Kapitel 3

Programmable Logic Array (PLA)

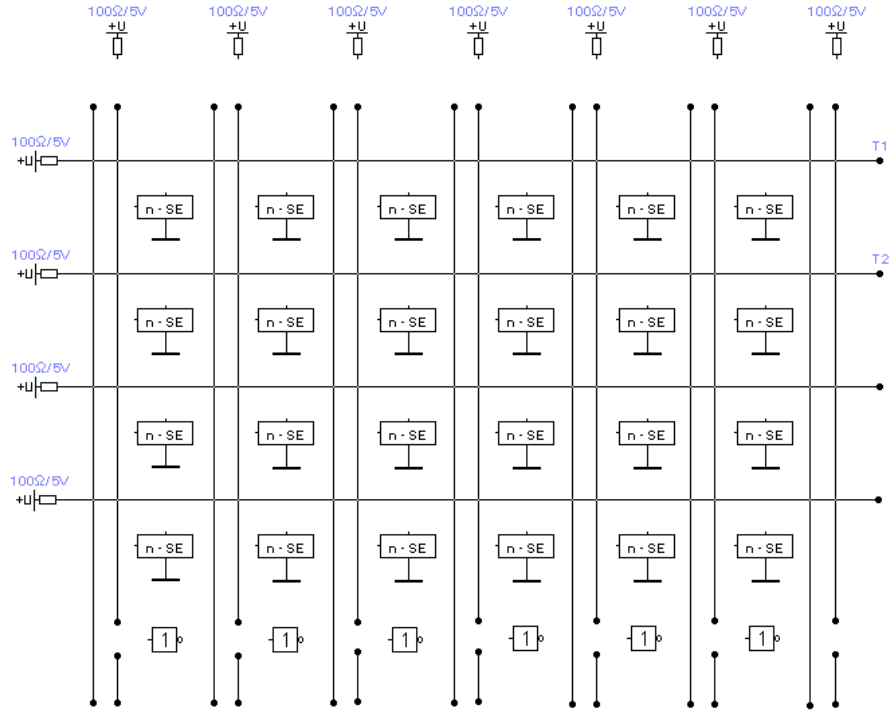
Die Idee eines PLA ist, dass bei der Chipherstellung ein homogenes Feld von Transistoren erzeugt wird. Die eigentliche Funktionalität wird dann durch Konfiguration und Programmierung durch den Benutzer festgelegt.

Wir benutzen im nachfolgenden Versuch ein PLA zur Implementierung zweier boolscher Funktionen. Dazu einige Vorüberlegungen.

Wie Sie sich vielleicht erinnern, kann man für jede boolsche Funktion mindestens eine zweistufige Schaltung angeben, die diese realisiert. Man erzeuge dazu die DNF (geht immer und ist eindeutig) und wähle soviele UND-Gatter aus, wie die DNF Terme hat. Diese UND-Gatter haben soviele Eingänge wie die Terme Literale haben. Besteht ein Literal aus einer negierten Variablen, muss noch ein Negationsgatter vorgeschaltet werden. Zum Schluss leite man alle Ausgänge der UND-Gatter in ein grosses ODER-Gatter, dessen Ausgang den Funktionswert liefert.

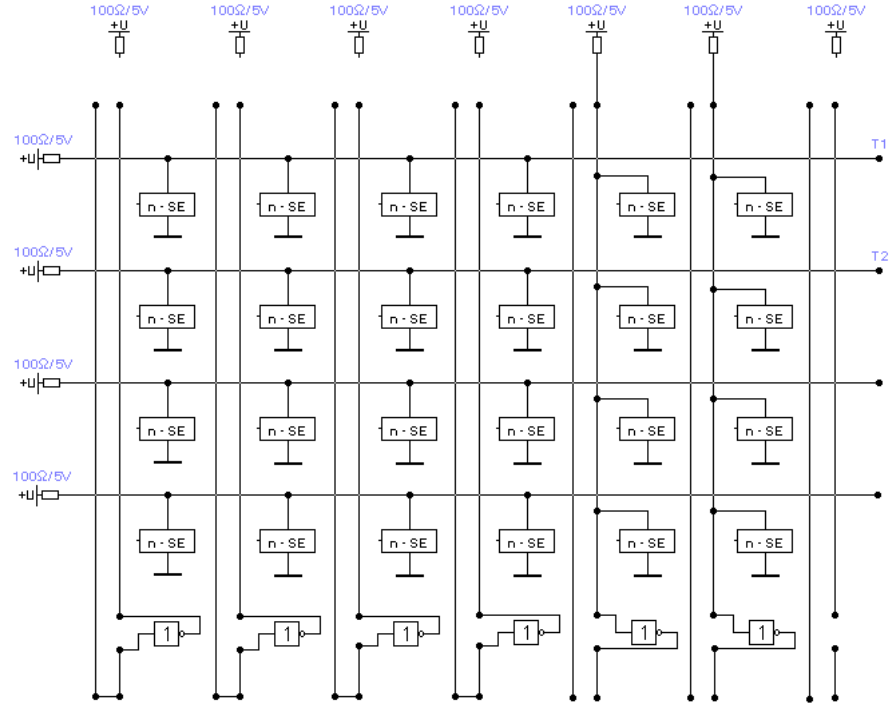
Obiges Verfahren garantiert immer eine Lösung. Möglicherweise ist diese aber sehr aufwendig, deshalb optimieren wir die Darstellung einer boolschen Funktion, um eine möglichst kurze disjunktive Form zu erhalten. Minimiert werden dadurch sowohl die Anzahl der Eingänge pro UND-Gatter als auch die Anzahl 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:



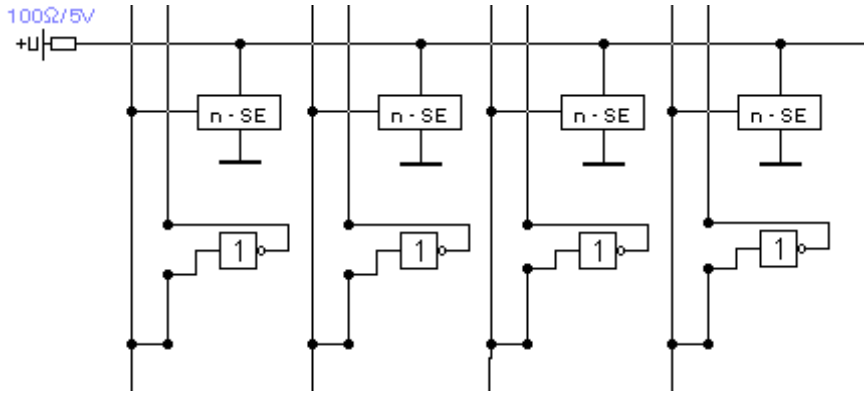
Unkonfiguriertes PLA

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



Konfiguriertes PLA

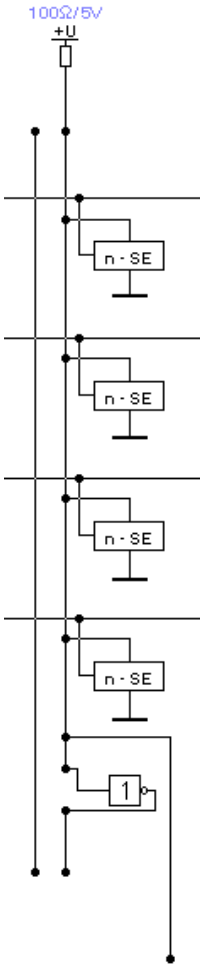
Im obigen Beispiel wurde ein 4 x 4 UND-Feld und ein 4 x 2 ODER-Feld konfiguriert. Dazu wurden die Transistoren zu Gattern zusammengeschaltet. Betrachten wir als erstes eine Zeile des UND-Feldes:



Die 100Ω/5V Quelle dient als Pullup-Widerstand für eine Wired-Schaltung. Die Drain-Anschlüsse aller Transistoren sind auf einem „Draht“ zusammengeschaltet (der auch den Ausgang des Gatters bildet), die Source-Anschlüsse sind auf Masse gelegt. Die senkrechten Leitungspaare sind die Eingänge, jede Variable wird normal und negiert zur Verfügung gestellt. Für die Analyse verbinden wir die Gate-Anschlüsse mit den nicht negierten Eingängen und ignorieren die Negationen. Der Ausgang des Gatters sei rechts oben.

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 Gate-Anschlüsse sind mit den waagerechten Ausgängen des UND-Feldes verbunden und stellen die Eingänge dieses Gatters dar. Zur Analyse ignorieren wir die Negation und greifen die Funktion direkt hinter dem letzten Transistor ab.

Welche logische Funktion hat dieses Gatter?

