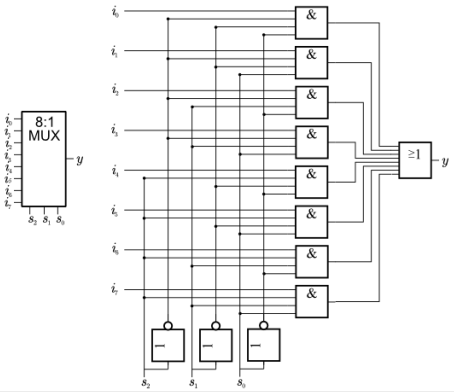


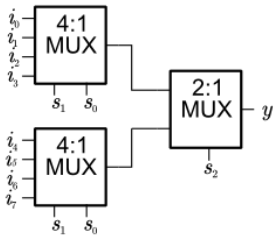
- Was ist sinnvoller
  - Kaskadierung oder
  - zweistufige Logik?

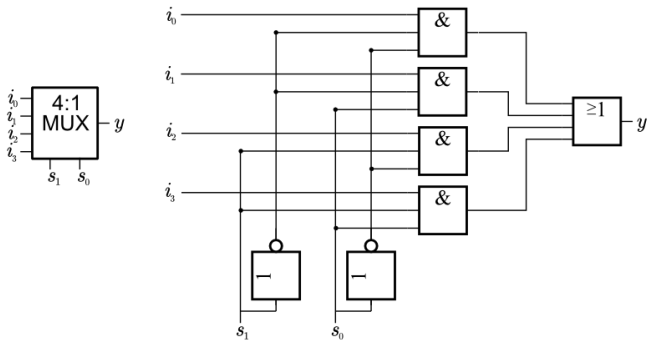


Kaskadierung von Multiplexern

- Kaskadierung oder zweistufige Logik ist abhängig vom Optimierungsziel
  - Bei zweistufiger Logik entstehen kleinere Gatterlaufzeiten
  - Eine Kaskadierung benötigt insgesamt weniger Eingänge
    - kann daher auf kleinerer Chipfläche realisiert werden
- Beispiel
  - zweistufige Logik benötigt UND-Gatter mit je $\log_2(n) + 1$ Eingängen
  - kaskadierte Lösung benötigt mehr UND-Gatter, diese haben aber **weniger Eingänge**
  - 8:1-Mux
    - zweistufig: 8 Gatter mit je *($log_2(8) + 1 = 4$)* vier Eingängen = 32 Eingänge
    - kaskadiert: 2 * 4 Gatter mit je 3 Eingängen + 2 Gatter mit 2 Eingängen = 28 Eingänge
    - Flächenaufwand proportional zur Anzahl der Eingänge
      ➔ kaskadierte Lösung hat geringeren Flächenverbrauch







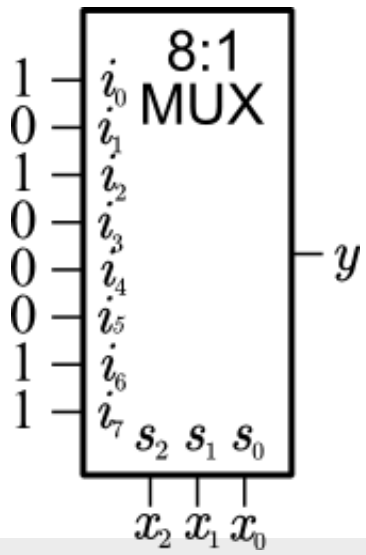
Multiplexer als universelle Logikbausteine

