— Informatik I —
Modul 5: Rechnerarithmetik (2)

Universität Zürich[UZH]

Modul 5: Rechnerarithmetik (2)

- Grundrechenarten
- Arithmetisch-logische Einheit

Addition und Subtraktion

- Schaltungen zur Addition von Festkomma-Dualzahlen:
  - Grundlage für die Durchführung aller arithmetischen Verknüpfungen
- Denn:
  - Subtraktion ≙ Addition der negativen Zahl
  - X - Y = X + (-Y)
- Multiplikation und Division lassen sich ebenfalls auf die Addition zurückführen.
  - Bei Gleitkommazahlen:
    - Mantisse und Exponent werden separat verarbeitet.
    - Hierbei bildet die Addition von Festkomma-Dualzahlen die Grundlage.
- Grundtypen von Addierern sind wichtig

Vom Halbaddierer zum Volladdierer

Bei der Addition zweier Dualzahlen (a und b):

Summe (s) und Übertrag (ü) entstehen als Ergebnis.

Funktionstabelle:

a	b	s	ü
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

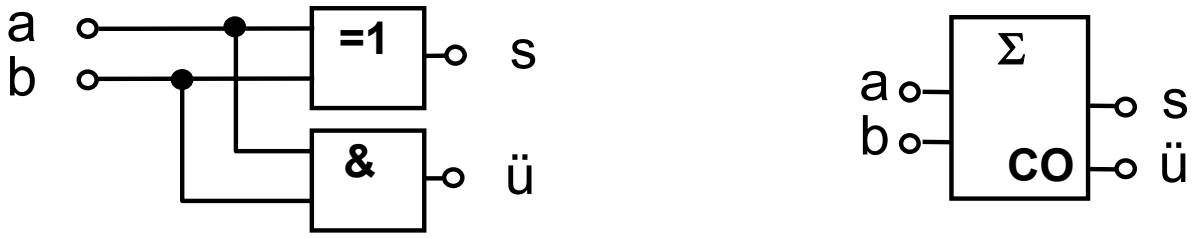
Man nennt dies einen Halbaddierer

Halbaddierer

Gleichungen: $s = a \bar{b} \vee \bar{a} b = a \not\leftrightarrow b$

$ü = a b$

Das Schaltbild und das Schaltsymbol:



Schaltbild und Schaltsymbol eines 1-Bit-Halbaddierers

Mehrstellige Dualzahlen

Zusätzlicher Eingang für den Übertrag der vorhergehenden Stellen ist nötig.

a_i	b_i	$ü_i$	s_i	$ü_{i+1}$
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

