

5. Rechnerarithmetik

- Verfahren und Implementierungsmöglichkeiten am Beispiel der vier Grundrechenarten
- Prinzipieller Aufbau einer arithmetisch-logischen Einheit, die mehrere Operationen integriert



Pentium Bug: 1994

- Fehler in Intels Pentium 80486 Prozessor im Divisionsalgorithmus für Gleitkommazahlen.
- Durch Optimierungsschritt wurde in Fällen, von denen Intel fälschlicherweise ausging, dass sie nie eintreten, eine 0 statt einer 2 zurückgegeben.
- Durch die Optimierung konnte die Hardware vereinfacht werden.
- Dieser Bug führte zu Ungenauigkeit an der 9. Stelle hinter dem Dezimalpunkt (meist wird nur eine Genauigkeit bis zur 4. Stelle benötigt)
- bei Multiplikation mit großen Zahlen wird aus kleiner Abweichung ein großer Wert! z.B. Statistiken
- Rückrufkosten für Intel: 300 Millionen Dollar.



5.4 Register Transfer Ebene

Für den Entwurf komplexerer Systeme erweist es sich als notwendig, vom detaillierten Verhalten konkreter Bausteine zu abstrahieren.

- Übergang von der Beschreibung der Schaltung auf der Gatterebene zu einer Beschreibung auf der sog. **Register-Transfer-Ebene (RT-Ebene)**

Grundelemente der RT-Ebene

- Register zur Speicherung der zu verarbeitenden Daten
- Funktionale Einheiten, wie Zähler, Addierer, Multiplexer und Demultiplexer, mit denen die Verarbeitung durchgeführt werden kann.



5.4 Register Transfer Ebene

Das Verhalten des Schaltkreises wird in der Register-Transfer-Ebene durch den Informationsfluss zwischen den Registern und Operationen (Addition, Multiplikation u. a.) repräsentiert.

Die Einheit des Informationsflusses zwischen diesen Elementen/Registern ist das **Datenwort**, das eine feste Anzahl zusammengehörender Bits umfasst.

- Die zu verarbeitenden Daten sind oft also keine einzelnen Bits, sondern Bit-Gruppen oder Bitvektoren.



Komponenten zur Realisierung spezieller Funktionen der RT-Ebene

Typ	Komponente	Funktion
Schaltnetze	Wortverknüpfung	Boolesche Operation
	Multiplexer (MUX)	Datenübertragung, allg. Funktionen
	Kodierer/Dekodierer	Codeprüfung, Codewandlung
	Programmierbare Matrizen (PLA)	Allgemeine Funktionen
	Arithmetische Elemente (Addierer, arithmetisch-Logische Einheit ALU)	Numerische Operationen
Schaltwerke	Register	Informationsspeicherung
	Schieberegister	Serien-/Parallel-Umwandlung
	Zähler	Erzeugung von Steuer- und Zeitgebersignalen

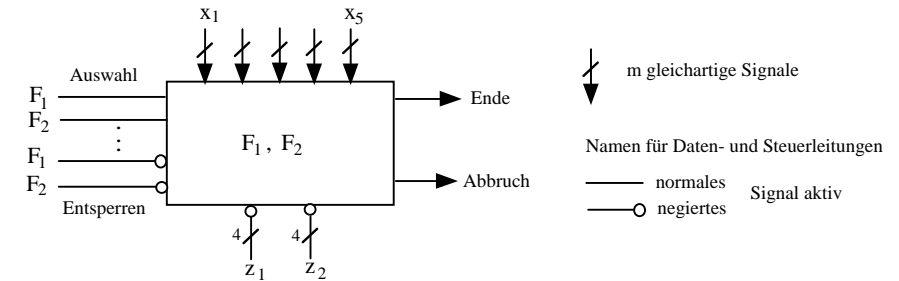
Multiplexer und PLA-Bausteine sind funktional vollständig, d.h. sie sind zusammen mit Registern für den Entwurf eines allgemeinen Schaltwerks ausreichend.



