technische universität
dortmund

Rechnerstrukturen WS 2012/13

- Speicher
  - SRAM-Realisierung
- Register
  - Schieberegister
  - Datenbus
- Taktung von Digitalrechnern
  - Takt- und Phasenableitung
- Zusammenfassung
- Anhang
- Ausblick

Hinweis: *Folien teilweise a. d. Basis von Materialien von Thomas Jansen*

20. November 2012

Speicher

Speicher

Erinnerung Wir können in einem Flip-Flop ein Bit speichern.

klar 1-Bit-Speicher reichen uns nicht.

klar Wir können in k Flips-Flops k Bits speichern.

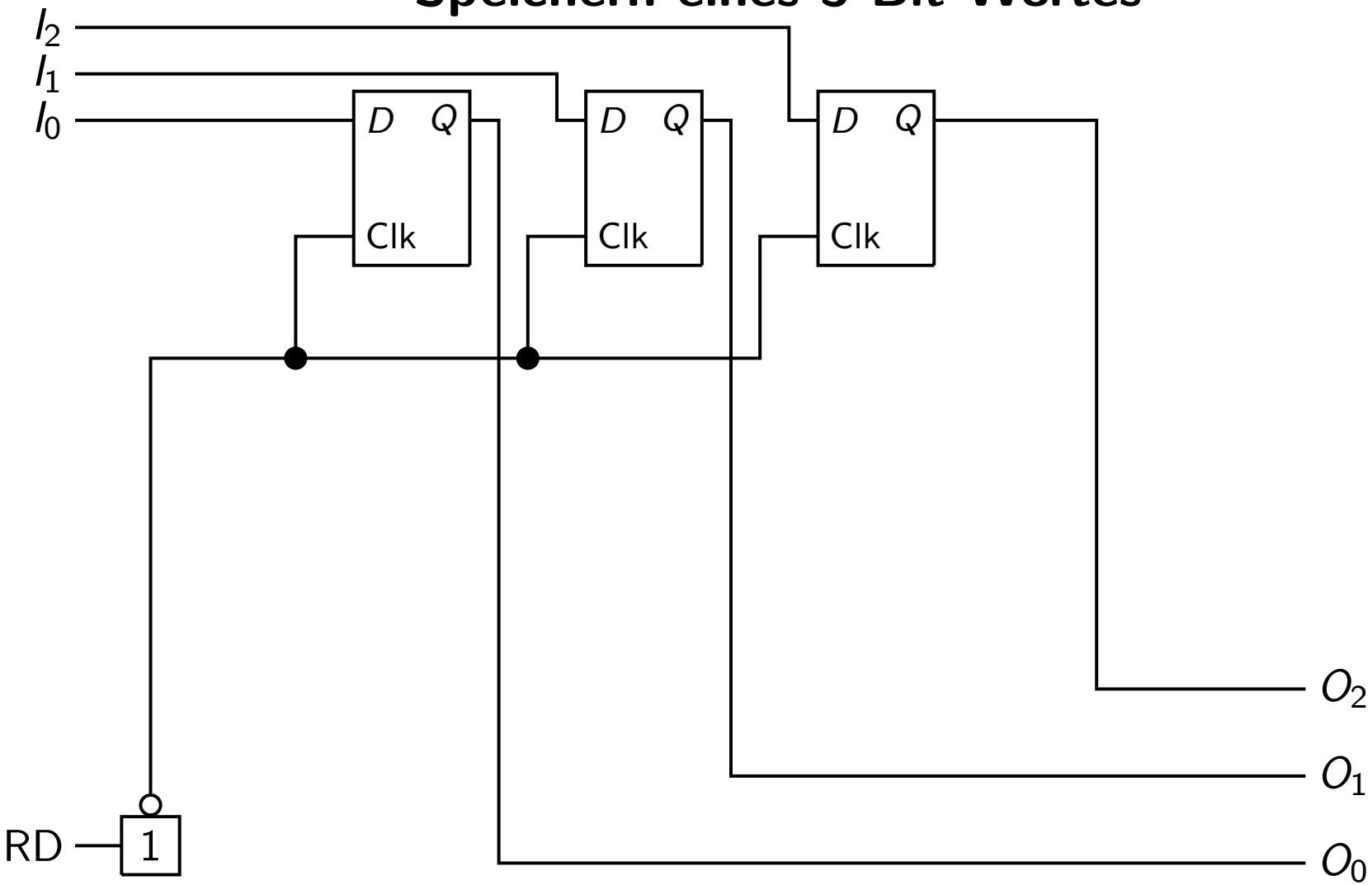
Wie organisieren wir das so, dass der Zugriff bequem ist?

exemplarisch Speicher für 3-Bit-Wörter

Warum 3? passt bequem auf eine Folie

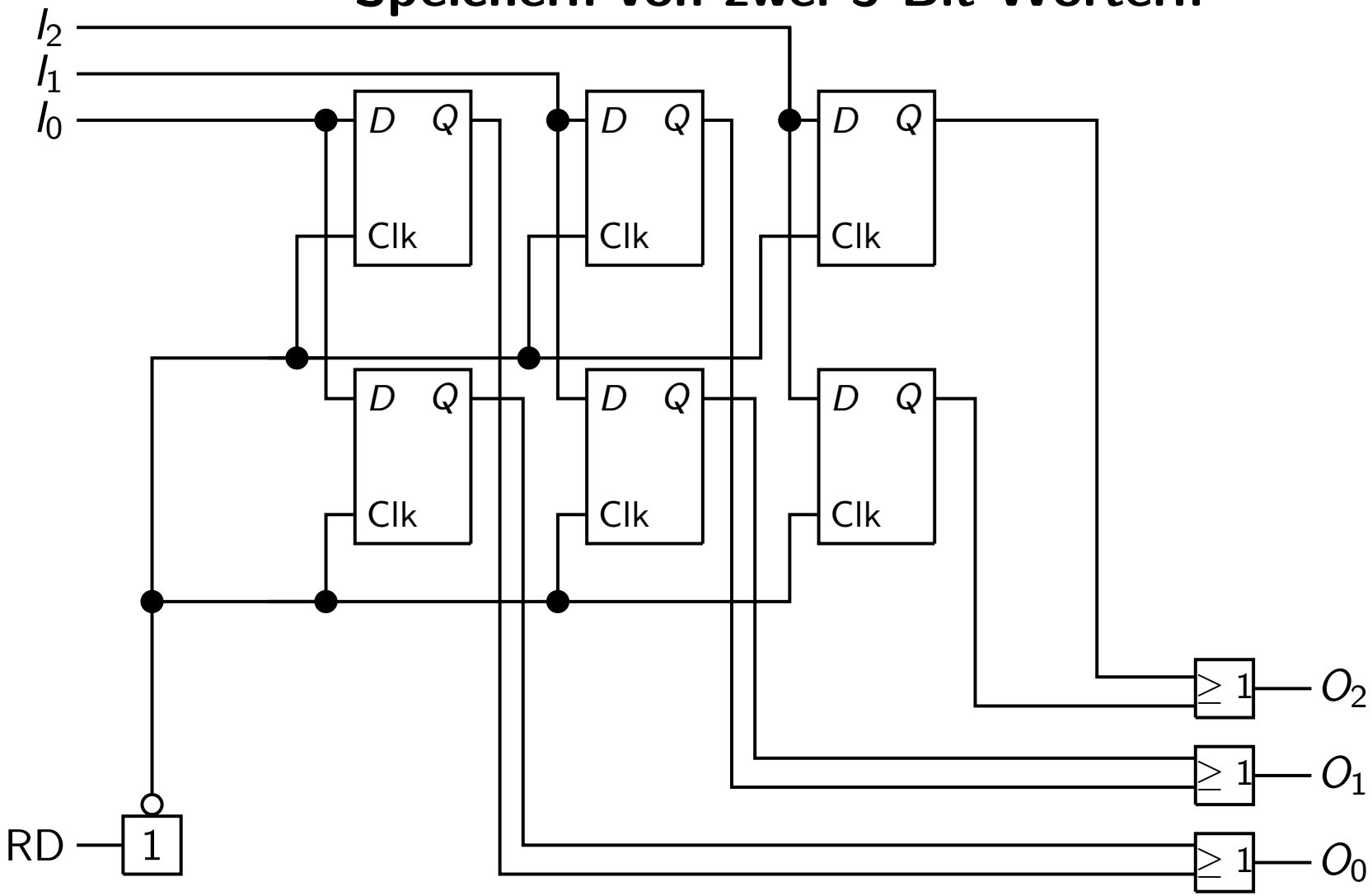
Anmerkung Auch 3-Bit-Speicher reicht in der Praxis nicht aus.

Speichern eines 3-Bit-Wortes



Wie können wir mehr als nur ein Wort speichern?

Speichern von zwei 3-Bit-Wörtern



Beobachtung Schaltung funktioniert so nicht

Speichern von zwei 3-Bit-Wörtern

klar Die Speicherwörter müssen getrennt ansprechbar sein.

also Wir wollen ein Wort wahlfrei addressieren.

klar Wir brauchen eine Adressleitung A_0.

