Kapitel 3 - PLA und Flip-Flops

Programmable Logic Array (PLA)

Die Idee eines PLAs ist, dass bei der Chipherstellung ein homogenes Feld von Transistoren erzeugt wird. Die eigentliche Funktionalität wird dann durch Konfiguration und Programmierung durch den Benutzer festgelegt.

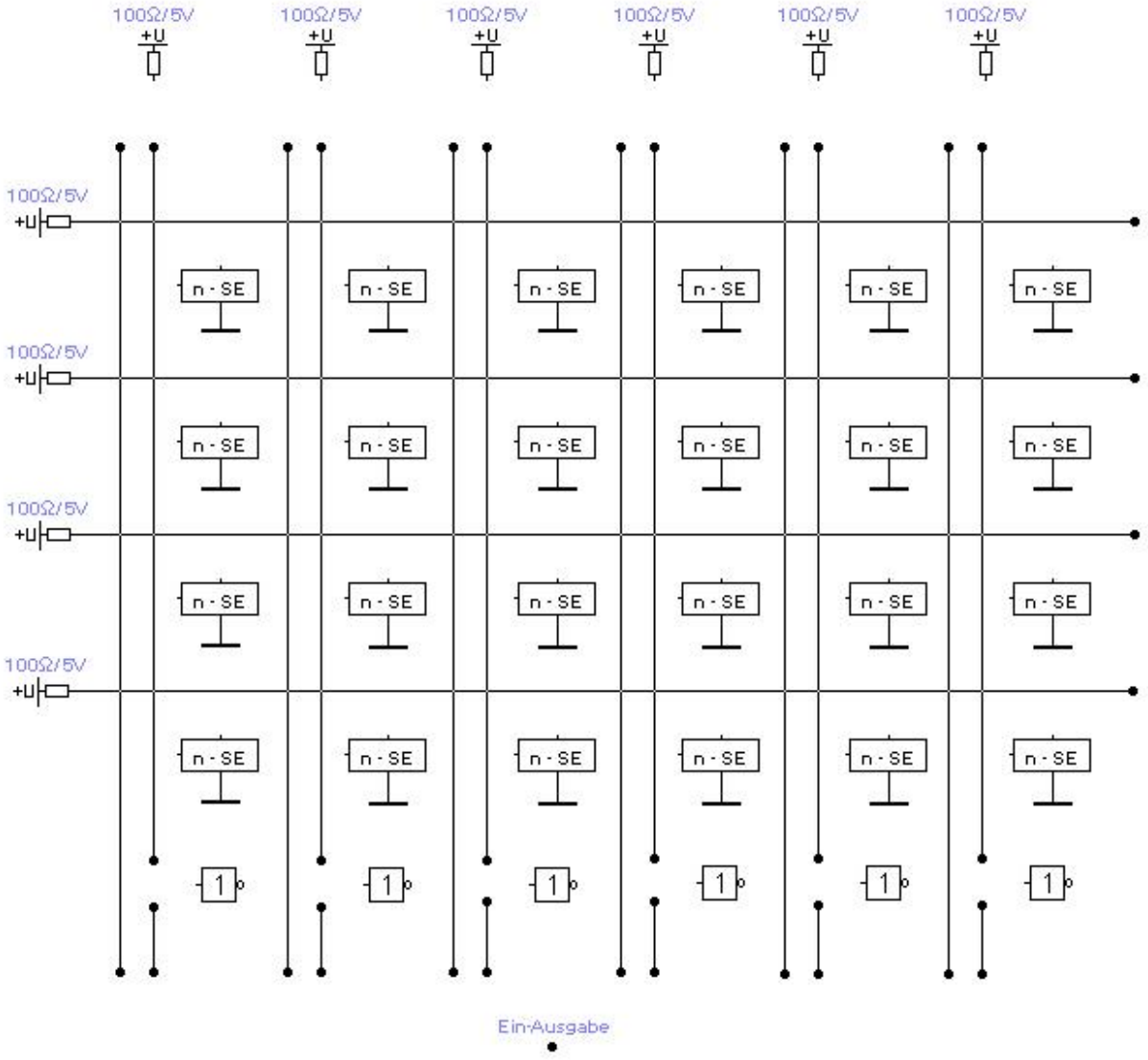
Wir benutzen im nachfolgenden Versuch ein PLA zur Implementierung zweier Boolescher Funktionen f und g. Dazu einige Vorüberlegungen.

Man kann für jede Boolesche Funktion mindestens eine zweistufige Schaltung angeben, die diese realisiert. Man erzeugt dazu die DNF (das geht immer und ist eindeutig) und wähle so viele UND-Gatter aus, wie die DNF Terme hat. Diese UND-Gatter haben so viele Eingänge wie die Terme Literale haben. Besteht ein Literal aus einer negierten Variablen, muss noch ein Negationsgatter vorgeschaltet werden. Zum Schluss leitet man alle Ausgänge der UND-Gatter in ein großes ODER-Gatter, dessen Ausgang den Funktionswert liefert.

Obiges Verfahren garantiert immer eine Lösung. Möglicherweise ist diese aber sehr aufwändig, weshalb wir die Darstellung einer Booleschen Funktion optimieren, um eine möglichst kurze disjunktive Form zu erhalten. Minimiert werden dadurch sowohl die Anzahl der Eingänge pro UND-Gatter als auch die Anzahl der Gatter selbst. Sie haben in der Vorlesung Rechnerstrukturen mehrere Verfahren zur Darstellung und Minimierung kennengelernt, von denen Sie einige jetzt einsetzen können.

