Hamming-Distanz

$F = D_{min} - 1$

F: Anzahl der Bitfehler
D_{min}: Geringste Distanz

Bsp.:

x: 00110

y: 00101

z: 01110

⇓

Hamming-Distanz
(Anzahl der unterschiedlichen Bits)

H-D. von x & y = 2

H-D. von x & z = 1 ⇒ 1 = kleinste = Hamming-Distanz des Codes ist 1.

H-D von z & y = 3

⇒ $F = 1 - 1 = 0$ ⇒ Fehler erkennbar

Eine andere wichtige Eigenschaft von Fehler erkennenden bzw. korrigierenden Codes ist ihre **Redundanz**. Sie gibt das Verhältnis von Gesamtbits zu Nutzbits an:

R = (n + p) / n

R = Redundanz
n = Anzahl der Nutzbits
p = Anzahl der Prüfbits.

Beispiel einer mittels **Quer**parität (even) gesicherten Textdatei (8-Bit-ASCII):

Text: **Pause!**

Datei: (Z = Zeichen, hex = Hexadezimal-Code, D_n = Datenbits, P = Parity-Bit)

Z	hex	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	P
P	50	0	1	0	1	0	0	0	0	**0**
a	61	0	1	1	0	0	0	0	1	**1**
u	75	0	1	1	1	0	1	0	1	**1**
s	73	0	1	1	1	0	0	1	1	**1**
e	65	0	1	1	0	0	1	0	1	**0**
!	21	0	0	1	0	0	0	0	1	**0**

Die Redundanz beträgt

R = (8+1)/8 = 1.125

Der Code-Umfang steigt um 12.5%!

Beispiel einer mittels **Längs**parität (even) gesicherten Textdatei (8-Bit-ASCII):

Text: **Pause!**

Datei: (Z = Zeichen, hex = Hexadezimal-Code, D_n = Datenbits, P = Parity-Bit)

Z	hex	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
P	50	0	1	0	1	0	0	0	0
a	61	0	1	1	0	0	0	0	1
u	75	0	1	1	1	0	1	0	1
s	73	0	1	1	1	0	0	1	1
e	65	0	1	1	0	0	1	0	1
!	21	0	0	1	0	0	0	0	1
P		**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**

Die Redundanz beträgt

R = (6+1)/6 = 1.167

Der Code-Umfang steigt um 16.7%!

Text: **Pause!**

Datei: (Z = Zeichen, hex = Hexadezimal-Code, D_n = Datenbits, P = Parity-Bit)

Z	hex	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	P
P	50	0	1	0	1	0	0	0	0	**0**
a	61	0	1	1	0	0	0	0	1	**1**
u	75	0	1	1	1	0	1	0	1	**1**
s	73	0	1	1	1	0	0	1	1	**1**
e	65	0	1	1	0	0	1	0	1	**0**
!	21	0	0	1	0	0	0	0	1	**0**
P		**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**

Die Redundanz beträgt

R = (48+15)/48 = 1.3125

Der Code-Umfang steigt um 31.25%!

Mindest-Hamming-Distanz $D_{min} = 3$

$\Rightarrow F = \frac{(D_{min} - 1)}{2}$ (Fehler Korriegierbar)

Bsp.:

a = 0000000000
b = 0000011111 ⇒ H-D = d = 5 ; d = 10
c = 1111100000
d = 1111111111

Wie viele Korriegierbar

$P = \frac{5-1}{2} = 2$ (Doppelbitfehler)

