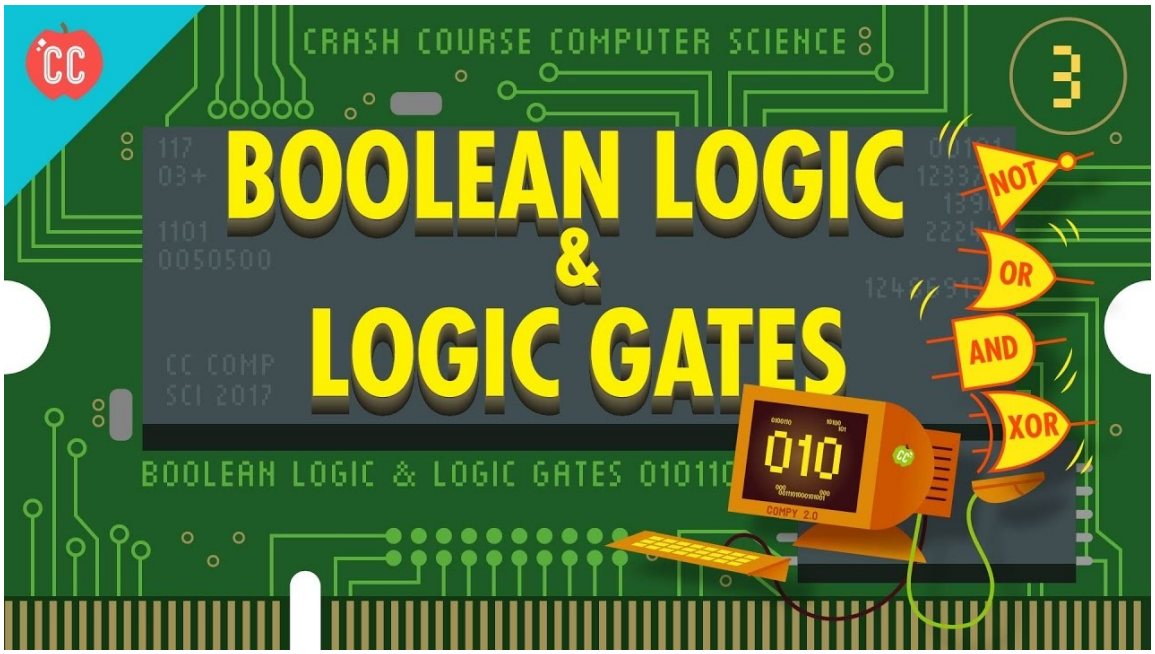
Logik-Elemente als Basis der arithmetischen Elemente eines Digitalrechners



Leibniz hat schon im 17-ten Jahrhundert diese Verbindung gesehen.

nach Wikipedia:
"Im weiteren Sinne war **Leibniz wegbereitend** für die Rechenmaschine im heutigen Sinne, den **Computer**. Er entdeckte, dass sich Rechenprozesse viel einfacher mit einer **binären Zahlencodierung** durchführen lassen, und ferner, dass sich mittels des binären Zahlencodes die ***Prinzipien der Arithmetik*** mit den ***Prinzipien der Logik***
verknüpfen lassen (siehe De progressione Dyadica, 1679; oder Explication de l'Arithmetique Binaire, 1703). Die hier erforschten Prinzipien wurden erst 230 Jahre später in der Konstruktion von Rechenmaschinen eingesetzt (z. B. bei der Zuse Z1)!
"

nach Wikipedia:
„Erst mit der Einführung der **Null** (in Indien bis ins 7. Jahrhundert n. Chr. hinziehende **„Entdeckung" der Zahl Null und durch die Einführung eines Schriftzeichens für diese** als vollwertige Ziffer, die 0), ist das Stellensystem so leistungsfähig geworden, wie es heute als selbstverständlich erachtet wird, mit dem nicht nur Zahlen dargestellt werden können, sondern auch einfach gerechnet werden kann.
Über Arabien kam diese Kenntnis im 13. Jahrhundert durch Fibonacci nach Europa und erst im 16. Jahrhundert verbreitete Adam Ries mit seinen Rechenbüchern (Rechenbrett) das **Stellenwertsystem** und das schriftliche Rechnen im deutschsprachigen Raum

„
Stellen-Wert-System
Wie der Name bereits vermuten lässt, wir die Darstellung einer Zahl durch einzelne Ziffern, die an der jeweiligen Stelle einen bestimmten Wert haben, dargestellt.
z.B. das Dezimal-Stellenwertsystem:
Wenn eine Zahl **4361** auf dem Papier steht, weiß jeder sofort:

vier-**tausend** drei-**hundert ein** und sechs-**zig**

Leider vertauschen wir in Deutschland die letzten beiden Stellen beim Nennen. Die Engländer machen das exakter:

vier-**tausend** drei-**hundert** sechs-**zig** und **eins**
(*four-thousand three-hundred and six-one*)

Der Stellenwert der Ziffern ist also: tausend, hundert, zehn, eins
Würde man z.B. bei der Zahl 4301 die 0 weglassen würde da 431 stehen, diese Zahl hat aber einen ganz anderen Wert. Die 0 ist eine Ziffer die keinen Wert besteuert, aber die Stellenwerte der vorhergehenden Ziffern sichert.
Das Verhalten kann man nun gut auch mathematisch beschreiben:

$$4361 = 4 * 10^3 + 3 * 10^2 + 6 * 10^1 + 1 * 10^0 = 4000 + 300 + 60 + 1$$

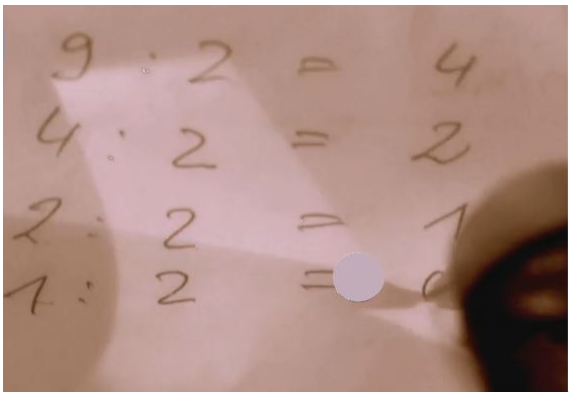
Gleichzeitig mit der Definition der 0 wurden auch Rechenregeln festgelegt, so gilt:

x + 0 = x,
x * 0 = 0 sofern $x \neq \infty$ ist und
$\mathbf{x^0 = 1}$ sofern $x \neq 0$ ist.

Damit ist der letzte Stellenwert $10^0 = 1$ und ist die Ziffer 0, ist das Ergebnis durch die Multiplikation auch 0 und im Wert der Zahl bringt die Stelle keinen Beitrag, da x + 0 = x ist!

Wie man vermuten kann, ist das die Beschreibung des Dezimalzahlensystems, mit den Eigenschaften: Basis = 10; Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Die Regeln gelten für alle Zahlensysteme. Aus der Informatik sind dazu das **Oktal-Zahlensystem, das Hexadezimal-Zahlensystem und das Dual-Zahlensystem** bekannt.



Aus den Aufzeichnungen von **Leibniz** geht hervor, dass er den heute noch genutzten Algorithmus zur Umrechnung von Dezimalzahlen in Dualzahlen kannte (erfunden?) und nutzte:

Wert	div. ganzz/2	ganzz Erg	Rest
9	: 2	= 4	**1**
4	: 2	= 2	**0**
2	: 2	= 1	**0**
1	: 2	= 0	**1**

Der Rest von unten nach oben dargestellt ergibt:
$\mathbf{1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =}$
8 + 0 + 0 + 1 = 9

Die Basis aller Zahlensysteme im Computer ist das Dual-Zahlensystem, geschuldet der Tatsache, das es bei Schaltern nur zwei Zustände, nämlich aus oder ein, bzw.. 0 oder 1 gibt.
Basis: 2; Ziffern: 0, 1
Damit muss es möglich sein alle Zahlenwerte darzustellen, z.B.

$$11001_{(2)} = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 = 16 + 8 + 0 + 0 + 1 = 25_{(10)}$$

Wie in Wikipedia dargestellt wurde, ist auch in den Stellenwertsystemen einfach zu rechnen, also auch im Dualzahlensystem: z.B.

```
  11001
+     1
  --------
```

Es wird wie dezimal beim geringsten Stellenwert begonnen und gleich gibt es ein Problem:
1 + 1 wäre ja 2, die Ziffer gibt es aber nicht!

