Versuchsprotokoll

Wintersemester 93/94

Marcus Müller, Frank Pilhofer

Donnerstag, 20. Januar 1994

Elektronik - Praktikum / Digital - Teil

Schaltungen mit Flip-Flops

Übung 5

Inhaltsverzeichnis

Grundlagen 2

RS-Flip-Flop 2

D-Flip-Flop 3

JK-Flip-Flop 4

Asynchronzähler 5

BCD-Zähler 5

Dekodierschaltungen 6

Ringzähler 7

Versuchsdurchführung 8

RS-Flip-Flop 8

D-Flip-Flop 8

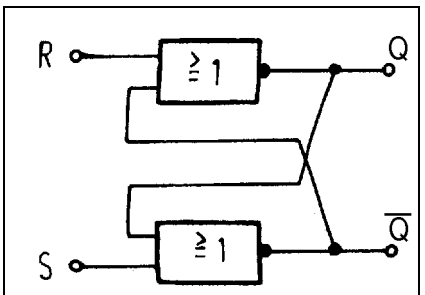
Binärzähler und BCD-Zähler 9

Dekoder 9

Ringzähler 10

Grundlagen

RS-Flip-Flop



Grafik 1 - RS-Flip-Flop

Ein RS-Flip-Flop kann entweder aus zwei NAND oder aus zwei NOR-Gattern aufgebaut werden. Betrachten wir zunächst die nebenan gezeichnete Darstellung eines NOR-Flip-Flops. Der Ausgang jedes Gatters ist auf den Eingang des jeweils anderen Gatters rückgekoppelt.

Wird eine 1 an S und 0 an R gelegt, so ist das untere NOR nicht erfüllt, an dessen Ausgang liegt eine 0. Damit liegt an beiden Eingängen des oberen NORs eine 0 an, an Q folgt damit eine 1. Ein ähnlicher Fall tritt ein, wenn die Eingänge umgepolt werden, d.h. S=0 und R=1. Mit ähnlichen Überlegungen folgt Q=0, $\overline{Q}$ =1. Gemäß ihrer Funktion werden die Eingänge mit Set und Reset bezeichnet.

Das interessante an der Schaltung erkennt man, wenn in diesem Zustand auch das Eingangssignal von R entfernt wird. Da $\overline{Q}$ noch immer auf 1 liegt, besitzt das obere NOR noch immer einen Eingang mit '1', schaltet also nicht um. Da sich an den Eingängen des unteren NORs nichts ändert, bleibt der Zustand auch dort erhalten. Wir halten fest: liegen beide Eingänge auf 0, so wird der zuvor erreichte Zustand beibehalten, d.h. gespeichert. Nach beliebiger Zeit kann an den Zuständen von Q und $\overline{Q}$ noch immer abgelesen, ob zuletzt ein Set oder Reset durchgeführt wurde.

Ein Nachteil ergibt sich jedoch aus der zeitlichen Ungewißheit. Zum Zeitpunkt des Auslesens ist nicht bekannt, wann der Flip-Flop das letzte Mal angesprochen wurde. Eine Verbesserung in dieser Hinsicht stellt das gleich zu behandelnde D-Flip-Flop dar.

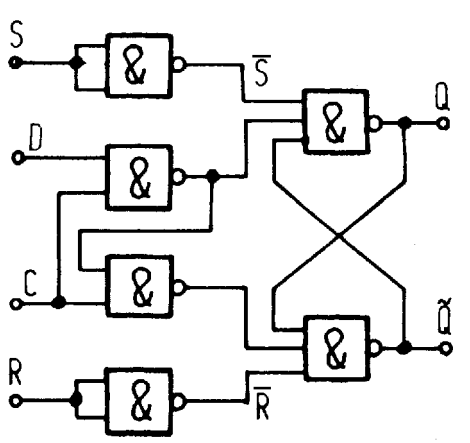
Eine kleine Überlegung am RS-Flip-Flop fehlt noch. Werden beide Eingänge der Schaltung auf '1' gelegt, so schalten logischerweise beide NORs auf '0'. Das Verhalten der Schaltung ist damit durchaus korrekt, dennoch ist dieser Zustand für den Alltagsgebrauch unbrauchbar. Sobald die Eingänge wieder auf '0' abfallen, müssen sich die Ausgänge des Flip-Flop wieder komplementär verhalten. Der Haken dabei ist, daß dieser Zustand nicht vorhersehbar ist. Sobald das Eingangssignal fällt, fangen beide NORs an, sich zu öffnen. Dasjenige mit der kleineren Schaltzeit (Differenz im Nanosekundenbereich) wird zuerst auf '1' geschaltet haben und mit der Rückkopplung das andere NOR auf '0' halten. In einigen Grenzfällen (Differenz der Schaltzeiten nahe 0, kapazitative oder induktive Kabelverbindungen) ist es durchaus vorstellbar, daß die Schaltung zunächst ein paarmal hin und hervibriert, bevor sich wieder ein eindeutiger Zustand einstellt.