- Multiplexer sind nicht nur zur Steuerung von Datenflüssen, sondern auch zur Realisierung logischer Funktionen verwendbar
- n-zu-1-Multiplexer können jede Funktion mit $\log_2(n)$-Variablen implementieren
  - Die **Variablen** $x_{k-1}, ..., x_0$ werden als Steuersignale $s_{k-1}, ..., s_0$ verwendet
  - Die Dateneingänge $i_{n-1}, ..., i_0$ werden auf 0 oder 1 gelegt
- Beispiel $y = m_0 \vee m_2 \vee m_6 \vee m_7$

$$\bar{x}_2\bar{x}_1\bar{x}_0 \vee \bar{x}_2 x_1 \bar{x}_0 \vee x_2 x_1 \bar{x}_0 \vee x_2 x_1 x_0$$

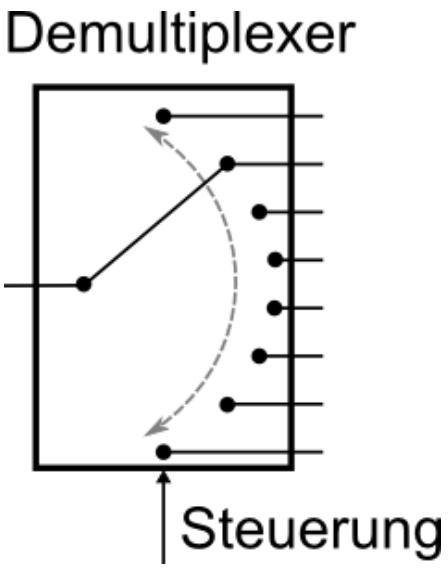
Wahrheitstafel:

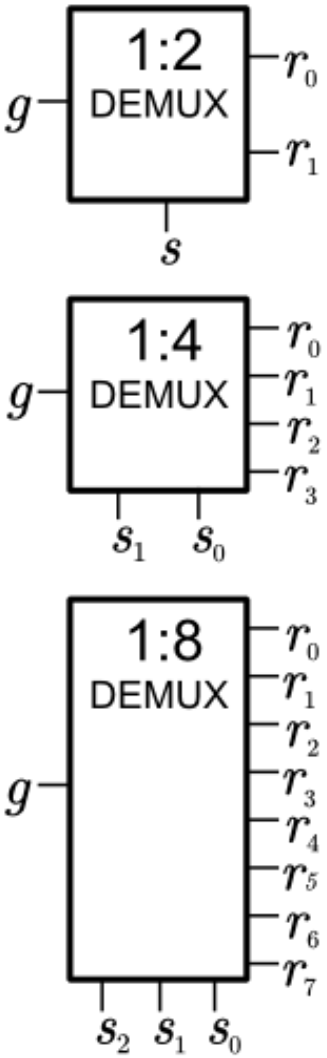
	x_2	x_1	x_0	y
0:	0	0	0	1
1:	0	0	1	0
2:	0	1	0	1
3:	0	1	1	0
4:	1	0	0	0
5:	1	0	1	0
6:	1	1	0	1
7:	1	1	1	1



Demultiplexer

- Der Demultiplexer stellt quasi die inverse Funktion zum Multiplexer dar
  - 1 Eingang g, k Steuereingänge s_{k-1},.., s_0 und n = 2^k Ausgänge r_{n-1}, ..., r_0
  - Der Eingang g wird häufig "Enabler" genannt
  - Abhängig von den Steuereingängen
    - wird der Eingang auf einen der Ausgänge geschaltet
    - die anderen Ausgänge sind 0





