Digitale Signalverarbeitung

WS03/04 Echtzeitsysteme Kap. 4

G. Schrott
nach dem Seminarvortrag von Robert Dörfel und Daniel Mentz

Was sind DSPs ?

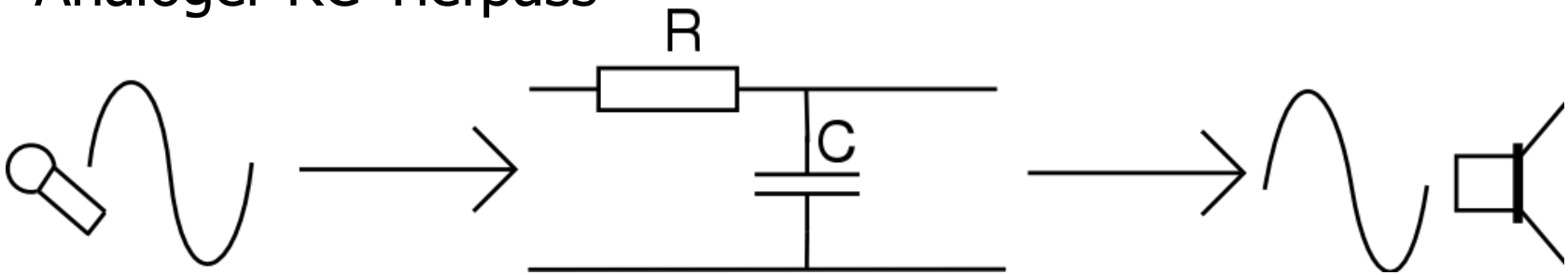
Digitale SignalProzessoren sind spezielle Mikroprozessoren, die auf die Umsetzung von Algorithmen aus der Digitalen Signalverarbeitung optimiert sind:

- Digitale Filter
- Kompression, Dekompression
- Modulation, Demodulation

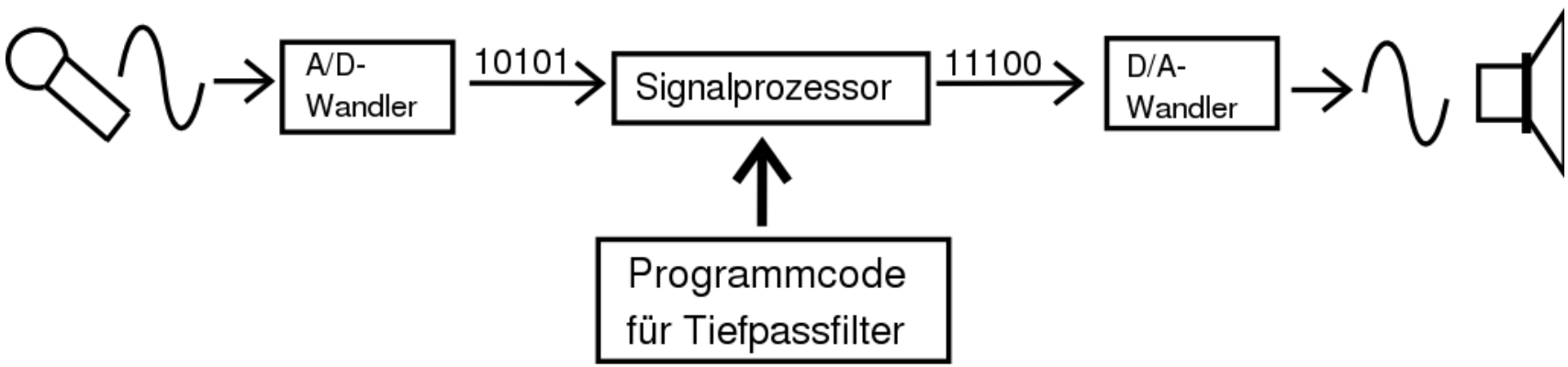
Vergleich Analog - Digital

Beispiel Tiefpassfilter

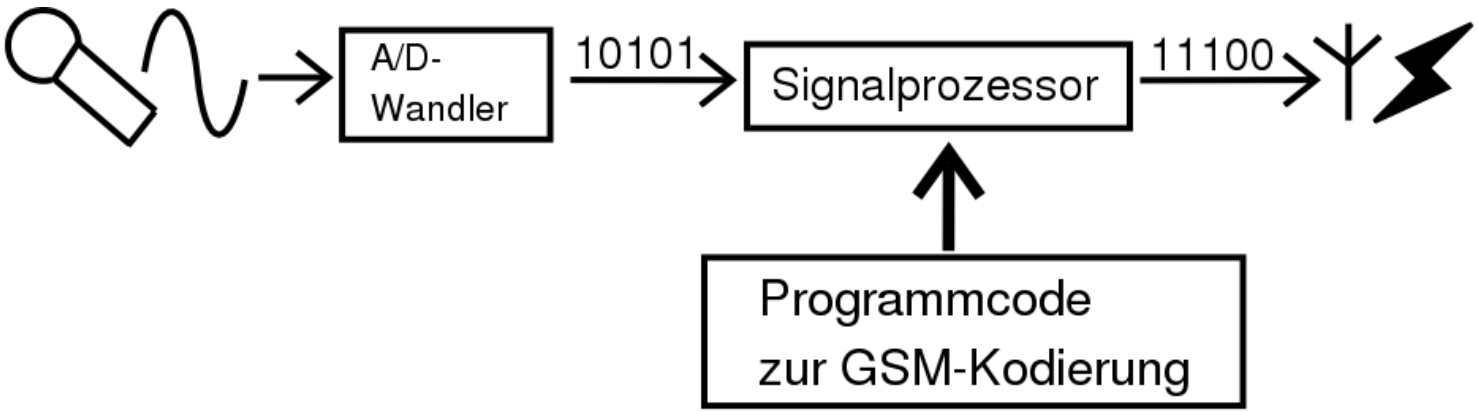
Analoger RC-Tiefpass



Digitale Variante mit Hilfe eines DSPs

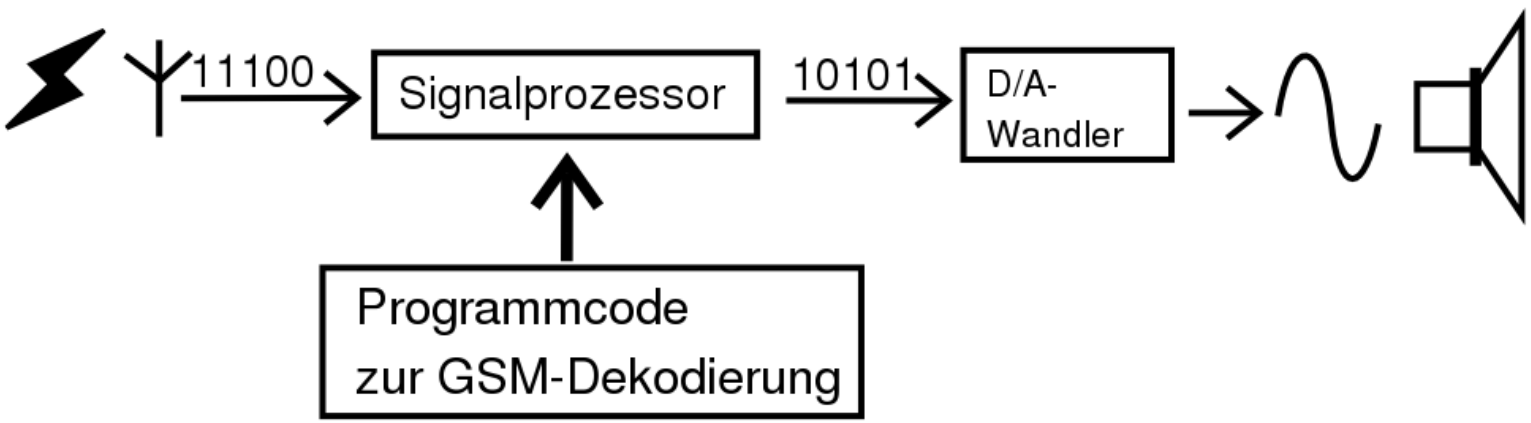


Anwendungsbeispiel



Senderseite

Empfängerseite



Vorteile von DSPs

- Programmierbarkeit, flexibler Einsatz
- Eine Hardware für viele Anwendungen
- Reproduzierbarkeit
- Stabilität, Keine Alterung, keine Temperaturabhängigkeit
- Höhere Genauigkeit
- Adaptive Algorithmen
- Neue Möglichkeiten der Signalverarbeitung, die mit analogen Bausteinen nicht oder nur schwer realisierbar sind
  - Spezielle Filter
  - Audiokompression, Videokompression
  - Echo, Nachhall
  - Digitale Modulationsverfahren (z.B GSM)
  - Fehlerkorrigierende Codes (ECC)

Anwendungsbereiche von DSPs

- Mobilfunk (GSM, PCS) und allgemein in der Telekommunikation
  Marktanteil beträgt hier 50 % (Stückzahltreiber)
- Modems, DSL-Modems
- Konsumerbereich, Unterhaltungselektronik
  (Digitales Fernsehen, DVD, ...)
- Meßtechnik, intelligente Sensoren
- Komplexe Digitale Regler
- Audio/Video-Kompression
- Mustererkennung, Bildverarbeitung
- Sprachsynthesizer
- großer Markt (im Jahr 2000 831 Millionen Chips verkauft)
- Marktführer: Texas Instruments

Architektur von DSPs

- Welche Anforderungen werden an Signalprozessoren gestellt.
- Wie unterscheiden sie sich von herkömmlichen Mikroprozessoren bzw. Mikrocontrollern ?
- Welche architektonischen Besonderheiten besitzen sie ?

Anforderungen an DSPs

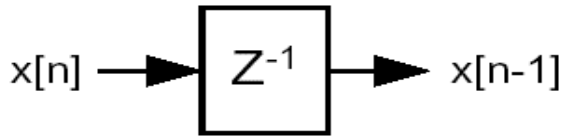
- Berechnungsintensive Algorithmen müssen in Echtzeit d.h. innerhalb eines Abtastintervalls ausgeführt werden.
- Anders als bei Prozessoren für den PC-Markt müssen vor allem arithmetische Operationen ausgeführt werden.