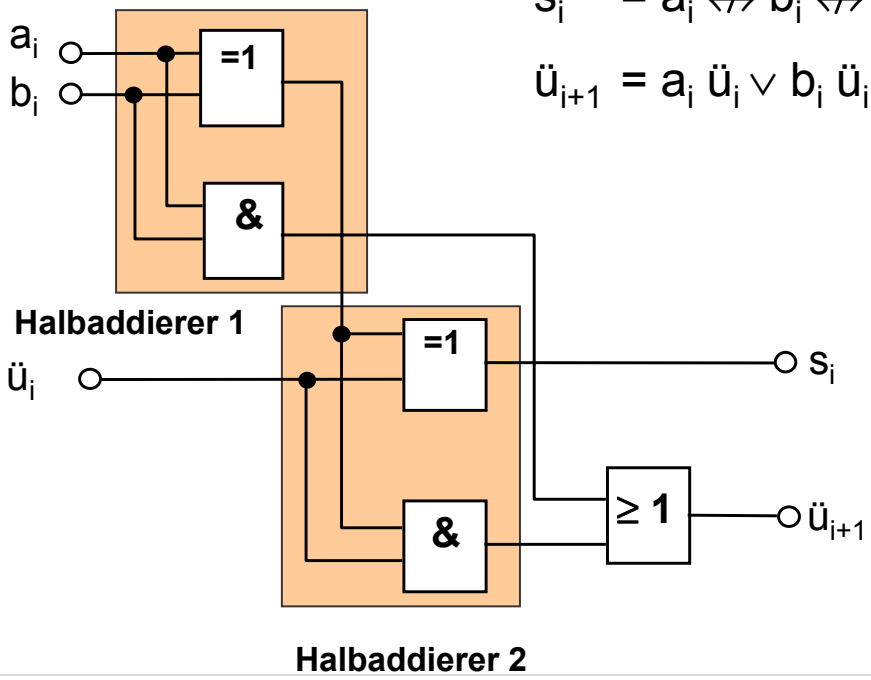
Dies nennt man einen Volladdierer

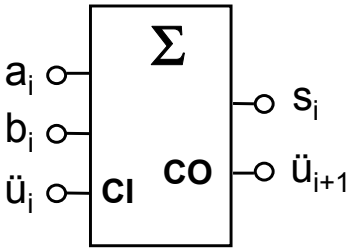
Gleichungen, Schaltnetz und Schaltsymbol

Ausgangsgleichungen:

$$s_i = a_i \leftrightarrow b_i \leftrightarrow \ddot{u}_i$$

$$\ddot{u}_{i+1} = a_i\,\ddot{u}_i \vee b_i\,\ddot{u}_i \vee a_i\,b_i = (a_i \leftrightarrow b_i)\,\ddot{u}_i \vee a_i\,b_i$$





Volladdierer

Carry-lookahead-Addierer (1)

- Alle Überträge direkt aus den Eingangsvariablen bestimmen (carry-lookahead).
  - Vermeidet Nachteil der großen Additionszeit des Carry-ripple-Addierers

- Es gilt:

$$\ddot{u}_{i+1} = a_i\,b_i \vee (a_i \leftrightarrow b_i)\,\ddot{u}_i$$
$$= g_i \vee p_i\,\ddot{u}_i$$
$$s_i = (a_i \leftrightarrow b_i) \leftrightarrow \ddot{u}_i$$
$$= p_i \leftrightarrow \ddot{u}_i$$

- mit

$g_i = a_i\,b_i$ (generate carry, erzeuge Übertrag) und

$p_i = (a_i \leftrightarrow b_i)$ (propagate carry, leite Übertrag weiter)

