Technische Universität Braunschweig

Informatik für Ingenieure – VL 6

Übersicht

Letztes Mal:

Einführung zu MOSFETS

Einführung zu CMOS Logik

Schaltnetzrealisierung

Heute:

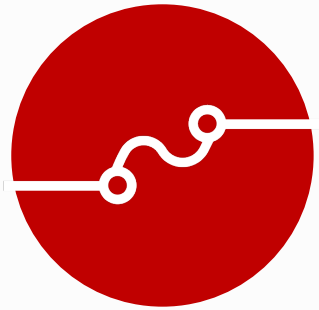
Programmierbare Logik

Einführung zu Schaltwerke

Teile des heutigen Vortrags basiert auf der Vorlesungen von
Prof. H Michalik (TU Braunschweig)
Prof. M. Luisier (ETH Zurich)

und die folgende Bücher
Fundamentals of Digital Logic with VHDL Design, von Brown, Vranesic
Digital Design and Computer Architecture, von Harris, Harris

Programmierbare Logik



Programmable Logic Devices (PLDs)

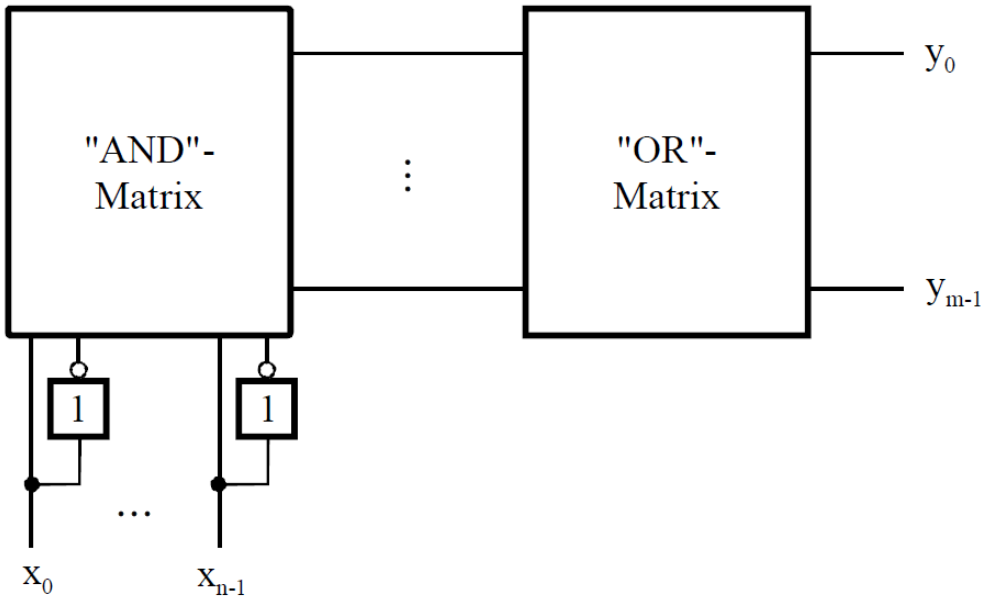
Ein PLD ist ein Allzweck-Chip für die Implementierung logischer Schaltungen.

Er enthält eine Sammlung von logischen Schaltungselementen, die auf verschiedene Weise angepasst werden können.

Ein PLD kann als eine "Blackbox" betrachtet werden, die logische Gatter und programmierbare Schalter enthält

Programmierbare Grundstrukturen

Die Basisrealisierung der DNF in 2-stufiger AND/OR/NOT Logik mit einem Netz aus AND/NOT Verknüpfungen in der ersten Stufe und OR-Verknüpfungen in der zweiten Stufe lassen sich wie im Bild unten dargestellt als "generischer" Aufbau von Schaltnetzen darstellen.

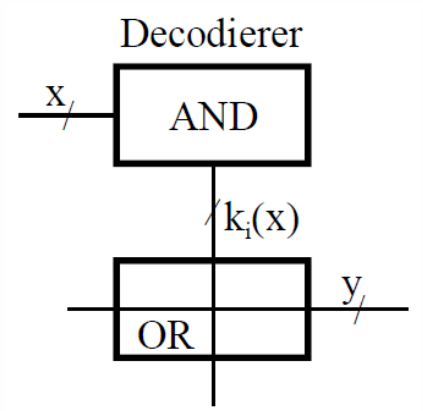


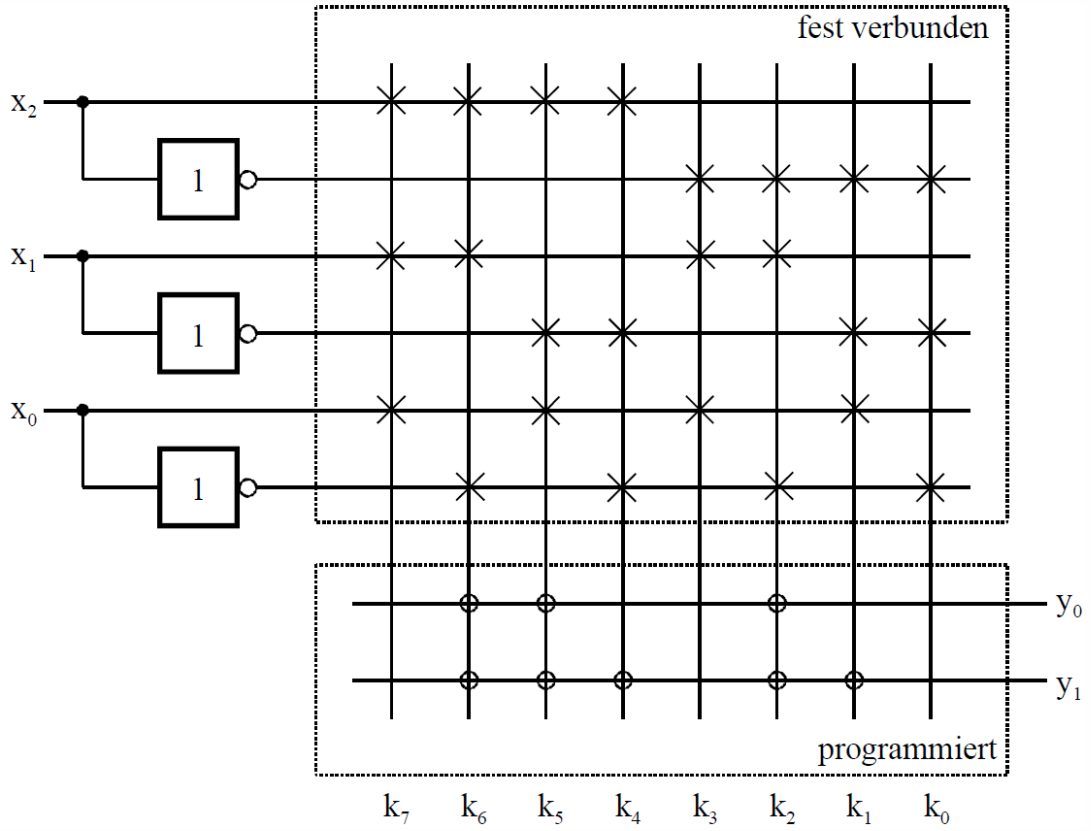
Je nachdem welche der Elemente programmierbar ausgeführt sind ergeben sich drei Typen von Standardrealisierungen

Speicher (ROM) -Realisierung

Das Bild zeigt eine Realisierung mit fester UND-Matrix (vollständig ausgeführt ˆ= Decodierer) in Form von verdrahteter Logik. Hier wird durch den Decodierer jeweils eine Spalte der ODER-Matrix durch Anlegen einer Eingangskombination x_{n-1} ... x_0 ausgewählt.

Die Zeilen der programmierbaren OR-Matrix entsprechen den jeweiligen Schaltfunktionen y_{m-1} ... y_0. Das ROM realisiert also die vollständige Wahrheitstabelle.

