

# Rechnerstrukturen, Teil 1

**Vorlesung      4 SWS      WS 14/15**

Prof. Dr Jian-Jia Chen

Dr. Lars Hildebrand

Fakultät für Informatik – Technische Universität Dortmund

[lars.hildebrand@tu-dortmund.de](mailto:lars.hildebrand@tu-dortmund.de)

<http://ls1-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. Synchroner Schaltwerke

# 7. Programmierbare Bausteine

---

## 7. Programmierbare Bausteine

### 1. Einleitung

2. Grundbausteine

3. Realisierung von Monomen und Polynomen

4. PLA als Speicher

5. Software PLAs

# 7.1 Einleitung

---

## Realisierung von Schaltnetzen

### Gedanken zur Anwendung

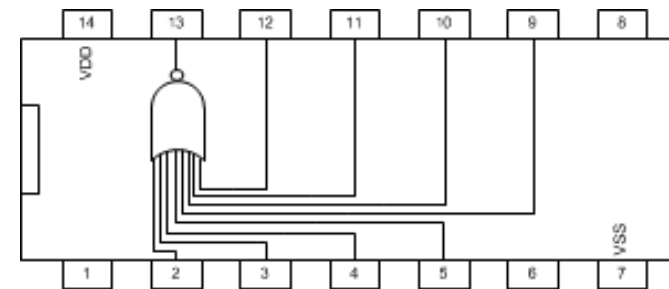
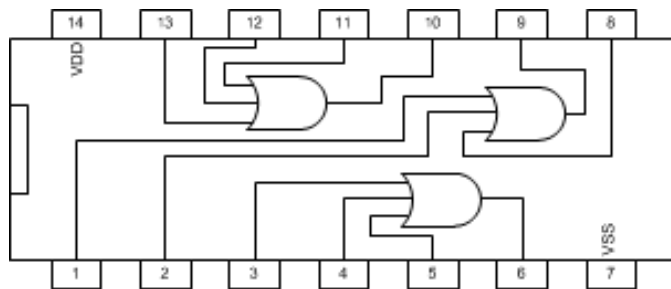
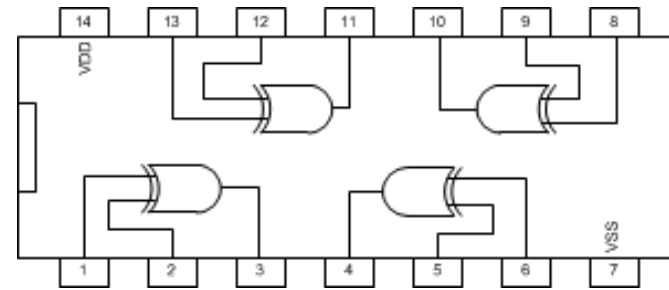
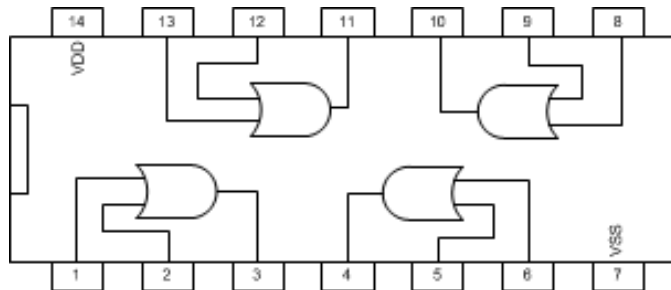
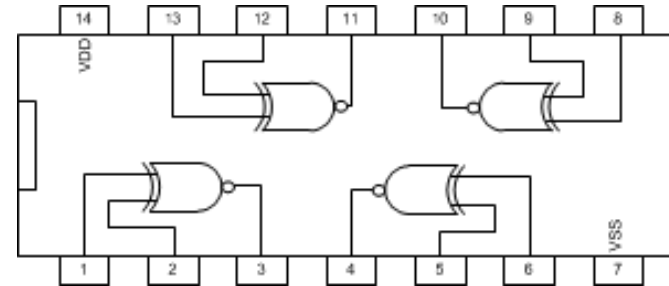
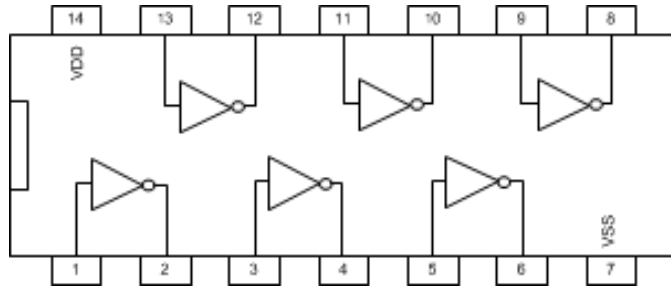
1. Problem
2. boolesche Funktion
3. Schaltnetz-Entwurf
4. Schaltnetz-Realisierung

### Realisierungen

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

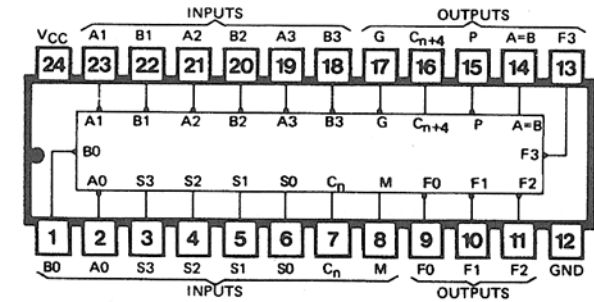
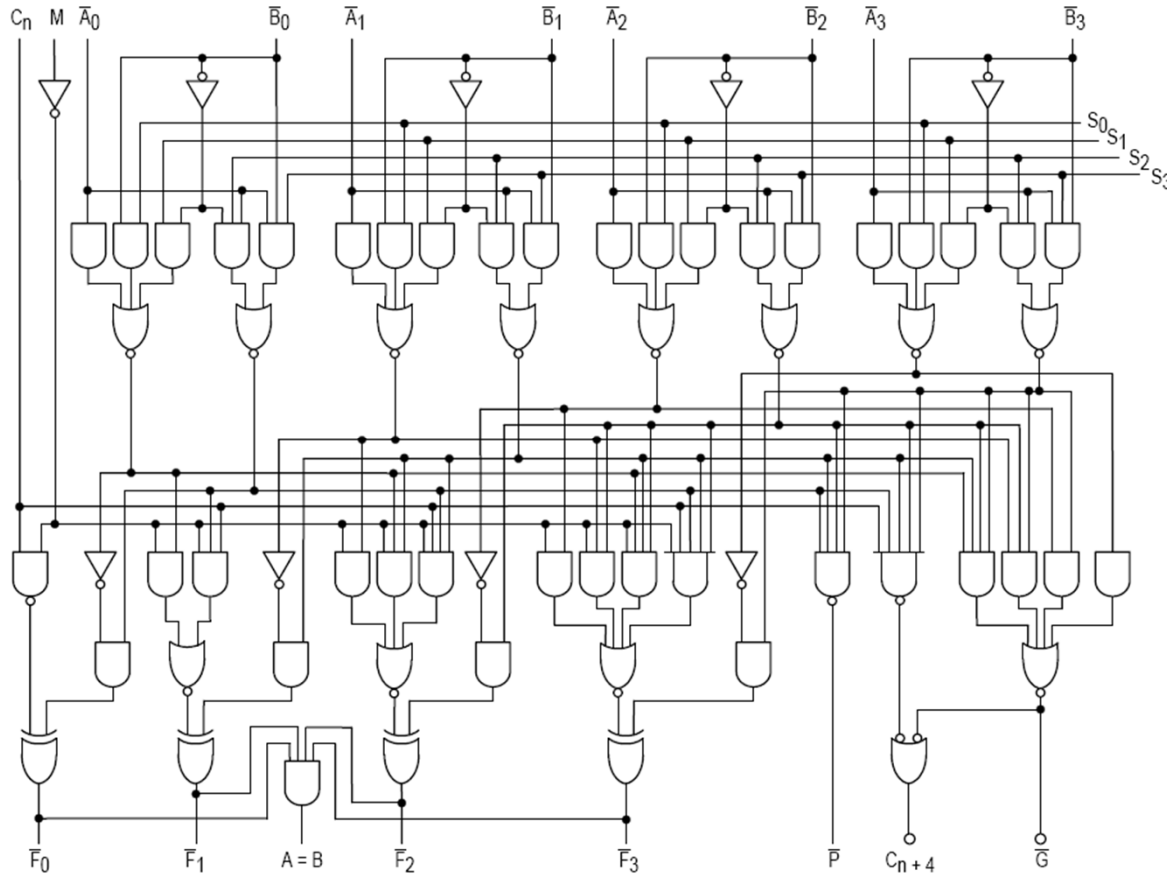
# 7.1 Einleitung

## 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	M = 0
					CIN = 1	CIN = 0
0	0	0	0	$\bar{A}$	$A$	$A + 1$
0	0	0	1	$\bar{A B}$	$A B$	$A B + 1$
0	0	1	0	$\bar{A}\bar{B}$	$A \bar{B}$	$A \bar{B} + 1$
0	0	1	1	0	-1	0
0	1	0	0	$\bar{A}\bar{B}$	$A + \bar{A}\bar{B}$	$A + \bar{A}\bar{B} + 1$
0	1	0	1	$\bar{B}$	$(A B) + \bar{A}\bar{B}$	$(A B) + \bar{A}\bar{B} + 1$
0	1	1	0	$A \oplus B$	$A - B - 1$	$A - B$
0	1	1	1	$\bar{A}\bar{B}$	$\bar{A}\bar{B} - 1$	$\bar{A}\bar{B}$
1	0	0	0	$\bar{A B}$	$A + \bar{A}\bar{B}$	$A + \bar{A}\bar{B} + 1$
1	0	0	1	$\bar{A \oplus B}$	$A + B$	$A + B + 1$
1	0	1	0	$B$	$(A \bar{B}) + \bar{A}\bar{B}$	$(A \bar{B}) + \bar{A}\bar{B} + 1$
1	0	1	1	$\bar{A}B$	$\bar{A}B - 1$	$\bar{A}B$
1	1	0	0	1	$2 * A$	$2 * A + 1$
1	1	0	1	$A \bar{B}$	$(A B) + A$	$(A B) + A + 1$
1	1	1	0	$A B$	$(A \bar{B}) + A$	$(A \bar{B}) + A + 1$
1	1	1	1	$A$	$A - 1$	$A$

# 7.1 Einleitung

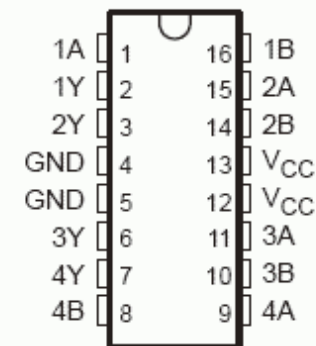
## Realisierung mit Gattern

### 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- $\mu$ m 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  $Y = A \cdot B$  or  $Y = \overline{\overline{A} + \overline{B}}$  in positive logic.

The 74AC11008 is characterized for operation from  $-40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ .

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)



# 7.1 Einleitung

## Programmable Logic Array (PLA) Datenblatt von Lattice



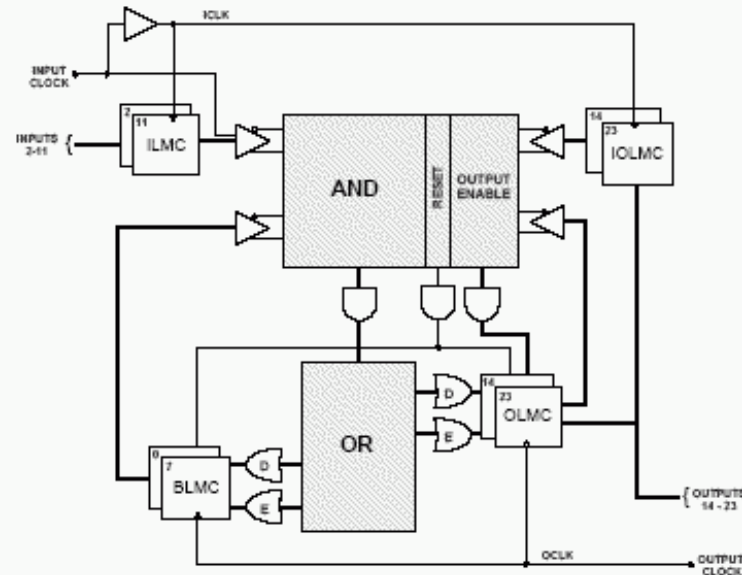
# GAL6001

High Performance E<sup>2</sup>CMOS FPLA  
Generic Array Logic™

### Features

- HIGH PERFORMANCE E<sup>2</sup>CMOS® 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 I<sub>cc</sub>
- E<sup>2</sup> 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

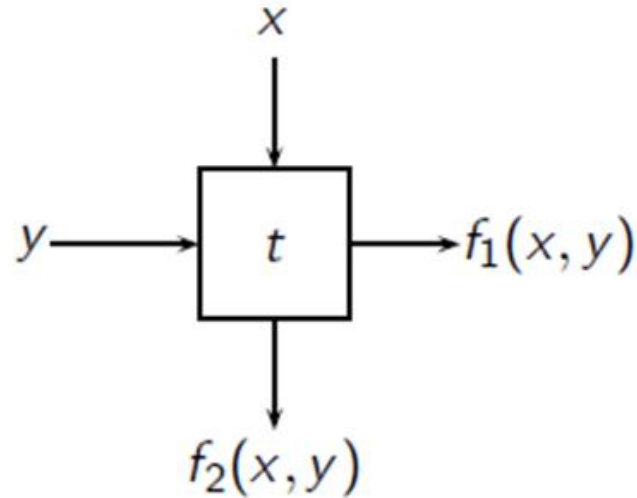
---

## 7. Programmierbare Bausteine

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

# 7.2 Grundbausteine

## 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	$y$	$x \wedge \bar{y}$

## Funktional vollstandig?

# 7.2 Grundbausteine

## PLA Grundbausteine

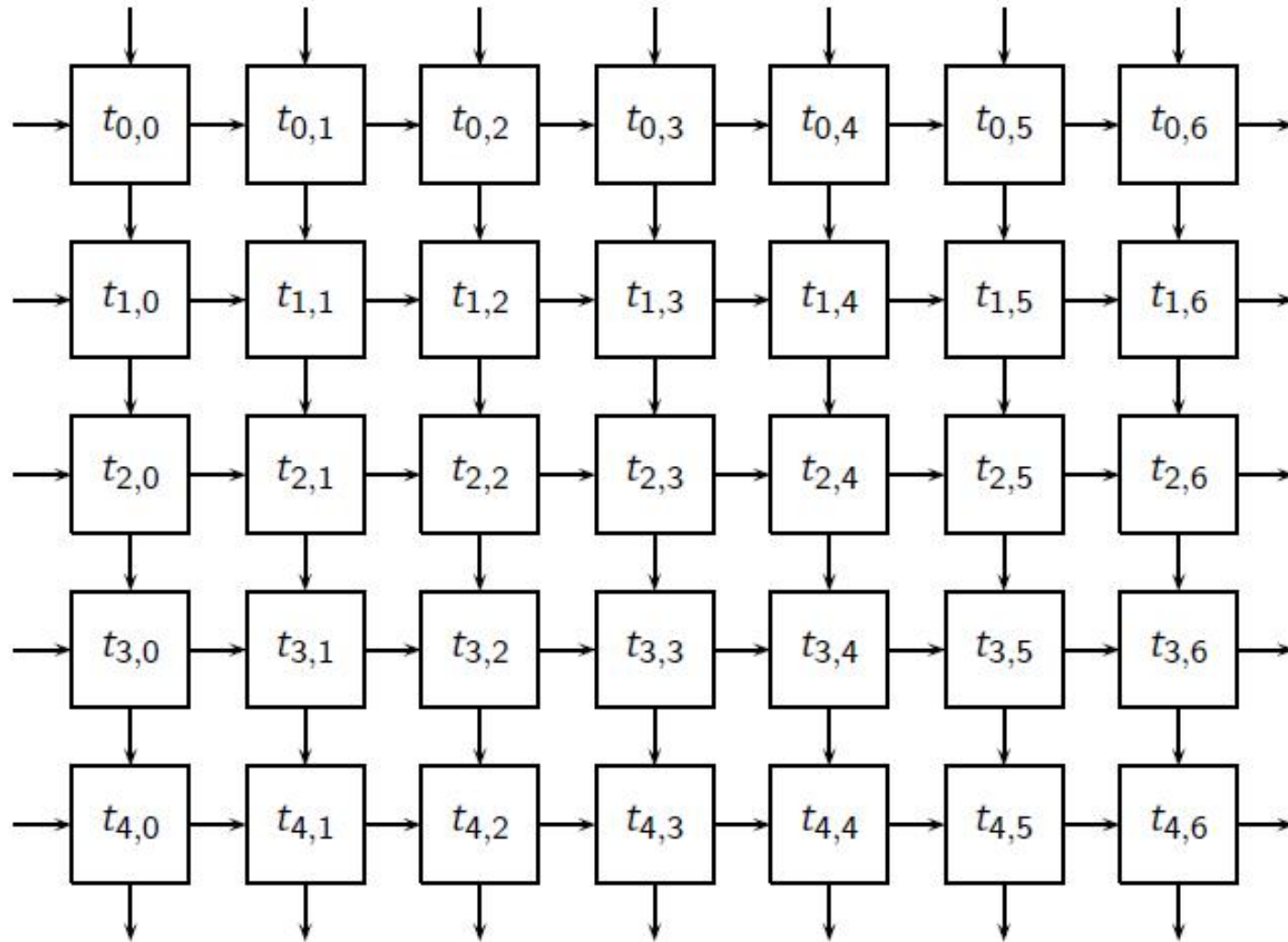
Name	Typ	$f_1(x, y)$	$f_2(x, y)$
Identer	0	$y$	$x$
Addierer	1	$x \vee y$	$x$
Multiplizierer	2	$y$	$x \wedge y$
Negat-Multiplizierer	3	$y$	$x \wedge \bar{y}$

### 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,  $f_2(1, y) = 1 \wedge \bar{y} = \bar{y}$

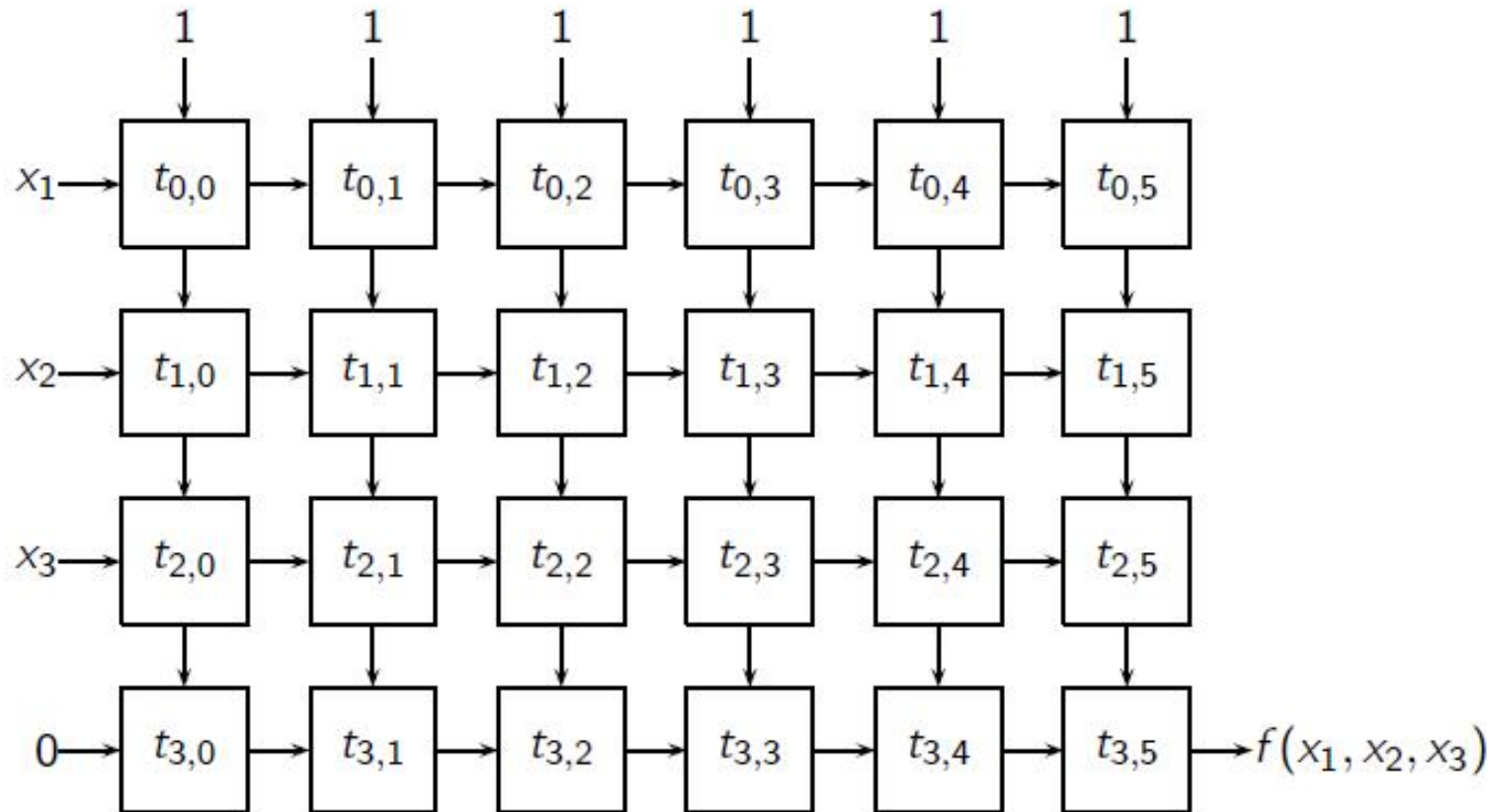
# 7.2 Grundbausteine

## Aufbau eines PLA



## 7.2 Grundbausteine

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



Wie wählt man die Bausteintypen?