Das RS-Flip-Flop kann wie bereits angedeutet auch mit NAND-Gattern aufgebaut werden. Am Verhalten der Schaltung ändert sich wenig; nur die Eingänge müssen anders angesprochen

werden. Der obige Eingang wird zu $\overline{R}$, der untere zu $\overline{S}$ umbenannt. Gespeichert wird nun für $\overline{R}=1$, $\overline{S}=1$. Auch alle anderen Funktionen sind genau umgedreht. Der Zustand $\overline{R}=0$, $\overline{S}=0$ muß vermieden werden.

Der Flip-Flop ist recht vielseitig zu verwenden, wenn er auch selten in dieser einfachsten Form benutzt wird. Weiter verbreitet sind die abgewandelten D-Flip-Flop und JK-Flip-Flop-Schaltungen. Der RS-Flip-Flop erfüllt seine Aufgabe normalerweise als Schalter. Schon durch einen kurzen Schaltimpuls an seinen Eingängen wird der Flip-Flop gekippt und merkt sich diesen Zustand bis zum nächsten Kippen.

D-Flip-Flop



Grafik 2 - D-Flip-Flop

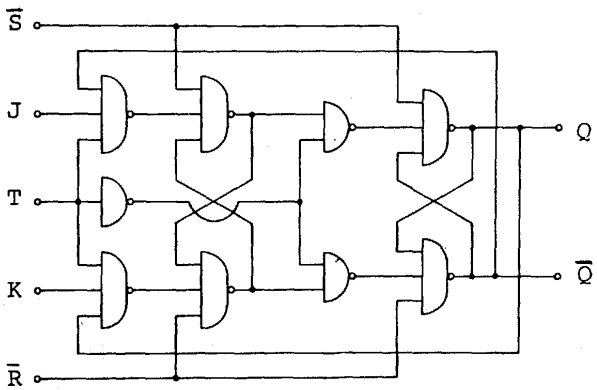
Das D in der Bezeichnung des D-Flip-Flops steht für "Data" und deutet an, daß Daten zumindest im entferntesten Sinne "verarbeitet" werden können. Rechts ist in der Grafik ein nichtinvertierender D-Flip-Flop aufgezeichnet. In der rechten Seite der Grafik erkennen wir ein normales NAND-RS-Flip-Flop wie oben. Auch der oberste und unterste Eingang sind uns nicht unbekannt. Hier können an das RS-Flip-Flop die Signale $\overline{R}$ und $\overline{S}$ gelegt werden; der Übersichtlichkeit halber werden an den Eingängen die Signale nichtinvertiert abgefragt und über ein NAND-Gatter invertiert. Mit diesen beiden Eingängen kann jederzeit der Ausgangszustand eingestellt werden. In der Mitte der Schaltung sind zwei neue Eingänge namens "D" (Data) und "C" (Clock) hinzugekommen.

Die Clock dient dazu, die Vorgänge in der Schaltung zu synchronisieren. Liegt die Clock auf '0', so liegen die Ausgänge der mittleren NANDs auf '1'; die Speicherfunktion des RS-Flip-Flop wird also nicht gestört. Wird die Clock auf '1' gesetzt, so wird der Zustand unseres Datenbits in den RS-Flip-Flop "übertragen". Ist D=0, so wird ein Reset, bei D=1 ein Set ausgeführt. Wird danach wieder C=1 gesetzt, so wird das Datenbit gespeichert und kann am Ausgang der Schaltung jederzeit abgefragt werden. Wir haben also einen 1-Bit-Mini-Speicher gebastelt.

