Boolesche Funktionen

a, b, c, d, e seien boolesche Variable, denen vermöge der Abbildungsvorschriften

(a, b, c) → f(a, b, c) bzw. (a, b) → f(a, b)

der boolsche Funktionswert f(a, b, c) bzw. f(a,b) zugeordnet wird.

Mit â oder NOT a bezeichnen wir die Negation von a.

1. Negation

a	â
0	1
1	0

2. AND (Konjunktion)

(a, b) → a AND b (Vereinbarung: a AND b = a · b = ab)

a	b	a · b
0	0	0
0	1	0
1	0	0
1	1	1

3. OR (Disjunktion)

(a, b) → a OR b (Vereinbarung: a OR b = a + b)

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

4. XOR (eXclusive OR)

(a, e) → a XOR e (Vereinbarung: a XOR e = a ⊕ e)

a	e	a ⊕ e	
0	0	0	
0	1	1	←
1	0	1	←
1	1	0	

Disjunktion der Konjunktionen:

a XOR e = 0 + â · e + a · ê + 0 = â · e + a · ê

5. Gegeben ist die Zuordnung (a, b, c) → f(a, b, c) mittels folgender Wertetabelle; ermittle einen möglichst einfachen Funktionsterm:

a	b	c	f(a, b, c)	
0	0	0	0	
0	0	1	0	
0	1	0	1	←
0	1	1	0	
1	0	0	1	←
1	0	1	0	
1	1	0	1	←
1	1	1	0	

Disjunktion der Konjunktionen:

f(a, b, c) = NOTa · b · NOTc + a · NOTb · NOTc + a · b · NOTc
= [NOTa · b + a · NOTb + a · b] NOTc
= [NOTa · b + a · (NOTb + b)] NOTc
= [NOTa · b + a · 1] · NOTc
= [NOTa · b + a] · NOTc
= [a + NOTa · b] · NOTc
= [(a + NOTa) · (a + b)] · NOTc
= [1 · (a+b)] · NOTc
= (a+b) · NOTc

Hinweis:
a(b+c) = ab + ac
a + bc = (a+b) · (a+c)

Theoreme zum Rechnen mit Booleschen Variablen

Voraussetzung: a, e, u seien Boolesche Variable, mit â, ê, û werden die Negationen von a, e, u bezeichnet.

Kommutativgesetz

(1) $a \cdot e = e \cdot a$ (1)' $a + e = e + a$

Assoziativgesetz

(2) $a \cdot (e \cdot u) = (a \cdot e) \cdot u$ (2)' $a + (e + u) = (a + e) + u$

Distributivgesetz

(3) $a \cdot (e + u) = a \cdot e + a \cdot u$ (3)' $a + e \cdot u = (a + e) \cdot (a + u)$

Absorptionsgesetz

(4) $a \cdot (a + e) = a$ (4)' $a + a \cdot e = a$

Tautologie

(5) $a \cdot a = a$ (5)' $a + a = a$

Gesetz über die Negation

(6) $a \cdot â = 0$ (6)' $a + â = 1$

Doppelte Negation

(7) NOT (NOT a) = a

De Morgans Gesetz

(8) NOT $(a \cdot e)$ = NOT a + NOT e (8)' NOT $(a + e)$ = NOT a · NOT e

Operationen mit 0 und 1

(9.1) $a \cdot 1 = a$ (9.1)' $a + 0 = a$

(9.2) $a \cdot 0 = 0$ (9.2)' $a + 1 = 1$

(9.3) NOT 0 = 1 (9.3)' NOT 1 = 0

Halbaddierer und Volladdierer

Die Ziffern einer im Dezimalsystem geschriebenen Zahl a ergeben sich als Aneinanderreihung der Koeffizienten aus der Dezimalzerlegung (Summe von Zehnerpotenzen) von a; entsprechend erhalten wir die Darstellung von a im Dualsystem als Aneinanderreihung der Koeffizienten aus der Dualzerlegung (Summe von Zweierpotenzen).

$87_{dezimal} = 8 \cdot 10^1 + 7 \cdot 10^0$

$87_{dezimal} = 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1010111_{dual}$

Addition der Dualzahlen
$a = a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$ und $b = b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$:

		a_3	a_2	a_1	a_0
+		b_3	b_2	b_1	b_0
	s_4	s_3	s_2	s_1	s_0

		1	1	0	1
+		1	0	1	1
	1	1	0	0	0

Den Übertrag („carry"), der sich aus der i-ten Stelle ergibt und der bei der Addition in der (i + 1)-ten Stelle zu berücksichtigen ist, bezeichnen wir mit c_{i+1}; $i \geq 0$.

Für die 0-te Stelle genügt ein Halbaddierer mit den Eingängen a_0 und b_0 und den Ergebnissen s_0 und c_1; die Addition in der i-ten Stelle, $i \geq 1$, erfordert einen Volladdierer mit den Eingängen a_i, b_i, c_i und den Ergebnissen s_i und c_{i+1}.

Halbaddierer HA

Wahrheitstafel:

a_0	b_0	s_0	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Wir ermitteln für $\mathbf{s_0}$ und $\mathbf{c_1}$ jeweils die disjunktive Normalform („Disjunktion der Konjunktionen"):

$$s_0 = \overline{a_0} \cdot b_0 + a_0 \cdot \overline{b_0} = a_0 \oplus b_0$$

$$c_1 = a_0 \cdot b_0$$

Volladdierer VA

Wahrheitstafel:

a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Wir ermitteln für $\mathbf{s_i}$ und $\mathbf{c_{i+1}}$ jeweils die disjunktive Normalform („Disjunktion der Konjunktionen") und vereinfachen ggf. die booleschen Funktionsterme:

$$s_i = \overline{a}_i \cdot \overline{b}_i \cdot c_i + \overline{a}_i \cdot b_i \cdot \overline{c}_i + a_i \cdot \overline{b}_i \cdot \overline{c}_i + a_i \cdot b_i \cdot c_i$$

ohne Index i geschrieben:

$$s = \overline{a} \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot c$$

$$s = (\overline{a} \cdot b + a \cdot \overline{b}) \cdot \overline{c} + \overline{a} \cdot \overline{b} \cdot c + a \cdot b \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + (\overline{a} \cdot \overline{b} + a \cdot b) \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + (0 + \overline{a} \cdot \overline{b} + a \cdot b + 0) \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + (\overline{a} \cdot a + \overline{a} \cdot \overline{b} + a \cdot b + b \cdot \overline{b}) \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + [(\overline{a} + b) \cdot (a + \overline{b})] \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + [(\overline{a} + \overline{\overline{b}}) \cdot (\overline{\overline{a}} + \overline{b})] \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + [\overline{(a \cdot \overline{b})} \cdot \overline{(\overline{a} \cdot b)}] \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + \overline{[a \cdot \overline{b} + \overline{a} \cdot b]} \cdot c$$

$$s = (a \oplus b) \cdot \overline{c} + \overline{(a \oplus b)} \cdot c$$

$$s = (a \oplus b) \oplus c$$

mit Index i erhält man:

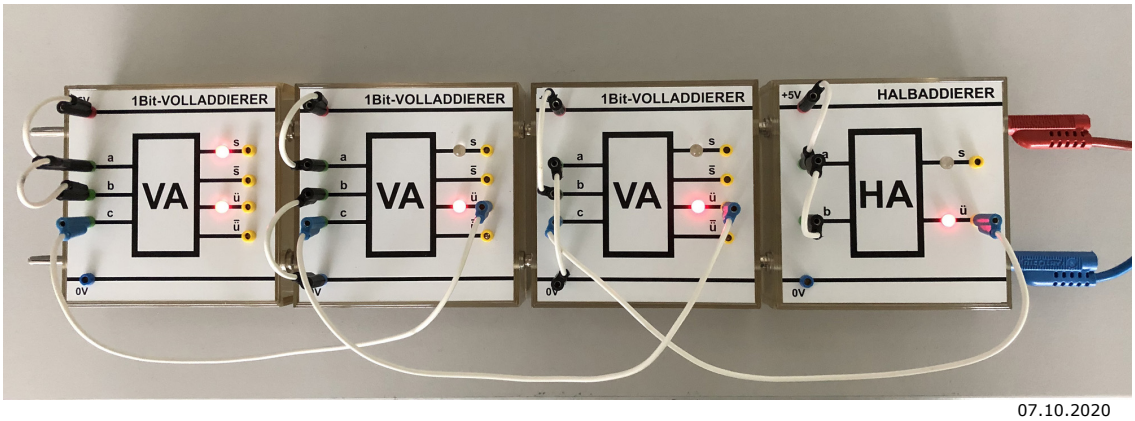
$$s_i = (a_i \oplus b_i) \oplus c_i$$

$$c_{i+1} = \overline{a}_i \cdot b_i \cdot c_i + a_i \cdot \overline{b}_i \cdot c_i + a_i \cdot b_i \cdot \overline{c}_i + a_i \cdot b_i \cdot c_i$$

$$c_{i+1} = (\overline{a}_i \cdot b_i + a_i \cdot \overline{b}_i) \cdot c_i + a_i \cdot b_i \cdot (\overline{c}_i + c_i)$$

$$c_{i+1} = (\overline{a}_i \cdot b_i + a_i \cdot \overline{b}_i) \cdot c_i + a_i \cdot b_i \cdot 1$$