# 7. Programmierbare Bausteine

---

## 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

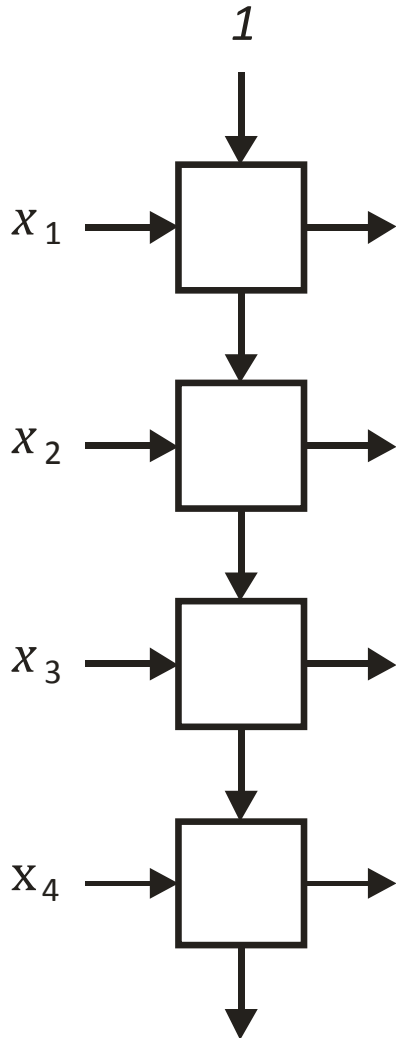
- Wie realisieren wir Monome?
- exemplarisch am Beispiel  $x_1\bar{x}_2x_4$



# 7.3 Realisierung von Monomen und Polynomen

## Realisierung von Monomen

**Beispiel:** Monom  $x_1 \bar{x}_2 x_4$

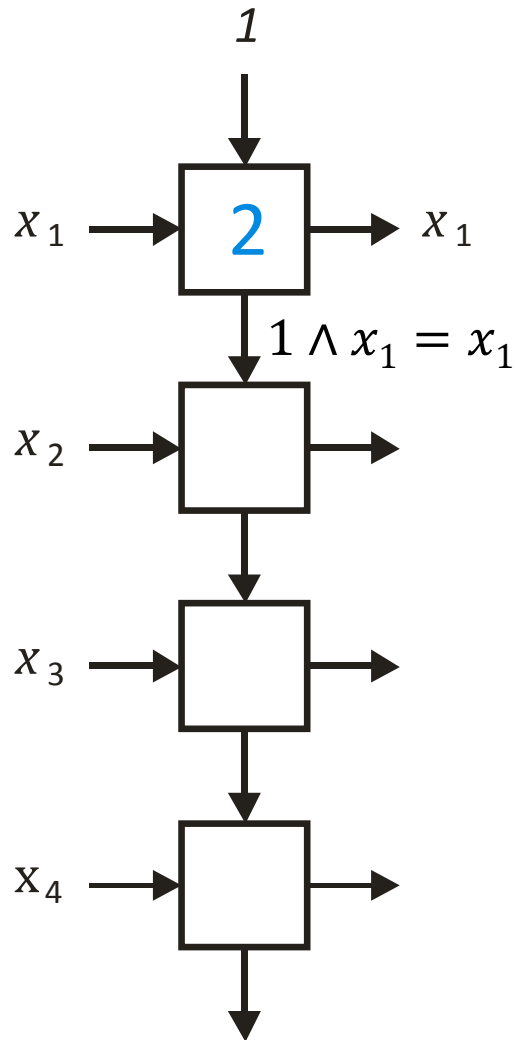


Name	Typ	$f_r(o, l)$	$f_u(o, l)$
Identer	0	$l$	$o$
Addierer	1	$l \vee o$	$o$
Multiplizierer	2	$l$	$o \wedge l$
Negat-Multiplizierer	3	$l$	$o \wedge \bar{l}$

# 7.3 Realisierung von Monomen und Polynomen

## Realisierung von Monomen

**Beispiel:** Monom  $x_1 \bar{x}_2 x_4$

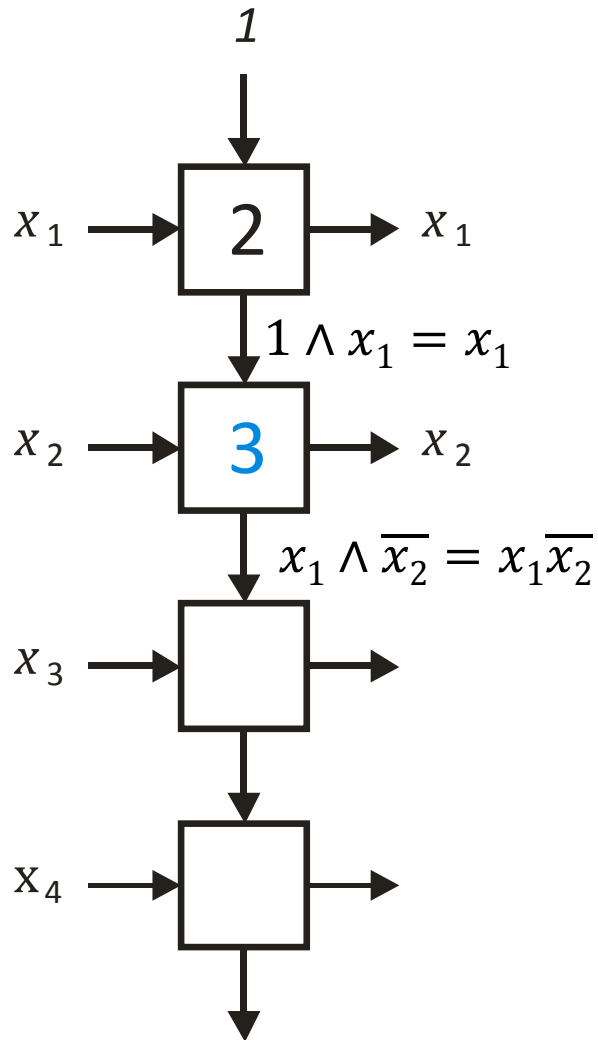


Name	Typ	$f_r(o, l)$	$f_u(o, l)$
Identer	0	$l$	$o$
Addierer	1	$l \vee o$	$o$
<b>Multiplizierer</b>	<b>2</b>	<b><math>l</math></b>	<b><math>o \wedge l</math></b>
Negat-Multiplizierer	3	$l$	$o \wedge \bar{l}$

# 7.3 Realisierung von Monomen und Polynomen

## Realisierung von Monomen

**Beispiel:** Monom  $x_1 \bar{x}_2 x_4$

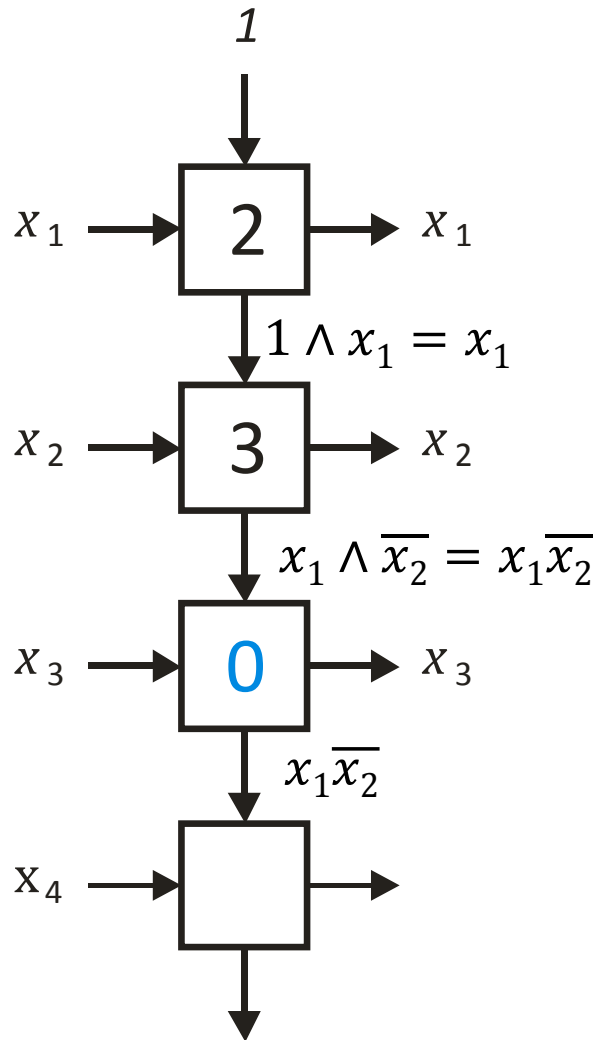


Name	Typ	$f_r(o, l)$	$f_u(o, l)$
Identer	0	$l$	$o$
Addierer	1	$l \vee o$	$o$
Multiplizierer	2	$l$	$o \wedge l$
<b>Negat-Multiplizierer</b>	<b>3</b>	<b><math>l</math></b>	<b><math>o \wedge \bar{l}</math></b>

# 7.3 Realisierung von Monomen und Polynomen

## Realisierung von Monomen

**Beispiel:** Monom  $x_1 \bar{x}_2 x_4$

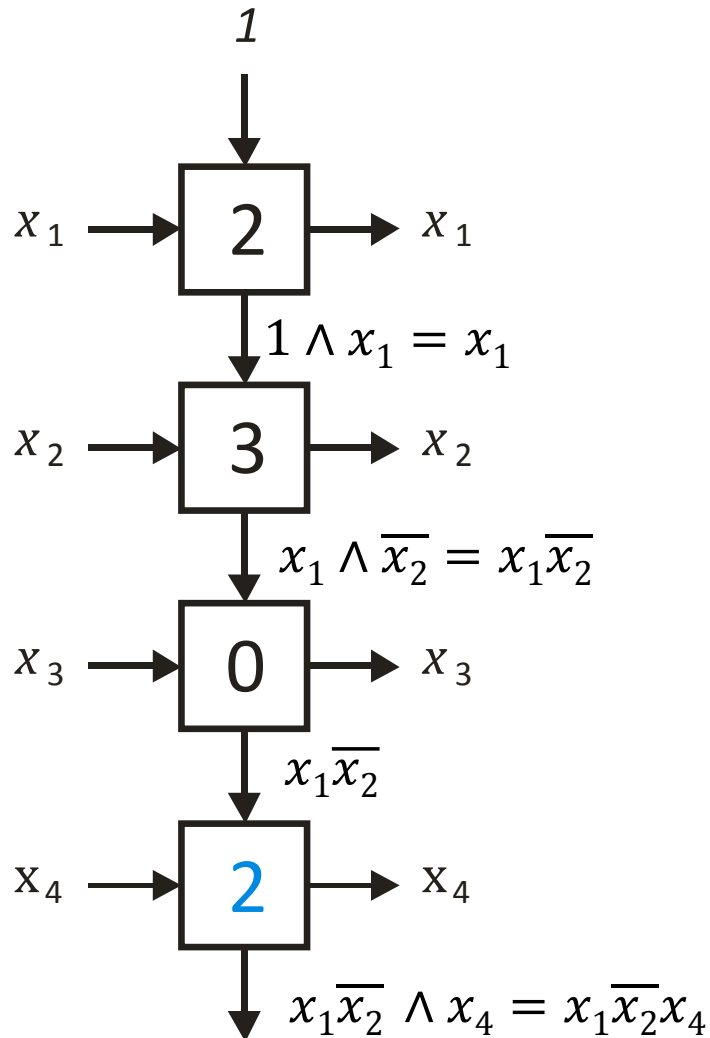


Name	Typ	$f_r(o, l)$	$f_u(o, l)$
<b>Identer</b>	<b>0</b>	<b><math>l</math></b>	<b><math>o</math></b>
Addierer	1	$l \vee o$	$o$
Multiplizierer	2	$l$	$o \wedge l$
Negat-Multiplizierer	3	$l$	$o \wedge \bar{l}$

