Digitaltechnik

Wintersemester 2024/2025

Vorlesung 5

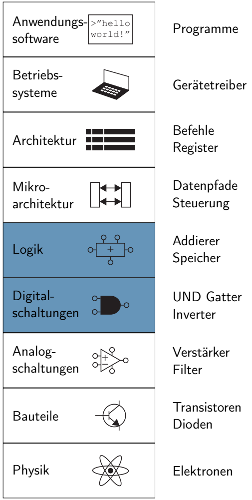
TECHNISCHE UNIVERSITÄT DARMSTADT

ENCRYPTO
CRYPTOGRAPHY AND PRIVACY ENGINEERING

Inhalt

1. Boole'sche Algebra
2. Bubble Pushing
3. Logik-Realisierung mit Basis-Gattern

Harris 2016
Kap. 2.3 - 2.5, 2.8



Abgabefrist für Hausaufgabe B zu Vorlesungen 03 und 04 nächste Woche Freitag 23:59!

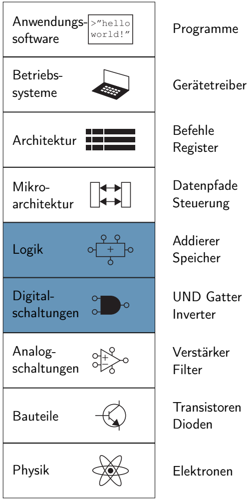
Wöchentliches Moodle-Quiz nicht vergessen!

Agenda

1 Boole'sche Algebra

2 Bubble Pushing

3 Logik-Realisierung mit Basis-Gattern



Boole'sche Algebra

LQ3-1

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

- Rechenregeln boole'scher Gleichungen
  - Axiome: grundlegende Annahmen der Algebra (nicht beweisbar)
  - Theoreme: komplexere Regeln, die sich aus Axiomen ergeben (beweisbar)
- analog zur Algebra auf natürlichen Zahlen
- ergänzt um Optimierungen durch Begrenzung auf $\mathbb{B}$
- Axiome und Theoreme haben jeweils duale Entsprechung: AND $\leftrightarrow$ OR, $0 \leftrightarrow 1$

Axiome der boole'schen Algebra
(siehe auch „Hilfsblatt Klausur“)

(Dualität: AND $\leftrightarrow$ OR, 0 $\leftrightarrow$ 1)

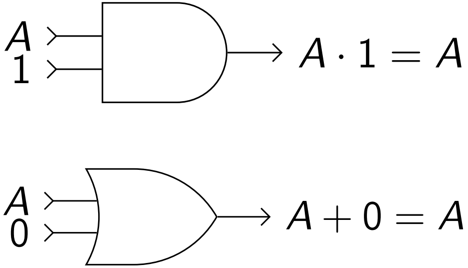
	Axiom		**Duales Axiom**	**Bedeutung**
A1	$B \neq 1 \Rightarrow B = 0$	A1'	$B \neq 0 \Rightarrow B = 1$	Dualität
A2	$\overline{0} = 1$	A2'	$\overline{1} = 0$	Negieren
A3	$0 \cdot 0 = 0$	A3'	$1 + 1 = 1$	Und / Oder
A4	$1 \cdot 1 = 1$	A4'	$0 + 0 = 0$	Und / Oder
A5	$0 \cdot 1 = 1 \cdot 0 = 0$	A5'	$1 + 0 = 0 + 1 = 1$	Und / Oder

Theoreme der boole'schen Algebra
(siehe auch „Hilfsblatt Klausur")