Hamming-Code für 16 Nutzbits

Potenz zu 2 → steht ein Prüfbit p_n

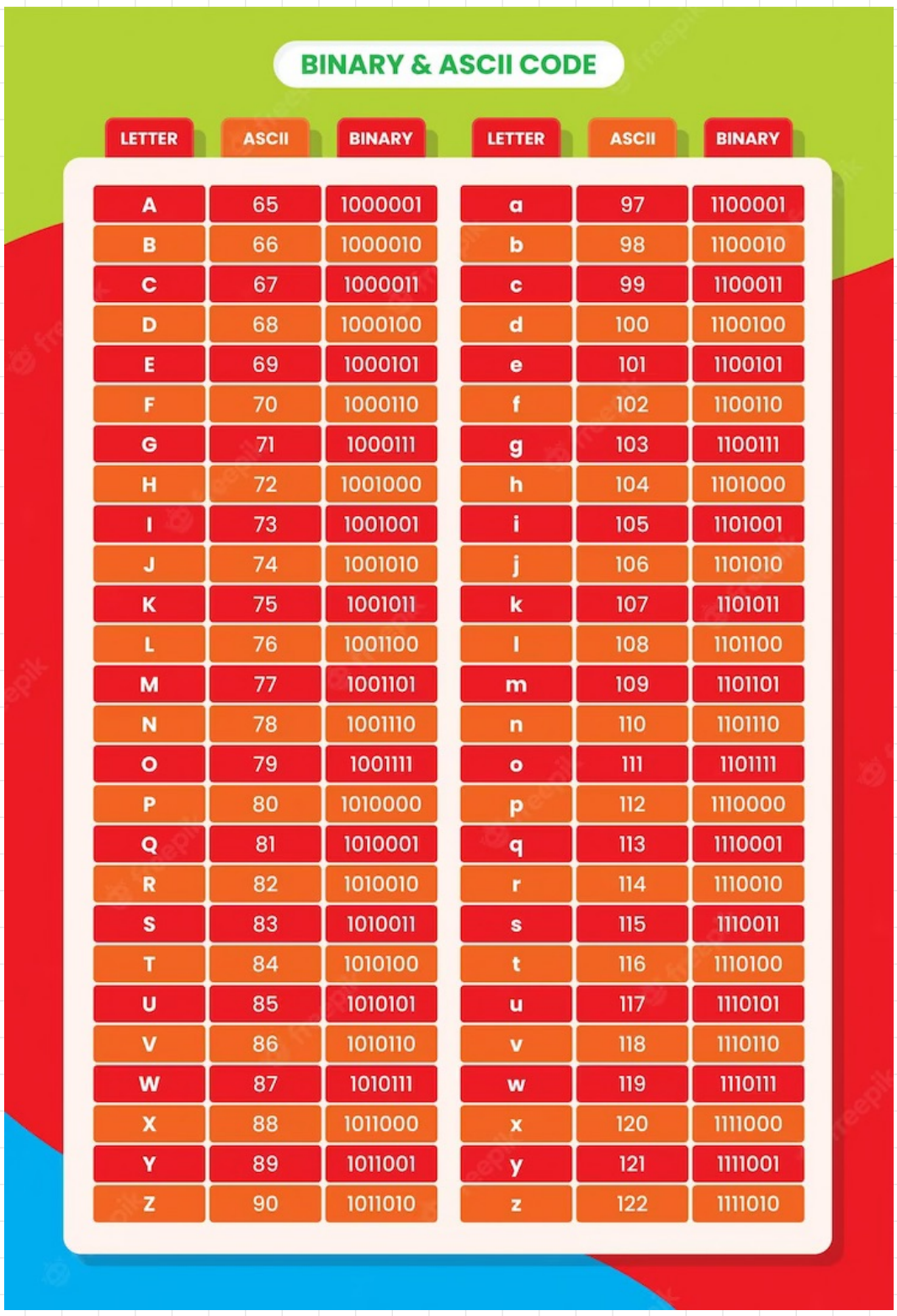
alle anderen Positionen Nutzbits d_n

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	p_0	p_1	d_0	p_2	d_1	d_2	d_3	p_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	p_4	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}
p_0	x		+		+		+		+		+		+		+		+		+		+
p_1		x	+			+	+			+	+			+	+			+	+		
p_2				x	+	+	+					+	+	+	+					+	+
p_3								x	+	+	+	+	+	+	+						
p_4																x	+	+	+	+	+

Beispiel 1: Datum ok	1	0	0	1	1	1	0	1	0	1	1	1	0	1	1	0	1	0	1	0	0
Parity ok?	✓	✓		✓				✓								✓					
Beispiel 2: Datum <u>nicht</u> ok	1	0	0	1	1	1	0	1	0	1	1	1	0	0	1	0	1	0	1	0	0
Parity ok?	✓	*f*		*f*				*f*						*f*		✓					

x = Parity-Bit
\+ = geht in die Berechnung des Parity-Bits ein
= geht in die Berechnung des Parity-Bits <u>nicht</u> ein

$$(2+4+8) = 14\ (d_9)$$



Zweier Komplement

Minuend Subtrahend

$01110111_2 - 11010100_2$

```
 11010100
 --------
 00101011
+00000001
 --------
 00101100
+01110111
 --------
-01011011
```

1.) Umkehren von Subtrahend

2.) Addieren mit 1

3) Addiere mit Minuend

4) Umkehren aufgrund 0 (Wäre 1 einfach – (kein 4.) und 5.))