Symbole auf der RT-Ebene

Als Schaltsymbole zur Kennzeichnung der Funktion eines Bausteins werden Kästen mit Funktionsangabe oder Normsymbole verwendet.

Beispiel:



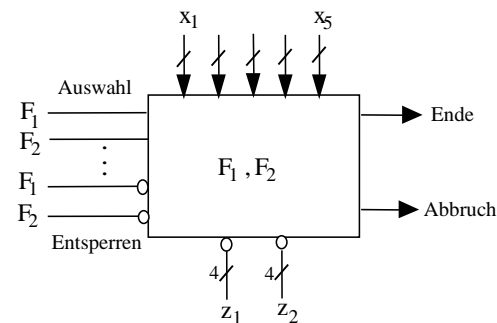
Symbole auf der RT-Ebene

Auswahlsignale:

Legen fest, welche Funktion realisiert werden soll.

Entsperrersignale:

Definieren den Zeitpunkt für den Beginn einer Operation oder erlauben deren Ablauf.



5.4.1 Schaltalgebra für Binärwörter

Man kann die Schaltalgebra für Binärzeichen auf Binärwörter erweitern (Übergang von skalaren auf vektorielle Größen).

$$F: (B^m)^n \rightarrow B^m \quad n \text{ Wörter } X_i \text{ mit } m \text{ Stellen}$$

$$F: (X_0, X_1, \dots, X_{n-1}) = [f(x_{00}, x_{10}, \dots, x_{(n-1)0}),$$

$$f(x_{01}, x_{11}, \dots, x_{(n-1)1}), \dots,$$

$$f(x_{0(m-1)}, x_{1(m-1)}, \dots, x_{(n-1)(m-1)})]$$



Schaltalgebra für Binärwörter

Analog zur Schaltalgebra für Binärzeichen ordnet eine **n**-stellige Funktion von **m**-Bit-Binärwörtern jeder möglichen Belegung der **m**-stelligen Variablen ein **m**-stelliges Ergebnis zu.

Die $2^{m \cdot n}$ möglichen Funktionen für **n** Wörter zu **m** Bits bilden ebenfalls eine Schaltalgebra.

Beispiel: (Skalare Vektoroperation)

Konjunktion: $\mathbf{y} \wedge \mathbf{X} = (\mathbf{y} \ x_0, \mathbf{y} \ x_1, \dots, \mathbf{y} \ x_{m-1})$

Disjunktion: $\mathbf{y} \vee \mathbf{X} = (\mathbf{y} \vee x_0, \mathbf{y} \vee x_1, \dots, \mathbf{y} \vee x_{m-1})$



Schaltalgebra für Binärwörter

Im Gegensatz zur Gatterebene gibt es auf der RT-Ebene keine strengen Entwurfsregeln.

Vergleich zur Assembler-Ebene beim Software-Entwurf:

„Kunst statt Wissenschaft“



Schaltalgebra für Binärwörter

Der Ansatz ergibt aus folgenden Gründen **keine Entwurfstheorie**:

- Die Binärwörter können unterschiedlich Daten und Codes (Dualzahlen, ASCII-Zeichen, ... usw.) repräsentieren.
- Numerische Operationen sind z.B. nicht leicht in die Schaltalgebra eingliederbar.
- Für viele komplexe logische Funktionen gelten die Gesetze der Schaltalgebra (Assoziativität, Kommutativität, Distributivität) nicht mehr.
- Es ist häufig notwendig, in einer Schaltung unterschiedliche Wortlängen zu verwenden oder nur Teile eines Wortes zu benutzen. Trotz fester Wortlänge **m** treten oft Abweichungen in der Bitzahl auf.



Beispiel

Das Ergebnis einer Abfrage

$$(\mathbf{x}_0, \dots, \mathbf{x}_{m-1}) = (\mathbf{y}_0, \dots, \mathbf{y}_{m-1})?$$

hat z.B. eine Wortlänge von **1** Bit, deshalb ist eine einheitliche Wortlänge nicht möglich.



