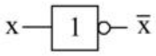
Junktoren

1. Negation

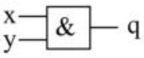
Die Verneinung eines Eingabewertes.



x	NOT
0	1
1	0

2. Konjunktion

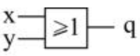
Die Und-Verknüpfung von zwei Eingabewerten.
„Nur wenn x UND y wahr sind, dann…“



x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1

3. Disjunktion

Die Oder-Verknüpfung von zwei Eingabewerten.
„Wenn x ODER y wahr sind, dann…“



x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1

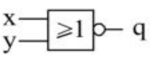
4. NAND (not And)

Eine Negierung einer Konjunktion.
„Nur wenn x UND y NICHT wahr sind, dann…“

x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR (not Or)

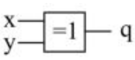
Eine Negierung einer Disjunktion.
„Nur wenn x ODER y NICHT wahr sind, dann…”



x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

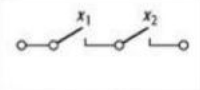
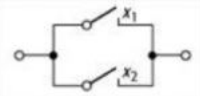
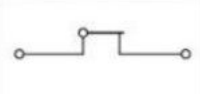
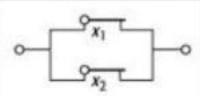
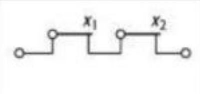
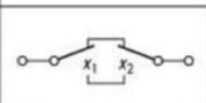
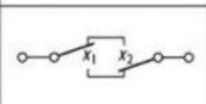
6. EXOR

Auch ausschließende Disjunktion genannt.
“Nur wenn x ODER y NICHT wahr sind,
sich aber nicht gleichen, dann…”



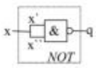
x	y	EXOR
0	0	0
0	1	1
1	0	1
1	1	0

Bauteilübersicht

Boolesche Funktion	Mathematische Darstellung	Kontaktanordnung	Wertetabelle	Symbol
Konjunktion (UND)	$x_1 \wedge x_2 = y$		$x_2 \backslash x_1$: Ø L Ø: Ø Ø L: Ø L	&
Disjunktion (ODER)	$x_1 \vee x_2 = y$		$x_2 \backslash x_1$: Ø L Ø: Ø L L: L L	1
Negation (NICHT)	$\bar{x} = y$		x: y Ø: L L: Ø	
Negiertes UND (NAND)	$\overline{x_1 \wedge x_2} = y$		$x_2 \backslash x_1$: Ø L Ø: L L L: L Ø	&
Negiertes ODER (NOR)	$\overline{x_1 \vee x_2} = y$		$x_2 \backslash x_1$: Ø L Ø: L Ø L: Ø Ø	1
Äquivalenz (Exklusiv-NOR)	$(\bar{x}_1 \wedge \bar{x}_2) \vee (x_1 \wedge x_2) = y$		$x_2 \backslash x_1$: Ø L Ø: L Ø L: Ø L	=
Antivalenz (Exklusiv-ODER)	$(\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2) = y$		$x_2 \backslash x_1$: Ø L Ø: Ø L L: L Ø	=1

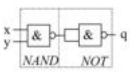
Realisierung von Verknüpfungen aus nur NAND-Gattern

Inverter



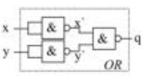
$q = \overline{x} = \overline{(x \wedge x)}$

AND-Verknüpfung



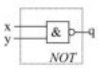
$q = x \wedge y = \overline{\overline{x \wedge y}} = \overline{\overline{(x \wedge y)} \wedge \overline{(x \wedge y)}}$

OR-Verknüpfung



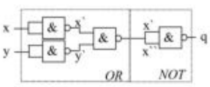
$q = x \vee y = \overline{\overline{x \vee y}} = \overline{\overline{x} \wedge \overline{y}} = \overline{\overline{(x \wedge x)} \wedge \overline{(y \wedge y)}}$