5) Addiere mit 1

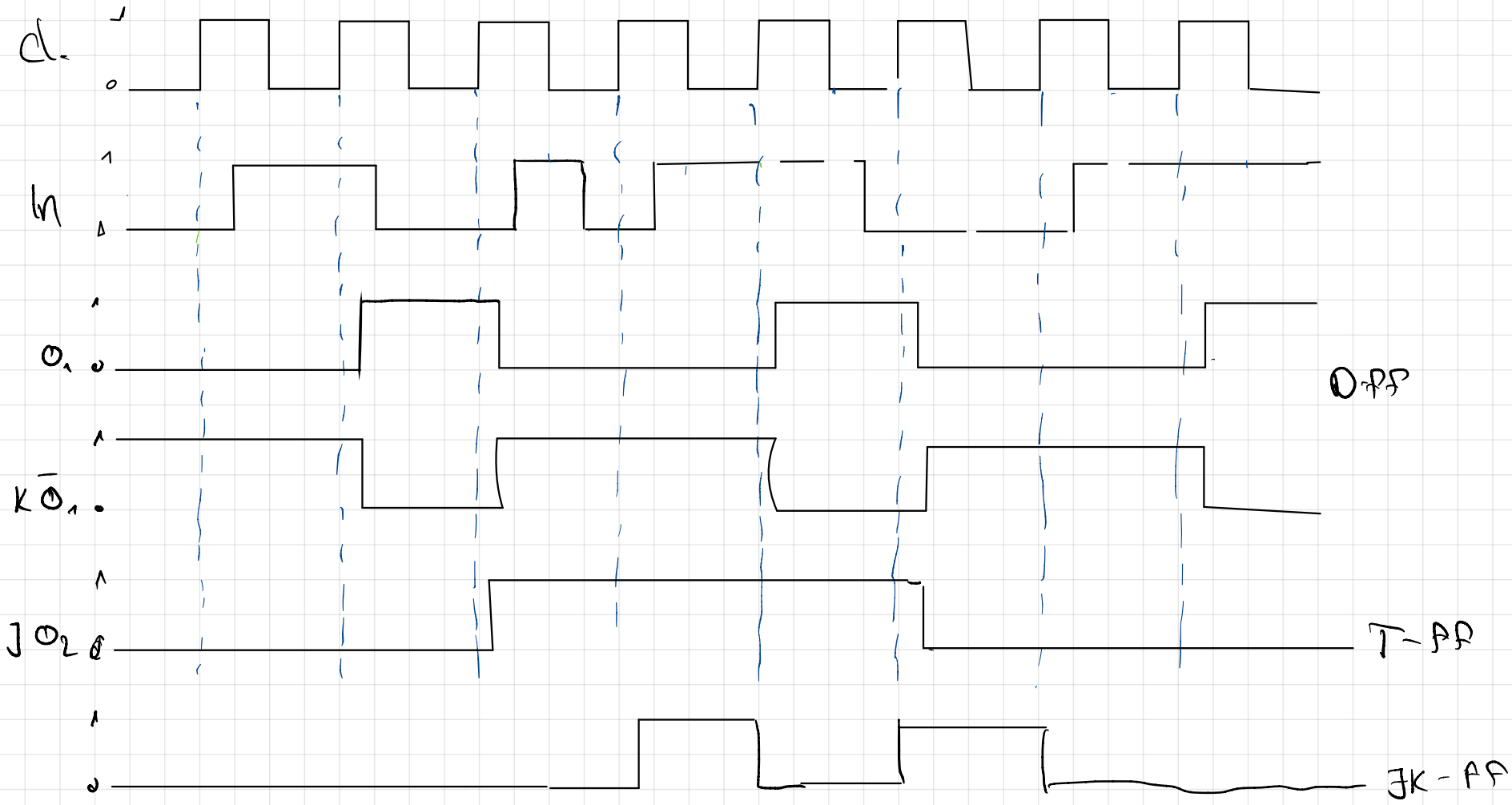
```
 01011100
+00000001
 --------
-01011101
```