Demultiplexer

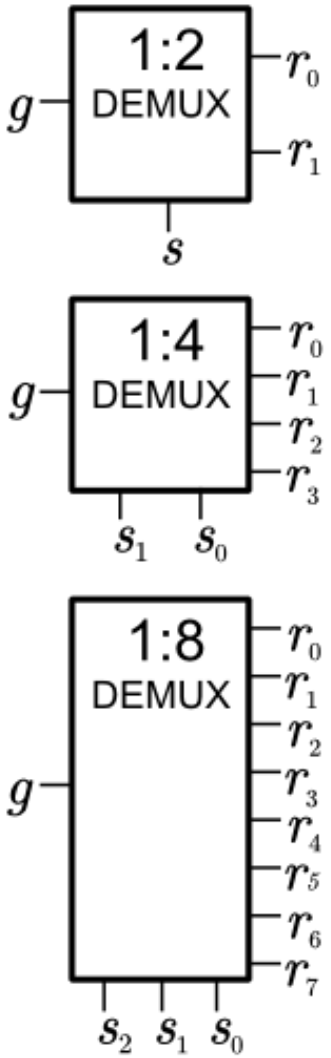
- 1-zu-2 Demultiplexer

$$r_0 = g\,\bar{s}$$
$$r_1 = g\,s$$

- 1-zu-4 Demultiplexer

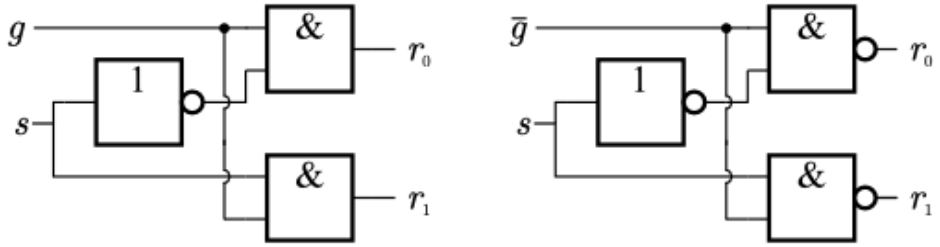
$$r_0 = g\,\bar{s}_1\,\bar{s}_0$$
$$r_1 = g\,\bar{s}_1\,s_0$$
$$r_2 = g\,s_1\,\bar{s}_0$$
$$r_3 = g\,s_1\,s_0$$

- ...

