Technische Informatik
Reguläre Logikschaltungen

Thorsten Thormählen

30. November 2021

Teil 7, Kapitel 1

Dies ist die Druck-Ansicht.

Aktiviere Präsentationsansicht

Steuerungstasten

→ nächste Folie (auch Enter oder Spacebar).
← vorherige Folie
d schaltet das Zeichnen auf Folien ein/aus
p wechselt zwischen Druck- und Präsentationsansicht
CTRL + vergrößert die Folien
CTRL - verkleinert die Folien
CTRL 0 setzt die Größenänderung zurück

Notation

Typ	Schriftart	Beispiele
Variablen (Skalare)	kursiv	a, b, x, y
Funktionen	aufrecht	$\mathrm{f}, \mathrm{g}(x), \max(x)$
Vektoren	fett, Elemente zeilenweise	$\mathbf{a}, \mathbf{b} = \begin{pmatrix} x \\ y \end{pmatrix} = (x, y)^\top$, $\mathbf{B} = (x, y, z)^\top$
Matrizen	Schreibmaschine	$\mathtt{A}, \mathtt{B} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
Mengen	kalligrafisch	$\mathcal{A}, B = \{a, b\}, b \in \mathcal{B}$
Zahlenbereiche, Koordinatenräume	doppelt gestrichen	$\mathbb{N}, \mathbb{Z}, \mathbb{R}^2, \mathbb{R}^3$

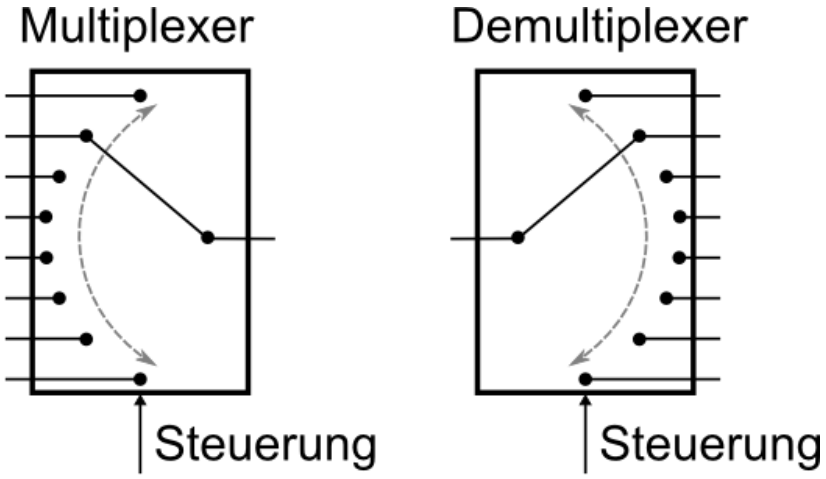
Inhalt

- Multiplexer / Demultiplexer
- Programmierbare logische Schaltungen
  - PLAs
  - PALs
  - PROMs

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

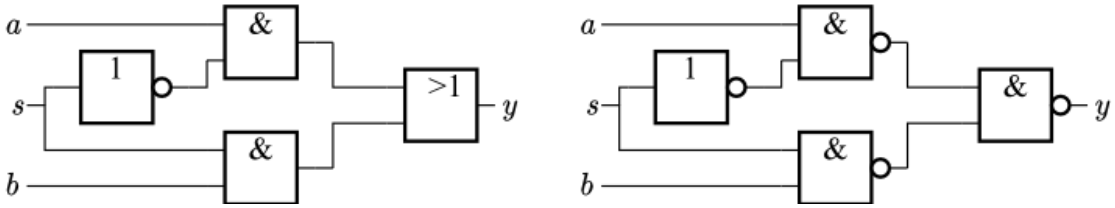
Für einen 2-zu-1 Multiplexer (auch "2:1 Mux") gilt:

Wenn das Steuersignal $s = 1$, dann ist $y = b$, ansonsten $y = a$

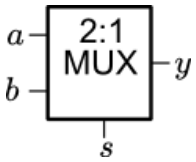
Diese entspricht einem if-then-else:

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

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



Da er häufig Bestandteil von digitalen Schaltungen ist, bekommt der Multiplexer ein eigenes Symbol



Multiplexer

2-zu-1 Multiplexer (2:1 Mux):

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

4-zu-1 Multiplexer (4:1 Mux):

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

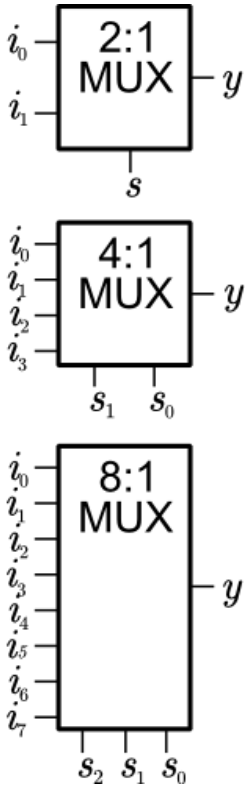
8-zu-1 Multiplexer (8:1 Mux):