# 7.3 Realisierung von Monomen und Polynomen

## Realisierung von Monomen

**Beispiel:** Monom  $x_1 \bar{x}_2 x_4$

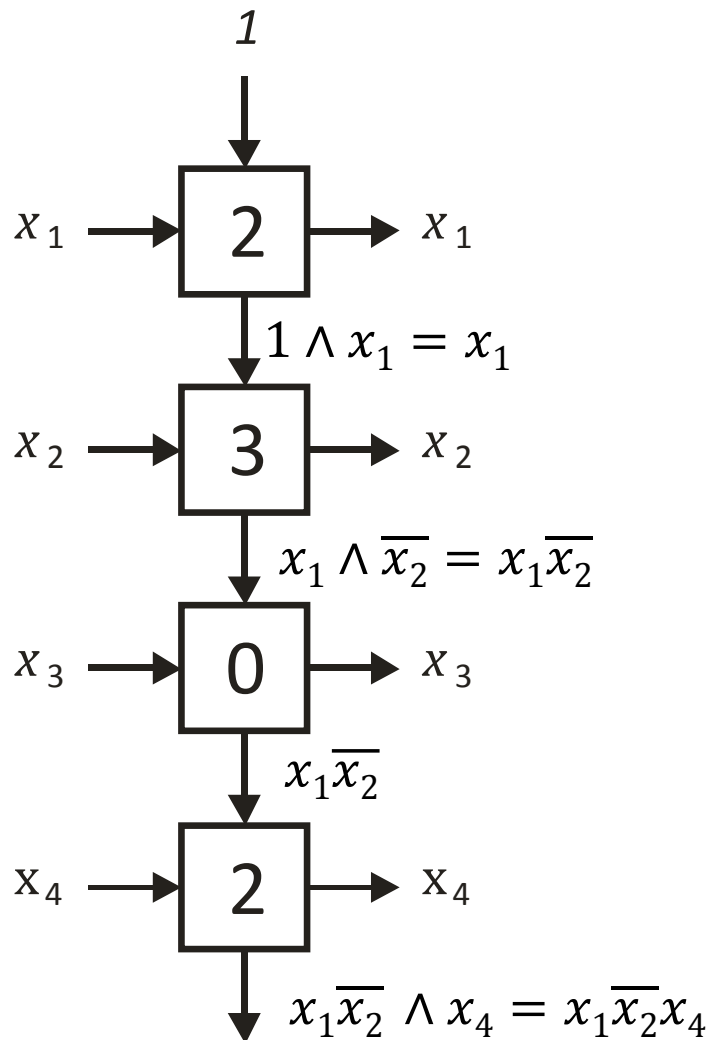


Name	Typ	$f_r(o, l)$	$f_u(o, l)$
Identer	0	$l$	$o$
Addierer	1	$l \vee o$	$o$
<b>Multiplizierer</b>	<b>2</b>	<b><math>l</math></b>	<b><math>o \wedge l</math></b>
Negat-Multiplizierer	3	$l$	$o \wedge \bar{l}$

# 7.3 Realisierung von Monomen und Polynomen

## Realisierung von Monomen

**Beispiel:** Monom  $x_1 \bar{x}_2 x_4$



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

### Monomrealisierung

- falls Variable fehlt Typ 0
- falls  $x_i$  vorkommt Typ 2
- falls  $\bar{x}_i$  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**  $m_1, m_2, \dots, m_k$
- in  $n$  Zeilen und  $k$  Spalten realisierbar

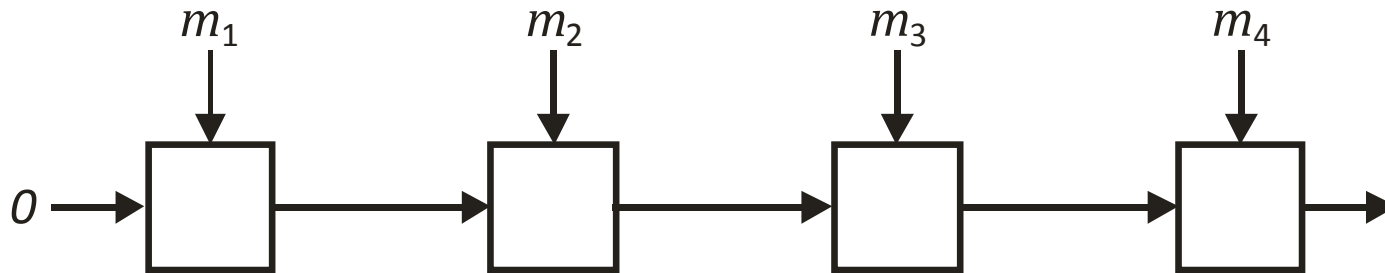
Wie können wir  $f$  realisieren, z.B.  $f = m_1 \vee m_2 \vee m_4$ ?

# 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	$l$	$o \wedge \bar{l}$



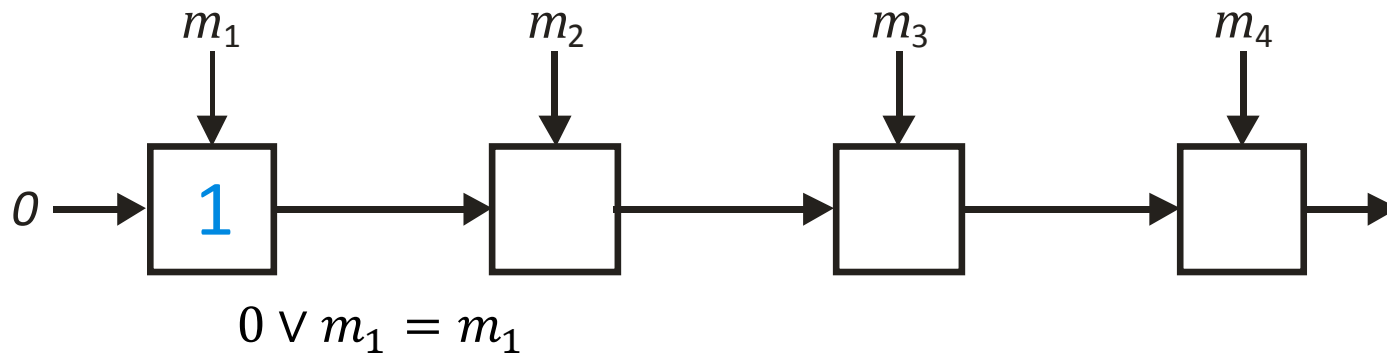


# 7.3 Realisierung von Monomen und Polynomen

## 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	$l$	$o$
<b>Addierer</b>	<b>1</b>	<b><math>l \vee o</math></b>	<b><math>o</math></b>
Multiplizierer	2	$l$	$o \wedge l$
Negat-Multiplizierer	3	$l$	$o \wedge \bar{l}$

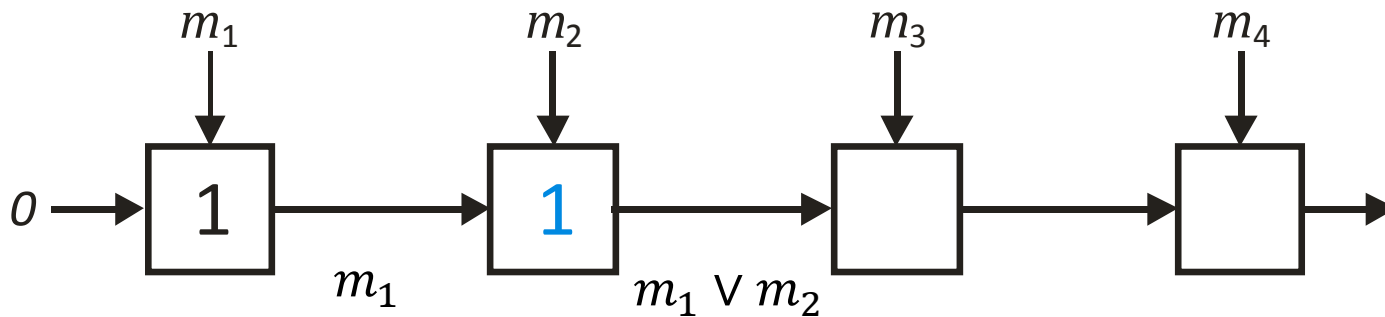


# 7.3 Realisierung von Monomen und Polynomen

## 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	$l$	$o$
<b>Addierer</b>	<b>1</b>	<b><math>l \vee o</math></b>	<b><math>o</math></b>
Multiplizierer	2	$l$	$o \wedge l$
Negat-Multiplizierer	3	$l$	$o \wedge \bar{l}$