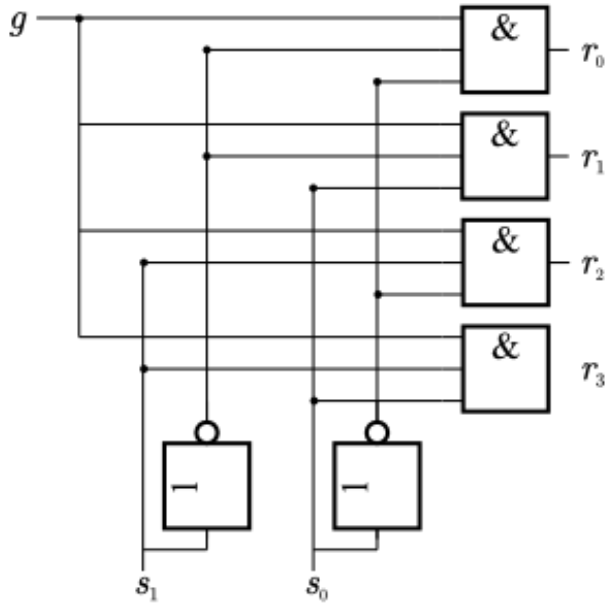


Realisierung von Demultiplexern

- 1-zu-2 Demultiplexer

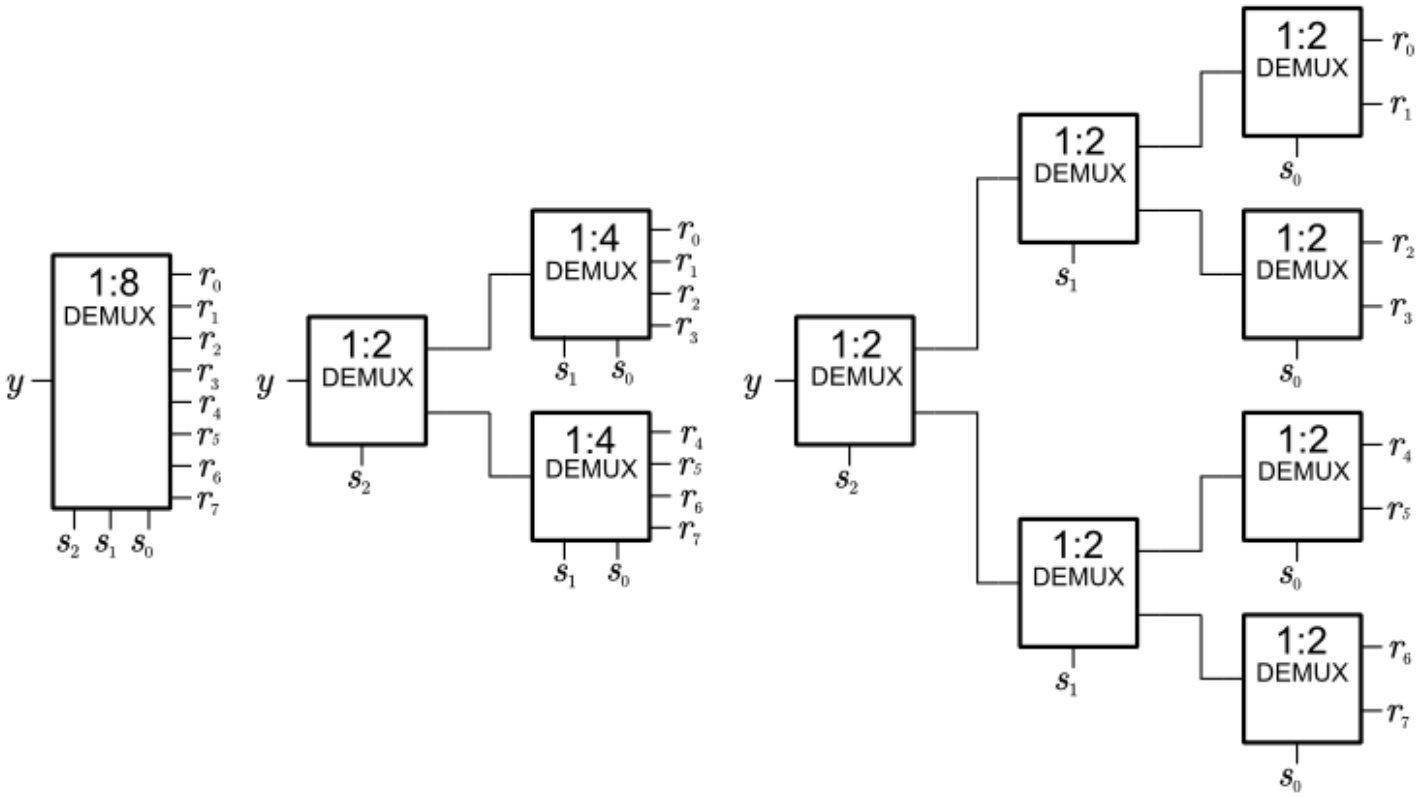


- 1-zu-4 Demultiplexer



Kaskadierung von Demultiplexern

- Große Demultiplexer können durch Kaskadierung von kleinen Demultiplexern aufgebaut werden



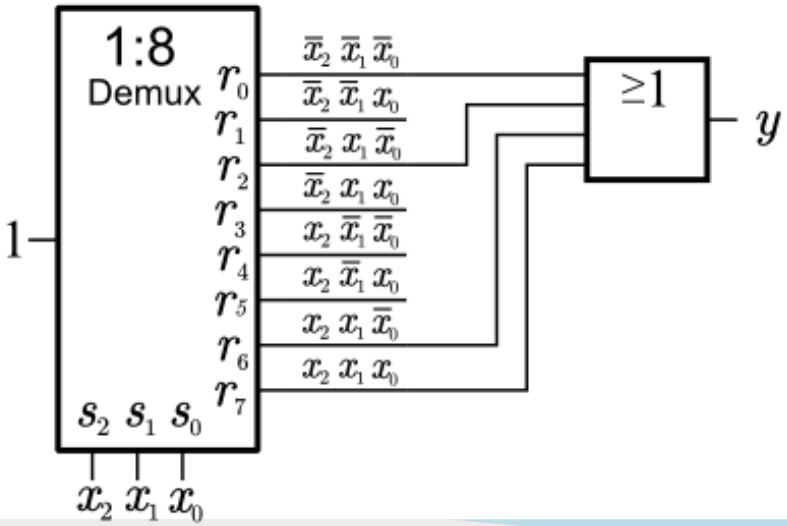
Demultiplexer als universelle Logikbausteine