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

n-zu-1 Multiplexer (n:1 Mux):

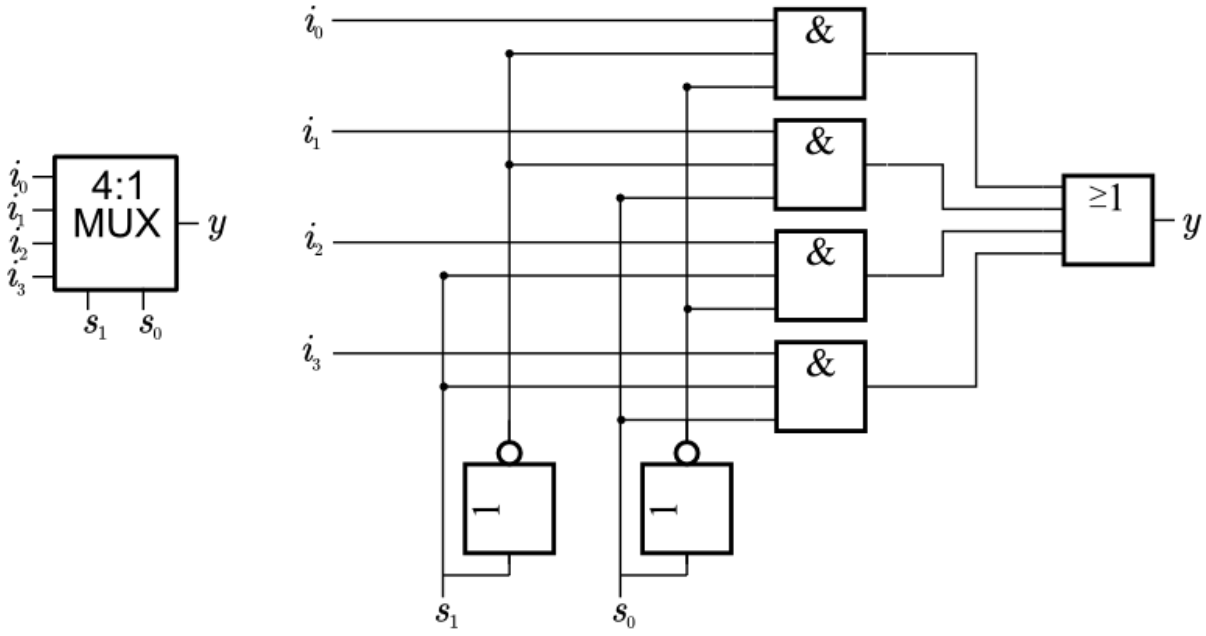
$$y = \bigvee_{j=0}^{n-1} m_j i_j$$

wobei m_j den Minterm der k Steuersignale $s_{k-1}, \ldots, s_1, s_0$ bezeichnet und gilt $n = 2^k$



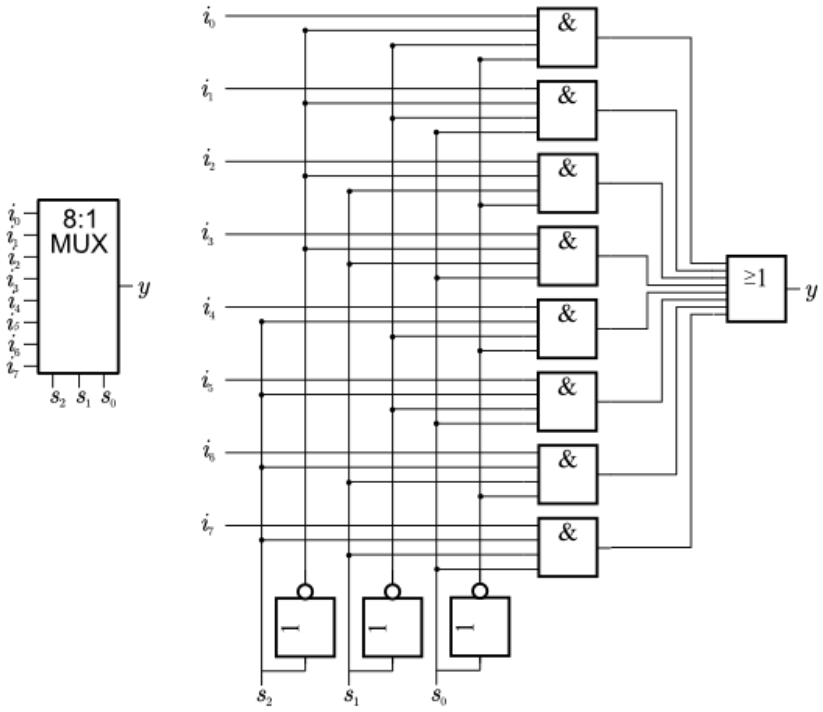
Realisierung von Multiplexern

Ein 4:1 Mux als zweistufige Logik:



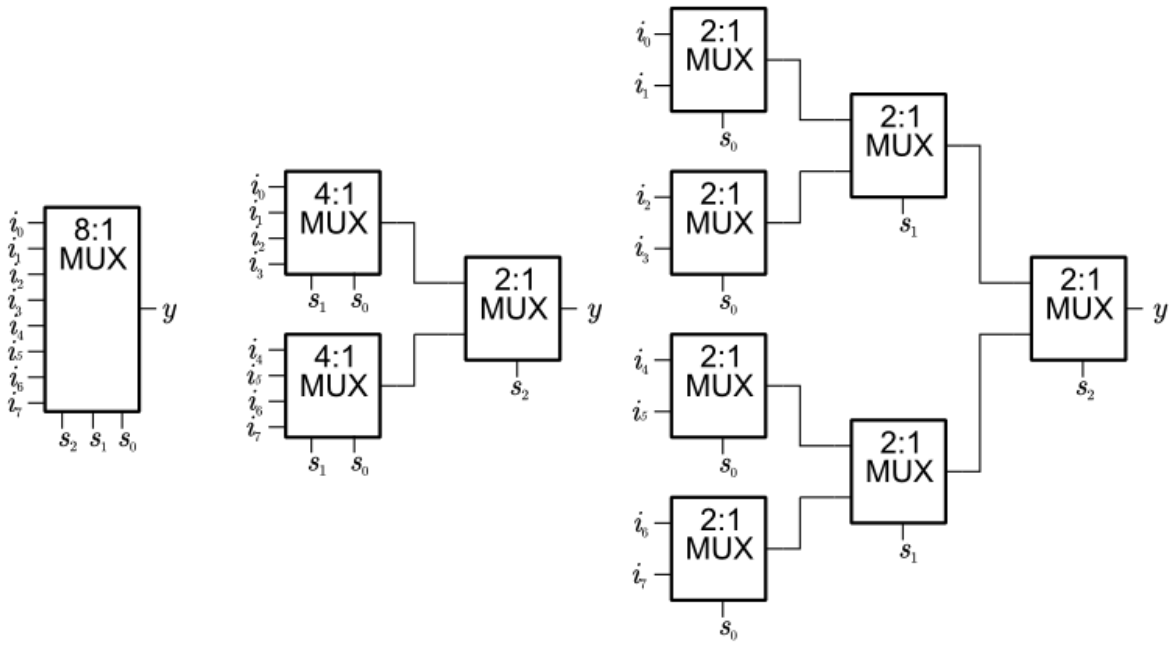
Realisierung von Multiplexern

Ein 8:1 Mux als zweistufige Logik:



Kaskadierung von Multiplexern

Große Multiplexer können durch Kaskadierung von kleinen Multiplexern aufgebaut werden



Was macht mehr Sinn? Kaskadierung oder zweistufige Logik?

Kaskadierung von Multiplexern

Was macht mehr Sinn? Kaskadierung oder zweistufige Logik?

Antwort: Es kommt darauf an, was optimiert werden soll.

- Gatterverzögerung: Bei zweistufige Logik entstehen kleinere Gatterlaufzeiten
- Chipfläche: Eine Kaskadierung benötigt insgesamt weniger Eingänge und kann daher auf kleinerer Chipfläche realisiert werden.

Beispielrechnung:

- Eine zweistufige Logik benötigt UND-Gatter mit $(log_2 n) + 1$ Eingängen.
- Eine kaskadierte Lösung hat mehr UND-Gatter, diese haben jedoch weniger Eingänge.
- Beispiel 8:1 Mux: $8 \cdot 4 = 32$ Eingänge im Vergleich zu $2 \cdot 2 \cdot 7 = 28$ Eingänge
- Wenn der Flächenbedarf proportional zu den Eingängen angenommen wird, benötigt die kaskadierte Realisierung insgesamt weniger Fläche

Multiplexer als universelle Logik

n-zu-1 Multiplexer können jede Funktion von $k = log_2 n$ Variablen implementieren

Die Variablen $x_{k-1}, \ldots, x_0$ werden als Steuervariablen $s_{k-1}, \ldots, s_0$ verwendet

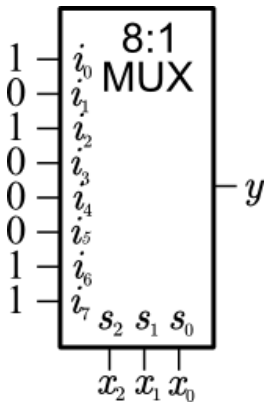
Dateneingänge $i_{n-1}, \ldots, i_0$ werden auf 0 bzw. 1 gelegt

