BASISWISSEN FLIPPER-ELEKTRONIK

Ein Service von www.flippermarkt.de

Kapitel 2

Version 1.0 – 17.1.2007
Version 1.01 – 25.1.2007
Version 1.02 – 25.5.2007

Schaltungsalgebra und deren Anwendung in digitalen Systemen

Um Stromlaufpläne, bei Flippern als Schematics bezeichnet, soweit verstehen zu können, damit durch Messungen Fehler in der Elektronik aufgespürt werden können, sind einige Grundkenntnisse dieser Algebra erforderlich.
Sie ist bekannt als **Boolsche Algebra**, benannt nach George Bool, dessen Arbeiten zusammen mit denen von Leibnitz die theoretische Grundlage der heutigen Digitaltechnik sind.
Dieser Algebra liegt eine **binäre Logik** zu Grunde, die sich durch erfrischende Einfachheit und geradezu ästhetische Klarheit auszeichnet.
Mit der „Logik“, wie sie in der Umgangssprache verwendet wird und die in engem Zusammenhang steht mit dem, was man als den gesunden Menschenverstand bezeichnet, gibt es allenfalls marginale Ähnlichkeiten.
Es gibt nur ja oder nein, keinen Kompromiss und keinen Konsens.

Eine Aussage ist entweder **wahr** oder **falsch**, dazwischen gibt es nichts**.**
Das sind die beiden **Boolschen Konstanten**.
Mathematisch wird **wahr** die Ziffer **1** zugeordnet, **falsch** die Ziffer **0**.

Emotionen gibt es nicht, wahr ist nicht besser als falsch. Falsch ist nicht gelogen.
Die **Konstanten** sind **gleichwertig**.
„Aussage“ kann für vieles stehen, für eine Bedingung, einen Zustand, ein Ereignis etc.
Es ist immer eine **Bool’sche Variable, die stets wahr (1) oder falsch (0) ist**.

Logische Verknüpfungen von wahren und falschen Aussagen führen zu neuen Aussagen, die wiederum wahr oder falsch sein können.
Solche Verknüpfungen erfolgen durch die **Boolschen Operatoren**:
AND (und), **OR** (oder), **NOT** (nicht).
Für **AND** verwenden wir das Symbol *****, diese Verknüpfung heißt **konjunktiv,**
für **OR** das Symbol **+,** diese Verknüpfung heißt **disjunktiv**
und für **NOT** das Symbol **/**.

Es sind auch andere Symbole im Umlauf, von denen uns aber nur NOT zu interessieren braucht:
Hinter dem Namen der Variablen steht das Symbol \, oder über dem Namen der Variablen befindet sich in ganzer Länge ein Strich. Letztere Notation wird in den Schematics der Flipper angewendet.

Beispiel: $\overline{\text{BLANK}}$, /BLANK und BLANK\ habe die gleiche Bedeutung.

Betrachten wir einen Schalter und beschreiben seine zwei möglichen Zustände.
Da würde man sagen, er ist entweder geschlossen oder offen.
Damit kommen wir in der Binärlogik leider nicht weiter.
Wir müssen uns schon zwischen zwei Aussagen entscheiden:
Geschlossen oder nicht geschlossen, korrekt: geschlossen - /geschlossen
Offen oder nicht offen, korrekt: offen - /offen

Wir sagen:
Ein Zimmer ist hell, wenn eine Lampe eingeschaltet ist und die Sicherung in Ordnung ist.
Die Variablen definieren wir: hell, Lampe, Sicherung.
Wir stellen eine AND-Verknüpfung her, und die Boolsche Gleichung lautet dann:
Lampe * Sicherung = hell

Wir sagen:
Ein Zimmer ist hell, wenn eine Lampe eingeschaltet ist oder Tageslicht herrscht.
Die Variablen definieren wir: hell, Lampe, Tageslicht.
Wir stellen eine OR-Verknüpfung her, und die Boolsche Gleichung lautet dann:
Lampe + Tageslicht = hell

