technische universität
dortmund

fakultät für informatik

Rechnerstrukturen, Teil 1

Vorlesung 4 SWS WS 16/17

Prof. Dr. Jian-Jia Chen
Fakultät für Informatik – Technische Universität Dortmund
jian-jia.chen@cs.uni-dortmund.de
http://ls12-www.cs.tu-dortmund.de

Übersicht

1. Organisatorisches ✓
2. Einleitung ✓
3. Repräsentation von Daten ✓
4. Boolesche Funktionen und Schaltnetze ✓
5. Rechnerarithmetik ✓
6. Optimierung von Schaltnetzen ✓
7. **Programmierbare Bausteine**
8. Synchrone Schaltwerke

7. Programmierbare Bausteine

7.1 Einleitung

Realisierung von Schaltnetzen

Gedanken zur Anwendung

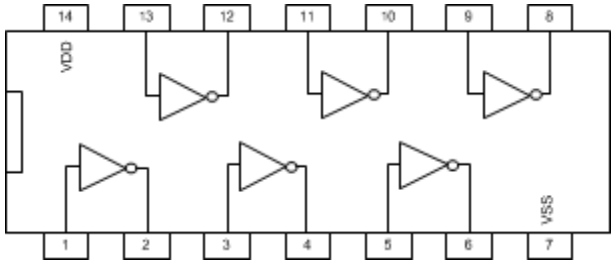
1. Problem
2. boolesche Funktion
3. Schaltnetz-Entwurf
4. Schaltnetz-Realisierung

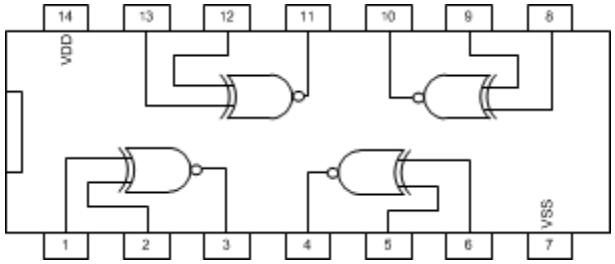
Realisierungen

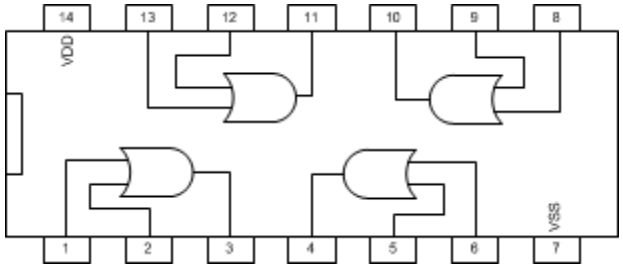
- hoch-integrierte Schaltung
  teuer Lohnt sich nur bei großen Stückzahlen.
- direkte Umsetzung mit Gattern
  umständlich

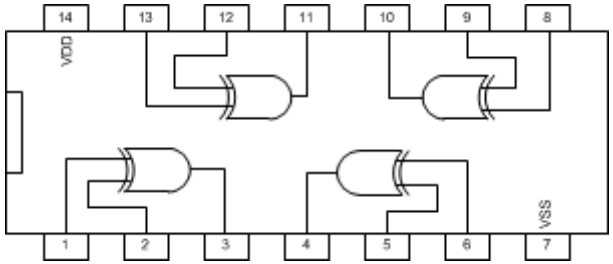
7.1 Einleitung

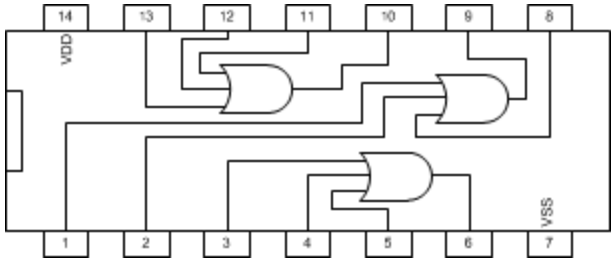
Realisierung mit Gattern

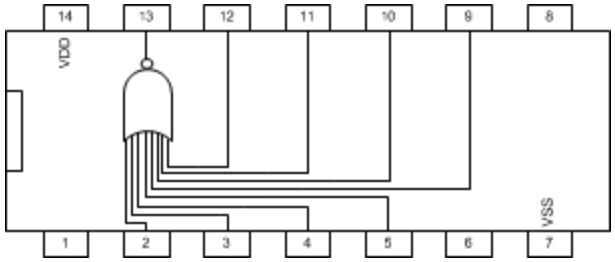






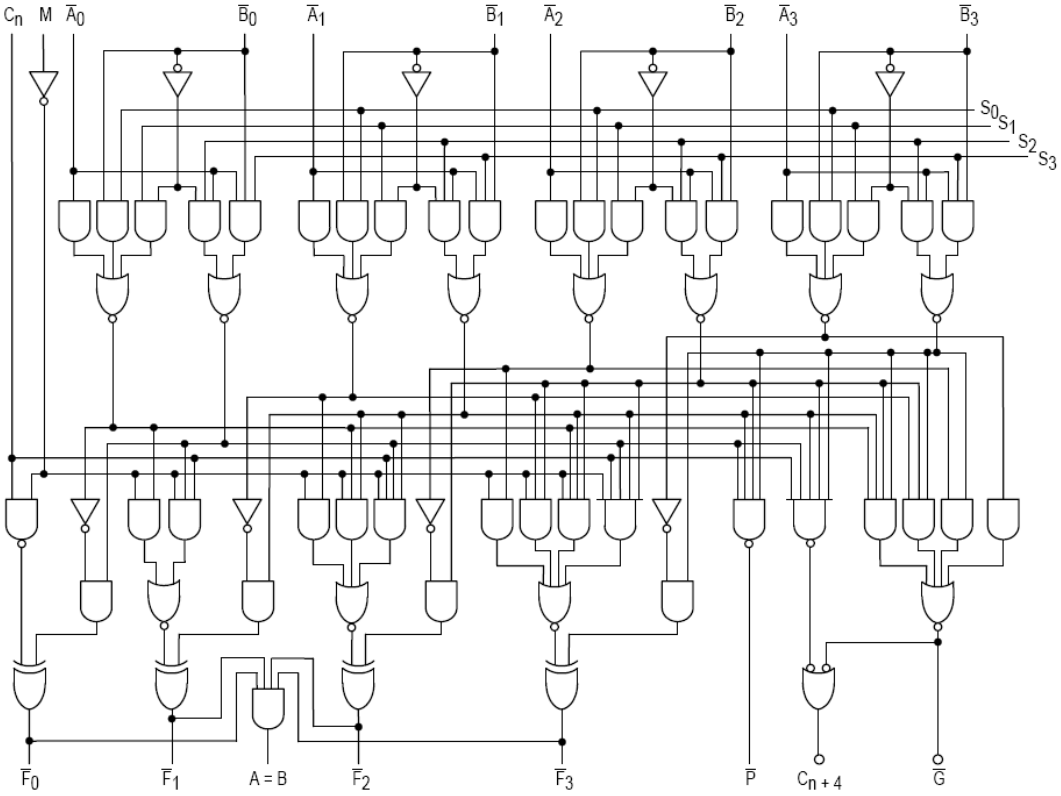


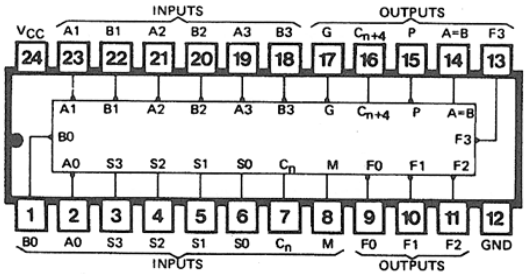




7.1 Einleitung

Realisierung mit Gattern





Quelle: en.wikibooks.org/wiki/Microprocessor_Design/ALU, Autor: Poil

$S3$	$S2$	$S1$	$S0$	$M=1$	$M=0$ $CIN=1$	$M=0$ $CIN=0$
0	0	0	0	$\overline{A}$	A	$A+1$
0	0	0	1	$\overline{A\|B}$	$A\|B$	$A\|B+1$
0	0	1	0	$\overline{A}B$	$A\|\overline{B}$	$A\|\overline{B}+1$
0	0	1	1	0	-1	0
0	1	0	0	$\overline{AB}$	$A+A\overline{B}$	$A+A\overline{B}+1$
0	1	0	1	$\overline{B}$	$(A\|B)+A\overline{B}$	$(A\|B)+A\overline{B}+1$
0	1	1	0	$A\oplus B$	$A-B-1$	$A-B$
0	1	1	1	$A\overline{B}$	$A\overline{B}-1$	$A\overline{B}$
1	0	0	0	$\overline{A}\|B$	$A+AB$	$A+AB+1$
1	0	0	1	$\overline{A\oplus B}$	$A+B$	$A+B+1$
1	0	1	0	B	$(A\|\overline{B})+AB$	$(A\|\overline{B})+AB+1$
1	0	1	1	AB	$AB-1$	AB
1	1	0	0	1	$2*A$	$2*A+1$
1	1	0	1	$A\|\overline{B}$	$(A\|B)+A$	$(A\|B)+A+1$
1	1	1	0	$A\|B$	$(A\|\overline{B})+A$	$(A\|\overline{B})+A+1$
1	1	1	1	A	$A-1$	A

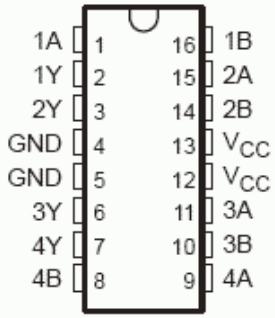
Realisierung mit Gattern

74AC11008
QUADRUPLE 2-INPUT POSITIVE-AND GATE

SCAS014C – AUGUST 1987 – REVISED APRIL 1996

- **Flow-Through Architecture Optimizes PCB Layout**
- **Center-Pin V_{CC} and GND Configurations Minimize High-Speed Switching Noise**
- ***EPIC*™ (Enhanced-Performance Implanted CMOS) 1-μm Process**
- **500-mA Typical Latch-Up Immunity at 125°C**
- **Package Options Include Plastic Small-Outline (D) and Thin Shrink Small-Outline (PW) Packages, and Standard Plastic 300-mil DIPs (N)**

D, N, OR PW PACKAGE
(TOP VIEW)



description

This device contains four independent 2-input AND gates. It performs the Boolean function $Y = A \bullet B$ or $Y = \overline{\overline{A} + \overline{B}}$ in positive logic.

The 74AC11008 is characterized for operation from –40°C to 85°C.

FUNCTION TABLE
(each gate)

INPUTS		OUTPUT
A	B	Y
H	H	H
L	X	L
X	L	L

7.1 Einleitung

Alternative Realisierung durch

- massenhaft produzierte
- darum **preisgünstige**
- nach der Fertigstellung in ihrer Funktion noch beeinflussbare
- **funktional vollständige**
- also **universelle** Standardbausteine

Programmable Logic Array (PLA)

Varianten PAL, PROM, FPGA, . . .
(zum Teil eingeschränkte Funktionalität)

Programmable Logic Array (PLA) Datenblatt von Lattice

Lattice Semiconductor Corporation

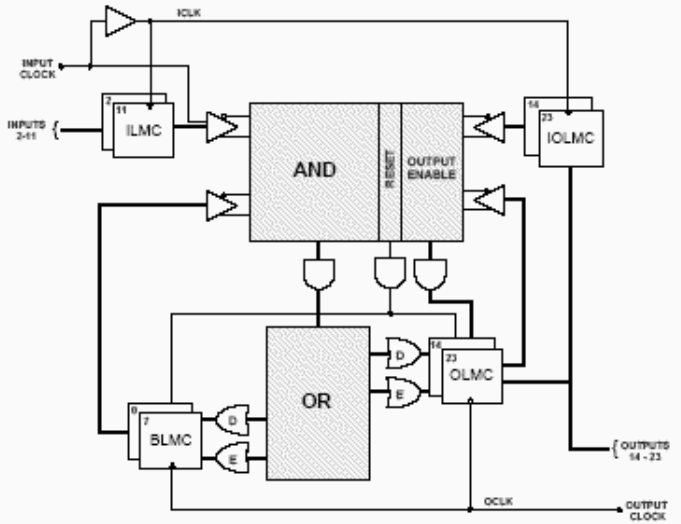
GAL6001

High Performance E²CMOS FPLA
Generic Array Logic™

Features

- HIGH PERFORMANCE E²CMOS® TECHNOLOGY
  - 30ns Maximum Propagation Delay
  - 27MHz Maximum Frequency
  - 12ns Maximum Clock to Output Delay
  - TTL Compatible 16mA Outputs
  - UltraMOS® Advanced CMOS Technology
- LOW POWER CMOS
  - 90mA Typical Icc
- E² CELL TECHNOLOGY
  - Reconfigurable Logic
  - Reprogrammable Cells
  - 100% Tested/100% Yields
  - High Speed Electrical Erasure (<100ms)
  - 20 Year Data Retention
- UNPRECEDENTED FUNCTIONAL DENSITY
  - 78 x 64 x 36 FPLA Architecture
  - 10 Output Logic Macrocells
  - 8 Buried Logic Macrocells
  - 20 Input and I/O Logic Macrocells
- HIGH-LEVEL DESIGN FLEXIBILITY
  - Asynchronous or Synchronous Clocking
  - Separate State Register and Input Clock Pins
  - Functional Superset of Existing 24-pin PAL® and FPLA Devices

Functional Block Diagram



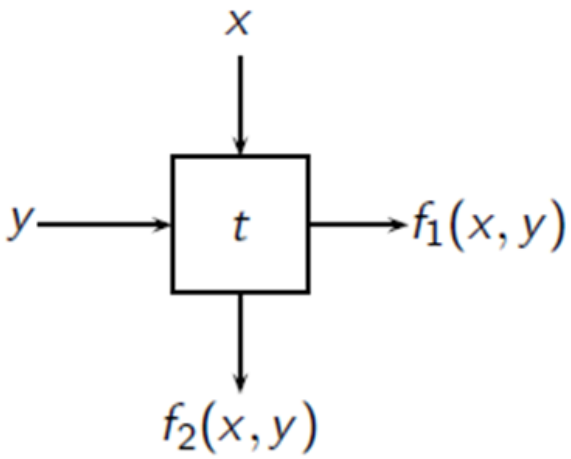
Macrocell Names

ILMC	INPUT LOGIC MACROCELL
IOLMC	I/O LOGIC MACROCELL
BLMC	BURIED LOGIC MACROCELL
OLMC	OUTPUT LOGIC MACROCELL

7. Programmierbare Bausteine

1. Einleitung 
2. **Grundbausteine**
3. Realisierung von Monomen und Polynomen
4. PLA als Speicher
5. Software PLAs

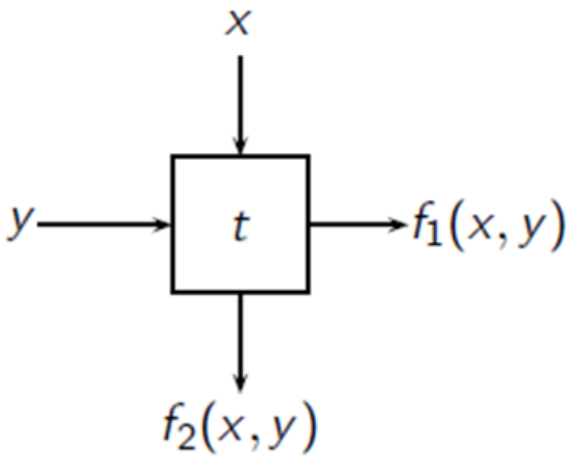
PLA Grundbausteine



Name	Typ	$f_1(x,y)$	$f_2(x,y)$
Identer	0	y	x
Addierer	1	x ∨ y	x
Multiplizierer	2	y	x ∧ y
Negat-Multiplizierer	3	y	x ∧ $\overline{y}$

Funktional vollständig?

PLA Grundbausteine



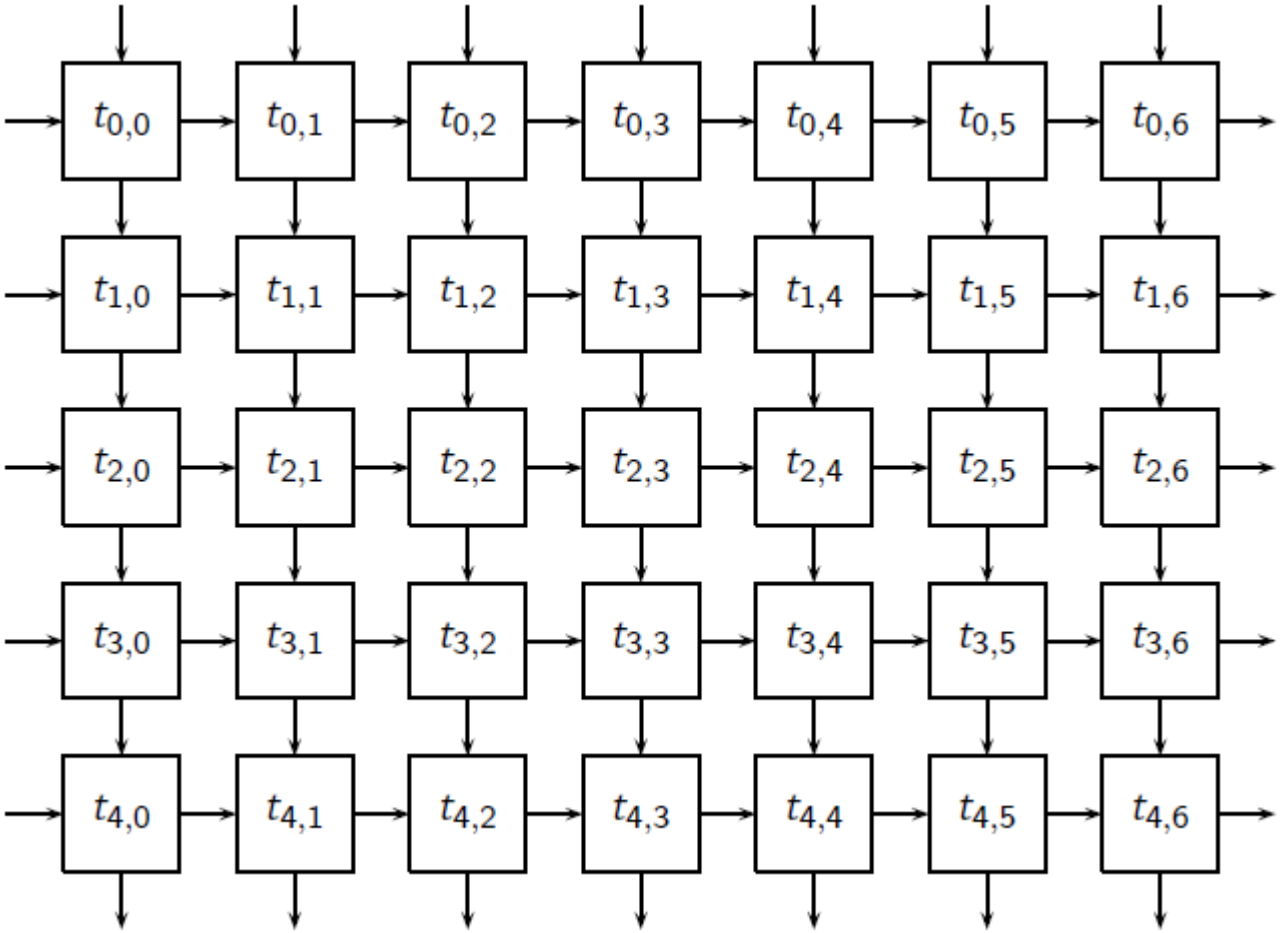
Name	Typ	$f_1(x,y)$	$f_2(x,y)$
Identer	0	y	x
Addierer	1	$x \vee y$	x
Multiplizierer	2	y	$x \wedge y$
Negat-Multiplizierer	3	y	$x \wedge \overline{y}$

Funktional vollständig!

- UND-Verknüpfung: Typ 2, $f_2(x,y) = x \wedge y$
- ODER-Verknüpfung: Typ 1, $f_1(x,y) = x \vee y$
- Negation: Typ 3, $f_2(1,y) = 1 \wedge \overline{y} = \overline{y}$

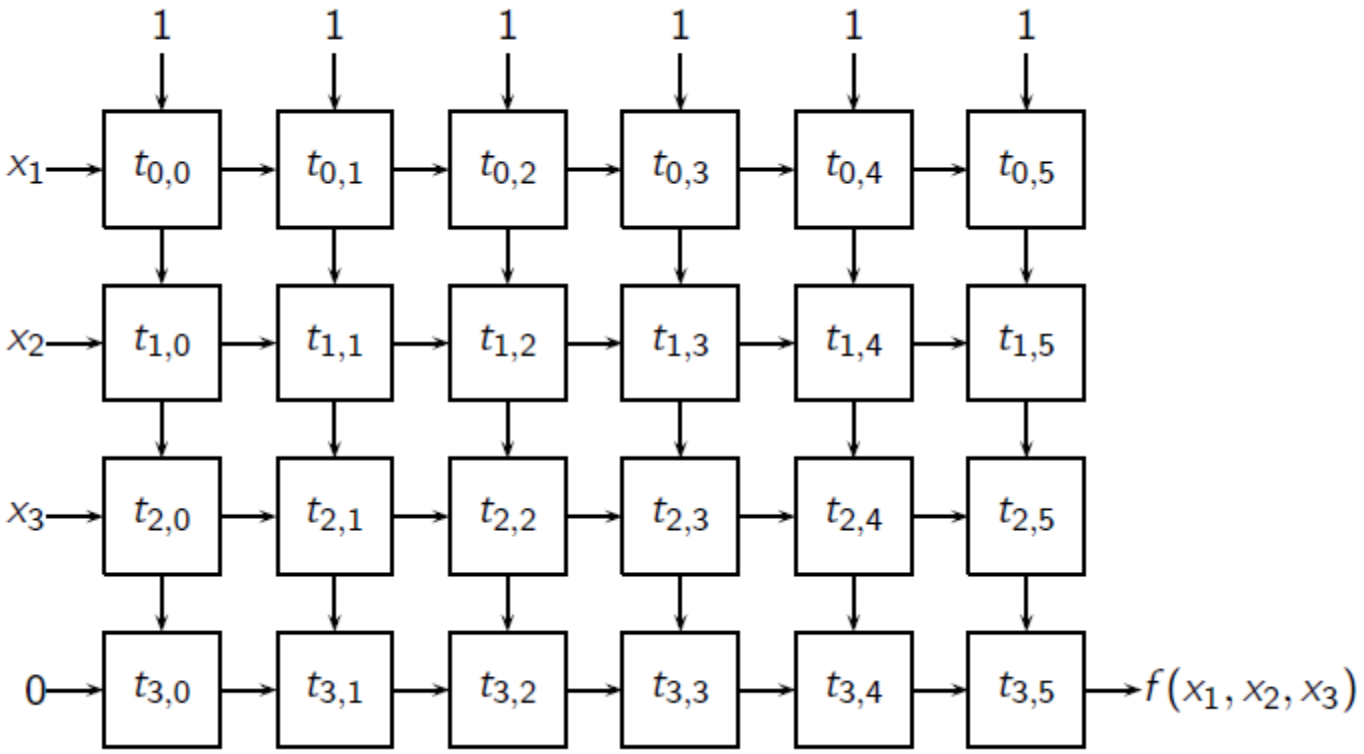
7.2 Grundbausteine

Aufbau eines PLA



7.2 Grundbausteine

PLA für $f : \{0, 1\}^3 \rightarrow \{0, 1\}$



Wie wählt man die Bausteintypen?

7. Programmierbare Bausteine

1. Einleitung ✓
2. Grundbausteine ✓
3. **Realisierung von Monomen und Polynomen**
4. PLA als Speicher
5. Software PLAs

7.3 Realisierung von Monomen und Polynomen

Auswahl der Bausteine

Feststellung jede Funktion als Polynom darstellbar

Erinnerung Polynom = Disjunktion einiger Monome

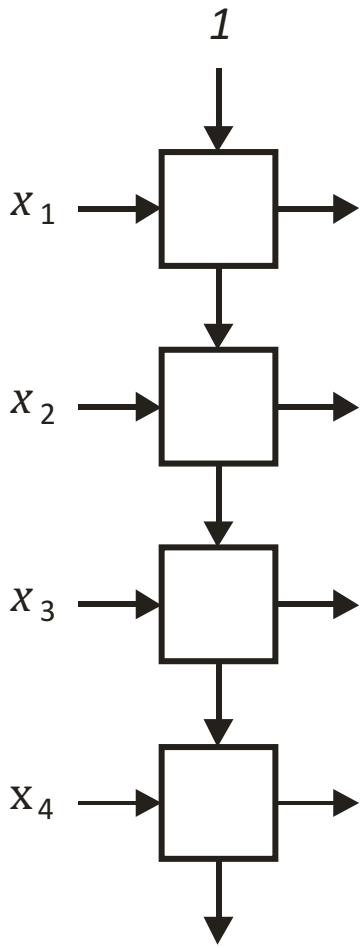
Erster Schritt

- Wie realisieren wir Monome?
- exemplarisch am Beispiel $x_1\overline{x}_2x_4$

7.3 Realisierung von Monomen und Polynomen

Realisierung von Monomen

Beispiel: Monom $x_1\overline{x}_2x_4$

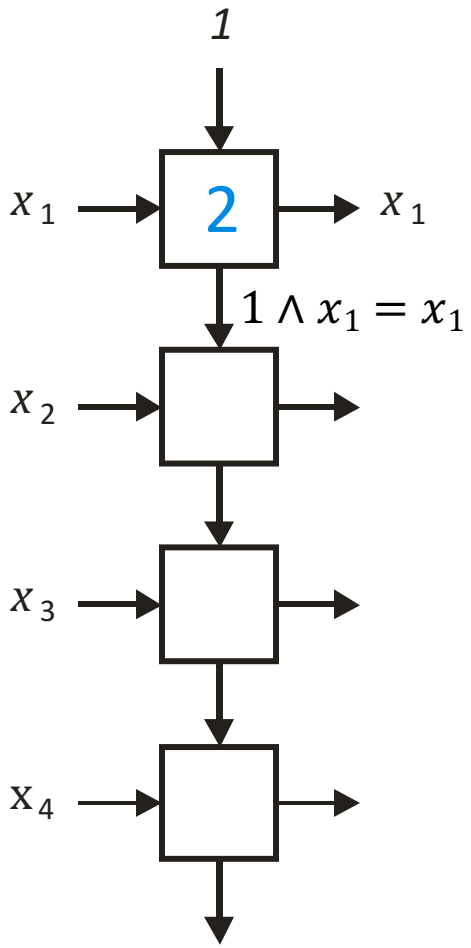


Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \overline{l}$

7.3 Realisierung von Monomen und Polynomen

Realisierung von Monomen

Beispiel: Monom $x_1\overline{x}_2x_4$

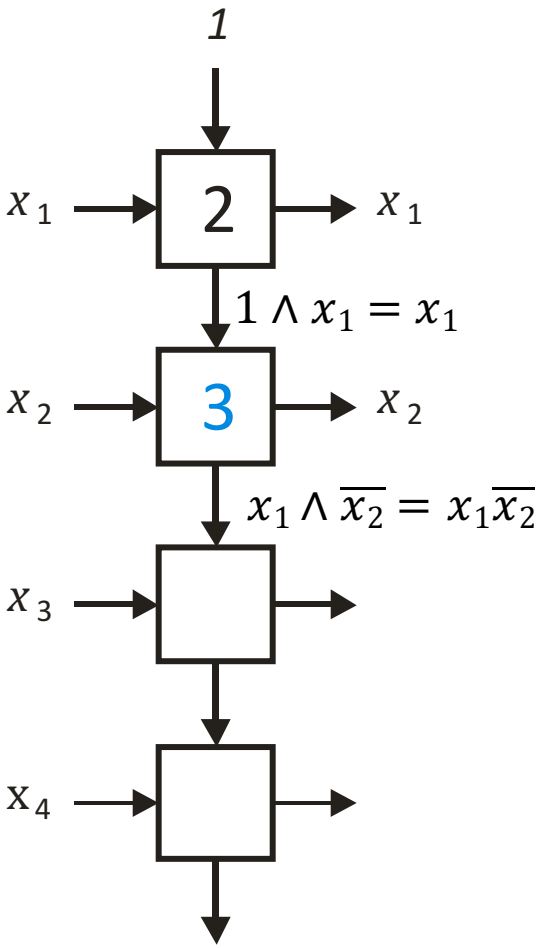


Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	**2**	$\boldsymbol{l}$	$\boldsymbol{o \wedge l}$
Negat-Multiplizierer	3	l	$o \wedge \overline{l}$

7.3 Realisierung von Monomen und Polynomen

Realisierung von Monomen

Beispiel: Monom $x_1\overline{x_2}x_4$

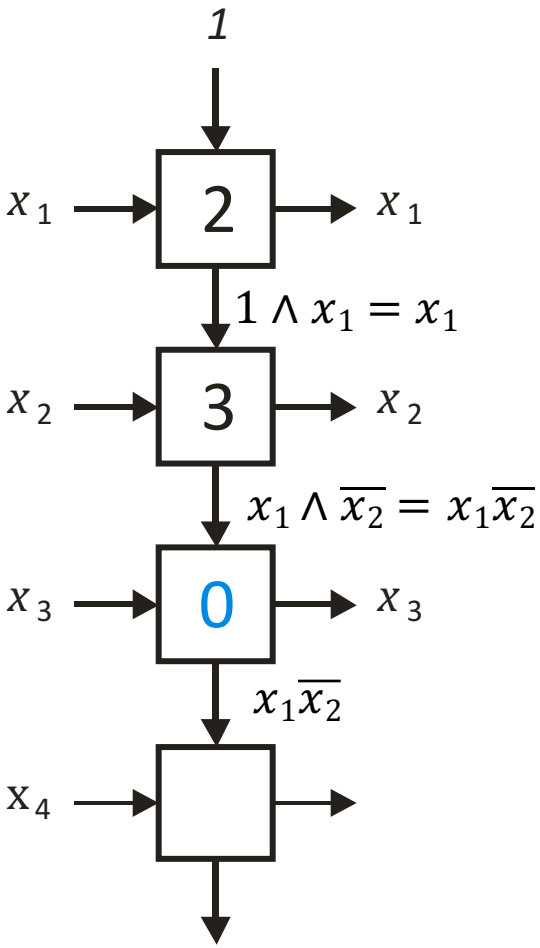


Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	**3**	$\boldsymbol{l}$	$\boldsymbol{o \wedge \bar{l}}$

7.3 Realisierung von Monomen und Polynomen

Realisierung von Monomen

Beispiel: Monom $x_1\overline{x_2}x_4$

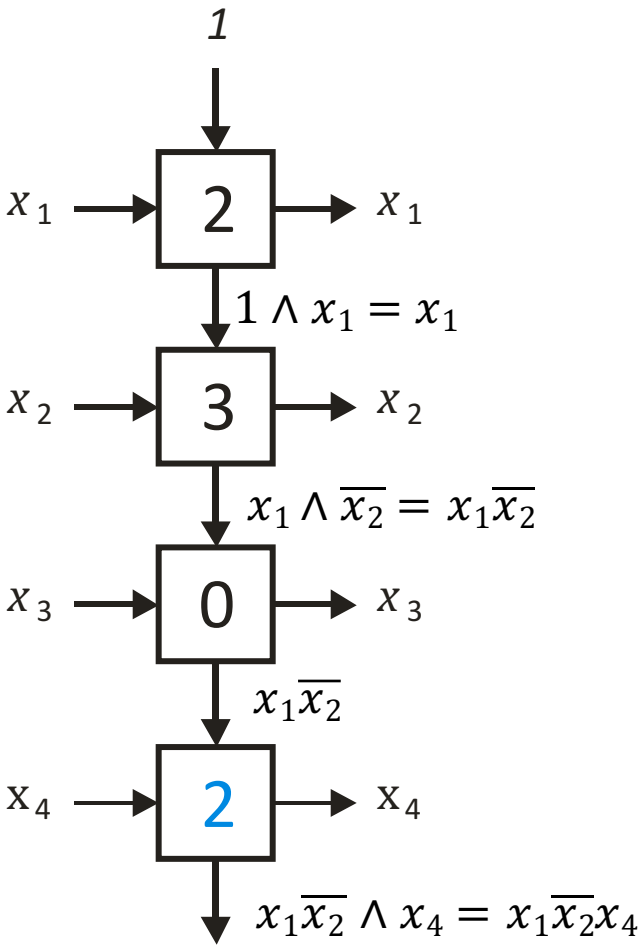


Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	**0**	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \overline{l}$

7.3 Realisierung von Monomen und Polynomen

Realisierung von Monomen

Beispiel: Monom $x_1\overline{x_2}x_4$

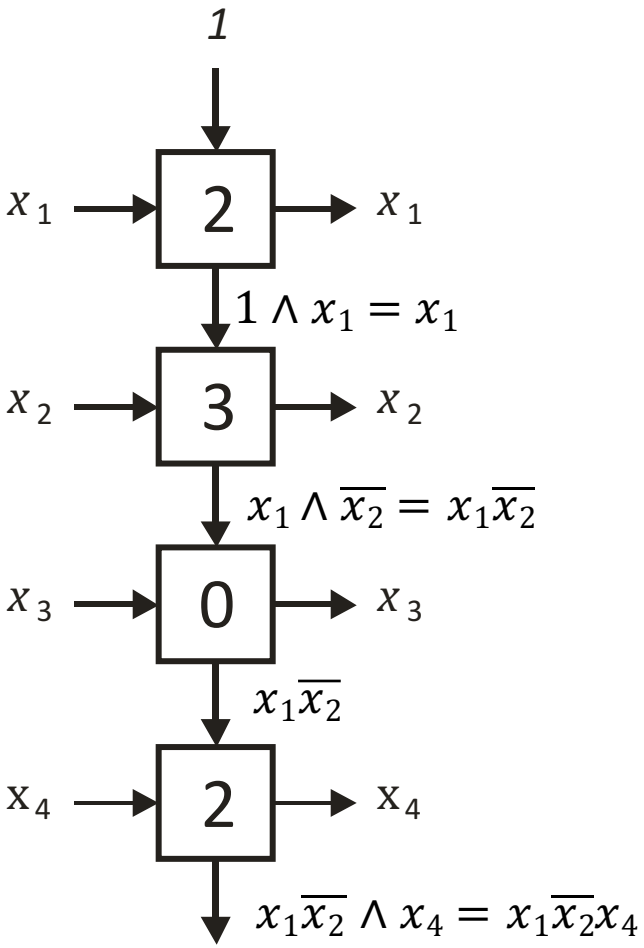


Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	**2**	$\boldsymbol{l}$	$\boldsymbol{o \wedge l}$
Negat-Multiplizierer	3	l	$o \wedge \overline{l}$

7.3 Realisierung von Monomen und Polynomen

Realisierung von Monomen

Beispiel: Monom $x_1\overline{x_2}x_4$



Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	**2**	$\boldsymbol{l}$	$\boldsymbol{o \wedge l}$
Negat-Multiplizierer	3	l	$o \wedge \overline{l}$

Monomrealisierung

- falls Variable fehlt Typ 0
- falls x_i vorkommt Typ 2
- falls $\overline{x_i}$ vorkommt Typ 3

7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beobachtung

- für f : $\{0, 1\}^n \rightarrow \{0, 1\}$
- **k** verschiedene **Monome** $m_1, m_2, \ldots, m_k$
- in *n* Zeilen und *k* Spalten realisierbar

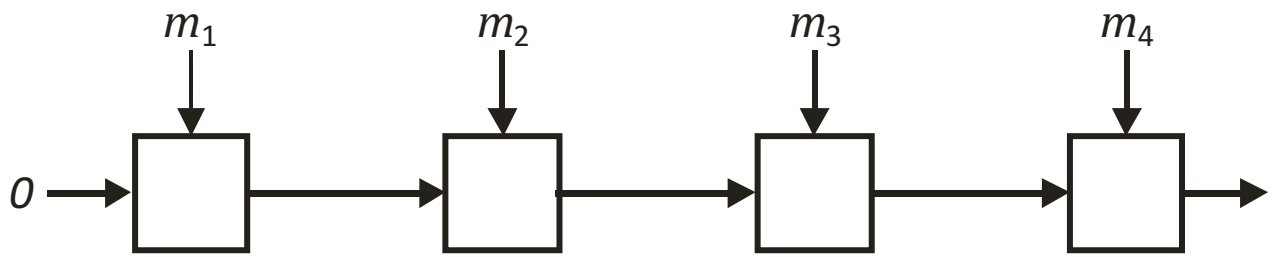
Wie können wir f realisieren, z.B. $f = m_1 \vee m_2 \vee m_4$?

7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beispiel: Polynom f = $m_1 \vee m_2 \vee m_4$

Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \bar{l}$

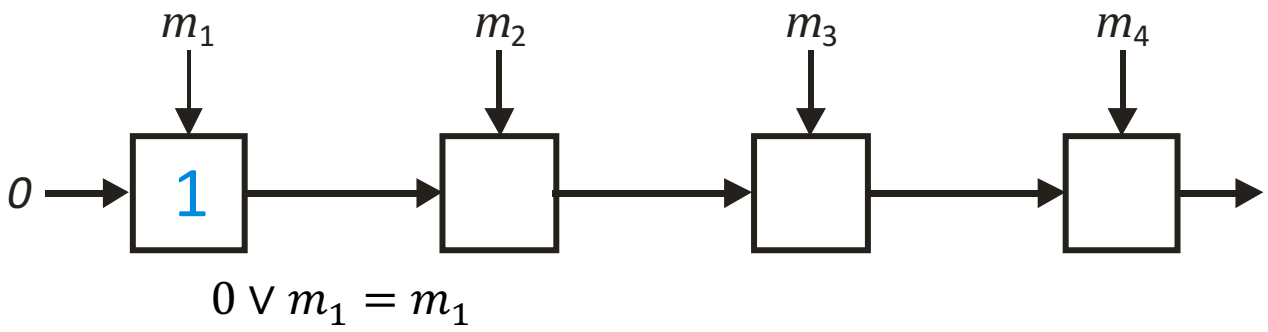


7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beispiel: Polynom $f = m_1 \vee m_2 \vee m_4$

Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	**1**	$\boldsymbol{l \vee o}$	$\boldsymbol{o}$
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \bar{l}$

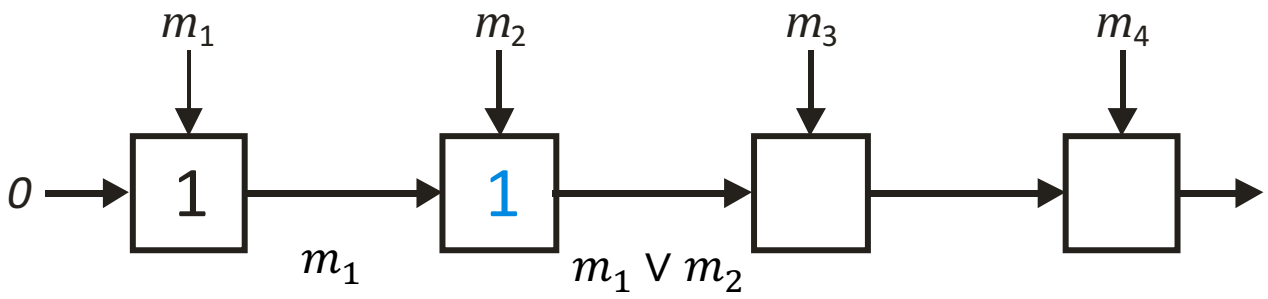


7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beispiel: Polynom $f = m_1 \vee m_2 \vee m_4$

Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	0	l	o
Addierer	**1**	$\boldsymbol{l \vee o}$	$\boldsymbol{o}$
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \bar{l}$

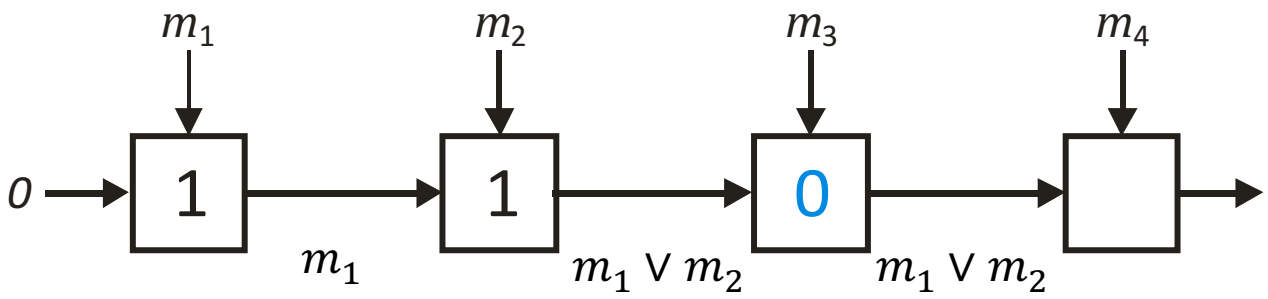


7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beispiel: Polynom $f = m_1 \vee m_2 \vee m_4$

Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	**0**	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \bar{l}$

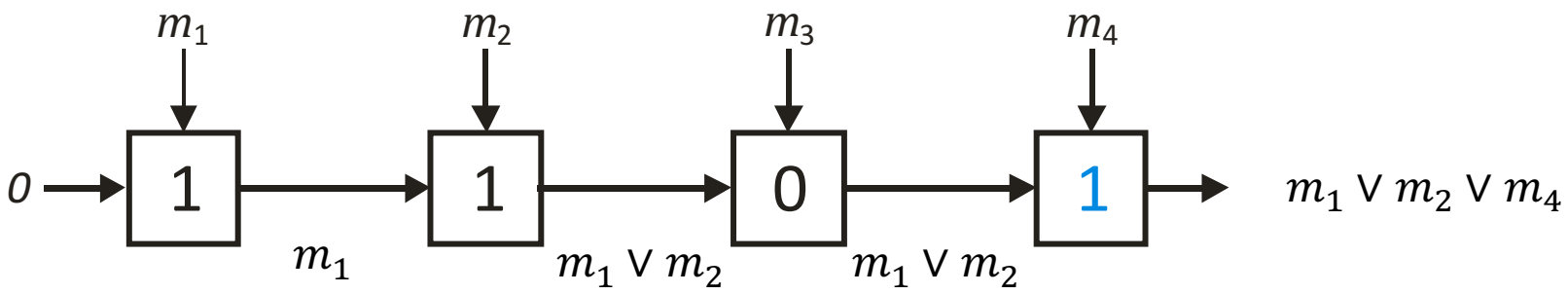


7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beispiel: Polynom $f = m_1 \vee m_2 \vee m_4$

Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	**0**	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \bar{l}$

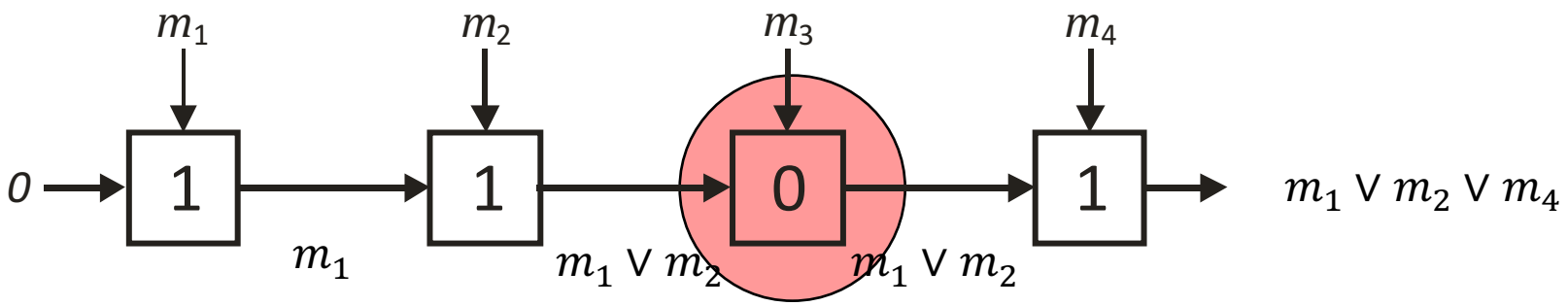


7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beispiel: Polynom $f = m_1 \vee m_2 \vee m_4$

Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	**0**	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \bar{l}$



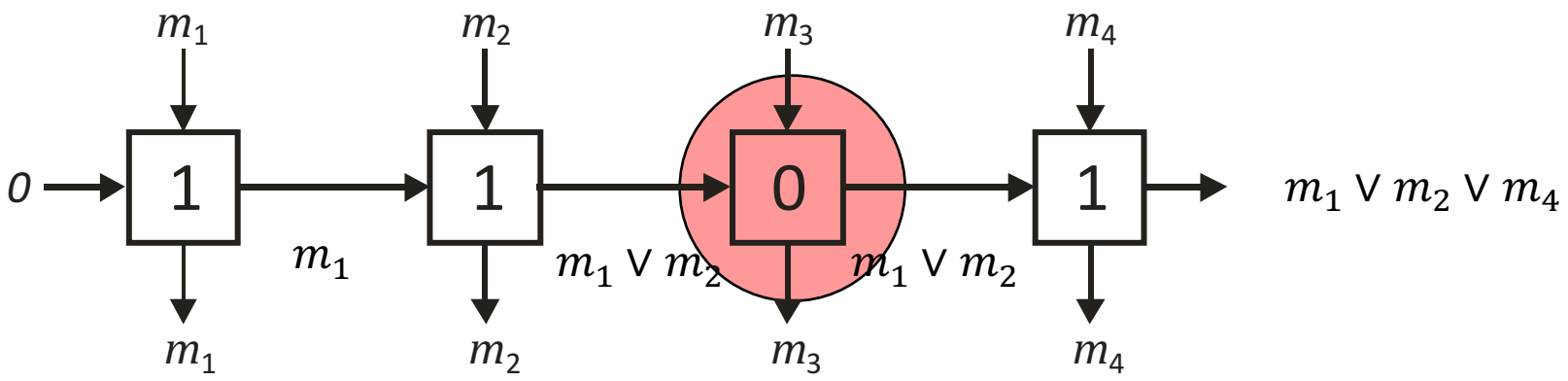
Warum ein Grundbaustein für das Weglassen eines Monoms?

7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beispiel: Polynom $f = m_1 \vee m_2 \vee m_4$

Name	Typ	$f_r(o,l)$	$f_u(o,l)$
Identer	**0**	l	o
Addierer	1	$l \vee o$	o
Multiplizierer	2	l	$o \wedge l$
Negat-Multiplizierer	3	l	$o \wedge \bar{l}$

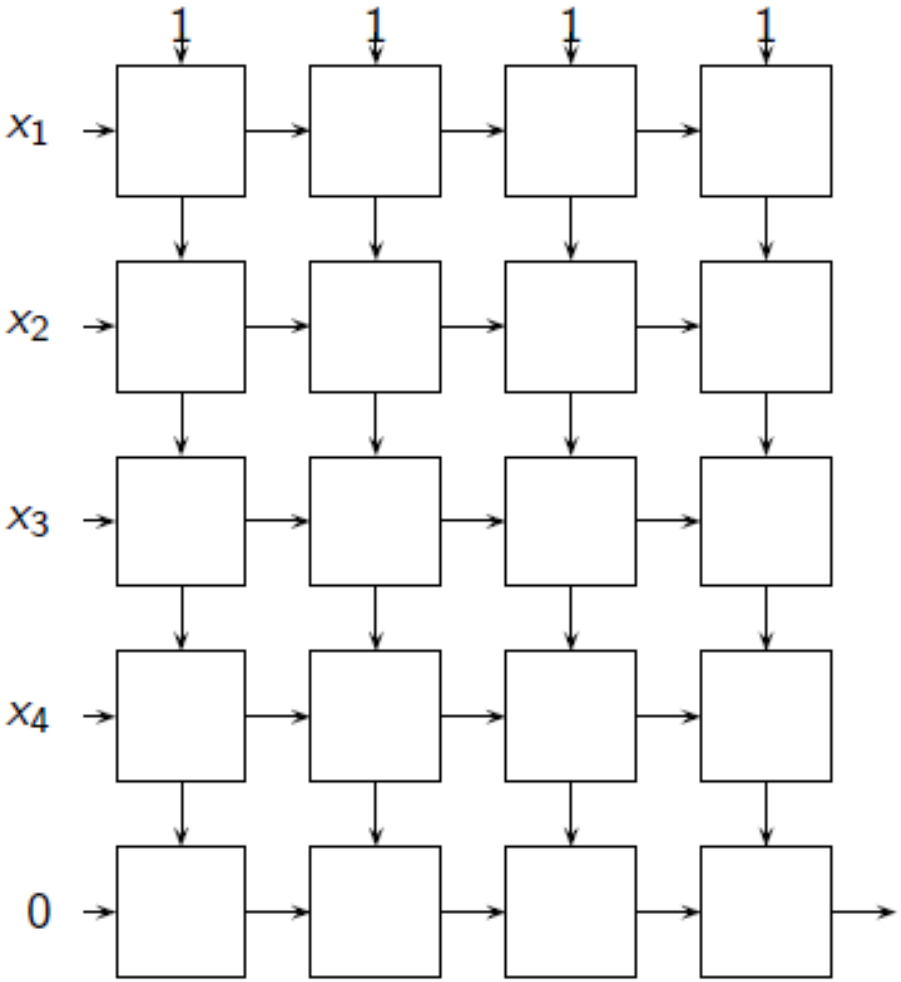


Warum ein Grundbaustein für das Weglassen eines Monoms?

7.3 Realisierung von Monomen und Polynomen

PLA: ein konkretes Beispiel

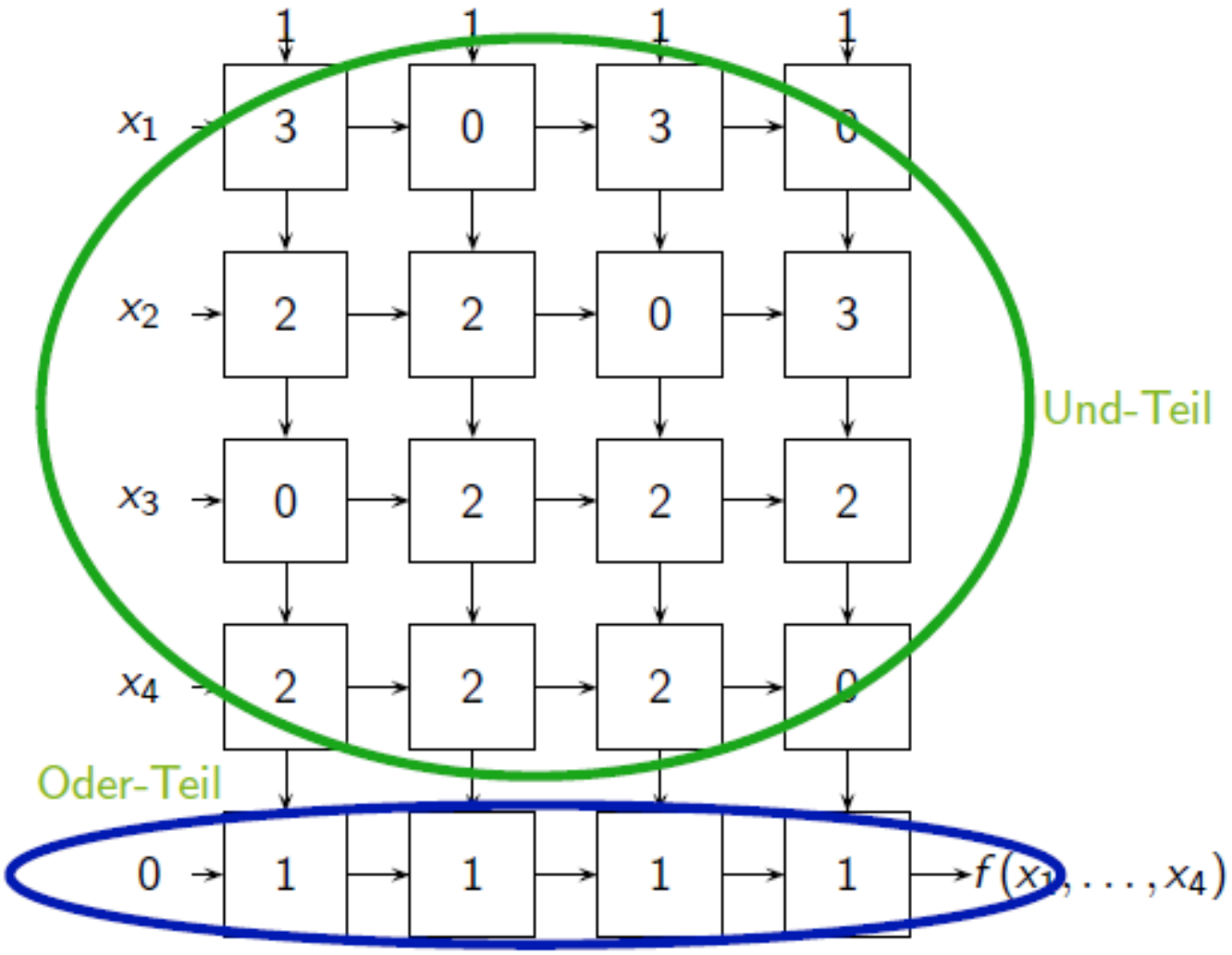
Beispiel $f(x_1, x_2, x_3, x_4) = \overline{x_1}\, x_2\, x_4 \vee x_2\, x_3\, x_4 \vee \overline{x_1}\, x_3\, x_4 \vee \overline{x_2}\, x_3$



7.3 Realisierung von Monomen und Polynomen

PLA: ein konkretes Beispiel

Beispiel $f(x_1, x_2, x_3, x_4) = \overline{x_1}\, x_2\, x_4 \vee x_2\, x_3\, x_4 \vee \overline{x_1}\, x_3\, x_4 \vee \overline{x_2}\, x_3$



7.3 Realisierung von Monomen und Polynomen

Realisierung von Polynomen

Beobachtung 1

- für f : $\{0, 1\}^n \rightarrow \{0, 1\}$
- k verschiedene **Monome** $m_1, m_2, \ldots, m_k$
- in *n+1* Zeilen und *k* Spalten realisierbar

Beobachtung 2

- Wir können jede Funktion f : $\{0, 1\}^n \rightarrow \{0, 1\}^m$,
- für deren **Polynom** insgesamt k **Implikanten** ausreichen,
- mit einem PLA mit *n + m* Zeilen und *k* Spalten realisieren.

7. Programmierbare Bausteine

1. Einleitung ✓
2. Grundbausteine ✓
3. Realisierung von Monomen und Polynomen ✓
4. **PLA als Speicher**
5. Software PLAs

7.4 PLA als Speicher

PLA als ROM

Aufgabe: Speichere 2^n „Wörter“ der Länge m.

$$w_0 = w_{0,0}w_{0,1}w_{0,2}\cdots w_{0,m-1} \in \{0,1\}^m$$

$$w_1 = w_{1,0}w_{1,1}w_{1,2}\cdots w_{1,m-1} \in \{0,1\}^m$$

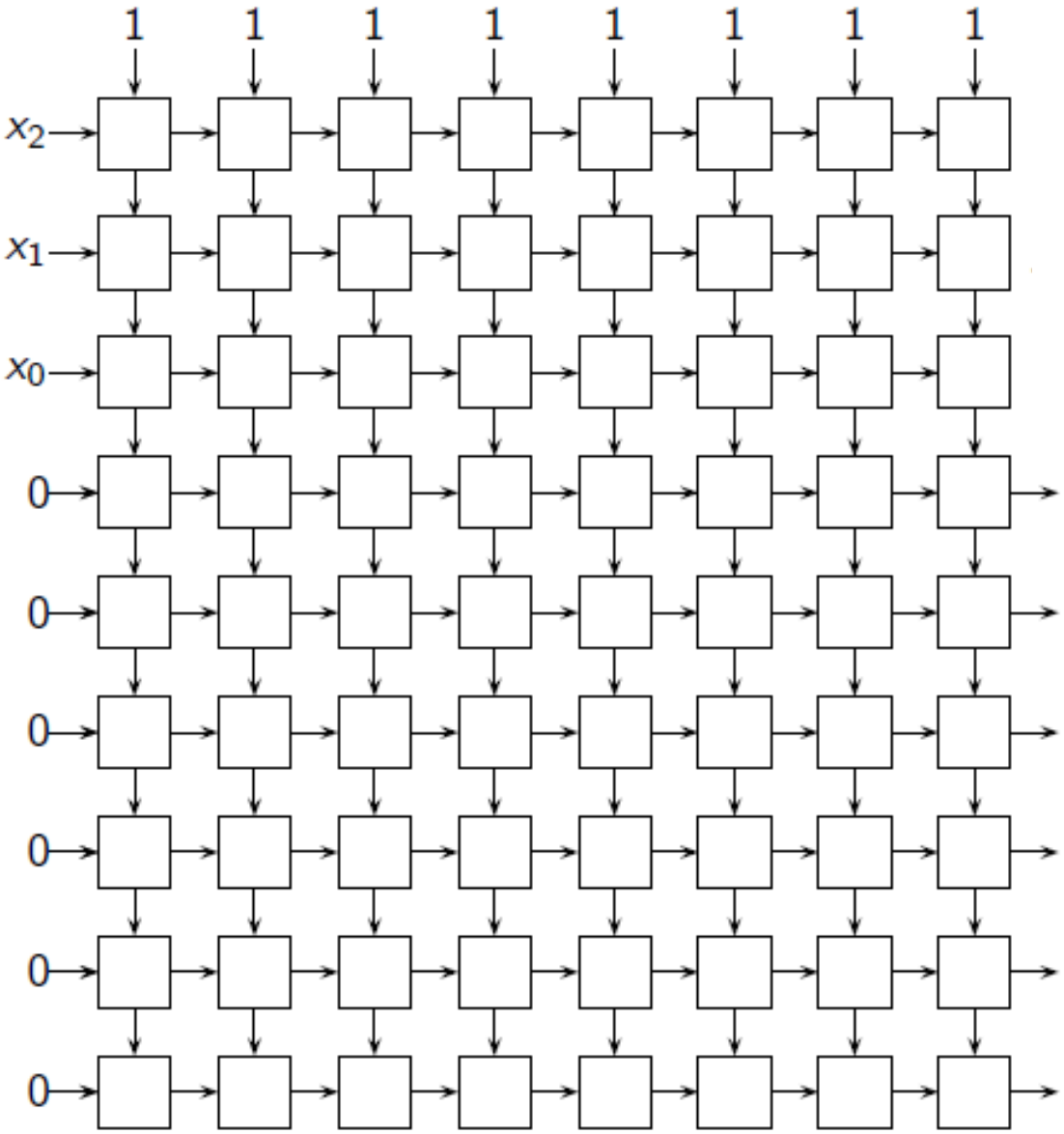
$$w_2 = w_{2,0}w_{2,1}w_{2,2}\cdots w_{2,m-1} \in \{0,1\}^m$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$w_{2^n-1} = w_{2^n-1,0}\, w_{2^n-1,1}\, w_{2^n-1,2}\cdots w_{2^n-1,m-1} \in \{0,1\}^m$$

Benutze	**PLA** mit n + m Zeilen, 2^n Spalten
Beobachtung	$m\cdot 2^n$ Zellen für $m\cdot 2^n$ zu speichernde Bits mindestens erforderlich
Adressierung	mit jeweils n Bits in n zusätzlichen Zeilen

7.4 PLA als Speicher



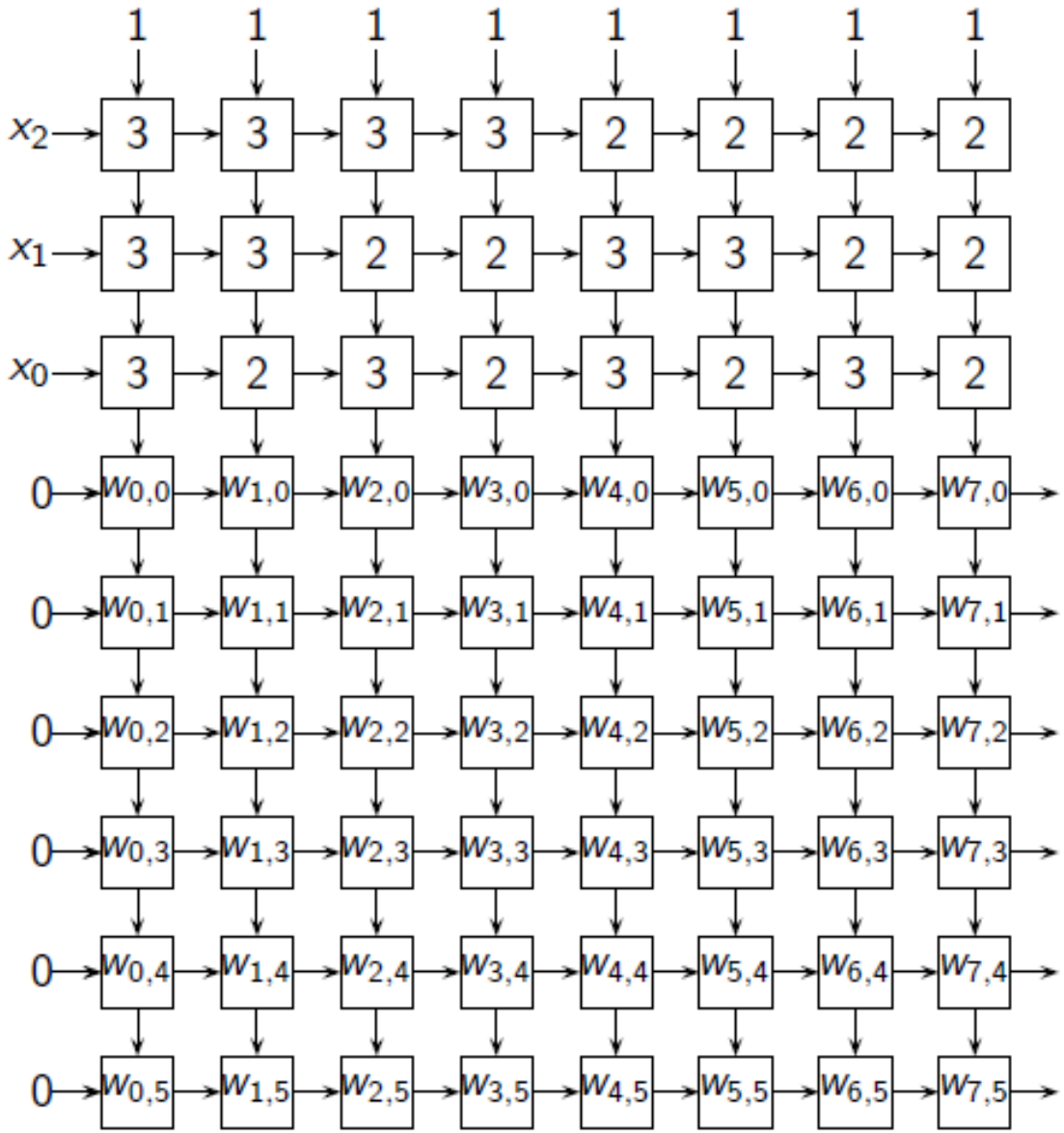
Beispiel

n = 3, m = 6

2^3 = 8 Wörter

jeweils 6 Bits

7.4 PLA als Speicher



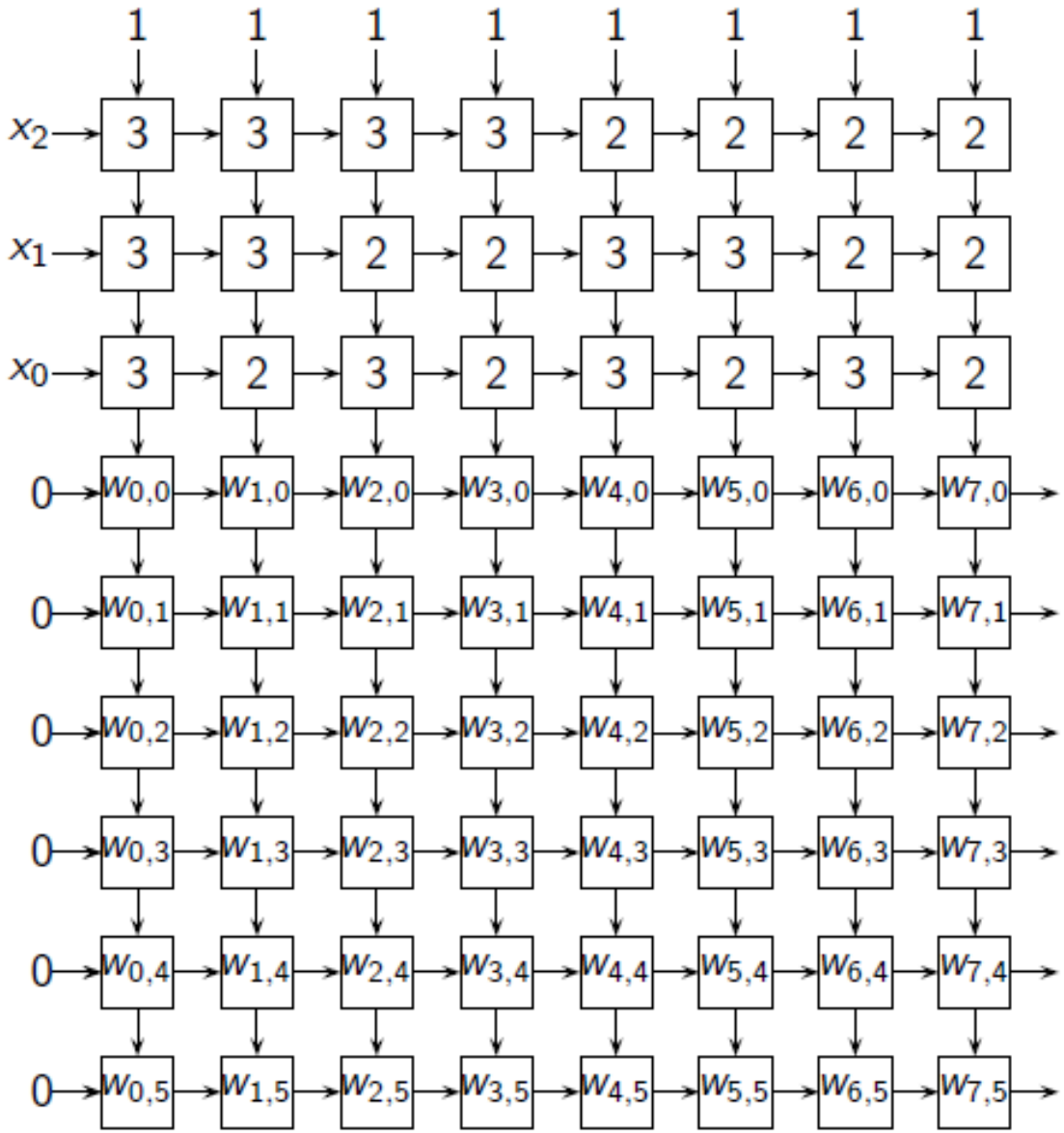
Beispiel

n = 3, m = 6

2^3 = 8 Wörter

jeweils 6 Bits

7.4 PLA als Speicher



Beispiel

n = 3, m = 6

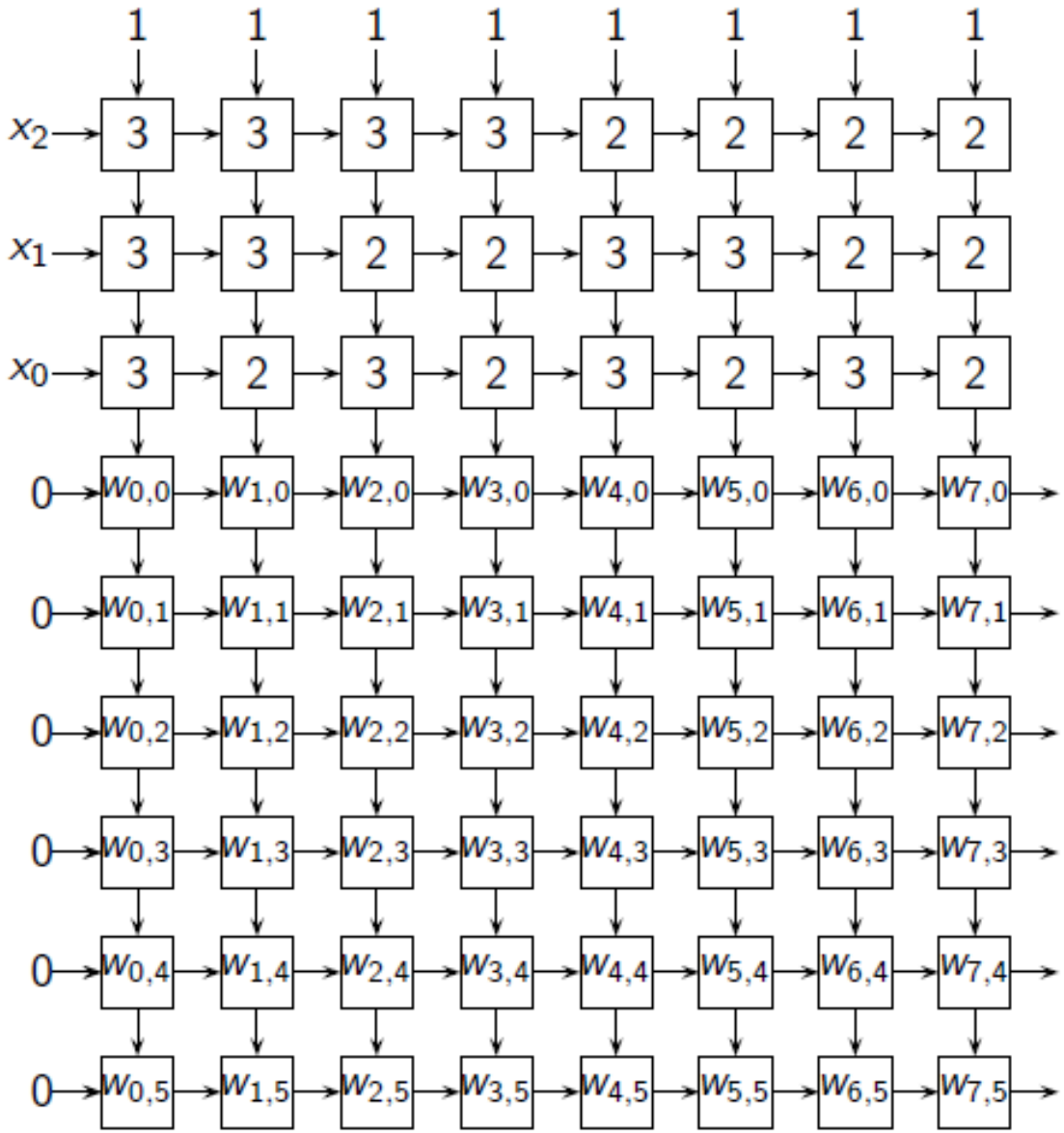
2^3 = 8 Wörter

jeweils 6 Bits

Beispiel

lies w_5

7.4 PLA als Speicher



Beispiel

n = 3, m = 6

2^3 = 8 Wörter

jeweils 6 Bits

Beispiel

lies w_5

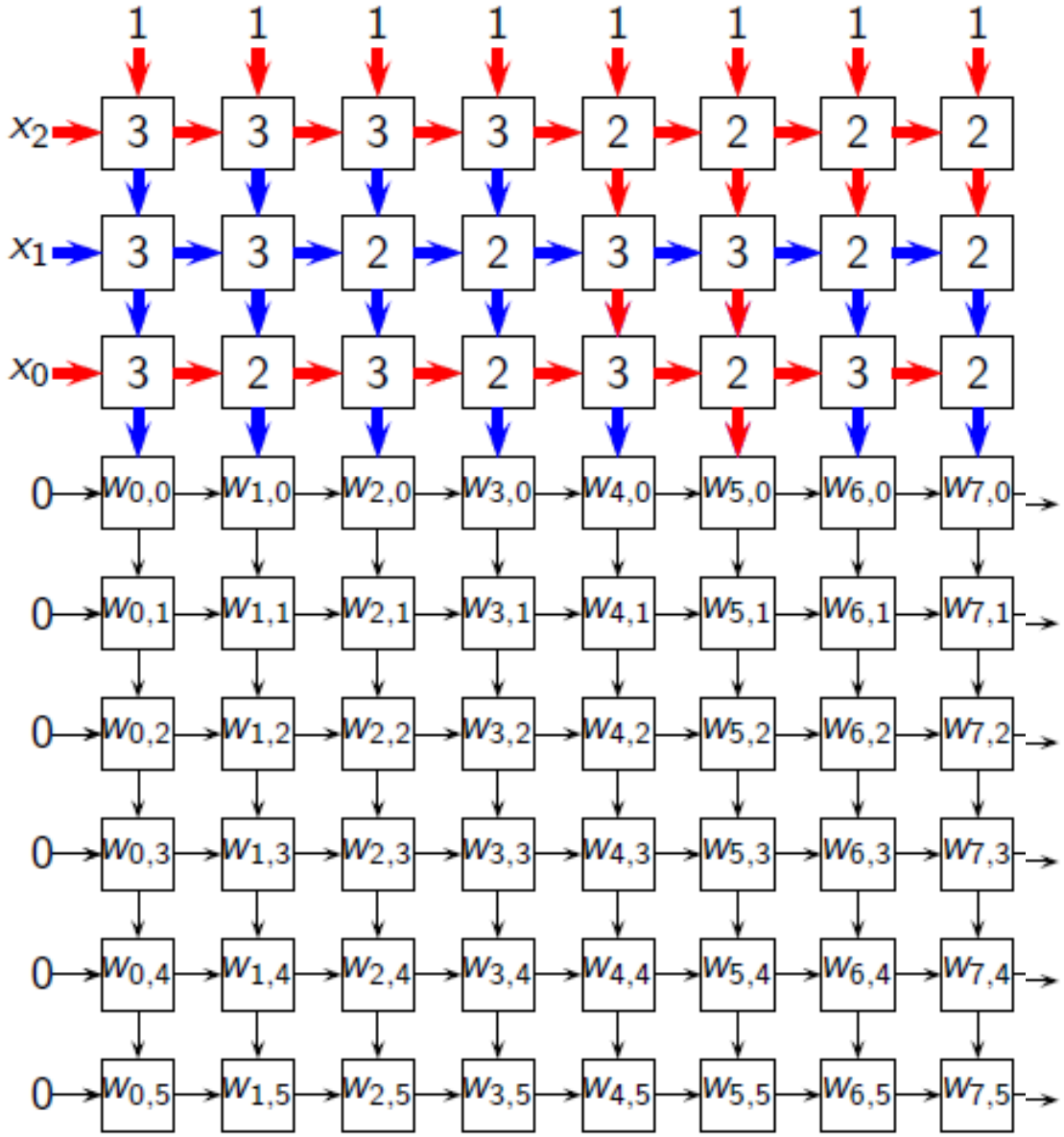
$5 = (101)_2$

$x_2 = 1$

$x_1 = 0$

$x_0 = 1$

7.4 PLA als Speicher



Beispiel

n = 3, m = 6

2^3 = 8 Wörter

jeweils 6 Bits

Beispiel

lies w_5

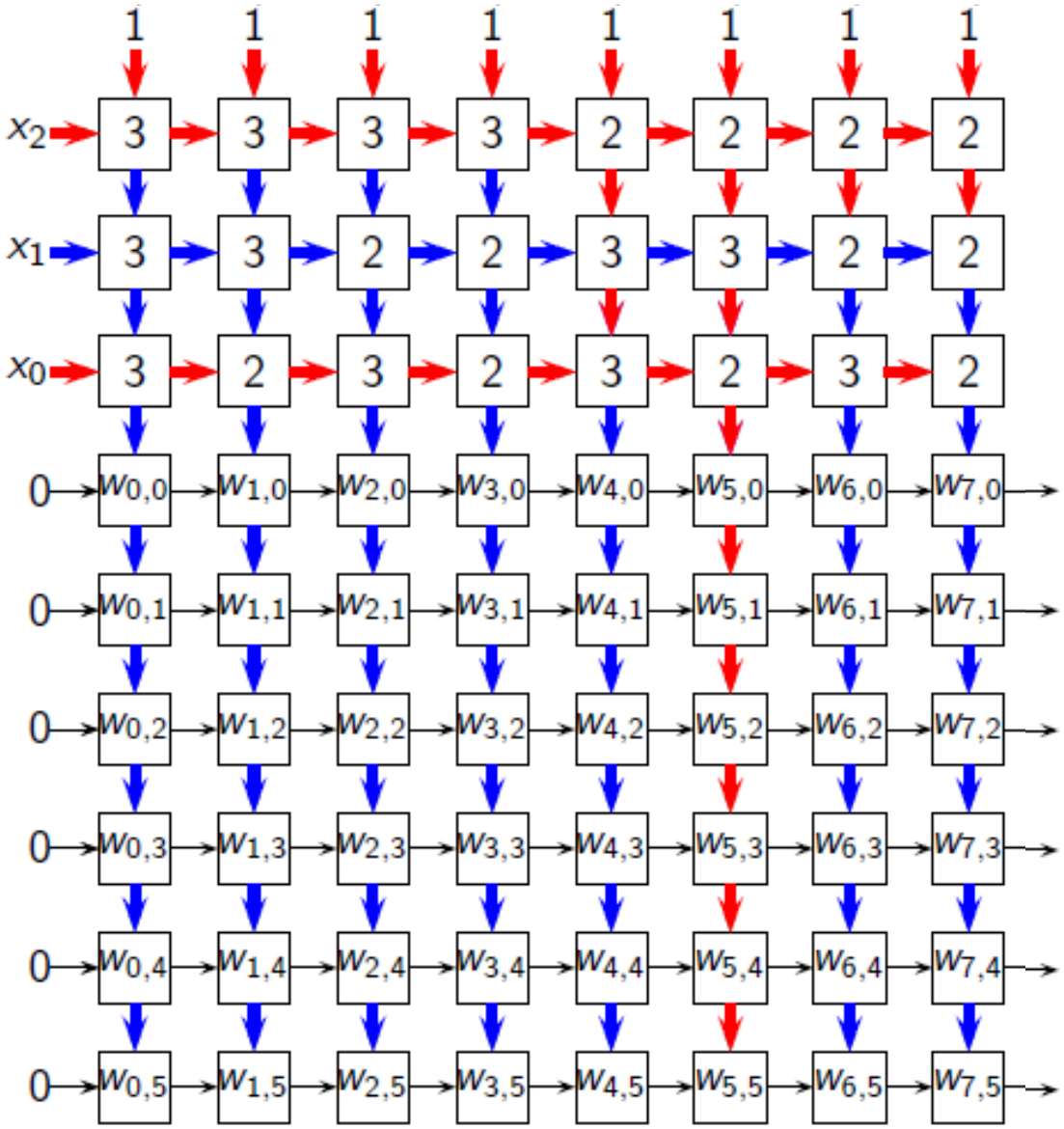
$5 = (101)_2$

$x_2 = 1$

$x_1 = 0$

$x_0 = 1$

7.4 PLA als Speicher



Beispiel

n = 3, m = 6

2^3 = 8 Wörter

jeweils 6 Bits

Beispiel

lies w_5

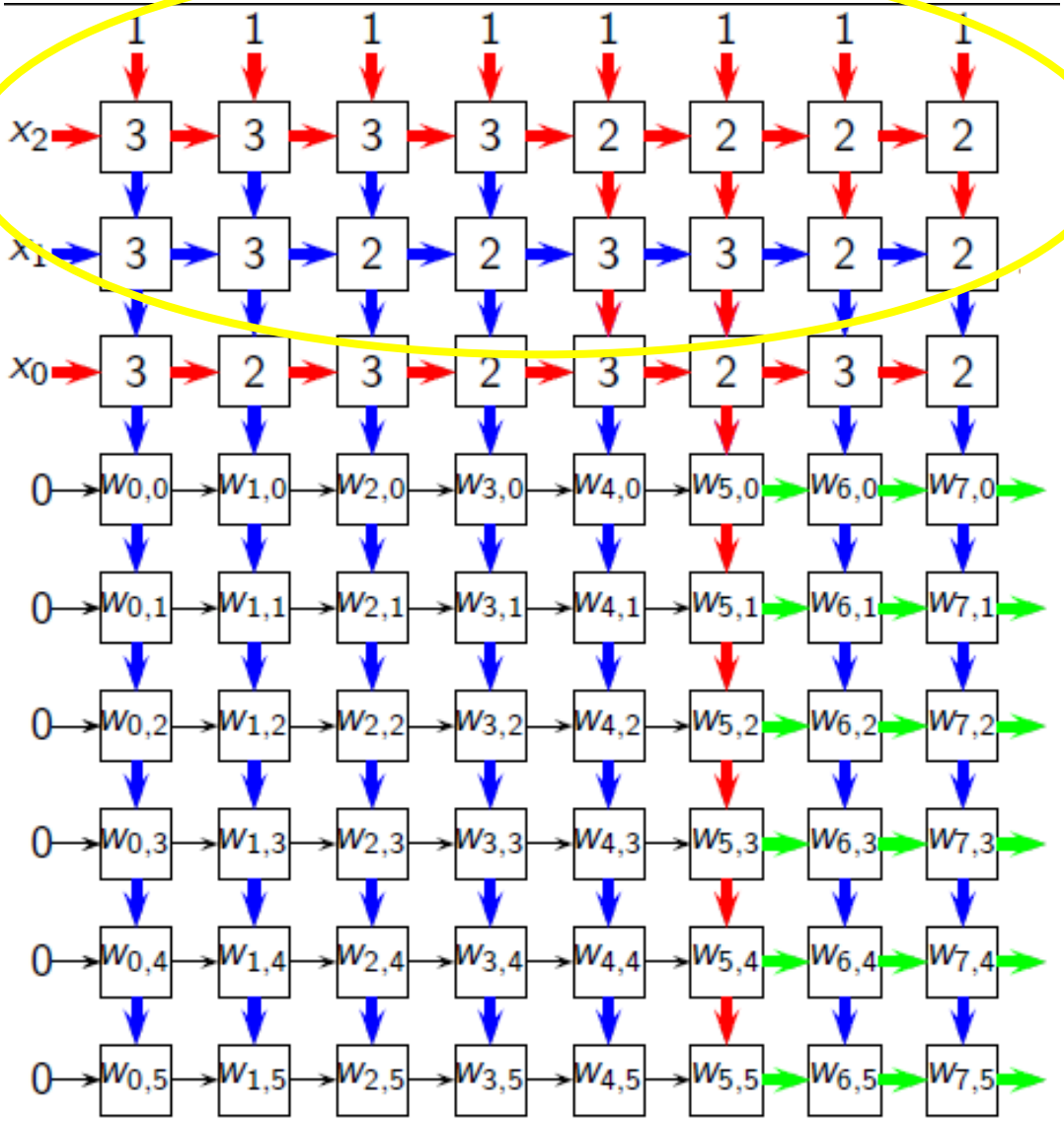
$5 = (101)_2$

$x_2 = 1$

$x_1 = 0$

$x_0 = 1$

7.4 PLA als Speicher



Beispiel

n = 3, m = 6

2^3 = 8 Wörter

jeweils 6 Bits

Beispiel

lies w_5

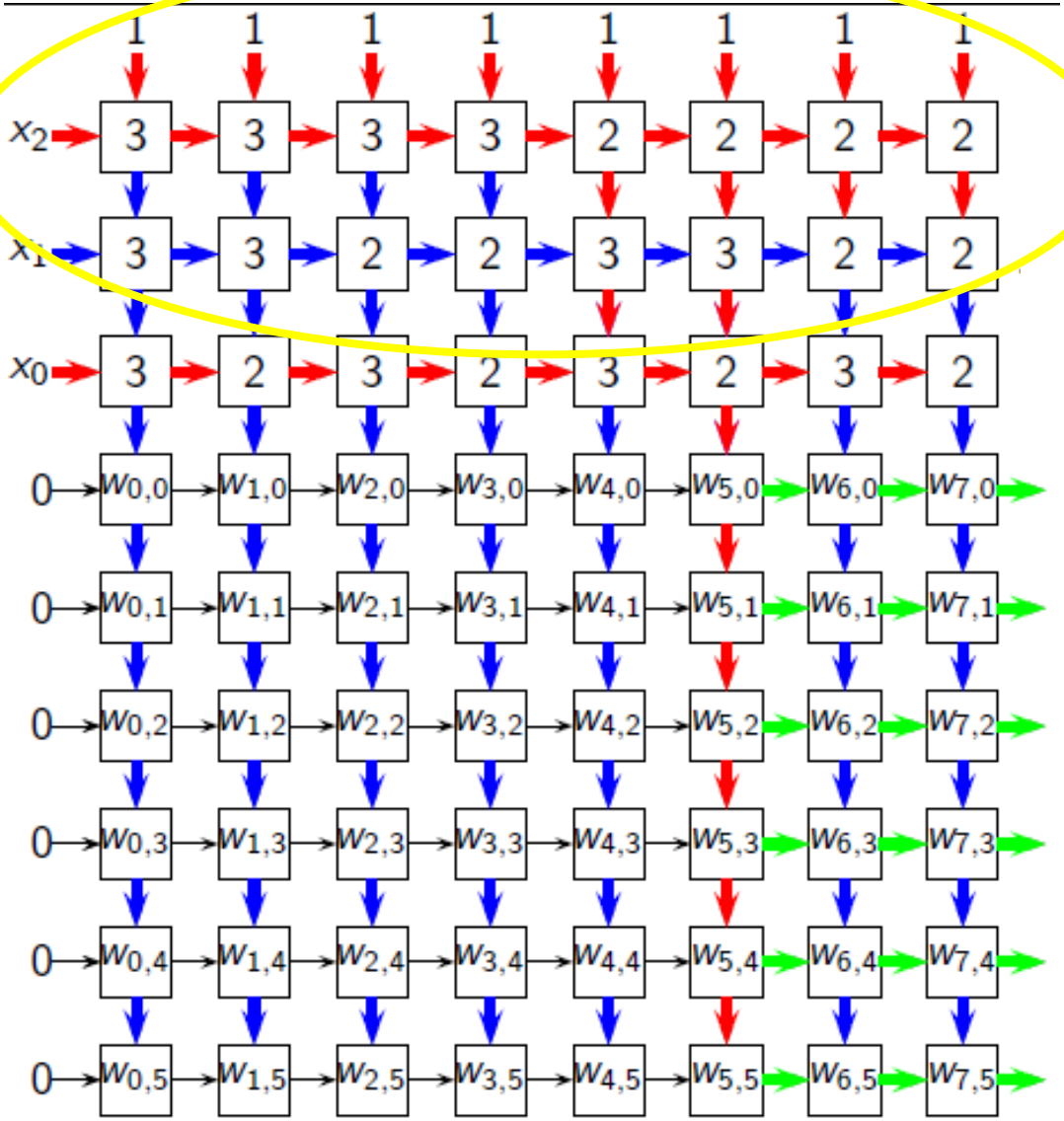
$5 = (101)_2$

$x_2 = 1$

$x_1 = 0$

$x_0 = 1$

7.4 PLA als Speicher



Beispiel

n = 3, m = 6

2^3 = 8 Wörter

jeweils 6 Bits

Beispiel

lies $w_5 : 5 = (101)_2$

$x_2 = 1$ $x_1 = 0$ $x_0 = 1$

Anmerkung

PLAs mit festem UND-Teil werden als PROM verkauft.

7.4 PLA als Speicher

Zwischen-Fazit PLAs

PLAs sind preiswerte, universelle Bausteine, die

- beliebige boolesche Funktionen leicht realisierbar machen,
- Minimalpolynomdarstellungen motivieren,
- Speicherung von 2^n Wötern der Länge m in einem

  $(n + m) \times 2^n$ -PLA erlauben.

Nachteil nur einmal programmierbar

Wie kann man PLAs beliebig neu programmierbar machen?

7. Programmierbare Bausteine

1. Einleitung ✓
2. Grundbausteine ✓
3. Realisierung von Monomen und Polynomen ✓
4. PLA als Speicher ✓
5. **Software PLAs**

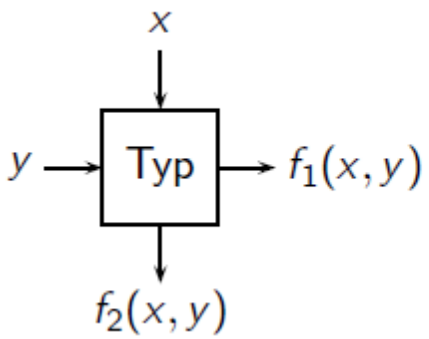
Software-PLAs

Beobachtung vier verschiedene Typen, mit zwei Bits codierbar

Idee

- erweitere PLA-Baustein um zwei zusätzliche Eingaben,
- die den Baustein-Typ codieren

Software-PLAs



Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	$x \vee y$	x
2	y	$x\,y$
3	y	$x\,\overline{y}$

Beobachtung vier verschiedene Typen, mit zwei Bits codierbar

Idee

- erweitere PLA-Baustein um zwei zusätzliche Eingaben,
- die den Baustein-Typ codieren

Lösung f_1 und f_2 als Funktionen von (s, t, x, y) darstellen

- $f_1(s, t, x, y) = y \vee \bar{s}\,t\,x$
- $f_2(s, t, x, y) = \bar{s}x \vee \bar{t}xy \vee tx\bar{y}$

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

1. Nachdenken über Kodierungen

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

1. Nachdenken über Kodierungen

- Wir betrachten beide Funktionen getrennt
- 4 unterschiedliche Funktionen
  - Kodierung der Funktion mittels zwei Bits (s, t)
  - jede Funktion hat zusätzlich zwei Eingangsvariablen (x, y)
  - f_1 und f_2 haben dann jeweils 4 Eingangsvariablen (s, t, x, y)
  - f_1 und f_2 haben dann jeweils 1 Ausgangsvariable

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

2. Funktionstabelle

s	t	x	y	f_1
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	

s	t	x	y	f_1
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	$x \vee y$	x
2	y	$x \wedge y$
3	y	$x \wedge \overline{y}$

3. KV-Diagramm erstellen (1. Hälfte)

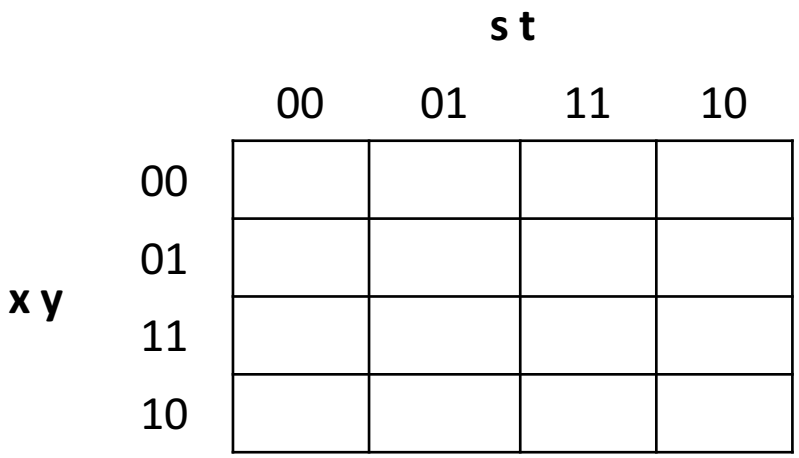
s	t	x	y	f_1
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

3. KV-Diagramm erstellen (1. Hälfte)



s	t	x	y	f_1
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

3. KV-Diagramm erstellen (2. Hälfte)

x y \ s t	00	01	11	10
00	0	0		
01	1	1		
11	1	1		
10	0	1		

s	t	x	y	f_1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

3. KV-Diagramm erstellen (2. Hälfte)

x y \ s t	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	1	0	0

s	t	x	y	f_1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

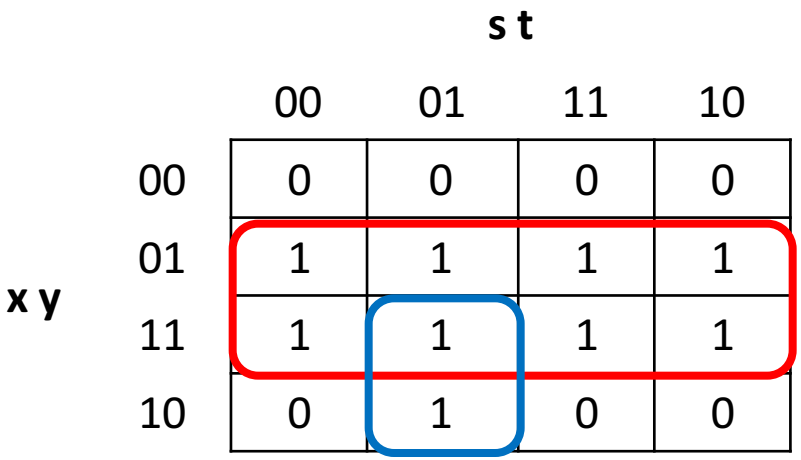
4. KV-Diagramm bearbeiten

x y \ s t	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	1	0	0

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

4. KV-Diagramm bearbeiten



7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

5. Primimplikanten aufstellen

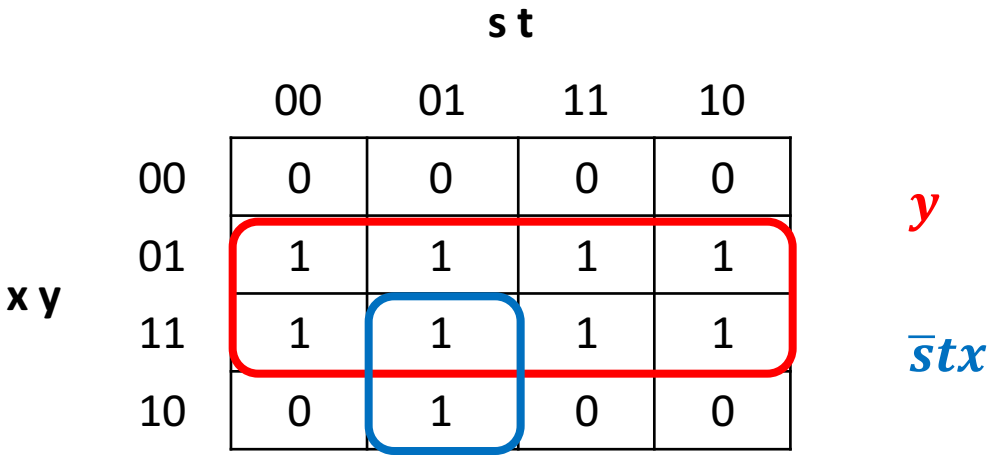
x y \ s t	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	1	0	0

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

5. Primimplikanten aufstellen



7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

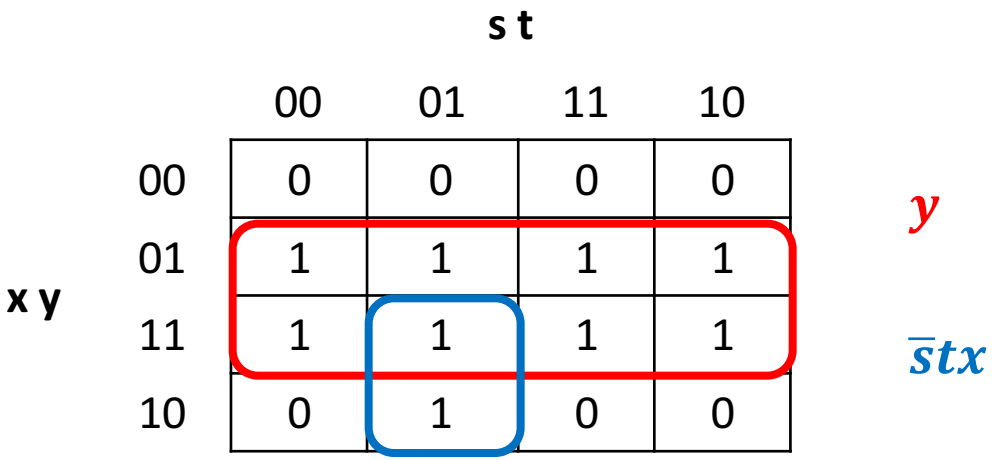
Vorgehensweise

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

6. Minimale Überdeckung finden



minimale Überdeckung

$$f_1(s,t,x,y) = y \vee \overline{s}tx$$

7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

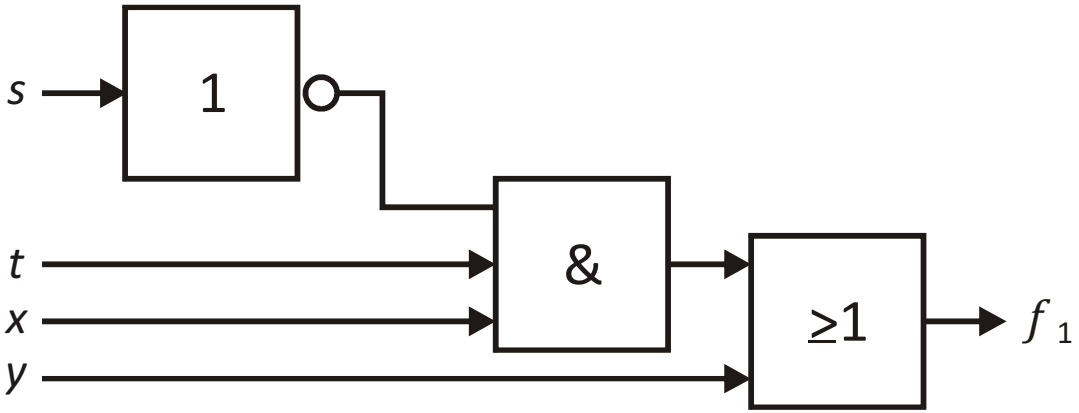
7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

7. Schaltung realisieren

$$f_1(s,t,x,y) = y \vee \bar{s}tx$$



7.5 Software PLAs

Darstellung von f_1 und f_2 als Funktionen

Typ	$f_1(x,y)$	$f_2(x,y)$
0	y	x
1	x ∨ y	x
2	y	x ∧ y
3	y	x ∧ $\overline{y}$

Vorgehensweise analog für f2

1. Nachdenken über Kodierungen
2. Funktionstabelle
3. KV-Diagramm erstellen
4. KV-Diagramm bearbeiten
5. Primimplikanten aufstellen
6. Minimale Überdeckung finden
7. Schaltung realisieren

7.5 Software PLAs

Software-PLAs

Bemerkung zum Einsatz

Beobachtung

- für ein n × m-PLA werden zur Programmierung
- $2nm$ Bits gebraucht.
- Man kann diese $2nm$ Bits gut in einem PROM speichern.

Fazit

- einfache und günstige Realisierung von booleschen Funktionen
- besonders geeignet für kleine Stückzahlen
- besonders geeignet bei nur temporärem Gebrauch

7.5 Software PLAs

Varianten von PLAs

- PAL (Programmable Array Logic)
  - PLA
  - nur UND-Array ist programmierbar
  - einmal programmierbar
- GAL (Generic Array Logic)
  - PLA
  - nur UND-Array ist programmierbar
  - löschbar (UV-Licht oder elektrisch)
- CPLD (Complex Programmable Logic Device)
  - programmierbare AND/OR-Matrix
  - programmierbare Rückkopplung
  - Ein-/Ausgabeblöcke aus Registern
- FPGA (Field Programmable Gate Array)
  - ähnlich CPLD, aber beliebige Funktion im Grundbaustein realisierbar

7. Programmierbare Bausteine

1. Einleitung ✓
2. Grundbausteine ✓
3. Realisierung von Monomen und Polynomen ✓
4. PLA als Speicher ✓
5. Software PLAs ✓