NAND-Verknüpfung



$q = \overline{x \wedge y}$

NOR-Verknüpfung



$q = \overline{x \vee y} = \overline{x} \wedge \overline{y} = \overline{\overline{\overline{x} \wedge \overline{y}}} = \overline{\overline{\overline{(x \wedge x)} \wedge \overline{(y \wedge y)}}}$

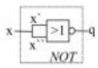
EXOR-Verknüpfung

…

$q = (x \wedge \overline{y}) \vee (\overline{x} \wedge y) = \overline{\overline{(x \wedge \overline{y}) \vee (\overline{x} \wedge y)}} = \overline{\overline{(x \wedge \overline{y})} \wedge \overline{(\overline{x} \wedge y)}}$
$= \overline{\overline{(x \wedge \overline{(y \wedge y)})} \wedge \overline{(\overline{(x \wedge x)} \wedge y)}}$

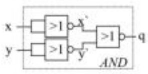
Realisierung von Verknüpfungen aus nur NOR-Gattern

Inverter



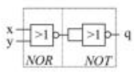
$q = \overline{x} = \overline{(x \wedge x)}$

AND-Verknüpfung



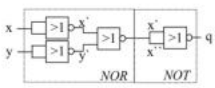
$q = x \wedge y = \overline{\overline{x \wedge y}} = \overline{\overline{x} \vee \overline{y}} = \overline{\overline{(x \vee x)} \vee \overline{(y \vee y)}}$

OR-Verknüpfung



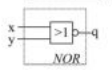
$q = x \vee y = \overline{\overline{x \vee y}} = \overline{\overline{(x \vee y)} \wedge \overline{(x \vee y)}}$

NAND-Verknüpfung



$q = \overline{x \wedge y} = \overline{x} \vee \overline{y} = \overline{\overline{\overline{x} \vee \overline{y}}} = \overline{\overline{\overline{(x \vee x)} \vee \overline{(y \vee y)}}}$

NOR-Verknüpfung



$q = \overline{x \vee y}$

EXOR-Verknüpfung

…

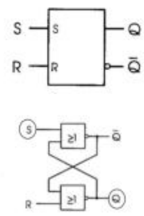
$q = (x \wedge \overline{y}) \vee (\overline{x} \wedge y) = \overline{\overline{\overline{\overline{(x \wedge \overline{y})}} \vee \overline{\overline{(\overline{x} \wedge y)}}}} = \overline{\overline{\overline{\overline{x} \vee \overline{\overline{y}}} \vee \overline{\overline{\overline{x}} \vee \overline{y}}}}$
$= \overline{\overline{\overline{(\overline{(x \vee x)} \vee y)} \vee \overline{(x \vee \overline{(y \vee y)})}}}$

Flip Flops

Flip Flops sind elektronische Schaltungen, die zwei stabile elektronische Zustände besitzen. Durch Eingangssignale kann dieser Zustand verändert werden. Typische Anwendung finden sie bei Schieberegistern, Speichern, Zählern und Frequenzteilern.

1. RS-Flip-Flop (NOR-Gatter)

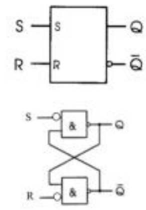
Der Eingabewert bleibt im Flip Flop erhalten. Ein Hoch bei beiden Eingängen sollte vermieden werden, da der Zustand nicht bestimmbar ist.



S	R	Q	$\overline{Q}$
0	0	Q_{-1}	$\overline{Q_{-1}}$
0	1	0	1
1	0	1	0
1	1	?	?

2. RS-Flip-Flop (NAND-Gatter)

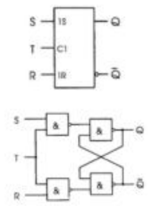
In der Industrie werden häufig NAND-Gatter verwendet. Also wird der RS-Flip Flop meist durch NAND-Gatter dargestellt. An der Funktion ändert sich nichts.



S	R	Q	$\overline{Q}$
0	0	Q_{-1}	$\overline{Q_{-1}}$
0	1	0	1
1	0	1	0
1	1	?	?