Natürlich gibt es auch andere Möglichkeiten, Daten zu speichern. Davon seien z.B. Magnetblasenspeicher und Ferritkernspeicher genannt. Beide speichern Daten mittels der magnetischen Nord-Süd-Information und haben gegenüber dem D-Flip-Flop (ein Halbleiterspeicher) die Nachteile großer Leistungsaufnahme und langsamen Zugriffs; jedoch den Vorteil, Informationen auch ohne Stromfluß zu speichern.

Eine leichte Variation des D-Flip-Flops besteht in der Umdrehung des Data-Signals. In Grafik 2 wird das nichtinvertierte Data-Signal an das obere NAND des RS-Flip-Flops gelegt. Durch Vertauschung wird der gespeicherte Wert invertiert am Ausgang wiedergegeben ($\overline{Q}$=D).

JK-Flip-Flop



Grafik 3 - JK-Master-Slave-Flip-Flop

Der JK-Flip-Flop ist eine konsequente Weiterentwicklung aus RS- und D-Flip-Flop. Das Synchronisationsproblem, welches auch noch beim D-Flip-Flop bei Rückkopplung des Ausgangssignals bestand, wird hier beseitigt. Im Schaltbild (Grafik erkennen wir die Rechtfertigung der ausführlichen Bezeichnung, "JK-Master-Slave-Flip-Flop". Links befindet sich der Master, rechts der Slave-Flip-Flop. Zwischen beiden wird das Taktsignal invertiert. Bei ansteigender Taktflanke wird das Eingangssignal in den Master-Flip-Flop übertragen. Noch bis zum Ende dieser Taktdauer kann am Ausgang des Slaves der alte Zustand ausgelesen werden. Erst bei fallender Taktflanke wird das neue Signal vom Master in den Slave übertragen und damit an die Ausgänge gelegt.

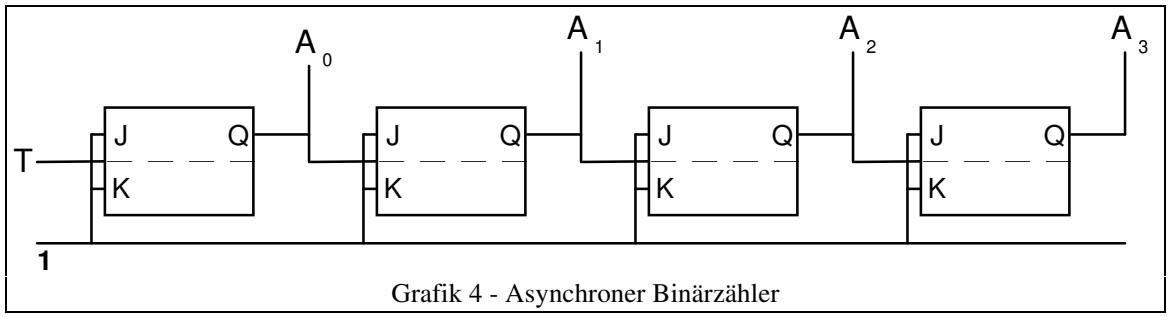
Die Schaltung soll hier nicht bis ins Detail erläutert werden; es sei deshalb nur die Wahrheitstabelle angegeben (jede Zeile entspricht einem Takt):

J	K	Q_{t-1}	Q_t	$\overline{Q}_t$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

In Kurzfassung gesagt heißt das: Ist J=K=0, so wird der alte Wert gespeichert. Bei J=1, K=0 wird eine 1 geschrieben; J=0, J=1 dementsprechend eine 0. Sind beide Eingänge J=K=1, so wird bei jedem Takt das Ausgangssignal invertiert (Frequenzteiler, Zähler). Wie immer kann die Schaltung jederzeit über die Set/Reset-Eingänge gesetzt werden.

Asynchronzähler

Basierend auf JK-Flip-Flops soll ein Zähler aufgebaut werden. Schon im letzten Absatz wurde darauf hingewiesen, daß der Flip-Flop (damit seien im folgenden immer JK-Flip-Flops gemeint) zum Zählen taugt, wenn er als Frequenzteiler betrieben wird. Wir verketten 4 Flip-Flops auf einfache Weise zu einem Zähler modulo 16, indem wir jedesmal den Ausgang *Q* eines Flip-Flop auf den Takteingang des nächsten legen. Es ergibt sich die folgende Schaltung (die Flip-Flops sind zur Übersicht durch ein kleines Kästchen ersetzt worden.