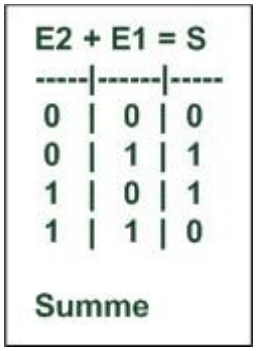
Es wird wie dezimal bei 9 + 1 verfahren: es ist 0 aber in der nächsten Stelle wird der Wert um 1 erhöht, so auch dual, das Ergebnis ist 0 aber in der nächsten Stelle der Wert + 1 :

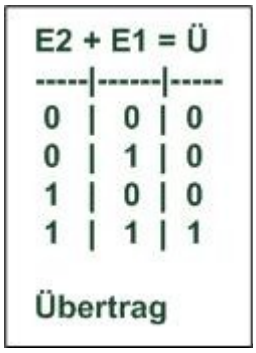
```
          11001
        +     1
          --------
Übertrag: 00010
Summe:    11010
```

Für die Addition von beliebigen zwei Zahlenwerte gilt:

E2	E1	S	Ü
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Nun kann man sich die Summe und den Übertrag noch einmal einzeln ansehen:

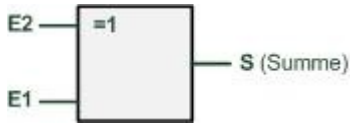


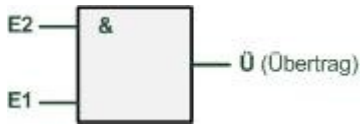


Man kann vermuten, dass **Herr Leibniz** das so ähnlich dargestellt hat und überrascht war, dass die Ergebnisse denen logischer Funktionen entsprechen.
Der **Übertrag** entspricht der **AND-Funktion**, die **Summe** der Antivalenz bzw. der **XOR-Funktion**.

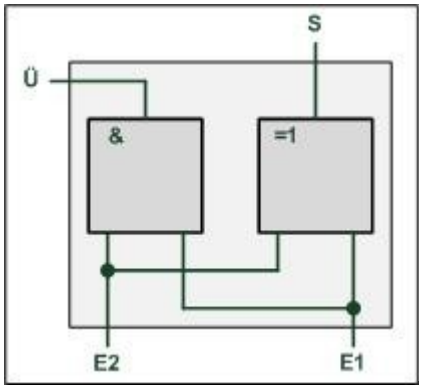
Also, die Parallelschaltung der logischen Funktionen **XOR** und **AND** realisiert die **arithmetische Rechenoperation Addition**!

Für die logischen Funktionen sind die folgenden Symbole bekannt:



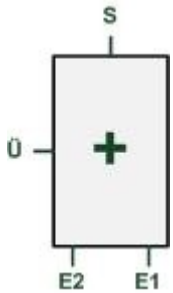


Diese kann man nun einfach ohne Wissen einer Realisierung zusammenschalten und erhält den Addierer (für zwei Werte):

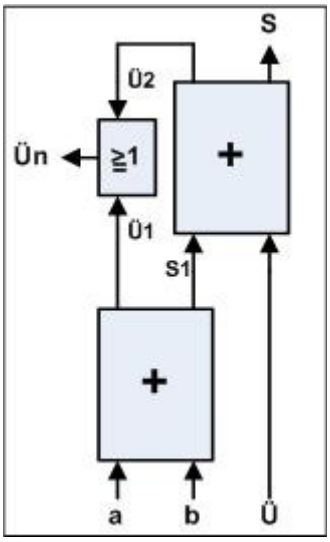


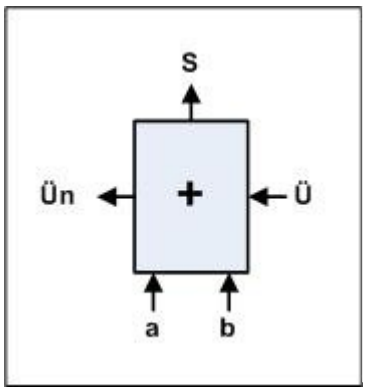
Für diese Anordnung wurde ein neues Symbol erstellt:

Man bezeichnet es als **HALB-ADDER**



Diese Anordnung funktioniert allerdings nur in der ersten Spalte (rechts) der Addition, denn bei allen folgenden kann es notwendig sein, auch den von der Vorstelle erzeugten Übertrag dazu zu rechnen. Das ist nicht kompliziert, man addiert die Summe mit dem Übertrag mit einer weiteren solchen Schaltung:





Zusätzlich wird noch eine logische Funktion **OR** gebraucht.
Die Schaltung wird als
VOLL-ADDER
bezeichnet, man schreibt dafür ein neues entsprechendes Symbol (rechts).

Technische Realisierung eines HALB-ADDERs

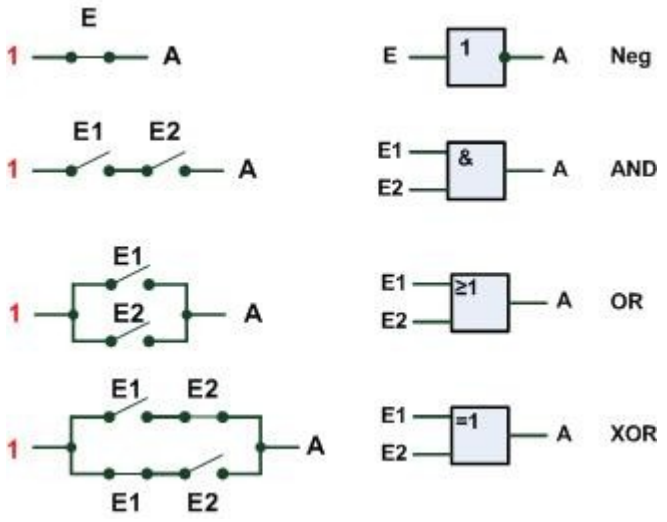
Zur Realisierung der logischen Funktionen sollte man auch immer die Schaltfunktion des Logik-Elementes parat haben, denn hat man keine einfache Lösung zur Hand, kann man diese durch Zusammenschaltung einfacher Funktionen realisieren, das sind die folgenden Funktionen:

log. **AND**: $A = E1 \wedge E2$
log. **XOR**: $A = E1 \wedge \neg E2 \vee \neg E1 \wedge E2$
log. **OR**: $A = E1 \vee E2$
Neg. $A = \neg E$ wird ebenfalls als **NOT** bezeichnet

mit: $\wedge$ AND, $\vee$ OR, $\neg$ Negation (NOT)(von)

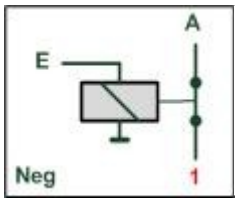
Für die Realisierung können nun die unterschiedlichsten technischen Medien genutzt werden, wie z.B. Elektrik, Elektronik aber auch Mechanik. Im Folgenden sollen deshalb einige logische Funktionen mit verschiedenen technischen Möglichkeiten dargestellt werden

Eigentlich braucht man nur eine gut funktionierende Lösung für die NAND-Funktion, denn mit Hilfe der Booleschen Algebra kann man alle anderen Elemente erzeugen, allerdings kann man meist einfachere direkte Lösungen für die jeweilige Funktion finden.



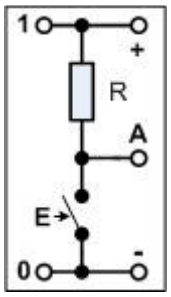
Am Besten kann man sich die Funktion des logischen Elements mittels Schalter veranschaulichen.
Die Negation ist somit ein geschlossener Schalter, wird er betätigt, wird er geöffnet, es gibt dann keinen Stromfluss durch den Schalter.

Negation: Relais-Realisierung
Zuse Z3 (Deutschland), Mark1 (USA)
Relais sind logische Bauelemente, denn sie können genau zwei Zustände darstellen, also muss man mit ihnen auch ein **arithmetisches PLUS** (**ADDER**) realisieren können.



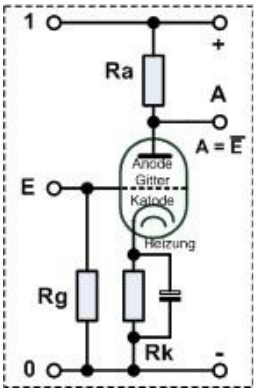
Das einfachste Element ist ein Negator, der Handschalter wird durch eine Relaisspule ausgetauscht, die Eingabe E = 1 bewirkt, dass der Schalter geöffnet wird,
A = ¬E = ¬1 = 0.
Der geschlossene Schalter stellt sofort den Ausgangszustand her, **A = 1**.

Negation: Elektronische-Realisierung
Man braucht ein elektronisches Bauelement mit dem eine Schaltfunktion realisierbar ist.
Die historisch älteste Lösung ist die Nutzung einer Elektronen-Röhre.



Das Bild zeigt die Basis einiger elektronischer Negationen. Anstelle des Schalters und E wird das elektronische Schaltelement eingesetzt.
Ist der Schalter offen liegt an A eine 1 (mit entsprechender Dimensionierung von R). Also E hat den Wert 0, am Ausgang erscheint eine 1, das entspricht der Negation.
Der Schalter wird geschlossen (E = 1), dann wird über den Schalter eine 0 an A gelegt. Ganz wichtig ist nun der Widerstand R, er verhindert einen Kurzschluss zwischen 1 und 0 (zwischen Betriebsspannung + und -)

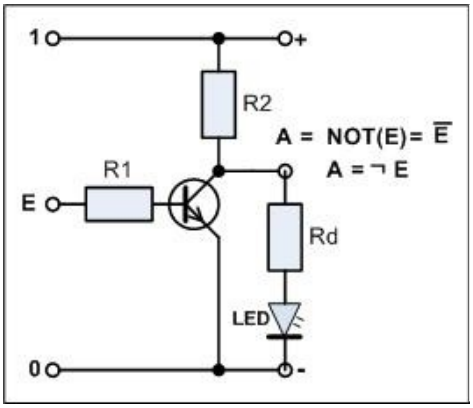
Negation: Elektronische-Realisierung mittels Elektronen-Röhre
ENIAC (USA) , Zuse Z22 (Deutschland), ZRA1 (DDR), BESM-1 (Sowjetunion)



Die Widerstände Ra und Rk dienen der Einstellung des Arbeitsbereiches der Röhre (Ra begrenzt den Strom durch die Röhre, Rk sorgt für eine negative Gittervorspannung).
Es gilt das Prinzip wie oben beschrieben, wird die Röhre über E nicht angesteuert, so hat sie einen größeren Innenwiderstand, an A liegt nahezu Betriebsspannung, also 1. wird E angesteuert (E = 1) wird ihr Widerstand klein und legt die Masse an den Ausgang (A = 0).