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



Multiplexer als universelle Logik

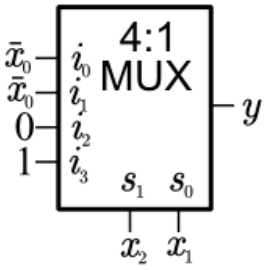
Alternative: n-zu-1 Multiplexer können jede Funktion von $k = (log_2 n) + 1$ Variablen implementieren

- Die Variablen $x_{k-1}, \ldots, x_1$ werden als Steuervariablen $s_{k-2}, \ldots, s_0$ verwendet
- Dateneingänge werden auf x_0, $\bar{x}_0$, 0 oder 1 gelegt

Beispiel:

Wahrheitstafel:

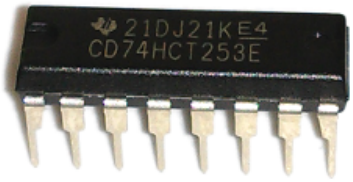
	x_2	x_1	x_0	y	
0:	0	0	0	1	$\bar{x}_0$
1:	0	0	1	0	
2:	0	1	0	1	$\bar{x}_0$
3:	0	1	1	0	
4:	1	0	0	0	0
5:	1	0	1	0	
6:	1	1	0	1	1
7:	1	1	1	1	

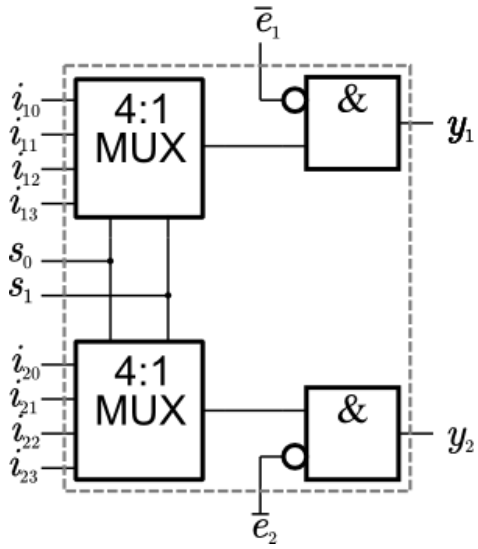


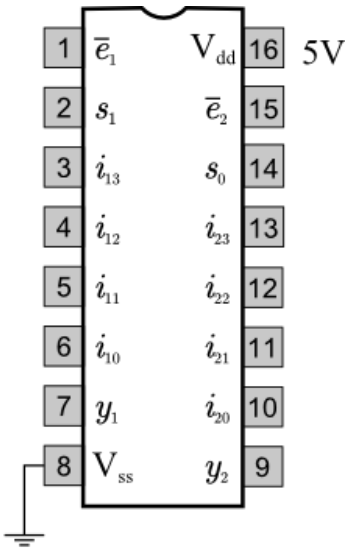
Praxisbeispiel: Multiplexer als universelle Logik

Im Folgenden soll mit Hilfe des CMOS IC CD74HCT253 eine reale Multiplexer-Schaltung aufbaut werden