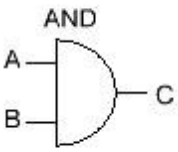
Wir sagen:
Ein Zimmer ist hell, wenn eine Lampe eingeschaltet ist und die Sicherung in Ordnung ist oder Tageslicht herrscht.
Die Variablen definieren wir: hell, Lampe, Sicherung, Tageslicht.
Wir verbinden eine OR-Verknüpfung mit einer AND-Verknüpfung, und die Boolsche Gleichung lautet dann:
Lampe * Sicherung + Tageslicht = hell
Wann ist das Zimmer dann nicht hell?

(/Lampe + /Sicherung) * /Tageslicht = /hell
Wir lernen daraus:
Die Variablen selbst und ihre Verknüpfungen müssen invertiert werden, um das Ergebnis zu invertieren.
Warum die Klammern um die OR-Verknüpfung?
Weil auch hier Punktrechnung vor Strichrechnung geht.
Hätten wir fälschlich geschrieben
/Lampe + /Sicherung * /Tageslicht = /hell
würde das übersetzt bedeuten:
Es ist /hell (dunkel), wenn die Lampe aus ist oder die Sicherung defekt ist und zudem Nacht ist.
Es wäre also auch bei Tag dunkel, wenn die Sicherung defekt ist.

Für die Boolschen Operatoren werden in Stromlaufplänen **Symbole** verwendet, in den Schematics der Flipper die amerikanischen Versionen, die deshalb hier verwendet werden.

Die auf der linken Seite des jeweiligen Symbols stehenden Variablen führen nach der Verknüpfung zu dem rechts stehenden Ergebnis.

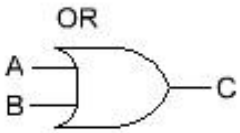
AND (UND)**: C = A * B**



Wahrheitstabelle

A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

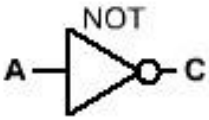
OR (ODER): **C = A + B**



Wahrheitstabelle

A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

NOT (NICHT) : C = /A



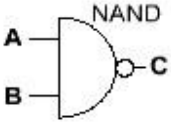
Wahrheitstabelle

A	C
0	1
1	0

Weitere Symbole sind aus Boolschen Operatoren zusammengesetzt:

AND und NOT führen zu NAND.

NAND: A * B = /C

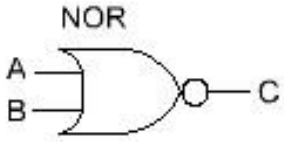


Wahrheitstabelle

A	B	C
0	0	1
1	0	1
0	1	1
1	1	0

OR und NOT führen zu NOR

NOR: A + B = /C



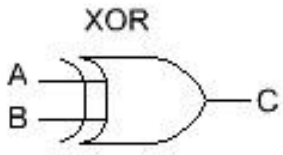
Wahrheitstabelle

A	B	C
0	0	1
1	0	0
0	1	0
1	1	0

Diese Symbole werden in der Digitaltechnik **Gatter** genannt
Die Gatter werden durch Integrierte Schaltungen implementiert, wobei ein Baustein meist mehrere Gatter enthält.

Zu diesen Gattern gehört auch das **Exklusive Oder.**

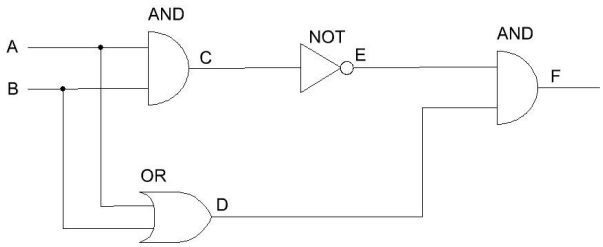
XOR: A * /B + /A * B = C



Wahrheitstabelle

A	B	C
0	0	0
1	0	1
0	1	1
1	1	0

Mit den reinen Boolschen Operatoren kann man das Exklusive Oder so darstellen:

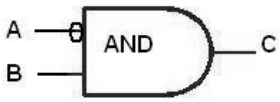


Wahrheitstabelle

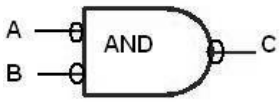
A	B	C (A*B)	D (A+B)	E (/C)	F (D*E)
0	0	0	0	1	0
1	0	0	1	1	1
0	1	0	1	1	1
1	1	1	1	0	0

Ein in einer Verknüpfung enthaltenes NOT wir durch **o** an der Variablen gekennzeichnet, auf die das NOT zutrifft, wie wir bei NAND und NOR gesehen haben.

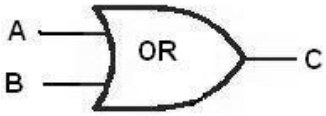
/A * B = C wird dann so dargestellt:



Und /A * /B = /C so:



Letztere Darstellung wird man jedoch selten finden, denn durch Invertierung der Operatoren und der Variablen kann jede konjunktive Verknüpfung in eine disjunktive umgewandelt werden und umgekehrt. /A * /B/ = /C entspricht also A + B = C, und deshalb würde man statt der letzten Darstellung der Klarheit wegen lieber diese wählen:



Das Verständnis der Äquivalenz von AND sowie OR ist wichtig, weil es bei der Implementierung einer Schaltung oft darauf ankommt, welche Bauteil-Funktionen sozusagen übrig sind.
Sind beispielsweise einige ANDs und NOTs frei, wird man versuchen, ein erforderliches OR durch diese zu realisieren, statt einen neuen Baustein einzuplanen.

Alle Boolsche Verknüpfungen lassen sich durch die Operatoren AND und NOT bilden, man spricht dann von der **konjunktiven Normalform**.
Ebenso lassen sich alle Boolsche Verknüpfungen durch die Operatoren OR und NOT bilden, man spricht dann von der **disjunktiven Normalform.**

Ein AND/NAND-Symbol in einem Stromlaufplan bedeutet nicht unbedingt, dass eine konjunktive Verknüpfung beabsichtigt war, und für ein OR/NOR-Symbol gilt dies vice versa.

Wir kommen jetzt zurück auf **1** für **wahr** und **0** für **falsch**.

In der Digitaltechnik bezeichnen wir die Boolschen Variablen als **Signale**, die entweder wahr oder falsch sind.
Diesen beiden Zuständen müssen wir physikalische Größen zuordnen, um sie eindeutig unterscheiden zu können.

Diese Zuordnung erfolgt durch den Wert einer Gleichspannung.
Allgemein spricht man von **positiver Logik**, wenn der Spannungswert von 1 größer ist als der von 0.
Und von **negativer Logik**, wenn der Spannungswert von 1 kleiner ist als der von 0.

Bei den Schaltungen in Flippern wird mit **positiver Logik gearbeitet.**

Fast alle sogenannten Logik-Bausteine werden mit einer Spannung von +5 V betrieben und bilden an ihrem Ausgang einen Schalter nach, der nur 2 Zustände kennt: Geschlossen und nicht geschlossen.
Ein Signal hat den Zustand 1, wenn seine Spannung einen Wert von +5 V (zwischen +3,5 V und +5 V) hat.
Ein Signal hat den Zustand 0, wenn seine Spannung einen Wert von 0 V (zwischen 0 V und +0,5 V) hat.

Der **Zustand** eines Signal wird als dessen **Potential** bezeichnet.

Die Idealwerte von 0V und +5 V gibt es, bedingt durch die Innereien der elektronischen Bauelemente, nur in Ausnahmefällen. Deshalb gelten die in Klammern gesetzten Bereiche als zulässig.
Spannungswerte zwischen diesen oder gar außerhalb dieser Bereiche sind unzulässig und deuten stets auf ein defektes Bauelement oder einen Fehler in der Schaltung hin.