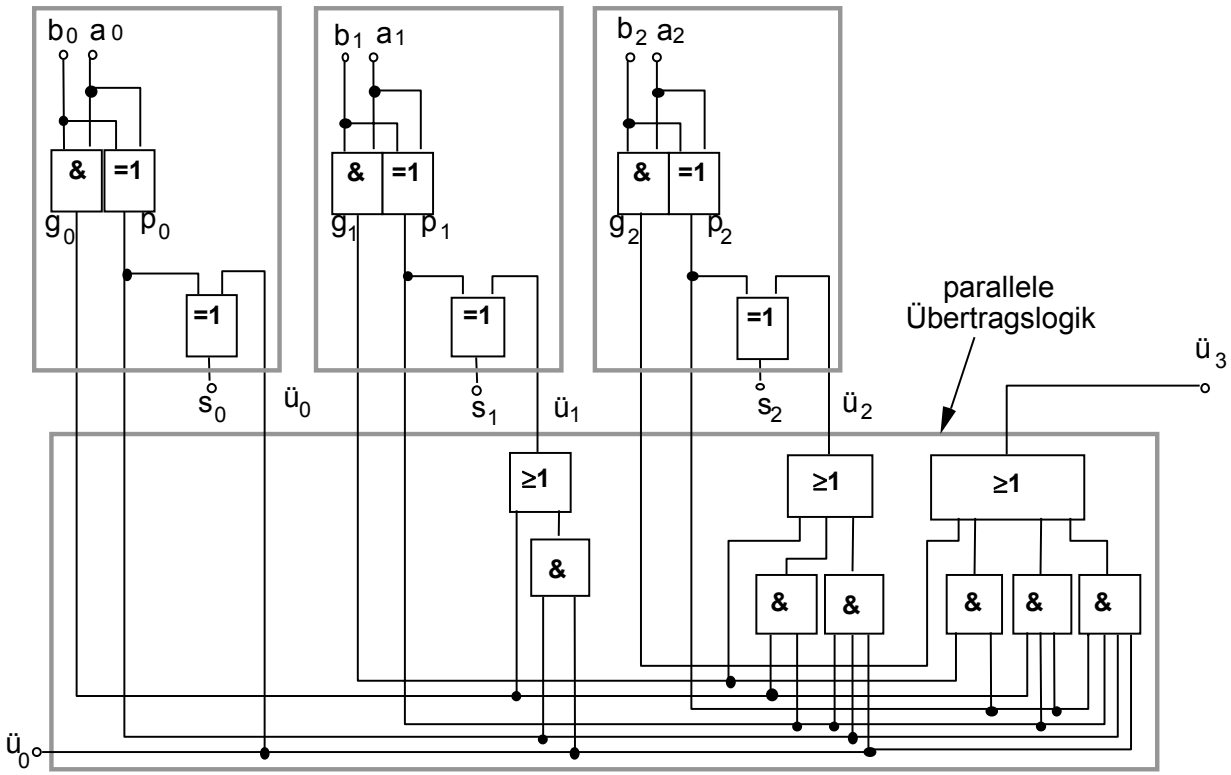
- g_i und p_i können direkt aus den Eingangsvariablen erzeugt werden.

Carry-lookahead-Addierer (2)

- Problem:
  - Größe des Hardware-Aufwands steigt mit steigender Stellenzahl stark an.

- Lösungen:
  - kleinere Carry-lookahead-Addierer mit paralleler Übertragserzeugung, die seriell kaskadiert werden
  - Blocküberträge der kleineren Blöcke parallel verarbeiten
  - Hierarchie von Carry-lookahead-Addierern.

Schaltbild: 3-Bit-Carry-lookahead-Addierer



Überlauferkennung

- Allgemeine Überlauferkennung bei dualer Addition:
  - Korrekte Addition: beide Überträge sind gleich.
  - Überlauf: beide Überträge sind ungleich.
- Realisierung z.B. durch ein Antivalenzgatter

Multiplikation und Division

- Zweierkomplementzahlen erwiesen sich für Addition und Subtraktion als besonders günstig, weil bei dieser Darstellung das Vorzeichen nicht explizit betrachtet werden mußte.
- Bei der Multiplikation existiert dieser Vorteil nicht.
- Lösungen:
  - Zweierkomplementzahlen zunächst in eine Form mit Betrag und Vorzeichen umwandeln. Zahlen dann miteinander zu multiplizieren und das Ergebnis schließlich wieder in die Zweierkomplementdarstellung umzusetzen.
  - Spezielle Multiplikationsalgorithmen für Zweierkomplementzahlen verwenden (Booth-Algorithmus, hier nicht tiefergehend behandelt).

Multiplikation

- Zur Multiplikation von Gleitkommazahlen muß man Mantissen beider Zahlen multiplizieren und ihre Exponenten addieren:

$$m_1\, b^{e1} \cdot m_2\, b^{e2} = (m_1 \cdot m_2)\, b^{e1\,+\,e2}$$

- Ist die Mantisse mit Hilfe von Betrag und Vorzeichen dargestellt,

  → ist der übliche Multiplikationsalgorithmus anwendbar.

- Das Ergebnis muß nach der Multiplikation unter Umständen noch normalisiert werden.