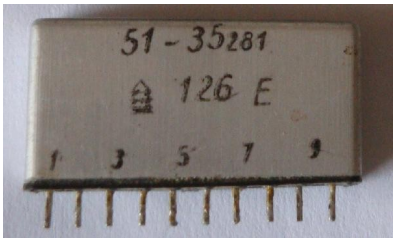
Theoretisch funktioniert das so, praktisch ist im angesteuerten Zustand der Widerstand nicht 0 wie beim Schalter und auch nicht ∞ bei Nichtansteuerung.

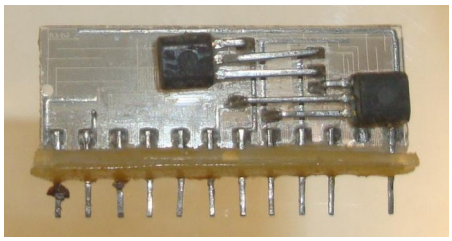
Negation: Elektronische-Realisierung mittels bipolarerTransistoren
TRADIC (USA) , IBM 608 (Deutschland), R100 (DDR)



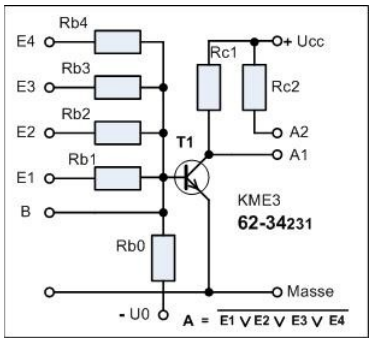
Der bipolare Transistor funktioniert etwa wie ein Schalter. Wird an E eine 1 gelegt, wird die Emitter - Collector – Strecke sehr niederohmig, quasi ein geschlossener Schalter, dann liegt an A der Wert 0 an.
Wird E mit 0 angesteuert, dann ist A = 1, die LED leuchtet.

Eine quasi **Zwischenstufe** zu integrierten Lösungen waren **KME3 Bausteine**.

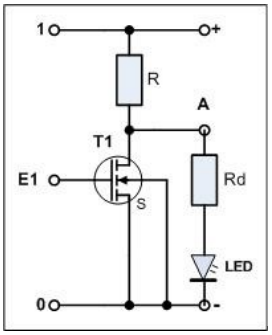




Es waren Integrierte Schaltkreise in **Dünnschicht-Hybrid-Technik**. Die Bausteine enthalten vorrangig digitale logische aber auch analoge Grundschaltungen.
Auf einem Glasträger (anfangs auch Keramik) wurde ein Widerstandsnetzwerk im Vakuum aufgedampft (alle Widerstände der Schaltung). Alle anderen Bauelemente, wie Transistoren, Dioden und Kondensatoren (kleinere wurden später auch durch Aufdampfen erzeugt) wurden auf das Netzwerk aufgeklebt und mit dem Netzwerk verlötet. Es wurde die NOR-Funktion realisiert, aus der sich ebenfalls alle anderen Funktionen erzeugen lassen.
Der Vorteil einer Lösung mit diesen Bauelementen war, dass außen kaum noch diskrete Bauelemente gebraucht wurden (diese Technik wurde jedoch schnell von integrierten Schaltkreisen überholt, KRS 4200 (DDR)).



Negation: Elektronische-Realisierung mittels unipolarer Transistoren

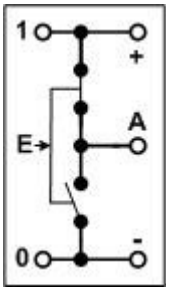


Feld-**E**ffekt-**T**ransistoren (FET)

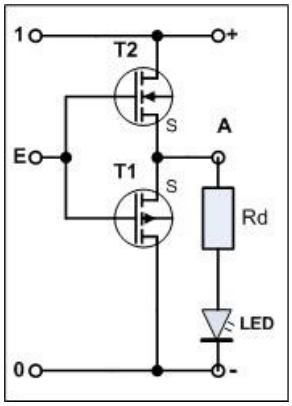
Das ist etwa die gleiche Schaltung wie mit der Röhre, es wird der Elektronenfluss gesteuert. Die Zustände des Transistors (FET) nähern sich dem eines Schalters, jedoch muss mit R die Stromstärke durch den Transistor begrenzt werden, das erfordert wieder Kompromisse.

Weitere Schaltungen mit FETs

Die Basis zu diesen Bauelementen sind zwei in Reihe geschaltete Schalter, ein Schließer und ein Öffner. Beide Schalter werden durch das gleiche Signal angesteuert:



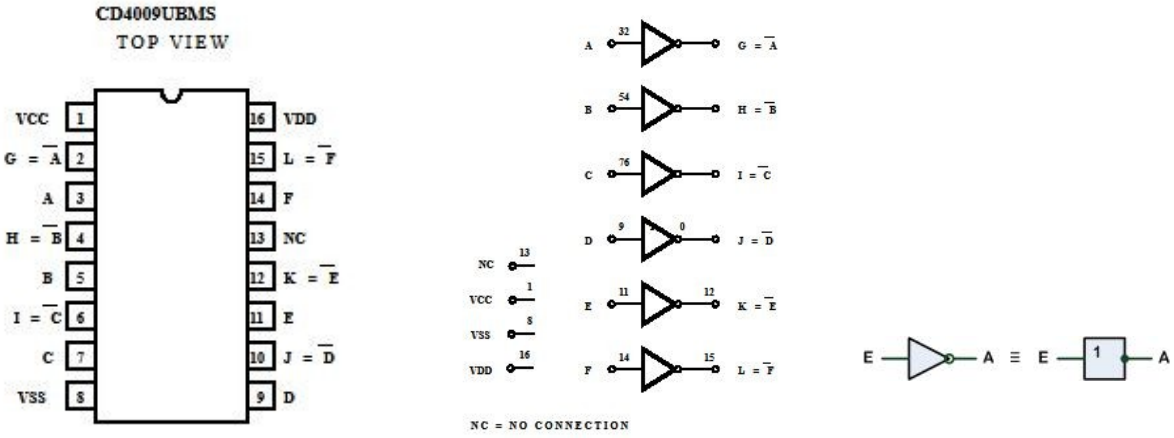
Das Bild zeigt den Zustand bei E = 0.
Der obere Schalter ist geschlossen, der untere offen, die 1 wird an den Ausgang A gelegt.
Bei E = 1 dreht sich die Situation. Der obere Schalter ist offen der untere geschlossen, A erhält der Wert 0.
Das Problem ist der Übergang zwischen den beiden Zuständen, es könnte sein, dass beide Schalter kurzzeitig offen sind – was passiert an A?



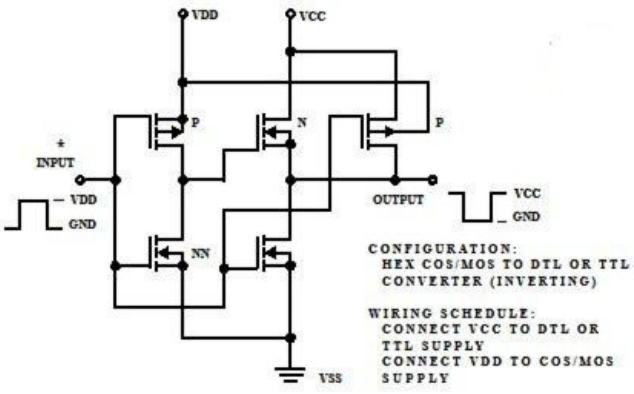
Metall-Oxid-Halbleiter-Feldeffekttransistor
(englisch: **m**etal-**o**xide-**s**emiconductor **f**ield-**e**ffect **t**ransistor)
kurz: **MOSFET** oder MOS-FET

Es gibt dabei solche die in nicht angesteuerten Zustand leitend sind und andere die sperrend sind, in der Schaltung ist unten ein sperrender(Pfeil raus) und oben ein leitender (Pfeil rein). Bei Ansteuerung E = 1 drehen sich die Verhältnisse um, oben sperren, unten leiten. Damit wird mit E = 1 an A = 0.
Das entspricht nun genau dem zuvor dargestelltem Prinzip mit Schaltern. Der Vorteil ist, dass die Ansteuerung nahezu stromlos erfolgt, der Energieverbrauch gering ist.
Dies ist die Basis von nahezu allen Computer-Realisierungen!