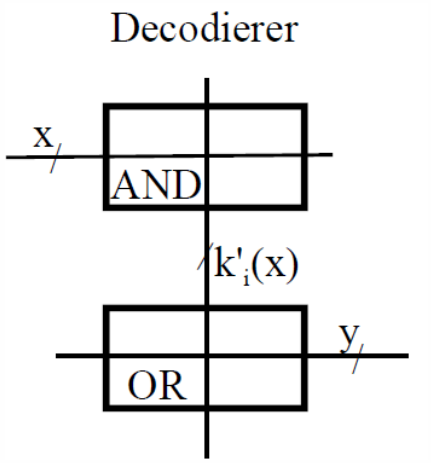


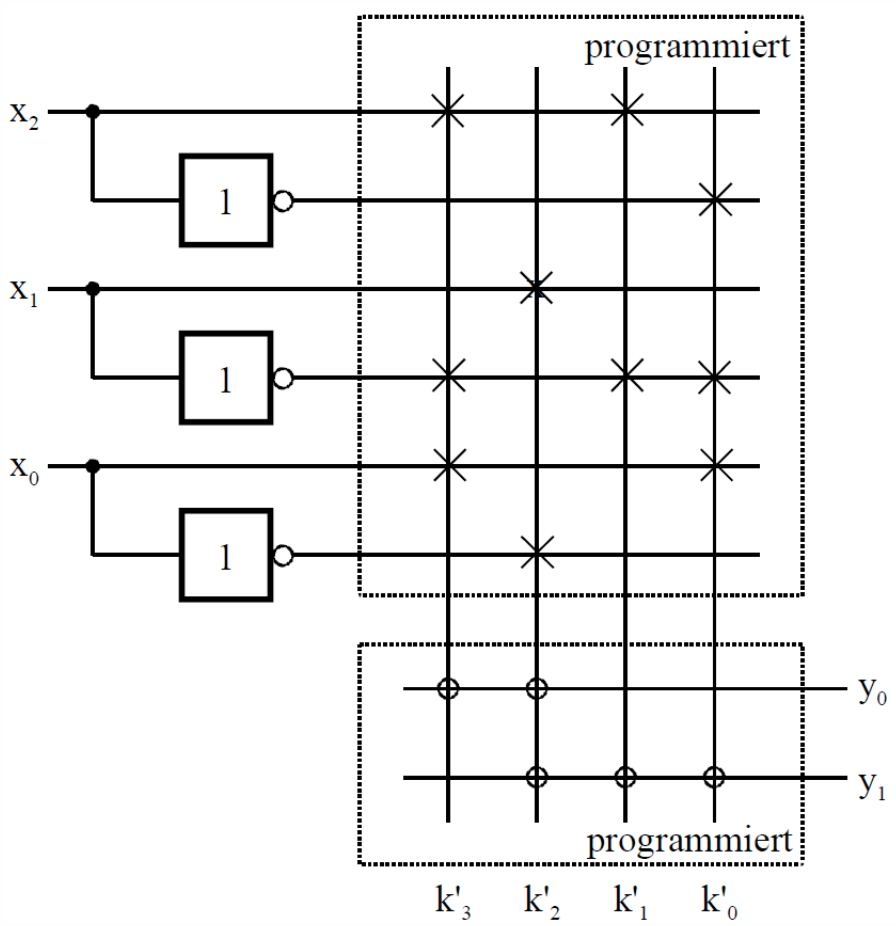


PLA -Realisierung

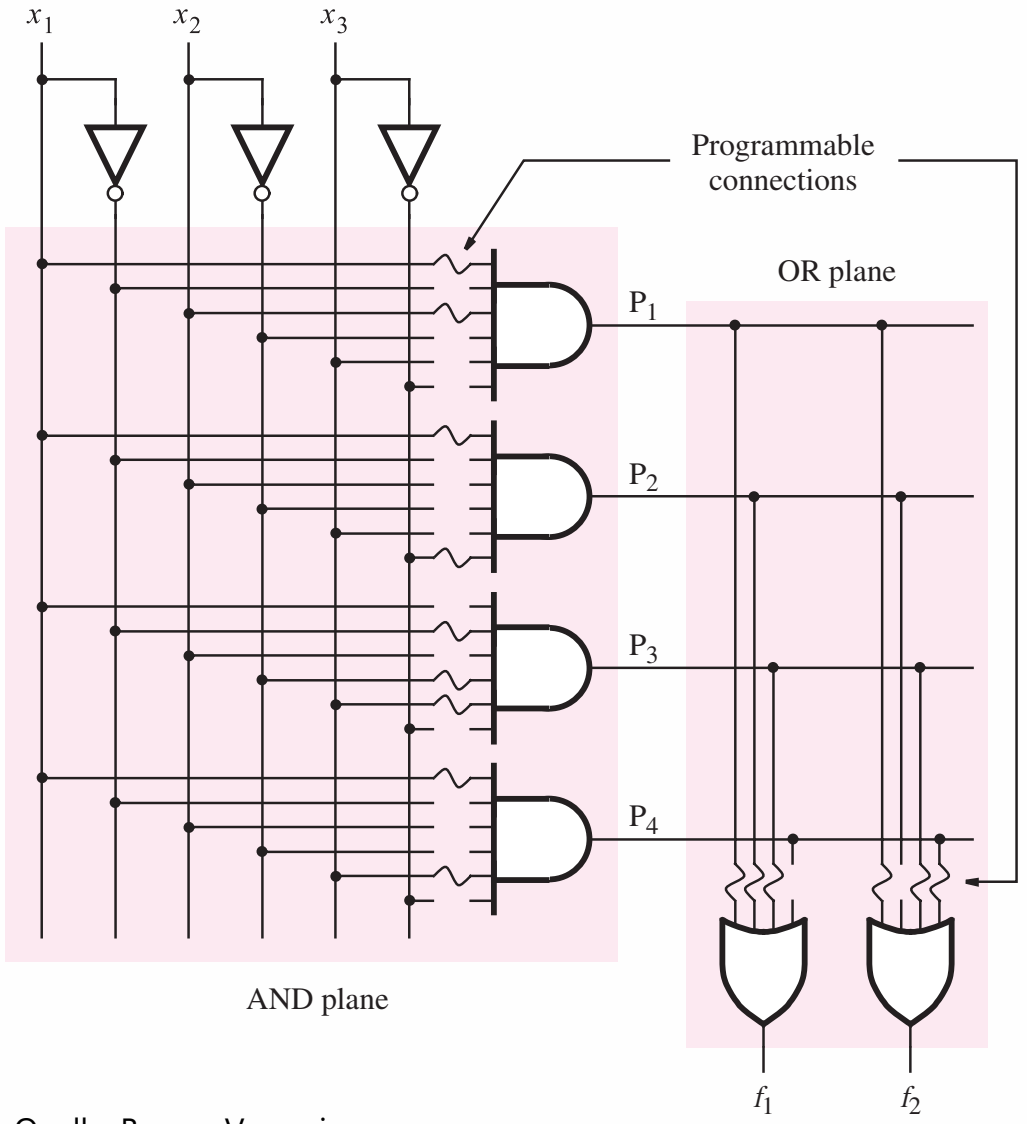
Wenn beide Matrizen programmierbar sind, spricht man von einem PLA (Programmable Logic Array)

Dies entspricht den vorher betrachteten DNF-Realisierungen, bei denen nur diejenigen Konjunktionen der Eingänge realisiert und disjunktiv verknüpft werden, die eine "1" in der Schaltfunktion liefern





PLA Schaltung



Quelle: Brown, Vranesic

programmierbare Schalter hatten zwei Probleme:

- Schwierig korrekt herzustellen
- Verringerte Geschwindigkeit/Leistung

Lösung:

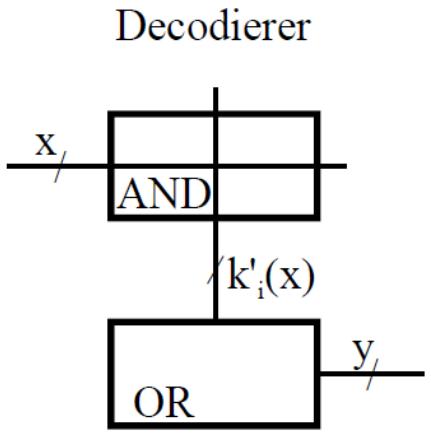
- die OR-Ebene festlegen
  Weniger programmierbare Schalter
- → Schneller, billiger Schalter