Grafik 4 - Asynchroner Binärzähler

Der erste Flip-Flop und damit der erste Ausgang A_0 kippt mit jeder fallenden Flanke des Taktes (denn erst bei fallender Flanke wird der Wert aus dem Master in den Slave übertragen). Bei der nächsten abfallenden Flanke wird A_0=0, was einer fallenden Flanke am Takteingang des nächsten Flip-Flops entspricht, weshalb A_1 kippt uswusf. Ein Oszillogramm des Verhaltens ist in der Versuchsdurchführung aufgezeichnet.

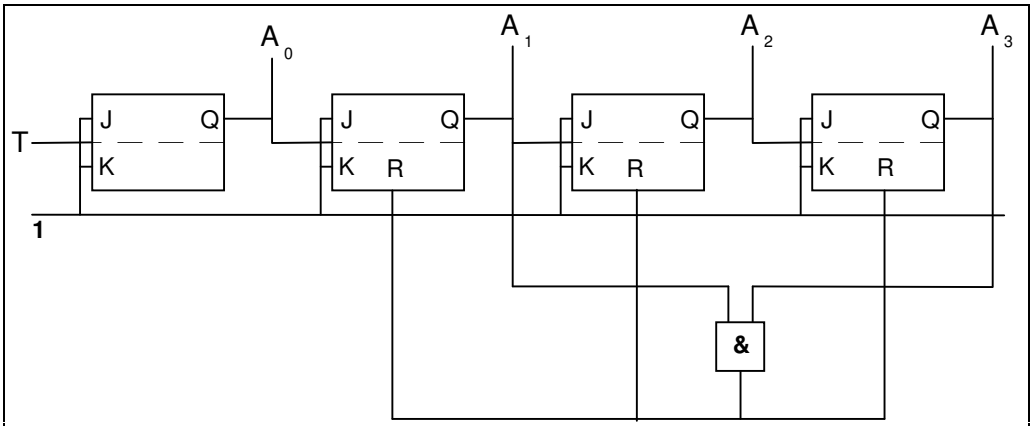
Weshalb der Zähler das Attribut 'asynchron' trägt, merkt man erst bei genauerem Hinsehen, laufen doch alle Vorgänge für das menschliche Auge synchron. In der Praxis sind Signalflanken nie senkrecht (das verbietet schon die Mathematik, die das Leben des Physikers immerfort bestimmt). Benötigt z.B. ein Flip-Flop 0,1µs, um einen eindeutigen Pegel zu erreichen, so "erfährt" das folgende Flip-Flop ebenfalls erst nach dieser Verzögerung, ob es zu schalten hat oder nicht. Der zweite Baustein wäre gegenüber dem ursprünglichen Takt bereits 0,2µs verzögert. Bei großen Schaltungen und kurzen Taktzyklen kann sich das zu einem echten Nachteil ausbauen, wenn der Ausgang dem Eingang um (im Extremfall) mehrere Takte hinterherhinkt.

Diese Schwierigkeit wird bereinigt, indem mit etwas erhöhtem Schaltungsaufwand ein Synchronzähler aufgebaut wird. Bei diesem wird das gleiche Taktsignal auf alle Flip-Flops gegeben. Über die Eingänge J,K wird dann so umgeschaltet, wie es zum Zählen nötig ist.

BCD-Zähler

Mit dem eben aufgebauten 4-Bit-Zähler konnten an den Ausgängen A_0, A_1, A_2, A_3 Werte zwischen 0 und 15 abgegriffen werden. Soviel Bedeutung die Rechnung modulo 16 in der Computertechnik und Informatik hat, sowenig einleuchtend ist sie ohne passenden Taschenrechner. Noch immer sind wir die Dezimalrechnung gewohnt.