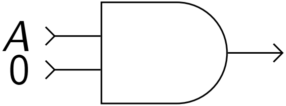
ENCRYPTO CRYPTOGRAPHY AND PRIVACY ENGINEERING

	Theorem		Duales Theorem	Bedeutung
T1	$A \cdot 1 = A$	T1'	$A + 0 = A$	Neutralität
T2	$A \cdot 0 = 0$	T2'	$A + 1 = 1$	Extremum
T3	$A \cdot A = A$	T3'	$A + A = A$	Idempotenz
T4	$\overline{\overline{A}} = A$			Involution
T5	$A \cdot \overline{A} = 0$	T5'	$A + \overline{A} = 1$	Komplement
T6	$A \cdot B = B \cdot A$	T6'	$A + B = B + A$	Kommutativität
T7	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	T7'	$A + (B + C) = (A + B) + C$	Assoziativität
T8	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	T8'	$A + (B \cdot C) = (A + B) \cdot (A + C)$	Distributivität
T9	$A \cdot (A + B) = A$	T9'	$A + (A \cdot B) = A$	Absorption
T10	$(A \cdot B) + (A \cdot \overline{B}) = A$	T10'	$(A + B) \cdot (A + \overline{B}) = A$	Zusammenfassen
T11	$(A \cdot B) + (\overline{A} \cdot C) + (B \cdot C) = (A \cdot B) + (\overline{A} \cdot C)$	T11'	$(A + B) \cdot (\overline{A} + C) \cdot (B + C) = (A + B) \cdot (\overline{A} + C)$	Konsensus
T12	$\overline{A \cdot B \cdot C \ldots} = \overline{A} + \overline{B} + \overline{C} \ldots$	T12'	$\overline{A + B + C \ldots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \ldots$	De Morgan

