

GTI - Grundlagen der Technischen Informatik


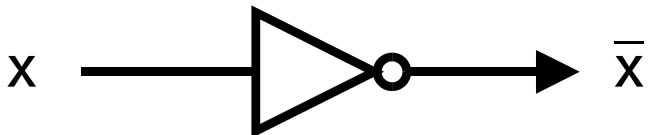
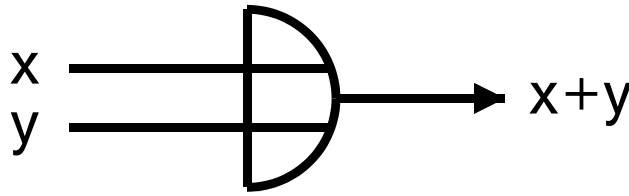
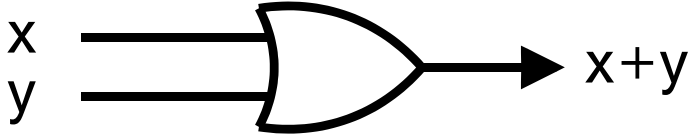
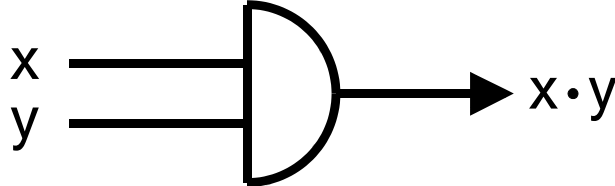
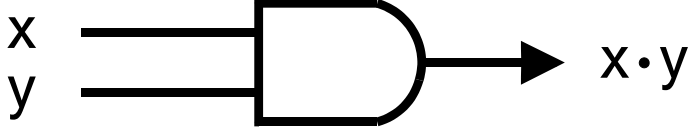
4. Schaltnetze

Thomas Studer
Institut für Informatik
Universität Bern

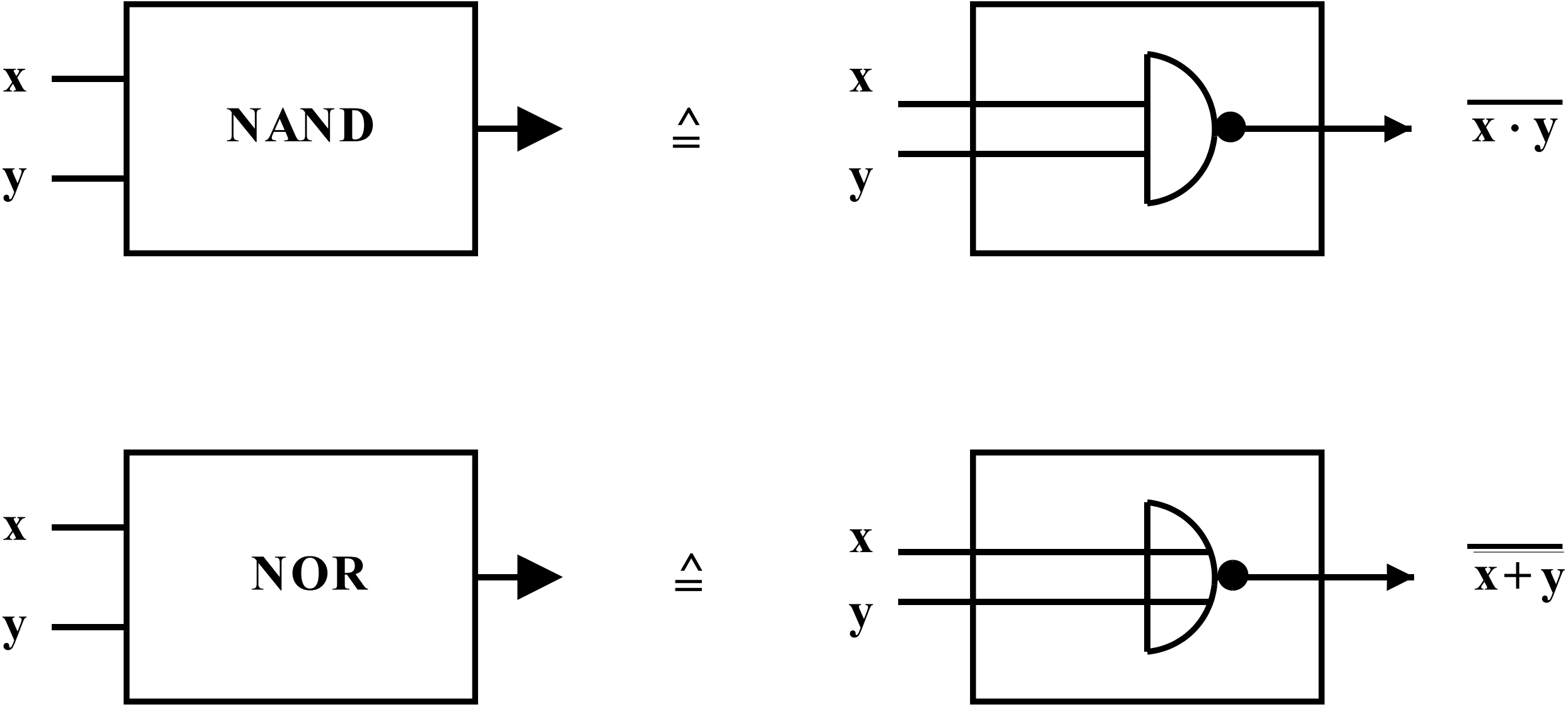
Inhalt

- > Entwurf von Schaltnetzen
- > Multiplexer und verwandte Bausteine
- > Addiernetze

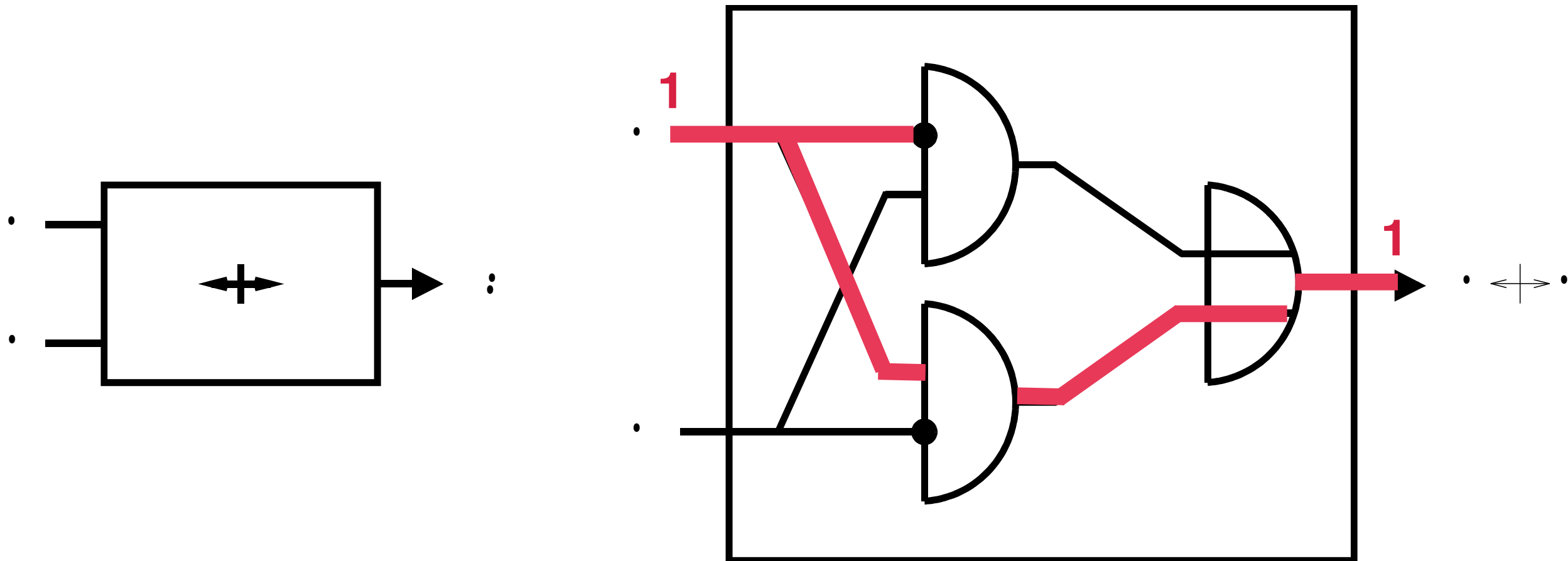
Grundbausteine zur Realisierung von Booleschen Funktionen

Funktion	Unser Symbol	IEEE-Symbol
Negation (Komplement-Gatter)		
Addition (Oder-Gatter)		
Multiplikation (Und-Gatter)		

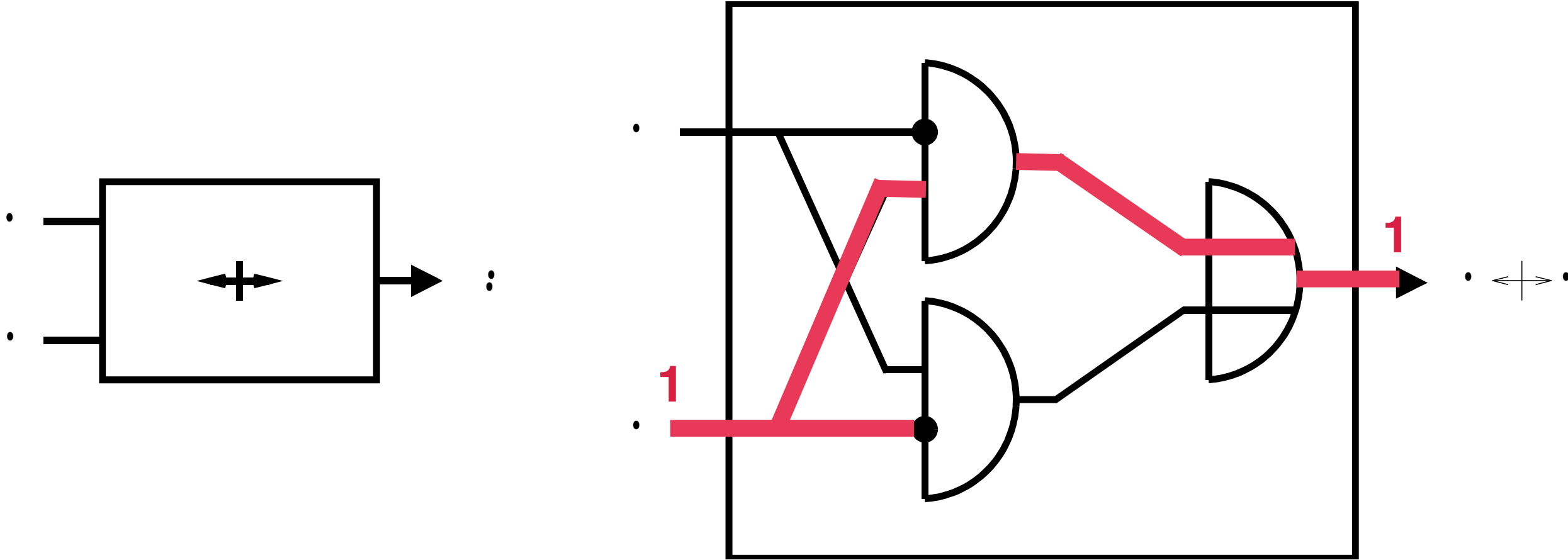
NAND und NOR



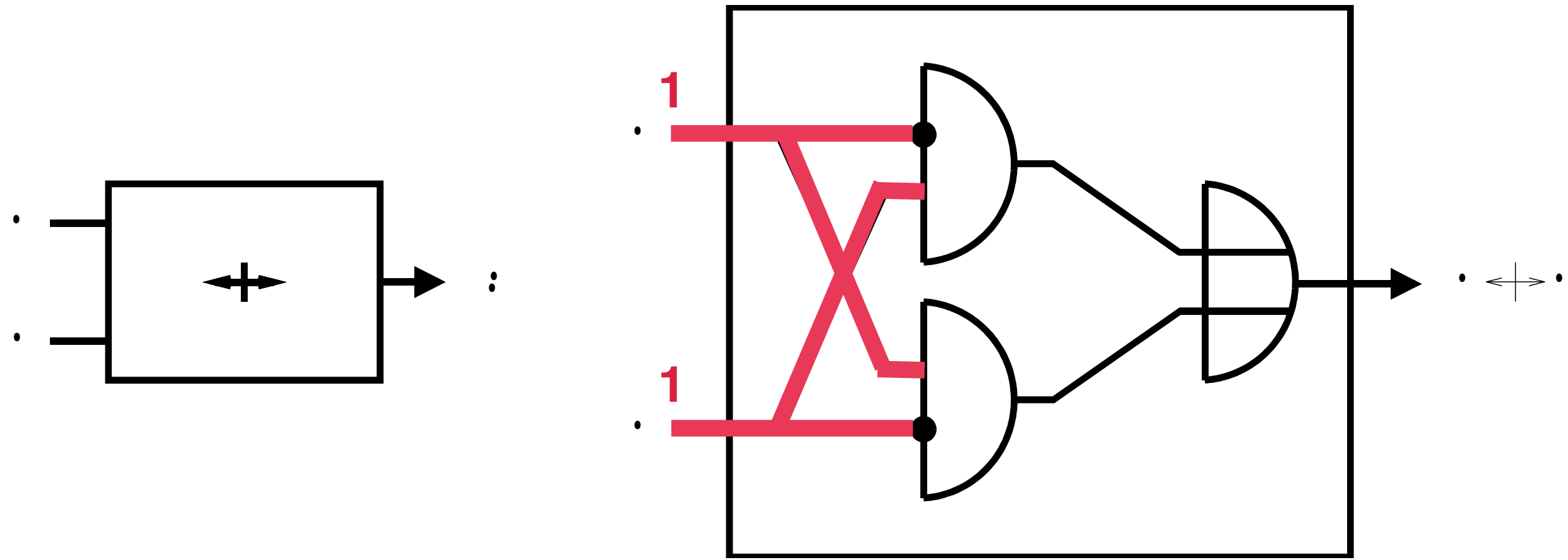
XOR



XOR



XOR



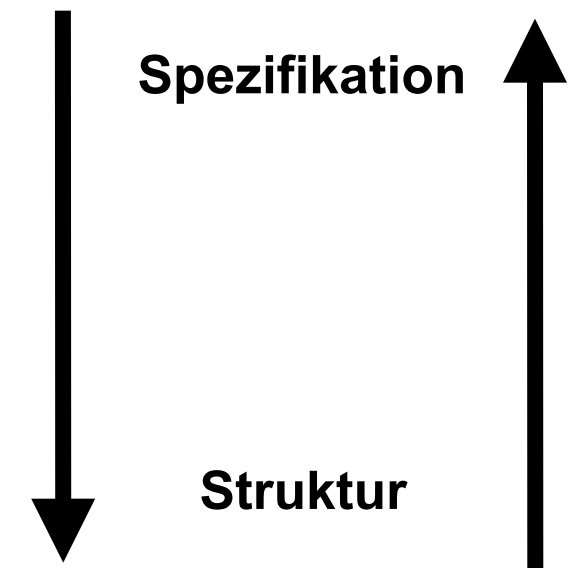
Entwurf von Schaltnetzen

> Bottom-Up

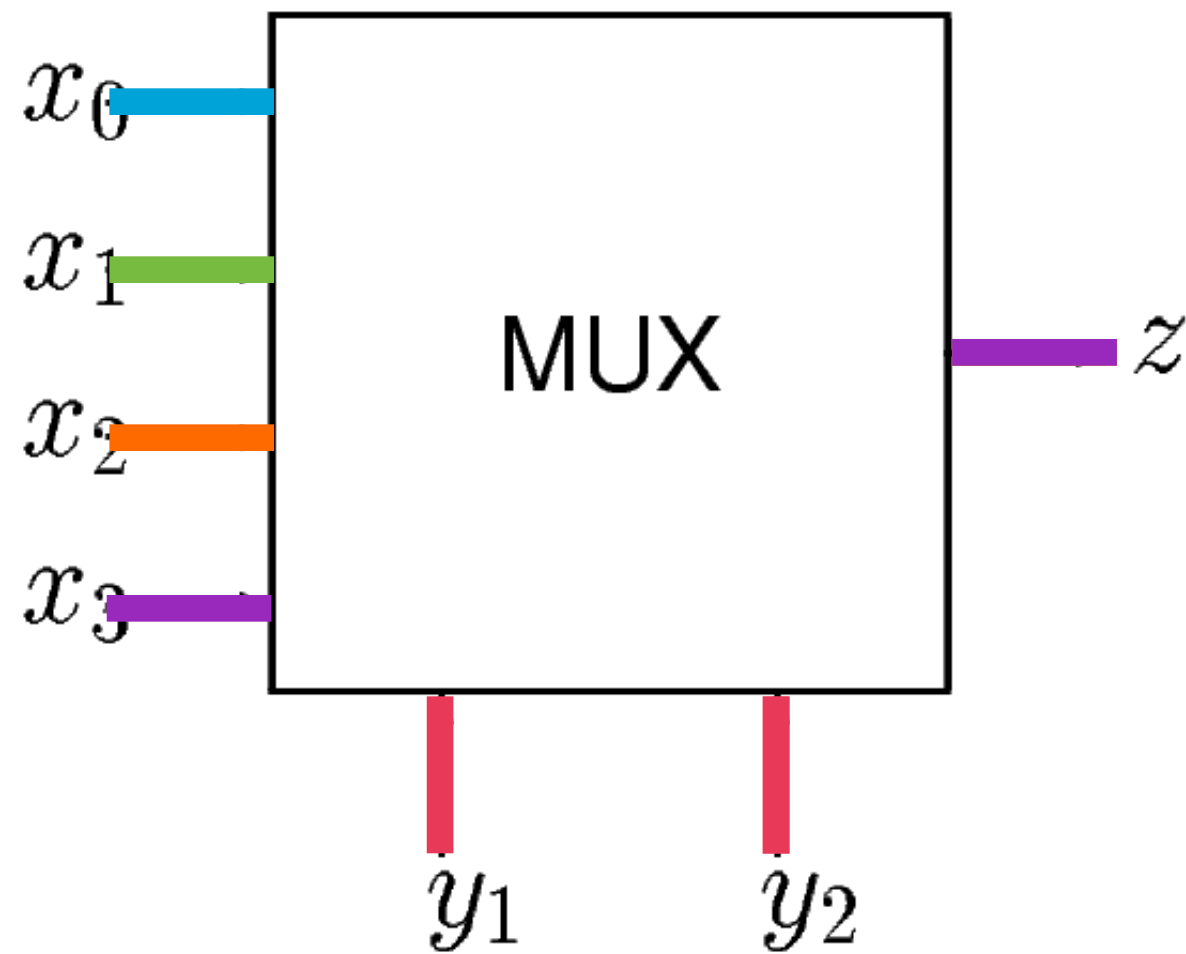
- Sukzessives Zusammensetzen komplexer Schaltungen aus elementaren Bausteinen (z.B. Grundbausteine wie NAND- oder NOR-Gatter)
- Integration:
Zusammenfassen von Grundbausteinen

> Top-Down

- Zerlegen der Gesamtaufgabe in wohl definierte Teilaufgaben mit mehrfacher Verfeinerung



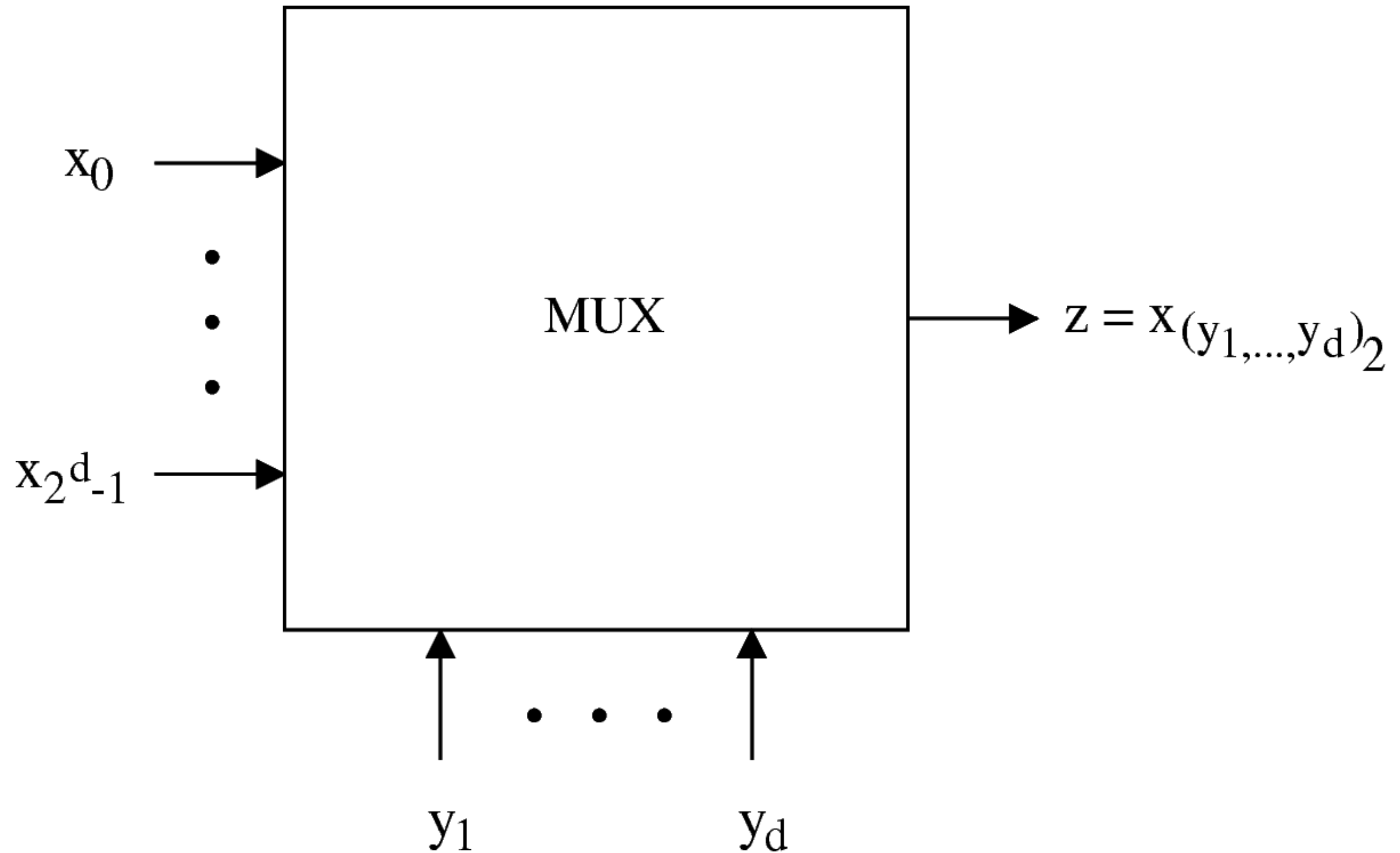
2-MUX (Prinzip)



→

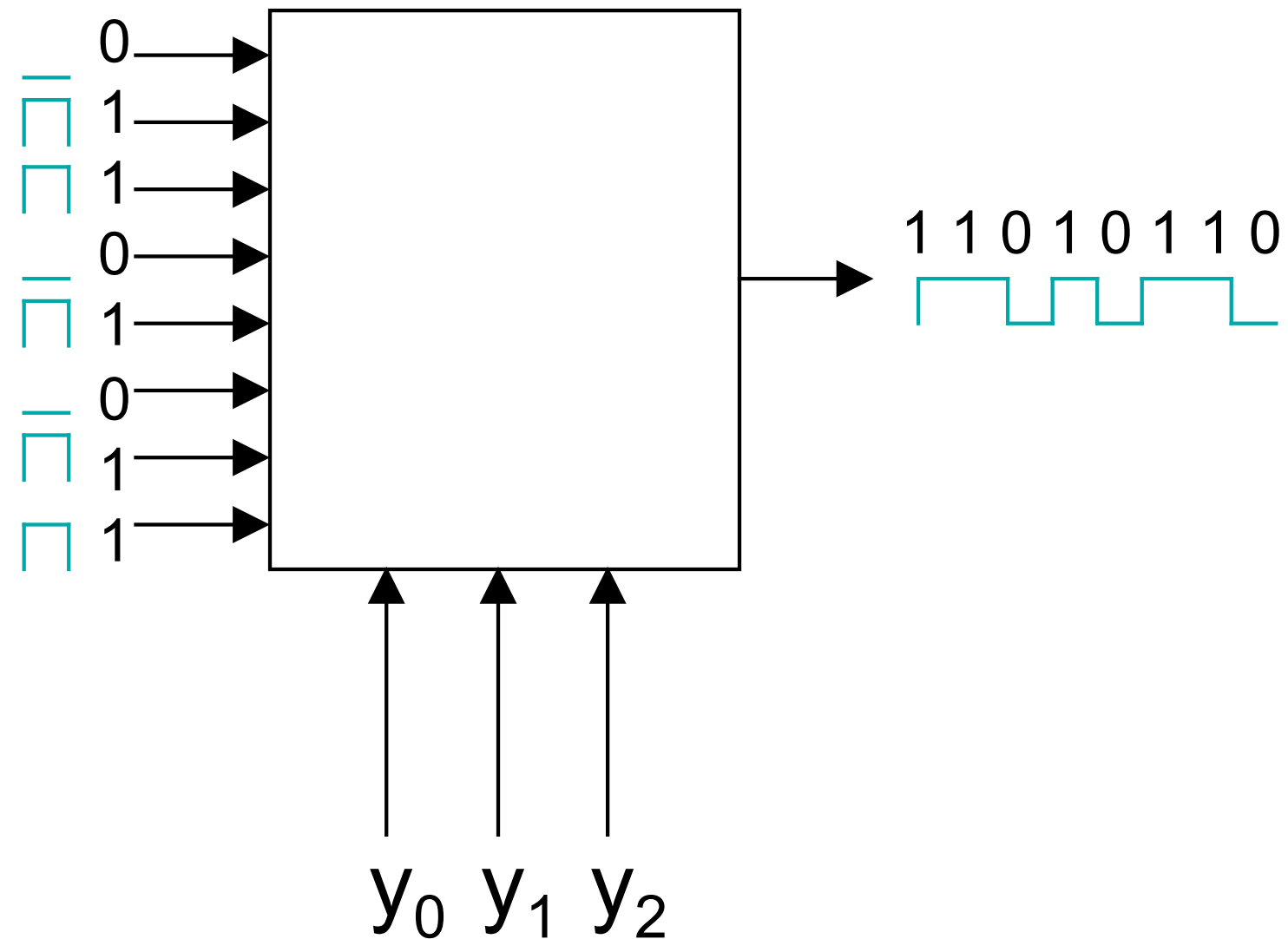
y_1	y_2	z
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