Mit Hilfe von ein paar Gattern kann der Modulo-16-Zähler so umgebaut werden, daß er nur noch bis 10 zählt. Die einfachste und hier verwendete Möglichkeit besteht darin, den Wert 10 an den Ausgängen abzutesten ($\overline{A_0} A_1 \overline{A_2} A_3$). Ist diese Bedingung erfüllt, so wird einfach ein Reset an die gesamte Schaltung geschickt:



Grafik 5 - BCD-Zähler

Die Logik wurde noch ein wenig vereinfacht. Als Eingänge in das AND brauchen nur die Werte A_1, A_3 verwendet werden. Beide zusammen stellen bereits die Zahl 10 dar. Ein Wert, bei dem außer diesen beiden auch noch A_0 oder A_2 zuträfe, darf also laut Voraussetzungen gar nicht vorkommen. Auch braucht der erste Flip-Flop nicht zurückgesetzt werden. Zum Zeitpunkt des Resets, wenn der Ausgang 10 zeigt, steht A_0 sowieso auf 0.

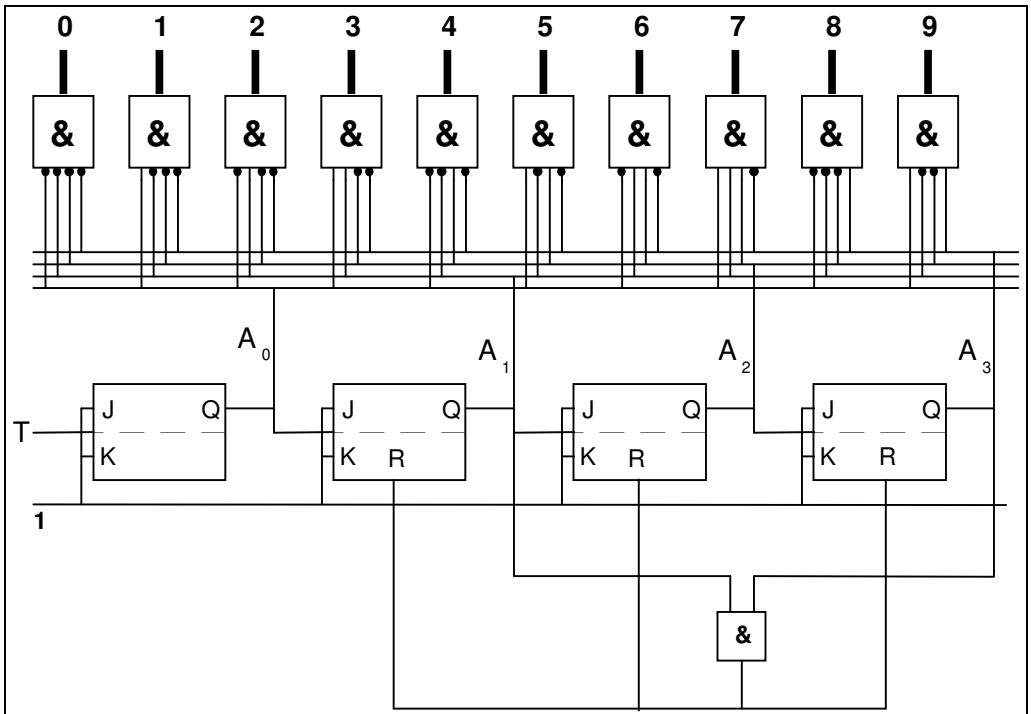
Dasselbe könnte man im Prinzip auch vom dritten Flip-Flop von links anmerken, doch hier offenbart sich ein Nachteil der Schaltung. Bis die 10 erkannt wird, muß der zweite Flip-Flop kurzzeitig auf '1' schalten; erst kurze Zeit später folgt der Reset. Würde der dritte FF kein Reset erhalten, so fände er an seinem Takteingang kurzzeitig eine '1'. Beim Reset des zweiten bekäme er eine negative Taktflanke und würde kippen, was wir natürlich nicht wollen.

Was hier in einem Nebensatz erwähnt wurde, ist natürlich nicht so trivial wie es scheint. Wir wollten einen BCD-Zähler bauen, und dieser darf laut Definition keine 10 an seinem Ausgang zeigen, egal wie kurz dieser Zustand dauert! Eine Verbesserung wäre durch umständliche Umbauten zu erreichen und laufen automatisch in Richtung Synchronzähler (da die Steuerung über die JK-Eingänge zu erfolgen hat).