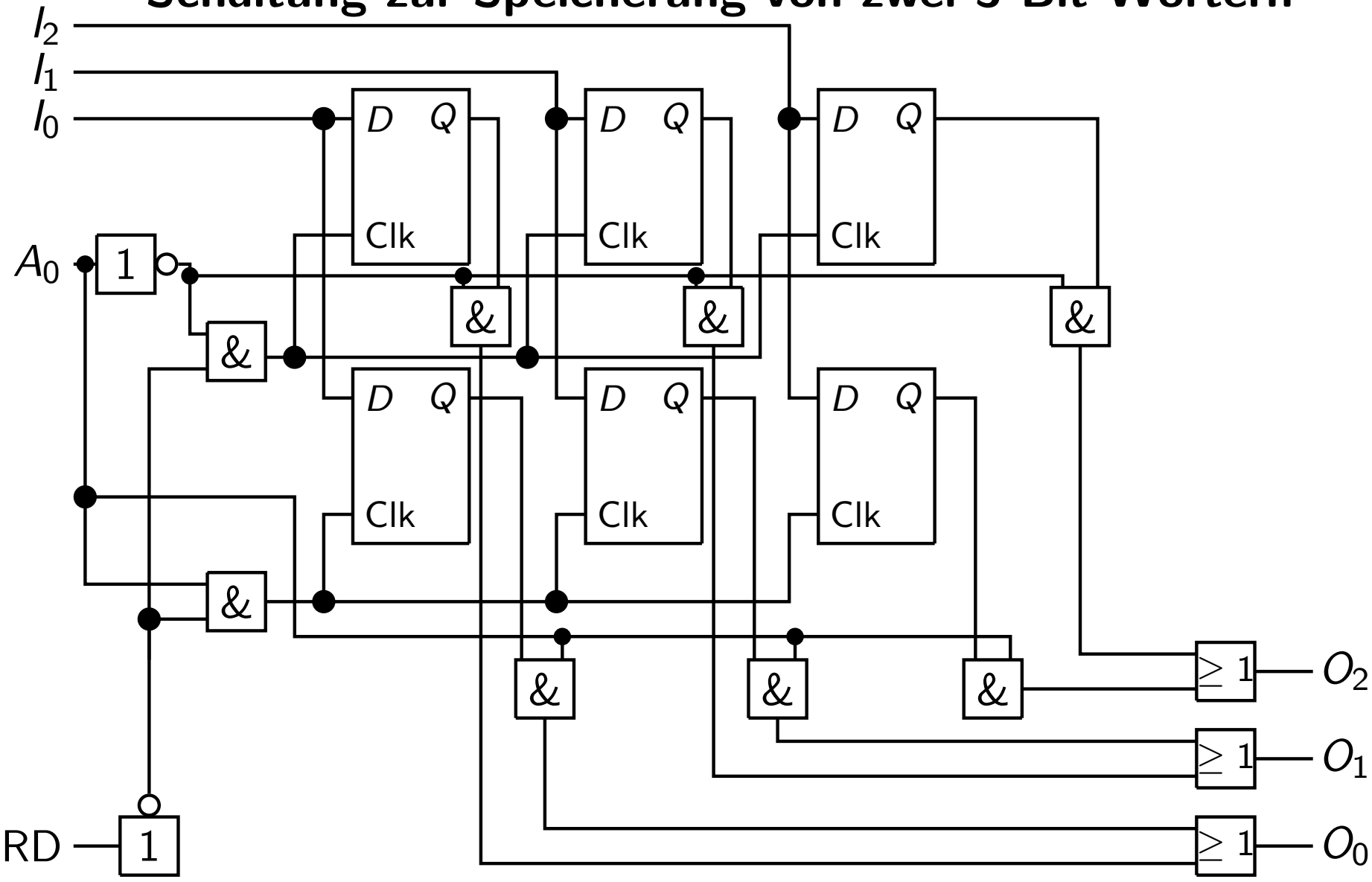
Interpretation

- $A_0 = 0$ Wort w_0 addressiert zum Lesen oder Schreiben
- $A_0 = 1$ Wort w_1 addressiert zum Lesen oder Schreiben

wie bisher RD bestimmt, ob gelesen oder geschrieben wird

- $RD = 1$ Speicherinhalt lesen
- $RD = 0$ Speicherinhalt schreiben

Schaltung zur Speicherung von zwei 3-Bit-Wörtern



Speichererweiterungen

angenommen Schaltung zur Speicherung von zwei 3-Bit-Wörtern im Einsatz

angenommen Speicher reicht im Betrieb nicht aus

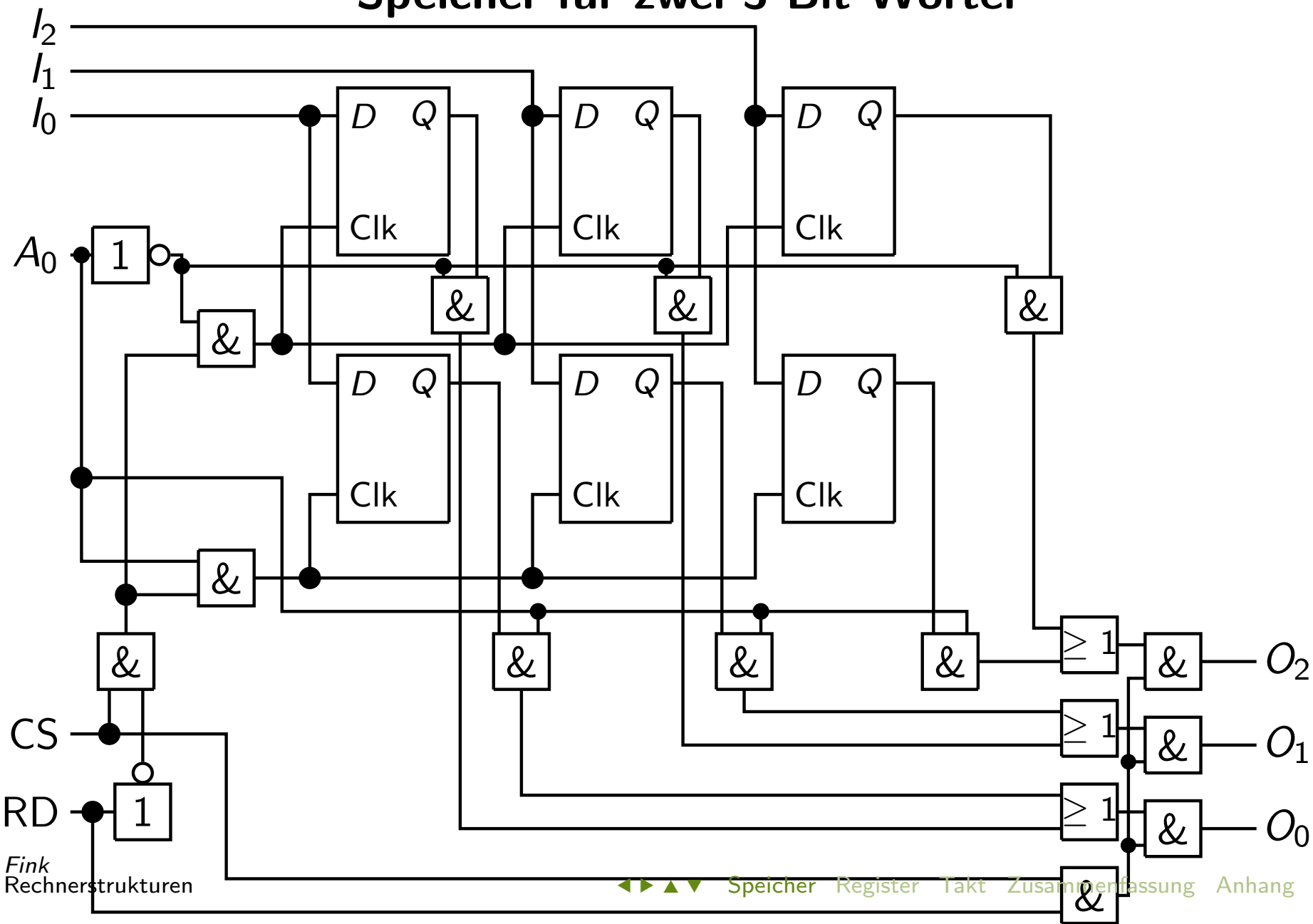
Wunsch Speichererweiterung

Beachte Speichererweiterung bedeutet
„weiteren Speicherbaustein (gleicher Art) hinzufügen“,
nicht „vorhandenen Speicherbaustein durch größeren Speicherbaustein ersetzen“

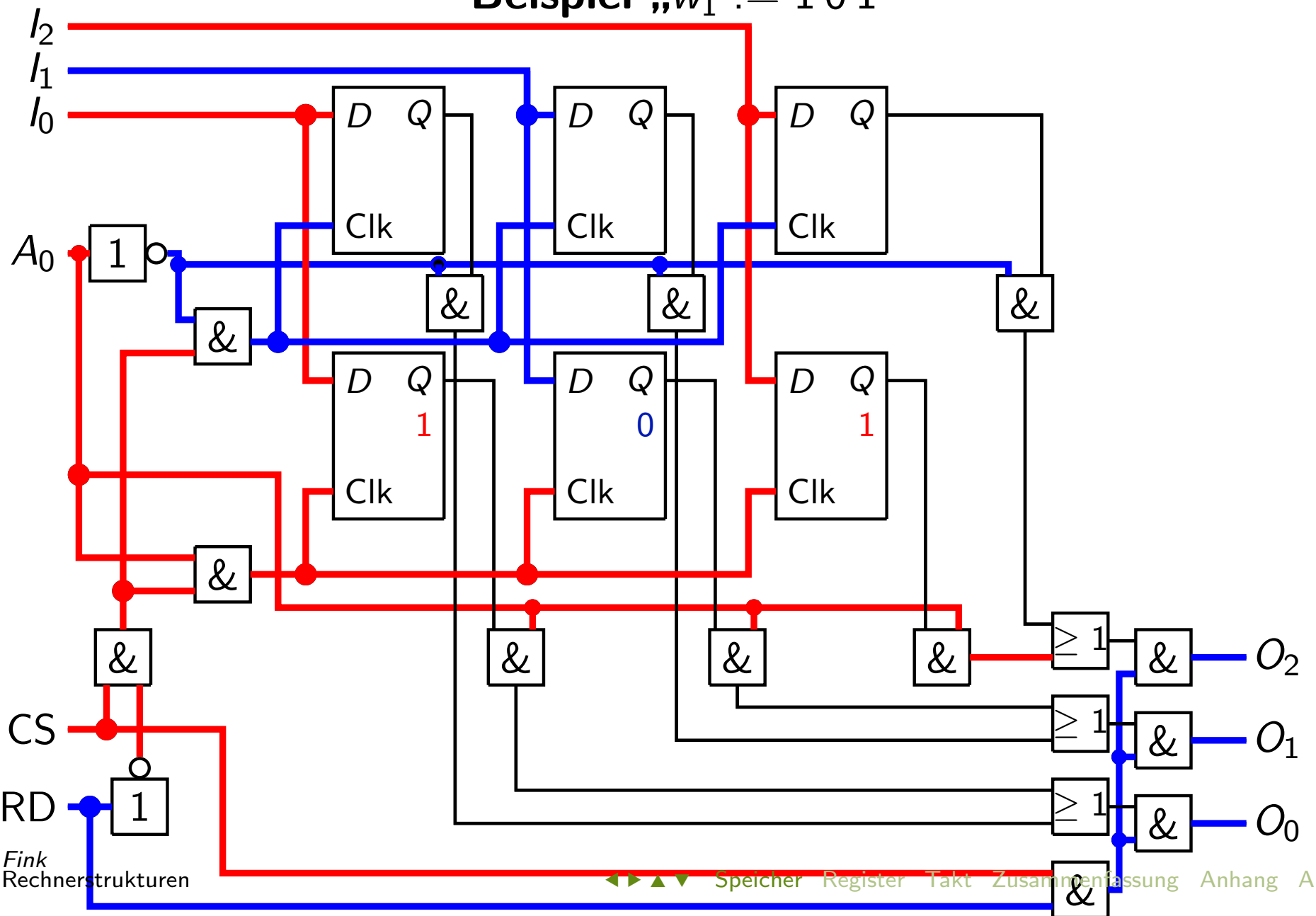
Wie können wir das unterstützen?

