Zwischenkapitel Datentypen und Operationen

Realisierung der Basiselemente in Hardware

ZK.1 Bitvektoren, natürliche Zahlen, ganze Zahlen

Bitvektoren

- kleinste Informationseinheit ist das BIT
- komplexere Informationseinheiten sind durch Bitvektoren darstellbar:
  - 4Bit = Nibble, 8Bit = Byte, 16Bit = Word 32Bit = Doubleword, 64Bit = Quadword
- Operationen
  - Schiebeoperationen
  - Test Bit i, Set Bit i
  - logische Verknüpfungen (AND, OR, ...)

Schiebeoperationen

- Verschieben eines Operanden um *n* Bitstellen
- Man unterscheidet (gemäß Behandlung der Datenformatgrenzen)
  - logisches Schieben
  - arithmetisches Schieben

- sll/srl (shift left/right logical)
  - Verschieben um *n* Stellen
  - Nachziehen von Nullen
  - herausfallende Bits in Übertragsbit (carry bit)

- sll/srl (shift left/right logical)
  - Verschieben um *n* Stellen
  - Nachziehen von Nullen
  - herausfallende Bits in Übertragsbit (carry bit)
- sla/sra (shift left/right arithmetic)
  - Verschieben um *n* Stellen
  - zieht von links das Bit mit höchster Wertigkeit nach
  - sla=Mult mit 2^n, sra=Div durch 2^n

Schieben um 1 Bit

Schieben um 1 Bit

- $srl_1(a_{n-1},\ldots,a_0) = (0,a_{n-1},\ldots,a_1)$

Schieben um 1 Bit

- $srl_1(a_{n-1},\ldots,a_0) = (0,a_{n-1},\ldots,a_1)$
- $sll_1(a_{n-1},\ldots,a_0) = (a_{n-2},\ldots,a_0,0)$

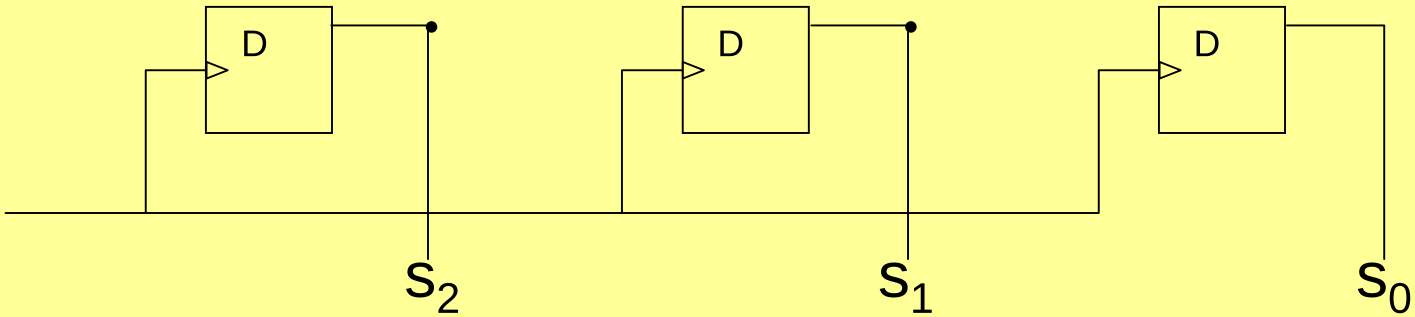
Schieben um 1 Bit

- $srl_1(a_{n-1},\ldots,a_0) = (0,a_{n-1},\ldots,a_1)$
- $sll_1(a_{n-1},\ldots,a_0) = (a_{n-2},\ldots,a_0\,,0)$
- $sra_1(a_{n-1},\ldots,a_0) = (a_{n-1},a_{n-1},\ldots,a_1)$

Schieben um 1 Bit

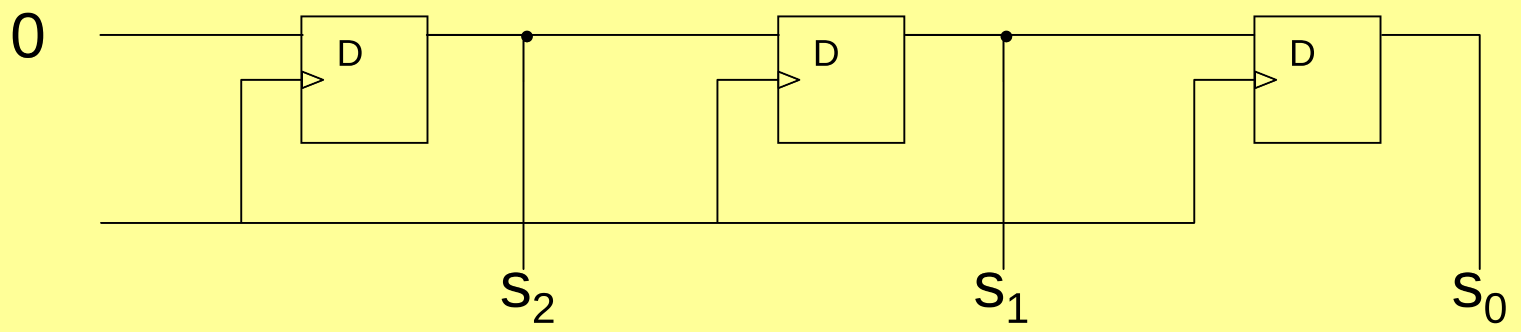
- $srl_1(a_{n-1},\ldots,a_0) = (0,a_{n-1},\ldots,a_1)$
- $sll_1(a_{n-1},\ldots,a_0) = (a_{n-2},\ldots,a_0\,,0)$
- $sra_1(a_{n-1},\ldots,a_0) = (a_{n-1},a_{n-1},\ldots,a_1)$
- $sla_1(a_{n-1},\ldots,a_0) = (a_{n-1},a_{n-3},\ldots,a_1\,,0)$