- Bei Addition der Charakteristiken $\mathbf{c_1 = e_1 + o}$ und $\mathbf{c_2 = e_2 + o}$ muß die Summe außerdem um den Offset **o** korrigiert werden, um die richtige Ergebnischarakteristik $\mathbf{c = (e_1 + e_2) + o}$ zu erhalten.

Parallele und serielle Multiplikation

- Papier und Bleistift Methode:
  - Analog zur Multiplikation mit Papier und Bleistift im Dezimalsystem kann man auch im Dualsystem vorgehen.

- Beispiele:

```
13 · 11          1101 · 1011
   13               1101
+  13                0000
  143               1101
                  +  1101
                  10001111
```

Vorzeichen-Betrag-Zahlen

- Bei Zahlen, die mit Betrag und Vorzeichen dargestellt sind, ergeben sich keine Probleme.
- Die Beträge der Zahlen werden wie positive Zahlen miteinander multipliziert.
- Das Vorzeichen des Ergebnisses ergibt sich aus der Antivalenzverknüpfung der Vorzeichen der beiden Faktoren.

Vorzeichenbehaftete Multiplikation

- Vorzeichenbehaftete Zahlen können grundsätzlich in die Darstellung mit Vorzeichen und Betrag gebracht werden.
- Das Vorzeichen des Produkts wird dann nach der Regel

  $$\text{sign}(X \bullet Y) = \text{sign}(X) \ \text{XOR} \ \text{sign}(Y)$$

  aus den beiden Faktorenvorzeichen durch die Exklusiv-ODER-Verknüpfung gewonnen.

Anmerkungen

- Bei vielen Anwendungen jedoch wechseln Addition und Multiplikation einander ständig ab.
- Umwandlung zwischen verschiedenen Zahlendarstellungen kann viel Zeit in Anspruch nehmen.
- Es wäre günstiger, wenn durchgängig (also auch bei der Multiplikation) im Zweierkomplement gerechnet wird, um die Vorteile bei der Addition nutzen zu können (z.B. Booth-Algorithmus).

Division

- Die Division von Dualzahlen folgt denselben Prinzipien wie die Multiplikation.
- Auch hier stellt die Papier-und-Bleistift-Methode die Basis für verschiedene Algorithmen dar.
- Beispiel:

$$
\begin{array}{rl}
1224 & : \; 12 \;=\; 102 \\
-\,12\downarrow & \\
\hline
02\downarrow & \\
024 & \\
-\;24 & \\
\hline
0 &
\end{array}
$$

Manuelle Division

- Im Dualsystem sind als Ergebnis einer Teildivison nur die Werte
  0 → Divisor > augenblicklicher Dividend
  1 → Divisor ≤ augenblicklicher Dividend
möglich.

- Bei manueller Division erkennt man sofort, ob das Ergebnis 0 ist und „eine weitere Stelle gebraucht wird“.

Maschinelle Division

- Drei Möglichkeiten:
  - 1. Komparatorschaltung, um den Divisor mit augenblicklichem Dividenden zu vergleichen.
  - 2. Subtraktion: Ergibt sich ein negatives Ergebnis, lädt man nochmals den alten Wert des Dividenden.
  - 3. Subtraktion: Bei einem negativen Ergebnis, addiert man den Divisor wieder (Rückaddition).