Betrachten wir ein PLA zunächst im Rohzustand:

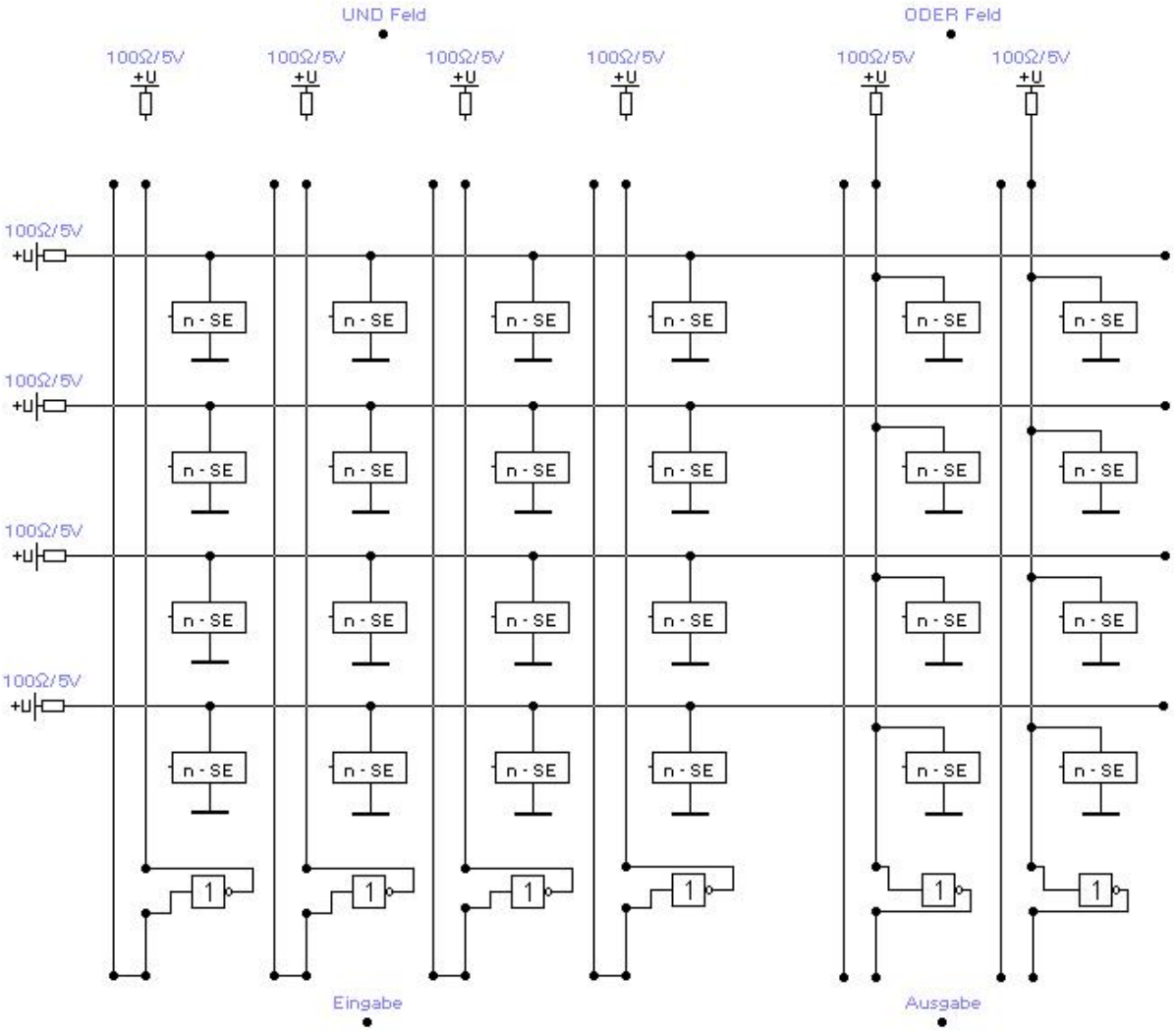
Unkonfiguriertes PLA:



Die Ein- und Ausgabeanschlüsse für den Benutzer sind auf dem Bild unten gegeben. Es sind insgesamt 12 Anschlüsse von denen 6 in beide Richtungen invertiert werden können.

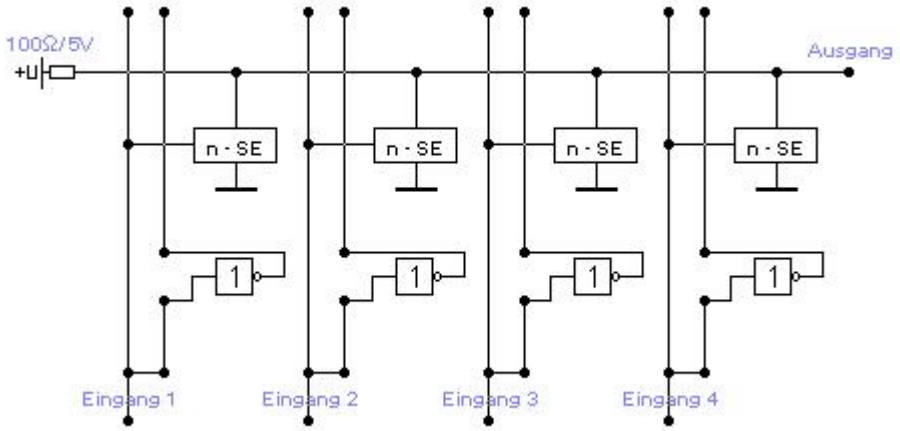
Im nächsten Schritt unterteilen wir das PLA in ein UND- und ein ODER-Feld:

Konfiguriertes PLA:



Man erkennt links unten 4 Eingabeleitungen und rechts unten 2 konfigurierte Ausgabeleitungen.

