Multiplikation vorzeichenbehafteter Zahlen

Zur Multiplikation vorzeichenbehafteter Zahlen (2er-Komplement) kann auf die Schaltung für vorzeichenlose Multiplikation zurückgegriffen werden, wenn negative Zahlen zuerst negiert werden, das Vorzeichen separat berechnet wird (XOR) und das Ergebnis ggf. noch invertiert wird.
Es gibt jedoch auch noch andere Verfahren wie z.B. den sog. *Baugh-Wooley*-Multiplizierer. Dieser ist sehr ähnlich wie der kombinatorische Multiplizierer für vorzeichenbehaftete Zahlen aufgebaut, verwendet jedoch an einigen Stellen ein NICHT-UND-Gatter statt eines UND-Gatters sowie einen zusätzlichen Halbaddierer für die höherwertigste Ergebnis-Stelle.

Multiplikation von Gleitkomma-Zahlen

- Zur Multiplikation von Gleitkommazahlen müssen die Mantissen inkl. führender ‘‘1,’’ als Festkommazahlen multipliziert werden.
- Die Exponenten werden addiert. Der Offset ‘‘k’’ ist nach der Addition doppelt berücksichtigt und muss deswegen vom Ergebnis noch einmal subtrahiert werden.
- Zur Re-Normalisierung wird die Ergebnis-Mantisse nach rechts geschoben und zum Exponenten die Anzahl der geschobenen Stellen addiert.

3.9 Subtraktion

Allgemein

Bezeichnungen: Minuend - Subtrahend = Differenz
Die Subtraktion zweier Zahlen wird stellenweise ausgeführt. Dabei kann es vorkommen, dass ein größerer Wert von einem kleineren Wert subtrahiert werden muss. Um dies zu bewerkstelligen, kann aus der nachfolgenden Stelle ein Wert geborgt werden.

Beispiel:

$$\begin{array}{cccccc} & & \multicolumn{2}{c}{11} & & \\ & 2+ & 1 & 1+ & 1 & 11 \\ & 4 & \cancel{3} & \cancel{2} & \cancel{1} & \\ - & 1 & 2 & 3 & 4 & \\ \hline = & 3 & 0 & 8 & 7 & \end{array}$$

- 4 und wieviel ist 1? ⇒ geht nicht ⇒ 1 von 10-er Stelle borgen ⇒ aus 1 wird 11
- 4 und wieviel ist 11? ⇒ 7
- Durch das Borgen steht an der Zehner-Stelle jetzt nur noch eine 1 statt einer 2
- 3 und wieviel ist 1? ⇒ geht nicht ⇒ 1 von 100-er Stelle borgen ⇒ aus 1 an der Zehner-Stelle wird 11
- 3 und wieviel ist 11? ⇒ 8
- Durch das Borgen steht an der Hunderter-Stelle jetzt nur noch eine 2 statt einer 3
- 2 und wieviel ist 2? ⇒ 0
- 1 und wieviel ist 4 ⇒ 1

Statt beim Borgen die Minuenden-Stellen zu verkleinern, kann die Subtrahenden-Stelle vergrößert werden (wie Übertrag).

$$\begin{array}{ccccc} & 4 & 3 & 2 & 1 \\ - & 1 & 2 & 3 & 4 \\ & {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 1} & \\ \hline = & 3 & 0 & 8 & 7 \end{array}$$

Das Ergebnis ist das gleiche, da die Differenz zwischen Minuenden-Stelle und Subtrahenden-Stelle gleich bleibt. Beim Borgen über mehrere Stellen hinweg kann einem dieses Vorgehen jedoch leichter fallen.

a) Subtrahieren Sie 11 - 6 = 5 im Binärsystem bei einer Wortbreite $n = 4$.

b) Subtrahieren Sie 12 - 5 = 7 im Binärsystem bei einer Wortbreite $n = 4$.

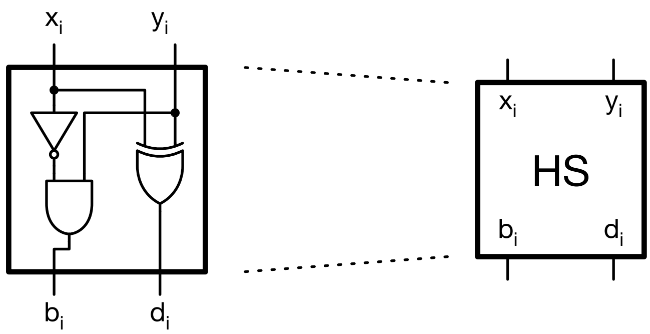
T c) Subtrahieren Sie 14 - 11 = 3 im Binärsystem bei einer Wortbreite $n = 4$.

Halb-Subtrahierer

Ein Halb-Subtrahierer ist ein Schaltung, die ein Eingangs-Bit y_i von einem Eingangs-Bit x_i subtrahiert. Das Ergebnis ist ein Differenz-Bit d_i und ein Borge-Bit b_i (b = borgen = engl. borrow).

Eingang x_i	Eingang y_i	Borgen b_i	Differenz d_i
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Die Differenz d_i entspricht der XOR-Verknüpfung der Eingänge; b_i hat den Wert 1, wenn der Minuend 0 ist und der Subtrahend 1 ist.



Halbsubtrahierer können Binärzahlen nur *halb* subtrahieren: Der Halbsubtrahierer an Stelle i erkennt zwar, ob er ein Bit von Stelle $i + 1$ borgen musste, kann jedoch selbst nicht berücksichtigen, ob der Halbsubtrahierer an Stelle $i - 1$ von ihm selbst ein Bit borgen musste.

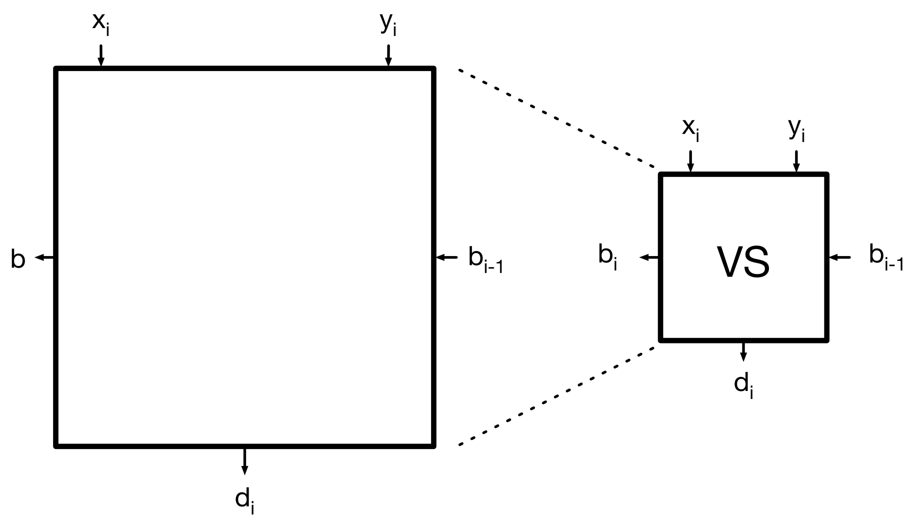
Voll-Subtrahierer

Im Gegensatz zum Halbsubtrahierer kann ein Vollsubtrahierer berücksichtigen, ob die vorangegangene Stelle $i-1$ ein Bit borgen musste.

a) Vervollständigen Sie nachfolgende Wertetabelle eines Vollsubtrahierers.

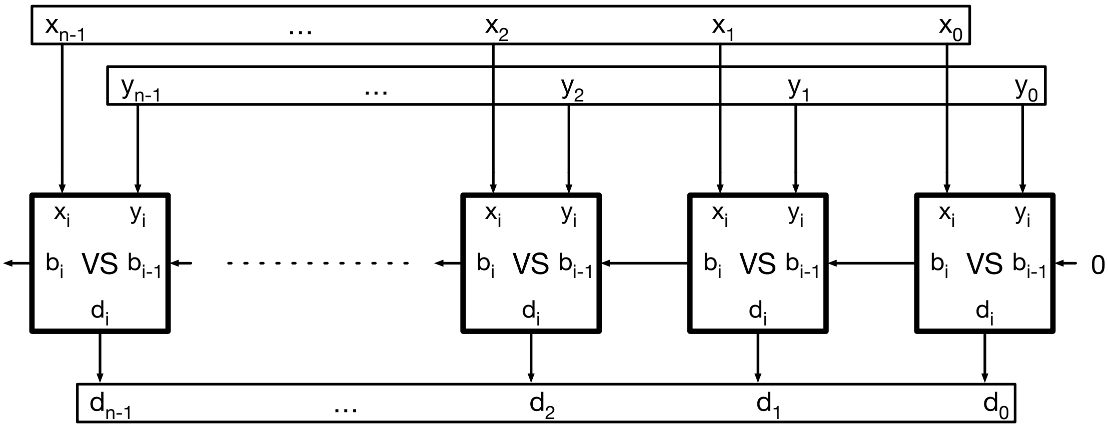
x_i	y_i	b_{i-1}	b_i	d_i
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

b) Tragen Sie in nachfolgende Abbildung (links) eine Implementierung einer Vollsubtrahierer-Schaltung ein.

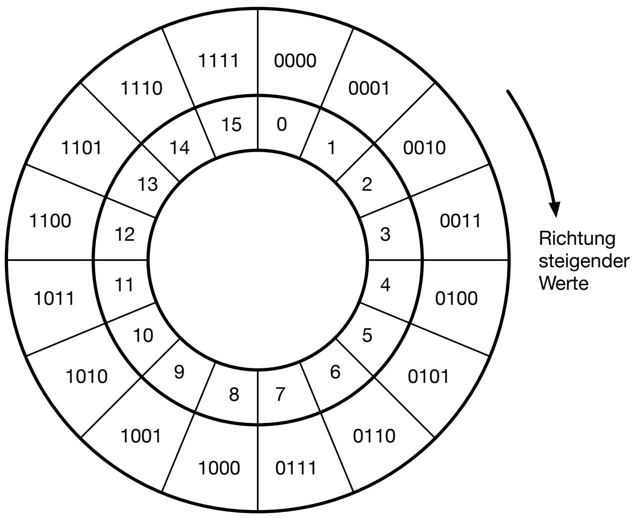


Ripple-Borrow-Subtrahierer

Beim Ripple-Borrow Subtrahierer werden n Vollsubtrahierer so verschaltet, dass sich damit die Differenz $d = x - y$ zweier n Bit breiter Zahlen berechnen lässt.

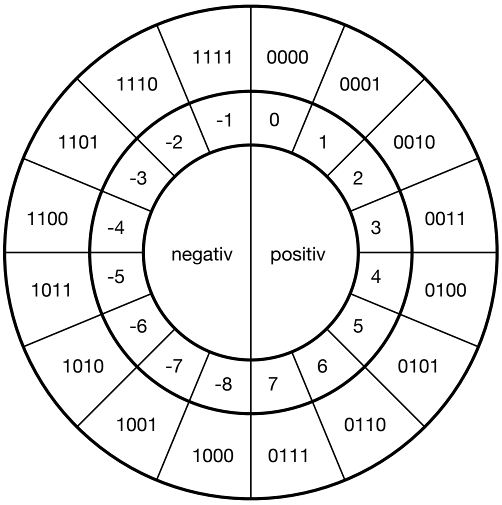


Betrachten Sie den Zahlenring für vorzeichenlose Zahlen.



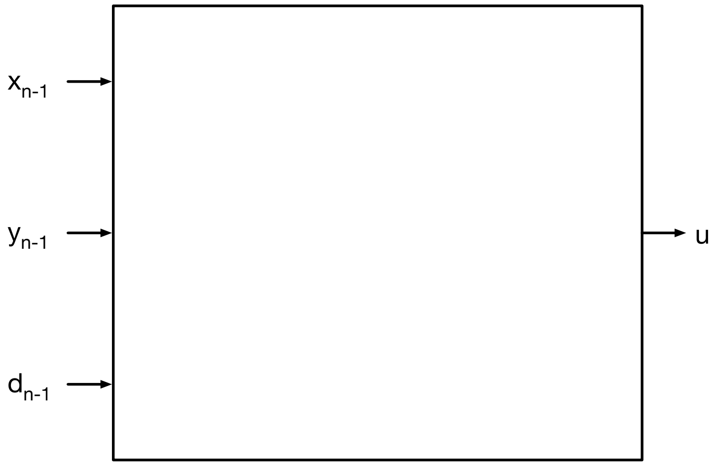
a) Nehmen Sie an, die Eingangswerte des entworfenen Ripple-Borrow-Subtrahierers sind vorzeichenlos. Welches Zahlenformat hat die Differenz d? Welche Funktion hat das ‘‘Borrow Out’’?

Betrachten Sie den Zahlenring für Zahlen im Zweier-Komplement:



b) Funktioniert der Subtrahierer auch mit dem Zweier-Komplement? Wenn ja: Wie kann man einen Überlauf feststellen? Wenn nein: Warum nicht?

c) Tragen Sie in nachfolgende Abbildung eine Schaltung ein, die einen Überlauf von Zahlen im Zweierkomplement feststellt.



3.10 Division

Allgemein

Bei der Division gilt allgemein:

Dividend / Divisor = Quotient + Rest

Division zur Basis 10, wie in der Schule gelernt:

1. Runde

9 8 7 6 : 0 0 5 4 = 0

1. Teildividend = 9

Passt 54 in 9? Nein, d.h. 0 mal.

2. Runde

9 8 7 6 : 0 0 5 4 = 0 1

2. Teildividend = 98

Passt 54 in 98? Ja ⇒ Wie oft?

98 - 54 = 44 (1 mal)
44 - 54 = -10 (negativ ⇒ bleibt bei 1 mal)

3. Runde

4 4 7 6 : 0 0 5 4 = 0 1 8

3. Teildividend = 447

Passt 54 in 447? Ja ⇒ Wie oft?

447 - 54 = 393 (1 mal)
393 - 54 = 339 (2 mal)
339 - 54 = 285 (3 mal)
285 - 54 = 231 (4 mal)
231 - 54 = 177 (5 mal)
177 - 54 = 123 (6 mal)
123 - 54 = 069 (7 mal)
069 - 54 = 015 (8 mal)
015 - 54 = -039 (negativ ⇒ bleibt bei 8 mal)

4. Runde

0 1 5 6 : 0 0 5 4 = 0 1 8 2 Rest 4 8

4. Teildividend = 156

Passt 54 in 156? Ja ⇒ Wie oft?

156 - 54 = 102 (1 mal)
102 - 54 = 048 (2 mal)
048 - 54 = -006 (negativ ⇒ bleibt bei 2 mal)

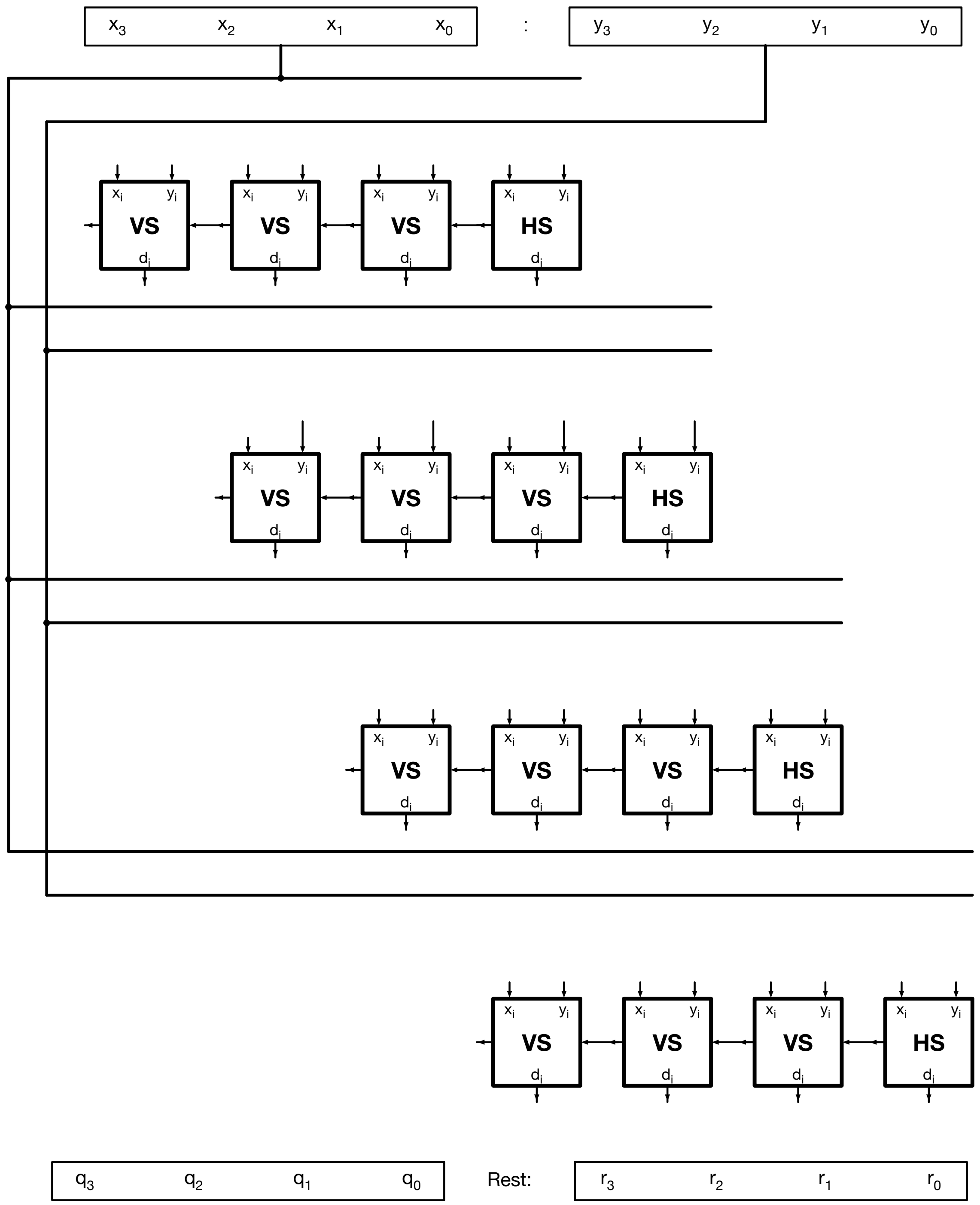
Die Division zur Basis 2 folgt demselben Prinzip wie die Division zur Basis 10. Da der Teildividend jedoch nur 0 oder 1 mal in den Divisor passen kann, ist die Bestimmung der jeweiligen Quotienten-Stelle wesentlich einfacher.

a) Berechnen Sie binär vorzeichenlos für $n = 4$ die Division $13/4 = 3$ Rest 1.

T b) Berechnen Sie binär vorzeichenlos für $n = 4$ die Division $10/3 = 3$ Rest 1.

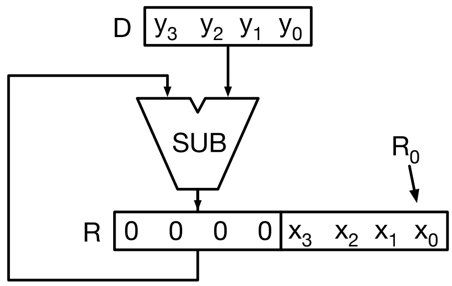
Kombinatorischer Dividierer

a) Vervollständigen Sie nachfolgende Abbildung um geeignete Bauelemente und Verbindungen zu einer Schaltung, die zwei vorzeichenlose 4 Bit breite Zahlen zu einem Quotienten q und einem Rest r dividiert.



Sequentieller Dividierer

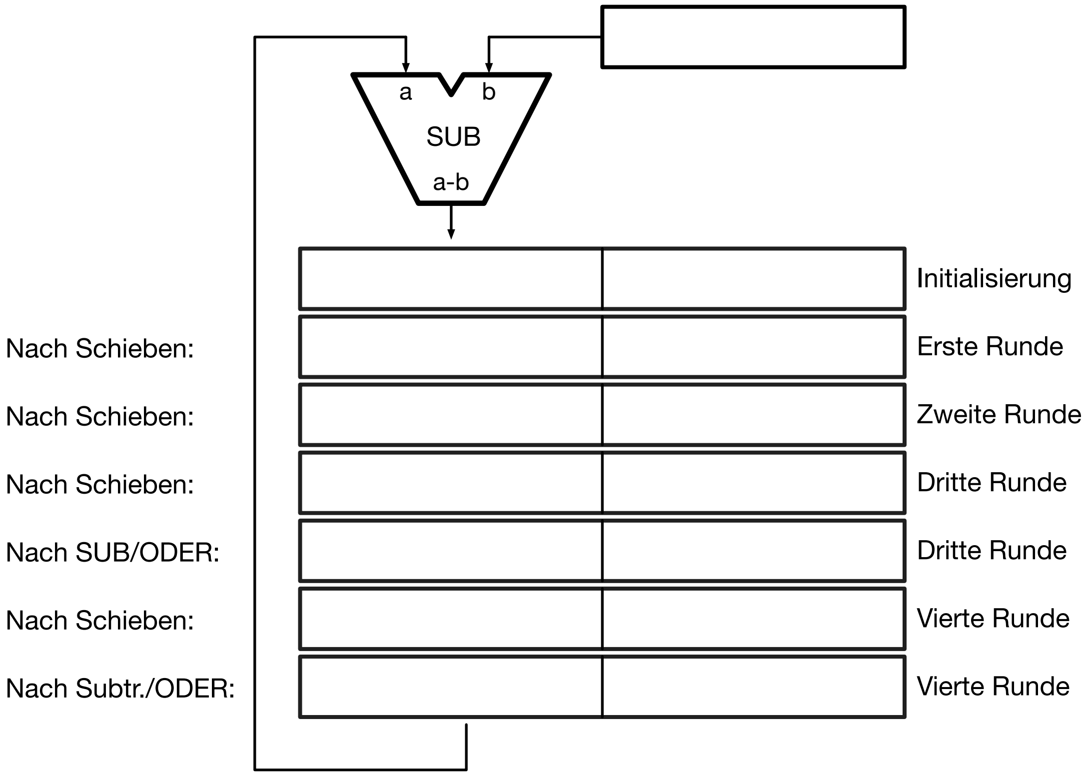
Nachfolgende Abbildung skizziert eine sequentielle Schaltung, die zur Division (hier: x/y) vorzeichenloser Zahlen der Wortbreite $n = 4$ verwendet werden kann.



Das Divisor-Register D ist $n = 4$ Bit breit, das Rest-Register R ist $2n = 8$ Bit breit.

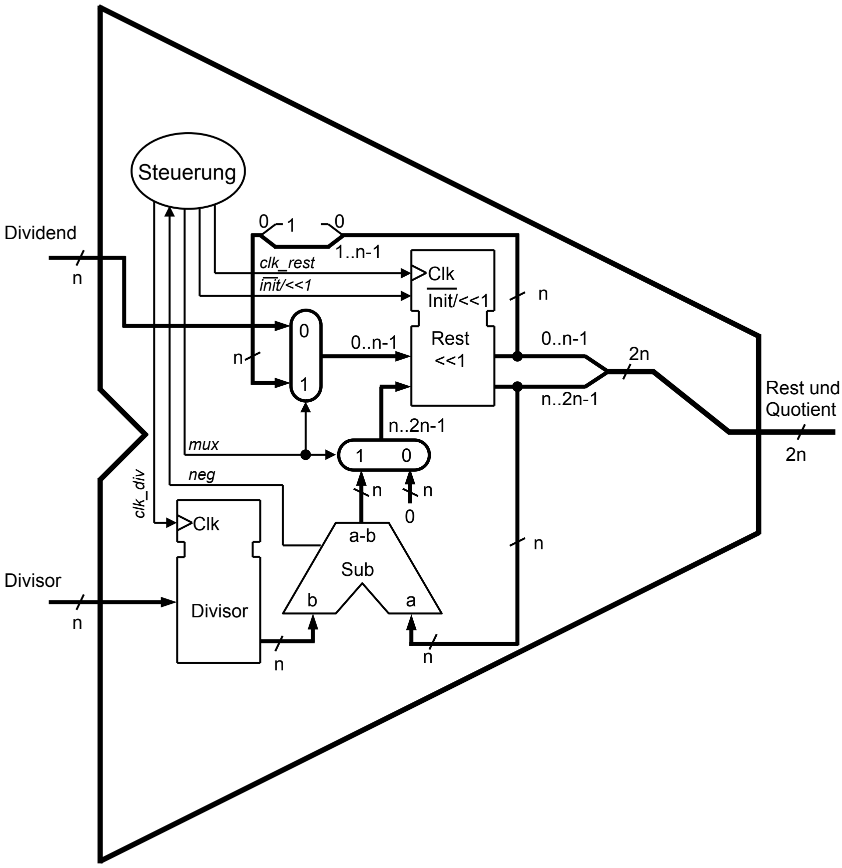
- Zuerst wird der Dividend in der rechten Hälfte des Rest-Registers R abgelegt; die linke Hälfte wird mit 0 initialisiert
- Der Divisor wird im Divisor-Register D abgelegt
- Anschließend wird iterativ $n = 4$ mal folgendes durchgeführt:
  - Rest-Register R um eine Stelle nach links schieben, dabei von rechts mit Nullen auffüllen.
  - Der Subtrahierer bestimmt mittels Subtraktion $R_{2n-1\,\ldots\,n} - D$, ob der Divisor D in den Teil-Dividenden $R_{2n-1\,\ldots\,n}$ passt.
  - Ist das Ergebnis der Subtraktion positiv, d.h. hat der Divisor in den Teil-Dividenden reingepasst,
    - wird R_0 auf 1 gesetzt und
    - das Ergebnis der Subtraktion (der Rest) in $R_{2n-1\,\ldots\,n}$ übernommen.
- Der Quotient findet sich in der rechten Hälfte des Rest-Regstiers, d.h. $R_{n-1\,\ldots\,0}$, der Divisions-Rest in der linken Hälfte, d.h. $R_{2n-1\,\ldots\,n}$.

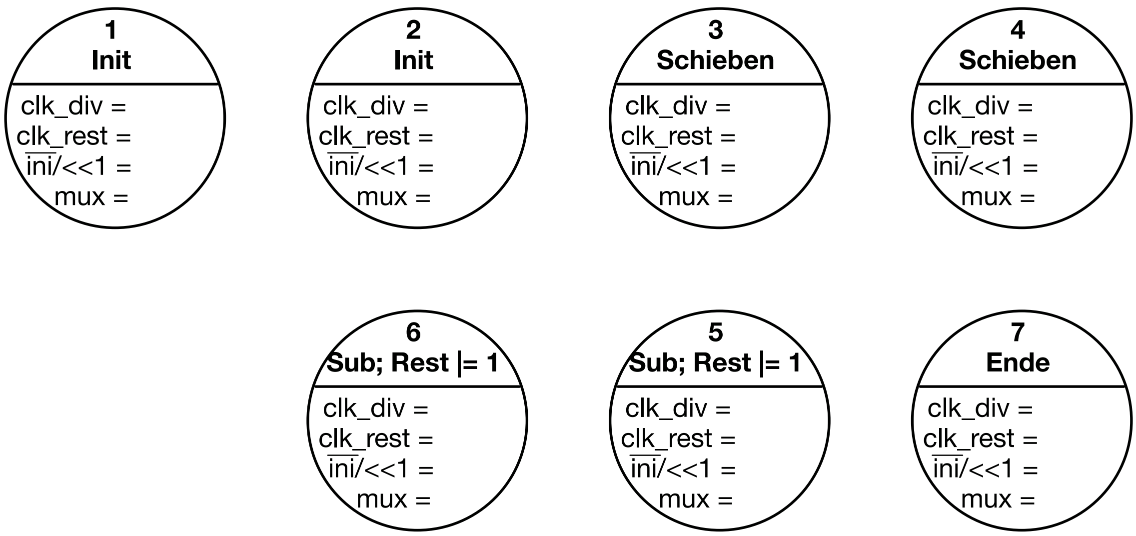
a) Tragen Sie in folgende Abbildung für $n = 4$ die Registerinhalte ein, die sich für die Division $13 : 4 = 3$ Rest 1 ergeben.



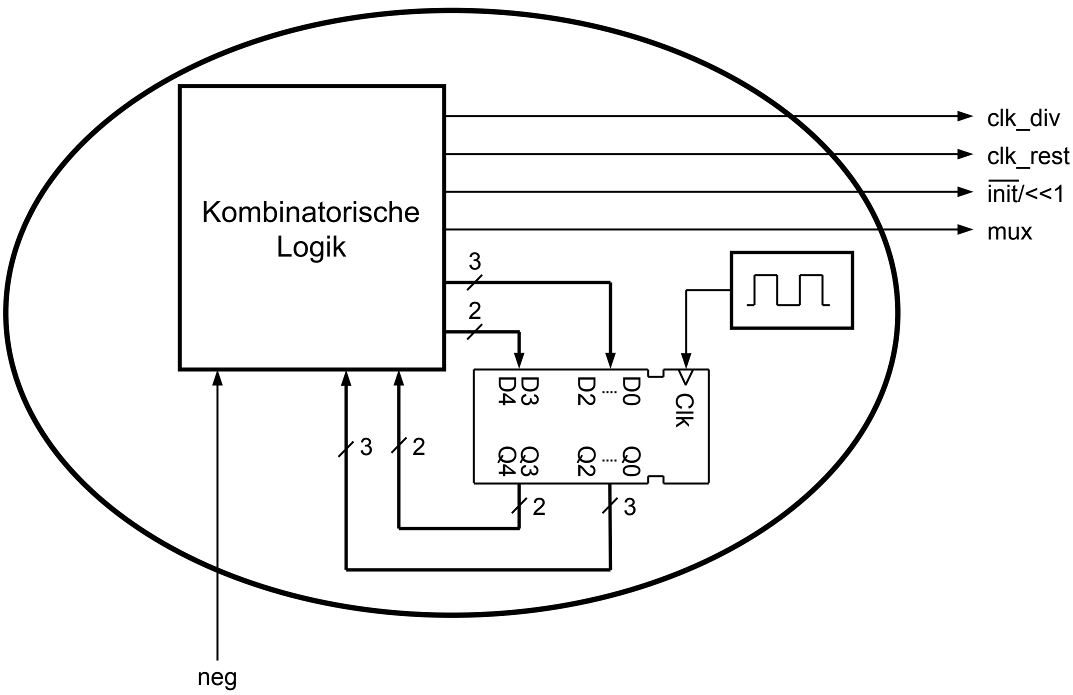
Nachfolgende Abbildungen zeigen eine Schaltung, welche die sequentielle Division implementiert, sowie den zugehörigen Zustandsautomaten.

T b) Tragen Sie in den Zustands-Automaten geeignete Übergänge und Ausgangssignale so ein, dass der Zustandsautomat die Schaltung in gewünschter Weise steuert.


Steuerung
Dividend
n
0
1
0
1..n-1
clk_rest
$\overline{init}$/<<1
Clk
Init/<<1
n
0
n
1
0..n-1
Rest
<<1
0..n-1
2n
n..2n-1
Rest und
Quotient
2n
n..2n-1
mux
1
0
neg
n
n
0
clk_div
Clk
a-b
Sub
n
Divisor
n
Divisor
b
a
n
n


1
Init
clk_div =
clk_rest =
$\overline{ini}$/<<1 =
mux =
2
Init
clk_div =
clk_rest =
$\overline{ini}$/<<1 =
mux =
3
Schieben
clk_div =
clk_rest =
$\overline{ini}$/<<1 =
mux =
4
Schieben
clk_div =
clk_rest =
$\overline{ini}$/<<1 =
mux =
6
Sub; Rest |= 1
clk_div =
clk_rest =
$\overline{ini}$/<<1 =
mux =
5
Sub; Rest |= 1
clk_div =
clk_rest =
$\overline{ini}$/<<1 =
mux =
7
Ende
clk_div =
clk_rest =
$\overline{ini}$/<<1 =
mux =

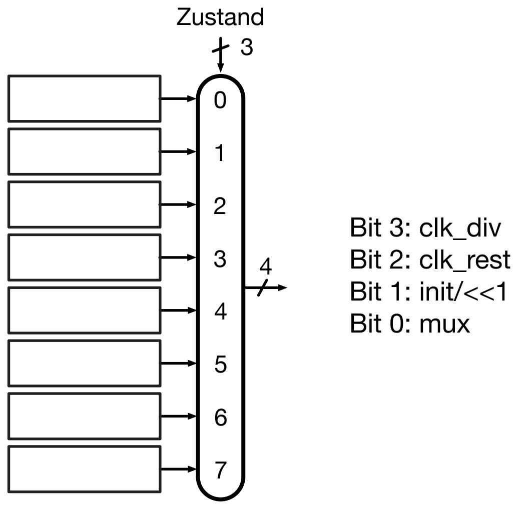
Die Steuerung der Dividierer-Schaltung wird nun für die Wortbreite $n = 4$ wie folgt implementiert:



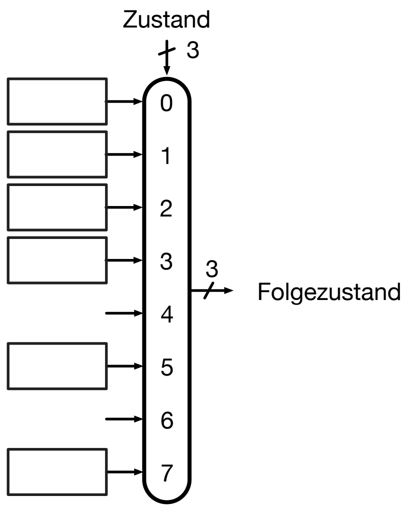
(T) c) In welchen Bits des Zustandsregisters wird der aktuelle Zustand und die Anzahl der bisher durchgeführten Runden abgespeichert?

Implementierung des Zustandsautomaten mit Multiplexern

Ⓣ a) Geben Sie für die Eingänge des Multiplexers binär die Ausgangsworte an, mit denen sich die *Ausgangsfunktion* des Moore-Automaten ergibt.



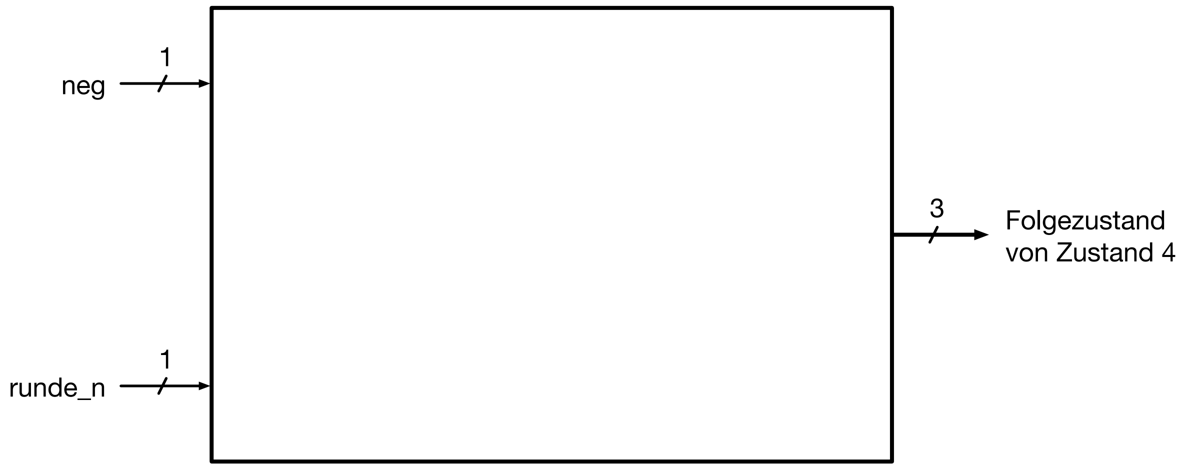
Ⓣ b) Geben Sie die Folgezustände für alle unbedingten Verzweigungen an.



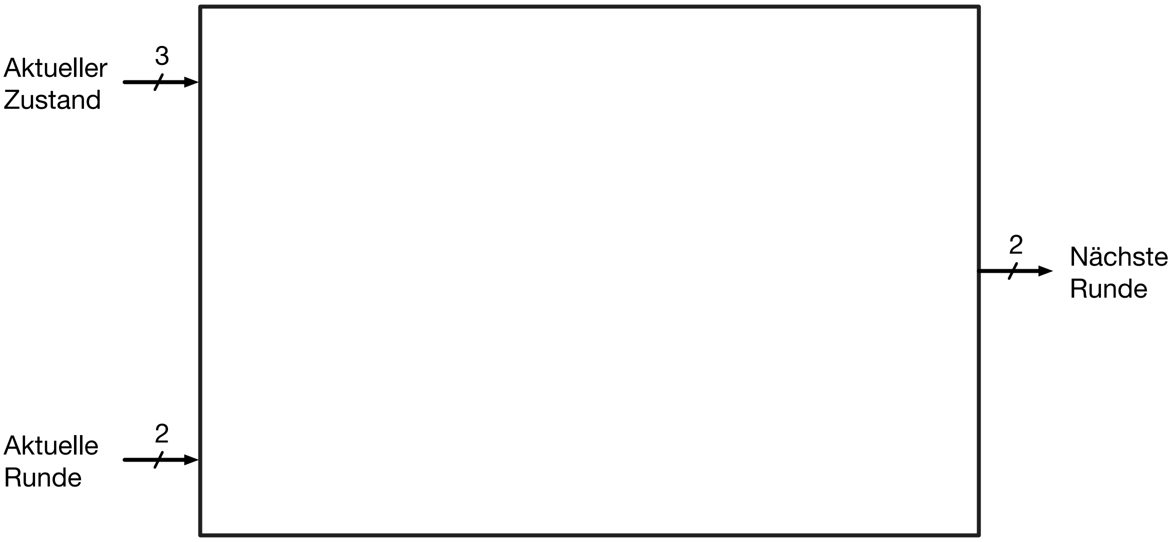
(T) c) Geben Sie eine Multiplexer-Schaltung an, die mittels des Signals *runde_n* die Folgezustände des Zustands 6 an ihrem Ausgang bereitstellt.

(T) d) Geben Sie eine Schaltung zur Bestimmung des Folgezustands von Zustands 6 an, die ohne Multiplexer auskommt.

(T) e) Geben Sie eine Multiplexer-Schaltung an, die mittels der Signale *runde_n* und *neg* den Folgezustand des Zustands 4 an ihrem Ausgang bereitstellt.



(T) f) Geben Sie eine kombinatorische Schaltung für den Rundenzähler an, der jedesmal, wenn sich der Moore-Automat im *Zustand 3* befindet, die in Bits 3 und 4 des Zustandsworts gespeicherte Rundenanzahl um Eins erhöht.

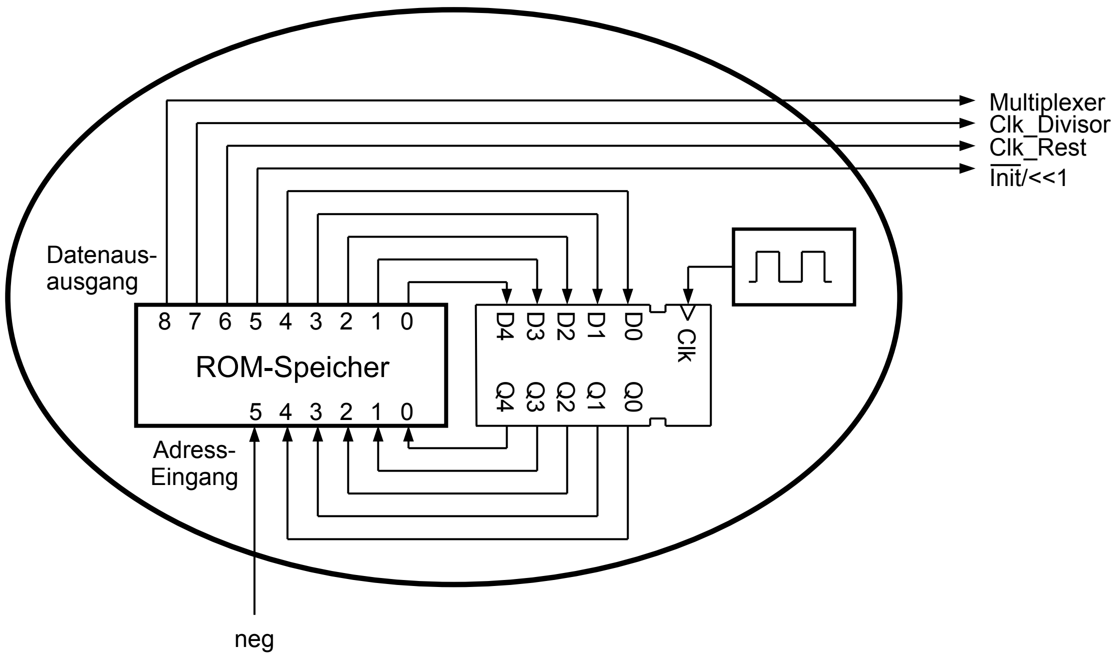


Der Rundenzähler zählt wie folgt: Runde 1 $\Leftrightarrow$ 01_2, Runde 1 $\Leftrightarrow$ 10_2, Runde 3 $\Leftrightarrow$ 11_2 und Runde 4 $\Leftrightarrow$ 00_2.

(T) g) Tragen Sie in nachfolgende Abbildung eine kombinatorische Schaltung ein, die in der 4. Runde, aus dem Rundenzähler das Signal *runde_n* erzeugt.

Implementierung des Zustandsautomaten mit Speicherbausteinen

Im Folgenden wird anstelle der kombinatorischen Logik ein ROM-Speicher verwendet.



Der ROM-Speicher funktioniert wie folgt: Die Bitkombination, die am Adress-Eingang anliegt, wird als Adresse interpretiert. Am Datenausgang wird dann das Datenwort ausgegeben, das an der durch den Adress-Eingang spezifizierte Adresse liegt.
Die sog. *Speicherorganisation* beschreibt den Speicheraufbau:

- Wie breit (in Bit) sind die Datenworte?
- Wieviele Datenworte können abgespeichert werden?

(T) a) Geben Sie die Organisation des gezeigten ROM-Speichers an.

Ⓣ b) Geben Sie den ROM-Inhalt an, der zur Implementierung der Zustände 1 und 2 benötigt wird.

	neg	Runde	Zustand	Ausgang	Folgerunde	Folgezust.
Zust. 1						
Zust. 2						

Ⓣ c) Geben Sie den ROM-Inhalt an, der zur Implementierung des Zustands 3 benötigt wird.

	neg	Runde	Zustand	Ausgang	Folgerunde	Folgezust.
Zust. 3						

T d) Geben Sie den ROM-Inhalt an, der zur Implementierung des Zustands 4 benötigt wird.

	neg	Runde	Zustand	Ausgang	Folgerunde	Folgezust.
Zust. 4						