Negation: Realisierung mittels elektronischen Schaltkreisen



Das bietet unter anderen die Industrie an, die Negatoren, es sind 6, sind natürlich wesentlich sicherer als die Prinzip-Schaltung (IC **4009**):

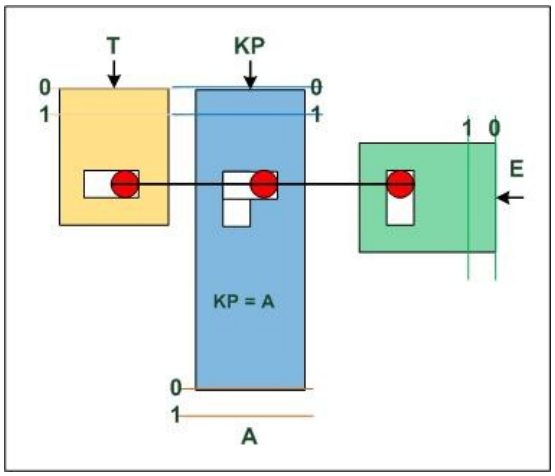


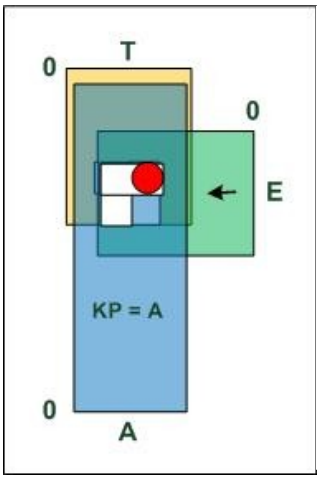
Die ersten beiden FETs realisieren die Funktion Negation (oder auch NOT). Die weiteren FETs dienen der sicheren Funktion des Bauelementes (z.B. Kurzschlussfest).

Negation: Mechanische Lösung
Zuse Z1 1937 (Deutschland)

Alle bis zu diesem Zeitpunkt erdachten und erstellten Rechenmaschinen waren mechanische Rechenmaschinen im Dezimalsystem
Wilhelm Schickard, Rechenmaschine mit Zahnrädern und Übertrag, 15/16-hundert;
Leibniz, Rechenmaschine mit Stufenzahnradwalze, 16/17-hundert).
Aber trotz dessen, dass es schon Relais gab (Joseph Henry, Relais-Stationen in der Telegrafie, 1935) und der bereits vorhandene Elektronrnröhre (unabhängig voneinander
R. von Lieben (1878 – 1913); **Lee de Forest** (1873 - 1961))
hat **Herr Zuse** konsequent den Gedankenansatz von **Leibniz** fortgesetzt und eine auf dualer Basis funktionierende Rechenmaschine konstruiert und aufgebaut, allerdings mit mechanischen Logik-Elementen. Es war eine programmgesteuerte Rechenmaschine mit den Blöcken Rechenwerk, Speicher, Steuerwerk usw., also entsprechend heutiger Computer.
Warum Herr **Zuse** nun trotz bestehender Voraussetzungen eine mechanische Lösung gewählt hat, ist mir unbekannt. Ich hätte ihn fragen können, denn ich habe ihn in einem Vortrag erlebt.

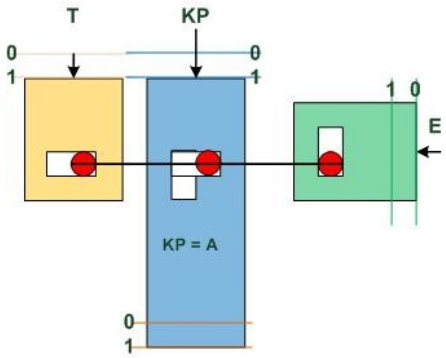
Mechanische Lösung:
Er hat drei Bleche (mindest) mit speziellen Ausschnitten zueinander angeordnet, die mit einem frei beweglichen Koppelstift verbunden sind. Die Bleche können sich nur in einer Richtung bewegen.





Es gibt eine Takt (T)-, Eingabe(E)- und Koppelplatte(KP). Die mögliche Bewegungsrichtung ist durch die Pfeile gekennzeichnet. Der rote Kreis ist der Koppelstift, der sich in den Ausschnitten der Platten bewegen kann, Die Platten sind in beliebiger Folge übereinander angeordnet,

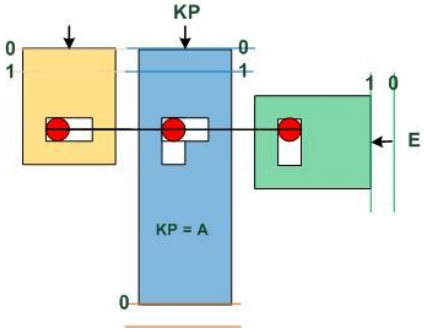
der Stift steht in allen Ausschnitten der Platte (der Strich im linken Bild soll das darstellen, die drei Kreise sind hier ein Stift). Besonderheit ist, dass die Koppelplatte zugleich Ausgabe ist.



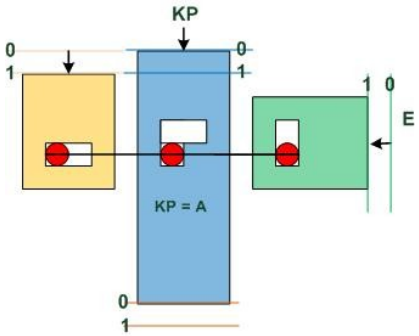
Das Betätigen des Taktes (nach unten) bewirkt ein Verschieben des Koppelstiftes nach unten. Das geht aber nicht, da in der Koppelplatte keine Bewegung nach unten möglich ist, es muss die Koppelplatte nach unten geschoben werden, in der Eingabeplatte kann der Stift nach unten rutschen. Der Ausgang zeigt auf 1. Das ist richtig, denn:

A = ¬E = ¬0 = 1

Nun muss man noch den anderen Fall untersuchen.

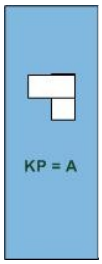


Der Eingang wird auf 1 gesetzt.
Der Stift wird nach links geschoben, was sowohl in der Taktplatte wie auch in der Koppelplatte möglich ist.
Im Gegensatz zum vorhergehenden Fall ist nun aber in der Koppelplatte unter dem Stift ein Schlitz.



Der Stift kann mit Betätigung der Taktplatte nach unten rutschen, die Koppelplatte bleibt an der Stelle.
Das ist exakt denn:

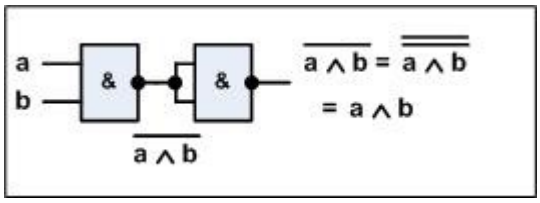
A = ¬E = ¬1 = 0



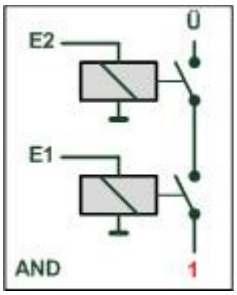
Dreht man die Koppelplatte um, erhält man einen Folger, einen Schließer.

Realisierung: AND bzw NAND

Die AND-Funktion wird ,wie oben beschrieben, zur Realisierung der arithmetischen PLUS-Funktion gebraucht.
Es gibt solche Realisierungen, aber oft ist es einfacher NAND zu erstellen und ein NOT folgen zu lassen, das ergibt auch AND:



AND: Relais-Realisierung



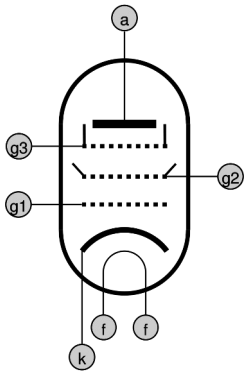
Man braucht zwei in Reihe geschaltete Schalter (Schließer), die in diesem Fall jeweils mit einem Relais betätigt werden.
Die 1 wird nur zum Ausgang durchgestellt, wenn Schalter1 und Schalter 2 geschlossen sind:

A = E1 AND E2

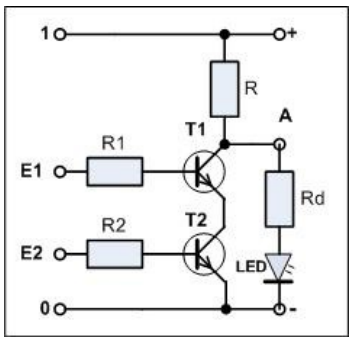
Anderenfalls wird nichts am Ausgang bereitgestellt, es wird als 0 interpretiert

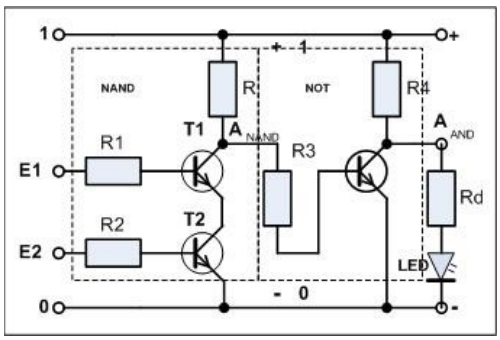
AND: Realisierung mit Elektronenröhre

Die übliche Nutzung führt auf eine Negation hinaus.
Verwendet man eine Pentode (Röhren mit weiteren Gittern), kann man den Elektronenstrom durch Ansteuerung mehrerer Eingangssignale Steuern. Am Ausgang wird das Ergebnisse wieder negiert, also NAND.
Man braucht also einen weiteren Negator.
Beides ginge wahrscheinlich in einem Gehäuse, z.B. ECH, eine Triode und eine Heptode.