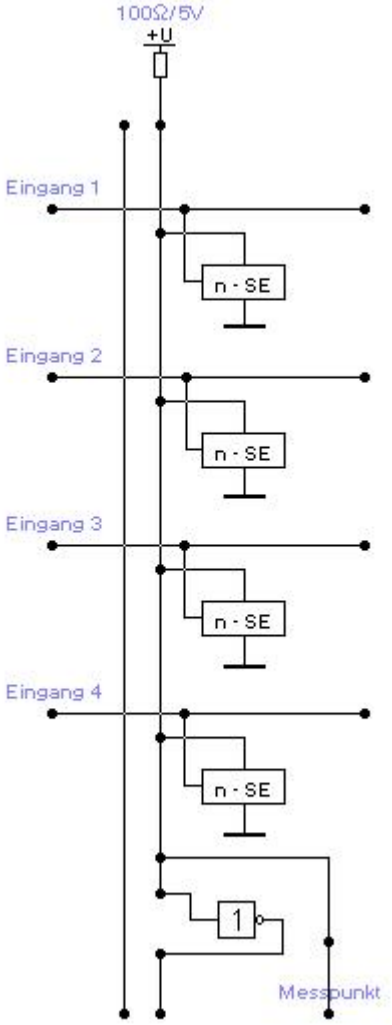
Im obigen Beispiel wurde ein 4 x 4 UND-Feld und ein 4 x 2 ODER-Feld konfiguriert. Dazu wurden die Transistoren zu Gattern zusammengeschaltet. Betrachten wir als erstes eine Zeile des UND-Feldes:



Die 100Ω/5V Quelle dient als Pullup-Widerstand für eine Wired-Schaltung. Das ist dasselbe wie eine starke 1 mit einem Widerstand (s. Kapitel 1). Die Drainanschlüsse aller Transistoren sind auf einem „Draht“ zusammengeschaltet, der rechts den Ausgang des Gatters bildet. Die Sourceanschlüsse sind auf 0 Volt gelegt. Die senkrechten Leitungspaare sind von unten her die Eingänge der Variablen einer Gleichung. Jede Variable wird normal und negiert zur Verfügung gestellt. In dieser Schaltung sind die Gate-Anschlüsse der n-SE mit den Eingängen der Variablen verbunden. Der Ausgang der Schaltung ist rechts oben. Mit Hilfe einer Wertetabelle können Sie nun die logische Funktion der Schaltung ermitteln.

Welche logische Funktion hat dieses Gatter?

Jetzt betrachten wir eine Spalte des ODER-Feldes.



Auch hier bildet die 100Ω/5V Quelle wieder den Pullup-Widerstand. Alle Drain-Anschlüsse sind mit dem Draht verbunden, alle Source-Anschlüsse auf Masse gelegt. Die Gateanschlüsse sind mit den waagerechten Ausgängen des UND-Feldes verbunden und stellen die Eingänge dieses Gatters dar. Um die logische Funktion dieser Schaltung zu ermitteln, verbinden Sie die Gateanschlüsse der n-SE mit den Eingängen der Variablen. Als Ausgang der Schaltung wird der Messpunkt betrachtet. Mit Hilfe einer Wertetabelle können Sie nun die logische Funktion ermitteln.

Welche logische Funktion hat dieses Gatter am Messpunkt?

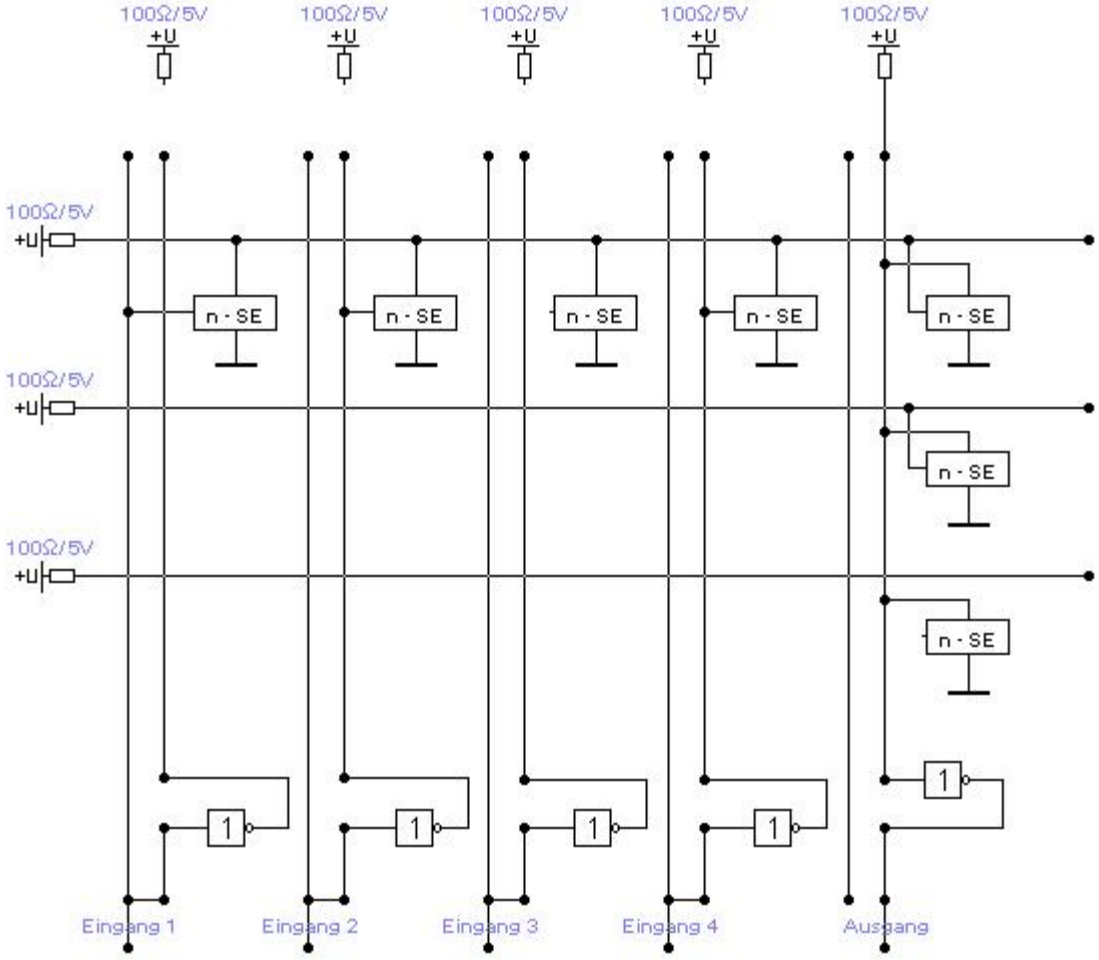
In beiden Fällen handelt es sich weder um ein UND- noch um ein ODER-Gatter, wie es eigentlich gewünscht wird.