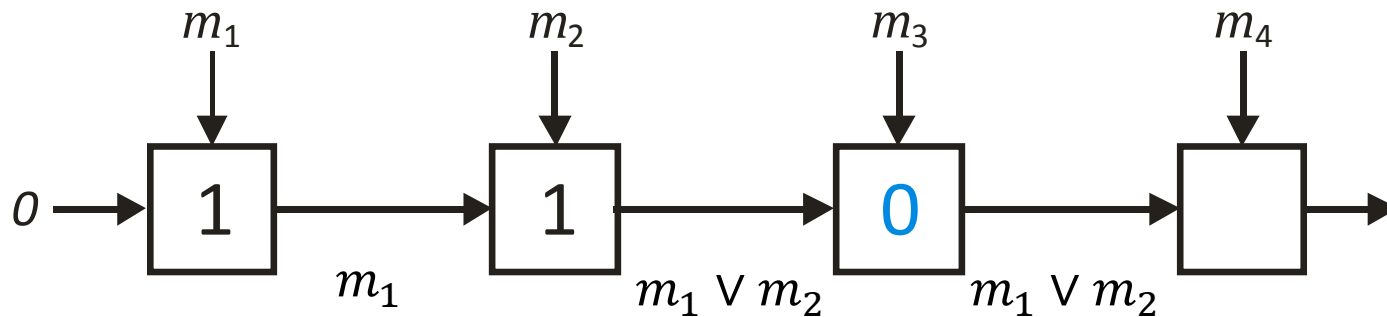


# 7.3 Realisierung von Monomen und Polynomen

## 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	$l$	$o \wedge \bar{l}$



$m_1$

$m_1 \vee m_2$

$m_1 \vee m_2$

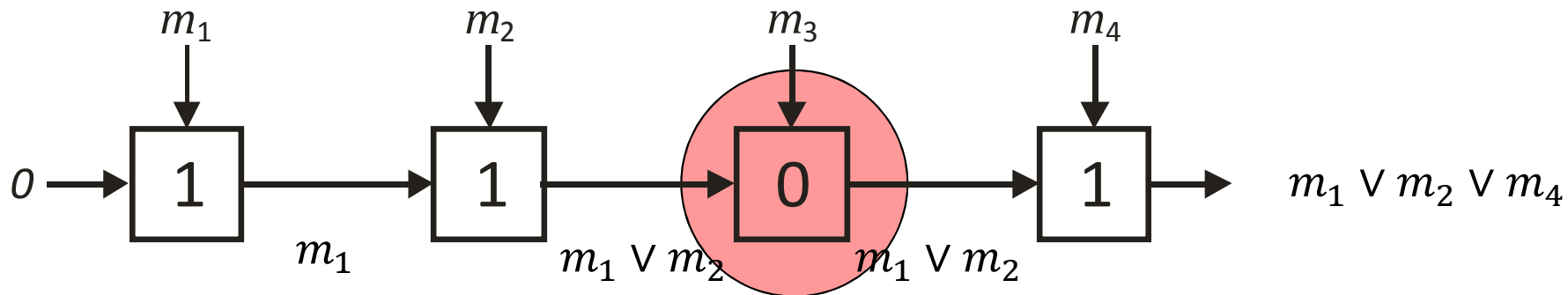
$m_1 \vee m_2 \vee m_4$

# 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	$l$	$o \wedge \bar{l}$



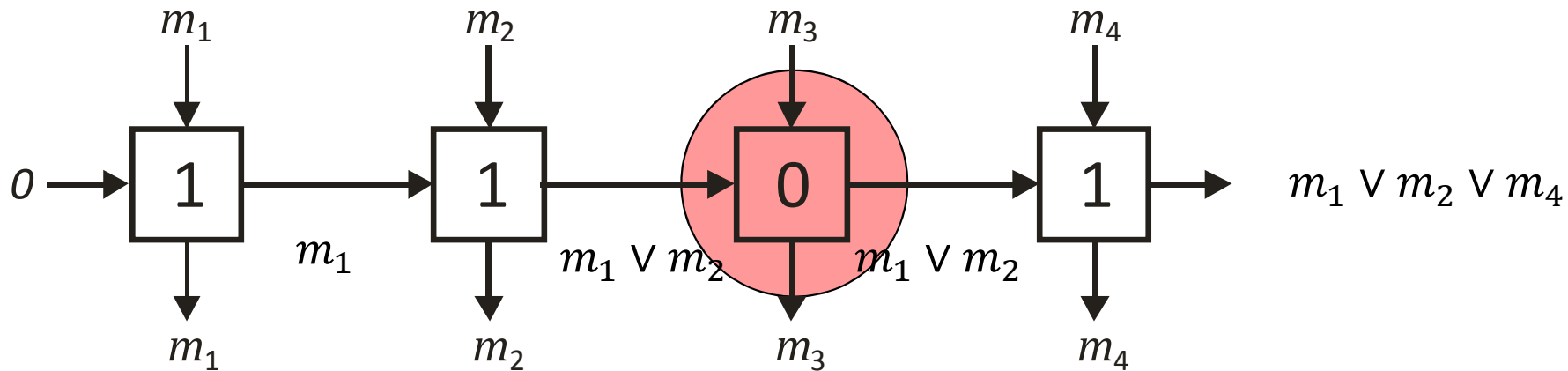
**Warum ein Grundbaustein für das Weglassen eines Monoms?**

# 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	$l$	$o \wedge \bar{l}$

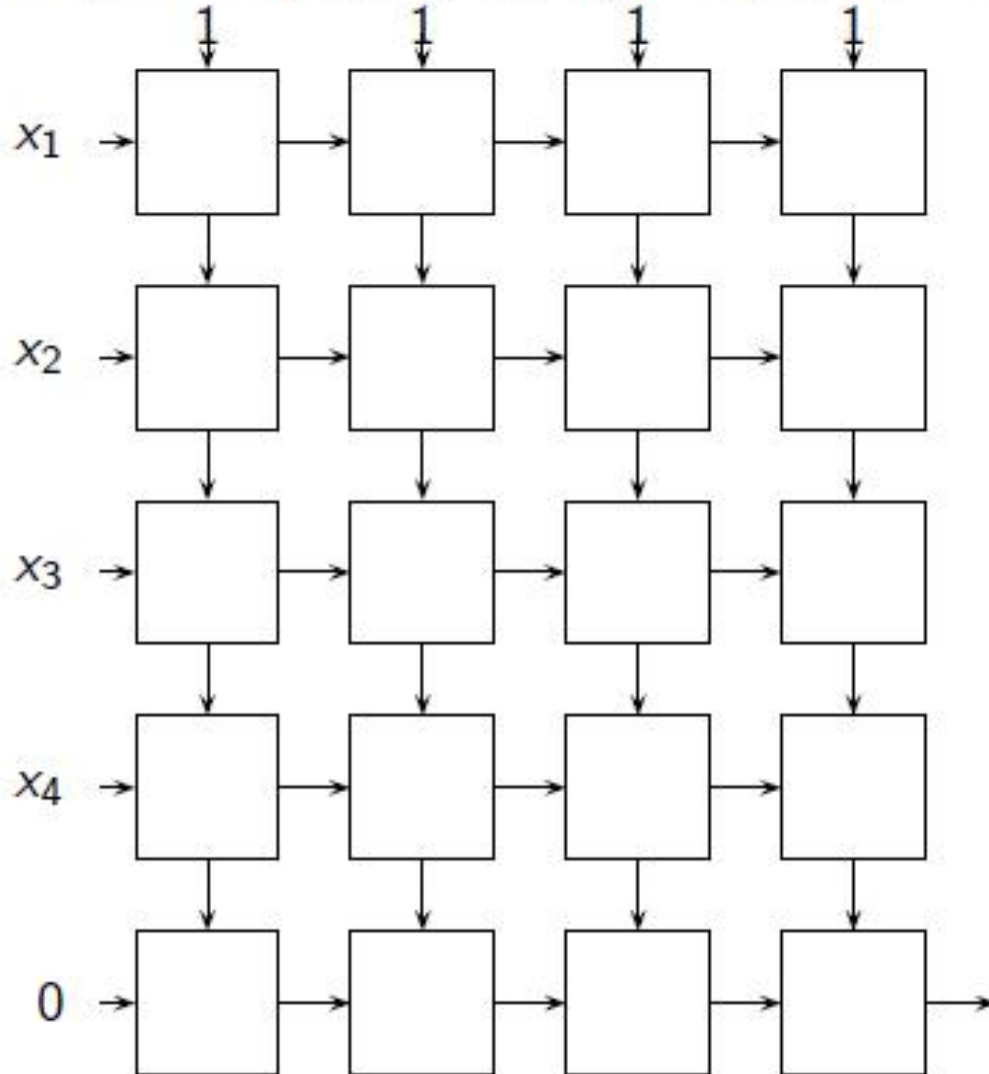


**Warum ein Grundbaustein für das Weglassen eines Monoms?**

# 7.3 Realisierung von Monomen und Polynomen

## PLA: ein konkretes Beispiel

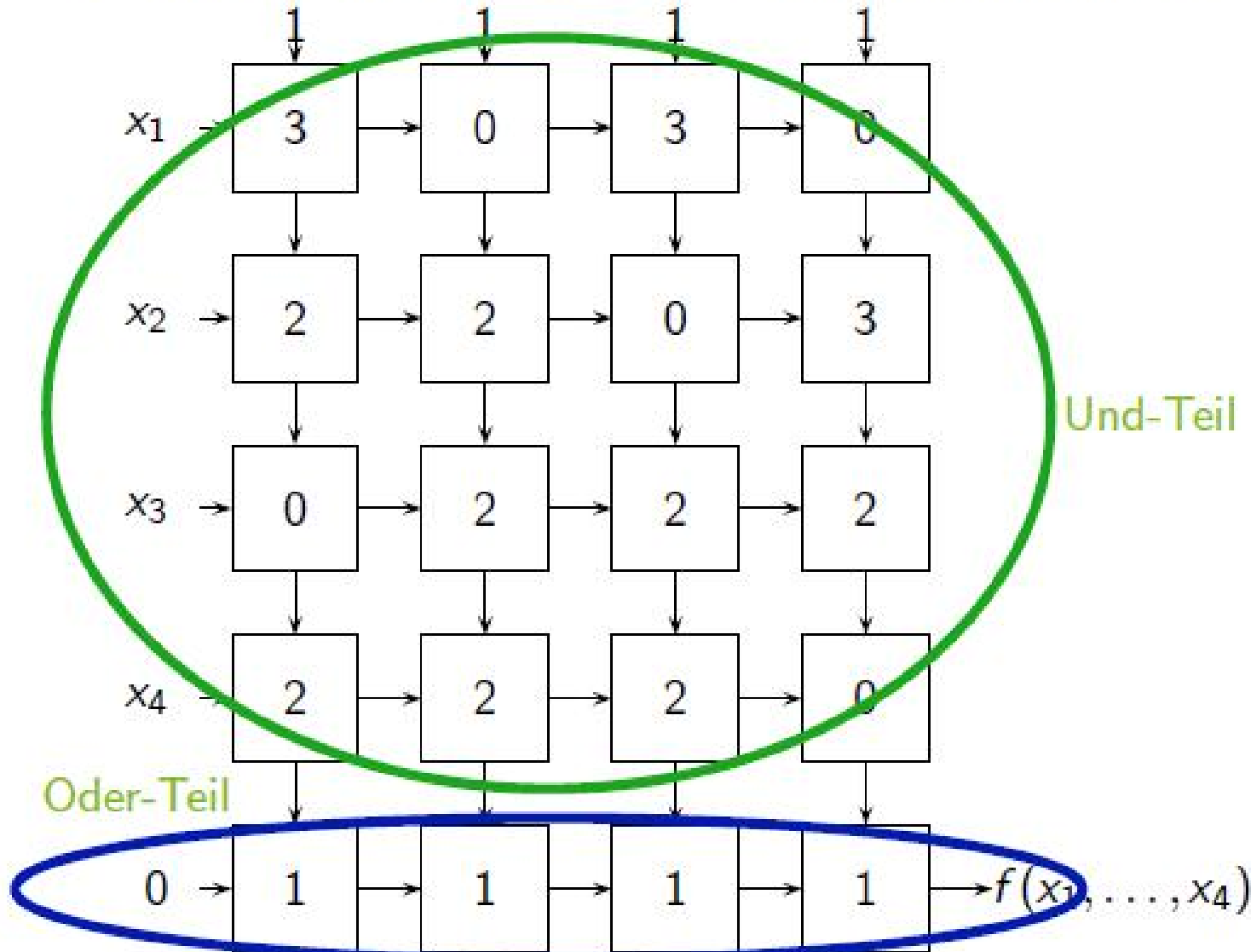
Beispiel  $f(x_1, x_2, x_3, x_4) = \overline{x_1} x_2 x_4 \vee x_2 x_3 x_4 \vee \overline{x_1} x_3 x_4 \vee \overline{x_2} x_3$



# 7.3 Realisierung von Monomen und Polynomen

## PLA: ein konkretes Beispiel

Beispiel  $f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 x_4 \vee x_2 x_3 x_4 \vee \bar{x}_1 x_3 x_4 \vee \bar{x}_2 x_3$





# 7.3 Realisierung von Monomen und Polynomen

---

## Realisierung von Polynomen

### Beobachtung 1

- für  $f : \{0, 1\}^n \rightarrow \{0, 1\}$
- $k$  verschiedene **Monome**  $m_1, m_2, \dots, m_k$
- in  $n$  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