Der 74HCT253 enthält zwei 4-zu-1 Multiplexer



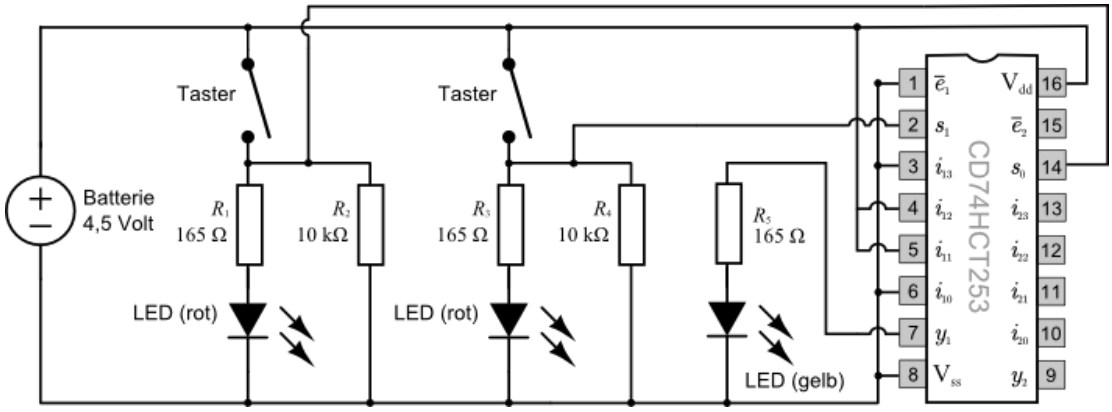




Praxisbeispiel: Multiplexer als universelle Logik

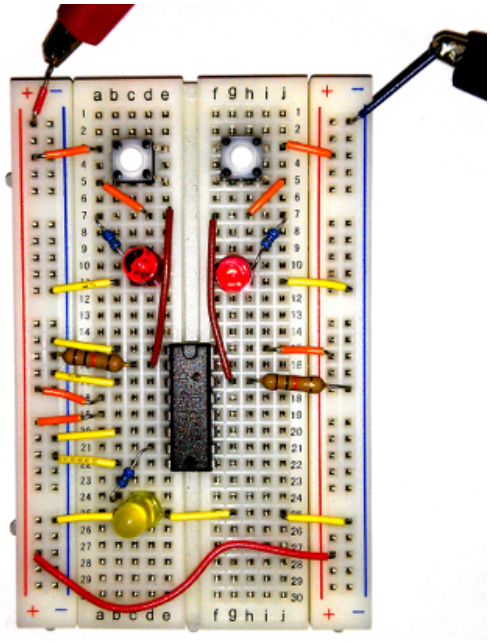
Einer der zwei 4-zu-1 Multiplexer wird derart beschaltet, dass ein XOR realisiert wird

s_1	s_0	y_1
0	0	0
0	1	1
1	0	1
1	1	0



Praxisbeispiel: Multiplexer als universelle Logik

Dieses Photo zeigt den Aufbau der Schaltung aus der vorangegangenen Folie auf eine Steckplatine



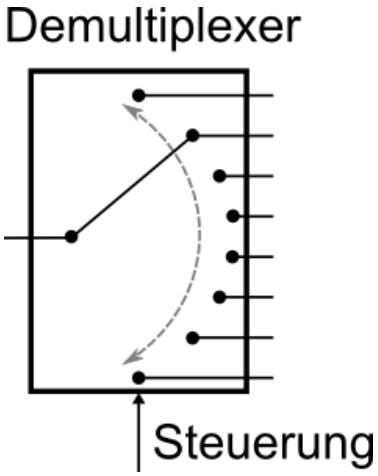
Demultiplexer

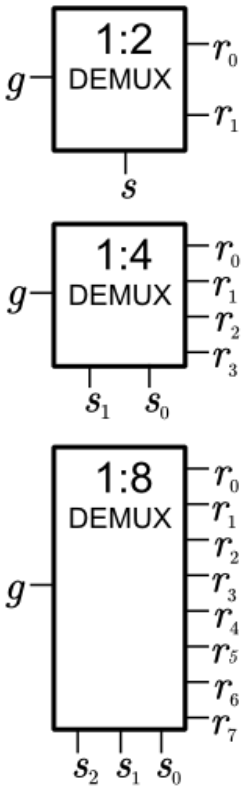
Das inverse Element zum Multiplexer ist der Demultiplexer

1 Eingang, k Steuereingänge, $n = 2^k$ Ausgänge $r_{n-1}, \ldots, r_0$

Der Eingang g wird oft "Enable" genannt

Abhängig von den Steuereingängen wird der Eingang auf einen der Ausgänge geschaltet, die anderen Ausgänge sind 0.





Demultiplexer

1-zu-2 Demultiplexer (1:2 Demux):

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

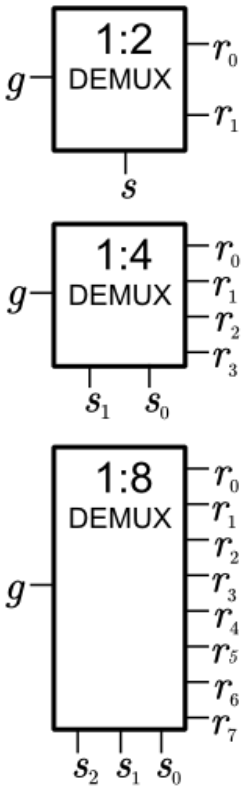
1-zu-4 Demultiplexer (1:4 Demux):

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

1-zu-n Demultiplexer (1:n Demux):

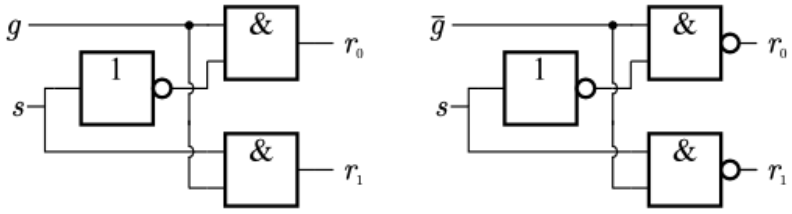
$r_j = g\,m_j \quad \forall\, r_{n-1}, \ldots, r_1, r_0$

wobei m_j den Minterm der k Steuersignale $s_{k-1}, \ldots, s_1, s_0$ bezeichnet und gilt $n = 2^k$

