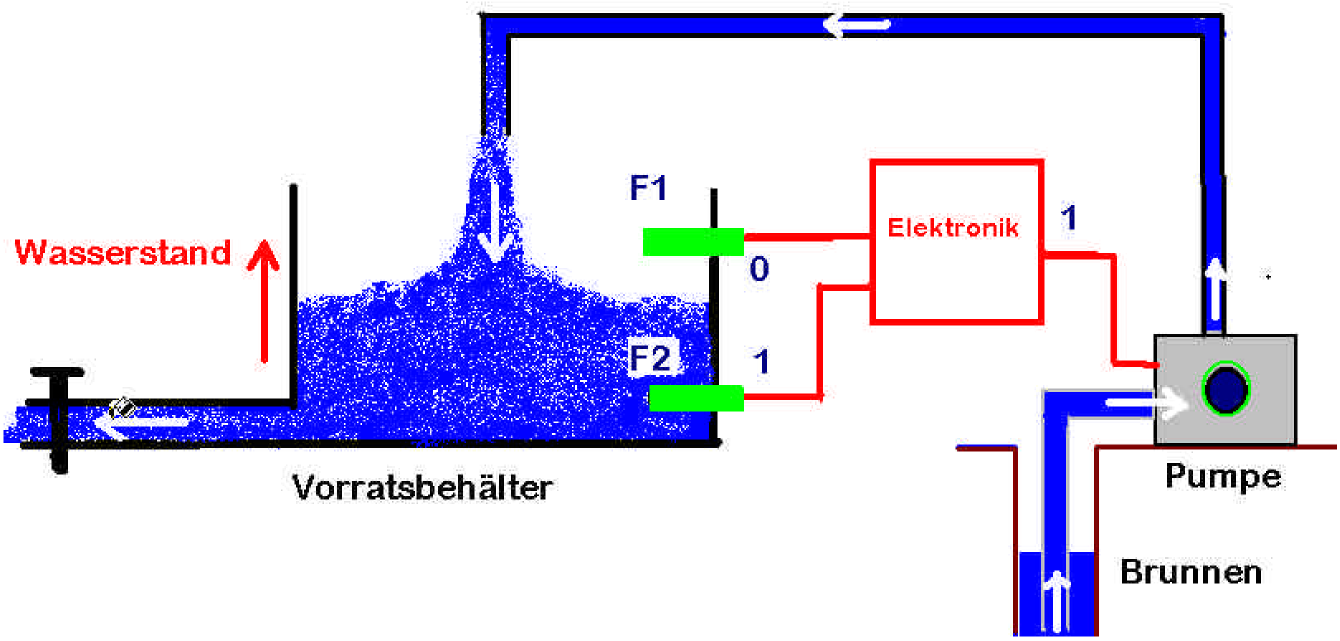
Füllstandsregelung

Zwei Feuchtigkeitsfühler (trocken F=0; feucht F=1) sollen zusammen mit einer geeigneten Elektronik dafür sorgen, dass das Wasser im Vorratsbehälter niemals unter das Niveau des Fühlers F2 sinkt und niemals über das Niveau des Fühlers F1 steigt.



Funktionstabelle:

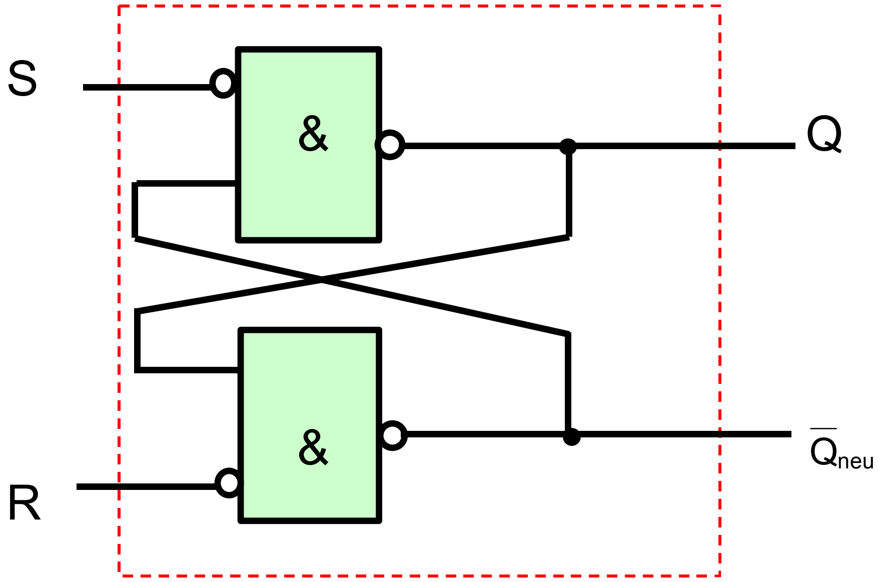
F1	F2	P	
1	1	0	
0	1	0	← bei fallendem Wasserstand
0	0	1	
0	1	1	← bei steigendem Wasserstand
1	1	0	
1	0	0	← unmöglicher Zustand (→ P=0) Fühler defekt !

Dies ist keine Tabelle einer Schaltfunktion!

Wir benötigen einen Speicherbaustein, der sich merken kann, ob der Wasserstand steigend oder fallend ist

Speicherbausteine (FLIP-FLOP's)

Das statische S-R-Flip-Flop



Funktionstabelle:

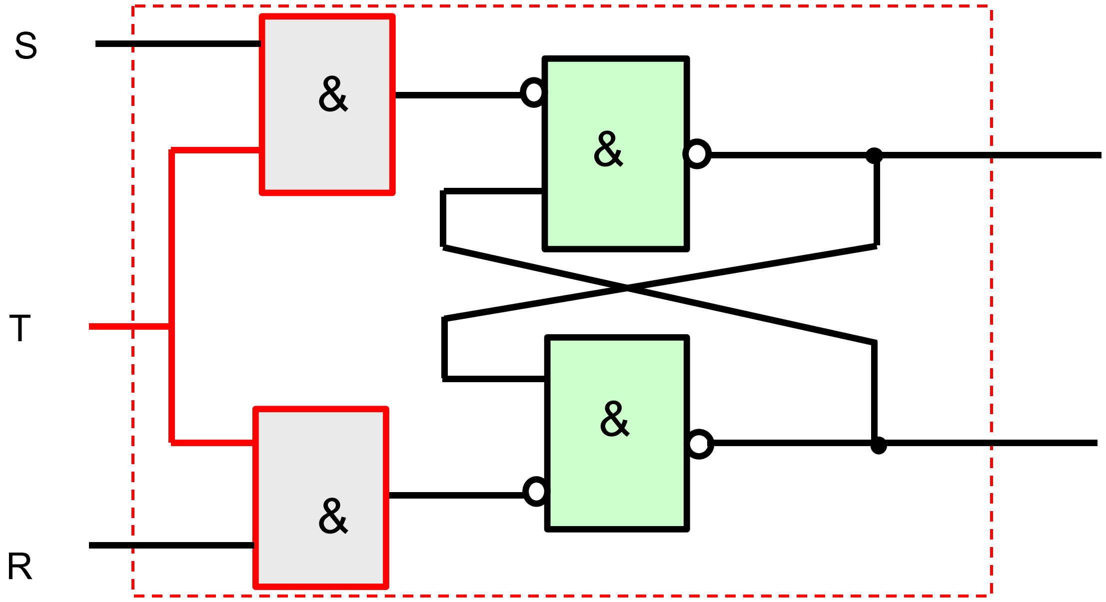
S	R	Q_{alt}	Q_{neu}	$\overline{Q}_{neu}$	Bemerkung
0	0	0	0	1	keine Änderung , Speicherfall
0	0	1	0	1	keine Änderung , Speicherfall
0	1	0	**0**	1	Reset
0	1	1	**0**	1	Reset
1	0	0	**1**	0	Set
1	0	1	**1**	0	Set
1	1	0	1	1	irregulär, vermeiden
1	1	1	1	1	irregulär, vermeiden

Schaltsymbol:

Testschaltung mit LOCAD : S-R-FF_statisch.LWS

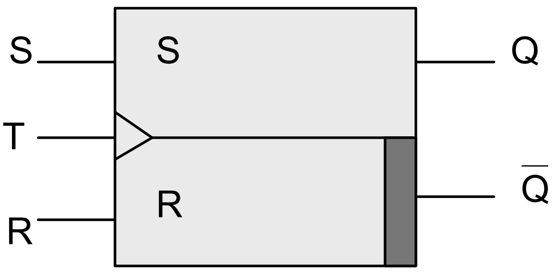
Nachteil dieses Speicherbausteins: Jede Änderung an den S-R-Eingängen beeinflusst unmittelbar das Verhalten der Ausgänge Q und $\overline{Q}$.
Einen Ausweg bilden taktgesteuerte Flip-Flops. Über eine Torschaltung wird gesteuert, ob eine Änderung des FF erreicht werden soll oder nicht :

Das taktgesteuerte R-S-Flip-Flop



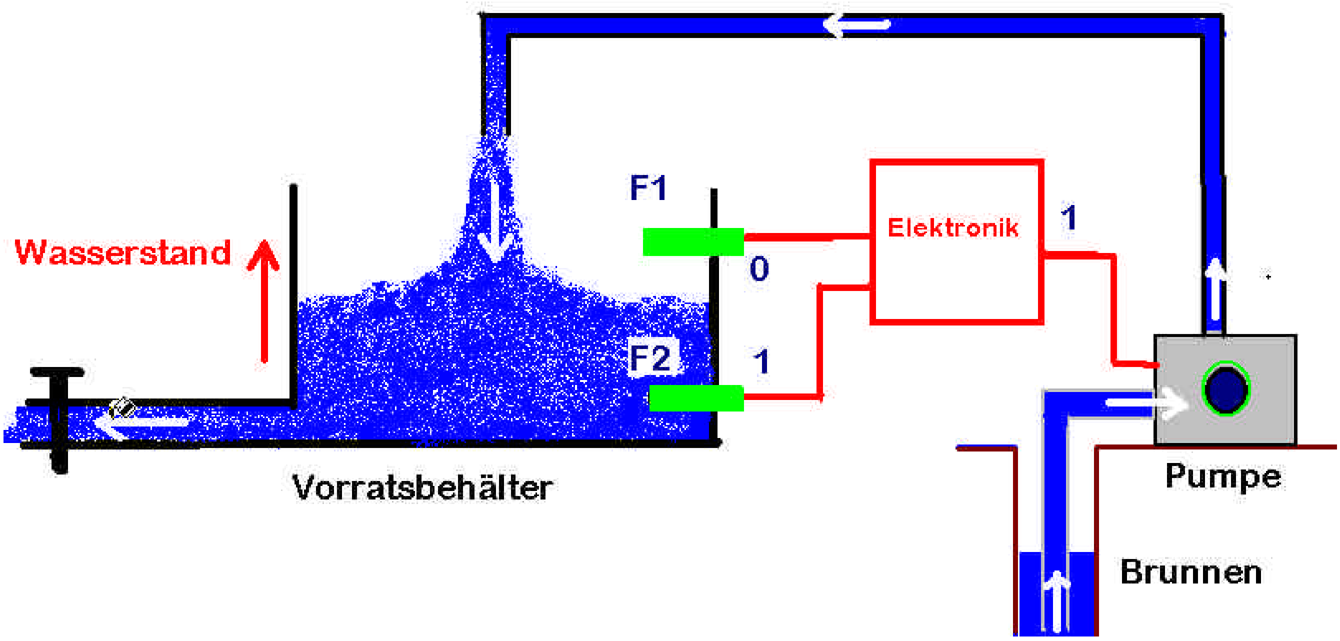
Ist T=0, dann haben beide FF-Eingänge den Wert 0 und damit ändert sich am FF nichts. Ist T=1, dann wird die durch S und R gewünschte Änderung am FF erreicht.

Schaltsymbol:



Testschaltung mit LOCAD : S-R-FF.LWS

Die Lösung des Pumpenproblems:



Zur Lösung verwenden wir ein torgesteuertes S-R-FF. Mit F1=1 und F2=1 wird am FF ein Set ausgelöst und mit F1=0 und F2=0 ein Reset:

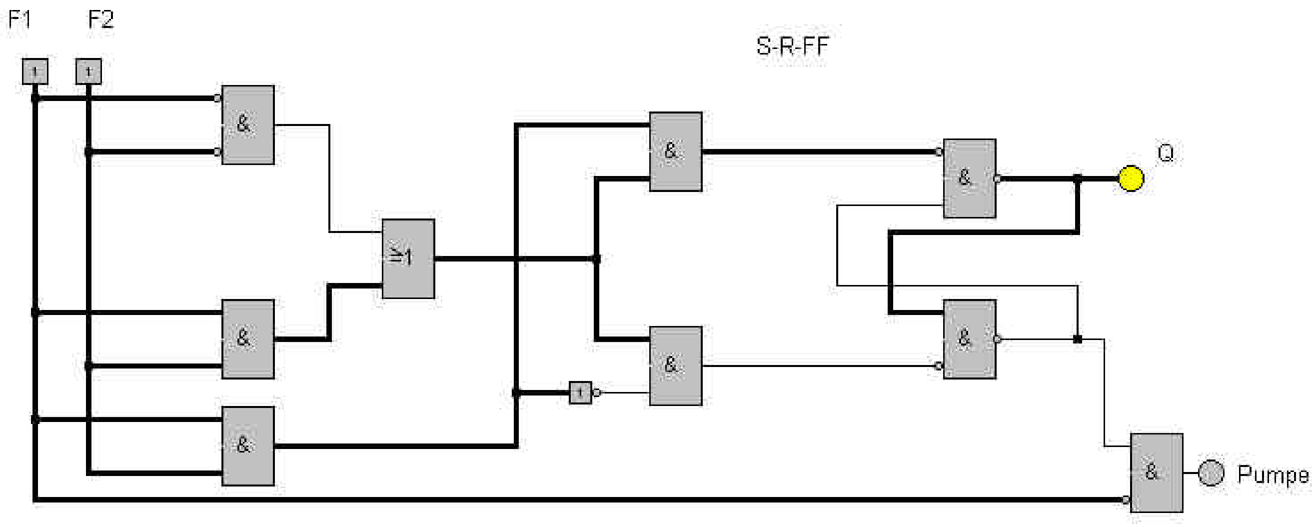
F1	F2	FF	P	
1	1	1	0	← Set
0	1	1	0	
0	0	0	1	← Reset
0	1	0	1	
1	1	0	0	irreguläre Zustände
1	0	0	0	
1	0	1	0	
0	0	1	0	

Für die zugehörige Schaltfunktion gilt dann

$$P = (\overline{F1} \wedge \overline{F2} \wedge \overline{FF}) \vee (\overline{F1} \wedge F2 \wedge \overline{FF}) = \overline{F1} \wedge \overline{FF}$$

D.h. Die Pumpe ist nur dann eingeschaltet, wenn der Fühler F1 trocken ist und das FF nicht gesetzt ist.

Innenschaltung der Elektronik :

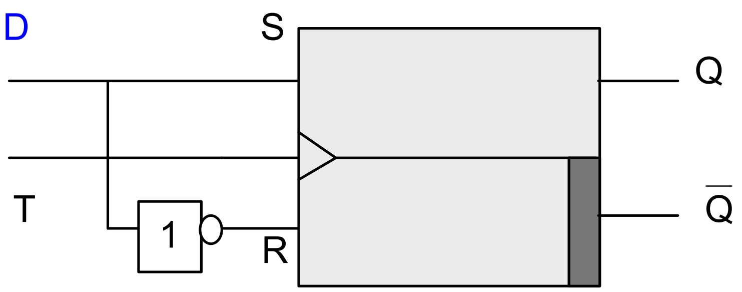


Testschaltung mit LOCAD : Pumpensteuerung.LWS

Das D-Flip-Flop

An den S und R-Eingängen kann jetzt nur der Setzfall (D=1) bzw. der Löschfall (D=0) auftreten.

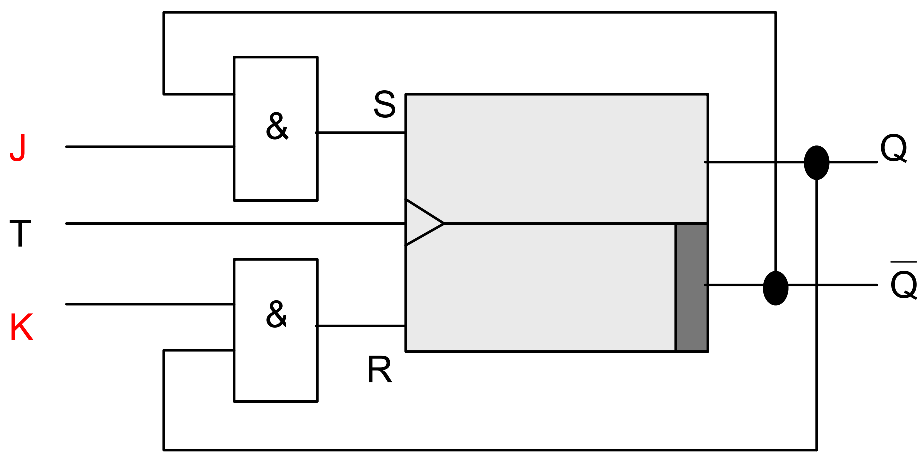
Das D-FF übernimmt und speichert das Datenbit 0 oder 1 , das am D-Eingang anliegt immer dann, wenn der Takteingang T auf 1 gesetzt wird.

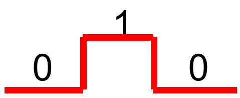


Testschaltung mit LOCAD : D-FF.LWS

Das J-K-Flip-Flop

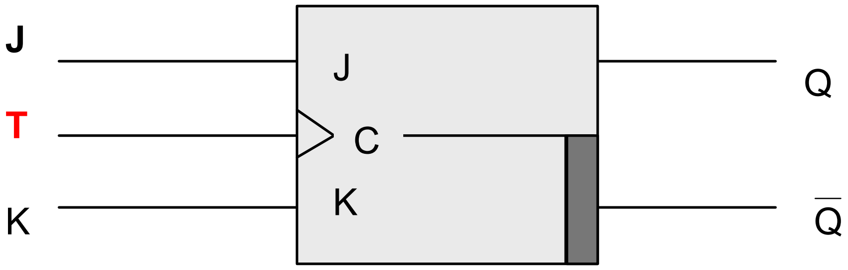
Die Q und $\overline{Q}$-Ausgänge werden über UND-Bausteine zusammen mit den J- und K-Eingängen an die S- und R-Eingänge eines S-R-FF gelegt.



Erreicht jetzt ein Taktimpuls  den Takteingang, dann hat man folgendes Verhalten:

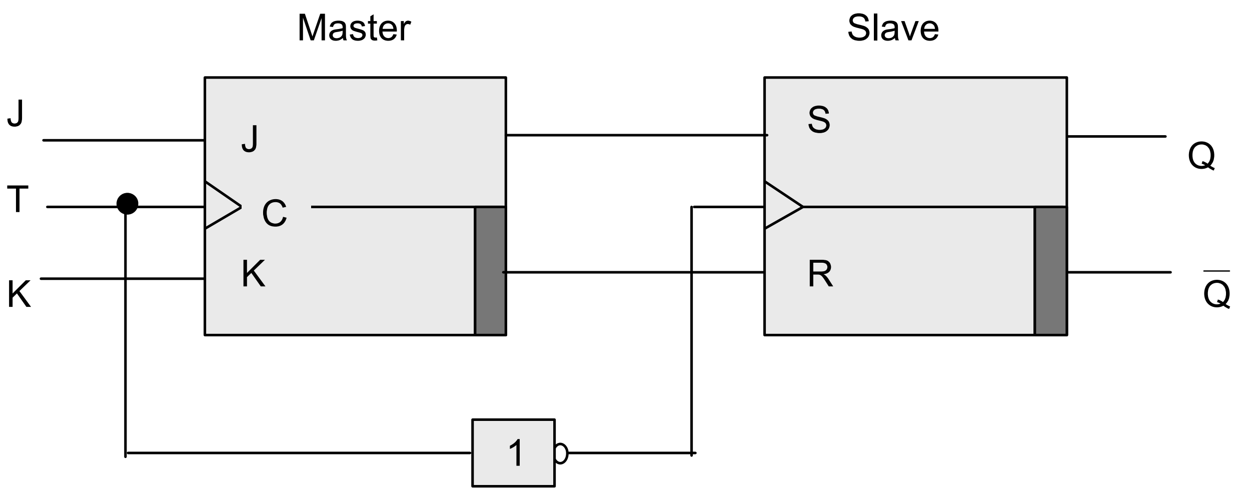
		vor dem Taktimpuls T=0		während des Taktimpulses T=1		nach dem Taktimpuls T=0		Bemerkung
J	K	Q	$\overline{Q}$	S	R	Q	$\overline{Q}$	
0	0	0	1	0	0	0	1	speichern
0	0	1	0	0	0	1	0	speichern
0	1	0	1	0	0	0	1	zurücksetzen
0	1	1	0	0	1	0	1	zurücksetzen
1	0	0	1	1	0	1	0	setzen
1	0	1	0	0	0	1	0	setzen
1	1	0	1	1	0	1	0	wechseln
1	1	1	0	0	1	0	1	wechseln

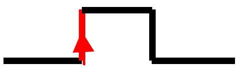
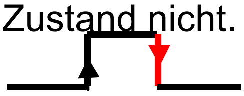
Schaltsymbol eines Taktimpulsgesteuerten J-K-FF



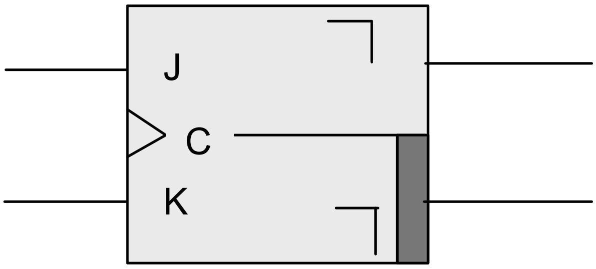
Testschaltung mit LOCAD : J-K-FF.LWS

Das Takt<u>flanken</u>gesteuerte J-K-Master-Slave-FF



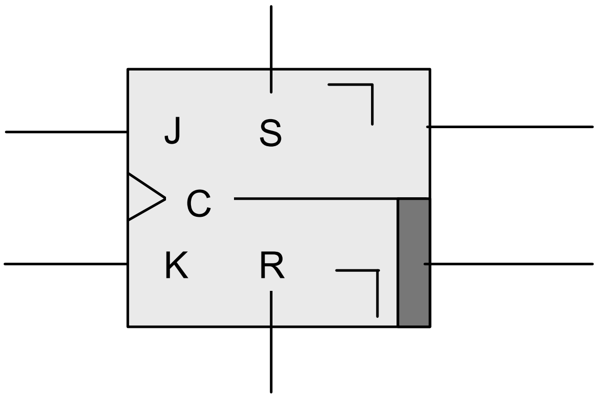
Bei der positiven (ansteigenden) Flanke des Taktsignals  übernimmt der Master den Wert, der durch die Belegung der J-K-Eingänge vorbestimmt ist. Der Sklave ändert dabei seinen Zustand nicht. Erst bei der negativen (abfallenden) Flanke des Taktsignals  übernimmt der Sklave den Zustand des Masters und gibt diesen aus.

Schaltsymbol und Funktionstabelle des J-K-Master-Slave-FF



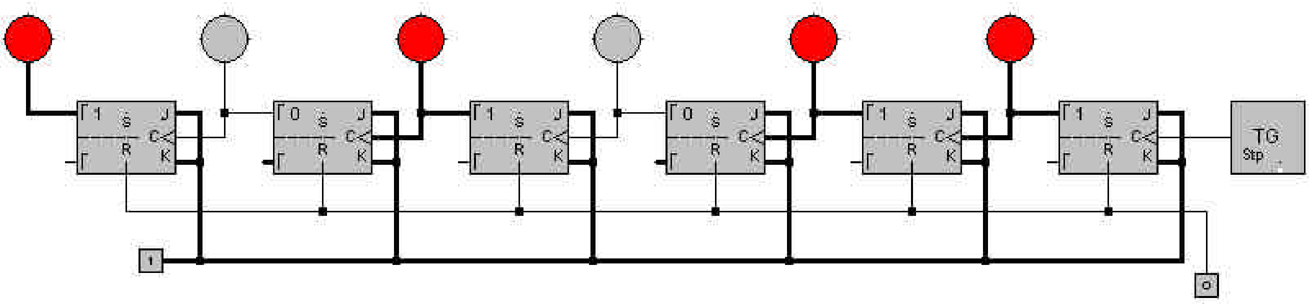
J	K	Q vor der negativen Flanke		Q nach der negativen Flanke	Bemerkung
0	0	0	1	0	speichern
0	0	1		1	
0	1	0		0	reset
0	1	1		0	
1	0	0	0	1	set
1	0	1		1	
1	1	0		1	kippen
1	1	1		0	

Das Universal-FF, das wir in Zukunft verwenden wollen hat noch zwei zusätzliche statische R- (reset) und S- (set) Eingänge, mit denen das FF taktunabhängig zu jeder Zeit gesetzt, bzw. zurückgesetzt werden kann:



S	R	
0	0	keine Funktion
0	1	reset
1	0	set
1	1	vermeiden

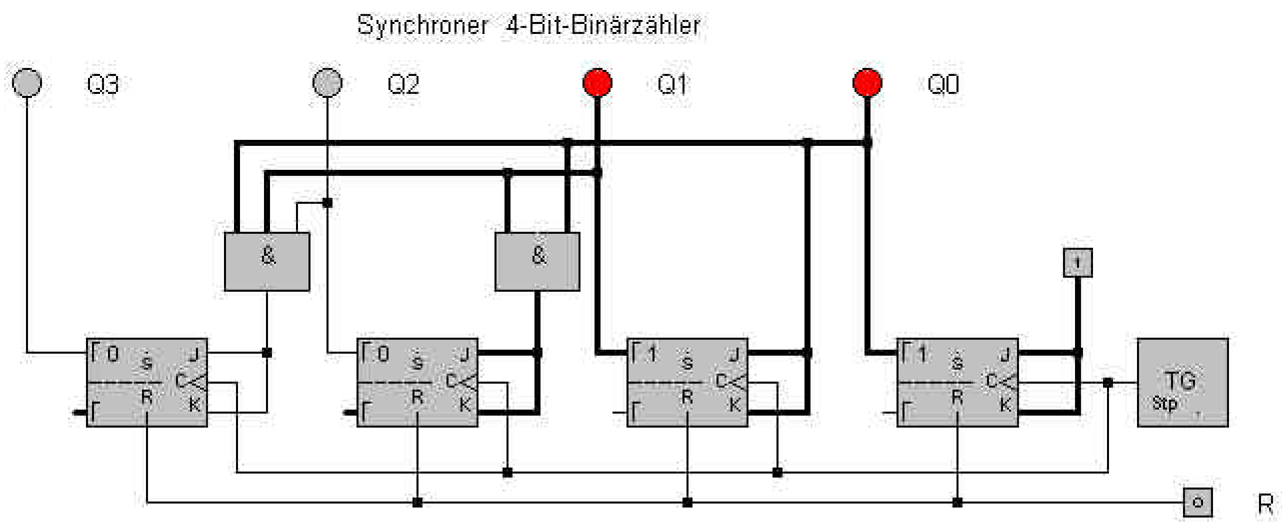
Aufbau eines asynchronen Binärzählers



Testschaltung mit LOCAD : BinaerZaehler.LWS

Alle FF sind mit J=K=1 auf kippen vorbereitet.
Das nachfolgende FF kippt nur dann, wenn am vorangehenden FF der Q-Ausgang einen Sprung von Q=1 auf Q=0 macht (negative Flanke!)
Dieser Zählertyp wird in der Praxis so gut wie nicht verwendet.

Aufbau eines synchronen Binärzählers:



Testschaltung mit LOCAD : Binaerzaehler_sync.LWS

Die Darstellung negativer Dualzahlen als Zweierkomplement

	1	0	0	1	1	0	1	0
-	0	1	0	0	1	0	0	1
	1						1	
	0	1	0	1	0	0	0	1

	1	5	4
-		7	3
	1		
		8	1

Da die Subtraktion nichts anderes ist, als die Addition mit der Gegenzahl, kann man sich die Darstellung der Zahl -73 im Zweiersystem leicht überlegen:

				1	0	0	1	1	0	1	0
+	 1	1	1	1	0	1	1	0	1	1	1
	 1	1	1		1	1	1	1	1		
	 0	0	0	0	1	0	1	0	0	0	1

	1	5	4
+	-	7	3
	1		
	+	8	1

Jetzt vertauschen wir den Minuenden und den Subtrahenden unserer Aufgabe :

				0	1	0	0	1	0	0	1
-				1	0	0	1	1	0	1	0
	 1	1	1		1	1	1	1	1		
	 1	1	1	1	0	1	0	1	1	1	1

		7	3
-	1	5	4
	1		
	-	8	1

Die unendlich vielen 1en geben hier an, dass das Ergebnis eine negative Zahl ist !

Damit kann man sich jetzt die Darstellung der Zahl -154 im Zweiersystem leicht überlegen:

				0	1	0	0	1	0	0	1
+	 1	1	1	0	1	1	0	0	1	1	0
				1							
	 1	1	1	1	0	1	0	1	1	1	1

			7	3
+	-	1	5	4
		1		
		-	8	1

Damit ergibt sich folgendes Verfahren zur Darstellung einer negativen Zahl im Zweiersystem:

81	=	....	0	0	0	0	1	0	1	0	0	0	1	81
$\overline{81}$	=	....	1	1	1	1	0	1	0	1	1	1	0	
$\overline{81}+1$	=	....	1	1	1	1	0	1	0	1	1	1	1	-81

Die Zahl $\overline{81}$ heißt Einerkomplement zu 81
Die Zahl $\overline{81}+1$ heißt Zweierkomplement zu 81

Merke:
Die Zahl -Z wird im Zweiersystem durch die Bildung des Zweier-komplements $\overline{Z}+1$ gebildet.

Die Darstellung negativer Zahlen mit unendlich vielen führenden 1en ist in der Praxis untauglich, da man nur endlich viele Speicherzellen zur Verfügung hat.