Dekodierschaltungen

Mittlerweile haben wir dem Zähler beigebracht, anstatt bis 15 nur bis 9 zu zählen. Das hört sich schon besser an, doch immer noch werden an den Ausgängen bloß langweilige Bitkombinationen ausgegeben. Um den Komfort zu steigern, sollen die Ausgänge noch durch eine 1-aus-10-Dekodierung entschlüsselt werden. Das heißt im Klartext, daß wir 10 Leuchtdioden anschließen, von denen die 0. brennen soll, wenn der Zähler 0 zeigt etc. Wir schreiben uns

dafür die Bedingungen hin, die für die einzelnen Ziffern gelten sollen, z.B. $\overline{A_0}\,\overline{A_1}\,\overline{A_2}\,\overline{A_3} \mathrel{\hat{=}} 0$, $A_0\overline{A_1}A_2\overline{A_3} \mathrel{\hat{=}} 5$ etc. Insgesamt würde sich für die Schaltung folgendes Wirrwarr ergeben:

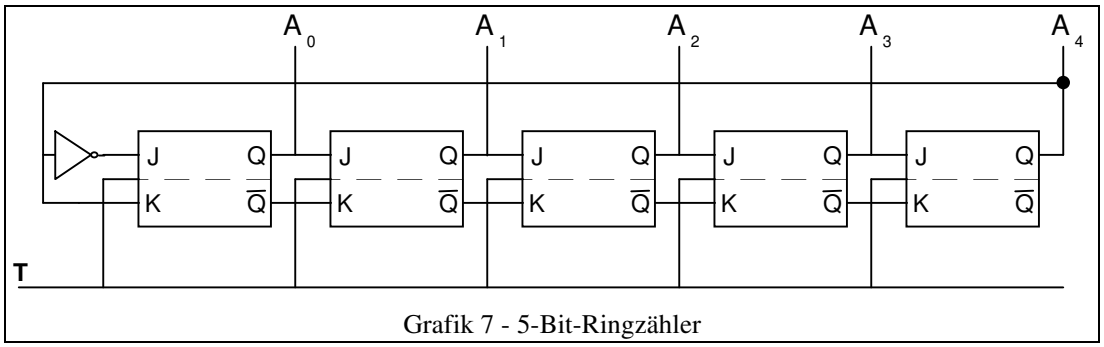


Grafik 6 - BCD-Zähler mit 1-aus-10-Dekoder

Die Schaltung sieht nicht nur teuflisch unübersichtlich aus, sie ist es auch, vor allem, wenn die gesamte obere Schaltungsmatrix nur mit NAND-Bausteinen geschaltet werden soll.

Ringzähler

Ein Ringzähler, eine Abwandlung des Schieberegisters, ähnelt Schaltungsmäßig nur auf dem ersten Blick dem Binärzähler. Der Aufbau aus Flip-Flops ist so ziemlich die einzige Gemeinsamkeit zwischen beiden.



Grafik 7 - 5-Bit-Ringzähler

Die Ausgänge eines jeden Flip-Flop wird als Eingang an den nächsten Flip-Flop weitergegeben. Nur der Ausgang des letzten Flip-Flops wird zunächst invertiert. Bei jedem Takt übernimmt ein Flip-Flop den Zustand seines Vorgängers; der erste wird mit dem invertierten Zustand des letzten gespeist. Nicht eingezeichnet sind die Reseteingänge, mit denen sich die Schaltung zurücksetzen läßt.

Versuchsdurchführung

RS-Flip-Flop

Ein RS-Flip-Flop wurde nach Grafik 1 aufgebaut; die Wahrheitstabelle wurde gemessen.

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

Das Ergebnis entspricht den Erwartungen. Ein einzelnes Reset oder Set schaltet den Flip-Flop. Der so eingestellte Wert wird solange gespeichert, wie beide Eingänge auf 0 liegen. Werden beide Eingänge auf 1 geschaltet, so stellt sich zwar ein logisch definierter Ausgangszustand ein, doch ist der Folgezustand nicht definierbar.

D-Flip-Flop

Der D-Flip-Flop wurde wie in Grafik 2 aufgebaut; die Wahrheitstabelle wurde gemessen.

R	S	C	D	Q	$\overline{Q}$
1	0	?	?	0	1
0	1	?	?	1	0
0	0	1	1	1	0
0	0	1	0	0	1
0	0	0	1	0	1
0	0	0	0	0	1
1	1	?	?	1	1
1	0	1	1	1	1
0	1	1	0	1	1

Auch bei dieser Schaltung werden die Erwartungen voll erfüllt. Ein Set oder Reset kann über die gleichnamigen Leitungen eingespeist werden. Liegt der Taktimpuls Low, so speichert das Flip-Flop; der Ausgang ändert sich nicht. Erst bei C=1 wird das Datum invertiert an den Ausgang übertragen.