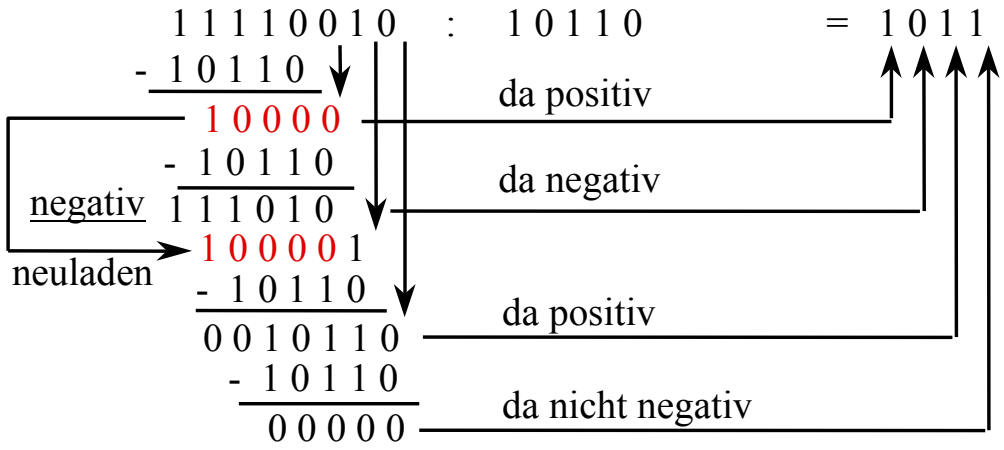
- Verkürzte Division: Rückaddition und Subtraktion des um eins nach rechts verschobenen Divisors zusammenziehen. Man addiert gleich den um eins verschobenen Divisor:

  + Divisor - ½ Divisor = + ½ Divisor

Division

- Durchführung der Subtraktion:
  - Direkte Subtraktion des Divisors
  - Addition des Divisor-Zweierkomplements.

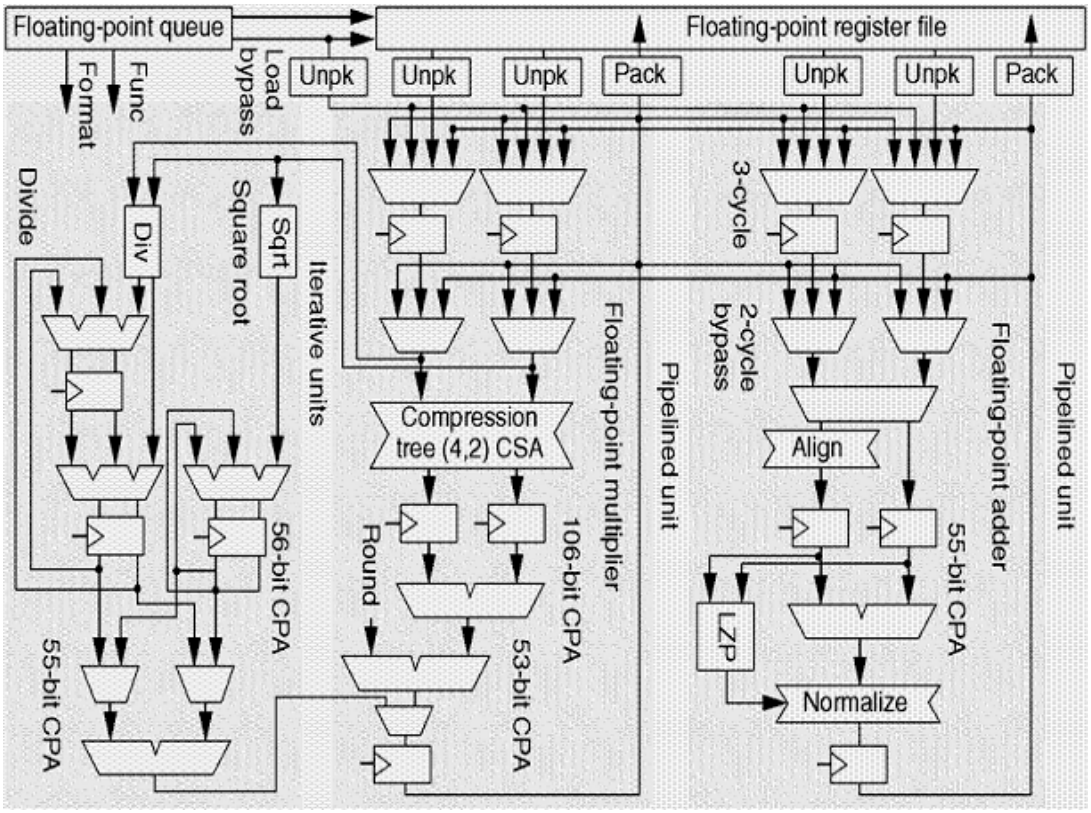
Direkte Subtraktion:



Bemerkungen

- Bei Division durch 0 muß ein Ausnahmezustand erkannt werden und an die Steuereinheit (Prozessor) weitergemeldet werden.
- Die Division muß abgebrochen werden, wenn die vorhandene Bitzahl des Ergebnisregisters ausgeschöpft ist (periodische Dualbrüche).
- Die Schaltungen für die Multiplikation können nach Modifikation auch für den Grundalgorithmus der Division eingesetzt werden:
  - Linksschieben des Dividenden (statt Rechtsschieben des Multiplikanten)
  - Subtraktion des Divisors (statt Addition des Multiplikators)

Beispiel — MIPS R10000 Floating-point Unit

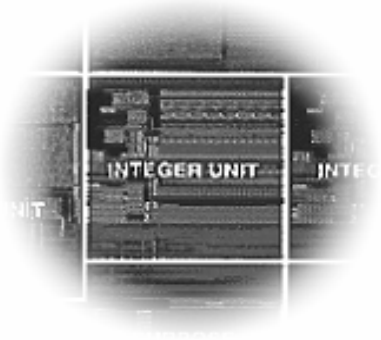


Modul 5: Rechnerarithmetik (2)

- Grundrechenarten
- Arithmetisch-logische Einheit

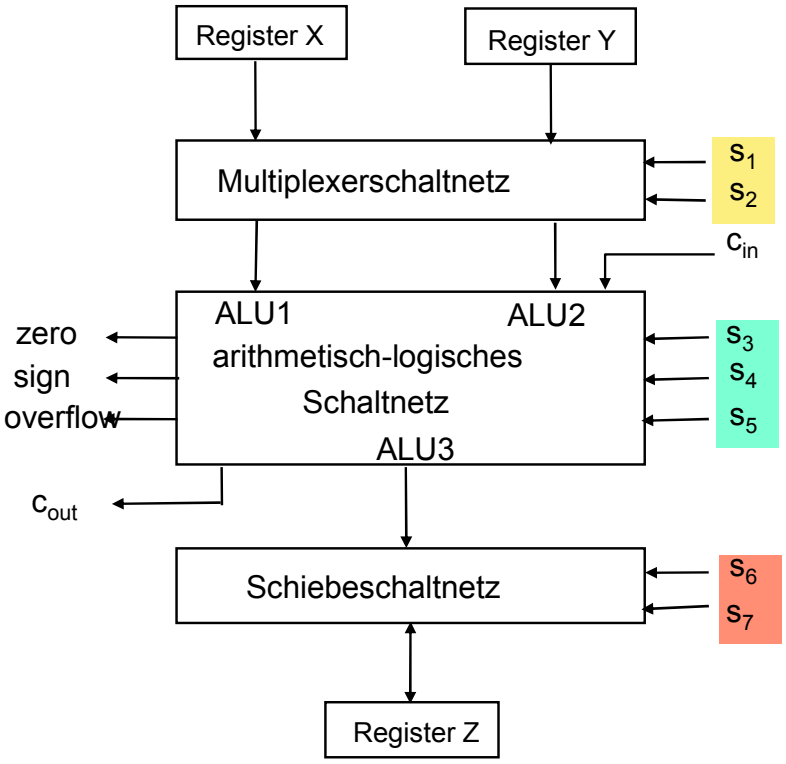
Arithmetisch-logische Einheit

- Arithmetisch-logische Einheit (ALU, arithmetic logic unit):
  - Rechenwerk, der funktionale Kern eines Digitalrechners logischer und arithmetischer Verknüpfungen.