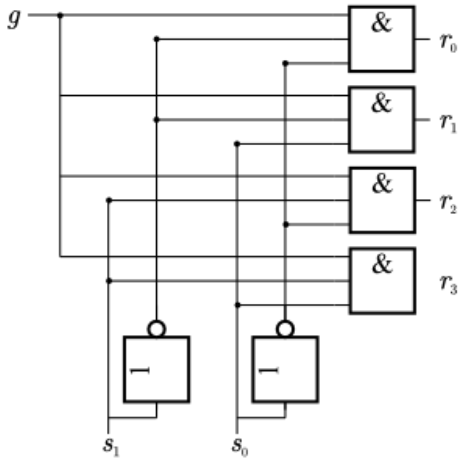


Realiserung von Demultiplexern

Ein 1:2 Demux mit Gattern:

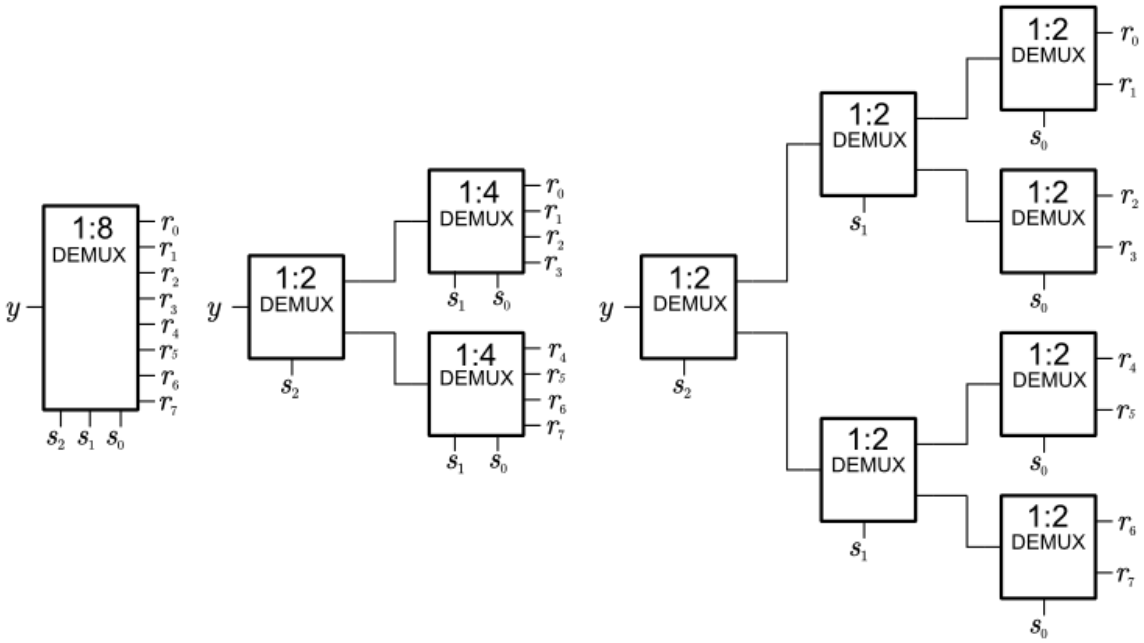


Ein 1:4 Demux mit Gattern:



Kaskadierung von Demultiplexern

Große Demultiplexer können durch Kaskadierung von kleinen Demultiplexern aufgebaut werden



Demultiplexer als universelle Logik

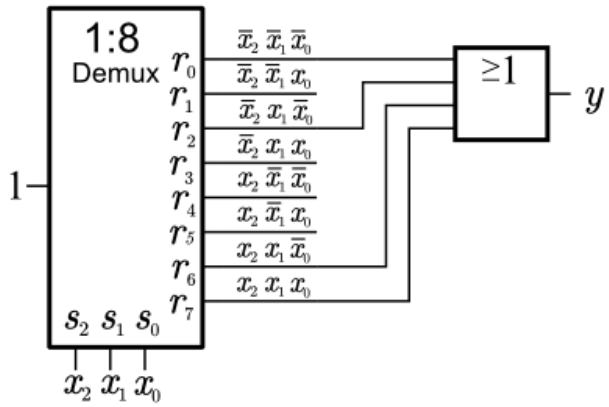
1-zu-n Demultiplexer können jede Funktion von $k = log_2 n$ Variablen implementieren

- Die Variablen $x_{k-1}, \ldots, x_0$ werden als Steuervariablen $s_{k-1}, \ldots, s_0$ verwendet
- 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