Allgemeiner MUX-Aufbau

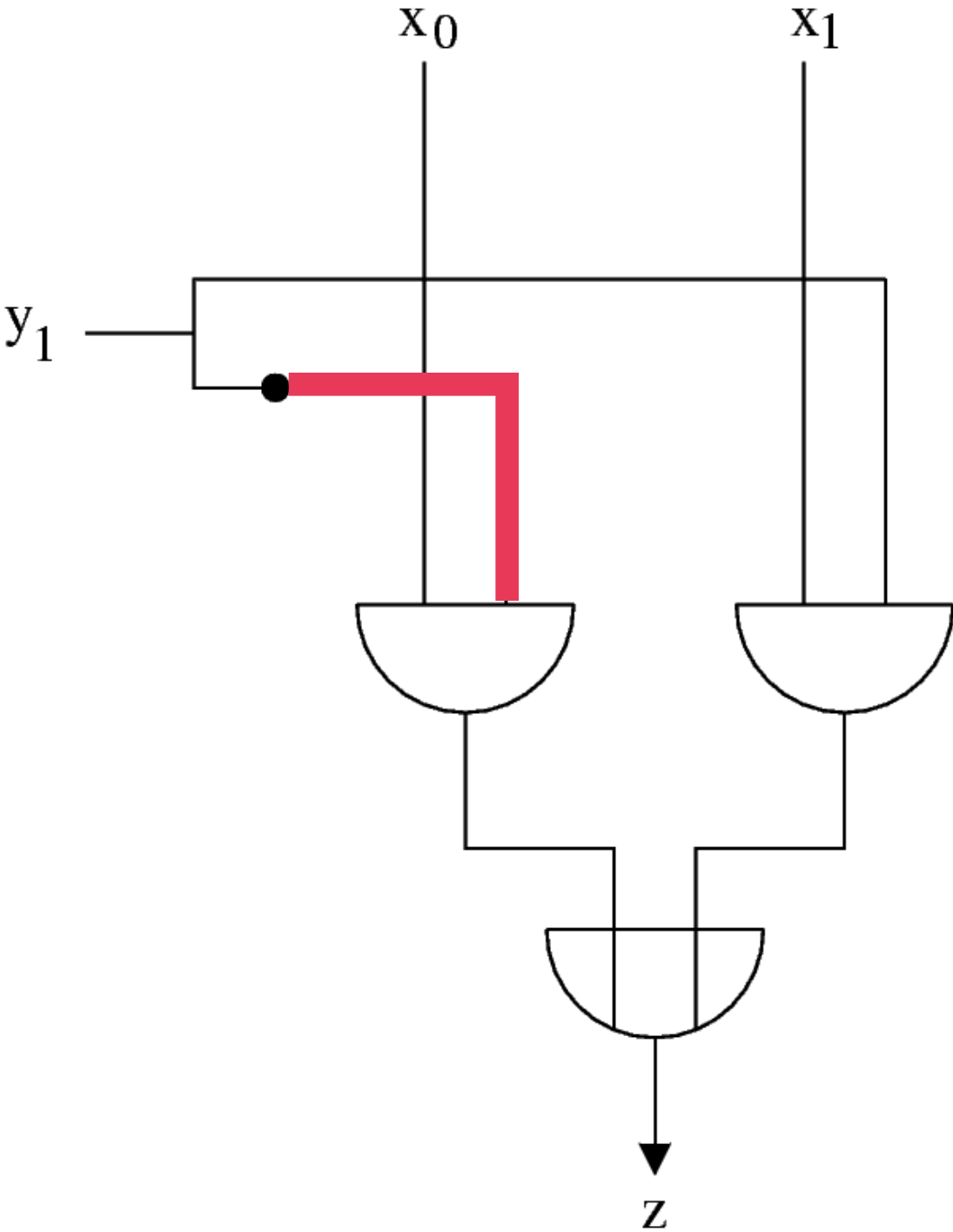


Anwendungsbeispiele

- > Serialisierung
- > Zeitmultiplexen

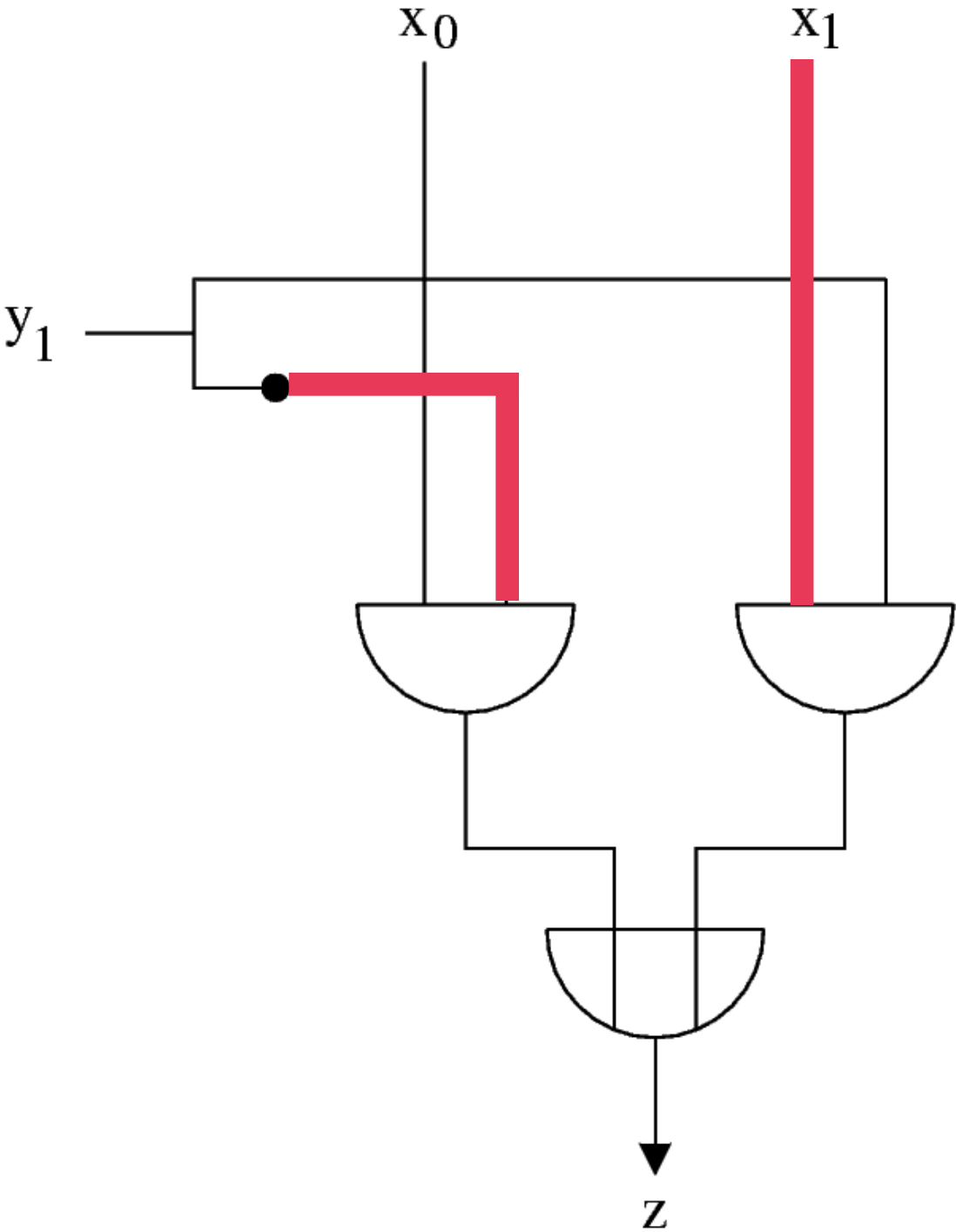


1-MUX



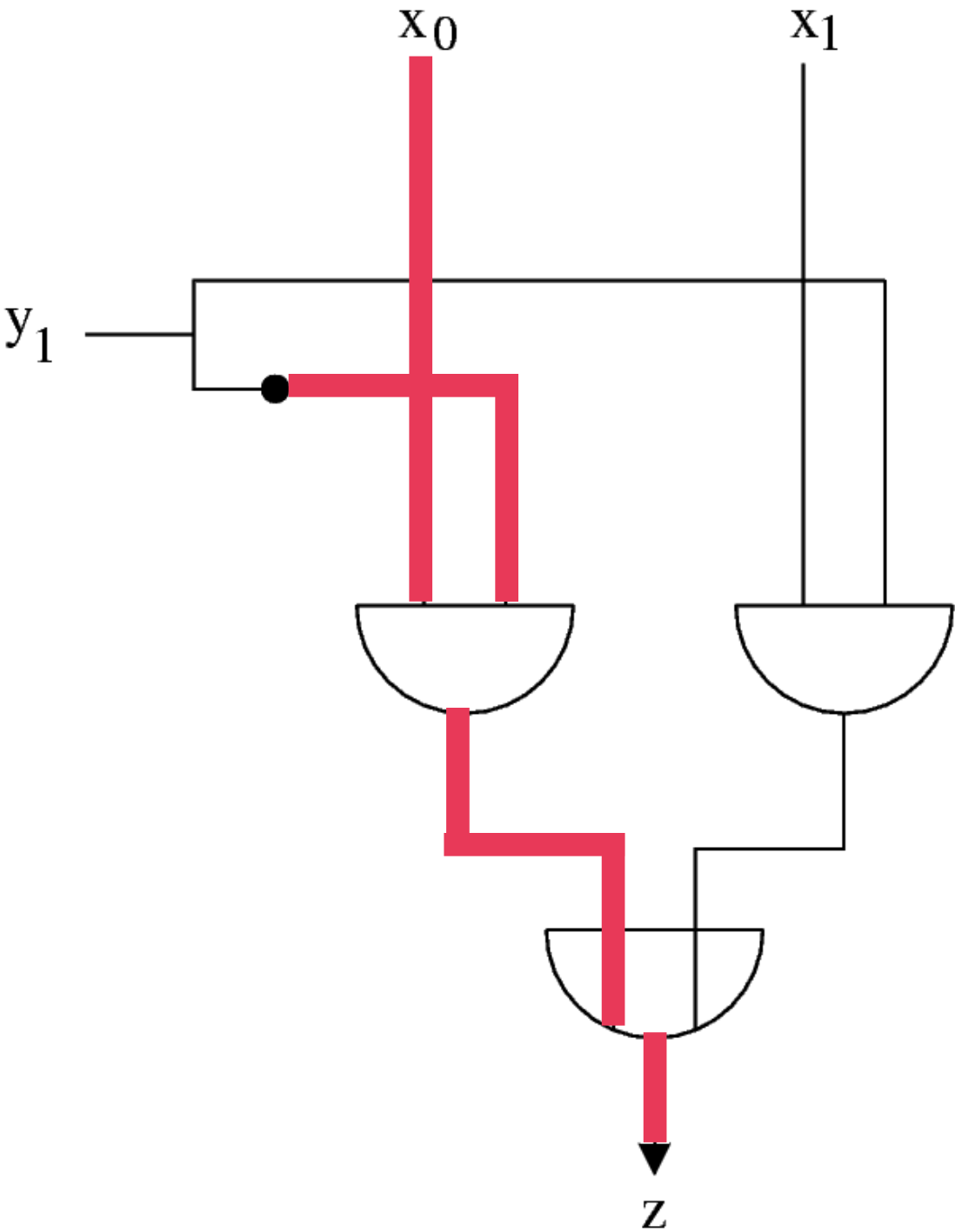
x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

1-MUX



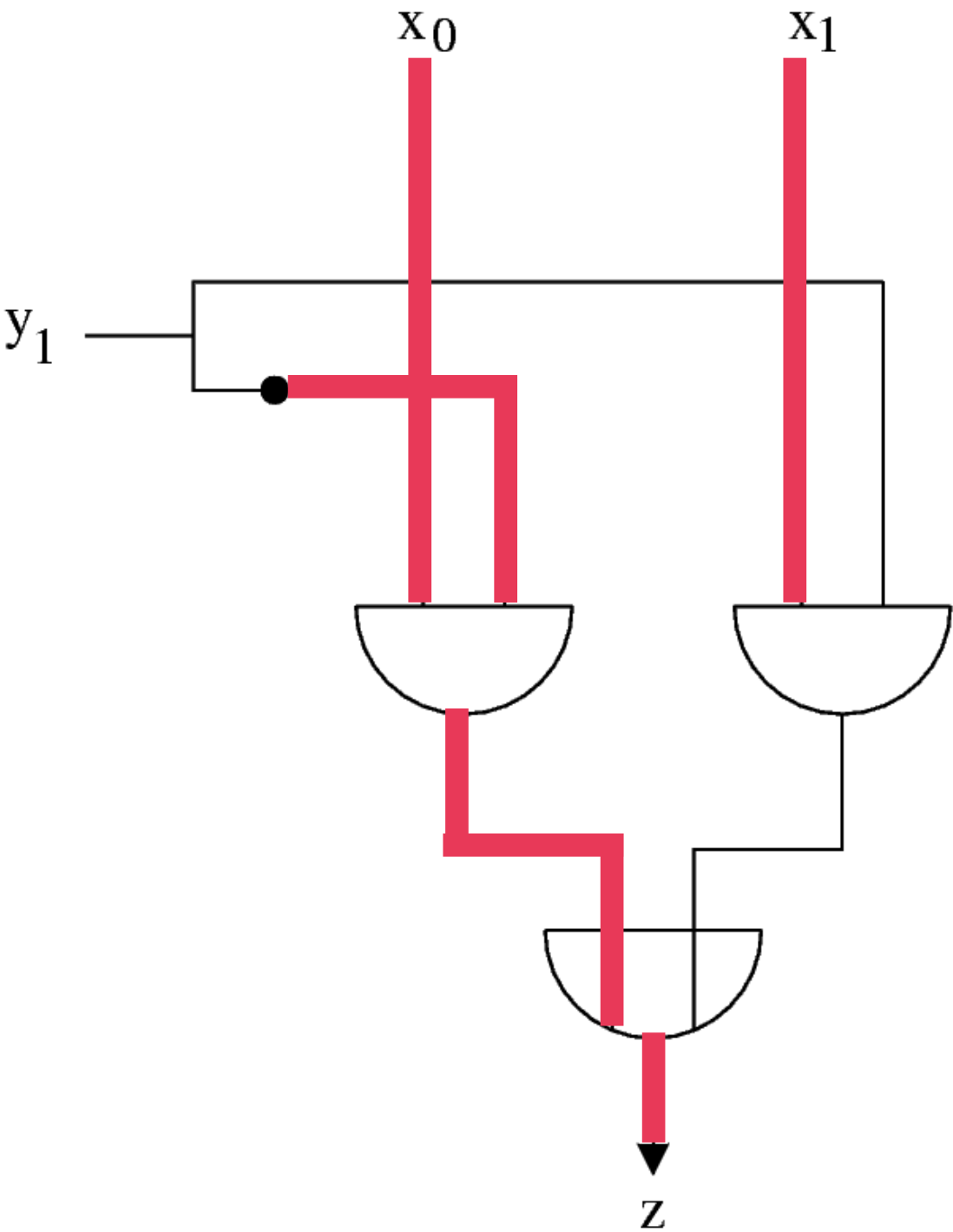
x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

1-MUX



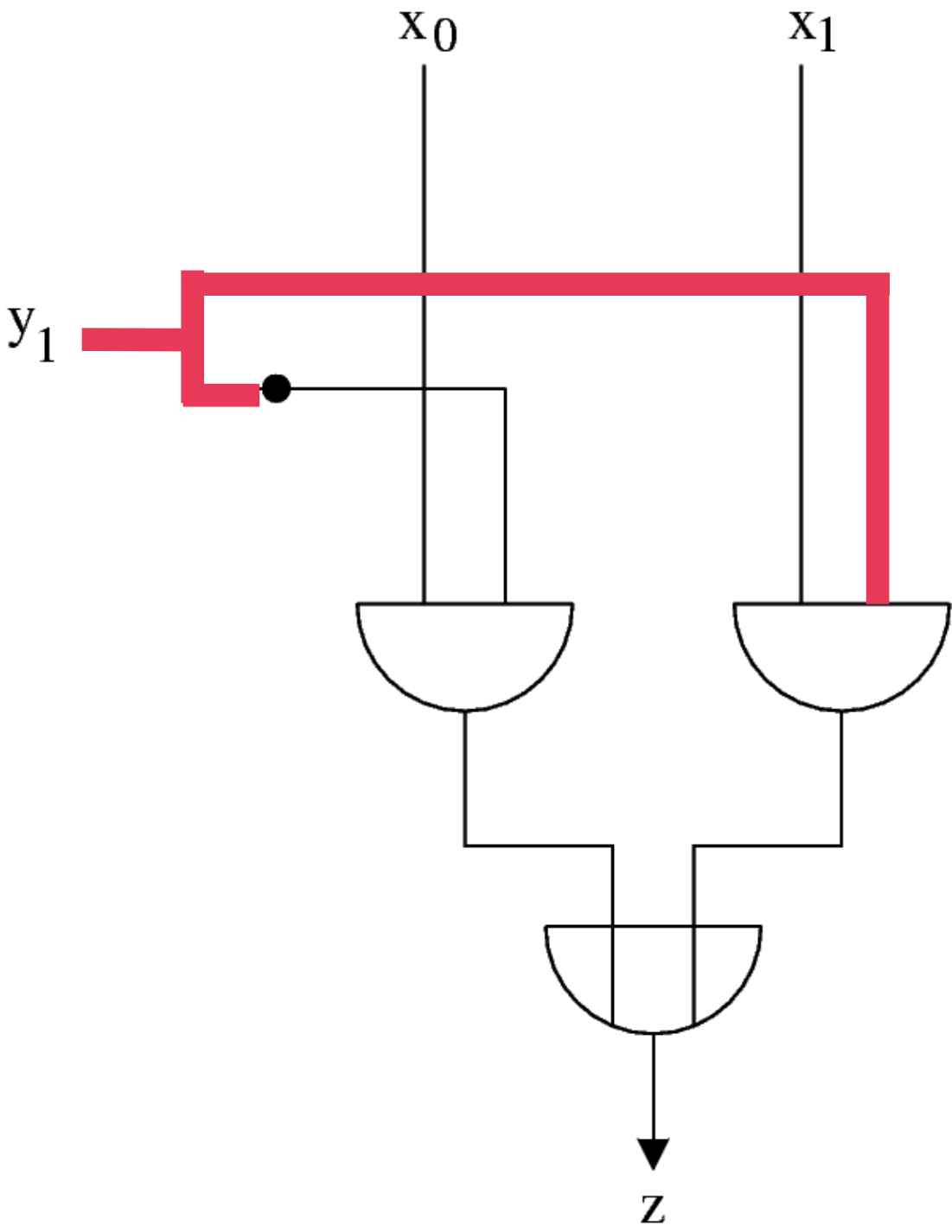
x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

1-MUX



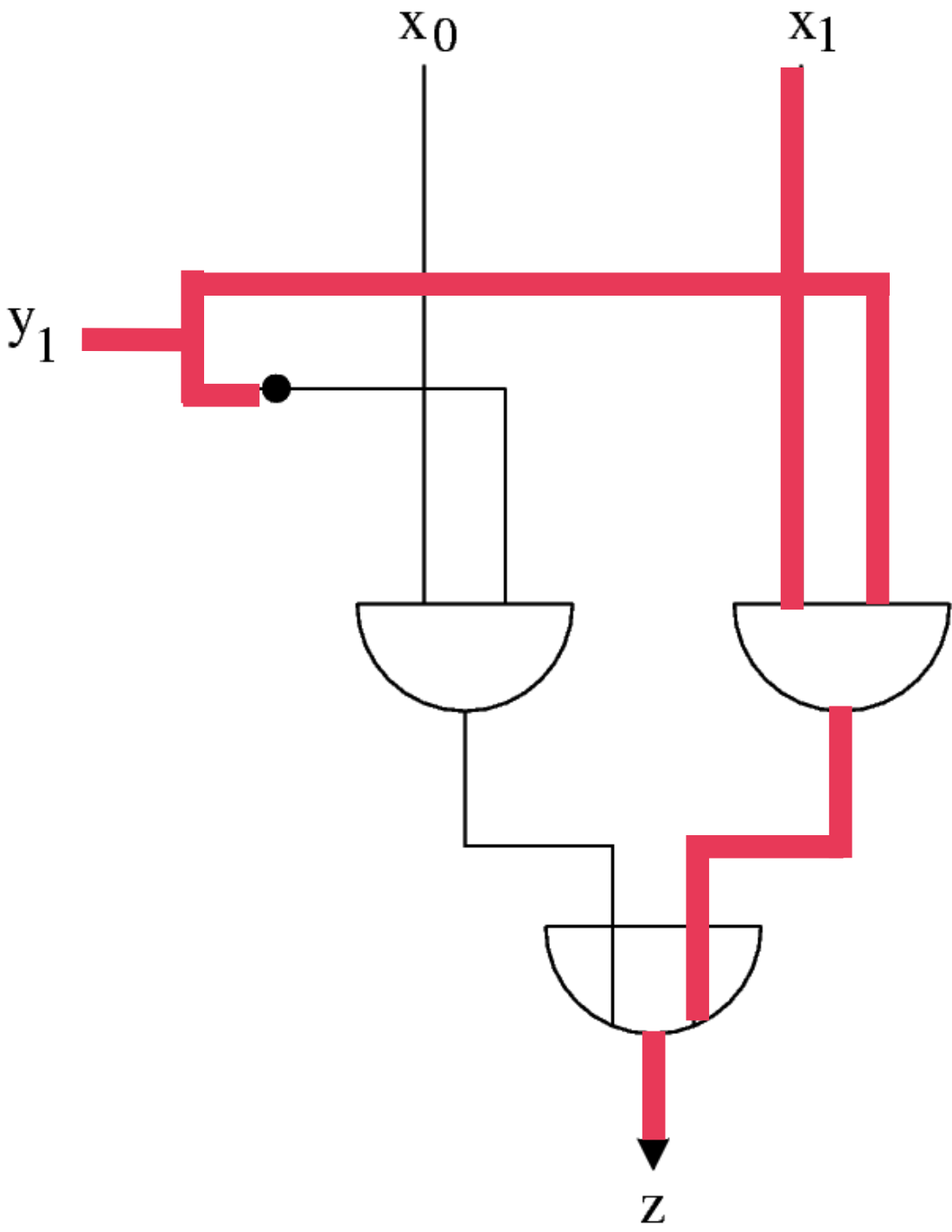
x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