Um diese Anforderungen zu erfüllen, bedient man sich im wesentlichen den zwei folgenden Konzepten:

- Parallelisierung soweit wie möglich
- Funktionseinheiten, die Operationen wie z.B. die Multiplikation in einem Taktzyklus ausführen können.
- Schneller interner Speicher

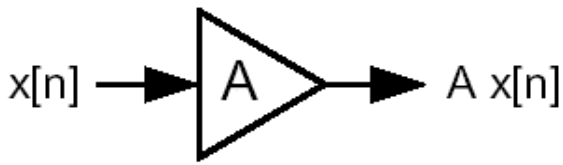
Bestandteile Digitaler Filter

- Verzögerungsglied:



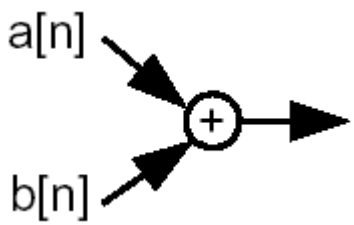
Zeitverzögerung um ein Abtastintervall
Ausgabe des Eingangssignal einen Takt später

- Multiplizierer:



Multiplikation von Signal x[n]
mit konstantem Faktor A

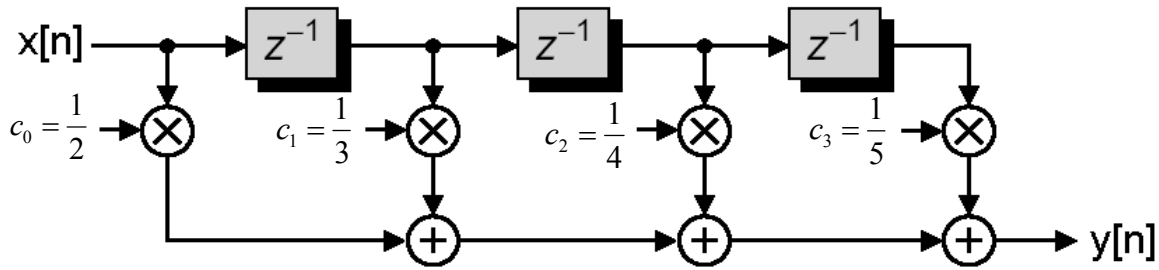
- Summierer/Addierer:



Summation von Signal a[n]
und Signal b[n]

Finite-Impulse-Response-Filter

- Allgemeiner Aufbau:



Die Berechnungsvorschrift hat die Form

$$y[n] = c_0 x[n] + c_1 x[n-1] + c_2 x[n-2] + c_3 x[n-3]$$

Eine Summe von Produkten (product of sums) ist häufig in der Digitalen Signalverarbeitung anzutreffen. Deshalb bieten die meisten DSPs eine MAC-Operation (Multiply and Accumulate) an.

$$ACC \Leftarrow ACC + c_i x[i]$$

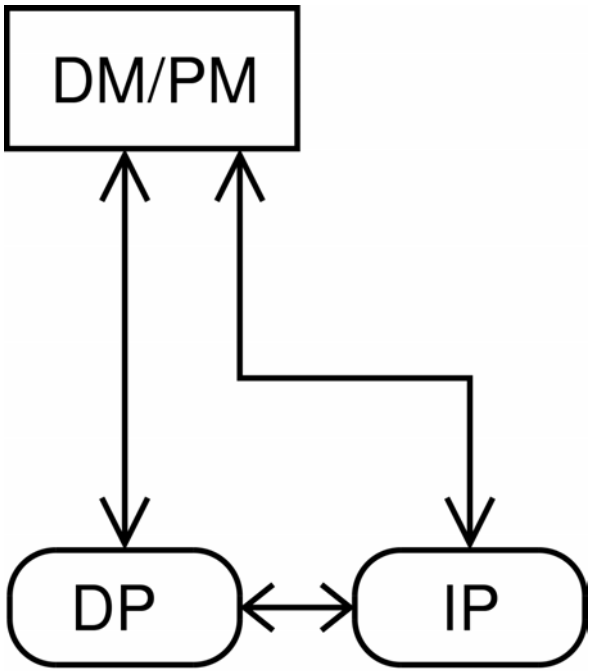
Weitere Operationen

Mit den zwei Operationen Addieren und Multiplizieren ist es noch nicht getan. Hinzu kommt eine Menge an Verwaltungsaufwand:

- Befehl laden
- Befehl dekodieren
- Speicheradresse berechnen
- Operanden laden (meistens zwei pro Operation)
- Ergebnis speichern
- Abfragen von Schleifenbedingungen und ggf. Verzweigen

Der Grundgedanke zur Optimierung von DSP-Architekturen ist das Parallelisieren möglichst vieler Operationen.

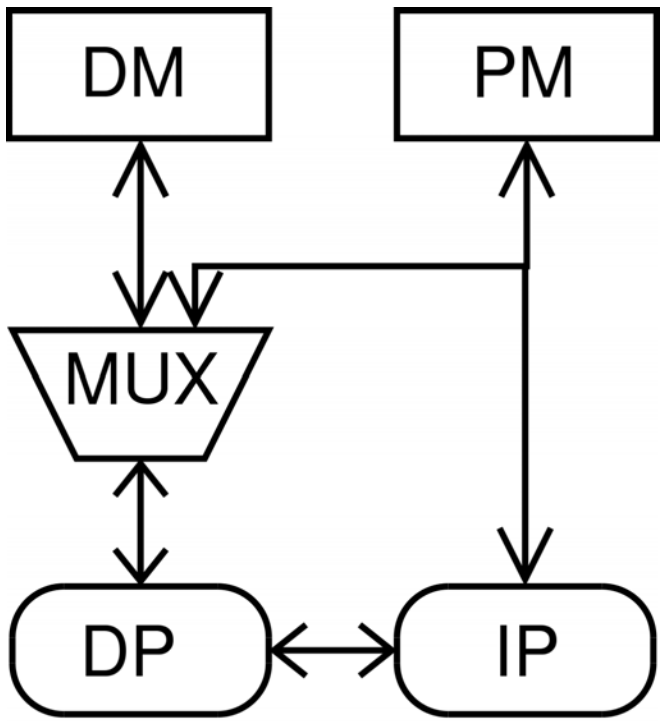
Von-Neumann Architektur



DM: Data Memory
PM: Program Memory
DP: Data Processor
IP: Instruction Processor

Ein wesentliches Merkmal der Von-Neumann Architektur, ist der gemeinsame Speicher für Daten und Programmcode.
Beim gleichzeitigen Zugriff auf den Hauptspeicher durch DP (Operanden laden) und IP (nächsten Befehl laden) kommt es zum Konflikt.

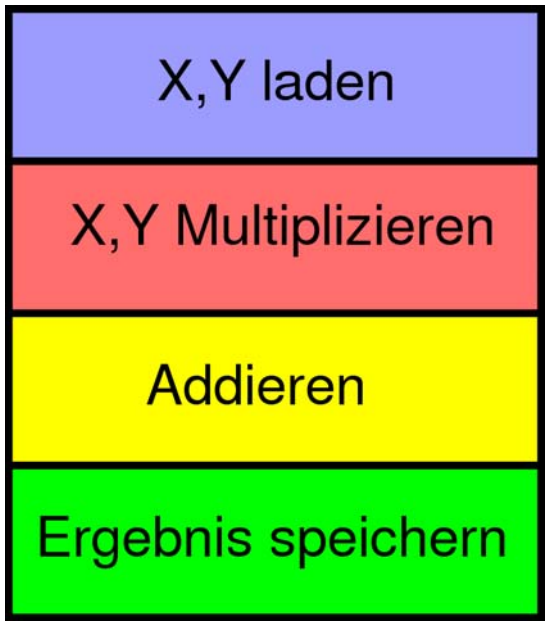
Harvard Architektur



DM: Data Memory
PM: Program Memory
DP: Data Processor
IP: Instruction Processor
MUX: Multiplexer

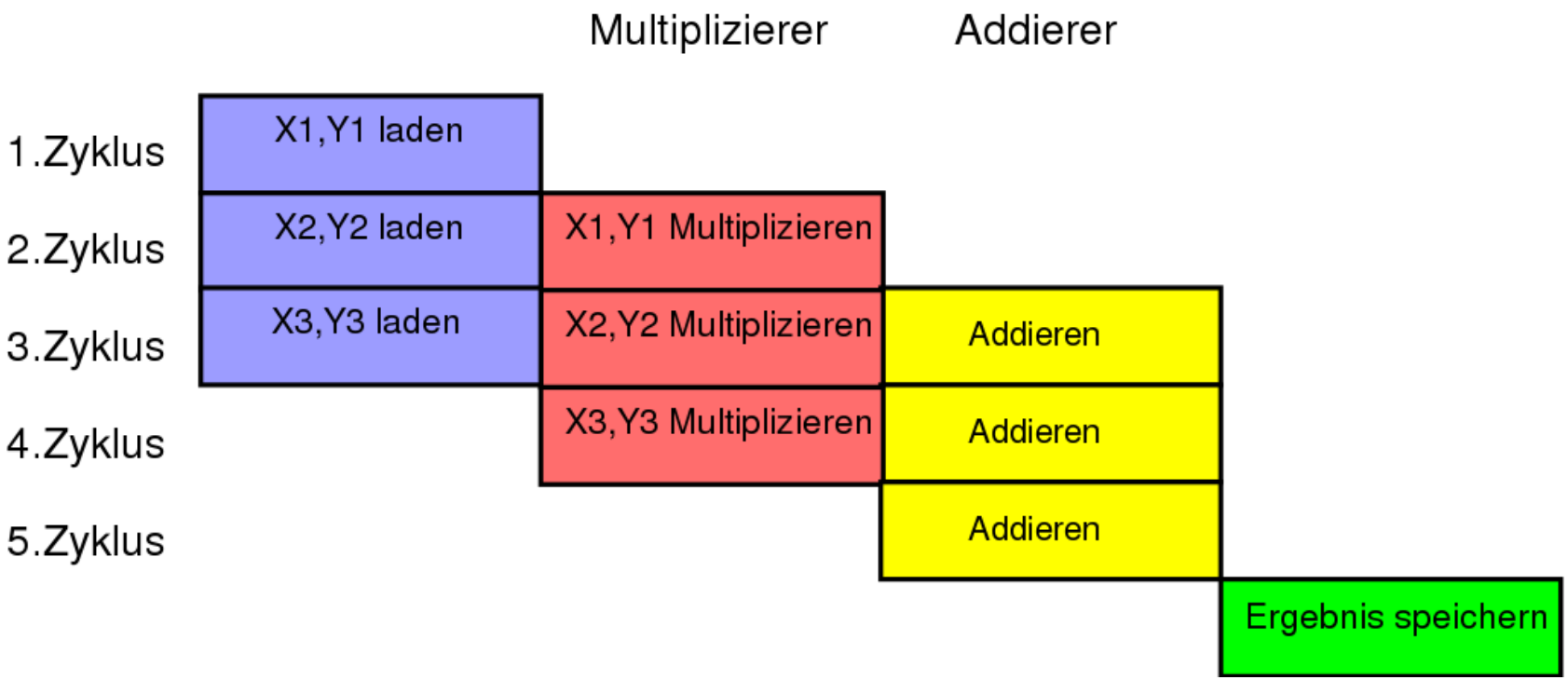
Mit der Harvard Architektur ist es möglich einen Operanden sowie den nächsten Befehl parallel aus dem Speicher zu lesen.