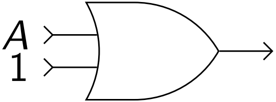
T1: Neutralität von 1 und 0



T2: Extremum von 0 und 1

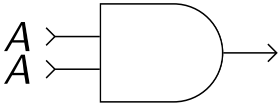


$A \cdot 0 = 0$

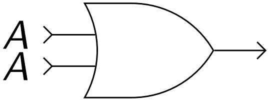


$A + 1 = 1$

T3: Idempotenz

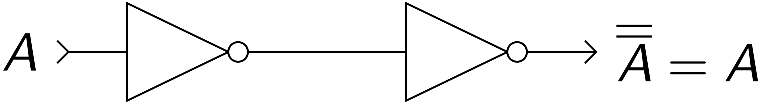


$$A \cdot A = A$$

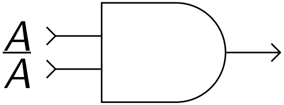


$$A + A = A$$

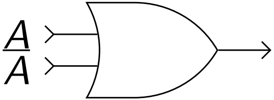
T4: Involution



T5: Komplement

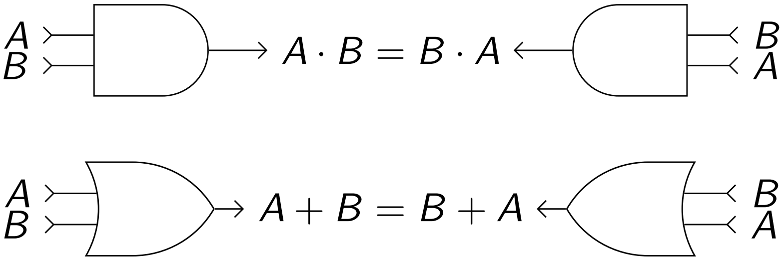


$$A \cdot \overline{A} = 0$$

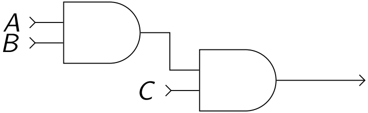


$$A + \overline{A} = 1$$

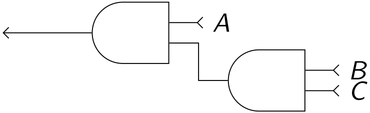
T6: Kommutativität

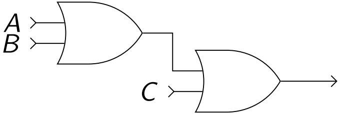


T7: Assoziativität

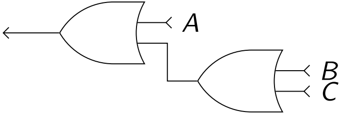


$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

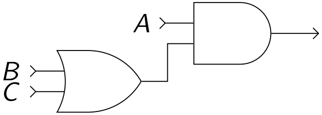




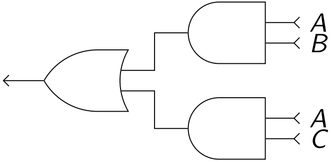
$$(A + B) + C = A + (B + C)$$

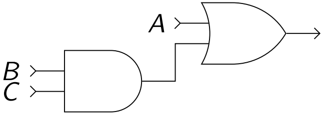


T8: Distributivität

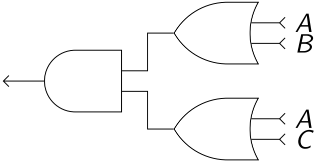


$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

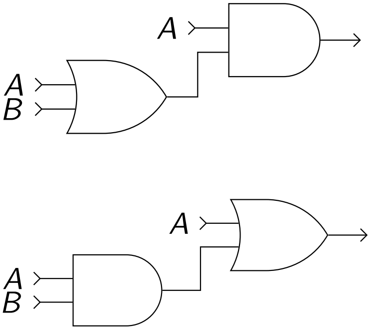




$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

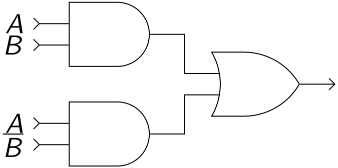


T9: Absorption

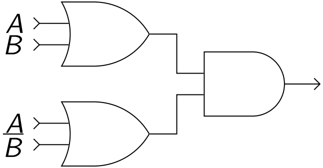


$$A \cdot (A + B) = A$$

$$A + (A \cdot B) = A$$

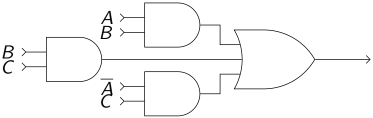


$$(A \cdot B) + (A \cdot \overline{B}) = A$$

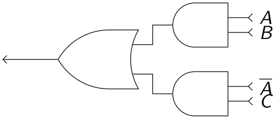


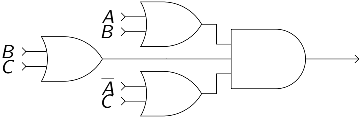
$$(A + B) \cdot (A + \overline{B}) = A$$

T11: Konsensus

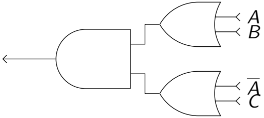


$$(A \cdot B) + (\overline{A} \cdot C) + (B \cdot C) = (A \cdot B) + (\overline{A} \cdot C)$$

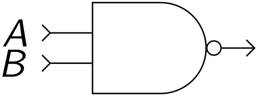




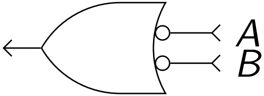
$$(A + B) \cdot (\overline{A} + C) \cdot (B + C) = (A + B) \cdot (\overline{A} + C)$$

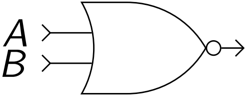


T12: De Morgan

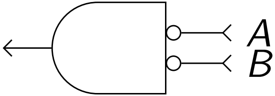


$$\overline{A \cdot B} = \overline{A} + \overline{B}$$





$$\overline{A + B} = \overline{A} \cdot \overline{B}$$



Augustus De Morgan, 1806 - 1871

- erster Präsident der London Mathematical Society
- Lehrer von Ada Lovelace
- De Morgan'sche Regeln:
  - Das Komplement des Produkts ist die Summe der Komplemente.
  - Das Komplement der Summe ist das Produkt der Komplemente.

Beweis für Theoreme

- Methode 1: Überprüfen aller Möglichkeiten
- Methode 2: Gleichung durch Axiome und andere Theoreme vereinfachen

Beweis für Distributivität (T8)
Durch Überprüfen aller Möglichkeiten

A	B	C	$B+C$	$A\,(B+C)$	$A\,B$	$A\,C$	$A\,B+A\,C$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Beweis für Absorption (T9)
Durch Anwendung von Axiomen und Theoremen

$$
\begin{aligned}
& A \cdot (A + B) && \text{Distributivität} \\
= {} & A \cdot A + A \cdot B && \text{Idempotenz} \\
= {} & A + A \cdot B && \text{Neutralität} \\
= {} & A \cdot 1 + A \cdot B && \text{Distributivität} \\
= {} & A \cdot (1 + B) && \text{Extremum} \\
= {} & A \cdot 1 && \text{Neutralität} \\
= {} & A && \text{q.e.d.}
\end{aligned}
$$

