Technische Informatik

Digitaltechnik 2

Übersicht

- **Halbaddierer**
- **Volladdierer**
- **Carry-Ripple-Addierer**
- **Carry-Lookahead-Addierer**
- **Festkommamultiplizierer**
- **Zählschaltungen**
- **Komparator**

DHBW
Duale Hochschule
Baden-Württemberg

Literatur

- Elektronik 4: Digitaltechnik, K. Beuth, Vogel Fachbuch
- Digitaltechnik, K. Fricke, Springer Vieweg
- Digitaltechnik, R. Woitowitz, Springer
- Grundlagen der Digitaltechnik, G. W. Wöstenkühler, Hanser

Halbaddierer

Halbaddierer

- Ein **Halbaddierer** ist ein Schaltnetz, das üblicherweise als digitale Schaltung realisiert wird.
- Er besteht aus zwei Eingängen und zwei Ausgängen.
- Mit einem Halbaddierer kann man zwei einstellige Binärzahlen addieren.
- Dabei liefert der Ausgang *s* („Summe“) und der Ausgang *c* (*carry* – „Übertrag“)

Halbaddierer

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 ← mit Übertrag

Die folgende Wahrheitstabelle zeigt die Funktionsweise eines Halbaddierers

x	y	Übertrag c	Summe s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Das entspricht den Gleichungen

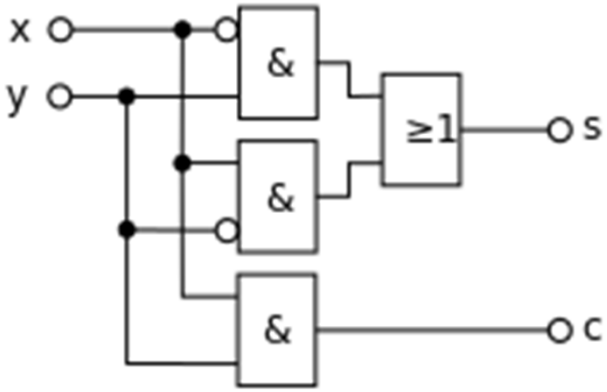
$$c = x \wedge y$$

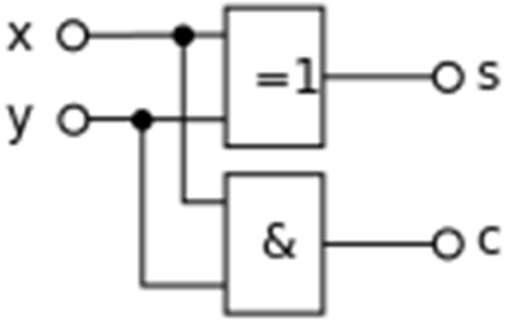
und

$$s = x \veebar y = x \oplus y = (x \wedge \neg y) \vee (\neg x \wedge y).$$

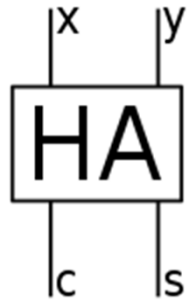
Halbaddierer

$$s = x \veebar y = x \oplus y = (x \wedge \neg y) \vee (\neg x \wedge y). \qquad c = x \wedge y$$





Schaltsymbol

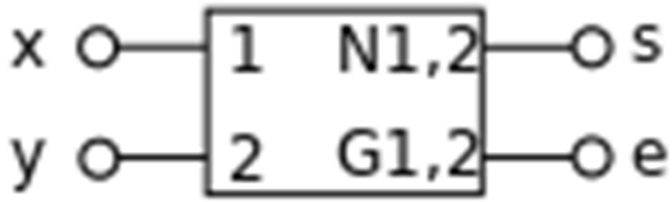


Abhängigkeitsnotation

Die **Abhängigkeitsnotation** ist eine Methode, um die Funktion von Schaltsymbolen digitaler Bauelemente in der Elektronik zu kennzeichnen.

Buch-stabe	Bedeutung	Beschreibung
G	AND	UND-Verknüpfung
V	OR	ODER-Verknüpfung (OR)
N	XOR, Negate	Exklusiv-Oder-Verknüpfung bzw. Negation
Z		unveränderte Übertragung
C	Control	Steuerung
S	Set	Setzen
R	Reset	Zurücksetzen
EN	Enable	Einschalten
M	Mode	Modus
L	Load	Laden
T	Toggle	Umschalten
A	Address	Adresse
CT	Content	(Zähler-/Speicher-) Inhalt

- Abhängigkeitsnotation lässt sich die Funktion komplizierter Bauelemente übersichtlich darstellen.
- Bei der Bezeichnung wird einerseits ein Kennbuchstabe verwendet, der die Funktion des Anschlusses angibt, sowie Kennzahlen, über die die Abhängigkeiten definiert werden.



Halbaddierer

- Halbaddierer sind häufig Bestandteil von Mikroprozessoren.
- Mit diskreten Logikbauelementen wird diese Schaltungsfunktion heute kaum mehr realisiert, da mit diesen Bauelementen die erforderlichen meist hohen Taktfrequenzen nicht erreicht werden können und der Schaltungsaufwand für den Aufbau und die Verdrahtung viel zu groß ist.
- Aus zwei Halbaddierern und einem zusätzlichen Oder-Gatter kann ein Volladdierer aufgebaut werden.
- Der Halbaddierer wird in Kombination mit Volladdierern zum Aufbau von Addiernetzen verwendet.

Volladdierer

Volladdierer

- Ein **Volladdierer** ist ein Schaltnetz, das üblicherweise als digitale Schaltung realisiert wird. Es besteht aus drei Eingängen (x, y, c_{in}) und zwei Ausgängen (s und c_{out}).
- Mit einem Volladdierer kann man drei einstellige Binärzahlen addieren.
- Dabei liefert der Ausgang s (Summe) die niederwertige Stelle des Ergebnisses, der Ausgang c_{out} (*carry (output)* – Übertrag (Ausgang)) die höherwertige.
- Die Bezeichner c_{in} und c_{out} legen hierbei eine Möglichkeit zur Übertragsbehandlung in Addiernetzen nahe

Volladdierer

Die folgende Wahrheitstabelle zeigt die Funktionsweise eines Volladdierers.

x	y	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Die disjunktive Normalform aus den Wahrheitswerten:

$$c_{\text{out}} = (\overline{x} \wedge y \wedge c_{\text{in}}) \vee (x \wedge \overline{y} \wedge c_{\text{in}}) \vee (x \wedge y \wedge \overline{c_{\text{in}}}) \vee (x \wedge y \wedge c_{\text{in}})$$
$$= (c_{\text{in}} \wedge (x \oplus y)) \vee (x \wedge y)$$

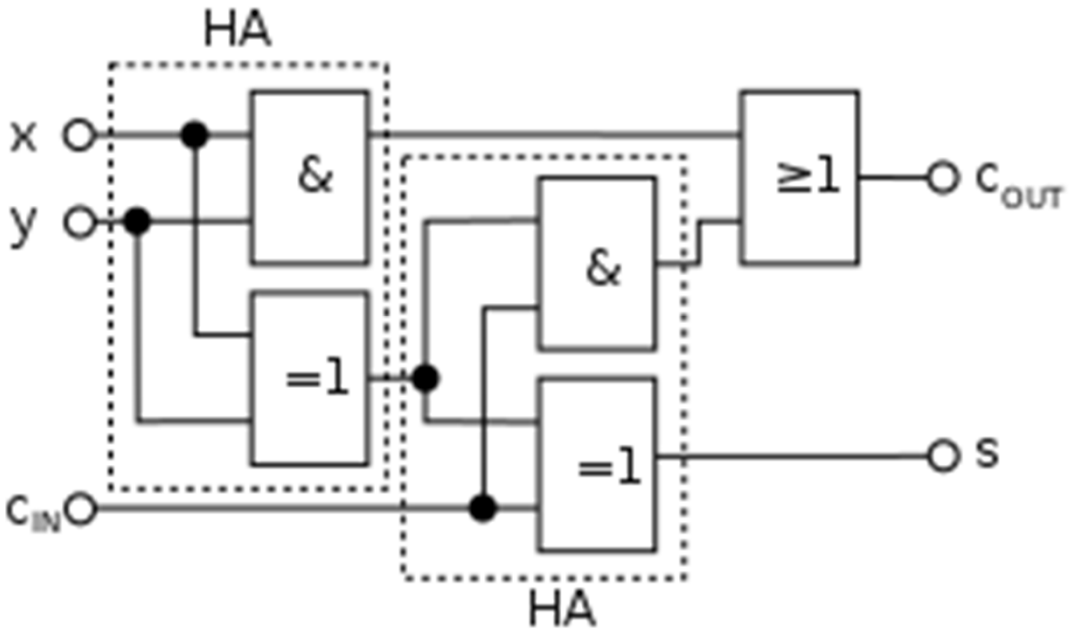
und

$$s = x \oplus y \oplus c_{\text{in}}$$

Volladdierer

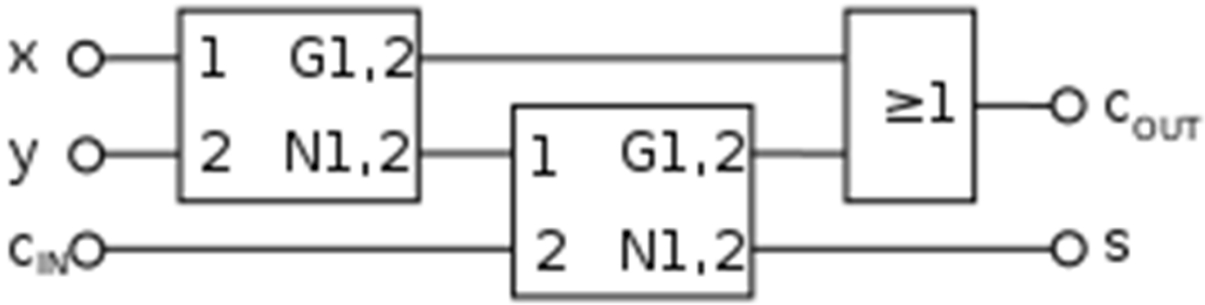
- Die Abbildung zeigt den Aufbau eines Volladdierers, wobei die Halbaddierer jeweils in ein Und-Gatter und ein Exklusiv-Oder-Gatter aufgetrennt wurden.

- Hierbei ist zu beachten, dass in beiden Abbildungen die Summenausgäng s jeweils unten und die Übertragsausgänge der Halbaddierer c_{out} jeweils oben dargestellt sind.



Volladdierer

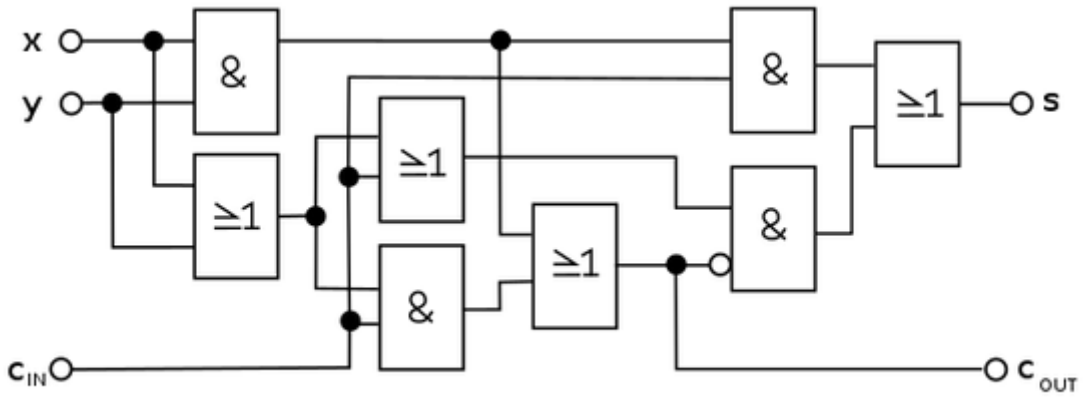
Die Abbildung zeigt den Aufbau eines Volladdierers mittels Halbaddierern und einem Oder-Gatter.



Volladdierer

Optimiert man den Ausdruck für den Volladdierer weiter, ohne den Carry-Pfad zu verlangsamen ergeben sich weitere Vereinfachungen:

$$
\begin{aligned}
I &= x \wedge y \\
J &= x \vee y \\
c_{out} &= I \vee (x \wedge c_{in}) \vee (y \wedge c_{in}) \\
c_{out} &= I \vee ((x \vee y) \wedge c_{in}) \\
c_{out} &= I \vee (J \wedge c_{in}) \\
s &= (x \wedge \overline{c_{out}}) \vee (y \wedge \overline{c_{out}}) \vee (c_{in} \wedge \overline{c_{out}}) \vee (I \wedge c_{in}) \\
s &= ((J \vee c_{in}) \wedge \overline{c_{out}}) \vee (I \wedge c_{in})
\end{aligned}
$$



Volladdierer

- Der Volladdierer wird zum Aufbau von Addierwerken und Multiplizierern verwendet, oft mit einem Halbaddierer am Anfang der Übertragkette.

- Bei der Invertierung aller Eingänge eines Volladdierers invertieren sich alle Ausgänge, dies kann zur Laufzeitoptimierung von Addierwerken verwendet werden, indem auf die Invertierung von c_{out} verzichtet wird.

Carry-Ripple-Addierer

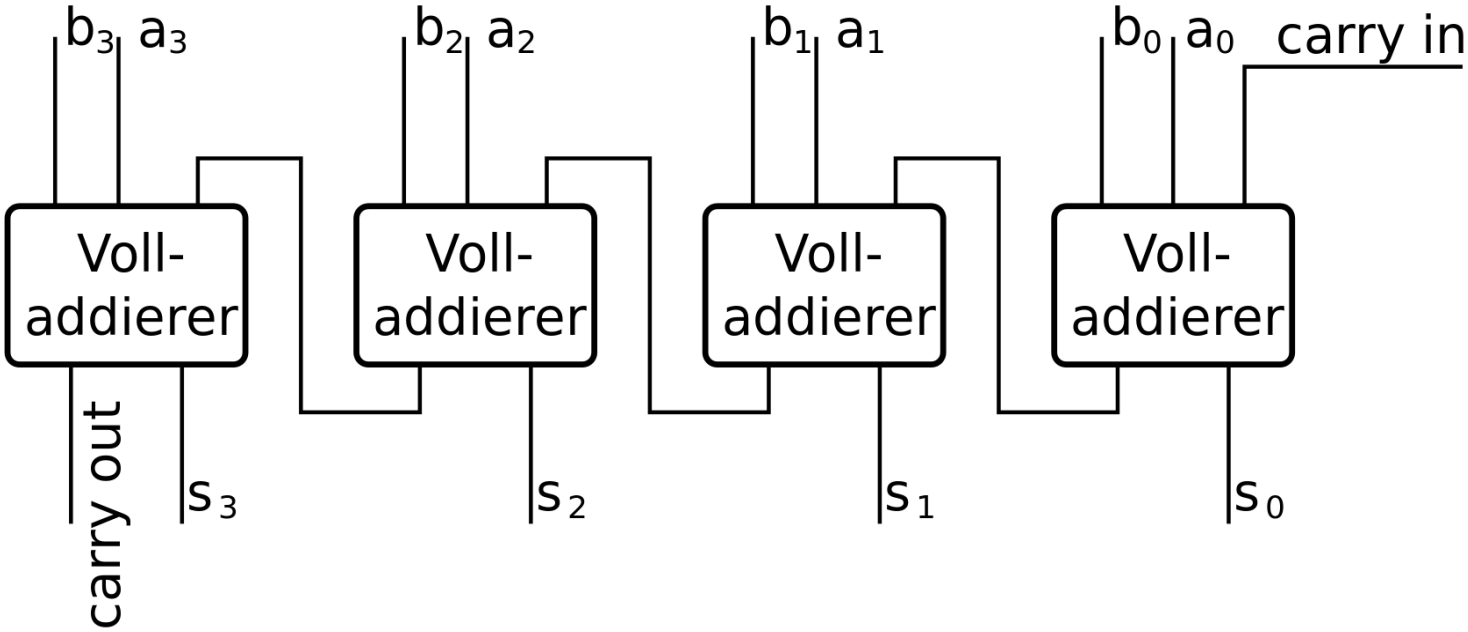
Carry-Ripple-Addierer

- Der **Carry-Ripple-Addierer** (von engl. carry – *Übertrag*, ripple – *rieseln*), auch *Ripple-Carry-Addierer* oder *ripple-through carry*, ist ein Addiernetz, dient also der Addition mehrstelliger Binärzahlen.

- Ein n-Bit-Carry-Ripple-Addierer kann zwei n-stellige Binärzahlen addieren, das Ergebnis hat n+1 Stellen. Das Schaltnetz hat damit 2n+1 (bzw. 2n ohne *Carry in*) Eingänge und n+1 Ausgänge.

- Er setzt sich aus n Volladdierern (bzw. aus n–1 Volladdierern und 1 Halbaddierer ohne *Carry in*) zusammen. Der Übertrags-Ausgang der Addierer wird jeweils an einen Eingang des nächsten Volladdierers angeschlossen. Der Übertrags-Ausgang des letzten Volladdierers bildet den (n+1)-ten Ausgang des Schaltnetzes.

Carry-Ripple-Addierer



Die Addition entspricht der EXOR - Verknüpfung: r = a xor b xor c, wobei a und b die i-ten Stellen der ersten und zweiten Summanden und c der Übertrag (engl. carry) ist.

Carry-Ripple-Addierer

- Gegeben sind zwei *n*-Bit-Zahlen *a* und *b*. Die Summe $s = a + b$ wird nach folgendem Schema berechnet, wobei die c_i die entstehenden Übertragsbits der jeweils vorherigen Stellen sind

$$
\begin{array}{cccccc}
 & a_{n-1} & \dots & a_1 & a_0 \\
\oplus & b_{n-1} & \dots & b_1 & b_0 \\
\oplus & c_n & c_{n-1} & \dots & c_1 & c_0 \\
\hline
= & s_n & s_{n-1} & \dots & s_1 & s_0
\end{array}
$$

- Die Operation $\oplus$ ist das logische Exklusiv-Oder (Xor).
- Im Folgenden werden ferner das Zeichen $\cdot$ für das logische Und sowie das Zeichen + für das logische Oder verwendet.
- Bei der Addition ist $c_0 = 0$; bei der Subtraktion werden die b_i invertiert und es ist $c_0 = 1$.

Carry-Ripple-Addierer

- Die Summenbits s_i können im Prinzip alle parallel berechnet werden, allerdings nur, wenn die Übertragsbits c_i bekannt sind:

  $$s_i = a_i \oplus b_i \oplus c_i \quad (i = 0, ..., n\text{-}1).$$

- Die Übertragsbits dagegen hängen vom jeweils vorhergehenden Übertragsbit ab:

  $$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i \quad (i = 0, ..., n\text{-}1).$$

- Die Schwierigkeit liegt also in der Berechnung der Übertragsbits.
  Jedes Übertragsbit c_i hängt indirekt von allen a_j und b_j mit $j < i$ ab.

- Am schwierigsten zu berechnen ist offenbar das Übertragsbit c_n, da es insgesamt von allen Stellen der Zahlen a und b abhängt.

Carry-Ripple-Addierer

- Da Volladdierer nicht unendlich schnell arbeiten, kann es zu Verzögerungen bei der Berechnung des Endergebnisses kommen, da der Volladdierer das korrekte Ergebnis erst dann ausgeben kann, wenn der vorhergehende Volladdierer das Übertragsbit geliefert hat.

- Im schlechtesten Fall führt die Addition $a_0 + b_0$ zu einem Übertrag, und für alle $i > 0$ gilt: $a_i + b_i \geq 1$.

- Dann muss das Übertragsbit durch das gesamte Addiernetz wandern, bevor das richtige Ergebnis ausgegeben wird (*Übertragspropagation*).

- Um diese langen Laufzeiten zu vermeiden, wurden beschleunigte Addiernetze entwickelt

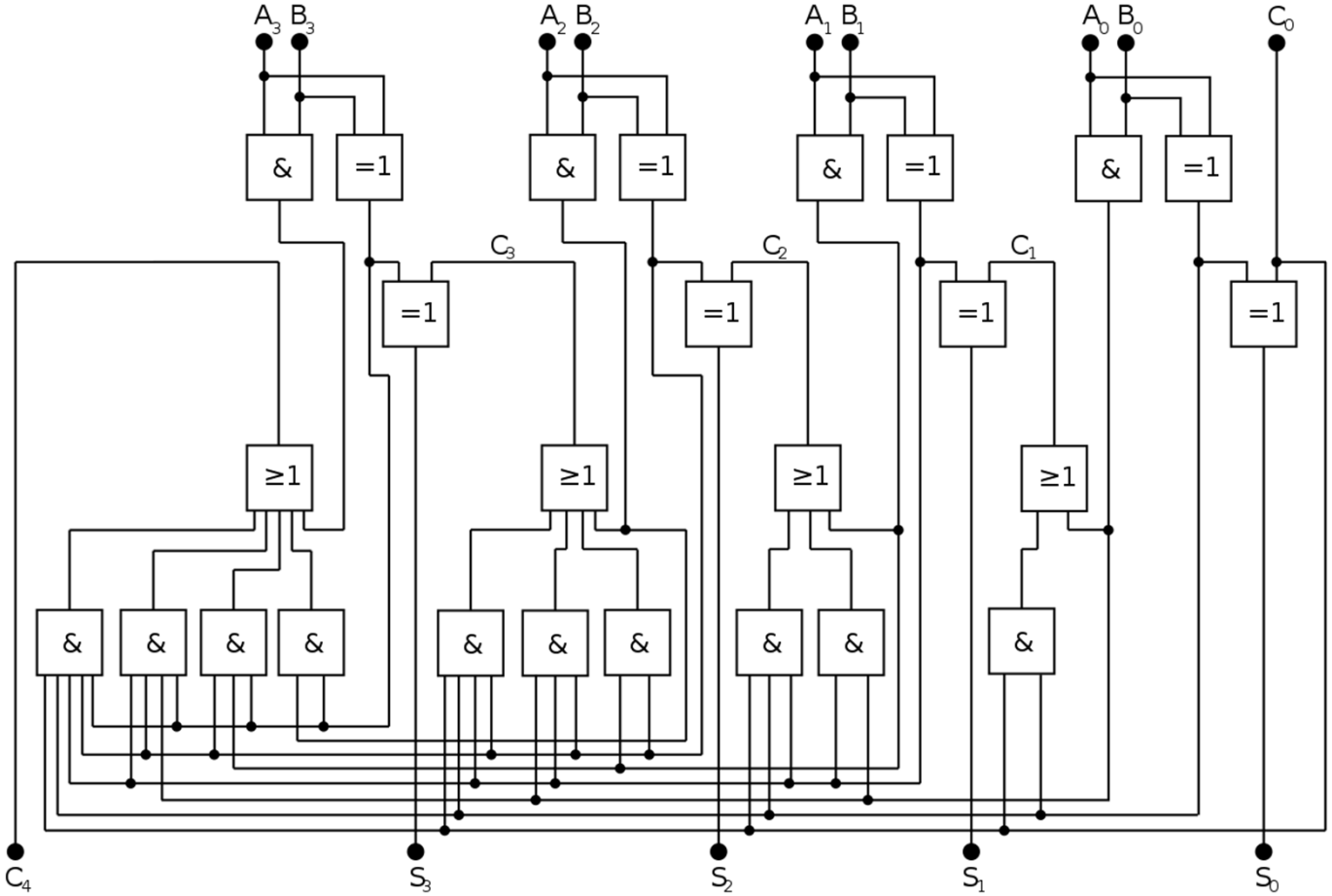
Carry-Lookahead-Addierer

DHBW
Duale Hochschule
Baden-Württemberg

Paralleladdierer mit Übertragsvorausberechnung

- Der **Paralleladdierer mit Übertragsvorausberechnung** bzw. **Carry-Look-Ahead-Addierer** (kurz: CLA-Addierer) ist eine logische Schaltung zur Addition mehrstelliger Binärzahlen.
- Der CLA-Addierer addiert zwei n-stellige Binärzahlen, verfügt also über 2·n Eingänge, sowie in der Regel über einen weiteren Übertragseingang.
- Da das Ergebnis einen etwaigen Übertrag enthalten kann, gibt es n+1 Ausgänge.
- Der Vorteil des CLA-Addierers ist, dass die Verzögerung der Schaltung nur logarithmisch zur Zahl seiner Eingänge ist, bei zugleich nur linearer Zahl an Logikgattern gemessen an der Zahl seiner Eingänge.

Carry-Lookahead-Addierer



DHBW
Duale Hochschule
Baden-Württemberg

Vorteile von Carry Look-Ahead Addierer

- Bei diesem Addierer wird die Ausbreitungsverzögerung verringert.
- Die Übertragsausgabe in jeder Stufe hängt nur vom anfänglichen Übertragsbit der Anfangsstufe ab.
- Mit diesem Addierer können die Zwischenergebnisse berechnet werden.
- Dieser Addierer ist der schnellste Addierer, der für die Berechnung verwendet wird.

Festkommamultiplizierer

Festkommamultiplizierer

- Die binäre Multiplikation verläuft analog wie im dezimalen System, und kann in digitalen Schaltungen als eine Abfolge von Additionen und Schieboperationen realisiert werden.

- In der folgenden Schaltung ist ein vorzeichenloser, paralleler Multiplizierer (MAC) für zwei je vier Bit breite Zahlen *X* und *Y* und dem Summanden *K* mit Volladdierern dargestellt.

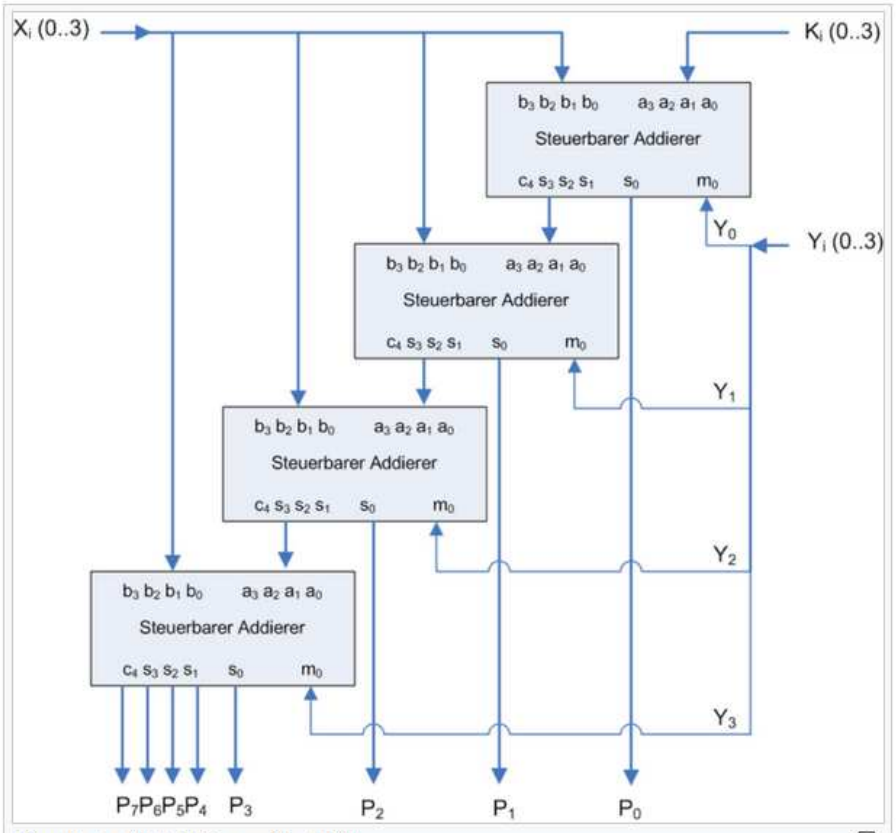
- Die acht Ausgabebits *P* werden in der kombinatorischen Logik mit folgender Gleichung gebildet:

$$P = X \cdot Y + K$$

Festkommamultiplizierer

Der Vorgang der Multiplikation gestaltet sich dabei nach folgendem Schema, die vier Eingangsbits *K* sind der Einfachheit wegen auf 0 gesetzt:

```
      1011    (X, entspricht
dezimal der Zahl 11)
    · 1110    (Y, entspricht
dezimal der Zahl 14)
    ------
 +    0000    (1011 · 0)
 +   1011     (1011 · 1, um eine
Stelle nach links verschoben)
 +  1011      (1011 · 1, um zwei
Stellen nach links verschoben)
 + 1011       (1011 · 1, um drei
Stellen nach links verschoben)
 ---------
  10011010    (P, entspricht
dezimal 154)
```



Paralleler Multiplizierer für 4 Bit.
s_i = Summanden
c_i = Carry-Output
m_0 = Input Enable b_i

Festkommamultiplizierer

- Dieser einfache Multiplizierer lässt sich aus einzelnen Volladdierern und die Schiebeoperation durch direkte Verschaltung realisieren.
- Die binären Stellen des Produktes *P* sind gleich der Summe der Stellen der beiden Faktoren *X* und *Y*.
- Ein in der Position fixer Kommapunkt wird generell nicht schaltungstechnisch abgebildet, sondern die Position des Kommapunktes im Produkt ergibt sich aus der Summe der Stellen nach dem Komma der beiden Eingangsfaktoren.
- Im obigen Beispiel ist bei beiden Faktoren die Stellenanzahl hinter dem Komma null, wodurch auch im Produkt der Kommapunkt rechts der letzten Stelle zu liegen kommt.

Zählschaltungen

Zählschaltungen – Grundlagen

- Ebenfalls häufige Anwendung von Flipflops
- Gezählt werden entweder unregelmäßige Impulse (z. B. von einer Lichtschranke) oder die regelmäßigen Impulse eines Taktgenerators (z. B. Uhrenquarz mit 32.768 Hz)
- Binäre Zählschaltungen meist: JK-, D- oder T-Flipflops
- Zählschaltungen können asynchron oder synchron betrieben werden.

Zählschaltungen

- Werte der Ausgänge der Flipflops = Zählerstand (direkt oder hinter einem Umcodierschaltnetz)
- Wichtig: Betriebsbeginn muss ein definierter Anfangszustand sein
- Daher: oft Flipflops mit zusätzlich asynchrone Setz- bzw. Rücksetzeingänge
- Zählschaltungen = flankengesteuerte – oder Master-Slave-Flipflops
  → hohe Stabilität

DHBW
Duale Hochschule
Baden-Württemberg

Asynchrone Zählschaltung:

– Kennzeichen: zählende Impulse werden nur dem Steuereingang des ersten Flipflops zugeführt;

– Impulse an den Steuereingängen der folgenden Flipflops werden jeweils durch das davor liegende Flipflop erzeugt.

– Informationsfluss = seriell = langsam.

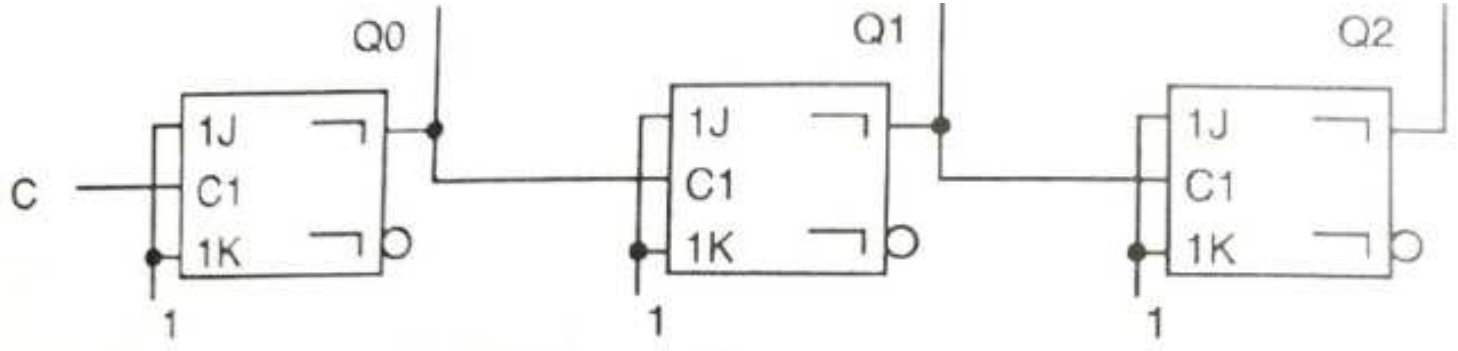
– Flipflops ändern ihre Ausgangswerte nacheinander (asynchron).

– Folge: zwischenzeitig falsche Zählerstände

Zählschaltungen

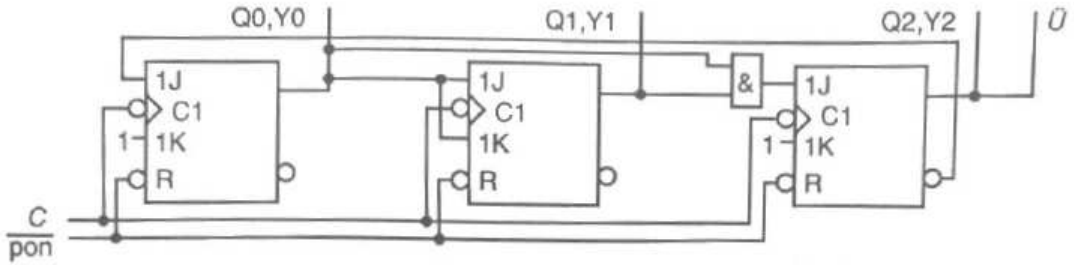
– **Vorteile:**
  - Dualcode-Zähler sind einfach zu entwerfen
  - geringer Schaltungsaufwand

– **Nachteile:**
  - alle Nachteile asynchroner Schaltwerke → nahezu kein Einsatz



Synchrone Zählschaltung:

- Kennzeichen: zählende Impulse werden parallel an die Steuereingänge aller Flipflops (beliebigen Typs!) gelegt.
- Gatter vor den Dateneingängen der einzelnen Flipflops = bei jedem neuen Zählimpuls ändert der Zähler seinen Zählerstand
- Änderungen der Ausganswerte erfolgen stets gleichzeitig (synchron)
- Allgemeine synchrone Schaltwerke haben oft sehr viele Eingangsgrößen, die das Verhalten steuern.



Synchroner mod-5 Zähler (vorwärts)

DHBW
Duale Hochschule
Baden-Württemberg

Synchrone Zählschaltung

Allgemeingültige Entwurfsmethodik

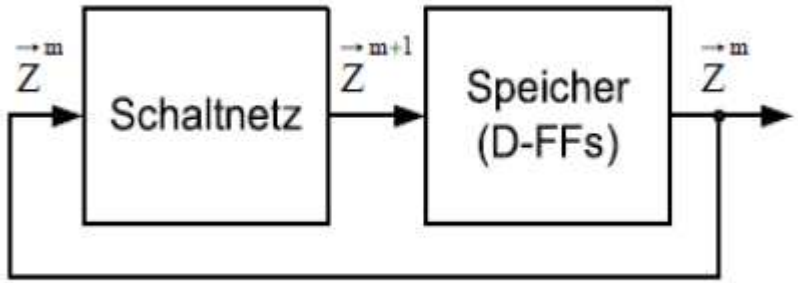
- Synchrone Zähler = einfache synchrone, endliche, binäre Automaten
- Entweder flankengesteuerte Einspeicher-Flipflops oder beliebig gesteuerten Master-Slave-Flipflops beliebigen Typs
- Zählerstand = Vektor der Ausgangswerte Q = Zustandsvektor → Medwedjew-Automaten
- Zähler kann aber auch ein Schaltnetz am Ausgang enthalten zur Umcodierung des Zustandsvektors in den Zählerstand → Moore-Automat
- Abfolge der Zustände abhängig von gewähltem Code für den Zählerstand und die Anzahl der Stellen, die das Schaltwerk beim Eintreffen der zu zählenden Impulse durchläuft.

Zählschaltungen

- Fasst man diese Impulse als Taktsignal auf und lässt die in der Regel notwendige Initialisierung des Zählers außer Acht, so kann ein solcher Zähler als autonomer Automat aufgefasst werden.

- Ein Schaltnetz kann aber auch den aktuellen Zustandsvektor so „modulieren“, dass beim nächsten Zählimpuls der richtige nächste Zählerstand als neuer Zustandsvektor erzeugt wird

- Schaltnetz ist dann meist eine Vielzahl von kleinen Schaltnetzen jeweils vor den Eingängen der Flipflops



Folgende Tabelle:

Zusammenstellung der Eingangssignale für verschiedene Flipflop-Typen, um von Ausgangswert Q_m zu Q_{m+1} zu kommen:

Geforderter Übergang des Flipflops		Notwendige Eingangsbelegung beim					
		SR-FF		JK-FF		D-FF	T-FF
Q^m	Q^{m+1}	S^m	R^m	J^m	K^m	D^m	T^m
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	0	1
1	1	X	0	X	0	1	0

DHBW
Duale Hochschule
Baden-Württemberg

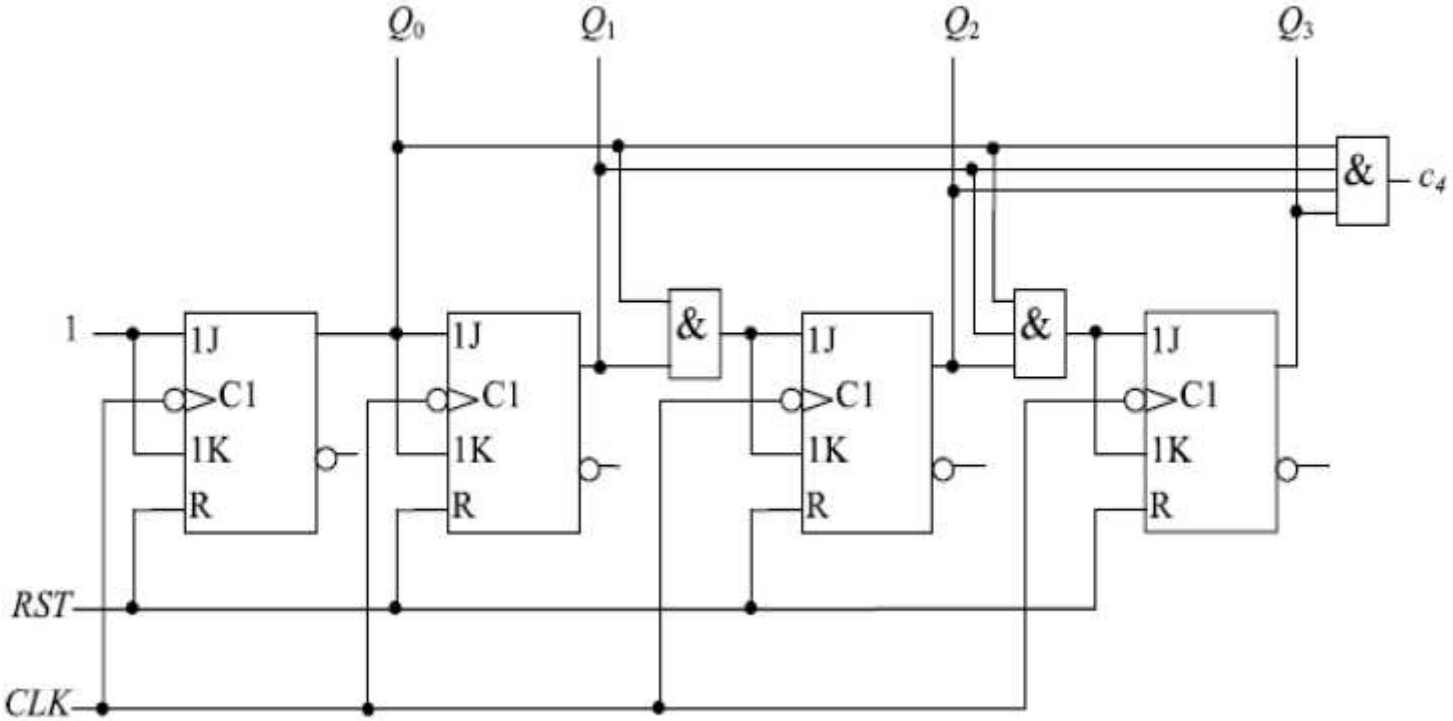
Synchrone Zählschaltung - 4-Bit-Dualzähler

- Es soll ein 4-Bit-Dualzähler mit vier JK-Flipflops aufgebaut werden.
- Er soll ein Übertragssignal c liefern, wenn er von 1111 nach 0000 schaltet.

Q_3^m	Q_2^m	Q_1^m	Q_0^m	Q_3^{m+1}	Q_2^{m+1}	Q_1^{m+1}	Q_0^{m+1}	Q_3^m	Q_2^m	Q_1^m	Q_0^m	Q_3^{m+1}	Q_2^{m+1}	Q_1^{m+1}	Q_0^{m+1}
0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0	1	0	0	1	1	0	1	0
0	0	1	0	0	0	1	1	1	0	1	0	1	0	1	1
0	0	1	1	0	1	0	0	1	0	1	1	1	1	0	0
0	1	0	0	0	1	0	1	1	1	0	0	1	1	0	1
0	1	0	1	0	1	1	0	1	1	0	1	1	1	1	0
0	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0

Zählschaltungen

Schaltbild:



Komparator

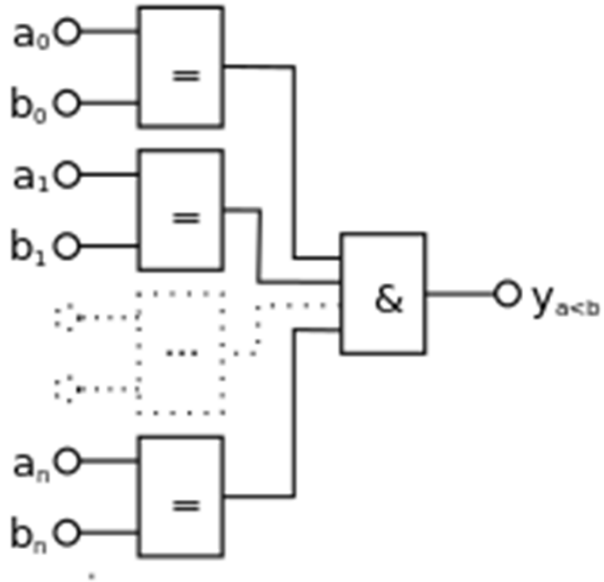
Komparator

Ein **Komparator** in der Digitaltechnik ist ein elektronischer Schaltkreis, der zwei# digitale Werte vergleicht.

Identitäts-Komparatoren

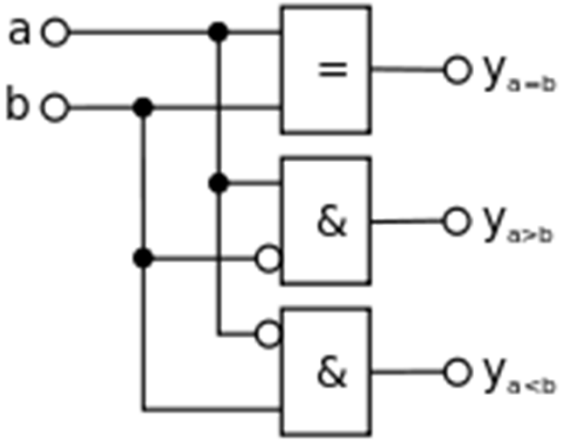
- Identitäts-Komparatoren testen zwei Bits auf Gleichheit, was mit Hilfe eines XNOR-Gatters erfolgt.

- Für den Vergleich von Bytes werden je zwei gleichwertige Bits miteinander verglichen und das Ergebnis mit einem Und-Gatter verknüpft.



Größen-Komparator

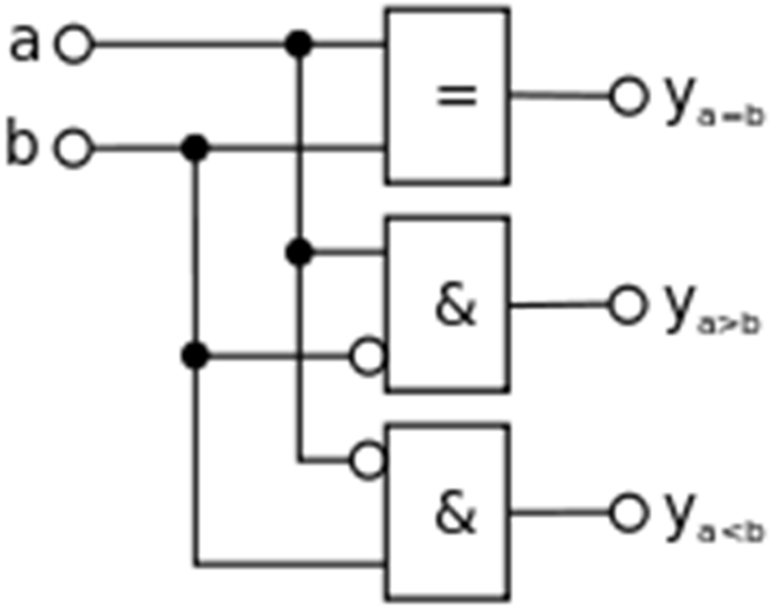
- Größen-Komparatoren können zusätzlich zur Gleichheit auch auf die Relationen Größer und Kleiner testen.
- Um den Größenvergleich durchführen zu können müssen die beiden Zahlen auf die gleiche Weise codiert sein.
- Zusätzlich muss der Größen-Komparator auf den jeweils verwendeten Code ausgelegt werden.



Komparator

Wahrheitstabelle für 1-Bit-Größen-Komparator

a	*b*		$y_{a>b}$	$y_{a=b}$	$y_{a<b}$
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0



DHBW
Duale Hochschule
Baden-Württemberg

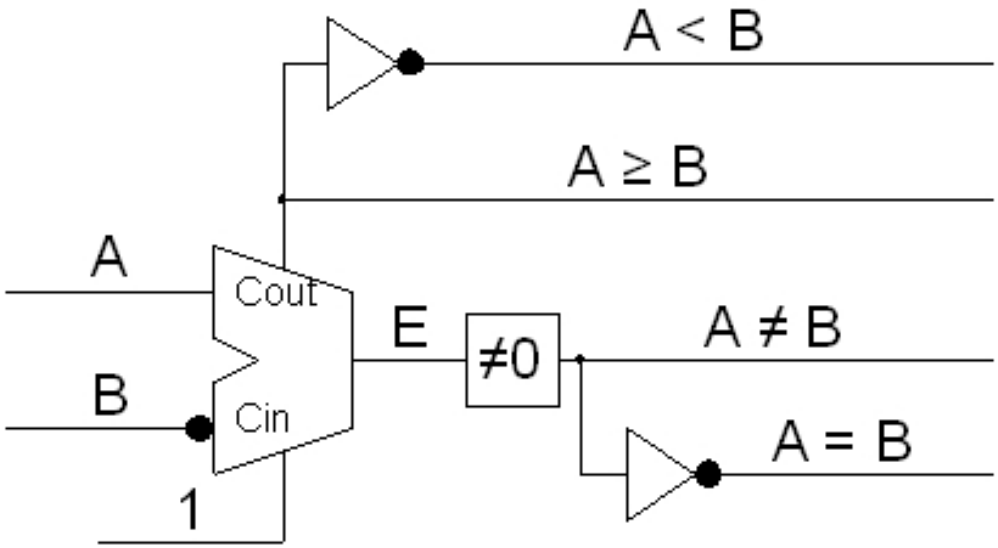
n-bit-Größen-Komparator

- Ein n-bit-Größen-Komparator beruht auf der Grundlage, dass die Differenz aus den zu vergleichenden Größen gebildet wird und das Ergebnis auf 0, <0, >0 geprüft wird.
- Das Addierwerk beruht im Dualcode auf der Addition des Zweierkomplement, also (-B) ist dasselbe wie (+!B + 1).
- Bei der Addition einer Zahl mit ihrer invertierten Zahl (z. B. 1001101 + 0110010 = 1111111) sind im Ergebnis alle Bits 1.
- Wird eine Zahl von sich selber abgezogen, (A – A = A + (!A + 1) = 0, carry=1) ist das Ergebnis 0, mit Übertrag 1.

Komparator

Soll A mit B verglichen werden, dann gilt:

Bedingung	Hinweis	Äquivalent	Zwischenergebnis	Carry	Ergebnis
A == B	A = B	B +!B + 1	b'1..11 + 1	1	0
A > B	A = B + d	B + d +!B + 1	b'1..11 + 1 + d	1	d
A < B	A = B - d	B - d +!B + 1	b1..11 + 1 -d	0	-d

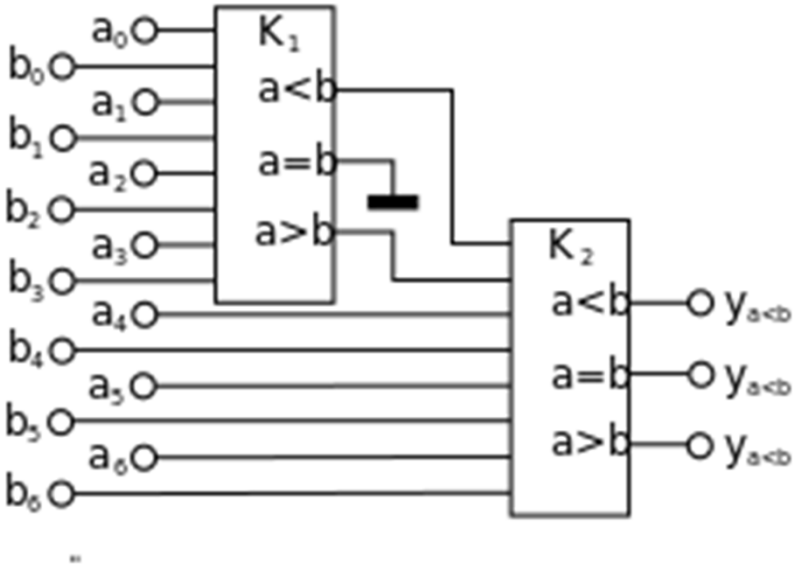


DHBW
Duale Hochschule
Baden-Württemberg

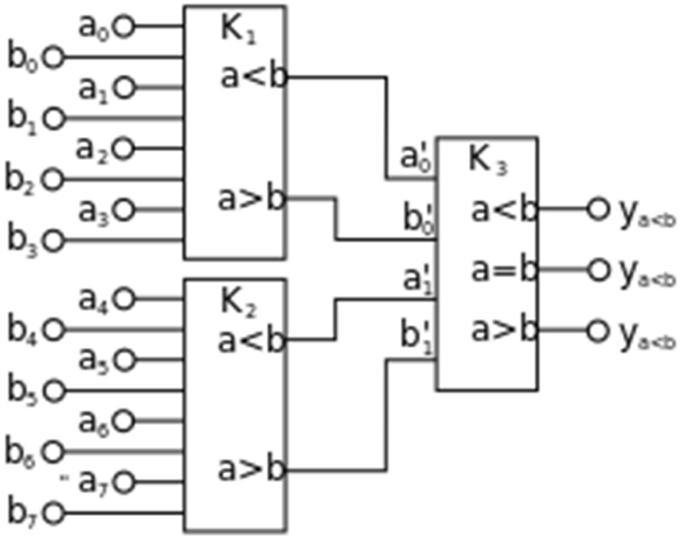
Erweiterung

- Zum Vergleich von Bytes, die mehr Stellen aufweisen, als dies vom Komparator-Baustein vorgegeben ist, kann man mehrere Komparatoren seriell oder parallel verschalten.
- Die parallele Lösung hat hierbei bei Bytes mit vielen Stellen den Vorteil einer geringeren Latenz, wodurch eine höhere Geschwindigkeit resultiert.
- Der serielle Aufbau empfiehlt sich lediglich, wenn dadurch weniger Komparator-Bausteine verwendet werden müssen

Komparator



Serieller Aufbau eines 7-Bit-Byte-Größen-Komparators



Paralleler Aufbau eines 8-Bit-Byte-Größen-Komparators

E N D E