Schieberegister

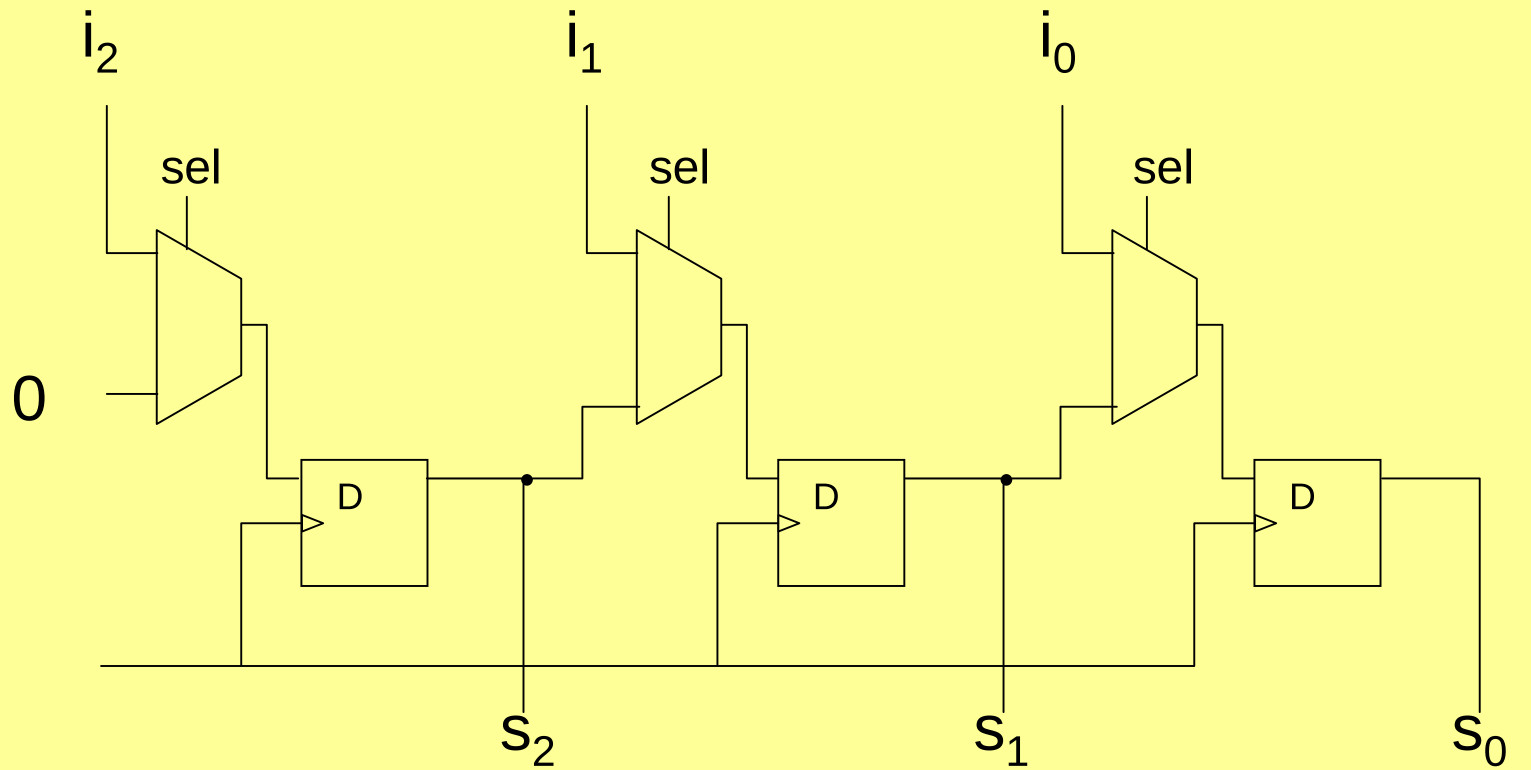


Schieberegister

$$srl_1(a_{n-1},\ldots,a_0) = (0,a_{n-1},\ldots,a_1)$$

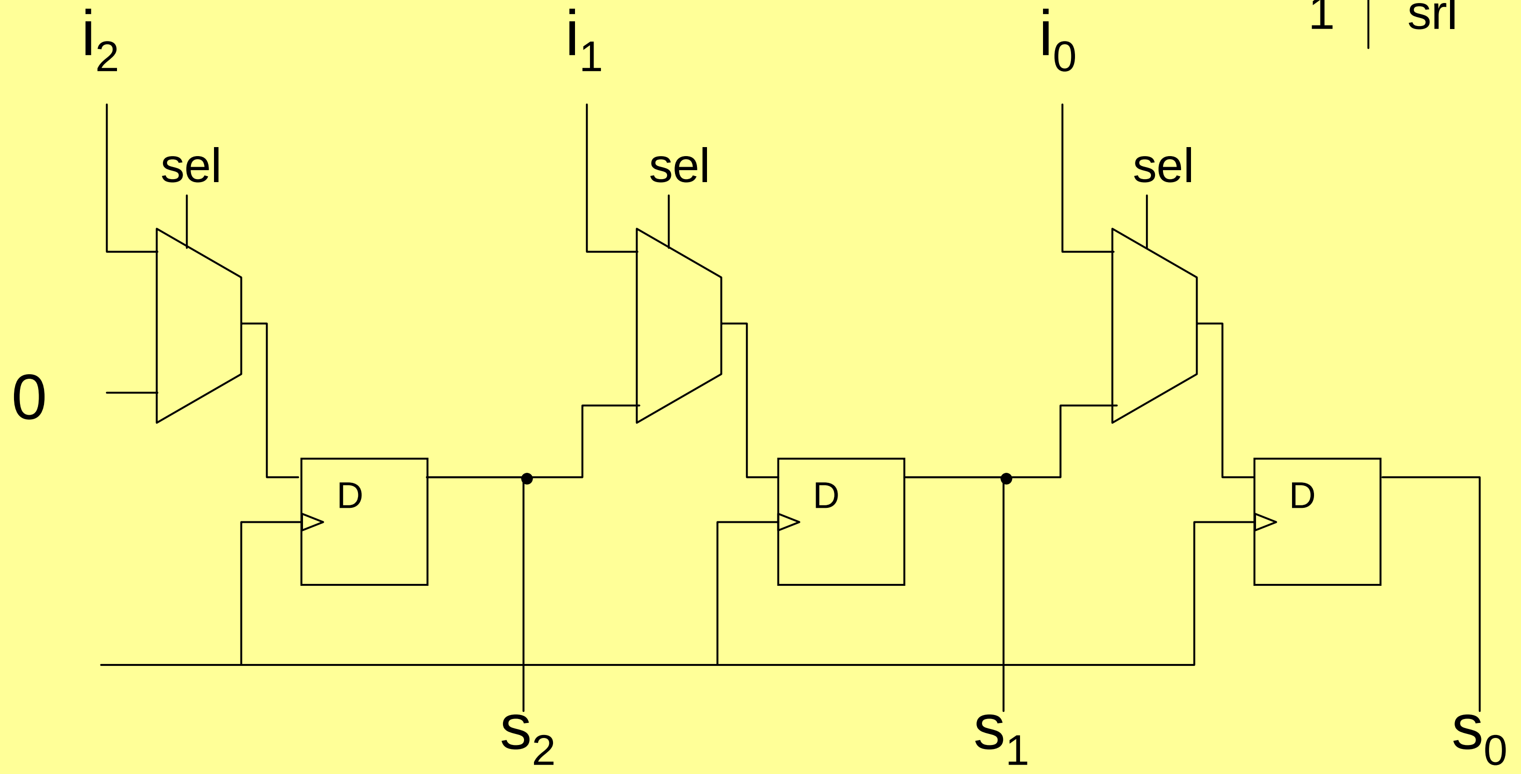


Schieberegister

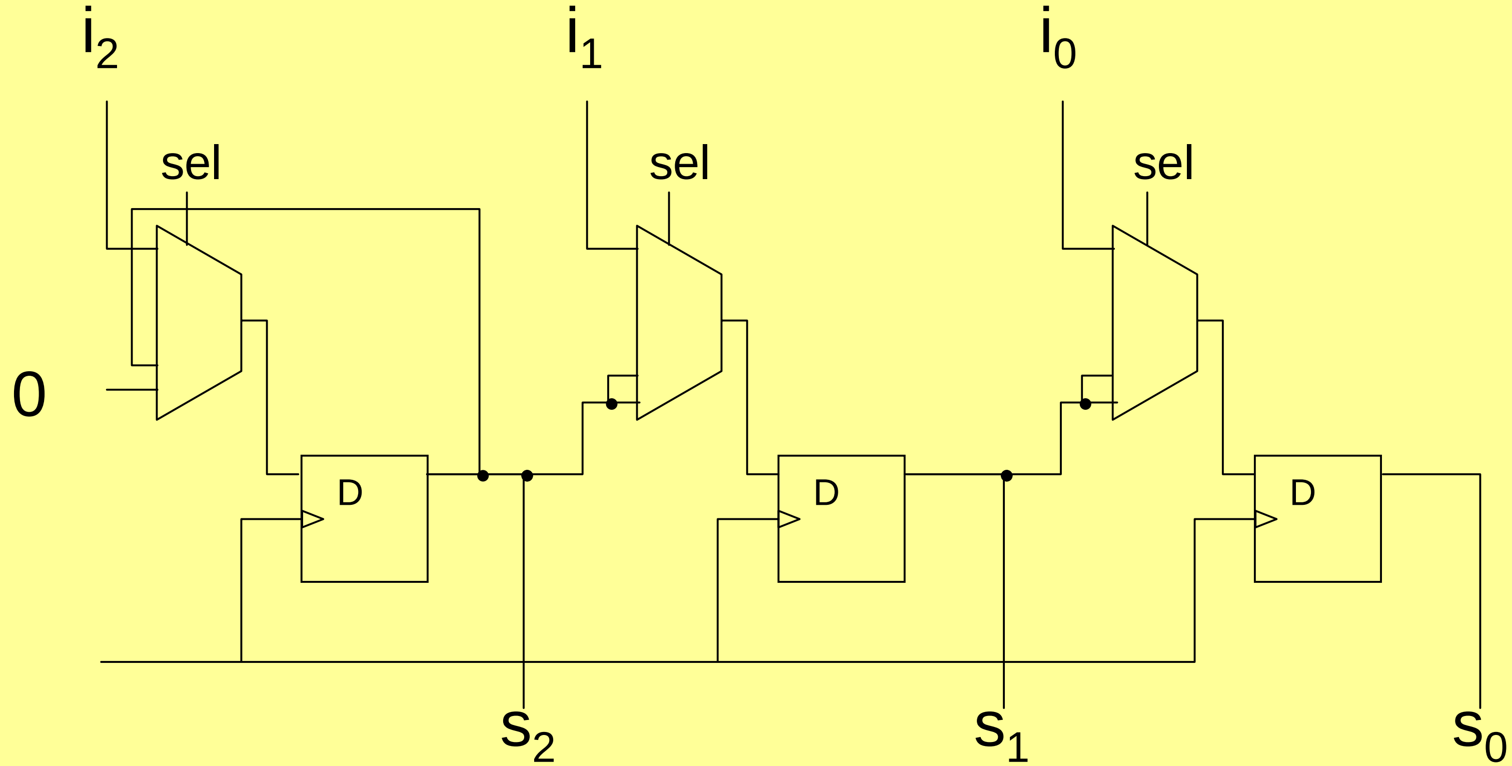


Schieberegister

sel	op
0	load
1	srl

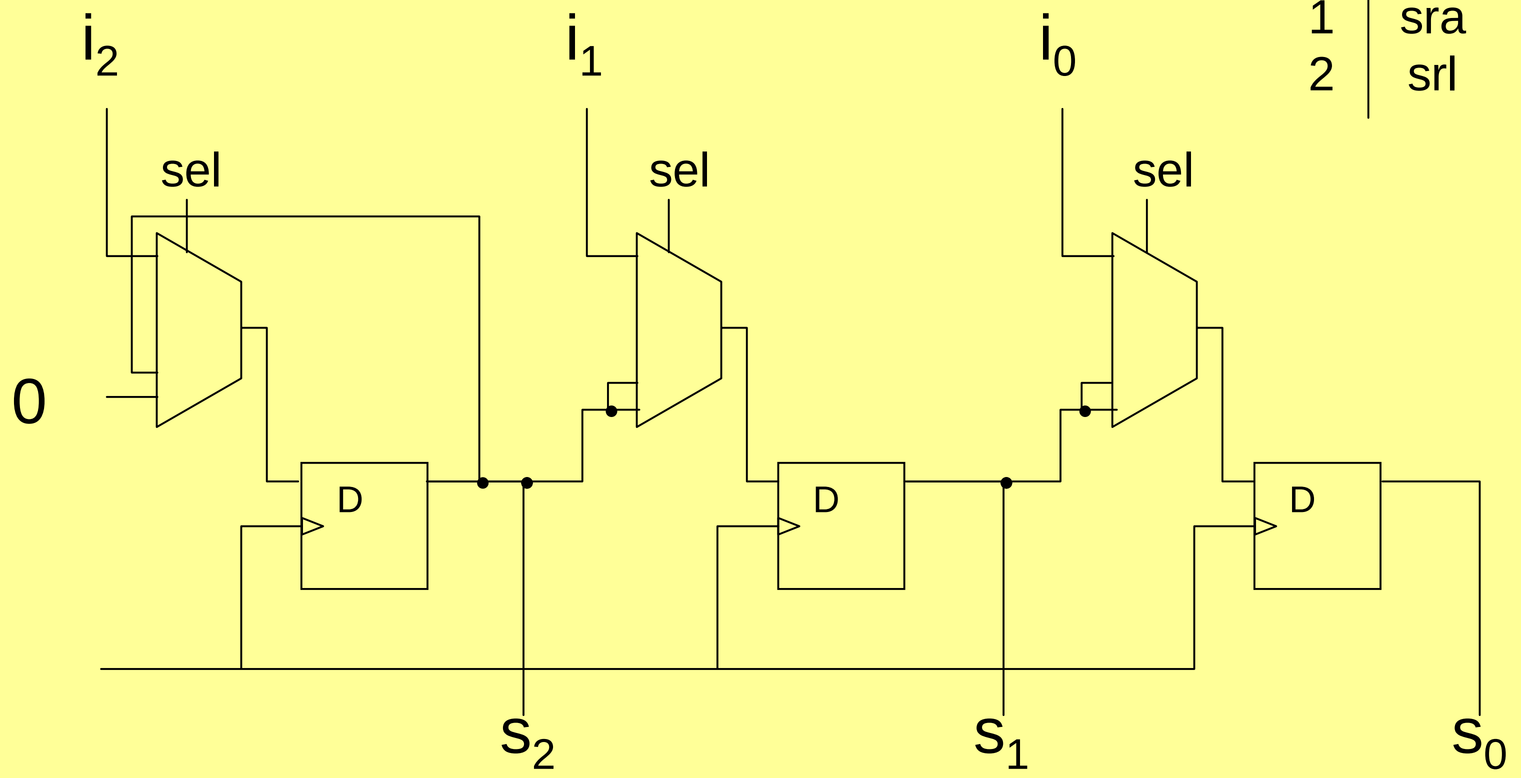


Schieberegister

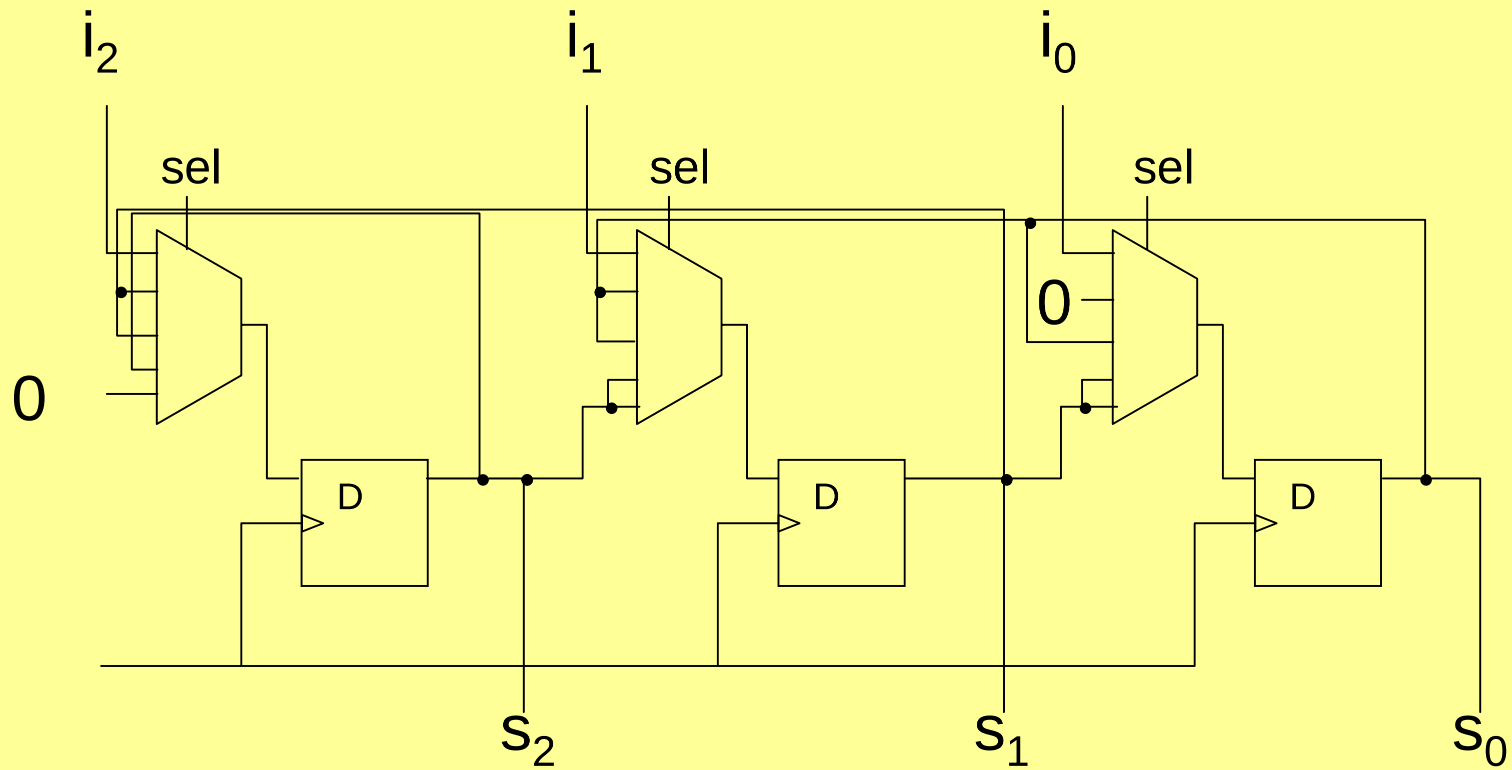


Schieberegister

sel	op
0	load
1	sra
2	srl

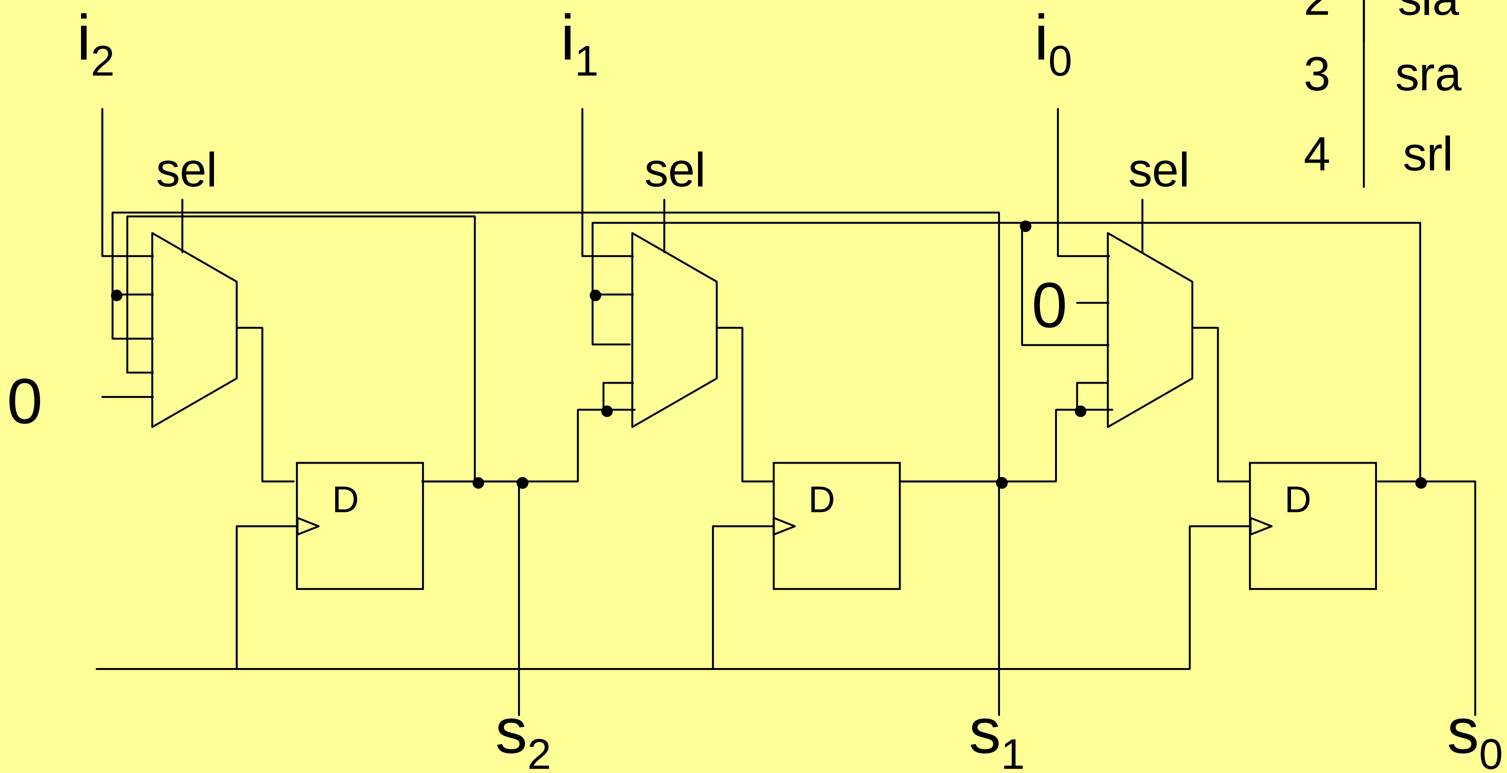


Schieberegister



Schieberegister

sel	op
0	load
1	sll
2	sla
3	sra
4	srl



Natürliche Zahlen

- Interpretation des Bitvektors als Dualzahl

  sei $a = (a_{n-1}, a_{n-2},\ldots, a_1, a_0)$

  $$nat(a) = 2^{n-1} \cdot a_{n-1} + 2^{n-2} \cdot a_{n-2} + \ldots + 2 \cdot a_1 + a_0$$

- Es gibt auch andere Kodierungen
  - BCD
  - Gray
  - ...