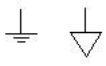
Wegen der Spannungszuordnung wird wegen der positiven Logik bei physikalischer Betrachtung eines Potentials **1** mit **high (H)**, **0** mit **low (L)** beschrieben.

Eine logische 1 ist physikalisch H.
Eine logische 0 ist physikalisch L.

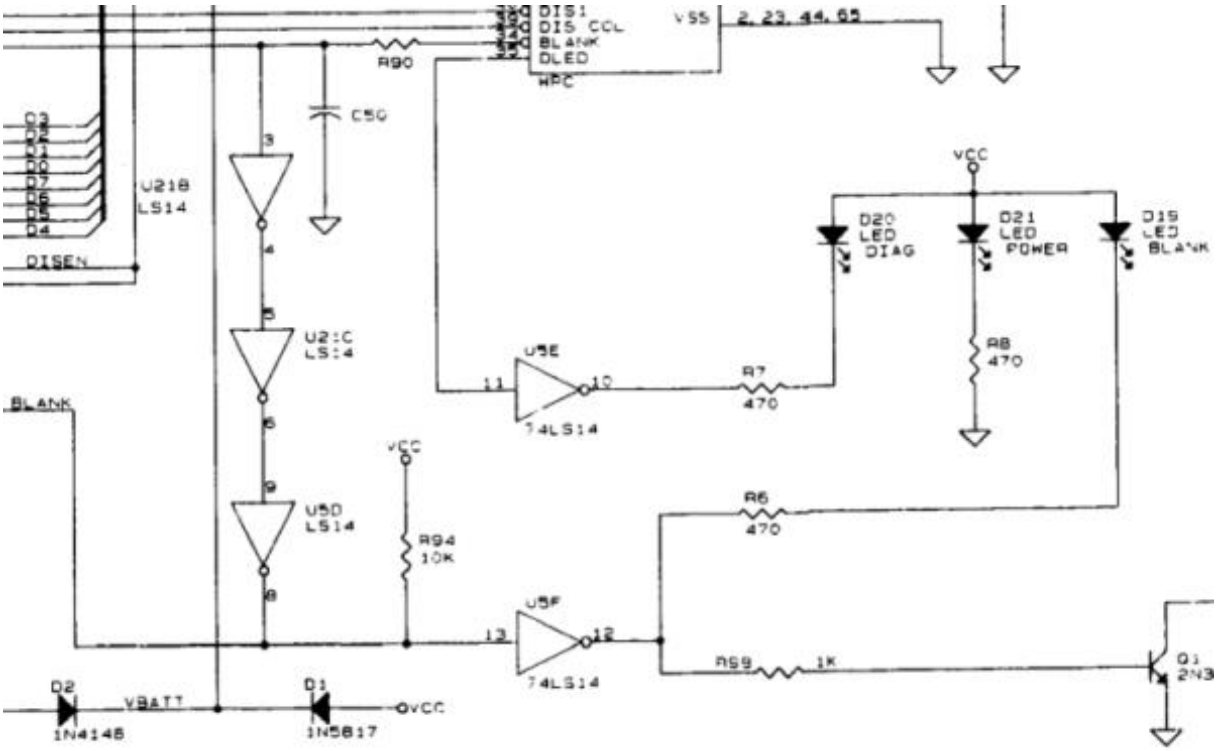
Folgendes ist offensichtlich:

+5 V, als Signal bezeichnet mit **VCC** oder dargestellt durch dieses Symbol, ist unter dieser Voraussetzung stets die Konstante **1** mit dem Potential **H.**

0 V, als Signal bezeichnet mit **GND** (Masse) oder dargestellt durch eines der nebenstehenden Symbole**,** ist unter dieser Voraussetzung stets die Konstante **0** mit dem Potential **L**.