3. RS-Flip-Flop mit Takteingang

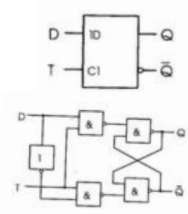
Ist eine Erweiterung zum R-S Flip-Flop. Nur wenn T eins ist, kann ein Input übernommen werden. Wenn T null ist wird der letzte Zustand gespeichert.



T	S	R	Q	$\overline{Q}$	
1	0	0	Q_{-1}	$\overline{Q_{-1}}$	Gleich mit R-S Flip-Flop
1	0	1	0	1	
1	1	0	1	0	
1	1	1	?	?	
0	X	X	Q_{-1}	$\overline{Q_{-1}}$	Speichert Zustand

4. D-Flip-Flop

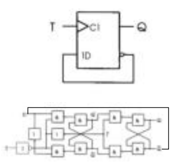
Vom Aufbau ähnelt das D-Flip Flop dem RS-Flip Flop. Jedoch wurde hier durch nur einen Input der Fehler des nicht definierten Zustands behoben.

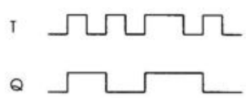


T	D	Q	$\overline{Q}$	
0	X	Q_{-1}	$\overline{Q_{-1}}$	Speichert Zustand
1	0	0	1	Transparent
1	1	1	0	

5. T-Flip-Flop

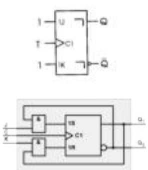
Toggle-Flip-Flops erhält man, wenn man den invertierten Ausgang eines Taktgesteuerten D-Flip-Flops wieder als Dateneingang benutzt. Diesen wechselnden Zustand nennt man Frequenzteiler.





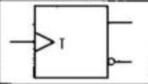
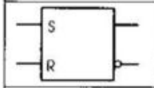
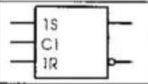
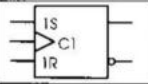
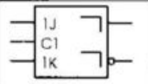
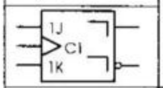
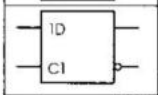
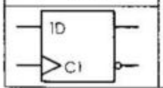
6. J-K-Flip-Flop

Der J-K-Flip-Flop ist eine Erweiterung der R-S-Flip-Flops. Durch Rückkoppelung der komplementären Ausgänge gibt es keinen undefinierten Zustand. Sind beide Inputs 1, toggled dieser wie ein T-Flip-Flop. In diesem Zustand teilt dieser den Takt durch zwei.

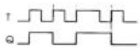


J	K	Q	$\overline{Q}$	
0	0	n	n	Speichert Zustand
0	1	0	1	Setzen von Werten
1	0	1	0	
1	1	X	X	Wechselbetrieb

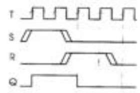
- **Ungetaktete Flip-Flops**: Ihr Zustand wird nur von den Setz-/Rücksetz-Eingängen gesteuert.
- **Getaktete Flip-Flops:** Der Zeitpunkt der Informationsübernahme wird durch ein Taktsignal vorgegeben.
- **Zustandsgesteuerte Flip-Flops:** Die Informationsübernahme wird durch einen Pegel des Steuersignals veranlasst.
- **Flankengesteuerte flip-Flops:** Die Informationsübernahme wird durch einen Zustandswechsel veranlasst.

Schaltsymbol	Flip-Flop	Steuerung
	T-Flip-Flop	getaktet, einflankengesteuert
	RS-Flip-Flop	nicht getaktet, zustandsgesteuert
		getaktet, einzustandsgesteuert
		getaktet, einflankengesteuert
	JK-Flip-Flop	getaktet, zweizustandsgesteuert 1.)
		getaktet, zweiflankengesteuert 1.)
	D-Flip-Flop	getaktet, einzustandsgesteuert
		getaktet, einflankengesteuert

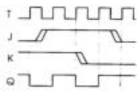
Das **T-Flip-Flop** teilt die Impulsanzahl des Taktes durch zwei. Bei annähernd konstanter Taktfrequenz spricht man von einem **Frequenzteiler**.