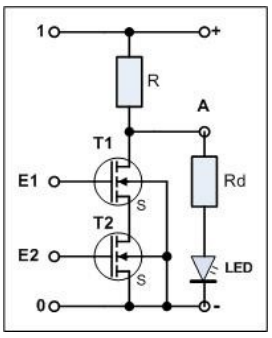
AND: Realisierung mittels bipolarer Transistoren

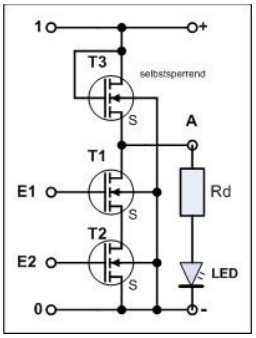




Das AND wird durch Negation von NAND erreicht.

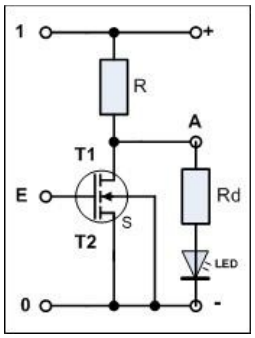
AND: Realisierung mittels unipolarer Transistoren

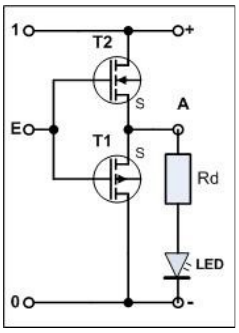




(linkes Bild) Die Realisierung der NAND-Funktion erfolgt etwa so wie mit bipolaren Transistoren.
Das rechte Bild realisiert den Widerstand R ebenfalls durch einen FET. Konstruktionsbedingt kann man ihn mit einem bestimmten Widerstand herstellen

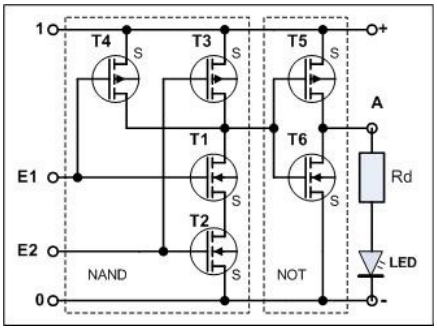
Dahinter braucht es wieder einen Negator um aus dem NAND ein AND herzustellen.





Die erste Variante eines FET-Negators (links) entspricht der bipolaren Transistorlösung, die zweite funktioniert nur mit komplementären FETs,

AND: Realisierung mittels unipolarer Transistoren - CMOS-FET



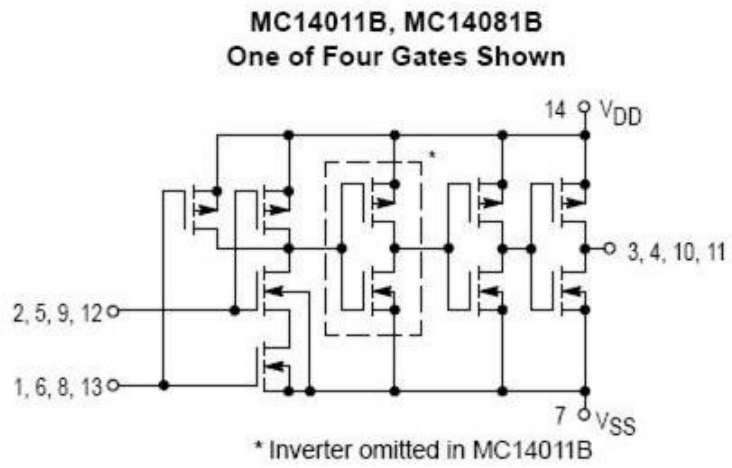
Auch hier wir wieder ein NAND durch Negation zum AND. Interessant ist, dass das NAND nur durch elegante Schaltung von zwei Negatoren entsteht.
Bei E1, E2 = 0 sind T4 und T3 leitend, T1 und T2 sperrend (offen), am Ausgang vom NAND erscheint 1.
Nun wird ein Eingang 1 der andere 0, ein Transistor (T3 oder T4) bleibt geschlossen und T1 oder T2 ist offen (sperrend), es gibt also keine Verbindung zu 0, sonder von oben eine zu 1.
Werden beide Eingänge auf 1 gelegt (E1 = E2 = 1), dann öffnen beide Transistoren oben (T3, T4) und T1 und T2 werden geschlossen, 0 wird an den Ausgang gelegt, die Schaltung zeigt NAND-Charakter:

E1	**E2**	**A_{NAND}**
0	**0**	**1**
1	**0**	**1**
0	**1**	**1**
1	**1**	**0**

Durch den anschließenden Negator (Transistoren T5 und T6) kommt als Ergebnis (Ausgang A):
0 0 0 1
heraus, das entspricht AND.

AND: Realisierung mittels elektronischen Schaltkreisen

Wer nun glaubt, neue Informationen zu bekomme, wird enttäuscht, denn die Industrie stellt AND-Schaltkreise nach dem eben beschriebenen Prinzip her:



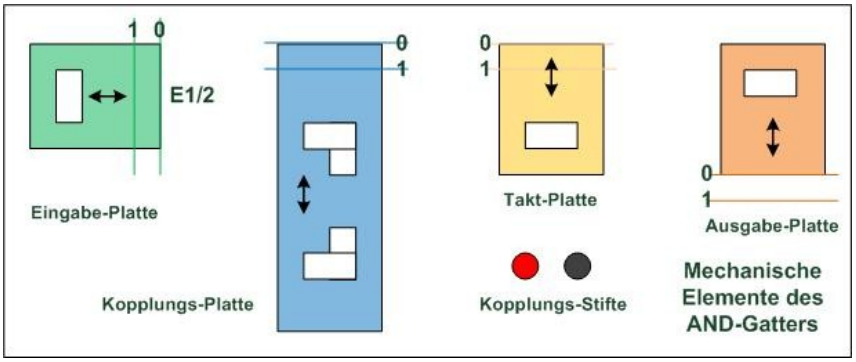
IC 4011

Die ersten 4 Transistoren entsprechen dem beschriebenen NAND mit CMOS-FETs.
Wird der gestrichelte Negator realisiert, ist es ein AND Gatter, das IC hat dann die Bezeichnung
IC 4081
Die beiden weiteren Negatoren haben keine logische Funktion, denn es gilt auch hier:

¬¬X = X

(die Negation der Negation hebt sich auf). Sie dienen der Strom- und Spannungsanpassung des Schaltkreises.

Mechanische Lösung:
Auch Herr Zuse kam bei dem Bau des Rechners Z1 (ein mechanischer Dualrechner) nicht umhin, AND-Gatter zu bauen und das natürlich mechanisch.
Das oben gezeigte Prinzip des Negators wird weiterentwickelt, es müssen nun zwei Eingänge den Ausgang entsprechend der AND-Funktion steuern. Folgende Platten werden dazu gebraucht:



Neu gegenüber dem Negator ist, dass die Koppelplatte zwei Ausschnitte hat, für die Ausgabe eine Ausgabeplatte und zwei Koppelstifte und Eingabeplatten.
Der erste Stift (oben) bewegt die Koppelplatte als Funktion von der Eingabe E1 und dem Takt T:

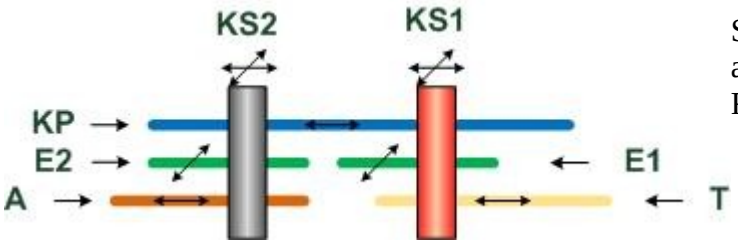
KP = f(E1,T)

Der zweite Stift (unten) bewegt die Ausgabeplatte als Funktion von E2 und KP:

A = f(E2,KP)

Der Takt T geht über die Koppelplatte in die Funktion von Stift2 ein.

Die Ausschnitte in der Koppelplatte stellen (wie man sich denken kann) die logische Funktion dar.



So werden die Platten übereinander angeordnet (die Reihenfolge spielt keine Rolle)

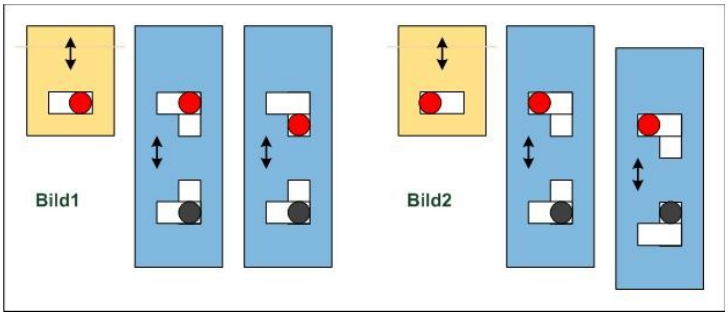


Bild1:
Die Eingabeplatte steht auf 0, somit steht der rote Kopplungsstift rechts in der Kopplungsplatte (auch der schwarze).Wird nun der Takt betätigt (auf 1 nach unten geschoben) rutscht der rote Kopplungsstift im Schlitz der Kopplungsplatte nach unten – die Kopplungsplatte wird nicht bewegt! Der schwarze Stift bleibt deshalb an seiner alten Position.