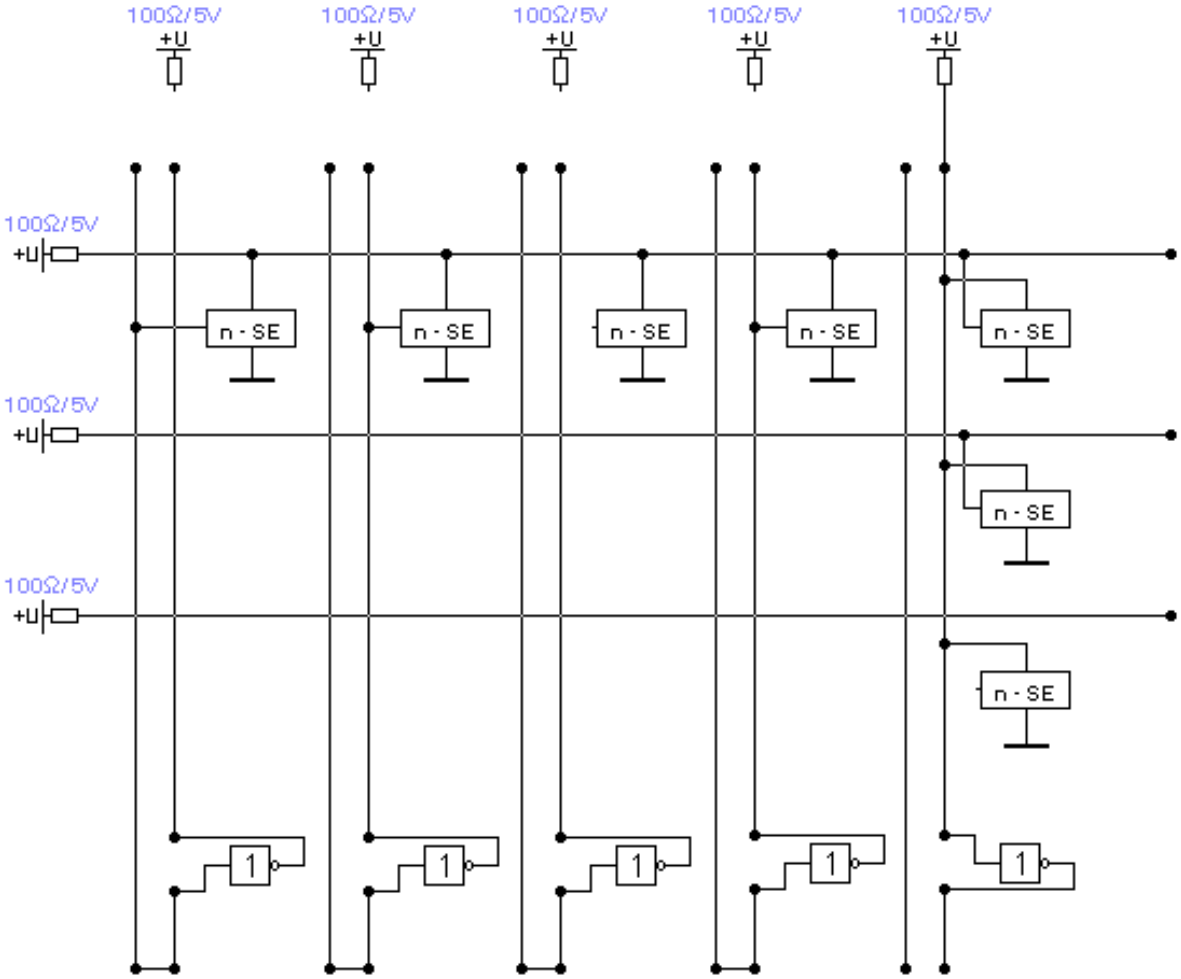
Offensichtlich haben wir ein Problem, denn in beiden Fällen handelt es sich weder um ein UND- noch um ein ODER-Gatter. Soviel sei aber verraten, dass obige Gatter jeweils alleine für sich schon einen vollständigen Bausteinsatz bilden.

Nehmen Sie nun zu den gefundenen Gatterfunktionen noch den Negationsoperator hinzu (obwohl das nicht nötig wäre) 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 müssen Sie das PLA programmieren. Programmieren bedeutet:

- im UND-Feld: die GateAnschlüsse 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.

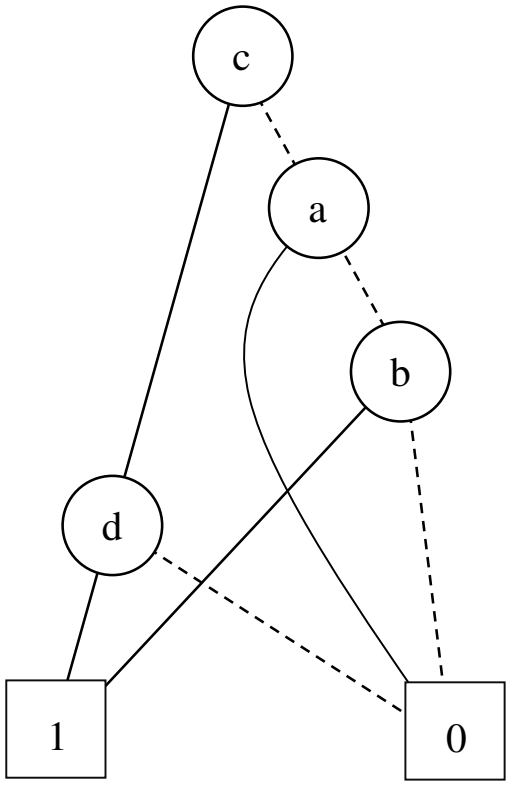
Beispielsweise (Ausschnitt mit nur einer Zeile und einer Spalte):



Versuch 300 PLA-Programmierung

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

f



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

Das OBDD ist bereits optimal. : Optimierung mit KV-Diagramm:

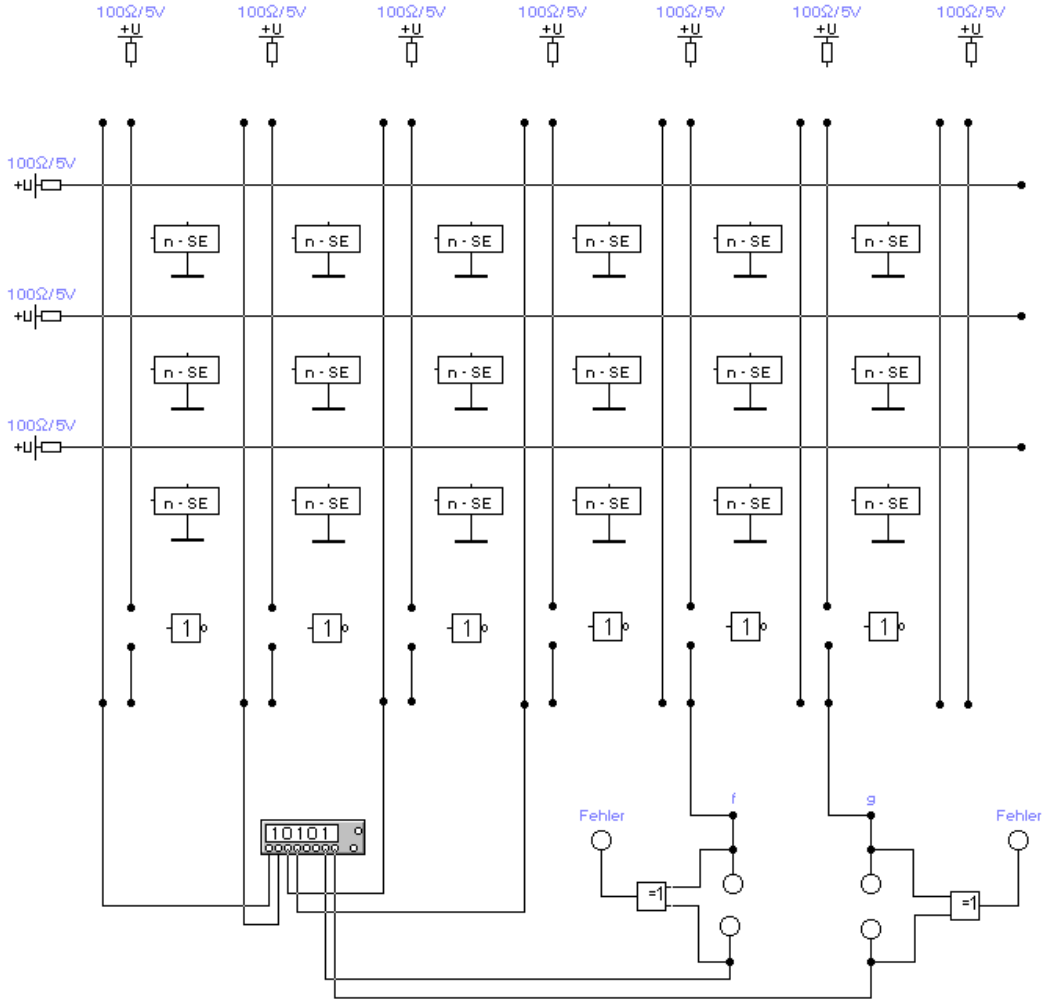
Minimale disjunktive Form:

f(a,b,c,d) =

Minimale disjunktive Form:

g(a,b,c,d) =

Programmieren Sie nun ein PLA, das beide Funktionen realisert. In der Datei v300 steht Ihnen hierfür ein 3 x 6 PLA zur Verfügung. Für ein grösseres hat das Uni-Budget leider nicht gereicht, sie müssen damit auskommen.



Legen Sie zunächst das UND-Feld und das ODER-Feld fest, indem Sie das PLA entsprechend **konfigurieren**. Programmieren sie nun beide Felder auf Grundlage der allgemeinen Überlegungen zu Beginn dieses Abschnittes, und den von Ihnen konkret berechneten und in ihrer Darstellung optimierten Funktionen f und g.

Testen Sie nun Ihre Schaltung. Offnen Sie den Bitmustergenerator und laden Sie die Datei v300 pla.dp. Die ersten vier Spalten liefern die Stimuli. Die zugehörigen Ausgänge sind mit den Eingängen Ihres PLAs verbunden. Die letzten beiden Spalten enthalten die Funktionswerte für f und g. Diese Ausgänge führen zu Lämpchen, die den Ausgängen Ihres PLAs gegenüberliegen.
Starten Sie den Bitmustergenerator mit CYCLE. 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

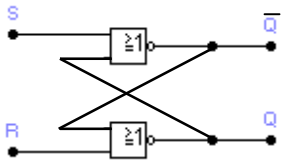
Mit logischen Gattern aufgebaute Netzwerke lassen sich in zwei Klassen einteilen:

• 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 Boolsche Algebra.