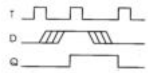
Der **RS-Flip-Flop** wird durch die positive Flanke des Taktsignals gesetzt, wenn der *Set*-Eingang auf H liegt. Wiederholtes Setzen ändert den Zustand nicht. Ist das *Reset*-Signal zum Zeitpunkt der positiven Taktflanke H, wird das Flip-Flop rückgesetzt. $R = S = 1$ führt zum Zeitpunkt der Taktflanke zu einem irregulären Zustand. Ansonsten ist die Kombination zulässig.

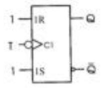


Beim taktflankengesteuerten **JK-Flip-Flop** hängen die Ausgangssignale von den asynchronen Vorbereitungseingängen J und K ab. Bei der Kombination $J = K = 1$ arbeitet das Flip-Flop als T-Flip-Flop, bei der Kombination $J = K = 0$ speichert es den vorherigen Zustand.

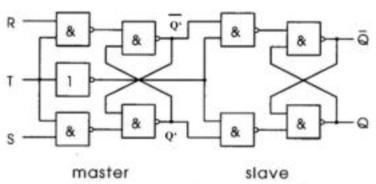


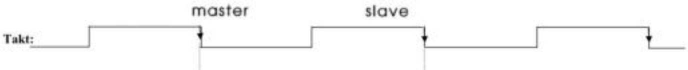
Das **D-Flip-Flop** übernimmt mit der positiven Taktflanke den Wert am Dateneingang. Die rampenartigen Flanken des D-Signals deuten an, dass der Zeitpunkt des Pegelwechsels innerhalb des gezeichneten Intervalls gleichgültig für die Funktion des Schaltkreises ist.



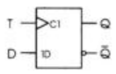


Flip-Flops
Flankengesteuertes RS-Flip-Flop (negativ flankengest.)



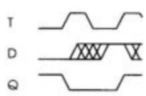


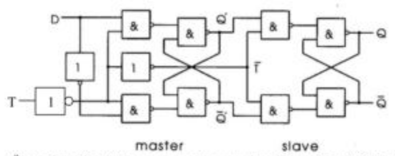
Master:	Mastereingang bis Q' kann sich jederzeit durch R,S ändern	Mastereingang gesperrt (durch T)	Mastereingang bis Q' kann sich jederzeit durch R,S ändern	Mastereingang gesperrt (durch T)	…
Slave:	Slaveeingang gesperrt (durch T)	Slave freigeschaltet und könnte sich jederzeit ändern (bleibt aber, da Master sich nicht mehr ändert)	Slaveeingang gesperrt (durch T)	Slave freigeschaltet und könnte sich jederzeit ändern (bleibt aber, da Master sich nicht mehr ändert)	…
Ausgang:	… sich nicht.	Akt. Mastereingangswert wird fixiert. Slave wird freigeschaltet und übernimmt Masterwert Q'	Ausgangswert ändert sich nicht	Akt. Mastereingangswert wird fixiert. Slave wird freigeschaltet und übernimmt Masterwert Q'	Ausgangswert ändert sich nicht



Flankengesteuertes D-Flip-Flop (positiv flankengesteuert)

Die Transparenz des D-Flip-Flops geht verloren, wenn man zwei hintereinander schaltet.
Die beiden Flip-Flops werden mit komplementären Taktsignal angesteuert.





Übernahme des Wertes vom master zum slave bei positiver Taktflanke.

Diese Anordnung bezeichnet man als master-slave-Flip-Flop.

Der Q-Ausgang des masters folgt dem D-Signal, solange T = 0 anliegt. Das slave-Flip-Flop bleibt solange verriegelt. Geht das Taktsignal auf 1, so verriegelt das master-Flip-Flop, und das folgenden slave-Flip-Flop übernimmt den logischen Zustand des Q-Ausgangs des masters.