Multiplikation von Dual

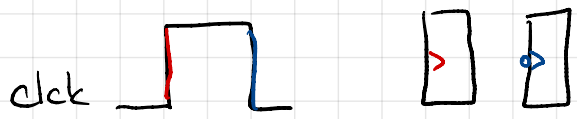
```
 1111 · 1001
 -----------
+     1111
+    0000
+   0000
+  1111
 -----------
  10000111
```

Division von Dual

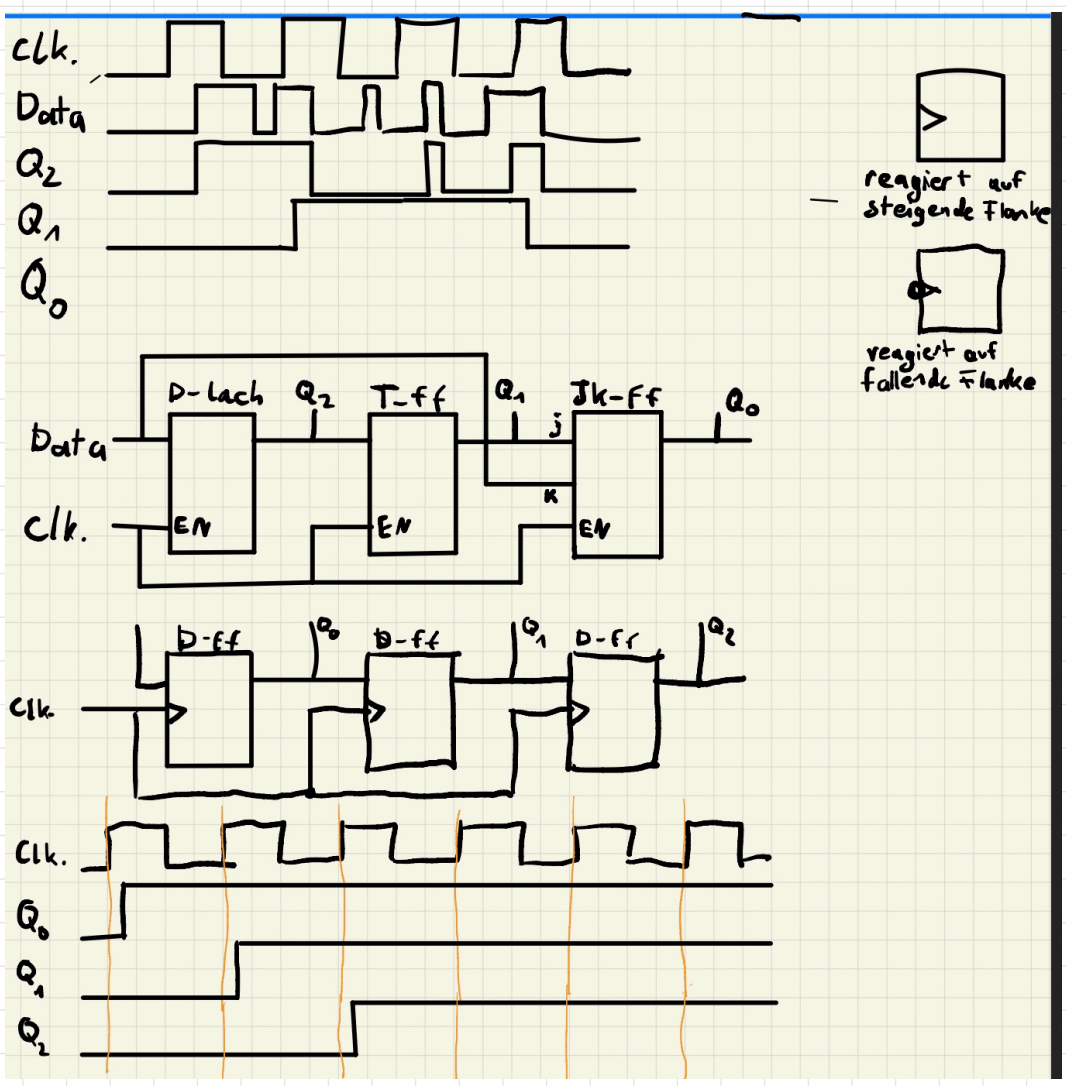
```
 11101 : 101 = 00101,11001
 1
 11
 111
-101
 ----
 010
  100
  1001
-  101
   ----
   0100
    1000
   -  101
      ----
      00110
       101
      ------
      001000
        101
      ------
       0011
```