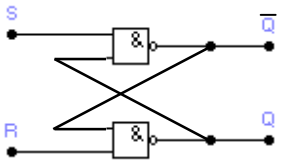
• Schaltwerke (Sequentielle Schaltungen) sind Schaltungen mit Speichervermögen, d.h. die zu einem Zeitpunkt *t* an den Ausgängen herrschenden Binärzustände hängen von den Binärzuständen an den Eingängen und von bestimmten internen Binärzuständen ab, die zu diesem Zeitpunkt in der Schaltung stabil herrschen. 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 Boolschen 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 einzunehmen 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



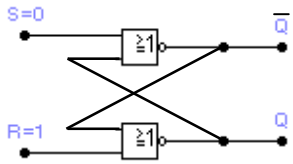
NAND-Basis-Flipflop

Theorie 303 Analyse eines NOR-Basis-Flipflops.

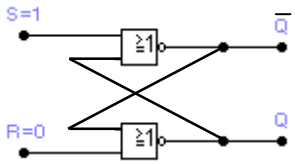
Sie sollen in den folgenden Schaltbilder die Funktionsweise eines NOR-Basis-Flipflops analysieren. Überlegen Sie, welche der vier Funktionalitäten (setzen, löschen, speichern, irregulärer Zustand) das NOR-Basis-Flipflop in den jeweiligen Fällen aufweist. Ein Flipflop befindet sich im irregulären Zustand,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

Zur Analyse berechnen Sie das Verhalten der Schaltung in allen vier nachfolgenden Fällen. Setzen Sie einmal Q = 0 und dann Q = 1.

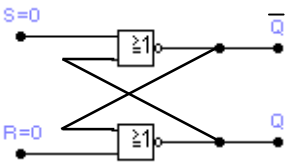
Fall 1



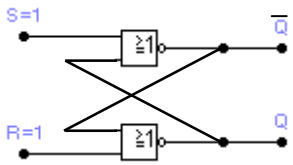
Fall 2



Fall 3



Fall 4



Ergänzen Sie die Einträge in der Ansteuertabelle des NOR-Basis-Flipflop.

NOR-Basis-Flipflop

S	R	Q	$\overline{Q}$	Funktionalität
0	0			
0	1			
1	0			
1	1			

Versuch 305 NAND-Basis-Flipflop

Überlegen Sie, nachdem Sie die Funktionstabelle des NOR-Basis-Flipflops analysiert haben, wie diese für ein NAND-Basis-Flipflop aussehen könnte. Gibt es Unterschiede in der Funktionsweise der beiden Basis-Flipflops?

S	R	Q	$\overline{Q}$	Funktionalität
0	0			
0	1			
1	0			
1	1			

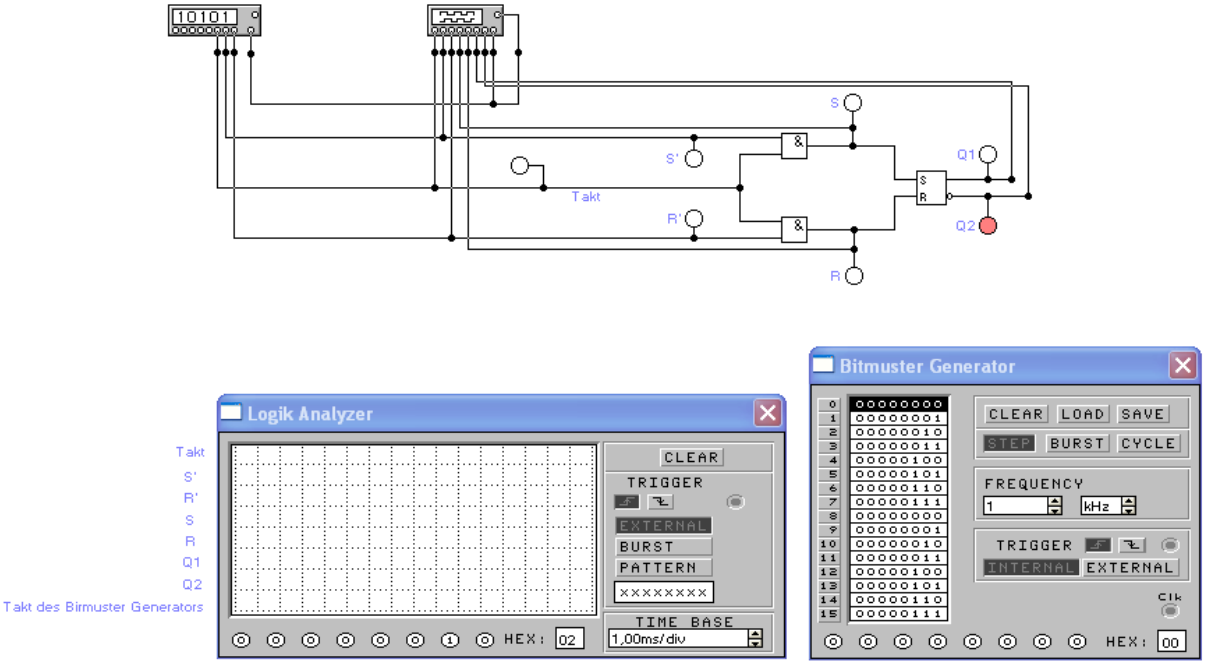
Das in EWB eingebaute RS-Flipflop entspricht einem der beiden Basis-Flipflops. Welchem? Testen Sie das Flipflop in der Datei v305.