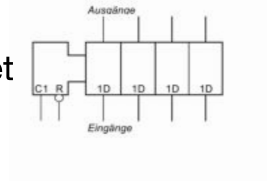
Ein Zustand wird also immer nur bei einer **Flanke** übernommen, man spricht von einem dynamischen Flip-Flop.

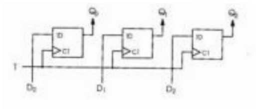
Register/Zähler

Flip Flops können hintereinandergeschaltet mehrere Aufgaben erfüllen. Sie können als Speicher verwendet werden oder zu Zählern zusammengeschlossen werden.

1. Speichermodul

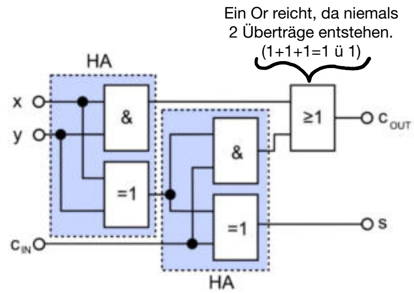
Eine Parallele Anordnung von D-Flip Flops kann zur Zwischenspeicherung von Signalzuständen verwendet werden.

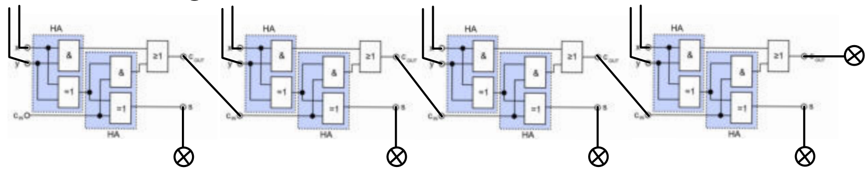




2. Halbaddierer/Addierer

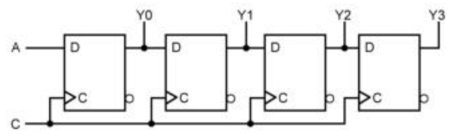
Ein Halbaddierer (HA) addiert zwei 1bit Zahlen zusammen. Als Outputs gibt es ein Ergebnis- und ein “Übertragsbit”. Schaltet man zwei Halbaddierer hintereinander, so wird auch das Übertragsbit berücksichtigt, und es wird ein Volladdierer.





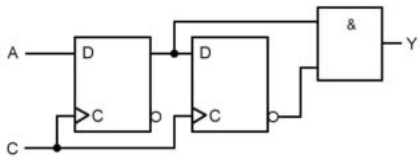
3. Schieberegister

Bei einem Schieberegister sind mehrere Flipflops hintereinander geschaltet. Der Takt schiebt den gespeicherten Zustand dann einen Flipflops weiter. Dadurch lässt sich ein gleichbleibenden sequenzielles Verhalten realisieren.



4. Synchrones Monoflop

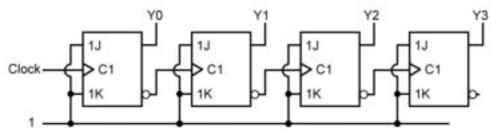
Bei einem Schieberegister sind mehrere Flipflops hintereinander geschaltet. Der Takt schiebt den gespeicherten Zustand dann einen Flipflops weiter. Dadurch lässt sich ein gleichbleibenden sequenzielles Verhalten realisieren.



Selbstständiges Rücksetzen

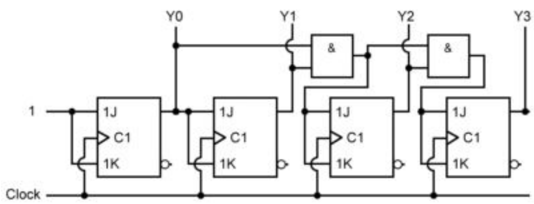
5. Asynchroner Zähler

Der asynchrone Zähler ist inkonsistent in den Ausgangssignalen. Längere Verzögerungen oder fehlerhafte Zustände sind unvermeidbar, dafür einfacher Aufbau.