1-MUX



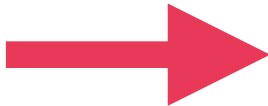
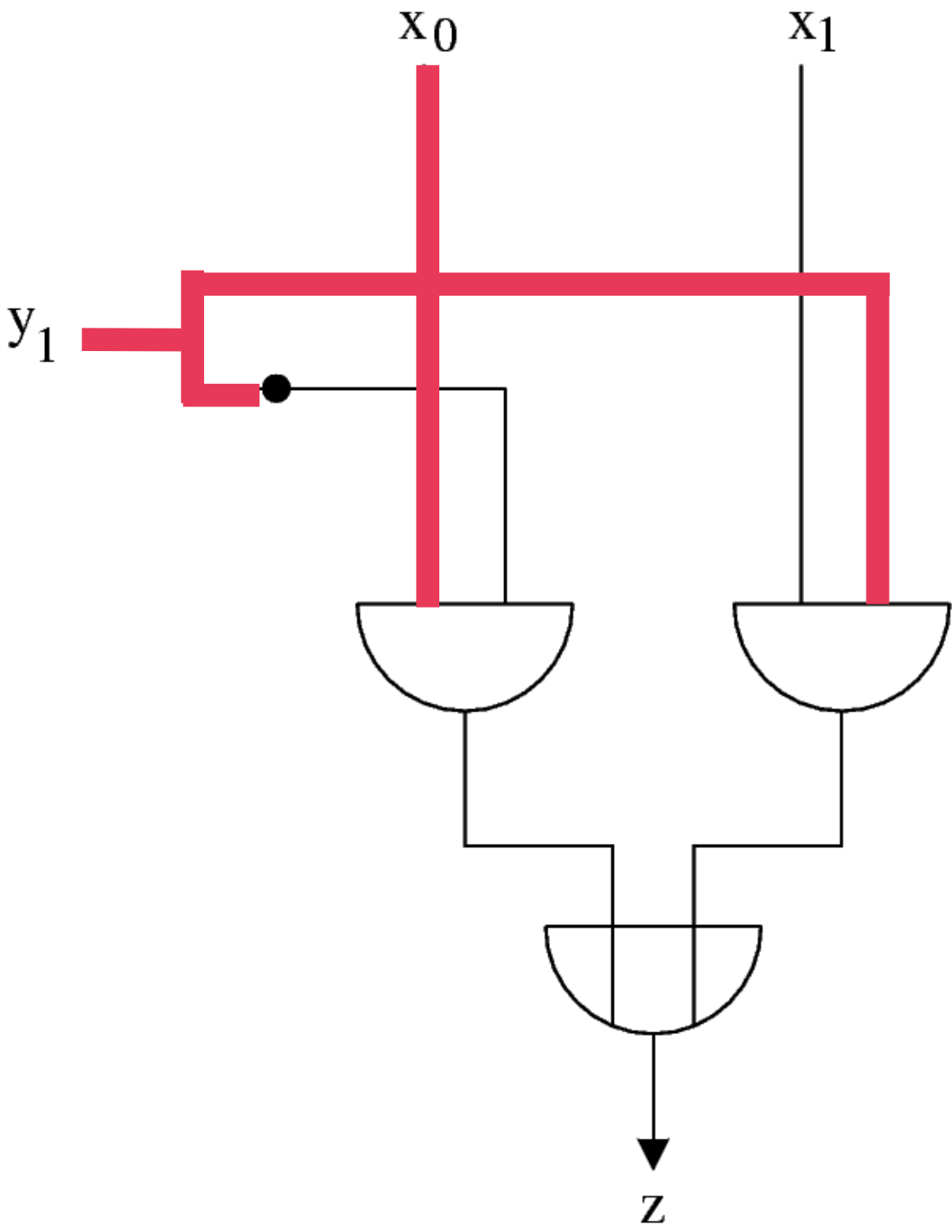
x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

1-MUX



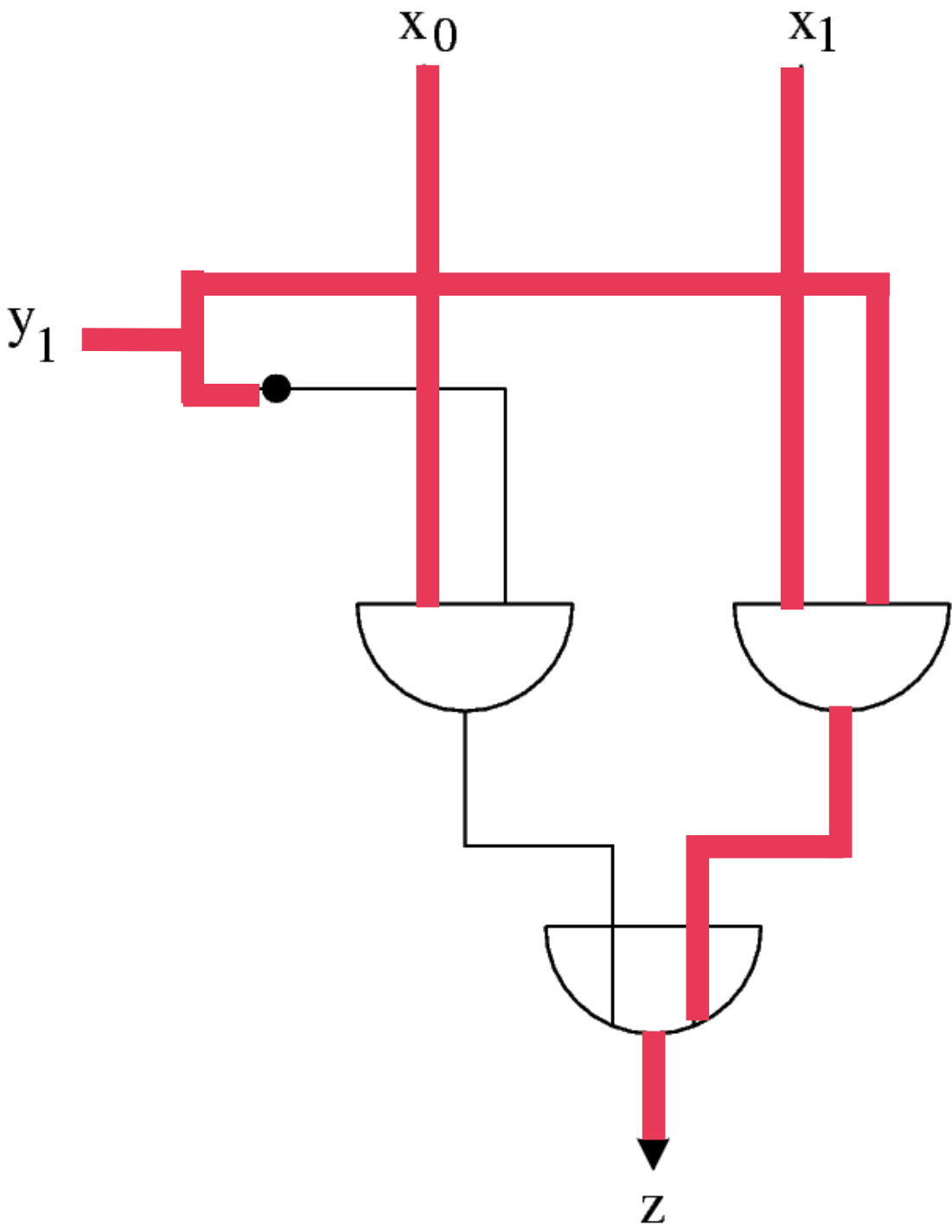
x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

1-MUX



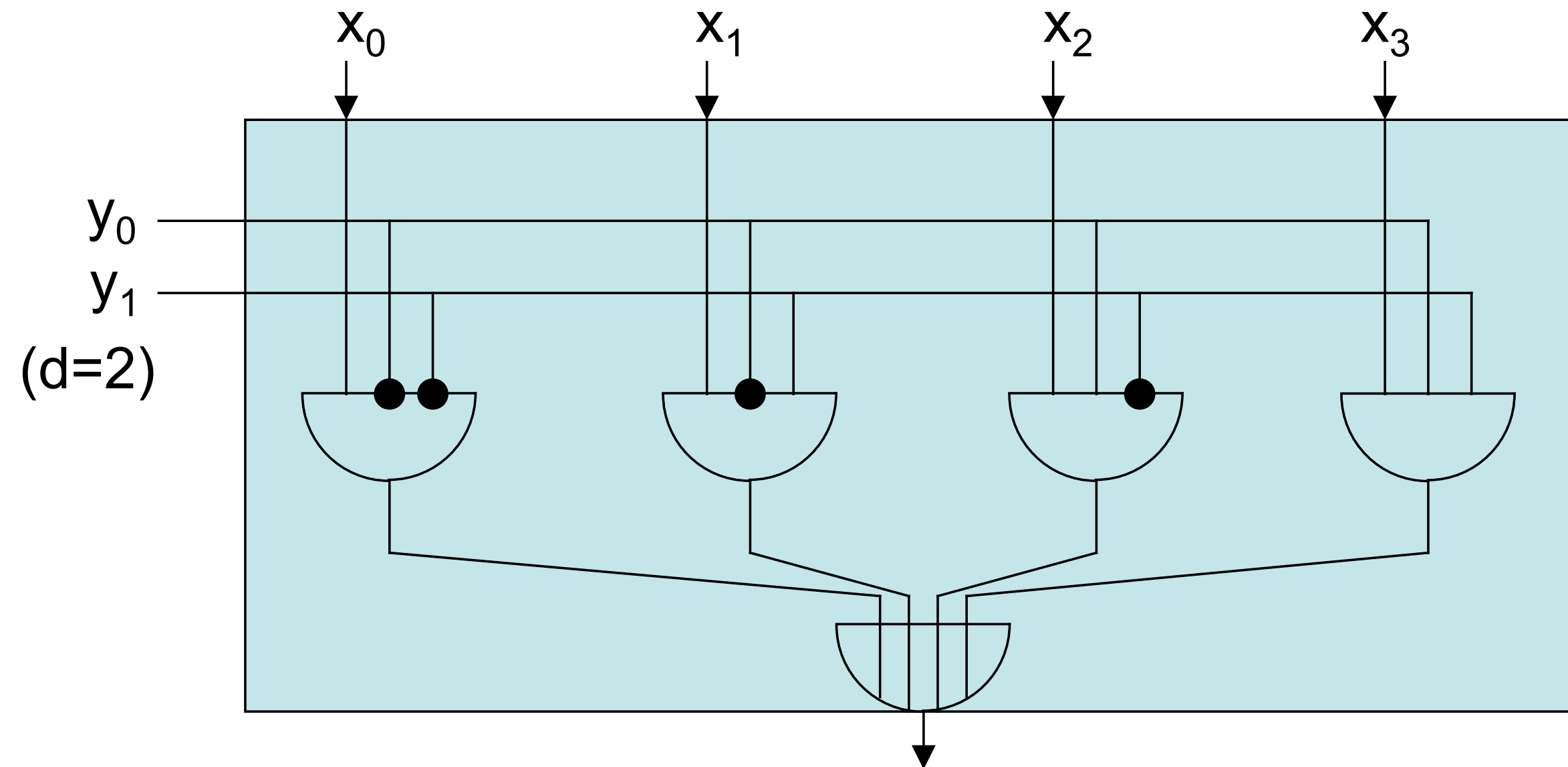
x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

1-MUX



x_0	x_1	y_1	z
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

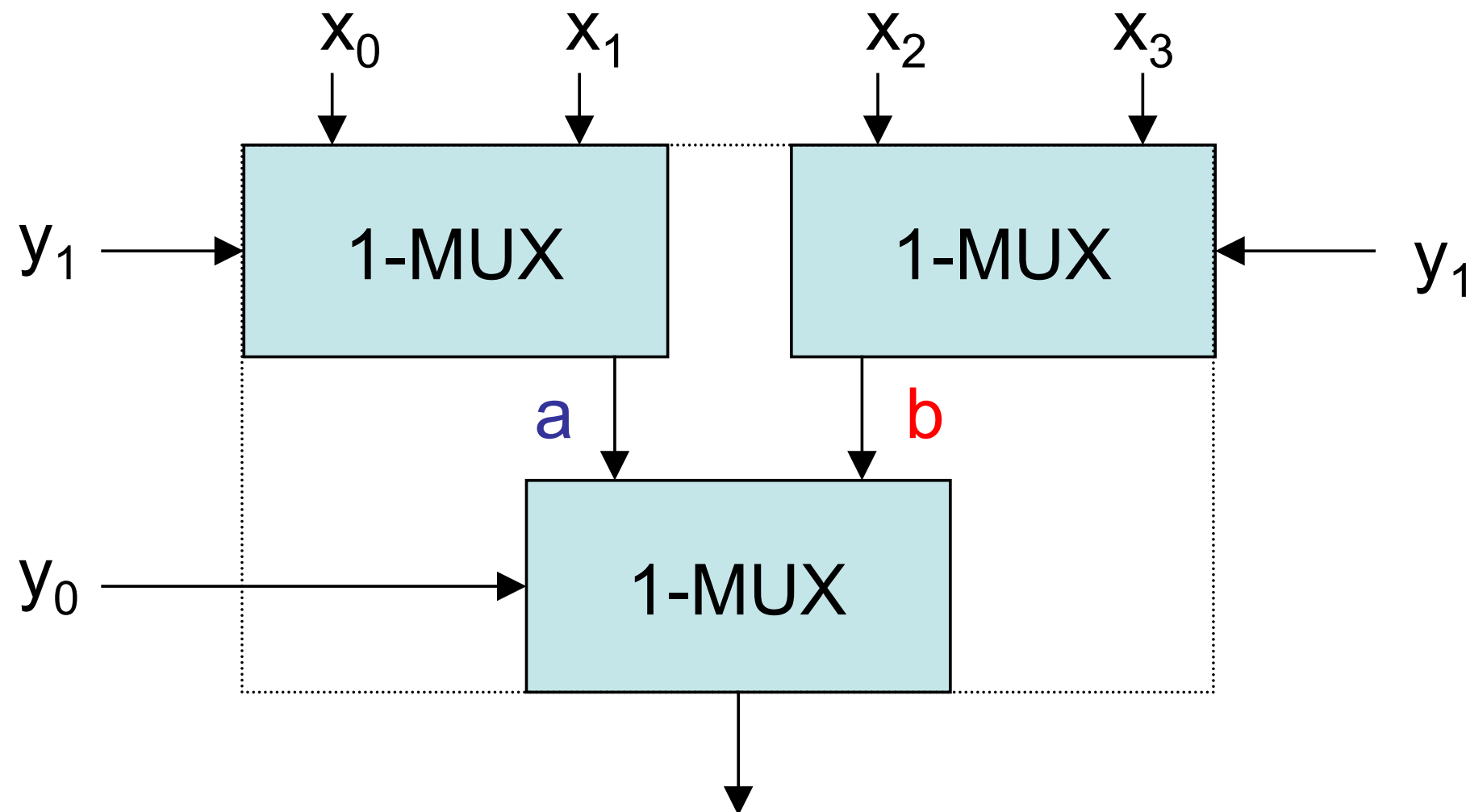
Bottom-Up 2-Multiplexer-Realisierung



$$z = x_0 \neg y_0 \neg y_1 + x_1 \neg y_0 y_1 + x_2 y_0 \neg y_1 + x_3 y_0 y_1$$

> hoher Fan-In: ODER-Gatter: 2^d , UND-Gatter: $d + 1$

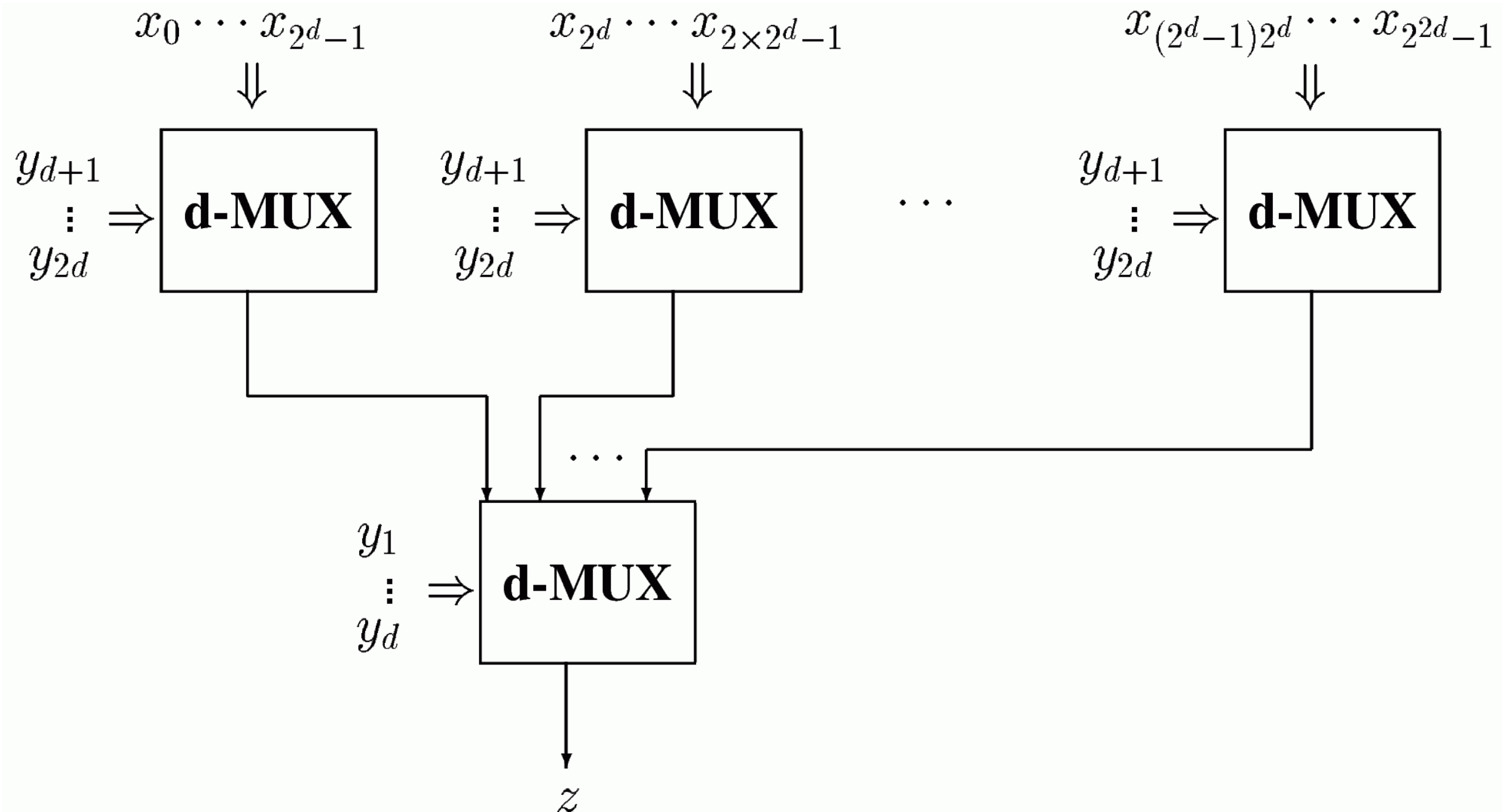
Top-Down 2-Multiplexer-Realisierung



$$\begin{aligned}
 z &= \neg y_0 a + y_0 b = \neg y_0 (\neg y_1 x_0 + y_1 x_1) + y_0 (\neg y_1 x_2 + y_1 x_3) \\
 &= x_0 \neg y_0 \neg y_1 + x_1 \neg y_0 y_1 + x_2 y_0 \neg y_1 + x_3 y_0 y_1
 \end{aligned}$$

> höhere Anzahl Stufen!

Top-Down-Multiplexer-Entwurf

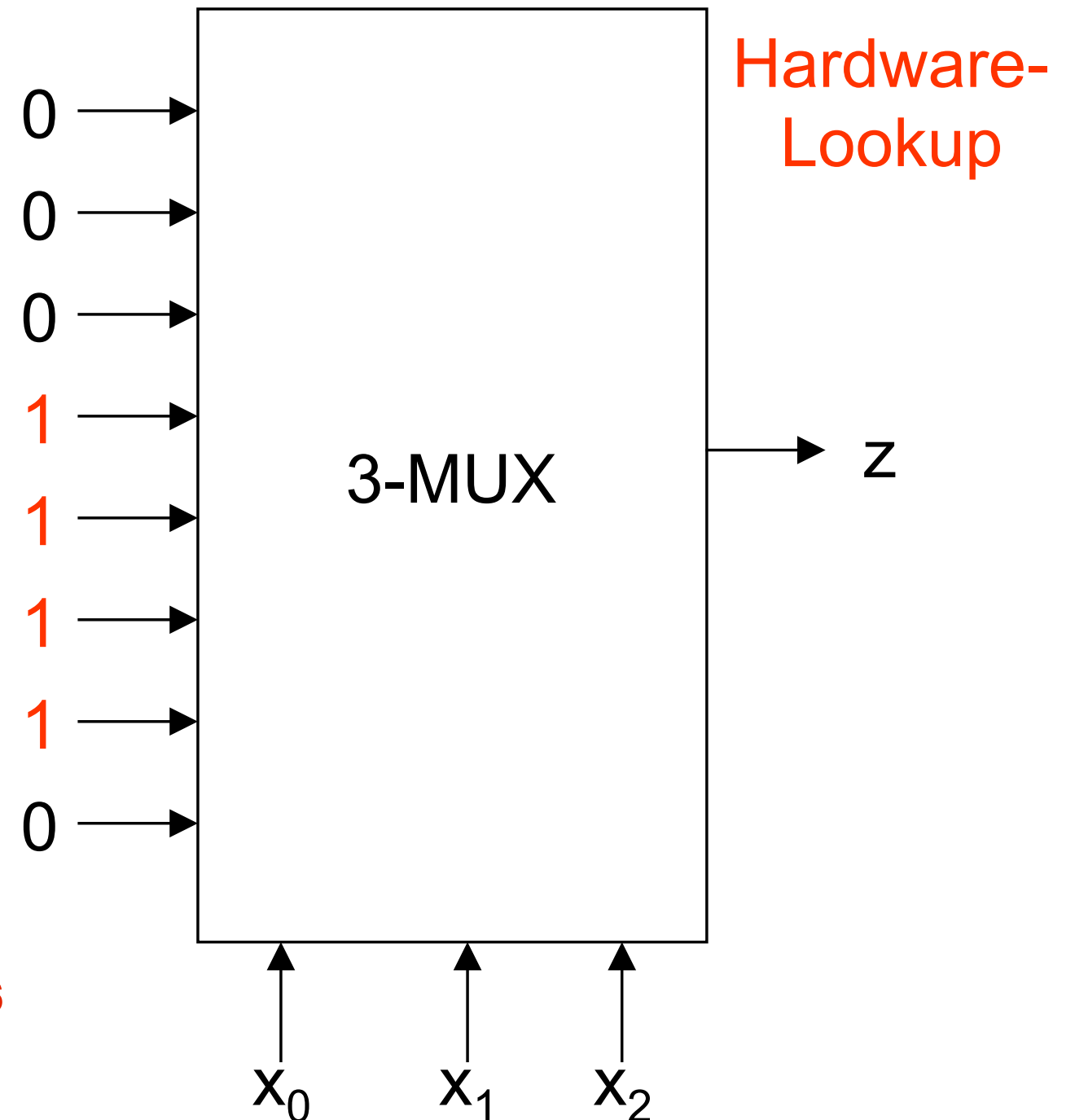


Multiplexer für Boolesche Funktionen I

x_0	x_1	x_2	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

> $f: B^3 \rightarrow B$

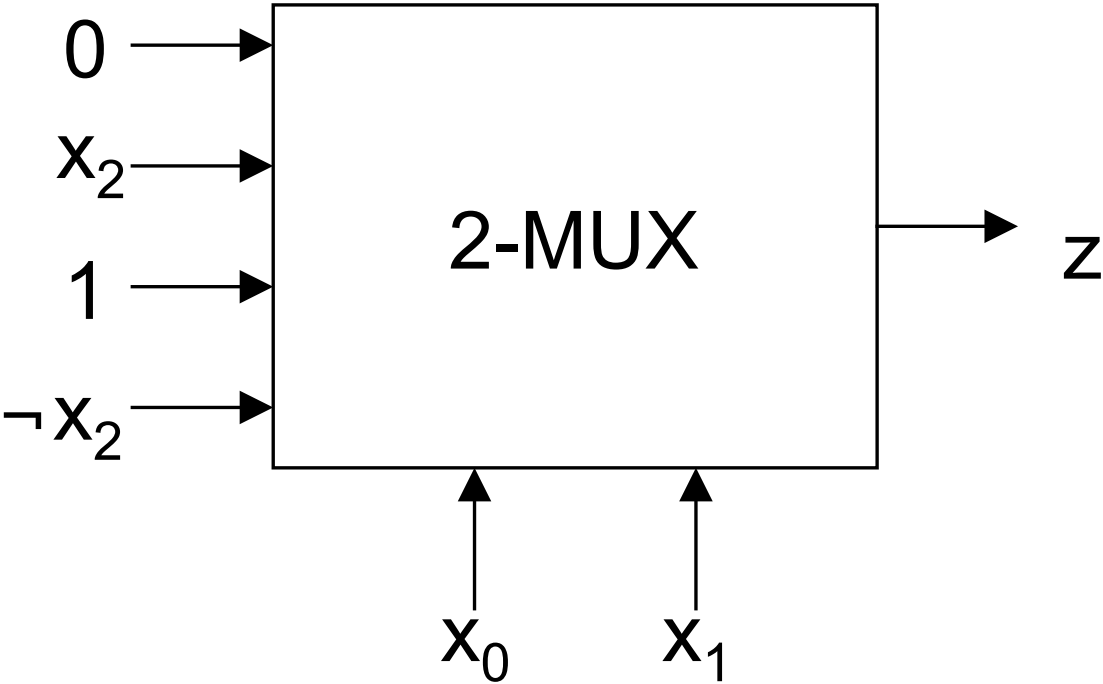
$$f(x_0, x_1, x_2) = m_3 + m_4 + m_5 + m_6$$