EWB-RS-FF =

Das in EWB-RS-Flipflop fällt nach jedem Einschalten des Simulators bei nicht angeschlossnen Eingängen (R = S = 0) immer in denselben Grundzustand. Welchen?

Q = **$\overline{Q}$ =**

Versuch 310 Taktzustandsgesteuertes SR-Flipflop



In der Schaltung in Datei v310 bilden die beiden AND-Gatter zusammen mit dem RS-Flipflop ein „taktzusandsgesteuertes SR-Flipflop“. Nur wenn der Taktzusand „Binär 1“ ist, „öffnen“ die beiden vor das SR-Flipflop geschalteten AND-Gatter.

Lassen Sie das Oszillogramm auf dem Logikanalysator durch mehrfaches drücken der Taste „Step“ an der Frontplatte des Bitmustergenerators entstehen. Analysieren Sie das entstandene Oszillogramm.

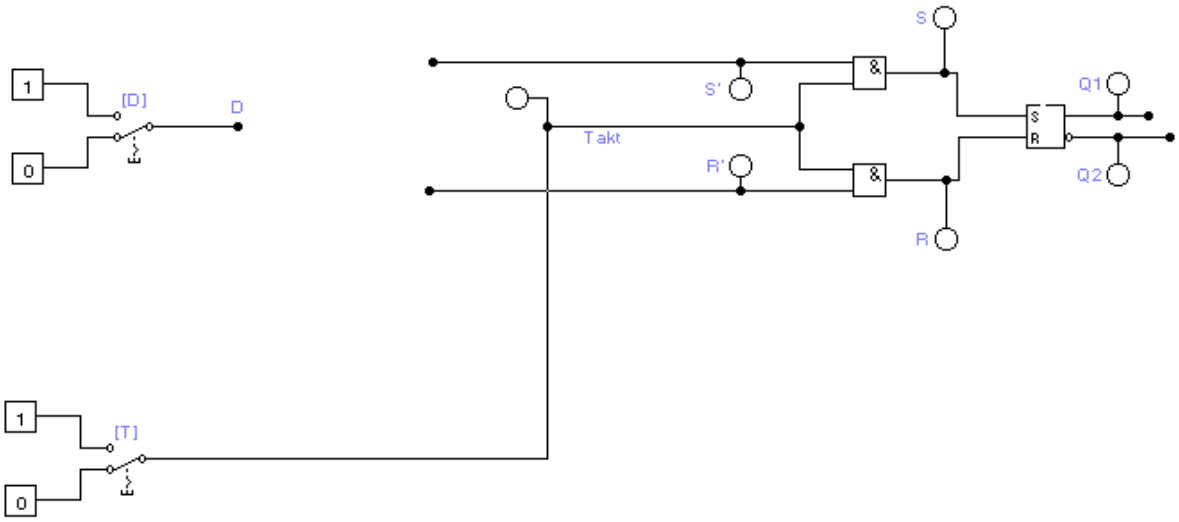
Gewinnen Sie aus dem Oszillogramm die Werte der folgenden Funktionstabelle und analysieren Sie für jede Zeile dieser Tabelle, ob das Flipflop setzt, löscht, speichert oder sich in einem irregulären Zustand befindet. Beim Einschalten der Simulation fällt das Flipflop in den Zustand Q1=0 und Q2=1. Vervollständigen Sie die Tabelle.

C S' R'	S	R	Q1	Q2	Funktionalität
0 0 0	0	0	0	1	speichern
0 0 1	0	0	0	1	speichern
0 1 0					
0 1 1					
1 0 0					
1 0 1					
1 1 0					
1 1 1					

Versuch 313 Taktzustandsgesteuertes D-Flipflop

Ein D-Flipflop besitzt nur einen Eingang D. Es hat nur zwei Betriebszustände:
Bei Takt = 0 behält es seinen bisherigen Zustand, es speichert.
Bei Takt = 1 übernimmt es den an D anliegenden Wert, es setzt also bei D=1 oder es löscht bei D=0.