Pipelining auf dem Datenpfad



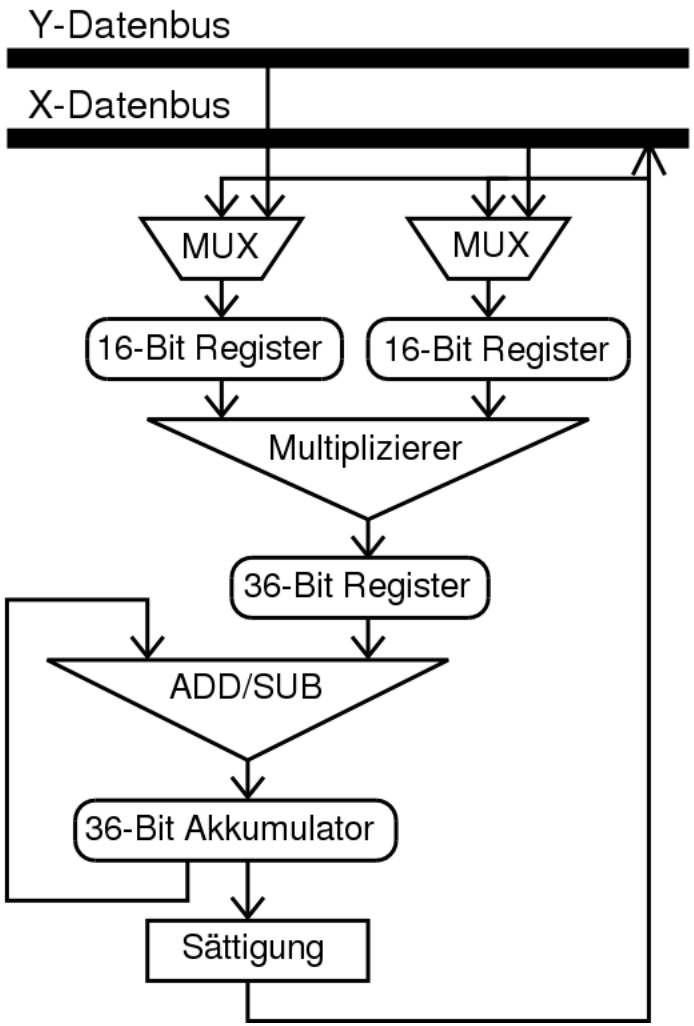
Das Ergebnis einer einzelnen MAC-Operation liegt nach vier Taktzyklen vor.

Pipelining auf dem Datenpfad



Durch Pipelining konvergiert die Ausführungszeit bei aufeinanderfolgenden MAC-Operationen auf einen Taktzyklus pro MAC-Operation.

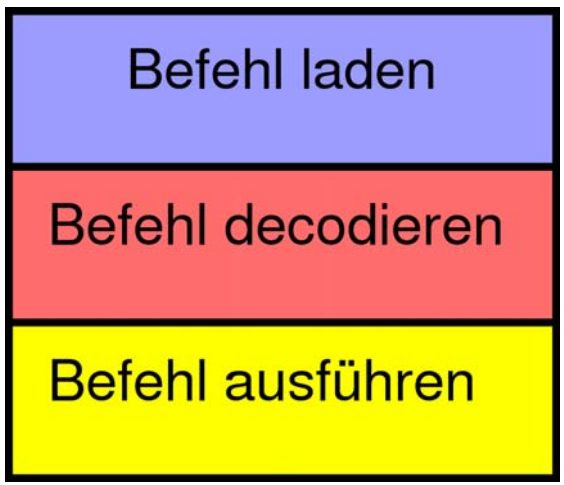
Kaskadierte Arithmetikeinheit



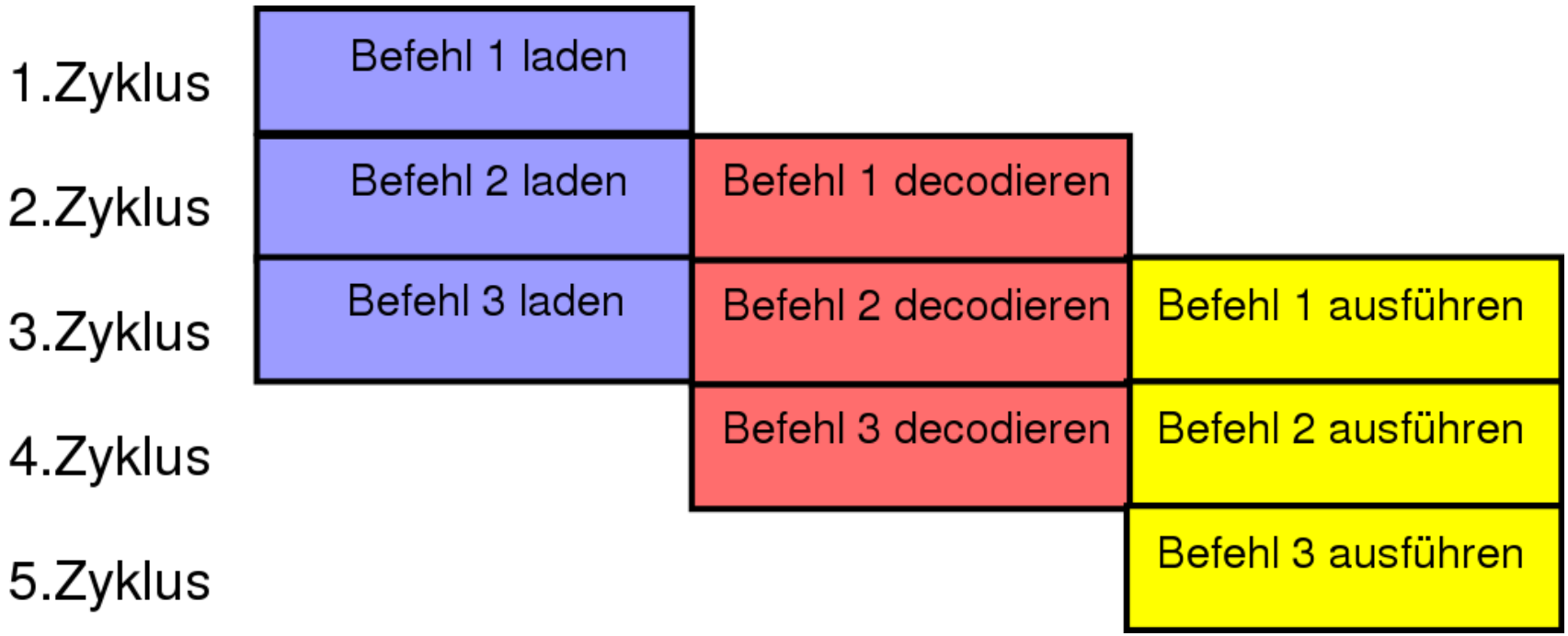
Bei einer kaskadierten Arithmetikeinheit konvergiert die Ausführungszeit mehrerer aufeinanderfolgender MAC-Operationen gegen einen Taktzyklus

Pipelining auf Befehlsebene

Die drei Stufen der Befehlsausführung



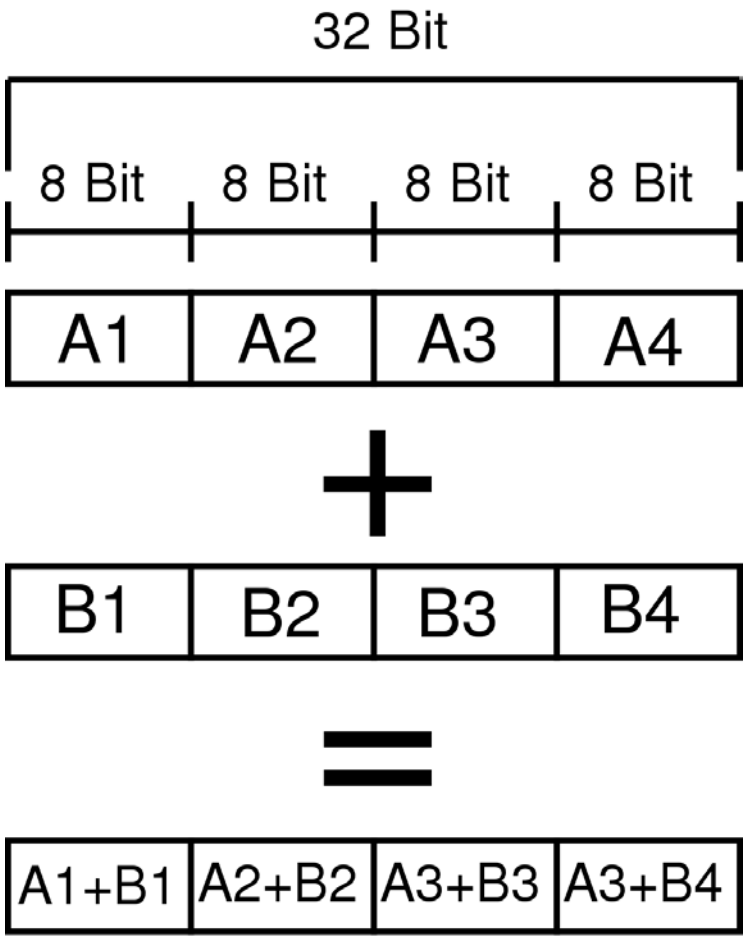
Pipelining auf Befehlsebene



Im Regelfall wird der Befehl, der sich an der nächsten Speicheradresse befindet geladen. Schwierigkeiten treten bei Verzweigungen bzw. Schleifen auf.

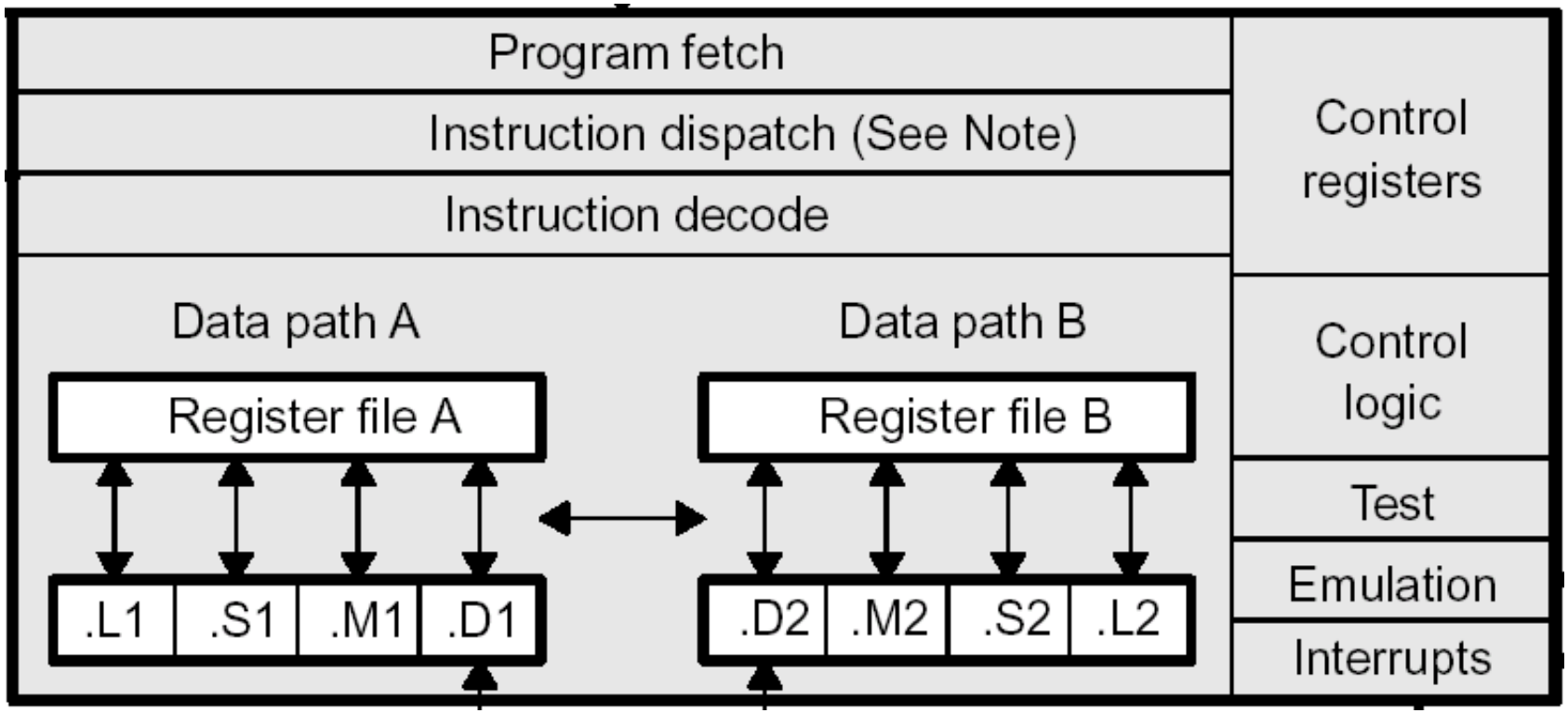
- Zero overhead looping
- Sprungvorhersage

Single Instruction Multiple Data



- Mit einem einzigen Befehl werden mehrere Paare von Operanden parallel verknüpft
- Ein Register wird in mehrere Teile aufgeteilt.
- Jeder Registerteil stellt einen Operanden dar.