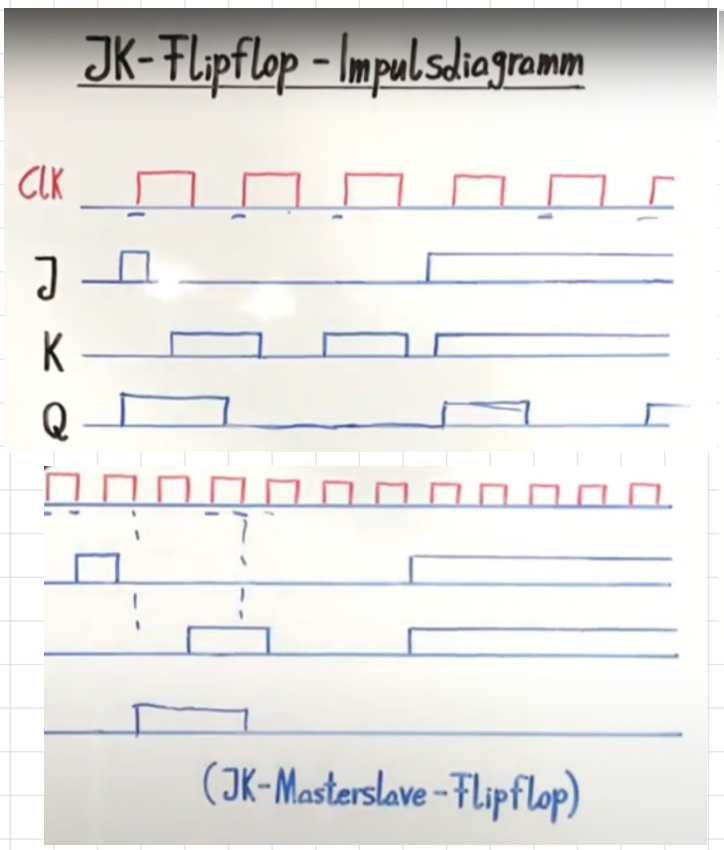


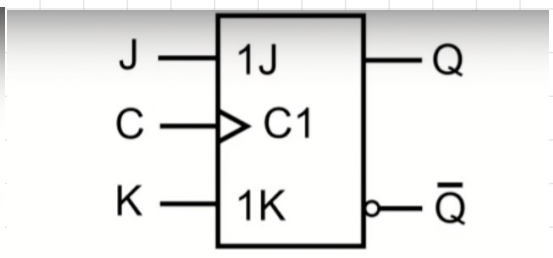
FF → Flipflop Veränderung wenn Clock auf eins springt