Beschreibung von RT-Operationen

Eine Beschreibung von Register-Transfer-Operationen enthält die **Grobstruktur des Datenflusses** zwischen den speichernden und den verarbeitenden Einheiten:

- Blockdiagramm, in dem Leitungsbündel Register und Funktionsbausteine miteinander verbinden oder
- Beschreibung mit einer Hardware-Beschreibungssprache wie z.B. VHDL, VerilogHDL (HDL=Hardware Description Language).



Register-Transfer-Anweisungen

Als Basis zur Beschreibung von RT-Operationen dienen sog. **Register-Transfer-Anweisungen** der Form:

$$Z \leftarrow F(X_1, X_2, \dots, X_n)$$

Z, X_1, X_2, \dots, X_n stellen Register dar.

F ist eine numerische oder Boolesche Funktion.

Die Anweisung kann wie folgt interpretiert werden:

Die Verknüpfung der Eingangsgrößen X_1, X_2, \dots, X_n durch F liefert den Registerinhalt für Z in den nächsten Taktschritten.



Register-Transfer-Anweisungen

Typische **Register-Transfer-Anweisungen** bestehen aus folgenden Schritten:

- Auswahl einer Menge von Datenworten, die in den Registern X_1, X_2, \dots, X_n gespeichert sind.
- Verarbeitung dieser Worte mit Schaltnetzen F_1, F_2, \dots, F_m
- Speicherung der Ergebnisse der Verarbeitung in einer Menge von Registern X'_1, X'_2, \dots, X'_m



Register Transfer Anweisungen

In einer Hardware-Beschreibungssprache wird eine Menge von RT-Operationen beschrieben durch:

$$X'_1 = F_1(X_1, X_2, \dots, X_n),$$

$$X'_2 = F_2(X_1, X_2, \dots, X_n),$$

...

$$X'_m = F_m(X_1, X_2, \dots, X_n);$$

Ein Komma trennt die während eines Taktes parallel ausgeführten Operationen; Ein Semikolon markiert das Ende des Taktzyklus.



Register-Transfer-Anweisungen

Eine Beeinflussung der Ausführung der Anweisung erfolgt durch eine Steuergröße c :

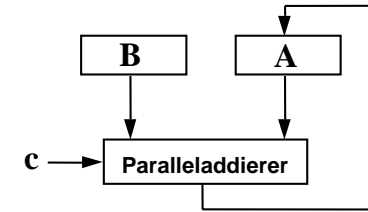
c : $Z \leftarrow F(X_1, X_2, \dots, X_n)$ oder
if ($c=1$) **then** $Z \leftarrow F(X_1, X_2, \dots, X_n)$

Die Anweisungen einer Register-Transfer-Sprache können durch Hardware-Komponenten interpretiert werden.

Register Transfer Anweisungen

Beispiel:

c : $A \leftarrow A + B$ lässt sich als Blockdiagramm darstellen.



Die Summe $A+B$ wird nach A übernommen, wenn die Bedingung $c = 1$ erfüllt ist.

Die Steuervariable c ist dabei offen und muss spezifiziert werden.

5.4.2 Logische Bausteine der RT-Ebene

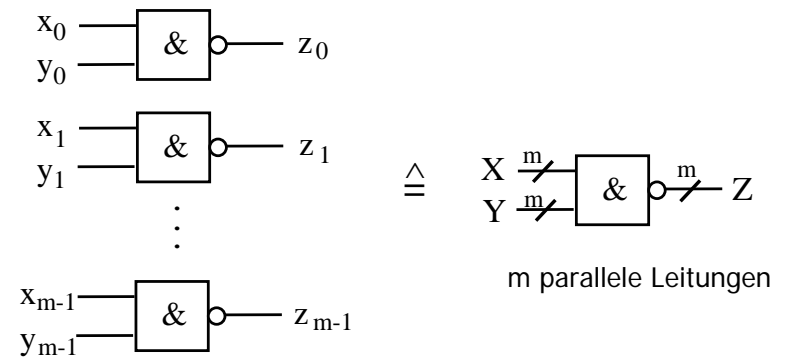
Um Register-Transfer-Operationen implementieren zu können, werden Bausteine verwendet, die Variablenbündel statt einzelner Variablen verarbeiten und somit mehrstellige Operationen ausführen.