Very Long Instruction Word



Der TI TMS320C62x verfügt über 8 „functional untis“ (2 Multiplizierer, 6 ALUs)

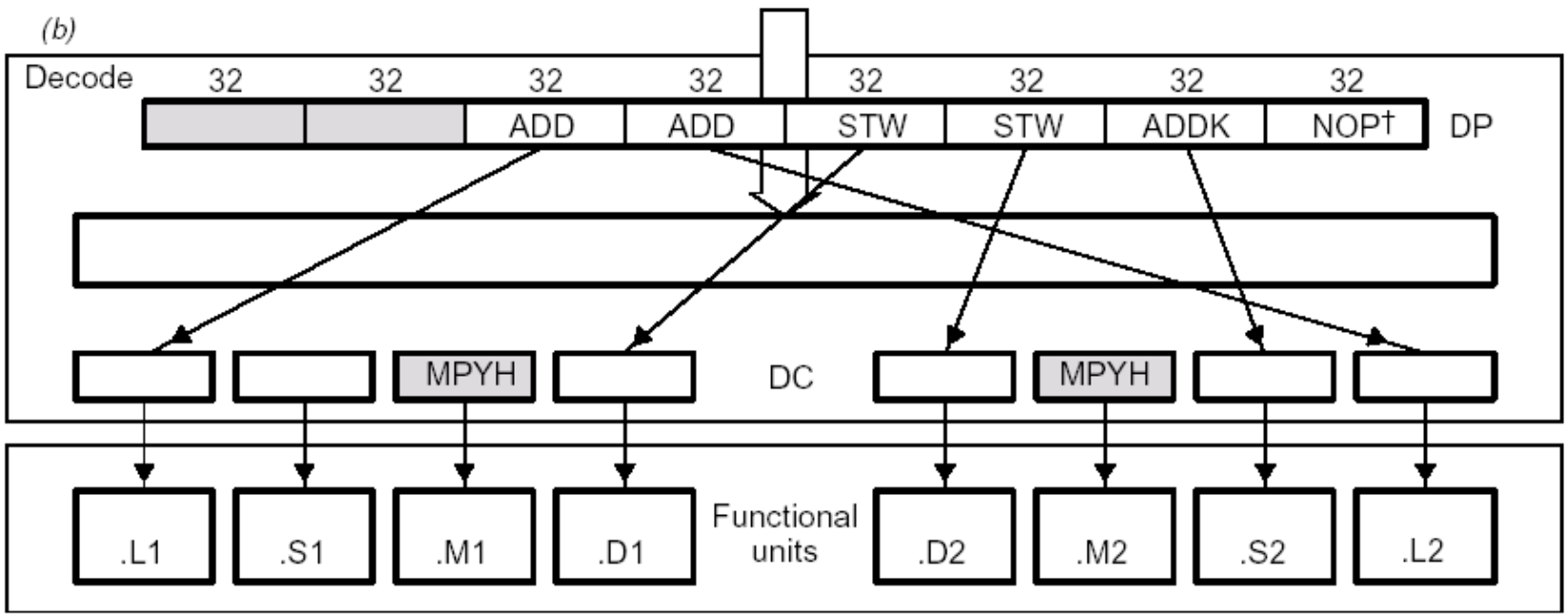
Very Long Instruction Word

```
        B       .S2     LOOP    ; branch to loop
||      SUB     .S1     A1,1,A1 ;* decrement loop counter
||      ZERO    .L1     A8      ; zero out sum0 accumulator
||      ZERO    .L2     B8      ; zero out sum0 accumulator
```

Alle Befehle denen „||“ vorangestellt ist, werden parallel mit dem aus der vorhergehenden Zeile ausgeführt.
In der dritten Spalte steht die Funktionseinheit, auf der der Befehl ausgeführt werden soll.

Very Long Instruction Word

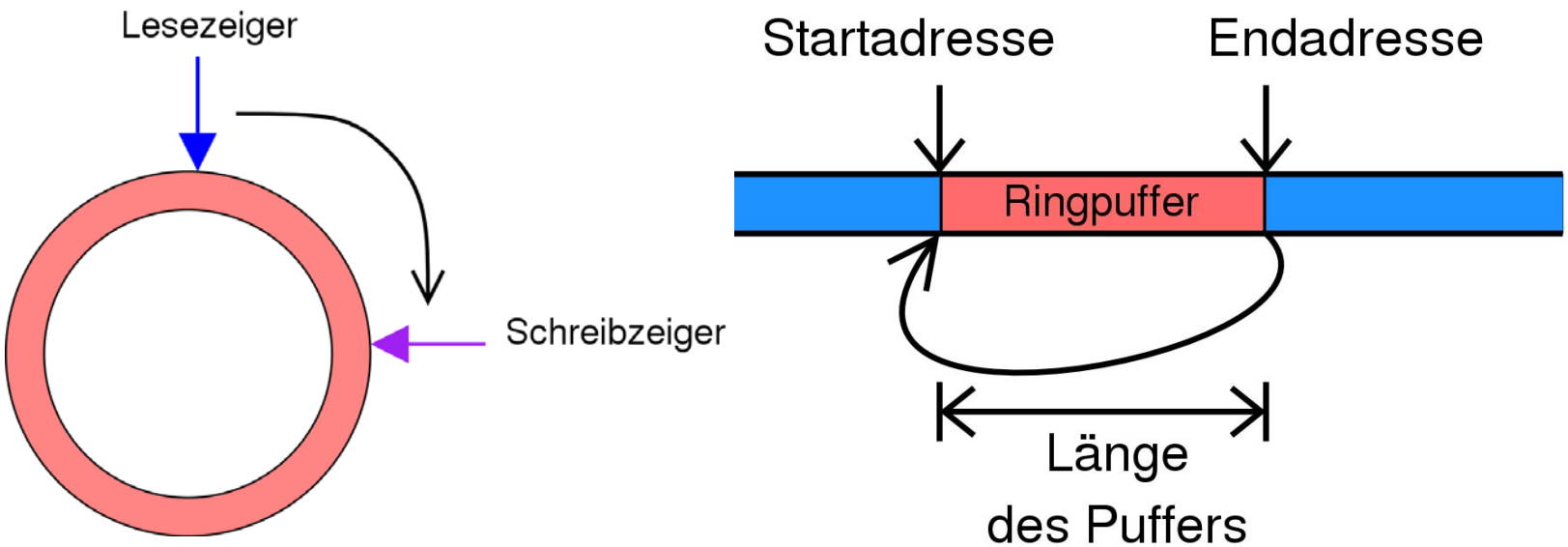
Die einzelnen Instruktion werden während der Dispatch-Phase auf die Funktionseinheiten verteilt



† NOP is not dispatched to a functional unit.

Adressierungsarten

- Registerindirekte Adressierung (*p)
- Pre- und post-modify (*++p, *p++)
- Modulo-Adressierung (zur Realisierung von Ringpuffern)
- Bit-reversed Addressing für die FFT
- Simultane Berechnung von zwei Adressen



Zahlendarstellungen in DSPs

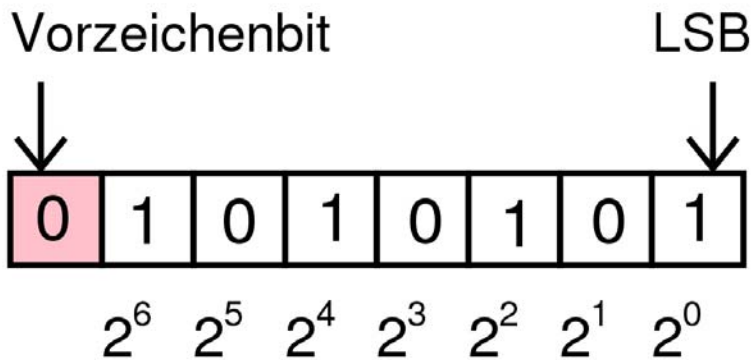
Festkommazahlen (Fixed-Point Numbers)
Gleitkommazahlen (Floating-Point Numbers)

Festkommazahlen

Wiederholung:

Im Integerformat von Mikroprozessoren können nur ganzzahlige Werte abgespeichert werden.

Sei n die Anzahl der verfügbaren Bits, dann erstreckt sich der Wertebereich bei vorzeichenbehafteten Werten von -2^{n-1} bis $2^{n-1}-1$. In 8 Bits lassen sich d.h. Werte zwischen –128 bis 127 speichern.

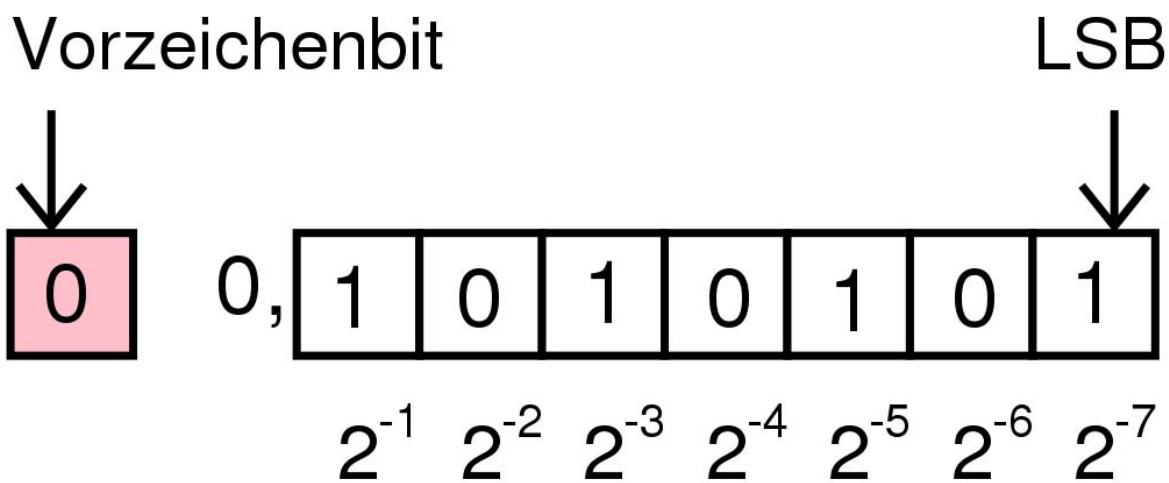


Schwierigkeiten können bei der Multiplikation entstehen: Multipliziert man zwei vorzeichenbehaftete 8-Bit Werte, so erhält man einen 15-Bit Wert.