Nehmen Sie zu den gefundenen Gatterfunktionen noch den Negationsoperator hinzu und zeigen Sie durch algebraische Umformungen, dass Sie trotzdem im UND-Feld UND-Verknüpfungen und im ODER-Feld ODER-Verknüpfungen realisieren können.

Nach der Konfiguration sollen Sie das PLA programmieren. Programmieren bedeutet:

- Im UND-Feld: die Gate-Anschlüsse mit einer der beiden senkrechten Leitungen links davon zu verbinden oder offen zu lassen.
- Im ODER-Feld: die Gate-Anschlüsse mit der waagerecht darüber liegenden Leitung zu verbinden oder offen zu lassen.

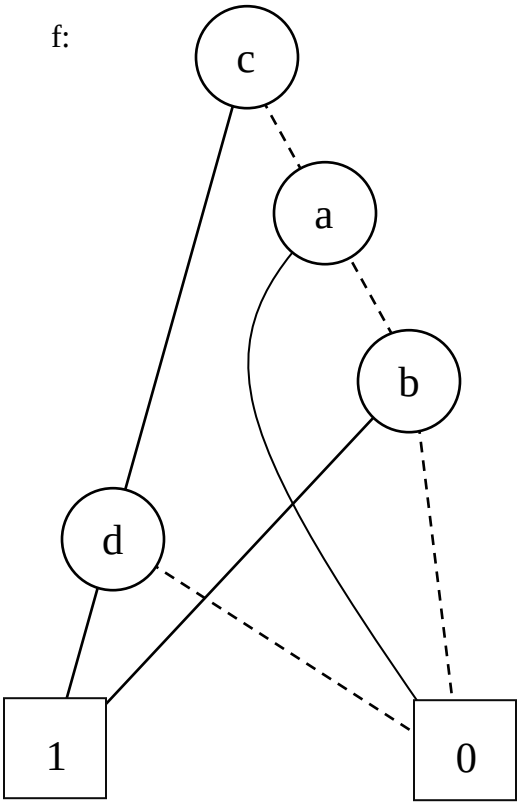
Beispielsweise (Ausschnitt mit nur einer Zeile und einer Spalte):



Versuch 300 PLA-Programmierung

Gegeben seien zwei Boolesche Funktionen f und g. Stellen sie diese in einer möglichst kurzen disjunktiven Form dar.

f:



Das OBDD von f ist bereits optimal.
Eine durchgezogene Linie bedeutet 1,
eine gestrichelte 0.

Minimale disjunktive Form:

f(a,b,c,d) =

g:

a	b	c	d	g
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
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	0
1	1	1	0	0
1	1	1	1	0

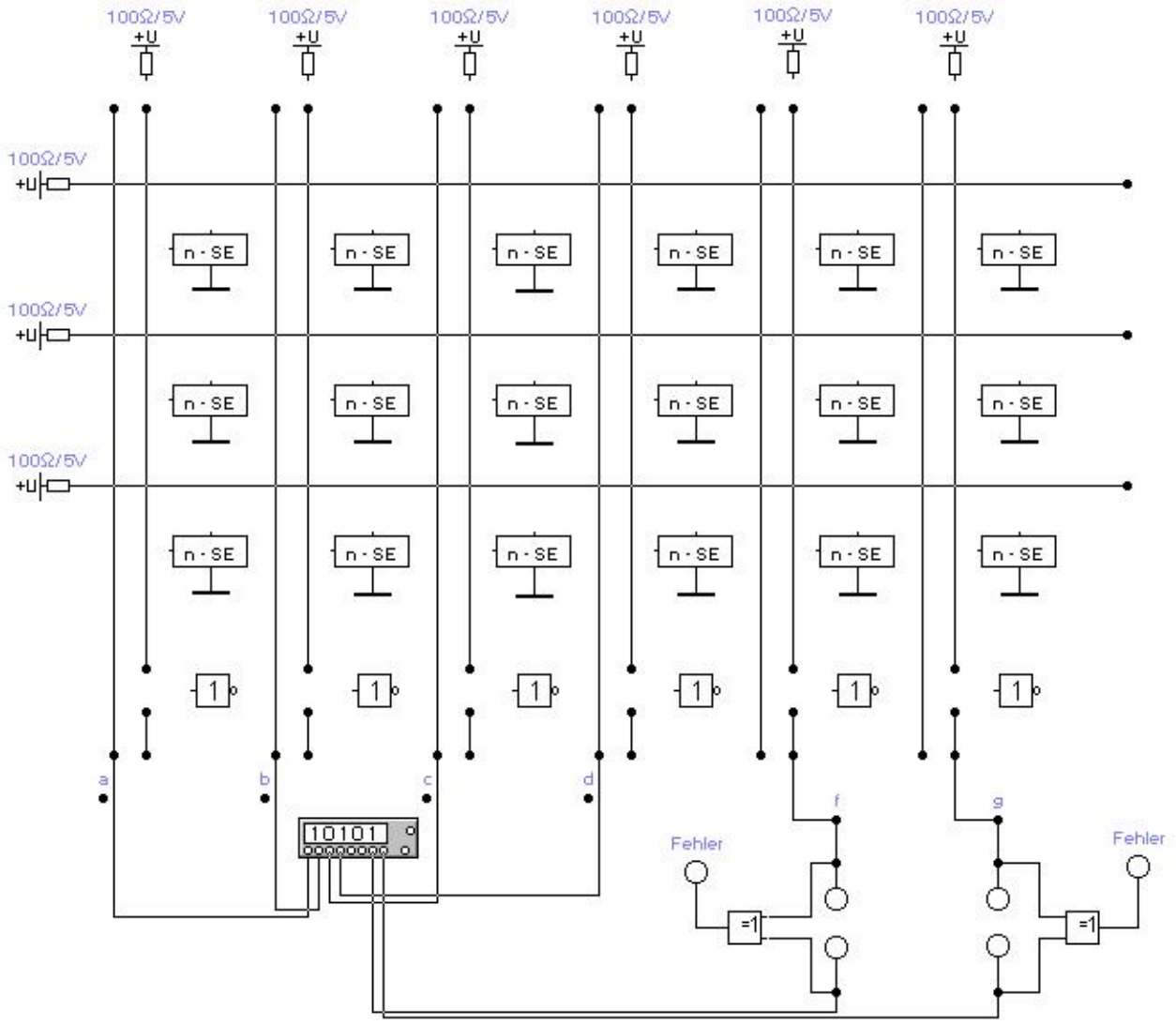
Optimierung von g mit KV-Diagramm:

g:	00	01	11	10
00				
01				
11				
10				

Minimale disjunktive Form:

g(a,b,c,d) =

Aufgabe v300:
Programmieren Sie ein PLA, das beide Funktionen f und g realisiert. In der Datei v300 steht Ihnen hierfür ein 3 x 6 PLA zur Verfügung.