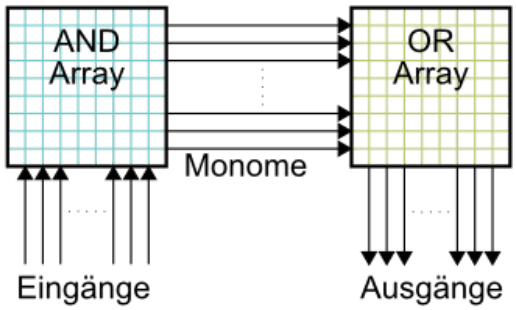


Programmierbare logische Schaltungen

Eine programmierbare logische Schaltung (engl.: Programmable Logic Device, PLD) ist ein vorgefertigter IC mit vielen AND- und OR-Gattern

Die Eingangsvariablen werden auf ein AND-Array geführt das Monome erzeugt

Die Monome werden durch ein OR-Array zu den Ausgangsfunktionen verknüpft



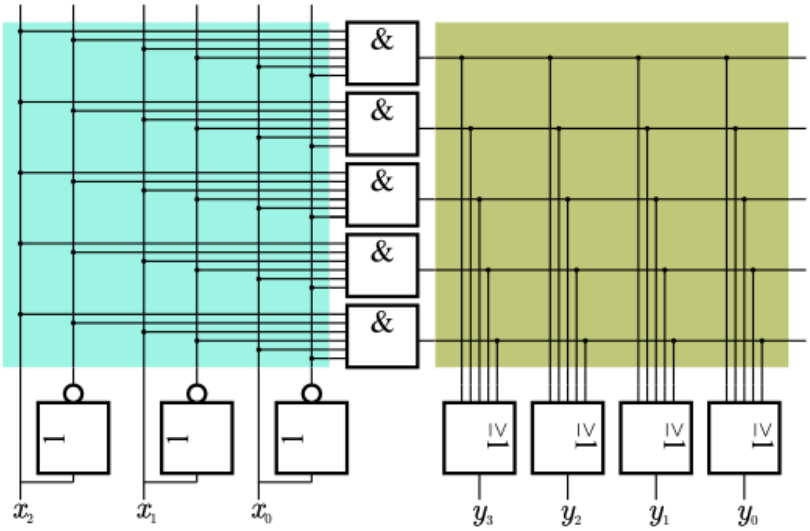
Vorteil: PLD können in großer Stückzahl gefertigt werden und erst später durch das Auftrennen von Verbindungen ihre Funktionalität bekommen ("programmiert werden")

PLA (Programmable Logic Array): AND-Array veränderbar, OR-Array veränderbar

PAL (Programmable Array Logic): AND-Array veränderbar, OR-Array fest

PROM (Programmable Read-Only Memory): AND-Array fest, OR-Array veränderbar

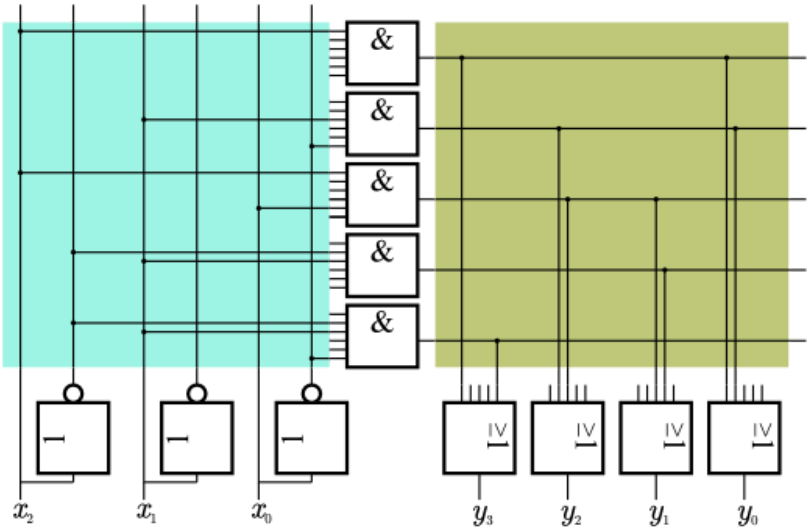
PLA (Programmable Logic Array)



Nicht benötigte Verbindungen werden aufgetrennt.