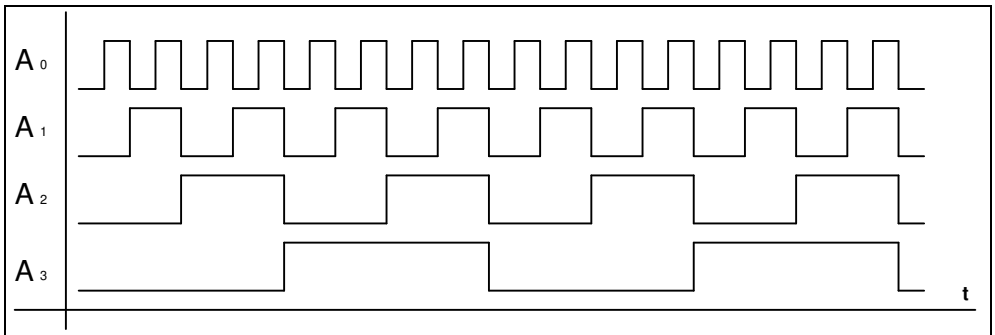
In den drei letzten Zeilen sind die gefundenen verbotenen Zustände eingetragen. Es ist natürlich sinnlos, an der Schaltung gleichzeitig ein Set und ein Reset durchführen zu wollen; die Schaltkreise reagieren mit logischem Schrott. Zwar ist der Ausgang logisch definiert, doch ist der Folgezustand wie oben undefiniert.

In den beiden anderen verbotenen Fällen soll gleichzeitig ein Datenbit abgespeichert und eine "komplementäre" Operation durchgeführt werden. Man sieht sofort, daß die Einspeisung eines Datums einen ganz normalen Reset (D=0) oder Set (D=1) durchführt. Die Operation

S=1, D=1 u.U. erzeugen zwar keine logisch verbotenen Zustände, doch ein solcher Zustand sollte auch vermieden werden.

Binärzähler und BCD-Zähler

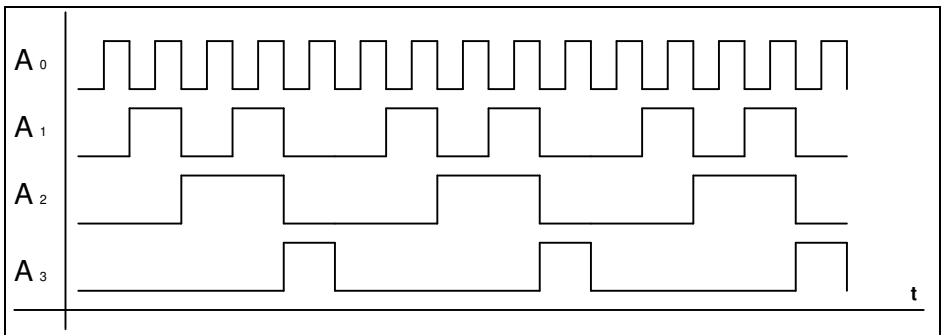
Die in Grafik 4 und 5 aufgemalten Schaltungen wurden aufgebaut. Die Ausgangswerte entsprachen den Wünschen, dennoch wurden sie notiert. Die Ausgänge sollen hier als Oszillographenbilder dargestellt werden. Hier zunächst das Bild des Binärzählers.



Grafik 8 - Oszillogramm des Binärzählers

Die Grafik beginnt links beim Zahlenwert 0. Bis zum rechten Rand wird zweimal bis 15 durchgezählt.

Zum Vergleich das Bild des BCD-Zählers gleich hinterher:



Grafik 9 - Oszillogramm des BCD-Zählers

Der BCD-Zähler zählt bis 9, dann wird er über den Reset auf 0 zurückgesetzt. Im Prinzip müßte - bei genügend genauem Hinsehen - während dieses Umsprungs ein sehr kurzer Ausschlag an A_1 zu sehen sein. Schließlich erfolgt der Reset erst dann, wenn dieses Bit auf 1 gesprungen ist.