- Eingangsdaten der ALU:
  - Daten und Steuersignalen vom Prozessor
- Ausgangsdaten der ALU:
  - Ergebnisse und Statussignale an den Prozessor.
- Oft können die in einen Prozessor integrierten ALUs nur Festkommazahlen verarbeiten. Gleitkommaoperationen werden dann entweder von einer Gleitkommaeinheit ausgeführt oder per Software in Folge von Festkommabefehlen umgewandelt.

Schema einer einfachen ALU



s_1	s_2	ALU1	ALU2
0	0	X	Y
0	1	X	0
1	0	Y	0
1	1	Y	X

s_3	s_4	s_5	ALU3
0	0	0	ALU1 + ALU2 $+c_{in}$
0	0	1	ALU1 – ALU2 - $\overline{c}_{in}$
0	1	0	ALU2 – ALU1 - $\overline{c}_{in}$
0	1	1	ALU1 $\vee$ ALU2
1	0	0	ALU1 $\wedge$ ALU2
1	0	1	$\overline{\text{ALU1}} \wedge$ ALU2
1	1	0	ALU1 $\not\leftrightarrow$ ALU2
1	1	1	ALU1 $\leftrightarrow$ ALU2

s_6	s_7	Z
0	0	ALU3
0	1	ALU3 2
1	0	ALU3 $\times$ 2
1	1	Z speichern

Bestandteile der ALU

- Registersatz
- Multiplexerschaltnetz
- Arithmetisch logisches Schaltnetz zur Durchführung arithmetisch logischer Operationen
- Schiebeschaltnetz

- Eingänge:
  - Datenworte X und Y, Steuersignale s_1… s_7 zur Festlegung der ALU-Operation

- Ausgänge:
  - Statussignale zero, sign und overflow
  - Hiermit kann das Steuerwerk bestimmte ALU-Zustände erkennen und darauf entsprechend zu reagieren.

Bitscheiben- (bitslice-) ALU

- Bitscheiben-ALU:
  - Erweiterbare Strukturen auf einem Baustein bei einer kurzen Wortlänge (m = 4 oder 8) → m-bit-ALU.

- Bitscheibe:
  - Verkettung von k der gleichen Bausteine (d.h. m-bit-ALU).
  - Erlaubt die Verarbeitung von Operanden der Wortlänge k*m.