01010101*00101010 = 110111110010

Festkommazahlen

Bei Festkommazahlen wird ein anderer Weg gegangen, da hier nur Nachkommastellen gespeichert werden. Die Menge der möglichen Werte ist durch das Interval [-1;1[gegeben. Der Abstände zwischen zwei benachbarten Werten liegen bei 2^{-n+1} und sind somit äquidistant.



Multipliziert man zwei Festkommazahlen miteinander, so erhält man wieder einen Wert, der im Interval [-1;1[liegt.

Gleitkommazahlen

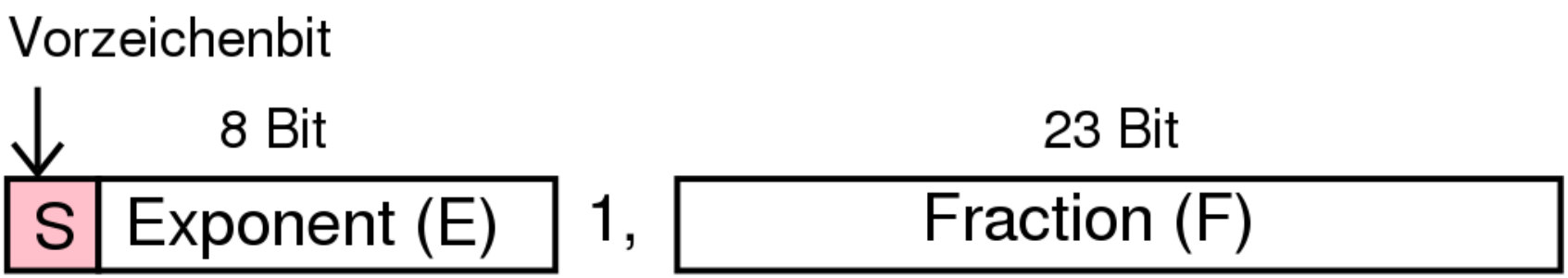
Gleitkommazahlen bestehen aus den drei Teilen

- Vorzeichen S
- Mantisse M
- und Exponent E

Der Wert einer Gleitkommazahl x errechnet sich wie folgt:

$x = (-1)^S * 2^{E-127} * 1.F$

Der darstellbare Zahlenbereich erstreckt sich von ca. 10^{-38} bis 10^{38}.



Ausblick

- Geringerer Leistungsverbrauch
- Höhere Integrationdichte
- DSP und Mikrocontroller in einem Chip (Dual-Core, z.B. für Benutzerschnittstelle)
- DSP-Multiprozessorsysteme
- DSP-Kerne (wird in eine komplexere Systemlösung monolithisch integriert)
- System-On-a-Chip (SOC) mit Schnittstellen zu PCI, USB, AC97

DSP TMS320C50

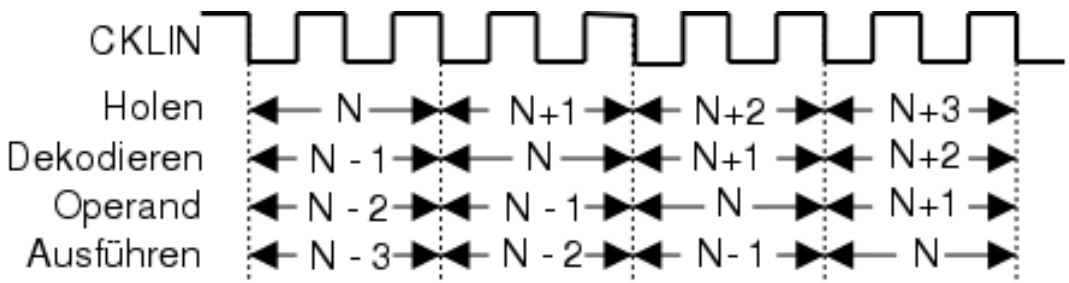
Ein Beispiel für einen Digitalen Signalprozessor: der TMS320C50 von Texas Instruments

DSP TMS320C50

- Allgemeine Eigenschaften
- Architektur
- Adressierung
- Befehlssatz
- Das DSP Starter Kit von Texas Instruments
- Beispielprogramme

Allgemeine Eigenschaften

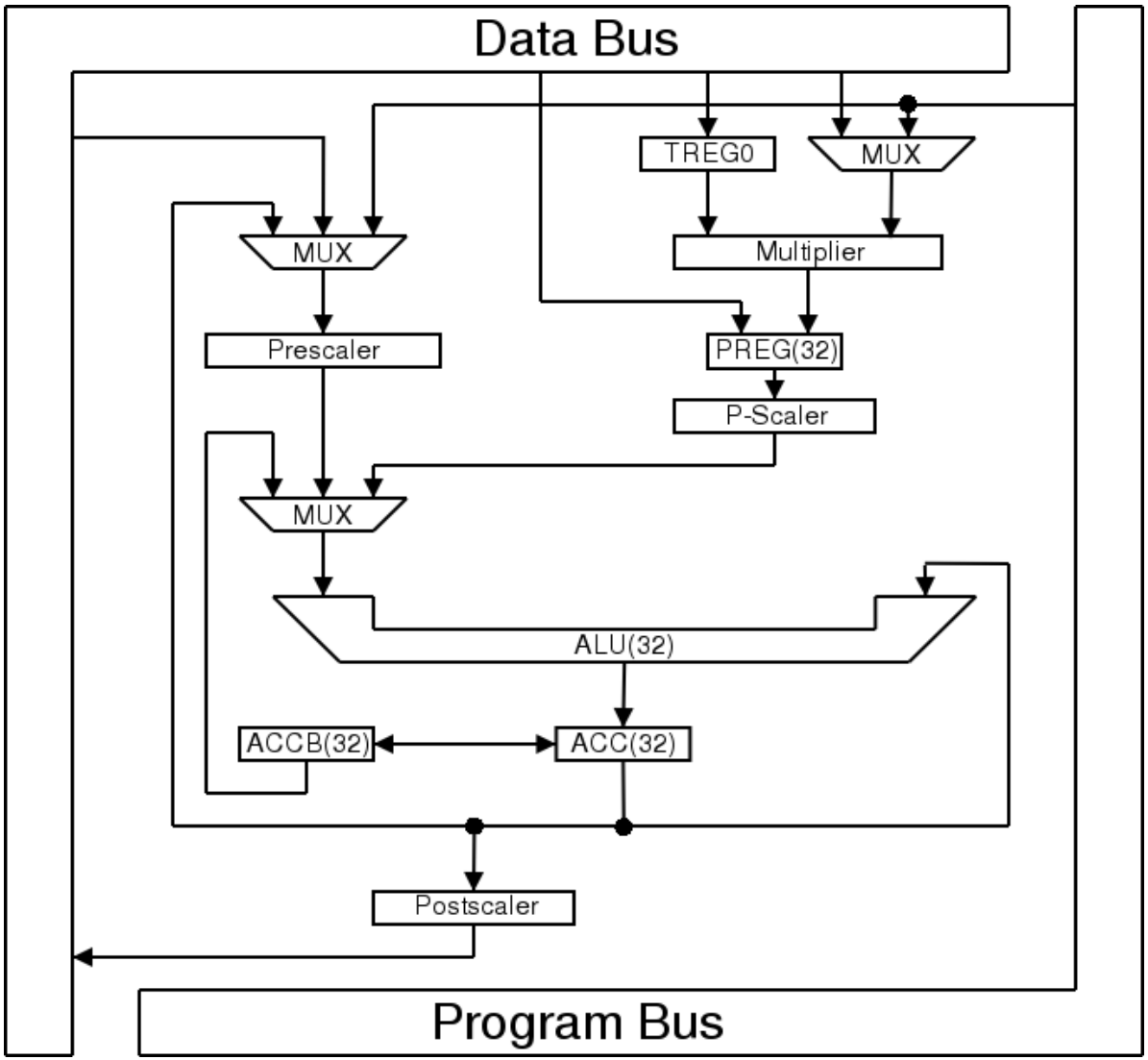
- Fixpunkt-DSP (2er Komplement-Darstellung)
- „Modifizierte" Harvard-Architektur:
  - Intern: Daten- und Programm-Bus sind getrennt
  - Nach außen: Ein Bus für Daten und Programme
- 4-Level-Pipelining auf Befehlsebene:



Architektur des TMS320C50

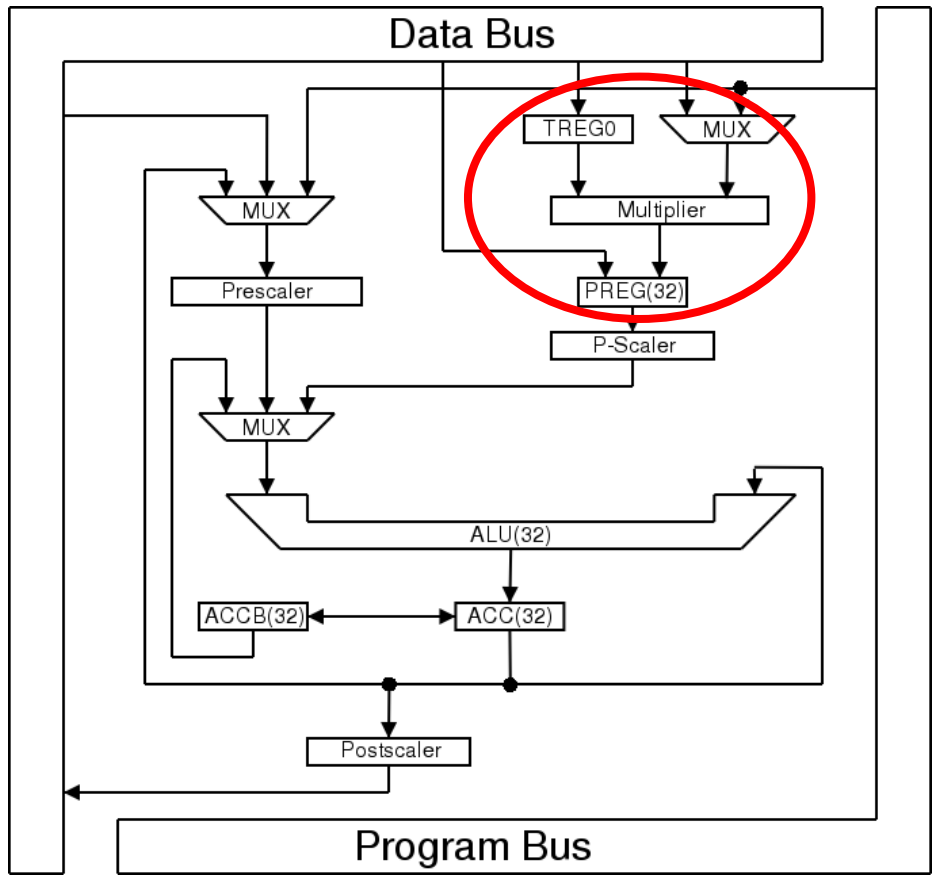
- Central processing unit (CPU):
  - Systemkontrolle
  - Central Arithmetic Logic Unit (CALU)
    - Schieber
    - Multiplizierer
    - ALU
  - Parallel Logic Unit (PLU)
- On-Chip-Speicher:
  - 1056 × 16 Bit dual-access Daten-RAM
  - 9K × 16 Bit Programm/Daten-RAM
  - 2K × 16 Bit Programm-ROM

Central Arithmetic Logic Unit (CALU)



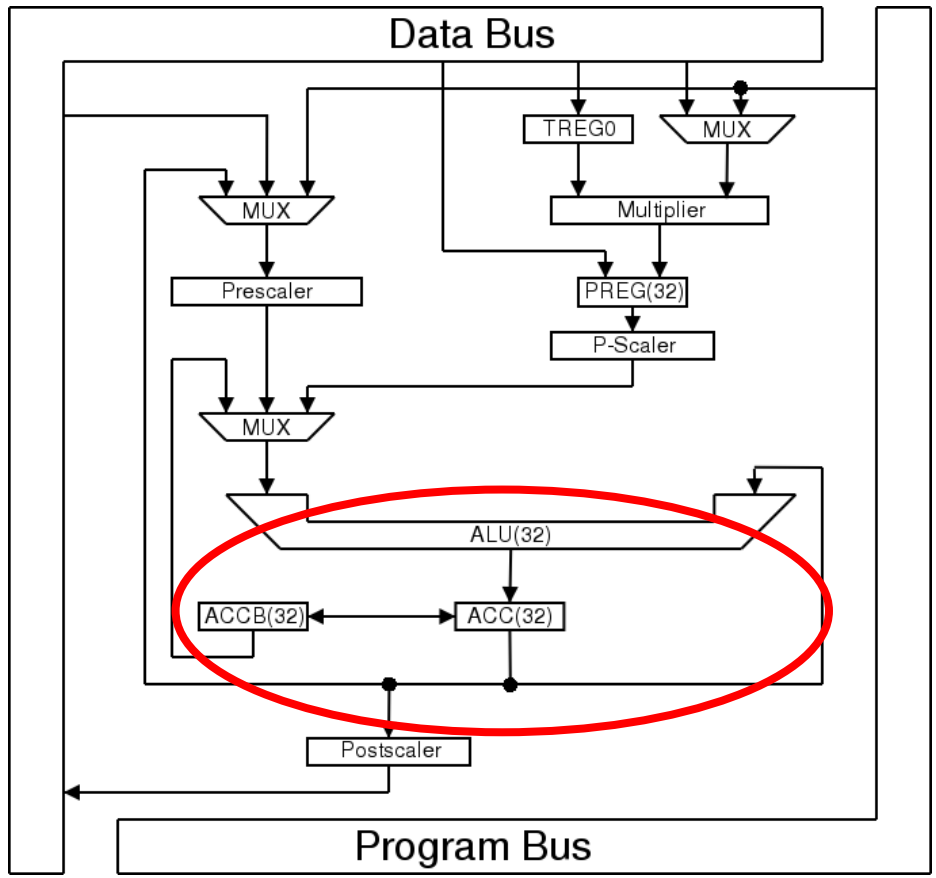
Multiplizierer

- Multipliziert zwei 16 Bit-Werte und speichert das Ergebnis in dem 32 Bit breitem Produkt-Register.
- Der erste Operand ist das TREG0 der zweite kommt entweder vom Programm- oder vom Daten-Bus
- Multiplikation wird in einem Takt ausgeführt
- Beispiele:
  - MPY #F5h
  - MPY *



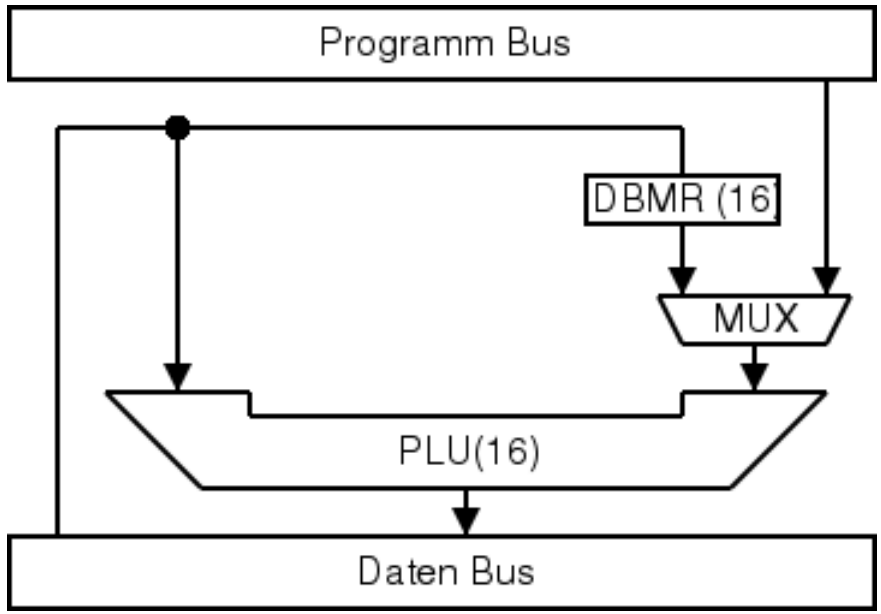
Arithmetic/Logic Unit (ALU)

- 32 Bit ALU
- Operationen:
  - Addition und Subtraktion
  - Logische Operationen wie Und-/Oder-Operationen
  - Schiebe und Rotationsbefehle
- Die meisten Operationen werden in einem Takt ausgeführt
- Beispiele:
  - ADD #20h
  - ADD 11h
  - ADD *

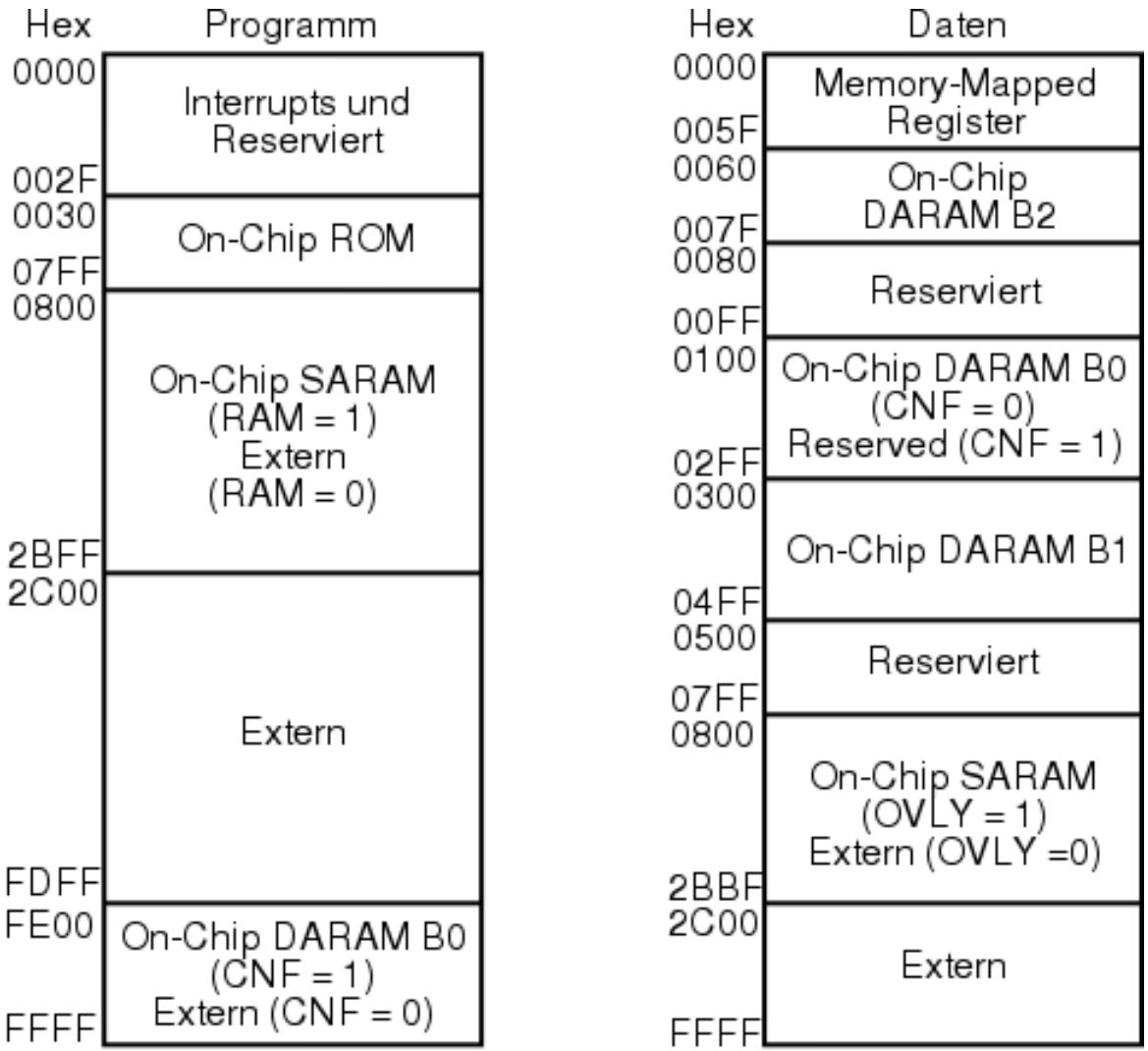


Parallel Logic Unit (PLU)

- Mit der PLU können einzelne Bits gesetzt gelöscht und getestet werden. Dabei hat man sowohl Zugriff auf die Kontroll- und Status-Bits als auch auf jede andere Stelle im Daten-Speicher.
- Beispiele:
  - APL #1011h, 33h
  - APL 33h



Adressraum

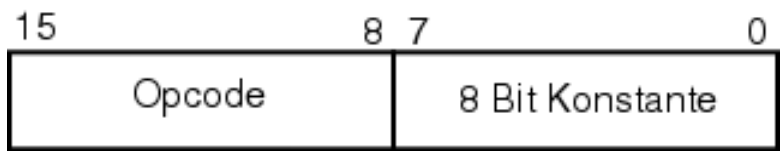

Hex
Programm
0000
Interrupts und Reserviert
002F
0030
On-Chip ROM
07FF
0800
On-Chip SARAM (RAM = 1) Extern (RAM = 0)
2BFF
2C00
Extern
FDFF
FE00
On-Chip DARAM B0 (CNF = 1) Extern (CNF = 0)
FFFF
Hex
Daten
0000
Memory-Mapped Register
005F
0060
On-Chip DARAM B2
007F
0080
Reserviert
00FF
0100
On-Chip DARAM B0 (CNF = 0) Reserved (CNF = 1)
02FF
0300
On-Chip DARAM B1
04FF
0500
Reserviert
07FF
0800
On-Chip SARAM (OVLY = 1) Extern (OVLY =0)
2BBF
2C00
Extern
FFFF

Adressierungsarten

- Short/Long Immediate Adressierung
- Direkte Adressierung
- Indirekte Adressierung
- Ringpuffer

Short/Long Immediate Adressierung

- Der Wert steht hinter dem Befehl im Programm-Speicher.
- Short:

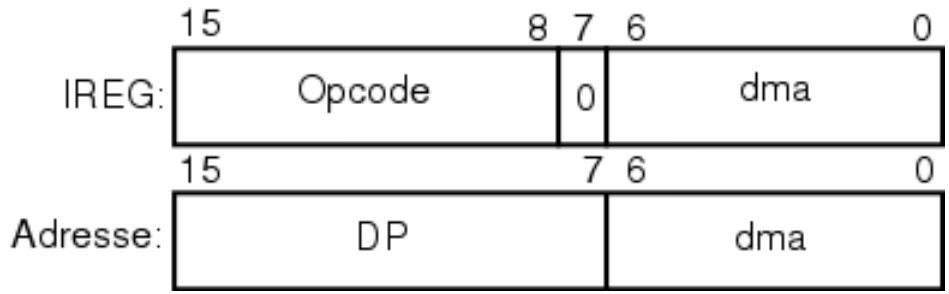


- Long:

- Beispiele:
  - ADD #15h
    short immediate Adressierung, da der Wert mit 8 Bit dargestellt werden kann.
  - ADD #FFFh
    long immediate Adressierung, da der Wert nicht mehr mit 8 Bit dargestellt werden kann.

Direkte Adressierung

- Data Memory Page Pointer (DP): 9 Bit verweisen auf eine der 512 möglichen Speicherseiten, die eine Länge von 128 Wörter haben.
- Die restlichen 7 Bit werden in der Operation angegeben.

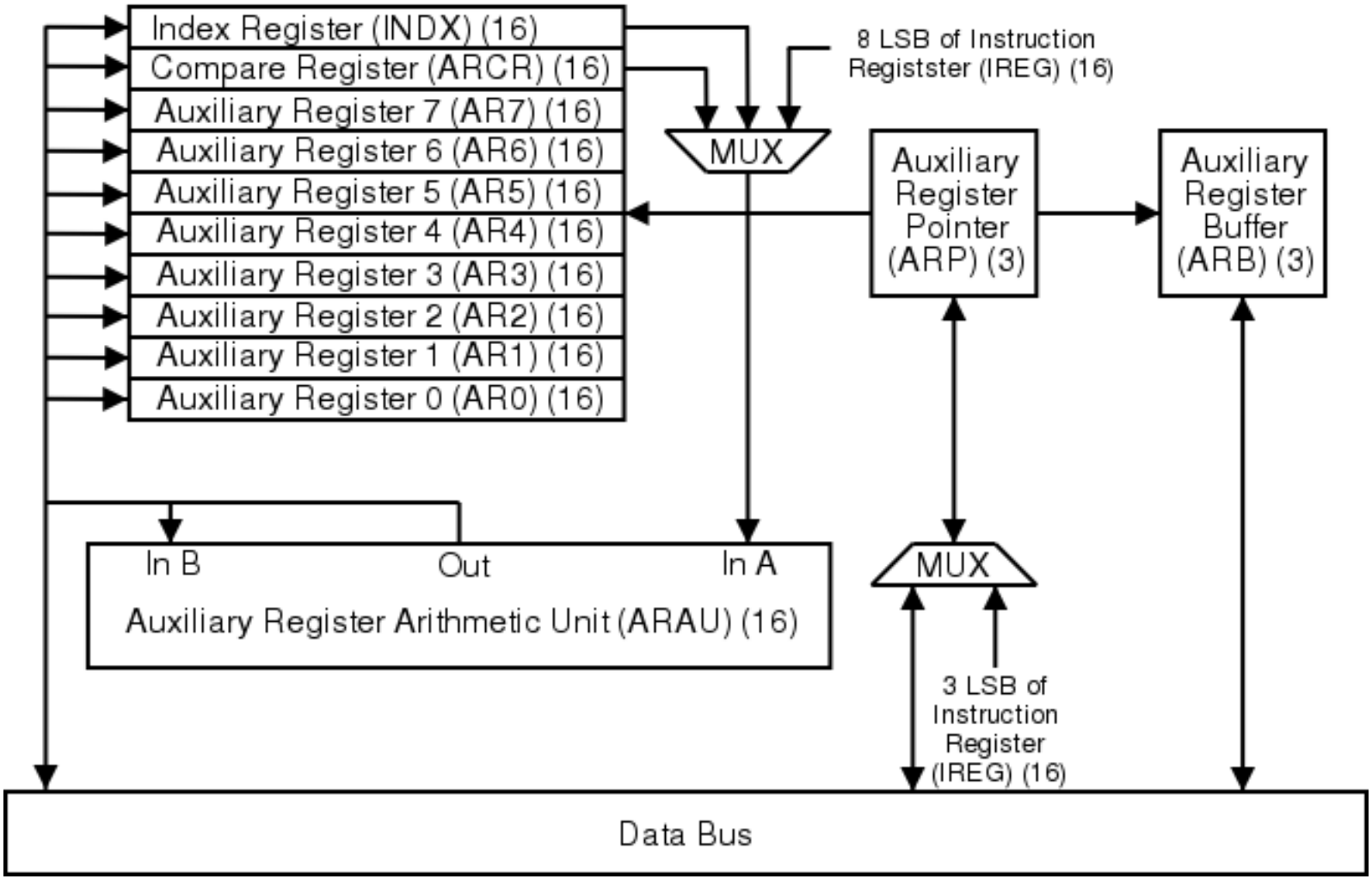


- Beispiel:
  ADD 54h ; (DP=1eh)
  Addiert den Wert von der Speicherzelle 0f54h auf den Akkumulator
  (1eh × 80h + 54h = 0f54h)

Indirekte Adressierung

- Die Indirekte Adressierung erfolgt über die acht so genannten Auxiliary Register (AR0 – AR7), die jeweils eine Breite von 16 Bit haben und so den gesamten Adressraum abdecken.
- Welches der acht Register gewählt wird, steht im 3-Bit breiten so genannten Auxiliary Register Pointer (ARP).
- Nach einem Speicherzugriff kann der Registinhalt der acht Auxiliary Register wie folgt geändert werden:
  - Um eins inkrementiert oder dekrementiert werden
  - Um den Inhalt des Indexregister (INDX) erhöht beziehungsweise erniedrigt werden. Es ist auch möglich das Indexregister mit umgekehrter Übertragsrichtung aufzuaddieren oder abzuziehen.
- Ebenso kann der Auxiliary Register Pointer mit einem neuen Wert gesetzt werden.

Indirekte Adressierung

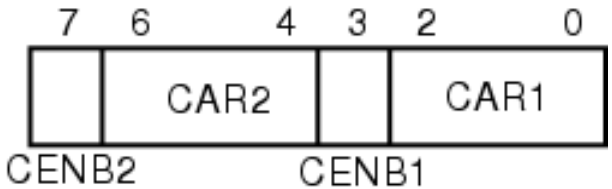


Indirekte Adressierung

- Beispiele: ARP = 2
  - ADD *
    Addiert den Speicherinhalt, auf den AR2 zeigt, auf den Akkumulator
  - ADD *+, AR5
    Addiert den Speicherinhalt, auf den AR2 zeigt, auf den Akkumulator und erhöht den Wert von AR2 um eins und setzt den ARP auf 5
  - ADD *0+, AR4
    Addiert den Speicherinhalt, auf den jetzt AR5 zeigt, auf den Akkumulator und addiert zusätzlich den Wert des Index-Register auf AR5. Der ARP zeigt nun auf AR4.
  - ADD *0-
    Addiert den Speicherinhalt, auf den AR4 zeigt, auf den Akkumulator und zieht den Indexregister-Inhalt von AR4 ab.
  - ADD *BR0+
    Addiert den jeweiligen Speicherinhalt auf den Akkumulator und erhöht den AR4 um den Wert des Indexregisters mit umgekehrten Übertragsrichtung.

Ringpuffer

- Folgende Register steuern die zwei Ringpuffer des TMS320C50:
  - CBSR1: Startadresse des ersten Ringpuffers
  - CBER1: Endadresse des ersten Ringpuffers
  - CBSR2: Startadresse des zweiten Ringpuffers
  - CBER2: Endadresse des zweiten Ringpuffers
  - CBCR: Kontrollregister für die zwei Ringpuffer:



    - CAR1: Identifiziert das Auxiliary Register für den ersten Puffer
    - CENB1: erster Ringpuffer: 0 für ausgeschaltet; 1 für eingeschaltet
    - CAR2: Identifiziert das Auxiliary Register für den zweiten Puffer
    - CENB2: zweiter Ringpuffer: 0 für ausgeschaltet; 1 für eingeschaltet

Befehlssatz des TMS320C50

- Befehle der für die ALU:
  - ADD: Addieren zu dem Akkumulator
  - ADCB: Addiert den Akkumulator-Puffer auf den Akkumulator
  - CMPL: Komplementiert den Akkumulat
  - AND / OR: Und- / Oder-Operation
  - ROL / ROR: Rotiert den Akkumulator nach links / rechts
  - ROLB / RORB: Rotiert den Akkumulator und den Akkumulatorpuffer nach links / rechts.
  - SACB: Speichert den Akkumulator im Akkumulatorpuffer
  - SBB: Subtrahiert den Akkumulatorpuffer vom Akkumulator
  - SFL / SFR: Schiebt den Akkumulator nach links / rechts.
  - SFLB / SFRB: Schiebt den Akkumulator und den Akkumulatorpuffer nach links / rechts.

Befehlssatz des TMS320C50

- Befehle für den Multiplizierer:
  - LT: Lädt das TREG0-Register mit einem Wert
  - LTA: Lädt das TREG0-Register und addiert das vorhergehende Produkt
  - MPY: Multipliziert einen Wert mit dem TREG0-Register
  - MPYA: Multipliziert wie MPY und addiert das vorhergehende Produkt
  - MPYS: Multipliziert wie MPY und subtrahiert das vorhergehende Produkt
  - MAC: Multipliziert und addiert Werte
  - SQRA: Quadriert einen Wert und addiert das vorhergehende Produkt
  - SPL: Speichert die unteren 16 Bit vom PREG
  - SPH: Speichert die oberen 16 Bit vom PREG

Befehlssatz des TMS320C50

- Befehle für die Auxiliary Register:
  - ADRK: Addiert einen short immediate Wert zu dem aktuellen AR
  - LAR: Lädt einen Wert in ein AR
  - MAR: Ändert das aktuelle AR und den ARP
  - SAR: Speichert ein AR in den Daten-Speicher
  - SBRK: Subtrahiert einen short immediate Wert zu dem aktuellen AR
- Befehle für die PLU:
  - APL: Und-Operation zwischen dem DBMR oder einer Konstante und einem Wert im Daten-Speicher
  - OPL: Oder-Operation zwischen dem DBMR oder einer Konstante und einem Wert im Daten-Speicher
  - SPLK: Speichert einen long immediate Wert in den Daten-Speicher
  - XPL: Exklusiv-Oder-Operation zwischen dem DBMR oder einer Konstante und einem Wert im Daten-Speicher

Befehlssatz des TMS320C50

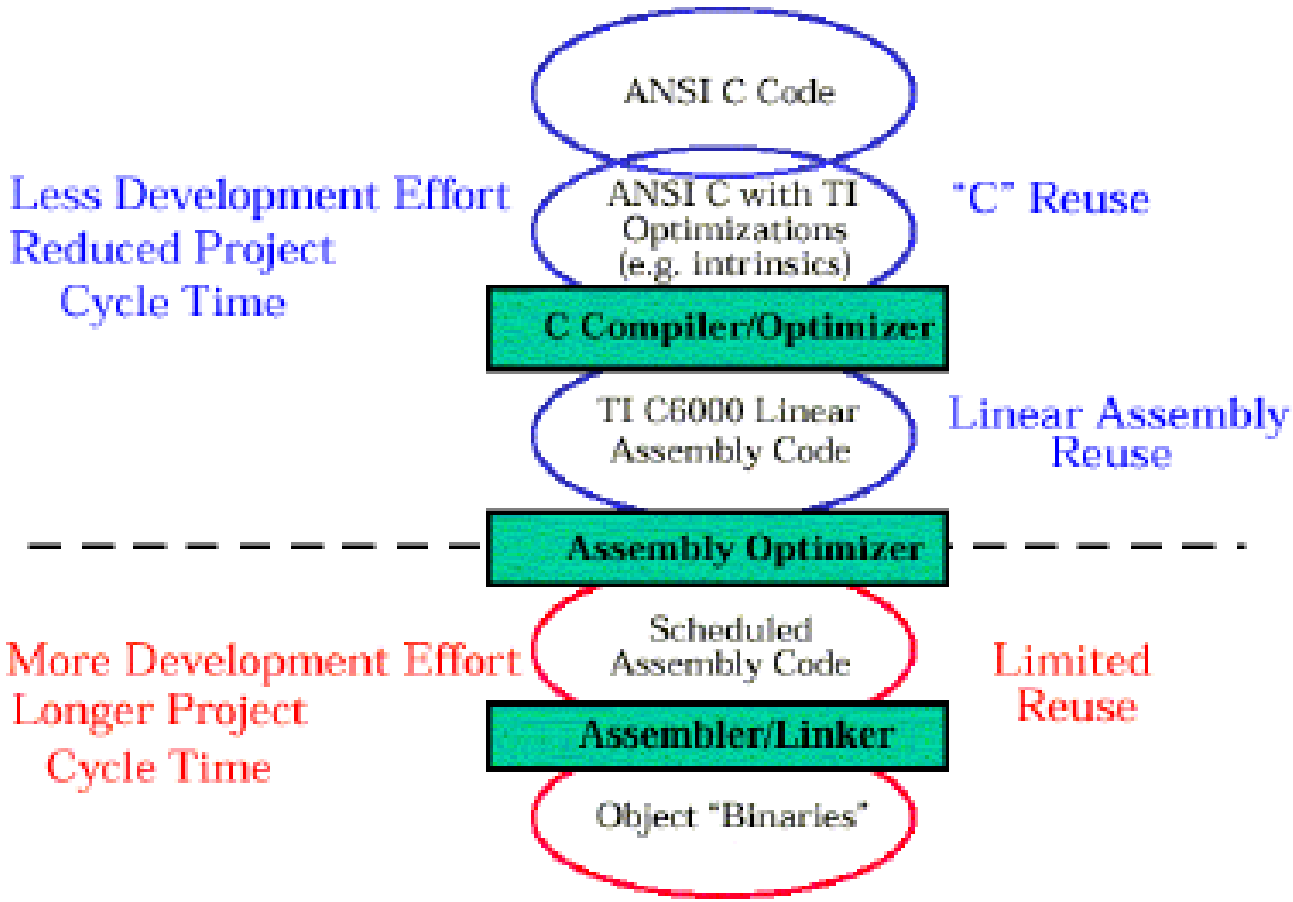
- Speicherzugriffsbefehle
  - BLDD: Kopiert einen Block innerhalb des Daten-Speichers
  - LMMR: Lädt ein memory-mapped Register
- Kontrollbefehle:
  - B: unbedingter Sprung
  - BCND: bedingter Sprung
  - CALL: Unterprogramm-Aufruf
  - CC: bedingter Unterprogramm-Aufruf
  - RET: Unterprogramm-Rücksprung
  - RETC: bedingter Unterprogramm-Rücksprung
- Wiederholungsanweisungen:
  - RPT: Wiederhole nächste Anweisung
  - RPTB: Wiederhole einen bestimmten Block

Beispielprogramm

Berechne Skalarprodukt

```
          .mmregs
          .ds       0f00h
result    .word     0
blank     .word     0,0,0,0,0,0
costs     .word     10,20,30,40,50,60,70
num       .word     5,10,3,2,0,34,17
          .ps       0a00h
again     ldp       #blank              ;Lade data point register
          lar       ar0,#costs          ;lade Adresse des arrays
          zap                           ;lösche akku
          sacl      result,0            ;lösche Ergebis
          rpt       #7                  ;setze wiederholungszähler
          mac       num,*+              ;multipliziere und addiere
          sacl      result,0            ;schreibe Ergebnis
          b         again
    .end
```

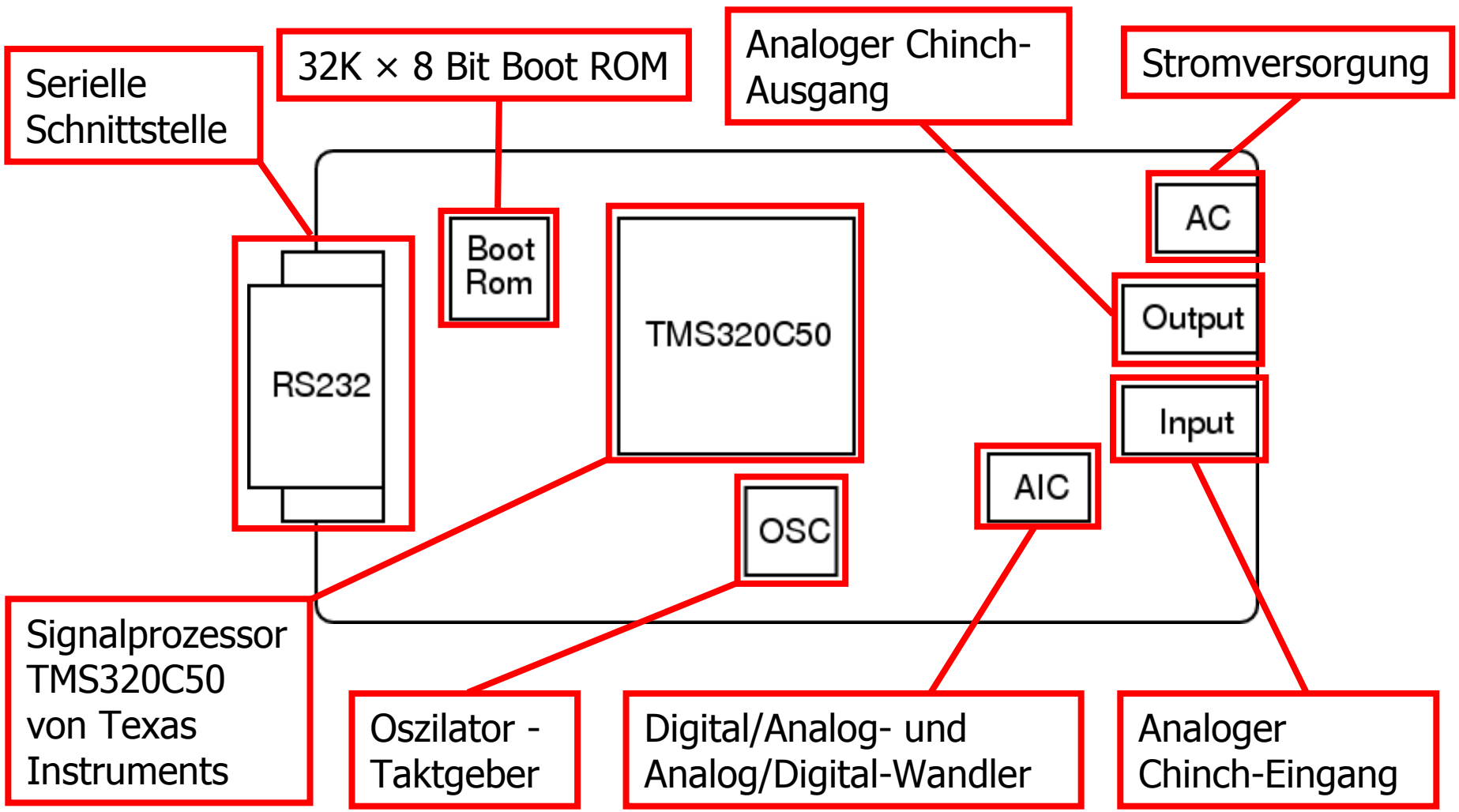
Software Entwicklung



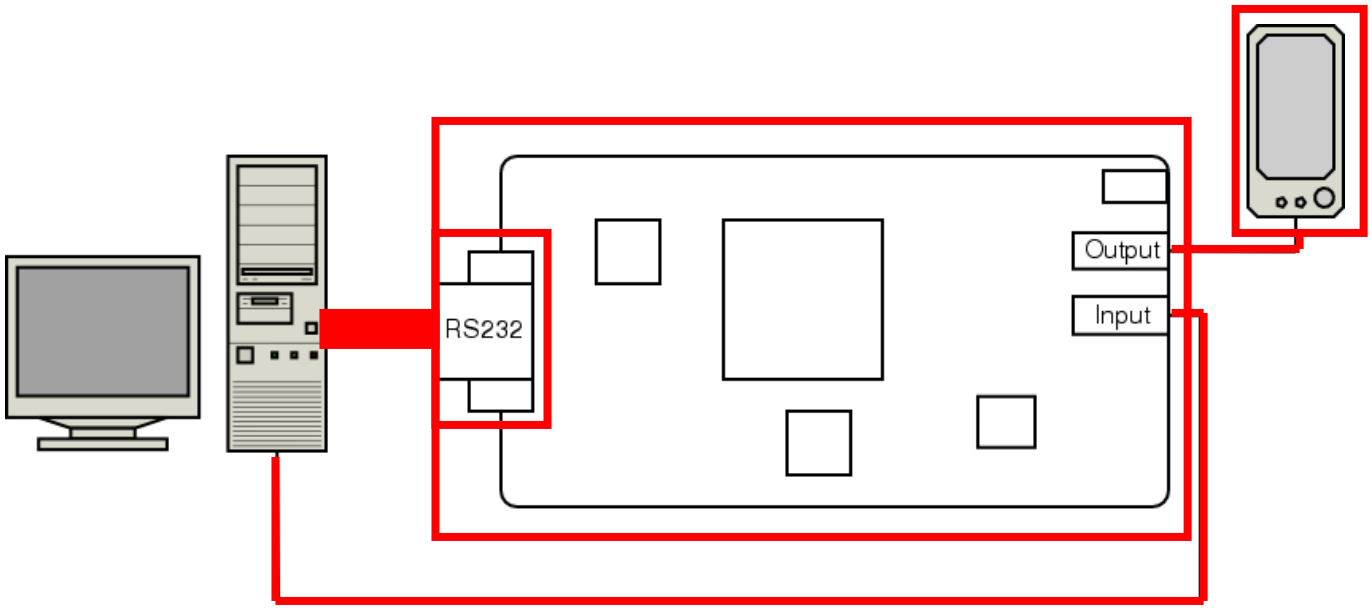
Das DSP Starter Kit (DSK)

- Software:
  - Der Assembler „dsk5a“
  - Der Debugger „dsk5d“
  - Der Loader „dsk5l“
  - Beispielprogramme
- Hardware: Das DSK-Board

Das DSK-Board



Konfiguration für die Beispiele



Dem DSP wird über die serielle Schnittstelle Programm und eventuell Daten in den On-Chip-Speicher geschrieben. Der DSP beginnt mit der Abarbeitung des Programms.

Am Computer werden per MP3-Player Musiksignale erzeugt, die über den Line-Out des Computer in den Eingang des DSK-Boards gelangen.

Die eingehenden Signale werden vom DSP verarbeitet.

Die gerechneten Signale werden über den Lautsprecher ausgegeben.

Beispielprogramme

- MACD-Beispiel
- Tiefpass, Hochpass
- Bandpass, Bandstop
- Oscilloskop
- Dialer
- Hall (Echokammer)
- Menuett