Daher hat man folgende Vereinbarung getroffen:

In einem Register der Breite n ist das führende BIT (MSB $\hat{=}$ most significant bit) das Vorzeichenbit

n=8:

0								positive Zahl
1								negative Zahl

Die größte positive Zahl MAXINT, die man in einem 8-Bit-Register darstellen kann ist also 127 :

0	1	1	1	1	1	1	1	127

Die kleinste negative Zahl, die man in einem 8-Bit-Register darstellen kann ist -128 :

1	0	0	0	0	0	0	0	-128

Da das MSB=1 ist, muss man noch rückkomplementieren, um den Betrag dieser negativen Zahl zu erhalten:

1	0	0	0	0	0	0	0	Zahl ist negativ !
0	1	1	1	1	1	1	1	Einerkomplement
1	0	0	0	0	0	0	0	Zweierkomplement ≙128

Also ist die dargestellte Zahl –128.

Zahlen, die man in einem 8-Bit-Register darstellen kann:

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	≙	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	≙	**1**
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	≙	**2**
0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	≙	**3**
0	**...**								
0	**...**								
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**	≙	**126**
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	≙	**127**
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	≙	**-128**
1	**0**	**0**	**0**	**0**	**0**	**0**	**1**	≙	**-127**
1	**0**	**0**	**0**	**0**	**0**	**1**	**0**	≙	**-126**
1	**...**								
1	**...**								
1	**1**	**1**	**1**	**1**	**1**	**0**	**0**	≙	**-4**
1	**1**	**1**	**1**	**1**	**1**	**0**	**1**	≙	**-3**
1	**1**	**1**	**1**	**1**	**1**	**1**	**0**	≙	**-2**
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	≙	**-1**

Die Zahl 127=01111111 wird mit **MAXINT** bezeichnet, die Zahl –128=10000000 entsprechend als **MININT.**

Merke: Bei der Datenwortbreite n gilt $Z + (\overline{Z} + 1) = 2^n$

Bsp. : n=8 **0001 1101 + 1110 0011 = 1 0000 0000 =** 2^8

Die Subtraktion von Dualzahlen:

$A - B = A + (\overline{B} + 1)$	falls A>B	
$A - B = \overline{A + (\overline{B} + 1)} + 1$	falls A<B	(Rückkomplementieren!)

Beispiel:

$100 - 73 = 100 + (\overline{73} + 1)$

100	0	1	1	0	0	1	0	0
73	0	1	0	0	1	0	0	1
$\overline{73} + 1$	1	0	1	1	0	1	1	1
$100 + (\overline{73} + 1) = 27$	0	0	0	1	1	0	1	1

$73 - 100 = -(\overline{73 + \overline{100} + 1} + 1)$

73	0	1	0	0	1	0	0	1
100	0	1	1	0	0	1	0	0
$\overline{100} + 1$	1	0	0	1	1	1	0	0
$73 + (\overline{100} + 1) = -27$	1	1	1	0	0	1	0	1
$\overline{73 + (\overline{100} + 1)} + 1$	0	0	0	1	1	0	1	1

Rechnen in Registern mit der Datenwortbreite 4 :

4 + (-3)

	0	1	0	0
+	1	1	0	1
	0	0	0	1

= 1

$3 = 0\ 0\ 1\ 1$

$\overline{3} = 1\ 1\ 0\ 0$

$\overline{3}+1 = 1\ 1\ 0\ 1$

---

3 + (-4)

	0	0	1	1
+	1	1	0	0
	1	1	1	1

= -1

$4 = 0\ 1\ 0\ 0$

$\overline{4} = 1\ 0\ 1\ 1$

$\overline{4}+1 = 1\ 1\ 0\ 0$ Ergebnis negativ

1 1 1 1 rückkomplementieren

0 0 0 0

0 0 0 1 =1

(-3)+(-4)

	1	1	0	1
+	1	1	0	0
	1	0	0	1

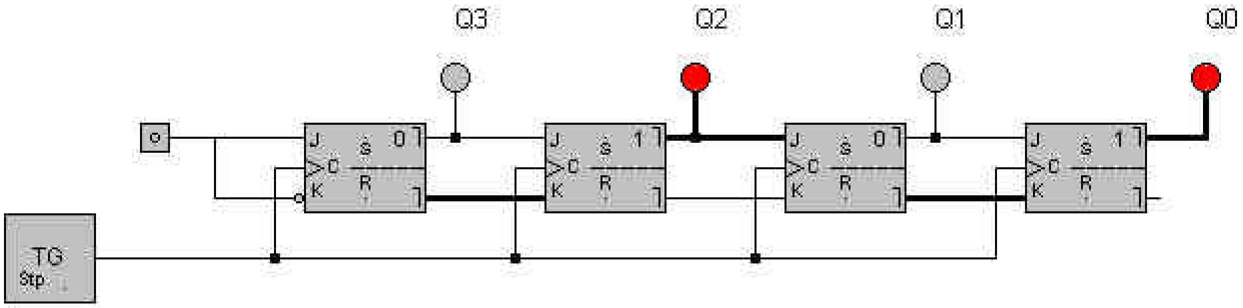
= -7

1 0 0 1 Ergebnis negativ

0 1 1 0 rückkomplementieren

0 1 1 1 =7

Aufbau eines 4-Bit-Rechtsschieberegisters mit JK-MS-FF



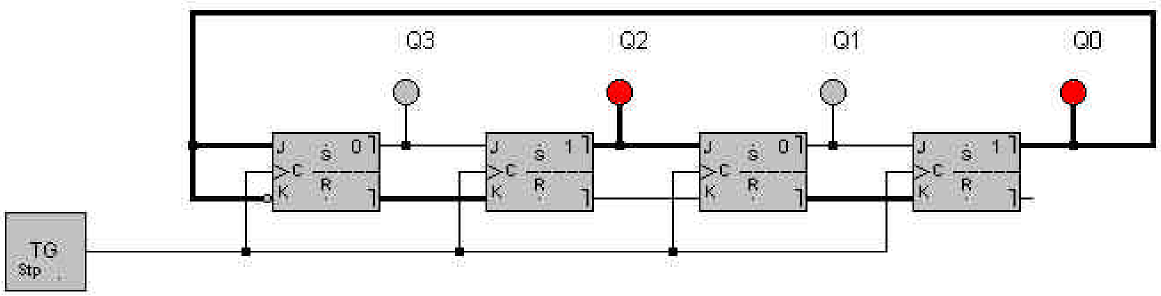
Testschaltung mit LOCAD : Rechtsschieberegister.LWS

Arbeitsweise:

	Q3	Q2	Q1	Q0
	0	1	0	1
1.Takt	0	0	1	0
2.Takt	0	0	0	1
3.Takt	0	0	0	0
4.Takt	0	0	0	0

Das Datenwort 0101 wird bei jedem Taktimpuls um eine Position nach rechts verschoben . Das am weitesten rechts stehende Bit geht verloren. Von links wird das Register mit Nullen aufgefüllt, da das FF3 auf Reset vorbereitet ist.