Bild2:
Die Eingabeplatte steht auf 1, also wurde der rote Stift nach links geschoben. Jetzt bewegt der Takt 1 die Kopplungsplatte nach unten, da es an dieser Stelle keinen Schlitz nach unten gibt. Aber der schwarze Stift kann in dem Schlitz nach oben rutschen – er bewegt sich nicht von der Stelle, d.h. die Ausgabe bleibt unverändert.
Nur wenn die Eingabe2 den schwarzen Stift nach links geschoben hatte, würde auch der schwarze Stift nach unten geschoben (andere Ausgabe).
Genau so soll ein AND-Gatter funktionieren, nur wenn beide Eingänge auf 1 stehen wird auch die Ausgabe auf 1 gesetzt!

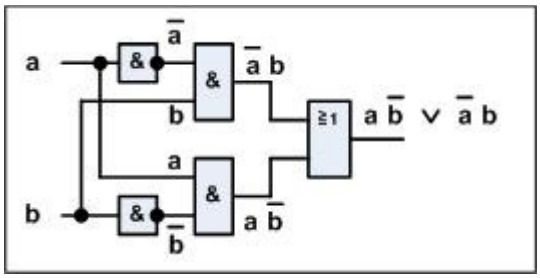
(hier gibt es ein Beispiel der AND-Simulation)

Realisierung: XOR

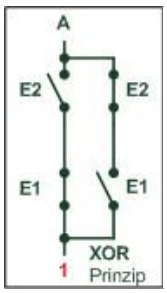
Um ein Plus-Element herzustellen, braucht man nun noch ein logisches XOR-Element.
Wenn man dazu keine spezielle Lösung findet, kann man immer die komplette Schaltfunktion realisieren. Hier fehlt allerdings noch das logische OR:

$$y = a\bar{b} \vee \bar{a}b$$

Aber gerade bei XOR gibt es bei den verschiedenen Realisierungsmedien sehr erstaunliche einfache Lösungen.



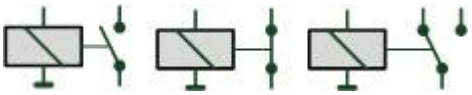
Das ist die einfach aus den logischen Schaltfunktionen zusammengesetzte Schaltung für XOR.
Vereinfachungen wurden nicht durchgeführt



Aber auch die schematische Darstellung der XOR-Funktionen mit Schaltern, denn diese ist immer die Basis für die Relais-Lösung, sollte man immer kennen.

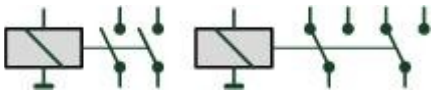
XOR: Relais-Realisierung

Es gibt einige verschiedenen Lösungen mit Relais, geschuldet der Tatsache, dass es unterschiedlichste Varianten davon gibt:

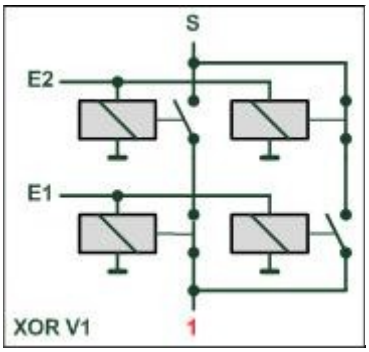


Bisher hatten wir nur die ersten beiden Varianten, damit kann man „Folger“ und „Negatoren“ realisieren.

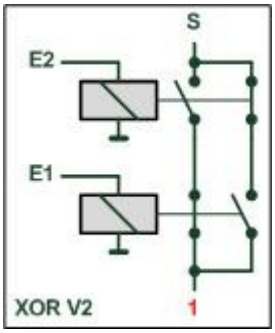
Die dritte Variante ist ein Umschalter, es ist immer eine Leitung geschaltet.



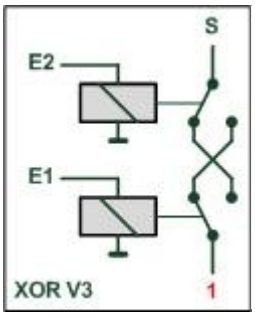
Relais können auch mehr als einen Kontakt haben, hier werden jeweils zwei dargestellt, es gibt aber auch solche, der sogar vier Kontakte haben, was viele neue Möglichkeiten eröffnet.
Mit den bereits dargestellten sind folgende XOR-Lösungen möglich:



Variante1:
Es wird einfach die oben dargestellte Schalterdarstellung mit mit vier Relais, zwei Schließer und zwei Öffner, zusammengeschaltet. Jedoch werden nur zwei Signale zur Steuerung gebraucht.



Variante2:
Hat man Relais mit zwei Umschaltern (siehe oben), kann die Realisierung des XORs auch so erfolgen. Man braucht nur noch zwei Relais.

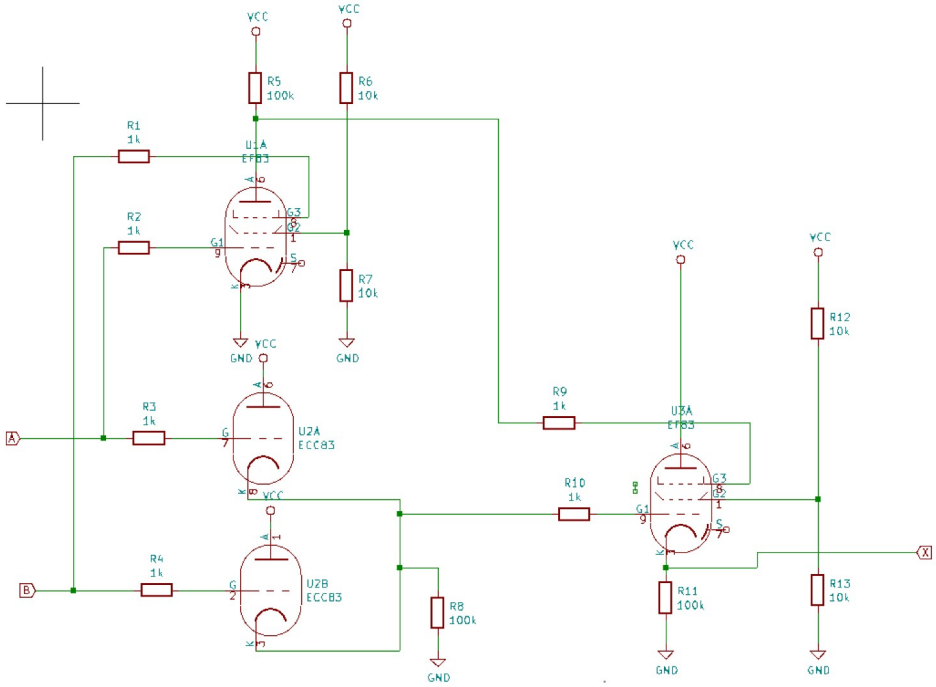


Variante3:
Diese Variante hat Herr Zuse wahrscheinlich bereits bei seinem Rechner Z3 genutzt (erfunden). Nur wenn ein Schalter umgeschaltet hat, kann Strom fließen.
Es lohnt sich doch manchmal die vorgegebenen Pfade zu verlassen!

Relais bieten sehr viele Möglichkeiten – leider sind sie zu groß, zu langsam, brauchen zu viel Energie usw.

XOR: Realisierung mit Elektronenröhre

Solche Lösungen muss es ja gegeben haben, denn es gab ja auch Rechner mit Elektronenröhren z.B. ZRA1 (DDR). Ich habe so eine Maschine noch in meinem Studium gesehen und es musste mächtig gekühlt werden, sonst hielten die Röhren nicht durch.
Im Internet habe ich dazu eine Lösung gefunden (habe die Quelle nicht parat, es ist **nicht** mein Eigentum).