Ganze Zahlen

- Darstellung im Zweierkomplement

  Sei $a = (a_{n-1}, \dots, a_0)$

  $int(a) = -2^{n-1} \cdot a_{n-1} + 2^{n-2} \cdot a_{n-2} + \dots + 2 \cdot a_1 + a_0$

- Zweierkomplement von a

  $a' = \overline{a} + 1$

Im Vergleich zu anderen Darstellungen

Darstellung	*redundant / irredundant*	*symmetrischer Zahlenbereich*	*geeignet zum Rechnen*
Betrag und Vorzeichen	**redundant**	**ja**	**bedingt**
Einer-Komplement	**redundant**	**ja**	**gut**
Zweier-komplement	**irredundant**	**nein**	**sehr gut**

Addition

Hardware-Realisierung eines 1-Bit Addition

Addition

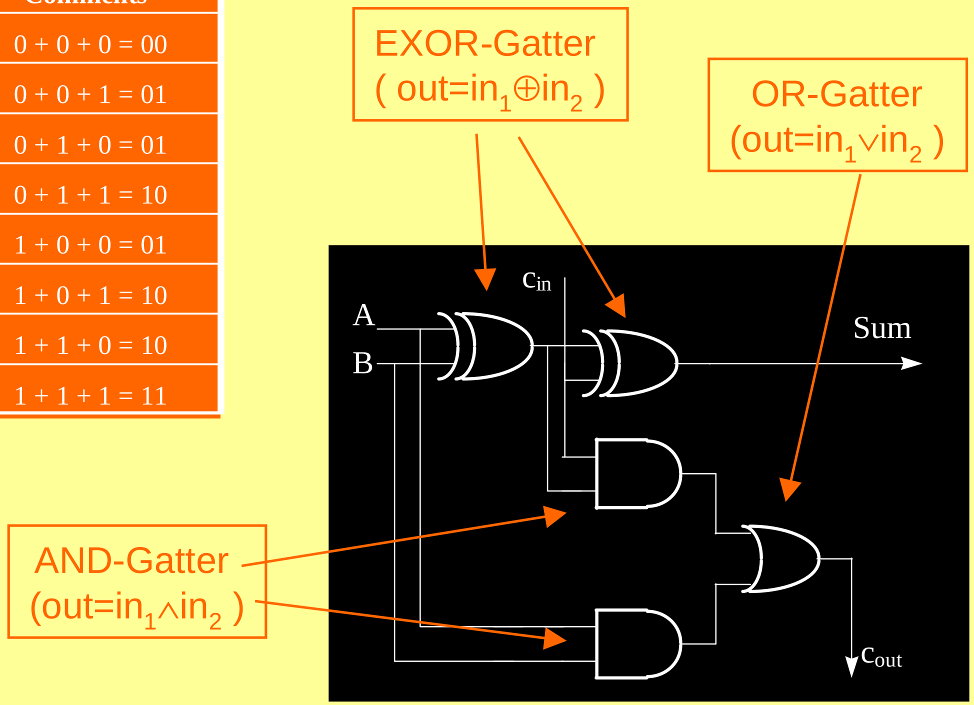
Hardware-Realisierung eines 1-Bit Addition

Inputs			Outputs		Comments
A	B	CarryIn	CarryOut	Sum	
0	0	0	0	0	0 + 0 + 0 = 00
0	0	1	0	1	0 + 0 + 1 = 01
0	1	0	0	1	0 + 1 + 0 = 01
0	1	1	1	0	0 + 1 + 1 = 10
1	0	0	0	1	1 + 0 + 0 = 01
1	0	1	1	0	1 + 0 + 1 = 10
1	1	0	1	0	1 + 1 + 0 = 10
1	1	1	1	1	1 + 1 + 1 = 11

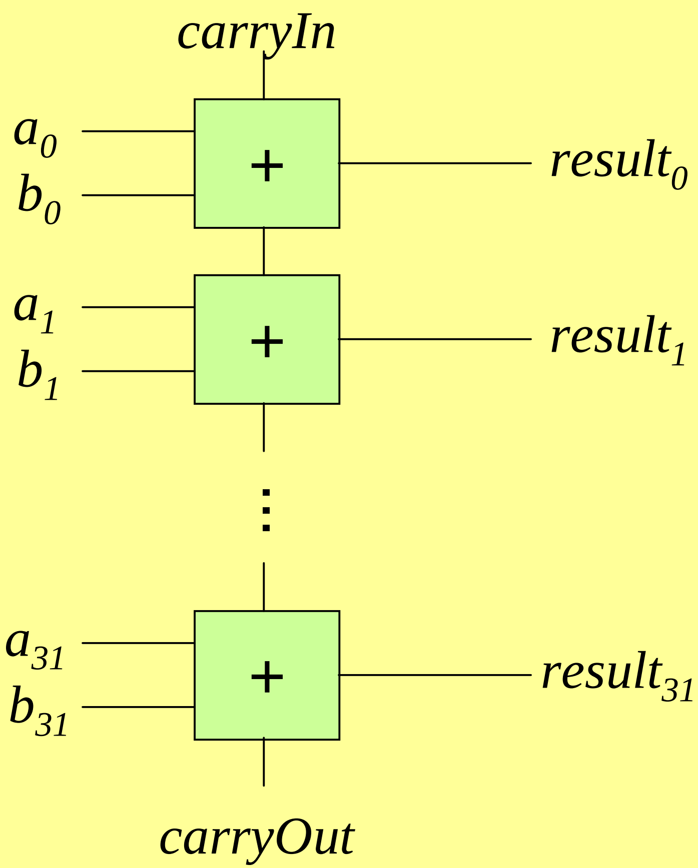
Addition

Hardware-Realisierung eines 1-Bit Addition

Inputs			Outputs		
A	**B**	**CarryIn**	**CarryOut**	**Sum**	**Comments**
0	0	0	0	0	0 + 0 + 0 = 00
0	0	1	0	1	0 + 0 + 1 = 01
0	1	0	0	1	0 + 1 + 0 = 01
0	1	1	1	0	0 + 1 + 1 = 10
1	0	0	0	1	1 + 0 + 0 = 01
1	0	1	1	0	1 + 0 + 1 = 10
1	1	0	1	0	1 + 1 + 0 = 10
1	1	1	1	1	1 + 1 + 1 = 11



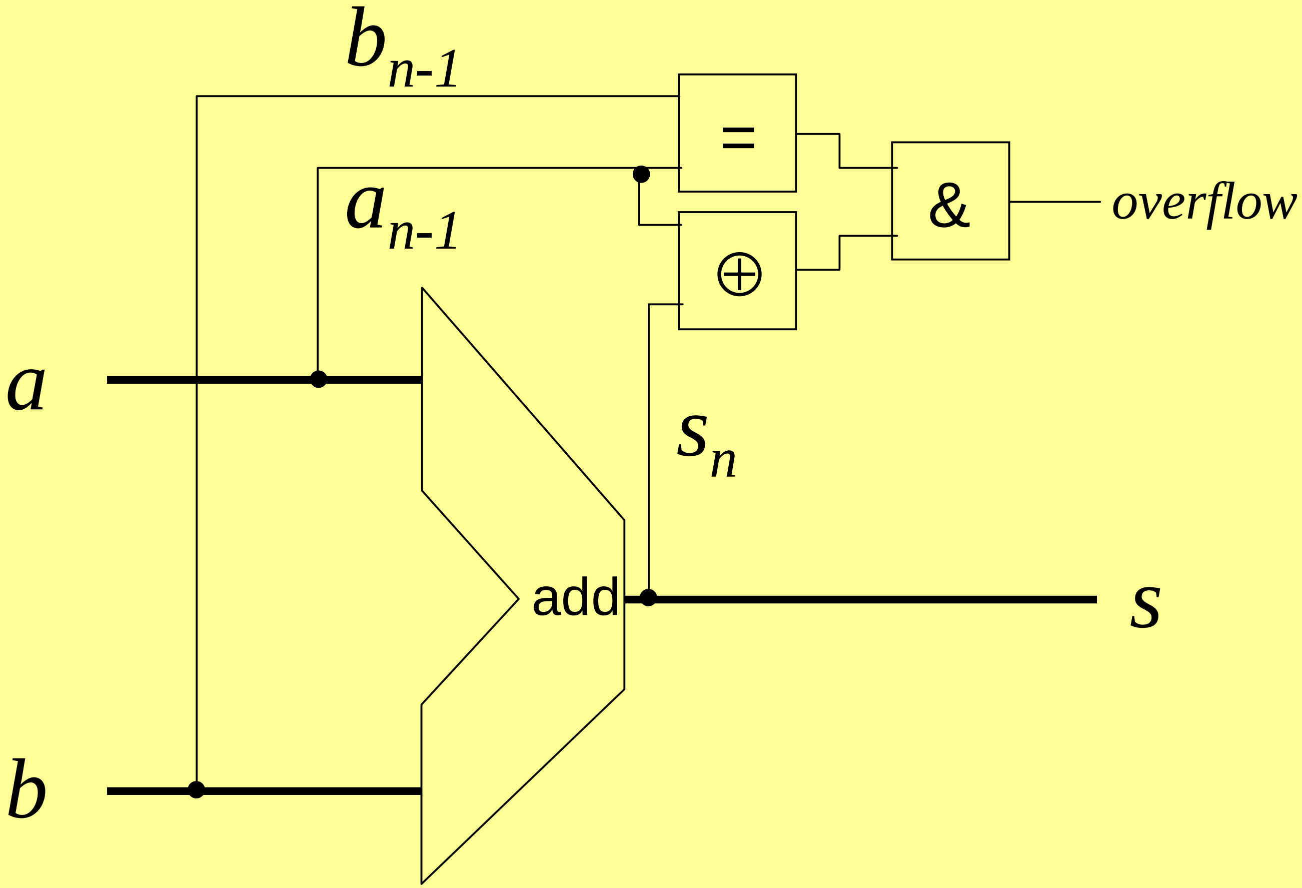
Addition: 32-Bit-Addierer



Überlauferkennung

- Bei natürlichen Zahlen: $c_{out} = 1$
- Bei ganzen Zahlen im Zweierkomplement sei $s = a + b$ mit $a = (a_{n-1}, a_{n-2},..., a_1, a_0)$ etc.
  - $a \geq 0$ und $b<0$ oder $a<0$ und $b\geq 0$
    - → kein Überlauf
  - $a \geq 0$ und $b \geq 0$
    - → Überlauf bei $s<0$
  - $a<0$ und $b<0$
    - → Überlauf bei $s \geq 0$

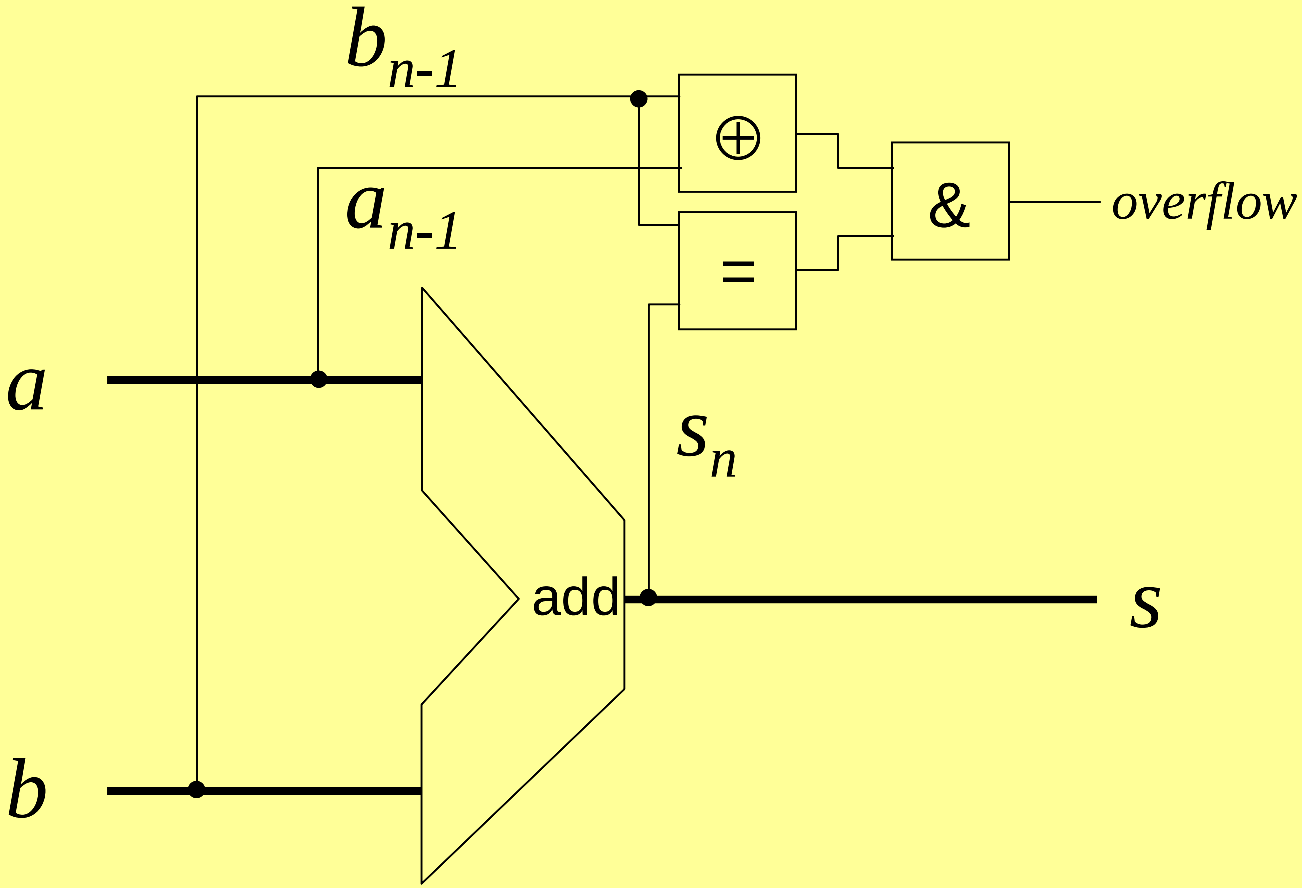
Überlauferkennung



Subtraktion

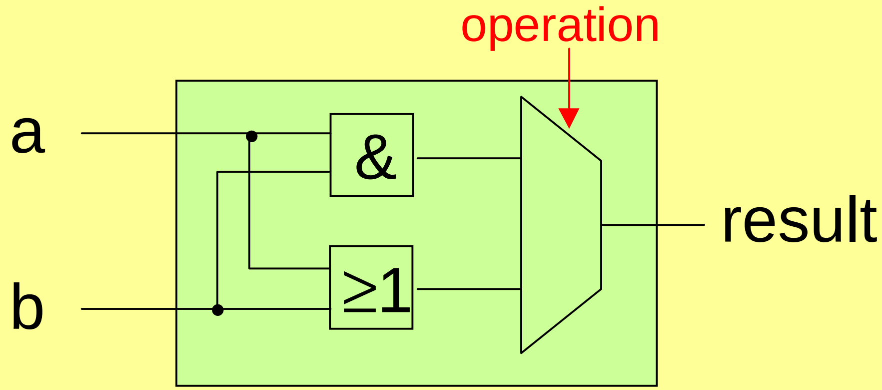
- Addition mit Zweierkomplement
- Überlauferkennung
  sei $s = a - b$ mit $a = (a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$ etc.
  - $a \geq 0$ und $b \geq 0$ oder $a<0$ und $b<0$
    - → kein Überlauf
  - $a \geq 0$ und $b<0$
    - → Überlauf bei $s<0$
  - $a<0$ und $b \geq 0$
    - → Überlauf bei $s \geq 0$