Ringschieberegister (Rechtschieberegister als Akkumulator)

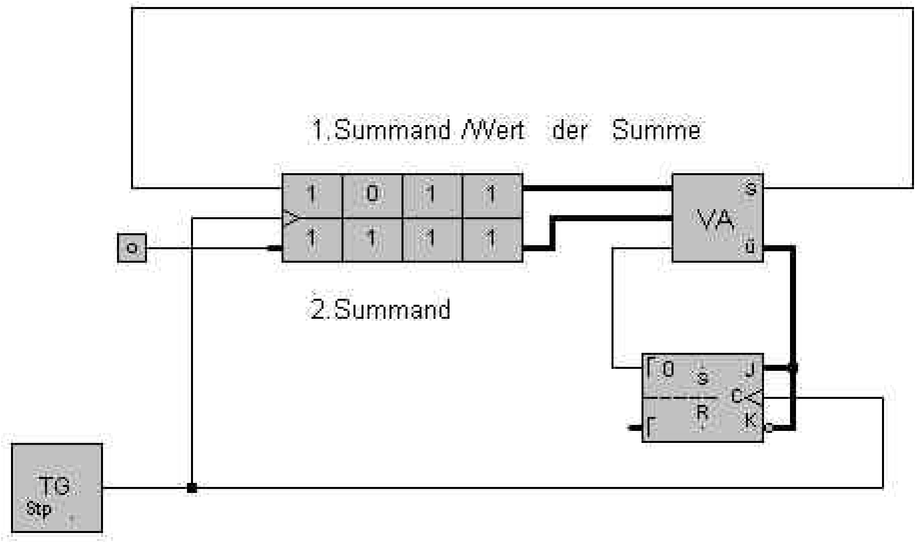


Testschaltung mit LOCAD : Rechtsschieberegister_als_Akku.LWS

	Q3	Q2	Q1	Q0
	0	1	0	1
1.Takt	1	0	1	0
2.Takt	0	1	0	1
3.Takt	1	0	1	0
4.Takt	0	1	0	1

Das Datenwort wird auch hier bei jedem Taktimpuls um eine Position nach rechts verschoben. Das am weitesten rechts stehende Bit wird dabei vorne in das Datenwort eingefügt. Die einzelnen Bits werden zyklisch vertauscht.

Aufbau eines Seriellen Addierwerks mit zwei Rechtsschiebe-registern:



	FF	A3	A2	A1	A0		B3	B2	B1	B0		S	Ü
	0	1	0	1	1		1	1	0	1		0	1
1.Takt	1	0	1	0	1		0	1	1	0		0	1
2.Takt	1	0	0	1	0		0	0	1	1		0	1
3.Takt	1	0	0	0	1		0	0	0	1		1	1
4.Takt	1	1	0	0	0		0	0	0	0		1	0

Bemerkung:
Wenn man die Registerinhalte als positive Zahlen sieht, wird die Summe 11+13=26 berechnet.

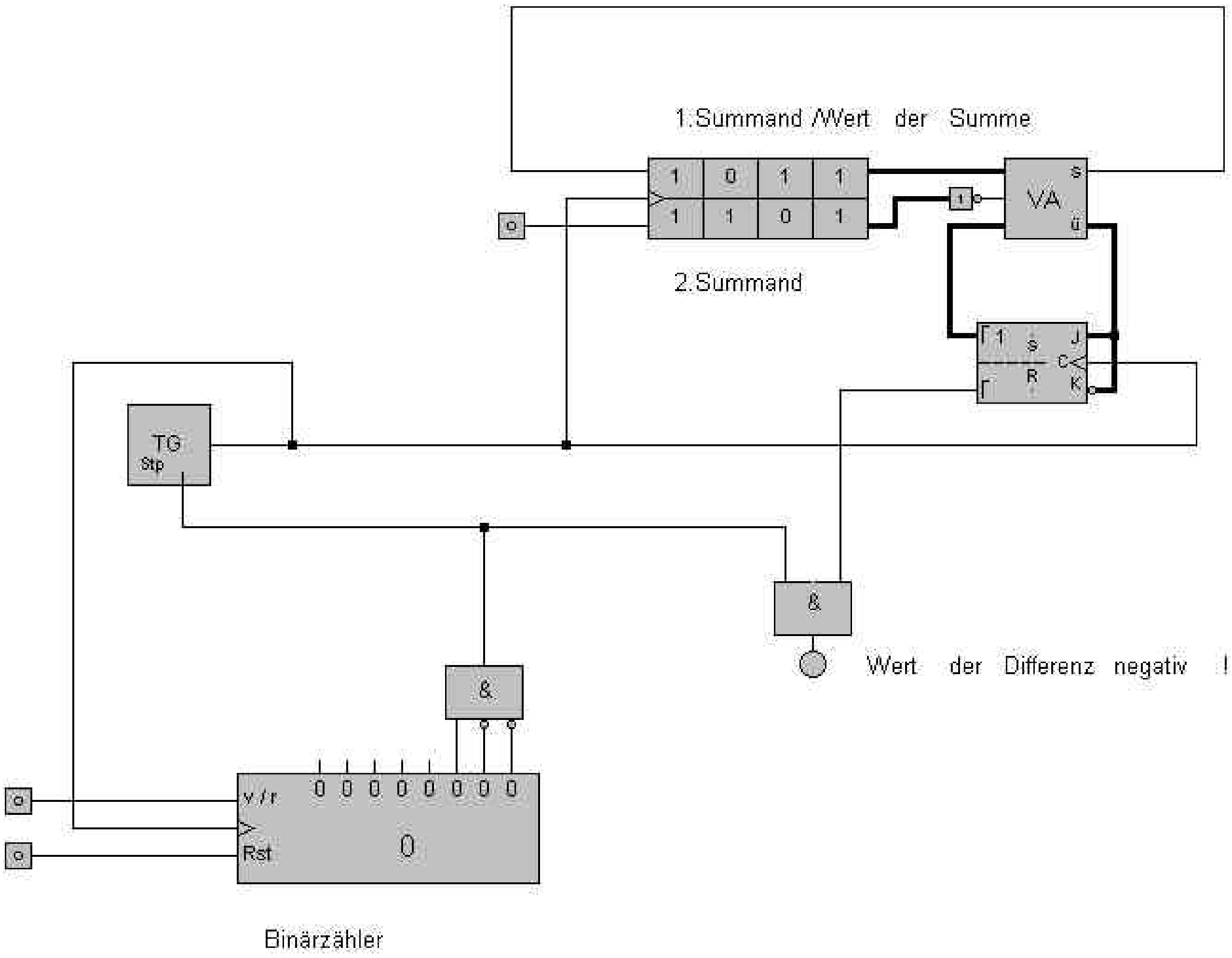
Interpretiert man die Zahlen wegen der führenden Einsen als negative Zahlen, dann wird die Summe (-5)+(-3) = (-8) berechnet.