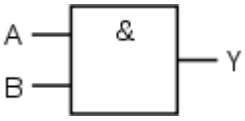
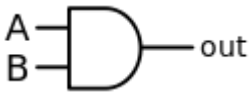
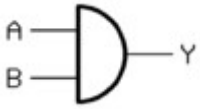
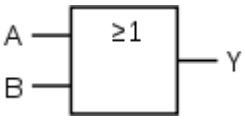
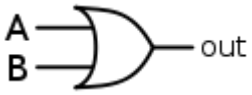
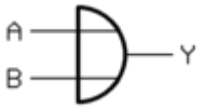
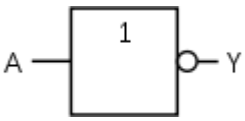
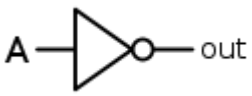
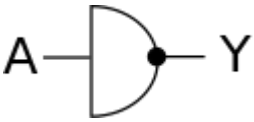
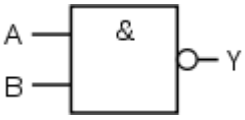
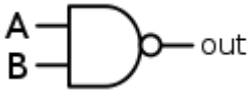
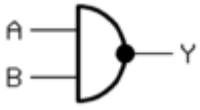
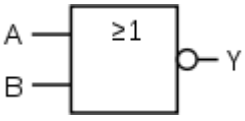
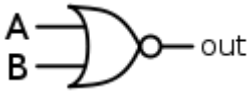
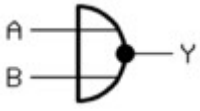
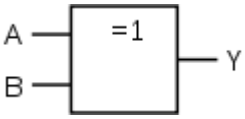
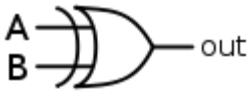
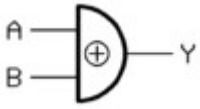
$$c_{i+1} = (a_i \oplus b_i) \cdot c_i + a_i \cdot b_i$$



07.10.2020

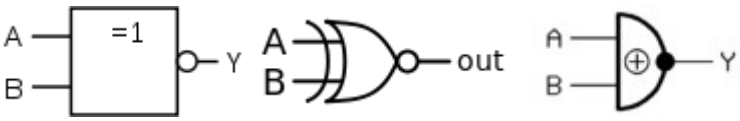
Typen von Logikgattern und Symbolik

Logikgatter werden mit Schaltsymbolen bezeichnet, die nach unterschiedlichen, mehr oder weniger parallel existierenden Standards definiert sind.

Name	Funktion	Symbol in Schaltplan			Wahrheits-tabelle
		IEC 60617-12 : 1997 & ANSI/IEEE Std 91/91a-1991	**ANSI/IEEE Std 91/91a-1991**	**DIN 40700 (vor 1976)**	
Und-Gatter (AND)	$Y=A \cdot B$				**A B Y** 0 0 0 0 1 0 1 0 0 1 1 1
Oder-Gatter (OR)	$Y=A+B$				**A B Y** 0 0 0 0 1 1 1 0 1 1 1 1
Nicht-Gatter (NOT)	$Y=\overline{A}$				**A Y** 0 1 1 0
NAND-Gatter (NICHT UND) (NOT AND)	$Y=\overline{A \cdot B}$				**A B Y** 0 0 1 0 1 1 1 0 1 1 1 0
NOR-Gatter (NICHT ODER) (NOT OR)	$Y=\overline{A+B}$				**A B Y** 0 0 1 0 1 0 1 0 0 1 1 0
XOR-Gatter (Exklusiv-ODER, Antivalenz) (eXclusiveOR)	$Y=A \oplus B$				**A B Y** 0 0 0 0 1 1 1 0 1 1 1 0

XNOR-Gatter
(Exklusiv-Nicht-ODER, Äquivalenz)
(eXclusive Not OR)

$Y = \overline{A \oplus B}$



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Früher waren auf dem europäischen Kontinent die deutschen Symbole (rechte Spalte) verbreitet; im englischen Sprachraum waren und sind die amerikanischen Symbole (mittlere Spalte) üblich. Die IEC-Symbole sind international auf beschränkte Akzeptanz gestoßen und werden in der amerikanischen Literatur (fast) durchgängig ignoriert.

JK-Flipflop

Ein Flip-Flop (bistabile Kippstufe oder bistabiler Multivibrator) hat zwei stabile Zustände am Ausgang Q; die Zustände heißen „gesetzt“ (set) oder „zurückgesetzt“ (reset). Ein 1-Bit-Speicher läßt sich somit als FlipFlop realisieren.

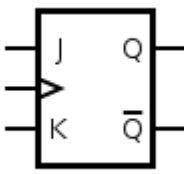
Ein JK-FlipFlop ist ein taktgesteuertes FlipFlop: die an den Eingängen J und K liegende Information wird mit einer Flanke (hier: der steigenden Flanke) des an C liegenden Taktsignals auf die Ausgänge Q und $\overline{Q}$ übernommen.

Mit dem Taktsignal (clock, C) und der Eingangsbelegung J = 1 und K = 0 wird am Ausgang Q eine 1 erzeugt und gespeichert, alternativ eine 0 bei J = 0 und K = 1.

Bei der Realisierung des JK-Flipflops als taktflankengesteuertes Flipflop kann der Eingang C für steigende Flanken (Wechsel von 0 auf 1) oder für fallende Flanken (Wechsel von 1 auf 0) ausgelegt sein.

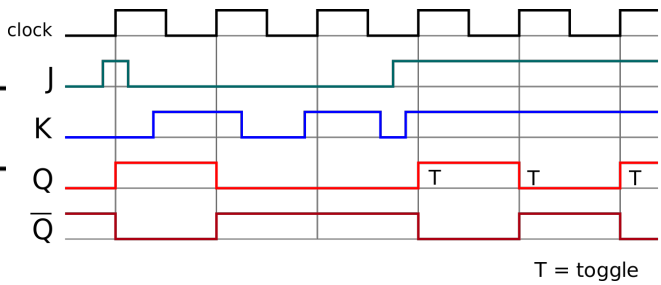
Name und Schaltzeichen

Flanken-gesteuertes JK-Flipflop



Signal-Zeit-Diagramm

Übernahme der Eingangsinformation durch steigende Flanke an C (clock)



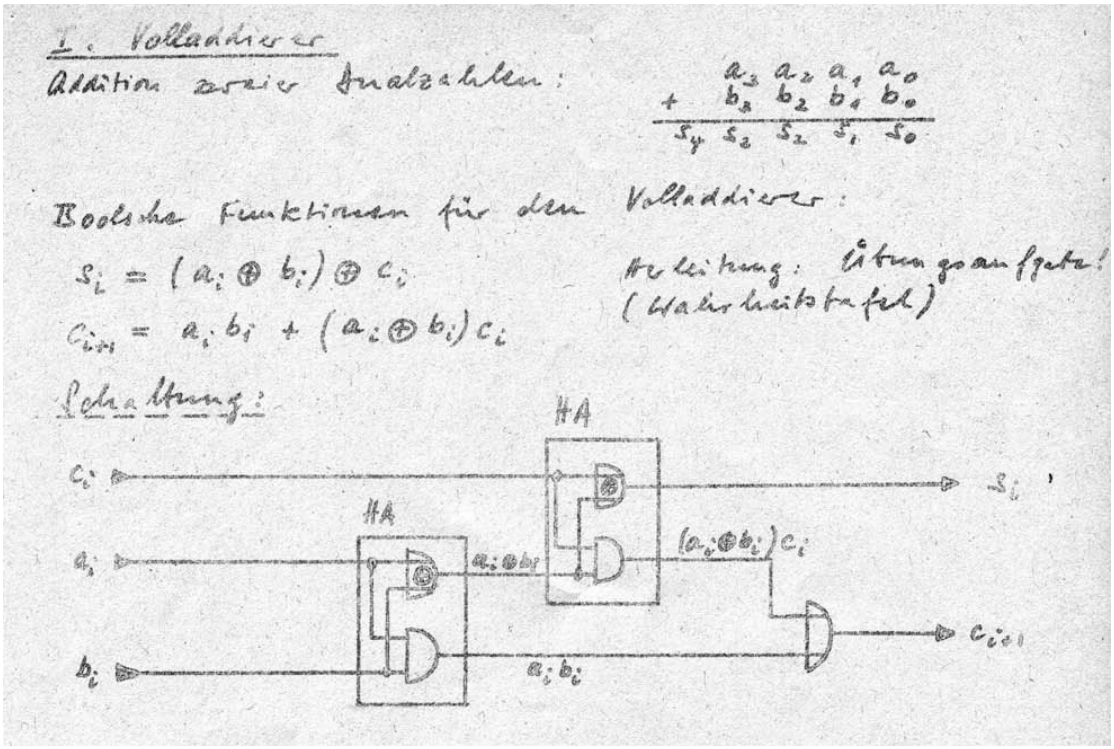
Funktionstabelle

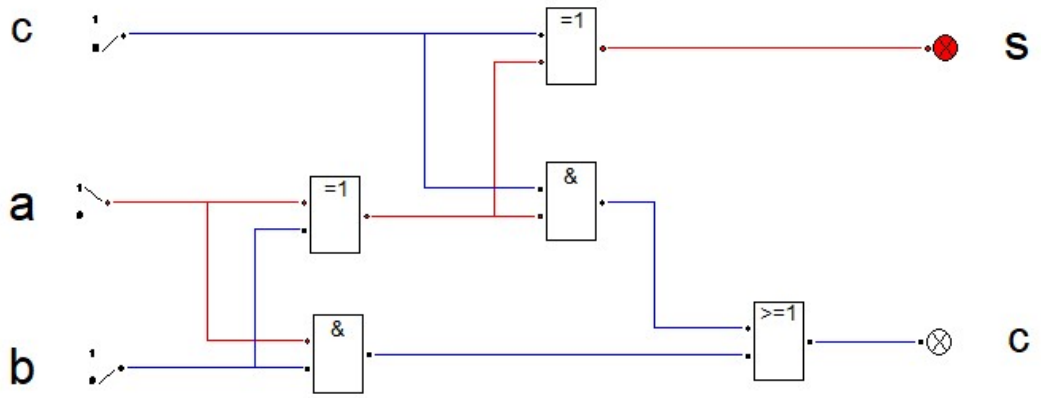
bis zur … n-ten Taktflanke		nach der … n-ten Taktflanke
J	**K**	**Q_n**
0	0	Q_{n-1} (unverändert)
0	1	0 (zurückgesetzt)
1	0	1 (gesetzt)
1	1	NOT Q_{n-1} (gewechselt)

(Wikipedia)

Halbaddierer (HA) und Volladdierer (VA)

Schaltungen





Informatik 13
Übungsaufgabe 07.10.2020

Gegeben ist die boolesche Funktion z = f(a, b, c) vermöge folgender Wahrheitstafel:

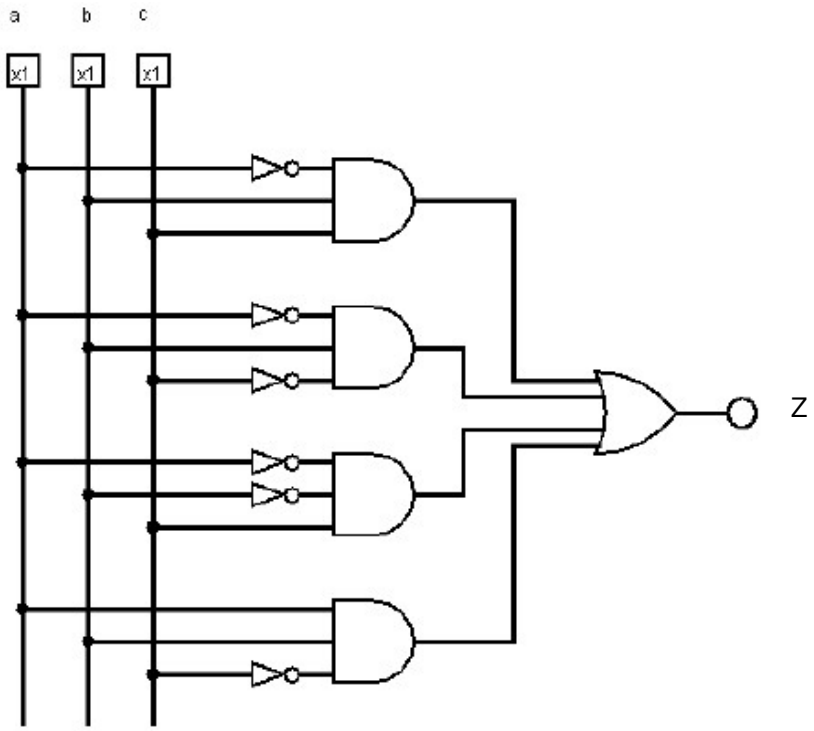
a	b	c	z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

a) Leite den Booleschen Funktionsterm für die Funktion f her.

b) Vereinfache diesen Term mit Hilfe der Rechenregeln für Boolesche Terme.

Aufgabenblatt Nr. 1 13inf 28.10.2020

1. Gegeben ist folgende digitale Schaltung mit den Eingängen a, b, c und dem Ausgang z:



a) Erstelle die Wahrheitstafel für diese Schaltung und ermittle die disjunktive Normalform für die Boolesche Funktion z = f(a,b,c).

b) Vereinfache den Funktionsterm für z und zeichne die vereinfachte Schaltung.

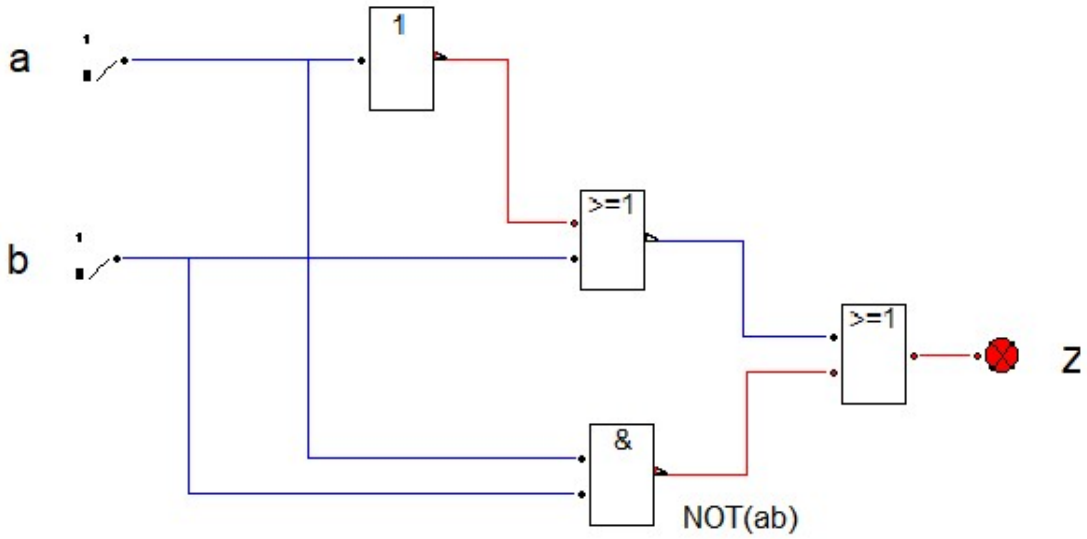
2. Die Boolesche Funktion z = f(a,b,c) ist durch folgende Wahrheitstafel gegeben:

a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

a) Ermittle die disjunktive Normalform für z und vereinfache den Funktionsterm.

b) Zeichne den Schaltplan für die optimierte Funktion z.

3. Gegeben ist folgende digitale Schaltung mit den Eingangsvariablen a, b und der Ausgangsvariablen z:

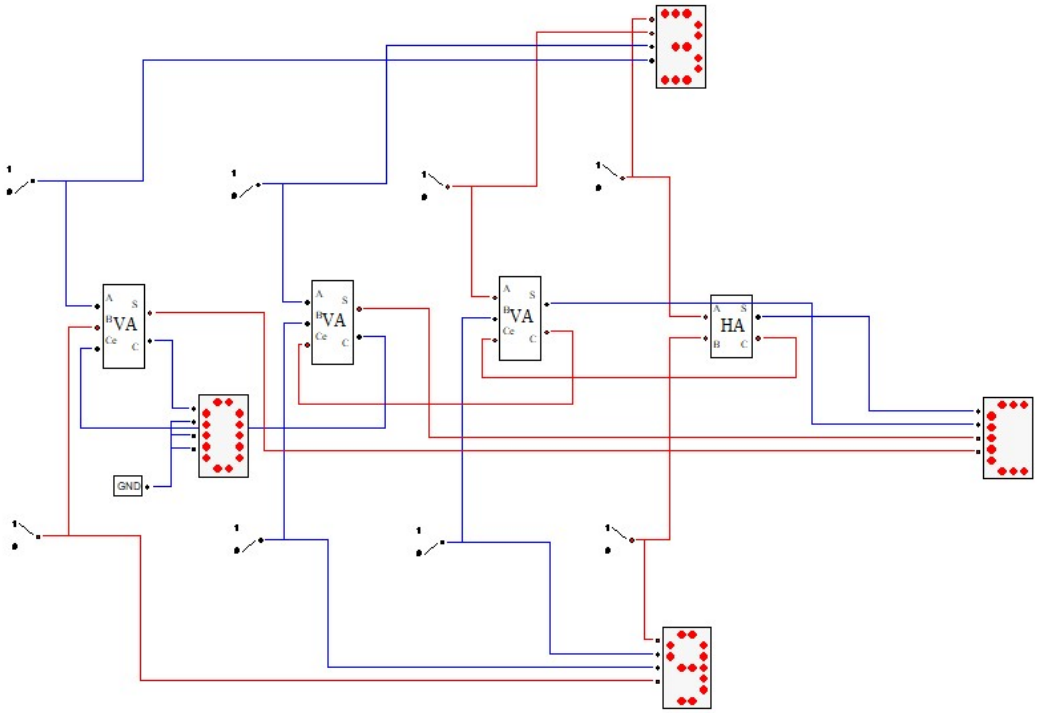


   a) Ermittle den Booleschen Term für die Boolesche Funktion z = f(a,b,c). Hinweis: Notiere am Ausgang jedes Gatters jeweils den Booleschen Term (Beispiel: $\overline{a \cdot b}$ am Ausgang des NAND-Gatters).

   b) Vereinfache den in a) erhaltenen Term unter Verwendung der Rechenregeln für Boolesche Ausdrücke; erstelle die Wahrheitstafel.

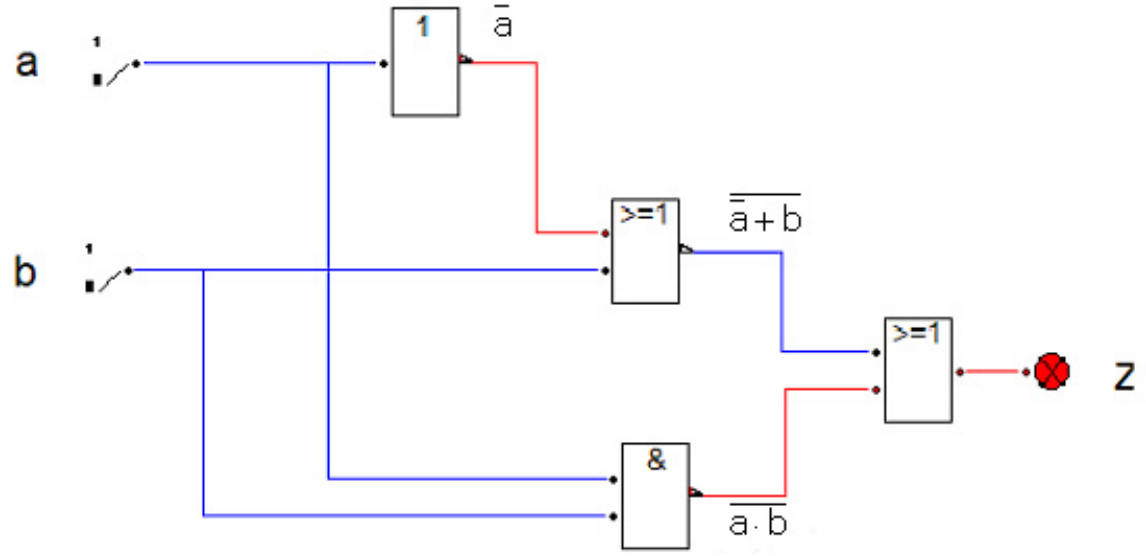
   c) Zeichne das Schaltbild für den vereinfachten Funktionsterm und teste beide Schaltungsvarianten mit einem Digitalsimulator.

4. 4-Bit-Paralleladdierer mit Anzeige der Summanden und der Summe jeweils im Hexadezimalformat



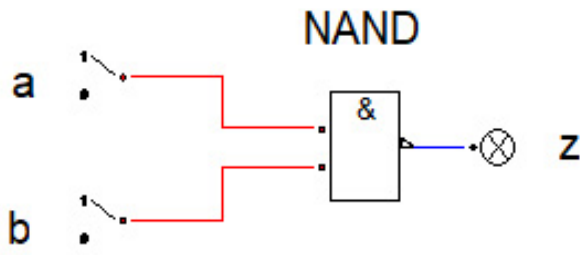
   Erweitere die Schaltung „4-bit-Paralleladdierer.dsim“ (auf www.kalle2000.de downloadbar) zu einem 8-Bit-Addierer mit numerischer Anzeige.

Lösung zu Nr. 3 von Aufgabenblatt 2 vom 29.10.2020



$$
\begin{aligned}
z &= \overline{\bar{a} + b} + \overline{a \cdot b} \\
&= \bar{\bar{a}} \cdot \bar{b} + (\bar{a} + \bar{b}) && \textit{(2-mal de Morgan)} \\
&= a \cdot \bar{b} + \bar{a} + \bar{b} && \textit{(wegen } \bar{\bar{a}} = a\textit{)} \\
&= \bar{a} + \bar{b} \cdot a + \bar{b} && \textit{(Kommutativgesetze)} \\
&= \bar{a} + \bar{b} \cdot a + \bar{b} \cdot 1 && \textit{(wegen } a = a \cdot 1\textit{)} \\
&= \bar{a} + \bar{b} \cdot (a + 1) && \textit{(Distributivgesetz)} \\
&= \bar{a} + \bar{b} && \textit{(wegen } a + 1 = 1\textit{)} \\
&= \overline{a \cdot b} && \textit{(de Morgan)}
\end{aligned}
$$

optimierte Schaltung:



Wertetabelle:

a	b	z
0	0	1
0	1	1
1	0	1
1	1	0

5. Für die Boolesche Funktion y = f(a,b,c) ist folgende Wertetafel gegeben:

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

a) Ermittle die DNF (disjunktive Normalform) für y.

b) Vereinfache den Funktionsterm unter Verwendung der Booleschen Rechenregeln.

c) Zeichne das Schaltbild für die vereinfachte Funktion.

13inf sHÜ 18.11.2020 Name: _____

Die Boolesche Funktion
z = f(a,b,c)
ist durch nebenstehende Wahrheitstafel gegeben:

a	b	c	z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

a) Ermittle die disjunktive Normalform für z.

b) Vereinfache den Funktionsterm unter Anwendung der Booleschen Rechengesetze.

c) Zeichne den Schaltplan für die optimierte Funktion z.

Lösung:

a) $z = \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot c + a \cdot b \cdot c$

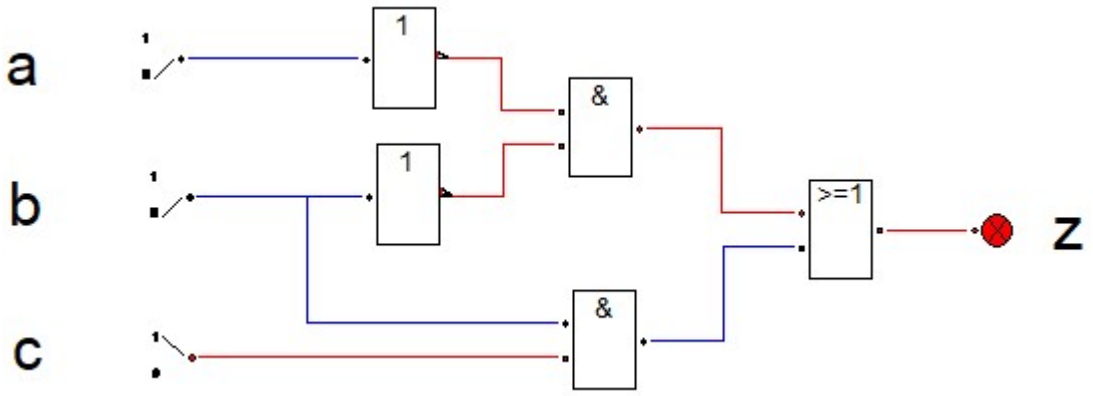
b) $z = \bar{a} \cdot \bar{b} \cdot (\bar{c} + c) + b \cdot c \cdot (\bar{a} + a)$ *Kommutativ- und Distributivgesetz*

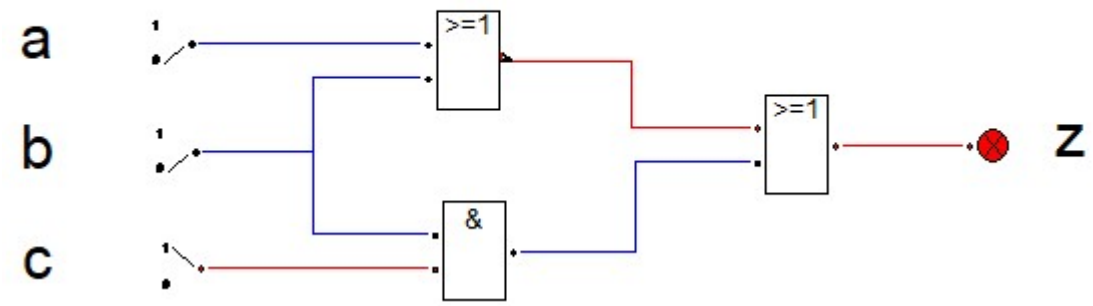
$= \bar{a} \cdot \bar{b} \cdot 1 + b \cdot c \cdot 1$

$= \bar{a} \cdot \bar{b} + b \cdot c$

$= \overline{a+b} + b \cdot c$

c)





Die wesentlichen Komponenten einer **CENTRAL PROCESSING UNIT** (**CPU**) bestehen aus der **CONTROL UNIT** (**CU**) und der **ARITHMETIC LOGIC UNIT** (**ALU**).

Die **ALU** berechnet arithmetische und logische Funktionen, die **CU** decodiert die im Arbeitsspeicher abgelegten Befehle und führt sie aus.

In der Minimalkonfiguration beherrscht die **ALU** die arithmetische Funktion **„Addition"** sowie die logischen Operationen **„Negation"** (NOT) und **„Konjunktion"** (AND). Zu Lasten der Rechenzeit lassen sich die übrigen arithmetischen und logischen Funktionen auf die genannten, minimal verfügbaren Operationen zurückführen.

1. Subtraktion

Die duale Subtraktion

	a_3	a_2	a_1	a_0
−	b_3	b_2	b_1	b_0
	d_3	d_2	d_1	d_0

läßt sich auf eine duale Addition nach folgendem Verfahren zurückführen:

- Bilde das Einerkomplement des Subtrahenden $b_3\ b_2\ b_1\ b_0$, indem man alle Ziffern negiert (invertiert; aus 0 wird 1 und aus 1 wird 0).
- Addiere das Einerkomplement und die Zahl 1 zum Minuenden.
- Das Ergebnis ist die gesuchte Differenz; dabei bleibt der Überlauf unberücksichtigt.

a) Verdeutliche das genannte Verfahren anhand einiger selbst gewählter Beispiele (ein Beweis des Verfahrens ist nicht erforderlich.).

b) Ergänze die Schaltung „4-bit-Paralleladdierer.dsim" so, daß man nach entsprechender Umschaltung wahlweise eine duale Addition oder eine duale Subtraktion durchführen kann.
Hinweise:
   - Ersetze den HA für das least significant bit (LSB) durch einen VA, um erforderlichenfalls eine „1" als Summand einspeisen zu können (wie?).
   - Die Invertierung der Ziffern des Subtrahenden gelingt z. B. durch den geeigneten Einsatz von XOR-Gattern.

2. Weitere Rechenoperationen

Gegeben sind die (im einfachsten Fall positiven ganzzahligen) Operanden a und b. Um zu verdeutlichen, wie man die „höheren" Rechenoperationen mittels geeigneter Iteration auf die Grundoperationen „Addition" und „Subtraktion" zurückführen kann, schreibe und teste ein Python-Programm, welches die Operationen „Multiplikation" (a*b), „Division" (a/b, ganzzahlige Division) und „Potenzierung" (a**b) realisiert.

3. Logische Operationen

Zeige examplarisch, daß sich die logischen Verknüpfungen

a) $a + b$
b) $a \oplus b$
c) $a \cdot (b+\overline{c})$

auf die Operationen NOT und AND zurückführen lassen.

11.11.20 # SCHIEBREGISTER und DUALZÄHLER

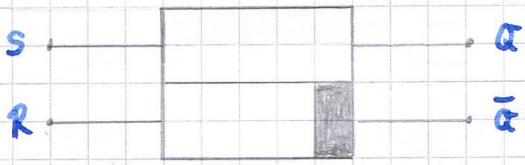
Definition: Unter einem Flip-Flop verstehen wir eine bistabile Kippstufe mit genau 2 stabilen Zustände, so dass der Informationsgehalt von 1 Bit gespeichert werden kann.

1 ≙ TRUE ≙ 5 Volt

0 ≙ FALSE ≙ 0 Volt

Schaltzeichen:

a) nicht getaktet:



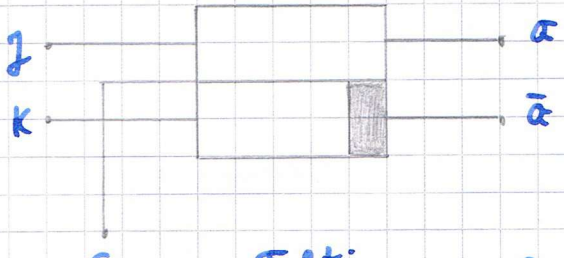
S = Set

R = Reset

Zustandstabelle:

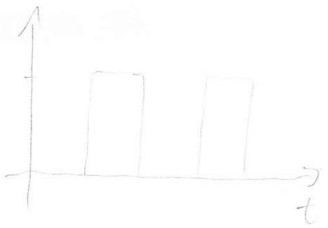
S	R	Folgezustand Q	Folgezustand $\bar{Q}$
0	0	keine Änderung	
0	1	0	1
1	0	1	0
1	1	–	–

b) getaktet:

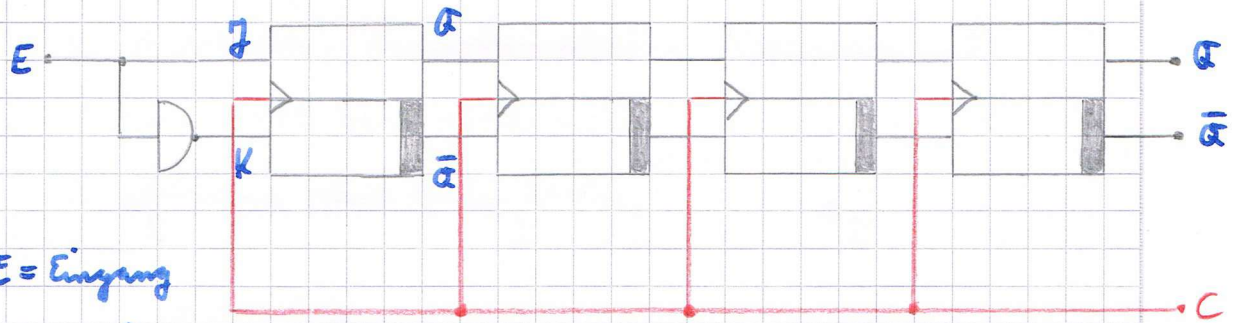


Takteingang: C = clock

Die an den Eingängen J und K komplementär anliegende Information wird erst dann auf die Ausgänge Q und $\bar{Q}$ übertragen, wenn an C die fallende (oder auch die steigende) Taktflanke eines Rechtecksignals erscheint



SCHIEBEREGISTER (hier: 4-Bit-Register)



E = Eingang

C = Clock

Bei jedem Taktimpuls an C wird die Information um eine Stelle nach „rechts" geschoben.

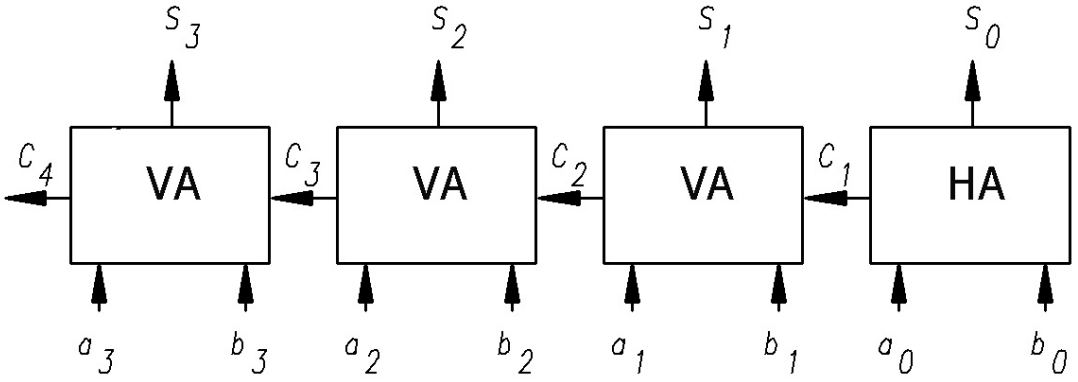
Robin Becher

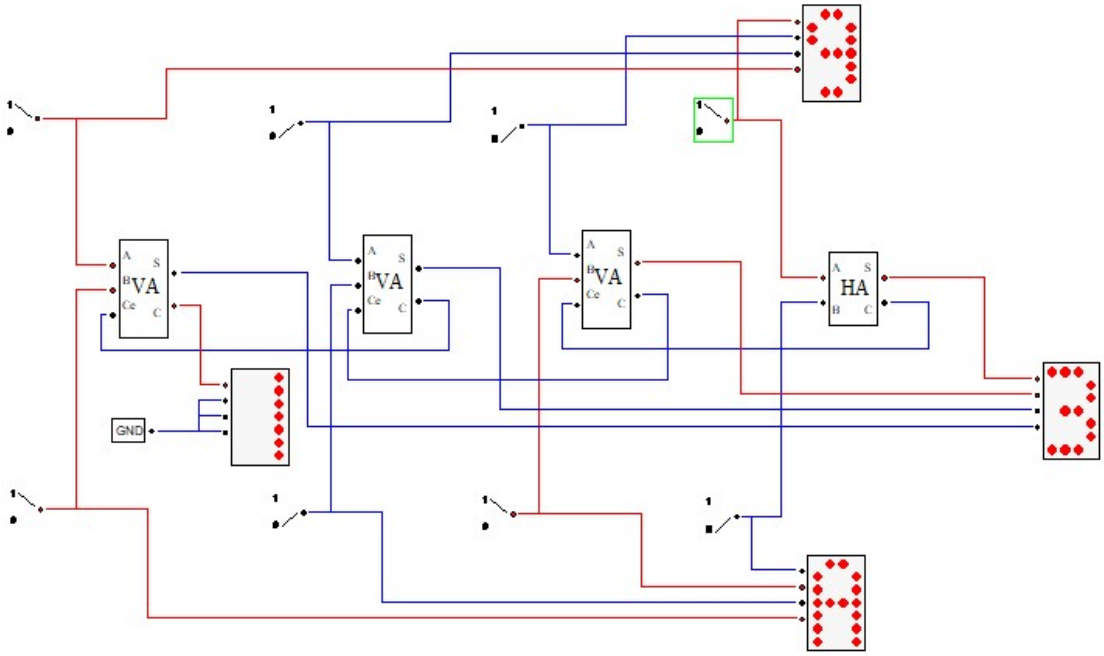
Addier-Schaltungen für Dualzahlen

		a_3	a_2	a_1	a_0
+		b_3	b_2	b_1	b_0
	s_4	s_3	s_2	s_1	s_0

1. Paralleladdierer mit seriellem Übertrag (hier: 4-Bit-Addierer)

Für das Least Significant Bit (LSB) genügt ein Halbaddierer (HA); die höherwertigen Bits erfordern jeweils einen Volladdierer, da hier der Übertrag aus der vorherigen Stelle zu berücksichtigen ist.



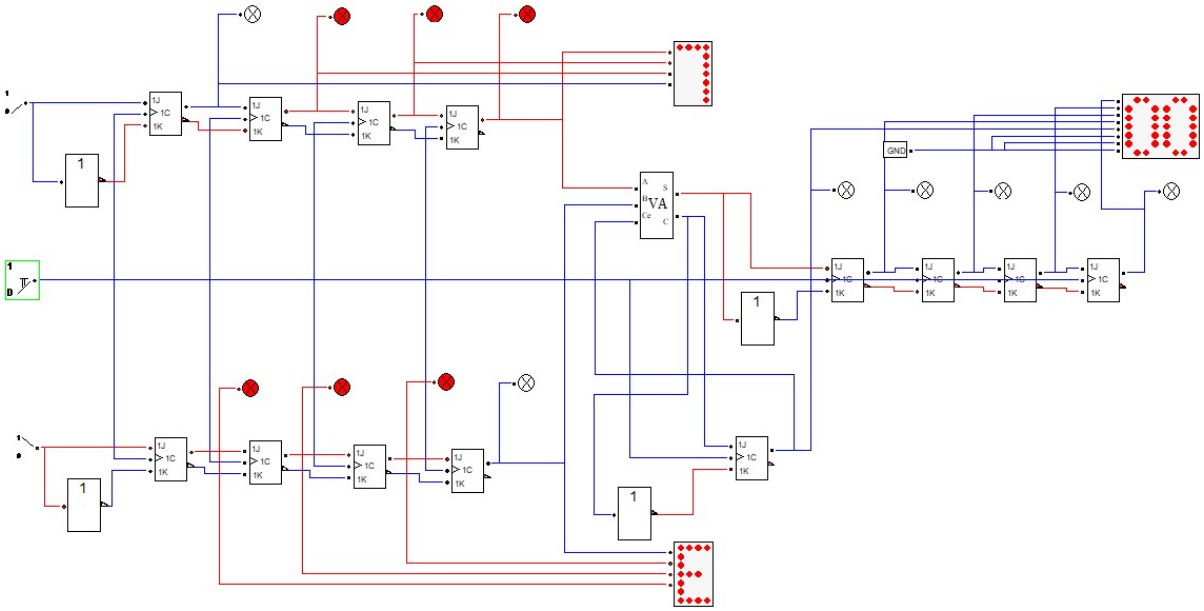



```
Dezimal:  09   Hexadezimal:  09    Dual: 0000 1001
        + 10                 + 0A        + 0000 1010
          19                   13          0001 0011
```

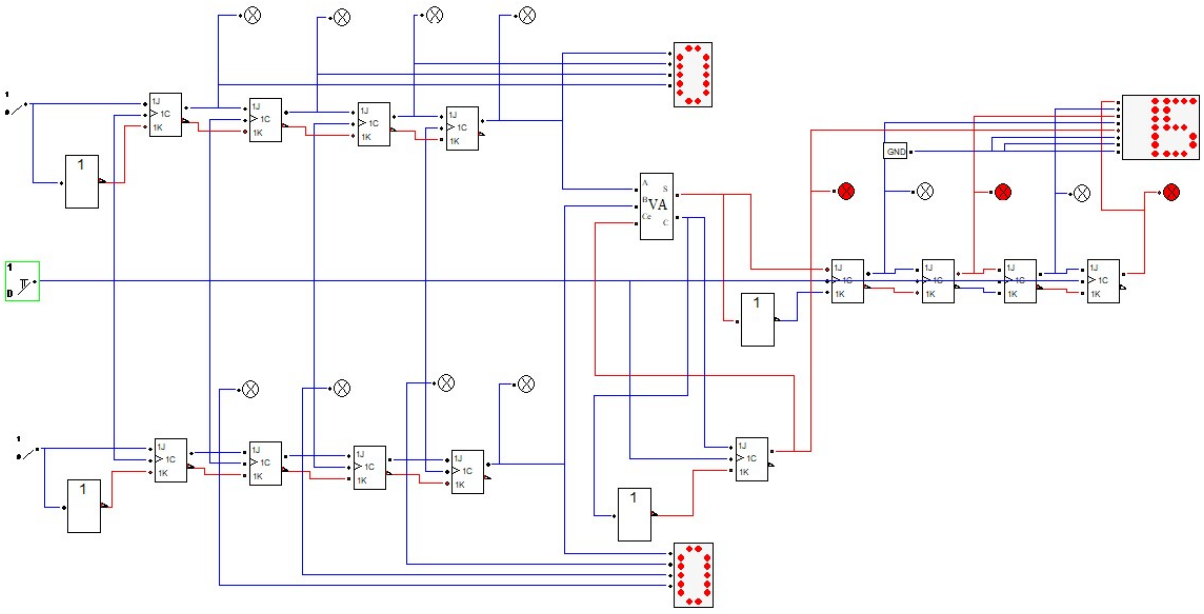
2. Serieller 1-Bit-Addierer für 4-stellige Dualzahlen

Die Operanden werden jeweils in einem 4-Bit-Schieberegister abgelegt, nach 4 Taktimpulsen finden wir das Ergebnis (hier: die Summe) in einem weiteren 4-Bit-Schieberegister.

Da der Übertrag aus der vorherigen Stelle für die Addition in der aktuellen Stelle zu berücksichtigen ist, wird er in einem Flip-Flop zwischengespeichert. Dieses Flip-Flop liefert auch das Most Significant Bit (MSB) des Ergebnisses.

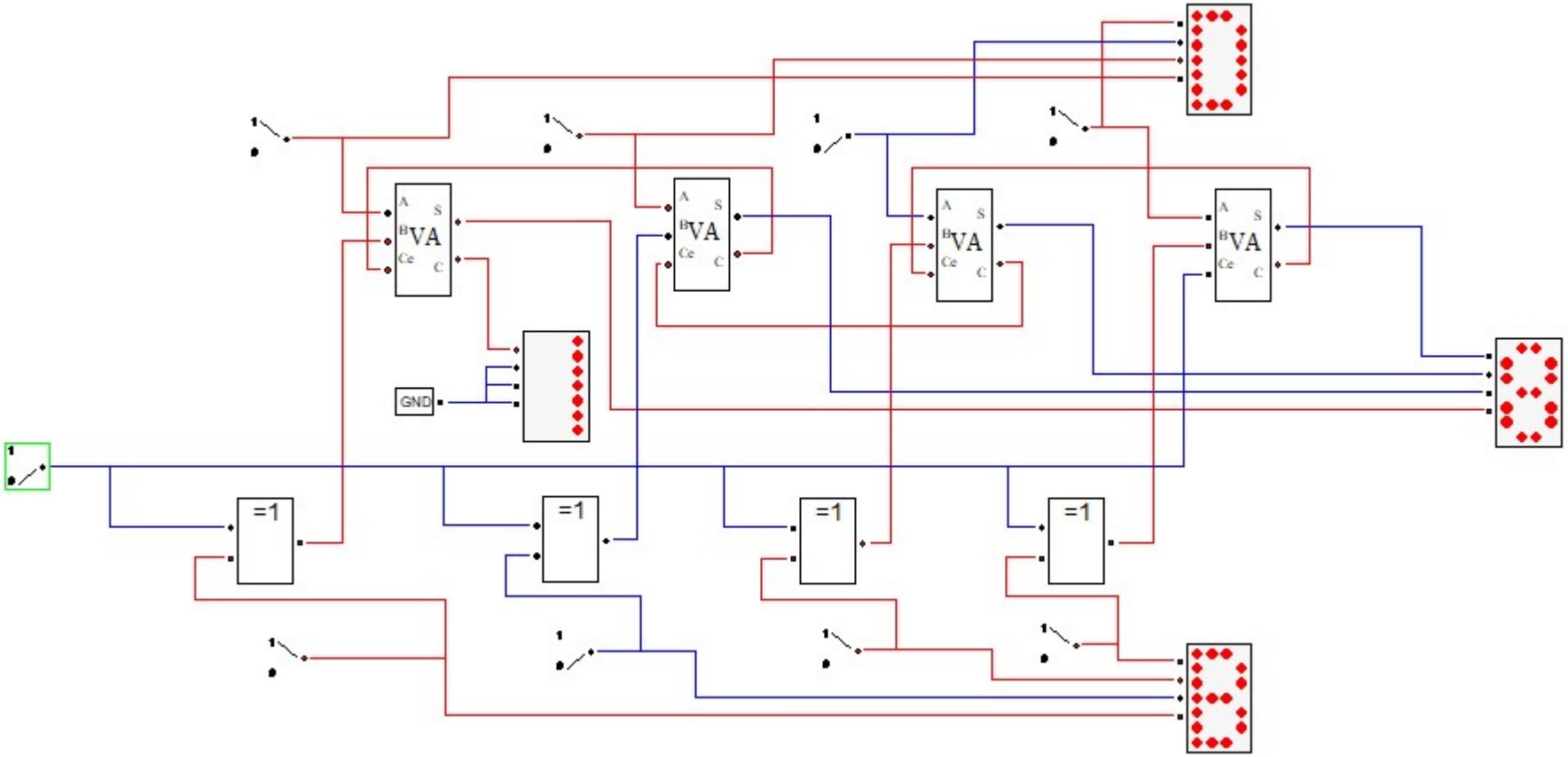


Nach 4 Taktimpulsen (hier: Triggerung der Flip-Flops auf der steigenden Taktflanke):



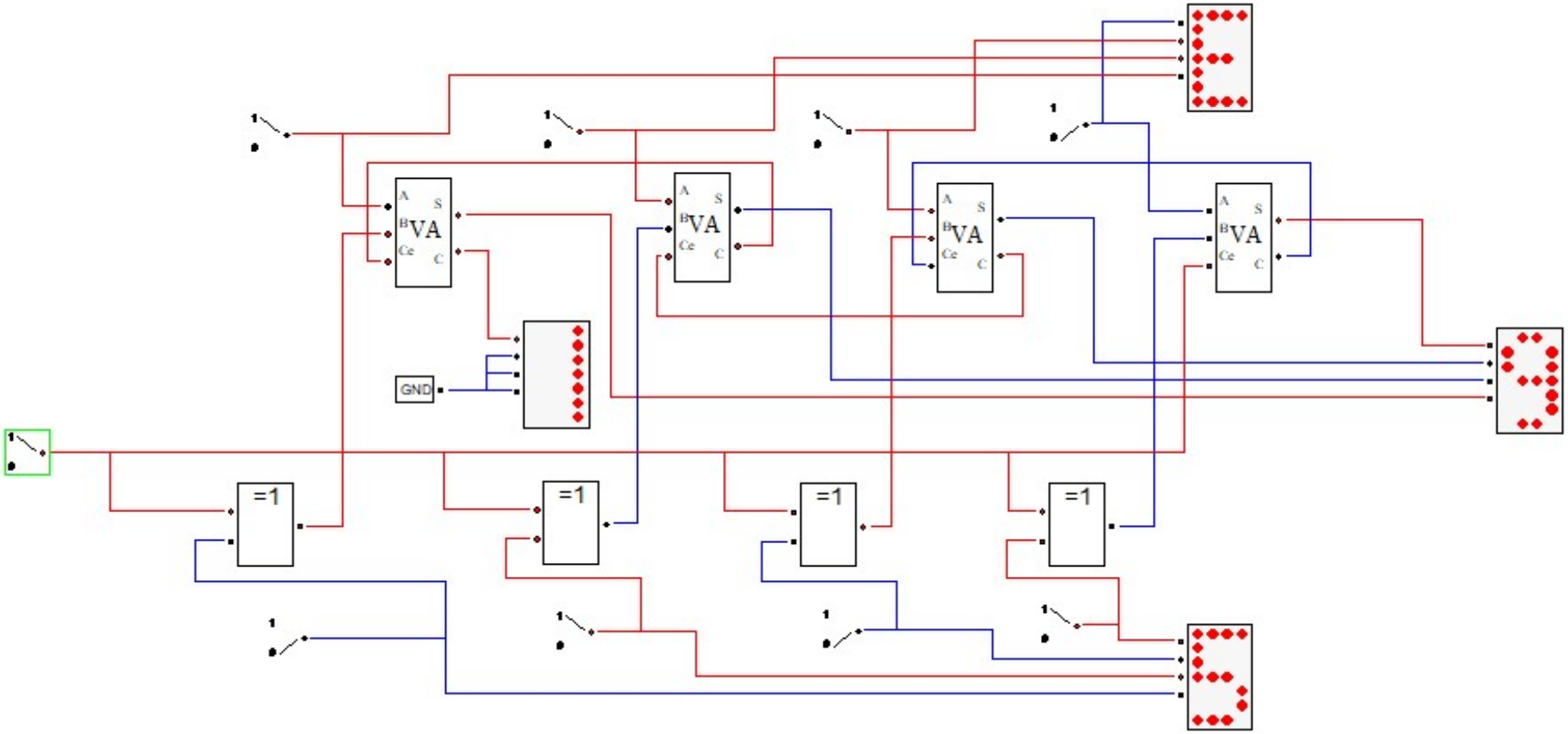
Dezimal:	Hexadezimal:	Dual:
07	07	0000 0111
+ 14	+ 0E	+ 0000 1110
21	15	0001 0101

4-Bit-Paralleladdier mit Umschaltung auf 4-Bit-Parallelsubtraktion



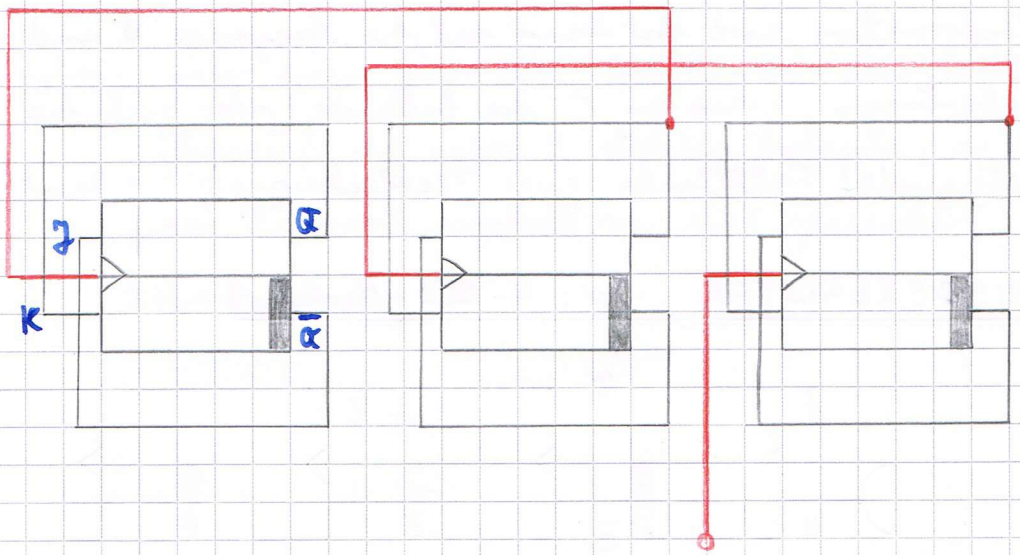
Dezimal:	13	Hexadezimal:	0D	Dual:	0000 1101
	+ 11		+ 0B		+ 0000 1011
	24		18		0001 1000

4-Bit-Paralleladdier mit Umschaltung auf 4-Bit-Parallelsubtraktion



Dezimal:	14	Hexadezimal:	0E	Dual:	0000 1110
	- 05		- 05		- 0000 0101
	09		09		0000 1001

18.11.20 **Dualzähler**



z = Anzahl der an C eingespeisten Taktimpulsen
(genauer: die Anzahl der triggernden Taktflanken,
hier: fallende Taktflanke)

	z	2^2	2^1	2^0
nach 0 Taktimpulsen	0	0	0	0
nach 1 Taktimpuls	1	0	0	1
nach 2 Taktimpulsen	2	0	1	0
nach 3 Taktimpulsen	3	0	1	1
nach 4 Taktimpulsen	4	1	0	0
nach 5 Taktimpulsen	5	1	0	1
nach 6 Taktimpulsen	6	1	1	0
nach 7 Taktimpulsen	7	1	1	1

Bei 3 Flip-Flops: 8 Zustände

Bei n Flip-Flops: 2^n Zustände

Zählbereich: 0 bis 2^n-1

Robin Becher

4-Bit-Dualzähler

Schaltung mit alphanumerischer Anzeige des Zählergebnisses

Anzahl der Zustände: 2^4

Zählbereich: 0 . . . . . $2^4 - 1$

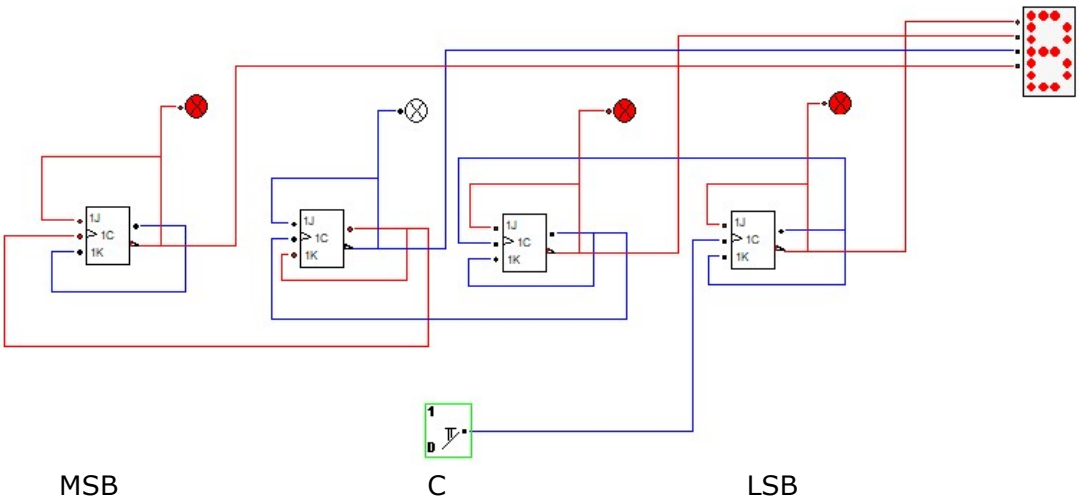
0000	-	1111	(dual)
00	-	FF	(hexadezimal)
0	-	255	(dezimal)

Die am Eingang C eintreffenden Taktimpulse werden gezählt.

Beachte:

Bei Triggerung auf der fallenden Taktflanke erfolgt die Anzeige jeweils am Ausgang Q.

Bei Triggerung auf der steigenden Taktflanke erfolgt die Anzeige jeweils am Ausgang $\overline{Q}$.



MSB C LSB

hier: Triggerung auf der steigenden Taktflanke, daher werden die Stellen (Bits) der Dualzahl jeweils dem Ausgang $\overline{Q}$ eines jeden Flip-Flops entnommen.