---

## 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  $m$ .

$$W_0 = W_{0,0}W_{0,1}W_{0,2} \cdots W_{0,m-1} \in \{0, 1\}^m$$

$$W_1 = W_{1,0}W_{1,1}W_{1,2} \cdots W_{1,m-1} \in \{0, 1\}^m$$

$$W_2 = W_{2,0}W_{2,1}W_{2,2} \cdots W_{2,m-1} \in \{0, 1\}^m$$

$\vdots \quad \vdots \quad \vdots$

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

**Benutze**

**PLA** mit  $n + m$  Zeilen,  $2^n$  Spalten

**Beobachtung**

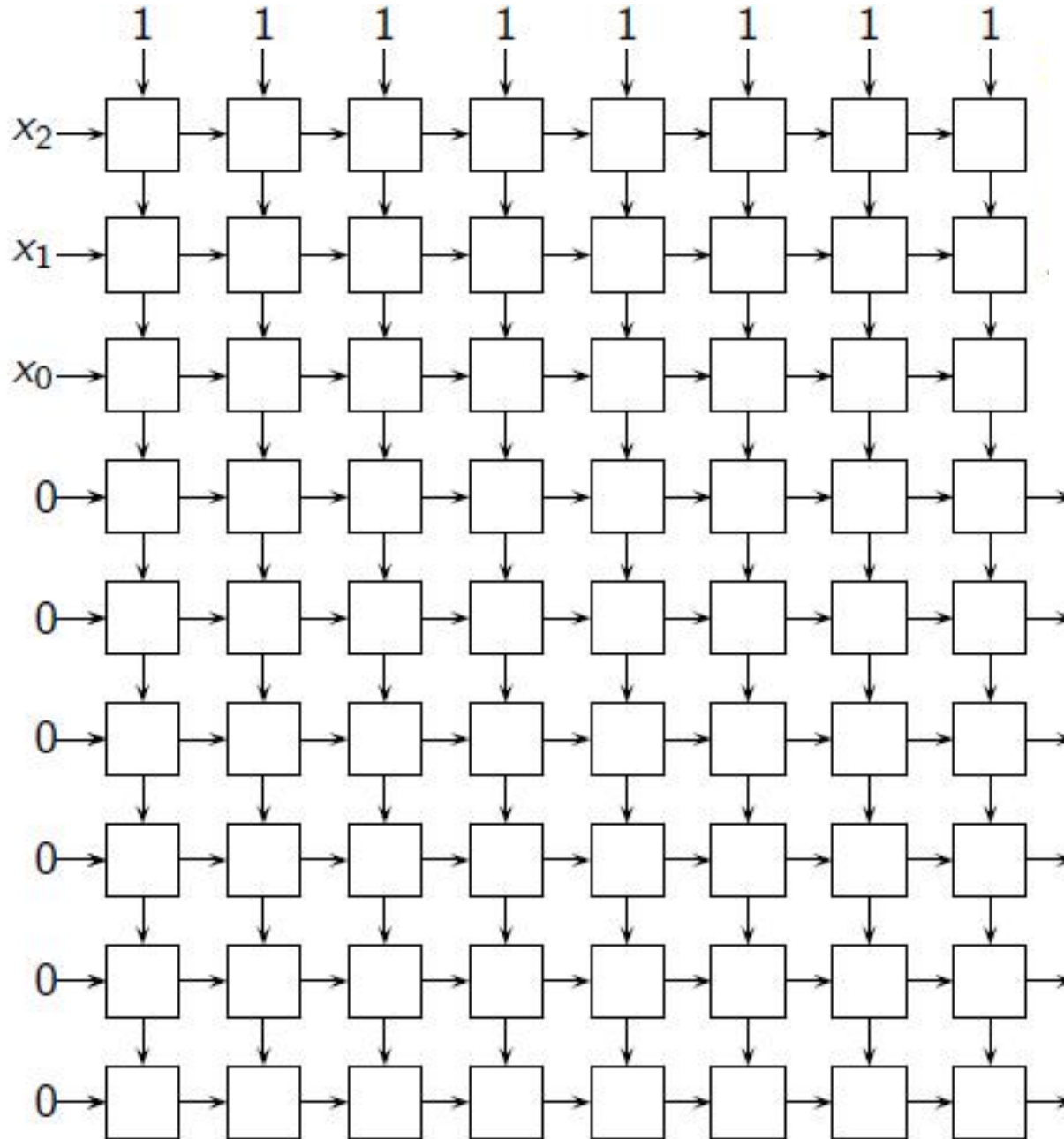
$m \cdot 2^n$  Zellen für  $m \cdot 2^n$  zu speichernde Bits

mindestens erforderlich

**Adressierung**

mit jeweils  $n$  Bits in  $n$  zusätzlichen Zeilen

# 7.4 PLA als Speicher



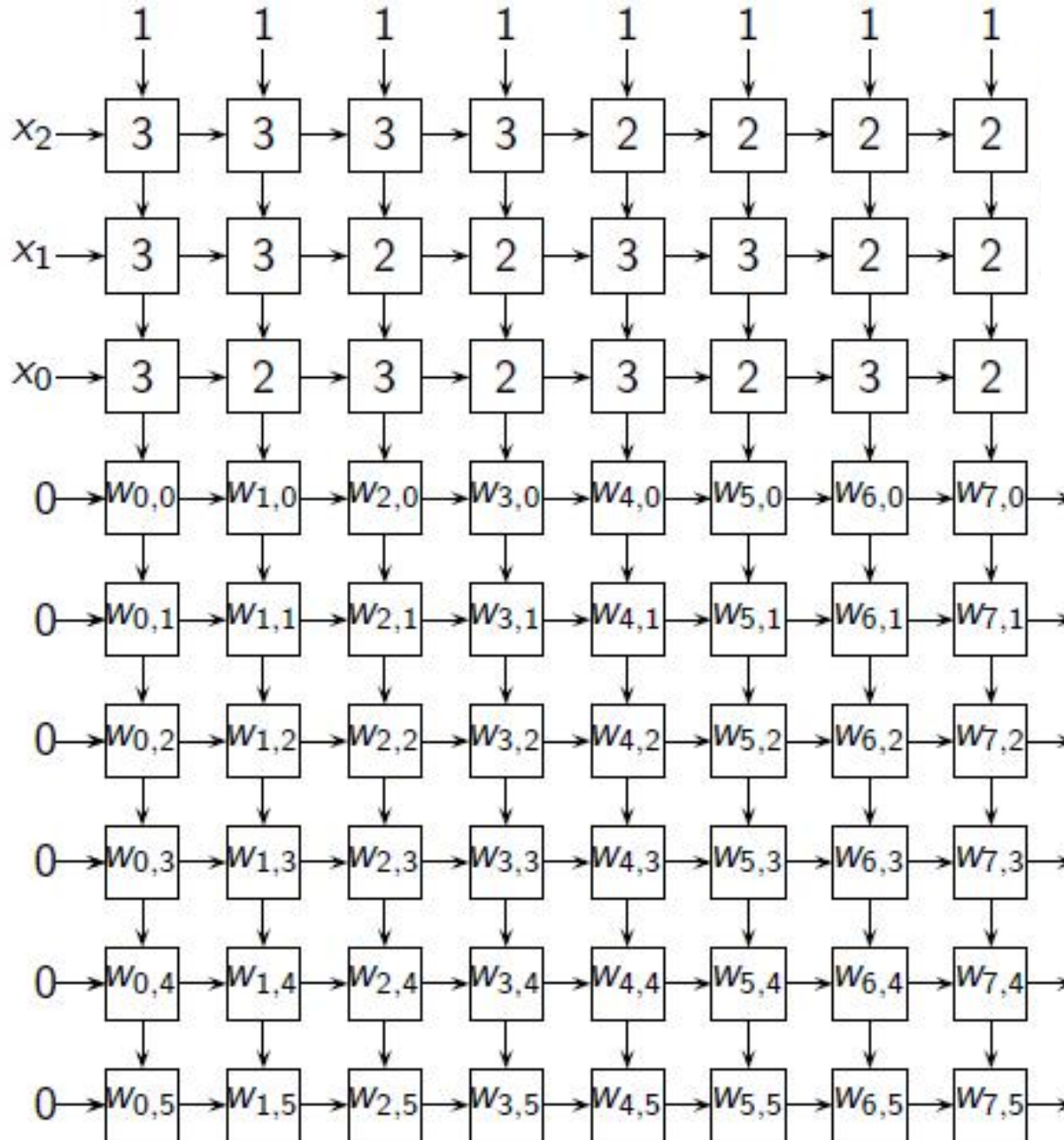
## Beispiel

$n = 3, m = 6$

$2^3 = 8$  Wörter

jeweils 6 Bits

# 7.4 PLA als Speicher



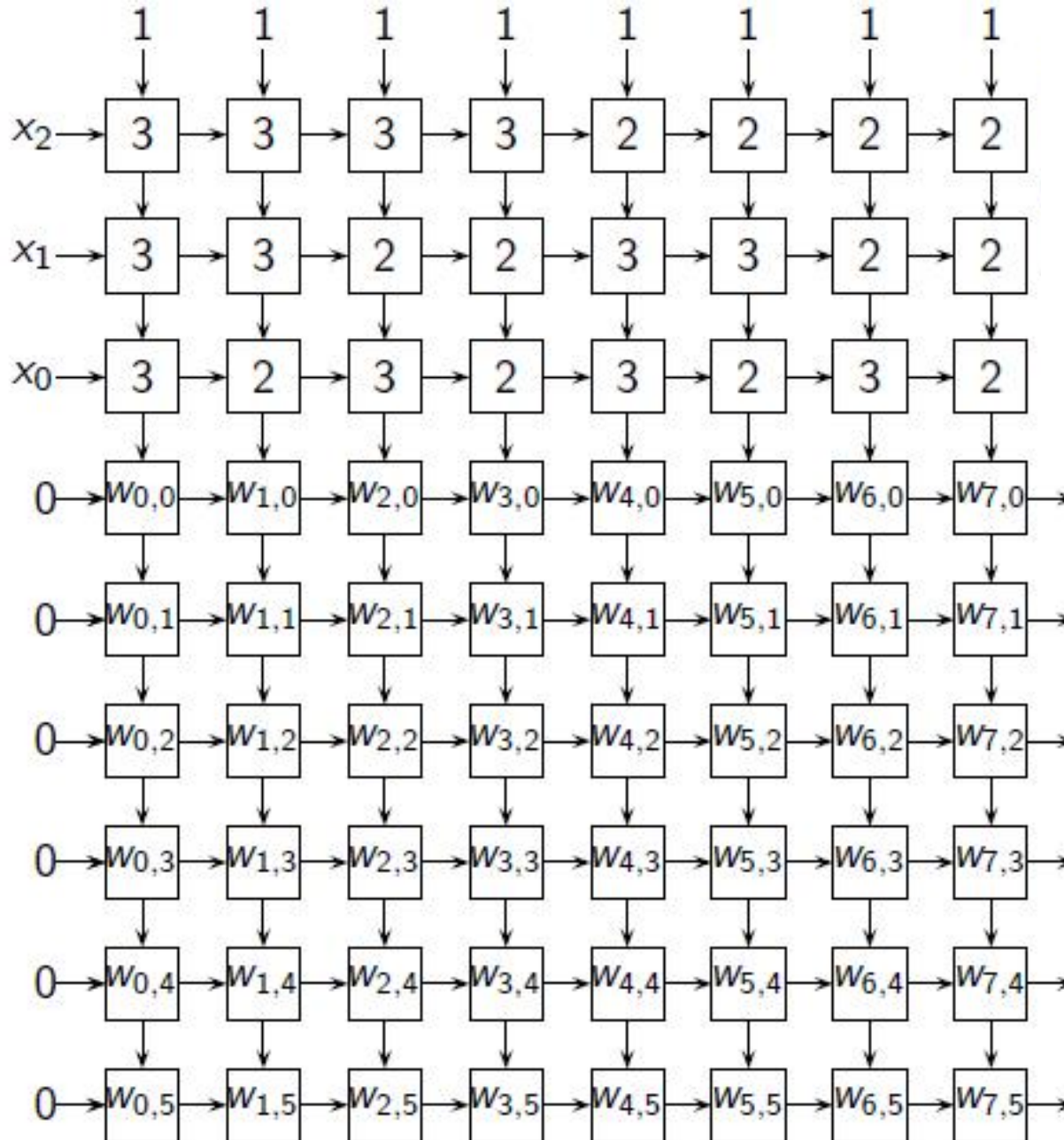
## Beispiel

$n = 3, m = 6$

$2^3 = 8$  Wörter

jeweils 6 Bits

# 7.4 PLA als Speicher



## Beispiel

$n = 3, m = 6$

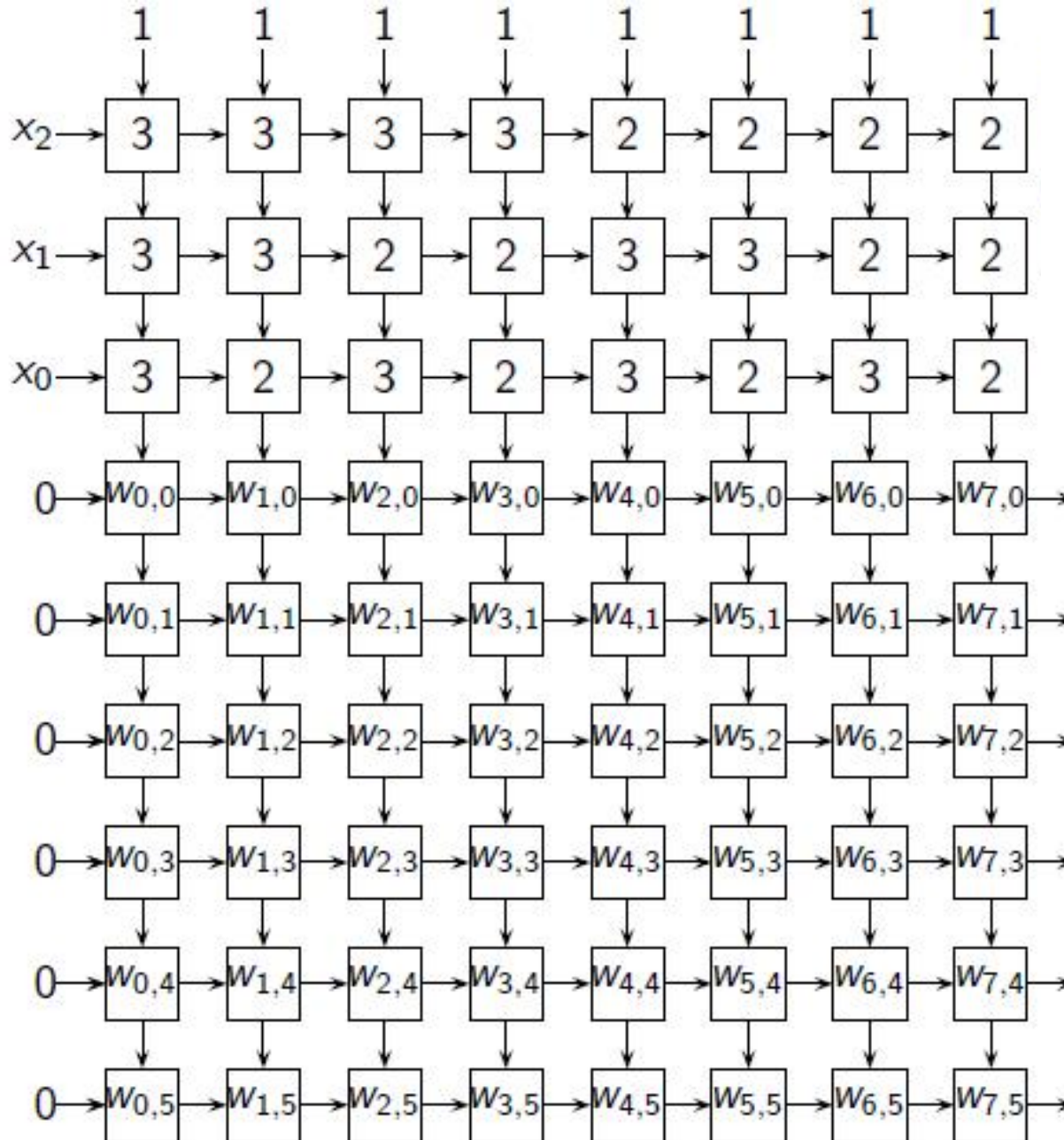
$2^3 = 8$  Wörter

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  $w_5$

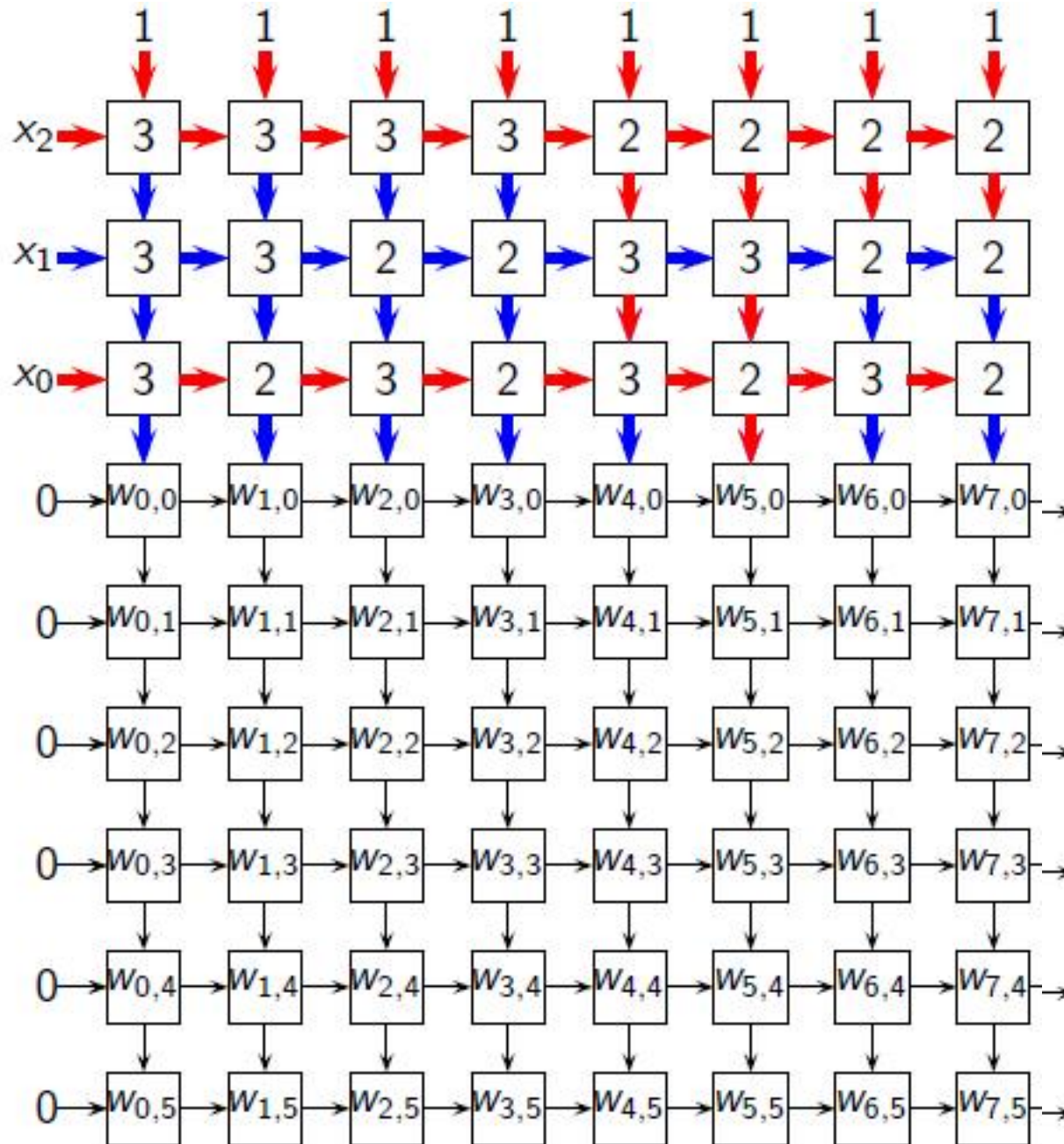
$5 = (101)_2$

$x_2 = 1$

$x_1 = 0$

$x_0 = 1$

# 7.4 PLA als Speicher



## Beispiel

$$n = 3, m = 6$$

$$2^3 = 8 \text{ Wörter}$$

jeweils 6 Bits

## Beispiel

lies  $w_5$

$$5 = (101)_2$$

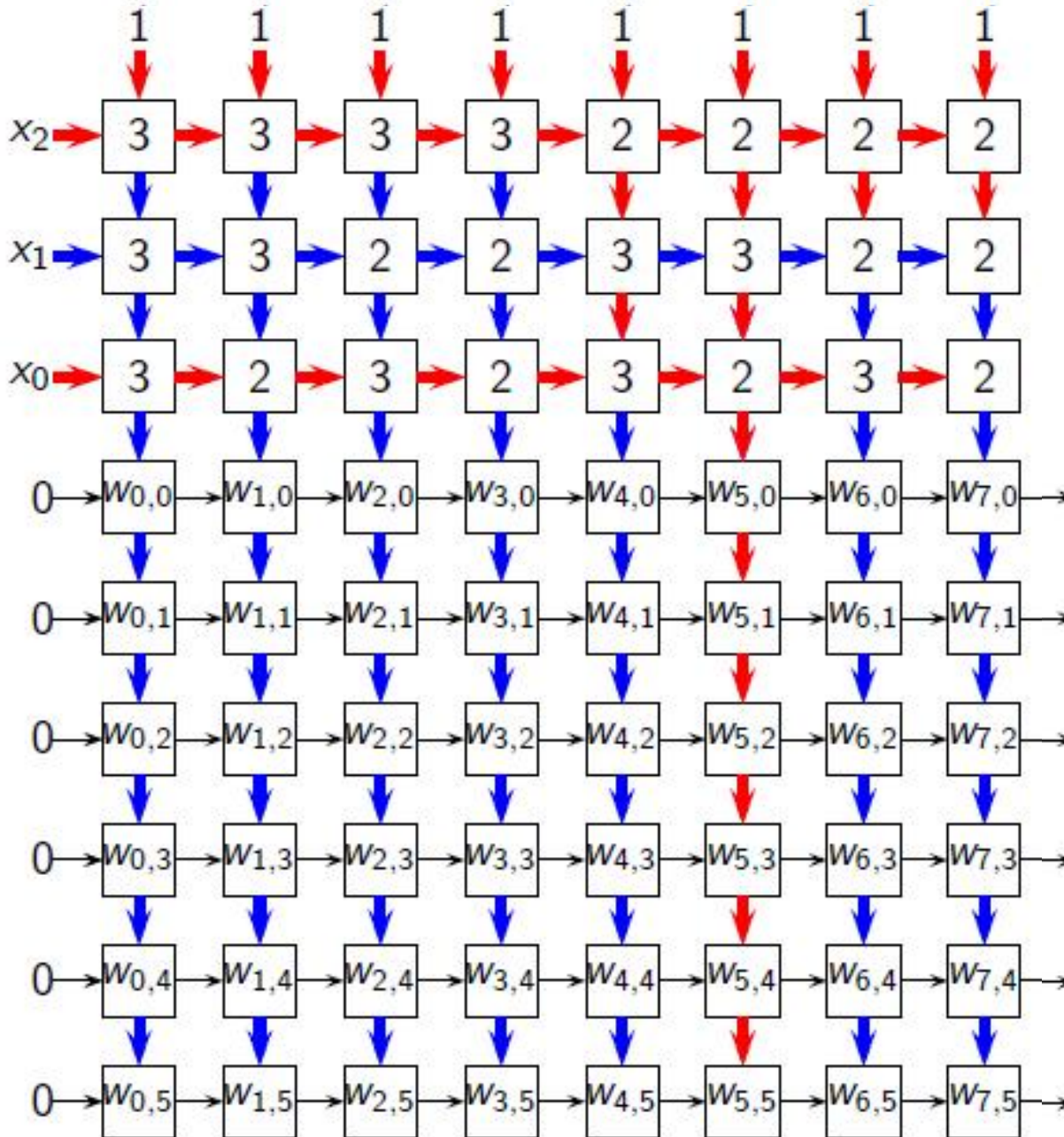
$$x_2 = 1$$

$$x_1 = 0$$

$$x_0 = 1$$



# 7.4 PLA als Speicher



## Beispiel

$n = 3, m = 6$

$2^3 = 8$  Wörter

jeweils 6 Bits

## Beispiel

lies  $w_5$

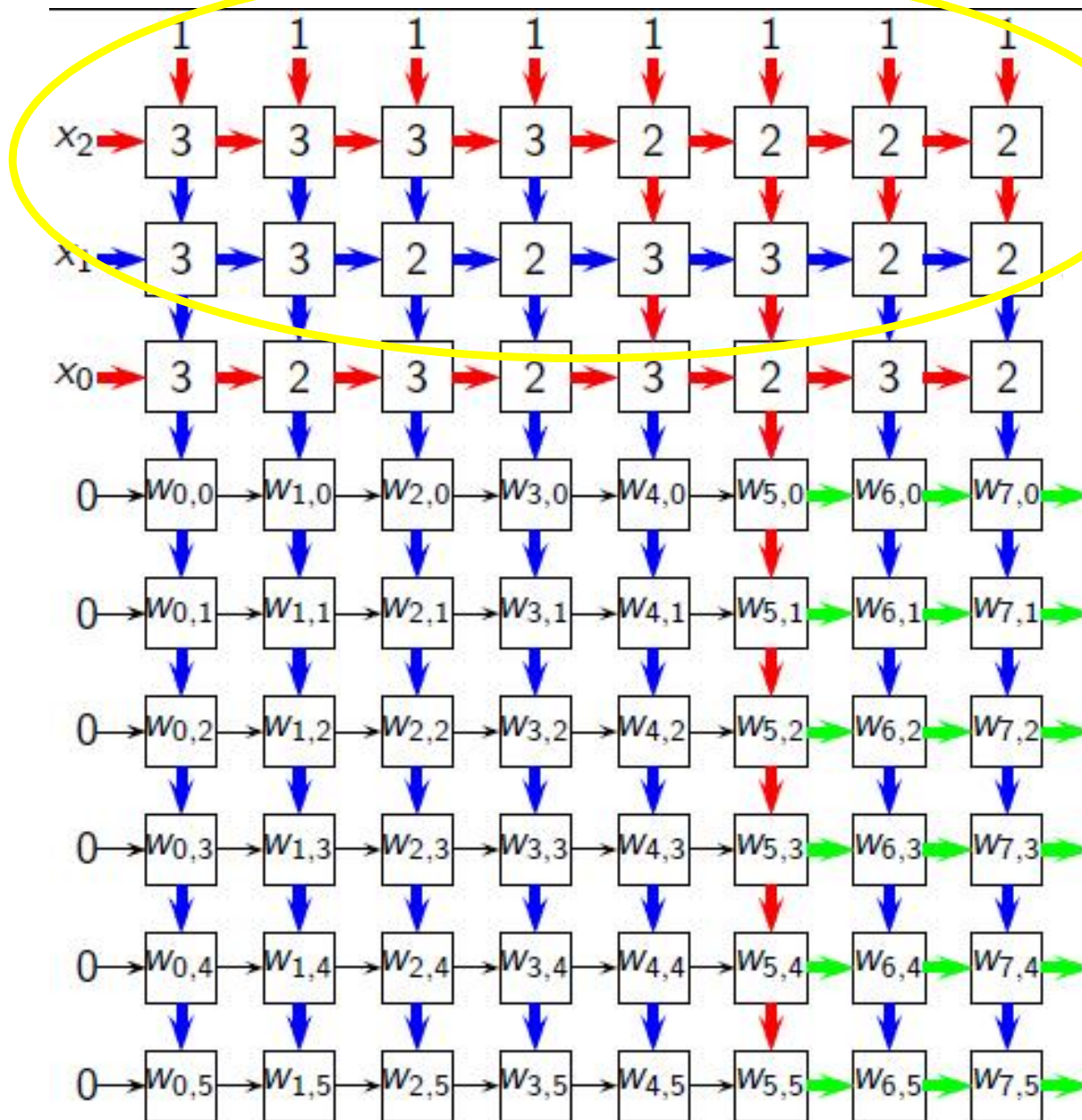
$5 = (101)_2$

$x_2 = 1$

$x_1 = 0$

$x_0 = 1$

# 7.4 PLA als Speicher



## Beispiel

$$n = 3, m = 6$$

$$2^3 = 8 \text{ Wörter}$$

jeweils 6 Bits

## Beispiel

lies  $w_5$

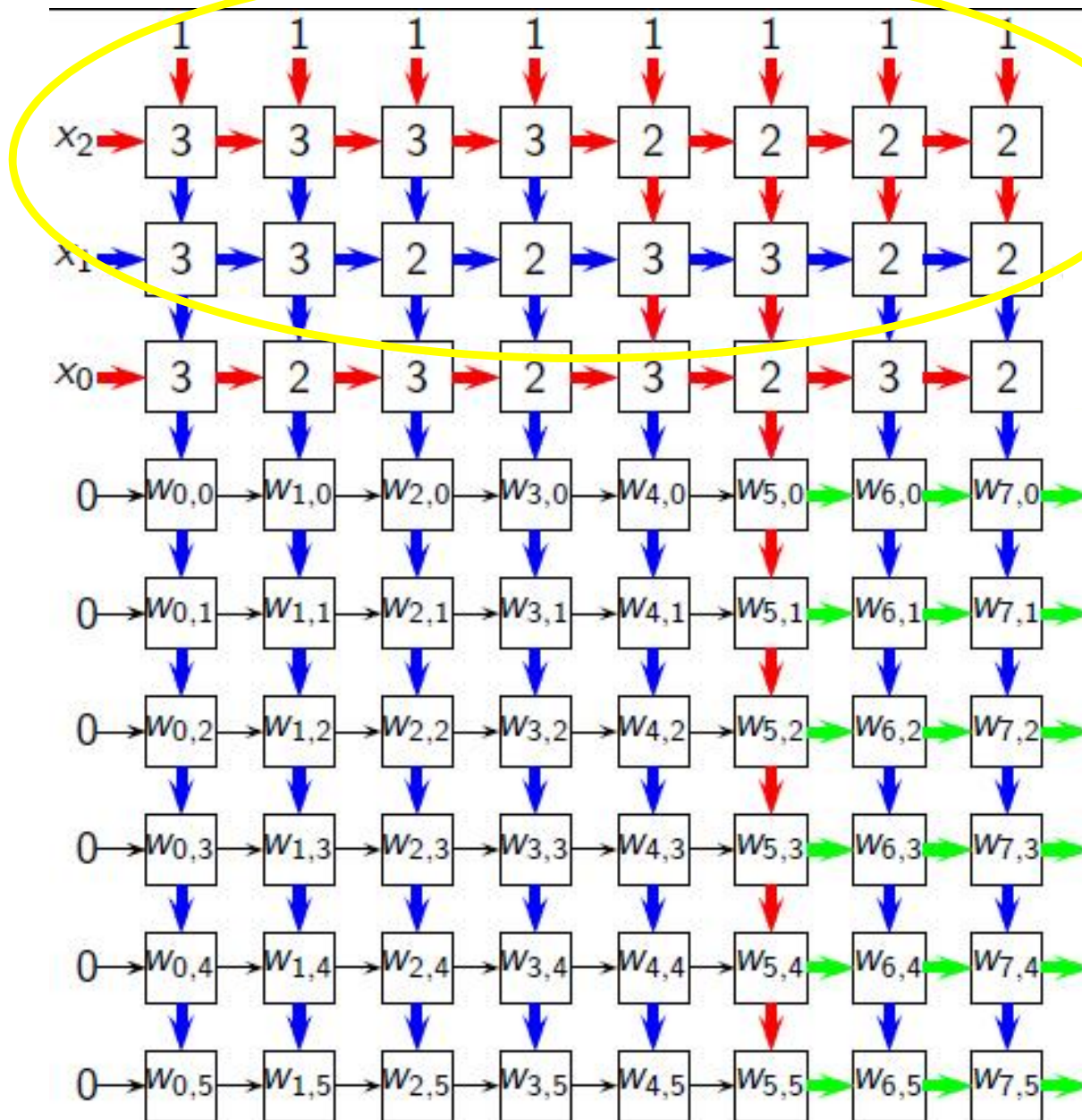
$$5 = (101)_2$$

$$x_2 = 1$$

$$x_1 = 0$$

$$x_0 = 1$$

# 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  $2^n$  Wörtern 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

---

## 7. Programmierbare Bausteine

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

# 7.5 Software PLAs

---