Idee zusätzliche Eingabe „Chip Selected“ (CS)

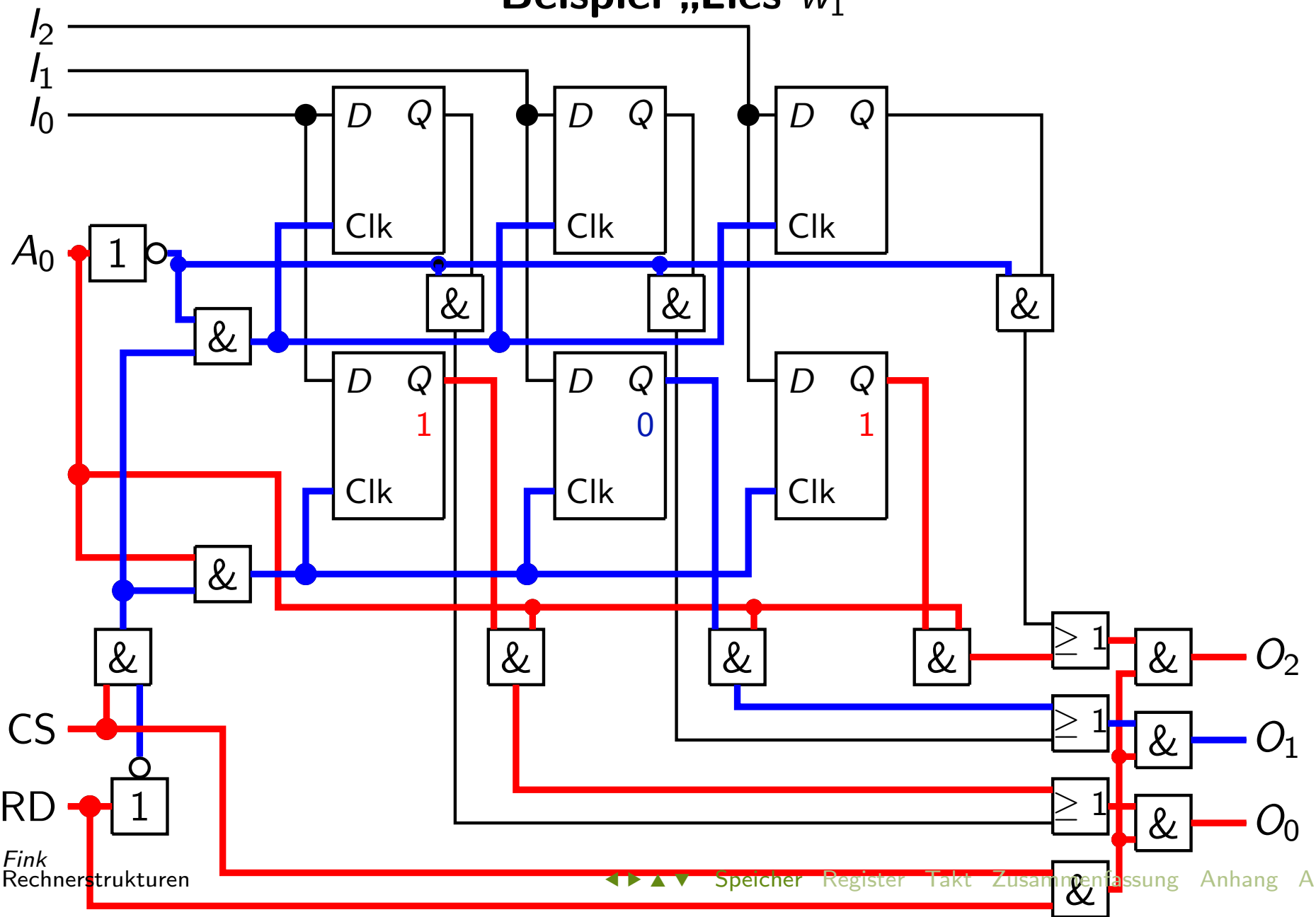
Speicher für zwei 3-Bit-Wörter



Beispiel „$w_1 := 1\,0\,1$“



Beispiel „Lies w_1“



Realistischere Speichergrößen

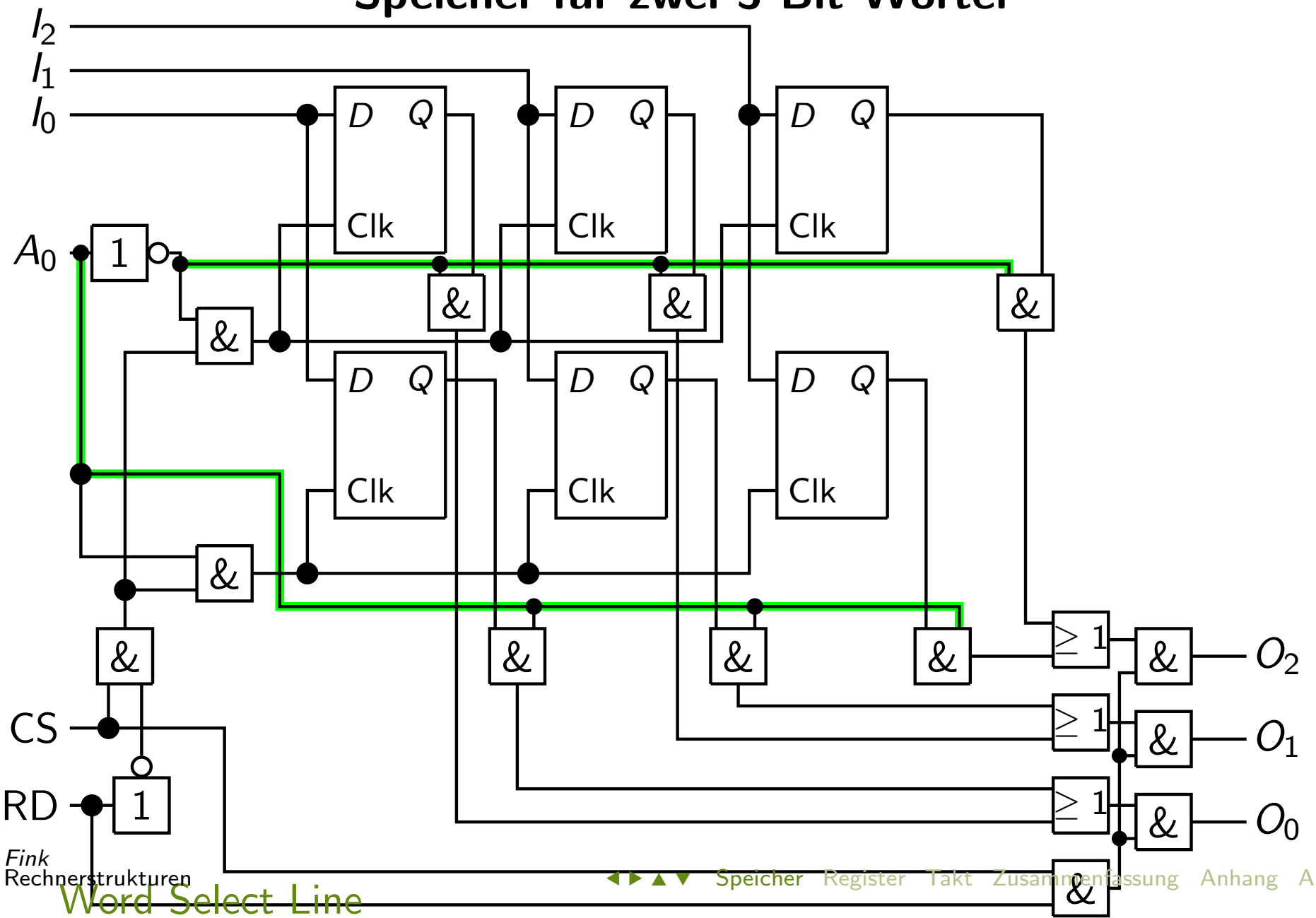
Beobachtung Speichergröße „2 Wörter“ ist nicht realistisch
auch nicht bei Benutzung einiger Bausteine

Wie kommen wir zu realistischen Speichergrößen?

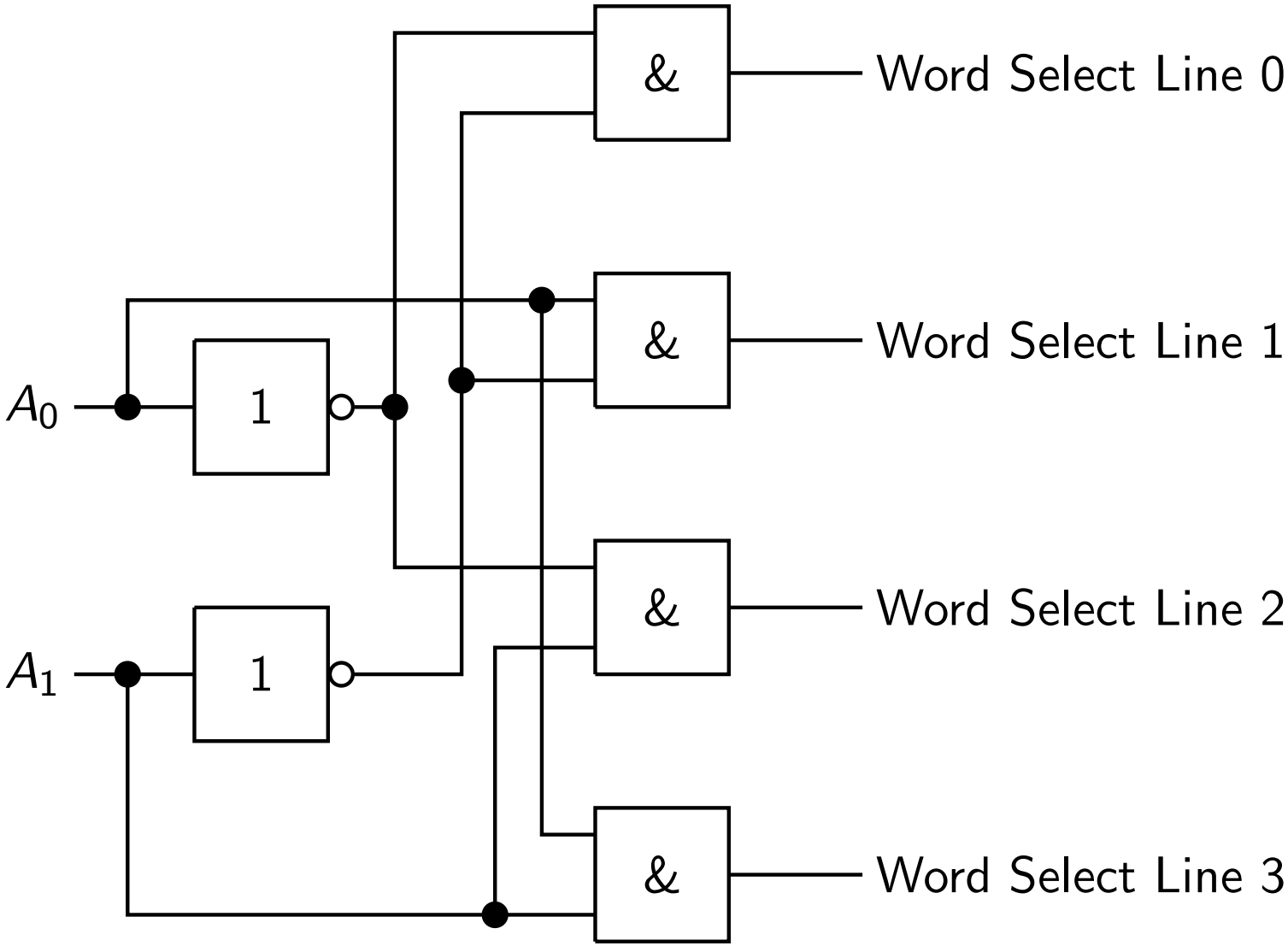
klar größere Wortlänge funktioniert im Prinzip gleich

offen Wie verallgemeinern wir auf > 2 Wörter?