Beweis für Zusammenfassen (T10)

Durch Anwendung von Axiomen und Theoremen

$$
\begin{aligned}
& A \cdot B + A\,\overline{B} && \text{Distributivität} \\
= {} & A \cdot (B + \overline{B}) && \text{Komplement} \\
= {} & A \cdot \quad 1 && \text{Neutralität} \\
= {} & A && \text{q.e.d.}
\end{aligned}
$$

Beweis für Konsensus (T11)

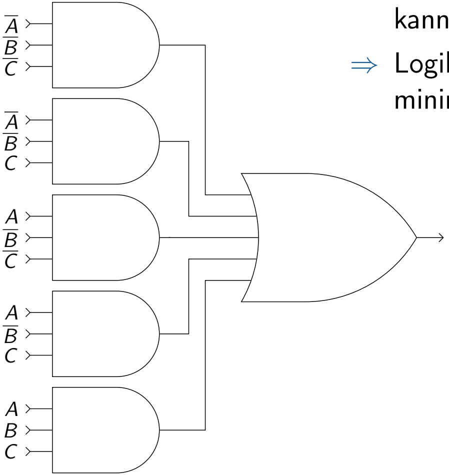
Durch Anwendung von Axiomen und Theoremen

ENCRYPTO CRYPTOGRAPHY AND PRIVACY ENGINEERING

$$
\begin{aligned}
& A \cdot B + \overline{A} \cdot C + B \cdot C && \text{Neutralität} \\
=\ & A \cdot B + \overline{A} \cdot C + 1 \cdot B \cdot C && \text{Komplement} \\
=\ & A \cdot B + \overline{A} \cdot C + (A + \overline{A}) \cdot B \cdot C && \text{Distributivität} \\
=\ & A \cdot B + \overline{A} \cdot C + A \cdot B \cdot C + \overline{A} \cdot B \cdot C && \text{Kommutativität} \\
=\ & A \cdot B + A \cdot B \cdot C + \overline{A} \cdot C + \overline{A} \cdot C \cdot B && \text{Neutralität} \\
=\ & A \cdot B \cdot 1 + A \cdot B \cdot C + \overline{A} \cdot C \cdot 1 + \overline{A} \cdot C \cdot B && \text{Distributivität} \\
=\ & A \cdot B \cdot (1 + C) + \overline{A} \cdot C \cdot (1 + B) && \text{Extremum} \\
=\ & A \cdot B \cdot 1 + \overline{A} \cdot C \cdot 1 && \text{Neutralität} \\
=\ & A \cdot B + \overline{A} \cdot C && \text{q.e.d.}
\end{aligned}
$$

- Gatter-Realisierung per DNF kann sehr aufwändig sein

$\Rightarrow$ Logik (Anzahl Literale und Operatoren) minimieren



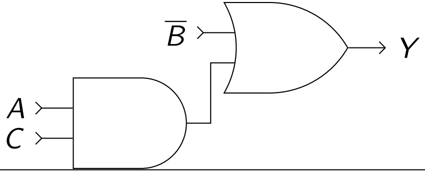
$$Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,C$$

Logikminimierung

RQ3-1

$$
\begin{aligned}
Y &= \overline{A}\;\overline{B}\;\overline{C} + \overline{A}\;\overline{B}\;C + A\;\overline{B}\;\overline{C} + A\;\overline{B}\;C + A\;B\;C \\
&= \overline{A}\;(\overline{B}\;\overline{C} + \overline{B}\;C) + A\;(\overline{B}\;\overline{C} + \overline{B}\;C) + A\;B\;C \\
&= \overline{A}\;(\overline{B}\;(\overline{C} + C)) + A\;(\overline{B}\;(\overline{C} + C)) + A\;B\;C \\
&= \overline{A}\;\overline{B} + A\;\overline{B} + A\;B\;C \\
&= (\overline{A} + A)\;\overline{B} + A\;B\;C \\
&= \overline{B} + A\;B\;C
\end{aligned}
$$