Multiplexer für Boolesche Funktionen II

x_0	x_1	x_2	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

x_0	x_1	z
0	0	0
0	1	x_2
1	0	1
1	1	$\neg x_2$



Demultiplexer

> d-MUX

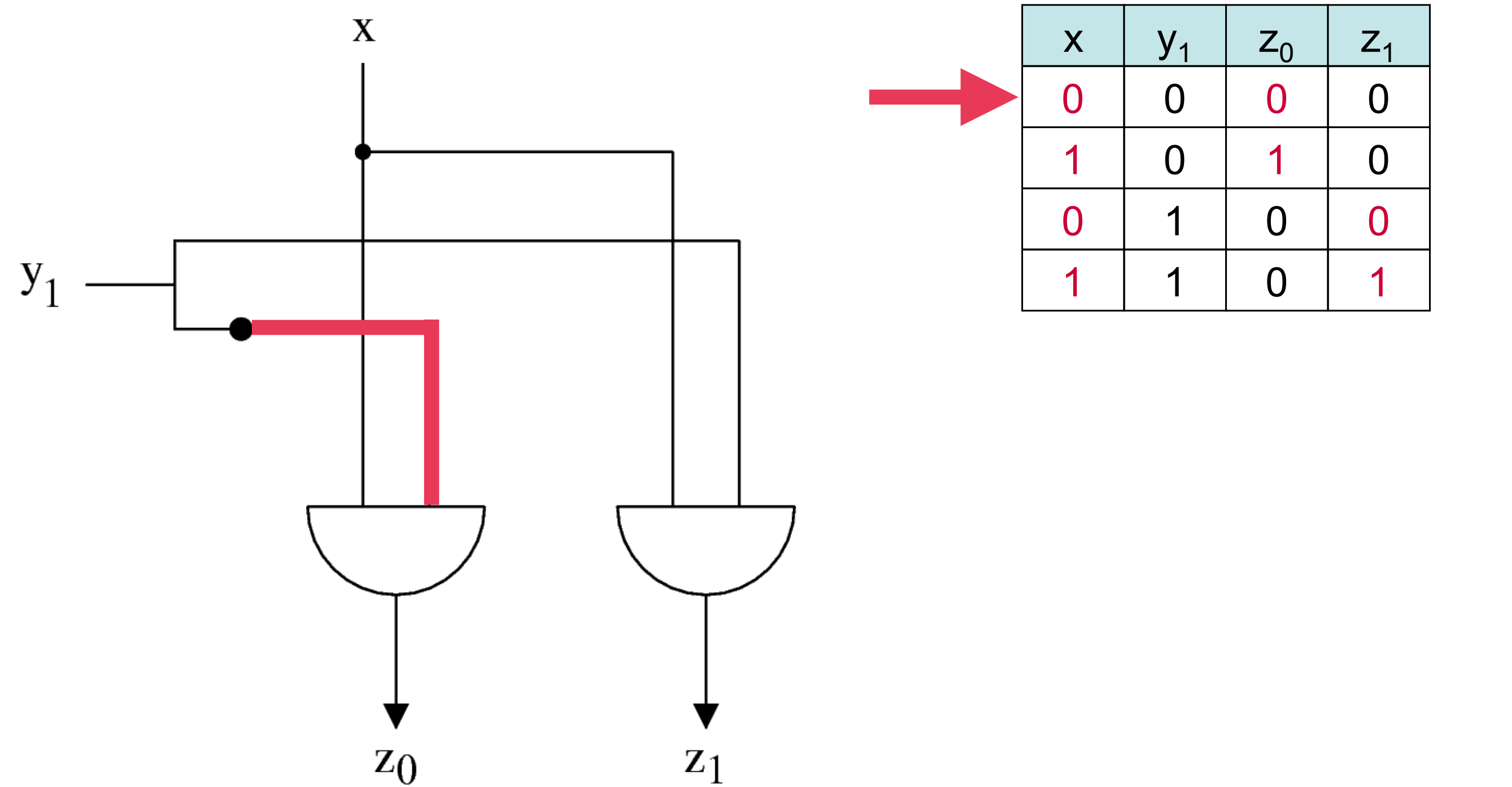
- 2^d Daten-Inputs $x_0, x_1, \dots, x_{2^d-1}$
- d Steuersignale y_0, \dots, y_{d-1}
- 1 Output z mit

$$z = \sum_{i=0}^{2^d-1} x_i \cdot m_i(y_0, y_1, \dots, y_{d-1})$$

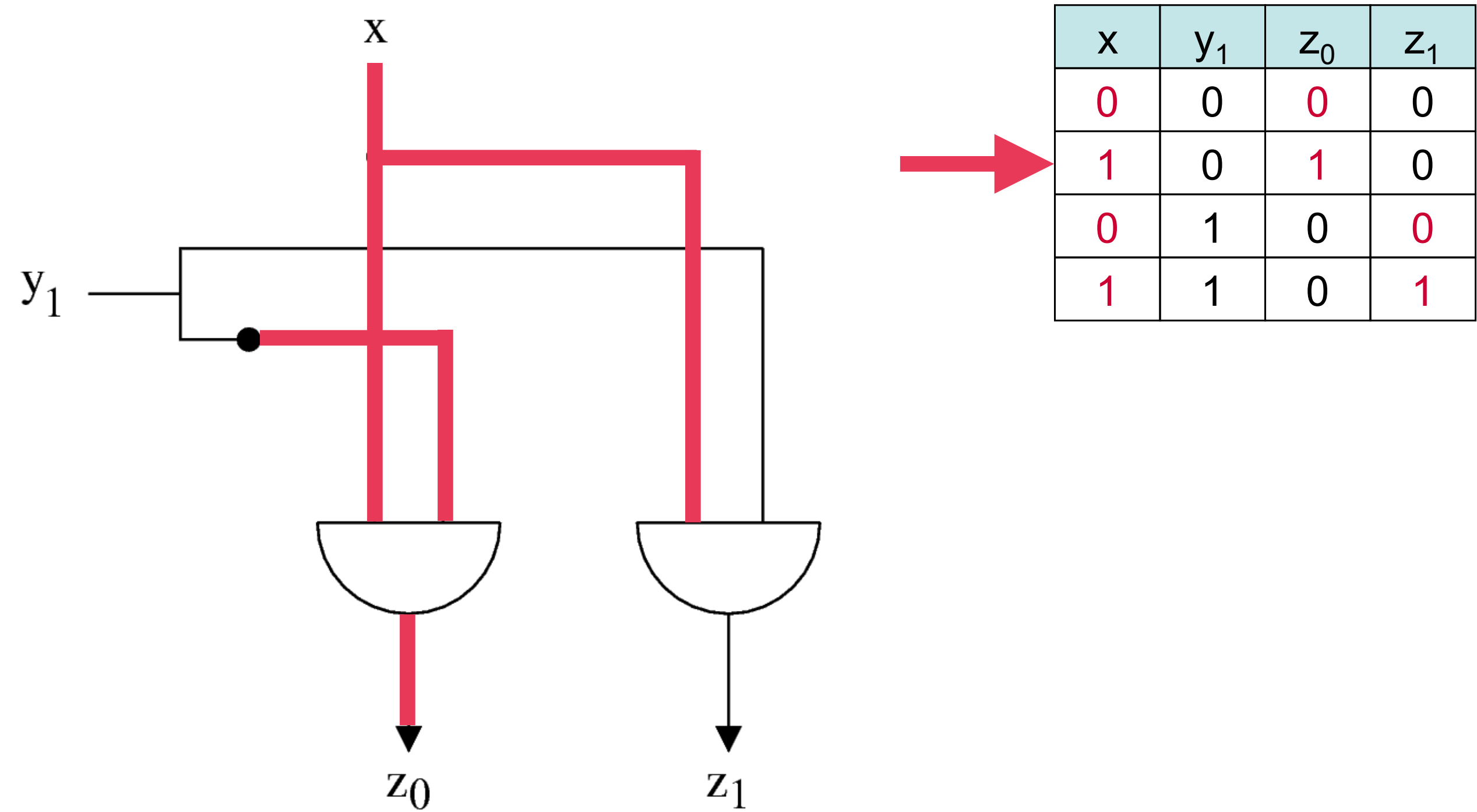
> d-DeMUX

- 1 Daten-Input x
- d Steuersignale y_0, \dots, y_{d-1}
- 2^d Outputs $z_0, z_1, \dots, z_{2^d-1}$ mit $z_i = x \cdot m_i(y_0, y_1, \dots, y_{d-1})$
- Steuersignale legen fest, auf welchem Output das Input-Signal gelegt wird.

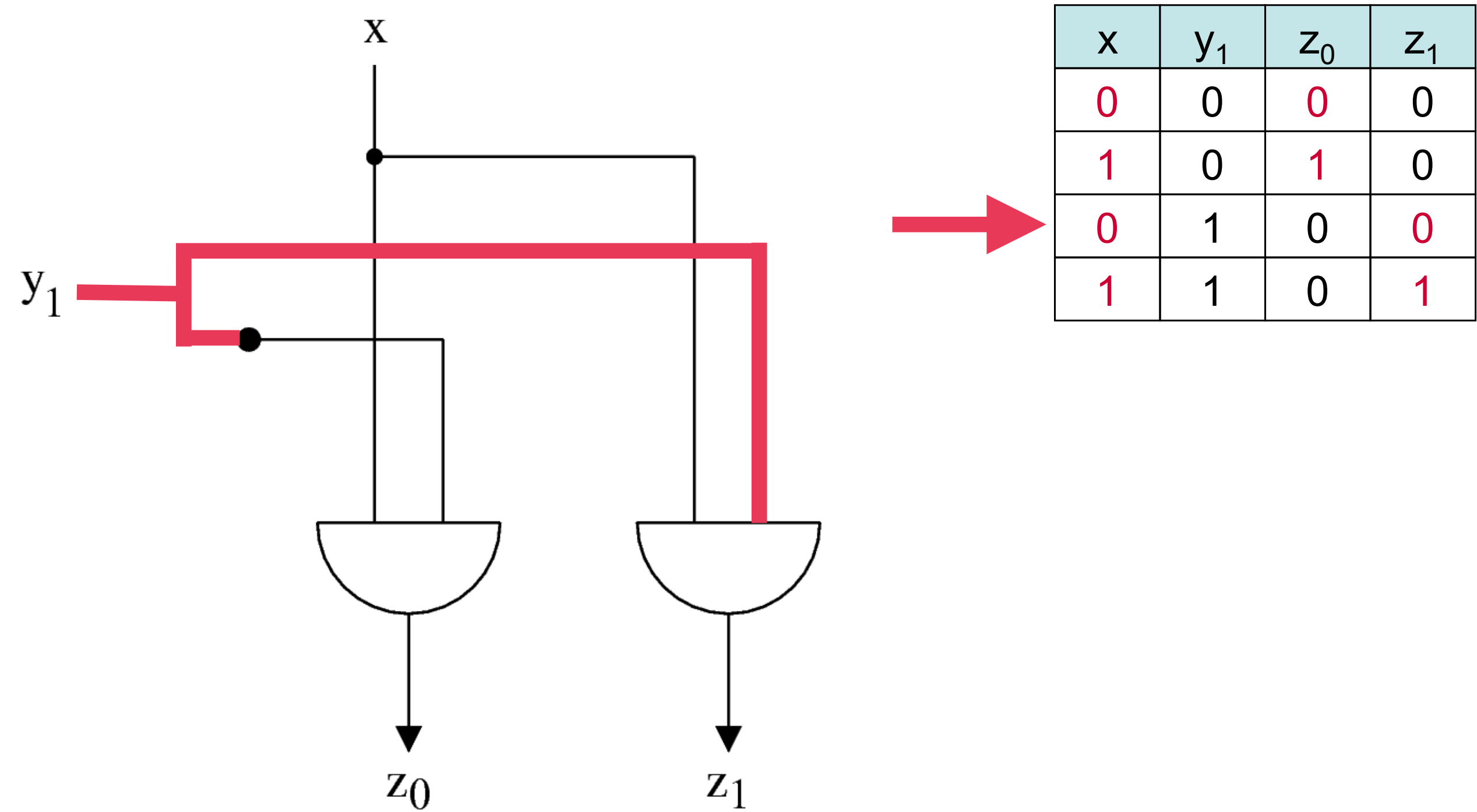
1-DeMUX



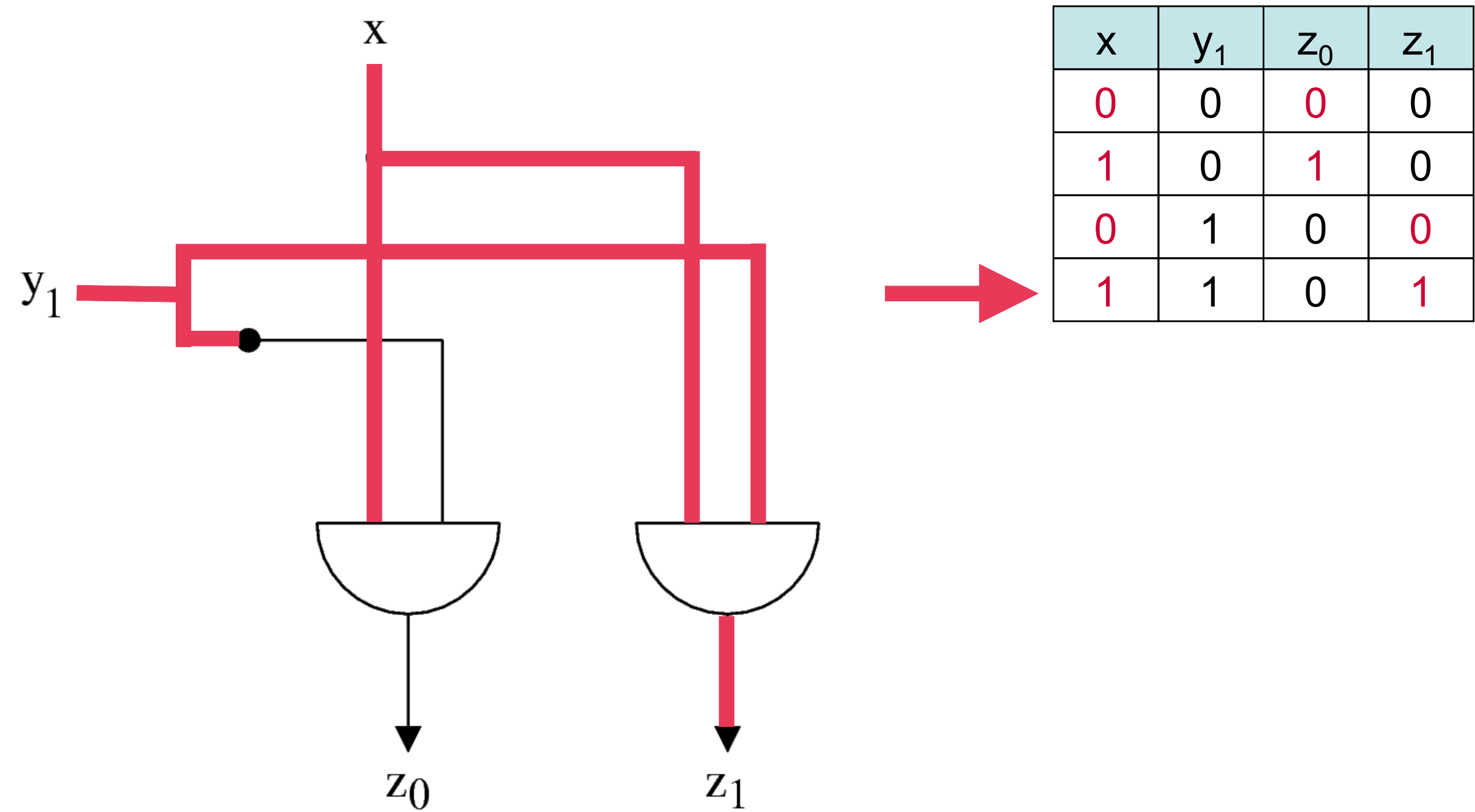
1-DeMUX



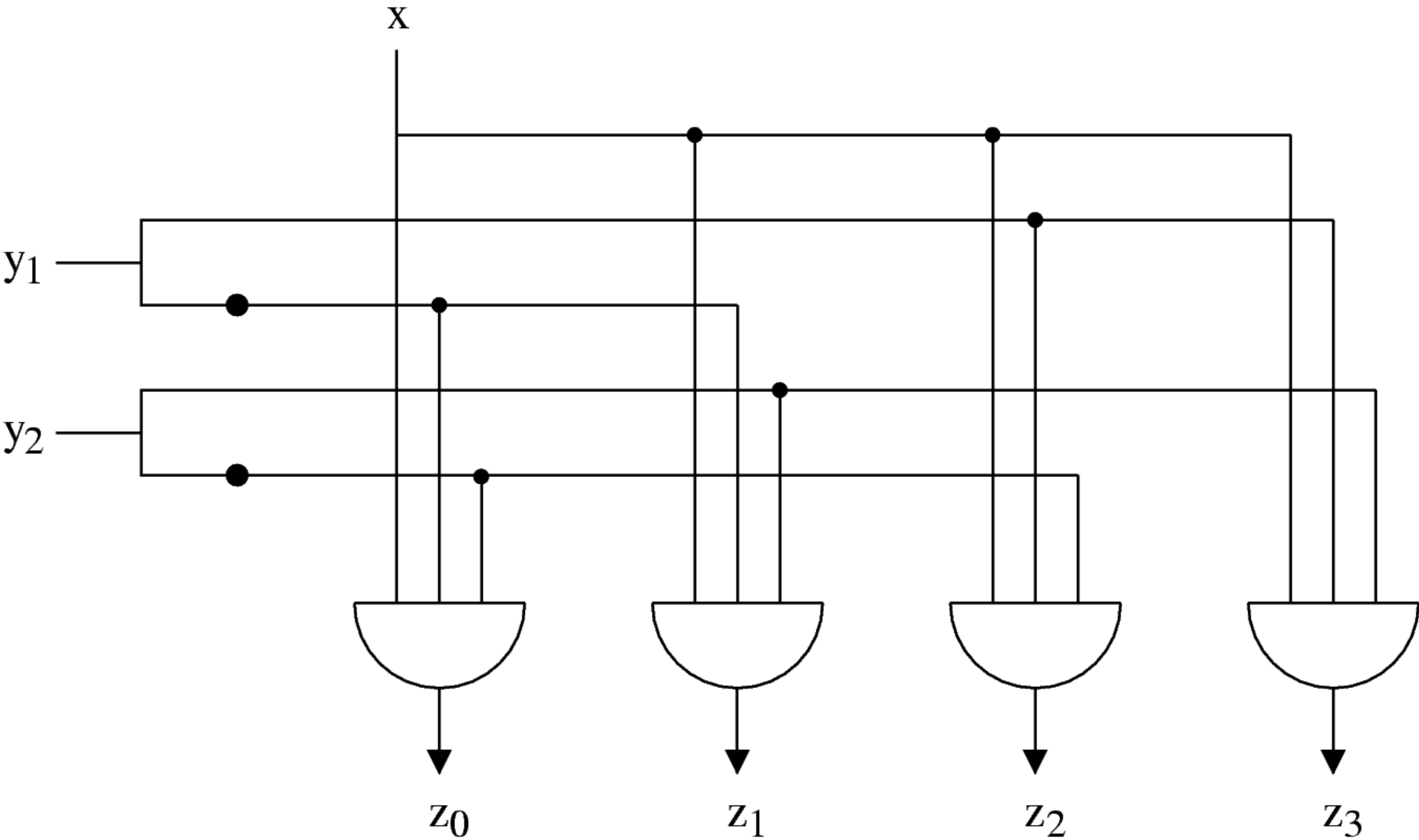
1-DeMUX



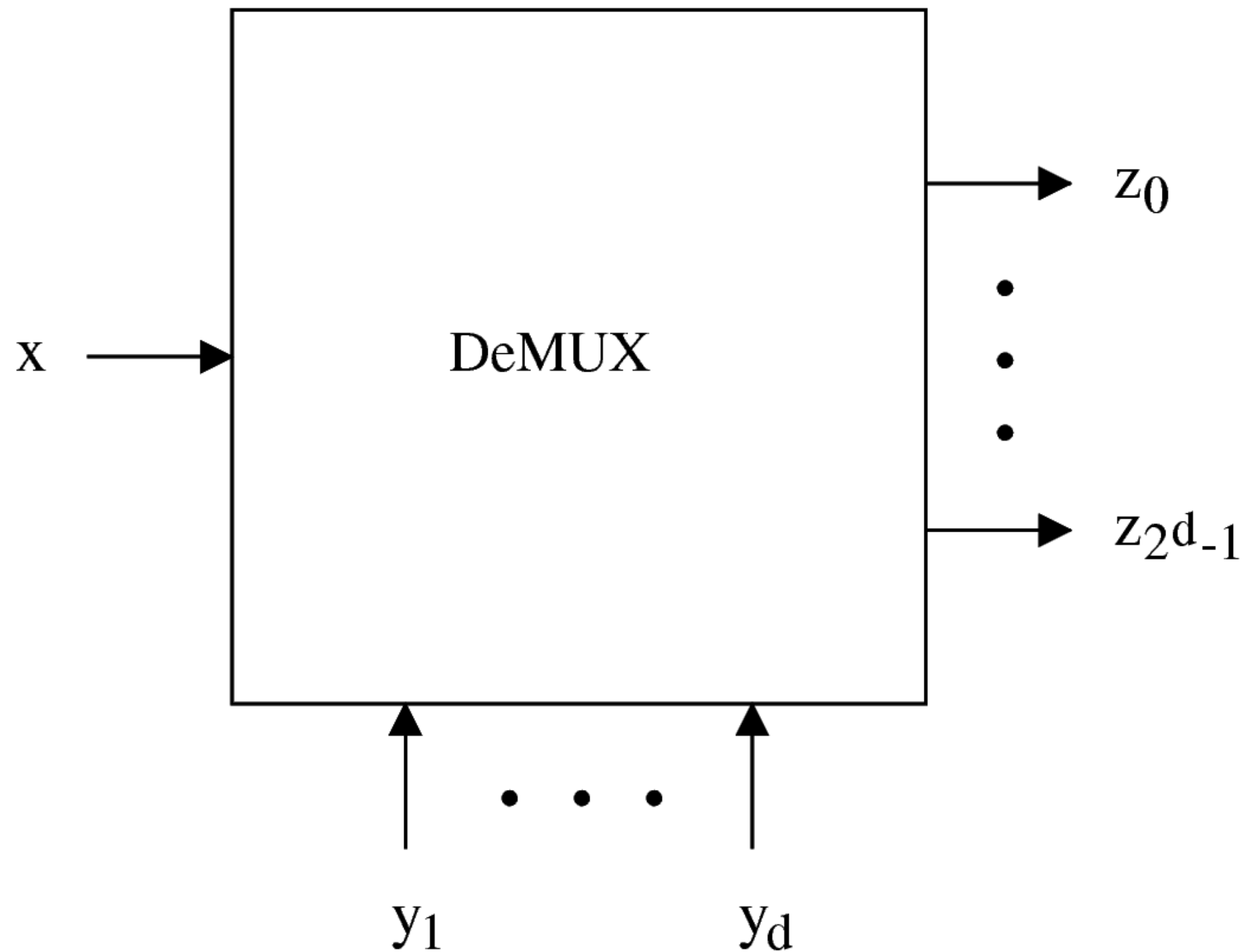
1-DeMUX



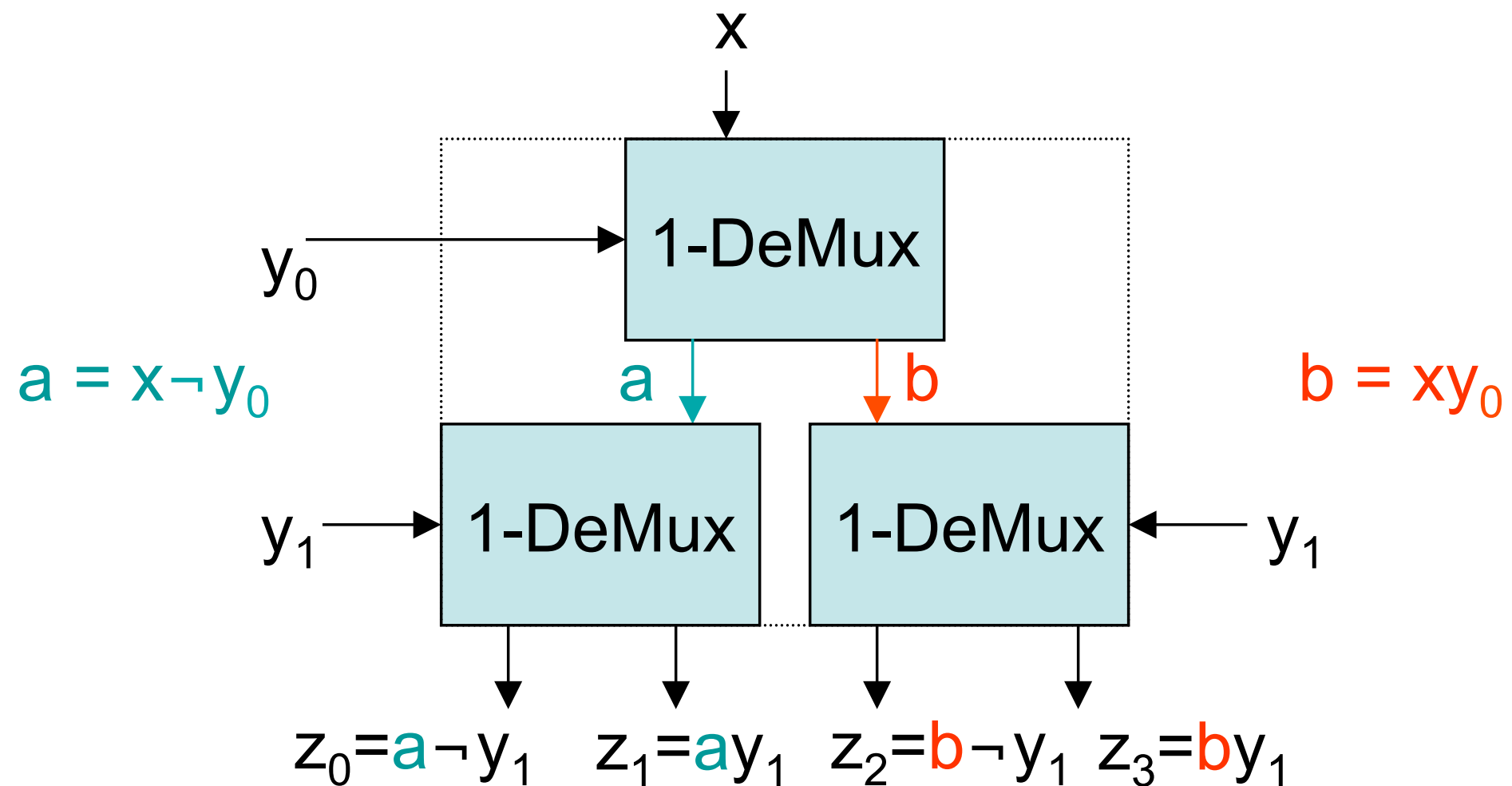
2-DeMUX



Allgemeiner Aufbau eines DeMUX



2-DeMUX Realisierung (Top-Down)