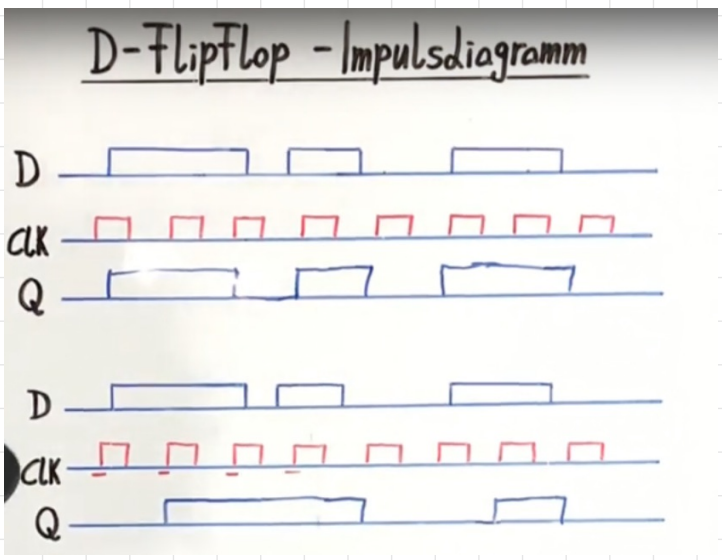
latch: Veränderung wenn Clck auf 1 ist



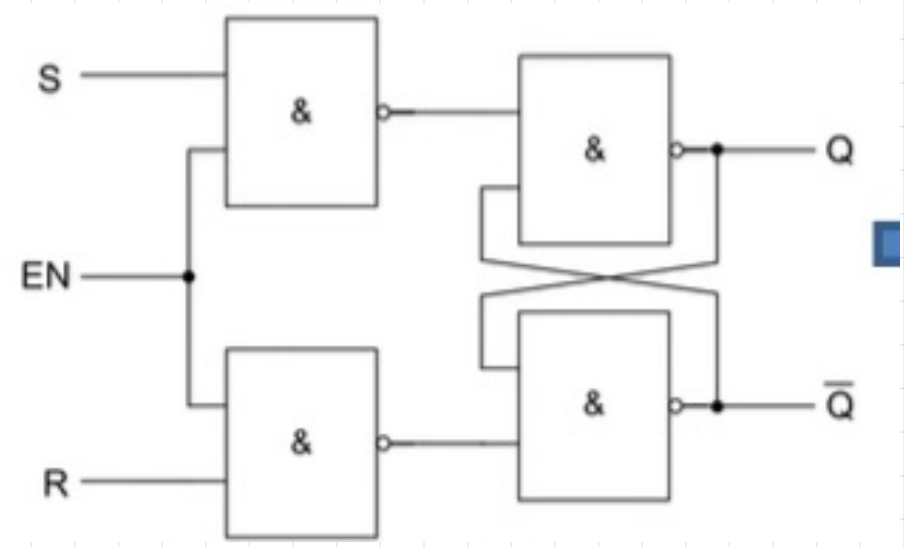


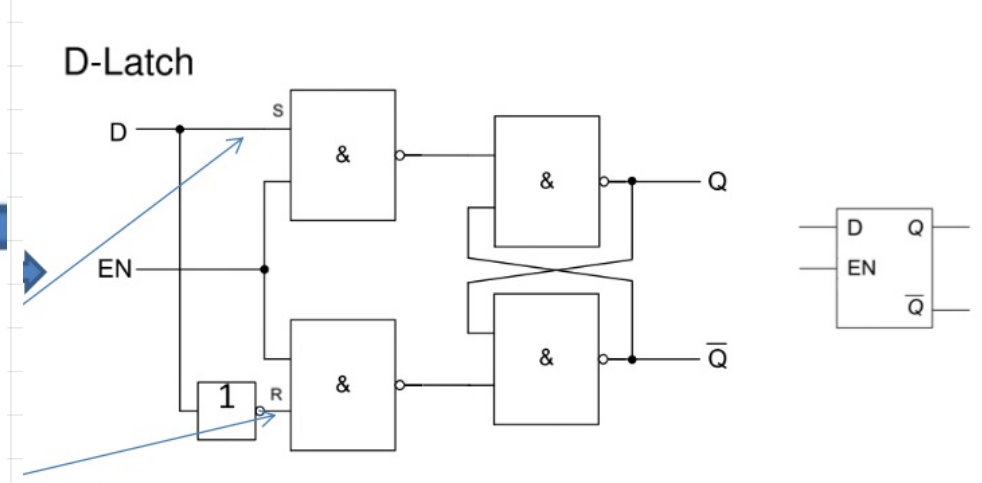


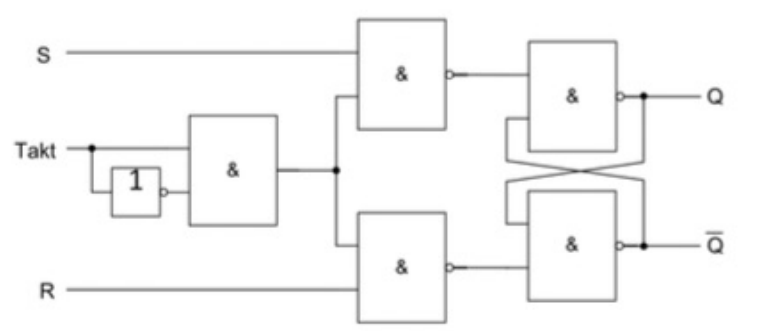
J	K	Q	$Q_{inv.}$
0	0	Q_{Alt}	$Q_{alt,inv}$
0	1	0	0
1	0	1	0
1	1	$Q_{alt,inv}$	Q_{alt}