Legen Sie zunächst das UND-Feld und das ODER-Feld fest, indem Sie das PLA entsprechend konfigurieren. Programmieren (verdrahten) Sie nun beide Felder auf Grundlage der von Ihnen berechneten und in ihrer Darstellung optimierten Funktionen f und g.

Testen Sie Ihre Schaltung. Öffnen Sie dazu den Bitmustergenerator (Doppelklick). Er ist bereits mit 0 und 1 vorbelegt (Datei v300 pla.dp). Die zugehörigen 4 Spalten und Ausgänge (links) sind mit den Eingängen (a,b,c,d) Ihres PLAs verbunden. Die rechten beiden Spalten und Ausgänge enthalten die Funktionswerte für f und g. Diese Ausgänge führen zu Fehler-Lämpchen, die den Ausgängen Ihres PLAs gegenüberliegen.
Starten Sie den Bitmustergenerator mit CYCLE oder STEP. Jeder PLA-Ausgang ist über ein XOR mit dem entsprechenden Testausgang des Bitmustergenerators verbunden. Falls Ihre Schaltung korrekt arbeitet, leuchten beide Fehler-Lämpchen nie auf.
Andernfalls finden Sie den oder die Fehler und korrigieren Sie Ihre Schaltung.

Sequentielle Schaltungen

Mit logischen Gattern aufgebaute Netzwerke lassen sich in zwei Klassen einteilen:

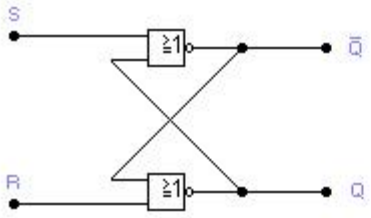
• **Schaltnetze (Kombinatorische Schaltungen)** sind Schaltungen ohne Speichervermögen, d.h. die zum Zeitpunkt t an den Ausgängen der Schaltung herrschenden Binärzustände hängen nur von den zu diesem Zeitpunkt an den Eingängen der Schaltung wirksamen Binärzustände ab. Grundlage der Theorie binärer Schaltungen ist die Boolesche Algebra.

• **Schaltwerke (Sequentielle Schaltungen)** sind Schaltungen mit Speichervermögen, d.h. die zu einem Zeitpunkt t an den Ausgängen herrschenden Binärzustände hängen von den Binärzuständen an den Eingängen und von bestimmten internen Binärzuständen ab, die zu diesem Zeitpunkt in der Schaltung stabil herrschen. Die internen Binärzustände entstanden durch Eingabe vor dem Zeitpunkt t und sind bis zum Zeitpunkt t gespeichert worden. Grundlage der Theorie binärer sequentieller Schaltungen sind die Booleschen Funktionen und die Automatentheorie.

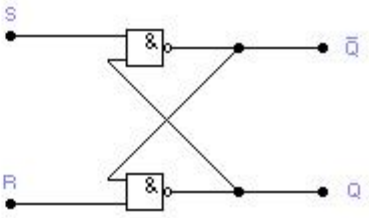
NOR- und NAND-Basis-Flipflops

Einer der Grundbausteine sequentieller Schaltungen ist das Flipflop. Diese einfache elektronische Schaltung ist im Stande, zwei Zustände anzunehmen und zu speichern, es hat also die Speicherkapazität eines Bits. Es gibt zwei elementare Realisierungsmöglichkeiten eines Flipflops, nämlich in Form eines NAND- oder NOR–Basis-Flipflops. Der Name der jeweiligen Realisierung hängt von den benutzten Gattern ab und von der Tatsache, dass sich alle anderen Flipflops aus diesen beiden aufbauen lassen.

NOR-Basis-Flipflop



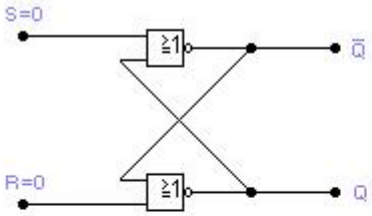
NAND-Basis-Flipflop

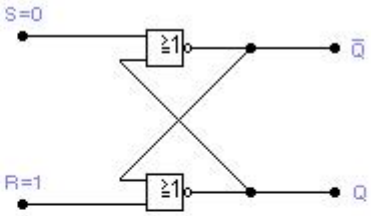


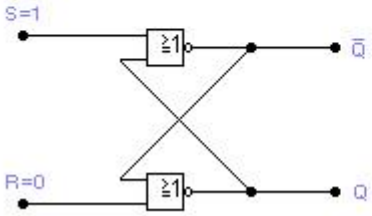
Theorie Analyse eines NOR-Basis-Flipflops

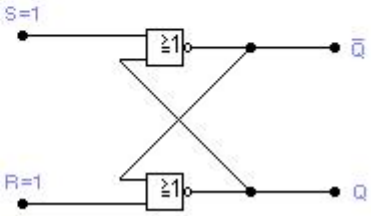
Sie sollen in den folgenden Schaltbildern die Funktionsweise eines NOR-Basis-Flipflops analysieren. Überlegen Sie, welche der vier Funktionalitäten (setzen, löschen, speichern, irregulärer Zustand) das NOR-Basis-Flipflop in den jeweiligen Fällen aufweist. Ein Flipflop befindet sich im irregulären Zustand, wenn Q und $\overline{Q}$ denselben logischen Wert besitzen.