Wir nehmen uns jetzt ein Signal vor, das in Flippern eine zentrale Wirkung hat: BLANK.
Es soll bewirken, dass während eines kurzen Zeitraums nach Einschalten des Geräts alle Spulen, Lampen sowie Sound und Display gesperrt sind.
Es wird auf dem CPU-Board durch das WPC-IC erzeugt.
Wir sehen, dass es dort aus Pin 34 austritt, und zwar in invertierter Form, wie wir an dem kleinen Kreis erkennen. Für BLANK = 1 steht dieser Ausgang also auf 0.



Im weitern Verlauf des BLANK-Signals sehen wir nur **Inverter**, mit einem Inverter wird das **NOT** realisiert.
Hier werden ICs (U21 und U5) des Typs 74LS14 verwendet, bei denen 6 Inverter in einem Gehäuse untergebracht sind.
Widerstände und Kondensatoren beachten wir nicht, sie dienen nur der Stabilisierung der Signale und der Strombegrenzung.

	WPC.34	U21.4	U21.6	U5.8	U5.12
BLANK	0	1	0	1	0

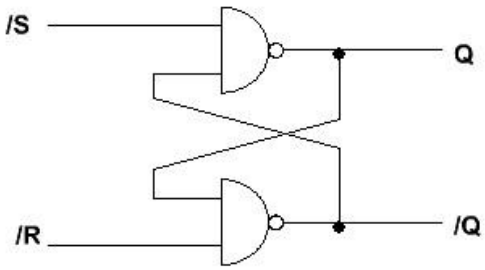
So wechselt das Potential des Signals auf der hier betrachteten Strecke.
Von U5.8 wird es im gesamten System verteilt. /BLANK an U5.12 bewirkt, dass Strom durch das LED D19 fließt.
In dem senkrechten Signallauf sehen wir 3 Inverter, von denen 2 überflüssig sind, was die Logik angeht. Das Signal wird jedoch durch jeden Inverter ein wenig verzögert, da dieser eine gewisse Reaktionszeit braucht. Ob diese Verzögerung tatsächlich erforderlich ist, mag dahinstehen.

Rechts unten sehen wir das Symbol für einen NPN-Transistor. Auch dieser stellt einen Inverter dar.
Die Ausgänge der Gatter lassen nur einen geringen Strom zu, je nach Typ maximal 16 mA.
Reicht das für das nachgeschaltete Bauelement nicht aus, nimmt man einen Transistor als Stromverstärker.

In allen halbwegs komplexen Systemen ist es erforderlich, Signale zu „verriegeln“
Der Zustand eines Signals soll zum Zeitpunkt t festgehalten und zu einem späteren Zeitpunkt t+n ausgewertet werden. Welches Potential das Signal zum Zeitpunkt t+n hat, interessiert dann nicht mehr.
Diese Verriegelung wird durch ein **Flipflop**, auch bistabile Kippstufe genannt, erreicht.
Ein Flipflop kann durch geeignete Verknüpfung der Boolschen Operatoren gebildet werden.

Flipflops gibt es vielen Varianten.

Für die einfachste Ausprägung, ein **R-S Flipflop**,
genügen 2 NAND-Gatter. R steht für Reset, S für Set.

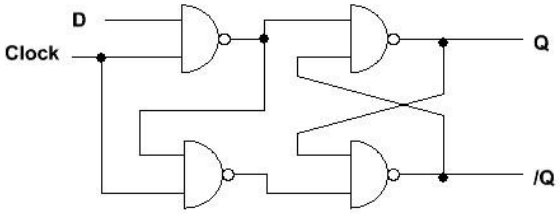


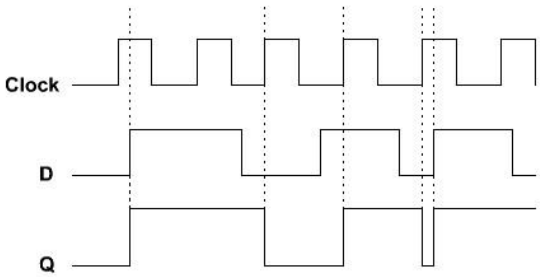
Wahrheitstabelle

S	R	Q	/Q	Aktion
1	0	1	0	Q setzen
0	1	0	1	Q löschen
1	1	1	1	Unzulässig
0	0	Q	/Q	Keine Änderung