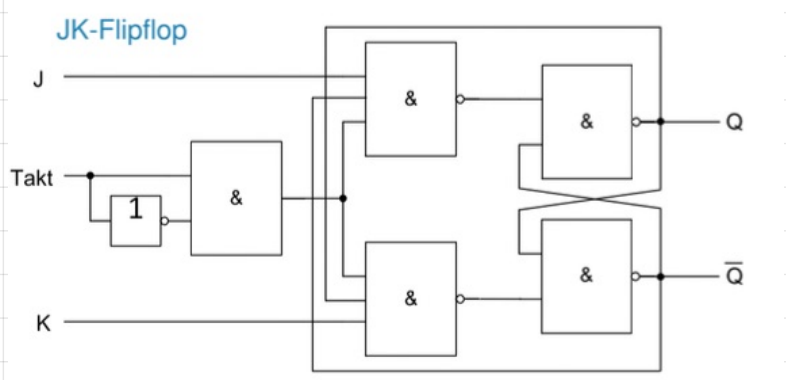


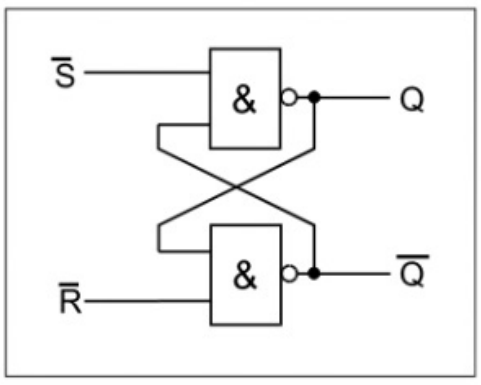
11





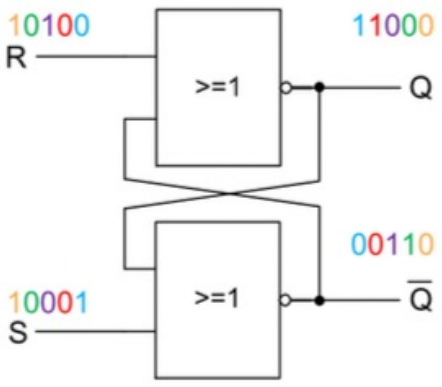






Für „active low“ SR-Latch (auch als NAND-Gatter Latch bekannt):

- $\overline{R}$ = Low (und $\overline{S}$ = High) → RESET-Zustand
- $\overline{S}$ = Low (und $\overline{R}$ = High) → SET-Zustand
- beide Eingänge High → keine Änderung
- beide Eingänge Low → Q und $\overline{Q}$ High (ungültig)!

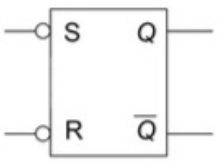


Für „active high“ SR-Latch (auch als NOR-Gatter Latch bekannt):

- R = High (und S = Low) → RESET-Zustand
- S = High (und R = Low) → SET-Zustand
- beide Eingänge Low → keine Änderung
- beide Eingänge High → Q und $\overline{Q}$ Low (ungültig)!

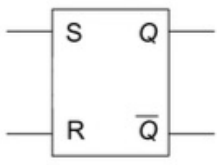
Wahrheitstabelle für SR-Latch mit „active low“ Eingang

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
1	1	NC	NC	No Change, Latch bleibt im aktuellen Zustand
0	1	1	0	Latch SET
1	0	0	1	Latch RESET
0	0	1	1	ungültige Bedingung