$$z_0 = x \neg y_0 \neg y_1$$

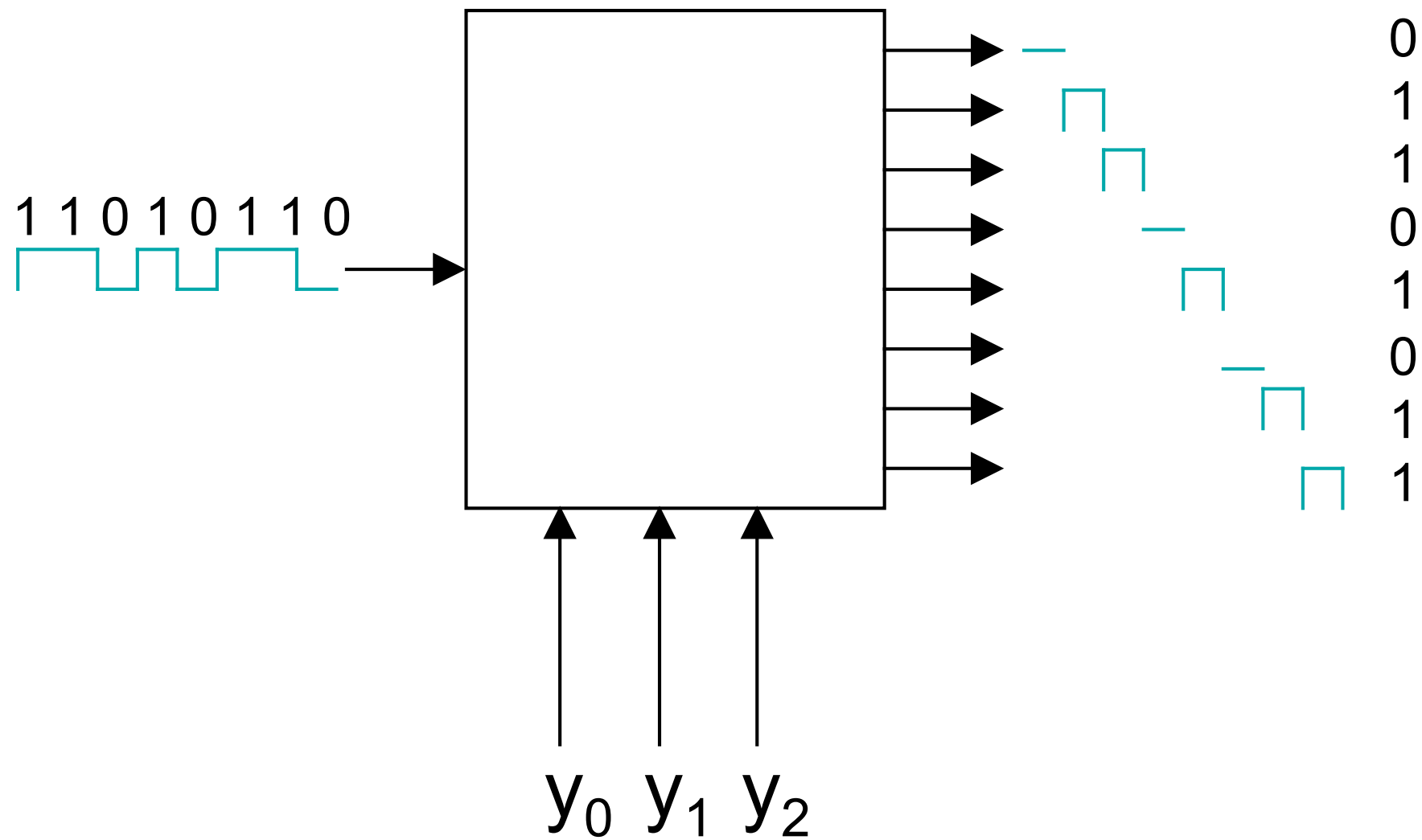
$$z_1 = x \neg y_0 y_1$$

$$z_2 = xy_0 \neg y_1$$

$$z_3 = xy_0 y_1$$

Anwendungsbeispiele

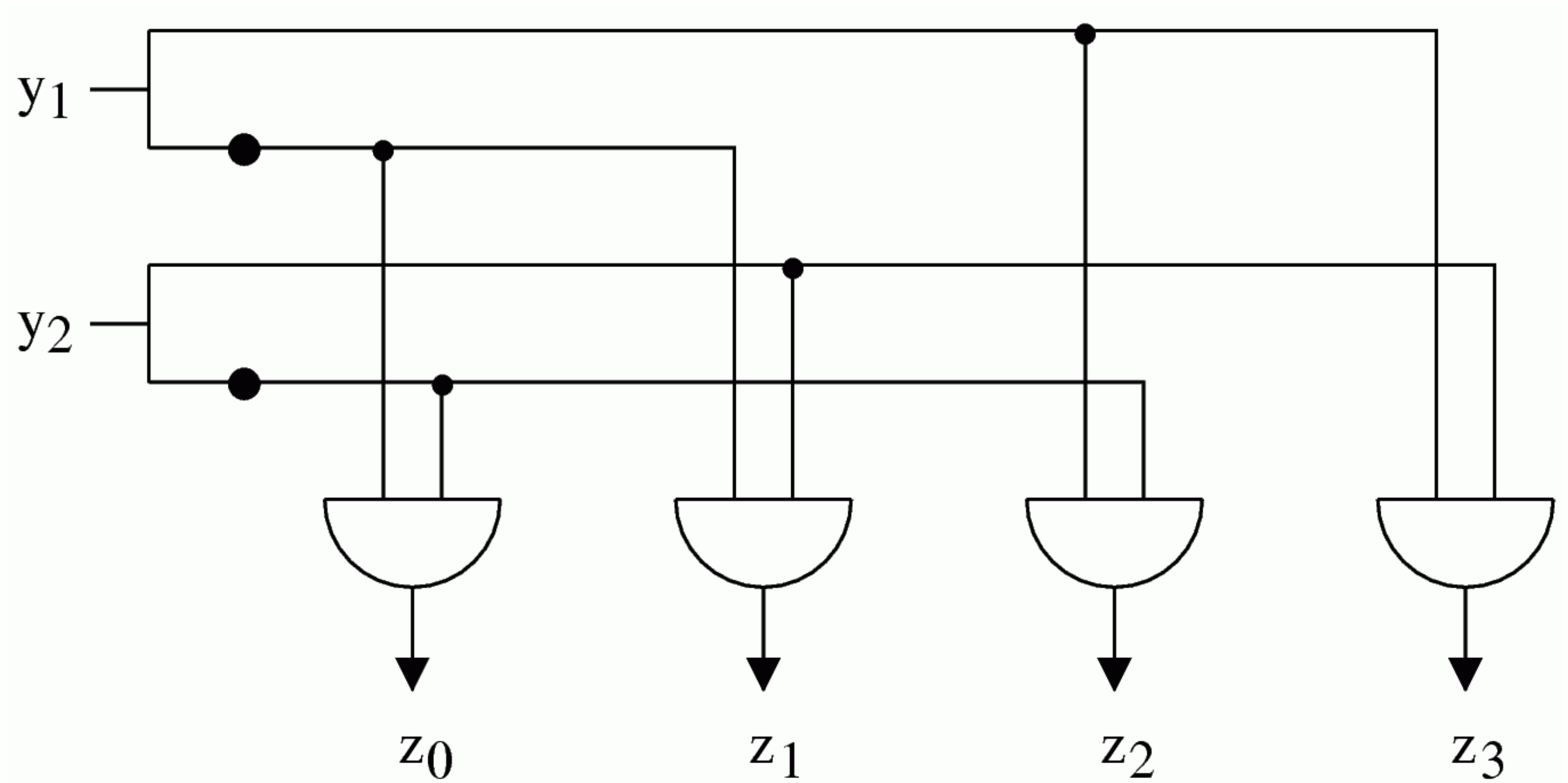
- > De-Serialisierung
- > Zeit-Demultiplexer



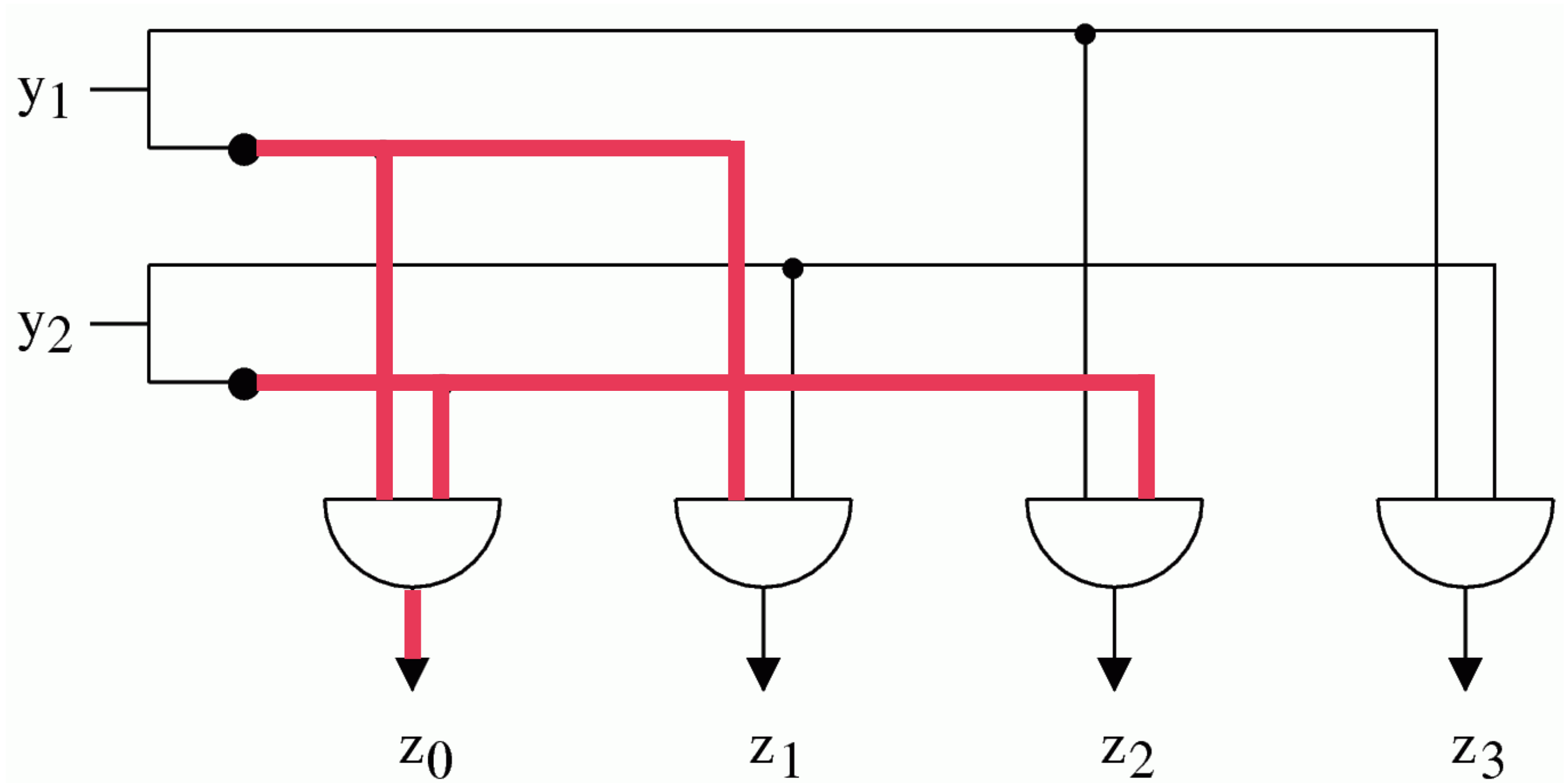
Decoder / Encoder

- > Decoder
 - Demultiplexer-Variante
 - Realisierung wie Demultiplexer, aber $x = 1$
 - x als Aktivierungssignal (Ein- / Ausschalter)
 - genau ein Output auf 1
- > Encoder
 - Umkehrfunktion des Decoders