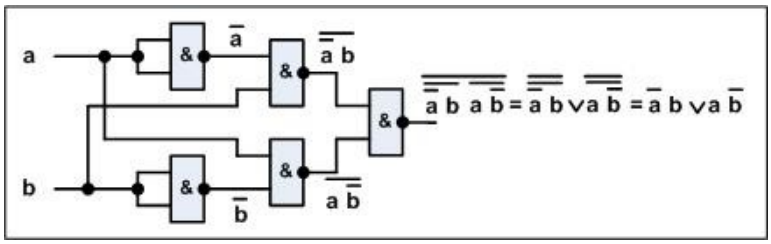


Es sieht so aus, als würde die Schaltfunktion exakt nachgebildet. Die beiden Trioden (eine Doppeltriode ECC83) realisieren die Negation des Eingangsignals. Mit den Pentoden (EF83) werden die beiden Eingangssignale mit AND verknüpf (z.B. A ∧ ¬B). Der Ausgang ist dann wieder eine Negation der zwei Zwischensignale (z.B. A∧¬B), also:

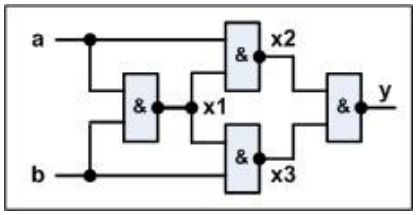
¬(ZwS1 ∧ ZwS2) = ¬ZwS1 ∨ ¬ZwS2

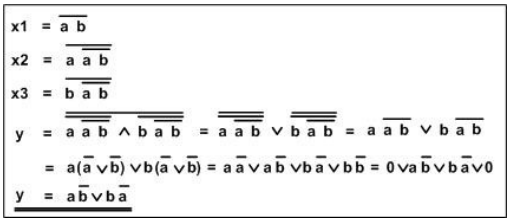
XOR: Realisierung mittels bipolarer Transistoren

Da sofort keine einfache Lösung bekannt ist, kann man die gegebene Schaltfunktion aus einzelnen logischen Elementen Realisieren. Das geht noch nicht da die OR-Funktion noch fehlt.

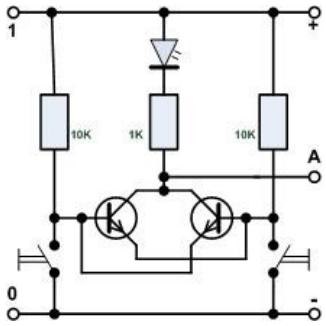


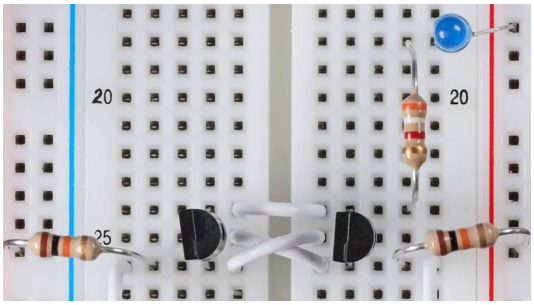
Man kann jedoch die ganze Schaltung mittels Boolescher Algebra umändern, so dass z.B. nur noch NAND-Elemente verwendet werden (wobei der Negator nicht mittels NAND hergestellt werden muss.
Mit NANDs kann man aber auch eine andere Schaltung erstellen:





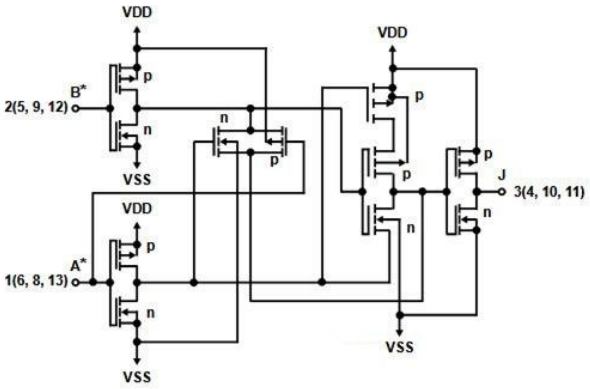
Das ist der klassische Lösungsansatz für Transistor-Lösungen.
Im Internet habe ich eine sehr interessante Lösung für XOR mit bipolaren Transistoren gefunden (Autor unbekannt, ich bin **nicht** der **Urheber**):





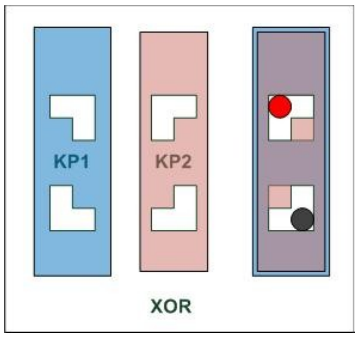
XOR: Realisierung mittels unipolarer Transistoren - CMOS-FET

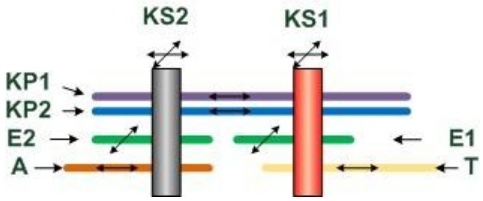
Bei der Realisierung von XOR mit diskreten FET Bauelementen oder mit Schaltkreisen bestehen kaum Unterschiede. Realisiert man das entsprechend der allgemeinen Schaltung mit NANDs, werden 16 FETs gebraucht.
Im IC werden ebenfalls CMOS_FETs eingesetzt, im IC 4070 gibt es vier XORs, die schon mit 11 FETs auskommen:



IC 4070; einer von 4 XOR auf dem IC

Mechanische Lösung:
Herr Zuse hat ja einen mechanischen Dualrechner zusammengebaut, folglich muss es auch für ein Adder ein mechanisches XOR gegeben haben. Wobei man aber immer zulassen muss, dass es auch spezielle Lösungen für ein spezielles Bauelement geben kann (siehe unten).
Für die Erstellung einer mechanischen logischen Funktion, müssen, wie oben gezeigt, spezielle Platten erstellt werden. Die jeweilige logische Funktion wird durch Ausschnitte in der Koppelplatte erzeugt.
AND wurde z.B. oben beschrieben. Durch Kombination von unterschiedlichsten Ausschnitten (habe ich probiert) kann man jedoch nicht die Funktion XOR erzeugen. Irgendwann kaum die Erleuchtung, dass Herr Zuse durch zwei Koppelplatten das Problem gelöst hat:





Die Platten werden wie dargestellt übereinander auf den Stapel der Platten gelegt, die Reihenfolge spielt keine Rolle. Die Koppelstifte verbinden alle Platten. Die Funktion dieser Anordnung ist nur schwer nachvollziehbar. Zwei Fälle (einmal gleiche Eingänge, einmal ungleich):

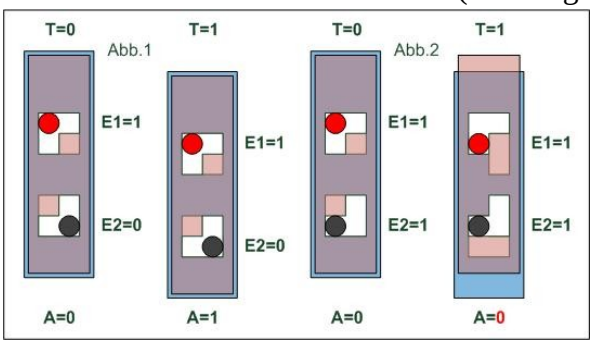
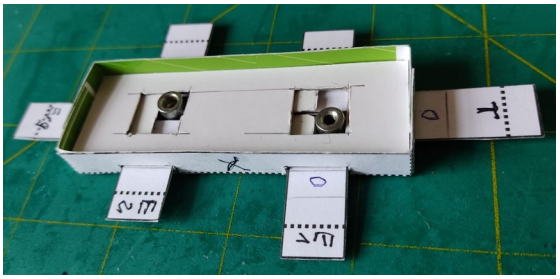
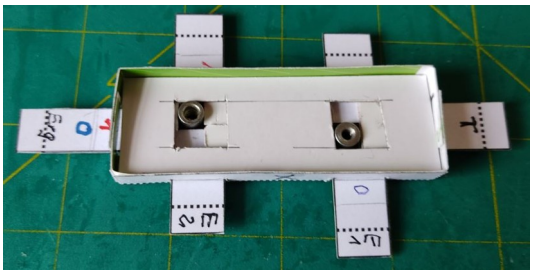


Abb.1: (E1=1; E2=0)
Der **rote Stift** steht links, in der blauen Platte muss er, wenn T=1 diese nach unten schieben (für rote Platte keine Wirkung, Schlitz nach unten).
Der **schwarze Stift** steht rechts, die blaue Platte schiebt ihn nach unten und somit auch die rote Platte. Das Ergebnis ist A=1, ist richtig für E1≠E2.
Abb.2: (E1=E2=1)
Der **rote Stift** steht links, in der blauen Platte muss er, wenn T=1 diese nach unten schieben (für rote Platte keine Wirkung, Schlitz nach unten).
Der **schwarze Stift** steht links, er führt nicht die Bewegung der blauen Platte aus, da über ihn ein Schlitz ist, auch die rote Platte bleibt damit stehen. Relevant ist, dass sich der schwarze **Stift nicht bewegt** und somit auch **nicht die Ausgabeplatte** (nicht dargestellt) A=0, ist richtig für E1=E2.