6. Synchroner Zähler

Der synchrone Zähler ist schnell und bietet zuverlässige Ausgangssignale. Deswegen wird dieser deutlich häufiger verwendet.



Zahlensysteme

Zahlensysteme sind vor allem in der Elektronik ein essenzieller Bestandteil. Unterschiedliche Darstellungen von Zahlen bieten jeweils andere Vorteile.

1. Dezimalsystem

Das Dezimalsystem gehört zu den meist benutzen Zahlensystemen. Wahrscheinlich hat es sich durch unsere Anatomie mit 10 Fingern etabliert.
Elemente: 1,2,3,4,5,6,7,8,9,0
Rechenregeln: Hoffentlich bekannt.

2. Binärsystem

Das Binärsystem wird häufig in der Elektronik eingesetzt, da man es auch durch einen hohen und niedrigen Strompegel darstellen kann.
Elemente: 1,0
Rechenregeln:

2.1: Umformen

Das Horner-Schema

125	
125 : 2 = 62	Rest: 1
62 : 2 = 31	Rest: 0
31 : 2 = 15	Rest: 1
15 : 2 = 7	Rest: 1
7 : 2 = 3	Rest: 1
3 : 2 = 1	Rest: 1
1 : 2 = 0	Rest: 1

⇒ 1111101

⟺

1111101	
0 · 2 + 1	= 1
1 · 2 + 1	= 3
3 · 2 + 1	= 7
7 · 2 + 1	= 15
15 · 2 + 1	= 31
31 · 2 + 0	= 62
62 · 2 + 1	= 125

⇒ 125

Die Umrechnungstabelle

2^5	2^4	2^3	2^2	2^1	2^0
32	16	8	4	2	1
1	1	1	1	0	1

⇒ 32+16+8+4+1 = 61

2.2: Rechnungen

```
  101010 | 42
+₁1100₁10 | 50
 1011100 | 92
```

```
1000 · 1010
   1000
    0000
     1000
+     0000
  1010000
```

```
11100 : 100 = 111
100
0110
 100
 0100
```

2.3: Komplemente

```
7: 0 1 1 1
6: 0 1 1 0
5: 0 1 0 1
4: 0 1 0 0
3: 0 0 1 1
2: 0 0 1 0
1: 0 0 0 1
0: 0 0 0 0
-1: 1 1 1 1
-2: 1 1 1 0
-3: 1 1 0 1
-4: 1 1 0 0
-5: 1 0 1 1
-6: 1 0 1 0
-7: 1 0 0 1
-8: 1 0 0 0
```

0010 ⇒ 1101 + 1 ⇒ 1110 (Einerkomplement)

1011 ⇒ 0100 + 1 ⇒ 0101

Bei der zweierkomplementdarstellung wird das Most significant Bit (ganz links) als Vorzeichen verwendet. Mit dieser kann man von -2^(n-1) bis 2^(n-1) -1 alle Zahlen darstellen. Eine Umwandlung ins Dezimalsystem ist bei positiven Komplementen immer möglich, bei negativen musst man wieder zurückrechnen und dann ein Minus hinzufügen.

+0
-0

Addition Einerkomplement

3. Oktalsystem

Das Oktalsystem war damals das vorherrschende Zahlensystem für Computer, da es sich leicht ins Binärsystem umwandeln lässt, denn 0=000 und 7=111.
Elemente: 1,2,3,4,5,6,7,0
Rechenregeln: Umrechnungstabelle

8^5	8^4	8^3	8^2	8^1	8^0
32768	4096	512	64	8	1
0	0	0	2	1	5

⇒ 128+8+1 = 137

4. Hexadezimalsystem

Das Hexadezimalsystem löst heutzutage das Oktalsystem größtenteils ab. Es ist kürzer und damit besser zu lesen und zu schreiben. Außerdem sind heutige Datenwerte fast nur noch in 16,32 oder 64Bit dargestellt.
Elemente: 1,2,3,4,5,6,7,8,9,0,A,B,C,D,E,F
Rechenregeln: Umrechnungstabelle

Normalformen

Eine Schaltung, die durch boolsche Funktionen definiert wurde, hat nahezu unendlich viele verschiedene Realisierungen. Um dies in einem standardisierten eindeutigen Therm definieren zu können, existieren die Normalformen.