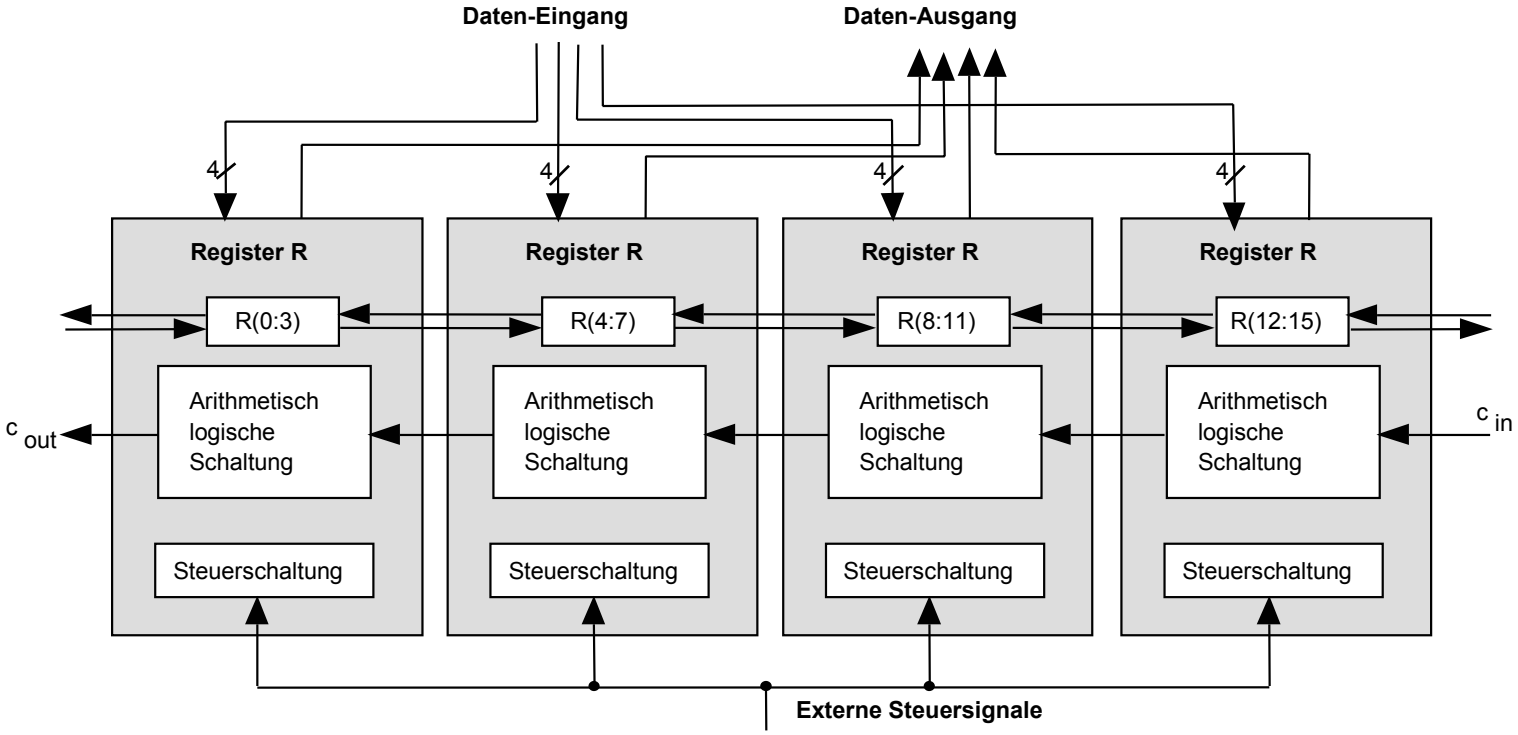
- Alle Bausteine erhalten die gleichen Steuersignale.
  - Sie führen parallel die gleiche Operation auf verschiedenen Teilen des Operanden aus.

- Ein Übertrag wird durch den Nachbarn berücksichtigt.

- Eine gemeinsame Steuerung erfolgt z.B. durch Mikroprogramm-ablauf und Mikroprogrammsteuereinheit.

Beispiel: AMD 2901-Baustein

Vier Bitscheiben der Länge 4 Bit



AMD 2901-Baustein

- Bestandteile des Bausteins:
  - Registersatz: 16 Register zu je 4 Bits (16x4-bit RAM)
  - Arithmetisch-logische Schaltung C, welche drei arithmetische und fünf logische Funktionen ausführen kann.
  - Befehlsbus I zur Auswahl einer Operation
  - Schiebeeinheit: Verschiebung nach rechts/links
- Eingänge der arithmetisch logischen Schaltung:
  - aus dem Registerspeicher, aus dem Register Q
  - aus dem externen Dateneingang D oder Null
  - Speicheradressen A und B wählen die Register für Quelle oder Ziel
- Ausgänge (Ergebnisse):
  - intern im Registersatz oder am Datenausgang Y
  - Steuerleitungen I, A und B sind parallel an alle Bitscheiben geschaltet.
  - Jede Bitscheibe verarbeitet den Übertrag c_{in} und gibt eventuell einen Übertrag c_{out} weiter, so dass ein Gruppenübertrag mit serieller Weitergabe möglich ist.
  - Merker F_0, OVR und Z zeigen den Status einer Operation an (Vorzeichen, Bereichsüberschreitung und Null).