Speicher für zwei 3-Bit-Wörter



Word Select Lines für vier Wörter



Speicher für viele Wörter

allgemein 2^k Wörter speichern

klar k Adressleitungen benötigt

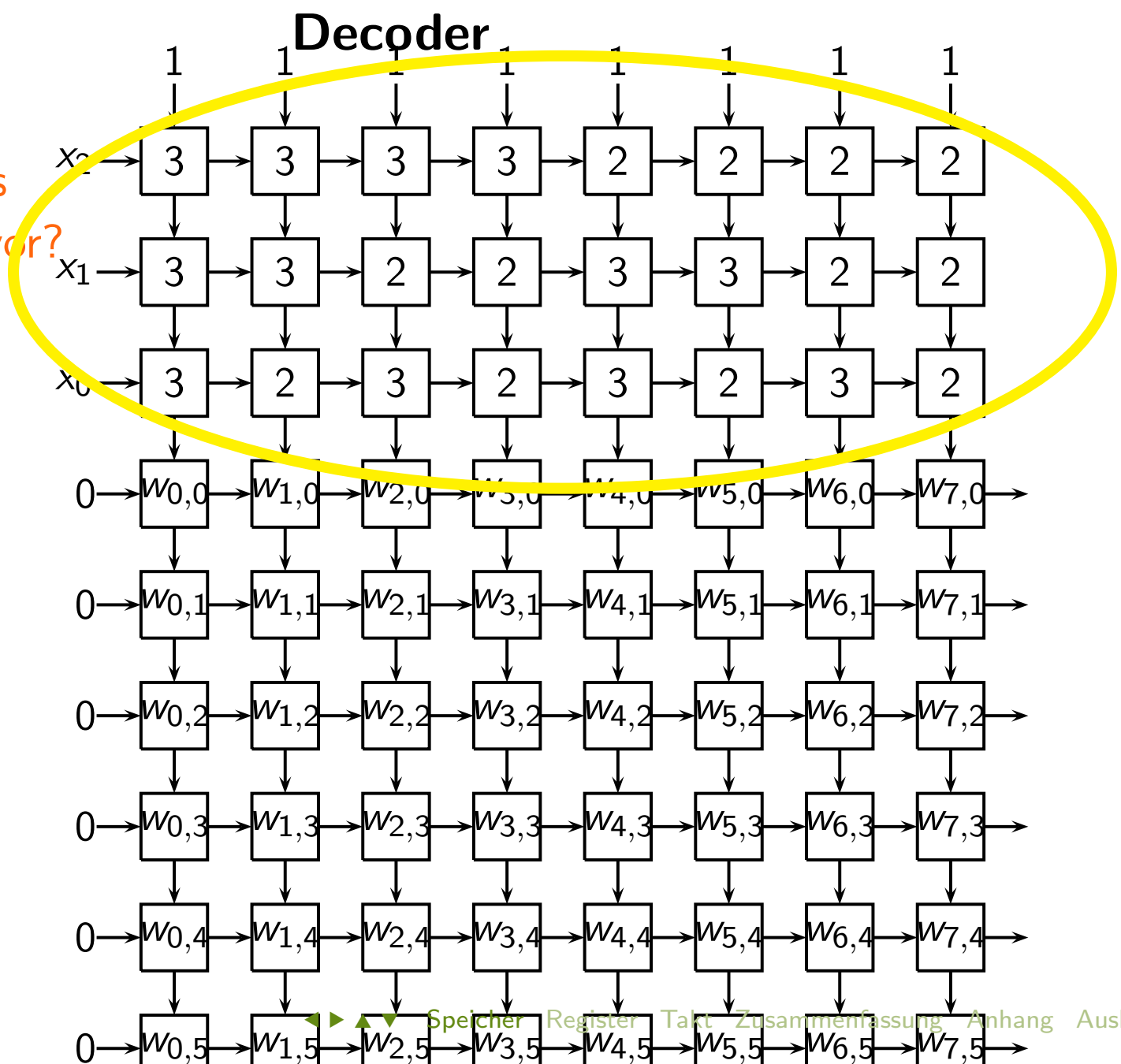
zunächst formale Beschreibung als boolesche Funktion

$f : \{0,1\}^k \to \{0,1\}^{2^k}$
mit $f(A_0, A_1, \ldots, A_{k-1}) = (s_0, s_1, \ldots, s_{2^k-1})$
mit $s_i = \begin{cases} 1 & \text{falls } (A_{k-1}\, A_{k-2} \cdots A_0)_2 = i \\ 0 & \text{sonst} \end{cases}$

Name der Funktion Decoder
genauer $k \times 2^k$-Decoder

Nachdenken

Kommt uns das nicht bekannt vor?



Beobachtung

Und-Teil realisiert Decoder

Demultiplexer

Erinnerung $\mathrm{MUX}_d(y_1, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}) = x_{(y_1\, y_2 \cdots y_d)_2}$

Erinnerung Decoder als boolesche Funktion

$\mathrm{DECODE}_d \colon \{0,1\}^d \to \{0,1\}^{2^d}$ mit

$\mathrm{DECODE}_d(A_0, A_1, \ldots, A_{d-1}) = (s_0, s_1, \ldots, s_{2^d-1})$ mit

$$s_i = \begin{cases} 1 & \text{falls } (A_{d-1}\, A_{d-2}\, \ldots A_0)_2 = i \\ 0 & \text{sonst} \end{cases}$$

Demultiplexer $\mathrm{DEMUX}_D \colon \{0,1\}^{d+1} \to \{0,1\}^{2^d}$ mit

$$\begin{aligned} &\mathrm{DEMUX}_D(x, A_0, A_1, \ldots, A_{d-1}) \\ &= \big(x \wedge \{\mathrm{DECODE}_d(A_0, A_1, \ldots, A_{d-1})\}_0, \\ &\quad x \wedge \{\mathrm{DECODE}_d(A_0, A_1, \ldots, A_{d-1})\}_1, \ldots, \\ &\quad x \wedge \{\mathrm{DECODE}_d(A_0, A_1, \ldots, A_{d-1})\}_{2^d-1}\big) \end{aligned}$$

Speicher

Realisiert man Speicher wirklich so?

... nicht für Hauptspeicher von Rechnern (kompakter als DRAM [dynamic random access memory] möglich)

tatsächlich typische SRAM-Realisierung (SRAM = static RAM)

Eigenschaften

- dauerhaft
- schnell
- Zugriffszeit von Daten unabhängig
- teuer
- hoher Stromverbrauch

Bemerkung typisch für Cache- oder Register-Speicher

Cache

$128\ \mathrm{K} \times 8$
$= 128 \cdot 1024 \times 8$
$= 2^{17} \times 8$
darum 17 Adress-leitungen A_0–A_{16}

Chip Enable
bei uns CS
Write Enable
bei uns RD $= 0$
Output Enable
bei uns RD $= 1$

Beobachtung
$128 \cdot 1024 \cdot 8$
$= 1\,048\,576$
Flip-Flops

SRAM

128K X 8 HIGH SPEED CMOS STATIC RAM

FEATURES

- Fast Address Access Times : 10/12/15ns
- Single 3.3V ±0.3V power supply
- Center power/ground pin configuration
- Low Power Consumption : 110/105/100mA
- TTL I/O compatible
- 2.0V data retention mode
- Automatic power-down when deselected
- Available packages :
  - 32-pin 300 mil and 400 mil SOJ
  - 32-pin TSOP 8x13.4mm and 8x20mm
  - 36-Ball CSP (8x10mm)

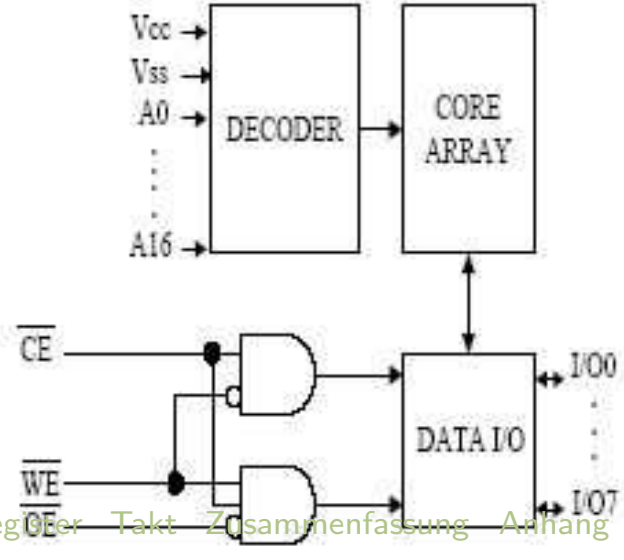
GENERAL DESCRIPTION

The T14L1024N is a one-megabit density, fast static random access memory organized as 131,072 words by 8 bits. It is designed for use in high performance memory applications such as main memory storage and high speed communication buffers. Fabricated using high performance CMOS technology, access times down to 10ns are achieved.

PART NUMBER EXAMPLES

	PACKAGE	SPEED
T14L1024N-10J	SOJ 300mil	10ns
T14L1024N-10W	SOJ 400 mil	10ns
T14L1024N-10P	TSOP 8x13.4mm	10ns
T14L1024N-10H	TSOP 8x20mm	10ns
T14L1024N-10C	36-Ball CSP	10ns

BLOCK DIAGRAM



Spezielle Speicher: Register

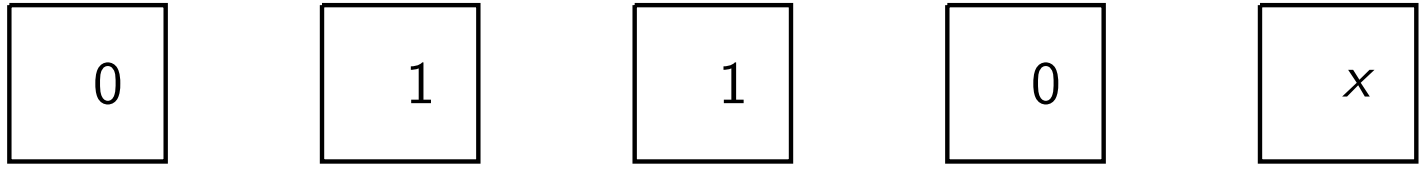
Fakt in Computern meist keine direkte Speichermanipulation

stattdessen spezielle, direkt der CPU zugehörige Speicherzellen
Register

klar Register sind auch nur Flip-Flops

aber spezielle Funktionalität $\rightsquigarrow$ gesonderte Betrachtung

Wir betrachten Schieberegister



Funktion Speicherinhalt nach links oder rechts verschieben

Was machen wir an den Rändern?

möglich zyklisch verschieben

möglich streichen und beliebig auffüllen

Entwurf Schieberegister

Entwurf nicht-zyklisches Schieberegister mit drei Bits

Warum gerade drei Bits?

- klein genug, um auf die Folie zu passen
- groß genug, um alles Wichtige zu enthalten
  - Bit am linken Rand
  - Bit am rechten Rand
  - mittleres Bit

Eingänge

- d (direction) Schieberichtung ($d = 0 \widehat{=}$links, $d = 1 \widehat{=}$rechts)
- x aufzufüllender Wert

Vereinfachung: betrachten nicht Ein-/Ausgänge zum Schreiben/Lesen des Registers als Ganzem

Mealy-Automat zum nicht-zyklischen Schieberegister

$\Sigma = \{00, 01, 10, 11\}$ Interpretation $d\,x \in \Sigma$

$\Delta = \emptyset$ also $\forall q \in Q, w \in \Sigma\colon \lambda(q, w) = \varepsilon$

$Q = \{000, 001, 010, 011, 100, 101, 110, 111\}$ Interpretation Registerinhalt

$q_0 = 000$ (völlig willkürlich)

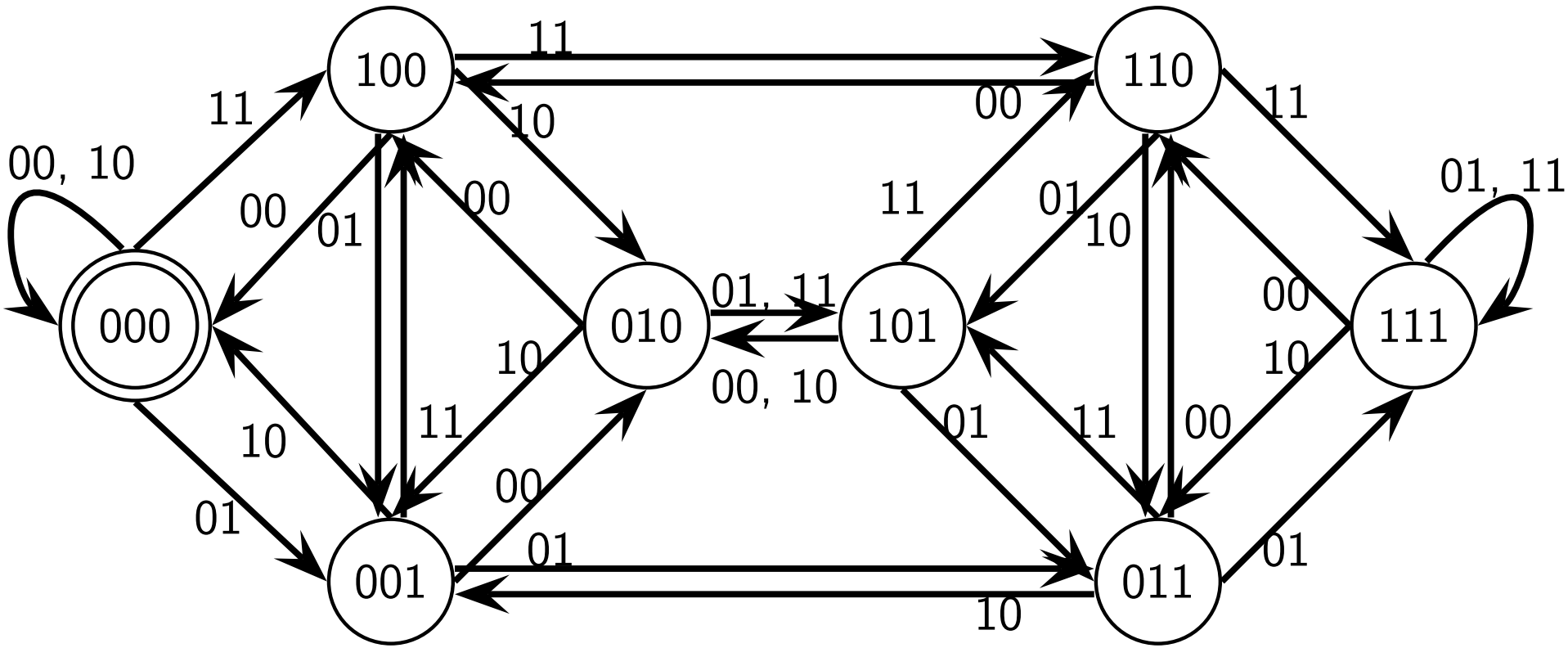
$\delta\colon Q \times \Sigma \to Q$

$q \in Q$	$w \in \Sigma$	$\delta(q, w) \in Q$
000	00	000
000	01	001
000	10	000
000	11	100
⋮	⋮	⋮

insgesamt 32 Zeilen

Grafische Darstellung Mealy-Automat

$\Sigma = \{00, 01, 10, 11\}$, $w = d\,x$ ($d = 0 \widehat{=}$links, $d = 1 \widehat{=}$rechts)
$\Delta = \emptyset$, $Q = \{0, 1\}^3$, $q_0 = 000$, $\lambda(q, w) = \varepsilon$



Schaltwerk-Synthese abgekürzt

Codierung

kanonisch $w = d\,x$ durch d, x
$q = q_l\,q_m\,q_r \in \{0,1\}^3$ durch q_l, q_m, q_r

Müssen wir jetzt Tabellen aufstellen?

Einsicht Strukturverständnis kann helfen

zum linken Bit

1. Fall $d = 0 \mathrel{\widehat{=}} \text{links}$

klar $q_{l(\text{neu})} = q_m$

2. Fall $d = 1 \mathrel{\widehat{=}} \text{rechts}$

klar $q_{l(\text{neu})} = x$

also $q_{l(\text{neu})} = \overline{d}\,q_m \vee d\;x$

Boolesche Funktionen für mittleres und rechtes Bit

zum rechten Bit

1. Fall $d = 0 \widehat{=} \text{links}$

klar $q_{r(\text{neu})} = x$

2. Fall $d = 1 \widehat{=} \text{rechts}$

klar $q_{r(\text{neu})} = q_m$

also $q_{r(\text{neu})} = \overline{d}\, x \vee d\ q_m$

zum mittleren Bit

1. Fall $d = 0 \widehat{=} \text{links}$

klar $q_{m(\text{neu})} = q_r$

2. Fall $d = 1 \widehat{=} \text{rechts}$

klar $q_{m(\text{neu})} = q_l$

also $q_{m(\text{neu})} = \overline{d}\, q_r \vee d\ q_l$

Schaltwerk nicht-zyklisches Schieberegister

Moment! Hätten wir nicht Flip-Flops wählen und
und Ansteuerfunktionen berechnen müssen?

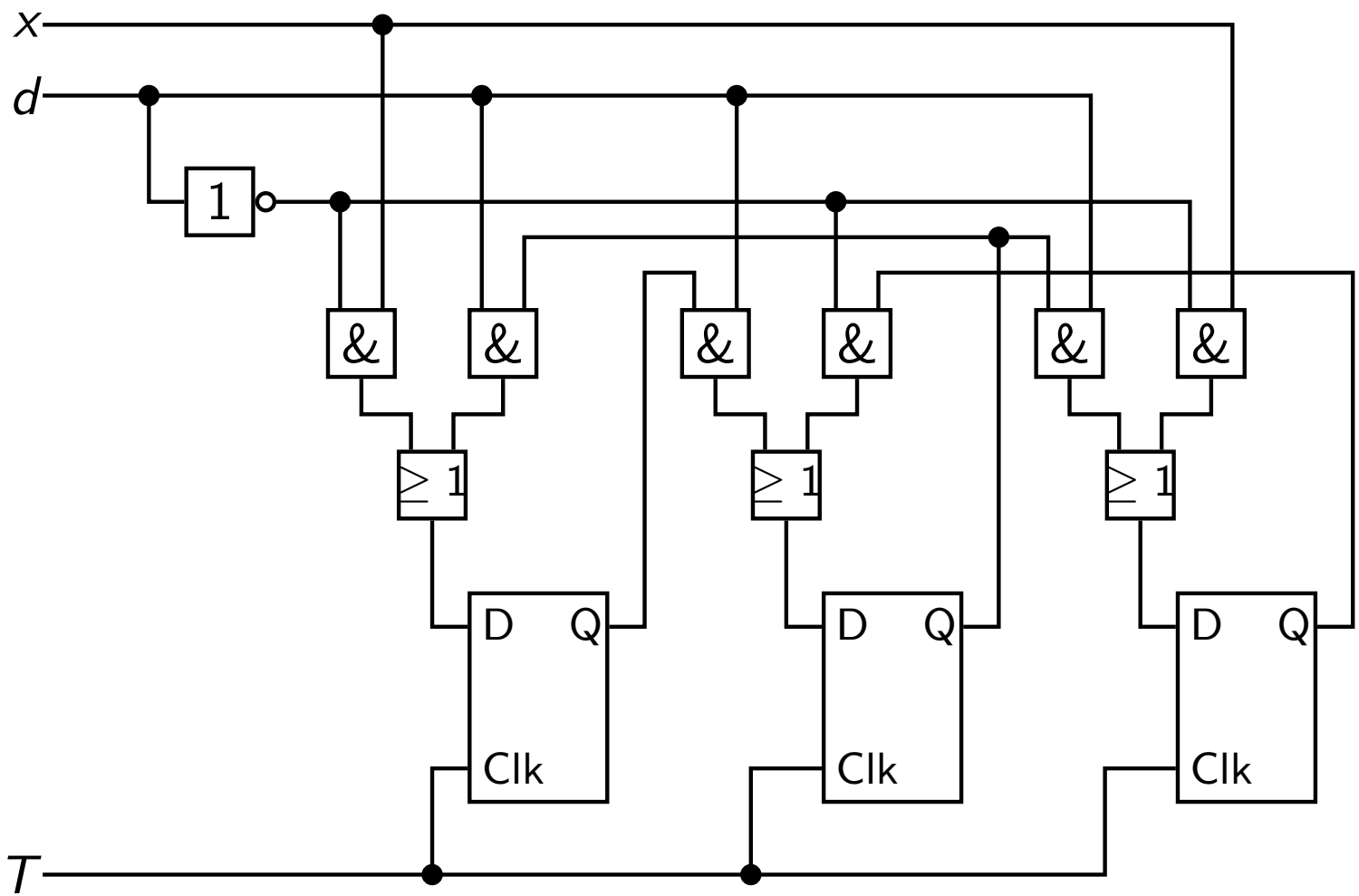
Einsicht nicht, wenn man D-Flip-Flops verwendet

also nur noch

$$q_{l(\text{neu})} = \overline{d}\, q_m \vee d\ x$$
$$q_{m(\text{neu})} = \overline{d}\, q_r \vee d\ q_l$$
$$q_{r(\text{neu})} = \overline{d}\, x \vee d\ q_m$$

direkt in ein Schaltwerk übertragen

Schaltwerk nicht-zyklisches Schieberegister



Zyklisches Schieberegister

jetzt zyklisches Schieberegister
wieder für drei Bits
wieder ohne Daten-Eingabe

zuerst Mealy-Automat

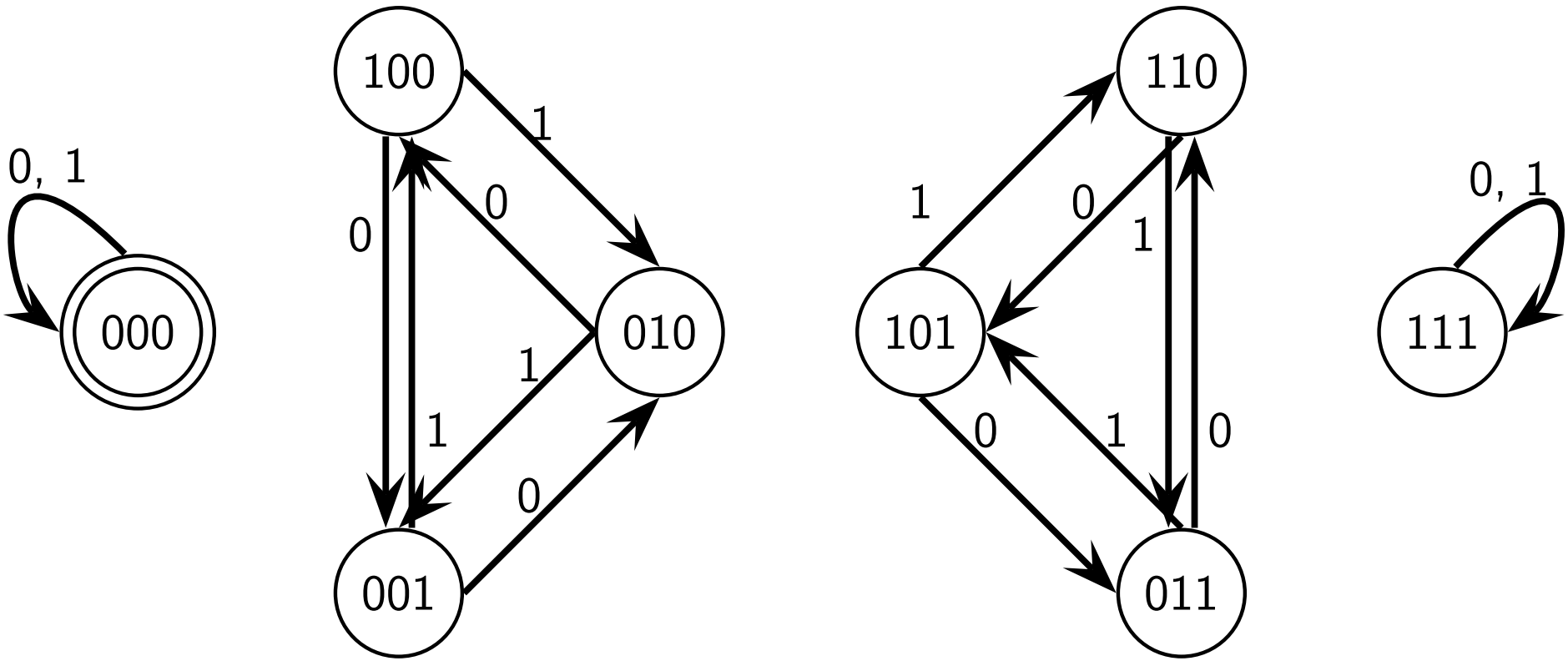
wie vorhin $Q = \{0,1\}^3$, $\Delta = \emptyset$, $\forall q \in Q, w \in \Sigma\colon \lambda(q,w) = \varepsilon$

anders $\Sigma = \{0,1\}$ (weil x fehlt)

wie vorhin Interpretation $0 \widehat{=}$links, $1 \widehat{=}$rechts

Mealy-Automat zyklisches Schieberegister

$\Sigma = \{0, 1\}$, $w = d$ ($d = 0 \widehat{=}$links, $d = 1 \widehat{=}$rechts)
$\Delta = \emptyset$, $Q = \{0, 1\}^3$, $q_0 = 000$, $\lambda(q, w) = \varepsilon$



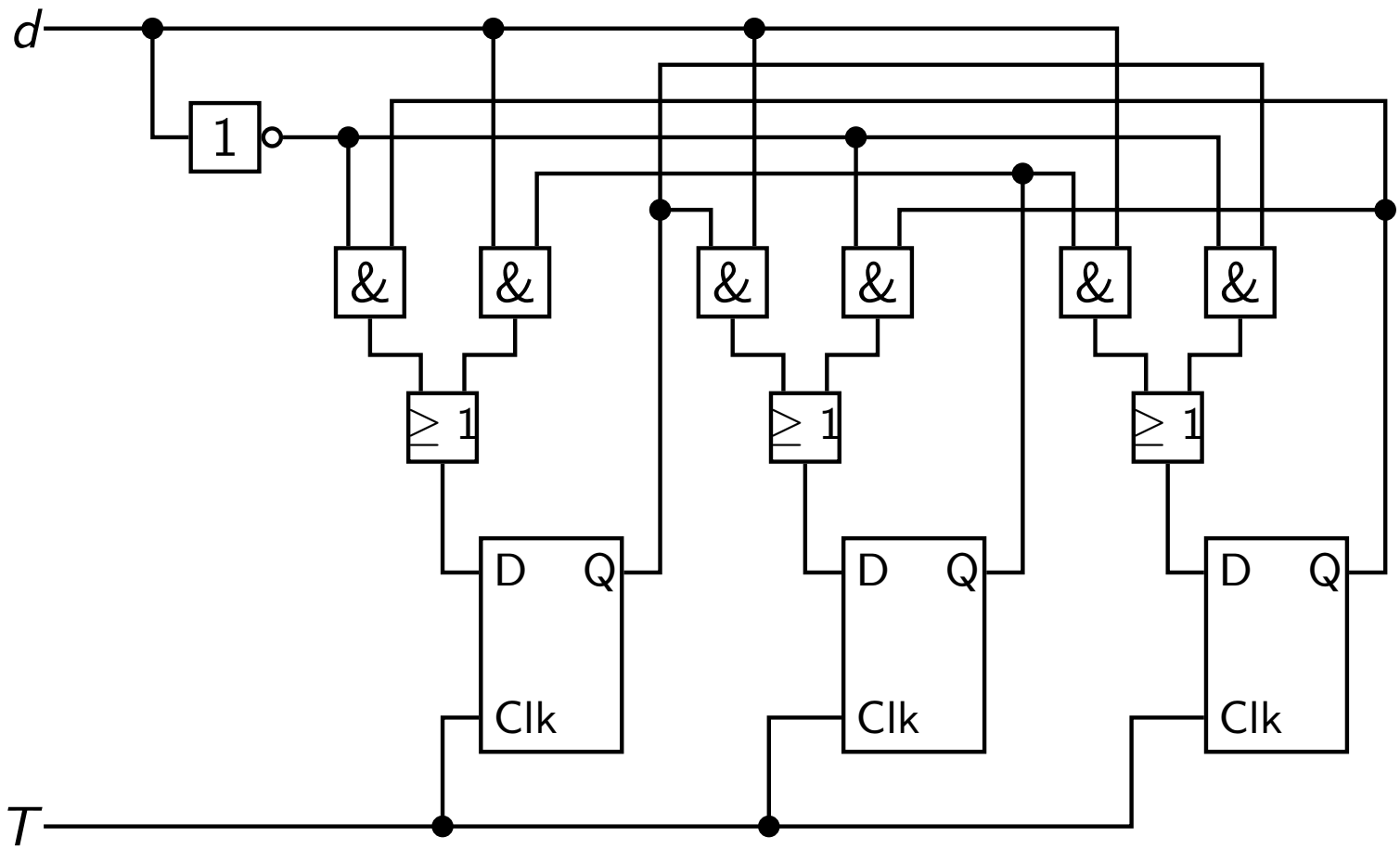
Boolesche Funktionen zum Mealy-Automaten

wie vorhin Funktionen für q_l, q_m, q_r direkt ableiten

- $q_l = \overline{d}\, q_m \vee d\, q_r$
- $q_m = \overline{d}\, q_r \vee d\, q_l$
- $q_r = \overline{d}\, q_l \vee d\, q_m$

jetzt Schaltwerk direkt angebbar
bei Verwendung von D-Flip-Flops

Schaltwerk zyklisches Schieberegister



[↗ nicht-zyklisch]

Schieberegister allgemein

Können wir die Funktionen für die Bits auch allgemein angeben?

Notation Bits $q_{n-1}, \ldots, q_1, q_0$
q_{n-1} am linken Rand, q_0 am rechten Rand

zyklisches Schieberegister

$q_i = \overline{d}\, q_{(i-1) \bmod n} \vee d\, q_{(i+1) \bmod n}$ auch am Rand

Auch am Rand?

nicht-zyklisches Schieberegister

$q_i = \overline{d}\, q_{i-1} \vee d\, q_{i+1}$ auch am Rand mit $q_{-1} := x$, $q_n := x$

Welche Operationen können Schieberegister realisieren?

Datentransfer

klar Daten müssen gelegentlich zwischen Registern transferiert werden

Beispiel Zwischenergebnis merken

Wie kann man Daten von einem Register in ein anderes transferieren?

klar auf Leitungen zwischen den Registern

Wie verbindet man Register geschickt?

Kriterien:

- Einfachheit (d. technischen Realisierung)
- Sparsamkeit (i.S.v. Schaltungsaufwand)
- Möglichkeit zum effizienten Datenaustausch

Register verbinden

eine Möglichkeit paarweise verbinden

Vorteile

- direkter Datenaustausch, also schnell
- bis $n/2$ Registerpaare gleichzeitig aktiv, also gute Parallelisierung

Nachteil

- $\binom{n}{2}$ Verbindungen

Beispiel $\binom{16}{2} = \frac{16 \cdot 15}{2} = 120$

Anmerkung meist nicht realisiert

Register sparsam verbinden

andere Möglichkeit alle Register mit einer gemeinsamen Leitung verbinden

Vorteile

- nur eine Leitung, also einfach
- Datenaustausch direkt, also schnell

Nachteile

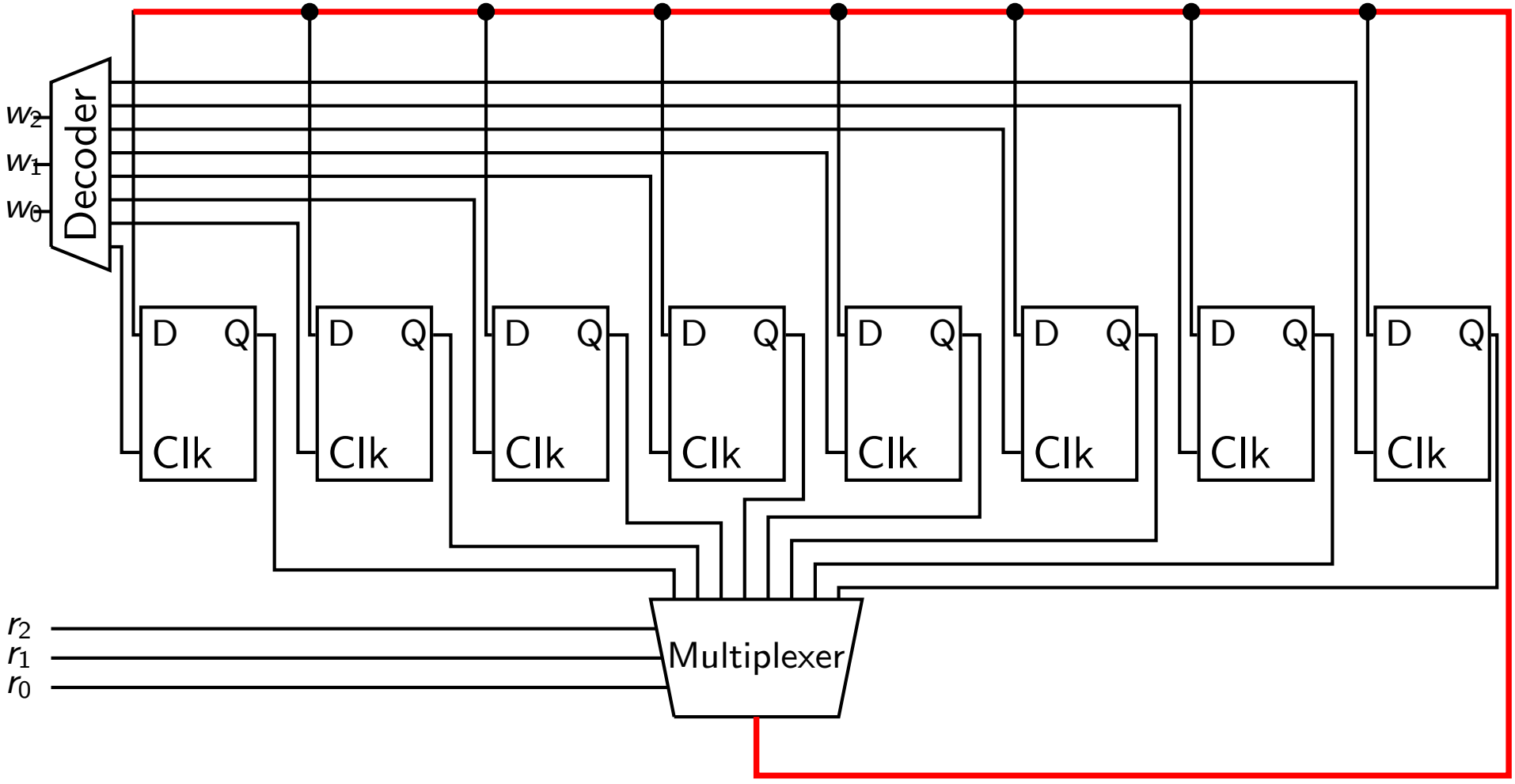
- immer nur ein Paar aktiv, also langsam
- Steuerlogik erforderlich, nicht ganz einfach

Name Bus

Anmerkungen

- häufig verwendet
- Engpass durch mehrere Busse entschärfbar

Bus für acht 1-Bit-Register



Verallgemeinerung

Verallgemeinerung auf mehr Register

- mehr Adressleitungen zum Adressieren des Quell- und Zielregisters
- größere Multiplexer und Decoder

Verallgemeinerung auf breitere Register

- je Register entsprechend mehr Flip-Flops (je Bit ein Flip-Flop)
- entsprechend mehr Multiplexer (je Bit ein Multiplexer)
- entsprechend mehr Bus-Leitungen (je Bit eine Bus-Leitung)

Taktung von Digitalrechnern

Taktung von Digitalrechnern

Gibt es in einem Rechner eigentlich **den** Takt?

jein oft mehrere Takte
aber nur eine „Uhr“

Wie funktioniert das?

Einsicht schneller geht nicht, verschoben oder langsamer schon

Langsamer – Wie geht das?

klar Takt(e) auslassen

klar dazu Zähler ausreichend

Beobachtung Zähler wie jedes Schaltwerk realisierbar

Taktreduktion mit Zählern

Idee: Zähle Taktimpulse → erzeuge in jedem n-ten Takt einen *neuen* Impuls für den langsameren abgeleiteten Takt

Erforderlich: Zähler modulo n, d.h. $z' \leftarrow (z+1) \bmod n$

Realisierung als synchrones Schaltwerk

- Zustand $\hat{=}$ Zählerstand, d.h.: $Q = \{0, 1, \ldots, n-1\}$ (speicherbar in $\log_2 n$ Bits / Flip-Flops)
- Ausgabe, z.B. wenn Zählerstand 0 erreicht → nicht betrachtet
- Eingabe? ... eigentlich nicht zwingend erforderlich, aber Umschaltung der Zählrichtung möglich!
  $\Rightarrow \Sigma = \{0, 1\}$ (d.h. 0 vorwärts, 1 rückwärts zählen)

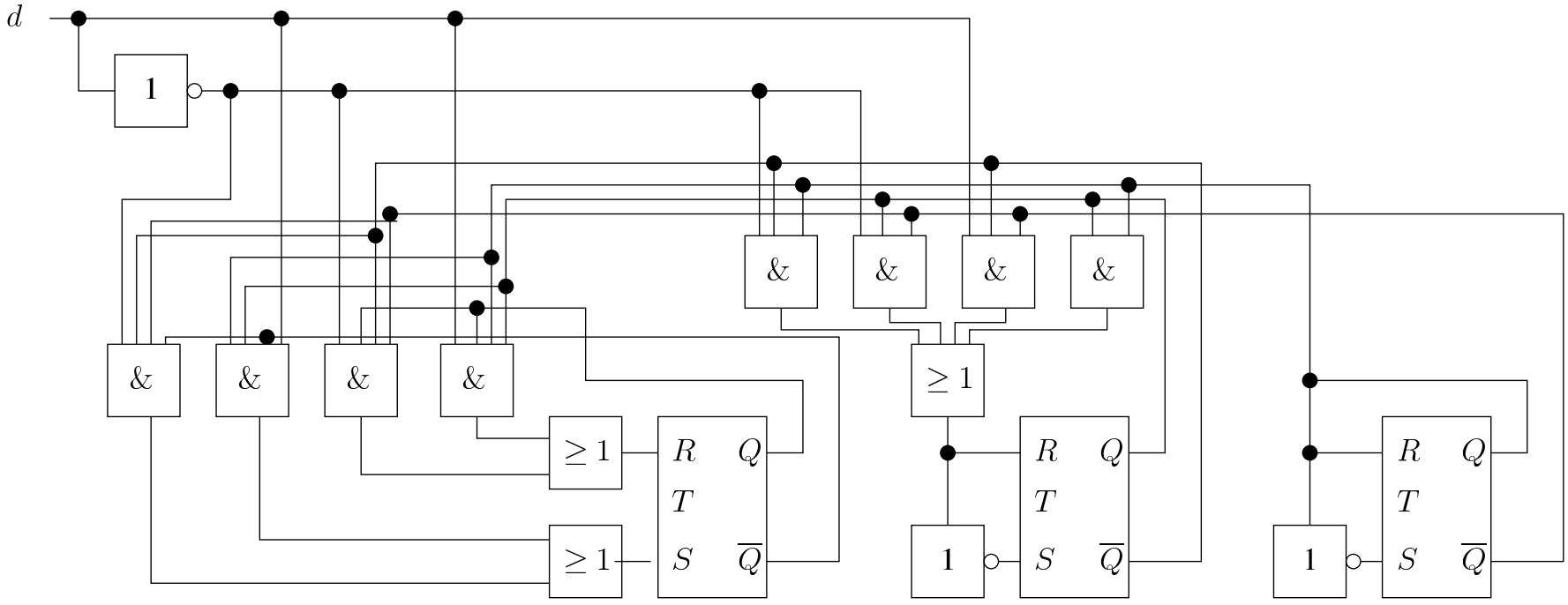
Beispiel: Zähler modulo 8

Zustandsüberführungsfunktion: $\delta(q, d) = (q + (-1)^d) \bmod 8$

Achtung bei Implementierung!

d	q			q_{neu}			R_l	S_l	R_m	S_m	R_r	S_r
0	0	0	0	1	1	1	0	1	0	1	0	1
0	0	0	1	0	0	0	$*$	0	$*$	0	1	0
0	0	1	0	0	0	1	$*$	0	1	0	0	1
0	0	1	1	0	1	0	$*$	0	0	$*$	1	0
0	1	0	0	0	1	1	1	0	0	1	0	1
0	1	0	1	1	0	0	0	$*$	$*$	0	1	0
0	1	1	0	1	0	1	0	$*$	1	0	0	1
0	1	1	1	1	1	0	0	$*$	0	$*$	1	0
1	0	0	0	0	0	1	$*$	0	$*$	0	0	1
1	0	0	1	0	1	0	$*$	0	0	1	1	0
1	0	1	0	0	1	1	$*$	0	0	$*$	0	1
1	0	1	1	1	0	0	0	1	1	0	1	0
1	1	0	0	1	0	1	0	$*$	$*$	0	0	1

Schaltwerk des Zählers modulo 8

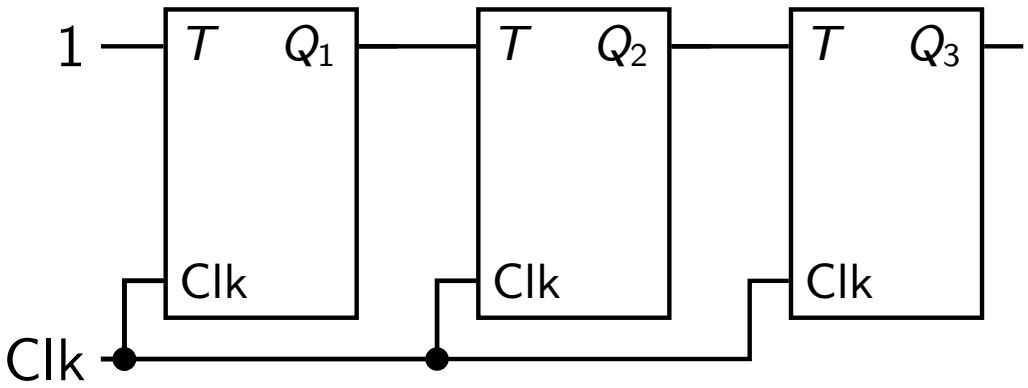


Nachteil: aufwendig!

Gibt es einfachere Möglichkeiten der Taktreduktion?

Taktableitung

Betrachte eine "Kette" von T-Flip-Flops

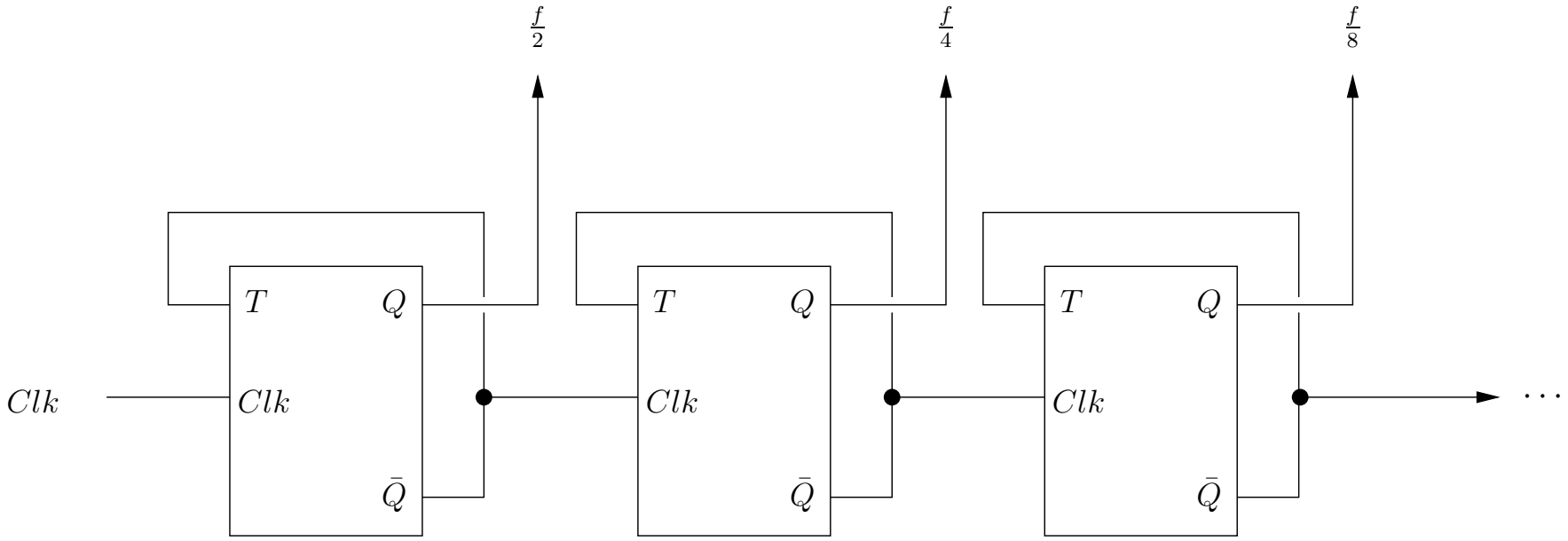


Takt	Q_1	Q_2	Q_3
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	1
4	0	0	0
5	1	0	0
6	0	1	0
7	1	1	1

Nachteil: Taktteilungsschema nicht homogen (ab Q_3)!

Taktteilung

Betrachte "Kette" von T-Flip-Flops in asynchroner Schaltung



Man erhält einen asynchronen n-bit Zähler (hier: 3 bit).

Hinweis: Zähler verwendet flankengetriggerte T-Flip-Flops.

Zusammenfassung

Zusammenfassung

- Repräsentation von Daten
  - natürliche Zahlen, ganze Zahlen
  - rationale Zahlen (IEEE 754)
  - *auch:* Texte, andere Daten
- Boolesche Funktionen und Schaltnetze
  - Boolesche Funktionen
    - Boolesche Algebra, Repräsentation boolescher Funktionen
    - Normalformen (DNF, KNF, RNF)
    - Repräsentation boolescher Funktionen mit OBDDs
  - Schaltnetze & Rechnerarithmetik
    - Addition (Ripple-Carry Addierer, Carry-Look-Ahead-Addierer)
    - Multiplikation (Carry-Save Addierer & Wallace-Tree)
    - Subtraktion $\hat{=}$ Addition negativer Zahlen
    - Gleitkommazahlen: Multiplikation, Addition
  - ...

Zusammenfassung II

- Boolesche Funktionen und Schaltnetze (cont.)
  - ...
  - Strukturierter Schaltnetz-Entwurf (insbes. Multiplexer)
  - Optimierung von Schaltnetzen
    - KV-Diagramme: Minimalpolynome
    - Algorithmus von Quine und McCluskey: PI-Tafel
  - Unvollständig definierte Funktionen
  - Hazards (insbes. Schaltungshazards)
  - Programmierbare Bausteine: PLAs
- Sequenzielle Schaltungen
  - Modellierung mit Automaten
  - Synchrone Schaltwerke: Flip-Flops
  - Schaltwerk-Entwurf: von Neumann-Addierwerk
  - Speicher (insbes. Adressierung, Demultiplexer)
  - Register: Schieberegister, Datenbus
  - Taktung von Digitalrechnern

Anhang

Schaltnetzentwurf: Begriffe

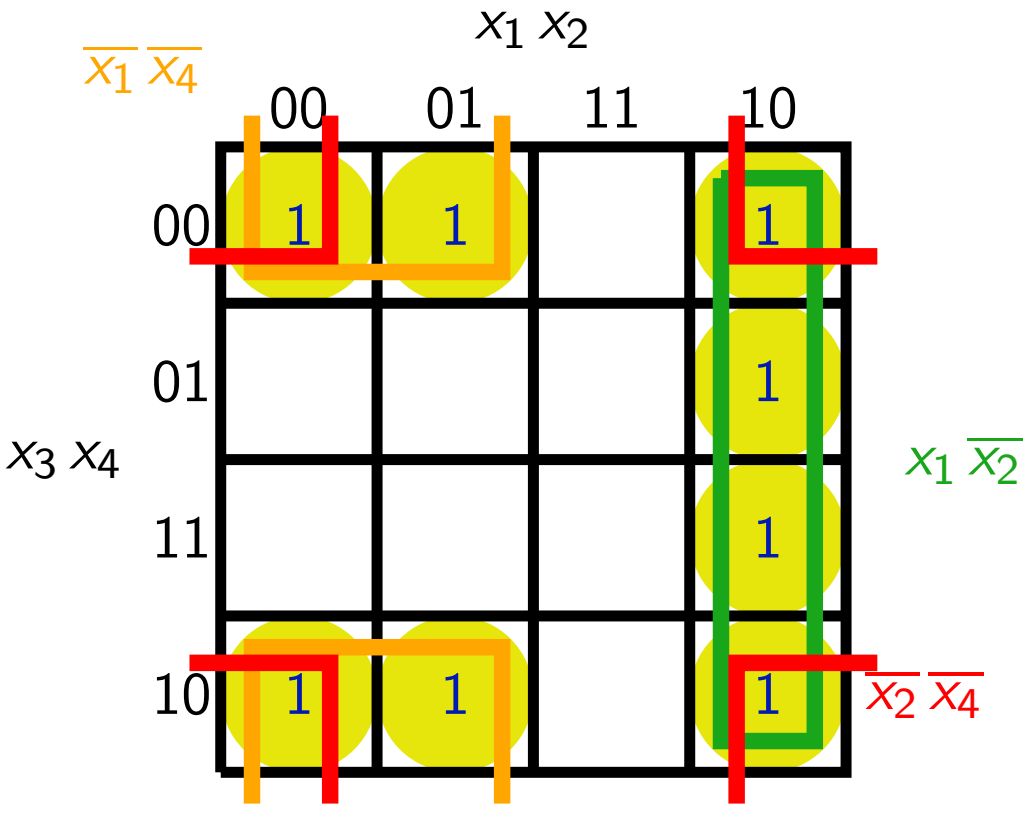
Begriff	Definition	Beispiele
Variable		x_1, x_7, x_{42}
Literal	(negierte) Variable	$x_1, \overline{x_4}, x_9$
Monom	Konjunktion von Literalen	$x_2\,\overline{x_7}, x_3\,x_4, x_{42}$
Polyonom	Disjunktion von Monomen	$x_2\,\overline{x_7} \vee x_3\,x_4, x_{42}$
Implikant	„Implikant(x)=1 $\Rightarrow f(x) = 1$“	Monom eines Polynoms
Verkürzung	„enthält Teilmenge der Literale“	$x_3\,x_4$ von $x_3\,x_4\,x_7$, x_1 von x_1
echte Verkürzung	„enthält echte Teilmenge der L.“	$x_3\,x_4$ von $x_3\,x_4\,x_7$
Prim-implikant	Implikant, keine echte Verkürzung auch Implikant	
Minimal-polynom	Polynom mit minimalen Kosten	

Erinnerung: Minimalpolynome enthalten nur Primimplikanten

Erinnerung: zweischrittiges Vorgehen

1. Berechne Menge aller PI.
2. Berechne minimale überdeckende Auswahl.

Primimplikantenbestimmung mit KV-Diagramm



Minimalpolynom $x_1\,\overline{x_2} \vee \overline{x_1}\,\overline{x_4}$

Algorithmus von Quine/McCluskey: PI-Tafel erstellen

bekannt $f : \{0,1\}^4 \to \{0,1\}$

$x_1\ x_2\ x_3\ x_4$	$f(x_1, x_2, x_3, x_4)$
0000	0
0001	0
0010	0
0011	1
0100	0
0101	0
0110	0
0111	0
1000	0
1001	0
1010	0
1011	1
1100	0
1101	0
1110	0
1111	1

mit $\mathrm{PI} = \{x_1\ x_3\ x_4, \overline{x_2}\ x_3\ x_4\}$

PI-Tafel

	0011	1011	1111
$x_1\ x_3\ x_4$		1	1
$\overline{x_2}\ x_3\ x_4$	1	1	

Algorithmus von Quine/McCluskey: PI-Tafel verkleinern

$x_1\ x_2\ x_4$												1	1		1	1
$x_1\ x_3\ x_4$									1	1					1	1
$\overline{x_2}\ x_3\ x_4$			1	1					1	1						
$x_1\ x_2\ x_5$											1		1	1		1
$x_1\ x_3\ x_5$								1		1				1		1
$\overline{x_2}\ x_3\ x_5$		1		1				1		1						
$\overline{x_4}\ x_5$	1	1			1	1	1	1			1			1		

- Streichung von Kernimplikanten-Zeilen
  Spalte s mit nur einer 1: Wähle Primimplikant der korrespondierenden Zeile, streiche diese Zeile und alle von ihr überdeckten Spalten.
- Streichung überdeckender Spalten
  Spalten s, s' mit $s \geq s'$: Streiche Spalte s.
- Streichung überdeckter Zeilen
  Zeilen z, z' mit $z \geq z'$: Streiche Zeile z'.

IEEE 754-1985 — Addition

x		1	1000	0110	010	1001	0000	0000	0000	0000
y	+	0	1000	0101	111	0000	0000	0000	0000	0000

Zahl mit größerem Exponenten ist x

also vorläufiges Vorzeichenbit 1
vorläufiger Exponent 1000 0110

Differenz der Exp. 1, also in y Komma um 1 Stelle nach links

aus x		1	,	010	1001	0000	0000	0000	0000
aus y	+	0	,	111	1000	0000	0000	0000	0000

unterschiedliche Vorzeichen, also Zweierkomplement

aus x		01	,	010	1001	0000	0000	0000	0000
aus y	+	11	,	000	1000	0000	0000	0000	0000
		1	00	,	011	0001	0000	0000	0000	0000

zum Normalisieren Komma 2 Stellen nach rechts, also Exp. -2

1 1000 0100 100 0100 0000 0000 0000 0000

Ausblick

Ab Montag, den 25.11.2012: *Peter Marwedel*