Zur Analyse ermitteln Sie das Verhalten der Schaltung in allen vier nachfolgenden Fällen. Berücksichtigen Sie bei Ihren Überlegungen für jeden einzelnen Fall, dass beim Einschalten zunächst Q = 0 oder Q = 1 sein kann. Anschließend werden **S**et und **R**eset wie angegeben geschaltet.









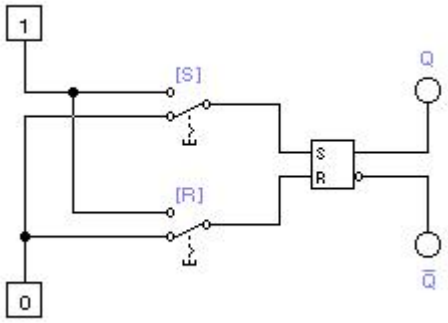
Ergänzen Sie die Einträge in der Ansteuertabelle des NOR-Basis-Flipflops mit den Werten, die Sie gemessen haben.

NOR-Basis-Flipflop:

S	R	Q	¬Q	Funktionalität
0	0			
0	1			
1	0			
1	1			

Versuch 310 EWB-Basis-Flipflop

Das in EWB eingebaute RS-Flipflop entspricht einem NOR-Basis-Flipflop, wobei der Ausgang Q optisch „oben“ liegt.. Testen Sie das Flipflop in der Datei v310.



Das EWB-RS-Flipflop fällt nach jedem Einschalten des Simulators (bei R = S = 0) immer in denselben Grundzustand. Welchen? (Q = ?, $\overline{Q}$ = ?)

Überprüfen Sie die Funktion des Flip-Flops in 3 Schritten:
1. Einschalten, S=1 schalten, Ergebnis notieren, Ausschalten
2. Einschalten, R=1 schalten, Ergebnis notieren, Ausschalten
3. Einschalten, S=1 und R=1 schalten, Ergebnis notieren, Ausschalten

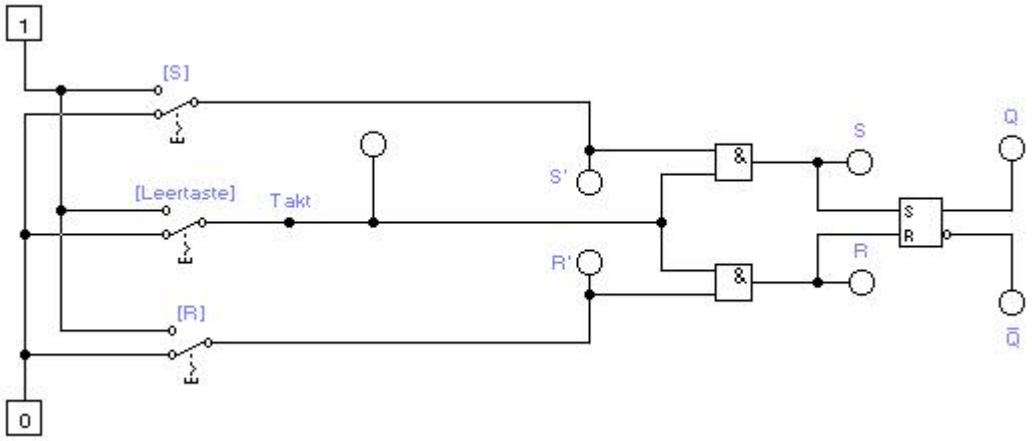
Was ergibt sich, wenn Sie S=1 und gleichzeitig R=1 schalten?

Q = $\overline{Q}$ =

Beurteilen Sie das gemessene Ergebnis.

Versuch 320 Taktzustandsgesteuertes RS-Flipflop

Um das Basis-FlipFlop anzusteuern, werden zwei Schaltimpulse (R und S) benötigt.



In der Schaltung in Datei v320 bilden die beiden UND-Gatter zusammen mit dem RS-Flipflop ein „taktzustandsgesteuertes RS-Flipflop“. Nur wenn der Taktzustand „Binär 1“ ist, können die Taktsignale durch die UND-Gatter weitergeleitet werden.

Analysieren Sie für jede Zeile dieser Tabelle, ob das Flipflop setzt, löscht, speichert oder sich in einem irregulären Zustand befindet. Beim Einschalten der Simulation fällt das Flipflop in den Zustand Q=0 und $\bar{Q}$=1. Vervollständigen Sie die Tabelle.

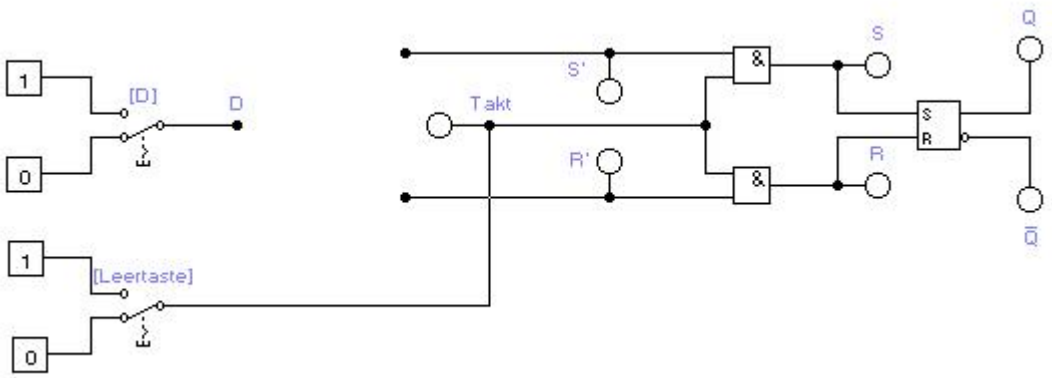
Hinweis: Um die genaue Funktion in jeder Zeile zu erforschen, müssen Sie für jede Zeile die beiden Startzustände Q=0 und Q=1 untersuchen, d.h. Sie müssen die Simulation für jede zu untersuchende Zeile neu starten und ggf. mehrfach takten.