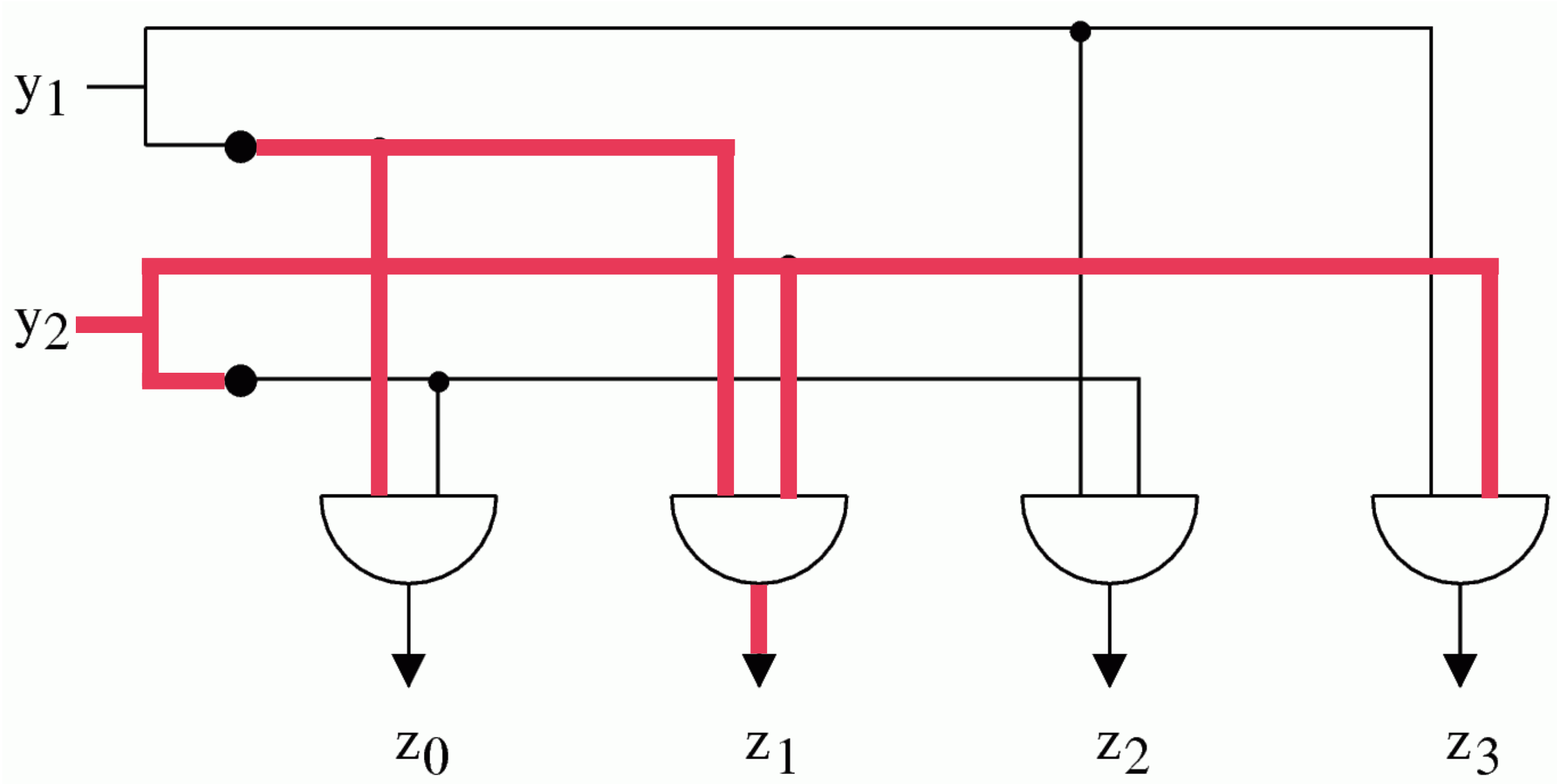
2 x 4 Decoder



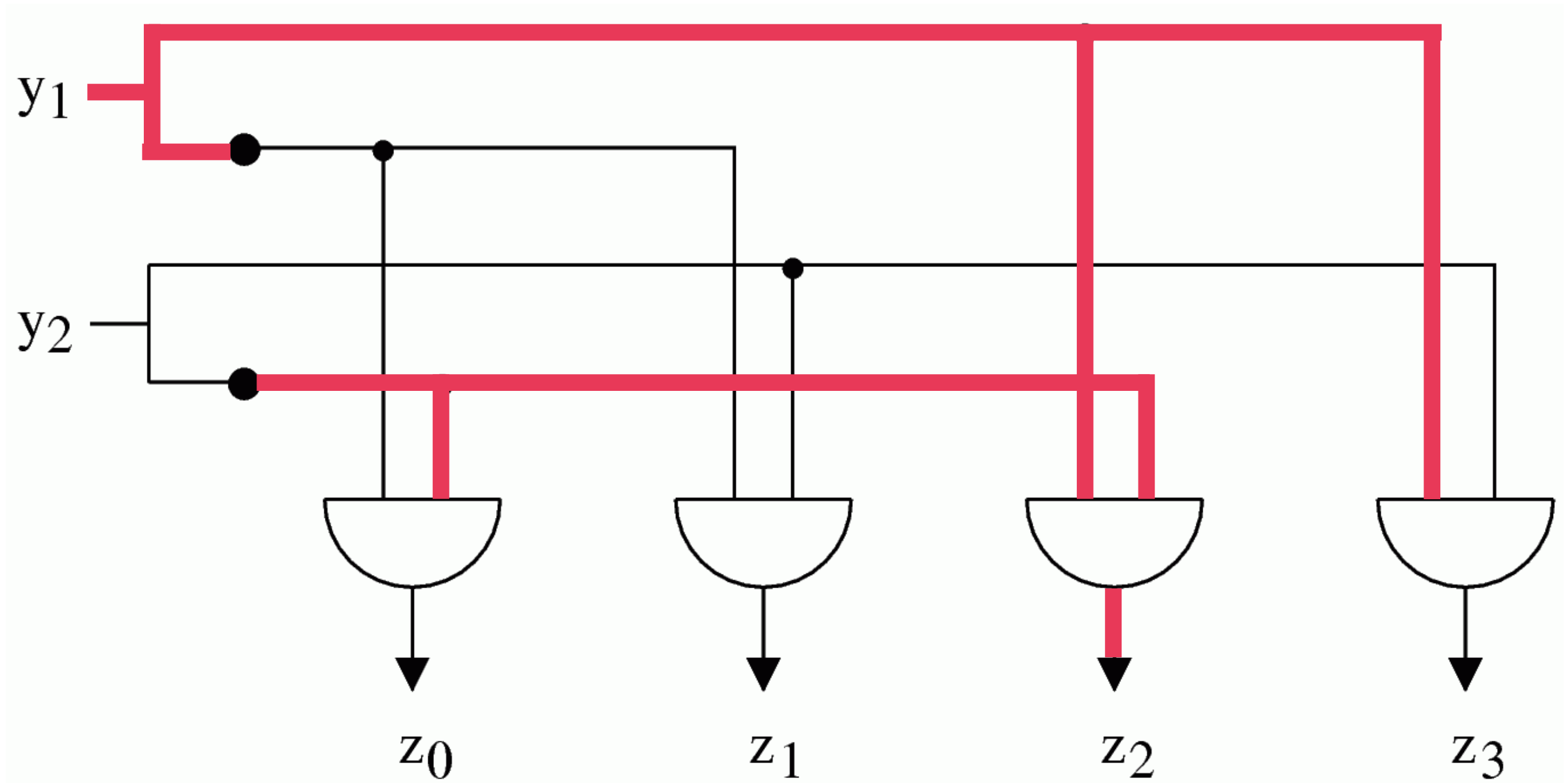
2 x 4 Decoder



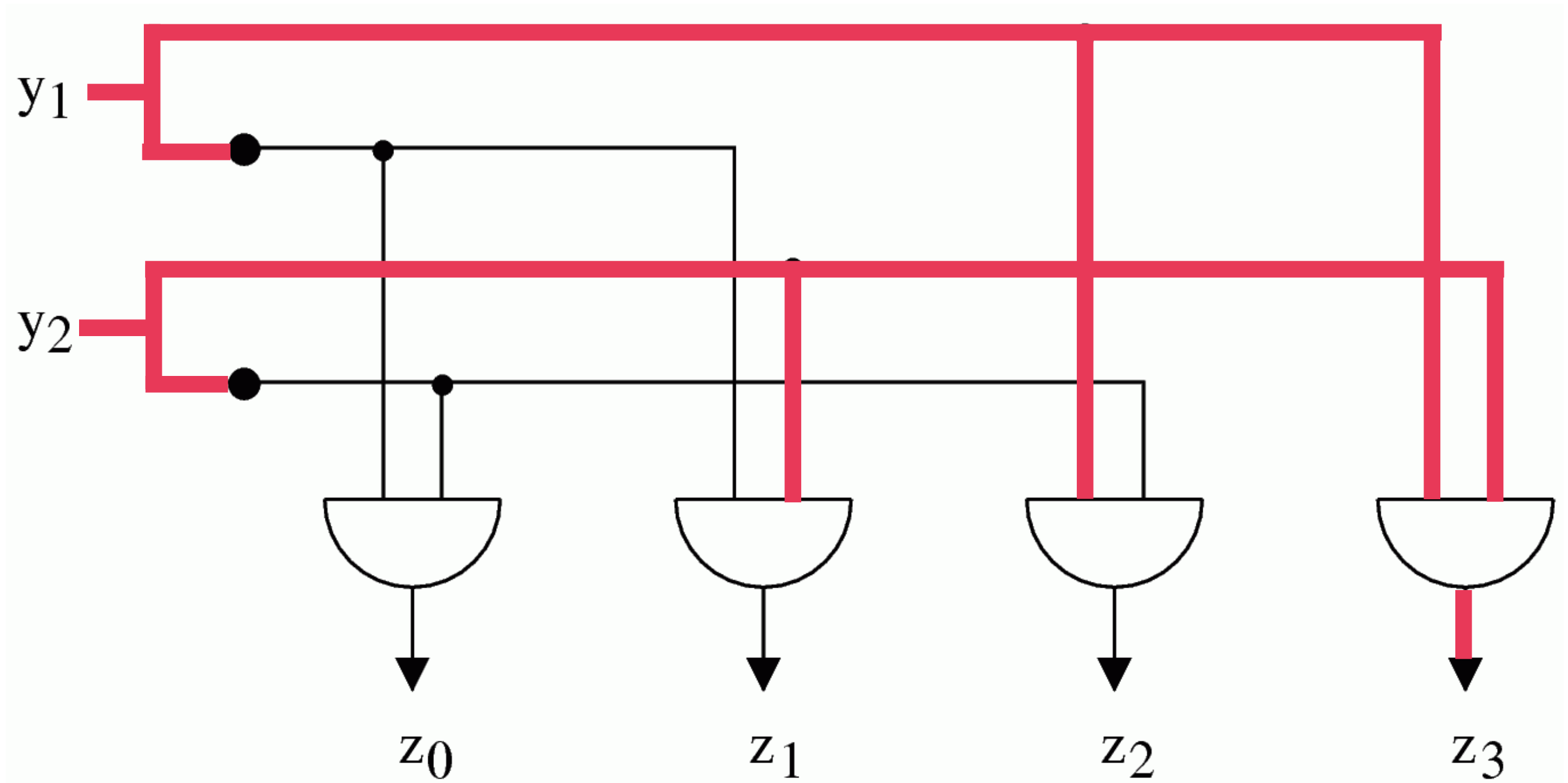
2 x 4 Decoder




2 x 4 Decoder



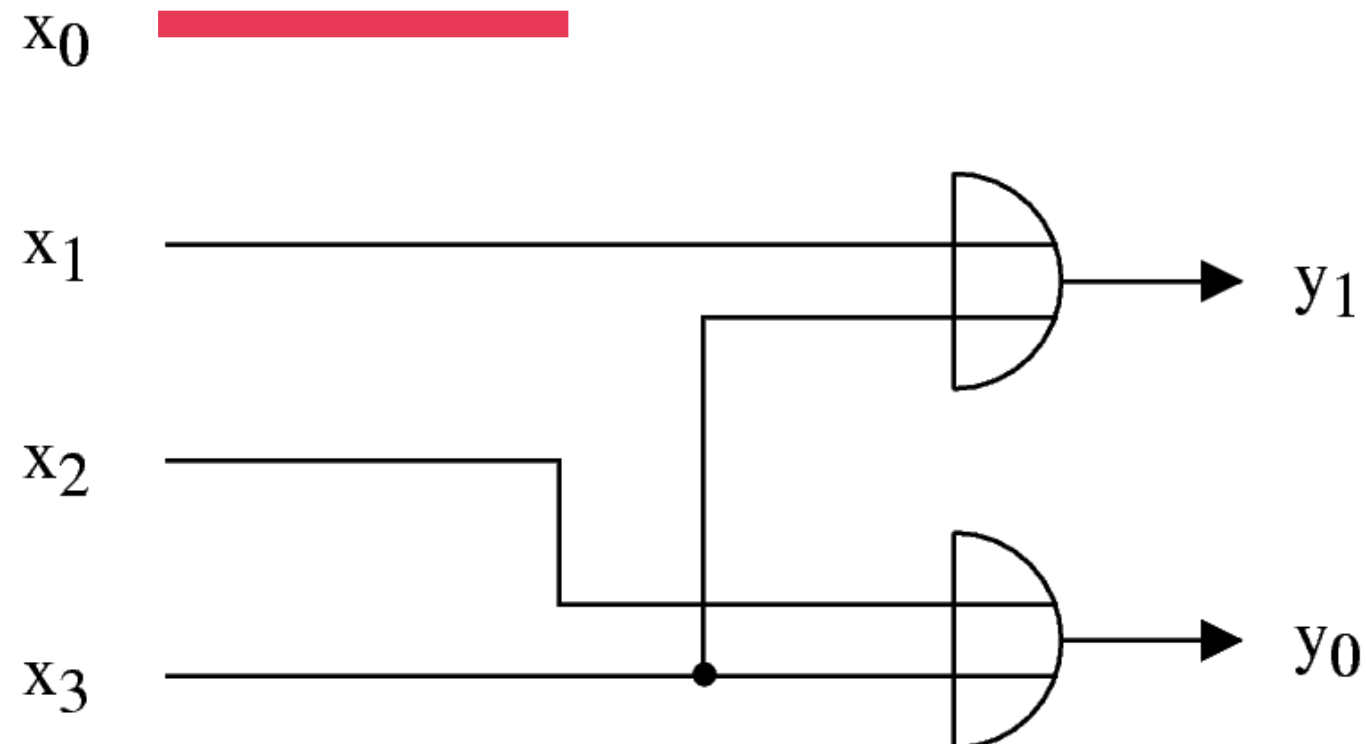
2 x 4 Decoder



4 x 2 Encoder

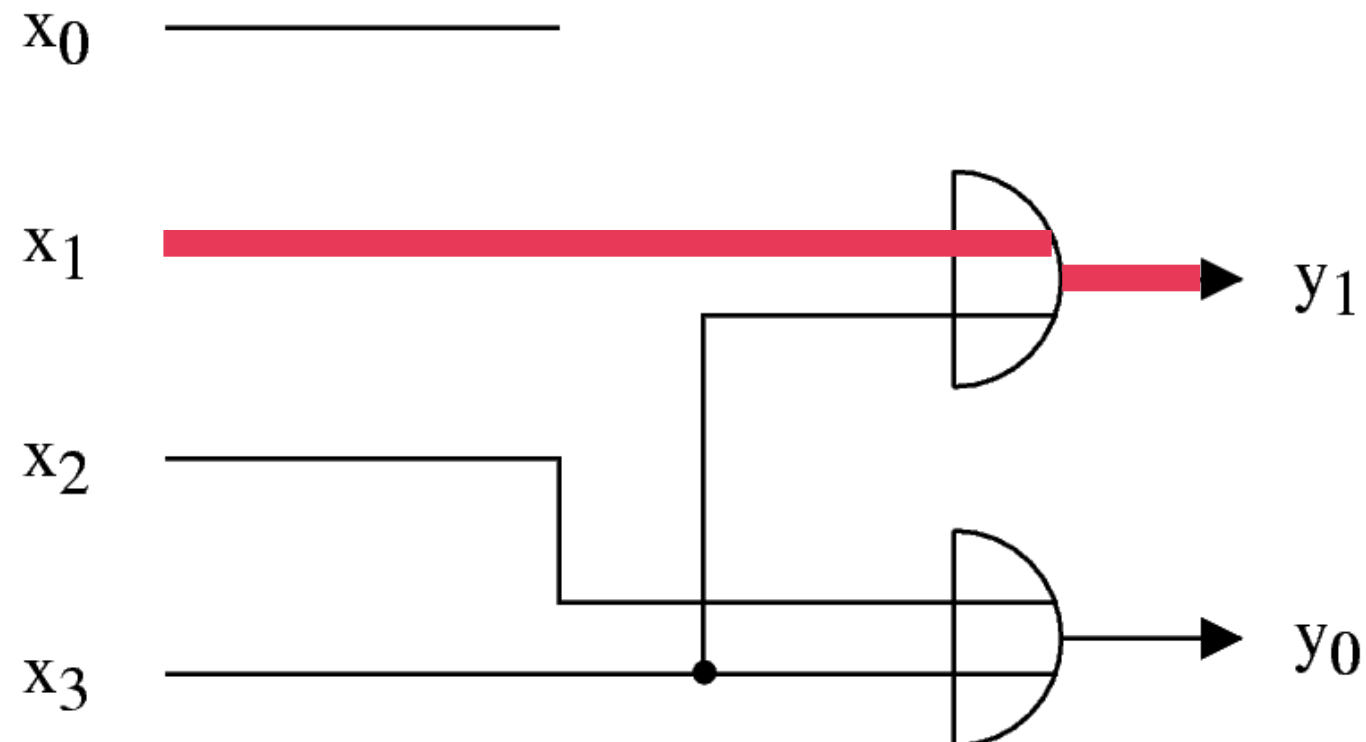


x_0	x_1	x_2	x_3	y_0	y_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



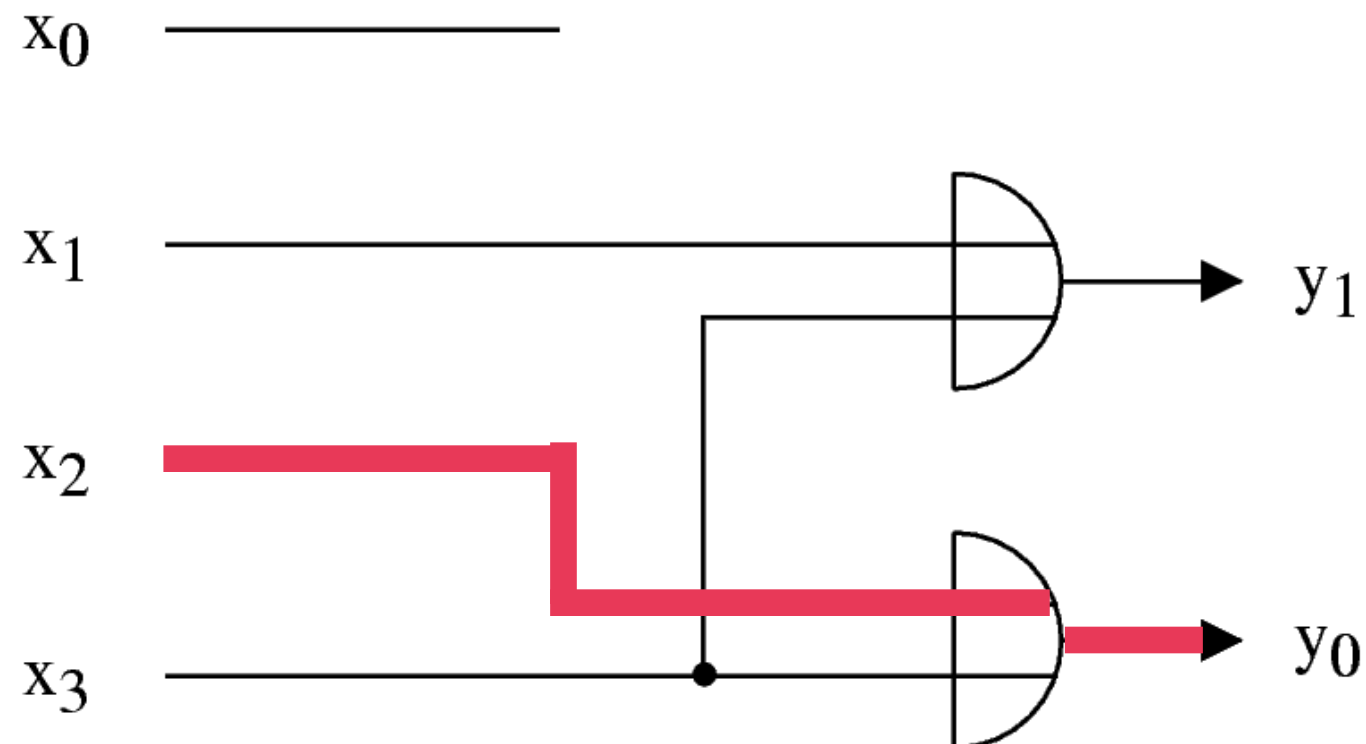
4 x 2 Encoder

	x_0	x_1	x_2	x_3	y_0	y_1
	1	0	0	0	0	0
→	0	1	0	0	0	1
	0	0	1	0	1	0
	0	0	0	1	1	1



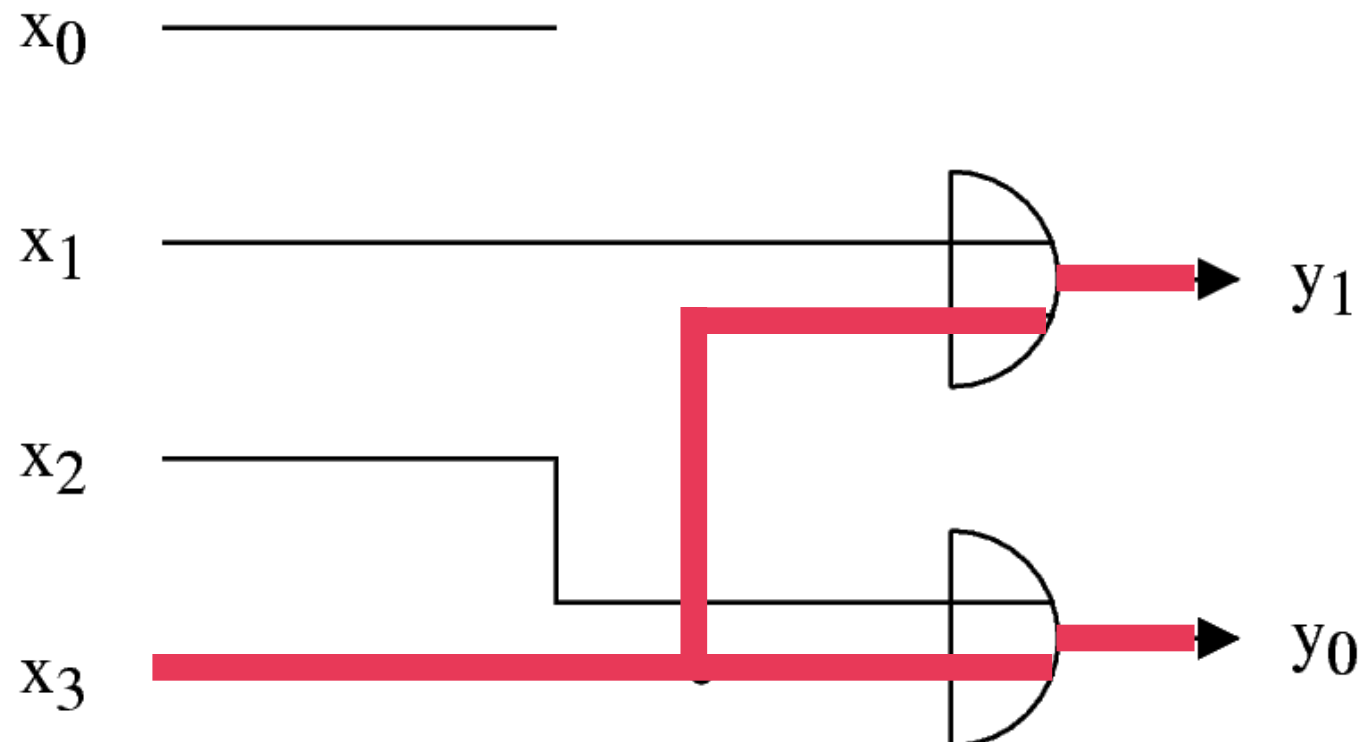
4 x 2 Encoder

	x_0	x_1	x_2	x_3	y_0	y_1
	1	0	0	0	0	0
	0	1	0	0	0	1
→	0	0	1	0	1	0
	0	0	0	1	1	1



4 x 2 Encoder

x_0	x_1	x_2	x_3	y_0	y_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



Realisierung Boolescher Funktionen

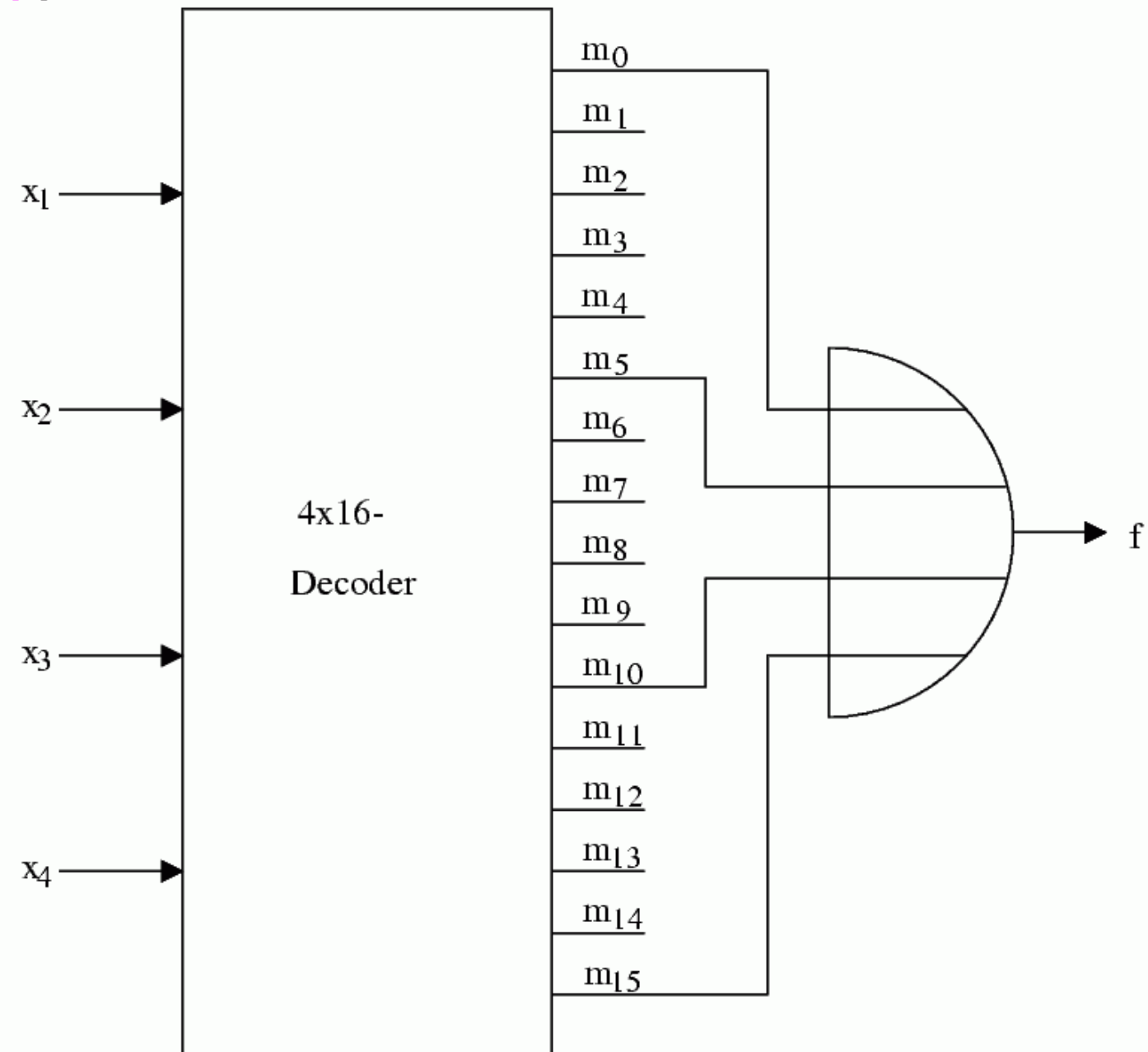
Beispiel:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4 + x_1x_2x_3x_4 + x_1\bar{x}_2x_3\bar{x}_4 :$$

1. mittels MUX: siehe oben

Realisierung Boolescher Funktionen (2)

2. mittels Decoder:



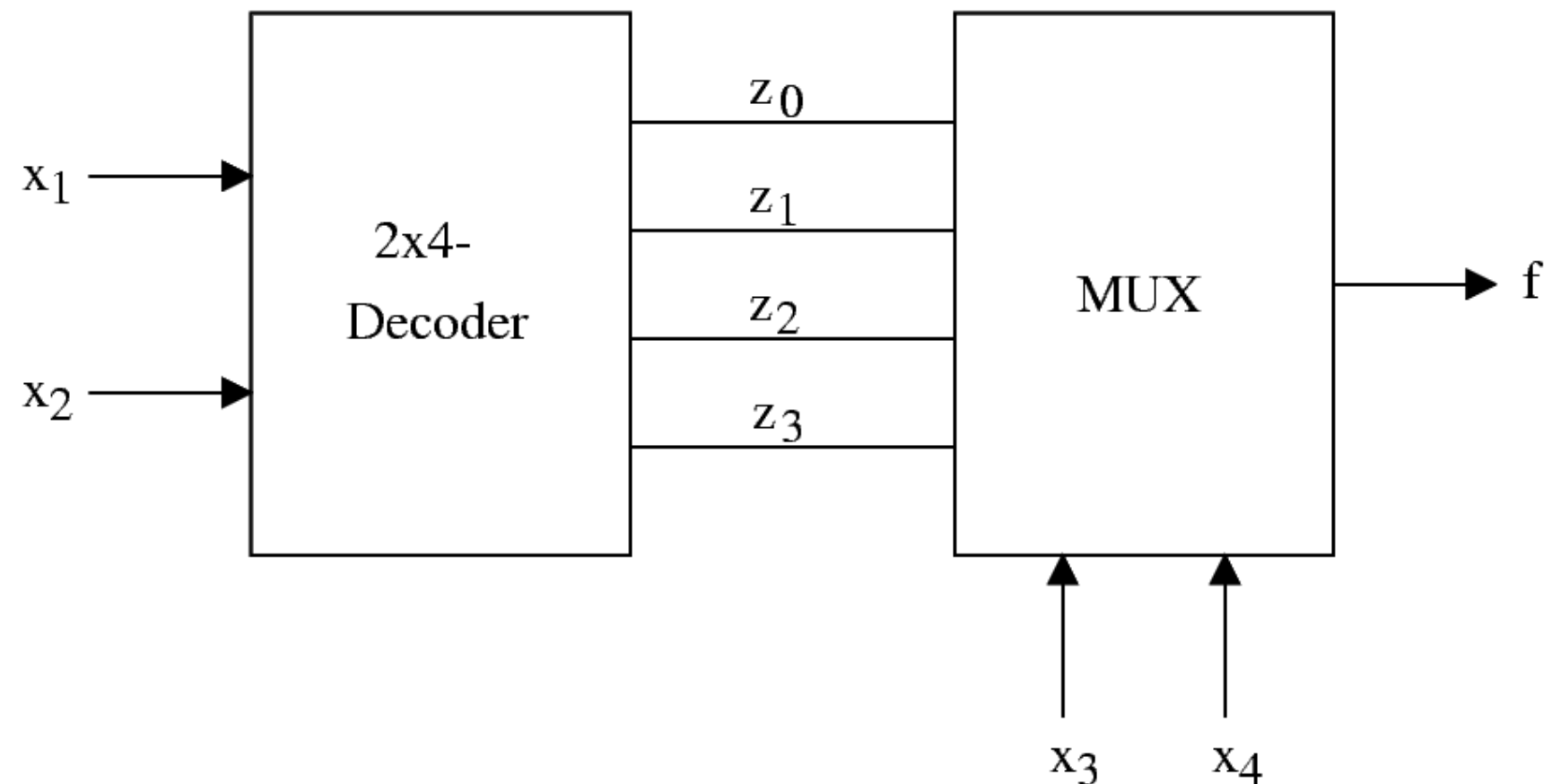
Realisierung Boolescher Funktionen (3)

3. mittels Kombination von Decoder und MUX:

Es gibt 4 Input-Kombinationen, für welche $f = 1$ gilt:

$x_1x_2 = 00$ und $x_3x_4 = 00$, $x_1x_2 = 01$ und $x_3x_4 = 01$,

$x_1x_2 = 11$ und $x_3x_4 = 11$, $x_1x_2 = 10$ und $x_3x_4 = 10$.



Addiernetze

- > Addition von zwei 16-stelligen Binärzahlen
- > Möglichkeit zur Realisierung
 - Schaltnetz für Schaltfunktion $A: B^{32} \rightarrow B^{17}$
 - 17 Boolesche Funktionen mit 2^{32} möglichen Inputwerten
 - 50% Wahrscheinlichkeit für Wert 1 einer Booleschen Funktion: $17 \cdot 2^{31}$ **einschlägige Minterme**
- > Anderer Ansatz: **Top-Down**

Schriftliche Addition

Binär-Addition

$$\begin{array}{r}
 10110111 \\
 +11000101 \\
 \hline
 101111100
 \end{array}$$

Diagram illustrating binary addition. The first number is 10110111, the second is 11000101. The result is 101111100. The carry bit 1 is shown on the left. The result is shown below a dashed line. The carry bit 1 is shown below the first number. The result is shown below a dashed line. The carry bit 1 is shown below the first number.

- > Top-Down:
Schaltung, welche das Resultat an einer Stelle ausrechnet.
- > Wie bei schriftlicher Addition
 1. Letzte Stelle ausrechnen
 2. Zweitletzte Stelle ausrechnen
 3. ...
- ➔ Letzte Stelle: Halbaddierer
- ➔ Beliebige Stelle: Volladdierer