- weitere Vereinfachungen möglich?
- $Y = \overline{B} + A\;C$
- Systematik notwendig, um minimale Ausdücke zu erkennen/finden



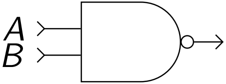
Agenda

1 Boole'sche Algebra

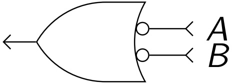
2 Bubble Pushing

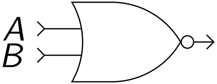
3 Logik-Realisierung mit Basis-Gattern

Anwendungssoftware >"hello world!"	Programme
Betriebssysteme	Gerätetreiber
Architektur	Befehle Register
Mikroarchitektur	Datenpfade Steuerung
Logik	Addierer Speicher
Digitalschaltungen	UND Gatter Inverter
Analogschaltungen	Verstärker Filter
Bauteile	Transistoren Dioden
Physik	Elektronen

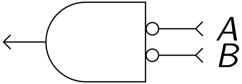


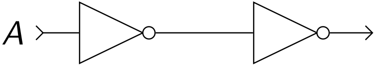
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$





$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

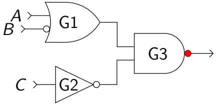




$$\overline{\overline{A}} = A$$

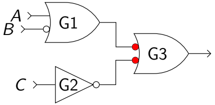
Invertierungsblasen verschieben
Bubble Pushing

- über Gatter (AND/OR/NOT/BUF) hinweg
  - vorwärts: Eingang $\rightarrow$ Ausgang
  - rückwärts: Ausgang $\rightarrow$ Eingang
  - Art des Gatters ändern: AND $\leftrightarrow$ OR
  - Blasen an *allen* Eingängen ändern: vorhanden $\leftrightarrow$ nicht vorhanden
  - Blase an Ausgang ändern: vorhanden $\leftrightarrow$ nicht vorhanden
- zwischen Gattern
  - vorwärts: Treiber $\rightarrow$ *alle* Empfänger
  - rückwärts: *alle* Empfänger $\rightarrow$ Treiber
  - doppelte Blasen heben sich gegenseitig auf (Involution)
- verbleibende Buffer (vorher Inverter) können entfernt werden



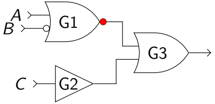
$Y = \overline{(A+\overline{B})\ \overline{C}}$

- De Morgan über G3
  - Blase am Ausgang $\rightarrow$ Blase an beiden Eingängen
  - AND $\rightarrow$ OR



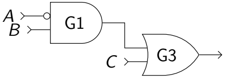
$Y = \overline{A+\overline{B}} + \overline{\overline{C}}$

- Blasen entlang Leitungen verschieben
  - G3 $\rightarrow$ G1
  - G3 $\rightarrow$ G2 (Doppelblase aufheben)



$Y = \overline{A+\overline{B}} + C$

- De Morgan über G1
  - Blasen an Ein- und Ausgängen *invertieren*
  - OR $\rightarrow$ AND
- Buffer G2 entfernen



$Y = \overline{A}\ B + C$

- zwei Inverter weniger

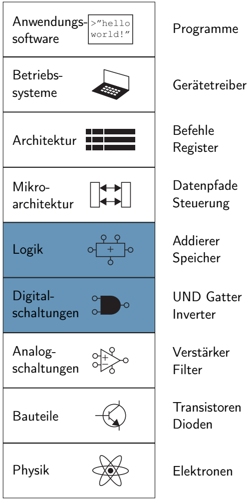
Anwendungen

- Schaltungen vereinfachen
  - weniger Inverter
  - weniger Literale (z.B. nur A statt $A, \overline{A}$)
  - weniger verschiedene Gatter-Arten
    $\rightarrow$ einfachere Zellbibliothek (z.B. nur AND, kein OR)
- Komplementäre Schaltungen für CMOS-Schaltung ableiten
  - Y für Pull-Up Netzwerk $\leftrightarrow$ $\overline{Y}$ für Pull-Down Netzwerk
  - $Y = \overline{A}B + C$
  - $\overline{Y} = \overline{\overline{A}B + C} = \overline{\overline{A}B}\ \overline{C} = (A + \overline{B})\overline{C}$