## Software-PLAs

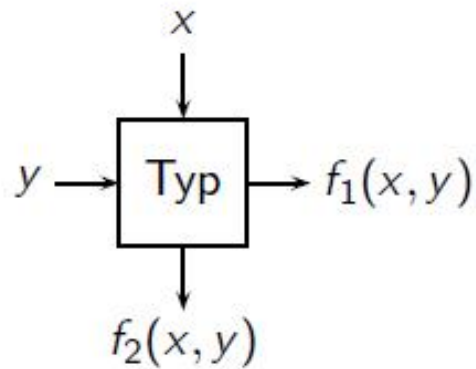
**Beobachtung** vier verschiedene Typen, mit zwei Bits codierbar

### Idee

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

# 7.5 Software PLAs

## Software-PLAs



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

**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

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

# 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$



# 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$

# 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$

## 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$

# 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$

# 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$

## 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$

# 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	$x \vee y$	$x$
2	$y$	$x \wedge y$
3	$y$	$x \wedge \bar{y}$

# 7.5 Software PLAs

---

## Software-PLAs

### Bemerkung zum Einsatz

#### Beobachtung

- für ein  $n \times m$ -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
  - löschar (UV-Licht oder elektrisch)
- CPLD (Complex Programmable Logic Device)
  - programmierbare AND/OR-Matrix
  - programmierbare Rückkopplung
  - Ein-/Ausgabeblocke aus Registern
- FPGA (Field Programmable Gate Array)
  - ähnlich CPLD, aber beliebige Funktion im Grundbaustein realisierbar

# 7. Programmierbare Bausteine

---

## 7. Programmierbare Bausteine

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