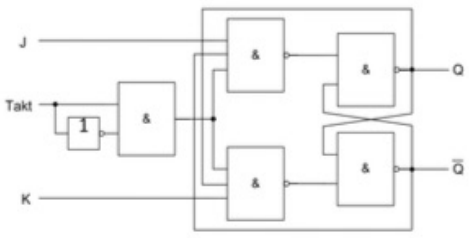


Wahrheitstabelle für SR-Latch mit „active high“ Eingang

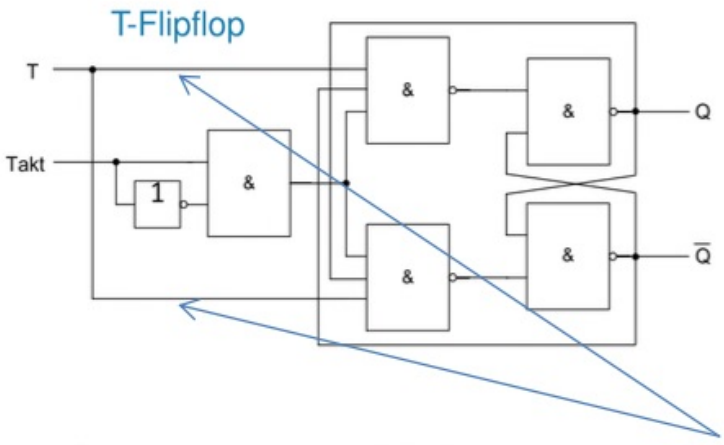
S	R	Q	$\overline{Q}$	
0	0	NC	NC	No Change, Latch bleibt im aktuellen Zustand
1	0	1	0	Latch SET
0	1	0	1	Latch RESET
1	1	0	0	ungültige Bedingung

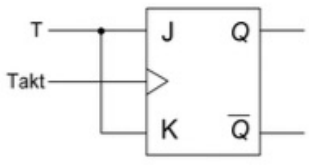


JK-Flipflop



T-Flipflop



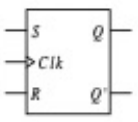
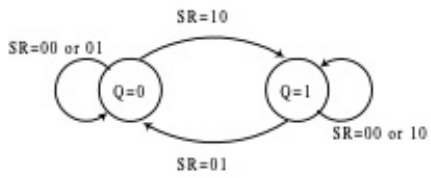
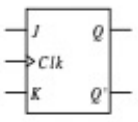
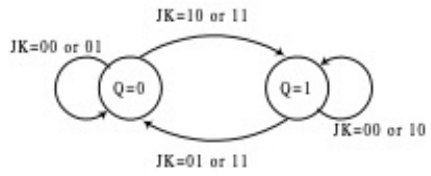
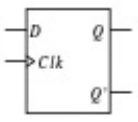
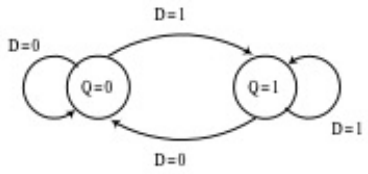
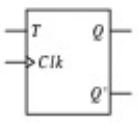
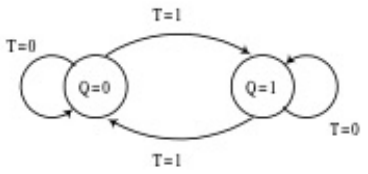


„Single-Input“-Version des JK-Flipflops, gebildet durch Verbinden beider Eingänge

Q	T	$Q(t+1)$
0	0	0
0	1	1
1	0	1
1	1	0

Charakteristische Tabelle

T	CLK	$Q(t+1)$	
0	↑	$Q(t)$	keine Änd.
1	↑	$\overline{Q(t)}$	Toggle

Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table
SR 	S R Q Q_{next} 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 × 1 1 1 ×	 $Q_{next} = S + R'Q$ $SR = 0$	Q Q_{next} S R 0 0 0 × 0 1 1 0 1 0 0 1 1 1 × 0
JK 	J K Q Q_{next} 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0	 $Q_{next} = J'K'Q + JK' + JKQ'$ $= J'K'Q + JK'Q + JK'Q' + JKQ'$ $= K'Q(J'+J) + JQ'(K'+K)$ $= K'Q + JQ'$	Q Q_{next} J K 0 0 0 × 0 1 1 × 1 0 × 1 1 1 × 0
D 	D Q Q_{next} 0 × 0 1 × 1	 $Q_{next} = D$	Q Q_{next} D 0 0 0 0 1 1 1 0 0 1 1 1
T 	T Q Q_{next} 0 0 0 0 1 1 1 0 1 1 1 0	 $Q_{next} = TQ' + T'Q = T \oplus Q$	Q Q_{next} T 0 0 0 0 1 1 1 0 1 1 1 0