Beispiel 1: Mehrstellige, parallele, logische Operationen

$Z = \overline{X \wedge Y}$ sei gleich bedeutend mit

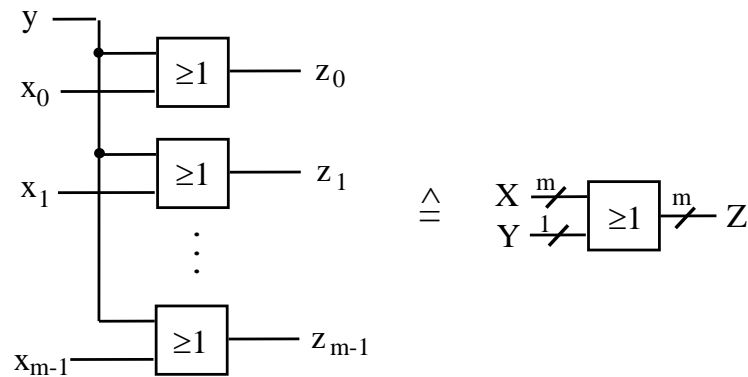
$$(z_0, z_1, \dots, z_{m-1}) = (\overline{y_0 x_0}, \overline{y_1 x_1}, \dots, \overline{y_{m-1} x_{m-1}})$$

Beispiel 1



Beispiel 2

Skalare Operationen sind in gleicher Weise darstellbar.



Weitere Bausteine

- ❑ Multiplexer,
- ❑ Demultiplexer/Dekodierer,
- ❑ Speicherbausteine (ROM, PROM, EPROM, RAM),
- ❑ programmierbare logische Matrizen (PLA, FPLA) und
- ❑ Register, Zähler und Arithmetikelemente



Zusammenfassung

Um eine Folge von RT-Anweisungen zu implementieren sind 4 Typen von Bauelementen notwendig:

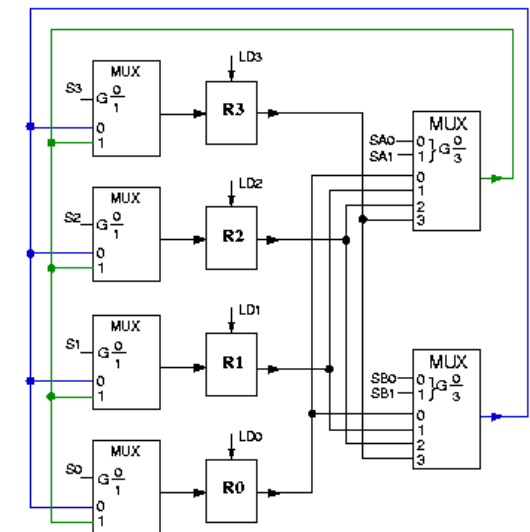
- Register zur Speicherung der zu verarbeitenden Daten
- Verarbeitungsschaltnetze, wie Gatter, Addierer, ALU's, PLA's zur Realisierung Boolescher Funktionen
- Multiplexer/Demultiplexer, um die richtigen Verarbeitungswege zu schalten
- Bauelemente, welche die Steuersignale für die MUX/DX, ALU's ... usw. erzeugen



Beispiel

Register-Transfer Operationen:

- K1: $R1 \leftarrow R2$
- K2: $R3 \leftarrow R2$
- K3: $R3 \leftarrow R0$
- K4: $R2 \leftarrow R1$