Für die beiden anderen Fälle kann man auch die richtige Funktion feststellen. Um das zu überprüfen, habe ich den Simulator von mechanischen Logikelementen von Herrn Timm Grams nachgebaut. Es ist ein Modell aus Pappe, man braucht kein Spezialmaterial oder Werkzeug für den Bau des Simulators:

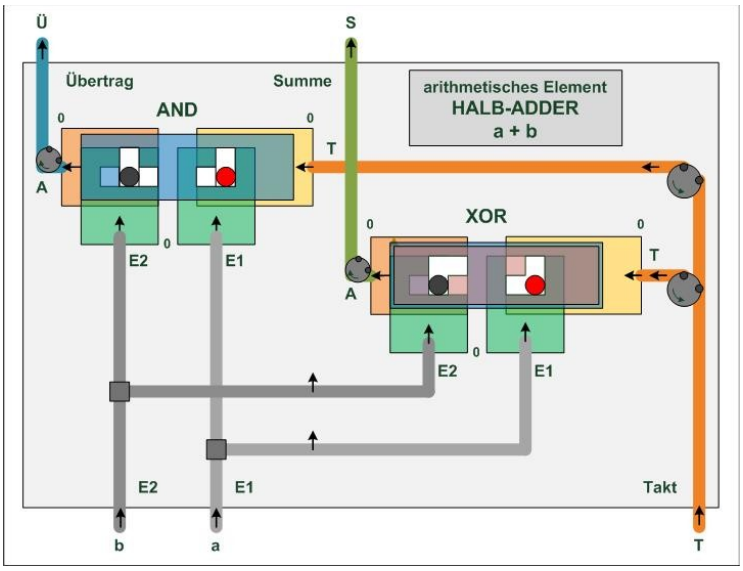




Timm Grams, Fulda; **Bastelbogen für ein mechanisches Schaltgliedmodell**

Der Simulator besteht aus einer kleinen „Pappkiste“ in denen mittels kleiner Schlitze die Eingabe(E_i)-, die Takt(T)- und die Ausgabeplatte(A; hier steht Erg.) bewegt werden können. Oben auf den Stapel liegen die Koppelplatten. Als Koppelstifte werden 6mm Distanzstücke verwendet (geht natürlich auch mit anderen Teilen). In dem Bild wird die Situation E1=0, E2=1, T=0 (links), T=1 (rechts) dargestellt.
Man sieht, dass die Ausgabeplatte ausgeschoben wird, also A=1.
Mit dem Modell kann man natürlich auch andere Koppelplatten ausprobieren.

Mit den gezeigten Möglichkeiten können nun mittels verschiedenster Medien Addierer aufgebaut werden. Auch mechanisch ist das möglich:

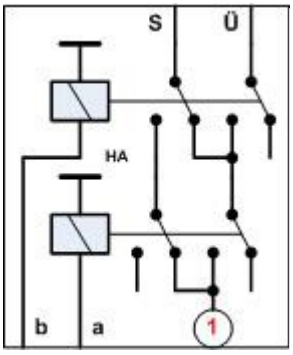


Wie oben festgestellt wurde, besteht auch die mechanische Lösung aus einem AND und XOR Element.
Es realisiert einen **Halb-Adder**. Für einen **Voll-Adder** braucht man die gleiche Anordnung noch einmal und zusätzlich ein OR-Element.
Die Verbindung der einzelnen Elemente erfolgt nun nicht elektrisch sondern mechanisch durch Stangen, wie auch die Ergebnisse, die durch Bewegung weitergeleitet werden können.

Ob das mit den vielen Platten und deren Lagerung und den Weiterleitstangen dauerhaft zuverlässig funktioniert hätte, konnte nicht nachgeprüft werden!

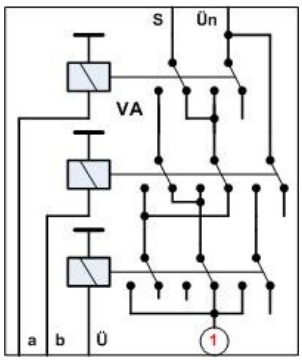
Ergänzungen

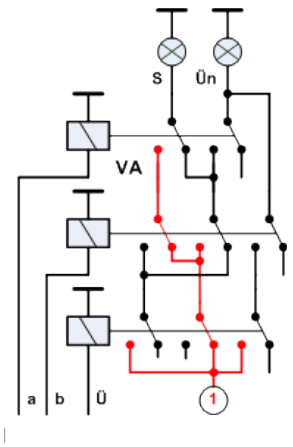
Für einige Realisierungen mit speziellen Medien, gibt es auch spezielle Lösungen, z.B. gibt es mit Relais sehr viele solcher Lösungen.
Basis dazu ist, das es Relais mit mehreren Umschaltkontakten gibt, üblich sind 2 oder 4, es gibt auch weitere mit mehr Kontakten.



Weiter oben bei der Darstellung der XOR-Funktion mittels Relais wurde auf Relais mit 2 Umschaltern hingewiesen.
Mit zwei solcher Relais kann man sehr einfach ein Halb-Adder realisieren, also alle zusammen AND und XOR!
Für einen Voll-Adder müssen wieder zwei solcher Gebilde zusammen geschaltet werden, OR braucht man bei Relais meist nicht.

Diese Schaltung, wie auch viele weitere unterschiedlichste wurden in: Digitalrechner, Grundlagen und Anwendungen;Friedr. Vieweg & Sohn; Braunschweig/Wiesbaden; 1990
veröffentlicht.





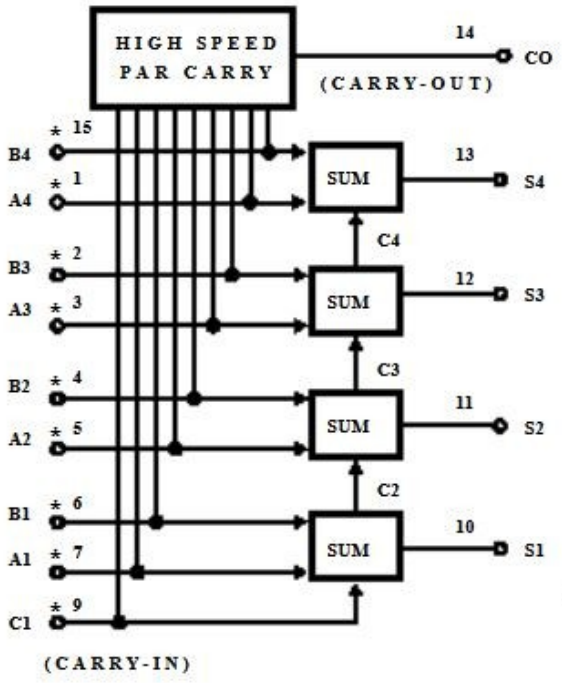
Simulation der Schaltung:

a	**b**	**Ü**
0	0	0
0	1	0
1	0	0
1	1	0
1	1	1

Hat man Relais mit mindest drei Umschalter zur Hand, kann man mit drei davon (2) einen Voll-Adder erstellen.
Oben an Ün gibt es eine Verbindung, das ist die OR-Funktion. Das geht, da Relais auch mit offenem Eingang funktionieren (im Gegensatz zu elektronischen Schaltung, da muss am Eingang immer ein definiertes Signal liegen (Pulldown, -up)

Höchst wahrscheinlich kann man auch bei den anderen Realisierungen kompakte Lösungen finden (habe ich nicht recherchiert).

Kompakte Lösung
Die Industrie hat auch für diesen Fall schon eine fertige Lösung!
Der Schaltkreis **IC 4008** beinhaltet vier komplette 1-Bit Addierer. Da braucht man nur noch a und b an die Eingänge legen und hat sofort die Summe am Ausgang.
Mit dem Übertrag funktioniert es etwas anders, die Überträge aus den 4 Addierern werden parallel gleichzeitig berechnet und als Gesamtübertrag für den 4-Bit-Addierer bereit gestellt. Dieses Verfahren hat die Bezeichnung: **Carry-Look-Ahead-Addierer**
Dieses Verfahren gestattet ein schnelles Addierwerk herzustellen im Gegensatz zur sequentiellen Berechnung (**Ripple-Carry Addierer**).

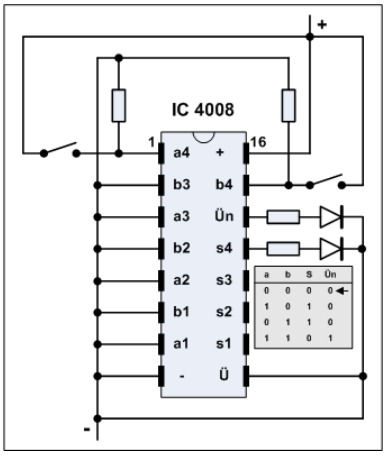


IC 4008

Description:
The NTE4008B is a 4−bit full adder in a 16−Lead DIP type package constructed with MOS
P−Channel and N−Channel enhancement mode devices in a single monolithic structure. This
device consists of four full adders with fast internal look−ahead carry output. It is useful in binary
addition and other arithmetic applications. The fast parallel carry output bit allows high−speed
operation when used with other adders in a system

Beschreibung:
Der NTE4008B ist ein **4-Bit-Volladdierer** in einem 16-poligen DIP-Gehäuse mit MOS-Konstruktion P-Kanal- und N-Kanal-Anreicherungsgeräte in einer einzigen monolithischen Struktur.
Der Schaltkreis beinhaltet vier Volladdierern mit schnellem internen **Look-Ahead-Carry-Ausgang**. Es ist nützlich im Binärformat Addition und andere arithmetische Anwendungen. Das schnelle

Parallel-Carry-Ausgangsbit ermöglicht eine hohe Geschwindigkeit Betrieb bei Verwendung mit anderen Addierern in einem System.



Simulation der Schaltung

mit:

∧ - AND: **X∧Y** oder **XY**

∨ - OR: **X∨Y**

¬ - Negation (NOT): **¬X** **oder** **¬(X∨Y)** X oder Y negiert

$\overline{X}$ oder $\overline{X \vee Y}$