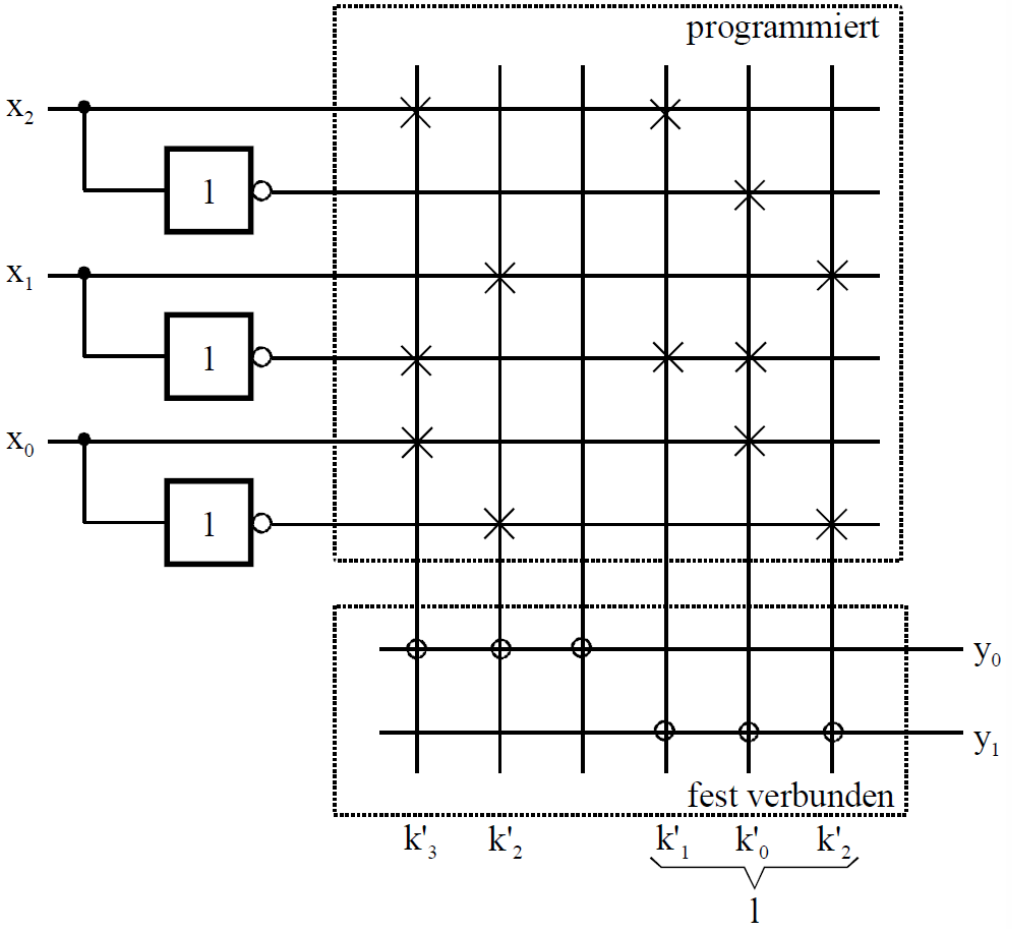
PAL-Realisierung

Wenn nur die AND-Matrix programmierbar ist, spricht man von einem PAL (Programmable Array Logic).

Bei vielen Schaltungen ist die Zahl der wirklich benötigten Disjunktionen pro Ausgangsfunktion begrenzt.

Deshalb stellt das PAL eine "vorkonfektionierte„ OR-Matrix zur Verfügung.



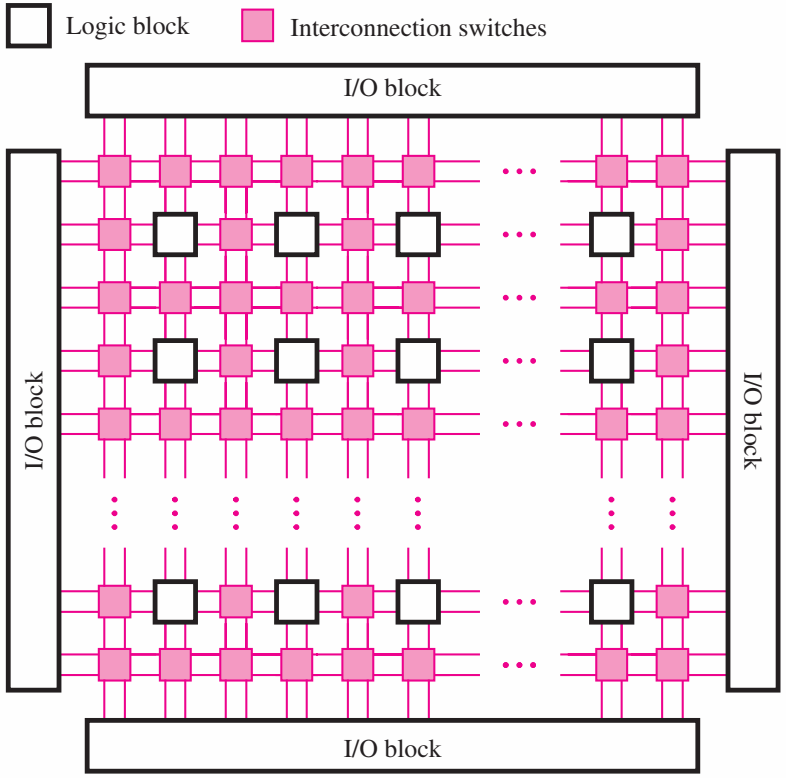


Field Programmable Gate Array (FPGA)

Konfigurierbare Hardware-Plattform

Flexibel und umprogrammierbar

Ideal für Parallelverarbeitung und schnelles Prototyping



Quelle: Brown, Vranesic

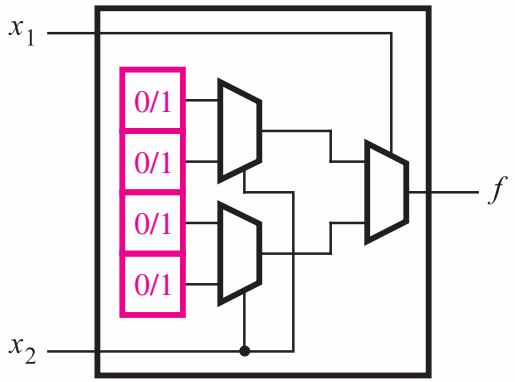
Look-Up Tables

Look-Up-Tables (LUTs) sind grundlegende Bausteine in FPGAs.

Sie werden verwendet, um kombinatorische Logikfunktionen umzusetzen.

LUTs sind speicherbasierte Komponenten, die Wahrheitstabellen oder funktional äquivalente Daten speichern.

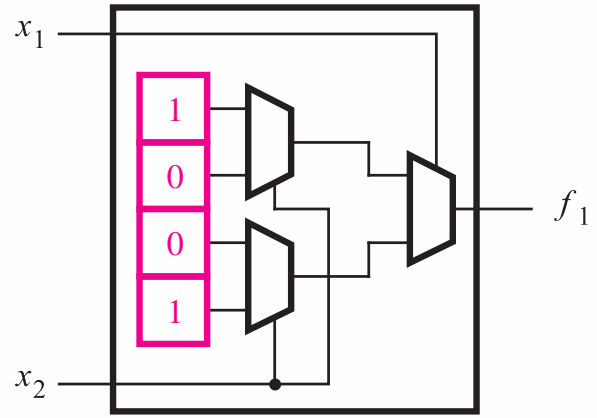
2 Bit LUT:



Beispiel Funktion:

x_1	x_2	f_1
0	0	1
0	1	0
1	0	0
1	1	1

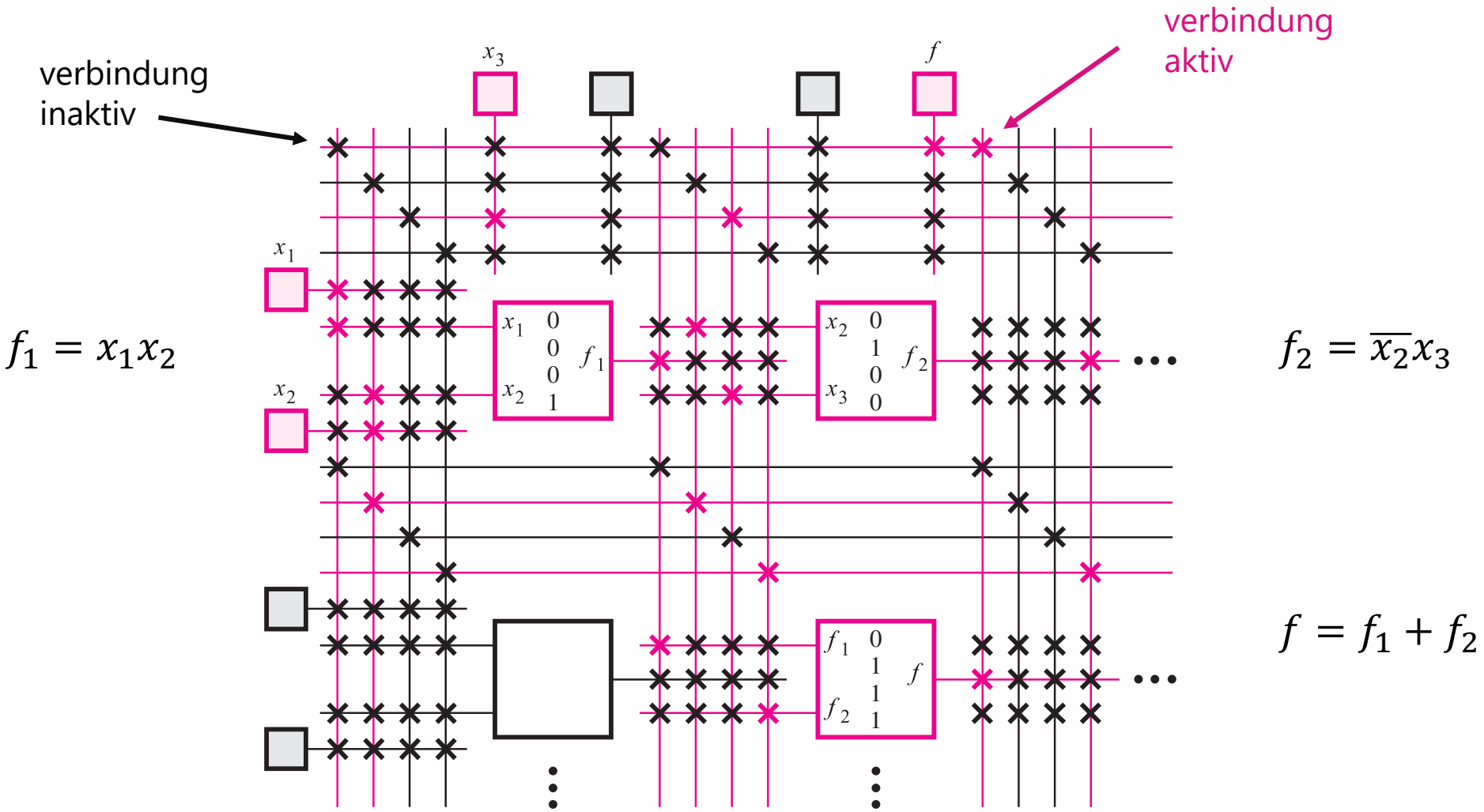
$$f_1 = \bar{x}_1\bar{x}_2 + x_1x_2$$



Quelle: Brown, Vranesic

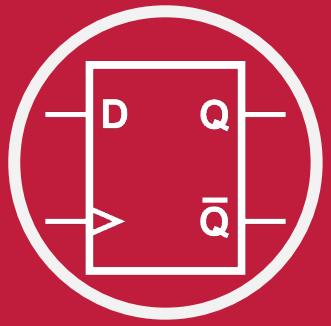
Einfaches FPGA Beispiel

Wenn das FPGA programmiert wird, müssen sowohl die LUT-Tabellen als auch die Verdrahtungsschalter konfiguriert werden



Quelle: Brown, Vranesic

3. ENTWURF UND ANALYSE SEQUENTIELLER SCHALTUNGEN



Kombinatorische vs. Sequentielle Schaltungen

Kombinatorische Schaltungen: die Ausgangsvariablen hängen nur von dem Verknüpfungsnetzwerk und von den Eingangsgrössen ab. Eine gegebene Eingangsbelegung X_i zur Zeit t_n erzeugt stets dieselben Ausgangswerte Y_j, unabhängig von der Vorgeschichte, d.h. von der Zeit t_{n+1}:

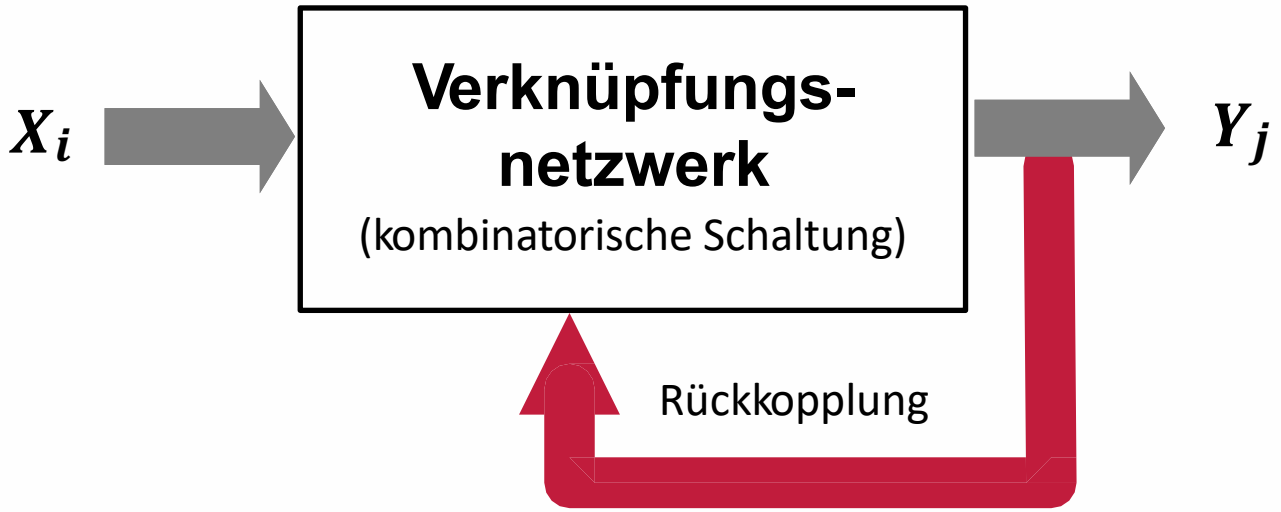
$$Y_j\,(t_n) = f(X_i(t_n))$$

Bis jetzt haben wir nur **kombinatorische Schaltungen** betrachtet (Grundgatter, Addierer, usw.)

Kombinatorische vs. Sequentielle Schaltungen

Sequentielle Schaltungen: im Gegensatz zu ihren kom- binatorischen Pendants, besitzen die sequentiellen Schaltungen eine oder mehrere Rückkopplungen, so dass ihre Ausgangswerte Y_j zur Zeit t_n auch von voran- gegangenen Werten bei t_{n+1} abhängen:

$$Y_j(t_n) = f(\, X_i(t_n)\, ,\, Y_j(t_{n+1})\,)$$



Einleitung

Ausgänge **sequentieller** Logik hängen ab von

- **aktuellen** Eingabewerten
- **vorherigen** Eingabewerten

Schaltung **speicher**t einen internen **Zustand**

Definitionen

- **Zustand**: interne Informationen, aus denen weiteres Schaltungsverhalten hergeleitet werden kann
- **Latches und Flip-Flops:** Speicherelemente für jeweils 1 Bit Zustand
- **Synchrone sequentielle Schaltung**: Kombinatorische Logik gefolgt von Flip-Flops

Zustandselemente

Der Zustand einer Schaltung beeinflusst das zukünftige Verhalten

Speicherelemente speichern Zustand:

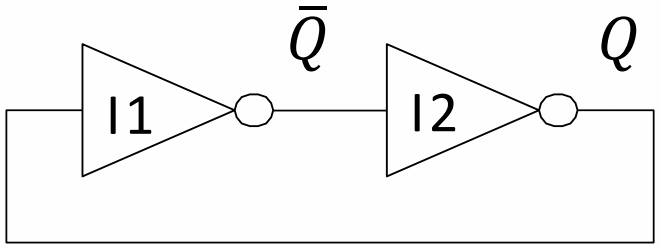
- Bistabile Schaltungen
- SR und D Latches
- SR, D, T, und JK Flip-Flop

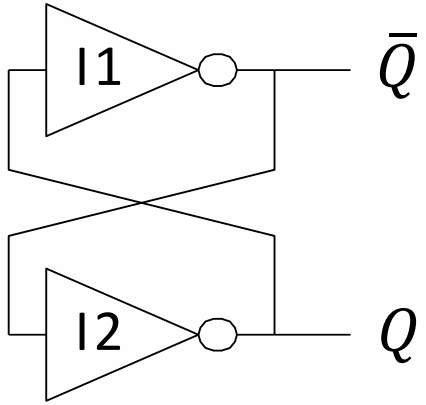
Bistabile Grundschaltung

Fundamentaler Baustein der anderen Speicherelemente

Zwei Ausgänge: $Q, \bar{Q}$

Keine Eingänge

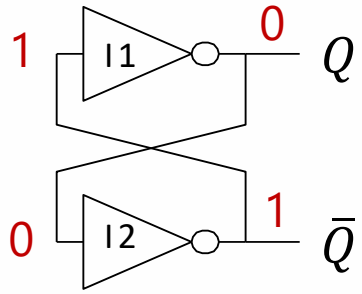




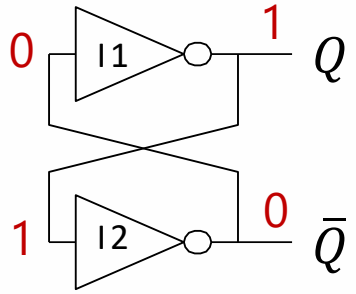
Analyse der bistabilen Grundschaltung

Betrachte zwei Möglichkeiten:

Q = 0: dann Q = 1 und Q = 0
Konsistent und stabil



Q = 1: dann Q = 0 und Q = 1
Konsistent und stabil



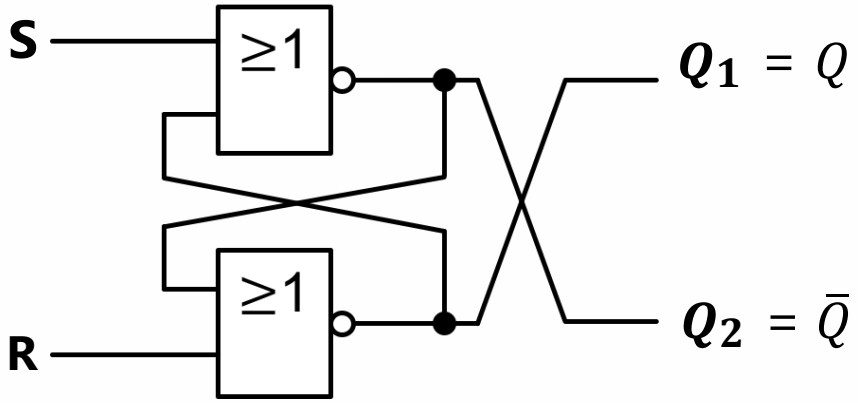
Bistabile Schaltung speichert 1 Zustandsbit in Zustandsvariable Q (oder Q)

Es gibt aber bisher keine Eingänge, um diesen Zustand zu beeinflussen

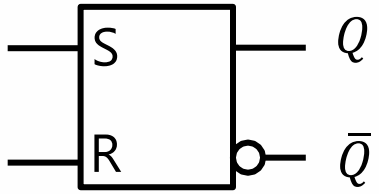
Das SR-Latch: Grundlagen

Das **Basis-SR-Latch** besitzt zwei Eingänge **S** (set) und **R** (reset), sowie in der Regel zwei Ausgänge Q und $\bar{Q}$. Es besteht aus zwei NOR-Gattern, die durch zwei Rück- Kopplungen ein Speicherelement aufbauen

Verknüpfungsnetzwerk



Schaltzeichen



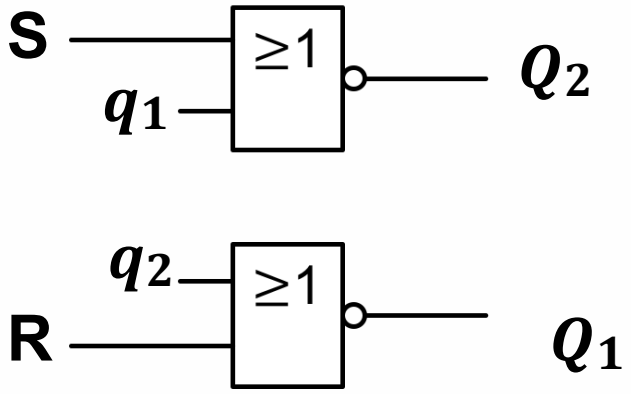
Wie analysiert man eine Schaltung mit Rückkopplung?

Die Rückkopplungen werden aufgetrennt und das Verhalten der offenen Schleife (‚open loop') wird bestimmt

Die Rückkopplungen werden miteinbezogen

Schaltungsanalyse ohne Rückkopplungen

SR-Latch ohne Rückkopplungen



Es gelten die folgenden Gleichungen

$$Q_1 = \overline{R \vee q_2} = \overline{R} \wedge \overline{q_2} \quad \text{und}$$
$$Q_2 = \overline{S \vee q_1} = \overline{S} \wedge \overline{q_1}$$

Die Wertetabellen für Q_1 und Q_2 werden in ein doppeltes KV-Diagramm für Minterme eingesetzt

Wertekombinationen, wo $Q_1 = q_1$ und $Q_2 = q_2$

SR / q_1q_2	00	01	11	10
00	11	01	00	10
01	01	01	00	00
11	00	00	00	00
10	10	00	00	10

Q_1Q_2

Schaltungsanalyse mit Rückkopplungen

Die beiden Rückkopplungen verlangen $\boldsymbol{Q_1 = q_1}$ und $\boldsymbol{Q_2 = q_2}$

Fünf Stellen im KV-Diagramm erfüllen diese Bedingungen

	Fall	S	R	Q_1	Q_2	
Setzzustand (**SZ**) →	1	1	0	1	0	
	2	0	0	1	0	$Q_1Q_2 = 10$, unverändert
Rücksetz-zustand (**RSZ**) →	3	0	1	0	1	
	4	0	0	0	1	$Q_1Q_2 = 01$, unverändert
	5	1	1	0	0	

Falls **S=1** und **R=0**, so wird $Q_1 = 1$ gesetzt und $Q_2 = 0$ (**SZ**)

Falls **S=0** und **R=1**, so wird $Q_1 = 0$ gelöscht und $Q_2 = 1$ (**RSZ**)

Im Fall **S=R=0** ändern sich die Ausgangssignale Q_1 und Q_2 nicht. Sie behalten ihren vorherigen Wert

Die Kombination **S=R=1** ist nicht zulässig, da immer $Q_1 = Q_2$ gelten muss.
Sonst ist das Latch in einem irregulären Zustand

Dynamisches Verhalten des SR-Latches

Anfangspunkt: $S = 1$, $R = 0$, $Q_1 = 1$ und $Q_2 = 0$

Anfangszustand

Endzustand

Änderung am Eingang:
$S\ (1 \rightarrow 0)$, $R\ (0 \rightarrow 0)$

q_1q_2 \ SR	00	01	11	10
00	11	01	00	10
01	01	01	00	00
11	00	00	00	00
10	10	00	00	10

Änderung am Eingang:
$S\ (1 \rightarrow 0)$, $R\ (0 \rightarrow 1)$

q_1q_2 \ SR	00	01	11	10
00	11	01	00	10
01	01	01	00	00
11	00	00	00	00
10	10	00	00	10

Die aktuellen Werte von Q_1 und Q_2 sind die nächsten Werte von q_1 und q_2. Diese beiden Variablen beeinflussen den nächsten Wert von Q_1 und Q_2 unter Berücksichtigung von S und R. Die Prozedur läuft weiter, bis $(Q_1,Q_2)=(q_1,q_2)$.

Beschreibung der Zustandsfolge

Das Verhalten eines Latches (SR und anderen) hängt nicht nur von den aktuellen Eingangsvariblen ab sondern auch von den **intern gespeicherten Zuständen** → **zustandsgesteuert**

Definitionen:

Der Zeitpunkt **vor** (**nach**) einem Zustands- oder Taktwechsel wird als t_n (t_{n+1}) gekennzeichtnet

Analog dazu bezeichnet $Q_{1,n}$ ($Q_{1,n+1}$) den Wert des Ausgangs Q_1 vor (nach) einem Wechsel

Zustandsfolge- (Folgezustands-) Tabelle des SR-Latches

Fall	S	R	$Q_{1,n+1}$	$Q_{2,n+1}$	
1	0	0	$Q_{1,n}$	$Q_{2,n}$	speichern
2	0	1	0	1	rücksetzen
3	1	0	1	0	setzen
4	1	1	-	-	unzulässig

Gleichung für $Q_{1,n+1}$

Um die **Gleichung** für $Q_{1,n+1}$ zu bestimmen, muss zuerst die komplette **Wertetabelle** für $Q_{1,n+1}$ erzeugt werden

Fall	S	R	$Q_{1,n}$	$Q_{1,n+1}$	
1	0	0	0	0	speichern
2	0	0	1	1	
3	0	1	0	0	rücksetzen
4	0	1	1	0	
5	1	0	0	1	setzen
6	1	0	1	1	
7	1	1	0	X	unzulässig
8	1	1	1	X	

Als nächster Schritt müssen diese Ergebnisse in ein KV- Diagramm eingetragen werden, um die DNF der Gleichung für $Q_{1,n+1}$ aufwandslos abzulesen

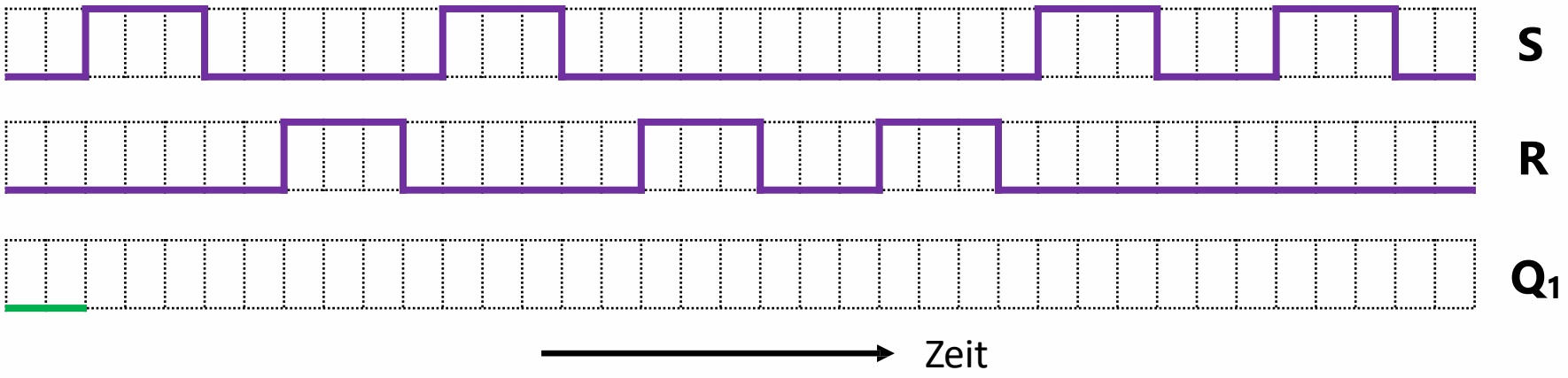
Vereinfachter Ausdruck für $Q_{1,n+1}$

KV-Diagramm für $Q_{1,n+1}$ als Funktion von S, R und $Q_{1,n}$

Q_{1n} \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

Damit finden wir $Q_{1,n+1} = S \vee (Q_{1,n} \wedge \bar{R})$ mit $R \wedge S = 0$

Beispiel: Zeitverhalten eines SR-Latches



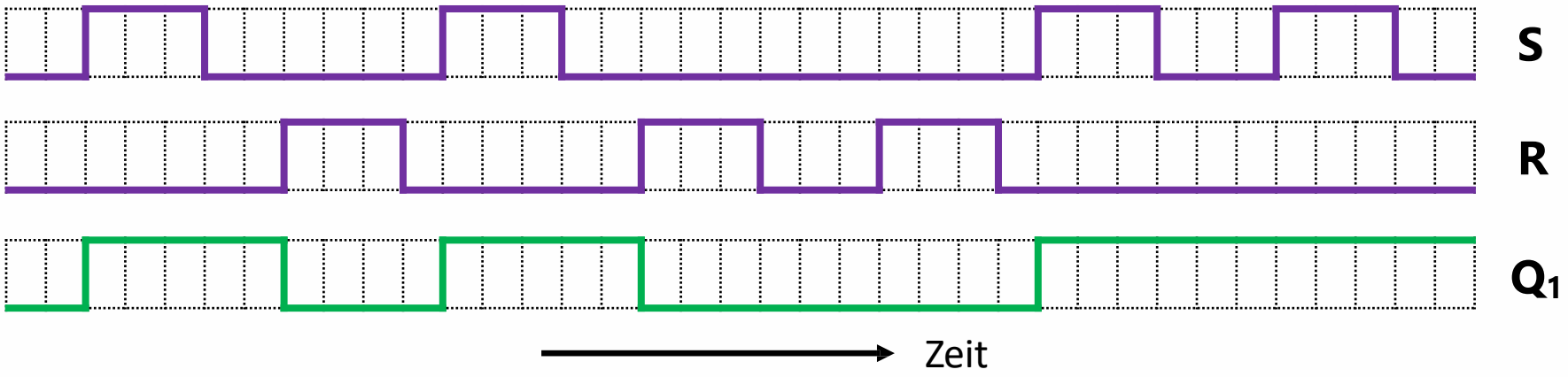
Vereinfachter Ausdruck für $Q_{1,n+1}$

KV-Diagramm für $Q_{1,n+1}$ als Funktion von S, R und $Q_{1,n}$

Q_{1n} \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

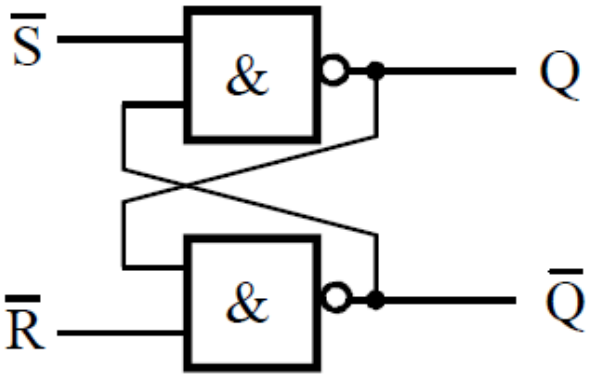
Damit finden wir $Q_{1,n+1} = S \vee (Q_{1,n} \wedge \bar{R})$ mit $R \wedge S = 0$

Beispiel: Zeitverhalten eines SR-Latches

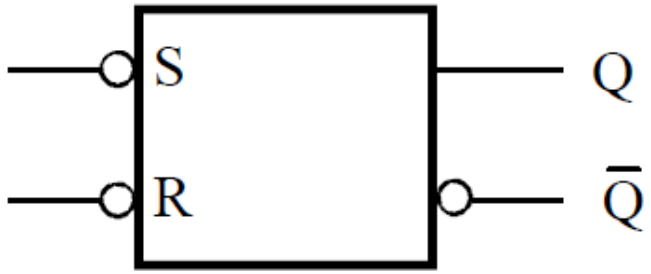


SR-Latch aus NAND Gattern

Verknüpfungsnetzwerk



Schaltzeichen



Wahrheitstabelle

S	R	Q	$\bar{Q}$	
1	1	Q_{old}	$\bar{Q}_{\text{old}}$	speichern
1	0	0	1	rücksetzen
0	1	1	0	setzen
0	0	1	1	nicht erlaubt

Getakte Latches

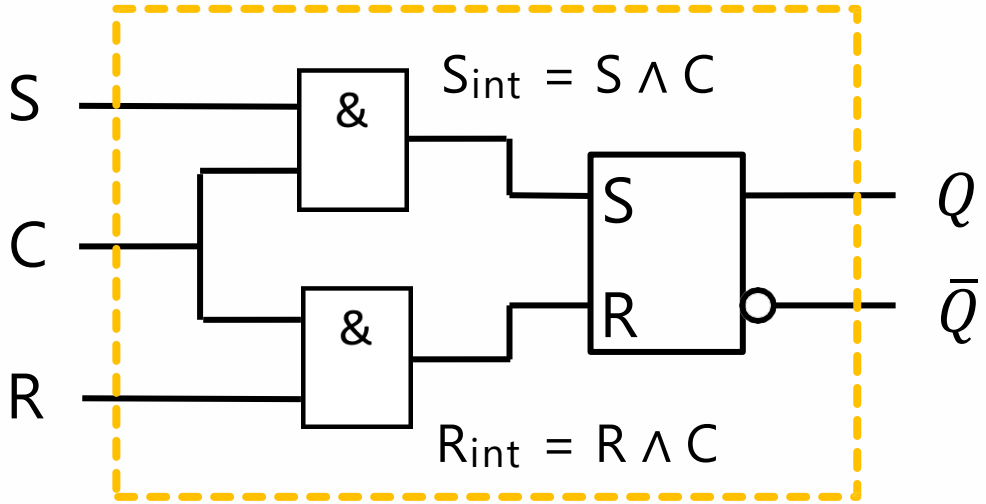
Taktzustandsgesteuertes SR-Latch

Bei dem **Basis-SR-Latch** wird ein Eingangssignal sofort und immer am Ausgang wirksam (zustandsgesteuert)

Oftmals wird gewünscht, dass Änderungen am Eingang nur in einem **definierten Zeitfenster** zugelassen werden, in Abhängigkeit von einem Taktsignal T, C oder CLK. Eine solche Variante heisst taktzustandsgesteuert

Lösung:

UND-Verknüpfung der SR-Eingänge mit einem **Taktsignal C**



Wenn C=0 → **S_{int}=0** und **R_{int}=0**

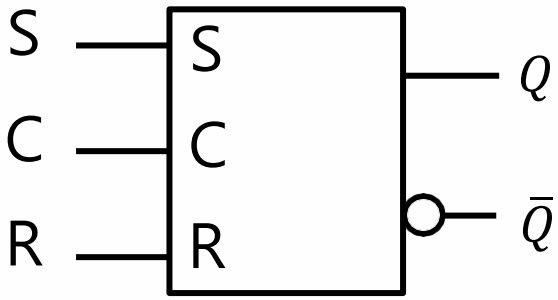
Dataspeicherung

Wenn C=1 → **S_{int}=S** und **R_{int}=R**

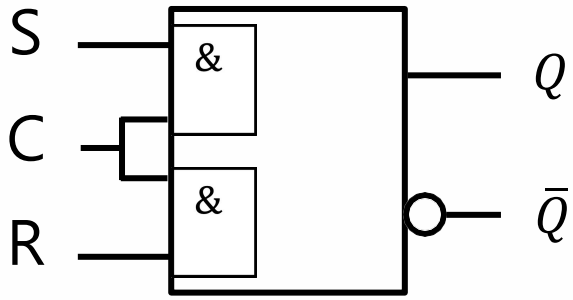
Normales Latch

SR-Latch: Schaltzeichen und Folgezustandstabelle

Schaltzeichen des taktzustandsgesteuerten SR-Latches:



Oder

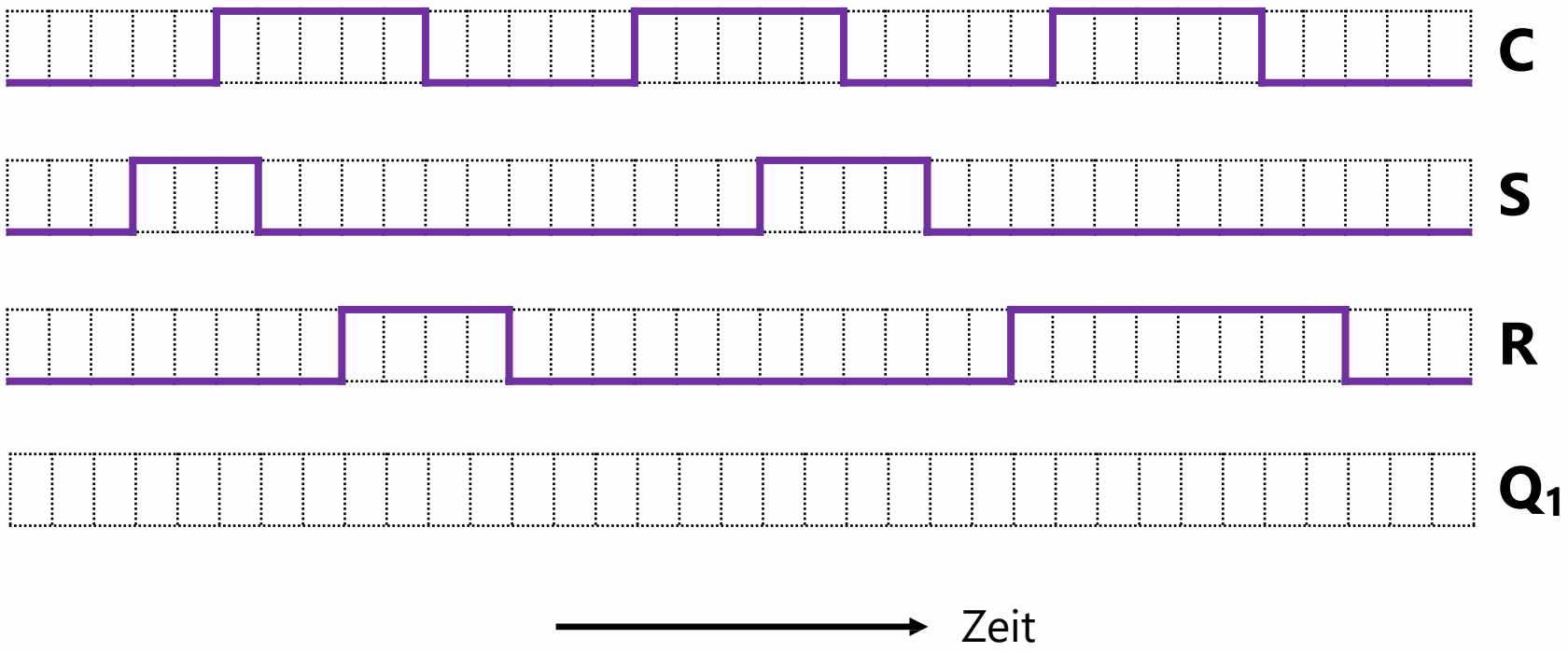


Folgezustandstabelle des SR-Latches:

Fall	C	S	R	Q_{1n+1}	Q_{1n+1}	
1	0	X	X	Q_{1n}	Q_{1n}	keine Änd.
2	1	0	0	Q_{1n}	Q_{1n}	speichern
3	1	0	1	0	1	rücksetzen
4	1	1	0	1	0	setzen
5	1	1	1	-	-	unzulässig