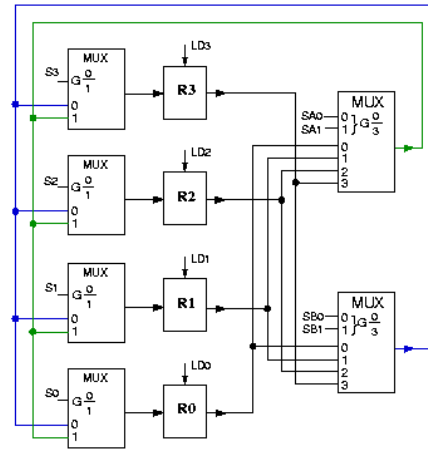


Beispiel

Die Anweisungen sollen in **möglichst wenigen Taktzyklen** ausgeführt werden

- K1: R1 ← R2
- K2: R3 ← R2
- K3: R3 ← R0
- K4: R2 ← R1

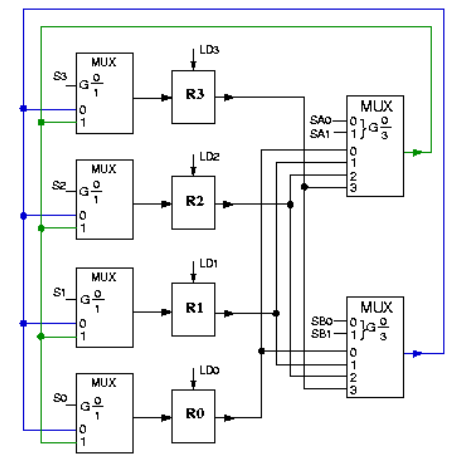
- 1. Takt:** R1 ← R2,
R3 ← R2,
R2 ← R1;
- 2. Takt:** R3 ← R0;



Beispi

Bitkombinationen zur Steuerung d

- 1. Takt:** R1 ← R2,
R3 ← R2,
R2 ← R1;
- 2. Takt:** R3 ← R0;



	S ₀	S ₁	S ₂	S ₃	LD ₀	LD ₁	LD ₂	LD ₃	SA ₀	SA ₁	SB ₀	SB ₁
1. Takt	-	1	0	1	0	1	1	1	0	1	1	0
2. Takt	-	-	-	0	0	0	0	1	-	-	0	0

Mikroprogramm!

5.5 Arithmetisch logische Einheit

Arithmetisch-logische Einheit (ALU, arithmetic logic unit):

Rechenwerk, der funktionale Kern eines Digitalrechners zur Durchführung logischer und arithmetischer Verknüpfungen.

Eingangsdaten der ALU:

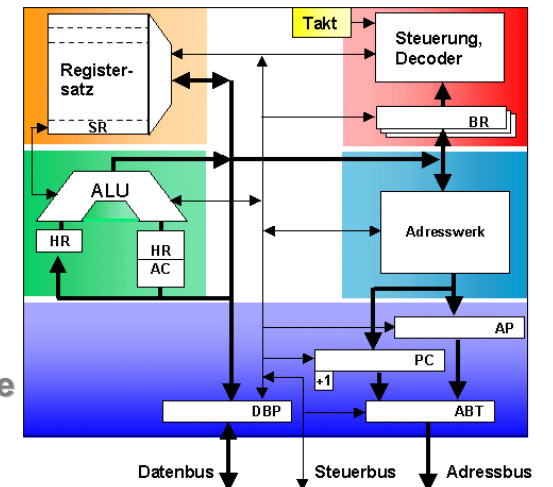
Daten und Steuersignale vom Prozessor

Ausgangsdaten der ALU:

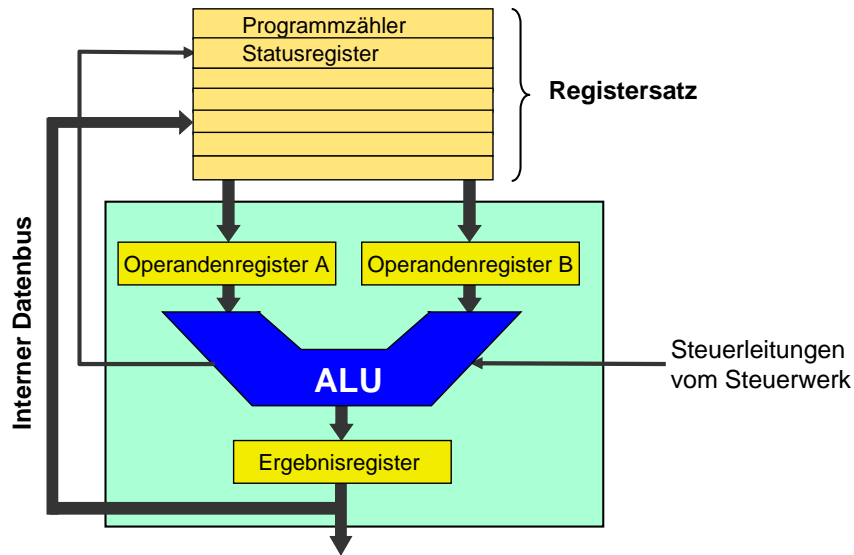
Ergebnisse und Statussignale an den Prozessor.

Interner Aufbau eines einfachen µP

- Steuerwerk
- **Rechenwerk**
- Adresswerk
- Registersatz
- Interne Busse
- Systembusschnittstelle



Rechenwerk

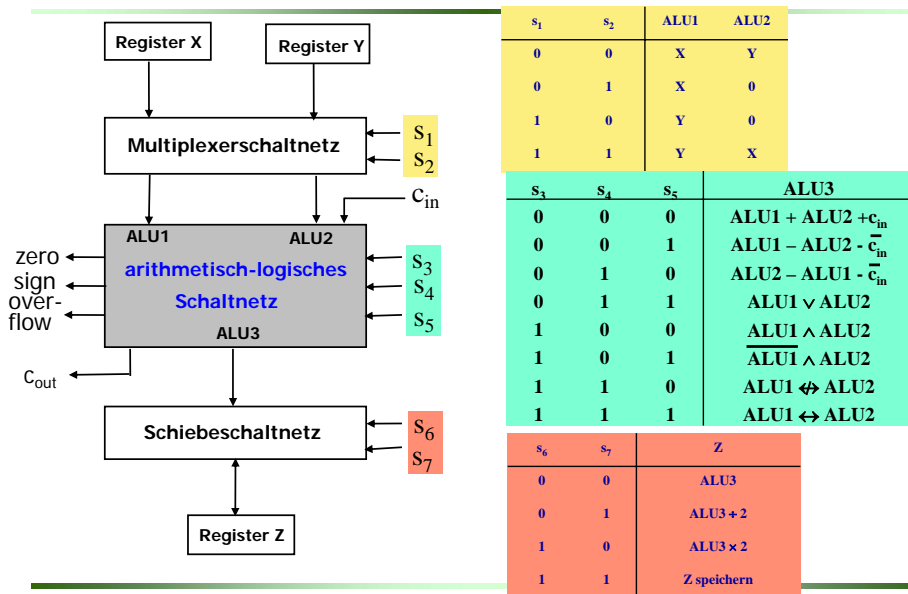


Anmerkungen

Oft können die in einen Prozessor integrierten ALU's nur Festkommazahlen verarbeiten.

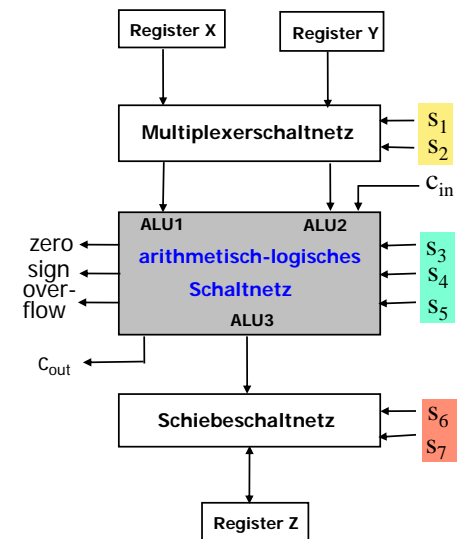
Die Gleitkommaoperationen werden dann entweder von einer Gleitkommaeinheit ausgeführt oder per Software in eine Folge von Festkommabefehlen umgewandelt.

Schema einer einfachen ALU



Bestandteile der ALU

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



Ein und Ausgänge

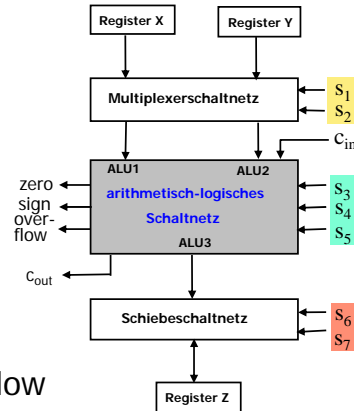
Eingänge:

- Datenworte **X** und **Y**
- Steuersignale $s_1 \dots s_7$ zur Festlegung der ALU-Operation

Ausgänge:

- Statussignale zero, sign und overflow

Hiermit kann das Steuerwerk bestimmte ALU-Zustände erkennen und darauf entsprechend zu reagieren.



Beispiele

- Einerkomplement von **Y** um ein Bit nach links verschoben in **Z** ablegen.

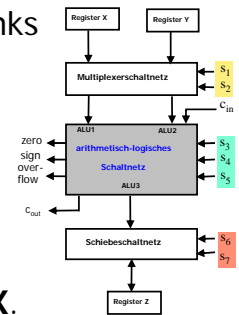
Steuersignale: $s_1 \dots s_7 = 10\ 111\ 10$

- Ist $X > Y$?

Statussignal "**sign**" bei der Operation $Y - X$.

Steuersignale:

$s_1 \dots s_7 = 00\ 010\ 00$ und $c_{in} = 1$



Bitscheiben 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**

Bitscheiben ALU

Alle Bausteine erhalten die gleichen Steuersignale.

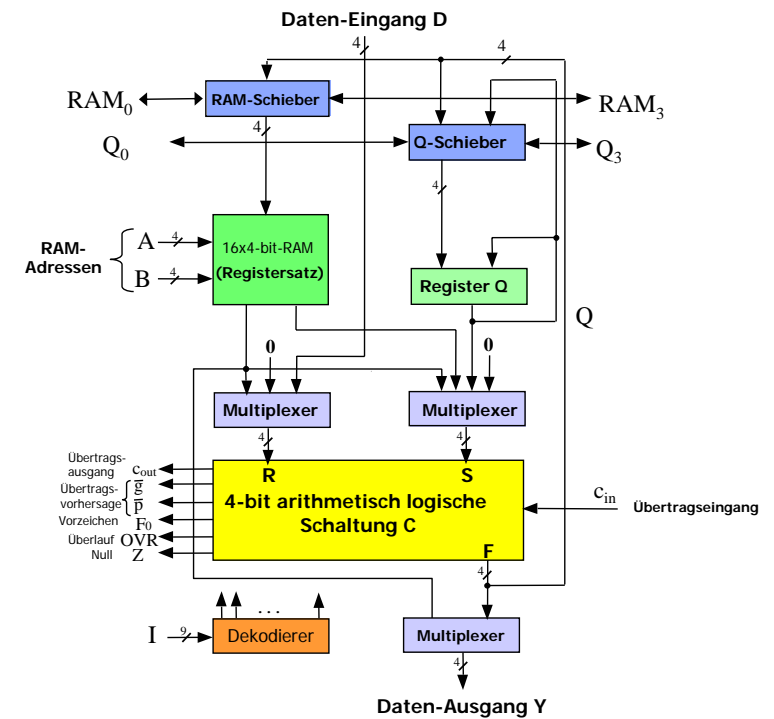
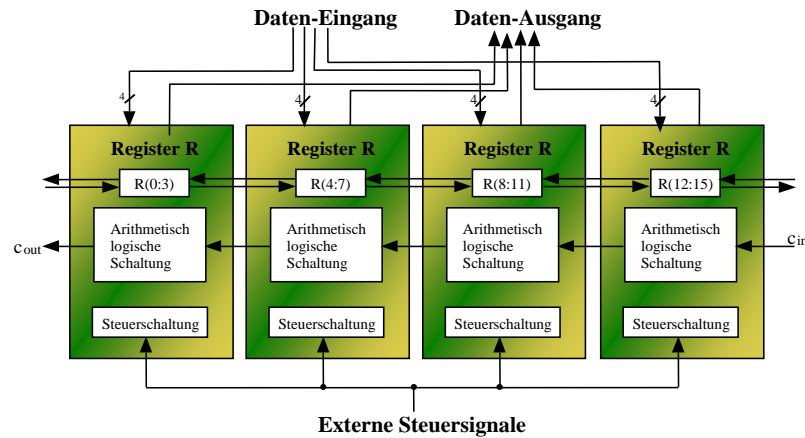
Sie führen parallel die gleiche Operation auf verschiedenen Teilen des Operanden aus.

Ein Übertrag wird durch den Nachbarn berücksichtigt.

Eine gemeinsame Steuerung erfolgt z. B. durch Mikroprogrammablauf und Mikroprogrammsteuereinheit.

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 / zur Auswahl einer Operation
- ❑ Die Schiebereinheit: Verschiebung nach rechts/links

AMD 2901 Baustein

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, welche als Quelle oder Ziel einer Operation dienen sollen.

AMD 2901 Baustein

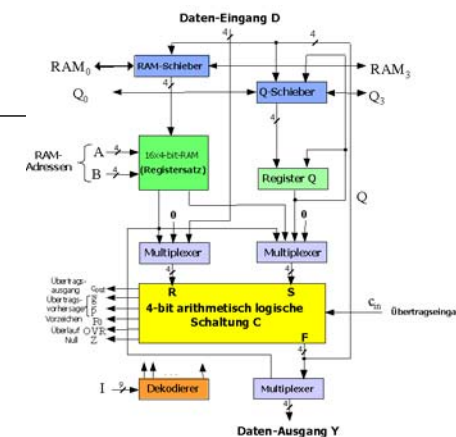
Ausgänge (Ergebnisse)

- intern im Registersatz
- 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 **F₀**, **OVR** und **Z** zeigen den Status einer Operation an (Vorzeichen, Bereichsüberschreitung und Null).

Mikrooperationen des 2901-Bausteins

Jedes Befehl I enthält 3 Felder I_s , I_f , I_d für Quellen-, Operations-, und Zielangaben und jedes Feld besteht aus 3 Bits.

$I_s = I_0I_1I_2$	Quellenangaben	
	R	S
000	RAM(A)	Q
001	RAM(A)	RAM(B)
010	0	Q
011	0	RAM(B)
100	0	RAM(A)
101	-	RAM(A)
110	-	-
111	-	0

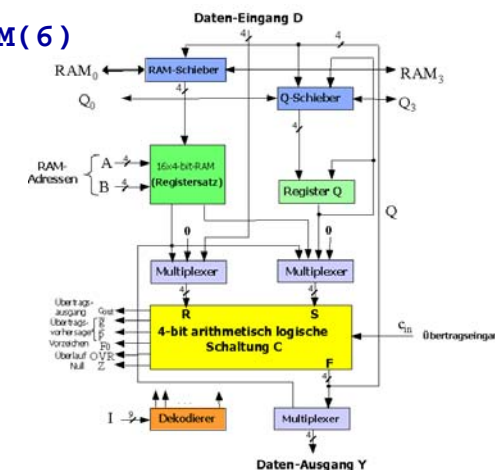


Mikrooperationen des 2901-Bausteins

$I_f = I_3I_4I_5$	Operation	$I_d = I_6I_7I_8$	Zielangaben		
			Y	RAM(B)	Q
000	$R + S - \bar{c}_{in}$	000	F	-	F
001	$S - R - \bar{c}_{in}$	001	F	-	-
010	$R - S - c_{in}$	010	RAM(A)	F	-
011	$R \vee S$	011	F	F	-
100	$\bar{R} \wedge S$	100	F	$F \div 2$	$Q \div 2$
101	$R \wedge S$	101	F	$F \div 2$	-
110	$R \leftrightarrow S$	110	F	$2 \times F$	$2 \times Q$
111	$R \leftrightarrow S$	111	F	$2 \times F$	-

Beispiel

$$RAM(6) \leftarrow RAM(7) - RAM(6)$$



$$A, B, I_s, I_f, I_d, c_{in} = 0111, 0110, 001, 010, 011, 0$$