Überlauferkennung



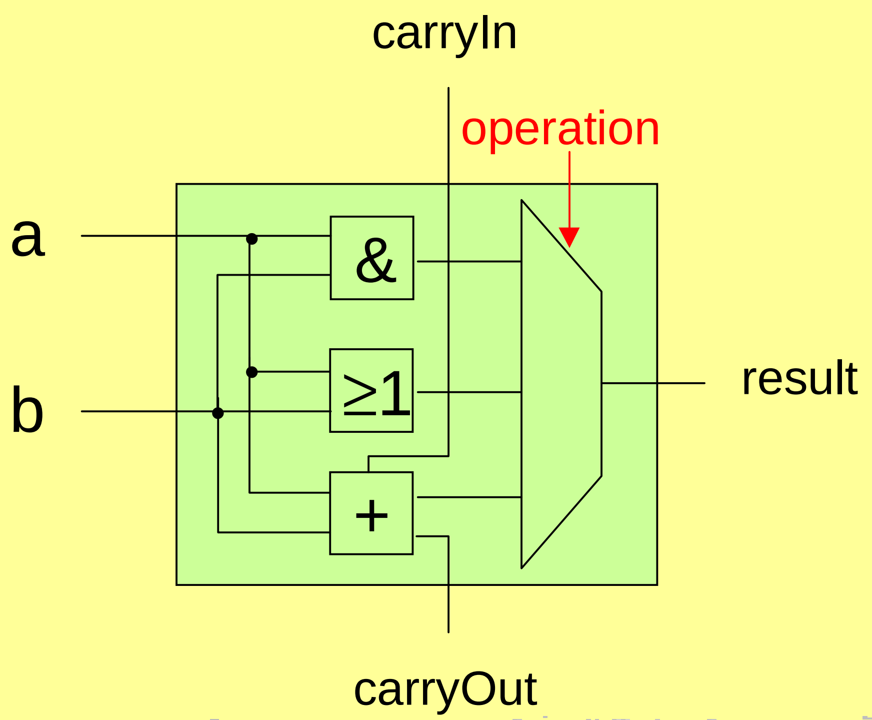
Aufbau einer einfachen ALU

- Logische Grundoperationen auf einem Bit



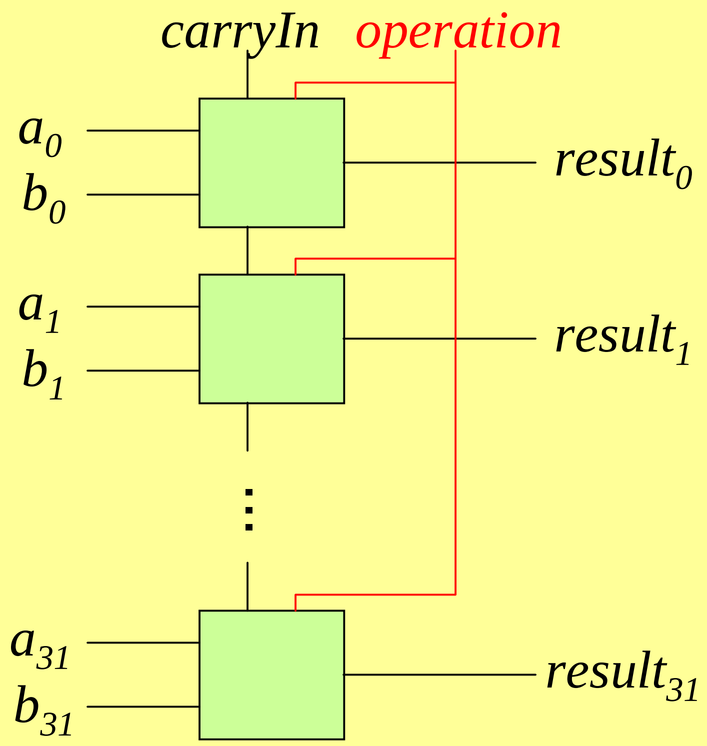
Aufbau einer einfachen ALU

- Addition



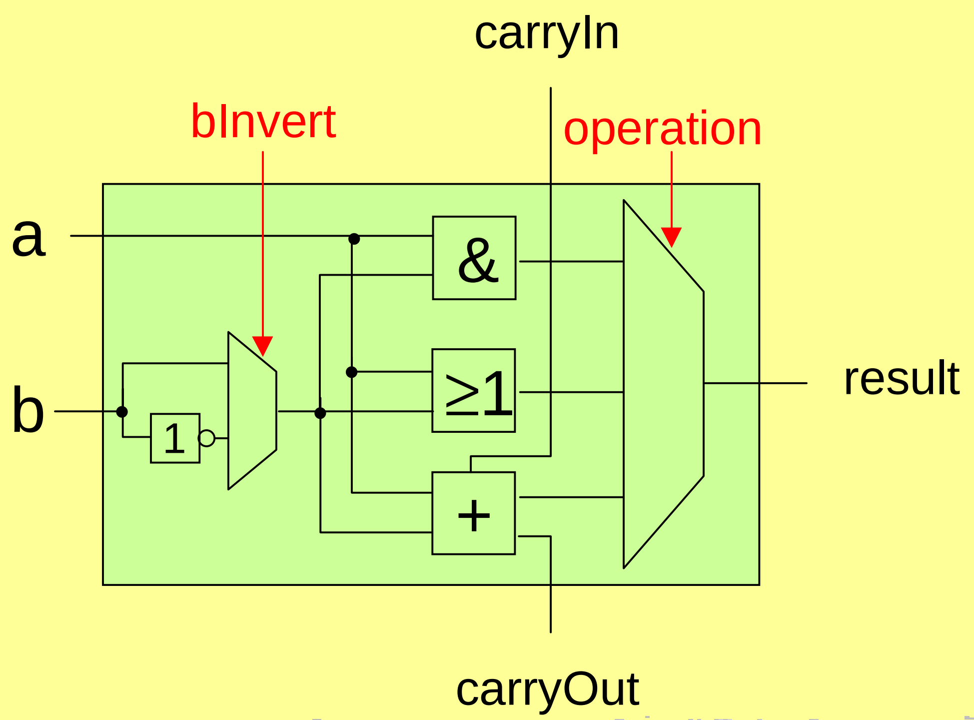
Aufbau einer einfachen ALU

- Zusammenschaltung



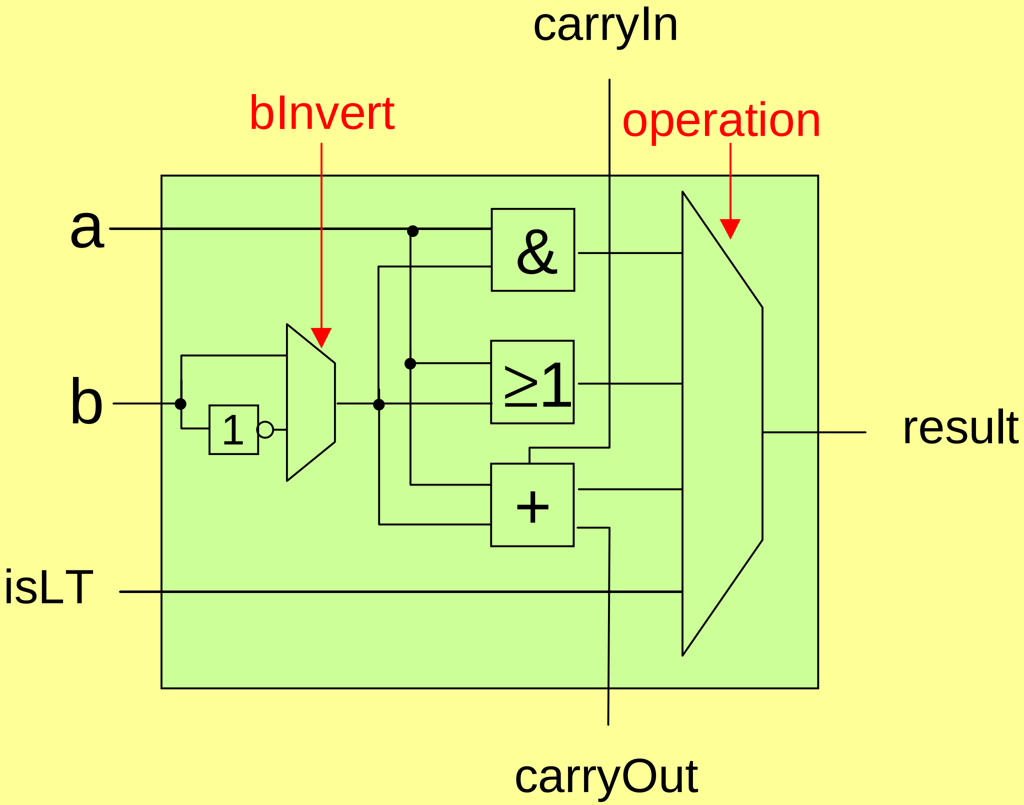
Aufbau einer einfachen ALU

- Subtraktion:
  $$a - b = a + (-b) = a + (\overline{b} + 1) = a + \overline{b} + 1$$



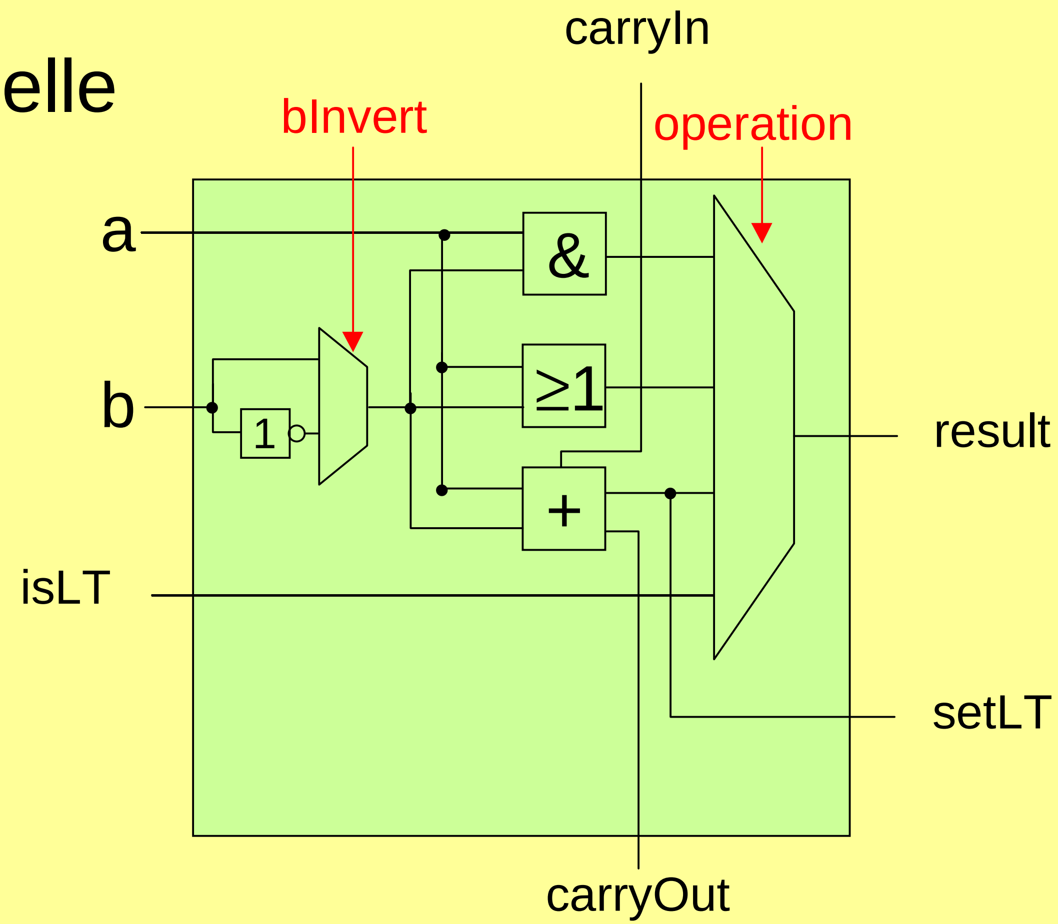
Aufbau einer einfachen ALU

- Vergleich a < b: $a < b \Leftrightarrow a - b < 0$



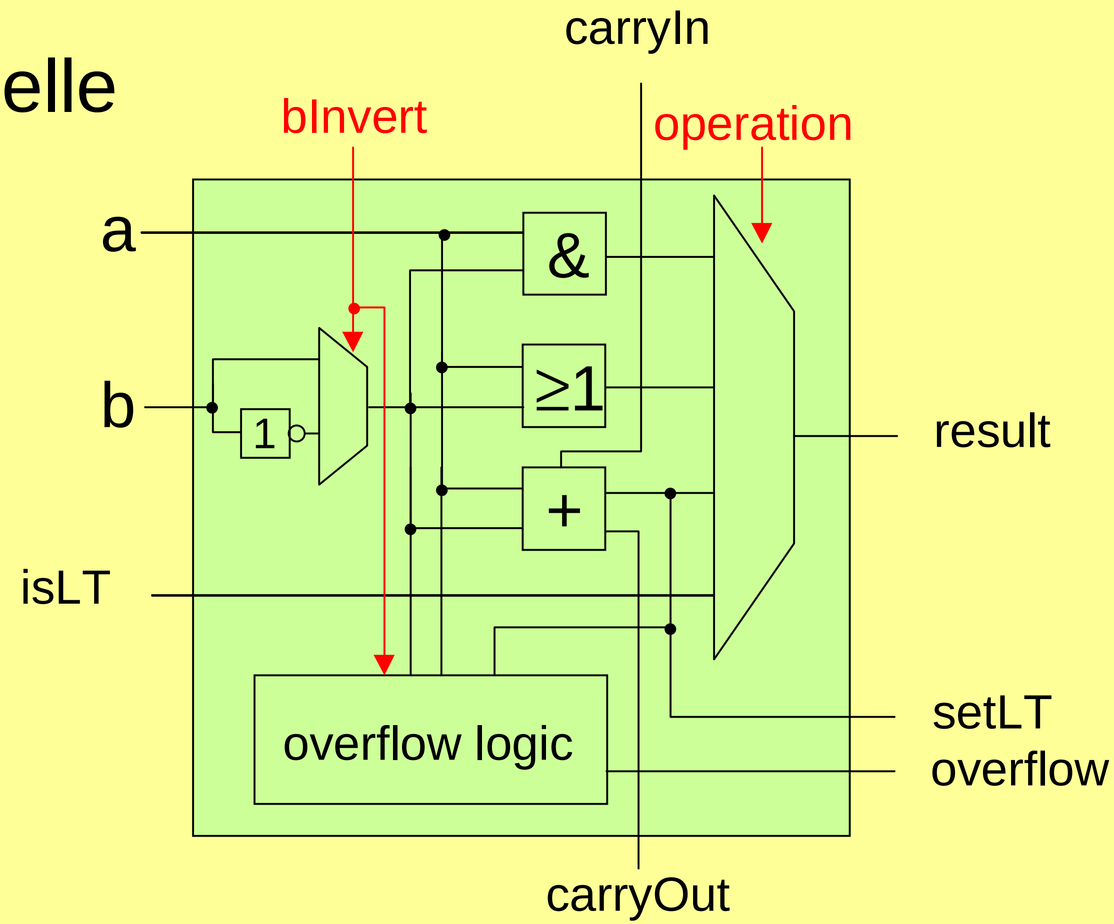
Aufbau einer einfachen ALU

MSB-Zelle

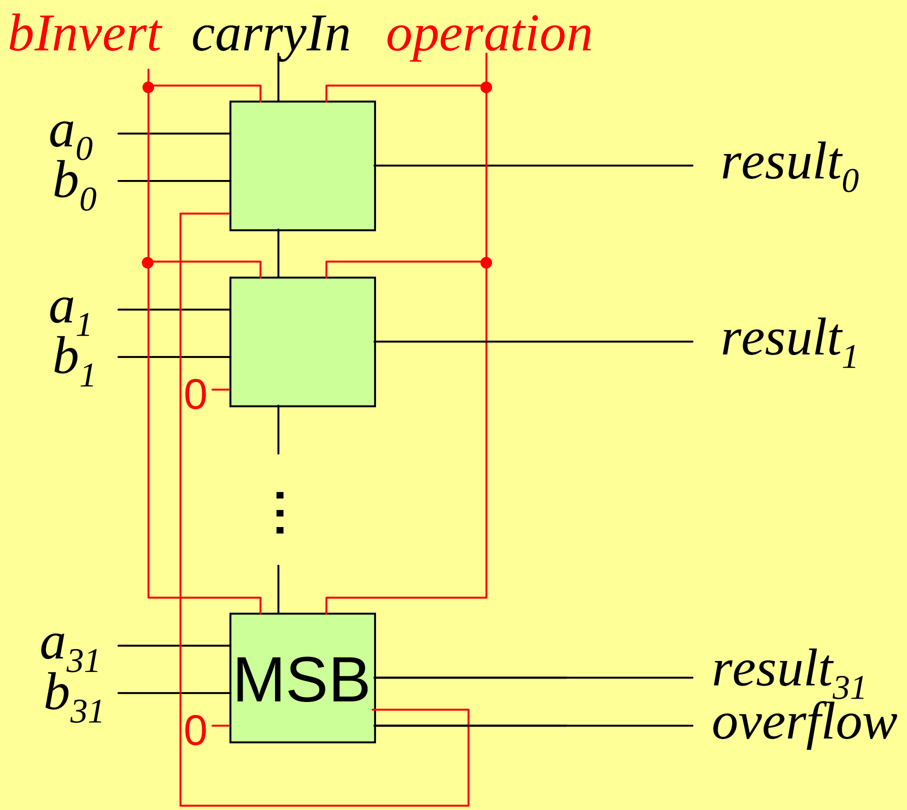


Aufbau einer einfachen ALU

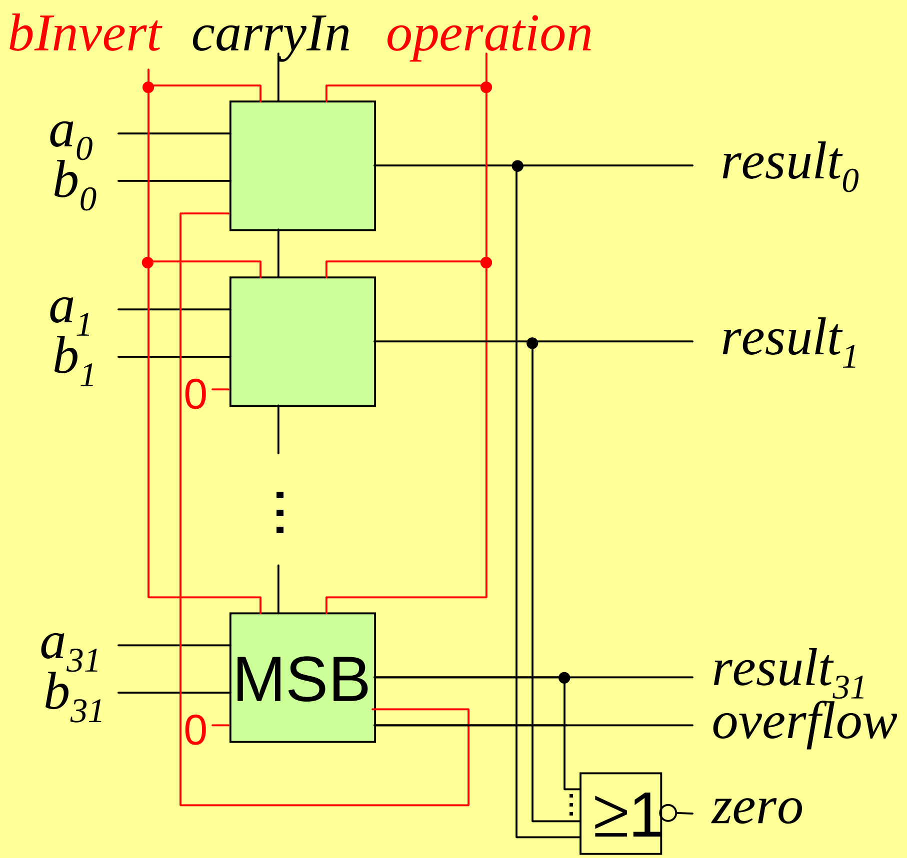
MSB-Zelle



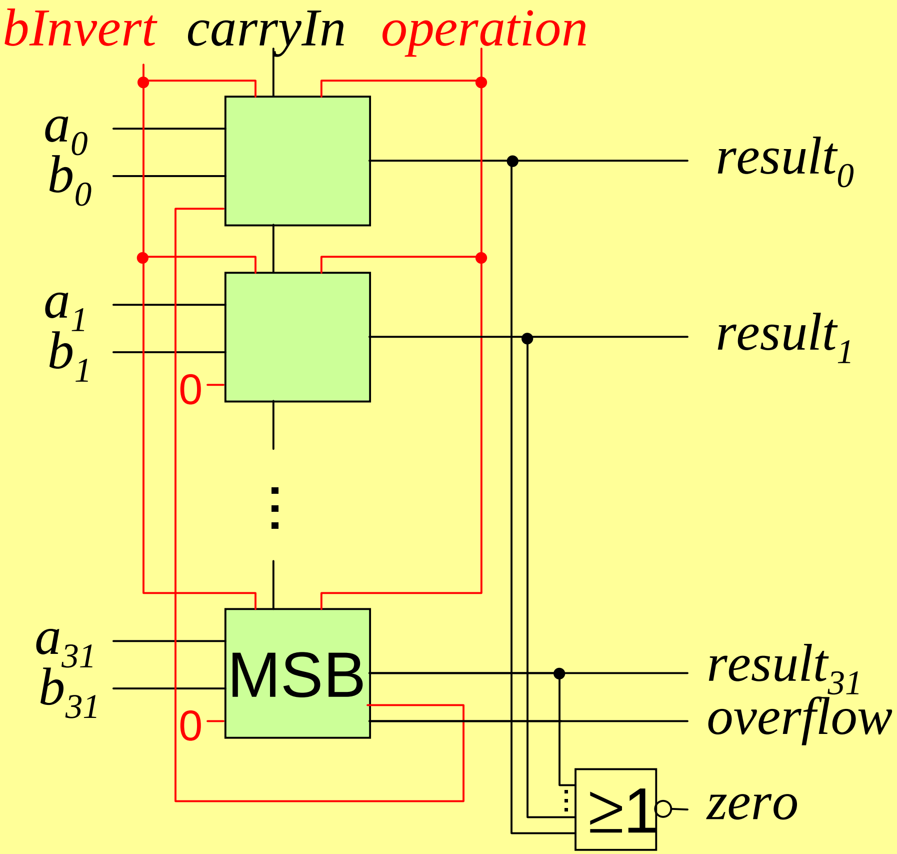
Aufbau einer einfachen ALU



Aufbau einer einfachen ALU



Aufbau einer einfachen ALU



bInvert	carryIn	operation	result
0	-	00	$a \wedge b$
0	-	01	$a \vee b$
1	-	00	$a \wedge \overline{b}$
1	-	01	$a \vee \overline{b}$
0	0	10	$a+b$
0	1	10	$a+b+1$
1	0	10	$a-b-1$
1	1	10	$a-b$
1	1	11	$a<b?1:0$
...	...	....	...

Multiplikation im Zweierkomplement

```
0101 * 0011
-----------
       0011
      0000
     0011
    0000
-----------
    0001111
```

Multiplikation im Zweierkomplement

Multiplikation nach der Schulmethode

```
0101 * 0011
-----------
       0011
      0000
     0011
    0000
-----------
    0001111
```

Multiplikation im Zweierkomplement

Multiplikation nach der Schulmethode

```
0101 * 0011
-----------
       0011
      0000
     0011
    0000
-----------
    0001111
```

Multiplikation im Zweierkomplement

Multiplikation nach der Schulmethode

```
0101 * 0011
-----------
       0011
      0000
     0011
    0000
-----------
    0001111
```

- Bei negativen Zahlen muss zuerst der Betrag gebildet werden, erst dann kann multipliziert werden.

Multiplikation im Zweierkomplement

Multiplikation nach der Schulmethode

```
0101 * 0011
-----------
       0011
      0000
     0011
    0000
-----------
    0001111
```

- Bei negativen Zahlen muss zuerst der Betrag gebildet werden, erst dann kann multipliziert werden.
- Verfahren ist ziemlich langsam, da insgesamt *n* Additionen ausgeführt werden, insbesondere Addition mit Null.

Verfahren von Booth

Verfahren von Booth

A. Booth. A Signed Binary Multiplication Scheme. Q.J.Mech.Appl.Math. 4:236:240 (1951)

Verfahren von Booth

A. Booth. A Signed Binary Multiplication Scheme. Q.J.Mech.Appl.Math. 4:236:240 (1951)

Beobachtung

Verfahren von Booth

A. Booth. A Signed Binary Multiplication Scheme. Q.J.Mech.Appl.Math. 4:236:240 (1951)

Beobachtung

- Enthält der Multiplikator *y* einen Nullblock der Länge *k*, so kann die Multiplikation durch ein Shift der Zwischensumme um *k* Stellen beschleunigt werden.

Verfahren von Booth

A. Booth. A Signed Binary Multiplication Scheme. Q.J.Mech.Appl.Math. 4:236:240 (1951)

Beobachtung

- Enthält der Multiplikator *y* einen Nullblock der Länge *k*, so kann die Multiplikation durch ein Shift der Zwischensumme um *k* Stellen beschleunigt werden.
- Enthält der Multiplikator *y* einen Einsblock von Stelle *u* bis Stelle *v*, z.B.:

0.....01........10............0

Verfahren von Booth

A. Booth. A Signed Binary Multiplication Scheme. Q.J.Mech.Appl.Math. 4:236:240 (1951)

Beobachtung

- Enthält der Multiplikator *y* einen Nullblock der Länge *k*, so kann die Multiplikation durch ein Shift der Zwischensumme um *k* Stellen beschleunigt werden.
- Enthält der Multiplikator *y* einen Einsblock von Stelle *u* bis Stelle *v*, z.B.:

$$\underset{v}{0.....01}.......\underset{u}{1}0...........0$$

Verfahren von Booth

A. Booth. A Signed Binary Multiplication Scheme. Q.J.Mech.Appl.Math. 4:236:240 (1951)

Beobachtung

- Enthält der Multiplikator *y* einen Nullblock der Länge *k*, so kann die Multiplikation durch ein Shift der Zwischensumme um *k* Stellen beschleunigt werden.
- Enthält der Multiplikator *y* einen Einsblock von Stelle *u* bis Stelle *v*, z.B.:

  0.....01.......10...........0

  v *u*

  so können die zum Einsblock gehörigen (v-u+1) Additionen der Multiplikation nach Schulmethode wegen int(0...01...10...0)=2^{v+1}-2^{u} durch eine Addition an der Stelle v+1 und eine Substraktion an der Stelle *u* ersetzt werden

Verfahren von Booth

$$X * Y \text{ mit } Y = 0...0\underbrace{\overset{v}{1}...\overset{u}{1}}_{\text{add/shift}}\underbrace{0...0}$$

$$Y = 0...01...10...0$$

shift shift

add/shift sub/shift

Verfahren von Booth

$$X * Y \text{ mit } Y = 0...0\underbrace{\overset{v}{1}...\overset{u}{1}}_{\text{add/shift}}\underbrace{0...0}$$

$$X*Y = X2^u+X2^{u+1}+...+X2^v = X\,(2^u+2^{u+1}+...+2^v)$$

$$Y = 0...0\underset{\substack{\downarrow\\ \text{add/shift}}}{1}\underbrace{...}_{\text{shift}}\underset{\substack{\downarrow\\ \text{sub/shift}}}{1}\underbrace{0...0}_{\text{shift}}$$

Verfahren von Booth

$$X * Y \text{ mit } Y = 0\ldots0\underbrace{\overset{v}{1}\ldots\overset{u}{1}}_{\text{add/shift}}\underbrace{0\ldots0}$$

$$X*Y = X2^u+X2^{u+1}+\ldots+X2^v = X\,(2^u+2^{u+1}+\ldots+2^v)$$

$$\text{NR: } 00\overset{v}{1}1\overset{u}{1}000 = 00\overset{v}{1}11\overset{u}{1}11 - 00000\overset{v}{1}1\overset{u}{1}$$

$$= (2^{v+1}-1) - (\,2^u-1\,) = 2^{v+1}- 2^u$$

$$Y = 0\ldots0\,{\downarrow}\,\underbrace{1\ldots1}_{\text{shift}}\,{\downarrow}\,\underbrace{0\ldots0}_{\text{shift}}$$

add/shift sub/shift

Verfahren von Booth

$$X * Y \text{ mit } Y = 0...0\overset{v}{1}...\overset{u}{1}0...0$$

$$\underbrace{1...1}_{\text{add/shift}}\underbrace{0...0}$$

$$X*Y = X2^u+X2^{u+1}+...+X2^v = X\,(2^u+2^{u+1}+...+2^v)$$

NR: $00\overset{v}{1}1\overset{u}{1}000 = 00\overset{v}{1}1\overset{u}{1}111 - 00000\overset{v}{1}\overset{u}{1}1$

$= (2^{v+1}-1) - (\ 2^u-1\ ) = 2^{v+1}-\ 2^u$

$$= X\,(2^{v+1}-2^u) = X2^{v+1} - X2^u$$

$$Y = 0...0\underset{\text{add/shift}}{\downarrow}1\underbrace{...}_{\text{shift}}1\underset{\text{sub/shift}}{\downarrow}\underbrace{0...0}_{\text{shift}}$$

Verfahren von Booth

$$X * Y \text{ mit } Y = 0...0\overset{v}{1}...\overset{u}{1}0...0$$

$$\underbrace{1...1}_{\text{add/shift}}\underbrace{0...0}$$

$$X*Y = X2^u+X2^{u+1}+...+X2^v = X\,(2^u+2^{u+1}+...+2^v)$$

NR: $00\overset{v}{1}1\overset{u}{1}000 = 00\overset{v}{1}1\overset{u}{1}111 - 00000\overset{v}{1}1\overset{u}{1}$

$= (2^{v+1}-1) - (\,2^u-1\,) = 2^{v+1} - 2^u$

$$= X\,(2^{v+1}-2^u) = X2^{v+1} - X2^u$$

$$Y = 0...0\underset{\substack{\downarrow\\\text{add/shift}}}{1}\underbrace{...}_{\text{shift}}\underset{\substack{\downarrow\\\text{sub/shift}}}{1}\underbrace{0...0}_{\text{shift}}$$

Verfahren von Booth

Verfahren von Booth

- Arithmetische Operationen sind nur an den 0→1 und 1→0 Wechsel im Multiplikator erforderlich.

Verfahren von Booth

- Arithmetische Operationen sind nur an den $0 \to 1$ und $1 \to 0$ Wechsel im Multiplikator erforderlich.
- Man erhält die Rechenvorschrift

Verfahren von Booth

- Arithmetische Operationen sind nur an den 0→1 und 1→0 Wechsel im Multiplikator erforderlich.
- Man erhält die Rechenvorschrift

y_i	y_{i-1}	Operation
0	0	shift
0	1	add; shift
1	0	sub; shift
1	1	shift

mit $y_{-1}=0$

Verfahren von Booth: Beispiel

x	y_3	y_2	y_1	y_0	Operation	Zwischenergebnis
0010	0	1	1	0		0000
				↑	shift	0000 0
			↑		sub (add 1110)	1110 0
			↑		shift	1111 00
		↑			shift	1111 100
	↑				add 0010	0001 100
	↑				shift	0000 1100

Korrektheit des Verfahrens von Booth

Korrektheit des Verfahrens von Booth

Satz

Das Verfahren von Booth multipliziert sowohl positive als auch negative Zahlen

Korrektheit des Verfahrens von Booth

Satz

Das Verfahren von Booth multipliziert sowohl positive als auch negative Zahlen

Beweis

Korrektheit des Verfahrens von Booth

Satz

Das Verfahren von Booth multipliziert sowohl positive als auch negative Zahlen

Beweis

Wir betrachten im Verfahren von Booth an jeder Stelle die Differenz (y_{i-1}-y_i) und berechnen das Multiplikationsergebnisses durch die Summe

Korrektheit des Verfahrens von Booth

Satz

Das Verfahren von Booth multipliziert sowohl positive als auch negative Zahlen

Beweis

Wir betrachten im Verfahren von Booth an jeder Stelle die Differenz $(y_{i-1}-y_i)$ und berechnen das Multiplikationsergebnisses durch die Summe

$$
\begin{aligned}
S \;= \;& (y_{-1}-y_0)\cdot 2^0 \cdot \mathrm{int}(x) + \\
& (y_0-y_1)\cdot 2^1 \cdot \mathrm{int}(x) + \\
& \quad \ldots \\
& (y_{n-3}-y_{n-2})\cdot 2^{n-2} \cdot \mathrm{int}(x) + \\
& (y_{n-2}-y_{n-1})\cdot 2^{n-1} \cdot \mathrm{int}(x)
\end{aligned}
$$

Korrektheit des Verfahrens von Booth

Satz

Das Verfahren von Booth multipliziert sowohl positive als auch negative Zahlen

Beweis

Wir betrachten im Verfahren von Booth an jeder Stelle die Differenz $(y_{i-1}-y_i)$ und berechnen das Multiplikationsergebnisses durch die Summe

$$
\begin{aligned}
S \;=\; & (y_{-1}-y_0)\cdot 2^0\cdot \mathrm{int}(x) + \\
& (y_0 - y_1)\cdot 2^1\cdot \mathrm{int}(x) + \\
& \qquad \ldots \\
& (y_{n-3}-y_{n-2})\cdot 2^{n-2}\cdot \mathrm{int}(x) + \\
& (y_{n-2}-y_{n-1})\cdot 2^{n-1}\cdot \mathrm{int}(x) \\
= \; & \mathrm{int}(x)\cdot(-y_{n-1}\cdot 2^{n-1} + y_{n-2}\cdot 2^{n-2} + \ldots + y_1\cdot 2^1 + y_0\cdot 2^0)
\end{aligned}
$$

Korrektheit des Verfahrens von Booth

Satz

Das Verfahren von Booth multipliziert sowohl positive als auch negative Zahlen

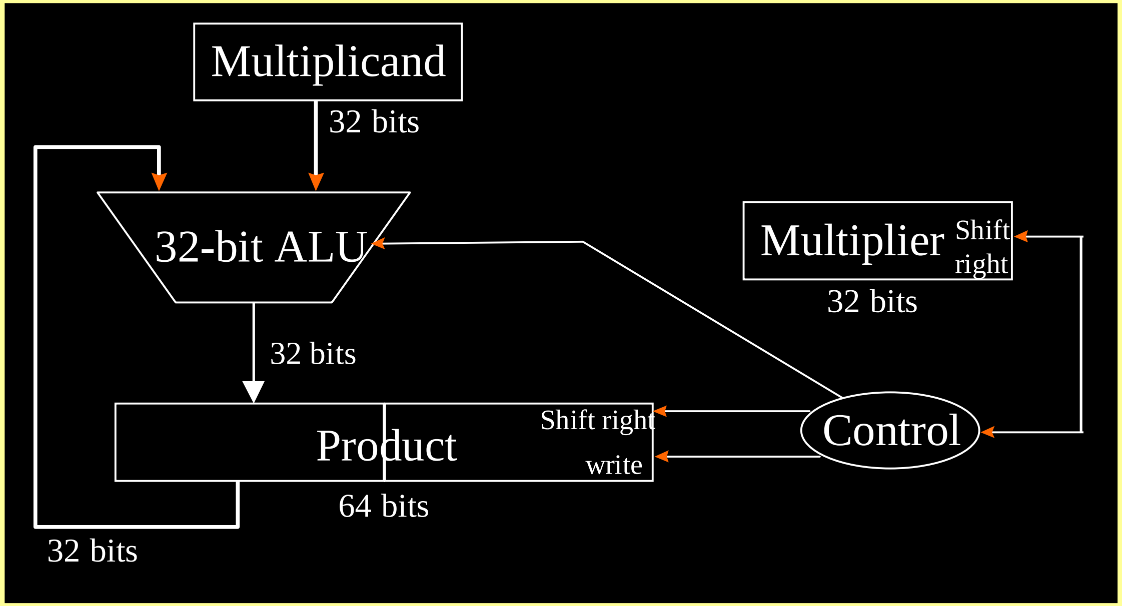
Beweis

Wir betrachten im Verfahren von Booth an jeder Stelle die Differenz $(y_{i-1}-y_i)$ und berechnen das Multiplikationsergebnisses durch die Summe

$$
\begin{aligned}
S \;&= (y_{-1}-y_0)\cdot 2^0\cdot \text{int}(x) + \\
&\quad (y_0 - y_1)\cdot 2^1\cdot \text{int}(x) + \\
&\quad\quad \ldots \\
&\quad (y_{n-3}-y_{n-2})\cdot 2^{n-2}\cdot \text{int}(x) + \\
&\quad (y_{n-2}-y_{n-1})\cdot 2^{n-1}\cdot \text{int}(x) \\
&= \text{int}(x)\cdot(-y_{n-1}\cdot 2^{n-1} + y_{n-2}\cdot 2^{n-2} + \ldots + y_1\cdot 2^1 + y_0\cdot 2^0) \\
&= \quad \text{int}(x) \cdot \text{int}(y)
\end{aligned}
$$

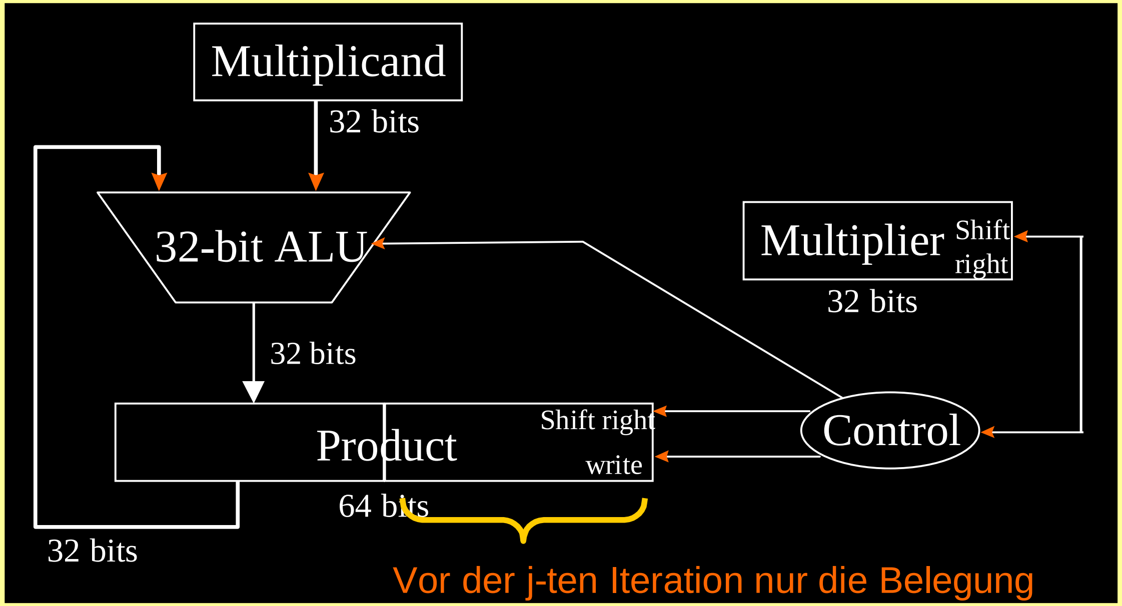
Hardwarerealisierung der Methode von Booth

- Multiplikation wird unter der Verwendung der Additionshardware implementiert:



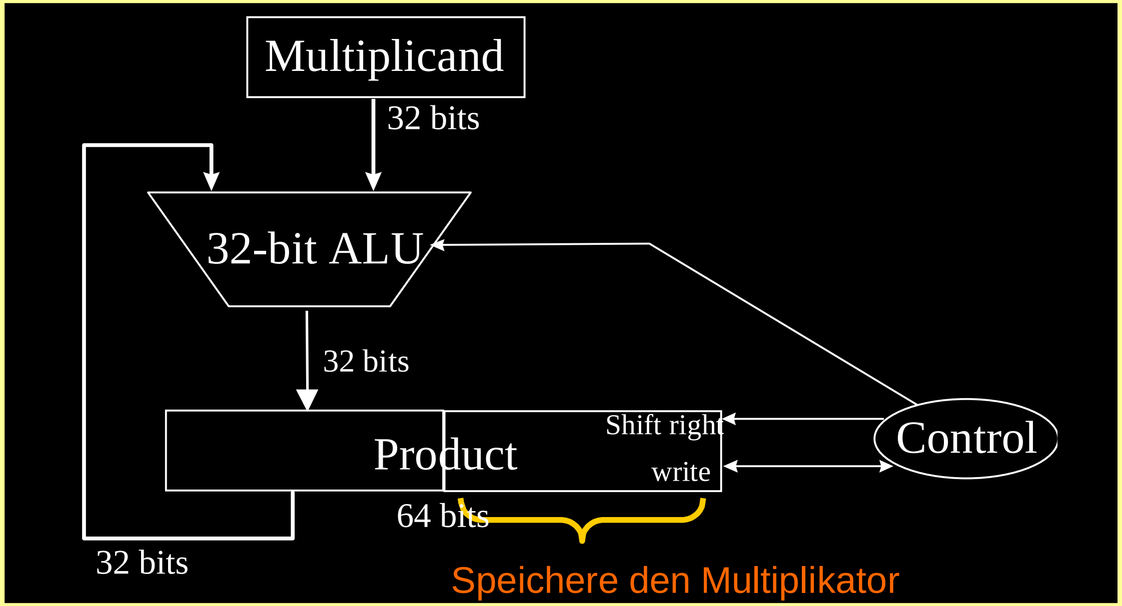
Hardwarerealisierung der Methode von Booth

- Multiplikation wird unter der Verwendung der Additionshardware implementiert:



Vor der j-ten Iteration nur die Belegung der j ersten Bitstellen von Bedeutung

Billigere Realisierung



Speichere den Multiplikator
in die hinteren 32 Bitstellen

Nachteil dieser Realisierung

Nachteil dieser Realisierung

- Vorgestellte Realisierung für die *n*-Bit Multiplikation benötigt $\geq n \cdot \log n$ Gatterlaufzeiten unter Benutzung eines schnellen Addierers

Nachteil dieser Realisierung

- Vorgestellte Realisierung für die *n*-Bit Multiplikation benötigt $\geq n \cdot \log n$ Gatterlaufzeiten unter Benutzung eines schnellen Addierers
- Es gibt effizientere Realisierungen für die *n*-Bit Multiplikation, die mit ungefähr $\log n$ Gatterlaufzeiten auskommen

- Division
  - restoring, non restoring
  - SRT-Division
  - iterative Verfahren: Newton-Verfahren, Goldschmidt-Verfahren

Division

- Schulmethode

```
10011010 : 1110 =
```

Division

- Schulmethode

```
10011010 : 1110 = 1
```

Division

- Schulmethode

```
10011010 : 1110 = 1
-1110
```

Division

- Schulmethode

```
10011010 : 1110 = 1
-1110
 ----
  101
```

Division

- Schulmethode

```
10011010 : 1110 = 1
-1110↓
 ────
  1010
```

Division

- Schulmethode

```
10011010 : 1110 = 10
-1110↓
  1010
```

Division

- Schulmethode

```
10011010 : 1110 = 10
-1110↓↓
  10101
```

Division

- Schulmethode

```
10011010 : 1110 = 101
-1110↓↓
  10101
```

Division

- Schulmethode

```
10011010 : 1110 = 101
-1110↓↓
-------
  10101
  -1110
  -----
```

Division

- Schulmethode

```
10011010 : 1110 = 101
-1110↓↓
  10101
  -1110
    111
```

Division

- Schulmethode

```
10011010 : 1110 = 101
-1110
-----
   10101
   -1110
   -----
     1110
```

Division

- Schulmethode

```
10011010 : 1110 = 1011
-1110
 -----
  10101
  -1110
  -----
    1110
```

Division

- Schulmethode

```
10011010 : 1110 = 1011
-1110
 10101
 -1110
   1110
  -1110
   0000
```

Division

- Abgewandelte Schulmethode

```
10011010 : 1110 =
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 =
-1110
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 =
-1110
 ----
  -101
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 =
-1110
 -101   <0
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
 -101   <0
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
 -101   <0
        restore
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
 ----
  -101   <0
+1110    restore
 ----
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
-----
  -101   <0
+1110    restore
-----
  1001
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
-----
  -101     <0
+1110      restore
-----
 1001
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
  -101    <0
+1110     restore
 10011
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
 ----
  -101    <0
+1110     restore
 ----
 10011
  -1110
  -----
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
 ----
  -101     <0
+1110      restore
 ----
 10011
 -1110
  ----
   101
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 0
-1110
 -101    <0
+1110    restore
 10011
 -1110
   101   ≥0
```

Division

- Abgewandelte Schulmethode

```
  10011010 : 1110 = 01
 -1110
  -101      <0
 +1110      restore
  10011
  -1110
    101     ≥0
```

Division

- Abgewandelte Schulmethode

```
 10011010 : 1110 = 01
-1110
 ----
  -101    <0
+1110     restore
 -----
 10011
  -1110
  -----
    101   ≥0
    ...
```

Division

- Abgewandelte Schulmethode

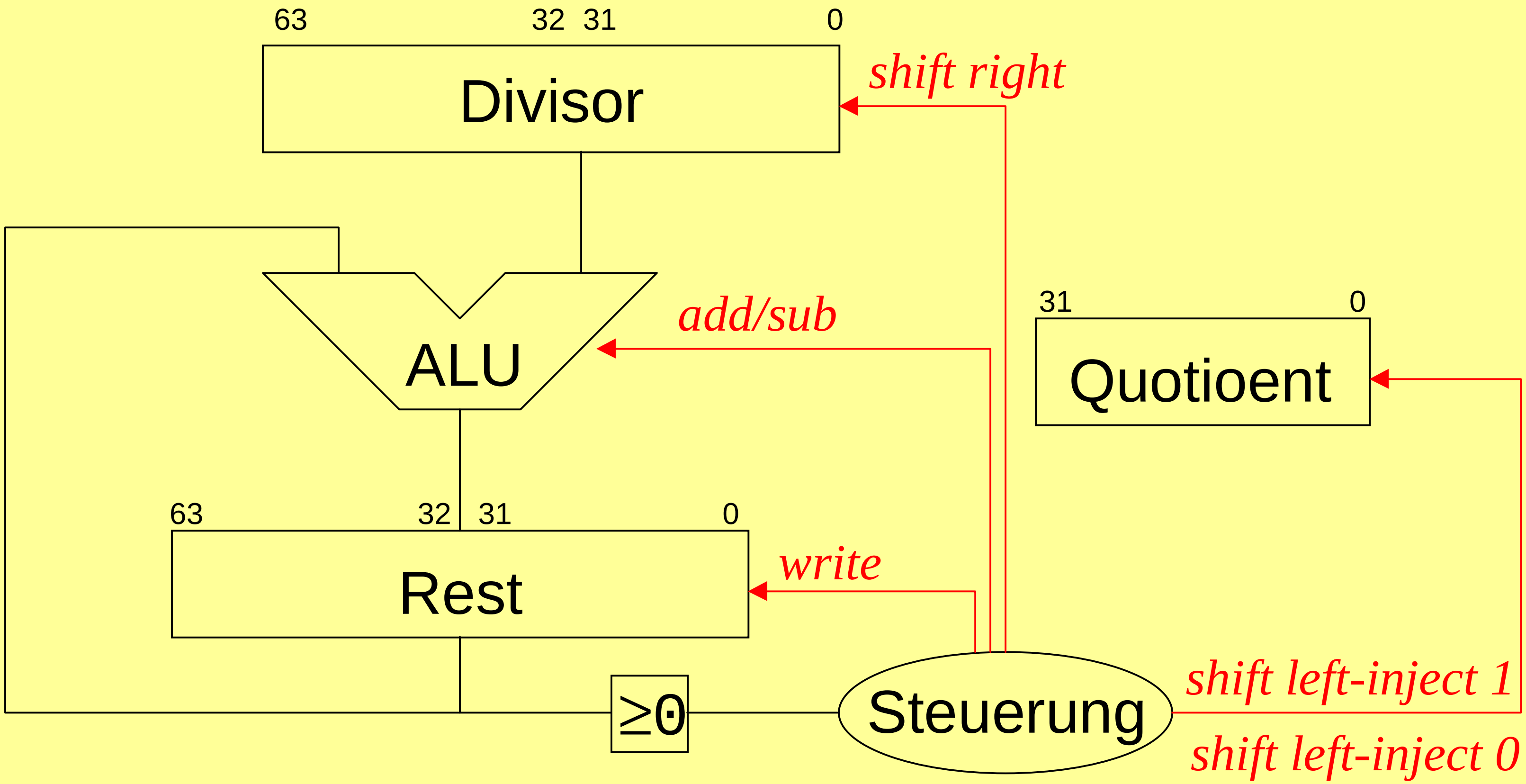
```
  10011010 : 1110 = 01...
 -1110
 -----
   -101    <0
 +1110     restore
 -----
  10011
  -1110
  -----
    101    ≥0
    ...
```

Division

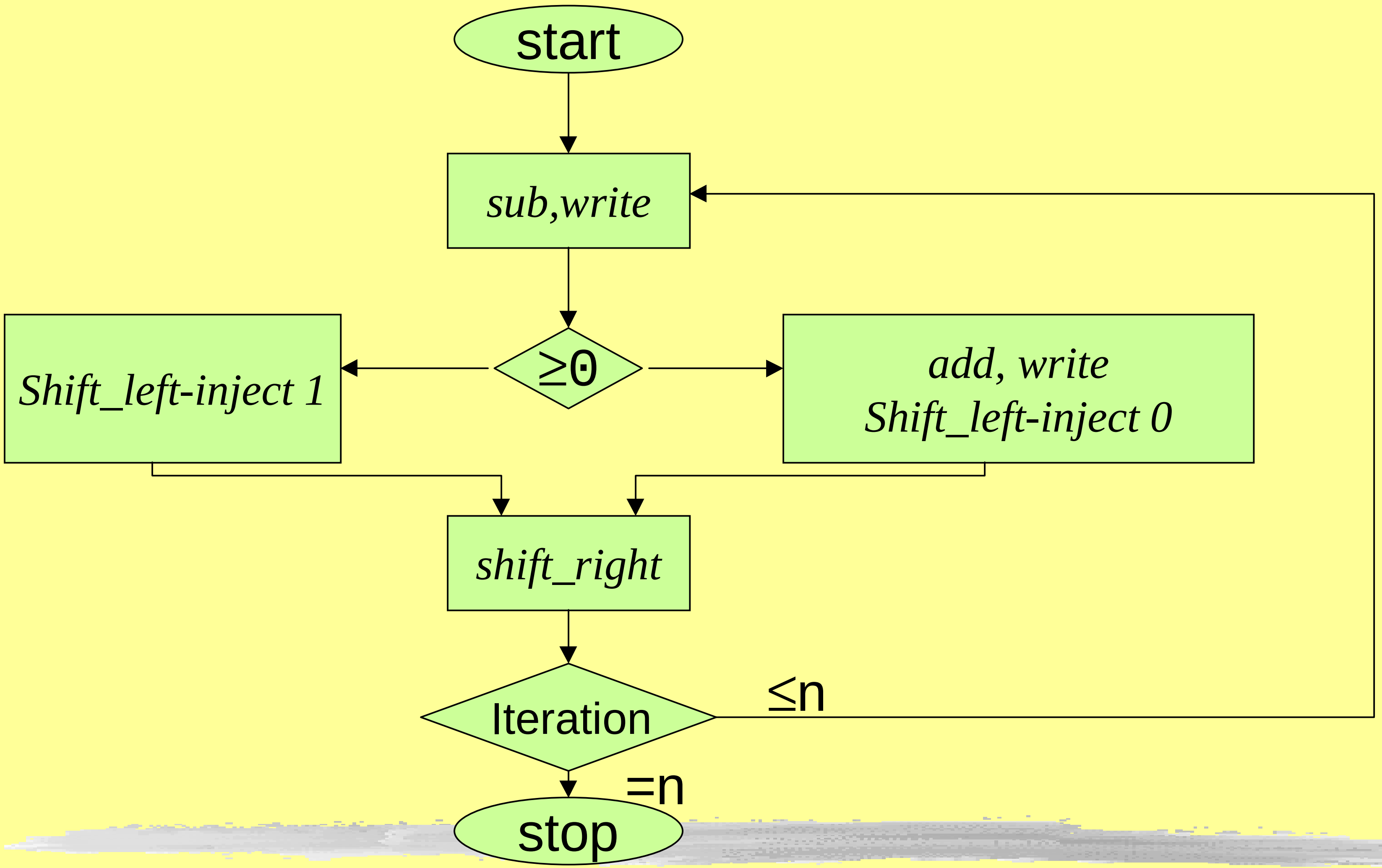
- Abgewandelte Schulmethode

```
Divisor  Dividend        Rest           Q
10011010-111000000000 = -110101100110 ⇒ 0
10011010- 11100000000 = -011001100110 ⇒ 0
10011010-  1110000000 = -001011100110 ⇒ 0
10011010-   111000000 = -000100100110 ⇒ 0
10011010-    11100000 = -000001000110 ⇒ 0
10011010-     1110000 =  000000101010 ⇒ 1
00101010-      111000 = -000000001110 ⇒ 0
00101010-       11100 =  000000001110 ⇒ 1
00001110-        1110 =  000000000000 ⇒ 1
```

Divisionswerk



Divisionswerk - Steuerung



- wrap-around Arithmetik: Überträge werden weggelassen
- bei Audio-, Videoanwendungen: Ausgabe des größten darstellbaren Wertes besser

---> Sättigungsarithmetik

auf sie kann bei vielen Prozessoren für die digitale Signalverarbeitung umgeschaltet werden

	Wrap-around Arithmetik	Sättigungs-Arithmetik
a	"1000"	"1000"
b	"1000"	"1000"
a+b	"0000"	"1111"

Probleme mit Festkommazahlen

Probleme mit Festkommazahlen

Bei Zweierkomplement-Darstellung mit *n* Stellen vor und *k* hinter dem Komma

Probleme mit Festkommazahlen

Bei Zweierkomplement-Darstellung mit *n* Stellen vor und *k* hinter dem Komma

- keine ganz grossen bzw. kleinen Zahlen darstellbar !

  Zahlen mit grösstem Absolutbetrag: $(2^{n+k-1}-1)/2^k$ und -2^{n-1}

  z.B. $0111.1111_b = 7.9375$ und $1000.0000_b = -8$

  Zahlen mit kleinstem Absolutbetrag: -2^{-k} und 2^{-k}

  z.B. $0000.0001_b = 0.0625$ und $1111.1111_b = -0.0625$

Probleme mit Festkommazahlen

Bei Zweierkomplement-Darstellung mit *n* Stellen vor und *k* hinter dem Komma

- keine ganz grossen bzw. kleinen Zahlen darstellbar !

  Zahlen mit grösstem Absolutbetrag: $(2^{n+k-1}-1)/2^k$ und -2^{n-1}

  z.B. $0111.1111_b = 7.9375$ und $1000.0000_b = -8$

  Zahlen mit kleinstem Absolutbetrag: -2^{-k} und 2^{-k}

  z.B. $0000.0001_b = 0.0625$ und $1111.1111_b = -0.0625$

- Operationen sind nicht abgeschlossen !

  $2^{n-1}+2^{n-1}$ ist nicht darstellbar, obwohl die Operanden darstellbar sind.

Probleme mit Festkommazahlen

Bei Zweierkomplement-Darstellung mit *n* Stellen vor und *k* hinter dem Komma

- keine ganz grossen bzw. kleinen Zahlen darstellbar !
  - Zahlen mit grösstem Absolutbetrag: $(2^{n+k-1}-1)/2^k$ und -2^{n-1}
    - z.B. $0111.1111_b = 7.9375$ und $1000.0000_b = -8$
  - Zahlen mit kleinstem Absolutbetrag: -2^{-k} und 2^{-k}
    - z.B. $0000.0001_b = 0.0625$ und $1111.1111_b = -0.0625$
- Operationen sind nicht abgeschlossen !
  $2^{n-1}+2^{n-1}$ ist nicht darstellbar, obwohl die Operanden darstellbar sind.
- Assoziativgesetz und Distributivgesetz gelten nicht !
  $(2^{n-1}+2^{n-1})-2^{n-1} \neq 2^{n-1}+(2^{n-1}-2^{n-1})$

ZK.2 Gleitkommazahlen

Gleitkommadarstellung

Gleitkommadarstellung

Idee: Repräsentiere Zahl durch Vorzeichen, Exponent und Mantisse,
Position des Kommas liegt also nicht fest!
Abdeckung eines größeren Zahlenbereichs bei gegebener Stellenanzahl

Gleitkommadarstellung

Idee: Repräsentiere Zahl durch Vorzeichen, Exponent und Mantisse,
Position des Kommas liegt also nicht fest!
Abdeckung eines größeren Zahlenbereichs bei gegebener Stellenanzahl

- Gleitkommadarstellung einfacher Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

Gleitkommadarstellung

Idee: Repräsentiere Zahl durch Vorzeichen, Exponent und Mantisse,
Position des Kommas liegt also nicht fest!
Abdeckung eines größeren Zahlenbereichs bei gegebener Stellenanzahl

- Gleitkommadarstellung einfacher Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse M

Gleitkommadarstellung

Idee: Repräsentiere Zahl durch Vorzeichen, Exponent und Mantisse,
Position des Kommas liegt also nicht fest!
Abdeckung eines größeren Zahlenbereichs bei gegebener Stellenanzahl

- Gleitkommadarstellung einfacher Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse M

- Gleitkommadarstellung doppelter Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

Gleitkommadarstellung

Idee: Repräsentiere Zahl durch Vorzeichen, Exponent und Mantisse,
Position des Kommas liegt also nicht fest!
Abdeckung eines größeren Zahlenbereichs bei gegebener Stellenanzahl

- Gleitkommadarstellung einfacher Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse M

- Gleitkommadarstellung doppelter Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

63	62 61 . . . 53 52	51 50 . . . 3 2 1 0
S	Exponent E	Mantisse M

Gleitkommadarstellung

Idee: Repräsentiere Zahl durch Vorzeichen, Exponent und Mantisse,
Position des Kommas liegt also nicht fest!
Abdeckung eines größeren Zahlenbereichs bei gegebener Stellenanzahl

- Gleitkommadarstellung einfacher Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse M

- Gleitkommadarstellung doppelter Genauigkeit: $(-1)^S \cdot M \cdot 2^E$

63	62 61 . . . 53 52	51 50 . . . 3 2 1 0
S	Exponent E	Mantisse M

- Es bleibt noch festzulegen, wie dieMantissenbits bzw. Exponentenbits als Zahlen M bzw. E interpretiert werden sollen.

Normalisierte Gleitkommadarstellungen

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Definition

Eine Gleitkommazahl (S,M,E) heisst **normalisiert**, wenn $1 \leq M < 2$

d.h. wenn M von der Form $1.m_{-1} \ldots m_{-k}$ ist.

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Definition

Eine Gleitkommazahl (S,M,E) heisst **normalisiert**, wenn $1 \leq M < 2$

d.h. wenn M von der Form $1.m_{-1} \ldots m_{-k}$ ist.

Die 1 vor dem Komma braucht nicht abgespeichert zu werden ($\rightarrow$ „**hidden bit**“)

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Definition

Eine Gleitkommazahl (S,M,E) heisst **normalisiert**, wenn $1 \leq M < 2$

d.h. wenn M von der Form $1.m_{-1} \ldots m_{-k}$ ist.

Die 1 vor dem Komma braucht nicht abgespeichert zu werden ($\rightarrow$ „**hidden bit**")

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Definition

Eine Gleitkommazahl (S,M,E) heisst **normalisiert**, wenn $1 \leq M < 2$

d.h. wenn M von der Form $1.m_{-1} \ldots m_{-k}$ ist.

Die 1 vor dem Komma braucht nicht abgespeichert zu werden (→ „**hidden bit**")

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m

$m_{-1}\, m_{-2}$ ... ... m_{-23}

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Definition

Eine Gleitkommazahl (S,M,E) heisst **normalisiert**, wenn $1 \leq M < 2$

d.h. wenn M von der Form $1.m_{-1} \ldots m_{-k}$ ist.

Die 1 vor dem Komma braucht nicht abgespeichert zu werden (→ „**hidden bit**")

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m

$m_{-1}\, m_{-2}$ ... ... m_{-23}

Für eine normalisierte Gleitkommazahl ergibt sich der Mantissenwert M als $M = 1 + \Sigma_{i=-1,\ldots,-k}\, m_i\, 2^i$.

Normalisierte Gleitkommadarstellungen

Beobachtung

Gleitkommadarstellung einer Zahl ist nicht eindeutig !

$0.111 \cdot 2^3 = 0.0111 \cdot 2^4$

Definition

Eine Gleitkommazahl (S,M,E) heisst **normalisiert**, wenn $1 \leq M < 2$

d.h. wenn M von der Form $1.m_{-1} \ldots m_{-k}$ ist.

Die 1 vor dem Komma braucht nicht abgespeichert zu werden (→ „**hidden bit**")

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m

$m_{-1}\, m_{-2}$ ... ... m_{-23}

Für eine normalisierte Gleitkommazahl ergibt sich der Mantissenwert M als $M = 1 + \Sigma_{i=-1,\ldots,-k}\, m_i\, 2^i$.

⇒ Die Zahl 0 muß als Spezialfall behandelt werden !

Gleitkommadarstellung - IEEE 754 Standard

Gleitkommadarstellung - IEEE 754 Standard

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m

Gleitkommadarstellung - IEEE 754 Standard

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m
	e_7 e_6 ... e_0	

Gleitkommadarstellung - IEEE 754 Standard

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m
	e_7 e_6 ... e_0	

- Gemäß IEEE 754-Standard werden die Exponentenbits als vorzeichenlose Zahl interpretiert.

Gleitkommadarstellung - IEEE 754 Standard

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m

e_7 e_6 ... e_0

- Gemäß IEEE 754-Standard werden die Exponentenbits als vorzeichenlose Zahl interpretiert.
- Um auch negative Exponenten darstellen zu können, wird von der Interpretation als vorzeichenlose Zahl eine Konstante, der sogenannte **Bias**, subtrahiert.

Gleitkommadarstellung - IEEE 754 Standard

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m
	e_7 e_6 ... e_0	

- Gemäß IEEE 754-Standard werden die Exponentenbits als vorzeichenlose Zahl interpretiert.
- Um auch negative Exponenten darstellen zu können, wird von der Interpretation als vorzeichenlose Zahl eine Konstante, der sogenannte **Bias**, subtrahiert.
- Bei *n* Exponentenbits wird der Bias gewählt als **BIAS = 2^{n-1}-1**, also bei einfacher Genauigkeit BIAS = 127, bei doppelter Genauigkeit BIAS = 1023.

Gleitkommadarstellung - IEEE 754 Standard

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m
	e_7 e_6 ... e_0	

- Gemäß IEEE 754-Standard werden die Exponentenbits als vorzeichenlose Zahl interpretiert.
- Um auch negative Exponenten darstellen zu können, wird von der Interpretation als vorzeichenlose Zahl eine Konstante, der sogenannte **Bias**, subtrahiert.
- Bei *n* Exponentenbits wird der Bias gewählt als **BIAS = 2^{n-1}-1**, also bei einfacher Genauigkeit BIAS = 127, bei doppelter Genauigkeit BIAS = 1023.
- Bei *n* Exponentenbits ergibt sich also für E:

Gleitkommadarstellung - IEEE 754 Standard

31	30 29 28 27 26 25 24 23	22 21 . . . 3 2 1 0
S	Exponent E	Mantisse m
	e_7 e_6 ... e_0	

- Gemäß IEEE 754-Standard werden die Exponentenbits als vorzeichenlose Zahl interpretiert.
- Um auch negative Exponenten darstellen zu können, wird von der Interpretation als vorzeichenlose Zahl eine Konstante, der sogenannte **Bias**, subtrahiert.
- Bei *n* Exponentenbits wird der Bias gewählt als **BIAS = 2^{n-1}-1**, also bei einfacher Genauigkeit BIAS = 127, bei doppelter Genauigkeit BIAS = 1023.
- Bei *n* Exponentenbits ergibt sich also für E:

$$E = \sum_{i=0,\ldots,n-1} e_i 2^i - BIAS$$

Sonderfälle IEEE 754 Standard

Sonderfälle IEEE 754 Standard

- **Der Exponent 0 spielt beim IEEE 754-Standard eine Sonderrolle**: Sind alle Exponentenbits 0, so wird **ausnahmsweise** das „hidden bit" der Mantissendarstellung weggelassen, so daß die Zahl

$$\left(\sum_{i=-1,\dots,-k} m_i 2^i\right) 2^{-126}$$

dargestellt wird.

Sonderfälle IEEE 754 Standard

- **Der Exponent 0 spielt beim IEEE 754-Standard eine Sonderrolle**: Sind alle Exponentenbits 0, so wird **ausnahmsweise** das „hidden bit" der Mantissendarstellung weggelassen, so daß die Zahl

$$(\Sigma_{i=-1,\ldots,-k} m_i 2^i)\ 2^{-126}$$

dargestellt wird.

- Auf diese Weise können „**denormalisierte Zahlen**" dargestellt werden, die kleiner als die kleinste darstellbare normalisierte Zahl sind.

Sonderfälle IEEE 754 Standard

- **Der Exponent 0 spielt beim IEEE 754-Standard eine Sonderrolle**: Sind alle Exponentenbits 0, so wird **ausnahmsweise** das „hidden bit" der Mantissendarstellung weggelassen, so daß die Zahl

$$\left(\Sigma_{i=-1,\ldots,-k}\, m_i 2^i\right) 2^{-126}$$

  dargestellt wird.

- Auf diese Weise können „**denormalisierte Zahlen**" dargestellt werden, die kleiner als die kleinste darstellbare normalisierte Zahl sind.

- Die **Null** wird folgendermaßen dargestellt: Sämtliche Mantissenbits und Exponentenbits sind 0.

Sonderfälle IEEE 754 Standard

- **Der Exponent 0 spielt beim IEEE 754-Standard eine Sonderrolle**: Sind alle Exponentenbits 0, so wird **ausnahmsweise** das „hidden bit" der Mantissendarstellung weggelassen, so daß die Zahl

$$\left(\Sigma_{i=-1,\dots,-k}\, m_i\, 2^i\right)\, 2^{-126}$$

dargestellt wird.

- Auf diese Weise können „**denormalisierte Zahlen**" dargestellt werden, die kleiner als die kleinste darstellbare normalisierte Zahl sind.

- Die **Null** wird folgendermaßen dargestellt: Sämtliche Mantissenbits und Exponentenbits sind 0.

- **Der Exponent 2^n-1 spielt ebenfalls eine Sonderrolle**: Sind alle Exponentenbits 1 und alle Mantissenbits 0, so wird der Wert ∞ dargestellt.

IEEE 754 Standard - Spezialfälle

Normalisierte Zahl	$\pm$	$0 < e < 255$ (4095)	m beliebig
Denormalisierte Zahl	$\pm$	0	$m \neq 0$ beliebig
Null	$\pm$	0	0
Unendlich	$\pm$	255 (4095)	0
Not a Number	$\pm$	255 (4095)	$m \neq 0$ beliebig

Darstellbare normalisierte Gleitkommazahlen

	single precision	double precision
Vorzeichenstellen	1	1
Exponentenstellen	8	11
Mantissenstellen (ohne hidden Bit)	23	52
Bitstellen insgesamt	32	64
Bias	127	1023
Exponentenbereich	-126 bis 127	-1022 bis 1023
Darstellbare normalisierte Zahl mit kleinstem Absolutbetrag	2^{-126}	2^{-1022}
Darstellbare normalisierte Zahl mit größtem Absolutbetrag	$(1-2^{-24})\ 2^{128}$	$(1-2^{-53})\ 2^{1024}$
Darstellbare denormalisierte Zahl mit kleinstem Absolutbetrag	2^{-149}	2^{-1074}
Darstellbare denormalisierte Zahl mit größtem Absolutbetrag	$(1-2^{-23})\ 2^{-126}$	$(1-2^{-52})\ 2^{-1022}$

IEEE 754 Standard - Eigenschaften

IEEE 754 Standard - Eigenschaften

- Eindeutige Zahlendarstellung, falls auf normalisierte Darstellungen beschränkt

IEEE 754 Standard - Eigenschaften

- Eindeutige Zahlendarstellung, falls auf normalisierte Darstellungen beschränkt
- Nicht alle Zahlen zwischen der kleinsten und grössten darstellbaren Zahl sind darstellbar.

IEEE 754 Standard - Eigenschaften

- Eindeutige Zahlendarstellung, falls auf normalisierte Darstellungen beschränkt
- Nicht alle Zahlen zwischen der kleinsten und grössten darstellbaren Zahl sind darstellbar.
- Je näher bei der Null, desto dichter liegen die darstellbaren Zahlen.

IEEE 754 Standard - Eigenschaften

- Eindeutige Zahlendarstellung, falls auf normalisierte Darstellungen beschränkt
- Nicht alle Zahlen zwischen der kleinsten und grössten darstellbaren Zahl sind darstellbar.
- Je näher bei der Null, desto dichter liegen die darstellbaren Zahlen.
- Arithmetische Operationen sind nicht abgeschlossen!

IEEE 754 Standard - Eigenschaften

- Eindeutige Zahlendarstellung, falls auf normalisierte Darstellungen beschränkt
- Nicht alle Zahlen zwischen der kleinsten und grössten darstellbaren Zahl sind darstellbar.
- Je näher bei der Null, desto dichter liegen die darstellbaren Zahlen.
- Arithmetische Operationen sind nicht abgeschlossen!
- Assoziativgesetz und Distributivgesetz gelten nicht, da bei Anwendung der Gesetze evtl. der darstellbare Zahlenbereich verlassen wird!

Addition von Gleitkommazahlen

Addition von Gleitkommazahlen

Rechenvorschrift

Addition von Gleitkommazahlen

Rechenvorschrift

- Angleichung des kleineren an den grösseren Exponenten

Addition von Gleitkommazahlen

Rechenvorschrift

- Angleichung des kleineren an den grösseren Exponenten
- Addition der Mantissen

Addition von Gleitkommazahlen

Rechenvorschrift

- Angleichung des kleineren an den grösseren Exponenten
- Addition der Mantissen
- Normalisierung (falls erforderlich)

Addition von Gleitkommazahlen

Rechenvorschrift

- Angleichung des kleineren an den grösseren Exponenten
- Addition der Mantissen
- Normalisierung (falls erforderlich)

Beispiel

Addition von Gleitkommazahlen

Rechenvorschrift

- Angleichung des kleineren an den grösseren Exponenten
- Addition der Mantissen
- Normalisierung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{-1} + -(1.110)_2 \cdot 2^{-2} \quad = +(1.000)_2 \cdot 2^{-1} + -(0.111)_2 \cdot 2^{-1}$

Addition von Gleitkommazahlen

Rechenvorschrift

- Angleichung des kleineren an den grösseren Exponenten
- Addition der Mantissen
- Normalisierung (falls erforderlich)

Beispiel

$$+(1.000)_2 \cdot 2^{-1} + -(1.110)_2 \cdot 2^{-2} = +(1.000)_2 \cdot 2^{-1} + -(0.111)_2 \cdot 2^{-1}$$

$$= +(0.001)_2 \cdot 2^{-1}$$

Addition von Gleitkommazahlen

Rechenvorschrift

- Angleichung des kleineren an den grösseren Exponenten
- Addition der Mantissen
- Normalisierung (falls erforderlich)

Beispiel

$$+(1.000)_2 \cdot 2^{-1} + -(1.110)_2 \cdot 2^{-2} = +(1.000)_2 \cdot 2^{-1} + -(0.111)_2 \cdot 2^{-1}$$
$$= +(0.001)_2 \cdot 2^{-1}$$
$$= +(1.000)_2 \cdot 2^{-4}$$

Multiplikation von Gleitkommazahlen

Multiplikation von Gleitkommazahlen

Rechenvorschrift

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Exponenten und substrahiere (einmal) den Bias-Wert

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Exponenten und substrahiere (einmal) den Bias-Wert
- Normalisierung (falls erforderlich)

Beispiel

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Exponenten und substrahiere (einmal) den Bias-Wert
- Normalisierung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{-1+BIAS} \times -(1.110)_2 \cdot 2^{-2+BIAS}$

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Exponenten und substrahiere (einmal) den Bias-Wert
- Normalisierung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{-1+BIAS} \times -(1.110)_2 \cdot 2^{-2+BIAS}$

Multiplikation der Vorzeichen: $0 \oplus 1 = 1$

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Exponenten und substrahiere (einmal) den Bias-Wert
- Normalisierung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{-1+BIAS} \times -(1.110)_2 \cdot 2^{-2+BIAS}$

Multiplikation der Vorzeichen: $0 \oplus 1 = 1$

Multiplikation der Mantissen: $(1.000)_2 \times (1.110)_2 = (1.110)_2$

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Exponenten und substrahiere (einmal) den Bias-Wert
- Normalisierung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{-1+BIAS} \times -(1.110)_2 \cdot 2^{-2+BIAS}$

Multiplikation der Vorzeichen: $0 \oplus 1 = 1$

Multiplikation der Mantissen: $(1.000)_2 \times (1.110)_2 = (1.110)_2$

Addition der Exponenten: $(-1+BIAS)+(-2+BIAS)-BIAS = (-3+BIAS)$

Multiplikation von Gleitkommazahlen

Rechenvorschrift

- Multipliziere die Vorzeichen
- Multipliziere die beiden Mantissen
- Addiere die beiden Exponenten und substrahiere (einmal) den Bias-Wert
- Normalisierung (falls erforderlich)

Beispiel

$+(1.000)_2 \cdot 2^{-1+BIAS} \times -(1.110)_2 \cdot 2^{-2+BIAS}$

Multiplikation der Vorzeichen: $0 \oplus 1 = 1$

Multiplikation der Mantissen: $(1.000)_2 \times (1.110)_2 = (1.110)_2$

Addition der Exponenten: $(-1+BIAS)+(-2+BIAS)-BIAS = (-3+BIAS)$

Resultat: $-(1.110)_2 \cdot 2^{-3+BIAS}$

Multiplikation