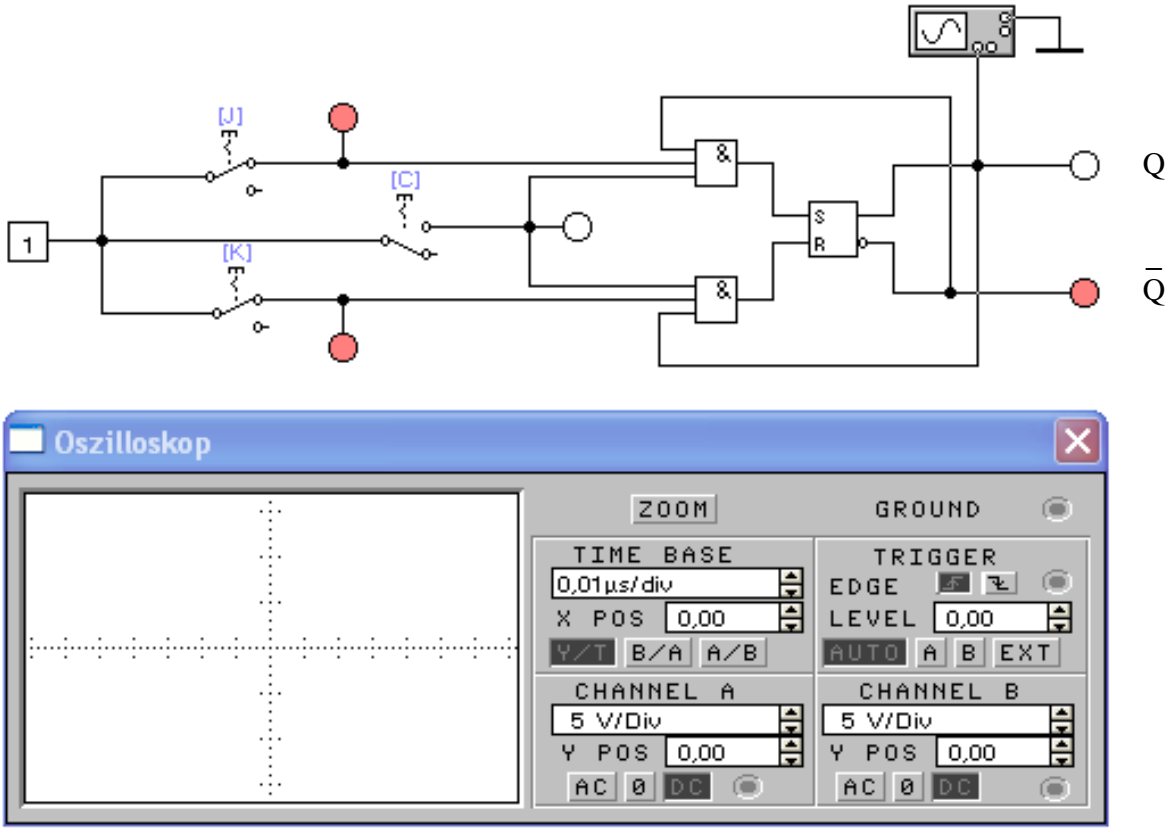
Ergänzen Sie die Schaltung in der Datei v313 so, dass sie obiges Verhalten zeigt.



Das SR-Fliflop ermöglicht uns 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. Das ist natürlich schade. Es wäre doch vom Vorteil, anstatt einer ungewollten Funktionalität, die sich bei dieser Belegung ergibt, eine neue und sinnvolle zu gewinnen. Könnten wir das taktzustandsgesteurte SR-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 315 Taktzustandsgesteuertes JK-Flipflop

Durch die beiden Rückkopplungen erhalten wir ein JK-Flipflop. Laden Sie die Datei v315. Setzen Sie zuerst J = K = 1 und takten Sie dann mit C von 0 auf 1. Zoomen Sie das Oszilloskop und schalten Sie die Simulation ein. Was beobachten Sie?



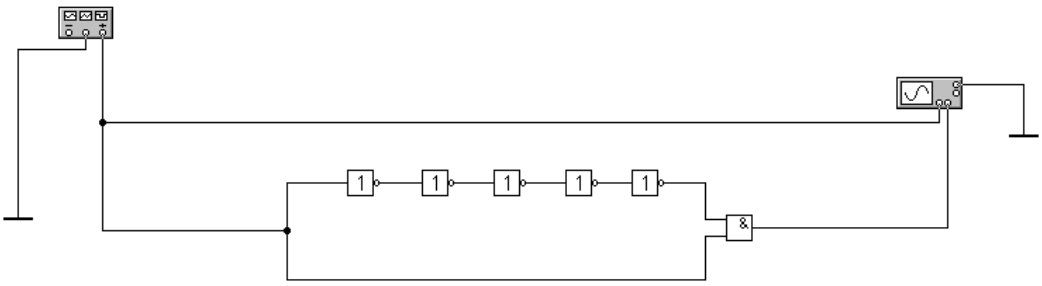
Das Flipflop zeigt drei mögliche Verhaltensweisen:

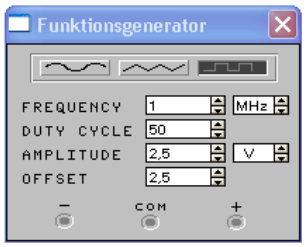
- es schwingt oder
- es speichert oder
- es wird gesetzt / gelöscht.