- 1-zu- Demultiplexer können jede Funktion von k = $\log_2(n)$ Variablen implementieren
  - Die Variablen x_{k-1}, ..., x_0 werden als Steuervariablen s_{k-1}, ..., s_0 verwendet
  - Der Eingang wird auf 1 gelegt
  - An den Ausgängen liegen die Minterme der Steuervariablen an und können durch ein ODER-Gatter verknüpft werden, um die Funktion zu implementieren
- Beispiel

$y = m_0 \vee m_2 \vee m_6 \vee m_7$

Wahrheitstafel:

	x_2	x_1	x_0	y
0:	0	0	0	1
1:	0	0	1	0
2:	0	1	0	1
3:	0	1	1	0
4:	1	0	0	0
5:	1	0	1	0
6:	1	1	0	1
7:	1	1	1	1

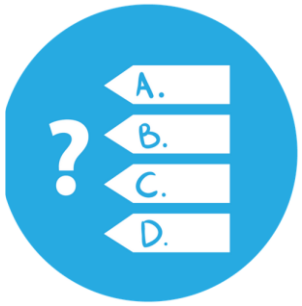


Addition

- Zur Entwicklung einer Schaltung für die Addition kann man sich an der schriftlichen Addition orientieren

```
  13       1101
+  5     + 0101
----     ------
  18      10010
```

- Die Addition wird Bit-weise von rechts nach links durchgeführt (angefangen beim niederwertigsten Bit)



- Warum ist eine Schaltung für die Addition so extrem wichtig?

Halbaddierer

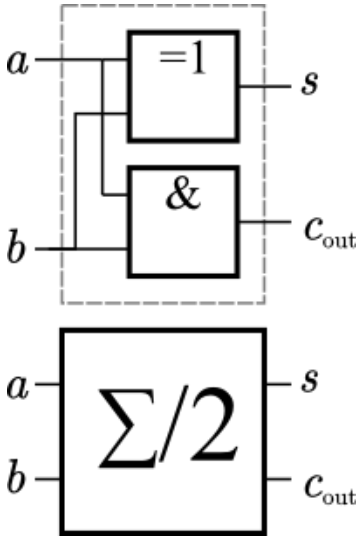
- Für die Addition zweier Dualzahlen
  - Summe und Übertrag entstehen als Ergebnis
- Beim niederwertigsten Bit müssen **nur** alle Kombinationen der zwei Summanden a und b betrachtet werden (4 Fälle)
  - ohne Übertrag
- **Wahrheitstabelle** von Eingangsvariablen a und b sowie das Ergebnis der Addition s und Übertrag c_{out} (Carry-out)

a	b	s	c_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

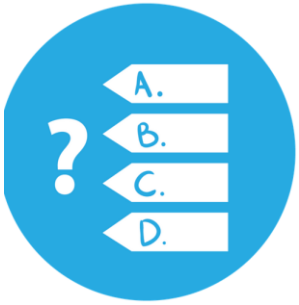
- Damit ergibt sich $s = \bar{a}b \vee a\bar{b} = a \nleftrightarrow b$

$$c_{out} = ab$$

- Die Realisierung mit Logikgattern wird Halbaddierer genannt und erhält ein eigenes Symbol (siehe rechts, 1-Bit-Halbaddierer)



Halbaddierer



- Warum heißt es eigentlich „Halbaddierer“? Was fehlt?