Es ist ein rastender Schalter, der nach Einschalten
des Stroms in einer zufälligen Position ist. Mit S wird er eingeschaltet, mit R ausgeschaltet.
An den Eingängen des FlipFlop liegen die jeweils invertierten Signale an.
Die Unzulänglichkeit dieses einfachen Flipflop ist offensichtlich. Es ist der unzulässige Zustand möglich, dass beide Ausgänge auf 1 stehen.
Die beiden Ausgänge eines Flipflop müssen stets gegensätzliches Potential führen.
Es muss also verhindert werden, dass S und R gleichzeitig Potential 1 führen.

Durch Hinzufügen von 2 Gattern kann man dem abhelfen. Man erhält so ein **D-Flipflop**.
Diese Variante ist bei Flippern die häufigste.



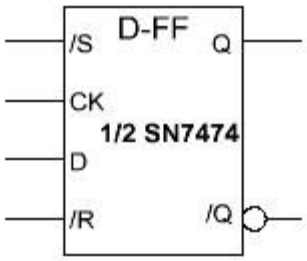


Das am D(aten)-Eingang anstehende Potential wird auf den Ausgang Q durchgeschaltet, wenn Clock auf 1 steht und dort verriegelt, wenn Clock auf 0 geht.
Man spricht bei dieser Ausführung von einem **D-Latch**.
Ein Latch ist nicht flankengesteuert. Das Potential an D wird sofort auf den Ausgang durchgeschaltet, wenn Clock auf 1 steht. Es muss also sichergestellt sein, dass das Potential an D stabil ist, so lange Clock auf 1 steht.
Das ist nicht immer gewährleistet, zudem wären zwei weitere Eingänge wünschenswert, einer, mit dem der Ausgang Q unabhängig von D und Clock auf 1 gesetzt und ein weiterer, mit dem er auf 0 zurückgesetzt werden kann.
Auch das ließe sich mit einzelnen Boolschen Operatoren darstellen, würde hier aber zu weit führen.
Integrierte Schaltungen haben ihren Namen daher, dass eine Vielzahl von Funktionen bzw. Verknüpfungen auf einem Chip implementiert sind.

Es gibt eine Vielzahl von ICs, in denen Flipflops integriert sind, als Latch oder flankengesteuert. Bis zu 8 Flipflops werden auf einem Chip untergebracht.

Das älteste und klassische **D-Flipflop** ist das IC SN7474, das zwei Funktionseinheiten in einem Gehäuse vereinigt.
Es ist flankengesteuert und hat einen Setz- und einen Rücksetzeingang.

Symbol D-Flipflop



Funktionstabelle D-Flipflop

Der Übersichtlichkeit halber wurden /S und /R eingesetzt statt S und R, wie es üblich ist.

/S	/R	CK	D	Q	/Q
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1	1
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Keine Änderung	Keine Änderung

X bedeutet 1 oder 0.

Auch hier gibt es die unzulässige Möglichkeit, dass die Ausgänge kein gegensätzliches Potential aufweisen. In der Praxis wird jedoch meist nur /S oder /R verwendet und der jeweils andere fest mit Potential 1 versehen.

Wegen seiner Vielfältigkeit ist dieses Flipflop in seiner Originalversion wie auch in davon hergeleiteten Versionen zahlreich in der Flipper-Elektronik vertreten.

Abschließend eine einfache Schaltung, wo es als asynchroner Frequenzuntersetzer fungiert: