

Dies ist nun also die freundlicherweise von mir mitgetippte Fassung der Vorlesung
Digitaltechnik (2. Semester) bei **Hr. Schillack** an der **BA-Mannheim**.

Ich hoffe ihr könnt damit was anfangen.

Fehler, Kritik, Anregungen und alles was euch sonst noch dazu einfällt mailt bitte an
st_tost@hotmail.com

Inhalte / Gliederung

17 Seiten

2. Semester - Schaltwerke (Schaltnetze mit Speichergliedern bzw. Rückwirkungen)

Flip-Flop's

- Zähler
- Register

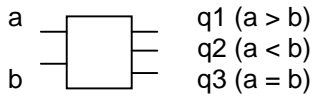
Speicherstrukturen

DA/AD - Wandler

Komparator (Vergleicher)	2
programmierbare Speicher	2
Diode	2
programmierbare Speicher	3
- Dioden-Matrizen	3
- UND- / ODER-Gatter mit Dioden	4
Gatterpegel für TTL-Schaltkreise	4
PROM – Diodenmatrizen	5
Aufbau von Matrizen	5
Signalschalter	6
Diodenmatrixanordnungen	6
- ROM / EPROM	6 / 7
- 4:1 MUX / 1:4 DMUX	7
- Volladdierer	8
Flip-Flop	8
NOR – Flip-Flop	9
Taktflankensteuerung	9
J-K – Flip-Flop	11
D – Flip-Flop	13
T – Flip-Flop	14
Flip-Flop – Schaltungsanalyse und –synthese	15
Übersicht Flip-Flops	15
Schaltungssynthese	16
- Ampel mit Flip-Flops	16
- Funktionen / Schaltplan	16
Schaltungsanalyse	17

Komparatoren > < =

? Schaltung die a und b vergleicht und ausgibt >, < oder = (wenn q 1 ist, ist die Bedingung erfüllt)



Wahrheitstabelle für die Schaltung

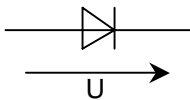
i	a	b	q1	q2	q3	P _i
0	0	0	0	0	1	$\neg a * \neg b$
1	0	1	0	1	0	$\neg a * b$
2	1	0	1	0	0	$a * \neg b$
3	1	1	0	0	1	$a * b$

q1 = a !b
 q2 = !a b
 q3 = !a !b + a b = a ↔ b

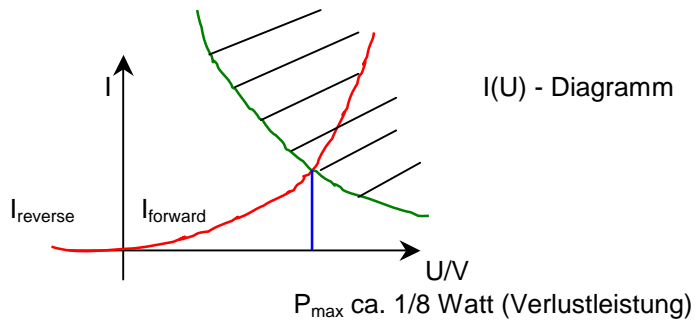
Programmierbare Speicher

- aus Diodenmatrizen in UND bzw. ODER Verknüpfungen

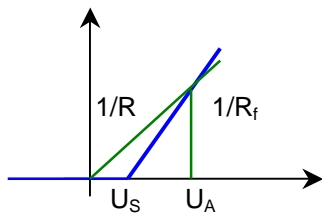
UND aus Dioden



$P = U * I \rightarrow I = P * 1/U$



Ersatzdiode



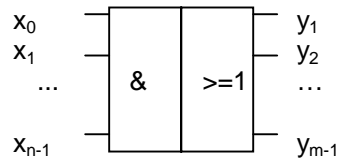
U_S – Schleusenspannung ~ 0,7 V
 U_A - Arbeitsspannung
 R_f - ~50 Ohm

Programmierbare Speicher

Festwertspeicher

PAL – programmable array logic
PLA

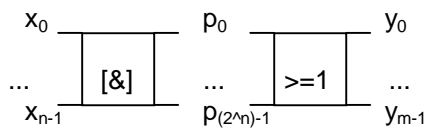
n Eingänge m Ausgänge



PAL [&] dualcodiert

Dioden-Matrizen

- fest verdrahtet bei dual PAL → direkte Abbildung der Funktionstabelle
- bei PLA variabel



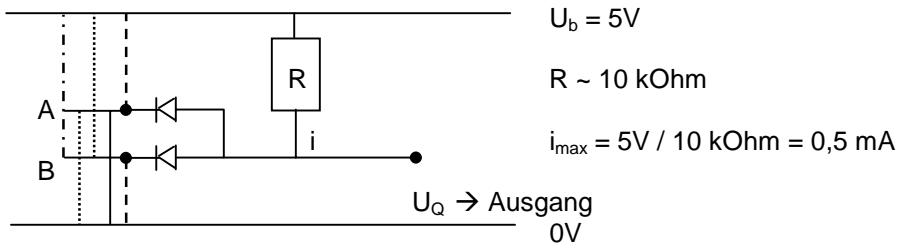
Funktionstabelle für 3 Variablen:

i	X2	X1	X0	Pi
0	0	0	0	!x2 !x1 !x0
1	0	0	1	!x2 !x1 x0
2	0	1	0	!x2 x1 !x0
3	0	1	1	!x2 x1 x0
4	1	0	0	...
5	1	0	1	...
6	
7				

Summe bilden → $y_0 - y_{m-1}$

Aus den Summentermen aus Pi werden die Ausgänge y_0 bis y_{m-1} belegt.

UND - Gatter mit Dioden / TTL-Pegel (5V)

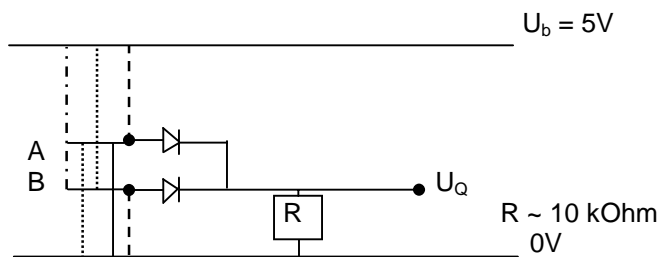


Die Dioden stellen einem Widerstand dar \rightarrow pro Diode 5 Ohm (bei einem Stromfluß von $i_{\max} = 0,5\text{mA}$)
 Also fallen an einer Diode 0,7V ab, wenn beide parallel geschaltet sind 0,35V.

	i	U_A	U_B	U_Q
———	0	0	0	0,35V
· - - -	1	5V	0	0,7V
· · · · ·	2	0	5V	0,7V
· - · - ·	3	5V	5V	5V

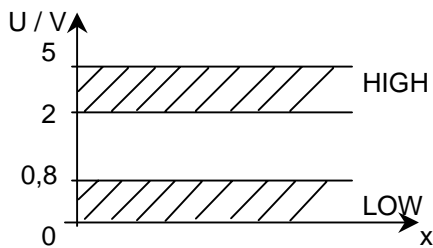
U_Q ist die Spannung, die zwischen dem Ausgang und $U=0$ abgegriffen werden kann.