T S' R'	**S**	**R**	**Q**	**¬Q**	Funktionalität
0 0 0	0	0	Q	¬Q	speichern
0 0 1	0	0	Q	¬Q	speichern
0 1 0					
0 1 1					
1 0 0					
1 0 1					
1 1 0					
1 1 1					

Versuch 330 Taktzustandsgesteuertes D-Flipflop

Ein D-Flipflop besitzt nur einen Dateneingang D und hat nur zwei Betriebszustände:
Bei T = 0 behält es seinen bisherigen Zustand, es speichert.
Bei T = 1 übernimmt es den an D anliegenden Wert. (D=0 => Q = 0 und D=1 => Q =1)

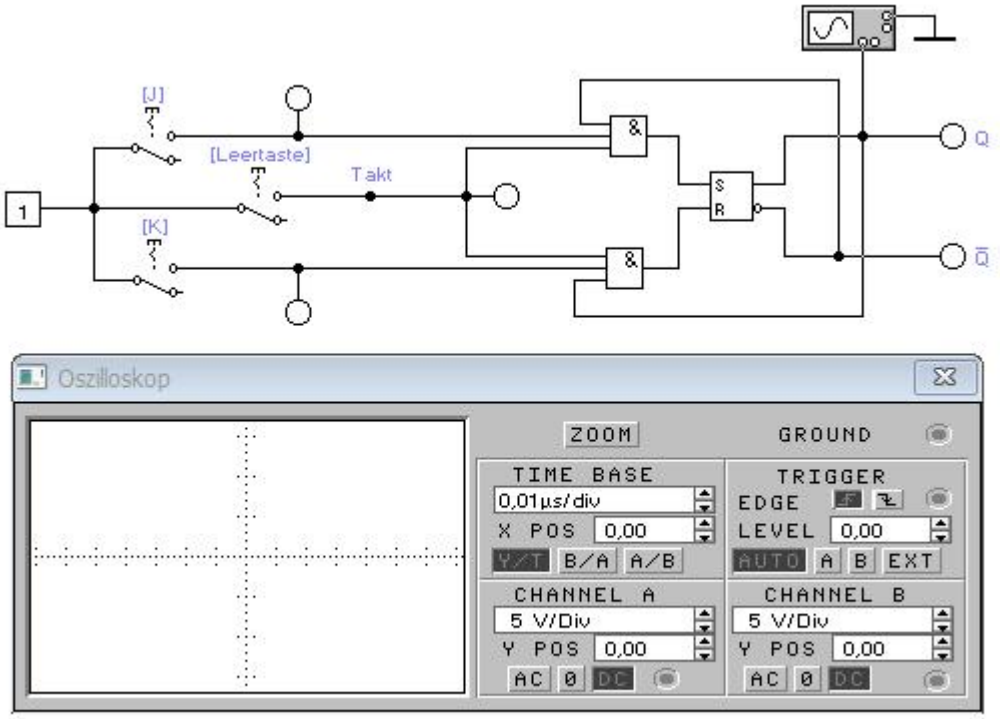
Ergänzen Sie die Schaltung in der Datei v330 so, dass sie obiges Verhalten zeigt.



Das RS-Flipflop ermöglicht also das Setzen, Löschen und Speichern eines Bits. Leider müssen wir in der weiter oben vorgestellten Realisierung auf eine der vier möglichen Belegungen des Flipflops verzichten, da diese zum irregulären Zustand führen könnte. Es wäre also von Vorteil, anstatt einer ungewollten Funktionalität, die sich bei dieser Belegung ergibt, eine neue und sinnvolle zu gewinnen. Könnten wir das taktzustandsgesteuerte RS-Flipflop so umbauen, das wir den irregulären Zustand vermeiden und zugleich alle Belegungen des Flipflops sinnvoll nutzen? Sehen wir uns die folgende Schaltung an.

Versuch 340 Taktzustandsgesteuertes JK-Flipflop

Durch die beiden Rückkopplungen erhalten wir ein JK-Flipflop. Laden Sie die Datei v340. Setzen Sie zuerst J = K = 1 und takten Sie dann von 0 auf 1. Zoomen Sie das Oszilloskop und schalten Sie die Simulation ein. Was beobachten Sie?



Das Flipflop zeigt vier mögliche Verhaltensweisen:

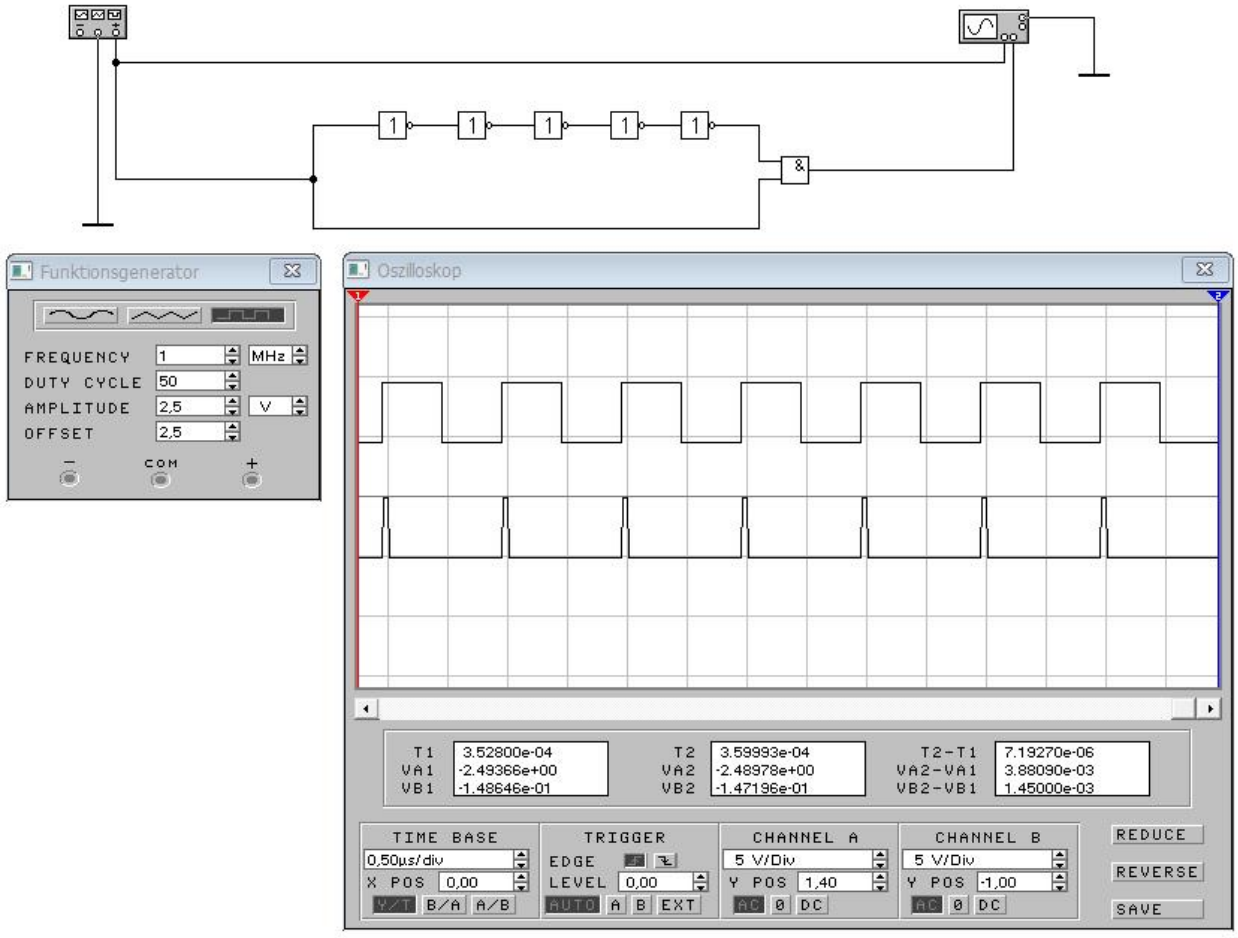
- es schwingt (J=K=T=1)
- es speichert (T=0)
- es wird gesetzt (J=T=1, K=0)
- es wird gelöscht (K=T=1, J=0)

Im obigen Versuch haben wir unsere Schaltung so erweitert, dass wir den irregulären Zustand beseitigt haben. Leider gibt es Fälle, in welchen unsere Schaltung anfängt zu schwingen. Unser Ziel, alle möglichen Belegungen für das Flipflop sinnvoll nutzen zu können, ist noch nicht erreicht.

Was verursacht diese Schwingung?

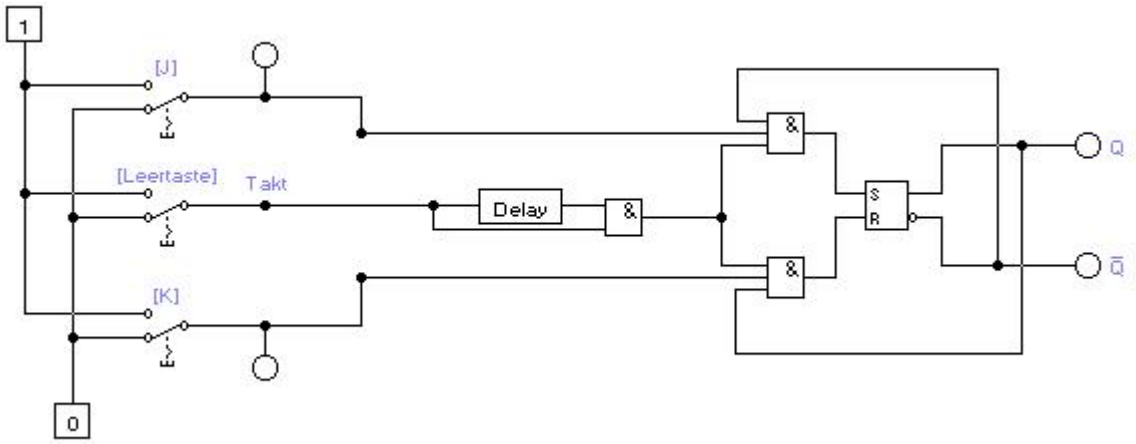
Versuch 350 Taktflankensteuerung

Die Schaltung in Datei v350 bewirkt, dass am Ausgang des UND-Gatters bei einer aufsteigenden (positiven) Flanke nur ein kurzer Impuls (Peak) entsteht. Dadurch wird ein nachgeschaltetes Flipflop nur für einen kurzen Augenblick angesteuert und Schwingungen werden vermieden.
Schalten Sie die Simulation ein und analysieren Sie die Schaltung. Erklären Sie das Zustandekommen des Impulses.



Was passiert, wenn man die Anzahl der Negationen variiert, z.B. eine mehr oder zehn mehr hinzufügt?

Versuch 360 Taktflankengesteuertes JK-Flipflop



Das Makro „Delay“ enthält wie im vorherigen Versuch fünf hintereinander geschaltete Inverter.

Untersuchen Sie das Verhalten des Flipflops und tragen Sie das Ergebnis in die Tabelle ein:

- Reagiert die Schaltung bei Eingabe einer positiven (↑) oder negativen (↓) Taktflanke?
- Bei welchem Eingabesignal (Takt, J, K) speichert diese Schaltung? (speichern)
- Bei welchen Eingabesignalen (Takt, J, K) wird Q = 0 eingeschrieben? (löschen)
- Bei welchen Eingabesignalen (Takt, J, K) wird Q = 1 eingeschrieben? (setzen)
- Wann toggelt das Flipflop? (toggeln)

Hinweis: Sie müssen möglicherweise mehrere Taktflanken eingeben, um ein genaues Ergebnis zu erhalten.

Taktflanke	J	K	Q	¬Q	Funktionalität
	0	0			
	0	1			
	1	0			
	1	1			