Dekoder

Da wir zu faul waren, die Schaltung aus Grafik 6 aufzubauen (es wären sicher auch nicht genügend Kabel zur Verfügung gewesen), begnügten wir uns damit, die Zustände der 3 und der 7 zu dekodieren und über eine Leuchtdiode anzuzeigen. Erschwert wurde die Dekodierarbeit, da nur NAND-Gatter verwendet werden sollten. Als kleine Hilfe benutzten wir neben den Standard-NANDs auch vierfach-NANDs. Mit Hilfe dieser Bauteile konnte die Logik aufgestellt werden.

Es liegt genau dann eine 3 am Ausgang, wenn $A_0 A_1 \overline{A_2}\,\overline{A_3}$ gilt. Dieses wird nun in NANDs umformuliert:

$$A_0 A_1 \overline{A_2}\,\overline{A_3} = \neg NAND\left(A_0, A_1, \overline{A_2}, \overline{A_3}\right) = \neg NAND\left(A_0, A_1, NAND\,(A_2, A_2), NAND\,(A_3, A_3)\right)$$

Analog gilt für die 7:

$$A_0 A_1 A_2 \overline{A_3} = \neg NAND\left(A_0, A_1, A_2, \overline{A_3}\right) = \neg NAND\left(A_0, A_1, A_2, NAND\,(A_3, A_3)\right)$$

Dabei habe ich die Aufschlüsselung des letzten Inverters unterschlagen, was aber klar sein sollte. Nachdem die Schaltkreise aufgebaut, die Signale abgezapft und die Ausgänge auf Leuchtdioden gelegt waren, konnte das Ergebnis bewundert werden: Die Schaltung funktionierte.

Bezüglich der Schaltlogik lassen sich jedoch noch einige Verbesserungen treffen. Die invertiert benötigten Ausgänge brauchen keinen eigenen Invertierer, vielmehr kann direkt das $\overline{Q_n}$ des passenden Ausgangs beschaltet werden. Auch kann ein vierfach-NAND-Eingang komplett eingespart werden. In den Fällen $A_0 A_1 \overline{A_2}\,\overline{A_3}$ und $A_0 A_1 A_2 \overline{A_3}$ kann auf die Abfrage von $\overline{A_3}$ verzichtet werden, schließlich beschalten wir einen BCD-Zähler. Wäre A_3=1, so hätten wir die Ausgänge 11 bzw. 15, was nicht der Fall sein kann.

Ringzähler

Der Ringzähler wurde nach Grafik 7 aufgebaut. Die Wahrheitstabelle ist wie folgt:

t	A_0	A_1	A_2	A_3	A_4
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1
10	0	0	0	0	0

Wir erkennen, daß dieser 5-Bit-Ringzähler ebenfalls als BCD-Zaehler tauglich ist, da er 10 unterscheidbare Zustände besitzt, die von 0 bis 9 numeriert werden können. Zunächst scheint der Ringzähler gegenüber dem Binärzähler den Nachteil zu haben, daß er ein Flip-Flop mehr benötigt. Beim genaueren Hinsehen werden jedoch viele Vorteile sichtbar.

So arbeitet der Ringzähler von Natur aus synchron, wir bekommen keine Timingprobleme und müssen die Schaltung nicht bei Erreichen der 10 mühsam Resetten. Der zweite deutliche Vorteil ergibt sich bei der zu erreichenden 1-aus-10-Dekodierung. Auch hier scheint es auf den

ersten Blick, daß es viel aufwendiger sein muß, 5 Ausgänge miteinander zu verknüpfen. Dank der Eigenschaften des Ringzählers (in jedem Schritt wird nur ein Bit geändert, und das durchgängig von links nach rechts) ist dieser Aufwand jedoch gar nicht nötig! Wie man leicht durch Überlegen nachprüft, ergeben sich folgende Produktterme zur Dekodierung:

Dekodiert	Verknüpfung	Dekodiert	Verknüpfung
0	$\overline{A_0}\,\overline{A_4}$	5	$A_0 A_4$
1	$A_0\,\overline{A_1}$	6	$A_1\,\overline{A_0}$
2	$A_1\,\overline{A_2}$	7	$A_2\,\overline{A_1}$
3	$A_2\,\overline{A_3}$	8	$A_3\,\overline{A_2}$
4	$A_3\,\overline{A_4}$	9	$A_4\,\overline{A_3}$

Das sind entscheidend kleinerer Aufwand als beim Binärzähler.