1. Minterm/Maxterm

Ein Minterm ist die konjunktive (Und) Verknüpfung aller Eingangsvariablen.
Ein Maxterm ist die disjunktive (oder) Verknüpfung aller Eingangsvariablen.

A	B	C	f(A, B,C)	Minterm
0	0	1	1	→ (¬A ∧ ¬B ∧ C)
1	0	0	1	→ (A ∧ ¬B ∧ ¬C)

A	B	C	f(A, B,C)	Maxterm
0	0	1	0	→ (A ∨ B ∨ ¬C)
1	0	0	0	→ (¬A ∨ B ∨ C)

2. Kanonische Disjunktive-/Konjunktive Normalform

Die kanonischen Normalformen verknüpfen dann diese Minterme/Maxterme dann disjunktiv/konjunktiv und stellen so dann einen eindeutigen Therm der Schaltung dar.
Hier müssen alle Eingangsvariablen enthalten sein (keine Vereinfachung)

f(A,B,C) = (¬A ∧ B ∧ C) ∨ (¬A ∧ B ∧ ¬C)

f(A,B,C) = (¬A ∨ B ∨ C) ∧ (A ∨ B ∨ ¬C)

3. Disjunktive-/Konjunktive Normalform

In den Normalformen werden abschließend die kanonischen Normalformen zusammengefasst. Wenn mehr wahre Ergebnisse (1) rauskommen, verwendet man die Disjunktive, andernfalls die Konjunktive Normalform.

a	b	c	f	Normalform (kanonisch)	Vereinfacht
0	0	0	0		
0	0	1	1	→ (¬a ∧ ¬b ∧ c) ∨	
0	1	0	0		(¬a ∧ c)
0	1	1	1	→ (¬a ∧ b ∧ c) ∨	∨
1	0	0	1	→ (a ∧ ¬b ∧ ¬c) ∨	(a ∧ ¬c)
1	0	1	0		
1	1	0	1	→ (a ∧ b ∧ ¬c)	
1	1	1	0		

f(A,B,C) = (¬a ∧ c) ∨ (a ∧ ¬c)

a	b	c	f	Normalform	Vereinfacht
0	0	0	1		
0	0	1	0	→ (a ∨ b ∨ ¬c) ∧	
0	1	0	0	→ (a ∨ ¬b ∨ c) ∧	
0	1	1	1		
1	0	0	0	→ (¬a ∨ b ∨ c) ∧	
1	0	1	1		
1	1	0	0	→ (¬a ∨ ¬b ∨ c)	
1	1	1	1		

f(a,b,c) = (a ∨ b ∨ ¬c) ∧ (a ∨ ¬b ∨ c) ∧ (¬a ∨ b ∨ c) ∧ (¬a ∨ ¬b ∨ c)

4. KV-Diagramm

Das KV-Diagramm dient zur übersichtlichen Darstellung und Vereinfachung boolscher Funktionen. Mit diesem lässt sich jede beliebige Disjunktive Normalform in einen minimalen Ausdruck umwandeln. Jeder einzelne Ausgabewert benötigt ein KV-Diagramm.

	¬B	B	B	¬B	
¬A	1	1	0	1	¬D
A	1	1	0	0	¬D
A	0	0	0	0	D
¬A	1	0	0	1	D
	¬C	¬C	C	C	

⇒ f(A,B,C,D) = (¬A ∧ ¬B) ∨ (¬C ∧ ¬D)

	¬B	B	B	¬B	
¬A	1	1	1	1	¬D
A	1	1	1	1	¬D
A	1	0	0	1	D
¬A	1	0	0	1	D
	¬C	¬C	C	C	

⇒ f(A,B,C,D) = (¬B) ∨ (¬C)