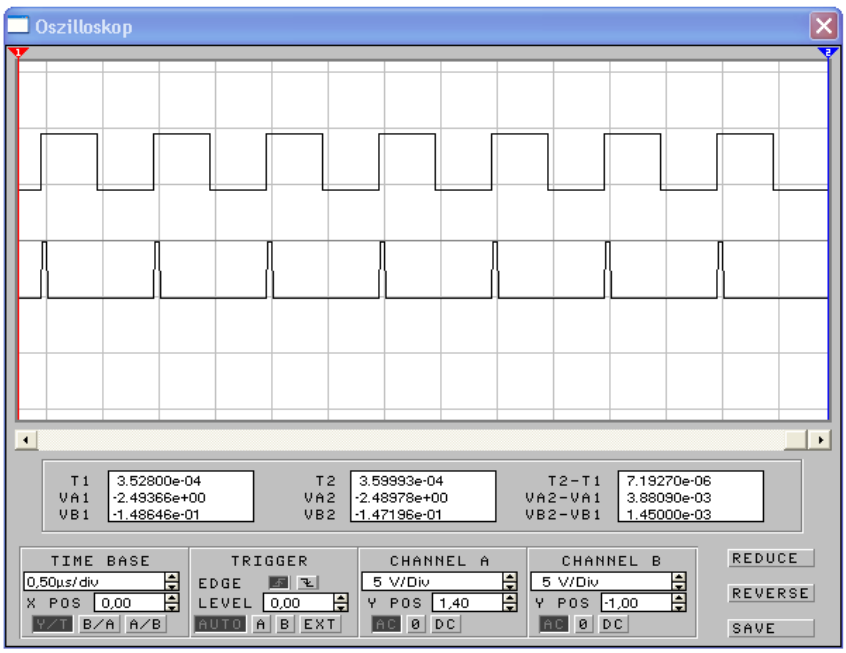
In obigem Versuch haben wir unsere Schaltung so erweitert, dass wir den irregulären Zustand beseitigt haben. Leider gibt es Fälle in welchen unsere Schaltung anfängt zu schwingen. Unser Ziel, alle möglichen Belegungen für das Flipflop sinnvoll nutzen zu können, ist noch nicht erreicht. Was verursacht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

Versuch 320 Taktflankensteuerung

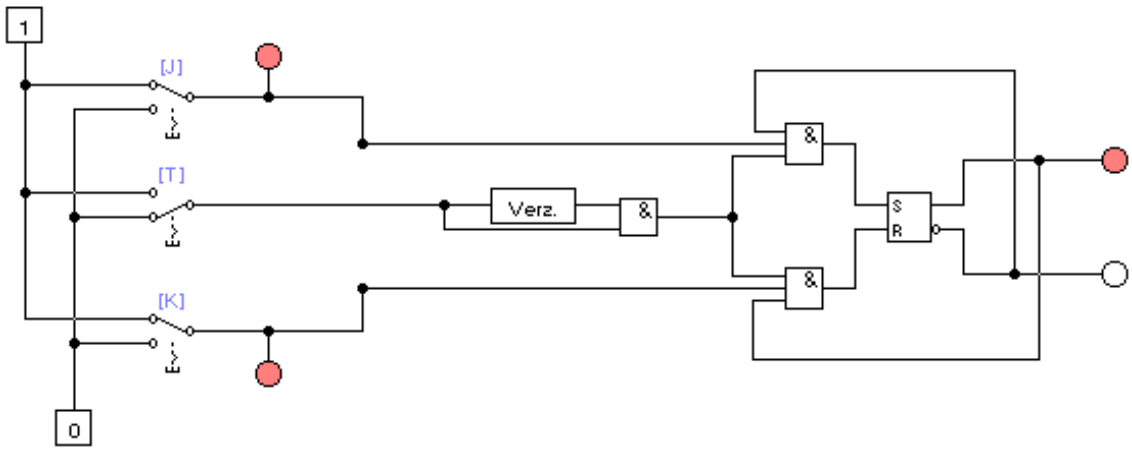
Die Schaltung in Datei v320 bewirkt, dass an ihrem Ausgang bei einer aufsteigenden (positiven) Flanke nur ein krurzer Impuls (peak) entsteht. Dadurch wird ein nachgeschaltetes Flipflop nur für einen kurzen Augenblick angesteuert und Schwingungen werden vermieden.
Schalten Sie die Simulation ein und analysieren 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 dazunimmt?







Versuch 325 Taktflankengesteuertes JK-Flipflop



Das Makro „Verz.“ enthält wie im vorherigen Versuch fünf hintereinander geschalteteInverter.

Geben Sie bei [J,K] = [1,1] eine Folge von Taktimpulsen ein. Wie verhält sich das Flipflop? Man bezeichnet dieses Verhalten mit dem Verb/Substantiv „toggle“ (toggle switch: Kippschalter). Weshalb?

Reagiert die Schaltung auf die bei J und K eingegebenen Zustände bei

- Eingabe einer positiven Taktflanke(↑) oder,
- Eingabe einer negativen Taktflanke(↓) oder,
- wenn der Binärzustand am Eingang „Takt“ konstant 1 oder konstant 0?

Bestimmen Sie in der folgenden Tabelle die Werte für Q und ¬Q, wenn die Schaltung reagiert.

J	K	Q	¬Q
0	0		
0	1		
1	0		
1	1		

- Bei welchen Eingabesignalen speichert diese Schaltung?
- Bei welchen Eingabesignalen wird ein Datum* eingeschrieben?
- Wann toggled das Flipflop?

*Datum ist der Singular von Daten, hier 0 oder 1.