Realisierte Monome können für verschiedene Ausgänge y_i wieder verwendet werden

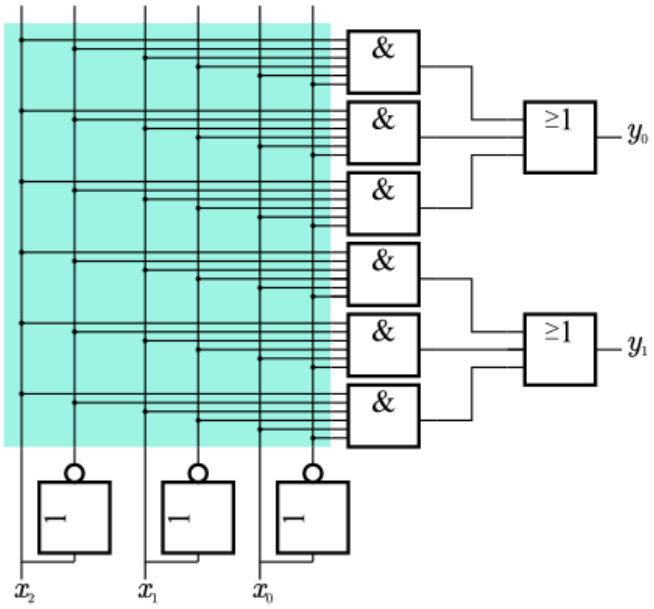
PLA (Programmable Logic Array)



Beispiel:

$$y_0 = x_1\bar{x}_0 \vee x_2$$
$$y_1 = \bar{x}_2x_1 \vee x_2x_0$$
$$y_2 = x_1\bar{x}_0 \vee x_2x_0$$
$$y_3 = x_2 \vee \bar{x}_2x_1\bar{x}_0$$

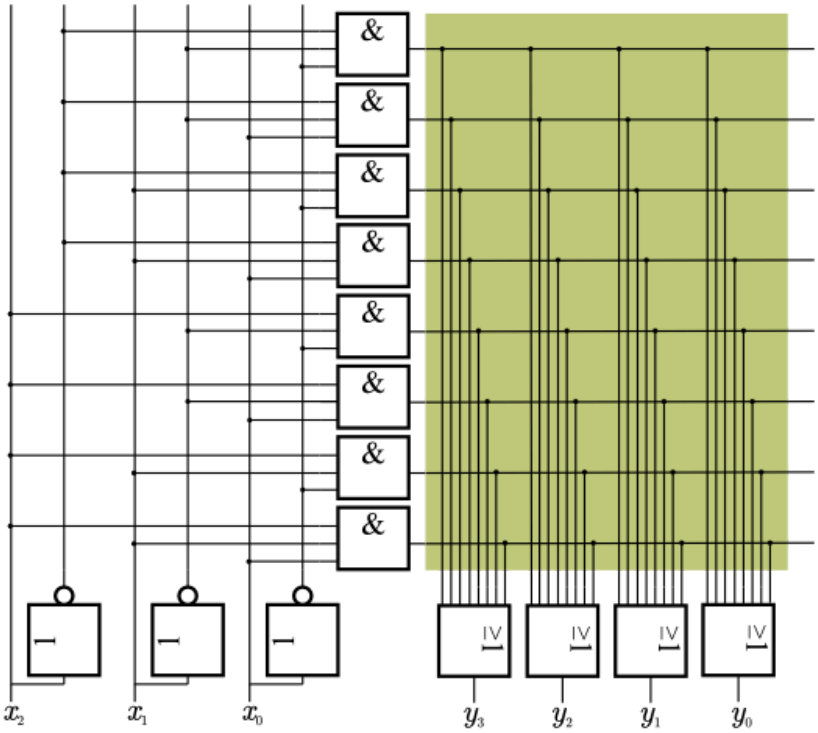
PAL (Programmable Array Logic)



Beim PAL ist nur das AND-Array veränderbar

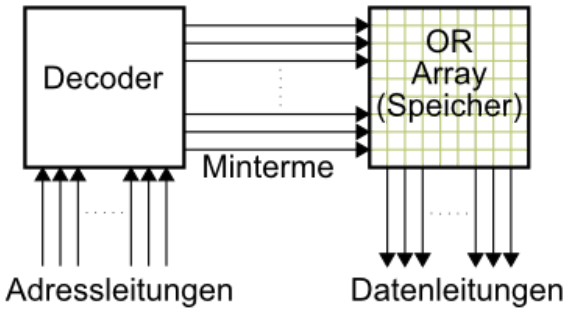
Die ODER-Verknüpfungen sind fest verdrahtet

PROM (Programmable Read-Only Memory)



Die UND-Verknüpfungen sind fest verdrahtet, alle Minterme realisiert

PROM (Programmable Read-Only Memory)



Beim Anlegen einer Adresse an die Eingänge liefert das PROM den Speicherinhalt für diese Adresse an den Ausgängen

Im Unterschied zum PLA hat das PROM ein fest verdrahtetes AND-Array, das alle möglichen Kombinationen der Eingänge (alle Minterme) realisiert

Bei k Adressleitungen entstehen somit 2^k Minterme (auch "Wortleitungen" genannt)

Bei m Ausgängen liegen im ROM-Speicher also 2^k Worte zu m Bits

Gibt es Fragen?

Anregungen oder Verbesserungsvorschläge können auch gerne per E-mail an mich gesendet werden: Kontakt

Weitere Vorlesungsfolien