Zeitdiagramm des SR-Latches

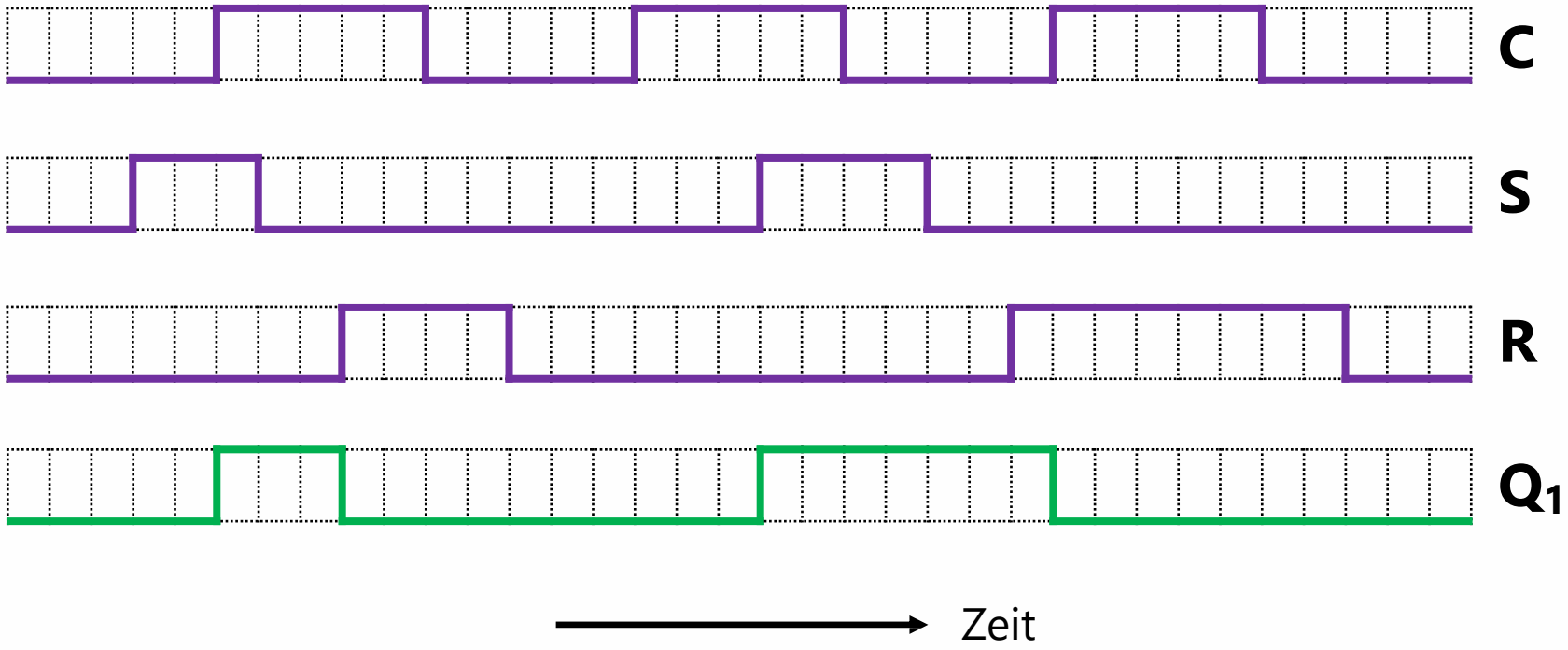
Zeitverhalten eines taktzustandsgesteuerten SR-Latches:



Nehmen Sie sich ein paar Minuten Zeit, um das **Zeitdiagramm** für Q_1 zu vervollständigen!

Zeitdiagramm des SR-Latches

Zeitverhalten eines taktzustandsgesteuerten SR-Latches:

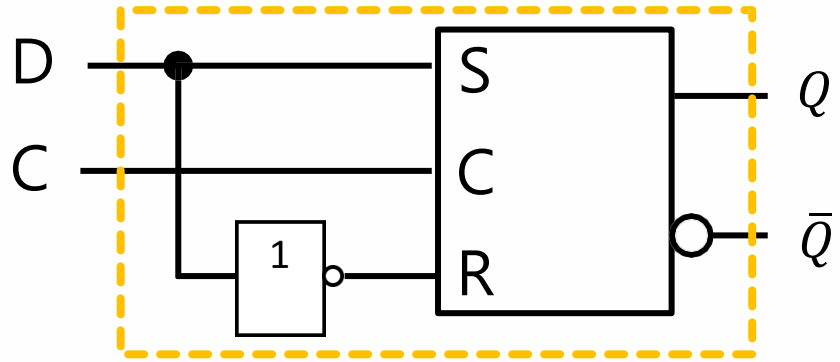


Nehmen Sie sich ein paar Minuten Zeit, um das **Zeitdiagramm** für Q_1 zu vervollständigen!

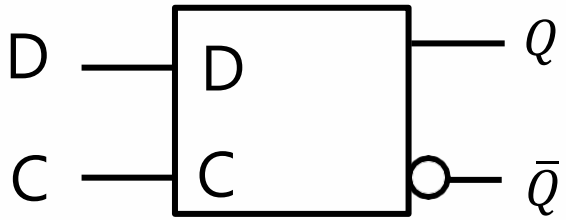
Grundlagen des D-Latches

Gesucht wird ein Bauelement, das Daten speichern kann, idealerweise für die gesamte Periodendauer eines Taktes C

Durch eine einfache Änderung des taktzustandsgesteuerten SR-Latches kann diese Funktionalität *teilweise* erreicht werden



Basisschaltung D-Latch



Schaltzeichen D-Latch

D-Latch: der Rücksetzeingang R ist nicht mehr nach aussen geführt, sondern wird aus dem Setzeingang S durch Invertierung gewonnen

Eigenschaften des D-Latches

Wertetabelle

C	D	$Q_{1,n}$	$Q_{1,n+1}$	
0	X	0	0	$Q_{1,n}$
0	X	1	1	
1	0	X	0	D
1	1	X	1	

$$Q_{1,n+1} = \left(Q_{1,n} \wedge \bar{C}\right) \vee (D \wedge C)$$

Im D-Latch wird durch die Invertierung von S die unzulässige Eingangskombination S=R=1 vermieden

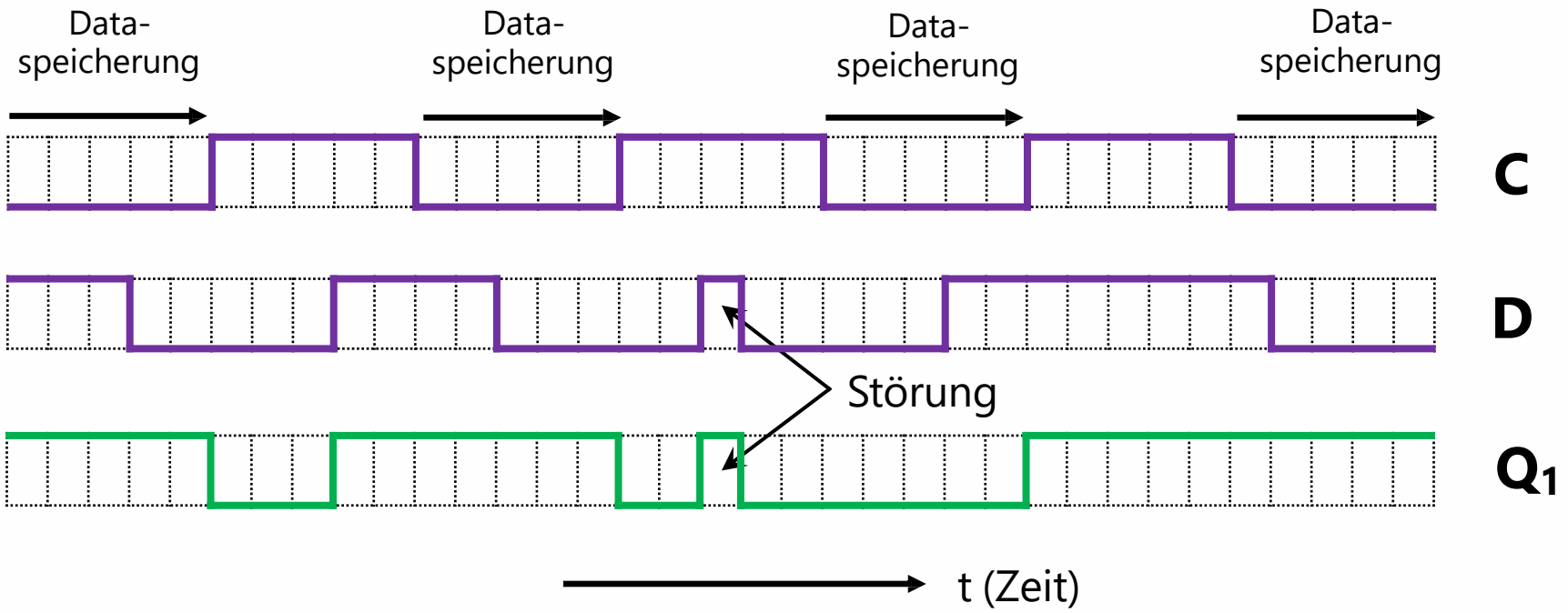
Im aktiven Taktzustand (C=1) ist das D-Latch transparent, also $\boldsymbol{Q_{1,n+1} = D}$ wenn **C=1**

Beim Übergang von C=1 nach C=0 „schnappt“ der Eingang „zu“ (latched), der letzte Eingangswert wird gespeichert

Zeitverhalten des D-Latches

Wie verhält sich das Ausgangssignal $\boldsymbol{Q}$ eines D-Latches als Funktion von D (Eingang), C (Taktsignal) und t (Zeit)?

Beispiel:



D-Latches speichern den Zustand, der beim 'C = 1 → 0' Übergang am Eingang liegt. Wie alle taktzustandsgesteuerten Schaltungen sind sie gegenüber Störimpulsen empfindlich, da bei C=1 jede Änderung am Eingang übernommen wird.