1 Boole'sche Algebra

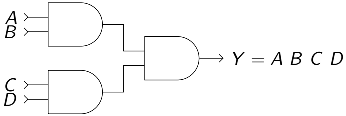
2 Bubble Pushing

3 Logik-Realisierung mit Basis-Gattern



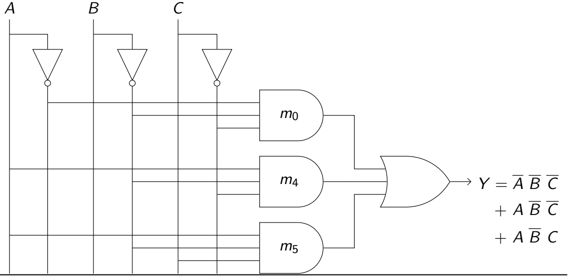
Zweistufige Logik

- direkte (konstruktive) Umsetzung der disjunktiven Normalform (DNF)
  - Eingangsliterale: ein Inverter pro Variable (falls benötigt)
  - Minterme: je ein „breites“ AND Gatter an passende Literale anschließen
  - Summe: alle Minterme an ein „breites“ OR Gatter anschließen



- Gatter mit vielen Inputs als Bäume kleinerer Gatter

$\Rightarrow$ jede boole'sche Funktion realisierbar mit Basisgattern
  - AND2
  - OR2
  - NOT



Konventionen für lesbare Schaltpläne

- Eingänge links (oder oben)
- Ausgänge rechts (oder unten)
- Gatter von links nach rechts (oben nach unten) angeordnet
- gerade (oder rechtwinklige) Verbindungen

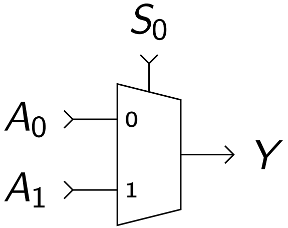
$\Rightarrow$ keine Schrägen oder Kurven

- 3-armige Kreuzungen gelten implizit als verbunden
- 4-armige Kreuzungen gelten nur bei Markierung (Punkt) als verbunden

Weitere kombinatorische Grundelemente

- zweistufige Logik
  - sehr mächtig
  - aufwändige Darstellung und Realisierung
  - realisiertes Verhalten nicht intuitiv ersichtlich
- weitere Basisgatter neben AND, OR, NOT:
  - XOR: Parität
  - Multiplexer (MUX): n zu 1
  - Dekodierer (DEC): n zu 2^n

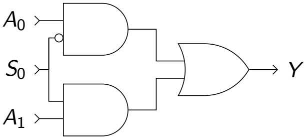
- Selektiert einen der n Dateneingänge $A_0, \ldots, A_{n-1}$ als Ausgang Y
- $k = \lceil \log_2 n \rceil$ Steuersignale $S_0, \ldots, S_{k-1}$
- $Y = A_{u_{2,k}(S_{k-1} \ldots S_0)}$




```
assign y = S0 ? A1 : A0;
```

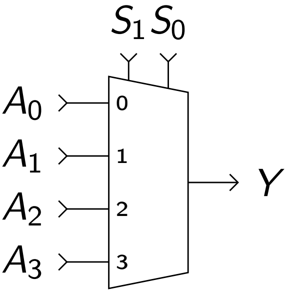
S_0	A_0	A_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