ODER - Gatter mit Dioden / TTL-Pegel (5V)

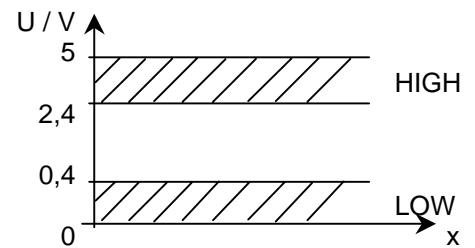


	i	U_A	U_B	U_Q
———	0	0	0	0V
· - - -	1	5V	0	4,3V
· · · · ·	2	0	5V	4,3V
· - · - ·	3	5V	5V	4,65V

Gattereingangspegel für TTL-Schaltkreise

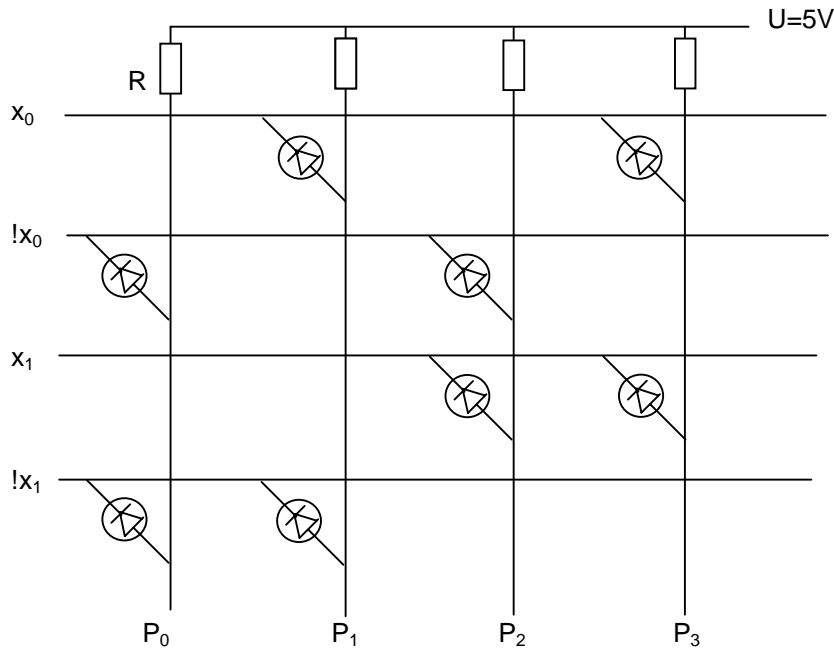


Gatterausgangspegel für TTL-Schaltkreise



PROM – Programmable-Read-Only-Memory

AND-Matrix



i	x ₁	x ₀	p _i	P _i
0	0	0	!x ₁ !x ₀	0
1	0	1	!x ₁ x ₀	0
2	1	0	x ₁ !x ₀	0
3	1	1	x ₁ x ₀	1

Um diese Funktionstabelle mit dieser Matrix darstellen zu können, werden nur die eingezeichneten Dioden benötigt. Die anderen werden weggelassen.

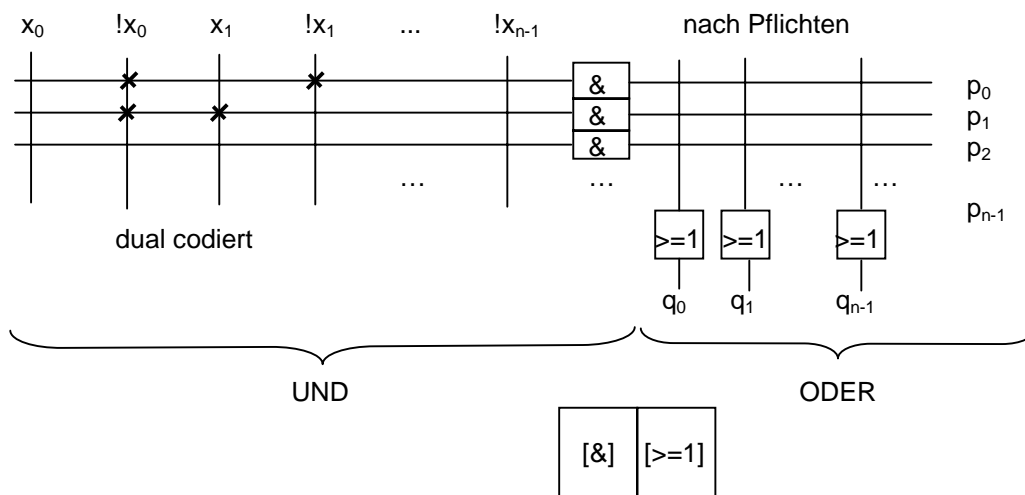
Am Ausgang liegt dann ,1' an, wenn beide Eingänge denselben Wert haben.

Beispiel für p₂: x₁ = 1 und !x₀ = 1

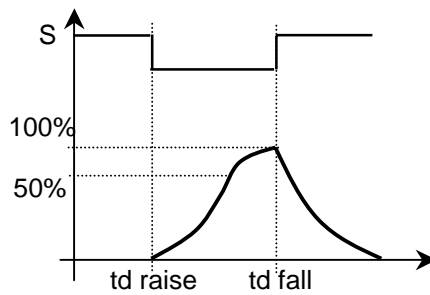
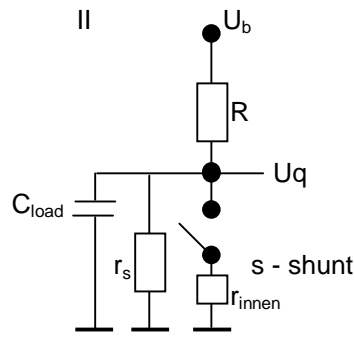
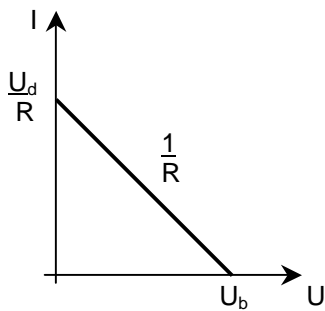
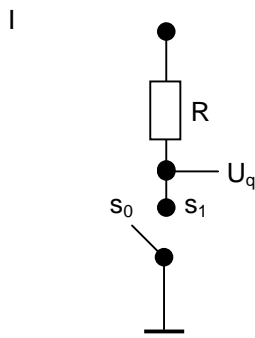
Der Strom fließt durch den Widerstand R zur Diode. Da an !x₀ ,1' anliegt, kann der Strom nicht über die Diode fließen; dasselbe an der nächsten Dioden bei x₁. Also kommt an P₂ ein Strom an.

Aufbau von Matrizen

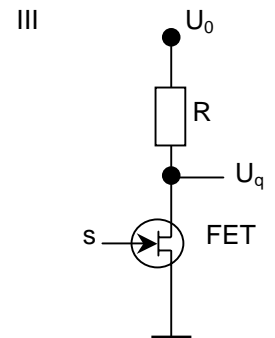
Schematischer Aufbau mit Ein- und Ausgängen



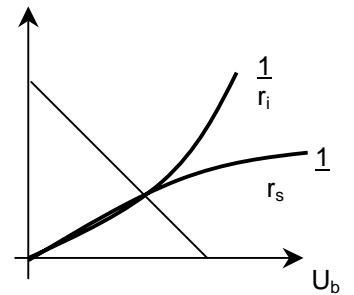
Signalschalter



td ~ 5-35 ns
Ein- und Ausschaltverzögerung

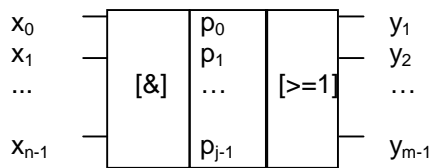


FET - Feldeffekttransistor



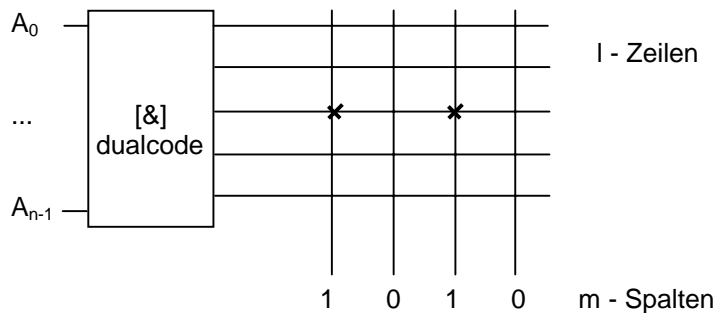
Diodenmatrixanordnungen

- 1) Grundgatter
- 2) beliebige Schaltungen
aufbauen → Funktionstabelle direkt auf Matrix abbilden
- 3) Speicher
Festwertspeicher



n Eingänge m Ausgänge
j - Zwischenvariablen

ROM - Read-Only-Memory

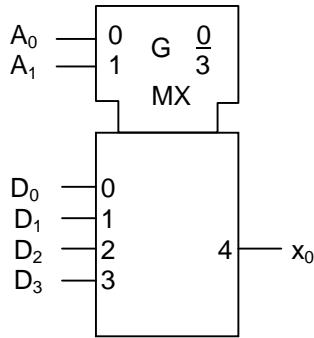


Fest in der Matrix gespeicherte Werte können ausgelesen werden, indem man die Matrix zeilenweise ansteuert. Die Eingänge dienen somit als Adresse.

EPROM - Erasable-Programmable-Read-Only-Memory

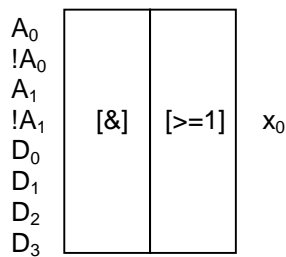
Hier ist die Matrix lösch- und wiederbeschreibbar.
Anstatt der Dioden werden Feldeffekttransistoren genutzt.

4:1 MUX

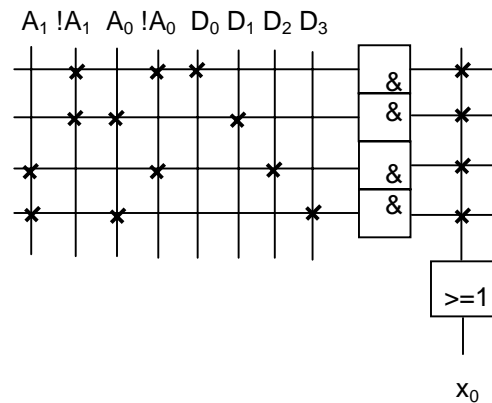


Funktionstabelle

i	A ₁	A ₀	x ₀	P _i
0	0	0	D ₀	$\overline{A_1} \overline{A_0} D_0$
1	0	1	D ₁	$\overline{A_1} A_0 D_1$
2	1	0	D ₂	$A_1 \overline{A_0} D_2$
3	1	1	D ₃	$A_1 A_0 D_3$

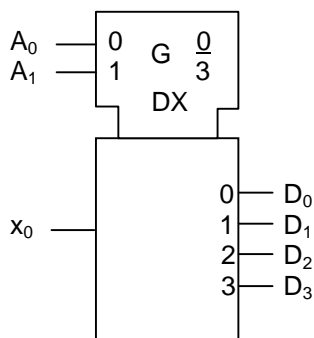


entsprechende Matrix:



n = 8 → Eingänge
l = 4 → Produktterme
m = 1 → Ausgang

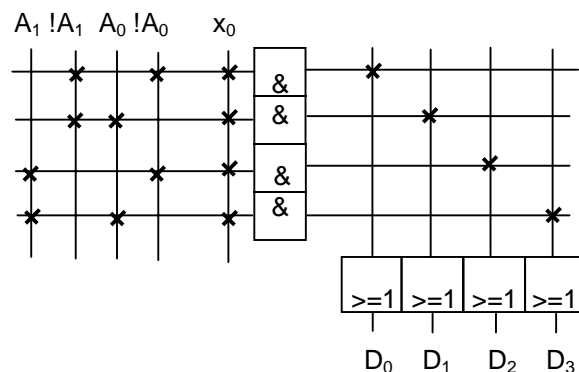
1:4 DMUX



Funktionstabelle

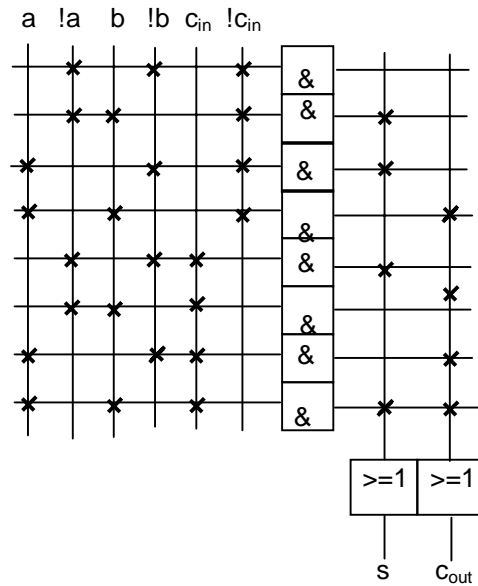
i	A ₁	A ₀	D ₀	D ₁	D ₂	D ₃	P _i
0	0	0	x ₀				$\overline{A_1} \overline{A_0} x_0$
1	0	1		x ₀			$\overline{A_1} A_0 x_0$
2	1	0			x ₀		$A_1 \overline{A_0} x_0$
3	1	1				x ₀	$A_1 A_0 x_0$

entsprechende Matrix:



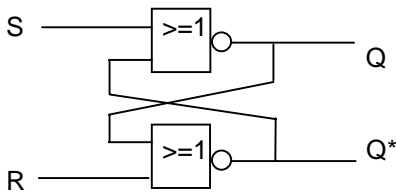
Volladdierer

i	a	b	C _{in}	S	C _{out}
0	0	0	0	0	0
1	0	1	0	1	0
2	1	0	0	1	0
3	1	1	0	0	1
4	0	0	1	1	0
5	0	1	1	0	1
6	1	0	1	0	1
7	1	1	1	1	1

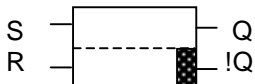


Flip-Flop

Am Anfang werden alle Ein- und Ausgänge auf NULL gesetzt.
 Nach Entfernen des Nullpotentials stellt sich ein bestimmter Zustand ein.



S – Setzeingang
 R – Rücksetzeingang



Zeitfolgetabelle

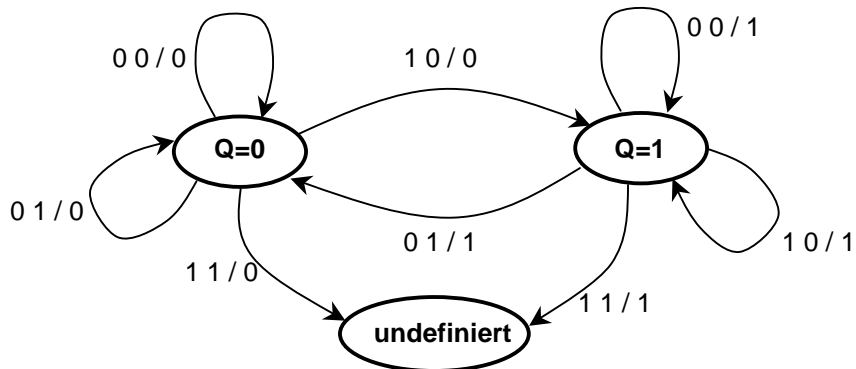
k	R	S	Q	Q*	
	0	0	0	0	Initialisierung
				1	dies ergibt sich
1	1	0	0	1	
2	0	0	0	1	
3	0	1	1	0	
4	0	0	1	0	
5	1	1	0	0	Q = Q* → irregulär

Vollständige Funktionstabelle (Zustandsfolgetabelle)

Q ^k	S ^k	R ^k	Q ^(k+1)	Info
0	0	0	0	bleibt erhalten (gespeichert)
0	1	0	1	setzen
0	0	1	0	zurücksetzen
0	1	1	?	undefiniert
1	0	0	1	bleibt erhalten (gespeichert)
1	1	0	1	setzen
1	0	1	0	zurücksetzen
1	1	1	?	undefiniert

Zustandsfolgegraph

Knoten – Zustände
Zweige – Zustandsänderung

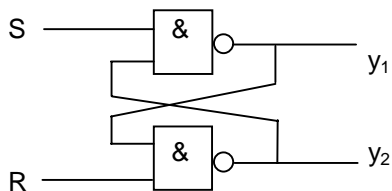


$$Q^{(k+1)} = !QS !R + Q !S!R + QS!R = (!Q + Q)S!R + QS!R$$

$$Q^{(k+1)} = !R(S + Q!S) = S!R + Q!R = !R(S + Q)$$

RS-FF
aus NAND

Flip-Flop mit AND-Gliedern



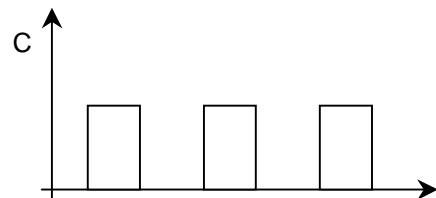
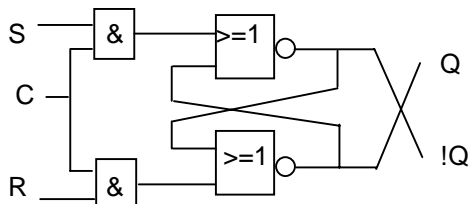
k	S set	R reset	y ₁ !Q	y ₂ Q	
0	0	0	0	0	Initialisierung
	0	0	1	1	→ so nicht erlaubt
0	1	1	1	1	Initialisierung
0	1	1	0	1	es ergibt sich
	1	0	0	1	
	1	1	0	1	
	0	1	1	0	
	1	0	0	1	
	0	1	1	0	
	1	1	1	0	

Q ^k	R	S	Q ^{k+1}
0	0	0	n.d.
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	n.d.
1	0	1	1
1	1	0	0
1	1	1	1

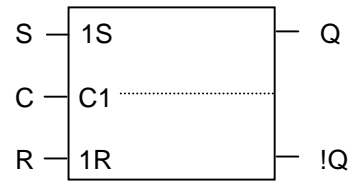
Nach Zeittakt
aufgeschlüsselt
ergibt sich
diese Tabelle.

NOR RS - Flip-Flop

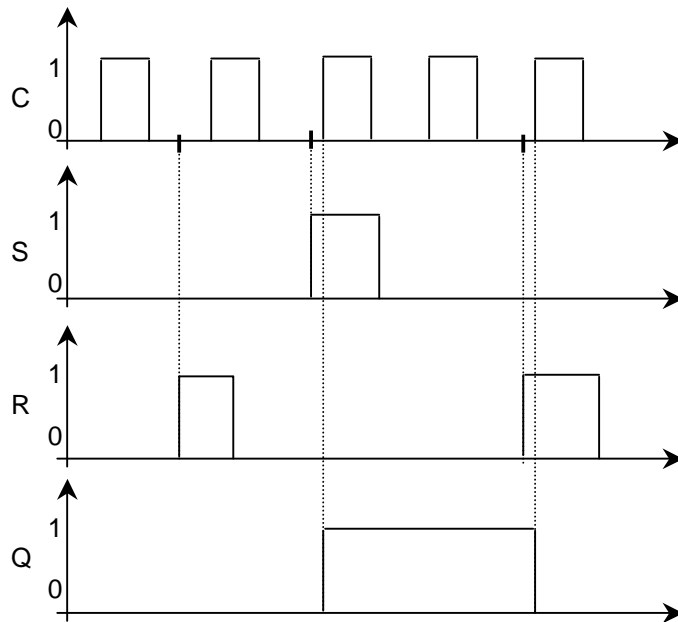
(Zustandselement, zeitgetaktet)



- die Eingänge liegen nur an, wenn Zeittakt gegeben (Clock = '1')
- Taktsteuerung durch Eingangstore (1:1 MUX)
- Erregungssignale sind während Clock-High aktiv, müssen aber während dieser Zeit konstant gehalten werden (Störanfälligkeit wegen langer Toröffnungszeiten)
- Taktzustandsgesteuert → Taktflankensteuerung



Zeitablaufdiagramm (taktgesteuert)



Taktflankensteuerung:

Das Ausgangssignal wird nur dann (je nach Eingangssignal) umgeschaltet, wenn an der Taktung (Clock) eine positive Flanke anliegt.

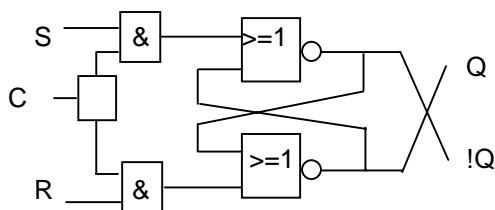
J-K-Flip-Flop

- Flip-Flop wo undefinierte (irreguläre) Zustände herausgefiltert werden
- Funktionstabelle gleich dem RS-FF, aber die Zeilen mit den irregulären Zuständen fehlen

NOR - Flip-Flop taktflankengesteuert

Taktflankensteuerung

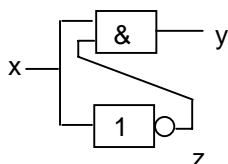
Eingangstore (1:1 MUX) werden durch differenziertes Clock-Signal gesteuert



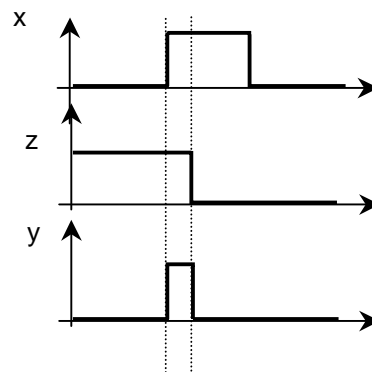
Hierbei steuert das Clock-Signal (Rechteckspannung) einen Differenziator an, der an die jeweiligen Ausgänge nur Flanken (sogenannte Spitzen) liefert.

Differenziator

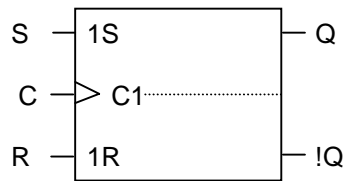
Ein Differenziator kann wie folgt realisiert werden:



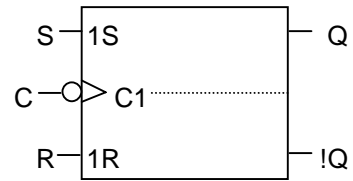
Durch den Inverter braucht das Signal eine Zeit t (~5ns) um wieder am UND-Glied anzukommen. Während dieser Zeit liegt an y eine ‚Nadel‘ an.



- keine Störanfälligkeit wegen langer Toröffnungszeiten
- Eingangssignale können auch sehr kurz sein

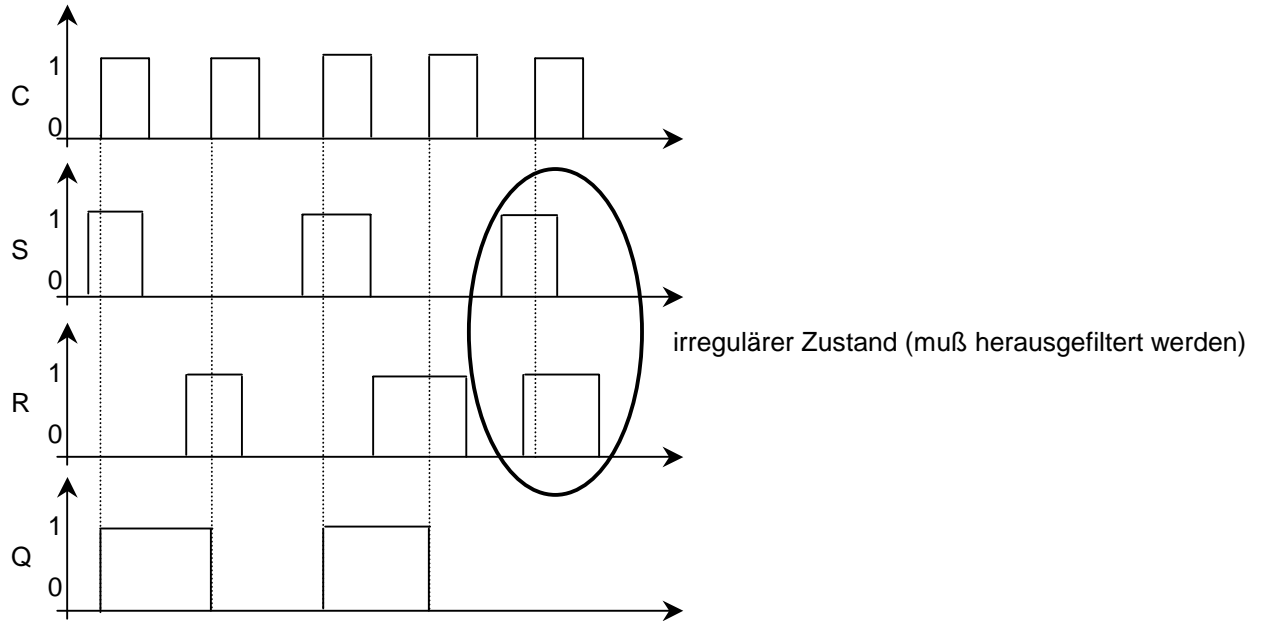


Steuerung auf steigende Taktflanke

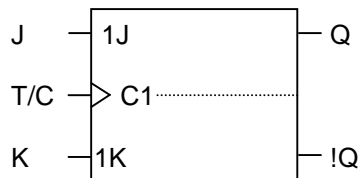


Steuerung auf fallende Taktflanke

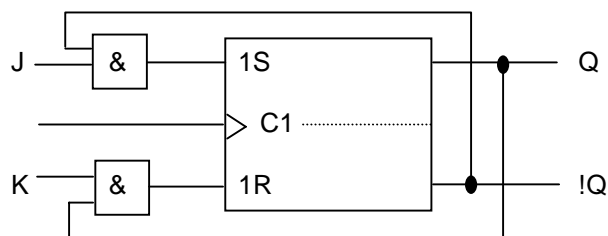
Zeitablaufdiagramm



J-K – Flip-Flop



Bei einem JK - Flip-Flop werden die irregulären Zustände herausgefiltert.



$$\begin{aligned} S &= J \cdot \bar{Q} \\ R &= K \cdot Q \\ \bar{Q} &= \bar{(Q)} \end{aligned}$$

Funktionstabelle JK-Flip-Flop

k	unabhängige Variablen			abhängige Variablen			Ergebnis	
	J	K	Q	!Q	S	R	Q ^(k+1)	
0	0	0	0	1	0	0	0	init
1	1	0	0	1	1	0	1	set
2	0	1	1	0	0	1	0	reset
3	1	1	0	1	1	0	1	set
4	1	1	1	0	0	1	0	reset
5	0	1	0	1	0	0	0	-
6	1	0	0	1	1	0	1	set
7	1	0	1	0	0	0	1	-
8	0	0	1	0	0	0	1	-

Funktionstabelle Flip-Flops

S	R	Q	Q _{RS} ^(k+1)	Q _{JK} ^(k+1)
0	0	0	0	0
1	0	0	1	1
0	1	0	0	0
1	1	0	--	1
0	0	1	1	1
1	0	1	1	1
0	1	1	0	0
1	1	1	--	0

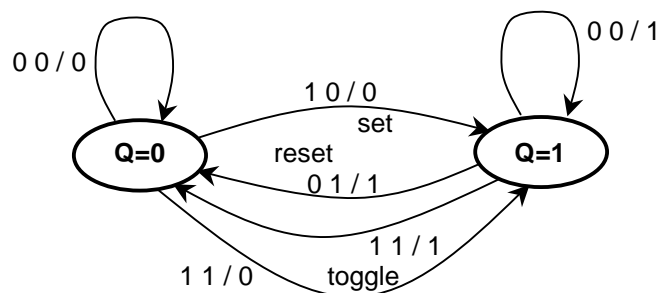
vereinfacht:

S	R	Q _{RS} ^(k+1)	J	K	Q _{JK} ^(k+1)	
0	0	Q ^k	0	0	Q ^k	speichern
1	0	1	1	0	1	set
0	1	0	0	1	0	reset
1	1	--	1	1	!Q ^k	toggle

Liegen an den JK - Flip-Flop-Eingängen HIGH-Potentiale, so wird der gespeicherte Wert invertiert (toggle).

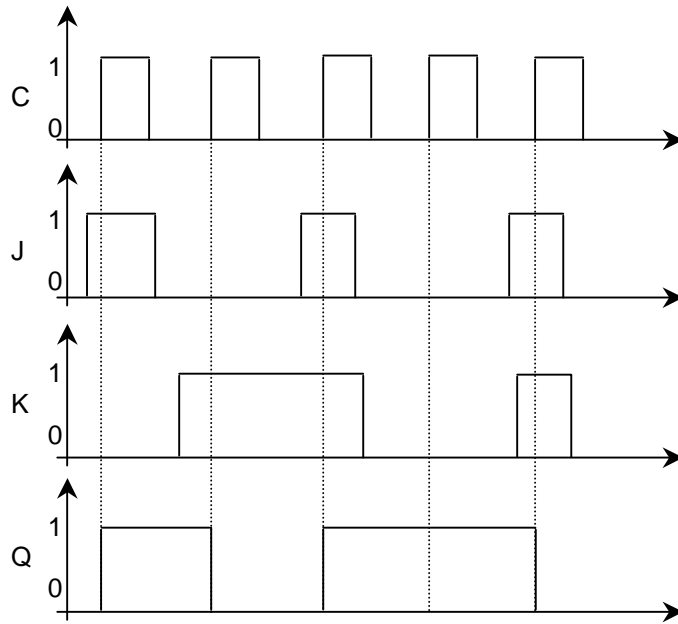
Zustandsfolgegraph

Knoten – Zustände
Zweige – Zustandsänderung



$$Q^{(k+1)} = !J !K Q + J !K (Q + !Q) + J K !Q = J !Q + !K Q$$

Zeitablaufdiagramm JK-Flip-Flop



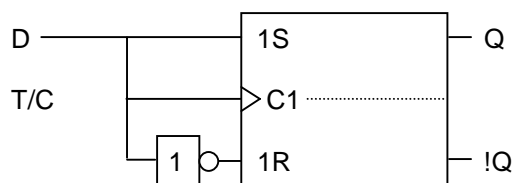
Um mit einer Formel möglichst einfach mehrere Flip-Flops beschreiben zu nutzen man folgende Formel:

$$Q^{(k+1)} = g_1 Q^k + g_2 !Q^k \quad \text{wobei } g_1 \text{ und } g_2 \text{ je nach Flip-Flop-Typ variieren.}$$

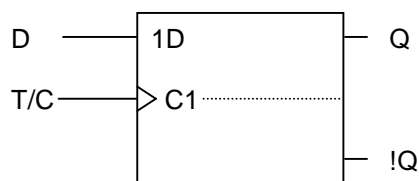
	g_1	g_2
JK	$!K$	J
RS	$!R$	S
T	$!T$	T
D	D	D

D – Flip-Flop

RS – Flip-Flop mit D-Ansteuerung



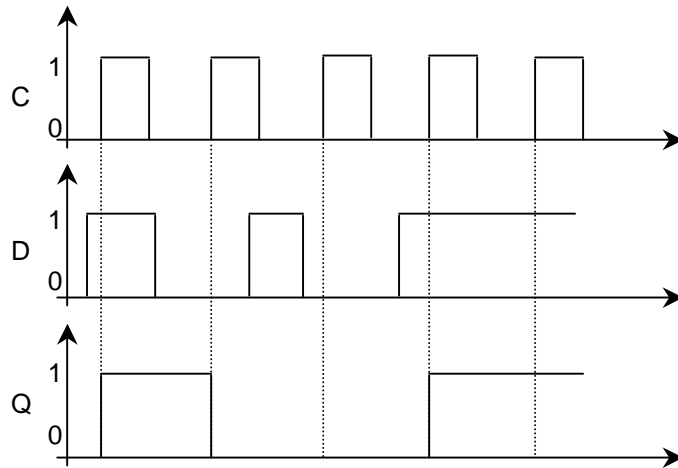
$$S = D, R = !D$$



S=D	R=!D	Q	$Q^{(k+1)}$	
0	0	0	0	
1	0	0	1	$D * !!D * !Q = D * !Q$
0	1	0	0	
1	1	0	--	
0	0	1	1	
1	0	1	1	$D * !!D * Q = D * Q$
0	1	1	0	
1	1	1	--	

$$Q^{k+1} = D * Q + D + !Q = D$$

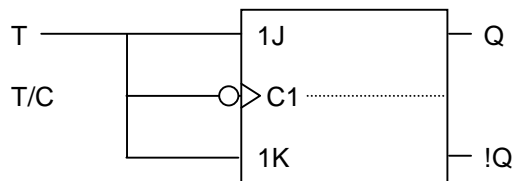
Zeitablaufdiagramm D-Flip-Flop



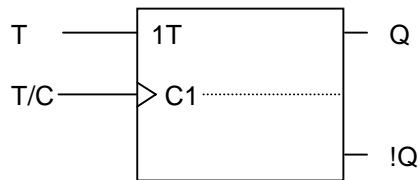
D – Datenspeicher (data-latch)
 - Wert des Datensignals wird gespeichert
 - Verlängerung des D-Impulses nach Taktsignal (Delay-Flip-Flop)

T – Flip-Flop

Toggle – Flip-Flop



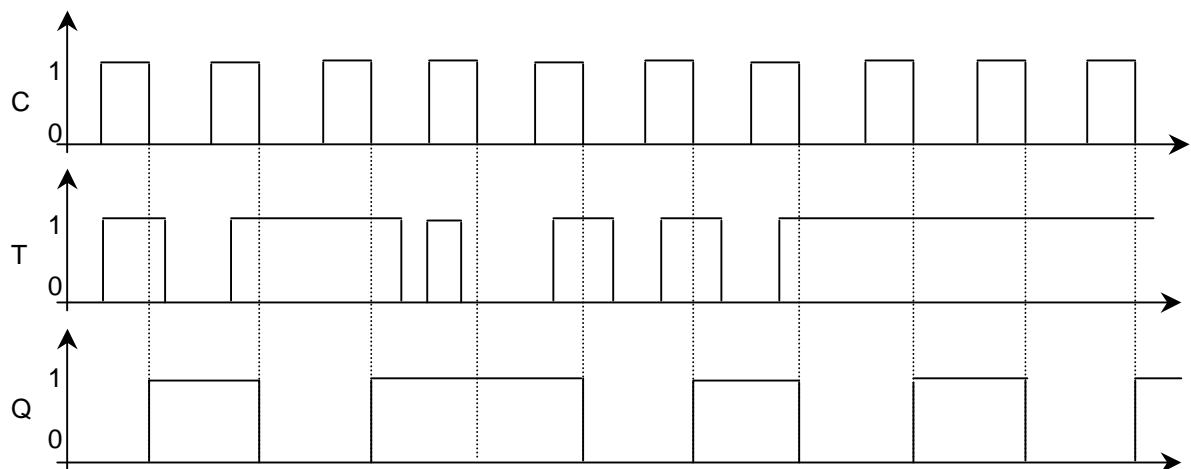
$J = T, K = T$



J=T	K=T	Q	Q ^(k+1)	
0	0	0	0	
1	0	0	1	$T * !T * !Q = 0$
0	1	0	0	
1	1	0	1	$T * T * !Q = T * !Q$
0	0	1	1	$!T * !T * Q = !T * Q$
1	0	1	1	$T * !T * Q = 0$
0	1	1	0	
1	1	1	0	

$Q^{k+1} = !T * Q + T + !Q$

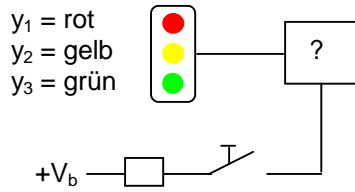
Zeitablaufdiagramm T-Flip-Flop



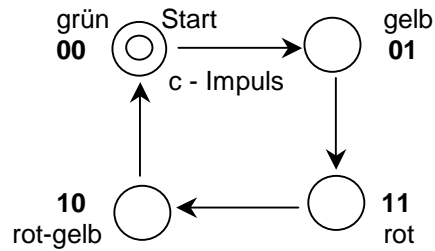
Bei T = konst. → Frequenzteiler → T = J = K = 1

Flip-Flop – Schaltungsanalyse und -synthese

Ampelschaltung mit Flip-Flops



mögliche Ampelzustände:



Anzahl der Zustände = $z = 4$ $z = 2^2 = 4 = 2^p$
 Anzahl der Flip-Flops = p

Codierung der Zustände:

i	z	Q_2	Q_1
1	grün	0	0
2	gelb	0	1
3	rot	1	1
4	rot-gelb	1	0
5	grün	0	0

Funktionstabelle (JK-Flip-Flop):

i	Q_2	Q_1	$Q_2^{(k+1)}$	$Q_1^{(k+1)}$	J_2	K_2	J_1	K_1	$y_{\text{grün}}$	y_{gelb}	y_{rot}
1	0	0	0	1	0	0	1	d	1	0	0
2	0	1	1	1	1	d	0	0	0	1	0
3	1	1	1	0	0	0	d	1	0	0	1
4	1	0	0	0	d	1	0	0	0	1	1

Übersicht Flip-Flops

$$Q^{(k+1)} = g_1 Q + g_2 !Q$$

Q^k	$Q^{(k+1)}$	S	R	J	K	D	T
0	0	0	d	0	d	0	0
0	1	1	0	1	d	1	1
1	0	0	1	d	1	0	1
1	1	d	0	d	0	1	0
		$g_1 = !R$		$g_1 = !K$		$g_1 = D$	$g_1 = !T$
		$g_2 = S$		$g_2 = J$		$g_2 = D$	$g_2 = T$
		SR = 0					

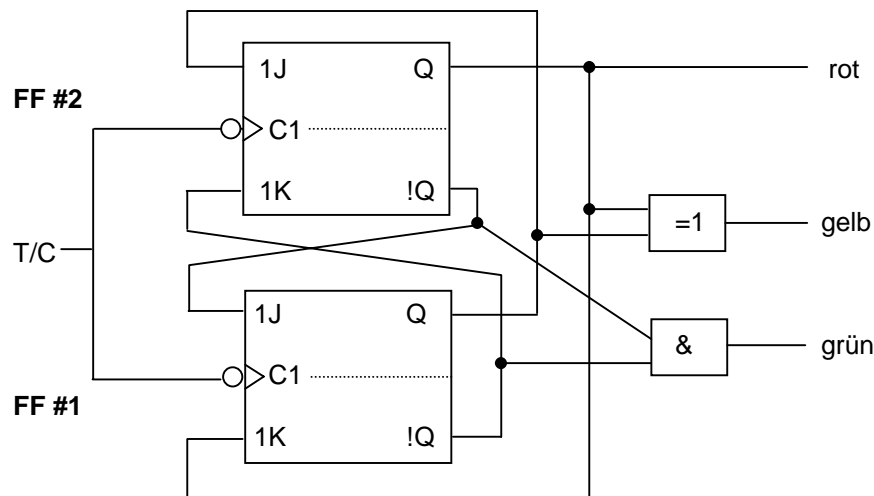
Ampel mit Flip-Flops

Funktionen

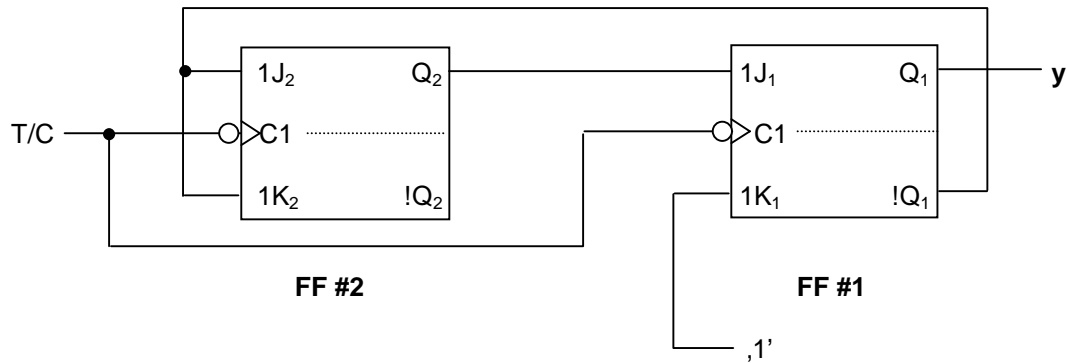
FF #1: $Q_1^{(k+1)} = !Q_2 !Q_1 + !Q_2 Q_1$
 $= !Q_2 Q_1 + !Q_2 !Q_1$
 $= g_1 Q_1 + g_2 !Q_1$
 $\rightarrow g_1 = J_1 = !Q_2$
 $\rightarrow g_2 = !K_1 = !Q_2 \rightarrow K_1 = Q_2$

FF #2: $Q_2^{(k+1)} = !Q_2 Q_1 + Q_2 Q_1$
 $= Q_1 Q_2 + Q_1 !Q_2$
 $= g_1 Q_2 + g_2 !Q_2$
 $\rightarrow g_1 = J_2 = Q_1$
 $\rightarrow g_2 = !K_2 = Q_1 \rightarrow K_2 = !Q_1$

Schaltplan

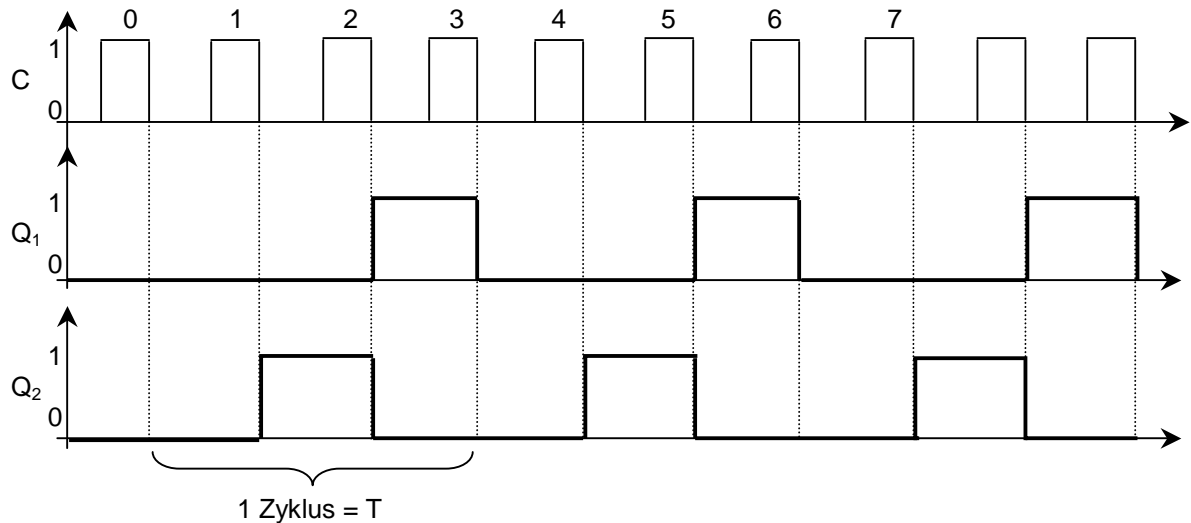


Schaltungsanalyse



Takt	Zustand k		FF #1		FF #2		Zustand k+1			
k	Q_1^k	Q_2^k	J_1	K_1	J_2	K_2	$Q_1^{(k+1)}$	$Q_2^{(k+1)}$	y	
0	0	0	0	1	1	1	0	1	0	Start: Q_1 und Q_2 auf '0' setzen
1	0	1	1	1	1	1	1	0	0	
2	1	0	0	1	0	0	0	0	1	
3 = 0	0	0	0	1	1	1	0	1	0	gleich erstem Zustand

Zeitablaufdiagramm



$T_Y = 3 * T_C$ $y : f_y = 1/3 f_c$ y – Frequenzteiler 3 : 1

Zustandsfolgegraph