Dual-Addition

> Letzte Stelle

— Eingabe

- 2 Dual-Ziffern

— Ausgabe

- Ergebnis $R = x \oplus y$
- Übertrag $U = x \cdot y$

x	y	R	U
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

> Beliebige andere Stelle

— Eingabe

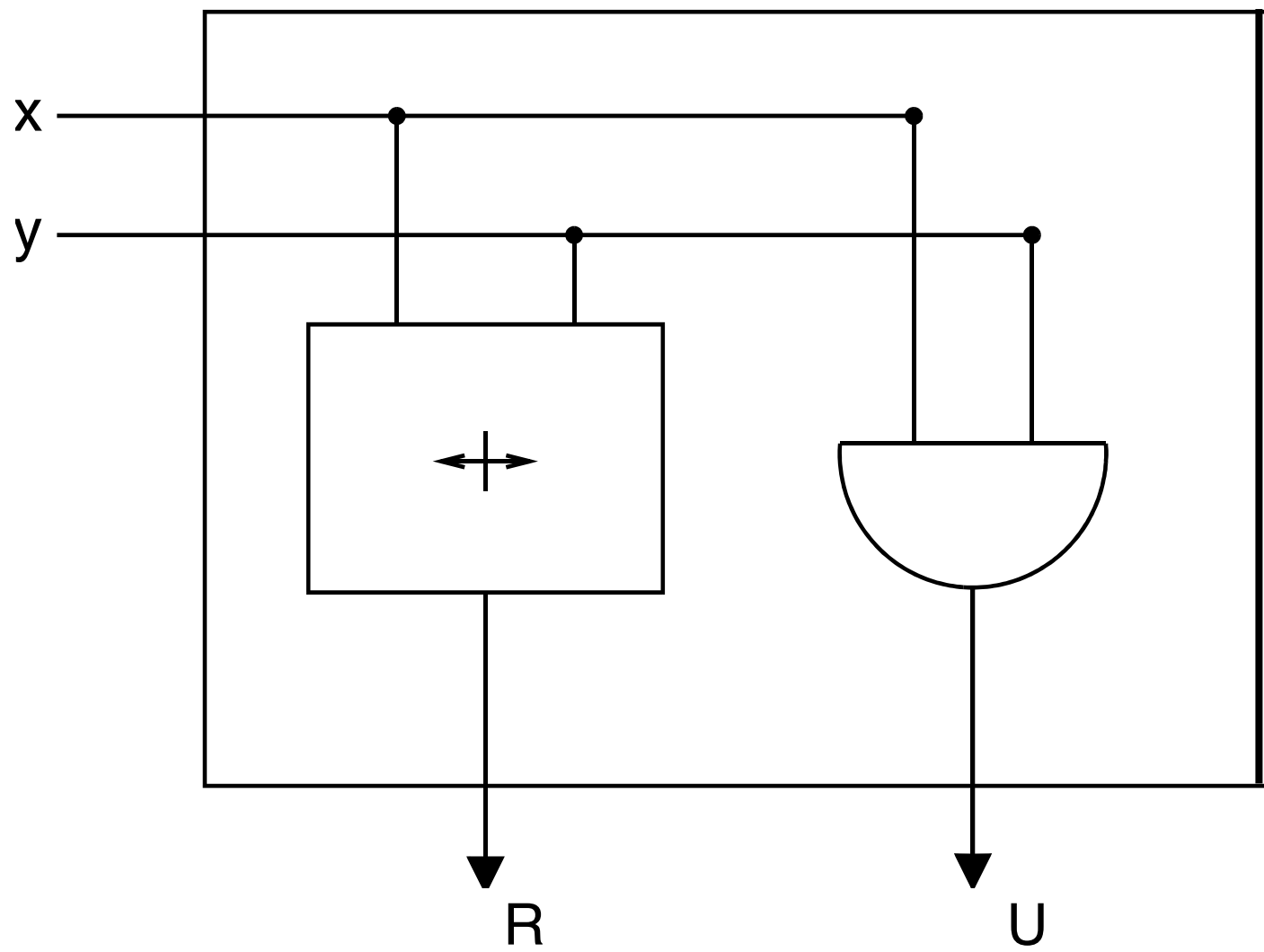
- 2 Dual-Ziffern und Übertrag

— Ausgabe

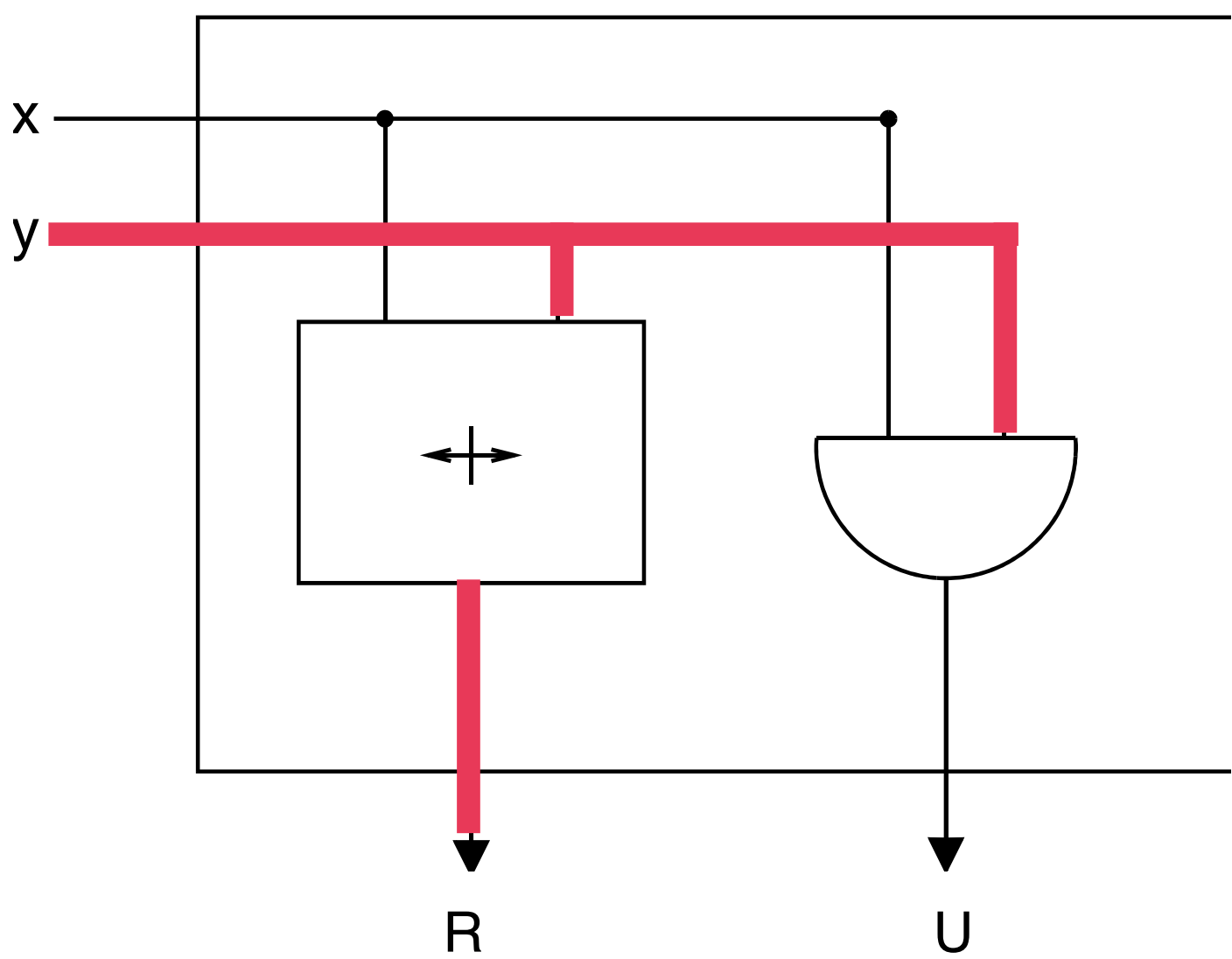
- Ergebnis $R = (x \oplus y) \oplus u$
- Übertrag $U = x \cdot y + x \cdot u + y \cdot u$
 $= x \cdot y + (x \oplus y) \cdot u$

u	x	y	R	U
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

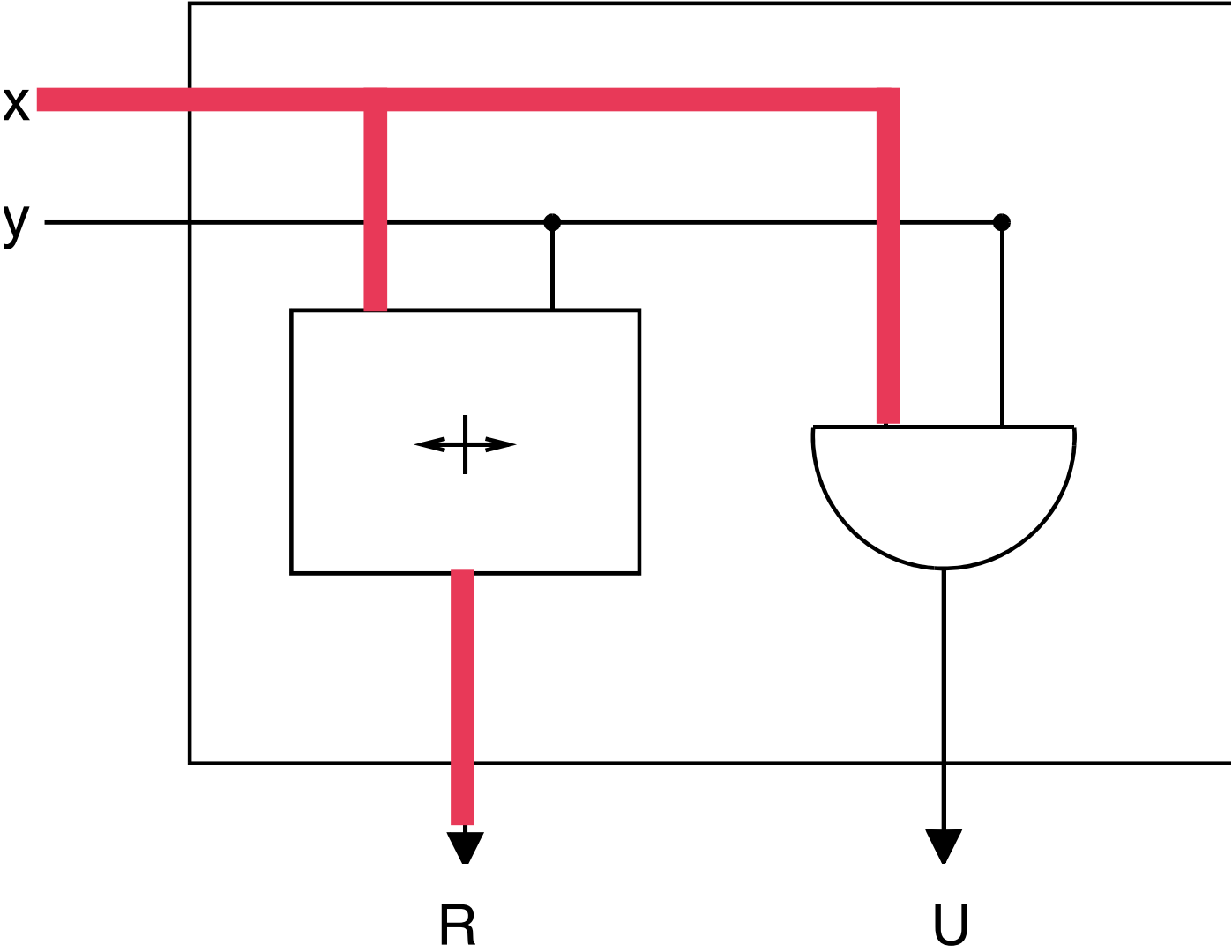
Halbaddierer



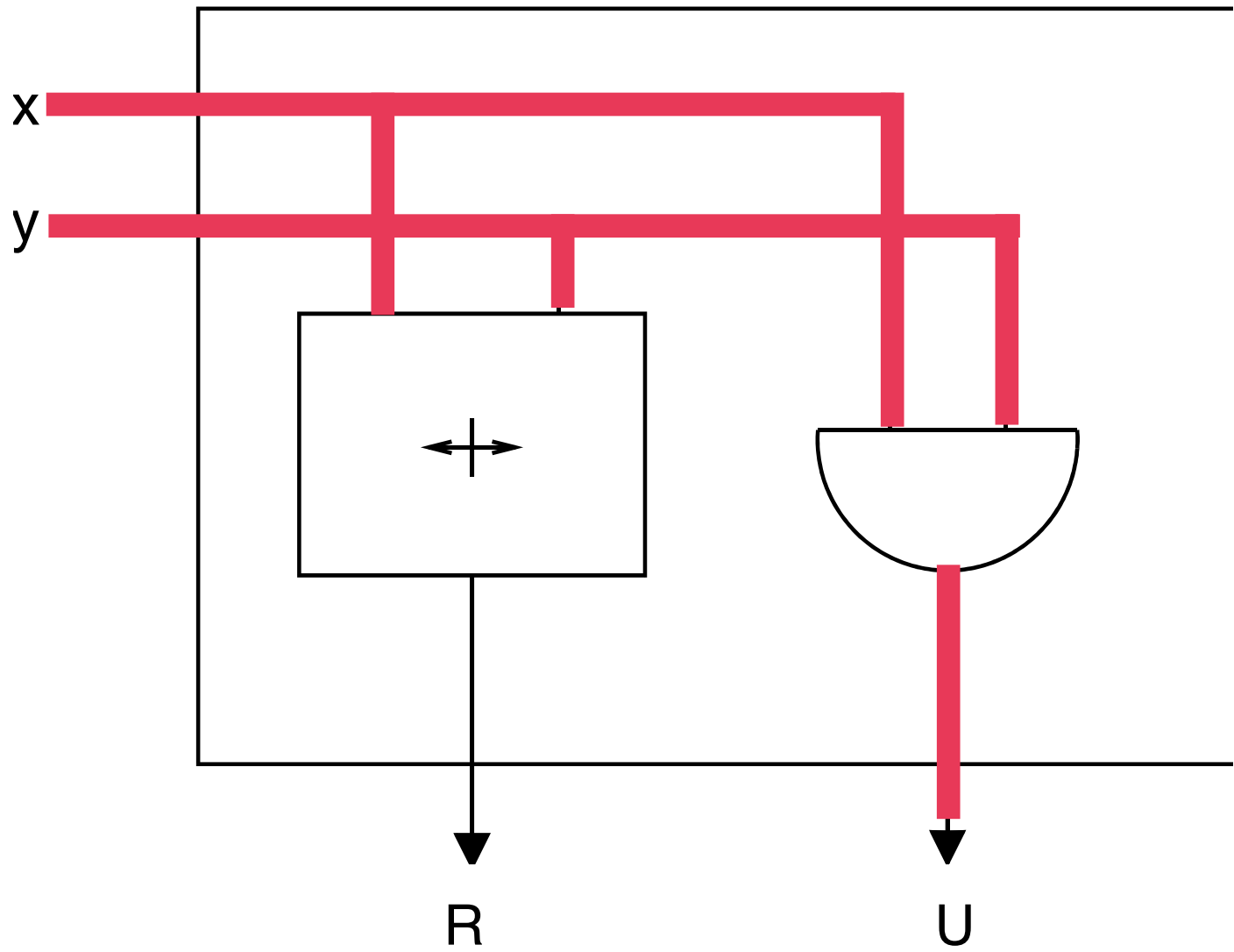
Halbaddierer



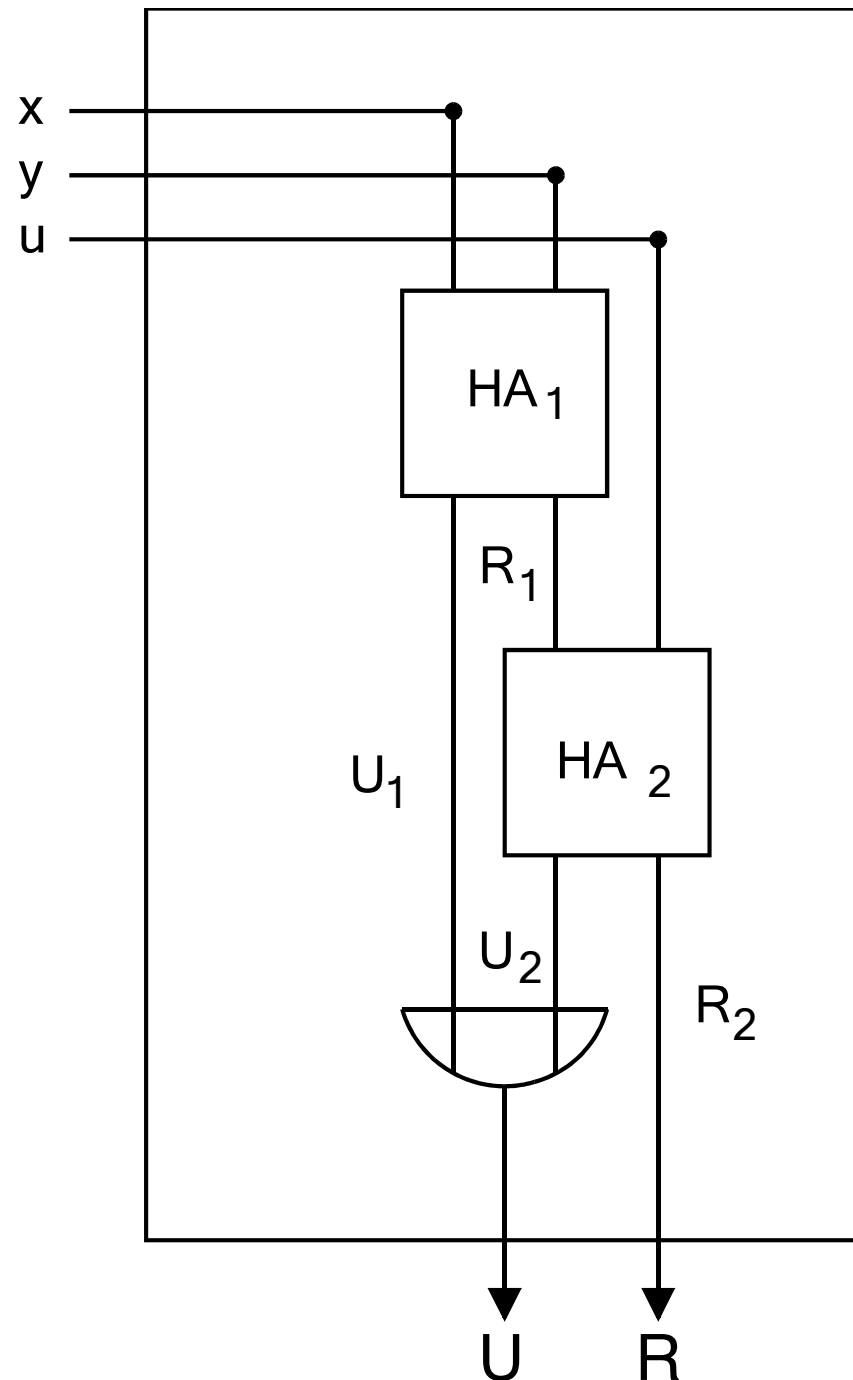
Halbaddierer



Halbaddierer



Volladdierer



$$> \begin{aligned} U_1 &= x \cdot y \\ R_1 &= x \oplus y = x \leftrightarrow y \end{aligned}$$

$$> \begin{aligned} U_2 &= (x \oplus y) \cdot u \\ R_2 &= (x \oplus y) \oplus u \end{aligned}$$

$$> \begin{aligned} U &= U_1 + U_2 \\ R &= R_2 \end{aligned}$$

Ripple Carry Adder

Binär-Addition

$$\begin{array}{r}
 10110111 \\
 +11000101 \\
 \hline
 101111100
 \end{array}$$

The diagram illustrates the ripple carry process. The first row is 10110111, the second row is +11000101, and the result is 101111100. A carry of 1 is shown on the left. The carry propagates through the columns, with the final carry being 1. The carry is shown as a vertical bar with a 1 inside, and the final carry is shown as a 1 on the left.

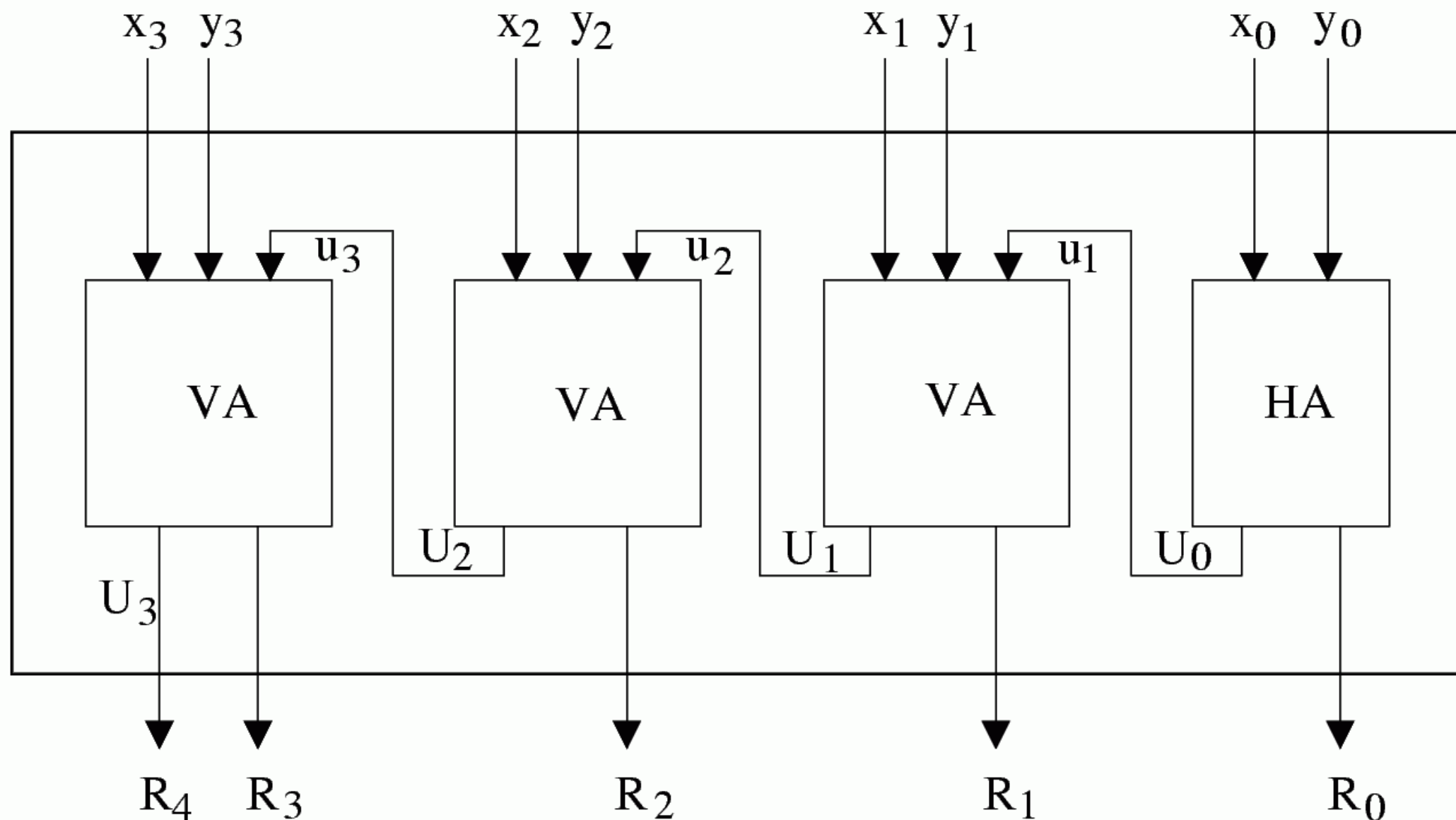
> Schriftliche Addition:

1. Letzte Stelle ausrechnen
2. Zweitletzte Stelle ausrechnen
3. ...

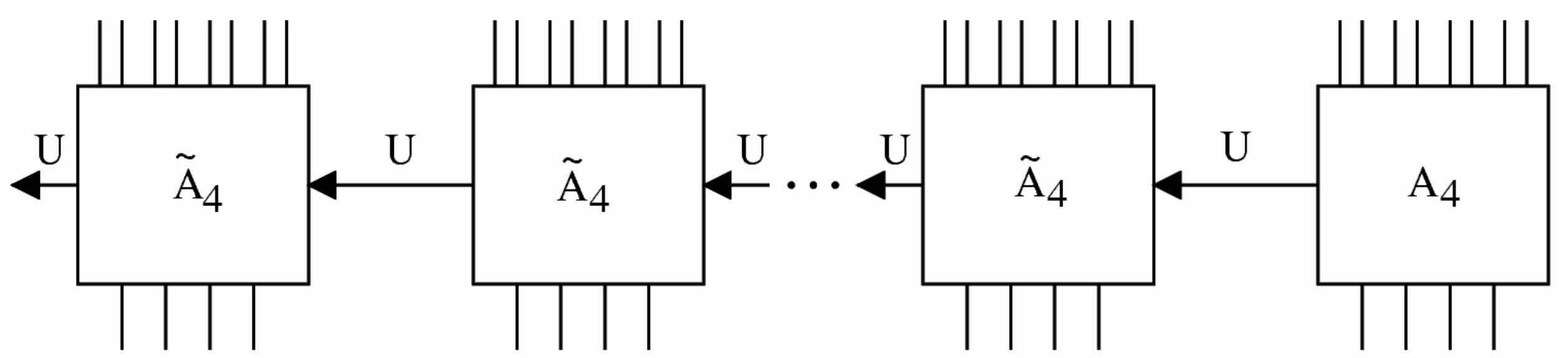
> Der **Übertrag rieselt** durch die Berechnung.
 ➔ Einfach, aber langsam.

Addiernetz für zwei 4-stellige Dualzahlen

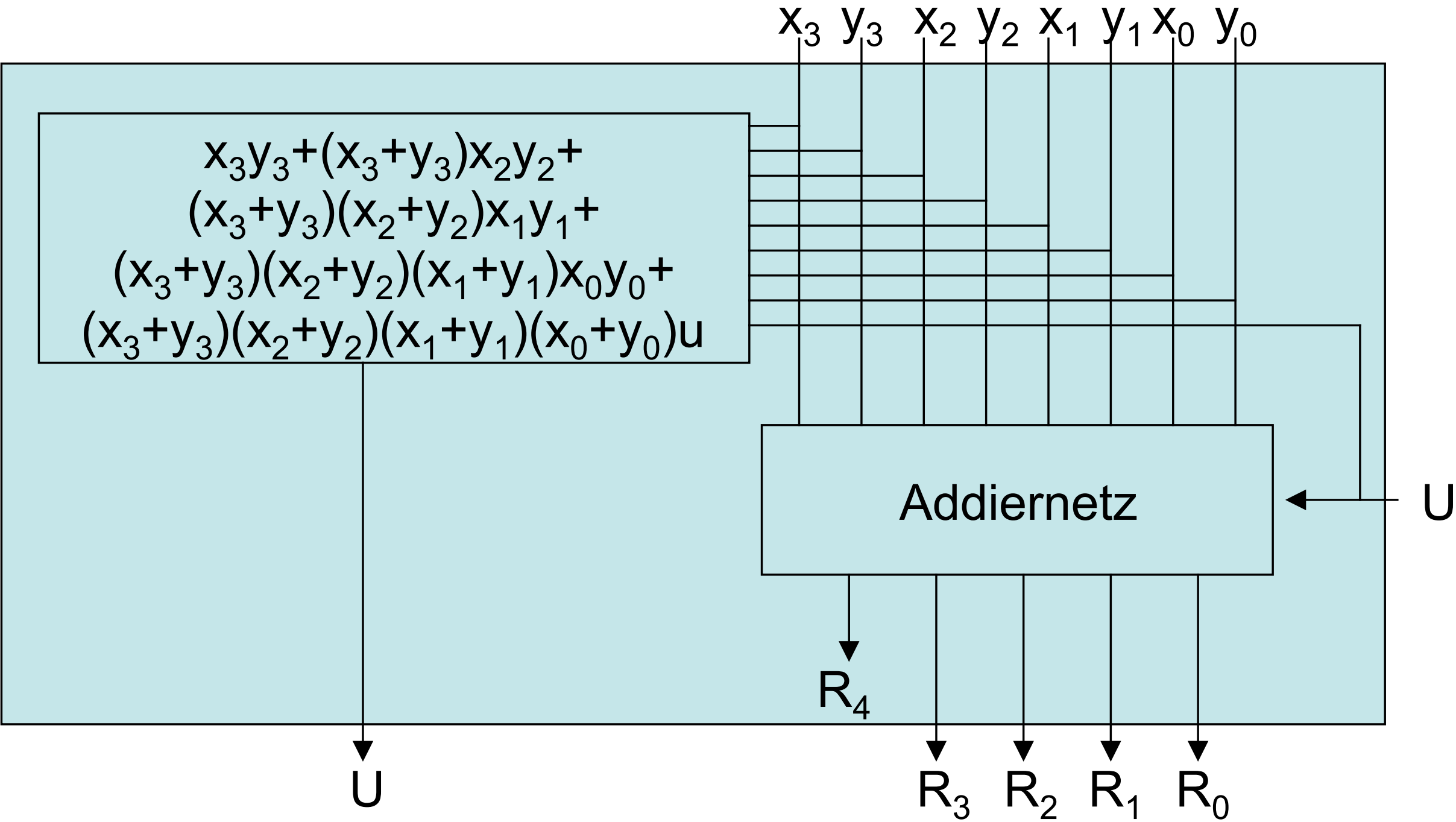
asynchrones (Parallel-) Addiernetz, Ripple-Carry-Adder:



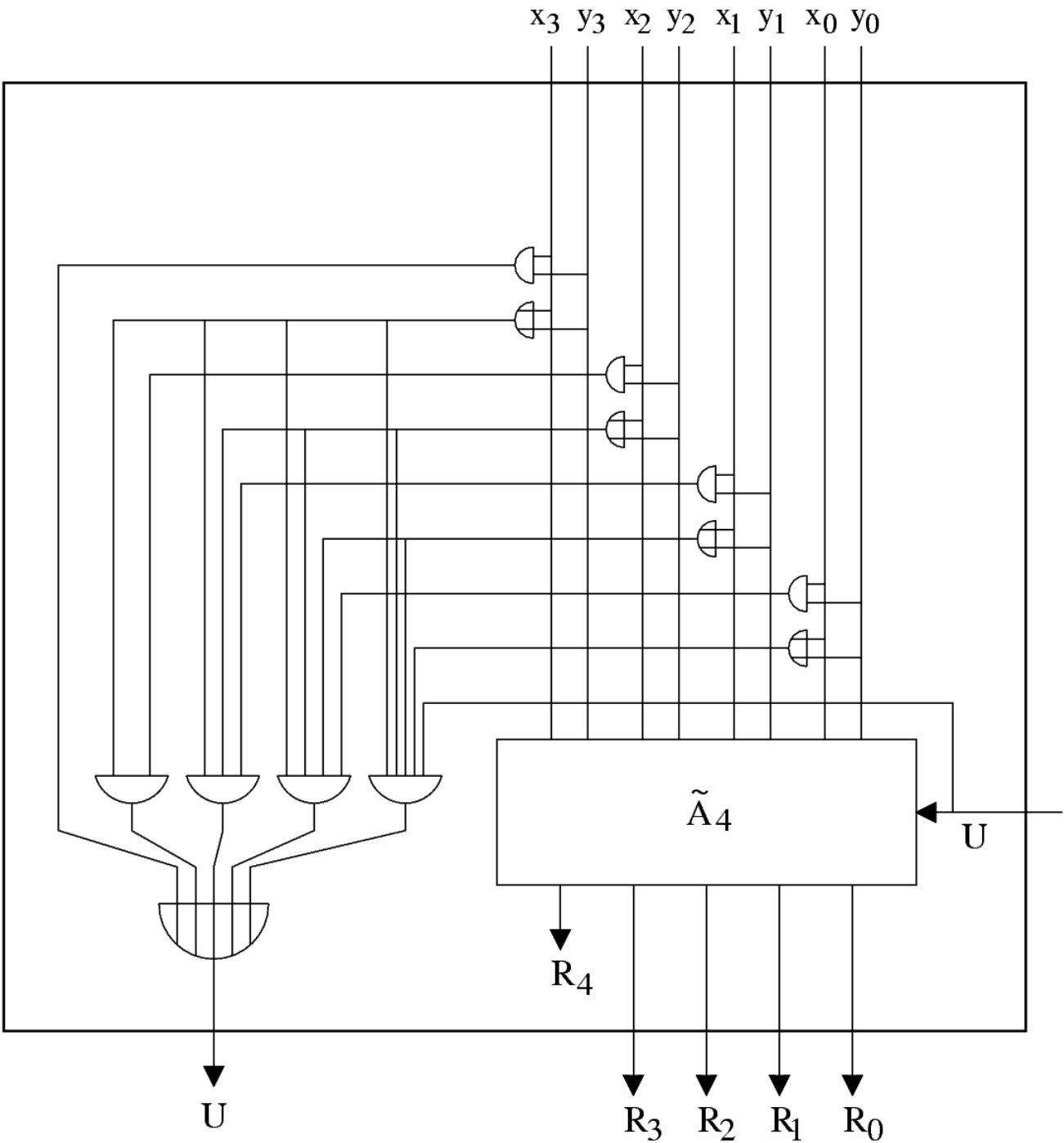
n-stelliges Addiernetz



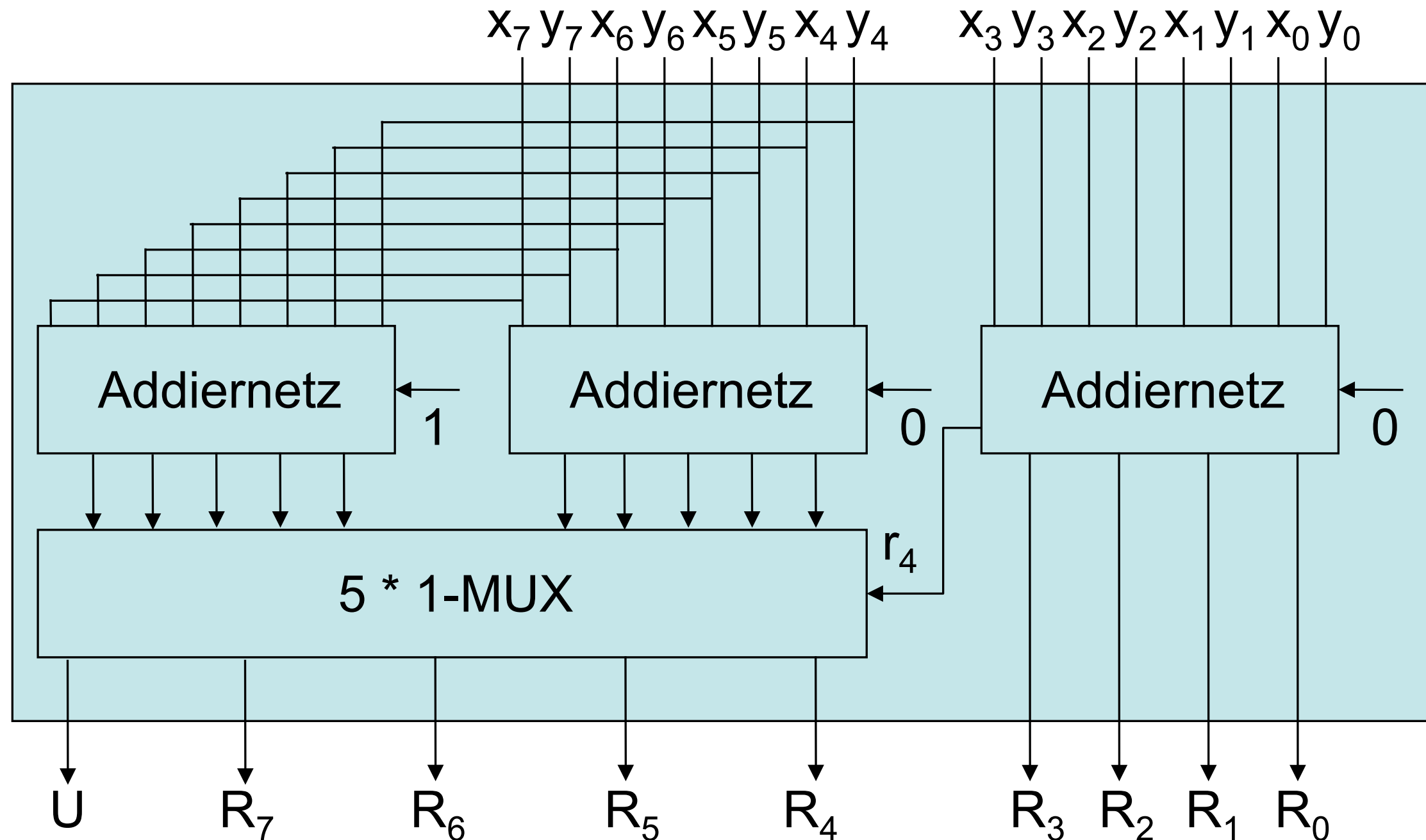
Carry-Bypass Addiernetz



Carry-Bypass-Addiernetz



Carry-Select Addiernetz

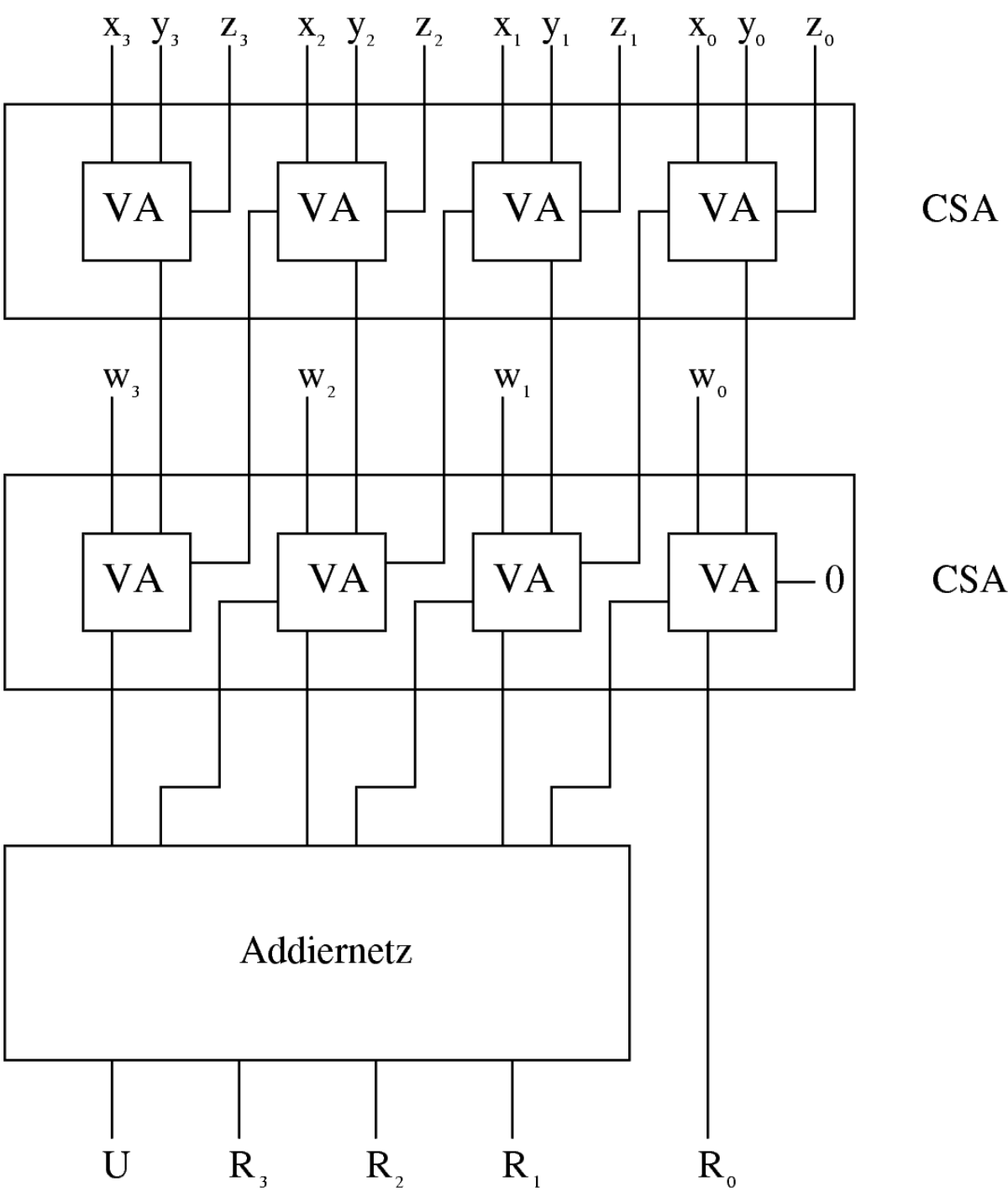


Carry-Save Addiernetz (CSA)

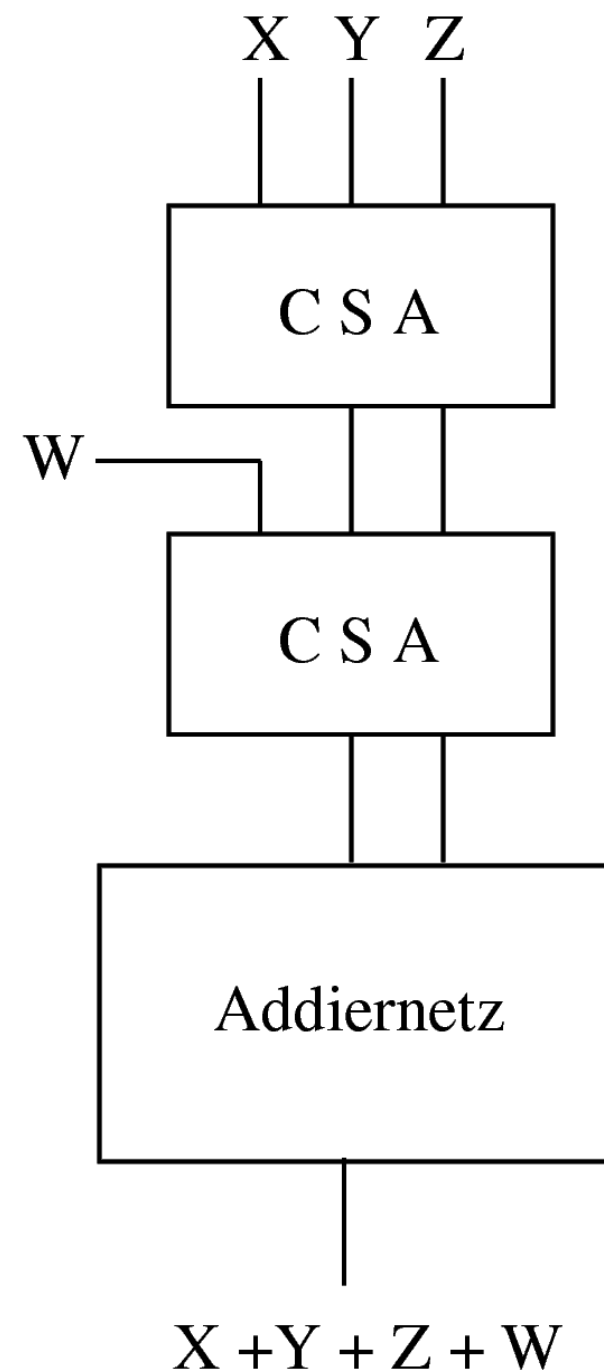
- > Addition von mehreren Summanden (im Beispiel: 4)
- > Carry-Save Addiernetz zur Addition von 3 Summanden
 - 1. Stufe: Addition von 3 Summanden
 - Summe1, Übertrag1
 - 2. Stufe: Addition des 4. Summanden, Summe1, Übertrag1
 - Summe2, Übertrag2
 - 3. Stufe: Addition von Summe2 und Übertrag2

X	010 1	Summe1	0010		
Y	001 1	Übertrag1	1010	Summe2	1001
Z	0100	W	0001	Übertrag2	0100
<hr/>					
Summe1	001 0	Summe2	1001	Summe	1101
Übertrag1	10 1 0	Übertrag2	0100		

Carry-Save-Addiernetz

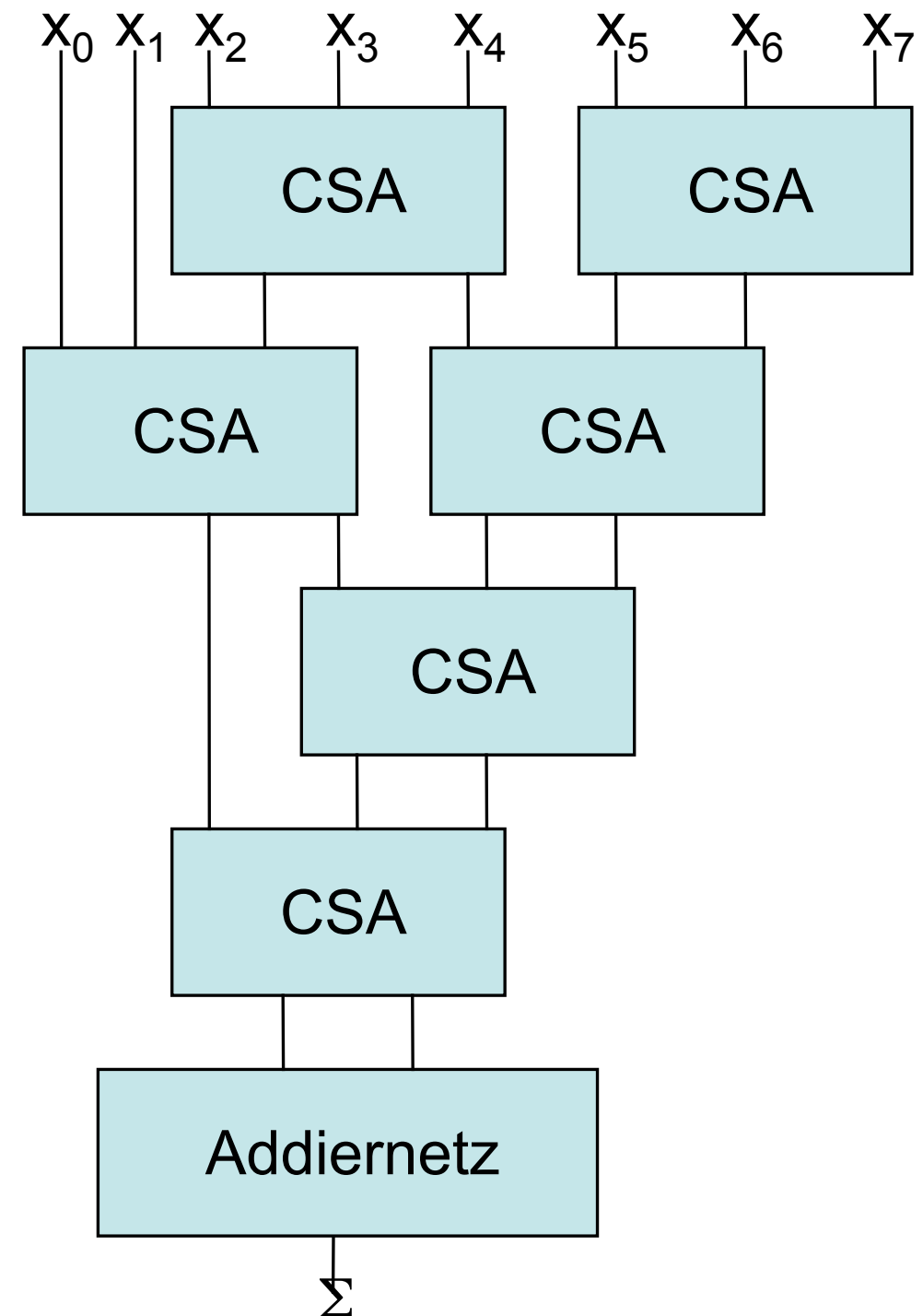


Prinzip der Carry-Save-Addition (CSA)



Wallace Tree

- > Allgemein:
- $m-2$ CSA zur Addition von m Summanden
 - Beispiel: $m=8$
 - parallele Berechnungen
 - Tiefe: $O(\log(m))$
 - Logarithmus zur Basis $3/2$



Ziele

Sie können sowohl mit dem top-down als auch dem bottom-up Ansatz Boolesche Schaltungen entwerfen.

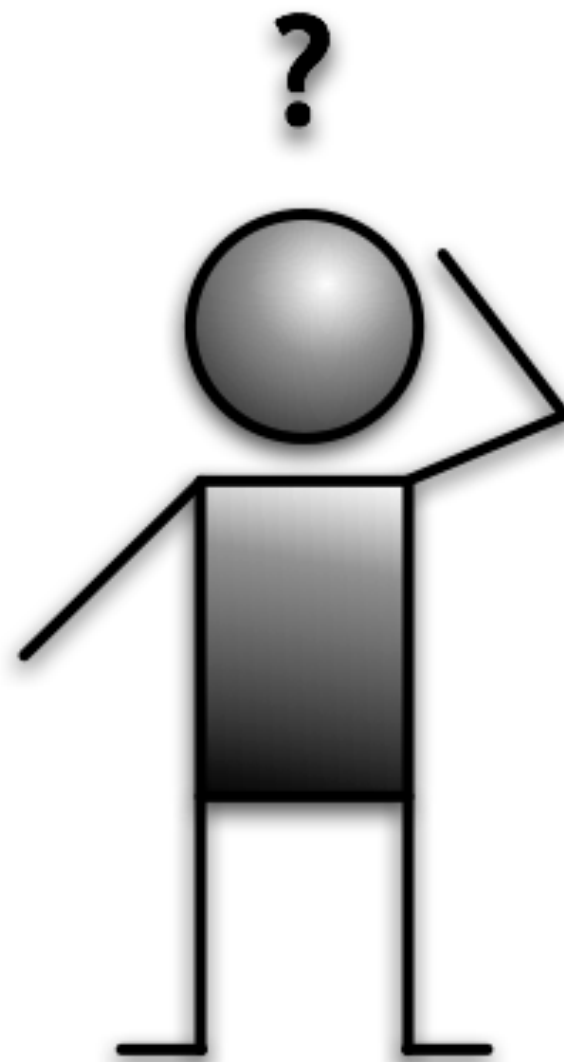
Sie kennen den Aufbau, Funktionsweise und Anwendungen (inkl. Decoder, Encoder) eines (De-)Multiplexers. Insbesondere können Sie beliebige Boolesche Funktionen mit einem Multiplexer realisieren.

Sie können Aufbau und Funktionsweise eines (Halb-)Addierers erklären.

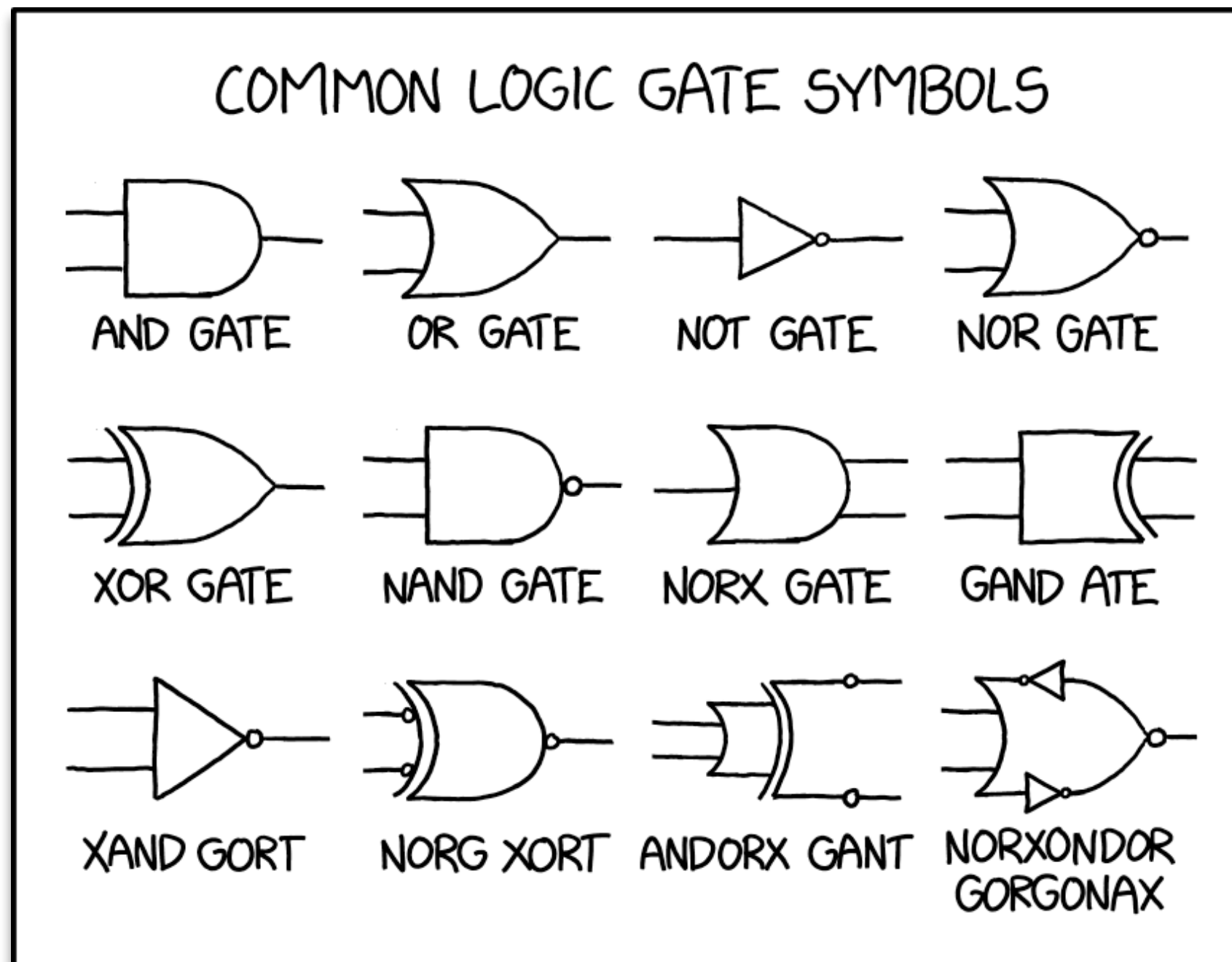
Sie können Aufbau und Funktionsweise verschiedener Addiernetze (ripple carry, carry bypass, carry select) erklären und diese miteinander vergleichen.

Sie können Aufbau und Funktionsweise eines carry save adders erklären (inklusive Wallace Tree).

Fragen?



Zum Schluss



<https://xkcd.com/2497>