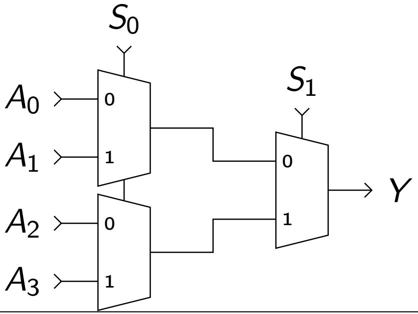
S_0	Y
0	A_0
1	A_1



$\text{MUX4} : \mathbb{B}^6 \rightarrow \mathbb{B}$

S_1	S_0	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

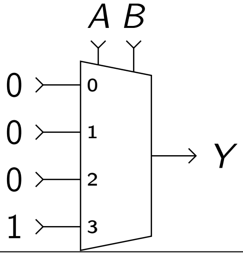


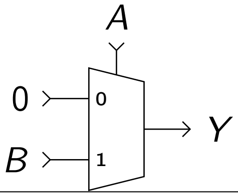


Logikrealisierung mit Multiplexern

- Variablen als Steuersignale verwenden
- Wahrheitswertetabelle als Konstanten an Dateneingängen
- entspricht adressiertem Speicherzugriff
  - Look-up Tabelle
  - ROM oder RAM $\rightarrow$ rekonfigurierbare Logik
- Beliebige Funktion mit N Variablen kann sogar via MUX2^{N-1} realisiert werden (s. Harris, Fig. 2.60)

A	B	$Y = A\ B$
0	0	0
0	1	0
1	0	0
1	1	1

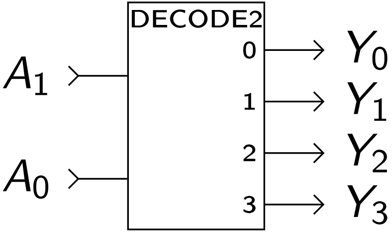




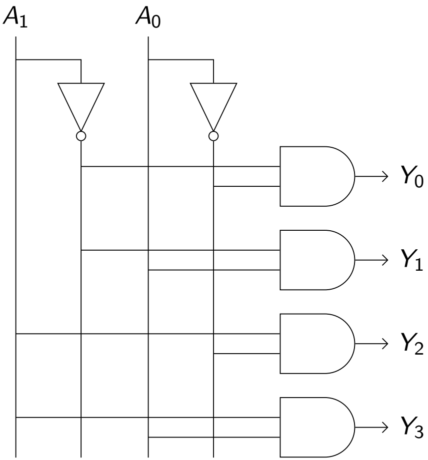
Dekodierer DECODE$n : \mathbb{B}^n \to \mathbb{B}^{2^n}$

- n Eingänge $A_0, \ldots, A_{n-1}$
- 2^n Ausgänge $Y_0, \ldots, Y_{2^n-1}$
- „One-Hot“ Kodierung: $Y_i = \mathrm{u}_{2,n}(A_{n-1} \ldots A_0) == i \; ? \; 1 : 0$

A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Implementierung von Dekodierern

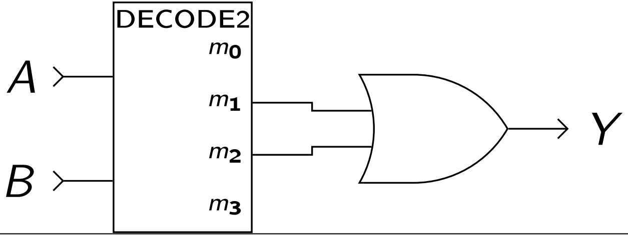


Logikrealisierung mit Decodern

- Summe über Minterme, auf denen Zielfunktion wahr ist

$\Rightarrow$ Decoder ersetzt erste Stufe der zweistufigen Logikrealisierung

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



Abgabegruppen-Börse im Anschluss

Einzelabgaben bei den Hausaufgaben werden **nicht** bewertet!

Zusammenfassung und Ausblick

1. Boole'sche Algebra
2. Bubble Pushing
3. Logik-Realisierung mit Basis-Gattern

nächste Vorlesung beinhaltet

- Logikminimierung
- Mehrwertige Logik
- Zeitverhalten in kombinatorischen Schaltungen

Hausaufgabe B zu Vorlesungen 03 und 04 muss bis nächste Woche Freitag 23:59 abgegeben werden. Wöchentliches Moodle-Quiz nicht vergessen!

Anwendungs-software >"hello world!"	Programme
Betriebs-systeme	Gerätetreiber
Architektur	Befehle Register
Mikro-architektur	Datenpfade Steuerung
Logik	Addierer Speicher
Digital-schaltungen	UND Gatter Inverter
Analog-schaltungen	Verstärker Filter
Bauteile	Transistoren Dioden
Physik	Elektronen