In beiden Fällen wird das korrekte Ergebnis angezeigt :

Zahlen positiv							Zahlen negativ						
1	1	0	0	0	=	24	1	1	0	0	0	=	-8
								↓					↑
							0	0	1	1	1		
							0	1	0	0	0	=	8

Aufbau eines 4-Bit Binärsubtrahierers

Die Subtraktion erfolgt durch Addition des Zweierkomplements:



Testschaltung mit LOCAD : Subtrahierschaltung_seriell.LWS

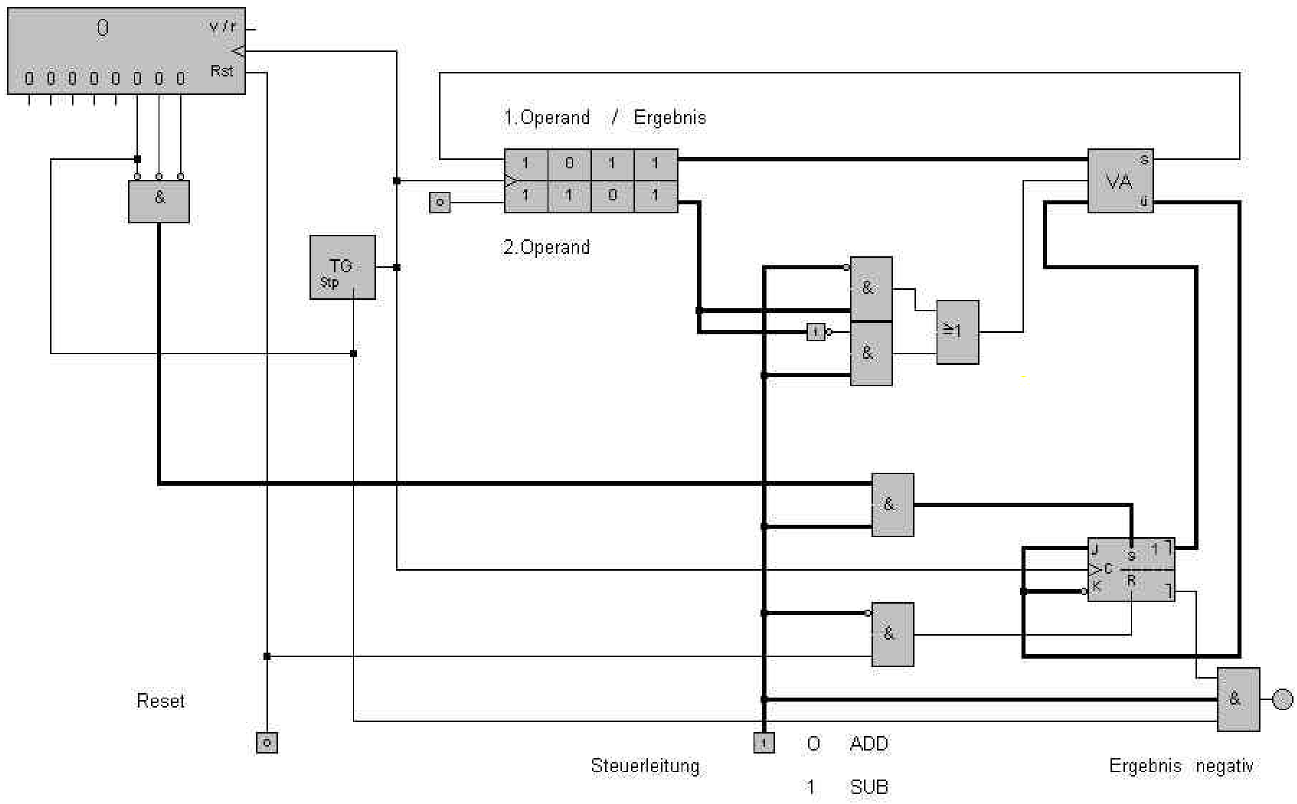
Auch hier wird in beiden Fällen wird das korrekte Ergebnis angezeigt :

Zahlen positiv 11+3 = 14								Zahlen negativ (-5) – (-3) = -2						
		1	0	1	1		11		1	0	1	1		-5
+		0	0	1	0	+	2	+	0	0	1	0	+	2
+			1	1	1	+	1			1	1	1	+	1
	0	1	1	1	0		14	0	1	1	1	0		-2

Ergebnis rückkomplementieren: $\overline{1110}+1=0001+1=0010=2$

Aufbau einer programmierbaren 4-Bit-Additions-/Subtraktionsschaltung

Programmierbar bedeutet hier, dass die Funktion der Schaltung mit Hilfe einer Steuerleitung beeinflusst werden kann.



gerechnet wird in der Abbildung wie oben : 11-13= -2

Testschaltung mit LOCAD : ProgADD_SUB.LWS

Der Trick besteht darin, dass bei der Subtraktion (Steuerleitung=1) das Einerkomplement des 2. Operanden an den VA gelegt wird und gleichzeitig vor dem ersten Taktimpuls das Übertrags-FF auf 1 gesetzt wird. Damit wird zum ersten Operanden das Zweierkomplement des 2. Operanden addiert.